

Darstellungs- und Interaktionstechniken zur effizienten Nutzung grafischer Oberflächen durch Blinde und Sehbehinderte

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart
zur Erlangung der Würde eines Doktors der Naturwissenschaften
(Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Christiane Taras

aus Rudolstadt

Hauptberichter:	Prof. Dr. rer. nat. Thomas Ertl
Mitberichter:	Prof. Dr. rer. nat. habil. Gerhard Weber
Tag der mündlichen Prüfung:	08.04.2011

Institut für Visualisierung und Interaktive Systeme der Universität Stuttgart

2011

Für meinen Mann Enrico.

Kurzfassung

Personal-Computer sind heute eines der wichtigsten Arbeits-, Kommunikations- und Lernmittel. Moderne Computer bieten hochwertige grafische Darstellungen, die die tägliche Arbeit erleichtern oder durch ihre Ästhetik schlicht Wohlbefinden und Freude hervorrufen sollen. Auch für blinde und sehbehinderte Menschen ist der Computer ein wichtiges Werkzeug. Er bringt Unabhängigkeit und verbessert gleichzeitig die Integration in die Gesellschaft. Durch den zunehmenden Einsatz digitaler Dokumente wird die Zugänglichkeit von Informationen stetig erhöht. Technische Hilfsmittel wie Vergrößerungssoftware oder Screenreader bieten Möglichkeiten, die Ausgabe der Dokumente an die eigenen Bedürfnisse anzupassen. Bei gedruckten oder handschriftlichen Dokumenten ist dies nicht ohne weiteres möglich.

Allerdings weisen aktuelle Technologien noch einige Defizite auf. So werden grafische Benutzungsschnittstellen für Blinde größtenteils durch textuelle Informationen mit semantischen Annotationen (wie Nennung des Typs und des Aktivierungszustandes eines Elements) präsentiert. Informationen über die grafische Darstellung an sich werden kaum bereitgestellt. Dabei sind diese auch für Blinde sehr interessant, da grafische Eigenschaften von Normalsichtigen häufig als alleinige Informationsträger oder auch als Kommunikationshilfe genutzt werden. Sehbehinderten, die noch die visuelle Ausgabe nutzen, werden zwar grafische Informationen präsentiert, aber die Darstellungsweise ist selten optimal für ein effizientes Arbeiten. So ist beispielsweise bei der Bearbeitung von Text häufig horizontales Scrollen nötig oder wichtige Bereiche der Bildschirmausgabe sind nicht immer sichtbar.

Diese Arbeit trägt dazu bei, den Zugang Blinder und Sehbehinderter zu grafischen Darstellungen weiter zu erleichtern. Dazu wurde untersucht, wie diese mit Hilfe neuer Technologien und Herangehensweisen besser aufbereitet und von den Betroffenen interaktiv genutzt und auch selbst produziert werden können. Eine wesentliche Erkenntnis liegt in der Ähnlichkeit der grundlegenden Fragestellung bei der Präsentation von grafischen Oberflächen für Sehbehinderte und Blinde, die ein grafisch-taktil Display nutzen. Basierend darauf wurde ein Framework entwickelt, mit dem sowohl taktile Darstellungen realisiert werden können wie auch ein neuartiges Konzept für vergrößerte Bildschirmausgaben für stärker sehbehinderte Computernutzer. Um auch die stetig wachsende Gruppe der zumeist altersbedingt von leichteren Seheinschränkungen Betroffenen besser zu unterstützen, wurde ein weiteres Vergrößerungskonzept samt einer prototypischen Umsetzung erarbeitet, welches nur minimale Änderungen an der Darstellung eines Programms umfasst und somit den Einarbeitungsaufwand und die damit verbundene Hemmschwelle minimiert. Zur Förderung der Kommunikation zwischen Normalsichtigen, die sich in ihren Beschreibungen häufig auf Farben beziehen, Sehbehinderten und Blinden wurden Konzepte zum Umgang mit Farben und farbigen Grafiken erforscht und Umsetzungen für monochrome, taktile Ausgabegeräte implementiert. Da die Voraussetzung für angepasste Darstellungen von GUIs und Grafiken deren zugängliche Gestaltung ist, wurden zudem Konzepte und Umsetzungen zur Unterstützung von Entwicklern und Designern bei der zugänglichen Gestaltung von GUIs und der Verbreitung von Standards zur Gestaltung zugänglicher Grafiken im SVG-Format erarbeitet.

Abstract

Personal computers become more and more important in people's lives. Digital information and user interfaces with high-quality graphical representations facilitate education as well as daily work and entertainment. Also for blind and visually impaired persons, the computer is important to improve their quality of live. It gives them independence and at the same time enhances their integration into society. Due to the increasing use of digital documents, the accessibility of information is raised steadily. By using assistive technologies such as magnification software and screen readers, the presentation of digital documents can be adapted to each user's needs. For printed or handwritten documents, this is not easily possible.

However, current technologies still need improvements. Screen readers, for example, present graphical user interfaces only by textual information including the visible textual content as well as some semantic information like the type of a user interface element or its current state. Information on the graphical representation itself is provided only rarely, although this is also useful for the blind. Visually enabled people often use graphical information like color or positioning as means of information or communication. Indeed, for users of magnification software graphical information is available. However, the way of presenting graphical user interfaces often not assists users in an optimal way. For instance, while working with documents to read and edit texts horizontal scrolling is often necessary which significantly stresses the user. Further, through the enlargement of the screen content to multiple screen sizes important parts of the graphical user interfaces are often not visible to the user. Therefore, there is a high risk of missing important information.

This thesis contributes to improving the access of blind and visually impaired people to graphical representations. Therefore, state-of-the-art technologies and new methods for improving the adaption of graphical user interfaces and documents as well as new approaches for interactive use and creation of graphics by blind and visually impaired persons themselves were examined. It could be shown that the requirements for presenting graphics to visually impaired persons and blind ones, who use graphical-tactile displays, are basically similar. Based on this finding a framework was developed which can be used to realize tactile representations as well as a newly developed screen magnification concept for partially sighted computer users. With respect to the steadily growing group of people with weaker and mostly age-related visual impairment another concept and prototype for screen magnification was developed which requires only minimal modification of a GUI's presentation, thus minimizing the effort for familiarizing with the magnification technique as well as the inhibition threshold for using screen magnification. To promote communication between visually enabled people, who often refer to colors, visually impaired and blind ones strategies for presenting colors and adapting colored graphics were investigated and implemented for monochrome, tactile output devices. As useful adaptations of graphical representations can only be generated if they are as accessible as possible, concepts and implementations to support developers and designers in the accessible design of GUIs and to propagate guidelines for accessible design of SVG graphics.

Danksagung

Wie die Widmung bereits vermuten lässt, geht ein besonderer Dank an meinen Mann Enrico dafür, dass er mich überzeugte, eine Promotion zu wagen, als stetig antreibende Kraft hinter mir stand und über die Jahre auch schwierige Phasen erduldet.

Auch meinen Eltern, meinen Großeltern und meiner Tante möchte ich danken für die Unterstützung und Anregung, die sie mir seit meiner Kindheit boten, und den Bildungsweg, den sie mir ermöglichten.

Meinem Doktorvater Thomas Ertl danke ich für die Aufnahme in das Institut für Visualisierung und Interaktive Systeme (VIS) und die hervorragende Forschungsumgebung, die er dort geschaffen hat.

Weiterhin danke ich ihm sowie Martin Rotard, Waltraud Schweikhardt und Rul Gunzenhäuser für die Einführung in den Themenbereich der Unterstützung sensorisch Behinderter und die kritischen Diskussionen und Hinweise, die mir bei der Themenfindung für meine Forschungsarbeit sehr behilflich waren.

Mein Dank geht auch an das Bundesministerium für Wirtschaft und Technologie und seine Vertreter für die Unterstützung und Förderung des Projektes „HyperBraille“, dass mich über einen Großteil meiner Promotionsphase begleitete und auf Grund dessen ich viele der in dieser Dissertation beschriebenen Konzepte entwickeln und erproben konnte. Auch den Projektpartnern, insbesondere Oliver Nadig, Ursula Weber, Friedrich Lüthi, Dirk Kochanek und Gerhard Weber, danke ich sehr für die Inspirationen und die erfolgreiche Zusammenarbeit.

Zuletzt möchte ich allen Mitarbeitern des VIS und VISUS, die ich über die Jahre kennenlernen durfte, insbesondere meinen Bürokollegen Martin Rotard, Martin Falk und Michael Raschke sowie meinem zeitweiligen Teamleiter Thomas Schlegel, und den Studenten, die meine Arbeit unterstützten, für die gute Gemeinschaft und Zusammenarbeit und die anregenden Diskussionen zu verschiedensten Themengebieten danken. Es war eine schöne Zeit!

Inhalt

1	Einleitung	13
1.1	Problemstellung.....	14
1.2	Struktur und Beiträge dieser Arbeit	15
2	Grundlagen der Arbeit und Stand der Technik	17
2.1	Hilfsmittel für blinde und sehbehinderte Computernutzer	17
2.1.1	Screenreader	18
2.1.2	Taktile Grafiken	20
2.1.3	Grafisch-taktile Displays	22
2.1.4	Bildschirmausgabe für Sehbehinderte	25
2.1.5	Zusammenfassung.....	30
2.2	Aufbau und Zugänglichkeit grafischer Oberflächen	31
2.2.1	Bestandteile und Darstellung grafischer Oberflächen	31
2.2.2	Zugang zu grafischen Oberflächen.....	35
2.2.3	Semantische Informationen in grafischen Oberflächen	42
2.2.4	Zusammenfassung.....	47
2.3	Richtlinien zur zugänglichen Gestaltung von grafischen Oberflächen.....	48
2.4	Zusammenfassung	50
3	Angepasste Darstellungen und erweiterte Interaktionstechniken	53
3.1	Gleichartigkeit angepasster Darstellungen für beide Zielgruppen	54
3.1.1	Gemeinsame Grundfragestellung	54
3.1.2	Gemeinsamkeiten in konkreten Anforderungen	55
3.1.3	Beachtung von verschiedenen Arbeitssituationen und Arbeitsbereichen	69
3.1.4	Erweiterte Interaktionsmöglichkeiten	71
3.1.5	Fazit	72
3.2	Konzepte zu Darstellungen und Interaktionen.....	73
3.2.1	Strategien zur Erzeugung angepasster Darstellungen	73
3.2.2	Erleichterung der Erfassung aller interessanten Informationen	75
3.2.3	Umgang mit Farben.....	90
3.2.4	Gesamtkonzepte zur Erschließung grafischer Benutzungsoberflächen	109
3.2.5	Kooperation zwischen Nutzern grafisch-taktile Displays und Sehenden	119
3.3	Umsetzungsbeispiele.....	123
3.3.1	Ein Grafikeditor für Blinde zur Erstellung farbiger Grafiken	123
3.3.2	Rendering-Framework für grafisch-taktile Displays.....	146
3.3.3	HyperReader-Magnifier – Vergrößerung mit dem HyperBraille-Konzept	172
3.3.4	FirefoxZoom – Eine Umsetzung der partiellen Vergrößerung.....	174
3.4	Zusammenfassung	178

4	Förderung der zugänglichen Gestaltung.....	179
4.1	Darstellung der Problematik.....	179
4.2	Lösungskonzepte	186
4.2.1	Alternative Darstellungen zu Rastergrafiken	186
4.2.2	Zugängliche Lehrmaterialien im Web	190
4.2.3	Zweckgebundene Labels und Gruppenvorschriften für zugänglichere GUIs...	193
4.2.4	Struktur und Annotation belohnen – mehr Nutzen für alle	201
4.3	Nutzung von SVG-Zugänglichkeitsattributen zur Webseitengestaltung.....	203
4.4	Allgemeine Richtlinien zur Förderung zugänglicher Gestaltung	208
4.5	Fazit.....	211
5	Ergebnisse der Arbeit und Ausblick	213
	Anhang.....	219
Anhang A	Fragebogen zur visuellen Darstellung der DFB-Farben	219
A.1	Geprüfte Farben.....	219
A.2	Gestaltung des Fragebogens.....	220
A.3	Ergebnisse der Umfrage.....	222
Anhang B	Ein Lernprogramm für Blindenschrift	225
Anhang C	Sonstige Verzeichnisse	228
C.1	Abkürzungsverzeichnis.....	228
C.2	Abbildungsverzeichnis	229
C.3	Tabellenverzeichnis.....	232
C.4	Verzeichnis der Listings.....	232
	Literaturverzeichnis	235

1 Einleitung

Grafische Darstellungen sind in der heutigen, digitalen Welt allgegenwärtig. Grafiken werden in Dokumenten zur Erläuterung von Sachverhalten eingesetzt oder zur Verschönerung von Darstellungen, wie es sehr häufig bei Webseiten und Präsentationen der Fall ist. Ein Endbenutzer-Programm ohne grafische Benutzungsoberfläche ist heute kaum mehr denkbar. Diese bieten durch strukturierte Menüs einen schnellen Überblick über die unterstützten Funktionalitäten. Die wichtigsten Funktionen sind meist in Toolbars (auch Werkzeugleisten genannt) und neuerdings in Ribbons (auch Multifunktionsleisten genannt) wie in Microsoft Word durch Icons übersichtlich dargestellt. Wichtige Bereiche werden meist farblich markiert und optische Gruppierungen werden geschaffen durch bestimmte Anordnungen, gleichartige Gestaltung oder Umrandungen. Farbverläufe und Schattierungen erzeugen ein angenehmes Erscheinungsbild. All diese Maßnahmen werden getroffen, um normalsichtigen und damit visuell geprägten Menschen die Arbeit am Computer zu erleichtern, sie so gut es geht zu unterstützen oder bei ihnen sogar Freude durch die Arbeit mit einem bestimmten Programm hervorzurufen.

Für blinde und sehbehinderte Menschen können aber gerade diese grafischen Unterstützungen zu unüberwindbaren Hürden werden. Dabei ist die digitale Welt für behinderte Menschen besonders wichtig, da nur digitale Dokumente und Darstellungen die Möglichkeit bieten, in angemessener Zeit an die verschiedensten Bedürfnisse wie unterschiedliche Schriftarten, Farbgebung und Ausgabeformen, angepasst zu werden. Ein gedrucktes Buch oder Formular beispielsweise kann für einen Blinden niemals so schnell zugänglich gemacht werden wie ein digitales. Dabei wäre es natürlich ideal, wenn die Anpassung der Darstellung voll automatisch erfolgen könnte und nicht manuell nachgebessert werden müsste. Dies brächte behinderten Menschen in der digitalen Welt absolute Unabhängigkeit und würde gleichzeitig die Integration in die Gesellschaft verbessern. Ungeachtet ihrer Behinderung könnten sie völlig selbständig Einkäufe erledigen, Anträge bei Behörden stellen oder einfach mit anderen Menschen weltweit kommunizieren.

Nun ist es nicht so, dass Blinde und Sehbehinderte derzeit von der digitalen Welt ausgeschlossen sind. Ganz im Gegenteil, sie gehören sogar zu den aktivsten Gruppen im Internet. Sie tauschen Informationen aus, verabreden sich, lesen Online-Zeitungen oder spielen sogar Audio-Computer-Spiele, wie sie beispielsweise auf [Cre10], [Kit10] und [DPSS10] angeboten werden. Und wer einmal die Chance hat, einem geübten Blinden bei der Computerarbeit zuzuschauen, kommt meist ins Staunen, wie rasch alles gehen kann. Mit einem Screenreader (Bildschirmvorleseprogramm), der den Bildschirminhalt analysiert und vorliest oder in Brailleschrift ausgibt, und einem guten Gedächtnis für Tastaturkürzel haben Blinde auch heute schon einen relativ guten Zugang zur digitalen Welt. Auch Sehbehinderte werden gut durch Sprachausgabe und Bildschirmvergrößerungssoftware unterstützt. Entgegen ihres Namens vergrößert diese nicht nur die Bildschirmausgabe, sondern kann auch Farbeinstellungen ändern oder wichtige Stellen hervorheben. Der Zugang blinder und sehbehinderter Menschen zu Computerprogrammen mit grafischen Benutzungsoberflächen und digitalen

1 Einleitung

Dokumenten wird durch die Entwicklung dieser Hilfsmittel enorm erleichtert oder gar erst ermöglicht. Aber immer noch gibt es unzählige Barrieren. Und je aufwändiger die grafische Gestaltung von Oberflächen und Dokumenten wird, desto größer werden sie.

Die vorliegende Arbeit zeigt Möglichkeiten auf, diese Barrieren mit Hilfe neuer Konzepte und Technologien zu mindern, um Blinde und Sehbehinderte noch besser in die digitale Welt zu integrieren. Dabei steht an oberster Stelle das Ziel, die Zusammenarbeit zwischen Normalsichtigen, Sehbehinderten und Blinden zu verbessern, um so letztlich eine gleichberechtigte Teilnahme an der digitalen Welt zu ermöglichen.

Ein großer Teil der in dieser Arbeit beschriebenen Konzepte und Umsetzungen entstand im Rahmen des Projektes „HyperBraille“ [hyp10]. Inhalt dieses Projektes war die Entwicklung eines grafischen Screenreaders, genannt HyperReader, mit Ansteuerung eines grafisch-taktilen Displays, dessen Prototyp als Nachfolger des BrailleDis 9000 ebenfalls im Rahmen von HyperBraille entwickelt wurde.

1.1 Problemstellung

Aufgrund der stark eingeschränkten bis zu nicht vorhandenen Sehfähigkeit sehbehinderter und blinder Computernutzer können diese die grafische Bildschirmausgabe nicht in der gleichen Weise erfassen wie Normalsichtige, für die diese Darstellung entworfen wurde. Sehbehinderte und Blinde benötigen speziell für sie angepasste Darstellungen, um effizient arbeiten zu können, sei es eine leichter erkennbare visuelle Ausgabe, eine taktile oder eine auditive Ausgabe. Auch entsprechende Interaktionsmöglichkeiten, wie beispielsweise das Unterbrechen der auditiven Ausgabe und erweiterte Navigationsfunktionen, müssen angeboten werden. Dabei soll es aber möglich sein, dass Blinde und Sehbehinderte mit den gleichen Softwareprodukten arbeiten und die gleichen Dokumente lesen, wie Normalsichtige. Nur so kann ihnen der uneingeschränkte Zugriff auf alle Informationen der digitalen Welt und Funktionen moderner Software ermöglicht werden, die auch Normalsichtigen zur Verfügung stehen. Dies erleichtert auch die Integration in normale Arbeitsumgebungen und die Teilnahme an sozialen Aktivitäten in der digitalen Welt. Zudem treten Sehbehinderungen oder Blindheit durch Erkrankungen oder Unfälle häufig erst im Erwachsenenalter auf. Betroffene sind dann den Umgang mit normalen oder ihren Arbeitsbereich betreffende Anwendungen schon gewohnt und wollen sie auch weiter nutzen. Eine Umstellung auf spezielle Anwendungen wäre neben der Änderung der Lebenssituation eine zusätzliche Belastung. Es sollen also möglichst keine speziellen Anwendungen entwickelt oder spezielle Dokumente erstellt werden, sondern die Darstellung gewöhnlicher Anwendungen für eine effiziente Arbeitsweise angepasst werden können.

Diese Anpassungen sollen natürlich nicht für jede Anwendung und jedes Dokument individuell erfolgen müssen. Vielmehr soll der Anpassungsprozess möglichst automatisiert und unabhängig vom Ausgangsmaterial erfolgen können. Um dies zu gewährleisten, müssen Anwendungen und Dokumente automatisch analysierbare Strukturen aufweisen, semantische

Informationen bereitstellen und den Zugriff über allgemeine Schnittstellen ermöglichen. Ist dies gegeben, können Hilfsprogramme wie Screenreader die gewünschten Darstellungen entsprechend allgemeiner Regeln und spezieller Einstellungen des Benutzers erzeugen. Ohne solch spezielle, zugängliche Gestaltung von Dokumenten und Anwendungen können angepasste Darstellungen und entsprechende Interaktionsmöglichkeiten nur in begrenztem Maße oder nur mit sehr hohem Aufwand bereitgestellt werden.

Aus diesem Grund gibt es seit einiger Zeit eine wachsende Anzahl an Richtlinien, Vorschriften und Verordnungen zur zugänglichen Gestaltung. Allerdings ist deren Auswirkung bisher sehr begrenzt. Dies liegt zum einen an der Unwissenheit der Mehrzahl von GUI- und Dokument-Erstellern über die Problematik der Zugänglichkeit und der mangelnden Unterstützung in Tools. Zum anderen bleibt für den normalen Benutzer die Mehrarbeit, die zugängliche Gestaltung meist erfordert, häufig ohne erkennbaren Nutzen. Um eine wirklich gleichberechtigte Teilnahme Normalsichtiger, Sehbehinderter und Blinder an der digitalen Welt zu realisieren, müssen auch hier Verbesserungen erzielt werden. Vor allem in Anbetracht von Entwicklungen wie dem Web 2.0, die die einfache Beteiligung von jedermann an der Gestaltung von Informationsquellen fördern, sollte nicht erwartet werden, dass jeder Autor von Dokumenten oder Anwendungen ein Experte in zugänglicher Gestaltung ist. Deshalb beschäftigt sich diese Arbeit neben Darstellungs- und Interaktionstechniken für Blinde und Sehbehinderte auch mit der Förderung zugänglicher Gestaltung.

1.2 Struktur und Beiträge dieser Arbeit

Das folgende Kapitel „Grundlagen der Arbeit und Stand der Technik“ gibt zunächst einen Überblick über derzeitig verfügbare Hilfsmittel für blinde und sehbehinderte Computernutzer und zeigt deren Defizite und die sich daraus ergebenden Barrieren auf. Aktuelle Formate und Vorgehensweisen zur Erstellung verschiedener Teile von grafischen Benutzungsoberflächen und deren Beziehung zur zugänglichen Gestaltung sowie Zugriffsschnittstellen zur Erzeugung angepasster Darstellungen werden diskutiert. Das Kapitel schließt mit einer Übersicht zu Richtlinien, Vorschriften und Initiativen zur zugänglichen Gestaltung.

Der Hauptteil der Arbeit gliedert sich in die beiden Aspekte der oben definierten Problemstellung „Angepasste Darstellungen und erweiterte Interaktionstechniken“ (Kapitel 3) und „Förderung der zugänglichen Gestaltung“ (Kapitel 4).

Kapitel 3 stellt zunächst die Anforderungen an angepasste Darstellungen für Blinde und Sehbehinderte zusammen, gibt Beispiele dazu und diskutiert, in wie weit diese Anforderungen durch bisherige Hilfsmittel und dazugehörige Implementierungen umgesetzt sind. Es präsentiert Konzepte zur Erfüllung der definierten Anforderungen mit modernen Technologien und Zugriffsschnittstellen und analysiert deren Vor- und Nachteile. Abschließend zeigt Kapitel 3 Umsetzungen, die im Rahmen dieser Arbeit entstanden. Dies umfasst einen Grafikeditor für die Braillezeile [TE09], Algorithmen zur Aufbereitung von Rastergrafiken, ein flexibles und konfigurierbares Framework zur Erzeugung taktiler Benutzungsoberflächen [TRSE10] sowie

1 Einleitung

zwei Vorschläge zu neuen Herangehensweisen für Vergrößerungssoftware [TRS+10] [TE08]. Bei allen Implementierungen wurde auf exakte Ergebnisse und hohe Performanz geachtet, um so die Grundlage für ein effizientes Arbeiten zu schaffen. Die beschriebenen Umsetzungen wurden im Rahmen dieser Arbeit mit kleineren Nutzergruppen (3-5 blinde oder sehbehinderte Personen) auf ihre Nutzbarkeit hin überprüft. Die Umsetzungen für grafisch-taktile Displays und der Grafikeditor kamen während des Projektes HyperBraille bei vielen Projektpartnern täglich zum Einsatz. Zudem wurde der HyperReader, in den die Konzepte und Umsetzungen zu grafisch-taktilen Displays einfließen, in umfassenden Benutzerstudien unter Leitung des Projektpartners TU Dresden evaluiert, wobei insbesondere untersucht wurde wie effizient alltägliche Arbeitsaufgaben erfüllt werden können (siehe bspw. [PNW10]).

Kapitel 4 präsentiert Ergebnisse zur Analyse von Problemen, die sich durch aktuelle Technologien, Formate und Werkzeuge bezüglich der zugänglichen Gestaltung ergeben, und gibt Vorschläge zu Lösungsmöglichkeiten [TSW+08] [TSSE09]. Es werden Lösungskonzepte zu Alternativtexten von Grafiken, Lehrmaterialien im Web und per GUI-Editor erstellten Dialogen vorgestellt. Anhand des Grafik-Standards SVG wird dargestellt, wie zugängliche Gestaltung zum Nutzen aller eingesetzt werden kann, um der Mehrarbeit, die oft erforderlich ist, auch einen höheren Nutzen gegenüberzustellen. Abschließend werden allgemeine Richtlinien formuliert, die der Förderung der zugänglichen Gestaltung durch jedermann dienen.

Kapitel 5 fasst die Ergebnisse der Arbeit zusammen und gibt einen Ausblick zu weiteren möglichen Arbeiten und Verwendungsmöglichkeiten in anderen Gebieten.

Über das Kernthema dieser Arbeit hinaus wurden im Kontext des Themenbereichs E-Learning, aus welchem wesentliche Erkenntnisse zur Zugänglichkeit von grafischen Oberflächen gewonnen werden konnten, die Beiträge „An E-Learning Course on Scientific Visualization“ [TRE07], „Lernen mit Web-basierten interaktiven Systemen“ [GGRT08], „Professor Volker Claus: vom o. Professor zum e-Professor“ [GTW09] und „BSLern - Ein Lernprogramm für Punktschrift“ [TM10] veröffentlicht.

2 Grundlagen der Arbeit und Stand der Technik

Die folgenden Abschnitte geben einen Überblick über den heutigen Stand der Technik zum Zugang Blinder und Sehbehinderter zu grafischen Oberflächen. Es werden aktuelle Hilfsmittel vorgestellt und deren Probleme angesprochen. Die Bedeutung des Begriffs „grafische Oberfläche“ im Rahmen dieser Arbeit wird verdeutlicht. Der Aufbau grafischer Oberflächen, Zugangsmöglichkeiten und Gestaltungsmöglichkeiten zur Erhöhung der Zugänglichkeit, sowie diesbezügliche Richtlinien werden dargestellt.

2.1 Hilfsmittel für blinde und sehbehinderte Computernutzer

Blinde Computernutzer erschließen sich die Monitor-Ausgabe heute hauptsächlich mit sogenannten Screenreadern. Diese analysieren den Bildschirminhalt und geben Informationen darüber per Sprachausgabe oder auf eine Braillezeile¹ (siehe Abbildung 1 links) aus. Dabei lassen sich nur wenige Informationen über die grafische Gestaltung darstellen. Reine Grafiken sind, bis auf ihre textuellen Beschreibungen in Alternativtexten, unzugänglich. Sollen auch Grafiken zugänglich gemacht werden, müssen diese in taktile Formen gewandelt werden. Solche taktile Grafiken existieren in unterschiedlichsten Formen und werden bereits verbreitet eingesetzt. Sie sind bisher allerdings meist statisch und werden auch deshalb eher nicht für die Darstellung der grafischen Oberfläche genutzt. Die dynamische Darstellung taktiler Grafiken, wie sie für einen entsprechenden Zugang zur grafischen Oberfläche nötig wäre, kann über grafisch-taktile Displays realisiert werden. Von diesen existieren derzeit meist nur Prototypen mit geringer Softwareunterstützung. Nur wenige Geräte sind am Markt erhältlich. Ein Screenreader, mit entsprechender Ausgabe existiert bisher nicht.

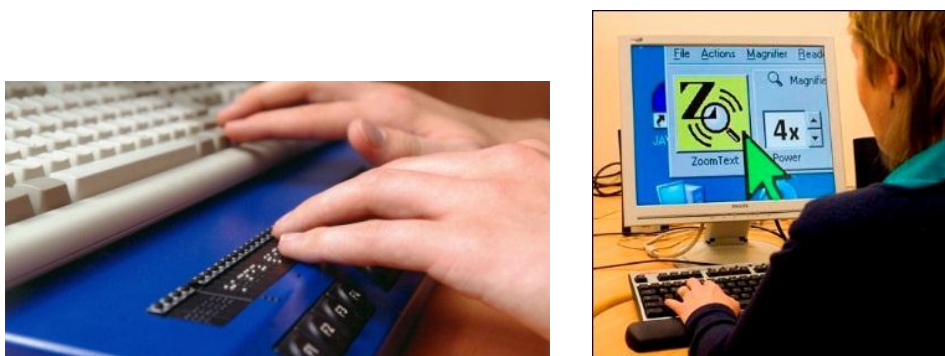


Abbildung 1 Hilfsmittel für blinde und sehbehinderte Computernutzer²
links: Braillezeile unter Tastatur, rechts: Bildschirmvergrößerungssoftware im Einsatz

¹ Eine Braillezeile ist ein technisches Gerät, das meist 40 oder 80 Braillezeichen in einer Zeile ausgeben kann und Navigationstasten zur Steuerung des Cursors bzw. Fokus, sowie zur Bedienung der grafischen Oberfläche besitzt. Es existieren verschiedenste Modelle.

² Bildquellen: <http://www.digitale-chancen.de/transfer/assets/468.jpg>,
<http://www.adaptivetech.co.nz/images/using%20a%20computer%20with%20zoomtext.jpg>

Sehbehinderte, die noch die visuelle Ausgabe nutzen, arbeiten heutzutage meist mit Bildschirmvergrößerungssoftware (engl. screen magnifier, siehe Abbildung 1 rechts). Hierbei wird die für den Monitor vorgesehene Ausgabe im Vollbild oder in einem Teilbereich des Monitors grafisch skaliert. Hinzu kommen Hervorhebungen des Textcursors, des Mauszeigers und des aktuellen Fokusbereichs sowie die Möglichkeit, andere Farbschemata zu nutzen. Stark Sehbehinderte nutzen häufig parallel die Sprachausgabe eines Screenreaders.

Die folgenden Abschnitte geben einen Überblick über die aktuellen Möglichkeiten und Einschränkungen von Screenreadern, taktilen Grafiken, grafisch-taktilen Displays und Bildschirmvergrößerungssoftware.

2.1.1 Screenreader

Zur täglichen Arbeit am Computer nutzen Blinde und teilweise auch Sehbehinderte heutzutage sogenannte Screenreader (deutsch: Bildschirmleseprogramme). Diese analysieren die Elemente der grafischen Benutzungsoberfläche, ermitteln deren Eigenschaften und passende textuelle Repräsentationen. Die gewonnenen Informationen werden per Sprachausgabe oder auf einer Braillezeile präsentiert. Dabei wird nicht nur, der am Bildschirm erkennbare Text wiedergegeben, sondern wenn gewünscht auch Informationen über das Element, zu dem dieser Text gehört. Zu einer Schaltfläche wird beispielsweise angegeben, dass es sich um eine Schaltfläche mit einer bestimmten Beschriftung handelt, ob sie gerade aktivierbar ist oder nicht, ob sie fokussiert ist oder nicht und mit welchem Tastenkürzel sie versehen wurde. Dabei analysieren Screenreader nicht nur jedes Element der Bildschirmausgabe separat, sondern vielmehr die gesamte Darstellungsstruktur. So können sie Zusammengehörigkeiten zwischen Elementen erfassen und in gewissem Umfang auch darstellen. Beispielsweise können in Dialogen mit einer Zeile von Schaltflächen für „Übernehmen“, „OK“ und „Abbrechen“ alle drei Schaltflächen auf einmal auf einer Braillezeile dargestellt und mit Hilfe der sogenannten Cursor-Routing-Tasten von Braillezeilen auch direkt aktiviert werden.

Zusätzlich bieten Screenreader erweiterte Navigationsfunktionen, die das schnelle Arbeiten mit grafischen Oberflächen und Dokumenten erleichtern. So kann beispielsweise zwischen Überschriften gesprungen werden oder von Absatz zu Absatz, wobei jeweils nur der Anfang des Absatzes vorgelesen wird. Auch diese Funktionen sind nur durch die Analyse der Gesamtstruktur möglich. Sind die nötigen Informationen dort nicht enthalten oder werden sie nicht über passende Schnittstellen zugänglich gemacht, so können diese erweiterten Funktionalitäten auch nicht angeboten werden.

Mit Sprachausgabe und Braillezeile können allerdings längst nicht alle Informationen der Bildschirmausgabe zugänglich gemacht werden. Ein großes Problem besteht darin, dass für beide Ausgabeformen eine linearisierte Darstellung der eigentlich zweidimensionalen Informationen nötig ist. Dadurch ist es schwer, einen Gesamtüberblick über die grafische Oberfläche zu erlangen. Ergeben Informationen nur im zweidimensionalen Zusammenhang einen Sinn, so ist es schwer, diese richtig zu erfassen. Einfach darstellen lässt sich dies am Beispiel

2.1 Hilfsmittel für blinde und sehbehinderte Computernutzer

einer Tabelle wie sie in Abbildung 2 zu sehen ist. Diese zeigt links die Originaldarstellung eines Vorlesungsplans in Tabellenform und rechts einen Ausschnitt der linearisierten Darstellung, wie sie durch Sprachausgabe oder auf der Braillezeile (zeilenweise) ausgegeben werden würde. In Tabellenform sind die Zusammenhänge zwischen Vorlesung, Tag und Uhrzeit, sowie Doppelbelegungen von Terminen klar erkennbar. In der linearisierten Ausgabe ist nicht zu erkennen, welche Vorlesung an welchem Tag stattfindet. Glücklicherweise bieten aktuelle Screenreader Funktionen zur Zuordnung von Tabellenköpfen zu Tabellenzellen. Allerdings funktioniert dies nur, wenn die Tabellen auch entsprechend ausgezeichnet sind. Und bei zeilen- oder spalten-übergreifenden Tabellenzellen, wie es im Beispiel zu sehen ist, gibt es auch häufig bei korrekter Auszeichnung Probleme.

Zeit	Montag	Dienstag			Mittwo
8		8:00 - Saal 2 Vorlesung: FVT (DW)	8:00 - Raum 1 Vorlesung: CV (GH) 14-tägig	8:00 - Raum 1 Übung: CV (GH) 14-tägig	
9					
9:45 - 11:15	Saal 1 Vorlesung: NN (GH)				9:45 - Saal 2 Vorlesung: BS (DW)
10					
11					
11:30 - 13:00	Raum 1 Vorlesung: CV (GH)	11:30 - Saal 2 Vorlesung: VIS (TE)			11:30 - Saal 2 Vorlesung: VIS (TE)
12					
13:10 - 14:00	Raum 2 Seminar: AC (DW)				13:00 - Saal 2 Sprechstunde: FGB (BS, MR, H)
13:30 - 14:00	Pool Aufgabenbesprechung: FGB (BS, MR, HB) Pool Fachpraktikum: FGP (DK, GM, AP)				
14:00 - 14:45	Raum 2 Übung: BS (DW)	14:00 - Saal 2 Vorlesung: BS (DW)	14:00 - Raum 3 Hauptseminar/Seminar: BDVA (SK, BH)		

Zeit	Montag	Dienstag	Mittwo
8			
8:00 - 9:30	Saal 2		
	Vorlesung: FVT (DW)		
8:00 - 9:30		Raum 1	
		Vorlesung: CV (GH)	
		14-tägig	
8:00 - 9:30			Raum 1
			Vorlesung: CV (GH)
			14-tägig
8:00 - 9:30			Raum 1
			Vorlesung: CV (GH)
			14-tägig
9:45 - 11:15	Saal 1		
	Vorlesung: NN (GH)		
9:45 - 11:15			Saal 2
			Vorlesung: BS (DW)
11:30 - 13:00			Saal 2
			Vorlesung: VIS (TE)
13:00 - 13:30			Saal 2
			Sprechstunde: FGB (BS, MR, H)
9:45 - 11:15	Saal 1		
	Vorlesung: NN (GH)		
9:45 - 11:15	Saal 2		
	Vorlesung: BS (DW)		

Abbildung 2 Vorlesungsplan in Tabellenform (links) und in linearisierter Textausgabe (rechts)

Zudem sind reine Grafiken, die keine zusätzliche Beschreibung haben, unzugänglich. Auch entfällt die Möglichkeit, komplexe Zusammenhänge, mittels grafischer Informationen anschaulicher zu vermitteln. Durch Screenreader werden derzeit lediglich wenige Informationen zur grafischen Gestaltung von Text wiedergegeben, wie beispielsweise die Textfarbe und Textattribute wie Fett, Kursiv und Unterstrichen.

Durch eine symbolhafte Darstellung können auf der Braillezeile aber zumindest in geringem Umfang Nachbarschaftsbeziehungen zwischen unterschiedlichen Formatierungen erfasst werden. So kann beispielsweise festgestellt werden, ob einzelne Buchstaben eines Wortes oder einzelne Worte eines Satzes unterschiedlich formatiert sind. Hier bietet die Braillezeile deutliche Vorteile gegenüber der Sprachausgabe. Gerade bei der Erstellung von Dokumenten sollte nicht Sprachausgabe allein eingesetzt werden, da damit keine Rechtschreibkontrolle durchgeführt werden kann. Außerdem ist über die Braillezeile zumindest ein Zugang zu simplen Grafiken im ASCII-Grafik-Format möglich. Die Braillezeile ist allerdings längst nicht in allen modernen Ländern verbreitet. Beispielsweise wird im englischsprachigen Raum meist nur mit Sprachausgabe gearbeitet. Und leider ist auch im deutschsprachigen Raum durch die zunehmende Qualität der Sprachausgabe und die stärkere Arbeit am Computer ein Rückgang an Braillelesern zu beobachten. Um diesem Trend entgegenzuwirken, wurde im Rahmen dieser Arbeit ein computerunterstütztes, leicht konfigurierbares Lernprogramm für Blindenschrift entwickelt (siehe Anhang B).

2.1.2 Taktile Grafiken

Taktile Grafiken geben blinden oder sehr stark sehbehinderten Menschen die Möglichkeit, grafische Darstellungen zu erfassen und deren Vorteile zu nutzen. Sie werden bereits seit Jahrhunderten in verschiedensten Ausprägungen eingesetzt, um Blinde an der Welt der Sehenden teilhaben zu lassen oder um ihnen eine Möglichkeit zu geben, eigene Gedanken kreativ auszudrücken. Kahlisch gibt einen Überblick über verschiedene Arten taktiler Grafiken [Kah98]. Heutzutage sind taktile Grafiken ein akzeptiertes Mittel zur Darstellung von Karten und werden auch im Schulunterricht vermehrt eingesetzt. Abbildung 3 zeigt zwei Beispiele. Viele Arbeits- und Forschungsgruppen erstellen Sammlungen taktiler Grafiken, um sie allgemein verfügbar zu machen (siehe beispielsweise [Nat10], [Bre10] und [Pro09]). Ihre Verwendung wird allerdings auch kontrovers diskutiert (siehe [AS01] und [SA01]). Ein Problem taktiler Grafiken ist deren aufwändige Erstellung. Auch wenn der Produktionsprozess durch die Einführung von grafikfähigen Blindenschriftdruckern wie den Emprint SpotDot von ViewPlus [Vie09b] wesentlich erleichtert und beschleunigt wurde, ist eine gute Aufbereitung von Grafiken für die taktile Darstellung unerlässlich. Da taktile Grafiken statisch sind, muss vor der eigentlichen Produktion gut überlegt werden, welche Teile einer grafischen Darstellung in welcher Weise taktil umgesetzt werden sollen. Ist die taktile Grafik erst einmal produziert, kann sie nicht mehr verändert werden. Der Blinde hat keine Möglichkeit störende Teile auszublenden oder sich Bereiche, die nicht gut tastbar sind, vergrößert darzustellen. Aus diesem Grund beschäftigen sich verschiedenste Arbeitsgruppen mit der Definition von Gestaltungsrichtlinien für taktile Grafiken (siehe beispielsweise [Hel01], [Ame10], [Has10], [Sch02] und [Kah98]). Werden solche Richtlinien nicht beachtet, ist das Ergebnis nur schlecht nutzbar. Die gute Gestaltung taktiler Grafiken erfordert somit Zeit und spezielle Kenntnisse. Dies ist ein großes Hindernis für ihren Einsatz im täglichen Leben. Für den Schulunterricht werden zwar heutzutage schon viele Grafiken aus Lehrbüchern in aufbereiteter Form angeboten, aber häufig entstehen Grafiken erst kurz vor ihrem Einsatz oder ad-hoc als Übungsaufgabe,

2.1 Hilfsmittel für blinde und sehbehinderte Computernutzer

zur Ergänzung der Erklärungen im Lehrbuch oder als Darstellung in Testaten. Im Arbeitsleben ist die Situation noch prekärer. Hier entstehen täglich neue Grafiken, die häufig nicht lange aktuell sind. Eine aufwändige, händische Aufbereitung lohnt sich daher meist nicht.



Abbildung 3 Beispiele für taktile Grafiken³
links: nach dem Tiefziehverfahren, rechts: mit Schwellpapier (siehe auch [Kah98])

Verschiedene Projekte suchen nach Wegen, die Nutzbarkeit taktiler Grafiken im Alltag zu erhöhen. Fredj und Duce [FD07] präsentieren einen Prozess zur Erzeugung von Grafiken, die sich zur automatischen Umwandlung in taktile Darstellungen eignen. Hierbei wird der darzustellende Inhalt zunächst in einer domainspezifischen Beschreibungssprache angegeben. Diese wird in eine Zwischenrepräsentation konvertiert, aus welcher dann verschiedene, grafische und textuelle Repräsentationen generiert werden können. Es ist allerdings immer fraglich, ob solche speziellen Prozesse jemals Einzug in den Alltag finden werden. Des Weiteren existieren spezielle Editoren für taktile Grafiken (beispielsweise [KNS08] und [Vie09a]). Zudem wurden verschiedene Anleitungen zur Aufbereitung von Grafiken mit Hilfe von Standard-Editoren wie Microsoft Word, Adobe Photoshop oder Corel Draw entwickelt (siehe beispielsweise [Tex08], [Hig10] und [Pro09]). Diese Editoren können allerdings meist nicht von Blinden selbst verwendet werden.

Sollen Grafiken, wie sie heute im Alltag vorkommen, schnell zugänglich gemacht werden, muss eine individuelle Erkundung von grafischen Darstellungen möglich sein. Dieser Herausforderung stellt sich die Firma ViewPlus mit ihrem System IVEO [Vie08]. Hierbei werden normale Grafiken über den grafikfähigen Blindenschriftdrucker Tiger ausgegeben und auf ein druckempfindliches Eingabetablet gelegt. Auf diesem können Bereiche der Grafik ausgewählt und in vergrößerter Darstellung erneut gedruckt werden. Ein Nachteil dabei ist natürlich das hohe Druckaufkommen bei ausgiebiger Exploration komplexer Grafiken. Außerdem erlaubt das System bisher nicht, einzelne Teile der Grafik auszublenden oder nach Farben zu selektieren. Die Interaktivität ist also nur sehr begrenzt.

Grafisch-taktile Displays bieten die Chance, viele Probleme mit taktilen Grafiken zu lösen. Auf ihnen können Grafiken aus dem Alltag schnell und mit hohem Interaktivitätsgrad präsen-

³ Bildquellen: http://shop.aph.org/wcsstore/APHConsumerDirect/images/catalog/products_large/1-08845-00_Basic_Tact_Anatomy.jpg, <http://www.gfai-sachsen.de/blwbt.htm>

tiert werden. Änderungen an den Grafiken können von den Benutzern selbst vorgenommen und gespeichert werden. Somit können sich blinde oder stark sehbehinderte Menschen selbst stärker an der Erstellung guter taktiler Grafiken beteiligen.

Eine andere Form des interaktiven Zugangs zu Grafiken für blinde und stark sehbehinderte Menschen bieten haptische Geräte wie Phantom [Sen10] oder Falcon [Nov10] (siehe Abbildung 4). Auch mit ihnen ist die individuelle Erkundung von grafischen Darstellungen und deren Änderung möglich. Sie können ihre Stärken aber vor allem im Bereich von dreidimensionalen Grafiken ausspielen. Zur Schaffung eines Überblicks über eine grafische Benutzungsschnittstelle und zur Interaktion damit sind sie aber weniger geeignet, weshalb sie im Rahmen dieser Arbeit nicht betrachtet wurden. [SW03] und [YKB03] präsentieren Arbeiten zur Exploration und Erstellung von Grafiken durch Blinde mit Hilfe haptischer Geräte.



Abbildung 4 Haptische Ein-Ausgabe-Geräte⁴ (Phantom Omni von SensAble Technologies [Sen10] und Falcon von Novint Technologies [Nov10])

2.1.3 Grafisch-taktile Displays

Seit den 1980er Jahren wurden verschiedenste grafisch-taktile Displays entwickelt. Vidal-Verdú und Hafez geben in [VH07] eine gute Übersicht dazu. Die Mehrzahl dieser Displays bestehen aus einer Matrix von Stiften, die einzeln angehoben und abgesenkt werden können. Sie unterscheiden sich in der Anzahl der verwendeten Stifte und dem Abstand der Stifte zueinander. Leider sind nur wenige dieser Geräte tatsächlich am Markt verfügbar. Diese haben einen ähnlichen Stiftabstand wie die Stifte in Modulen einer Braillezeile und bieten zwei Stiftzustände – gehoben oder gesenkt. Auch die Ortsauflösung und die Größe sind eher gering. Deshalb werden sie nur genutzt, um Icons, einzelne Schwarzschriftzeichen oder Ausschnitte aus Funktionsgrafiken darzustellen (siehe Abbildung 5 [Han10] und [ABT10]). Mit diesen Geräten einen Überblick über eine grafische Oberfläche zu erhalten, ist jedoch schwierig. Ein etwas größeres Display bietet KGS Corporation mit dem ebenfalls käuflich erhältlichen „Dot View“ (siehe Abbildung 5 rechts [KGS10]). Größere Ausgabeflächen, wie sie auch die Stiftplatte der Firma Metec von 1986 anbot [Kl087], bringen neben einer besseren Übersicht über grafische Darstellungen auch den Vorteil, dass Text in Braille mehrzeilig mit

⁴ Bildquellen: <http://www.sensable.com/documents/images/LargePHANTOMOmniImage.jpg>,
<http://www.slipperybrick.com/2008/02/novint-falcon-black/>

2.1 Hilfsmittel für blinde und sehbehinderte Computernutzer

einer angemessenen Zeichenzahl darstellbar ist. So können beispielsweise Vorgänge wie das Umstellen von Formeln gut unterstützt werden. Aber auch die allgemeine Übersicht über den Text wird so erleichtert.



Abbildung 5 Beispiele erwerbbarer grafisch-taktile Displays⁵
v. l. n. r. Handy Tech GWP [Han10], Abtim VideoTIM [ABT10], KGS Dot View DV-1 [KGS10]

Das Hauptproblem der grafisch-taktilen Displays ist ihr hoher Anschaffungspreis bei gleichzeitig geringem Nutzen. Zwar wurden schon verschiedene Softwareanwendungen präsentiert und deren Nützlichkeit nachgewiesen (siehe z.B. [Alb06], [ABT10], [KW04], [ROE04], [IKT+08] und [WKOY06]), aber bisher sind dies nur Insellösungen. Sie zeigen lediglich spezielle Anwendungen oder gar nur Teile davon. Eine Bedienung grafischer Benutzungsoberflächen ist bisher nicht möglich. Dadurch können grafisch-taktile Displays derzeit nur als Zusatz zu Braillezeile und Sprachausgabe eingesetzt werden. Dies rechtfertigt den Anschaffungspreis in keinsten Weise. Der aktuelle Prototyp der Firma Metec „BrailleDis 9000“ [VWB08], welcher auch im Rahmen dieser Arbeit eingesetzt wurde, wird beispielsweise mit einem Preis von etwa 85.000 Euro beziffert. Das kleine GWP von Handytech kostet etwa 10.000 Euro [Ins10]. VideoTIM von Abtim wird mit 27.000 Euro angegeben [Kat10]. Natürlich wird der Preis solcher Displays in absehbarer Zeit nie in dem Bereich liegen, wie etwa normale Monitore. Auch Braillezeilen, die mittlerweile zumindest im deutschsprachigen Raum verbreitet und Standardausstattung für Blinde sind, kosten zwischen 3.000 und 10.000 Euro [flu10]. Wenn aber durch entsprechende Softwareunterstützung grafisch-taktile Displays als alleinige Ausgabegeräte zur Bedienung eines Computers ausreichend wären, so würde sich auch deren Verbreitung erhöhen und damit deren Preise verringern. Zudem wird durch verschiedenste Forschungsgruppen ständig an Technologien geforscht, die die Herstellungskosten für taktile Displays verringern und gleichzeitig für portable, großflächige Displays geeignet sind (siehe z.B. Abbildung 6, [KST+07] und [RP09]). Besonders im Bereich der virtuellen Realität werden derzeit taktile Ausgabegeräte entwickelt, deren Technologien auch für grafisch-taktile Displays einsetzbar sind (siehe z.B. [MLWS09] und [KMB+03]). Da der Markt der Virtuellen Realität ein stetig wachsender ist, könnte dies zu einer drastischen Verringerung der Produktionskosten taktiler Displays führen. Auch in anderen Bereichen wurde der Nut-

⁵ Bildquellen: <https://www.handytech.de/de/normal/produkte/fuer-blinde/gwp/index.html>,
http://www.abtim.com/Zeitschriftlesen_mit_VideoTIM.jpg, <http://www.kgs-jpn.co.jp/img/dv01.gif>

zen taktiler Displays erkannt. So zeigte die Eberle GmbH im Projekt „digitalSTROM“ wie ein taktiler Display zur Bedienung einer Haustechnikanlage durch Blinde und Sehbehinderte genutzt werden kann [Ebe10]. Selbst im militärischen Bereich werden Einsatzmöglichkeiten taktiler Displays eruiert, wie etwa [YC05] zeigt. Da sie dort meist in Einsatzkleidung integriert werden sollen, könnten hier besonders flexible und gleichzeitig sehr robuste Technologien entstehen. Grafisch-taktile Displays haben also auch aus rein technologischer Sicht Zukunft.

Statische taktile Grafiken können und sollen aber durch den Einsatz grafisch-taktile Displays nicht abgelöst werden. Derzeitige taktile Displays können durch ihre geringe Auflösung nie die gleiche Qualität erreichen wie beispielsweise Schwellpapiergrafiken (siehe Abbildung 3). Es wurden zwar schon höher auflösende Displays entwickelt (siehe [NIS09]) aber auch deren Ausgabequalität ist nicht mit guten taktilen Grafiken vergleichbar. Zudem ist bei solchen Displays nur schwer eine gute Braille-Darstellung möglich, was für den Gebrauch im Arbeitsalltag entscheidend ist. Erst wenn grafisch-taktile Displays in der Art produziert werden können, wie Jonathan Lucas sie in seiner Design-Studie „Siafu“ entworfen hat [Luc09], können sie Darstellungen von gleicher Qualität wie statische taktile Grafiken liefern (siehe Abbildung 6). Gut aufbereitete taktile Grafiken zu Unterrichtszwecken oder in Museumsführern und gute taktile Karten werden aber auch dann immer noch nützlich und sinnvoll sein, da eine automatische Aufbereitung nie so gut auf den jeweiligen Zweck der Grafik und die Zielgruppe abgestimmt sein kann, wie eine durch Menschen durchgeführte Aufbereitung. Zudem werden auch statische Grafiken zum Beispiel als feste Installation in Museen und an Ausflugsorten oder auch als Umgebungs- und Notfallpläne weiterhin sinnvoll sein, da diese für Blinde auch ohne weitere Infrastruktur wie etwa Computernetzwerke und elektronische Displays direkt zugänglich sind. Diese Arbeit verfolgt keinesfalls das Ziel, sie zu ersetzen. Für eine effiziente Arbeit mit grafischen Benutzungsoberflächen sind grafisch-taktile Displays aber unerlässlich.

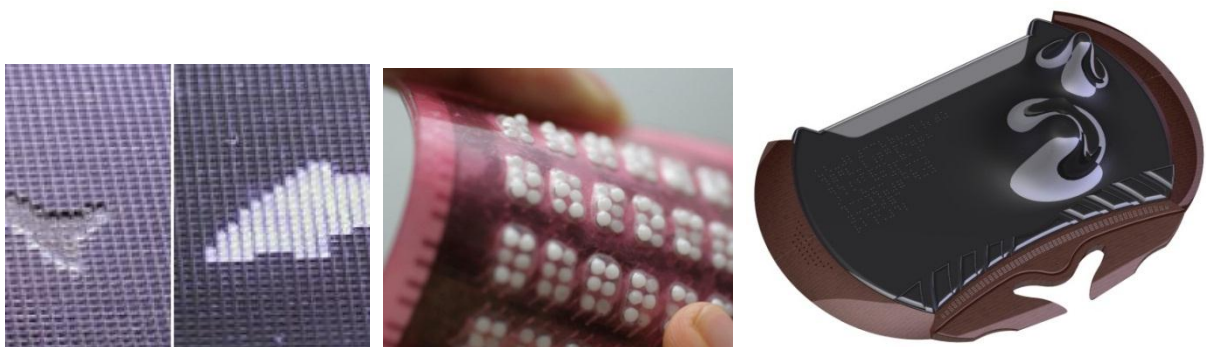


Abbildung 6 Ergebnisse aktueller Forschung zu neuen Technologien für taktile Displays⁶
v. l. n. r.: [KST+07], [RP09], Designstudie "Siafu" von Jonathan Lucas [Luc09]

⁶ Bildquellen: <http://www.stern.de/wissen/technik/forschung-ein-display-zum-lesen-und-fuehlen-652258.html>,
http://www.ntech.t.u-tokyo.ac.jp/Archive/Archive_download/image_4_low.jpg,
<http://www.coroflot.com/lucasite/siafu/1>

2.1.4 Bildschirmausgabe für Sehbehinderte

Von sehbehinderten Menschen wird für die Arbeit am Computer vorwiegend explizite Bildschirmvergrößerungssoftware eingesetzt. Diese wird von verschiedenen Herstellern angeboten. [INC08] gibt dazu einen guten Überblick. Allen ist gemein, dass sie die vergrößerte Darstellung durch eine einfache Skalierung der originalen Bildschirmausgabe und einer Nachbearbeitung von Schrift und Farben zur Erhöhung der Qualität erzeugen. Die vergrößerte Darstellung kann wahlweise auf drei Arten angezeigt werden:

- Bildschirmfüllend (Abbildung 7 links): Es wird nur die vergrößerte Darstellung gezeigt. Diese nutzt die gesamte Darstellungsfläche des Bildschirms.
- Split-Screen (Abbildung 7 mittig): Ein Teil des Bildschirms zeigt die originale Darstellung, der andere die vergrößerte. Die Aufteilung des Bildschirms ist variable bezüglich der Anordnung (links, rechts, oben, unten) und Größe (1:1, 1:2 etc.) der Bereiche.
- Lupe (Abbildung 7 rechts): Nur das fokussierte Element oder das Element unter dem Mauszeiger und dessen nähere Umgebung werden in einem kleinen Bereich der Bildschirmausgabe angezeigt und überdecken die Originaldarstellung in diesem Bereich. Der Bereich kann sich an die Größe des Elementes anpassen. Er kann an einer Stelle auf dem Bildschirm fixiert sein oder dem Mauszeiger bzw. dem Fokus folgen.

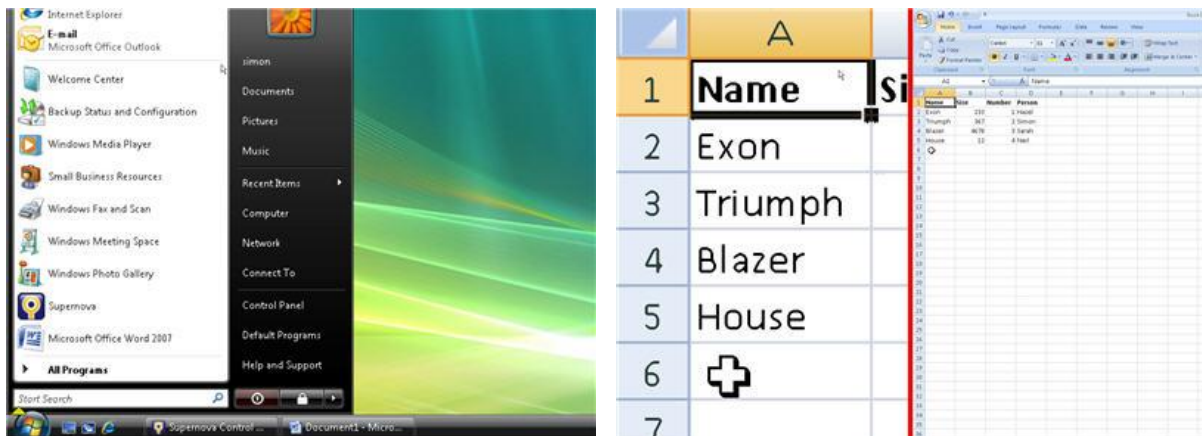


Abbildung 7 Verschiedene Darstellungsmodi bei Bildschirmvergrößerungssoftware am Beispiel des Produktes „Lunar“ der Firma Dolphin [Dol10]
v. l. n. r.: bildschirmfüllend, Split-Screen, Lupe mit Mausverfolgung und Größenanpassung

2 Grundlagen der Arbeit und Stand der Technik

Neben der Vergrößerung selbst bietet solche Software viele weitere nützliche Funktionen, wie beispielsweise Änderungen der Farbeinstellungen, spezielle Markierungen für den Textcursor⁷ oder die Verfolgung und deutliche Markierung des fokussierten Elementes.

Diese spezielle Bildschirmvergrößerungssoftware ist allerdings nicht in Betriebssystemen integriert und so bei Menschen, die nur eine leichte, meist altersbedingte Seheinschränkung haben, sich selbst also nicht als „sehbehindert“ bezeichnen würden, kaum bekannt. Sie wird deshalb von diesen eher nicht genutzt, auch wenn sie ihnen Vorteile bringen könnte. Selbst wenn Computernutzer mit Seheinschränkungen Kenntnis über die Existenz solcher Software haben, sind häufig die gewisse Umstellung der Arbeitsweise am Computer, die die Nutzung einer solchen Software mit sich bringt, und der recht hohe Preis Gründe dafür, dass diese Software nicht zum Einsatz kommt. Dagegen finden Vergrößerungsmöglichkeiten, die Betriebssysteme oder Anwendungen mitbringen, mittlerweile verbreiteten Einsatz. Dies zeigt, dass sich die Anwender durchaus über die Nützlichkeit von Vergrößerungsmöglichkeiten bewusst sind. Zu diesen leicht zugänglichen Vergrößerungsmöglichkeiten gehören:

- Anwendungsbezogene Vergrößerung, wie sie durch Webbrowser und Office-Programme angeboten wird,
- mit Betriebssystemen ausgelieferte Bildschirmlupen-Software,
- Möglichkeiten zur Änderung der Standardschriftgröße,
- Möglichkeiten zur Änderung der Auflösung der Bildschirmausgabe.

Dabei sind Bildschirmlupen-Programme in ihrer Funktionsweise vergleichbar mit Bildschirmvergrößerungssoftware. Allerdings bieten sie wesentlich weniger Funktionen und die Qualität der vergrößerten Darstellung ist wesentlich schlechter. Auch die Änderung der Auflösung führt zu einer Verschlechterung der Darstellungsqualität. Das Bild wird schwammig. Dadurch werden die Augen gerade beim Lesen von Texten stärker beansprucht als bei einer klaren Darstellung. Eine mit der Verringerung der Auflösung konzeptionell vergleichbare Auswirkung auf die Darstellung der Anwendungen am Bildschirm hat die Änderung der Standardschriftgröße. Auch hier wird das Verhältnis zwischen Schriftgröße und für die Darstellung zur Verfügung stehender Platz verändert. Man erreicht hier aber eine wesentlich bessere Darstellungsqualität. Allerdings wirkt sich die Änderung der Standardschriftgröße aufgrund von nachlässiger Implementierung nicht auf alle Anwendungen in korrekter Weise aus. Manche Anwendungen reagieren überhaupt nicht auf diese Einstellung. Bei anderen Anwendungen überlappen sich verschiedene Elemente der Benutzungsoberfläche oder es werden nicht mehr alle Interaktionselemente oder beschreibenden Texte dargestellt, da sich die Größe des umgebenden Fensters nicht an die Größe der Elemente anpasst und auch keine Scrollbalken ergänzt werden. In diesem Fall bietet eine Darstellung bei verringerter Auflösung aber auch keine besseren Ergebnisse. Nur kann die Auflösung derzeit meist wesentlich schneller geändert werden als die Standardschriftgröße. Aufgrund der technologischen Prob-

⁷ Der Textcursor wird auch als Caret oder Textmarke bezeichnet.

2.1 Hilfsmittel für blinde und sehbehinderte Computernutzer

leme sind diese beiden Vergrößerungsmöglichkeiten derzeit nur für relativ geringe Vergrößerungen im Bereich bis etwa 1,5-fach ohne größere Nachteile einsetzbar. Abbildung 8 zeigt Beispiele zu den drei zuvor beschriebenen Vergrößerungsmöglichkeiten und illustriert die angesprochenen Probleme.

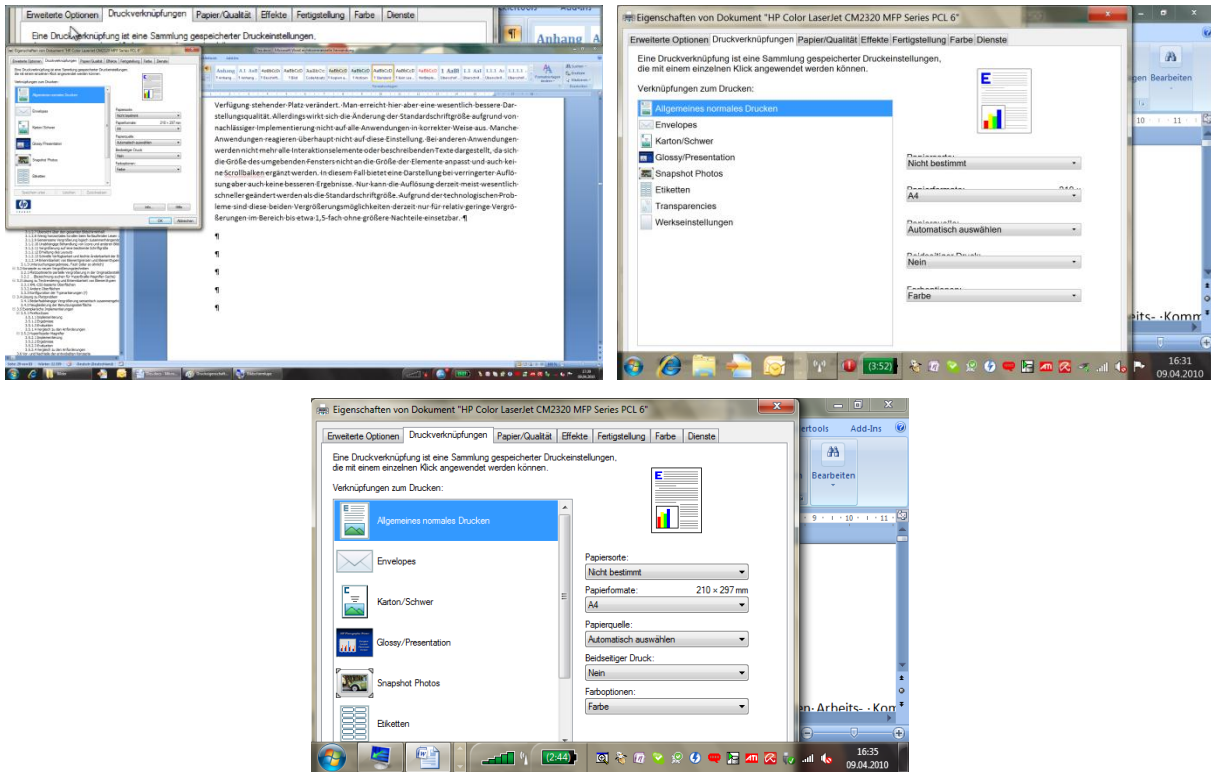


Abbildung 8 Beispiele für Vergrößerung mit Bildschirmlupe, Erhöhung der Standardschriftgröße und Verringerung der Auflösung (in den beiden letzteren Fällen sind nicht mehr alle Interaktionselemente des Dialoges sichtbar bzw. mit der Maus bedienbar)

Bei Verwendung der anwendungsbezogenen Vergrößerung dagegen liegt die maximal verfügbare Vergrößerung meist zwischen 5- und 10-fach. Da Schrift und Vektorgrafiken bei dieser Vergrößerungsmöglichkeit explizit für die gewählte Vergrößerungsstufe gerendert werden, ist die Darstellungsqualität entsprechend gut. Allerdings findet die Vergrößerung bisher nur im Bereich des Anwendungsdokumentes statt und wirkt sich nicht auf die Benutzeroberfläche aus. Diese Vergrößerungsmöglichkeit kann also nicht als vollwertige Unterstützung bei Seheinschränkungen gelten. Die anwendungsbezogene Vergrößerung tritt in zwei Varianten auf. Variante 1 entspricht der Funktionsweise der Bildschirmvergrößerungssoftware. Das heißt, die Darstellung des Anwendungsdokumentes wird auf ein Vielfaches des ursprünglichen Darstellungsbereichs vergrößert. Das Originallayout des Dokumentes bleibt erhalten. Bei Variante 2 bleibt der Darstellungsbereich in der Breite auf den ursprünglichen Bereich beschränkt. Das Dokument wird passend umgebrochen. Abbildung 9 illustriert dies. Je nach Anwendung kann der Benutzer sich zwischen den beiden Varianten frei entscheiden oder sie werden kombiniert angewendet oder eine der beiden Varianten fest vorgegeben.

2 Grundlagen der Arbeit und Stand der Technik

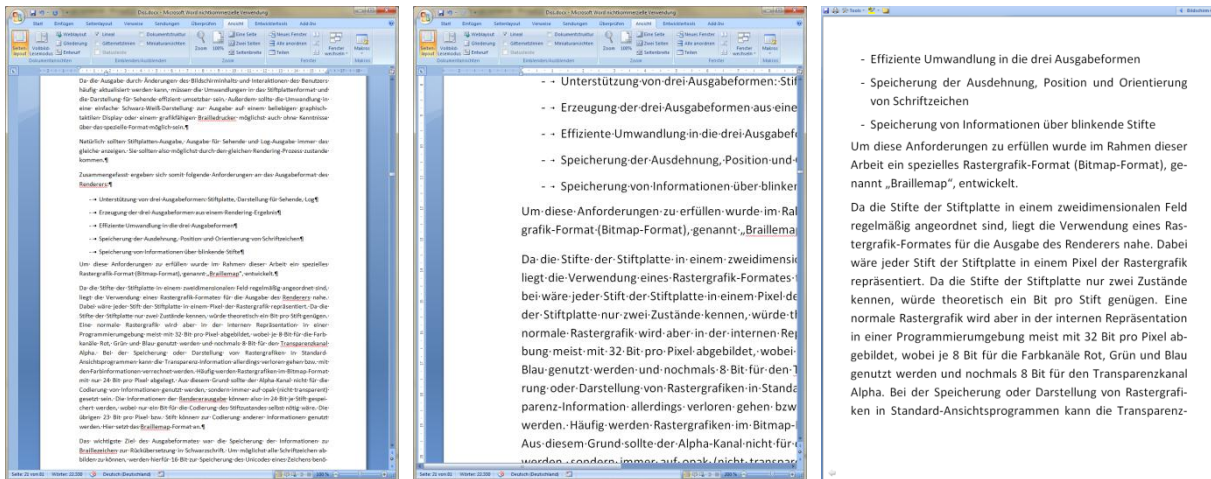


Abbildung 9 Varianten der anwendungsbezogenen Vergrößerung v. l. n. r.: Originalansicht, Vergrößerung mit Variante 1, Vergrößerung mit Variante 2

Die vorgestellten Vergrößerungsmöglichkeiten lassen sich entsprechend der Konzepte, die durch sie verwirklicht werden, in folgender Taxonomie eingliedern.

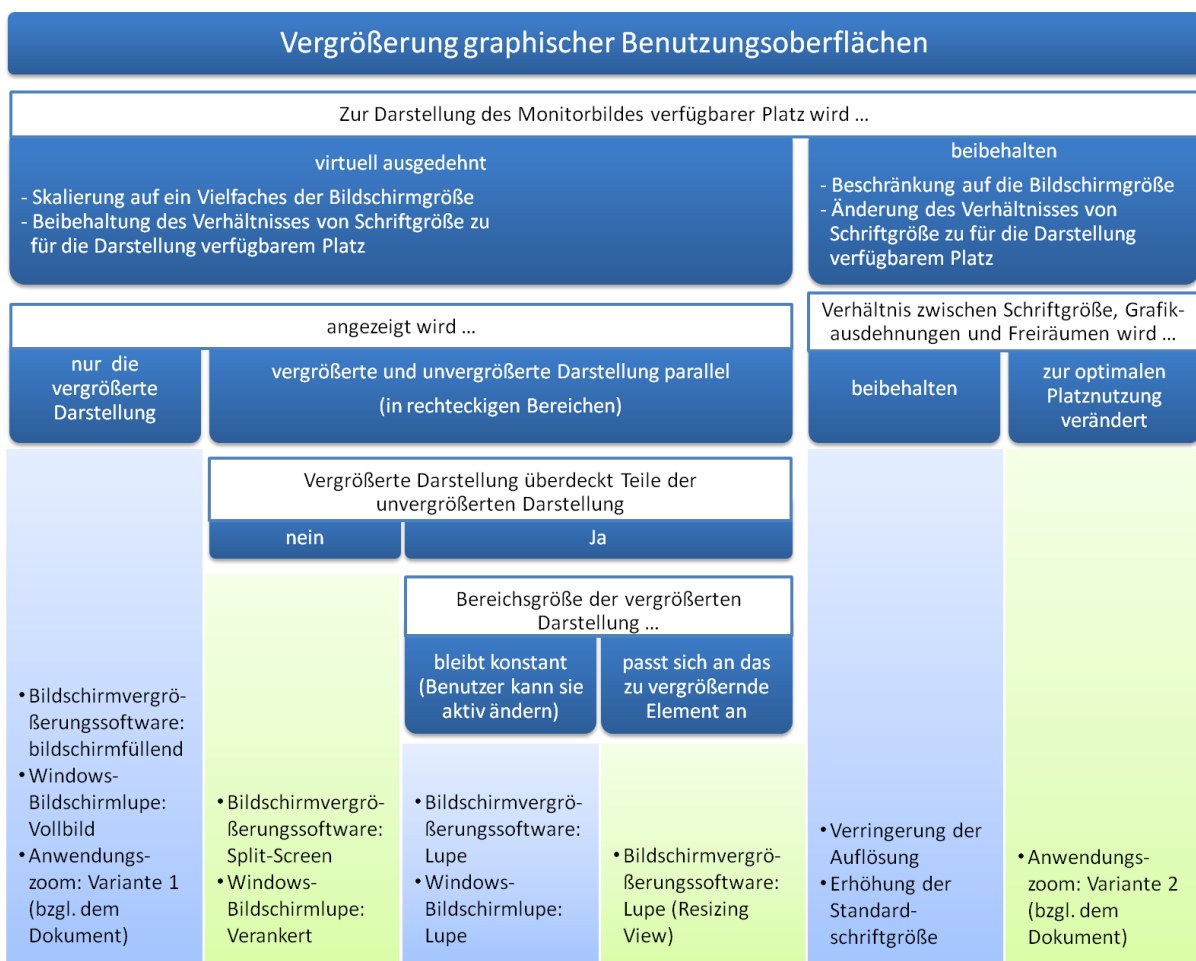


Abbildung 10 Taxonomie der Vergrößerungstechniken

Dabei ermöglichen derzeit nur die Umsetzungen der Bildschirmvergrößerungs-Programme in allen Fällen die gewünschte Vergrößerung. Aber auch diese Programme bieten noch keine

2.1 Hilfsmittel für blinde und sehbehinderte Computernutzer

optimale Unterstützung für Computerbenutzer mit Sehbehinderung oder Seheinschränkungen. Beispielsweise ist zur Erfassung der gesamten vergrößerten Darstellung ein hoher Scroll- bzw. Navigationsaufwand nötig, sowohl vertikal als auch horizontal. Dies bedeutet für den Benutzer eine Umstellung in der Arbeitsweise am PC. Zwar ist vertikales und in speziellen Situationen auch horizontales Scrollen fester Bestandteil der Arbeit am Computer, es bezieht sich aber immer nur auf das Anwendungsdokument und nicht, wie bei der vergrößerten Darstellung, auch auf die Benutzungsoberfläche. Untersuchungen zeigten, dass dadurch die Arbeitseffizienz beeinträchtigt wird [BL96]. Im Wesentlichen begründet sich dies in der virtuellen Ausdehnung der Bildschirmgröße auf ein Vielfaches der Originalgröße. Dadurch werden nützliche Funktionalitäten von Anwendungen außer Kraft gesetzt, die die Anzeige auf die Bildschirmgröße beschränken, um einen guten und schnellen Überblick zu gewährleisten. So kommt es häufig vor, dass Änderungen in der Darstellung, wie beispielsweise das Erscheinen von Dialogen oder anderen Meldungen, vom Benutzer nicht wahrgenommen werden können, da sie nicht in der aktuellen Bildschirmanzeige erscheinen. Abbildung 11 illustriert dies.

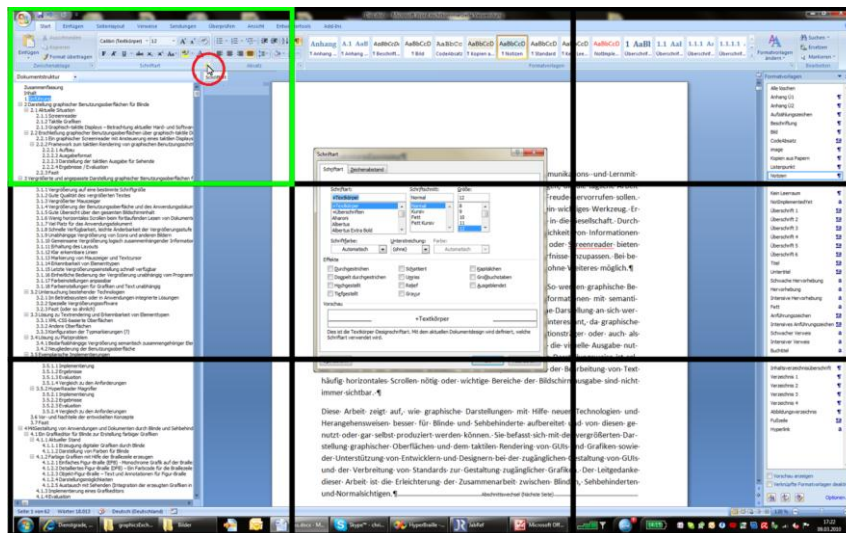


Abbildung 11 Darstellung einer Bildschirmvergrößerung auf 300%

Die Bildschirmausgabe wird auf die Größe von 9 Monitoren gestreckt. Im Beispiel befindet sich der Mauszeiger im oberen, linken Bereich eines maximierten Wordfensters, auf der Erweiterungsschaltfläche des Bereichs „Schriftart“. Der Benutzer kann hier allerdings nicht interagieren, da ein bereits geöffneter Dialog das Fenster blockiert. Dieses ist aber für den Benutzer nicht sichtbar, da er nicht in den oberen, linken Bereich der Bildschirmausgabe hineinragt.

Wie bereits erwähnt ist neben der Vergrößerung der Darstellung die Neueinfärbung bzw. die Änderung des Farbschemas eine wesentliche Funktionalität, die von Sehbehinderten benötigt wird. Auch hier können entweder Funktionen des Betriebssystems, einzelner Anwendungen oder von Bildschirmvergrößerungssoftware genutzt werden. Wie schon bei der Vergrößerung scheitern im Betriebssystem integrierte Funktionen auch hier häufig an unzureichenden Implementierungen von Anwendungen, wie Abbildung 12 zeigt.

2 Grundlagen der Arbeit und Stand der Technik

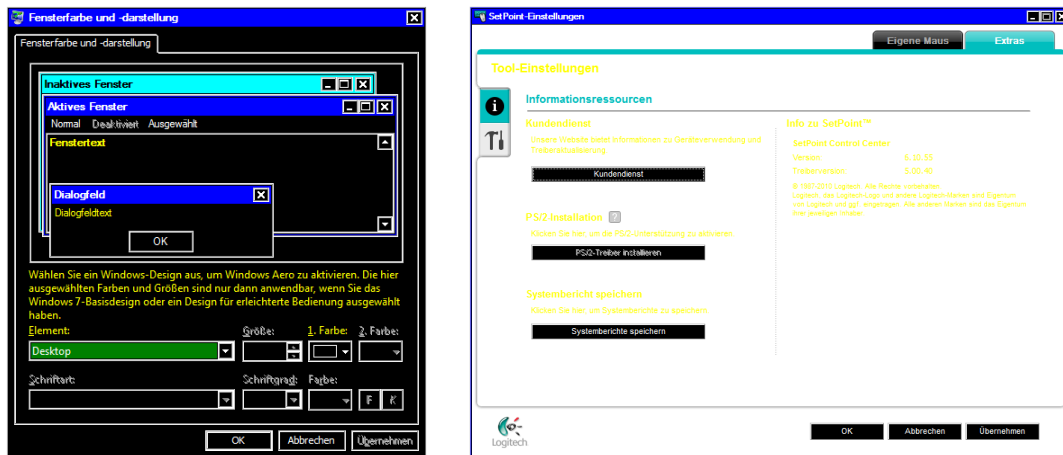


Abbildung 12 Beispiel zu Problemen bei Umstellung des Farbschemas
links: Darstellung des Farbschemas „Kontrast Nr. 1“ im Windows-Dialog
rechts: Ein Beispieldialog, bei dem Teile der Schrift auf die neue Schriftfarbe Gelb umgestellt wurden. Der Hintergrund wurde aber größtenteils nicht auf Schwarz umgestellt, sondern blieb weiß. Dadurch entsteht ein sehr geringer Kontrast zwischen Hintergrund und Schrift, so dass diese nicht lesbar ist.

Da die Anpassung von Farben bei Bildschirmvergrößerungssoftware erst auf Basis der fertigen Bildschirmausgabe (durch Nachbearbeitung) geschieht, können hier wieder bessere Ergebnisse erzielt werden. Allerdings werden dadurch auch immer die Farben von Grafiken geändert. Diese sind so zwar evtl. leichter zu erkennen. Bezieht sich ein erklärender Text allerdings auf Farben innerhalb der Grafik, so können diese Verknüpfungen nicht mehr hergestellt werden. Der Benutzer muss dann zwischen der Ansicht in Originalfarben und der angepassten Ansicht wechseln, um sowohl die Farben korrekt sehen zu können als auch den erklärenden Text leicht lesen zu können. Auch eine Split-Screen-Ansicht hilft hier nicht, da dann die Ansicht in Originalfarben nicht vergrößert ist. Es müsste also noch eine weitere Vergrößerungsmöglichkeit herangezogen werden, sofern diese existiert.

2.1.5 Zusammenfassung

Insgesamt bestehen bereits vielfältige Möglichkeiten, um Blinden und Sehbehinderten einen Zugang zu grafischen Oberflächen zu ermöglichen. Allerdings stößt man mit den heutigen Mitteln immer noch auf Grenzen. Vor allem die Gewinnung eines Überblicks und das effiziente Arbeiten mit mehreren Bereichen einer grafischen Oberfläche (wie beispielsweise Toolbar und Dokument) fallen mit den heutigen Mitteln schwer. Auch Informationen, die über Farben und Layout vermittelt oder einfach leichter erfassbar sind (wie etwa tabellarische Strukturen) sind besonders für blinde Computernutzer nur schlecht darstellbar. Aber auch Sehbehinderte benötigen hier noch mehr Unterstützung. Häufig entstehen Probleme heute aber auch durch schlechte Gestaltung der Oberflächen. Auch hier müssen noch Fortschritte erzielt werden.

2.2 Aufbau und Zugänglichkeit grafischer Oberflächen

Dieses Kapitel gibt einen Überblick zu grafischen Oberflächen, deren Bestandteile, Darstellungsprozesse und Zugangsmöglichkeiten, sowie die Möglichkeiten semantische Informationen zu hinterlegen.

2.2.1 Bestandteile und Darstellung grafischer Oberflächen

Praktisch alle modernen Computeranwendungen verfügen über eine grafische Bedienoberfläche. Selbst die Anzeige im Windows-Kommandozeilen-Fenster ist im eigentlichen Sinne nicht mehr über eine Textschnittstelle zugänglich. Grafische Oberflächen bestehen aus Bedienelementen wie Schaltflächen und Menüs, Beschriftungen (textuelle Elemente), Zustandsanzeigen wie Fortschrittsbalken und ähnlichem. Häufig enthalten sie auch ein Anwendungsdokument, welches mit Hilfe der Bedienelemente dargestellt und eventuell auch bearbeitet werden kann. Dabei wird im Rahmen dieser Arbeit alles als Anwendungsdokument bezeichnet, was nicht inhärenter Bestandteil der Bedienoberfläche ist und durch Speicherung in eine Datei potentiell in verschiedenen Anwendungen betrachtet werden kann. Dies umfasst sowohl Textdokumente wie auch Grafiken, Videos oder CAD-Darstellungen. Dabei können Dokumente selbst wieder Bedienoberflächen enthalten, welche auch wieder Dokumente darstellen können. Beispiele dafür sind Web-Office-Anwendungen wie „Google Docs“ [Goo] oder Wiki-Editoren, wie Abbildung 13 zeigt. Grafische Bedienoberflächen, die kein Anwendungsdokument enthalten sind beispielsweise Installationsdialoge, ein Datei-Explorer oder auch Spiele (siehe Abbildung 14).

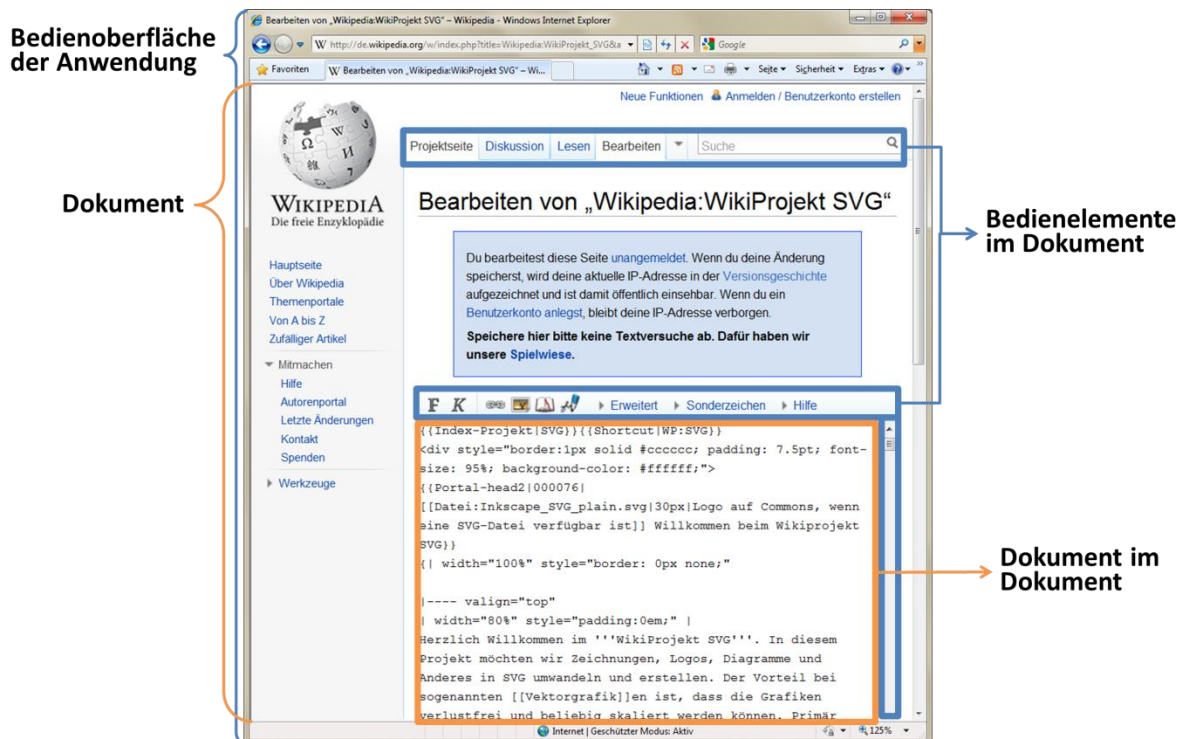


Abbildung 13 Wikipedia-Seite im Bearbeiten-Modus
als Beispiel für Bedienelemente und Dokumente in Dokumenten

2 Grundlagen der Arbeit und Stand der Technik



Abbildung 14 Beispiele für grafische Benutzeroberflächen ohne Anwendungsdokumente
links: Windows Solitär, rechts: Install Shield Dialog

Oberflächen und Dokumente werden auf unterschiedlichste Weise definiert und durchlaufen unterschiedlichste Darstellungsprozesse. Das Ziel ist letztendlich immer eine Darstellung als Rastergrafik, also als zweidimensionale Matrix von Farbwerten, zur Ausgabe auf dem Monitor. Auch textuelle Elemente werden letztlich am Monitor in Form einer Rastergrafik dargestellt und sind deshalb auch grafische Ausgaben. Durch die vielfältigen Gestaltungsmöglichkeiten, die heutzutage bestehen, tragen ja auch textuelle Elemente grafische Informationen wie Farbe und Schriftart.

Wie diese rasterisierte Darstellung zustande kommt, hängt vom genutzten Betriebssystem, dem der Anwendung zu Grunde liegenden Toolkit oder Rendering-Framework, dem Format des darzustellenden Dokuments und letztlich der Umsetzung der Anwendung selbst ab. Dabei kann es vorkommen, dass der Anwendungsentwickler lediglich auf Standardkomponenten wie vordefinierte Schaltflächen und Menüs zurückgreift oder diese Standardkomponenten durch Variation ihrer Eigenschaften (wie beispielsweise die Größe und Farbauswahl) verändert und die Erzeugung der Rastergrafik den zu Grunde liegenden Schichten überlässt. Dieses Vorgehen kommt bei den meisten Einstellungs- und Abfrage-Dialogen wie dem Installationsdialog in Abbildung 14 zum Einsatz. Es ist aber auch möglich, die Darstellungsroutinen der Standardkomponenten durch eigens definierte Routinen zu ersetzen oder die Darstellung aus reinen Grafikkomponenten zu erzeugen, die mit Interaktionen verknüpft werden (wie im in Abbildung 14 gezeigten Solitärspiel). Auch Oberflächen, die mit Hilfe ursprünglich für dynamische Grafiken entwickelten Formaten wie der vom World Wide Web Consortium (W3C) definierte Vektorgrafik-Standards SVG (Scalable Vector Graphics) [Worf] und Adobe Flash [Adoa] erstellt wurden, fallen in diese Kategorie. Die Darstellung eines Dokuments ist zudem auch abhängig davon, in welcher Anwendung es gezeigt wird. So wird beispielsweise die in Abbildung 13 dargestellte Webseite in einem Texteditor natürlich anders präsentiert als in einem Webbrowser. Der Texteditor zeigt den HTML-Quellcode der Webseite als einfachen Text an, während der Webbrowser den Quellcode interpretiert und eine entsprechende, formatierte Ausgabe mit verschiedensten Schriftgrößen, unterschiedlichen Anordnungen von Elementen und vielem Weiteren erzeugt. Der Zugriff auf die dargestellte Datei allein, sofern er überhaupt möglich ist, würde also nicht ausreichen, um zu ermitteln, wie die grafische Darstellung am Monitor aussieht. Deutlicher wird dieser Unterschied noch bei der Be-

2.2 Aufbau und Zugänglichkeit grafischer Oberflächen

trachtung von Visualisierungsprogrammen, die aufgrund von in einfachen Zahlenwerten ausgedrückten Daten (wie in Abbildung 15 links) komplexe Darstellungen erzeugen (wie in Abbildung 15 rechts). Der Darstellungsprozess und meist auch die Information darüber, wie die Ausgangsdaten zu interpretieren sind, sind hier gänzlich in der Anwendung selbst gekapselt. Zwar wären die Ausgangsdaten oftmals in Textform darstellbar, die Bedeutung der Daten wäre dadurch aber häufig nicht erkennbar. Dagegen ist die tatsächliche Programmausgabe eine eventuell sehr komplexe und bedeutungstragende grafische Darstellung.

```
70000 0.9908885610384242
0.820173 0.205461 0.495881 0.000800 1 19981
0.034453 0.559996 0.513017 0.000800 1 19947
0.459960 0.570904 0.553282 0.000800 1 19938
0.720132 0.795127 0.442842 0.000800 1 19917
0.645301 0.108976 0.600264 0.000800 1 19901
0.484750 0.800266 0.255277 0.000800 1 19898
0.619897 0.506497 0.336696 0.000800 1 19894
0.305001 0.379967 0.379982 0.000800 1 19884
...
```

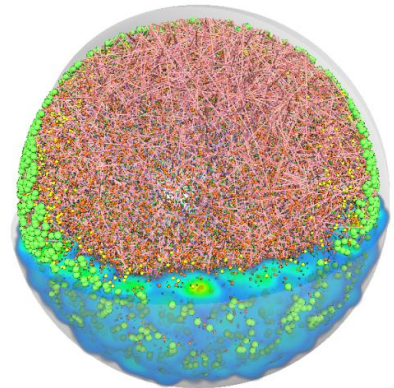


Abbildung 15 Beispiel einer komplexen Visualisierung aus der Systembiologie auf Grundlage einer sehr einfachen Datendatei (nur als Ausschnitt dargestellt, Quelle: Martin Falk, VIS)

Auch wenn der Darstellungsprozess nach festen, öffentlich zugänglichen Regeln definiert ist, kann die tatsächliche Darstellung in unterschiedlichen Umgebungen voneinander abweichen. Meist begründet sich die in einer unvollständigen Implementierungen oder Implementierungsfehlern. Ein Beispiel dafür ist die unterschiedliche Darstellung von SVG-Grafiken in den verschiedenen Webbrowsern.

Vektorgrafiken sind eine Form von Grafikformaten, bei denen die grafische Darstellung aus Primitiven wie Linien, Rechtecken, Polygonen, Ellipsen und auch Text oder komplexen grafischen Formen zusammengesetzt sind. Dabei werden die Primitive nicht über ihre einzelnen Bildpunkte definiert, also nicht als Rastergrafiken abgelegt, sondern über die für sie wichtigen geometrischen Merkmale wie etwa die Eckpunkte oder Mittelpunkt und Radius und Merkmale zur grafischen Gestaltung wie Linienfarbe, Füllfarbe und Schattierung. Vektorgrafiken sind somit, ähnlich wie die meisten Bedienoberflächen und Dokumente, nach ihrer Erstellung noch nicht abgeschlossen, sondern müssen noch rasterisiert werden. Dies bringt Vorteile, wie etwa die verlustfreie Skalierbarkeit, also die Möglichkeit die Grafiken in verschiedenen Ausgabegrößen in guter Qualität darzustellen. Es birgt aber eben auch den Nachteil, dass die endgültige rasterisierte Ausgabe auch bei gleicher Ausgabegröße unterschiedlich ausfallen kann. Abbildung 16 zeigt die Darstellung einer SVG-Grafik im Mozilla Firefox und Microsoft Internet Explorer mit dem Adobe SVG Viewer [Adob]. Da im Firefox ein Text-Gestaltungsattribut namens „baseline-shift“ nicht beachtet wird (siehe [Mozb]), wird der in der Grafik enthaltene Text an verschiedenen Positionen dargestellt. Listing 1 zeigt den Quellcode der SVG-Grafik aus Abbildung 16.

2 Grundlagen der Arbeit und Stand der Technik

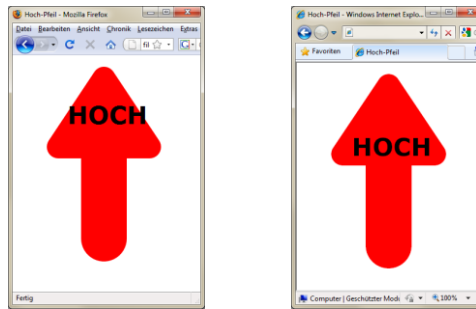


Abbildung 16 Darstellung einer SVG-Datei im Mozilla Firefox (links) und Internet Explorer (rechts)

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
  width="320" height="400" viewBox="0 0 40 50"
  stroke-linecap="round" stroke-linejoin="round">
  <path stroke="red" fill="red" stroke-width="5"
    d="M 10 20 1 20 0 1 -10 -15 z" />
  <line stroke="#FF0000" stroke-width="10"
    x1="20" y1="20" x2="20" y2="40" />
  <text x="12" y="20" style="font:bold 4pt Arial; baseline-shift:-4pt">
    HOCH</text>
</svg>
```

Listing 1 SVG-Quellcode zu Abbildung 16

Zur Präsentation grafischer Oberflächen für Blinde und Sehbehinderte kann es also wichtig sein, neben Informationen über den dargestellten Inhalt auch Informationen über den Darstellungsprozess zu erlangen. Der Darstellungsprozess entscheidet letztlich darüber, wo welche Information positioniert wird, wie sie gestaltet wird und bisweilen auch welche Bedeutung sie eigentlich besitzt. Durch den Darstellungsprozess werden also weitere Informationen über die grafische Oberfläche erzeugt, die aus dem darzustellenden Inhalt selbst nicht hervorgehen, für den Benutzer aber in unterschiedlichen Arbeitssituationen durchaus interessant sein können. Dies gilt nicht nur für Dokumente, sondern auch für Bedienoberflächen. Bei der Darstellung von Bedienoberflächen kommen häufig Layout-Algorithmen zum Einsatz, die die Bedienelemente entsprechend der zur Verfügung stehenden Ausgabefläche unterschiedlich positionieren oder auch unterschiedlich groß gestalten. In modernen Oberflächen wie beispielsweise den Ribbons von Microsoft Office können sich die Darstellung von Bedienelementen abhängig von der Ausgabegröße sogar völlig verändern, wie Abbildung 17 illustriert. Dabei können sogar verschieden Bedienelemente von der Oberfläche verschwinden. Dies kann auch Einfluss auf die Bedienung der Oberfläche haben. So unterscheidet sich beispielsweise bei dem in Abbildung 17 gezeigten Beispiel nicht nur die Darstellung der Elemente, sondern auch die Reihenfolge, in der die Elemente bei Navigation durch das Ribbon mit den Pfeiltasten oder der Tabulatortaste angesprungen werden.

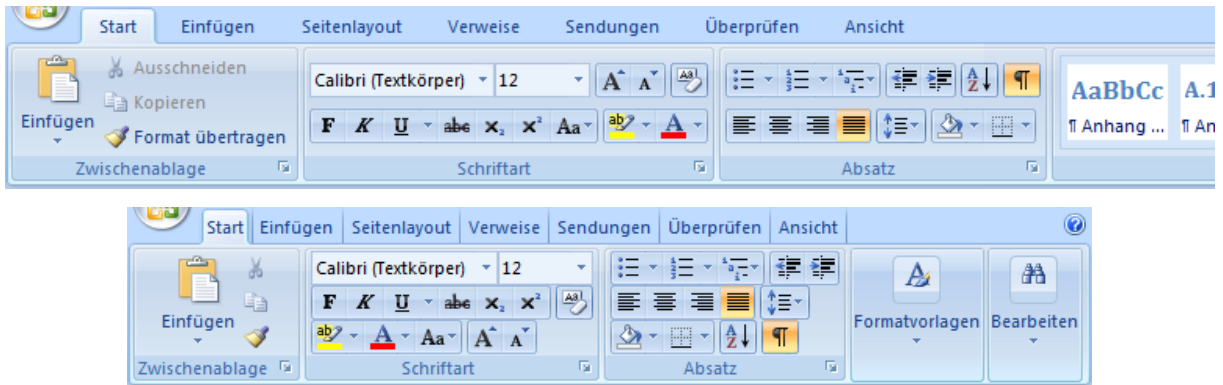


Abbildung 17 Darstellung des Ribbons von Microsoft Word in unterschiedlichen Größen
oben: Ausgabebreite etwa 1000 Pixel. Die Schaltfläche zur Anzeige von Formatierungssymbolen (Absatzmarke etc.) ist innerhalb ihrer Gruppe „Absatz“ rechts oben in der ersten Zeile von Bedienelementen positioniert. Zur Auswahl der Formatvorlagen steht ein einzelnes Bedienelement für jede Formatvorlage direkt zur Verfügung.
unten: Ausgabebreite etwa 600 Pixel. Die Schaltfläche zur Anzeige von Formatierungssymbolen (Absatzmarke etc.) ist innerhalb ihrer Gruppe „Absatz“ rechts unten in der dritten Zeile von Bedienelementen positioniert. Die einzelnen Bedienelemente zur Auswahl der Formatvorlagen wurden zu einem einzigen Bedienelement zusammengefasst.

2.2.2 Zugang zu grafischen Oberflächen

Zugang zu grafischen Oberflächen bedeutet im Rahmen dieser Arbeit einerseits die Ermittlung von Informationen über die dargestellten Inhalte und den Darstellungsprozess bzw. die dadurch erzeugte Darstellung und andererseits auch die Möglichkeit, Einfluss auf den Darstellungsprozess zu nehmen. In beiden Fällen stehen verschiedene Schnittstellen zur Verfügung, die wie auch die Darstellungsprozesse abhängig sind vom Betriebssystem, genutztem Toolkit oder Framework, Dokumentformat und der Anwendung selbst.

Durch Beeinflussung des Darstellungsprozesses kann die für die Monitorausgabe erzeugte Darstellung verändert und so direkt in eine angepasste Darstellung verwandelt werden. Zwei Möglichkeiten dazu wurden in Kapitel 2.1.4 bereits angesprochen, der anwendungsbezogene Zoom und die Änderung der Auflösung des Ausgabebildes. Die wichtigste Beeinflussungsmöglichkeit ist aber der Austausch bzw. die Änderung von Stilvorlagen (auch Designs oder Formatvorlagen genannt), die vom Darstellungsprozess genutzt werden. Stilvorlagen können sich auf verschiedenste darstellungsrelevante Attribute von Elementen beziehen, wie beispielsweise die Vorder- und Hintergrundfarbe, Schriftattribute, Größe, Position und Abstand zu anderen Elementen oder auch die Sichtbarkeit des Elements. Verschiedene Anwendungen und Betriebssysteme unterstützen hier unterschiedliche Formate mit variierendem Funktionsumfang. Auch die Vorgehensweisen zur Änderung der Stilvorlagen können sehr unterschiedlich sein. So bieten Betriebssysteme meist Systemdialoge zur Auswahl oder Erstellung von Designs, die die Darstellung von Standardbedien- oder -textelementen verändern. Auf die Sichtbarkeit und Positionierung von Elementen kann so allerdings kein direkter Einfluss ausgeübt werden. Natürlich können Elemente durch die Auswahl ungeeigneter Farben unsichtbar werden oder durch Größenänderungen Verschiebungen von Elementen auftreten.

2 Grundlagen der Arbeit und Stand der Technik

Erzeugen Anwendungen ihre Darstellung nicht aus Standardelementen, so haben die Systemeinstellungen natürlich keinen Einfluss. Einige Anwendungen bieten deshalb eigene Einstellungsdialoge oder vorgefertigte Anwendungsdesigns, aus denen der Benutzer eines wählen kann. Die Einstellungen beziehen sich dabei oft nicht nur auf die Bedienelemente, sondern auch auf das Anwendungsdokument. So stellen Programmierumgebungen meist die Möglichkeit bereit, die farbliche Gestaltung des Syntax-Highlighting zu verändern. Da diese Farbgestaltung nicht Bestandteil des dargestellten Quellcode-Dokuments ist, wird dieses durch die Änderung der Gestaltung auch nicht beeinflusst. Die Gestaltung ist also auch nicht aus dem Dokument ablesbar. Dies verhält sich anders bei modernen Office-Dokumenten. Hier besteht meist die Möglichkeit mit Formatvorlagen zu arbeiten, die beispielsweise die Gestaltung von Überschriften bestimmen. Da die grafische Gestaltung bei Office-Dokumenten heute ein wichtiger Bestandteil ist, wird sie im Dokument gespeichert, so dass sie jedem Betrachter zur Verfügung steht. Zwar werden die Formatvorlagen bei Erstellung von Office-Dokumenten zunächst aus einer Dokumentvorlage bezogen. Diese ist aber später für die Darstellung des Dokumentes nicht mehr nötig. Die Formatvorlagen werden in das Dokument kopiert. Eine Änderung der Formatvorlagen bedeutet also eine Änderung des Dokuments. Mit Hilfe der Dokumentvorlagen können die Formatvorlagen im Dokument aber auch nachträglich leicht durch einen Satz gleichnamiger aber mit anderen Darstellungsoptionen versehenen Formatvorlagen ausgetauscht werden. So ist es trotzdem möglich, Formatvorlagen flexibel für die Erzeugung angepasster Darstellungen zu nutzen (wie Abbildung 18). Vorsicht ist nur geboten bei den strukturellen Informationen, die sich häufig in Formatvorlagen befinden. So definieren Formatvorlagen in Microsoft Word beispielsweise auch die Gliederungsebene eines mit der Formatvorlage verknüpften Absatzes. Zur Erhaltung der Struktur eines Dokuments müssen diese Strukturdefinitionen also in die alternativen Formatvorlagen zur Erzeugung der angepassten Darstellung übernommen werden.

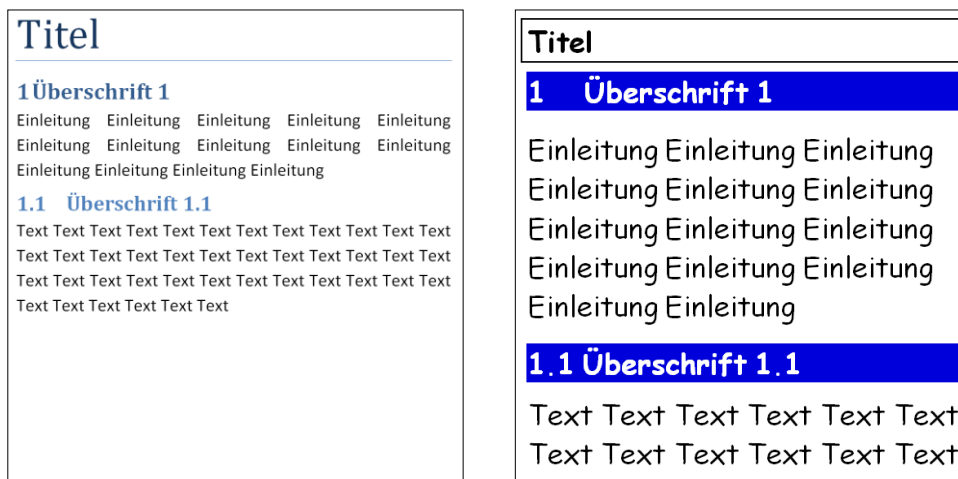


Abbildung 18 *Formatierung eines Dokuments mit unterschiedlichen Formatvorlagen*
links: Standard-Formatvorlagen von Microsoft Office, rechts: alternative Formatvorlagen

Der derzeit bekannteste Vertreter von Stilvorlagen ist CSS (Cascading Style Sheets [Wora]). CSS wurde ursprünglich für die Gestaltung von Webdokumenten entwickelt. Wird aber mittlerweile auch für Bedienoberflächen, wie beispielsweise die in XUL (XML User Interface Lan-

guage [Mozc]) definierte Oberfläche des Mozilla Firefox, oder Grafiken im SVG-Format genutzt. CSS wird seit einigen Jahren vom World Wide Web Consortium (W3C) als Webstandard betreut und weiterentwickelt. CSS ist darauf ausgelegt, die Darstellung strukturierter Dokumente zu beeinflussen. Die Trennung zwischen Struktur und Formatierung ist hier gelungen. CSS-Definitionen haben keinen Einfluss darauf, welche Bedeutung ein Element innerhalb des formatierten Dokuments hat. CSS wird meist auf Dokumente im HTML oder in XML-Formaten angewendet, ist aber nicht nur dafür verwendbar. In CSS werden die zu formatierenden Elemente über sogenannte Selektoren angegeben. Diese können sich auf den Typ der Elemente beziehen, auf deren Eigenschaften oder ihre Hierarchie im Dokument (also ihre Einordnung in Elternelemente). Listing 2 zeigt ein Beispiel zu einer HTML-Datei und einer zugehörigen CSS Formatierungsdatei. Abbildung 19 zeigt die Darstellung der HTML-Datei im Webbrowser mit und ohne Formatierung. Wie das Beispiel zeigt, können CSS-Definitionen direkt im Dokument angegeben sein oder in ein zusätzliches Dokument ausgelagert werden, das im zu formatierenden Dokument nur referenziert wird. Die Darstellung kann dann natürlich nur entsprechend der CSS-Definitionen im referenzierten Dokument formatiert werden, wenn dieses auch zur Verfügung steht. Ansonsten werden die Standarddarstellungen verwendet. Die Stilvorlagen in CSS werden nicht mit Namen versehen. So können sie also auch nicht durch gleichnamige Definitionen ausgetauscht werden. Mit den Selektoren wurde aber ein wesentlich mächtigeres Konzept geschaffen. Elemente, deren Gestaltung geändert werden sollen, können völlig frei ausgesucht werden. Zudem können Definitionen andere Definitionen, die früher im Dokument stehen, überschreiben. So überschreibt im Beispiel in der CSS-Datei die Definition der Textfarbe (`color`) für die Überschriften die, die für alle Elemente gilt. In der HTML-Datei wird die Textfarbe für das `h2`-Element nochmals überschrieben.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
  <title>CSS-Beispiel</title>
  <link href="css_test.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <h1>Kapitel</h1>
  <p>Einleitungstext</p>
  <h2 style="color: yellow">Unterkapitel</h2>
  <p>Erklärung</p>
</body>
</html>
```

```
* {
  font-family: "Comic Sans MS";
  color: navy; }

h1, h2 {
  background-color: navy;
  color: white;
  padding-left: 10px; }
```

Listing 2 Quellcode einer HTML-Datei (oben) mit CSS und eine CSS-Datei (unten)



Abbildung 19 Darstellung der HTML-Datei aus Listing 2 mit CSS-Formatierungen (links) und ohne

Sofern von Betriebssystemen oder Anwendungen passende Schnittstellen bereitgestellt werden, können Stilvorlagen auch programmatisch geändert und ausgetauscht werden. In Microsoft Windows können beispielsweise die System-Farben, die zur Darstellung von Standardbedienelementen genutzt werden, über die Methode `SetSysColors [Micd]` geändert werden. Bei Anwendungen können Schnittstellen, die zur Erweiterung der Anwendung gedacht sind (z.B. Plug-In- und Add-In-Schnittstellen), genutzt werden. Diese Schnittstellen können auch häufig zu umfassenderen Änderungen an der dargestellten Anwendung oder dem Dokument und natürlich auch zur Ermittlung von Informationen über die Anwendung, das Dokument und deren Darstellung genutzt werden. So können beispielsweise die sogenannten Extensions von Mozilla Firefox [Moza] über die für XML-basierte Dokumente und Anwendungen definierte DOM-Schnittstelle (DOM = Document Object Model) [Worb] Zugriff auf alle Elemente der grafischen Oberfläche und der dargestellten Dokumente erlangen. Über DOM ist es allgemein möglich, zur Laufzeit Elemente einer XML-Hierarchie nach ihren Eigenschaften abzufragen und diese zu verändern und sogar auch Elemente zu entfernen oder neue Elemente zu erstellen. Eine der veränderbaren Eigenschaften ist natürlich auch die Stilvorlage. Die DOM-Schnittstellen werden von verschiedensten Anwendungen, die auf XML aufbauen, XML-basierte Dokumente darstellen oder zumindest intern eine XML-basierte Struktur verwalten, unterstützt.

DOM definiert allerdings nur, in welcher Art und Weise auf die Elemente in einer Hierarchie und deren Eigenschaften zugegriffen werden kann und welche Möglichkeiten zur Veränderungen bestehen. Es definiert nicht, welche Elemente und Eigenschaften existieren. Jede Anwendung kann eine eigene Element-Hierarchie mit eigenen Eigenschaften festlegen. Wird also über die DOM-Schnittstellen auf Informationen über die grafische Oberfläche oder ein Dokument zugegriffen, so muss stets eine Interpretation der Element- und Eigenschaftsnamen durchgeführt werden, um deren Bedeutung zu erfassen. Abbildung 20 zeigt beispielhaft einen Dialog zur Eingabe des Vor- und Nachnamens, einmal in XAML (Extensible Application Markup Language) [Micf]. und einmal in XUL geschrieben. Listing 3 und Listing 4 zeigen jeweils den Quellcode zu einem der Dialoge. Die Unterschiede werden vor allem bei den Layout-Komponenten, den Gestaltungsvorgaben, der Beschriftung der Gruppe und der Angabe der Access-Keys deutlich.

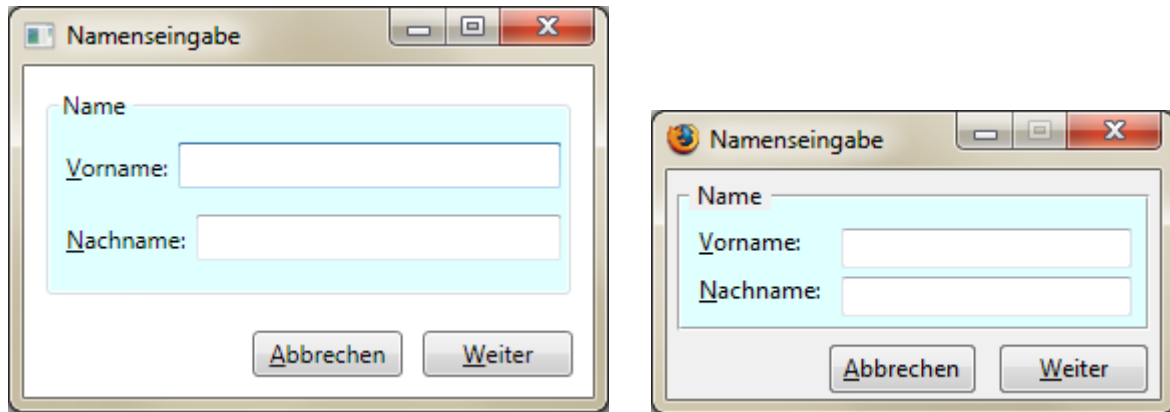


Abbildung 20 Dialoge zur Eingabe des Vor- und Nachnamens in XAML (links) und XUL (rechts)

```
<Window x:Class="WpfTestApplication.Window1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="Namenseingabe" ToolTip="Dialog zur Eingabe Ihres Namens."
Height="203" Width="300"
FocusManager.FocusedElement="{Binding ElementName=textBoxFirstName}">
<Grid>
  <GroupBox Header="Name" Name="groupBoxName" VerticalAlignment="Top"
    Margin="10,10,10,10" Background="LightCyan">
    <StackPanel>
      <DockPanel Name="panelFirstName" Margin="0,10,0,0">
        <Label Name="labelFirstName"
          Target="{Binding ElementName=textBoxFirstName}"
          Height="26" DockPanel.Dock="Left">_Vorname: </Label>
        <TextBox Name="textBoxFirstName" ToolTip="Geben Sie hier
          Ihren Vornamen ein." Height="23" DockPanel.Dock="Right" />
      </DockPanel>
      <DockPanel Name="panelLastName" Margin="0,10,0,10">
        <Label Name="labelLastName"
          Target="{Binding ElementName=textBoxLastName}"
          Height="26" DockPanel.Dock="Left">_Nachname: </Label>
        <TextBox Name="textBoxLastName" ToolTip="Geben Sie hier Ihren
          Nachnamen ein." Height="23" DockPanel.Dock="Right" />
      </DockPanel>
    </StackPanel>
  </GroupBox>
  <DockPanel Name="panelButtons" HorizontalAlignment="Right"
    VerticalAlignment="Bottom" Margin="10,10,10,10">
    <Button Name="buttonCancel" ToolTip="Drücken Sie Abbrechen, um
      den Dialog zu verlassen, ohne die Daten zu übernehmen."
      Height="23" Width="75" Margin="0,0,10,0">_Abbrechen</Button>
    <Button Name="buttonOk" ToolTip="Drücken Sie Weiter, um die
      eingegebenen Daten zu übernehmen und den Dialog zu verlassen."
      Height="23" Width="75">_Weiter</Button>
  </DockPanel>
</Grid>
</Window>
```

Listing 3 Quellcode des Dialogs zur Eingabe von Vor- und Nachname in XAML aus Abbildung 20

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/global.css" type="text/css"?>
<window
xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"
title="Namenseingabe" tooltip="Dialog zur Eingabe Ihres Namens.">
<vbox>
  <groupbox id="groupBoxName"
    style="margin: 10 10 10 10; background-color: lightcyan">
    <caption label="Name" />
    <vbox>
      <hbox id="panelFirstName" style="margin: 0 10 0 0">
        <label id="labelFirstName" control="textBoxFirstName"
          accesskey="V" style="height:26">Vorname: </label>
        <spacer flex="1" />
        <textbox id="textBoxFirstName" tooltip="Geben Sie hier
          Ihren Vornamen ein." style="height:23" />
      </hbox>
      <hbox id="panelLastName" style="margin: 0 10 0 10">
        <label id="labelLastName" control="textBoxLastName"
          accesskey="N" style="height:26">Nachname: </label>
        <spacer flex="1" />
        <textbox id="textBoxLastName" tooltip="Geben Sie hier Ihren
          Nachnamen ein." style="height:23" />
      </hbox>
    </vbox>
  </groupbox>
  <hbox id="panelButtons" pack="end" style="margin: 10 10 10 10">
    <button id="buttonCancel" label="Abbrechen" accesskey="A"
      tooltip="Drücken Sie Abbrechen, um den Dialog zu verlassen,
        ohne die Daten zu übernehmen." style="height: 23; width: 75;
        margin: 0 0 10 0; text-align: center"/>
    <button id="buttonOk" label="Weiter" accesskey="W"
      tooltip="Drücken Sie Weiter, um die eingegebenen Daten zu
        übernehmen und den Dialog zu verlassen."
      style="height: 23; width: 75; text-align: center"/>
  </hbox>
</vbox>
</window>
```

Listing 4 Quellcode des Dialogs zur Eingabe von Vor- und Nachname in XUL aus Abbildung 20

Zur Erfassung von Informationen über grafische Oberflächen existieren mit den sogenannten Accessibility-Schnittstellen weitere allgemeine Schnittstellen, bei denen diese Art der Interpretation nicht nötig ist. Dazu gehören unter anderem Microsoft Active Accessibility (MSAA, auch IAccessible genannt) [Micc, Mica], deren Nachfolger UIAutomation (UIA) [Mice] und IAccessible2 [The] und auch das GNOME Accessibility Toolkit (ATK) [gno]. Diese Schnittstellen definieren feste Typen von Bedienelementen und entsprechende Eigenschaften. Sie können theoretisch von beliebigen Anwendungen implementiert werden. Der Zugriff erfolgt über globale Dienste, die von den Implementierungen in den Anwendungen mit den nötigen Informationen versorgt werden. Dazu bilden die Anwendungen ihre jeweiligen internen Elementtypen und Eigenschaften auf die vorgegebenen Möglichkeiten ab. Abbildung 21 zeigt beispielhaft die Darstellung der UIAutomation-Informationen des XAML-Dialogs aus Abbildung 20 mit Hilfe des Programms UISpy [uis]. Hierbei können natürlich Informationen verloren gehen. Für den Dialog aus Abbildung 20 ist beispielsweise die Hintergrundfarbe der Gruppe (GroupBox) nicht ermittelbar. Die Schnittstellen erlauben auch das automatisierte

2.2 Aufbau und Zugänglichkeit grafischer Oberflächen

Interagieren mit den Oberflächen, also beispielsweise Klick auf eine Schaltfläche oder Eingabe eines Textes in ein Eingabefeld. Eine Veränderung der Oberflächen ist allerdings nicht möglich. Außerdem besteht auch kein Zugriff auf Informationen über dargestellte Dokumente. Die Accessibility-Schnittstellen müssen also zur vollständigen Erfassung der grafischen Oberfläche immer mit anderen Schnittstellen kombiniert werden.

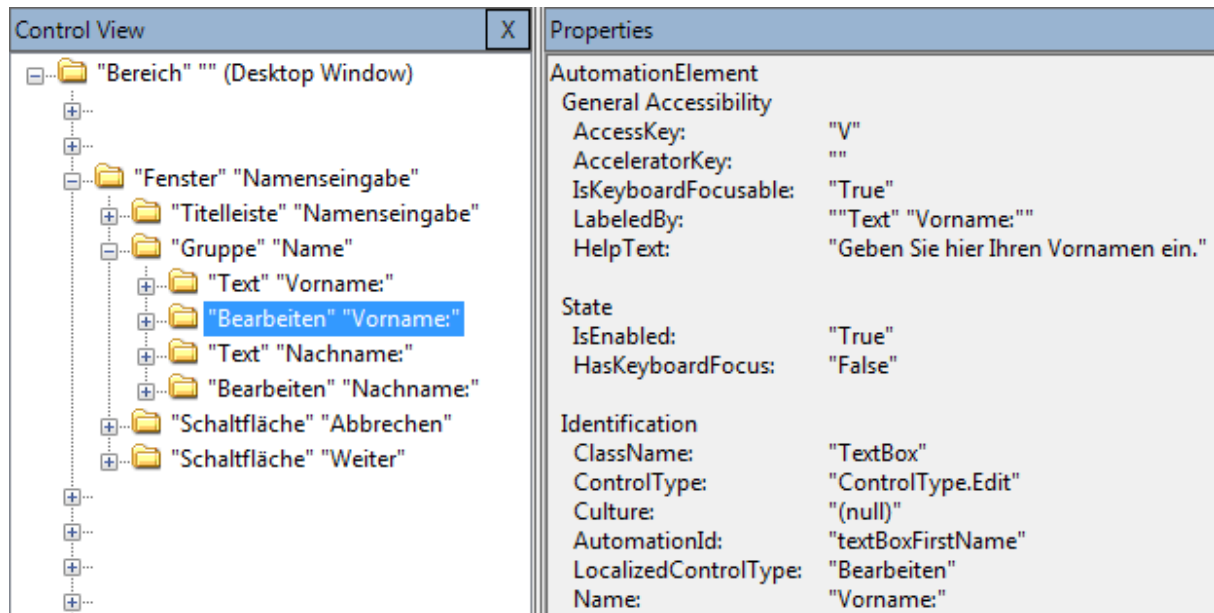


Abbildung 21 UIAutomation-Baum des XAML-Dialogs aus Abbildung 20

Bei korrekter Implementierung der Unterstützung für UIA-Schnittstellen weist der XUL-Dialog den gleichen UIAutomation-Baum auf.

Der Zugang über Schnittstellen wie DOM und UIA spielt sich auf einer Ebene ab, auf der kaum Informationen über den Darstellungsprozess zur Verfügung stehen. Hier sind zum größten Teil nur semantische und strukturelle Informationen verfügbar, wie die Typen von Bedienelementen und deren Verbindung zu anderen Elementen. Zwar kann auch auf Informationen über die grafische Ausgabe zugegriffen werden, dies aber nur in sehr beschränktem Maße. So werden meist Informationen zur Position und Größe der Elemente bereitgestellt und auch Angaben zur farblichen Gestaltung oder zur Gestaltung von Schrift sind auslesbar. Aber all diese Informationen lassen keinen eindeutigen Schluss über die tatsächliche Darstellung am Bildschirm zu. Beispielsweise kann daraus nicht abgelesen werden, ob eine Schaltfläche rechteckig gezeichnet wird oder als Ellipse.

Neben den zuvor besprochenen Zugangsmöglichkeiten existieren natürlich je nach Betriebssystem, Anwendung und Dokument noch verschiedenste weitere Zugangswege und Informationsquellen. Beispielsweise können Informationen über den Inhalt von Rastergrafiken teilweise in Eigenschaftsfeldern des jeweiligen Bildformates abgelegt werden. So unterstützt das JPEG-Format (JPEG = Joint Photographic Experts Group) die Speicherung von Metadaten nach dem IPTC-Standard (IPTC = International Press Telecommunications Council) [Int09], die auch Felder zur Beschreibung des Bildinhalts vorsehen. Grafiken im PNG-Format (PNG = Portable Network Graphics) können in Datenblöcken, die mit dem Schlüsselwort „tEXt“ ge-

kennzeichnet sind, beliebige Textinformationen speichern [Wore]. Zum Zugriff auf diese Daten müssen natürlich entsprechende Implementierungen umgesetzt werden, die sehr verschieden ausfallen können. Außerdem muss auf die Datei selbst zugegriffen werden können. Wurde die Grafik beispielsweise in ein Dokument oder ein Bedienelement eingebunden, ist dies meist nicht möglich. Gerade die Verfügbarkeit von Originaldateien ist ein weiterer wichtiger Aspekt für den Zugang zu Dokumenten. Wurden beispielsweise PDF-Dokumente mit Hilfe von LaTeX erzeugt, kann es oft hilfreich sein, Zugriff auf die LaTeX-Quelldateien zu haben, da bei der Umwandlung in PDF Informationen verloren gehen können oder auch einfache Texte aufgrund spezieller Formatierungen unzugänglich werden können. Ein einfaches Beispiel dafür ist die Darstellung von Formeln. In der linearen LaTeX-Darstellung sind diese auf einer Braillezeile leichter zu erfassen, als in ihrer zweidimensionalen und häufig grafischen Darstellung im PDF.

Viele der beschriebenen Schnittstellen werden von Screenreadern und Bildschirmvergrößerungssoftware in der einen oder anderen Weise bereits erfolgreich genutzt. Hierbei spielen besonders die semantischen Informationen eine wichtige Rolle. Wie gut diese Informationen über die verschiedenen Schnittstellen ermittelbar sind, hängt dabei nicht nur von der Zugangsmöglichkeit an sich ab, sondern in entscheidendem Maße auch davon, ob und wie diese Informationen überhaupt digital zu Verfügung gestellt werden.

2.2.3 Semantische Informationen in grafischen Oberflächen

Wie bereits erwähnt sind semantische Informationen ein wichtiges Mittel, um den Zugang zu grafischen Oberflächen für Blinde und Sehbehinderte zu erleichtern. Sie dienen zur Anpassung und Beschreibung der Darstellung oder zur Unterstützung der Interaktion. Auf semantische Informationen kann über verschiedene Zugangsmöglichkeiten zugegriffen werden (siehe Abschnitt 2.2.2). Genauso unterschiedlich sind auch die Möglichkeiten, diese Informationen für den Computer zugänglich abzulegen.

Zu den semantischen Informationen gehören im Rahmen dieser Arbeit jegliche Informationen, die über die Rastergrafik-Information, also die zweidimensionale Matrix von Farbwerten hinausgehen. Auch für Normalsichtige ist die rasterisierte Ausgabe am Bildschirm, neben den Änderungen, die ihr aufgrund von Interaktionen oder Systemereignissen widerfahren, die einzige Informationsquelle. Bei der Betrachtung der Bildschirmausgabe findet ein umfangreicher, mehrstufiger Erkennungsprozess statt, bei dem eigentlich sinnfreien Bildpunkten eine komplexe Semantik zugeordnet wird. Erst dieser Prozess ermöglicht die Arbeit mit der Bildschirmausgabe bzw. mit den Anwendungen, deren grafische Darstellung am Bildschirm zu sehen ist. Die einzelnen Bildpunkte können von Normalsichtigen aufgrund ihrer farblichen Gestaltung und durch die Erfahrung des Betrachters meist leicht zu sinntragenden Informationseinheiten, wie etwa einem Buchstaben oder einer Schaltfläche kombiniert werden. Aufgrund von Beziehungen zwischen mehreren Informationseinheiten (wie beispielsweise Nähe und Ähnlichkeit) werden schnell neue, größere Informationseinheiten und damit ein größerer Sinnzusammenhang erkannt. So werden beispielsweise mehrere größer ge-

schriebene Buchstaben, denen Zahlen vorangestellt sind, als Überschrift identifiziert oder eine Reihe von Schaltflächen am unteren Rand des Bildschirms als Taskleiste. Auch der Zustand von Bedienelementen (wie etwa Schaltflächen) ist durch deren farbliche Gestaltung erkennbar. Dabei kann der Mensch sehr gut mit Unschärfen umgehen wie etwa unterschiedlichen Größen oder gar Verzerrungen und Änderungen in der gesamten Farbgebung, sowie Licht- und Schatteneffekten.

Kann dieser Erkennungsprozess nicht oder nur eingeschränkt stattfinden, wie es bei Blinden und Sehbehinderten der Fall ist, so muss die Bedeutung der dargestellten Bildpunkte auf andere Weise ermittelt werden. Nur so ist es möglich, den Inhalt auch in einer für den Benutzer angepassten Darstellung korrekt widerzugeben. Dabei kann nicht davon ausgegangen werden, dass eine in jedem Fall zuverlässige automatisierte Erkennung einer fertigen Bildschirmausgabe möglich ist. Der vom Menschen durchgeführte Erkennungsprozess arbeitet mit komplexen Regeln, die durch jahrelange Erfahrung aufgebaut wurden und viele Abhängigkeiten aufweisen. Hinzukommt, dass sich die Bedeutung der Darstellung, auch für Menschen in einigen Fällen erst durch die Interaktion mit dieser erschließen lässt. Ein einfaches Beispiel ist hier die Darstellung von Screenshots. Wird beispielsweise das Abbild eines Dialogs in einem Bildbetrachtungsprogramm dargestellt und dabei nicht skaliert oder anderweitig verändert, so kann dies allein durch die statische Bildinformation nicht von dem tatsächlichen Dialog unterschieden werden. Auch Menschen fallen dieser Täuschung häufig zum Opfer und versuchen den dargestellten Dialog zu bedienen. Da Maus- oder Tastaturinteraktion in diesem Fall aber nicht die, aufgrund von Erfahrungen aufgebaute, erwartete Änderung in der visuellen Ausgabe erzeugen, wird der Fehler meist schnell erkannt. Dieser Prozess ist automatisiert nur mit sehr hohem Aufwand und eventuell unerwünschten Nebeneffekten nachstellbar. So müsste beispielsweise mindestens der Mauszeiger über ein Bedienelement einer Oberfläche bewegt werden, um eine Änderung in der visuellen Darstellung des Elements oder des Mauszeigers zu erreichen, sofern das Bedienelement tatsächlich eines ist. Manchmal ist aber auch dann noch nicht erkennbar, welche Aktion durch die Bedienung ausgelöst wird. Um dies zu erfahren, muss die Bedienung dann tatsächlich durchgeführt werden, was natürlich Einfluss auf die dargestellte Anwendung haben kann. Zwar muss auch der Mensch eine Programmoberfläche evtl. auf diese Weise erkunden, allerdings erweitert er damit auch seine Erfahrungen kontinuierlich und muss so nicht jedes neue Bedienelement erst durch Interaktion analysieren. Zur korrekten automatischen Erkennung der Semantik der Bildschirmausgabe müsste letztlich eine künstliche Intelligenz aufgebaut werden, die dem Menschen ebenbürtig ist. Da dies (zumindest in naher Zukunft) nicht realisiert werden kann, muss die Semantik, die hinter der grafischen Darstellung steckt, aus anderen Informationen als den Bildpunkten gewonnen werden.

Zu diesen semantischen Informationen gehören strukturelle Informationen, beschreibende Informationen, Informationen über die grafische Gestaltung und Informationen zu Interaktionsmöglichkeiten (siehe Tabelle 1). Strukturelle Informationen umfassen Informationen über die Beziehungen zwischen Elementen (wie zwischen Beschriftungen und Eingabefeldern), Gruppierungen von Elementen, Elementtypen und den hierarchischen Aufbau von

Anwendungen und Dokumenten. Zu den beschreibenden Informationen zählen Hinweise zum Inhalt eines Dokuments oder zum Nutzen einer Anwendung, Benennungen von Bedienelementen (z.B. über „Name“- oder „ID“-Attribute), sowie Tooltip-Texte oder andere Hinweise zur Bedeutung eines Bedienelements. Auch Informationen wie Autor und Erstellungsdatum sind beschreibende Informationen, die allerdings für die Zugänglichkeit eher weniger interessant sind. Informationen über die grafische Gestaltung umfassen vor allem die Position und Größe von Elementen, sowie farbliche Gestaltung und Schriftattribute. Zu den Informationen über Interaktionsmöglichkeiten zählen Short-Cuts (wie z.B. Strg+O zur Anzeige eines Öffnen-Dialogs) und Access-Keys (z.B. Alt+D zum Öffnen des Menüs Datei, gefolgt von ‚F‘ zur Anzeige eines Öffnen-Dialogs), aber auch Angaben zu Verknüpfungen mit Funktionen, wie es häufig bei HTML-Bedienelementen mit JavaScript-Funktionen zu sehen ist.

Semantische Informationen			
Strukturelle Informationen	Beschreibende Informationen	Inf. über die grafische Gestaltung	Inf. zu Interaktionsmöglichkeiten
<ul style="list-style-type: none"> - Element-hierarchie - Beziehungen zwischen Elementen - Gruppierungen - Elementtypen - ... 	<ul style="list-style-type: none"> - Hinweise zum Inhalt eines Dokuments - Hinweise zum Nutzen einer Anwendung - Benennung von Elementen - Tooltip-Texte - Autor - Erstellungsdatum - ... 	<ul style="list-style-type: none"> - Position und Größe von Elementen - Farbliche Gestaltung - Schriftattribute - ... 	<ul style="list-style-type: none"> - Short-Cuts (z. B. Strg+O) - Access-Keys (z. B. Alt+D,F) - Angaben zur Verknüpfung mit Funktionen - ...

Tabelle 1 Übersicht zu semantischen Informationen

Viele Formate grafischer Darstellungen bieten Möglichkeiten, entsprechende Informationen zu hinterlegen. So können, wie bereits in Abschnitt 2.2.2 erwähnt, bei einigen Rastergrafik-Formaten beschreibende Informationen in den Grafikdateien gespeichert werden, beispielsweise nach dem IPTC-Standard [Int09]. Strukturelle Informationen können allerdings nicht abgelegt werden. Häufig können diese für Rastergrafiken aber auch nicht oder nur mit unvermeidbar hohem Aufwand angegeben werden. Bei Fotos beispielsweise müssten alle abgebildeten Objekte manuell markiert, klassifiziert und gruppiert werden.

Bei textuellen Dokumenten ist dies anders. Strukturelle Informationen können und sollten hier während des Erstellungsprozesses immer eingepflegt werden. Dass strukturelle Informationen hier immer vorhanden sind, erkennt man beispielsweise an einfachen Textdokumenten, in denen verschiedene Sonderzeichen und Einrückungen verwendet werden, um Überschriften hervorzuheben, Aufzählungen zu definieren oder Dokumentabschnitte voneinander abzutrennen. Um diese Informationen besser darstellen und damit auch verarbeiten zu können, wurden strukturierte Formate wie beispielsweise HTML (Hypertext Markup Lan-

guage) [Worc] oder auch die verschiedenen Formate für Office-Dokumente entwickelt. Hier können Überschriften und verschiedene Textabschnitte mit entsprechenden Markierungen ausgezeichnet werden. Auch beschreibende Elemente können in HTML in Form von IDs (Identifikatoren), Namensattributen, Metadaten und Kommentaren hinterlegt werden. Hinweise zu einer vom Standardverhalten abweichenden grafischen Gestaltung können mit Hilfe spezieller Formatierungssymbole wie `<i>` für kursiv (engl. italic) oder `` für fett (engl. bold) oder auch über CSS (siehe Abschnitt 2.2.2) definiert werden.

In Bedienoberflächen können semantische Informationen ähnlich abgelegt werden, wie in strukturierten Dokumenten. Auch hier existieren immer semantische Informationen, die dargestellt werden können und sollten. Bedienelemente werden typisiert und können durch Panel- oder Group-Elemente gruppiert werden. Beschriftungen und Eingabefelder sind in modernen Formaten für Bedienoberflächen häufig über entsprechende Attribute verknüpfbar. Zudem kann meist zu jedem Element der Oberfläche ein Name und oft auch eine Beschreibung (wenn auch nur in Form eines Tooltips) angegeben werden. Die Quellcode-Ausschnitte in Listing 3 und Listing 4 zeigen Beispiele dazu.

Höhere Grafik-Formate wie SVG (Scalable Vector Graphics) bieten ebenfalls Möglichkeiten zur Strukturierung, Verknüpfung und Erläuterung von Elementen. Die grafische Darstellung wird in Vektorgrafiken immer über Stilelemente angegeben, da sie essentieller Teil der Grafik ist. Allein durch die Verwendung grafischer Primitive bringen die heute zur Erstellung komplexerer Grafiken meist genutzten Vektorgrafik-Formate einen wesentlichen Vorteil gegenüber Rastergrafiken, da so die einzelnen Bestandteile der Grafik mit ihren visuellen Effekten separiert sind. Das mittlerweile immer stärker verwendete 2D-Vektorgrafikformat SVG wurde mit Bedacht auf Zugänglichkeit ausgelegt. Zum einen ist es ein offenes XML-basiertes Format, dass somit Zugriff über die Schnittstellen des Document Object Model (DOM) [Worb] und Gestaltung über CSS (Cascading Style Sheets) [Wora] ermöglicht. Zum anderen wurden Annotationsmöglichkeiten für jedes Element vorgesehen. Auch Wiederverwendung von Grafikteilen wird explizit ermöglicht. Und natürlich wurden Gruppierungselemente definiert. Listing 5 zeigt eine durch Gruppierungen (`<g>`) und Annotationen (mit `id`, `<desc>` und `<title>`) verbesserte Version von Listing 1.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" width="320" height="400"
  viewBox="0 0 40 50" stroke-linecap="round" stroke-linejoin="round">
  <title>Hoch-Pfeil</title>
  <desc>Ein roter Pfeil mit der Beschriftung HOCH.</desc>
  <g id="pfeil">
    <g id="spitze">
      <title id="spitzeTitel">Pfeilspitze</title>
      <desc id="spitzeDesc">Die Spitze des Pfeils.</desc>
      <path stroke="red" fill="red" stroke-width="5"
        d="M 10 20 l 20 0 l -10 -15 z" id="dreieck">
        <title id="dreieckTitel">spitzes Dreieck</title>
        <desc id="dreieckDesc">Die Spitze des Pfeils ist ein spitzes,
          gleichschenkliges Dreieck mit abgerundeten Ecken.</desc>
      </path>
    </g>
    <g id="rumpf">
      <title id="rumpfTitel">Pfeilrumpf</title>
      <desc id="rumpfDesc">Der Rumpf des Pfeils.</desc>
      <line stroke="#FF0000" stroke-width="10" x1="20"
        y1="20" x2="20" y2="40" id="rumpfLinie">
        <title>vertikale Linie</title>
        <desc>Der Rumpf des Pfeils ist eine dicke vertikale Linie mit
          abgerundeten Linienende.</desc>
      </line>
    </g>
  </g>
  <text id="hochText" x="12" y="20"
    style="font: bold 4pt Arial; baseline-shift:-4pt">HOCH</text>
</svg>
```

Listing 5 SVG-Quellcode zu Abbildung 16 mit Gruppierungen (*g*) und Metadaten (*title*, *desc* und *id*) Erweiterung von Listing 1

Wie in Listing 5 zu sehen ist, existieren unterschiedliche Ebenen von semantischen Informationen. So ist für das Objekt (den Pfad), das die Pfeilspitze bildet, einerseits interessant, dass es sich um die Spitze des Pfeils handelt. Andererseits sagt dies aber nichts darüber aus, wie eine Spitze aussieht. Für Menschen, die noch nie eine Pfeilspitze gesehen haben, kann deshalb auch die einfache Information, dass es sich um ein Dreieck handelt, wertvoll sein. Sofern wie bei SVG die Möglichkeit besteht, semantische Informationen auf mehreren Ebenen zu hinterlegen, sollte dies auch genutzt werden, denn auch der menschliche Erkennungsprozess findet in mehreren Stufen statt. Verschiedene semantische Ebenen bilden dies nach und erleichtern dadurch das Verstehen einer Grafik, wenn der Erkennungsprozess gestört ist.

Aufgrund ihrer Flexibilität werden Vektorgrafikformate heute auch oft als Grundlage für grafische Benutzungsoberflächen genutzt. Formate wie SVG und Flash bieten die Möglichkeit Interaktionen einzubinden. Auch bei XAML verschwimmen die Grenzen zwischen Grafik und Benutzungsschnittstelle. Designer können so Interaktionselemente beliebig gestalten und

sind nicht auf die Vorgaben von Toolkits angewiesen. Für die Zugänglichkeit bringt dies aber oft Probleme mit sich, da beispielsweise keine Typisierung der Bedienelemente angegeben ist und auch nicht immer klar erkennbar ist, welches die Beschriftung für ein Element ist. Die Accessible Rich Internet Applications (ARIA) Suite der Web Accessibility Initiative (WAI) des World Wide Web Consortium (W3C) soll hier Verbesserungen bringen [Webc]. Durch ARIA definiert Attribute, die Auskunft über die Bedeutung eines Elements als Bedienelement und dessen Interaktionsmöglichkeiten geben. So existieren beispielsweise verschiedene Rollen, wie „menu“ und „treeitem“, und verschiedene Zustände wie „haspopup“ für Menüs mit Untermenüs. Mit Hilfe dieser Attribute können die zuvor rein grafisch definierten Bedienelemente für Schnittstellen wie UIAutomation sinnvoll zugänglich gemacht werden. So kann beispielsweise der dem SVG-Pfeil aus Listing 5 mit Hilfe der Angabe „role='button'“ in der Eigenschaftsliste der Pfeil-Gruppe als Schaltfläche gekennzeichnet werden, wenn er denn als solche eingesetzt wird.

2.2.4 Zusammenfassung

Wie dieses Kapitel zeigte, können grafische Oberflächen sehr komplex sein und aus verschiedensten Elementen bestehen. Dokumente, Grafiken und Bedienelemente gehen fließend ineinander über. Aus der grafischen Ausgabe am Bildschirm allein können hierfür nur schlecht Darstellungen abgeleitet werden, die Blinden und Sehbehinderten ein effizientes Arbeiten ermöglichen. Aus diesem Grund wurden mit Accessibility-Schnittstellen andere Zugangsmöglichkeiten entwickelt, die Zugriff auf semantische Informationen bieten. Kombiniert mit den für Add-Ins und ähnliches vorhandenen Schnittstellen können diese für mächtige Werkzeuge genutzt werden. Entscheidend ist aber, dass die entsprechenden Informationen auch in Dokumenten und Oberflächen hinterlegt sind. Hierfür bieten die verschiedenen Technologien, die zur Gestaltung von grafischen Oberflächen genutzt werden, verschiedenste Möglichkeiten. Nur wenn diese in der richtigen Weise eingesetzt werden, können Hilfsttechnologien wie Screenreader semantische Informationen auch effektiv nutzen.

2.3 Richtlinien zur zugänglichen Gestaltung von grafischen Oberflächen

Zugängliche Gestaltung von grafischen Oberflächen ist die wesentliche Voraussetzung, um Blinden und Sehbehinderten einen effizienten Zugang zu modernen Anwendungen und Informationsquellen zu gewährleisten. Dabei bedeutet „zugängliche Gestaltung“ eine Gestaltung in der Art, dass die grafische Darstellung von vorn herein gut erkennbar aber auch leicht an spezielle Bedürfnisse anpassbar ist, keine wichtigen Informationen durch eine Anpassung der grafischen Darstellung verloren gehen und alle wichtigen Informationen auch ohne Zugang zur grafischen Ausgabe, also über andere Schnittstellen als die Bildinformationen, ermittelt werden können.

Viele Webseiten, deren Zielgruppen primär Blinde und Sehbehinderte sind (wie beispielsweise [Rai10], [Gra10] und [bsb] in Abbildung 22), sind deutlich anders gestaltet als eine durchschnittliche Webseite, ein Hinweis darauf, dass Webseiten nicht ohne weiteres gut zugänglich sind. Viele Gestaltungselemente bergen Probleme.

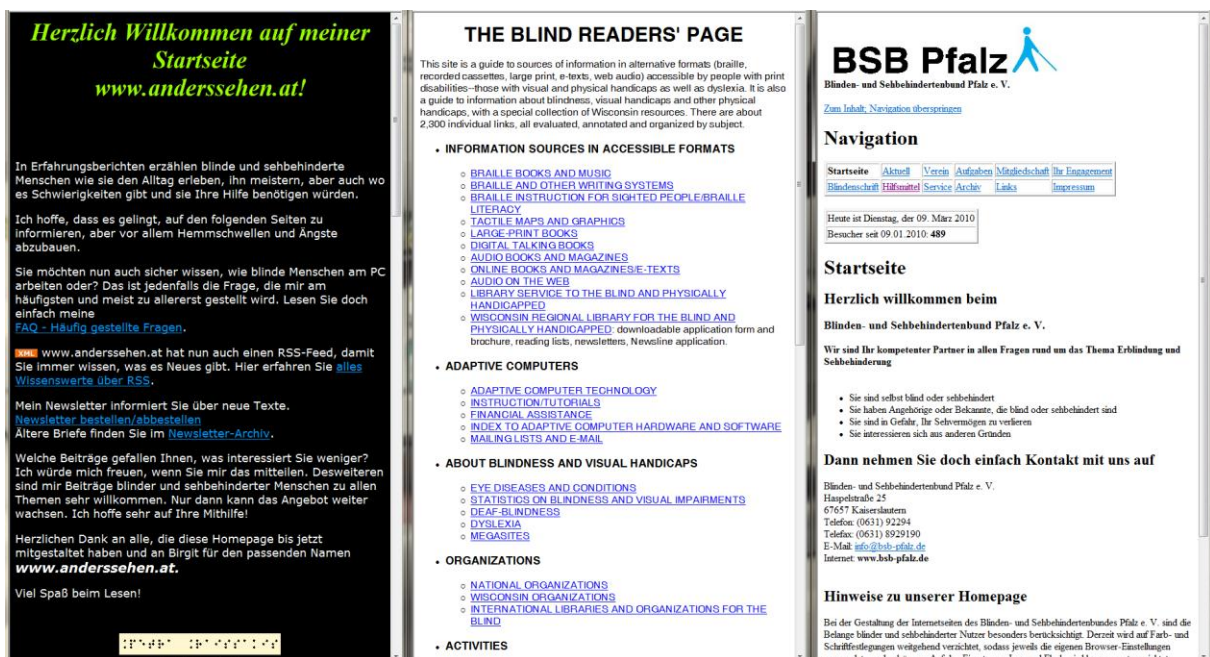


Abbildung 22 Beispiele für Webseiten mit den Zielgruppen Blinde und Sehbehinderte
Quellen: [Rai10], [Gra10] und [bsb]

Da Webseiten heute ein wesentliches Informationsmittel für das tägliche Leben sind und somit allen zugänglich sein sollten, gibt es besonders in diesem Bereich viele Initiativen, um Barrieren abzubauen bzw. zu verhindern. Die bekannteste davon ist sicher die Web Accessibility Initiative (WAI) des World Wide Web Consortiums (W3C) [Webd], aus der die Web Content Accessibility Guidelines (WCAG) [Webe] hervorgegangen sind, die Richtlinien zur Gestaltung von Webseiten geben. Die meisten dieser Richtlinien sind auch auf andere Dokumente und auf Bedienoberflächen übertragbar. Die WCAG sind nach den vier Grundsätzen „Wahrnehmbar“ (Perceivable), „Bedienbar“ (Operable), „Verständlich“ (Understandable)

2.3 Richtlinien zur zugänglichen Gestaltung von grafischen Oberflächen

und „Robust“ (Robust) gegliedert. Zu jedem der vier Grundsätze sind spezielle Richtlinien definiert (insgesamt 12). Zu jeder Richtlinie existieren Erfolgskriterien, an denen gemessen werden kann, wie gut eine Richtlinie erfüllt wurde. Die Erfolgskriterien sind in die drei Stufen A (niedrigste Stufe), AA und AAA (höchste Stufe) unterteilt. Erfolgskriterien der Stufe A können mit relativ geringem Aufwand erreicht werden. Erfolgskriterien der Stufe AAA sind teilweise nur sehr aufwändig umsetzbar, wie beispielsweise die Bereitstellung von Gebärdenvideos zu jedem Audio-Inhalt. Zu jedem der Erfolgskriterien werden in einem Zusatzdokument Beispiele und Technologien präsentiert, die zur Erreichung des Kriteriums führen können. Neben den WCAG stellt das W3C auch Richtlinien zur Gestaltung von End-Nutzer-Clients im und für das Web bereit, die User Agent Accessibility Guidelines (UAAG) [Webb], sowie mit den Authoring Tool Accessibility Guidelines (ATAG) [Weba] auch zur Gestaltung von Autorenwerkzeugen zur Erstellung von Webinhalten.

Die zugängliche Gestaltung grafischer Oberflächen findet heute nicht mehr nur auf freiwilliger Basis statt, sondern wird teilweise gesetzlich verlangt. So bestimmt das bundesdeutsche „Gesetz zur Gleichstellung behinderter Menschen“ (Behindertengleichstellungsgesetz - BGG) [bgg] in § 11 „Barrierefreie Informationstechnik“ folgendes:

- (1) Träger öffentlicher Gewalt [...] gestalten ihre Internetauftritte [...], sowie die von ihnen zur Verfügung gestellten grafischen Programmoberflächen [...] so, dass sie von behinderten Menschen grundsätzlich uneingeschränkt genutzt werden können. [...]
- (2) Die Bundesregierung wirkt darauf hin, dass auch gewerbsmäßige Anbieter von Internetseiten sowie von grafischen Programmoberflächen [...] ihre Produkte entsprechend den technischen Standards nach Absatz 1 gestalten.

Dem BGG ähnliche Gesetze existieren auch in anderen Staaten. In den USA ist dies beispielsweise der „Rehabilitation Act“. Darin regelt Abschnitt 508 (Section 508, siehe [sec]) die Zugänglichkeit von Informationstechnik. Auch wenn dies sich, wie das BGG, nicht direkt auf die Angebote kommerzieller Unternehmen bezieht, so führte gerade das US-amerikanische Gesetz zu einer deutlichen Steigerung der Aktivitäten zur zugänglichen Gestaltung, da sämtliche öffentlichen Träger nur dann eine Software einsetzen dürfen, wenn diese auch zugänglich ist. Unternehmen können also nur zugängliche Software an öffentliche Träger verkaufen. Dies ist ein bedeutender Markt. Das BGG dagegen bezieht sich nur auf Software und Angebote, die von den öffentlichen Trägern selbst kommen.

Zusätzlich zu den gesetzlichen Festlegungen in Section 508 wurden direkt auch Standards definiert, die zeigen, wie die Forderungen aus Section 508 erfüllt werden können, die sogenannten „Section 508 Standards“. Diese enthalten Vorgaben zu einem Großteil der heute bekannten Informationstechniksysteme. Ein ähnliches Dokument existiert auch zum BGG. Dies ist die „Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz“ (Barrierefreie Informationstechnik-Verordnung - BITV) [bit]. Bei der Erstellung orientierte man sich an den WCAG in der Version 1.0. Die BITV definiert 14 Anforderungen mit zugehörigen Bedingungen, die in zwei Prioritäten eingestuft sind. Dabei

sind Bedingungen mit der Priorität 1 unbedingt umzusetzen, um grundsätzlich einen behindertengerechten Zugang zu Angeboten der Informationstechnik zu ermöglichen.

Da die zugängliche Gestaltung grafischer Oberflächen ein eher junges Arbeitsfeld ist und sich die zugrundeliegenden Technologien schnell verändern, befinden sich natürlich auch die entsprechenden Richtlinien in ständiger Entwicklung. Häufig definieren Softwarehersteller und anderer Firmen abseits von den anerkannten Standards eigene Richtlinien, die sich eventuell konkreter auf spezielle bzw. die eigenen Technologien beziehen, wie beispielsweise die „Developer guidelines“ aus dem IBM „Human Ability and Accessibility Center“ [IBM] oder die Richtlinien von Apple zur Erstellung zugänglicher iPhone-Anwendungen [App]. All diese Richtlinien wie auch die allgemein anerkannten Vorgaben sollten immer kritisch betrachtet werden. Nicht immer sind Richtlinien sinnvoll oder sinnvoll definiert. So verlangen die WCAG beispielsweise in Richtlinie 1.4 „Unterscheidbar“, dass Texte auf bis zu 200 Prozent vergrößert werden können, ohne dabei die Funktionalität der Webseite einzuschränken. Sie definieren aber nirgends eine Mindestschriftgröße.

2.4 Zusammenfassung

In diesem Kapitel wurde gezeigt, dass Blinde und Sehbehinderte heute durchaus über Mittel verfügen können, die die Arbeit mit grafischen Oberflächen ermöglicht. Es bestehen verschiedene Möglichkeiten, die zur Arbeit im Wesentlichen nötigen Informationen aus grafischen Oberflächen programmatisch auszulesen und damit angepasste Darstellungen zu erzeugen und erleichternde Navigationsmöglichkeiten anzubieten. Hilfstechnologien nutzen diese Möglichkeiten heute bereits vielfältig aus. Allerdings stoßen Blinde und Sehbehinderte noch auf vielfältige Barrieren, die den Zugang zu manchen Informationen ganz verschließen. Durch die immer komplexer werdenden grafischen Darstellungen steigen diese Probleme zunehmend an. Zwar existieren Richtlinien zur zugänglichen Gestaltung, die dem entgegenwirken sollen. Diese werden allerdings in vielen Fällen noch nicht beachtet.

Für blinde Computernutzer ist besonders das Fehlen des flächigen Zugangs zu Dokumenten problematisch. Tabellen können nur mit hohem Aufwand und nur bei richtiger Gestaltung der Tabelle gut bearbeitet werden. Der Mehrwert einer Grafik zur Visualisierung von Zusammenhängen ist nicht erfassbar. Auch Tätigkeiten wie das Umstellen einer mathematischen Formel sind durch die einzeilige Darstellung auf der Braillezeile schwerer zu bewältigen als für Normalsichtige. Grafisch-taktile Displays, also flächige, interaktive Ausgabegeräte, können hier einen wesentlichen Mehrwert bringen, werden aber von aktuellen Screenreadern noch nicht unterstützt.

Bei sehbehinderten Computernutzern stand lange Zeit nur die grafische Vergrößerung der Bildschirmausgabe im Vordergrund. Unterstützende Navigationsmöglichkeiten und zusätzliche Darstellungsmöglichkeiten werden erst seit kurzem und nur in geringem Maße angeboten. Insgesamt erfolgt hier noch wenig Nutzung der verschiedenen Zugangsmöglichkeiten zu grafischen Oberflächen. Funktionalitäten, die die Anwendungen oder Formate selbst bereit-

stellen und für Sehbehinderte nützlich sein könnten, werden häufig nicht genutzt oder gar außer Kraft gesetzt. Die gebotene Darstellung basiert immer auf der Bildschirmausgabe. Eine völlige Umgestaltung zur Erzeugung effizienterer Darstellungen erfolgt nicht.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Aufgrund der unterschiedlichen Charakteristika heutiger Hilfsmittel für Blinde und Sehbehinderte – die lineare und textuelle Braillezeilen- und Sprachausgabe und im Gegensatz dazu die zweidimensionale, grafische, vergrößerte Bildschirmdarstellung – mag zunächst seltsam erscheinen, beide Gruppen in einer gemeinsamen Arbeit zu betrachten. Durch die Entwicklungen auf dem Gebiet der grafisch-taktilen Displays, die für die Nutzergruppe der Blinden in dieser Arbeit vornehmlich betrachtet werden, wachsen aber beide Welten und ihre Anforderungen an angepasste Darstellungen immer mehr zusammen. Dabei bezieht sich diese Arbeit bezüglich der Vergrößerung auf Standardmonitore, also keine übergroßen Monitore oder besonders kleinen Displays wie bei Handys. Bezüglich der grafisch-taktilen Ausgabe wird von großflächigen Displays, wie dem „BrailleDis 9000“ der Firma Metec ausgegangen, da nur mit solchen Displays überhaupt ein Überblick über komplexere Darstellungen möglich ist. Das „BrailleDis 9000“ bietet eine monochrome taktile Darstellung mit einem auf Braille-Darstellung optimierten aber gleichmäßigen Stiftabstand. Das Setzen aller Stifte dauert 50 ms. Daher sind auch animierte und blinkende Darstellungen gut möglich. Das Display ist durch Eingabemodule, die sich über jeweils 2×5 Stifte erstrecken Multitouch-fähig.



Abbildung 23 Foto des BrailleDis 9000

Zu Beginn dieses Kapitels wird in Abschnitt 3.1 die Gemeinsamkeit der nötigen Anpassungen von Darstellungen für Blinde und Sehbehinderte im Allgemeinen und an konkreten Anforderungen verdeutlicht. Defizite aktueller Hilfsmittel werden aufgezeigt. Danach stellt Abschnitt 3.2 verschiedene konzeptionelle Möglichkeiten vor, diese Anforderungen zu erfüllen und diskutiert deren Vor- und Nachteile. Abschnitt 3.3 beschreibt Umsetzungen zu den vorgestellten Konzepten, die im Rahmen dieser Arbeit entwickelt wurden.

Die für das „BrailleDis 9000“ entwickelten Konzepte und Umsetzungen können natürlich leicht auch auf ähnliche grafisch-taktile Displays übertragen werden.

3.1 Gleichartigkeit angepasster Darstellungen für beide Zielgruppen

3.1.1 Gemeinsame Grundfragestellung

Eine wesentliche Anforderung bei der Darstellung grafischer Oberflächen ist die Lesbarkeit von textuellen Elementen. Für Blinde bedeutet dies die Darstellung in Punktschrift, für Sehbehinderte unter anderem die Nutzung einer bestimmten Schriftgröße. Das grafisch-taktile Display „BrailleDis 9000“ besitzt eine Auflösung von 120 × 60 Punkten, die vollständig für Brailleschrift mit den durch Braillezeilen gewohnten Maßen genutzt werden können. Damit können in 6-Punkt-Brailleschrift, der kleinstmöglichen Schrift, mit Beachtung der nötigen Leerräume zwischen Zeichen und Zeilen maximal 15 Zeilen Text mit je 40 Zeichen dargestellt werden. Bei 8-Punkt-Brailleschrift sind nur 12 Zeilen zu 40 Zeichen möglich. Ähnliche Rahmenbedingungen ergeben sich auch bei Nutzung der Schriftgröße 60 pt am Bildschirm, was bei der in Dokumenten zumeist genutzten Schriftgröße von 12 pt einer 5-fachen Vergrößerung entspricht. Wie Abbildung 24 zeigt lassen sich so auf einem Bildschirm mit einer Auflösung von 1920 × 1200 Pixeln 12 Zeilen Text mit je 43 Zeichen darstellen.

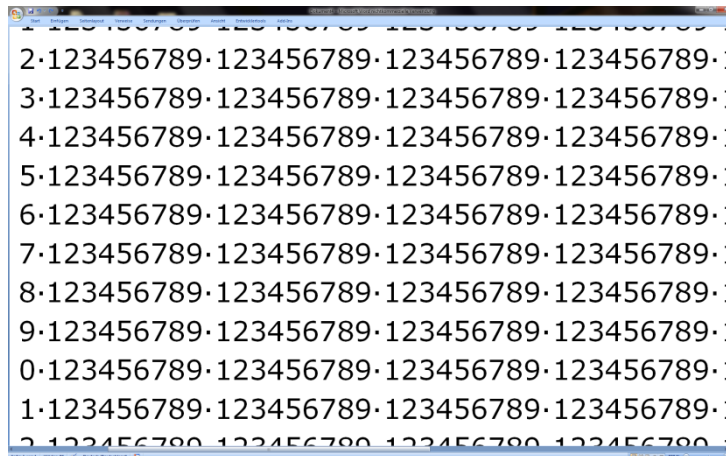


Abbildung 24 Anzahl der darstellbaren Zeichen bei Schriftgröße 60 pt bzw. 5-facher Vergrößerung der in Dokumenten meist verwendeten Standardschriftgröße 12 pt

Hierdurch zeigt sich, dass die grundsätzliche Fragestellung bei der Darstellung grafischer Oberflächen für Blinde (auf einem grafisch-taktilen Display) und Sehbehinderte (am Monitor) dieselbe ist: Wie kann die komplexe Bildschirmausgabe der Anwendungen auf eine im Verhältnis zur nötigen Schriftgröße kleinen Ausgabefläche gut zugänglich gemacht werden?⁸

⁸ Neben der Abbildung auf Monitor oder ein grafisch-taktilen Display ist für Blinde und Sehbehinderte natürlich auch die Sprachausgabe ein wichtiges Hilfsmittel für die Erschließung grafischer Oberflächen. Sie wurde aber im Rahmen dieser Arbeit nicht näher behandelt.

3.1.2 Gemeinsamkeiten in konkreten Anforderungen

Die in Abschnitt 2.1.4 diskutierten Hilfsmittel zur Anpassung der Bildschirmausgabe für Sehbehinderte zeigen bereits verschiedene Lösungsansätze für die in Abschnitt 3.1.1 genannte Grundfragestellung. Aus den Funktionalitäten und Defiziten dieser Hilfsmittel lassen sich wichtige Anforderungen an angepasste Darstellungen ableiten. Auch Gestaltungsvorschriften für taktile Grafiken und die von Screenreadern etablierten Arbeitstechniken, sowie die bisher für grafisch-taktile Displays entwickelten Anwendungen geben Hinweise dazu. Im Folgenden werden die wichtigsten Anforderungen an Darstellungen, die Blinden und Sehbehinderten ein effizientes Arbeiten mit grafischen Oberflächen aber auch Zusammenarbeit mit Normalsichtigen ermöglichen, beschrieben. Diese sind:

1. Anpassung der Darstellung von Bedienelementen und Dokumenten
2. Gemeinsame Darstellung logisch zusammenhängender Informationen
3. Darstellung mit einer bestimmten Schriftgröße
4. Wenig horizontales Scrollen beim fortlaufenden Lesen von Dokumenten
5. Effiziente Platzausnutzung
6. Änderbarkeit von Schriftart und Schriftattributen
7. Veränderte Darstellung von Farben
8. Unabhängige Behandlung von Icons und anderen Bildern
9. Erkennbarkeit von Bedienelementen
10. Erkennbarkeit von Mauszeiger, Textcursor und Fokus
11. Klar erkennbare Linien
12. Erhaltung des Layouts
13. Gute Übersicht über den Bildschirminhalt
14. Orientierungsmöglichkeiten

Hierbei wird gezeigt, dass die Gemeinsamkeiten zwischen Darstellungen für Blinde und Sehbehinderte nicht nur in der grundsätzlichen Fragestellung liegen, sondern auch in den konkreten Anforderungen wiederzufinden sind.

3.1.2.1 Anpassung der Darstellung von Bedienelementen und Dokumenten

Selbstverständlich müssen angepasste Darstellungen für Blinde und Sehbehinderte sowohl die Bedienoberfläche wie auch die Anwendungsdokumente abbilden. Anpassungen, die sich nur auf einen Teil beschränken, wie beispielsweise die Vergrößerungsmöglichkeiten in Webbrowsern oder Office-Programmen reichen nicht aus, um dem Benutzer ein effizientes Arbeiten am PC zu ermöglichen, wie Abbildung 25 zeigt.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

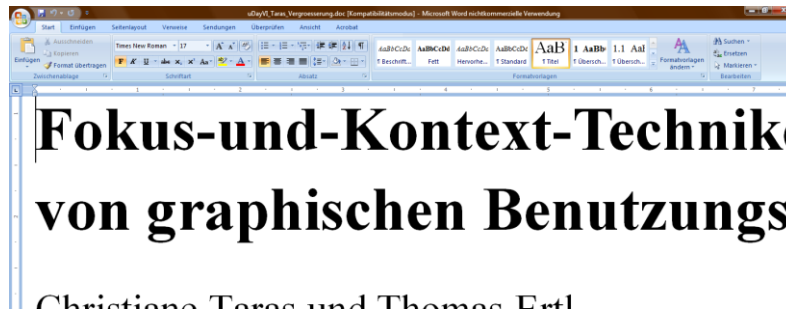


Abbildung 25 Vergrößerung mit anwendungsbezogenem Zoom in Word auf 500%
Nur das Dokument wird vergrößert, die Bedienelemente bleiben klein und damit nicht erkennbar.

3.1.2.2 Gemeinsame Darstellung logisch zusammenhängender Informationen

Für eine effiziente Arbeitsweise ist es wichtig, dass Informationen, die für den Benutzer logisch zusammengehören, auch gemeinsam erkennbar dargestellt werden. Nur so kann die gesamte Information erfasst werden oder effizient aus einem Angebot von mehreren Möglichkeiten ausgewählt werden. Beispiele dafür sind alle Einträge eines Menüs, eine Schaltfläche und deren Tooltip oder auch ein Hyperlink und dessen Zusatzinformation, die meist in der Statuszeile erscheint.

Abbildung 26 zeigt als Negativbeispiel die Vergrößerung eines Menüs mit der Windows-Bildschirmleupe. Der erscheinende Tooltip mit der interessanten Information über das Tastenkürzel der Funktion „Öffnen“ wird nicht mit vergrößert. Da er nur erscheint, wenn sich der Mauszeiger über dem Menü befindet, wird er allerdings nie in der Vergrößerung sichtbar sein und ist somit für den Sehbehinderten unzugänglich.

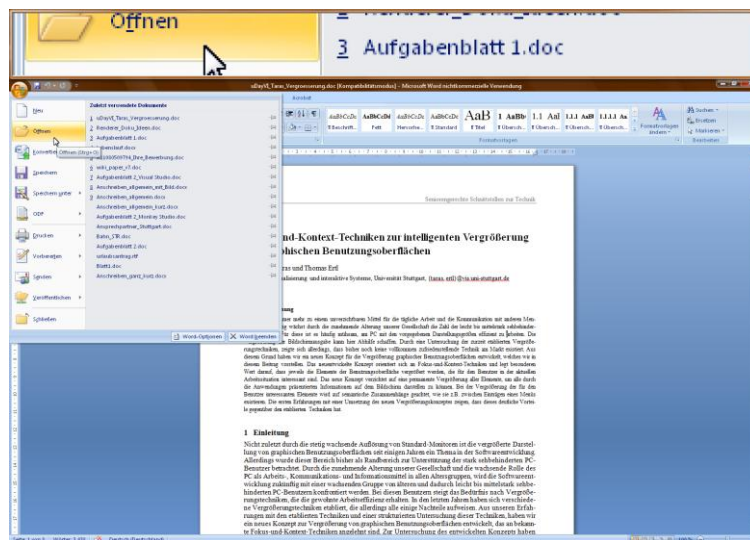


Abbildung 26 Vergrößerung eines rechteckigen Bereiches mit der Bildschirmleupe
Der Tooltip mit Zusatzinformationen zum fokussierten Menüeintrag und die umgebenden Menüeinträge werden nicht in der Vergrößerung dargestellt.

Auch die Auswahl des richtigen Menüeintrags fällt mit dieser Art der Vergrößerung schwer. Dies ist vergleichbar mit der einzeiligen Darstellung von Menüs in der Braille-Ausgabe von

3.1 Gleichartigkeit angepasster Darstellungen für beide Zielgruppen

Screenreadern. Es ist offensichtlich, dass eine mehrzeilige taktile Darstellung auf einem grafisch-taktilen Display einen schnelleren Überblick über das Menü ermöglicht. Gleiches gilt, wenn alle Menüelemente gemeinsam vergrößert werden.

3.1.2.3 Darstellung mit einer bestimmten Schriftgröße

Für die taktile Darstellung von Text in Brailleschrift wird selbstverständlich nur eine Schriftgröße verwendet (2 × 3 Punkte bei 6-Punkt-Schrift und 2 × 4 Punkte bei 8-Punkt-Schrift). Die Darstellung der Benutzungsoberfläche sollte an diese Schriftgröße angepasst werden, um nutzlose Leerräume zu vermeiden und so die geringe Ausgabefläche gut auszunutzen.

Auch zur Unterstützung Sehbehinderter sollte die Darstellung auf eine bestimmte Schriftgröße angepasst werden. Die zum größten Teil etablierte Vergrößerung um einen Faktor ist eigentlich nicht im Sinne des Benutzers, da dieser auf eine bestimmte Mindestgröße der Schrift angewiesen ist. Das Ergebnis der Anwendung eines Vergrößerungsfaktors ist allerdings von der Ausgangsschriftgröße abhängig und kann somit sehr unterschiedlich ausfallen. Ist die Ausgangsschriftart des Textes, der gelesen werden soll, sehr klein, muss ein sehr hoher Vergrößerungsfaktor angewendet werden. Dabei werden aber auch größere Texte mit vergrößert und nehmen somit eine unnötig große Fläche ein, sodass sie nicht mehr im Ganzen in der aktuellen Ansicht dargestellt werden können. Dies erhöht den Scroll-Aufwand unnötig. Abbildung 27 verdeutlicht dies.

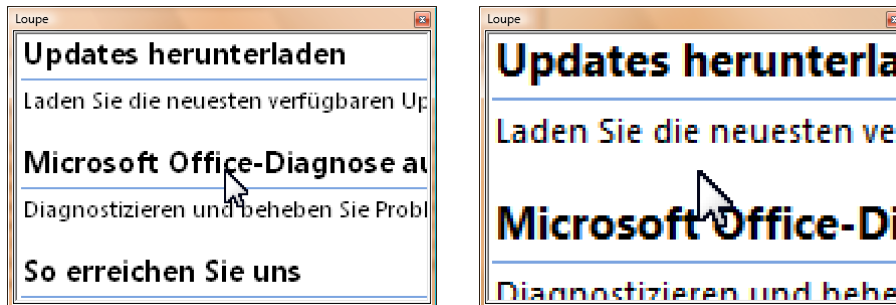


Abbildung 27 Vergrößerung eines Dialoges mit Hilfe einer Bildschirmlupe auf 200% bzw. 300%
Im linken Bild ist die große Schrift etwa genauso groß wie die kleine Schrift im rechten Bild.

Die Darstellung aller Texte in einer bestimmten Schriftgröße führt zwar dazu, dass zum Beispiel normaler Text genauso groß dargestellt wird wie Überschriften und somit die Schriftgröße als Unterscheidungsmerkmal entfällt. Da für solche Unterscheidungen aber meist noch andere Gestaltungsmittel eingesetzt werden, wie etwa fette Schrift oder Nummerierungen, ist dies kein wesentlicher Nachteil.

3.1.2.4 Wenig horizontales Scrollen beim fortlaufenden Lesen von Dokumenten

Der wesentliche Grund für die Arbeit am PC ist das Lesen und Bearbeiten der Anwendungsdokumente. Deshalb sollte das fortlaufende Lesen von Dokumenten für den Benutzer möglichst einfach sein, um eine möglichst hohe Arbeitseffizienz zu erreichen. Durch eine Vergrö-

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

ßerung sollte möglichst keine Notwendigkeit zum horizontalen Scrollen entstehen, da dies für den Benutzer sehr ermüdend und häufig auch verwirrend ist [BL96]. Dabei entsteht das Problem meist nicht beim Lesen einer Zeile von links nach rechts, sondern beim Aufsuchen der nächsten Zeile. Eine Vergrößerungstechnik sollte also möglichst viel Platz für das Anwendungsdokument zur Verfügung stellen und wenn möglich den Text auf die zur Verfügung stehende Bildschirmbreite umbrechen. Bei Verringerung der Auflösung beispielsweise wird der Platz für das Anwendungsdokument meist stark eingeschränkt, wie Abbildung 28 im Vergleich mit der reinen Vergrößerung des Dokuments zeigt.

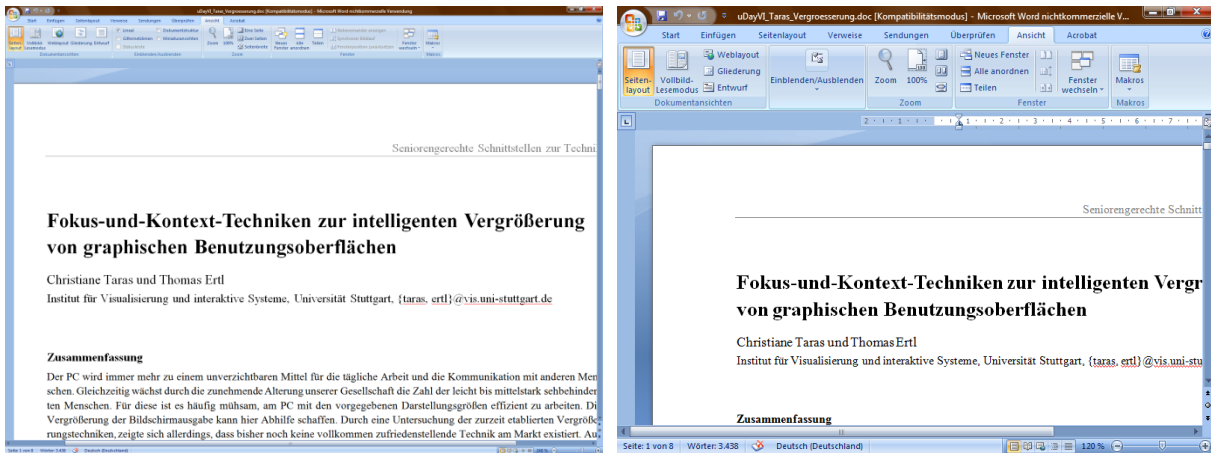


Abbildung 28 Auswirkungen einer Halbierung der Auflösung auf die Darstellung des Anwendungsdokumentes (Links wurde der Anwendungszoom angepasst, damit die Schriftgröße der im rechten Bild entspricht. Trotzdem ist ein wesentlich größerer Teil des Dokumentes erkennbar. Rechts ist also ein höherer Scrollaufwand beim Lesen nötig.)

Auch bei der Nutzung von taktilen Ausgaben sollte, wenn ein Text gelesen werden soll, möglichst die gesamte Ausgabefläche genutzt und der Text auf die Ausgabebreite umgebrochen werden. Zwar können sich Blinde, zumindest wenn sie zweihändig arbeiten, besser an den Zeilen orientieren als Sehbehinderte, aber auch für sie bedeutet die Möglichkeit, in zwei Richtungen scrollen zu müssen, einen höheren kognitiven Aufwand.

3.1.2.5 Effiziente Platzausnutzung

Da für die angepasste Darstellung nur verhältnismäßig wenig Platz zur Verfügung steht, sollte dieser effizient genutzt werden, um möglichst viele Informationen in der Größe der Ausgabefläche darstellen zu können und dem Nutzer damit Scrollarbeit zu ersparen. Werden wie bei Bildschirmvergrößerungssoftware die Proportionen zwischen für die Darstellung genutztem Platz und vorhandenen Freiräumen beibehalten, so wird viel Platz verschwendet, der eigentlich für die vergrößerte Darstellung genutzt werden könnte. Abbildung 29 zeigt, wie allein durch Verkleinerung von Leerräumen und Änderung des Größenverhältnisses zwischen Icons und Text eine platzeffiziente dreifach vergrößerte Darstellung des Startmenüs realisierbar ist, die vollständig auf den Bildschirm passt.

3.1 Gleichartigkeit angepasster Darstellungen für beide Zielgruppen

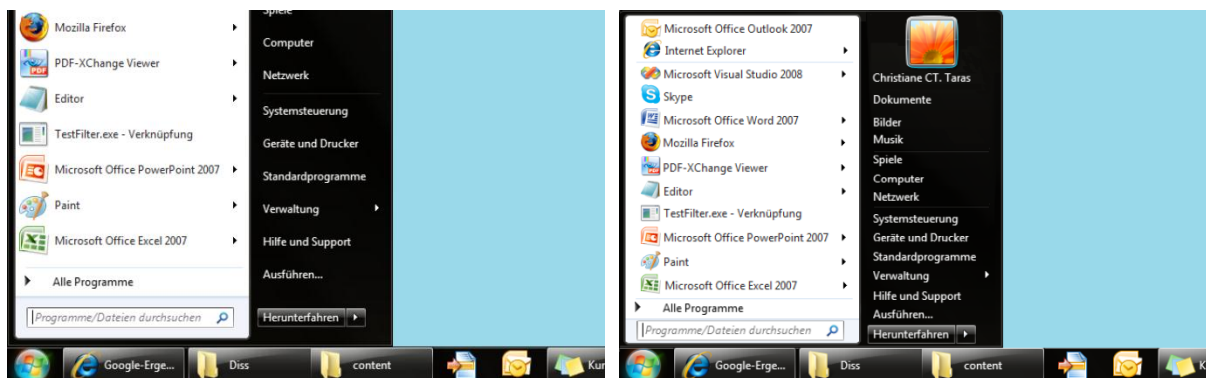


Abbildung 29 Darstellung der Auswirkung sinnvoller Platzausnutzung bei der Vergrößerung (3-fach)

Bei der Änderung der Standardschriftgröße kann man in geringem Maße bereits heute eine Optimierung der Leerräume beobachten. Auch bei Änderung der Auflösung wäre dies durch bessere Implementierung von Oberflächen potentiell möglich. Bei der Vergrößerung auf ein Vielfaches der Bildschirmgröße kann eine solche Anpassung allerdings nicht erreicht werden, da die Anwendungen selbst nicht über die veränderte Darstellung informiert werden. Durch diesen Umstand werden leider auch weitere nützliche Funktionen umgangen, die die Darstellung auf die Bildschirmbreite beschränken und so den Scrollaufwand verringern, wie beispielsweise Layoutalgorithmen für Menüs (siehe Abbildung 30).

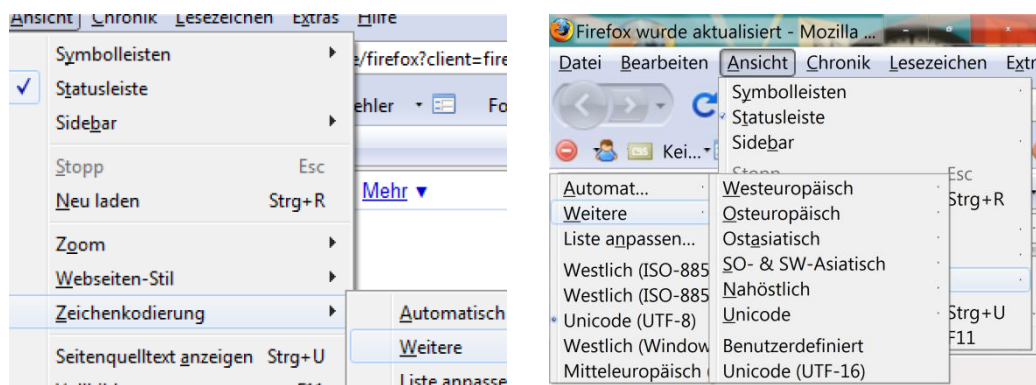


Abbildung 30 Darstellung der Auswirkung von Vergrößerung (5-fach) auf die Menüdarstellung
links: Vergrößerung mit Bildschirmlupe im Vollbild; rechts: erhöhte Standardschriftart

3.1.2.6 Änderbarkeit von Schriftart und Schriftattributen

Die Lesbarkeit von Text hängt maßgeblich von der verwendeten Schriftart ab. Bei Blinden wird dies schnell deutlich. Text ist in seiner grafischen Darstellung auch auf einer flächigen Ausgabe nur schwer erfassbar. Für effizientes Arbeiten muss eine taktil leicht lesbare Schrift, wie die Brailleschrift, eingesetzt werden. Allerdings wird diese längst nicht von allen Blinden beherrscht. So ist auch die Darstellung in taktiler Schwarzschrift, wie sie Abbildung 31 (Mitte) zeigt, nützlich, vor allem, wenn keine Sprachausgabe genutzt werden kann. Selbst bei Nutzung von Brailleschrift muss ein Austausch der Schrift leicht möglich sein, um beispielsweise zwischen 6-Punkt- und 8-Punkt-Schrift zu wechseln oder die Darstellung spezieller Zeichen wie Mathematik- oder Musiksymbole zu ermöglichen. Letztlich existieren neben der Braille-

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

schrift auch weitere Punktschriften, wie etwa die Fakoo-Schrift [Fak] (Abbildung 31 unten), deren Einsatz auch möglich sein sollte.

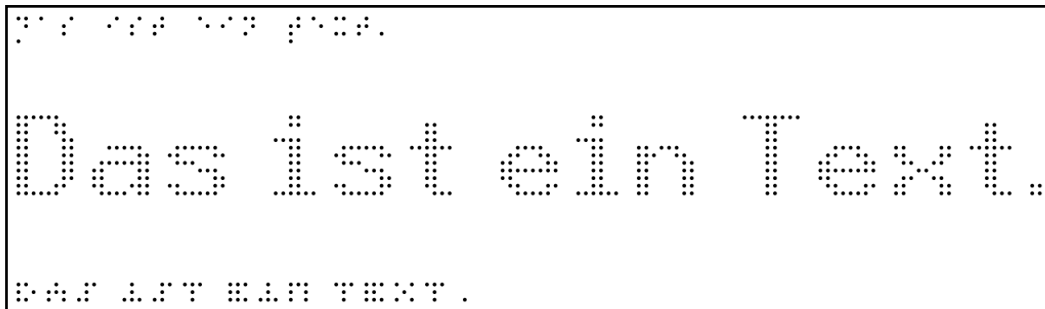


Abbildung 31 Verschiedene taktile Schriften
v. o. n. u.: 8-Punkt-Computerbraille, taktile Schwarzschrift (nach einem unveröffentlichten Entwurf des Zentrum für Multimedia, FH Kiel), Fakoo-Schrift von Alexander Fakoó [Fak]

Auch bei Sehbehinderten ist die Wahl der Schriftart ein wesentlicher Faktor, wie Abbildung 32 veranschaulicht. Verschnörkelte Schriften, Schriften, die mit sehr feinen Linien gezeichnet werden, oder Serifenschriften sind nicht so gut erkennbar wie einfache serifenlose Schriften mit einer stärkeren Linienführung. Dies führt dazu, dass eine höhere Vergrößerungsstufe genutzt werden muss, als eigentlich (bei anderer Schriftart) nötig wäre. Je nach Schriftart sind verschiedene Buchstaben auch nicht eindeutig identifizierbar oder verschwimmen ineinander, was den Leser zusätzlich belastet. Eine Hilfssoftware für Sehbehinderte sollte also auch eine Änderung der zur Darstellung des Textes verwendeten Schriftart ermöglichen. Auch die Umstellung auf fette und nicht kursive Schrift sollte möglich sein.

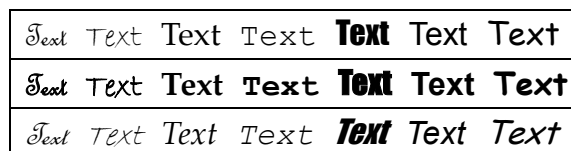


Abbildung 32 Auswirkung verschiedener Schriftarten und Schriftattribute auf die Lesbarkeit
(Schriftgröße jeweils 12 pt)

3.1.2.7 Veränderte Darstellung von Farben

Sehbehinderte sind meist auf hohe Kontraste angewiesen. Nur so können Sie Vordergrund und Hintergrund gut unterscheiden. Gibt die Benutzungsschnittstelle oder das Dokument nur schwache Kontraste vor, wie beispielsweise im Ribbon von Word (siehe Abbildung 33 oben), so fällt das Lesen schwer. Eine kontrastreiche Darstellung, wie sie in Abbildung 33 unten gezeigt ist, kann hier helfen. Es sollte dem Sehbehinderten möglich sein, diese Einstellung schnell und den eigenen Bedürfnissen entsprechend zu ändern.

3.1 Gleichartigkeit angepasster Darstellungen für beide Zielgruppen



Abbildung 33 Ribbon von Word in Originalfarben (oben) und kontrastreicher Darstellung (unten)
Die kontrastreiche Darstellung wurde über das Windows-Schema „Kontrast Nr. 2“ erzeugt.

Aktuelle Bildschirmvergrößerungssoftware bietet umfangreiche Möglichkeiten, Farben gegen andere auszutauschen. Der Austausch erfolgt dabei, wie schon in Abschnitt 2.1.4 beschrieben, auf Grundlage der Bildschirmrastergrafik. Somit werden alle Farbpunkte gleich behandelt, unabhängig von der Anwendung, durch die sie erzeugt wurden, und auch unabhängig davon, ob sie von einem Bild oder von Text stammen.

Da sich Erläuterungen zu Bildern und Anwendungen häufig auf die darin enthaltenen Farben beziehen, sollten die Originalfarben der Darstellung leicht zugänglich sein, während der Text lesbar bleiben sollte. Da die Originalfarben für den Benutzer eventuell nicht erkennbar sind, müssen Ersatzdarstellungen gefunden werden. Gleiches gilt offensichtlich für die Darstellung auf grafisch-taktilen Displays, da diese derzeit meist nur zwei Stiftzustände, also nur zwei Farben, unterstützen. In statischen taktilen Grafiken werden Einfärbungen meist reduziert und in Texturen umgewandelt, wobei die Farbinformation an sich meist verloren geht, da keine Legenden zur Zuordnung zu Farben ergänzt werden. Bei dem interaktiven System von [Vie08] werden Farben verbalisiert. Gleiches geschieht bei SVG4Blind von Rotard et al. [ROE04]. Hier können zusätzlich Farbfilter verwendet werden, wodurch die Verteilung von einer Farbe innerhalb der Ausgabefläche erfasst werden kann. Auch Screenreader erlauben bei Textelementen bereits Zugriff auf Farbinformationen. Bei Bildschirmvergrößerungssoftware sind die Originalfarben nur zugänglich, wenn kein veränderndes Farbschema aktiv ist. Möglicherweise sind sie aber auch dann für den Benutzer nicht erfassbar.

3.1.2.8 Unabhängige Behandlung von Icons und anderen Bildern

Eine unabhängige Behandlung von Bildern ist nicht nur bezüglich der Farbdarstellung interessant. Bilder und insbesondere Icons sollten immer gesondert behandelt werden können.

Icons sollen die Verständlichkeit einer Benutzungsoberfläche erhöhen und die Arbeit mit einer Oberfläche beschleunigen. Dies kann natürlich nur gelingen, wenn sie vom Benutzer auch gut und vor allem schnell erkannt werden können. Werden Icons in Ihrer Originaldarstellung (bzw. einem monochromen Abbild davon) auf grafisch-taktilen Displays dargestellt, so können sie nur sehr schwer erfasst werden. Eine solche Darstellung ist nur interessant, wenn tatsächlich das Icon an sich erfasst werden soll. Für ein effizientes Arbeiten sollte das Icon entweder gar nicht dargestellt werden, sofern es redundant zu einem Text in der Ober-

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

fläche ist, oder durch eine Ersatzdarstellung, wie dem Alternativtext, dem Namen oder einem leichter erfassbaren, besonders designten taktilen Icon ersetzt werden.

Auch bei der vergrößerten Darstellung für Sehbehinderte sollten Icons gesondert behandelt werden. Natürlich müssen sie in einer möglichst guten Qualität auf die vom Benutzer bevorzugte Größe skaliert werden können. Es ist allerdings nicht immer nötig, alle Icons einfach passend zum Skalierungsfaktor der Schrift zu vergrößern. Häufig sind Icons schon groß genug oder zumindest wesentlich größer als die Schrift, wie das Icon für „Einfügen“ in Abbildung 33 zeigt. In diesen Fällen ist ein vergrößertes Icon eventuell gar nicht erforderlich und würde nur unnötig Platz verschwenden.

Ebenso ist eine Vergrößerung anderer Bilder entsprechend der Schriftvergrößerung nicht immer sinnvoll. Befindet sich das Bild in einem Dokument, in dem der Benutzer vor allem den Text lesen will, würde ein vergrößertes Bild möglicherweise nur stören, da es unnötig Platz einnehmen und damit den Scrollaufwand erhöhen würde. Ist das Bild für den Benutzer doch interessant, so ist eine pauschale Vergrößerung aber eventuell auch nicht sinnvoll, da automatisiert nur schlecht festgestellt werden kann, um wie viel das Bild vergrößert werden müsste, um für den Benutzer gut erkennbar zu sein. Somit ist es am sinnvollsten, wenn der Benutzer Bilder getrennt von der Schriftskalierung vergrößern kann.

In der taktilen Darstellung ist eine Trennung von Text und Bildern ebenso wichtig. Da Bildpunkte mit der gleichen Auflösung dargestellt werden, wie die Punktschrift, ist eine Unterscheidung aufwändiger als am Bildschirm. Die gemischte Darstellung von Text und Bildern sollte vermieden werden. Eine Ersetzung der Bilder durch ihren Alternativtext (sofern vorhanden) ist sinnvoller. Hat der Benutzer Interesse an dem Bild, so sollte er es in einer gesonderten Darstellung aufrufen können. Dieses Vorgehen ist auch für Sehbehinderte sinnvoll.

3.1.2.9 Erkennbarkeit von Bedienelementen

Die verschiedenen Bedienelemente grafischer Oberflächen sind meist durch verschieden große Rechtecke realisiert, die sich nur schwer unterscheiden lassen. Lediglich leichte Unterschiede in der Farbgebung lassen Rückschlüsse auf den Typ des Bedienelements und damit die Funktion und Interaktionsmöglichkeiten des Elements zu. Durch eine kontrastreiche Darstellung verschlechtert sich dies eventuell sogar noch, da dabei die genutzten Farben eingeschränkt werden, wie Abbildung 33 zeigt. Das Eingabefeld ist in der kontrastreichen Darstellung quasi nicht mehr von den Schaltflächen zu unterscheiden. Gleiches würde geschehen, wenn in einer taktilen Darstellung Bedienelemente nur durch Rechtecke (mit Text) dargestellt werden würden. Für ein effizientes Arbeiten müssen die verschiedenen Typen von Bedienelementen aber gut unterscheidbar sein. Sie sollten somit speziell markiert werden, wie dies bei der Braille-Ausgabe von Screenreadern durch Buchstabenkürzel (beispielsweise „sln“ für Schaltflächen) bereits etabliert ist. Auch in der grafischen Darstellung können Bedienelemente so gestaltet werden, dass ihre Typen leicht erkennbar sind. Kristin Albert entwickelte in ihrer Belegarbeit entsprechende Typ- und Zustandsmarker, aus denen taktile Widgets zusammengesetzt werden können [Alb08] (siehe Abbildung 34 links). Ähnliche Ge-

3.1 Gleichartigkeit angepasster Darstellungen für beide Zielgruppen

staltungen können auch die Darstellung am Bildschirm verbessern, wie der Entwurf in Abbildung 34 (rechts) zeigt. Die Schaltflächen und Eingabefelder des Ribbon-Ausschnitts sind deutlich unterscheidbar.

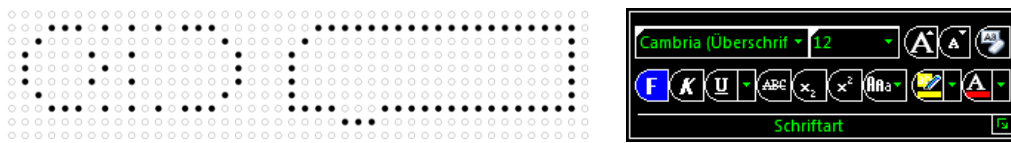


Abbildung 34 Entwurf zu Markierung von Bedienelementtypen
links: Entwürfe für taktile Widgets von Albert (OK-Schaltfläche und ausklappbares Eingabefeld)
rechts: ein Ausschnitt aus dem Word-Ribbon mit verbesserten Schaltflächen und Eingabefeldern

3.1.2.10 Erkennbarkeit von Mauszeiger, Textcursor und Fokus

Der Mauszeiger ist das wesentliche Interaktionsmittel für grafische Benutzungsschnittstellen. Zwar sind zumindest die wichtigsten Funktionen auch über Tastaturbefehle abrufbar, trotzdem nutzen aber auch Sehbehinderte häufig die Maus als Auswahl- und Navigationsinstrument. Letztlich ist die Mausnutzung bei eher unzugänglichen Anwendungen und auch vielen Webseiten die einzige verbleibende Möglichkeit. Deshalb muss der Mauszeiger immer gut erkennbar sein, also zumindest entsprechend der gewählten Vergrößerungsstufe vergrößert sein. Eine spezielle Markierung des Mauszeigers, wie in Abbildung 35 gezeigt, ist häufig nützlich.

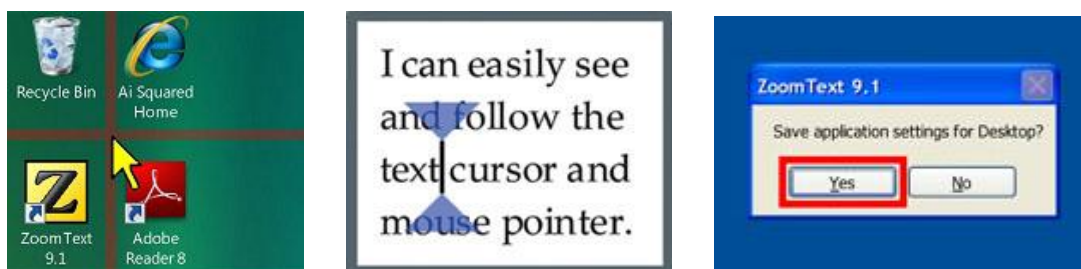


Abbildung 35 Beispiele zur Vergrößerung und Markierung von Mauszeiger, Textcursor und fokussiertem Element aus ZoomText 9.1 [Ai 09]

Ebenso sollten der Textcursor (auch als Caret oder Textmarke bezeichnet), der die aktuelle Stelle zur Texteingabe markiert, und das fokussierte Element für den Benutzer leicht erkennbar und auffindbar sein, damit dieser sichergehen kann, dass Eingaben an der richtigen Stelle eingefügt bzw. auf das richtige Element angewendet werden.

In taktilen Darstellungen können die in Abbildung 35 gezeigten grafischen Darstellung (Liniencursor, große Textmarke und besonders gerahmtes Bedienelement) nachgebildet werden. Für die Arbeit an einem berührungsempfindlichen Display wie dem BrailleDis 9000 ist für den blinden Benutzer eine Darstellung des Mauszeigers zwar eigentlich nicht nötig. Für die Zusammenarbeit mit Sehenden kann sie aber nützlich sein.

3.1.2.11 Klar erkennbare Linien

Wie schon die verschiedenen Schriften in Abbildung 32 zeigen, sind Darstellungen, die klare, dicke Linien nutzen, leichter zu erkennen als solche mit dünnen Linien. Dies gilt nicht nur für Texte, sondern auch für grafische Darstellungen, wie die aufbereitete Sinus-Kurve in Abbildung 36 (links unten) zeigt. Auch aus Gestaltungsvorschriften für taktile Grafiken ist bekannt, dass klare und möglichst ununterbrochene Linien das Erfassen von Grafiken erleichtern. Aufgrund der geringen Auflösung aktueller grafisch-taktile Displays ist besonders bei solchen Darstellungen wichtig, dass Linien nicht unterbrochen sind. Sonst kann der Benutzer nicht feststellen, ob die angehobenen Stifte wirklich zusammengehören oder nicht, wie Abbildung 36 (rechts unten) zeigt.

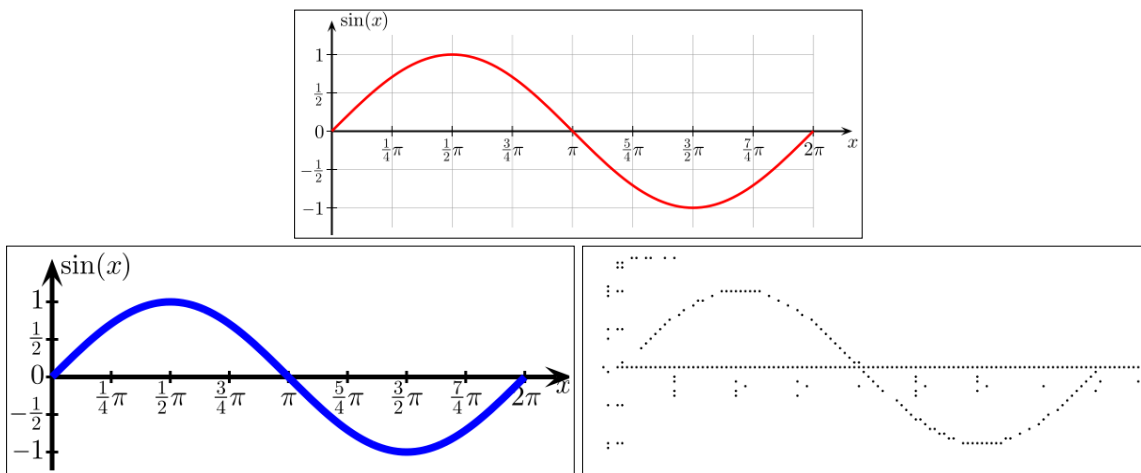


Abbildung 36 *Verschiedene Darstellungen einer Sinus-Kurve
oben: ursprüngliche Darstellung (erzeugt aus einem SVG)
links unten: Darstellung für Sehbehinderte mit stärkeren Linien und größerer Schrift (Positivbeispiel, trotz gleicher Größe, ist die Grafik deutlich besser erkennbar)
rechts unten: lückenhafte Darstellung auf grafisch-taktilem Display (Negativbeispiel, wurde erzeugt durch einfache Verkleinerung der Originaldarstellung und Anwendung eines mittleren Helligkeitsschwellwert zur Konvertierung in eine monochrome Darstellung)*

Die schlechte Darstellung in Abbildung 36 (rechts unten) entsteht vor allem durch schlechte Umsetzung der Verkleinerung der Darstellung auf 120×60 Punkte. Aber auch bei vergrößerten Darstellungen können unklare Kanten entstehen. Besonders bei vergrößertem Text wirken sich verschwommene oder blockige Kanten negativ aus, da durch sie das Lesen schwerer fällt als bei klaren Kanten. Das Lesen von Texten ist meist Ziel der Arbeit am Computer. Besonders hier ist also eine kontrastreiche Darstellung wichtig. Diese kann nur entstehen, wenn der vergrößerte Text klare Kanten aufweist. Abbildung 37 zeigt, wie sich unterschiedliche Methoden zur Vergrößerung auf die Qualität der Schriftdarstellung auswirken können.

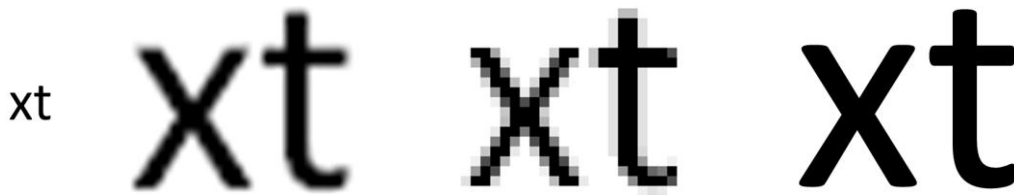


Abbildung 37 Qualitative Unterschiede bei der Vergrößerung von Schrift
(v. l. n. r.: Originaltext, Vergrößerung mit bilinearer Interpolation, Vergrößerung mit Nearest-Neighbor-Verfahren, Vergrößerung durch Änderung der Schriftgröße)

3.1.2.12 Erhaltung des Layouts

Die Erhaltung des ursprünglichen Layouts der grafischen Oberfläche ist aus mehreren Gründen wichtig. Einerseits verschlechtert sich die Sehfähigkeit bei vielen Menschen langsam nach und nach und sie sind die Arbeit mit einer Anwendung seit längerer Zeit gewohnt. Eine wesentliche Veränderung der Anordnung der Elemente der Oberfläche in der angepassten Darstellung würde dann nur zusätzliche Belastung erzeugen. Besteht nur eine leichte Sehbehinderung könnte eine zu starke Änderung der Darstellung den Benutzer auch abschrecken und so von der für ihn eigentlich hilfreichen Anwendung einer Vergrößerung abhalten. Aber auch wenn der Benutzer die Anwendung erst kennenlernt, kann eine Änderung des Layouts der Benutzungsoberfläche hinderlich sein, da Programmhilfen die Position von Menüs oder Steuerelementen häufig absolut angeben (wie z.B. „oben auf der linken Seite“) oder auf Screenshots der Originaldarstellung anzeigen. Letztlich ist es auch für die Zusammenarbeit mit Normalsichtigen von Vorteil, wenn der sehbehinderte oder blinde Benutzer Kenntnisse über das Originallayout der Anwendung hat. Häufig werden durch das Layout auch logische Zusammengehörigkeiten dargestellt, die sonst nicht in der grafischen Oberfläche hinterlegt sind. Bei Dokumenten ist die Erhaltung des Layouts ebenfalls sinnvoll da der Benutzer nur so die Möglichkeit hat, gut strukturierte Dokumente zu erstellen bzw. das Layout der von ihm erstellten Dokumente zu prüfen. Außerdem sind auch in Dokumenten logische Zusammenhänge oder die Bedeutung von Teilen des Dokuments häufig über das Layout ausgedrückt. Vor allem bei tabellarischen Strukturen ist die Erhaltung des Layouts wichtig, um Informationsverlust zu vermeiden.

3.1.2.13 Gute Übersicht über den Bildschirminhalt

Die wesentliche Herausforderung ist die Erzeugung von Darstellungen, die einen guten Überblick über Anwendungen und Dokumente ermöglichen. Ein schneller Überblick ist nur möglich, wenn die gesamte Darstellung auf einmal im Ausgabebereich erkennbar ist. Kann die Darstellung nur mit Hilfe von Scrolloperationen vollständig erfasst werden, ist ein erhöhter kognitiver Aufwand nötig. Die Zusammenhänge zwischen einzelnen Bestandteilen der Darstellung können nur schwerer erfasst werden. Außerdem können, wenn sehr viele Scrolloperationen nötig sind, Teile der Darstellung leicht übersehen werden. Dies ist besonders dann nachteilhaft, wenn Teile der grafischen Oberfläche andere in ihrer Funktionalität einschränken oder blockieren, wie dies häufig bei Meldungs- oder Abfragedialogen der Fall ist.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Dies ist ein Problem aktueller Bildschirmvergrößerungssoftware, wie Abbildung 38 verdeutlicht. Diese zeigt beispielhaft den Ablauf beim Versuch, ein nicht mehr verfügbares Dokument zu öffnen. Dabei erscheint das blockierende Meldungsfenster mit dem passenden Hinweis außerhalb des aktuellen Sichtbereichs des Nutzers.

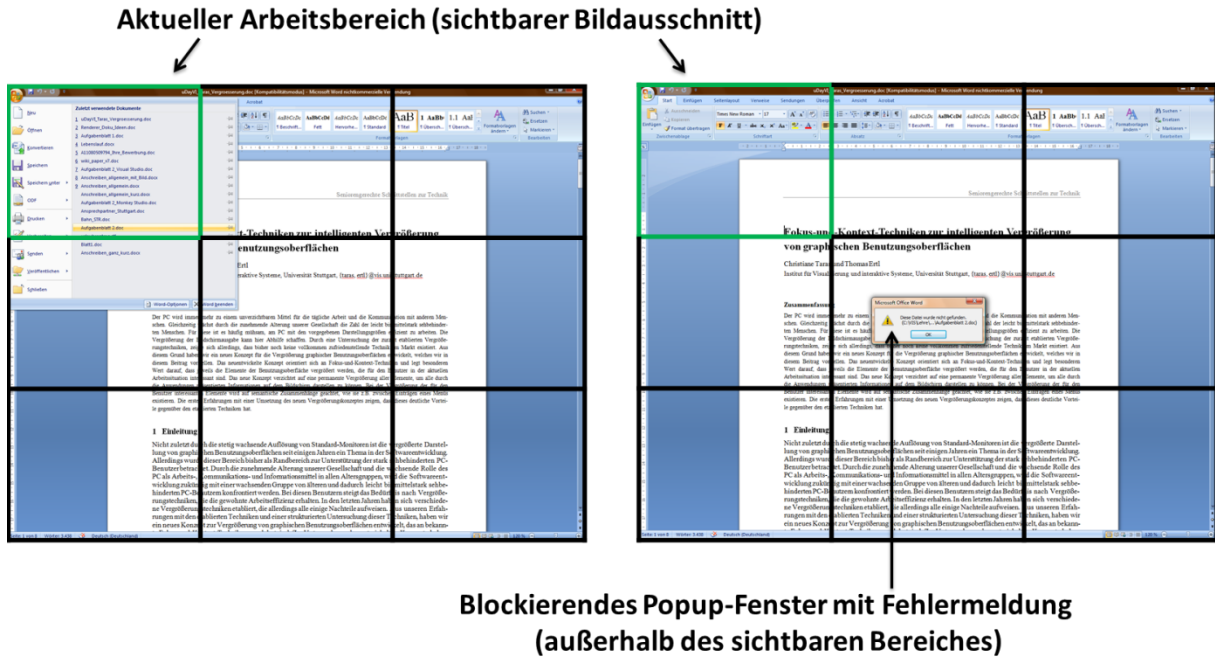


Abbildung 38 Nicht sichtbare Meldung bei Vergrößerung auf ein Vielfaches der Bildschirmgröße

Eine Darstellung, die eine gute Übersicht ermöglicht, kann stark von den darzustellenden Inhalten, den im jeweiligen Kontext zu vermittelnden Informationen und auch den Bedürfnissen und Vorlieben des Benutzers abhängig sein. Häufig ist auch eine Reduzierung des Darstellungsdetails, wie es im Bereich der taktilen Grafiken üblich ist (siehe Abbildung 39), nötig und sinnvoll.

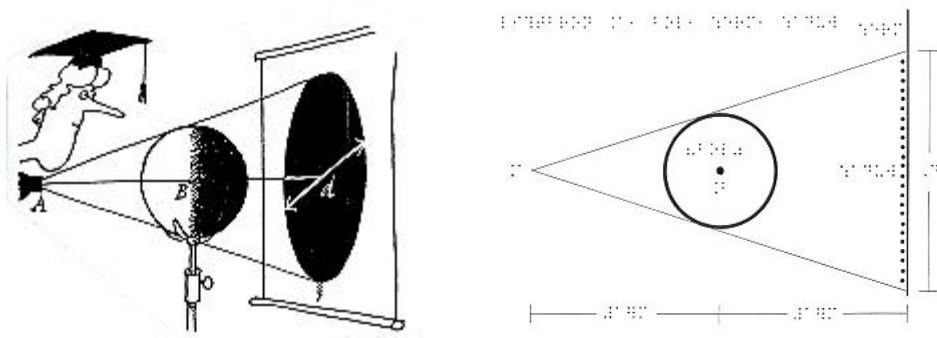


Abbildung 39 Beispiel zur Reduktion von Darstellungsdetails in taktilen Grafiken links: Original-Grafik, rechts: taktile Umsetzung (Quelle: [Sch02])

Dieses Prinzip kann auch auf die Darstellung grafischer Benutzungsoberflächen angewendet werden. So genügt zur Übersicht über die geöffneten Fenster auf einem grafisch-taktilen Display beispielsweise eine Liste der Fenstertitel. Die Anordnung der Fenster auf dem Bild-

3.1 Gleichartigkeit angepasster Darstellungen für beide Zielgruppen

schirm ist eventuell weniger wichtig. Interessiert die Anordnung am Bildschirm oder wird eine grafikbetontere Darstellung bevorzugt, wie bei der Ausgabe für Sehbehinderte, so wäre es für einen ersten Überblick sinnvoll, nur die Rahmen der Fenster mit deren Titel zu zeigen. Für grafisch-taktile Displays erscheint diese Darstellungsform nur natürlich. Bei der Bildschirmausgabe zur Unterstützung von Sehbehinderten wird sie allerdings bisher nicht eingesetzt, obwohl sie deutliche Vorteile bringen kann, wie Abbildung 40 zeigt.

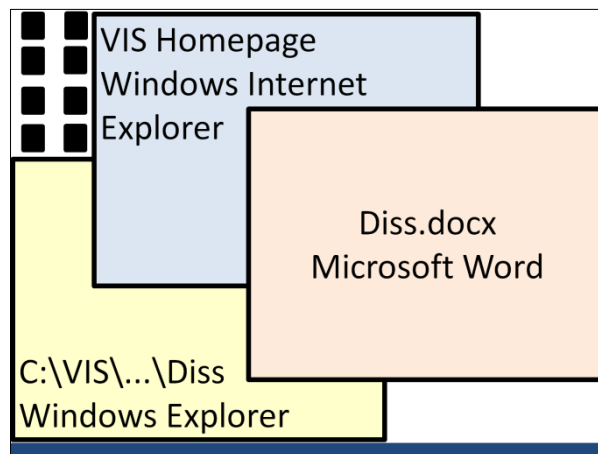
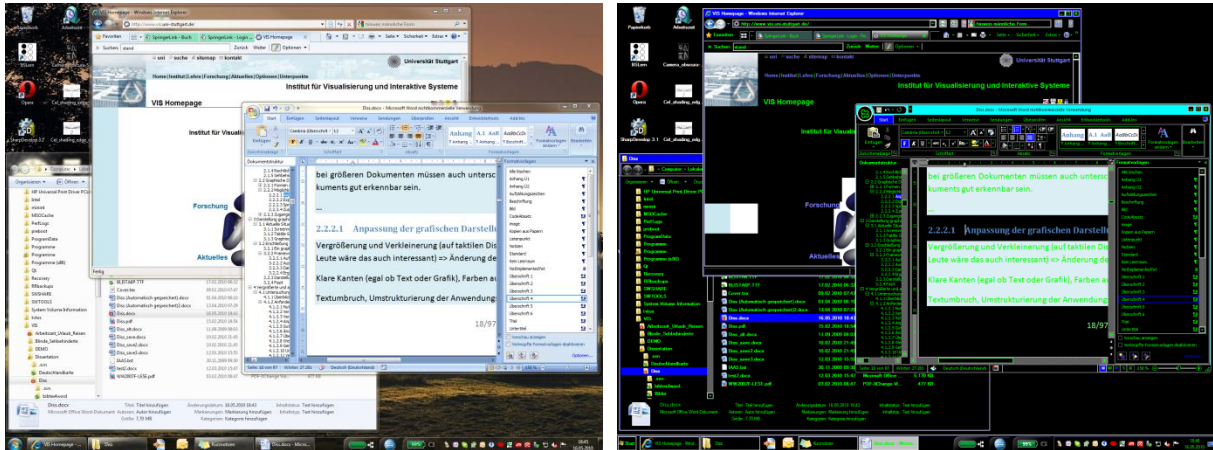


Abbildung 40 Beispiel für Übersichtsdarstellung des Desktops für Sehbehinderte
oben links: Original-Darstellung des Desktops; oben rechts: Darstellung mit hohem Kontrast (Die Fensterrahmen sind kaum besser sichtbar als im Original. Es lässt sich nur schwer erkennen, um welche Anwendung es sich handelt. Ein Icon am Desktop ist kaum sichtbar)
unten: eigener Vorschlag zur verbesserten Übersichtsdarstellung (Fensterrahmen und -titel, sowie Desktop-Icons und Taskbar sind leicht erkennbar. Die Schriftgröße der Fenstertitel ist etwa 6 Mal so groß wie in der Original-Darstellung.)

3.1.2.14 Orientierungsmöglichkeiten

Arbeitet ein Nutzer mit einer Darstellung, die nur einen Teil der originalen Bildschirmgrafik zeigt, muss es ihm möglich sein, auf einfache Weise zu erkennen, in welchem Bereich der Bildschirmausgabe er sich gerade befindet. Aktuelle Bildschirmvergrößerungsprogramme bieten dazu sogenannte Übersichts-Modi an, in denen schlicht die Originaldarstellung der Bildschirmausgabe gezeigt wird (siehe Abbildung 41). Diese dienen allerdings nur zur Markierung des aktuellen Vergrößerungsausschnittes innerhalb der originalen Bildschirmausgabe.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

be. Mehr Informationen, wie beispielsweise die Grenzen von kleineren Dialogen, sind für Sehbehinderte in dieser Darstellung kaum erfassbar, sonst wäre eine Vergrößerung ja nicht nötig. Die Übersichts-Modi sind somit vergleichbar mit sogenannten Minimap-Darstellungen aus Editoren für großflächige Diagramme und ähnlichem.

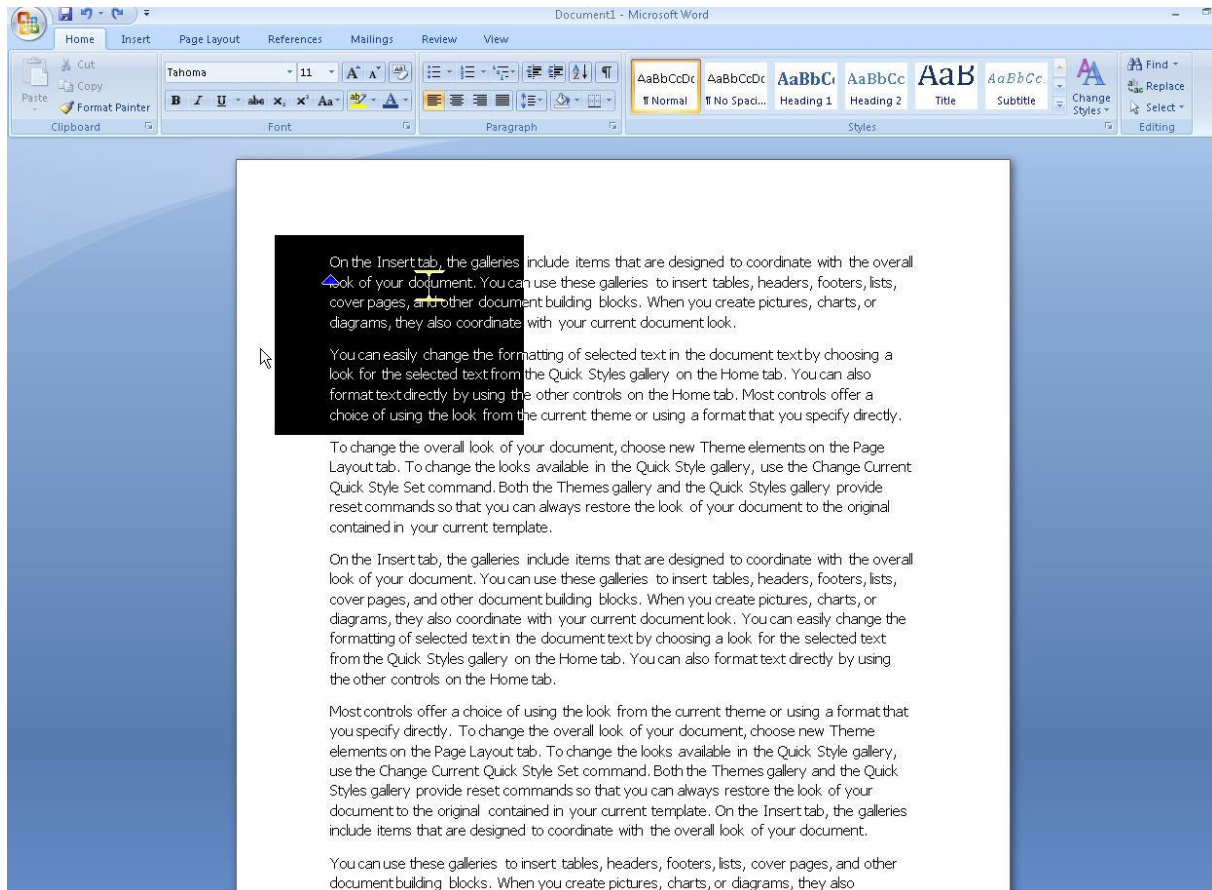


Abbildung 41 Übersichts-Modus bei Bildschirmvergrößerungsprogrammen⁹
Die Bildschirmausgabe wird in Originalgröße gezeigt. Das schwarze Rechteck markiert den Bereich, der in der vergrößerten Darstellung auf dem Bildschirm angezeigt wird.

Die Anzeige eines Orientierungsrechtecks ist zwar allgemein für Darstellungen, die ein Vielfaches der Ausgabefläche benötigen, sinnvoll. Dem Nutzer sollte aber neben der absoluten Orientierung in der Fläche auch möglich sein, zu erkennen, in welchen Bereichen der Ausgabe sich weitere interessante Objekte befinden und wie er dorthin gelangt. Eine Kombination mit gefilterten und reduzierten Darstellungen wie in Abschnitt 3.1.2.13 ist hier sinnvoll.

Die Übersichtsdarstellung mit Markierung des aktuellen Bereichs durch ein Rechteck lässt sich auch auf grafisch-taktile Displays abbilden. Wird das Markierungsrechteck mit blinkendem Rahmen angezeigt, kann es gut aufgefunden werden, wie Nutzertests in HyperBraille zeigten. Sind andere interessante Objekte genügend deutlich erkennbar, so kann diese Darstellungsform auch gut zur Navigation innerhalb der Gesamtdarstellung genutzt werden.

⁹ Bildquelle: <http://www.yourdolphin.com/>

3.1.3 Beachtung von verschiedenen Arbeitssituationen und Arbeitsbereichen

Offensichtlich stehen einige der zuvor genannten Anforderungen im Widerspruch zu einander, beispielsweise „Erhaltung des Layouts“ und „wenig horizontales Scrollen“. Allerdings ist es keinesfalls, nötig mit jeder Darstellung alle Anforderungen zu erfüllen. So ist in vielen Situationen zunächst nicht die eigentliche grafische Gestaltung (für Normalsichtige) wesentlich, sondern die Erkennung von Strukturen und des Inhalts der Darstellung. Beispielsweise ist beim Erstellen neuer Dokumente vor allem ein effizientes Arbeiten wichtig. Zunächst steht die Erstellung und gute Erfassbarkeit des Inhalts im Vordergrund. Die grafische Gestaltung ist anfangs weniger interessant. Für eine abschließende Kontrolle des Dokuments, bevor es anderen evtl. auch Normalsichtigen zur Verfügung gestellt wird, ist aber ein Zugang zur Darstellung für Normalsichtige wichtig. Nur so können Blinde und Sehbehinderte sicher sein, Dokumente zu erstellen, die von Normalsichtigen ohne weiteres akzeptiert werden und nicht gleich auf die Behinderung hinweisen. Auch für die allgemeine Zusammenarbeit mit Normalsichtigen oder die Arbeit mit Dokumenten und Anwendungen, deren grafische Darstellung dem Benutzer schon vertraut ist, kann eine Darstellung, die der Originaldarstellung entspricht oder diese zumindest sehr gut nachstellt, sehr nützlich sein.

Zur optimalen Unterstützung von blinden und sehbehinderten Computernutzern sollten also durchaus verschiedene Darstellungen für ein und dieselbe Bildschirmausgabe angeboten werden, zwischen denen der Benutzer entsprechend der aktuellen Arbeitssituation leicht auswählen kann. Dieses Prinzip ist auch bereits in einigen Anwendungen und Hilfsmitteln etabliert. So bietet beispielsweise Microsoft Word verschiedene Ansichten auf ein Dokument an (Seitenlayout, Lesemodus, Gliederung, Entwurf), in denen einige Einstellungen zur verwendeten Schrift und zur Vergrößerung auch unabhängig voneinander vorgenommen werden können. In Bildschirmvergrößerungssoftware können Texte in separaten Fenstern angezeigt werden, in denen veränderte Farbeinstellungen möglich sind und der Text am Fenster Rand umgebrochen wird, um horizontales Scrollen unnötig zu machen. Bei Screenreadern kann für Braillezeilen-Ausgaben zwischen verschiedenen Darstellungsmodi gewählt werden, die sich auf die Positionierung der Elemente in der Ausgabe und die über die Elemente dargestellten Informationen auswirken. So kann entweder die flächige Verteilung der Elemente am Bildschirm wiedergegeben oder eine möglichst leerraumfreie Darstellung präsentiert werden. Zu den dargestellten Elementen kann deren reine textuelle Information angezeigt werden oder auch Informationen über Farbgebung und sonstige Gestaltungsoptionen. Würden all diese Informationen ständig angezeigt werden, wäre kein effizientes Arbeiten möglich. Sie stehen dem Benutzer aber jederzeit zur Abfrage bereit.

Neben den verschiedenen Arbeitssituationen sollte bei der Erstellung angepasster Darstellungen auch beachtet werden, dass verschiedene Arbeitsbereiche existieren, die evtl. auch relativ weit auseinander liegen aber dennoch für einen Arbeitsschritt gleichzeitig interessant sein können. Beispielsweise kommt es häufig vor, dass ein Teil eines Dokuments und eine Toolbar gleichzeitig von Interesse sind und somit auch beide in erkennbarer Form im Ausga-

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

bebereich liegen sollten, während andere Bereiche, wie etwa dazwischenliegende Toolbars oder ein Lineal, gerade nicht interessant sind und somit auch keine Vergrößerung benötigen bzw. nicht oder zumindest nicht detailliert dargestellt werden müssen. Abbildung 42 zeigt an Hand eines E-Mail-Fensters beispielhaft, wie sich eine Unterscheidung zwischen Arbeitsbereichen auf die Sichtbarkeit von Elementen auswirken kann. Ziel des Bearbeiters ist hier das Formatieren eines Textes. Bei Vergrößerung des Fensters durch einfache Skalierung auf das 7-fache nehmen allerdings die Bedienelemente für Adress- und Betreff-Angabe im Kopfbereich so viel Platz ein, dass das zur Bearbeitung zu nutzende Ribbon und der Text nicht gleichzeitig erfasst werden können. Durch Einklappen des Kopfbereichs, wie es aus Elementen von Webseiten bereits bekannt ist, kann dieses Problem gelöst werden. Bildschirmvergrößerungssoftware bietet mittlerweile eine weitere Möglichkeit, diesem Problem zu begegnen. Der Benutzer kann sogenannte „Hooked Areas“ bestimmen (siehe Abbildung 43). Dies sind rechteckige Bereiche der Bildschirmausgabe, die für den Benutzer unabhängig vom aktuell sichtbaren Bildausschnitt immer angezeigt werden. Sie überlagern die normale Vergrößerungsausgabe.

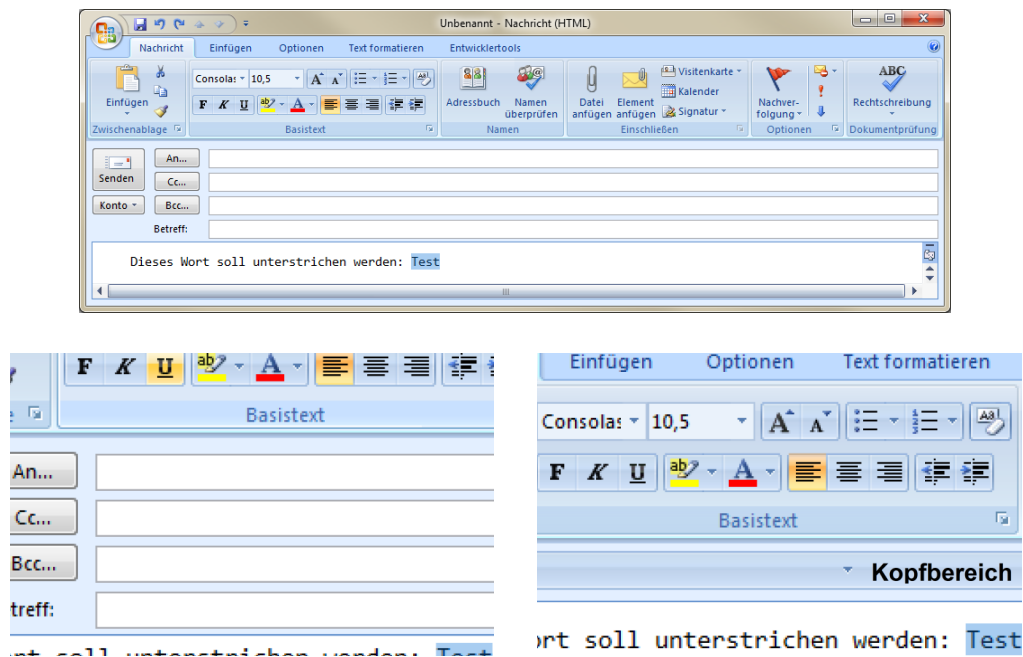


Abbildung 42 Entwurf zur Beachtung von Arbeitsbereichen an Hand eines E-Mail-Fensters
oben: Originaldarstellung; unten links: Vergrößerung (7-fach) des Fensters durch einfache Skalierung (Ribbon und Text können nicht gleichzeitig erfasst werden); unten rechts: Vergrößerung mit „eingeklapptem“ Kopfbereich (Ribbon und Text sind vollständig sichtbar)

3.1 Gleichartigkeit angepasster Darstellungen für beide Zielgruppen



20	Neil	
21	Hazel	
22	Simon	
23	Sarah	
24	Neil	
25	Hazel	
26	Simon	
27	Sarah	D24
28	Neil	
29	Hazel	

Abbildung 43 Beispiel zu *Hooked Areas* aus *Bildschirmvergrößerungssoftware von Dolphin [Dol10]*
Das rot markierte Feld zeigt das Zell-Adress-Feld von Excel und bleibt an dieser Stelle, wenn sich der Benutzer in der Tabelle bewegt, also den sichtbaren Bildschirm-Ausschnitt verschiebt.

3.1.4 Erweiterte Interaktionsmöglichkeiten

Neben den Interaktionsmöglichkeiten, die durch Maus und Tastatur normalerweise in Anwendungen geboten werden, sind für die angepassten Darstellungen bzw. zur optimalen Unterstützung blinder und sehbehinderter Computernutzer weitere Interaktionen nötig.

Wie in den bestehenden Hilfsmitteln bereits etabliert, werden natürlich Funktionen zur Anpassung der Darstellungsattribute, wie Farbe, Schrift und Größenfaktoren benötigt, sowie zusätzliche Scrollfunktionen zum Verschieben des dargestellten Bildschirminhalts, die sich auf die angepasste Darstellung beziehen und nicht auf die Anwendung selbst.

Werden unterschiedliche Ansichtsarten geboten, so muss der Benutzer leicht zwischen diesen wechseln können oder auch in verschiedenen Bereichen der Ausgabefläche parallel anwenden können, um beispielsweise verschiedene Darstellungsoptionen auf Grafiken und deren textuelle Erklärung gleichzeitig anzuwenden. In Übersichtsdarstellungen muss es natürlich möglich sein, ausgehend von einer sehr groben Darstellung zu detaillierteren Darstellungen zu kommen. Solche Explorationstechniken sind im Bereich der taktilen Grafiken bereits selbstverständlich (siehe beispielsweise [Hah04]). Dabei werden bei statischen Grafiken schlicht mehrere Versionen einer Darstellung angeboten, die nacheinander ertastet werden können. Auch die Exploration von Grafiken auf grafisch-taktilen Displays durch schrittweisen Aufbau und Filterung nach bestimmten Elementen und Farben wurden bereits erfolgreich demonstriert [ROE04]. In textuellen Dokumenten können solche Explorationsfunktionen ebenfalls hilfreich sein, wie beispielsweise die Linklisten von Screenreadern zeigen. Bei reiner Bildschirmvergrößerungssoftware werden solche Funktionen bisher nicht geboten.

Auch zusätzliche Navigationsfunktionen, mit denen gezielt bestimmte Bereiche angesprungen werden können, sind wichtig. In Screenreadern sind diese für strukturierte, textuelle Dokumente bereits zahlreich vorhanden und erhöhen die Arbeitseffizienz deutlich. So gibt es Tastenkombinationen zur schnellen Navigation zwischen Überschriften, Absätzen, Tabellenzellen und ähnlichem auch in Dokumenten und Anwendungen, in denen diese sonst nicht verfügbar ist, wie beispielsweise Webbrowsern. In reinen Bildschirmvergrößerungsprogrammen sind diese Funktionen bisher nicht verfügbar, weshalb Nutzer mit stärkerer Sehbehinderung häufig eine Kombination aus Bildschirmvergrößerungssoftware und Screenreader

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

einsetzen. Auch die verschiedenen Bereiche der Bedienoberflächen sollten leicht erreichbar sein. Für Menüleisten und Toolbars bringen Anwendungen meist selbst entsprechende Funktionalitäten mit. Für andere Bereiche, wie etwa die Statuszeile gilt dies aber nicht. Auch hier sind zusätzliche Funktionen nötig. Da zusätzliche Navigationsfunktionen derzeit nur für Screenreader entwickelt werden, gibt es keine besondere Unterstützung für grafische Dokumente. Auch hier könnte zwischen unterschiedlichen Bereich gesprungen werden und intelligente Navigationsfunktionen, die beispielsweise Diagrammpfeilen folgen oder große Leerräume überspringen, könnten sehr hilfreich sein.

3.1.5 Fazit

Insbesondere die Anforderungen und Gestaltungsvorschläge in Abschnitt 3.1.2 zeigen deutlich, dass die Anforderungen an Darstellungen für Blinde und für Sehbehinderte zum größten Teil gleich sind, sofern es sich um zweidimensionale grafische bzw. grafisch-taktile Darstellungen handelt. Es existieren nur wenige Unterschiede. Zum einen kann bei Darstellungen am Monitor mehr mit farbiger Gestaltung gearbeitet werden als am grafisch-taktilen Display, auch wenn diese Farbgebung eventuelle nicht der Originaldarstellung entspricht. Zum anderen ist bei Blinden paralleles Erfassen von Informationen, also auch das Erfassen der gesamten Darstellung „auf einen Blick“ kaum möglich. Bei Sehbehinderten kann noch mehr damit gearbeitet werden. Wobei gerade beim Einsatz starker Vergrößerungen auch kaum mehr davon gesprochen werden kann. Wie die aufgeführten Beispiele zeigen, können beide Seiten positiv aufeinander einwirken. So können bewährte Techniken zur Aufbereitung der grafischen Ausgabe von Bildschirmvergrößerungssoftware auf grafisch-taktile Displays übertragen werden. Zum anderen können Navigations- und Explorationstechniken, die von Screenreadern und taktilen Grafiken bekannt sind, auch die Arbeit mit angepassten Bildschirmdarstellungen erleichtern.

Allerdings bieten existierende Hilfsmittel und Anwendungen für Blinde und Sehbehinderte insgesamt noch keine ausreichende Unterstützung bei der PC-Arbeit. Insbesondere ist es für die Benutzer immer noch schwierig, einen Überblick über die Bildschirmausgabe zu erlangen. Platzeffiziente Darstellungen und intelligente Navigations- und Explorationstechniken können hier Fortschritte bringen. Auch der Zugang zu den Farbinformationen der Originaldarstellung muss insbesondere für Sehbehinderte noch vereinfacht werden, um eine effiziente Kommunikation mit Normalsichtigen bzw. ein effizientes Arbeiten mit sich auf Farben beziehende Erläuterungen zu ermöglichen. Für grafisch-taktile Displays fehlt insgesamt noch ein einheitliches Konzept zur Darstellung und Bedienung von Anwendungen. Nur damit können sie nutzbringend bei der täglichen Arbeit eingesetzt werden. Da hier in keinsten Weise die gleichen grafischen Möglichkeiten zur Verfügung stehen wie bei einem Monitor, reicht es trotz der gleichen Grundfragestellung nicht aus, die Darstellungen von Bildschirmvergrößerungssoftware direkt zu übernehmen. Die etablierten Algorithmen zur Bildaufbereitung und Änderungen am Farbschema der Darstellung können aber hilfreich sein.

3.2 Konzepte zu Darstellungen und Interaktionen

Die folgenden Abschnitte stellen Konzepte zur Umsetzung angepasster Darstellungen und Interaktionen vor. Vor allem die Verbesserung des Überblicks und der Umgang mit Farben werden betrachtet. Aber auch Gesamtkonzepte für die Computer-Arbeit mit grafisch-taktilen Displays und vergrößerter Bildschirmdarstellungen sowie zur Zusammenarbeit zwischen Nutzern grafisch-taktiler Displays und Sehenden werden vorgestellt.

3.2.1 Strategien zur Erzeugung angepasster Darstellungen

Zur Umsetzung angepasster Darstellung können verschiedene Mittel zum Einsatz kommen, die auf verschiedenen Ebenen im Darstellungsprozess ansetzen und auch kombiniert eingesetzt werden können. Folgende Anpassungsstrategien lassen sich unterscheiden:

- Direkte Abstimmung einer Anwendung auf bestimmte Ein- und Ausgabegeräte
- Beeinflussung des Darstellungsprozesses einer Anwendung
- Aufbereitung der rasterisierten Programmausgabe
- Erzeugung einer neuen Ausgabegrafik aufgrund jeglicher über den Darstellungsprozess und den Darstellungsinhalt ermittelbaren Informationen

Alle Anpassungsstrategien haben Vor- und Nachteile. Bei der direkten Abstimmung der Programmausgabe können die meisten semantischen Informationen über den darzustellenden Inhalt, auch die, die sich nur in den Köpfen der Entwickler befinden, in die Gestaltung der angepassten Darstellung einfließen. Auch speziell auf den Inhalt abgestimmte Interaktionstechniken lassen sich gut umsetzen. Allerdings kann so nur schlecht ein Gesamtkonzept für Bedienung und Darstellung umgesetzt werden. Alle Anwendungsentwickler müssten sich genau daran halten und bei Änderungen ihre Anwendung entsprechend anpassen. Auch die Unterstützung verschiedener Ein- und Ausgabegeräte macht viel Arbeit. Fast alle für grafisch-taktile Displays bisher vorgestellten Anwendungen fallen in diese Kategorie. So beispielsweise auch der speziell für die Stuttgarter Stiftplatte entwickelte taktile Webbrowser von Rotard et al. [RKE05]. Bei diesem zeigte sich auch schnell ein weiterer Nachteil solcher Anwendungen. Soll eine spezielle Anwendung für Blinde und Sehbehinderte als Ersatz für bestehende „normale“ Desktopanwendungen dienen, so muss diese auch möglichst alle Funktionen der ersetzten Anwendung bieten. Dazu müsste eine Parallelentwicklung mit gleichem Personal- und Zeitaufwand durchgeführt werden, was unrealistisch ist. Zudem leidet dabei meist der Aspekt der Förderung der Zusammenarbeit mit Normalsichtigen.

Wird lediglich die Darstellung einer bestehenden Anwendung geändert, wie bei Austausch von Farben und Schriften durch Windows-Designs, können deren Funktionalitäten genutzt werden und die Nutzbarkeit für Normalsichtige bleibt meist erhalten. Wird die Beeinflussung des Darstellungsprozesses nicht direkt von den Anwendungsentwicklern umgesetzt, was bezüglich eines durchgängigen Darstellungs- und Bedienkonzepts die gleichen Nachteile

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

brächte wie die direkte Anpassung, so sind die Möglichkeiten zur Anpassung natürlich von den angebotenen Schnittstellen abhängig. Anwendungen und Dokumente, die mit CSS-ähnlichen Styledefinitionen arbeiten, bieten hier die besten Möglichkeiten, wie auch der im Rahmen diese Arbeit entwickelte Nachfolger des taktilen Webbrowsers auf Basis von Mozilla Firefox zeigte (siehe [RTE08] und [Spi07]). Bei diesem wurde die Darstellung von Webseiten durch Anpassung der Schrift und Seitenbreite so verändert, dass sich das durch die Firefox-Rendering-Engine erzeugte Ergebnis 1-zu-1 auf das grafisch-taktile Display übertragen ließ. Zur Erzeugung der Ausgabegrafik wurde der dargestellte Text samt der Ausgabeposition abgefragt und in Brailleschrift an die Zielposition in der Grafik geschrieben, die durch einfache Skalierung der Ausgabeposition ermittelt wurde. Hier wurde also die Beeinflussung des Darstellungsprozesses mit der Erzeugung einer eigenen Ausgabegrafik kombiniert. Allerdings wurde dabei die Darstellung am Bildschirm sehr stark geändert. Zwar war sie für Normalsichtige noch nutzbar, allerdings nicht in der gewohnten Form (siehe Abbildung 44). Neben der Abhängigkeit von Schnittstellen, die evtl. auch je nach Anwendung unterschiedlich sind und unterschiedlich angesprochen werden müssen, ist die evtl. starke Veränderung gegenüber der Originaldarstellung ein weiterer Nachteil der Beeinflussung des Darstellungsprozesses.



Abbildung 44 Ablauf der Anpassung im taktilen Firefox [Spi07]
Beeinflussung des Darstellungsprozesses gefolgt von Erzeugung eigener Ausgabegrafik

Eine eigene Ausgabegrafik zu erzeugen, bringt die meiste Gestaltungsfreiheit mit sich. Objekte können in beliebiger Darstellung an beliebiger Position ausgegeben werden. Dabei können beliebige Objekteigenschaften besonders hervorgehoben werden (wie beispielsweise der Fokus in Bildschirmvergrößerungssoftware) oder auch weggelassen werden, um leichter erkennbare reduzierte Darstellungen zu erzeugen. Auf diese Weise kann am ehesten ein durchgängiges Darstellungskonzept umgesetzt werden, das auf die Bedürfnisse Blinder und

Sehbehinderter zugeschnitten ist. Aktuelle Screenreader arbeiten nach diesem Prinzip (auch wenn hier statt Grafik, Sprach- und Braillezeilen-Ausgabe erzeugt wird). Allerdings müssen die für die Darstellung interessanten Informationen auch über entsprechende Schnittstellen ermittelbar sein. Je nach Schnittstelle stehen hier auch viele semantische Informationen zur Verfügung, allerdings nie so viele, wie bei der direkten Umsetzung der angepassten Darstellung durch die Anwendungsentwickler selbst.

Völlig unabhängig von Zugriffsschnittstellen ist die Aufbereitung der rasterisierten Programmausgabe. Sie ist somit für jedes Programm, das am Bildschirm dargestellt wird, nutzbar und bietet auch die zuverlässigsten Informationen über die Originaldarstellung. Allerdings stehen hier die wenigsten semantischen Informationen zur Verfügung. Lediglich die Farbverteilung kann direkt ermittelt werden. Für alle anderen Informationen sind eventuell aufwändige Bildanalysen nötig. Aktuelle Bildschirmvergrößerungssoftware basiert zum Großteil auf diesem Vorgehen.

Angepasste Darstellungen können nur funktionieren, wenn auch entsprechende Navigations- und Explorationsfunktionen existieren. Nur durch diese kann der Benutzer mit der Darstellung interagieren und so letztlich die dargestellte Anwendung bedienen. Navigation und Exploration sind ebenso wie die Darstellung auf unterschiedlichen Ebenen zu betrachten. Sie können sich auf Objekte innerhalb der Darstellung beziehen oder auf deren grafischen Eigenschaften.

3.2.2 Erleichterung der Erfassung aller interessanten Informationen

Abschnitt 3.1 zeigt deutlich, dass die Erfassung einer grafischen Oberfläche umso besser möglich ist, je leichter sich der Nutzer einen Überblick dazu verschaffen kann. Dies wiederum ist umso leichter, je weniger Interaktionen nötig sind, um die für die Arbeitssituation gerade interessanten Informationen zu erfassen. Ziel einer angepassten Darstellung sollte es also immer sein, die gerade interessanten Eigenschaften aller gerade interessanten Objekte im gerade interessanten Detailgrad in einer leicht erfassbaren Darstellung auf einer möglichst kleinen Fläche (im Idealfall der einfachen Ausgabefläche) darzustellen.

Hierzu können verschiedene Layout-Optionen, Filter-Möglichkeiten, verkürzte Darstellungen und Zoom-Strategien dienen. Ist es nicht möglich, alle gewünschten Objekte im nötigen Detailgrad und der nötigen Vollständigkeit im einfachen Ausgabebereich erkennbar darzustellen, müssen Navigations- und Explorationsfunktionen angeboten werden, die den Benutzer möglichst gut bei der Erfassung der interessanten Informationen unterstützen. Die folgenden Abschnitte stellen verschiedene Konzepte dazu vor.

3.2.2.1 Layout-Möglichkeiten

Zur Gestaltung einer angepassten Darstellung stehen grundsätzlich drei Möglichkeiten zur Verfügung – layoutgetreue, layoutunabhängige und layoutgerechte Darstellung.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Layoutgetreue Darstellungen geben die absolute Positionierung und Ausdehnung von Objekten (und den Leerräumen zwischen ihnen) in der originalen Bildschirmausgabe wieder. Sie eignen sich gut, zur Orientierung innerhalb der Fläche, beispielsweise der Erfassung von Datenpunkten in einem Diagramm, und um Größenverhältnisse und Abstände einzuschätzen. Da der Darstellungsplatz für ein Objekt aber nur durch gleichmäßige Skalierung der gesamten Darstellungsfläche vergrößert werden kann, können layoutgetreue Darstellungen sehr groß werden, wenn viele Details oder sehr kleine Objekte erkennbar sein sollen. In diesem Fall werden Scrolloperationen in beiden Richtungen nötig.

Bei layoutunabhängigen Darstellungen dagegen, können Positionierung und Ausdehnung von Objekten frei bestimmt werden. Damit können Darstellungen geschaffen werden, die die zur Verfügung stehende Ausgabefläche optimal nutzen. Sie eignen sich besonders dann, wenn geometrische Informationen nicht interessant sind, wie beispielsweise beim fortlaufenden Lesen von Texten oder bei der Auswahl einer Menüoption. Sie geben dem Benutzer allerdings nicht direkt zuverlässige Informationen über die Positionierung der dargestellten Objekte in der Originalausgabe. Layoutbezogene Aufgaben sind somit nicht oder nur sehr schwer bearbeitbar. Weiterhin kann dies die Kommunikation mit Sehenden (oder auch Nutzern kleinerer oder größerer Displays) und das Verständnis von Erläuterungen zu Programmen behindern. Außerdem bedeutet es eine starke Umstellung des Benutzers, falls er die Anwendung in der Originaldarstellung schon gewohnt ist.

Als Kompromiss zwischen Erkennbarkeit des Layouts und kompakter Darstellung können layoutgerechte Darstellungen genutzt werden. Hierbei kann die Ausdehnung der Objekte entsprechend der darzustellenden Informationen frei gewählt werden. Die Positionierung geschieht aber mit Beachtung des ursprünglichen Layouts, so dass die relative Positionierung der Objekte zueinander erhalten bleibt. Dies ermöglicht kompakte Darstellungen, die gleichzeitig die Orientierungsmöglichkeiten erhalten (siehe Abbildung 45). Die Möglichkeit, Positionen relativ bestimmen zu können (im Sinne von „ganz links“ oder „rechts davon“) reicht zur Orientierung und Kommunikation mit anderen meist aus.

Bei der Erzeugung von Darstellungen in den drei genannten Layout-Möglichkeiten lassen sich potentiell alle drei in Abschnitt 3.2.1 beschriebenen Anpassungsstrategien nutzen. Einige Randbedingungen müssen allerdings erfüllt sein.

Für layoutgetreue Darstellungen muss bei Anpassung des Darstellungsprozesses und Aufbereitung der rasterisierten Darstellung darauf geachtet werden, dass die Begrenzungen der darzustellenden Objekte nicht verändert oder überschritten werden. Die Erzeugung eigener Rasterisierungen ist nur möglich, wenn die genauen Positionen und Ausdehnungen der darzustellenden Objekte verfügbar sind.

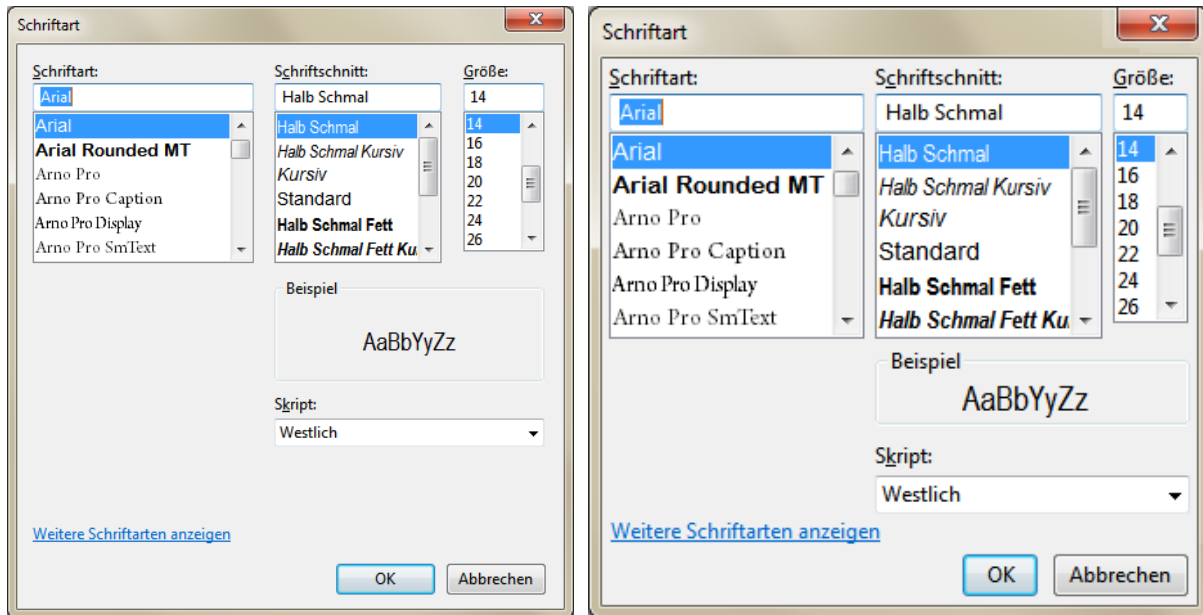


Abbildung 45 Beispiel zur layoutgerechten Darstellung

Durch bloße Verringerung der Leerräume konnte bei gleicher Höhe eine Vergrößerung der Schrift um den Faktor 1,3 erreicht werden. Bei grafisch-taktilen Displays zeigt sich der Effekt aufgrund der kleinen Schriftgröße noch deutlicher. Die relative Positionierung der Elemente blieb erhalten.

Für layoutunabhängige Darstellungen bieten eigene Rasterisierungen natürlich die besten Möglichkeiten, sofern die darzustellenden Informationen in passender Form zur Verfügung stehen. Vorhandene Darstellungsprozesse lassen sich meist nicht oder nur schwer soweit beeinflussen, dass alle Vorteile von layoutunabhängigen Darstellungen genutzt werden können. Bei modernen Oberflächen und Webseiten gibt es aber gute Möglichkeiten dazu. Und selbst durch Aufbereitung der rasterisierten Darstellung können durch geschicktes Zerstückeln und Neuzusammensetzen der Rastergrafik noch layoutunabhängige Darstellungen entstehen. Dazu können wie beispielsweise [GS90] für die Exploration von Dokumenten zeigte, Algorithmen zur Erkennung von Leerräumen eingesetzt werden, an denen die Rastergrafik aufgetrennt werden kann. Allerdings können die dabei nötigen Analysen sehr aufwändig sein und benötigen bei Darstellungen mit Transparenzen und Schatteneffekten, wie sie heute bei Bedienoberflächen und Grafiken üblich sind, gute Heuristiken, um zuverlässig arbeiten zu können. Die Einbeziehung des Nutzers und von Informationen über den Ursprung der Rastergrafik können hier helfen. So kann der Nutzer beispielsweise Bereiche markieren, die zusammenhängende Informationseinheiten darstellen, wie etwa die Abschnitte eines Ribbons. Und die Kenntnis darüber, welche Anwendung in der Grafik dargestellt ist, kann Auskunft dazu geben, wie Trennelemente in der Bedienoberfläche gestaltet sind und von welcher Art das dargestellte Dokument ist.

Layoutgerechte Darstellungen lassen sich besonders gut umsetzen, wenn der Darstellungsprozess die Beeinflussung von Leerräumen bzw. Abständen von Elementen zueinander zulässt und die Größe von Elementen bzw. deren Darstellungsplatz automatisch an den dargestellten Inhalt anpasst. So können die Layoutfunktionen des Darstellungsprozesses direkt genutzt werden. Die Positionen und Ausdehnungen der Elemente dürfen dazu natürlich

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

nicht in absoluten Koordinaten festgelegt worden sein. Soll eine layoutgerechte Darstellung durch eigene Rasterisierung umgesetzt werden, so müssen wie bei der layoutgetreuen Darstellung die genauen Positionen und Ausdehnungen in der Originaldarstellung bekannt sein, um die dargestellten Informationen so anordnen zu können, dass sowohl horizontale als auch vertikale Ausrichtungslinien zur Orientierung erhalten bleiben.

Auch durch Aufbereitung von rasterisierten Darstellungen können nützliche layoutgerechte Darstellungen entstehen, wenn auch manchmal unbewusst (siehe dazu Abschnitt 3.2.3.4 angepasste Skalierung). Hier können ähnliche Vorgehen zum Einsatz kommen wie bei der Erzeugung von layoutunabhängigen Darstellungen. Allein die einfache Entfernung von Leerräumen kann wie bei Bedienoberflächen auch gute Ergebnisse bringen und ist bei einfachen Grafiken (wie etwa Abbildung 46) effizient durchführbar.

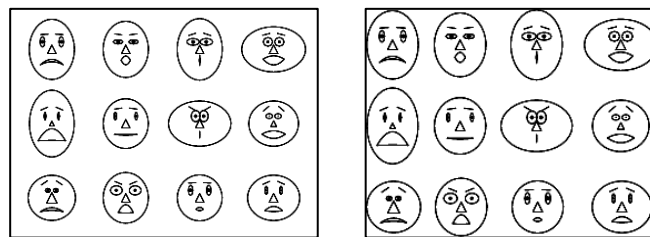


Abbildung 46 Beispiel für den Nutzen der Entfernung von Leerräumen in Rastergrafiken

3.2.2.2 Filterung

Die Möglichkeiten zur Filterung von grafischen Oberflächen, also zur Einschränkung der dargestellten Objekte und Eigenschaften sind stark davon abhängig, auf welcher Ebene eine Aufbereitung ansetzt. Steht nur eine Rastergrafik zur Aufbereitung zur Verfügung, kann auch nur nach rein grafischen Eigenschaften, wie der Farbverteilung, gefiltert werden oder nach solchen, die sich durch Algorithmen aus den grafischen Eigenschaften ableiten lassen wie Kanten oder durch OCR erkannten Texten. Steht eine ausführliche semantische Beschreibung der Objekte auf dem Bildschirm zur Verfügung, können vielfältige Eigenschaften zur Filterung herangezogen werden, auch solche, die in der Bildschirmausgabe nicht erkennbar sind, wie beispielsweise textuelle Beschreibungen zu SVG-Elementen. Besitzt eine Anwendung bereits eigene Filtermöglichkeiten, können natürlich auch diese im Sinne der Anpassung des Darstellungsprozesses eingebunden werden. Wenn beispielsweise bekannt ist, mit welcher Farbe die Ergebnisse einer Textsuche in einer Darstellung markiert werden, so genügt nach Ausführung der Suche eine einfache Farbsuche zur Markierung der Suchergebnisse in der angepassten Darstellung.

Natürlich können objekt- und grafikbezogene Informationen zur Filterung kombiniert werden. So können beispielsweise die Informationen zu Elementtypen von Accessibility-Schnittstellen mit den aus der Bildschirmgrafik ermittelbaren Farbinformationen für die Suche nach roten Schaltflächen kombiniert werden, um die Schließen-Schaltfläche hervorzuheben. Bei Kombination unterschiedlicher Filterungen, egal ob rein objektbezogen, rein grafisch oder gemischt, muss immer beachtet werden, in welcher Weise der Informationsraum durch die

Filterung eingeschränkt wird. Nicht immer können Filterungen rein sequenziell abgearbeitet werden, also auf den Ergebnissen der vorigen Filterung aufbauen, selbst wenn sie logisch Und-verknüpft sind. Beispielsweise stehen im Ergebnisbild einer Kantenfilterung meist keine Farbinformationen mehr zur Verfügung, so dass auf dieser Basis eine Farbfilterung nicht mehr möglich ist. Sollen beispielsweise zum Finden der Schließen-Schaltfläche rote Bereiche in einer Kantendarstellung hervorgehoben werden, muss die Ausgangsgrafik normalerweise einmal nach Kanten und einmal nach Farben gefiltert. Die Ergebnisgrafiken müssen dann passend kombiniert werden. Um die Kombination verschiedener Filterergebnisse zu erleichtern, sollten deshalb in Ergebnisgrafiken von Aufbereitungen und eigenen Rasterisierungen Bildpunkte, die nicht informativer Bestandteil der Grafik sind, als transparent definiert sein.

Eine wichtige Fragestellung bei der Filterung liegt darin, in welcher Weise der Benutzer die zu benutzenden Parameter bestimmen kann. Die für Benutzer einfachste und für Entwickler sicherste Möglichkeit ist, einige Auswahlmöglichkeiten fest vorzugeben, wie beispielsweise eine Auswahl an Farben oder Elementtypen, nach denen gefiltert werden kann. So kann ein Entwickler eine dafür optimal passende und für den Benutzer leicht verständliche Bedienoberfläche erstellen. Dem Benutzer wird schnell klar, welche Möglichkeiten bestehen und wie er diese angeben kann. Durch Kombination mehrerer Auswahlmöglichkeiten können auch komplexere Filterungen stattfinden. Die Möglichkeiten sind aber natürlich beschränkt. Eine offene Schnittstelle, in der der Benutzer uneingeschränkt Eigenschaftsnamen und –werte, nach denen gefiltert werden soll, angeben und durch logische Verknüpfungen verbinden kann, bietet mehr Möglichkeiten. Sie ist aber auch schwieriger zu bedienen, da der Nutzer wissen muss, welche Eigenschaften überhaupt zur Verfügung stehen und wie deren Werte anzugeben sind. In beiden Fällen sollten Filter zur leichteren Bedienung immer sowohl positiv im Sinne von „diese Objekte oder Eigenschaften darstellen“ als auch negativ im Sinne von „diese Objekte oder Eigenschaften ausblenden“ formuliert werden können.

Nicht immer weiß der Benutzer von sich aus, nach welchen Eigenschaften mit welchen Werten er überhaupt filtern will. Beispielsweise bei der Betrachtung eines Diagramms mit mehreren Datenreihen oder Funktionen kann es für den Nutzer sinnvoll sein, die verschiedenen Informationseinheiten zunächst einzeln zu erkunden. Um dies zu erreichen, muss der Benutzer zunächst ermitteln können, wodurch sich die Datenreihen oder Funktionen unterscheiden und in wie weit nach diesen Unterschieden gefiltert werden kann. Dem Nutzer sollten also Möglichkeiten zur Verfügung stehen, die Eigenschaften von Teilen der Ausgabe zu ermitteln und dazugehörige Filtereigenschaften zu erkennen. Dabei sollten auch mehrere Objekte bzw. Abschnitte in der Ausgabe auf einmal betrachtet werden können, wobei Gemeinsamkeiten und Unterschiede in den Eigenschaften kenntlich gemacht werden sollten. Dies kann die Bestimmung der nötigen Filterkonfiguration zur Erfüllung einer Arbeitsaufgabe, wie beispielsweise der Auswertung von Datenreihen, deutlich erleichtern. Ist der Benutzer sich bereits im Klaren über die zu verwendende Filtereigenschaft, nur nicht deren Wert, so kann auch ein vereinfachtes Vorgehen zum Einsatz kommen. Der Benutzer kann die Filtereigenschaft wählen und einen Bereich in der Ausgabe, aufgrund dessen der Wert der Eigenschaft gefüllt wird. So können beispielsweise schnell alle Datenpunkte eines Diagramms angezeigt

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

werden, die in der gleichen Farbe dargestellt sind wie ein ausgewählter Datenpunkt. Natürlich können auch mehrere Eigenschaften Verwendung finden. Auch hier kann dem Benutzer ermöglicht werden, aus einer vorgegebenen Liste von „Filtermodi“ auszuwählen.

Wann immer der Benutzer eine Bedienoberfläche benutzt, um Filtereinstellungen zu tätigen, sollte diese auch Normalsichtigen gut zugänglich sein. Bei Darstellungen für Sehbehinderte ist dies meist kein Problem, da die Oberflächen auch für den Sehbehinderten am Monitor angezeigt werden müssen. Bei Blinden tendieren Entwickler häufig dazu, die nötigen Dialoge nur für den Benutzer selbst am grafisch-taktilen Display zu präsentieren, da sie augenscheinlich nur für den Benutzer interessant sind. Dies ist aber der Zusammenarbeit mit Sehenden nicht dienlich. Werden die Dialoge dagegen auch am Monitor angezeigt, wie dies bei Screenreadern üblich ist, so können Sehende die Arbeit des Blinden verfolgen und eventuell auch bei der Auswahl geeigneter Filter helfen.

3.2.2.3 Verkürzte Darstellungen

Eine weitere Möglichkeit, die zur Verfügung stehenden Ausgabefläche für die aktuelle Arbeitsaufgabe besser zu nutzen, sind verkürzte Darstellungen, also Darstellungen, in denen ein Teil der Informationen nur unvollständig dargestellt ist oder gar gänzlich entfällt. So können beispielsweise Werkzeugleisten in einer Anwendung ausgeblendet oder ähnlich wie in Abbildung 42 zu einem ausklappbaren Bereich zusammengefasst werden, um Dokument und Menü näher aneinanderzurücken. In Webseiten und sehr komplexen Oberflächen sind solche Mechanismen bereits weit verbreitet. In anderen Dokumenten findet man sie eher selten. Dabei können sie sogar in Grafiken sinnvoll eingesetzt werden. So können, wie beispielsweise Abbildung 47 zeigt, horizontal oder vertikal durchgehende Bereiche aus Grafiken entfernt werden um wie hier den Schnittpunkt einer Funktionsgraph-Darstellung näher an die Achse zu bringen und so die Koordinaten des Schnittpunkts leichter ablesen zu können.

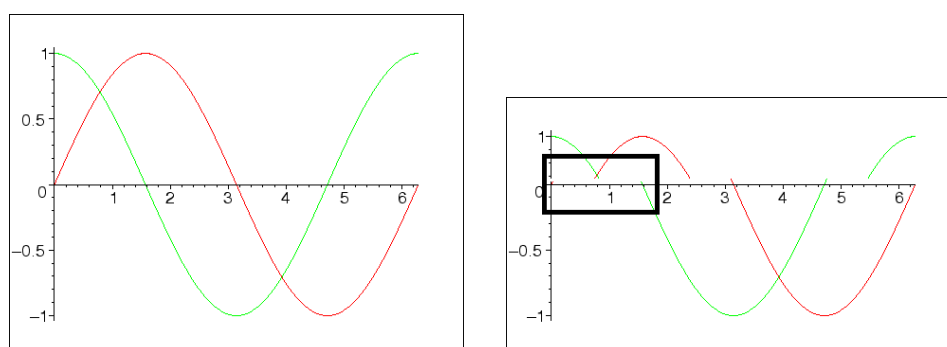


Abbildung 47 Beispiel zur verkürzten Darstellung von Grafiken
links: Darstellung von Sinus (rot) und Cosinus (grün) im Koordinatensystem (1. und 4. Quadrant, schwarz) aus Online-Lehrmaterialien zur Mathematik [mat]
rechts: verkürzte Darstellung zum Ablesen des x-Wertes des ersten Schnittpunkts zwischen Sinus und Cosinus mit Markierung des auf dem grafisch-taktilen Display sichtbaren Bereichs

Eine Verkürzung von Rastergrafiken, wie oben gezeigt, kann natürlich nur benutzergesteuert sinnvoll durchgeführt werden. Folgende Vorgehensweisen sind hier denkbar:

- Nutzung von Höhe und Breite des dargestellten Bildausschnittes als Höhe bzw. Breite des auszublenhenden horizontalen bzw. vertikalen Bereichs (So muss der Nutzer nur eine Aktion ausführen. Eventuell kann der auszublenhende Bereich um weitere Höhen oder Breiten ergänzt werden.)
- Der Nutzer wechselt in einen „Verkürzungsmodus“ und markiert eine Stelle im dargestellten Bildausschnitt, scrollt dann zur Stelle, die das Ende der Verkürzung sein soll, markiert diese und veranlasst die Ausblendung.

Bei der Verkürzung von Rastergrafiken sollte, wie in Abbildung 47 zu sehen, darauf geachtet werden, dass dem Nutzer bewusst ist, dass es sich um eine verkürzte Darstellung handelt. Dies kann beispielsweise durch Ergänzung einer dicken Linie mit bestimmtem Muster zwischen den zusammengeführten Bereichen erfolgen oder durch Freilassen eines Leerraum. Werden die Bereiche direkt zusammengeführt, könnte dies zu Fehlinterpretationen führen. Im Beispiel könnte der Nutzer davon ausgehen, dass die Y-Achse direkt durchgängig ist und so eine falsche Koordinate für den Schnittpunkt ablesen. Ähnliche Kennzeichnungen sollten auch in Oberflächen erfolgen, sonst läuft der Nutzer Gefahr, Teile der Oberfläche zu übersehen, wodurch er in seiner Arbeit eingeschränkt werden könnte.

Weitere Möglichkeiten der Verkürzung sind die Ersetzung von Elementen oder ganzen Bereichen, wie beispielsweise einem Login-Bereich oder einer Navigationsleiste auf einer Webseite, durch leicht identifizierbare Symbole und die Kürzung bekannter Texte. So könnte beispielsweise in dem in Abbildung 30 gezeigten Firefox-Menü „Ansicht“ der Text „Seitenquelltext anzeigen“ durch „Quelltext anzeigen“ ersetzt werden, wodurch sich das gesamte Untermenü verschmälern würde. Somit wäre im Bildausschnitt ein größerer Teil des Untermenüs zu „Zeichenkodierung“ sichtbar. Noch bekanntere Texte können auch deutlicher abgekürzt werden, so beispielsweise „Bearbeiten“ zu „Bearb.“. Somit muss der Nutzer auch nicht unter einer Sprache mit langen Bezeichnungen leiden (im Englischen stände statt „Bearbeiten“ nur „Edit“). Abbildung 48 zeigt ein Beispiel dazu mit dem Menü des Mozilla Firefox. Auch in weniger leicht beeinflussbaren Anwendungen können solche Verkürzungen oft durch Änderung von Lokalisierungsdateien vorgenommen werden.

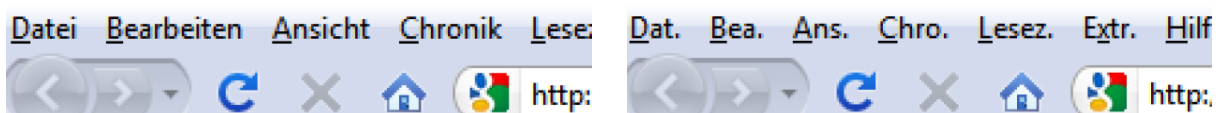


Abbildung 48 Auswirkung von Textverkürzungen im Menü des Mozilla Firefox
(8-fache Vergrößerung auf einem Monitor mit 1920 Pixeln in der Breite)

3.2.2.4 Zoom

Auch mit dem kompaktesten Layout lassen sich Scroll-Operationen nicht gänzlich vermeiden. Sollen viele Detailinformationen erkennbar angezeigt werden, entsteht schnell eine Darstellung, die die zur Verfügung stehende Ausgabefläche übersteigt. Dennoch soll eine Orientierung innerhalb der Darstellung gut möglich sein. Zum einen kann dazu die Anzeige der Scroll-Position dienen. Zum anderen können kleinere Darstellungen als Übersichtsdarstellungen herangezogen werden, von denen aus der Nutzer zur größeren Darstellung wechseln kann. Hierbei können verschiedene Darstellungs- und Steuerungs-Strategien zum Einsatz kommen, die unter dem Begriff „Zoom“ zusammengefasst werden können.

Die technisch einfachste Darstellungs-Strategie ist der auch bei Bildschirmvergrößerungssoftware eingesetzte grafische Zoom. Dies ist eine Form des geometrischen Zooms, was bedeutet, dass die zur Verfügung stehende Ausgabefläche mit einem Faktor skaliert und die Darstellung an die resultierende Größe angepasst wird, was beim grafischen Zoom durch einfache Bildskalierung geschieht. Beim geometrischen Zoom bleiben also die Proportionen von Objektgrößen und Abständen immer gleich. Abbildung 49 veranschaulicht dies. Zudem kann der ursprünglich gezeigte Bildausschnitt durch ein Rechteck markiert werden (sogenannte Minimap-Darstellung, siehe Abschnitt 3.1.2.14). Bei grafisch verkleinerten Darstellungen können ohne weitere Aufbereitung allerdings nur große, gleichartige Bereiche erkannt werden. Bei sehr feingliedrigen Darstellungen ist somit keine Orientierung bezüglich der Darstellungsinhalte möglich, lediglich eine Orientierung bezüglich der absoluten Position innerhalb der Darstellung.

Um sich in einer kleinen Darstellung auch an den Darstellungsinhalten orientieren zu können und evtl. gezielt zu bestimmten Inhalten navigieren zu können, eignet sich besser ein semantischer Zoom [PF93] (siehe Abbildung 50). Hierbei wird die Detailstufe der Darstellung angepasst, um eine gut erkennbare Darstellung zu schaffen. Dazu können zum Beispiel kleine Objekte zu größeren Regionen vereint werden oder durch symbolhafte Darstellungen ersetzt werden. Der semantische Zoom kann als geometrischer oder als nicht-geometrischer Zoom umgesetzt werden. Das heißt, die Darstellungsgröße kann genauso skaliert werden, wie beim grafischen Zoom, so dass die Darstellung an die zur Verfügung stehende Größe angepasst wird. Es kann aber auch umgekehrt eine Anpassung der Ausgabefläche an die für das gewählte Darstellungsdetail nötige Größe stattfinden. Das Verhältnis von Höhe und Breite der kleinen Darstellung muss dabei nicht dem Verhältnis der großen Darstellung entsprechen. Somit ist beim nicht-geometrischen Zoom die Orientierung bezüglich der absoluten Position innerhalb der Darstellung nur schlecht möglich. Die Darstellung eines Orientierungsrechteckes empfiehlt sich nicht. Vielmehr sollte das Objekt oder die Objektgruppe hervorgehoben werden, das/die in der großen Darstellung gerade sichtbar war.

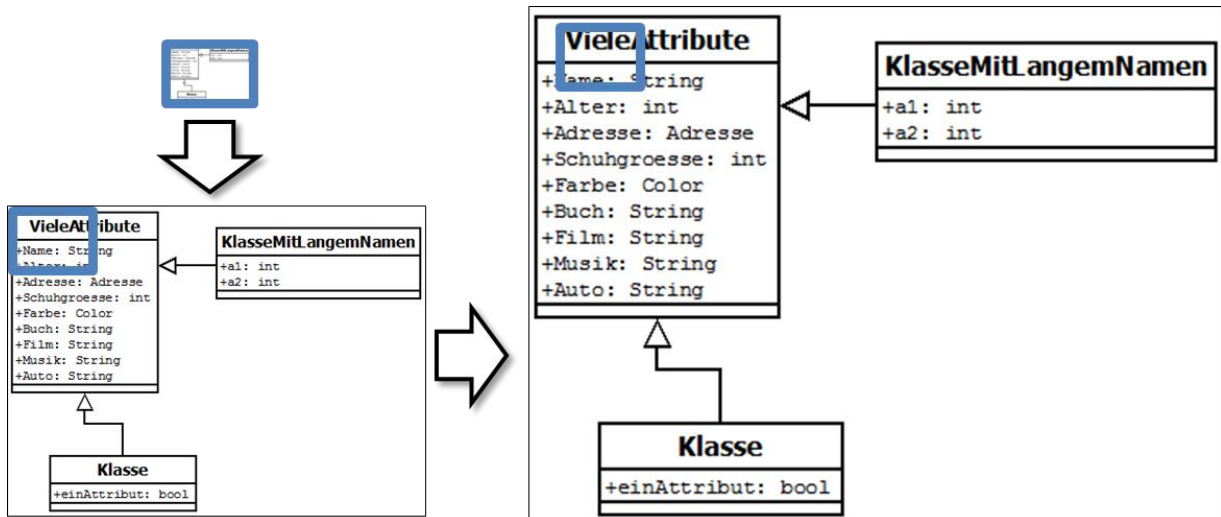


Abbildung 49 Darstellung des grafischen Zooms am Beispiel eines Klassendiagramms

Die Darstellung soll eine für Sehbehinderte nötige Vergrößerung ausgehend von der Normaldarstellung veranschaulichen. Dabei ist das Bild links oben die Normaldarstellung. Das Bild darunter zeigt eine Vergrößerung soweit, dass die Namen der Klassen lesbar sind. Im rechten Bild wurde der Zoomfaktor weiter erhöht, um auch die Attribute lesen zu können. Die Größe des Originalbildes ist jeweils mit einem blauen, dicken Rechteck markiert, um den nötigen Scrollaufwand zu verdeutlichen. Die Größen- und Abstandsverhältnisse bleiben immer erhalten. Man kann deutlich erkennen, dass der Nutzer aus der Normaldarstellung keine Informationen gewinnen kann. In der ersten Vergrößerungsstufe ist, zur Erkennung aller Klassennamen, ein hoher Scrollaufwand nötig.

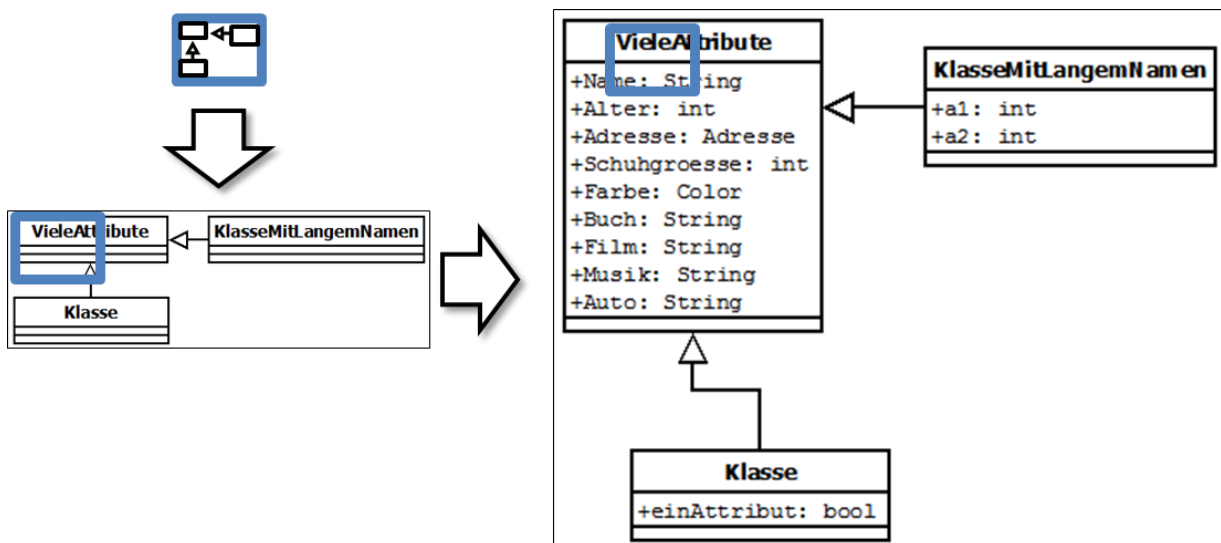


Abbildung 50 Nicht-geometrischer, semantischer Zoom am Beispiel eines Klassendiagramms

Die Darstellung veranschaulicht den gleichen Prozess der Vergrößerung wie in Abbildung 49. Dabei wurde die Darstellung in der kleinsten Ansicht (einfache Monitorgröße) so aufbereitet, dass für den Nutzer Informationen gewinnbar sind. Man kann deutlich drei Rechtecke (Klassen) mit Vererbungspfeilen erkennen. Die Darstellung zur Ermittlung der Klassennamen beschränkt sich auf die wesentlichen Informationen, die Struktur des Diagramms und die Klassennamen. So ist der Scrollaufwand deutlich geringer als in Abbildung 49. Auch die größte Darstellung entspricht nicht exakt der Darstellung in Abbildung 49. Hier wurde darauf verzichtet, die Klassennamen weiter zu vergrößern. Dies spart Platz und verringert den Scrollaufwand. Derartige Veränderungen sind natürlich nur bei Vorliegen entsprechender Informationen über das Klassendiagramm möglich. Weiterhin ist erkennbar, dass sich die Größen- und Abstandsverhältnisse zwischen den Elementen verändern. Die Orientierung anhand der Position in der Darstellung ist damit schwierig.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Geometrische Zoom-Strategien eignen sich auch dazu, schrittweise durch Änderung des Skalierungsfaktors von einer in die Ausgabefläche eingepassten Darstellung (mit Anzeige eines Orientierungsrechteckes) in eine größere Darstellung überzugehen, in der die nötigen Details erkennbar sind. Dabei sollte beachtet werden, dass Zoomstufen, die dem Benutzer keine neuen Informationen geben, übersprungen werden können, um eine schnelle Exploration der Ausgabe zu ermöglichen und unnötiges Suchen nach Veränderungen zu beschränken. In Bildschirmvergrößerungssoftware werden aus diesem Grund nur eine bestimmte Menge vorgegebener Skalierungsfaktoren angeboten. Unnütze Zoomstufen könnten aber auch dynamisch übersprungen werden, in dem durch einen Bildvergleich ermittelt wird, ob die neue Darstellung eine Mindestanzahl von Änderungen enthält. Natürlich müsste dies sehr effizient implementiert sein, um dem Benutzer eine gute Arbeitsgeschwindigkeit zu ermöglichen. Beim semantisch Zoom können die Zoomstufen auch abhängig von den semantischen Darstellungen gewählt werden, so dass der Benutzer beispielsweise von einer Darstellung, die nur Symbole für alle Objekte anzeigt, direkt zu einer Darstellung wechseln kann, in der die genauen Begrenzungen aller Elemente erkennbar sind, und weiter zu einer Darstellung, in der auch alle Texte lesbar sind. Dieses Vorgehen eignet sich natürlich auch für einen nicht-geometrischen, semantischen Zoom.

Neben der Auswahl von Zoomstufen kann die Steuerung eines Zooms auch über die Hierarchie einer Darstellung erfolgen, also durch die Auswahl eines Objektes, das als Wurzelobjekt der Darstellung dienen soll und die Darstellung eventuell bestimmt. Abbildung 51 veranschaulicht dies. Bezüglich der Steuerung-Strategien kann also zwischen stufenbestimmtem Zoom und hierarchischem Zoom unterschieden werden. Beim hierarchischen Zoom muss darauf geachtet werden, dass die nächstgrößere Darstellung immer mindestens eine weitere Hierarchiestufe mit all ihren Elementen zeigt, damit die Zoomsteuerung uneingeschränkt fortgesetzt werden kann.

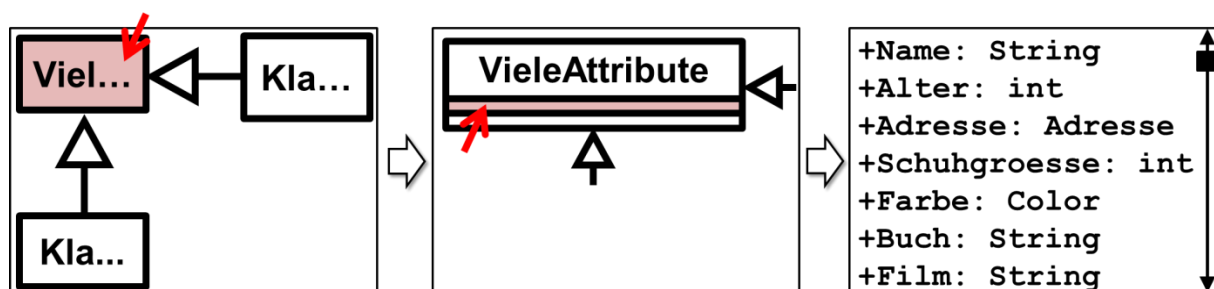


Abbildung 51 Darstellung des hierarchischen Zooms in einem Klassendiagramm

Der Nutzer sieht zunächst eine Grobstruktur des Diagramms. Er wählt eine Klasse aus, die detaillierter dargestellt werden soll. Daraufhin wird deren Name vollständig sichtbar sowie die Felder für Attribute und Methoden. Der Nutzer wählt das Attribut-Feld aus, woraufhin die Attribute in einer eigenen Ansicht dargestellt werden. Da diese Ansicht nicht vollständig auf einen Bildschirm passt, ist hier Scrollen nötig, wie die Scrollbar im rechten Bild verdeutlicht.

Zoom-Darstellungs- und -Steuerungs-Strategien können im Arbeitsablauf beliebig kombiniert werden. So kann beispielsweise ein Nutzer zunächst in einer auf Größe der Ausgabefläche skalierte Fensterrahmen-Darstellung des Desktops ein Visio-Fenster als Wurzel eines hierar-

chischen Zoom auswählen, für welches die untergeordneten Bereiche wie Menüleiste, Werkzeugleisten und Diagramm zunächst als Liste präsentiert werden. Nach Auswahl des Diagramms kann dieses zunächst so kompakt wie möglich, layoutgetreu in einer strukturellen Übersicht dargestellt werden. Bei einem Klassendiagramm wäre dies zum Beispiel mit speziellen grafischen Symbolen für Klassen und Interfaces möglich. Durch geometrischen, semantischen Zoom kann dann von der Symbol-Darstellung zu einer Darstellung mit lesbaren Texten übergegangen werden, wobei die zur Verfügung stehende Ausgabegröße entsprechend erhöht wird.

Bei der Verwendung und besonders bei der Kombination der verschiedenen Zoom-Strategien muss allerdings beachtet werden, dass der Nutzer immer gut darüber informiert ist, welche Zoomvorgänge ausgeführt wurden oder werden können und welche Konsequenzen dies hat. Eventuell sind auch Schulungen nötig, um einen effizienten Einsatz ohne Missverständnisse zu gewährleisten. Bei nicht-geometrischem Zoom muss besonders darauf hingewiesen werden, dass eine Orientierung bezüglich absoluter Positionen nicht möglich ist. Dies ist für die meisten Nutzer in Verbindung mit Zoom eher ungewohnt. Bei hierarchischem Zoom besteht das Problem, dass dem Nutzer klar sein muss, in welcher Ebene er sich befindet und wie er zu einer anderen Ebene zurückkommt. Im HyperBraille-Projekt wurde auch die hierarchische Steuerung mit geometrischer Darstellung eingesetzt. Es zeigte sich schnell, dass die Nutzer öfter das Bedürfnis hatten, aus dem gewählten Objekt auszubrechen. Zwar half es den Nutzer beispielsweise bei Betrachtung eines Dialogfeldes, auf diesen Dialog beschränkt zu sein. So waren sie sicher, nichts zu übersehen. Für die Nutzer war dies aber auch von vornherein ein eigenständiger abgeschlossener Bereich. Bei anderen Darstellungen, wie beispielsweise einer Tabelle innerhalb eines Dokuments empfanden Nutzer die Auswahl der Tabelle meist nicht intuitiv als Einschränkung auf einen bestimmten Bereich, sondern lediglich als Möglichkeit, die Tabelle direkt in den Ausgabebereich einzupassen. Zwar war es auch hier bei der Erkundung der Tabelle nützlich, mit Hilfe von Scroll-Balken, die sich nur auf die Tabelle bezogen, einschätzen zu können, wo in der Tabelle man sich befand. Allerdings wollten die Nutzer nach der Arbeit mit der Tabelle aber auch einfach im Dokument weiterscrollen. Es muss noch untersucht werden, wie beide durchaus verständlichen, aber widersprüchlichen Anforderungen vereint werden können. Sobald sich das Verhältnis zwischen der Größe der Originaldarstellung und der zur Verfügung stehenden Ausgabefläche ändern kann, wie dies beispielsweise im Darstellungskonzept von HyperBraille durch Ein- und Ausblenden verschiedener Darstellungsbereiche möglich ist (siehe Abschnitt 3.2.4.1), stellt sich beim geometrischen Zoom die Fragen, in welcher Weise der Skalierungsfaktor verwaltet wird. Er kann sowohl relativ zur Originalgröße wie auch relativ zur Ausgabegröße gesehen werden. Ein Skalierungsfaktor relativ zur Ausgabegröße hat den Vorteil, dass der Benutzer leicht erkennen kann, wie oft er den dargestellten Ausschnitt verschieben muss, um die gesamte Darstellung erfasst zu haben. Die in die Ausgabefläche eingepasste Darstellung ist immer Zoomstufe 1. Zoomstufe 2 bedeutet, dass die Darstellung 2 Mal die Breite und 2 Mal die Höhe des Ausgabebereichs bedeckt. Ändert sich allerdings die Größe des Ausgabebereichs, so ändert sich auch die Darstellung. Dabei können bei Verkleinerung des Ausgabebereichs De-

tails verschwinden, die vorher sichtbar waren, und umgekehrt. Dies geschieht nicht, wenn der Skalierungsfaktor relativ zur Originalgröße angegeben ist. Allerdings kann hier strenggenommen keine einfache Einpassung der Darstellung in den Ausgabebereich stattfinden, was als Ausgangspunkt einer Exploration immer sehr hilfreich ist, da nicht klar ist, welcher Skalierungsfaktor bei der nächsten Vergrößerung angewendet werden soll. Der Nutzer muss die Vergrößerung über Befehle wie „doppelt so groß“ oder „fünf Mal so groß“ steuern können. Geschieht dies nach einer Einpassung in die Ausgabefläche, werden dem Nutzer je nach Größe der Ausgabefläche unterschiedliche Skalierungsfaktoren präsentiert. Für den Nutzer wäre trotz Skalierung relativ zur Originalgröße nicht klar, wie er eben diese auch darstellen kann. Bei diesem Vorgehen muss also zur Darstellung im Rahmen der einfachen Ausgabefläche eine Verkleinerung entsprechend der zur Verfügung stehenden Skalierungsfaktoren stattfinden solange, bis die Darstellung in die Ausgabefläche passt. Dabei wird eventuell wertvoller Platz verschwendet.

3.2.2.5 Navigation, Selektion und Exploration

Navigation, also die Verschiebung der Gesamtausgabe, so dass verschiedene Teile im Ausgabebereich sichtbar werden, ist eine elementare Interaktionstechnik bei der Arbeit am PC. Bei Blinden und Sehbehinderten kommt neben der Navigation, die von Anwendungen selbst gefordert wird, wie beispielsweise das Scrollen eines Dokuments, noch die Navigation hinzu, die nötig wird, wenn die angepasste Darstellung nicht auf einmal auf die Ausgabefläche passt. Dem Nutzer müssen also zwei Navigationsmodi angeboten werden. Wobei immer dann, wenn die Ausgabe auf ein Objekt beschränkt ist, bei dem Scrollen erforderlich ist, sofern technisch möglich beide Navigationsebenen vereint werden sollten. So muss der Nutzer nur eine Art Interaktion ausführen, um sowohl den für ihn erkennbaren Ausschnitt der Bildschirmgrafik zu verändern, wie auch den am Bildschirm erkennbaren Ausschnitt des Inhalts. Dies erleichtert die Orientierung, da sich der Nutzer nur in einer Ebene zurechtfinden muss.

Da bei detaillierten Darstellungen häufig viel Navigationsarbeit nötig ist, um die gesamte Darstellung zu erfassen, sollten dem Nutzer erweiterte Navigationstechniken und Navigationshilfen zur Verfügung gestellt werden, um beispielsweise gezielt bestimmte Objekte oder Bereiche der Ausgabe anzuspringen. Vor allem sollte es möglich sein, große Leerräume zu überspringen. Liegt ein Objekt-Modell für die Darstellung vor, bedeutet dies das Scrollen zum nächsten Objekt oder zum nächsten nicht leeren Objekt (um beispielsweise leere Tabellenzellen zu überspringen). Ist nur eine Rastergrafik gegeben kann Weiß als Leerraum angesehen werden oder die Farbe, die den Großteil der aktuellen Ansicht ausmacht.

Um gezielt bestimmte Bereiche anzuspringen, kann der Nutzer bei Vorlage einer Rastergrafik beispielsweise zunächst eine Farbe auswählen und danach angeben, in welche Richtung navigiert werden soll. Die Angabe der Richtung kann über eine Kompassnavigation z.B. mit den Pfeiltasten erfolgen, die die Darstellung in vier oder bei Kombination zweier Pfeiltasten auch acht Zielbereiche einteilen (siehe Abbildung 52). Auf Grundlage eines Objektmodells kann eine Navigationsstrategie mit höherer Semantik verwendet werden, die das Navigieren zwi-

schen ähnlichen Objekten, wie etwa Überschriften, und mit Beachtung verschiedener Ebenen unterstützt, in die der Benutzer tiefer hineingehen kann (beispielsweise von der Navigation über Absätze zur Navigation über Worte zur Navigation über einzelne Buchstaben). Auch hier muss natürlich eine Richtung angegeben werden können.

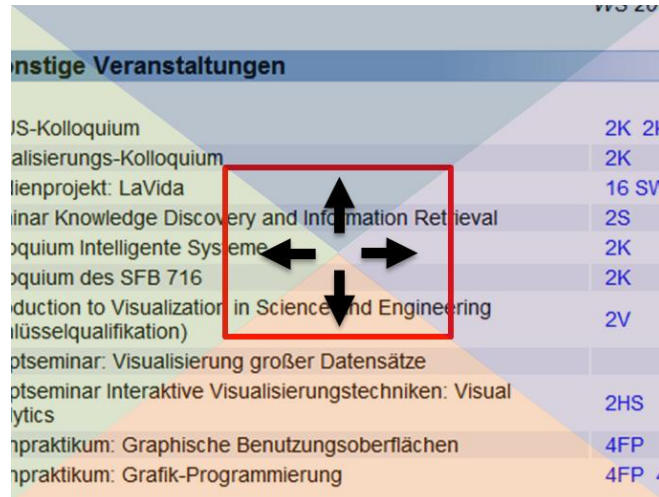


Abbildung 52 Beispiel zur Kompassnavigation in einer vergrößerten Darstellung

Das Bild zeigt einen Ausschnitt einer Webseite zur Übersicht über Lehrveranstaltungen am VIS. Das rote Rechteck markiert den aktuellen Bildschirmausschnitt. Hier hat der Nutzer beispielsweise den Titel der Veranstaltung „Kolloquium Intelligente Systeme“ vollständig gelesen und kann nun mittels des Befehls „nach rechts zum nächsten Objekt“ oder „nach rechts zum nächsten blauen Punkt“ schnell zur Angabe der Semesterwochenstunden der Veranstaltung springen.

Damit der Nutzer nicht Unmengen an Tastenkombinationen erlernen muss, empfiehlt es sich, verschiedene Navigationsmodi zu bieten, zwischen denen der Nutzer einfach z.B. durch mehrfaches Drücken einer einzelnen Taste wechseln kann, so dass zur eigentlichen Navigation nur die Angabe der Richtung nötig ist. Als Alternative zur Richtungsangabe kann auch eine zyklische Navigation ähnlich der Tab-Navigation in GUIs verwendet werden. So kann beispielsweise leicht über alle Worte eines Absatzes oder eben die Elemente einer GUI navigiert werden, ohne dabei deren Lage zueinander kennen zu müssen. Natürlich muss eine sinnvolle Reihenfolge der entsprechenden Objekte ermittelbar sein. Bei Diagrammen bietet sich auch eine Navigation an, die den Verbindungslinien folgt, wie in [Bok08] vorgeschlagen. Beim Wechsel der Navigationsmodi muss dem Nutzer natürlich angegeben werden, in welchen Modus er gewechselt hat. Diese Information sollte auch zu jeder Zeit abfragbar sein.

Sofern eine Ansicht layoutgetreu, aber gefiltert ist, so dass nicht alle Objekte der Originaldarstellung sichtbar sind, muss entschieden werden, ob alle Objekte in die Navigation einfließen sollen oder nur die sichtbaren. Fließen alle Objekte ein, kann es vorkommen, dass die Ansicht an eine Stelle navigiert wird, an der kein Objekt dargestellt ist. Dies kann den Benutzer verwirren. Er könnte denken, dass die Navigation nicht funktionierte. Wird allerdings nur über die sichtbaren Objekte navigiert, könnte der Nutzer für ihn wichtige Objekte auch übersehen und fälschlicherweise annehmen, dass sich an den leeren Stellen auch wirklich keine Objekte befinden. Da das Übersehen von Objekten die schlechtere Alternative ist, sollte der

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Standardmodus immer die Navigation über alle Objekte sein. Das Überspringen der nicht sichtbaren Objekte sollte vom Benutzer bewusst gewählt werden.

Die durch Navigation angesprungenen Objekte (oder auch kleineren Farbbereiche) sollte markiert werden, um vom Benutzer leicht auffindbar zu sein. Dies kann ähnlich wie die Fokusmarkierung in Bildschirmvergrößerungssoftware oder auf dem grafisch-taktilen Display durch Blinken erfolgen, welches aber auch abschaltbar sein muss, um dem Nutzer das ungestörte Erfassen der Ausgabe zu ermöglichen. Letztlich kann die Navigation als eine Art Fokusveränderung gesehen werden, so wie es bei Screenreadern bereits durch die Verwendung verschiedener Cursor üblich ist. Wird eine Navigation mit Ebenenauswahl durchgeführt, sollten entsprechend die gewählten Ebenen markiert werden (natürlich nur so lange, bis die nächste Interaktion durchgeführt wurde).

Auf durch die Navigation angesprungene und damit markierte Objekte sollten genauso wie dies mit einem Fokus möglich ist, Aktionen ausgeführt werden können, die entweder eine weitere Erkundung ermöglichen (z.B. durch Zoom und Filterung), nähere Informationen zum Objekt liefern oder eine Interaktion mit dem dargestellten Inhalt auslösen. Solch eine Selektion kann auch durch eine Exploration unterstützt werden. Lässt sich der Nutzer eine Grafik schrittweise anzeigen, so sollte es ihm möglich sein, leicht Informationen über das zuletzt eingeblendete Objekt zu erhalten oder dieses als Ausgangspunkt für eine weitere Exploration zu setzen. Natürlich sollte der Nutzer auch leicht in die vorige Exploration zurückgehen können, die dann in dem Zustand sein sollte, wie der Nutzer sie verlassen hat. Dieses Vorgehen wurde ansatzweise bereits in [ROE04] mit der Exploration von Gruppen in einer SVG-Grafik präsentiert. Allerdings sollte eine Exploration nicht wie in [ROE04] beschrieben immer von links nach rechts erfolgen. Beispielsweise bei Organigrammen macht dies meist keinen Sinn, wie Abbildung 53 veranschaulicht. Vielmehr sollten ähnliche Strategien zum Einsatz kommen wie bei der Navigation.

Natürlich sollte auch eine direkte Selektion in der Darstellung weiterhin möglich sein. Bei Monitordarstellungen genügt dazu die normale Mauszeiger-Behandlung (mit hervorgehobenem Mauszeiger). An berührungsempfindlichen grafisch-taktilen Displays ist die direkte Selektion nicht trivial. Das „Eingabegerät“, der Finger, ist im Vergleich zur Darstellung sehr groß. Er kann leicht mehrere Objekte überdecken. Beim BrailleDis 9000 kommt hinzu, dass nicht jeder Stift einzeln berührungsempfindlich ist, sondern mit Eingabemodulen gearbeitet wird, die jeweils 10 (2×5) Stifte umfassen. Die Berührung mit einem Finger kann deshalb leicht mehrere Eingabemodule auslösen. Zwar könnte immer einfach der Mittelpunkt oder bei verschiedenen Berührungstärken der Schwerpunkt als Eingabekoordinate gewählt werden. Dies ist aber nicht immer zielführend, vor allem dann, wenn mehrere Objekte dicht beieinander liegen oder sehr kleine Bereiche wie etwa ein einzelner Punkt beispielsweise zum Zeichnen einer Linie gezielt angewählt werden soll. Eine stärker benutzergesteuerte Selektion kann hier bessere Ergebnisse bringen.

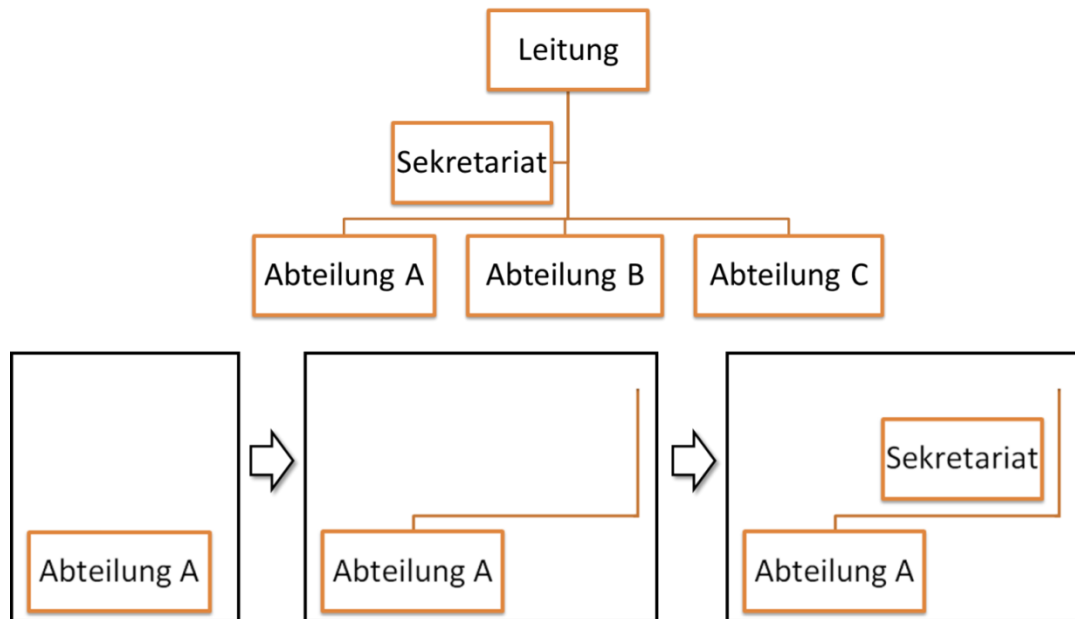


Abbildung 53 Beispiel-Organigramm mit schrittweisem Aufbau von links oben nach rechts unten
Es ist deutlich zu sehen, dass dieser Aufbau zur Exploration des Organigramms nicht sinnvoll ist.

Ausgehend davon, dass der Benutzer die Selektion mit einem Finger durchführt und so eine zweite Hand frei hat, mit der Tasten bedient werden können, kann der durch die Berührung erkannte Selektionsbereich schrittweise in kleinere Bereiche unterteilt werden. Zunächst werden alle Stifte des erkannten Bereichs angehoben. Der Nutzer schränkt diesen Bereich dann mit Hilfe von vier oder fünf Tasten für „oben links“, „oben rechts“, „unten links“, „unten rechts“ und eventuell „Mitte“ auf einen kleineren Bereich ein. Dabei sollten vor allem beim Einsatz von „Mitte“ Überlappungen zwischen den Bereichen existieren. Nach Einschränkung des Bereichs sind nur noch die Stifte des kleineren angehoben. So kann die Einschränkung bis zu einem einzelnen Stift erfolgen. Natürlich sollte der Nutzer die eigentlich gewünschte Aktion jederzeit vorher auch auslösen können. Möchte der Nutzer beispielsweise Informationen zu einem relativ großen Objekt erhalten, das bereits den zu Beginn erkannten, großen Bereich gänzlich einschließt oder zumindest einen Großteil davon abdeckt, so ist keine Einschränkung der Selektion nötig. Immer dann, wenn ein Objektmodell für die Darstellung vorliegt, sollte die Interaktionsverarbeitung bei Selektion mehrerer Stift für eine Aktion das Objekt auswählen, in dem die meisten Selektionspunkte enthalten sind.

3.2.2.6 Hot-Spots und Views

Bei der Arbeit mit Bedienoberflächen sind für den Benutzer häufig immer wieder die gleichen Positionen und Objekte interessant, wie beispielsweise bestimmte Eingabefelder und Links in Webseiten oder Felder mit Statusanzeigen. Nicht alle dieser Punkte sind mit Filter- und Navigationsfunktionen immer leicht zu finden. Deshalb kann es für den Benutzer sehr hilfreich sein, wenn diese Positionen oder Objekte als Navigationsziele, sogenannte Hot-Spots, gespeichert und später leicht wieder angesprochen werden können. So ist es auch einfacher, Informationen zu vergleichen oder zu kombinieren, die nicht gleichzeitig erkennbar

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

in der Ausgabefläche dargestellt werden können. Um leicht wieder aufrufbar zu sein, sollten Hot-Spots vom Benutzer mit Namen versehen werden können. Beispielsweise bei der Untersuchung eines Funktionsgraphen kann ein Benutzer so die gefundenen Nullstellen, Extrema und Wendepunkte markieren. Da Hot-Spots unabhängig von den Darstellungsoptionen, also Layout-, Zoom- und Filter-Einstellungen, sind, können sie auch nach Änderung der Darstellungsoptionen wieder angesprungen werden. So können beispielsweise im Funktionsgraphen die charakteristischen Merkmale zunächst in einer kleinen Zoomstufe markiert werden, um ihre genaue Position dann in einer größeren Zoomstufe zu ermitteln.

Neben der Position ist natürlich auch die Erkennbarkeit der dort befindlichen Information interessant. Diese ist von den angewendeten Darstellungsoptionen abhängig. Häufig kann es für den Benutzer aufwändig sein, gute Darstellungsoptionen zu finden, um eine Arbeitsaufgabe effizient zu lösen. Verschiedene Layout-, Zoom- und Filtereinstellung müssen ausprobiert werden. Hat der Benutzer eine gute Darstellung gefunden, so sollte es ihm möglich sein, diese später leicht wieder anzuwenden. Hier kommt das Konzept von „Views“ zum Einsatz. Diese speichern alle Darstellungsoptionen und sollten wie Hot-Spots benannt werden können, um leicht wieder aufrufbar zu sein. Soweit möglich sollten Zoom- und Scrolleinstellungen in Views relativ zu dem für die View interessanten Objekt, wie etwa einem bestimmten Anwendungsfenster oder einer Grafik in einem Dokument, gespeichert werden. So können sie auch noch sinnvoll angewendet werden, wenn sich die Größe und Position dieses Objektes in der Gesamtdarstellung ändern.

Mit Hilfe von Hot-Spots und Views können Blinde und Sehbehinderte sich selbst an der Verbesserung der angepassten Darstellungen beteiligen. Gute Darstellungsoptionen und wichtige Hot-Spots können anderen Nutzern bereitgestellt werden, ähnlich wie dies heute auch schon mit Anpassungsskripten zu Screenreadern üblich ist.

3.2.3 Umgang mit Farben

Da Normalsichtige Farben häufig als Kommunikationsmittel einsetzen, sollte es Blinden und Sehbehinderten möglich sein, diese auch gut zu erfassen. Zwar können Sehbehinderte Farben an sich meist wahrnehmen, aber häufig wählen sie zur Erleichterung der Wahrnehmung ein von der normalen Ansicht abweichendes Farbschema aus. Zudem existieren natürlich Farbsehschwächen oder vollständige Farbenblindheit, die das Erkennen oder zumindest das Unterscheiden einiger Farben unmöglich machen. Diesem Umstand wird zwar in Gestaltungsrichtlinien durch Forderung nach zusätzlichen Auszeichnungsmitteln als nur Farben Rechnung getragen (siehe z.B. [Webe]). Allerdings lösen solche Richtlinien, auch wenn sie von allen eingehalten werden, das Problem nicht gänzlich. Zum einen ändert sich dadurch der Sprachgebrauch Normalsichtiger nicht, zum anderen ist es insbesondere bei Grafiken meist schlecht möglich, bessere Auszeichnungsmittel als Farben zu finden. Zwar könnten teilweise zusätzlich verschiedene Muster angewendet werden. Aber diese sind häufig für Sehbehinderte auch nicht erkennbar und oft auch nur schwierig zu beschreiben. Für Blinde können sie sogar einen Nachteil bringen, da sich eine einfache Farbinformation durch ein Computerpro-

gramm wesentlich leichter analysieren lässt als ein Muster. Letztlich sollte man Blinden und Sehbehinderten nicht das Recht oder den Wunsch absprechen, über die Farbigkeit einer Darstellung informiert zu sein.

Im Umgang mit Farben stellen sich also zwei wesentliche Herausforderungen. Zum einen die Erzeugung erkennbarer Darstellungen, in denen im Idealfall die durch die farbige Gestaltung vermittelten Informationen erhalten bleiben, und zum anderen die Schaffung von Möglichkeiten, um die ursprünglich für die Darstellung genutzten Farben zu ermitteln und zu erkennen, welche Bereiche oder Objekte der Ausgabe mit diesen Farben gestaltet wurden. Nur so kann eine Kommunikation über die Farben stattfinden und beispielsweise Erläuterungen zu einer Grafik, die sich auch auf deren farbliche Gestaltung bezieht, verstanden werden. Die Abschnitte 3.2.3.3 und 3.2.3.4 präsentieren einige Konzepte dazu.

Damit Kommunikation über Farben und die Verknüpfung von Erläuterungen mit farbigen Darstellungen sinnvoll stattfinden kann, müssen alltägliche Benennungen, wie Rot, Grün und Blau, verwendet werden. Die in Computerprogrammen üblichen RGB-Angaben sind zur Kommunikation nicht sinnvoll. Weder werden Normalsichtige diese in ihren Erklärungen verwenden noch würden sie verstehen, welche Farbe gemeint ist, wenn ein Blinder oder Sehbehinderter diese als RGB-Wert ausdrückt. Hinzu kommt, dass auch Blinde oder Sehbehinderte selbst sich nur schwer einen Eindruck von einem Bild machen können, wenn dessen Farben nur als RGB-Werte repräsentiert sind. Sprechende Bezeichnungen sind hier hilfreicher, zumindest wenn eine mentale Vorstellung der Farbe existiert, was bei Menschen, deren Sehbehinderung nicht von Geburt an bestand, meist der Fall ist.

Es muss also eine Zuordnung zwischen von Menschen im natürlichen Sprachgebrauch genutzten Farbnamen und in Computer-verarbeitbaren Formaten ausgedrückten Farben stattfinden. Diese kann letztlich nicht nur zur Kommunikation und zum leichteren Verständnis einer farbigen Gestaltung dienen, sondern auch der Umgestaltung in eine erkennbare Darstellung. Beispielsweise kann so ein Benutzer mit Rot-Grün-Sehschwäche eine Ersatzdarstellung für rote und grüne Farben bestimmen, ohne dabei alle in Frage kommenden RGB-Werte einzeln auflisten zu müssen. Die Abschnitte 3.2.3.1 und 3.2.3.2 zeigen, wie eine solche Zuordnung geschehen kann.

3.2.3.1 Zuordnung von Farben zu Farbnamen

Im Allgemeinen genügen zur Kommunikation über Farben wenige Farbnamen, beispielsweise die von Berlin und Kay [BK99] identifizierten „Basic Color Terms“ Weiß, Schwarz, Rot, Grün, Gelb, Blau, Braun, Violett, Pink, Orange und Grau. Genauere immer noch intuitiv verständliche Definition können durch die Ergänzung von Attributen wie „hell“, „dunkel“, „gesättigt“ und „blass“ erfolgen oder durch die Nutzung eines größeren Farbnamen-Raums. Im englischsprachigen Raum bieten sich hier beispielsweise die Bezeichnungen der Web-Farben an [Wik10b]. Bei Verwendung eines größeren Farbnamen-Raums sollte allerdings eine Reduktion auf die genannten Grundfarben möglich sein, da beispielsweise Farbbezeichnungen wie die Web-Farbe „Moccasin“, welche ein sehr helles Braun bezeichnet und sogar zu den

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Gelbtönen gezählt wird, oder „eierschalenfarben“, welche sowohl für Weiß wie auch für ein helles Braun stehen könnte, missverständlich sein. Es ist daher sinnvoll, dem Benutzer eine mehrstufige Farbbezeichnung anzubieten, die ihm die Möglichkeit bietet, sowohl eine grobe Zuordnung zu wenigen Basisfarben zu erfragen, um eine schnelle Kommunikation zu ermöglichen, als auch feinere Zuordnungen und letztlich auch den konkreten Farbwert in Computerdarstellung, falls eine detaillierte Bildanalyse gewünscht ist.

Grundsätzlich bestehen für die Zuordnung von Farben zu Farbnamen bzw. zu einer Auswahl an Grundfarben zwei Möglichkeiten – zum einen die absolute Einordnung in einen Farbraum und zum anderen die relative (vergleichende) Zuordnung zu einer Auswahl von Farben.

Bei der absoluten Einordnung wird ein gegebener Farbraum anhand verschiedener Kriterien in Farbsegmente aufgeteilt. Die Segmente werden benannt. Die zur Aufteilung verwendeten Kriterien werden schließlich zur Einordnung von beliebigen Farben in die Segmente genutzt. Eine solche relative Einordnung basierend auf dem Farbraum HSV (siehe Abbildung 54 links) wird in [Con92] vorgestellt. Ein Ausschnitt dazu ist zwecks Illustration der Methode in Listing 6 gegeben. Die im Modell vorgeschlagene Farb-Segmentierung ist allerdings nicht zu empfehlen. Wie Listing 6 zeigt wird darin beispielsweise der Wert $(0^\circ, 100\%, 100\%)_{\text{HSV}}$ (bzw. im Modell $(0, 1, 1)_{\text{HSV}}$) der Farbe Orange zugeordnet, obwohl dieser reines Rot beschreibt.

Bei der relativen Zuordnung zu einer Auswahl von Farben wird eine Menge von Farben definiert – die Basis- oder Falschfarben. Mit Hilfe einer Metrik werden die Abstände zwischen allen Basisfarben und der zuzuordnenden Farbe berechnet. Zur Benennung der zuzuordnenden Farbe wird die Basisfarbe mit dem geringsten Abstand gewählt.

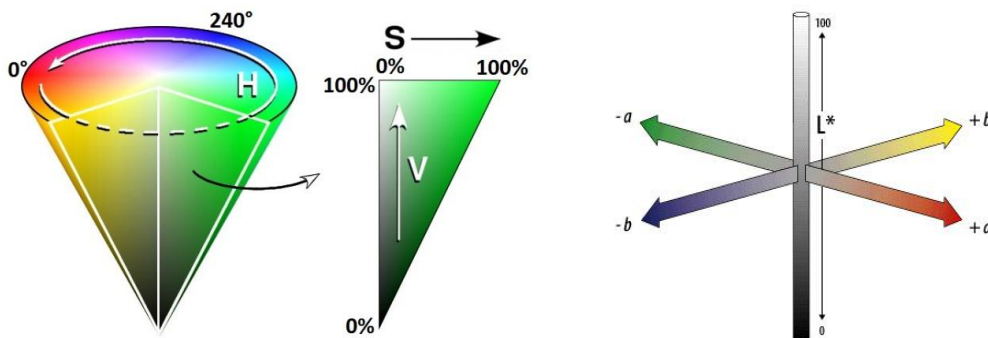


Abbildung 54 Schematische Darstellungen der Farbräume HSV (links) und $L^*a^*b^*$ (rechts)
HSV: H = Hue = Farbwert ($0^\circ - 360^\circ$), S = Sättigung ($0\% - 100\%$), V = Value = Helligkeit ($0\% - 100\%$)
 $L^*a^*b^*$: L = Luminance = Helligkeit, a^* = Rot-Grün-Achse, b^* = Gelb-Blau-Achse

Colour	← Luminance Tone	iff $0.0 \leq L < 0.1$	or $0.0 \leq S < 0.1$
	← Luminance Saturation Hue	otherwise	
Tone	← black	iff $0.0 \leq L < 0.1$	
	...		
Luminance	← dark	iff $0.1 \leq L < 0.3$	
	...		
Saturation	...		
Hue	← brown	iff $0.0 \leq H < 0.1$	and $0.0 \leq L < 0.5$
	← orange	iff $0.0 \leq H < 0.1$	and $0.5 \leq L \leq 1.0$
	...		
	← magenta	iff $0.8 \leq H < 0.9$	
	← red	iff $0.9 \leq H \leq 1.0$	

Listing 6 Ausschnitt aus einem Modell zur absoluten Farbzuoordnung aus [Con92]
 Das Modell soll lediglich zur Illustration der Methode der absoluten Farbeinordnung dienen. Die konkrete Zuordnung selbst ist nicht zu empfehlen, weshalb hier nur ein Ausschnitt gegeben ist.

Sowohl absolute als auch relative Zuordnung haben Vor- und Nachteile. Die absolute Zuordnung kann über einen Regelsatz effizienter implementiert werden als die relative. Außerdem kann sie das komplexe Farbempfinden des Menschen potenziell besser abbilden als jede Abstandsberechnung, denn eine Farbzuoordnung entspricht nicht unbedingt einer Abstandsermittlung. Eigentlich sind dies zwei verschiedene Fragestellungen, wie folgendes Beispiel zeigen soll. Tabelle 2 zeigt drei Farben. Links ein reines Blau, rechts ein reines Weiß. Beide Farbtöne werden meist auch als Basisfarbe für die entsprechenden Basic Color Terms verwendet. Die Farbe in der Mitte ist zuzuordnen. Stellt man nun Betrachtern die Aufgabe, die mittlere Farbe spontan zu benennen, so erhält man Antworten, die einen Blauton beschreiben. Fragt man allerdings, ob diese Farbe näher an dem links gezeigten Blau oder dem rechts gezeigten Weiß liegt, fällt die Entscheidung meist auf Weiß.

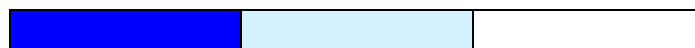


Tabelle 2 Farbbeispiel zum Unterschied zwischen Farbzuoordnung und Abstandsbetrachtung

Die relative Zuordnung hat den Vorteil, dass die Basisfarben leicht verändert bzw. ausgetauscht werden. Sind beispielsweise in einem Diagramm verschiedene Kategorien mit Hilfe verschiedener Grauwerte dargestellt, so nützt es dem Benutzer nichts, eine Zuordnung zu den oben genannten Basic Color Terms durchzuführen. Sinnvoller ist eine Zuordnung zu den Farben der Diagramm-Legende. Aufgrund der größeren Flexibilität der relativen Zuordnung bezieht sich die Arbeit im Folgenden nur noch auf diese Methode.

Da verschiedene Menschen Farben unterschiedlich empfinden und Farben je nach Einstellung des Ausgabegerätes (Monitor, Drucker etc.) unterschiedlich erscheinen können, ist nicht unbedingt eine eindeutige Zuordnung von Farben zu Farbnamen sinnvoll. Sofern es die gewählte Darstellungsform zulässt, sollte eine mehrfache Zuordnung in Betracht gezogen werden, wie beispielsweise Rotard das bereits in [Rot05] vorschlug.

3.2.3.2 Implementierungsaspekte zur relativen Farbzuoordnung

Bei Algorithmen zur relativen Farbzuoordnung sind zwei Aspekte wesentlich:

1. Die Korrespondenz zwischen Zuordnungsergebnis und dem menschlichen Empfinden: Entspricht das Zuordnungsergebnis nicht dem menschlichen Empfinden kann es zu Missverständnissen kommen, die eventuell eine größere Kommunikationsbarriere schaffen, als wenn es dem Blinden oder Sehbehinderten gar nicht möglich ist, auf Ebene der Farben zu kommunizieren.
2. Die Zeitspanne, die zur Ermittlung der Zuordnung nötig ist (Performanz): Muss der Benutzer mehrere Sekunden oder gar Minuten auf das Ergebnis einer Zuordnung warten, so kann er damit nicht effizient arbeiten. Vor allem wenn Rastergrafiken zum Zwecke der Farbfilterung segmentiert werden, ist die Performanz des Zuordnungsalgorithmus entscheidend, da hier jeder Bildpunkt der Grafik einzeln analysiert werden muss. Bei Formaten, in denen die farbliche Gestaltung über Objekteigenschaften definiert ist, wie beispielsweise Vektorgrafiken oder Dokumente mit Stilvorlagen, ist die Performanz meist weniger kritisch, da nur wenige Vergleiche durchgeführt werden müssen.

Beide Aspekte werden sowohl durch die zur Abstandsberechnung genutzte Metrik, wie auch die verwendeten Basisfarben beeinflusst. Eine Metrik, die komplexe Berechnungen erfordert, verlangsamt den Algorithmus, kann aber bessere Ergebnisse hervorbringen. Da die zuzuordnende Farbe mit allen Basisfarben verglichen werden muss, um den geringsten Abstand zu ermitteln, ist die Performanz eines einfachen Zuordnungs-Algorithmus linear von der Anzahl der Basisfarben abhängig. Für eine schnelle Zuordnung sollten also nicht zu viele Basisfarben verwendet werden. Zu wenige Basisfarben führen allerdings zu einer unzureichenden Farbzuoordnung. Werden beispielsweise die Basisfarben Rot, Grün, Gelb und Blau auf ein Schwarz-Weiß-Bild angewendet, so kann es nur zu missverständlichen Zuordnungen kommen. In solchen Fällen, also bei zu großem Abstand zwischen Basisfarben und zuzuordnender Farbe sollte besser keine Zuordnung stattfinden. Dies muss dem Benutzer allerdings auch deutlich gemacht werden, um Missverständnisse zu vermeiden.

Häufig wird zur Abstandsberechnung eine einfache euklidische Metrik genutzt. Diese kann allerdings nur korrekte Zuordnungen bringen, wenn sie auf einem wahrnehmungs-uniformen Farbraum angewendet wird, also einem Farbraum, in dem gleiche Änderungen in Farbkomponenten gleiche Auswirkungen auf das Farbempfinden haben. Wird eine euklidische Metrik auf einem andersartigen Farbraum angewendet, wie beispielsweise dem lediglich wahrnehmungs-konformen Farbraum HSV, kann es zu deutlichen Fehlzuoordnungen kommen. Auch der Farbraum Lab ist nicht vollständig wahrnehmungs-uniform, weshalb auch hier Fehlzuoordnungen bei der Anwendung der euklidischen Metrik entstehen. Bisher wurde allerdings kein besserer Farbraum entwickelt. Es wurden aber bessere Metriken zur Abstandsberechnung im Lab-Raum präsentiert. Die derzeit aktuelle Metrik ist die CIEDE2000 [SWD05]. Diese erzeugt deutlich bessere Zuordnungen als die euklidische Metrik und auch als ihr Vorgänger die CIE94 [Lin03] [URBS07], welche von Kritikern der CIEDE2000 noch emp-

fohlen wird. Zur Untersuchung und Illustration der Unterschiede in den Zuordnungsergebnissen wurden die vier genannten Metriken auf eine Rastergrafik angewendet, in der alle im RGB-Raum darstellbaren Farben vorkommen. Jeder Pixel der Rastergrafik besitzt einen anderen RGB-Wert (siehe Abbildung 55 links). Diese Grafik wurde mit Hilfe der Metriken segmentiert, wobei einige Unterschiede feststellbar sind, wie die Vergleichsgrafiken in Abbildung 55 demonstrieren. Als Basisfarben wurden in diesem Vergleich die Basic Color Terms verwendet, welche mit den in Tabelle 3 gegebenen RGB-Werten verknüpft wurden.

Name	R	G	B	Farbe
Weiß	255	255	255	
Schwarz	0	0	0	
Rot	255	0	0	
Grün	0	191	0	
Gelb	255	255	0	
Blau	0	0	255	

Name	R	G	B	Farbe
Braun	140	69	19	
Violett	169	0	255	
Pink	255	0	255	
Orange	255	166	0	
Grau	128	128	128	

Tabelle 3 Segmentierungsfarben zum Test verschiedener Metriken

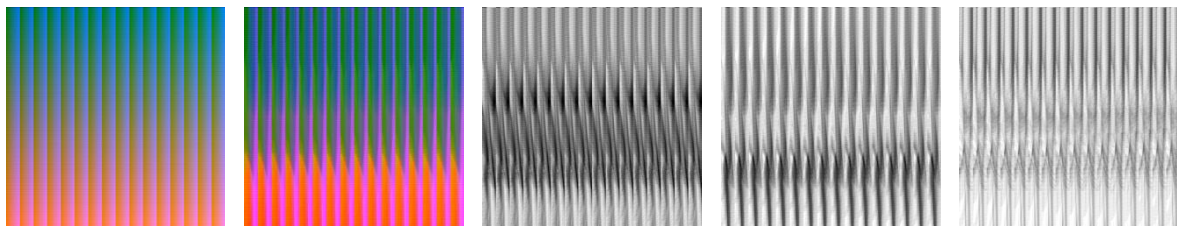


Abbildung 55 Grafiken zum Metriktest (Originalauflösung: 4096 × 4096 Pixel)

1. Testgrafik mit allen möglichen Farbwerten im RGB-Raum ($2^{24} = 16.777.216$ Farben)
2. Ergebnis der Segmentierung mit der Metrik CIEDE2000 den Basisfarben aus Tabelle 3
- 3.-4. Grafiken zum Vergleich der drei anderen Metriken mit CIEDE2000 (Stellen, an denen sich die resultierenden Basisfarben unterscheiden wurden Schwarz markiert)
3. Unterschiede zwischen euklidischer Metrik über HSV und CIEDE2000 (7.698.111 Unterschiede)
4. Unterschiede zwischen euklidischer Metrik über Lab und CIEDE2000 (5.977.755 Unterschiede)
5. Unterschiede zwischen CIE94 und CIEDE2000 (4.151.161 Unterschiede)

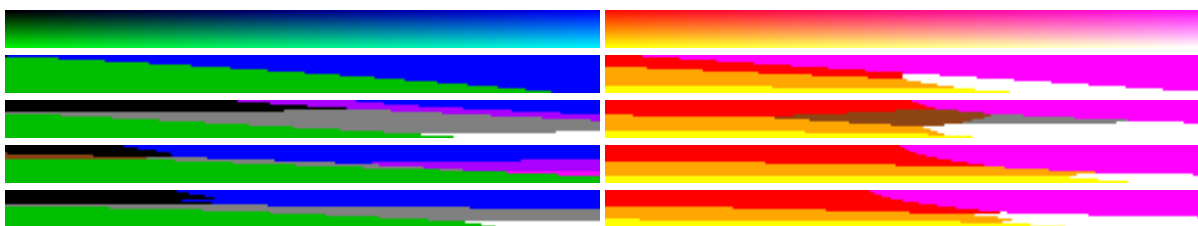


Abbildung 56 Ausschnitte aus den Segmentierungsergebnissen

- links: Ausschnitt von links oben (0,0) – (255, 15), rechts: rechts unten (3841, 4081) – (4096, 4096)
 oben: Ausschnitt aus Originalgrafik
 danach: Ausschnitte aus den Segmentierungsergebnissen mit den verschiedenen Metriken
 v. o. n. u.: euklidische Metrik über HSV, euklidische Metrik über Lab, CIE94, CIEDE2000

Die Ausschnitte aus den Segmentierungsergebnissen in Abbildung 56 zeigen deutlich, dass die Metrik CIEDE2000 zu bevorzugen ist. Die euklidischen Metriken zeigen in beiden Aus-

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

schnitten deutliche Fehlzuordnungen. So bildet die HSV-Metrik im linken Ausschnitt keine Farbe auf schwarz ab außer das reine Schwarz $(0, 0, 0)_{\text{RGB}}$ selbst. Im rechten Ausschnitt werden sowohl von der HSV-Metrik wie auch der euklidischen Metrik über Lab zu viele Farben auf Weiß abgebildet. Mit der CIE94 ergeben sich bereits bessere Zuordnungen. Im linken Ausschnitt werden hier allerdings Violett und Pink zugeordnet, obwohl dieser Abschnitt nur aus RGB-Werten aufgebaut ist, die einen Rot-Wert von 0 besitzen. Mit der CIEDE2000 ergibt sich diese Fehlzuordnung nicht.

Unter dem Aspekt der Performanz schneidet die CIEDE2000-Metrik aufgrund ihrer komplexeren Berechnungsvorschrift leider deutlich schlechter ab als die drei Vergleichsmetriken. In einem Vergleichstest auf einem aktuellen Arbeitsrechner wurden für die Segmentierung der oben gezeigten Testgrafik, in der jeder Pixel einen anderen RGB-Wert aufweist, mit der CIEDE2000-Metrik über zwei Minuten benötigt, während die euklidischen Metriken im Bereich von Sekunden arbeiteten. Die Segmentierung wurde in C# implementiert, wobei bereits eine der Sprache entsprechende effiziente Implementierung der CIEDE2000-Metrik verwendet wurde. Beispielsweise wurden wo möglich vorberechnete Werte verwendet und zur Quadrierung einfache Multiplikationen eingesetzt statt der vom .Net-Framework bereitgestellten Funktion `Math.Pow()`. Tabelle 4 zeigt die Messergebnisse. Der Unterschied zwischen Anwendung der Euklidischen Metrik über dem HSV-Raum und über dem Lab-Raum ergibt sich durch die aufwändigere Umwandlung von RGB nach Lab im Gegensatz zu RGB nach HSV.

Metrik	Gesamtzeit	Zeit pro Pixel
HSV euklidisch	00:00:03,14	$1,87 \cdot 10^{-7}$ s
Lab euklidisch	00:00:18,24	$10,87 \cdot 10^{-7}$ s
Lab CIE94	00:01:22,69	$49,29 \cdot 10^{-7}$ s
Lab CIEDE2000	00:02:40,60	$95,73 \cdot 10^{-7}$ s

Tabelle 4 Zeitspannen, die zur Segmentierung der Testgrafik (siehe Abbildung 55) benötigt wurden

Allerdings kann die Testgrafik nicht als repräsentativ für Grafiken gesehen werden, mit denen Computernutzer täglich konfrontiert werden. Sie ist deutlich größer als die meisten Bilder und weist vor allem deutlich mehr Farben auf. Enthält ein Bild weniger Farben als Pixel, so kann die Segmentierung mit Hilfe einer LookUp-Tabelle beschleunigt werden. Hierzu wurde ein Vergleichstest mit einer Menge realistischer Bilder durchgeführt, die sich aus den Bildern, die für diese Arbeit erstellt wurden, und den Bildern aus einem E-Learning-Kurs zur Visualisierung zusammensetzt. Insgesamt wurden 784 Bilder für diesen Test segmentiert. Diese umfassten unter anderem kleine Schwarz-Weiß-Darstellungen, Diagramme, Screenshots von Desktopanwendungen und medizinischen Visualisierungen, wie auch Darstellungen von Wetterdaten und Fotografien von Kunstwerken und Geräten. Abbildung 57 zeigt einige Beispiel-Bilder aus dem E-Learning-Kurs. Insgesamt enthielten die Bilder lediglich etwa 3 Millionen verschiedene Farben, also deutlich weniger als die möglichen 16 Millionen Far-

3.2 Konzepte zu Darstellungen und Interaktionen

ben. Der Durchschnitt verschiedener Farben pro Bild lag gerade einmal bei knapp 9000 Farben. Tabelle 5 zeigt die Kenndaten der Testbilder.

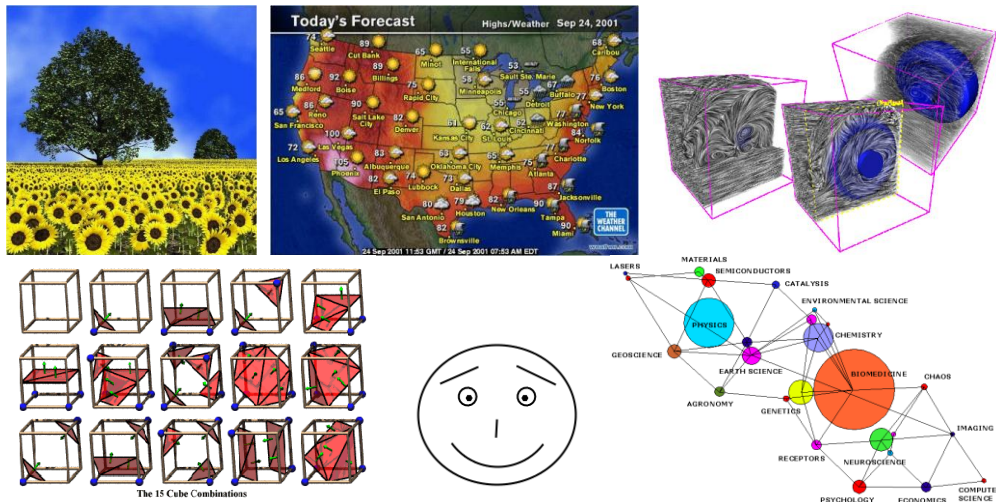


Abbildung 57 Beispielgrafiken aus dem E-Learning-Kurs (zum Test der CIEDE2000-Performanz)

Anzahl Bilder		784
Höhe in Pixeln	Min	3
	Max	3.600
Breite in Pixeln	Min	11
	Max	7.200
Anzahl Pixel	Min	33
	Max	25.920.000
	Gesamt	478.192.355
	Durchschnitt	609.939
Anzahl Farben	Min	2
	Max	302.030
	Gesamt (inkl. Überschneidungen zwischen Bildern)	6.773.237
	Durchschnitt	8.639
	Gesamt (von Überschneidungen bereinigt)	3.214.143

Tabelle 5 Kenndaten des Testdatensatzes für den CIEDE2000-Performanz-Test

	Ohne LookUp-Tabelle	Mit LookUp-Tabelle (pro Bild)	Mit LookUp-Tabelle (über alle Bilder)
Gesamtzeit	01:03:33,437	00:01:22,537	00:00:48,462
Durchschnitt pro Bild	00:00:04,864	00:00:00,105	00:00:00,062
Durchschnitt pro Pixel	$79,74 \cdot 10^{-7}$ s	$1,73 \cdot 10^{-7}$ s	$1,01 \cdot 10^{-7}$ s

Tabelle 6 Messergebnisse zum CIEDE2000-Performanz-Test

Wie die Messergebnisse in Tabelle 6 zeigen, kann mit Hilfe einer LookUp-Tabelle tatsächlich eine deutliche Beschleunigung erreicht werden, durch die die Segmentierung mit Hilfe der CIEDE2000-Metrik zur effizienten Arbeit einsetzbar ist. Durch Anwendung einer LookUp-Tabelle pro Bild reduzierte sich die durchschnittliche Segmentierungszeit pro Bild von über 4

Sekunden auf etwa 0,1 Sekunden. Durch Anwendung einer globalen LookUp-Tabelle konnte eine weitere Reduzierung auf 0,06 Sekunden erreicht werden. Die LookUp-Tabellen wurden dabei nicht im Vorfeld der Segmentierung aufgebaut, sondern währenddessen.

3.2.3.3 Darstellung von Farben

Durch Zuordnung von Farbnamen ist es leicht möglich, Benutzern bei Auswahl einer Position oder eines Objektes in der Ausgabe, verständliche Farbnamen und eventuell weitere Informationen zu den verwendeten Farben zu präsentieren. Am Bildschirm kann dazu ein einfaches kleines Anwendungsfenster genutzt werden, das sogenannten Color-Picker-Anwendungen ähnelt. Auf grafisch-taktilen Displays eignet sich am besten eine Audio-Ausgabe. Aber auch eine Ausgabe in Braille in einem separaten Bereich der Ausgabe ist möglich. Natürlich muss der Benutzer dem System zuvor mitteilen können, dass er sich über Farben in der Ausgabe informieren möchte. Mit diesen Methoden der Exploration kann sich ein Benutzer über die Farben innerhalb der Ausgabe informieren und Normalsichtige durch Nennung einer Farbe auf einen Bereich in der Ausgabe hinweisen. Allerdings ist es so natürlich schwierig, einen Überblick über die Farbverteilung in der Ausgabe zu erhalten. Dazu müssen Darstellungen erzeugt werden, in denen die Farbinformationen möglichst direkt erkennbar sind. Je nach Arbeitsaufgabe bzw. Zielstellung und Ausgangsdarstellung können verschiedene Anpassungen sinnvoll sein. Um beispielsweise die schwarzen Konturen in Comic-Darstellungen gut erkennen zu können oder bestimmte Farbbereiche wie beispielsweise alle gelben Spielsteine in einem SameGame¹⁰ zu finden, eignen sich sehr gut Filterungen nach Farben, bei denen jeweils nur die gewählte Farbe in layoutgetreuer Darstellung erkennbar angezeigt wird (siehe dazu auch Abschnitt 3.2.2.2). Abbildung 58 zeigt zwei Beispiele dazu.

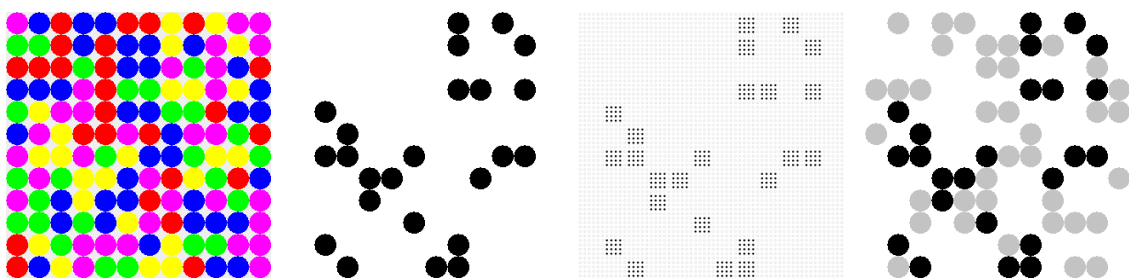


Abbildung 58 Beispiel zu selektiver Farbdarstellung durch Filterung (Originalgröße: 312 × 312 Pixel) v.l.n.r.: Screenshot des SameGame jBubbleBreaker (<http://jbubblebreaker.sourceforge.net/>) Filterung nach Gelb für Monitorausgabe, Filterung für Ausgabe auf grafisch-taktilen Display skaliert auf 60 × 60 Pixel, Filterung nach Blau und Gelb für Monitorausgabe mit Graustufen

Derart selektive Farbdarstellungen sind aber nur schlecht geeignet, den Zusammenhang zwischen Objekten verschiedener Farben zu erkennen. Um beispielsweise auf dem in Abbildung 58 abgebildeten Spielfeld herauszufinden, welche gelben Bereiche durch Entfernen anderer Steine zusammengeführt werden können und welche Auswirkungen dies auf anderer Bereiche hat, muss der Nutzer mehrere Filterungen durchführen und deren Ergebnisse in

¹⁰ Siehe <http://de.wikipedia.org/wiki/SameGame>

einem mentales Abbild des Spielfelds zusammenführen. Dies bedeutet einen hohen kognitiven Aufwand. Deshalb sollten dem Benutzer Möglichkeiten gegeben werden, mehrere (möglichst alle) Farben gleichzeitig und dabei unterscheidbar darzustellen. Bei Monitor Darstellungen wird dazu neben einfachen Farbersetzungen, die in der Kommunikation mit anderen natürlich zu Missverständnissen führen könnten, häufig mit verschiedenen Graustufen gearbeitet. Bei sehr wenigen Graustufen bzw. Farben (wie in Abbildung 58 rechts) kann dies sinnvoll sein. Mit steigender Farbanzahl sinkt die Unterscheidbarkeit aber deutlich, wie Abbildung 59 links mit nur fünf Farben deutlich zeigt. Natürlich darf die Zuweisung der Graustufen nicht rein nach Helligkeitswert erfolgen, sondern muss entsprechend der Farbunterschiede angewendet werden. Sonst sind gleich helle Farben nicht unterscheidbar. Kontrastreichere und damit leichter erfassbare Darstellungen können durch Anwendung verschieden starker Konturen, wie sie auch im Bereich der taktilen Grafiken Gang und Gäbe sind, erzeugt werden (Abbildung 59 zweite von links). Konturdarstellungen haben allerdings den Nachteil, dass die Ausdehnung der Objekte nicht immer deutlich hervorkommt. So ist im Beispiel nicht unbedingt klar, ob es sich um ausgefüllte Kreise handelt oder tatsächlich nur um leere Kreise mit dicken Rändern. Im Beispiel wird aber auch deutlich, dass dies nicht immer von Belang ist. Um das Spiel spielen zu können, muss lediglich entschieden werden können, ob Objekte gleich oder verschieden sind. Ihre exakte Gestaltung ist für den Nutzer nicht direkt relevant. Konturdarstellungen können auch zusätzlich zur Originaldarstellung (als Overlay) eingesetzt werden (Abbildung 59 zweite von rechts). So bleibt die Information über die Ausdehnung der Objekte besser erhalten und auch die Farbinformation ist sofern für den Nutzer erkennbar noch direkt zugänglich. Die Kontraste werden dadurch aber gemindert. Um die Kommunikation mit anderen zu fördern, sollte dem Nutzer bei solchen Ersatzdarstellungen immer eine Legende angeboten werden, um den Zusammenhang zwischen Originalfarben und Ersatzdarstellung schnell herstellen zu können.

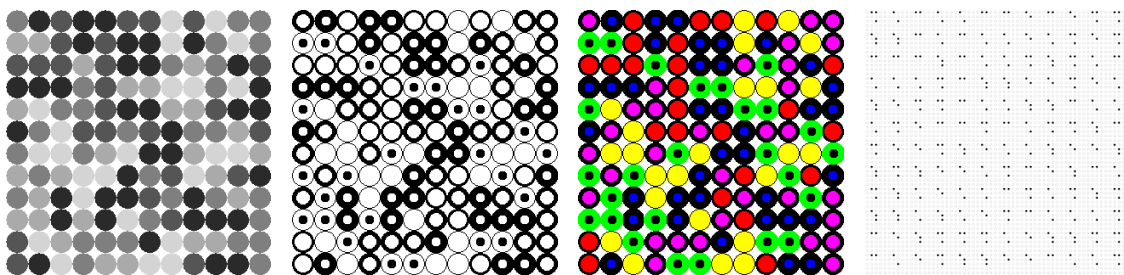


Abbildung 59 Beispiele zu gleichzeitiger, unterscheidbarer Darstellung von Farben
v.l.n.r.: Spielfeld aus Abbildung 58 links in Graustufen (schlecht unterscheidbar), Anwendung verschieden starker Konturen, Konturen und Farben kombiniert (Probleme mit Kontrast möglich), taktile Darstellung mit Farbcodes¹¹ (ein Zeichen pro Spielstein, skaliert auf 60 × 60 Pixel)

Für die Ausgabe auf grafisch-taktilen Displays sind Darstellungen mit verschieden starken Konturen nur schlecht geeignet. Zum einen entstehen dadurch eher große Darstellungen. Um beispielsweise die Konturen aus Abbildung 59 auf ein grafisch-taktilen Display zu übertragen, muss die Größe der Darstellung, also 312 × 312 Pixel, erhalten bleiben. Zum anderen

¹¹ Zu Farbcodes siehe Abschnitt 3.3.1.2.2

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

muss die Darstellung detailliert ertastet werden, um die jeweiligen Konturstärken zu erkennen. Auf grafisch-taktilen Displays kann aber ausgenutzt werden, dass aufgrund der Größe der Stifte und ihrer Abstände schon Kombinationen aus wenigen Pixeln gut unterschieden werden können. Braille-Zeichen bestehen nur aus 2×3 Pixeln und können von Blinden schnell erfasst werden. Damit lässt sich eine sehr kompakte Farbrepräsentation in Form eines Farbcodes umsetzen, wie Abbildung 59 rechts zeigt, wobei eine eindeutige Zuordnung zwischen Farben und Farbcode-Zeichen erfolgen kann. Auch hier ist zwar die Form der Spielsteine nicht erkennbar, aber dies ist wie schon erwähnt auch nicht unbedingt relevant. Der hier verwendete Farbcode wird in Abschnitt 3.3.1 erläutert.

In größeren Flächen können sowohl bei Monitordarstellungen wie auch bei grafisch-taktilen Darstellungen auch Texturen zum Einsatz kommen. Abbildung 60 zeigt Beispiele dazu. Texturen helfen dabei, die Ausdehnung von Flächen leichter zu erkennen. Da bei grafisch-taktilen Displays im Gegensatz zu taktilen Grafiken in Schwellpapier oder Ähnlichem kein Höhenunterschied zwischen Texturen und Konturen möglich ist, sollte allerdings darauf geachtet werden, dass Konturen von genügend nicht texturiertem Bereich umgeben sind, um ertastbar zu sein. Im Balkendiagramm in Abbildung 60 links konnten die Konturen von Probanden leicht erfasst werden. Im Kreisdiagramm daneben hatten Probanden dagegen Schwierigkeiten, obwohl hier schon jeweils eine Stiftreihe Abstand zwischen Textur und Kontur gelassen wurde. Um eine direkte Zuordnung zwischen Textur und Farbe zu erzielen, kann auch ein Farbcode zur Texturierung eingesetzt werden, wie Abbildung 60 dritte von links zeigt. Allerdings bewerteten auch hier Probanden die zum Vergleich gebotene Darstellung mit nur einem Farbcode pro Fläche als besser erfassbar (in Abbildung 60 daneben). Höchstwahrscheinlich ist auch dies auf die Abgrenzung zu den Konturen zurückzuführen. Dabei fiel es den Probanden nicht schwer zu erkennen, dass die wenigen Punkte innerhalb der Flächen einen Farbcode darstellen, auch wenn zuvor nicht gesagt wurde, dass sich Farbcodes in der Darstellung befinden. Kombiniert mit einer wechselweisen Füllung von Flächen, wie in Abbildung 60 unten links, kann auch bei Anzeige einzelner Farbcodes der flächige Zusammenhang gut verdeutlicht werden. In komplexeren Bildern kann dieser Vorteil allerdings nicht immer voll ausgenutzt werden, wie Abbildung 60 unten Mitte und rechts zeigt. Nicht immer grenzen alle Flächen so aneinander, dass eine abwechselnde Füllung gut möglich ist. Zudem kann es für den Nutzer verwirrend sein, wenn Flächen gleicher Farbe einmal gefüllt und einmal nicht gefüllt dargestellt werden.

Für Liniendarstellung wie in Abbildung 61 links sind Texturen nur schlecht einsetzbar. Sie können die Darstellung wie starke Konturen auch deutlich verfälschen. Deshalb wurde im Rahmen dieser Arbeit untersucht, inwieweit auf grafisch-taktilen Displays Blinken, also schnelles Heben und Senken von Stiften, eingesetzt werden kann, um Farben oder auch Objekte gleichzeitig aber unterscheidbar darzustellen. Moderne Displays bieten aufgrund ihrer kurzen Bildaufbauzeiten gute Möglichkeiten auch großflächige blinkende Darstellungen zu erzeugen. Es zeigte sich, dass auch blinkende Darstellungen von Nutzern gut angenommen werden. Es konnte schnell erfasst werden, wo sich unterschiedliche Farbbereiche befanden. Unterschiedliche Bereiche mit gleicher Blinkfrequenz wurden erkannt. So kann die blinkende

Darstellung auch gut zur Zuordnung von Diagrammen und Legenden, deren Darstellung meist zu klein ist, um Texturen einzusetzen (wie in Abbildung 61 rechts), genutzt werden. Zur detaillierten Erfassung der Darstellung muss das Blinken allerdings auch abgeschaltet werden können. Es sollte also nur eine Darstellungsoption von mehreren sein. Ein Tester berichtete zudem über Übelkeit bei gewissen Blinkfrequenzen. Deshalb sollte vor einer Markteinführung von blinkenden taktilen Darstellungen (insbesondere großflächigen) untersucht werden, ob diese ähnliche Effekte hervorrufen können wie Blinken am Monitor und beispielsweise zu epileptischen Anfällen führen könnten.

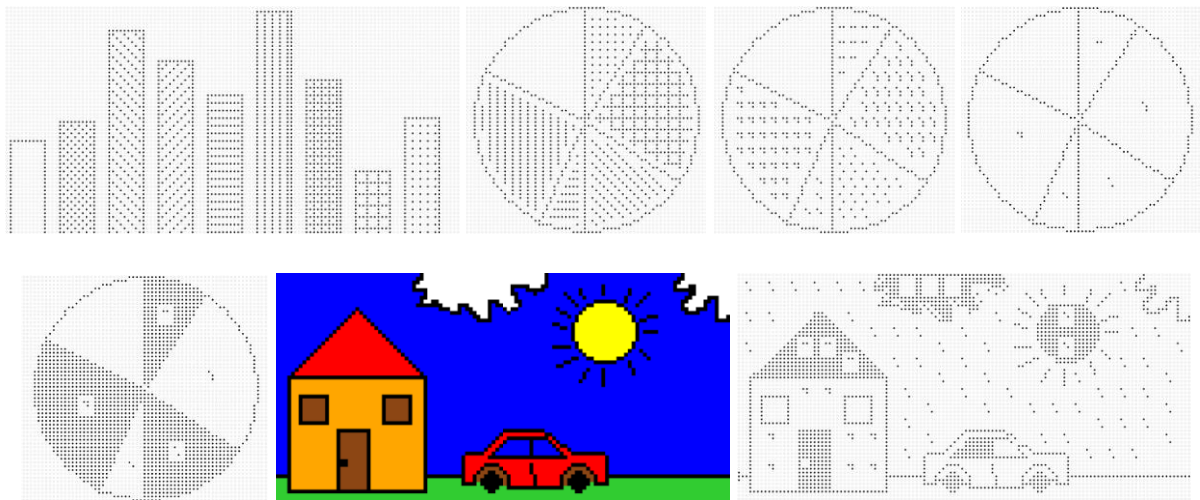


Abbildung 60 Beispiele zu taktilen Texturen und Darstellungen mit taktilen Farbcode
o. l. n. u. r.: Balkendiagramm mit verschiedenen taktilen Texturen, Kreisdiagramm mit verschiedenen taktilen Texturen, Kreisdiagramm mit Farbcode-Texturierung, Kreisdiagramm mit einzelnen Farbcode-Zeichen pro Fläche, Kreisdiagramm mit einzelnen Farbcodezeichen und wechselweiser Füllung der Flächen, Beispielbild mit einer etwas komplexeren Szene, taktiler Darstellung des Beispielbildes mit dünner Farbcodetextur und wechselweiser Füllung (soweit möglich)¹²

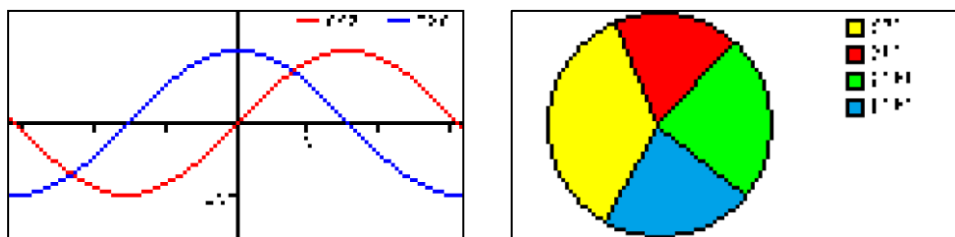


Abbildung 61 Beispiele, die zur Untersuchung von blinkenden Darstellungen eingesetzt wurden
links: Sinus (rot) und Cosinus (blau) im Koordinatensystem (schwarz) mit Legende (oben rechts)
rechts: Kreisdiagramm mit 4 Abschnitten und Legende

Mit unterschiedlichen Blinkfrequenzen lassen sich auch verschiedene Helligkeitsstufen intuitiv abbilden, wie Probanden bestätigten. Angaben zur Helligkeit werden von Nutzern häufig gut verstanden, da viele noch eine Vorstellung von Hell und Dunkel haben oder dies selbst sogar noch im Alltag unterscheiden können. Das Testbild in Abbildung 62 wurde wie folgt auf

¹² Die taktiler Darstellung wurde automatisch mit dem im Rahmen dieser Arbeit entwickelten Grafik-Editor erzeugt (siehe Abschnitt 3.3.1).

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Blinken abgebildet. Stifte in schwarzen Bereichen waren immer gehoben, Stifte in weißen Bereichen immer gesenkt. Dunkelgraue Bereiche wurden auf schnelles Blinken abgebildet, so dass die zugehörigen Stifte häufig als gehoben erfassbar waren. Je heller ein Bereich ist, desto niedriger ist die Blinkfrequenz wodurch die zugehörigen Stifte seltener als gehoben und häufiger als gesenkt wahrgenommen werden können. Ohne einen Hinweis zum Inhalt des Testbildes konnten Probanden den Helligkeitsverlauf im rechten Teil erkennen. Die Aussagen zur relativen Verteilung der Helligkeit im Balkendiagramm waren korrekt.

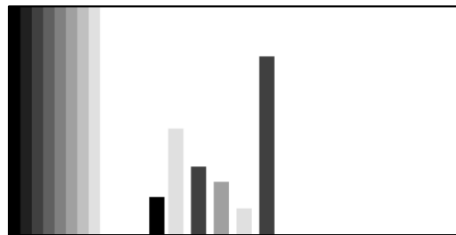


Abbildung 62 Testbild zur Untersuchung der Abbildung von Helligkeitsstufen auf Blinken

Allerdings sind natürlich auch hier den möglichen Abstufungen Grenzen gesetzt. Zu feine Unterschiede können nicht erfasst werden. Für feinstufigere Erkundung der Helligkeit ist eine benutzergesteuerte Filterung sinnvoller. Dabei kann auf zwei Weisen vorgegangen werden. Zum einen kann jeder Helligkeitsbereich einzeln dargestellt werden. So kann ein Nutzer die Helligkeitsverteilung gut erkennen. Zum anderen kann das Bild entsprechend der Helligkeit schrittweise aufgebaut werden. So können nach und nach mehr Details eines Bildes angezeigt werden, um letztlich zur Gesamtdarstellung zu gelangen. In beiden Fällen sollten Helligkeitsstufen, die keine Änderung des Bildes ergeben bzw. nicht im Bild vorhanden sind, übersprungen werden. Nutzer von grafisch-taktilen Displays müssen diese immer vollständig sequentiell abtasten, um deren Inhalt zu erfassen. Werden dem Nutzer fortlaufend Bilder geboten, die sich gegenüber dem vorangegangenen Bild nicht geändert haben, wird dem Nutzer wertvolle Zeit gestohlen. Durch eine Audioausgabe oder eine taktile Ausgabe in einem Informationsbereich sollte dem Nutzer mitgeteilt werden, welche Helligkeitsstufe angezeigt bzw. hinzugefügt wurde. Weiterhin sollte dem Nutzer die Möglichkeit gegeben werden, zu kleine Änderungen auszulassen. Dies kann geschehen, indem der Nutzer einen bestimmten Schwellwert für eine Mindeständerung angibt, vorzugsweise prozentual bezüglich der Darstellungsgröße. Eine andere Möglichkeit ist, jede Änderung darzustellen und die Anzahl der geänderten Stifte auditiv oder taktil auszugeben. So kann der Nutzer flexibel und schnell entscheiden, ob er das neue Bild erkunden möchte. Beim schrittweisen Aufbau sollten die Änderungen durch Blinken hervorgehoben werden können, um sie leicht identifizieren zu können. Weiterhin sollte der Nutzer entscheiden können, ob bezüglich der Änderungen nur der aktuell angezeigte Ausschnitt beachtet werden soll oder die gesamte Darstellung.

Wird eine Grafik skaliert dargestellt, muss zudem entschieden werden, ob die Helligkeitsverteilung der Originalgrafik oder der skalierten Grafik betrachtet werden. Beides hat Vor- und Nachteile. Durch die Skalierung der Grafik können neue Helligkeitswerte entstehen und andere verschwinden. Wird die Originalgrafik betrachtet, kann der Nutzer gut erkennen, wie

sich die Helligkeit im Original verteilt. Allerdings muss dem Benutzer klar sein, dass verschiedene Bereiche der Originalgrafik auf den gleichen Bereich der skalierten Grafik abgebildet werden, und somit in der skalierten Grafik an einer Stelle mehrere Helligkeiten auftreten können. Ist sich der Nutzer dem nicht bewusst, kann dies zu Verwirrungen führen. Bei schrittweisem Bildaufbau werden dem Nutzer dann Bilder zu Helligkeitsstufen präsentiert, ohne dass eine Änderung im Ausgabebild vorhanden ist. Wird dagegen die skalierte Grafik betrachtet, muss sich der Nutzer bewusst sein, dass die erkundbaren Helligkeitsstufen nicht mit denen in der Originalgrafik übereinstimmen. Zudem können sich diese bei einer anderen Skalierung des Bildes ändern. Dem Nutzer werden also eventuell in verschiedenen Zoomstufen unterschiedliche Helligkeitswerte angegeben. Dafür hängen die Helligkeitswerte direkt mit Änderungen im Ausgabebild zusammen.

Da Helligkeitswerte (außer hell und dunkel) selten zur Kommunikation verwendet werden, können durchaus beide Vorgehen verwendet werden. Im Umgang mit Farben sollte allerdings immer das Ausgangsbild betrachtet werden, wie das folgende Kapitel verdeutlicht.

3.2.3.4 Angepasste Skalierung zur Farbfilterung in kleinen Auflösungen

Durch die geringe Auflösung der grafisch-taktilen Displays ergibt sich bei der Segmentierung und Filterung nach Farben ein Problem. Normalerweise sind Grafiken in allgemein verfügbaren Informations- und Lehrmaterialien nicht speziell auf die Auflösung von grafisch-taktilen Displays ausgelegt (beim BrailleDis 120×60 Pixel). Bildet man dann jeden Pixel der Grafik auf einen Stift des Displays ab, so bedeutet dies, dass der Benutzer viel Scrollen muss, um die gesamte Grafik erfassen zu können. Die Sinus-Cosinus-Darstellung in Abbildung 47 mit einer Auflösung von 480×360 Pixeln ist beispielsweise 24 Mal so groß wie die Anzeigefläche des BrailleDis. Ein Benutzer müsste also mindestens 23 Scrolloperationen ausführen, um zumindest jeden Teil der Grafik einmal ertasten zu können. Zum wirklichen Verstehen der Grafik reicht dies allerdings nicht aus. Um den Linien folgen zu können und die einzelnen Teile der Grafik in Verbindung bringen zu können, müssen kleinere Bewegungen als eine Verschiebung um die gesamte Anzeigebreite oder -höhe durchgeführt werden. Der Scrollaufwand zur Erfassung dieser doch recht einfachen Grafik wäre mit dieser Darstellung somit enorm.

Die Grafik sollte zumindest für die erste Darstellung also auf die Größe des Displays skaliert werden, um zunächst einen groben Überblick zu gewährleisten. So kann der Benutzer die Struktur der Grafik erkennen und sich dann einzelne interessante Bereiche im Detail erschließen. Wie Abbildung 61 zeigt, sollten solch einfache Grafiken wie die Sinus- und Cosinus-Darstellung problemlos in 120×60 Pixel darstellbar und auch nach Farben filterbar sein. Skaliert man allerdings die Grafik aus Abbildung 47 (links) mit einem herkömmlichen Skalierungsalgorithmus, wie er in Bibliotheken zur Bildbearbeitung (beispielsweise .Net Graphics) angeboten wird, und filtert dann zur Exploration der Lage der grünen Kurve im Koordinatensystem nach grün und schwarz, so ergibt sich die wenig zufriedenstellende Darstellung, die in Abbildung 63 rechts oben gezeigt ist. Auf dem grafisch-taktilen Display werden lediglich

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

noch zwei angehobene Stifte erkennbar. Als Referenzfarben für die Segmentierung wurden für Grün $(0, 191, 0)_{\text{RGB}}$ verwendet und für Schwarz $(0, 0, 0)_{\text{RGB}}$.

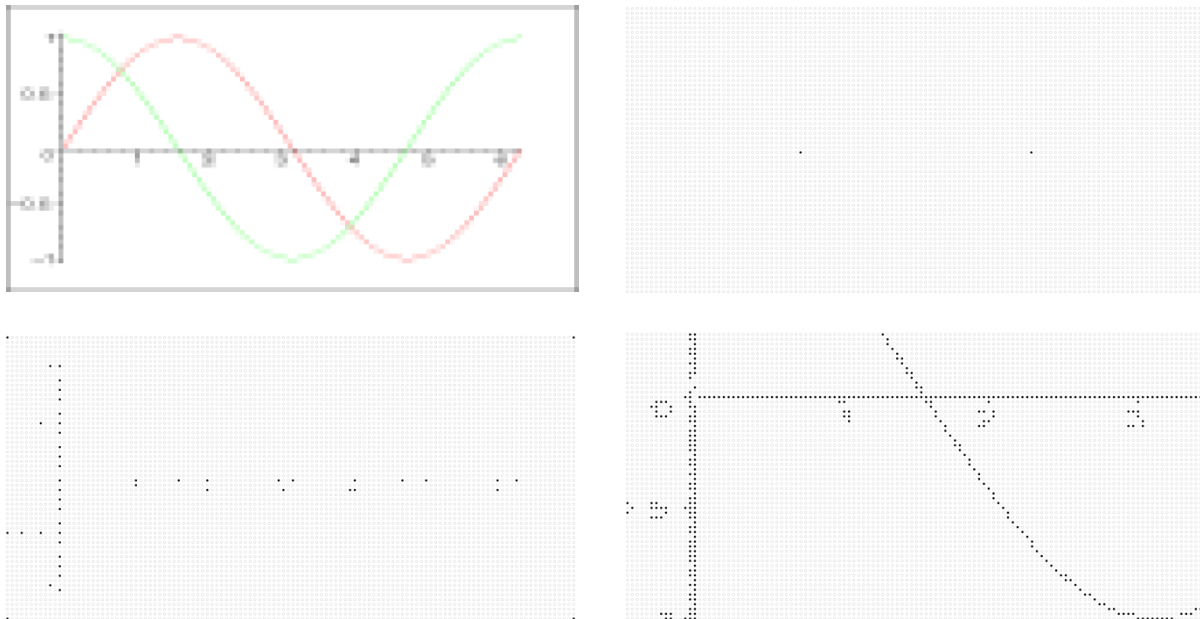


Abbildung 63 Ergebnis der Farbfilterung bei herkömmlicher Skalierung und anschließender Filterung
links oben: Abbildung 47 links skaliert auf 120×60 Pixel (vergrößerte Darstellung)
rechts oben: Ergebnis der Filterung nach Schwarz (Koordinatensystem) und Grün (Cosinus)
links unten: Ergebnis der Filterung nach Schwarz, Grün und Grau
rechts unten: Ergebnis der Filterung nach Schwarz, Grün und Grau bei Skalierung auf 240×120

In Abbildung 63 links oben, die eine vergrößerte Darstellung der auf 120×60 Pixel skalierten Grafik zeigt, ist deutlich zu erkennen, wie dieses schlechte Ergebnis entsteht. Bei herkömmlicher Verkleinerung eines Bildes werden die Farben der Pixel des Ursprungsbildes, die auf einen einzelnen Pixel des Ergebnisbildes abgebildet werden, entsprechend ihrer Anteile am Ergebnispixel gemischt. Dabei können natürlich unterschiedlichste Mischfarben entstehen, deren Ursprungsfarben auch nicht eindeutig bestimmbar sind. So entstehen bei der Skalierung der Sinus-Cosinus-Darstellung aus den ursprünglich reinen Farben Schwarz, Grün, Rot und Weiß verschiedenste Grautöne, helle Grün- und Rottöne und an einigen Stellen auch Farbwerte, die ins Braune gehen. Auf dieser Grundlage ist es schwierig und teilweise sogar unmöglich nach den Farben zu filtern, die ein Sehender im Ursprungsbild wahrnimmt und eventuell zur Beschreibung dieses Bildes nutzt. Dies wird im Beispiel in Abbildung 63 besonders am Koordinatensystem deutlich. Im Ursprungsbild ist dies schwarz. In der verkleinerten Darstellung würde auch ein Sehender die Farbe des Koordinatensystems nicht als Schwarz, sondern eindeutig als Grau bezeichnen. Mit dieser Kenntnis über Mischfarben könnte der Benutzer bei der Aufgabe, nach dem schwarzen Koordinatensystem zu filtern zwar auch selbständig entscheiden, zusätzlich Grau als Filterfarbe auszuwählen. Allerdings kann man dies erstens nur wenigen Blinden zumuten und zweitens führt es nur zu dem unbefriedigenden Ergebnis, das in Abbildung 63 links unten zu sehen ist. Gerade einmal 16 Pixel werden dargestellt, also 16 Stifte werden auf dem Display angehoben. Die restlichen Grauwerte sind so hell, dass sie vom Segmentierungsalgorithmus Weiß zugewiesen werden, da der Abstand

zu Weiß mit $(255, 255, 255)_{\text{RGB}}$ kleiner ist als zur Referenzfarbe für Grau mit $(128, 128, 128)_{\text{RGB}}$. Gleiches gilt für die grünlichen Pixel. Sie liegen näher an Weiß als am Referenzgrün mit $(0, 191, 0)_{\text{RGB}}$. Zwar ließe sich dieses Problem durch eine Verschiebung der Referenzfarben zu hellen Farbtönen bzw. die Auswahl mehrerer Referenzfarben für eine Segmentierungsfarbe lösen. Dies ist aber aus mehreren Gründen nicht zielführend. Zum einen können durch die Anpassung der Referenzfarben nie alle Mischergebnisse so abgedeckt werden, dass das erwartete Ergebnis entsteht. Die in diesem Beispiel entstehenden Pixel mit Brauntönen beispielsweise würden auch bei angepassten Referenzfarben nicht im Ergebnisbild auftauchen und somit eine lückenhafte Darstellung erzeugen, sofern eine Referenzfarbe für Braun definiert ist. Dies ist in Abbildung 63 rechts unten im Nullpunkt zu sehen. Hier entstehen Brauntöne aufgrund der Mischung von Rot und Schwarz. Zum anderen ist die für normalsichtige Menschen klare Entscheidung zu Gunsten von Grün für die Cosinus-Kurve in Abbildung 63 links oben für den Computer nicht so leicht zu treffen, da sie nicht vom Farbwert der Pixel allein abhängt, sondern unter anderem auch von den umgebenden Farben und gewissen Vorkenntnissen über die Grafik. Abbildung 64 demonstriert dies. Hier wurde die Cosinus-Kurve weiß eingefärbt und der Hintergrund grünlich gestaltet (links). Durch die Skalierung auf 120×60 Pixel (rechts) entstehen wiederum Mischwerte aus Grün und Weiß. Dabei würden Normalsichtige die Cosinus-Kurve durchaus auch in der verkleinerten Darstellung mit Weiß bezeichnen und das obwohl die grünen Mischwerte hier eigentlich sogar dunkler sind, als die der skalierten Cosinus-Kurve in Abbildung 63. Ein Algorithmus, der solche Faktoren berücksichtigt, wäre äußerst komplex und würde dementsprechende Verarbeitungszeiten erfordern, was die Exploration eines Bildes deutlich behindert.

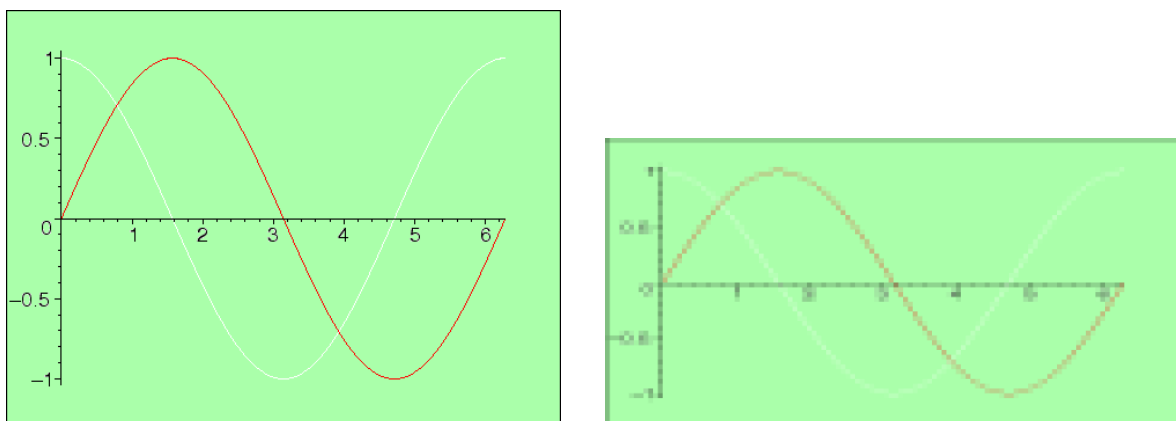


Abbildung 64 Sinus-Cosinus-Bild mit veränderten Farben (Cosinus weiß, Hintergrund hellgrün) in Originalgröße (links) und skaliert auf 120×60 Pixel (rechts, vergrößerte Darstellung)

Für eine gute Darstellung und Farbfilterung von Rastergrafiken auf grafisch-taktilen Displays ist die Anwendung eines herkömmlichen Skalierungsalgorithmus mit anschließender Segmentierung somit offensichtlich nicht geeignet.

Bessere Darstellungen ergeben sich, wenn das Ursprungsbild erst nach den gewünschten Farben gefiltert wird (also die gewünschten Farben auf Schwarz abgebildet werden) und danach auf die gewünschte Größe skaliert wird. Zwar entstehen hierbei auch wieder Misch-

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

werte, in diesem Fall verschiedene Grautöne, für die entschieden werden müsste, ob sie durch gehobene Stifte (schwarz) oder durch gesenkte Stifte (weiß) abgebildet werden sollen. Es zeigte sich aber, dass im Allgemeinen die Abbildung aller Pixel, die nicht Weiß sind, auf Schwarz, zu guten Ergebnissen führt, wie in Abbildung 65 zeigt.

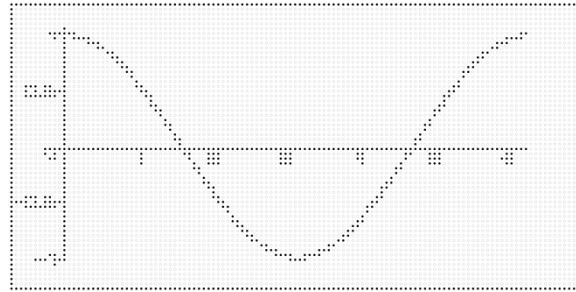


Abbildung 65 Sinus-Cosinus-Bild segmentiert nach grün und schwarz, dann skaliert und alle nicht-weißen Pixel auf Schwarz gesetzt

Ausgehend von diesen Ergebnissen wurde im Rahmen dieser Arbeit ein Algorithmus zur Segmentierung und Skalierung umgesetzt, der auch die Exploration der Grafik nach Farben optimal unterstützt. Im oben beschriebenen Vorgehen müsste bei jeder Änderung des Farbfilters eine neue Segmentierung durchgeführt werden. Auch für die Abfrage der Farbe an einer Stelle oder in einem berührten Bereich durch den Nutzer müsste zumindest dieser Bereich erneut segmentiert werden. Das Segmentierungsergebnis sollten also nach Farben getrennt gespeichert werden, um schnellen Zugriff darauf zu haben. So ist nur eine Segmentierung nötig. Diese separate Speicherung kann in einzelnen (echten) Bitmaps erfolgen, was aus Speicher und Performanzgründen (zumindest in C#) nicht zu empfehlen ist. Einfache Arrays sind ausreichend, um diese Aufgabe zu lösen. Dabei wird, wie Abbildung 66 zeigt, pro Segmentierungsfarbe ein Bit-Array angelegt, in welchem angegeben wird, ob der entsprechende Pixel die jeweilige Farbe aufweist. So ist auch die Abbildung einer Farbe auf mehrere Segmentierungsfarben möglich (beispielsweise für gelb-grüne Töne sowohl Gelb als auch Grün). Je nach Anzahl der Segmentierungsfarben kann die Information auch in einem Array mit mehreren Bits gespeichert werden. Zur Skalierung der Grafik werden die Bit-Werte der Pixel einer Farbe addiert und in kleineren Arrays zusammengefasst. Befindet sich in einem Array-Feld einer anzuzeigenden Farbe ein Wert größer 0, so wird der zugehörige Stift auf dem grafisch-taktilen Display angehoben. In den kleineren Arrays ist direkt gespeichert, welche Farbe mit welchem Anteil auf einen Stift des Displays abgebildet wurde, somit kann diese Informationen ohne weitere Berechnungen bei Bedarf an den Nutzer weitergegeben werden. Ändert der Nutzer den Farbfiler, müssen lediglich die kleinen Arrays neu ausgewertet werden.

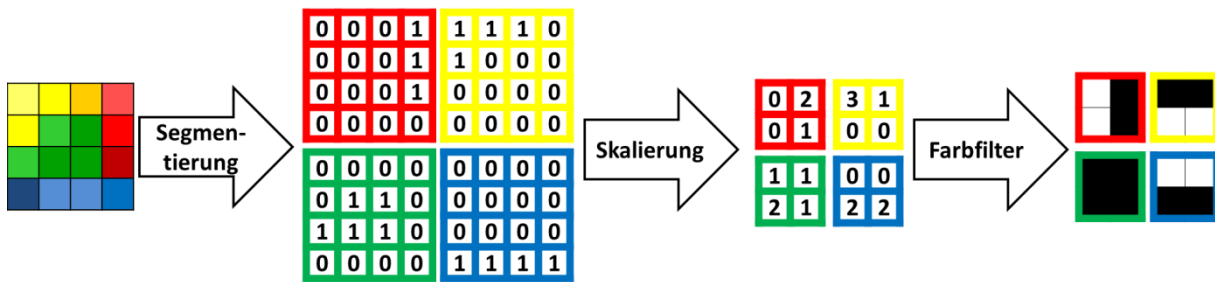


Abbildung 66 Ablauf des angepassten Skalierungsalgorithmus

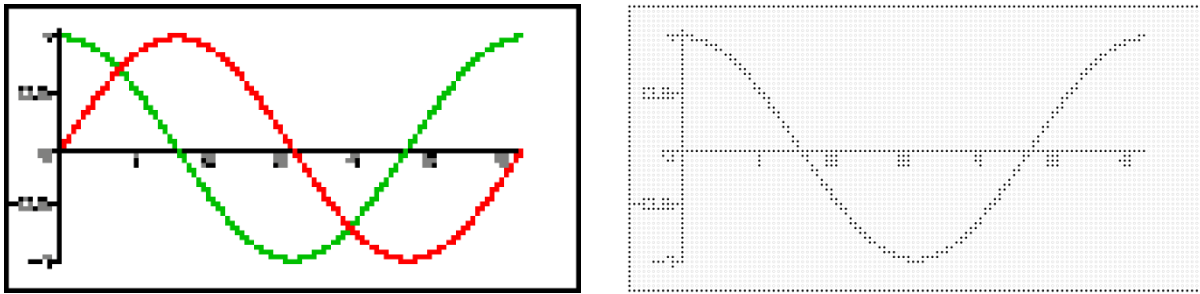


Abbildung 67 Ergebnis der Skalierung und Filterung von Abbildung 47

links: Darstellung der Skalierung (zu beachten ist, dass einige Farbflächen andere überlagern)
 rechts: gefiltert nach schwarz und grün (nach dem links angezeigten grau wurde nicht gefiltert, rechts sind die Punkte trotzdem zu sehen, weil sich an diesen Stellen auch schwarz befindet)

Wie Abbildung 67 deutlich zeigt, findet bei diesem Vorgehen natürlich eine Überbewertung der Farben statt. Die Koordinatenachsen beispielsweise sind in der verkleinerten Darstellung genauso wie in der großen Originaldarstellung immer noch ein Pixel breit. Die Darstellung ist somit genau genommen nicht mehr layoutgetreu, sondern nur layoutgerecht. Zur Erkennung der Struktur der Darstellung ist dies besonders bei dünnbesetzten Darstellungen, als Grafiken mit einem hohen Hintergrundanteil, eher nützlich als schädlich, da so keine Lücken entstehen. Bei sehr dichten Darstellungen können natürlich Lücken, die zur Erkennung der Struktur dienlich wären, verloren gehen, wie Abbildung 68 links zeigt. Besonders im letzten Kasten (ganz rechts) verändert die Überbewertung von Schwarz die Aussage des Bildes deutlich. Durch Änderung der Sortierung, in der die Farben überlagert werden, kann sich die Sichtbarkeit der Strukturen verbessern (siehe Abbildung 68 rechts).

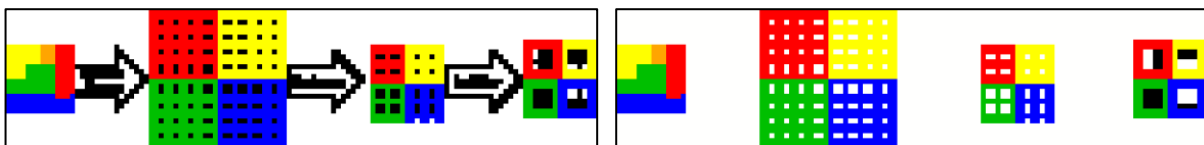


Abbildung 68 Auswirkung der Überbewertung von Farben durch den Skalierungsalgorithmus

Abbildung 66 wurde auf 120×28 Pixel skaliert

links: schwarze Bereiche überlagern weiße, rechts: weiße Bereiche überlagern schwarze

Die Überbewertung der Farben kann auch gewinnbringend bei der vergrößerten Darstellung am Monitor eingesetzt werden, um bestimmte Farbbereiche hervorzuheben. Abbildung 69 zeigt, wie auf diese Weise einzig auf Basis der Bildschirmgrafik schnell Rechtschreibfehler in

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

einem Word-Dokument hervorgehoben werden können. Da Word Rechtschreibfehler mit roten Linien markiert, kann danach gefiltert werden. Durch eine starke Verkleinerung und Filterung mit dem vorgestellten Algorithmus und anschließende Rückskalierung kann eine deutlichere Markierung geschaffen werden. Dieses Vorgehen benötigt bei effizienter Umsetzung weniger als eine Sekunde (siehe Abschnitt 3.2.3.2).

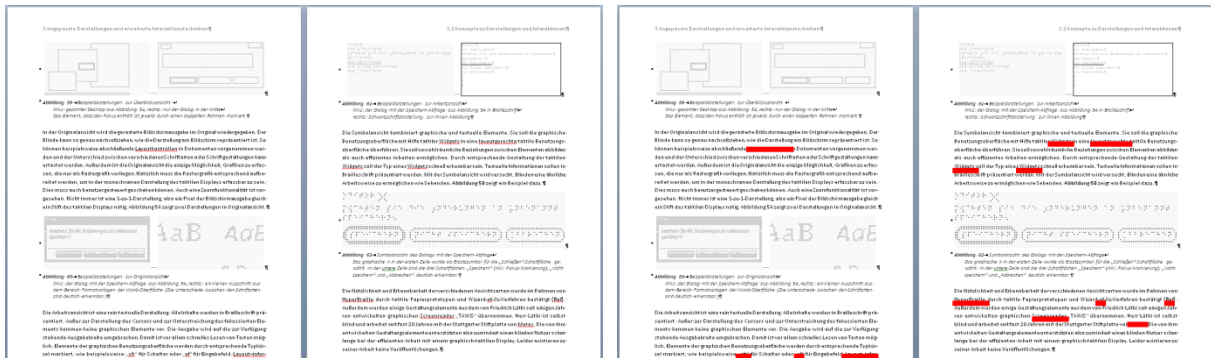


Abbildung 69 Nutzen des Skalierungsalgorithmus bei vergrößerter Bildschirmdarstellung links: Screenshot eines Worddokuments, rechts: Screenshot mit Überlagerung einer Hervorhebung von Rot (dazu wurde der linke Screenshot mit dem Skalierungsalgorithmus auf ein Zehntel verkleinert nach Rot gefiltert und wieder auf die Originalgröße vergrößert)

3.2.4 Gesamtkonzepte zur Erschließung grafischer Benutzungsoberflächen

Um grafische Oberflächen in angepassten Darstellungen durchgängig bedienbar zu machen, müssen nicht nur einzelne Konzepte zur Aufbereitung von Darstellungen und Interaktion geschaffen werden, sondern auch Gesamtkonzepte, die die einzelnen Aspekte in größeren Zusammenhang setzen. Im Folgenden werden drei solcher Konzepte präsentiert. Zunächst das Darstellungskonzept, das im Projekt HyperBraille für grafisch-taktile Displays entwickelt wurde, danach eine Adaption des HyperBraille-Konzepts für vergrößerte Darstellungen am Monitor und zuletzt ein Konzept zur Unterstützung von Nutzern, die nur eine relativ geringe Vergrößerung am Monitor benötigen. Die im Rahmen dieser Arbeit entwickelten Umsetzungen zu diesen Konzepten werden in Abschnitt 3.3 beschrieben.

3.2.4.1 Das Darstellungskonzept von HyperBraille für grafisch-taktile Displays

Wie in allen Bereichen der Mensch-Computer-Interaktion ist es auch bei der Arbeit an grafisch-taktilen Displays entscheidend, auf ein möglichst durchgängiges Darstellungs- und Bedienkonzept vertrauen zu können. Dies gibt dem Nutzer Sicherheit und erleichtert das Einarbeiten in neue Anwendungen. Dazu erarbeiteten verschiedene Partner des HyperBraille-Projektes zunächst ein Grundkonzept, welches in [SKNW09] beschrieben ist. Auch im Rahmen dieser Arbeit wurden Beiträge dazu geleistet. Das Grundkonzept definiert unterschiedliche Ansichtsarten, die sich in ihren Anteilen an textuellen und grafischen Informationen unterscheiden, und sieht eine Aufteilung der Ausgabefläche in mehrere Bereiche vor, die verschiedene klar vorgegebene Informationen enthalten. Prescher [Pre09] ergänzte das Konzept um zusätzliche Bereiche und Interaktionen zum Vergrößern und Verkleinern, sowie zum Ein- und Ausblenden der Bereiche. Insgesamt entstand so die in Abbildung 70 dargestellte Aufteilung in Darstellungsbereich, Kopfbereich, Ansichtsartenliste, Strukturleiste, Detailbereich und Fenstertitelleiste.

Der Darstellungsbereich ist der wesentliche und deshalb auch größte Teil der taktilen Ausgabe. Hier werden die Bildschirminhalte in verschiedenen Ansichten dargestellt. Er kann geteilt werden, um beispielsweise eine in einem Dokument enthaltene Grafik und deren textuelle Erläuterung gleichzeitig zu betrachten. Der Kopfbereich zeigt Informationen aus der Titelleiste oder der Menüleiste des aktiven Fensters. Die Strukturleiste gibt Hinweise zu Formatierungen eines Dokuments, wie Überschriften, Fett- und Kursivschrift, farbliche Gestaltung oder auch zur Position des Cursors, von Kommentaren oder Rechtschreibfehlern. Diese Angaben beziehen sich immer auf den aktiven Darstellungsbereich und sind passend zur dort gezeigten Darstellung in der Strukturleiste positioniert. Im Detailbereich werden ausführliche Informationen zu einzelnen Objekten präsentiert z.B. zum gerade fokussierten Objekt oder zu dem Objekt, das der Benutzer gerade im Darstellungsbereich, dem Kopfbereich oder der Strukturleiste berührt hat. Außerdem wird der Detailbereich zur Information über gerade erfolgte Interaktionen genutzt wie beispielsweise Meldungen zur aktuellen Zoomstufe. Die Fenstertitelleiste listet die Titel aller auf dem Bildschirm zu sehenden Fenster auf und mar-

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

kiert die Fenster, die gerade im Darstellungsbereich angezeigt werden. Über die Fenstertitel-
leiste können die darzustellenden Fenster auch ausgewählt werden. Die Ansichtsartenliste
gibt Auskunft darüber, welche Ansichtsart gerade im aktiven Darstellungsbereich verwendet
wird. Außerdem kann über sie die Ansichtsart gewechselt werden.

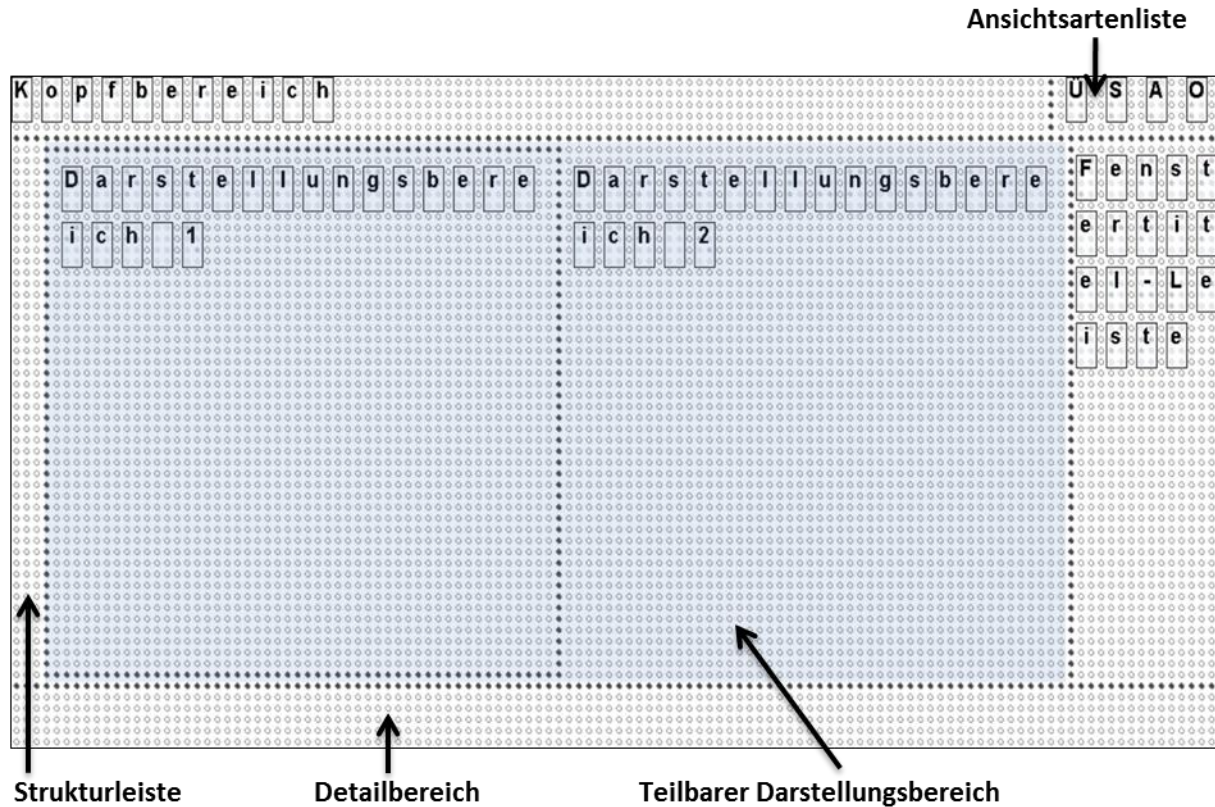


Abbildung 70 Aufteilung der Ausgabebläche des BrailleDis 9000 in verschiedene Regionen (Bereiche)

Die unterschiedlichen Ansichtsarten sind ein wesentlicher Bestandteil des Darstellungskonzepts. Es wurden vier Ansichtsarten definiert: die Überblicksansicht, die Symbolansicht, die Arbeitsansicht und die Originalansicht. Damit wurde der Tatsache Rechnung getragen, dass unterschiedliche Arbeitssituationen existieren und diese auch unterschiedliche Darstellungen benötigen.

Die Überblicksansicht soll dazu dienen, das Layout der dargestellten Objekte schnell zu erfassen und sich in der zweidimensionalen Darstellung zurechtzufinden. Dazu werden alle dargestellten Objekte layoutgetreu als einfaches Rechteck entsprechend der originalen Position und Größe dargestellt. Brailleschrift wird nicht verwendet. Die Überblicksansicht soll zoombar sein. Je nach Größe der Darstellung sollen Objekte weggelassen werden, um die Ausgabe nicht zu überfrachten. So sollen beispielsweise bei der Darstellung des gesamten Desktops auf der einfachen Größe des taktilen Ausgabegerätes nur die Rahmen der geöffneten Fenster dargestellt werden. Wird aber beispielsweise nur ein kleiner Dialog angezeigt, so sollen auch die zugehörigen Schalter erkennbar sein. Abbildung 72 zeigt zwei Beispieldarstellung von Überblicksansichten zur Bildschirmausgabe in Abbildung 71.

3.2 Konzepte zu Darstellungen und Interaktionen

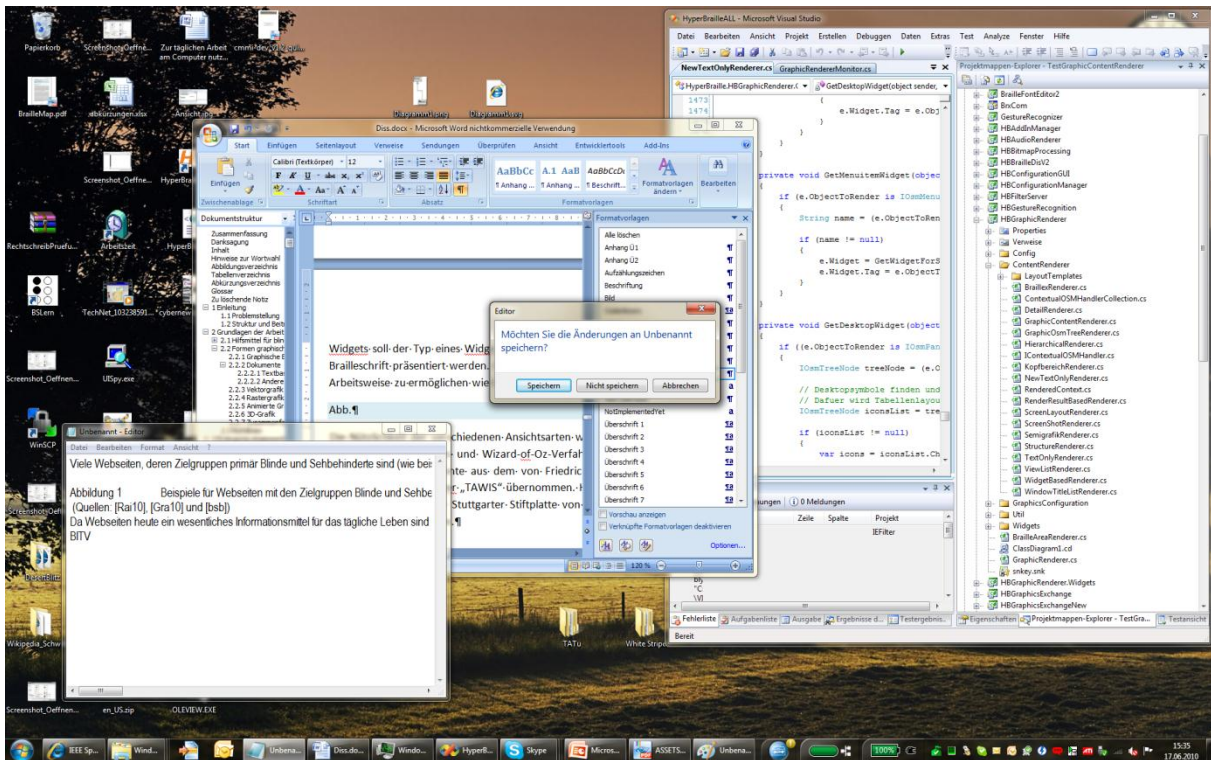


Abbildung 71 Beispiel-Screenshot zu Illustration der Ansichtsarten

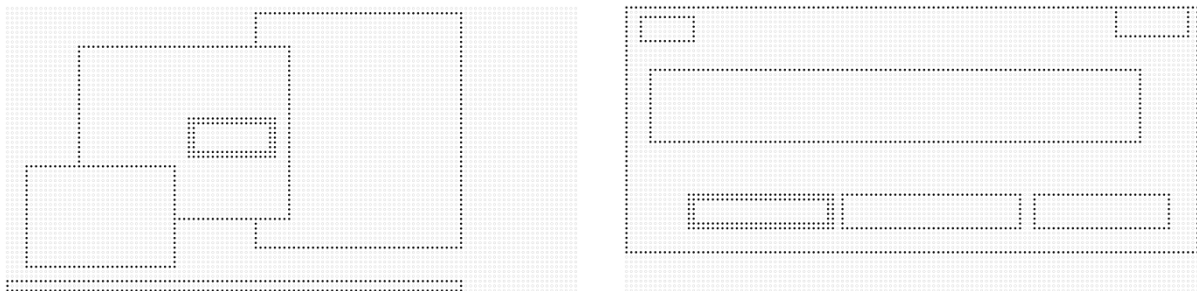


Abbildung 72 Beispieldarstellungen zur Überblicksansicht
links: gesamter Desktop aus Abbildung 71, rechts: nur der Dialog in der Mitte
Das Element, dass den Fokus enthält ist jeweils durch einen doppelten Rahmen markiert.

In der Originalansicht wird die gerasterte Bildschirmausgabe im Original wiedergegeben. Der Blinde kann so genau nachvollziehen, wie die Darstellung am Bildschirm repräsentiert ist. So können beispielsweise abschließende Layoutkontrollen in Dokumenten vorgenommen werden und der Unterschied zwischen verschiedenen Schriftarten oder Schriftgestaltungen kann ertastet werden. Außerdem ist die Originalansicht die einzige Möglichkeit, Grafiken zu erfassen, die nur als Rastergrafik vorliegen. Natürlich muss die Rastergrafik entsprechend aufbereitet werden, um in der monochromen Darstellung des taktilen Displays erfassbar zu sein. Dies muss auch benutzergesteuert geschehen können. Auch eine Zoomfunktionalität ist vorgesehen. Nicht immer ist eine 1-zu-1-Darstellung, also ein Pixel der Bildschirmausgabe gleich ein Stift des taktilen Displays nötig. Abbildung 73 zeigt zwei Darstellungen in Originalansicht.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

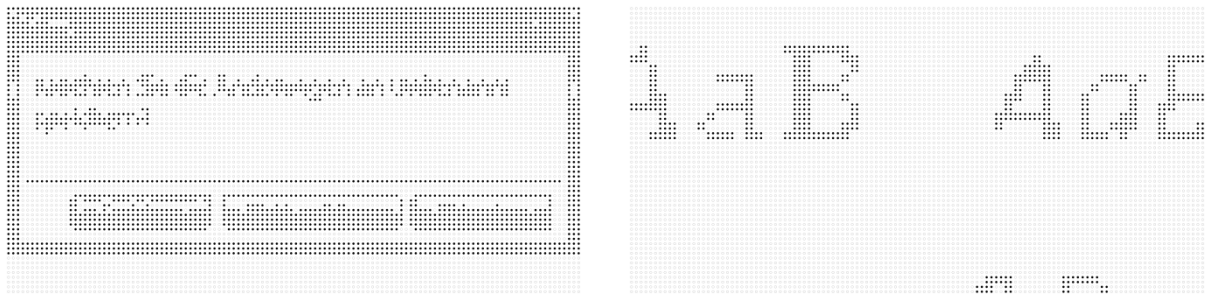


Abbildung 73 Beispieldarstellungen zur Originalansicht

links: der Dialog mit der Speichern-Abfrage aus Abbildung 71, rechts: ein kleiner Ausschnitt aus dem Bereich Formatvorlagen der Word-Oberfläche (Die Unterschiede zwischen den Schriftarten sind deutlich erkennbar.)

Die Arbeitsansicht ist eine rein textuelle Darstellung. Alle Inhalte werden in Brailleschrift präsentiert. Außer zur Darstellung des Cursors und zur Unterstreichung des fokussierten Elements kommen keine grafischen Elemente vor. Die Ausgabe wird auf die zur Verfügung stehende Ausgabebreite umgebrochen. Damit ist vor allem schnelles Lesen von Texten möglich. Elemente der grafischen Benutzungsoberfläche werden durch entsprechende Typkürzel markiert, wie beispielsweise „slt“ für Schalter oder „ef“ für Eingabefeld. Layoutinformationen werden nicht präsentiert. Lediglich für die Darstellung von Tabellen wurde eine Ausnahme zugelassen. Bei Bedarf werden diese in korrekter Zeilen und Spaltenstruktur dargestellt und dürfen in diesem Fall auch die Ausgabebreite überschreiten. Grafiken werden in der Arbeitsansicht nur über ihren Alternativtext und die Markierung „gfk“ dargestellt.

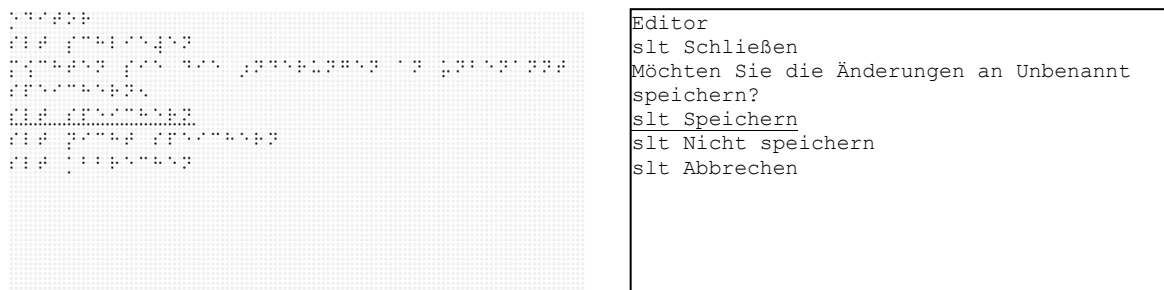


Abbildung 74 Beispieldarstellungen zur Arbeitsansicht

links: der Dialog mit der Speichern-Abfrage aus Abbildung 71 in Brailleschrift
rechts: Schwarzschriftdarstellung zur linken Abbildung

Die Symbolansicht kombiniert grafische und textuelle Elemente. Sie soll die grafische Benutzungsoberfläche mit Hilfe taktilel Widgets in eine layoutgerechte taktile Benutzungsoberfläche überführen. Sie soll sowohl räumliche Beziehungen zwischen Elementen abbilden als auch effizientes Arbeiten ermöglichen. Durch entsprechende Gestaltung der taktilel Widgets soll der Typ eines Widgets schnell erkennbar sein. Textuelle Informationen sollen in Brailleschrift präsentiert werden. Mit der Symbolansicht wird versucht, Blinden eine ähnliche Arbeitsweise zu ermöglichen wie Sehenden. Abbildung 75 zeigt ein Beispiel dazu.

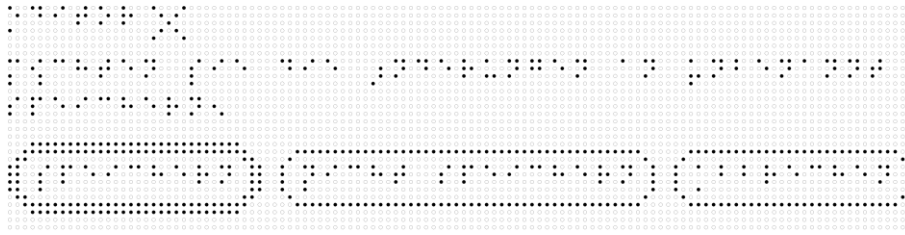


Abbildung 75 Symbolansicht des Dialogs mit der Speichern-Abfrage

Das grafische X in der ersten Zeile wurde als Ersatzsymbol für die „Schließen“-Schaltfläche gewählt. In der unteren Zeile sind die drei Schaltflächen „Speichern“ (inkl. Fokus-Markierung), „Nicht speichern“ und „Abbrechen“ deutlich erkennbar.

Listing 7 zeigt abschließend einen Entwurf aus dem Bedienkonzept, zur Darstellung einer Webseite (bzw. deren ersten Zeilen) in der Arbeitsansicht mit Kopf-, Darstellungs- und Detailbereich, sowie Strukturleiste auf der rechten Seite.

```

+-----+
|blista : Startseite          |
+-----+
|Liste, 3 Einträge          |la|
|Zur servicenavigation springen |u-|
|(Suchen, Impressum usw.)   |  |
|Zur Hauptnavigation springen |u |
|Zum Inhalt springen         |u |
|Liste Ende                  |le|
|blista logo Blista          |% |
|Deutsche Blindenstudienanstalt e.V. |  |
|Liste, 6 Einträge          |la|
+-----+
|http://www.blista.de/      |
+-----+

```

Listing 7 Beispiel zur Arbeitsansicht einer Webseite aus dem HyperBraille-Bedienkonzept

Die Nützlichkeit und Erkennbarkeit der verschiedenen Ansichtsarten wurde im Rahmen von HyperBraille durch taktile Papierprototypen und Wizard-of-Oz-Verfahren bestätigt [SKNW09]. Außerdem wurden einige Gestaltungselemente aus dem von Friedrich Lüthi seit einigen Jahren entwickelten grafischen Screenreader „TAWIS“ übernommen. Herr Lüthi ist selbst blind und arbeitet seit fast 20 Jahren mit der Stuttgarter Stiftplatte von Metec. Die von ihm entwickelten Gestaltungselemente unterstützten also zumindest einen blinden Nutzer schon lange bei der effizienten Arbeit mit einem grafisch-taktilen Display. Leider existieren zu seiner Arbeit keine Veröffentlichungen. Besonders bei der Symbolansicht kam schnell die Frage auf, ob Grafikelemente und Braille hinreichend gut voneinander unterschieden werden können. In den Benutzerstudien zeigten sich diesbezüglich keinerlei Probleme.

Aus technischer Sicht können auch Strukturleiste, Detailbereich, Fenstertitel-Leiste und Kopfbereich als Ansichtsarten angesehen werden, die lediglich verschiedene Filter auf den darzustellenden Inhalt anwenden. Beispielsweise kann eine strukturelle Ansicht auch in Größe des Darstellungsbereichs sinnvoll sein, um die Gliederung eines Briefes zu überprüfen. Prinzipiell können aus technischer Sicht durch flexible Filter beliebige Ansichten erstellt wer-

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

den. Wie aber bereits bei der Filterung angemerkt, kann dies den Nutzer überfordern. Eine gewisse Einschränkung der Möglichkeiten hilft dem Nutzer, sich im System zurechtzufinden.

Das Darstellungskonzept wurde mit Hilfe des in Abschnitt 3.3.2 beschriebenen Rendering-Frameworks für grafisch-taktile Displays umgesetzt und während des HyperBraille-Projektes mehrfach mit positiven Ergebnissen evaluiert.

3.2.4.2 Adaption des HyperBraille-Konzepts für Bildschirmausgabe

Das Darstellungskonzept von HyperBraille lässt sich auch auf vergrößerte Darstellungen für Sehbehinderte übertragen. So können auch dafür die verschiedenen Darstellungsoptionen genutzt werden und sowohl effizientes Arbeiten als auch das Erfassen der Originaldarstellung unterstützt werden. Abbildung 76 zeigt Beispiele, wie eine direkte Übertragung der oben beschriebenen Ansichten in visuelle Darstellungen aussehen kann. Diese sind natürlich auf die Ausdehnungen des Monitors als zur Verfügung stehende Ausgabefläche beschränkt. Da sich in der Bildschirmdarstellung Texte und Grafiken leichter unterscheiden lassen, als in der taktilen Darstellung können mehr Gestaltungsmöglichkeiten genutzt werden als auf dem grafisch-taktilen Display. Es können auch Farben eingesetzt werden. Natürlich werden sämtliche Texte in den verschiedenen Bereichen und Ansichten (ausgenommen Überblicks- und Originalansicht) in einer vom Benutzer gewählten Schrifteinstellung dargestellt.

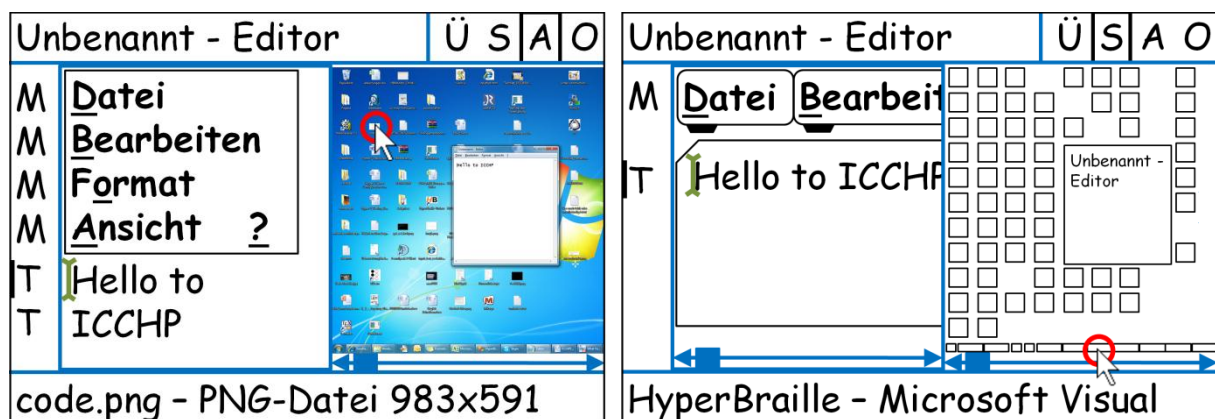


Abbildung 76 Darstellungen für Sehbehinderte abgeleitet aus dem HyperBraille-Konzept
Beide Darstellungen zeigen den Kopfbereich mit Ansichtsartenliste, die Strukturleiste, den Detailbereich und einen geteilten Darstellungsbereich mit verschiedenen (v.l.n.r.: Arbeitsansicht des Editors, Originalansicht des Desktops mit Editor-Fenster, Symbolansicht des Editors und Überblicksansicht des Desktops mit Editor-Fenster). Mauszeiger und Textcursor sind jeweils deutlich hervorgehoben. In der Strukturleiste sind Markierungen für Menüeinträge und normalen Text. Der Detailbereich zeigt jeweils Informationen zu dem mit dem Mauszeiger ausgewählten Objekt.

Die Darstellungen in Abbildung 76 zeigen deutlich, dass die verschiedenen Ansichten in der Bildschirmdarstellung leicht zu unterscheiden sind. Somit ist die Ansichtsarten-Liste hier nicht notwendig. Die Umschaltung zwischen den Ansichten kann genauso gut über ein besonderes Menü erfolgen. Der dadurch gewonnene Platz kann genutzt werden, um mehr Informationen im Kopfbereich zu platzieren. So könnten z.B. die wichtigsten Bedienelemente einer aktivierten Werkzeugleiste angezeigt werden, um diese schnell erreichen zu können,

wie Abbildung 77 links zeigt. Des Weiteren kann in der visuellen Darstellung mit Überlagerungen gearbeitet werden. So können Originalansicht und Überblicksansicht zu einer Ansicht vereint werden, die die Möglichkeit bietet, wahlweise einen aufbereiteten Screenshot oder lediglich die, die Ausdehnungen der Objekte darstellenden Rechtecke anzuzeigen, oder auch beides kombiniert. Auch die Menüdarstellung kann als Overlay der vergrößerten Darstellung mit Hilfe üblicher Menü-Layout-Algorithmen umgesetzt werden (Abbildung 77 rechts). Das Menü muss dabei natürlich auf die Ausgabefläche begrenzt werden, um zusätzliches Scrollen zu vermeiden. So können Menüdarstellungen wie in Abbildung 30 rechts erreicht werden.

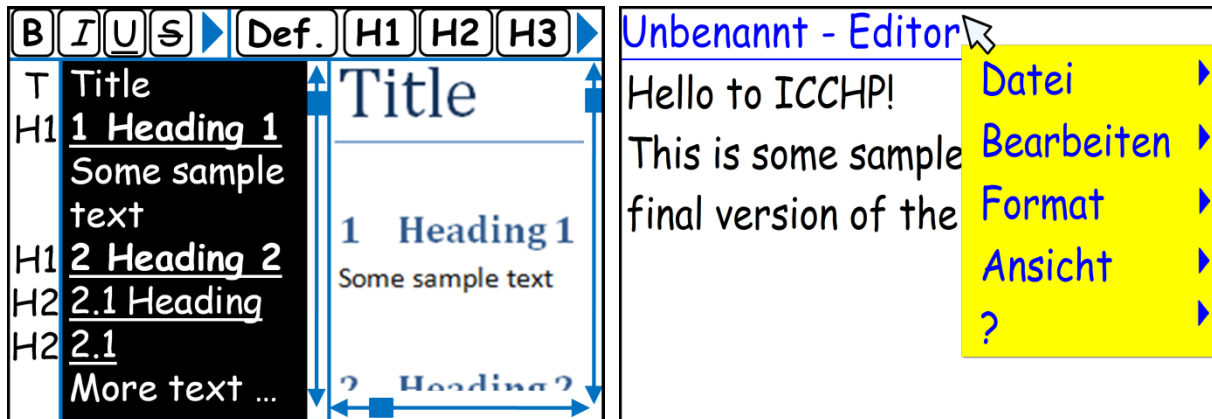


Abbildung 77 Entwürfe zu abgewandelten, vergrößerten Darstellungen auf Basis von HyperBraille

Die gezeigten Darstellungen können natürlich nicht durch bloße Beeinflussung des Darstellungsprozesses von Anwendungen erreicht werden. Hier ist wie für die taktile Darstellung auch ein eigener Darstellungsprozess nötig.

Abschnitt 3.3.3 beschreibt Aspekte, die bei der prototypischen Umsetzung dieses Vergrößerungskonzeptes zu Tage traten.

3.2.4.3 Partielle Vergrößerung in der Originaldarstellung

Derzeitig verfügbare Bildschirmvergrößerungssoftware wird vor allem für Menschen entwickelt, die unter starken Seheinschränkungen leiden und offiziell als sehbehindert gelten. Auch die im vorigen Abschnitt beschriebene Adaption des HyperBraille-Konzepts für die Bildschirmvergrößerung wurde für diese Zielgruppe entworfen. Durch die zunehmende Alterung unserer Gesellschaft werden Softwarehersteller in Zukunft aber mit einer wachsenden Gruppe von älteren und dadurch leicht bis mittelstark seheingeschränkten PC-Benutzern konfrontiert werden. Bei diesen Benutzern steigt das Bedürfnis nach flexibler und schnell verfügbarer Vergrößerung, die die gewohnte Arbeitseffizienz erhält. Eine Umstellung auf spezielle Vergrößerungssoftware wird in dieser Gruppe nur schwer stattfinden, auch wenn diese hilfreich sein könnte. Es würde aber zum einen eine Umstellung der Arbeitsweise und Gewöhnung an neue, sich von den gewohnten deutlich unterscheidende Darstellungen bedeuten und zum anderen auch, dass der Benutzer sich selbst als sehbehindert akzeptiert. Mit Blick auf diese Benutzergruppe wurde im Rahmen dieser Arbeit das an Fokus-und-

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Kontext-Techniken angelehnte Konzept der partiellen Vergrößerung in der Originaldarstellung entwickelt.

Durch Betrachtung bestehender Vergrößerungstechniken ist leicht zu erkennen, dass die Erhaltung des Layouts und Gewährung des Überblicks bei gleichzeitiger und permanenter Vergrößerung aller auf dem Bildschirm befindlichen Elemente nicht möglich ist. Allerdings ist es gar nicht nötig, alle Elemente gleichzeitig zu vergrößern. Der Benutzer ist je nach Arbeitssituation meist nur an einem Teil der grafischen Oberfläche wirklich interessiert. Ist er beispielsweise gerade mit dem Lesen eines Dokumentes beschäftigt, so müssen Menü und Symbolleisten der zugehörigen Anwendung nicht vergrößert dargestellt sein. Sie sollten aber auf jeden Fall schnell, erreichbar sein, so dass der Benutzer effizient mit dem Programm arbeiten kann. Um dies zu erreichen, sieht das Konzept vor, dass die Originaldarstellung von Anwendungen als Grundlage erhalten bleibt. Vor allem findet keine Ausdehnung des zur Verfügung stehenden Ausgabebereichs statt. Die Darstellung bleibt auf die Monitorgröße bzw. die vom Benutzer gewählte Fenstergröße einer Anwendung beschränkt. Die Übersicht soll dadurch gewährleistet werden, dass keine bzw. nur wenige Elemente permanent vergrößert werden. Dies erhält auch den für das Anwendungsdokument zur Verfügung stehenden Platz innerhalb eines Anwendungsfensters. Vergrößert werden soll nur, was den Benutzer in der aktuellen Arbeitssituation interessiert. Dies kann z.B. durch die Position des Mauszeigers oder des Eingabecursors, durch Eye-Tracking oder über eine Auswahl von Elementen durch den Benutzer ermittelt werden. Dabei werden, wie in Abschnitt 3.1.2.2 gefordert, nicht einzelne Elemente vergrößert, sondern ganze Arbeitsbereiche, die in semantischem Zusammenhang stehen (z.B. die gesamte Menüleiste oder eine Toolbar im Ganzen und auch Tooltips, die zu diesen Elementen gehören). Für Elemente, die für den Benutzer auch interessant sein können, wenn er sie nicht fokussiert, wie beispielsweise die Statusleisten, sollte eine permanente Vergrößerung einstellbar sein. Indirekt findet so auch eine Beachtung der verschiedenen Arbeitsbereiche statt. Abbildung 78 zeigt einen Vergleich der partiellen Vergrößerung, wie sie in dem Prototypen, der in Abschnitt 3.3.4 beschrieben ist, umgesetzt wurde mit der Vergrößerung auf ein Vielfaches der Bildschirmdarstellung mit einem geöffneten Menü. In der partiellen Vergrößerung ist die gesamte Menüleiste sichtbar, der Benutzer kann also alle Auswahlmöglichkeiten erkennen. Für das Anwendungsdokument steht mehr Platz zur Verfügung. Zudem sind auch die Statusleiste des Anwendungsfensters und die allgemeine Taskleiste sichtbar und schnell erreichbar. Meldungen, die im Info-Bereich der Taskleiste erscheinen, und Hinweise in der Statusleiste, wie beispielsweise die Zieladresse eines Hyperlinks, können so wahrgenommen werden. Bei der Vergrößerung auf ein Vielfaches ist dies nicht möglich. Insbesondere bei dem Menü zeigt sich, dass durch die Beibehaltung des Ausgabebereichs die vorhandenen Layoutalgorithmen dazu führen, dass alle Elemente innerhalb des Bildschirms sichtbar bleiben.

3.2 Konzepte zu Darstellungen und Interaktionen

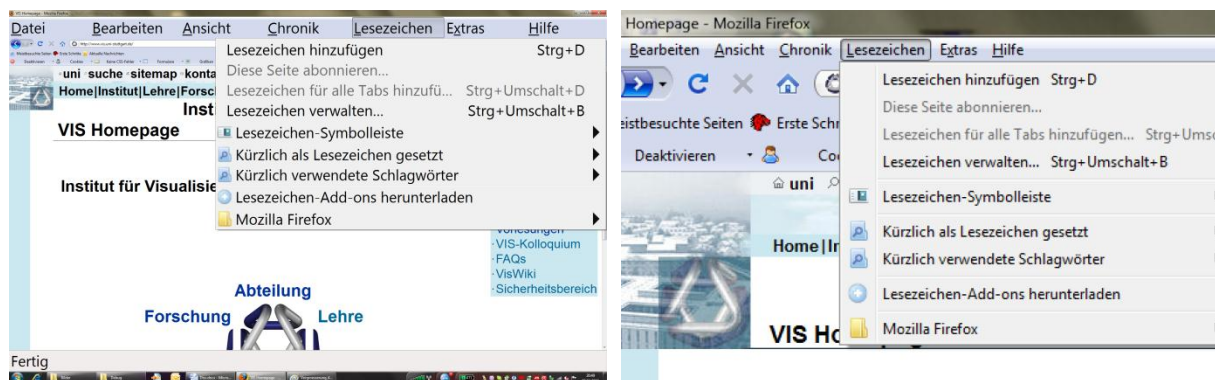


Abbildung 78 Vergleich zwischen partieller Vergrößerung (links) und Vergrößerung auf ein Vielfaches der Bildschirmgröße (rechts) (Vergrößerung entspricht 4-fach; zum besseren Vergleich wurde links auch das Anwendungsdokument vergrößert, wobei Bilder nicht skaliert wurden)

Die Vergrößerung soll natürlich immer so erfolgen, dass textuelle Elemente in einer vom Benutzer festgelegten Schriftgröße angezeigt werden. Wie in der Symbol- und Arbeitsansicht der aus HyperBraille abgeleiteten Vergrößerung wird die Darstellung entsprechend der Anforderung aus Abschnitt 3.1.2.3 also nicht einfach prozentual vergrößert, sondern an die Schriftgröße angepasst. Dabei muss unterschieden werden zwischen Elementen, die in der Benutzungsoberfläche direkt sichtbar sind und Elementen, die durch Benutzer-Interaktionen erst sichtbar werden. Elemente, die direkt sichtbar sind, müssen für den Benutzer ohne Probleme anvisierbar sein und dürfen so ihre Position bei Fokussierung, welche ja zur Vergrößerung führt, nicht verändern. Bereits sichtbare Elemente müssen somit auch bei Vergrößerung auf ihre ursprüngliche Ausdehnung beschränkt werden. Elemente, die erst später sichtbar werden und auch wieder verschwinden, wie beispielsweise Menüs oder kleine Dialoge, sollten, sofern sie dann noch vollständig sichtbar bzw. bedienbar sind, schon kurz vor ihrem Erscheinen vergrößert werden. Sie sind im Moment ihres Erscheinens auch meist für den Benutzer interessant und sollten deshalb sofort lesbar sein. Da der Benutzer diese Elemente dann erst im vergrößerten Zustand sieht, führt eine Veränderung der absoluten Position dieser (oder der darin enthaltenen) Elemente im Vergleich zur Originaldarstellung durch die Vergrößerung nicht zu einer Irritation beim Benutzer. Die relative Positionierung der Elemente zueinander sollte gewahrt bleiben, um die Erwartungen über die Anordnung der Elemente, die sich der Benutzer eventuell in jahrelanger Arbeit mit der Anwendung aufgebaut hat, zu erfüllen und Kommunikationsprobleme zu vermeiden. Leerräume sollten zum Zwecke der optimalen Platzausnutzung möglichst klein gehalten werden.

Gerade bei kleinen, aber meist wichtigen Dialogen, wie beispielsweise einer Passwortabfrage zeigt sich deutlich, dass die Beschränkung auf die Monitorgröße als Ausgabefläche und die vergrößerte Darstellung nicht unbedingt im Widerspruch stehen, sondern eine Kombination sogar Vorteile bringen kann. Da solche kleineren Dialoge meist in der Mitte der Bildschirmdarstellung auftauchen, muss bei einer Vergrößerung auf ein Vielfaches der Bildschirmgröße die Ansicht des Nutzers häufig verschoben werden, um die Dialog sichtbar zu machen. Dadurch kann ein Benutzer leicht verwirrt werden oder gar die Orientierung verlieren. Falls die Verschiebung nicht automatisch passiert, muss der Benutzer diese Dialoge sogar erst suchen.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Durch die Beschränkung des Ausgabebereichs öffnen sich solche Dialoge immer im Sichtfeld des Benutzers. Aufgrund ihrer meist geringen Größe ist trotzdem eine gute Vergrößerung zu erreichen, wie Abbildung 79 zeigt. Größere Dialoge, die nicht gänzlich vergrößert auf den Bildschirm passen, sollten wie Anwendungsfenster behandelt werden. Sie bestehen meist aus mehreren Teilbereichen, die auch getrennt je nach Bedarf vergrößert werden sollten.

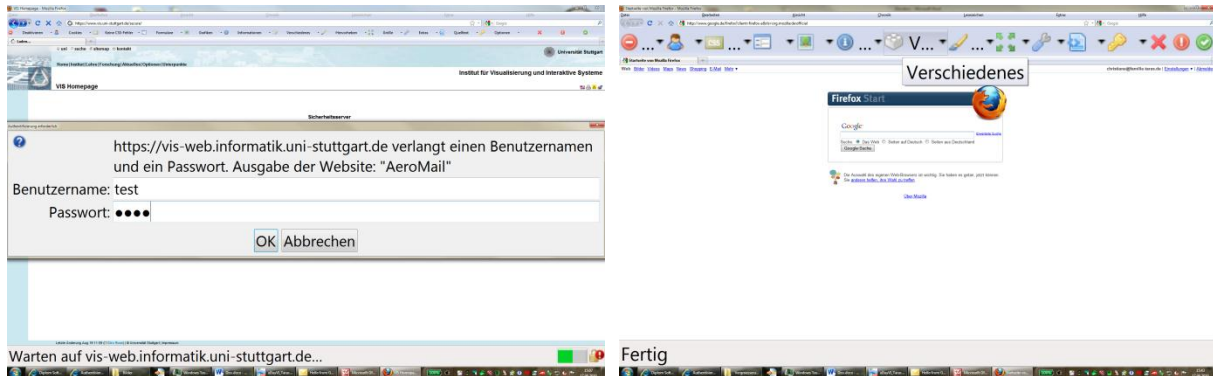


Abbildung 79 Beispiele zur partiellen Vergrößerung in der Originaldarstellung
links: ein kleiner Dialogs zur Passwortabfrage mit Schriftgröße 48 pt (4-fache Schriftgröße)
rechts: eine Werkzeugleiste mit Schriftgröße 60 pt (5-fache Schriftgröße)

Die partielle Vergrößerung in der Originaldarstellung bringt zumindest für Benutzer mit leichten bis mittelstarken Seheinschränkungen deutliche Vorteile. Der Überblick über die Bildschirmausgabe bleibt erhalten. Die Kombination der partiellen Vergrößerung mit anwendungsbezogenen Vergrößerungsmöglichkeiten kann nahtlos erfolgen. In den Anwendungen integrierte Layoutalgorithmen, die die Darstellung auf die Ausgabebreite beschränken können genutzt werden. Horizontales Scrollen wird dadurch soweit wie möglich eingeschränkt. Natürlich hat das Konzept durch die Beschränkung auf die Bildschirmgröße auch Grenzen. Abbildung 79 rechts zeigt deutlich, dass die maximale Vergrößerung bzw. die maximal verwendbare Schriftgröße im Vergleich zu Bildschirmvergrößerungssoftware sehr beschränkt ist. Bei der in Abbildung 79 mit einer Schriftgröße von 60 pt dargestellten Werkzeugleiste, was in etwa einer Vergrößerung auf das 5-fache entspricht, sind nur noch die Icons der einzelnen Einträge sichtbar. Der für die einzelnen Einträge zur Verfügung stehende Platz reicht nicht aus, um weitere Informationen darzustellen. Eine noch stärkere Vergrößerung ist kaum sinnvoll. Bildschirmvergrößerungssoftware bietet dagegen Vergrößerungen bis zu 60-fach an, wobei laut [INC08] höhere Vergrößerungsstufen als 16-fach kaum genutzt werden. Allerdings können bei 16-facher Vergrößerung bei einer Schriftart von 12 pt noch etwa drei bis vier Zeilen Text in der Höhe auf modernen Monitoren dargestellt werden, bei 60-fach nur noch eine Zeile. Diese Vergrößerungsstufen sind also für die zuvor genannte Zielgruppe weniger interessant. Eine 5-fache Vergrößerung ist mehr als ausreichend, wie erste Nutzerreaktionen zeigten.

Im Allgemeinen nutzt die partielle Vergrößerung in der Originaldarstellung aus, dass für grafische Bedienelemente häufig mehr Platz zur Verfügung steht, als von diesen tatsächlich genutzt wird. Diese Leerräume werden bei Vergrößerung auf ein Vielfaches einfach mit skaliert und nicht zur Darstellung von Informationen genutzt. Um auch Leerräume nutzen zu kön-

nen, die sich hinter oder unter einer Reihe direkt sichtbarer Elemente wie beispielsweise Menüs befinden, welche sich bei Vergrößerung ja nicht verschieben sollen, kann eine gewisse Anpassung der Originaldarstellung vor Vergrößerung nützlich sein, um die Leerräume gleichmäßig auf die Elemente zu verteilen und damit als Darstellungsplatz zur Verfügung zu stellen. Abbildung 80 illustriert dies am Beispiel einer Menüleiste. Durch diese leichte Anpassung ist keine wirkliche Umgewöhnung des Benutzers bei der Bedienung einer Anwendung nötig. Der Grundidee des Konzepts wird dadurch nicht widersprochen.

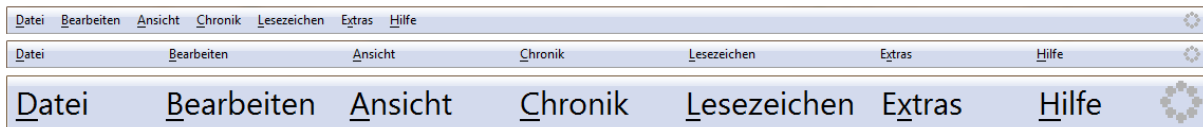


Abbildung 80 Vorbereitung und Vergrößerung der Menüleiste
v. o. n. u.: normale Darstellung der Menüleiste, vorbereitete Menüleiste, vergrößerte Menüleiste

Bei sehr dichtbesetzten Oberflächen wie beispielsweise den Ribbons von Microsoft Office stehen kaum nutzbare Leerräume zur Verfügung. Hier kann die partielle Vergrößerung nicht aus der Originaldarstellung heraus geschehen. Es können aber Gruppierungstechniken verwendet werden, wie sie bereits für die Ribbon-Darstellung in kleinen Fenstern umgesetzt sind (siehe Abbildung 17). Solche Gruppierungen stellen allerdings eine deutliche Umgestaltung der Bedienoberfläche dar und erfordern somit auch eine Umgewöhnung des Benutzers, was zu Irritationen führen kann. Eine so umgestaltete Darstellung kann nicht wirklich als Originaldarstellung betrachtet werden.

3.2.5 Kooperation zwischen Nutzern grafisch-taktiler Displays und Sehenden

Mit Hilfe des HyperBraille-Darstellungskonzepts sowie den in 3.2.2 und 3.2.3 beschriebenen Konzepten, können sich blinde Nutzer von grafisch-taktilen Displays die „Welt der Sehenden“ gut erschließen und Erklärungen Sehender über die Darstellung gut verstehen. Da sich die auf dem taktilen Display präsentierte Darstellung allerdings stark von der Bildschirmausgabe unterscheiden kann, sollte die Kommunikationsbasis auch in die andere Richtung erweitert werden. Dazu muss es Sehenden möglich sein, die auf dem grafisch-taktilen Display gebotene Darstellung leicht zu erfassen, wobei dem Sehenden nicht mehr Informationen zur Verfügung stehen sollten, als dem Blinden. Nur so kann der Sehende auch eventuell auftretende Schwierigkeiten des Blinden wirklich verstehen. Ideal wäre es, wenn die Darstellung am Bildschirm exakt das Stiftmuster zeigen würde, das auf dem Ausgabegerät tatsächlich dargestellt ist inklusive eventuell aufgetretener Fehlsetzungen von Stiften. Hierzu wäre natürlich eine entsprechend zuverlässige Rückmeldung des Ausgabegeräts nötig. Ist eine solche Rückmeldung nicht verfügbar, so kann dem Sehenden nur gezeigt werden, welches Stiftmuster zur Darstellung an das Ausgabegerät weitergeleitet wurde. Ist das Ausgabegerät zuverlässig, so stimmen auch beide Darstellungen überein. Die Ausgabe für Sehende sollte die Anzeigefläche des grafisch-taktilen Displays möglichst exakt wiedergeben, das heißt Größe

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

und Abstände der Stifte sollten möglichst originalgetreu wiedergegeben werden, um dem Sehenden deutlich zu zeigen, dass für den Blinden eigentlich keine durchgezogenen Linien oder ähnliches existieren.

Sofern das grafisch-taktile Display eine Erkennung der Fingerposition des Blinden ermöglicht, sollte diese dem Sehenden auch angezeigt werden. So kann der Blinde einen sehenden Kommunikationspartner leicht auf einen bestimmten Teil der Ausgabe hinweisen. Außerdem können Sehende so leichter verstehen, wie Blinde die taktile Ausgabe erkunden. Das BrailleDis 9000 bietet hier durch seine berührungsempfindliche Oberfläche mit 720 einzelnen Eingabemodulen hervorragende Möglichkeiten, wie Abbildung 81 zeigt. Umgekehrt sollte auch der Sehende die Möglichkeit haben, durch Interaktion mit der Ausgabe für Sehende den blinden Kommunikationspartner auf eine bestimmte Stelle in der Ausgabe hinzuweisen. Dies könnte beispielsweise geschehen, indem der Mauszeiger, sofern er sich innerhalb der Ausgabe für Sehende befindet auf dem grafisch-taktilen Display an passende Stelle als blinkendes Rechteck dargestellt wird.

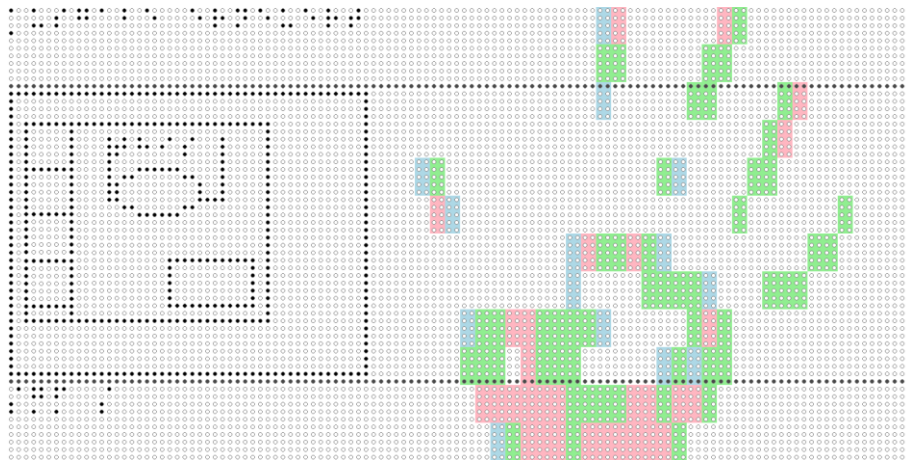


Abbildung 81 Darstellung der durch Berührung mit der flachen Hand ausgelösten Eingabesignale an einem BrailleDis 9000 in einer Ausgabe für Sehende
rote Bereiche zeigen starke Berührung an, grüne mittelstarke und blaue schwache Berührung

Da Sehende nur selten Brailleschrift vollständig beherrschen, sollten für sie die in Brailleschrift ausgegebenen Texte in Schwarzschrift dargestellt werden. Dies widerspricht nicht dem Anspruch, dem Sehenden nicht mehr Information zu bieten, als dem Blinden. Vielmehr stellt es sicher, dass beide Kommunikationspartner die gleichen Informationen bekommen, zumindest wenn der blinde Nutzer Brailleschrift lesen kann. Da der Blinde für ihn lesbare Texte auch zur Kommunikation nutzen wird, ist es essentiell, dass der Sehende leicht die durch die Stiftmuster gezeigten Texte identifizieren kann. Vor allem dann, wenn Texte verändert wurden, um eine kompaktere Darstellung zu erreichen (z.B. „abbr.“ statt „Abbrechen“ als Beschriftung einer Schaltfläche) oder auch wenn Texte als Auszeichnung von Bedienelementen genutzt werden und somit in der Originaldarstellung gar nicht vorhanden sind (z.B. „slt“ zur Markierung von Schaltflächen in der Arbeitsansicht).

Grundsätzlich bestehen zwei Möglichkeiten, textuelle Inhalte der grafisch-taktilen Ausgabe für Sehende lesbar zu machen:

1. Die direkte Darstellung der ausgegebenen Texte in Schwarzschrift
2. Die Rückwandlung des ausgegebenen Stiftmusters in Schwarzschrift

Beide Möglichkeiten haben Vor- und Nachteile.

Die direkte Darstellung der ausgegebenen Texte in Schwarzschrift bedeutet, dass bei der Erstellung der Ausgabe gespeichert wird, welche Zeichen wohin ausgegeben wurden. In der Darstellung für Sehende können dann an diesen Stellen direkt die passenden Schwarzschriftzeichen platziert werden. Abbildung 70 zeigt eine solche Darstellung. Der Sehende hat so einen sehr schnellen und direkten Zugang zu den Texten. Allerdings hat er auch einen gewissen Vorteil gegenüber dem Blinden. Da die heute üblichen grafisch-taktilen Displays keine Unterscheidung zwischen Stiften, die zur Anzeige von Schrift dienen, und solchen, die zur Anzeige von Grafik dienen, zulassen, könnte theoretisch jeder Stift zu einem Schriftzeichen gehören. Der Blinde muss erst identifizieren, welche Stifte im Zusammenhang wirklich sinnvolle Texte ergeben und somit welche Texte in der Ausgabe tatsächlich dargestellt sind. Der Sehende bekommt diese Information mit der direkten Darstellung sofort. Für den Sehenden entstehen durch die direkte Darstellung allerdings auch Nachteile. Erkennt der Blinde Texte in Stiftmustern, die ursprünglich nicht durch die Ausgabe von Texten entstanden, so kann der Sehende den Ausführungen des Blinden nicht folgen, wenn dieser über die erkannten Texte spricht, oder auch dessen Fragen zu den eventuell seltsamen Texten nicht nachvollziehen. Weiterhin ist die Übersetzung von Schwarzschrift in Brailleschrift aufgrund der begrenzten Möglichkeiten von Punktkombinationen in einem Braillezeichen nicht uneindeutig. Gleiche Zeichen in Brailleschrift können unterschiedliche Schwarzschriftzeichen repräsentieren. Häufig ist der Kontext entscheidend, um die richtige Bedeutung eines Braillezeichens zu erkennen. Beispielsweise werden Zahlen in 6-Punkt-Braille durch Buchstaben repräsentiert, denen das sogenannte Zahlzeichen als Ankündigung des Kontextes „Zahl“ vorangestellt ist. Fehlt dieser Kontext (z.B. durch Verschiebung der Ausgabe) oder wird er vom blinden Nutzer nicht erfasst, so kann es durchaus vorkommen, dass dieser ein Braillezeichen falsch interpretiert und beispielsweise über Buchstaben statt Zahlen spricht. Die direkte Darstellung der ausgegebenen Texte in Schwarzschrift verwehrt dem Sehenden den Zugang zu solchen Informationen und kann dadurch die Kommunikation mit dem Blinden hemmen. Hier kann eine Schnittstelle helfen, die es dem Sehenden erlaubt, eine Rückwandlung des ausgegebenen Stiftmusters in Schwarzschrift durchzuführen. Diese Schnittstelle sollte dabei nicht intelligent sein und automatisch erkennen, wo sinnvoller Text steht. Vielmehr sollte sie beginnend bei einem vom Sehenden frei wählbaren Startpunkt einen einfachen Mustervergleich durchführen und eventuelle Mehrdeutigkeiten anzeigen. So hat der Sehende Zugang zu allen möglichen Zeichenkombinationen, die der Blinde erkennen könnte. Die Schnittstelle könnte wie in Abbildung 82 aussehen.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

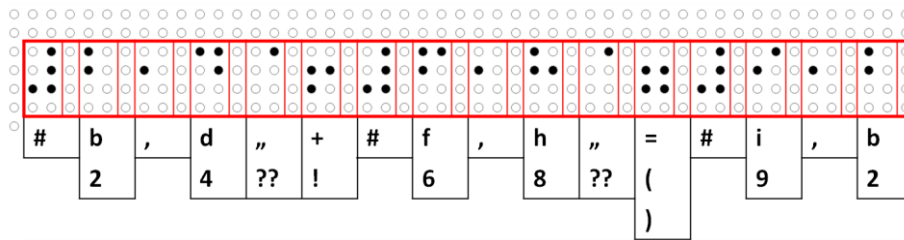


Abbildung 82 Entwurf eines Widgets zur Anzeige rückgewandelter Braille-Zeichen

Zu jedem Braille-Zeichen werden alle möglichen Umsetzungen in Schwarzschrift angezeigt. Braille-Zeichen, die sich nicht in einem Schwarzschriftzeichen ausdrücken lassen, wie etwa das An kündigungszeichen für mathematische Symbole (Punkt 4) können durch einen allgemeinen Ersatzausdruck gekennzeichnet werden (im Entwurf „??“). Durch Interaktion kann der Sehende dann weitere Informationen dazu erhalten. Das Widget ist einzeilig entworfen (also jeweils zur Anzeige einer Zeile), um eine leichte Zuordnung zwischen Braille-Zeichen und den verschiedenen Schwarzschriftvarianten zu ermöglichen.

Natürlich ist die Verwendung der Rückwandlung des ausgegebenen Stiftmusters in Schwarzschrift recht aufwändig. Schlimmsten Falls müsste der Sehende die gesamte Ausgabe Stift für Stift absuchen um den passenden Startpunkt für Texte zu finden. Der Zugang zu den tatsächlich dargestellten textuellen Elementen kann damit nur langsam geschehen, was die Kommunikation mit dem Blinden sehr einschränken würde. Sinnvoll ist daher eine Kombination der beiden Möglichkeiten. Zunächst sollten die ausgegebenen Texte direkt in Schwarzschrift dargestellt werden. Kommt es zu Kommunikationsproblemen, weil der Blinde andere Texte erkennt, so sollte der Sehende eine Schnittstelle zur Rückwandlung des ausgegebenen Stiftmusters in Schwarzschrift aktivieren können. Wird dem Sehenden die Position der Finger des Blinden angezeigt, so kann dieser leicht die Startposition der Rückwandlung festlegen und erhält somit schnellen Zugriff auf die vom Blinden erkannten Zeichen.

3.3 Umsetzungsbeispiele

Im Folgenden werden Implementierungen beschrieben, die im Rahmen dieser Arbeit entstanden und einige der zuvor genannten Konzepte umsetzen. Dies umfasst einen Editor zur Erstellung farbiger Rastergrafiken mit Hilfe der Braillezeile und grafisch-taktile Displays, ein Rendering-Framework zur einfachen Umsetzung von Darstellungen und Anwendungen auf grafisch-taktile Displays, welches unter anderem zur Umsetzung des HyperBraille-Darstellungskonzepts eingesetzt werden kann, eine prototypische Umsetzung des Vergrößerungskonzeptes auf Basis des HyperBraille-Darstellungskonzeptes sowie eine Umsetzung der partiellen Vergrößerung als Mozilla-Firefox-Extension.

Der Grafikeditor (Abschnitt 3.3.1) wurde als spezielle Anwendung für Blinde umgesetzt, da die Editier- und Darstellungskonzepte nicht sinnvoll in ein bestehendes Grafikerstellungsprogramm integriert werden konnten. Er setzt viele der in Abschnitt 3.2.3 vorgestellten Konzepte zum Umgang mit Farben um sowie das Konzept der Hot-Spots und Views.

Das Rendering-Framework stellt Komponenten zur einfachen Umsetzung grafisch-taktile Darstellungen inklusive Anzeige am Bildschirm bereit. Dazu wurde auch eine umfassende Bibliothek konfigurierbarer, taktile Widgets erstellt. Es integriert die bereits im Grafikeditor verwendeten Routinen zur Segmentierung von Rastergrafiken und Umwandlung in verschiedene Formate. Teile des Rendering-Frameworks wurden auch in den HyperReader-Magnifier, also die Umsetzung des Vergrößerungskonzeptes auf Basis von HyperBraille, übernommen. In beiden Fällen wird eine neue Rasterisierung des darzustellenden Inhalts auf Basis verschiedenster Informationen durchgeführt, sowie eine Aufbereitung der Bildschirmgrafik. Filterungen nach beliebigen, im Objekt-Modell des darzustellenden Inhalts oder der Bildschirmgrafik abgebildeten Informationen sind möglich. Die verschiedenen Ansichten des HyperBraille-Darstellungskonzeptes wurde mit den in Abschnitt 3.2.2.1 beschriebenen Layout-Möglichkeiten umgesetzt. In Überblicks- und Originalansicht wurden die verschiedenen Zoom-Strategien erprobt. Nutzern des Frameworks bzw. der damit erstellten Darstellungen stehen verschiedene Möglichkeiten zur Beeinflussung der Darstellung bereit.

Die Umsetzung der partiellen Vergrößerung als Firefox-Extension, genannt FirefoxZoom, nutzt die umfassenden Möglichkeiten zur Beeinflussung des Darstellungsprozesses von Firefox über CSS und DOM. Bei der Umsetzung zeigte sich die starke Abhängigkeit des Konzepts von der Strukturierung der grafischen Oberfläche. Verschiedene Konzepte zur Einbeziehung des Nutzers in die Bestimmung des zu vergrößernden Inhalts wurden entwickelt.

3.3.1 Ein Grafikeditor für Blinde zur Erstellung farbiger Grafiken

In der digitalen Welt werden viele Grafiken zur Veranschaulichung von Gedanken, Ideen und Zusammenhängen eingesetzt. Um Blinde vollständig in diese Welt zu integrieren, muss es ihnen möglich sein, selbst digitale Grafiken zu erzeugen. Dabei sollte der Erstellungsprozess möglichst einfach, schnell und intuitiv sein und den Einsatz von Farbe unterstützen.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Der Gedanke, dass Blinde selbst Grafiken erstellen können sollen, ist nicht neu. Es wurden schon verschiedene Techniken entwickelt mit deren Hilfe Blinde digitale Grafiken erzeugen können. Aber alle diese Techniken erfüllen nicht die genannten Anforderungen. Aus diesem Grund wurden im Rahmen dieser Arbeit neuartige Darstellungsformen entwickelt, die Blinden die Erstellung farbiger digitaler Grafiken mit Hilfe der Braillezeile auf intuitive Weise ermöglichen. Der dazu entworfene Editor kombiniert die Darstellungsformen und erleichtert die Erkundung der dargestellten Grafik. Durch Umwandlung der Darstellungsformen in Rastergrafiken und SVG wird die Bereitstellung der Grafiken an Sehende ermöglicht. Auch der Editor unterstützt die Zusammenarbeit mit Sehenden. Da Braillezeilen zumindest im deutschsprachigen Raum weit verbreitet sind, verursacht die Nutzung der neuen Grafikdarstellungen zudem keine zusätzlichen Anschaffungskosten für teure Geräte.

3.3.1.1 Stand der Forschung

Schon 1986 entwickelte Thomas Fehrle [Feh85] einen Zeichenarbeitsplatz für Blinde mit der Stuttgarter Stiftplatte als Ausgabegerät. Die Zeichnung erfolgte durch Eingabe von Zeichenbefehlen in natürlicher Sprache und war auf den Bereich geometrischer Formen beschränkt. Die gezeichneten Objekte werden benannt und können so auch verändert werden. Ziel war vor allem die selbständige Bearbeitung von Aufgabenstellungen aus der Geometrie und Analysis. Eine ähnliche Eingabesprache nutzt BPlot2 [FFO+08]. Hier können zusätzlich einfache Befehle der Grafikbearbeitung wie „rotate“ und Text verwendet werden. Die Ausgabe erfolgt bisher nur über einen grafikfähigen Brailledrucker und visuell zur Unterstützung der Zusammenarbeit mit Sehenden. Die Kopplung mit taktilen Displays ist geplant. Der Einsatz von Farbe wird nicht direkt unterstützt. Allerdings kann die für den Druck verwendete Punktstärke für jedes gezeichnete Objekt separat definiert werden, was der Verwendung von Graustufen ähnelt. BPlot2 wurde zur Erstellung von Grafiken in Blindenschriftbüchern entwickelt und ist derzeit das einzige Programm, das ein kooperatives Erstellen solcher Grafiken zwischen Blinden und Sehenden ermöglicht.

Martin Kurze präsentierte 1996 mit TDraw [Kur96] ein Programm, über das Blinde digitale Grafiken durch Zeichnen mit einem Thermostift auf Schwellpapier über einem digitalen Zeichenbrett erstellen konnten. Der Thermostift lässt das Schwellpapier an den berührten Stellen anschwellen. Gleichzeitig wurde die Zeichenbewegung an den Computer übertragen. Die erstellte Zeichnung konnte so von den blinden Benutzern direkt erfasst werden. Allerdings war es nicht möglich, einmal Gezeichnetes wieder zu löschen. Und auch die Änderung einer vor längerer Zeit erstellten Zeichnung war nur mit großem Aufwand möglich. Einfacher ist dies bei dem von Kobayashi und Watanabe entwickelten Mimizu [KW04], welches ein taktilles Display zur Ausgabe und einen digitalen Stift zur Eingabe nutzt. Die Grafik wird durch direktes Zeichnen mit dem Stift auf dem Display erzeugt. Durch Verwendung einer speziellen Taste kann der Eingabemodus von Malen auf Löschen geändert werden.



Abbildung 83 Darstellung des MIMIZU von Kobayashi und Watanabe [KW04]

Als Ausgabegerät dient ein Dot-View-Display der KGS Corporation, die Eingabe erfolgt über einen digitalen Stift mit Infrarotsender.

Eine verbreitete Möglichkeit zur Grafikerstellung durch Blinde ist die Verwendung von ASCII-Grafiken [Hel01]. Dabei werden normale Textzeichen genutzt, um das visuelle Erscheinungsbild von Grafiken nachzubauen. So können sie einfach in einem Texteditor erstellt werden und bedürfen weder besonderer Ein- noch Ausgabengeräte. Die Ausgabe erfolgt einfach über die Braillezeile, auf der die Grafik dann zeilenweise ertastet werden kann. Problematisch ist allerdings, dass die Punktanzahl der Braillezeichen in keinsten Weise mit der visuellen Darstellung der Zeichen übereinstimmt. Beispielsweise entspricht die Form eines Schrägstrichs, der zur Darstellung einer schrägen Linie genutzt wird, in Braille eher einer rechtwinkligen Ecke (wie Abbildung 84 veranschaulicht). Um die Grafik richtig zu interpretieren muss der Blinde also ständig die visuelle Repräsentation der Zeichen im Kopf haben.

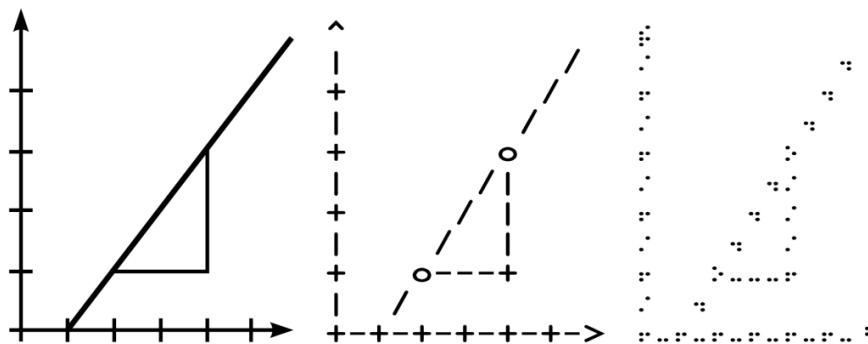


Abbildung 84 Beispiel für die Umsetzung einer Grafik für Blinde als ASCII-Grafik aus [Hel01]

links: Originalgrafik, Mitte: ASCII-Grafik, rechts: Darstellung der ASCII-Zeichen in Brailleschrift. Es wird deutlich, dass die Brailledarstellung die Linien der Originalgrafik nur schlecht repräsentiert.

Auch SVG bietet eine Möglichkeit zur Erstellung von Grafiken ohne besondere Hilfsmittel. Die Verwendung durch Blinde ist ähnlich zu BPlot [FFO+08]. Es stehen grafische Primitive und einfache Manipulationsfunktionen zur Verfügung. Ein direktes Malen ist nicht möglich und die erstellte Grafik lässt sich nur durch einen taktilen Ausdruck wirklich überprüfen. Allerdings wird der Einsatz von Farben direkt unterstützt und der Austausch mit Sehenden ist sehr einfach, weshalb SVG schon von einigen Blinden eingesetzt wird. Hinzukommt die Mög-

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

lichkeit, die Grafik umfassend zu annotieren und einzelne Objekte durch Gruppierungen zusammenzufassen. Dies hilft auch anderen Blinden, die Grafik zu erfassen.

Farbe ist ein wesentliches Konzept der Grafikdarstellung für Sehende. Farben werden nicht nur aus ästhetischen Gründen genutzt, sondern auch weil sie leichter zu unterscheiden und einprägsamer sind als z.B. verschiedene Graustufen oder Texturen. Sehende werden in Grafiken immer Farben verwenden und auch erwarten, da sie die Ausdrucksstärke der Grafik fördern und die Kommunikation über die Grafik erleichtern. Aus diesem Grund sind Farben selbst für von Geburt an blinde Personen wichtig. Werden sie in taktilen Grafiken nicht abgebildet, so stört dies die Kommunikation mit Sehenden. Können sie von Blinden nicht für die Erstellung von Grafiken genutzt werden, so stört dies die Integration der Blinden in die digitale Kommunikation. Außerdem äußern blinde Personen selbst den Wunsch, sich mit Farben ausdrücken zu können, vor allem solche, die Farben einmal visuell wahrnehmen konnten. Letztlich sollte die Vorstellungskraft des Menschen diesbezüglich auch nicht unterschätzt werden. Selbst von Geburt an blinde Menschen können einen inneren Eindruck von Farben haben, wie die Arbeiten des blinden Malers Eşref Armağan [Arm10] illustrieren. Abbildung 85 zeigt ein Werk von Eşref Armağan.

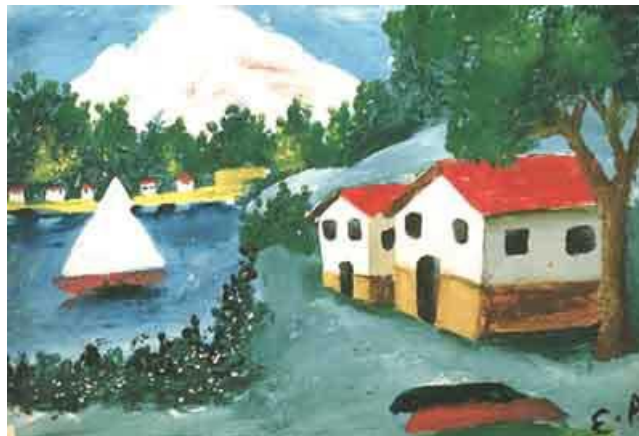


Abbildung 85 Ein Gemälde des von Geburt an blinden Malers Eşref Armağan

Aus diesen Gründen beschäftigen sich viele Arbeitsgruppen damit, wie blinden Menschen Farben präsentiert werden können. Häufig werden sie wie bei Screenreadern einfach verbalisiert, also per Sprache oder als Braille-Text ausgegeben. So ist es allerdings schwierig, einen umfassenden Überblick über die Grafik zu erhalten. Einfacher ist dies beim Einsatz von Texturen, wie sie bei taktilen Ausdrucken mit Brilledruckern und Schwellpapier verwendet werden. Eine weitere häufig verwendete Methode ist die Zuordnung zwischen Farben und bestimmten Materialien. Dies findet besonders häufig in Büchern für blinde Kinder Verwendung. Wird aber auch zur Darstellung von Kunstwerken eingesetzt (siehe Abbildung 86). Leider gibt es weder bei den Materialien noch bei den Texturen eine allgemein anerkannte Zuordnungsvorschrift oder gar Standardisierung. So kann es sein, dass die Farbrepräsentation mit jeder Grafik neu erlernt werden muss. Schlimmsten Falls kann es sogar zu Kommunikationsproblemen zwischen Blinden und Sehenden kommen, wenn eine einmal erlernte Zuordnung auf eine Grafik angewendet wird, zu der sie nicht passt.

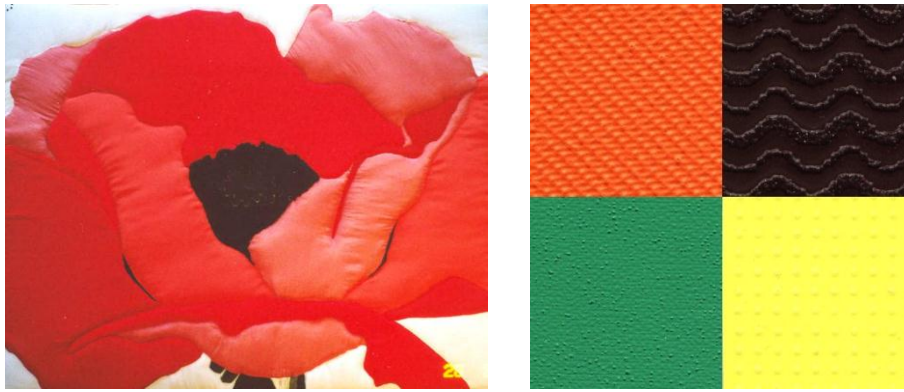


Abbildung 86 Ansätze zur Standardisierung der Farbdarstellung für Blinde
 links: Beispiel zum „The Barker Code of Color/Fabric Representation“ [Bar10]
 rechts: Vier Farbplatten mit Texturen aus dem „Tactile Colour“-System [Tac10]

„The Barker Code of Color/Fabric Representation“ von Sally Barker [Bar10] und das „Tactile Colour“-System der Tactile Colour Communication Society [Tac10] versuchen hier eine Lösung zu bringen. Der Barker-Code basiert auf der Verwendung verschiedener Stoffe auf verschieden weichem Untergrund. Der verwendete Stoff definiert den Farbton, der Untergrund die Helligkeit der Farbe. Dies erzeugt ein sehr angenehmes Tastgefühl, erfordert aber Fertigkeiten im Nähen. Die Erstellung der Bilder ist also sehr aufwändig und kann nur schwer von Blinden selbst durchgeführt werden. Eine Verbreitung des Barker Codes ist deshalb unwahrscheinlich. Hinzukommt, dass Sally Barker nur Empfehlungen für die zu verwendenden Stoffe gibt, sie aber selbst nicht anbieten kann, was einer wirklichen Standardisierung im Wege steht. Das „Tactile Colour“-System bietet hier bessere Chancen. Die zwölf für das System definierten Farben werden durch Oberflächenstrukturen repräsentiert. Diese kann man in Form selbstklebender Vinyl-Bögen käuflich erwerben. Abbildung 86 zeigt vier davon. So können Blinde und Sehende gleichermaßen immer sichergehen, das richtige Material für die richtige Farbe einzusetzen. Die Vinyl-Bögen lassen sich einfach mit der Schere bearbeiten und können so auch von Kindern genutzt werden. Zudem können die Oberflächenstrukturen zur taktilen Ausgabe digitaler Grafiken auch direkt an die passenden Stellen geprägt werden. Das „Tactile Colour“-System hat dadurch gute Chancen eine weitere Verbreitung zu finden. Allerdings lässt es sich nur schwer auf ein taktilen Display oder gar eine Braillezeile abbilden. Die Bearbeitung digitaler, farbiger Grafiken kann dadurch also nicht unterstützt werden.

3.3.1.2 Grafik-Formate für die Braillezeile

Die Braillezeile ist ein verbreitetes Arbeitsmittel unter Blinden. Deshalb ist es sinnvoll, Möglichkeiten zu schaffen, die die Braillezeile zur Erstellung digitaler Grafiken nutzen. Wie in Abschnitt 3.3.1.1 dargestellt, sind aktuelle Techniken dafür nicht nutzbar. Außerdem bieten aktuelle Techniken Blinden nicht immer die Möglichkeit, digitale Grafiken selbst zu erstellen und einfach zu überprüfen.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Im Rahmen dieser Arbeit wurden zwei Datenformate entwickelt, die Blinden die Erstellung digitaler, farbiger Grafiken mit Hilfe der Braillezeile ermöglichen und dabei die folgenden Anforderungen erfüllen:

- Intuitiver und schneller Eingabeprozess
- Eingabe sollte der grafischen Ausgabe möglichst ähnlich sein
- Grafische Ausgabe soll durch den Blinden leicht prüfbar sein
- Ausgabe für den Blinden und Sehenden sollen möglichst ähnlich sein
- Einmal erstellte Grafiken sollen leicht zu ändern sein
- Ausgabe sollte für Sehende leicht verständlich sein
- Farbe soll unterstützt werden

Die Datenformate wurden bezeichnet mit „Einfaches Figur-Braille“ (EFB) und „Detailliertes Figur-Braille“ (DFB). Dabei dient EFB zur monochromen Darstellung von Grafiken und DFB zur Darstellung farbiger Grafiken mit Hilfe eines neuentwickelten Farbcodes.

Ein drittes Datenformat, genannt „Objekt-Figur-Braille“ (OFB), unterstützt zudem die Zugänglichkeit der erstellten Grafiken und die Integration von Text. Es ist allerdings nur mit dem ebenfalls im Rahmen dieser Arbeit entwickelten Editor wirklich gut zu handhaben.

3.3.1.2.1 Einfaches Figur-Braille (EFB) – Monochrome Grafik auf der Braillezeile

Einfaches Figur-Braille (EFB) könnte man als die elektronische Form der Grafiken bezeichnen, die mit Blindenschriftschreibmaschinen erzeugt werden. Dabei werden die Punkte der Braillezeichen einfach als Bildpunkte (Pixel) einer Rastergrafik interpretiert. Im einfachsten Fall steht dabei ein gehobener Stift für einen schwarzen Pixel und ein gesenkter Stift für einen weißen. Per Eingabe über die Brailletastatur kann so intuitiv eine monochrome Rastergrafik durch Setzen der einzelnen Pixel erstellt werden. Über die Brailletastatur kann jeder Punkt eines Braillezeichens separat per Tastendruck gesetzt werden. Zur Eingabe eines Braillezeichens, das aus mehreren Punkten besteht, werden mehrere Tasten gleichzeitig gedrückt. Natürlich kann die Eingabe auch über eine normale Schwarzschriftastatur geschehen. Dabei muss der Benutzer allerdings im Kopf die Zuordnung zwischen Schwarzschriftzeichen und Punktmuster herstellen.

Aus technischen Gründen wird für EFB nur 6-Punkt-Schrift genutzt, obwohl Braillezeilen 8-Punkt-Schrift unterstützen. Die Umwandlung von Schwarzschrift in 8-Punkt-Braille ist allerdings sprachabhängig und unterscheidet sich deshalb je nach Einstellung der Systemsprache bzw. der Sprache des Braillezeilentreibers. Die Zuordnung zwischen 6-Punkt-Braille und Schwarzschrift ist zumindest für westeuropäische Sprachen sprachunabhängig umsetzbar (siehe <http://www.braille.ch/eb-allg.htm>). In EFB werden Rastergrafiken also aus Zeichen zusammengesetzt, die jeweils den Zustand von 2×3 Pixeln repräsentieren. Tabelle 7 zeigt die für einfaches Figur-Braille verwendete Codierung.

Punkte	...456							
	...000	...100	...010	...110	...001	...101	...011	...111
000...	leer	"	!	>	'	\$	<	_
100...	a	c	e	d	1	3	5	4
010...	,	i	:	j	?	9	/	w
110...	b	f	h	g	2	6	8	7
001...	.		*	`	-	0)	#
101...	k	m	o	n	u	x	z	y
011...	;	s	+	t	(~	=	}
111...	l	p	r	q	v	&	{	%

Tabelle 7 Übersicht zur EFB-Codierung.

Die Matrix enthält die ersten drei Punkte eines Braille-Zeichens (linke Spalte eines Braille-Zeichens) in den Zeilen und die letzten drei (rechte Spalte) in den Spalten.

EFB kann nicht nur zur Erstellung rein monochromer Grafiken genutzt werden, sondern auch beim Umgang mit farbigen Grafiken behilflich sein. In diesem Fall wird die EFB-Darstellung als Strukturansicht zum Überblick über die Grafik genutzt. Welche Pixel durch gehobene Stifte repräsentiert werden und welche durch gesenkte wird über einen Farbfilter definiert. Im einfachsten Fall werden alle Pixel, die nicht weiß sind, in gehobene Stifte umgesetzt. Weiß wird häufig als Hintergrundfarbe in Grafiken eingesetzt. Somit zeigt diese Umsetzung den gesamten Inhalt der Grafik an. Tabelle 8 zeigt drei Beispiele dazu. Diese indirekte Darstellung von Farbe ist natürlich nur mit Unterstützung durch einen Editor möglich, über den passende Farbfilter ausgewählt werden können. Abschnitt 3.3.1.3 stellt einen solchen Editor vor, der im Rahmen dieser Arbeit entwickelt wurde.

Bei der Arbeit mit EFB müssen zwei Dinge beachtet werden. Einerseits hat die Braillezeile gegenüber dem Bildschirm eine sehr geringe Auflösung und kann auch nur eine geringe Anzahl von Pixeln gleichzeitig anzeigen. Andererseits sind die Stifte der Braillezeile nicht wie Pixel dicht aneinander, sondern besitzen einen deutlich wahrnehmbaren Abstand zueinander. Dieser variiert zudem noch, da zwischen den Zellen für die einzelnen Braillezeichen jeweils ein etwas größerer Abstand besteht als zwischen den Stiften eines einzelnen Braillezeichens. Die Brailleschrift-Darstellung in Tabelle 8 verdeutlicht dies. Diese beiden Aspekte haben Einfluss auf die Größe der Grafiken, die zur Darstellung auf der Braillezeile sinnvoll ist. Aufgrund der geringen Pixelanzahl bei größeren Grafiken ist ein hoher Scrollaufwand nötig, um die Grafik ganz zu erfassen. Wählt man allerdings nur sehr kleine Grafiken, so kann es aufgrund der geringen Auflösung leicht zu Missverständnissen kommen. Beispielsweise sieht ein sehr kleiner Kreis genauso aus wie ein Achteck, wie Tabelle 8 rechts zeigt. Selbst bei der Darstellung von Dreiecken bevorzugten Probanden die größere Darstellung in Tabelle 8 Mitte gegenüber der kleinen einzeiligen Darstellung in Tabelle 8 links. Dies begründet sich darin, dass bei größeren Darstellungen der zusätzliche Abstand zwischen den Braillezeichen leichter kompensiert werden kann. Allgemein zeigte sich bei Nutzertests mit der EFB-Darstellung,

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

dass die Verzerrung aufgrund des zusätzlichen Abstands zwischen den Braillezeichen anfangs Probleme macht. Diese konnten aber von den Probanden recht schnell überwunden werden.




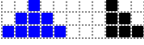
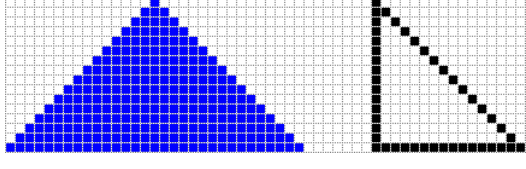
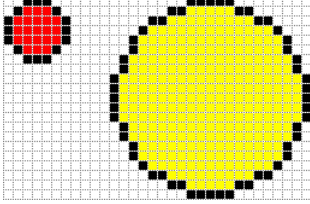

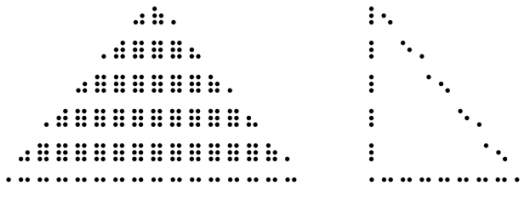
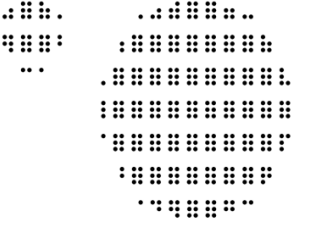
		
		
<pre>) { . { .</pre>	<pre>) { . _ ? ' } % % % (_ e .) % % % % % { . _ " ? ' } % % % % % % % (_ e .) % % % % % % % % % { . _ " ? "cccccccccccccccc "ccccccca</pre>	<pre>) % { . ') } % % = - 7 % % b < % % % % % % { ca ' % % % % % % % % v _ % % % % % % % % % " % % % % % % % % p > % % % % % % q "d7 % % g c</pre>
		

Tabelle 8 Grafiken mit zugehöriger EFB-Darstellung in Schwarzschrift und Brailleschrift
 Zeile 1 zeigt die Grafiken in Originalgröße (mit 11 × 3, 54 × 16 bzw. 32 × 21 Pixeln).
 Zeile 2 zeigt vergrößerte Darstellungen mit einem Raster zur Markierung der einzelnen Pixel.
 Zeile 3 zeigt die zugehörige EFB-Darstellung in Schwarzschrift.
 Zeile 4 zeigt die EFB-Darstellung in Brailleschrift mit den dabei entstehenden Leerräumen. Zu beachten ist, dass auf der Braillezeile gleichzeitig nur eine der Grafik-Zeilen aus Tabellenzeile 4 angezeigt werden kann.

Die EFB-Darstellung wurde in größerem Umfang bereits erfolgreich im HyperBraille-Projekt eingesetzt. Sie diente zur Entwicklung von Darstellungen für das grafisch-taktile Display. Dabei arbeiteten blinde und sehende Projektpartner Hand in Hand. Hierbei zeigte sich der Vorteil der pixelgenauen Darstellungsmöglichkeit deutlich. Da nur so eine exakte Definition der taktilen Darstellung möglich ist. Die blinden Partner nutzten zur Entwicklung der Darstellungen tatsächlich lange Zeit die Braillezeile. Das grafisch-taktile Display war zu Beginn des Projekts noch nicht verfügbar. Trotzdem wurden erfolgreich auch größere Darstellungen von den Blinden selbst erzeugt. Abbildung 87 zeigt ein Beispiel dazu.

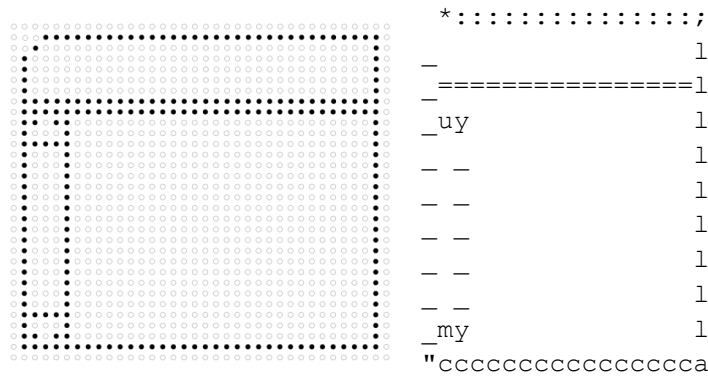


Abbildung 87 Beispiel eines Widget-Entwurfs aus dem Projekt HyperBraille, der mit dem EFB-Format gemeinsam von blinden und sehenden Projektpartnern erstellt wurde (links in der Ansicht für Sehende, rechts in EFB-Darstellung)

Die EFB-Darstellung wurde später auch für grafisch-taktile Displays umgesetzt. Dabei wurde kein Abstand zwischen Braillezeichen oder Zeilen gelassen, so dass die Grafiken wie in Abbildung 87 links dargestellt zwar mit Abstand zwischen den Pixeln aber unverzerrt erscheinen.

Wie bereits erwähnt kann die EFB-Darstellung mit Hilfe eines passenden Editors und Farbfilerung auch zur indirekten Darstellung von Farben verwendet werden. Allerdings ist es so natürlich schwierig, einen Überblick über die Farbverteilung zu erhalten bzw. zu kontrollieren, ob alle Bildpunkte die gewünschte Farbe haben. Außerdem ist mit EFB alleine (ohne zusätzlichen Editor) die Erstellung farbiger Grafiken nicht möglich. Und auch mit Editor könnte die Erstellung nur schrittweise Farbe für Farbe erfolgen, was eher umständlich ist. Deshalb wurde im Rahmen dieser Arbeit eine Farbcodierung für die Erstellung farbiger Rastergrafiken mit Hilfe der Braillezeile entwickelt. Diese wird im folgenden Abschnitt beschrieben.

3.3.1.2.2 Detailliertes Figur-Braille (DFB) – Ein Farbcode für die Braillezeile

Mit detailliertem Figur-Braille (DFB) ist die pixelgenaue Erstellung farbiger Rastergrafiken mit Hilfe der Braillezeile möglich. Dabei erfüllt die Darstellung der Farbinformationen folgende im Rahmen dieser Arbeit definierten Anforderungen:

- Die Zuordnung zwischen Bildpunkt und Farbinformation sollte intuitiv sein.
- Die Farbinformation sollte möglichst kompakt dargestellt werden, um möglichst viele Bildpunkte gleichzeitig darstellen zu können und eine schnelle Eingabe zu erlauben.
- Die Zuordnung zwischen der Darstellung und der zugehörigen Farbe sollte möglichst einfach sein.
- Die Farbdarstellung sollte gleichermaßen für Braillezeilen und für grafisch-taktile Displays verwendbar sein.

Natürlich kann die Farbinformation an sich nicht in einem einzelnen Stift repräsentiert werden. Die Zuordnung „ein Stift gleich ein Pixel“ wie bei EFB ist also nicht möglich. Da die Zuordnung bei DFB aber ähnlich intuitiv sein sollte, wurde hier „ein Braillezeichen gleich ein

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Pixel“ gewählt. Dadurch vergrößert sich natürlich entsprechend die Darstellung der Grafik auf der Braillezeile und die Anzahl der gleichzeitig sichtbaren Pixel verringert sich. In der DFB-Darstellung kann nur noch eine Pixelzeile gleichzeitig auf der Braillezeile angezeigt werden. Eine kompaktere Darstellung ist bei gleichem Informationsgehalt allerdings kaum möglich. Eine denkbare Alternative wäre die Zuordnung der vier oberen Stifte eines Braillezeichens zu einem Pixel und der vier unteren zu dem darunterliegenden. Allerdings bestehen, wie schon bei EFB beschrieben, Probleme bei der Übersetzung von Schwarzschrift in 8-Punkt-Brailleschrift unter Ausnutzung aller möglichen Stiftkombinationen. Somit ist diese Alternative nicht sinnvoll realisierbar. Bei der Verwendung von 6-Punkt-Schrift wäre es auch möglich, jede Spalte eines Braillezeichens einem Pixel zuzuordnen. Somit ständen pro Pixel drei Braillezeichen-Punkte zur Verfügung. Da allerdings einer dieser Punkte als Orientierungspunkt dienen müsste, wie weiter unten beschrieben, blieben nur noch zwei für die tatsächliche Farbcodierung. Es wären also neben Schwarz und Weiß nur noch zwei weitere Farben möglich. Dies ist nicht ausreichend. Bei Verwendung aller sechs Punkte können trotz Orientierungspunkt noch 32 Farben abgebildet werden. Hinzukommt, dass bei der Verwendung des gesamten Braillezeichens eine gute Abgrenzung für jeden Pixel vorhanden ist. Hier ist der größere Abstand zwischen den Braillezeichen auf der Braillezeile sogar von Vorteil.

Um die Codierung möglichst einprägsam zu gestalten wurde sie entsprechend der subtraktiv Farbmischung aufgebaut, die schon aus der Grundschule bekannt ist. Passend zur EFB-Codierung bedeutet ein leeres Braillezeichen (alle Stifte abgesenkt) Weiß und ein Vollzeichen (alle Stifte gehoben) Schwarz. Die Grundfarben sind Rot, Gelb und Blau. Den meisten Menschen ist diese Farbmischung vertrauter als das aus der Computerwelt gewohnte additive Mischen mit Rot, Grün und Blau. Die Grundfarben sind jeweils einem Braillezeichen-Punkt zugeordnet. Zur leichteren Erkennung sind sie in einer Spalte von oben (Rot) nach unten (Blau) angeordnet. Die Mischfarben Orange, Violett, Grün und Braun werden durch heben aller Stifte der zugehörigen Grundfarben dargestellt. Dabei gilt:

- Orange = Rot + Gelb → Stifte für Rot und Gelb angehoben
- Violett = Rot + Blau → Stifte für Rot und Blau angehoben
- Grün = Gelb + Blau → Stifte für Gelb und Blau angehoben
- Braun = Rot + Grün = Rot + Gelb + Blau → Stifte aller Grundfarben angehoben

Um den DFB-Code neben der Braillezeile auch auf einem grafisch-taktilen Display nutzen zu können, ist zusätzlich zu den Farbstiften ein Orientierungsstift notwendig. Sonst kann es für den Benutzer sehr aufwändig werden, die Farbe richtig zu bestimmen. Auf Braillezeilen wird jeweils nur eine Zeile von Farbcodes gleichzeitig angezeigt. Die Braillezeilen-Module sind für das schnelle Lesen dieser einen Zeile optimiert und bieten dem Finger durch ihre Form Orientierungshilfen. So kann der Lesende schnell erkennen, welcher Stift gehoben ist, auch wenn er der einzige Gehobene in diesem Zeichen ist. Auf grafisch-taktilen Displays fehlen diese Orientierungshilfen, da sie auf eine flächige Darstellung ausgelegt sind und Text potentiell an einer beliebigen Stelle auf dem Display stehen kann. So muss der Leser selbst ent-

scheiden, ob er beispielsweise ein „c“ liest (Punkt 1 und 4) oder ein „:“ (Punkt 2 und 5). Abbildung 88 zeigt die Nummerierung der Punkte eines Braillezeichens.

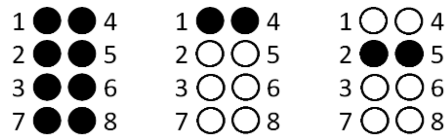


Abbildung 88 Nummerierung der Punkte eines Braillezeichens sowie Darstellung der Zeichen „c“ (Mitte) und „:“ (rechts) in Brailleschrift

Bei Wörtern oder längeren Texten ergibt sich dies meist schnell aus dem Kontext. Eine zusätzliche Orientierung ist hier hilfreich aber nicht unbedingt nötig. Bei den Zeichen des Farbcodes ist eine Erschließung aus dem Kontext aber nicht möglich. Ein rot gefülltes Rechteck wird bis auf eine Verschiebung um eine Stiftzeile genauso dargestellt wie ein gelbes. Eine rot-gelb gepunktete Linie könnte auch eine gelb-blau gepunktete Linie sein. Abbildung 89 illustriert dies. Wenn nicht gerade ein schwarzer Pixel (Vollzeichen) oder ein brauner Pixel (eine ganze Spalte) an ein solches Rechteck oder eine solche Linie angrenzt, gibt die Grafik selbst dem Benutzer keine Hinweise, um welche Farbe es sich wirklich handelt. Aus diesem Grund ist eine Orientierungshilfe im Farbcode notwendig. Für die DFB-Codierung wurde der Punkt 1 des Braillezeichens (oben links) gewählt, der bei jeder Farbe außer Weiß mit angezeigt wird. Abbildung 89 zeigt, dass die Farben so eindeutig unterscheidbar sind.

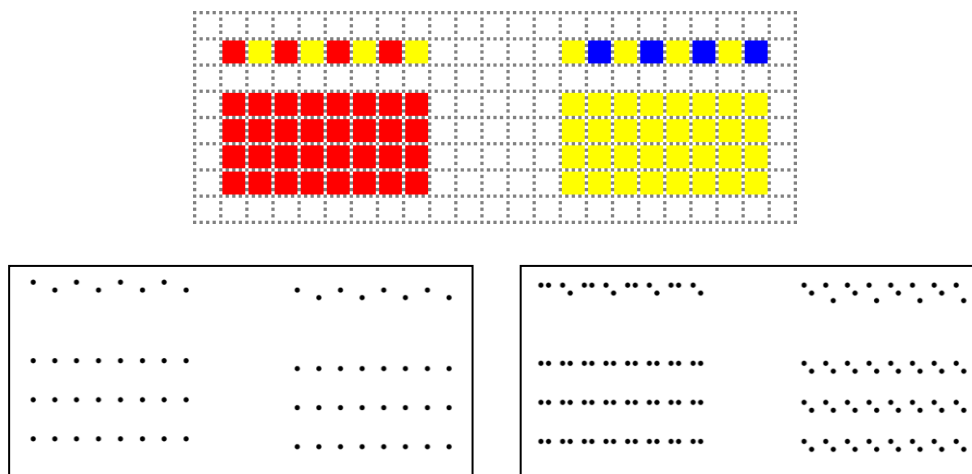


Abbildung 89 Beispiel zu DFB-Code ohne Orientierungspunkt (links unten) und mit (rechts unten) Rot wird jeweils durch Punkt 4 des Braillezeichens repräsentiert, Gelb durch Punkt 5 und Blau durch Punkt 6. Punkt 1 ist der Orientierungspunkt. Es ist deutlich zu erkennen, dass bei flächiger Darstellung ein Orientierungspunkt nötig ist, um die Farben eindeutig zu identifizieren.

Tabelle 9 zeigt eine Übersicht zur DFB-Codierung. Um die Grundfarben bei Verwendung des Orientierungspunktes in einer Spalte halten zu können, werden sie in der zweiten Spalte des Braillezeichens abgebildet (Punkte 4, 5 und 6). Die nun noch zur Verfügung stehenden Punkte 2 und 3 werden genutzt, um Helligkeitswerte abzubilden. Punkt 2 steht für eine Aufhellung der Farbe, Punkt 3 für eine Abdunkelung. In Analogie dazu stehen die Punkte 2 und 3 allein mit Orientierungspunkt für Hellgrau bzw. Dunkelgrau. Die Mischung aus Hellgrau und

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Dunkelgrau ergibt ein normales Grau (Punkte 1, 2 und 3). Damit wurde jedem der 6 Punkte eines 6-Punkt-Braillezeichens eine eindeutige Bedeutung zugewiesen. Inklusive Weiß und Schwarz sind 26 Farben darstellbar.

Name	Punkte	Name	Punkte	Name	Punkte	Name	Punkte
Weiß	keine	Orange	145	Hellgrau	12	Hell-...	+2
Rot	14	Violett	146	Dunkelgrau	13	Dunkel-...	+3
Gelb	15	Grün	156	Grau	123		
Blau	16	Braun	1456	Schwarz	alle		

Tabelle 9 Übersicht über die DFB-Codierung mit Angabe der Grundfarben in RGB

Neben der Festlegung der Codierung in Braillezeichen benötigt die DFB-Darstellung natürlich auch eine Definition der visuellen Darstellung der codierten Farben. Damit sich Sehende und Blinde ohne Missverständnisse miteinander über die Farben unterhalten können, wurden Farbwerte definiert, die gut unterscheidbar sind und gut zu den Farbnamen des DFB-Codes passen. Diese Farben werden bei der Konvertierung der DFB-Darstellung in eine normale Rastergrafik angewendet. Im umgekehrten Fall können natürlich auch andere Farben auftreten. In diesem Fall werden die Originalfarben der Rastergrafik über die in Abschnitt 3.2.3.2 diskutierten Metrik CIEDE2000 [SWD05] auf die nächstliegenden DFB-Farben abgebildet.

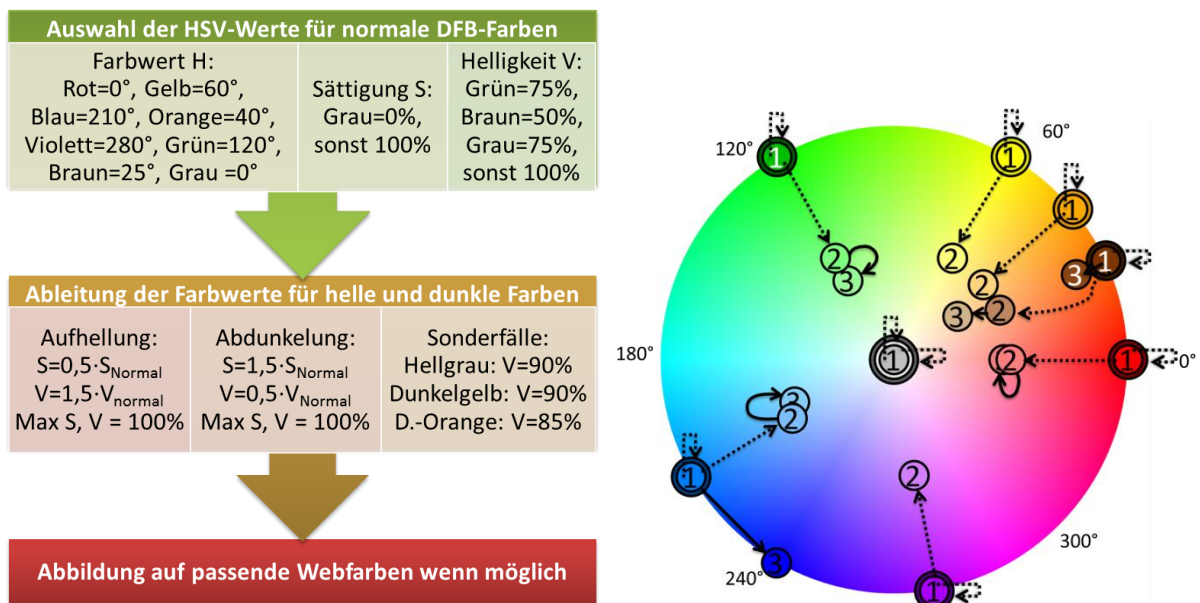


Abbildung 90 Darstellung des Prozesses zur Definition der Farbwerte für die DFB-Farben

Die Auswahl der Farbwerte erfolgte in einem mehrstufigen Prozess (wie Abbildung 90 illustriert). Zunächst wurden für die DFB-Farben mit normaler Helligkeit passende Werte aus dem HSV-Farbraum gewählt [Smi78]. Im HSV-Raum ist die Auswahl gleichmäßig heller und gesättigter Farben besonders einfach. Anschließend wurden diese für die hellen Farben durch Absenkung der Sättigung aufgehellt und für die dunklen Farben durch Senkung der Helligkeit abgedunkelt. Zum Schluss wurde versucht, die entstandenen Farbwerte passenden Webfarben zuzuordnen [Wik10b]. Dadurch wird die Verwendung der Farben in unterschied-

lichen Umgebungen einfacher und die Kommunikation über die Farben wird unterstützt. Leider ließen sich nicht zu allen DFB-Farben geeignete Webfarben finden. Tabelle 10 zeigt das Resultat dieses Prozesses.

Bei der Auswahl der HSV-Werte wurden zunächst für Rot, Orange, Gelb und Grün die Standardwerte 0° , 30° , 60° und 120° gewählt. Braun wurde bei 25° angeordnet, da es etwas rötlicher ist als Orange. Für Violett wurde 280° gewählt, um es deutlich von Magenta abzugrenzen und so Kommunikationsprobleme zu vermeiden. Blau wurde zur Abgrenzung von Violett zu 210° zugeordnet. Bei der Auswahl der passenden Webfarbe zeigte sich allerdings, dass der Standardwert von 240° für normales und dunkles Blau gut eignet. Damit ist das Farbspektrum relativ gleichmäßig abgedeckt, wobei zugunsten von Orange und Braun zwei größere Lücken im Bereich Pink bzw. Magenta und Cyan bestehen. Die mit „1“ markierten Punkte in Abbildung 90 rechts illustrieren dies. Die Abbildung zeigt den Prozess der Farbdefinition in einer Draufsicht auf den HSV-Raum. Die Helligkeitsdimension wurde zugunsten der Übersichtlichkeit weggelassen. Für alle normalen Farben außer Grau, Braun und Grün wurden volle Sättigung (100%) und voller Farbwert (100%) gewählt. Grau ist per Definition völlig ungesättigt. Die Grauwerte liegen direkt auf der Mittelachse des HSV-Kegels zwischen Schwarz und Weiß, haben also eine Sättigung von 0%. Braun ist aus sich heraus eine eher dunkle Farbe und wurde deshalb von Anfang an etwas abgedunkelt auf $V=50\%$. Ein volles Grün ($RGB = 0, 255, 0$) erscheint an den meisten Bildschirmen sehr leuchtend. Ein natürliches Grün ist aber eher matt. Deshalb wurde Grün im Farbwert auf 75% reduziert. Dies brachte auch eine deutliche Annäherung zwischen Bildschirm- und Druckausgabe.

Bei der Auswahl von Farben muss beachtet werden, dass diese je nach Bildschirmeinstellungen sehr unterschiedlich dargestellt werden können. Sollen Bilder gedruckt werden, ist auch die Druckdarstellung zu beachten, die nicht selten von der Bildschirmdarstellung abweicht. Im Allgemeinen kann man nicht davon ausgehen, dass Bildschirme und Drucker korrekt kalibriert sind. Da der Farbcode zur Erstellung digitaler Grafiken dienen soll, wurde mehr Wert auf die Darstellung am Bildschirm gelegt als auf die Druckfarbe. Vor allem im Bereich Blau und Violett zeigten sich bei leichten Änderungen der Farbwerte größere Abweichungen. Um die Eignung der gewählten Farben für eine unmissverständliche Bildschirmwiedergabe zu prüfen, wurde eine Online-Umfrage durchgeführt. Diese zeigte, dass die gewählten Farben zum großen Teil wie beabsichtigt interpretiert wurden. Lediglich bei Hell- und Dunkelorange, sowie Dunkelgelb zeigten sich deutliche Probleme. Hier sollte eine weitere Studie durchgeführt werden, bei der die Nutzer eine Farbe zum angegebenen Farbnamen wählen. Anhang A stellt den Fragebogen zur Online-Umfrage vor und gibt eine Übersicht zu den Ergebnissen.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

	Hell	Normal	Dunkel		
Rot	0°, 50%, 100% 255, 127, 127	0°, 100%, 100% 255, 0, 0	0°, 100%, 50% 127, 0, 0	HSV	Original
	LightCoral 239, 127, 127	Red	DarkRed 140, 0, 0	Name	Webabbild
	0°, 47%, 94%		0°, 100%, 55%	RGB	HSV
Gelb	60°, 50%, 100% 255, 255, 127	60°, 100%, 100% 255, 255, 0	60°, 100%, 90% 229, 229, 0	HSV	Original
	#FFFF7F	Yellow	#E5E500	Name	Webabbild
				RGB	HSV
Blau	210°, 50%, 100% 127, 191, 255	210°, 100%, 100% 0, 127, 255	210°, 100%, 50% 0, 63, 127	HSV	Original
	LightSkyBlue 134, 205, 249	Blue 0, 0, 255	Navy 0, 0, 127	Name	Webabbild
	203°, 46%, 98%	240°, 100%, 100%	240°, 100%, 50%	RGB	HSV
Orange	40°, 50%, 100% 255, 212, 127	40°, 100%, 100% 255, 170, 0	40°, 100%, 85% 216, 144, 0	HSV	Original
	#FFD47F	Orange 255, 165, 0	#D89000	Name	Webabbild
		39°, 100%, 100%		RGB	HSV
Violett	280°, 50%, 100% 212, 127, 255	280°, 100%, 100% 169, 0, 255	280°, 100%, 50% 84, 0, 127	HSV	Original
	#D47FFF	#A900FF	#54007F	Name	Webabbild
				RGB	HSV
Grün	120°, 50%, 100% 127, 255, 127	120°, 100%, 75% 0, 191, 0	120°, 100%, 38% 0, 96, 0	HSV	Original
	PaleGreen 152, 249, 152	#00BF00	DarkGreen 0, 99, 0	Name	Webabbild
	120°, 39%, 98%		120°, 100%, 39%	RGB	HSV
Braun	25°, 50%, 75% 191, 135, 95	25°, 100%, 50% 127, 53, 0	25°, 100%, 25% 63, 26, 0	HSV	Original
	Tan 209, 179, 140	SaddleBrown 140, 69, 19	#3F1A00	Name	Webabbild
	34°, 33%, 82%	25°, 86%, 55%		RGB	HSV
Grau	0°, 0%, 90% 229, 229, 229	0°, 0%, 75% 191, 191, 191	0°, 0%, 38% 96, 96, 96	HSV	Original
	Gainsboro 219, 219, 219	Silver	DimGray 104, 104, 104	Name	Webabbild
	0°, 0%, 86%		0°, 0%, 41%	RGB	HSV

Tabelle 10 Darstellung der DFB-Farben mit der zugehörigen Webfarbe, sowie RGB- und HSV-Werten
Entspricht die Farbe keiner Webfarbe, so ist der RGB-Hex-Code angegeben. Die RGB-Werte verstehen sich in einer Skala von 0-255.

3.3.1.2.3 Objekt-Figur-Braille (OFB) – Text und Annotationen für Figur-Braille

EFB und DFB wurden ursprünglich als reine Grafikformate entwickelt. Die Integration von textuellen Elementen wurde zunächst nicht beachtet. Die Verwendung von Text muss aber gewährleistet sein, wenn dem Benutzer ein vollwertiges Werkzeug zur Erstellung von Grafiken, wie Diagrammen und ähnlichem, zur Verfügung stehen soll. Bei der Einbeziehung von Text ergaben sich allerdings zwei Probleme. Natürlich sollen die Texte für den Blinden lesbar in Braille dargestellt werden. Beachtet man dabei sowohl Braillezeilen, wie auch grafisch-taktile Displays, so sind unterschiedliche Darstellungen nötig. Auf grafisch-taktilen Displays sollten die Braillezeichen mit einer Stiftspalte Abstand angezeigt werden, auf Braillezeilen darf dieser Abstand nicht verwendet werden, da sich sonst eine Verschiebung der Braillezeichen zu den Braillemodulen der Zeile ergibt. Das zweite Problem entsteht durch die Verwendung des 6-Punkt-Codes. Soll Text in 8-Punkt-Schrift dargestellt werden, so müssen die Braillezeichen in zwei EFB-Zeilen aufgespalten werden. Das gleiche Problem tritt auf, wenn zwar nur 6-Punkt-Schrift verwendet wird, der Text aber nicht an den EFB-Zeilen ausgerichtet ist. Aus diesen Gründen wurde das objektbasierte Format „Objekt-Figur-Braille“ (OFB) entwickelt. OFB ist ein XML-basiertes Format, welches verschiedene Layer für Texte und Grafiken enthält. Beides wird in textueller Form in einer XML-Datei gespeichert. Dabei werden Grafiken in DFB abgelegt. Listing 8 zeigt ein Beispiel dazu. OFB-Dateien sind somit auch für Blinde ohne ein spezielles Werkzeug les- und änderbar. Allerdings ist es durch die getrennte Darstellung natürlich schwierig, die Gesamtgrafik zu erfassen. In OFB können jeweils mehrere Text- und Grafik-Layer gespeichert werden. Sie können benannt werden und sind so leicht identifizierbar. Außerdem bietet die Verwendung einzelner Objekte die Möglichkeit, die Grafik schrittweise zu erfassen und auch die Änderung einer Grafik wird dadurch erleichtert. Die Attribute „freispalten“ und „pixelzeilen“ in Text-Layern geben an, für welche Abstände zwischen Braillezeichen bzw. Textzeilen der Text und die Grafik darum entworfen wurden. Die Attribute „maxBreite“ und „maxHoehe“ dienen zum automatischen Umbrechen und zur Begrenzung von Text.

```

<ofb>
  <graphic X="0" Y="0" title="Bunter Rahmen"
           height="7" width="16" id="0"><![CDATA[
ccc%111
c      1
c      1
%      %
5      d
5      d
555%ddd
]]></graphic>
  <text X="2" Y="2" title="einfacher Text" freispalten="1"
        pixelzeilen="0" maxBreite="0" maxHoehe="0" id="1"><![CDATA[
text
]]></text>
</ofb>

```

Listing 8 Beispiel für OFB-Dateien (OFB = Objekt-Figur-Braille)

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Neben Text- und DFB-Elementen könnten in OFB auch einfache Grafik-Objekte wie Linien, Rechtecke und Kreise integriert werden. Durch solche Ergänzungen nähert sich OFB natürlich an bestehende objekt-basierte Formate wie SVG an. Eine Untersuchung zeigte allerdings, dass eine Integration mit SVG nur schlecht möglich ist, da die Behandlung von Umrandungen in SVG bezüglich einer Pixel-genauen Grafikerstellung, wie es mit DFB geschieht und für die Darstellung auf der Braillezeile nötig ist, wenig intuitiv ist. Rahmen werden in SVG immer zur Hälfte außerhalb der definierten Objektgrenzen gezeichnet. Möchte der Benutzer die Rahmenstärke eines Objektes ändern, ohne die Ausdehnung zu beeinflussen, so muss er also auch Höhe und Breite des Objektes neu bestimmen. Dies ist einem Benutzer nur schwer zu vermitteln. Deshalb wurde OFB nicht wie in [TE09] vorgeschlagen in SVG integriert. Eine Umwandlung zwischen OFB und SVG kann aber natürlich vorgenommen werden.

3.3.1.3 Implementierung eines Grafikeditors

Im Rahmen dieser Arbeit wurde ein Grafikeditor entwickelt, um die Nutzbarkeit der vorgestellten Datenformate zu demonstrieren. Der Editor wurde als Windows-Anwendung in C# erstellt, um gute Kompatibilität mit den gängigen Screenreadern zu gewährleisten.

Die erste Version des Grafikeditors (genannt HBGraphicsExchange, siehe Abbildung 91) wurde ursprünglich entwickelt, um den blinden und sehenden Projektpartnern das gemeinsame Erstellen und Bewerten von Designs für die verschiedenen Ansichten des HyperBraille-Darstellungskonzepts zu ermöglichen. Die erstellten Grafiken wurden größtenteils für Evaluationen mit Papier-Prototypen verwendet, wozu sie mit einem Index-Drucker in äquidistanter Darstellung ausgegeben wurden. Die grafisch-taktilen Displays standen zu Beginn des Projekts noch nicht zur Verfügung. Später wurde natürlich auch eine direkte Ausgabe auf das BrailleDis 9000 umgesetzt. HBGraphicsExchange stellte EFB und einen Vorgänger von DFB in einfachen Textfeldern dar, die mit Hilfe von Screenreadern gut über die Braillezeile erfasst und verändert werden konnten. Der Vorgänger von DFB war eine einfache 1/0-Darstellung mit ‚1‘ für einen gehobenen Stift und ‚0‘ für einen gesenkten. Sehende erstellten die Grafiken in der sogenannten Schwellpapieransicht, in der jeder Stift als kleiner Kreis dargestellt wird (gehobene Stifte schwarz, gesenkte weiß mit grauem Rand). Diese wurde ursprünglich entwickelt, um die erstellten Grafiken auch für Papier-Prototypen-Tests mit Schwellpapier zu verwenden. Deshalb entspricht das Verhältnis von Kreis- bzw. Stiftgröße zu -abstand ursprünglich dem des BrailleDis 9000 und den damit vergleichbaren Displays. Dadurch eignet sich diese Darstellung sehr gut, um Sehenden einen Eindruck von den Größenverhältnissen zu vermitteln und wurde deshalb auch in folgenden Versionen beibehalten. Bei Betrachtung einfacher Rastergrafiken (im Editor als Pixeldarstellung bezeichnet), waren Sehende oft geneigt zu große Grafiken zu erstellen bzw. die Erkennbarkeit kleiner Grafiken anzuzweifeln. Die Schwellpapieransicht ist auch für Sehbehinderte gut erkennbar. Zum leichteren Editieren kann die Schwellpapieransicht auch leicht abgewandelt werden, so dass der Durchmesser der Kreise die gesamte für einen Stift zur Verfügung stehenden Breite einnimmt. Die Kreise wurden allerdings erhalten und nicht durch Quadrate ersetzt, um eine gut sichtbare Trennung zwischen den Stiften beizubehalten.

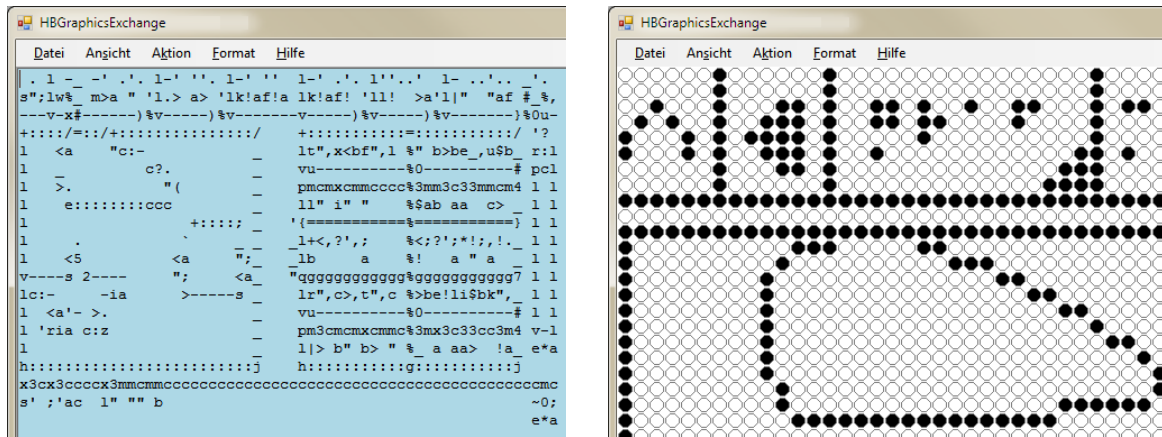


Abbildung 91 Ansichten des HBGraphicsExchange (erste Version des Grafikeditors)
links: EFB-Ansicht, rechts: Ausschnitt der zugehörigen Schwellpapieransicht im Editier-Modus

In HBGraphicsExchange wurden die verschiedenen Ansichten jeweils separat dargestellt. Es war also zu jeder Zeit immer nur eine Ansicht sichtbar und editierbar. Damit war natürlich die direkte Zusammenarbeit zwischen Blinden und Sehenden behindert. Auch Sehbehinderte, die sowohl die taktile, wie auch eine gute visuelle Ausgabe nutzen wollen, wurden damit nicht optimal unterstützt, da die Ausgabe auf die Braillezeile in der Schwellpapieransicht nicht unterstützt wurde. Die folgende Version zeigte deshalb alle Ansichten parallel (siehe Abbildung 92 oben), wobei die Pixeldarstellung entfiel, da sie nur zum Austausch der Grafiken, nicht aber zur Ansicht, genutzt wurde. EFB und DFB wurden dabei ebenfalls in großen Textfeldern angezeigt, die nebeneinander positioniert. Dabei zeigten sich aber zwei Probleme. Zum einem wurde für den Sehenden die eingeschränkte Sichtbarkeit auf der Braillezeile nicht deutlich. Zum anderen ergaben sich technische Probleme mit der Intelligenz der Screenreader. In verschiedenen Anzeigemodi wurden EFB- und DFB-Zeilen, die auf einer Höhe waren, nebeneinander angezeigt, was im Kontext des Grafikeditors nicht sinnvoll ist. Es mussten also besondere Einstellungen getroffen werden, um dies zu umgehen. Der Editor sollte aber möglichst mit jedem Screenreader ohne besondere Einstellungen leicht bedienbar sein. Aus diesen Gründen werden in der dritten Version EFB und DFB jeweils in einzeiligen Textfeldern angezeigt, die sich über die gesamte Breite der GUI erstrecken.

Die dritte Version des Grafikeditors (Abbildung 92 unten) unterstützt als erste OFB. Die Liste der Elemente ist links in der GUI angeordnet. Über diese können die einzelnen Objekte editiert, gelöscht, ein- und ausgeblendet und in der Grafik direkt angesprungen werden. EFB- und DFB-Textfeld werden dabei synchronisiert, so dass beim Wechsel zwischen den Textfeldern per Tabulator-Taste immer die gleiche Position im Bild angezeigt wird. Die Objekte werden nicht direkt in der Gesamtansicht editiert, sondern in separaten Oberflächen, die jeweils nur das Objekt zeigen. Dies bedeutet natürlich, dass der Benutzer während der Änderung eines Objektes keinen Zugriff auf den Kontext des Objektes hat. Es sollte untersucht werden, ob auch eine Lösung für ein Editieren in der Gesamtansicht gefunden werden kann. Zumindest die Positionierung der Objekte sollte in der Gesamtansicht erfolgen können, da hier der Kontext besonders interessant ist. Die einzelnen Layer können bereits in der aktuellen Implementierung selektiert werden. Mit Hilfe der Navigationstasten könnte in einem

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

gesonderten „Positionierungsmodus“ die Verschiebung durchgeführt werden. Die Darstellung auf der Braillezeile muss der Verschiebung natürlich folgen können.

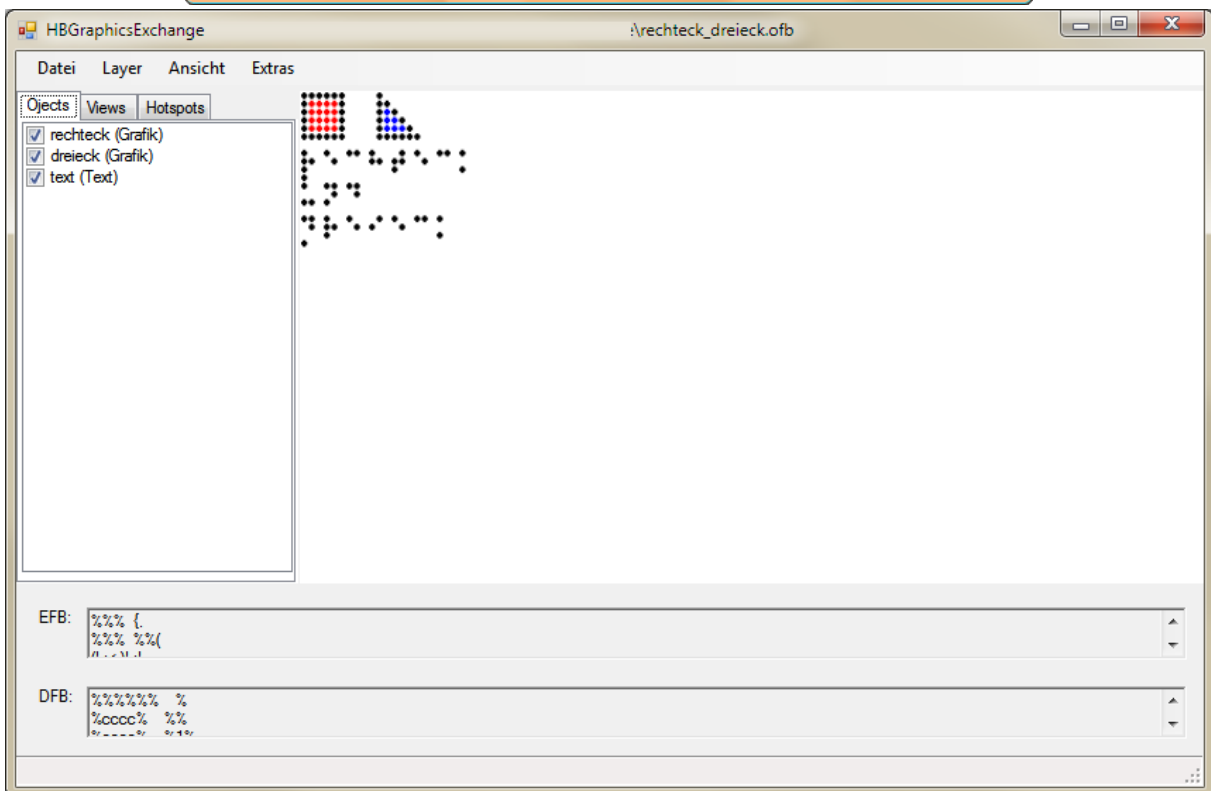
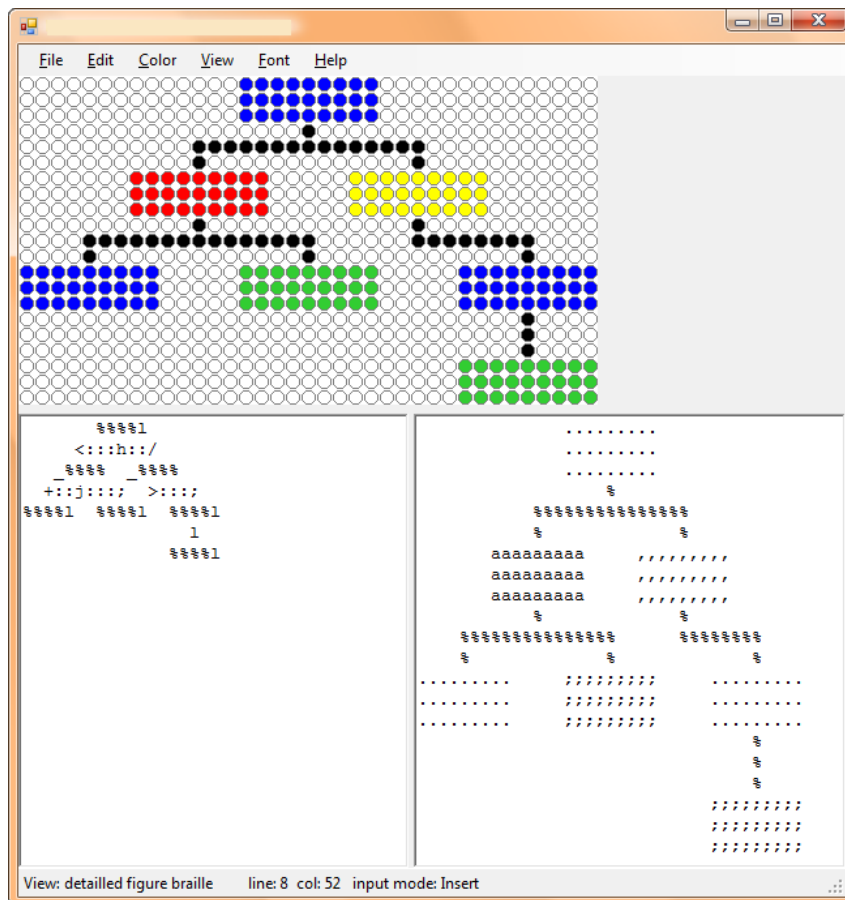


Abbildung 92 Screenshots von HBGraphicsExchange Version 2 (oben) und 3 (unten)

Auf die Gesamtdarstellung und auch einzelne Grafikobjekte kann ein Farbfilter angewendet werden. Dieser wird über die Farbcodes des DFB gesteuert. Zudem kann in eine Konturdarstellung geschaltet werden und für grafisch-taktile Displays sind die in Abschnitt 3.2.3.3 vorgeschlagenen kombinierten Darstellungen aus Konturen und Farbcodes verfügbar. Für Sehende werden diese in einem separaten Fenster dargestellt. Editieren ist in diesen Ansichten, sowohl für Blinde wie auch Sehende, nicht möglich. Das in Abschnitt 3.2.2.6 beschriebene Konzept von Hotspots und Views wurde ebenfalls umgesetzt. Zu einer Datei angelegte Hotspots und Views werden in der aktuellen Implementierung in einer separaten Datei abgelegt, die bei Öffnen der Datei im Editor mit geladen wird. So kann jeder Benutzer leicht eigene Hotspots und Views definieren. Allerdings entstehen so auch zusätzliche Dateien, die eventuell verloren gehen können.

Eine besondere Herausforderung bei der Implementierung des Grafikeditors war die Umsetzung des Editierens in EFB. Da EFB eine kompaktere Darstellung bietet als DFB sollte das Editieren möglich sein. Allerdings sollen dabei Farbinformationen erhalten bleiben und auch Farben verwendet werden können. Aus technischer Sicht bedeutet dies, dass alle Eingaben des Benutzers genau überwacht werden müssen, um auch Selektionen und damit eingeleitetes Löschen oder Ersetzen ganzer Bereiche der Grafik zu überwachen. Aus konzeptioneller Sicht mussten verschiedene Editiermodi beachtet werden. Üblicherweise existieren beim Eingeben von Zeichen zwei Editiermodi – das Erweitern des Textes durch Einfügen der eingegebenen Zeichen (sog. Einfüge-Modus) und das Überschreiben bestehender Zeichen mit den eingegebenen Zeichen (sog. Überschreiben-Modus). Beim Editieren in EFB existiert noch ein weiterer Modus. Es sollte auf einfache Weise möglich sein, den durch ein EFB-Zeichen repräsentierten sechs Pixeln verschiedene Farben zuzuweisen. Dazu muss ein EFB-Zeichen ergänzt werden können, in dem weitere farbige Punkte zu den bestehenden Punkten hinzugefügt werden können (siehe Abbildung 93). Nur so kann beispielsweise innerhalb einer EFB-Zeile eine blaue Linie direkt unter eine rote gezeichnet werden. Dabei müssen die neu eingegebenen Punkte alle alten Punkte überlagern, um jegliche Ergänzung zu ermöglichen.

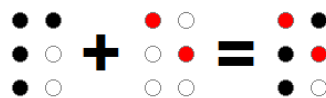


Abbildung 93 Beispiel zur Ergänzung eines EFB-Zeichens
schwarzes p + rotes e = gemischtes q mit Punkt 2,3 und 4 in Schwarz und Punkt 1 und 5 in Rot

Glücklicherweise ist der Überschreiben-Modus zum Editieren nicht unbedingt nötig. Er kann genauso gut durch Eingabe der neuen Zeichen nach vorherigem Entfernen oder Markieren der alten Zeichen nachgestellt werden. Somit müssen nur zwei Editiermodi unterstützt werden – das Einfügen und das Ergänzen von EFB-Zeichen. Das Ergänzen tritt an Stelle des Überschreiben-Modus. Zwischen beiden Modi kann wie gewohnt mit der Einfügen-Taste umgeschaltet werden. Zum farbigen Editieren in EFB muss natürlich auch eine Malfarbe gewählt werden können. Dies geschieht im Editor über ein Menü, dessen Einträgen die DFB-Farbcodes als Accesskeys zugewiesen wurden.

3.3.1.4 Bereitstellung der erzeugten Grafiken für Andere

Ziel der Erstellung von Grafiken ist natürlich meist, diese auch anderen bereitzustellen, um bestimmte Sachverhalte in anderer Form als Text darzustellen und eventuell darüber zu diskutieren. Zum Austausch zwischen Blinden genügen dabei die Formate EFB, DFB und OFB. Zum Austausch mit Sehenden sollte aber eine Konvertierung in gängige Grafikformate durchgeführt werden. So können die erstellten Grafiken auch leichter als Grafik kenntlich in Dokumente integriert werden. Dies kommt auch Blinden zugute, da so leichter zu erkennen ist, wo es sich um eine Grafik handelt und wo um normalen Text, als wenn die Figur-Braille-Darstellungen direkt in Dokumente integriert werden. Natürlich sollten bei der Umwandlung die Alternativdarstellungen zu Grafiken in Dokumenten oder den Grafiken selbst genutzt werden. Die Figur-Braille-Formate können hier integriert werden. Sie sollten aber markiert sein, sodass beispielsweise bei der Verwendung im Alternativtext die Sprachausgabe eines Screenreaders gestoppt werden kann, wenn sie zu einer Figur-Braille-Darstellung gelangt. Diese vorzulesen wäre nicht hilfreich.

Die Umwandlung kann je nach Zielsetzung und Format auf unterschiedliche Weise erfolgen. Für Visualisierungen von Darstellungen für grafisch-taktile Displays (wie beispielsweise in Abbildung 87 links) eignet sich besonders die für Schwellpapier entwickelte Darstellung, die bereits im vorigen Abschnitt beschrieben wurde. Sie ist allerdings nur nützlich, wenn absichtlich gezeigt werden soll, dass dem Blinden keine kontinuierliche Darstellung zur Verfügung steht. In diesem Fall sollte dann auch auf den Einsatz von Farbe verzichtet werden. Normalerweise ist dies aber nicht gewünscht. Die einfachste Form, eine kontinuierliche Darstellung zu erstellen, ist die Rückübersetzung in die ursprüngliche Idee von Figur-Braille, also die Umsetzung jedes DFB-Zeichen in ein Pixel. Dabei entstehen allerdings potentiell sehr kleine Grafiken, die lediglich als Austauschformat dienen können. Dabei ist zu beachten, dass die Farbinformationen bei der Speicherung nicht durch Komprimierungsalgorithmen verfälscht werden. Zur Darstellung für Sehende müssen diese Grafiken vergrößert werden. Eine Vergrößerung auf das 8- bis 10-fache über ein Nearest-Neighbor-Verfahren hat sich dabei bewährt (siehe Abbildung 94 links). Bilineare Interpolation sollte bei der Vergrößerung nicht verwendet werden, da dies zu einer sehr verschwommenen Darstellung führt, die keinerlei Vorteile bringt, wie Abbildung 94 Mitte zeigt. Kontinuierlichere Darstellungen lassen sich mit sogenannten Pixel-Art-Skalierungsalgorithmen erreichen, wie beispielsweise den Hqx-Algorithmen von Maxim Stepin [Wik10a] (Abbildung 94 rechts).



Abbildung 94 Grafik mit zwei Kreisen (wie in Tabelle 8 rechts) in Originalgröße und in 9-facher Vergrößerung mit verschiedenen Algorithmen
v.l.n.r.: Nearest-Neighbor, Bilineare Interpolation, 2-fache Anwendung von Hq3x

In Webseiten bietet es sich auch an, ein zoombares Format zu verwenden, so dass der Betrachter selbst über die Vergrößerungsstärke entscheiden kann. Dies kann durch Einbindung der unvergrößerten Rastergrafik in SVG über das `<image>`-Tag erfolgen. Die Nearest-Neighbor-Skalierung kann durch Setzen des Attributs `image-rendering` auf den Wert `optimizeSpeed` aktiviert werden. Allerdings sind dabei zur Anzeige immer zwei Dateien nötig, die SVG-Datei und die referenzierte Rastergrafik. Um dies zu vermeiden, kann die DFB-Darstellung auch in eine reine SVG-Darstellung gewandelt werden. Am sinnvollsten ist es dabei, alle Pixel bzw. Zeichen einer Farbe in einem `path`-Element zusammenzufassen, wobei es sich nicht um einen verbundenen Pfad handelt, sondern lediglich um eine Sammlung einzelner Punkte. Die ursprüngliche EFB- und DFB-Darstellung kann als eine Art Alternativdarstellung durch `CDATA`-Objekte eingebunden werden, um auch Blinden einen schnellen Zugriff auf die Darstellung zu ermöglichen. Diese werden von SVG-Renderern nicht interpretiert und können beliebige Zeichen, außer der Kombination „`]] >`“, und damit alle Zeichen des EFB-Codes enthalten. Bei OFB bietet sich die Umwandlung in SVG natürlich unmittelbar an, um die Informationen über die einzelnen Layer und damit die Explorierbarkeit erhalten zu können. Text-Layer können dabei in SVG-Text-Elemente umgesetzt und so in Schwarzschrift dargestellt werden. Für Grafik-Layer kann die oben beschriebene Umwandlung angewendet werden. Zusätzlich können die EFB- und DFB-Darstellung für die Gesamtgrafik eingebunden werden. Nur so kann auch Blinden ein Zugang zur Gesamtdarstellung ermöglicht werden. Listing 9 zeigt die SVG-Darstellung der OFB-Datei aus Listing 8. Abbildung 95 zeigt die zugehörigen Darstellungen im Grafikeditor und im Webbrowser.

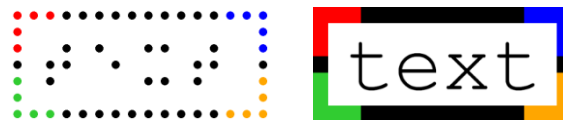


Abbildung 95 *Beispieldarstellungen zu OFB und SVG zu Listing 8 bzw. Listing 9
links: Darstellung für Sehende aus dem Grafikeditor
rechts: zugehörige SVG- Darstellung aus einem Webbrowser*

Wie aus Listing 9 leicht erkenntlich ist, kann die SVG-Datei auch leicht in OFB zurückübersetzt werden. Es ist also nicht nötig beide Dateien bereitzustellen. Die SVG-Datei genügt als Austauschformat. Allerdings ist es nur schlecht möglich, die Darstellung direkt in der SVG-Datei zu verändern. Es erfordert viel Vorstellungsvermögen, die gespeicherten Pfade korrekt zu verändern. Auch für Sehende ist diese Darstellung in einem SVG-Editor nicht gut bearbeitbar. Zudem müssten diese auch die eventuell integrierten EFB- und DFB-Darstellungen anpassen. Die SVG-Austausch-Darstellung ist also kein Ersatz für OFB.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"[
<!ENTITY E2 'M 3 0 4 0 M 4 0 5 0 M 5 0 6 0 M 6 0 7 0 M 7 0 8 0
      M 8 0 9 0 M 9 0 10 0 M 10 0 11 0 M 11 0 12 0 M 12 0 13 0 M 0 3 1 3
      M 15 3 16 3 M 3 6 4 6 M 4 6 5 6 M 5 6 6 6 M 6 6 7 6 M 7 6 8 6
      M 8 6 9 6 M 9 6 10 6 M 10 6 11 6 M 11 6 12 6 M 12 6 13 6 '>
<!ENTITY E4 'M 0 4 1 4 M 0 5 1 5 M 0 6 1 6 M 1 6 2 6 M 2 6 3 6 '>
<!ENTITY E1 'M 0 0 1 0 M 1 0 2 0 M 2 0 3 0 M 0 1 1 1 M 0 2 1 2 '>
<!ENTITY E5 'M 15 4 16 4 M 15 5 16 5 M 13 6 14 6 M 14 6 15 6
      M 15 6 16 6 '>
<!ENTITY E3 'M 13 0 14 0 M 14 0 15 0 M 15 0 16 0 M 15 1 16 1
      M 15 2 16 2 '>
<!ENTITY E6 'font-family:Courier New;font-size:50px;'>]
<svg xmlns="http://www.w3.org/2000/svg" height="70" width="160">
  <g transform="translate(0,5)"><![CDATA[efb:
p33cxcm4
lf a:>a_
ccccccc
]]><![CDATA[dfb:
ccc%l11
c 1
c % % % % % 1
% % % % % %
5 % % % % d
5 d
555%ddd
]]>
  <g transform="scale(10,10)"><title>Bunter Rahmen</title>
  <path d="&E1;" stroke="#FF0000"/><path d="&E2;" stroke="#000000"/>
  <path d="&E3;" stroke="#0000FF"/><path d="&E4;" stroke="#31CC31"/>
  <path d="&E5;" stroke="#FFA600"/>
  <![CDATA[efb:
pcccccc4
l
ccccccc
]]><![CDATA[dfb:
ccc%l11
c 1
c 1
% %
5 d
5 d
555%ddd
]]>
  </g><g transform="translate(20,45)"><title>einfacher Text</title>
  <text style="&E6;">text</text></g>
</g>
</svg>
```

Listing 9 SVG-Darstellung zur OFB-Datei aus Listing 8

3.3.1.5 Fazit

Erster Nutzerreaktionen auf den Grafikeditor und die Erfahrungen aus dem HyperBraille-Projekt zeigen, dass es tatsächlich möglich ist, Grafiken auf diese Weise mit Hilfe der Braillezeile zu erstellen. Die Nutzer schätzen die Möglichkeit den Verlauf von Linien besser als in ASCII-Grafik erkennen zu können und Sehenden direkt gewöhnliche Rastergrafiken anbieten zu können, über deren Aussehen sie auch gut informiert waren. Die meisten Nutzer waren sehr interessiert daran, auch farbige Grafiken erstellen zu können. Alle haben den verwendeten Farbcode schnell erlernt, wobei einige anmerkten, dass sie sich die Farben an sich nicht vorstellen können, da sie nie Farben wahrnehmen konnten.

Der entwickelte Farbcode wurde neben dem Grafikeditor auf dem grafisch-taktilen Display auch für das mit farbigen Feldern arbeitende Spiel BubbleBreaker eingesetzt, eine Form des SameGame (siehe <http://de.wikipedia.org/wiki/SameGame>). Da die Spieloberfläche gleichzeitig taktil und visuell dargestellt wurde, konnten Sehende und Blinde gemeinsam spielen. Die Kommunikation konnte mit Hilfe der Farben gut geführt werden. Auf diese Weise wurde der Farbcode von den Blinden besonders schnell aufgenommen.

3.3.2 Rendering-Framework für grafisch-taktile Displays

Als Teil dieser Arbeit wurde im Rahmen des Projekts HyperBraille ein Rendering-Framework für grafisch-taktile Displays entwickelt. Das Framework besteht aus mehreren Komponenten, die auch unabhängig vom HyperReader, der im Rahmen des Projektes entwickelten Gesamtsoftware, verwendet werden können.

Ziel des Rendering-Frameworks war die Unterstützung des Darstellungsprozesses im HyperReader. Dieser startet wie in Abbildung 96 dargestellt bei gewöhnlichen Desktopanwendungen für Microsoft Windows, wie etwa PowerPoint, Notepad, Solitär und Excel. Diese Anwendungen werden von sogenannten Filtern nach ihrer Struktur und ihrem Inhalt analysiert. Dabei werden je nach Anwendung verschiedene Schnittstellen, wie beispielsweise UIAutomation oder Browser-Helper-Objects (BHO) genutzt. Die Ergebnisse der Filter werden in einem zentral verwalteten Off-Screen-Model (OSM) abgelegt [KVV08]. Aus diesem wird letztlich die grafisch-taktile Darstellung erzeugt. Im Rahmen von HyperBraille wurden zu demonstrativen Zwecken Filter für verschiedene Anwendungen entwickelt. So auch ein auf JavaScript, XUL und C++ basierender Firefox-Filter, dessen Entwicklung im Rahmen dieser Arbeit betreut wurde [Gao08].

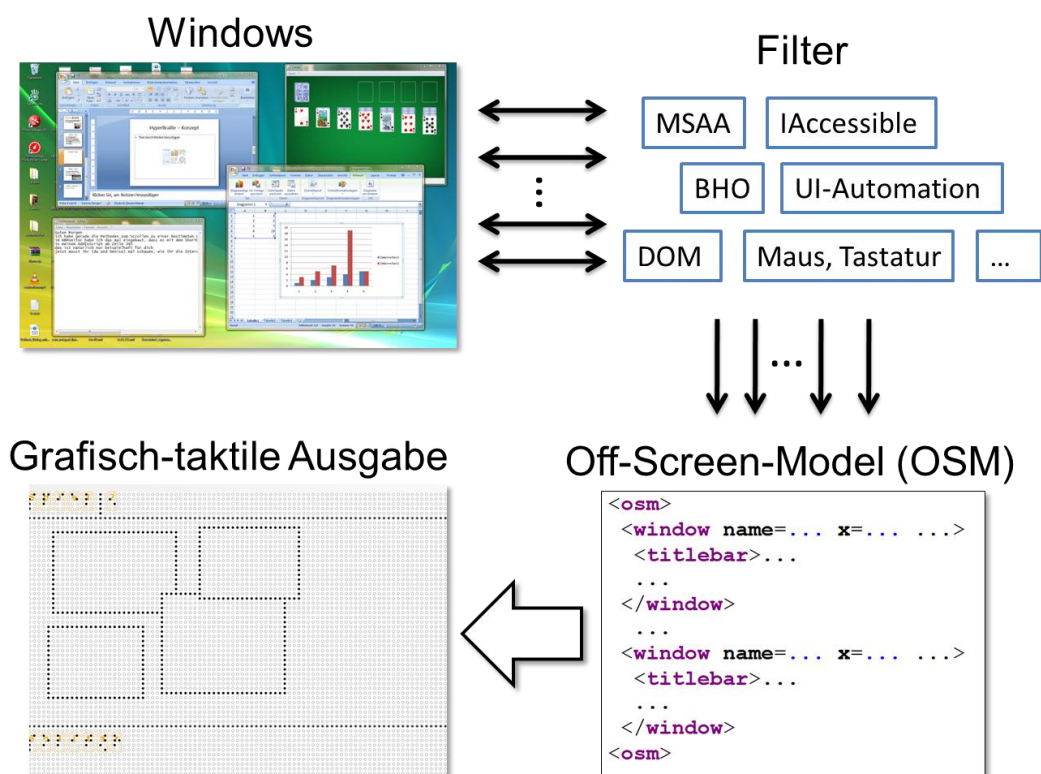


Abbildung 96 Grobansicht des Darstellungsprozesses im HyperReader
 Struktur und Inhalt von Windows-Desktop-Anwendungen werden von Filtern (basierend auf MSAAA, UIAutomation und ähnlichem) analysiert und in einem zentralen Off-Screen-Model (OSM) [KVV08] abgelegt, aus welchem die grafisch-taktile Darstellung erzeugt wird

Im Folgenden werden die wesentlichen Aspekte der Implementierung dargestellt.

3.3.2.1 Aufbau des Frameworks

Für die Umsetzung der in Abschnitt 3.2.4.1 vorgestellten Einteilung der Ausgabefläche des grafisch-taktilen Displays in unterschiedliche Bereiche und Anwendung unterschiedlicher Ansichtsarten wurde ein dreistufiges Renderingkonzept aufgebaut (siehe Abbildung 97).

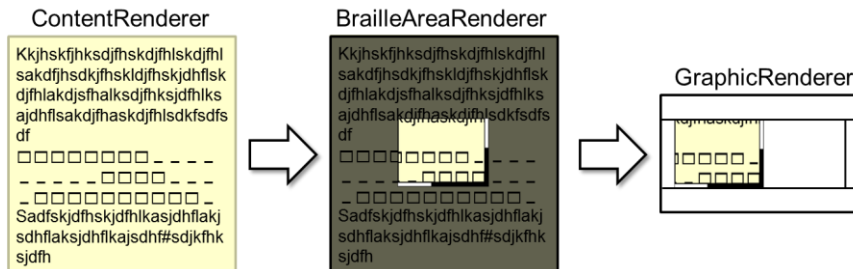


Abbildung 97 Dreistufiger Rendering-Prozess für den HyperReader

Der zum Rendering ausgewählte Inhalt des OSM wird zunächst in einem ContentRenderer in der gewählten Ansicht gerendert (so vollständig wie nötig, um die Ausgabegröße zu ermitteln). Aus der erzeugten Ausgabe wird durch einen BrailleAreaRenderer der Teil ausgeschnitten, der dem Benutzer in dem zur Ausgabe bestimmten Bereich angezeigt werden kann. Scrollbalken werden der Ausgabe hinzugefügt, falls nötig. Alternativ wird durch den BrailleAreaRenderer, wenn gewünscht, eine Übersichtsdarstellung (Minimap-Darstellung) erzeugt, wie in Abschnitt 3.1.2.14 beschrieben. Die Ergebnisse aller BrailleAreaRenderer werden letztlich vom GraphicRenderer zur Gesamtausgabe zusammengesetzt. Die Dreiteilung ermöglicht eine hohe Wiederverwendbarkeit und die Aufteilung des Renderings in mehrere parallele Prozesse. Für andere Komponenten des HyperReaders steht der GraphicRenderer als zentrale Schnittstelle bereit. Hierüber wird der gesamte Rendering-Prozess angestoßen. Änderungen der Darstellung aufgrund von Interaktionen wie Scrollen und Zoomen können über den GraphicRenderer initiiert werden. Auch zur Ermittlung der an bestimmten Stellen der Ausgabe dargestellten Inhalte, was zur Auswertung von Gesten und damit auch zur Steuerung der grafischen Oberfläche nötig ist, müssen andere Komponenten nur mit dem GraphicRenderer kommunizieren. BrailleAreaRenderer und ContentRenderer können über entsprechende Schnittstellen dynamisch mit dem GraphicRenderer verbunden werden. So kann der GraphicRenderer leicht auch andere Darstellungskonzepte unterstützen. Auch die Größe der vom GraphicRenderer genutzten Ausgabefläche ist veränderbar, da sie bei Initialisierung über Parameter festgesetzt wird. Die Größe und Anordnung der Darstellungsbereiche wird erst bei Ausführung des Renderings bestimmt, wobei auch Überlagerungen zwischen den Bereichen möglich sind.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

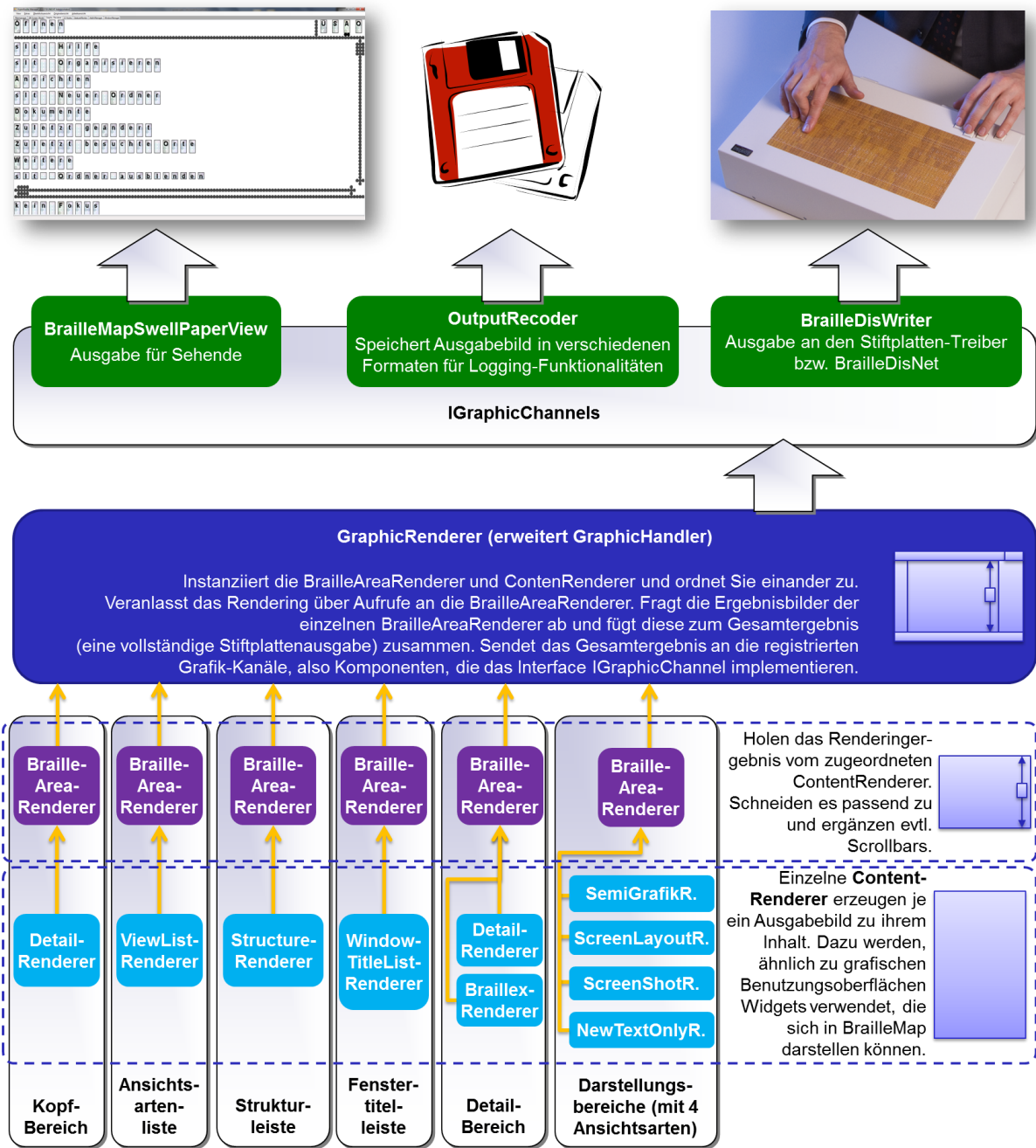


Abbildung 98 Grobübersicht über den Darstellungsprozess und die Komponenten des grafisch-taktilen Renderers

In CamelCase und weißer Schrift gehaltene Bezeichner sind die Namen der im Rahmen dieser Arbeit implementierten Klassen und Schnittstellen. BrailleDisNet ist eine durch den Projektpartner Metec in C# implementierte Kapselung des Stiftplattentreibers.

Wie Abbildung 98 zeigt, stellt der GraphicRenderer seine Ausgabe nicht direkt auf dem grafisch-taktilen Display dar. Er speichert sie lediglich in dem in Abschnitt 3.3.2.4 beschriebenen Ausgabeformat „BrailleMap“ und liefert sie an verschiedene Grafikkanäle, die dynamisch mit dem GraphicRenderer verbunden werden können. Hierzu erweitert der GraphicRenderer die eigenständige Klasse GraphicHandler (siehe Abbildung 99), die zusammen mit einigen Klassen, die Grafikkanäle implementieren, in einem separaten Paket gruppiert wurde. Der GraphicHandler stellt einen Grafik-Kontext bereit, mit dessen Hilfe Zeichenfunktionen im Brail-

leMap-Format ausgeführt werden können. Über einen Refresh-Befehl kann der GraphicHandler veranlasst werden, das zum Grafik-Kontext gehörende Bitmap an die registrierten Grafikkanäle zu schicken, sodass diese die Darstellung vornehmen können. Als Grafikkanäle wurden im Rahmen dieser Arbeit drei Klassen implementiert (Erzeugung der Darstellung auf grafisch-taktilen Displays, Ausgabe für Sehende am Monitor und Speicherung der Ausgabe in Bilddateien). Der GraphicHandler behandelt zudem zentral die durch Blink-Informationen im BrailleMap nötigen Änderungen der Ausgabe. So ist sichergestellt, dass jeder Grafikkanal die gleiche Blinkfrequenz wiedergibt. Der im GraphicHandler bereitgestellte Grafik-Kontext ist ein Objekt der Klasse BrailleMapGraphics, welche Grafikroutinen bereitstellt, die wie in [Krö08] beschrieben denen der .Net-Klasse „Graphics“ nachempfunden sind. Diese unterstützen auch das Ausgeben von Text in Blindenschrift, deren Definition aus XML-Dateien geladen werden kann, wie sie Listing 10 zeigt. Darin werden rechteckige Felder mit beliebigen Kombinationen aus gehobenen und gesenkten Stiften zu Unicode-Zeichen zugeordnet. So können damit verschiedenste taktile Schriften abgebildet werden (siehe Abschnitt 3.1.2.6). Die Dateien sind auch durch Blinde selbst leicht zu erstellen. Zur Ausgabe von Text können mehrere Schriftdefinitionen angegeben werden, die sich ergänzen. So kann ein Benutzer leicht Darstellungen einzelner Zeichen ändern oder zu einer Standard-Definition von Blindenschrift ergänzen, ohne selbst eine vollständige Schriftdefinitionsdatei erstellen zu müssen. Da zu jeder Textausgabe separat angegeben werden kann, welche Schriftdefinitionen verwendet werden sollen, können leicht in unterschiedlichen Kontexten verschiedene Schriften zum Einsatz kommen. Dabei werden die Schriftdefinitionen natürlich nicht bei jeder Textausgabe neu eingelesen, sondern lediglich einmal und dann wie in [Krö08] beschrieben, in Bitmaps abgelegt, die leicht in die BrailleMap-Ausgabe eingefügt werden können¹³.

Mit Hilfe des GraphicHandler und der implementierten Grafikkanäle können sehr schnell unabhängig vom HyperReader weitere Anwendungen für grafisch-taktile Displays erstellt werden, die neben der taktilen Ausgabe auch eine Ausgabe für Sehende bieten (siehe dazu [TRSE10]). Für andere Ausgabegeräte als das BrailleDis 9000 müssen natürlich weitere Grafikkanäle implementiert werden, die den entsprechenden Gerätetreiber ansprechen. Als Grafikkanal kann auch eine Remoteausgabe umgesetzt werden. Zudem können mehrere grafisch-taktile Displays gleichzeitig angesteuert werden, was die Zusammenarbeit zwischen Blinden fördern kann.

Zur Unterstützung von Interaktionen muss es möglich sein, zu ermitteln, welche OSM-Objekte auf welchen Punkt in der Ausgabe abgebildet wurden. Dazu bietet das Rendering-Framework entsprechende Schnittstellen, die berührte Objekte, also Objekte in einem bestimmten Bereich von Stiften, zusammen mit einer Angabe von Zonen, in denen sie berührt wurden, sowie den Koordinaten des Bildschirmbereichs, der auf die Stifte abgebildet wurde (sofern ermittelbar), zurückgeben. Dabei wird die gesamte Objekt-Hierarchie zurückgegeben, also nicht nur die Objekte, die direkt berührt wurden, sondern auch deren Eltern, um

¹³ Die in [Krö08] beschriebenen Methoden zum Einlesen der Schriftdateien wurden im Rahmen dieser Arbeit an das später eingeführte XML-Format und die Änderungen im BrailleMap-Format angepasst.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Interaktionen wie den hierarchischen Zoom (siehe Abschnitt 3.2.2.3) zu unterstützen. Die Einteilung in Zonen wurde wie in [Krö08] beschrieben umgesetzt, da auf einem grafisch-taktilen Display mit einem Finger ein größerer Bereich von Stiften abgedeckt wird und somit evtl. auch mehrere Objekte berührt werden. Durch Angabe von Zonen können Objekte, die großflächig abgedeckt wurden, von solchen, die nur leicht berührt wurden, unterschieden werden. Zudem können unterschiedliche Zonen verschiedene, interessante Bereiche eines Objekts beschreiben beispielsweise den Aufklappschalter in einer Combobox. Diese Art der Objektermittlung aus Koordinaten des grafisch-taktilen Displays wurde im entwickelten Rendering-Framework auch zur Umsetzung der Strukturleiste genutzt. In der Strukturleiste werden Buchstabenkürzel für bestimmte Eigenschaften im aktiven Darstellungsbereich verwendet. Zur Ermittlung der Objekte mit diesen Eigenschaften wird der Darstellungsbereich entsprechend der verwendeten Schriftgröße der Strukturleiste in Bereiche eingeteilt, die mit der Methode zur Ermittlung berührter Objekte bearbeitet werden.

```
<!-- Eurobraille -->
<braillefont ...>
...
<sign name="A"><unicode format="hex">0041</unicode>
  <braille width="2" height="4">
    <line>10</line>
    <line>00</line>
    <line>00</line>
    <line>10</line>
  </braille>
</sign>
<sign name="B"><unicode format="hex">0042</unicode>
  <braille width="2" height="4">
    <line>10</line>
    <line>10</line>
    <line>00</line>
    <line>10</line>
  </braille>
</sign>
...
</braillefont>

<!-- Fakoo-Schrift (siehe http://www.fakoo.de/fakoo.html) -->
<braillefont ...>
...
<sign name="A"><unicode format="hex">0041</unicode>
  <braille width="3" height="3">
    <line>010</line>
    <line>111</line>
    <line>101</line>
  </braille>
</sign>
...
</braillefont>
```

Listing 10 Ausschnitte aus Blindenschrift-Definitionsdateien für das Rendering-Framework

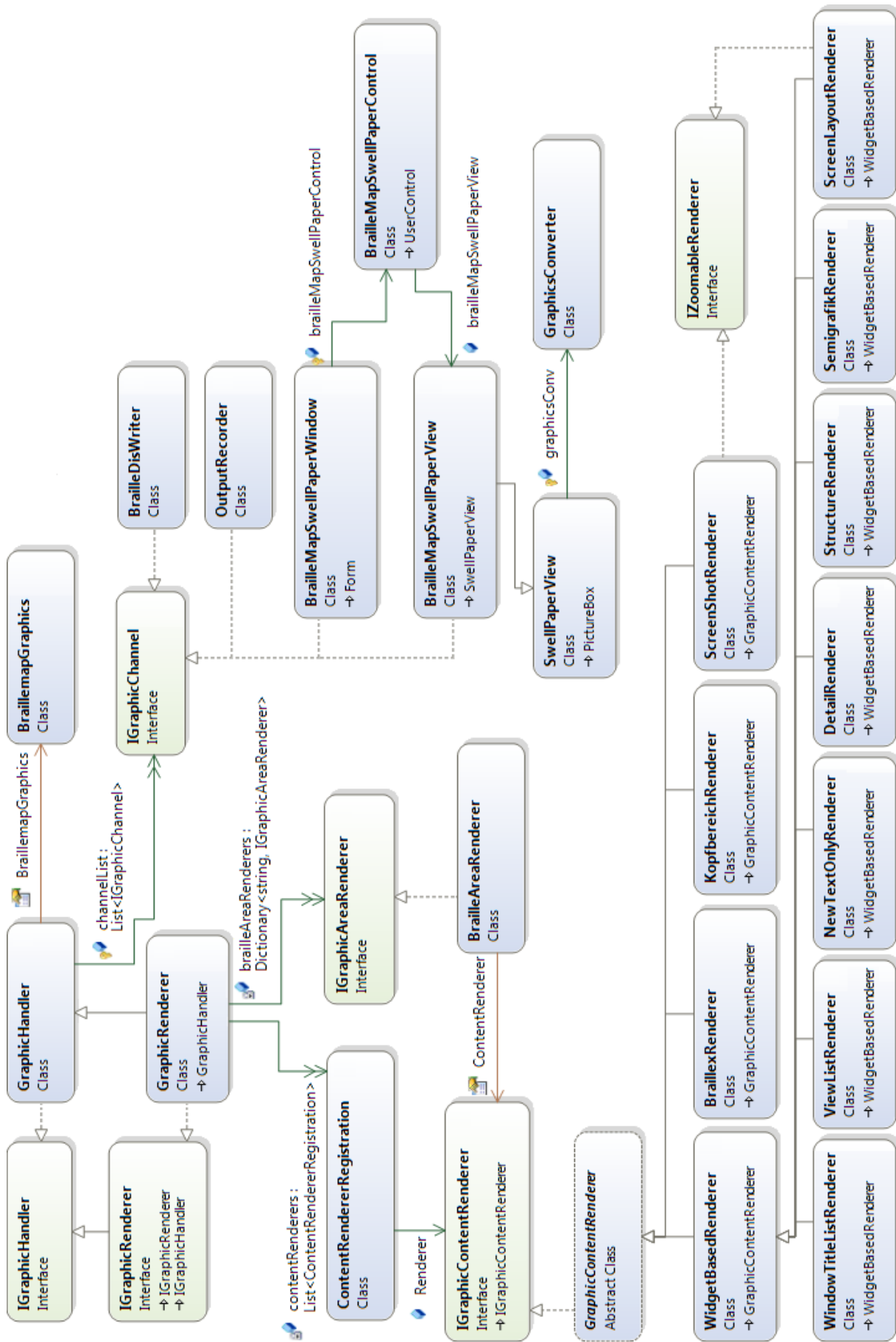


Abbildung 99 Diagramm der wesentlichen Klassen und Interfaces des grafisch-taktilen Renderers

3.3.2.2 Erweiterung und Konfiguration des Rendering-Prozesses

Der wichtigste Teil des Rendering-Frameworks für den HyperReader sind die Content-Renderer. Diese erzeugen die grafisch-taktile Darstellung von OSM-Objekten oder auch anderer Objekte, wie beispielsweise die Fenstertitel-Liste oder die Ansichtsarten-Liste. Ihre Schnittstellen sind diesbezüglich nicht beschränkt. Im Rahmen dieser Arbeit wurden für den HyperReader insgesamt zehn ContentRenderer umgesetzt (siehe Abbildung 98), davon je einer für die vier Ansichtsarten des Darstellungsbereichs und je einer für die fünf anderen Bereiche im HyperBraille-Darstellungskonzept, sowie einer zur Anzeige von Ausgaben aktueller Screenreader, die alternativ zum Detailbereich genutzt werden können. Jeder ContentRenderer implementiert eigene Darstellungsvorschriften zur Erzeugung eines BrailleMap aus den gegebenen Objekten und stellt entsprechende Eigenschaften und Methoden bereit, über die die Darstellung beeinflusst und mit Interaktionen gesteuert werden kann. So kann in der Arbeitsansicht beispielsweise entschieden werden, ob Tabellen in einem tabellarischen Layout präsentiert werden sollen oder nicht. Ein tabellarisches Layout kann bei kurzen Zellinhalten einen guten Überblick über die Tabelle und Zusammenhänge in deren Zellen geben. Spalten- oder zeilenübergreifende Zellen lassen sich so gut erfassen. Ein linearisiertes Layout vermeidet horizontales Scrollen, was sonst besonders bei längeren Zellinhalten schnell auftreten kann. Abbildung 100 und Abbildung 101 illustrieren dies anhand einer Auswertungstabelle zur Lesegeschwindigkeit von Braille auf verschiedenen Medien. Im linearen Layout (Abbildung 100) können für ein Medium wie beispielsweise die Braillezeile alle erfassten Ergebnisse ohne weiteres Scrollen gelesen werden. Die Zuordnung der einzelnen Werte zu den Probanden ist aber schwierig. Im tabellarischen Layout (Abbildung 101) ist dies leicht möglich. Allerdings ist hier horizontales Scrollen nötig, um zu erfassen, welches Medium in welcher Tabellenzeile abgebildet ist.

gelesene Wörter / min (probandenspezifisch)	
Proband 1	Proband 2
Proband 3	Proband 4
Proband 6	Proband 7
Proband 8	
Braillezeile	
1 2, 8 2	7 1, 0 0
5 3	4 4, 8
6 3	5 9, 2
1 9	
6 0, 4	3 0, 2
4 1, 4	
Papier	
1 3 0 1	6 5, 0 5
5 8, 9 1	1 6, 2 6
9 2, 8 0	5 5, 6
4 7, 6	7 9, 2
9 2, 8	2 2, 6

Abbildung 100 Grafisch-taktile Ausgabe einer Excel-Tabelle in linearisierter Tabellendarstellung
Alle Ergebnisse zum Medium „Braillezeile“ sind ohne weiteres Scrollen lesbar.

g e l e s e n e W ö r t e r / m i n (p r o b a n d e n s p e z i f i s c h)									
P r o b a n d 1		P r o b a n d 2		P r o b a n d 3		P r o b a n d 4			
5 3	4 4 , 8	6 3	5 9 , 2						
5 5 , 6	4 7 , 6	7 9 , 2	9 2 , 8						
4 4 , 6	3 3 , 6	7 6 , 2	5 8 , 4						
3 9 , 4	3 0 , 2	5 3 , 6	5 9 , 6						
4 8 , 2 0	3 9 , 0 5	6 8 , 0 0	6 7 , 5 0						
4 6 , 9 5	4 4 , 6 0	7 7 , 7 6	6 0 , 0 0						
4 9 , 0 5	3 2 , 8 3	5 6 , 0 3	6 9 , 8 4						
5 1 , 2 0	4 2 , 4 6	6 9 , 0 0	7 0 , 4 6						

Abbildung 101 Grafisch-taktiler Ausgabe einer Excel-Tabelle in tabellarischer Tabellendarstellung

Die Ergebnisse lassen sich gut zu den einzelnen Probanden zuordnen. Um die gesamte Tabelle zu erfassen ist aber sowohl vertikales wie auch horizontales Scrollen nötig, wie die Scrollbalken in der Darstellung zeigen.

Der ContentRenderer für die Originalansicht setzt viele der in Abschnitt 3.2.3 vorgestellten Möglichkeiten zur Aufbereitung von Rastergrafiken um und bietet daher beispielsweise Einstellungen zur Konfiguration der Segmentierungsfarben und eines Helligkeitsschwellwertes. Zudem verfügen alle ContentRenderer über Angaben zum Viewport, der definiert, welcher Ausgabeplatz dem Renderer zur Verfügung steht und welcher Bereich der Darstellung des Renderers gerade sichtbar ist. So kann die Darstellung an den Viewport angepasst werden und die Darstellung verändernde Algorithmen können effizient arbeiten. Schnittstellen für Scrollfunktionen ermöglichen das Verschieben des Viewports. Zoombare Ansichten, also Original- und Überblicksansicht, bieten entsprechende Schnittstellen zur Einpassung der Darstellung in den Viewport oder ein Vielfaches davon an. Dabei wurde der hierarchische Zoom (siehe Abschnitt 3.2.2.3) nicht direkt in den jeweiligen Renderern umgesetzt, sondern mit Hilfe der Interaktionskomponente des HyperReaders, die von Projektpartnern entwickelt wurde und unter anderem den anzuzeigenden OSM-Inhalt bestimmt. In der Überblicksansicht kann zudem mit einem semantischen Zoom in layoutgetreuer Darstellung gearbeitet werden. Die Originalansicht setzt eher grafischen Zoom ein, wobei auch der in Abschnitt 3.2.3.4 beschriebene angepasste Skalierungsalgorithmus eingesetzt wird.

Ein wesentlicher Aspekt bei der Entwicklung der ContentRenderer war deren Erweiterbarkeit durch externe Komponenten. Besonders bei Symbol- und Arbeitsansicht ist es sehr aufwändig, für jede Anwendung eine optimale Darstellung umzusetzen. Deshalb sollten die Rendering-Vorschriften beispielsweise durch die Entwickler der Anwendung oder auch durch Blinde selbst leicht zu ergänzen sein. Diese Ergänzungen sollten über das allgemein für den HyperReader genutzte Add-In-System möglich sein [SKW10]. Als einzig umsetzbarer Lösung

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

ergab sich dabei die Verwendung eines Event-Handler-Mechanismus. Jeder Content-Renderer bietet ein Event an, für das sich externe Komponenten mit Handler-Methoden registrieren können. Der für eine bestimmte Ansicht oder einen Bereich zu erweiternde Content-Renderer kann über den Graphic-Renderer ermittelt werden. Sobald ein Objekt gerendert werden soll, löst der Content-Renderer das Event aus und stößt damit die registrierten Event-Handler an. Die Abarbeitung der Event-Handler erfolgt sequentiell beginnend mit dem zuletzt registrierten. So können neu hinzugekommene Komponenten die Implementierungen anderer überschreiben. Sobald ein Event-Handler ein Rendering-Ergebnis liefert, wird die Bearbeitung des Events abgebrochen. Alternativ kann ein Event-Handler auch ein Cancel-Attribut setzen, um die Behandlung zum Beispiel zum Ausblenden eines Objekts ohne Rendering-Ergebnis zu beenden. Ein Event-Handler kann die Ereignis-Behandlungsroutine des Content-Renderers auch selbst aufrufen, um das Rendering-Ergebnis zu ermitteln, welches entstehen würde, wenn er selbst nicht in das Rendering eingreifen würde. So können bestehende Implementierungen genutzt werden, wenn nur leichte Anpassungen, wie beispielsweise die Ergänzung eines Rahmens, umgesetzt werden sollen. Die Ereignis-Behandlungsroutine des Content-Renderers schließt Event-Handler, die sich bereits im Aufruf-Stapel befinden, automatisch aus, um Endlosaufrufe zu vermeiden, zumindest wenn diese Event-Handler die Ereignisbehandlung mit dem gleichen darzustellenden Objekt aufrufen, mit dem sie selbst aufgerufen wurden. Listing 11 zeigt den entsprechenden Ausschnitt der Ereignis-Behandlungsroutine. Der Aufruf der Ereignis-Behandlungsroutine des Content-Renderers dient auch dazu, Kindobjekte des darzustellenden Objekts zu bearbeiten. Ein Event-Handler ist immer für die Gesamtdarstellung des darzustellenden Objekts verantwortlich.

Um auch zwischen Event-Handlern aus verschiedenen Komponenten Rendering-Parameter austauschen zu können, bieten Content-Renderer einen Rendering-Kontext, in den beliebige Informationen kombiniert mit einem Zugriffsschlüssel eingetragen werden können. Dieser wird beispielsweise in der Umsetzung der Überblicksansicht zur Speicherung der Position des Parent-Widgets für das darzustellende Objekt genutzt, um die im OSM mit absoluten Koordinaten abgelegten Positionsangaben in die für die Darstellung nötigen relativen Koordinaten umzusetzen.

Listing 12 zeigt ein beispielhaftes Add-In zur Beeinflussung der Symbolansicht des Windows-Spiels „Solitär“. Seine Auswirkungen werden in Abbildung 102, welche einen Screenshot eines Solitär-Spiels zeigt, Abbildung 103, die das Standardrendering der Symbolansicht für dieses Solitär-Spiel darstellt, und Abbildung 104, die das Rendering unter Einfluss des Add-Ins zeigt, veranschaulicht. Das Add-In registriert drei Event-Handler. Davon setzt der erste bei Aufruf durch das Rendering-Event für ein Fenster namens „Solitär“ einen entsprechenden Eintrag in den Ereignis-Kontext. Die anderen beiden Handler führen ihre Änderungen nur aus, wenn dieser Eintrag vorhanden ist. Diese wiederum blenden das Systemmenü aus bzw. kürzen die Beschriftungen der Schaltflächen, welche die Spielkarten repräsentieren.

```

Dictionary<MethodBase, List<Object>> eventCallers = new ...();
public virtual Object GetRenderObjectFor
    (Object obj, IDictionary<string, object> context) {
    RenderEventArgs eventArgs = new RenderEventArgs(obj, context) {
        Cancel = false, RenderObject = null };

    if (RenderEvent != null) {
        Delegate[] handlers = RenderEvent.GetInvocationList();

    if (handlers != null) {
        int h = handlers.Length;
        EventHandler<RenderEventArgs> handler = null;
        bool callThisHandler; bool renderResultFound = false;
        List<Object> callObjects = null;

        while ((h > 0) && (!renderResultFound)) {
            h--; callThisHandler = true;
            handler = handlers[h] as EventHandler<RenderEventArgs>;

            if (handler != null) {
                // check if handler was called before with the same object
                if (eventCallers.TryGetValue(handler.Method, out callObjects)) {
                    if (callObjects.Contains(obj)) {
                        callThisHandler = false; // don't call this handler again
                    } else {
                        // remember that the handler will be called with this object
                        callObjects.Add(obj); }
                } else {
                    // remember that the handler will be called with this object
                    eventCallers.Add(handler.Method, new List<Object>() { obj });
                }

                if (callThisHandler) {
                    try {
                        handler(this, eventArgs); // call handler
                        if ((eventArgs.RenderObject != null) || eventArgs.Cancel) {
                            renderResultFound = true; }
                    } catch (Exception e) { //...
                    } finally {
                        // remove handler (with current object) from callers storage
                        callObjects = this.eventCallers[handler.Method];
                        callObjects.Remove(obj);
                        if (callObjects.Count == 0) {
                            this.eventCallers.Remove(handler.Method);
                        }
                    }
                }
            }
        }
    }
    return eventArgs.RenderObject;
}

```

Listing 11 Ausschnitt aus der Behandlungsroutine des Rendering-Events der ContentRenderer, der die Vermeidung von Endlosaufrufen implementiert

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

```
public class SolitaireAddIn : AddIns.IAddInInitialize {
    Dictionary<String,String> farben = new Dictionary<string,string>() {
        {"Kreuz", "k"}, {"Karo", "c"}, {"Herz", "h"}, {"Pik", "p"} };
    Dictionary<String,String> werte = new Dictionary<string,string>() {
        {"zwei", "2"}, {"drei", "3"}, {"vier", "4"}, {"fünf", "5"}, {"sechs", "6"},
        {"sieben", "7"}, {"acht", "8"}, {"neun", "9"}, {"zehn", "10"},
        {"bube", "b"}, {"dame", "d"}, {"könig", "k"}, {"ass", "a"} };

    public void Initialize(AddIns.IHostModuleGetter moduleMaster) {
        AddIns.IGraphicRenderer gr = moduleMaster.GetGraphicsRenderer();
        if (gr != null) {
            ContentRenderer symbolRenderer = gr.GetContentRenderer(
                AreaNames.MAIN_AREA1_NAME, ViewNames.SYMBOL_VIEW_NAME, null, null);
            if (symbolRenderer != null) {
                symbolRenderer.RenderEvent +=
                    new EventHandler<RenderEventArgs>(solitaerKontextSetzen);
                symbolRenderer.RenderEvent +=
                    new EventHandler<RenderEventArgs>(systemMenueAusblenden);
                symbolRenderer.RenderEvent +=
                    new EventHandler<RenderEventArgs>(textErsetzung);
            } }

        void solitaerKontextSetzen(object sender, RenderEventArgs e) {
            if (e.ObjectToRender is IOsmWindow) {
                if ((e.ObjectToRender as IOsmWindow).Name.Equals("Solitär")
                    && !e.Context.ContainsKey("Solitär")) {
                    e.Context.Add("Solitär", null);
                } }
        }

        void systemMenueAusblenden(object sender, RenderEventArgs e) {
            IOsmComponent comp = e.ObjectToRender as IOsmComponent;
            if (e.Context.ContainsKey("Solitär") && (comp != null)
                && comp.Name.Equals("Systemmenüleiste")) {
                e.Cancel = true;
            }
        }

        void textErsetzung(object sender, RenderEventArgs e) {
            ContentRenderer renderer = (sender as ContentRenderer);
            if (e.Context.ContainsKey("Solitär")
                && (e.ObjectToRender is IOsmButton) && (renderer != null)) {

                e.RenderObject = renderer.GetRenderObjectFor(
                    e.ObjectToRender, e.Context);

                if (e.RenderObject is HBW.CommandButton) {
                    String text = String.Empty;
                    String name = (e.ObjectToRender as IOsmButton).Name;

                    foreach (KeyValuePair<String, String> kvp in farben) {
                        if (name.IndexOf(kvp.Key) > -1) {
                            text = text + kvp.Value; break;
                        }
                    }
                    foreach (KeyValuePair<String, String> kvp in werte) {
                        if (name.IndexOf(kvp.Key) > -1) {
                            text = text + kvp.Value; break;
                        }
                    }
                    if (text.Length > 0) {
                        (e.RenderObject as HBW.CommandButton).Text = text;
                    }
                } } }
    }
}
```

Listing 12 Ausschnitt aus einem Rendering-Add-In für Solitär

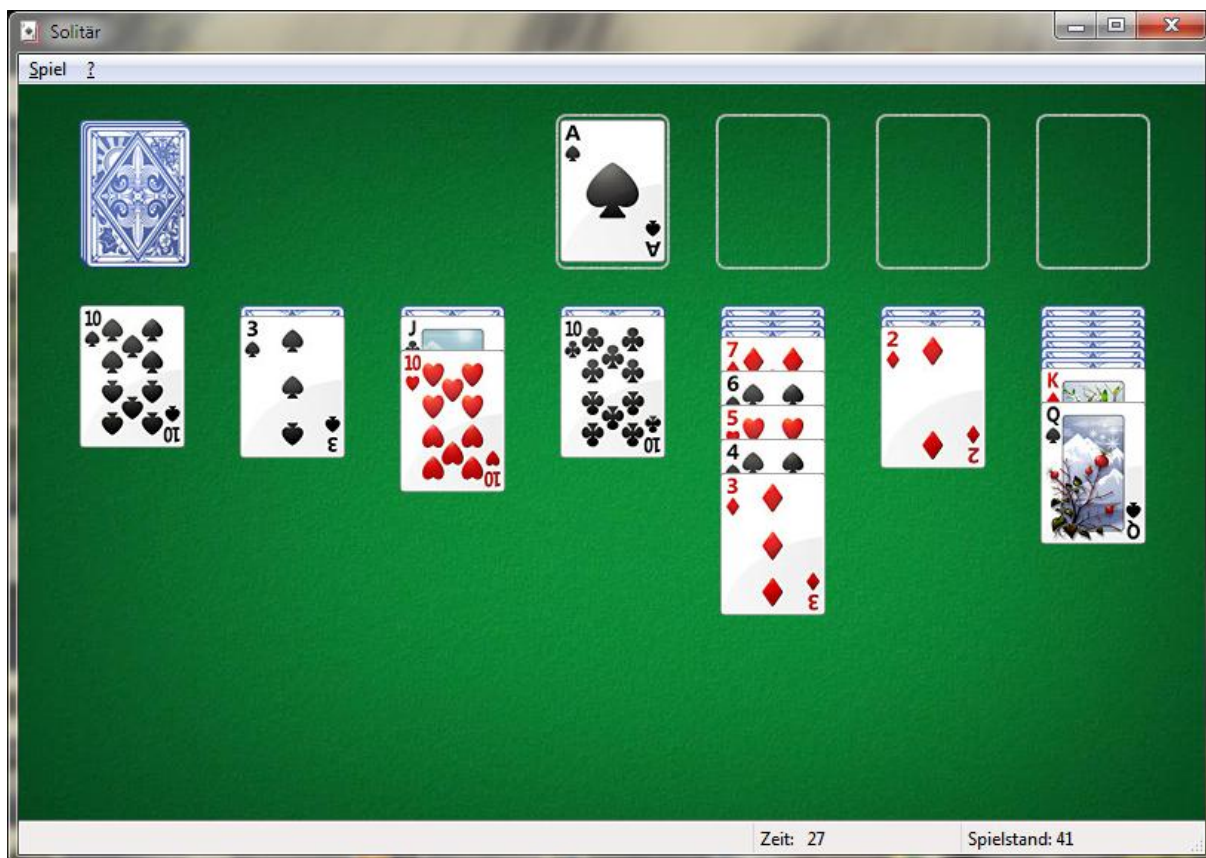


Abbildung 102 Screenshot eines Solitär-Spiels als Beispiel für die Wirkungsmöglichkeiten von Add-Ins

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

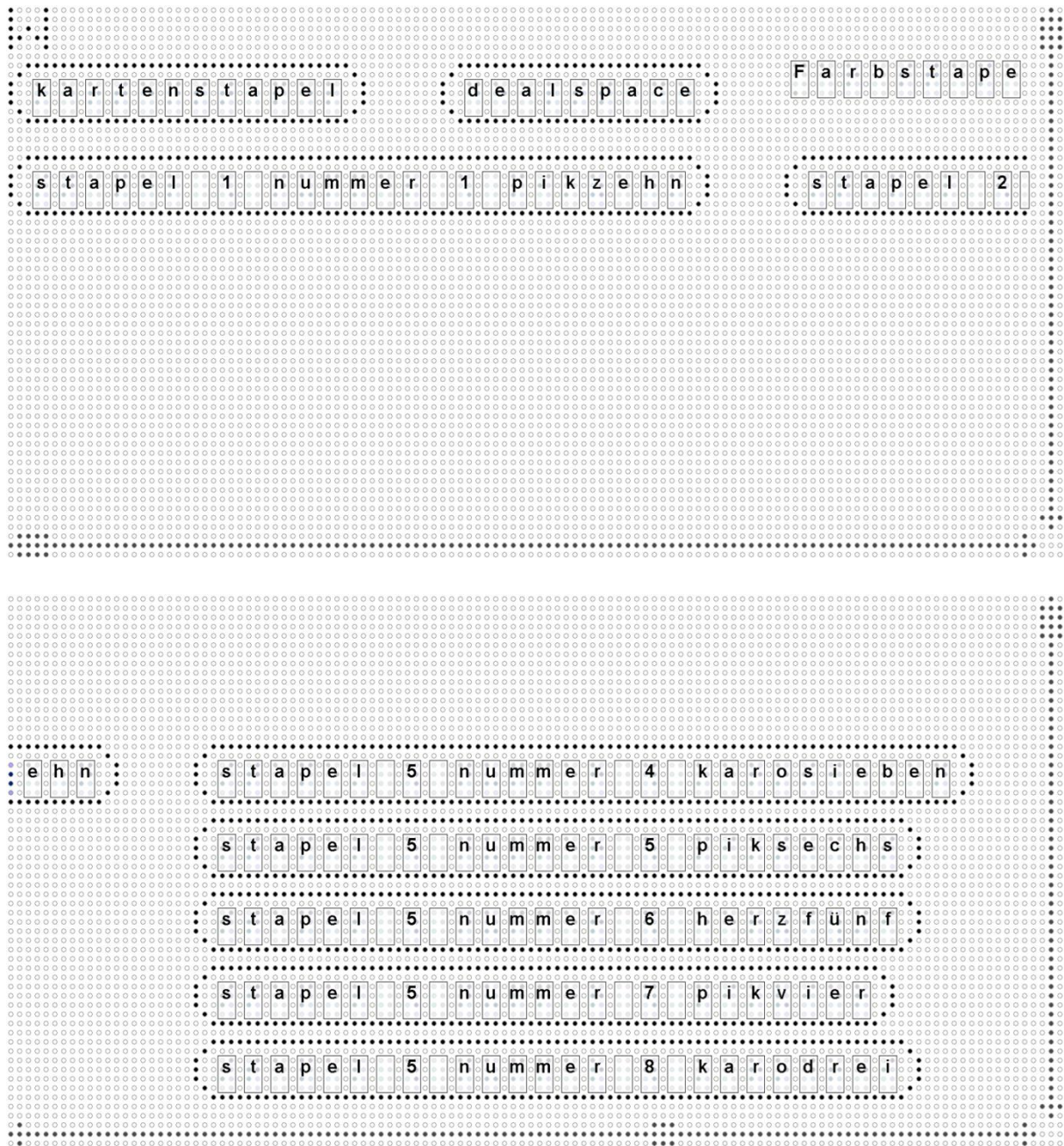


Abbildung 103 Zwei Ausschnitte aus der taktilen Ausgabe für das Solitär-Spiel aus Abbildung 102 ohne Anwendung des Add-Ins aus Listing 12

Die Schaltflächen, die die Spielkarten repräsentieren, tragen hier die über UIAutomation ermittelten Beschriftungen, die vor allem für eine gute Sprachausgabe optimiert wurden, z.B. „Stapel 5 Nummer 4 Karosieben“. Auch für Braillezeilen sind diese Angaben sinnvoll, da sich der Nutzer mit Hilfe einer einzeiligen Ausgabe auf dem zweidimensionalen Spielfeld orientieren muss. Für ein grafisch-taktilen Display ist dies aber nicht nötig und erzeugt nur enormen Scrollaufwand.

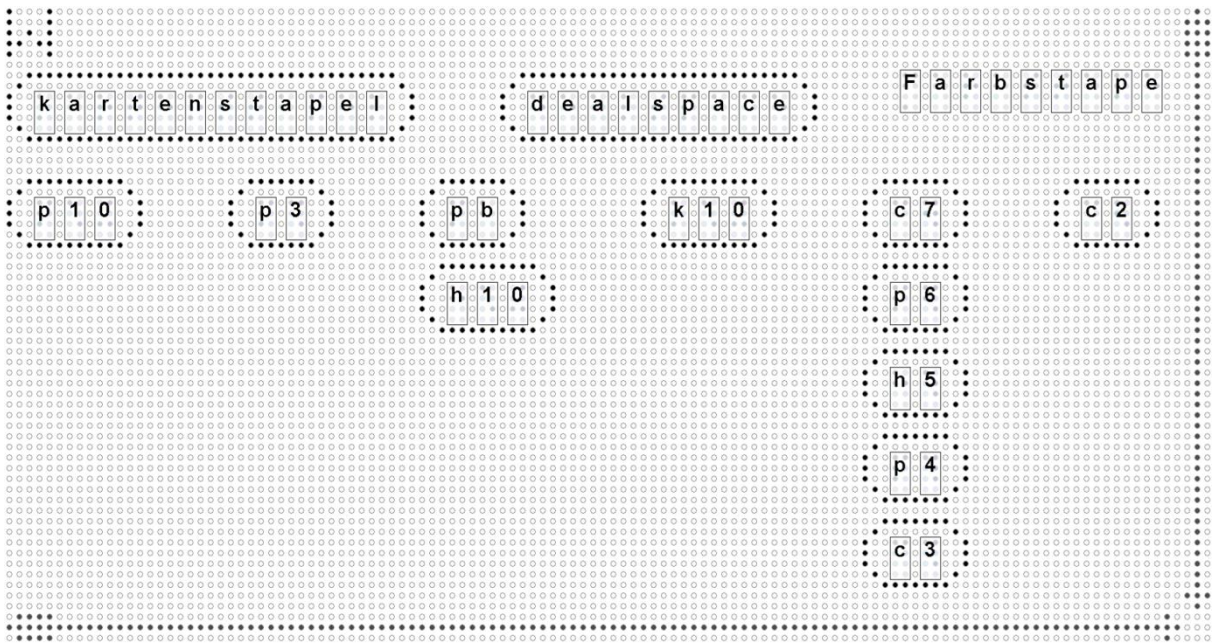


Abbildung 104 Ausschnitt aus der taktilen Ausgabe für das Solitär-Spiel aus Abbildung 102 mit Anwendung des Add-Ins aus Listing 12

Die Beschriftungen der Schaltflächen, die die Spielkarten repräsentieren, wurden durch die Textersetzung stark gekürzt. So wurde beispielsweise aus der langen Beschriftung „Stapel 5 Nummer 4 Karosieben“ einfach „c7“, was für „Karosieben“ steht. Die Information „Stapel 5“ ist aufgrund der zweidimensionalen Ausgabe nicht mehr nötig. Die Information „Nummer 4“, welche darauf hinweist, dass sich unter der Karosieben noch drei verdeckte Karten befinden, wird in der vollständigen Version des Add-Ins durch die entsprechende Anzahl horizontaler Linien oberhalb der Schaltfläche repräsentiert (siehe [TRSE10]). Durch die Kürzung der Beschriftungen passt bereits fast das gesamte Spielfeld auf die Ausgabefläche des BrailleDis 9000. Durch Anpassung der Abstände zwischen den Schaltflächen kann die nötige Ausgabefläche weiter reduziert werden.

Neben den Add-In-Schnittstellen, die zur Beeinflussung des Renderings Programmierung erfordern, wurden ähnlich zu den bereits vorgestellten Schriftdateien viele weitere Möglichkeiten zur Einflussnahme auf die Darstellung und teilweise auch Interaktion integriert, die ohne Programmierung auskommen und auch durch Blinde selbst genutzt werden können. Dabei wurde Wert darauf gelegt, verschiedene Techniken und Formate zu verwenden, um deren Nutzen und Akzeptanz durch die Anwender zu evaluieren. Für die Symbolansicht können in einer Datei Textverkürzungen definiert werden. Die für die Strukturleiste verwendeten Eigenschaften und Kürzel werden ebenfalls in einer XML-Datei abgelegt. Hierzu können wie Listing 13 zeigt einfachen Und-, Oder- und Not-Verknüpfungen auf C#-Properties von Objekten und deren Werte angewendet werden. Die für den Detailbereich und die Arbeitsansicht definierten Typkürzel wie „slt“ können mit Hilfe des im HyperReader integrierten Konfigurations-Moduls zur Laufzeit geändert werden. Ebenso kann die Darstellung von fokussierten Elementen und der Minimap, sowie der Bezugspunkt beim Zoomen (obere linke Ecke oder Mitte) über das Konfigurations-Modul geändert werden.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

```
<?xml version="1.0" encoding="utf-8" ?>
<structureProperties>
  <structureProperty shortName="S" longName="Spezial (Fett und Kursiv)">
    <and>
      <property name="FontWeight" value="bold"/>
      <property name="FontStyle" value="italic"/>
    </and>
  </structureProperty>
  <structureProperty shortName="F" longName="Fett und nicht Kursiv">
    <and>
      <property name="FontWeight" value="bold"/>
      <not><property name="FontStyle" value="italic"/></not>
    </and>
  </structureProperty>
  <structureProperty shortName="D" longName="nach Description">
    <or>
      <property name="Description" value="Wort"/>
      <property name="Description" value="Punkt"/>
    </or>
  </structureProperty>
</structureProperties>
```

Listing 13 Beispiel einer Konfigurationsdatei für die Strukturleiste

Durch diese Konfiguration würden in der Strukturleiste Elemente, die fett und kursiv geschrieben sind, mit „S“ markiert werden, Elemente, die nur fett, aber nicht kursiv sind, mit „F“ und Elemente, deren Beschreibung „Wort“ oder „Punkt“ lautet mit „D“.

Alle Konfigurationsmöglichkeiten wurden von den Projektpartnern gut angenommen. Es zeigte sich, dass XML-Dateien hilfreicher sind als bloße Konfigurationen über das Konfigurations-Modul. Dies lag vor allem daran, dass die als Beispiele abgelegten XML-Dateien bereits alle Konfigurationsoptionen deutlich machten. Bei den einfacheren Konfigurationen über das Konfigurationsmodul war ohne Dokumentation nicht erkennbar, welche Konfigurationswerte angegeben werden können. Entsprechend aufbereitete Dialoge wären hier sinnvoll. Allerdings ist der Abgleich zwischen Konfigurations-Modul und in Code verwendeten Konfigurationswerten noch umständlich, da er nicht automatisch erfolgt und keine Beschränkung durch das Konfigurations-Modul¹⁴ gesetzt wird. Die bisher verwendeten, einfachen XML-Dateien wurden von allen Projektpartnern auch gut verstanden. Bei komplexeren Definitionen, wie beispielsweise den XML-Dateien für das in [Haa10] vorgestellte Konfigurations-Modul für das Rendering-Framework zur Beeinflussung von Widgets, sowie Erstellung neuer Widgets und Rendering-Vorschriften (sog. Layoutvorlagen) aus XML-Dateien, wären sicher auch Editoren nötig. Listing 14 und Listing 15 zeigen zwei Beispiele dazu

¹⁴ Das HyperBraille-Konfigurations-Modul wurde von einem anderen Projekt-Partner entwickelt


```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <widgets><container
className="^HyperBraille\.\HBGraphicRenderer\.\Widgets\.\CommandButton$"
  <value key="enabledBorder"
        type="HyperBraille.HBGraphicRenderer.Widgets.Border"
        dashPattern="01011"/>
  <value key="vFarPadding" type="int">0</value>
  <value key="replacements" type="HyperBraille.HBGraphicRenderer.
        GraphicsConfiguration.PropertyValueReplacements">
    <item>
      <property>Text</property>
      <replacement>
        <original type="string">Abbrechen</original>
        <newValue type="string">abbr</newValue>
      </replacement>
    </item>
  </value>
</container></widgets>
</configuration>

```

Listing 14 Beispiel zur Änderung eines Widgets mit dem Konfigurations-Modul aus [Haa10]
 Hierbei wird für das CommandButton-Widget allgemein ein gestrichelter Rahmen definiert. Der Abstand zwischen Rahmen und Text wird auf 0 gesetzt. Weiterhin wird eine Textersetzung von „Abbrechen“ nach „abbr“ definiert.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <layoutTemplates><template base="HyperBraille.HBGraphicRenderer.
    ContentRenderer.LayoutTemplates.BorderLayoutTemplate" name="DialogBox">
    <category name="ControlButton" noFurtherProcessing="true">
      <pane index="0"/></category>
      <pane side="east"/>
    </template></layoutTemplates>
  <rendering>
    <container interfaceName="IOsmButton$" priority="high">
      <filter><equal>
        <property name="Name"><sourceObject/></property>
        <literal type="string">Cancel</literal>
      </equal></filter>
      <value key="renderAs" type="System.String">
        HyperBraille.HBGraphicRenderer.Widgets.AdaptableWidget</value>
      <value key="renderProps" type="HyperBraille.HBGraphicRenderer.
        GraphicsConfiguration.RenderedObjectSettings">
        <literal>
          <property name="Settings" type="HyperBraille.HBGraphicRenderer.
            GraphicsConfiguration.AdaptableWidgetSettings">
            <bottomBorder dashPattern="10110" crossSection="11" visible="true"/>
            <subElement type="text" extendBorderHorizontally="true"
              extendBorderVertically="true" horizontalAnchor="Near"
              verticalAnchor="Near" text="Abbrechen"/>
            <subElement type="text" extendBorderVertically="true"
              verticalAnchor="Center" horizontalAnchor="AfterEnd" text="123"/>
          </property>
          <property name="AutoSize" type="bool">true</property>
        </literal>
      </value>
    </container>
    <container interfaceName="IOsmButton$" name="Help">
      <value key="hide" type="bool">true</value>
    </container>
    <container interfaceName="IOsmButton$" name="OK">
      <value key="layoutCategory" type="HyperBraille.HBGraphicRenderer.
        ContentRenderer.LayoutTemplates.LayoutCategories">
        <category priority="1">ControlButton</category>
      </value>
    </container>
    <container interfaceName="IOsmPane" name="ExampleDlg">
      <value key="layoutTemplate" type="string">DialogBox</value>
    </container>
  </rendering>
</configuration>
```

Listing 15 Beispiel zur Änderung des Rendering-Prozesses mit dem Konfigurationsmodul aus [Haa10] Die Datei definiert ein Template namens „DialogBox“, in dem Elemente der Kategorie „Control-Button“ in einer eigenen Spalte rechts dargestellt werden sollen. Dieses Template wird auf ein Element des Typs „IOsmPane“ mit dem Namen „ExampleDlg“ angewendet. Ein Element des Typs „IOsmButton“ mit dem Namen „OK“ wird der Kategorie „ControlButton“ zugewiesen. Ein „IOsm-Button“ mit dem Namen „Help“ wird ausgeblendet. Für den „IOsmButton“ mit Namen „Cancel“ wird auf Grundlage des speziell für diesen Zweck erstellen „AdaptableWidget“ ein neues Widget definiert, welches unten einen dicken, gepunkteten Rahmen hat und mit „Abbrechen“ und „123“ beschriftet ist, wobei „123“ unter dem Rahmen steht.

3.3.2.3 Umsetzung taktiler Widgets

Die im Rahmen dieser Arbeit entwickelten ContentRenderer arbeiten mit eigens für Hyper-Braille entwickelten Widgets. Diese definieren sich, wie Abbildung 105 am Beispiel des Button-Widgets zeigt, durch Tactons, Rahmen und Braille. Tactons sind taktile Symbole, die einen hohen Wiedererkennungswert und Unterscheidungswert besitzen sollen und sich in ihrer Größe nie ändern. Sie dienen vor allem der deutlichen Markierung der Elementtypen (vergleiche Abschnitt 3.1.2.9). Zur leichten Änderbarkeit werden Tactons aus ähnlichen XML-Dateien geladen wie die Schriftdefinitionen (siehe Listing 10). Eine detaillierte Beschreibung zum Aufbau der Widgets befindet sich in [Haa10]. Auch Renderer, die augenscheinlich nur Text ausgeben, wie in der Arbeitsansicht, nutzen Widgets, um beispielsweise Layoutoptionen, Unterstreichungen und Cursordarstellungen zu unterstützen. Alle Widgets zeichnen sich als Rastergrafik im BrailleMap-Format (siehe Abschnitt 3.3.2.4) in die Ausgabe.

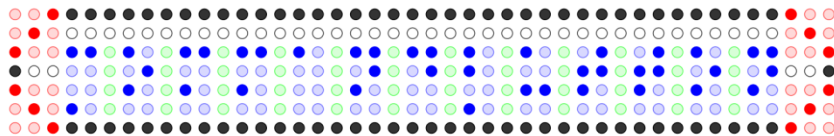


Abbildung 105 *CommandButton-Widget aus [Haa10] mit Markierung der einzelnen Bestandteile*
 Die Eckmarkierungen (vergleiche [Alb08]) wurden über Tactons realisiert (rot). Ein Rahmen bildet den Körper des Buttons (schwarz). In der Mitte befindet sich eine Beschriftung in Braille (Zeichen in blau, Abstände in grün).

Die erste Version der Renderer arbeitete rein auf Rastergrafiken. Jedes darzustellende Element wurde direkt als BrailleMap erstellt. Dies beeinträchtigt aber die Anpassbarkeit der Darstellung. Beispielsweise bei der Erstellung einer tabellarischen Darstellung kann ein Add-In bei Verwendung eines Tabellen-Layout-Widgets leicht den Abstand zwischen Tabellenzellen ändern, sofern entsprechende Attribute vom Widget selbst angeboten werden. Bei direkter Erzeugung einer Rastergrafik dagegen, muss das Add-In den gesamten Rendering-Prozess nachstellen. Zwar könnte der Renderer selbst Attribute für den Abstand von Tabellenzellen anbieten, die durch das Add-In verändert werden. Dies würde aber bedeuten, dass im Renderer alle möglichen Darstellungsattribute vorhergesehen werden müssen. Zudem wäre es sehr aufwändig, in einer Darstellung verschiedene Tabellenlayouts zu nutzen.

Vor Implementierung einer vollständig eigenständigen Widget-Bibliothek wurde zunächst noch die Nutzbarkeit der Windows.Forms-Komponenten des .Net-Frameworks untersucht. Der Vorteil einer darauf basierenden Umsetzung wäre gewesen, dass bestehende und einer fortlaufenden Verbesserung unterliegende Layout-Komponenten direkt hätten genutzt werden können und für Entwickler, die mit Windows.Forms vertraut sind, kaum Einarbeitungsaufwand bestanden hätte. Es zeigten sich allerdings Konflikte mit dem UI-Automation-Filter. Die Windows.Forms-Elemente, die zum Rendering aufgrund von Informationen erstellt wurden, die vom UI-Automation-Filter geliefert wurden, lösten offensichtlich Ereignisse im UI-Automation-Handling aus. Diese wurden zwar nicht zum Filter geleitet, da sie durch einen anderen als dem gefilterten Prozess erzeugt wurden, sie benötigten allerdings Verarbeitungszeit im UI-Automation-Handling selbst, die die Performance des UI-Automation-Filters

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

beeinträchtigte. Des Weiteren können Windows.Forms-Elemente nur innerhalb einer Anwendungs-Domäne ausgetauscht werden, wodurch alle Add-Ins, die Einfluss auf den Renderer nehmen wollen in der gleichen Anwendungs-Domäne wie der Renderer und damit der HyperReader selbst laufen müssen. Dies mindert die Flexibilität der Add-Ins. Diese sollen eigentlich je nach fokussierter Anwendung flexibel ge- und entladen werden können. Müssen sie aber in der gleichen Anwendungsdomäne wie der HyperReader-Prozess laufen, so müssen sie bei Start des HyperReader-Prozesses geladen werden und können erst bei Beendigung des HyperReader-Prozesses wieder entladen werden. Letztlich sind Windows.Forms-Elemente auch nicht thread-sicher, was hohe Anforderungen an die Entwickler von Add-Ins stellt und bei falscher Implementierung schnell dazu führen kann, dass die gesamte Ansicht nicht dargestellt werden kann. Die eigenständige Widget-Bibliothek umgeht diese Nachteile durch thread-sichere Implementierung und Serialisierbarkeit. Dabei wurden wesentliche Attribute von Windows.Forms samt ihrer Benennung und Zugriffsfunktionen übernommen, um Entwicklern mit Erfahrung in Windows.Forms einen leichten Zugang zu ermöglichen.

3.3.2.4 Ein Ausgabeformat für grafisch-taktilen Rendering

Das Ausgabeformat des Renderers sollte neben der Nutzbarkeit zur Ausgabe des Rendering-Ergebnisses auf grafisch-taktilen Displays einige weitere Anforderungen erfüllen. An erster Stelle stand die Unterstützung einer Ausgabe für Sehende. Wie in Kapitel 3.2.5 beschrieben sollte die ausgegebene Brailleschrift für Sehende als Schwarzschrift lesbar gemacht werden und zwar genau in der Größe, der Ausrichtung und an der Position, wie sie auf dem grafisch-taktilen Display dargestellt ist. Zu jedem Stift des grafisch-taktilen Displays muss also ermittelbar sein, ob er Teil eines Schriftzeichens ist und wenn ja, welches Zeichen dort repräsentiert ist. Außerdem sollte das Ausgabeformat das Entfernen abgeschnittener Braillezeichen vor Ausgabe auf dem grafisch-taktilen Display unterstützen, da diese dem Benutzer nichts nützen oder ihn gar verwirren. In der Brailleschrift ist es, im Gegensatz zur Schwarzschrift, nahezu unmöglich aus einem halb abgeschnittenen Schriftzeichen das Original zu erkennen. Somit muss neben der Information, welches Schriftzeichen repräsentiert ist, auch gespeichert werden, ob alle Stifte des Schriftzeichens dargestellt sind.

Das Ausgabeformat sollte auch zum Logging genutzt werden können. Das heißt, jedes Rendering-Ergebnis sollte leicht speicherbar sein und auch leicht aus dem Log wieder auf das grafisch-taktile Display oder in die Darstellung für Sehende geladen werden können. Um Log-Ausgaben auch für Veröffentlichungen und Schulungsmaterialien nutzen zu können, sollte das Ausgabeformat auch mit Standardprogrammen verständlich darstellbar sein.

Wichtige Bereiche in der Ausgabe, wie der Cursor oder ein fokussiertes Element, sollten durch Blinken (wechselndes Heben und Senken der Stifte) markiert werden können. Da die Stifte nur gehoben oder gesenkt sein können, ist eine andere Art der Hervorhebung nicht möglich. Auch dies soll durch das Ausgabeformat unterstützt werden und auch in der Darstellung für Sehende und in Log-Ausgaben erkennbar und später wiederherstellbar sein.

Da die Ausgabe durch Änderungen des Bildschirminhalts und Interaktionen des Benutzers häufig aktualisiert werden kann, müssen die Umwandlungen in die Eingabeformate von grafisch-taktilen Displays und die Darstellung für Sehende effizient umsetzbar sein. Außerdem sollte die Umwandlung in eine einfache Schwarz-Weiß-Darstellung zur Ausgabe auf einem beliebigen grafisch-taktilen Display oder einem grafikfähigen Brailledrucker möglichst auch ohne Kenntnisse über das spezielle Format möglich sein.

Natürlich sollten die Ausgabe auf dem grafisch-taktilen Display, die Ausgabe für Sehende und die Log-Ausgabe immer das Gleiche anzeigen. Sie sollten also möglichst durch den gleichen Rendering-Prozess zustande kommen.

Zusammengefasst ergeben sich somit folgende Anforderungen an das Ausgabeformat:

- Unterstützung von drei Ausgabeformen: grafisch-taktil, für Sehende, Log
- Erzeugung der drei Ausgabeformen aus einem Rendering-Ergebnis
- Effiziente Umwandlung in die drei Ausgabeformen
- Speicherung der Ausdehnung, Position und Orientierung von Schriftzeichen
- Speicherung von Informationen über blinkende Stifte

Um diese Anforderungen zu erfüllen, wurde im Rahmen dieser Arbeit ein spezielles Rastergrafik-Format (Bitmap-Format), genannt „BrailleMap“, entwickelt.

Da die Stifte von grafisch-taktilen Displays in einem zweidimensionalen Feld regelmäßig angeordnet sind, liegt die Verwendung eines Rastergrafik-Formates für die Ausgabe des Renderers nahe. Dabei wäre jeder Stift in einem Pixel der Rastergrafik repräsentiert. Da für die Stifte nur zwei Zustände zur Verfügung stehen, würde theoretisch ein Bit pro Stift genügen. Eine normale Rastergrafik wird aber in der internen Repräsentation in einer Programmierumgebung meist mit 32 Bit pro Pixel abgebildet, wobei je 8 Bit für die Farbkanäle Rot, Grün und Blau genutzt werden und nochmals 8 Bit für den Transparenzkanal Alpha. Bei der Speicherung oder Darstellung von Rastergrafiken in Standard-Ansichtsprogrammen kann die Transparenz-Information allerdings verloren gehen bzw. mit den Farbinformationen verrechnet werden. Häufig werden Rastergrafiken im Bitmap-Format mit nur 24 Bit pro Pixel abgelegt. Aus diesem Grund sollte der Alpha-Kanal nicht für die Codierung von Informationen genutzt werden, sondern immer auf opak (nicht transparent) gesetzt sein. Die Informationen der Renderer-Ausgabe können also in 24 Bit je Stift gespeichert werden, wobei nur ein Bit für die Codierung des Stiftzustandes selbst nötig wäre. Die übrigen 23 Bit pro Pixel bzw. Stift können für andere Informationen genutzt werden. Hier setzt das BrailleMap-Format an. Damit werden alle zur Erfüllung der oben genannten Anforderungen nötigen Informationen direkt in einem Bitmap gespeichert. Vor allem im Hinblick auf die in Abschnitt 3.3.2.2 beschriebenen Rendering-Add-Ins und das Logging ist es sinnvoll, nur eine Speicherstruktur zu benutzen, die sich auch leicht in einem bekannten Dateiformat speichern lässt.

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Das wichtigste Ziel des Ausgabeformats war die Speicherung der Informationen zu taktilen Schriftzeichen zur Rückübersetzung in Schwarzschrift. Um möglichst alle Schriftzeichen abbilden zu können, werden hierfür 16 Bit zur Speicherung des Unicodes eines Zeichens benötigt. Um eine leichte Umwandlung des Ausgabeformates in ein Schwarz-Weiß-Bild auch ohne Kenntnis des genauen Formates zu gewährleisten, muss der Luminanzwert für einen Pixel, der einen gesetzten Stift darstellt unter der halben maximalen Helligkeit liegen und der für einen ungesetzten Stift darüber. So werden ungesetzte Stifte durch Weiß dargestellt und gesetzte durch Schwarz, was der allgemeinen Vorgehensweise entspricht. Der passende Luminanzwert muss zuverlässig unabhängig vom Unicode und anderen Informationen erreicht werden können. Ausgehend von der Luminanzgleichung (Formel (1)), die meist zur Ermittlung des Helligkeitswerts verwendet wird, ist dies nur möglich, wenn die jeweils führenden Bits der Farbkanäle zur Codierung des Stiftzustandes verwendet werden.

$$L = 0,3 \cdot Rot + 0,59 \cdot Grün + 0,11 \cdot Blau \quad (1)$$

Für gesenkte Stifte werden diese entsprechend auf 1 gesetzt und für gehobene auf 0. So ergibt sich bei einer maximalen Helligkeit von 255 für die Extremfälle folgendes:

- gehobener Stift und alle anderen Bits auf 0:
 $L = 0,3 \cdot 0 + 0,59 \cdot 0 + 0,11 \cdot 0 = 0 < 127,5 \Rightarrow Schwarz$
- gesenkter Stift und alle anderen Bits auf 0:
 $L = 0,3 \cdot 128 + 0,59 \cdot 128 + 0,11 \cdot 128 = 128 > 127,5 \Rightarrow Weiß$
- gehobener Stift und alle anderen Bits auf 1:
 $L = 0,3 \cdot 127 + 0,59 \cdot 127 + 0,11 \cdot 127 = 127 < 127,5 \Rightarrow Schwarz$
- gesenkter Stift und alle anderen Bits auf 1:
 $L = 0,3 \cdot 255 + 0,59 \cdot 255 + 0,11 \cdot 255 = 255 > 127,5 \Rightarrow Weiß$

Für ein Pixel, das zu einem Schriftzeichen gehört, sind somit bereits 16 Bit für den Unicode belegt und 3 für den Stiftzustand. Es verbleiben 5 Bit für die Codierung der Ausdehnung und Orientierung des Schriftzeichens. Hierbei ist zu beachten, dass die Größe der Schriftzeichen nicht konstant ist. Normale Brailleschrift allein kann als 6-Punkt oder 8-Punkt-Schrift dargestellt sein, also mit 2×3 Stiften oder 2×4 Stiften. In der Ausgabe des grafisch-taktilen Renderers können diese Formen auch gemischt auftreten. Außerdem sollte die Möglichkeit offen gehalten werden, auch andere Ausdehnungen, um beispielsweise die in Abschnitt 3.1.2.6 vorgestellten Schriften zuzulassen, mathematische Zeichen wie Wurzeln in speziellen Formen darzustellen oder Formatierungszeichen in Dokumenten deutlich vom normalen Text abzuheben. Bei der Ausdehnung eines Schriftzeichens kann also nicht von einem oder mehreren Standardwerten ausgegangen werden, die sich in wenigen Bits codieren ließen. Die einzig mögliche Lösung, ist die Markierung der Zeichengrenzen. Ein Pixel speichert also immer, ob er zum oberen, unteren, linken oder rechten Rand eines Schriftzeichens gehört. Insgesamt gibt es mindestens 9 Zustände, die ein Pixel bezüglich des Randes annehmen kann. Diese sind „oben links“, „oben rechts“, „unten links“, „unten rechts“, „nur links“, „nur oben“, „nur rechts“, „nur unten“ und „nicht am Rand“. Für die Codierung der Randinformation wer-

den also mindestens 4 Bit benötigt. Somit liegt es nahe, die 4 Bit jeweils einem Rand („oben“, „unten“, „links“ und „rechts“) zuzuweisen. So ist jede Kombination möglich.

Durch die Randinformationen lässt sich leicht feststellen, ob ein Schriftzeichen vollständig in der Ausgabe dargestellt ist oder nicht. Ist ein Schriftzeichen-Pixel ohne Randinformation in eine Richtung nur von Grafik-Pixeln umgeben oder nur von Schriftzeichen-Pixeln, die ebenfalls keine Randinformation besitzen oder andere Unicodes enthalten, so ist das Schriftzeichen abgeschnitten und alle Pixel des Schriftzeichens können aus der Ausgabe entfernt werden, also durch Grafik-Pixel mit gesenktem Stift ersetzt werden. Da die Ausgabe nach der Bereinigung nur vollständige Schriftzeichen enthält, genügen die Randinformationen auch, um die Orientierung des Schriftzeichens festzustellen. Die Randinformation bezieht sich immer auf die Braille-Darstellung des Schriftzeichens selbst und nicht auf die Positionierung und Lage in der Ausgabe. Bei einem normalen Braillezeichen sind die Punkte 1 und 4 also immer der obere Rand usw. Abbildung 106 veranschaulicht dies.

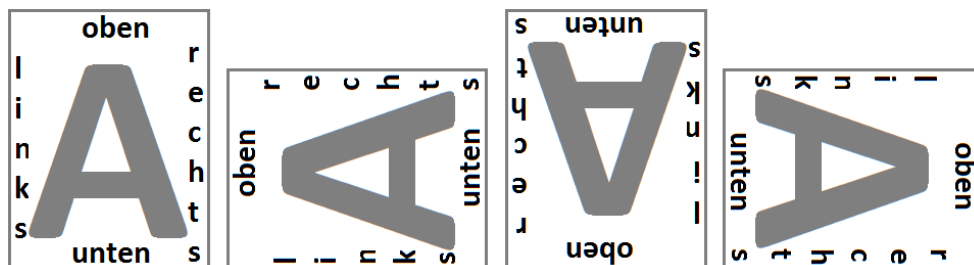


Abbildung 106 Darstellung der Randinformationen zu Schriftzeichen

Damit können mit 23 von 24 verfügbaren Bits bei der Codierung eines Schriftzeichens fast alle oben genannten Anforderungen erfüllt werden. Die Stifthöhe ist eindeutig definiert und kann durch einfache Luminanz-Umwandlung in ein für grafisch-taktile Displays geeignetes Format konvertiert werden. Zum Logging kann die Ausgabe als normale Rastergrafik im Bitmap-Format gespeichert werden, wodurch sie mit allen Standard-Bilddarstellungsprogrammen angezeigt werden kann. Die Ausgabe kann von abgeschnittenen Braillezeichen bereinigt werden. Zur Darstellung für Sehende kann die Ausgabe einfach vergrößert werden und die Braillezeichen können durch Schwarzschriftzeichen in passender Größe und Orientierung dargestellt werden. Für beide Vorgänge ist nur ein Bilddurchlauf nötig. Die Umwandlung ist somit auch effizient ausführbar. Lediglich blinkende Stifte können mit der bisherigen Codierung nicht dargestellt werden. Für Schriftzeichen ist dies allerdings auch nicht gewünscht, da es nur das Lesen erschwert. Zur Markierung eines wichtigen Textes soll besser eine darunter befindliche blinkende Linie erzeugt werden, also ein Grafik-Objekt. Das verbleibende Bit kann sinnvoller dazu genutzt werden, Pixel von Schriftzeichen und Grafik-Objekten zu unterscheiden. Dies bringt enorme Vorteile für die Codierung der Grafik-Pixel.

Durch das Bit zur Entscheidung über Schrift- oder Grafikpixel können 23 von 24 Bit frei für die Codierung der Grafikpixel-Informationen verwendet werden. Neben dem Stifzustand muss dabei lediglich die Blink-Information gespeichert werden. Hierfür werden 2 Bit zur Verfügung gestellt, die drei verschiedene Möglichkeiten des Blinkens codieren, sowie den Zu-

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

stand „kein Blinken“. Die Blink-Bits und das Entscheidungs-Bit werden auf die hinteren Bits der Farbkanäle gelegt, um möglichst wenig Einfluss auf die Farbinformation zu haben. Somit können pro Farbkanal jeweils 7 Bit zur Codierung des Stiftzustandes genutzt werden, wodurch fast reines Weiß und fast reines Schwarz darstellbar ist. Grafikpixel können so auch durch einfache Luminanzumwandlung in ein für grafisch-taktile Displays passendes Format gewandelt werden und für die Darstellung für Sehende sogar direkt ohne Umwandlung genutzt werden. Auch für die Logging-Ausgabe ist diese Codierung sehr günstig, da so die Ausgabe für Sehende direkt erkennbar ist, wie Abbildung 107 zeigt. In dieser vergrößerten Log-Ausgabe im BrailleMap-Format sind gehobene Grafik-Stifte deutlich in Schwarz und Dunkelgrau zu erkennen. In der Umsetzung werden neben schwarz und weiß sechs vordefinierte Graustufen zur Verwendung angeboten. Schriftzeichen-Pixel sind aufgrund der enthaltenen Unicode-Informationen verschiedenfarbig. Dennoch sind auch dort die gehobenen Stifte deutlich von den gesenkten unterscheidbar.

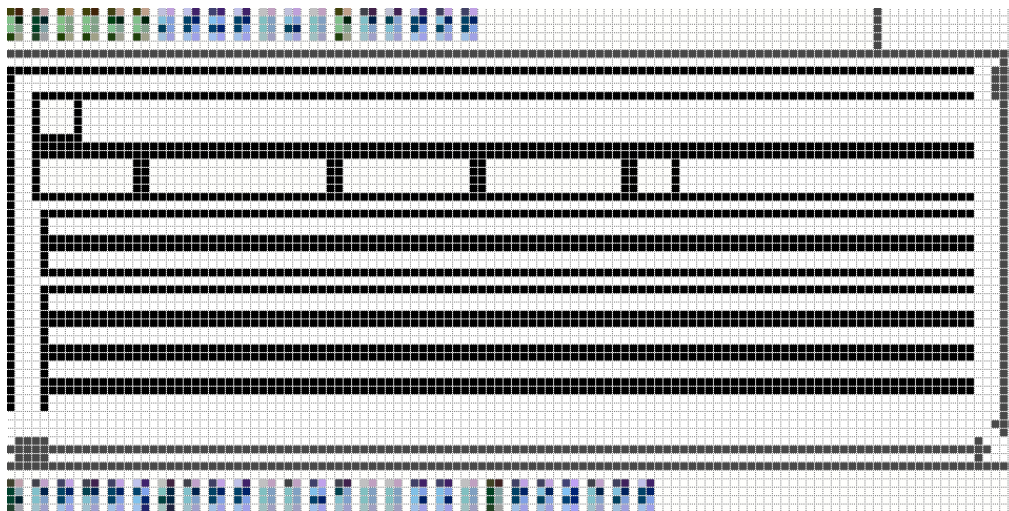


Abbildung 107 Darstellung einer Ausgabe des grafisch-taktilen Renderers im BrailleMap-Format

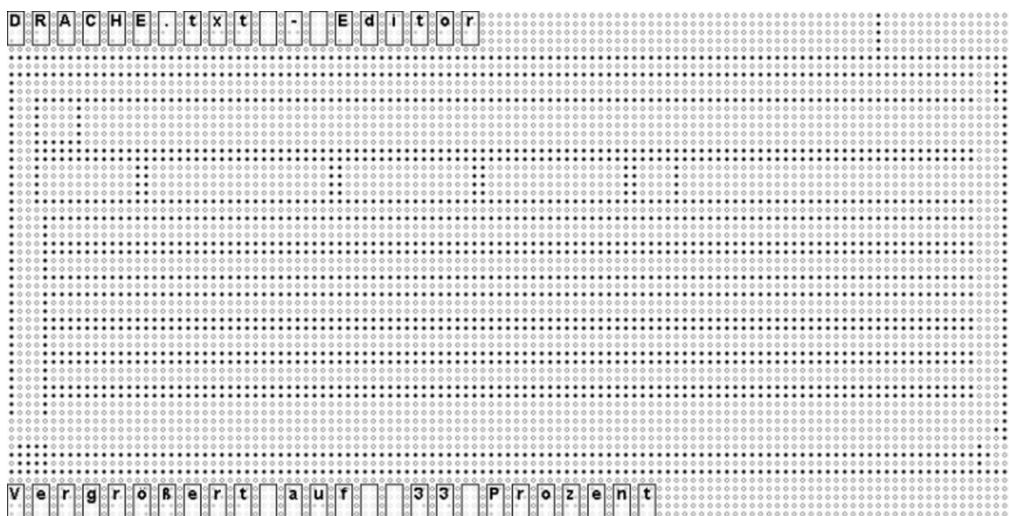


Abbildung 108 Darstellung der Renderer-Ausgabe aus Abbildung 107 in der Ansicht für Sehende

Solch eine vergrößerte Form der BrailleMap-Grafik wird in der Umsetzung direkt für die Darstellung für Sehende verwendet. Zur Annäherung an die tatsächliche Darstellung durch Stifte statt Pixel wird sie lediglich mit einer Textur aus Kreisen überzogen. Dies ergab sich als effizienteste Umsetzungsvariante für C# und .Net-Graphics. Dabei bleibt die Färbung der Schriftzeichen-Pixel zwar sichtbar. Da diese aber von der Schwarzschrift-Darstellung überdeckt werden, wirkt sich das nicht störend aus, wie Abbildung 108 verdeutlicht.

Zusammengefasst entstand aus den obigen Überlegungen folgende Codierung (Tabelle 11):

	Rot								Grün								Blau							
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Text	S	BT	BB	U15	U14	U13	U12	U11	S	BL	BR	U10	U9	U8	U7	U6	S	U5	U4	U3	U2	U1	U0	T
Grafik	F	F	F	F	F	F	F	B	F	F	F	F	F	F	F	B	F	F	F	F	F	F	F	T

Tabelle 11 Übersicht über die BrailleMap-Codierung.

Kürzel	Bit-Name	Erklärung	Werte
T	Text	Gibt an, ob der Pixel zu einem Braillezeichen gehört	1 für Text, 0 für Grafik
S	Stiftzustand	Gibt an, ob der Stift gehoben oder gesenkt ist	0 für gehoben, 1 für gesenkt
BT, BB, BL, BR	Border Top, Border Bottom, Border Left, Border Right	Gibt an, ob der Pixel zum Rand eines Braillezeichens gehört (BT für oberer Rand, BB für unterer Rand, BL für linker Rand und BR für rechter Rand)	0 für ja, 1 für nein
U0 - U15	Unicode Bit 0-15	Speichert den Unicode des Zeichens, zu dem der Pixel gehört	Je 0 oder 1 entsprechend dem Unicode
F	Farbe	Speichert die Farbe des Grafikpixels. (In der Umsetzung werden standardmäßig 8 Graustufen mit entsprechenden Farbwerten unterstützt.)	Je 0 oder 1 entsprechend der Farbe
B	Blinktyp	Gibt an, wie sich der Pixel beim Blinken verhält.	00: kein Blinken 01: gegenteilige Farbe zeigen 10: Stift heben 11: Stift senken

Tabelle 12 Legende zur BrailleMap-Codierung in Tabelle 11.

Die Bedeutung der Codierungs-Bits wurde insgesamt so gewählt, dass auch bei fälschlicher Verwendung von reinem Schwarz und reinem Weiß zum Zeichnen in einem BrailleMap keine ungewollten Effekte auftreten. Reines Schwarz, also ARGB=0x0000 erzeugt einen gehobenen, nicht blinkenden Grafik-Stift. Reines Weiß, also ARGB=0xFFFF erzeugt einen gesenkten Schriftzeichen-Stift ohne Randinformationen. Da der Stift nicht von anderen Stiften mit passendem Unicode und Randinformationen umgeben sein wird, wird er bei der Entfernung abgeschnittener Braillezeichen durch einen gesenkten Grafik-Stift ersetzt. Somit ist das Brail-

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

leMap-Format für einfache Grafikausgabe auf grafisch-taktilen Displays auch ohne besondere Kenntnisse leicht nutzbar und voll kompatibel zu einfachen Schwarz-Weiß-Grafiken, die sonst häufig zur Erzeugung von Ausgaben für grafisch-taktile Displays verwendet werden.

Bei anders gewählten Bedeutungen für die Codierungs-Bits würden Fehler auftreten. Wären zum Beispiel die Randinformationen jeweils mit „1 für ja“ codiert, so würde ein durch Weiß erzeugte Schriftzeichen-Stift nicht durch einen Grafik-Stift ersetzt werden und in der Darstellung für Sehende würde für jeden so entstandenen Pixel eine Überlagerung durch ein Schwarzschrift-Zeichen angezeigt werden. Wäre das Entscheidungs-Bit für Schrift- oder Grafik-Pixel umgekehrt codiert, würde durch reines Schwarz ein gehobener Schriftzeichen-Stift entstehen, der in der Darstellung für Sehende durch ein Schwarzschriftzeichen überlagert wäre oder bei anderer Codierung der Randinformationen sogar bei der Entfernung abgeschnittener Braillezeichen durch einen gesenkten Grafik-Stift ersetzt würde.

3.3.2.5 Darstellung der taktilen Ausgabe für Sehende

Wie bereits in Abschnitt 3.3.2.1 erwähnt, wurde für den HyperReader auch eine Anzeige der grafisch-taktilen Ausgabe als Grafikkanal des GraphicRenderers umgesetzt. Dabei wurde auf Grundlage des BrailleMap-Formats die in Abschnitt 3.2.5 beschriebene direkte Darstellung der ausgegebenen Texte in Schwarzschrift zusammen mit einigen Darstellungsoptionen umgesetzt. So kann der Betrachter entscheiden, ob die Schwarzschrift überhaupt angezeigt werden soll und wenn ja nur im Hintergrund der Stiftdarstellung oder diese überdeckend, um leichter erkennbar zu sein. Die Darstellung basiert auf der in 3.3.1.3 beschriebenen Schwellpapier-Darstellung. So können hier ebenfalls gesenkte Stifte durch nichtausgefüllte Kreise dargestellt oder ausgeblendet werden. Neben der Darstellung der grafisch-taktilen Ausgabe werden zudem die über das BrailleDis 9000 ermittelbaren Berührungspunkte des Benutzers wie in Abbildung 81 angezeigt.

Bei der Implementierung musste besonders darauf geachtet werden, das System durch die Ausgabe für Sehende wenig zu belasten (sowohl bezüglich des Speichers als auch der Leistung). Die Zusammenarbeit mit Sehenden soll die Arbeit des Blinden nicht beeinträchtigen. Der Blinde soll die Ausgabe Sehenden gerne zur Verfügung stellen. Um dies zu erreichen sollten bei einer Implementierung in C#, wie es für das Projekt HyperBraille vorgegeben war, die durch .Net bereitgestellten Zugriffsfunktionen der Klasse Bitmap vermieden werden. Dies betrifft nicht nur die Methoden `GetPixel` und `SetPixel`, sondern auch die Eigenschaften `Width` und `Height`. Zur Arbeit mit Bitmaps muss direkt auf den reinen Farbinformationen gearbeitet werden, die über das in Listing 16 gezeigte Vorgehen in ein Integer-Array und zurück in ein Bild geschrieben werden können. Die Informationen über Höhe und Breite eines Bildes sollten für Schleifendurchläufe in Variablen gespeichert werden.

```

/// <summary>Returns the colors of the images pixel in an int array
/// as 32 bit ARGB values.</summary>
int[] GetColorValues(Bitmap input) {
    Bitmap image;
    int width = input.Width, height = input.Height;
    byte bytesPerColorValue = 4;

    // convert image to 32bppArgb if necessary
    if (!PixelFormat.Format32bppArgb.Equals(input.PixelFormat)) {
        image = new Bitmap(width, height, PixelFormat.Format32bppArgb);
        Graphics.FromImage(image).DrawImage(input, 0, 0);
    } else {
        image = input; }

    // fast copy of pixel data from input image
    BitmapData bitmapData = image.LockBits(new Rectangle(0, 0, width,
        height), ImageLockMode.ReadOnly, image.PixelFormat);
    IntPtr pixelDataStartAddress = bitmapData.Scan0;
    int lineSize = bitmapData.Stride, byteCount = lineSize * height;
    int[] colorValues = new int[byteCount / bytesPerColorValue];
    System.Runtime.InteropServices.Marshal.Copy(pixelDataStartAddress,
        colorValues, 0, colorValues.Length);
    image.UnlockBits(bitmapData);
    return colorValues;
}

/// <summary>Writes the given color values in 32bppArgb format to a
/// bitmap of the given width and height</summary>
Bitmap WriteColorValues(int[] colorValues, int width, int height) {
    Bitmap image = new Bitmap(width, height, PixelFormat.Format32bppArgb);
    BitmapData bitmapData = image.LockBits(new Rectangle(0, 0, width,
        height), ImageLockMode.ReadWrite, image.PixelFormat);
    IntPtr pixelDataStartAddress = bitmapData.Scan0;
    int lineSize = bitmapData.Stride, byteCount = lineSize * height;
    System.Runtime.InteropServices.Marshal.Copy(colorValues, 0,
        pixelDataStartAddress, colorValues.Length);
    image.UnlockBits(bitmapData);
    return image;
}

```

Listing 16 C#-Methoden zum Kopieren von Farbdaten aus einem Bitmap in einen int[] und umgekehrt

3.3.2.6 Fazit

Das implementierte Framework setzt viele der in Abschnitt 3.2 vorgestellten Konzepte um, darunter die Filterung nach Farben, Exploration nach Helligkeit, verschiedene Zoomstrategien, Layoutoptionen, der Austausch von Schriften, die Markierung von Elementtypen und die Unterstützung der Kooperation mit Sehenden. Die Funktionalität der Umsetzung und Nützlichkeit der umgesetzten Konzepte konnte in verschiedensten Nutzertests von Projektpartnern gezeigt werden, die sich beispielsweise auf die Erkennbarkeit von Layoutfehlern in Worddokumenten mit Hilfe der Originalansicht aber auch die selbständige Durchführung der Konfiguration und Aspekte der Schrifterkennung bezogen. Einige Ergebnisse dazu wurden in [PNW10] veröffentlicht.

Weitere Konzepte, wie das Darstellen von Farben durch Blinken, die Anwendung von Texturen und die Steuerung des Ablaufs von Animationen in GIF-Dateien wurden zwar noch nicht

in die ContentRenderer integriert. Sie wurden aber durch kleinere Anwendungen, die auf Basis des Frameworks entwickelt wurden, umgesetzt. Eine Umsetzung in den ContentRenderern ist daher leicht möglich, muss aber auch durch die Interaktionskomponente des HyperReaders unterstützt werden. Auch die erweiterten Navigations- und Explorationstechniken konnten noch nicht in den HyperReader integriert werden, da hierzu erst eine entsprechende Erweiterung des Gesamt-Bedienkonzeptes nötig ist. Die Konfigurationsmöglichkeiten der verschiedenen ContentRenderer wurden von den Projektpartnern und insbesondere den Blinden selbst umfassend genutzt und als positiv bewertet. Die Erweiterungsmöglichkeiten durch Add-Ins bewährten sich bei der Umsetzung spezieller Darstellungen für die gefilterten Programme darunter Visio, PowerPoint und Excel durch andere Projektpartner.

3.3.3 HyperReader-Magnifier – Vergrößerung mit dem HyperBraille-Konzept

Für die Umsetzung der Vergrößerung auf Basis des HyperBraille-Darstellungskonzepts konnten viele Teile des im vorigen Abschnitt vorgestellten Rendering-Frameworks übernommen bzw. auf einfache Weise für eine visuelle Darstellung adaptiert werden. Dazu gehören die Dreiteilung des Rendering-Prozesses, die AddIn-Architektur und die Konfigurationsmöglichkeiten für Textverkürzungen, Strukturleiste und Layoutvorlagen. Der HyperReader-Magnifier wurde als eigenes Fenster umgesetzt, das am Bildschirm beliebig verschoben und in seiner Größe verändert werden kann. Auf diese Weise können leicht die durch aktuelle Bildschirmvergrößerungssoftware gebotenen Darstellungsmodi nachgestellt werden (siehe Abbildung 7). Da das Fenster die zu bedienende Oberfläche häufig überlagert, muss es, sofern die ursprüngliche Oberfläche bedient werden soll, gegenüber Mausaktionen durchlässig gesetzt werden, bevor der Mauszeiger automatisch zum zu bedienenden Objekt bewegt und die Aktion dort ausgeführt wird. Danach muss der Mauszeiger natürlich zurückgeführt werden, um den Benutzer nicht zu verwirren. Mit Hilfe des in Listing 17 gezeigten Quellcodes geschieht dieser Prozess, ohne dass er vom Benutzer wahrgenommen werden kann.

```

[DllImport("user32.dll")]
static extern bool SetLayeredWindowAttributes(...);
[DllImport("user32.dll")]
static extern int SetWindowLong(...);
[DllImport("user32.dll")]
static extern void mouse_event(...);

// constants for unmanaged methods
uint MouseEventFlags_LEFTDOWN = 0x00000002;
...
int GWL_ExStyle = -20, WSEX_Transparent = 0x20, WSEX_Layered = 0x80000;
uint LWA_Alpha = 0x2;
// some factors for calculation of mouse position
float MOUSE_RASTER_VAL = 65535f;
Rectangle screenBounds = Screen.PrimaryScreen.Bounds;
float widthRation = MOUSE_RASTER_VAL / (float)screenBounds.Width;
float heightRation = ...;

void DoLeftClick(int x, int y) {
    Point currentCursorPos = Cursor.Position;
    // Set form transparent to mouse event;
    SetWindowLong(Handle, GWL_ExStyle,
        InitialStyle | WSEX_Layered | WSEX_Transparent);
    SetLayeredWindowAttributes(Handle, 0, 255, LWA_Alpha);
    // move mouse pointer, do left click and move pointer back to start
    mouse_event(MouseEventFlags_MOVE | MouseEventFlags_ABSOLUTE,
        (uint)(x * widthRation), ..., 0, 0);
    mouse_event(MouseEventFlags_LEFTDOWN ..., ..., ..., 0, 0);
    mouse_event(MouseEventFlags_LEFTUP ..., ..., ..., 0, 0);
    Cursor.Position = currentCursorPos;
    // Set form opaque to mouse event;
    SetWindowLong(Handle, GWL_ExStyle, InitialStyle | WSEX_Layered);
    SetLayeredWindowAttributes(Handle, 0, 255, LWA_Alpha);
}

```

Listing 17 Quellcode für einen Linksklick durch ein Fenster hindurch

Auch die Erfassung der Originaldarstellung per Screenshot kann hier nicht angewendet werden. Dies kann aber, wie bei aktueller Bildschirmvergrößerungssoftware auch beispielsweise mit Hilfe der Magnification-API [Micb] gelöst werden. Hier zeigt sich auch, dass sich das Konzept des HyperReader-Magnifiers sehr gut für einen schrittweisen Umstieg von den aktuellen Bildschirmvergrößerungskonzepten zu neuen Darstellungen genutzt werden kann. Natürlich ist die Umsetzung der verschiedenen Ansichten für jede Anwendung sehr aufwändig. Auch wenn sich vieles durch allgemeine Definitionen lösen lässt, so muss es doch besonders bei komplexeren Anwendungen immer auch einige spezielle Umsetzungen geben. Wird eine aktuelle Bildschirmvergrößerungssoftware in die Originalansicht eingebunden, so entstehen dem Nutzer keine Nachteile, wenn noch nicht alle Anwendungen durch die anderen Ansichten unterstützt werden. Ähnliches wurde auch beim HyperReader (mit taktilem Ausgabe) durch die Einbindung des Screenreaders JAWS vorgegangen.

3.3.4 FirefoxZoom – Eine Umsetzung der partiellen Vergrößerung

Zur Demonstration des Konzepts der partiellen Vergrößerung in der Originaldarstellung wurde im Rahmen dieser Arbeit zunächst eine prototypische Umsetzung als Extension für den Webbrowser Mozilla Firefox umgesetzt [TE08]. Diese Umsetzung wurde im Rahmen einer Studienarbeit verbessert und erweitert [Pet09]. Die dabei entstandene Extension steht zum Download unter <http://www.vis.uni-stuttgart.de/~taras/FirefoxZoom.html> bereit. Sie soll vor allem zur Evaluierung des Konzeptes dienen. Dabei lag der Fokus auf der Vergrößerung der Darstellung. Weitere Anpassungen der Darstellung, wie etwa Änderungen der farblichen Darstellungen, wurden nur eingeschränkt betrachtet. Mozilla Firefox eignete sich zur Umsetzung des Konzeptes besonders, da dessen Oberfläche in XUL [Mozc] geschrieben ist und sich der Darstellungsprozess somit durch JavaScript und CSS leicht beeinflussen lässt.

Grundlage der Vergrößerung ist das Setzen des Schriftgröße-Attributs (`font-size`), wobei Elemente, die bereits vor der Vergrößerung eine höhere Schriftgröße besitzen als die vom Benutzer vorgegebene Mindestschriftgröße, durch die Vergrößerung nicht beeinflusst werden. Elemente wie Icons, Schaltflächen und Eingabefelder werden durch Setzen der Größen-Attribute (`width` und `height`) passend skaliert bzw. in ihrer Ausdehnung begrenzt. Da semantische Zusammenhänge nicht direkt aus der Benutzungsoberfläche bzw. der dargestellten Webseiten ermittelbar sind, basiert die partielle Vergrößerung meist auf strukturellen Zusammenhängen. An einigen Stellen wurde zudem das Wissen der Autoren über den semantischen Aufbau der Oberfläche in die Vergrößerungsalgorithmen eingepflegt.

Elemente der Benutzungsoberfläche, die direkt sichtbar sind (Menüleisten, Werkzeugleisten etc.), werden vergrößert, sobald der Benutzer eine kurze Zeit mit dem Mauszeiger über ihnen verweilt. Die Vergrößerung erfolgt nicht sofort, da sonst bei Bewegungen der Maus ständig Vergrößerungen und Rücksetzungen stattfinden würden, was in ein Flackern ausartet. Verlässt der Mauszeiger die Elemente wieder, so werden die vergrößerten Elemente auch nicht immer sofort zurückgesetzt. Wurde beispielsweise die Menüleiste vergrößert, so bleibt diese bei Bewegungen des Mauszeigers nach unten noch vergrößert, da es dem Benutzer sonst nicht möglich wäre, die darunter liegende Leiste zu treffen. Durch die Vergrößerung der Menüleiste sind die darunterliegenden Elemente nach unten verschoben. Sobald sich die Menüleiste verkleinert, schieben sich alle darunterliegenden Elemente wieder nach oben. Bei stärkerer Vergrößerung können sich die Positionen von Elementen so stark verschieben, dass an Stelle einer Leiste, die sich zuvor direkt unter der vergrößerten Menüleiste befand, eine völlig andere Leiste dargestellt ist. Menüleiste und Werkzeugleisten werden in solch einem Fall erst wieder verkleinert, wenn der Benutzer diesen gesamten Bereich verlässt.

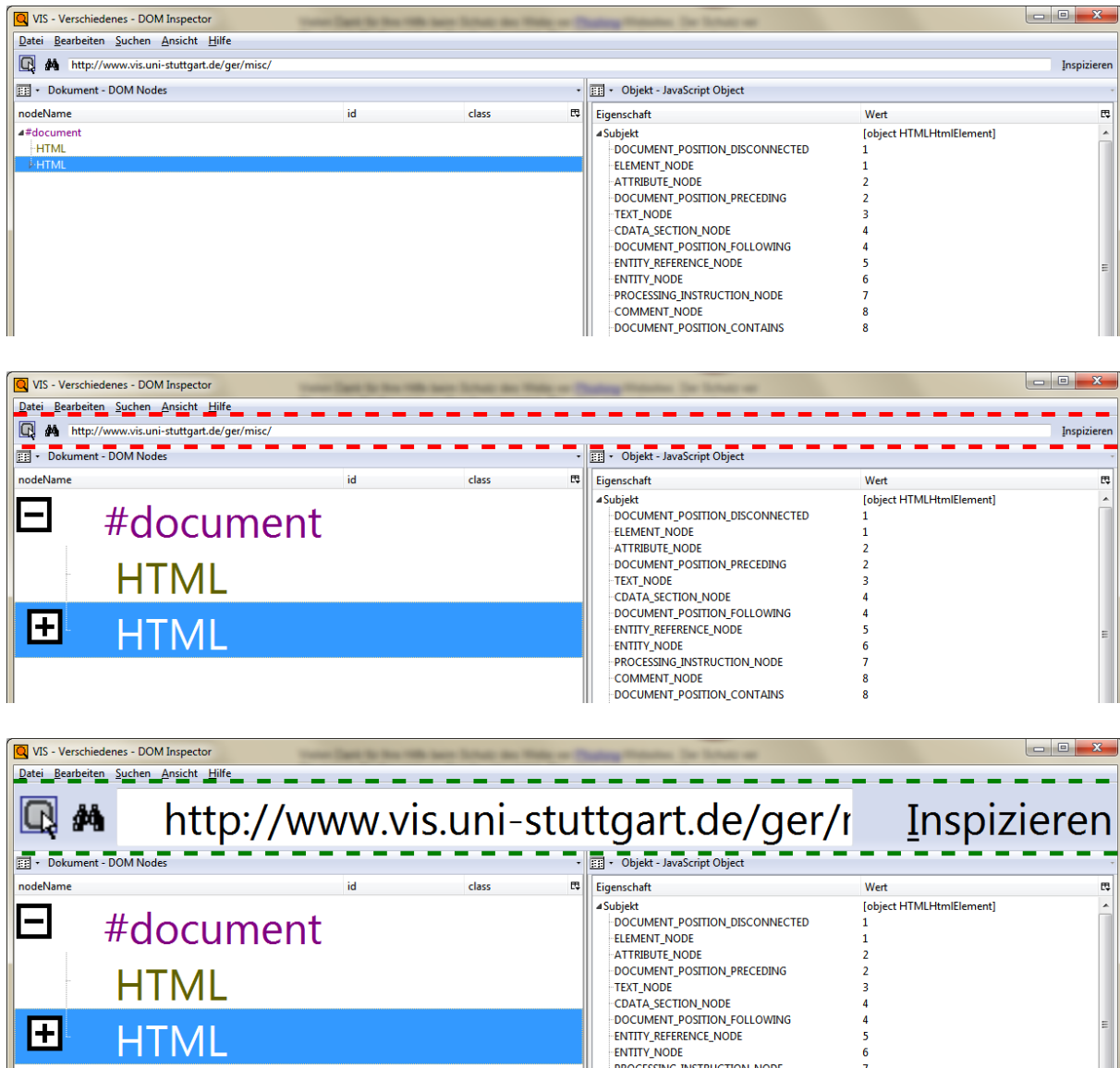


Abbildung 109 Ablauf der benutzergesteuerten Vergrößerung am Beispiel des DOM-Inspectors
 v. o. n. u.: Originaldarstellung, nach Vergrößerung des linken Bereichs und bei Auswahl des zweiten Bereichs zur Vergrößerung (mit rotem, gesticheltem Rahmen markiert), nach Vergrößerung des zweiten Bereichs und bei Auswahl des zweiten Bereichs zum Rücksetzen (mit grünem, gesticheltem Rahmen markiert); der Mauszeiger befindet sich jeweils über den markierten Bereichen

Für größere Dialoge und Webseiten wurde eine stärker benutzergesteuerte Vergrößerung umgesetzt. Überfährt der Benutzer einen Dialog oder eine Webseite, so werden zunächst die jeweils zusammenhängenden Bereiche durch einen Rahmen markiert. Der Benutzer kann solch einen Bereich dann per Tastendruck vergrößern. Das Rücksetzen erfolgt ebenfalls aktiv durch den Benutzer. Abbildung 109 illustriert dies. So können leicht mehrere Bereiche gleichzeitig vergrößert werden, wodurch ähnliche Darstellungen wie in Abbildung 42 erreicht werden können. In Dialogen kann die gewählte Vergrößerung auch gespeichert und bei erneutem Erscheinen des Dialogs wieder hergestellt werden. Dies erfolgt mit Hilfe der in Firefox verfügbaren Properties. In Dialogen war die partielle Vergrößerung in der Originaldarstellung auf diese Weise gut umsetzbar. In Webseiten zeigte sich allerdings deutlich die Abhängigkeit von einer guten Strukturierung und flexiblen Layouts. Häufig konnten nur schlechte

3 Angepasste Darstellungen und erweiterte Interaktionstechniken

Ergebnisse erzielt werden. Da moderne Webseiten aber oft ausreichend Leerräume aufweisen, die ohne Informationsverlust durch ein weiteres Fenster überlagert werden können, wurde auch eine indirekte Vergrößerung mit sogenannten Lupen umgesetzt (siehe Abbildung 110). In diesen ist eine freie Wahl der Schriftgestaltung möglich und Text wird passend zur Breite der Lupe umgebrochen, so dass kein horizontales Scrollen nötig ist. Für die Lupen wurde die Benutzungsschnittstelle der „Semantic Lense“ von Rotard et al. aufgegriffen und erweitert [RGE07]. Jede neue Lupe besitzt eine eigene Rahmenfarbe, über die der in der Lupe vergrößerte Bereich in der Webseite wiedergefunden werden kann. Verweilt der Benutzer eine kurze Zeit mit dem Mauszeiger über einem Bereich der Webseite, so wird dieser vergrößert in die gerade aktive Lupe geladen und mit der Rahmenfarbe der Lupe markiert. Da der gesamte Inhalt des Bereiches aus dem DOM-Baum der Webseite in die Lupe kopiert wird, verhalten sich auch Eingabefelder, Links und andere Bedienelemente innerhalb der Lupe wie im originalen Abschnitt.



Abbildung 110 Screenshot des Firefox-Prototypen zur partiellen Vergrößerung mit Lupe

Die ersten Erfahrungen bei der Anwendung der partiellen Vergrößerung in der Originaldarstellung zeigen, dass diese deutliche Vorteile gegenüber den etablierten Techniken bringt. Die Übersicht über den Bildschirminhalt bleibt erhalten und dem Benutzer entgehen keine Informationen. Alle aufpoppenden Dialoge, Menüs und Tooltips sind sichtbar und werden auch vergrößert. Die Vergrößerung basiert nicht auf einem Vergrößerungsfaktor, sondern auf einer vom Benutzer vorgegebenen Mindestschriftgröße. Dabei werden auch Icons passend zur Änderung der Schriftgröße skaliert. Da die Vergrößerung der Oberflächenelemente nicht permanent besteht, bleibt für das Anwendungsdokument genauso viel Platz wie ohne eine Vergrößerungstechnik. Bei der Umsetzung muss besonders darauf geachtet werden, dass durch die selektive Vergrößerung kein störendes Flackern durch einen ständigen Wech-

sel von vergrößertem und unvergrößertem Zustand eines Elementes erzeugt wird. Zudem ist die benutzergesteuerte Vergrößerung natürlich gewöhnungsbedürftig und erfordert bei starker Verschachtelung Fingerspitzengefühl. Dies könnte vereinfacht werden, wenn sich die einzelnen Regionen, die vergrößert werden können, durch die Tastatur ansteuern ließen. Dazu sollte eine wie in Abschnitt 3.2.2.5 beschriebene erweiterte Navigation verwendet werden. Zudem sollte eine Art Vorschaufunktion evaluiert werden, bei der direkt eine Vergrößerung des gerade anvisierten Bereichs stattfindet, allerdings nur so, dass die Größe des Bereichs nicht verändert wird. Zumindest der Beginn eines Bereichs wäre somit sofort erkennbar. Es würde aber kein Flackern auftreten. Um nicht nur den Beginn zu zeigen, könnten auch Textverkürzungen eingesetzt werden, die für kleine Displays wie etwa von Smartphones entwickelt wurden (siehe beispielsweise [LB05]).

In der aktuellen Implementierung von FirefoxZoom werden fast alle der in Abschnitt 3.1.2 aufgezeigten Anforderungen umgesetzt. Einschränkungen bestehen größtenteils bei dem Umgang mit Farben. So können für die Benutzungsoberfläche noch keine veränderten Farbeinstellungen genutzt werden, außer natürlich über die Design-Einstellungen des Betriebssystems. Die Änderung der farblichen Gestaltung wäre zwar problemlos machbar. Allerdings wäre es schwierig, dem Benutzer wie in Abschnitt 3.2.3 beschrieben, die Originalfarbe zugänglich zu machen. Auch eine Änderung der Darstellung von Bildern findet bisher nicht statt. Zwar werden Icons an die Schriftgrößenänderung angepasst und Bilder in Webseiten werden je nach Benutzereinstellung vergrößert. Eine weitere Bearbeitung fehlt aber bisher. Sie könnte wahrscheinlich über die Integration einer C++-Komponente in die Extension integriert werden. Weiterhin konnte die Umgestaltung der Bedienelemente zur besseren Erkennbarkeit in der gewählten Technologie nicht zufriedenstellend umgesetzt werden. Auch Mauszeiger und Textcursor lassen sich nicht beliebig gestalten, werden aber zumindest an die Größe der Schrift angepasst. Die Auswirkungen hervorgehobener Cursor können aber hinreichend mit den aktuellen Bildschirmvergrößerungsprogrammen evaluiert werden.

Die implementierte Extension zeigt, dass das Konzept der partiellen Vergrößerung in der Originaldarstellung umsetzbar ist, und kann zur weiteren Evaluierung und Verbesserung des Konzepts eingesetzt werden. Durch eine Untersuchung der Umsetzungsmöglichkeiten in anderen Technologien zeigte sich allerdings deutlich, dass das Konzept nicht in naher Zukunft für alle Desktop-Anwendungen umsetzbar ist, da viele Technologien nicht auf diese Weise beeinflusst werden können. Zudem müsste das Konzept für die verschiedenen Technologien einzeln umgesetzt werden. Meist ist dazu eine Integration in die grundlegenden Frameworks, mit denen die Oberflächen erstellt werden, nötig. Eine globale Lösung ist bei den derzeitigen Technologien nicht möglich, da meist keine nachträgliche Beeinflussung der Darstellung möglich ist. Solche separaten Umsetzungen bedeuten neben viel Arbeitsaufwand allerdings auch, dass eventuell verschiedene Bedienkonzepte pro Technologie verwendet werden, was für den Benutzer sehr nachteilig ist.

3.4 Zusammenfassung

In diesem Kapitel wurde gezeigt, wie stark die Erstellung angepasster Darstellungen für Blinde und Sehbehinderte zusammenhängen und welche Probleme bei beiden Gruppen auftreten. Die Konzepte und Umsetzungsbeispiele zeigen deutlich, dass noch enorme Verbesserungen möglich sind. Dabei kann eine gemeinsame Plattform, wie beispielsweise der Hyper-Reader, für beide Gruppen Vorteile bringen, da wesentliche Komponenten nur einmal entwickelt werden müssen.

4 Förderung der zugänglichen Gestaltung

Wie bereits in Kapitel 2.2 angesprochen wurde, ist eine gute Strukturierung und Annotation von Dokumenten und Bedienoberflächen essentiell für deren Zugänglichkeit. Viele der im vorhergehenden Kapitel aufgeführten Konzepte sind nur umsetzbar, wenn die darzustellenden Inhalte gut zugänglich sind. Häufig trifft man allerdings auf Programme und Dokumente, die diese Voraussetzung nicht oder zumindest nicht vollständig erfüllen. Dies liegt im Wesentlichen an der mangelnden Unterstützung in Werkzeugen. Weder unterstützen Editoren ausreichend bei der Erstellung zugänglicher Materialien, noch belohnen darstellende Programme eine zugängliche Gestaltung ausreichend, um für den normalen Nutzer einen erkennbaren Mehrwert zu bringen. Mitunter vernichten verarbeitende Programme sogar Zugänglichkeitsinformationen, auch wenn diese erhalten bleiben könnten. Im Folgenden wird die bestehende Problematik an einigen Beispielen verdeutlicht. Lösungskonzepte und eine konkrete Umsetzung zur Verbesserung der Gestaltung von SVG werden vorgestellt. Zuletzt werden allgemeine Richtlinien zur Förderung der zugänglichen Gestaltung formuliert.

4.1 Darstellung der Problematik

Besonders häufig fallen Probleme mit fehlender Struktur und fehlenden oder nicht auffindbaren Beschreibungen bei Dialogen auf. Beschreibungen zu Elementen sind zwar oft genügend vorhanden, wurden aber meist nur visuell in der Nähe eines Elements positioniert und nicht mit dem Element verknüpft. Für Screenreader-Nutzer kann dann keine Verbindung zwischen Beschreibung und Element hergestellt werden. Teilweise hilft hier zwar die Arbeit im sogenannten Flächenmodus, bei dem Elemente entsprechend ihrer visuellen Anordnung in Zeilen dargestellt werden. Bei einem Dialog, wie etwa dem in Abbildung 111, können dabei aber auch Elemente zusammen dargestellt werden, die nicht zusammen gehören.

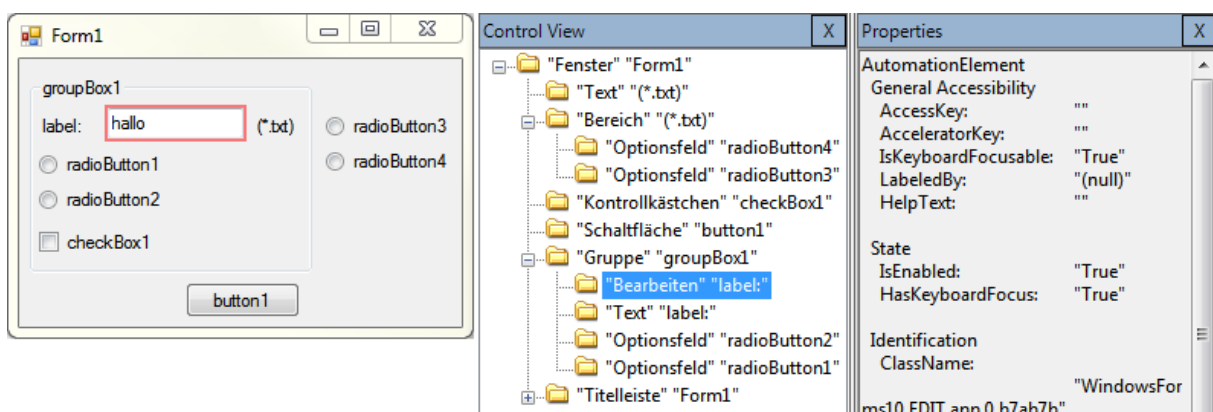


Abbildung 111 Ein per Drag&Drop in VisualStudio erstellter Beispieldialog mit Ausgabe von UISpy

Abbildung 111 zeigt einen Dialog, der mit dem GUI-Editor von Visual Studio per Drag&Drop gestaltet wurde, so wie es die meisten GUI-Ersteller tun würden. Dabei wurden unbewusst mehrere Probleme erzeugt, wie der Ausschnitt aus der Darstellung in UISpy [uis] zeigt, welches den UIAutomation-Baum einer Anwendung darstellt, also die Informationen, die auch

4 Förderung der zugänglichen Gestaltung

Screenreadern zur Verfügung stehen. So ist das Kontrollkästchen „checkbox1“, obwohl visuell eindeutig in der Gruppe „groupBox1“ befindlich, dieser strukturell nicht zugeordnet. Dies geschah, da das Kontrollkästchen zunächst unter der Gruppe positioniert war und der Gruppenrahmen dann einfach um das Element erweitert wurde. Die Beschriftung „label:“ ist dem Eingabefeld nur halb zugeordnet. Zwar trägt das Eingabefeld den Namen „label:“ (siehe Markierung im UISpy-Screenshot) aber das Attribut „LabeledBy“ ist nicht mit dem Beschriftungs-Element besetzt. Dies entstand, da das Eingabefeld zunächst an dem schon eingefügten Label ausgerichtet wurde, wodurch automatisch eine Zuordnung geschieht. Danach wurde aber das Label „(*.txt)“ am Eingabefeld ausgerichtet, wodurch eine weitere Zuordnung stattfand. Als später das Optionsfeld „radioButton3“ an der Beschriftung „(*.txt)“ ausgerichtet wurde, passierte auch hier eine Zuordnung, was zum Entstehen des völlig irreführenden „Bereich (*.txt)“ führte. Dafür ging die Verbindung zum Eingabefeld verloren. Betrachtet man nur den UIAutomation-Baum, ist es recht schwer den Dialog zu erfassen. Dabei ist der Erstellungsvorgang durchaus als realistisch zu betrachten. Viele ähnliche Beispiele lassen sich in Dialogen verschiedenster Anwendungen finden (beispielsweise der Dialog zum Konfigurieren von Absatz-Einstellungen in Word oder auch der Standard-Drucken-Dialog in Windows). Der Flächenmodus des Screenreaders kann hier zwar zur Verknüpfung der Beschriftungen „label:“ und „(*.txt)“ mit dem Eingabefeld hilfreich sein, da alle diese Elemente gleichzeitig auf der Braillezeile angezeigt werden würden. Er würde aber auch die Elemente „radioButton1“ und „radioButton4“ in einen Zusammenhang stellen, der vom logischen Aufbau des Dialogs der gar nicht vorgesehen ist.

Neben den für Screenreader problematischen strukturellen Fehlern entstehen bei durch Drag&Drop erstellten Dialogen meist auch layoutbezogene Probleme, die die Verwendung der Dialoge bei veränderter Schriftgröße beeinträchtigen (siehe Abbildung 8). Häufig werden in solchen Dialogen absolute Positionen für die eingefügten Elemente gespeichert, auch wenn der Nutzer diese mit Hilfe von Ausrichtungslinien positioniert hat. Eine wesentliche Verbesserung können hier GUI-Generatoren bringen, die Dialoge aufgrund semantischer Modelle erstellen und so auch leicht an verschiedene Gegebenheiten anpassen können, wie beispielsweise SUPPLE [GW04]. Allerdings sind solche Generatoren für normale Nutzer oft nicht leicht handhabbar, wie die Untersuchung in [SRK+10] zeigt. Die meisten Nutzer wollen ihre Gedanken direkt visualisieren, anstatt sie zunächst in abstrakten Modellen zu formulieren. Es müssen also eher Verbesserungen in die Drag&Drop-Editoren integriert werden.

Ein hervorstechendes Beispiel dafür, wie wenig Zugänglichkeitspotentiale genutzt werden, ist SVG. Hier zeigt sich deutlich, dass das Vorhandensein von Strukturelementen und Attributen zur Erstellung zugänglicher Materialien allein nicht ausreicht, sogar wenn sie nicht wie meist erst nach und nach eingeführt wurden, sondern von vornherein festgelegt waren. Natürlich ist es zunächst einmal positiv, dass immer mehr SVG-Grafiken Einzug ins Web finden. So gerät dieses für die Zugänglichkeit grafischer Darstellungen wertvolle Format nicht in Vergessenheit. Zwar wurde SVG bereits häufig totgesagt. Durch die wachsende Anzahl von SVG-Editoren oder Editoren mit Exportmöglichkeiten nach SVG, die ständige Verbesserung der Darstellung durch Webbrowser und vor allem die SVG-Initiative von Wikipedia [Wik10c]

erfreut sich SVG seit einiger Zeit aber wieder zunehmender Beliebtheit. Das größte Argument gegen den Einsatz von SVG war bislang die mangelnde oder von zusätzlichen Modulen abhängige Darstellbarkeit in älteren Browsern. Wikipedia bzw. die zugrunde liegende Software MediaWiki bietet hier eine gute Lösung. Die Autoren von Seiten können Grafiken im SVG-Format hochladen und angeben, in welcher Größe sie präsentiert werden sollen. Zur Darstellung in der Webseite werden die SVG-Grafiken serverseitig, automatisch in Rastergrafiken gewandelt und inklusive eines Links zur Originaldatei in die Webseite eingebunden. SVG-Dateien sind somit für den Autor nützlich, da er sie leicht in verschiedenen Webseiten in verschiedenen Auflösungen nutzen kann. Gleichzeitig sind sie über den Link für Aufbereitungen für Sehbehinderte und Blinde abrufbar. Betrachtet man die bereitgestellten SVGs allerdings näher, so stellt man fest, dass nur wenige davon wirklich als zugänglich bezeichnet werden können. Nur selten sind Annotationen und eine gute Strukturierung zu finden, welche aber für Aufbereitungen, wie sie beispielsweise [AW08] und [ROE04] zeigten, notwendig sind. Abbildung 112 zeigt zwei typische Beispiele für SVG-Grafiken aus dem Web.

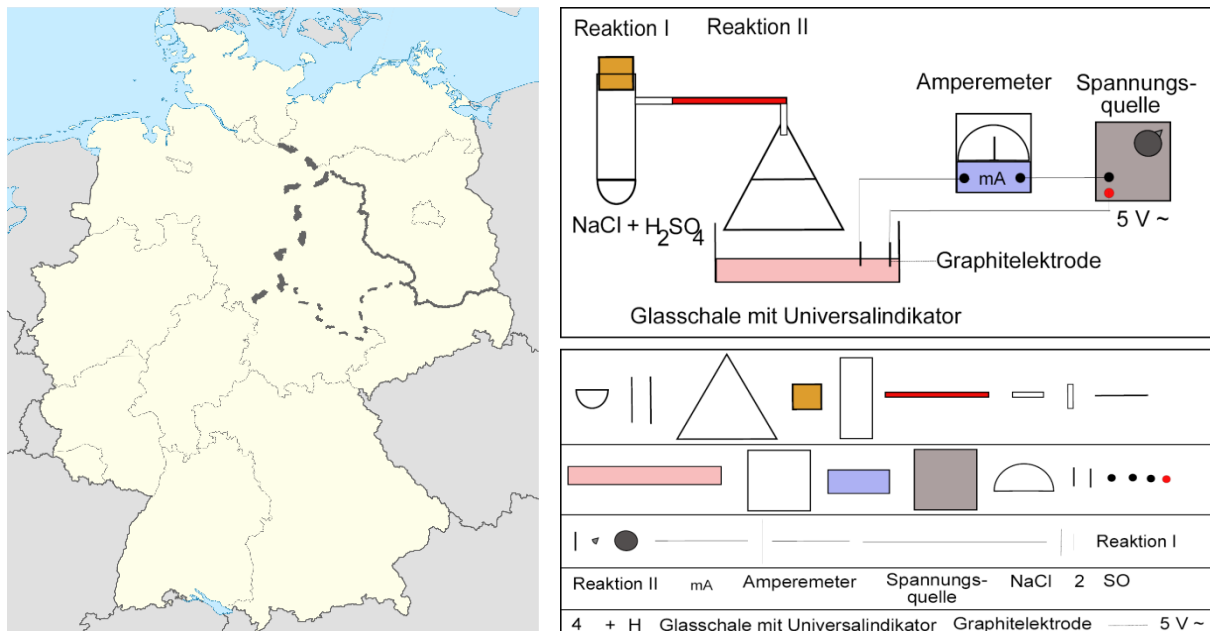


Abbildung 112 Beispiele für (schlecht zugängliche) SVG aus dem Web¹⁵

links: Deutschlandkarte mit Hervorhebung von drei Pfad-Elementen, die u. a. die Grenzen von Sachsen-Anhalt bilden; rechts oben: Darstellung eines Versuchsaufbaus aus der Chemie; rechts unten: die Elemente des Versuchsaufbaus in der Reihenfolge ihres Auftretens in der Quelldatei

In der Deutschlandkarte (links), die sogar für die Verwendung in Wikipedia empfohlen wird, sind nicht, wie man vermuten könnte, die einzelnen Bundesländer eigenständige Elemente der SVG-Grafik, sondern ergeben sich nur aus Grenzlinien, wie die hervorgehobenen Linien in der Abbildung zeigen. Diese sind nicht einmal so gestaltet, dass damit die Bundesländer jeweils einzeln dargestellt werden könnten. Sie erstrecken sich über mehrere Bundeslandgrenzen. In der Darstellung des Versuchsaufbaus (Abbildung 112 Mitte) wurde keinerlei

¹⁵ Bildquellen: <http://commons.wikimedia.org/wiki/File:Deutschland.svg>,
http://de.wikibooks.org/wiki/Anorganische_Chemie_für_Schüler:_Säure_-_Basen_-_Reaktionen

4 Förderung der zugänglichen Gestaltung

Gruppierung auf die Elemente angewendet. Auch die Reihenfolge, in der die Elemente im XML-Code auftauchen, ist nicht sinnvoll, wie Abbildung 112 (rechts) zeigt

Dabei wurde bereits bei der Entwicklung von SVG speziell darauf geachtet, dass ein Grafikformat entsteht, welches die Zugänglichkeit von Grafiken erhöht. Die Zielsetzung des World Wide Web Consortium (W3C) ist eindeutig: „Publish highly-structured documents, not just graphical representations“¹⁶. Die schon in Kapitel 2.2 dargestellten Elemente zur Strukturierung und Annotation waren von Anfang an in SVG definiert. Der mangelnde Einsatz begründet sich also nicht darin, dass das Format mehrfach geändert wurde und bestehende Grafiken noch nicht aktualisiert wurden. Vielmehr liegt es daran, dass weder eine gut durchdachte Struktur noch Annotationen einen Einfluss auf die visuelle Darstellung der SVG-Grafiken haben. Diese ist aber entscheidend für den Großteil der Webseiten-Ersteller und -Benutzer, weshalb Entwickler von SVG-Editoren sich auch fast ausschließlich auf diesen Aspekt konzentrieren. Dadurch wiederum wird es auch gewillten SVG-Autoren schwer gemacht, gute (also sinnvoll strukturierte und annotierte) SVG-Grafiken zu erstellen. Wie [TSSE09] zeigt, muss dazu meist der XML-Quellcode direkt editiert werden.

Selbst Anwendungen, die die Möglichkeit bieten, SVG automatisch aus anderen Daten zu erzeugen, die semantische Informationen in sich tragen und bereits gruppiert sind, erzeugen häufig nur schlechte SVG-Dateien. Dabei wäre hier häufig nicht einmal Mehrarbeit nötig gewesen, wenn bei der Entwicklung des SVG-Exports einfach von vornherein die Strukturen aus den Anwendungsdaten erhalten worden wären. Dies zeigt wiederum, dass sich Entwickler grafischer Oberflächen dem Nutzen strukturierter und annotierter Daten nicht immer bewusst sind. Ein Beispiel dafür ist das durchaus beliebte Programm „Dia“ (siehe <http://live.gnome.org/Dia>). Listing 18 zeigt einen Auszug aus der Dia-Datei, die das UML-Diagramm mit zwei Klassen aus Abbildung 113 beschreibt. In der Datei ist klar ersichtlich, dass beiden Klassen durch Vererbung miteinander verbunden sind. Listing 19 zeigt den mit Dia erzeugten SVG Export. Darin sind keinerlei Annotationen zu finden, die dem Betrachter beim Verständnis des Diagramms helfen könnten, obwohl im Original-Datenformat alle nötigen Informationen vorhanden sind. Zudem ist auch die Aufteilung in grafische Elemente nicht gelungen, da jeweils die Füllung und der Rahmen eines Elements in zwei SVG-Elementen abgebildet sind. Dies verlängert eine Exploration unnötig und verlangsamt auch das Rendering der SVG-Grafik. Zumindest wurden Gruppen für die drei verschiedenen Elemente angelegt. Dies ist bei vielen SVG-Generierungen nicht einmal der Fall wie beispielsweise bei dem Funktionsplotter von <http://www.mathmlcentral.com/>.

¹⁶ <http://www.w3.org/TR/2003/REC-SVG11-20030114/access.html>

```

<?xml version="1.0" encoding="UTF-8"?>
<dia:diagram xmlns:dia="http://www.lysator.liu.se/~alla/dia/">
  <dia:diagramdata>
    ...
  </dia:diagramdata>
  <dia:layer name="Hintergrund" visible="true" active="true">
    <dia:object type="UML - Class" version="0" id="00">
      ...
      <dia:attribute name="name">
        <dia:string>#Klasse1#</dia:string></dia:attribute>
      <dia:attribute name="stereotype">
        <dia:string>##</dia:string></dia:attribute>
      <dia:attribute name="comment">
        <dia:string>##</dia:string></dia:attribute>
      <dia:attribute name="abstract">
        <dia:boolean val="false"/></dia:attribute>
      ...
    </dia:object>
    <dia:object type="UML - Class" version="0" id="01">
      ...
      <dia:attribute name="name">
        <dia:string>#Klasse2#</dia:string></dia:attribute>
      ...
    </dia:object>
    <dia:object type="UML - Generalization" version="1" id="02">
      ...
      <dia:connections>
        <dia:connection handle="0" to="00" connection="8"/>
        <dia:connection handle="1" to="01" connection="1"/>
      </dia:connections>
    </dia:object>
  </dia:layer>
</dia:diagram>

```

Listing 18 Auszug aus einer Dia-Datei (ein UML-Diagramm mit zwei Klassen und Vererbungsfeil)

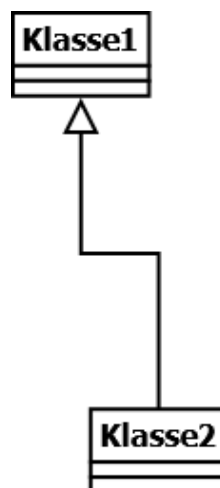


Abbildung 113 Grafische Darstellung der Dia-Datei aus Listing 18

4 Förderung der zugänglichen Gestaltung

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/PR-SVG-20010719/DTD/svg10.dtd">
<svg width="6cm" height="13cm" viewBox="108 89 115 256"
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <g>
    <rect style="fill: #ffffff"
      x="109" y="90" width="71.8" height="28"/>
    <rect style="fill: none; fill-opacity:0; stroke-width: 2;
      stroke: #000000" x="109" y="90" width="71.8" height="28"/>
    <text style="fill: #000000;text-anchor:middle;
      font-size:16;font-family:sanserif;font-style:normal;
      font-weight:700" x="144.9" y="110">Klasse1</text>
    <rect style="fill: #ffffff" .../>
    <rect style="fill: none; ... stroke: #000000" .../>
    <rect style="fill: #ffffff" .../>
    <rect style="fill: none; ... stroke: #000000" .../>
  </g>
  <g>
    <rect style="fill: #ffffff" .../>
    <rect style="fill: none; ... stroke: #000000" .../>
    <text ...>Klasse2</text>
    <rect style="fill: #ffffff" .../>
    <rect style="fill: none; ... stroke: #000000" .../>
    <rect style="fill: #ffffff" .../>
    <rect style="fill: none; ... stroke: #000000" .../>
  </g>
  <g>
    <polyline style="fill: none; fill-opacity:0; stroke-width: 2;
      stroke: #000000" points="144.9,153.242 144.9,217.503
      185.9,217.503 185.9,300 "/>
    <polygon style="fill: #ffffff" points="... "/>
    <polygon style="fill: none;... stroke: #000000" points="... "/>
  </g>
</svg>
```

Listing 19 Auszug aus der durch Dia erzeugten SVG-Datei zu Listing 18

Solch ein Verlust an strukturellen und semantischen Informationen findet sogar bei Konvertierungen zwischen seit langem etablierten Dokumentformaten statt. Beispielsweise werden Listen aus Word (wie die in Abbildung 114) in HTML nicht etwa mit einem HTML-Listenelement dargestellt, sondern durch mehrere Absätze, die das Aufzählungszeichen als Text enthalten mit festen Leerzeichen als Abstand zum Text des Listenpunkts, wie Listing 20 zeigt.

- Listenpunkt 1
 - o Unterpunkt 1a
 - o Unterpunkt 1b
- Listenpunkt 2

Abbildung 114 Abbildung einer Liste aus einer Word-Datei

4.2 Lösungskonzepte

Zugängliche Gestaltung kann mit verschiedenen Mitteln unterstützt werden, wie die folgenden Beispiele zeigen. Vor allem müssen Autoren motiviert werden, Strukturierung und Annotation gut anzuwenden. Dies kann direkt, durch Belohnung der eventuell entstandenen Mehrarbeit geschehen und indirekt durch Vereinfachung der Nutzung entsprechender Konstrukte und Erschwerung der Nutzung schlechter Konstrukte. Grundlage muss natürlich immer eine Analyse der Anforderungen der Betroffenen sein. Solche Analysen können nur von entsprechenden Institutionen durchgeführt werden. Stellen diese Ihre Ergebnisse mit erläuternden Erklärungen, entsprechenden Richtlinien und Umsetzungshinweisen bereit, so wie dies beispielsweise beim W3C durch die WCAG geschieht, ist ein wichtiger Schritt getan. Allerdings darf nicht erwartet werden, dass diese Dokumente von einem Großteil der Autoren von Oberflächen und Dokumenten gelesen und beachtet wird. Auch die Nachbearbeitung jeglicher verfügbaren Dokumente und Oberflächen durch speziell geschulte Personen ist nicht möglich. Der Aufwand dafür wäre kaum zu rechtfertigen, vor allem unter dem Gesichtspunkt, dass die wenigsten direkt von zugänglicher Gestaltung betroffen sind. Vielmehr müssen die Richtlinien in Werkzeuge einfließen, die bereits verbreitet zur Erstellung und Betrachtung von Oberflächen und Dokumenten zum Einsatz kommen.

4.2.1 Alternative Darstellungen zu Rastergrafiken

Rastergrafiken sind das zur Übertragung von statischen Bildinformationen am häufigsten eingesetzte Mittel, da sie meist leicht zu erzeugen, zu übertragen und in andere Dokumente oder Bedienoberflächen einzubinden sind und zuverlässig angezeigt werden. Für den Zugang Blinder und Sehbehinderte zu den, in Rastergrafiken ausgedrückten Informationen, sind textuelle Alternativen das wesentliche Mittel. Auch falls die Verbreitung von grafisch-taktilen Displays deutlich zunimmt, wird sich dies nicht ändern. Die Exploration einer Grafik wird auch dann noch zeitaufwändig sein. Eine textuelle Alternative mit Informationen zum Inhalt der Grafik kann diese einerseits erleichtern und andererseits dem Benutzer Hinweise geben, ob sich eine Exploration überhaupt lohnt.

Bei der Gestaltung von Webseiten hat sich die Angabe von Alternativtexten zu Grafiken mittlerweile etabliert. Die meisten Editoren enthalten Abfragen dazu oder zeigen Warnungen bei fehlendem `alt`-Attribut. Zudem enthalten Tutorials zur Erstellung von Webseiten entsprechende Hinweise. Allerdings wird der Alternativtext häufig nicht sinnvoll ausgefüllt. Dies liegt einerseits daran, dass Editoren, dieses Attribut zwar abfragen, aber keinen Hinweis dazu geben, wozu es wirklich dient. Zum anderen fließt der Alternativ-Text von Bildern in Webseiten zu einem gewissen Teil in die Indizierungsalgorithmen von Suchmaschinen ein, was dazu verleitet, nicht Informationen anzugeben, die direkt das gezeigte Bild bzw. dessen Zweck in der Webseite betreffen, sondern Stichwörter, die den Inhalt der Seite an sich bezeichnen, um deren Häufigkeit zu erhöhen. Aber auch eine detailliert Beschreibung des Bildinhalts entspricht nicht dem eigentlichen Zweck des `alt`-Attributs. Dieser ist, eine textuelle Repräsen-

tation zu bieten, die für die entsprechende Webseite den gleichen Zweck erfüllt wie die Grafik (siehe [Word] und [Webe]). Dieser kann auf verschiedenen Webseiten sehr unterschiedlich sein. So kann beispielsweise ein Icon mit einer Sonne auf einer Wetter-Seite für die Aussage „sonnig“ stehen und auf einer Astronomie-Seite als Link zu Informationen über die Sonne dienen. Weitere anschauliche Beispiele zum Nutzen des `alt`-Attributs besonders für Textbrowser (und damit jegliche rein textuelle oder sprachliche Darstellung) zeigt [Höh99].

Auch für Blinde und Sehbehinderte ist eine zweckbezogene Textalternative als erste Information über die Grafik sinnvoll. So kann der Nutzer schnell entscheiden, ob er zum Verständnis des Dokuments nähere Informationen zur Grafik benötigt. Aus diesem Grund sollten auch in anderen Dokumenten als Webseiten entsprechende Informationen ablegbar bzw. zugänglich sein. Allerdings muss dies nicht unbedingt in einem zusätzlichen, für normale Betrachter nicht nützlichen, Attribut erfolgen. Bei Grafiken, die Erläuterungen in Texten illustrieren oder Sachverhalte veranschaulichen, werden meist Bildbeschriftungen (oder Bildunterschriften) angegeben. Oft geben diese oder zumindest deren erste Zeile den Zweck der Grafik an, so wie es das `alt`-Attribut verlangt. Zur Angabe der Bildbeschriftung muss ein Autor meist bestimmte Konstrukte oder Eingabefelder ausfüllen. Werden hier entsprechende Aufteilungen in etwas wie „Kurzbeschreibung“ und „weitere Informationen“ vorgenommen, können sowohl die Bildunterschrift wie auch das `alt`-Attribut entsprechend gefüllt werden, ohne Mehraufwand für den Autor zu erzeugen. Weiterhin kann leicht ein Abbildungsverzeichnis generiert werden, was wie von den meisten Autoren gewünscht zu jeder Abbildung nur die Kurzbeschreibung enthält und nicht mehrere Zeilen von Text. Dadurch entsteht zusätzlich ein Mehrwert für den Autor selbst.

Zweckbezogene Alternativen zu Grafiken und auch Bildbeschriftungen in Dokumenten sind allerdings meist nicht hilfreich, wenn der Nutzer tatsächlich am Inhalt der Grafik interessiert ist. Häufig beschreiben diese nicht den Bildinhalt, da er entweder für den vorgesehenen Zweck der Grafik nicht interessant ist oder ja eindeutig zu sehen ist. Es ist aber auch nicht sinnvoll, die Beschreibung des Inhalts in dokumentgebundenen Texten zu hinterlegen. Sie gehören nicht zum Dokument, sondern zur Grafik selbst und sollten deshalb auch darin gespeichert sein. Wie bereits in Kapitel 2.2 angesprochen, bieten verschiedene Grafikformate unterschiedliche Möglichkeiten dazu. Wesentlich ist hierbei, dass die in der Grafik selbst abgelegten Beschreibungen bei Konvertierung in andere Formate und bei Einbindung in Dokumente verfügbar bleiben, sofern es das Format zulässt. Gehen Informationen bei Konvertierungen verloren, sollte der Benutzer darauf hingewiesen werden. Leider ist all dies bei den meisten aktuellen Werkzeugen nicht gegeben. Konvertierungen sind diesbezüglich fast immer verlustbehaftet. Und auch bei der Einbindung in Dokumente, die nicht die ursprüngliche Datei selbst referenzieren bleiben die eventuell mühevoll angelegten Beschreibungen nicht erhalten. Dabei könnten sie hier sogar genutzt werden. Dem Autor könnte die Möglichkeit geboten werden, Teile der Bildbeschreibung in seinen Text einzubinden. So wären diese auch normalen Benutzern leicht zugänglich. Besonders bei Grafiken, die komplexere Sachverhalte veranschaulichen, werden Erläuterungen, die den Inhalt der Grafik recht detailliert wiedergeben häufig im Text selbst präsentiert. Natürlich muss der so eingefügte Text bei

4 Förderung der zugänglichen Gestaltung

Entfernen der Grafik wieder gelöscht werden. Werden Rastergrafiken durch Konvertierung aus Grafik- und Dokumentformaten erstellt, in denen Informationen über die Semantik der Darstellung vorliegen, könnte sogar automatisch eine Bildbeschreibung in der Rastergrafik abgelegt werden. So könnten beispielsweise Daten, aus denen ein Diagramm erstellt wurde auch in der Rastergrafik leicht verfügbar gemacht und später in einem anderen Dokument wieder dargestellt werden, ohne dass ein Autor dadurch mehr Aufwand hätte.

Durch Verbesserungen bei Bild-Konvertierungen und Einbindung in Dokumente sowie Erweiterungen bei den Benutzerschnittstellen zur Bildbearbeitung oder -speicherung können in Grafikdateien abgelegte Beschreibungen einen wertvollen Beitrag zur Zugänglichkeit von Grafiken leisten. Allerdings hängt die Qualität der abgelegten Informationen immer vom Autor und dessen Fähigkeiten zur Bildbeschreibung ab, sofern sie nicht automatisch erzeugt wurden. Die Beschreibung kann nicht durch andere verbessert werden, sofern die Datei nicht in einer kollaborativen Umgebung wie etwa Wikipedia zur Verfügung steht. Ebenso kann keine Beschreibung ergänzt werden. Da häufig normale Nutzer aber weder die Zeit noch die nötigen Kenntnisse haben, um eine gute Beschreibung zu erstellen, wäre es sinnvoll, wenn Vertretergruppen von Blinden und Sehbehinderten oder gar die Betroffenen selbst Beschreibungen anlegen und diese mit den beschriebenen Dateien verknüpfen könnten. Dies könnte mit Hilfe einer Datenbank ermöglicht werden. Für Grafiken im Web könnten zur Identifikation der Grafiken einfach deren URLs genutzt werden. Dabei könnte regelmäßig eine Abfrage stattfinden, ob diese URLs noch verfügbar sind. Über das Erstellungsdatum der darunter abgelegten Datei oder einen Hashwert könnte geprüft werden, ob die Datei verändert wurde. Ebenso könnten auch Beschreibungen zu in Grafiken aus Büchern oder anderen offiziellen Dokumenten abgelegt werden. Die Identifizierung könnte dabei über den Titel des Buches oder Dokumentes oder einen eindeutigen Identifikator wie ISBN, ISSN oder DOI¹⁷ und die Nummer der Abbildung erfolgen (dies wurde beispielsweise in [tga] bereits so angewendet, siehe Abbildung 115).

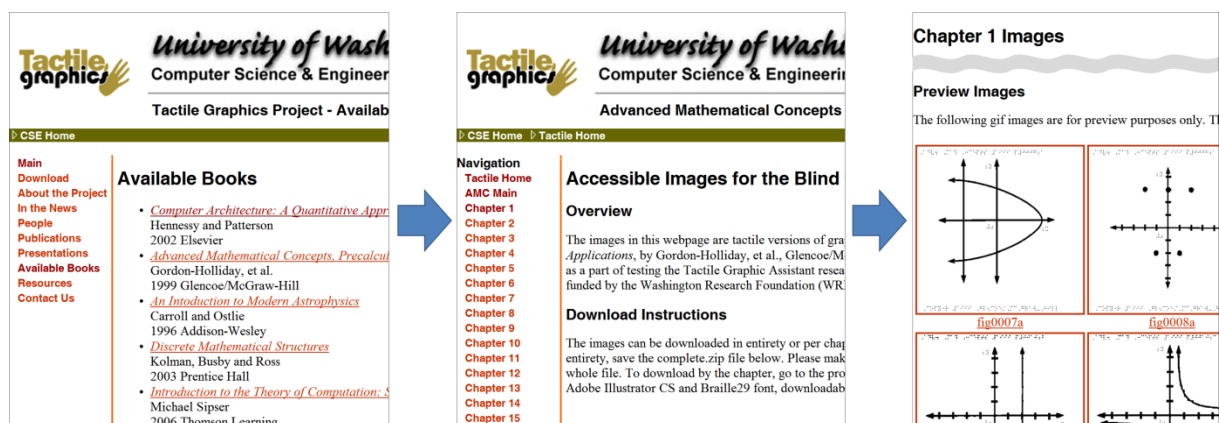


Abbildung 115 Beispiel zur Ablage und Auswahl alternativer Darstellungen zu Grafiken in Büchern. Auf der Webseite „Tactile Graphics“ werden zu einer Auswahl wichtiger Lehrbücher Grafiken für den grafikfähigen Braille-Drucker Tiger bereitgestellt. Die Auswahl erfolgt über den Name des Buches, die Kapitelnummer und die Nummer der Abbildung.

¹⁷ DOI = Digital Object Identifier (siehe <http://www.doi.org/>)

Gerade bei Büchern kann eine Datenbank-Lösung viele Vorteile bringen. Zum einen können so auch Darstellungen zugänglich werden, die nur in gedruckter Form vorliegen. Zum anderen kann sich der Arbeitsaufwand auf verschiedene Institutionen verteilen. Häufig werden Abbildungen aus Lehrbüchern durch verschiedene Personen immer wieder verbalisiert und für taktile Ausgabegeräte aufbereitet, um Studenten oder Schülern bereitgestellt zu werden. Diese Mehrfacharbeit kann durch eine zentral zugängliche Datenbank reduziert werden. Dabei können neben den textuellen Beschreibungen auch Vorlagen für taktile Grafiken abgelegt oder verknüpft werden, die ebenfalls kollaborativ entstehen könnten oder zumindest verbessert werden könnten. [Bre10] zeigt mit der Übersetzung von Vorlagen taktiler Grafiken einen ersten Ansatz dazu. Da besonders in Grundlagen-Lehrbüchern verschiedener Autoren zum gleichen Themengebiet häufig ähnliche Darstellungen auftauchen, sollten die alternativen Darstellungen nur lose mit den Original-Grafiken verknüpft sein, damit sie leicht auch für andere Grafiken genutzt werden können. So ist es auch möglich, verschiedene, taktile Darstellungen mit der Originalgrafik zu verknüpfen, die vielleicht auch nur einen Teil der Grafik abbilden oder nur einen bestimmten Aspekt. Bei komplexeren Grafiken kann dies sinnvoll sein. Vorlagen zu taktilen Grafiken sollten ebenfalls beschrieben sein, um Nutzern bei der Erkundung der Grafik zu helfen und eventuell anzugeben, auf welchen Teil oder Aspekt einer Grafik sie sich beziehen. Zudem sollten sie verschlagwortet werden, um auch unabhängig von einer visuellen Darstellung leicht auffindbar zu sein. So ist es auch möglich, bestehende Vorlagen wiederauffindbar in die Datenbank einzupflegen, ohne sie mit einer speziellen Grafik zu verknüpfen. Institutionen, die sich seit längerer Zeit in der Unterstützung von Blinden und Sehbehinderten engagieren, verfügen meist bereits über einen großen Fundus an taktilen Grafiken. Auch wenn diese bzw. deren Vorlagen nicht kostenfrei zur Verfügung gestellt werden sollen, könnten entsprechende Angaben zur Vorlage und Kontaktangaben zum Erwerb der Grafik in der Datenbank abgelegt werden. So sind sie für Betroffene und eventuell an einem Kauf interessierte Institutionen auch auffindbar. Weiterhin könnten auch Verknüpfungen zu alternativen Vektorgrafiken, die eventuell sogar der Ursprung der Rastergrafik waren, abgelegt werden. Natürlich kann auch in solch einer Datenbank nicht jede Grafik beschrieben werden, die im Web oder in Büchern auftaucht. Besonders im Web kommen täglich unzählige Grafiken hinzu. Die Entwicklungen auf dem Gebiet der Bildersuche können hier aber behilflich sein. Viele Grafiken im Web sind Kopien anderer Grafiken oder ähneln diesen sehr. Ist eines dieser Bilder bereits in der Datenbank abgelegt kann es durch Bildsuchverfahren aufgefunden und zur Beschreibung herangezogen werden. Dabei ersetzen die Beschreibungen natürlich nicht die zweckbezogenen Textalternativen. Sie beschreiben die konkrete Darstellung in der Grafik und nicht deren Sinn innerhalb eines bestimmten Dokuments. Da ausführlichere Bildbeschreibungen auch für andere Benutzer als nur Blinde und Sehbehinderte interessant sein können, sollte die Datenbank möglichst öffentlich zugänglich sein. So könnten beispielsweise auch Lehrer und Dozenten angeregt werden, in Lehrmaterialien Grafiken zu nutzen, für die bereits zugängliche Alternativen bestehen.

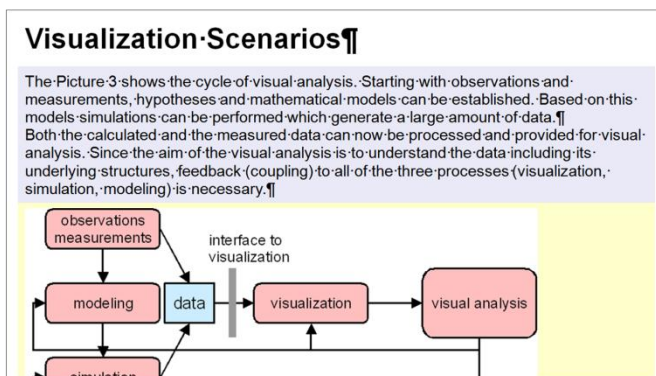
4.2.2 Zugängliche Lehrmaterialien im Web

Aufgrund der umfangreichen Bestrebungen zur Zugänglichkeit von Webangeboten, kann die webbasierte Darstellung von Lehrmaterialien einen großen Beitrag zur Zugänglichkeit von schulischer und universitärer Ausbildung und Weiterbildungsangeboten leisten. Natürlich wird das Web bereits umfangreich zur Bereitstellung von Lehrmaterialien eingesetzt. Meist werden aber gerade Skripte und Folien nur als PDF-Dateien bereitgestellt, da dies für die Autoren die einfachste und zuverlässigste Möglichkeit darstellt. PDF-Dateien können mittlerweile aus fast allen Editoren heraus erstellt werden und auch von allen Nutzern angezeigt werden. Zudem eignen sie sich gut zum Druck der bereitgestellten Materialien, was von vielen Lernenden noch gefordert wird. PDF-Dateien bergen allerdings häufig Zugänglichkeitsprobleme in sich. Außerdem bieten sie natürlich kaum Möglichkeiten zur Interaktion. Zwar ist den meisten Autoren bewusst, dass Interaktivität beim Lernen nützlich ist und webbasierte Lösungen hier viele Vorteile bringen können. Die Erstellung entsprechender Materialien ist aber meist zu aufwändig und erfordert häufig Expertenwissen. Oftmals ist die Einarbeitung in spezielle Programme und Verwaltungsmechanismen nötig, was selbst gewillte Autoren häufig vor unlösbare Probleme stellt. Die Nutzungsweise von Lernplattformen zeigt dies deutlich. Zwar haben sich Lernplattformen im Web mittlerweile in vielen Bereichen (auch außerhalb von Schulen und Universitäten) etabliert. Neben verschiedenen kommerziellen Produkten existiert eine Vielfalt an Open-Source-Entwicklungen (siehe beispielsweise [cam10]). Und häufig bieten diese Plattformen auch Werkzeuge zur Erstellung von webbasierten Lehrmaterialien. Mittlerweile sind diese sogar teilweise in bestehende Editoren integriert, wie beispielsweise bei der Plattform Metacoocn [met08], was auch eine deutliche Erleichterung für die Erstellung von Materialien bringt, wie [Rot05] zeigte. Allerdings ist zu beobachten, dass diese Möglichkeiten noch nicht umfassend genutzt werden. Auch in diesen Umgebungen werden die meisten Materialien nur im PDF-Format bereitgestellt. Die Werkzeuge sind für viele Autoren immer noch zu umständlich zu benutzen.

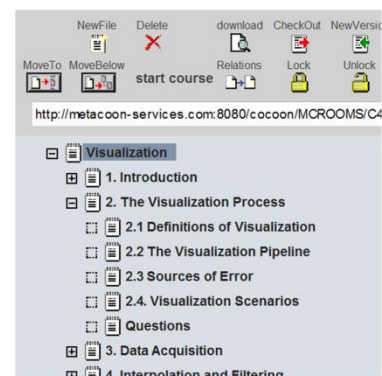
Bei Metacoocn beispielsweise bestehen zwei wesentliche Probleme. Zum einen kann nur OpenOffice-Writer (als normale Büroanwendung) zur Erstellung der Lerndokumente genutzt werden. Weder wird die Konvertierung von Folien, welche häufig als Lehrmaterialien entstehen, unterstützt noch irgendeine andere (durchaus auch häufiger eingesetzte) Software zur Erstellung von Lehr- und Informationsmaterialien wie etwa Word oder besonders im wissenschaftlichen Bereich auch LaTeX. Dieses Problem wurde von den Entwicklern glücklicherweise bereits erkannt. Zum anderen wird es Autoren schwer gemacht, Dokumente anzubieten, die bereits in Kapitel und Unterkapitel gegliedert sind und darüber im Web auch leicht navigierbar sein sollen. Zwar bietet Metacoocn in Webmaterialien eine Navigationsleiste für Kapitelstrukturen an, um diese zu erzeugen, muss ein Autor sein eventuell bestehendes Gesamtdokument aber in Teildokumente untergliedern, die er dann in einem Online-Autorenwerkzeug wieder in einer Kapitel- und Unterkapitelstruktur zu einem Lernkurs zusammensetzt (siehe Abbildung 116). Bezüglich des ursprünglichen Gedankens der Plattform ist dies zwar sinnvoll, da so leicht kleinere Lerneinheiten immer wieder neu in Kursen verwendet und kombiniert werden können und auch der Austausch einzelner Einheiten leicht

fällt. Es verschreckt aber eine Menge von Autoren, die eher in buch- oder skriptartigen Gesamtdokumenten, also fortlaufenden und thematisch abgeschlossenen, arbeiten. Eine Aufteilung solcher Dokumente ist sehr aufwändig und stellt den Autor danach noch vor das Problem, den Überblick über die vielen kleinen Dokumente zu behalten. Metacoön sollte hier eine Möglichkeit bieten, auch Gesamtdokumente abzulegen, die dann automatisch entsprechend der bestehenden Kapitelstruktur in einzelne, navigierbare Webseiten aufgegliedert werden. Dabei sollte der Autor bestimmen können, an welcher Überschriftenebene die Aufteilung durchgeführt wird. Auf diese Weise kann der Autor wie gewohnt arbeiten und sein Dokument trotzdem leicht als Web-Dokument zur Verfügung stellen und somit die Zugänglichkeit seiner Materialien erhöhen.

Erstellung von Kursfragmenten in einzelnen OpenOffice-Dokumenten mit vorgegebenen Formatvorlagen (Offline)



Anordnung der Kursfragmente zu einem vollständigen Kurs in der Metacoön-Online-Plattform



Darstellung der Kursfragmente als einzelne Webseiten mit Navigationsmöglichkeiten

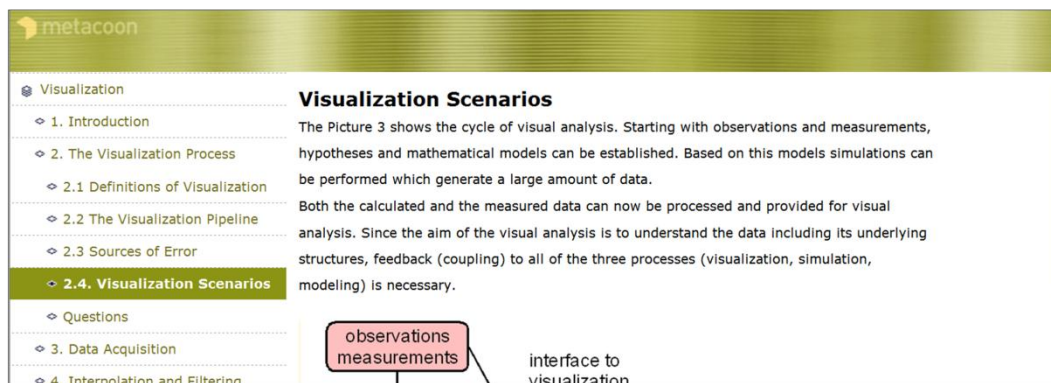


Abbildung 116 Erstellung eines webbasierten Lernkurses über Metacoön am Beispiel des im Rahmen dieser Arbeit erstellten Kurses zur Visualisierung [TRE07]

Die Lerninhalte müssen in kleine Einheiten unterteilt werden, die später als einzelne Webseiten dargestellt werden. Über den Metacoön-Kurskonfigurator (im Bild oben rechts) können bzw. müssen Lerneinheiten zu Kursen zusammengefügt werden, so dass ein Navigation möglich ist.

Natürlich sind auch auf diese Weise erstellte Dokumente nicht von sich aus völlig barrierefrei. Metacoön nutzt zwar schon einen sehr wertvollen Ansatz. Verschiedene Dokumentbereiche können leicht durch vordefinierte Formatvorlagen genauer spezifiziert werden (z.B. Einleitung, Lernziel, Übung, Algorithmus). Diese Auszeichnung hat auch direkten Einfluss auf die Darstellung und Nutzungsmöglichkeiten in der Webseite. Die Arbeit des Autors ist also

4 Förderung der zugänglichen Gestaltung

nicht umsonst. Allerdings kann es natürlich trotzdem vorkommen, dass beispielsweise Tabellen schlecht eingesetzt oder Bilder nicht beschrieben sind. Stehen die Materialien online, könnte leicht ein System etabliert werden, über das Betroffene solche Probleme in den Dokumenten markieren können. Der Autor wird so darauf hingewiesen und muss nicht selbst Experte in diesem Aspekt sein. Da Metacoon auch als Versionsverwaltungssystem für die abgelegten Materialien dient, könnte der Autor seine Materialien auch zur Bearbeitung durch Experten im Gebiet Zugänglichkeit freigeben, wodurch auf einfache Weise sowohl Expertenkenntnisse im inhaltlichen Bereich der Materialien wie auch im gestalterischen zusammengeführt werden können.

Ein Punkt, der die Nützlichkeit der Lernplattformen bedeutend schmälert, ist die mangelnde Unterstützung zur Bereitstellung von Präsentationsfolien. Besonders PowerPoint-Folien werden aufgrund der weiten Verbreitung und leichten Handhabbarkeit häufig eingesetzt, um Lerninhalte zu vermitteln. Die Bewegung des „Rapid e-Learning“ zeigt deutlich den Bedarf nach entsprechender Unterstützung zur leichten Bereitstellung von PowerPoint-Folien im Web [Pre05]. Dabei sollen natürlich insbesondere animierte Abläufe erhalten bleiben. Allerdings soll die PowerPoint-Datei selbst meist nicht einfach zum Download angeboten werden, um die Arbeit des Autors zu schützen. Aus diesem Grund werden Präsentationsdokumente heute im Web häufig auch als Video angeboten. Animationen können hier gut dargestellt werden. Für die Zugänglichkeit der Materialien ist dies natürlich fatal. Zwar sind diese Präsentationen oft mit Audiokommentaren verbunden, diese geben aber oft nicht den Text auf den Folien wieder und beschreiben auch nur selten die grafische Darstellung. Somit stehen einem Blinden oder Sehbehinderten wieder wesentlich weniger Informationen zur Verfügung als Normalsichtigen. Eine weitere häufig eingesetzte und oft sogar in PowerPoint integrierbare Möglichkeit zur Bereitstellung im Web ist die Konvertierung in Flash (siehe beispielsweise <http://www.pptflashstudio.com/>). Leider werden die Präsentations-Folien hier aber oft in Rastergrafiken gewandelt, obwohl auch mit diesem Zielformat deutlich bessere Umwandlungen möglich wären. Flash-Anwendungen, die die aktuellste Flash-Version nutzen, können, wenn sie gut erstellt wurden, auch für Screenreader zugänglich sein [Krü07]. Werden Werkzeuge zur Konvertierungen unter Beachtung der aktuellsten Technologien und Richtlinien erstellt, kann damit auch eine sorgfältige Umsetzung gewährleistet werden.

Eine positive Entwicklung bei Web-Folien zeigt sich durch „Google Docs“ [Goo]. Hier können PowerPoint-Präsentationen mit wenigen Klicks in den Online-Editor geladen und auf Webseiten bereitgestellt werden. Dabei erfolgt die Bereitstellung in einem HTML-Format. Damit sind Texte leicht zugänglich und die Darstellung kann durch eigene CSS-Stylesheets beeinflusst werden. Ein Export nach PDF ist für jeden Betrachter möglich. Bisher sind in Google-Docs-Präsentationen zwar noch keine Animationen möglich. Die Entwickler des Web-Folien-Formats S5 [Mey10] zeigten aber bereits, dass dies umsetzbar ist. Durch die Verwendung von HTML bzw. XHTML werden natürlich noch weitere Möglichkeiten eröffnet. Auch Grafiken im SVG-Format und Formeln in MathML [Worh] können integriert werden. Zwar bestehen hier häufig Bedenken, dass diese Formate noch nicht von allen Browsern korrekt unterstützt werden. Für diesen Fall können aber leicht Ersatzdarstellungen generiert werden, wie

beispielsweise die SVG-Behandlung in Wikipedia zeigt [Wik10c]. Mit Hilfe des `object`-Tag von HTML können in Webseiten verschiedenste solcher Ersatzdarstellungen zu einem Objekt angegeben werden [Word]. Listing 21 illustriert dies.

```

<!-- First, try the Python applet -->
<OBJECT title="The Earth as seen from space"
        classid="http://www.observer.mars/TheEarth.py">
    <!-- Else, try the MPEG video -->
    <OBJECT data="TheEarth.mpeg" type="application/mpeg">
        <!-- Else, try the GIF image -->
        <OBJECT data="TheEarth.gif" type="image/gif">
            <!-- Else render the text -->
            The <STRONG>Earth</STRONG> as seen from space.
        </OBJECT>
    </OBJECT>
</OBJECT>

```

Listing 21 Beispiel zur verschachtelten Angabe von Alternativdarstellungen über das HTML-Object aus [Word]

Durch HTML5 [Wor10] wird sich die Unterstützung für SVG und MathML aber auch deutlich verbessern. Dabei würde die Erzeugung von SVG und MathML für den Autor keinerlei Mehrarbeit bedeuten. Ganz im Gegenteil würden sie zur Erhaltung der vom Autor angelegten Strukturen dienen. So können beispielsweise auch PowerPoint-Vektorgrafiken, die mit Animationen verbunden sind, in SVG abgebildet werden. Die entsprechende Funktionalität könnte also bei gleichzeitiger Erzeugung zugänglicher Webmaterialien weitestgehend erhalten bleiben. Dies wäre auch für Normalsichtige nützlicher als die Bereitstellung als Video oder PDF. Da auch PowerPoint-Dateien mittlerweile in einem XML-Format abgelegt werden, sind für entsprechende Konvertierungen die besten Voraussetzungen gegeben. Die meisten DrawingML-Konstrukte [oox08], in denen Grafiken in PowerPoint abgelegt sind, lassen sich in SVG umwandeln. Weiterhin sind beispielsweise auch Datenwerte aus eingebundenen Excel-Dateien enthalten, aus denen Diagramme erstellt wurden. Diese könnten in die Beschreibungen des dazu generierten SVG einfließen und auch in die alternative Textdarstellung des HTML-Object.

4.2.3 Zweckgebundene Labels und Gruppenvorschriften für zugänglichere GUIs

Betrachtet man Dialoge wie den in Abschnitt 4.1 diskutierten Dialog (Abbildung 111 links) oder die Dialoge in Abbildung 117 näher, so kann man feststellen, dass viele Probleme im Wesentlichen durch zwei Dinge verursacht werden. Zum einen können Autoren durch freie Anordnung von Elementen leicht visuelle Gruppierungen schaffen, die in der Struktur des Dialogs nicht repräsentiert sind. Zum anderen Schaffen aktuelle Drag&Drop-GUI-Editoren bei Ausrichtung von Elementen Gruppierungen, die für den Autor nicht erkennbar sind, wie etwa die Zuordnung des Labels „(*.txt)“ zur Gruppe mit Radiobutton 3 und 4 im Dialog aus Abschnitt 4.1. Besonders offensichtlich sind die so entstehenden falschen Gruppierungen

4 Förderung der zugänglichen Gestaltung

oder auch die so eben nicht entstehende Gruppierung wie bereits erläutert bei Beschriftungen. Aber auch Gruppierungen zwischen mehreren Eingabeelementen können von diesem Problem betroffen sein, wie etwa die durch Anordnung in einer Linie zusammengefassten Eingabefelder für „Latitude“ im Dialog in Abbildung 117 links oder die durch Einrückung dem Radiobutton „Scale all at“ zugeordnete Checkbox „Shrink Oversize Layouts to Page“. Zwei einfache Maßnahmen können hier wesentliche Verbesserungen bringen. Diese sind:

1. Abschaffung freier Labels (und Icons) in Drag&Drop-Editoren
2. Anwendung von Gestaltungsregeln für Gruppierungen und Visualisierung jeglicher Gruppierung

Beide Maßnahmen schränken Autoren zwar etwas in ihrer Gestaltungsfreiheit ein, bringen ihnen aber, wie im Folgenden dargestellt wird, auch viele Erleichterungen. Neben der Zugänglichkeit steigt auch die Bedienbarkeit (Usability) an sich. Somit bringen die Maßnahmen Vorteile für alle Nutzer. Die folgenden Erläuterungen beziehen sich jeweils nur auf Labels. Sie gelten aber genauso auch für frei verwendbare Icons, wie beispielsweise die symbolischen Darstellungen der Seitenreihenfolge in Abbildung 117 rechts. So wie dies bei Schaltflächen heute meist schon üblich ist, sollte für den Autor immer die Möglichkeit bestehen, zwischen einer „Beschriftung“ durch Text, ein Icon oder beides zu wählen. Zu Icons sollte natürlich ein Alternativtext hinterlegt sein.

Frei verwendbare Beschriftungen (Labels) werden heute zur Benennung von Bedienelementen und für Hinweise an den Benutzer (z.B. zur Bedienung eines Dialogs oder zur Angabe von Einschränkungen der möglichen Eingabe) eingesetzt. Beschriftungen dienen also immer einem bestimmten Zweck. Dieser sollte auch aus ihnen ableitbar sein. Ein erster Schritt dazu wurde zwar mit der Einführung der „labeled by“- oder „label for“-Attribute gemacht. Es zeigt sich aber immer wieder, dass die Besetzung dieses Attributs nicht durchgeführt wird. Selbst auf den Webseiten des W3C, in denen der Aufbau von HTML-Formularen beschrieben wird [Wor99], befindet sich direkt nach dem Beispiel zur Verknüpfung von Labels und Bedienelementen ein Formular-Beispiel, in dem diese Verknüpfung nicht angewendet wird, obwohl dieses Beispiel sogar unter der Überschrift „*Adding structure to forms ...*“ steht. Die Zuweisung von Beschriftungen zu Bedienelementen muss für den Autor also einfacher (direkter) erfolgen. Für Bedienelemente wie Schaltflächen (Button) und Kontrollfelder (Checkbox) ist dies durch einfache Angabe eines Titels oder Namen bereits Gang und Gäbe. Auch Gruppen können mit Titeln versehen werden, die als Beschriftung angezeigt werden. Ebenso sollte für Eingabefelder und ähnliche derzeit beschriftungslose Bedienelemente verfahren werden.

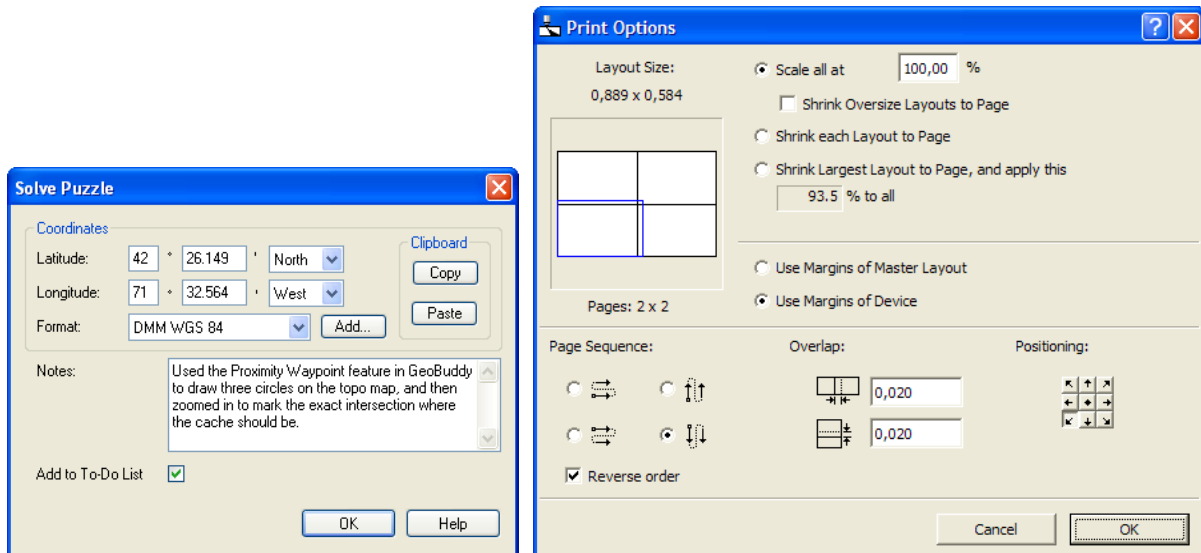


Abbildung 117 Beispiele für Dialoge¹⁸

In Dialogen werden Beschriftungen meist als Bezeichnung (Titel), als Beschreibung und als Hilfe- oder Hinweistext für den Benutzer eingesetzt (wie etwa das Element „(*.txt)“ im Dialog aus Abschnitt 4.1 Abbildung 111 links). Sie beziehen sich jeweils auf ein einzelnes Element, eine Gruppe von Elementen oder auch den ganzen Dialog. Diese Arten von Beschriftungen sollten einem Autor also als Eigenschaft von Elementen, Gruppen oder Dialogen zur Verfügung stehen. Natürlich müssen nicht immer alle Angaben ausgefüllt werden. Die Titel-Label sollten aber bei Einfügen eines Elements oder einer Gruppe bzw. bei Anlegen eines neuen Dialogs direkt dargestellt werden. Einerseits erspart dies dem Autor einen Bedienschritt, den er in den meisten Fällen sowieso ausführen würde, zum anderen regt es ihn an, diese Titel sinnvoll zu füllen. Da dem Autor keine andere Möglichkeit zur Bezeichnung zur Verfügung steht, wird er diese Titel auch sinnvoll belegen und nur dann leeren, wenn auch für Sehende Nutzer keine Beschriftung nötig ist. Beim Zugriff über Zugänglichkeitsschnittstellen entstehen so keine Nachteile mehr gegenüber dem visuellen Zugriff. Natürlich müssen dem Autor Layoutoptionen geboten werden, um beispielsweise die Beschriftung eines Eingabefeldes wahlweise links davor, darüber oder auch wie bei der Angabe der „Latitude“-Werte dahinter zu platzieren. Feste Layoutoptionen verbessern einerseits die Bedienbarkeit (beispielsweise große Abstände zwischen Beschriftung und Bedienelement wie bei den „Notes“ in Abbildung 117 links werden dadurch seltener auftreten) und andererseits spart dies dem Autor viel Arbeit. Er muss nicht mehr selbst für eine gute Ausrichtung sorgen. Da das System weiß, welches Label zu welchem Bedienelement gehört, kann die Anordnung automatisch passieren. Dies verbessert also auch die Anpassung an andere Schrifteinstellungen. Mit einer Option für „freies Layout“ kann der Autor seine Kreativität immer noch ausleben. Die meisten Autoren werden aber die vorgegebenen Layouts nutzen, da diese ihren Zweck erfüllen und leicht zu handhaben sind. Durch die direkte Verknüpfung zwischen Bedienele-

¹⁸ Bildquellen: http://www.geobuddy.com/help/images/dlg/dlg_solve_puzzle.png,
<http://www.archicadwiki.com/Archicad%2010/Project%20Migration/Printing-Plotting%20Publisher%20sets%20should%20be%20reviewed>

4 Förderung der zugänglichen Gestaltung

menten und Titeln kann das System auch automatisch Accesskeys erzeugen, und so dem Autor einen weiteren Arbeitsschritt abnehmen, der vordergründig nur der Zugänglichkeit dient, letztlich aber allen Nutzern zu Gute kommt. Als Accesskey wird meist der erste Buchstabe einer Element-Beschriftung verwendet. Ist dieser schon vergeben, kann das System leicht einen anderen Buchstaben wählen. Bei Konflikten könnte es notfalls auch den Autor zur Bestimmung eindeutiger Accesskeys auffordern. Somit wären alle Bedienelemente immer mit Accesskeys versehen, auch wenn eine Oberfläche von einem Autor erstellt wurde, der kein Wissen über zugängliche Gestaltung hat. Natürlich muss ein Autor, der seine Oberfläche bewusst zugänglich gestalten möchte, die Möglichkeit haben, die gesetzten Accesskeys zu verändern.

Auch die Positionierung der Beschreibungen und Hinweistexte sollte in festen Layouts erfolgen. Da Beschreibungen auch visuell einem Element gut zuordenbar sein sollen, dürfen sie nicht irgendwo im Dialog stehen. Beschreibungen zu einzelnen Elementen sollten immer nach diesen Elementen stehen (rechts oder darunter), Beschreibungen zu Gruppen und Dialogen sind nur zu Beginn oder am Ende der Gruppe bzw. des Dialogs sinnvoll. Wo immer heute längere Texte zwischen Bedienelementen verwendet werden, sind dies entweder Beschreibungen zu einem Element oder zu mehreren Elementen, die dann gruppiert sein sollten. Existiert die Notwendigkeit, einen erklärenden Text zu mehreren Elementen anzugeben, so stehen diese Elemente auch im Zusammenhang. Der Text kann also bei Abschaffung frei verwendbarer Labels als Gruppenbeschreibung umgesetzt werden. Können keine Labels direkt verwendet werden, wird der Autor einer Bedienoberfläche also automatisch gezwungen, zusammengehörige Elemente zu gruppieren, wenn er dem Nutzer weitere Hinweise dazu geben will, wie dies beispielsweise in Abbildung 117 rechts bei den Bedienelementen für „Page Sequence“, „Overlap“ und „Positioning“ der Fall ist. Natürlich sollte der Autor entscheiden können, ob eine Beschreibung überhaupt direkt im Dialog angezeigt werden soll oder beispielsweise als Tooltip. Zum Hinweis auf Hilfetexte können zunächst die heute schon üblichen Fragezeichensymbole dargestellt werden, die den Hilfetext als Tooltip anzeigen oder bei Klick drauf einen speziellen Hilfebereich öffnen. Durch die Typisierung der Beschriftungen kann das System automatisch übliche Darstellungen generieren. Dem Autor wird dadurch viel Arbeit abgenommen.

Wie Abbildung 118 zeigt, werden Labels nicht nur für Beschriftungen und Beschreibungen genutzt. Ein häufiges Einsatzgebiet ist auch die einfache Anzeige von Werten wie beispielsweise den Eigenschaften einer Datei oder zur Übersicht über einen Datensatz. Für diese Verwendungen müssen Ersatzelemente geschaffen werden. Die Darstellung von Werten kann mit Hilfe eines speziellen Wert-Anzeige-Labels realisiert werden, das ähnlich zu einem beschrifteten Eingabefeld strukturiert ist, also Titel, Beschreibung und Hilfetext enthalten kann, und aus mindestens zwei Darstellungsteilen besteht – in diesem Fall dem Titel (oder Eigenschaftsnamen) und dem Wert. Damit das Wert-Anzeige-Label nicht einfach als freie Beschriftung genutzt werden kann, muss es zwingend ein Trennzeichen, wie etwa den Doppelpunkt oder ein Gleichheitszeichen anzeigen. Natürlich kann nur schlecht verhindert wer-

den, dass sowohl Titel, als auch Wert leer sind (gerade beim Wertfeld kann dies häufig auftreten). Durch das Trennzeichen wird der Autor aber angeregt, beides sinnvoll zu füllen.

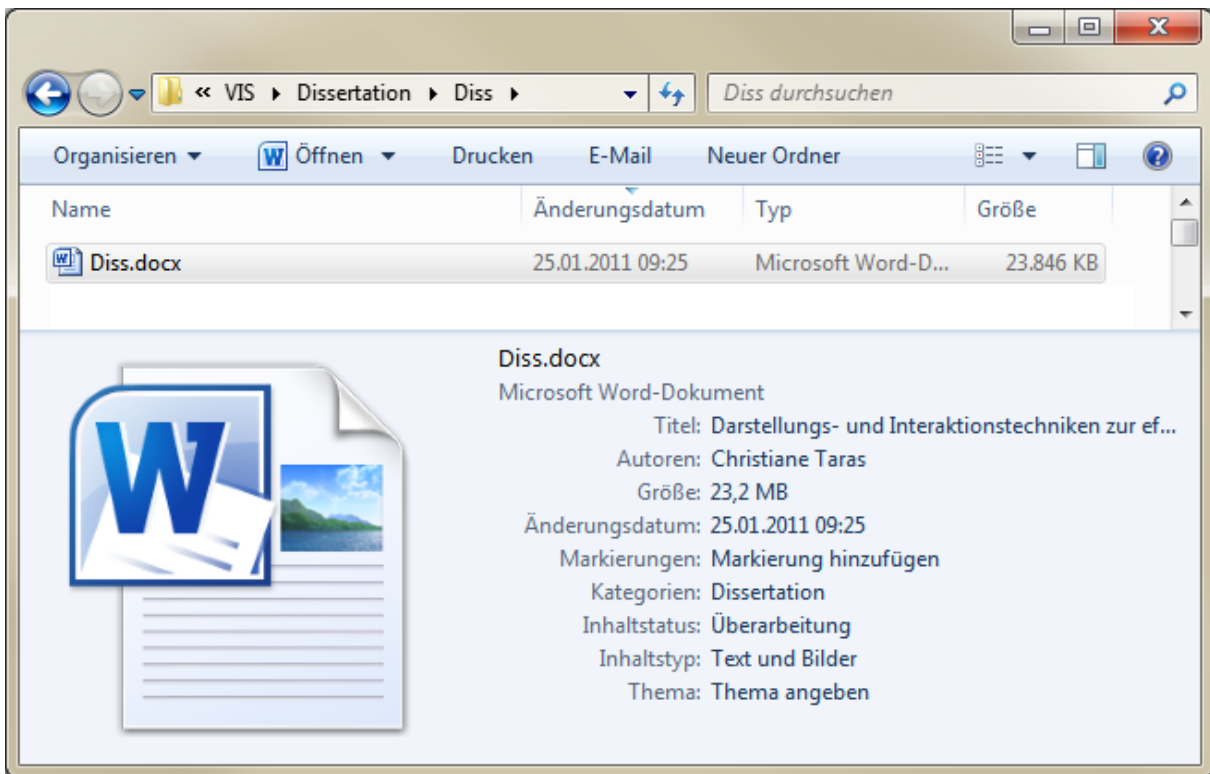


Abbildung 118 Beispiel zur Verwendung von Labels zur Anzeige von Eigenschaftswerten

Soll kein Trennzeichen verwendet werden, kann eine Anordnung von Name und Wert in einer Tabelle verwendet werden. Dabei sollten Tabellen so umgesetzt sein, wie dies beispielsweise in Java der Fall ist. Das heißt, es sollten keine Bedienelemente in die Tabelle hineingezogen werden, sondern der Typ einer Tabellenzelle sollte bestimmt werden können, also beispielsweise Eingabefeld, Checkbox oder Label. Dabei werden in Tabellen für Bedienelemente keine eigenen Titel dargestellt. Möchte der Autor die Elemente beschriften, so muss er Spalten- und Zeilen-Köpfe nutzen, aus denen sich der Titel jedes Elements automatisch ergibt. Besteht das Bedürfnis, Elementen in Tabellen jeweils separate Titel zu geben, so kann man davon ausgehen, dass der Autor die Tabelle nicht, wie es eigentlich vorgesehen ist, zur semantisch sinnvollen Gruppierung von Elementen nutzen wollte, sondern lediglich zur tabellarischen Ausrichtung von Elementen. Zu diesem Zweck sollte aber eine entsprechende Layout-Option für Gruppen existieren. Die Trennung zwischen diesen beiden Tabellenarten ist sinnvoll. Eine lediglich für Layoutzwecke genutzte Tabelle kann ohne Informationsverlust für eine angepasste Darstellung linearisiert werden. Bei einer echten tabellarischen Gruppierung ist dies nicht unbedingt möglich.

Um den Gruppierung von Elementen zu unterstützen, sollten Gruppen auch wesentlich dynamischer erzeugt werden können, als dies heute in GUI-Editoren möglich ist. Zur Erstellung einer Gruppe muss meist zunächst ein Gruppenelement eingefügt werden, in welches die entsprechenden Bedienelemente abgelegt werden. Die Einführung einer Gruppe im Nach-

4 Förderung der zugänglichen Gestaltung

hinein ist also relativ aufwändig. Es sollte zusätzlich ein Verfahren angewendet werden, wie es aus Grafik-Editoren bekannt ist, bei dem einfach eine Menge von Elementen markiert und über einen Menübefehl gruppiert werden kann. Natürlich muss vor Ausführung der Gruppierung vom System geprüft werden, ob nötige Randbedingungen erfüllt sind, also beispielsweise keine Elemente markiert wurden, die sich in verschiedenen Gruppen befinden, ohne dass jeweils die gesamte Gruppe in die Markierung einbezogen wurde. Der Autor sollte in solch einem Fall gefragt werden, ob die markierten Elemente aus ihren bisherigen Gruppen entnommen und in der neuen Gruppe neu positioniert werden sollen. Eine Änderung der Positionen der markierten Elemente ist in solch einem Fall nötig, da Elemente von Gruppen in Bedienoberflächen im Gegensatz zu Gruppierungen in Grafiken natürlich immer dicht beieinander liegen müssen. Sie sollten von einem rechteckigen Rahmen umschlossen werden können, ohne dass dieser Elemente überdeckt, die nicht in der Gruppe sind. Bei Erstellung einer Gruppe sollte zunächst auch immer ein Rahmen dargestellt werden. Dies fördert meist die Bedienbarkeit, da die Gruppierung direkt ersichtlich ist und somit leicht ein Zusammenhang zwischen den Elementen erkannt werden kann. Über eine entsprechende Eigenschaft sollte der Rahmen ähnlich zu „border“ in CSS gestaltbar sein, so dass beispielsweise auch nur die untere Kante dargestellt werden kann. Dies schränkt auch die Nutzung von horizontalen Trennlinien ein, die nur visuell gruppierend wirken.

Innerhalb einer Gruppe sollte immer eine Layoutvorgabe für die Kindelemente bestehen, beispielsweise „vertikal angeordnet“ oder „tabellarisch“. Dies erspart dem Autor Positionierungsarbeit, sorgt direkt für eine gute Ausrichtung der Elemente, was die Bedienbarkeit erhöht, und verbessert die Anpassung an andere Schrifteinstellungen. Vor allem aber weist es den Autor auf mögliche strukturelle Fehler hin. Lassen sich Elemente im vorgegebenen Layout nicht so positionieren, wie gewünscht, ist dies meist ein Hinweis darauf, dass die Elemente zu mehreren Gruppen gehören. In Abbildung 117 links würden in einem vertikalen Layout beispielsweise die sechs Eingabefelder für „Latitude“ und „Longitude“ alle untereinander positioniert werden, in einem horizontalen Layout alle nebeneinander. Ein tabellarisches Layout kann hier helfen, ist aber durch die weiteren Elemente für „Format“ nur schwer richtig zu definieren. Höchstwahrscheinlich würde ein Autor dadurch angeregt werden, weitere Gruppierungen vorzunehmen und so auch (wie es semantisch sinnvoll ist) die Schaltfläche „Add“ über eine Gruppierung dem Eingabefeld für „Format“ zuzuordnen. Die Layoutoptionen selbst sollten schnell änderbar sein. Die Verwendung der heute üblichen Layout-Container ist für den Autor nicht wirklich hilfreich, da sie einen Layoutwechsel sehr aufwändig machen. Natürlich sollte weiterhin ein freies Positionieren möglich sein. Es sollte aber nicht als Grundeinstellung angewendet werden, um den Autor anzuregen, sich mit den vorgegebenen Layoutoptionen auseinanderzusetzen. Weiterhin ist eine Layoutoption empfehlenswert, die alle Titel-Label von Eingabefeldern und ähnlichem visuell in einer Spalte und die Eingabefelder in einer zweiten Spalte ausrichtet. Solche Layouts werden häufig verwendet, wie auch Abbildung 117 links zeigt.

Eine weitere Möglichkeit, Gruppierung anzuregen, ist die Kennzeichnung von Gruppenbereichen bzw. den Zuordnungen zwischen Beschriftungen und Bedienelementen. Im Dialog in

Abbildung 117 links ist beispielsweise die Beschriftung „Latitude“ nur dem direkt folgenden Eingabefeld zugeordnet. Die bisher beschriebenen Maßnahmen machen solch eine Zuordnung nicht weniger wahrscheinlich. Visualisiert man dem Autor allerdings diese Zuordnung, so ist es wahrscheinlich, dass dieser erkennt, dass sich „Latitude“ eigentlich auf alle drei Eingabefelder beziehen sollte. Kann er nun die dargestellte Zuordnung durch Erweiterung der Markierung einfach ändern, so führt er automatisch eine Gruppierung aus, wobei „Latitude“ als Titel der Gruppe gesetzt wird, der in diesem Fall ähnlich einer Zeilenbeschriftung in Tabellen links vor allen Elementen der Gruppe steht. Der Bedienvorgang ist einfach und somit ist es wahrscheinlich, dass er auch durchgeführt wird. Hier muss das System natürlich verhindern, dass die Markierung über die Gruppe, in der sich das „Latitude“-Bedienelement befindet, hinausgezogen wird. Kein Element darf direkt mehreren Gruppierungen angehören. Dies ist in Bedienoberflächen nicht sinnvoll.

Ebenso dürfen sich innerhalb eines Bereichs von gruppierten (oder einander zugeordneten) Elementen keine Elemente befinden, die nicht der Gruppe angehören. Eine Positionierung von Elementen innerhalb einer Gruppe muss immer deren Aufnahme in die Gruppe bewirken, um die durch die Positionierung entstehende visuelle Gruppierung in der Struktur des Dialogs abzubilden. Dies bedeutet zum einen, dass Elemente, wie oben bereits diskutiert, in das Layout einer Gruppe eingepasst werden, was den Autor auf strukturelle Fehler hinweisen kann. Zum anderen entstehen dadurch aber auch Restriktionen in der Positionierung von Elementen. So dürfen beispielsweise keine Elemente zwischen einer Beschriftung und dem zugehörigen Bedienelement eingefügt werden. Legt ein Autor ein Element per Drag&Drop an solch einer Position ab, ist dies ein Hinweis darauf, dass die Beschriftung eigentlich eine Gruppenbeschriftung für mehrere Elemente sein sollte. In solch einem Fall sollte der Autor also gefragt werden, ob eine Gruppe aus den beiden Bedienelementen erstellt werden soll, die die Beschriftung des bereit vorhandenen Bedienelements trägt. Lehnt der Autor dies ab, sollte das neu eingefügte Element entsprechend der Position, an der es abgelegt wurde, vor oder nach dem bereits vorhandenen Element positioniert werden. Bei diesem Prozess ist es für den Autor hilfreich, wenn während des Drag&Drop-Vorgangs markiert würde, welcher Gruppe das Element zugeordnet werden würde, wenn es an der aktuellen Position abgelegt werden würde. Bei der freien Positionierung von Elementen werden solche Markierungen (die Ausrichtungslinien) heute bereits dargestellt.

Ähnliche Positionierungseinschränkungen müssen auch bei Gruppen angewendet werden, die eine Beschränkung auf bestimmte Elementtypen voraussetzen. Beispielsweise ist es heute bereits üblich, Radiobuttons in einer Gruppe zu vereinen, um eine automatische Behandlung der Deaktivierung von Radiobuttons bei Aktivierung eines anderen Radiobuttons zu erhalten. Auch wenn diese Gruppierungen primär vorgenommen werden, um eine gewisse Funktionalität leicht umzusetzen, sind sie auch semantisch sinnvoll. Radiobuttons dienen der Auswahl einer Option aus mehreren möglichen Optionen, somit stehen sie auch direkt in Zusammenhang und die Anordnung anderer Elemente innerhalb dieser Gruppe macht bezüglich der Gruppe keinen Sinn. Derartig positionierte Elemente existieren zwar häufig, wie beispielsweise auch in Abbildung 117 rechts, sie stehen aber immer im Zusammenhang mit

4 Förderung der zugänglichen Gestaltung

einem bestimmten Radiobutton und dienen der Angabe weiterer Parameter zu der durch den Radiobutton gewählten Option. Sie sind also semantisch nicht Teil der gesamten Radiobutton-Gruppe, sondern müssen als neue Gruppe dem einzelnen Radiobutton zugeordnet werden, wobei wieder automatische Layoutoptionen wie beispielsweise „unterhalb eingerückt“ angewendet werden können. Durch diese Gruppierung kann das System auch automatisch eine funktionelle Verknüpfung vornehmen, ähnlich wie bei der Radiobuttongruppe. So kann, wie dies heute auch häufig bereits umgesetzt ist, ein mit einem Radiobutton verknüpftes Eingabefeld automatisch deaktiviert sein, wenn der Radiobutton nicht angewählt ist und ein Klick auf das Eingabefeld kann den Radiobutton automatisch aktivieren. Autoren von Bedienoberflächen müssten diese Funktionalität somit nicht mehr immer und immer wieder über eigens programmierten Code umsetzen. Weiterhin könnte das System auf Wunsch des Autors Untergruppen automatisch einklappbar gestalten, was besonders für größere Untergruppen interessant ist.

Auch wenn die Forderung nach einem Wegfall frei verwendbarer Beschriftungen in Drag&Drop-Editoren zunächst unrealistisch klingen mag, so haben die Beispiele deutlich gezeigt, dass dies durchaus mit einem sehr positiven Effekt für alle Autoren und Nutzer umsetzbar ist. Es bringt keine Nachteile für Autoren von Bedienoberflächen, sondern erleichtert ihnen sogar die Arbeit bei gleichzeitiger Erhöhung der Qualität ihrer Arbeit. Natürlich bedeutet diese Maßnahme nicht, dass Labels generell nicht frei verwendbar sein sollen. Das Programmiersprachen-Konstrukt „Label“ soll und muss auch zur Umsetzung der beschriebenen Konstrukte erhalten bleiben. Es soll lediglich in GUI-Editoren nicht oder zumindest um vieles schlechter zugänglich sein. Programmierer benötigen Labels natürlich weiterhin zur Gestaltung neuer Widgets und eventuell komplexerer Oberflächen. Es soll lediglich für jeden, der auf einfache Weise mit einem GUI-Editor eine Oberfläche erzeugen will, die Möglichkeit geschaffen werden, ohne spezielle Kenntnisse und Maßnahmen direkt gut gestaltete und damit zugängliche GUIs zu erzeugen. Im Allgemeinen verwenden Menschen zunächst immer die Gestaltungselemente- und -möglichkeiten, die für sie am leichtesten zugänglich sind. Bringt ein Eingabefeld also gleich eine Beschriftung mit, die auch noch leicht zu positionieren ist, so besteht keine Notwendigkeit ein weiteres Label zu verwenden. Sind Labels auch nicht in der Elementauswahl des Editors auffindbar, so werden sich auch nur einige die Mühe machen, Labels zu nutzen, wenn die angestrebte Funktionalität auch anders leicht erreicht werden kann. Deshalb sollten in einer Elementauswahl möglichst auch semantische Elemente leicht verfügbar sein. Ausgehend von den Gestaltungsvorschriften für Gruppierungen und vor allem auch der Bindung von Layoutoptionen an Gruppierungen ist der Schritt zur Verwendung semantischer Widgets, also solcher Widgets, die der Autor zunächst nach ihrer Funktion und nicht nach ihrer Gestaltung aussucht, nicht mehr weit. So kann dem Autor beispielsweise ein Auswahl-Widget geboten werden, bei dem über Optionen für Mehrfachauswahl, Layout und andere Darstellungsparameter leicht verschiedenste Darstellungen erreicht werden können. Solche Widgets können mit bestehenden Mitteln bereit umgesetzt und in Widget-Bibliotheken eingepflegt werden.

4.2.4 Struktur und Annotation belohnen – mehr Nutzen für alle

Struktur und Annotation in Darstellungen sollten nicht nur der Zugänglichkeit nützen, sondern allen Betrachtern einen Vorteil bringen. Beispielsweise die Darstellung von Überschriften als Navigationsleiste in Textverarbeitungsprogrammen zeigt, dass auf diese Weise Autoren leicht angeregt werden, zugänglichere Darstellungen zu produzieren. Navigationsfunktionen wie diese und auch die für angepasste Darstellungen in Kapitel 3 beschriebenen sollten für jeden Nutzer auf jeglichen strukturierten Darstellungen in Editoren wie auch und in reinen Betrachtungsprogrammen leicht verfügbar sein, um einen Bedarf nach Struktur und sinnvoller Benennung von Elementen zu wecken. Hat sich ein Autor beispielsweise die Mühe gemacht, in PowerPoint Gruppierungen anzulegen und diesen Titel zuzuweisen, so sollten diese Titel in Aufgabenbereichen wie etwa „Auswahl und Sichtbarkeit“¹⁹ auch angezeigt werden und nicht, wie Abbildung 119 zeigt, nur die nichtssagenden IDs der Gruppen.

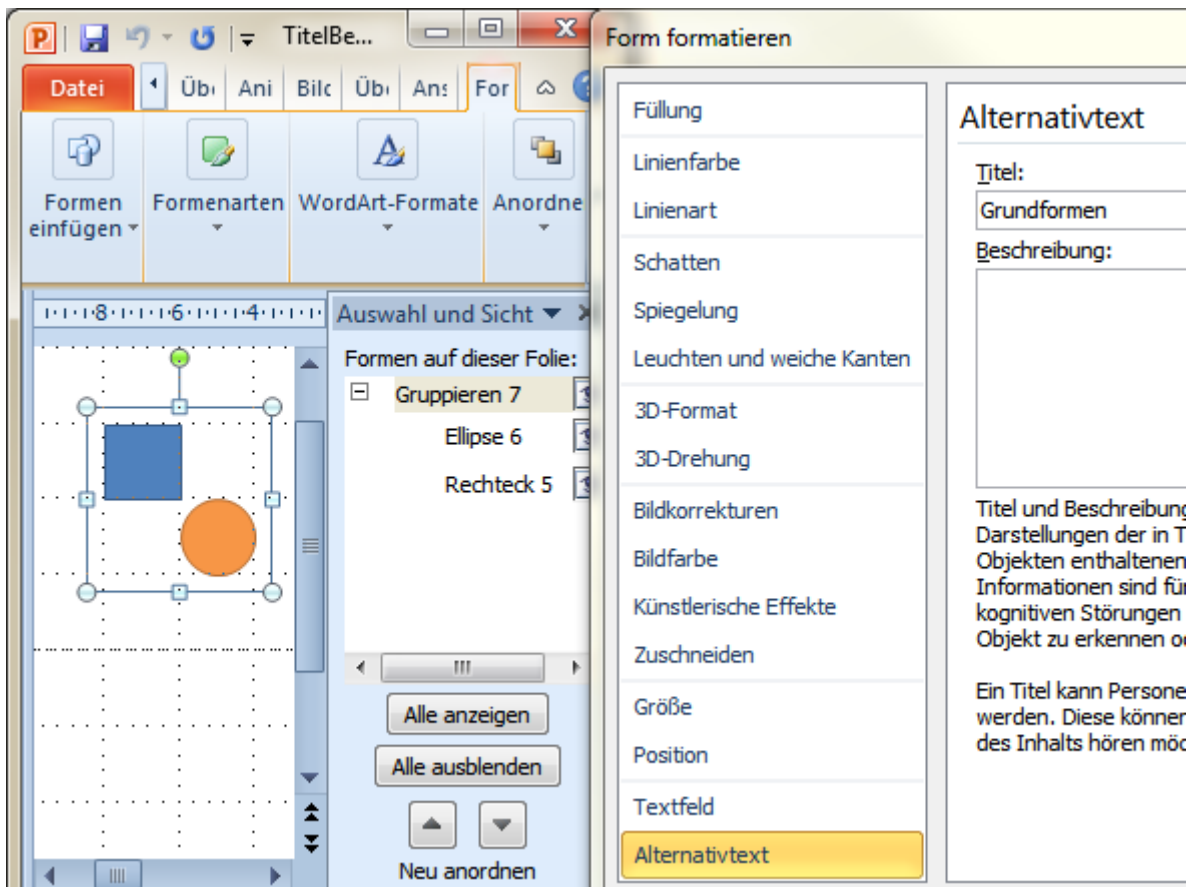


Abbildung 119 Screenshot aus PowerPoint als Beispiel zu besser nutzbaren Zugänglichkeitsattributen. Der Screenshot zeigt links eine PowerPoint-Folie mit gruppierten Formen (blaues Quadrat und oranger Kreis) und die zugehörige Ansicht des Aufgabenbereichs „Auswahl und Sichtbarkeit“. Leider werden in diesem Bereich nur die IDs der Formen und Gruppierung angezeigt (Gruppierung 7, Ellipse 6 und Rechteck 5) und nicht die im Bereich „Alternativtext“ des Dialogs „Form formatieren“ vergebenen Titel. Der rechte Teil der Abbildung zeigt diesen Dialog beispielhaft für die gewählte Gruppierung. Als Titel wurde „Grundformen“ vergeben. Dieser Titel wäre auch für den Foliensautor eine hilfreichere Angabe als die ID „Gruppieren 7“.

¹⁹ Der Aufgabenbereich "Auswahl und Sichtbarkeit" kann in PowerPoint 2010 angezeigt werden durch Auswahl einer Form und Klick in der Registerkarte "Format" in der Gruppe "Anordnen" auf "Auswahlbereich".

4 Förderung der zugänglichen Gestaltung

Eine schnelle Navigation zwischen Überschriften sollte standardmäßig in jedem Webbrowser zur Verfügung stehen. Auch das Einklappen von Bereichen unter einer Überschrift sollte möglich sein. Dadurch wird ein stärkerer Bedarf nach sinnvoller Einteilung von Darstellungen in semantisch zusammenhängende Bereiche geweckt. Sind Bedienelemente mit Beschriftungen verknüpft, können Accesskeys automatisch erzeugt werden. Und auch die Tabulatorreihenfolge kann vom System selbst immer sinnvoll gesetzt werden, solange ein Autor nichts Besonderes angibt. Viele weitere solcher Beispiele existieren, die häufig einfach umsetzbare Lösungen haben. So wäre beispielsweise zur Navigation durch Überschriften per Tab in einem Webbrowser einzig die Ergänzung des Attributs „`tabindex`“ bei den Überschriften nötig. Dies könnte von Webseiteneditoren und bei Export aus Textverarbeitungsprogrammen automatisch geschehen. Der folgende Abschnitt beschreibt eine konkrete Umsetzung solcher Maßnahmen für SVG.

4.3 Nutzung von SVG-Zugänglichkeitsattributen zur Webseitengestaltung

Langfristig kann die Qualität von SVG nur nachhaltig verbessert werden, wenn wie bei anderen Dokumenten und grafischen Oberflächen auch entsprechende Unterstützung durch Editierprogramme besteht. Entwickler solcher Programme arbeiten aber primär an dem, was die Nutzer auch als wichtig empfinden. Bei den Nutzern bzw. Autoren von SVG-Grafiken muss also zunächst einmal ein Bedarf geweckt werden. Sie müssen die Möglichkeit erhalten, zu erkennen, was mit gut aufgebauten SVGs möglich ist. Einige Beispiele im Web zeigen bereits, wie nützlich SVG für interaktive Webseiten sein kann (siehe z.B. [Bal09] und [Der05]). Allerdings sind dies immer mühevoll umgesetzte Einzelanwendungen. Und häufig werden auch genau nur die Konstrukte verwendet, die für die entsprechende Anwendung entscheidend sind. Gerade Annotationen sind nicht vorhanden, selbst wenn die Grafiken zur Darstellung eines Datenbestands dienen und offensichtlich auch aus diesem generiert wurden. Um Nutzern und Autoren die Möglichkeit zu geben, die Gestaltungsmerkmale schätzen zu lernen, die die Zugänglichkeit von SVG erhöhen, müssen genau diese auf einfache Weise umfassend nutzbar sein. Zu diesem Zweck wurde im Rahmen dieser Arbeit ein JavaScript-Framework entwickelt, das es ermöglicht, aus SVG-Dateien unter Ausnutzung von gruppierenden Elementen und Annotationen mit wenigen Codezeilen interaktive Webseiten zu erzeugen [TSSE09]. Vorbild der Funktionen des Frameworks waren dabei, die Möglichkeiten, die Blinden durch SVG4Blind [ROE04] zur Exploration von SVG gegeben werden, sowie die Funktionen, die in Beispielen wie den oben genannten umgesetzt wurden. Das Framework richtet sich an Webseitenautoren, die zumindest über grundlegende Kenntnisse in XML, CSS und JavaScript verfügen.

Ein wesentlicher Punkt des Frameworks ist dabei die Trennung von Grafik (und Grafikanimation) von dokumentbezogener Interaktion und Darstellung. Dies ermöglicht die Verwendung ein- und derselben Grafik in verschiedensten Webseiten zu verschiedensten Zwecken. Beispielsweise kann eine SVG-Deutschlandkarte an einer Stelle zur Erläuterung der Aufteilung Deutschlands in Bundesländer dienen, an einer anderen zur Information über lediglich ein einzelnes Bundesland und an einer dritten Stelle zur Erkundung der Flüsse oder Landschaften. Der Grafikdesigner muss nur eine Grafik erstellen. Der Webseiten-Designer kann diese dann flexibel in verschiedenen Webseiten einsetzen. Je mehr Einsatzmöglichkeiten für eine SVG-Grafik bestehen, desto eher lohnt sich eine gute Aufbereitung. Das Framework integriert zwar Interaktion in SVG-Grafiken und verändert eventuell deren Darstellung. Dies erfolgt aber erst, nachdem die SVG-Datei zur Laufzeit automatisch durch das Framework in die Webseite geladen wurde. Die ursprüngliche SVG-Datei wird nicht verändert.

Das Framework benutzt die JavaScript-Bibliothek „jQuery“ [jQu] und deren Plugin „jQuery SVG“ [Woo] und arbeitet über die vom W3C definierten DOM-Schnittstellen [Worg]. So kann es in verschiedenen Browsern und mit verschiedenen SVG-Viewern verwendet werden. Zwar zeigte sich bei der Entwicklung, dass trotz Verwendung angeblich browserunabhängiger Bibliotheken einige Besonderheiten der unterschiedlichen SVG-Umsetzungen beachtet werden

4 Förderung der zugänglichen Gestaltung

müssen. Dies wird aber für den Nutzer des Frameworks nicht sichtbar. Das Framework kann auch mit weiteren JavaScript-Bibliotheken verknüpft werden, wie beispielsweise der Tooltips-Bibliothek von Walter Zorn²⁰. Auf die Integration bestehender Bibliotheken wurde bei der Entwicklung des Frameworks großer Wert gelegt, um einen Beitrag zur Verknüpfung der vielfältigen Entwicklungen, die im Internet kursieren, zu leisten.

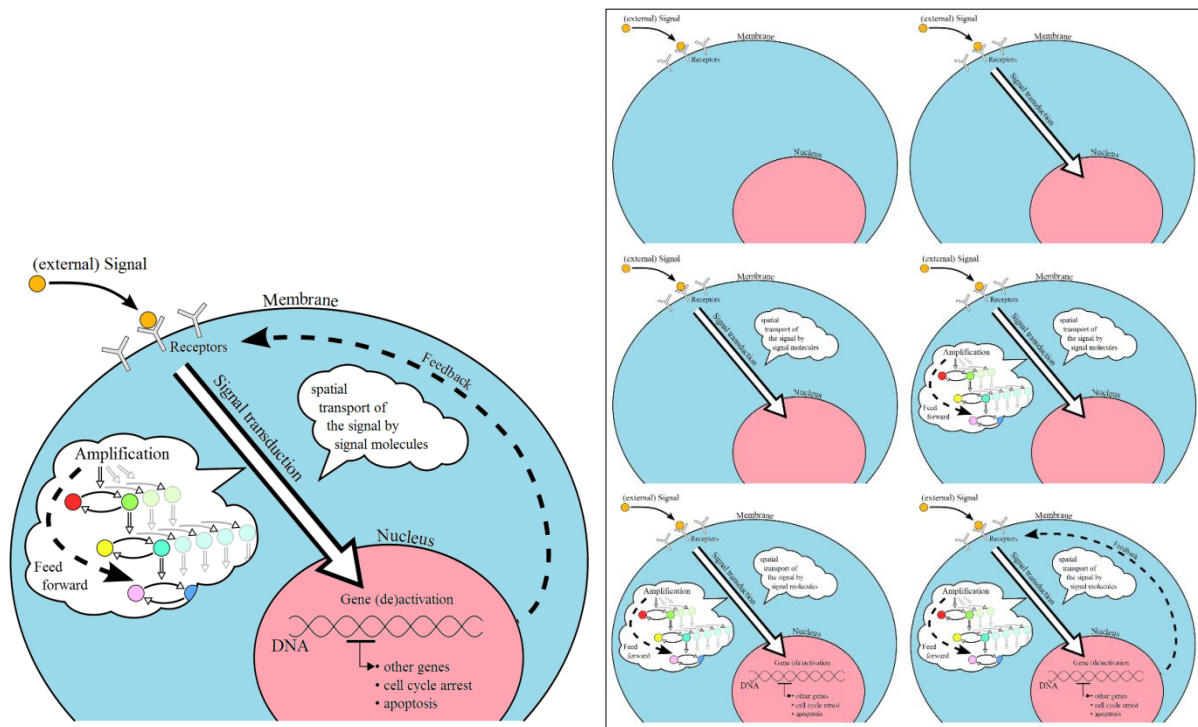


Abbildung 120 SVG-Darstellung zur Visualisierung der Signalausbreitung in einer Zelle
links: Gesamtdarstellung (von Martin Falk, VIS), rechts: schrittweiser Aufbau

SVG4Blind bietet den Nutzern die Möglichkeit, eine SVG-Grafik schrittweise aufzubauen und dabei zu jedem Element die passende Beschreibung anzuhören. Für die taktile Erfassung ist dies essentiell, da es hier durch den eher sequentiellen Aufnahmekanal wesentlich schwieriger ist, einen Gesamteindruck zu gewinnen. Aber auch andere Nutzer können davon profitieren. Beschreibungen zu einer Grafik tragen allgemein zum Verständnis der Grafik bei (besonders bei komplexeren Darstellung wie der Darstellung der Signalausbreitung in Zellen in Abbildung 120 links). Über die Elemente <title> und <desc> bietet SVG wie bereits in Kapitel 2.2 gezeigt, hervorragende Möglichkeiten, sogar mehrstufige Beschreibungen direkt in der Grafik abzulegen. So können diese einfach durch Weitergabe der Grafik-Datei jedem zur Verfügung gestellt werden. Es ist keine zusätzliche Datei nötig und somit kann die Beschreibung auch nicht verloren gehen. Mit Hilfe des entwickelten Frameworks können Texte aus <title>- und <desc>-Elementen in Webseiten als Teil des Textes oder als Tooltip integriert werden. Dabei ist auch eine Gliederung mit Hilfe von Überschriften möglich. Durch die automatische Integration ist es zudem möglich, bei Selektion eines Elements in der SVG-

²⁰ Leider verstarb Walter Zorn im Jahre 2009, weshalb seine Webseite nicht mehr zur Verfügung steht. Die letzte Version seiner Tooltip-Bibliothek konnte zum Zeitpunkt der Veröffentlichung dieser Arbeit unter <http://sourceforge.net/projects/wztip/> abgerufen werden.

4.3 Nutzung von SVG-Zugänglichkeitsattributen zur Webseitengestaltung

Grafik den dazugehörigen eingefügten Text hervorzuheben. So kann ein Betrachter die Verknüpfung zwischen Grafik und zugehöriger Beschreibung leichter herstellen. Umgekehrt, also von Text zu Grafik, ist die Interaktion ebenso umsetzbar. Dies alles ist natürlich nur möglich, wenn die Elemente den Annotationen auch zuordenbar sind und einzeln in ihrer Darstellung verändert werden können, wenn die Grafik also gut strukturiert ist. Sind die Elemente zusätzlich sinnvoll geordnet, so kann die Grafik, wie in Abbildung 120 rechts gezeigt, Stück für Stück dargestellt werden. Auch dies kann das Verständnis der Grafik fördern.

Durch den schrittweisen Aufbau kann die Aufmerksamkeit des Betrachters nach und nach auf die einzelnen Bestandteile gelenkt werden, die für sich eventuell leichter zu erfassen sind. Die zugehörigen Beschreibungen können genauso selektiv angezeigt werden, um den Leser nicht mit einer Fülle von Text zu überfordern. Besonders im E-Learning-Bereich kann dies sehr nützlich sein. Die für eine solche Exploration nötigen Bedienelemente können durch das Framework automatisch in die Webseite eingefügt und mit den nötigen Interaktionen verknüpft werden. Dafür sind nur wenige Zeilen und kaum Programmierkenntnisse nötig, wie Listing 22 zeigt. Der Webseitenautor muss sich weder um die korrekte Einbindung der SVG-Datei kümmern noch um die Behandlung der Eingabeereignisse des Benutzers. Die Exploration erfolgt auf Grundlage der Gruppen-Elemente, wobei die Hierarchiestufe angegeben werden kann, auf der der schrittweise Aufbau erfolgen soll. Abbildung 121 zeigt die durch Listing 22 erzeugte Webseite mit einer strukturierten Version des Versuchsaufbaus aus Abbildung 112 nach Einblendung der Glasschale.

```
<html><head>
  <title>Versuchsaufbau</title>
  <script type="text/javascript" src="jquery-1.3.min.js"></script>
  <script type="text/javascript" src="jquery.svg.js"></script>
  <script type="text/javascript" src="vis_svg_framework.js"></script>
  <script type="text/javascript">
    function afterSVGsLoaded() {
      generateExplorationForm("versuch_svg", 1); }
  </script>
  <style type="text/css">@import "jquery.svg.css";</style>
</head><body>
  <h1 style="text-align:center">Versuchsaufbau</h1>
  <div style="float:left">
    <div class="svg" id="versuch_svg" style="width:90mm; height:50mm;">
      Versuchsaufbau1_strukturiert.svg</div>
    <form id="exploreForm" name="exploreForm" action=""></form>
  </div>
  <div id="titleDescTemplate">
    <h2 class="title">Titel</h2><p class="desc">Desc</p>
  </div>
</body></html>
```

Listing 22 Quellcode einer Webseite zur Exploration einer SVG-Grafik

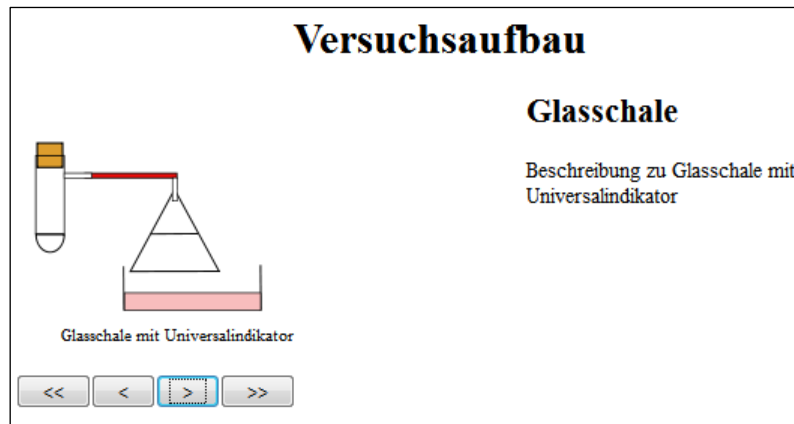


Abbildung 121 Screenshot der durch Listing 22 erzeugten Webseite während der Exploration

Auch Checkboxes zum Ein- und Ausblenden einzelner Elemente oder Gruppen, können erzeugt werden. Dabei können zur Angabe von Gruppen nicht nur die strukturellen Gruppierungs-Elemente von SVG genutzt werden, sondern auch semantische Gruppierungen, die durch CSS-Klassen definiert sind. So kann man beispielsweise in einer Deutschlandkarte für jedes Bundesland eine SVG-Gruppe anlegen, die neben der Begrenzung des Bundeslandes auch alle darin befindlichen Kreise und Städte enthält und trotzdem alle Städte über die CSS-Klasse „Stadt“ gruppieren. Die CSS-Klasse ist also nicht nur ein Hilfsmittel für die Gestaltung der Elemente. Deshalb sollten hier sinnvolle Benennungen verwendet werden. Durch das entwickelte Framework wird auch dies gefördert. Zur Anwendung von Funktionen des Frameworks können die Elemente, die in einer Funktion bearbeitet werden sollen, durch ihre IDs oder Klassen-Namen angegeben werden. Sind diese sinnvoll benannt, so hilft dies nicht nur beispielsweise bei der taktilen Exploration, sondern macht es auch dem Webseitenautor einfacher, sich an diese zu erinnern und somit sie zur Erstellung der Seite zu nutzen und zu erkennen, ob der geschriebene Quellcode sich auf die richtigen Elemente bezieht.

Die Funktionen des Frameworks, wie Ausblenden oder Hervorheben verschiedener Teile der SVG-Grafik, können natürlich auch ohne Benutzerinteraktion verwendet werden, um statische Webseiten zu erzeugen, die Teilaspekte einer SVG-Grafik nutzen. Wie Abbildung 122 oben zeigt kann eine SVG-Grafik auch mehrfach in einer Webseite genutzt werden. In diesem Beispiel wird bei Überfahren eines Bundeslandes mit der Maus in der Deutschlandkarte links das entsprechende Bundesland in einem zweiten Bereich rechts separat und vergrößert inklusive Name und Beschreibung dargestellt. Auch die Kombination von mehreren SVGs oder das Einfügen einzelner SVG-Elemente ist möglich. So können beispielsweise besondere Elemente zur Markierung eingefügt werden, wie etwa ein Standort-Symbol in einer Karte oder ein Hinweispfel. Weitere Funktionen des Frameworks sind die Anzeige eines Tooltips bei Mausbewegung über ein Element und die Generierung von Eingabedialogen mit Auswertungsfunktion auf Basis von Text-Elementen aus dem SVG (siehe z.B. Abbildung 122 unten). Dies fördert die Einbindung von Text über das SVG-Text-Element anstatt als Path-Element (also lediglich grafische Repräsentation). Zur Erstellung einer solchen Quizseite muss wieder nur eine einzige Funktion des Frameworks aufgerufen werden mit Angabe der Elemente, die abgefragt werden sollen, und der zugehörigen Frage. Die Texte der abzufragenden Elemente

4.3 Nutzung von SVG-Zugänglichkeitsattributen zur Webseitengestaltung

gelten als richtige Antwort und werden natürlich bei Start des Quiz ausgeblendet. SVG-Grafiken können so z.B. auch für Selbsttests im E-Learning-Bereich eingesetzt werden.

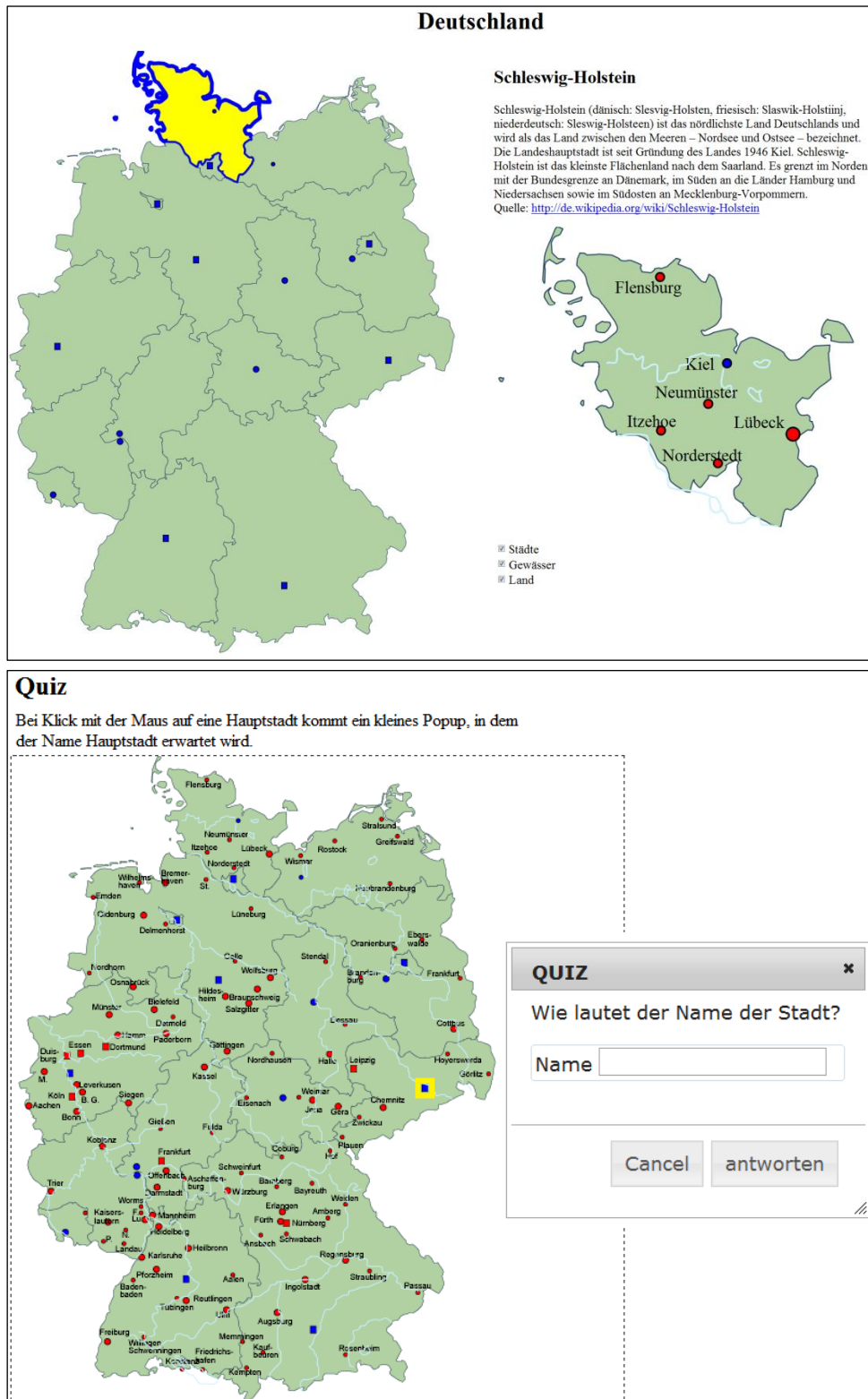


Abbildung 122 Zwei weitere Anwendungsbeispiele des JavaScript-SVG-Frameworks links: interaktive Deutschlandkarte zur Information über Bundesländer rechts: Deutschlandkarte mit Hauptstadt-Quiz

4.4 Allgemeine Richtlinien zur Förderung zugänglicher Gestaltung

Aus den gezeigten Beispielen und Lösungsmöglichkeiten lassen sich einige allgemeine Richtlinien zur Förderung der zugänglichen Gestaltung formulieren:

1. Vordefinierte Komponenten, die eventuell auch aus anderen Strukturen automatisch erstellt werden, müssen aus sich selbst heraus so zugänglich wie möglich sein. In bestehenden Widgets und generierten Oberflächen und Dokumenten müssen also alle nötigen Maßnahmen zur Sicherstellung der Zugänglichkeit bereits umgesetzt sein. Auch Hilfsfunktionen, die dem Benutzer zu Gestaltung angeboten werden sollten immer zu einer möglichst zugänglichen Gestaltung führen. So ist es zwar sinnvoll, Aufgrund von Ausrichtungen verschiedener Bedienelemente aneinander, eine Gruppierung durchzuführen. Dies muss aber auch auf sinnvolle Weise geschehen und darf beispielsweise nicht Elemente aus verschiedenen Gruppen einander zuordnen.
2. Die Gestaltungsfreiheiten können für normale Benutzer oftmals stärker eingeschränkt werden, als dies bisher geschieht. Wann immer möglich sollte eine Gestaltung durch Auswahl aus vorgegebenen Optionen einer völlig freien Gestaltung gegenüber bevorzugt werden. Dies soll nicht bedeuten, dass keine freie Gestaltung mehr möglich sein darf. Die dazu nötigen Interaktionskomponenten sollten aber schwerer erreichbar sein als andere. So sollte beispielsweise in Word die Anwendung einer Formatvorlage wesentlich einfacher sein als die direkte Veränderung von Schriftart, -größe und -farbe. Mit Word 2007 wurde hier zwar gegenüber früheren Versionen schon eine Verbesserung erreicht, da die Formatvorlagen zumindest im gleichen Ribbon angezeigt werden, wie die Bedienelemente zur Änderung der Schrift und somit auch sofort sichtbar sind. Allerdings wurde auch ein Kontextmenü eingeführt, das die Schrift-Änderungs-Bedienelemente dem Benutzer näher bringt als die die zur Angabe einer Formatvorlage. So wird der Nutzer verleitet, die Schrifteinstellungen direkt zu verändern, was weder für ihn sinnvoll ist, da so spätere globale Änderungen nicht möglich sind, noch der Zugänglichkeit des Dokuments dient.
3. Für Elemente, die in einer gewissen Art und Weise oder in Kombination mit anderen Elementen immer wieder auftreten, sollten vordefinierte Komponenten existieren, die die nötige Parametrisierung ermöglichen. Besonders bei komplexeren Konstrukten ist dies für die meisten Nutzer hilfreicher als umfangreiche Sammlungen an Einzelkomponenten, die beliebig kombiniert werden können. Bei Bedienelementen ist dies bereits Gang und Gäbe, wenn auch noch nicht gänzlich durchgezogen, wie z.B. die fehlende Verbindung von Eingabefeldern mit Beschriftungen zeigt. Dass dies auch bei Grafiken gut möglich und vom Nutzer gewünscht ist, zeigen Programme wie SmartDraw (<http://www.smartdraw.com/>) oder die SmartArts von Microsoft.

4.4 Allgemeine Richtlinien zur Förderung zugänglicher Gestaltung

4. Wie durch Abbildung 111 deutlich gezeigt wurde, sollten Konstrukte, die visuell gruppierend wirken, strukturell dieselbe Gruppierung bewirken. Visuelle Gruppierung sollte also nicht ohne strukturelle Gruppierung möglich sein. Gibt es keinen direkten Zusammenhang zwischen visueller und struktureller Gruppierung, kann ein Autor nicht erkennen, dass die Struktur seines erstellten Inhalts eventuell nicht korrekt ist.
5. Analysefunktionen zur Erkennung von Zugänglichkeitsproblemen sollten, ähnlich wie die Rechtschreibprüfung in modernen Editoren, im Hintergrund immer ablaufen und gefundene Probleme markieren. Eine entsprechende Übersicht sollte für den Benutzer leicht abrufbar oder gar zumindest in Kurzform immer sichtbar sein, wie dies auch bei Rechtschreibprüfung und Wortstatistik üblich ist. Solche Analysefunktionen sind zwar besonders im Webbereich durchaus vorhanden, müssen aber meist separat angestoßen werden. Oft führen sie nur sehr einfache Analysen durch wie beispielsweise die Kontrolle, ob Bildern Alternativtexte zugeordnet wurden. Dabei könnte mit Hilfe von Heuristiken z.B. auch erkannt werden, wenn Überschriften nicht als solche markiert wurden oder Tabellen zu Layoutzwecken anstatt zur Repräsentation von Daten verwendet wurden, wie beispielsweise [Spi07] zeigte. Zwar existieren zumeist nur Richtlinien für Webdokumente und -oberflächen, aber diese können zu einem Großteil auch auf andere Gebiete angewendet werden.
6. Sind zur Herstellung der Zugänglichkeit Benutzeraktionen nötig (wie etwa zur Angabe eines Alternativtextes), so sollten dem Benutzer genügend Informationen über das bestehende Problem und die Möglichkeiten zur Behebung gegeben werden (eventuell als Auswahl von Lösungen). Dabei sollten die Erklärungen kurz und allgemein verständlich gehalten werden. Erläuterungen wie *„Das World Wide Web Consortium sieht das align-Attribut jetzt als veraltet an. Es empfiehlt sich, neuere Konstrukte zu verwenden.“* (aus Microsoft Expression Web) helfen den wenigsten Nutzern weiter.
7. Jegliche Strukturangaben und Annotationen, die ein Nutzer direkt oder indirekt angegeben hat oder automatisch erzeugt wurden, dürfen bei Speicherung und Konvertierung in andere Formate nicht verloren gehen, sofern das Speicherformat irgendeine Möglichkeit bietet, diese Informationen abzulegen. Vielmehr noch sollten gerade Konvertierungsvorgänge genutzt werden, um Analysen durchzuführen und den Benutzer auf Zugänglichkeitsprobleme hinzuweisen. Zum einen muss bei Konvertierungen meist sowieso der gesamte Inhalt analysiert werden. Zum anderen werden Konvertierungen (wozu auch die Kompilierung von Bedienoberflächen gezählt werden kann) meist dann durchgeführt, wenn ein Dokument oder eine Oberfläche auch anderen zur Verfügung gestellt werden soll.

4 Förderung der zugänglichen Gestaltung

8. Aufgaben, die für den Benutzer bei sinnvoller Ausführung aufwändig sind, sollten nicht zu oft von ihm abverlangt werden. Beispielsweise ist die Abfrage eines Alternativtextes zu einem Bild direkt bei Einfügen in ein Dokument nicht unbedingt sinnvoll. Häufig werden Bilder nur testweise eingefügt. Erscheint dann ein Abfragedialog fühlen sich die meisten Benutzer nur belästigt und schließen ihn einfach oder fügen gar einen nicht sinnvollen Alternativtext ein („um dem System zu genügen“). Sinnvoller ist die Abfrage des Alternativtextes in dem Moment, wo der Benutzer eine Bildbeschreibung einfügt. Beides kann dann im gleichen Dialog erfragt werden. Dies wird zwar nicht immer ausgeführt, ein fehlender Alternativtext sollte aber durch eine Analyse erkannt und markiert werden.
9. Verbesserungen der Zugänglichkeit sollten allen Benutzern zu Gute kommen und möglichst direkt erkennbar sein. Je mehr Vorteile zugängliche Gestaltung auch dem normalen Nutzer bringt, desto eher werden die dazu nötigen Tätigkeiten durchgeführt, ohne sie als Zusatzarbeit zu empfinden. Dies sieht man beispielsweise an der Verwendung von Formatvorlagen für Überschriften. Da diese direkt zur Erzeugung von Inhaltsverzeichnissen und Gliederungsansichten dienen, hat sich deren Verwendung bereits weitgehend durchgesetzt.
10. Die Beteiligung von Betroffenen und deren Vertretergruppen muss gefördert werden. Blinde und Sehbehinderte selbst wissen am besten, welche Unterstützung sie benötigen und wo Verbesserungen nötig wären. Ihre Vertretergruppen setzen sich meist intensiv mit Zugänglichkeitsaspekten auseinander. Deshalb sollte es diesen ermöglicht werden, sich an der Verbesserung der Zugänglichkeit zu beteiligen. Auch wenn dies eventuell nur durch gezielte Problemmeldungen geschehen kann, ist es hilfreich für alle Beteiligten. Auch hier müssen Mittel geschaffen werden, wie solche Meldungen möglichst einfach abgegeben werden können. Müssen längliche Formulare ausgefüllt werden oder erst Menüs durchsucht werden, um eine Stelle zu finden, an der man ein Feedback abgeben kann, so wird sich die Beteiligung in Grenzen halten oder ist evtl. aufgrund weiterer Zugänglichkeitsprobleme gar nicht möglich.

4.5 Fazit

In diesem Kapitel wurde gezeigt, dass einige Möglichkeiten bestehen, die Zugänglichkeit von grafischen Oberflächen zu verbessern. Dabei sollten immer die Werkzeuge im Vordergrund stehen, statt der letztlich damit erstellten Inhalte. Sind die Werkzeuge gut entwickelt, ergeben sich auch leichter zugängliche Inhalte. Vor allem in Anbetracht von Entwicklungen wie dem Web 2.0, bei denen jeder die Möglichkeit hat, auf einfache Weise neue Inhalte bereitzustellen, sollte nicht erwartet werden, dass die Autoren selbst immer genügend Kenntnisse über die Problematik der Zugänglichkeit haben. Zugängliche Gestaltung wird sich nur ausbreiten, wenn sie automatisch durch die Editierprogramme geschaffen wird oder für den Autor deutlich sichtbare Vorteile bringt. Hier sind auch die Vertretergruppen von Blinden und Sehbehinderten aufgefordert, sich stärker zu engagieren. Häufig führen diese zwar Untersuchungen durch, welche Webseiten und Anwendungen wie zugänglich sind. Dies resultiert aber meist nur in einer händischen Verbesserung der bemängelten Dokumente und Oberflächen. Die eigentlichen Ursachen werden oft nicht gesucht und können damit auch nicht bekämpft werden.

Eine besondere Herausforderung dabei ist die Einbringung von semantischen Informationen. Die Erstellung einer Oberfläche, eines Dokuments oder einer Grafik ist immer schon eine Transformation der Gedanken und Absichten des Autors in die vorgegebenen Gestaltungsmöglichkeiten. Dabei gehen gerade bei abstrakten Grafiken oft schon semantische Informationen verloren. In grafischen Oberflächen kommt es auch häufig vor, dass Entwickler eine bestimmte Kombination von Widgets zur Lösung einer Aufgabe nur einsetzen, weil zur Entwicklungszeit keine besseren Mittel zur Verfügung standen oder sie nicht darüber informiert waren. Ist in der Repräsentation nicht erkennbar, welchem übergeordneten Zweck die Elemente dienen, so ist es auch nur schwer möglich, auf bessere Konstrukte umzustellen, wenn spezifizierende Dokumente fehlen und der Entwickler nicht mehr greifbar ist. Die Erhaltung der exakten Funktionalität steht meist im Vordergrund, auch wenn dies nicht nötig wäre.

5 Ergebnisse der Arbeit und Ausblick

Grafische Darstellungen werden in unserem Leben für Ausbildung und die tägliche Arbeit immer wichtiger. Grafiken werden zur Darstellung verschiedenster Sachverhalte eingesetzt und fast alle Computerprogramme nutzen grafische Oberflächen. Die durch größer werdende Monitore und bessere Auflösungen steigende Komplexität von Bedienoberflächen und auch Dokumenten erschwert Nutzern, deren visueller Kanal eingeschränkt ist, zunehmend den Zugang zu digitalen Informationen. Gerade für diese Nutzer sind solche Informationsquellen aufgrund der Möglichkeit, ihre Darstellung an die eigenen Bedürfnisse anzupassen, aber sehr wichtig. Deshalb entstanden vor allem im Bereich des World Wide Web in den letzten Jahren auch umfangreiche Initiativen zur Verbesserung der Zugänglichkeit digitaler Dokumente und grafischer Oberflächen. Richtlinien zur zugänglichen Gestaltung wurden geschaffen, Gesetze wurden erlassen und Formate für grafische Darstellungen wurden verbessert. Trotzdem bestehen immer noch wesentliche Barrieren, die Betroffenen den Zugang erschweren. Wie diese Arbeit deutlich zeigte, muss das Ziel, zugängliche Inhalte zu schaffen von mehreren Seiten angegangen werden. Zum einen müssen die Anpassungen der gebotenen Inhalte an die Bedürfnisse des Nutzers weiter verbessert werden, zum anderen müssen aber auch Mittel geschaffen werden, die es auch Autoren ohne Kenntnisse über Zugänglichkeitsanforderungen ermöglichen, zugängliche Darstellungen zu erstellen.

Durch die intensive Beteiligung am Projekt „HyperBraille“ [hyp10] lag der Schwerpunkt dieser Arbeit auf der Entwicklung und Umsetzung von Konzepten für Darstellungen auf grafisch-taktilen Displays für Blinde. Im Laufe dieser Arbeit zeigte sich aber deutlich, dass diese Konzepte auch im Bereich der vergrößerten Darstellungen am Bildschirm für Sehbehinderte wesentliche Verbesserungen bringen können. Wie in Abschnitt 3.1 erarbeitet wurde, sind die Grundfragestellung und auch die konkreten Anforderungen an angepasste Darstellungen bei beiden Nutzergruppen sehr ähnlich. Eine stärkerer Verbindung der Entwicklungsarbeiten für Blinde und Sehbehinderte kann beiden Gruppen nur Vorteile bringen, wie beispielhaft in Abschnitt 3.2.4.1 durch die Adaption des eigentlich für die grafisch-taktile Ausgabe entwickelten Darstellungskonzeptes von HyperBraille für die vergrößerte Darstellung am Monitor demonstriert wurde. Besonders deutlich wurde dabei, dass auch für Sehbehinderte eine stärkere Unterscheidung zwischen verschiedenen Arbeitssituationen und auch verschiedenen Arbeitsbereichen für eine optimale Unterstützung nötig ist. Bei der Gestaltung der taktilen Ausgabe wurde dies intuitiv von vornherein beachtet, was sich wohl vor allem darin begründet, dass Blinde zum effizienten Arbeiten primär auf lesbare Ausgabe von Text angewiesen sind. Die grafische Information ist zwar für einige Arbeiten wichtig, steht aber nicht im Vordergrund. Bei der vergrößerten Darstellung am Bildschirm wurde diese Trennung wahrscheinlich aufgrund des Leistungsvermögens des Ausgabegerätes Monitor bisher nicht wirklich vollzogen. Dem Design der taktilen Ausgabe kamen hier die eingeschränkten Möglichkeiten der Ausgabegeräte zu Gute.

Das Darstellungs- und Bedienkonzept von HyperBraille und das in dieser Arbeit entwickelte Rendering-Framework für grafisch-taktile Displays sind ein erster Schritt hin zu einem gra-

5 Ergebnisse der Arbeit und Ausblick

fisch-taktilen Desktop für Blinde, auf dem auch Interaktionen wie Drag&Drop oder Freihandzeichnungen möglich gemacht werden können. Grafisch-taktile Displays können Blinden viele Vorteile bei der Arbeit am PC bringen. Sie ermöglichen die Übersicht über zweidimensionale Darstellungen wie etwa komplexere Tabellen oder Dialoge und auch mehrzeilige Textausgaben, die beispielsweise zum Umstellen einer mathematischen Formel sehr hilfreich sein können. Ihre Verbreitung wird sich aber nur erhöhen, wenn auch entsprechende Software vorhanden ist, über die sämtliche Arbeitsaufgaben am PC erledigt werden können. Durch die Etablierung flexibler und konfigurierbarer taktiler Widgets mit zugehörigen Bedienelementen kann die Entwicklung für grafisch-taktile Displays in die nächste Phase übergehen – weg von einzelnen Inselanwendungen, die fest mit dem System und dem jeweiligen Displays verbunden sind, hin zu umfangreichen Programmierschnittstellen. Dies ist eine wesentliche Grundlage für eine gute Softwareunterstützung. Wie Abschnitt 2.1.2 deutlich macht, sind aber auch weitere Arbeiten im Bereich der Hardware erforderlich – zum einen, um die Displays wirklich portabel zu gestalten und besonders ihre Leistungsfähigkeit weiter zu verbessern, zum anderen aber auch, um ihren Preis zu senken. Bei den derzeitigen Kosten für ein großflächiges grafisch-taktiler Display können sich nur wenige Einrichtungen zur Unterstützung von Blinden und Blinde selbst an der Entwicklung entsprechender Software beteiligen. Dies ist aber für eine Beschleunigung der Entwicklungen auf diesem Gebiet unerlässlich. Mit dem im Rahmen von HyperBraille entwickelten Nachfolger des „BrailleDis 9000“ wurde auch hier bereits ein wichtiger Schritt getan. Sowohl die Größe wurde deutlich verringert, wie Abbildung 123 zeigt, als auch die Produktionskosten.



Abbildung 123 Gehäuse des neuen grafisch-taktilen Displays der Firma Metec (rechts) im Vergleich zum BrailleDis 9000 (links)
Das neue Display wiegt lediglich 5 kg und kann in einem Rucksack transportiert werden.

Mit Hilfe gemeinsamer Plattformen für die grafisch-taktile Ausgabe und die vergrößerte Ausgabe am Bildschirm können, wie mit dem HyperReader-Magnifier demonstriert wurde, auch unabhängig von grafisch-taktilen Displays weitere Fortschritte auf diesem Gebiet erzielt werden. Wesentliche Prozesse, wie die Analyse und Aufbereitung der Bildschirminhalte und Navigationsfunktionen, können in gemeinsamen Komponenten gekapselt werden. So bringen Fortschritte bei der vergrößerten Bildschirmausgabe direkt auch Fortschritte für die gra-

fisch-taktile Ausgabe. Lediglich die Module zur Ausgabe der Darstellung an sich und die Interaktionsverarbeitung unterscheiden sich in gewissem Maße. Aber auch hier können durch weitere Verallgemeinerungen größere Gemeinsamkeiten geschaffen werden. So könnten für das Rendering Widgets verwendet werden, die sich je nach Ausgabeziel unterschiedlich darstellen. Die Widgets selbst müssten so nur einmal angelegt werden. Auch ihre Zusammenstellung für die Ausgabe müsste nicht mehrfach programmiert werden. Lediglich die Implementierung verschiedener Paint-Methoden für unterschiedliche Ausgaben wäre nötig.

Natürlich wird die taktile Ausgabefläche auch bei höherer Auflösung der Displays immer wesentlich weniger Informationen bieten als die Monitorfläche. Der taktile Kanal ist schlichtweg nicht so leistungsfähig wie der visuelle. Interaktionen wie Zoom und Filterung, wie sie in Abschnitt 3.2.2 vorgestellt wurden, können diesen Nachteil aber mindern. Zusätzlich können Darstellungen für Blinde und Sehbehinderte von den Entwicklungen im Bereich der Darstellungen für kleine Displays, wie sie an Smartphones vorkommen, profitieren. Hier wird letztlich das gleiche Problem gelöst. Informationen, die eigentlich für die Ausgabe an einem normalen Monitor aufbereitet sind, müssen auf einer verhältnismäßig kleinen Ausgabefläche erkennbar präsentiert werden. Auch hier werden Interaktionsmöglichkeiten und Darstellungsformen untersucht, die eine einfache Navigation ermöglichen und genügend Überblick über die dargestellten Inhalte bieten (siehe beispielsweise [Bau10]). Einige Konzepte aus diesem Forschungsgebiet können auch auf Darstellungen für Blinde und Sehbehinderte übertragen werden und auch umgekehrt sind Einflüsse möglich. So könnten Darstellungen von Webseiten auf Smartphones beispielsweise auch von verkürzten Darstellungen, wie beispielsweise den bei Screenreadern üblichen Überschriftenlisten, profitieren.

Ein besonderes Augenmerk lag bei dieser Arbeit auf der Zusammenarbeit zwischen Blinden, Sehbehinderten und Normalsichtigen. Alle Nutzergruppen sollen mit den gleichen Dokumenten und Anwendungen arbeiten können, um gemeinsam Inhalte zu erstellen und sich gegenseitig helfen zu können. Aus diesem Grund wurden im Rahmen von HyperBraille auch nicht spezielle Anwendungen für grafisch-taktile Displays entwickelt, sondern ein grafischer Screenreader, der am Bildschirm dargestellte Anwendungen analysiert und in nützliche taktile Darstellungen umsetzt. Neben den eigentlichen Inhalten sollten dabei zur Förderung der Kommunikation und des Verständnisses für einander auch die unterschiedlichen Darstellungswelten für den jeweils anderen gut zugänglich sein. Deshalb wurden ein spezielles Bildformat und einige darstellende Komponenten entwickelt, die es Sehenden ermöglichen, die grafisch-taktile Ausgabe am Monitor zu betrachten und dabei enthaltene Texte in Schwarzschrift darstellen zu können. Die dafür nötigen Komponenten des entwickelten Rendering-Frameworks können auch unabhängig vom HyperReader und einem grafisch-taktilen Display genutzt werden. Dies hilft zusätzlich die Entwicklung weiterer Darstellungs- und Interaktionskonzepte für die grafisch-taktile Ausgabe voranzubringen. Weiterhin wurde in dieser Arbeit der Umgang mit Farben besonders auf den monochromen taktilen Displays eingehend untersucht (siehe Abschnitte 3.2.3 und 3.3.1), um Blinden die Erläuterungen von Normalsichtigen, die sich häufig auf Farben beziehen, verständlich zu machen. Verschiedene Darstellungs- und Explorationskonzepte wurden erarbeitet und umgesetzt. Dabei zeigte sich

häufig, dass trotz aller Neuentwicklungen im Bereich grafischer Formate und deren Flexibilität die Rastergrafik-Darstellung immer eine wichtige Rolle bei der Darstellung von Inhalten am Bildschirm spielen wird. Die möglichst automatische Aufbereitung dieser Grafikform wird daher auch in den kommenden Jahren ein zentraler Bereich der Unterstützung blinder und sehbehinderter Computerbenutzer sein. Entwicklungen im Bereich der Bilderkennung für Robotik und Suchmaschinen können hier positiven Einfluss haben. Um Blinden die Möglichkeit zu geben, farbige Rastergrafiken mit aktuellen Hilfsmitteln auch selbst zu erstellen, wurde ein Grafikeditor für die Braillezeile entwickelt. Die positiven Erfahrungen mit dem Editor bei der Zusammenarbeit von blinden und normalsichtigen Partnern zur Gestaltung von Darstellungen im Projekt HyperBraille zeigen, dass sich eine Weiterentwicklung in diesem Bereich lohnt. Sinnvolle Erweiterungen wären die Ergänzung von grafischen Primitiven als vordefinierte Zeichenobjekte und eine Einbindung von Sprachsteuerung z. B. zur Benennung von Objekten (wie etwa in [Kur96]).

Die automatische Aufbereitung von Anwendungen und Dokumenten für angepasste Darstellungen für Blinde und Sehbehinderte kann nur gut gelingen, wenn der Computer möglichst viel von den aufzubereitenden Inhalten „verstehen“ kann. Voraussetzung dafür ist die sinnvolle Strukturierung und Annotation der Inhalte. Nicht jeder Autor ist aber Experte in zugänglicher Gestaltung und sollte es auch nicht sein müssen. Wie Kapitel 4 zeigte, sind an vielen Erstellungs-, Konvertierungs- und Darstellungsprogrammen noch Verbesserungen möglich, die auch einem Laien die Erstellung gut zugänglicher Materialien erlauben, ohne ihn dabei übermäßig zu belasten und so in seiner eigentlichen Arbeit zu behindern. Dabei sind die nötigen Änderungen oftmals gering, wie beispielsweise die einfache Erhaltung von Benennungen und Beschreibungen bei Konvertierung zwischen verschiedenen Formaten. Viele der für Blinde und Sehbehinderte vorgestellten Darstellungs- und Interaktionskonzepte können auch für andere Benutzer hilfreich sein, wie etwa eine leichtere Navigation in Webseiten und großen Diagrammen. Werden diese Konzepte allgemein zugänglich gemacht, so steigert dies den Bedarf nach Struktur und Annotation bei allen Nutzern und fördert somit die Zugänglichkeit. Das in Abschnitt 4.3 vorgestellte SVG-Framework zeigt beispielhaft eine entsprechende Umsetzung. Entwicklungen wie diese bringen nicht nur Blinden und Sehbehinderten Vorteile, sondern auch vielen anderen Nutzergruppen. So ist die Annotation von Grafiken und die Darstellung dieser Annotation im Zusammenhang mit den zugehörigen Elementen der Grafik zum Beispiel auch für Lernbehinderte sehr nützlich, bei sehr komplexen Grafiken sogar für alle Betrachter.

Zusammenfassend konnte in dieser Arbeit gezeigt werden, dass eine Vielzahl grafischer Darstellungen mit Hilfe guter Aufbereitung auf grafisch-taktilen Displays mit höherem Informationsgehalt bzw. Leichter verständlich als auf Braillezeilen präsentiert und so der Arbeitsalltag Blinder weiter erleichtert werden kann. Zudem wurde deutlich, dass durch grafisch-taktile Displays die Welten der Computerdarstellungen für Sehbehinderte und Blinde enger zusammen wachsen und sich gegenseitig positiv beeinflussen können und sollten. Vor allem konnte aber herausgestellt werden, dass die eigentliche Problematik meist nicht in der Aufbereitung grafischer Darstellungen selbst liegt. Moderne GUI- und Grafik-Formate und

-Schnittstellen bieten hier umfassende Möglichkeiten, sofern das aufzubereitende Material die nötigen Informationen enthält. Die eigentliche Herausforderung besteht darin, die bereitgestellten GUIs und Dokumente von vornherein möglichst zugänglich zu gestalten. Dazu müssen Editoren und Konvertierungswerkzeuge Autoren wesentlich besser bei der Gestaltung von Dokumenten und Benutzungsschnittstellen im Sinne der Zugänglichkeit unterstützen, als es derzeit der Fall ist. Auch die Nutzung von Zugänglichkeitsattributen zur allgemeinen Verbesserung von Darstellungen und Unterstützung der Nutzer bei der Arbeit mit komplexen Anwendungen sollte deutlich ausgebaut werden, um zugängliche Gestaltung für jeden sinnvoll zu machen und damit ihre Verbreitung zu fördern. Dieser Aufgabenbereich sollte zukünftig stärker in die Forschung einbezogen werden, um Konzepte, wie sie in Abschnitt 4.2 vorgestellt wurden, weiter auszubauen und in Alltagssoftware einfließen zu lassen.

Anhang

Anhang A Fragebogen zur visuellen Darstellung der DFB-Farben

Um die Eignung der visuellen Farben, die für detailliertes Figur-Braille (DFB) gewählt wurden, nachzuweisen, wurde eine Online-Umfrage durchgeführt. Diese sollte zeigen, ob die Farben wie erwartet benannt werden, so dass die Kommunikation zwischen Sehenden und Blinden reibungslos funktionieren kann. Der Fragebogen wurde insgesamt von 239 Personen aufgerufen, wovon 143 tatsächlich an der Umfrage teilgenommen haben. Von 113 Personen wurde der Fragebogen vollständig bearbeitet. Die restlichen 30 Teilnehmer haben den Fragebogen nach Beantwortung von einer bis 26 Fragen abgebrochen. Vor der Online-Umfrage wurde eine Vorstudie mit fünf Probanden durchgeführt. Aufgrund der dabei gewonnenen Erkenntnisse, wurden die Einführungs- und Befragungstexte leicht geändert und die Befragung leicht verkürzt. Im Folgenden werden der Inhalt und die Gestaltung des Fragebogens beschrieben und die Ergebnisse der Umfrage dargestellt.

A.1 Geprüfte Farben

In der Befragung wurden insgesamt 31 Farben geprüft. Diese waren die 21 bunten Farben und die drei Grautöne, die für die DFB-Codierung festgelegt wurden, sowie Varianten zu 7 dieser Farben. Varianten wurden zu den Farben geprüft, bei denen sich in einer Vorstudie mit 5 Probanden zeigte, dass die intuitive Benennung zum großen Teil nicht wie erwartet erfolgte. Schwarz und Weiß wurden nicht getestet. Tabelle 13 zeigt die Varianten. Die übrigen 24 DFB-Farben können in Tabelle 10 in Abschnitt 3.3.1.2.2 nachgeschlagen werden.

	Hell	Normal	Dunkel	
Rot		0°, 100%, 95% 255, 234, 0		
Gelb		55°, 100%, 100% 255, 234, 0		HSV RGB
Orange			30°, 100%, 100% 255, 128, 0	HSV RGB
			25°, 100%, 100% 255, 106, 0	HSV RGB
Violett	270°, 50%, 100% 191, 127, 255	270°, 100%, 100% 127, 0, 255	270°, 100%, 50% 63, 0, 127	HSV RGB

Tabelle 13 Übersicht der als Alternativen verwendeten Farben in der Online-Umfrage zu DFB-Farben

A.2 Gestaltung des Fragebogens

Zur Gestaltung des Fragebogens wurde die Befragungssoftware „oFb“ in Version 2.0 verwendet [Lei10]. Diese wurde gewählt, da sie ein werbefreies, neutrales Layout mit weißem Hintergrund bot. So wurden die Probanden nicht durch andere Farben oder gar durch sich bewegende Elemente abgelenkt. Außerdem konnten mit dieser Software alle Fragen wie gewünscht und sehr einfach umgesetzt werden.

Die Prüfung der Farben erfolgte in zwei Phasen:

1. Freitexteingabe von Farbnamen mit Auswahl der Helligkeit
2. Entscheidung, ob eine vorgegebene Zuordnung richtig erscheint

In der ersten Phase wurden kurze Freitexteingaben zugelassen ohne dabei irgendwelche Farbbezeichnungen vorzugeben. Dies sollte zeigen wie die Probanden die Farben im natürlich-sprachlichen Umgang mit anderen benennen würden. Da sowohl normale, wie auch helle und dunkle DFB-Farben geprüft wurden, wurde eine vorgegebene Auswahl für die Helligkeit präsentiert, um den Probanden Schreibaufwand zu ersparen und eine Auswahl der Helligkeit zu erzwingen.

In der zweiten Phase wurden Farben und deren Zuordnung vorgegeben. Die Probanden sollten die Zuordnung bewerten. Dies sollte einerseits als Gegenprobe zur ersten Phase dienen. Andererseits sollte dies die Situation simulieren, dass der Proband mit den DFB-Farbnamen vertraut ist und nur diese zur Benennung von Farben anwendet. Man kann davon ausgehen, dass Sehende bei häufigerem Kontakt mit Blinden Kenntnisse über deren Werkzeuge besitzen und sich so auch an das DFB-Farbsystem anpassen würden.

Die Farben wurden jeweils einzeln auf einer Seite gezeigt und es war nicht möglich, auf eine vorherige Seite zurückzukehren. So sollten die Probanden davon abgehalten werden, ihre Entscheidungen über eine Farbe aufgrund einer anderen zu revidieren. Die Reihenfolge der Farben wurde zufällig gewählt, um Seiteneffekte aufgrund bestimmter Farbreihenfolgen möglichst gering zu halten. Die Farben wurden jeweils als Rechtecke mit ausreichend weißem Rand präsentiert. Abbildung 124 zeigt die Darstellung der Fragen in den beiden Phasen. Um die Umfrage kurz zu halten, wurden in der ersten Phase nicht alle Farben gezeigt. Aus 15 Farben, die sich in der Vorstudie als eher eindeutig erwiesen haben, wurde jeweils nur zufällig die Hälfte zur Abfrage gewählt. In der Vorstudie zeigte sich, dass eine zulange Abfrage eher zum Abbruch führt und dadurch der wichtige zweite Teil nicht mehr bearbeitet wird. Die Phasen konnten aber auch nicht getauscht werden, um die Probanden in ihren freien Farbangaben nicht zu beeinflussen. Die 15 Farben waren: Hellrot, Dunkelrot, Hellgelb, Normalgelb, Normalblau, Hellblau, Dunkelblau, Hellgrün, Dunkelgrün, Normalbraun, Hellbraun, Dunkelbraun, Hellgrau, Dunkelgrau. Die restlichen Farben wurden allen Probanden präsentiert. In der zweiten Phase wurden immer alle Farben angezeigt, da zur Beantwortung der Fragen nur wenig Zeit nötig war.

1. Welche Farbe hat das Rechteck?



- Hell
- Mittel
- Dunkel

1. Bitte geben Sie an, ob Sie mit der Farbzuoordnung einverstanden sind.



Blau, mittel

Die Zuordnung ist ...

- gut.
- nicht gut.

Abbildung 124 Darstellung der Fragen zur Evaluierung der DFB-Farben
links: Freitexteingabe, rechts: Entscheidung über eine vorgegebene Farbzuoordnung

Der Fragebogen wurde mit folgendem Erklärungstext eingeleitet:

Herzlich Willkommen beim Fragebogen zur Farbdarstellung an Bildschirmen

Bitte lesen Sie die folgende Erläuterung zum Fragebogen

Dieser Fragebogen zeigt Rechtecke mit Farben. Ihre Aufgabe ist es, die gezeigten Farben zu benennen oder anzugeben, ob eine Farbzuoordnung für Sie korrekt erscheint. Antworten Sie bitte spontan, also ohne lange darüber nachzudenken.

Es gibt kein richtig oder falsch!

Die Bearbeitung dauert etwa **5 Minuten**.

Ihr Monitor muss dafür nicht kalibriert sein. Es geht eben darum herauszufinden, wie die Farben an den verschiedensten Monitoren mit den verschiedensten Einstellungen wahrgenommen werden. Falls Sie Zugriff auf mehrere Monitore haben, können Sie den Fragebogen auch gerne mehrmals ausfüllen (einmal pro Monitor).

Die Datenerfassung ist anonym. Ihre IP wird nicht gespeichert. Persönliche Daten werden nicht abgefragt.

Diese Umfrage ist Teil meiner Dissertation zum Thema "Graphische Darstellungen für Blinde und Sehbehinderte".

Vielen Dank für Ihr Interesse

Danach wurde abgefragt, ob eine Farbsehschwäche vorliegt, um dies in der Auswertung beachten zu können. Daraufhin folgten die zwei Phasen der Befragung, die mit den folgenden Texten eingeleitet wurden:

Teil 1: Eingabe von Farbnamen

In diesem Teil sollen Sie Farbnamen als Freitext eingeben. Dabei sollen Sie auch die Helligkeit angeben mit "Hell", "Mittel" oder "Dunkel". Dazu tragen Sie den Farbnamen einfach in das passende Textfeld ein und gehen durch Drücken der Enter-Taste oder des Weiter-Buttons zur nächsten Frage.

Antworten Sie spontan! Verschiedene Farben können mehrfach vorkommen. Sie dürfen also gleiche Farbnamen auch mehrfach verwenden. Verwenden Sie am besten einfache Farbnamen wie "Rot" oder "Blau" und nicht "backsteinfarben" oder "Indigo".

ACHTUNG: Manchmal dauert das Anzeigen der nächsten Seite etwas länger. Drücken Sie die Enter-Taste oder den Weiter-Button nicht schnell mehrfach hintereinander. Sie könnten dann Seiten überspringen.

Teil 2: Zuordnungen bewerten

In diesem Teil sollen Sie angeben, ob eine vorgegebene Zuordnung Ihrer Meinung nach gut oder nicht gut ist. Nach Auswahl von "gut" oder "nicht gut" können Sie durch Drücken der Enter-Taste oder des Weiter-Buttons zur nächsten Frage wechseln.

ACHTUNG: Manchmal dauert das Anzeigen der nächsten Seite etwas länger. Drücken Sie die Enter-Taste oder den Weiter-Button nicht schnell mehrfach hintereinander. Sie könnten dann Seiten überspringen.

A.3 Ergebnisse der Umfrage

Tabelle 14, Abbildung 125 und Abbildung 126 zeigen die Ergebnisse der Umfrage. Dabei ist zu beachten, dass nicht jeder Proband tatsächlich alle Fragen ausgefüllt hat. Dargestellt werden zu den Freitextfragen jeweils die Übereinstimmung mit dem erwarteten Farbnamen, sowie mit der gesamten, erwarteten Farbangabe (Name und Helligkeit). Zur Auswertung wurden Freitexteingaben gleichen Inhalts zusammengefasst. Zunächst erfolgte lediglich eine Anpassung von Schreibweisen. Diese Werte werden in den Ergebnissen als „original“ bezeichnet. Die als „angepasst“ bezeichneten Ergebnisse entstanden durch eine Zusammenfassung nach der Bedeutung. Eine wesentliche Auswirkung hatte dies aber nur auf die Ergebnisse zur Farbe „hellrot“, da diese meist als „Rosa“ bezeichnet wurde. Auch bei den Entscheidungsfragen ergaben sich nur schlechte Werte für Hellrot. Dies ist wohl auf die Vorliebe für die Angabe „Rosa“ zurückzuführen. Bei „Dunkelgelb“, „Hellorange“ und „Dunkelorange“ wurde in den Entscheidungs- und ebenso den Freitextfragen nur selten die erwartete Antwort gegeben. Bei den Freitextfragen zeigen auch „Hellbraun“ und „Dunkelbraun“ schlechte Ergebnisse. Für alle anderen DFB-Farben gaben mindestens 79% der Probanden den erwarteten Farbnamen an. Die Angaben der Helligkeit weichen teilweise stärker vom erwarteten Ergebnis ab. Für „Dunkelorange“ zeigen die Alternativfarben deutliche Verbesserungen. Statt

nur 28% richtiger Namensangaben bei der ursprünglichen DFB-Farbe, waren es für das Orange bei 30° 91%. Die Freitextfragen zeigen allerdings, dass es eher für Normalorange eingesetzt werden sollte. Sonst zeigten die Alternativfarben keine deutlichen Verbesserungen.

Farbe	Freitextangabe									Entscheidung	
	Be-fragte	Farbe richtig		Alles richtig		nur Hel-lichkeit angegeben	andere relevante Antwort			gut	nicht gut
		origi-nal	ange-passt	origi-nal	ange-passt		Name	abso-lut	re-lativ		
rfn	118	113	113	85	85	4	Hell-Rot	23	19%	73%	6%
rfh	76	31	69	26	64	1				55%	22%
rfd	73	62	66	56	60	0				74%	4%
gefn	69	65	65	34	34	4	Hell-Gelb	29	42%	66%	12%
gefhn	66	60	64	57	61	1				76%	3%
gefd	118	78	81	22	22	8	Norm-Gelb	26	22%	37%	41%
blfn	69	67	67	59	60	2				76%	2%
blfh	72	66	69	64	67	1				78%	1%
blfd	70	61	63	53	55	1	Norm-Blau	8	11%	76%	2%
ofn	119	97	97	57	57	5	Hell-Orange	34	29%	75%	3%
ofhn	114	49	51	43	45	4	Hell-Gelb	24	21%	52%	27%
ofd	115	32	32	15	15	2	Hell-Braun	28	24%	43%	36%
							Dunkel-Braun	24	21%		
vfn	116	105	105	78	78	4	Hell-Violett	19	16%	75%	4%
vfnh	117	93	96	76	77	4	Norm-Violett	18	15%	76%	3%
vfd	116	100	100	87	87	3	Norm-Violett	13	11%	74%	3%
grfn	117	114	114	92	92	2	Hell-Grün	18	15%	76%	2%
grfnh	62	55	55	52	52	1				77%	1%
grfd	70	66	68	59	61	0	Norm-Grün	7	10%	77%	1%
brfn	65	60	60	39	39	1	Dunkel-Braun	20	31%	67%	12%
brfnh	68	43	49	42	48	3				66%	13%
brfd	73	47	47	43	44	1	Schwarz	19	26%	59%	19%
gufn	67	61	61	41	41	4	Hell-Grau	18	27%	67%	12%
gufnh	73	63	65	46	49	2	Norm-Grau	16	22%	76%	1%
gufd	76	68	70	42	43	2	Norm-Grau	25	33%	74%	5%
v270n	117	72	72	48	48	3	Norm-Blau	31	26%	63%	13%
v270h	105	90	94	72	74	2	Norm-Violett	20	19%	76%	3%
v270d	108	57	57	51	51	1	Dunkel-Blau	47	44%	64%	15%
o30d	116	106	106	23	23	4	Norm-Orange	70	60%	38%	41%
o25d	118	103	103	20	20	2	Norm-Orange	69	58%	46%	33%
ge_rote_r_n	113	107	108	65	66	4	Hell-Gelb	27	24%	73%	5%
r_dunkler_n	118	113	114	86	87	3	Hell-Rot	14	12%	75%	4%

Tabelle 14 Übersicht zu den Ergebnissen der Online-Umfrage zu DFB-Farben

Abkürzungen: r=rot, ge=gelb, bl=blau, o=orange, v=violett, gr=grün, br=braun, gu=grau, f=final (Farbe aus Tabelle 10), n=normal (mittel), h=hell, d=dunkel, Norm=Normal (Mittel)

Werte bei „angepasst“ wurden nach Zusammenfassung bedeutungsgleicher Antworten ermittelt

Anhang A Fragebogen zur visuellen Darstellung der DFB-Farben

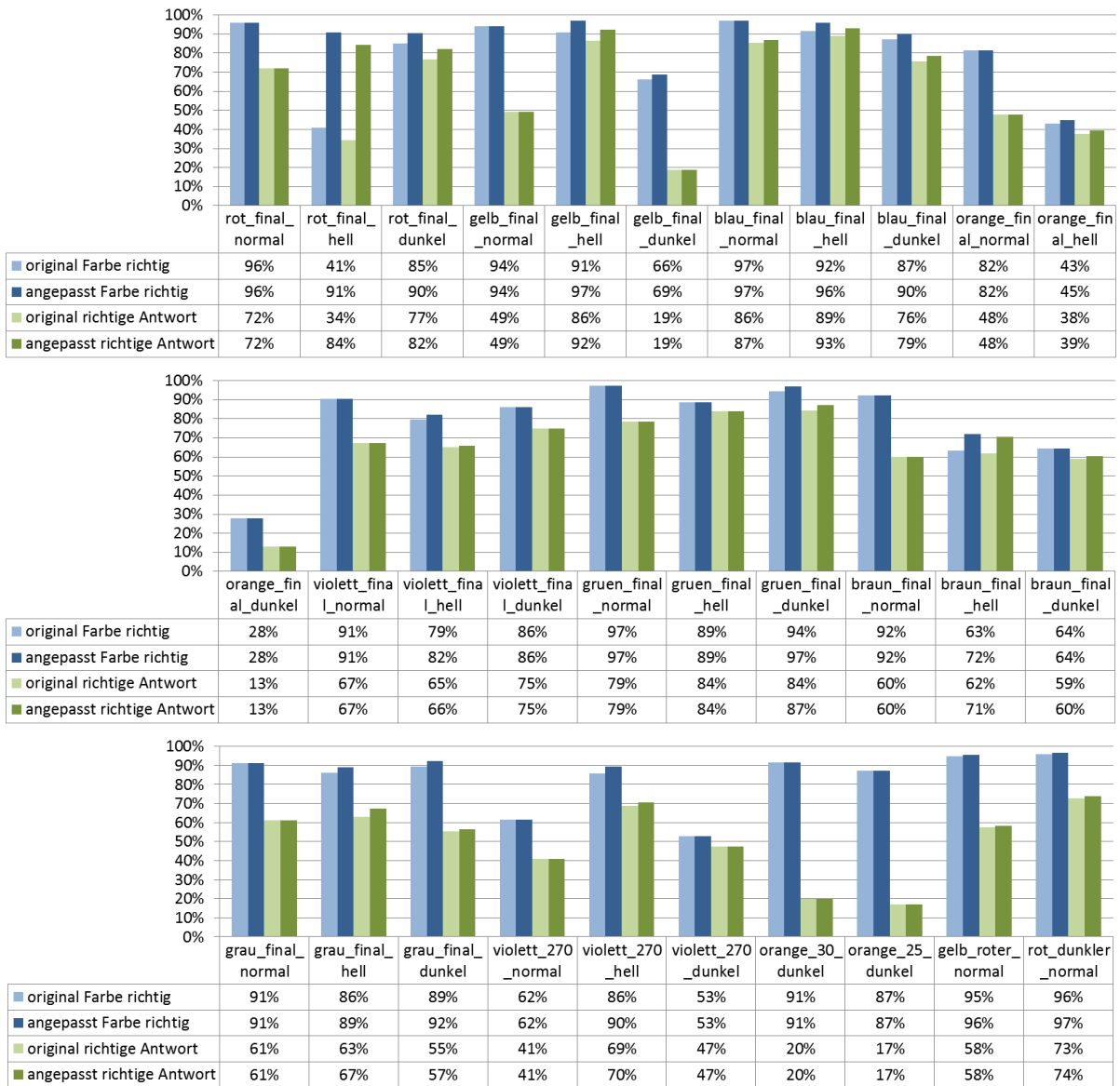


Abbildung 125 Auswertung der Eingabefragen

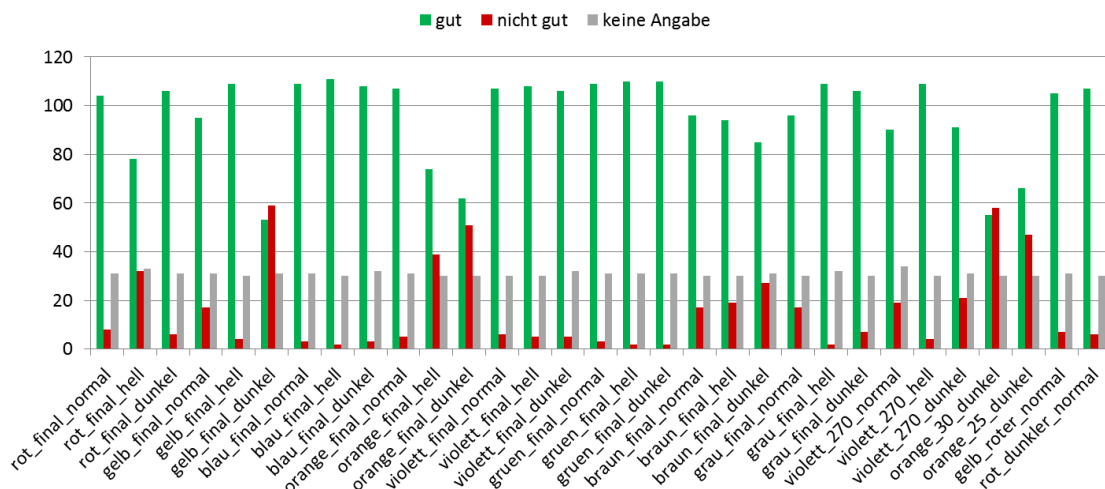


Abbildung 126 Auswertung der Auswahlfragen

Anhang B Ein Lernprogramm für Brailleschrift

Die Erstellung textbasierter Dokumente ist ein wesentlicher Teil der Arbeit am Computer. Damit auch Blinde dies in fehlerfreier Rechtschreibung meistern können, benötigen sie Zugang zu Dokumenten in schriftlicher Form, also in Brailleschrift oder zumindest taktiler Schwarzschrift. Dabei ist die Brailleschrift bei den heute verbreiteten Hilfsmitteln natürlich die bevorzugte Darstellungsform. Sie ist heutzutage eines der wichtigsten Kommunikationsmittel für Blinde und wird deshalb bereits in der Grundschule unterrichtet. Obwohl der Unterricht in großen Teilen bereits am Computer stattfindet, um die Schüler von Anfang an mit dem Computer und seinen Möglichkeiten, sowie der Benutzung einer Braillezeile vertraut zu machen, existiert bisher kein computerunterstütztes Lernprogramm, wie es für Fremdsprachen oder Mathematik heutzutage üblich ist. Dies ist insbesondere für Späterblindete problematisch. Zwar erhalten auch diese Unterricht in Brailleschrift. Allerdings kann er meist nicht so intensiv sein wie in der Grundschule. Auch sind die Lernfortschritte aufgrund unterschiedlichen Alters oder anderer Gegebenheiten häufig noch unterschiedlicher als bei Kindern. Ein Lernprogramm kann hier helfen, individueller zu Lernen.

Aus diesem Grund wurde im Rahmen dieser Arbeit ein Lernprogramm für Brailleschrift entwickelt [TM10]. Das Programm, namens BSLern, ist eine in C++ geschriebene Windowsanwendung und erzeugt Sprach- und Braillezeilen-Ausgabe mit Hilfe von JAWS [Sci10], dem derzeit am häufigsten genutzten Screenreader. Dabei ist BSLern nicht fest an JAWS gebunden. Es wurden lediglich passende JAWS-Skripte geschrieben. Die Unterstützung anderer Screenreader ist damit also möglich, sofern diese ähnliche Möglichkeiten zur Anpassung der Sprach- und Braillezeilen-Ausgabe bieten. Somit kann BSLern auch in Zukunft verbreitet eingesetzt werden. Das Programm kann kostenlos von der Webseite <http://www.vis.uni-stuttgart.de/~taras/BSLern.html> heruntergeladen werden. Dort steht eine Setupdatei zur Verfügung, die das Programm mit den nötigen JAWS-Skripten und einem Vollschriftkurs für Brailleanfänger installiert. Außerdem finden sich dort eine ausführliche Dokumentation zum Programm und seinen Konfigurationsmöglichkeiten.

Die Entwicklung wurde getrennt in ein Rahmenprogramm, das den Ablauf, die Steuerungsmöglichkeiten und die Hilfestellungen definiert, sowie textuelle Dateien, in denen Lernkurse mit ihren Lektionen gestaltet werden können. Auf diese Weise können leicht verschiedene Lernabfolgen und verschiedene Lehrtexte unterstützt werden. Die Kursdateien sind auch für Laien leicht änderbar. So können Brailleschriftlehrer selbst spezielle Kurse entsprechend den Bedürfnissen ihrer Schüler oder neuesten didaktischen Erkenntnissen erstellen. Dazu sind neben den Lektionsinhalten auch der Zeichensatz und die Vollschrift-Kürzungen konfigurierbar. So können Kurse für verschiedene Sprachen und Schrifttypen erstellt werden.

Die Idee, ein Lernprogramm für Punktschrift zu entwickeln, ist nicht neu. Auch an der Universität Stuttgart wurden unter Leitung von Frau Dr. Waltraud Schweikhardt schon verschiedene Ansätze untersucht und erfolgreich eingesetzt (beispielsweise [CS80]). Aber leider kam

es bisher nicht zu einer weiteren Verbreitung. Dies lag vor allem an den damals verfügbaren Technologien, die eine schnelle Verbreitung und Vervielfältigung der Materialien nicht zuließen. Mit den heutigen Technologien ist eine Verbreitung leicht und eine Vervielfältigung von Übungsmaterialien nicht mehr notwendig. Außerdem ist eine Änderung der Lehrinhalte wesentlich einfacher geworden. Da sich allerdings die Konzepte der damaligen Programme als sinnvoll erwiesen haben, wurden sie für BSLern wieder aufgegriffen. Deshalb arbeitet auch BSLern mit Lernkursen, die aus Lektionen bestehen. Und jede Lektion enthält die Abschnitte „Einführung“, „Leseübung“ und „Schreibübung“. In der Einführung werden die Inhalte der Lektion mit Sprach- und Braillezeilen-Ausgabe vorgestellt. Die Übungen enthalten Buchstaben- und Wortkombinationen. In der Leseübung erscheinen diese nacheinander auf der Braillezeile und sollen ertastet werden. Danach kann das Gelesene auf der Schwarzschriftastatur eingegeben oder per Sprachausgabe angehört werden. In der Schreibübung werden die Übungsbeispiele diktiert und sollen über die Punktschriftastatur oder ersatzweise mit den Tasten in der Grundreihe der Schwarzschriftastatur in Punktschrift eingegeben werden, wobei je eine Taste für einen Punkt eines Braillezeichens steht²¹. Bei fehlerhaften Eingaben gibt BSLern Hilfestellungen. Diese erfolgen in mehreren Stufen, angefangen vom einfachen Hinweis, dass die Eingabe nicht richtig war, bis hin zur Nennung der Punkte des Braillezeichens, das als nächstes eingegeben werden muss. Weiterhin bietet BSLern eine allgemeine Hilfe zur Programmbedienung und Hilfetexte zu den einzelnen Lektionen, die in den Lektionsdateien angegeben werden können. Außerdem können Punktschriftmuster zu Schwarzschriftzeichen und umgekehrt erfragt werden.

BSLern ist so gestaltet, dass auch Sehende den Programmablauf verfolgen können, um z. B. gemeinsames Lernen von Eltern mit blinden Kindern zu ermöglichen. Dazu kann die Braillezeilen-Ausgabe in Punkt- und Schwarzschrift auf dem Bildschirm angezeigt werden (siehe Abbildung 127).

²¹ Zur Eingabe von Braillezeichen müssen somit bis zu acht Tasten auf einmal gedrückt werden. Bei vielen Tastaturen wird dies allerdings nicht korrekt erkannt. Deshalb wurde im Programm ein Modus eingebaut, in dem die einzelnen Tasten für die Braillezeichen-Punkte nacheinander gedrückt werden können. Das Zeichen wird durch Drücken der Leertaste abgeschlossen.

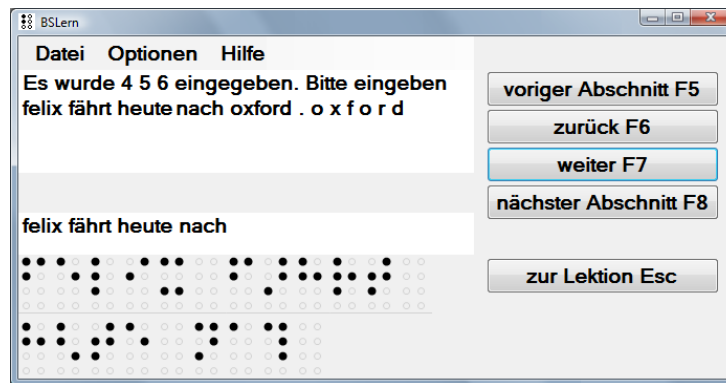


Abbildung 127 Screenshot von BSLern

Die Abbildung zeigt das Programm nach einer Falscheingabe bei dem Übungssatz „felix fährt heute nach oxford“. Zu sehen ist im oberen Bereich der Text, der über Sprache ausgegeben wird, und darunter der bereits eingegebene Text in Schwarzschrift und visueller Brailleschrift. Rechts befinden sich Schaltflächen für die wichtigsten Programmbefehle. Alle anderen Befehle sind über das Menü erreichbar. Außerdem wurden allen Befehlen Tastenkürzel zugewiesen.

BSLern wurde bereits erfolgreich in der Nikolauspflge in Stuttgart eingesetzt und wird auch weiterhin genutzt. Dabei wurden neue Vollschriftkurse für Achtpunkt-Leser für Kinder und Erwachsene entwickelt. Das Programm wird standardmäßig mit einem Vollschriftkurs für Braille-Anfänger ausgeliefert, der sich an älteren Lehrbüchern orientierte.

Anhang C Sonstige Verzeichnisse

C.1 Abkürzungsverzeichnis

2D	2-dimensional
ARGB	Alpha (Transparenz), Rot , Grün , Blau
ARIA	Accessible Rich Internet Applications
ASCII	American Standard Code for Information Interchange
ATAG	Authoring Tool Accessibility Guidelines
ATK	GNOME Accessibility Toolkit
BGG	Behindertengleichstellungsgesetz
BHO	Browser Helper Objects
BITV	Barrierefreie Informationstechnik-Verordnung
CAD	Computer Aided Design
CIE	Commission Internationale de l'Éclairage
CSS	Cascading Style Sheets
DFB	detailliertes Figur-Braille
DOI	Digital Object Identifier
DOM	Document Object Model
EFB	einfaches Figur-Braille
GIF	Graphics Interchange Format
GUI	Graphical User Interface
HSV	Hue, Saturation, Value
HTML	Hypertext Markup Language
ID	Identifikator
IPTC	International Press Telecommunications Council
JPEG	Joint Photographic Experts Group
Lab	L-Achse, a-Achse, b-Achse
MathML	Mathematical Markup Language
MSAA	Microsoft Active Accessibility
OCR	Optical Character Recognition
OFB	Objekt-Figur-Braille
OSM	Off-Screen Modell
PDF	Portable Document Format
PNG	Portable Network Graphics
RGB	Rot, Grün, Blau
SVG	Scalable Vector Graphics
UAAG	User Agent Accessibility Guidelines
UI	User Interface
UIA	UI Automation
UML	Unified Modeling Language
URL	Uniform Resource Locator
VIS	Institut für Visualisierung und Interaktive Systeme
W3C	World Wide Web Consortium
WAI	Web Accessibility Initiative
WCAG	Web Content Accessibility Guidelines

XAML	Extensible Application Markup Language
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XUL	XML User Interface Language

C.2 Abbildungsverzeichnis

Abb. 1	Hilfsmittel für blinde und sehbehinderte Computernutzer	17
Abb. 2	Vorlesungsplan in Tabellenform und in linearisierter Textausgabe	19
Abb. 3	Beispiele für taktile Grafiken	21
Abb. 4	Haptische Ein-Ausgabe-Geräte.....	22
Abb. 5	Beispiele erwerbbarer grafisch-taktile Displays	23
Abb. 6	Ergebnisse aktueller Forschung zu neuen Technologien für taktile Displays	24
Abb. 7	Verschiedene Darstellungsmodi bei Bildschirmvergrößerungssoftware.....	25
Abb. 8	Beispiele für Vergrößerung	27
Abb. 9	Varianten der anwendungsbezogenen Vergrößerung.....	28
Abb. 10	Taxonomie der Vergrößerungstechniken	28
Abb. 11	Darstellung einer Bildschirmvergrößerung auf 300%.....	29
Abb. 12	Beispiel zu Problemen bei Umstellung des Farbschemas	30
Abb. 13	Wikipedia-Seite im Bearbeiten-Modus.....	31
Abb. 14	Beispiele für grafische Bedienoberflächen ohne Anwendungsdokumente	32
Abb. 15	Beispiel einer komplexen Visualisierung auf Grundlage einer Datendatei	33
Abb. 16	Darstellung einer SVG-Datei im Mozilla Firefox und Internet Explorer.....	34
Abb. 17	Darstellung des Ribbons von Microsoft Word in unterschiedlichen Größen.....	35
Abb. 18	Formatierung eines Dokuments mit unterschiedlichen Formatvorlagen	36
Abb. 19	Darstellung der HTML-Datei aus Listing 2 mit CSS-Formatierungen und ohne.....	38
Abb. 20	Dialoge zur Eingabe des Vor- und Nachnamens in XAML und XUL	39
Abb. 21	UIAutomation-Baum des XAML-Dialogs aus Abbildung 20	41
Abb. 22	Beispiele für Webseiten mit den Zielgruppen Blinde und Sehbehinderte	48
Abb. 23	Foto des BrailleDis 9000.....	53
Abb. 24	Anzahl der darstellbaren Zeichen bei Schriftgröße 60 pt	54
Abb. 25	Vergrößerung mit anwendungsbezogenem Zoom in Word auf 500%.....	56
Abb. 26	Vergrößerung eines rechteckigen Bereiches mit der Bildschirmlupe	56
Abb. 27	Vergrößerung eines Dialoges mit einer Bildschirmlupe auf 200% bzw. 300%.....	57
Abb. 28	Auswirkungen einer Halbierung der Auflösung auf die Darstellung	58
Abb. 29	Darstellung der Auswirkung sinnvoller Platzausnutzung bei der Vergrößerung...	59
Abb. 30	Darstellung der Auswirkung von Vergrößerung auf die Menüdarstellung	59
Abb. 31	Verschiedene taktile Schriften.....	60
Abb. 32	Auswirkung verschiedener Schriftarten und Schriftattribute auf die Lesbarkeit..	60
Abb. 33	Ribbon von Word in Originalfarben und kontrastreicher Darstellung	61
Abb. 34	Entwurf zu Markierung von Bedienelementtypen	63
Abb. 35	Beispiele zur Vergrößerung und Markierung	63

Anhang C Sonstige Verzeichnisse

Abb. 36	Verschiedene Darstellungen einer Sinus-Kurve.....	64
Abb. 37	Qualitative Unterschiede bei der Vergrößerung von Schrift.....	65
Abb. 38	Nicht sichtbare Meldung bei Vergrößerung auf ein Vielfaches.....	66
Abb. 39	Beispiel zur Reduktion von Darstellungsdetails in taktilen Grafiken.....	66
Abb. 40	Beispiel für Übersichtsdarstellung des Desktops für Sehbehinderte	67
Abb. 41	Übersichts-Modus bei Bildschirmvergrößerungsprogrammen	68
Abb. 42	Entwurf zur Beachtung von Arbeitsbereichen an Hand eines E-Mail-Fensters	70
Abb. 43	Beispiel zu Hooked Areas aus Bildschirmvergrößerungssoftware von Dolphin	71
Abb. 44	Ablauf der Anpassung im taktilen Firefox.....	74
Abb. 45	Beispiel zur layoutgerechten Darstellung	77
Abb. 46	Beispiel für den Nutzen der Entfernung von Leerräumen in Rastergrafiken	78
Abb. 47	Beispiel zur verkürzten Darstellung von Grafiken	80
Abb. 48	Auswirkung von Textverkürzungen im Menü des Mozilla Firefox.....	81
Abb. 49	Darstellung des grafischen Zooms am Beispiel eines Klassendiagramms.....	83
Abb. 50	Darstellung des nicht-geometrischen, semantischen Zooms.....	83
Abb. 51	Darstellung des hierarchischen Zooms in einem Klassendiagramm	84
Abb. 52	Beispiel zur Kompassnavigation in einer vergrößerten Darstellung	87
Abb. 53	Beispiel-Organigramm mit schrittweisem Aufbau von links	89
Abb. 54	Schematische Darstellungen der Farbräume HSV und $L^*a^*b^*$	92
Abb. 55	Grafiken zum Metriktest.....	95
Abb. 56	Ausschnitte aus den Segmentierungsergebnissen	95
Abb. 57	Beispielgrafiken aus dem E-Learning-Kurs.....	97
Abb. 58	Beispiel zu selektiver Farbdarstellung durch Filterung.....	98
Abb. 59	Beispiele zu gleichzeitiger, unterscheidbarer Darstellung von Farben	99
Abb. 60	Beispiele zu taktilen Texturen und Darstellungen mit taktilem Farbcode	101
Abb. 61	Beispiele zur Untersuchung von blinkenden Darstellungen.....	101
Abb. 62	Testbild zur Untersuchung der Abbildung von Helligkeitsstufen auf Blinken	102
Abb. 63	Ergebnis der Farbfilterung bei herkömmlicher Skalierung.....	104
Abb. 64	Sinus-Cosinus-Bild mit veränderten Farben	105
Abb. 65	Sinus-Cosinus-Bild segmentiert und skaliert	106
Abb. 66	Ablauf des angepassten Skalierungsalgorithmus	107
Abb. 67	Ergebnis der Skalierung und Filterung von Abbildung 47.....	107
Abb. 68	Überbewertung von Farben durch den Skalierungsalgorithmus	107
Abb. 69	Nutzen des Skalierungsalgorithmus bei vergrößerter Bildschirmdarstellung.....	108
Abb. 70	Aufteilung der Ausgabefläche des BrailleDis 9000 in verschiedene Regionen ...	110
Abb. 71	Beispiel-Screenshot zu Illustration der Ansichtsarten	111
Abb. 72	Beispieldarstellungen zur Überblicksansicht	111
Abb. 73	Beispieldarstellungen zur Originalansicht	112
Abb. 74	Beispieldarstellungen zur Arbeitsansicht.....	112
Abb. 75	Symbolansicht des Dialogs mit der Speichern-Abfrage.....	113
Abb. 76	Darstellungen für Sehbehinderte abgeleitet aus dem HyperBraille-Konzept.....	114
Abb. 77	Entwürfe zu vergrößerten Darstellungen auf Basis von HyperBraille	115

Abb. 78	Vergleich von partieller Vergrößerung und Vergrößerung auf ein Vielfaches	117
Abb. 79	Beispiele zur partiellen Vergrößerung in der Originaldarstellung	118
Abb. 80	Vorbereitung und Vergrößerung der Menüleiste.....	119
Abb. 81	Darstellung der Berührungssignale in einer Ausgabe für Sehende	120
Abb. 82	Entwurf eines Widgets zur Anzeige rückgewandelter Braille-Zeichen.....	122
Abb. 83	Darstellung des MIMIZU von Kobayashi und Watanabe	125
Abb. 84	Beispiel für die Umsetzung einer Grafik für Blinde als ASCII-Grafik	125
Abb. 85	Ein Gemälde des von Geburt an blinden Malers Eşref Armağan	126
Abb. 86	Ansätze zur Standardisierung der Farbdarstellung für Blinde.....	127
Abb. 87	Beispiel eines Widget-Entwurfs aus dem Projekt HyperBraille	131
Abb. 88	Nummerierung der Punkte eines Braillezeichens sowie	133
Abb. 89	Beispiel zu DFB-Code ohne Orientierungspunkt und mit	133
Abb. 90	Darstellung des Prozesses zur Definition der Farbwerte für die DFB-Farben	134
Abb. 91	Ansichten des HBGraphicsExchange.....	139
Abb. 92	Screenshots von HBGraphicsExchange Version 2 und 3	140
Abb. 93	Beispiel zur Ergänzung eines EFB-Zeichens	141
Abb. 94	Grafik in 9-facher Vergrößerung mit verschiedenen Algorithmen.....	142
Abb. 95	Beispieldarstellungen zu OFB und SVG.....	143
Abb. 96	Grobansicht des Darstellungsprozesses im HyperReader	146
Abb. 97	Dreistufiger Rendering-Prozess für den HyperReader	147
Abb. 98	Darstellungsprozess und Komponenten des grafisch-taktilen Renderers	148
Abb. 99	Diagramm der Klassen und Interfaces des grafisch-taktilen Renderers.....	151
Abb. 100	Grafisch-taktiler Ausgabe einer Tabelle in linearisierter Tabellendarstellung	152
Abb. 101	Grafisch-taktiler Ausgabe einer Tabelle in tabellarischer Tabellendarstellung...	153
Abb. 102	Screenshot von Solitär als Beispiel für die Wirkungsmöglichkeiten von Add-Ins	157
Abb. 103	Ausschnitte aus der taktilen Ausgabe für Solitär ohne Anwendung des Add-Ins	158
Abb. 104	Ausschnitt aus der taktilen Ausgabe für Solitär mit Anwendung des Add-Ins	159
Abb. 105	CommandButton-Widget mit Markierung der einzelnen Bestandteile	163
Abb. 106	Darstellung der Randinformationen zu Schriftzeichen.....	167
Abb. 107	Darstellung einer Ausgabe des Renderers im BrailleMap-Format	168
Abb. 108	Darstellung der Renderer-Ausgabe in der Ansicht für Sehende.....	168
Abb. 109	Ablauf der benutzergesteuerten Vergrößerung	175
Abb. 110	Screenshot des Firefox-Prototypen zur partiellen Vergrößerung mit Lupe	176
Abb. 111	Ein per Drag&Drop erstellter Beispieldialog mit Ausgabe von UISpy	179
Abb. 112	Beispiele für (schlecht zugängliche) SVG aus dem Web	181
Abb. 113	Grafische Darstellung der Dia-Datei aus Listing 18.....	183
Abb. 114	Abbildung einer Liste aus einer Word-Datei.....	184
Abb. 115	Beispiel zur Ablage und Auswahl alternativer Darstellungen zu Grafiken	188
Abb. 116	Abbildung einer Liste aus einer Word-Datei.....	191
Abb. 117	Beispiele für Dialoge	195
Abb. 118	Beispiel zur Verwendung von Labels zur Anzeige von Eigenschaftswerten	197
Abb. 119	Beispiel zu besser nutzbaren Zugänglichkeitsattributen (PowerPoint).....	201

Abb. 120	SVG-Darstellung zur Visualisierung der Signalausbreitung in einer Zelle.....	204
Abb. 121	Screenshot der durch Listing 22 erzeugten Webseite während der Exploration	206
Abb. 122	Zwei weitere Anwendungsbeispiele des JavaScript-SVG-Frameworks	207
Abb. 123	Gehäuse des neuen Displays von Metec im Vergleich zum BrailleDis 9000	214
Abb. 124	Darstellung der Fragen zur Evaluierung der DFB-Farben	221
Abb. 125	Auswertung der Eingabefragen	224
Abb. 126	Auswertung der Auswahlfragen	224
Abb. 127	Screenshot von BSLern	227

C.3 Tabellenverzeichnis

Tab. 1	Übersicht zu semantischen Informationen	44
Tab. 2	Unterschied zwischen Farbzuordnung und Abstandsbetrachtung	93
Tab. 3	Segmentierungsfarben zum Test verschiedener Metriken	95
Tab. 4	Zeitspannen, die zur Segmentierung der Testgrafik benötigt wurden.....	96
Tab. 5	Kenndaten des Testdatensatzes für den CIEDE2000-Performanz-Test.....	97
Tab. 6	Messergebnisse zum CIEDE2000-Performanz-Test.....	97
Tab. 7	Übersicht zur EFB-Codierung	129
Tab. 8	Grafiken mit EFB-Darstellung in Schwarzschrift und Brailleschrift.....	130
Tab. 9	Übersicht über die DFB-Codierung mit Angabe der Grundfarben in RGB.....	134
Tab. 10	Darstellung der DFB-Farben mit Webfarbe, RGB- und HSV-Werten.....	136
Tab. 11	Übersicht über die BrailleMap-Codierung	169
Tab. 12	Legende zur BrailleMap-Codierung in Tabelle 11.....	169
Tab. 13	Übersicht der alternativen Farben in der Online-Umfrage zu DFB-Farben.....	219
Tab. 14	Übersicht zu den Ergebnissen der Online-Umfrage zu DFB-Farben.....	223

C.4 Verzeichnis der Listings

List. 1	SVG-Quellcode zu Abbildung 16	34
List. 2	Quellcode einer HTML-Datei mit CSS und eine CSS-Datei.....	37
List. 3	Quellcode des Dialogs zur Eingabe von Vor- und Nachname in XAML	39
List. 4	Quellcode des Dialogs zur Eingabe von Vor- und Nachname in XUL	40
List. 5	SVG-Quellcode zu Abbildung 16 mit Gruppierungen und Metadaten	46
List. 6	Ausschnitt aus einem Modell zur absoluten Farbzuordnung.....	93
List. 7	Beispiel zur Arbeitsansicht einer Webseite	113
List. 8	Beispiel für OFB-Dateien.....	137
List. 9	SVG-Darstellung zur OFB-Datei aus Listing 8.....	144
List. 10	Ausschnitte aus Blindenschrift-Definitionsdateien	150
List. 11	Ausschnitt aus der Behandlungsroutine des Rendering-Events.....	155
List. 12	Ausschnitt aus der Ausschnitt aus einem Rendering-Add-In für Solitär.....	156
List. 13	Beispiel einer Konfigurationsdatei für die Strukturleiste	160
List. 14	Beispiel zur Änderung eines Widgets mit dem Konfigurations-Modul	161

List. 15	Beispiel zur Änderung des Rendering-Prozesses	162
List. 16	C#-Methoden zum Kopieren von Farbdaten zwischen Bitmap und int[]	171
List. 17	Quellcode für einen Linksklick durch ein Fenster hindurch.....	173
List. 18	Auszug aus einer Dia-Datei	183
List. 19	Auszug aus der durch Dia erzeugten SVG-Datei zu Listing 18	184
List. 20	Auszug aus der durch Word erzeugten HTML-Datei zu Abbildung 100	185
List. 21	Beispiel zur Angabe von Alternativdarstellungen über das HTML-Object	193
List. 22	Quellcode einer Webseite zur Exploration einer SVG-Grafik	205

Literaturverzeichnis

- [ABT10] ABTIM BLINDENHILFSMITTEL: *VideoTIM - Der Blindenmonitor*. Webseite. http://www.abtim.com/home_d/_videotim/videotim.html. Version:2010. – zuletzt besucht: 12.02.2010
- [Adoa] ADOBE: *Adobe Flash Professional CS5*. Webseite. <http://www.adobe.com/products/flash/>. – zuletzt besucht: 29.06.2010
- [Adob] ADOBE: *Adobe SVG Viewer download area*. Webseite. <http://www.adobe.com/svg-viewer/install/>. – zuletzt besucht: 29.06.2010
- [Ai 09] AI SQUARED: *ZoomText*. Webseite. <http://www.aisquared.com/zoomtext>. Version:2009. – zuletzt besucht: 26.03.2010
- [Alb06] ALBERT, Peter: *Math Class: An Application for Dynamic Tactile Graphics*. In: MIESENBERGER, Klaus (Hrsg.) ; KLAUS, Joachim (Hrsg.) ; ZAGLER, Wolfgang L. (Hrsg.) ; KARSHMER, Arthur I. (Hrsg.): *ICCHP Bd. 4061*, Springer, 2006 (Lecture Notes in Computer Science). – ISBN 3–540–36020–4, S. 1118–1121
- [Alb08] ALBERT, Kristin: *Konzeption und Entwicklung eines taktilen User Interfaces*, Institut für angewandte Informatik, TU Dresden, Großer Beleg, 2008
- [Ame10] AMERICAN PRINTING HOUSE FOR THE BLIND: *APH Educational Research - Guidelines for Design of Tactile Graphics*. Webseite. <http://www.aph.org/edresearch/guides.htm>. Version:2010. – zuletzt besucht: 15.02.2010
- [App] APPLE INC.: *Accessibility Programming Guide for iPhone OS: Making Your iPhone Application Accessible*. http://developer.apple.com/iphone/library/documentation/UserExperience/Conceptual/iPhoneAccessibility/Making_Application_Accessible/Making_Application_Accessible.html
- [Arm10] ARMAGAN, Esref: *Paintings*. Webseite. <http://www.armagan.com/paintings.asp>. Version:2010. – zuletzt besucht: 17.02.2010
- [AS01] ALDRICH, Frances K. ; SHEPPARD, Linda: *Tactile graphics in school education: perspectives from pupils*. In: *British Journal of Visual Impairment* 19 (2001), Nr. 2, 69-73. <http://dx.doi.org/10.1177/026461960101900204>. – DOI 10.1177/026461960101900204
- [AW08] ALTMANNINGER, Kerstin ; WÖß, Wolfram: *Accessible Graphics in Web Applications: Dynamic Generation, Analysis and Verification*. In: [MKZK08], S. 378–385
- [Bal09] BALZER, Michael: *Preis-Kaleidoskop*. interaktives SVG. <http://www.destatis.de/Voronoi/PreisKaleidoskop.svg>. Version:2009. – zuletzt besucht: 27.08.2010
- [Bar10] BARKER, Sally: *The Barker Code of Color/Fabric Representation*. Webseite. <http://www.tactilecolor.com/>. Version:2010. – zuletzt besucht: 17.02.2010

- [Bau10] BAUDISCH, Patrick: *patrick baudisch's projects* . Webseite. <http://patrickbaudisch.com/projects/>. Version:2010. – zuletzt besucht: 09.09.2010
- [bgg] *Gesetz zur Gleichstellung behinderter Menschen (Behindertengleichstellungsgesetz - BGG)*. Elektronisch verfügbar. <http://www.gesetze-im-internet.de/bgg/>. – Vollzitat: Behindertengleichstellungsgesetz vom 27. April 2002 (BGBl. I S. 1467, 1468), das zuletzt durch Artikel 12 des Gesetzes vom 19. Dezember 2007 (BGBl. I S. 3024) geändert worden ist
- [bit] *Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie Informationstechnik-Verordnung - BITV)*. Elektronisch verfügbar. <http://www.gesetze-im-internet.de/bitv/>. – Vollzitat: Barrierefreie Informationstechnik-Verordnung vom 17. Juli 2002 (BGBl. I S. 2654)
- [BK99] BERLIN, Brent ; KAY, Paul: *Basic Color Terms: Their Universality and Evolution*. Center for the Study of Language and Inf, 1999 <http://www.worldcat.org/isbn/1575861623>. – ISBN 1575861623
- [BL96] BECKMANN, Paul J. ; LEGGE, Gordon E.: Psychophysics of Reading–XIV. The Page Navigation Problem in Using Magnifiers. In: *Vision Research* 36 (1996), Nr. 22, 3723 - 3733. [http://dx.doi.org/DOI: 10.1016/0042-6989\(96\)00084-3](http://dx.doi.org/DOI: 10.1016/0042-6989(96)00084-3). – DOI DOI: 10.1016/0042-6989(96)00084-3. – ISSN 0042-6989
- [Bok08] BOKTOR, Andrew: *Keyboard Supported Diagram Editing and Navigating*, Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart, Bachelorarbeit, 2008. – Nr. 2180
- [Bre10] BREIDER, Jaap: *TactileView - sehen und fühlen, fühlen und sehen*. <http://www.tactileview.com/>. Version:2010. – zuletzt besucht: 17.02.2010
- [bsb] *Blinden- und Sehbehindertenbund Pfalz e. V.* Webseite. <http://www.bsb-pfalz.de/>. – zuletzt besucht 2010
- [cam10] *CampusSource*. Webseite. <http://www.campussource.de/>. Version:2010. – zuletzt besucht: 08.09.2010
- [Con92] CONWAY, Damian: An experimental comparison of three natural language colour naming models. In: *Proc. East-West International Conference on Human-Computer Interactions*, 1992, S. 328–339
- [Cre10] CREATIVE HEROES: *AudioGames, your resource for audiogames, games for the blind, games for the visually impaired!* Webseite. <http://www.audiogames.net/>. Version:2010. – zuletzt besucht 09.03.2010
- [CS80] CSIMA, Feodora ; SCHWEIKHARDT, Waltraud: Eine rechnerunterstützte Lehr- und Übungsstrategie für die Blindenschrift. In: *Angewandte Informatik* 22 (1980), Nr. 11, S. 462–468

- [Der05] DER BUNDESWAHLLLEITER: *Online-Wahlatlas zur Bundestagswahl 2005*. Webseite. http://www.bundeswahlleiter.de/de/bundestagswahlen/BTW_BUND_05/onlineatlas/. Version:2005. – zuletzt besucht: 27.08.2010
- [Dol10] DOLPHIN: *Dolphin Computer Access - Screen Reader and Magnification Software*. Webseite. <http://www.yourdolphin.com/>. Version:2010. – zuletzt besucht: 08.09.2010
- [DPSS10] DANNECKER, Tanja ; PASEDAG, Matthias ; STOLL, Christa ; STURM, Heinrich: *Der Tag wird zur Nacht - Audio-Computerspiel für blinde und sehbehinderte Kinder*. Webseite. <http://www.dertagwirdzurnacht.de/>. Version:2010. – zuletzt besucht: 09.03.2010
- [Ebe10] EBERLE GMBH: *Aktuelles*. Webseite. <http://www.fe-tronic.de/aktuelles.php>. Version:2010. – zuletzt besucht: 21.04.2010
- [Fak] FAKOÓ, Alexander: *Fakoo, die alternative Punktschrift für Blinde und Sehende - Fakoo-Alphabet*. Webseite. <http://www.fakoo.de/fakoo.html>. – zuletzt besucht: 01.09.2010
- [FD07] FREDJ, Z. B. ; DUCE, D. A.: GraSSML: Accessible Smart Schematic Diagrams for All. In: *Universal Access in the Information Society* 6 (2007), Nr. 3, S. 233–247. <http://dx.doi.org/http://dx.doi.org/10.1007/s10209-007-0085-9>. – DOI <http://dx.doi.org/10.1007/s10209-007-0085-9>. – ISSN 1615–5289
- [Feh85] FEHRLE, Thomas: *Ein rechnerunterstützter Zeichenplatz für Blinde*, Institut für Informatik, Universität Stuttgart, Diplomarbeit, 1985
- [FFO⁺08] FUJIYOSHI, Mamoru ; FUJIYOSHI, Akio ; OHTAKE, Nobuyuki ; YAMAGUCHI, Katsuhito ; TESHIMA, Yoshinori: The Development of a Universal Design Tactile Graphics Production System BLOT2. In: [MKZK08], S. 938–945
- [flu10] FLUSOFT SPEZIAL COMPUTER TECHNIK: *Braillezeilen*. Webseite. <http://www.flusoft.de/-produkte/brz/index.html>. Version:2010. – zuletzt besucht: 12.02.2010
- [Gao08] GAO, Meng: *Untersuchung möglicher Schnittstellen zur Erhebung von Informationen für das HyperBraille-Off-Screen-Model aus XUL-basierten Anwendungen*, Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart, Diplomarbeit, 2008. – Nr. 2706
- [GGRT08] GROTTTEL, Sebastian ; GUNZENHÄUSER, Rul ; ROTARD, Martin ; TARAS, Christiane: Lernen mit Web-basierten interaktiven Systemen. In: *Navigationen - Zeitschrift für Medien- und Kulturwissenschaften* 8 (2008), Nr. 1, S. 43–58. – ISSN 1619–1641
- [gno] GNOME Documentation Library - *How Accessibility Works in GNOME*. Webseite. <http://library.gnome.org/devel/accessibility-devel-guide/nightly/gad-how-it-works.html.en>. – zuletzt besucht: 28.06.2010
- [Goo] GOOGLE: *Google Docs - Online documents, spreadsheets, presentations, surveys, file storage and more*. Webseite. <http://docs.google.com>. – zuletzt besucht: 29.06.2010

Literaturverzeichnis

- [Gra10] GRACZYK, William: *The Blind Readers' Page*. Webseite. <http://blindreaders.info/>.
Version:2010. – zuletzt besucht: 09.03.2010
- [GS90] GUNZENHÄUSER, Rul ; SCHWEIKHARDT, Waltraud: Tactile Representation of Scanned Documents. In: FELLBAUM, K. (Hrsg.): *Access to Visual Computer Information by Blind Persons, State of the Art and Proposals for Projects, Concerted Action of Technology and Blindness*, 1990, S. 87–95
- [GTW09] GUNZENHÄUSER, Rul ; TARAS, Christiane ; WÖRNER, Michael: Professor Volker Claus: vom o. Professor zum e-Professor. In: VOLKER DIEKERT AND KARSTEN WEICKER AND NICOLE WEICKER (Hrsg.): *Informatik als Dialog zwischen Theorie und Anwendung - Festschrift für Volker Claus zum 65. Geburtstag*. Vieweg+Teubner, 2009. – ISBN 9783834808240, S. 83–92
- [GW04] GAJOS, Krzysztof ; WELD, Daniel S.: SUPPLE: automatically generating user interfaces. In: *IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces*. New York, NY, USA : ACM, 2004. – ISBN 1–58113–815–6, 93–100
- [Haa10] HAAG, Florian: *Entwicklung eines flexiblen Widget-Sets für die Stuttgarter Stiftplatte*, Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart, Diplomarbeit, 2010. – Nr. 2893
- [Hah04] HAHN, Volker F.: Ikonisch-mediale Unterstützung Blinder beim Mathematik lernen - der neue GEOMETRIE-ATLAS. In: *blind-sehbehindert* 3 (2004), 188–203. <http://www.grenzenlos-erfurt.de/>
- [Han10] HANDY TECH ELEKTRONIK GMBH: *Graphic Window Professional (GWP)*. Webseite. <https://www.handytech.de/de/normal/produkte/fuer-blinde/gwp/index.html>.
Version:2010. – zuletzt besucht: 12.02.2010
- [Has10] HASTY, Lucia: *TactileGraphics.org: Design Principles for Tactile Graphics*. Webseite. <http://www.tactilegraphics.org/designingtgs.html>. Version:2010. – zuletzt besucht: 15.02.2010
- [Hel01] HELIOS, Dietmar ; UNIVERSITÄT KARLSRUHE, STUDIENZENTRUM FÜR SEHGESCHÄDIGTE (Hrsg.): *Handbuch zur Erstellung taktiler Grafiken*. 2001 <http://www.szs.uni-karlsruhe.de/download/grafik.pdf>. – Online verfügbar
- [Höh99] HÖHRMANN, Björn: *Warum ALT Attribute Pflicht sind*. Webseite. <http://www.bjoernsworld.de/html/alt-text.html>. Version:1999. – zuletzt besucht: 25.08.2010
- [Hig10] HIGH TECH CENTER TRAINING UNIT (HTCTU): *Photoshop Tactile Graphic Instructions*. Webseite. http://www.htctu.net/divisions/altmedia/FAQs/-Photoshop_Tactile_Instructions.htm. Version:2010. – zuletzt besucht: 17.02.2010
- [hyp10] *HyperBraille*. Webseite. <http://www.hyperbraille.de/>. Version:2010. – zuletzt besucht: 09.09.2010

- [IBM] IBM: *Human Ability and Accessibility Center - Developer guidelines*. Webseite. <http://www-03.ibm.com/able/guidelines/index.html>. – zuletzt besucht: 29.06.2010
- [IKT⁺08] ITOU, Kazuyuki ; KATO, Baku ; TANIGUCHI, Masaru ; OTOGAWA, Toshio ; ITOH, Kazuyuki ; KIYOTA, Kimiyasu ; EZAKI, Nobuo ; UCHIMURA, Keiichi: Learning Support System Based on Note-Taking Method for People with Acquired Visual Disabilities. In: [MKZK08], S. 813–820
- [INC08] INCOBS - INFORMATIONSPPOOL COMPUTERHILFSMITTEL FÜR BLINDE UND SEHBEHINDERTE: *Vergrößerungssoftware - Marktübersicht*. Webseite. <http://www.incobs.de/produktinfos/-grossbild/einzelprodukte.php>. Version:2008. – zuletzt besucht: 26.03.2010
- [Ins10] INSTITUT DER DEUTSCHEN WIRTSCHAFT KÖLN E.V.: *REHADAT - Informationssystem zur beruflichen Rehabilitation*. Webseite. <http://www.rehadat.de/>. Version:2010. – zuletzt besucht: 21.04.2010
- [Int09] INTERNATIONAL PRESS TELECOMMUNICATIONS COUNCIL: *IPTC Standard Photo Metadata*. [http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata\(200907\)_1.pdf](http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata(200907)_1.pdf). Version:07 2009
- [jQu] JQUERY PROJECT TEAM: *jQuery: The Write Less, Do More, JavaScript Library*. Webseite. <http://jquery.com/>. – zuletzt besucht: 28.08.2010
- [Kah98] KAHLISCH, Thomas: *Software-ergonomische Aspekte der Studierumgebung blinder Menschen*. ISBN 3-86064-797-0, Technische Universität Dresden, Dissertation, 1998. <http://www.kahlisch.de/phd/>
- [Kat10] KATHOLISCHES BLINDEN- UND SEHBEHINDERTENWERK NORDDEUTSCHLAND E.V.: *Monitor für Blinde*. http://www.kbwn.de/html/monitor_fur_blinde.html. Version:2010. – zuletzt besucht: 21.04.2010
- [KGS10] KGS CORPORATION: *KGS – Dot View DV-1*. Webseite. http://www.kgs-jpn.co.jp/-b_dv.html. Version:2010. – zuletzt besucht: 12.02.2010
- [Kit10] KITCHEN'S INC: <http://www.kitchensinc.net/>. Webseite. <http://www.kitchensinc.net/>. Version:2010. – zuletzt besucht: 09.03.2010
- [Klöß87] KLÖPFER, Klaus: *Ein multifunktionaler Büroarbeitsplatz für Blinde*, Universität Stuttgart, Dissertation, 1987
- [KMB⁺03] KU, Jeonghun ; MRAZ, Richard ; BAKER, Nicole ; ZAKZANIS, Konstantine K. ; LEE, Jang H. ; KIM, In Y. ; KIM, Sun I. ; GRAHAM, Simon J.: A Data Glove with Tactile Feedback for fMRI of Virtual Reality Experiments. In: *CyberPsychology & Behavior* 6 (2003), Nr. 5, S. 497–508

- [KMZB04] KLAUS, Joachim (Hrsg.) ; MIESENBERGER, Klaus (Hrsg.) ; ZAGLER, Wolfgang L. (Hrsg.) ; BURGER, Dominique (Hrsg.): *Computers Helping People with Special Needs, 9th International Conference, ICCHP 2004, Paris, France, July 7-9, 2004, Proceedings. Bd. 3118*. Springer, 2004 (Lecture Notes in Computer Science). – ISBN 3–540–22334–7
- [KNS08] KANAHORI, Toshihiro ; NAKA, Masayuki ; SUZUKI, Masakazu: Braille-Embedded Tactile Graphics Editor with Infty System. In: [MKZK08], S. 919–925
- [Krü07] KRÜGER, Maria: *Barrierefreie Gestaltung für Blinde im E-Lernen am Beispiel einer Flash-basierten Anwendung*, Fachhochschule für Technik und Wirtschaft Berlin, Fachbereich Wirtschaftswissenschaften II, Diplomarbeit, 2007. <http://www.f4.htw-berlin.de/~weberwu/papers/diplom-Maria-Krueger.pdf>
- [Krö08] KRÖKER, Andreas: *Entwurf eines graphisch-taktilen Renderers für HyperBraille*, Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart, Diplomarbeit, 2008. – Nr. 2730
- [KST⁺07] KATO, Yusaku ; SEKITANI, Tsuyoshi ; TAKAMIYA, Makoto ; DOI, Masao ; ASAKA, Kinji ; SAKURAI, Takayasu ; SOMEYA, Takao: Sheet-Type Braille Displays by Integrating Organic Field-Effect Transistors and Polymeric Actuators. In: *Electron Devices, IEEE Transactions on* 54 (2007), feb., Nr. 2, S. 202–209. <http://dx.doi.org/10.1109/TED.2006.888678>. – DOI 10.1109/TED.2006.888678. – ISSN 0018–9383
- [Kur96] KURZE, Martin: TDraw: A Computer-based Tactile Drawing Tool for Blind People. In: *Assets '96: Proceedings of the Second Annual ACM Conference on Assistive Technologies*. New York, NY, USA : ACM, 1996. – ISBN 0–89791–776–6, S. 131–138
- [KVW08] KRAUS, Michael ; VÖLKEL, Thorsten ; WEBER, Gerhard: An Off-Screen Model for Tactile Graphical User Interfaces. In: [MKZK08], S. 865–872
- [KW04] KOBAYASHI, Makoto ; WATANABE, Tetsuya: Communication System for the Blind Using Tactile Displays and Ultrasonic Pens - MIMIZU. In: [KMZB04], S. 731–738
- [LB05] LAM, Heidi ; BAUDISCH, Patrick: Summary thumbnails: readable overviews for small screen web browsers. In: *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2005. – ISBN 1–58113–998–5, 681–690
- [Lei10] LEINER, Dominik: *oFb - der onlineFragebogen 2.0 – Fragebogen online erstellen, Befragung im Internet durchführen*. Webseite. <https://www.soscisurvey.de/>. Version:2010. – zuletzt besucht: 24.02.2010
- [Lin03] LINDBLOOM, Bruce: *Delta E (CIE 1994)*. Webseite. http://www.brucelindbloom.com/index.html?Egn_DeltaE_CIE94.html. Version:2003. – zuletzt besucht: 08.09.2010

- [Luc09] LUCAS, Jonathan: *Jonathan Lucas - Long Beach, California - SIAFU*. Webseite. http://www.coroflot.com/public/-individual_file.asp?portfolio_id=974269&individual_id=190761. Version:2009. – zuletzt besucht: 17.02.2010
- [mat] *Mathematik-Online*. Webseite. <http://mo.mathematik.uni-stuttgart.de/>. – zuletzt besucht: 27.05.2010
- [met08] *metacoon*. Webseite. <http://www.metacoon.net/>. Version:2008. – zuletzt besucht: 08.09.2010
- [Mey10] MEYER, Eric: *S5 – Ein simples, standardbasiertes Slideshowsystem*. Webseite. <http://yatil.de/s5/>. Version:2010. – zuletzt besucht: 08.09.2010
- [Mica] MICROSOFT: *IAccessible Interface*. Webseite. <http://msdn.microsoft.com/en-us/library/accessibility.iaccessible.aspx>. – zuletzt besucht: 28.06.2010
- [Micb] MICROSOFT: *Magnification API*. Webseite. <http://msdn.microsoft.com/en-us/library/ms692162.aspx>. – zuletzt besucht: 08.09.2010
- [Micc] MICROSOFT: *Microsoft Active Accessibility*. Webseite. <http://msdn.microsoft.com/en-us/library/ms971350.aspx>. – zuletzt besucht: 28.06.2010
- [Midd] MICROSOFT: *SetSysColors Function (Windows)*. Webseite. <http://msdn.microsoft.com/en-us/library/ms724940.aspx>. – zuletzt besucht: 29.06.2010
- [Mice] MICROSOFT: *UI Automation Specification*. Webseite. <http://msdn.microsoft.com/en-us/library/dd561923.aspx>. – zuletzt besucht: 28.06.2010
- [Micf] MICROSOFT: *XAML Overview (WPF)*. Webseite. <http://msdn.microsoft.com/en-us/library/ms752059.aspx>. – zuletzt besucht: 29.06.2010
- [MKZK08] MIESENBERGER, Klaus (Hrsg.) ; KLAUS, Joachim (Hrsg.) ; ZAGLER, Wolfgang L. (Hrsg.) ; KARSHMER, Arthur I. (Hrsg.): *Computers Helping People with Special Needs, 11th International Conference, ICCHP 2008, Linz, Austria, July 9-11, 2008. Proceedings. Bd. 5105*. Springer, 2008 (Lecture Notes in Computer Science). – ISBN 978-3-540-70539-0
- [MKZK10] MIESENBERGER, Klaus (Hrsg.) ; KLAUS, Joachim (Hrsg.) ; ZAGLER, Wolfgang L. (Hrsg.) ; KARSHMER, Arthur I. (Hrsg.): *Computers Helping People with Special Needs, 12th International Conference, ICCHP 2010, Vienna, Austria, July 14-16, 2010, Proceedings, Part II. Bd. 6180*. Springer, 2010 (Lecture Notes in Computer Science). – ISBN 978-3-642-14099-0
- [MLWS09] MATYSEK, M. ; LOTZ, P. ; WINTERSTEIN, T. ; SCHLAAK, H.F.: *Dielectric Elastomer Actuators for Tactile Displays*. In: *EuroHaptics conference, 2009 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2009. Third Joint*, 2009, S. 290–295

Literaturverzeichnis

- [Moza] MOZILLA DEVELOPER NETWORK (MDN): *Extensions - Mozilla Developer Center (MDC)*. Webseite. <https://developer.mozilla.org/en/extensions>. – zuletzt besucht: 29.06.2010
- [Mozb] MOZILLA DEVELOPER NETWORK (MDN): *SVG in Firefox - Mozilla Developer Center (MDC)*. Webseite. https://developer.mozilla.org/en/svg_in_firefox. – zuletzt besucht: 29.06.2010
- [Mozc] MOZILLA DEVELOPER NETWORK (MDN): *XUL - Mozilla Developer Center (MDC)*. Webseite. <https://developer.mozilla.org/en/xul>. – zuletzt besucht: 29.06.2010
- [Nat10] NATURAL RESOURCES CANADA - EARTH SCIENCES SECTOR: *GeoGratis - Tactile Maps of Canada*. Webseite. <http://geogratis.gc.ca/geogratis/en/option/select.do?id=36865>. Version:2010. – zuletzt besucht: 16.02.2010
- [NIS09] NIST - NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY: *NIST 'Pins' Down Imaging System for the Blind*. Webseite. http://www.nist.gov/public_affairs/factsheet/-visualdisplay.htm. Version:2009. – zuletzt besucht: 17.02.2010
- [Nov10] NOVINT TECHNOLOGIES, INC.: *Get Wowed! with the Novint Falcon*. Webseite. http://home.novint.com/products/novint_falcon.php. Version:2010. – zuletzt besucht: 15.02.2010
- [oox08] ; ECMA International (Veranst.): *Standard ECMA-376 - Office Open XML File Formats*. <http://www.ecma-international.org/publications/standards/Ecma-376.htm>. Version:2008
- [Pet09] PETERS, Wadim: *Umsetzung einer Fokus-und-Kontext-Technik zur Vergrößerung von graphischen Benutzungsoberflächen als Firefox-Extension*, Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart, Studienarbeit, 2009. – Nr. 2189
- [PF93] PERLIN, Ken ; FOX, David: *Pad: an alternative approach to the computer interface*. In: *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM, 1993. – ISBN 0-89791-601-8, S. 57-64
- [PNW10] PRESCHER, Denise ; NADIG, Oliver ; WEBER, Gerhard: *Reading Braille and Tactile Ink-Print on a Planar Tactile Display*. In: [MKZK10], S. 482-489
- [Pre05] PRESTELE, Stefan L.: *Rapid E-Learning - Aus PowerPoint-Folien werden interaktive Lernprogramme*. PDF. <http://www.competence-site.de/e-learning/Aus-PowerPoint-Folien-werden-interaktive-Lernprogramme>. Version:2005. – zuletzt besucht: 08.09.2010
- [Pre09] PRESCHER, Denise: *Ein taktiles Fenstersystem mit Multitouch-Bedienung*, Institut für angewandte Informatik, TU Dresden, Diplomarbeit, 2009
- [Pro09] PROVINCIAL RESOURCE CENTRE FOR THE VISUALLY IMPAIRED (PRCVI): *Tactile Graphics with Corel Draw*. Webseite. <http://www.prcvi.org/tactilegraphics/>. Version:2009. – zuletzt besucht: 17.02.2010

- [Rai10] RAISSAKIS, Petra: *Eine Homepage über den Alltag blinder und sehbehinderter Menschen*. Webseite. <http://www.anderssehen.at/>. Version:2010. – zuletzt besucht: 09.03.2010
- [RGE07] ROTARD, Martin ; GIERETH, Mark ; ERTL, Thomas: Semantic lenses: Seamless augmentation of web pages with context information from implicit queries. In: *Computers & Graphics* 31 (2007), Nr. 3, S. 361–369
- [RKE05] ROTARD, Martin ; KNÖDLER, Sven ; ERTL, Thomas: A tactile web browser for the visually disabled. In: *HYPertext '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*. New York, NY, USA : ACM, 2005. – ISBN 1–59593–168–6, S. 15–22
- [ROE04] ROTARD, Martin ; OTTE, Kerstin ; ERTL, Thomas: Exploring Scalable Vector Graphics for Visually Impaired Users. In: [KMZB04], S. 725–730
- [Rot05] ROTARD, Martin: *Standardisierte Auszeichnungssprachen der Computergraphik für interaktive Systeme*, Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart, Dissertation, 2005. <http://elib.uni-stuttgart.de/opus/volltexte/2005/2389/>
- [RP09] RICHTER, Andreas ; PASCHEW, Georgi: Optoelectrothermic Control of Highly Integrated Polymer-Based MEMS Applied in an Artificial Skin. In: *Advanced Materials* 21 (2009), Nr. 9, 979–983. <http://dx.doi.org/10.1002/adma.200802737>. – DOI 10.1002/adma.200802737
- [RTE08] ROTARD, Martin ; TARAS, Christiane ; ERTL, Thomas: Tactile web browsing for blind people. In: *Multimedia Tools Appl.* 37 (2008), Nr. 1, S. 53–69
- [SA01] SHEPPARD, Linda ; ALDRICH, Frances K.: Tactile graphics in school education: perspectives from teachers. In: *British Journal of Visual Impairment* 19 (2001), Nr. 3, 93-97. <http://dx.doi.org/10.1177/026461960101900303>. – DOI 10.1177/026461960101900303
- [Sch02] SCHUFFELEN, Marco: *On Editing Graphics For the Blind*. Online-Buch. <http://homepage.mac.com/schuffelen/grbl/grbl0.html>. Version:2002. – zuletzt besucht: 17.02.2010
- [Sci10] SCIENTIFIC, Freedom: *JAWS for Windows Screen Reading Software*. Webseite. <http://www.freedomscientific.com/products/fs/jaws-product-page.asp>. Version:2010. – zuletzt besucht: 08.09.2010
- [sec] *Section 508: The Road to Accessibility*. Webseite. <http://www.section508.gov/>. – zuletzt besucht: 29.06.2010
- [Sen10] SENSABLE TECHNOLOGIES, INC.: *PHANTOM Omni® Haptic Device*. Webseite. <http://www.sensable.com/haptic-phantom-omni.htm>. Version:2010

- [SKNW09] SCHIEWE, Maria ; KÖHLMANN, Wiebke ; NADIG, Oliver ; WEBER, Gerhard: What You Feel Is What You Get: Mapping GUIs on Planar Tactile Displays. In: STEPHANIDIS, Constantine (Hrsg.): *HCI (6)* Bd. 5615, Springer, 2009 (Lecture Notes in Computer Science). – ISBN 978-3-642-02709-3, S. 564–573
- [SKW10] SPINDLER, Martin ; KRAUS, Michael ; WEBER, Gerhard: A Graphical Tactile Screen-Explorer. In: [MKZK10], S. 474–481
- [Smi78] SMITH, Alvy R.: Color gamut transform pairs. In: *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM, 1978, S. 12–19
- [Spi07] SPINTRE, Horst: *Taktiles Webbrowser für blinde Menschen auf Basis von Firefox*, Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart, Diplomarbeit, 2007. – Nr. 2540
- [SRK⁺10] SCHLEGEL, Thomas ; RASCHKE, Michael ; KNITTIG, Markus ; DRIDIGER, Alexander ; WOKUSCH, Stefan ; TARAS, Christiane: Evaluation of Current User Interface Generator Frameworks for Graphical Interactive Systems. In: *Proceedings of the IADIS International Conferences Interfaces and Human Computer Interaction 2010 and Game and Entertainment Technologies 2010*, 2010, S. 385–390
- [SW03] SPRINGSGUTH, Christian ; WEBER, Gerhard: Design Issues of Relief Maps for Haptic Displays. In: STEPHANIDIS, Constantine (Hrsg.): *Universal Access in HCI : inclusive design in the information society, Proceedings of HCI International 2003, Heraklion, Crete, Greece* Bd. 4, Lawrence Erlbaum Associates, Mahwah, New Jersey, Juni 2003, 1477–1481
- [SWD05] SHARMA, Gaurav ; WU, Wencheng ; DALAL, Edul N.: The CIEDE2000 Color-Difference Formula: Implementation Notes, Supplementary Test Data, and Mathematical Observations. In: *Color Research & Application* 30 (2005), S. 21–30. <http://dx.doi.org/10.1002/col.20070>. – DOI 10.1002/col.20070
- [Tac10] TACTILE COLOUR COMMUNICATION SOCIETY: *Colours by Touch*. <http://www.tactile.org/-TactileColourinformation.html>. Version:2010. – zuletzt besucht: 29.03.2010
- [TE08] TARAS, Christiane ; ERTL, Thomas: Fokus-und-Kontext-Techniken zur intelligenten Vergrößerung von graphischen Benutzungsoberflächen. In: MAIER, Edith (Hrsg.) ; ROUX, Pascale (Hrsg.): *Seniorenerechte Schnittstellen zur Technik - Zusammenfassung der Beiträge zum Usability Day VI (16.05.2008)*, 2008, S. 136–143
- [TE09] TARAS, Christiane ; ERTL, Thomas: Interaction with Colored Graphical Representations on Braille Devices. In: STEPHANIDIS, Constantine (Hrsg.): *Universal Access in Human-Computer Interaction. Addressing Diversity, 5th International Conference, UAHCI 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19-24, 2009. Proceedings, Part I* Bd. 5614, Springer, 2009 (Lecture Notes in Computer Science), S. 164–173

- [Tex08] TEXAS SCHOOL FOR THE BLIND AND VISUALLY IMPAIRED (TSBVI): *Tactile Graphics Resources*. Webseite. <http://www.tsbvi.edu/Education/tactile-graphics.htm>. Version:2008. – zuletzt besucht: 17.02.2010
- [tga] ; Computer Science & Engineering University of Washington (Veranst.): *Tactile Graphics Project - Improving Access to Graphical Images for Blind Students*. <http://tactilegraphics.cs.washington.edu/>
- [The] THE LINUX FOUNDATION: *IAccessible2*. Webseite. <http://www.linuxfoundation.org/collaborate/workgroups/accessibility/iaccessible2>. – zuletzt besucht: 28.06.2010
- [TM10] TARAS, Christiane ; MARQUARDT, Thomas: "BSLern" - Ein Lernprogramm für Punktschrift. In: *horus - Marburger Beiträge zur Integration Blinder und Sehbehinderter 2* (2010). <http://www.dvbs-online.de/horus/2010-2-4671.htm>
- [TRE07] TARAS, Christiane ; ROTARD, Martin ; ERTL, Thomas: An E-Learning Course on Scientific Visualization. In: *EG 2007 - Education Papers*, Eurographics Association, 2007, 17-22
- [TRS⁺10] TARAS, Christiane ; RASCHKE, Michael ; SCHLEGEL, Thomas ; ERTL, Thomas ; PRESCHER, Denise ; WEBER, Gerhard: Improving Screen Magnification Using the HyperBraille Multiview Windowing Technique. In: [MKZK10], S. 506–512
- [TRSE10] TARAS, Christiane ; RASCHKE, Michael ; SCHLEGEL, Thomas ; ERTL, Thomas: Running Graphical Desktop Applications on Tactile Graphics Displays Made Easy. In: PAUL GNANAYUTHAM AND HUGO PAREDES AND IOANNIS T. REKANOS (Hrsg.): *Proceedings of 3rd International Conference on Software Development for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2010), 25-26 November 2010, Oxford, United Kingdom*, UTAD - Universidade de Trás-os-Montes e Alto Douro, 2010. – ISBN 987–972–669–994–1, S. 141–147
- [TSSE09] TARAS, Christiane ; SANTOSO, Michael ; SCHLEGEL, Thomas ; ERTL, Thomas: Ein JavaScript-Framework zur erweiterten Nutzung strukturierter und annotierter SVG-Grafiken in Webseiten. In: PHILIPP VON HELLBERG AND GUIDO KEMPTER (Hrsg.): *Technologie-nutzung ohne Barrieren - Zusammenfassung der Beiträge zum Usability Day VII (06.03.2009)*, 2009, S. 123–130
- [TSW⁺08] TARAS, Christiane ; SIEMONEIT, Oliver ; WEIBER, Nico ; ROTARD, Martin ; ERTL, Thomas: Improving the Accessibility of Wikis. In: [MKZK08], S. 430–437
- [uis] MICROSOFT (Hrsg.): *Windows Presentation Foundation Tools: UI Spy (UISpy.exe)*. <http://msdn.microsoft.com/en-us/library/ms727247.aspx>
- [URBS07] URBAN, Philipp ; ROSEN, Mitchell R. ; BERNS, Roy S. ; SCHLEICHER, Dierk: Embedding non-Euclidean color spaces into Euclidean color spaces with minimal isometric disagreement. In: *J. Opt. Soc. Am. A* 24 (2007), Nr. 6, 1516–1528. <http://josaa.osa.org/abstract.cfm?URI=josaa-24-6-1516>

Literaturverzeichnis

- [Vie08] VIEWPLUS: *IVEO Hands-on-Learning System*. Webseite. <http://www.iveo.org/>. Version:2008. – zuletzt besucht: 15.02.2010
- [Vie09a] VIEWPLUS: *Braille Translation Software Tiger Software Suite (TSS)*. Software. <http://www.viewplus.eu/products/software/braille-translator/>. Version:2009. – zuletzt besucht: 17.02.2010
- [Vie09b] VIEWPLUS: *Emprint SpotDot Color Braille Printer*. Webseite. <http://www.viewplus.eu/products/ink-braille-printers/emprint-spotdot/>. Version:2009. – zuletzt besucht:15.02.2010
- [VvH07] VIDAL-VERDÚ, Fernando ; HAFEZ, Moustapha: Graphical Tactile Displays for Visually-Impaired People. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 15 (2007), März, Nr. 1, S. 119–130. <http://dx.doi.org/10.1109/TNSRE.2007.891375>. – DOI 10.1109/TNSRE.2007.891375. – ISSN 1534–4320
- [VWB08] VÖLKE, Thorsten ; WEBER, Gerhard ; BAUMANN, Ulrich: Tactile Graphics Revised: The Novel BrailleDis 9000 Pin-Matrix Device with Multitouch Input. In: [MKZK08], S. 835–842
- [Weba] WEB ACCESSIBILITY INITIATIVE (WAI): *Authoring Tool Accessibility Guidelines (ATAG) Overview*. Webseite. <http://www.w3.org/WAI/intro/atag.php>. – zuletzt besucht: 29.06.2010
- [Webb] WEB ACCESSIBILITY INITIATIVE (WAI): *User Agent Accessibility Guidelines (UAAG) Overview*. Webseite. <http://www.w3.org/WAI/intro/uaag.php>. – zuletzt besucht: 29.06.2010
- [Webc] WEB ACCESSIBILITY INITIATIVE (WAI): *WAI-ARIA Overview*. Webseite. <http://www.w3.org/WAI/intro/aria.php>. – zuletzt besucht: 21.07.2010
- [Webd] WEB ACCESSIBILITY INITIATIVE (WAI): *Web Accessibility Initiative (WAI)*. Webseite. <http://www.w3.org/WAI/>. – zuletzt besucht: 29.06.2010
- [Webe] WEB ACCESSIBILITY INITIATIVE (WAI): *Web Content Accessibility Guidelines (WCAG) Overview*. Webseite. <http://www.w3.org/WAI/intro/wcag.php>. – zuletzt besucht: 29.06.2010
- [Wik10a] WIKIPEDIA: *Pixel art scaling algorithms*. Webseite. http://en.wikipedia.org/wiki/Pixel_art_scaling_algorithms. Version:2010. – zuletzt besucht: 17.08.2010
- [Wik10b] WIKIPEDIA: *Web colors*. Webseite. http://en.wikipedia.org/wiki/Web_colors. Version:2010. – zuletzt besucht: 19.02.2010
- [Wik10c] WIKIPEDIA: *WikiProjekt SVG*. Webseite. http://de.wikipedia.org/wiki/Wikipedia:WikiProjekt_SVG. Version:2010. – zuletzt besucht: 27.08.2010

- [WKOY06] WATANABE, Tetsuya ; KOBAYASHI, Makoto ; ONO, Shoichiro ; YOKOYAMA, Keiko: Practical use of interactive tactile graphic display system at a school for the blind. In: MENDEZ-VILAS, A. (Hrsg.) ; MARTI, A. S. (Hrsg.) ; GONZALEZ, J. M. (Hrsg.) ; GONZALEZ, J.A. M. (Hrsg.): *Technological Science Education, Collaborative Learning, Knowledge Management* Bd. 2. FORMATEX, 2006. – ISBN 978–84–690–2472–8, S. 1111–1115
- [Woo] WOOD, Keith: *jQuery SVG Demo*. Webseite. <http://keith-wood.name/svg.html>. – zuletzt besucht: 28.08.2010
- [Wora] WORLD WIDE WEB CONSORTIUM (W3C): *Cascading Style Sheets - home page*. Webseite. <http://www.w3.org/Style/CSS/>. – zuletzt besucht: 29.06.2010
- [Worb] WORLD WIDE WEB CONSORTIUM (W3C): *Document Object Model (DOM)*. Webseite. <http://www.w3.org/DOM/>. – zuletzt besucht: 29.06.2010
- [Worc] WORLD WIDE WEB CONSORTIUM (W3C): *HTML*. Webseite. <http://www.w3.org/html/>. – zuletzt besucht: 29.06.2010
- [Word] WORLD WIDE WEB CONSORTIUM (W3C): *HTML 4.01 Specification - 13 Objects, Images, and Applets*. Webseite. <http://www.w3.org/TR/html401/struct/objects.html>. – zuletzt besucht: 26.08.2010
- [Wore] WORLD WIDE WEB CONSORTIUM (W3C): *Portable Network Graphics (PNG) Specification (Second Edition)*. Webseite. <http://www.w3.org/TR/PNG/>. – zuletzt besucht: 29.06.2010
- [Worf] WORLD WIDE WEB CONSORTIUM (W3C): *Scalable Vector Graphics (SVG) - XML Graphics for the Web*. Webseite. <http://www.w3.org/Graphics/SVG/>. – zuletzt besucht: 29.06.2010
- [Worg] WORLD WIDE WEB CONSORTIUM (W3C): *SVG Document Object Model (DOM) – SVG 1.1 (Second Edition)*. Webseite. <http://www.w3.org/TR/SVG/svgdom.html>. – zuletzt besucht: 28.08.2010
- [Worh] WORLD WIDE WEB CONSORTIUM (W3C): *W3C Math Home*. Webseite. <http://www.w3.org/Math/>. – zuletzt besucht: 08.09.2010
- [Wor99] WORLD WIDE WEB CONSORTIUM (W3C): *Forms in HTML documents*. Webseite. <http://www.w3.org/TR/REC-html40/interact/forms.html>. Version:1999. – zuletzt besucht: 08.09.2010
- [Wor10] WORLD WIDE WEB CONSORTIUM (W3C): *HTML5 - A vocabulary and associated APIs for HTML and XHTML - Editor's Draft 8 September 2010*. Webseite. <http://dev.w3.org/html5/spec/Overview.html>. Version:2010. – zuletzt besucht: 08.09.2010
- [YC05] YEOW, John T. ; CHEUNG, Bob: An Integrated Tactile and Visual Display Motion Simulator for Air and Land Application. In: *Mechatronics and Automation, 2005 IEEE International Conference* Bd. 1, 2005, S. 298–302 Vol. 1

Literaturverzeichnis

- [YKB03] YU, Wai ; KANGAS, K. ; BREWSTER, S.: Web-based haptic applications for blind people to create virtual graphs, 2003, S. 318 – 325

Lebenslauf

Persönliche Angaben

Name	Taras
Vorname	Christiane
Geburtsname	Butterich
Geburtsdatum	16.02.1983
Geburtsort	Rudolstadt / Thüringen
Staatsangehörigkeit	deutsch
E-Mail	christiane@familie-taras.de



Schulbildung und Studium

1989 - 1993	Grundschule „Otto Grotewohl“ / „Annegret Kellner“ in Rudolstadt
1993 - 2001	Gymnasium „Friedrich Fröbel“ in Bad Blankenburg Abschluss: Allgemeine Hochschulreife Abschlussnote: 1,2
Okt. 2001 - Mai 2006	Studium der Softwaretechnik an der Universität Stuttgart Abschluss: Diplom-Informatiker (Dipl.-Inf.) Abschlussnote: sehr gut

Berufliche Tätigkeiten

Feb. 2002 - Apr. 2002	Praktikum bei Tenovis in Stuttgart
Jun. 2004 - Okt. 2005	Werkstudentin bei Porsche Engineering Services in Bietigheim-Bissingen
Jun. 2006 - Jan. 2011	Doktorandin / wissenschaftliche Mitarbeiterin am Institut für Visualisierung und Interaktive Systeme der Universität Stuttgart