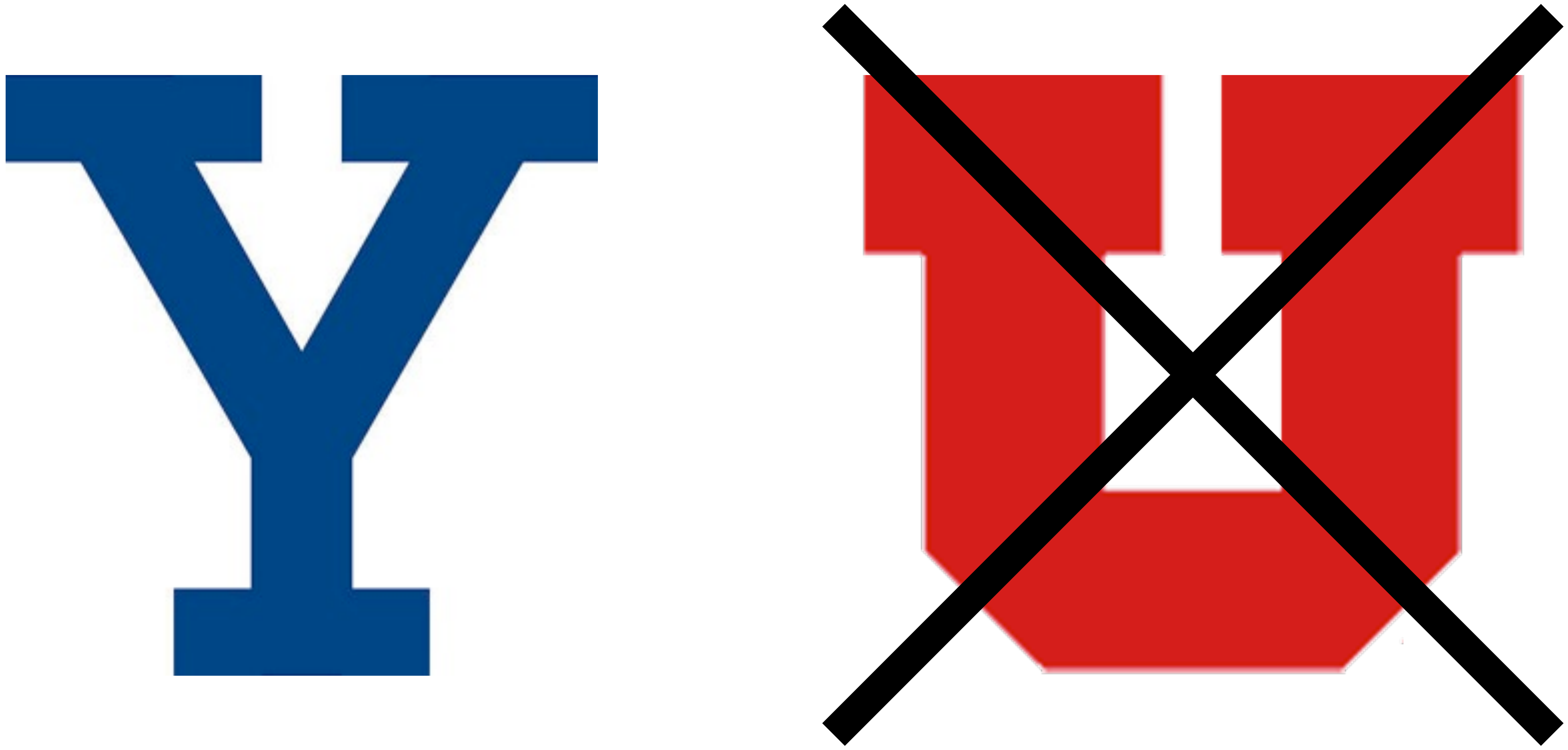
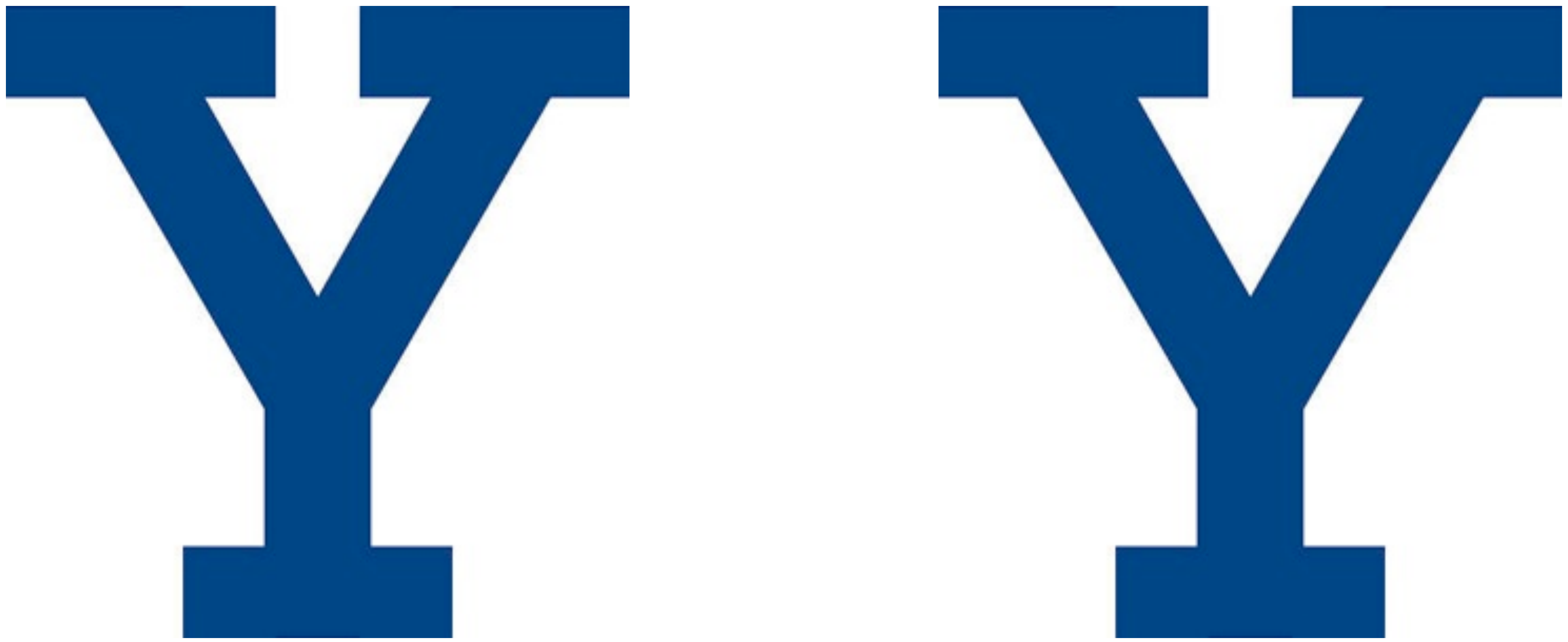
Multiple Coordinate Spaces

CS 355: Interactive Graphics and Image Processing

Objects



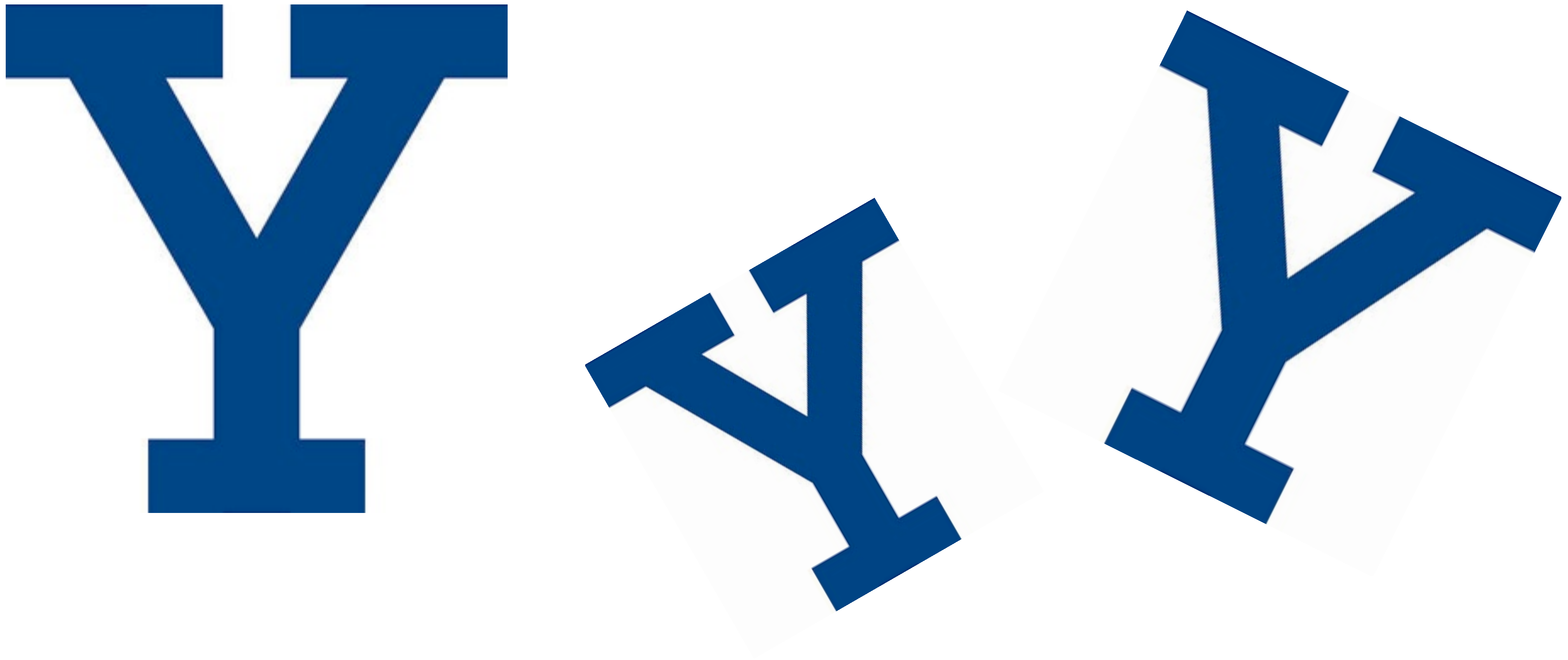
Objects



Objects

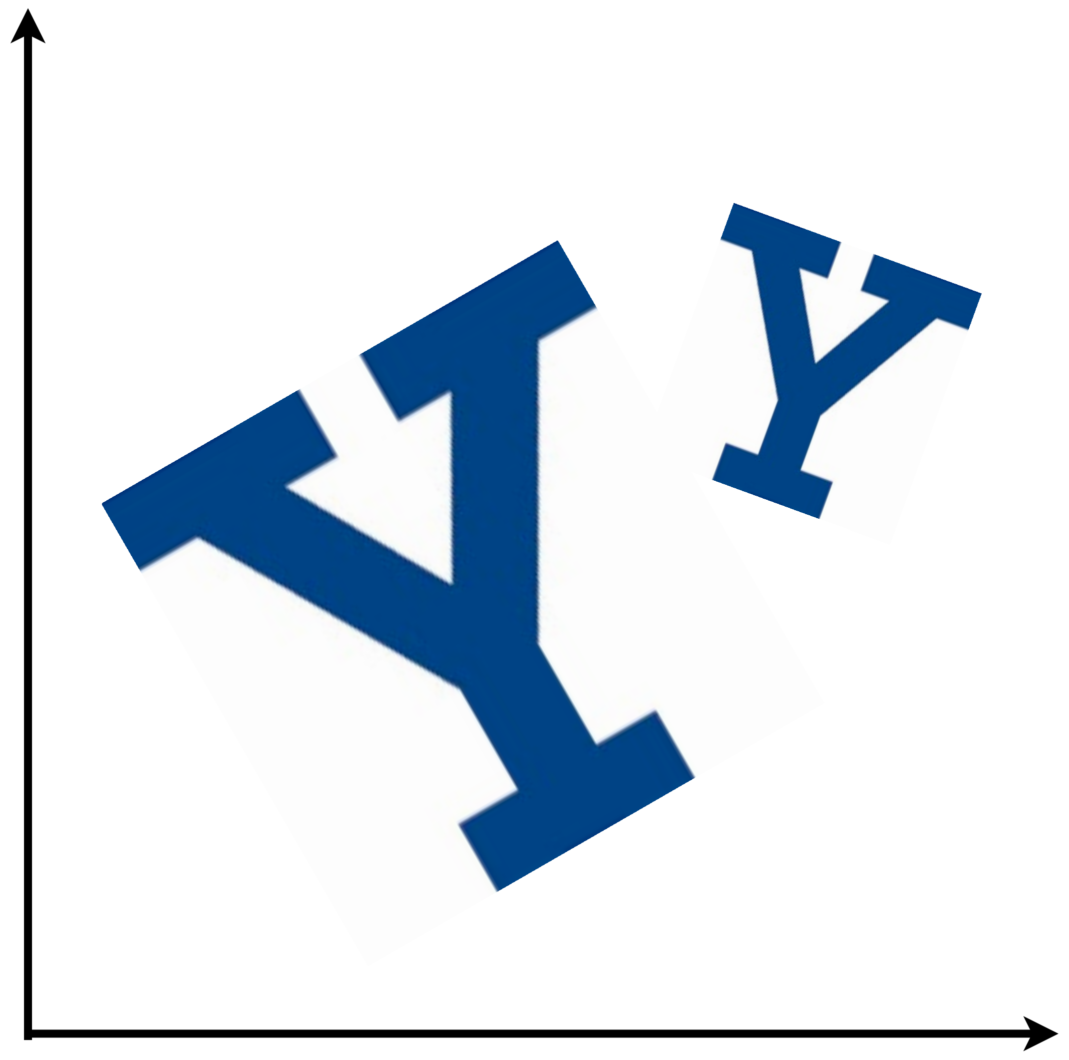
Objects

Objects



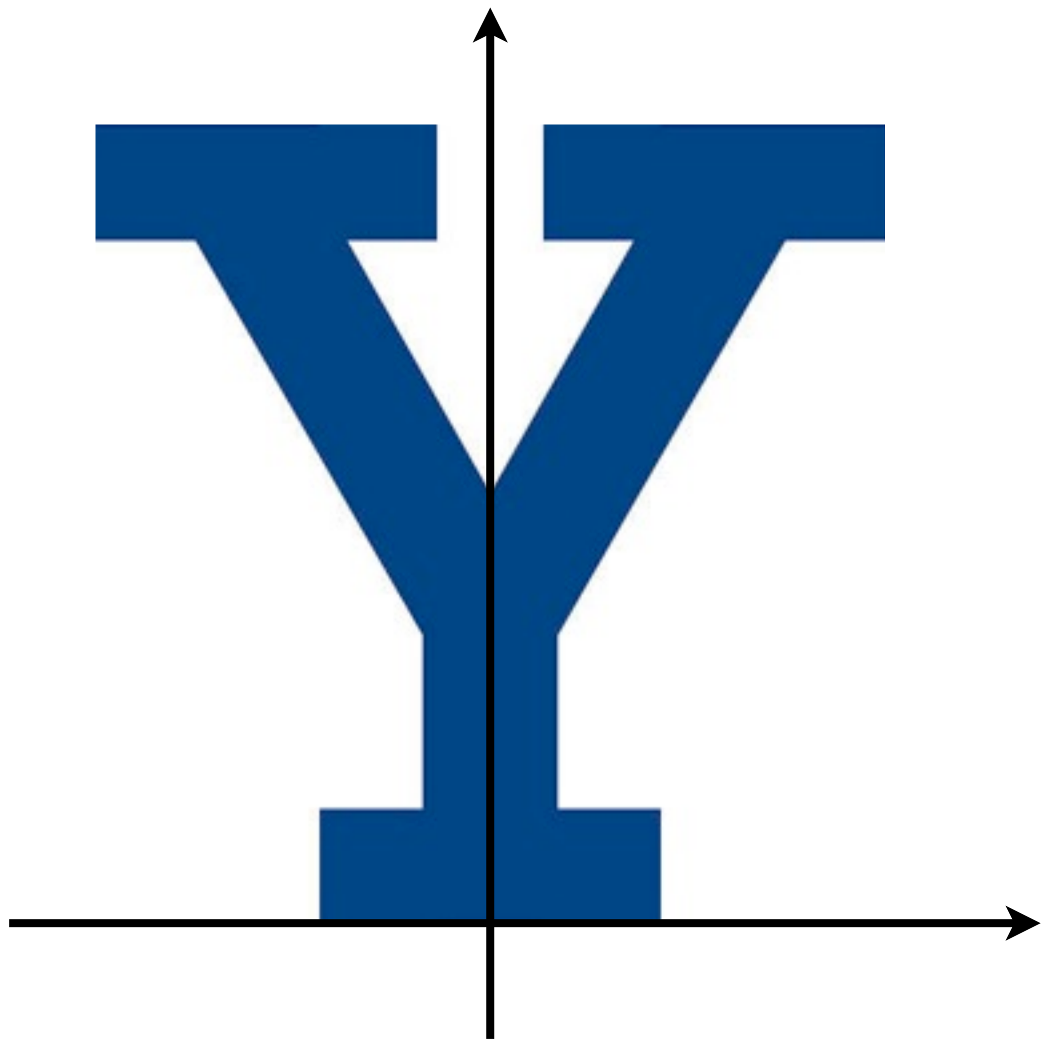
World Space

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



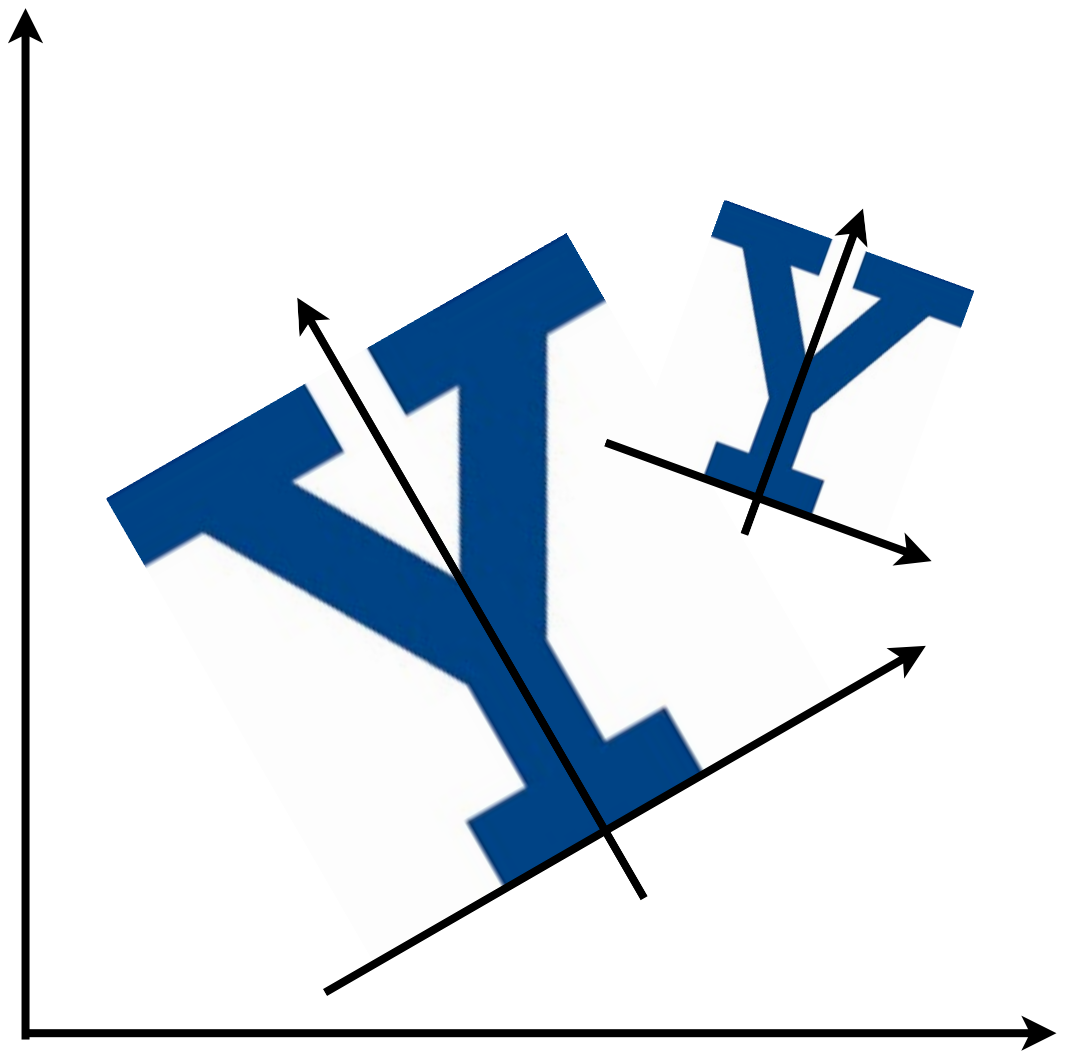
Object Space

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



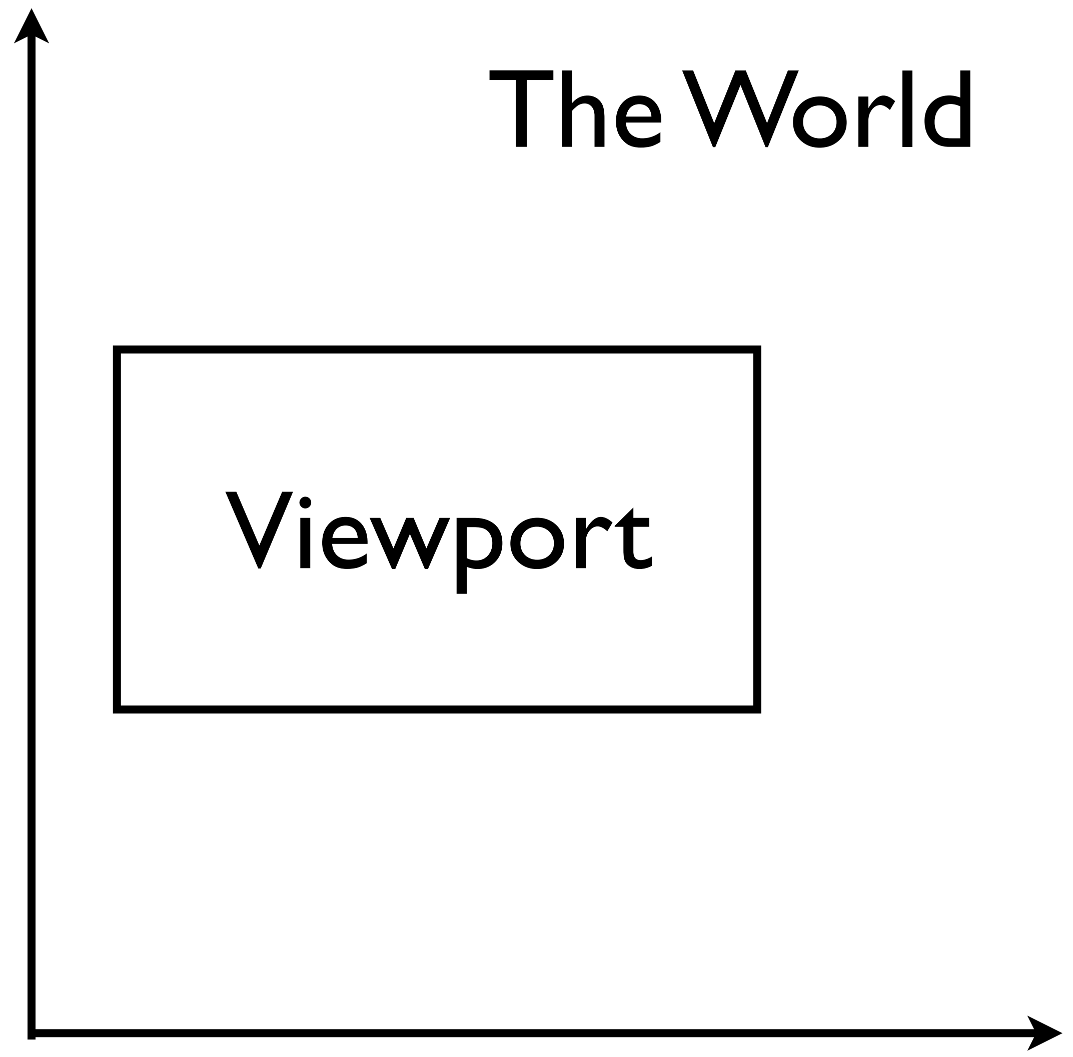
Objects in the World

- 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

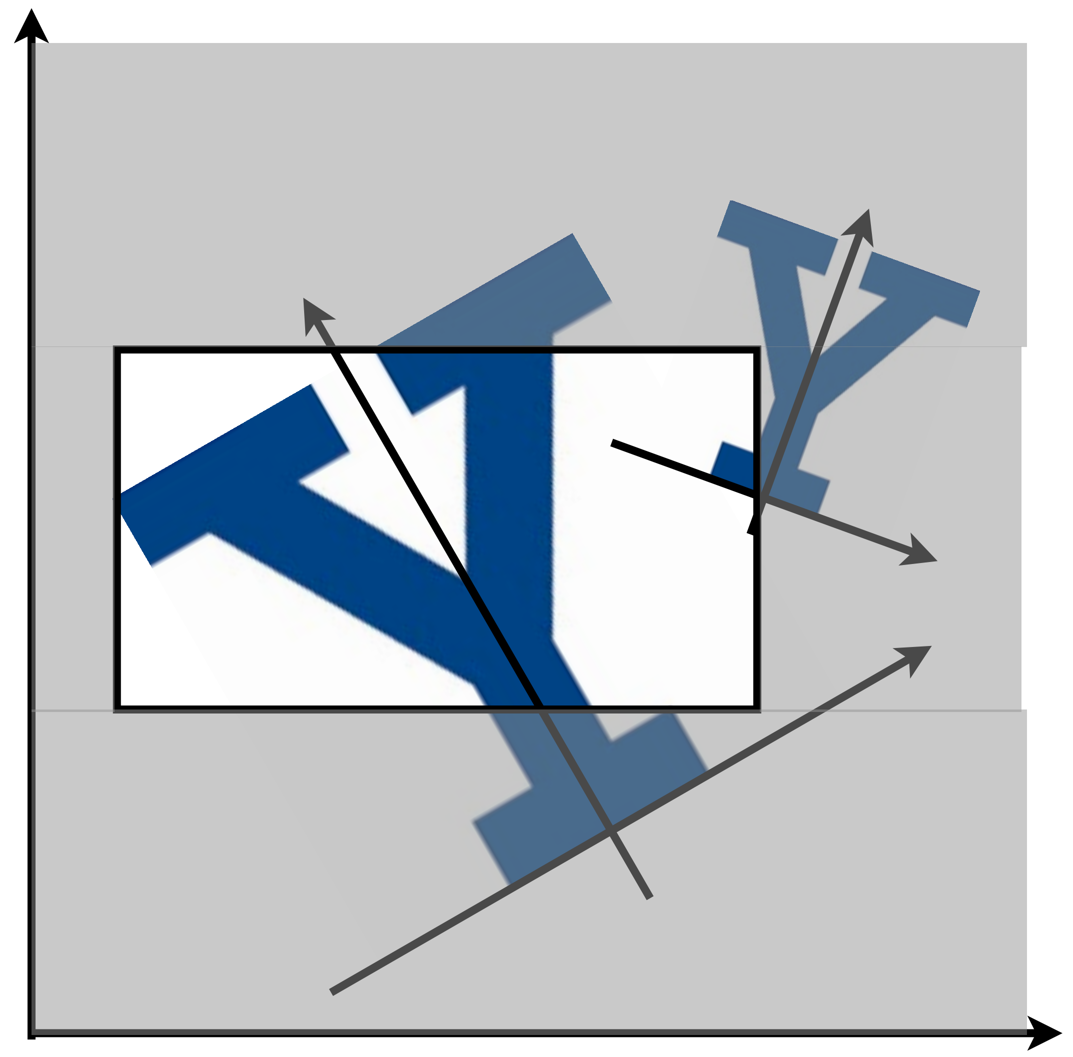
- Object Coordinates

Lab #2

- World Coordinates

Lab #3

- Viewing Coordinates



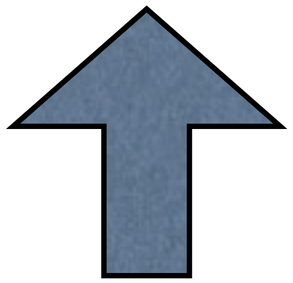
3D Modeling & Rendering

- Define a 3D model in object space
- Place that 3D model in the world
  (object space to world space)
- Figure out where it is *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

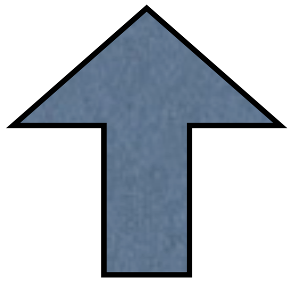
Going the Other Way

- Object Coordinates

Test to see
if in shape

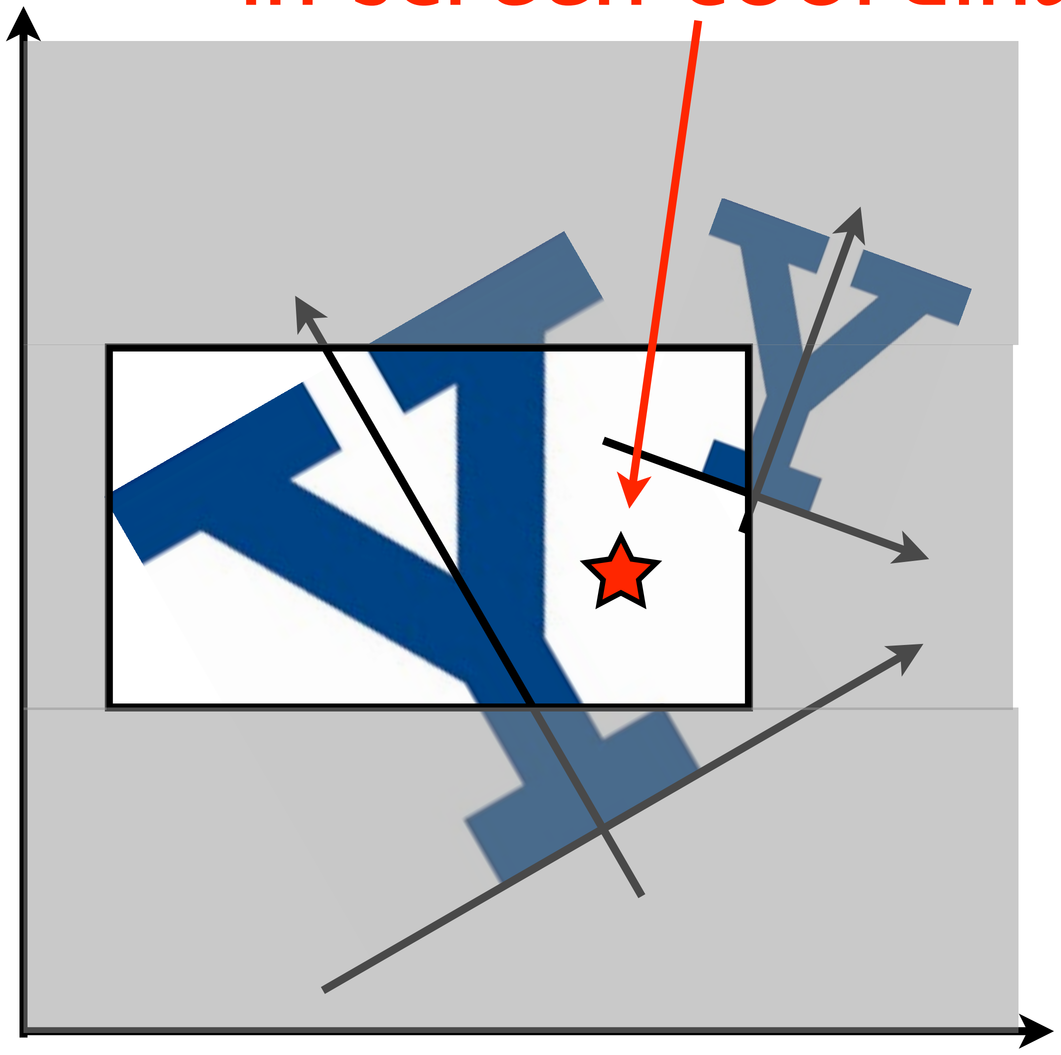


- World Coordinates



- Viewing Coordinates

Mouse click
in screen coordinates



Lab #2

- Clicking in a shape selects it
  - Can change its color
  - Display manipulation “handles”
- Dragging shapes
  - Clicking in a shape and dragging the mouse moves the shape
- Resizing shapes
  - Clicking a corner handle and dragging resizes the shape
- Rotating shapes
  - Clicking the rotation handle and dragging rotates the shape

Lab #2

- Parent `Shape` class:
  - color
  - center
  - rotation angle
- Child subclasses:
  - Shape parameters *in object space*
  - Exception: lines can stay endpoints in world space

Lab #2

- Java's `2DGraphics` drawing supports drawing with transformations
- Supports *affine transformations*
  - Translation (changing position)
  - Rotation (changing orientation)
  - Scale (changing size)
  - and more we won't need for this lab...

Object to World

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

Order matters!

Transformed Drawing

```
Graphics2D g;

// rotate by π / 4
g.rotate(Math.PI / 4);

// translate to (100,50)
g.translate(100,50);

g.fillRect(0,0,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 <u>position</u> and an <u>orientation</u>
    - First: move (translate) *back from* the position in world space to object space
    - Second: rotate in object space *back from* desired world-space orientation

Example

Forward (Drawing)

- Represent as simple rectangle centered at origin
- Rotate by π / 4
- Translate by (100,50)

Backward (Selecting)

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

Coming up...

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