Key Frame Animation Made Easy With GraphicMentor

Francis Murphy
Department of Computer Science
Michigan Technological University
Houghton, Michigan 49931
fgmurphy@mtu.edu

Abstract

Traditionally the topic of key frame animation has been too complex a topic to be covered in introductory computer graphics courses. The difficulties in including this topic have stemmed from the need for relatively difficult mathematical concepts such as B-spline interpolation in order to generate frames for the animation. In this paper we present improvements to a pedagogical tool called GraphicMentor designed to allow students to visualize the basic concepts of computer graphic. The basic concepts illustrated are the specification and interaction of the camera, light sources, and objects. Our improvements to GraphicMentor include features to help students visualize the mechanics behind key frame animation. We also present a hierarchical modeling system to allow more complex animations to be created and a dynamic module system so that students may write their own code to handle various tasks within the framework GraphicMentor provides.

1 Introduction

Computer graphics poses a unique challenge to educators in that they must teach students to create images from mathematics and code. Key frame animation at the highest level is a relatively simple concept to grasp, important or ”key” frames are specified by the animator and then the computer generates additional frames to make a smooth animation. There are however several difficult concepts for students to understand in key frame animation. These concepts are how to specify the key frames quickly and how the computer generates the additional frames to complete the animation. When students are beginning to learn computer graphics they do not have the experience base to know when a scene or animation is correct. Further, writing the code to generate sufficient examples to cover the myriad of scenes students can create is far to time consuming for educators. A pedagogical tool that would allow students to visualize how the different parts of computer graphics fit together would be very useful to educators.

GraphicMentor is a pedagogical tool that was designed to allow students to visualize fundamental concepts in computer graphics. The first concept GraphicMentor sought to allow students to visualize was the interaction of objects, light sources, and a camera in a scene. Being able to create and change these three components in real time allows students to develop the background knowledge needed to know when a scene is correct. This visualization system also allows educators to quickly build example scenes for their students. The visualization is handled by allowing the user to use a GUI to change parameters associated with objects, lights, and the camera in real time. GraphicMentor then executes the appropriate OpenGL code to display the scene in real time. The second concept GraphicMentor seeks to visualize is the difference between global and local illumination models. This portion of the visualization is accomplished through the ability to render scenes using ray tracing of photon mapping in addition to OpenGL’s local illumination model. Once students understand the basics of building a scene they can move on to using GraphicMentor to visualize the key frame animation process.

Animation while a long time part of GraphicMentor is the most recent concept that GraphicMentor seeks to help students visualize. In the past GraphicMentor was capable of animation, but did not seek to visualize it. This was in part because the math necessary for animation can be difficult for students new to computer graphics to understand. However, recent research in educational techniques [8, 9] has show students to be quite capable of learning those mathematical topics through visualization. Therefore, recent improvements to GraphicMentor have sought to illustrate the various components of animation so that it may be taught in introductory computer graphics courses.

The rest of the paper is organized in the following fashion. Section 2 discusses previous work done on GraphicMentor. Followed in Section 3 by a discussion of GraphicMentor’s functionality. Section 4 is a discussion into the improvements made to GraphicMentor to allow visualization of key frame animation. Section 5 introduces hierarchical modeling a technique used to make key frame animation more efficient. Dynamic modules allow students to replace sections of GraphicMentor with their own code and are discussed in

Section 6. Finally, Section 7 provides a conclusion to this paper.

2 Previous Work

The development of GraphicMentor began several years ago [1, 2]. The basis of the system was the ability to build OpenGL scenes without writing code. By removing the responsibility of coding from the user it meant they could visualize what was happening in the scene without being required to already have an in depth knowledge of how to use OpenGL. Additionally students can use GraphicMentor as a way to check if the results from code that they write are correct. The first version of GraphicMentor did a good job of visualizing the interaction of lights, objects, and the camera. However, while it included key frame animation it did not offer students much insight into the mechanics behind GraphicMentor's animation system.

GraphicMentor can also be used to demonstrate more subtle topics in computer graphics. When and why do Mach Bands appear? What are the practical differences between local and global illumination models? Students now have a way to try different illumination models so that they can compare these different techniques and answer the questions just posed. Teaching and Learning Computer Graphics Made Easy with Graphics Mentor discusses in detail more of these applications of GraphicMentor along with an expanded discussion of its core features. Students now have an environment in which they can visualize the concepts with which they are having difficulties understanding as presented in their text book or course materials.

The other side of this problem is for educators teaching computer graphics. Writing a program in OpenGL or another graphics API can be very time consuming. GraphicMentor seeks to provide a tool with which quick examples can be generated in order to demonstrate various concepts to students. In as little as a few minutes a rudimentary animation can be assembled to demonstrate the interactions between the camera, lights, and objects. Likewise the effect of tessellation can be demonstrated, an image can be ray traced with POV-RAY [6], or many other effects that can be created with the various options GraphicMentor provides.

Ray tracing while good has some problems caused primarily by the fact that rays are shot from the camera and not from light sources. That means that effects like color bleeding will not be displayed correctly. These short coming were discussed as they pertain to GraphicMentor in Teaching and Learning Computer Graphics Made Easy With GraphicMentor. One technique to overcome these short comings is photon mapping a process in which photons are shot from each light source in a scene. These photons are then traced reflecting when they hit an object until they run out of energy and end. By shifting the focus of the tracing from the camera to the light sources, many shortcomings of ray tracing can be over come at a lower cost than other techniques such as radiosity. Recently Yu et al. have created and presented a photon mapping system in Photon Mapping Made Easy. GraphicMentor can now render a scene using this system. For more detailed information on the photon mapping technology readers are referred to Photon Mapping Made Easy.

The primary focuses of this paper are the recent improvements to the animation system, the addition of hierarchical modeling, and dynamic modules to GraphicMentor. Hierarchical modeling allows users to group objects together. These groupings can be simple like a planet and its rings or it can be more complicated, a person could be modeled with many groups representing their limbs and body parts nested within each other for instance. Of particular interest are the uses of hierarchical modeling with animation, this is the primary application where hierarchical modeling is useful. Dynamic modules are essentially plugins for GraphicMentor except instead of loading something extending GraphicMentor's functionality it replaces part of the functionality. There are still interfaces that have to be adhered to, but aside from that the user is free to fill in the functions as they see fit.

3 Functionality of GraphicMentor

The design of GraphicMentor has been an evolutionary process including some major reengineering recently. GraphicMentor was not originally designed using an object oriented paradigm. Migrating to the object oriented paradigm was necessary in order to facilitate the dynamic module system discussed later on in this paper. The OpenGL Utility Tool Kit (GLUT) while a platform independent way of creating GUIs for OpenGL applications is not by nature object oriented. Therefore, when the GUI for GraphicMentor was redesigned we selected GLOW [5] to use to create the GUI. GLOW is an object oriented wrapper for GLUT that is compatible with any platform that supports OpenGL and GLUT.

A camera is the most fundamental component of every scene. Misunderstanding how a camera is specified can be a very frustrating experience for students new to computer graphics when confronted with a black screen instead of their scene. GraphicMentor seeks to alleviate this problem by drawing a representation of the camera in the environment view windows used for visualizing the camera and light sources in relation to the objects. This representation of the camera illustrates the position, up vector, view volume, and look at point of the camera. As students adjust the parameters that define the view volume they can see how their scene is impacted. Students can also see how moving and rotating the camera changes their scene. This clearly illustrates the strength of education through visualization as the user experiments they develop an intuition for what different concepts in computer graphics like the camera do independent of any API or platform.

By default GraphicMentor places a single light in a new scene with common settings i.e. white light coming from directly above. The user can adjust the lights color, position, and whether or not it is a spot light. While visualizing how lights interact with a scene is important there is a more complex topic to tackle with lighting, illumination models. GraphicMentor seeks to give the user a feel for the difference between local and global illumination models by allowing user to render their scene with ray tracing or photon mapping in addition to OpenGL's lighting model which GraphicMentor uses to render the scene. The ray tracing is done via POV-RAY and the photon mapping with Yu's [3] system. Unfortunately it is too computationally intensive to expect systems to ray trace or photon map a scene in real time. This limits GraphicMentor to visualizing the difference between illumination models. While the user can see how different settings for the camera, lights, and objects impact their scene it becomes quite tedious to render many scenes in succession using a global illumination model.

The Object system of GraphicMentor allows users to add objects to their scene and choose one of six basic shapes for the object to take. GraphicMentor allows the user to experiment with the RGBA values for the material properties ambient, diffuse, and specular in real time. Transformations are also handled in real time allowing the user to translate, rotate, and scale their object. More subtle concepts such as the difference between smooth and flat shading can also be explored once an object is specified; adjusting the tessellation of an object is also possible.

Scenes can be animated with GraphicMentor giving the users a way to play with a different set of object and camera specifications for each scene instead of just a single configuration. During the re-engineering process user testing was very important because it gave some feed back as to whether or not the features were intuitive to use. It quickly became clear that users had very little idea of how GraphicMentor created an animation. Users saved a series of still frames, and could iterate forwards and backwards through them. When a user pressed play from a small number of frames they specified a complete animation was created, but there was no indication of how it was created. As mentioned in the introduction this was on of the key issues we sought to address with GraphicMentor.

4 Improvements to the Animation System

Young children are often taught to take a pad, draw a slightly different picture on each page, and then flip through them quickly. This creates a very simple animation and is the intuition behind key frame animation. Key frame animation is the type of animation which GraphicMentor utilizes. Each frame that a user specifies is a key frame once a user has specified all of the key frames some process is applied to generate a sufficient number of additional frames (at least twenty frames per second is advisable) between each key frame to create a smooth animation; this process is called tweening. GraphicMentor performs tweening through B-spline interpolation.

A B-spline curve is composed of a set of n+1 control points which we will designate $P_0, ..., P_i, ..., P_n$ and a vector of m+1 knots $U = u_0, ..., u_i, ..., u_n$. The curve itself is defined by the function:

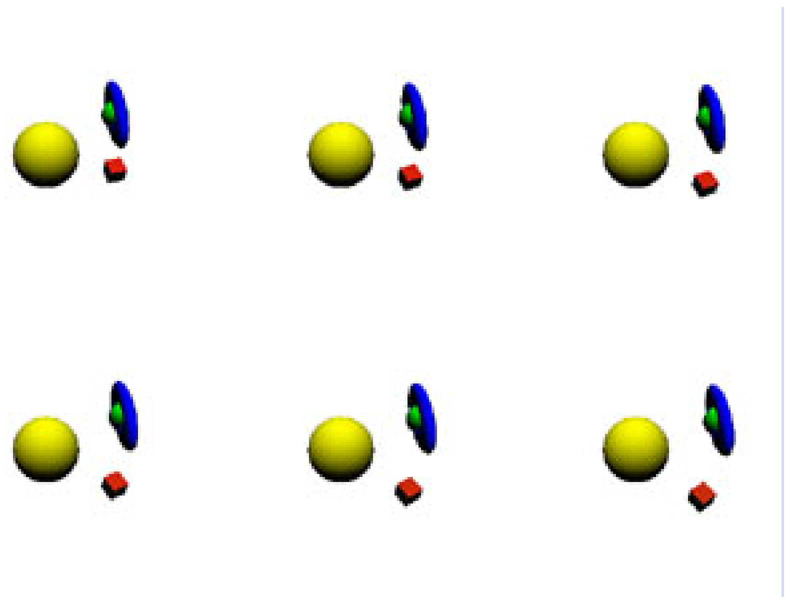

Figure 1: The Film Strip of a Solar System Animation

$C(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i.$

$$N_{i,0}(u) = \begin{cases} 1 & \text{if} \quad u_i <= u < u_{i+1} \\ 0 & \quad otherwise \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{(u_{i+p+1} - u)}{(u_{i+p+1} - u_{i+1})} N_{i+1,p-1}(u)$$

Where $N_{i,p}$ is the B-spline basis function. For a more in depth discussion of B-spline curves readers are directed to Michigan Technological University's Computing with Geometry Webpage. The problem of visualizing curves and surfaces has been addressed in detail in DesignMentor [8]. As such it is not the goal of GraphicMentor to teach students what B-spline curves are. Rather, the goal is to teach students how interpolation, B-spline curves, and key frames can be combined to create an animation.

Teaching key frame animation to students is an important step in learning the fundamentals of computer graphics. Building a static scene is often not enough the real world is dynamic and constantly changing. Therefore, it only makes sense that computer graphics either be interactive, or an animation to illustrate some event from the real world. GraphicMentor allows a user to save key frames which include the object, lights, and camera of a scene and all of their associated properties mentioned above. The user can step forwards or backwards through the frames they have specified and of course play the animation. This functionality lets users create animations, but not understand them. To understand key frame animation students must overcome the obstacle of tweening (interpolation and B-spline curves in the case of GraphicMentor).

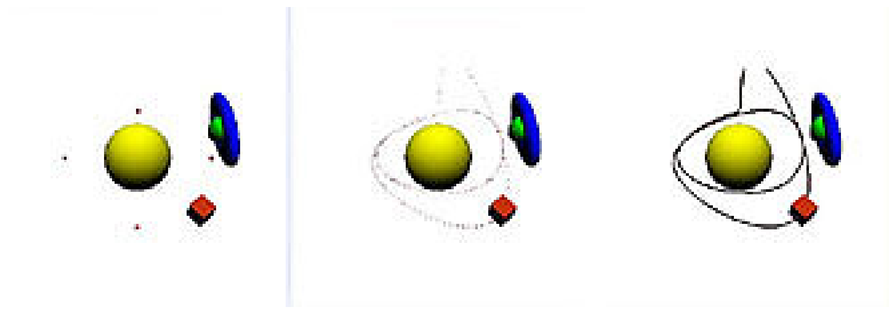

Figure 2: Parts a, b, and c. From left to right the key frames for the block planet, the interpolated frames for the block planet, and the interpolated path for the block planet.

The functionality of GraphicMentor's animation system has been extended with a new feature called Film Strip Mode to help users visualize key frame animation. When Film Strip Mode is activated as you can see in Figure 1 the user now views six frames of the animation concurrently. The animation used in figure one is that of two planets orbiting around a sun; each frame is slightly farther ahead in time than its predecessor. This allows users to understand that the animation is nothing more than a large series of still frames being shown in rapid succession, like the pad of paper mentioned above. To visualize the process of tweening Film Strip Mode allows the user to visualize either the key frames, interpolated frames, or interpolated path for any subset of the objects in the scene. Figure 2 illustrates this using the same solar system animation as in Figure 1, but this time display the key frames, interpolated frames, and interpolated path respectively in parts a, b, and c. Displaying the different types of frames is meant to give students an intuitive idea of how interpolation is used in key frame animation. Being able to visualize all the parts of key frame animation should make the concepts easier to understand allowing students to learn this technique earlier in their education.

5 Hierarchical Modeling

As users begin to understand key frame animation and begin building more complex animations they will quickly realize that moving and changing large numbers of objects for each key frame is quite tedious. Students need to understand that key frame animation is a viable technique for creating animations. If students cannot understand when a technique will be used they have no incentive to learn that particular technique. Hierarchical modeling allows users to define groups that are composed of other groups and objects. A group contained within another group is said to be nested inside of it. Hierarchical models address the problem of making key frame animation a practical tool.

With hierarchical modeling the user can apply transformations to everything contained within a group as a single unit. Figure 3 shows a simple robot built in GraphicMentor;

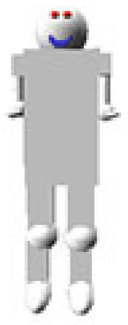

Figure 3: A hierarchical model of a robot.

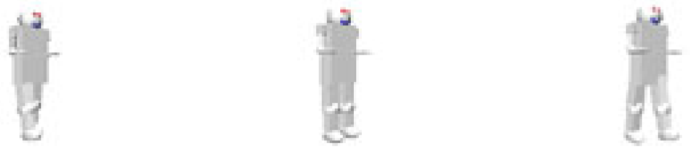

Figure 4: Three key frames from an animation of a walking robot.

the robots head, legs, and arms are all groups, these groups are then nested in other groups to compose a torso group and finally the robot group. Figure 4 shows three sample frames of an animation using many objects in which the robot goose steps across the screen. Instead of moving each individual part of the robot the user simply rotates the leg groups and translates the robot group in each scene. The ability to apply transformations to groups of objects greatly simplifies the task of animating a scene giving students a way to visualize how key frame animation can be used to create more significant animations than a ball rolling down the hill or some other graphical equivalent of Hello World.

Hierarchical modeling can also be of value to educators. Once a user is familiar with hierarchical modeling it becomes much easier to quickly create animations. These animations can be saved and later used as examples by students. Previously educators would have either had to use GraphicMentor to create the animation transforming each individual object or written the corresponding code which would be an even more time consuming process. Hierarchical modeling is one of the aspects of GraphicMentor that clearly aids both students and educators directly.

6 Dynamic Modules

The intention of dynamic modules is to allow GraphicMentor to become a tool for teaching students coding as well as theoretical concepts. One of the driving forces behind re-engineering GraphicMentor in an object oriented paradigm was to facilitate dynamic modules. Each major system in GraphicMentor is implemented as an object. Ideally, each of these objects will be implemented via a dynamic module; either a dynamic link library for windows or a shared object library for UNIX based systems. Students who wish to experiment with code for the various modules need simply replace the associated library file with one of their own. This feature also lends itself very well to educators who wish to assign students coding related to a particular aspect of computer graphics without all the overhead of either the educator or student having to write a program to facilitate displaying the scene. Having a preexisting framework also greatly reduces the possibility that there are bugs in code unrelated to the assigned work.

The first system to be implemented as a dynamic module was the interpolation functions for animation. The interface for this first module was kept as simple as possible. When GraphicMentor generates an animation it calls three functions animationInit, animationWork, and animationClean. By limiting GraphicMentor's calls to these three functions the user is left to write the necessary code with as little restriction as possible on their implementation. Further, when the functions are called GraphicMentor passes in pointers to the data points that were gathered from the user specified key frames, knot vector, and parameters for the B-spline. This allows users to either deal with just interpolating the control points to define the B-spline or they can generate their own knot vector and parameters replacing the ones that GraphicMentor generated and passed in. A broad specification of the dynamic module for animation will allow educators to utilize this feature to create an assignment that matches the needs of their computer graphics curriculum.

Dynamic modules fills an important gap in GraphicMentor's capabilities as a pedagogical tool. Helping students to visualize the fundamentals of computer graphics is very important, but it doesn't change the fact that they still need to learn how to write code utilizing these concepts. Animation is a good first step for this feature because the interpolation code is independent of the graphics API. Being both API independent and dealing with a topic that is generally more in depth than most introductory computer graphics classes cover will make it very useful in expanding or enriching the topics taught during introductory graphics classes. As feedback is generated from the use of dynamic modules future API dependant models can better be shaped to match the needs of the courses they are being used in.

7 Conclusions

GraphicMentor is a pedagogical tool designed to help overcome the inherent difficulties in teaching students computer graphics. The greatest of these difficulties is that computer graphics is based around describing visual elements with code and mathematics. It is a much more natural process for students to visualize the fundamental elements of computer graphics. Further, GraphicMentor provides a way for educators to quickly develop examples of common computer graphics problems such as mach banding or the differences between local and global illumination models.

The contribution of this paper is to introduce GraphicMentor's functionality for visualizing the topic of key frame animation. Traditionally the interpolation needed for tweening would make key frame animation too complicated to be introduced in introductory computer graphics classes. Film Strip Mode provides not only a way for students to visualize the animation as a series of frames, but also visualizations of the interpolation process. Once a student understands the concepts behind a process; the process itself becomes much easier to understand.

Hierarchical modeling provided the means to both create meaningful examples such as the walking robot and a motivation to learn key frame animation. Being able to transform multiple objects concurrently is a necessity for key frame animation to be a practical technique. When students see that key frame animation is both practical and simple they will be motivated to learn and understand it giving them a good entry point into the fundamentals of animation.

Finally, dynamic modules provide a way for GraphicMentor to be used to test students understanding of the concepts being taught. Assigning students to code up individual modules such as the animation code affords students the chance to write a relatively small portion of code and test it without worrying about bugs in the rest of the code necessary to display the module they have written. Text books, lecture, and visualization are good ways to teach students, but there is no assurance that they have learned the material until they are forced to use it for themselves. GraphicMentor will allow students to better understand the basics of computer graphics and learn more complex topics such as key frame animation earlier on in their education through visualization.

References

[1] Dejan Nikolic and Ching-Kuang Shene, *"GraphicsMentor: A Tool for Learning Graphics Fundamentals, "*ACM 33rd Annual SIGCSE Technical Symposium, Febuary 2002, pp. 242-246.

[2] Ching-Kuang Shene, *"Teaching and Learning Computer Graphics Made Easy with GraphicsMentor,"* Interactive Multimedia Electronic Journal of Computer-Enhanced Learning, October 2002.

[3] Tin-Tin Yu, John Lowther and Ching-Kuang Shene, *"Photon Mapping Made Easy,"*, ACM 36th Annual SIGCSE Technical Symposium, Febuary 2005.

[4] Mark Kilgard, *"GLUT - The OpenGL Utility Toolkit,"* http://www.opengl.org/resources/libraries/glut.html.

[5] Daniel Azuma, *"The GLOW Toolkit, "* http://glow.sourceforge.net/, October 2000.

[6] *"POV-Ray - The Persistence of Vision Raytracer,"* http://www.povray.org/, 2005.

[7] Ching-Kuang Shene, *"HMichigan Technological University Computing with Geometry Webpage,"* http://www.csl.mtu.edu/cs3621/www/Home.html, Fall 2004.

[8] John L. Lowther and Ching-Kuang Shene *"DesignMentor: An Interactive Environment for Learning the Fundamentals of Curve and Surface Design,"* ASEE 54th Annual Engineering Design Graphics Mid-Year Conference, November 1999, pp. 193-199.

[9] John Lowther and Ching-Kuang Shene, *Teaching B-splines is Not Difficult!,"* ACM 34th Annual SIGCSE Technical Symposium, Febuary 2003, pp. 381-385.