Simulating 3D deformation using connected polygons

J T Tarigan, I Jaya, S M Hardi and E M Zamzami

Faculty of Computer Science and Information Technology,
Universitas Sumatera Utara

jostarigan@usu.ac.id

Abstract. In modern 3D application, interaction between user and the virtual world is one of an important factor to increase the realism. This interaction can be visualized in many forms; one of them is object deformation. There are many ways to simulate object deformation in virtual 3D world; each comes with different level of realism and performance. Our objective is to present a new method to simulate object deformation by using a graph-connected polygon. In this solution, each object contains multiple level of polygons in different level of volume. The proposed solution focuses on performance rather while maintaining the acceptable level of realism. In this paper, we present the design and implementation of our solution and show that this solution is usable in performance sensitive 3D application such as games and virtual reality.

1. Introduction

Deformation effects such as fractured walls, shattering glass, or destructions of objects are becoming common feature in today's computer games. These effects may increase the realism of a scene and gives a better experience to the player by allowing a more immersive interaction between the player and the 3D objects. However, based on the approach used in the system, simulating deformation of a 3D object can be very expensive performance-wise. Creating a deformation with a high level of realism is complicated and requires complex calculation. Moreover, the deformation may be vary based on many properties such as point of impact, direction, and object rigidness. Examining all these properties may increase the level of realism but may significantly hit the performance.

Real time systems such as games and virtual reality, requires a certain amount of performance efficiency that does not allow complex calculation to be performed in-game. Hence, computer graphic researchers have developed algorithms that able to simulate object deformation in real time. Most of the methods proposed to simulate deformation in real time system requires pre-calculated information regarding the deformation process. This concept allows the system to simply go through the pre-calculated data to render the result. Other even proposed a pre-calculated model to simply replace the collided object with a deformed version of the object and add smoke and mirrors effects to hide the replacement process.

In our paper, we propose a method to implement deformation by using pre-calculated connected polygons. Each object contains multiple level of polygons that are connected to a polygon in a different level. These objects are then stored in a modified obj file format. Later, we use this obj file to simulate the object and its deformation in a 3D application. Our objective is to show that our algorithm can successfully simulate object deformation with the least amount of processing time. Moreover, we would also like to implement our method in a 3D computer graphic application to show the usability of our system. This paper is structured as follow: in the next section we will go through some works
that has been done previously related to our case. Section 3 will describe the algorithm of our method followed by the thorough description on how we implemented the system. The later section will discuss the test and results followed by conclusion.

2. Related Works

Object deformation is not new in computer graphic research. One of the first and pioneer in object deformation simulation was presented by Terzopoulos et al. [7] where they presented a formulation of deformable models by using reference component. There are a lot of methods algorithm introduced to simulate object deformation in 3D world. Usually, the algorithm is categorized into two groups; real-time and pre-calculated. The real time methods focus on simulating the deformation in-game. Algorithm that falls into this category mostly considers user interaction such as the point of impact. Hence, the result may be more accurate but there will be a performance hit. Parker et al. introduces a real-time deformation and fracture suitable for games [1]. The deformation process is based around a corotational tetrahedral finite element method. The authors also provide an example of the process implemented in published console game. An interesting work from Glodu et al. shows how fracture patterns in a photograph can be used as samples to create cracks and fractures in 3D objects [2]. While the authors argue that the algorithm can be performed in real time, it requires sampling/learning processes that may take 15 to 25 minutes based on the complexity of the sample. Brown et al. presented a new method to simulate object deformation in surgical training 3D application [3]. The algorithm developed is complex due to the accuracy required in detecting collision and performing deformation. Yet, the paper shows that the deformation process can be done in real time. Bao et al. proposed a hybrid approach to simulate fracture on rigid materials [5]. The system uses rigid body simulation for efficient dynamic evolution and fully linearized, small displacement, finite elements to determine rich and realistic fracture pattern. Tang et al. developed a system that allows mobile users to interact with 3D computer graphic application and perform object deformation [6]. However, due to the nature of complexity of the deformation process, the calculation is done by server and streamed to user's mobile phone.

The second category is pre-calculated method where the deformation polygons are built before the application starts. This method is common in games where performance in critical and deformation details are less required. Clothier et al. introduces a method to create a destructible object using 3D Voronoi Subdivision tree [4]. This method divides a 3D object into two Voronoi cell regions and store the subdivisions in a tree. Based on the impact on the object, the system will move downward to the lower level of the tree. This process is repeated until it reached the lowest level of the tree which means the object has been completely removed from the scene. Another precomputed method is presented by Muller et al. [8]. The method perform a computation to find a linear stiffness matrix to be used during runtime.

3. Deformation Based on Connected Polygons

The basic idea of our method is simple: each polygon is connected to other polygon based on their connection to each other. There are two relation between polygon in an object. The first relation is group. Polygons connected in the same group will act equally. If one of the polygon is deformed, the other polygons that are connected in the same group are also deformed. Each group has a starting polygon that starts the deformation process. The same polygon will also activate the starting polygon in the lower leve. The second relation is level. Each polygon contains a level. This level value defines the layer of the polygon where polygon at outmost layer will have level 0.
The deformation process goes as follows: a collision triggered a group and deformation process is activated. The group's starting polygon will set itself invisible and trigger the next polygon in the same group deactivate. This process is continued until it reached the starting polygon. At the same time, the starting polygon will trigger the starting polygon from the next level to activate itself and other polygon in the the same group. The deformation process is shown in the next figure.

4. Implementation

4.1 Building the Object

The first step of the implementation is to build a 3D objects with multiple layer of polygons. We use a common 3D content generation application for this process. The creation process starts with the non-deformed shape of the objects, which contains polygons with the lowest level 0. The polygons are grouped based on their position. Later we deformed the polygon and increase the level by 1. This process is repeated until we've reached the smallest deformed shape of the object. To simplify this process, we treat each group as a single object since we have to manually merge the polygon in every level manually. Figure 2 shows the non-deformed object grouped by colour.

4.2 Exporting to obj File Format

Since we have to export each state of the object as a single obj file, it is important to merge all the obj file into one file. Unfortunately, there is no easier way to do this, we have to go through each obj, find the corresponding polygons and group them. We also have to enter the group and level information to
the obj file manually. This information is attached in the end of each line of face data. Figure 3 shows how the modification is made to add an information that the corresponding face is group 4 and level 2.

| Original obj Format |
|---------------------|
| f 62/66/80 63/67/81 57/69/82 |

| Modified obj Format |
|---------------------|
| f 62/66/80 63/67/81 57/69/82 4 2 |

Figure 4. Modified obj format that contains group and level data

4.3 Testing the Object in Application
To test wether the method can sucessfully simulate deformation in real time, we develop a 3D computer graphic application. The application is built using C++ and OpenGL library. To help the windowing and I/O, we use GLUT library. At our current test, we do not collect performance data (such as frame per second) since the 3D object used in the test is too simple to affect the performance. Figure 4 shows the application running as a desktop application.

Figure 5. 3D Application to test the deformation process

Based on our observation during the test, the proposed method is capable to simulate deformation. There is no performance or memory issue arise during the test.

5. Conclusion and Future Works
We have shown that our connected polygon method is capable in simulating object deformation in real time. Our proposed system uses predefined 3D object with deformation data. Each polygons of the 3D object is connected to other polygon. When a deformation is triggered, the system will go through connected polygon to find which polygon is activated and which is deactivated.

There are a lot of improvement that needed to be added in the future. The most significant problem is the grouping and leveling process is done manually by modifying the obj file. The process is exhaustive considering an object may contain thousands of polygon. Modifying thousand lines of faces information is exhaustive and time consuming. In the future, we would like to do these process automatically by creating a plug-in in the 3D content generation application. Hence, the deformation modelling and object exporting can automatically creates a file based on the method we proposed.

6. Acknowledgement
The authors would like to thank the head and staff of "Lembaga Penelitian USU (Research Center of Universitas Sumatera Utara)", Prof. Dr. Erman Munir, MSc, and staff for all the support during this research. The author would also like to thank the Dean of Faculty of Computer Science and
Information Technology Universitas Sumatera Utara, Prof. Dr. Opim Salim Sitompul, M.Sc for their widest to support this work.

References

[1] Parker E G and O'Brien J F 2009 Real Time Deformation and Fracture in a Game Environment Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation 2009 p 165-175

[2] Glondu L, Muguercia L, Marchal M, Bosch C, Rushmeier H, Dumont G and Drettakis G 2012 Computer Graphics Forum vol 31 no. 4 p 1547-1556

[3] Brown J, Sorkin S, Bruyns C, Latombe J C, Montgomery K, Stephanides M 2001 Real-Time Simulation of Deformable Objects: Tools and Application Proceedings Computer Animation 2001. Fourteenth Conference on Computer Animation p 228-258

[4] Clothier M and Bailey M 2015 Creating Destructible Objects Using a 3D Voronoi Subdivision Tree Proceedings of 15th International Conference on Computational Science and Its Applications p 42-46

[5] Bao Z, Hong J, Teran J, and Fedkiw R 2007 Fracturing Rigid Materials IEEE Transactions on Visualization and Computer Graphics vol. 13 no. 2 p 370-378

[6] Tang Z, Ozbek O, and Guo X 2011 Rea-Time 3D interaction with deformable model on mobile devices Proceedings of the 19th ACM International Conference on Multimedia p 1009-1012

[7] Terzopoulos D and Witkin A 1988 Physically Based Models with Rigid and Deformable Components Proceedings of Graphics Interface Conference p 146-154

[8] Muller M, Dorset J, McMillan L, Jagnow R and Cutler B 2002 Stable real-time deformations Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation p 49-55