GraphicsMentor: A Tool for Learning Graphics Fundamentals[*]

Dejan Nikolic and Ching-Kuang Shene[†]
Department of Computer Science
Michigan Technological University
Houghton, MI 49931–1295
Email: {dnikolic,shene}@mtu.edu

Abstract

This paper discusses the functionality of GraphicsMentor. GraphicsMentor permits a student to modify many parameters of the camera, objects, and light sources interactively, and to visualize the rendered result on-the-fly. It also supports basic keyframe animation. With GraphicsMentor, a student should be able to grasp the fundamentals of computer graphics quickly and easily in a learning-by-doing way.

1 Motivation

Hoare once said, "You can't teach beginning programmers top-down design because they don't know which way is up." Similarly, we cannot teach graphics programming to beginners because they do not know if their programs function properly before knowing what the anticipated effect would be. Thus, "I do and I understand" (Lao-Tzu) is not enough for learning graphics, and we should add "I see and I remember" for a beginner to recognize what is correct and what is not. Our experience shows that many students have difficulties in camera setup and the use of light sources. Frequently, a blank image is generated because the camera points to a wrong direction, or the scene is not illuminated properly because of incorrect positions of light sources. To address these and other problems, we developed GraphicsMentor, a pedagogical tool, written in C/C++, OpenGL and GLUT, that runs on Linux, SGI, Sun Solaris, and Windows. It allows a student to alter almost all parameters of the camera, light source and object material property, and visualize the result on-the-fly. A limited keyframe animation capability is also available. Because of its flexibility and interactive nature, GraphicsMentor can help students learn most graphics fundamentals easily. Hence, GraphicsMentor can be used in a computer-equipped classroom or in a lab for students to practice and experiment various concepts. This paper briefly reviews GraphicsMentor and presents its possible classroom uses. The interested readers may refer to [1] for a detailed discussion of many related work.

[*]This work was supported by the National Science Foundation under grants DUE-9653244, DUE-9952621 and DUE-0127401.

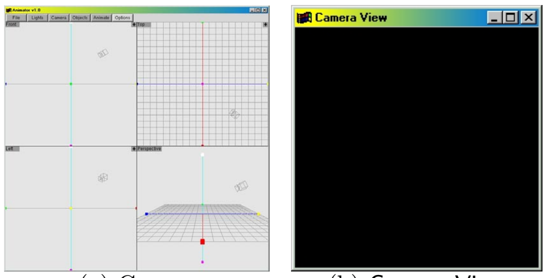
[†]Corresponding author.

2 The Environment

When GraphicsMentor starts, it displays the main canvas window and the Camera View window. The former is divided into four subwindows. The upper-left, upper-right, lower-left and lower-right subwindows display the front view, top view, left view, and world view of the scene (Figure 1(a)). The first three can be switched to the back, bottom, and right views by clicking on the button in the upper-left corner of each subwindow. Initially, the scene only has a camera and a light source as shown by the camera icon and a white dot in the subwindows. The Camera View window displays what the camera sees. Since nothing is in the scene, it shows the background color (Figure 1(b)). The main canvas also has a number of menu buttons for activating the camera, lights, objects and animation. With this setup, a student has no difficulty in placing objects into the scene, manipulating the position of the camera, and changing the position, color, and type of the light sources.

3 Adding Objects - A Simple Way

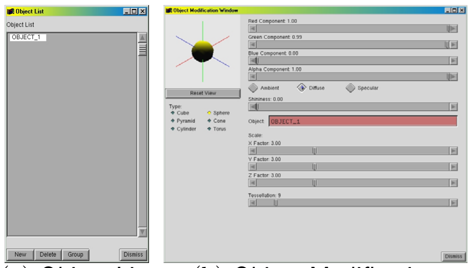
The first thing for a student to do is adding objects to the scene. GraphicsMentor supports six primitives (*i.e.*, cube, pyramid, cylinder, cone, sphere and torus), although more complicated ones such as B-spline and NURBS surfaces will be added in the future. To add an object, click on the `Objects` menu button to bring up the Object List window (Figure 2(a)). The display



(a) Canvas (b) Camera View

Figure 1: GraphicsMentor's Basic Environment

area of the Object List window shows the default name of each object (*e.g.*, `OBJECT_1`). Then, click on the `New` and `Delete` buttons for adding and removing an object. Clicking on the `New` button brings up the Object Modification window as shown in Figure 2(b). This window provides six primitives. Click on one of them to display it in the upper-left display area and on the canvas. Since these primitives have their default sizes, a student may use the scaling slides to scale the object in the x-, y-, and z- directions.



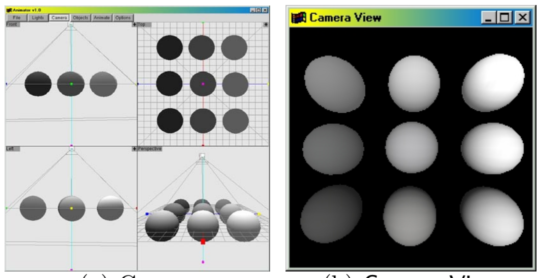
Red Component: 1.00
Green Component: 0.99
Blue Component: 0.00
Alpha Component: 1.00
Shininess: 0.00
Object: OBJECT_1
Scale:
X Factor: 3.00
Y Factor: 3.00
Z Factor: 3.00
Tessellation: 9

(a) Object List (b) Object Modification

Figure 2: Object Modeling

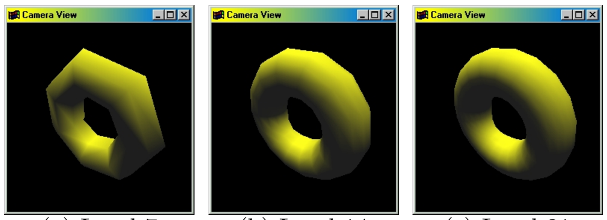
The default color of an object is mid gray. The top four slides are used to change the RGB color components and alpha channel, and this change can be applied to the ambient, diffuse, and specular components of the object's material property. The fifth slide is used for selecting the shininess of the object. Moreover, a student can left-drag (*resp.*, right-drag) to perform translation (*resp.*, trackball type rotation). With these capabilities, a student can quickly grasp the impact of different ambient and diffuse values on rendered shapes. In fact, it is easy to place a number of spheres in the scene, each of which has a different combination of ambient and diffuse values, and observe the differences among the rendered results. Figure 3 is such an example. The spheres on the first, second and third rows have ambient values 0.6, 0.4 and 0.2, and the spheres on the first, second and third columns have diffuse values 0.2, 0.5 and 0.8.

Objects are triangulated, and the default triangulation may be too coarse for a particular application. A student may use the tessellation slide to set the level of tessellation. Figure 4 shows the impact of different tessellation level on the rendered result. Although the shapes are rendered with smooth (*i.e.*, Gouraud) shading, the silhouettes of the shapes with lower level of triangulation are still polyhedral as shown in the figures.



(a) Canvas (b) Camera View

Figure 3: Ambient and Diffuse Experiment



(a) Level 7 (b) Level 14 (c) Level 21

Figure 4: Tessellation Levels

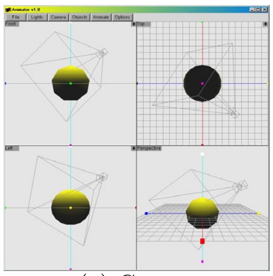
Thus, students can quickly and easily learn the meaning and impact of each component of an object's material property, the use of geometric transformations (*i.e.*, scaling with slides and translations and rotations with mouse buttons), and the impact of tessellation level. As a result, they will be able to write programs and recognize problems with confidence.

4 The Camera

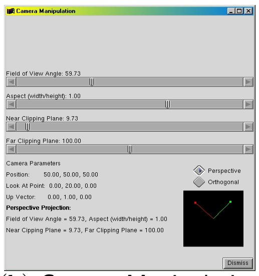
The camera is the most important item because no image can be produced without a camera. Click on the `Camera` menu button to display all characteristics of the camera on the canvas (Figure 5(a)), and brings up the Camera Manipulation window (Figure 5(b)).

GraphicsMentor permits a student to modify view angle, aspect ratio, near and far clipping planes, look-at point, and camera orientation. Moreover, a student can choose the perspective or orthogonal view. Left-click on one of the side-view subwindows to place the camera at that position, while left-drag to move the camera. Right-click also changes the camera's default look-at point.

All four subwindows on the canvas show the camera and the view volume. A student can use the first slide of the Camera Manipulation window to change the view



(a) Canvas



Camera Manipulation

Field of View Angle: 59.73

Aspect (width/height): 1.00

Near Clipping Plane: 9.73

Far Clipping Plane: 100.00

Camera Parameters

Position: 50.00, 50.00, 50.00

Look At Point: 0.00, 20.00, 0.00

Up Vector: 0.00, 1.00, 0.00

Perspective Projection:

Field of View Angle = 59.73, Aspect (width/height) = 1.00

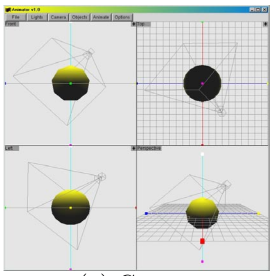
Near Cipping Plane = 9.73, Far Clipping Plane = 100.00

(b) Camera Manipulation

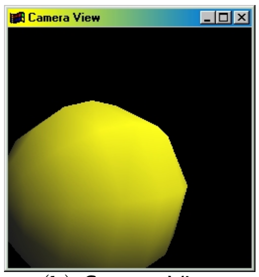
Figure 5: Manipulating the Camera

angle, which mimics zooming. The second slide is for changing the aspect ratio of the view volume; however, since the Camera View window has an aspect ratio of 1:1, altering the aspect ratio changes the aspect ratio of the view volume only and distorted objects may occur in the Camera View window. To change camera orientation, a student can left-drag the two perpendicular (*i.e.*, the right and up) vectors in the small window in the lower-right corner of the Camera Manipulation window.

The last two slides are for changing the positions of the near and far clipping planes. GraphicsMentor does not allow these two planes to cross each other. Figure 6(a) shows a view volume with the near clipping plane close to the camera. The far clipping plane cuts part of the sphere off; however, this cannot be seen because the front side is not transparent. If we move the near clipping plane towards the sphere as shown in Figure 7, the Camera View window shows a clipped sphere. Through the front hole, one can see that the back-end is also clipped. All viewing parameters are shown in the lower half of the Camera Manipulation window (Figure 5(b)).



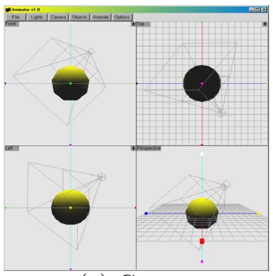
(a) Canvas



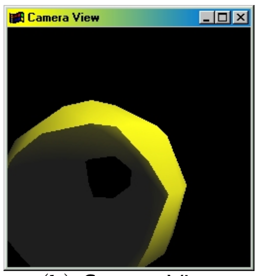
(b) Camera View

Figure 6: Before Modifying the Near Clipping Plane

With these detailed information and interactive capability, a student is able to learn all viewing related knowledge without writing any program. In our experience, placing the camera at a position with proper orientation, and the correct use of the look-at point, view angle, and clipping planes could be very challenging to beginners, especially to those who do not have extensive experience in playing with cameras and camcorders. As we all know, many students generate blank screens! Fortunately, GraphicsMentor virtually eliminates this problem by providing students with an environment to practice and experience the effect of different parameters.



(a) Canvas



(b) Camera View

Figure 7: After Modifying the Near Clipping Plane

5 Lights, Please!

Light sources are as important as the camera, because they illuminate the scene. GraphicsMentor supports maximum eight lights, and when it starts, GraphicsMentor includes only one active light. To create, activate, deactivate, or modify a light, click on the `Lights` menu button to bring up the Light List window in which all created lights are shown along with their default names (Figure 8). A student can click on the `New` button to create a new light. Once a light is listed, click `Remove` to deactivate, `Active` to activate, and `Modify` to modify.

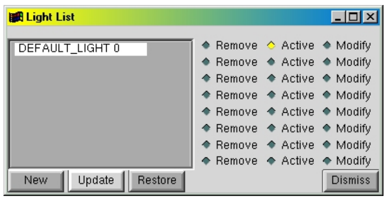
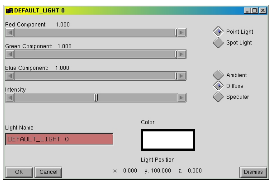


Figure 8: The Light List Window

GraphicsMentor supports two types of light, point light (default) and spot light. A point light is represented by a dot in all four subwindows and shown in the color of the light. Except for the default light 0, whose initial position is above the coordinate origin, all other lights have their initial positions at the origin. Click on the `Modify` button to bring up a light window (Figure 9(a)). The bottom part of this window shows the name, color and coordinates (*i.e.*, position) of the light being modified. The upper-right corner buttons permit a student to switch between point light and spot light instantly. The top three slides control the RGB components of the light color, and the three buttons to the right of the slides allow a student to choose one of the ambient, diffuse, and specular components to modify.

Once a light is selected, a student can left-drag it in any



DEFAULT_LIGHT 0

Red Component: 1.000

Green Component: 1.000

Blue Component: 1.000

Intensity

[✓ Point Light] Spot Light

Ambient [✓ Diffuse] Specular

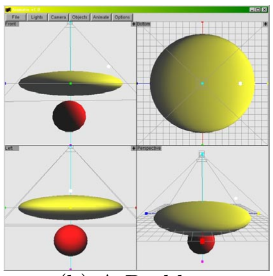
Light Name: DEFAULT_LIGHT 0

Color:

Light Position x: 0.000 y: 100.000 z: 0.000

OK Cancel Dismiss

(a) Point Light



(b) A Problem

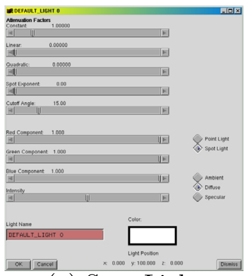
Figure 9: Point Light

side-view subwindow. Thus, combined with the camera and material property of each object, a student can experiment and verify many important concepts and effects of an illumination/lighting model. A must-show example is two of the most important drawbacks of local illumination models: no shadows and non-blocking light. Figure 9(b) has a big yellow oblate ellipsoid between the light source and the red sphere. It is clear that the light cannot illuminate the sphere in reality; however, the image clearly shows that part of the red sphere facing the light is illuminated and the oblate ellipsoid casts no shadow on the red sphere!

GraphicsMentor also supports spot light. Clicking on the `Modify` button of a light brings up its light window. Then, click on the `Spot Light` button to change the light to a spot light (Figure 10(a)). In fact, the spot light window is identical to the point light version, except that there are five more slides for modifying the characteristics of a spot light. The spot light icon is a dot with a cone attached shown in the color of the light. A student can left-drag to change the position, and right-drag to modify the direction of a spot light. The top three slides are for modifying the attenuation factors k_c, k_l and k_q, which are defined as follows:

$$\text{attenuation factor} = \frac{1}{k_c + k_l \cdot d + k_q \cdot d^2}$$

The `Spot Exponent` slide controls the concentration of the spot light, and the `Cutoff Angle` slide controls the spot light cone angle.



DEFAULT_LIGHT 0

Attenuation Factors

Constant: 1.00000

Linear: 0.00000

Quadratic: 0.00000

Spot Exponent: 0.00

Cutoff Angle: 15.00

Red Component: 1.000

Green Component: 1.000

Blue Component: 1.000

Intensity

Point Light [✓ Spot Light]

Ambient [✓ Diffuse] Specular

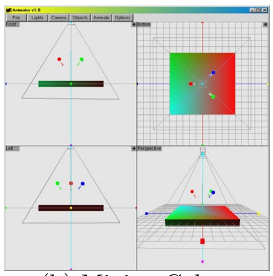
Light Name: DEFAULT_LIGHT 0

Color:

Light Position x: 0.000 y: 100.000 z: 0.000

OK Cancel Dismiss

(a) Spot Light

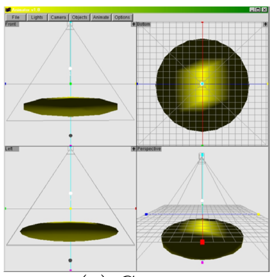


(b) Mixing Colors

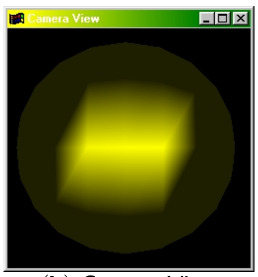
Figure 10: Spot Light

An interesting experiment for students to do is to visualize the mix of colored lights (Figure 10(b)). A thin white box is placed in the scene along with three spot lights with colors red, green and blue, each of which has a small cutoff angle so that it can only illuminate the vicinity of the coordinate origin. The camera is placed above the origin. A student can choose different position, color, cutoff angle, spot exponent of a spot light to generate different effect. In this way, an instructor may be able to cover some color models easily.

GraphicsMentor is also useful in helping students visualize other shortcomings of local illumination models. An oblate ellipsoid with coarse triangulation (for demonstrating the relationship between coarse triangulation and spot light) is placed in the scene (Figure 11(a)). There are two lights. The top one is a spot light of white color, placed close enough to the surface so that its light cone only covers a portion of the surface. The bottom one is a mid gray point light for illuminating the bottom part of the surface. The camera is placed above the surface. Because the triangulation is coarse, the spot light does not shed a circular shape on the surface. Instead, the shape is a polygon whose shape depends on the way of triangulating the surface. See the top view and world view subwindows. The silhouette of the surface in the Camera View window is barely seen, and the spot light generates a hexagonal illuminated area on the surface. Mach band is clearly seen!



(a) Canvas



(b) Camera View

Figure 11: The Impact of Tessellation in Rendering

6 Basic Animation

GraphicsMentor supports basic keyframe animation. Click on the `Animation` menu button to bring up the Animation Control window (Figure 12). Then, the characteristics of the camera, light sources, and objects may be altered. Once this is done, click on the `Record` button in the Animation Control window to record a *keyframe*. Repeat this process until all keyframes are recorded. GraphicsMentor will take these keyframes, calculate the frame rate, and interpolate necessary frames between two adjacent keyframes. Then, click `Play` to play, `Stop` to stop, and `Record` to append more keyframes.

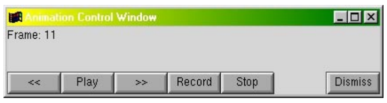


Figure 12: The Animation Control Window

Figure 13 shows a simple example. Three spheres, in yellow, red, and blue, rotate about the origin with camera pointing downward. Eight keyframes are recorded as shown. The yellow sphere rotates about 130°, the red one rotates about 180° and is a little faster, and the blue one rotates about 360° and is the fastest moving object. After all eight keyframes are recorded, GraphicsMentor plays back the animation smoothly.

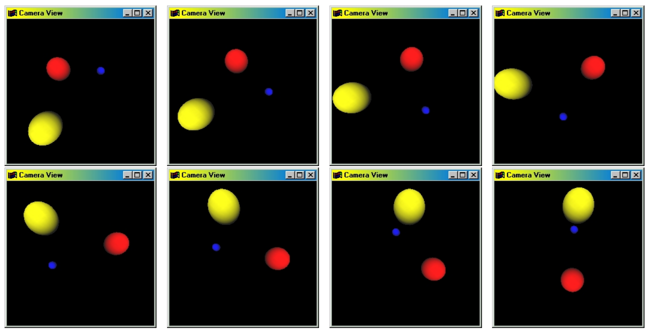


Figure 13: A Simple Animation Sequence

7 Future Work

The goal of this first stage development is to design a usable prototype quickly to help students learn the fundamentals. As a result, many important concepts are not supported fully, and will be added in the future. Here is a to-do list for future development. **First**, the object module should be improved to include more primitives such as superquadrics, B-spline and NURBS. **Second**, the current simple hierarchical modeling capability should be extended to support more sophisticated modeling activities and basic inverse-kinematic animation. **Third**, elementary texture mapping should be made available. **Fourth**, while GraphicsMentor is not capable of global illumination, it can convert the scene to a POV-Ray input for raytracing so that a student can compare the differences between local and global illumination models. We are currently developing a small raytracer that supports radiosity, and hope that eventually raytracing, radiosity, and photon-map will become available in GraphicsMentor along with a stand-alone radiosity system. **Fifth**, in addition to inverse-kinematics, the animation module should be expanded to include non-linear editing so that a student can design multiple animation sequences and join them together into a single stream. **Sixth**, since digital image editing has become a household tool due to the popularity of digital cameras and camcorders, it would be useful for GraphicsMentor to support a few basic postprocessing filters (*e.g.*, motion/Gaussian blur, sharpening, and morphing). **Seventh**, we intend to make GraphicsMentor more modularized so that an instructor can select a component of GraphicsMentor for students to modify, extend, or even re-implement. In this way, students will have a better platform to practice graphics programming.

8 Conclusions

We have presented an pedagogical tool, GraphicsMentor, for teaching and learning the fundamentals of computer graphics. It compensates the raytracing approach that has been used in our Introduction to Computing with Geometry course [2] for helping students who do not have graphics background pickup basic concepts in graphics quickly. GraphicsMentor can certainly be used in an introduction to computer graphics course to illustrate and demonstrate various concepts and algorithms. While there are numerous Java appelets available on the Internet that may serve the same purpose, we believe a stand-alone approach is superior because it can integrate many components into a single, uniform interface and yet perform efficiently. We plan to continually develop GraphicsMentor into a full-blown system in the near future. The interested readers may find more about our work, software availability, and future announcement at the following site:

`http://www.cs.mtu.edu/~shene/NSF-2`

Acknowledgment

We tank the reviewers for their constructive comments.

References

[1] Lowther, J. and Shene, C.-K., Rendering + Modeling + Animation + Postprocessing = Computer Graphics, *The Journal of Computing in Small Colleges*, Vol. 16 (2000), No. 1 (November), pp. 20–28. Reprinted in *Computer Graphics*, Vol. 34 (2000), No. 4 (November), pp. 15–18. Reprinted in *Computer Graphics*, Vol. 24 (2000), No. 4 (November).

[2] Lowther, J. and Shene, C.-K., Computing with Geometry as an Undergraduate Course: A Three-Year Experience, *ACM 32nd SIGCSE Technical Symposium*, February 21–25, 2001, pp. 119–123.