ECSE 4750 Computer Graphics

Term Project

Jing Cheng, Carlos Lazo, Chris Rainey

December 6, 2006

0.1 Abstract

This paper will serve to give an insight into the world of Computer Graphics, specifically analyzing Bezier curves and surfaces in the OpenGL API. We begin with an introduction into the Computer Graphics field and its evolution throughout the past century. An overview of Bezier curves and surfaces will then be provided, inclusive of both the mathematical and OpenGL intricacies. Finally, an OpenGL program created by our research group will be demonstrated and explained in full detail, pointing out how we manipulated C code to create an oscillating Bezier surface animation.

0.2 What is Computer Graphics?

Computer Graphics can be described as a field of visual computation, where one uses computing power to create visual representations of real world images utilizing an immense variety of graphical techniques. The range that spans the Computer Graphics world in which we live is composed of the following main areas: real-time 3D rendering, computer animation, video capture, special effects, image editing, and modeling.

3D Computer Graphics came to exist after the creation of workstation computers. Most 3D graphics engines are created with built in systems that create and store points in three dimensional spaces. Geometric objects can be displayed, or rendered on a computer. Using coordinates to store locations, each side of an object can be treated as a flat surface bordered by three or more outlined edges. The computer calculates the distinct perspectives of each of the objects and proceeds to draw their specific outlines on the screen. Once this wire frame model is done, hidden line removal is utilized to ensure that only the edges visible to the viewer are drawn. Once this is done, then the objects can be appropriately drawn in three dimensional spaces and filled in to create the desired visual effect.

Shading is another very important area that the field of Computer Graphics encompasses. It involves simulating virtual light sources in three dimensional spaces and realizing its effect on all objects within the environment. There are many different types of shading (such as Flat, Gouraud, Phong, etc) all involving very different methods of point-space calculations and producing different visual effects. Shading plays a great part with Ray Tracing, which governs certain object qualities such as radiosity, reflection, shadows, surface texture (bumpiness), transparency, etc.

Researchers constantly search for innovative ways to harness the true computing power of processors in order to produce more life-like images and visual representations with each passing day. The Computer Graphics field has and will continue to be an exponentially growing field with limitless possibilities.

0.3 History of Computer Graphics

Computer Graphics has evolved drastically over the past 50-60 years. The movement began in the early 1950's where output was relayed using teletypes, lineprinters, and the Cathode Ray Tube (CRT). The oscilloscope was invented, utilized to record an image of oscillating frequencies onto high speed film. MIT also developed a computer that was the first to display real time video, along with having the ability to produce real time graphics on a large oscilloscopic screen.

The 1960's and 1970's proved to be an important time period for developmental graphics processes. The term 'Computer Graphics' was coined in 1960 by William Fetter, officially naming the newly founded graphical movement. The development of the first mouse along with Sketchpad, an interactive Computer Graphics system create by Ivan Sutherland, proved to be monumentally important steps in the graphical world. Such things such as line-drawing algorithms, frame buffers, the first Pong game fabricated in 1972, and the famous CG teapot fabrication using Bezier patches truly helped propagate further research in the field.

The 1980's and 1990's brought a great deal of creativity and innovation to the Computer Graphics era. Steven Lisberger helped animate the movie Tron. This was the first Disney movie which took an incredible advantage of 3D Computer Graphics. The birth of AutoCAD, the NES gaming system, and the use of CGI in movies made by Disney and Pixar was absolutely stunning in the eyes of the general public. The introduction of VGA and SVGA allowed for personal computing to easily display photographically realistic images and movies. The OpenGL API specification which took place in 1992 can be seen as one of the most recently crucial advancements in the Computer Graphics field.

In our current society, Computer Graphics will continue to play an important role in our every day lives. It has had a tremendous affect on society over the past half-decade, and will revolutionize the world around us. Applications can be seen in the medical field (computer vision/biometric scanners), military (battle simulations), crime-reenactments, etc. In the near future, one could only imagine the kinds of technologies that will be developed, for example in the fields of robotics and warfare. The history of Computer Graphics continues to be paved with each passing year - human curiosity and exploration will hopefully lead it to an incredible and promising future.

0.4 Our Main Program

Simply put, a Bezier surface is a surface patch which is defined by a Bezier curve in both directions. In our project we used Bezier surface patches to model the oscillations of a surface over time. We do this by specifying different weights for the control points, chosen in such a way as to produce a smooth animation.

The formula for a general Bezier surface made of $n \times m$ patches is

$$r(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \omega_{ij} P_{ij} B_i^n(u) B_j^m(v)$$

where

$$B_i^k(t) = \binom{k}{i} (1-t)^{k-i} t^i$$

for $k = m, n$ and $t = u, v$ are the Bernstein polynomials which are used to derive the equations of the Bezier curve in the u and v directions respectively. The terms P_{ij} are the control points of the Bezier mesh, and the ω_{ij} terms are the weights attached to each function. The way that a Bezier mesh draws a surface is by drawing a bezier curve in the u direction for each $v = constant$, and then drawing a Bezier curve in the v direction for each $u = constant$. Note that we specify the u and then the v direction as the way in which the surface is drawn, but this convention is arbitrary, and it turns out that we can draw curves in the v direction first and then the u direction and get the same surface.

The formula thus determines a mesh of points $P_{i,j}$, called control points, corresponding to the length of the Bezier curves in each direction. The control points then determine how the surface is drawn. Just as a Bezier curve in one dimension lies within the convex hull of its control points, we have that a Bezier Surface lies entirely within the convex hull of its control mesh. Also, the manipulation of the control points in both directions determines how the surface is manipulated.

The effect of this is that each control point can be varied independently from the other control points, thus easily allowing someone to make perturbations of arbitrary size in the value of the surface for some neighborhood of a point on the surface which corresponds to the point on the control mesh. This means that for a Bezier surface which is comprised of n patches defined by $(n+1)^2$ control points, we have $(n+1)^2$ degrees of freedom, since every control point represents a degree of freedom. In our project we manipulated the middle four control points, and therefore we animated a four degree of freedom system.

What we tried to accomplish in this project is to show an animation of a flat surface (i.e., a plane) that begins to oscillate in the middle of the patch at some initial time $t = 0$ and keeps on oscillating for any time $t > 0$. The way that we accomplished this is by weighting the middle four control points with sine and cosine functions of various amplitudes, and keeping the control points on the boundary of the patch fixed. We start the surface off flat, and then we have two of the four control points rise in the positive z direction by weighting them with a sine function which has a positive period, and a make the other two control points lower into the negative z direction by weighting it with a sine function with a negative period. Since we are taking the weights to be functions of time, we get an animation of a surface whose middle area rises and falls over time, with opposite corners going in opposite directions. Mathematically, we are manipulating the middle four control points: $P_{1,1}, P_{2,1}, P_{1,2}$, and $P_{1,3}$. Thus we make the corresponding weights $\omega_{1,1}, \omega_{2,1}, \omega_{1,2}$, and $\omega_{1,3}$ to be equal to the sine functions, and all the other weights are equal to one, since we want the twelve boundary points to remain fixed over time.

We admit that this may not exactly correspond to a physical situation such as a wave traveling through a media in a bounded volume, if we took the displacement of the media to be represented by the surface, since then the boundary control points would not stay fixed. However, this is a simple visualization and it illustrates OpenGL's ability to use Bezier surfaces to produce good animations with relatively simple mathematical formulations.

0.5 Term Project Team Contribution

With a coherent mathematical foundation for Bezier surfaces set in stone, our group decided to explore these complex geometries utilizing the OpenGL API. The analysis of three programs will help supplement the theoretical basis, along with providing one with a visual representation of Bezier curvature.

Our first program, Bezier3D_Grid_Edit.cpp, takes a look at a Bezier grid with the ability to modify control points in 2D space. Initially starting with a 2×2 or 4×4 grid of points, the user can alter the position of the individual control points and immediately notice the effects on the Bezier surface patch in real-time. Different options are given when right clicking on the screen, with a resizing option available for the window. The Bezier surfaces patches are setup with a set of defined control points defined within two matrices. OpenGL evaluators are used to render the patches in a 2D plane until the control points stretch it into a 3D geometry. The mouse position is compared to the current grid position to determine where the object should be located. This helps determine the priority of the surfaces in reference to the screen in 3D space.

Our second program, Bezier3D_Oscillator.cpp, created a 3×3 Bezier surface mesh. In this variation, the eight outer control points are fixed in the plane with the middle point oscillating with respect to time utilizing a sine function (as can be seen in the code). We have only one center point for this specific representation - for n^2 control points, there are $(n-2)^2$ central control points.

The remainder of our demonstrations (as can be seen in the Appendices) all create different order Bezier surfaces. Utilizing fixed points, alternating color schemes, and one light source for shading effects, the surface molds itself according to the fixed control points and oscillating time dependant sinusoidal control points. Many different examples are provided which all yield a variety of different animations.

All of the programs are filled with detailed comments and documentation in order to understand the basics behind Bezier surfaces in Open GL.

0.6 Conclusion

Bezier surfaces are extremely important geometrical entities in the Computer Graphics world. Modeling curvature and geometry using tangential calculations and control point manipulation can serve to be a great asset to many different applications and fields. As can be seen in the many different programs we created, simple planes can be deformed and stretched into many different configurations. Life-like imagery from the real world can be simply modeled using Bezier patches and proper control point orientation, such as mountains, oceans, vehicles, and even the human anatomy. We feel that Bezier surface analysis is an interesting and revolutionary field in Computer Graphics and have thoroughly enjoyed our exploration of this important topic.

0.7 Bibliography

http://en.wikipedia.org/wiki/Computer_graphics

http://www.comphist.org/computing_history/new_page_6.htm

http://www.opengl.org/resources/code/samples/mjktips/grid/index.html

http://www.ecse.rpi.edu/ wrf/wiki/Research/bezier.pdf

Wright, Richard S. and Benjamin Lipchak, *OpenGL SuperBible*. SAMS, Indianapolis, Indiana. 2005

Angel, Ed. *Interactive Computer Graphics : A Top−Down Approach Using OpenGL* Addison-Wesley, Boston, Massachusetts. 2006.