



Multiple Coordinate Spaces

CS 355: Interactive Graphics and Image Processing

Objects



Objects



Objects



Objects

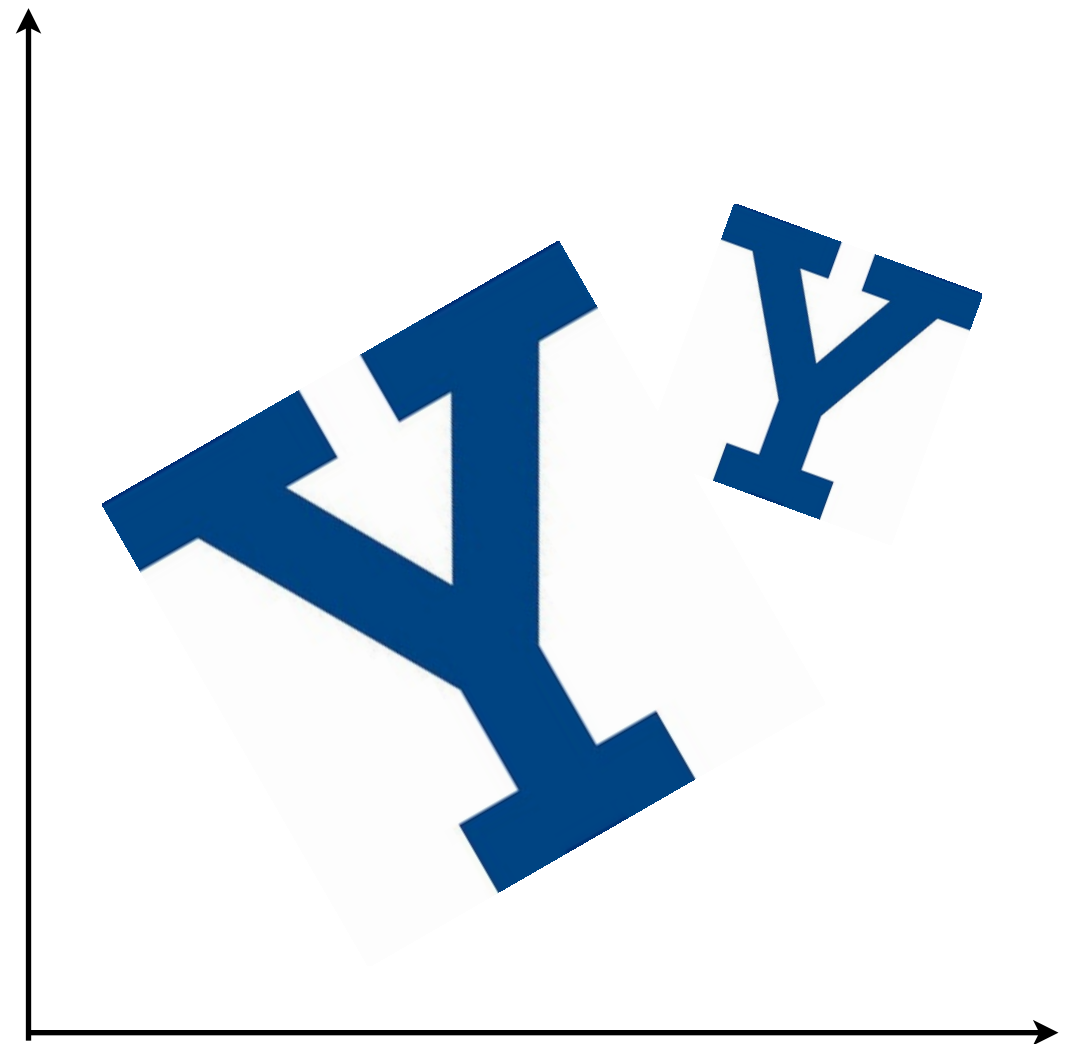


Objects



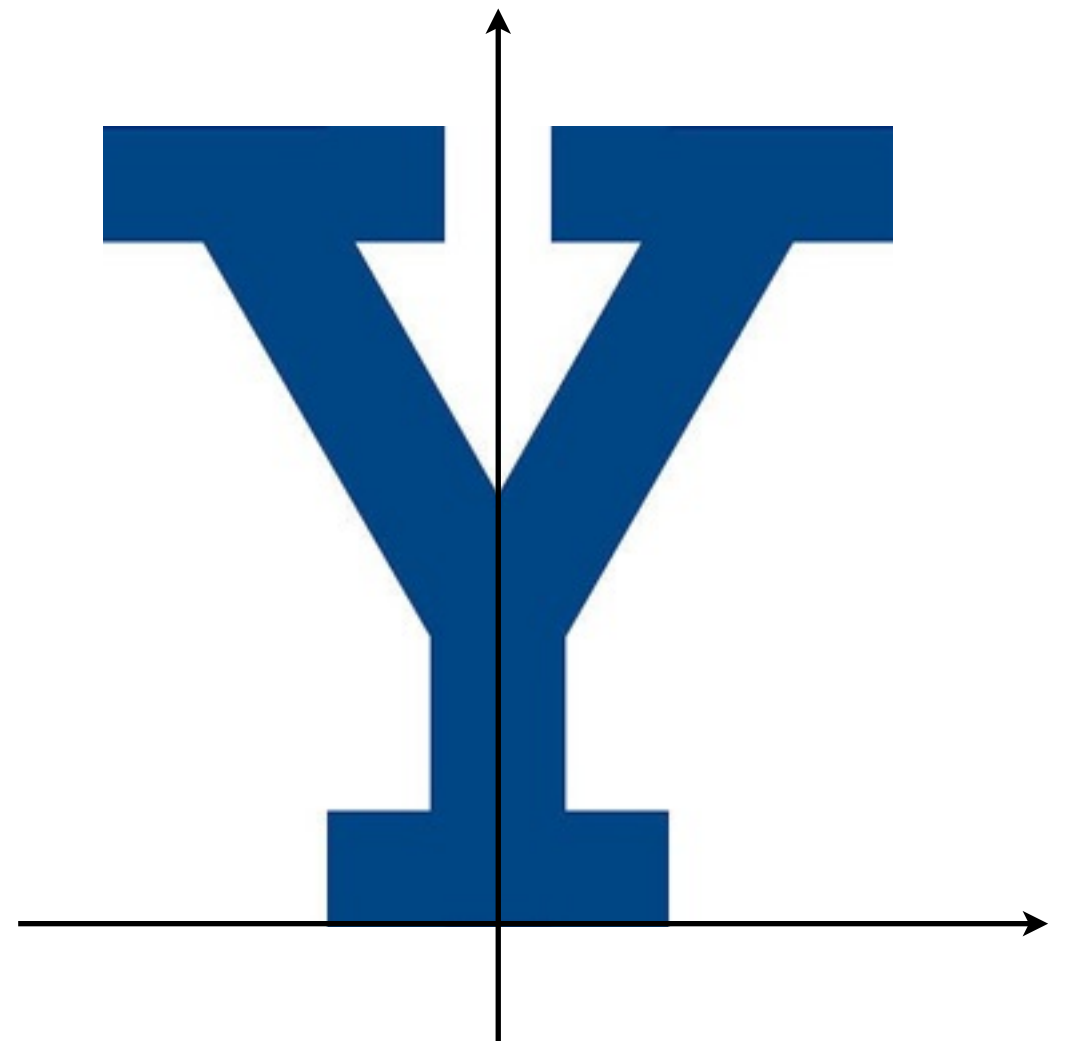
World Space

- The “world space” defines the space in which objects can live
- Choice of origin and coordinate system is arbitrary



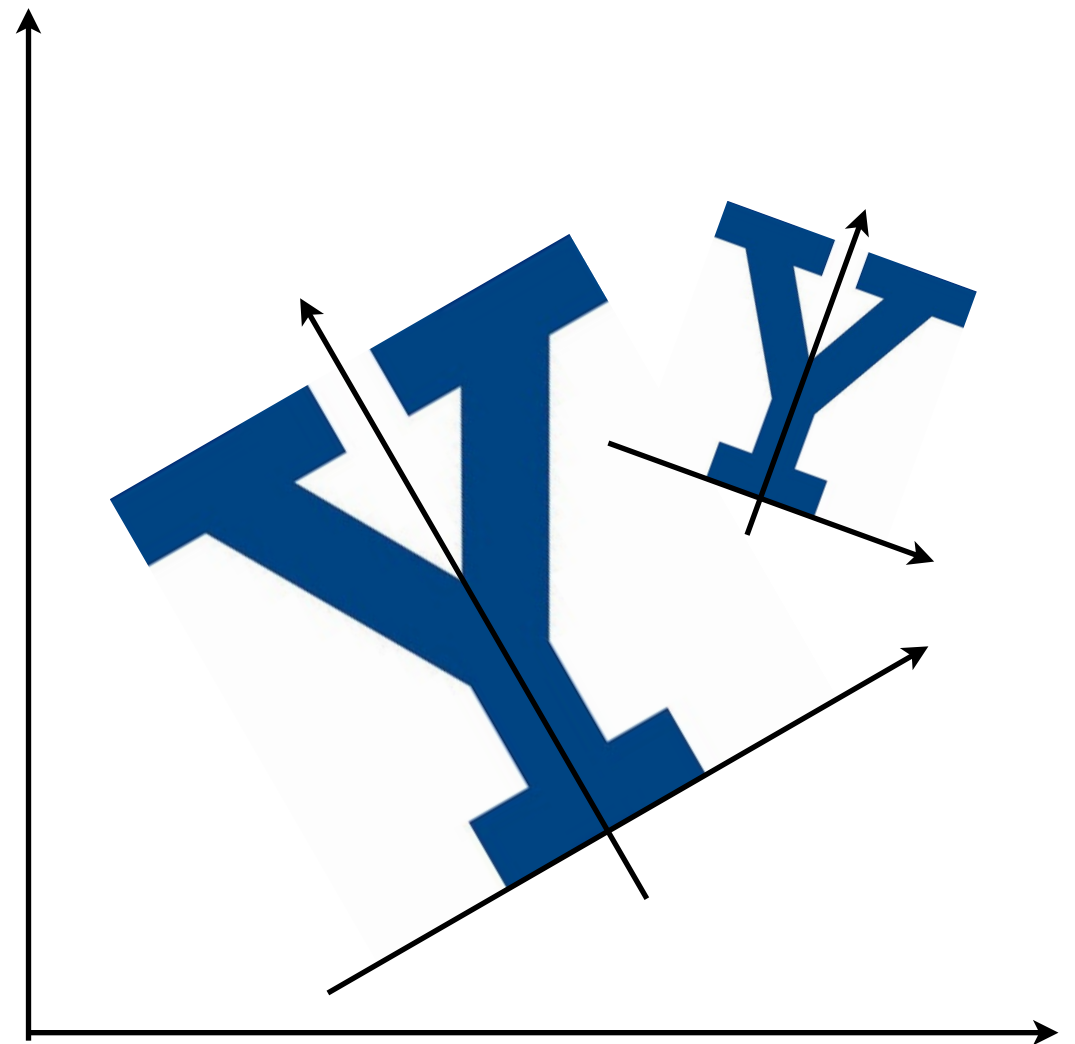
Object Space

- The coordinate system used to define an object
- Choice of origin and coordinate axes also arbitrary
- But usually chosen to make object definition the simplest



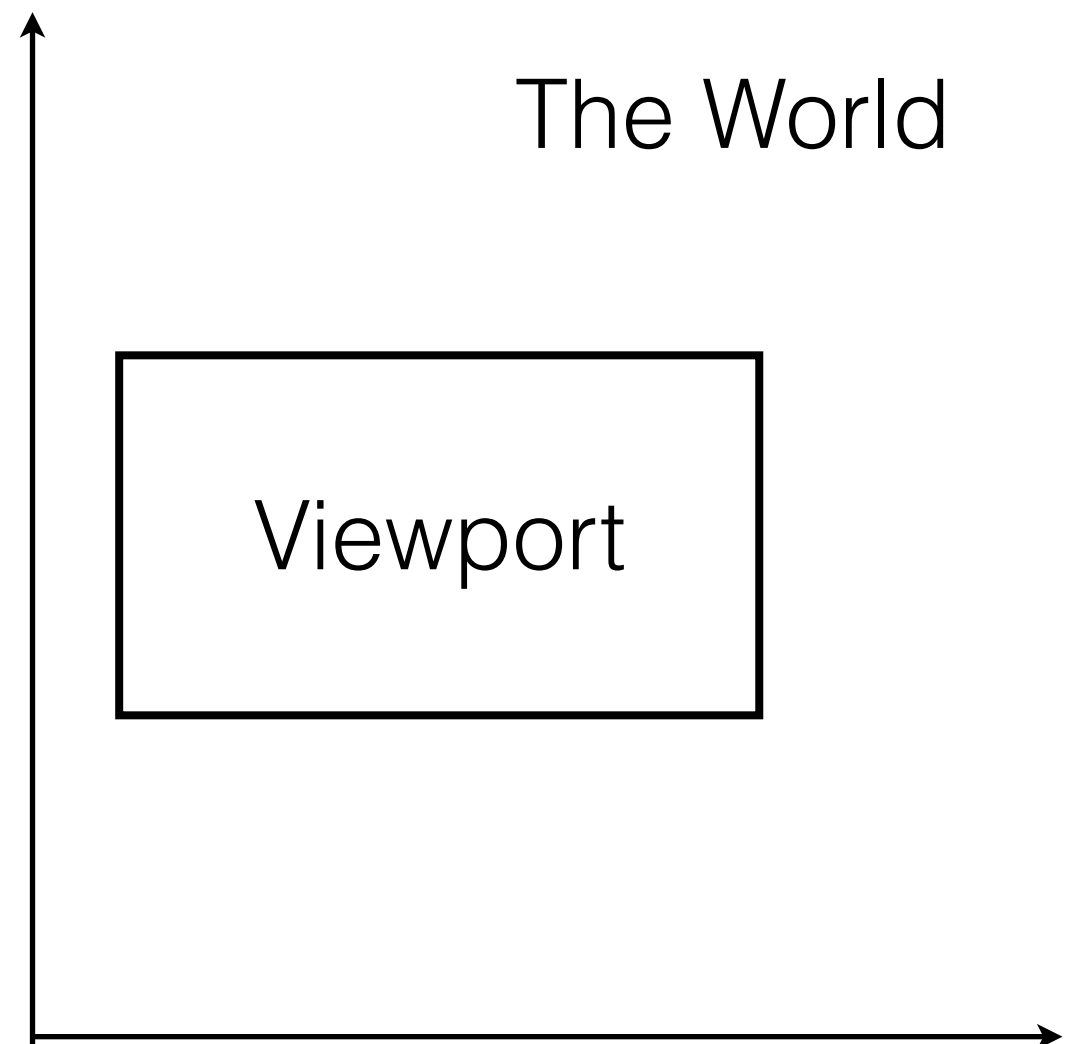
Objects in the World

- Placing an object in the world:
 - Location
 - Orientation
 - Size
- These define an *object-to-world transformation*



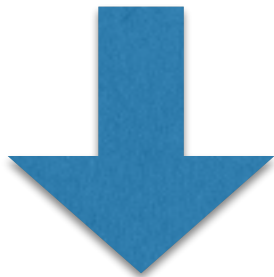
Viewing Coordinates

- We don't see the entire world, only a window into it
 - Location
 - Orientation
 - Size
- These define a *world-to-view transformation*



All Together

- Object Coordinates



Lab #2

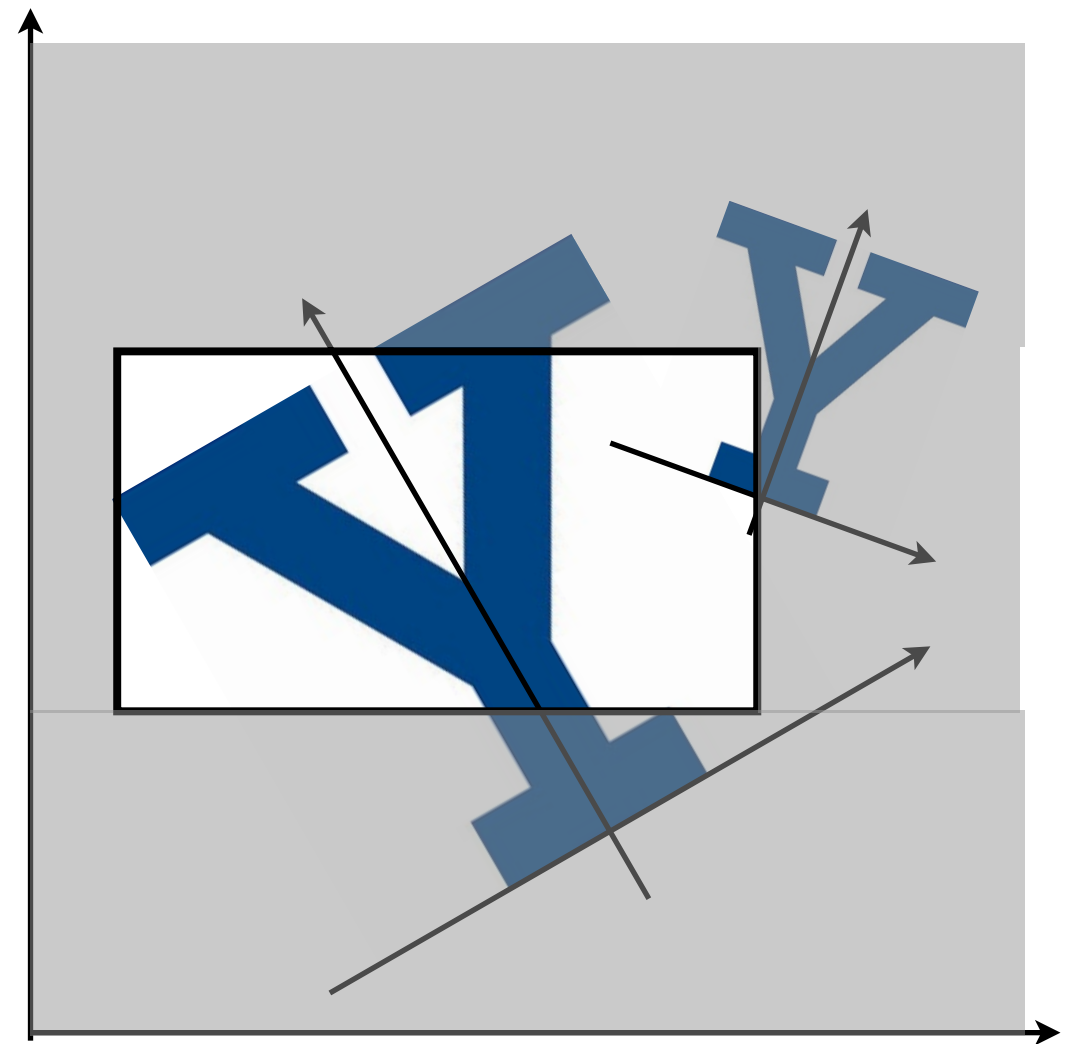
Lab #3

- World Coordinates



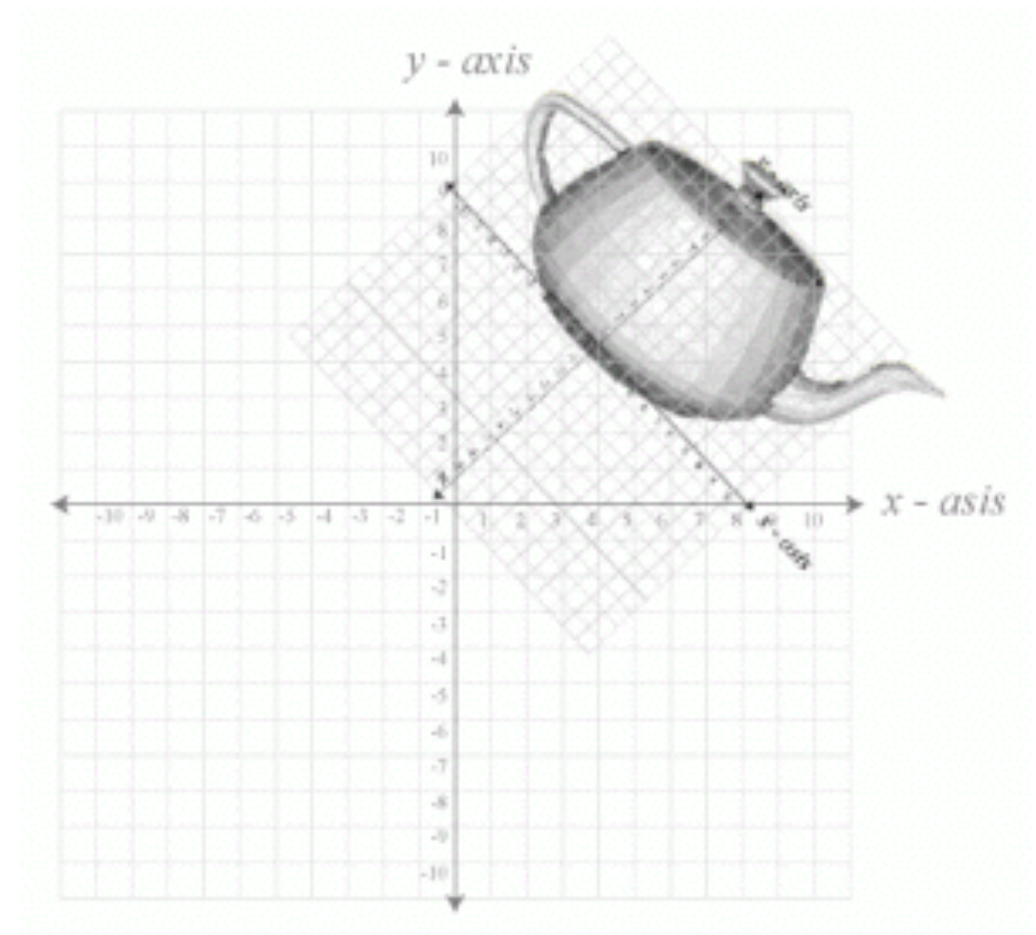
Lab #4

- Viewing Coordinates



3D Modeling & Rendering

- Define a 3D model in object space
- Place that 3D model in the world (object space to world space)
- Compute position relative to the camera (world space to camera space)
- Project 3D to a virtual camera (and apply perspective if desired)
- Apply a 2D viewport if desired




More on this later when we get to 3D...

Going the Other Way

- Object Coordinates

Test to see
if in shape

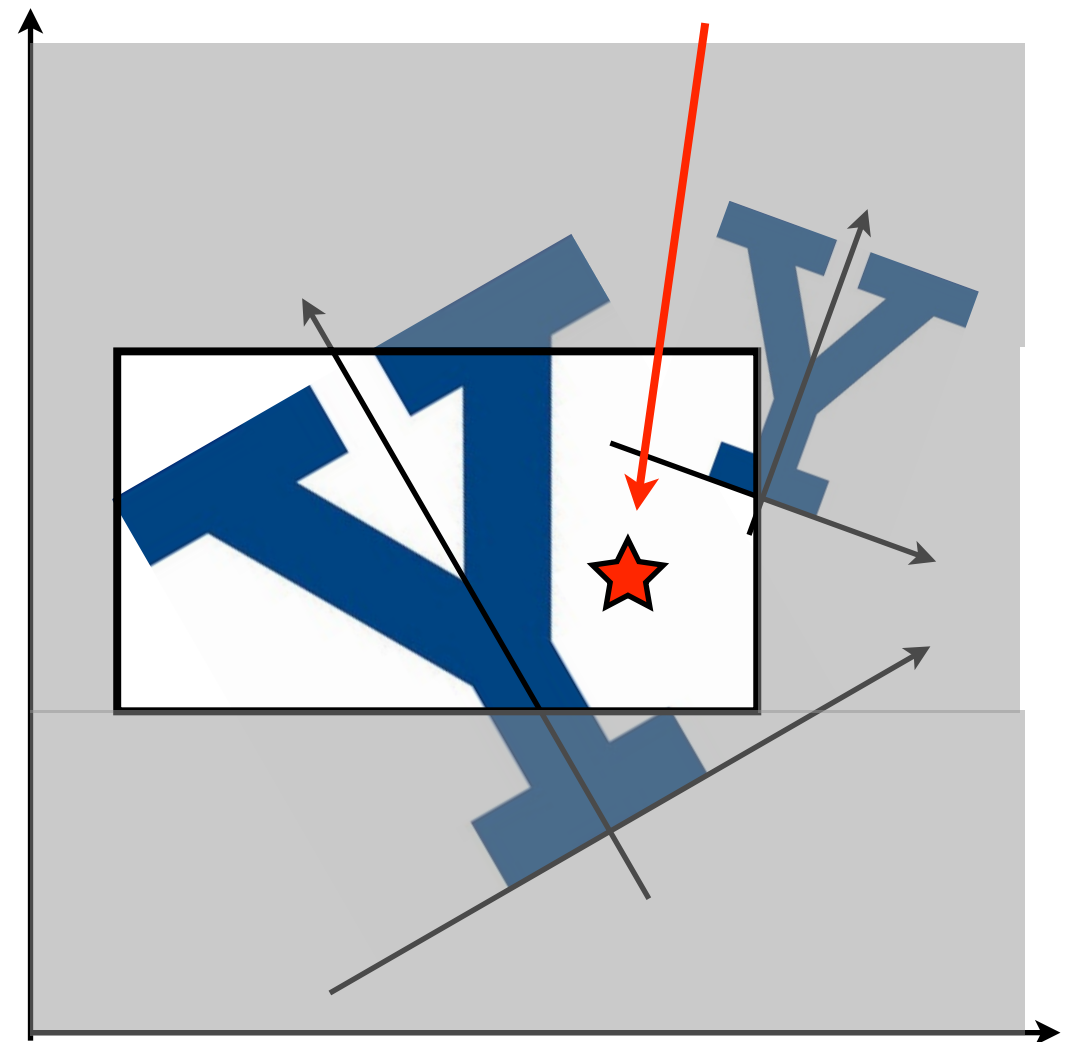


- World Coordinates



- Viewing Coordinates

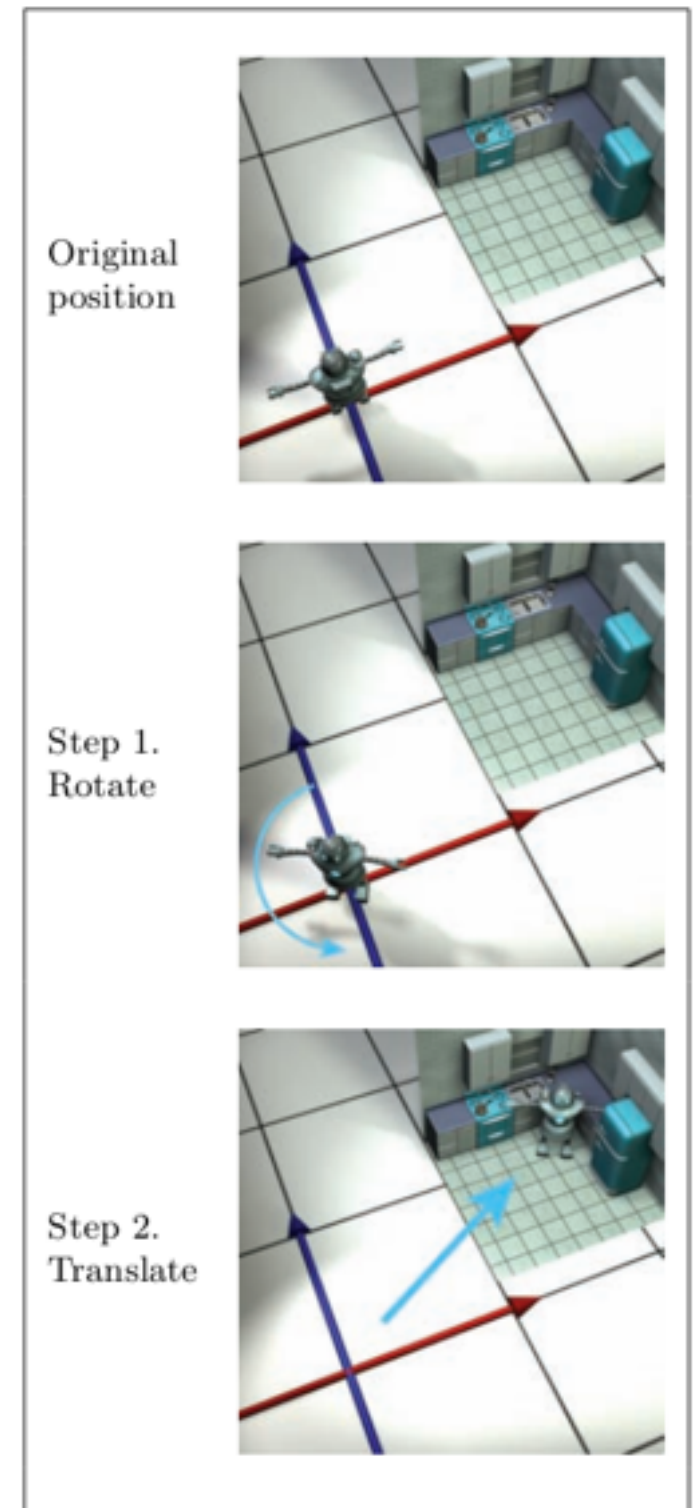
Mouse click
in screen coordinates



Object to World

- An object has a *position* and an *orientation*
- First: **rotate** in object space to desired world-space orientation
- Second: **translate** (move) to the position in world space

Order matters!

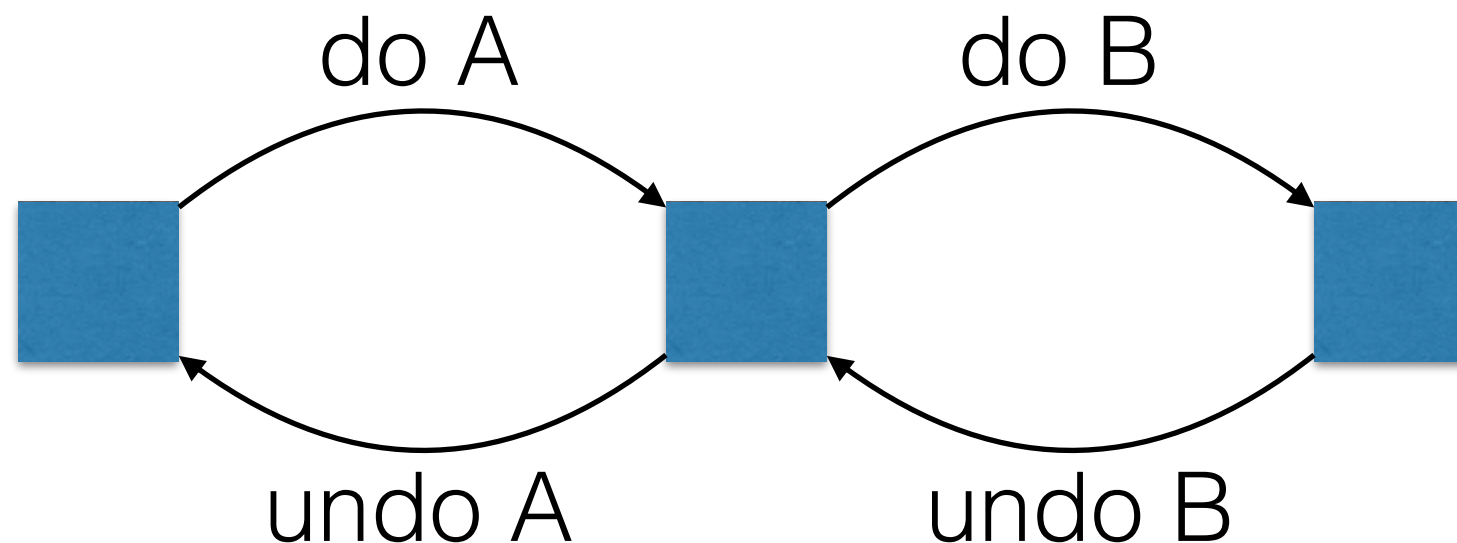


Transformed Drawing

```
Graphics2D g;  
  
// create a new transformation (defaults to identity)  
AffineTransform objToWorld = new AffineTransform();  
  
// rotate by  $\pi / 4$   
objToWorld.rotate(Math.PI / 4);  
  
// translate to (100,50)  
objToWorld.translate(100,50);  
  
// set the drawing transformation  
g.setTransform(objToWorld);  
  
// and finally draw  
g.fillRect(-width/2,-height/2,width,height);
```

Inverse Transformations

- The inverse of a transformation “undoes” that transformation
- If a sequence:
 - do sequence of steps backwards
 - do each step backwards



World to Object

- An object has a position and an orientation
 - First: **translate** (move) back from the position in world space to object space
 - Second: **rotate** in object space back from desired world-space orientation

Example

Forward (Drawing)

- Store as simple rectangle centered at origin
- Rotate by $\pi / 4$
- Translate by (100,50)

Backward (Selecting)

- Translate by (-100,-50)
- Rotate by $-\pi / 4$
- Test against simple rectangle centered at origin

Transformed Selection

```
Point2D worldCoord; // location of screen click
Point2D objCoord; // location relative to object

// create a new transformation (defaults to identity)
AffineTransform worldToObj = new AffineTransform();

// translate back from (100,50)
worldToObj.translate(-100,-50);

// rotate back from  $\pi / 4$ 
worldToObj.rotate(- Math.PI / 4);

// and transform point from world to object
worldToObj.transform(worldCoord, objCoord);
```

Lab #2 Preview

- Model:
 - Parent **Shape** class includes position and orientation
 - Child classes stored in object-centric coordinate system
- Drawing (viewer):
 - Use an object-to-world **AffineTransform** for the drawing
 - Then draw as if centered at origin and not rotated
- Selection (controller or model):
 - Use a world-to-object **AffineTransform** to convert mouse positions to object space
 - Perform simple selection tests in object space

Lab #2 Preview

- New selection tool
- Click in a shape to select (or within 4 pixels of a line)
- Once selected, highlight shape and draw rotation “handle”
(no rotation handle for lines; optional for circles)
- Once selected,
 - Change color
 - Click in shape and drag to move
(just change the stored position)
 - Click in rotation handle and drag to turn
(just change the stored orientation)

Coming up...

- Selection geometry
- Introduction to matrices
- Matrix transformations
 - Forward
 - Inverse