Chapter 3

Computer Graphics Software

Exercises

3-1. We set the color of a display window to dark gray by selecting a value for each of the three RGB parameters in the `glClearColor` command and passing the values to the function. To set the display window to dark gray, we could call:

```
glClearColor (0.1, 0.1, 0.1, 0.0);
```

Any values near zero would do, as long as all three values are equal.
To set the display window to white, we could call:

```
glClearColor (1.0, 1.0, 1.0, 0.0);
```

3-2. The following statements set the lower-left corner of a 200 pixel by 150 pixel display window at screen coordinates (75, 200):

```
glutInitWindowPosition (75, 50);
glutInitWindowSize (200, 150);
```

3-3. The following OpenGL statements draws a line from the upper-right corner of a display window of width 150 and height 250 to the lower-left corner:

```
glBegin (GL_LINES);
      glVertex2i (150, 0);
      glVertex2i (0, 250);
glEnd ( );
```

3-4. The following statement invokes an OpenGL callback function to register the rectangle function with the display window:

```
glutDisplayFunc(rectangle);
```

3-5. A callback function is a procedure that is to be invoked whenever a particular action occurs. Such a procedure is registered by listing the procedure name in an appropriate OpenGL function.

3-6. Modeling coordinates are specified in a coordinate reference frame that is local to a given object. They are used to define the relative positions of sub-components of the object in terms of other subcomponents. World coordinates are specified in a global coordinate system that contains many objects, each with their own modeling coordinates. Each object's modeling coordinates must be transformed into world coordinates in order to place the object at a given position and orientation in the world coordinate frame.

3-7. Normalized coordinates are 2D coordinates that have been mapped from a given range of horizontal and vertical positions to ranges that are either -1 to 1 or 0 to 1. This is useful because specifying coordinates in this way makes the coordinates device-independent; that is, independent of the specific range of coordinates for a given graphics hardware device. Thus, the coordinates output by the software package can be used by many different devices without needing to specify the device or its particular ranges.

Themed Exercises

3-1. The student should write a specification that is sufficiently detailed in description given the fairly abstract material presented so far. Layman terms for object properties and dynamics are sufficient, but should be specific enough to clarify the design. The initial iterations of the application will deal with two-dimensional representations; therefore, it is ideal if the student can come up with a two-dimensional formulation of the application and a rough draft of a three-dimensional as well. Incorporation of complex shapes and textures, curved surfaces, hierarchical object models, varying lighting conditions, responsiveness to user input, and animation are also important considerations for future exercises.

3-2. Information on obtaining the required libraries for a given system can be found at the Web sites provided. The student should have set up an appropriate environment and run the example program in this chapter. Though the examples in the book are all in C++, bindings for languages other than C++ can be found via these sites as well. If the student has obtained approval from the instructor to use a language other than C++, he or she should successfully compile and run the example program in this chapter in their preferred language to ensure that writing a graphics application in this language will be feasible.