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: Ma. Sc. Peter Mindek
Supervision assistant 2

Wien, 12.12.2013

(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: Ma. Sc. Peter Mindek
Supervision assistant 2

Vienna, 12.12.2013

(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)

Danksagung

Hier fügen Sie optional eine Danksagung ein.

Acknowledgements

Optional acknowledgements may be inserted here.

Kurzfassung

Hier fügen Sie die Kurzfassung auf Deutsch gemäß den Vorgaben der Fakultät ein.

Abstract

According to the guidelines of the faculty, an abstract in English has to be inserted here.

Contents

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.

Illustrations have been since the paleolithic age [7] to give their viewers a graphical representation of things seen, experienced or imagined [9]. 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. [7]

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 [2]

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 isn't cut and stays in place, while the split parts are displaced, revealing said object.
In an interactive System this displacement -the "explosion" can be shown as an animation, with all parts starting at the original place and gradually moving into the desired place. The User can also choose, which part of the system is interesting to him, and view the object from different angles, which also results in an animated change of the part's location. This can lead to massive displacement of the parts, especially if the viewer is looking in the direction of the displacement axis.
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.

CHAPTER 2

State of the Art Description

2.1 Exploded Views

What are the main problems when creating exploded views?

To create an exploded view, a few considerations have to be made Li et al. provide a comprehensive list of these challenges in their 2008 paper [3].
First of all what do I want to tell with the image, in essence what are the objects of interest that should attract the viewers attention.
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. [3]

Especially if the graphic conveys technical instructions or how-things-work-descriptions descriptions these axes are usually the directions in which they are assembled [1] or - if applicable the axis around which they rotate [4]. 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.
In more complex objects hierarchy of the exploding parts is another challenge to face.

> In many complex models, individual parts are grouped into sub-assemblies (i.e., collections of parts). [3]

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.

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 would be to use a view dependent force-based displacement behavior as suggested by Bruckner and Gröller [2]: 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 magnitude 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.

What solutions for these Problems exist?

Explosion graph - Hierarchical explosion graph - force-based placement

What are the advantages and disadvantages of exploded views?

2.2 Ghosting

What is the purpose of Ghosting?

show interior or obstructed parts - information about foreground hinted -

What are the main problems when visualizing objects using Ghosting?

good values for α - smart visibility -

What are the advantages and disadvantages of Ghosting?

Objects can be visualized “Where they Are” - compact - some information always lost -

2.3 Combination of these two techniques

CHAPTER 3

Practical part

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. I built my work upon an Existing plug-in, that split meshes in image space.

3.1 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 antural force-field animation and prevent unnecessary large offsets by introducing ghosting techniques.

3.2 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.

3.3 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 a manually defined offset, from a manually defined split plane. The fragment shader then rejects fragments that were be-

hind or in front of the also translated split plane, rendering them in a predetermined fashion.
The first step towards an exploded view was now to use the already implemented group selection feature to define an object of interest that would not be split or translated like the rest of the mesh. To realize this I introduced a third rendering of the mesh in the display function. This render pass would render only the object of interest in its original location in the non-displaced mesh.
This also required that the "renderMesh"-function be modified, introducing a new boolean parameter "split" that tspecifies, if the object is rendered in split mode, or in the "object-of-interest-mode". When the function loops through the groups of a mesh, it checks if the the group is in the group . If that is the case and the function parameter "split" is false, the group will be rendered. If split is false and the group is not selected, it will also be drawn, this time displaced.
Also a new option "split" for the shader was introduced, given that there is no need to pass an offset or split plane to render the object of interest. This basically reverts the fragment shader to the original trianglemesh-shader
After that was done an exploded view of the object was now rendered with manual definition of the offset and split plane with one usability glitch: A colour picking algorithm was already implemented, but it didn't consider the splitting and translation of the object. This resulted in a behaviour where clicking on a part of the object when it was split would cause false selection or deselection.
To set this right, I modified the the overlay function so that it would also be rendered three times like the normal rendering, modifying the "renderGroup" function once more so that it could also render the overlay function and adding an "overlay"option to the shader.

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

The next step would be to automatically place the two halves of the object so that they would not collide with the object of interest and to find a suiting plane to split the object.
As I was aiming for an animated transition between offsets, I chose an implicit approach to finding the ideal offset of the two halves:
Each time the object would be rendered I would first render the three parts (Object of interest, front half, back half) with the low-cost overlay shader counting the rendered pixels of the non-occluded object ($p_{unoccluded}$), counting the amount of pixels drawn using openGL occlusion queries and during the actual rendering counting the amount of pixels drawn with possible occlusions ($p_{occluded}$). The ratio r determined by

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

for

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

is multiplied with the speed specified by user input resulting in the speed at which the offset grows toward an ideal offset which is the maximum offset described in Milestone 3 or an offset of 0 if the difference between ($p_{unoccluded}$) and ($p_{occluded}$) is 0 with the ideal offset different than the maximum offset.

Because the ratio tends towards 0 the more the object is revealed the growth of the offset diminishes the more is revealed , 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,I introduced a minimum speed ϵ 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. The first step is to make the transition between two offsets smooth with the transition speed s proportional to Δ_{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 is (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 a plane normal on either side of the plane 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) the offset would even grow to infinity.
To circumvent this problem I defined a maximum offset o_{max} of the objects diameter which is the 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 exploded parts can never exceed twice the diameter of the object.
To avoid parts of the object of interest now being occluded I used a simple ghosting technique: The front part, meaning the exploded part that is between the viewer and the split plane, is being rendered translucently if the distance becomes too large. 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 r so that the object stays solid while its not occluding anything and is 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 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.

CHAPTER 4

Bibliographic Issues

bruckner-2006-EVV [2]
ruiz [6]
Li:2008 [3]
og/AgrawalaPHHKHT [**?**]
roc:conf/si3d/NiederauerHAH03 [5]
Viola-05-Smart [7]
journals/cacm/MitraYYLA13 [**?**]
Ward:2010:IDV:1893097 [8]

Bibliography

[1] 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.

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

[3] 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.

[4] 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.

[5] Christopher Niederauer, Mike Houston, Maneesh Agrawala, and Greg Humphreys. Noninvasive interactive visualization of dynamic architectural environments. In Michael Zyda, Michael V. Capps, Randy F. Pausch, and Gary Bishop, editors, *SI3D*, pages 55–58. ACM, 2003.

[6] Marc Ruiz, Ivan Viola, Imma Boada, Stefan Bruckner, Miquel Feixas, and Mateu Sbert. Similarity-based exploded views. In *Proceedings of Smart Graphics 2008*, pages 154–165, 2008.

[7] Ivan Viola and Meister Eduard Gröller. Smart visibility in visualization. In B. Gooch W. Purgathofer L. Neumann, M. Sbert, editor, *Proceedings of EG Workshop on Computational Aesthetics Computational Aesthetics in Graphics, Visualization and Imaging*, pages 209–216, 5 2005.

[8] Matthew Ward, Georges Grinstein, and Daniel Keim. *Interactive Data Visualization: Foundations, Techniques, and Applications*. A. K. Peters, Ltd., Natick, MA, USA, 2010.

[9] Wikipedia: Illustration. http://en.wikipedia.org/wiki/illustration. Accessed: 2013-12-12.