

# CMPE 466

# COMPUTER

# GRAPHICS

---

## Chapter 3

## Computer Graphics Software

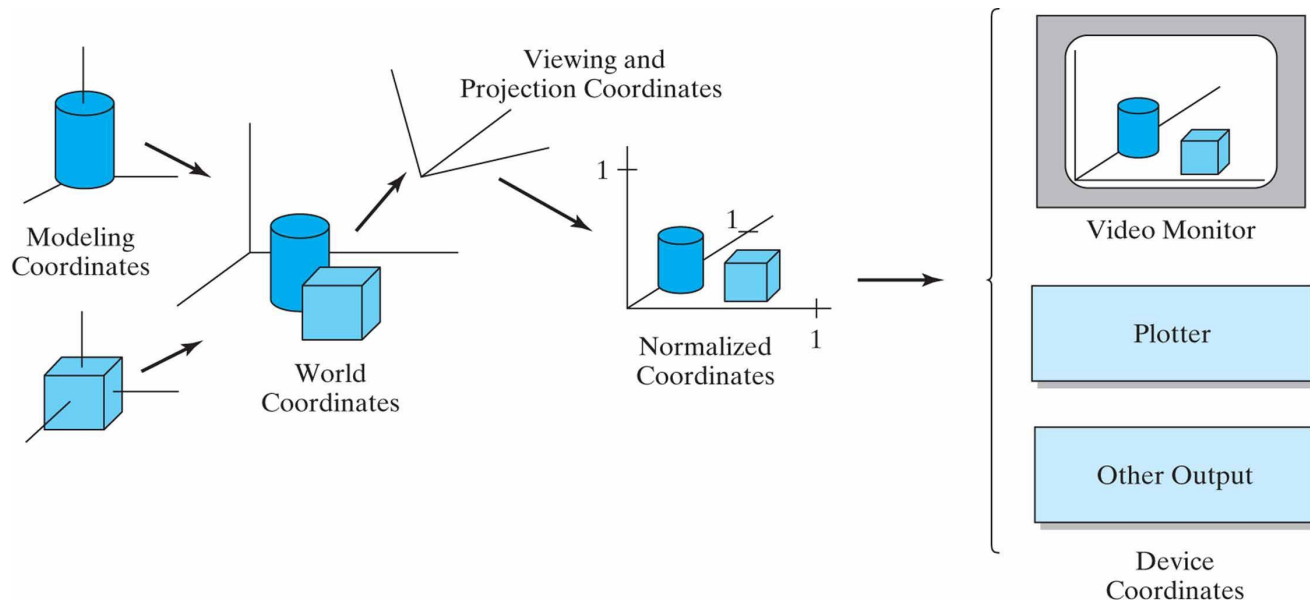
Instructor: D. Arifler

Material based on

- *Computer Graphics with OpenGL*<sup>®</sup>, Fourth Edition by Donald Hearn, M. Pauline Baker, and Warren R. Carithers
- *Fundamentals of Computer Graphics*, Third Edition by Peter Shirley and Steve Marschner

# Coordinate representations

**Figure 3-1** 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. Next, world-coordinate specifications are transformed through the viewing pipeline to viewing and projection coordinates and then to normalized coordinates. At the final step, individual device drivers transfer the normalized-coordinate representation of the scene to the output devices for display.



Right-handed vs. left-handed coordinate reference frame

# Graphics functions

- **Output primitives**: plot character strings, points, straight lines, curved lines, polygons, etc.
- **Attributes**: set properties of output primitives such as color specifications, line styles, fill patterns, etc.
- **Geometric transformations**: change size, position, orientation of an object
- **Viewing transformations**: select a view of the scene, type of projection to be used, location on video monitor where the view is to be displayed
- **Input functions**: control and process the data flow from interactive devices
- **Control operations**: house-keeping tasks such as clearing a screen display area

# Software standards

- Primary goal: Portability
- Graphical Kernel System (GKS)
- Programmer's Hierarchical Interactive Graphics System (PHIGS) and PHIGS+
- Graphics Library (GL)
- OpenGL (hardware-independent version of GL)

# Language binding

- A language binding is defined for a particular high-level programming language
- This binding gives the syntax for accessing various graphics functions
- OpenGL bindings for C and C++ are the same
- OpenGL bindings are also available for Java and Python

# Other graphics packages

- Open Inventor
- Virtual Reality Modeling Language (VRML)
- Java 2D
- Java 3D
- RenderMan Interface
- Libraries for Mathematica, MATLAB, and Maple

# OpenGL

- OpenGL basic (core) library
  - Function names start with “gl”
- OpenGL Utility (GLU) library
  - Constants and data type names begin with “GL”
  - Function names start with “glu”
- Window management operations are device-dependent
  - OpenGL extensions to the X Windows System (GLX)
  - Apple GL (AGL)
  - Windows GL (WGL)
- OpenGL Utility Toolkit (GLUT) library
  - Library of functions for interacting with any windowing system
  - Functions are prefixed with “glut”

# OpenGL header files

- Windows

  - `#include <windows.h>`

  - `#include <GL/gl.h>`

  - `#include <GL/glu.h>`

    - or simply

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

    - GLUT ensures that `gl.h` and `glu.h` are included

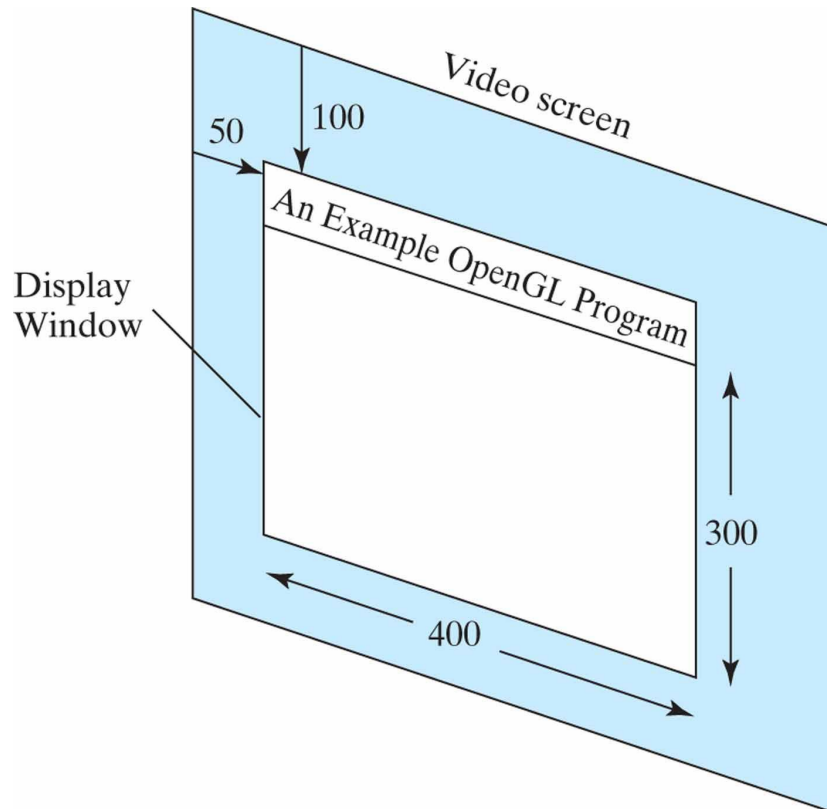
- Apple OS X

  - `#include <GLUT/glut.h>`

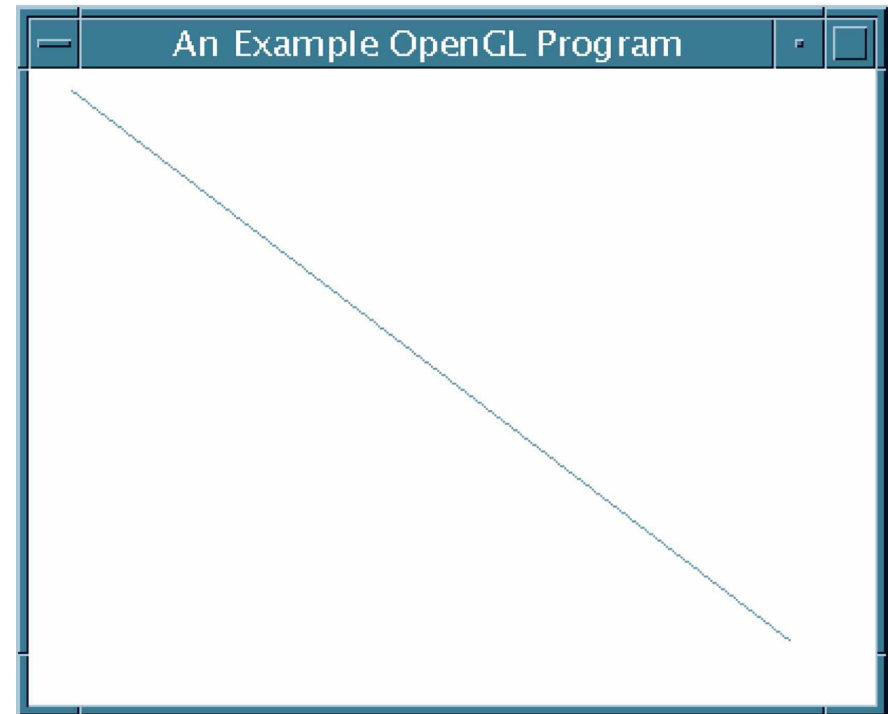


# An example program

**Figure 3-2** A 400 by 300 display window at position (50, 100) relative to the top-left corner of the video display.



**Figure 3-3** The display window and line segment produced by the example program.



# An example program

```
#include <GL/glut.h>          // (or others, depending on the system in use)

void init (void)
{
    glClearColor (1.0, 1.0, 1.0, 0.0); // Set display-window color to white.

    glMatrixMode (GL_PROJECTION);      // Set projection parameters.
    gluOrtho2D (0.0, 200.0, 0.0, 150.0);
}

void lineSegment (void)
{
    glClear (GL_COLOR_BUFFER_BIT); // Clear display window.

    glColor3f (0.0, 0.4, 0.2);      // Set line segment color to green.
    glBegin (GL_LINES);
        glVertex2i (180, 15);      // Specify line-segment geometry.
        glVertex2i (10, 145);
    glEnd ( );

    glFlush ( ); // Process all OpenGL routines as quickly as possible.
}
```

