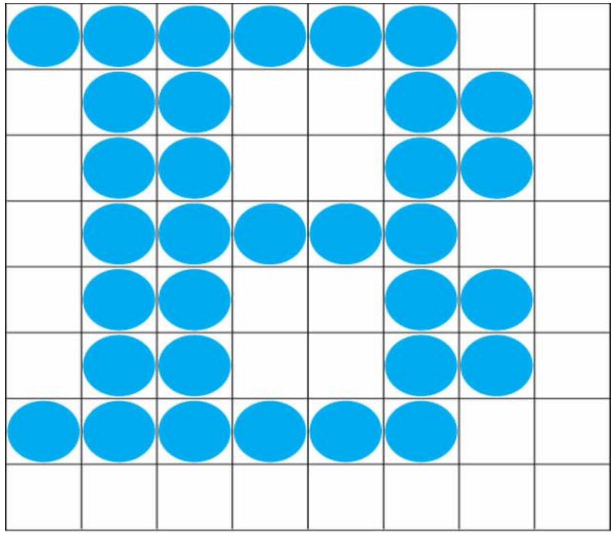
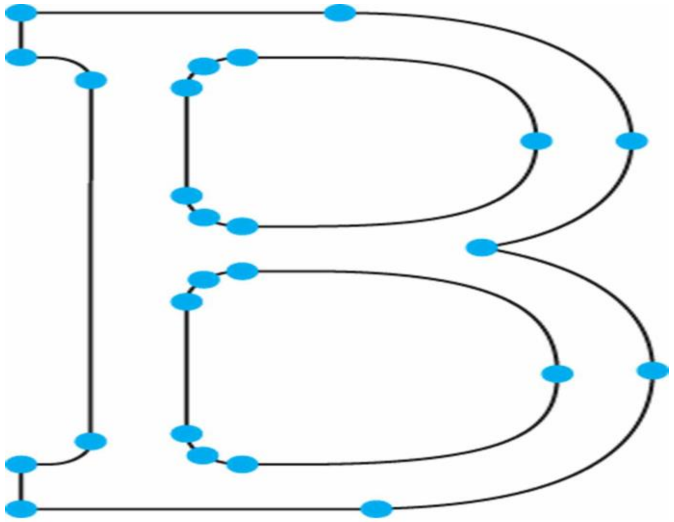
2D & 3D Graphics Summary

M.K.

Raster vs. Vector Graphics

- Raster graphics produced by defining a pix map that creates a picture by setting the color of all pixels in the display window. Vector Graphics on the other hand define graphics using mathematical formulae that only color specific pixels





Graphics Implementation Algorithms

- Line Drawing
- Circle Drawing
- Ellipse
- Other Curves
- Fill Algorithms
- Anti Aliasing
- OpenGL Attributes and Primitives for these

Line Drawing Algorithms

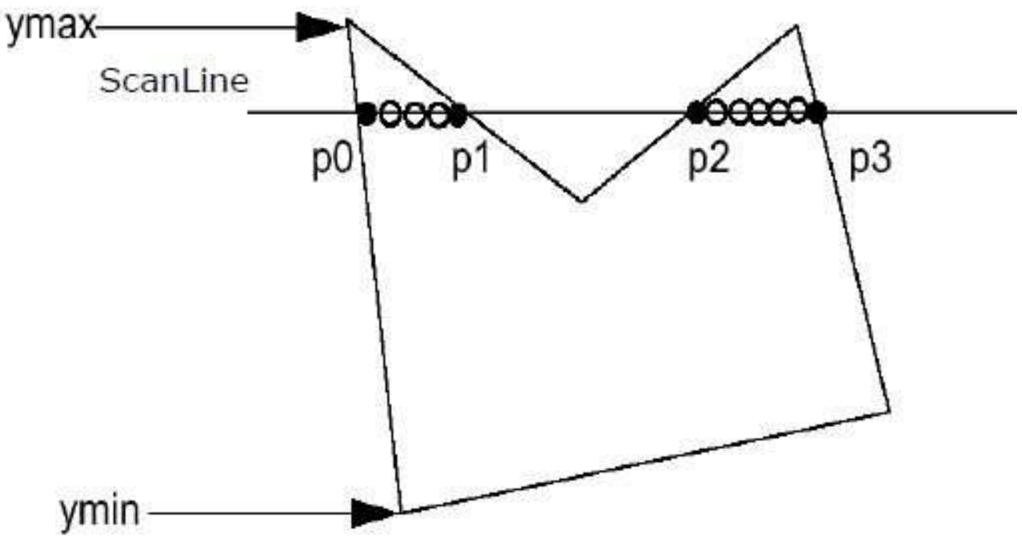
- Line Equation – uses line equation to determine pixels along the line
- DDA Algorithm – determines next pixel from previous
- Bresenham’s Line Algorithm

Circle Drawing Algorithms

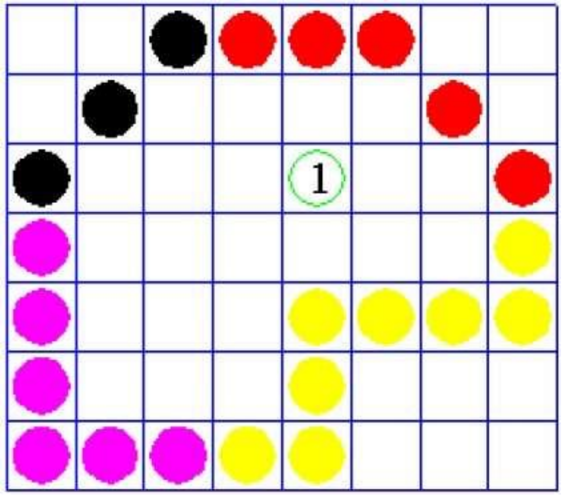
- Bresenham’s Circle Algoritm
- Mid Point Circle Algorithm

Polygon Filling Algorithms

- Scanline Algorithm – as the screen is scanned top to bottom, the pixels between the boundaries are filled



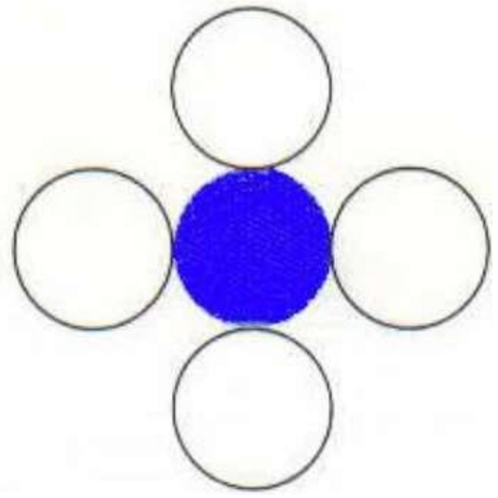
- Flood Fill Algorithm – relies on the color of the pixels and changes their color until no more of a particular color are found



Polygon Filling Algorithms

- Boundary Fill Algorithm – requires different colors for fill area and boundary to work, picks a point in the area and fills it in a line until it finds a different color

- 4 Connected Polygon/8 Connected Polygon – picks a pixel and fills the adjacent 4 pixels and continues until area is filled.



Algorithms Summary

- Inside Outside Test – Reading assignment

Introduction to Graphics Transformations

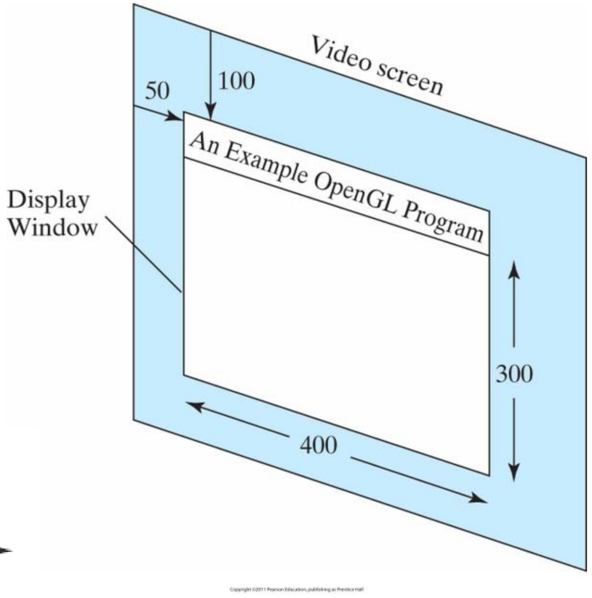
These are operations applied to objects to reposition, orient or resize them and include:-

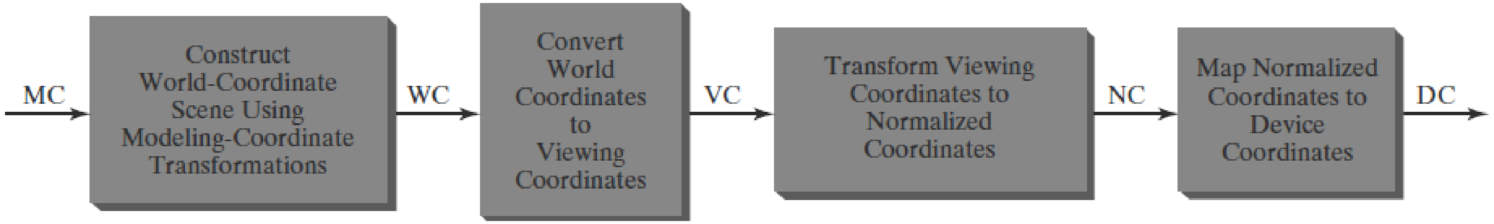
- Translation
- Rotation
- Scaling
- Reflection?
- Shearing?
- Clipping?

Introduction to Graphics Transformations

- Viewing routines convert these world coordinate system descriptions to be displayed on either a raster or vector device viewport that may vary in size to the world coordinate system used to construct the object
- Operations include viewport transformations e.g. scaling and translation, viewing and clipping functions for lines, circles, polygons etc and these are implemented using various algorithms

Viewport and display window:-





OpenGL Viewing Functions

TABLE 8-1

Summary of OpenGL Two-Dimensional Viewing Functions

Function	Description
`gluOrtho2D`	Specifies clipping-window coordinates as parameters for a two-dimensional orthogonal projection.
`glViewport`	Specifies screen-coordinate parameters for a viewport.
`glGetIntegerv`	Uses arguments `GL_VIEWPORT` and `vpArray` to obtain parameters for the currently active viewport.
`glutInit`	Initializes the GLUT library.
`glutInitWindowPosition`	Specifies coordinates for the top-left corner of a display window.
`glutInitWindowSize`	Specifies width and height for a display window.
`glutCreateWindow`	Creates a display window (which is assigned an integer identifier) and specify a display-window title.
`glutInitDisplayMode`	Selects parameters such as buffering and color mode for a display window.
`glClearColor`	Specifies a background RGB color for a display window.
`glClearIndex`	Specifies a background color for a display window using color-index mode.

`glutDestroyWindow`	Specifies an identifier number for a display window that is to be deleted.
`glutSetWindow`	Specifies the identifier number for a display window that is to be the current display window.
`glutPositionWindow`	Resets the screen location for the current display window.
`glutReshapeWindow`	Resets the width and height for the current display window.
`glutFullScreen`	Sets current display window to the size of the video screen.
`glutReshapeFunc`	Specifies a function that is to be invoked when display-window size is changed.
`glutIconifyWindow`	Converts the current display window to an icon.
`glutSetIconTitle`	Specifies a label for a display-window icon.
`glutSetWindowTitle`	Specifies new title for the current display window.
`glutPopWindow`	Moves current display window to the “top”; i.e., in front of
`glutPushWindow`	Moves current display window to the “bottom”; i.e., behind all other windows.
`glutShowWindow`	Returns the current display window to the screen.
`glutCreateSubWindow`	Creates a second-level window within a display window.
`glutSetCursor`	Selects a shape for the screen cursor.
`glutDisplayFunc`	Invokes a function to create a picture within the current display window.
`glutPostRedisplay`	Renews the contents of the current display window.
`glutMainLoop`	Executes the computer-graphics program.
`glutIdleFunc`	Specifies a function to execute when the system is idle.
`glutGet`	Queries the system about a specified state parameter.

Introduction to Graphics Transformations

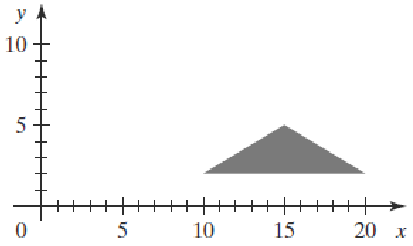
- Zooming can be used to scale up or down a viewport and is defined at pixel level
- Panning – moving the selected zoom area to screen center
- Inking & Scissoring?

2D Transformations: Translation

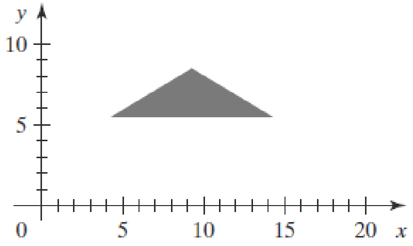
Move object rigidly without deformation from one position to another.

Can simply use a transformation vector that is applied to all points

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



(a)



(b)

2D Transformations: Rotation

- Rotate an object about a point by a certain angle:-

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

where the rotation matrix is

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

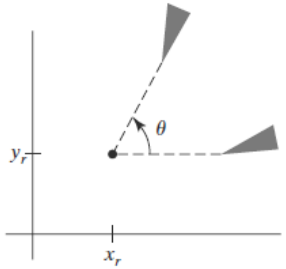


FIGURE 3
Rotation of an object through angle about the pivot point (x_r, y_r).

2D Transformations: Scaling

- Scaling matrix and example:-

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

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

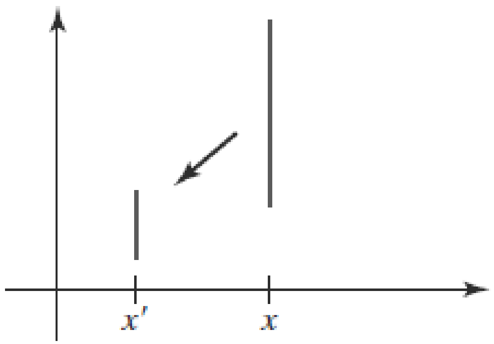


FIGURE 7
A line scaled with Equation 12 using $s_x = s_y = 0.5$ is reduced in size and moved closer to the coordinate origin.

Composite Transformations

- Can do a single or composite transformation using homogenous coordinates and 3x3 matrix

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

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{S}(s_x, s_y) \cdot \mathbf{P}$$

Inverse Transformations

- These produce the same transformations but in the opposite direction and are usually accomplished by negating the factors in the transformation matrices e.g. for translation , rotation and scaling:-

$$\mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R}^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{S}^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Other Transformations: Reflection

- Reflection on the X and y axes or both:-

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

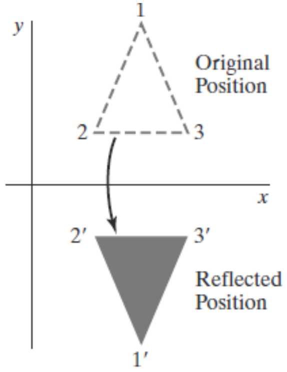


FIGURE 16
Reflection of an object about the x axis.

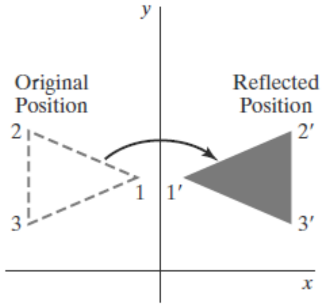


FIGURE 17
Reflection of an object about the y axis.

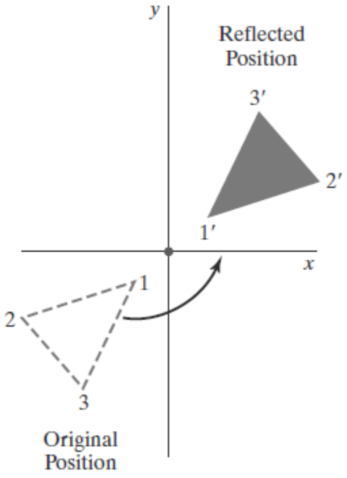
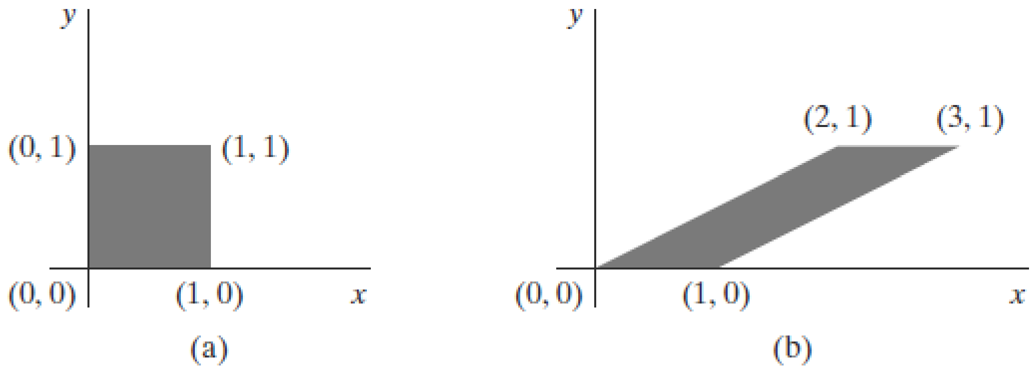


FIGURE 18
Reflection of an object relative to the coordinate origin. This transformation can be accomplished with a rotation in the xy plane about the coordinate origin.

Other Transformations: Shear

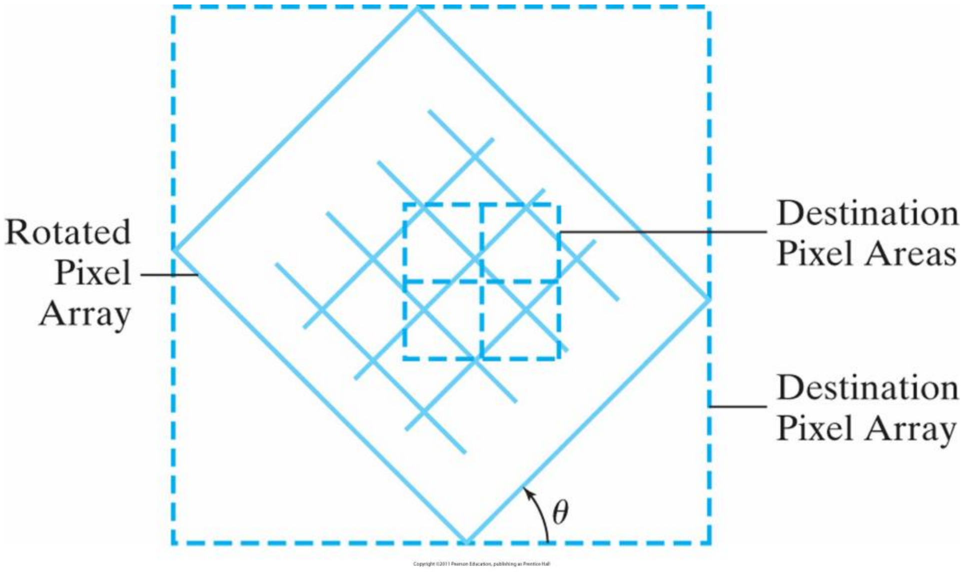
- Shearing matrix:-

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



Raster Graphics Transformations

- These are accomplished by either translating, rotating or scaling an entire array of pixels e.g.



OpenGL Raster Primitive examples among others:-

- glCopyPixels
- glDrawPixels
- glReadPixels
- glPixelZoom

3D Transformations

- Main difference is presence of 3rd dimension z but same transformations apply with same methods but a few unique to 3D come into play

TABLE 7-1

Summary of OpenGL Geometric Transformation Functions

Function	Description
glTranslate*	Specifies translation parameters.
glRotate*	Specifies parameters for rotation about any axis through the origin.
glScale*	Specifies scaling parameters with respect to coordinate origin.
glMatrixMode	Specifies current matrix for geometric-viewing transformations, projection transformations, texture transformations, or color transformations.
glLoadIdentity	Sets current matrix to identity.
glLoadMatrix* (elems);	Sets elements of current matrix.
glMultMatrix* (elems);	Postmultiplies the current matrix by the specified matrix.
glPixelZoom	Specifies two-dimensional scaling parameters for raster operations.