TECHNISCHE UNIVERSITÄT DRESDEN

Computer Graphics and Visualization

Faculty of Computer Science

Institute of Software and Multimedia Technology
Chair of Computer Graphics and Visualization

Bachelor Thesis

Occlusion Avoidance for Immersive Inspection of 3D Cell Complexes and Cell Surfaces

Joris Grau

Born on: 6th January 2001 in Dresden
Matriculation number: 4901060

18th December 2022

First referee
Prof. Dr. Stefan Gumhold

Second referee
Dr. Dmytro Shleszinger

Supervisor
Dipl.-Inf. Franziska Kahlert

TECHNISCHE UNIVERSITÄT DRESDEN

Computer Graphics and Visualization

Faculty of Computer Science Chair of Computer Graphics and Visualization

Task for the preparation of a Bachelor Thesis

Course:	Computer Science
Name:	Joris Grau
Matriculation number:	4901060
Matriculation year:	2019
Title:	Occlusion Avoidance for Immersive Inspection of 3D Cell Complexes and Cell Surfaces

Objectives of work

Discrete simulation models for the formation of cellular tissue has matured to a state where large datasets with high spatial and temporal resolution can be generated based on a description of the simulation model. Inspection of this data is difficult due to the dense packing of cells leading to severe occlusion problems. Especially, for surface properties standard techniques like slicing does not help. The goal of this thesis is to design and implement a VR-based visual analysis tool that exploits and potentially extends known occlusion avoidance techniques like X-Ray views or explosion views to provide visual access to cell surfaces.

Focus of work

- Literature research on occlusion avoidance in real-time setting with static and dynamic 3D environments as well as corresponding interaction design.
- Concept design for interactive cell complex exploration in VR with direct interaction and time control.
- Import of Dataset provided by collaboration partners and preprocessing necessary for real-time interaction
- Concept for implementation including informed selection of VR framework
- Prototypical implementation of immersive exploration concept
- Evaluation of performance and with expert feedback

Optional tasks

- Visualization of attributes on cell surfaces
- User study to determine usability score

TECHNISCHE UNIVERSITÄT DRESDEN

Computer Graphics and Visualization

Faculty of Computer Science Chair of Computer Graphics and Visualization

First referee:	Prof. Dr. Stefan Gumhold
Second referee:	Dr. Dmytro Shleszinger
Supervisor:	Dipl.-Inf. Franziska Kahlert
Issued on:	4th October 2022
Due date for submission:	19th December 2022

Prof. Dr. Stefan Gumhold
Supervising professor

Statement of authorship

I hereby certify that I have authored this document entitled *Occlusion Avoidance for Immersive Inspection of 3D Cell Complexes and Cell Surfaces* independently and without undue assistance from third parties. No other than the resources and references indicated in this document have been used. I have marked both literal and accordingly adopted quotations as such. There were no additional persons involved in the intellectual preparation of the present document. I am aware that violations of this declaration may lead to subsequent withdrawal of the academic degree.

Dresden, 18th December 2022

Joris Grau

Abstract

In this work, it is examined how explosion views can be utilized to avoid occlusion during immersive cell exploration in virtual space. To achieve this, a literature review of related work is conducted and four promising methods are derived. Furthermore, it is tested how time-varying data can be displayed with explosion views and which interaction possibilities of the Virtual Reality are useful for the exploration of these. The findings are then implemented and compared in a prototype. The usability and effectiveness of the methods is then evaluated by two experts and a performance analysis of the prototype is made.

Contents

Symbols and Acronyms

VR	Virtual Reality
AR	Augmented Reality
GPU	Graphics Processing Unit
CAD	computer-aided design
XML	Extensible Markup Language
UI	User Interface
HCI	Human Computer Interaction
NURBS	Non-Uniform Rational B-Spline

1 Introduction

The human brain consists of 86 billion neurons. If one wanted to pick out, visualize and examine just a tiny fraction of these, it would be very difficult to make any meaningful conclusions, as the sheer volume of data prevents close inspection.[Pei15] Nevertheless, correct and precise analysis of datasets is a cornerstone of any research, but this is often challenging, especially as the complexity and size of the dataset increases. A common problem with such large data sets is that the data you are trying to examine is obscured and blocked by other data, this is especially the case with three dimensional data sets and is called occlusion. Therefore, it is important to develop methods and programs that filter or transform data sets and allow a closer look at individual parts of the data set as well as their context. There are many different techniques to avoid and reduce occlusion. These can roughly be divided into two types: hiding unimportant data and transforming the data set so that occlusion is reduced or completely avoided. For example, it is possible to use clipping planes to hide foreground data. A common problem with using them is that the context of the data is difficult to recognize, because only a part of the whole is shown. To reduce this, the position of individual data can be changed so that no more occlusion occurs, but all data is still displayed. One way of transforming the data set is through the use of exploded views. Here, individual parts of a model or data set are pulled apart in such a way that each part of interest can be viewed in detail and the original composition of the data set remains recognizable.

The goal of this work is to examine and compare different methods for generating such exploded views for cell complexes. Furthermore, it will be tested which of these methods are suitable for inspection in virtual reality (VR) and what new possibilities and difficulties this innovative environment brings. Since the immersive visualization of such models in virtual reality and its intuitive interaction creates completely new possibilities for exploded views, it is important to find out which techniques are effective and which established methods are not suitable. To do this, an analysis of the state of research and a classification of the different methods will be developed, followed by a prototypical implementation of some selected techniques for exploded views and a comparison of their effectiveness in the virtual space. For this purpose, a data set of a cell complex is available which is first visualized and then transformed. This dataset was simulated using an external tool and it describes the change of the cell complex over a defined period of time. Therefore it is necessary that the

visualization and the explosion view can also show the temporal change of the individual cells of the cell complex. Furthermore, different interaction types are to be tested.

In order to achieve this, the work is divided into the following structure. First, general terminologies and problems of exploded views are explained, then a thorough literature analysis takes place on related works and their solution approaches on the subject of exploded views and occlusion avoidance of dynamic and static data sets. It continues with a more precise classification of the topic and the solution approaches that are pursued in this work. Then the results are shown and explained in a subsequent discussion, followed by an analysis by two external experts on the subject. At the end there is a summary of the work as well as further approaches that could be explored in a future work.

2 Background

When examining complex data sets or models, it is often important to understand the composition of the individual components and their order of assembly. This is especially the case for technical and biological models, where the order of composition defines the function. A problem that often arises is how to inspect the inner parts of such a model without breaking the compositional order. One option is to use filters and thus cut away the occluding parts, but this is done at the expense of clarity and can make it difficult for the viewer to correctly identify the position of the part in the object and gain an understanding of the overview. To avoid this, this work focuses on exploded views, as they offer the possibility to visualize internal parts of complex models while maintaining their spatial relationships with other parts. Furthermore, they offer good interactive possibilities, which can also be transferred to the interaction in virtual reality.

Exploded views are drawings or information graphics that pull apart a complicated object as if it were blown up in a controlled manner. The individual parts are then spatially separated so that their position inside the object or model becomes visible. Often the projection of the object takes place in a planar view from the side or from slightly above. This type of representation has a long history and can be found in many technical drawings, as the individual parts can be named and the order of assembly becomes clearly recognizable.

Looking at traditional hand-drawn explosion views, it becomes apparent that both the explosion direction in which the parts are moved and the spacing between the parts must be well chosen. In order to create the most informative explosion views, Li et al.[Li+08] describe five additional desirable conventions in their paper. They can also be seen in figure 2.1.

- **Blocking constraints:** Parts should be translated so that they do not pass through other parts, this is to show the viewer how the components fit together and indicate their relative position.
- **Visibility:** The spacing of all parts should be chosen so that each part of interest is visible.
- **Compactness:** Parts should be moved as little as possible from their original position to facilitate the viewer's mental reconstruction.
- **Canonical explosion directions:** Most objects have a canonical direction, i.e. an axis in which the object can best be aligned after the explosion. This is determined by various

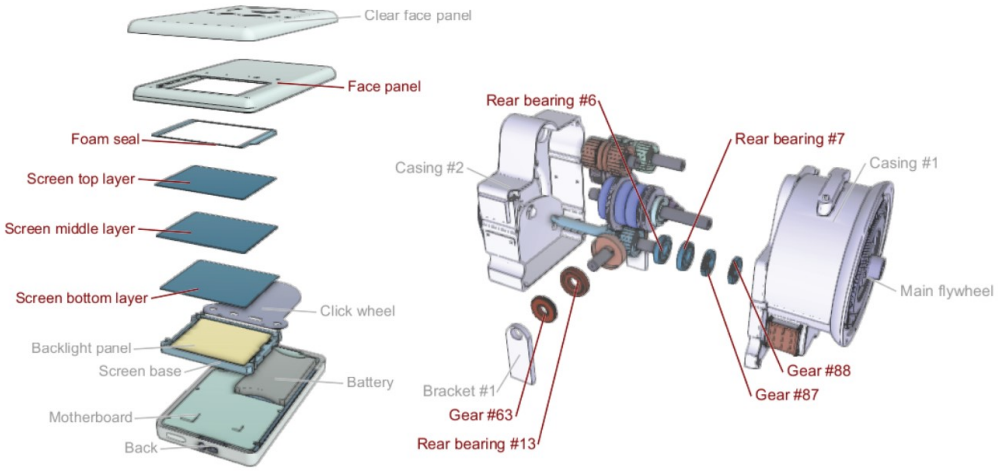


Figure 2.1: Exemplary exploded views generated with Li et al.'s system. On the left an exploded view of an iPod, on the right that of a transmission. Parts of interest are marked in red.[[Li+08], Figure 10]

object-specific properties. In order to reduce the visual clutter, only a few of these axes should be chosen, otherwise the viewer might have difficulty recognizing the original composition.

- **Part hierarchy:** Complex models are sometimes divided into different exploded views and axes, this can illustrate the composition of smaller parts of the object.

If internal parts of an object are completely enclosed, it is necessary to either remove or cut open the outer part. According to Li et al., it should be ensured that when the outer object is cut, the distance with which the outer parts are pulled apart is kept as minimal as possible, whilst still leaving the inner parts visible.[Li+08] In such cases it is also common to hide the outer parts to reveal the inner composition as it can be seen in figure 2.2.

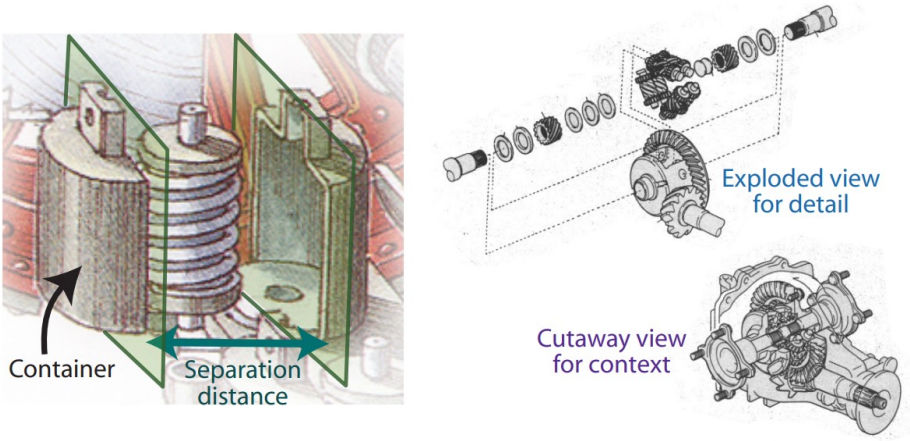


Figure 2.2: Ways to separate containers. The container is split on the left and shown as a contextual cutaway on the right.[[Li+08], Figure 3]

A distinctive feature which is relevant for this work are the utilized data sets. They represent the growth of an embryo, which begins with a few stem cells, which divide and reproduce over the course of the simulation. This creates new cells and cell populations. They have been

generated using the program *Morpheus*[1] which is being developed at the TU Dresden.[Dre22] *Morpheus* can be used to simulate and visualize the temporal changes of cell complexes and their evolution. The datasets thus represents a biological model in which there is no fixed order of composition and the individual parts may change their position and size over time. Figure 2.3 shows a two-dimensional view of the data set at three time steps, which is created by Morpheus. Here it can be seen how the cell structure changes over time and how two populations form at a later time step. These are colored with blue and green. Figure 2.4 shows a three-dimensional visualization of another dataset that was created with *Morpheus*. This was created with ParaView[2].

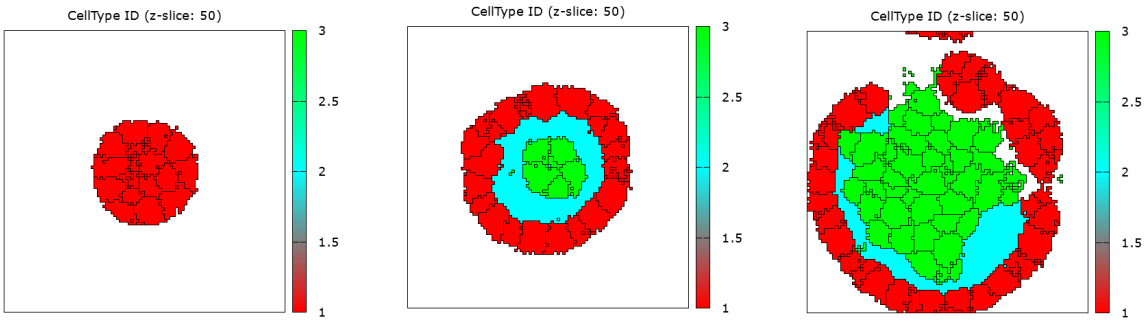


Figure 2.3: Two-dimensional data set visualization, which was created with *Morpheus* at three different time steps t. Left: t=100, Middle: t=1500, Right: t=3000.

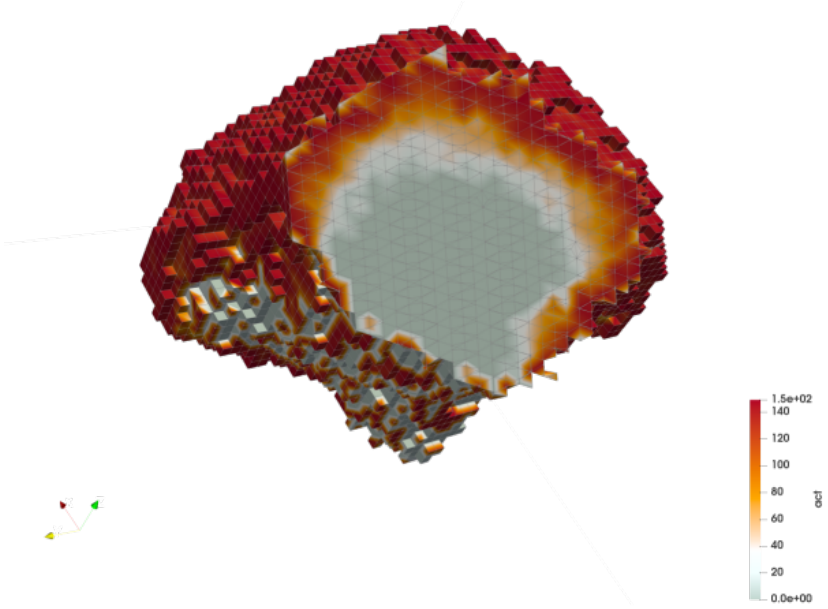


Figure 2.4: 3D visualization of a Hapatocyte (liver cell) dataset which was simulated using Morpheus and created with ParaView.

[1] Morpheus website: https://morpheus.gitlab.io/
[2] ParaView website: https://www.paraview.org/

3 Related Work

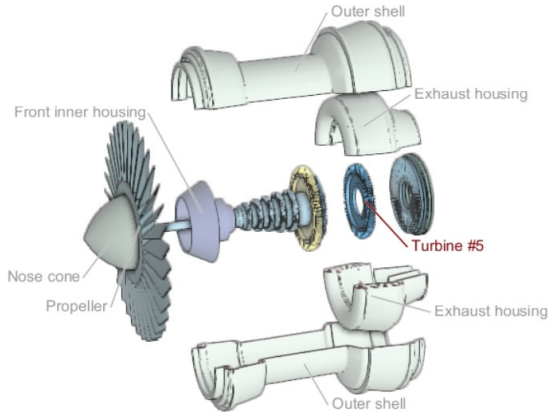
Occlusion avoidance is a well-studied area of computer visualization that is constantly evolving as technology advances. Innovative ways to expand and enhance existing concepts and techniques are made possible by new devices and technologies. This is especially true for visualization methods like cutting planes and exploded views, which greatly profit from these novel interaction possibilities.

Li et al. provide an example of this.[LAS04] In their work, they demonstrate an algorithm that automatically separates traditional drawn explosion views to make them interactive. Their algorithm takes 2D images of explosion view diagrams and automatically cuts them apart, both reducing visual clutter and clarifying the spatial separation of the individual components. Furthermore, the separated parts can be labeled more precisely and retracted and extended as needed. Figure 3.1 illustrates this approach.

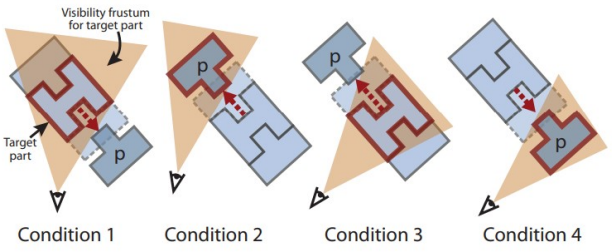
Figure 3.1: Interactive car diagram in its fully expanded (left) and fully collapsed configurations (right). [[LAS04], Figure 16]

Since the interaction possibilities of two-dimensional images are limited and data sets and models have become increasingly complex, the question arises as to how the same principles of exploded views can be transferred to three-dimensional space. For this purpose, Mohammad et al.[MK93] developed a tool that creates exploded views for three-dimensional CAD models. It shows both precise spatial relationships and the order in which the object was assembled. This is especially useful when visualizing machines and technical objects, as it gives the viewer a clear idea of how the parts are arranged. One disadvantage of their implementation is that the individual relations must be clearly defined by a designer beforehand in order to generate the explosion view and calculate the position of the exploded parts. As a result, the order of composition and the blocking elements must be known and manually defined.

Li et al. therefore presented a system that automatically extracts non-blocking exploded views from a 3D model, focusing on rearranging parts instead of hiding obscuring geometry.[Li+08] They also provide a list of tools to interact with the exploded views and dynamically select and show parts of interest. Their implementation works for both hierarchical and non-hierarchical



(a) Interactive illustration generated by the system presented by Li et al.[[Li+08], Figure 1]



(b) Conditions for moving part p. For each condition, the target part is outlined in red. The orange visibility frusta show how unwanted occlusions have been eliminated in each case. [[Li+08], Figure 8]

models, which also allows it to process biological datasets where there is no fixed assembly order. An example of an exploded Turbine can be seen in figure 3.2a. The algorithm works by calculating an explosion graph when loading the model, which describes the blocking elements of each part from different angles. This allows to retrieve at runtime the sequence of elements needed to disassemble the object without parts passing through others. Thus a dynamic explosion graph can be generated which shows an animated composition from all viewing directions. An important part of this is the generation of a correct explosion graph. To accomplish this, two problems have to be solved: first, how to move the parts to uncover the target parts without occluding them, and second, how to deal with enclosed parts. Li et al. solve this problem by iteratively going through all the parts and testing for two conditions: each part must be moved so that none of the target parts are obscured. Figure 3.2b illustrates these steps. In order to isolate target parts from other touching parts, it is also made sure that they are completely visible and close parts are moved further away. If one part is completely enclosed by another, the outer one is separated in the bounding box center and pulled apart so that the inner parts are completely visible, then the algorithm continues. This can be seen in figure 3.3 The resulting application generates animated exploded views for models with up to fifty parts. However, a disadvantage of this implementation is that it only works for static data sets and does not provide any solution for time-varying data sets.

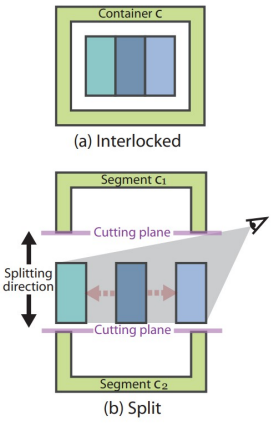


Figure 3.3: Splitting container to reveal enclosed parts [[Li+08], Figure 7]

Tatzgern et al. improve on the work of Li et al. by finding frequently recurring subsets in a mesh and grouping them, then selecting the best representative of that group and exploding it based on a quality score.[TKS10] The frequently recurring subsets are found automatically based on a frequent sub-graph search. The resulting explosion diagrams are especially useful for technological models where there are many identical subsets, and the explosion displays only one of them instead of doing this for each of these subsets and taking up a lot of screen space. A resulting exploded view and the selection of a representative can be seen in figure 3.4. For biological datasets, however, this extension is less useful, since it brings little advantage due to the distinct structure of biological objects. More relevant, however, is

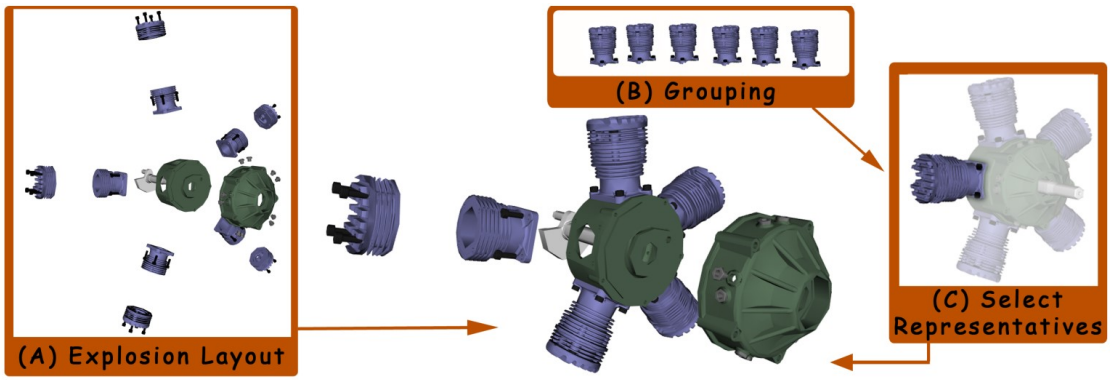


Figure 3.4: System Architecture presented by Tatzgern et al. They describe it as follows: "Our system consists of three different modules which affect the rendering of compact explosion diagrams. By supplying a 3D CAD model, it automatically computes an initial explosion layout (A), it finds groups of equal parts (B) and it selects a representative (C) before it initiates the rendering."[[TKS10], Figure 3]

its quality score which is used to select the representative. This is also applicable to general explosion views and can be used to quantitatively describe the quality of an explosion view. It is defined by the following evaluation criteria:

- **Size of the footprint of the exploded view:** Describes the entire screen space that the exploded view occupies.
- **Visibility of parts of the exploded group:** Describes the relative measure for the general visibility of the parts.
- **Part directions relative to current camera viewpoint:** Assumes that explosions similar to viewing direction are more difficult to read, they compute the average dot product between the viewing angle and the explosion direction.
- **Size of footprint of all other similar groups without any displacements:** Describes how well other similar groups are visible when selected representative is exploded.

These criteria are then weighted by Tatzgern et al., which influences the selection of the representative. Even if not all of these points are suitable for use in virtual reality, some ideas can still be applied. In particular, using the dot product between the camera position and the direction of the exploding parts is a helpful approach, which could be used to always push parts out of the line of sight without obstructing the view.

While Li et al. and Tatzgern et al. relied on calculating the position of the exploded parts, Bruckner et al. used various **force-based techniques** to transform parts to generate explosion diagrams.[BG06] Their presented method works with volumetric data sets and splits them into pieces before exploding them. For this purpose, they present three tools that interactively split the dataset at runtime using split axes and cutting planes. The first tool splits the first object hit by a ray based on the camera's viewing direction. The second splits all objects not just the first one, while the third allows the user to draw a line onto which all parts are projected, if the projection lies on the line the part is split. These tools are designed to work with volumetric data sets and can also be applied to voxel data sets and are therefore suitable for biological objects. Another interesting approach they use is the definition of forces acting on all parts to explode the object. These forces are defined by Bruckner et al. as follows:

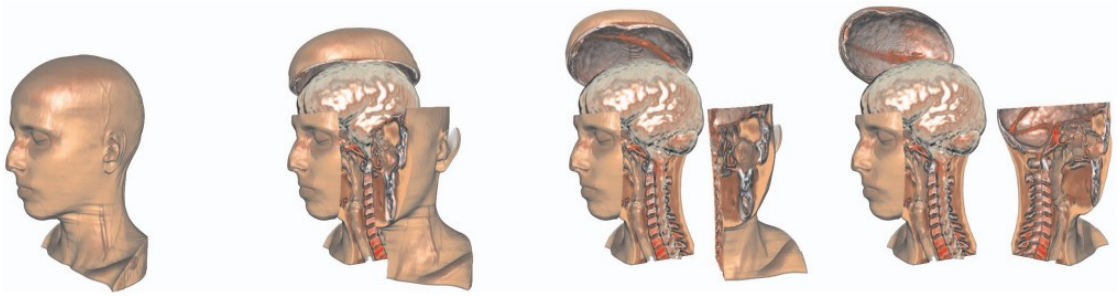


Figure 3.5: Interactive exploded-view illustration of a human head with increasing degrees-of-explosion created by Bruckner et al. using their force-based approach. [[BG06], Figure 2]

- **Return force:** The part should move as little as possible from its original position. Therefore, there is a force pushing the part back to its original position. Bruckner et al use the following formula, where r is the vector from the current vertex position to that of its original and c_r is a constant factor.

$$F_r = c_r * ln(\|r\|) * \frac{r}{\|r\|} \tag{3.1}$$

- **Explosion force:** The user can select individual parts and the force will push all other parts away from the selected parts. Bruckner et al. use the following formula to describe the force F_e that emanates from each selected part and acts on every part P_i.

$$F_e = \frac{c_e}{e^{\|r\|}} * \frac{r}{\|r\|} \tag{3.2}$$

  Here c_e is a constant factor and r is a vector from the explosion point to the closest point of the geometry of part P_i.
- **Viewing force:** To make the exploded view interactive, Bruckner et al. introduce another force that takes the camera position into account, obscuring parts are thus pushed away from the viewing direction. The procedure is described by Bruckner et al. as follows: "For a part P_i we determine the point along the viewing ray corresponding to the explosion point's projection which is closest to the center of P_i. The force F_v is then:"

$$F_v = \frac{c_v}{\|r\|} * \frac{r}{\|r\|} \tag{3.3}$$

  Here c_v is again a constant factor and r a vector which points from the shortest point on the viewing axis to the center of the part P_i.
- **Spacing force:** The last force F_s pushes all parts away from each other to avoid overlapping and is described by the following formula:

$$F_v = \frac{c_s}{\|r\|^2} * \frac{r}{\|r\|} \tag{3.4}$$

  Again, c_s is a constant factor and r is a vector pointing from the center of part P_i to the center P_j of every other part.

These forces are then weighted by Bruckner et al. and applied to each part. The result is a view-dependent explosion diagram which can be edited and expanded at runtime, an example can be seen in figure 3.5. Thus, it is also possible to uncover enclosed parts by separating the outer part through user input.

Sonnet et al. also use a similar force-based approach.[SCS04] However, their application differs in that a probe is used for explosion which the user can move through the dataset and whose effect radius determines the translation of the parts. An interesting addition presented for dealing with enclosed objects is that when loading the data set, the size of the bounding box determines the weight of the object. Smaller objects that are inside a larger one will hence be pushed away stronger from the effect radius of the probe. This method is a simple way to avoid occlusion in the case of an enclosed object, but it causes problems when the point used to explode the inner and outer object are the same, since the weighting then has no influence.

Virtual reality opens up new possibilities to interact with and study exploded views. In their paper, He et al. describe several types of interaction methods to explore a CAD model of a brain using VR devices.[HGM17] To compute an explosion graph that defines the transformation of the parts, they use the bounding box of each mesh, then build a complete bounding volume hierarchy from the bottom-up. This is necessary because medical data often do not have a clearly defined order of assembly that can be revealed. In their work, they use parent-child relationships to constrain the transformation and guide it in a constructive manner. To allow for easy manipulation of the exploded view's axis, they use Hermit splines, which can be drawn with VR controllers. They also describe several ways of interacting with the mesh to trigger different exploded views, as illustrated in figure 3.6:

- **linear explosion:** The user defines an axis by moving the controllers apart in parallel, the mesh is then pulled apart along this axis.
- **leafing interaction:** He et al. describe this interaction as slicing the object and then leafing through the pieces as if one were leafing through a book.
- **fanning interaction:** This interaction also first cuts the object into slices, then the individual slices are fanned out as if you were holding playing cards with their backs facing upwards in front of you.

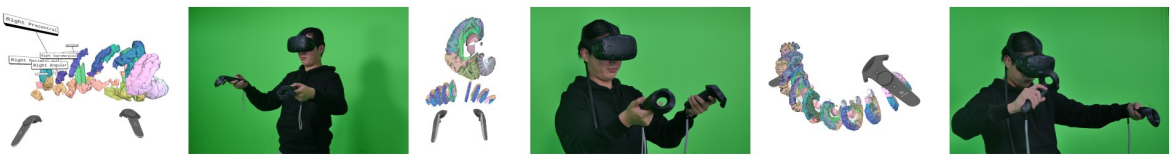


Figure 3.6: Interaction methods described by He et al. with the controller gestures that generate them. Linear explosion(left), leafing(middle) and fanning(right). [[HGM17], Figure 1]

Not only the interaction with VR plays an important role in this work, but also the special feature of a **time-varying data set**. Special measures are necessary to display these in an exploded view, but after extensive research no related work on this subject could be found. Therefore, in this work, a focus must be placed on solving this problem.

Although exploded views are the main part of this work, they are not the only way to avoid occlusion. Another approach that can be combined well with interaction in VR is the use of **magic lenses**. This is like a kind of magnifying glass that allows the user to look inside an object and hide the obscuring geometry of the surface. One advantage of this over cutting planes is that the context is preserved because only a selective subarea is hidden. An example of their work can be seen in figure 3.7. One possible implementation of this has been described by Viega et al. among others.[Vie+96] They

hide the surface of a hand to reveal the underlying skeleton. The same concept was extended by Hua et al. to work in augmented reality with a physical prototype.[HB06] This also functions as a magnifying glass that can be moved across a desktop to allow a look inside various objects. For example, you can look inside the buildings of a city model or explore the inside of a human body. Their user study showed that "magic lenses are efficient cognitive filters that dynamically organize and display data when and where it is needed."[HB06] They proved to be more efficient and intuitive than a traditional interface, although their prototype lacked stability. Another alternative approach, which is combinable, is proposed by Preim et al.[PRS97] In their paper, they use different scaling techniques to enlarge and dynamically label objects of interest, making it easier to identify details.

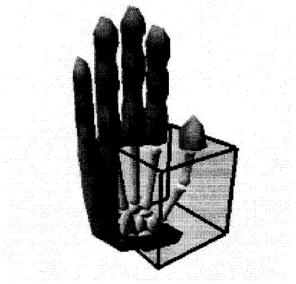


Figure 3.7: Magic lenses as presented by Viega et al. [[Vie+96], Figure 1]

Another unique method was presented by McGuffin et al.[MTB03] They show various tools they have developed to inspect volumetric data by peeling and cutting parts away. They introduce a Leafer tool to slice and open the dataset, and a Peeler tool to show different layers of the dataset. They also introduce a Boxspreader tool that pushes voxels away, a Hinge Spreader tool, and a Sphere Extender tool to cut and extend different layers of the dataset. Figure 3.8 shows these tools on a volumetric dataset of a human head. With these, complex transformations of the data is possible, which can provide insights into the interior of various volumetric data sets. One observation made by McGuffin et al. is that animations have greatly improved the usability of these tools, because without them, users had difficulty understanding how the tools operated.

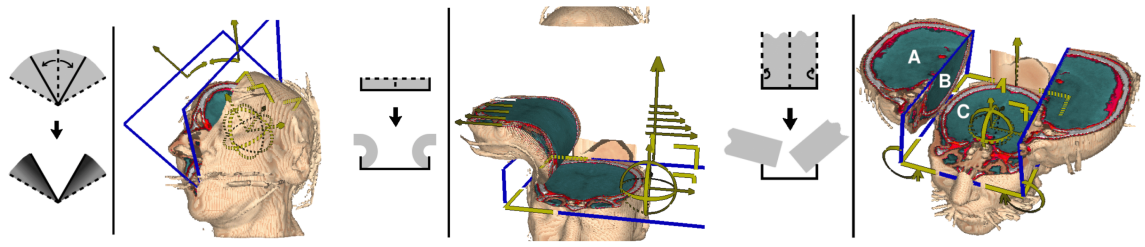


Figure 3.8: Tools presented by McGuffin et al. Hinge Spreader(left), Peeler tool(middle) and the Leafer tool(right). [[MTB03], Figure 4,11,8]

Another related work was presented by Tania Krisanty at the Chair of Computer Graphics at the TU Dresden. She uses a similar slightly simplified version of the same data set used in this work and shows the use of cutting planes in virtual reality, illustrated in figure 3.9.[Kri22] She presents several interactions in virtual reality, like grabbing a new cutting plane from an imaginary backpack.

One issue that arises is the loss of context when viewing the inner cells; one solution developed for this is the selective deactivation of cells, which facilitates the examination of individual cells in the interior but does not completely solve the problem.

Figure 3.9: Demonstration of the clipping plane interaction developed by Tania Krisanty.[[Kri22]]

4 Methods and Implementation

In order to explain the developed proposed solutions, this chapter will first conduct a problem analysis, then a theoretical classification of the explosion views, and finally the implemented methods are explained. To illustrate the implementation, first the structure of the program is presented, then the individual approaches are introduced, and finally the VR interaction and surface inspection are demonstrated and the data set and its change over time are discussed.

4.1 Problem Analysis

The problem addressed by this work can be divided into three sub-problems. On the one hand, this is the representation of three-dimensional cells and cell complexes; on the other hand, this is the investigation of possible ways to avoid occlusion and to use immersive techniques and interactions to enable cell and cell surface exploration. As mentioned in the introductory chapters, different types of explosion views will be tested and compared in order to solve the problem of occlusion. Special requirements and restrictions apply in this problem scenario, as both the data set and the use of VR technologies necessitate significant changes to traditional explosion views.

Therefore, the structure of the data set should be explained first. It was generated using the Morpheus program and depicts the development of cells over a predetermined time period. The appendix contains a diagram A.1 showing the structure of the data set. The individual cells consist of voxels which are arranged in a grid layout. Each cell has additional properties such as a unique ID, a cell center and a population to which the cell belongs, as well as data describing the surface properties. A population in this context refers to a cell type with unique propagation properties that can be defined in Morpheus. Different cell populations form a cell complex which is simulated by Morpheus. During the simulation, Morpheus then generates snapshots of the current state of the cell complex at regular intervals. These are saved as XML files and form the data set used for the visualization in this work. In this way, each snapshot describes a temporal state that precisely defines the arrangement of the individual cells in the form of a list of position and surface property values. The decisive factor here is that the shape and position of an individual cell can change significantly over

the course of the simulation. The chosen method of occlusion avoidance must therefore take this into account and the visualization should enable the precise inspection of a single cell at any time.

If one now wants to look at the inner cells of the data set and inspect the interaction of the different populations more closely, this is not possible due to the outer cells that cover them. So, occlusion occurs because data is obscured by other foreground data. As demonstrated by Krisanty's example shown in figure 3.9, using cutting planes is not a suitable method to avoid this problem while still being able to observe the interaction of individual cells with one another.

4.2 Methods

Explosive views are suitable for viewing the complete data set and, in particular, for looking into the interior of a cell complex and for inspecting individual cells in a targeted manner. This type of representation translates the individual parts of a model or dataset in such a way that each part is detached from its touching neighbors. It should be ensured that the original position of each part is recognizable or comprehensible. To achieve this, exploded views should follow the conventions and properties outlined by Li et al. in the background chapter 2. Exploded views that take these properties into account not only allow for the representation of relative spatial relationships, but also allow the viewer to mentally reconstruct the object and observe the arrangement of the individual parts within. This is further enhanced when the explosion strength can be adjusted interactively and is therefore a suitable method for the problem at hand. When drawing or generating an exploded view, the following parameters are of importance and must be specified individually for each exploded object:

- The canonical axis of the object, this is the main axis of expansion in which the parts should be exploded to make the mental reconstruction as simple as possible. This depends on many different factors of the object that is being inspected, for mechanical objects this is often related to the assembly order. The definition is more challenging for biological datasets and models. For the dataset at hand, it is not possible to define a clear canonical axis, because the cells change over time and there is no main direction in which the propagation of the cells is oriented. In this case, it is useful to make the choice of the axis dynamically adjustable, as this allows the axis to be adapted to the current state of the cell complex.
- The choice of perspective. Traditional exploded views are often drawn from the side of the object or from slightly above, looking down at the object. An orthographic view is frequently used to better indicate the distance between the individual parts. This is not the case with interactive systems like Li et al.'s, which allow the camera perspective to be adjusted to provide a more realistic representation of the object.[Li+08] Because this work focuses on the use of VR technologies, a perspective view should be chosen. Furthermore, the camera viewpoint is determined by the position of the headset and should change as the user moves. The use of VR headsets allows for a much more intuitive exploration of the data set and aids in understanding the object's composition. One issue with this is that it can cause visual clutter because the user's field of vision may be obscured by scattered parts.

- **View-dependent exploded views.** Closely related to the choice of perspective in interactive exploded views is the choice of whether the position of the exploded parts depends on the camera perspective or not. This results in two categories for exploded views in interactive systems. View-independent exploded views that transform parts along an axis or away from a point and are not affected by the camera. This allows for a more thorough inspection of all parts and the entire model. Each part is displayed in such a way that its position within the whole is discernible. View-dependent exploded views are the second type. Here, certain parts or points are selected that are always visible, no matter from which angle they are viewed. This allows for a close examination of specific parts or the object interior. Which of these possibilities is the better one for the given data set has to be investigated.

Interactive explosion views can be generated in two different ways. Either by calculating the final positions of the exploded parts or by defining forces that push the parts apart to create meaningful explosion views. Calculating the positions gives more control over the exploded view and allows to display the composition order. This is more difficult with force-based systems, since the forces would have to be defined so that no elements overlap and no blocking constraints are violated during the explosion. Both methods can generate qualitative exploded views, however, an advantage of the force-based method is that the strength of the forces can easily be adjusted at runtime and thus new explosion views can be generated. In this work both methods are implemented and compared.

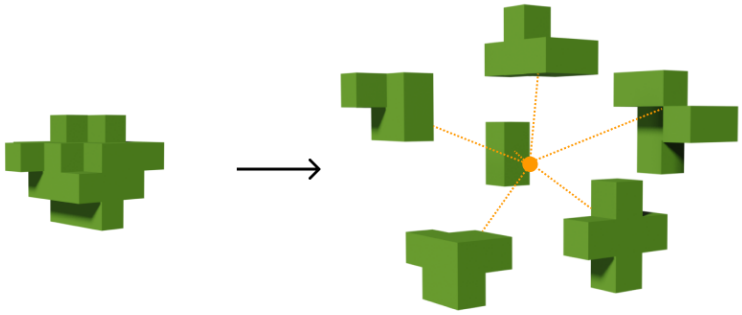


Figure 4.1: Exploding parts away from a single point. P_c is marked as the yellow dot.

4.2.1 Point explosion

The simplest method to generate an exploded view is to explode from a single point as pictured in figure 4.1. The initial position of the cells, which is read from the data set, is used as a reference point P_o. The user can then place a control point P_c at any position from which the explosion originates. The new position of the parts is thus determined by the translation along the linear line, which is defined by the control point and the initial reference point. The target position P_t is determined for each part P_i by the following formula.

$$P_t = P_o + (P_o - P_c) * F_{max} \tag{4.1}$$

Here, F_{max} is a constant factor that describes the maximum explosion distance. At the end the position of the part is determined by interpolating between the reference point P_o and the target point P_t with the user adjustable explosion strength $F_e \in [0, 1]$. It must be

ensured that the points are in the same coordinate system, this is not always the case with the implementation used, since the individual parts are child objects of a container object, which allows the data set to be scaled. To clarify the spatial relations, a further adjustable factor F_l has been added which is scaled with the length of the vector from the control point P_c to the initial position P_o. This can be used to amplify the spatial distances and to restrict the view to nearby objects. The final target position is thus described by the following formula:

$$\vec{d} = (P_o - P_c) \tag{4.2}$$

$$P_t = P_o + \hat{d} * F_{max} + \vec{d} * F_l * \|d\| \tag{4.3}$$

The point explosion is particularly suitable for selecting parts in a targeted manner and for examining them more closely. This is a view-independent explosion view, because the position of the camera has no effect on the position of the parts. However, this method does not take into account any blocking directions and relations. Enclosed objects are also not recognized and could remain hidden. Furthermore, there is no consideration of the canonical axes of an object. The method is similar to the implementation of the explosion probe by Sonnet et al.[SCS04] but has no effect radius, so all parts, no matter how far away from the control point, are pushed away.

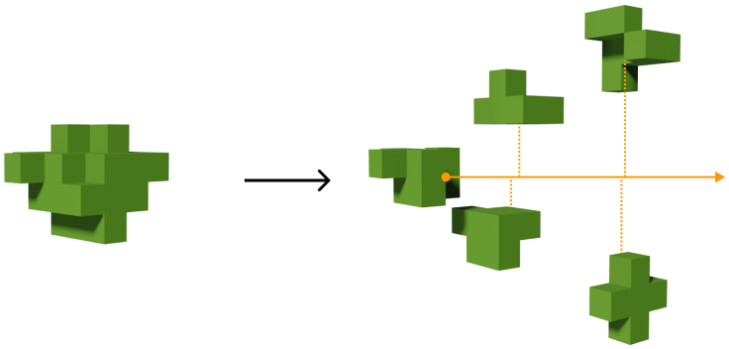


Figure 4.2: Exploding parts away from an axis defined by two points. The control point P_s is marked as a yellow dot on the left side of the line, point P_d as an arrow.

4.2.2 Line explosion

An advanced exploded view that allows more control is achieved by defining an axis and transforming the parts away from that axis, as shown in figure 4.2. This is done by allowing the user to set two control points in space, a line is then drawn between these two control points. The first of the control points P_s indicates the starting position of the line, the second point P_d the direction in which the line should progress. Now each part is projected onto this axis defined by the line, if this projection lies in front of the starting point, the part is pushed away from this projection point. By moving the control points, it is possible to select which parts of the model will explode from their original position and which will remain rigid. Moving the control point that defines the direction of the axis allows the user to create different explosion views. The projection P_{proj_i} of each part P_i onto the defined line can be calculated by projecting the vector $\vec{d_p}$ that goes from the starting point P_s to the respective

part onto the line $\vec{d_{ab}}$ that passes through the two control points. The projection point is thus given by the following formula:

$$P_{proj_i} = P_s + \frac{\langle \vec{d_p}, \vec{d_{ab}} \rangle}{\langle \vec{d_{ab}}, \vec{d_{ab}} \rangle} * \vec{d_{ab}} \tag{4.4}$$

The target position sought can then be determined by multiplying the vector $\vec{d_{expl}}$, which runs from the calculated point P_{proj_i} to the reference position of the part, by the explosion strength F_e. To move only the parts where the projection point lies on the line, the dot product of the vectors $\vec{d_{aProj}}$, which is defined by the starting point of the line P_s and the point P_{proj_i}, and the line $\vec{d_{ab}}$ can be used. Thus, only the parts are transformed where the dot product is equal to one. This allows the user, for example, to explode only half of the object and leave the remaining parts in their original position to get a better understanding of the composition. To improve the visualization and make it suitable for each type of cell, three further customizable factors have been introduced that change the target position of the parts and thus generate different explosion views. This makes it possible to customize the explosion shape of the transformed parts.

- The first factor F_1 amplifies the distance between the starting point P_s of the line and the projection point P_{proj_j} of the part. This allows to adjust the length of the body of the exploded view and to clarify the distances between the individual parts.
- The second factor F_2 amplifies the strength of explosion of the part based on the distance of the part's projection point to the starting point P_s of the line. This causes a cone shape of the explosive body, i.e. the area where the parts are effected by the explosion.
- The third factor F_3 adds a constant value to the explosion vector $\vec{d_{expl}}$. This leads to a cylindrical explosion view or can be used to enlarge the explosion in case of a low explosion strength.

By adjusting these three factors, different explosion views can be generated. The different possibilities with the corresponding factors can be seen in Figure 4.3. The explosion can be thought of as a cone or a cylinder, whose shape can be adjusted using the parameters F_1 to F_3. The factor F_1 increases the distance between the parts along the defined line, which ideally is placed on the canonical axis of the model. This increases the distance between the parts, which can be thought of as increasing the height of the cylinder. The radius of the base area of the explosion cylinder that is further away from the starting point can be changed with the factor F_2. Thus, the expansion of the parts further away from the control point P_s can be increased and a cone-shaped exploded view can be created. The factor F_3, on the other hand, adjusts the radius of the base of the cylinder. The final calculation of the target

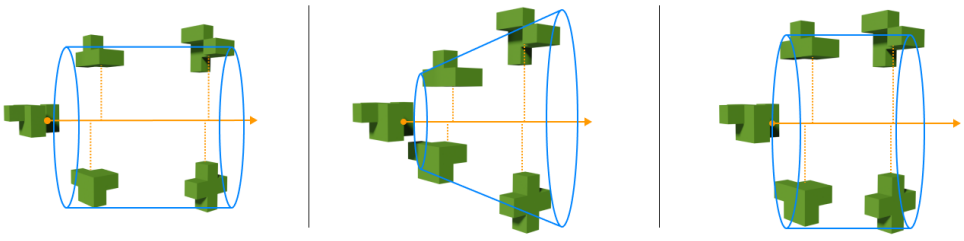


Figure 4.3: Influence of the factors F_1 (left), F_2 (middle) and F_3 (right) on the shape of the exploded view.

point is shown in Figure 4.4. Again, it must be ensured that all points and directions are in the same coordinate system; this was not taken into account in Figure 4.4. The complete implementation can be found on the GitHub page[1]. The line explosion diagram calculated by this method is also view-independent, since the camera has no influence on the target position of the parts. It is also possible to place the line in the canonical axis of the object and thus expand it in a meaningful way.

```
List parts
Position a, b
Float F1, F2, F3

function Explode(float F_e){
        var AB = b - a

        foreach(part in parts){
                var d_p = part.position - a
                var proj = a + DotProduct(d_p, AB) / DotProduct(AB, AB) * AB

                var d_expl = part.position - proj
                var d_a_proj = proj - a

                if(DotProduct(AB, proj) == 1){
                        part.targetPosition = part.initalPosition + d_expl.normalized * F3
                        + d_a_proj.magnitude * F1 * d_expl.normalized
                        + d_a_proj * F2
                }
                else
                part.targetPosition = part.initalPosition

                //LinearInterpolate(a, b, c) is a function that linearly interpolates from Position a
                     to a Position b, by c
                //so the returned value is a + (b - a) * c
                part.position = LinearInterpolate(part.initialPosition, part.targetPosition, F_e)
        }
}
```

Figure 4.4: This shows a pseudo code implementation of the algorithm used for exploding the parts away from a line.

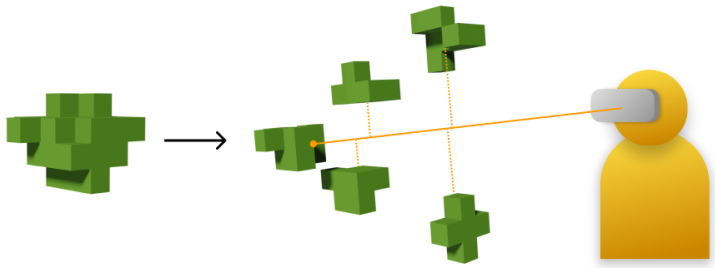


Figure 4.5: Concept of the head-mounted explosion. The point P_d is set to be the same as the headsets position. This allows a clear view onto the leftmost part, no matter from which angle the user looks.

4.2.3 Head-mounted line explosion

This method is based on the same implementation as the line explosion, but extends it to make it view-dependent. The only difference is that the control point P_d which determines the direction of the line is set to be the same as the position of the headset in that frame.

[1] https://github.com/JorGra/Bachelor-Thesis-Exploded-Views

This concept is illustrated in figure 4.5. When the first control point is set within the object, an exploded view is created, allowing the user to dynamically look into the object. All parts that are in front of the first control point are automatically moved away. This technique allows the user to walk around the object to inspect the interior of the cell from any angle. The head-mounted line explosion can easily converted to the previously described line explosion by pausing the position update of the point P_d, allowing for a closer examination of specific parts.

4.2.4 Force-based explosion

Unlike the previous techniques, this method determines the exploded position of the parts using a force-based approach. This approach is based on the same principles described by Bruckner et al., but the forces described by Bruckner et al. have been adapted to give good results with the existing program structure.[BG06] In comparison to Bruckner et al.'s method, however, the reference points of the parts are used to apply the forces and not the vertices of a volumetric data set. To create an explosion view, the user selects the desired target parts. It is possible to select multiple target parts, these are only affected by the return force and remain in their original position. Now four forces with different strengths are applied to all parts which push them away and allow a free line of sight to the selected parts. The four forces acting on each part are defined as follows:

- The first force is the **return force**, which causes the pieces to be pushed back to their original position. For this, the vector r which goes from the current position of the part to its original position is calculated and the part is pushed in the direction of this vector. To reduce the jittering when the parts are close to their original point, the vector r is normalized and multiplied by the logarithm of its length. This reduces the strength of the force when the length of the vector is small. The following formula is therefore used to describe the force. It is identical to the formula used by Bruckner et al. and affects every part.

$$F_r = c_r * ln(\|r\|) * \hat{r} \tag{4.5}$$

  Here, the constant term c_r refers to a variable that affects the force's strength.
- Any piece that is not a selected target piece is affected by the **explosive force** F_e. This pushes all parts apart and creates the exploded view, it emanates from each target part. It can be described by the following formula:

$$F_e = \frac{c_e}{e^{\|r\|}} * \frac{c_e}{P_{count}} * r * f_{expl} \tag{4.6}$$

  Where c_e is again a constant factor, r is a vector that goes from the target part to the part being handled and f_{expl} is a factor that amplifies the force. c_e is a value between zero and one that can be determined by the user, which is also why the factor f_{expl} was introduced to strengthen the force. The strength of the force is divided by the number of target parts P_{count} to remain constant on all datasets.
- The **viewing force** F_v makes the explosion diagram view-dependent. This is achieved by calculating an axis from the headset to each selected target part, from which other, non-target, parts are pushed away. The projection of each part onto the axis is calculated using the same formula 4.4 introduced earlier in the Line Explosion. The direction r

in which the parts are pushed is thus determined from the projection point of the equation and the position of the part. The strength of the force is then calculated as follows and applied to each part that is not a target part:

$$F_v = \frac{c_v}{\|r\|} * \hat{r} \tag{4.7}$$

The strength of the force F_v is here divided by the length of the vector r, this is useful to minimize the distance of the pushed away parts from their original position. Here, each part is pushed away no matter whether the projection of the considered part is behind or in front of the currently considered target part. This results in less visual clutter, since the target part is clearly separated from other parts and can be seen directly on the background without other parts behind it crowding the view. c_v is again a constant factor which allows the strength of the force to be adjusted.

- The spacing force F_s is responsible for preventing the pieces from overlapping and acts from any piece that is not a target piece to any other non-target piece. It is calculated as follows:

$$F_s = \frac{c_s}{\|r\|^2} * \frac{r}{P_{count}} \tag{4.8}$$

Here c_s is again a constant force, which is variable and adjusts the strength of the force. The vector r runs from the currently treated part P_i to the other part P_j, while P_{count} is the total count of all parts. This is used to normalize the force.

The individual forces can thus be adjusted by the variables c_r, c_e, c_v and c_s to generate different resulting representations. An advantage compared to the other methods is that it is possible to select any number of target parts. Since they remain at their original position, they can be viewed in detail and their environment can be observed.

4.2.5 Dynamic data sets for exploded views

A major problem that arises in the visualization is the temporal change of the data set. This is to be solved in two different ways. On the one hand, the change in the cell over time is visualized and displayed with less opacity. It can be selected how many previous time steps are displayed at the same time. This allows to view the temporal change of the cell over a period of time. The second approach adjusts the reference point used to determine the exploded position. This is set, at the user's request, from the centers of mass defined in the data set and interpolated between the individual time steps. The reference position of the last time step is defined as the starting point, now it is interpolated to the reference position of the part at the current time step. This happens at the interval with which the time steps are progressed. A problem that can occur is that the parts are close to the projection lines that are calculated to determine the target position. If this is the case, the target position can change abruptly, when the used reference point jumps over the projection line and thus cause a lot of visual movement which makes the inspection of individual areas difficult. In the dataset used for testing, it was found to be more useful to use the initial position of the cell as a permanent reference point. However, this has the obvious disadvantage that the cell can move far away from this initial point over the simulation, which can complicate the spatial comprehension of the structure.

4.3 Implementation

In order to test the developed methods, a prototype was built which allows the immersive exploration of the dataset. Existing frameworks for interaction and rendering in virtual reality were used to reduce the development time for device-independent development and to have more time for refining the methods.

4.3.1 Unity Application

The dataset is visualized and the exploded views are implemented using Unity 2021.3[2]. Unity is a 3D engine that can be used for game, program and movie development. Since it has been developed for over a decade, there are a variety of plugins and resources that can be used to increase development speed. It also allows platform-independent development and supports all conventional graphics cards. It uses C# as the programming language for implementing business logic while rendering is implemented natively with C++. The Unity-XR-Interaction-Toolkit[3] and the Unity-InputSystem[4] are used for VR support. These are two plugins developed by Unity that simplify the support of all conventional VR headsets by allowing them to be accessed with a unified API. The universal rendering pipeline[5] is used for more performant and lightweight rendering. It is a striped down version of Unity's standard renderer and allows for more frames per second on mobile devices at the expense of more advanced rendering effects. Since this work is about data visualization and high performance is critical for VR headsets, using this rendering pipeline is beneficial. An Oculus Quest 2[6] connected to the computer via Oculus link is utilized to test the implementation.

At program start the Resource folder is checked for valid Morpheus datasets. For each existing file that contains a time step, the corresponding Xml-file is read and loaded. Since each time step contains information about the cell populations and the states of the individual cells, these are loaded one after the other. For each time step the program loops through the data set and checks if a cell with the same ID already exists, if this is not the case a new object is created which represents the cell, stores its properties and contains a mesh for each time step of the cell. If a cell object with the same ID already exists, a mesh is generated which visualizes the state of the cell and is attached to the cell object as a child object. At runtime, only the child objects that depict the current time step are active while all others are deactivated. A manager class allows switching between the different implementations of the exploded views and sends the object transforms that are used to define the exploded positions to the script that drives the explosion. An interface implementation of the exploded view method allows for an extendable usage and development. A detailed object diagram which clarifies the program structure can be found in the appendix (A.2). The implementation of the methods has turned out to be quite straight forward. Only the scaling of the object which visualizes the data set complicated the implementation, because here the conversion of the coordinate systems must be taken into consideration. The force-based method uses

[2] https://unity.com/
[3] https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.3/manual/index.html
[4] https://docs.unity3d.com/Packages/com.unity.inputsystem@1.0/manual/QuickStartGuide.html
[5] https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@15.0/manual/index.html
[6] https://www.meta.com/de/quest/products/quest-2/

Unity's physics engine to determine the location of the exploded parts. It is also used to dampen the movement of the parts over time, preventing possible jittering of the parts. However, this can also lead to reduced responsiveness when changing the position of the parts. For all methods there is the possibility to visualize the most important directions used for the calculations by lines. This is to facilitate the comprehension of the applied methods and to make it easier for the user to understand the transformation of the parts.

4.3.2 VR interaction

All non-force based techniques presented are operated by control points that can be placed in the scene using a ray interactor. This allows the user to point at the control points, grab them and dynamically adjust their position. Furthermore, a user interface was implemented that allows the user to adjust all important parameters of the individual methods. So the best representation can be found at runtime. Additionally, a UI panel that enables the control of the time steps and subsequently permits the selection of a particular time step was added. At any time, it is possible for the user to separate out individual parts and take a closer inspection as it can be seen in figure 4.6. The part is then placed in the user's hand and can be rotated and scaled with the keys on the controller. Furthermore, a UI panel shows the ID and population to which the cell belongs. Grabbing a cell therefore enables basic surface inspection but the displayed information is limited and could be improved and expanded in future work.

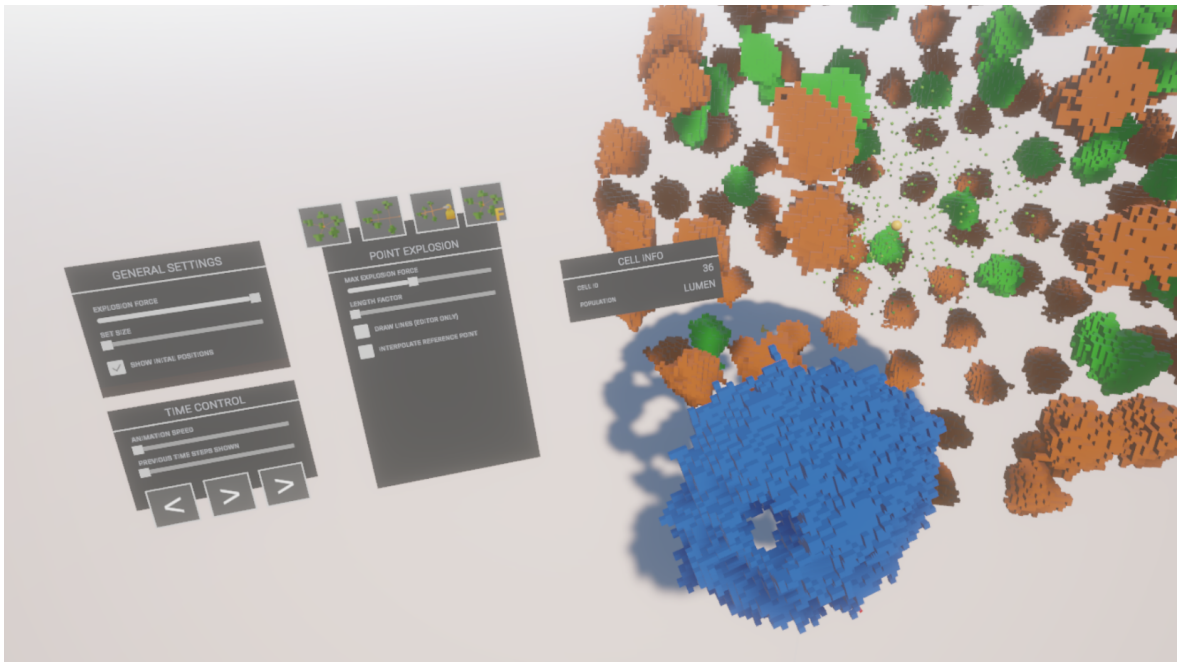


Figure 4.6: Picture of the prototype, on the left you can see the UI with which the user can interact. On the right, the lumen (blue cell) was taken from the data set and is examined.

5 Results

This chapter illustrates the implemented methods. The visualization of the data set can be seen in Figure 5.1. Here the point explosion was used to explode the cell. The dataset contains three different cell populations, which are colored orange, green and blue. The colors are set randomly at the start of the program when a new population is found in the dataset. Orange is the cell population named "cells", the population "cells_inner" is green, and the "lumen" is the largest blue colored cell around the center. Also small green dots can be seen in the cell center. These mark the initial center of mass of the cell, which is used as the reference point to create the exploded view.

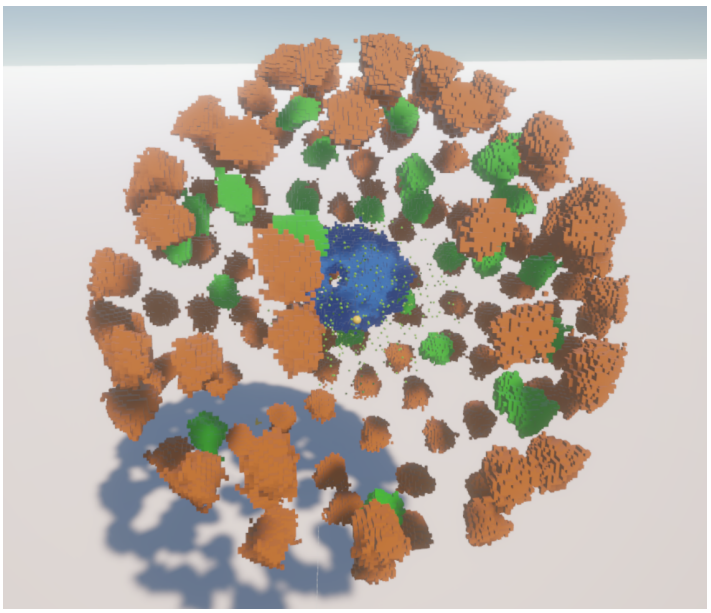


Figure 5.1: Point explosion of the dataset. The different cell populations were marked in orange, green and blue.

Figure 5.2 shows the data set as it is transformed with the line explosion. The yellow arrow indicates the axis on which the parts are projected as described in chapter 4.2.2. The yellow arrow is an interactable and can be positioned by the user. The three pictures show different parameter combinations of the factors $F = (F_1, F_2, F_3)$. So here on the left picture each part is translated along the line without influencing the distance of the parts to the axis. This can be achieved with $F = (1, 0, 0)$ and allows the expansion along the axis, similar to drawn

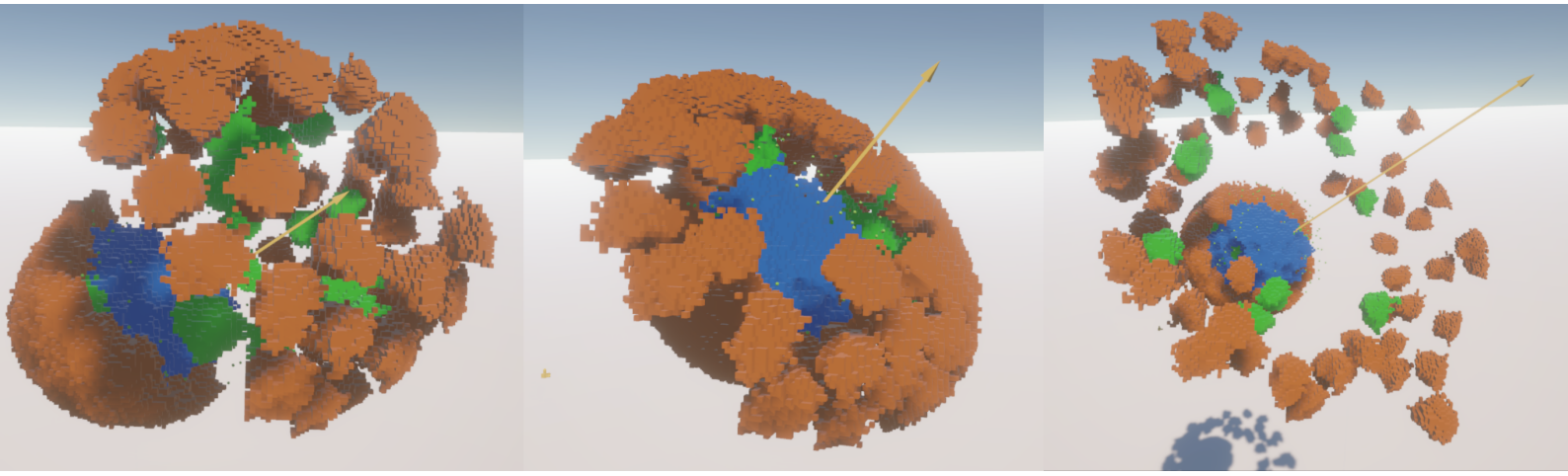


Figure 5.2: Different lines explosions created with different factor combinations of the parameters $F = (F_1, F_2, F_3)$. On the left $F = (1, 0, 0)$, in the middle $F = (0, 0, 1)$ and on the right $F = (0.3, 1, 0)$.

explosion views. The object structure remains largely intact and is merely pulled apart. It can also be seen that the control point P_s was placed inside the cell complex, and thus only the right half of the data set is expanded. In the middle picture you can see the factor combination $F = (0, 0, 1)$, which pushes the parts from their original position outwards, away from the line. This allows a clear view of the inside of the object. Interesting is the factor combination $F = (0.3, 1, 0)$, which can be seen in the right picture of Figure 5.2. Here a conical explosion is created which shows the inside of the object and also allows a clear view of the individual parts. Since the parameters can be adjusted via the UI panel at runtime, an analytical inspection is possible, as the best view can easily be found by adjusting the factors as needed. Figure 5.3 shows the extension of the line explosion to be view-dependent. Here, the point P_s was also placed inside the cell complex, pushing all occluding parts out of the view of the observer. On the right graphic, the lumen, i.e. the blue cell, was taken by the user, thus allowing an unobstructed view of the cells behind it or on the lumen itself.

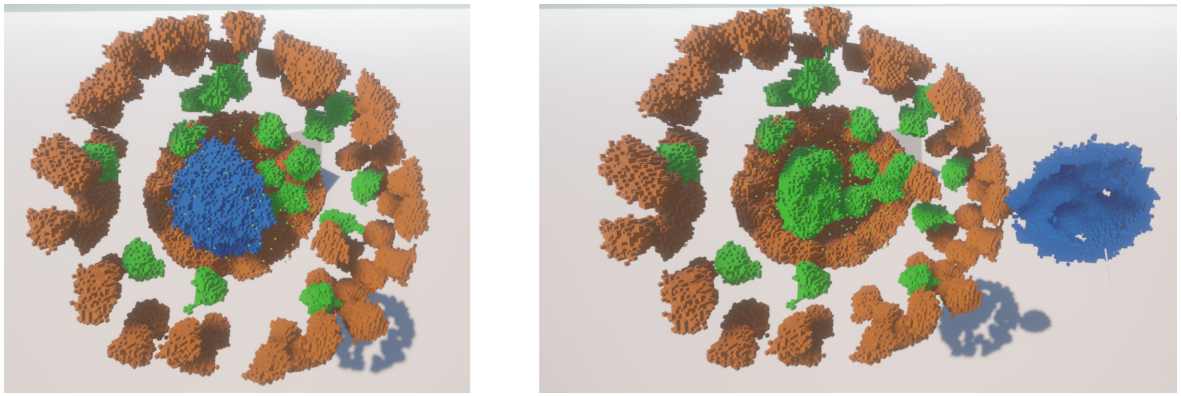


Figure 5.3: Extension of the line explosion to produce a view-dependent representation. On the right, the lumen, i.e. the blue cell, was taken out by the user to allow a clear view on the cells behind it.

Figure 5.4 shows the force-based explosion. Here, three cells were selected from which the explosion originates. These selected parts remain in their original position, while all other parts are pushed away. In the left of the two images, the viewingforce has been increased so that all parts that are not selected are pushed out of the line of sight and the target parts can be seen clearly. On the right image, the viewingforce has been set to zero, which means that

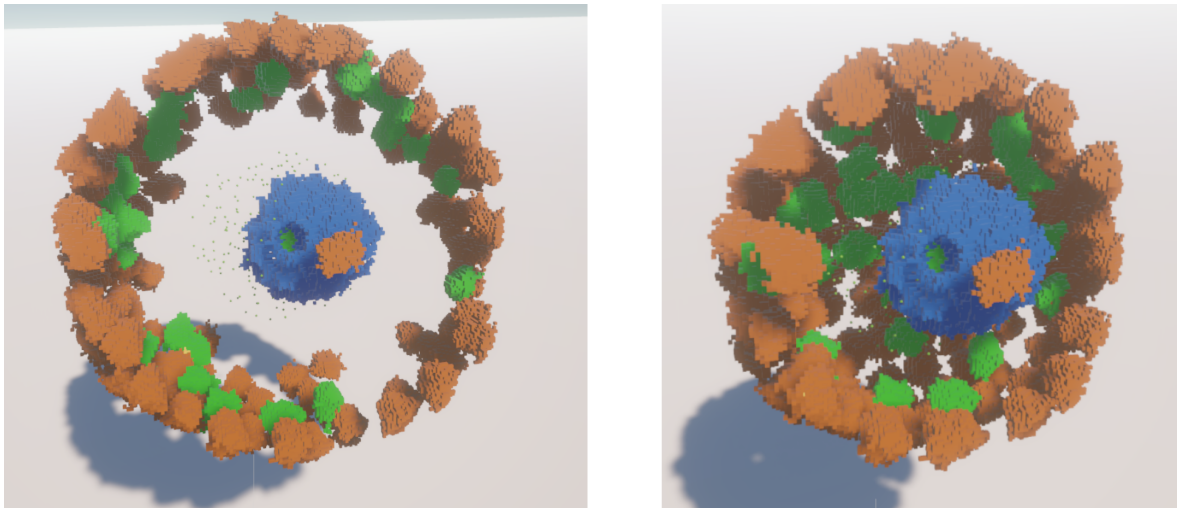


Figure 5.4: Force-based explosion, left with viewingforce, on the right the viewingforce is set to zero.

all parts are pushed away, but the object structure is broken up less. Figure 5.5 shows the UI panel with which the user can interact and adjust the parameters of the explosion views. Furthermore, the time control can be seen on the left. On the right side the population and ID of the selected cell is displayed. Also, a cell was taken and held in front of the panel to show the transparency, which shows the change of the cell since the last time step. This can also be seen in figure A.3 in the appendix, with the lumen cell of another simulation.

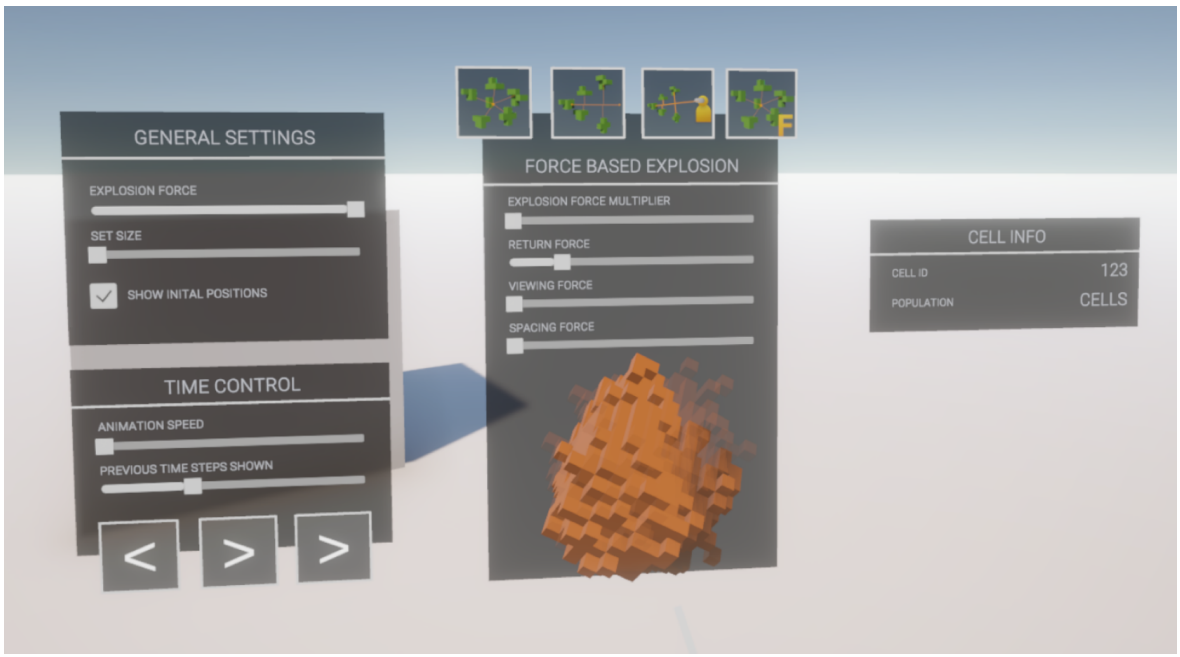


Figure 5.5: UI panel with which the user can adjust the parameters of the explosion and the time. A cell is held in front of the panel to show the transparency, which visualizes the change of the cell with respect to the last time step.

6 Evaluation

6.1 Qualitative Analysis

In order to evaluate the developed methods, first a qualitative analysis is performed, in which the individual methods are discussed and evaluated with the results of the expert evaluation, followed by a quantitative analysis, which is done by conducting a performance test. The expert feedback was recorded in a meeting with two external scientists of the TU Dresden and developers of the Morpheus program. For this, the developed methods were first presented, then both had time to try out the methods themselves. A first generation HTC Vive was used and notes on feedback were taken during testing, then questions were asked about effectiveness, usability and areas for improvement, the specific questions can be found in the appendix(A.1). Both experts stated that they had little to no experience with virtual reality headsets. Overall, there was very positive feedback on the methods presented and the possibilities they offer for exploring the data sets. They also stated that the various methods are all useful in different ways and that no single method is best for every application. Positively, it was pointed out that both the occlusion that occurs can be avoided with the methods and the context of the cells is preserved, thus achieving the goal of the work. It was also noted that context preservation is not always desirable and that it can be useful to break it up in order to better extract other desired information. The ability to extract and view individual cells was also found to be very useful and intuitive. Furthermore, a variety of extension possibilities and suggestions for improvement of the methods were made. According to how they are used, the various techniques are to be classified as follows.

6.1.1 Point explosion

The point explosion is the easiest method to understand and helps the viewer to get an understanding of the composition of the object, especially for the initial observation. Here it is easy to orientate and select individual parts of the data set. The ease of use with only one control point is also helpful here, as well as the steadier expansion compared to the other methods presented. With this method, the context of a single cell is hard to see, as cells that are close to each other can be far apart in the exploded view. However, according to the

experts, this is not decisive, since a good mental reconstruction of the cell complex is still possible. Also, this exploded view leads to a well presentable visualization, which is easily understandable even for uninformed people, since the object structure and shape remain similar. However, the implemented length factor does not have the desired effect here and is more of a hindrance, since it breaks up the original cell structure, which is important for the mental reconstruction. Here, an effect radius as described by Sonnet et al. might be more useful. It could also be advantageous to be able to set the explosion point, from which the explosion originates, directly on individual parts, so that these remain at their original position. This is currently possible if the effect point is set to the reference point of the part, but is difficult in practice because the necessary precision is not available with the controllers. The point explosion is therefore particularly suitable for orienting oneself in the data set and quickly gaining an understanding of the rough object structure and the object interior.

6.1.2 Line explosion

The line explosion has proven to be the most effective method for analytical observation. Due to the possibility of exploding the object only partially, when the control point is placed inside the cell complex, the mental reconstruction is facilitated and the context is better preserved. Nevertheless, it is possible to inspect the interior of the cell complex and the individual cell interactions. If the dataset is scaled to a small size, the head-mounted method can be used to easily view the interior of the cell from each side. As all parts that are in front of the control point are dynamically pushed away. This allows for an easy intuitive inspection of the cell and its composition. Although the operation of the control points in the original implementation is a bit cumbersome, the expansion of the parts is still easy to understand. The interaction is improved with the head-mounted method, as there is only one control point that needs to be placed. One improvement that has been suggested is easier switching between the head-mounted method and normal line explosion using two control points. This is currently implemented as a toggle using a UI panel and could be improved if this is set to a button on the controller to be able to quickly switch between the view-dependent and view-independent version. Adjusting the three factors F1, F2, and F3 can effectively be used to either better represent the composition or allow an unobstructed view of the cell interior. These options also allow flexible customization of the exploded view depending on whether context preservation or free representation is desired. The experts found the method to be intuitive, reactive and interactive, but also had a number of suggestions for improvement. For example, the naming of parameters and methods was considered to be not very meaningful. Here it could help to make the naming less implementation-oriented and to promote a better mental model of the user. In the HCI, this is described as gulf of evaluation and describes the discrepancy between what the system displays and what the user expects. So the current naming is too system-centered and could be improved by considering the factors F_1 to F_3 as parameters of a cone. This deviates slightly from the implementation, but creates a more representative metaphor for how the parameters work. Furthermore, it was suggested that the method be could called conic explosion instead of line explosion to reinforce this mental model. This could also change the current interaction, in that the user no longer adjusts the two control points and then changes the parameters, but can move a cone-shaped object that can be scaled and changed by gestures. This could then be pushed into the model and thus determines the expansion of the parts. The parameters would thus be mapped more

intuitively as a three-dimensional object, which is much easier for the user to understand. Nevertheless, this method, especially with the extension to the view-dependent view, has proven to be the best method for analytical examination. It can be used best for exploring the object structure and still get a good understanding of the composition.

6.1.3 Force-based explosion

The force-based method uses a different approach to create the exploded view than the other methods. The advantage of this is the dynamic nature of the explosion created. Any number of target parts can be selected here to define the explosion. Because the target parts remain in their original positions, it is possible to select a subset of the dataset and view it while pushing away all other parts. So it is possible to view groups of cells over time without completely hiding the others. If the viewing force is increased, all parts that are not target parts are pushed away from the viewing direction. This allows an unobstructed view of the selected parts and has a similar effect as if these parts were deactivated. However, since this is not the case and the parts are merely pushed away, the object structure is preserved better than if the parts were only hidden. Because the explosion emanates from all selected parts, the view is particularly useful for viewing cells that make a large spatial change, as all parts are dynamically moved away from it. A problem, which occurs due to the implementation, is the sluggishness, with which the cells move. This is partly due to the drag value of the Unity Physics objects. This value introduces friction into the calculation and reduces jittering of the parts when they bounce beyond their target position. A downside of this, however, is that the parts become less responsive. This can be partially counteracted by increasing the strength of all forces. However, finding suitable values that both generate a good explosion view and allow parts to jump quickly to their final position without moving beyond their target position remains a difficult task. Furthermore, the implementation of the spacing force needs improvement, as here every part is tested against each other resulting in an $O(n^2)$ complexity of the algorithm. This exponential complexity quickly leads to performance problems as the number of parts increases. This is where using a spatial hashgrid, into which each part is inserted and each part is checked only against the parts in adjacent grid segments, might help. The method is therefore most useful when a group of cells and their development over a period of time is to be examined more closely. It serves as a middle ground between the simpler point explosion and the more complex line explosion.

The discussion with the experts also led to further suggestions for improvement, which could be explored. For example, it was suggested to refrain from the implemented egocentric view and the current locomotion system and to create an observer view, in which the user does not move in space, but instead the dataset is transformed. This could make orientation easier as well as navigating the control points and transforming the dataset. Likewise, with the simplified usability, it would also make it easier to get started using the program. Similarly, operation could be made easier if more intuitive virtual reality interaction capabilities were utilized. For example, scaling the dataset could be done by pressing the controller's grip buttons and making an expanding or contracting motion. Furthermore, it was suggested that the selection of individual cells could be extended to subsets for closer inspection. While the program structure used, gives good results for the dataset, it will not work for any cell structure. It is necessary, for example, that each cell has a reference point. However, there

are cell complexes where this is not the case because the cells span the entire cell complex and do not have a confined localized shape, and exploding them with the method used would not yield a useful result. For this, either the cell structure must be broken up and a reference point must be chosen, or a completely different approach to the explosion is required.

6.1.4 Visualizing the change in time

One of the problems that this work tries to solve is how to deal with the temporal change of the cells and how to represent it with explosion views. One challenge that arises is the choice of the reference point that is used to calculate the target position when the explosion occurs. Most of the methods presented use the initial centroid of each cell for this purpose, which gets set when the dataset is imported. However, if the cell moves far away from this initial point over time, this complicates the analysis of the cell complex, as the geometry of the cell may be far from its target position. For the point explosion, it was then tested how this could be improved. The proposed method interpolates the reference point between each time step, between the centers of mass that exist in the dataset. Thus, the spatial change of the cell is visualized. This allows individual cells to be tracked and observed over time. This method is useful to get a rough idea of the local change, but it also brings problems. For instance, the spatial change of the cells is erratic and can differ greatly between the individual time steps. This makes mental reconstruction and analysis of the parts much more difficult. Increasing the time between time steps reduces the resulting visual clutter as the parts move less quickly between positions, but does not solve the problem. This can also be further improved if a lower time interval is selected for creating the snapshots when generating the data set. Furthermore, cells may partially overlap because the cell centroid is used as the center. So this method is only useful if the cells have strong locality, because single outsiders can lead to strong deviations of the cell centroid. This problem occurs rather rarely in the datasets used, but also becomes a problem with enclosed cells. Also, discontinuities occur in the data set when a cell propagates and splits. Then the original cell vanishes and two new ones are created, this leads to abrupt changes of single cells between time steps. To solve this, a reference to the parent cell could be stored in the child cell and interpolated between the parent cell and the child cell during the cell division. However, this would have to be done when the dataset is generated and is currently not possible. The second approach, which has been tested to represent the temporal change, is to use transparency. However, it turned out that transparency leads to too much visually presented information and makes it more difficult instead of simplifying the comprehension of the presented information. Furthermore, the representation still remains very erratic, which is not changed by displaying the last time step. Here it was suggested to put the control of the time steps on more accessible control buttons to allow a quick navigation in time. This could help to make the temporal change of the cell more understandable, as it would be possible to quickly switch between time steps. Overall, the representation of temporal change is a difficult problem to solve and requires further techniques to be represented in a well comprehensible way.

During an informal meeting, the program was also tested by five other persons. For this, the program was streamed from the editor to an Oculus Quest 2 via Oculus link. Four of the five

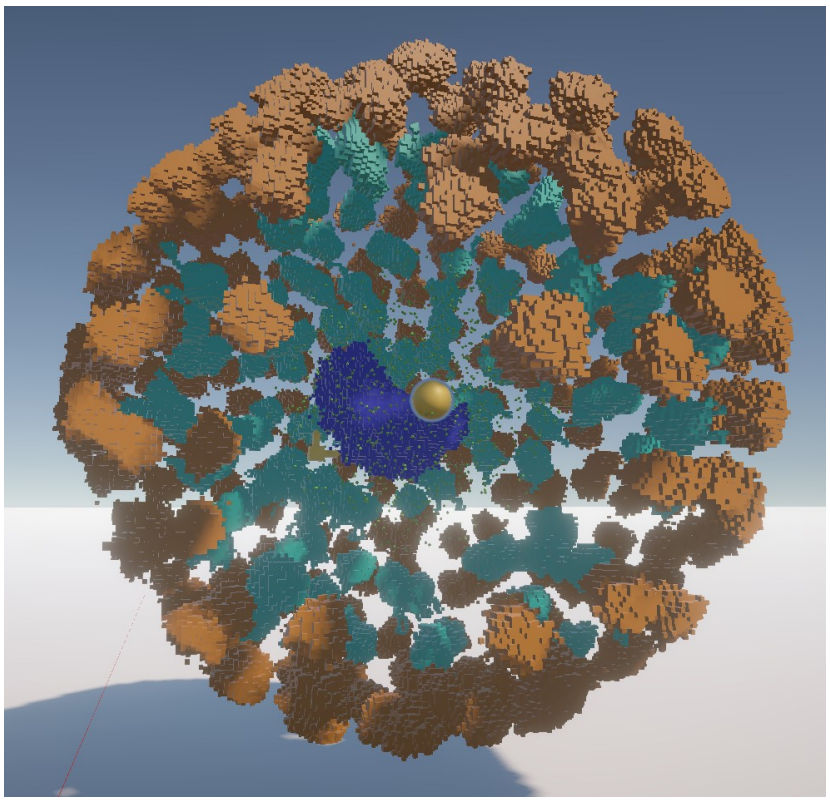


Figure 6.1: Last time step of a data set with 448 parts.

people had little to no experience with VR headsets and navigating the virtual environment, but are experienced in computer science. Some usability issues came to light. Most found navigation and orientation difficult, and using the control points was also described as tricky. One person tried to throw the control points and parts intuitively, another had person with visual impairment had difficulties to use the menu and the control points as they are quite small. Usability could therefore be improved with additional user feedback such as vibrations, intuitive gestures and scalable menu.

To improve the comprehensibility of the methods, the option to draw lines between the cell parts and the projection lines was added. However, this turned out not to be useful, as it leads to great visual clutter, especially when there are many parts active at the same time, and makes the analysis more difficult.

6.2 Performance analysis

A performance analysis was conducted to test the scalability of the methods. The program structure used runs at an average of 101 frames per second on datasets with 448 individual cells at thirty time steps on a computer with an NVIDIA GeForce GTX 970 and an Intel(R) Core(TM) i7-5820K. This is also the largest data set tested and can be seen in figure 6.1. In this dataset, showing the last timestep over 9.5 million tris are active and need to be rendered. During the last time step the cell complex is at its largest state and the most individual cells are displayed. The render time here in the editor is 13.4 milliseconds on average, so an average of 74.4 frames per second. The CPU is the limiting factor here, since there are

over 3400 draw calls which increases the render time for such a large number of cells. The complete results can be seen in figure 6.1. These have been recorded on a standalone process using the point explosion. It can therefore be seen that the frames per second decrease with a higher number of parts, but the memory utilization increases primarily with the number of time steps. This was to be expected due to the program structure used. The performance is therefore sufficient for the required visualization and better than expected, but could still be significantly optimized. First of all, a reduction in the number of draw calls should be attempted, which should greatly improve the performance for time steps with many parts. The choice of Unity's Universal Rendering Pipeline has proven to be useful. Furthermore, the number of vertices could be reduced substantially, as currently each voxel is represented as a cube with six sides, i.e. 12 tris. This could be limited to only the surface structure of the cells when importing the dataset, thus massively reducing the tris count. Furthermore, running the program within the editor greatly affects performance. Here, a GUI must also be rendered and additional debug information is displayed.If the program is exported as executable, this also increases the performance, but some options are only usable via the UI panel of the editor. Since the data set is imported and completely computed at the program start, the initial load-up time becomes longer with increasing data set sizes. Also each time step remains loaded permanently in the memory, which could also become a bottleneck. This could be prevented if the time steps were loaded dynamically, but could also lead to problems, since the meshes of the cells would have to be calculated at runtime. The implementation of the line visualization is also in need of improvement. This currently adds a new object per line and therefore quickly reduces the performance. Since this visualization, however, did not prove to be useful, it was not addressed. This could be rectified by combining the individual lines to a single mesh and object or by drawing the lines on the GPU. The program structure used has proven to be very robust, which not only delivers performant results, but is also easily expandable.

Table 6.1: Performance analysis results of different data sets using the point explosion. δ is the maximum number of simultaneously active cells.

Dataset name	δ	average FPS	number of time steps	memory allocated (MB)
Example-CellSorting-3D	60	170	10	204
VirtualEmbryo_1	189	160	10	272
VirtualEmbryo_2	392	145	12	465
VirtualEmbryo_3	419	130	14	732
VirtualEmbryo_4	448	101	30	1228

7 Conclusion and Further Work

In order to solve the problem of occlusion, which occurs in the considered dataset, first an analysis of related works was performed and four methods were derived and presented. These were then compared and tested and evaluated by two experts. Furthermore, it was investigated how explosion views can be combined with time-varying data sets and which possibilities arise in this context. The methods presented for creating explosion views were described by the experts as effective and target-oriented. Thus, the goal of avoiding occlusion was achieved. Context preservation is also given with the line method. In addition, a prototype was developed which visualizes the data set and its individual time steps and allows them to be explored interactively in a virtual environment. Here, the methods can be tested and their parameters changed via a UI panel. Subsequently, the prototype was tested and evaluated by a test group. This also showed that apart from some usability improvements, the implemented methods are promising for solving the occlusion problem. Also, the performance of the prototype was analyzed and suggestions for improvement were discussed.

Evaluating the methods revealed additional possibilities and improvements that can be taken up in future work. Among the improvements mentioned in the results chapter, the temporal change of the cell surfaces between the time steps could be morphed. This would improve the still jagged transition between time steps and could provide a more linear time progression that is easier to inspect. Another possibility, which was suggested by the experts, would be to reduce the cell complexity by using NURBS. This could further reduce the visual complexity of the dataset, making it easier to analyze. This could also be achieved by applying the Marching Cubes algorithm and would additionally remove the verticies inside the cells, which could lead to a significant performance increase. An advantage of this would also be that the complexity could be dynamically adjusted. Furthermore, it should be investigated how additional cell properties could be visualized in a meaningful way without adding too much visual complexity. There are existing properties in the dataset that could be used for this task. It was also suggested to give each individual cell a shading to make them more distinguishable in the population. Similarly, the explosion views could be abandoned and other methods of occlusion avoidance could be tested. Here, for example, the use of the magic lenses described in the related work chapter would be a possibility. A combination of methods might also be feasible. Overall, however, the work has yielded promising results

and suggested new approaches to interactive exploration of datasets, which can be taken up further.

Bibliography

[FV82] James D. Foley and Andries Van Dam. *Fundamentals of Interactive Computer Graphics*. USA: Addison-Wesley Longman Publishing Co., Inc., 1982. ISBN: 0201144689.

[MK93] R. Mohammad and E. Kroll. "Automatic generation of exploded view by graph transformation". In: *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*. 1993, pp. 368–374. DOI: 10.1109/CAIA.1993.366643.

[Vie+96] John Viega et al. "3D Magic Lenses". In: *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*. UIST '96. Seattle, Washington, USA: Association for Computing Machinery, 1996, pp. 51–58. ISBN: 0897917987. DOI: 10.1145/237091.237098. URL: https://doi.org/10.1145/237091.237098.

[PRS97] Bernhard Preim, Andreas Raab, and Thomas Strothotte. "Coherent Zooming of Illustrations with 3D-Graphics and Text." In: Jan. 1997, pp. 105–113.

[MTB03] M.J. McGuffin, L. Tancau, and R. Balakrishnan. "Using deformations for browsing volumetric data". In: *IEEE Visualization, 2003. VIS 2003.* 2003, pp. 401–408. DOI: 10.1109/VISUAL.2003.1250400.

[LAS04] Wilmot Li, Maneesh Agrawala, and David Salesin. "Interactive image-based exploded view diagrams". In: *Proceedings of Graphics Interface 2004*. 2004, pp. 203–212.

[SCS04] Henry Sonnet, Sheelagh Carpendale, and Thomas Strothotte. "Integrating Expanding Annotations with a 3D Explosion Probe". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '04. Gallipoli, Italy: Association for Computing Machinery, 2004, pp. 63–70. ISBN: 1581138679. DOI: 10.1145/989863.989871. URL: https://doi.org/10.1145/989863.989871.

[BG06] Stefan Bruckner and M. Eduard Groller. "Exploded Views for Volume Data". In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 1077–1084. DOI: 10.1109/TVCG.2006.140.

[HB06] H. Hua and L. D. Brown. "Magic Lenses for Augmented Virtual Environments". In: *IEEE Computer Graphics and Applications* 26.04 (July 2006), pp. 64–73. ISSN: 1558-1756. DOI: 10.1109/MCG.2006.84.

[Mil+06] Peter A. Milder et al. “Fast and Accurate Resource Estimation of Automatically Generated Custom DFT IP Cores”. In: *Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays*. FPGA ’06. Monterey, California, USA: Association for Computing Machinery, 2006, pp. 211–220. ISBN: 1595932925. DOI: 10.1145/1117201.1117232. URL: https://doi.org/10.1145/1117201.1117232.

[Li+08] Wilmot Li et al. “Automated Generation of Interactive 3D Exploded View Diagrams”. In: *ACM Trans. Graph.* 27.3 (Aug. 2008), pp. 1–7. ISSN: 0730-0301. DOI: 10.1145/1360612.1360700. URL: https://doi.org/10.1145/1360612.1360700.

[TKS10] Markus Tatzgern, Denis Kalkofen, and Dieter Schmalstieg. “Compact explosion diagrams”. In: June 2010, pp. 17–26. DOI: 10.1145/1809939.1809942.

[Pei15] Leo Peichl. *Wie viele Nervenzellen hat das gehirn?* Apr. 2015. URL: https://www.helmholtz.de/newsroom/artikel/wie-viele-nervenzellen-hat-das-gehirn/#:~:text=Tats%C3%A4chlich%20geht%20man%20heute%20von%20etwa%2086%20Milliarden%20Nervenzellen%20aus..

[HGM17] Lindun He, Alejandro Guayaquil-Sosa, and Tim McGraw. “Medical Image Atlas Interaction in Virtual Reality”. In: 2017.

[Dre22] TU Dresden. *Morpheus, a modeling and simulation environment for the study of multi-scale and multicellular systems*. Nov. 2022. URL: https://morpheus.gitlab.io/.

[Kri22] Tania Krisanty. *VR_CA_VIS: A virtual reality visualization for cellular automata simulation*. Mar. 2022. URL: https://github.com/taniakrisanty/vr_ca_vis.

[kit] kitware. *Open-source, multi-platform data analysis and visualization application*. URL: https://www.paraview.org/.

A Appendix I

Figure A.1: This figure shows the structural layout of the Xml-file of the dataset. Some important attributes are written behind the name of the Xml-nodes. This figure omits many nodes that are not relevant for this work.

A.1 Expert evaluation questions:

Questions were asked in German.
Fragen:
Wie viel Erfahrung haben Sie mit VR?
Haben Sie Motion Sickness?
Wie sinnvoll ist die momentane Explosionsansicht? Warum?
Was sind Verbesserungsmöglichkeiten?
Wie effektiv ist die Okklusionsvermeidung?
Wie effektiv kann der Kontext der Zellen betrachtet/erhalten werden?
Wie verständlich sind die Methoden?

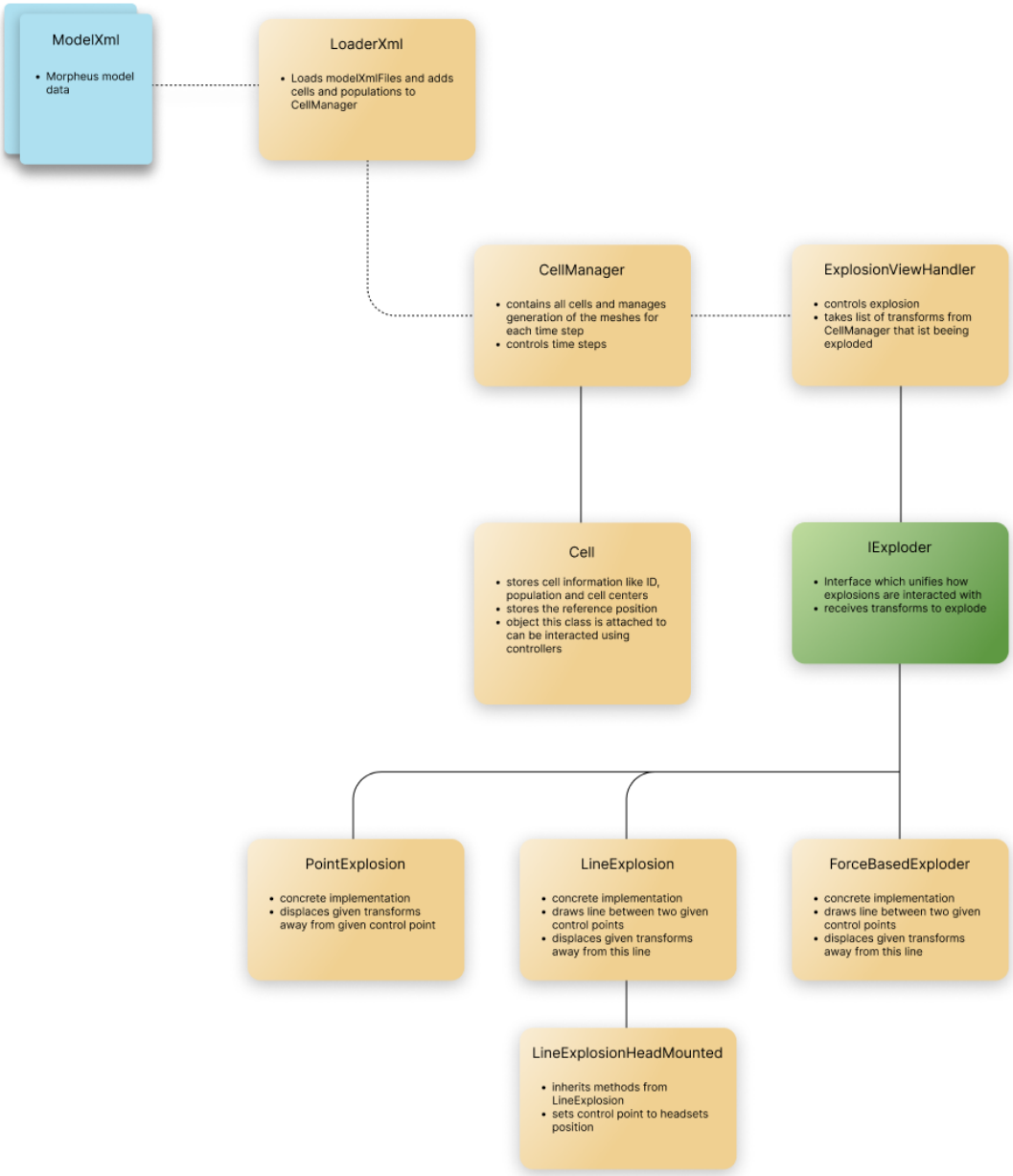


Figure A.2: Object diagram of the used program structure.

Wie leicht fällt die Nutzung?

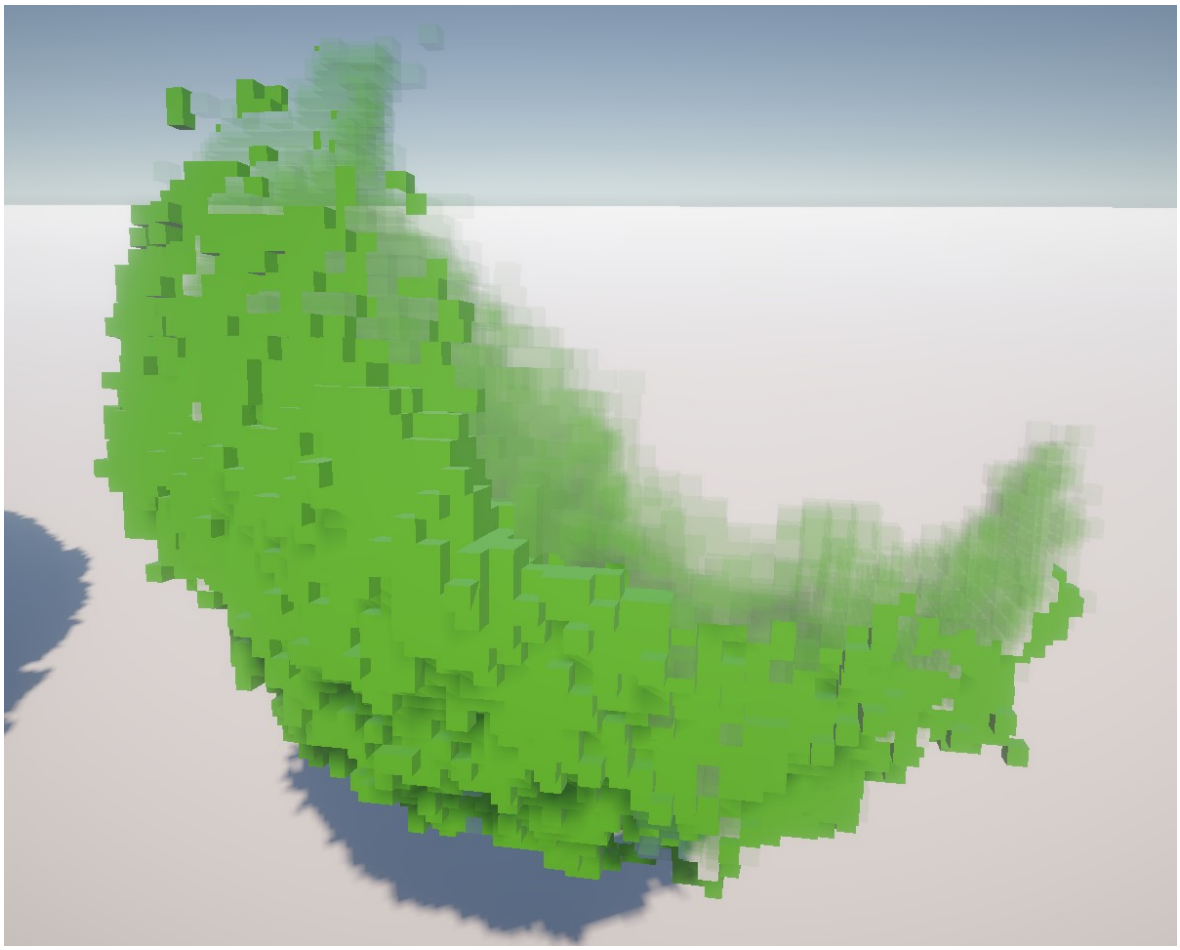


Figure A.3: The change of a cell compared to the last time step is also shown here.