A 3-Dimensional Geometric Modeler Using OpenGL

Technical Report 96-09

David Phillip Cleveland

Department of Computer Science and Engineering

Auburn University

Auburn University, Alabama 36849-5374

August 8, 1996

Project Abstract

A 3-Dimensional Geometric Modeler Using OpenGL

David Phillip Cleveland

Directed by Kai-Hsiung Chang

The design of a Three-Dimensional Geometric Modeling application must be user friendly in addition to having useful features. Many commercial applications push features at the expense of user-friendliness in an effort to sell the product. No matter how powerful the application may be, if the interface is so anti-intuitive that the user is constantly lost, the powerful features are wasted. For many owners of a home pc who would like to use such a modeler for personal use, these feature-packed modelers are too expensive and difficult to use.

This project shows that an intuitive user-friendly interface can be maintained while still supplying the necessary features for a useful Three-Dimensional Geometric Modeler. OpenGL is used in the implementation of the Modeler to supply the powerful modeling features and Windows 95 is used to support the graphical user interface.

CHAPTER I

Introduction

1.1 Introduction

Three dimensional (3D) geometric modeling has proven to be very useful in today's industry. These industries include automobile, chemical, television, movie, multimedia games, and many more where computers are used for visualization of data. In the past, the intense graphics demands and computational complexity of 3D modeling software required it to be run on workstations such as those produced by Silicon Graphics. These workstations, with their powerful high-speed processors and sophisticated graphics capabilities are able to provide the necessary power to perform the modeling and spectacular special effects associated with 3D modeling. Now, with the appearance of faster, more powerful processors and hardware, 3D modeling applications are becoming available for the desktop PC. Unfortunately, many of these applications are so sophisticated a user must seek special training just to understand the interface and perform simple tasks. Furthermore, most of these applications are very expensive, sometimes more than $10,000 dollars. This amount is beyond what most ordinary users of a Home PC are willing to pay. The user certainly isn't going to pay for expensive training. It is likely that the user would never use many of the sophisticated capabilities associated with high-end modelers and really only needs a simple Modeler that is easier to use. For example, a low-end user is unlikely to use inverse kinematics, morphing, or animation. However, even the simplest modelers need to perform certain functions, and these functions require the programmer to have extensive knowledge of mathematical computation and 3D geometry. As a result, even the simplest modeling application is going to be relatively expensive to a PC user. Fortunately,

sophisticated graphics packages are starting to appear for the home PC which have some amazing features. One such graphics package is called OpenGL.

1.2 OpenGL

OpenGL is a powerful graphics library which has been around for some time. It is a standard library which was originally developed for Silicon Graphics workstations in the early 1980s [7]. It has since been ported to other platforms and operating systems, particularly Windows NT and Windows 95 which have nearly monopolized the Home PC market. Best of all, OpenGL for Windows NT and Windows 95 is FREE! OpenGL provides many functions to draw 2D and 3D objects on the screen. It has a rich set of features including lighting, texture mapping, blending, antialiasing, depth cueing, atmospheric effects, animation, and more [5,7]. Its arrival greatly simplifies the creation of 3D applications. A programmer familiar with OpenGL and Windows programming can quickly and inexpensively create a 3D Modeler which can have many impressive features.

1.3 Scope

The focus of this project has been the design and development of a usable Three-Dimensional Modeler while maintaining an intuitive user-friendly interface. Many of the advanced features of the OpenGL graphics package have not been incorporated into the prototype application. The emphasis has been placed on providing an interface that allows the user to perform an action on the 3D model in an efficient and intuitive manner. The features of OpenGL that are realized in this prototype include solid modeling using material properties and lighting.

The design and development of a Three-Dimensional Geometric Modeler complete with texture mapping, blending, warping, antialiasing, atmospheric effects, animation, etc. is beyond the scope of this project. However, this project provides a foundation for such a tool.

1.4 Benefits

The benefits of having a Three-Dimensional Geometric Modeler complete with all the features described above are many. These types of modelers are primarily used in the field of Mechanical Engineering where they are used to assist in designing and modeling engine parts, mechanical gearing, and other mechanical devices. These modeling tools are used in the field of Medicine to model DNA strands, body parts, and even the human brain. Recently, these types of tools have increasingly been used in the Television and Movie industries for their ability to produce realistic animation sequences. These tools are being used in the video and computer gaming industry for producing realistic three-dimensional landscapes and other effects. These modelers can also be used by artists and students for school related work or just for entertainment.

CHAPTER II

Literature Review

2.1 Software Review

There are many 3D modeling programs on the market today, and new ones are becoming available practically every day. Consequently, it would be impossible to purchase and review even a handful of these modelers personally. Fortunately, some of these modelers have demos which allow one to get the feel for it before buying. For those that do not have demos, there are usually review articles which discuss the pros and cons of the modeler. Following are reviews of some of the most popular 3D modeling applications available.

Crystal TOPAS Professional 5.1, by CrystalGraphics is a high-end Modeler and costs $2,495. This application seems to be popular for its high-quality results, not-so-steep learning curve, and excellent features [6]. According to [6], it has excellent control over the model making process, texture mapping, surface modeling, lighting and shadow quality. TOPAS also supports good animation features and control [6].

3D Studio Release 4.0 ($2,995) by Autodesk is the ultimate in 3D modeling software. It has been around for years, and even release 2.0 produces exceptional quality images. Release 2.0 was difficult to use, and according to [6], the learning curve for release 4.0 is still steep. But if a user can overcome this learning curve he will benefit from an amazing set of features. One of the features is the ability to add third party plug-ins which extend or enhance the already abundant list of features [6]. Another impressive feature is inverse kinematics support [6]. This allows the user to give realistic motion control to living models such as humans. All of its features has made 3D Studio very useful in the film and game development industries.

Truespace ($1095) by Caligari is an interesting alternative to the previous two modelers. Unlike the previous two modelers, Truespace uses splines instead of polygons to model curved surfaces. This allows for easier curve drawing and manipulating. Individual points in the curve can be moved, effectively molding the curve as if it were made of clay. Included in the package are over 600 3D objects and more than 300 textures that can be added to an existing model. Truespace uses ray-tracing to render the model. This results in slower render speeds, but the results are exceptional images.

Real 3D 2.4 ($1,295) from RealSoft is another Modeler for which the author was able to download a trial version. According to [6], Real 3D doesn't have a lot of fancy features, but instead focuses on its particle system generation, spline modeling, fractals, and collision detection. Like Truespace, the rendering is based on ray-tracing which is slow, but produces high-quality images. The example images show off some of Real 3D's interesting features, such as blurring. Real 3D has a good animation control and even supports inverse kinematics, which was one of the advanced features on the high-end 3D Studio Modeler [6]. Although [6] claims that Real 3D is not difficult to use, the author found it to have a slightly steep learning curve. There are many buttons that have little images on them. These images are supposed to remind the user of the button's function, but there are too many buttons and the images are not helpful.

Imagine 3.0 ($795) from Impulse is based on 2-D splines and polygons [6]. It has good control over splines and also allows an object to be modified as if a magnet were pulling at it [6]. It has over 100 textures and a particle system for creating snow, rain, dust, etc [6]. It has a Bones feature that is similar to inverse kinematics. With special effects such a morphing and stereo imaging, Imagine is relatively simple Modeler with excellent features that make it interesting and productive [6].

CHAPTER III

Problem Statement

The 3D modelers which are used by professionals are either too expensive or too difficult to learn. The high cost of the modelers is a reflection of the difficulty in developing and implementing all of the sophisticated features into the Modeler. As more and more features are packed in, it becomes increasingly difficult to provide an easy, intuitive interface by which to access and utilize these features.

3.1 Interface Design

With so many Three-Dimensional Geometric Modeling applications appearing on the market, it can be difficult to choose the best one. Consequently, the developers of these products are packing their product with more and more features which inevitably increases its price. As a result, the product with the most features tends to be the one on top. In this type of marketplace where features rule, the graphical user interface takes second place. The major shortcoming found in the products personally reviewed by this author is the inability to move parts of the model by simply dragging it. This was difficult to do because the user only had one window in which to work. The view of the model in this one window must be rotated in order to move an object in three dimensions. Another shortcoming was the lack of an intermediate rendering of the model. The intermediate rendering gives the user a smooth or flat shaded view of the model as they work. Some of the modelers reviewed did not provide this feature, and as a result, it was difficult at times to visualize what the final rendering would look like. These two deficiencies are satisfied in the prototype for this project.

CHAPTER IV

Implementation

4.1 Overview

There are two primary issues considered in the design and implementation of the project prototype. The first issue studied was the inability to drag an object in the commercial modelers. This problem was solved by providing three separate views of the model on the screen at the same time. These views contain a top view, front view, and right side view of the model. The user can click on a shape in any window and drag in the two dimensions characterized by that view. The user can quickly move the cursor to the shape in either of the other two views to move in the third dimension if necessary. The top view shows the XZ plane, the front view shows the XY plane and the right view shows the YZ plane.

The second issue considered was the lack of an intermediate rendering of the model in some of the modelers reviewed. A quick rendering of the model given the current placement of the objects, current camera position and settings, and current lighting can give the user a good idea of what the final rendering will look like. Having to imagine what the final scene will look like can be especially difficult when there are many objects. To solve this problem, a fourth window was added to the main screen which contains a smooth rendered view of the model.

4.2 Primitive Shapes

In order to fully utilize the quick rendering speed of the OpenGL rendering engine, the data structure used to describe a shape was based off of the OpenGL geometric primitive types. These types include points, lines, line strip, line loop, polygon, quads, quad strip, triangles,

triangle strip and triangle fan as shown in Figure 4-1. The data structure also has to provide support for drawing the wireframe of a shape in the top, front and right views. These views will give an orthographic projection of the model. The OpenGL view, or rendered view, will give a perspective projection of the model.

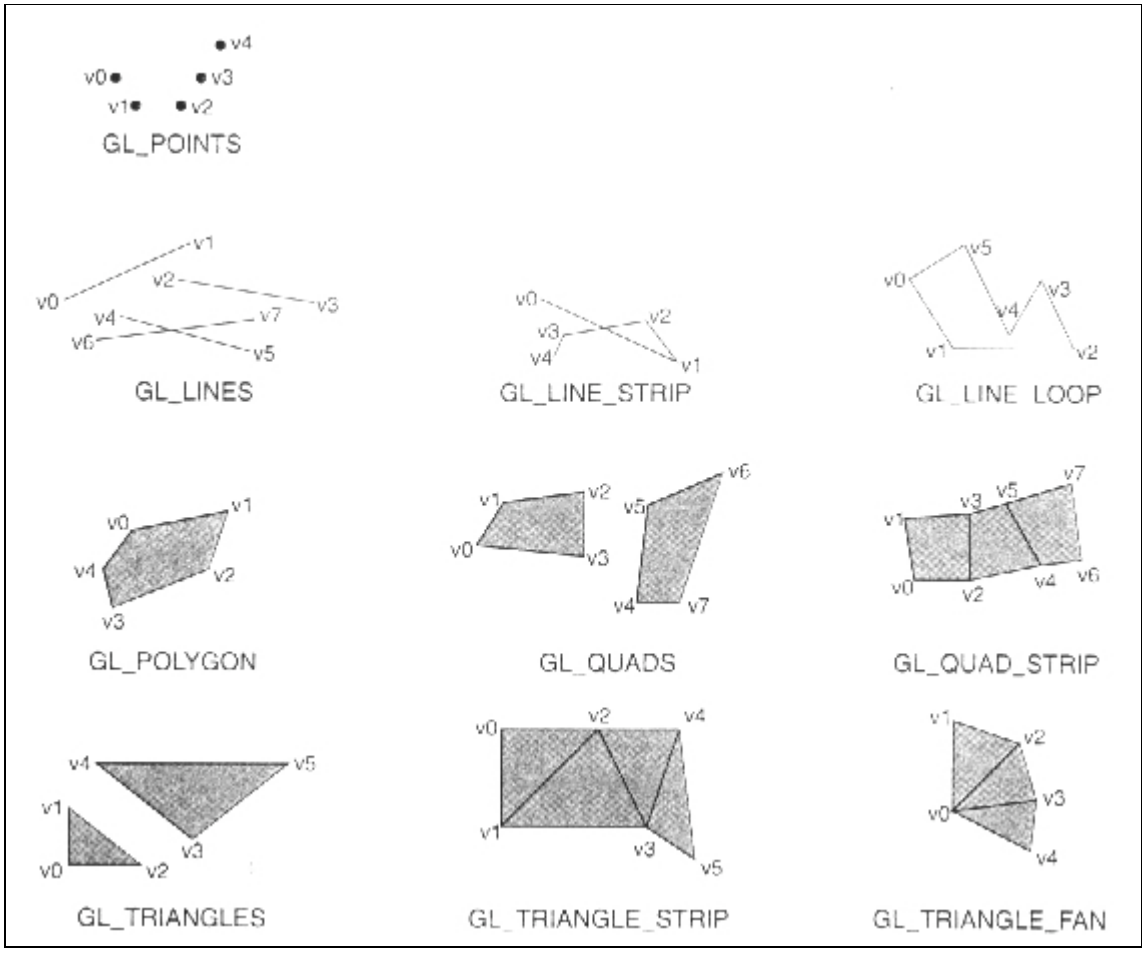

Figure 4-1 OpenGL Primitive Types

In OpenGL, all of the primitive types are described by a list of vertices. OpenGL is made aware of a vertex by a call to the function, `glVertex3f(x,y,z)`. To tell OpenGL what type of

primitive these vertices belong to, they must fall between calls to the `glBegin()` and `glEnd()` functions. For example, to create a polygon the following code is used:

```
glBegin(GL_POLYGON);
      glVertex3f(5.0, -5.0, 0.0);
      glVertex3f(5.0, 5.0, 0.0);
      glVertex3f(-5.0, 5.0, 0.0);
      glVertex3f(-5.0, -5.0, 0.0);
glEnd();
```

Value	Meaning
GL_POINTS	individual points
GL_LINES	pairs of vertices interpreted as individual line segments
GL_POLYGON	boundary of a simple, convex polygon
GL_TRIANGLES	triples of vertices interpreted as triangles
GL_QUADS	quadruples of vertices interpreted as four-sided polygons
GL_LINE_STRIP	series of connected line segments
GL_LINE_LOOP	same as above, with a segment added between last and first vertices
GL_TRIANGLE_STRIP	linked strip of triangles
GL_TRIANGLE_FAN	linked fan of triangles
GL_QUAD_STRIP	linked strip of quadrilaterals

Table 4-1 Geometric Primitive Names and Meanings

Table 4-1 lists the OpenGL names that can be included in a call to `glBegin()`. In addition to the vertex data, OpenGL also needs to know the position of the camera and all lights in order to render the scene. The camera and lights will be described in section 4.3.

At a higher level, the OpenGL primitive types are combined to form the five basic 3D shapes supported by the prototype modeler. These five types include box, cone, cylinder, sphere and torus. A box is described by a length, width and depth. A cone is described by a radius, height and number of slices. A cylinder is described by a base radius, a top radius, a base inner radius, a top inner radius, height and number of slices. A sphere is described by a radius, number of slices, and number of stacks. A torus is described by a ring radius, tub radius, number of slices, and number of stacks. In addition, each shape also has data specifying its ambient, diffuse

and specular color values which OpenGL uses to describe how to color and highlight the object. Finally each shape has a shininess factor which is used by OpenGL along with the specular color value to decide how shiny to make the shape.

4.3 Projection

The rendered view displays a perspective projection of the model while the top, front and right views simultaneously display an orthographic projection of the view. Each orthographic view shows the user what the model looks like from each of the three positive axes. The top view shows the scene as viewed from the positive y axis. The front view shows the scene as viewed from the positive z axis. The right view shows the scene as viewed from the positive x axis. The vertex data is used to create each of the orthographic projections. In order to create the perspective projection in the rendered view, camera and lighting data must be supplied to OpenGL.

The three orthographic views of the model are not created using the OpenGL orthographic projection facilities. Doing so would not allow the user to select and move objects in those three views. Alternatively, these views are created using basic Windows 95 line drawing routines. All shapes are projected except for the Sphere. Since its projection and bounding rectangles are identical in all views, a circle is drawn as its projection. This makes updating the views much more efficient since spheres usually contain many vertices and take much longer to draw than the other shapes.

The OpenGL perspective projection facility and the OpenGL lighting and material facilities are used to create the rendered view. All objects contained in the viewing volume are displayed.

The Camera

The camera is described by a position, field of view angle, near plane, and a far plane. The near plane, far plane and field of view angle are used to create a viewport through which the model is rendered and viewed. The area between the near plane and far plane and within the field of view angle is called the viewing volume. Only those objects which are contained inside the viewing volume are drawn. Any part of an object that is contained outside the viewing volume is clipped so that only the part that is in the viewing volume is displayed. The viewport that is created is mapped to the rendered view's window. Figure 4-2 shows a graphical depiction of how the viewing volume is created from the field of view angle and the near and far plane data.

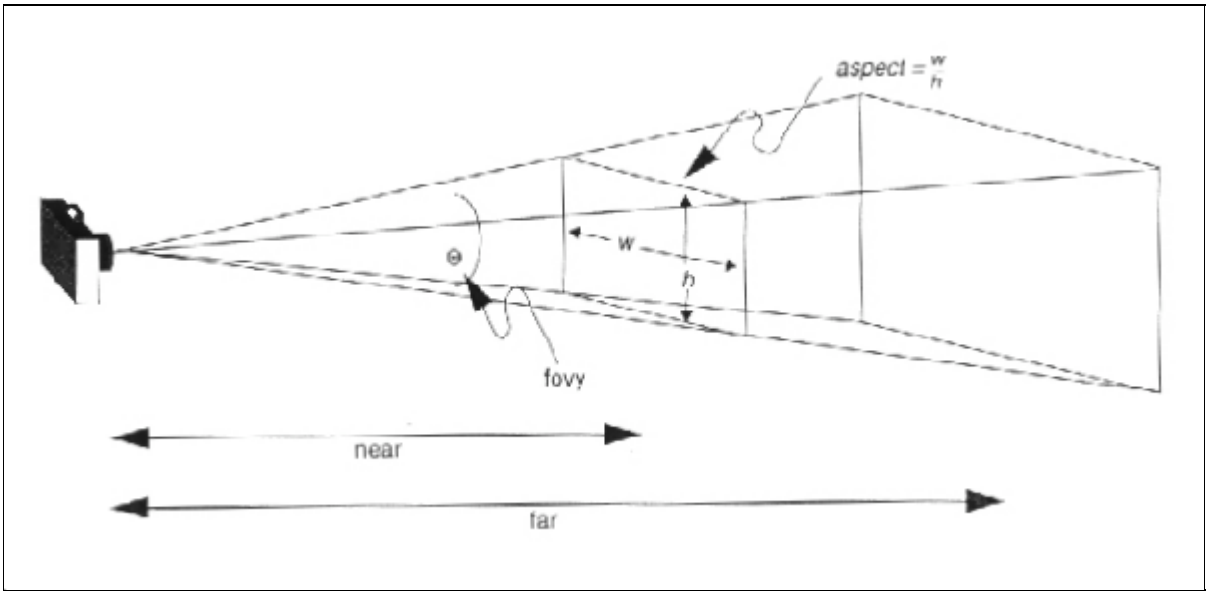

Figure 4-2 Viewing Volume

Lighting

Each light is characterized by a position and ambient, diffuse, and specular light contributions. Ambient light is the light that is scattered and doesn't appear to come from any source. Light in a room has a high amount of ambient light since most of the light has bounced off other objects in the room before reaching the eye. Conversely, a flashlight used outdoors has a very low ambient light since most of the light is focused into one direction and very little light will ever reach the eye after bouncing off an object. Diffuse light comes from a direction. The flashlight mentioned above will have a high diffuse component. The light is brighter if it hits an object normal to the surface than if it only glances it and the light will scatter equally in all directions when it strikes an object. Specular light comes from a particular direction also but tends to bounce off in a preferred direction. A laser beam is a good example of a light with a high specular component.

OpenGL combines the effects of all the lights used in order to decide how to color a shape. The final color of a vertex is calculated using the ambient, diffuse, and specular light and the ambient, diffuse, specular and shininess properties of the shape. Refer to [2] for a more detailed discussion of lighting mathematics.

4.4 Graphical User Interface

The graphical user interface is used to display the orthographic and perspective projections of the model and collect data from the user which specifies how to create and manipulate shapes displayed in the projections. One of the features of the interface is the ability to click on a shape displayed in an orthographic projection and drag it to a new location. The shape is automatically updated and redrawn in all views including the rendered view. Figure 4-3 shows a view of the main window as it would appear after executing the program.

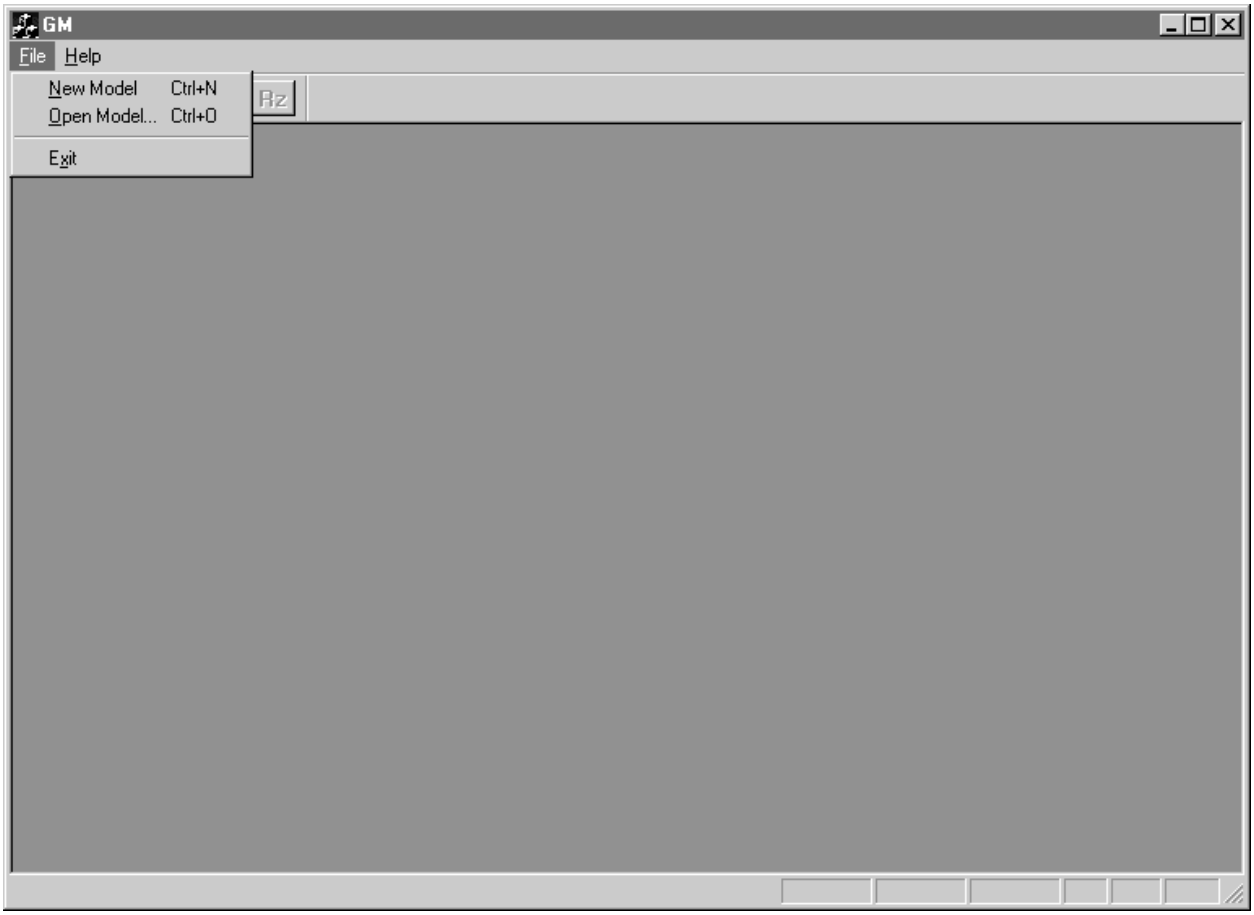

Figure 4-3 File Menu on Startup Screen

After creating a new file, the top, front, right and rendered views are displayed. The top view is shown in the upper left corner of the main window. The front view is in the bottom left corner. The right view is in the bottom right corner and the rendered view is on the top right corner. Figure 4-4 shows the screen after creating new file.

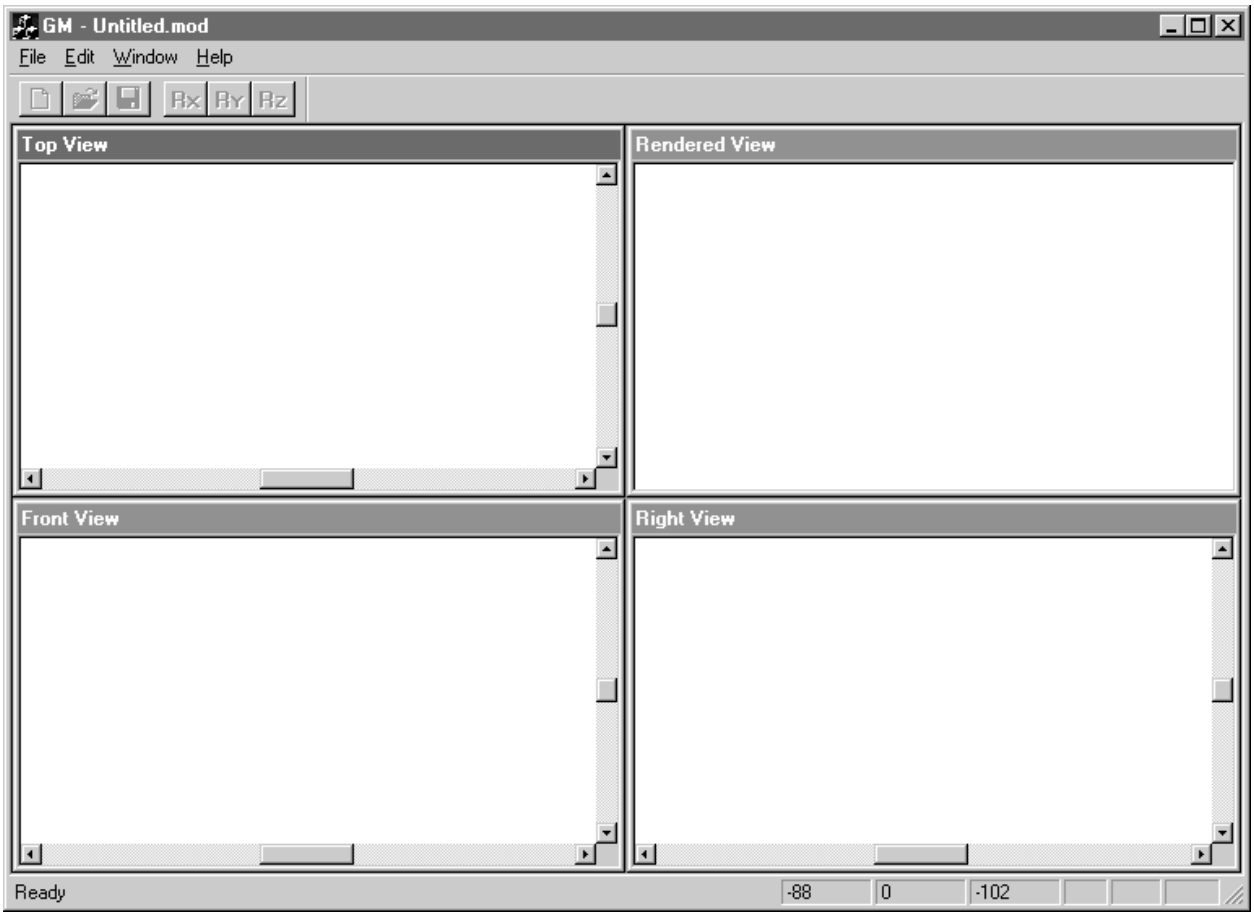

Figure 4-4 New Model Screen

The rendered view remains white until a camera is added to the model. In order to add a shape, camera or light to the model, the user moves the mouse pointer to either the top, front or right view and clicks the right mouse button. This will bring up a menu as shown in Figure 4-5. The user can then select the desired object and it will be placed at the coordinate of the point at which the right mouse button was clicked. The coordinates of the mouse are displayed in the first three boxes on the status bar at the bottom of the main window. The first, second and third box show the X, Y, and Z coordinate respectively.

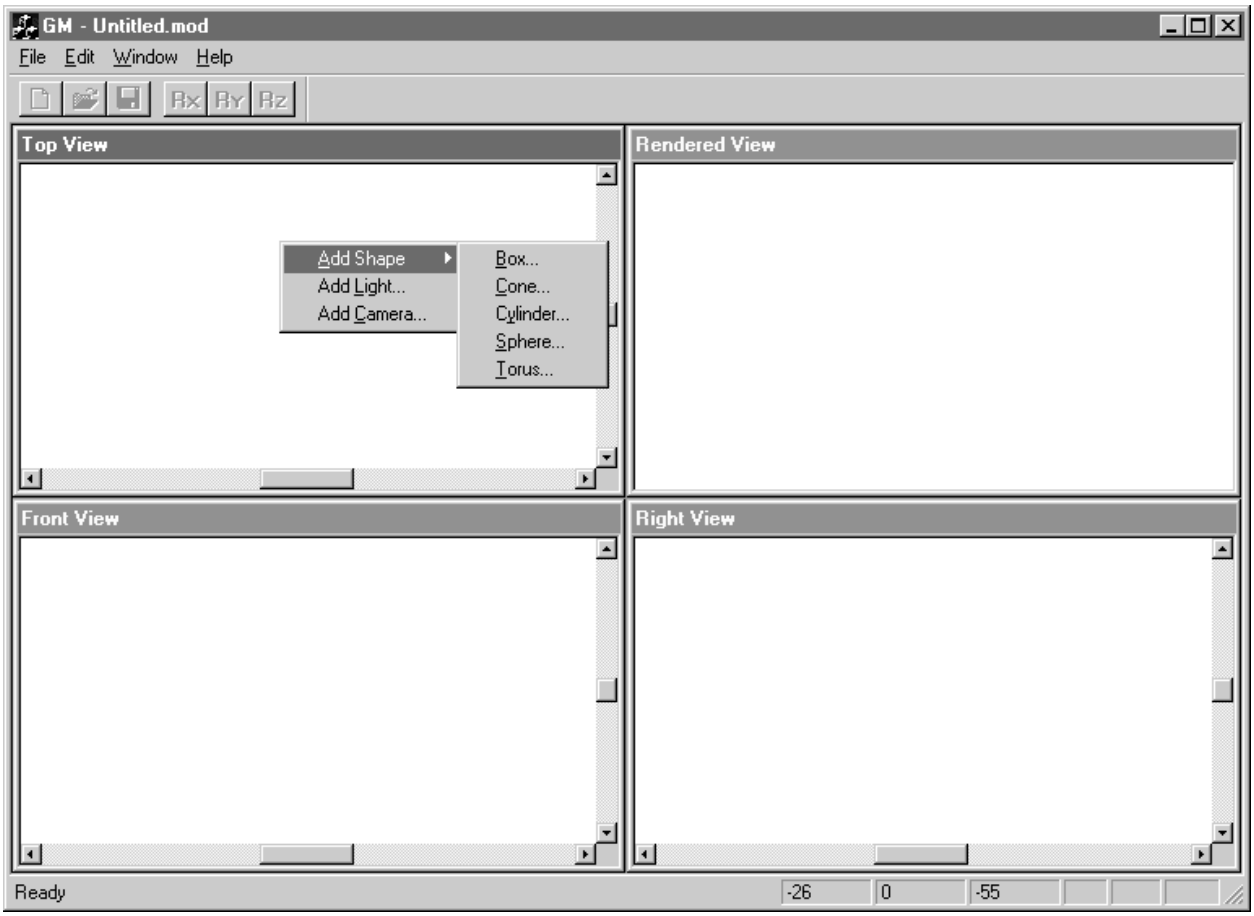

Figure 4-5 Floating Menu for Inserting Objects

After selecting a shape, a dialog box appears prompting the user to enter data that describes the physical properties of the shape. After entering the physical data, another dialog box appears prompting the user to enter data describing the material properties of the shape, i.e. the shapes ambient, diffuse, specular and shininess values. Figure 4-6 shows the Physical Properties dialog boxes and Figure 4-7 shows the Material Properties dialog box. The Material Properties dialog box is used by all shapes and the lights, although the lights do not use the shininess factor.

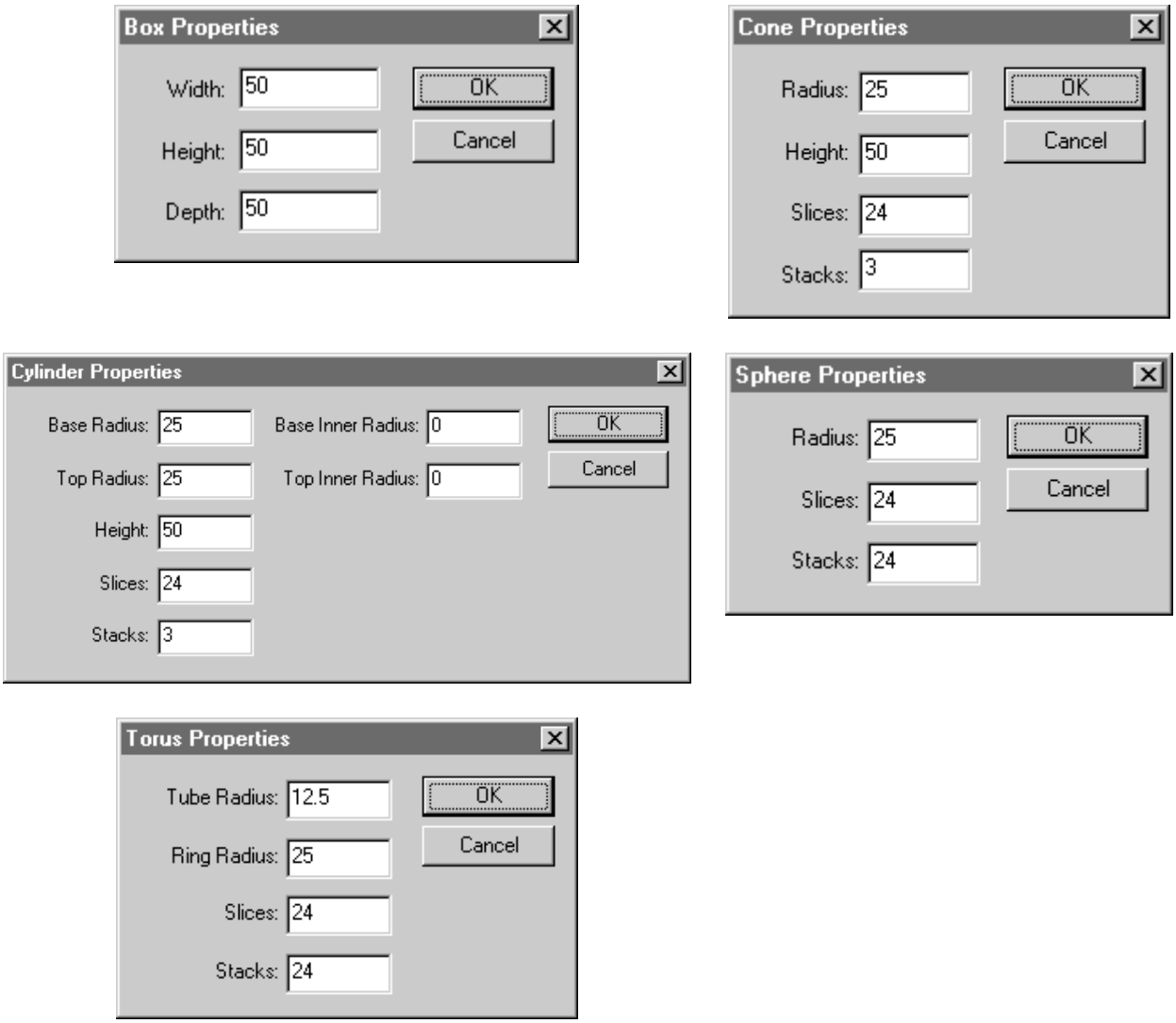

Figure 4-6 Physical Properties Dialog Boxes for all Shapes

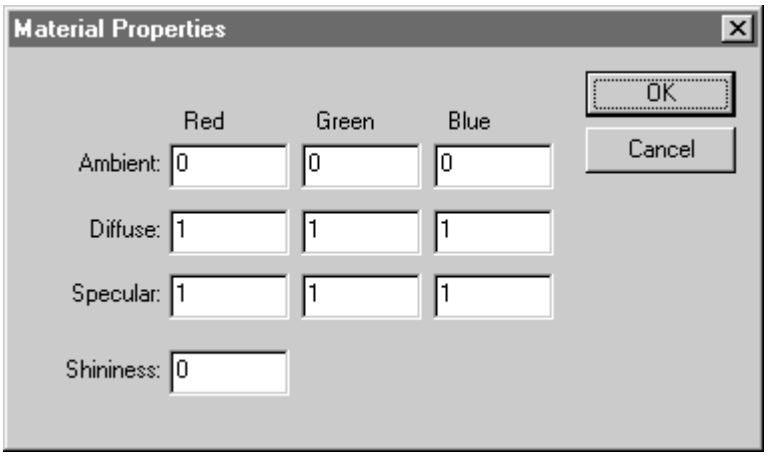

Figure 4-7 Material Properties Dialog Box

By choosing **Camera** from the menu, the Camera Properties dialog box shown in Figure 4-8 is displayed and the user is prompted to enter the field of view angle. This value is used by OpenGL to create the viewing volume and the viewport which is mapped to the rendered view.

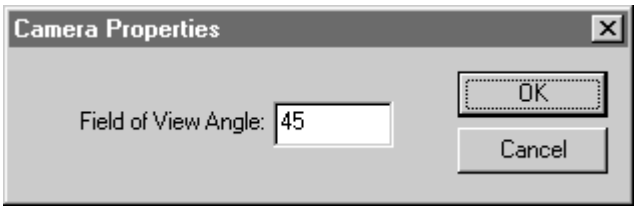

Figure 4-8 Camera Properties Dialog Box

After entering the appropriate data, an object is created and displayed in the top, front and right views. A light is displayed as an orange circle and a the camera is displayed as a blue circle. The camera has a focus point that is displayed as a smaller blue circle. A line is drawn between the camera and focus point to show the user where the camera is pointing and where it is focused. All shapes have a small dot at the center which is initially green. When the mouse

moves over an object the dot changes to yellow. If the user clicks on an object, the dot turns red to indicate to the user that it is currently the selected object. The selected object can be moved by pressing and holding down the left mouse button over the object and then moving the object to the desired location. While an object is being moved, a purple box appears in the place of the object and is moved along with the mouse. The box is the size of the smallest rectangle that contains the object. When the user releases the mouse button, the purple box is replaced with the original object and all views are updated so that the model is viewed correctly. An object can be rotated by selecting the object and pressing either the Rx, Ry or Rz button on the menu bar to rotate the object about the x, y or z axes respectively. The object will be rotated by an angle equal to the current rotation angle value. This value is initially 10 degrees but can be set to any value between negative and positive 180 degrees.

The focus point attached to the camera is used to rotate the camera. It also specifies the near and far planes. The longer the line connecting the focus point and the camera, i.e. the further away the camera is focused, the further away the near and far plane will be. Increasing the field of view angle will also cause the scene to appear further away from the viewer. Figure 4-9 shows the main screen after adding a cone. This cone will model the lead of a pencil.

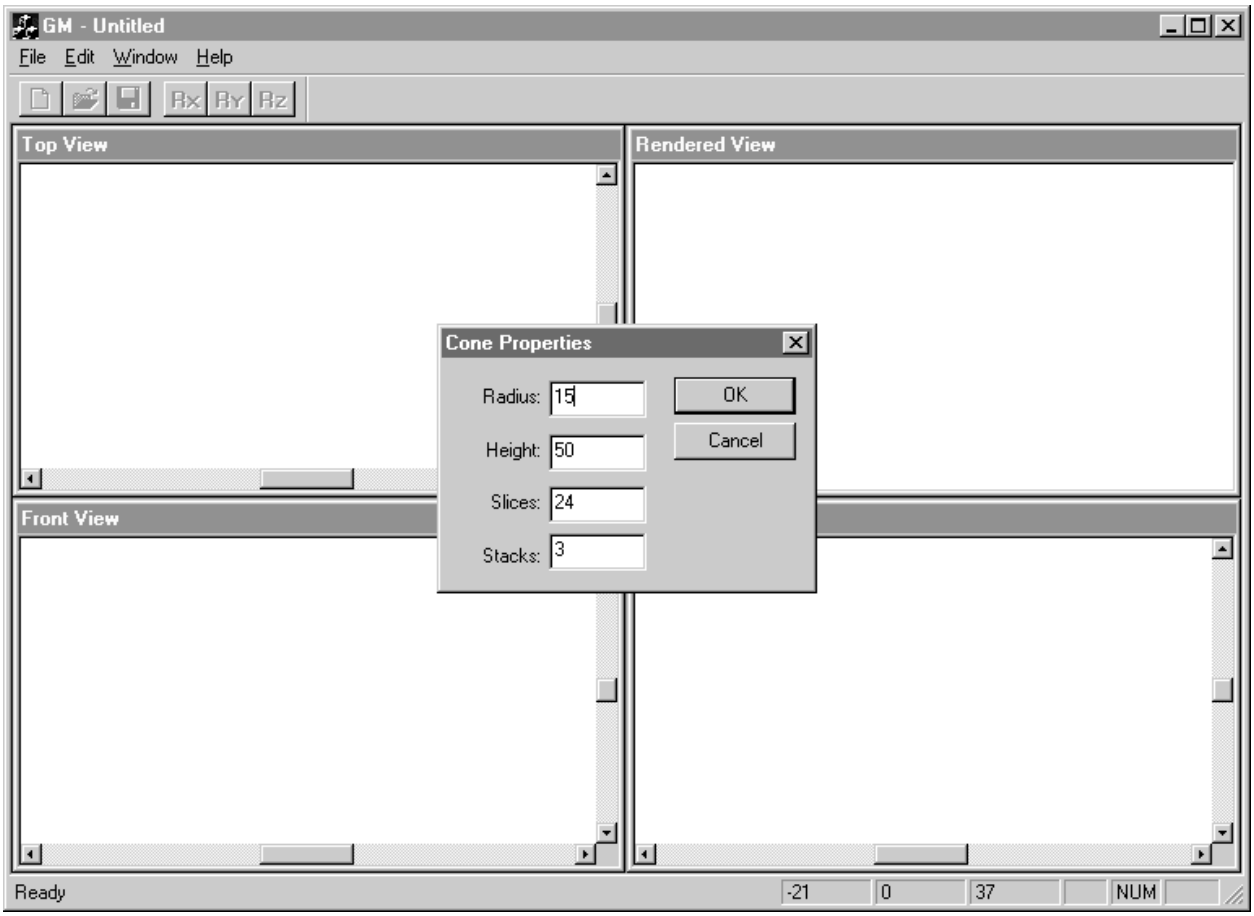

Figure 4-9 Main Screen with Cone Properties dialog

After selected OK, the Material Properties dialog box appears and the user is prompted to enter the amount of ambient, diffuse and specular light the object reflects. Note that all the dialog boxes have default values and minimum and maximum ranges. Figure 4-10 shows the screen with the Material Properties dialog box displayed.

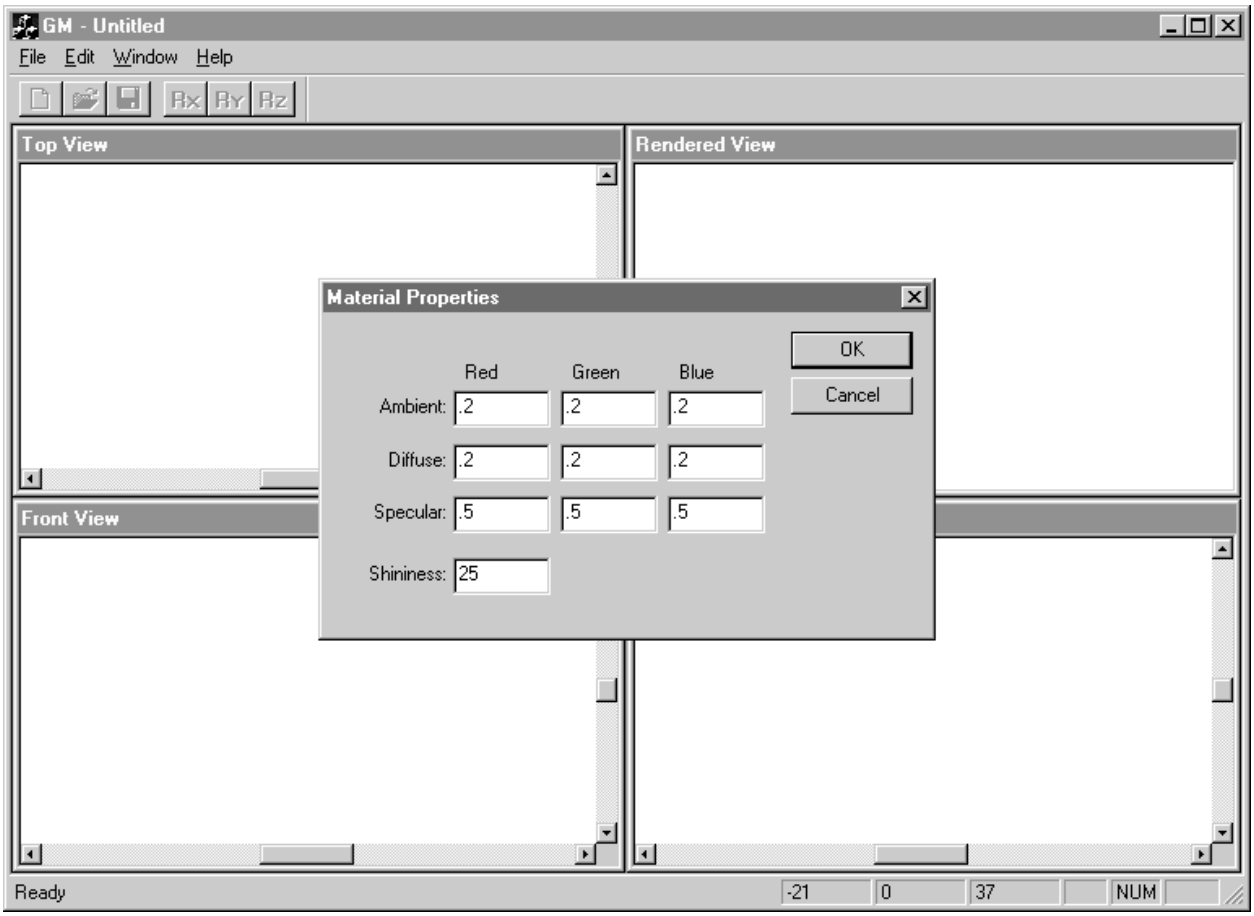

Figure 4-10 Material Properties dialog for New Shape

After entering the necessary information or choosing to use the default values, the newly created shape is displayed in the Top, Front and Right orthographic views as shown in Figure 4-11. The shape will appear centered at the location at which the user clicked the mouse. Each object has a small green dot at its center. When the user moves the mouse over the shape, the dot changes to yellow. If the user left clicks on a shape the dot changes to red indicating the it is the currently selected shape. Any other shape that was previously selected becomes de-selected and changes back to green.

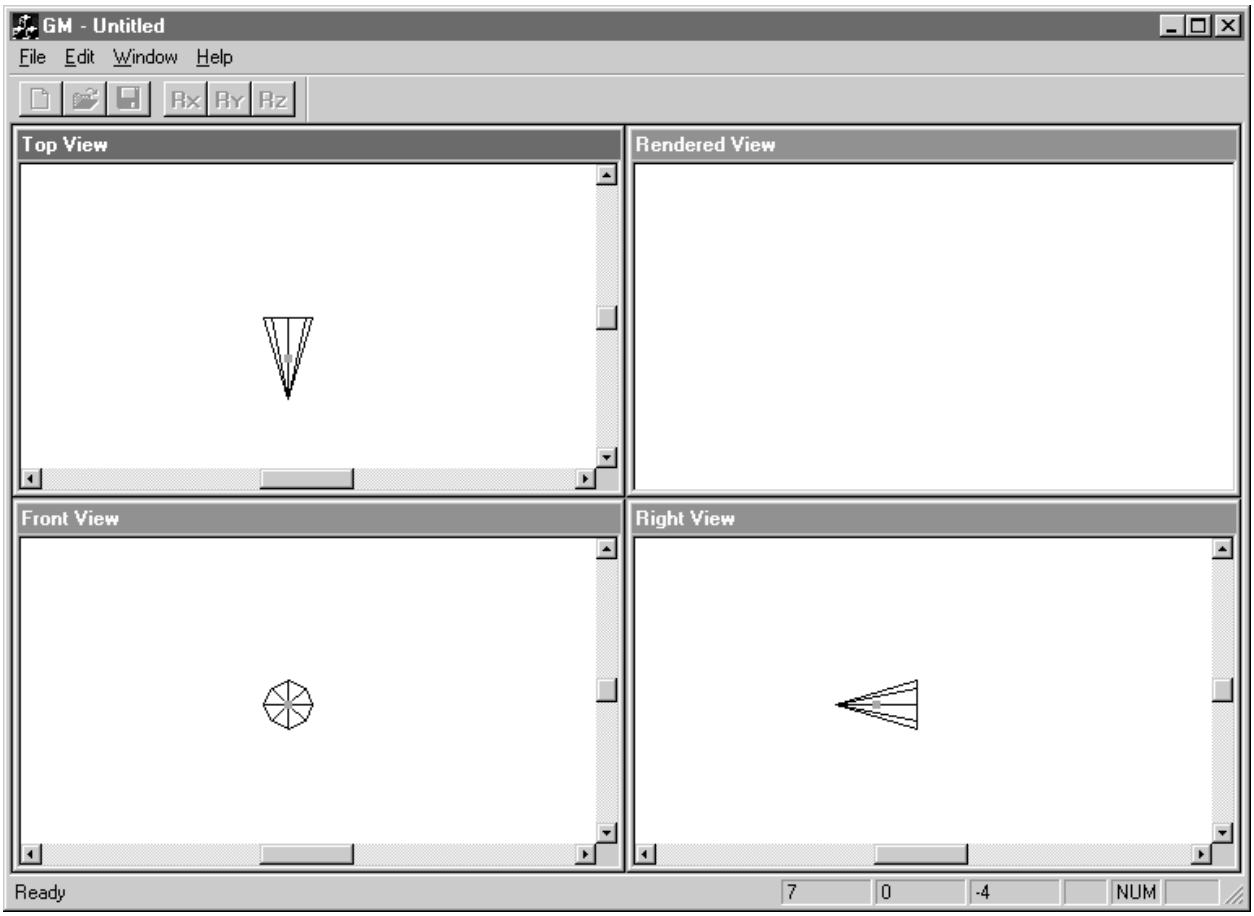

Figure 4-11 Orthographic Views of New Shape

By choosing **Add Light** from the floating menu, the Material Properties dialog box appears and prompts the user to enter the amount of ambient, diffuse and specular light that is emitted by the new light. Figure 4-12 shows the dialog box.

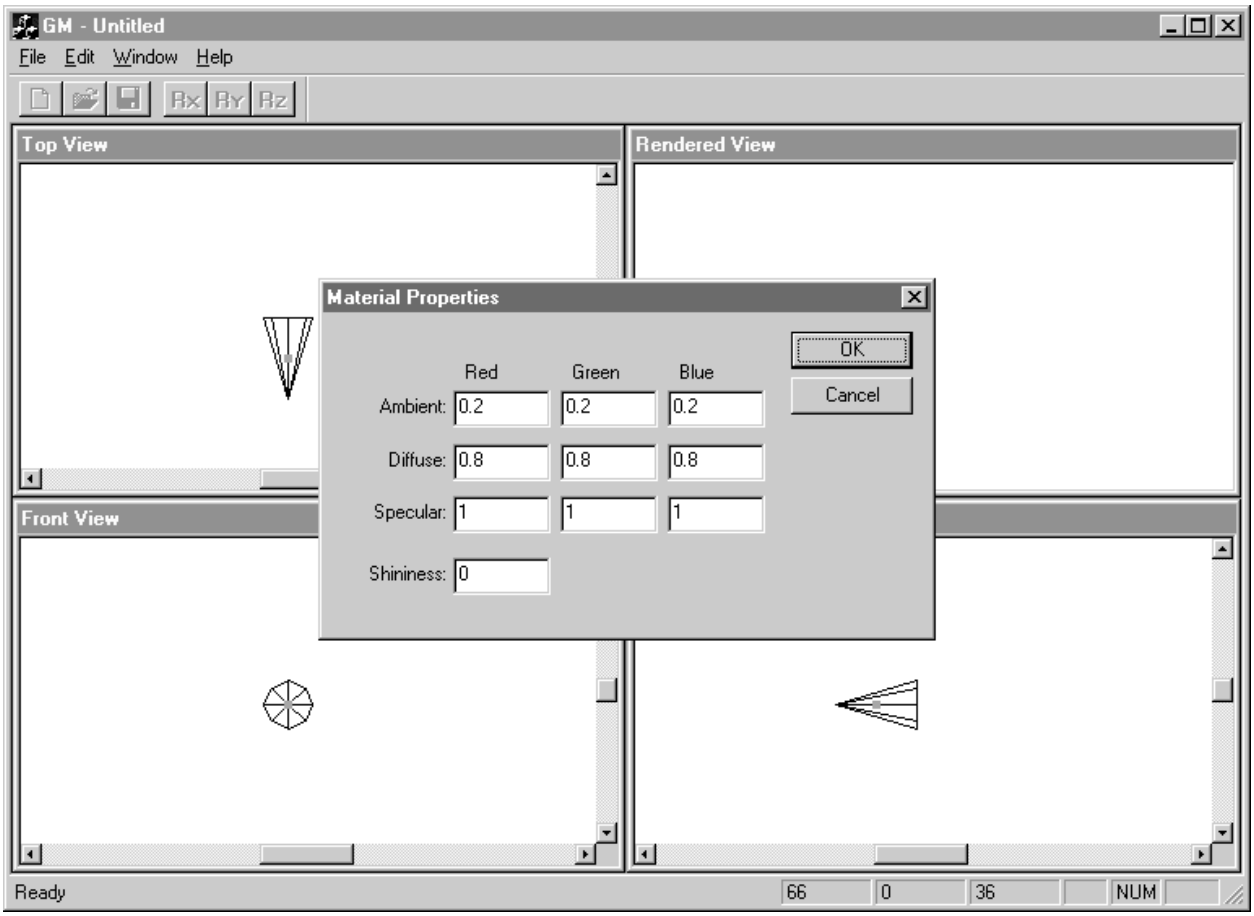

Figure 4-12 Properties Used by Each Light

Figure 4-13 shows the screen after the light is added to the views. It also shows the Camera Properties dialog box that appears after selected **Add Camera** from the floating menu.

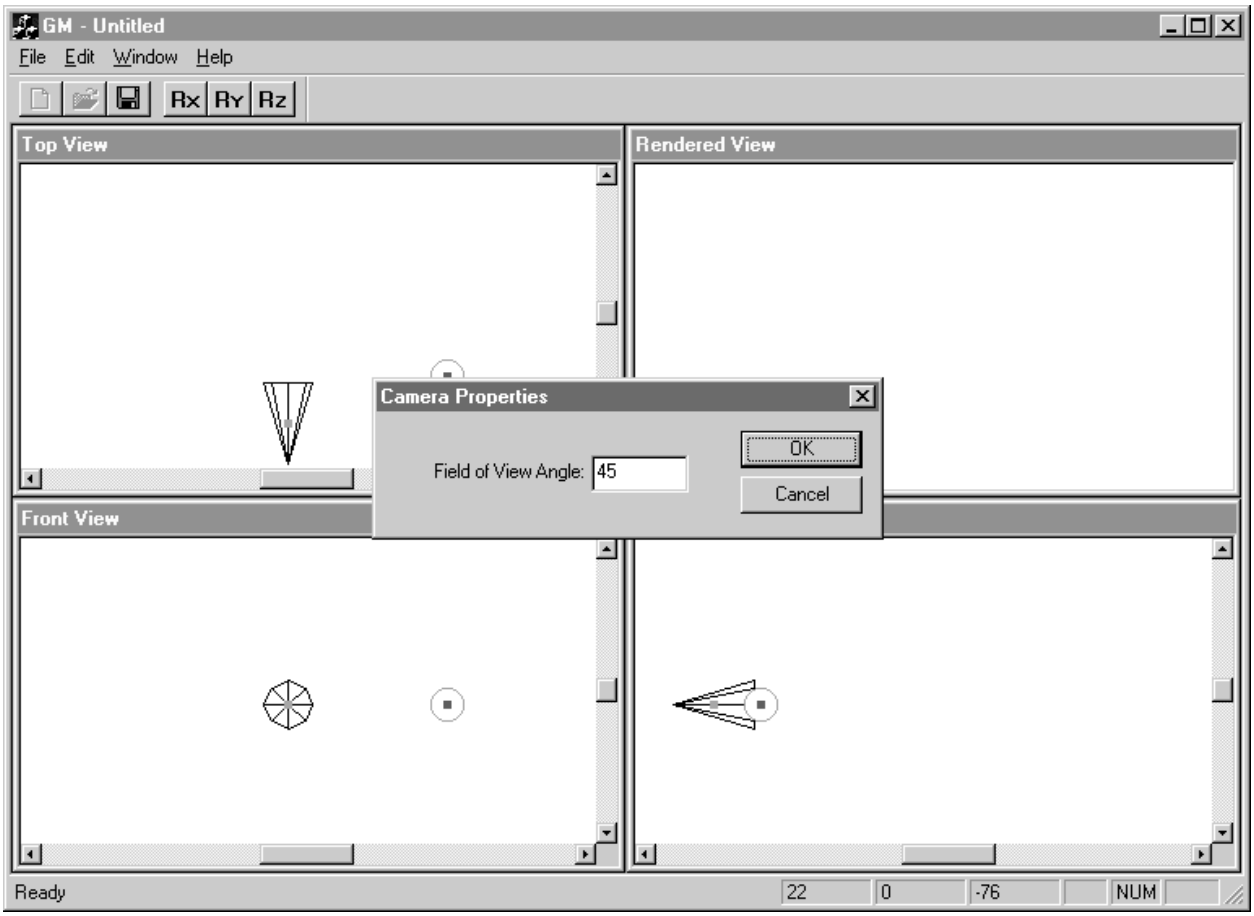

Figure 4-13 Camera Properties

Figure 4-14 shows the screen after the camera is added. The camera appears as a blue circle. The camera focus point is drawn as a smaller blue circle. There is a line between the camera and the camera focus point indicating the direction the camera is facing. By moving the camera and focus point, the model can be rendered from any location and angle.

Notice that the focus point and light have been moved. To move an object, the user must press and hold the left mouse button over an object. As the user moves the mouse to a new location, a small purple box appears indicating where the object will be translated if the user releases the mouse button. When the user releases the mouse, the object is erased at its current

location and redrawn at the new location. The dragging box appears in the Top, Front and Right views simultaneously as the user drags the box.

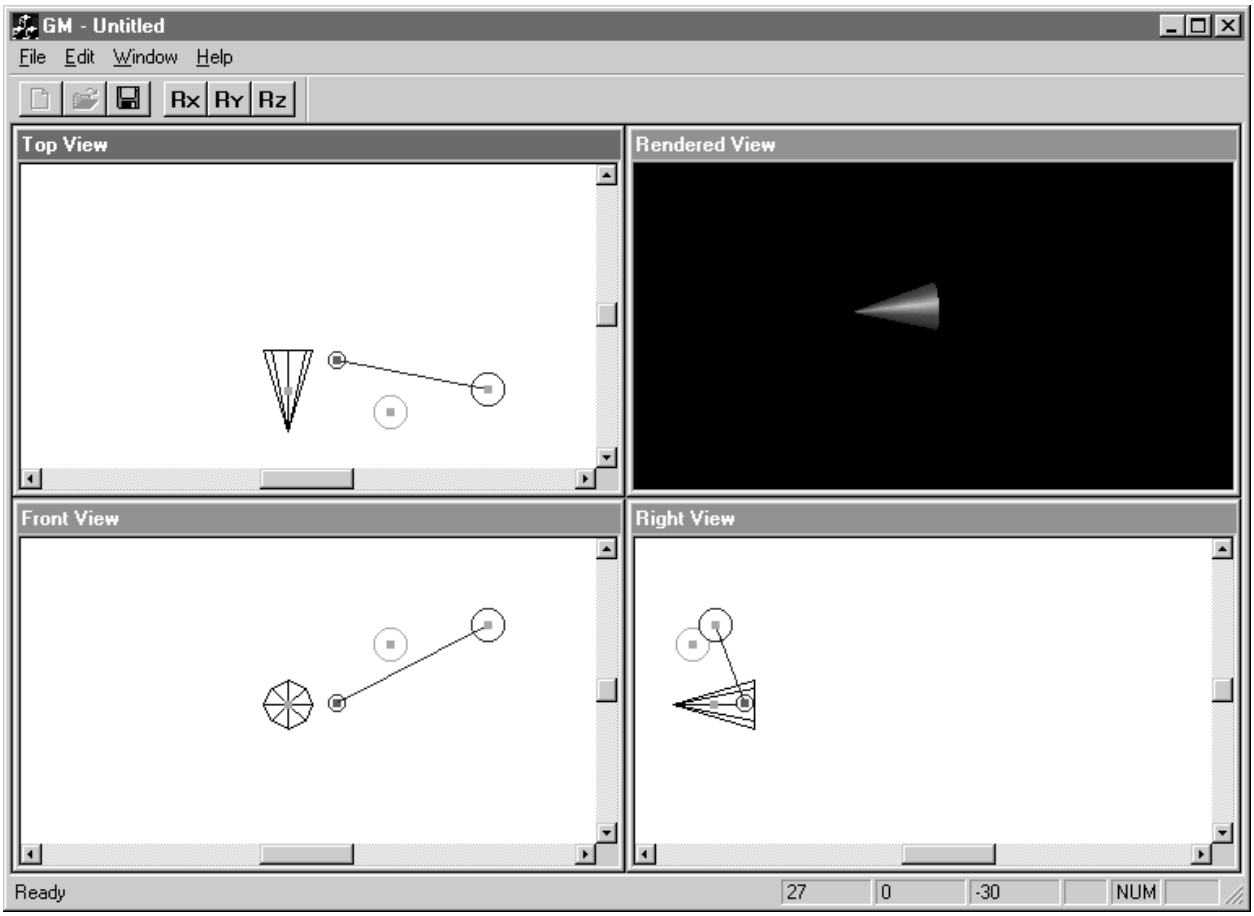

Figure 4-14 Camera

After choosing to add a new cone to model the wooden end of the pencil, the scene will be displayed as shown in Figure 4-15.

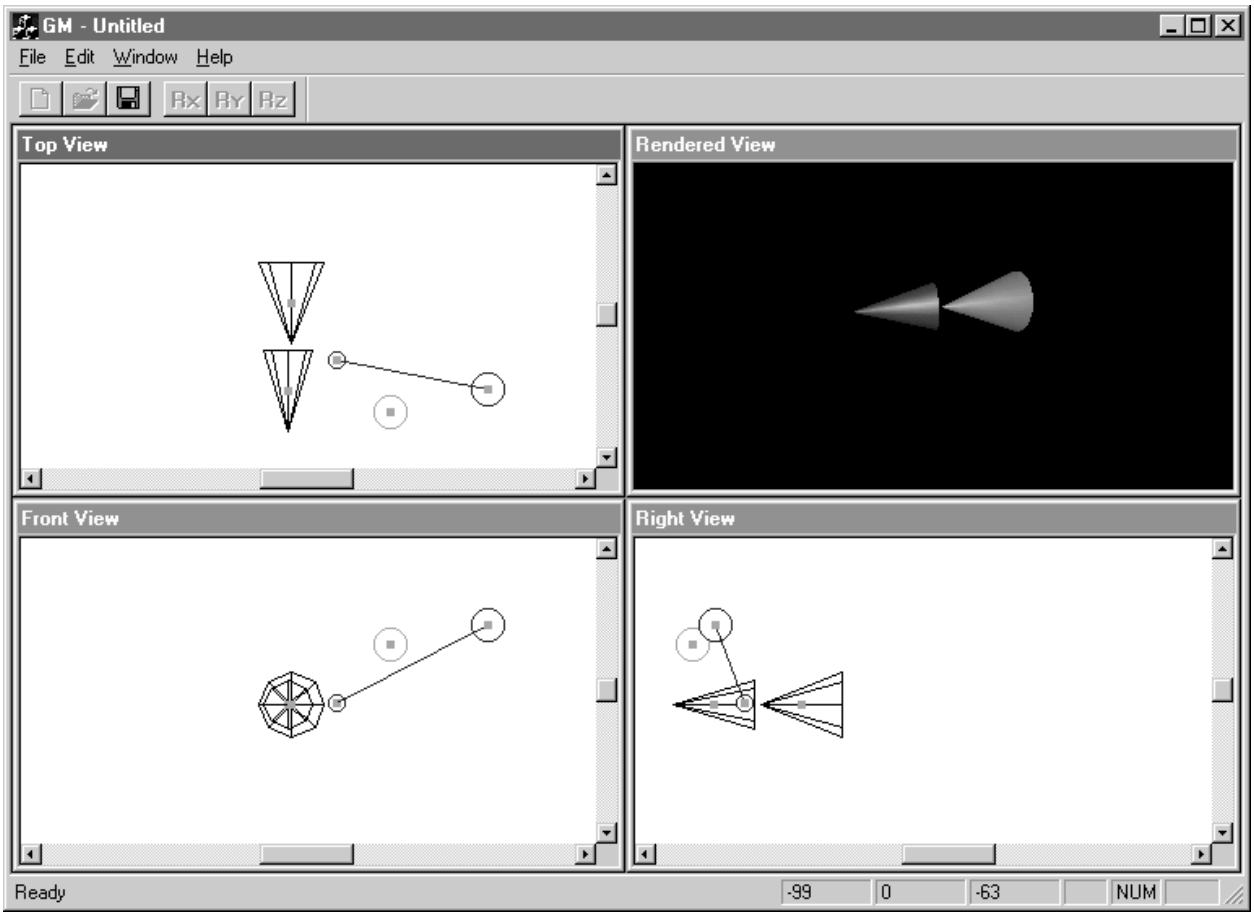

Figure 4-15 Added a Wooden Cone

Figure 4-16 shows the scene after adding a cylinder. The wooden cone and the cylinder are moved so the cone overlaps the first cone and the cylinder is flush with the wooden cone. The cylinder is colored yellow.

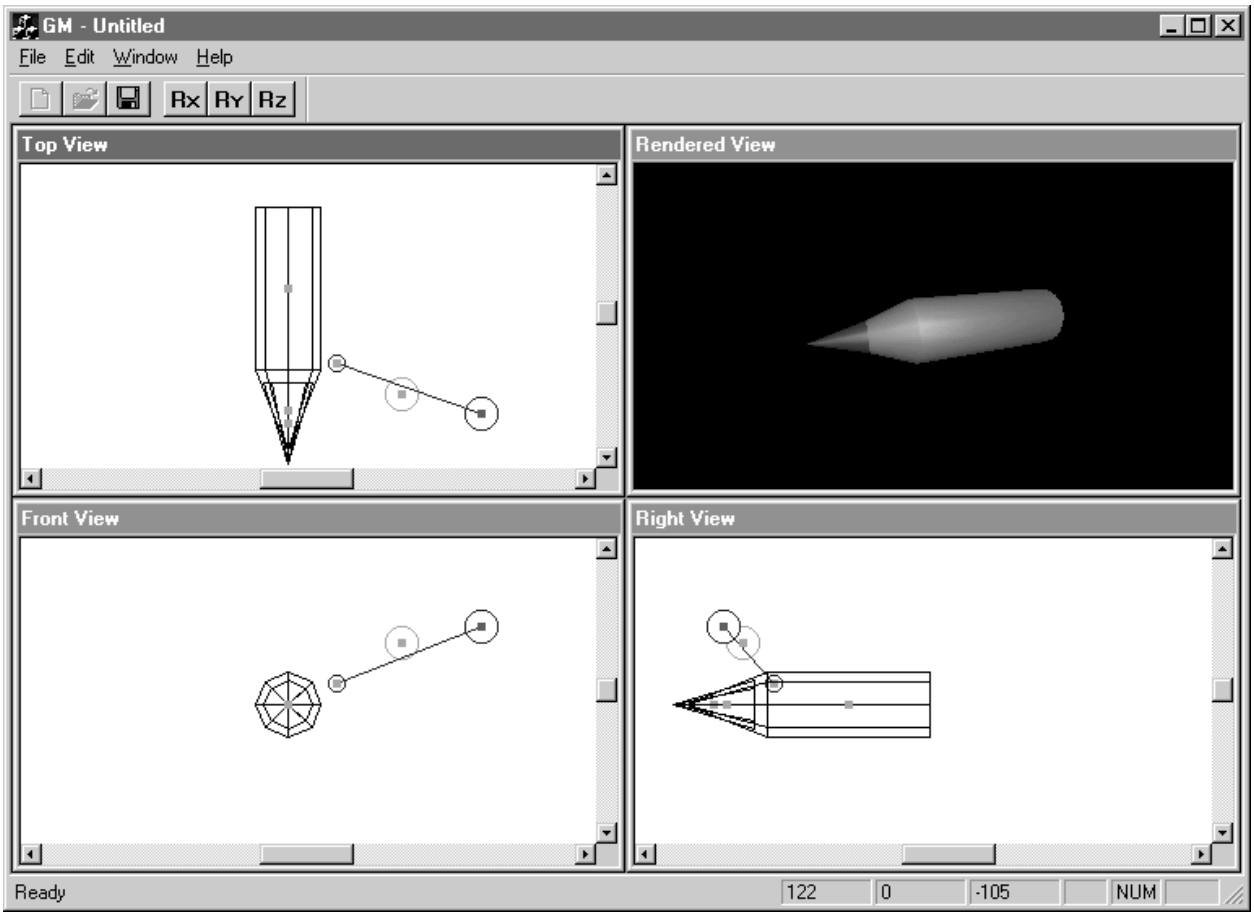

Figure 4-16 Added Cylinder and Lined Up Shapes

Finally, Figure 4-17 shows the scene after adding a torus and another cylinder. The torus is given material properties so that it appears to made of metal and will be used to model the metal ring that holds the eraser on the pencil. The cylinder is colored orange to model an eraser.

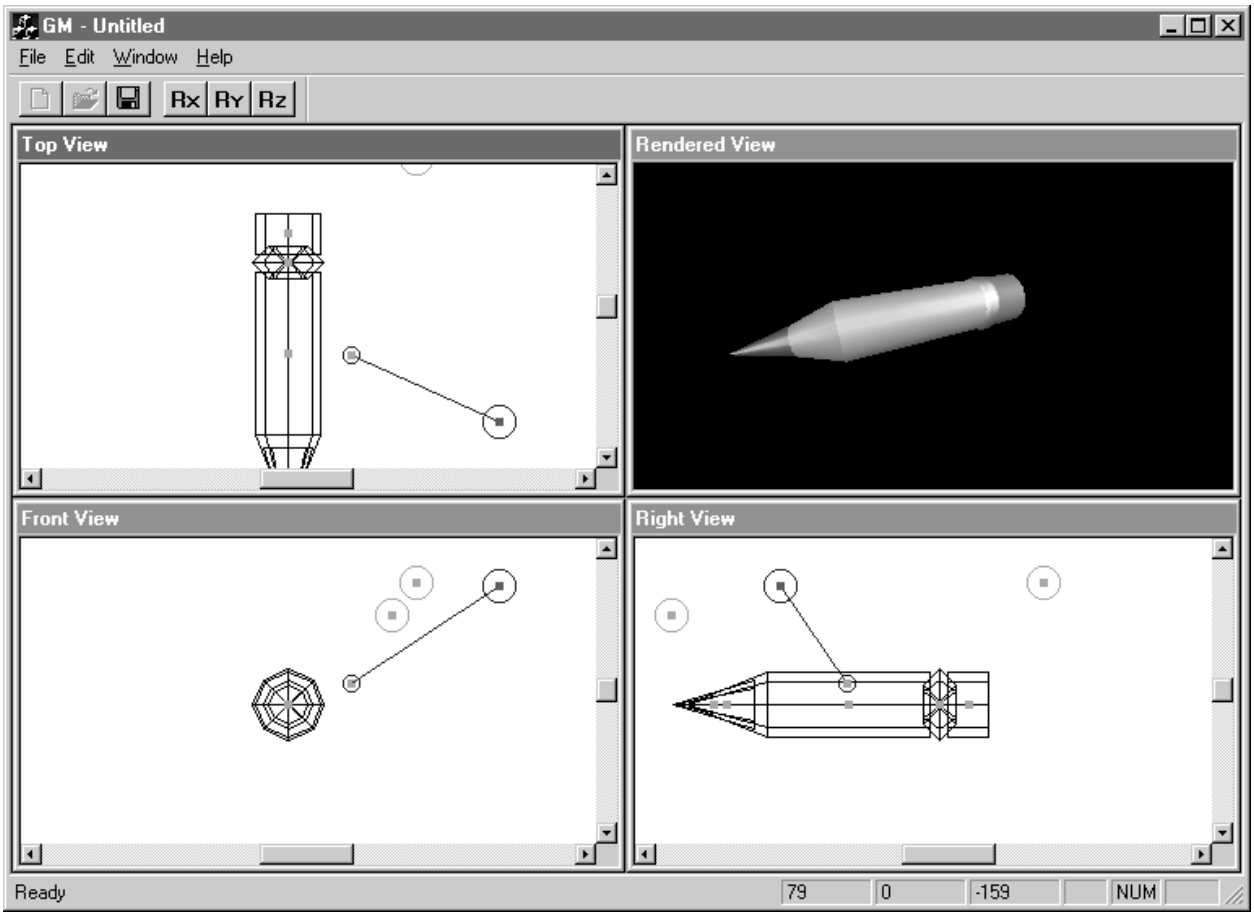

Figure 4-17 Pencil Model

By right clicking the mouse on any object displayed in the top, front or right view another menu appears that allows the user to delete the object or change the objects properties or material. If the object is a shape or light, choosing **Material** will open the Material Properties dialog box and the user has the option of changing the ambient, diffuse, specular or shininess values. If the object is a camera, the selection will have no effect. Choosing **Properties** will open the appropriate Properties Dialog box and the user has the option of changing the shapes dimensions. If the object is a camera, the Camera Properties dialog box is displayed and the user

can change the camera's field of view angle. If an object is currently selected, its properties and material can also be changed by selecting the appropriate items from the Edit menu.

CHAPTER V

Conclusion and Future Work

5.1 Conclusion

This project has focused on the design and development of a useful, intuitive and user-friendly Three-Dimensional Geometric Modeling application. The prototype application has succeeded in providing this interface to the user. Clearly many useful features have been realized without compromising the interface and a foundation has been built on which to add other powerful features.

Other Three-Dimensional Geometric Modeling applications on the market were reviewed. Some of these products, although rich in features had a steep learning curve and were nearly unusable to the untrained user.

In an attempt to solve some the deficiencies of the reviewed systems, a prototype of a more intuitive and user-friendly Three-Dimensional Geometric Modeling application was constructed. This improved interface allows the user to view the object from several angles and to see a rendering of the model at all times. This allows the user to easily move objects in space and to immediately see the effects the movement has on the rendered scene.

5.2 Lessons Learned

Several lessons were learned during the development of this project. The primary goal was to obtain a better understanding of three-dimensional modeling using high-resolution graphics. Designing an intuitive user-friendly interface for the modeler was another goal. Since OpenGL for Windows 95 was the only implementation of OpenGL available to the author,

Windows 95 was chosen as the target operating system for the software. During the developmental stages of the project, Windows 95 programming became the hardest obstacle to overcome. Having not had any prior programming experience for the Windows 95 operating system, simply drawing a line in a window or responding to a button being pressed became programming obstacles. As a result, most of the time and effort spent in the development of this project was in creating the interface instead of adding more features to the modeling side. Although much was learning about three-dimensional modeling and graphic programming, perhaps the most difficult skill learning was how to write an application for Windows 95.

5.3 Future Work

The prototype developed in this project provides only the basic, though powerful, features of a full-blown implementation of a Three-Dimensional Geometric Modeling application. Such an application would provide features such as antialiasing, blending, warping, texture mapping, animation, and inverse kinematics. All of these features except for inverse kinematics are supported by OpenGL. These areas should be considered in the order given should any future work be attempted. Each addition should be designed and implemented in an effort to provide added functionality while maintaining an intuitive user-friendly interface.

REFERENCES

1. Glaeser, Georg. *Fast Algorithms for 3D-Graphics*. Springer-Verlag, 1994.

2. Neider, Jackie, Tom Davis, and Mason Woo. *OpenGL Programming Guide : The Official Guide to Learning OpenGL, Release 1*. Addison-Wesley, 1993.

3. Taylor, Philip H. *3D Graphic Programming in Windows*. Addison-Wesley, 1994.

4. Watt, Alan H. *3D Computer Graphics*. Addison-Wesley, 1993.

5. Walnum, Calyton. *3-D Graphics Programming with OpenGL*. Que Corp., 1995.

6. MacNicol, G., C. Lyn, D. Barnard and G. Williams. "Hot 3-D." *Digital Video Magazine* 3(2):37-58, Feb. 1995.

7. Foust, John. "3-D Steps Forward." *BYTE* 20:123-130, July 1995.