EC-301 Computer Graphics

Lecture Slides- Wk5 & 6:

- Overview of Graphics Systems (Cont.)
- Geometric Transformations
- Introduction to OpenGL (self-study)

Overview of Graphics Systems (Cont.)

- **Hard-Copy Devices**

  plotters and printers, quality depends on dot size and number of dots per inch

  - Plotters
    - 2D moving pen with stationary paper
    - 1D pen and 1D moving paper
  - Printers
    - Impact devices
      - Inked ribbon
    - Non impact devices
      - Laser, ink-jet, etc

Overview of Graphics Systems (Cont.)

- **Graphics Networks**

  resources (processors, printers, plotters and data files) distributed on a network and shared by multiple users, server & client

Overview of Graphics Systems (Cont.)

- **Graphics on the Internet**

  computers communicate using TCP/IP, www provides hypertext system, uniform resource locator (url), http, ftp sites, hypertext markup language (HTML), web browsers, search engines

Overview of Graphics Systems (Cont.)

- **Graphics Software**
  - Special purpose
    - For non programmers
    - CAD, painting, animation
  - General programming packages
    - C, C++, Java, …
    - Library of graphics functions (OpenGL)
      - Picture components (lines, shapes, …)
      - Transformations (color, texture, shading, rotating, …)

Overview of Graphics Systems (Cont.)

- **Graphics Software (Cont.)**

  coordinate representations—

  - To generate a picture using a programming package, we first need to give the geometric descriptions of the objects – location and shape

  - For example, a box is specified by the positions of its corners, a sphere is specified by its center position and radius

Overview of Graphics Systems (Cont.)

- **Graphics Software (Cont.)**
  coordinate representations—

  Cartesian coordinates

  - Modeling coordinates (local, master)
  in the reference frame of each object

  - World coordinates
  in the scene reference frame

  - Viewing coordinates
  view we want of a scene

  - Normalized coordinates
  normalized between –1 and 1 or 0 and 1

  - Device coordinates (screen coordinates)

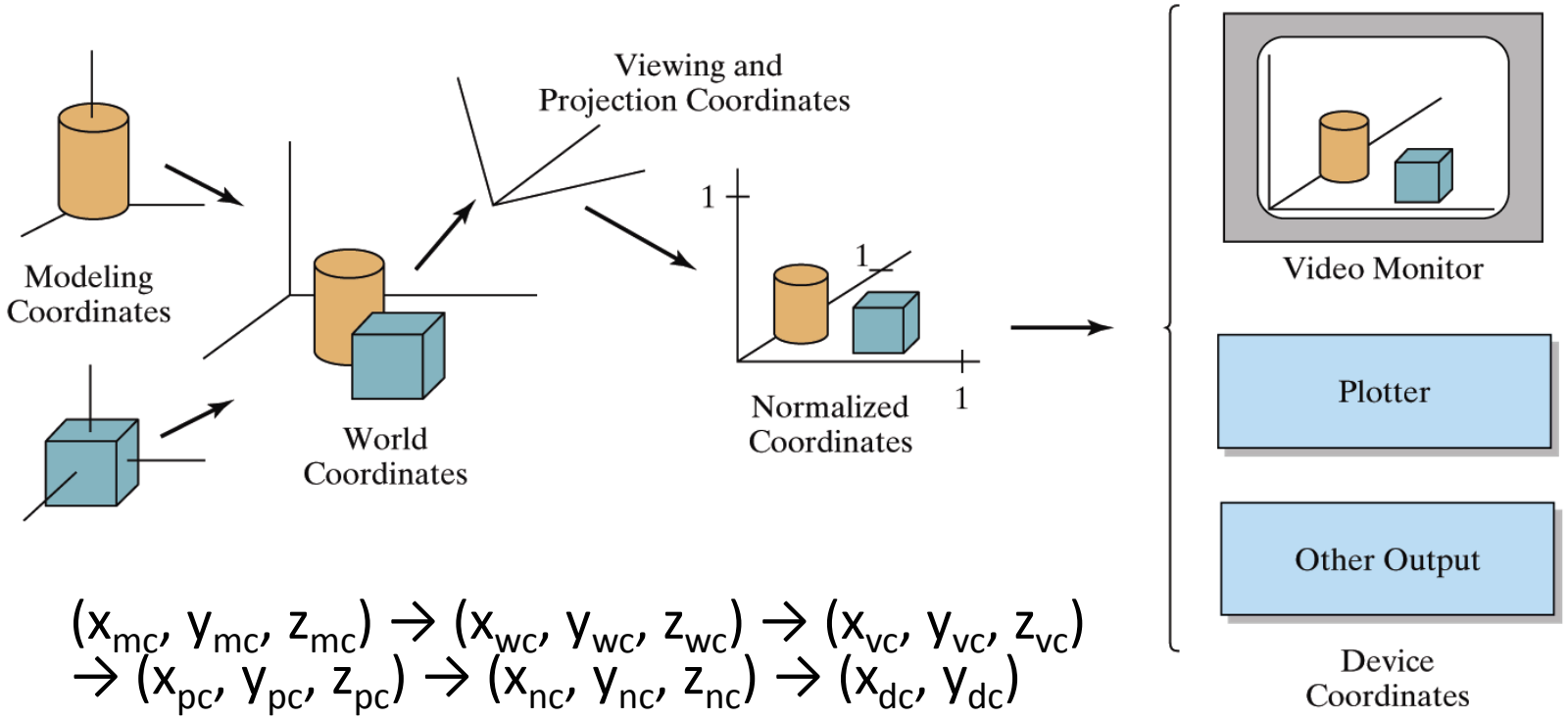


Figure 2-60

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-coordinaterepresentation of the scene to the output devices for display. (H&B Book, 3rd Ed)

Overview of Graphics Systems (Cont.)

- **Graphics Software (Cont.)**

  Graphics Functions—

  Graphics Output Primitives

  – Character strings, lines, filled color areas (usually polygons)

  – Basic tools for constructing pictures

  Attributes

  – Color, line style, text style, area filling patterns

  Geometric Transformations

  – Change size, position or orientation of an object

Overview of Graphics Systems (Cont.)

- **Graphics Software (Cont.)**

  Graphics Functions—

  Viewing Transformations

  – Select a view of the scene, type of projection to be used, location of the view on the video monitor

  Input functions

  – Control and process data flow from interactive devices

  Control operations

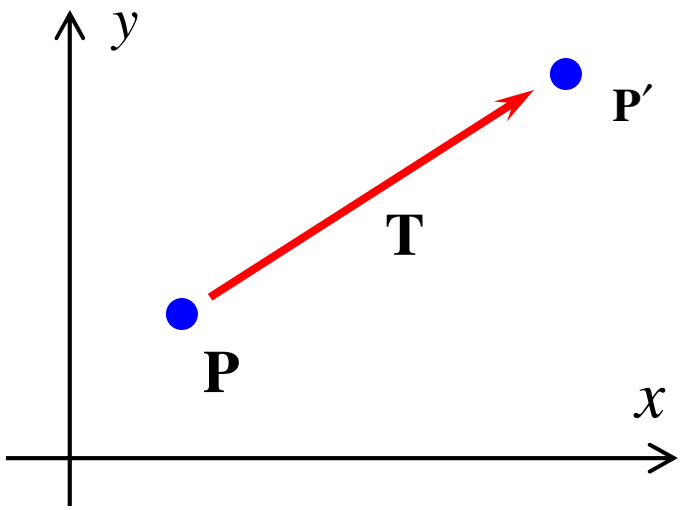
  – Clear the screen display area, initialize parameters

Geometric Transformations

- Operations that are applied to the geometric description of an object to change its position, orientation, or size
  - an architect creates a layout by arranging the orientation and size of the component parts of a design
  - a computer animator develops a video sequence by moving the “camera” position or the objects in a scene along specified paths

2D Translation

- translation on a single coordinate point is performed by adding offsets to its coordinates so as to generate a new coordinate position

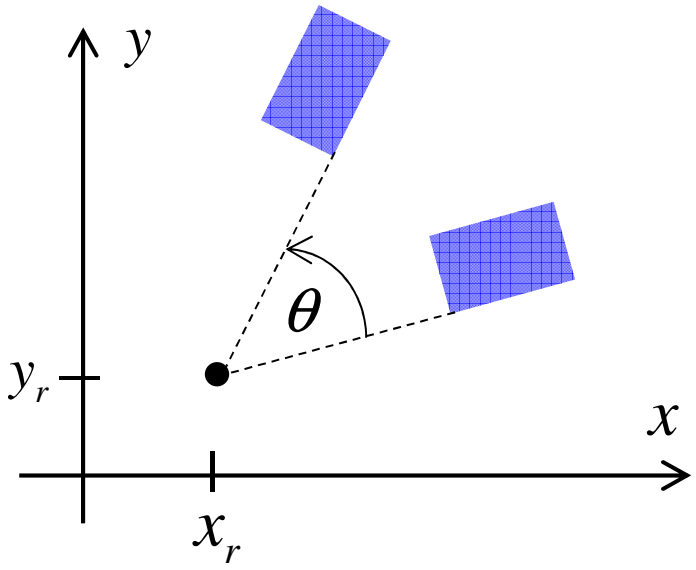


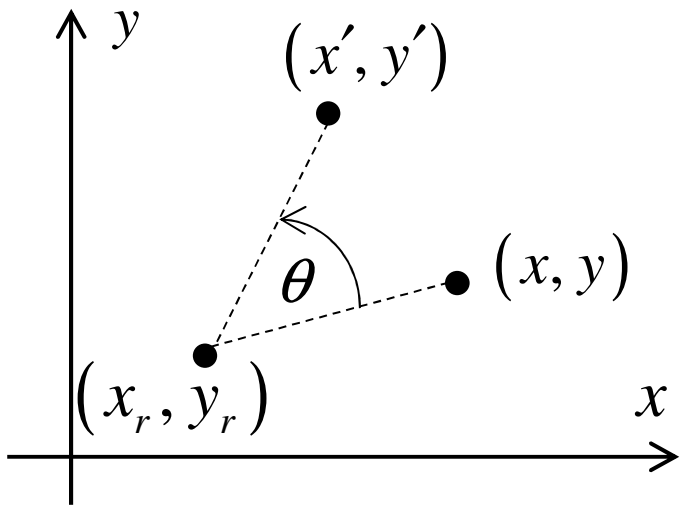
$$x' = x + t_x, \quad y' = y + t_y$$

$$\mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{P}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{P} + \mathbf{T}$$

2D Rotation (ccw)





Rotation in angle θ about a pivot (rotation) point (x_r, y_r).

$$P' = T_r . R_\theta . T_{-r}(P)$$

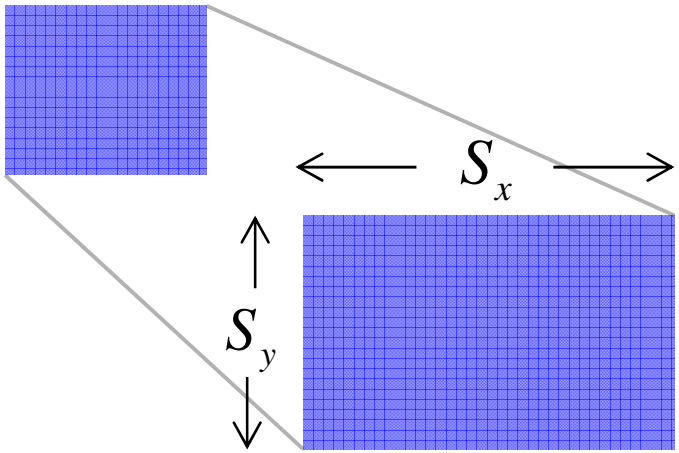
$$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta$$

$$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta$$

$$\mathbf{P}' = \mathbf{P}_r + \mathbf{R} \cdot (\mathbf{P} - \mathbf{P}_r)$$

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

2D Scaling



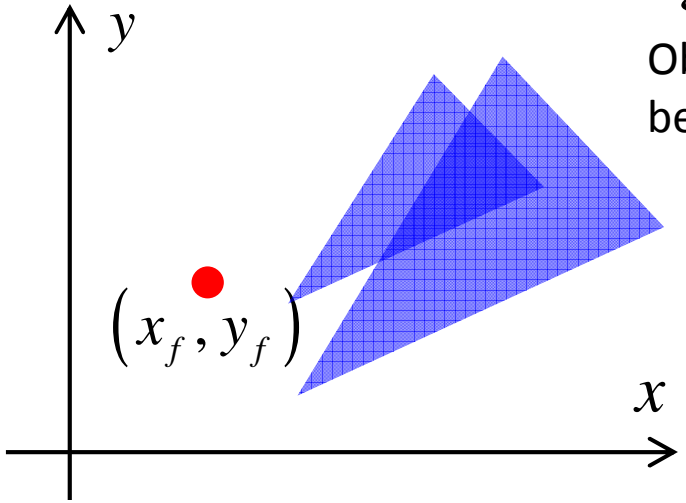
$$x' = x \cdot s_x, \quad y' = y \cdot s_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{S} \cdot \mathbf{P}$$

Scaling about a fixed point (x_f, y_f)

Objects are resized by scaling the distances between object points and the fixed points



$$x' = x \cdot s_x + x_f (1 - s_x)$$

$$y' = y \cdot s_y + y_f (1 - s_y)$$

$$\mathbf{P}' = \mathbf{P} \cdot \mathbf{S} + \mathbf{P}_f \cdot (\mathbf{1} - \mathbf{S})$$

Homogeneous Coordinates

- General matrix form representation of 2D transformations for a point **P:** $\mathbf{P' = M_1 \cdot P + M_2}$
  - $\mathbf{M_1}$: 2 × 2 rotation or scaling matrix. $\mathbf{M_2}$: 2 × 1 displacement vector
  - To perform sequence of transformations (S→R→T): requires one step for each operation
  - How can we combine all transformation steps?
- Make transformations linear with **Homogeneous Coordinates**
  - $(x, y) \rightarrow (x, y, 1)$. Transformations turn into 3 × 3 matrices

2D Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{P}' = \mathbf{T}\left(t_x, t_y\right) \cdot \mathbf{P}$$

2D Rotation (ccw)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{P}' = \mathbf{R}\left(\theta\right) \cdot \mathbf{P}$$

2D Scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{P}' = \mathbf{S}\left(S_x, S_y\right) \cdot \mathbf{P}$$

Inverse transformations:

$$\mathbf{T}^{-1}=\begin{bmatrix}1 & 0 & -t_x\\ 0 & 1 & -t_y\\ 0 & 0 & 1\end{bmatrix},\ \mathbf{R}^{-1}=\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix},\ \mathbf{S}^{-1}=\begin{bmatrix}1/S_x & 0 & 0\\ 0 & 1/S_y & 0\\ 0 & 0 & 1\end{bmatrix}$$

Composite transformations: $\mathbf{P}'=\mathbf{M}_2\left(\mathbf{M}_1\cdot\mathbf{P}\right)=\left(\mathbf{M}_2\cdot\mathbf{M}_1\right)\cdot\mathbf{P}=\mathbf{M}\cdot\mathbf{P}$

$$\mathbf{P}'=\mathbf{T}\left(t_{2x},t_{2y}\right)\left\{\mathbf{T}\left(t_{1x},t_{1y}\right)\cdot\mathbf{P}\right\}=\left\{\mathbf{T}\left(t_{2x},t_{2y}\right)\cdot\mathbf{T}\left(t_{1x},t_{1y}\right)\right\}\cdot\mathbf{P}$$

Composite translations:

$$\begin{bmatrix}1 & 0 & t_{2x}\\ 0 & 1 & t_{2y}\\ 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}1 & 0 & t_{1x}\\ 0 & 1 & t_{1y}\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & t_{1x}+t_{2x}\\ 0 & 1 & t_{1y}+t_{2y}\\ 0 & 0 & 1\end{bmatrix}$$

$$\mathbf{T}\left(t_{2x},t_{2y}\right)\cdot\mathbf{T}\left(t_{1x},t_{1y}\right)=\mathbf{T}\left(t_{1x}+t_{2x},t_{1y}+t_{2y}\right)$$

$$\mathbf{P}' = \mathbf{R}(\theta_2)\{\mathbf{R}(\theta_1)\cdot\mathbf{P}\} = \{\mathbf{R}(\theta_2)\cdot\mathbf{R}(\theta_1)\}\cdot\mathbf{P}$$

Composite Rotations:

$$\mathbf{R}(\theta_2)\cdot\mathbf{R}(\theta_1) = \mathbf{R}(\theta_1+\theta_2)$$

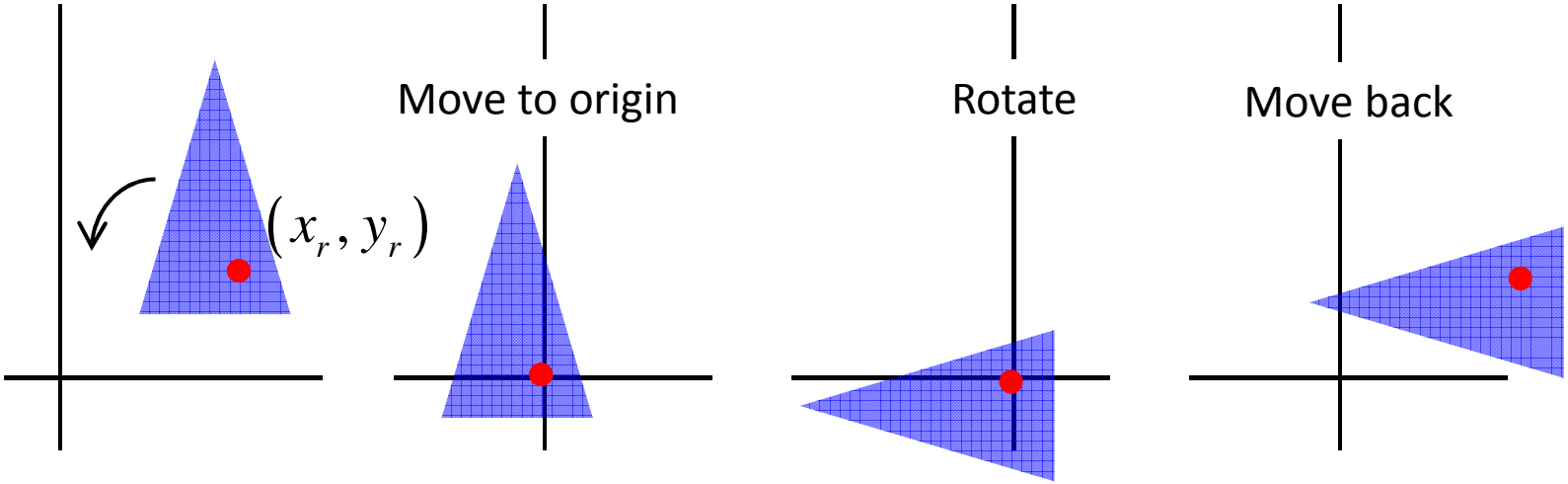
$$\mathbf{P}' = \mathbf{R}(\theta_1+\theta_2)\cdot\mathbf{P}$$

$$\begin{bmatrix} S_{2x} & 0 & 0 \\ 0 & S_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix}\cdot\begin{bmatrix} S_{1x} & 0 & 0 \\ 0 & S_{1y} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_{1x}\cdot S_{2x} & 0 & 0 \\ 0 & S_{1y}\cdot S_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Composite Scaling:

$$\mathbf{S}(S_{2x}, S_{2y})\cdot\mathbf{S}(S_{1x}, S_{1y}) = \mathbf{S}(S_{1x}\cdot S_{2x}, S_{1y}\cdot S_{2y})$$

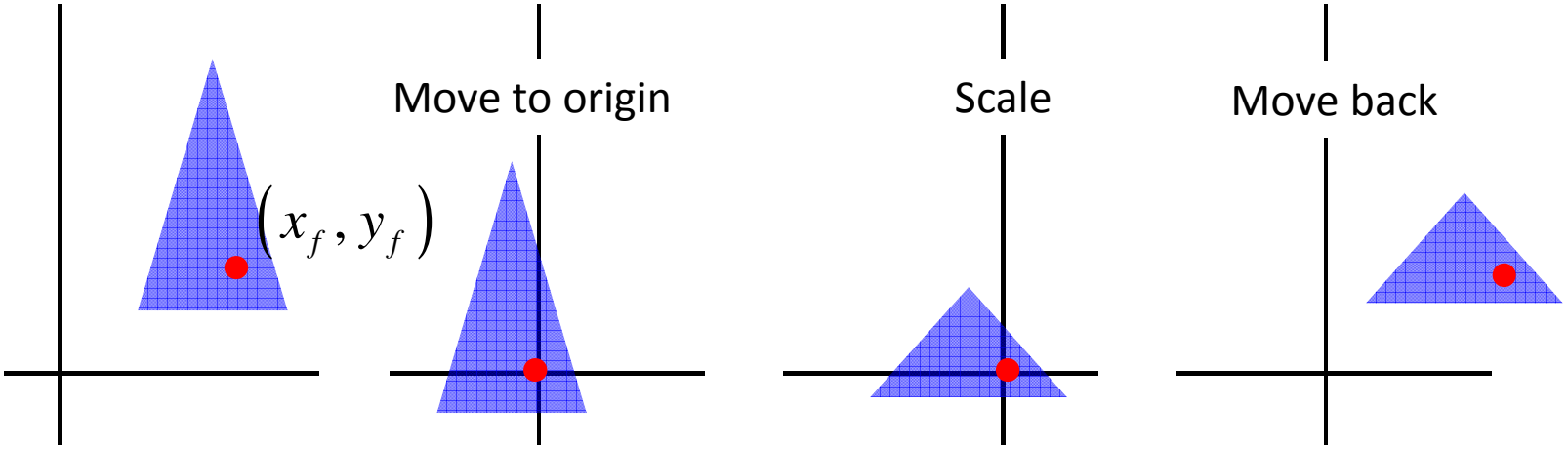
General 2D Pivot-Point Rotation



$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix} =$$

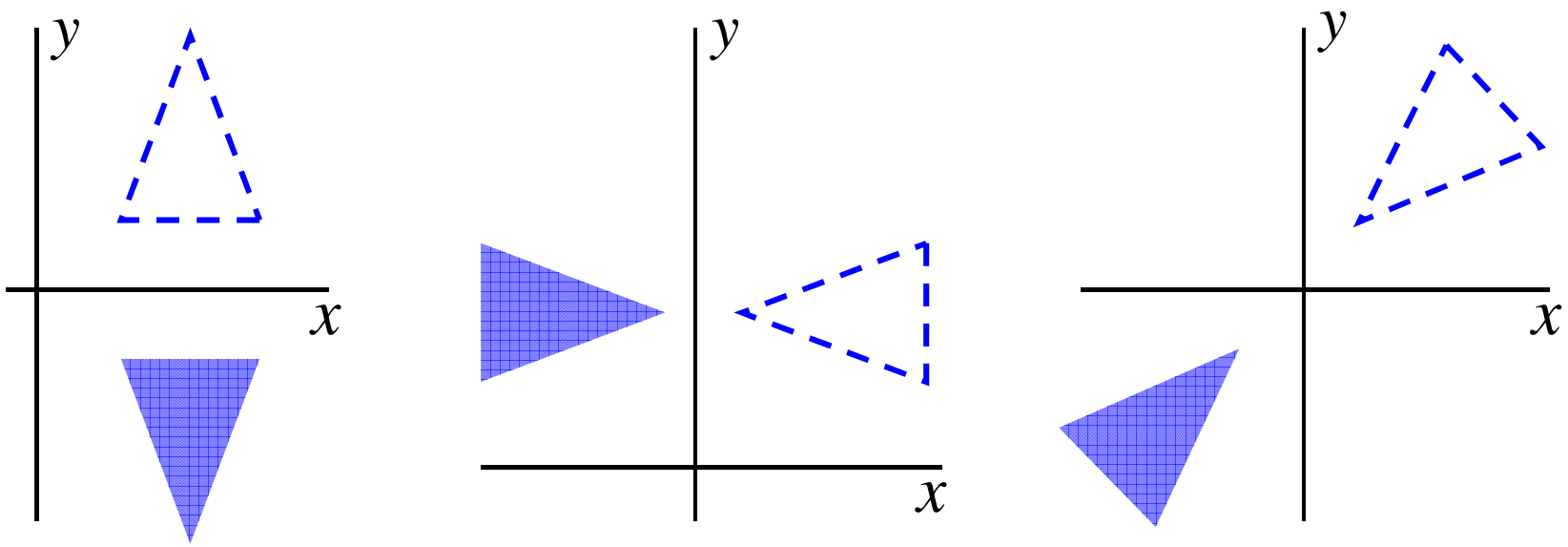
$$\begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta)+y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta)-x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

General 2D Fixed-Point Scaling

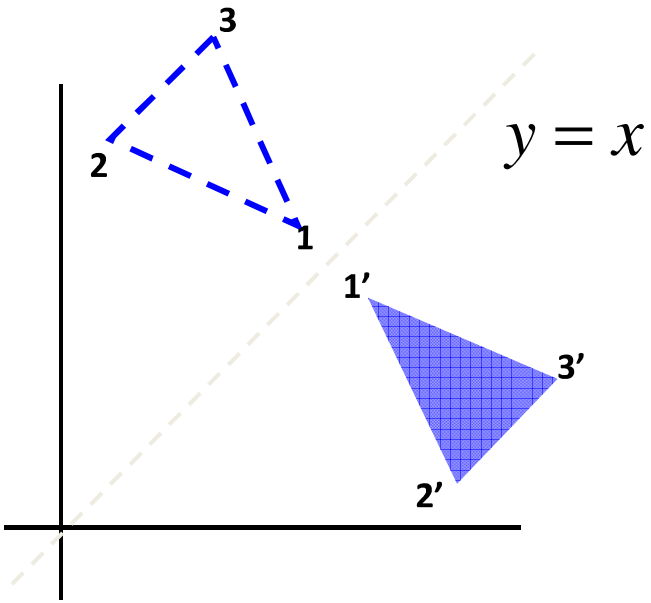


$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & x_f(1-S_x) \\ 0 & S_y & y_f(1-S_y) \\ 0 & 0 & 1 \end{bmatrix}$$

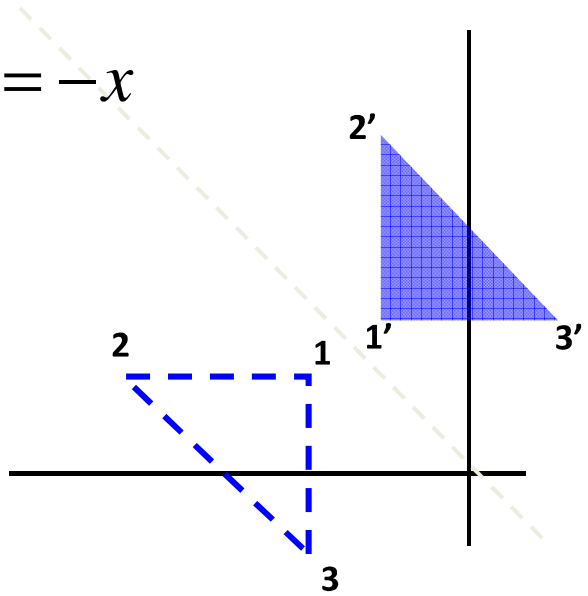
2D Reflections



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$y = x$



$y = -x$

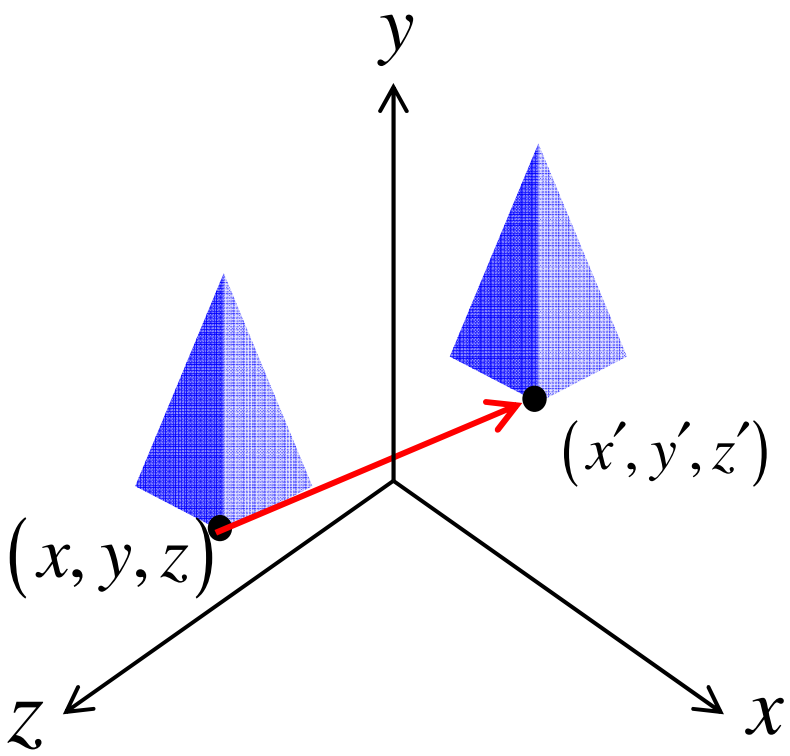
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3D Transformations

Very similar to 2D. Using 4x4 matrices rather than 3x3.

3D Translation:



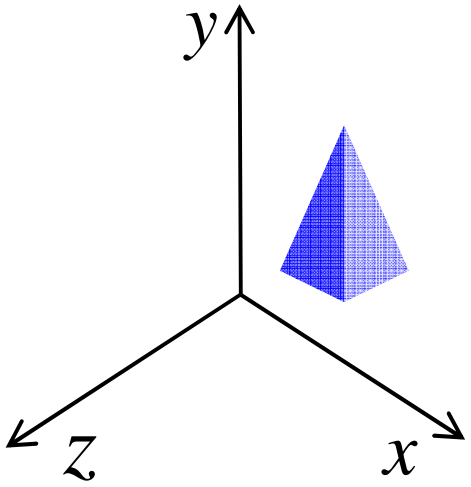
$$x' = x + t_x$$

$$y' = y + t_y$$

$$z' = z + t_z$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

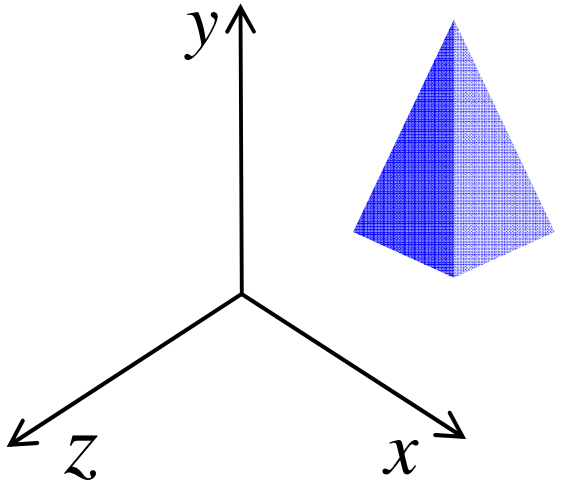
3D Scaling:



$$x' = x \cdot S_x$$

$$y' = y \cdot S_y$$

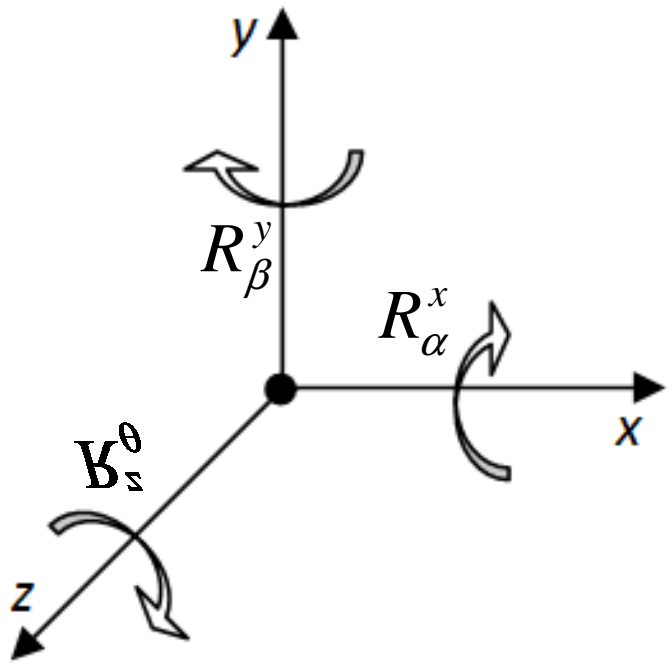
$$z' = x \cdot S_z$$



Enlarging object also moves it from origin

$$\mathbf{P}' = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{S} \cdot \mathbf{P}$$

3D Rotation (CW):



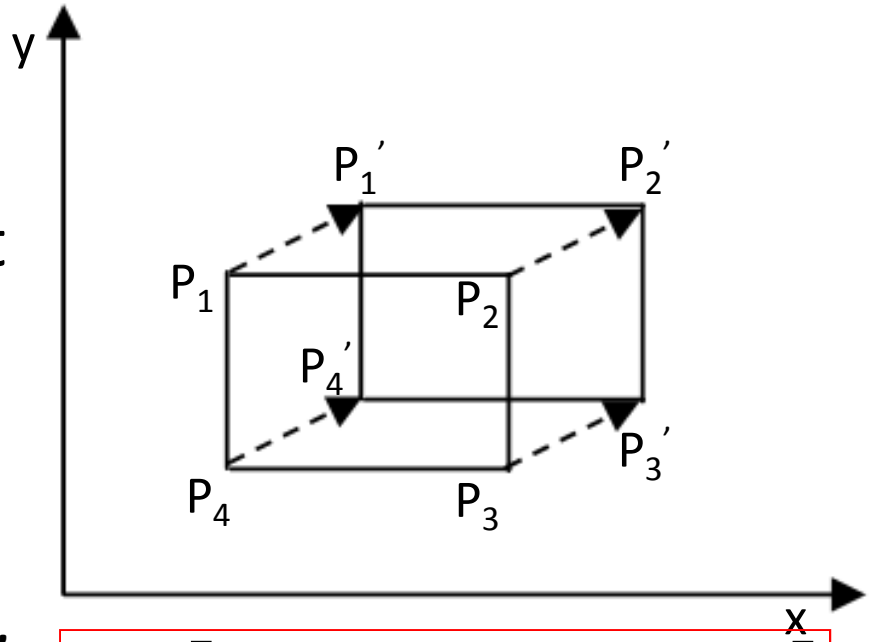
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathrm{Cos}\alpha & \mathrm{Sin}\alpha & 0 \\ 0 & -\mathrm{Sin}\alpha & \mathrm{Cos}\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \textit{around x-axis}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{Cos}\beta & 0 & -\mathrm{Sin}\beta & 0 \\ 0 & 1 & 0 & 0 \\ \mathrm{Sin}\beta & 0 & \mathrm{Cos}\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \textit{around y-axis}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{Cos}\theta & \mathrm{Sin}\theta & 0 & 0 \\ -\mathrm{Sin}\theta & \mathrm{Cos}\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \textit{around z-axis}$$

Transformation for Set of Points

- For a set of *m* points, construct a matrix **V** of dimension 4 x *m,* such that each column represents a point
  - Transformation: $\mathbf{V}' = \mathbf{AV}$
  - i^{th} column, j^{th} element of $\mathbf{V}'$ is transformed point corresponding to $\mathbf{V}_{i,j}$



$$V = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_m \\ y_1 & y_2 & y_3 & \dots & y_m \\ z_1 & z_2 & z_3 & \dots & z_m \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$

Practice Questions

Q1. A unit cube with vertices at (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0) and (1,1,1) is scaled using the scale factors S_x=2, S_y=3 and S_z=4. What are vertices of the transformed figure.

$$\mathbf{V' = SV}$$

$$V = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_m \\ y_1 & y_2 & y_3 & \dots & y_m \\ z_1 & z_2 & z_3 & \dots & z_m \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$

Practice Questions

Q2 Consider a 3D point $[2\ 1\ 2]^T$. What would be coordinates of the point after applying the following composite transformation:
(i) CCW rotation of 90 degrees around the x-axis,
(ii) translation by *dx = -2, dy = 1, dz = 1,* and
(iii) scaling by *sx = 1, sy = 2 and sz = 0.5.*

$$Rotation\ \ Matrix\ \ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos\,\alpha & -Sin\,\alpha & 0 \\ 0 & Sin\,\alpha & Cos\,\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Practice Questions

Q3 What is the rotation matrix for an object rotation of 30 deg around the z-axis, followed by 60 deg around the x-axis, and followed by a rotation of 90 deg around the y-axis. All rotations are counter clockwise.

$$= \begin{bmatrix} Cos\beta & 0 & Sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -Sin\beta & 0 & Cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos\alpha & -Sin\alpha & 0 \\ 0 & Sin\alpha & Cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Cos\theta & -Sin\theta & 0 & 0 \\ Sin\theta & Cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Introduction to OpenGL (self-study)

- Silicon Graphics Inc. (SGI) workstations were first that came with a set of routines called GL
- GL very soon became a de facto graphics standard
- Its routines were designed for fast, real time rendering
- Became hardware independent in early 1990s and named as OpenGL
- OpenGL is specifically designed for efficient processing of 3D applications
- Can also handle 2D scenes descriptions as a special case of 3D where z-coordinate is 0.

OpenGL

- Graphics rendering API (Application Programmer's Interface)
  - A software interface to graphics hardware
  - Generation of high-quality color images composed of geometric and image primitives
  - Window system independent
  - Operating system independent
  - Close to the hardware
  - Algorithms and implementations are important

OpenGL and the Windowing System

- OpenGL is concerned only with rendering
  - It is window system independent
  - No input functions
- OpenGL must interact with the underlying OS and windowing system
  - Need minimal interface which may be system dependent
    - Done through additional libraries: AGL (Apple), GLX (X-Window), WGL(Windows)

GLU and GLUT

- GLU (OpenGL Utility Library)
  - Part of OpenGL
  - Provides B-splines, surface-rendering, etc
- GLUT (OpenGL Utility Toolkit)
  - A portable windowing API
  - Not officially part of OpenGL
  - Extremely useful for helping to create images quickly without worrying about the underlying windowing system being used

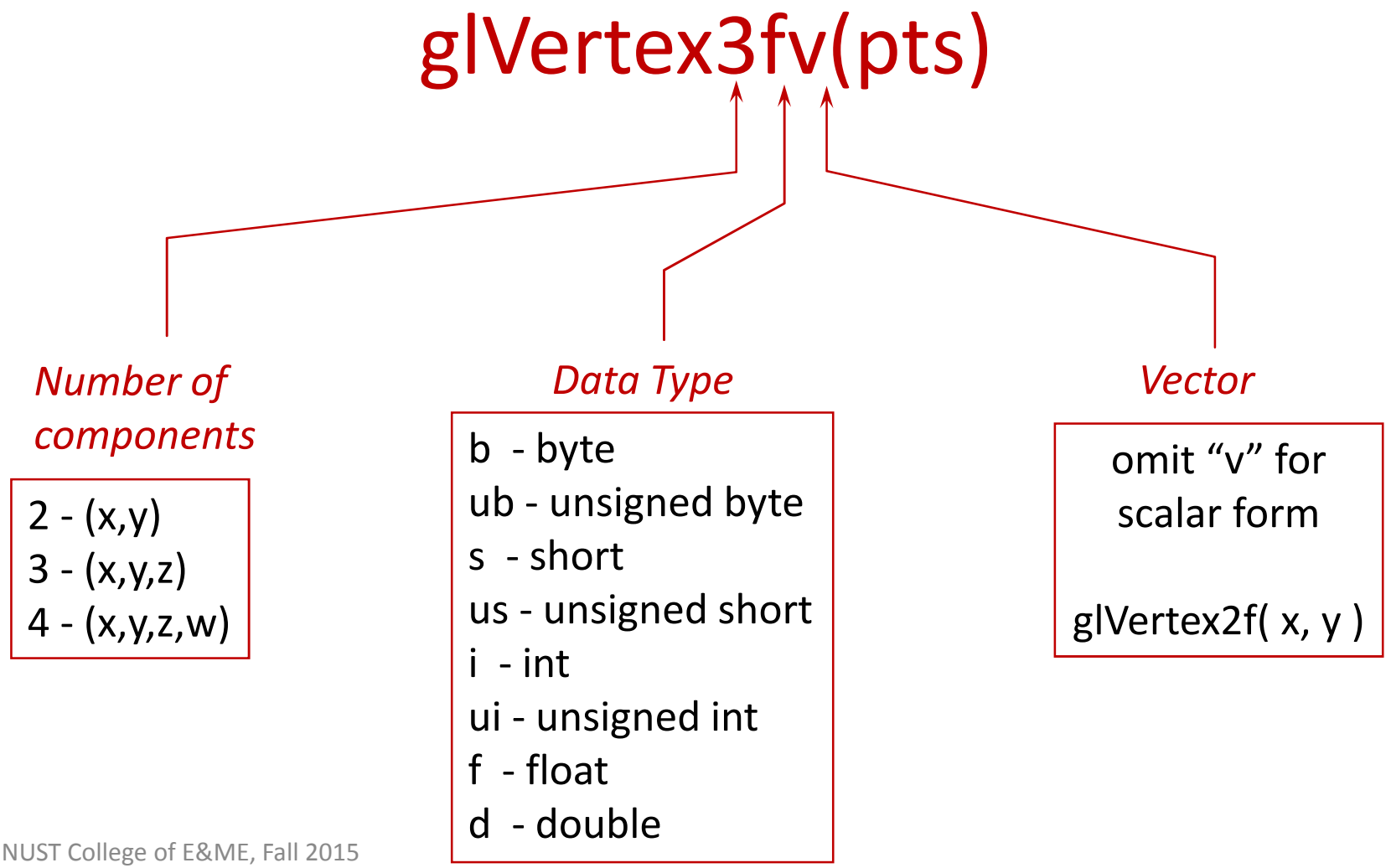
    http://reality.sgi.com/opengl/glut3/glut3.html

    www.opengl.org/resources/libraries/glut/

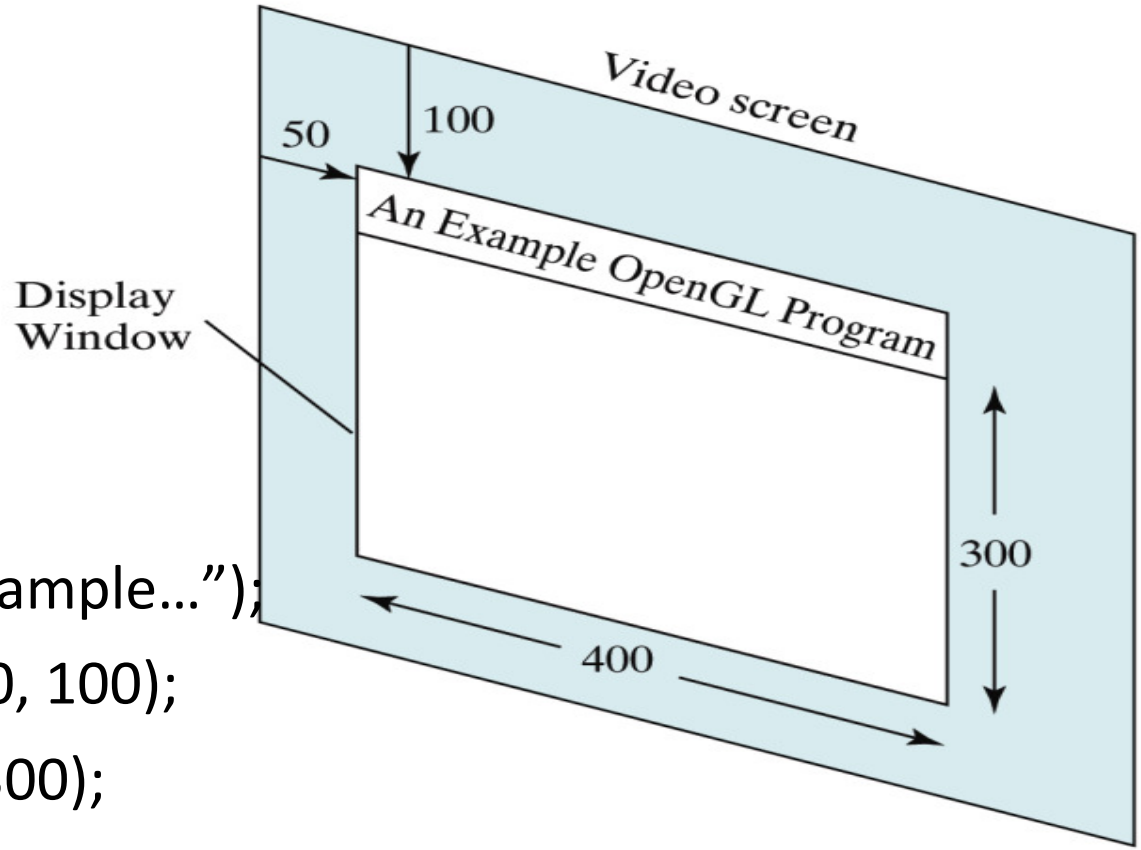
Basic OpenGL Syntax

- Function names are prefixed with gl for core library, glu for GLU, glut for GLUT library
  - glBegin, glClear, gluOrtho2D,glutInit
- Constants
  - GL_2D, GL_RGB, GLUT_SINGLE
- Data types
  - GLbyte, GLshort, GLint, GLfloat, GLdouble

OpenGL Command Formats



Display Window



glutCreateWindow ("An Example…");

glutInitWindowPosition (50, 100);

glutInitWindowSize (400, 300);

Figure 2-61

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

Sample 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);
}
```

(https://www.opengl.org/resources/libraries/glut/)

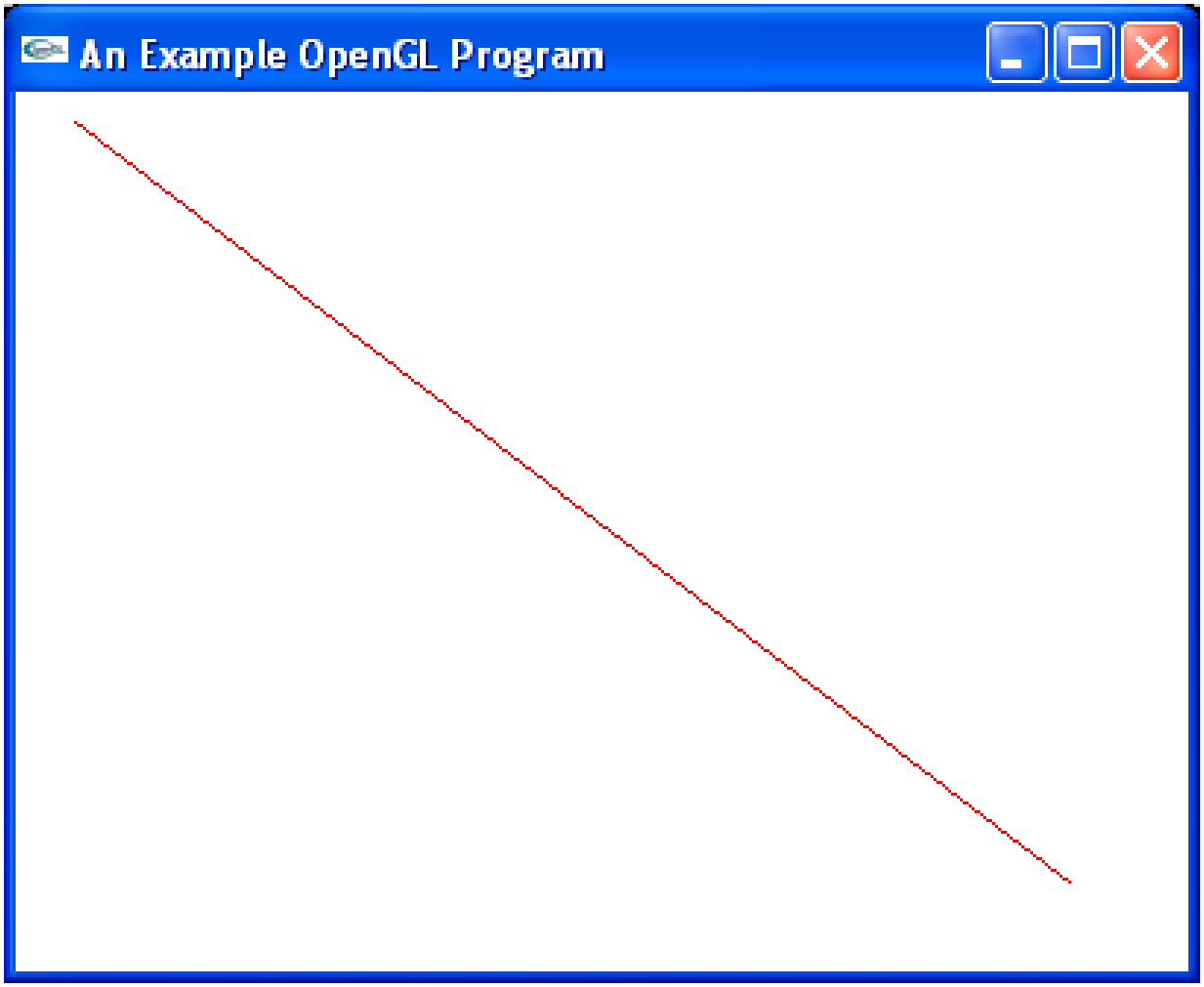
Sample Program

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

    glColor3f (1.0, 0.0, 0.0);
// Set line segment color to red.
    glBegin (GL_LINES);
        glVertex2i (180, 15);
// Specify line-segment geometry.
        glVertex2i (10, 145);
    glEnd ( );

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

Sample Program

```
void main (int argc, char** argv)
{
    glutInit (&argc, argv); // Initialize GLUT.
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  // Set display mode.
    glutInitWindowPosition (50, 100);
// Set top-left display-window position.
    glutInitWindowSize (400, 300);
// Set display-window width and height.
    glutCreateWindow ("An Example OpenGL
  Program"); // Create display window.
    init ( );
// Execute initialization procedure.
    glutDisplayFunc (lineSegment);
// Send graphics to display window.
    glutMainLoop ( );
// Display everything and wait.
}
```

Output



Suggested Reading:

Ch 2 & Ch 5— Computer Graphics with OpenGL, 3rd Ed, Donald Hearn & M. Pauline Baker, Prentice Hall, 2004.

Next Week:

Graphics Output Primitives