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, 12.8.2013

(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, 12.8.2013

(Signature of Author) (Signature of Advisor)

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

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

1.1 General Information

This document is intended as a template and guideline and should support the author in the course of doing the bachelor's thesis. Assessment criteria comprise the quality of the theoretical and/or practical work as well as structure, content and wording of the written bachelor's thesis. Careful attention should be given to the basics of scientific work (e.g., correct citation).

1.2 Structure of the Bachelor's Thesis

If the curriculum regulates the language of the bachelor's thesis to be English, the thesis has to be written in English. Otherwise, the bachelor's thesis may be written in English or in German. The table of contents is followed by the introduction and the main part, which can vary according to the content. The bachelor's thesis ends with the bibliography (compulsory) and the appendix (optional).

- Cover page
- Acknowledgments
- Abstract of the thesis in English and German
- Table of contents
- Introduction
  - motivation
  - problem statement (which problem should be solved?)
  - aim of the work

  - methodological approach
  - structure of the work

- State of the art / analysis of existing approaches
  - literature studies
  - analysis
  - comparison and summary of existing approaches

- Methodology
  - used concepts
  - methods and/or models
  - languages
  - design methods
  - data models
  - analysis methods
  - formalisms

- Suggested solution/implementation

- Critical reflection
  - comparison with related work
  - discussion of open issues

- Summary and future work

- Appendix: source code, data models, ...

- Bibliography

CHAPTER 2

Typographic Design

For working with LaTeXyou can take advantage of a variety of books and free introductions and tutorials on the internet. A competent contact point for LaTeXbeginners is the LaTeXWikibook, which is available under `http://en.wikibooks.org/wiki/LaTeX`.

The following sections give examples of the most important LaTeXenvironments and commands.

2.1 Tables

Tables have to be realized with the help of the *table* environment. Tables shall be sequentially numbered for each chapter and described in terms of a short caption (cf. Table 2.1).

Name	**Date**	**Title**
Mustermann Adam	18.5	T1
Musterfrau Eva	22.6	T2

Table 2.1: Seminar for Master Students

2.2 Figures

Like tables, figures shall be sequentially numbered for each chapter and described in terms of a short caption). You could either produce your drawings directly inside LaTeXusing PSTricks[1], Tikz[2], or any set of macros dedicated to your requirements (cf. Figure 2.1). Alternatively, you may include figures prepared in external tools (cf. Figure 2.2). Note, to ensure high quality printing, all figures must have at least 300 dpi.

[1] `http://tug.org/PSTricks`

[2] `http://sourceforge.net/projects/pgf`

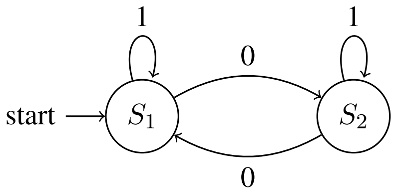

Figure 2.1: Sample figure

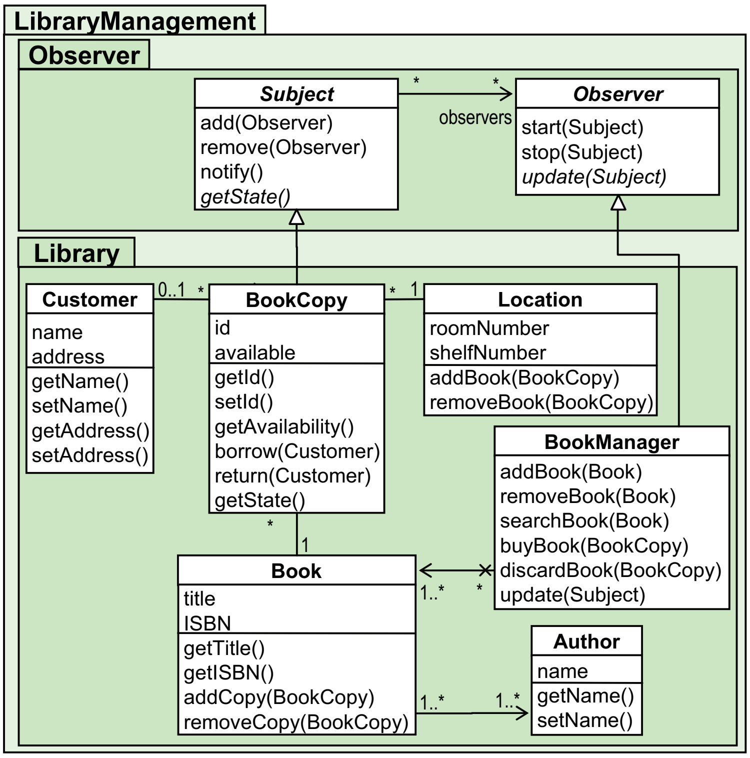

Figure 2.2: Sample figure

2.3 Fonts

When introducing important terms for the first time use *emphasize*. For a consistent look and feel of proper names like Class Diagram and `Observer` pattern you may define macros in the main document `thesis.tex`.

2.4 Code

For short code fragments use the *verbatim* environment.

```
//Start Program
System.out.println("Hello World!");
//End Program
```

A much better alternative is the *algorithm* environment (cf. Algorithm 2.1). This environment offers special formatting features for loops, operations and comments.

```
input  : A bitmap Im of size w × l
output: A partition of the bitmap
special treatment of the first line;
for i ← 2 to l do
    special treatment of the first element of line i;
    for j ← 2 to w do
        left ← FindCompress(Im[i, j − 1]);
        up ← FindCompress(Im[i − 1, ]);
        this ← FindCompress(Im[i, j]);
        if left compatible with this then ;                // O(left,this)==1

            if left < this then Union(left,this);
            else Union(this,left);
        end
        if up compatible with this then ;                  // O(up,this)==1

            if up < this then Union(up,this);
            // this is put under up to keep tree as flat as
               possible
            else Union(this,up);                     // this linked to up
        end
    end
    foreach element e of the line i do FindCompress(p)
end
```

Algorithm 2.1: Sample algorithm

CHAPTER 3

Introduction

Exploded Views are awesome, if you want to show the inner works of a complex Object, for example the gears of an engine or the inner organs of an animal. To visualize, that the viewer is seeing something, that would normally be occlouded by another part of the object, this part is is drawn cut in multible pieces and moved away from the Object of interest in question. This thesis explores the possibilities, limitations problems and possible solutions should one want to generate Exploded views automatically and documents the authors effort to create a plug-in for the visualization Framework Volumeshop that in combination with ghosting generates such an exploded view.

CHAPTER 4

State of the Art Description

4.1 Exploded Views

4.2 Ghosting

CHAPTER 5

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.

5.1 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.

5.2 Documentation of the implementation each milestone

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

T he 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 behind 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 that would render the object of interest only.

This also required that the “renderMesh”-function be modified, so that it checks whether or not a group is selected and depending on whether or not a split mesh is rendered draw the group or not. 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.
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” once more function 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

T he 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

$$r = \frac{p_{unoccluded} - p_{occluded}}{p_{unoccluded}} \tag{5.1}$$

for

$$p_{occluded} \neq 0 \tag{5.2}$$

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.

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}

$$s = s_u \cdot \Delta_{offset} \tag{5.3}$$

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 $plane\vec{N}ormal \cdot viewin\vec{g}Vector = \pm 1$ 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

$$\alpha = \frac{(o_{max} - o_c}{o_{max} \cdot 0.3} \cdot 0.5 + 0.5 \tag{5.4}$$

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 6

Bibliographic Issues

6.1 Literature Search

Information on online libraries and literature search, e.g., interesting magazines, journals, conferences, and organizations may be found at `http://www.big.tuwien.ac.at/teaching/info.html`.

6.2 BibTeX

BibTeX should be used for referencing.

The LaTeXsource document of this pdf document provides you with different samples for references to journals [3], conference papers [6], books [2], book chapters [7], electronic standards [5], dissertations [8], masters' theses [4], and web sites [1]. The respective BibTeX entries may be found in the file `references.bib`. For administration of the BibTeX references we recommend `http://www.citeulike.org` or JabRef for offline administration, respectively.

Bibliography

[1] Business Informatics Group. http://www.big.tuwien.ac.at. Accessed: 2010-11-09.

[2] M. Hitz, G. Kappel, E. Kapsammer, and W. Retschitzegger. *UML @ Work, Objektorientierte Modellierung mit UML 2*. dpunkt.verlag, 3. edition, 2005 (in German).

[3] Christian Huemer, Philipp Liegl, Rainer Schuster, and Marco Zapletal. B2B Services: Worksheet-Driven Development of Modeling Artifacts and Code. *Computer Journal*, 52(2):28–67, 2009.

[4] P. Langer. Konflikterkennung in der Modellversionierung. Master's thesis, Vienna University of Technology, 2009.

[5] OASIS. *Business Process Execution Language 2.0 (WS-BPEL 2.0)*, 2007.

[6] A. Schauerhuber, M. Wimmer, W. Schwinger, E. Kapsammer, and W. Retschitzegger. Aspect-Oriented Modeling of Ubiquitous Web Applications: The aspectWebML Approach. In *Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS '07), March 26-29, Tucson, Arizona, USA*, pages 569–576. IEEE CS Press, 2007.

[7] W. Schwinger and N. Koch. Modeling Web Applications. In G. Kappel, B. Pröll, S. Reich, and W. Retschitzegger, editors, *Web Engineering*, pages 39–64. John Wiley & Sons, Ltd, 2006.

[8] M. Wimmer. *From Mining to Mapping and Roundtrip Transformations - A Systematic Approach to Model-based Tool Integration*. PhD thesis, Vienna University of Technology, 2008.