FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Visualization of Composite Objects Through Techniques of Exploded Views and Ghosting

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Hartwig Wutscher
Matrikelnummer 0426961

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: o.Univ.-Prof. Dipl.-Ing. Mag. Dr. Monika Musterprofessorin
Mitwirkung: Ing. Peter Mindek
Supervision assistant 2

Wien, 18.12.2015

(Unterschrift Verfasser) (Unterschrift Betreuer)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Visualization of Composite Objects Through Techniques of Exploded Views and Ghosting

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Hartwig Wutscher
Registration Number 0426961

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: o.Univ.-Prof. Dipl.-Ing. Mag. Dr. Monika Musterprofessorin
Assistance: Ing. Peter Mindek
Supervision assistant 2

Vienna, 18.12.2015

(Signature of Author)

(Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Hartwig Wutscher
Nesselgasse 4/22, 1170 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Verfasser)

Kurzfassung

Die Explosionsgrafik ist eine Technik der Illustrativen Visualisierung, die die Funktion oder den Aufbau eines komplexen Objekts darstellt, in dem es in seine Einzelteile zerlegt wird, die dann so im Raum platziert werden, dass sie im Idealfall vollständig sichtbar sind und sich durch ihre Position auf ihre ursprüngliche Position innerhalb des Objekts rückschließen lässt. Diese Bachelorarbeit beschäftigt sich mit der Implementierung eines Plug-Ins für die Visualisierungssoftware VolumeShop, das in der Lage ist aus zusammengesetzten Dreiecks-Meshes einfache Explosionsgrafiken dynamisch zu erzeugen. Da es sich hier um eine interaktive Visualierung handelt, bei der man vor Allem den Blickwinkel verändern kann, ist es möglich die grafik von einem Punkt nahe der Explosionsachse zu betrachten. In diesem Fall würde die Explosionsdistanz sehr groß, wodurch die einzelnen Teile entweder sehr klein dargestellt oder ausserhalb des Bildes platziert würden. um dieses problem zu umgehen, wurde für diese Blickwinkel eine Kombination aus Explosionsgrafik und Ghosting eingesetzt. Ghosting ist eine andere Illustrationstechnik, bei der Objekte, die sich zwischen dem Betrachter und einem wichtigen Objekt befinden transparent dargestellt werden.

Abstract

Exploded views are a technique in illustrative visualization where the inner working of complex objects is revealed by segmenting it into several parts that are then displaced so that ideally they are fully visible and their positions convey information about their original place in the object. This thesis deals with implementing a plug-in for the visualization software VolumeShop that is able to create simple dynamic exploding views from composite triangle meshes. Because the Illustration is interactive and can be viewed from all angles, the explosion distance becomes very large to infinite when looking at the object from a viewpoint close to the explosion axis. To avoid this exploded views were combined with ghosting, which is a different illustrative technique where an object between the viewer and a more important object is drawn translucently.

Contents

@Peter: This is a draft, there are some issues with the references, I intend to finish over the weekend, as well as a paragraph in the section about architecture that is missing

CHAPTER 1

Introduction

> The aim of illustration is to generate expressive images that effectively convey certain information via the visual channel to the human observer. [9]

Illustrations have been used since the paleolithic age [**?**] to give their viewers a graphical representation of things seen, experienced or imagined [10]. Since the development of perspective and the necessity to intuitively explain complex technical, medical and scientific matters, especially since the industrial revolution, several techniques to successfully achieve this have been developed.

> Illustrative techniques are often designed in a way that even a person with no technical understanding clearly understands the piece of art. [**?**]

With the arrival of computers that are able to create complex graphics that react in real-time to user interactions, the transfer of these static illustrations to a dynamic visual representation has posed new challenges to the art of illustration. Exploded views are a technique used in illustrative visualization, where complex real world objects are drawn in a way that conveys how these objects are built, how they are assembled or how they work. Typically these objects are systems of interacting parts that may occlude each other or even be completely hidden by an outer hull of the visualized objects.
Examples for this would be a motor that consists of many independent parts, where the Illustration should give an idea of how it might work, a chest of drawers bought as a set of boards and connectors, that has a visual assembly instruction or the depiction of an anatomic feature.

> In an exploded view the object is decomposed into several parts which are displaced so that internal details are visible [5]

So for technical illustrations the parts that constitute a machine could be moved on an axis so that they are all visible and their position relative to this axis would still give information as

to where the place of each part would be inside the machine. As for the assembly instruction the displacement would happen along the axes they will be put together to make it clear to the viewer how to assemble the object.
The goal of my work was to create an interactive visualization that would dynamically generate simple exploded views. Building upon an existing plugin for the VolumeShop visualization platform, that splits objects along a plane and displaces the two parts, I created a plugin where the user can define an object of interest, that is not cut and stays in place, while the split parts are displaced, revealing said object.
The visualization system is interactive, the User can also choose, which part of the system is interesting to him, and view the object from different angles. From each perspective and foreach object of interest the system has to ensure that the object of interest is visible, the resulting visualization is compact but not too cluttered. Because of this the ideal position of the exploded parts is constantly shifting upon interaction. To give the transitions a more natural and therefore aesthetically pleasing look this displacement- the ”explosion”- can be shown as an animation.
When looking from a direction close to the displacement axis, the explosion distance becomes so large, that the object of interest is either to small, because it’s to far away when the whole graphic is displayed or parts of the graphic missing becuse they are behind the camera. To prevent this, I designed a system with a maximum distance of displacement. To prevent complete occlusion I combined the exploded view with ghosting, which is a technique where Objects occluding other objects are being drawn transparently.1.1

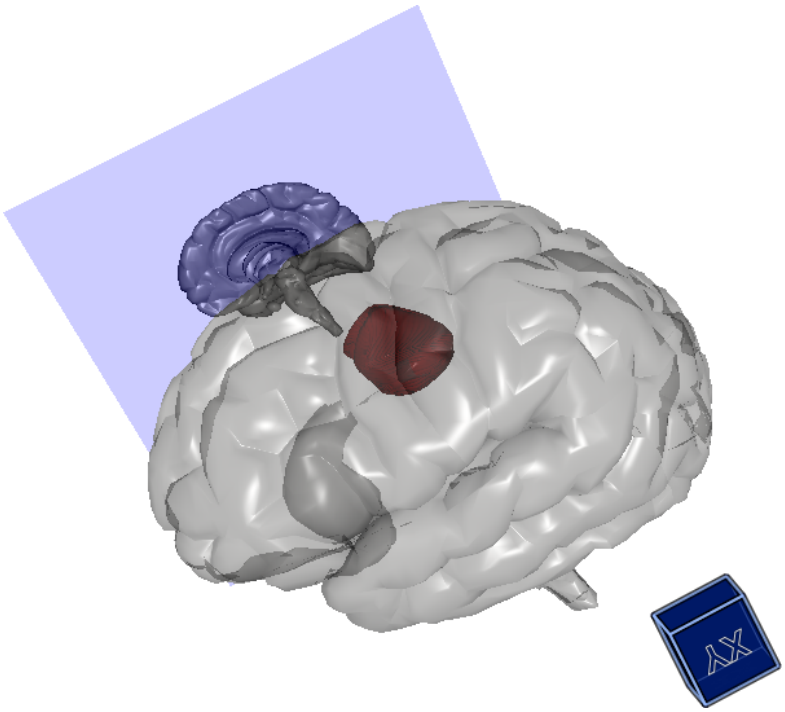

Figure 1.1: Illustration of a brain, combining ghosting and an exploded view

CHAPTER 2

Related Work

Like many other techniques exploded views by themselves are nothing new, hand-drawn and -designed instances of this technique have existed since the rennaissance (see 2.1).

> They (illustrators) carefully choose the size and shape of cuts, as well as the placement of the parts relative to one another, to expose and highlight the internal structure and spatial relationships between parts. [2]

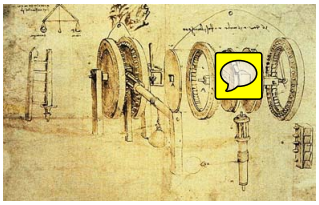

Figure 2.1: Exploded view by Leonardo da Vinci

Agrawala et al. [2] describe methods of how to generally create and evaluate design principles for visual communication: by analyzing existing examples of a technique, analyzing them using insights from cognitive and perceptive science, design rules are hypothesized. These design priciples are often qualitative guidelines and can not be simply translated into a generative algorith. They are then to be evaluated by user feedback from making the visualization available to the public and user studies.

2.1 What are the main challenges when creating exploded views?

For the concrete problem of creating exploded views Li et al. [6] provide a comprehensive list of such guidelines and what the challenges of the process of implementing them in an interactive system are.
Most important of all is the question what information should be communicated with the image and what the purpose of the illustration is. In essence what are the objects of interest that should attract the viewers attention? This usually is determined by preliminary definition of interesting features, definitions of parts, subsets of parts of the data or user interaction while viewing the graphic.

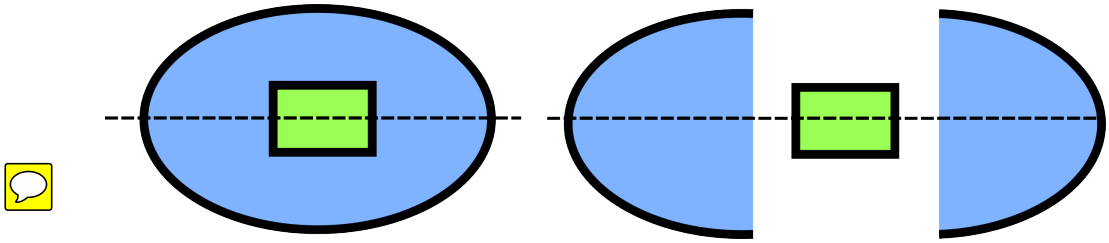

Figure 2.2: The object of interest is revealed by splitting its container and moving the two segments away from the object in the center.

Explosion direction

Also one has to consider in which direction the parts should be displaced.

> Many objects have a canonical coordinate frame that may be defined by a number of factors, including symmetry , real-world orientation, and domain-specific conventions. In most exploded views, parts are exploded only along these canonical axes. [6]

Especially if the graphic conveys technical instructions or how-things-work-descriptions descriptions these axes are usually the directions in which they are assembled [3] or - if applicable- the axis around which they rotate [7]. If the interesting parts are inside a container that is also part of the object, this container is split and its segments are also exploded. In the most simple approach, which is what I implemented, the cutting plane is the normal plane of the explosion direction, with the center of the object's bounding box (see 2.2).

Part hierarchy

Take for example an object that has a container and a lot of small parts that inside whose function should be clarified by exploding them in their canonical direction. In this case it might be convenient to split and explode the container along a different axis than the internal parts to get a more compact visualization(see 2.3).

In some cases, especially in the case technical visualizations e.g. assembly instructions or visual descriptions of machines the parts would block each other if they are not disassembled in the correct order and direction. A solution for this is an explosion graph representation of the Object [6]: Each node of this directional acyclic graph has edges pointing to parts blocking it from exploding (see 2.4). It is generated by an algorithm, that, starting with all parts in the active set, would recursively check for unblocked parts,remove them from the active set and draw an edge from the unblocked part to any active part that would touch it.
To further expand the possibilities, several parts can be grouped into sub-assemblies that are then part of a hierarchical explosion graph that has explosion graphs as nodes.

If one now wanted to see a certain part of the object of interest in an exploded view, all of the parts that its edges in the explosion graph point to would have to be exploded first. If it were

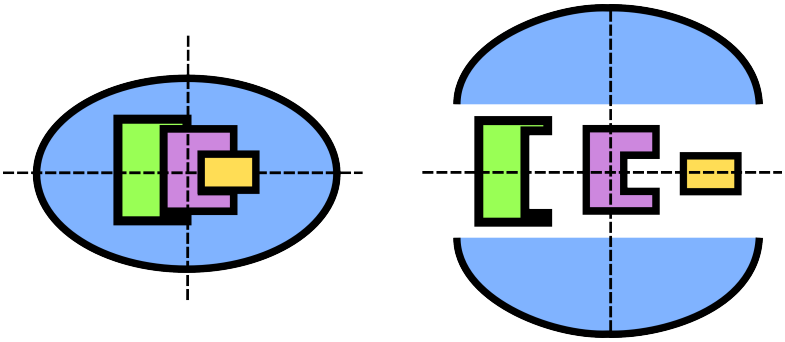

Figure 2.3: The interior parts of an object are exploded in a different direction than their container

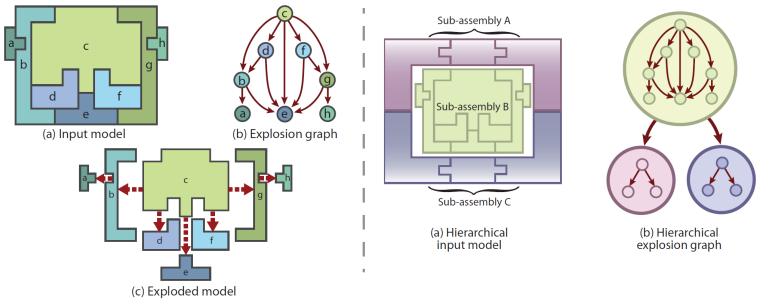

Figure 2.4: Explosion graph and hierarchical explosion graph [6]

also part of a sub-assembly, all sub-assemblies that would have edges pointing from the part's subassembly would need to be exploded before that.

Visibility

Another challenge is the decision how far to displace the objects. Ideally each part should be fully visible but if there are many objects to be exploded or system the displacement direction is similar to the viewing direction, the size of the graph would grow to be enormous therefore resulting in loss of detail and expressiveness of the visualization. That may make it necessary to have some overlap as a trade-off to retain the compactness of the visualization.

With a freely rotatable and movable viewpoint the user can avoid visual clutter or lack of compactness (depending on how the system behaves) by choosing a viewpoint that provides as little clutter as possible,which narrows down the possibilities of expressive viewpoints to a minimum.

A solution for this would be to use a view dependent force-based displacement behavior as suggested by Bruckner and Gröller [5]: Each exploding part is being displaced by a sum of multiple forces:

- **Explosion force** This force is pushing the part away from its original location, its magni-

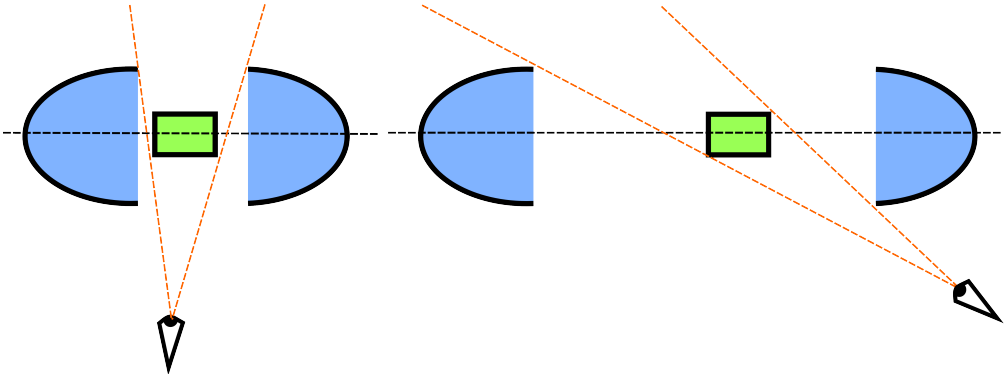

Figure 2.5: The closer the viewpoint comes to the displacement axis, the larger has to be the offset to reveal the whole object of interest

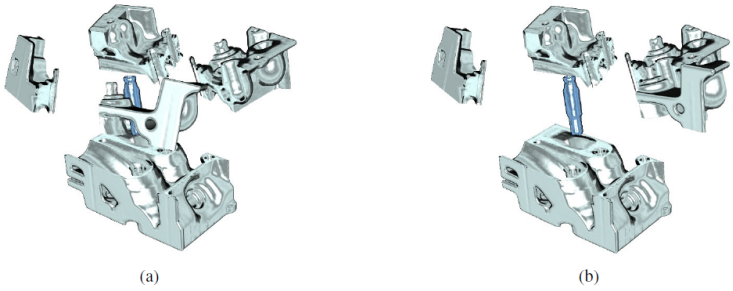

Figure 2.6: (a) force-based exploded view without viewing force (interesting part is occluded) (b) with viewing force [5]

tude is indirectly proportional to $e^{||r||}$ where r is the distance between the object and the explosion point

- **Spacing force** This repulsive force that each exploded part effects on all other exploded parts prevents parts from clustering and is indirectly proportional to r^2 where r is the distance between the two parts.
- **Viewing force**Additionally, a viewing force is introduced that pushes the parts away from the viewing ray, thus preventing occlusions. It is indirectly proportional to the distance r between the viewing ray and the part.

This approach results in very intuitive manipulation of the graphic, making the part of the object of interest that one is looking at always visible with smooth transitions between the positions of the parts when changing viewpoints.
In this approach the parts don't have to explode along a predefined axis, the explosion direction can be dynamically determined by the forces, which looks impressive for anatomical data, for a technical illustration it is less desirable, for one of the key advantages of exploded views,

the retention of information of a subpart's location in a system is partially lost without a clear displacement direction.
The solution presented in the paper is able to define axes along which to explode, this, though with similarly intuitive interaction by looking at a certain point still has the problem of little oversight when viewing from certain angles.

Elxploded views in architectural visualization

In architecture the need to visually communicate complex ideas about what a building should look like are essential to the craft. Exploded views are one of the techiques employed for this, in many shopping malls or complex public buildings(e.g. TU Wien) the floor plans are simple exploded views of teh building. Niederauer et al. [**?**] have devised an interesting non-invasive method to achieve this kind of floor layout:
First a possible candidate for a splitting plane needs to be found, ideally this is the ceiling of a floor. By counting the number of downward facing poligons at each possible vertex height, and define those heights with an exceptionally large number of downward facing poligons as a ceiling of floor. They then split and displace the meshes upwards until everything is visible, which results in an exploded view of the architecture that was analized. The most intriguing thing is that they do that by manipulating the openGL output Stream of third party software (among others Quake III) without the need to change that software in any way.
Fleiss [**?**] aproaches a similar problem though not just limited to drawing a floor plan with only one explosion direction, but a complex view that aims to illustrate the layout of walls, roofs, down torooms and objects inside the room. For each of these types of elements a grammar of how they should exploded according to rules based on the layout of the object type, e.g. a building is similar to a cuboid and has walls that explode along this cuboids axes and a roof that explodes upwards. He evaluates the efficacy of his approach with a simple user study, comparing the difficulty to perform a task for different user categeorys.

Ghosting

As the exploded view, ghosting is a smart visibility technique to illustrate the normally occluded inner workings of a complex object. In this case smart visibility is defined like this.

> Smart Visibility considers more than just light propagation. For example also the relevance of the individual objects is taken into account. An important object might shine through an otherwise occluding object closer to the viewer. [9]

The advantages of this technique are that the in contrast to the exploded view, objects visualized with ghosting do not take up more space than the objects themselves. Additionally, the inner parts of an object are shown at their actual position inside the object, thus being able to show the composition of an object accurately than an Exploded view might be able to do. Another advantage is that because the occluding object is not completely surpressed,
On the other Hand, ghosted views are usually more cluttered because of this, which can lead to loss of detail and insight. Also they tend to look confusing in case two objects are in between the

viewer and the object of interest. To avoid these problems and achieve good results the following rules should be considered: [9]

- faces of transparent objects never shine through
- objects occluded by two transparent objects do not shine through
- transparency falls off close to the edges of transparent objects

Interactivity

The biggest advantage of a computerized illustration over the classic static illustration is interactivity, most obviously the possibility to view an object from every different direction, which as described above creates its own challenges. Another aspect is the possibility for the user to choose which part of the graphic are interesting to her or him.
In Li et al.'s solution [6] this is restricted to the predefined parts of the data sets that can be selected by clicking on them which causes the explosion to adapt, so that the selected part is fully visible. Another possibility is dragging a part along its explosion axis which causes which causes all parts on that axis to move along with it until a part that moves along a different axis is encountered. Also a 3D fish-eye viewing technique can be used that, akin to the viewing force [5] can be used to move the parts along their axes.
The force-based solution is -due to the use of volume data- more flexible in the definition of parts that can be exploded: the data can be freely partitioned in cuboid subspaces that can be linked by hinges or to an axis, that can be exploded by the forces three above, each of which can be varied between a 0 and a maximum value along with the degree of explosion

CHAPTER 3

Practical part

3.1 Analysis and conclusions from other work

Since my task required creating exploded views for objects defined by triangle meshes, my approach would be rather like Li et al.'s solution, to use the groups of the mesh as the different parts of the objects which from which an object of interest would be chosen by clicking on it.
This object of interest satys at its place in the object, while the rest of the object is split along a cutting plane and then moved in a smooth animation to a point that leaves the object of interest fully revealed. This object would move along the axis defined by the user in a speed that is determined by how much of the Object is visible - the more the object is occluded, the faster the occluding parts move. This non-linear acceleration drew its inspiration from the before mentioned force-based approach to displacing the elements of the object, though my goal was to fully reveal the centered object, which though it normally works, is not guaranteed.
The biggest problem to tackle was now how to handle viewing directions that cause the graphic to grow up to -in th worst case- infinity.
Neither the prospect of visual clutter, indeterminate explosion directions nor the idea of a graphic that would grow infinitely when viewed from the wrong angle proved satisfactory, especially from a usability standpoint: Though the system allows for a freely movable viewpoint, most perspectives just deliver unsatisfactory or even useless results.
Since a maximum threshold for the explosion along an axis was inevitable, my intention was to keep it low enough to retain a satisfactory level of compactness I chose ghosting as a technique to still convey information about the object of interest to the user when it was occluded by another part of the object.

3.2 Implementation

The practical part of my thesis was to create a plug-in for the visualization application "Volumeshop " that was developed at the Computer graphics institute at TU Wien. [4]. VolumeShop was initially developed to create illustrative visualizations from data volumes acquired by e.g. a CT scan. It has been expanded by many plugins ever since and can now also process triangle meshes. I built my work upon an existing plug-in [8], that split meshes in image space.

Plan and milestone definition:

The practical part was split into three milestones containing the following tasks:

- **Milestone 1** *Selection of split meshes, selected parts are not split and stay in place* Make a simple, intuitive but manual way of creating exploded Views.
- **Milestone 2** *Find a safe distance, find a split plane* Automatize the creation of the visualization, by automatically finding a split plane and an offset.
- **Milestone 3** *Optimize Distance, force-field animation of split, optimize fringe distance cases* Make the visualization more pleasant to look at by adding a seemingly natural force-field animation and prevent unnecessary large offsets by introducing ghosting techniques.

3.3 Tools and languages

Since the project is based upon an existing framework, I used its languages, c++ for the main program and the GLSL for the openGL shaders.

Documentation of the implementation each milestone

Milestone 1: Selection of split meshes, selected parts are not split and stay in place

The original plug-in drew split meshes by drawing them twice, with the distance between the two parts set by the User with a slider in the interface. The plane that splits the object is also set by the user, also defining the displacement direction as the normal of the splitting plane. This plane is then translated with each rendering of the object with its normal always poining away from the Original split plane(see fig.) .The fragment shader then rejects fragments that are behind the also transformed split plane. The way the backfaces of the formerly closed object are rendered can also be chosen by user input.

The first step towards an exploded view is not to split the whole object, but only the portions of it that are not the object of Interest. To do this, the already implemented group selection feature is used to define an object of interest that is not split and translated like the rest of the mesh. Before each rendering, the renderer now loops through the sections of the object and checks if they are part of the object of interest. If the are not they are split and displaced as in the original plugin. For the interesting parts a third rendering of the mesh has been added to the display function. This render pass renders only the object of interest in its original location in the non-displaced mesh.

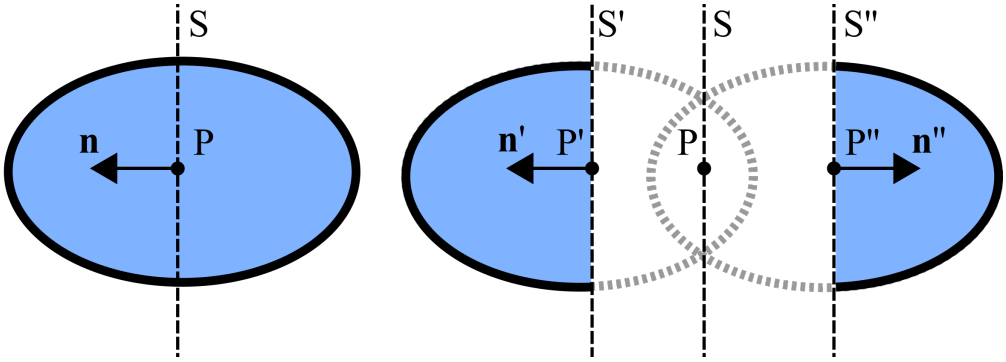

Figure 3.1: The object is rendered twice with the parts behind the translated splitting planes S' and S'' being ommited

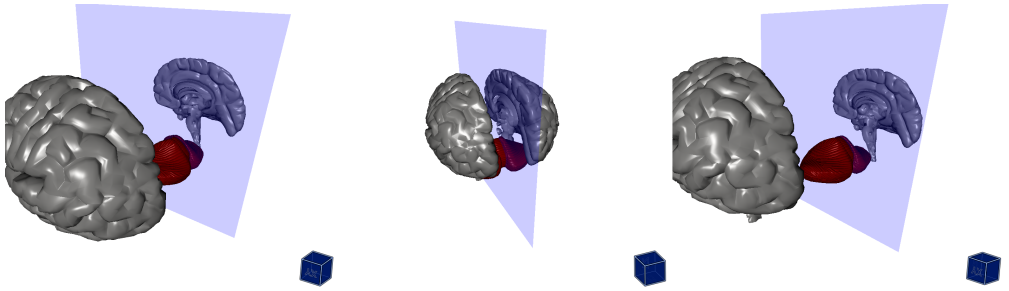

Figure 3.2: Exploded view of a brain with the offset of the displaced halves are set manually

To draw the Object of interest the shader doesn't need to bother with split planes, backfaces and a displacement vector. Thus the shader is reverted to a basic triangle mesh shader when created with the options corresponding to the object of interest.
After that is done an exploded view of the object is now rendered with manual definition of the offset and split plane as seen in fig 3.2.
One problem remains though: A colour picking algorithm was already implemented, which on a mouse click to the canvas. draws an overlay that has the group number encoded as an rgb color value. The color value at the click location is then instantly evaluated returning the the integer id of the selected part of the mesh. This works as designed but it doesn't conider that the mesh is now partially split and its parts may have been displaced. The result is that to select something, one has to click the location where it would be if it had not been displaced.
To set this right, the the overlay function was modified so that it would also be rendered three times like the normal rendering.

Milestone 2: find a safe distance, find a split plane

The next step is now to automatically displace place the two halves of the objectalong the explosion direction so that the object of interest is fully visible from every angle. To achieve this an

animated displacement of the object has two big advantages over statically placing the objects at the safe distance:

- **Aesthetics** A smooth transition of an objects movement from one point to another is conceived as more natural and therefore better-looking than an aprupt change of position. This conception can be reinforced by using animation techniques like slow in and slow out. [1]

- **Simplicity** Rather than finding the ideal distance by deterministically calculating it before displacing the object a much simpler approach is to guess such a distance by checking, if the object of interest is fully visible and then displacing the exploding parts otwards as long as this is not the case.

With these considerations in mind an implicit approach to finding the ideal offset of the two halves was used:
To see if the object is occluded it is drawn before the other objects that might occlude it and the number of pixels drawn to the screen is counted. Then, after the rest has been drawn the object of interest is drawn and its pixels are counted again. Because of the depth buffer, only the parts of the object that are not blocked from view are drawn, which means that if the both amounts of pixels from both renderings are the same the object is unoccluded.
To boost performance, the first rendering of the object of interest is done using the low cost overlay shader used in the color picking algorithm. Counting the rendered pixels of object of interest is done by using OpenGL queries.
Now it is possible to automatically generate an exploding view by making the offset of the exploding part grow until the object is fully visible. The offset grows at a constant speed which can be chosen in the user interface. If the object is unoccluded after a user interaction the offset shrinks until the first frame with a partial occlusion after which it grows again and stops when fully visible.
To achieve a more natural movement the relative amount of occlusion

$$r = \frac{p_{unoccluded} - p_{occluded}}{p_{unoccluded}} \tag{3.1}$$

for

$$p_{occluded} \neq 0 \tag{3.2}$$

can be used to generate an ease-out-effect: The speed is multiplied with r specified by user input resulting in the speed, making the animation faster when little of the object of interest is visible and slower if much of it can be seen, coming to a halt as soon as the whole object is fully revealed which is the moment the ideal offset is set to the current offset.
This movement is akin to the movement of the object being pulled by a spring toward the point of full revelation of the object of interest, though not a linear spring, because the change of speed is determined by the amount of Pixels that are revealed in each iteration making the spring constant proportional to r. To avoid that the Object still moves at very low speeds, resulting in huge amounts of unnecesary costly redraws, a minimum speed ϵ is introduced so that the object

comes to a halt earlier.

Milestone 3: Optimize Distance, force-field animation of split, optimize fringe distance cases

The objective of this last Milestone is to give a smooth appearance to the graphic while the view is changed by User interaction. To do this we define the animation as a transition between the current position and a target position, defined by the objects offset along the explosion axis. Initially the current offset set to 0, so the object appears its original loaction and the target offset is set to an arbirtary maximum offset of 20.0, which is close to the distance of the camera to the object centre. The first step is to make the transition between two offsets appear natural by adding an ease-out-effect to it. This is achieved by making the transition speed s proportional to Δ_{offset} between the current and the target offset,

$$s = s_u \cdot \Delta_{offset} \tag{3.3}$$

thus creating a movement with linear deceleration that comes to a halt when the ideal offset is reached. This behaviour can be observed if the option “Dynamic Offset “ is deactivated.
With dynamic ratio turned on and the object of interest (partially) occluded the ideal offset is set to the maximum offset and speed s is multiplied by the ratio r described in milestone 2 so that when the split parts move away from the object of interest the movement halts when the whole object is fully revealed setting the current offset as the ideal offset.
In case the object is revealed and the explosion needs to be collapsed the ideal offset is set to 0.0 until the object of interest is no longer fully visible, in which case the movement ideal offset is set to maximum and now grows outwards as described before.
This creates a visualization that smoothly adapts to new viewing points and changes in the splitting plane, but has one major disadvantage:
If the explosion direction points approximately in the same direction as the viewing vector, the offset needed to reveal the whole object of interest become very huge in comparison to the object itself. This may prove fatal to the expressiveness of the visualization, given that the goal is to represent an object in context of the parts that are exploded, but the distance between the components is either so large that parts of the components partially move outside of the screen or even completely outside or behind the viewing plane or it is necessary to zoom out or move the camera back so that the whole object is visible causing substantial loss of detail, due to the large offset. If the $plane\vec{N}ormal \cdot viewing\vec{V}ector = \pm 1$ or the object has a certain shape (e.g large at the end in direction of the offset, see 3.3) the offset would even grow to infinity.
A simple solution is now to define a maximum explosion distance so that no part can be placed too far away from its original loaction. By experimentation twice the total size of the mesh proved to be a distance that ensures that for most angles full visibility is achieved and the image stays compact. This distance o_{max} is the objects diameter which is acquired by calculating length of the distance between the minimum and maximum corners of the bounding box of the mesh. Since the mesh itself has no bounding box, the bounding box has to be accumulated by combining the bounding boxes of the groups of the mesh. This way the distance between the

Figure 3.3: Different objects of interest from roghly the same viewpoint cause the offset to grow, placing the object out of view

exploded parts can never exceed twice the diameter of the object.
To avoid that parts of the object of interest are now being completely occluded because of this a simple ghosting technique was implemented: The front part, meaning the exploded part that is between the viewer and the split plane, is being rendered translucently if the offset grows to be close to o_{max}. If the current offset o_c exceeds $o_{max} \cdot 0.7$ the opacity α of the front part is linearly interpolated between 1.0 at $o_c = o_{max} \cdot 0.7$ and 0.5 at $o_c = o_{max}$ using the formula

$$\alpha = \frac{o_{max} - o_c}{o_{max} \cdot 0.3} \cdot 0.5 + 0.5 \tag{3.4}$$

This opacity factor α is then multiplied by the occlusion r 3.1 so that the object stays solid while its not occluding anything and is rendered the most translucent if the whole object of interest is occluded completely.
To determine which part is the front part, I calculated the dot product of the viewing vector and the plane normal, using its sign to determine whether the part shifted in direction of the plane normal is the front part or not. This also determines in which order the parts are rendered, with the front part being the last, so that it can be translucently blended over the solid parts.
A problem that now appears is that backface culling is deactivated to so that the cutaway can be rendered correctly, resulting in the translucent objects' backfaces also visible, which causes the graphic to appear slightly confusing and aesthetically unpleasant. This problem of transparent faces shining through has been mentioned as one of the pitfalls when implementing ghosting by Bruckner and Gröller [5]. This can be avoided by allowing backface culling for a translucent object, if its offset vector doesn't point away from the camera. Because this may be the case if the camera is placed between the original and the translated switch plane, the dot product has to be calculated once more, now for the translated split plane.

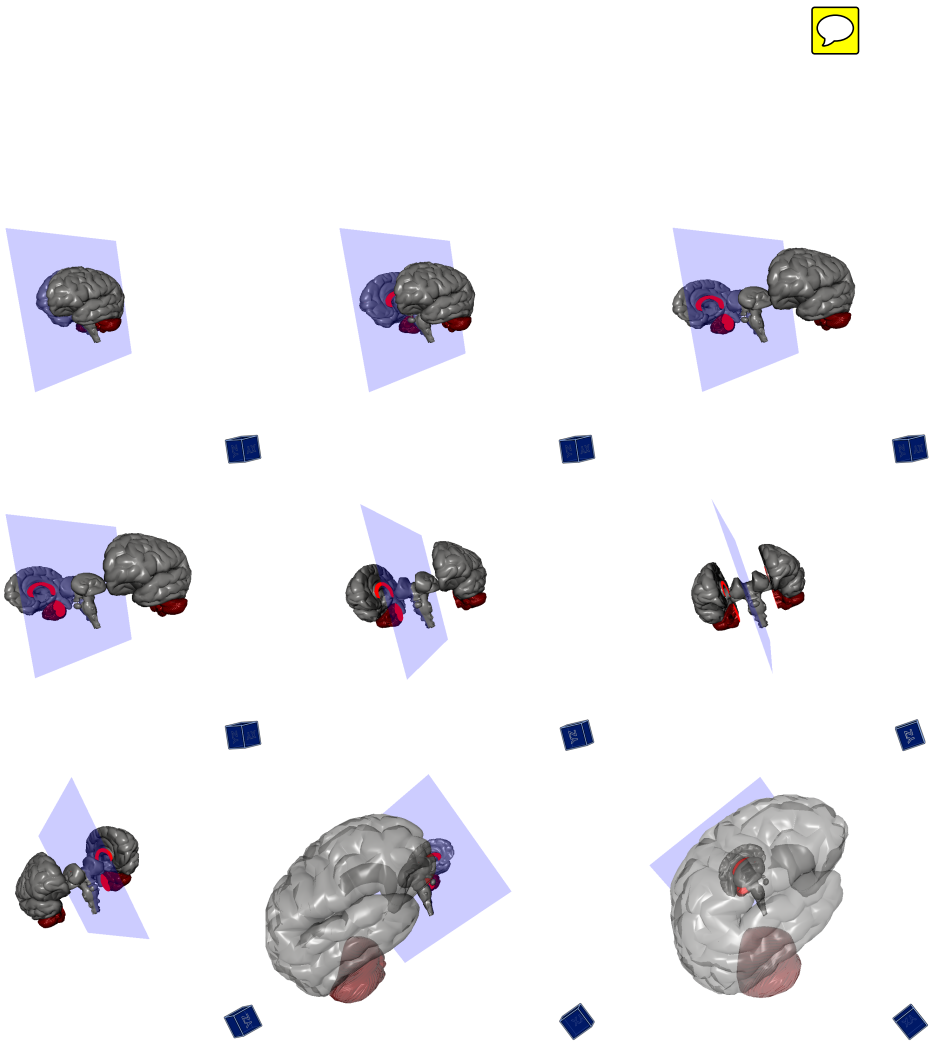

Figure 3.4: Depiction of the animations and ghosting in the final version of the plugin

CHAPTER 4

Conclusion

When given the possibility to view an interactive visualisation from all angles, the user is inclined to do so to satisfy his curiosity about the grapchis' subject. With exploded views alone the expressiveness of the visualzation suffer from this, when looking in the explosion direction. The combination of ghosting and exploded views provides a possible solution for this sacrificing partially the advantages of both techniques:

- Exploded views lose the property of showing the object of interest as well as the other parts in full detail, possibly appearing cluttered by the overimposed ghosted object.
- Ghosting on the other hand loses one of its key advantages, namely that all parts of the object appear in their original location.

What outweighs these disadvantages, especially in illustrations of technical assemblies, is that the explosion direction is always preserved and the original position of a displaced object can be easily identified, whereas in e.g. the force-based solution described in chapter 2 the displacement direction is not that clear.
Because the aim is to create a simple and intuitive interactive visualisation, the usability of the system is what bothers me the most though the zooming and viewing direction control is fairly intuitive.

4.1 Future Work

At the moment the explosive view is very basic, with the explosion only along one axis, and a hierarchy that consists of only an object of interest and its context objects. The next steps to imrove uüpon the visualisation would be to introduce more levels into the part hierarchy and devise a way to explode along mutible axes and exploring ways to quasi-naturally animate these behaviours. As ghosting concerned, an interesting addition would be to reduce the opacity of the blocking object only where it ocludes the object of interest and the vicinity of this area, with a smooth transition from transparent to solid in the area around the object of interest.

Furthermore a quanitative evaluation in a user study comparing different visualizations of the same Problem would be interesting, along with a usability overhaul of the User interface reducing its complexity and giving the user a better experience by enabling him to manipulate the view with a touch interface.

Bibliography

[1]

[2] Maneesh Agrawala, Wilmot Li, and Floraine Berthouzoz. Design principles for visual communication. *Commun. ACM*, 54(4):60–69, April 2011.

[3] Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. Designing effective step-by-step assembly instructions. *ACM Trans. Graph.*, 22(3):828–837, 2003.

[4] Stefan Bruckner and Meister Eduard Gröller. Volumeshop: An interactive system for direct volume illustration. In H. Rushmeier C. T. Silva, E. Gröller, editor, *Proceedings of IEEE Visualization 2005*, pages 671–678, October 2005.

[5] Stefan Bruckner and Meister Eduard Gröller. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084, 9 2006.

[6] Wilmot Li, Maneesh Agrawala, Brian Curless, and David Salesin. Automated generation of interactive 3d exploded view diagrams. *SIGGRAPH 2008*, pages 101:1–101:7, August 2008.

[7] Niloy J. Mitra, Yong-Liang Yang, Dong-Ming Yan, Wilmot Li, and Maneesh Agrawala. Illustrating how mechanical assemblies work. *Commun. ACM*, 56(1):106–114, 2013.

[8] Barbara Schwankl. Splitting of meshes in image-space, 2013.

[9] Ivan Viola and Meister E. Gröller. Smart visibility in visualization. In *Proceedings of the First Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging*, Computational Aesthetics'05, pages 209–216, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[10] Wikipedia: Illustration. http://en.wikipedia.org/wiki/illustration. Accessed: 2015-12-17.