Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit Nr.

Visual Debugging for Particle-based Simulations of Fluids

Otilia-Camelia Dumitrescu

Course of Study:	Master of Science Information Technology
Examiner:	Prof. Dr. Daniel Weiskopf
Supervisors:	Dipl.-Inf. Markus Huber, Dr. rer. nat. Michael Krone, Stefan Reinhardt, M. A.
Commenced:	June 1, 2016
Completed:	December 1, 2016
CR-Classification:	I.7.2

Abstract

Visualizations represent an important tool that we have at our disposal when it comes to analyzing large data sets. A significant amount of data comes from simulations such as fluid, weather, biology and chemistry simulations. Due to increases in computation power the simulations have become more comprehensive, resulting in a larger amount of data. Increased volumes of the simulations require more specialized tools that can offer an insight so we can better understand the phenomena that is reproduced.

The present thesis presents a visual debugging plug-in for Particle-based simulations of fluids that can help the researchers to better explain the simulation scenario and to identify possible errors. Moreover, the tool can be used to comprehend modeling and development of new techniques. The environment in which I have implemented the plug-in is MegaMol, a system software focus on visualizing particle-based simulations.

There are four modules that I have implemented to enhance MegaMol functionality. In order to import a specific multidimensional data set I have created the $BGEODataSource$ module which converts Houdini geometry formats into MegaMol Particle List Data (MMPLD). By doing this, the simulation data is available for other modules that are already implemented.

To explore different particles that have certain properties I have created the $ScatterPlot$ module that offers a way to select and visualize interesting regions of the attribute space. The user can select two attributes that will generate a scatter plot and interact with it by brushing.

In order to get another perspective on the data I have implemented the $ParallelCoordPlot$ module which allow the user to identify different patterns and trends between various attributes. By choosing distinct attributes we can see the correlation between different properties and clusters within a specific value range.

The modules mentioned above work in the 2D space for observing the feature space. In $SimpleSpherePickingRenderer$ module we can select particles in the 3D space that will serve as input data for the $ScatterPlot$ and $ParallelCoordPlot$. This is done by a simple selection of the region of interest.

Contents

List of Figures

1 Introduction

1.1 Physics-based simulations for fluid animation

A relatively new and challenging field in computer graphics with encouraging computational and visually results it is represented by the physics-based simulation for fluid animations. There are two main concerns that researchers have to handle when developing simulations: from one point of view there is the visual aspect that should be fulfilled and on the other hand is the computational efficiency aspect.

There are several domains where the physics-based simulation can represent the flow of a fluid so well that the human eye cannot distinguish between reality and fiction. One application area where the fluid animation should pursue in the minutest details the real world it is illustrated by the film industry[Sol10]. Particular features that can confirm the authenticity of a moving fluid are small-scale features like splashes, droplets and foam. For high resolution fluid simulations these features cannot be omitted since a natural feeling of the environment can be achieved through this kind of small details. The process for creating such simulations is very time consuming and the developers have to increase the processing power to obtain good results. For this reason, simulations that have as main goal recreating real phenomenon are not computed in real-time.

From the other point of view, the computational efficiency aspect of fluid simulation is crucial in computer games industry and virtual reality industry [Sol10]. Here, all that matters is to run a stable simulation in real-time despite the lack of real world imitation. To create such a simulation, the developer has to reduce the physics complexity of the fluid solver and to rely on a low-resolution animation of fluid.

Despite of the area of application, whether film industry, medical simulations or computer games, the researchers have as main goal the improvement of both the fluid behavior and the visual quality of the resulting animations. In the next section, I will present two approaches for simulating the motion of fluids. In Chapter 2, I will describe the fluid solver that it is used for simulations imported by the visual debugging tool. Chapters 3 and 4 will focus on the visualization and implementation of the tool and in the last chapter I will present some use cases and results.

1.2 Eulerian and Lagrangian Fluid Dynamics

As I have mentioned before there are two main approaches for simulating the motion of a fluid. The idea behind Eulerian fluids is that the fluid is organized in aligned grid cells, and it cannot move outside this grid[Kel06]. Each cell contains fluid molecules or particles that have associated properties like pressure, density evaluated at fixed points of the grid (e.g. cell centers or nodes) . Hence, these quantities are time and space dependent. A special case of the Eulerian method is depicted in Fig.1.1 where, all the surface of a fluid is represented in grid cells from the same row. For a better understanding the picture is represented in 2D.

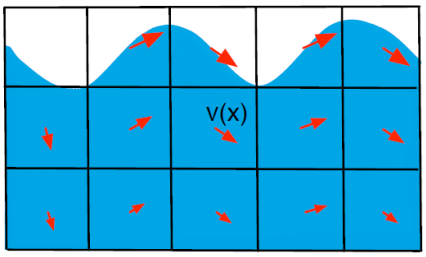

Figure 1.1: Illustration of the Eulerian approach, discretizing the fluid with a fixed (regular) grid. Quantities of the particles are measured at the cell centers like velocity **v**(x). [Sol10].

This method is prohibitive for interactive solutions of fluid animation but there are still some advantages that persuade computer graphics researchers to embrace Eulerian method. One of these advantages would be for sure the solution for incompressibility condition. The pressure values could be found much easier from the system of equation thanks to the discretization of the grid, so the incompressibility could be constrained in acceptable computational time [Sol10]. Another asset is a better simulation of smooth surfaces if the only interested part of a stand-alone fluid is its surface. Sinusoidal and gravity waves are very efficient implemented and used for this kind of simulations [Kel06]. The grid approach has usually advantages in the description of some properties such as pressure field and mass-density.

The tremendous disadvantage of the Eulerian fluid dynamics is the grid itself. It is more difficult to handle complex boundaries, interfaces between different fluids, free surfaces, splashes or droplets[CGFO06][LSSF06], when the fluid is deprived of its natural flow and has to stay within a pre designed grid. In addition, for a pleasant imitation of the real motion of the fluid, the grid has to be very fine and this will just overload the memory consumption. The other negative aspect it is represented by the advection part of the Navier-Stokes equations which leads to undesired damping due to the averaging

operations [MCPN08]. The shallow water equations [LP02] represent a hybrid solution between gravity waves and Navier-Stokes equations. This solution gives good results for simulations that combine dynamic waves and complex boundaries.

Nowadays, there are special data structures that could overcome some of these disadvantages by lowering the computation time or increasing the level of details for fluid simulations based on the Eulerian-grid method. For example, in [LP02] to simulate one frame, on a grid of 150x200x150 cells, of a viscous mud it takes 3 minutes but this result is still to slow and hence this method cannot be used for real-time simulations for computer games.

The alternative to the Eulerian approach is the Lagrangian fluid model. Fig. 1.2 presents the fluid discretization in 2D. Instead of a grid representation the entire fluid is composed of particles that can move freely with the fluid, so the quantities that are carried depend only on time.

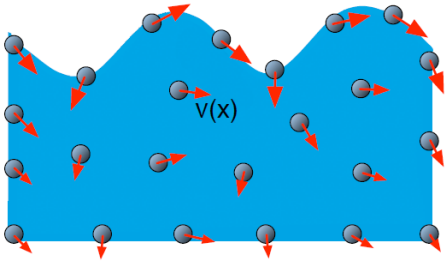

Figure 1.2: Illustration of the Lagrangian approach where the fluid is discretized by particles transporting physical quantities like velocity **v**(x) [Sol10].

The size of the particles can vary from macro to nano scales depending on the discretization of the domain. In molecular dynamics each particle represent one molecule while in SPH a particle represents a certain amount of volume [Sol10].

Compared to the previous presented method, Lagrangian method can handle advection more easily since the particles move freely along the velocity field. Moreover, complex boundaries, free surfaces, droplets or splashes could be more naturally handled [MCG03][MSKG05] [KAGBDG05] [SSP09] [SP09] [BTT09]. Although the computational effort is increased, this method can generate more accurate high-resolution fluid simulations.

A disadvantage of the particle-based approach is the difficulty to satisfy incompressibility condition of fluids. Instead of an incompressible fluid the method uses a slightly compressible one for which the pressure values are computed by an equation of

state(EOS)[Mon94] [BT07]. Another drawback is the high cost of reconstructing smooth surfaces out of particles.

This thesis uses simulations with the Lagrangian fluid solver based on the SPH particle method as source/input files. Hence in the next Chapter I will briefly present the SPH method.

1.3 Goal of the Thesis

The goal of the current thesis is to provide a tool that allows visual debugging of SPH simulations in order to identify simulation failures or parts of the simulation that require special attention.

There are two main areas of focus for this purpose:

- Spatial data view
- Non-spatial data view

The Spatial data view represents the first part of the thesis and one of the main tasks is the implementation of the import of animations from BGEO files(a BGEO file is a binary file format for storing Houdini geometry introduced in version 12) and assigning of simulation quantities to the particle rendering. Another task is the implementation of picking technique of particles to allow the selection of single particles, concerned regions of particles or a "smart selection" of particles (based on value ranges of particles properties).

The tasks for the non-spatial view are: implementation of a diagram view for the particle data, in particular scatterplots and parallel coordinate plots [Ins09]; the diagram view should be able to show information about all the particles as well as show information of a special selection. Furthermore the implementation of brushing in the diagrams that is linked to the particle view is another task for the non-spatial view.

In the last part of the thesis the benefits of the visual debugging tool are evaluated on the basis of a case study.

2 Smoothed Particle Hydrodynamics (SPH)

2.1 Navier-Stokes equations

The Navier-Stokes equations are generally used to describe the motion for an isothermal, incompressible fluid flow over time t [ESHD05]:

$$\rho(\frac{\partial}{\partial t} + v \cdot \nabla)v = -\nabla p + \mu\nabla \cdot (\nabla v) + f, \tag{2.1}$$

$$\nabla \cdot v = 0, \tag{2.2}$$

where ρ represents the mass-density quantity, v represents the velocity, *p* represents the pressure, μ represents the viscosity constant and *f* represents the sum of external forces that act on the fluid, e.g. gravity.

This formulation for the motion of a fluid is based on a grid method where the quantities depend on time and space(grid position). Equation (2.1) it is basically the Newton's 2^{nd} law for a fluid which describes the conservation of momentum. The second equation (2.2) was originally formulated in Euler's equation and describes the conservation of mass for an incompressible fluid.

A much easier approach is the Lagrangian formulation of Navier-Stokes equation for an isothermal, incompressible fluid:

$$\rho\frac{dv}{dt} = -\nabla p + \mu\nabla^2 v + f. \tag{2.3}$$

The grid representation can be replaced by a particle-based method where we assume that the number of particles is stable during the simulation and the mass for each particle is fixed. This implies that the mass conservation is guaranted, so the equation (2.2) can be omitted [Kel06].

The Newton's 2^{nd} law, $F = m \cdot a$, could be easily observed in the Lagrangian approach if we multiply each part of the equation (2.3) with the volume V_i for a particle with index *i* [Sol10].

$$m_i \frac{dv_i}{dt} = -V_i \nabla p_i + V_i \mu \nabla^2 v_i + V_i f. \tag{2.4}$$

The ordinary time derivative of velocity $v(i)$ is the acceleration for a Lagrangian fluid particle, while the right hand side of the equation (2.4) represents the sum of external and internal forces.

$$a_i = \frac{dv_i}{dt} = \frac{F_i}{m_i}, \text{where } F_i = F_i^{pressure} + F_i^{viscosity} + F_i^{external}. \tag{2.5}$$

Then, SPH approximations are applied to find the acceleration value a_i for a particle *i*. This value depends on the pressure p_i, the density ρ and the total forces F_i that have to be derived.

2.2 Force based solver

The SPH formulation that is used today in computer graphics was prior developed to simulate astrophysical problems [GM77], [Luc77]. The standard EOS-based SPH model was designed for compressible flow problems but nowadays there are various reformulations in order to enforce incompressibility. The SPH solver is a force-based solver that uses discrete sample points in order to approximate values and derivatives of continuous field quantities. The sample points can represent local properties of the field such as temperature, pressure or density that are obtained by weighted averages of a surrounding area. The particles also carry specific quantities like mass, position or velocity [Kel06].

SPH is an interpolation method that integrates over the domain Ω any function or field variable $A(x)$

$$A(x) = \int_\Omega A(x')\delta(x - x')dx', \tag{2.6}$$

where the *dx'* is the differential volume element and $\delta(x - x')$ is the Dirac delta function. Instead of the Dirac delta function the integral representation of $A(x)$ can use a smooth kernel W with *h* as width or core radius, but this is only an approximation of equation 2.6:

$$A(x) = \int_{\Omega} A(x')W(x - x', h)dx', \tag{2.7}$$

The kernel function W must satisfy the following constraints in order to be used in the integral representation:

- the kernel function has to be normalized

$$\int W(x - x', h)dx' = 1. \tag{2.8}$$

- the Delta function property has to be satisfied

$$\lim_{h \to 0} W(x - x', h) = \delta(x - x'). \tag{2.9}$$

- the non-zero area of the smoothing function has to be defined (compact support property)

$$W(x - x', h) = 0, \text{when } |x - x'| > h. \tag{2.10}$$

There are two golden rules of SPH [Mon92]: the first rule stipulates that the best assumption for a kernel of a new interpretation of a SPH equation is a Gaussian. Equation 2.11 presents an isotropic Gaussian kernel, in *n* dimensions.

$$W_{gaussian}(x, h) = \frac{1}{(2\pi h^2)^{\frac{3}{2}}} \cdot e^{-(||x||^2/2h^2)}, h > 0. \tag{2.11}$$

The smoothing kernels used for Lagrangian approach are though the B-Spline and Q-Spline kernels because they have compact support and are more accurate from the computational point of view. The second golden rule refers to the reformulation of equation 2.7 such that the density is placed inside the operators.

Replacing the integral with a summation and the differential volume element dx' with volume V we obtain the discretized form of equation 2.7 over a finite set of fluid particles.

$$A(x_i) = \sum_j A_j V_j W(x_i - x_j, h). \tag{2.12}$$

Knowing that the volume V_j can be written as $V_j = \frac{m_j}{\rho_j}$, where m_j is the mass of a particle j and the ρ_j is its mass-density we obtain the basis formulation of the SPH method. Now we are able to generate approximations for several continuous quantity fields, via

$$A(x_i) = \sum_j A_j \frac{m_j}{\rho_j} W(x_i - x_j, h). \tag{2.13}$$

The Gradient and the Laplacian of the smoothed quantity field can be easily computed since the only affected term by derivatives is the smoothing kernel

$$\nabla A(x_i) = \sum_j A_j \frac{m_j}{\rho_j} \nabla W(x_i - x_j, h), \tag{2.14}$$

$$\nabla^2 A(x_i) = \sum_j A_j \frac{m_j}{\rho_j} \nabla^2 W(x_i - x_j, h). \tag{2.15}$$

2.3 Different forces overview

As we can see from equation 2.5 there are two types of forces that affect the flow of a fluid:

- Internal forces: pressure and viscosity
- External forces: gravity, buoyancy and surface tension

All these forces depend on known values of the mass and mass-density quantities.

2.3.1 Density

As mentioned above, the formulation of a SPH method depends on the mass and mass-density quantities. Although the mass of a particle has a constant value prior defined by the user, the mass-density quantity has to be computed before the other approximations of the SPH [Kel06]. A continuous field of the fluid can be computed using equation 2.12. Since the mass-density depends only on the mass of a particle *i*, the mass-density becomes

$$\rho_i = \sum_j \rho_j \frac{m_j}{\rho_j} W(x_i - x_j, h), = \sum_j m_j W(x_i - x_j, h). \tag{2.16}$$

Equation 2.16 introduces unwanted errors for particles that are close to the boundary. At these locations, averaging over the surrounding area will include empty parts of the neighborhood and will create an underestimated value for mass-density that will propagate to pressure and other forces [Sol10]. The continuity equation [Mon94] solves the underestimated densities problem but the results for simulations with large time steps are not stable. So, the standard density summation equation is more often used.

2.3.2 Pressure

The pressure force of a particle in one of the forces that arise from within the fluid. According to the ideal gas law the pressure is

$$p = \frac{nRT}{V}, \tag{2.17}$$

where $V = \frac{1}{\rho}$ is the volume per unit mass, *n* is the number of moles, *R* is the ideal gas constant and *T* is the absolute temperature of the gas. In case of an isothermal fluid with constant mass, the pressure is $p = k\rho$ where *k* is a gas stiffness constant. Hence, knowing the mass-density of a particle we also know its pressure.

The pressure force at particle with index *i* in SPH approach is

$$\begin{aligned} \mathbf{f}_i^{pressure} &= -\nabla p(\mathbf{x}_i) \\ &= -\sum_{j \neq i} p_i \frac{m_j}{\rho_j} \nabla W(\mathbf{x}_i - \mathbf{x}_j, h). \end{aligned} \tag{2.18}$$

A well known problem of equation 2.18 is that the resulting pressure force is not symmetrical. A simple and stable solution to this issue is presented in [MCG03] where the pressure force yields

$$\mathbf{f}_i^{pressure} = -\frac{m_i}{\rho_i} \sum_{j \neq i} \frac{m_j}{\rho_j} \frac{p_1 + p_j}{2}, \nabla W(\mathbf{x}_i - \mathbf{x}_j, h). \tag{2.19}$$

2.3.3 Viscosity

The other internal force of a particle with index *i* is its resistance to flow, also known as viscosity. The main use of viscosity force is to stabilize a particle system but it

also represents the internal friction of a fluid. In SPH method the viscosity force is approximated via

$$\begin{aligned} f_i^{viscosity} &= \mu \nabla^2 v(x_i) \\ &= \mu \sum_{j \neq i} v_i \frac{m_j}{\rho_j} \nabla^2 W(x_i - x_j, h), \end{aligned} \tag{2.20}$$

where μ is the viscosity constant of a particle which defines the strength of how viscous is a fluid. The same asymmetrical problem, as in pressure force, appears to the viscous force because of the differences in velocity from particle to particle. The solution is also presented in [MCG03]

$$\mathbf{f}_i^{viscosity} = \mu \sum_{j \neq i} (v_j - v_i) \frac{m_j}{\rho_j} \nabla^2 W(\mathbf{x}_i - \mathbf{x}_j, h). \tag{2.21}$$

2.3.4 Gravity

The gravitational force field is an external force that acts on the particles of the fluid. All the fluid particles obey equally to the gravitational force which is

$$f_i^{gravity} = \rho_i g, \tag{2.22}$$

where *g* is the downward gravitational acceleration [Kel06].

3 Visualization

According to Moore's Law from 1975, the number of transistors in a dense integrated circuit doubles approximately every two years. The exponential improvement that the law describes pushed also the limits of storage mechanisms so today's computer systems can store very large amounts of data. Nowadays, almost each human action can generate valuable information that is automatically recorded via sensors and monitoring systems. Among these, the simple action of paying with credit card or entering into a building with the access card are just some examples of every day data that are collected.

Every year 1 Exabyte of data are generated and an impressive part is stored in digital form according to researchers from the University of Berkley. The lack of exploration methods for this huge volumes of data can lead in some cases from useful information to useless information [Kei02]. The visual data exploration process aims to bring insight into large amounts of data by direct involving the user perception in the course of analysis.

Among the benefits of the visual data exploration process is a reduced required level of understanding complex mathematics or statistical algorithms and also it is more intuitive for the user. Moreover, the process can deal with highly inhomogeneous and noisy data and these are advantages over automatic data mining techniques from machine learning or statistics. In cases where automatic algorithms fail, the visual data exploration facilitates a faster data analysis with better results and a high degree of confidence in the findings through the direct involvement of the user [Kei02].

3.1 Visual data mining techniques

The visual data mining techniques can be classified based on three criteria [Kei02]:

- The data to be visualized
- The visualization technique
- The interaction and distortion technique

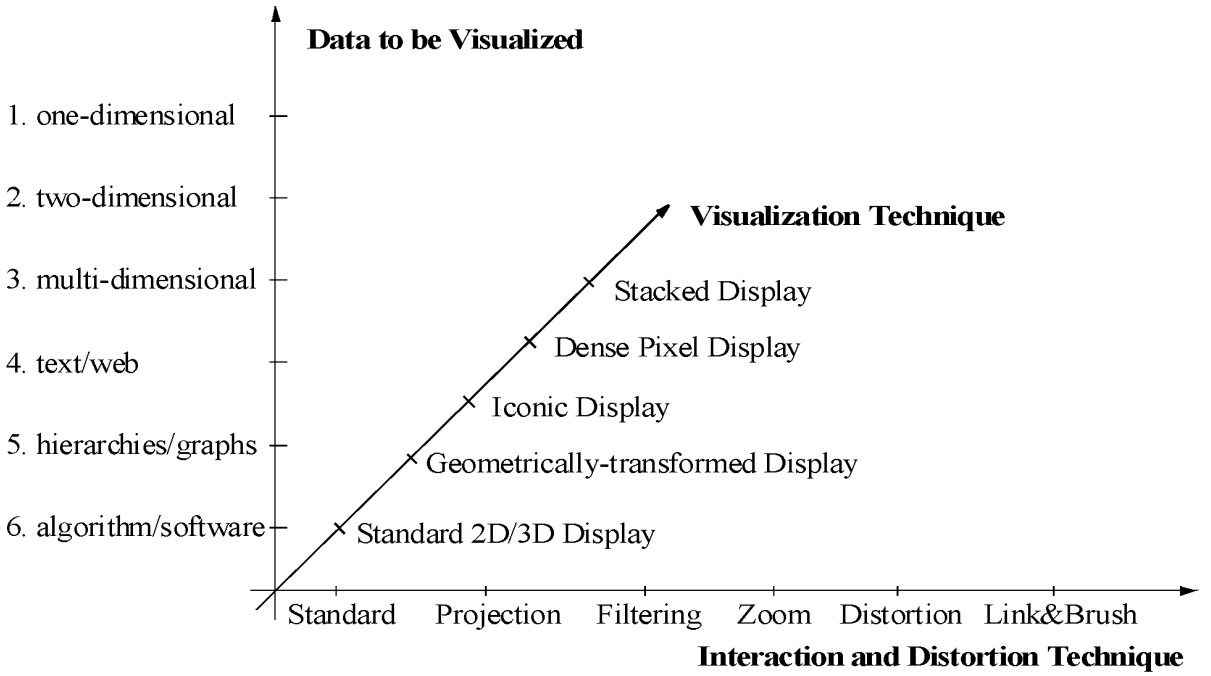

Figure 3.1: Classification of Information Visualization Techniques [Kei02].

The present thesis uses simulations that have multidimensional data and as visualization techniques the standard 2D/3D displays and parallel coordinates plot. Because of the orthogonality of the visual data mining techniques, I have used in conjunction with the visualization techniques, the interactive brushing and linking as interaction and distortion technique presented in Fig. 3.1.

In this chapter I would like to show the above mentioned techniques for visualization that can help us to identify errors or a better understanding of the simulation scenario. The visual debugging mechanism developed in MegaMol can cover several use cases and can be used to comprehend modeling and development of new techniques. In the next pages I will present each feature of the tool, but also some existing features of MegaMol that the current tool can operate with for a better understanding of some complex simulations. After reading this chapter the user should be able to use the debugging tool so it can also be seen/used as an effective handbook.

In chapter 4, I will explain the underlying pipeline and the data flow to generate visualizations, but now I want to emphasize the components of the visualization tools. The first feature that I would like to present refers to the simulation itself. The current tool can import and display different particle-based simulations independent the number or type of attributes that each particle data set is carrying. Fig. 3.2 presents some of the data sets that I have tested.

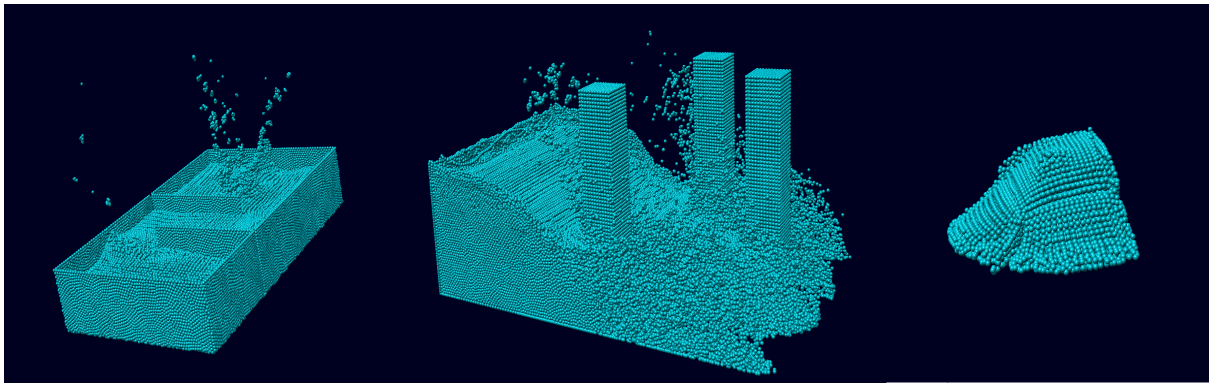

Figure 3.2: Different Particle-based Simulations.

3.2 3D View

Making use of the already implemented 3D Viewer of MegaMol the user has various directions for visual analysis of the simulation. First of all the user can adjust the camera viewport by holding the left click button and moving the mouse over the simulation. The possibility to move the camera around the simulation is a great feature because in some cases splashing phenomena may hide a part of the particles that can be seen from other viewing angles. Another particularity of the 3D Viewer is the option to zoom in/out by holding the scroll button and moving the mouse up or down over the screen. Hence, the user can also navigate inside the fluid body.

Fig. 3.3 presents a simulation in the 3D Viewer where the user can only see the spatial representation of the imported simulation of the fluid, namely the position of the particles represented by spheres.

The animation instance of the 3D Viewer is one more option that can be used to evaluate the simulation frame by frame or even to speed it up if there are less interesting parts. These aspects and many others like camera light instance or stereo instance can be set in the Parameters drop-down menu that is depicted in Fig. 3.3.

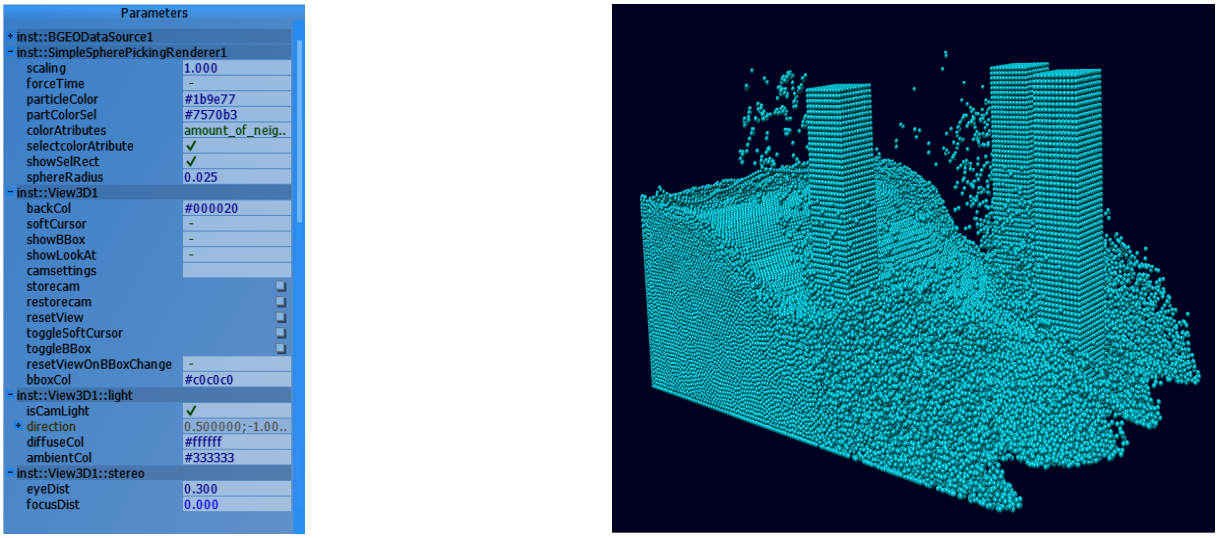

Figure 3.3: Parameters drop-down menu and a 3D visualization of the simulation.

3.3 Non-spatial data views

In order to get another perspective on the data sets there are two different non-spatial data views: the scatter plot and the parallel coordinates plot. The scatter plot offers a way to select and visualize interesting regions of the attribute space. The user can find useful information about picked attributes but what it is more important, the user can see a correlation between two meaningful attributes.

Fig. 3.4 left shows a scatter plot that maps the density attribute to the x axis, and the magnitude of the velocity to the y axis for a data set that contains collision objects among the fluid particles. We can identify a cluster of points in the lower-left part of the diagram which corresponds to particles that have low density values and varying velocity values. From a physical perspective the low density particles are the particles that are splattered when colliding with rigid objects. This region gives us a hint regarding the splashing of the fluid. By selecting this region we can have a visual confirmation of the presumed particles.

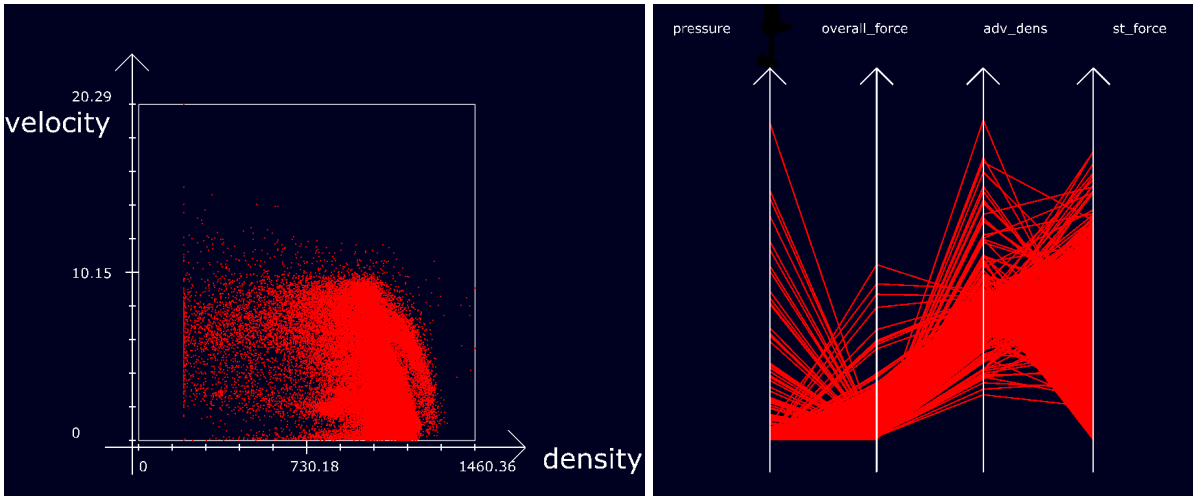

Figure 3.4: Scatter plot where the velocity is plotted over the density (left side). Parallel Coordinates plot displaying pressure, overall force, advection density and surface tension force attributes (right side).

The other non-spatial representation,the parallel coordinates plot, is a visualization technique for multidimensional data. The construction of the diagram was made by placing axes in parallel with respect to the x axes of the display, which is the most common approach also in literature [HW13] [Ins09]. The parallel coordinates plot can show different patterns and trends between various attributes which are arbitrarily exchangeable. In Fig. 3.4 (right) there are 4 axes representing pressure, overall force, advection density and surface tension force attributes of the data set. Although there are many polygonal lines that intersect the 4 vertical axes, the user can still identify a pattern between low pressure values, low overall forces values, average advection density values and a broad range for the surface tension force values.

3.4 Brushing and linking

By pressing the TAB key, the user can switch from camera view to selection view in the 3D space. The selection is one of the most important features of this debugging tool and first I would like to present the selection of particles in the spatial domain. By holding the left click button and moving the mouse over the particles, a semi transparent rectangle will appear. When releasing the button all the particles that were in the selected region will have a color that differs from the initial one which user can choose from the Parameters menu.The left part of Fig.3.5 shows the afore mentioned scenario.

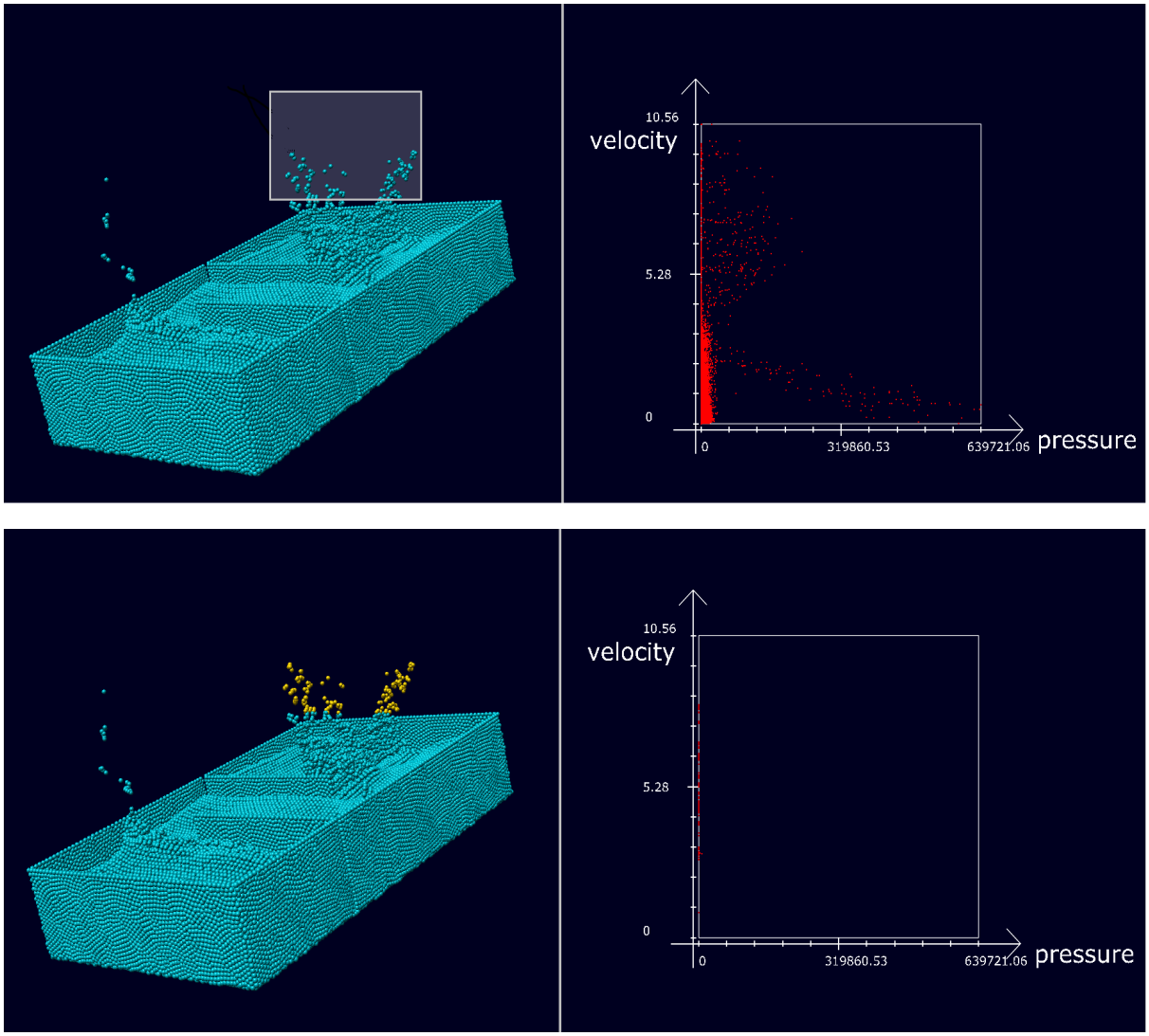

Figure 3.5: Brushing and linking technique in the 3D viewer with results in Scatter plot.

By stand-alone the selection of particles in 3D space does not offer any new information for the selected region. Using the linking technique, the 3D view can be connected either to a scatter plot or a parallel coordinates plot. The selection of an area in the spatial domain exposes only the scatter points that correspond to their values of picked particles in the scatter plot. This situation is presented in Fig. 3.5. The scatter points from the 2D diagram represent different values of the corresponding attributes for the selected particles in the 3D view.

The selection mechanism works in both directions. So, if the user requires to visualize the particles between a fixed range for two picked attributes, all that he should do is to draw a rectangle over the desired region in the scatter plot. During the selection the

user should press and hold the ALT key and the left click button of the mouse. When releasing the mouse button, the selected scatter points in the diagram will be displayed as particles with a different color from the already simulated ones.

Moreover, the selection mechanism in the Scatter plot has another feature, the possibility to draw multiple rectangles on different regions of the diagram and link this multiple-selection to the 3D view. If the user wants to select a single region, after each analyzed area he/she has to check the $resetSelection$ parameter from the Parameters menu since the multiple selection mechanism is by default enabled. Fig. 3.6 presents brushing and linking techniques with respect to single and multiple selection.

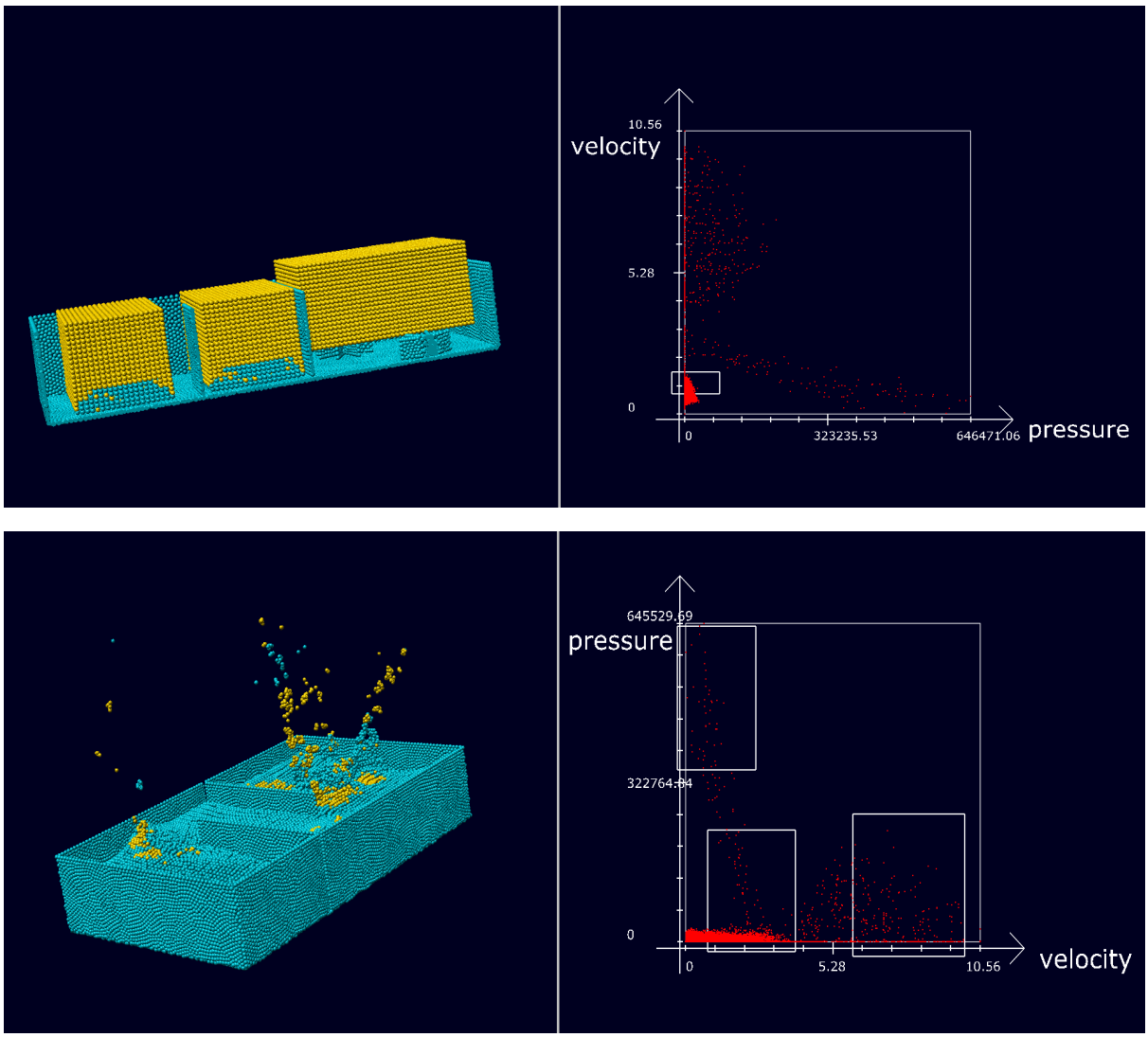

Figure 3.6: Brushing and linking technique in the scatter plot with results in 3D viewer.

The linking between the parallel coordinates plot and the 3D simulation is mutual. In other words, the user can select either a region in the parallel coordinates plot and visualize the result in the 3D viewer or a sector over the simulation and interpret the outcome of the plot. Both cases have a good visual impact as shown in Fig. 3.7. The selection in the parallel coordinates diagram has the same mechanism as in the scatter plot.

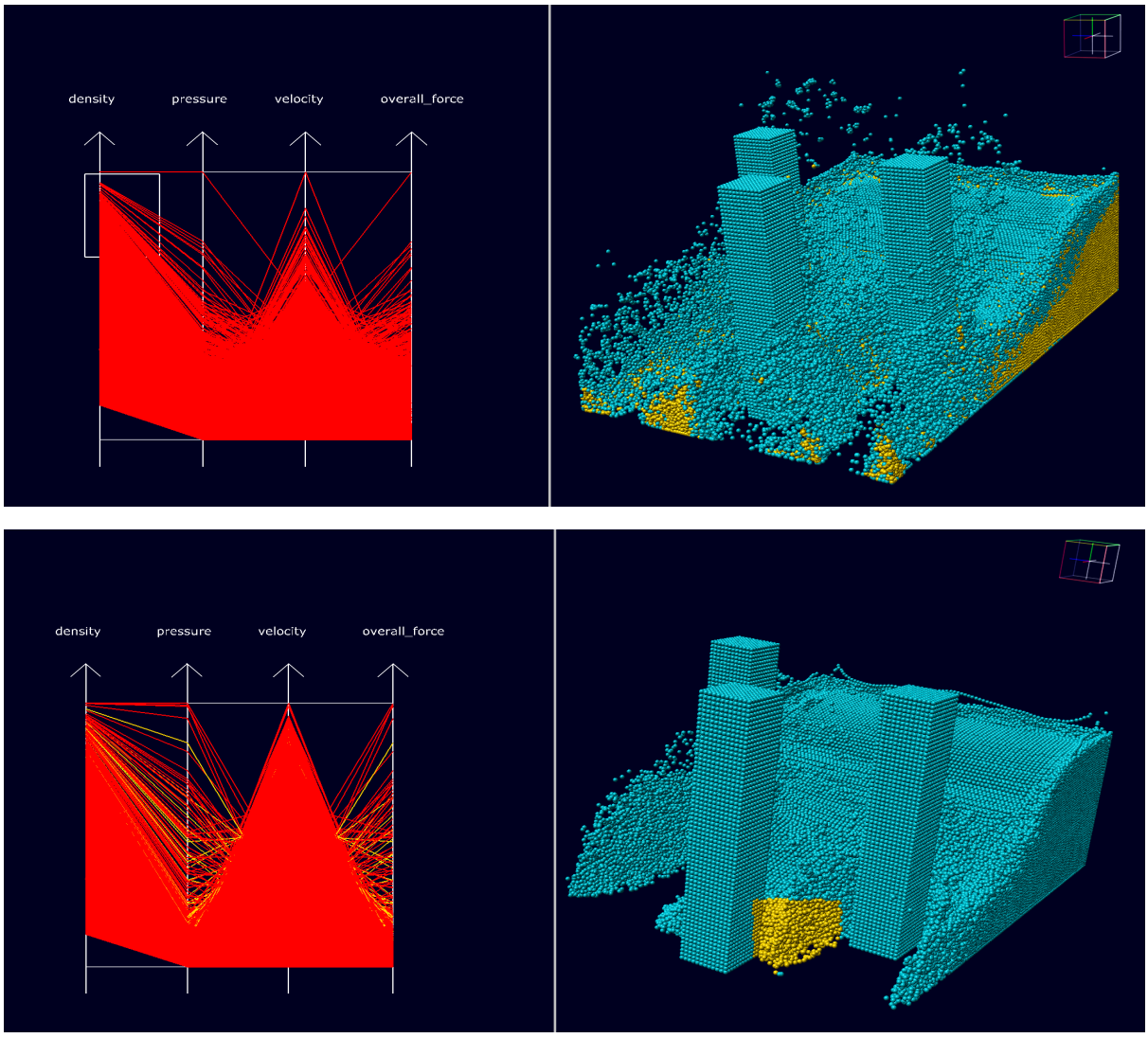

Figure 3.7: Brushing and linking technique between Parallel coordinates plot and 3D viewer.

3.5 Clip Plane

Even though the selection mechanism helps the user to identify special cases in the flow simulation, there are situations when still some information is occluded. A good example is presented in Fig.3.7 where a part of the particles are selected but the user cannot see them all, because some of them are at the surface of the fluid and the rest are within it. Although the zoom in mechanism of the 3D Viewer allows an inside view of the fluid, the user cannot have a perspective view for a further analysis.

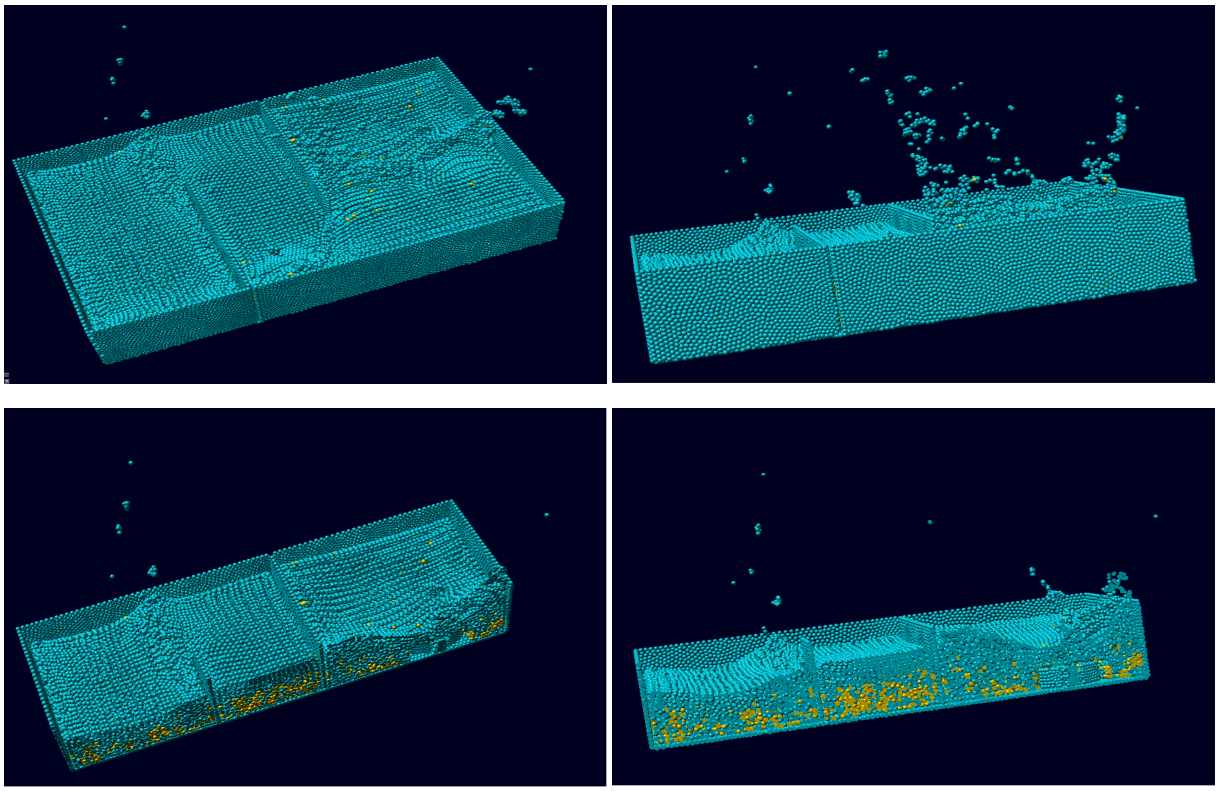

Figure 3.8: Overview: the whole simulation from above (a), aside (b) and with a clip plane on x axis from above (c) and aside (d).

MegaMol tool has a feature named Clip Plane that the discussed debugging visualization makes use of. As the name suggests, it gives the possibility to cut a simulation by a plane set by a direction and a point in the x,y,z coordinates. Fig. 3.8 displays the same simulation as in Fig. 3.8 but now a clip plane on the x axis is enabled from the Parameters drop-down menu. Using this feature the user can easily determine all the selected particles even if some of them are obscured by the surface of the fluid.

3.6 Changing attributes

The visualization techniques used in the debugging tool, the standard 2D/3D display and the geometrically transformed display, have unfortunately some limitations. The data exploration has as border the number of attributes, also known as the dimensionality of the data set. In the 2D scatter plot the maximum number of attributes that can be displayed at a time is two and in the parallel coordinates plot, I have restricted the k-dimensional space to k=4 that means there are four axes that are arbitrarily exchangeable, but it could be easily extended to more axes. The multi-dimensional data sets of particle-based simulations have on average 7 to 10 attributes, so it is clear that the use of these visualization techniques can not display all the information.

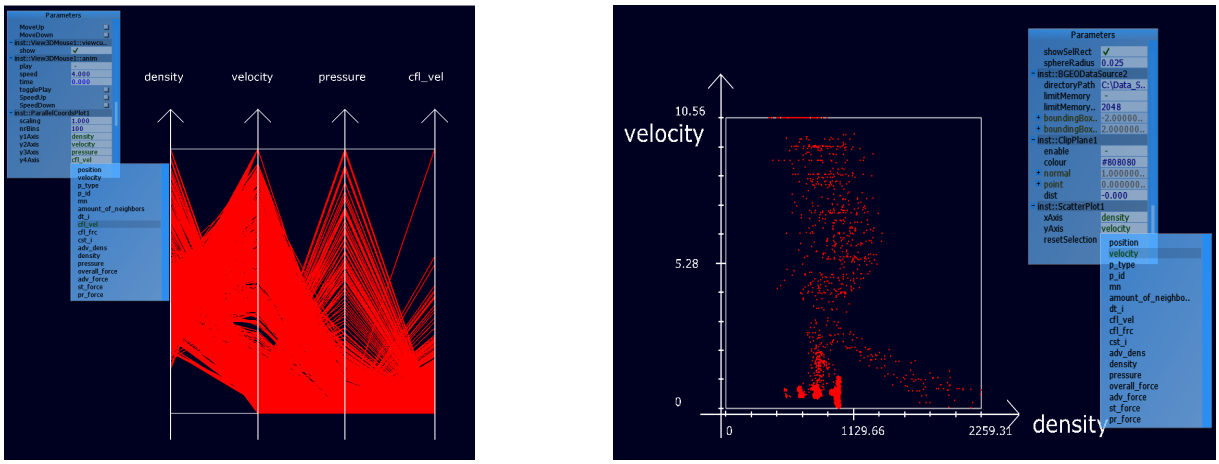

Figure 3.9: Changing attributes

To overcome this, the present tool has as option the possibility to change the attributes name, and implicit their values, from the Parameters menu. In Fig. 3.9 the user can select from a predetermined list of attributes which one to be displayed on the x axis, and then independent the choice he/she made, can choose the attribute for the y axis. Picking the same attribute for both axes it is also possible but the outcome visualization will contain scatter points along the line with equation y=x. The utility in this case would be that the user can better identify dense regions and maybe select them to get more details.

There might be cases, especially when changing the simulation data sets, where the desired attribute name will not match any of the actually attributes of the particles, since it is missing from the simulation file. In this situation the concerned parameter will be set by default to the first parameter of the list.

In the parallel coordinates plot, according to Fig. 3.7, the user has the same degree of freedom of selecting which attributes to be displayed. Setting all axes to the same attribute will draw parallel lines with the x axis of the display, since the attribute axes are parallel to the y axis of the display.

Although the possibility to select an attribute for each axis will not give the user an overview of all the aspects of the simulation, he/she still can analyze each of the attributes and correlations between them without loosing valuable information.

3.7 Attribute values reflected in color

Another feature implemented in the 3D renderer is the option to change the color of the rendered particles depending on the picked attribute. The color map is generated by normalizing the attribute range for each frame and the information is stored on the red channel. In case of a negative value of an attribute the green channel is also set to its maximum value.

Fig. 3.10 presents a color map for the simulation where user can identify different particle types: boundary particles are colored with dark blue, fluid particles with pink and the propeller which spins the particles with cyan.

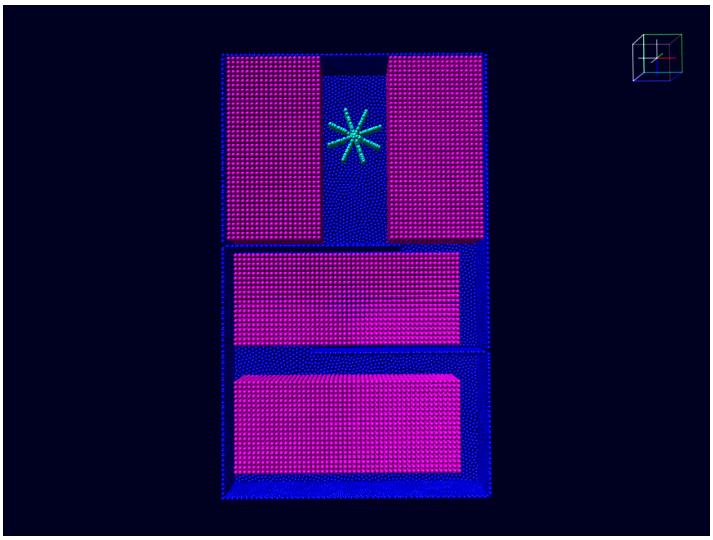

Figure 3.10: Color Map depending on particle type.

Some attributes have very similar values in a narrow range and mapping these values to color will not give much insight to figure out possible errors because the human eye has a deficit in distinguishing akin colors side by side. However, in simulation of fluids from

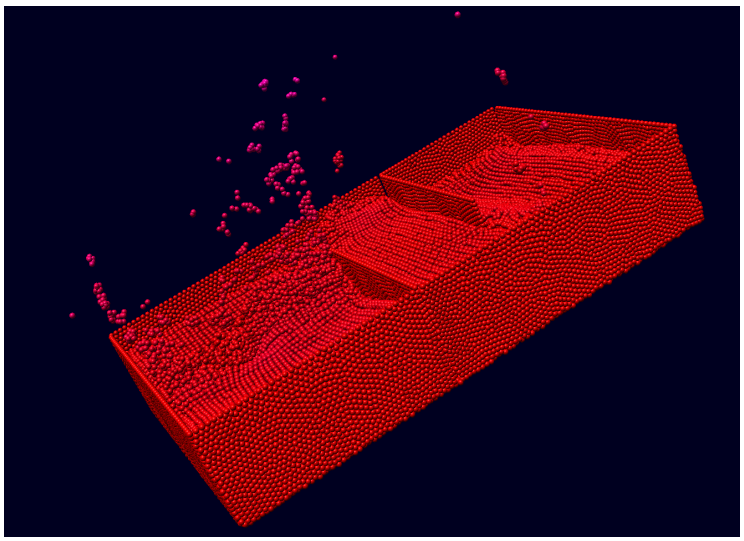

Figure 3.11: Color Map depending on magnitude of velocity field.

one frame to another there could be some particles that have big differences for the same attribute. Fig. 3.11 displays also the velocity attribute in color. When the two fluids interact or when a fluid collides in a fixed boundary then splashes appear, so the velocity values of the particles are changing sharply. Splashes and droplets are way more difficult to simulate and require more attention when validating a simulation, so the possibility to have also a colored visualization of this special cases has only advantages.

Since this feature can be harmful when the user makes interactive changes in the scatter plot or parallel coordinates plot, in addition to the parameter that picks one attribute from the list, the user should also enable the 'selectcolorAttribute' parameter.

3.8 Sampling in Parallel Coordinates Plot

The parallel coordinates technique has as main advantage the potential to create visualizations for multidimensional data. Displaying each multidimensional data item as a polygonal line which crosses all the axes can create cluttered visualizations which in most cases have no good use. To overcome this drawback, using the idea presented in the work by[Ell08], the discussed tool has a feature that allows random-sampling of the data items and displays only a percentage of them.

The approach proposed by [Ell08] uses z-index method for choosing the items in the sample. For selecting samples of random particles, I have created a vector that stores all the indices of the particles in the simulation and then I randomly shuffle it. Then, I use x percent from the beginning of this shuffled vector as indices to render that x percent of particles. Knowing that the number of particles is steady over the entire simulation I randomize the particles only once. If the number of particles differs between frames than, the random generator should be applied for each frame. However this could create unwanted artefacts.

The sampling rate of the particles can be adjust smoothly down to 1% from the Parameters menu. In order to create a reliable visualization the tool provides display continuity such that the user cannot see flickering artefacts when re-displaying deleted lines in the reverse order of removal [Ell08].

Fig. 3.12 and Fig. 3.13 presents parallel coordinates plots at different sampling rates. At 100% percent the user can get an idea about the aspect of the correlation between attributes but the lines are too tangled to identify templates. Around 33% the user can start to detect some patterns but there are still bushy areas of the plot. If the down-samples to 5%, than narrow strips start to appear and the user can find more easily trends in the connected lines between axes.

Among the advantages of sampling for clutter reduction we can mention that it does not affect the characteristics of the plotted data items, it is suitable for interactive applications such the present debugging plug-in and excels at scalability [Ell08]. An important drawback of this technique is the possible loss of outliers since these values are more sparse. When sampling, it is more likely that the outliers will disappear before the other occluding lines. Another inconvenience is that the user cannot avoid co-incident lines by sampling but the solution in literature refers to reducing the opacity of the lines. This can create better visualizations from which the user can get an insight.

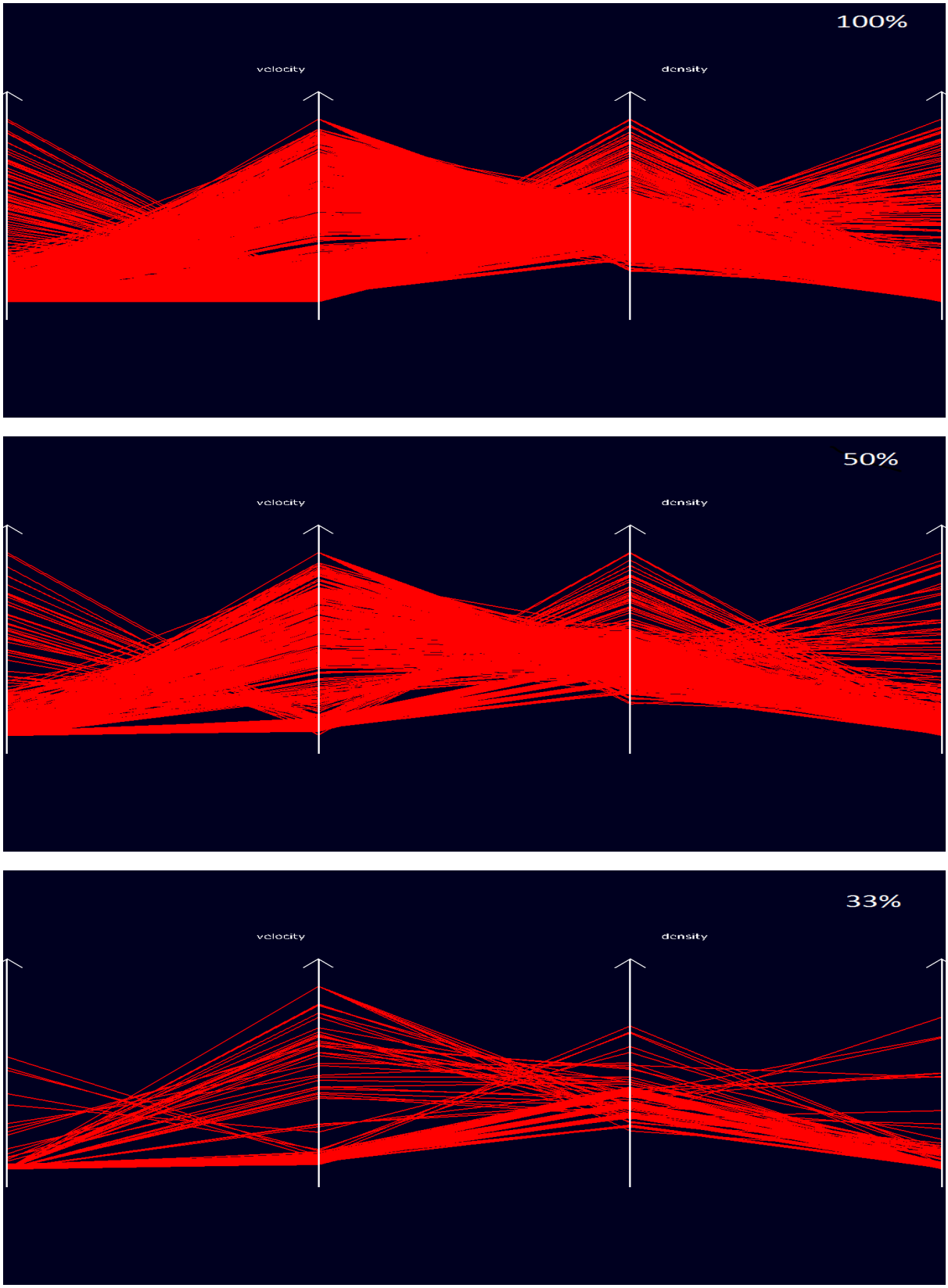

Figure 3.12: Parallel coordinates visualization at sampling rates from 100% to 33%.

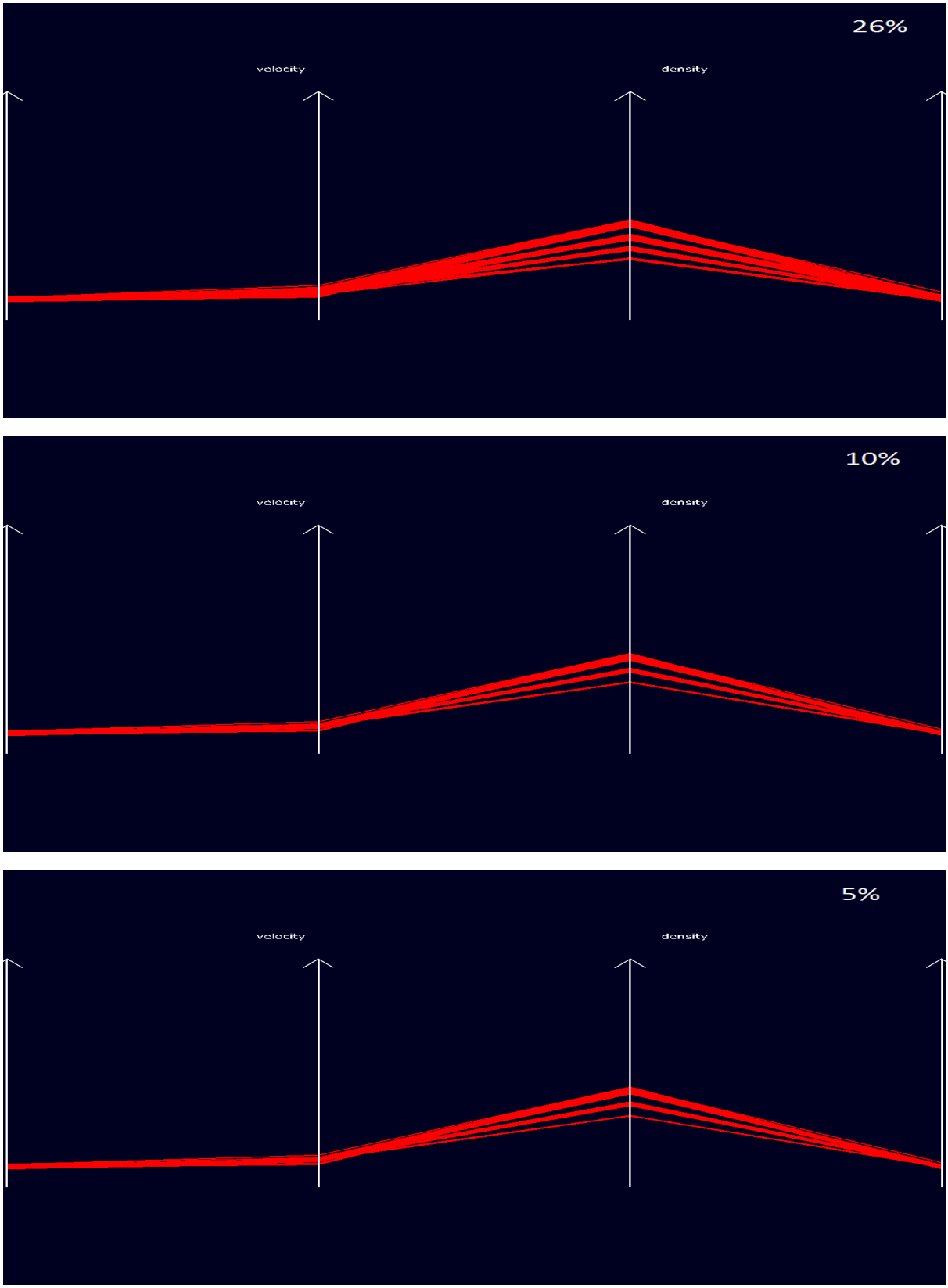

Figure 3.13: Parallel coordinates visualization at sampling rates from 26% to 5%.

4 Implementation in MegaMol tool

4.1 Introduction to MegaMol

Particle-based fluid simulations produce particle-based datasets that can comprise up to several millions of particles and several thousand time steps. MegaMol is an appropriate tool that could handle such amount of information and also with good results in visualization output.

The system software MegaMol for visualization research on particle-based data was created as part of a collaborative research center between biologists, visualization experts and physicists [GKMRE15]. It was developed by S. Grottel, M. Krone, C. Muller, G. Reina and prof. T. Ertl as a research project at Visualization Research Center of the University of Stuttgart.

The idea behind MegaMol was rather to create a visualization prototyping system than a comprehensive application like ParaView which is more concerned on abstraction and complexity [SML96]. It provides reusable components and data management functionality, which helps visualization researchers to focus more on the actually algorithms rather than UI toolkit part. The system arhitecture of MegaMol has two levels of abstraction: the upper level that links the front end and the back end via the Core library and the lower level, the application logic which contains Modules and Calls.

The MegaMol Core library provides general functionalities with respect to the configuration of application management and loading of plugins at runtime. It also ensures the interface between the front end and the back end which has well-known Modules and Calls for loading, rendering and displaying the data set. Despite this library, accordingly to the application goals, users can develop independent plugins that will be loaded at runtime by the core library.

The application logic, as I have mentioned above, consists of two important components: Modules and Calls. The Modules represent the functional part while the interconnection part is defined by the Calls. The developer can freely decide for the granularity of the functional component because there are no limits specified, e.g. a multi-pass rendering algorithm can be implemented in a single Module or in multiple Modules interconnected [GKMRE15]. One important design aspect of MegaMol is the control flow represented by

Call classes which model interfaces. To actually connect to a Module you need instances of Call classes which provides access to the Module's functionalities. The pull pattern was implemented for the Call classes to avoid copy operations and duplication of data. A Call can connect multiple callback functions modeling different calling Intents specifying the reason for invoking the Call [GKMRE15].

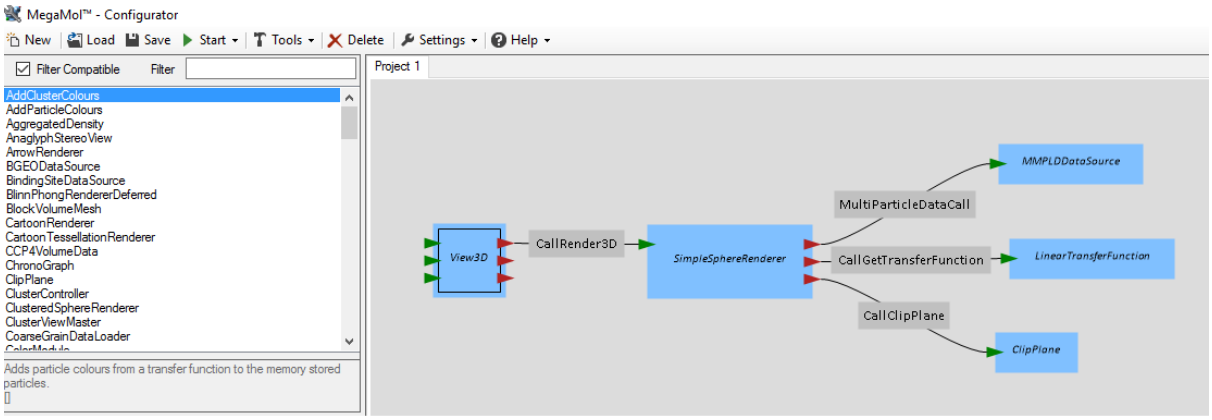

Figure 4.1: A simple Module Graph example in the MegaMol Configurator application.

Fig. 4.1 presents a simple Module Graph for particle visualization. Starting from the left, the View3D Module represents the entry Module and it has the functionality of a camera view. This Module requests information via the CallRenderer3D, an Intent to generate the output image, to the SimpleSphereRenderer Module. The rendering technique implemented in this functional component is sphere-rendering using GPU-based ray-casting [Gum03], [GRDE10]. Connected to this central Module there are three others: MMPLDDataSource Module loads particles from MMPLD files(MegaMol™ Particle List Data is a very fast loading binary memory dump of MegaMol™ [Meg]) and sends them to the rendering Module when requested via MultiParticleDataCall. Moreover, the renderer can use a ClipPlane Module for clipping the particles by x, y or z axis and a LinearTransferFuncion Module that maps a scalar attribute to a color (e.g. density).

The user can interact with all Modules through Parameters using a GUI front end and configuration files. Parameters encapsulate a value (float, int, string, file name) and meta-information about its type (e.g. a description and maximum/minimum value). MegaMol provides several front ends so each expert can decide which one to use accordingly to the targeted application domain.

There are two more types of interfacing points in modules: the caller slots and the callee slots [GKMRE15]. The caller slots indicated in the graph module with red arrows request information from the callee slots, green arrows in the module graph.

Fig 4.1 shows a specialized Configurator front end for MegaMol for building module graphs. This configurator loads the Core library and all the plugins that have been built and lists all the Modules and Calls that were included. It stores the module graph in the required XML format.

The key features of MegaMol that the developers have identified during successful projects are [2]:

- focus on particle data
- plugin system and no monolithic structures
- intention driven module interconnect

One important feature of MegaMol for this work is the focus on particle data. This was also the main reason for developing the visual debugging tool in MegaMol, since it is optimized for interactive visualization of 3D particles. It contains multiple glyph render implementations and a call mechanism which is proper for transporting time-dependent data sets. Furthermore, the framework does not impose the functional granularity of the modules and the user has the freedom to create multiple pipelines. Control flow is another essential feature. The application logic depends on interconnection between modules through connection end points defined by Call classes. Through one Call connection a Module can request all data or just a part of them, communicate meta data such as the corresponding attributes or just probe for data updates. All these are examples of invokable Intents that the Call mechanism of MegaMol can handle.

4.2 SPHVIS Plugin

For the purpose of the thesis I have developed a Plugin named $sphvis$. I have created two module graphs using the MegaMol Configurator front end to display different functionalities of the plugin.

Fig. 4.2 shows a Module Graph that contains both Modules and Calls from sphvis plugin but also from the Core library of MegaMol. Starting from the left to the right, the first Module is the SplitView Module from the Core library. With this Module I can split the view Module and display in parallel two different views. Via the CallRenderView, the SplitView requests information from two different view Modules. The first one, in the upper part of the figure, called View3DMouse represents a 3D View Module

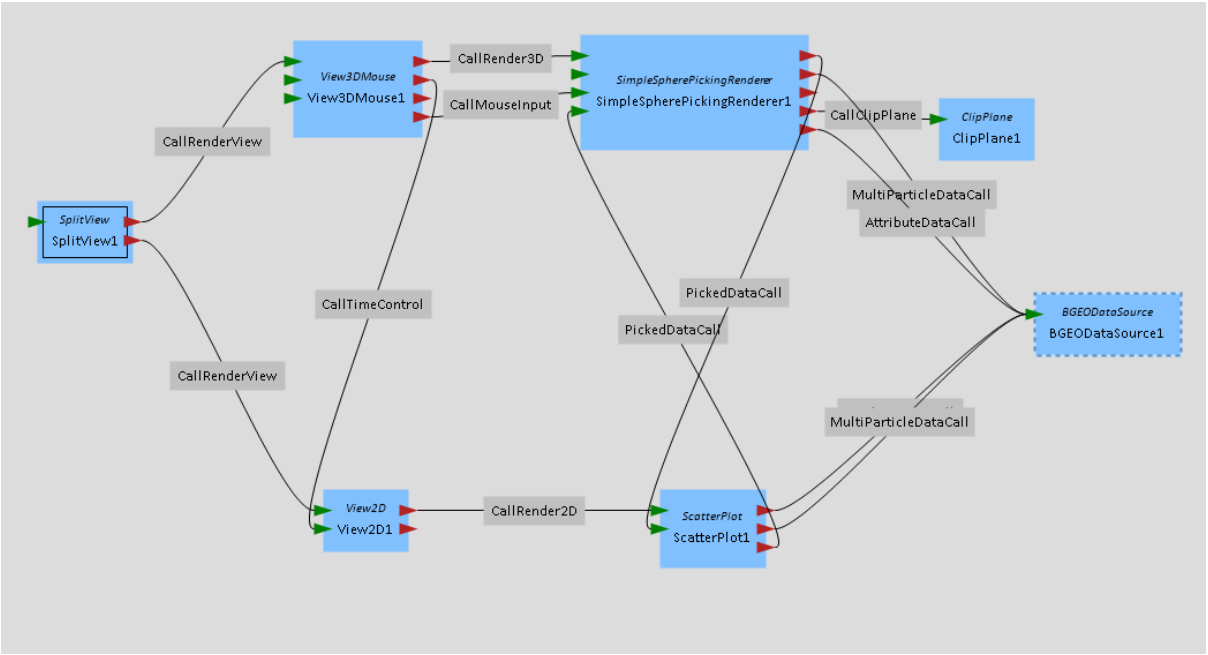

Figure 4.2: Module Graph containing the load Module (on the right), the 3D rendering Module and the Scatter Plot Module (in the middle) and the 2D View and 3D View (on the left).

that sends mouse events to the renderer. The other one is the View2D Module which can only display 2D images. Both viewers have the functionality of a camera view and are part of the Core library. I can synchronize the animation runtime through the Call CallTimeControl which works on the master-slave principle. In this case the View3DMouse Module is the master and controls the runtime for the View2D Module.

Furthermore I will explain the upper branch of the graph. The SimpleSpherePickingRenderer Module is a module developed for rendering the spatial data view. It is an extension from the SimpleSphereRenderer functional component which implements as rendering technique the sphere-rendering using GPU-based ray-casting. What makes the difference between these two modules is the development of a picking technique for the selection of a particle or a desired region of particles. To generate a frame, the View3DMouse requests information via the CallRender3D and sends the mouse information through the CallMouseInput Call. Particle data sets are requested by the rendering Module via the MultiParticleDataCall from the BGEODataSource Module but could be also requested from the ScatterPlot Module along PickedDataCall in case of a selection of particles in the diagram view. In the end, the BGEODataSource is a file loading module that has the main task to import time-dependent particle data from BGEO files and it was developed in a way that respects the MMPLD standard. That is, all Renderer Modules that are

already available in MegaMol and that use the MMPLDDataCall can be used with the new BGEODataSource.

The lower branch of the graph module contains a View2D Module which requests information through the Intent CallRender2D to output an image. The Module ScatterPlot it is derived from a Render2DModule and it is developed to construct a plot or a mathematical diagram using Cartesian coordinates to display values for two variables for particle data sets. Via the MultiParticleDataCall the rendering module can request only the position and the RGBA color for particles. The idea behind the ScatterPlot Module is to display different attributes of the particles and dependencies between them. For this reason I have developed another Call between the rendering module and the loader named AttributeDataCall where all the attributes of a particle may be sent. In case of a selection in the 3DRenderer the ScatterPlot Module will get the data from the SimpleSpherePickingRenderer via the PickedDataCall.

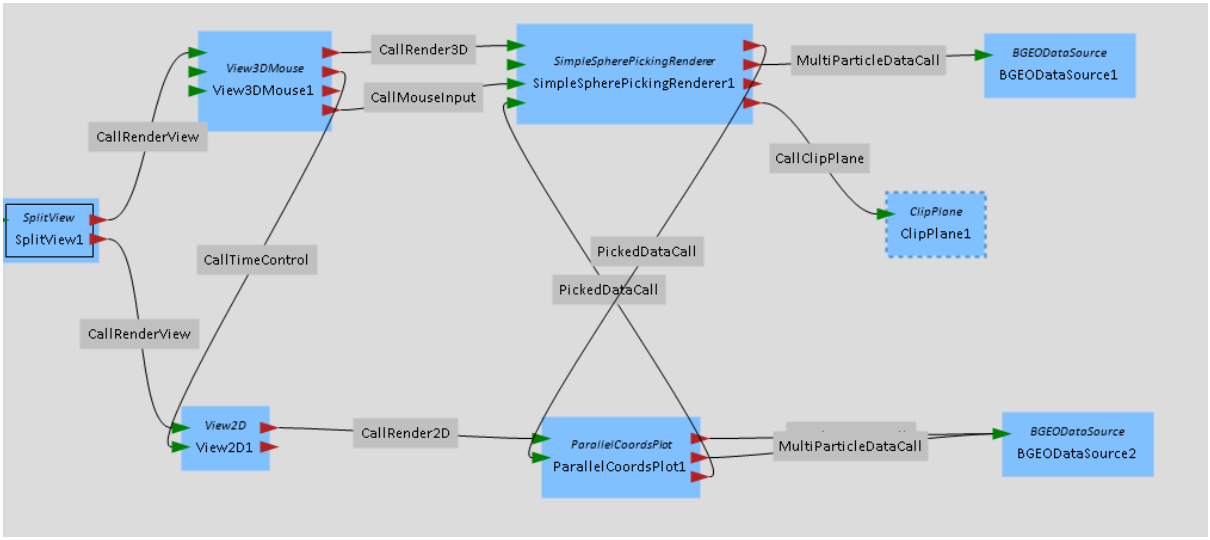

Figure 4.3: Module Graph containing the load Module(on the right), the 3D rendering Module and the Parallel Coordinate Plot

Fig. 4.3 presents the second Module graph. The output image of the 2D View of this graph is a Parallel Coordinate Plot that can be configured to display multiple attributes. The ParallelCoordsPlot is a 2DRendererModule that requests data from the loader or, in case of a selection in the 3DRenderer, from SimpleSpherePickingRenderer.

The output view of the first Module Graph is presented in Fig. 4.4. The left view displays a frame from the animation where the particles are little spheres and only the position of each particle matters. The right view shows a scatter plot that has on the x axis values for the Density attribute and on the y axis values for the Pressure attribute.

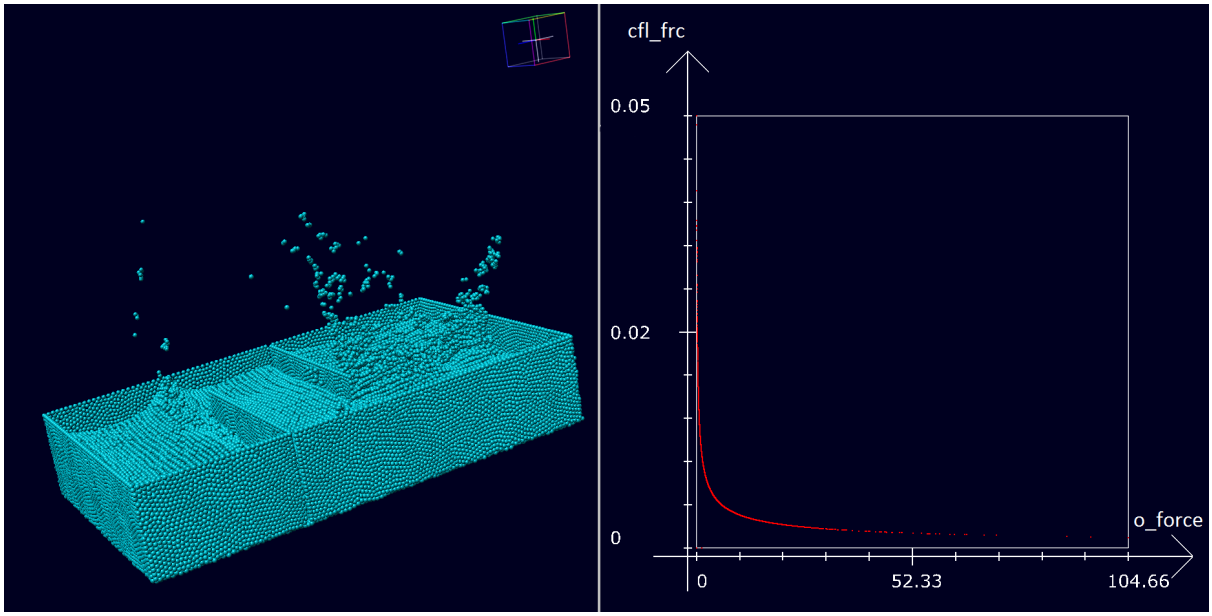

Figure 4.4: Output of the SplitView using 3D rendering Module and the Scatter Plot

4.3 Implementation

I have implemented the visual debugging tool in C++ and it was tested on Microsoft Windows operating system in 64 bit. I developed 4 Modules and 2 Calls in the plugin sphvis as follows:

- BGEODataSource Module
- SimpleSpherePickingRenderer Module
- ScatterPlot Module
- ParallelCoords Module
- AttributeDataCall
- PickedDataCall

1. BGEODataSource Module

   The first Module that I have developed is the BGEODataSource. The main functionality of this Module is to import time-dependent data sets from BGEO files (binary files format for storing Houdini geometry introduced in version 12). Each frame of the data set is stored as a binary file, so for loading the entire data set I have created a parameter that allows you to introduce the path to one file of the concerned simulation and then all the .bgeo files from the same directory are

loaded. Depending on the available memory at runtime a frame cache size is set and only when there is free space in the cache another frame is loaded. For a more accurate visualization the loading module sets a bounding box for the imported data set that can be configured from the Configurator GUI through 2 parameters: maximum and minimum value.

To access the actual information for each particle I have used the Partio library developed by Walt Disney Animation Studios [Par]. Through this library I could read the .bgeo files and get the desired information regarding the number of particles, the number of attributes for each particle and for each attribute the name, type, count and index. Using this information I have created a LoadFrame function that follows the MMPLD standard[File Format Specification MMPLD, Sebastian Grottel, Date: 17.05.2016] regarding the internal data structure for using the MultiParticleDataCall to intercommunicate to other Modules. The downside of using this standard was that I could store only the vertex data type(maximum 4 values: x,y,z coordinates and the radius) and the color data type(maximum 4 color components: RGBA) for each particle.

One solution would be to map one attribute per color component but the number of attributes for simulations that I have used range between 6 and 17. Since I needed all the attributes for the scatter plot and the parallel coordinate plot I have decided to create another call, AttributeDaraCall, through which the rendering modules can get the number of attributes, the name of the attributes and the values for each attribute. The LoadFrame function stores for each attribute only one value regardless of the attribute type, which could be integer, float or vector. In case of a vector attribute (e.g. position, force) the function stores the magnitude of the attribute.

2. SimpleSpherePickingRenderer Module

   The second Module that I have developed is the SimpleSpherePickingRenderer. The rendering technique that I have used is sphere-rendering using GPU-based ray-casting it is also used in the SimpleSphereRendering Module. The notable difference between the Module that I have developed and the SimpleSphereRendering is the possibility to select a particle or a desired region of particles using the mouse control. For this purpose I have created a Renderer3DMouse class that can handle mouse events and the rendering module inherits from this one.

   The selection mechanism that I have developed depends on a boolean array that it is updated in real time. To switch between selection mode and view mode the user has to press the TAB key. Each time the user presses and holds the left button of the mouse the selection mechanism begin. As the user holds down the left button and moves the mouse over the particles, a semi-transparent rectangle is drawn.

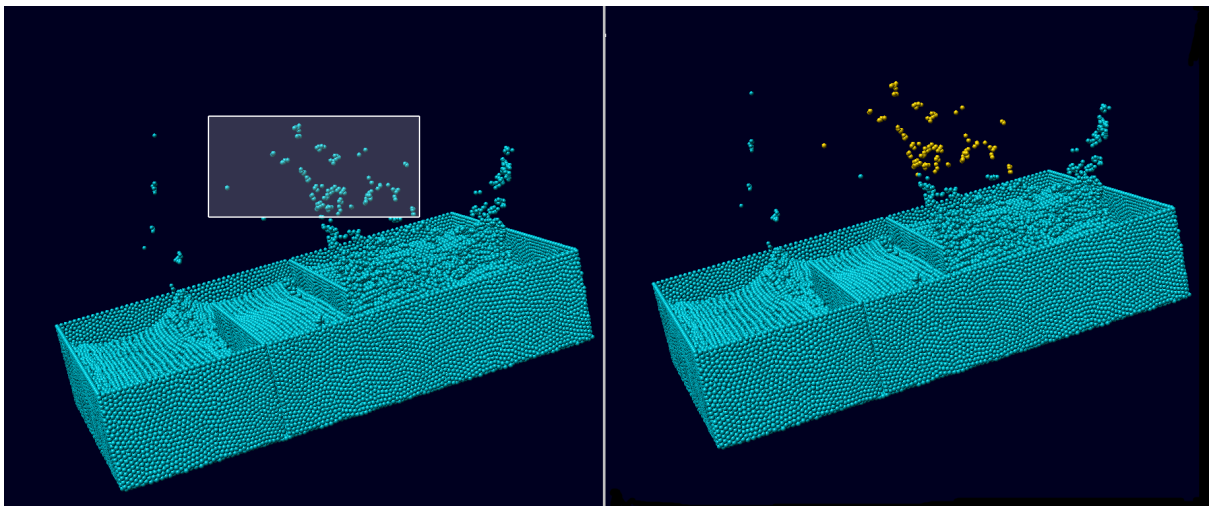

Figure 4.5: Selection mechanism of the SimpleSpherePickingRenderer

When releasing the button, each particle that was inside that rectangle it is marked as "selected" particle in the boolean array. The rendering function than reads the selection array and, depending on the values, sets the color of each particle to green if it is a true value or blue otherwise. Fig. 4.5 presents this mechanism.

The MouseEvent function has as parameters the x and y coordinates of the mouse and the MouseFlags which are defined in MegaMol::Core::View class. Depending on this values I compute the normalized device coordinates of the selected rectangle, taking care that independent of the direction of drawing, the coordinates for the end point of the rectangle are bigger than the start point coordinates. Then I compute the normalized device coordinates for each particle position which is received in object space and compare these values with the rectangle boundaries.

The Render function of SimpleSpherePickingRenderer gets pointers to the two Calls that are connected via the caller slots: MultiParticleDataCall and PickedDataCall. Because the selection mechanism is also available in the other modules, ScatterPlot and ParallelCoordsPlot, these modules could send their own particle selection boolean array to current module. If the pointer to PickedDataCall it is not null, than the Render function reads the particle selection array from the Call and rewrites its own local particle selection array. Depending on the local array the Render function then writes the color of particles to the GPU.

3. ScatterPlot Module

   The ScatterPlot Module implemets a diagram view of the particle data sets that are fetched from the BGEO loader Module via the MultiParticleDataCall. Although this call gives access to the position of the particles and the respective color, the

functionality of the diagram implies the access to all the attributes of each particle. For this reason I have implemented another call named AttributeDataCall that fetches all the names and values for the whole range of attributes.

Since one functionality of the module is to create a plot in 2D, I have decided to inherit the Render2DModule that also facilitates the interaction with mouse events. The rendering of the current frame in the ScatterPlot Module is triggered by the 2D View Module via CallRender2D. The PickedDataCall can connect either to a caller slot or a callee slot of the current module, since the selection mechanism is available in all three rendering modules of the sphvis plugin.

The data points are rendered by the GPU using simple vertex and fragment shaders. Through functions like DrawAxes, DrawTicks or LabelAxes the scatter plot takes shape and appears in the form of an actual mathematical diagram. Each of the x and y axes can be labeled with an attribute name that the user can choose from a drop-down menu. The 2D positions of the data points are computed in the Render function and their values are mapped to the range 0-1 (that is, the position values are normalized).

The other functionality of the module is the possibility to select a region from the scatter plot. This functionality is achieved in the MouseEvent function where the scatter points that fit in the selected rectangle are marked with true in a boolean array that it is local to the ScatterPlot Module. This array is then sent via the PickedDataCall to the rendering modules to render only the particles that correspond.

4. ParallelCoordsPlot Module

   The last module that I have implemented in sphvis plugin is the ParallelCoordsPlot. The connectivity of this module is similar with the one of the ScatterPlot regarding both the transport of particle data sets and the output of the rendering. The parallel coordinates plot is also a 2D plot therefore I have used the same Render2DModule class as parent class for creating the diagram and for selection of one more specific region.

   I have decided to display only 4 vertical axes when I have created the diagram using the DrawAxes function. I think that a larger number of axes would make it more confusing and difficult to observe all of them at once since the human eye tends to focus on groups of two axes if a larger number is used. For each of the 4 axes the user can choose from the parameters menu which attribute to display and of course the axes are interchangeable, so one selection cannot restrict the choosing pool for the others.

The selection mechanism is also available in this module. It was implemented in the MouseEvent function and the selection of one region through the mouse coordinates works between two axes if the user finds an interesting shape or pattern and would like to find out which particles are involved or even across axes and then the user can see particles that have a fixed value for the under-lied attribute.

5 A Study Case and Performance Results

5.1 Identifying stirring boundaries Use Case

In this section I would like to present a use case where the debugging tool can have great impact on exploring the simulated data-set. As I have mentioned in previous chapters, the tool can import different Houdini simulations, but now I would like to explain the utility of the visualizations over the data-set "async-pool". Fig. 3.10 shows a 3D visualization of the particles from the data-set whose color map represents the values of the particle type: blue is fixed boundary, cyan is stirring boundary and pink is the fluid itself. By visualizing this feature, the user can have an overview of the simulation, but combining it with the scatter plot visualization will allow the data analyst to drill-down and access details of the data.

The present Use Case presents a method to identify stirring boundaries from all the particles that are displayed. Inspecting the first frame of the simulation, the user can say that the propeller is not a fixed boundary since its function is maybe to spin the fluid particles. This is correct, but he/she needs a higher degree of confidence in these findings. Using the linking and brushing technique between the scatter plot and the 3D viewer the user can really rely on the visualization output.

Fig. 5.1 displays the particles in 3D and the scatter points in the 2D diagram where the x axis represents the density attribute and the y axis represents the velocity attribute. Because I am analyzing the first frame of the animation, most of the particles have velocity near to 0 and density around 500. As can be depicted from the scatter plot there are clouds of scatter points in the lower left corner of the diagram but also outliers towards maximum velocity and maximum density.

When selecting the scatter points with maximum velocity, in the 3D visualization the propeller tips are marked as selected particles and displayed with a different color(green). There is also a physical explanation of this visualization since the propeller spins the tip has higher velocity than the rest of the blade. In addition, when I select another rectangle where I include scatter points with smaller velocity, but above the dense zone, the fans of the propeller are getting green starting from the tip to the root because the velocity is decreasing in this manner.

By using the clipping plane feature of the 3D viewer I could find out that there is a second propeller in the simulation which was hide by fluid particles. By slicing the simulation along y axis I could obtain a good visualization that contains both stirring boundaries of the simulation shown in Fig. 5.1.

There is still a missing part of the propeller in order to say that the detection of the kinematic boundary is working properly. The spinner/hub of the propeller is a fixed part so the velocity is near to zero but the main aspect of the hub is its high density. Since in the first frame all the fluid particles have small densities, the easiest way to identify the spinner is to select the outliers towards high density values as shown in 5.2.

To conclude, the detection of stirring boundaries can be easily achieved by selecting both regions with scatter points outside the dense region in the velocity-density scatter plot. Picking the upper left area will provide the particles corresponding to the blades of the propeller and the lower right region will afford the selection of the spinner. Fig. 5.2 presents the detection of both parts, blades and spinner.

During the tests that I have made for the above mentioned study case, I have found out that the fluid particles are divided into 3 categories as shown in Fig. 5.3. By selecting the cluster corresponding to density values around 900 and using a clip plane on x and y axes I can display in the 3D view the inner particles of the fluid. The second class of particles are the particles that represent the surface of the liquid. These particles have the density value around 700. The last cluster, with lower values of the density attribute, represents the edge particles of the fluid.

5.2 Performance results

The performance tests were made on a desktop with the following configuration:

- Processor: Intel Core i5-4690 CPU @ 3,50GHz
- Installed memory (RAM): 8.00 GB
- GPU: Nvidia GeForce GTX 970
- System type: 64-bit, Windows 10 Operating System

Table 5.1 presents evaluations of the module graph presented in Fig. 4.3 where the output visualization contains the 3D simulation of particles and the ParallelCoordinatesPlot. The tests were made on 3 different data sets: the "8K cube" data set represents a cube containing 8000 fluid particles, the "async_pool" data set consists of 90880 particles of 3 different types: fixed boundary particles, stirring boundary particles and the fluid

particles. The last data set, "brd_with_col_obj", has 201180 particles that also include 3 collision objects.

The first three lines from Tabel 5.1 emphasize the meta-information of the data sets such as the number of attributes for each particle, the number of particles and the number of frames for each simulation. The rest of the rows present performances of the visual debugging tool when loading or running the simulation data sets. The FPS represents the rendering speed of the 2D/3D modules. The BGEO loading speed is measured in ms/frame and represents the time it takes to load a frame (one frame equals one BGEO file).

	$8Kcube$	$asyncpool$	$brdwithcolobj$
Nr. of attributes	8	17	7
Nr. of particles	8000	90880	201180
Nr. of frames	1501	751	501
FPS when loading	235	23	11
FPS when running	240	23	11
CPU usage	34 %	31.5 %	32 %
BGEO loading speed	92.17	93.20	90.32

Table 5.1: Meta-information and performance results when visualizing the 3D simulation and the Parallel Coordinates Plot.

Table 5.2 presents evaluations for the same data sets but now the 2D view shows a Parallel Coordinates Plot. There are small differences regarding the rendering speed on loading or running the simulation data set but there is a major difference for the first data set with respect to the BGEO loading speed. When visualizing the $ParallelCoordsPlot$ the BGEO loading speed is 92.17 ms/frame in contrast with plotting scatterpoints where the loading speed is 3ms/frame.

	$8Kcube$	$asyncpool$	$brdwithcolobj$
FPS when loading	265	27	13
FPS when running	265	10	13
CPU usage	33 %	53 %	55 %
BGEO loading speed	3.30	87.37	87.25

Table 5.2: Performance results when visualizing the 3D simulation and the Scatterplot.

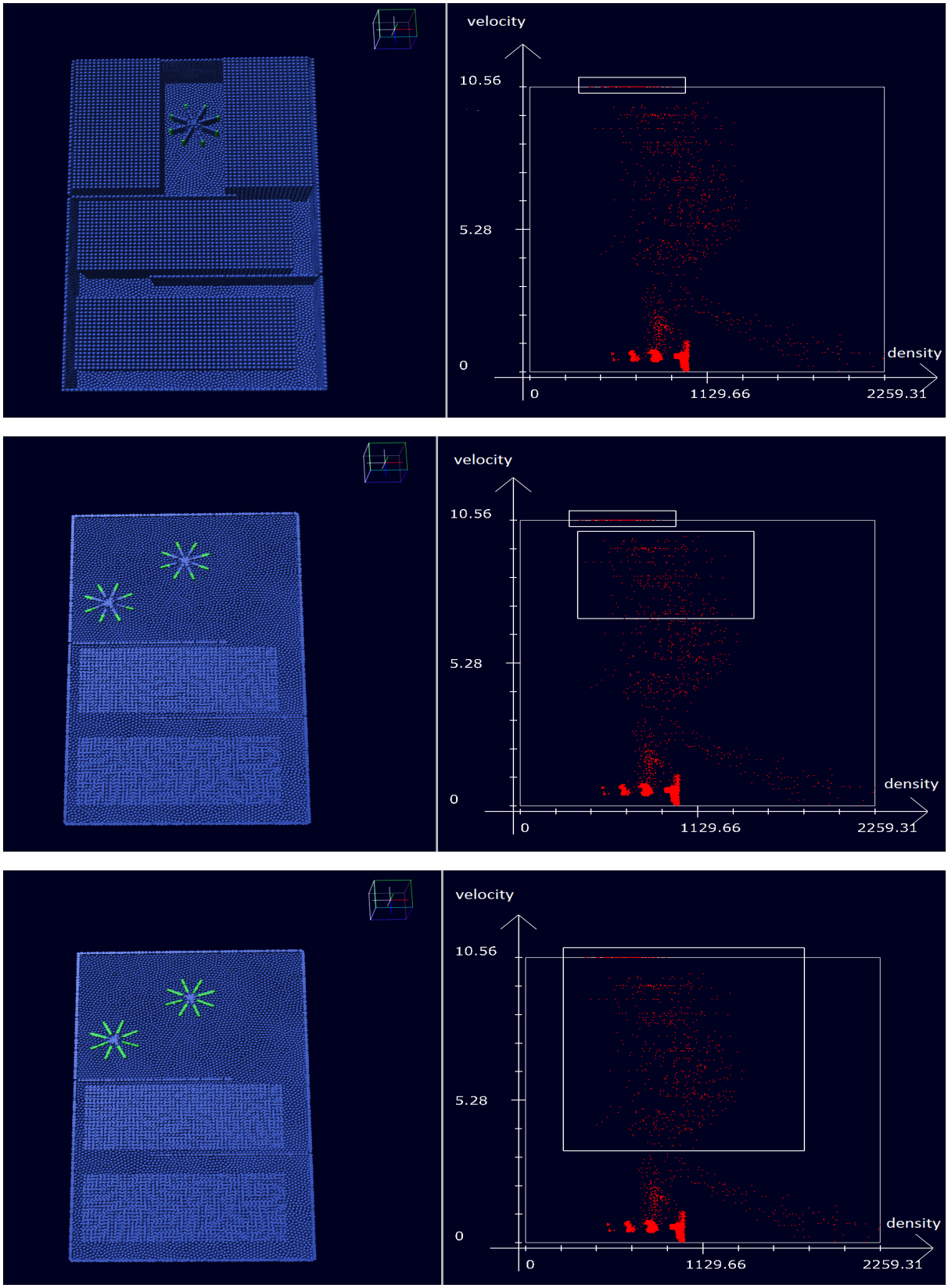

Figure 5.1: Detection of the propellers blades using linking and brushing technique

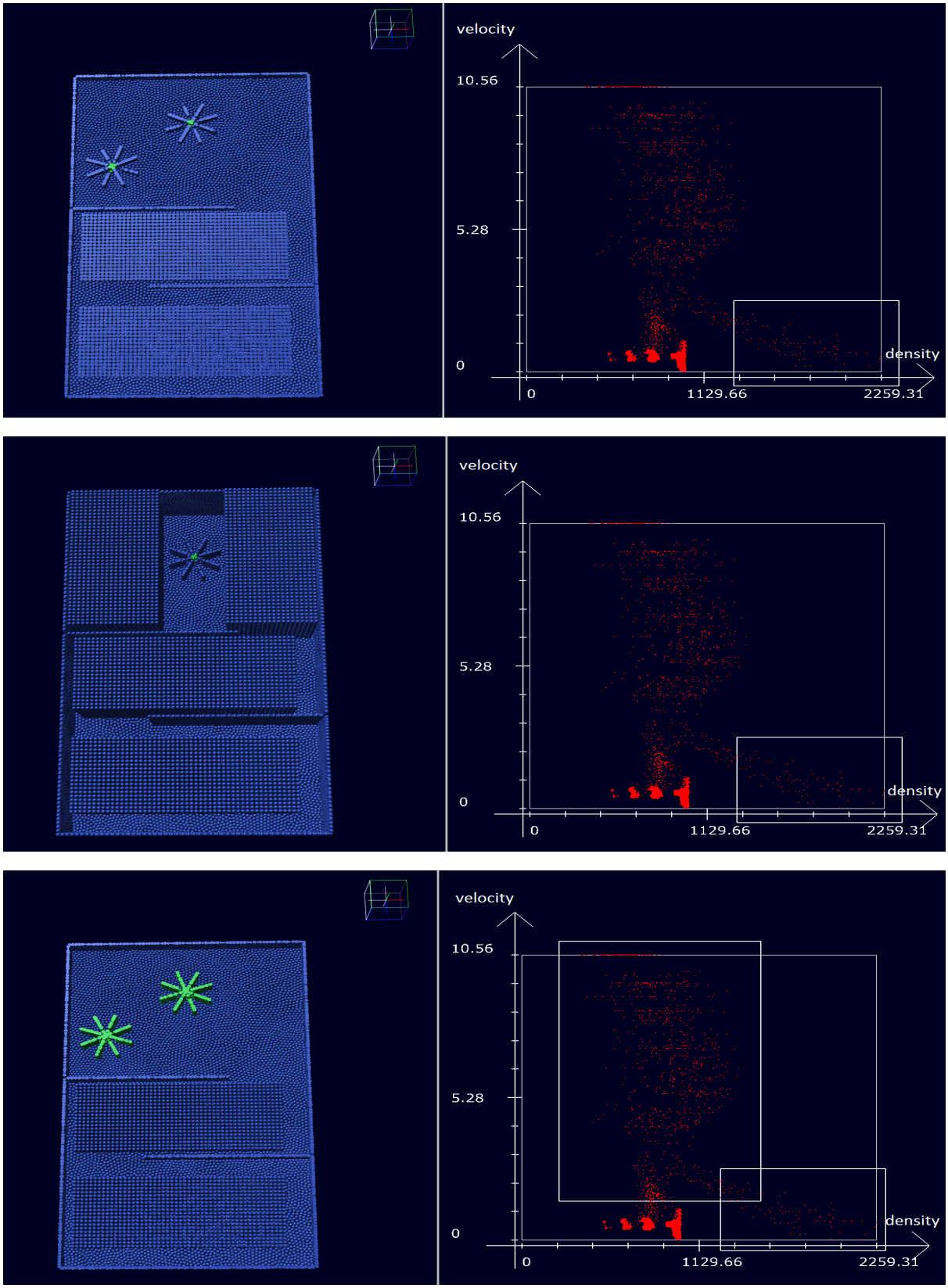

Figure 5.2: Detection of the propeller's spinner and the propeller entirely using linking and brushing technique.

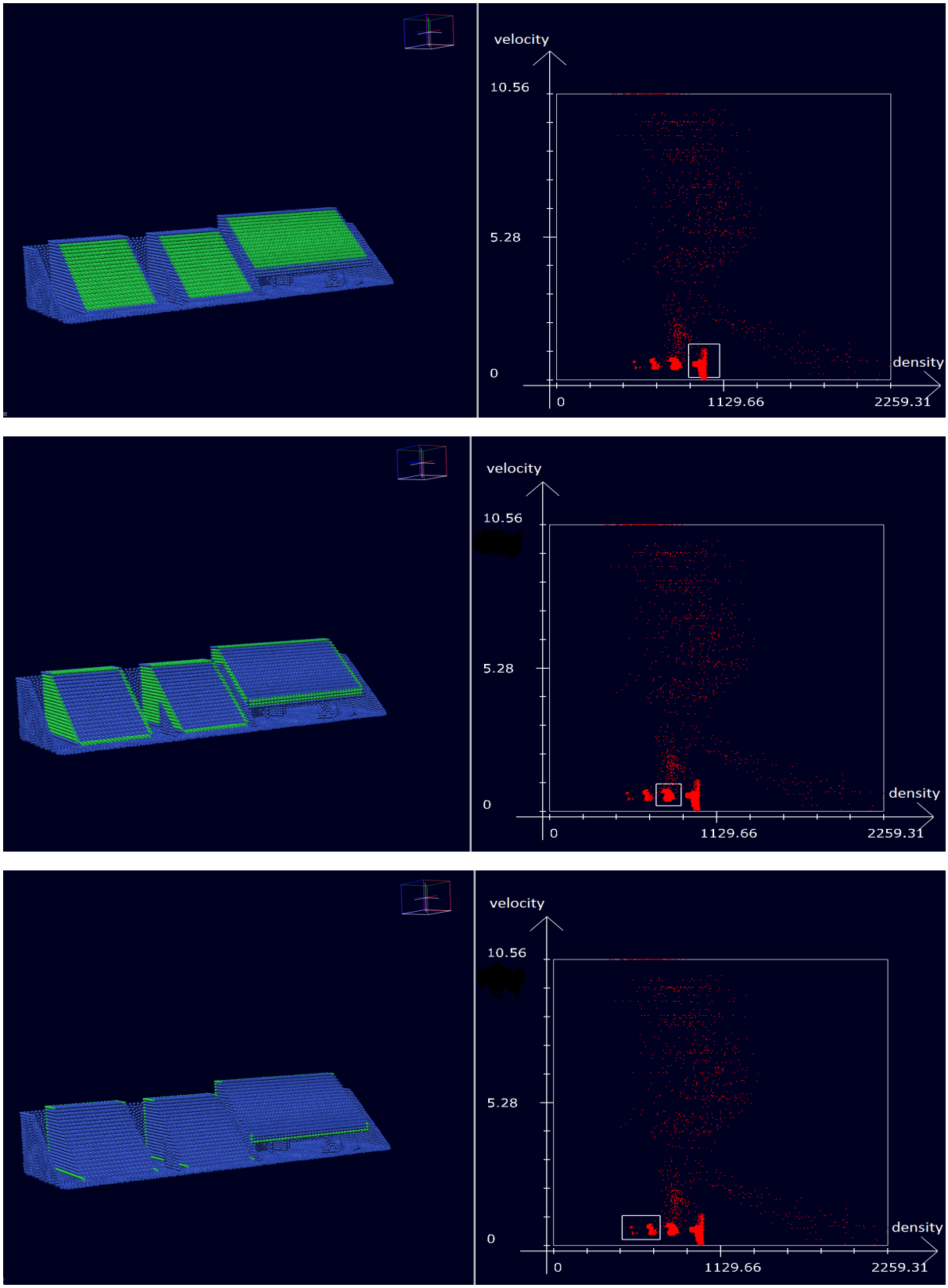

Figure 5.3: The fluid particles division: inner particles, surface particles and edge particles.

6 Conclusions

6.1 Summary

Particle-based simulations of fluids require specialized tools in order to get an insight and really understand the phenomena that is simulated. There are challenges both in the visually pleasing and the computational efficiency aspects for researchers in the field of computer graphics when they use this approach.

The current thesis presents a visual debugging plug-in for Particle-based simulations of fluid developed in MegaMol environment. The visualization results obtained by using this plug-in can help the user better comprehend simulated scenario, identify possible errors in the simulation and understand the model to develop new techniques.

I have implemented four modules in order to augment the MegaMol functionality and with no significant penalty in performance. The visualization techniques that I have used, standard 2D/3D displaying and geometrically-transformed Display, are implemented in $ScatterPlot$ and $ParallelCoordPlot$. There are different interactions possible between these modules and the 3D rendered scene facilitated through the brushing and linking technique.

In the $ScatterPlot$ particles can be selected according to different properties and displayed with a contrasting colour in the 3D viewer. This helps to highlight area of interest in the space of two chosen attributes.

$ParallelCoordPlot$ helps us view the data set from another perspective. By following each particle from one parallel axis to another we can see areas of the attribute space where most of the particles cluster or we can detect outliers in the data set. To reduce clutter, due to the high number of lines, the user can choose to see just a fraction of the total information.

$BGEODataSource$ adds a new type of file format to the MegaMol environment. This helps translating the Houdini geometry format into MegaMol Particle List Data structure

6.2 Directions for future work

The $ScattePlot$ and the $ParallelCoordPlot$ have a limitation when it comes to high amounts of data. This comes from the fact that areas with a high point/line density are hard to compare to other high density regions. As future work I would like to also add density information of the area so the density of the region will be mapped to color.

Another improvement of the plug-in will be to render in the 3D viewer only the selected particles from the $ScattePlot$ or the $ParallelCoordPlot$. The MegaMol environment offers the possibility to clip planes in the simulation in order to explore occluded areas, but I think that visualizing only the selected particles can help the user to understand in more detail the simulated fluid.

Bibliography

[BT07] M. Becker, M. Teschner. “Weakly compressible SPH for free surface ows.” In: *In Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation* (2007), pp. 209–217 (cit. on p. 12).

[BTT09] M. Becker, H. Tessendorf, M. Teschner. “Direct forcing for Lagrangian rigid-uid coupling.” In: *IEEE Transactions on Visualization and Computer Graphics* 15 (May 2009), pp. 493–503 (cit. on p. 11).

[CGFO06] N. Chentanez, T. G. Goktekin, B. E. Feldman, J. F. O’Brien. “Simultaneous coupling of fluids and deformable bodies.” In: *In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), pp. 83–89 (cit. on p. 10).

[Ell08] G. P. Ellis. “Random Sampling as a Clutter Reduction Technique to Facilitate Interactive Visualisation of Large Datasets.” PhD thesis. Computing Department, Lancaster University UK, Sept. 2008 (cit. on p. 31).

[ESHD05] K. Erleben, J. Sporring, K. Henriksen, H. Dohlmann. *Physics-Based Animation*. Charles River Media, 2005. ISBN: 1584503807 (cit. on p. 13).

[GKMRE15] S. Grottel, M. Krone, C. Muller, G. Reina, T. Ertl. “MegaMol—A Prototyping Framework for Particle-Based Visualization.” In: *Annual Review of Astronomy and Astrophysics* 21.2 (Feb. 2015), pp. 201–214 (cit. on pp. 35–37).

[GM77] R. A. Gingold, J. J. Monaghan. “Smoothed Particle Hydrodynamics: theory and application to non-SPHerical stars.” In: *Monthly Notices of the Royal Astronomical Society* 181 (1977), 375–389 (cit. on p. 14).

[GRDE10] S. Grottel, G. Reina, C. Dachsbacher, T. Ertl. “Coherent culling and shading for large molecular dynamics visualization.” In: *Proc. Comput. Graph. Forum* 29 (2010), 953–962 (cit. on p. 36).

[Gum03] S. Gumhold. “Splatting illuminated ellipsoids with depth correction.” In: *in Proc. VMV* (2003), 245–252 (cit. on p. 36).

[HW13] J. Heinrich, D. Weiskopf. *State of the Art of Parallel Coordinates*. Tech. rep. 2. Eurographics. The address of the publisher: Visualization Research Center, University of Stuttgart, July 2013 (cit. on p. 23).

[Ins09] A. Inselberg. *Parallel Coordinates*. Springer, 2009. ISBN: 978-0-387-68628-8 (cit. on pp. 12, 23).

[KAGBDG05] R. Keiser, B. Adams, D. Gasser, P. Bazzi, P. Dutre, M. Gross. "A unied Lagrangian approach to solid-uid animation." In: *In Proceedings of the Eurographics Symposium on Point-Based Graphics* (2005), pp. 125–133 (cit. on p. 11).

[Kei02] D. A. Keim. "Information Visualization and Visual Data Mining." In: *IEEE Transactions on visualization and computer graphics* 7.1 (Jan. 2002), pp. 101–107 (cit. on pp. 19, 20).

[Kel06] M. Kelager. "Lagrangian Fluid Dynamics Using Smoothed Particle Hydrodynamics." Graduate project. MA thesis. Universitetsparken 1, DK-2100 Copenhagen, Denmark: The school of the thesis, Jan. 2006 (cit. on pp. 10, 13, 14, 16, 18).

[LP02] A. T. Layton, M. van de Panne. "Numerically Efficient and Stable Algorithm for Animating Water Waves." In: *The Visual Computer* 18.1 (Jan. 2002), pp. 41–53 (cit. on p. 11).

[LSSF06] F. Losasso, T. Shinar, A. Selle, R. Fedkiw. "Multiple interacting liquids." In: *ACM Trans. Graph. (SIGGRAPH Proceedings)* 25 (July 2006), pp. 812–819 (cit. on p. 10).

[Luc77] L. B. Lucy. "A numerical approach to testing of the ssion hypothesis." In: *Astronomical Journal,* 82 (1977), 1013–1024 (cit. on p. 14).

[MCG03] M. Muller, D. Charypar, M. Gross. "Particlebased fluid simulation for interactive applications." In: *In Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation* (2003), pp. 154–159 (cit. on pp. 11, 17, 18).

[MCPN08] J. Molemaker, J. Cohen, S. Patel, J. Noh. "Low viscosity flow simulations for animation." In: *In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (July 2008), pp. 9–18 (cit. on p. 11).

[Meg] *MegaMol*. https://svn.vis.uni-stuttgart.de/trac/megamol. Accessed: 2016-07-1 (cit. on p. 36).

[Mon92] J. J. Monaghan. "Smoothed Particle Hydrodynamics." In: *Annual Review of Astronomy and Astrophysics* 30 (1992), pp. 543–574 (cit. on p. 15).

[Mon94] J. J. Monaghan. "Simulating free surface ows with SPH." In: *J. Comput. Phys.* 110 (1994), pp. 399–406 (cit. on pp. 12, 17).

[MSKG05] M. Muller, B. Solenthaler, R. Keiser, M. Gross. "Particle-based uid-uid interaction." In: *In Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation* (2005), pp. 237–244 (cit. on p. 11).

[Par] *p*. https://www.disneyanimation.com/technology/partio.html. Accessed: 2016-08-3 (cit. on p. 41).

[SML96] W. J. Schroeder, K. M. Martin, W. E. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3-D Graphics*. Prentice Hall PTR, 1996. ISBN: 0131998374 (cit. on p. 35).

[Sol10] B. Solenthaler. "Incompressible fluid simulation and advanced surface handling with SPH." PhD thesis. Strickhofstrasse 39 CH-8057 Zurich: Faculty of Economics, Business Administration and Information Technology of the University of Zurich, Jan. 2010 (cit. on pp. 9–11, 14, 17).

[SP09] B. Solenthaler, R. Pajarola. "Density contrast SPH interfaces." In: *In Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation* 18 (2009), pp. 211–218 (cit. on p. 11).

[SSP09] B. Solenthaler, J. Schai, R. Pajarola. "A unied particle model for uid-solid interactions." In: *Journal of Computer Animation and Virtual Worlds* 18 (2009), pp. 69–82 (cit. on p. 11).

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature