1

**Objects and Models**

Chapter Objectives:

- Introduce basic graphics concepts — object, model, image, graphics library, frame buffer, scan-conversion, clipping, and anti-aliasing
- Set up an OpenGL programming environment
- Understand simple OpenGL programs

1.1 Graphics Models and Libraries

A graphics *display* is a drawing area comprised of an array of fine points called pixels. At the heart of a graphics system there is a magic pen, which can move at lightning speed to a specific pixel and draw the pixel with a specific color — a red, green, and blue (RGB) vector value. This pen can be controlled directly by hand through an input device (mouse or keyboard) like a simple paintbrush. In this case, we can draw whatever we imagine, but it takes a real artist to come up with a good painting. Computer graphics, however, is about using this pen automatically through programming.

A real or imaginary *object* is represented in a computer as a model, and is displayed as an image. A *model* is an abstract description of the object’s shape (vertices) and attributes (colors), which can be used to find all the points and colors on the object corresponding to the pixels in the drawing area. Given a model, the application program will control the pen through a graphics library to generate the corresponding image. An *image* is simply a 2D array of pixels.

A *graphics library* provides a set of graphics commands or functions. These commands can be bound in *C*, *C++*, *Java*, or other programming languages on different platforms. Graphics commands can specify primitive 2D and 3D geometric
Objects and Models

Models to be digitized and displayed. Here *primitive* means that only certain simple shapes (such as points, lines, and polygons) can be accepted by a graphics library. To draw a complex shape, we need an application program to dissect it into pieces of simple shapes (primitives). We have the magic pen that draws a pixel. If we can draw a pixel, we can draw a line, a polygon, a curve, a block, a building, an airplane, and so forth. A general application program can be included into a graphics library as a command to draw a complex shape. Since our pen is magically fast, we can draw a complex object, clear the drawing area, draw the object at a slightly different location, and repeat the above processes — the object is now *animated*.

*OpenGL* is a graphics library, which we will integrate with the *C programming language* to introduce graphics theory, programming, and applications.

### 1.2 OpenGL Programming

OpenGL is the most widely used graphics library (GL) or application programming interface (API), which is supported across all popular desktop and workstation platforms, ensuring wide application deployment. First, let’s spend some time to set up our working environment, compile Example 1.1.point.c, and run the program. The following file contains links to all the example programs in this book, and detailed information for setting up working environments on different platforms:

http://www.cs.gmu.edu/~jchen/graphics/setup.html

/* Example 1.1.point.c: draw randomly generated points */

```c
#include <stdlib.h>
#include <GL/glut.h>

#define Height 400
#define Width 400

void display(void)
{
   int x, y;

   //a. generate a random point
   x = rand() % Width;
   y = rand() % Height;
```