Computer Graphics

Dr./ Ahmed Mohamed Rabie

Chapter 1

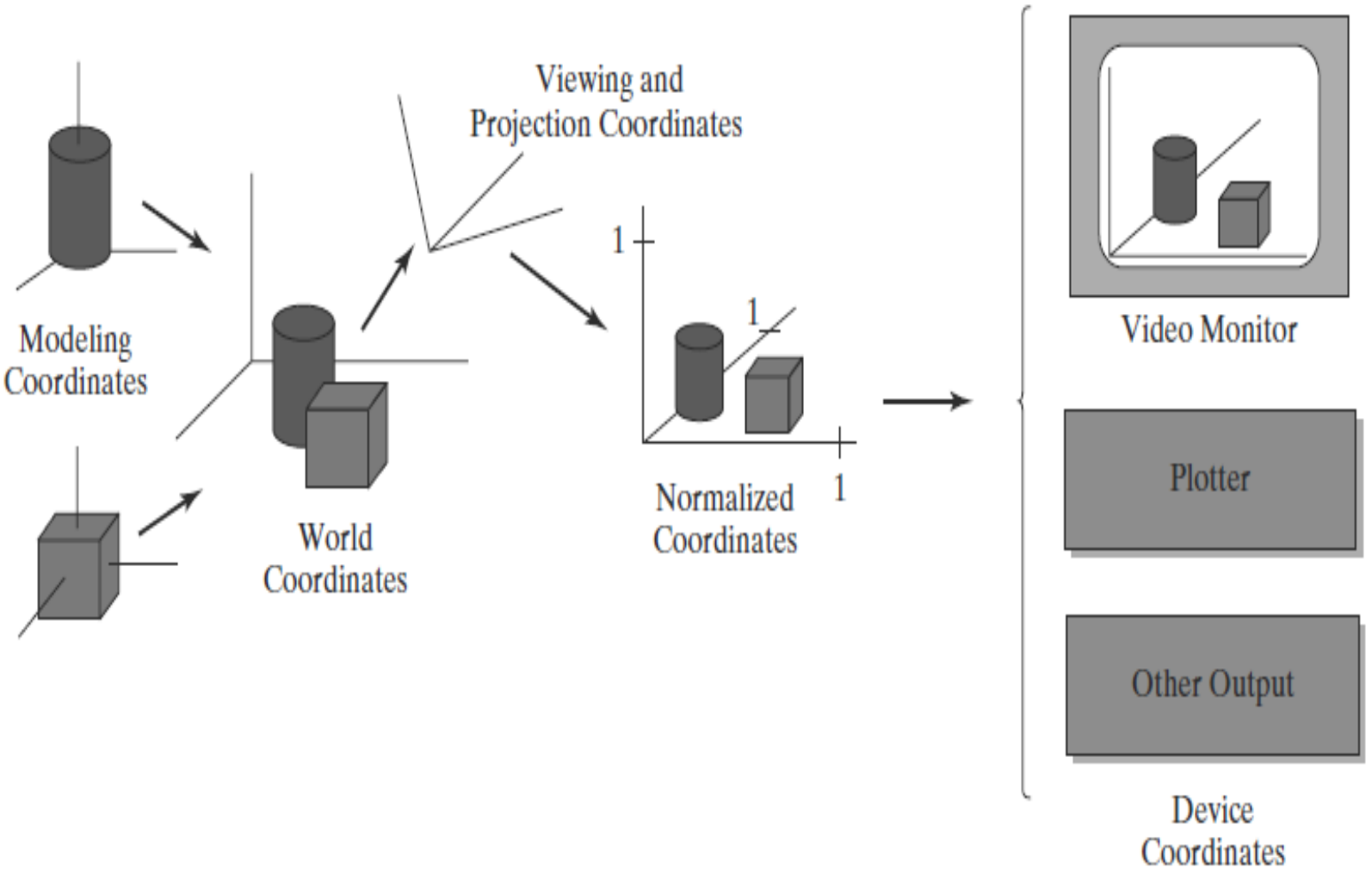
Introduction

Coordinate Representations

To generate a picture using a programming package, we first need to give the geometric descriptions of the objects that are to be displayed. These descriptions determine the locations and shapes of the objects. For example, a box is specified by the positions of its corners (vertices), and a sphere is defined by its center position and radius.

The transformation sequence from modeling coordinates to device coordinates for a three-dimensional scene. Object shapes can be individually defined in modeling-coordinate reference systems. Then the shapes are positioned within the world-coordinate scene.

First, we can define the shapes of individual objects, such as trees or furniture, within a separate reference frame for each object. These reference frames are called modeling coordinates. Once the individual object shapes have been specified, we can construct ("model") a scene by placing the objects into appropriate locations within a scene reference frame called world coordinates.


Viewing and
Projection Coordinates
1
1
1
Modeling
Coordinates
World
Coordinates
Normalized
Coordinates
Video Monitor
Plotter
Other Output
Device
Coordinates

This step involves the transformation of the individual modeling-coordinate frames to specified positions and orientations within the world-coordinate frame. Geometric descriptions in modeling coordinates and world coordinates can be given in any convenient floating-point or integer values, without regard for the constraints of a particular output device.

After all parts of a scene have been specified, the overall world-coordinate description is processed through various routines onto one or more output-device reference frames for display. This process is called the viewing pipeline. World coordinate positions are first converted to viewing coordinates corresponding to the view we want of a scene, based on the position and orientation of a hypothetical camera.

Then object locations are transformed to a two-dimensional (2D) projection of the scene, which corresponds to what we will see on the output device. The scene is then stored in normalized coordinates, where each coordinate value is in the range from −1 to 1 or in the range from 0 to 1, depending on the system.

Normalized coordinates are also referred to as normalized device coordinates, since using this representation makes a graphics package independent of the coordinate range for any specific output device. Finally, the picture is scan-converted into the refresh buffer of a raster system for display. The coordinate systems for display devices are generally called device coordinates.

A general graphics package provides users with a variety of functions for creating and manipulating pictures. These routines can be broadly classified according to whether they deal with graphics output, input, attributes, transformations, viewing, subdividing pictures, or general control.

Graphic Functions

A general-purpose graphics package provides users with a variety of functions for creating and manipulating pictures. These routines can be broadly classified according to whether they deal with graphics output, input, attributes, transformations, viewing, subdividing pictures, or general control.

The basic building blocks for pictures are referred to as **graphics output primitives**. They include character strings and geometric entities, such as points, straight lines, curved lines, filled color areas (usually polygons), and shapes defined with arrays of color points.

In addition, some graphics packages provide functions for displaying more complex shapes such as spheres, cones, and cylinders. Routines for generating output primitives provide the basic tools for constructing pictures. **Attributes** are properties of the output primitives; that is, an attribute describes how a particular primitive is to be displayed. This includes color specifications, line styles, text styles, and area-filling patterns.

We can change the size, position, or orientation of an object within a scene using **geometric transformations**. Some graphics packages provide an additional set of functions for performing modeling transformations, which are used to construct a scene where individual object descriptions are given in local coordinates.

Such packages usually provide a mechanism for describing complex objects (such as an electrical circuit or a bicycle) with a tree (hierarchical) structure. Other packages simply provide the geometric-transformation routines and leave modeling details to the programmer.

Viewing transformations are used to select a view of the scene, the type of projection to be used, and the location on a video monitor where the view is to be displayed. Other routines are available for managing the screen display area by specifying its position, size, and structure. For three-dimensional scenes, visible objects are identified and the lighting conditions are applied.

Interactive graphics applications use various kinds of input devices, including a mouse, a tablet, and a joystick. Input functions are used to control and process the data flow from these interactive devices. Some graphics packages also provide routines for subdividing a picture description into a named set of component parts. And other routines may be available for manipulating these picture components in various ways.

Finally, a graphics package contains a number of housekeeping tasks, such as clearing a screen display area to a selected color and initializing parameters. We can lump the functions for carrying out these chores under the heading control operations.

Software Standard

When packages are designed with standard graphics functions, software can be moved easily from one hardware system to another and used in different implementations and applications.

Without standards, programs designed for one hardware system often cannot be transferred to another system without extensive rewriting of the programs.

GL (Graphics Library), which very soon became a widely used package in the graphics community. Thus, GL became a de facto graphics standard. The GL routines were designed for fast, real-time rendering, and soon this package was being extended to other hardware systems.

The **OpenGL library** is specifically designed for efficient processing of three-dimensional applications, but it can also handle two-dimensional scene descriptions as a special case of three dimensions where all the z coordinate values are 0.

Function names in the OpenGL basic library (also called the OpenGL core library) are prefixed with **gl**, and each component word within a function name has its first letter capitalized.

```
glBegin,   glClear,   glCopyPixels,   glPolygonMode
```

The OpenGL functions also expect specific **data types**. To indicate a specific data type, OpenGL uses special built-in, data-type names,

```
GLbyte,  GLshort,  GLint,  GLfloat,  GLdouble,  GLboolean
```

There are a number of associated libraries for handling special operations. **The OpenGL Utility (GLU)** provides routines for setting up viewing and projection matrices, describing complex objects with line and polygon approximations, displaying quadrics and B-splines.

Every OpenGL implementation includes the **GLU** library, and all GLU function names start with the prefix **glu**. There is also an object oriented toolkit based on OpenGL, called Open Inventor, which provides routines and predefined object shapes for interactive three-dimensional applications.

There are several **window-system libraries** that support OpenGL functions for a variety of machines. The OpenGL Extension to the X Window System (GLX) provides a set of routines that are prefixed with the letters **glX**.

The **OpenGL Utility Toolkit (GLUT)** provides a library of **functions for interacting with any screen-windowing system**. The GLUT library functions are prefixed with **glut**. This library also contains methods for describing and rendering quadric curves and surfaces.

In all of our graphics programs, we will need to **include the header** file for the OpenGL core library. For most applications we will also need GLU, and on many systems we will need to include the header file for the window system.

```
#include <windows.h>
#include <GL/gl.h>
#include <GL/glu.h>
```

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

Display window is to be created on the screen with a given caption for the title bar. This is accomplished with the function

```
glutCreateWindow ("An Example OpenGL Program");
```

We use the **glutInitWindowPosition** function to give an initial location for the upperleft corner of the display window.

```
glutInitWindowPosition (50, 100);
```

The OpenGL and GLU libraries have a relatively simple method of recording errors. When OpenGL detects an error in a call to a base library routine or a GLU routine, it records an error code internally, and the routine which caused the error is ignored (so that it has no effect on either the internal OpenGL state or the contents of the frame buffer).

However, OpenGL only records one error code at a time. Once an error occurs, no other error code will be recorded until your program explicitly queries the OpenGL error state: