Dynamic volume deformation using surfels

Olafur Thor Gunnarsson

A dissertation submitted in partial fulfilment of the requirement for the award of the degree of MSc in Computer Game Technology

University of Abertay Dundee

School of Computing and Creative Technologies

October 2010

**University of Abertay Dundee**

Permission to copy

Author: Olafur Thor Gunnarsson

Title: Dynamic volume deformation using surfels

Qualification: MSc Computer Games Technology

Year: 2010

1. I certify that the above mentioned project is my original work
2. I agree that this dissertation may be reproduced, stored or transmitted, in any form and by any means without the written consent of the undersigned.

Signature …………………….........................

Date …………………………..

**Abstract**

**Acknowledgements**

Thanks to:

My beautiful wife, Hrafnhildur and our son, Mikael Máni. For their love and support throughout my studies.

**Table of contents**

Contents

[1. Introduction 1](#_Toc270597211)

[2. Literature Review 3](#_Toc270597212)

[2.1 Mesh-free methods 3](#_Toc270597213)

[2.1.1 Voxels 3](#_Toc270597214)

[2.2.2 Surfels 4](#_Toc270597215)

[7. References 7](#_Toc270597216)

[8. Image references 8](#_Toc270597217)

[9. Bibliography 9](#_Toc270597218)

**List of tables**

**List of figures**

[Figure 1 - Fallout 3 nuclear weapon launcher (Sigger 2008) 1](file:///C:\Dropbox\My%20Dropbox\Skoli\Programming\Masters\Dissertation%20-%20Olafur%20Thor%20Gunnarsson%20-%200900128.docx#_Toc270621731)

[Figure 2 - 15 Cube triangulations (Lorensen and Cline 1987, p165) 3](file:///C:\Dropbox\My%20Dropbox\Skoli\Programming\Masters\Dissertation%20-%20Olafur%20Thor%20Gunnarsson%20-%200900128.docx#_Toc270621732)

[Figure 3 - Voxel sphere of resolution 20, 40 and 80 respectively (Author’s student project) 3](file:///C:\Dropbox\My%20Dropbox\Skoli\Programming\Masters\Dissertation%20-%20Olafur%20Thor%20Gunnarsson%20-%200900128.docx#_Toc270621733)

[Figure 4 - Surfel 4](file:///C:\Dropbox\My%20Dropbox\Skoli\Programming\Masters\Dissertation%20-%20Olafur%20Thor%20Gunnarsson%20-%200900128.docx#_Toc270621734)

[Figure 5 - Charlemagne model (600.000 points) with 2.000, 10.000, 70.000 and 600.000 surfels respectively (Gross and Pfister 2007, p 133) 4](file:///C:\Dropbox\My%20Dropbox\Skoli\Programming\Masters\Dissertation%20-%20Olafur%20Thor%20Gunnarsson%20-%200900128.docx#_Toc270621735)

[Figure 6 - Elliptical surfels covering a smooth 3D surface. The surfels draw over the adjacent surfels to cover up holes (Pajarola, Sainz And Guidotti 2004, p 599) 5](file:///C:\Dropbox\My%20Dropbox\Skoli\Programming\Masters\Dissertation%20-%20Olafur%20Thor%20Gunnarsson%20-%200900128.docx#_Toc270621736)

# 1. Introduction

Computer game realism has improved tremendously since the days of Doom and Wolfenstein 3D. Bill Gates said in a promotional video for Windows ’95 where he was superimposed into Doom 2: “These games are getting really realistic” (Matthewandrewtaylor 2006). Fifteen years later this statement is still true, games are getting really realistic. But what is video game realism? Is it the visual quality of the game world, or is it the behaviour of the world as the player expects it to be? Video game realism is a combination of the two, it needs to look well and behave properly. There is no reason creating the most real looking video game when there is no logic to the world behaviour.

The game world behaviour has many layers, from physics to night and day cycles. The behaviour that will be covered in this project is the volumetric deformation of walls. This feature is more often than not excluded from modern big budget games. Fallout 3 (Bethesda Game Studios 2008) is a perfect example of this discrepancy. In this game the player is given a rocket launcher that fires mini nuclear weapons. While this weapon works wonders on killing mutants and any other computer generated villain, it does no damage to structures. In a computer generated world that is so rich of interesting characters and beautiful graphics, this omission does make the game world feel like it lacks something. The behaviour of the game world therefore plays an important role in maintaining the user immersion.



Figure - Fallout 3 nuclear weapon launcher (Sigger 2008)

A few reasons exist why this is not incorporated into every 3D realistic game. The first one is because static environments simplify the game development by a large margin. In a game world where structures are static, a single wall’s vertices can be pre-calculated and placed into a vertex buffer. This vertex buffer is in turn used when the wall needs to be drawn. If the wall is dynamic and destructible there are many systems that need to be updated. First of all there needs to be an internal mechanism that controls how the wall deforms, which needs to be updated every time a collision occurs. In addition to the interior, the exterior needs updating as does the system that handles the collision detection. In addition to these steps, the deformation needs to be realistic, which is something which can be hard to do.

This project will aim to find a realistic and yet efficient method of deforming 3D volumes dynamically in video games. The methods used in this project will be a combination of surface and physical elements (surfels and phyxels respectively). Where the surfels are used to simulate the exterior of the model, but the phyxels simulate the interior.

# 2. Literature Review

This chapter is dedicated to the immense research and knowledge that exists in the field of volumetric deformation. Each method will be summarized and discussed with special focus on this project.

## 2.1 Mesh-free methods

This sub-chapter will cover the various mesh-free methods for volume visualization.

### 2.1.1 Voxels

The Marching Cubes algorithm was introduced in 1987 by Lorensen and Cline for 3D visualization of medical data, such as magnetic resonance (MR) and computed tomography (CT) scans. This algorithm takes eight voxels (points) that make up a cube and checks if the eight points are inside or outside of the model. Since each point can be in two states the triangulation possibilities are 28 = 256. Through intuitive thinking, Lorensen and Cline reduced the number of possible triangulation from 256 down to 15 which can be seen in Figure 2. This reduction comes from the use of inverses and rotation, as can be seen from case 0 which happens when all or no points are in the volume. Alternatively case 1 happens when only one point is inside or outside of the volume. When a single cube has been triangulated, the algorithm marches to the next cube (Lorensen and Cline 1987).



Figure - 15 Cube triangulations (Lorensen and Cline 1987, p165)

However, this method does have its downside because the point set needs to be dense enough so the volume does not look triangulated. In Figure 3 three voxel spheres were created using the marching cube algorithm. These spheres are all of equal size, but vary in their resolution and voxel size. The leftmost sphere is 20 voxels in height, depth and width and has a voxel size of 1. This results in a very course representation of the sphere. When the resolution becomes higher than 40, the voxelized object starts to resemble a sphere. At this resolution the sample point count is 64000. It is therefore apparent that a detailed model will need a lot of sample points.



Figure - Voxel sphere of resolution 20, 40 and 80 respectively (Author’s student project)

The reason why this method was not chosen is because of the large point set size. When the volume is deformed, the marching cube algorithm needs to be run again. If the point set is very large, the marching cube algorithm takes a while to finish and the fact that a large number of these voxel cubes will result in no triangulation.

### 2.2.2 Surfels

Over the last decade, research for point based graphics has been ever increasing. Surface elements (surfels) have become very popular when it comes to point based graphics, mainly because of their efficiency to approximate surface (Gross and Pfister 2007). A surfel is defined as a point with a normal and two tangent axes that define an ellipsoidal plate. This plate can in addition contain surface information, such as texture coordinates or colour. Figure 4 shows an image of a surfel with normal **N**, two tangents **t1** and **t2** and position **P**. The tangents are tangent to each other as well as to the normal.



Figure - Surfel

A surfel based surface is created by splatting surfels onto the set of input points. If the point set is sufficiently dense and the surfels are correctly set up, the surfel surface can resemble the expected model. It is the author’s opinion that this method is more efficient than using voxels since only a set of exterior points is used to approximate the surface. However, this method does have its drawbacks. Firstly this method has considerable overdraw, because of the way surfels are used to create a watertight surface. It is apparent from Figure 6 that surfel based rendering techniques do draw over pixels that have been drawn already. Secondly, if the point set is obtained from a 3D scanning some additional work needs to be carried out to set up the surfels correctly. A popular method is to use the Moving Least Squares (MLS) algorithm.



Figure - Charlemagne model (600.000 points) with 2.000, 10.000, 70.000 and 600.000 surfels respectively (Gross and Pfister 2007, p 133)

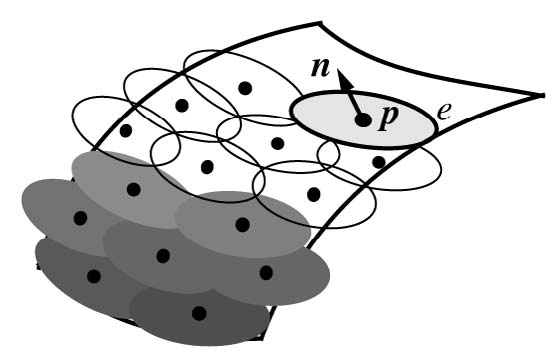


Figure - Elliptical surfels covering a smooth 3D surface. The surfels draw over the adjacent surfels to cover up holes (Pajarola, Sainz And Guidotti 2004, p 599)

# 7. References

Bethesda Game Studios. 2008. *Fallout 3.* [Disk]. Windows XP. Ubisoft.

Gross, M. and Pfister, H. eds. 2007. *Point based graphics.* Burlington, MA: Morgan Kaufman.

Lorensen, W. E. and Cline H. E. 1987. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics.* 21(4), pp. 163 – 169

Matthewandrewtaylor. 2006. *Bill Gates in Doom 2*. [Online]. YouTube. Available from: <http://www.youtube.com/watch?v=xh0JM7pD4qM> [Accessed 24 August 2010]

Pfister, H., et al. 2000. Surfels: surface elements as rendering primitive. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques.* New York: ACM. pp. 335 – 342. [Online]. Available from: <http://portal.acm.org/citation.cfm?id=344779.344936> [Accessed 21 April 2010]

Pajarola, R., Sainz, M. And Guidotti, P. 2004. Confetti: Object-Space Point Blending and Splatting. *IEEE transactions on visualization and computer graphics.* 10 (5), pp 598 – 608

# 8. Image references

Sigger, J. 2008. *No fat man for you!*. [Online image]. Available from: <http://armchairgeneralist.typepad.com/my_weblog/2008/11/no-fat-man-for.html> [Accessed 24 August 2010]

# 9. Bibliography