3D Object Morphing with Metaballs

Sindhu Kommareddy
University of Illinois at Chicago
1200 W Harrison St, Chicago, IL 60607 USA
bkomma2@uic.edu

Jed Siripun
University of Illinois at Chicago
1200 W Harrison St, Chicago, IL, 60607 USA
jsirip2@uic.edu

Jenny Sum
University of Illinois at Chicago
1200 W Harrison St, Chicago, IL, 60607 USA
sum1@uic.edu

ABSTRACT

This paper goes over the 3D morphing technique known as the Marching Cubes algorithm used to render isosurfaces in volumetric data. In this particular instance, this algorithm is used to morph 3D metaballs, otherwise known as blobby objects. The most important task for this type of morphing is to extract a polygonal mesh, from a three-dimensional scalar field, also known as voxels. The Marching Cubes algorithm is mainly used for medical visualizations such as CT and MRI scan data images. It has also been used for special effects in motion pictures, as well as 3D modeling with metaballs or other objects that have metasurfaces.

Author Keywords

Marching cubes; 3D Morphing; Polygonal mesh; Metaballs; Voxels; Isosurface

ACM Classification Keywords

D.3.2 C++

INTRODUCTION

3D computer graphics has become more important, increasingly popular, and more commonly used in different fields in recent years. 3D object or model morphing is one of the special effects in 3D computer graphics. In computer graphics terms, morphing is defined as an animated transformation of one image into another image. It is a special effect that has been used in many motion pictures and animations, where an image or shape seamlessly transitions from one image or shape to another through a type of warping or cross dissolving technique. In film, cross dissolving is often used. In this paper, we will be discussing a simple, but highly effective and useful morphing technique, using the Marching Cubes algorithm.

William E. Lorensen and Harvey E. Cline developed the Marching Cubes algorithm. It was the result of their research while they worked at General Electric to efficiently visualize data from CT and MRI devices.

The typical method of modeling surfaces is using parametric equations to define points on the surface. These points can be connected to form polygonal meshes. This morphing technique is very useful for performing transformations. An alternative to parametric methods is using implicit representations. This is when the surface is defined as the zero contour of a function with 2 or more variables. Implicit representations are popular to use when it comes to blending and metamorphosis.

Metaballs are a type of implicit modeling technique. One of the prevailing aspects of metaballs is the way they are able to combine with one another, resulting in a very smooth blending of spherical objects.

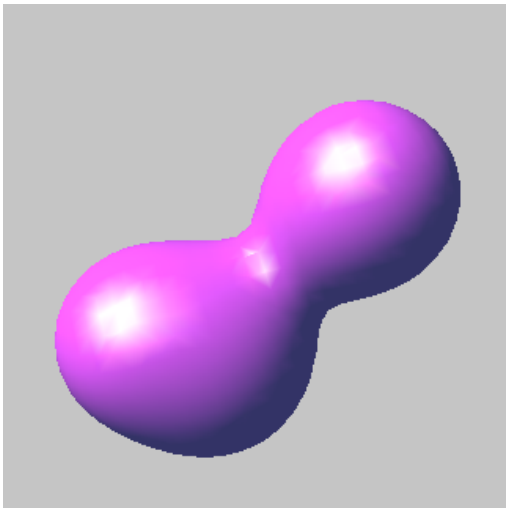

Figure 1. Two metaballs with a lower threshold, showing smooth blending when combined.

RELATED WORK

The Marching Cubes algorithm has been studied and used by many throughout the years. A popular paper written by Paul Bourke displays how this algorithm works, and how efficient this algorithm runs. It is simple, but has incredible speed because it works almost entirely on lookup tables.

The two applications of this technique that Paul Bourke focuses on, is the reconstruction of a surface from medical volumetric datasets, for example MRI scans, and creating a 3D contour of a mathematical scalar field. In this instance, the function is known everywhere, but they are sampled at the vertices of a regular 3D grid.

With the resolution of the sampling grid, Bourke was able to do a very desirable control. It allowed for the fine approximation to the isosurface to be generated on the smoothness required and the processing power available to display the surface. In figure 7, "bobby molecules", as Blinn calls them, are generated at different grid sizes to create a smoother surface.

ALGORITHM

The algorithm takes eight neighbor locations at a time through the scalar field. This forms an imaginary cube. Afterward, the algorithm determines the amount and types of polygons needed to represent the part of the isosurface that passes through this cube. Afterward, these individual polygons are fused into the desired surface.

The desired surface is formed by creating an index to a pre-calculated array of 256 possible polygon configurations within the cube. Each of the 8 scalar values is treated as a bit in an 8-bit integer. The algorithm then checks to see if the scalar's value is higher or lower than the iso-value. If the scalar's value is higher than the iso-value, this means that it is inside the surface, then the appropriate bit is set to one. If the scalar's value is lower han the iso-value, which means it is outside the surface, then the appropriate bit is set to zero. After all eight scalars are checked, the final value is the actual index to the polygon indices array.

In the final step of the algorithm, each vertex of the generated polygons is placed in the appropriate position along the cube's edge.

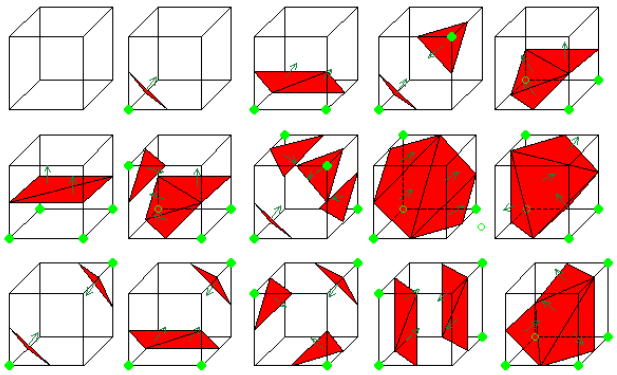

Figure 2. How the marching cube algorithm behaves.

METABALLS CREATION AND INITIALIZATION

The metaballs are organic-looking n-dimensional objects. The rendering technique of metaballs was invited by Jim Blinn. In this project, the metaballs are defined as a function in 3-dimensions, f(x,y,z).

$$\sum_{i=0}^{n} \text{metaball}_i(x,y,z) \leq \text{threshold}$$

The threshold value is chosen to define a solid volume, and the above inequality represents whether or not the volume enclosed by the surface defined by n metaballs is filled at (x,y,z).

This is the formula used for the metaballs.

$$f(x,y,z) = 1/((x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2)$$

(x0,y0,z0) is the center of the meatball. However, this can be very computationally expensive, so polynomial functions are used instead in the program.

The program apples a simple falloff curve to distance-from-surfaces. It uses this formula:

$$f(r) = (1-r^2)^2,$$

where r is the distance to the point. By using this formula, expensive square root calls are avoided.

In the program, there are five metaballs initialized. These five metaballs are initialized using a 3D vector that approximates location, and a float value that approximates size. In this particular program, there is one larger metaball and four larger metaballs that surround it.

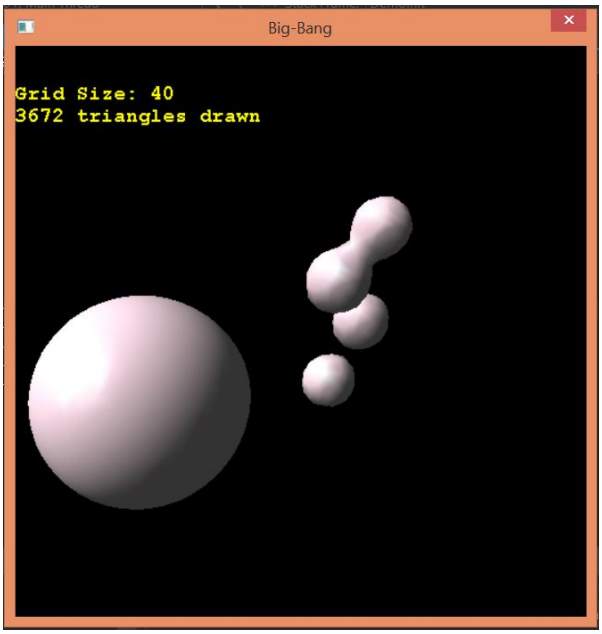

Figure 3. One larger metaball surrounded by four equal-sized smaller metaballs.

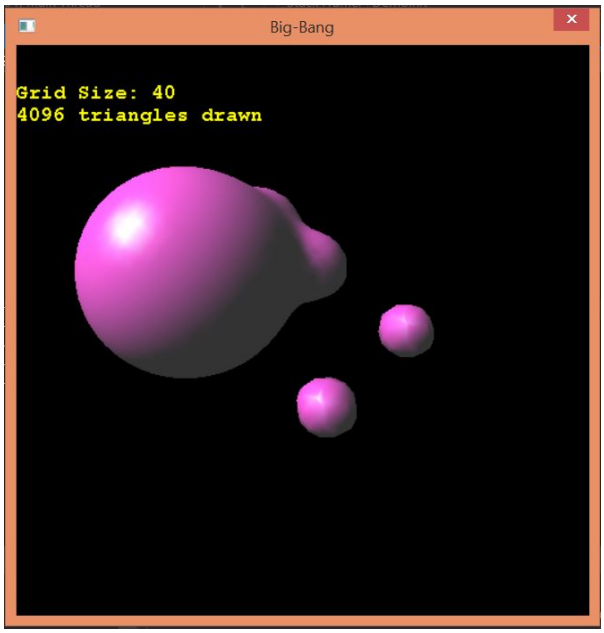

Figure 4. The metaballs change colors after the user presses the space key.

METABALL SIZES

The program allows the user to change the size of the metaball. By pressing the 'Q' key, the squared radius of the original cube is expanded by a floating point of 0.1. When the user presses the 'A' key, the floating point of 0.1 shrinks the squared radius of the original cube.

The reason why it is referred to as a cube, originally, is because of the marching cube algorithm. Though metaballs are of spherical shapes, the shape is created from the ancestral cube.

METABALL COLORS

The program allows the user to change the color of the metaballs into three different shades: green, magenta, and gray. The user can access these three different shades by tapping onto the space bar.

The colors are changed through the values in diffuse lighting of the phong shader. These values are passed into a function, and changed according to the key press.

WIREFRAMING

The program allows the user to do wire framing which creates a grid-like texture onto the metaballs. This grid-like texture shows the polygon formations on the surface of the metaballs. These polygon formations are the same surface polygons that are explained under the algorithms section.

When the user presses the 'W' key, the metaballs shows the wire framing. The user can toggle this function by pressing the 'F' function and filling the metaball with a smooth and glossy finish. This finish demonstrates the three types of lighting: ambient, diffuse, and specular lighting.

Figure 5. The wire framing of the metaballs are shown.

GRID SIZE

Along with being able to see the metaballs in as a wireframe, the user is able to increase or decrease the size of the cube grid. The cube grid is the cube in which the polygons from the wire framing live. When the cube grid size is increased, the polygons that are interpolated onto this cube is also increased.

When the user presses the up arrow key, the cube grid increases, and when the user presses the down arrow key, the cube grid decreases.

PAUSING METABALLS

The metaballs run on a timer function within the program. This timer is constantly running, and tracking the motions of the metaball. The metaballs are in constant motion, morphing, and fusing together.

When the user presses the 'P' key, the timer function is paused, and the motions of the metaballs come to a still. When the user presses the 'U' key, the timer is unpaused, and the motions of the metaballs are continued. By being able to toggle the program's timer, the user can clearly see the changes that are made in color and size in both the metaball radius as well as the grid.

Figure 6. The user has decreased the grid size of the metaballs.

CONCLUSION

The Marching Cubes algorithm is an algorithm that is used for 3D surface construction. It is often used for medical purposes such as MRI and CT scans, but can be used for 3D modeling and 3D morphing as well, as seen in the metaballs of this program.

Metaballs, also known as blobby objects, are an implicit modeling technique that has an interesting way of combining with one another to create a smooth blending between spherical objects.

In this program, the metaballs have the direct flexibility of changing sizes, shapes, and textures. The user is able to control time to better see these changes, and within the code, the amount of metaballs can be changed as well. The more metaballs there are, the more limited space is, and the more collisions occur.

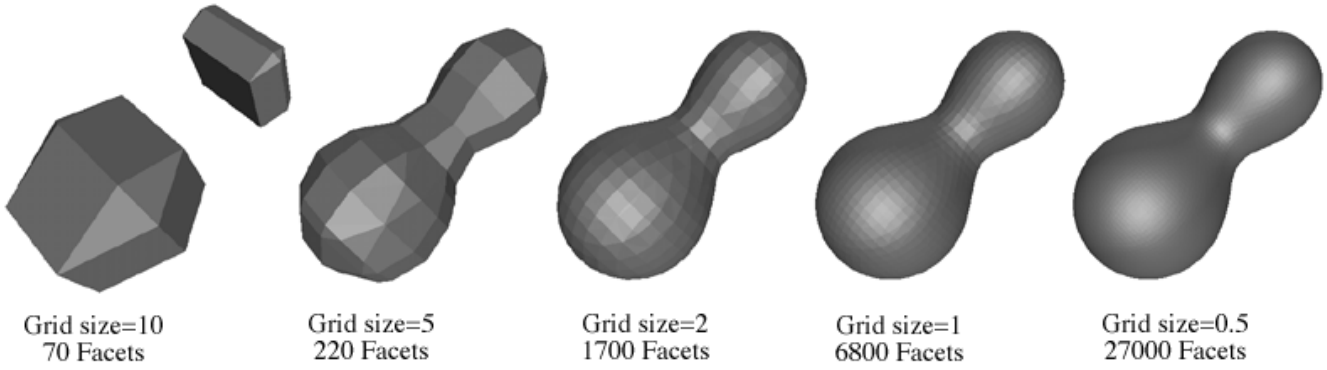

Figure 7: The "bobby molecules" that are generated at different grid sizes to create the appearance of a smoother surface.

ACKNOWLEDGMENTS

We would like to thank Professor Angus Forbes and Kyle Almryde for giving us the tools and knowledge that was necessary to complete this programming project.

REFERENCES

1. Blinn, J. (1982). A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics, 1*(3), 235-256.
2. Hansen, C., & Johnson, C. (2004). Visualization Handbook. 9-9.
3. Lopes, A., & Bordlie, K. (2005). Interactive approaces to contouring and isosurfaces for geovisualization. *Exploring Geovisualizing,* 352-353.
4. Lorensen, W., & Cline, H. (1987). Marching Cubes: A high resolution 3D surface construction algorithm. *Computer Graphics, 21*(4).
5. Menon, J. (1996). An Introduction to Implicit Techniques. *SIGGRAPH Course Notes on Implicit Surfaces for Geometric Modeling and Computer Graphics*.
6. Ward, M. (1999). An Overview of Metaballs/Blobby Objects. *Math and Physics*.
7. Ward, M. (1999). Overview of Marching Cubes Algorithm. *Math and Physics*.