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

1st December 2022

First referee
Prof. Dr. Stefan Gumhold

Second referee
2nd Referee

Supervisor
Supervisor I

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:	2020
Title:	Occlusion Avoidance for Immersive Inspection of 3D Cell Complexes and Cell Surfaces

Objectives of work

Momentan ist das besagte Thema in aller Munde. Insbesondere wird es gerade in vielen – wenn nicht sogar in allen – Medien diskutiert. Es ist momentan noch nicht abzusehen, ob und wann sich diese Situation ändert. Eine kurzfristige Verlagerung aus dem Fokus der Öffentlichkeit wird nicht erwartet. Als Ziel dieser Arbeit soll identifiziert werden, warum das Thema gerade so omnipräsent ist und wie dieser Effekt abgeschwächt werden könnte. Zusätzlich sind Methoden zu entwickeln, mit denen sich ein ähnlicher Vorgang zukünftig vermeiden lässt.

Focus of work

- Recherche & Analyse
- Entwicklung eines Konzeptes & Anwendung der entwickelten Methodik
- Dokumentation und grafische Aufbereitung der Ergebnisse

First referee:	Prof. Dr. Stefan Gumhold
Second referee:	2nd Referee
Supervisor:	Supervisor I
Issued on:	1st April 2015
Due date for submission:	1st October 2015

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, 1st December 2022

Joris Grau

Abstract

This is an abstract. The abstract is written after finishing the work and should give an overview about the motivation, used methods, as well as the results. It is here to inform the reader about the core topics of the work and if it is relevant to his research. The abstract stands for itself and uses no components of the rest of the work. In consequence, there are no references nor citations used here. It should be around 100 to 250 words. There should always be an english version of your abstract, regardless of the language the work is actually written in.

Zusammenfassung

Das ist eine Zusammenfassung. Die Zusammenfassung wird geschrieben, nachdem die Arbeit ferttiggestellt ist und sollte einen Überblick über Motivation, Methoden und die Ergebnisse geben. Die Zusammenfassung informiert den Leser über die Kernthemen der Arbeit und ob die Arbeit für seine Forschung relevant ist. Die Zusammenfassung ist von der Arbeit entkoppelt und verwendet keine anderen Komponenten der Arbeit. In der Folge werden hier keine Referenzen oder Zitierungen genutzt. Sie sollte zwischen 100 und 250 Worten umfassen. Unabhängig von der Sprache, in der die Arbeit verfasst wurde, sollte es immer eine englische Version der Zusammenfassung geben.

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

1 Introduction

The human brain consists of 86 billion neurons. Even 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. Nevertheless, correct and precise analysis of datasets is a cornerstone of any research, but this is often challenging and error-prone, especially when 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 methods can roughly be divided into two types: hiding unimportant data and transforming the data set so that occlusion is 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 VR. For this purpose, a data set of a cell complex is available which is visualized and transformed. This dataset was simulated 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 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. At the end there is a summary of the work as well as further approaches that could be explored in a future work.

If you look at traditional hand-drawn explosion views, you can see that both the explosion direction in which the parts are moved and the spacing of the parts must be well chosen. A number of other properties are also helpful for exploded views that are as meaningful as possible.

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 possibility is to use filters and to cut away the frontal parts, but this is done at the expense of the overview and can make it difficult for the viewer to correctly identify the position of the part in the object. To avoid this, this work focuses on exploded views, as they offer the possibility to reveal internal parts of complex models while preserving their position in relation to 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. describe five additional desirable conventions in their paper.

- Blocking constraints: Parts should be translated so that they do not pass 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

object-specific properties. In order to reduce the visual clutter, only a few of these axes should be chosen, otherwise the viewer will 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. In such cases it is also common to hide the outer parts to reveal the inner composition.

A distinctive feature which is relevant for this work are the utilized data sets They have been generated using the program *Morpheus* which is being developed at the TU Dresden. *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.

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.

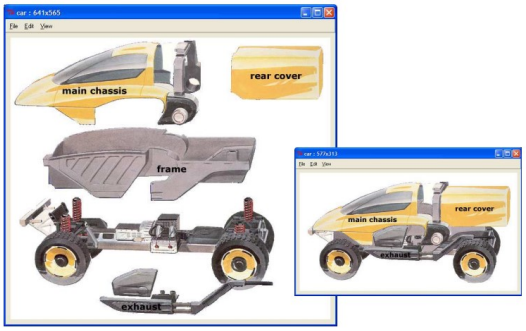
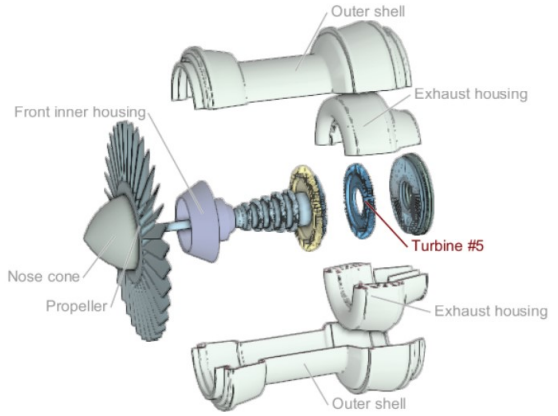


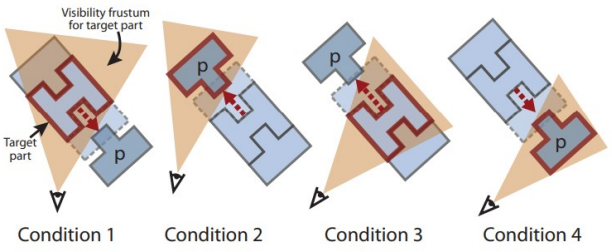
Figure 3.1: Interactive car diagram in its fully expanded (left) and fully collapsed configurations (right). [[LAS04], Figure 16]

Since the interaction capabilities 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 applied to three-dimensional objects. For this purpose, Mohammad et al. 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.[MK93] 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



(a) 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]

and show parts of interest. Their implementation works for both hierarchical and non-hierarchical models, which also allows it to process biological datasets where there is no fixed assembly order. 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; if the part is a target part, it must not be obscured by any part that has already been visited. 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. 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.

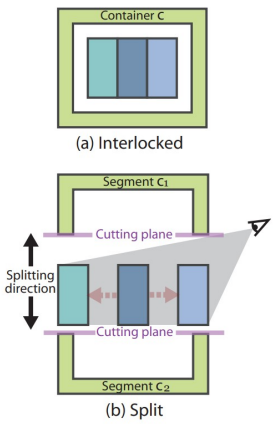


Figure 3.3: Splitting container to reveal enclosed parts [[Li+08], Figure 7]

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. For biological datasets, however, this extension is less useful, since it brings

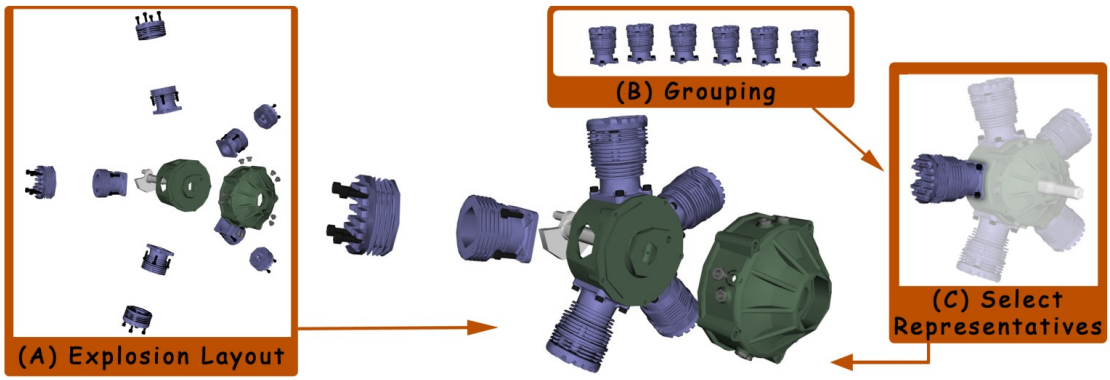


Figure 3.4: System Architecture presented by Tatzgern et al. They describe it as follows: "Our system consists of three different modules which affect the rendering of compact explosion diagrams. By supplying a 3D CAD model, it automatically computes an initial explosion layout (A), it finds groups of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

little advantage due to the distinct structure of biological objects. More relevant, however, is 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.

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, 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:

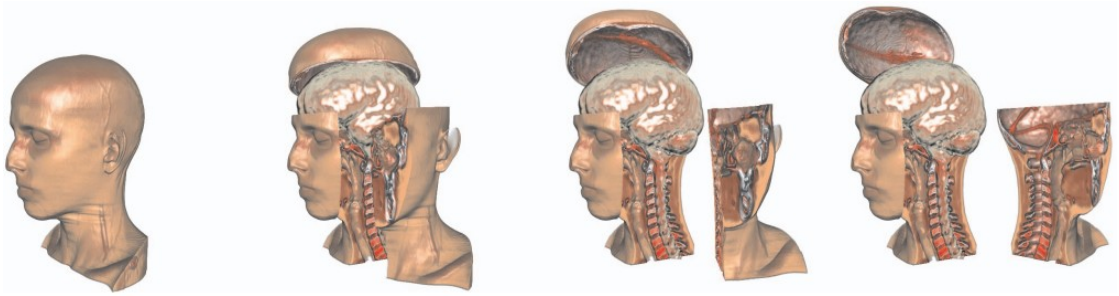


Figure 3.5: Interactive exploded-view illustration of a human head with increasing degrees-of-explosion created by Bruckner et al. using their force-based approach. [[BG06], Figure 2]

- **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 Fv 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. 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 does not have a specific assembly order 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:

- **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 could be found. Therefore, in this work, a focus must be placed on solving this problem.

While exploded views are a large 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. 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. In their paper, they use different scaling techniques to enlarge and dynamically label objects of interest, making it easier to identify details.[PRS97]

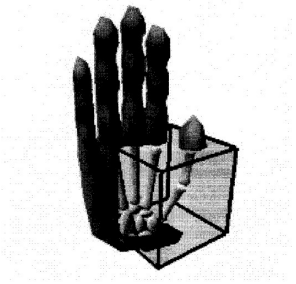


Figure 3.7: Magic lenses as presented by Viega et al. [[Vie+96], Figure 1]

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. With these, complex transformations of the data are 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.

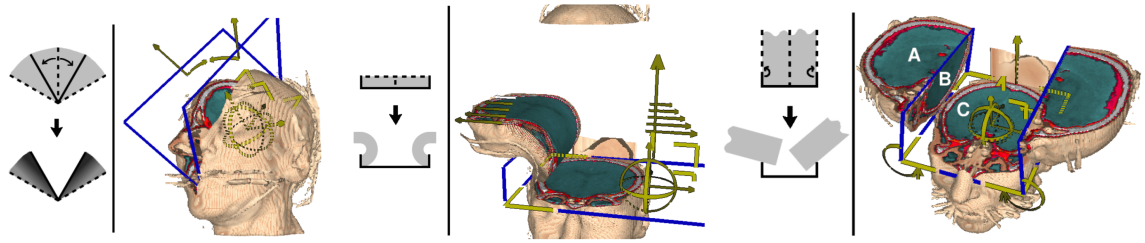


Figure 3.8: Tools presented by McGuffin et al. Hinge Spreader(left), Peeler tool(middle) and the Leafer tool(right). [[MTB03], Figure 4,11,8]

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.[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. To illustrate the implementation, first the structure of the program is presented, then the individual approaches are introduced, and finally the data set and its change over time are discussed and the surface inspection is demonstrated. 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 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 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 in 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, using cutting planes is not a suitable method of avoiding this problem while still observing the interaction of individual cells with one another.

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. 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 being viewed and 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 direction in which the propagation of the cells is focused. In this case, it is useful to make the choice of the axis dynamically selectable, 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. 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. Certain parts or points are chosen here that are always visible regardless of the angle from which they are viewed. This allows for a close examination of specific parts. 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 explosion 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, for this the program structure and the utilized tools are briefly explained.

4.1 Implementation

The dataset is visualized and the exploded views are implemented using Unity 2021.3. 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 porgram code while rendering is implemented natively with C++. The Unity-XR-Interaction-Toolkit and the Unity-InputSystem 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 is used for more performant and lightweight rendering, allowing 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 connected to the computer via Airlink 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 it is checked 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; all others are deactivated. A manger class allows switching between the different implementations of the exploded views. A detailed class diagram which clarifies the program structure can be found in the appendix. For this work, four different methods for generating exploded views were implemented.

4.1.1 Pointexplosion

The simplest method to generate an exploded view is to explode from a single point. 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 my implementation, 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:

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

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

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.

4.1.2 Line explosion

An expanded exploded view that allows more control is achieved by defining a line and transforming the parts away from that line. This is achieved 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, 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. In order to enable a targeted determination of the effected parts, only those parts are exploded where 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 $\vec{d_{ab}}$ is equal to one. This allows the user, for example, to explode only half of the parts of the object and leave the remaining parts in their original position to get a better understanding of the object. To improve the display 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 shape of the exploded view.

- 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 explode.
- 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 . The final calculation of the target point is shown in Figure 4.1. Again, it must be ensured that all points and directions are in the same coordinate system; this was not taken into account in Figure 4.1. The complete implementation can be found on the GitHub page. The line explosion diagram calculated

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

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.

4.1.3 Head mounted line explosion

This method is based on the same implementation as the line explosion, but extended 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. When the first control point is set within the object, an exploded view is created, allowing the user to look into the object dynamically. 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. A function has also been added that stops updating the position of the point P_d, turning the function into the previously described line explosion and allowing the user to examine specific parts.

4.1.4 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. The first force is the return force, which causes the parts 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 why the factor f_{expl} was introduced to amplify the force. The strength of the force is divided by the number of target parts P_{count} to remain constant.
- 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 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:

$$F_v = \frac{1}{\|r\|} * c_v * \hat{r} \tag{4.7}$$

  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. Unity's physics engine is used to dampen the movement of the parts over time, preventing possible jittering of the parts. However, this can also lead to a delay when changing the position of the parts, which reduces the responsiveness of the motion. 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.

4.1.5 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 to the control points 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 particular time steps was added. At any time, it is possible for the user to separate out individual parts and take a closer inspection. The part is then placed in front of the user and can be rotated and scaled with the control keys. Furthermore, a UI panel shows the population to which the cell belongs. Basic cell surface inspection is therefore possible but not particularly informative.

4.1.6 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 read, at the user's request, from the centers of mass defined in the data set and interpolated between the individual temporal points. 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 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.

5 Results

6 Conclusion and Further Work

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.

[Kri22] Tania Krisanty. *$VR_CA_VIS : A_virtual_reality_visualization_for_cellular_automata_simulation.$* Mar. 2022. URL: https://github.com/taniakrisanty/vr_ca_vis.

List of Figures

List of Tables

A Appendix I

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

B Appendix II

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.