OpenGL Transformations, Start of 3D transformations

CS116A

Chris Pollett

Oct 18, 2004.

Outline

- OpenGL Raster Transformations
- Transformations Between 2D Coordinates
- Geometric Transformations in 3D

OpenGL Raster Transformations

- Copying pixels from one buffer area to another can be accomplished with

  glCopyPixel(xmin, ymin, width, height, GL_COLOR);

  - GL_COLOR says what is to be copied (color values)
  - Copied to refresh buffer at same loc

More OpenGL Raster Transformations

- To read into an array:

  glReadPixels(xmin, ymin, width, height, GL_RGB, GL_UNSIGNED_BYTE, colorArray);

  – To do a 90 degree rotation could rearrange rows and columns of array, then place back to refresh buffer at current raster position

  glDrawPixels(width, height, GL_RGB, GL_UNSIGNED_BYTE, colorArray);

Yet more OpenGL Raster Transformations

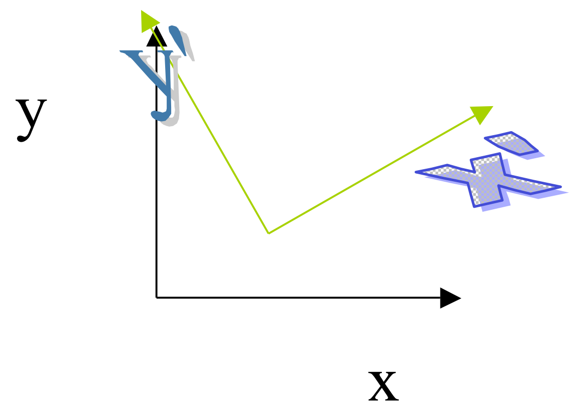
- To scale an area use:

  glPixelZoom(sx,sy);

  where sx and sy are any nonzero floating-point values. (Negative values cause reflections.
- Then use glCopyPixels or glDrawPixels to get/draw the pixels with the given scaling.

Transformations Between 2D Coordinates

- Want to be able to switch between one coordinate system and another:



How to do 2D coordinate transformations

- If want to go from xy system to x'y'
  - First translate origin of x'y' system to origin of xy system with **T**(-x0, -y0):

$$\begin{bmatrix} 1 & 0 & -x0 \\ 0 & 1 & -y0 \\ 0 & 0 & 1 \end{bmatrix}$$

  - Then rotate result **R**($-\theta$) so

    xy coordinate now usual

    coordinates:

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Another method for 2D-coordinate transformations

- Pick a vector **V** specifying the direction of the y' axis.
  - Then **v = V/|V| = (v_x, v_y)** is a unit vector in this direction
  - So **u = (v_y, -v_x)** will complete the coordinate system
  - Transformation can be written as:

$$\begin{bmatrix} v\_y & -v\_x & 0 \\ v\_x & v\_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  - Finally, a translation by a point **P0** can make this general

Geometric Transformations in 3D

- Many of the kinds of transformations done for 2D can be extended to 3D transformations
- For example, 3D translations are like 2D translation except now also can move in z direction.
- For rotations things are a bit more complicated. Can build up out of rotations around the three coordinate axes.
- We also will use homogeneous coordinates for 3D transformations. Thus, will use 4x4 matrices to describe operations.

3D Translations

- As an example of what 3D transformations look like consider matrix for a translation

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$