Universität Stuttgart

Visuallisierung Interaktive Systeme, Abt.
Augemented Reality Virtual Reality

Allmandring 19
70569 Stuttgart

Bachelorarbeit

Entwicklung von Augemented Reality Wordles

Jonas Österlein

Studiengang:	Informatik - B.Sc.
1. Prüfer:	Prof. Dr. M. Sedlmair
2. Prüfer:	
Betreuer:	Katrin Angerbauer, M.Sc., Michael Becher, M.Sc
begonnen am:	October 15, 2021
beendet am:	April 15, 2022

Kurzfassung

Diese Arbeit bringt Wordles vom Bildschirm in die Augmented Reality. Sie vereint dabei viele verschiedene und etablierte Wordle-Algorithmen in einer Anwendung. Als Plattform diente hierfür die Hololens 2. Eine eigenständige, aber auch leistungsschwächere AR-Plattform mit ihren eignen Herausforderungen. Für die Implementierung in Unity mussten viele Abwägungen und Dos and Dont's beachtet werden. Zuletzt wurde eine Pilotstudie durchgeführt, um die Nutzererfahrung mit Wordles in einer Augmented Reality und Gestikmanipulation auszuwerten und zu beurteilen.

Abstract

This work brings Wordles from the screen into augmented reality. It combines many different and established Wordle algorithms in one application. The platform used was the Hololens 2, an independent but also less powerful AR platform with its own challenges. For the implementation in Unity, many trade-offs and dos and don’ts had to be considered. Finally, a pilot study was conducted to evaluate and assess the user experience with Wordles in an augmented reality and gesture manipulation.

Inhaltsverzeichnis

1 Einleitung

Wordles sind Wörter, die lose zusammen ein Bild ergeben. Also Begriffe bzw. “Tags”, die kompakt und semantisch angeordnet sind. Sie erfreuen sich seit der Veröffentlichung auf der Website “wordle.net” einer großen Beliebtheit. Sie gelten als kreatives Tool Schlagwörter zu einem bestimmten Thema visuell darzustellen. Diese Arbeit beschäftigt sich mit dem Ansatz, Wordles von Bildschirmen in die dritte Dimension der Augmented Reality zu bringen. Hierfür bedient sie sich der Hololens2 und Unity als Plattform. Die Gliederung der Arbeit ist wie folgt: In Kapitel 2 werden die Grundlagen zur Arbeit beschrieben. Kapitel 3 fasst verwandte Arbeiten zusammen und liefert teils einen Einblick in andere Wordle-Ansätze, die umgesetzt worden sind. Diese werden dann in Kapitel 4 ausführlich in ihrer mathematischen und algorithmischen Form dargestellt und erklärt. Kapitel 5 befasst sich dann mit der Umsetzung in Unity und beschreibt den grundlegenden Aufbau und Limitierungen und Schwierigkeiten, die bei Einbindungen und Integrationen entstehen können. Das Kapitel 6 handelt von den Herausforderungen, den Abwägungen und den Stolperfallen, die bei einer naiven Implementierung in Unity auftreten. Um die Umsetzung Auswerten zu können, wurde in Kapitel 7 eine Pilotstudie durchgeführt. Zuletzt fasst Kapitel 8 die Arbeit und Ihre Ergebnisse zusammen und liefert einen Ausblick.

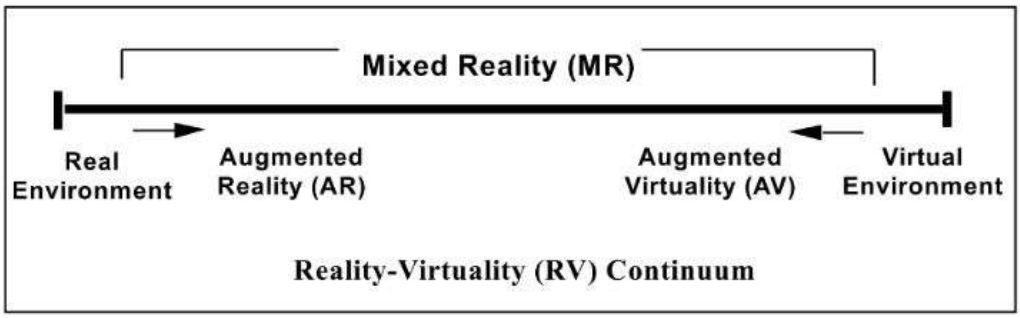
2 Grundlagen

Wordles in AR brige die zwei Bereiche, Augmented Reality und Word-Clouds zusammen. Im folgenden folgt eine kurze Einführung und Erklärung zu Augmented Reality, der verwendeten Plattform Hololens und zu Word-Clouds.

2.1 Augmented Reality

Bei Augmented Reality(AR) geht es um die “augmentierung”, genauer gesagt die Erweiterung der Realität um digitale Inhalte. Diese werden auf unterschiedliche Arten und Weisen dargestellt. Grunlegend zählt AR, nach Milgram, als Bestandteil der übergeordneten Technologieklasse Mixed Reality (MR) [MTUK94]. Mixed Reality beschreibt das ganze Spektrum zwischen Realität und reine virtuelle Umgebung. Das Spektrum kann der Abbildung 2.1 entnommen werden.

Abbildung 2.1: Reality-Virtuality Koninoum nach Milgram [MTUK94]



Am rechten Ende des Spektrum ist die Virtual Reality (VR). Hier immersiert sich der User komplett in eine digitale Welt und besitzt nur noch minimale Interaktionspunkte mit der reellen Umgebung. Links davon befindet sich die Augmented Virtuality (AV). Hier hält sich der Anwender ebenfalls in einer vollkommenen digitalen Umgebung auf. Reelle Objekte werden dabei geschickt integriert und bilden einen festen Bestandteil.

Auf der linken Seite des Spektrums befindet sich die echte Umgebung ohne jeglichen digitalen Zusatz der sich einbettet oder hinzugefügt wird. Im Bereich der Augmented Reality ist die reelle Umgebung weiterhin fester Bestandteil. Diese wird durch digitale Artefakte, auch "Hologramme"genannt erweitert. Der User kann mit diesen interagieren, entweder direkt durch Sensoren oder indirekt durch die Manipulation von reell gegebenen Objekten, die vom System erkannt werden.

Azuma, R. definiert AR in drei Charakteristiken[Azu97]:

1. Sie kombiniert reale und virtuelle Welt
2. es lässt sich mit jener in Echtzeit interagieren
3. und sie findet in 3-D statt.
Diese Definition wird heutzutage als meistverbreitete akzeptiert.

Für die Umsetzung von AR haben sich verschiedene Ansätze etabliert. Sie unterscheiden sich in der Darstellungsmethodik. Hologramme werden entweder durch "Head-Mounted-Displays", Projektoren [Res12] oder durch "Portale"wie Kameras mit Displays erzeugt. Alle drei bieten verschiedene Vorteile und Nachteile und können auch miteinander kombiniert werden, um ein besseres Erlebnis zu erreichen.

2.1.1 Portale in eine Augmented Reality

Bei Portalen verwendet der Nutzer meist ein Handgreifliches Gerät. Dieses ist mit einem Bildschirm und Kamera ausgestattet ist. Die Kamera zeichnet die Umgebung vor dem Nutzer auf und gibt dieses auf dem Bildschirm live wieder. Die Anwendung erweitert dann diese Anzeige durch Hologramme. Der Benutzer schaut dann Sozusagen wie durch ein Portal in eine andere Welt. Eine Welt die der realen entspringt und durch Hololgrame passend “augmentiert” bzw. erweitert wird. Ein bekanntes Beispiel ist das in den letzen Jahren viral gegangen Videospiel "Pokemon Go"[Cha18] oder die Kamera-Filter der Video/SocialMedia-Platformen [Ins17], welche zusätzliche Elemente einblenden oder die Aufzeichnung abändern. Der Vorteil dieser Umsetzung, ist die Portabilität der Geräte. Ein AR fähiges Smartphone benötigt wenig Setup und schränkt den Nutzer nur minimal ein. Darüber hinaus sind Touch-Anwendungen durch den Bildschirm einfach zu Realisieren. Ein Nachteil ist hierbei, dass ständige Halten des Smartphones zwischen sich und der zu betrachtenden Umgebung. Zudem sind die Interaktionsmöglichkeiten sehr einfach gehalten. Interaktionen sind meist nur über den Touchscreen möglich.

2.1.2 Spatial Augemnted Reality/Projektion Mapping

Ein weiterer Ansatz ist es Augmented Reality zu projizieren [AWE17]. Bei Projektion Mapping, früher als Spatial Augmented Reality bekannt, werden Projektoren eingesetzt um Hologramme auf Flächen und Objekte zu Projizieren. Der Vorteil hierbei ist, mit wenig Aufwand eine Darstellung für vielen User gleichzeitig anzuzeigen. Eine solche Umsetzung kommt häufig auf Kosten der Interaktionsmöglichkeit. Jedoch gibt es seit längeren neue Projekte und Ansätze welche Userinteraktion mit in die Darstellung einfließen lassen. Ein Ansatz von Microsoft [Res12]ist der "Wearable Multitouch Projector". Hierbei trägt der User den Projektor auf der Schulter. Dieser ist in der Lage zeitgleich Tiefeninformationen zu verarbeiten und Gesten zu interpretieren. Eine Vielzahl von Ansätzen und Kombinationen mit Projektion Mapping verwendet die Firma "Perch Interactive"[**perch**]. Sie hat sich spezialisiert Produkte um digitale Informationen und Interaktionsmöglichkeiten im Einzelhandel zu erweitern. Wenn ein Kunde ein Produkt berührt oder zur genaueren Betrachtung aufhebt, ändern sich die digitalen Darstellungen in der unmittelbaren Umgebung. Dies geschieht um weitere Details zum Produkt zu liefern und zusätzliche anreizte für den Kauf

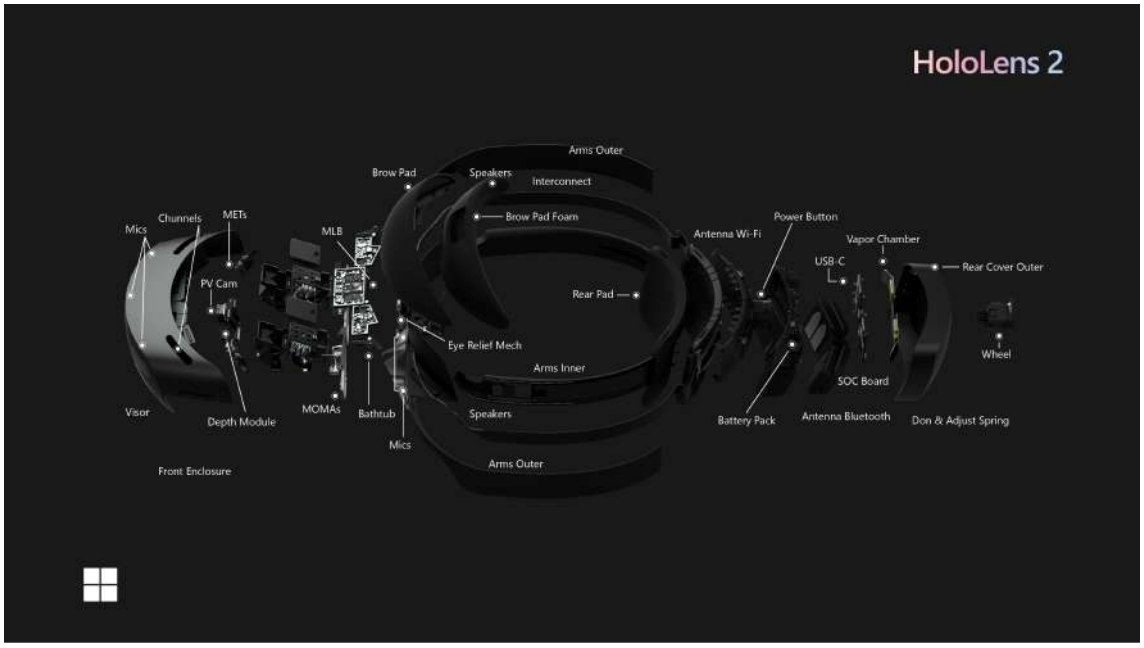
2.2 Head-Mounted-Display

"Head-Mounted-Displays"(HMD) erlauben eine sehr immersive AR-Experience. Hierbei trägt der Anwender ein durchsichtiges Display direkt vor den Augen. Dadurch können Informationen passend im Sichtfeld abgebildet werden. Head-Mounted-Displays sind mit zusätzlichen Sensoren ausgestattet, welche die Umgebung über die Kamera hinaus tracken können. Damit ist es z. B. dem User möglich durch Kopf und Hand-Bewegungen mit den Hologrammen zu interagieren. Konkrete Umsetzungen [RBW05] kommen in der Automobil- und Luftfahrt-Industrie vor. Hierbei findet sie u. a. Anwendung beim Kabinendesign oder liefert Unterstützung bei individuell zugeschnittener Verkabelung von Lastkraftwagen.

2.2.1 Hololens

Eine weit verbreitertes HeadMounted Display ist die Microsoft Hololens[Micd]. Diese wurde erstmals 2016 veröffentlicht und erhielt 2019 ihren Nachfolger, die Hololens2. Die Hololens ist eine Mixed Reality Brille mit durchsichtigen Displays, auf dem den User Informationen dargestellt werden können. Die genauen Spezifikationen werden von Microsoft unter Verschluß gehalten, jedoch ist einiges über sie bekannt [Mica]. Sie hat ein Field of Fiew von 52 Grad und der Bildschirm har etwa 2,5 tausend Radiants mit einer 2k Auflösung in einem 2 zu 3 Verhältnis. Die CPU ist ein Qualcomm SnapDragon 850. Dieser taktet mit 2,0 GHz und im Boost bis zu 2,96GHz, auf bis zu acht Kernen.

Abbildung 2.2: Hololens Breakdown. [Mica]



Damit ist die Hololens leistungsstark genug, um einfache Applikationen komplett eigenständig zu berechnen. Sie kann zudem auf Ihre USB, Bluetooth oder WLAN-Schnittstelle zurückgreifen, um Berechnungen zur Laufzeit auszulagern.

Zur genauen Darstellung von Hologrammen greift die Hololens auf eine Vielzahl von Sensoren zurück. Sie verfolgt dafür unter anderem die Kopfbewegung des Nutzers mit Hilfe Ihrer vier Graustufen-Kameras, einer Tiefenkamera und einem eingebauten Gyroskop, Beschleunigungsmesser und Magnetometer. Mit zwei nach innen gerichteten Infrarotkameras kann die Hololens die Augen des Anwenders tracken. Zu dem besitzt Sie eine 8-Megapixel-Frontkamera, die Video und Bilder mit 1080p und 30 Frames die Sekunde schafft. Das Betriebssystem der Hololens ist das Windows Holographic OS. Dies ist ein auf Windows 10 aufbauendes Betriebssystem mit zusätzlichen Funktionen für Administration und Entwicklung.

Zur Interaktion bietet die Hololens 2 Grundlegende Eingabemöglichkeiten. Einmal besitzt die Hololens ein Mikrofon, mit diesem ist sie in der Lage, Sprachbefehle zu verstehen und umsetzen. Zur wirklichen immersiven Interaktion zählt die Gestenerkennung. Mit dieser ist die Hololens in der Lage, beide Hände gleichzeitig zu erkennen, zu verfolgen und zu interpretieren. Während die Hololens eigentlich in der Lage ist, alle Möglichen Gestiken zu Erkennen wird von Microsoft der “Airtap” empfohlen [Micc]. Einerseits weil die Gestik einfach zu erkennen ist und anderseits eine grundlegende Interaktion zu schaffen, die unabhängig aller Applikationen ist. Das Ziel hierbei ist es, mit einheitlichen Interaktionen den Anwender nicht zu überfordern. Hierbei ballt der User seine Hand zuerst leicht und streckt den Zeigefinger nach oben und den Daumen noch vorne aus. Dann tippt er mit dem Zeigefinger auf den Daumen. Mit dieser Gestik lässt sich vieles auf der Hololens umsetzen. Lässt der User den Zeigefinger gestreckt, so kann er damit etwas selektieren. Tippt er nur kurz mit dem Finger, zählt dies als Auswählen. Tappt der Nutzer und lässt die Hand geschlossen, so wird das als greifen erkannt und Objekte können verschoben werden. Durch das Tracking der gesamten Hand und Kombination mit linker und rechter Hand lassen sich somit eine Vielzahl von Interaktionen umsetzen. Besonders hervorzuheben ist, dass die Hololens in der Lage ist, Tiefeninformationen zur Hand im Zusammenhang von Objekten tracken kann. Dadurch sind Interaktionen in der "näheünd "ferne"möglich. Auch können so Buttons realistisch mit einem Push-Verhalten umgesetzt werden.

2.3 WortWolken

Word-Clouds sind eine beliebte Wahl, um komplexere Themen und längere Texte auf wenige Schlüsselwörter zu reduzieren und darzustellen. Sie helfen dem Betrachter, einen schnellen Überblick über wichtige und sekundäre Aussagen zu einem Thema zu bekommen. Word-Clouds sind dabei in ihrer Bildung sehr uneingeschränkt. So gibt es Tag-Clouds, bei denen primär Funktionalitätim Vordergrund steht und Wordles, bei denen es vorrangig um Aussehen und Ästhetik geht.

2.3.1 TagCloud

Tag-Clouds sind textbasierte Visualisierungen von ausgewählten Tags/Schlagwörtern. Ihr Ziel ist es, komplexere Zusammenhänge und Labels zu einem bestimmten Subjekt einfach lesbar und interpretierbar darzustellen. Dabei wird das darzustellende Thema in Tags aufgebrochen und diese nach ihrer Wichtigkeit sortiert. Diese werden dann visuell verschieden dargestellt. Dabei erhalten häufig die wichtigsten Tags eine besondere Stellung und werden gezielt hervorgehoben. Dies geschieht mit einer größeren Darstellung, farblichen Abgrenzung oder durch eine bestimmte

Positionierung. Die daraus resultierende Abstraktion ermöglicht es Anwendern, eine schnelle und verständliche grobe Übersicht zu erhalten. Über die Jahre wurden mehrere Tag-Cloud-Generatoren entworfen und vorgestellt.

TreeCloud

2010 Stellten Philippe Gambette und Jean Véronis TreeClouds vor [GV09]. Dieser Generator untersucht die Tags nach ihrer semantischen Nähe und platziert sie dementsprechend in einem Baum. Das Resultat ist eine Tag-Cloud in der Tags nicht nur nach ihrer Wichtigkeit und Vorkommen im Text visualisiert werden. Es nutzt den Raum gezielt dazu aus, Wörter in Abhängigkeit zueinander zu stellen. Dadurch können zum Beispiel Reden analysiert werden. Diese enthalten oft mehrere Thematiken, diese Tags alle in einen Topf zu werfen und darzustellen ist folglich nicht zielführend. TreeClouds schafft diesem Problem eine Abhilfe. Tags werden thematisch sortiert und in Zusammenhang gebracht. Damit ist es möglich Relationen von Tags in einem und zwischen Themen darzustellen.

SparkCloud

Eine weitere Tag-Cloud ist die SparkCloud [LRKC10]. Diese fügt zu den einzelnen Tag einen Graphen hinzu, der die Häufigkeit des Tags über die Zeit plottet. Dadurch ist es möglich eine Zeitliche Abhängigkeiten und Trends in der Tag-Cloud darzustellen. Eine mögliche Anwendung ist es politische Reden über die Dauer einer Wahlkampagne zu analysieren. Es lässt sich dadurch zeigen, welche Tags im Laufe der Kampagne an zulauf gewonnen haben und welche an relevant verloren haben.

Wordle

Der neuste Trend in Tag-Clouds sind Wordles [VWF09]. Dies sind Tag-Clouds die den Bereich zwischen den Tags möglichst minimieren und ein kompaktes Bild wiedergeben. Sie erlangten sehr schnell eine große Beliebtheit und sind der Standard für Tag-Clouds geworden. Genaueres ist dem Kapitel 3 zu entnehmen.

3 Related Work

Im Bereich der Wordles wurden in den letzten Jahren mehrere Paper und Arbeiten veröffentlicht, die sich mit Verschiedenen Eigenschaften von Wordles auseinander setzen. Im folgenden werden die in dieser Arbeit relevanten und verwendeten Wordles beschrieben und eingeführt.

3.1 Wordle

Im Juni 2008 stellte Jonathan Feinberg seine Version einer Tag-Cloud online [VWF09]. Diese nannte sich Wordle und wurde in weniger Zeit äußerst beliebt. Innerhalb von neun Monaten wurden über 600.000 Wordles erstellt. Sie wurden somit fast zum neuen Standard für Tag-Clouds. Wordles besitzen einen wesentlichen unterschied zu traditionellen Tag-Clouds. Während Tag-Cloud den Raum zwischen den Tags komplett Vernachlässigen oder gezielt für eine bestimmte Darstellung nutzen, versucht Wordle diesen zu minimieren. Dies sorgt für ein sehr kompaktes Layout, die meist Kreis bis Ei-förmig aussieht. Worte wichtiger Tags werden insbesondere durch ihre Größe hervorgehoben und lassen andere wie eine Füllung um eine Form zu erhalten wirken. Ein Beispiel kann der Abbildung 3.1 entnommen werden.

Abbildung 3.1: Eine Word-Cloud des EdWordle [WCB+18]

Wordles beschreibt sich selbst eher als SSpielzeugünd nicht als seriöses Tool zur Darstellung von Daten. Doch die kreative Freiheit, die es bietet und der Fokus visuell ansprechend zu sein, ist am Ende das, was die meisten Anwender zum Vorzug von Wordles über andere Tag-Clouds brachte. Wordles lassen sich mittlerweile überall finden, vom Klassenraum bis hin zur Zeitung haben diese einfache und vor allem kreative Darstellung von Tag-Clouds ihren Einzug erhalten.

Wordle erzeugt eine Tag-Cloud aus einer archimedischen Spirale heraus. Diese fängt im Zentrum der Cloud an und füllt diese nach und nach mit Tags. Die Wörter werden nach Größe sortiert, und der Spirale entlang verteilt. Dabei versucht der Algorithmus Wörter so nah wie möglich an das Zentrum zu legen. Er schreitet hierfür die Spirale schrittweise ab und überprüft, ob das neu zu platzierende Wort ein bereits platziertes überschneiden würde. Findet keine Überschneidung statt, wird das Word platziert, ansonsten geht der Algorithmus weiter die Spirale entlang. Dadurch Liegen die Tags nahe beieinander und die Tag-Cloud wächst mit jeder neuen Platzierung zu einem

Oval an. Da zum Schluss nur noch die kleinsten Wörter platziert werden, findet eine natürliche Rundung am Cloud-Layout statt. Über die kommenden Jahre wurden viele weiter Versionen von Wordle Generatoren erstellt. Häufig mit verschieden Schwerpunkten und Zielen. Eine Auswahl solcher Generator sind ManiWorlde [KLKS10], ShapeWordle [WCZ+20],EdWordle [WCB+18] und WordlePlus [JLS15].

3.2 ManiWordle

ManiWorlde [KLKS10] erlaubt es den Nutzer, die platzierten Tags zu manipulieren und zu editieren. Es ermöglicht dem Anwender Farbe, Font, Platzierung und Rotation von den platzierten Tags zu verändern. Findet dabei eine Überschneidung von Tags statt, berechnet ManiWorlde eine neue passende Position. Hierbei wird weiterhin der Wordle-Algorithmus mit seiner archimedischen Spirale zur Hilfe genommen. Der Clou ist aber nun, den Spiralenursprung an die Position des neu zu platzierenden Tags zu legen. Dadurch bleibt der Tag lokal möglichst nahe am Ausgangsort und endet nicht am anderen Ende der Cloud. Darüber hinaus, wird die Spiralengröße an die Taggröße angepasst, dies ermöglicht ein schnelleres Finden von passenden Positionen.

3.3 EdWordle

Ein weiterer Generator ist EdWordle [WCB+18]. EdWordle ist eine Wordle mit dem Ziel, Editierung und Kompaktheit zu vereinen. Im Gegensatz zu anderen Wordles, besitzen alle Tags in EdWordle einen *Rigidbody*. Eine Physiksimulation berechnet nach der klassischen Erstellung ein dichteres Layout. Hierbei besitzt das Zentrum der Spirale eine Gravitation, die alle Wörter zu sich zieht. Diese clustern sich nun auf natürliche Weise mit Kollisionen möglichst nah am Gravitationszentrum. Um die Cloud noch kompakter zu gestalten, wirkt nicht nur das Zentrum eine Anziehungskraft auf die Tags, sondern auch die Wörter untereinander. Alle Tags, deren Zentrum direkt ohne Kollision mit anderen verbunden werden kann, ziehen sich zusätzlich an. Dadurch clustern sich die Wörter nebenbei untereinander und sorgen für ein kompakteres Layout. Diese Nachbarschaft-Eigenschaft wird auch für das Editieren von Tags verwendet. Muss ein Wort durch eine Verschiebung neu platziert werden. Berechnet EdWordle nicht den nächstbesten Freien Ort aus, sondern eine Menge an freien Plätzen. Diese werden dann nach dem Erhalt der Nachbarschaft-Eigenschaft sortiert. Dies sorgt dafür, dass die neu platzierten Wörter nicht nur möglichst nah an ihrer Ursprungsposition verbleiben, sondern auch möglichsten denselben Tags wie zuvor anliegen. Darüber hinaus bietet EdWordle eine "Re-Wordle"Funktion an. Diese berechnet für alle weit außen liegende Tags eine neue lokale Position zwischen dem Zentrum und der alten. Dadurch wird das Wordle nach vielen Editierungen wieder kompakter und erhält die relative Position von Tags zum Zentrum der Cloud.

3.4 ShapeWordle

In ShapeWorlde entlegen die Tag-Clouds ihr ovales Aussehen [WCZ+20]. Das Ziel ist es nicht nur eine Word-Cloud nach Wordle zu erstellen, sondern auch dafür zu sorgen, dass die Wordles eine bestimmte Form oder Muster einnehmen. Dafür wird die archimedische Spirale angepasst. Diese

startet nun im Zentrum der Form und bewegt sich entlang der Gradienten des Distanzfeldes zum Rand hin. Sie füllt also systematisch die Form vom Zentrum aus auf. Die Tags werden dann entlang dieser geformten Spirale gelegt. Die Besonderheit ist nun, dass ein neuer Tag nicht nur andere nicht überlappen darf, sonder sich auch in der Form befinden muss. Geht es über den Rand hinaus, wird eine neue Position berechnet. Zu dem werden nun zusätzliche kleine Tags generiert, um die Form vollständig auszufüllen. Das Resultat ist eine Tag-Cloud, die ein klares artistisches Bild darstellt.

3.5 WorldePlus

WordlePlus [JLS15] ist ein Wordle Generator mit Fokus auf Manipulierbarkeit von Wörtern. Es bietet die gleichen Funktionen wie andere Wordles. So lassen sich Tag hinzufügen, bewegen und löschen. Das Besondere hierbei ist aber die Interaktion der Tags untereinander. Schiebt der Anwender mit beiden Händen gleichzeitig zwei Wörter nebeneinander, so werden diese zu einen neuen konkateniert. Auch ist es möglich, Tags mit dem gleichen Inhalt zu Mergen. Wird ein Tag üseräuf ein zweites üser"geschoben, verschmelzen diese in einen üserTag, der nun die Größe von beiden besitzt. Eine weitere Funktion ist es, zwei Tags zueinander auszurichten. Diese werden dann zu einer Gruppe. In dieser teilen sich alle Tags eine Position und eine gemeinsame Rotation. Damit ist das Manipulieren von vielen Wörtern auf einmal möglich.

3.6 DeepClouds

DeepClouds ist ein Wordle Prototyp [JSJ+18], der Wordles von einem zweidimensionalen Raum in einen dreidimensionalen bringt. Diese werden stereoskopisch auf einer Powerwall abgebildet und können durch Gesten manipuliert werden. DeepCloud berechnet für sein Layout konische Spiralen und ordnet Wörter nach ihrer Relevanz absteigend in der z-Achse an. Hierbei untersucht das Paper unterschiedliche Ansätze für die Spiralen, um ein möglichst 2D-Wordle getreues Abbild zu schaffen.

4 Wordles in Augemented Reality

Für die Umsetzung der Wordles in AR wurden eine Vielzahl von Algorithmen anderer Wordles kombiniert und integriert. Das Resultat ist ein Wordle, das nicht nur in AR funktioniert, sondern eine Menge verschiedener Funktionen und Eigenschaften besitzt. Die umgesetzten Algorithmen sind: EdWordle für ein kompaktes Layout, ShapeWorlde für ein Layout, das eine Form beschreibt und ManiWorlde und WordlePlus für eine Vielzahl von Interaktion und Operationen.

4.1 Kompaktes Layouts mit EdWordle

Ein umgesetzter Word-Cloud-Algorithmus ist EdWordle. Dieser versucht mithilfe von Rigidbodies und Nachbarschaftseigenschaften möglichst kompakte Wordclouds zu erzeugen [WCB+18].

Rigidbodies sind eine digitale Version von festen Körpern der realen Welt. Sie stellen ein fixes und unverformbares Objekt da, die einander nicht durchdringen können und auf Kräfte reagieren. Treffen zwei *Rigidbody* aufeinander, so stoßen sich diese ab, bis keine Kollision mehr stattfindet. Damit Rigidbodies realistisch auf Kräfte reagieren können, besitzen diese Maße und Drag. Mit einer Physiksimulation kann berechnen werden, wie sehr ein Objekt auf eine Kraft wie z. B. Gravitation reagiert. Da diese Berechnungen schrittweise geschehen, besitzen alle Rigidbodies einen Geschwindigkeit und Rotationsgeschwindigkeit-Vektor. Dieser enthält die für den Schritt berechneten Bewegungswerte in Relation zur Weltenkoordinate.

Unity unterstützt Rigidbodies von Haus aus und berechnet diese in seiner Physikengine. Damit Sie miteinander interagieren können, benötigt es einen *Collider*. Diese sind virtuelle Räume, die die Grenzen und Kanten eines Körpers definieren. Diese sind einfache Primitive wie Quader, Ellipsoids, Kugeln oder das Mesh eines Objektes. Grundsätzlich gilt, je simpler der *Collidier*, des so schneller die Kollisionsberechnung. Für Wordles eignen sich Quader, in Unity **BoxCollider** genannt, am besten. Diese Umspannen das Wort und sorgen dafür das eine Überlappung mittels Kollisionsberechnung nicht passiert. Um Performance zu bekommen und ungewollte Interaktionen zu verhindern, liegen alle Wordles in einem eigenen Unity **Layer**. In den Physikeinstellungen des Projekts sind alle Kollisionen mit anderen **Layern** ausgeschaltet. Dies ist teils notwendig, da Micsrosfts MRTK gerne mit Kollidieren arbeitet und so viele unerwartet Interaktionen vermieden werden können.

Damit beim editieren von Wordles nicht alle anderen Wörter direkt auf Änderungen reagieren, gibt es zusätzlich einen **MovementLayer**. Wählt der Nutzer ein Wordle aus, so wird es in diesem verschoben. Da der **WordleLayer** und **MovementLayer** nicht miteinander kollideren, lassen sich bequem kollisionsfreie Änderungen vornehmen. Dies erlaubt es Leistung zu sparen und hält andere Wordles an Ort und Stelle. Würde sonst ein Wordle von der Mitte nach außen geschoben, würde dieses alle anderen zur Seite drücken und es müsste mühselig wieder Ordnung geschaffen werden.

EdWordle schlägt zusätzlich vor Wörter ab einer bestimmten Größe mit einen komplexeren *Collider* aus zustatten. Diese besitzen nicht mehr einen **BoxCollider** für das gesamte Wort, sondern mehrere verschiedene pro Buchstabe. Die Kollisionsüberprüfung iteriert dann durch jeden einzelnen *BuchstabenCollider*. Der Vorteil eines komplexeren *Collider* liegt darin, Freiräume die durch vereinzelt hohe Buchstaben wie **t**,**l** oder **i** entstehen, zu minimieren. Dies ist für eine Darstellung auf der Hololens sehr praktisch. Der holographische Darstellungsraum ist in die Höhe mehr beschränkt als in die Breite. Zusätzlich ist es für den Anwender angenehmer, den Kopf seitlich zu bewegen, anstatt auf und ab zu nicken. Besonders mit einer halben Kilo schweren Hololens aufgesetzt. Jedoch stellt sich in der Praxis heraus, das die zusätzliche Berechnungen der *BuchstabenCollider*, wie schon in im Papaer selbst [WCB+18] angemerkt, sehr Performance hungrig wird. Auch wird die Horizontale, obwohl genutzt selten komplett ausgereizt. Es ist generell angenehmer das gesamte Bild zu verkleinern, anstelle die kleinen Räume maximal auszunutzen. Zudem ist es einfacher mit Wordles zu interagieren, wenn diese nicht möglichst dicht aneinander liegen und etwas Spielraum zwischen den Kanten existiert.

4.1.1 Nachbarschaftseigenschaft

Um ein kompaktes und möglichst beständiges Layout zu erzeugen, definiert EdWordle eine Nachbarschaft für Wörter. Zwei Wörter gelten als Nachbar, wenn das Zentrum beider mit einer Geraden Verbunden werden kann, die von keinem andern unterbrochen wird. Diese Gerade ist ein Vektor $V_{nachbar}$ auf dem sich die Wörter beeinflussen können.

Diese Eigenschaft wird sehr oft verwendet und es ist ratsam, diese effektiv umzusetzen. Ein naiver Ansatz ist es Nachbarn mittel Raytracing zu bestimmen. Da alle Wordles schon einen *Collider* besitzen, lässt sich dieser mit der **Physics.Raycast** API querien. Wir iterieren durch alle Wörter und bestimmen deren Zentrum. Dann wird ein Raycast vom Startwort ***A*** ein Raycast in Richtung des potentiellen Nachbarn ***B*** gestartet. Da der Startpunkt des Rays schon in dem *Collider* von ***A*** liegt, wird dieser von Unity ignoriert. Nach erfolgreicher Ausführung liefert der Raycast einen *Collider* eines Wortes ***X***. Ist ***X*** das selbe Wort wie ***B***, so sind ***A*** und ***B*** Nachbarn. Andernfalls sind sie keine und wir haben ein anderes Wort ***C*** getroffen. Es ist hierbei zu beachten, dass das Resultat keine Aussage über das Nachbarschaftsverhältnis zwischen ***A*** und ***C*** tätigt. Es ist möglich das Zwischen den Zentren von ***A*** und ***B*** ein andere *Collider* ***D*** liegt und diese beiden deshalb nicht als Nachbarn zählen.

Raycast sind jedoch sehr teuer. Es ist daher wesentlich effizienter die Nachbarn anderweitig zu berechnen. Um zu überprüfen, ob die beiden Wörter ***A*** und ***B*** Nachbarn sind, wird ein Vektor zwischen den beiden aufgestellt. Dieser verläuft von der Mitte des Wortes ***A*** zur Mitte das Wortes ***B***. Nun werden alle anderen überprüft, ob diese den Vektor schneiden. Falls es einen Treffer gibt, muss dieser nach der Distanz ausgewertet werden. Es kann vorkommen das ein Wort ***C*** den Vektor schneidet, aber sich hinter ***B*** befinden.

Ein Vorteil der Nachbarschaftseigenschaft ist seine bidirektionalität. Wenn Wort ***A*** der Nachbar des Wortes ***B*** ist, dann ist auch ***B*** der Nachbar von ***A***. Dies ermöglicht es die benötigten Iterationen, um alle NxN Nachbarn zu Berechnen von $n * (n-1)$ auf $n(n+1)/2$ zu reduzieren.

4.1.2 Gravitation

Mit den *Rigidbodies* an den Wörtern und Ihrer Nachbarschaften ausgewertet, kann nun die kompakte WordCloud von EdWordle berechnet werden. Um die Wörter kompakt zueinander zubringen, werden zwei wesentliche Kräfte angewendet. Eine, die nahe stehende Wörter näher aneinander bringt und eine andere, die alle zentralisiert. Dank der *Collider*, werden die Wörter davon abgehalten, einander zu überlappen.

Die Kräfte zwischen den einzelnen Wörtern beruht auf der Nachbarschaftseigenschaft. Sind zwei Wörter ***A*** und ***B*** Nachbarn, so wirken sie eine Anziehungskraft aufeinander aus. Diese wird wie folgt definiert:

$$F_A^B = m_A \times m_B / r_{A,B}^2$$

m_A und m_B bilden hier die Masse der einzelnen Wörter. Je größer ein Wort, desto Massereicher ist es und desto stärker wirkt seine Anziehungskraft auf das andere. Die euklidische Distanz zwischen den beiden Wörtern wird als $r_{A,B}^2$ mit einbezogen. Die Kraft ist also invers zum Abstand. Wörter, die weit zueinander entfernt sind, wirken sich nur schwach aufeinander aus.

Am Ende ist die gesamte Nachbarskraft die ein Wort erhält als $F_i^{Nachbar} = \sum_{j=1}^{n_f} m_i \times m_j / r_{i,j}^2$ geben, wobei n_f alle Nachbarn des Wortes $\boldsymbol{i}$ ist. Obwohl die Wörter selbst nur eine zwei dimensionale Anziehung aufeinander wirken, befinden sie sich in AR in einem dreidimensionalen Raum. Die einzelnen Kräfte entsprechen daher der Kraft multipliziert mit dem den normalisierten Nachbarschaft-Vektor $V_{Nachbar}$. Die Kraft, die am Ende auf benachbarte Wörter auf den *Rigidbody* wirkt ist:

$$V_i^{Nachbar} = \sum_{j=1}^{n_f} m_i \times m_j / r_{i,j}^2 \times |V_{m,i}|$$

Damit die Wörter nicht lokal Klumpen und sich zentrieren, wird eine zusätzliche Zentrale Kraft angewendet. Sinngemäß befindet sich ein unsichtbarer Körper im Zentrum, der alle Tags unabhängig ihrer Position zu sich zieht. Diese ist proportional zur Distanz und sorgt dafür, dass weit entfernte Wörter sich zentral sammeln und dann durch Ihre direkte Nähe und Nachbarschaft einander beeinflussen.

$$V_i^{Zentral} = m_i \times M_{Zentrum} \times r_{i,j}^2 \times |V_{m,Zentrum}|$$

$M_{Zentrum}$ bildet hierbei die Masse des Zentrums. Hat ein Wort mehrere potenzielle Zentren, zu dem es gezogen werden könnte, lassen diese sich dadurch gewichten.

Insgesamt lässt sich die Kraft, die ein Wort ***A*** bei jedem Simulationsschritt erfährt als:

$$V_A = V_A^{Nachbar} + \alpha V_A^{Zentral}$$

definieren. α ist hierbei eine zusätzlich Konstante, die es erlaubt die Zentrale und Nachbarschaftskräfte zu einender Gewichten. Zu große zentral Kräfte zerstören die Nachbarschaftseigenschaften und machen den Berechnungsaufwand zunichte.

Werden diese Kräfte nun berechnet und iterativ immer wieder angewandt, kommt es dazu, dass das System nicht in einen still stehenden Zustand gerät. Die Wörter wirken Kräfte aufeinander aus, bis sie zentral Klumpen und kollidieren. Sobald dies geschieht, werden die *Collider* eine Gegenkraft berechnen, um eine Überschneidung zu lösen. Ein perfektes Equilibrium ist dahingegen fast komplett sehr unwahrscheinlich, wenn nicht gar unmöglich. Auch ist das andere Wort, mit dem eine Kollision stattfindet nicht starr und wird leicht weggedrückt. Dadurch gerät das Layout in eine leichte, oszillierende Schwingung.

Um dem entgegen zu wirken muss die Kraft, die auf die Wörter Wirkt zunehmend reduziert werden. EdWordle führt hierfür eine DampingForce (Dämpfende Kraft) ein. Diese reduziert die Kraft V_A mit zunehmender Dauer. Dadurch kommt das System langsam zum Stillstand. Für jede Iteration t ist die Gesamtkraft die auf ein Wort wirkt wie folgt:

$$F_A^{Damp}(t) = F_A(t) \times g(t)$$

Hier bei ist $G(t)$ der Dämpfungsfaktor im Schritt **t**. Dieser wird durch $g(t) = \beta/(t+1)$ bestimmt. Mit β als Schwächungskonstante. Dies schwächt aber nur die neuhinzuwirkende Kräfte ab, bereits existierende Wirken weiterhin. Um diese zu minimieren werden die Bewegungen mit einem Faktor von $0,8$ pro Schritt multipliziert und reduziert. Um minimales Zittern komplett auszuschließen, gibt es zudem einen fixen Treshhold. Bewegen sich die Körper unter einer gewissen Geschwindigkeit, so wird diese genullt. Zudem wird auch eine maximale Iterationszeit gesetzt.

4.1.3 ReWordle

Nachdem die Simulation beendet ist, sind die Worte nahe aneinander gerückt. Dennoch kann es dazu kommen, dass das Layout sehr hoch oder breit ist. Da die *Collider* fixe Boxen sind. Können sich Wörter nur schwer um einander bewegen. Ein initiales Layout, das von Anfang an einen Turn gleicht, wird bei der Simulation nicht in die Breite wachsen. Selbiges gilt für ein initiales Layout, das die Horinzontale präferiert. Um dem entgegenzuwirken gibt es die Funktion "Rewordle". Deren Ziel ist es, Wörter wieder gleichmäßiger um das Zentrum zu verteilen. Rewordle führt dafür zwei Schritte aus. Zuerst werden Grenzwörter bestimmt und dann für diese eine neue Position berechnet.

Um Grenzwörter zu bestimmen wird eine Box b um alle Wörter berechnet. Diese besteht aus der minimalen und maximalen X/Y Koordinate der Ränder aller Wörter. Sie ist also die Kleinst mögliche Box, die um alle Wörter gezogen werden kann und keines schneidet. In dessen Zentrum liegt ein Grenzkreis, dieser hat den Radius von $\beta \times min(breite_b, höhe_b)$. Alle Wörter außerhalb dieses Radius gelten als Grenzwörter und sind Kandidaten die von einer anderen Position profitieren könnten. Sie alle werden vorerst alle aus dem Wordle entnommen und nicht weiter für die neuen Positionen in Betracht gezogen. Um eine neue Position zu bestimmen wird erneut eine Spiraleniteration durchgeführt. Um das Wort möglichst nahe an der alten Position zu halten, ist das Zentrum der neuen Iteration die Mitte zwischen dem Gravitationzentrum und der Mitte das Wortes. Die Spirale wird nun wie anfangs schrittweise abgelaufen, bis eine neue freie Position gefunden wird. Um diese möglich getreu der alten zuhalten, werden auch die neuen Nachbarn berechnet und die Schnittmenge

mit der alten ausgewertet. Es kann dazu kommen, dass die neue Position zwar nahe der alten liegt, aber nur noch wenige Nachbarn beibehält. Um dieses Kriterium möglichst zu bewahren, wird die Spiraleniteration weiter ausgeführt, bis k Positionen Kandidaten gefunden wurden. Dann wird aus diesen die gewählt, die die meisten Nachbarn beibehält und nahe dem Spiralenzentrum liegt.

Dieses Iterationsverfahren wird auch dazu genutzt neue Positionen zu bestimmen, wenn es zu einer Überlappung kommt. Dies kann geschehen, wenn der Anwender mit den Wordles interagiert.

4.2 Wordles in Formen mit ShapeWordle

Um Wörter in Position zu bringen, benutzen die meisten gängigen Wordles eine archimedische Spirale. Dies ist eine euklidische Spirale, die von (0,0) aus gleichmäßig wächst. In Polarkoordinaten ist sie definiert als:

$$r(\theta) = m\theta + b$$

Hier ist θ der Polarwinkel, r die radiale Distanz, b die initiale Distanz vom Startpunkt und m kontrolliert den Abstand zwischen den Windungen. Einen einheitlichen Windungsabstand von $2m\pi$ zu wählen ist eine wichtige Charakteristik der archimedischen Spirale. Dies erlaubt es gleichverteilt und kompakt den Raum zu füllen. Dadurch können Wörter möglichst eng zueinander initialisiert werden.

Die Welt in Unity und Bildinformationen lassen sich am einfachsten mit einem kartesischen Koordinatensystem ausdrücken. Um eine archimedische Spirale in diesem Raum darzustellen, bedarf es der Hilfe der trigonometrischen Funktionen.

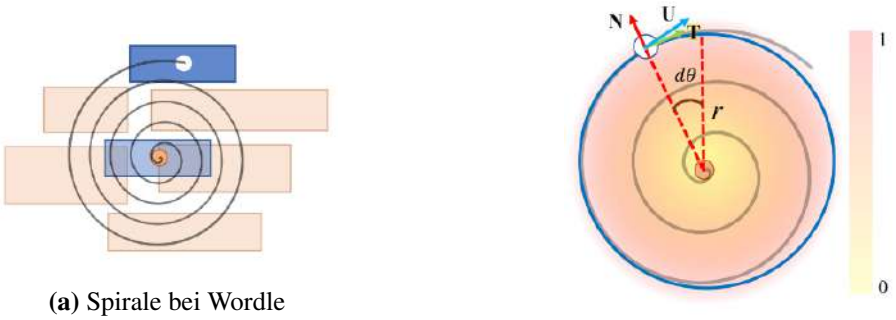
$$\begin{pmatrix} x \\ y \end{pmatrix} = r(\theta) \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$$

Um die Spirale schrittweise ablaufen zu können, wird, diese nach θ abgeleitet. Der Iterationsschritt lässt sich nun wie folgt darstellen:

$$\begin{pmatrix} \frac{dx}{d\theta} \\ \frac{dy}{d\theta} \end{pmatrix} = \begin{pmatrix} m\cos\theta - r(\theta)\sin\theta \\ m\sin\theta + r(\theta)\cos\theta \end{pmatrix}$$

Dies ist Differenz zu dem jetzigen Punkt (x, y) an θ bzw. dem Iterationsschritt **i** und der nächsten Position **i+1** dieser lässt sich in zwei Vektoren N und T zerlegen. N ist der Normalenvektor weg vom Zentrum der Spirale und T ist der Vektor der Tangente entlang eines Kreises. Der Kreis hat einen Radius, der gleich der Distanz der Spirale zu ihrem Zentrum ist. Dieser kann auch als Isolinie eines Distanzfeldes $\phi(x, y) = \sqrt{x^2 + y^2}$ interpretiert werden. Dieses Distanzfeld bestimmt und steuert die fortlaufende Iteration der Spirale. Wird dieses nun durch eine anderes mit einer Form ersetzt, wird sich diese verformt nach außen winden.

Ein Distanzfeld ist eine Abbildung, die $\mathbb{R}^2$ nach $\mathbb{R}$ abbildet. Sie gibt für eine Koordinate (x,y) die kürzeste Distanz zur Kontur an. Eine Kontur ist eine Menge von Punkten, die eine Region komplett eingrenzt. Formal definiert lautet die Abbildung wie folgt:



(a) Spirale bei Wordle

(b) Richtungsvektoren der Windung

Abbildung 4.1: Archimedische Spirale Quelle: [WCZ+20]

$$\phi(p \in \mathbb{R}^2, \Omega) = \min_{q \in \omega} \|p - q\|$$

p ist die Koordinate (x,y), Ω die Kontur und q die Punkte der Kontur. ϕ wird größer, je weiter p sich von der Kontur entfernt.

Um nun die Spirale einer Form folgen zu lassen, wird das Distanzfeld durch das der Form ausgetauscht. Um die Berechnung von Isolinien zu ersparen, ersetzt der Gradient des Distanzfeldes N [GK96]. Diese Ersetzung ist in der Lage, für kontinuierliche Skalarfelder die N zu bestimmen. Sobald N bekannt ist, ist T einfach zu berechnen, da dieser orthogonal zu N steht. Die Endformel lautet dann wie folgt:

$$\begin{pmatrix} dx \\ dy \end{pmatrix} = md\theta N + rd\theta T$$

Dies erzeugt nun eine Spirale, die sich von Zentrum aus nach außen windet und dabei gleichmäßig die Form füllt. Bei Formen mit einer hohen Krümmung kann es dazu kommen, dass die Spirale nicht in der Lage ist, Spitzen sauber auszufüllen. Um kleine Regionen und scharfe Kanten weiterhin zu berücksichtigen, wird die lokale Krümmung weiter untersucht. Dies geschieht durch eine Approximation der Kurve durch zwei kleine tangentiale Bewegungen. Diese sind senkrecht zu N und werden bestimmt durch $Rd\eta$ und durch $rd\theta$. R ist der lokale Krümmungsradius und η die Winkelgeschwindigkeit entlang der lokalen Krümmung. Letzteres ist ein freier Parameter der gewählt werden kann. Je größer, desto stärker beeinflusst die lokale Krümmung die Spiralen Iteration. Damit kann $d\theta$ durch Folgendes ersetzt werden:

$$d\theta = d\frac{R\eta}{r}$$

Hierbei ist r die lokale Krümmung des generischen Distanzfeldes die der ursprünglichen archimedischen Spirale entspringt. Wobei R die Krümmung des geformten Distanzfeldes ist. Um R an einen beliebigen Punkt zu berechnen, wird die Hesse-Matrix verwendet.

Um nun vollständig die Spirale abzulaufen sieht die Funktion zur Berechnung eines einzelnen Iterationsschrittes wie folgt aus:

$$\begin{pmatrix} dx \\ dy \end{pmatrix} = md\frac{R\eta}{r} + dR\eta T$$

Hierbei ändern sich R und r mit jedem Schritt. R abhängig der lokalen Krümmung und r abhängig zum Zentrum. Die zwei Parameter m und $d\eta$ sind konstant und frei wählbar. m kontrolliert die Windungen der Spirale direkt. Die Größe m bestimmt die einzelnen Windungsabstände und die Form füllt sich entsprechend schneller oder langsamer. Die Stärke des Einflusses der lokalen Krümmung lasst sich mit $d\eta$ steuern.

4.3 Wordle Interaktionen mit ManiWordle und WordlePlus

Um dem Nutzer zusätzliche Interaktionen zu bieten, werden einige Features von anderen Wordles implementiert. Diese sind:

1. Bewegen
2. Skallieren
3. Farbe ändern
4. Konkatenieren
5. Mergen

Wird ein Wort ausgewählt wird es für weiter Interaktionen Gehighlitet. Der Anwender kann es nun greifen und es durch den Raum über den Canvas ziehen. An den Ecken befinden sich zusätzlich Highlights, an denen das Wort Skalliert werden kann. Das Gewicht, das für die EdWordle Simulation verwendet wird, wird entsprechend angepasst. Über das UI kann die Farbe abgeändert werden. Dort befindet sich im zugehörigen Untermenü drei Regler. An diesen kann der Farbanteil von Rot, Grün und Blau manipuliert werden. Zusätzlich kann das Überlappungsverhalten geändert werden. Bei Konkatenieren und Mergen wird anstelle Wordles neu zu positionieren, erst das Wort mit der größten Überlappung bestimmt. Ist diese über 30%, so werden die beiden entweder konkateniert oder gemerged. Das neu entstandene Wort erhält die Originalgröße der größten Überlappung und bekommt zusätzlich 50% des drüber bewegten.

Bei einer neuen Position wird überprüft, ob sich das Wort noch in der gleichen Region befindet. Kommt es in eine neue Region, wird diese dem Wort zugewiesen und es erhält den nächstbesten Extrempunkt dieser als Zentrum des dazugehörigen Distanzfeldes.

Müssen andere Wörter der neuen Position weichen, so werden für diese iterativ die Distanzfelder der zugehörigen Region abgelaufen. Der Startpunkt bildet hierbei der Mittelpunkt zwischen dem Zentrum der Region und der Originalposition des Wortes. Zusätzlich wird die Spiralwindung m abhängig der Wortgröße angepasst. Große Wörter können sich größere Schritte leisten, bevor eine neue passende Position gefunden wird. Kommt es dabei vor, dass die Region komplett erschöpft wird, fällt der Algorithmus auf die klassische archimedische Spirale zurück. Damit wird gewährleistet, dass immer eine neue Position eventuell gefunden wird. Um die Nachbarschaftseigenschaft zu erhalten, werden sieben mögliche Platzierungen bestimmt und ausgewertet. Das Wort wird der Position zugewiesen, die die meisten Nachbarn teilt und nahe der Anfang der Iteration ist.

```
Überprüfe ob neue Position in Region befindet
Wenn nein:
    Überprüfe ob neue Position in anderer Region
    Wenn ja:
        Weise neue Region und nächstbesten Extrempunkt zu
    Wenn nein:
        Weise den nächstbesten Extrempunkt und dessen Region zu
Überprüfe alle Wörter nach einer Überlappung
Wenn Modus Konkatenieren oder Merge:
    Bestimme größte Überlappung
    Wenn größte Überlappung > 30\%:
        Passe das Wort mit der größten Überlappung an
        Wechsel Modus in Bewegen und gehe zu Schritt 8

Für alle Wörter mit Überlappung:
    Bestimme alle jetzigen Nachbarn
    Setzte Spiralen Zentrum zwischen Extrempunkt und jetzige Position.
    Für 7 Iterationen:
        Gehe entlang des Distanzfeldes, bis eine neue Position gefunden wird
        Wenn Formspirale ausläuft:
            Laufe entlang der archimedischen Spirale
        Bestimme die Schnittmenge der neuen Nachbarn mit den alten
    Wähle Position, die diegrößtee Nachbarschnittmenge hat
```

4.4 Platzierung in Augemnted Reality

Um die Wordles in der Welt richtig zu Positionieren Bedarf es einer Auswertung des Umfeldes. Hierfür wird das SpatialMesh der Hololens Observer ausgewertet (mehr zu in Abschnitt **??**). Mit einem Raycasts von der Kamera aus, kann die Distanz zur Umgebung ausgerechnet werden, dies erlaubt es den Punkt für die Platzierung des Canvas zu bestimmen. Um die Wordles richtig in die Umgebung zu legen, werden vier zusätzliche *Raycasts* um die Kamera herum ausgeführt. Mit diesen vier Punkten kann eine Fläche aufgespannt werden, dessen Orientierung der Fläche vor dem Nutzer gleicht. Damit ist es möglich das Wordle in der Umgebung richtig zu auszurichten.

4.5 Worldes schnell berechnen

Anders als bei einem einfachen Bild ist es ungünstig, ein ganzes Wordle auf einmal zu berechnen. Dauert dies im Hauptthread zu lange wird kein neues Bild gezeichnet und der Anwender sieht nichts. Um auch eine Benutzbarkeit zu gewähren sollten Berechnungen dort nicht länger als 33ms (30FPS) oder 16 ms (60FPS) in Anspruch nehmen. Wordle Algorithmen besitzen aber grundlegend das Problem eine hohe asymptotische Laufzeit zu haben. Für jede neue Position muss ein Wort mit bis zu allen andern bereits Platzierten überprüft werden. Das sorgt im worse-case für eine Komplexität von O(n*i). Wobei n die Anzahl aller nicht überschneidenden Wörter und i die Anzahl aller Schritte ist

bis eine neue Position gefunden wird ist. Um also den Wordle Algorithmus effizient durchzuführen muss n und i möglichst klein gehalten werden. i lässt sich direkt mit der Windungsgeschwindigkeit m aus Abschnitt 4.2 beeinflussen. Je größer m gewählt wird desto schneller wird die Fläche aufgebraucht und weniger Schritte sind von Nöten. Der dabei entstehende Nachteil ist, das die Wörter unter Umständen nicht so kompakt sind wie sie sein könnten. Dieses Problem kann aber durch EdWordle gelöst werden. Dies erlaubt es ein großzügigeres m zu wählen. Für ein möglichst kleines n Bedarfes entweder eine kleine Anzahl von Wörtern oder eine geschickte Iteration. Sobald bei der Suche eine Überschneidung zweier Wörter auftritt, kann diese beendet werden und in die nächste übergehen. Um dies maximal zu nutzen, empfiehlt es sich, die Wörter vor hinein zu sortieren. Hierbei helfen die Extrempunkte von ShapeWordle. Ein Wort überschneidet wahrscheinlicher ein anders, dass die gleiche Spirale oder genauer gesagt den identischen Extrempunkt teilt. Werden diese zuerst durchlaufen, lassen sich Kollisionen früher feststellen und Zeit wird gespart.

5 Projekt Implementierung

Für die Umsetzung von Wordles in einer Augmented Reality wurde als Plattform die Hololens verwendet. Dieses Kapitel beschreibt zuerst die dafür verwendeten ThirdPartyTools (Abschnitt 5.1) und dafür benötigten Initialisierungen (Abschnitt 5.2). Danach werden die grundlegenden Elemente der Applikation in Abschnitt 5.3 und 5.4 vorgestellt. Die Umsetzung der QR-Codes und der Bildverarbeitung der Formen ist in 5.5 und 5.6 beschrieben. Zuletzt beschreibt Abschnitt 5.7 die Möglichkeiten die Applikation zu testen und zu debuggen und was dabei zu beachten gilt.

5.1 ThirdPartyTools

Um nicht alle notwendigen Tools selbst implementieren zu müssen, wurde auf eine Vielzahl von verschiedenen 3rd Party Tools zurückgegriffen. All diese sind frei erhältlich und bieten eine Vielzahl von nützlichen Funktionen, die die Entwicklung wesentlich vereinfachen. In den meisten Fällen baten sich sogar alternative Implementierungen an. Beide sind in diesem Abschnitt Beschrieben und mit Abwägungen versehen.

5.1.1 Unity

Zur Umsetzung in AR bedient sich die Arbeit der Unity Crossplattform Gameengine. Diese wird von Unity Technologies entwickelt und ist frei für nicht kommerzielle Zwecken erhältlich. Kosten fallen nur ab einen gewissen jährlichen Umsatz an. Unity wurde 2005 das erste Mal auf der Worldwide Developers Conference von Apple Inc. vorgestellt. Das Ziel war es eine MacOS exklusive game Engine zu entwickeln. Über die Jahre hinweg wurde Unity immer erfolgreicher und expandierte über MacOs hinaus und unterstützt heutzutage alle erdenklichen Plattformen und Spielekonsolen. Seit der Umstellung von einem reinen bezahlten Lizenzmodell auf einen auch teils kostenloses erfreut sich Unity einer wachsenden Beliebtheit. Insbesondere durch die leichte Bedienung ohne große Programmierkenntnise wird Unity zu einem ziemlichen Go-to Produkt für ziemlich alle möglichen Anwendungsfälle. Von Videospielen bis hin zur Unterstützung in Film Produktionen wie Disney Lionking 2019 [Vyn20]. Seit der Unterstützung von AR in Unity schaffte diese auch neue Interessenten weit über den Entertainmentmark hinaus zu gewinnen [Ede18]. Dank der Unitys selbst erstellten Framework ÄR-Foundationïst es Anwender möglich, Entwicklungen für AR durch einen vereinheitlichten Workflow auf alle möglichen Geräten umzusetzen.

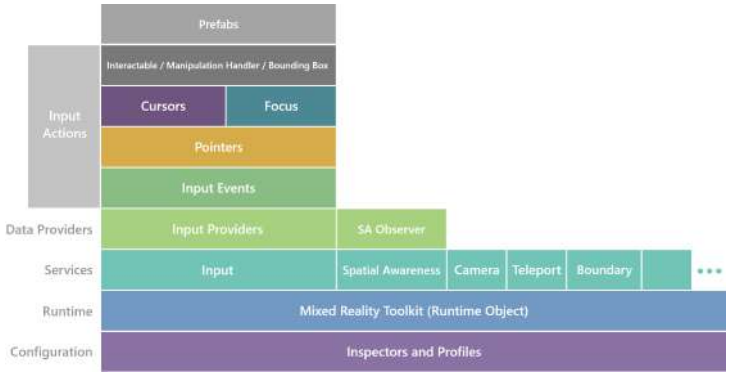
Unity besteht grundsätzlich aus verschiedenen Szenen zwischen denen gewechselt werden kann. Eine Szene enthält sogenannte GameObjects. Diese sind Instanzen mit einer Skalierung, Rotation und X,Y,Z-Koordinate. Diese können um Komponenten erweitert werden. Komponenten sind das, was der User meist nachher wahrnimmt. Dies sind unter anderem eine Geometry, eine Textur, eine Audioquelle und eine Skriptdatei mit dem Verhaltensmuster des Objekts. Diese

Verhaltensskripte, MonoBehaviors genannt sind ein Grundbaustein von Unity und ermöglichen es den Programmierer jegliche Logik seinem Programm zu geben. Unity selbst läuft in C++, diese Skripte sind jedoch in .Net Bzw. C geschrieben. Dieser Ansatz ist einer der grundlegenden Ideen von Unitys Benutzerfreundlichkeit. Programmiersprachen wie C sind wesentlich Anwender freundlicher als C++ und das Handhaben der Logik als geschriebene SZusätzeërmöglicht es, Fehler auszulösen, ohne dass das gesamte Programm abstürzt. Die Monobehaviors können durch Ihre Natur jederzeit zur Laufzeit hinzu und entladen werden. Das ermöglicht eine Entwicklung und Editierung der Logik, während das Unity Programm aktiv am Laufen ist, und zusätzliche Kompilierungen entfallen fürs Erste.

5.1.2 Mixed Reality Toolkit (MRTK)

Neben der Crossplattform Unterstützung von unity hat Microsoft sein eigenes Mixedreality Toolkit vorgestellt [Mice]. Windows Mixed Reality wurde als Teil von Windows 10 eingeführt. Das Ziel war es, damit unteranderem die eigens entwickelte Virtual-Reality-Brille Hololens zu bewerben. Das Ziel mit MRTK ist eine uniforme und einfache Toolbox für alle Anwender und Applikationen zu schaffen. Darüber hinaus soll sie auch in der Lage sein, auf offene Standards wie OpenVR zurückzufallen und dadurch andere Plattformen zu unterstützen. In Abbildung 5.1 ist eine Highleveldarstellung des MRTKS zu entnehmen.

Abbildung 5.1: Aufbau des MRTKs [Micb]



Ganz unten Stehen generelle Konfigurationen und Profile. Diese können je nach Target Plattform ausgetauscht und abgeändert werden. So kann derselbe Code mit nur dem Wechsel der Konfiguration von der Hololens 2 auf die ältere Hololens 1 gebracht werden. Dadrüber liegt die MRTK Runtime. Diese ist eine einzige Singletoninstanz die alle MRTK Tools managet. Da zu zählen das Bereitstellen und Organisieren von Services, Events und die Berechnung von Hologrammen. Auf der Runtime liegen die Services. Diese können individuell zu Laufzeit hinzu und abgeschaltet werden. Diese Services sind Anbieter für den Programmierer Informationen zur Umwelt zu enthalten. Für konkrete Informationen dienen die Data Provider. Diese Verarbeiten die Eingaben und bereiten diese für Services vor. So kann ein Input Provider ein Meshprovider sein, der aus den Sensoren der Hololens eine Meshumgebung erstellt. Zuletzt liegen auf dem MRTK die Inputaktionen. Diese sind alle Informationen und Eingaben, die vom User aus kommen und umweltunabhängig sind. Hierzu zählt die Gestik des Benutzers, ob er gerade mit einem Hologramm interagiert, wo der Fokus seiner Augen liegt etc pp.

5.1.3 Open Computer Vision (OCV)

OpenCV ist eine Open Source Bildverarbeitungsbibliothek [BK08]. Sie wurde ursprünglich 1999 von Intel entwickelt, um CPU intensive CV Probleme effektiv und Effizienz zu lösen. Die anfänglichen Ziele des Projekts waren:

1. Cumputer Vision mit offenen und optimierten Code voranbringen um eine gemeinsame Grundlage zu schaffen
2. Eine gemeinsame Infastruktur bereitzustellen, so dass Code Änderungen und Übertragungen einfach sind.
3. Computer Vision Anwendungen kommerziell voranbringen, in dem portabler, offener und Lizenzfreier Code zur Verfügung gestellt wird.

Die erste Alphaversion wurde auf der Conference on Computer Vision and Pattern Recognition (CVPR) im Jahre 2000 vorgestellt und erhielt seitdem regelmäßige Updates. Mittlerweile ist OpenCV für alle gängigen Plattformen erhältlich und unterstützt eine breite Vielfalt an verbreitetem Bildverarbeitungsalgorithmen. Zusätzlich bietet OpenCV inzwischen auch eine Bibliothek für maschinelles Lernen. OpenCV ist an sich in C/C++ geschrieben. Um eine besser portabilisierung zu gewährleisten, gibt es zu dem offizielle Interfaces für Java und Python. Da OpenCV grundsätzlich free-to-use ist, haben sich über die Jahre weitere OpenCV Wrapper und Umsetzungen etabliert. So gibt es Warpper in .Net und Implementierungen in JavaScript und anderen Sprachen.

5.2 Projekt Initialisierung

Das Projekt besteht zuerst aus einer neuen 3D Anwendung in Unity 2020.3.21f1. Unity 2017 ist mindestens für AR mit MRTK vorausgesetzt. Es ist zu empfehlen den Projekt Ordner möglichst nah am Root der Festplatte zu wählen und den Projektnamen kurz zu halten. Das Problem das Hierbei Auftreten kann, ist die `MAX_PATH_` Limitierung von Windows, die auf 255 Zeichen beschränkt ist. Einige Pfade werden im MRTK sehr lang und es kann zu Problemen führen, das gewisse Dateien nicht mehr gefunden werden. In Unity selbst sind die Buildsettings (Ctrl+Shift+B) wie folgt:

1. Scenes in Build: Wordl/WordlScene 2. Platform: Universal Windows Platform 3. Target Device: HoloLens 4. Architektur: ARM64 5. Build Type: D3D Projekt 6. Target SDK Version: 10.0.19041.0 7. Visual Studio version: 2019 8. Build Configuration: Release

5.2.1 Mixed Reality Toolkit

Danach wird das Mixed Reality Feature Tool ausgeführt. Es kann im Microsoft Download Center (https://aka.ms/MRFeatureTool) heruntergeladen werden. Nach dem Starten des Feature Tools muss der Projekt Pfad ausgewählt werden. Es erkennt dann das Unity Projekt und die neusten Features können nun heruntergeladen werden. Für die Arbeit benötigt es mindestens die **Mixed Reality Toolkit -> MixedReality Toolkit Foundation** und **Platform Support Mixed -> Reality OpenXR Plugin**

Empfehlenswert sind hierbei die Examples, diese fügen eine Vielzahl von Templates und fertiggestellten Prefabs zum Projekt hinzu. Auch Beispielszenen mit besagten Prefabs sind enthalten. Nach Abschluss der Installation führt Unity eine Reihe von eigenen Aktualisierungen durch und es öffnet sich im bestenfalls das MRTK -Hilfe-Fenster in Unity. Dieses Hilfefenster stellt nun einen Leitfaden für weiter projektspezifische Einstellungen. Andernfalls kann dieser unter den neuen Reiter im Menü **Mixed Reality -> Toolkit -> Utilities -> Configure Projekt for MRTK** geöffnet werden. Es ist eine Schritt-für-Schritt Anleitung das Unity Projekt vollends zu konfigurieren. Es ist in der Lage, die meisten Settings von selbst um zusetzen und benötigt bei ein paar Einstellungen manuelle Hilfe bei den Settings. Es kann vorkommen, dass eine Vielzahl von Warnungen in der Konsole erschienen. Diese geben weiter Hinweise auf noch nicht vollständig eingerichtete Einstellungen. Zusätzlich ist es möglich, unter **Mixed Reality -> Project** empfohlene Konfigurationen für die Hololens 2 einzurichten. Um das MRTK in die Szene einzubinden, lässt sich dieses unter **Mixed Reality -> Toolkit -> Add to Scene and Configure...** hinzufügen. Dies fügt zwei GameObjects mit dem Namen **MixedRealityToolkit** und **MixedRealityPlayspace** hinzu. Das **MixedRealityToolkit** enthällt das **MixedRealityToolkit** MonoScript. Hier lassen sich Konfigurationen mittels Profile ändern und zusätzliche Features an und ausschalten. Das **MixedRealityPlayspace** enthält zu Beginn die **Main Camera**. Hier kommen zur Laufzeit alle Möglichen zusätzlichen, von **MixedRealityToolkit** erzeugten und gemanagten Objekte hinzu. Dazu Zählen das Diagnosefenster mit Fps und Speicherverbrauch. Hier befindet sich auch der Cursor und Handskelette vom User.

5.2.2 OpenCV

Zur Bestimmung der Formen und Auswertung der Kamera Bildern bedient sich diese Arbeit OpenCV. Da OpenCV nicht direkt in Unity verwendet werden kann, benötigt es einen Wrapper. Es gibt eine Vielzahl von verschiedenen Einbindungsmöglichkeiten für diese Arbeit wurden zwei näher betrachtet. **OpenCV for Unity** aus dem Unity Asset Store und **OpenCVSharp**.

OpenCV for Unity

OpenCV for Unity ist ein unitypackage von Enox Software (https://enoxsoftware.com/). Es kann direkt über den Unity Asset Store erworben und installiert werden. Alternativ lässt sich es unter **Assets -> Import Package -> Custom Package** Importieren. Dabei öffnet sich ein Dialog mit der Auswahlmöglichkeit, nur bestimmte Teile des Pakets hinzuzufügen. Die Mindestanforderung hier sind der **org**-, der **Editor**-Ordner, **Notices.txt**, **EnoxSoftware.OpenCVForUnity.asmdef**, **OpenCVLicense.txt** und die **WSA** und **Windows**-Ordner in **Plugins**. Zusätzlich bittet OpenCV eine kleine Vielzahl an Beispiel Szenen und Skripte in **Examples**. Es ist unter Umständen nötig, die VisualStudio Projekt Datei neu zu genieren, damit der OpenCVForUnity Namespace erkannt wird. OpenCVforUnity ergänzt OpenCV um eigene Funktionen, die für den Umgang mit Unity nützlich sind. So gibt es fertige Funktionen die es ermöglichen direkt zwischen OpenCVs Mats und Unitys Texture2D zu konvertieren. Eine **README** für OpenCVforUnity kann dem neu hinzugefügten Ordner in Asset entnommen werden. Dort sind alle Konfigurationsschritte gelistet und es gibt einen kurzen QA-Troubleshoot-Anhang. Die API für OpenCVforUnity lässt sich der Website (https://docs.opencv.org/4.5.0/) entnehmen. Diese gleicht der von OpenCV Java 4.5.0.

Da OpenCVforUnity als DLL in Unity geladen wird, müssen diese für das Deployment auf der HoloLens konfiguriert werden. Im Ordner **OpenCVForUnity/Plugins/WSA/UWP** sollten alle DLLs in ARM, x86 und x64 Als inkludierte Platform den WSA-Player besitzen und als SKW UWP haben. Die CPU ist dem Ordner Namen entsprechend anzupassen.

OpenCVSharp

Eine Kostenfreie Alternative für OpenCVforUnify stellt OpenCVSharp. Dies ist eine .NET wrapper für OpenCV und ist öffentlich mit der Apache License 2.0 im Git(github.com/shimat/opencvsharp) erhältlich und Verwendbar. Es lässt sich in Unity als NuGet einbinden (Zur Installation von NuGet siehe QR Einbindung). Dieser Wrapper bietet eine API getreue Implementierung von OpenCV, jedoch keine direkte Unity Unterstützung. Im laufe der Arbeit war es leider nicht möglich OpenCVSharp auf der HoloLens zum laufen zu bringen. Die Hürde hierbei liegt es die dafür benötigte DLL richtig im deployeten Programm einzubinden. Unity verpackt diese und überträgt sie mit auf die HoloLens, jedoch kann es dennoch zu DLL nicht finden oder hatte Probleme diese richtig zu binden.

5.2.3 QR Einbindung

Um Microsofts QR Implementierung nutzbar zu machen, sind eine Handvoll von Schritten notwendig. Leider bittest das Mixed Reality Feature Installation Tool von sich aus keine Installation eines QR-Pakets an. Dieses muss zusätzlich von als NuGet von Microsoft geladen werden. Dieses ist unter (https://www.nuget.org/packages/Microsoft.MixedReality.QR) erhältlich. Um NuGets Bestmöglich in Unity zu integrieren empfiehlt sich **NuGetForUnity** (https://github.com/GlitchEnzo/NuGetForUnity). Dies ist ein NuGet Client der innerhalb des Unitys Editor funktioniert. Dieser kann als **.unitypackage** von Github geladen werden. Dieses Package kann dann unter **Asset -> Import Package** hinzugefügt werden.

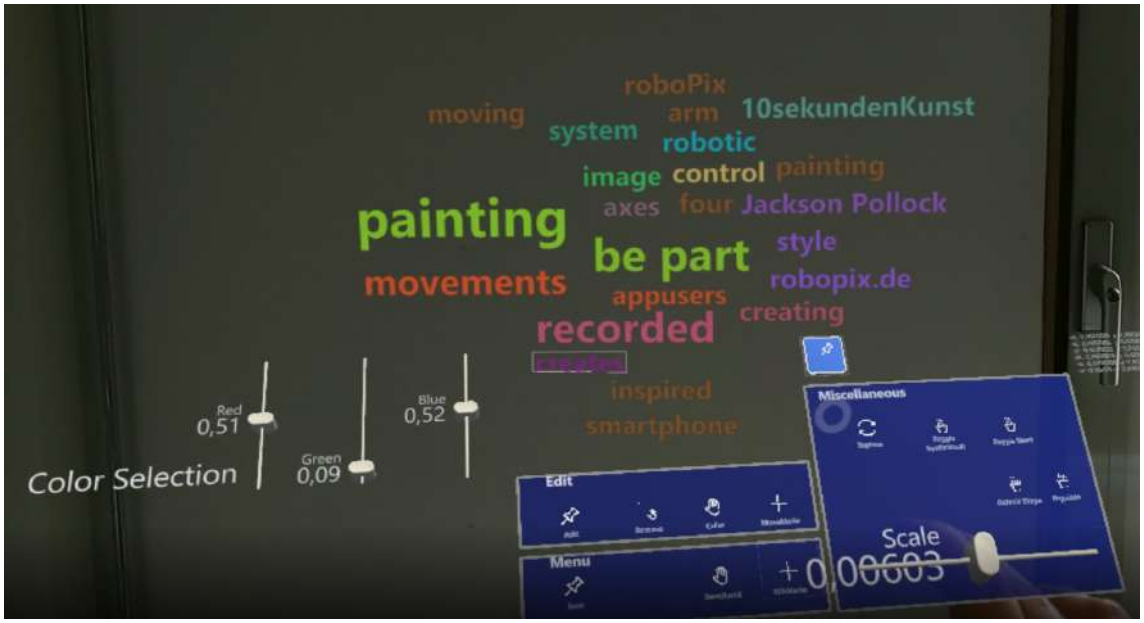
Nach der Installation von NuGetForUnity wird dieser als neuer Reiter in der Menüleiste von Unity als **NuGet** erscheinen. Unter **Manage NuGet Packages** können die installierten NuGets verwaltet werden. Es ist wichtig zu beachten, das die hier installierten NuGets nicht direkt in die VisualStudio Projekt Datei eingebunden werden. Sie sind also im NuGet Manager von VisualStudio nicht sichtbar. Daher kann es dazu kommen, dass der NameSpace nicht von Visual Studio erkannt wird. Es ist daher eventuell nötig die Projektdatei neu zu generieren. Auch kann es zur Asynchronität zwischen NuGetForUnity und Unity selbst kommen und installierte NuGets werden nicht gelistet. Ein Neustart von Unity behebt dies in der Regel. Neben **Microsoft.MixedReality.QR** sollte NuGetForUnity auch **Microsoft.VCRTForwarders.140** installieren. Dies ist eine DLL, die für das QR-NuGet benötigt wird.

5.3 GameScene

Unity Programme bestehen grundsätzlich immer aus mindestens einer Szene. Diese bildet das Spielfeld mit Weltenkoordinaten, Kamera und Objekten, die miteinander in dieser Welt interagieren. Für das Projekt reicht eine Szene aus und sie besteht aus vier wichtigen Komponentenverbänden: 1. MixedRealityToolkit 2. WordleCanvas 3. UI 4. RuntimeObjekte

Eine Vorschau der AR-Wordles-Applikation ist in der Abbildung 5.3 dargestellt.

Abbildung 5.2: Wordles in AR



5.3.1 MixedReality Toolkit

Das Mixed Reality Toolkit besteht aus dem GameObject MixedRealityToolkit mit dem MixedRealityToolkitSkript und dem MixedRealityPlayspace. Letzteres enthält die Kamera zum Rendern der Sicht und weitere Objekte, die von MRTK nativ gemanagt werden. Dazu zählen Pointer, Handskelett und andere Eingabedarstellungen. Ersteres verwaltet und konfiguriert die Verfügbaren AR-Systeme. Dazu zählt die Kamera, das Input, das Spatial Awareness-System. Dies sind Systeme die nativ mit der Hololens kommunizieren und wichtige Informationen zur Umwelt und Interaktion zur Verfügung stellen. Zusätzlich können hier weitere Services eingebunden werden, die von der die Applikation verwendet werden.

5.3.2 Worlds Canvas

Das WordleCanvas ist eine Anzahl von ineinander geschachtelten GameObjects. Auf der untersten Ebene sind in die GameObjects des Wordles. Jedes der darüber liegende **Gameobjects** erfüllt eine bestimmte Transformation. Dies ermöglicht es gezielt die lokale Position der Wörter zu verändern, ohne sich mit den Transformationsverhalten in den Weltenkoordinaten auseinandersetzten zu müssen.

Das äußerste GameObject ist das “WordlSpace”. Dieses dient dazu, die Wordles in der Welt zu platzieren. Seine X,Y und Z-Koordinate entspricht der Weltkoordinate der reellen Welt in Unity und die Rotation gleicht die der Ebene auf der sich die Wordles in der Welt befinden. Zusätzlich enthält es eine Skalierung, mit der die gesamte Wordle-Szene skaliert werden kann. In diesem liegt ein Ausgleichsgameobjekt. Dieses hat eine Negativen X und Y Position. Diese entsprechen der halben Canvasgröße. Das Ziel dieses Gameobjects ist es, das Zentrum des Canvas auf die (0,0,0)-Koordinate des WordlSpace zu mappen. Dies erlaubt es uns auf dem Canvas mit ausschließlich Positiven Koordinaten zu arbeiten. Ein Wordle das sich in einem 100x100 Bild an der Koordinate 50,50 befindet liegt dadurch im Zentrum (0,0) des WorldSpace. Im Innersten liegt der Canvas, dies ist der Container für alle Wörter.

5.3.3 User Interface

Das UserInterface besteht aus einer Handvoll von Menüs für alle möglichen Interkationen und Einstellungen. Die meisten Funktionen sind der Übersicht halber in weitere Untermenüs aufgeteilt. Diese spezialisieren sich jeweils auf eine bestimmte Interaktion. Dazu zählt das Svae/Load-Menü das es erlaubt das Wordle in Saveslots zu hinterlegen, das Scan-Menü das es erlaubt QR-Codes und Formen mit den Kameras einzulesen und das Edit-Menü, welches die Bearbeitung von Wörtern erlaubt. Zusätzlich gibt es ein Canvas-Menü, das es erlaubt das gesamte Wordle neu zu platzieren und zu skalieren.

5.3.4 Runtime Objekte

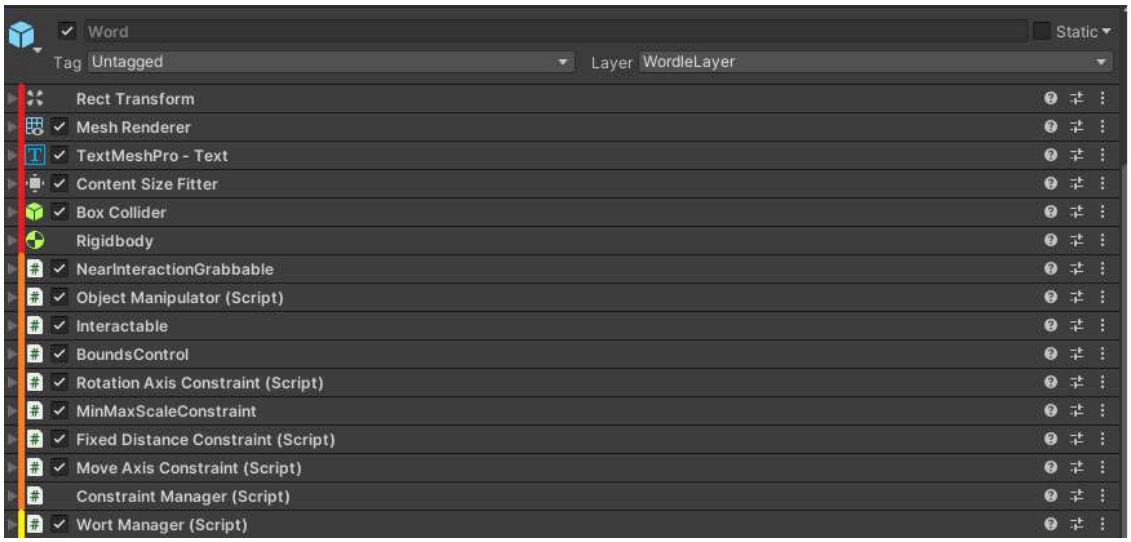
Da für die Ausführung von Skripten diese immer an ein aktives Gameobjekt gebunden sein müssen, gibt es eine kleine Anzahl von Runtimeobjekten. Diese sind dauerhaft aktiv, sind aber für den Anwender weder sichtbar noch interagierbar. Hierzu zählen alle Skripte die durch den Start eine eigens Initialisierung ausführen und oder nicht aus einem statischen Kontext heraus miteinander oder mit Unity interagieren können. Dazu gehört z. b. Die KameraHandler und das MainWorlde Skript, da diese Initial einiges an Set-up vornehmen.

5.4 Das Wordle Prefab

Um zur Laufzeit einfach Wort Instanzen zu erzeugen, wird ein Unity-Prefab verwendet. Dieses enthält eine Menge an Komponenten. Diese lassen sich in 3 Kategorien unterteilen. In der Abbildung 5.3 ist eine Auflistung der Komponenten zu entnehmen. Die drei Kategorien sind farblich hervorgehoben. Rot sind Komponenten von Unity. Sie ermöglichen es, das ein Wordle gezeichnet wird und an

sich existiert. Orange sind Skripte des MRTKs. Diese erlauben es dem Benutzer mit dem Wort zu interagieren. Dazu zählen zusätzlich Constraints. Mit denen das Verhalten bedingt werden kann. So zum Beispiel das *Move Axis Constraint* das dafür sorgt das ein Wort nur entlang der Vertikalen und Horizontalen bewegt werden kann. Als Letztes ist das eigen Entworfene *WortManager* Skript. Dieses handhabt jegliche Logik zum Wort und ist die Schnittstelle für alle anderen Skripte in der Szene zum Wort und für die MRTK Komponenten zur Umgebung.

Abbildung 5.3: Wordle Prefab



5.5 IO auf der Hololens mit QR Codes

Um Wordles zu generieren bedarf es Wörter zum Darstellen. Diese können aus vielen verschiednen Quellen kommen. Die Hololens an sich selbst nur beschränkte Möglichkeiten bietet Informationen von außerhalb zu übertragen. Mögliche Eingabenquellen sind Dateien, die Internetverbindung, Bluetooth, Mikrofon oder Kameras. Dateien müssen manuell immer wieder aufs Neue übertragen werden, was eine einfache und flexible Handhabung erschwert. Die drahtlosen Schnittstellen sind dagegen flexibel und nur gering fehleranfällig. Aber der Aufwand entsprechende Schnittstellen aufzusetzen ist sehr hoch. Es Bedarf für die WLAN-Schnittstelle, einen Webserver, einen freien Port und ein geteiltes Netzwerk. Auch eine Übertragung durch Audio stellt eine Herausforderung da. Die Hololens2 unterstützt von sich aus Text-To-Spech, und würde sich somit eignen Wörter einzusprechen. Jedoch ist es damit umständlich viele Informationen auf einmal und korrekt einzugeben. Die beste Lösung stellt also die Kamera da. Mithilfe von Bildern lassen sich viele Informationen kompakt und direkt übertragen. Eine solche Darstellung ist der QR-Code.

QR-Code können schnell erstellt, verbreitet und angewendet werden. Zudem besitzen sind sie genormt und besitzen eine eingebaute Fehlerkorrektur. Damit lassen sich also viele Informationen schnell und sicher übertragen. Außerdem sind QR-Code weitverbreitet und es gibt viele Bibliotheken, die QR-Codes unterstützen. Ein naiver Ansatz ist es, die benötigten Informationen als JSON String zu decodieren und mit QR-Codes an die Hololens zu übertragen. Jedoch Haben QR-Codes auch Ihre Limitierungen. QR-Code haben eine maximale Kapazität von 23.648 Bit (2.956 Byte) bzw.

4296 Zeichen. JSON ist nicht besonders gut darin kurze Informationen effektiv abzuspeichern. Es erzeugt eine Menge Overhead durch Anführungszeichen, Klammern und Doppelpunkten. Ein deutsches Wort enthält im Schnitt 10,6 Buchstaben [Dud]. Alleine nur für die Codierung mittels Anführungszeichen um jedem Wort kommt es zu einem Overhead von knapp 20%. Wenn jedes Wort als einzelnes Objekt codiert wird, kommt es bei einer schlichten Form von {**"word":"Wort"**} zu einem 50:50-Verhältnis von Wrapper zum Inhalt.

Um Worte also effektiv zu übertragen wurden sie als ein String konkateniert und durch **,** als Delimiter getrennt. Der Einfachheit halber blieb der Container weiterhin ein JSON-String. Dieser enthält die Komponente **T** und **C**. **T** besitzt einen String für den Typ der übertragenen Daten und **C** den entsprechenden Content. Diese Implementierung erlaubt es zukünftig flexible verschiedene Daten einfach zu erzeugen und kompakt ohne viel Overhead zu übertragen. Ein Beispiel QR-Code ist in Abbildung 5.4. Dieser enthällt eine Liste von Wörtern die für die Nutzerstudie in 7 verwendet wurde.

Abbildung 5.4: Beispiel eines QRCodes mit 23 Wörtern

5.5.1 OpenCV und QR Codes

Mit dem Release von OpenCV 4.0 am 18. November 2018, erhielt OpenCV ein eigenes QR-Modul **QRCodeDetector**. Der **QRCodeDetector** bietet die Funktion **Detect** um QRCodes in Mats zu erkennen. Zusätzlich liefert **Detect** Vertices für den QR-Code im Bild. Damit ist es möglich, die Realposition des QR-Codes zu berechnen und zu highlighten. Mit **Decode** lässt sich mit den Vertices und der Mat der QR-Code zu decodieren und dessen Inhalt als String zu bekommen. Es ist zusätzlich möglich die Vertices anderweitig zu bestimmen oder die Funktion **DetectANdDecode** zu verwenden um beides in einem Schritt zu tun.

Der Vorteil von QRCodeDetektion durch OpenCV liegt darin, das OpenCV schon für die Formerkennung genutzt wird. Die dafür benötigte Bibliotheken und Verweise sind dementsprechend bereits integriert. Leider ist diese Methode nicht sehr robust. OpenCVs QR-Code-Detektor hat Schwierigkeiten, bestimmte QR-Code Versionen zu erkennen. Zudem benötigt OpenCV ein fertiges Bild zur Verarbeitung.

Es muss daher die Kamera solange angesteuert werden und dessen Bild ausgewertet, bis ein QR-Code erkannt wird. Das kann auf Dauer sehr viel Leistung einnehmen. Auch ist es schwer, den einmal erkannten QR-Code zu tracken. Der QR-Code-Detektor von OpenCV eignet sich daher am besten für QR-Code niedrigen Versionen mit wenig Informationen, da diese mit am einfachsten zu erkennen sind. Dies limitiert aber die Menge an Daten die übertragen werden können.

5.5.2 MRTK und QR Codes

Die Hololens 2 Bietet von sich aus einen QR-Code-Support an. Alleine wenn man sich im Hauptmenü befindet, überprüft Sie fortlaufend die Umgebung nach QR-Codes und hebt diese mit einem Play-Button hervor. Die Hololens verwendet zum Erkennen der QR-Code nicht die Webcam, sondern ihre Trackingkameras. Dies erlaubt es Ihr QR-Codes schon außerhalb des Sichtfelds vom Nutzer zu erkennen und zu tracken. Zusätzlich Verbrauchen die Trackingkameras wesentlich weniger Strom als eine aktive WebCam. Die Auswertung und Verfolgung läuft auf der Holons in der Driver/System-Ebene. Sie ist daher sehr performant und geschieht parallel zu anderen Programmen. Es ist daher nur nötig, den Driver nach den erkannten QR-Code zu fragen und der Rest wird vom System übernommen.

Das ursprüngliche Konzept hinter der nativen Verfolgung ist es, QR-Codes als Spatial Anchors zu verwenden. Dies erlaubt es die virtuelle Welten von mehreren Hololenses zu synchronisieren. D Das native Hololens-Qr-Tracking unterstützt die QR Versionen 1-10 und alle Micro QR-Code Versionen. Andere artistisch abgeänderte QR-Code wie Logos, Invertierte und Ähnliches werden nicht unterstützt. Dennoch kann es ein, dass die Fehlerkorrektur einfache Änderungen lösen kann. Größere QR-Code Versionen können laut Dokumentation unter Umständen funktionieren. Es ist also möglich, je nach Error Checking and Correction(ECC) Level 174 bis 395 Zeichen zu übertragen. Das reicht für rund 30 Wörter. Die für die Wordles Fertiggestellten es QR-Codes haben allesamt eine Version von 9 mit minimalen ECC. Damit ist es möglich, an die 20 Wörterproblemloss zu verschicken.

Da die QR-Codes auf Systemebene erkannt und getrackt werden, ist etwas Housekeeping von Nöten. Das QR-Code-Modul liefert immer alle erkannten QR-Codes, unabhängig davon, wann sie gesichtet worden sind. Es ist daher nötig die Erkennungszeiten zu überprüfen und zu filtern, ob ein QR-Code relevant ist oder nicht. Insbesondere da das Herausnehmen von QR-Code aus der Umgebung zu Tracking Artefakten führen kann. Es ist daher wichtig, einen Sanitycheck auf allen QR-Codes die gelistet werden durchzuführen.

Obwohl QrCodes Native auf der Hololens zur Verfügung stehen ist die **Microsoft.MixedReality.QR** API kein fester Bestandteil des Mixed Reality Toolkits. Es muss zusätzlich als NuGet hinzugefügt werden. Obwohl das Trackign Mittels der TrackignKamera funktionieren, Muss im Unity Player zusätzlich die **Webcam** Funktionalität freigeschaltet werden. Ist die Api engebunden Kann das Qr Tracking Mittels dem **Microsoft.MixedReality.QR.QRCodeWatcher** geschehen. Dieser stellt eine **Start** und eine **Stop** Funktion zur Verfügung. Um nun QrCode Änderungen zu verarbeiten liefert der **QRCodeWatcher** drei EventHandler. **Added** um neu erkannte QrCodes zu verkünden. Wenn sich ein QRCode in der reellen Welt bewegt wird **Updatet** ausgelöst. Und mit **Removed** lässt sich festtellen wenn ein QrCode nicht länger erkannt wird.

Zu jedem Event liefert der **QRCodeWatcher** einen **Microsoft.MixedReality.QR.QrCode** dieser enthält eine Reihe von wichtigen Informationen zum QR-Code. Dazu zählt der Inhalt in **QR-Code.Data**, die physikalische Größe in der realen Welt in **QRCode.PhysicalSideLength**, die Erkennungzeit mit **LastDetectedTime** und eine Guid für den Spatial Graoh Knoten. Mit diesem ist es möglich, den QR-Code in der Szene zu verfolgen.

5.6 Bildverarbeitung

Um Formen in Bildern zu erkennen, bedarf es Bildverarbeitung Algorithmen. OpenCV bietet hierfür eine Bibliothek mit den benötigten Algorithmen. Der Ablauf für AR Wordle ist wie folgt:

```
Schneide aus dem Eingabebild die Form aus
Skaliere und rotiere das Bild auf die Zielgroesze
Erweiter die Form um einen weißen Rahmen
Konvertiere das Bild in Graustufen
Erzeuge ein Binaerbild mittels Treshhold
Analysiere das Bild auf verbundene Komponenten
Führe für jede Komponente eine Konturanalyse durch
Berechne und Filter die Konturen und deren Gebiete
Berechne die Distanzfelder
Glaette die Distanzfelder und bestimme Ihre Extrempunkte/Zentren
```

Ein visueller Ablauf ist in der Abbildung 5.5 dargestellt.

Ein Bild von der Hololens-Webcam hat eine Standarte Auflösung von 3904 auf 2196 und bildet einen sehr großen Raum ab. Da sich die vom User gewünschte Form sich zentral vor der Kamera aufhält wird, muss die Aufnahme entsprechend zugeschnitten werden. Da die Hololens sich auf den Kopf des Nutzers befindet, kann diese in Schräglage geraten. Um dieser entgegenzuwirken wird das Bild zuerst durch eine affine Transformation waagrecht zur Welt rotiert. Danach wird es auf ein kleineres Zielcanvas der Wordle skaliert. Dies geschieht um Speicher und Rechenzeit einzusparen. Größere Krümmungen bleiben bei der Skalierung erhalten und die Formen ändern sich nicht zu sehr. Die Aufnahme wird dann um einen weißen Rand erweitert, dies geschieht, um mögliche Fehler beider Konturerkennnung zu vermeiden. Um die Formen zu erhalten, wird das Bild zuerst in Graustufen geändert. Und dann mittel Treshhold in ein Binärbild verarbeitet. Um Belichtungsverhältnissen und nicht schwarzen Farben entgegenzuwirken, lässt sich der Treshhold Parameter manuell zur Laufzeit vom Nutzer anpassen.

(a) Original **(b)** Graustufe

(c) Threshold **(d)** Distanzfeld

Abbildung 5.5: Schritte der Bildverarbeitung

Das Bild besteht nun aus Schwarz für die Freiräume und weiß für die Formen. Um die Regionen zu erkennen, erhält jede einzelne durch eine Komponentenanalyse ein Label. Hierfür wird der OpenCV Standard 4-Bit Konnektivitätsalgorithmus verwendet. Hierbei handelt es sich um das Spaghetti-Labeling von Federico Bolelli, Stefano Allegretti, und Costantino Grana [BAG21].

Nachdem alle Regionen im Bild erkannt worden sind, kann für jede eine Konturanalyse ausgeführt werden. Die Konturauswertung von OpenCV verwendet den Algorithmus von Satoshi Suzuki und Keiichi [Sb85]. Dieser erkennt Konturen, die den Bildrand berühren nicht. Durch den zusätzlichen Whitespace, der am Rand hinzugefügt wurde, werden alle Formen die auf den Bild liegen sauber erfasst.

Um die erkannten Formen besser auswerten zu können, wird die Konturfläche mit dem Satz von Green berechnet. Dies erlaubt es kleine unliebsame Artefakte zu filtern und die Wörter später gezielter zu verteilen.

Nun können die bestimmten Regionen in Distanzfelder umgewandelt werden. Hierfür verwendet OpenCV den Algorithmus von Gunilla Borgefors [Bor86]. Da kein akkurates Ergebnis nötig ist, wird der Algorithmus mit der euklidischen Distanz und einer 3×3 Matrix ausgeführt. Dies ist zu gleich die performanteste Distanzfeldberechnung. Um grobe Kanten zu beheben wird das erzeugt Distanzfeld nochmals nachträglich geglättet.

Zuletzt werden die Distanzfelder auf Ihre Extrempunkte untersucht. Es kann vorkommen, dass eine Region mehrer Extrempunkten besitzt. Liegen diese nah beieinander, wird der mit der größten Distanz zur Kontur gewählt. Mit allen Distanzfeldern und Regionen berechnet können nun die Wordles verteilt werden.

5.7 Logging und Debugging

Das Loggingskript hat die Aufgabe, Änderungen und Meldungen mit zuschreiben und diese permanent im Speicher zu hinterlegen. Dafür bietet sie die Möglichkeit Strings und Exceptions Explizit zu Logen. Zusätzlich registriert sie sich bei **Application.LogCallback** als Callback. Dadurch bekommt sie alle Debug Nachrichten aus dem gesamten Programm mit und kann diese Protokollieren. Die Logging Komponente speichert zudem alle 30 Sekunden alle Änderungen als TXT-Datei im Applikationsordner ab. Dieser kann über das Device Portal (Abschnitt 5.7.2) geöffnet werden. Ursprünglich war geplant, nur zum Applikationsende zu speichern, jedoch stellte sich heraus, **MonoBehaviour.OnDestroy()** nicht immer richtig ausgeführt wird. Eine Möglichkeit hierfür kann sein, dass die HoloLens das Programm nicht wie von Unity erwartet, beendet. Dies führt dann dazu, das die Destroyfunktion auf allen SzenenObjekten von Unity nicht ausgeführt wird

5.7.1 Hololens Emulator

Um Programme für die Hololens zu testen, bietet Microsoft den Hololens-Emulator an. Dieser kann von Microsofts Website heruntergeladen werden. Die Systemvoraussetzungen sind:

1. 4-bit Windows 10 Pro, Enterprise, or
2. (BIOS) Hardware-assisted virtualization
3. (BIOS) Second Level Address Translation (SLAT)
4. (BIOS) Hardware-based Data Execution Prevention (DEP)

WDDM 2.5 graphics driver (HoloLens 2 Emulator) GPU requirements

5. WDDM 2.5 graphics driver (HoloLens 2 Emulator)

Zusätzlich muss **Hyper-V** freigeschaltet sein. Dies benötigt eine CPU mit VM-Monitor-Mode-Extension. Bei Intel als VT-c bekannt. Für AMD Plattformen ist dies der SVM Modus auch geläufig als AMD-V. Diese müssen eventuell noch im BIOS eingeschaltet werden. Der Emulator unterstütze für Hololens2 Builds auf die Plattform **x86** und **x64**. Nach dem das Unity Build für VisualStudo fertiggestellt wird. Erkennt dies den Emulator fürs Deployment. Jedoch stellte sich das unter Umständen als nicht ganz reibungslos heraus. Wenn VisualStudio direkt auf den Emulator deployet startet dieses eine neue Instanz. Da das Booten unter Umständen sehr lange dauern kann, kommt es hin und wieder zu Deploymentfehler. Es war dahingehend zielführender das Programm über das DevicePortal auf den Hololens-Emulator zu Deployment. Und denselben Emulator wieder zu gebrauchen. Insgesamt hat es sich mehr bewährt zum Debuggen den Playmode von Unity zu verwenden und für Direktes testen auf die echte Hololens zu deployen.

5.7.2 Device Portal

Um auf die Hololens zuzugreifen zu können, bedarf es des Hololens-DevicePortal. Ist eine Hololens mit einem Rechner gekoppelt, kann über den Browser ein HTTP-Server erreicht werden. Dieser Besitz eine IPv4-Adresse, die dem Entwicklermenü der Hololens entnommen werden kann. Im DevicePortal können wichtige Informationen zum Status der Hololens eingesehen werden. Dazu zählen eine Übersicht der Prozesse, die Temperatur, der Batteriestand und eine Live-Preview der Kameras. Zudem kann über das DevicePortal auf die Festplatte der Hololens zugegriffen werden. Damit ist es möglich Dateien die von Programmen gespeichert wurden problemlos hin und her zu kopieren.

5.7.3 Holographic Remoting

Um Hologramme vom Computer aus auf der Hololens anzeigen zu lassen bietet Microsoft Holographic Remoting. Damit können über den Webserver der Hololens die Hologramme, die Unity berechnet, direkt auf dieser dargestellt werden. Der Vorteil ist die live Preview und Editierbarkeit in Unity und dessen Editor. Auch ermöglicht dies teurere Berechnungen auf leistungsfähigere Hardware auszuführen. Nachteilhaft hierbei ist, das einige Schnittstellen der Hololens nicht direkt zugreifbar sind und das ein Leistungsprofil der Hololens wegfällt.

5.7.4 Debugging von OpenCV

Generell ist mit OpenCV und Unity zu arbeiten, zusätzliche Sorgsamkeit zu empfehlen. Einerseits sind viele Fehler innerhalb OpenCV auf DLL Memory Ebene und können daher nur sehr wage Aussagen geben, warum es zum Fehler kam. Die mit am häufigsten auftretenden Memoryfehler sind Indexe, die out of bounds sind. Daher empfiehlt es sich, Höhen und Breiten vor einem Aufruf nochmals zu überprüfen. Einige OpenCV Funktionen erwarten schon eine entsprechend große Mat mit passenden Typ zum Schreiben, wohingegen andere diese zur Laufzeit anpassen. Auch unterscheiden sich die Farbkanäle von Unities Texture2D und OpenCV. United Texture 2D speichert Standardmäßig seine Bildinformationen als Color ab. Diese sind über **.r,.g,.b,.a** floats zwischen 0 und 1 definiert. Eine Umformatierung auf 0-255 Bytes kann entweder manuell oder durch einen impliziten cast auf Color32 vorgenommen werden. Obwohl OpenCVforUnitys Mat jeglichen Wert

abspeichern kann, ist es für Bilder sinnvoll, diese Pixel Informationen als Byte Array zu hinterlegen und abzufragen. Es ist aber hierbei zu beachten, dass die Reihenfolge der Farbkannäle nicht RGBA ist. OpenCVForUnits Farbkannäle sind BGRA. Mit Index 0 für Blau, 1 für Grün, Index 2 für Rot und Index 3 für den Alpha-Kanal. So zumindest sind sonst Rot und Blau vertauscht, wenn die Mat als .png abgespeichert wird.

Beim debuggen von Unity mithilfe von Angehängten VisualStuido kann es unter Umständen dazu kommen, dass Unity komplett abstürzt. Dies geschieht hin und wieder beim einzelnen Durchschreiten von Funktionen die auf OpenCV zugreifen. Es ist daher ratsam, vor dem Debugging die Szene zu speichern und falls es zu diesem Problem häufiger kommt, nur mit Breakpoints durch den Code zu springen und gegebenenfalls mehrere Zeilen auf einmal zu nehmen. Eine weitere Herausforderung beim Debuggen ist es, das VisualStudio keine Vorschau von Mats in OpenCV bietet. Diese sind nur Memory Adressen und die Vorschau beinhaltet nur die Maße und den Typ. Um also ein besseres Bild zu bekommen, was gerade geschieht, empfiehlt sich die Mat bei gewünschten Punkten mittels abzuspeichern und getrennt zu analysieren. Das ist in OpenCVFOrUnity Mittels **ImgcodecsModule.Imgcodecs.imwrite(filePath.png,imgMat)** und in OpenCVShapr mit **imgMat.SaveImage(filePath.png)** möglich.

6 Performance mit Unity auf der Hololens2

Obwohl die HoloLens selbst eine sehr leistungsstarke Plattform ist, so hat sie dennoch ihre leistungsschwächer, insbesondere wenn es um Leistungpeaks geht. Eine leistungshungrige Berechnung lässt sich nicht in einem nicht spürbaren Rahmen in Unity direkt ohne Weiteres umsetzten. Auch besitzt Unity einiges an Standard API-Funktionien, die unbedacht auf leistungsschwächeren Plattformen sehr teuer werden können. Wie bereits in 4.5 erwähnt, zählen Wordles eher zu den leistungshunrigen Algorithmen. Insbsondere wenn Bildverarbeitung mit dazu kommt. Dieses Kapitel beschreibt eine Menge dos and donts die es mit Unity zu beachten gilt und schlägt eine umgesetzte Ansätze vor.

6.1 Coroutinen, Jobs und Threads

In Unity läuft alles standartmäßig in einem Hauptthread. Dieser loopt endlos und ruft pro Frame Update auf allen Objekten und allen Komponenten in diesen auf. Dadurch ist die Framerate und ein wichtiger Bestandteil der Userexperience an die Performance aller Methoden gekoppelt, die nicht explizit anderweitig operieren. Es empfiehlt sich daher, teure Berechnungen vom Hauptthread und damit der Framerate zu entkoppeln und nach getaner Arbeit zu joinen. Hierfür gibt es eine Handvoll an Möglichkeiten.

6.1.1 Coroutine

Eine Coroutine besteht aus einen Unitys eigenen IEnumerator. Dieser yielded ent weder null, oder eine Vielzahl von anderen Wartebefehlen, die Unity zur Verfügung stellt. Diese Wartebefehle erlauben es Arbeit an einem späteren Zeitpunkt fortzuführen. In Listing ist ein Beispiel dafür, eine Berechnung nach 20 Sekunden fortzusetzen. Sobald Unity den yield statement erhält, macht es sich einen vermerkt und führt diese Methode entsprechend später fort. Coroutinen selbst laufen aber dennoch auf dem Hauptthread. Sie eignen sich daher besser für kleine zusätzliche Arbeitsschritte, die geplant verteilt werden können. Abseits der Aufteilung bieten sie auch noch einen anderen Vorteil. Ein Beispiel ist es mit dem mit dem Befehl **WaitForEndOfFrame()** bis zum Ende aller Berechnungen eines Durchlaufes zu warten Alle Wordles besitzen zum Interagieren eine **Interactable**-Komponente. Diese interagiert mit Unitys Physikengine um Bewegungen zu berechnen. Da die Wordles beim instanziieren nicht direkt ihre richtige Größe haben, kommt es zu Artefakten. Diese werden noch im Frame selbst gelöst, da aber die Physikengine nicht auf ein vollständiges Update wartet, treten ungewünschte Interaktionen auf. Alle Wordles besitzen daher beim Instanziieren eine Coroutine, die bis zum FramEnde wartet, um dann erst die *PhysikCollider* frei zu geben. Dadurch wird vermieden, dass bevor der erste Frame gezeichnet wird, 2 nahe liegende Objekte nicht ineinander liegen und eine Kollision erzeugen. Das ist bei größeren Objekten kein

Problem, da sie sich einander wegschieben. Das Verhältnis der Initialen Bounds und der eigentlichen Physicollider sind leider so groß, das die Collider komplett ineinander liegen können. Dies führt zu einer Physik Berechnung, die beide *Collider* massiv voneinander wegschießt.

6.1.2 Jobs

Unity bietet von Haus aus eine Komponente zum multithreaden, das C Job System. Dies ist ein System, das Arbeit auf WorkerThreads auslagert. Es erzeugt keine neuen Threads sondern unterteil die Aufgaben gennant Jobs auf diese auf und managet zeitgleich Racingconditions und Abhängigkeiten. Damit ist es möglich eine hohe Parallelisierung mit wenigen Eigenaufwand zu erreichen. Dies kommt aber zugleich mit kosten. Um jegliche Racingconditions zu vermeiden, übergibt Unity keine Referenzen an die Workerthreads. Daten werden in Structs mit primitiven kopiert und an die Worker gegeben. Diese Berechnen dann auf ihrer eigenen Kopie das Ergebnis. Um dem etwas entgegenzukommen bietet Unity **NativeContainer** an. Das sind sicher C Wrapper für nativen Speicher. Es erlaubt einen Job Daten mit dem Mainthread zu teilen, anstelle eine Kopie zu erstellen. Unity bietet auch eine kleine Anzahl von NativesContainer an, die einfache Datenstrukturen verwirklichen. Dazu zählen **NativeArray** und **NativeList**. **NativeHashMap** und **NativeMultiHashMap** und **NativeQueue**. Je nach Anwendungsfall lässt sich auch der Allokationstyp der **NativeContainer** bestimmen. DIese sind **Allocator.Temp**(1 Frame), **Allocator.TempJob**(4 Frames) und **Allocator.Persistent**(bis zum Applikation ende).

Jobs eigenen sich also sehr gut, um für kurze Zeitabschnitte Arbeit auf mehrere Threads auszulagern. Sie sind jedoch eher ungeeignet wenn eine Task für einen langen Zeitraum berechnet werden muss. Insbesondere wenn die benötigten Daten nicht problemlos in die dafür Benötigten Container passen. Dafür können sie sehr performant sein. Es ist möglich den BurstCompiler auf Jobs an zu wenden, somit laufen diese in sehr effizienten nativen Code.

6.1.3 Threads

Eine gängige Alternative zu multithreaden ist die Thread-Klasse von C. Diese erlaubt es dem Anwender Arbeit komplett auf andere Threads auszulegen. Hierfür kann ein eigner Thread gestartet oder sich dem Threadpool bedient werden. Die Standardthreads besitzen von sich aus keine Threadsicherheit und um Racingconditions muss sich selbst gekümmert werden. Anwendungsfall stellt dies aber keine Herausforderung da. Da wir keine natürliche Racingcondition haben, sondern nur Arbeit auslagern wollen. Alle Threads könnten nach dem “Run and forget”-Prinzip laufen, jedoch stellt sich das nicht als trivial heraus. Unity erlaubt es abseits dem Hauptthread keinen anderen mit der Unity-API zu interagieren. Das bedeutet das Manipulieren, oder nur das schlichte Auslesen von Gameobjekten ist nicht möglich. Auch sind andere Objekte wie Texture2D tabu. Wir müssen also alle Arbeit noch im Hauptthread in nicht Unity-Objekte kopieren und bis zur Bedingung des sekundär Threads warten, um das Ergebnis in der Szene selbst darstellen zu können. Da es nur einen Hauptthread in Unity gibt, ist es nicht möglich den Arbeitsthread mit einem **Thread.Join()** zu joinen. Es ist daher notwendig, den Thread regelmäßig zu fragen, ob dieser seine Arbeit abgeschlossen hat. Ein naiver Ansatz ist es in der Updateschleife diesen immer wieder pro Frame zu fragen, ob er denn fertig sei. Da Updates von Unity möglichst häufigst wie möglich ausgeführt werden und der dabei entstehende Kontextwechsel wie in 6.2 beschrieben teuer ist. Ist es eine Überlegung wert, den Thread nur über Coroutinen aufzurufen oder Fixedupdate zu verwenden.

Da wir nicht eine direkte Umsetzung so schnell wie möglich brauchen, bietet sich Fixedupdate sehr gut an. Jedoch zeigt sich in der Praxis, dass ein Update mit einem einfachen “if” in einem Objekt nicht wirklich mehr Leistung kostet.

6.1.4 Wann wie was Parallelisieren

Grundsätzlich gilt es abzuwägen wie welche Arbeit parallelisiert oder auch gestreckt wird. Da Wordles nicht eine Liveaktualisierung benötigen, erlaubt dies die Iterationen auf mehrere Frames aufzuteilen. Hierfür ist eine gute Laufzeitabschätzung der Iterrartionsschritte notwendig. Diese ist aber von vielen Faktoren, wie generelle Hololensauslastung, Anzahl von Wörtern, Anzahl von möglichen Überschneidungen und weiterem abhängig. Deshalb wurde ein naiver Ansatz gewählt. Nach jeder Iterration wird die verstrichene Zeit **Time.deltaTime** überprüft. Übersteigt diese die Zielframerate wird die Iteration pausiert und in einer Coroutine für den nächsten Frame ausgelagert.

Wird ein Wort neu platziert, müssen häufig mehrerer Wörter zeitgleich weichen. Um dies performant zu bewerkstelligen würden sich Jobs anbieten. Jedoch gestaltet sich die Umsetzung auf **NativeContainern** als eine Hürde. Informationen fortlaufend zu packen und zu entpacken ist selbst kostspielig. Zudem muss das Ergebnis immer aufs neue abgeholt werden. Dies sorgt für ein hohen Bedarf an Managements. Es gestaltete sich als zielführender die Algorithmen nicht auf Worker auszulagern.

Die Verarbeitung von Bildern mit OpenCV benötigt ebenfalls viel Leistung. Obwohl OpenCV stark um performant bemüht ist, kommt es auf der Hololens zu einem klaren Bottleneck. Glücklicherweise benötigt die Berechnung in OpenCV keine Unity-APIs. Dadurch kann diese komplett auf einen eigenen Thread im Threadpool ausgelagert werden. Der Hauptthread überprüft dann solange diesen, bis er das Ergebnis bereitstellt. Sollte dies der Fall sein, kann dieser dann das Ergebnis an die Objekte mit den Unity-APIs übergeben.

6.2 Update und FixedUpdate

Update() ist eine Standard Unity API Funktion. Sie wird beim Erzeugen eines jeden neuen Skriptes hinzugefügt. Ihr Ziel ist es Operationen zu ermöglichen, die jeden Frame passieren sollen. Änderungen hier sind sofort für den Nutzer sichtbar, sobald das nächste Bild gezeichnet wird. Obwohl sehr nützlich sind sie auch äußerst performancehungrig. **Update()** ist außerdem eine Unity Message. Sie wird zu jedem Frame sobald die Engine bereit dafür ist ausgelöst. Hierfür geht Unity durch alle aktiven Objekte in der Szene und überprüft für jedes Objekt, ob es eine Skriptkomponente hat und ob diese **Update()** implementieren. Findet Unity eine Updatefunktion. Wechselt es in den Kontext des Skripts und ruft diese auf. Dies geschieht selbst wenn die Funktion leer ist. Bei hundert Wörtern mit je drei Skripten und 30 Frames die Sekunden sind das 9000 Aufrufe, ohne das sich etwas ändert. Es ist daher wichtig zu beachten, dass leere Update-Funktionen im Skript komplett entfernt werden, und wenn vorhanden nicht auf zu vielen Objekten zeitgleich existieren. Es ist effektiver, die Logik in eine einzige Updatefunktion zu packen, die alles Änderungen verwaltet.

Eine alternative zu **Update** ist **FixedUpdate**. Ungleich **Update** wird **FixedUpdate** nicht jeden Frame aufgerufen. Der Call passiert jedes mal wenn die PhysikEngine eine Berechnung durchführt. Der Vorteil daran ist, das Unity darum bemüht ist **FixedUpdate** so regelmäßig wie möglich aufzurufen. In **FixedUpdate** werden am besten Änderungen von Rigidbodies gehandhabt, so das diese direkt verarbeitet werden können. Andere Logik in **FixedUpdate** zu legen kann Vor- und Nachteilhaft sein. Der Vorteil liegt darin, das man über die PlayerSettings oder Unities **Time** Klasse die Intervalldauer manuell abändern kann. Dadurch ist es möglich Codezyklen zu stauchen und zu strecken. Es ist aber weiterhin zu beachten, dass **FixedUpdate** dennoch auf dem Mainthread läuft und **Update** und andere Funktionen warten müssen, bis die laufenden Berechnungen abgeschloßen sind. Auch kann es dazu kommen, dass sich bei einer sehr niedrigen Framerate und kleiner Intervallzeit **FixedUpdate** mehrmals pro Frame Aufgerufen wird. Es ist außerdem zu bedenken, dass eine Änderung, neben der Physik alle **FixedUpdate** Funktionen gleichzeitig betrifft.

Wartet eine Funktion auf eine Statusänderung im Unity-Zyklus, kann es je nach Situation besser sein, dies in **FixedUpdate** oder in Update zu legen. Unabhängig der Wahl, ist es sinnvoll die Anzahl von **Update** und **FixedUpdate**-Funktionen gering zu halten. Beides sind UnityMessages und forcieren einen Kontextwechsel, der bei einer häufigen Ausführung sehr teuer werden kann, insbesondere wenn die Methoden selbst leer sind.

6.3 GetComponents und Pointer

Eine gängige und häufig verwendete Unity API ist die Funktion **GameObject.GetComponent**. Damit kann auf Skripte und Komponenten wie Mesh, Text und *Collider* zugegriffen werden. Obwohl die Zielkomponente schon zur Kompilierzeit feststeht und sich nicht mehr ändert, handhabt Unity jeden **GetComponent** Aufruf als Suche. Das heißt, Unity durchsucht alle am GameObject angebrachten Komponenten, bis es den ersten Treffer findet. Dies wird bei Objekten mit vielen Komponenten schnell teuer. Insbesondere wenn diese Suche auf vielen Objekten gleichzeitig läuft. Es ist daher ratsam, die Komponenten einmal am Anfang in **Start()** aufzurufen und dann intern im Skript als Referenz zu speichern und zu managen. Dies kostet zwar zusätzlichen Speicherplatz, der aber sehr vernachlässigbar im Vergleich zur Performance ist.

Selbiges gilt auch für die Suche von GameObjekten. Es ist zwar angenehm GameObjekte mit **Object.Find()** zu referenzieren, aber unglaublich teuer. Hier durchsucht Unity die Gesamte Szene mit allen Objekten und Ihren Kindern, bis es einen Treffer landet. Fixe GameObjekte die immer anwesend sind, sollten daher in Ihrem Skript oder in einem Masterskript als Referenz gespeichert werden, so das der Aufruf in *O(1)* passiert. Es ist zusätzlich zu erwähnen, das **Camera.main** kein direkter Pointer ist. Hinter diesem Aufruf liegt ein **FindGameObjectsWithTag()**. Also eine Suche durch alle Objekte in der Szene.

6.4 Sonstiges

Neben dem Multithreading gilt es in Unity einige weiter Stolpersteinen zu beachten. Diese können massiv an Leistung einnehmen und sind häufig die erste Wahl, wenn man sich nicht tiefer gehend mit Unity und seinen Eigenheiten auseinandergesetzt hat. Es handelt sich hierbei überwiegend um

Empfehlungen. All die angesprochenen Themen sind bei minimaler Anwendung nicht bemerkbar. Jedoch werden sie bei nicht Beachtung und einer hohen komplexen Laufzeit, wie sie Wordles gerne haben, zu starken Botelnecks, wenn nicht sogar zu den größten Leistungsfresser.

6.4.1 Object Pooling

Da Unity nur auf seinem Mainthread Interaktionen mit seiner API erlaubt, ist es wichtig, die Anzahl von Initialisierungen zur Laufzeit klein zu halten. Ein Objekt zu erzeugen, mit zusätzlichen Komponenten zu versehen und zum Start zu konfigurieren, ist sehr teuer. Insbesondere wenn die Anzahl von Objekten die zeitgleich generiert werden hoch ist. Um dem entgegenzuwirken, empfiehlt sich Objekt-Pooling. Es ist die Idee, Objekte am Ende Ihrer Lebenszeit nicht zu zerstören, sondern zu deaktivieren und später wieder zu verwenden. Damit reduziert man nicht nur die Zeit der Initialisierung, sondern auch den Aufwand, der durch das Zerstören anfällt. Hinzukommt das eine Grabage-Kollektion seltener passiert und Unity weniger Zeit mit Memorymanagement verbringt. Eine einfache Umsetzung ist es die GameObjects in einer Liste zu führen und diese nach lebenden, toten und inaktiven zu unterscheiden. Wobei hierbei die toten und inaktive GameObjekte sind, die eine bereits initialisierte Instanz für ein neues GameObjekt darstellen. Um dies zu Verwirklichen bietet Unity die **setActive(bool val)** Funktion an GameObjekten an. Ist ein GameObjekt inaktiv, wird es weder gezeichnet, noch können die hinzugefügten Skripte und Komponenten agieren oder werden von Unity beachtet.

6.4.2 LINQ

Language-Integrated Query (LINQ) ist eine .NET Framework Komponente, die es erlaubt Queries sauber und einfach im Code zu schreiben. Leider kommt LINQ mit einem Overhead, der insbesondere bei vielen Queries zu einem erhöhten Speicher und Rechenbedarf führt. Es ist daher besser Queries traditionell als Schleifen zu schreiben und LINQ weitestgehend zu meiden. Auch ermöglichen selbst geschriebene Lösungen zusätzliche Performance, wenn weitere oder schärfere Abbruchbedingungen gesetzt werden können.

6.4.3 Raycast

Um das exakte Verhältnis von User zum Umgebungsmesh festzustellen, ist häufig ein Raycast nötig. Raycast sind von sich aus sehr teuer, insbesondere da die Umgebung mit einem **MeshCollider** versehen ist. **MeshCollider** sind bei weitem am teuersten auszuwerten. Um die Rechenzeit minimal zuhalten, ist es sinnvoll möglichst wenige Raycast für die Umgebung zu verwenden und Ergebnisse zwischen zu speichern. Unitys Raycast benötigt zusätzlich eine **LayerMaske** und eine maximale Distanz. Beides sind Parameter die dazu genutzt werden können Performant zu erhalten. Da die AR-Welt in Unity einen Meter in einer Einheit misst, ist eine kleine maximale Distanz sehr sinnvoll. Befindet sich die Anwendung in einem geschlossenen Raum sind ein Paar Meter schon mehr als Ausreichend. Standardmäßig befindet sich das SpatialAwarness Mesh auf dem **Layer 31** (Spatial Awarness). Die Bit Maske hierfür ist 1«31. Damit überprüft der Ray explizit das SpatialMesh und das Ergebnis muss nicht noch mal extra gefiltert werden.

6.5 SendMesage BrodcastMessage und Delegates und Events

Für die asynchrone Kommunikation zwischen zwei Objekten bieten und Unity und C verschiedene APIs. Hierbei gilt Gleiches wie bei GetComponent und Pointers. Unitys eigene Eventsysteme sind sehr ineffizient und sind eine Suche durch alle Objekte einer Szene und deren Komponenten. Mit SendMessage und BrodcastMessage kann ein Methodenaufruf auf allen Skripten des Zielobjekts ausgeführt werden. Ein Beispiel für eine BroadcatsMessage isz die Update()-Funktion. Diese wird von der Unity-Engine selbst ausgelöst und hat, wie in 6.2 beschrieben, einige Nachteile die es zu beachten gilt. Zur direkteren Kommunikation gibt es UnityEvent und das C native Delegate Eventsystem. Der Leistungunterschied hierbei ist drastisch. Delegates laufen um ein vielfaches schneller als Unity-Events. Es ist hierbei jedoch anzumerken, dass UnityEvents wesentlich freundlicher im Design sind. So ist es möglich, diese im Editor zu integrieren, auszuführen und zu testen. Das ermöglicht es, verhalten im Design festzulegen und nicht erst zur Laufzeit.

7 Evaluation/Nutzerstudie

Im Rahmen zur Auswertung der Wordles in AR wurde eine Nutzerstudie durchgeführt. Das Ziel dieser Pilotstudie war es, die AR-Umsetzung zu testen und zu untersuchen, wie gut sich Nutzer Wordles in AR erstellen können. Den Probanden wurde eine Vielzahl von rudimentären Aufgaben bzgl. Wordles gestellt und sie sollten diese umsetzten, bis sie mit dem Ergebnis zufrieden waren. Danach sollten Sie in einem Fragebogen die Benutzbarkeit bewerten.

7.1 Aufbau und Ablauf

Als erstes haben alle Probanden eine kurze Einführung in alle Features erhalten. Die erste Aufgabe bestand da drin ein Wort zu Erstellen, dieses zu bewegen, zu skalieren und die Farbe zu ändern. Danach wurde ihnen eine bereits generierte Word-Cloud präsentiert. Diese diente dazu, ihnen das Verhalten von Wörtern untereinander beizubringen, sowie die Modifikationen von Mergen und Konkatenieren. Nach erfolgreicher Einführung wurden die Probanden gebeten, einen QR-Code zu scannen. (Siehe Abbildung 5.4). Dieser enthält eine Liste von Wörtern, die zu einem Bild im Raum gehörten. Die Probanden sollten diese dann nach eigenen Vorlieben modifizieren, während sie nebenher das Bild zu einsehen konnten. Zeitgleich wurde ihnen gezeigt, wie sie das Canvas skalieren und neu im Raum positionieren können, und sie wurden aufgefordert, jenes und sich selbst so zu platzieren, dass es für sie am angenehmsten ist. Dabei wurden sie beobachtet und es wurden Notizen gemacht, welche Änderungen wie sehr beansprucht worden sind. Als zweite Aufgabe sollten die Probanden einen anderen QR-Code einlesen. Dieser lies die Applikation ein bestimmtes Prefab aus dem Speicher laden und darstellen. Dieses enthielt zwei Listen von Vokabeln. Die Probanden bekamen damit die Aufgabe, diese richtig zu zuordnen. Die Herausforderung hierbei war, dass das Wordle nicht mehr vertikal, sondern horizontal präsentiert worden ist. Als letzte praktische Aufgabe sollten die Probanden eine Form auf einem Bild einscannen und diese für ein Wordle verwenden. Zum Abschluss der Evaluation wurden die Probanden gebeten einen Fragebogen mit Fragen bezüglich ihrer Erfahrung zu beantworten.

7.2 Ergebnisse

Aufgrund der aktuellen Corona-Situation konnte die Pilotstudie nur in einem sehr kleinen Rahmen durchgeführt werden. Insgesamt wurde die Studie mit vier Probanden durchgeführt. Dennoch waren die Resultate bei der Studie aussagekräftig und lassen ein klares Bild erkennen. Trotzdem ist eine durchführung mit mehr Probanten zukünftig sinnvoll. Die Pilotstudie lief reibungslos ab und Probleme auftraten waren systematischer Natur und lieferten interessante Erkenntnisse zu Faktoren die zuerst gar nicht in Betracht gezogen worden sind.

(a) kleine Wörter (7x3cm) **(b)** große Wörter (12x6cm)

Abbildung 7.1: Verhältnis Hand zu Wortgröße. Die gesamte Darstellung entspricht dem Sichtfeld bei rund 60cm Abstand

7.2.1 Positionierung des Wordles

In der Pilotstudie wurden Wordles an einer horizontalen und auf einer vertikalen Fläche verglichen. Es zeigte sich, dass die Probanden das Arbeiten auf der Vertikalen um ein Vielfaches bevorzugten. Das Arbeiten auf der Horizontalen kommt mit einigen Herausforderungen und Unannehmlichkeiten. Es ist für den Benutzer nicht angenehm, auf langer Dauer genickt Richtung Boden zuschauen, dafür belastet die Hololens den Nacken zu sehr. Auch zeigte sich eine Herausforderung der Interagierbarkeit. Die beste Distanz zum Arbeiten ist etwa ein halbe bis eineinhalb Armlängen. Um dies auf der Horizontalen zu bewerkstelligen müssen sich die Wordles auf einer dementsprechend angenehmen Höhe befinden. Probanden viel es sehr schwer, auf einem Stehtisch von einem Meter Höhe zu arbeiten, auf einem Beistelltisch von nur einem halben Meter hatten sie weniger Probleme.

7.2.2 Größe

Ein auf dem Bildschirm erzeugtes Wordle ist sehr kompakt und die Wörter selbst sind klein. Als Beispiel der Standart-CSS-Font im Browser ist 16px, dies entspricht einer Worthöhe von rund 0,423 Zentimetern als Ausdruck [w3o]. Ein maximal großes Wort mit xxx-large fällt mit 48px auf 1,27 Zentimeter. Es bedarf also keiner hohen Größe, um Wörter lesbar zu halten. Doch kommt der Aspekt der Interaktion durch Gesten hinzu, sieht es anders aus. In der Pilotstudie hat sich gezeigt, das Wörter mit einer Größe von 7 auf 3 Zentimetern als klein empfunden werden. Dies entspricht einer Fontgröße von knapp 113px. Grund dafür ist die mangelnde und schlechte Interagierbarkeit von kleinen Objekten. Kleine Wörter zu bewegen stellte sich nicht als Herausforderung da. Das Problem lag in der Skalierbarkeit. Kleine Objekte besitzen kleine Ecken und gestalten sich dementsprechend schwerer zu greifen. Ab einer Größe von 12 auf 6 Zentimetern aufwärts wurden die Wörter als groß empfunden. Ein Vergleich von den zwei Größen im Verhältnis einer Hand ist in der Abbildung 7.1 dargestellt. Die Empfindungen lassen sich entsprechend korrelieren. Darstellungen die kleiner als zwei Finger sind, gelten als schwer zu greifen. Wörter größer als die Hand selbst werden als riesig wahrgenommen. Die bevorzugten Maße der Wörter stellen sich im Vergleich zu anderen Hologrammen nicht als überraschend heraus. So besitzt das Hauptmenü der Windows Distribution auf der Hololens eine Höhe von 24 Zentimetern und eine Breite von 12. Auch sind Knöpfe und Selektionen mit einer Breite von maximal 9 Zentimetern und minimal 2 Zentimetern versehen. Es zeigte sich das es für Benutzer der Hololens am angenehmsten ist, mit Objekten in diesen Größen Spektrum zu interagieren.

7.2.3 Anzahl

Wordles sind für ihr kompaktes Layout bekannt. Dadurch sind sie in der Lage, möglichst viele Wörter zeitgleich darzustellen. Während der Pilotstudie zeigte sich, dass eine kleinere Anzahl von Wörtern bevorzugt wird. So werden weniger und dafür größere Wörter bevorzugt. Dies ist überwiegend der mangelnden Interagierbarkeit geschuldet. Dennoch zeigt sich das selbst bei einem benutzerfreundlichen Layout, bestehend aus kleinen Wörtern, nur 5 auf 14 Wörtern kompakt dargestellt werden Können. Diese Darstellung füllt das holografische Display komplett aus. Ein sinnvolles und übersichtliches Wordles entspricht dabei eher einer Anzahl von rund 25 Wörtern zeitgleich. Abbildung 7.1 zeigt die Größen im Verhältnis zum Darstellungsraum, wenn sich der Nutzer etwa eine Armlänge entfernt befindet. Es ist gut zu erkennen, dass mit kleinen Wörtern mehr dargestellt werden kann, aber auch die Interagierbarkeit leidet. Auch ist zu erkennen, dass es bei einer Zahl von 20 Wörtern aufwärts, viele aus dem Sichtfeld fallen und mehr nebensächlich sind. Würde sich der Nutzer weiter entfernen, fallen die Probleme der Interagierbarkeit noch stärker aus. Zwar würde die Übersichtlichkeit steigern, aber der Nutzen fällt zunehmend ab.

7.2.4 Erkennen von QR und Formen

Eine Aufgabe war es Worldes mit QR-Codes einzuscannen und diese in eine Form im Raum zu bringen. Das Scannen der QR-Code lief dabei ohne Probleme ab. Es gab nur minimale Probleme bei der Verwendung. Diese sind einer nicht ganz durchdachten Auswahlmöglichkeit geschuldet. Um einen QR-Code auszuwählen wurde dieser mit einem virtuellen Knopf versehen. Dieser lag unter Umständen zu nahe auf dem QR-Code, so das eine Bestätigung nicht richtig erkannt werden konnte. Das Erkennen der Form stellte sich hingegen als große Herausforderung da. Eine Erkennung dauerte im Schnitt 5 bis 7 Sekunden. War der Threshold nicht richtig gesetzt oder die Kamera nicht zentriert zieht sich die Formerkennung unter Umständen über eine Minute hin. Auch war die erkannte Form nicht zwangsläufig an der gewünschten Position. Es stellte sich zudem heraus, das es für den Anwender intuitiver war die Form mit den Augen anzuschauen, anstatt diese in der Markierung für die Erkennung zu halten.

7.2.5 Subjektives Nutzerfeedback

Abseits der Probleme mit der Benutzbarkeit wurden die Wordles in AR positiv aufgenommen. Die Probanden gaben an, viel Freude bei der Editierung von Wordles zu empfinden. Auch das sie sich im begrenzten Rahmen kreativ ausleben können. Es stellte sich heraus, dass eine freie Editierung von Objekten in AR mit Gesten sehr positiv aufgenommen wird. Ein Problem, das sich zeigte, war das Probanden häufig den Überblick verloren haben. Trotz einer moderaten Anzahl von 15 Wörtern viel auf, dass neu hinzugefügte Wörter oder Änderungen am besten hervorgehoben werden sollten. Da der Nutzer nicht immer alle Wörter auf einmal klar einsehen kann und somit Änderungen leicht übersieht.

7.2.6 Performance Analyse

Generell laufen Wordles auf der Hololens ohne Leistungsprobleme. Grundsätzlich sind 50 bis 60 FPS zu erwarten. Mit steigenden Worten fallen diese bis zu 50-40, bleiben aber weiterhin in einem absolut abnehmbaren und benutzerfreundlichen Rahmen. Eine schlechte Performance jedoch wurde bei der Generierung, der Neuplatzierung von vielen Wörtern und der Verarbeitung von Bildinformationen. Die dafür benötigten Leistungsschübe übersteigen mit der aktuellen Implementierung die Kapazitäten der Hololens um einen Grad, der sich negativ auf die Benutzererfahrung auswirkt.

7.3 Zusammenfassung

Die Nutzerstudie zeigte sich als sehr aufschlussreich. Es stellte sich heraus, dass eine geschickte Abwägung zwischen Sichtbarkeit und Interagierbarkeit gewählt werden muss. Kleine, kompakte Layouts sind gut lesbar und informativ. Sie leiden dafür stark unter einer positiven Interagierbarkeit. Dies ist nicht nur zuletzt der Genauigkeit des Handtracking zu schulden, sondern auch dem Verhältnis der Handgröße im Vergleich zum Wort. Generell ist es für den Anwender am angenehmsten, die Wörter direkt vor sich zu editieren. Dadurch verliert das Wordle aber den Bezug zur Umgebung und dient mehr als eine Mindmap. Generell stellte sich die Erkennung des Umfelds als eine Herausforderung da. Anders als bei einfachen, klar strukturierten PNGS haben Bilder der Umgebung eine Menge Fehlerquellen und Störungen, die es erst zu korrigieren gilt. Zusätzlich ist die Identifizierung von beliebigen Formen nicht immer einfach. Es ist dadurch nicht trivial, unliebsame Erkennungen zu filtern und auszusortieren. Sind Formen einmal vordefiniert, wie ein QR-Code und entsprechend optimiert, dann funktioniert die Detektion sehr gut und ohne große Zeitkosten.

8 Zusammenfassung und Ausblick

im Laufe der Arbeit wurden Wordles von der klassischen zweidimensionalen Anwendung in die dritte Dimension und eine andere Realität gebracht. Dabei wurden viele verschiedene Wordle Ansätze kombiniert, um ein vollständiges und rundes Erlebnis zu schaffen. Zudem wurde eine Implementierung auf einer leistungsschwächeren Plattform durchgeführt und evaluiert. Es zeigte sich, dass die Wordle Algorithmen sehr hohe Laufzeitkosten haben und es für eine gute Live-Interaktion viel zu beachten gilt. Zusätzlich stellte sich die Entwicklung auf der Hololens als nicht trivial heraus. Viele Entwürfe, die problemfrei auf einem klassischen Computer funktionieren, können auf der Hololens zu einer Hürde werden. Dies gilt insbesondere für die Integration weiter Bibliotheken und Tools. Auch zeigte sich, dass die reale Welt, ungleich einer perfekt vorgegeben und computergenerierten Vorlage viele Herausforderungen besitzt. Dazu zählen die vielen Imperfektionen, die beachtet und gefiltert werden müssen. Dies geschieht auf Lasten der Performance. Das sorgt dafür, dass bereits leistungsschwache Plattformen noch mehr eingeschränkt werden. In der Pilotstudie stellte sich heraus, dass die ursprüngliche Idee von Wordles als “Spielzeug” weiterhin hält und eine klare Charakteristik und Stärke von Wordles sind. Selbst mit großen Erschwernissen werden diese generell positiv aufgenommen und als spaßig empfunden. Grundsätzlich stellten sich zwei Hürden heraus. Um eine gute Editierbarkeit zu Erhalten Bedarf es einer gesunden Mindestgröße, jedoch kommt dies auf Kosten der Übersichtlichkeit und Darstellung. Wird ein Wordle nur dargestellt und keine Editieren zugelassen, entfällt ein wichtiger Aspekt der Wordles, welches sie unter anderem auszeichnet.

Ausblick

Wordles in Augmented Reality haben einen interessanten Einblick in AR Anwendungen gegeben. Sie stellen eine ganze Bandbreite an Herausforderungen, die es zu bewältigen gilt. Jede Einzelne an sich sind Probleme, die es für gute AR Anwendungen zukünftig relevant werden können. Sie fordern eine einfache Interagierbarkeit, ohne dabei unübersichtlich zu werden und haben hohe Performanceansprüche und benötigen dementsprechend ausgeklügelte Lösungen und Ansätze für schwächere Hardware. Wordles in AR aus einem Prototyp-Status zu bringen, kann sich zukünftig als aufschlussreich erweisen. Zudem bieten sie ein weites Anwendungsfeld. Grundsätzlich dienen sie eher als “Spielzeug”, jedoch können auf der Grundlage viele verschiedene Applikationen entworfen werden. Aus der Pilotstudie und Arbeit heraus hat sich ergeben, dass Wordles sich gut als Ergänzungsmaterial eignen. Sei es als interaktive und interessante Infotexte zu Ausstellungen oder als einfache Post-its und Memos am Arbeitsfeld. Es ist vorstellbar, dass in einer Zukunft mit etablierter Augmented Reality, Wordles und andere Arten von Tag-Clouds als gute Lückenfüller und Info-Happen dienen. Hierbei ist insbesondere die Erfahrung der Benutzer hervorzuheben. Wordles gelten auch in AR als eine grundlegend positive Benutzererfahrung.

Literaturverzeichnis

[AWE17] AWE. *Brett Jones (Lightform): Projection Mapping - Shared Augmented Reality for Out of Home*. Youtube. 2017. URL: https://www.youtube.com/watch?v=EGw2JOx11IM (zitiert auf S. 10).

[Azu97] R. T. Azuma. „A Survey of Augmented Reality“. In: *Presence: Teleoperators and Virtual Environments* 6.4 (Aug. 1997), S. 355–385. DOI: 10.1162/pres.1997.6.4.355. eprint: https://direct.mit.edu/pvar/article-pdf/6/4/355/1623026/pres.1997.6.4.355.pdf. URL: https://doi.org/10.1162/pres.1997.6.4.355 (zitiert auf S. 9).

[BAG21] F. Bolelli, S. Allegretti, C. Grana. „One DAG to Rule Them All“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), S. 1–1. DOI: 10.1109/TPAMI.2021.3055337 (zitiert auf S. 39).

[BK08] G. Bradski, A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008. ISBN: 9780596554040. URL: https://books.google.de/books?id=seAgiOfu2EIC (zitiert auf S. 31).

[Bor86] G. Borgefors. „Distance transformations in digital images“. In: *Computer Vision, Graphics, and Image Processing* 34.3 (1986), S. 344–371. ISSN: 0734-189X. DOI: https://doi.org/10.1016/S0734-189X(86)80047-0. URL: https://www.sciencedirect.com/science/article/pii/S0734189X86800470 (zitiert auf S. 40).

[Cha18] J. Chamary. *Why 'Pokémon GO' Is The World's Most Important Game*. Hrsg. von Forbes.com. [Online; posted Feb 10, 2018,06:59pm EST]. Feb. 2018. URL: https://www.forbes.com/sites/jvchamary/2018/02/10/pokemon-go-science-health-benefits/?sh=64dae7093ab0 (zitiert auf S. 10).

[Dud] Duden. *Durchschnittliche Länge eines deutschen Wortes*. Duden. URL: https://www.duden.de/sprachwissen/sprachratgeber/Durchschnittliche-Lange-eines-deutschen-Wortes (zitiert auf S. 37).

[Ede18] S. Edelstein. *How gaming company Unity is driving automakers toward virtual reality*. Hrsg. von digitaltrends.com. Mai 2018. URL: https://www.digitaltrends.com/cars/unity-automotive-virtual-reality-and-hmi/ (zitiert auf S. 29).

[GK96] A. V. Gelder, K. Kim. „Direct volume rendering with shading via three-dimensional textures“. In: *Proceedings of 1996 Symposium on Volume Visualization* (1996), S. 23–30 (zitiert auf S. 24).

[GV09] P. Gambette, J. Véronis. „Visualising a Text with a Tree Cloud“. In: *IFCS'09: International Federation of Classification Societies Conference*. Studies in Classification, Data Analysis, and Knowledge Organization. Dresde, Germany: Springer Berlin / Heidelberg, März 2009, S. 561–569. DOI: 10.1007/978-3-642-10745-0_61. URL: https://hal-lirmm.ccsd.cnrs.fr/lirmm-00373643 (zitiert auf S. 13).

[Ins17] Instagram. *Introducing Face Filters More on Instagram*. Hrsg. von Instagram.com. [Online; posted May 16, 2017]. Mai 2017. URL: https://about.instagram.com/blog/announcements/introducing-face-filters-and-more-on-instagram (zitiert auf S. 10).

[JLS15] J. Jo, B. Lee, J. Seo. „WordlePlus: Expanding Wordle's Use through Natural Interaction and Animation". In: *IEEE Computer Graphics and Applications* 35.6 (2015), S. 20–28. DOI: 10.1109/MCG.2015.113 (zitiert auf S. 16, 17).

[JSJ+18] W. Jentner, F. Stoffel, D. Jäckle, A. Gärtner, D. A. Keim. „DeepClouds : Stereoscopic 3D Wordle based on Conical Spirals". In: *Workshop on Visualization as Added Value in the Development, Use and Evaluation of Language Resources (VisLR III) LREC*. 2018. URL: https://scibib.dbvis.de/publications/view/755 (zitiert auf S. 17).

[KLKS10] K. Koh, B. Lee, B. Kim, J. Seo. „ManiWordle: Providing Flexible Control over Wordle". In: *IEEE transactions on visualization and computer graphics* 16 (Nov. 2010), S. 1190–7. DOI: 10.1109/TVCG.2010.175 (zitiert auf S. 16).

[LRKC10] B. Lee, N. H. Riche, A. K. Karlson, S. Carpendale. „SparkClouds: Visualizing Trends in Tag Clouds". In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), S. 1182–1189. DOI: 10.1109/TVCG.2010.194 (zitiert auf S. 13).

[Mica] Microsoft. *About HoloLens 2*. Microsoft. URL: https://docs.microsoft.com/en-us/hololens/hololens2-hardware (zitiert auf S. 11).

[Micb] Microsoft. *Architecture overview*. Microsoft. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/architecture/overview?view=mrtkunity-2021-05s (zitiert auf S. 30).

[Micc] Microsoft. *Introducing instinctual interactions*. Microsoft. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/design/interaction-fundamentals (zitiert auf S. 12).

[Micd] Microsoft. *Microsoft HoloLens 2*. Microsoft. URL: https://www.microsoft.com/de-de/hololens (zitiert auf S. 11).

[Mice] Microsoft. *Mixed Reality documentation*. Microsoft. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/discover/mixed-reality (zitiert auf S. 30).

[MTUK94] P. Milgram, H. Takemura, A. Utsumi, F. Kishino. „Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum". In: 1994, S. 282–292 (zitiert auf S. 9).

[RBW05] H. Regenbrecht, G. Baratoff, W. Wilke. „Augmented reality projects in the automotive and aerospace industries". In: *IEEE Computer Graphics and Applications* 25.6 (2005), S. 48–56. DOI: 10.1109/MCG.2005.124 (zitiert auf S. 11).

[Res12] M. Research. *Wearable Multitouch Projector*. Youtube. 2012. URL: https://www.youtube.com/watch?v=WLoMecZ80BQ (zitiert auf S. 10).

[Sb85] S. Suzuki, K. be. „Topological structural analysis of digitized binary images by border following". In: *Computer Vision, Graphics, and Image Processing* 30.1 (1985), S. 32–46. ISSN: 0734-189X. DOI: https://doi.org/10.1016/0734-189X(85)90016-7. URL: https://www.sciencedirect.com/science/article/pii/0734189X85900167 (zitiert auf S. 40).

[VWF09] F. B. Viegas, M. Wattenberg, J. Feinberg. „Participatory Visualization with Wordle“. In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), S. 1137–1144. DOI: 10.1109/TVCG.2009.171 (zitiert auf S. 13, 15).

[Vyn20] G. D. Vynck. *Unity Technologies Aims to Bring Video Game Tools Into the Real World*. Hrsg. von Bloomberg.com. Mai 2020. URL: https://www.bloomberg.com/news/articles/2020-05-07/unity-technologies-aims-to-bring-video-game-tools-into-the-real-world (zitiert auf S. 29).

[w3o] w3.org. *em, px, pt, cm, in. . .* w3.org. URL: https://www.w3.org/Style/Examples/007/units.html (zitiert auf S. 50).

[WCB+18] Y. Wang, X. Chu, C. Bao, L. Zhu, O. Deussen, B. Chen, M. Sedlmair. „EdWordle: Consistency-Preserving Word Cloud Editing“. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), S. 647–656. DOI: 10.1109/TVCG.2017.2745859 (zitiert auf S. 15, 16, 19, 20).

[WCZ+20] Y. Wang, X. Chu, K. Zhang, C. Bao, X. Li, J. Zhang, C.-W. Fu, C. Hurter, O. Deussen, B. Lee. „ShapeWordle: Tailoring Wordles using Shape-aware Archimedean Spirals“. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), S. 991–1000. DOI: 10.1109/TVCG.2019.2934783 (zitiert auf S. 16, 24).

Alle URLs wurden zuletzt am 13. 04. 2022 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Laufen a.N. 15.04.22 Jonas Österlein

Ort, Datum, Unterschrift