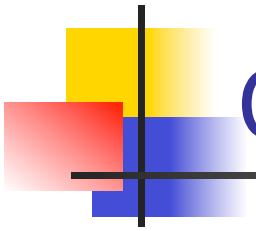


OpenGL Transformation Composition

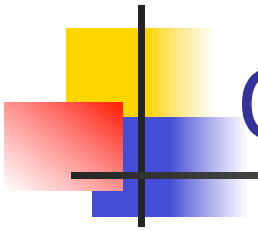
- A global modeling transformation matrix (GL_MODELVIEW, called it M here)

  glMatrixMode(GL_MODELVIEW)

- The user is responsible to reset it if necessary

  glLoadIdentity()

  -> M = 1 0 0
  0 1 0
  0 0 1



OpenGL Transformation Composition

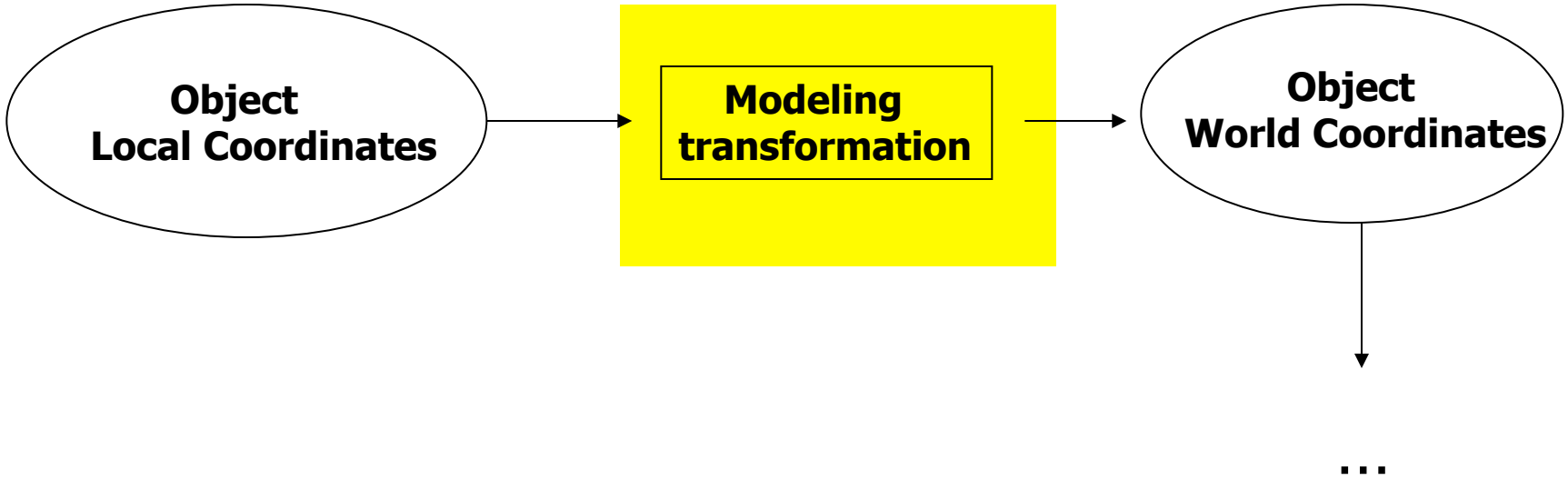
- Matrices for performing user-specified transformations are multiplied to the model view global matrix
- For example,

glTranslated(1,1 0); $M = M \times \begin{vmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$

- All the vertices P will go through the transformation (modeling transformation)

$$P' = M \times P$$

Transformation Pipeline



OpenGL Transformation

- OpenGL postmultiplies each new transformation matrix

  $M = M \times M_{new}$

- Example: perform translation, then rotation

  0) M = Identity

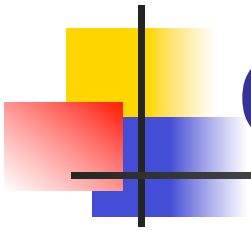
  1) translation T(tx,ty,0) -> M = M x T(tx,ty,0)

  2) rotation R(θ) -> M = M x R(θ)

  3) Now, transform a point P -> P' = M x P

  = T(tx, ty, 0) x R(θ) x P

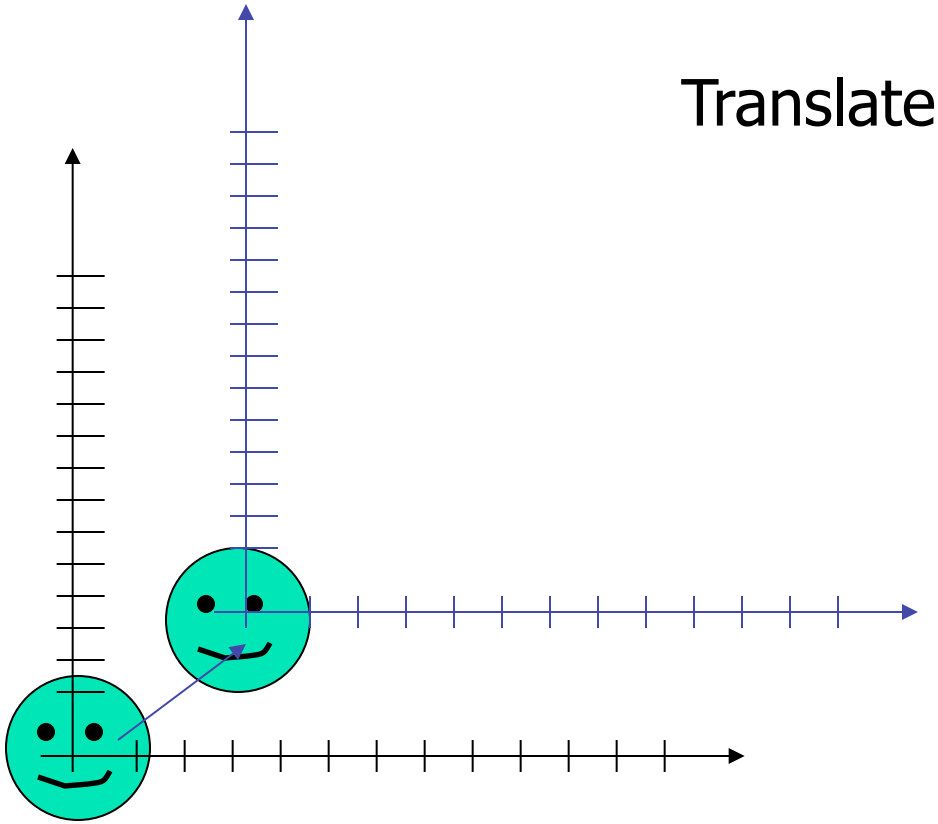
Wrong!!!



OpenGL Transformation

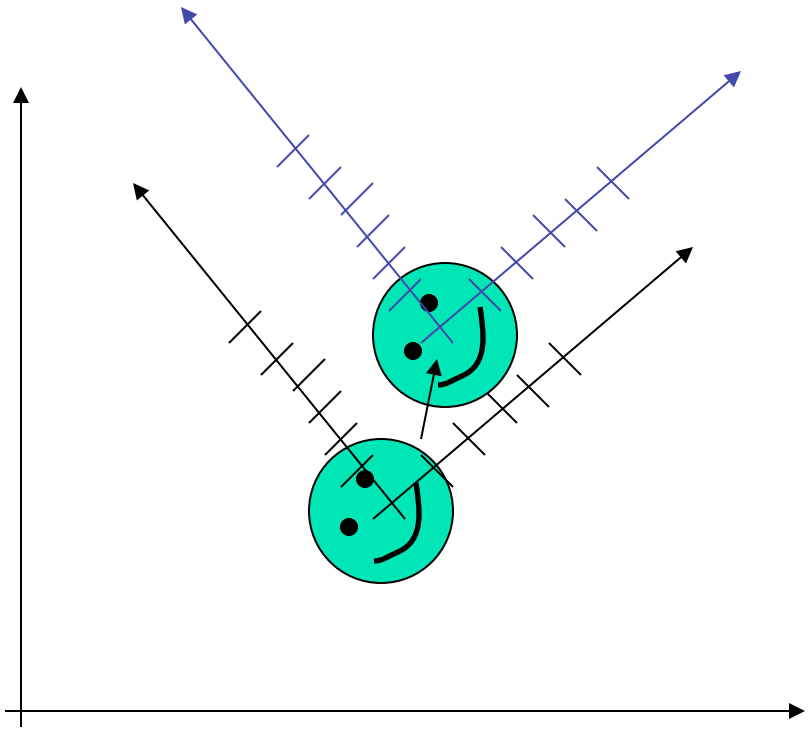
- When use OpenGL, we need to think of object transformations as moving its local coordinate frame
- All the transformations are performed relative to the current coordinate frame origin and axes

Translate Coordinate Frame



Translate (3,3)?

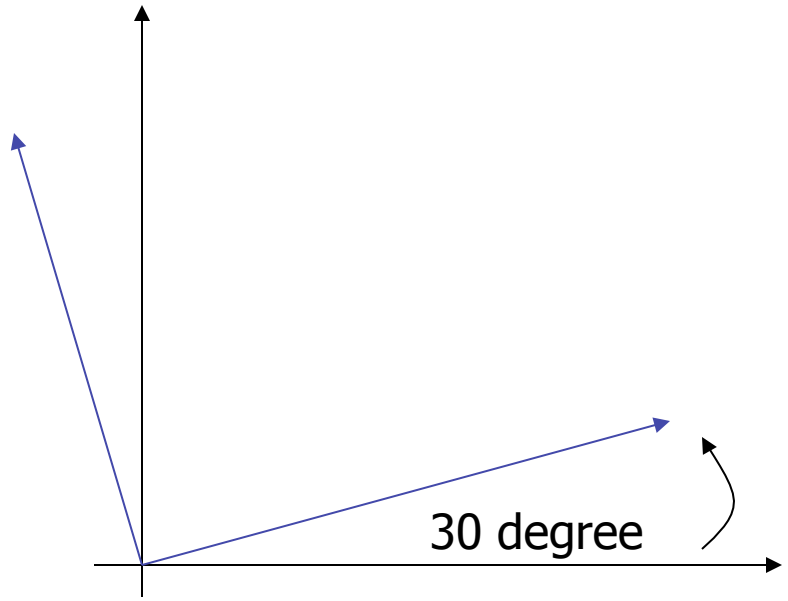
Translate Coordinate Frame (2)



Translate (3,3)?

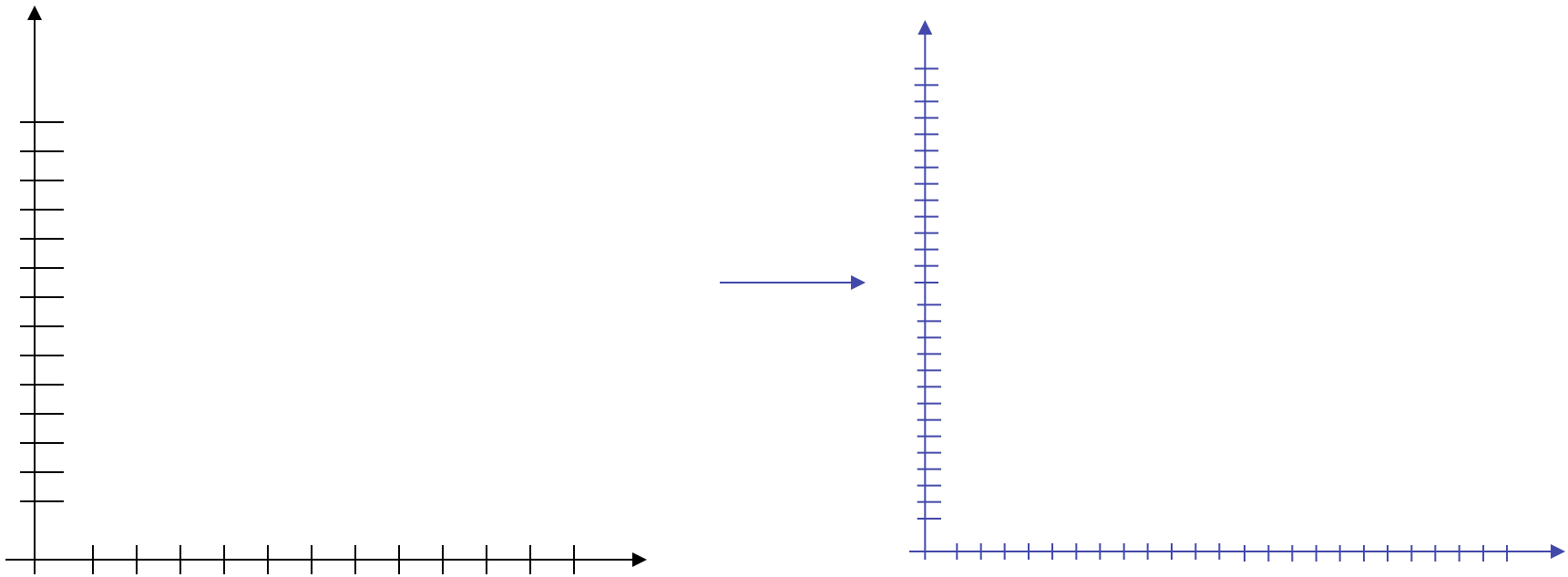
Rotate Coordinate Frame

Rotate 30 degree?

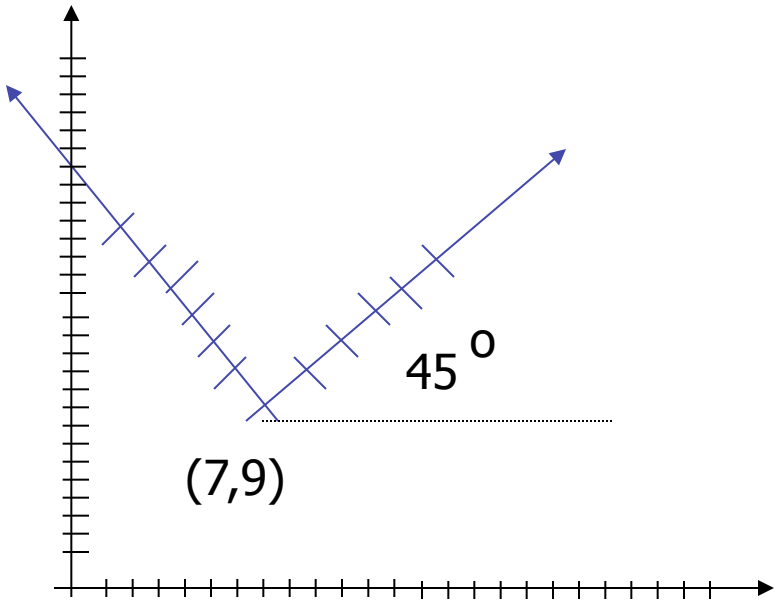


Scale Coordinate Frame

Scale (0.5,0.5)?



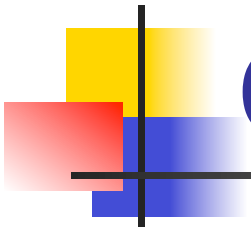
Compose Transformations



Transformations?

Answer:

1. Translate(7,9)
2. Rotate 45
3. Scale (2,2)



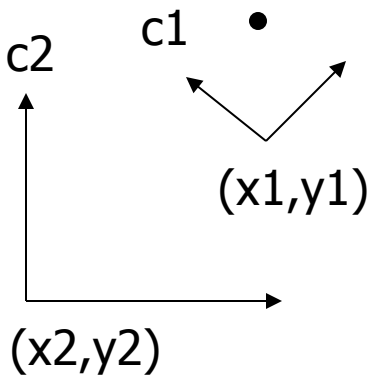
OpenGL Transformation

- Think of transformation as moving coordinate frames
- Call OpenGL transformation functions in that order
- OpenGL will actually perform the transformations in the reverse order

Transform Coordinates

Coordinate system transformation

- Transform an object from coordinate system C1 with the origin at (x1,y1) or (x1,y1,z1) in 3D, to coordinate system C2 with the origin (x2,y2) or (x2,y2,z1) in 3D



1. Find the transformation sequence to move C2 to C1 (so C2 will align with C1)
   - Move the origin of C2 to coincide with the origin of C1
   - Rotate the basis vectors of C2 so that they coincide wih C1's.
   - Scale the unit if necessary
2. Apply the above transformation sequence to the object in the opposite order