Game Engineering: 2D

CS420-2010F-07

Objects in 2D

David Galles

Department of Computer Science
University of San Francisco

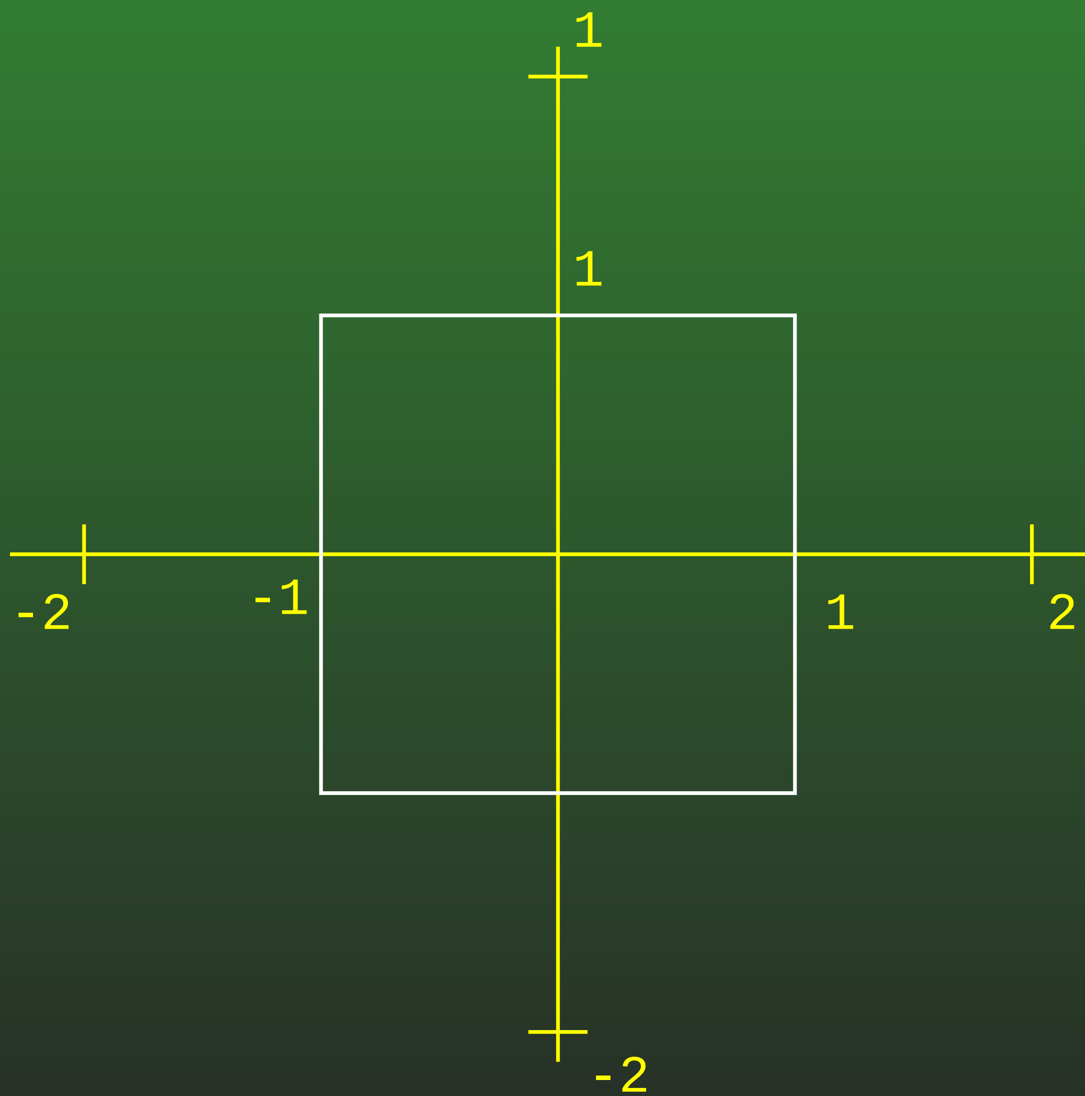
07-0: Representing Polygons

- We want to represent a simple polygon
  - Triangle, rectangle, square, etc
  - Assume for the moment our game only uses these simple shapes
  - No curves for the moment ...
- How could we represent one of those polygons?

07-1: Representing Polygons

- We want to represent a simple polygon
  - Triangle, rectangle, square, etc
  - Assume for the moment our game only uses these simple shapes
  - No curves for the moment ...
- How could we represent one of those polygons?
  - List of points, starting with some arbitrary point, moving “clockwise” around the polygon

07-2: Representing Polygons



(1,1), (1,-1), (-1,-1), (-1,1)

07-3: Representing Polygons

- How could we represent a simple circle?

07-4: Representing Polygons

- How could we represent a simple circle?
  - Center position (Vector)
  - Radius (scalar)

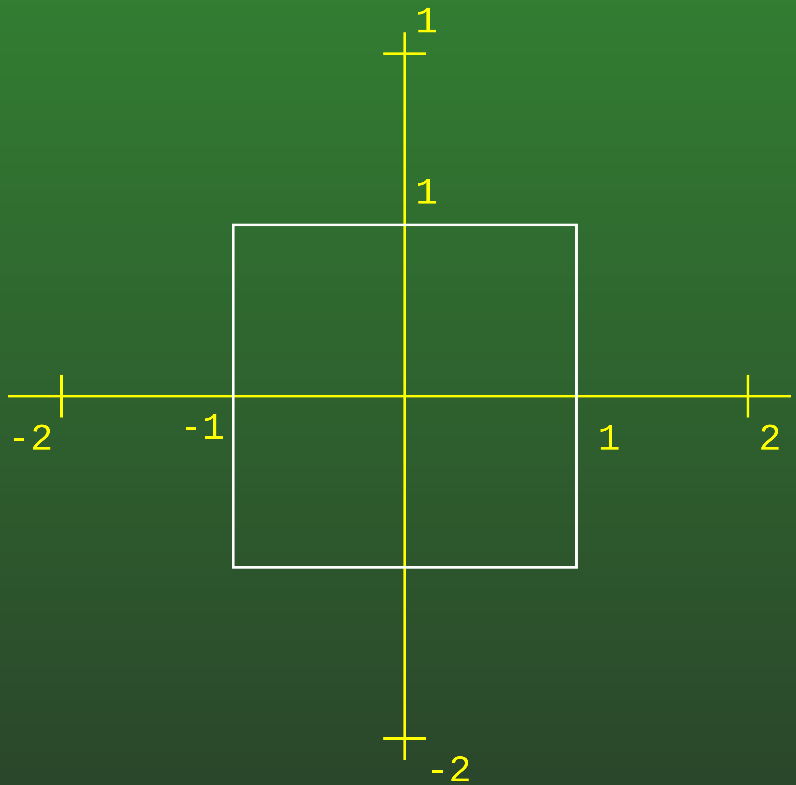
07-5: Modifying Polygons

- What if we wanted to modify a polygon
  - Translation, rotation, scaling, etc
- Start with an easy one – how can we translate (move) a polygon?

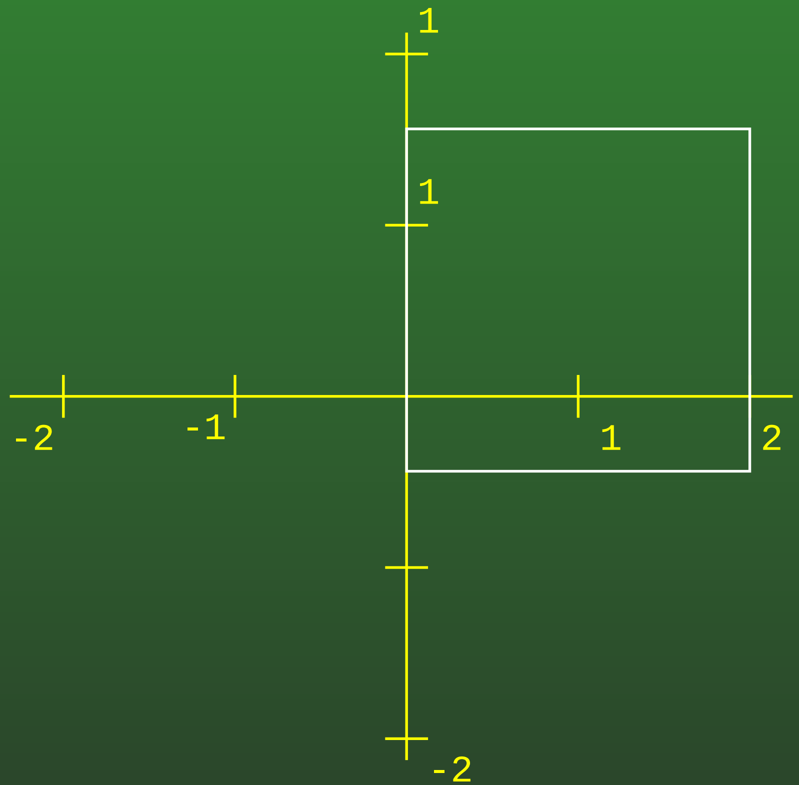
07-6: Translating Polygon

- To translate a polygon, all we need to do is translate each one of its points
  - Move a polygon over 1 unit, up 0.5 units
    - Add (1, 0.5) to each point
    - Points are just translations from origin

07-7: Translating Polygon



(1,1), (1,-1), (-1,-1), (-1,1)

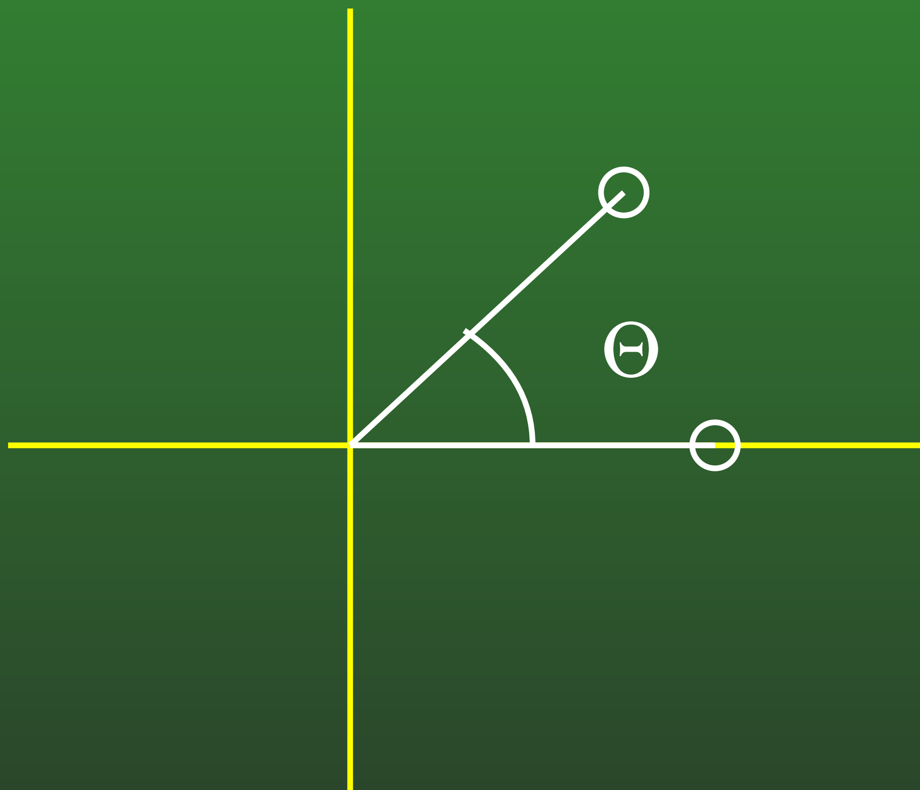


Translate over 1, up 0.5
Add (1,0.5) to each point in polygon
(2,1.5), (2,-0.5), (0,-.5), (-0,1.5)

07-8: Rotation

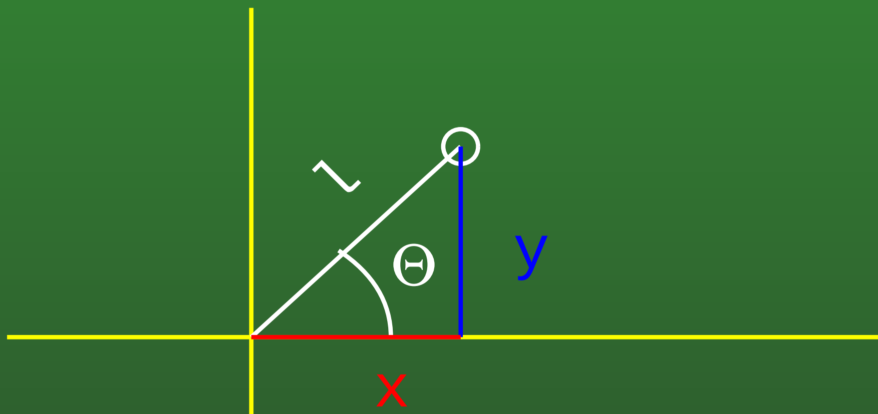
- Rotations are a bit tricky
- Start with a simnplier case
  - Rotate a point around the origin

07-9: Rotation



- Rotate the point (0,x) counterclockwise around the origin by Θ degrees
- What is the new point?

07-10: Rotation



$$\sin \Theta = \frac{y}{l} \qquad y = l \sin \Theta$$

$$\cos \Theta = \frac{x}{l} \qquad x = l \cos \Theta$$

- Rotate the point (0,x) counterclockwise around the origin by Θ degrees
- What is the new point?

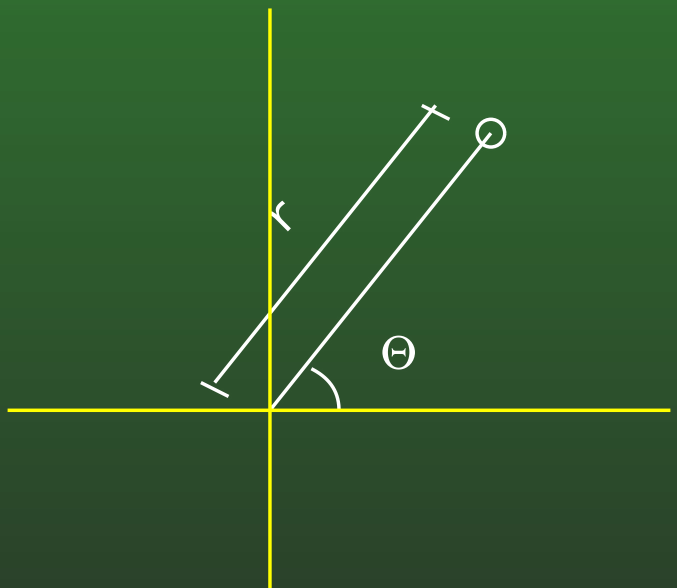
07-11: Rotation

- Original point is at $(x, 1)$
  - distance from the origin $l = x$
- New x position = $x \cos \Theta$
- New y postion = $x \sin \Theta$

Was easy beacuse the disance from the origin l was easy to calculate. What if original point was not on an axis?

07-12: Rotation

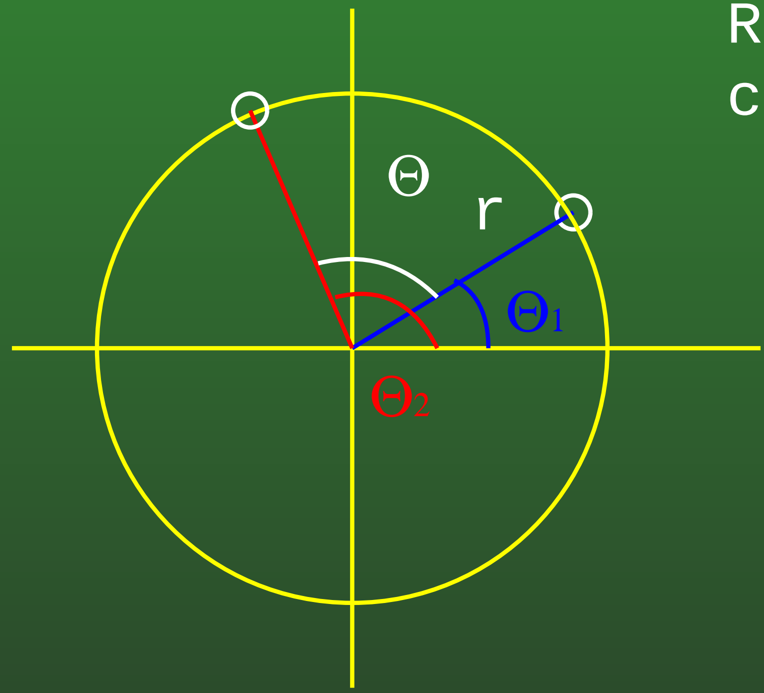
- Rotating a point *not* on an axis
- Use polar coordinates!



07-13: Rotation

- Using polar coordinates for rotation sounds kind of like cheating
  - Of course it is easy to rotate in polar coordinates!
  - Translation is harder though ...
  - (Probably) don’t want to write all our game logic using polar coordinates
    - Depends on the game ...
- Transform into polar coordinates, do rotation, transform back. Hope everything simplifies nicely!

07-14: Rotation



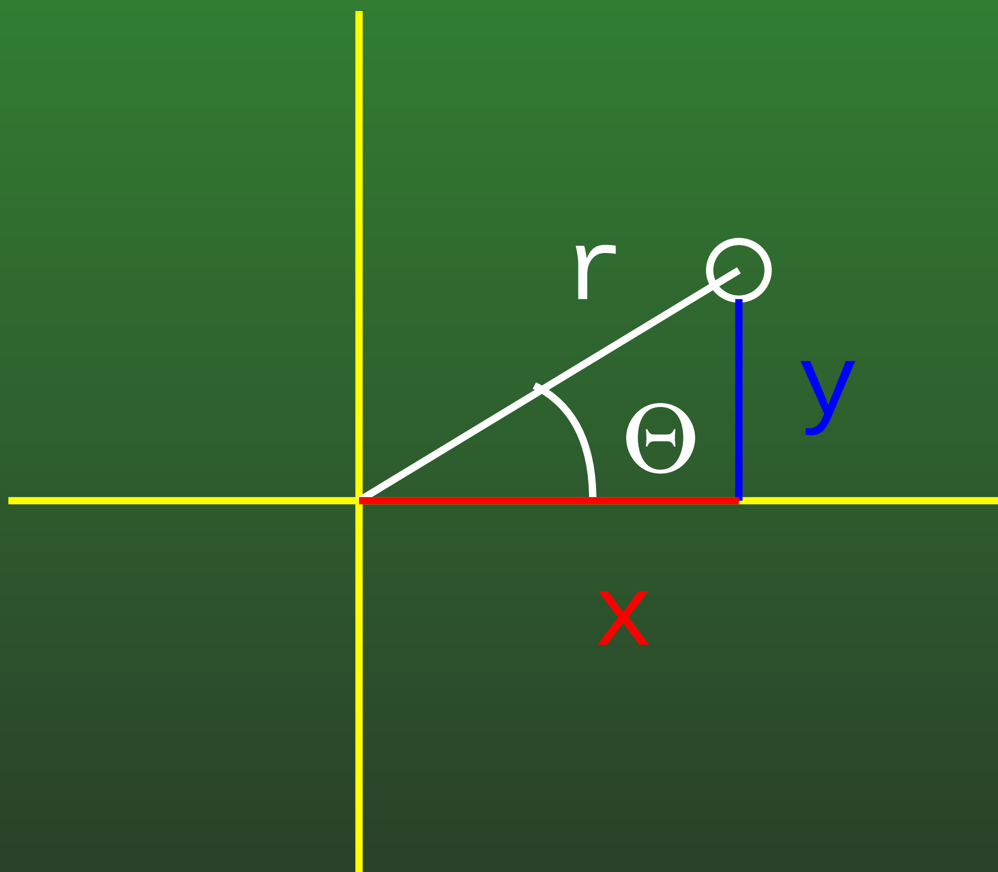
Rotate point at (r, Θ_1) Θ degrees counterclockwise

New point (r, Q2) = $(r, \Theta_1 + \Theta)$

07-15: Rotation

- Conversion from Polar coordinates to Cartesian coordinates
  - Given a point (r, Θ) in Polar coordinates, how can we create a point (x, y) in Cartesian coordinates?

07-16: Polar $\Rightarrow$ Cartesian



$y = r \sin \Theta$

$x = r \cos \Theta$

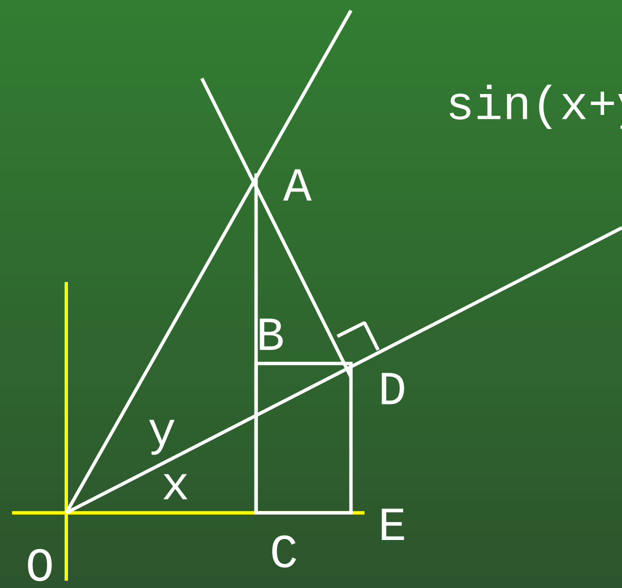
07-17: Rotation

- Point p_1 at (r, Θ_1), rotate p_1 by Θ
- New point p_2 at $(r, \Theta_1 + \Theta)$
- In Cartesian coordinates:

07-18: Rotation

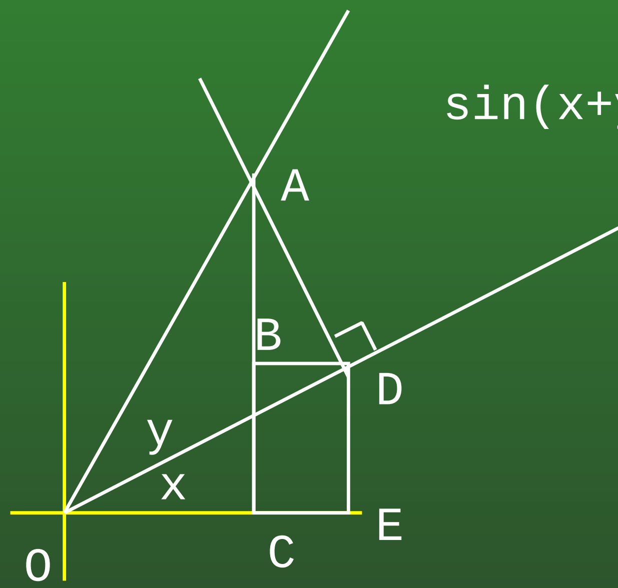
- Point p_1 at (r, Θ_1), rotate p_1 by Θ
- New point p_2 at $(r, \Theta_1 + \Theta)$
- In Cartesian coordinates:
  - $x = r\cos(\Theta_1 + \Theta)$
  - $y = r\sin(\Theta_1 + \Theta)$
- How do we compute $\sin(\Theta_1 + \Theta)$?

07-19: sin(x+y)



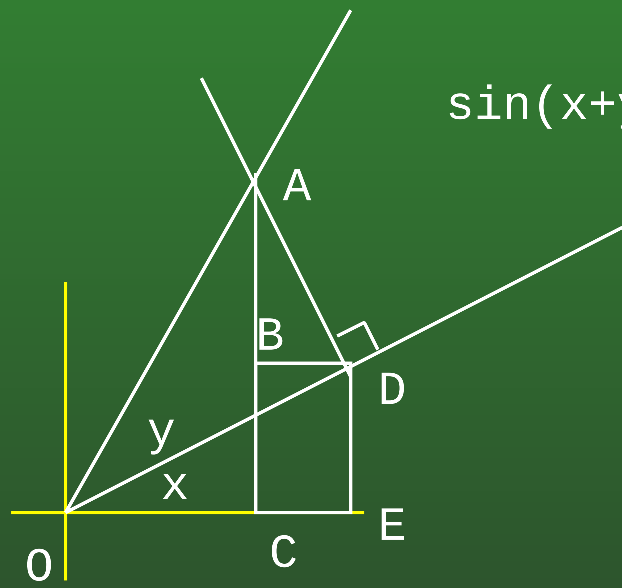
sin(x+y)=

07-20: sin(x+y)



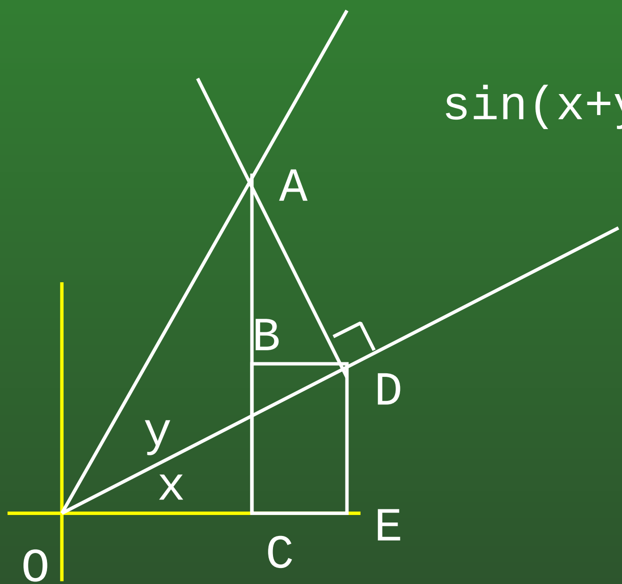
sin(x+y)= (AB + BC) / OA

07-21: cos(x+y)




```
sin(x+y)= (AB + BC) / OA
        = (AB + DE) / OA
        = AB / OA + DE / OA
```

07-22: sin(x+y)



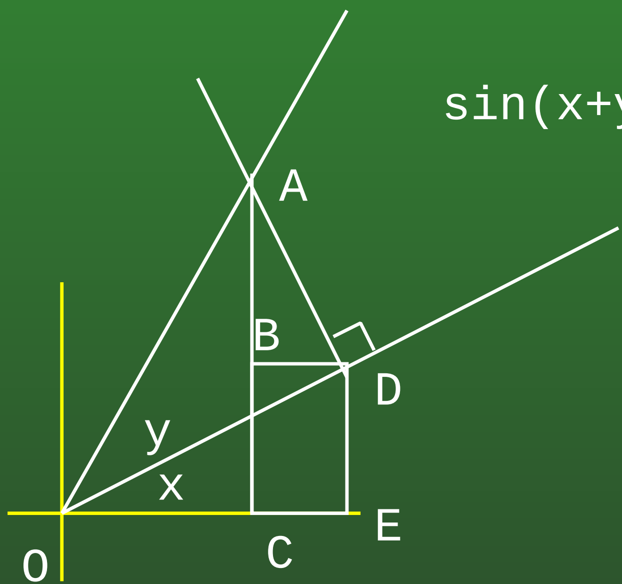
sin(x+y)= (AB + BC) / OA

= (AB + DE) / OA

= AB / OA + DE / OA

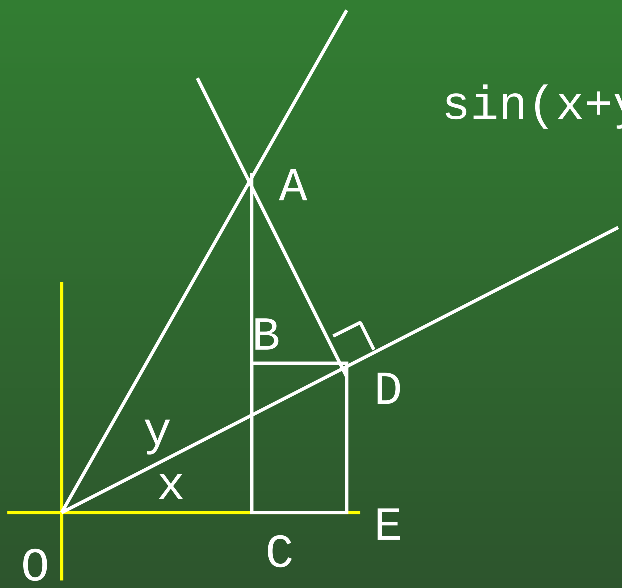
= (AB / OA) (AD / AD) + (DE / OA)(OD / OD)

07-23: sin(x+y)



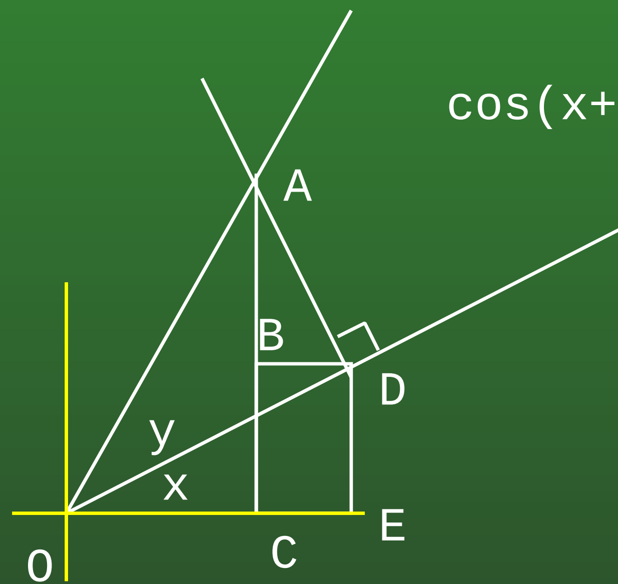
$$
\begin{aligned}
\sin(x+y) &= (AB + BC) / OA \\
&= (AB + DE) / OA \\
&= AB / OA + DE / OA \\
&= (AB / OA)(AD / AD) + (DE / OA)(OD / OD) \\
&= (AB / AD)(AD / OA) + (DE / OD)(OD / OA)
\end{aligned}
$$

07-24: sin(x+y)



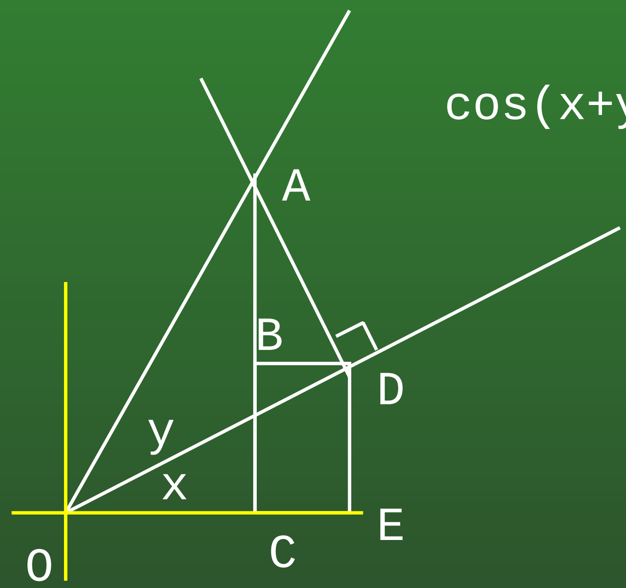

```
sin(x+y)= (AB + BC) / OA
        = (AB + DE) / OA
        = AB / OA + DE / OA
        = (AB / OA) (AD / AD) + (DE / OA)(OD / OD)
        = (AB / AD) (AD / OA) + (DE / OD)(OD / OA)
        = (cos x)(sin y) + (sin x)(cos y)
```

07-25: cos(x+y)



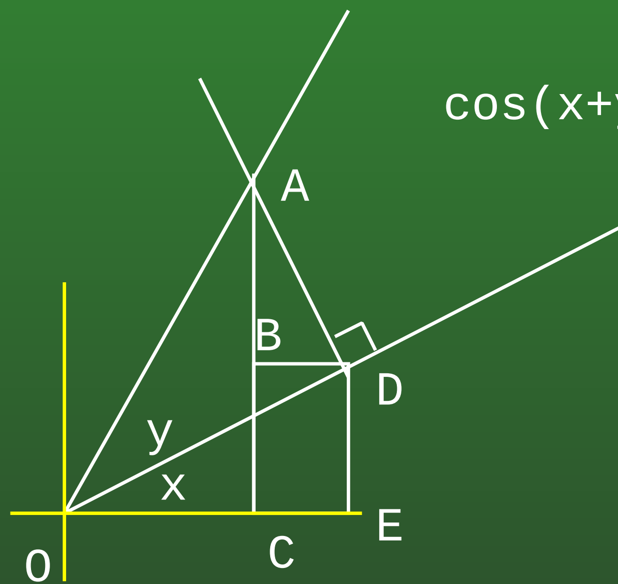
cos(x+y)=

07-26: cos(x+y)



cos(x+y)= OC / OA

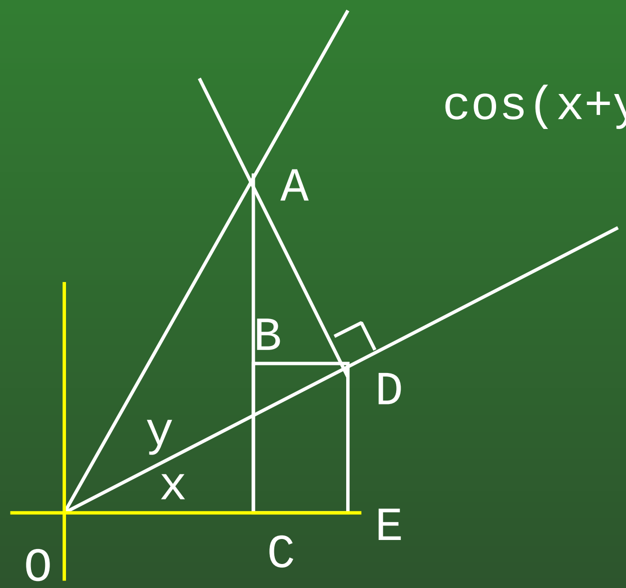
07-27: cos(x+y)



cos(x+y)= OC / OA

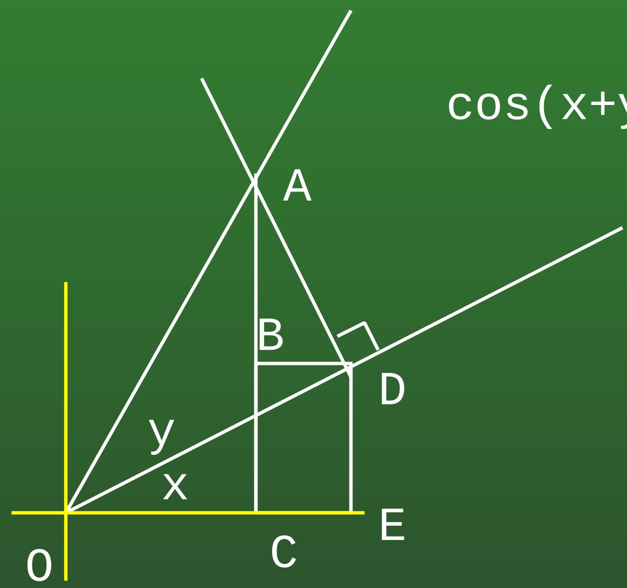
= (OE - CE) / OA

07-28: cos(x+y)



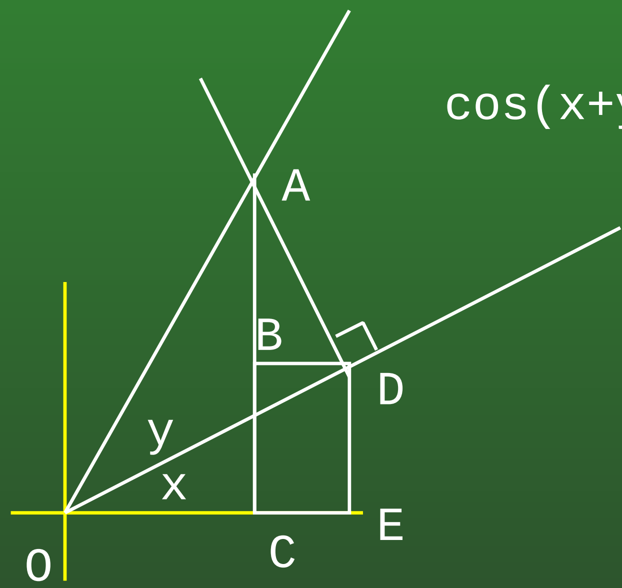

```
cos(x+y)= OC / OA
        = (OE - CE) / OA
        = (OE - BD) / OA
```

07-29: cos(x+y)



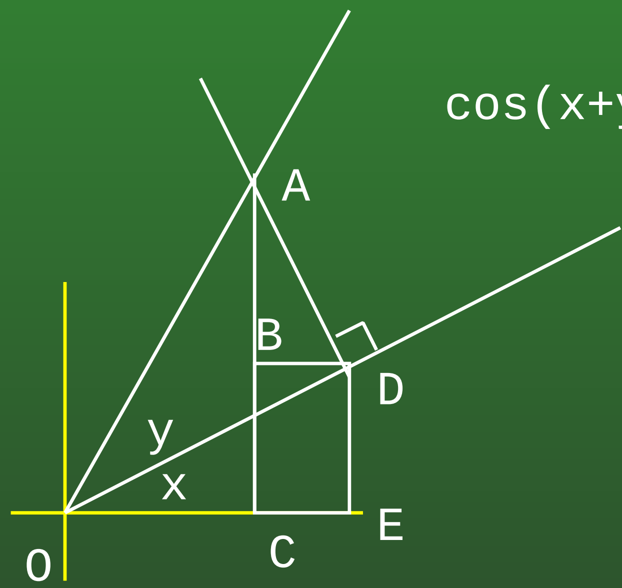

```
cos(x+y)= OC / OA
        = (OE - CE) / OA
        = (OE - BD) / OA
        = OE / OA - BD / OA
```

07-30: cos(x+y)



$$
\begin{aligned}
\cos(x+y) &= OC / OA \\
&= (OE - CE) / OA \\
&= (OE - BD) / OA \\
&= OE / OA - BD / OA \\
&= (OE/OA)(OD/OD) - (BD/OA)(AD/AD) \\
&= (OE/OD)(OD/OA) - (BD/AD)(AD/OA)
\end{aligned}
$$

07-31: cos(x+y)



$$\cos(x+y) = OC / OA$$
$$= (OE - CE) / OA$$
$$= (OE - BD) / OA$$
$$= OE / OA - BD / OA$$
$$= (OE/OA)(OD/OD) - (BD/OA)(AD/AD)$$
$$= (OE/OD)(OD/OA) - (BD/AD)(AD/OA)$$
$$= (\cos x)(\cos y) - (\sin x)(\sin y)$$

Back to Rotation!

- $x_{new} = r\cos(\Theta_1 + \Theta)$
- $x_{new} = r((\cos\Theta_1)(\cos\Theta) - (\sin\Theta_1)(\sin\Theta))$

07-33: Back to Rotation!

- $x_{new} = r\cos(\Theta_1 + \Theta)$
- $x_{new} = r((\cos\Theta_1)(\cos\Theta) - (\sin\Theta_1)(\sin\Theta))$
- $x_{new} = (r(\cos\Theta_1))\cos\Theta - (r(\sin\Theta_1))\sin\Theta$

07-34: Back to Rotation!

- $x_{new} = r\cos(\Theta_1 + \Theta)$
- $x_{new} = r((\cos\Theta_1)(\cos\Theta) - (\sin\Theta_1)(\sin\Theta))$
- $x_{new} = (r(\cos\Theta_1))\cos\Theta - (r(\sin\Theta_1))\sin\Theta$
- $x_{new} = x\cos\Theta - y\sin\Theta$

07-35: Back to Rotation!

- $y_{new} = r\sin(\Theta_1 + \Theta)$
- $y_{new} = r((\cos\Theta_1)(\sin\Theta) + (\sin\Theta_1)(\cos\Theta))$

07-36: Back to Rotation!

- $y_{new} = r\sin(\Theta_1 + \Theta)$
- $y_{new} = r((\cos\Theta_1)(\sin\Theta) + (\sin\Theta_1)(\cos\Theta))$
- $y_{new} = (r(\cos\Theta_1))\sin\Theta + (r(\sin\Theta_1))\cos\Theta$

07-37: Back to Rotation!

- $y_{new} = r\sin(\Theta_1 + \Theta)$
- $y_{new} = r((\cos\Theta_1)(\sin\Theta) + (\sin\Theta_1)(\cos\Theta))$
- $y_{new} = (r(\cos\Theta_1))\sin\Theta + (r(\sin\Theta_1))\cos\Theta$
- $y_{new} = x\sin\Theta + y\cos\Theta$

07-38: Back to Rotation!

- Given a point (x, y), we can rotate it around the origin as follows:
  - $x_{new} = x \cos \Theta - y \sin \Theta$
  - $y_{new} = x \sin \Theta + y \cos \Theta$
- We can do this with a matrix multiplication

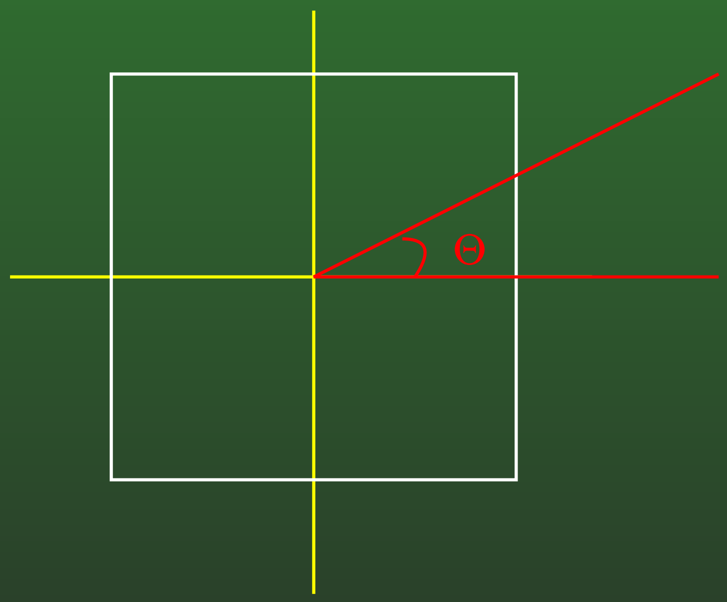
07-39: Back to Rotation!

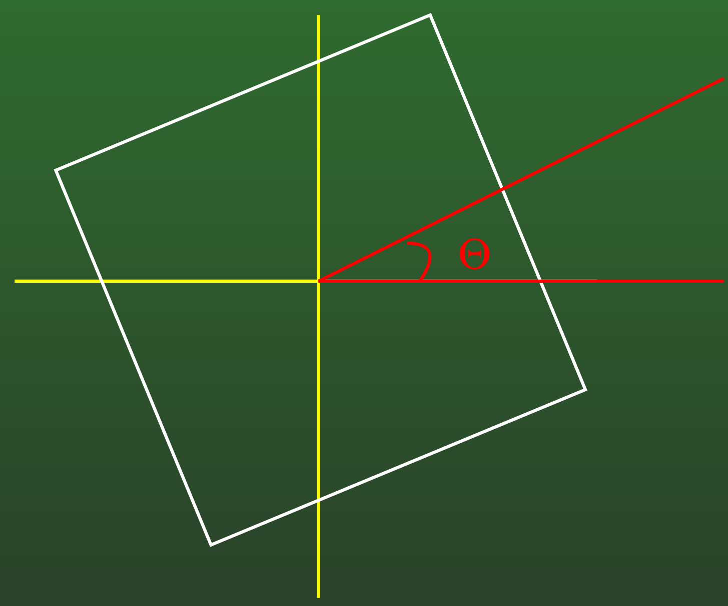
$$[x, y] \begin{bmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{bmatrix}$$

$$= [x\cos\Theta - y\sin\Theta, x\sin\Theta + y\cos\Theta]$$

07-40: Rotating Objects

- Polygon, consisting of a list of points
- Rotate the polygon by angle Θ around origin





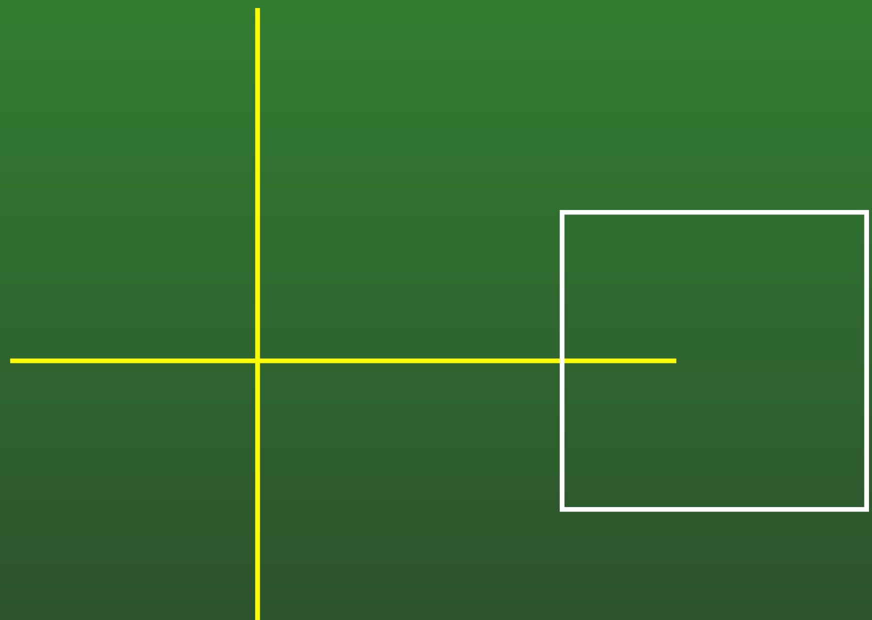
07-41: Rotating Objects

- Polygon, consisting of a list of points
- Rotate the polygon by angle Θ around origin
- Rotate each point individually around the origin
  - Done with a matrix multiplication

07-42: Rotating Objects

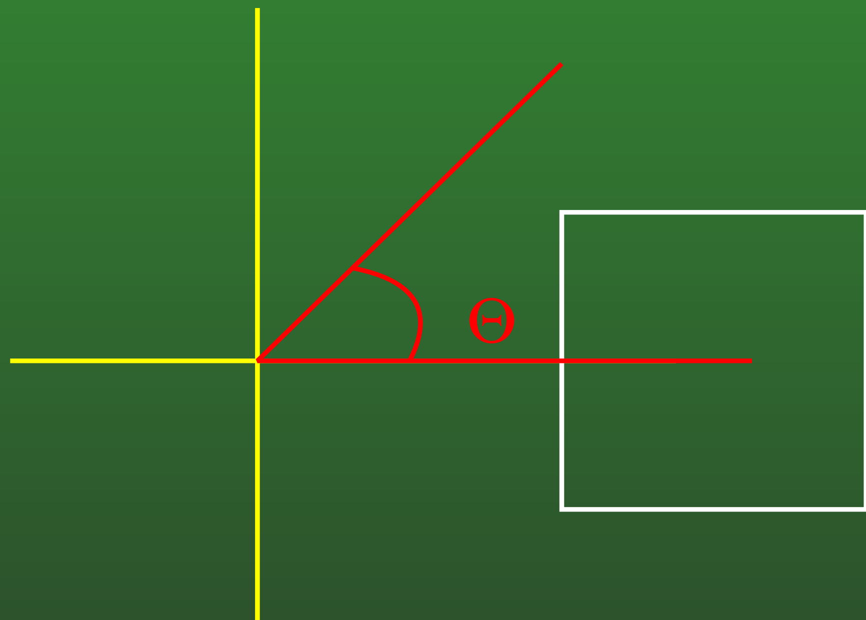
- Original Polygon: $p_0, p_1 \ldots p_n$
- New Polygon: $p_0 M, p_1 M, \ldots p_n M$
  - where $M = \begin{bmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{bmatrix}$

07-43: Rotating Objects



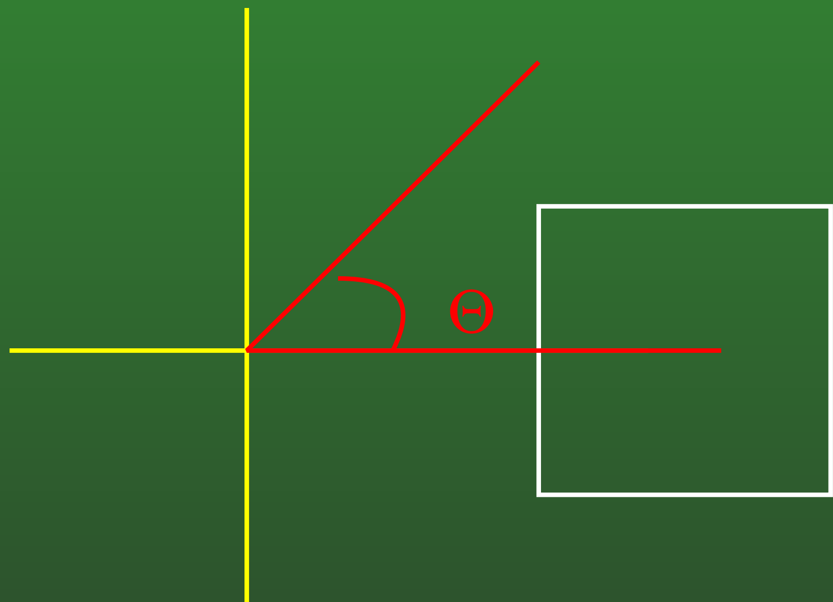
$$\begin{bmatrix} \cos 45 & \sin 45 \\ -\sin 45 & \cos 45 \end{bmatrix}$$

07-44: Rotating Objects

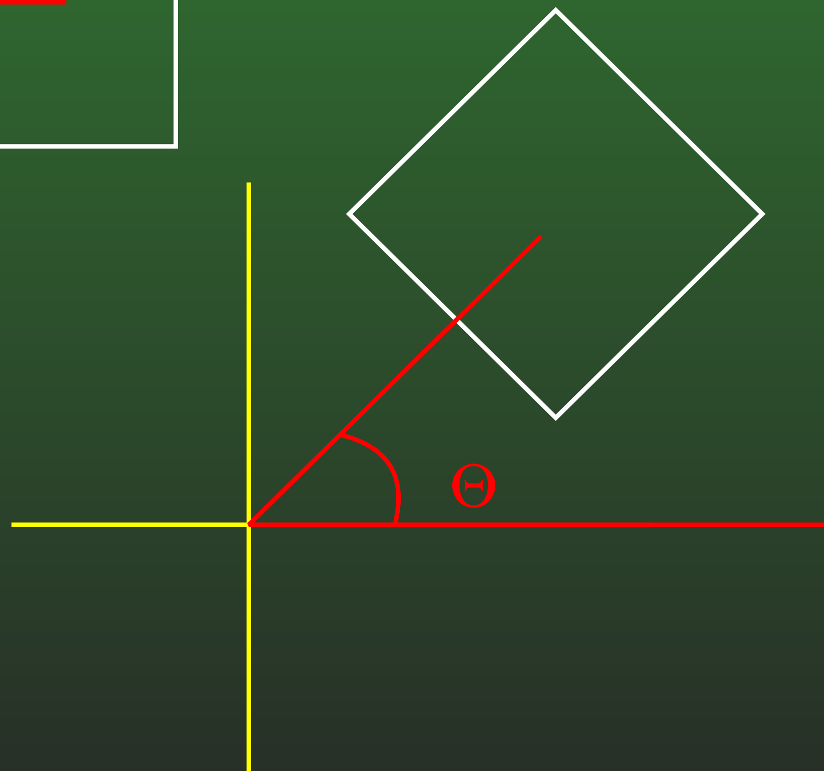


$$\begin{bmatrix} \cos 45 & \sin 45 \\ -\sin 45 & \cos 45 \end{bmatrix}$$

07-45: Rotating Objects

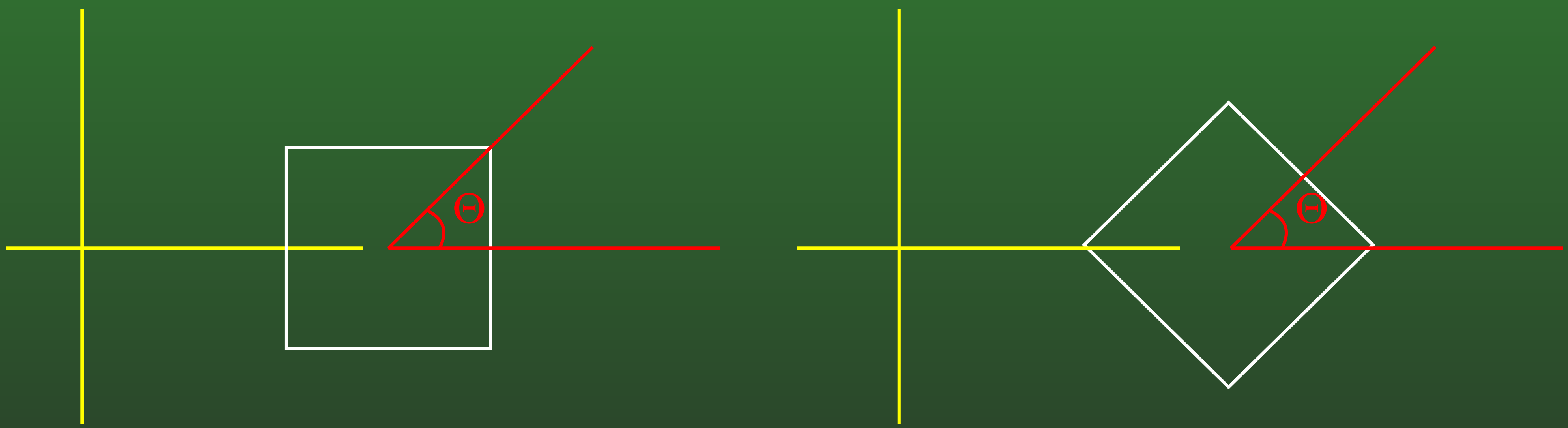


$$\begin{bmatrix} \cos 45 & \sin 45 \\ -\sin 45 & \cos 45 \end{bmatrix}$$



07-46: Rotating Objects

- How can we rotate an object around some point *other* than the origin?

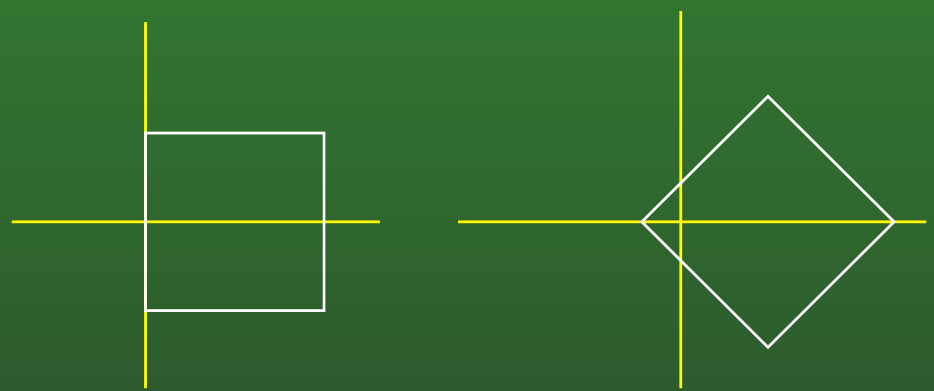


07-47: Rotating Objects

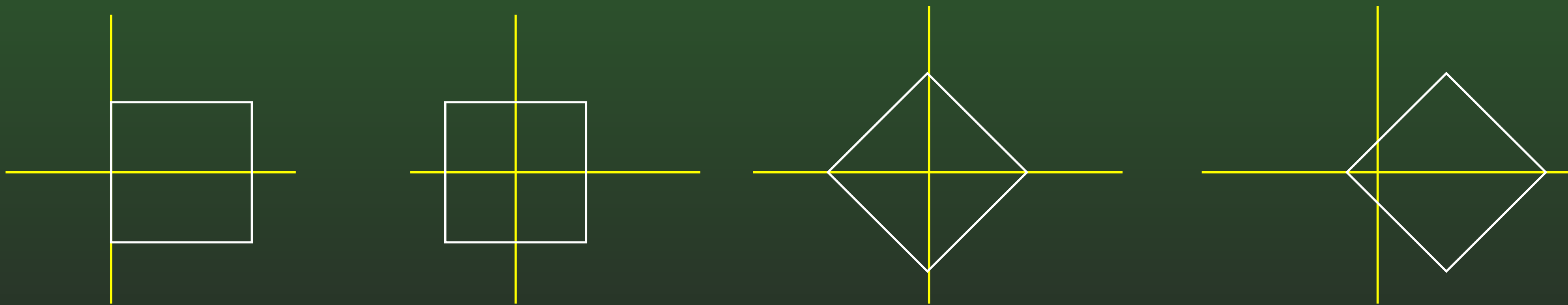
- How can we rotate an object around some point *other* than the origin?
  - Translate to the origin
  - Rotate
  - Translate back

07-48: Rotating Objects

Rotate π/4 around (1,0)



Translate to origin, Rotate π/4 around (0,0), Translate back



07-49: Multiple Cooridinate Systems

- World Space
- Camera (Screen) Space
- Inertial Space
- Object Space

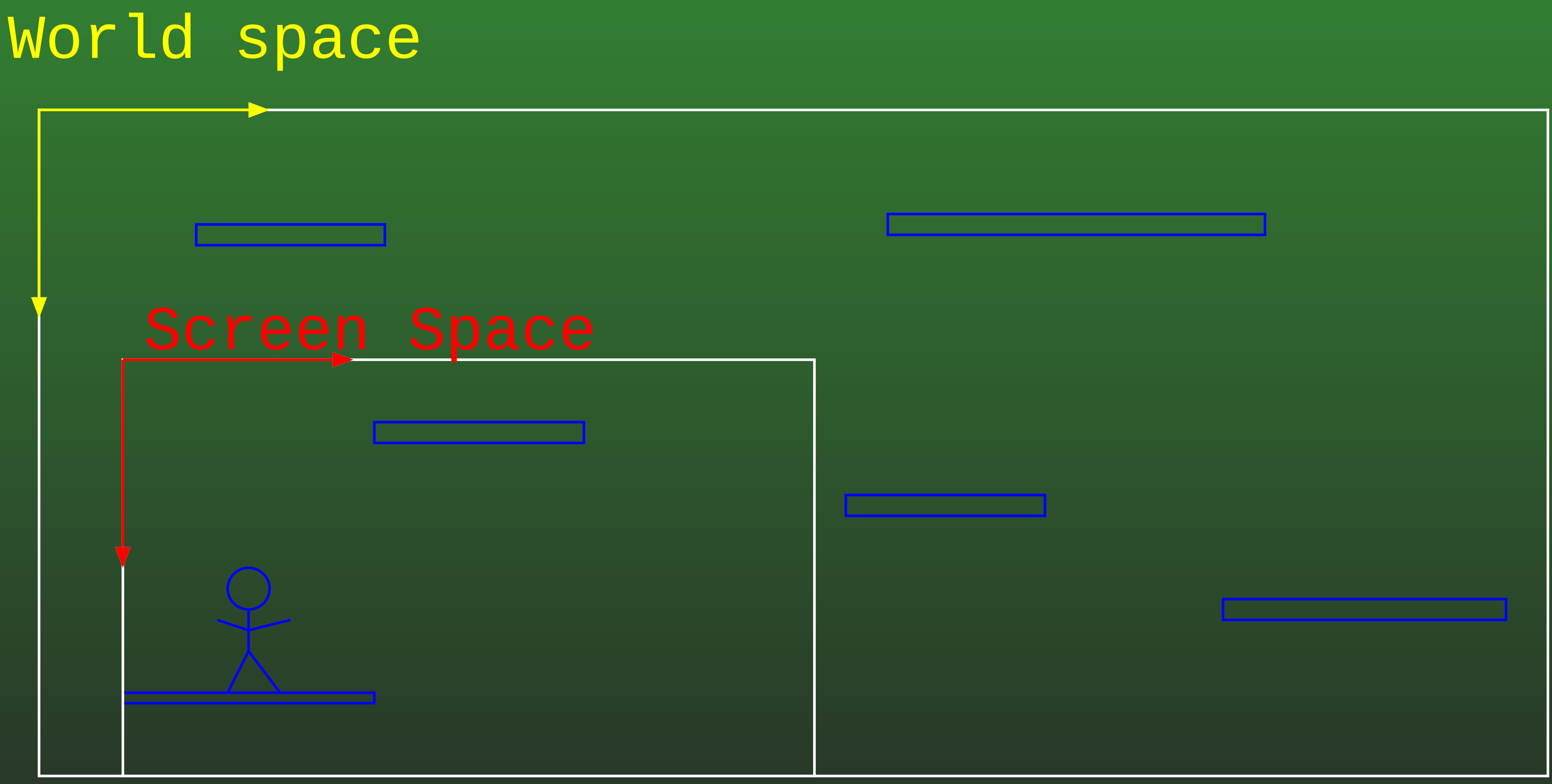
07-50: World Space

- Define an origin for your world
  - Could be in the middle of your world, in one corner, etc
- Define each object's position in the world as an offset from thhis point

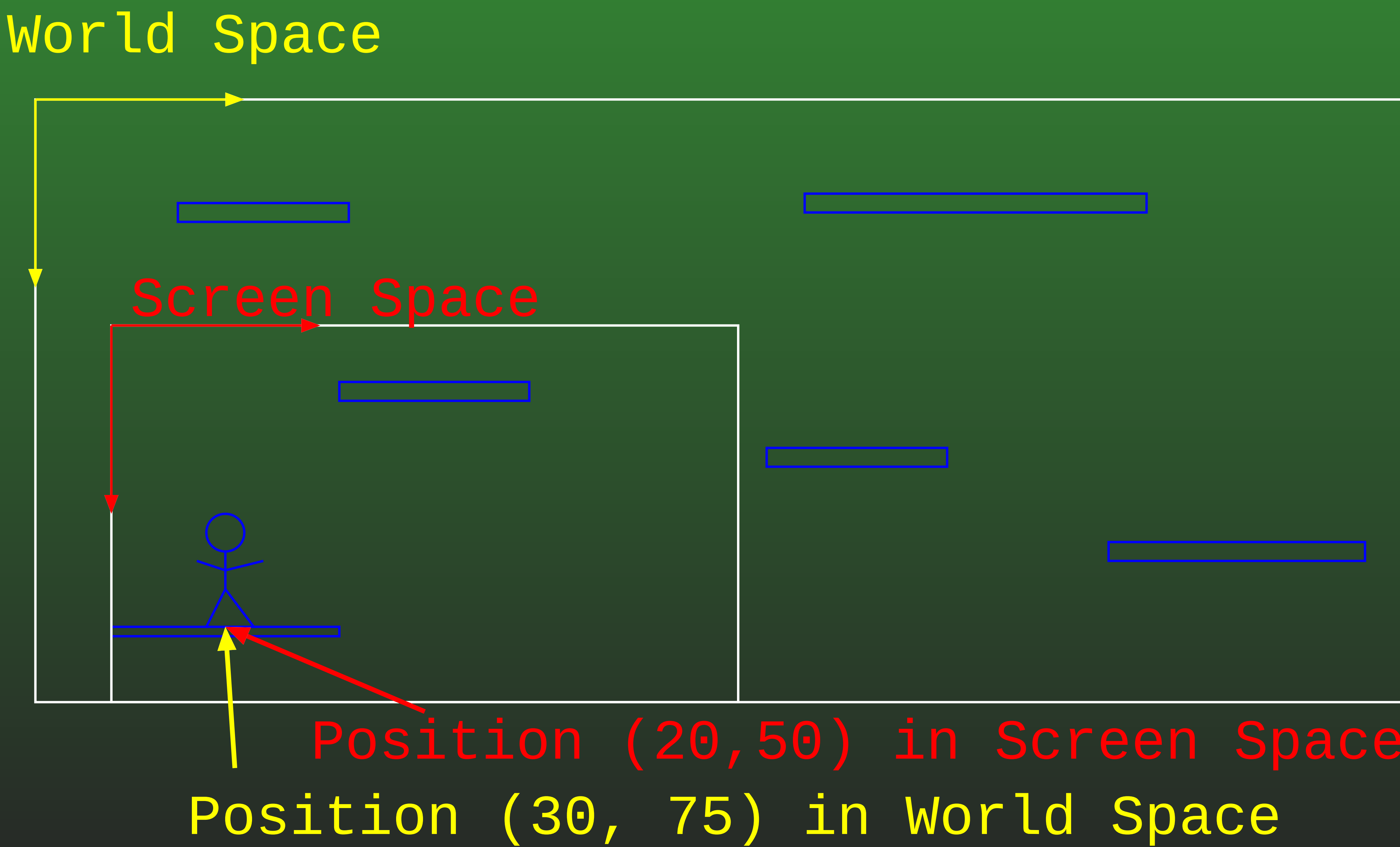
07-51: Camera (screen) Space

- Position that object appears on the screen
- Not always the same as world space!
  - Could have a much larger world, that screen scrolls across
  - “zoom in”

07-52: Camera (screen) Space



07-53: Camera (screen) Space



07-54: Camera (screen) Space

- You *could* calculate everything in Camera space ...
  - Moving camera becomes difficult – need to move all objects in the world along with the camrea
  - Objects are moving on their own, need to combine movements
  - Zooming in becomes problematic

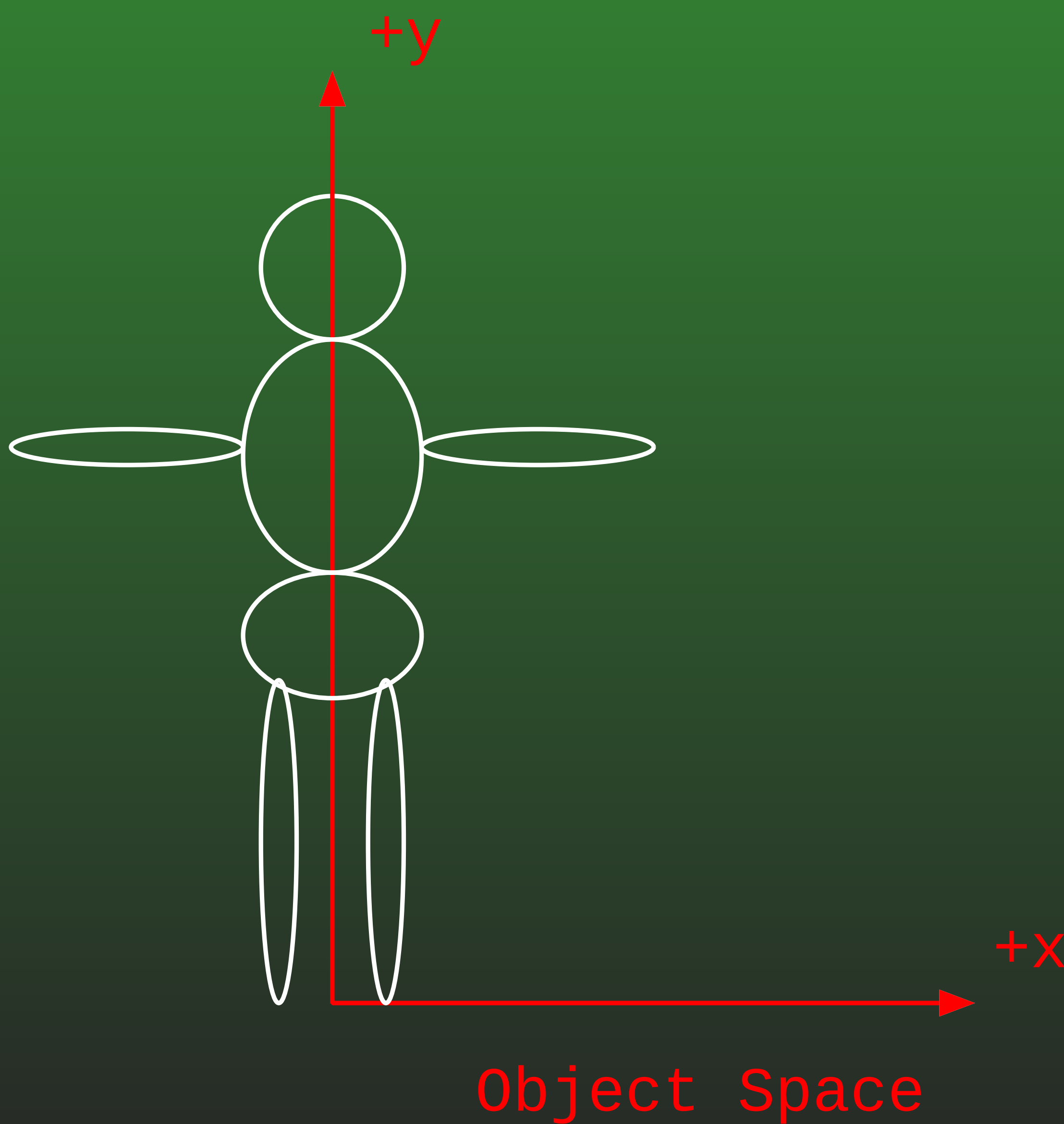
07-55: Object Space

- New Coordinate system based on the object
- Origin is at the base (or center) of the objet
- Axes are nicely aligned

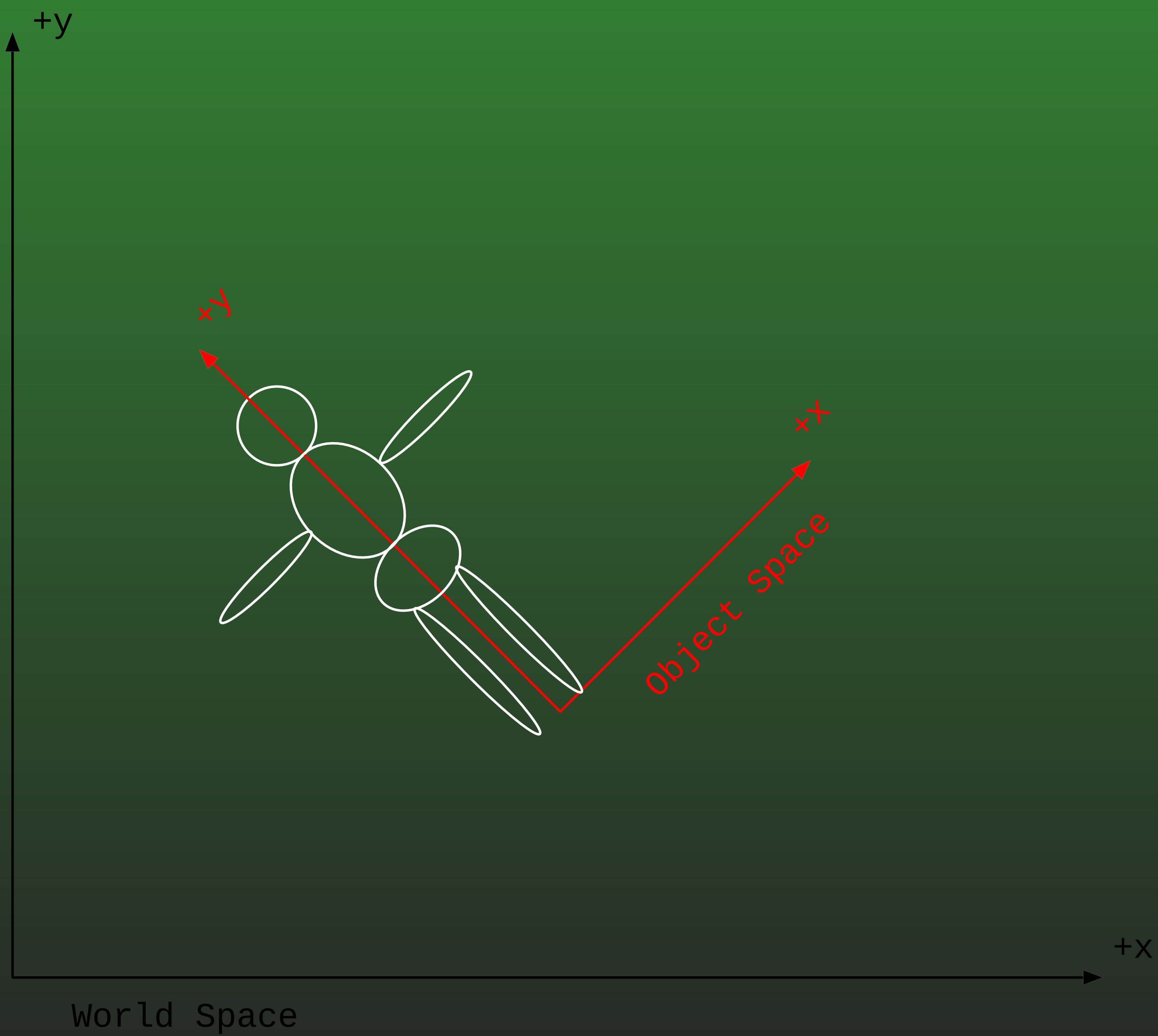
07-56: Intertial Space

- Halfway betwen object space and world space
- Axes parallel to world space
- Origin same as object space

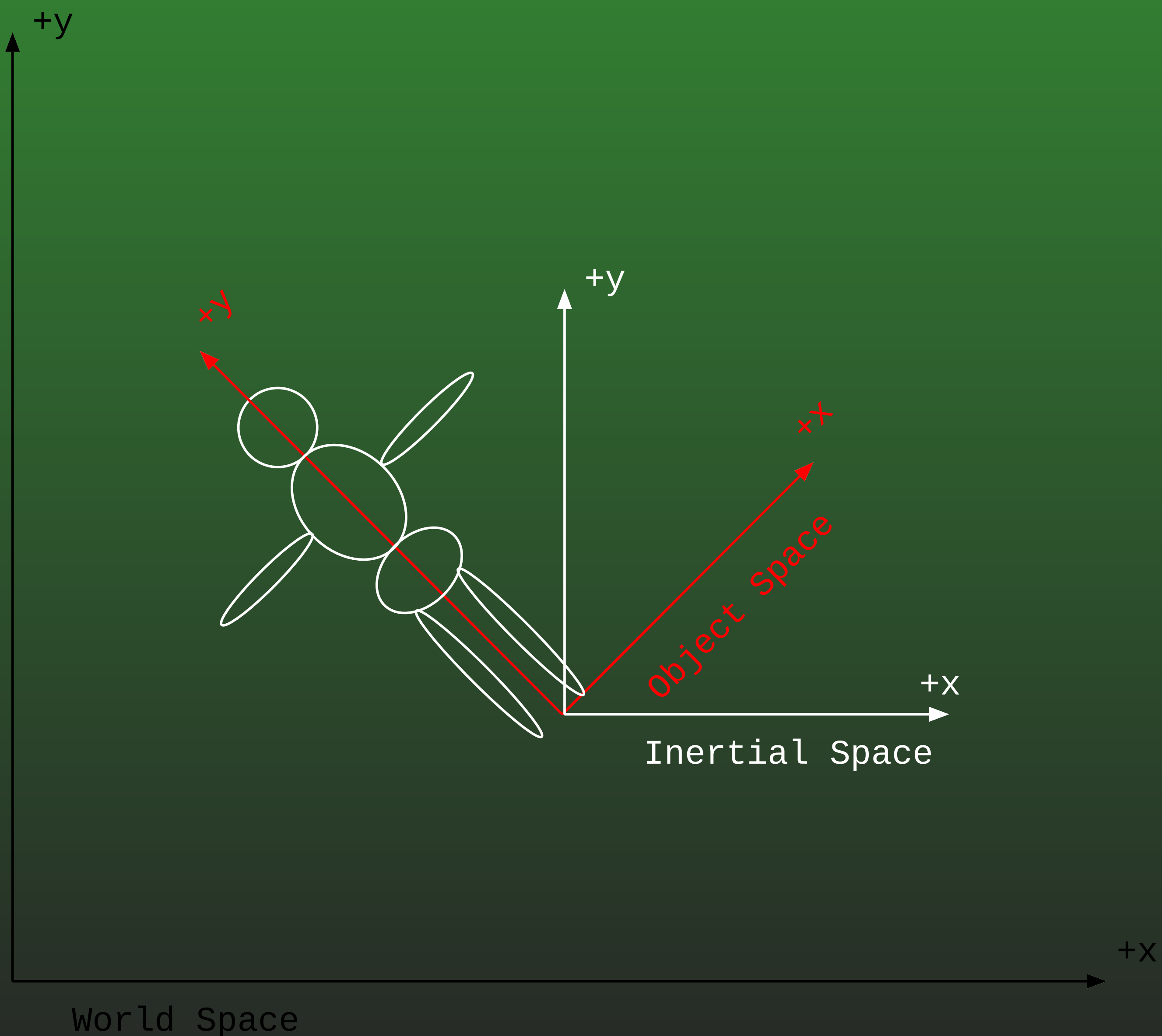
07-57: Inertial Space



07-58: Inertial Space



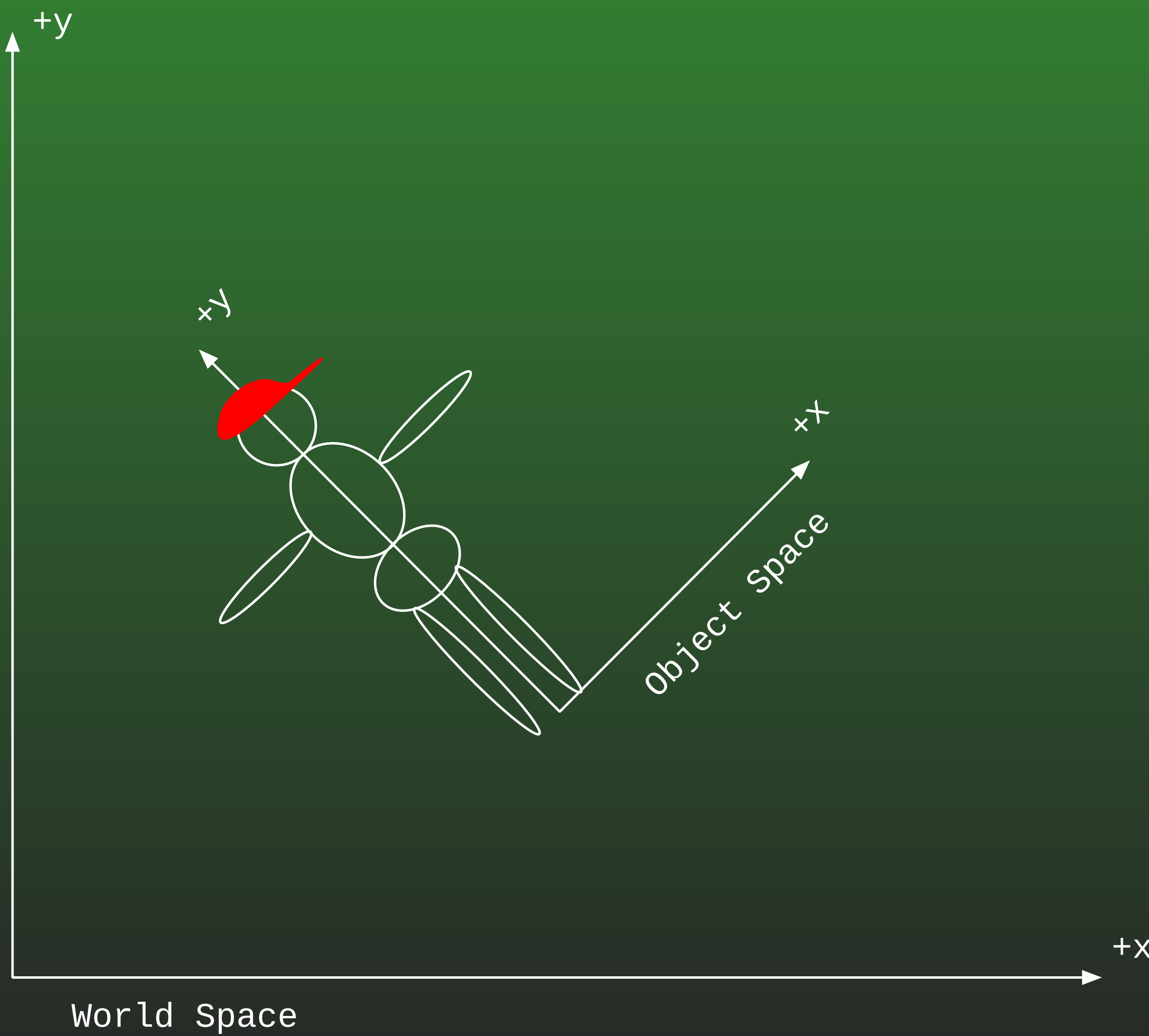
07-59: Inertial Space



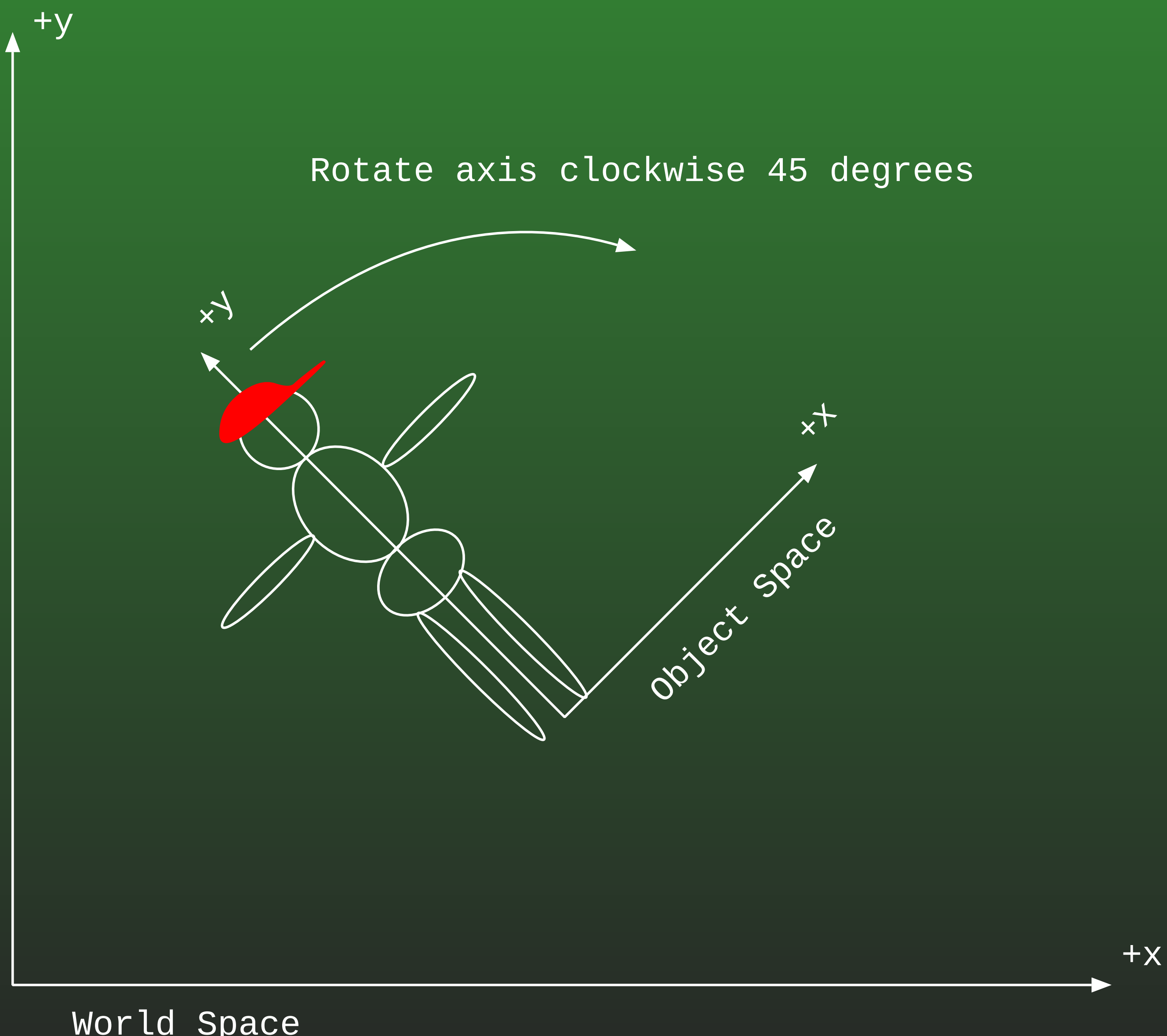
07-60: Changing Coordinate Spaces

- Our character is wearing a red hat
- The hat is at position (0,100) in object space
- What is the position of the hat in world space?
- To make life easier, we will think about rotating the axes, instead of moving the objects

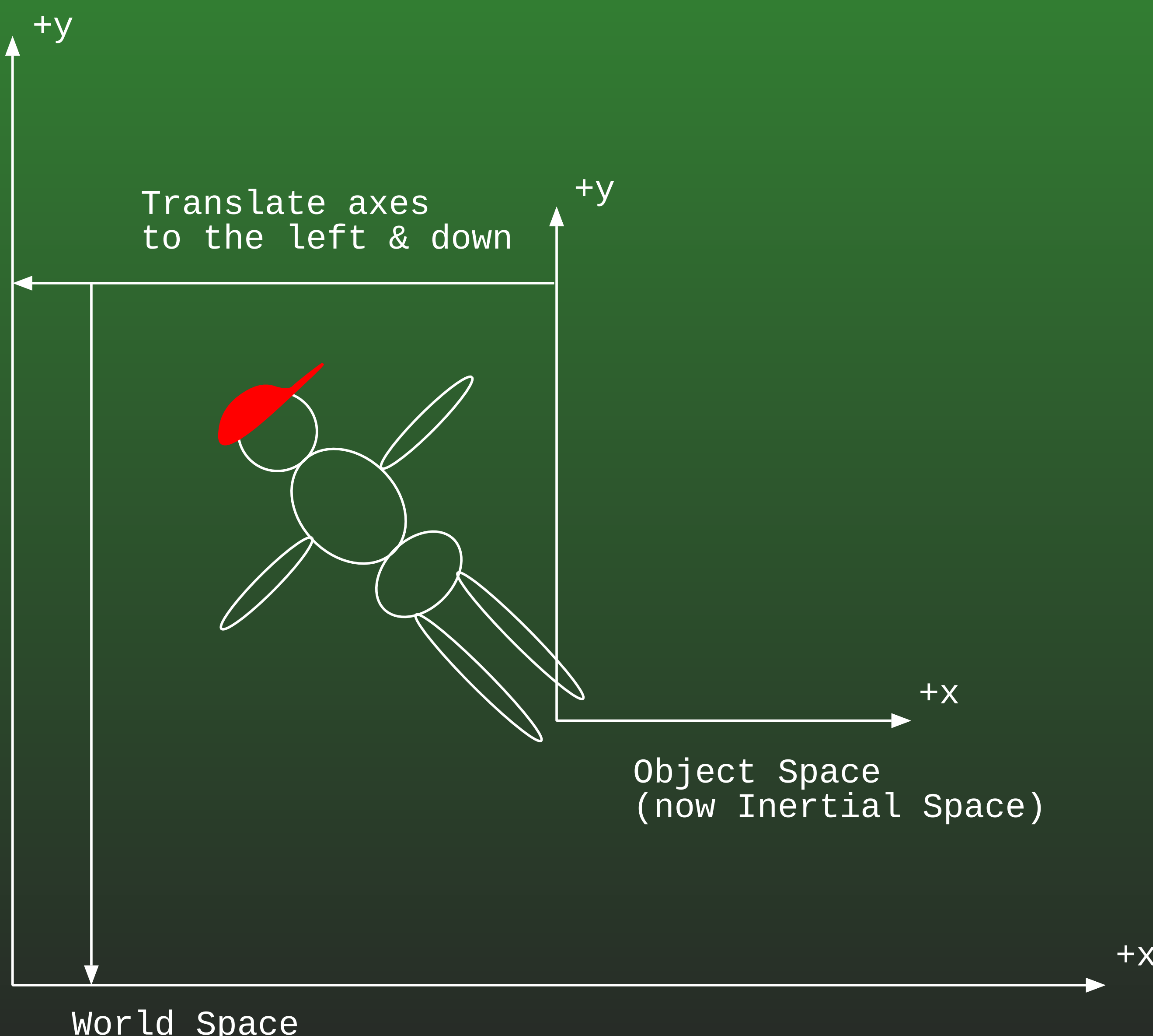
07-61: Changing Coordinate Spaces



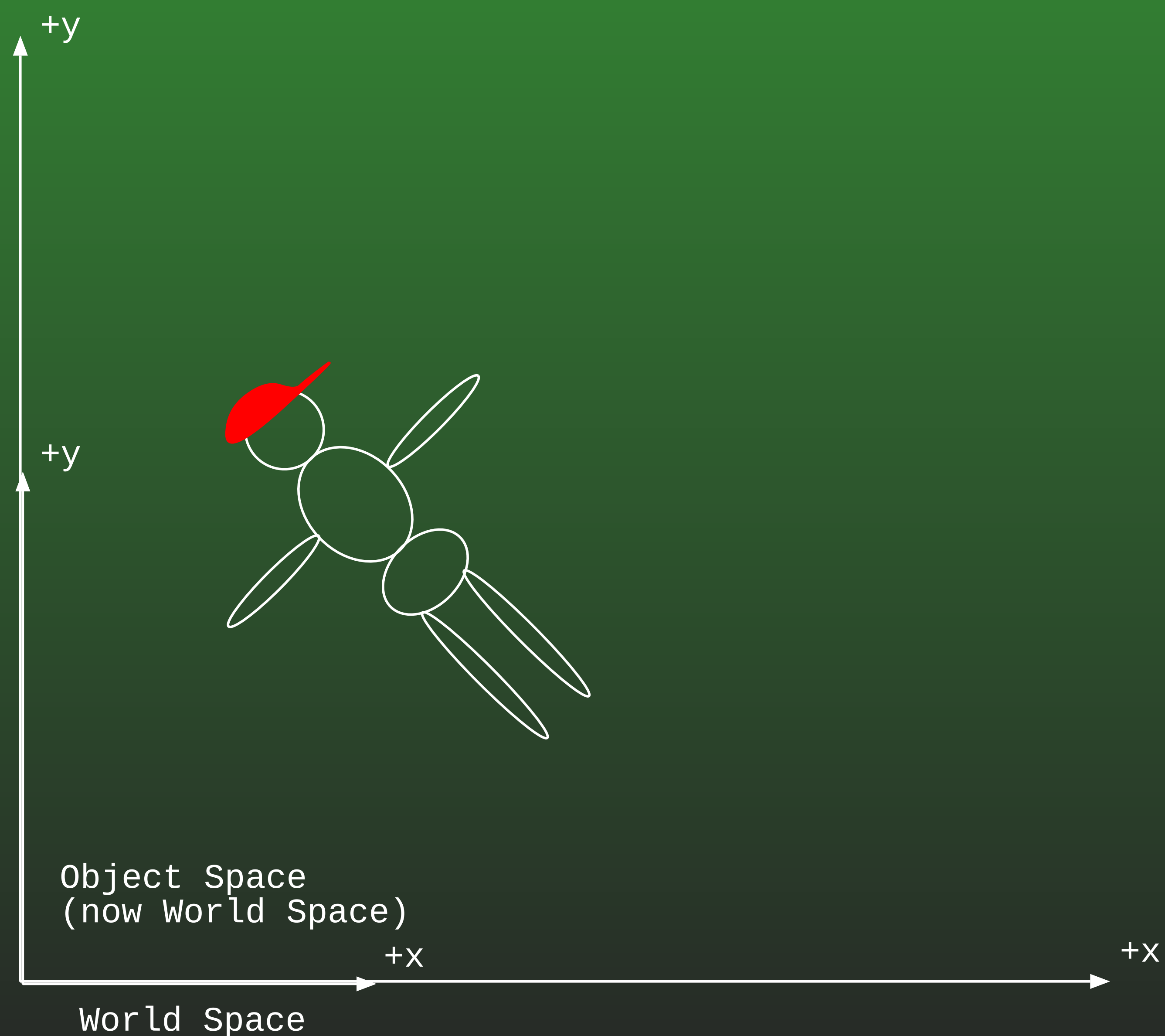
07-62: Changing Coordinate Spaces



07-63: Changing Coordinate Spaces



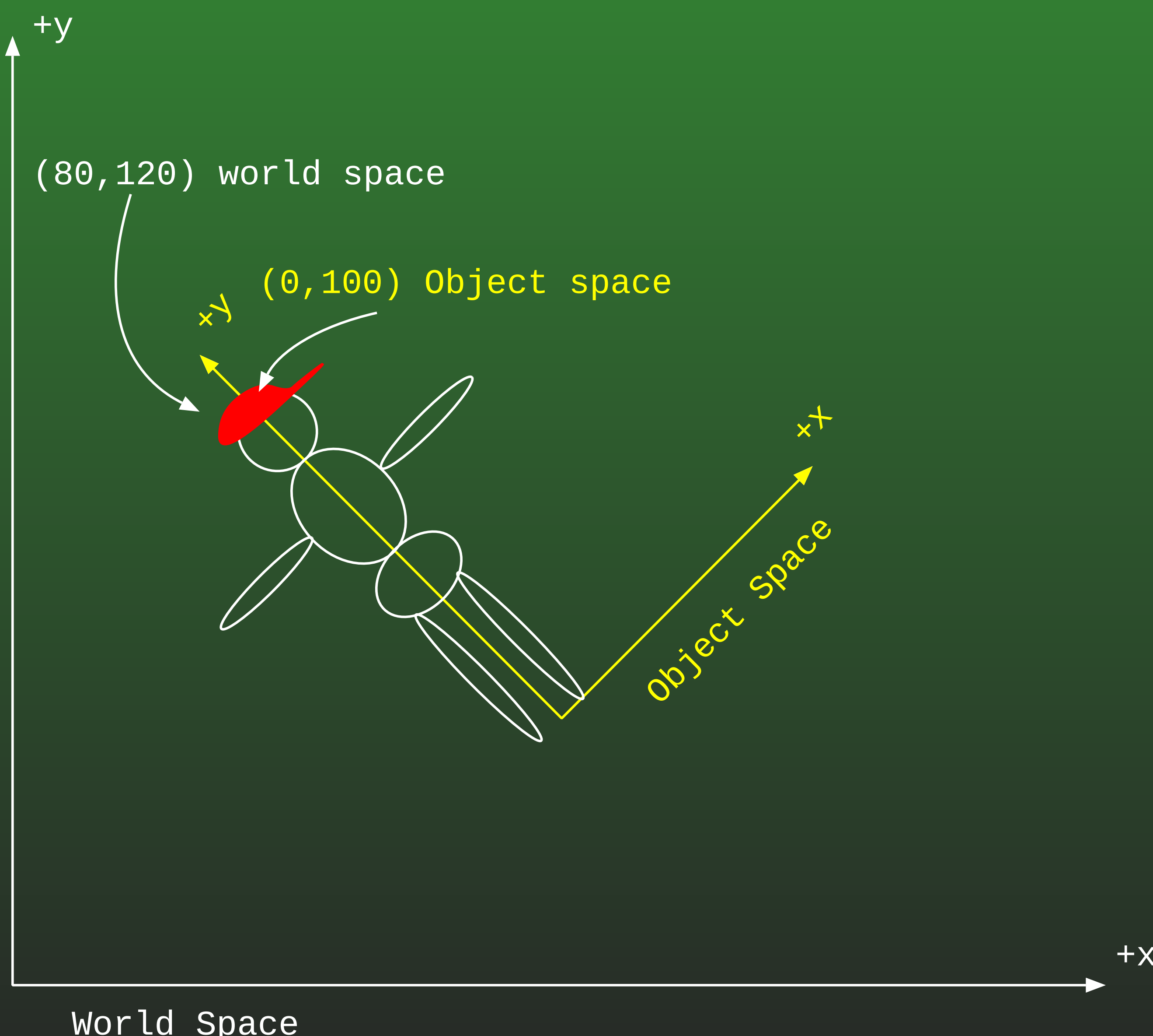
07-64: Changing Coordinate Spaces



07-65: Changing Coordinate Spaces

- Rotate axes to the left 45 degrees
  - Hat rotates the the right 45 degrees, from (0,100) to (-70, 70)
- Translate axes to the left 150, and down 50
  - Hat rotates to the right 150 and up 50, to (80, 120)

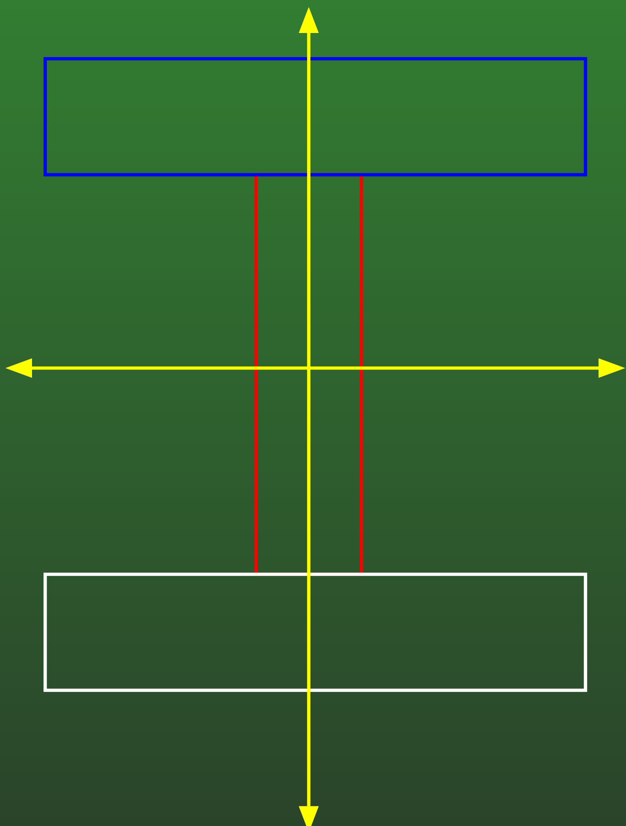
07-66: Changing Coordinate Spaces



07-67: Composite Objects

- More complicated object – made of multiple elements
  - Several polygons, circles
- Define in Object Space – origin the center of the object

07-68: Composite Objects

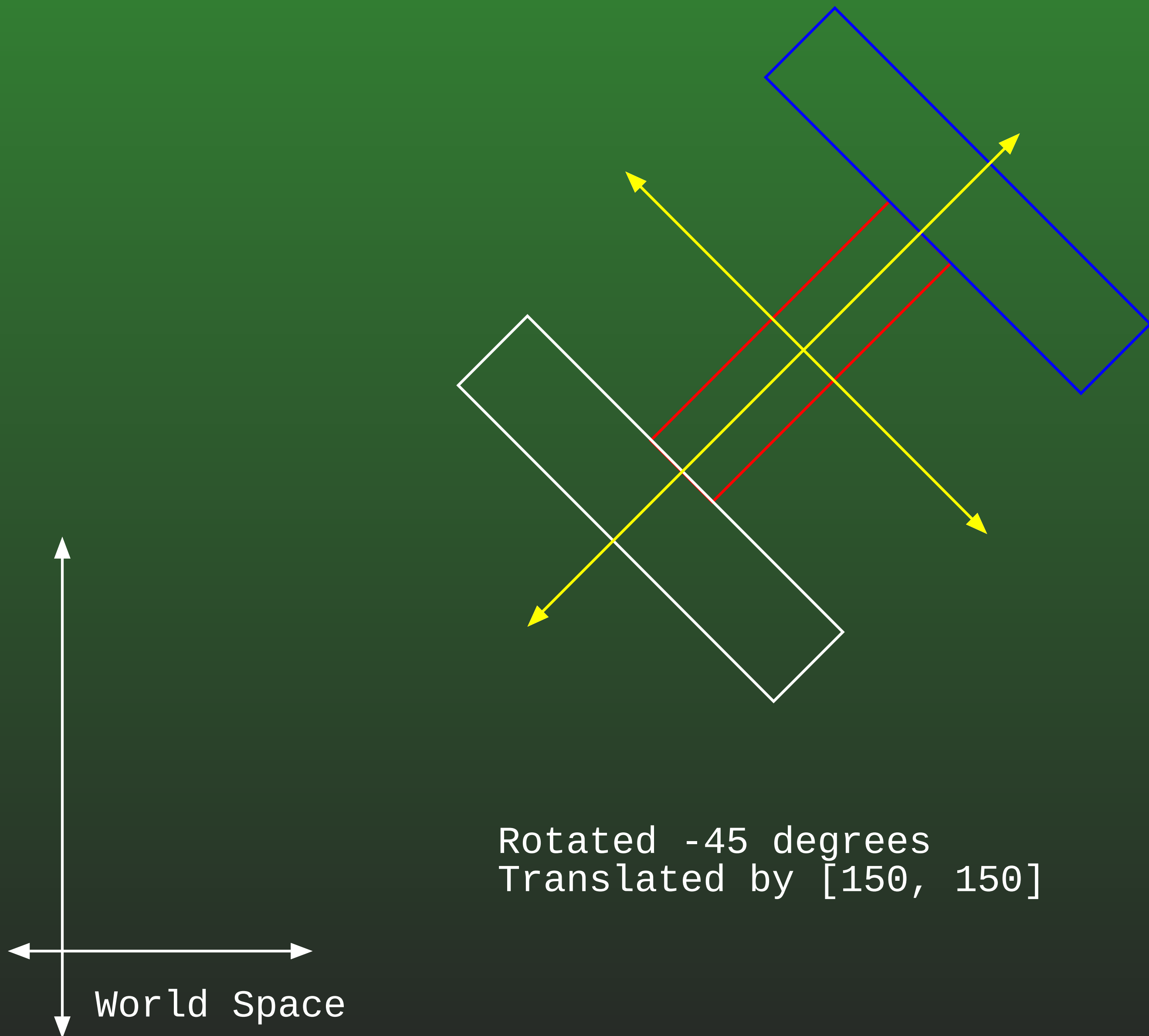


Poly1: (-30,30), (30,30), (30,20), (-30,20)

Poly2: (-5,20), (5,20), (5,-20), (-5,-20)

Poly3: (-30,-20), (30,-20), (30,-30), (-30,-30)

07-69: Composite Objects



07-70: Composite Objects

- We store the rotation and translation of entire object
- When we want to know the points of any sub-object in world space
  - Doing collision, for instance
- Rotate and then translate the points of the subobject

Composite Objects

- Want to know the position of the 4 points of the blue rectangle in world space
- Local space positions are p_1, p_2, p_3, p_4
- World space postions are:

07-72: Composite Objects

- Want to know the position of the 4 points of the blue rectangle in world space
- Local space positions are p_1, p_2, p_3, p_4
- World space postions are:

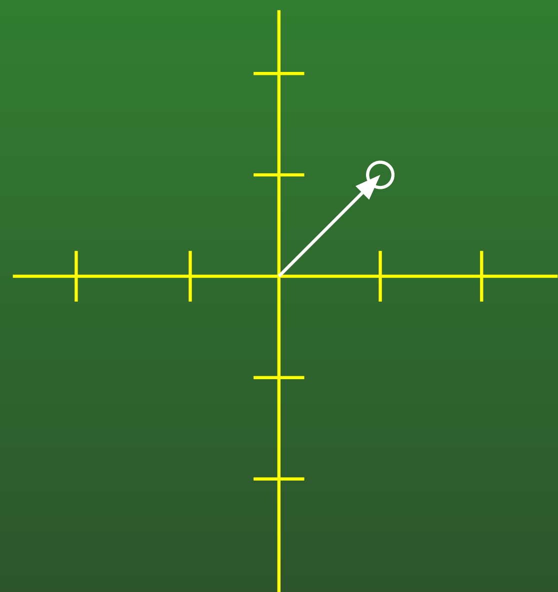
new p_1 = $p_1 \left[\begin{array}{cc} \cos(-45) & \sin(-45) \\ -\sin(-45) & \cos(-45) \end{array} \right] + [150, 150]$

new p_2 = $p_2 \left[\begin{array}{cc} \cos(-45) & \sin(-45) \\ -\sin(-45) & \cos(-45) \end{array} \right] + [150, 150]$

new p_3 = $p_3 \left[\begin{array}{cc} \cos(-45) & \sin(-45) \\ -\sin(-45) & \cos(-45) \end{array} \right] + [150, 150]$

... etc

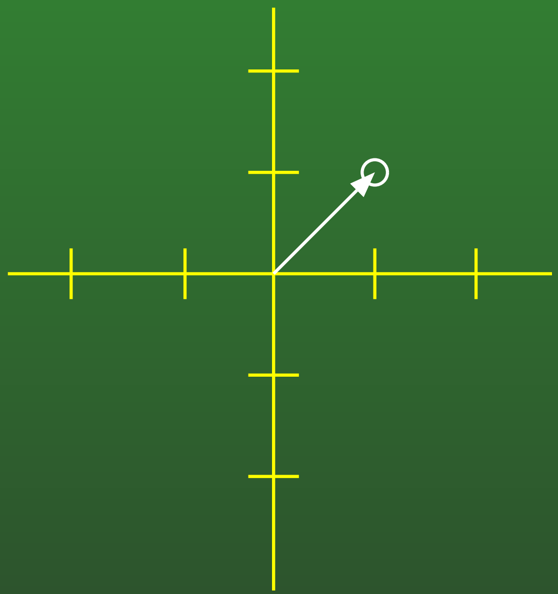
07-73: Other Transformations



Transform with:

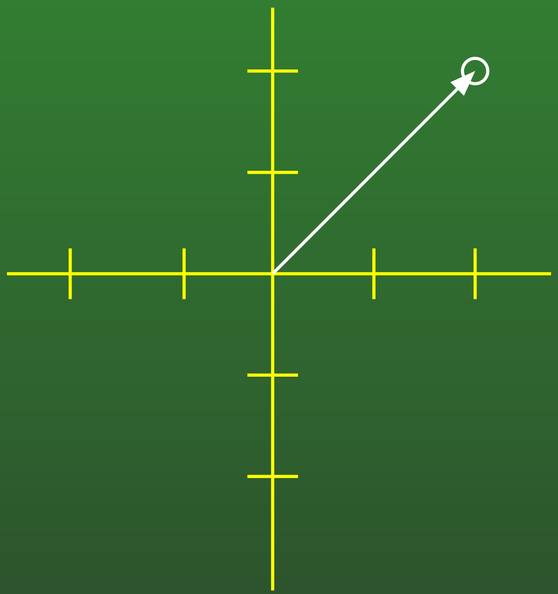
$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

07-74: Other Transformations

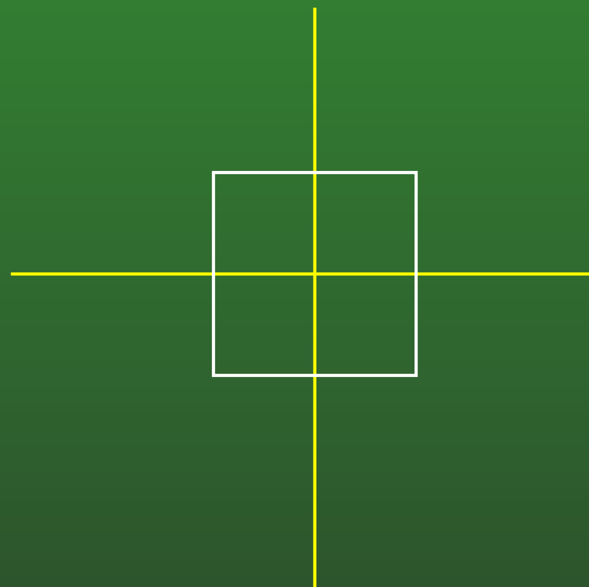


Transform with:

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$



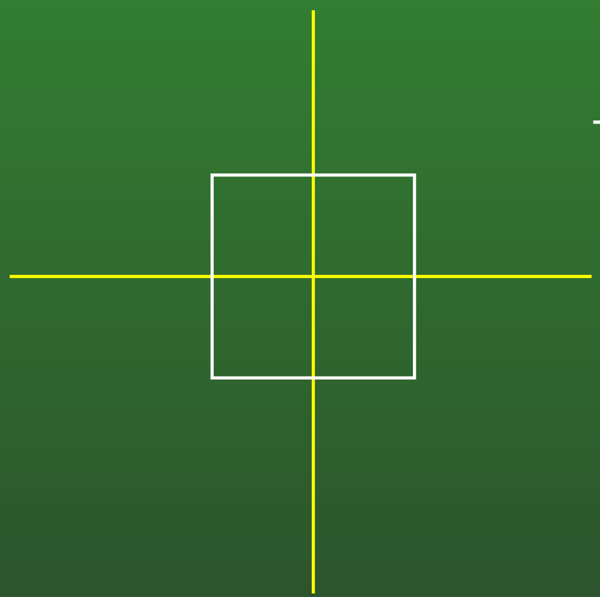
07-75: Other Transformations



Transform with:

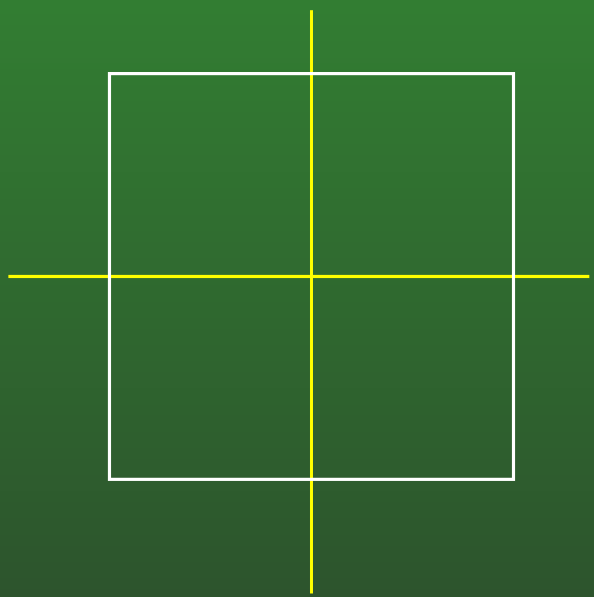
$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

07-76: Other Transformations



Transform with:

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$



07-77: Uniform Scaling

- A matrix of the form:

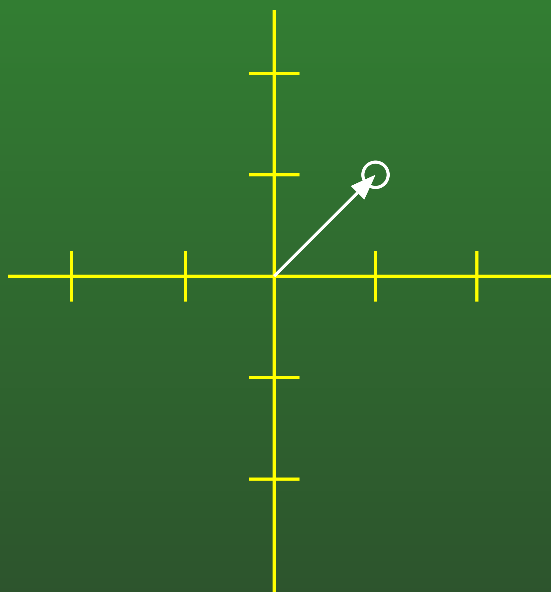
$$\begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$$

will uniformly scale an object. What happens if $k = 1$? $k > 1$? $0 < k < 1$?

07-78: Nonuniform Scaling

- A matrix can also be used to scale in different amounts on different axes.
- Object will be stretched / distorted

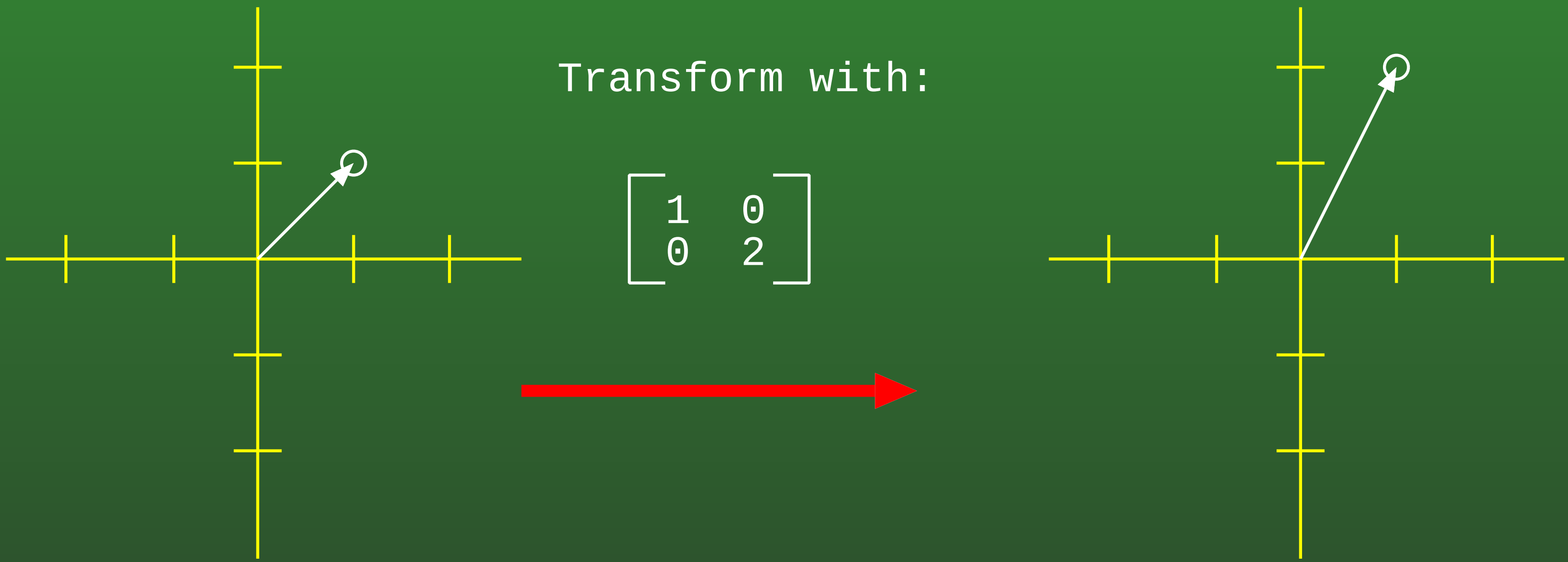
07-79: Nonuniform Scaling



Transform with:

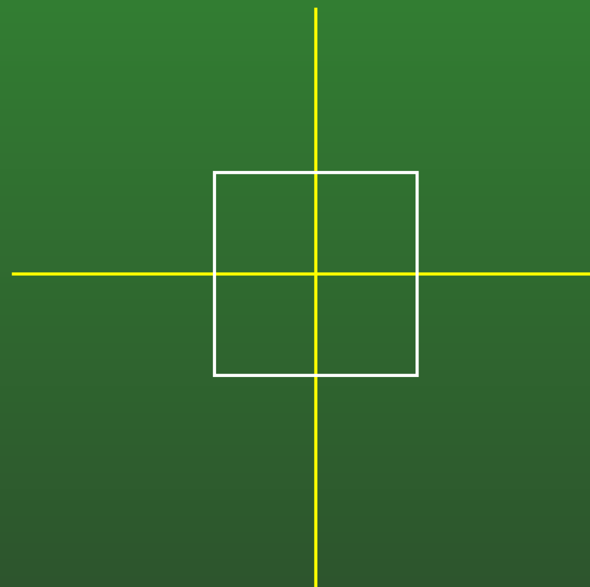
$$\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

07-80: Nonuniform Scaling



$$\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

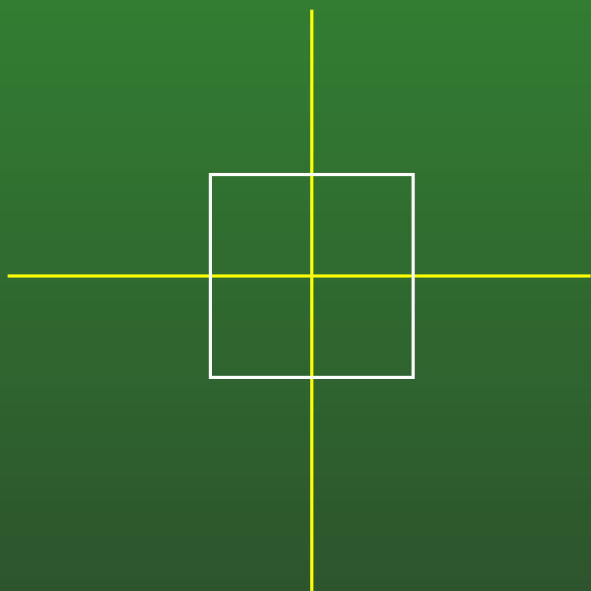
07-81: Nonuniform Scaling



Transform with:

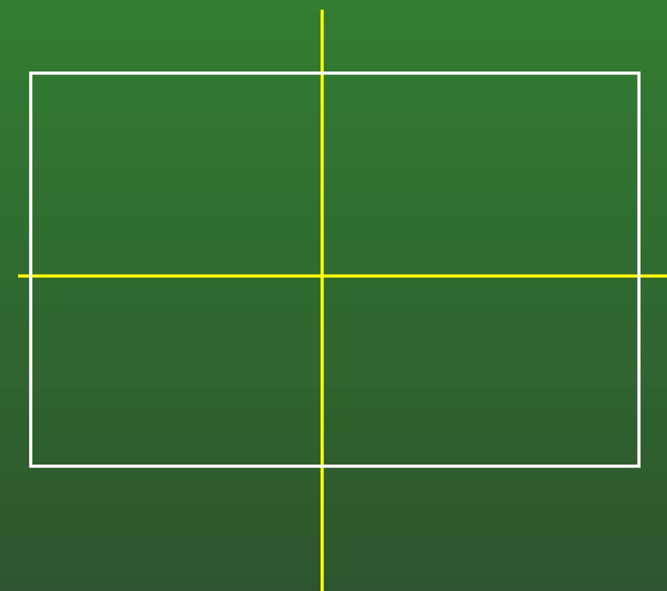
$$\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$$

07-82: Nonuniform Scaling



Transform with:

$$\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$$



Combining Transforms

- What would happen to a point that was transformed twice?

$$[x, y] \begin{bmatrix} \cos\Theta_1 & \sin\Theta_1 \\ -\sin\Theta_1 & \cos\Theta_1 \end{bmatrix} \begin{bmatrix} \cos\Theta_2 & \sin\Theta_2 \\ -\sin\Theta_2 & \cos\Theta_2 \end{bmatrix}$$

07-84: Combining Transforms

- What would happen to a point that was transformed twice?

$$\left([x,y]\begin{bmatrix}\cos\Theta_1 & \sin\Theta_1\\ -\sin\Theta_1 & \cos\Theta_1\end{bmatrix}\right)\begin{bmatrix}\cos\Theta_2 & \sin\Theta_2\\ -\sin\Theta_2 & \cos\Theta_2\end{bmatrix}=$$

$$[x,y]\left(\begin{bmatrix}\cos\Theta_1 & \sin\Theta_1\\ -\sin\Theta_1 & \cos\Theta_1\end{bmatrix}\begin{bmatrix}\cos\Theta_2 & \sin\Theta_2\\ -\sin\Theta_2 & \cos\Theta_2\end{bmatrix}\right)$$

07-85: Combining Transforms

$$\begin{bmatrix} \cos\Theta_1 & \sin\Theta_1 \\ -\sin\Theta_1 & \cos\Theta_1 \end{bmatrix} \begin{bmatrix} \cos\Theta_2 & \sin\Theta_2 \\ -\sin\Theta_2 & \cos\Theta_2 \end{bmatrix}$$

$$\begin{bmatrix} \cos\Theta_1\cos\Theta_2 - \sin\Theta_1\sin\Theta_2 & \cos\Theta_1\sin\Theta 2 + \sin\Theta_1\cos\Theta_2 \\ -\sin\Theta_1\cos\Theta_2 - \cos\Theta_1\sin\Theta_2 & -\sin\Theta_1\sin\Theta 2 + \cos\Theta_1\cos\Theta_2 \end{bmatrix}$$

07-86: Combining Transforms

$$\begin{bmatrix} \cos\Theta_1 & \sin\Theta_1 \\ -\sin\Theta_1 & \cos\Theta_1 \end{bmatrix} \begin{bmatrix} \cos\Theta_2 & \sin\Theta_2 \\ -\sin\Theta_2 & \cos\Theta_2 \end{bmatrix}$$

$$\begin{bmatrix} \cos\Theta_1\cos\Theta_2 - \sin\Theta_1\sin\Theta_2 & \cos\Theta_1\sin\Theta 2 + \sin\Theta_1\cos\Theta_2 \\ -\sin\Theta_1\cos\Theta_2 - \cos\Theta_1\sin\Theta_2 & -\sin\Theta_1\sin\Theta 2 + \cos\Theta_1\cos\Theta_2 \end{bmatrix}$$

$$\begin{bmatrix} \cos(\Theta_1 + \Theta_2) & \sin(\Theta_1 + \Theta_2) \\ -\sin(\Theta_1 + \Theta_2) & \cos(\Theta_1 + \Theta_2) \end{bmatrix}$$

07-87: Combining Transforms

- We can also combine scaling and rotating

$$[x, y] \begin{bmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$$

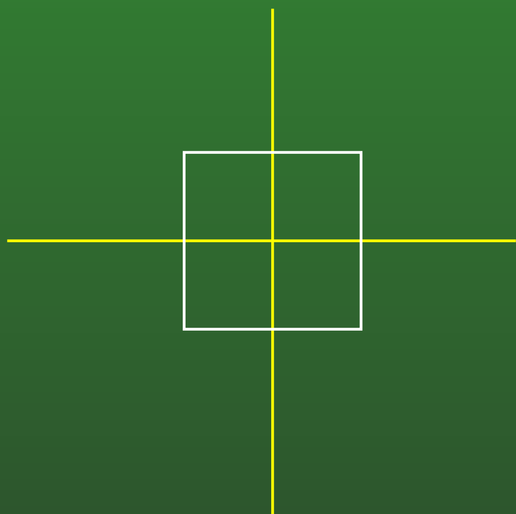
- With uniform scaling, we get the same result if we scale, then rotate as if we rotated, and then scaled.

07-88: Combining Transforms

$$\begin{bmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix} = \begin{bmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$$

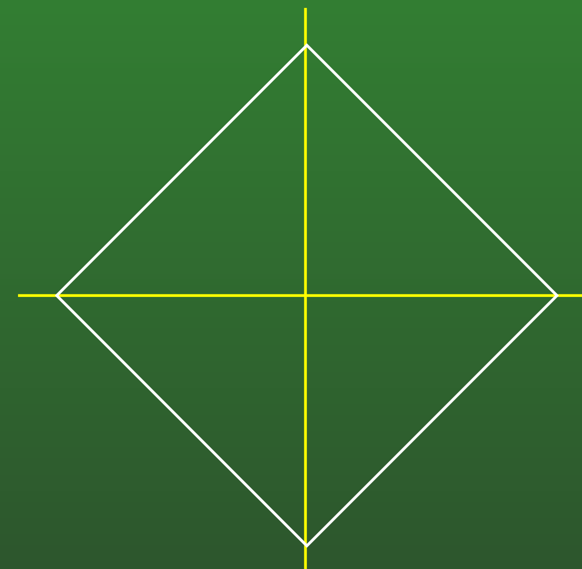
$$= \begin{bmatrix} k\cos\Theta & k\sin\Theta \\ -k\sin\Theta & k\cos\Theta \end{bmatrix}$$

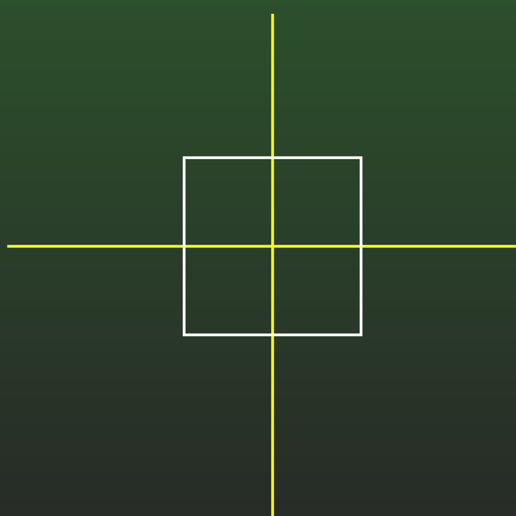
07-89: Combining Transforms



Transform with:

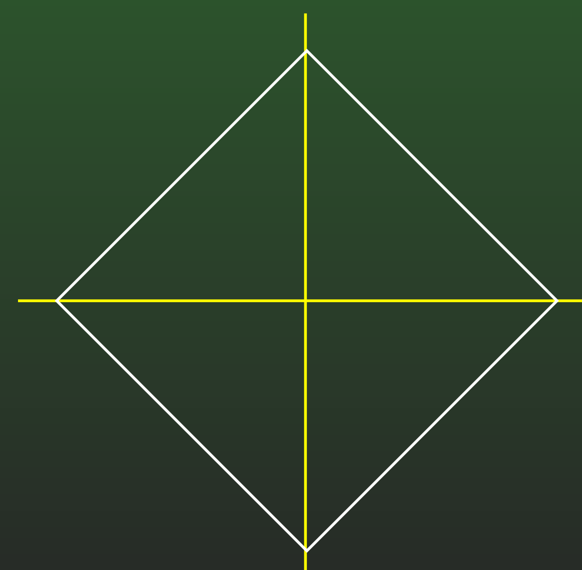
$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \cos 45 & \sin 45 \\ -\sin 45 & \cos 45 \end{bmatrix}$$



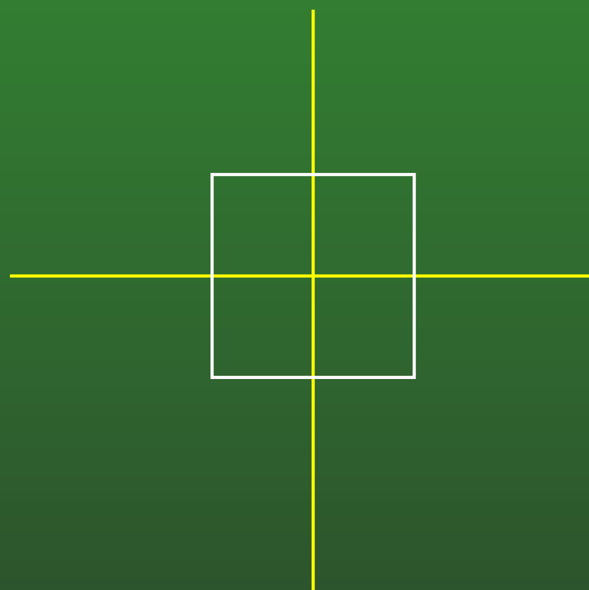


Transform with:

$$\begin{bmatrix} \cos 45 & \sin 45 \\ -\sin 45 & \cos 45 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

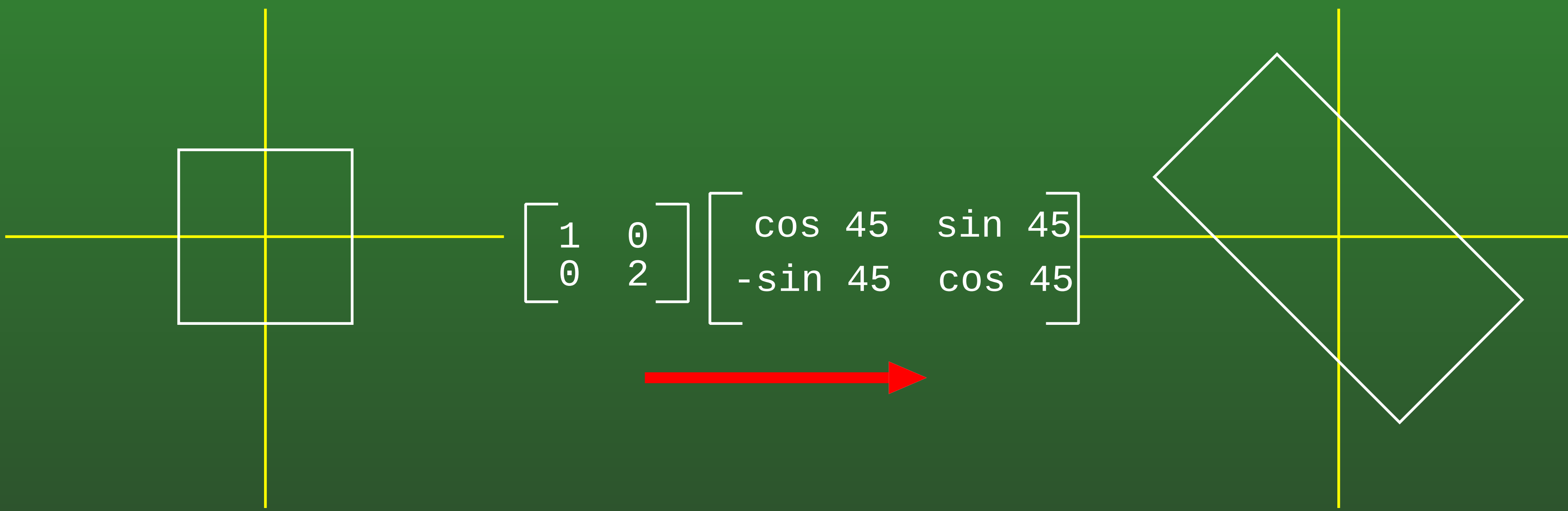


07-90: Non-Uniform Scale, Rotate

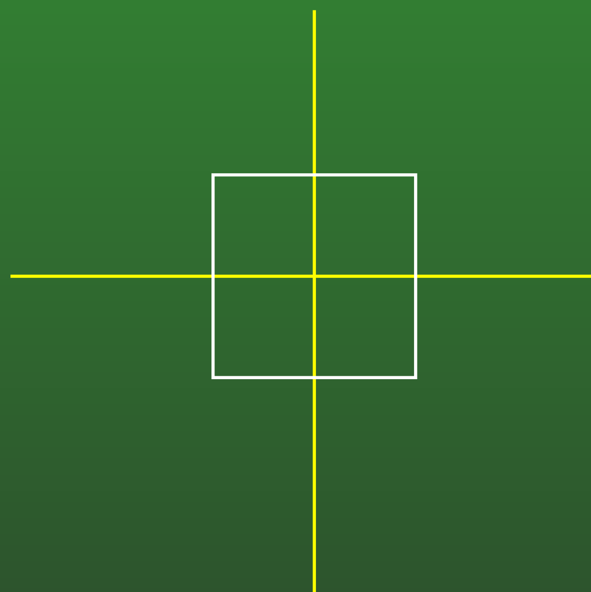


$$\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \cos 45 & \sin 45 \\ -\sin 45 & \cos 45 \end{bmatrix}$$

07-91: Non-Uniform Scale, Rotate

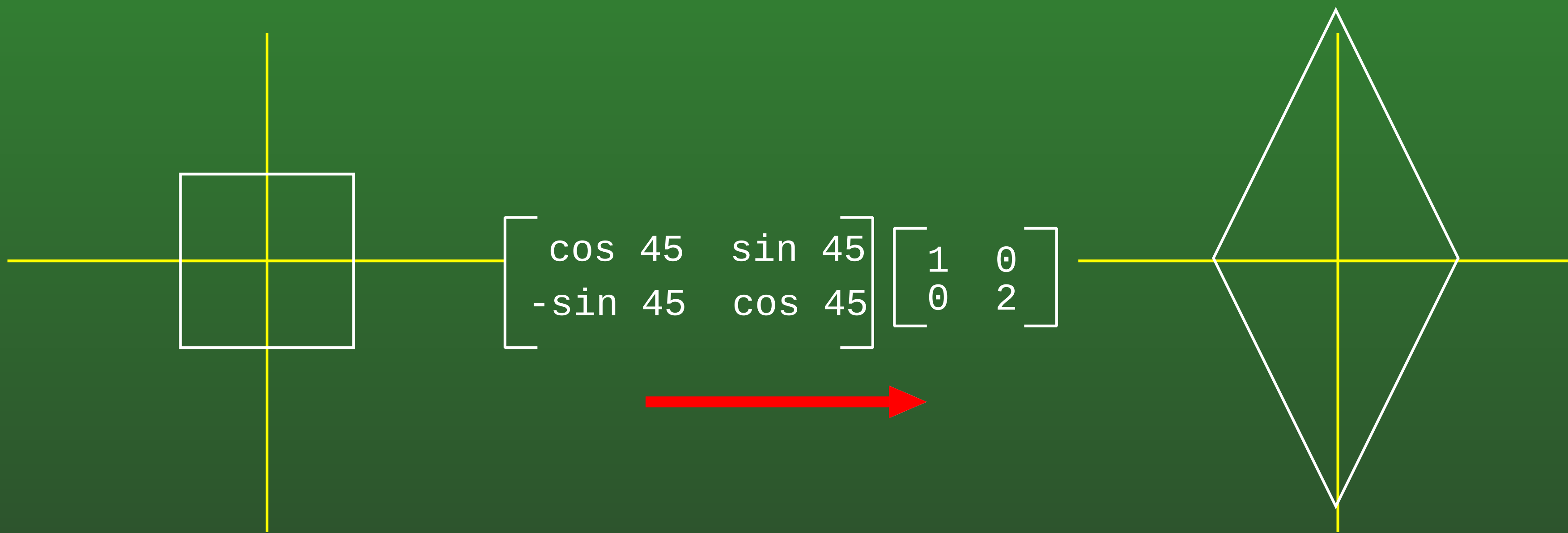


07-92: Non-Uniform Scale, Rotate



$$\begin{bmatrix} \cos 45 & \sin 45 \\ -\sin 45 & \cos 45 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

07-93: Non-Uniform Scale, Rotate

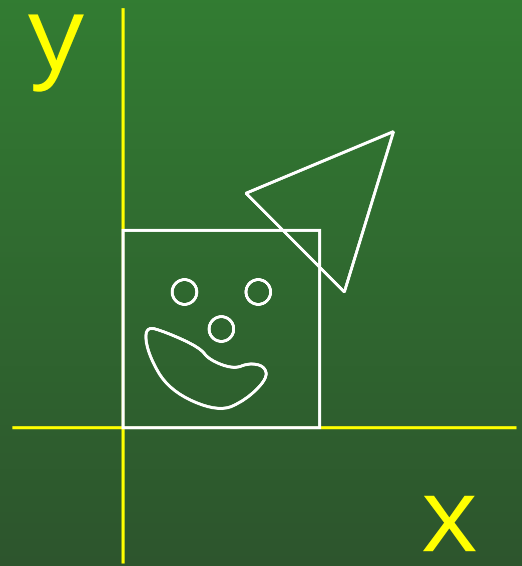


07-94: Other Transformations

- How would the following matrix transform an object?

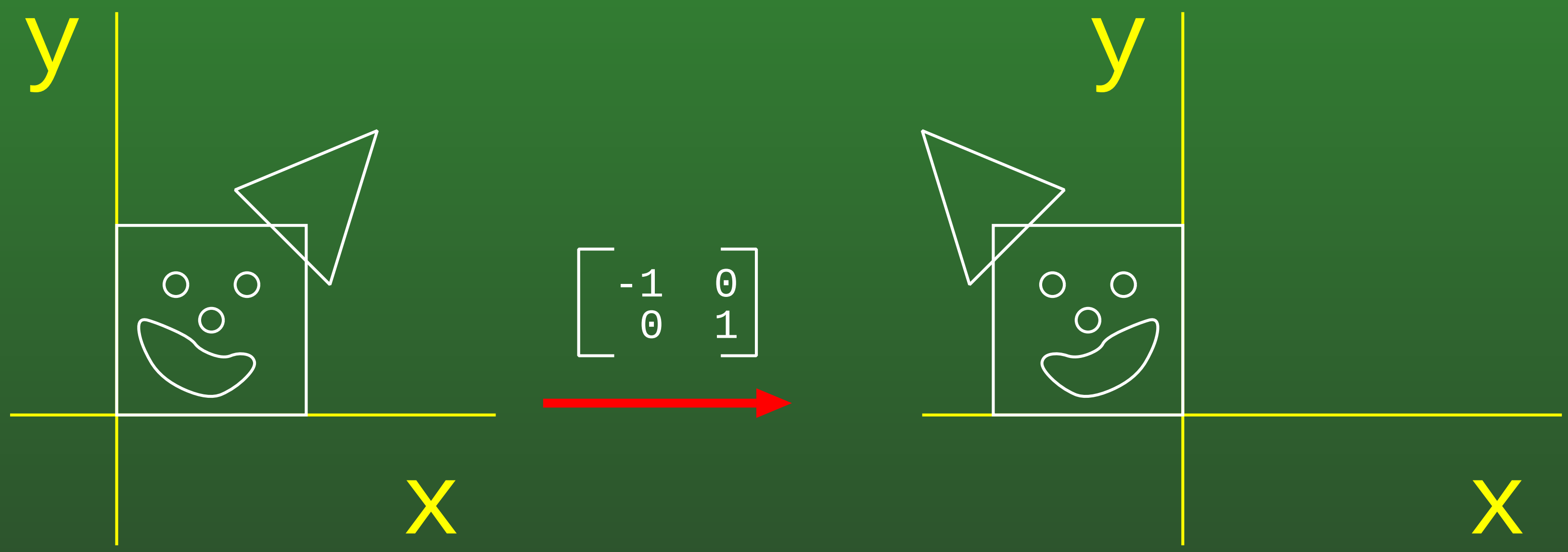
$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

07-95: Reflection



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

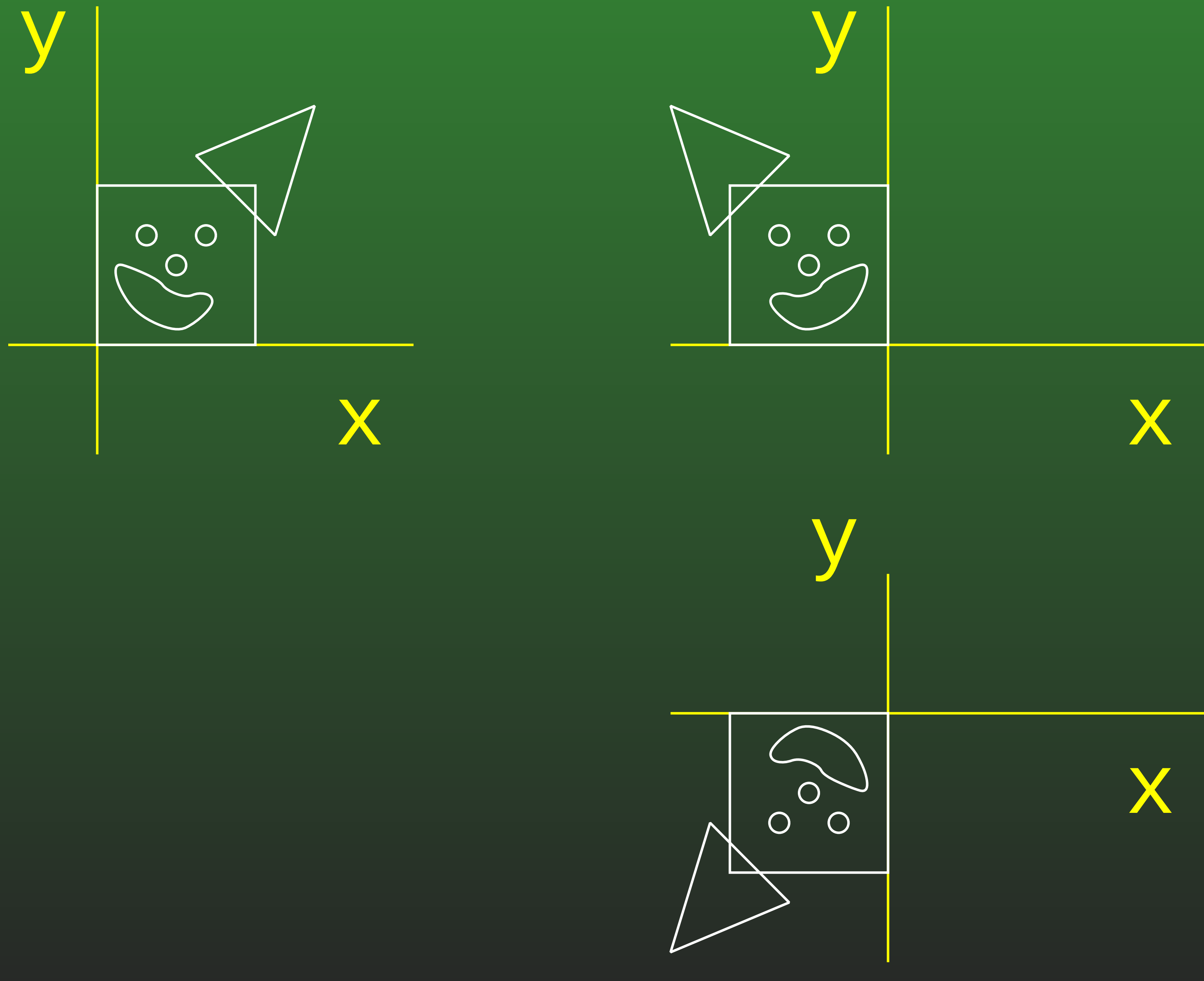
07-96: Reflection



07-97: Reflection

- Reflecting an object twice around the same axis brings and object back to its original state
- Reflecting around the y-axis and then reflecting around the x-axis is the same as ...

07-98: Reflection



07-99: Reflection

- How could we reflect an object around its own center line, instead of around the x- or y- axis?

07-100: Reflection

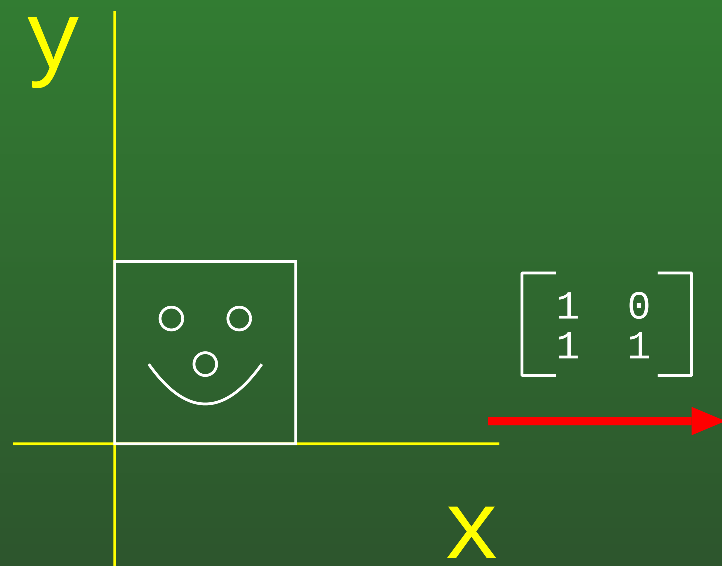
- How could we reflect an object around its own center line, instead of around the x- or y- axis?
  - Translate and rotate the object so that its own center line is the same as the x- or y- axis
  - Reflect
  - Translate, rotate back
- If the centerline of an object is either the x- or y- axis in object space, this is easier ...

07-101: Shearing

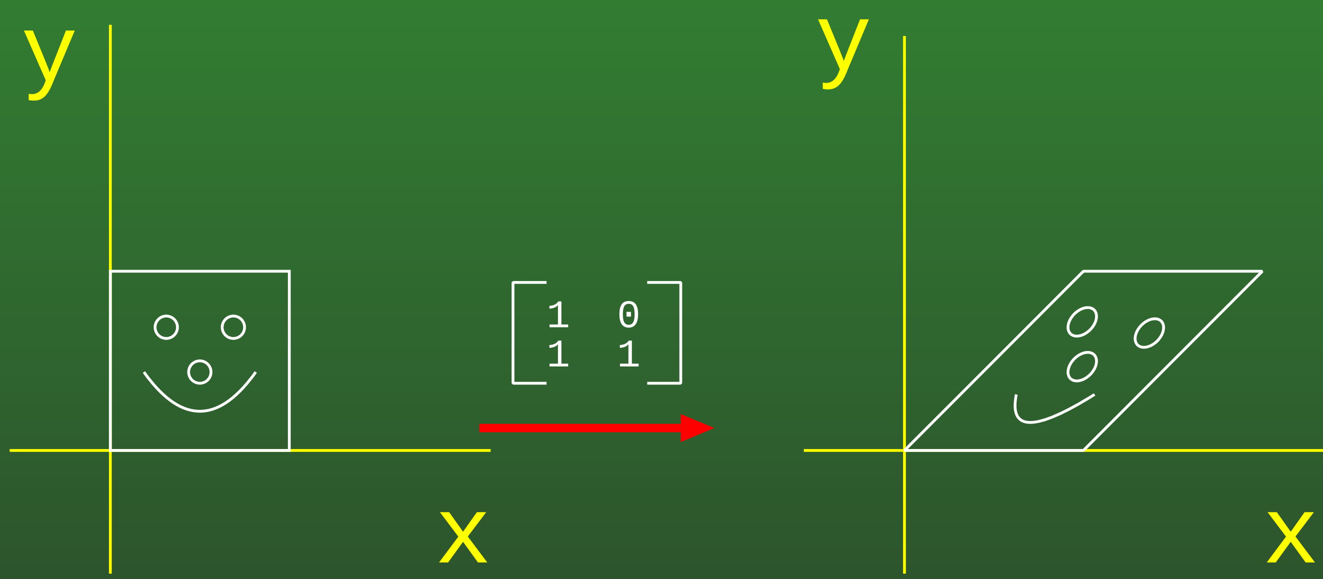
- How would the following matrix transform an object?

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

07-102: Shearing



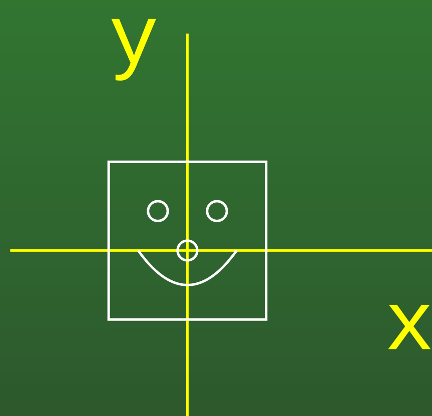
07-103: Shearing



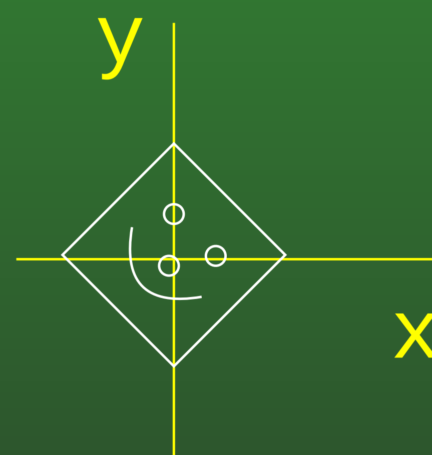
07-104: Shearing

- Shearing an object is the same as:
  - Rotating the object
  - Non-uniform scale
  - Rotating back (not by the same angle)

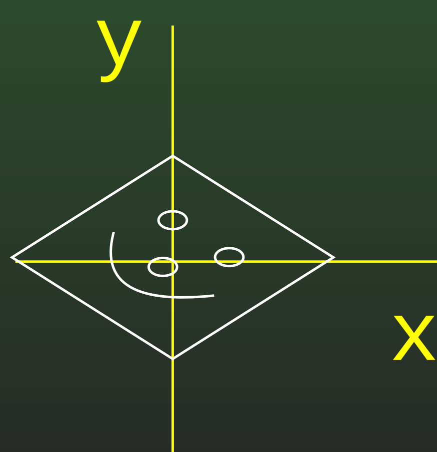
07-105: Shearing



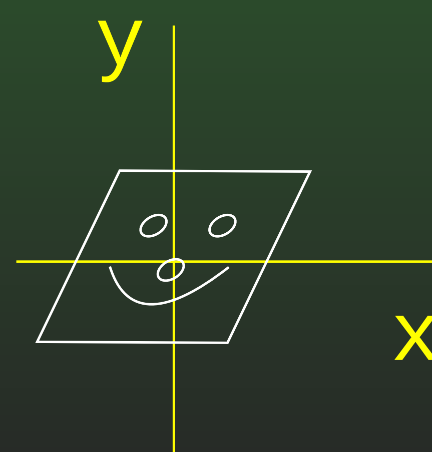
Rotate clockwise 45



Non-uniform scale
(strech x, shrink y)



Rotate counter-
clockwise (~32)



07-106: Linear Transforms

- Matrix operations represent linear transformation of objects
- Number of points in in a line before the transformation, still be in a line after the transformation
  - Line may be stretched and rotated, still be a line

07-107: Linear Transforms

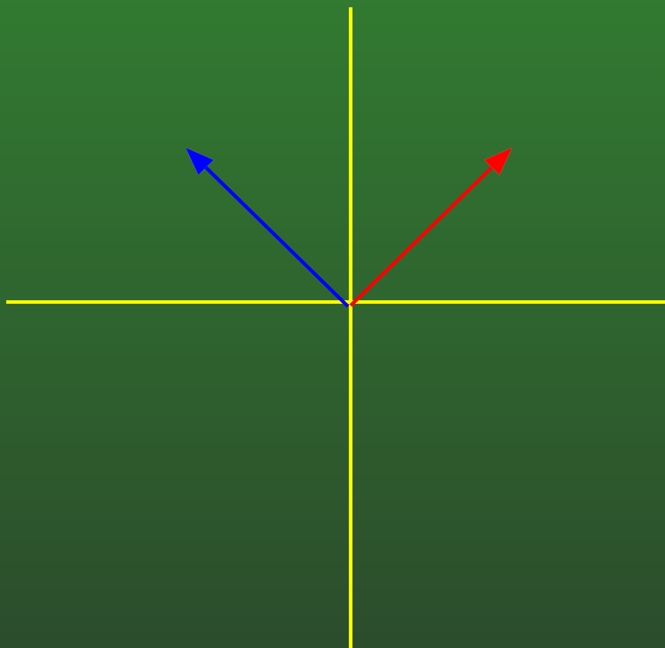
- This gives us a handy way of seeing how a matrix will transform an object
  - See how the matrix will transform the axes [1,0], [0,1]
  - Object will be transformed in the same way

07-108: Linear Transforms

How will this transform an object

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

07-109: Linear Transforms



How will this transform an object

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

x-axis

y-axis

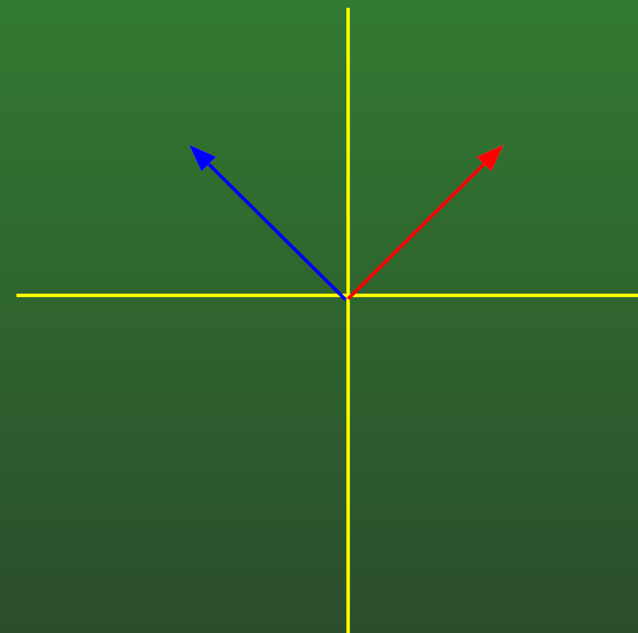
07-110: Linear Transforms

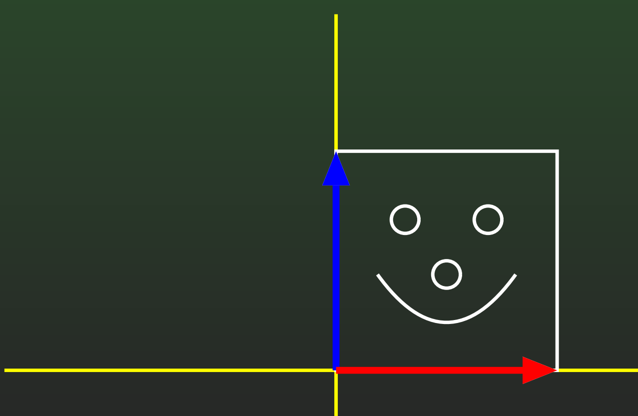
How will this transform an object

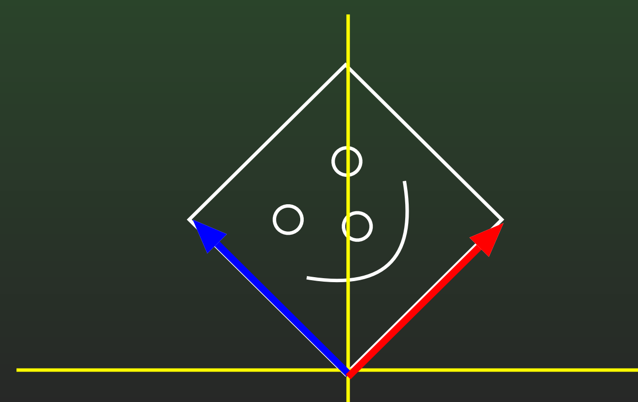
$$\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

x-axis

y-axis







07-111: Linear Transforms

How will this transform an object

$$\begin{bmatrix} \sqrt{2} & \sqrt{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

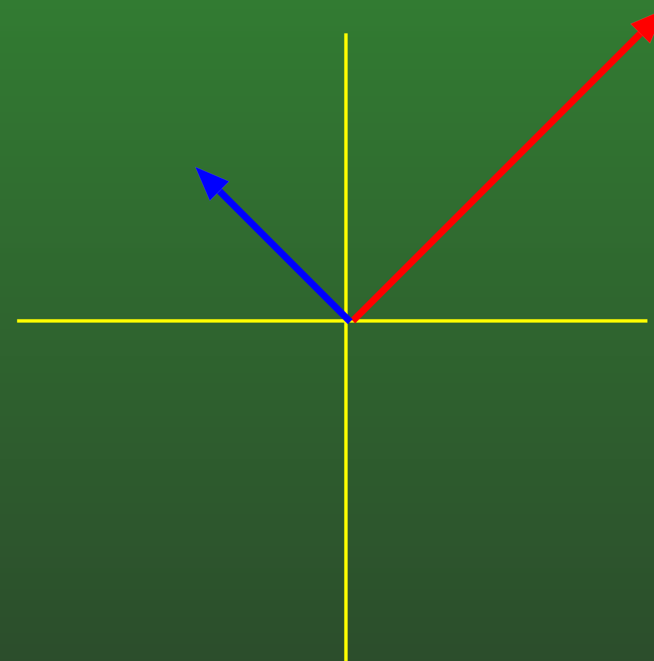
07-112: Linear Transforms

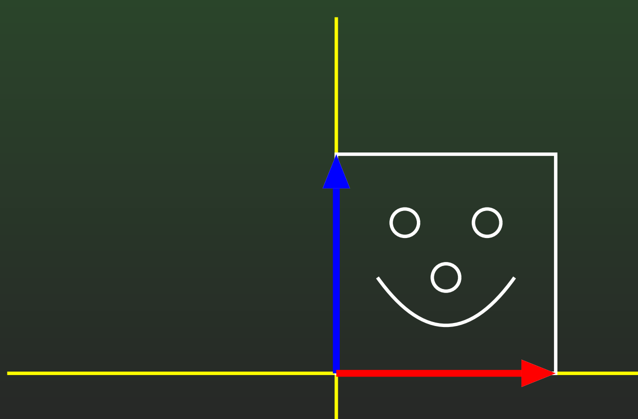
How will this transform an object

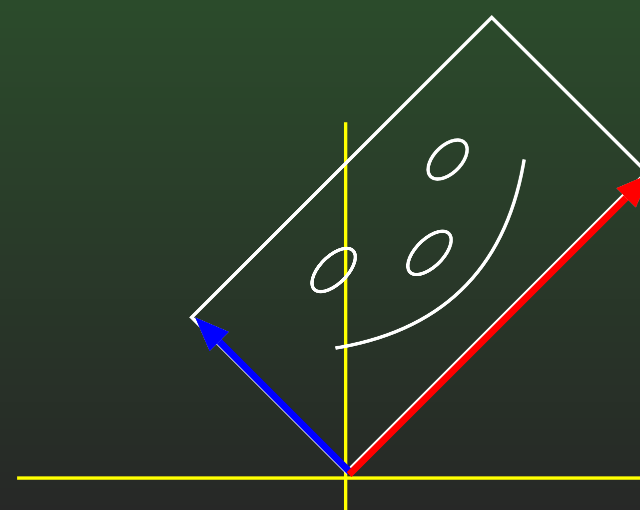
$$\begin{bmatrix} \sqrt{2} & \sqrt{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

x-axis

y-axis







07-113: Linear Transforms

How will this transform an object

$$\begin{bmatrix} 1 & 0.1 \\ 0.1 & 1 \end{bmatrix}$$

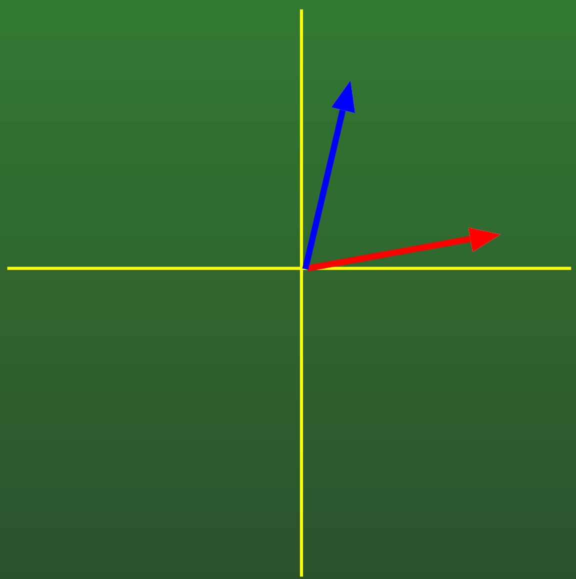
07-114: Linear Transforms

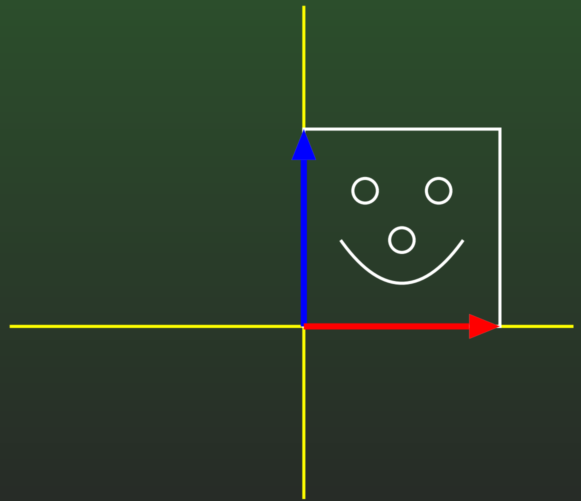
How will this transform an object

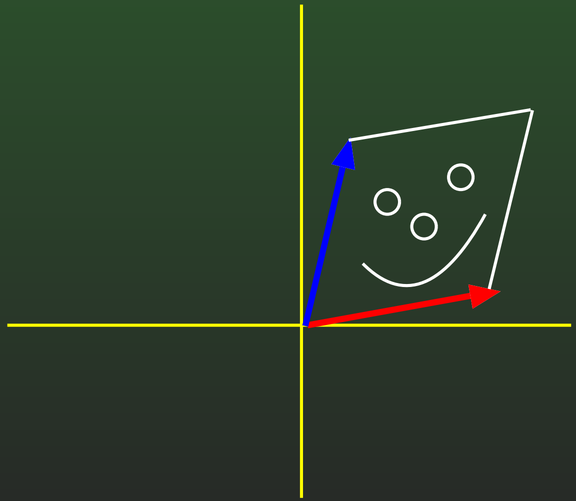
$$\begin{bmatrix} 1 & 0.1 \\ 0.1 & 1 \end{bmatrix}$$

x-axis

y-axis







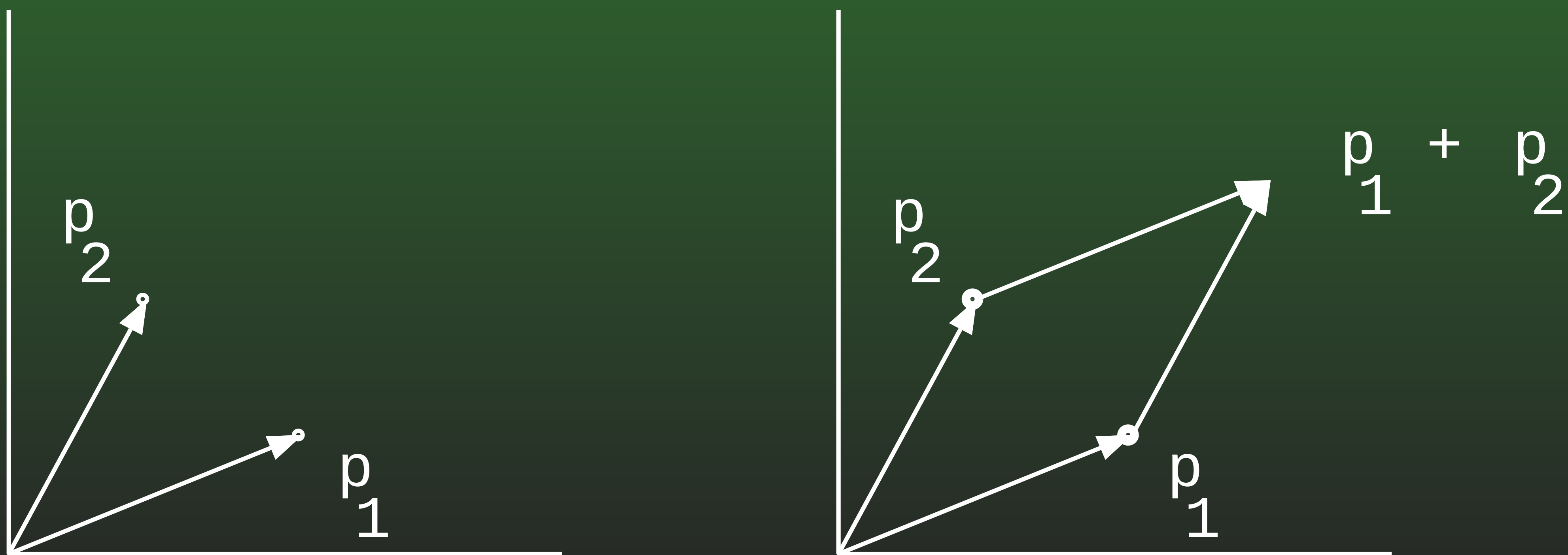
07-115: Determinant

- Determinant of 2x2 matrix:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - cb$$

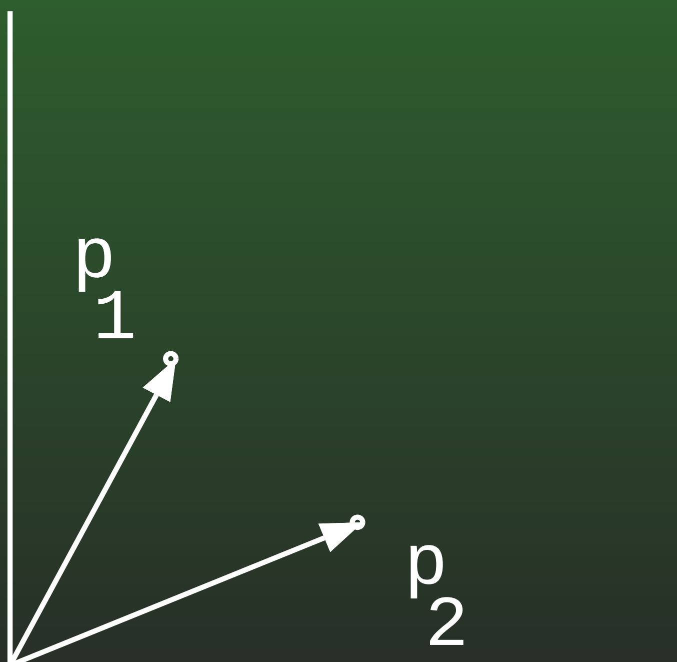
07-116: Determinant

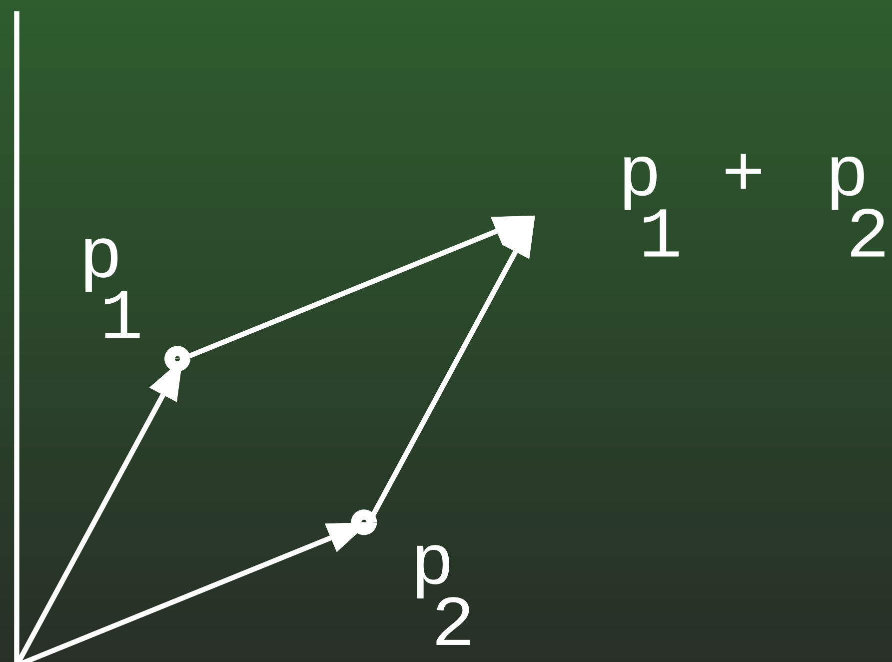
$$\begin{vmatrix} p1_x & p1_y \\ p2_x & p2_y \end{vmatrix} = \text{Area of Parallelogram}$$



07-117: Determinant

$$\begin{vmatrix} p1_x & p1_y \\ p2_x & p2_y \end{vmatrix} = -\text{(Area of Parallelogram)}$$





07-118: Determinant

- Signed area of parallelogram
  - If transformation includes a reflection, then ...

07-119: Determinant

- Signed area of parallelogram
  - If transformation includes a reflection, then
    - Determinant is negative

Determinant

- Signed area of parallelogram
  - If is a pure rotation (no scale, no shear, no rotation) ...

07-121: Determinant

- Signed area of parallelogram
  - If is a pure rotation (no scale, no shear, no rotation) ...
    - Determinant = 1
    - (Other direction is not always true..)

07-122: Translation

- We can implement rotation / scale / reflection / shearing using matrix operations
- What about translation?

07-123: Translation

- Can’t translate an object using a 2x2 matrix
- Consider a point at the origin
  - How will a point a the origin be modified by a matrix?

07-124: Translation

- Can’t translate an object using a 2x2 matrix
- Consider a point at the origin
  - How will a point a the origin be modified by a matrix?
  - Not changed by *any* matrix!

07-125: Translation

- Matrices can only do linear transformations
- Translation is not a linear transformation
  - Can’t use matrices to do translation
  - ... Unless we cheat a little!

07-126: Translation

- We can use matrices to do translation – as long as we use something different than 2x2 matrices
  - Add a dummy value to the end of all points (always 1)
  - Add a new row / column to matrix

$$[x, y, 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x, y, 1]$$

07-127: Translation

$$[x, y, 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \delta_x & \delta_y & 1 \end{bmatrix} =$$

07-128: Translation

$$[x, y, 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \delta_x & \delta_y & 1 \end{bmatrix} = [x + \delta_x, y + \delta_y, 1]$$

Translation

- Adding rotation

$$[x, y] \begin{bmatrix} a & b \\ c & d \end{bmatrix} = [xa + yc, xb + yd]$$

$$[x, y, 1] \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix} = [xa + yc, xb + yd, 1]$$

07-130: Translation

$$[x, y, 1] \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ \delta_x & \delta_y & 1 \end{bmatrix} = [xa + yc + \delta_x, xb + yd + \delta_y, 1]$$

- So, we can use a matrix to do both rotation and translation

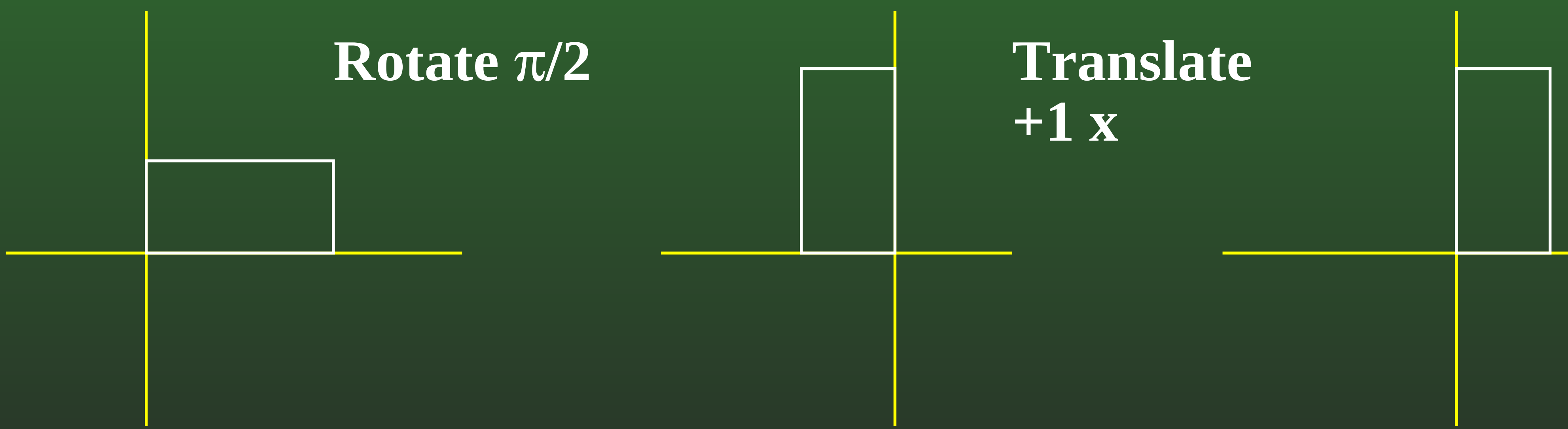
07-131: Translation

$$\begin{bmatrix} \cos\Theta & \sin\Theta & 0 \\ -\sin\Theta & \cos\Theta & 0 \\ \delta_x & \delta_y & 1 \end{bmatrix}$$

- First, rotate counter-clockwise by Θ
- Then translate by $[\delta_x, \delta_y]$

07-132: Combining Transforms

- Let's look at an example
  - First rotate by $\pi/2$ (90 degrees) counterclockwise
  - Then translate x by +1

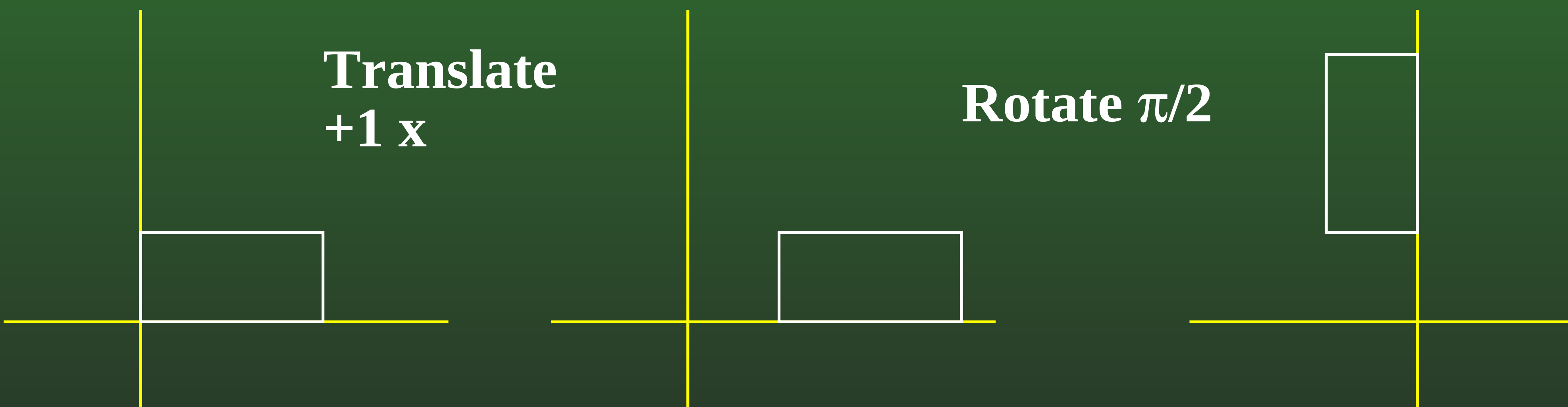


Combining Transforms

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

07-134: Combining Transforms

- Another example
  - First translate x by +1
  - Then rotate by $\pi/2$ (90 degrees) counterclockwise

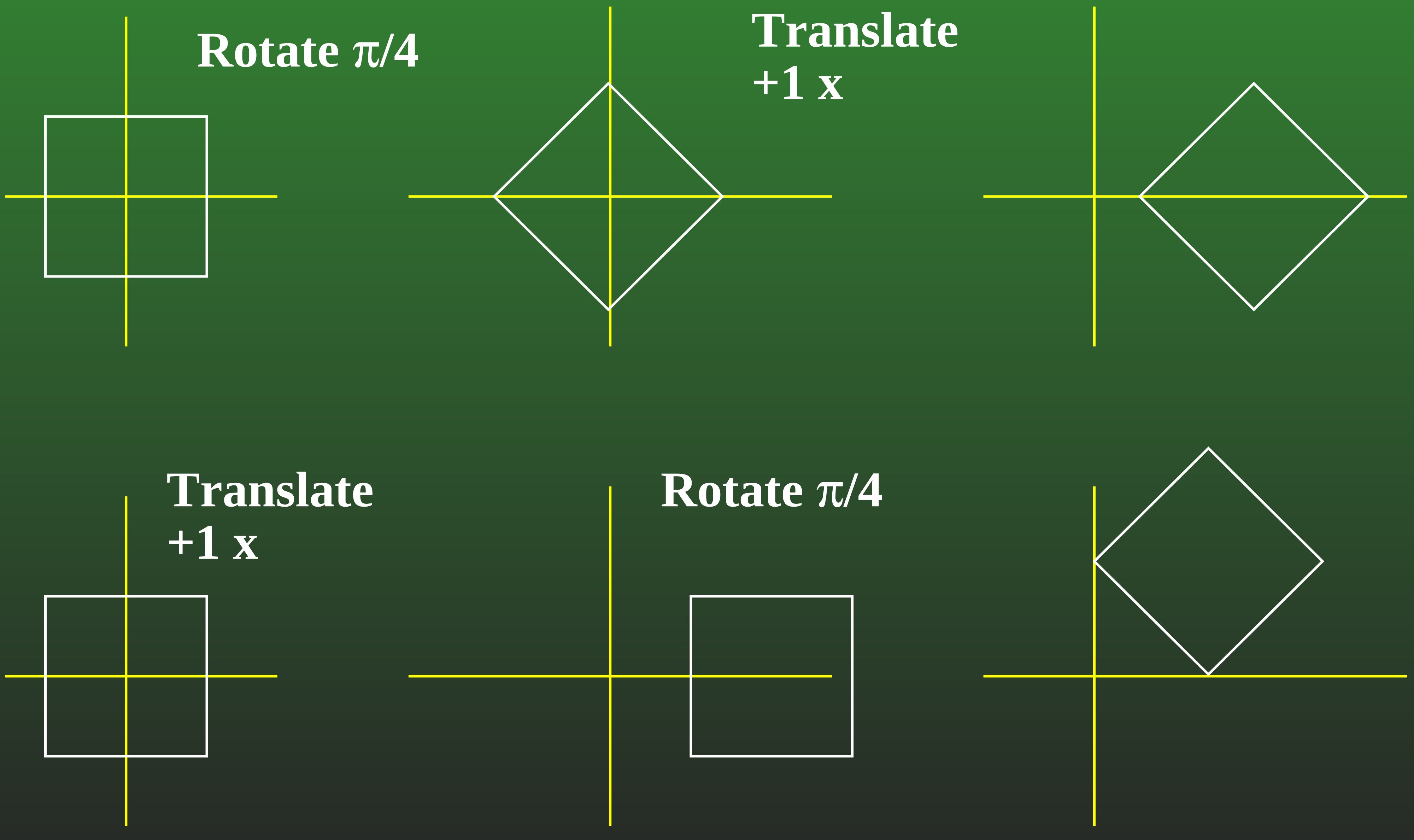


07-135: Combining Transforms

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

- Same as rotating, and then moving up $+y$

07-136: Combining Transforms



07-137: Combining Transforms

- Rotating by $\pi/4$, then translating 1 unit $+x$

$$\begin{bmatrix} \cos\Theta & \sin\Theta & 0 \\ -\sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & \sin\Theta & 0 \\ -\sin\Theta & \cos\Theta & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

07-138: Combining Transforms

- Translating 1 unit $+x$, then rotating by $\pi/4$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & \sin\Theta & 0 \\ -\sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & \sin\Theta & 0 \\ -\sin\Theta & \cos\Theta & 0 \\ \cos\Theta & \sin\Theta & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 1 \end{bmatrix}$$

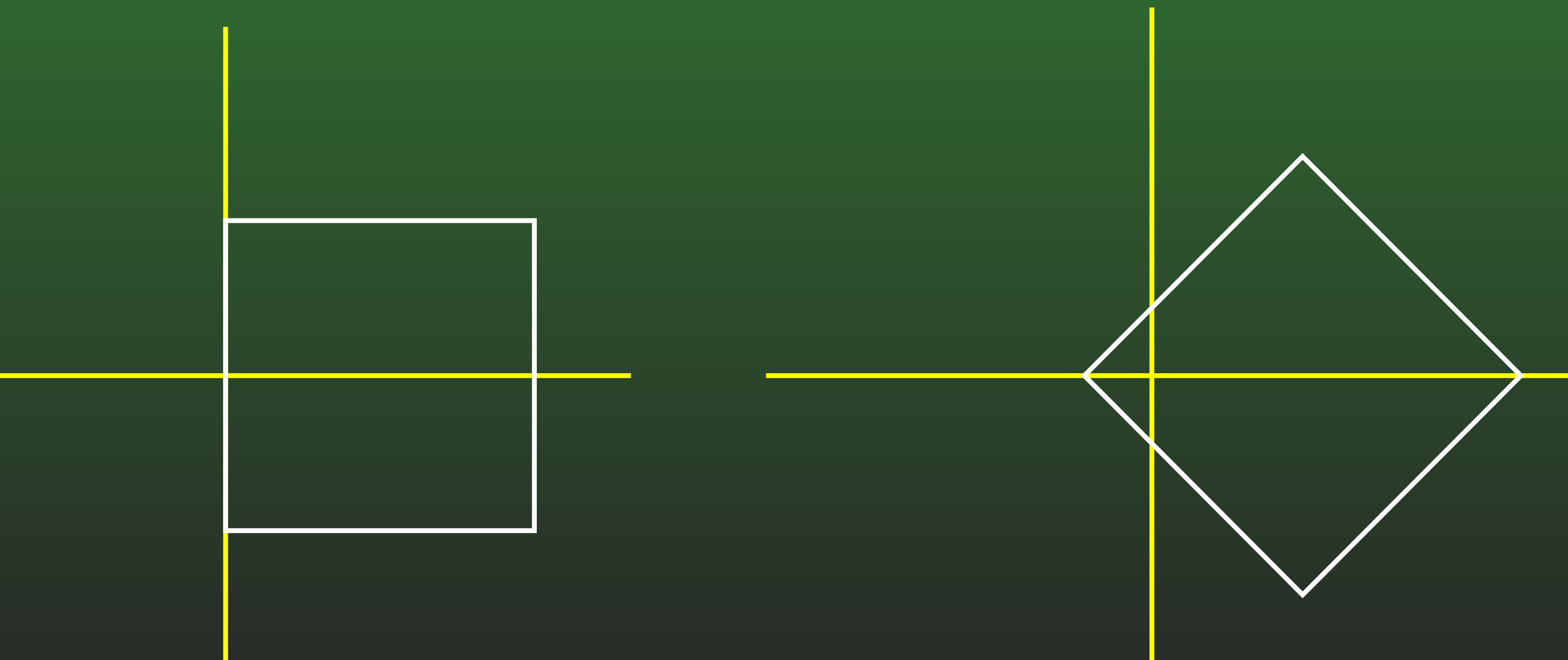
- Same as rotating $\pi/4$ counterclockwise, and then translating over (+x) $1/\sqrt{2}$ and up (+y) $1/\sqrt{2}$

Non-Standard Axes

- We want to rotate around an axis that does not go through the origin
- Rotate around point at 1,0
- Create the approprate 3x3 vector

Non-Standard Axes

Rotate $\pi/4$ around (1,0)



07-141: Non-Standard Axes

- First, translate to the origin
- Then, do the rotation
- Finally, translate back

07-142: Non-Standard Axes

- First, translate to the origin

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

- Then, do the rotation
- Finally, translate back

07-143: Non-Standard Axes

- First, translate to the origin
- Then, do the rotation

$$\begin{bmatrix} \cos\Theta & \sin\Theta & 0 \\ -\sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Finally, translate back

07-144: Non-Standard Axes

- First, translate to the origin
- Then, do the rotation
- Finally, translate back

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

07-145: Non-Standard Axes

- Final matrix:

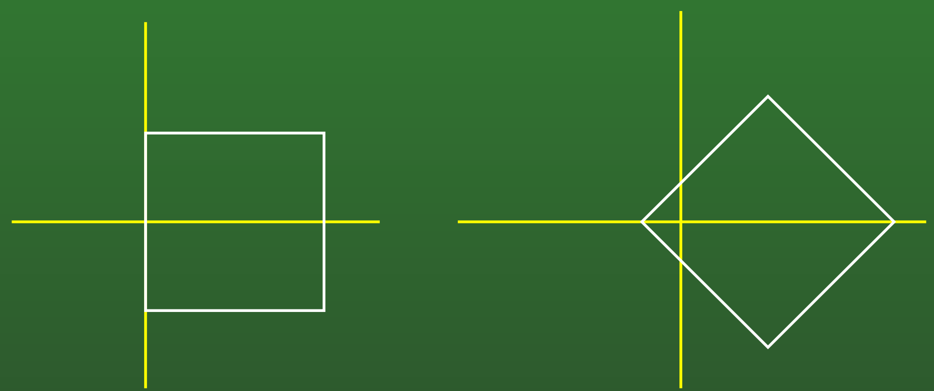
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & -1/\sqrt{2} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

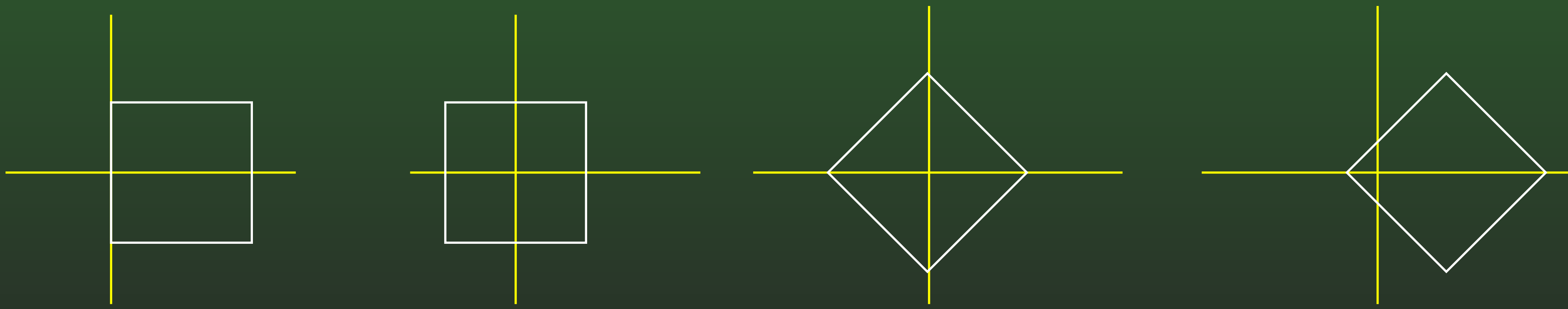
$$\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1-1/\sqrt{2} & -1/\sqrt{2} & 1 \end{bmatrix}$$

07-146: Non-Standard Axes

Rotate π/4 around (1,0)

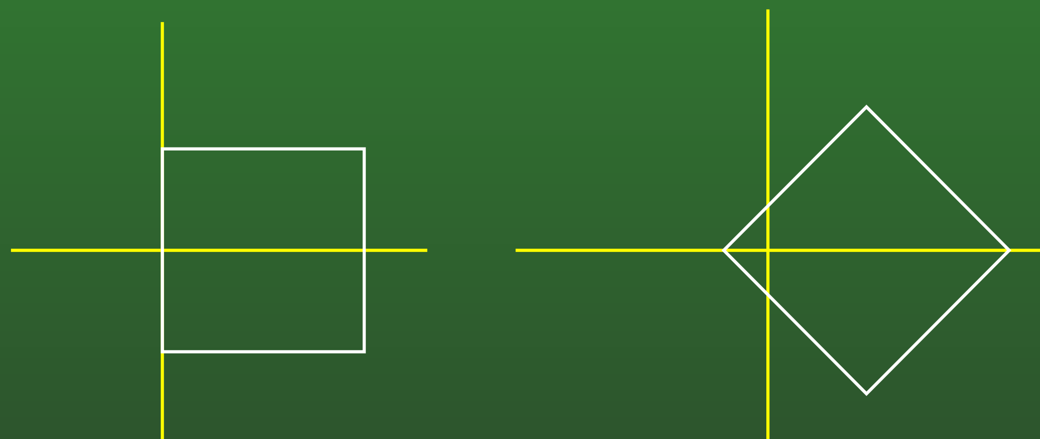


Translate to origin, Rotate π/4 around (0,0), Translate back

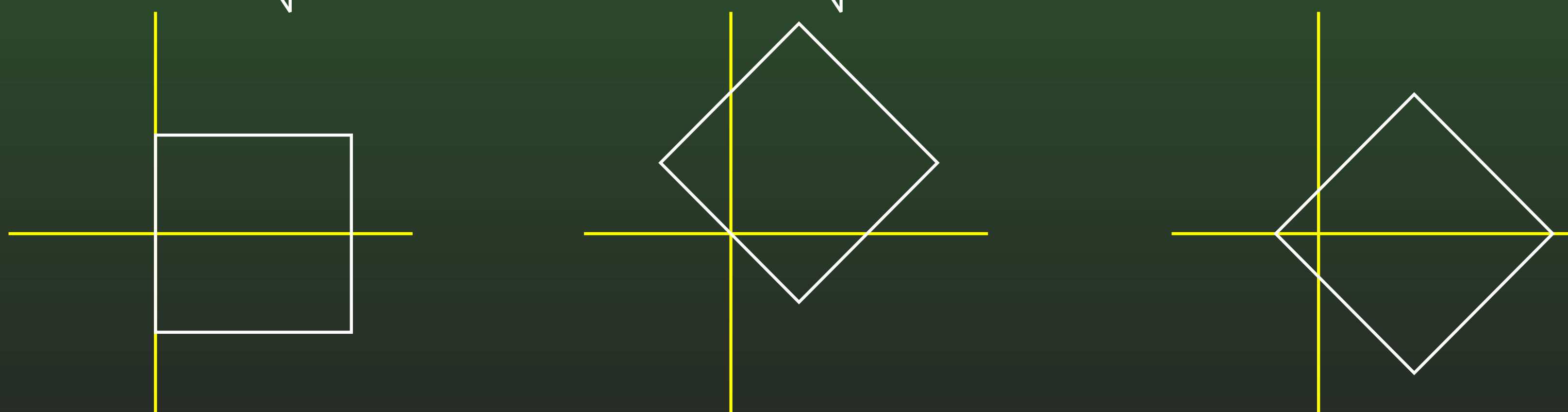


07-147: Non-Standard Axes

Rotate π/4 around (1,0)



Rotate p/4 around (0,0), then translate over 1 - 1 /$\sqrt{2}$ and down 1/$\sqrt{2}$



07-148: Non-Standard Axes

- Note that the *rotation* component (upper left 2x2 matrix) is the same as if we were rotating around the origin
- Only the *position* component is altered.
- In general, whenever we do a rotation and a number of translations, the rotation component will be unchanged

Linear Transforms?

- Matricies can only do linear transformations
- Translation is *not* a linear transform
- ... but we are using matrices to do translation
  - What’s up?

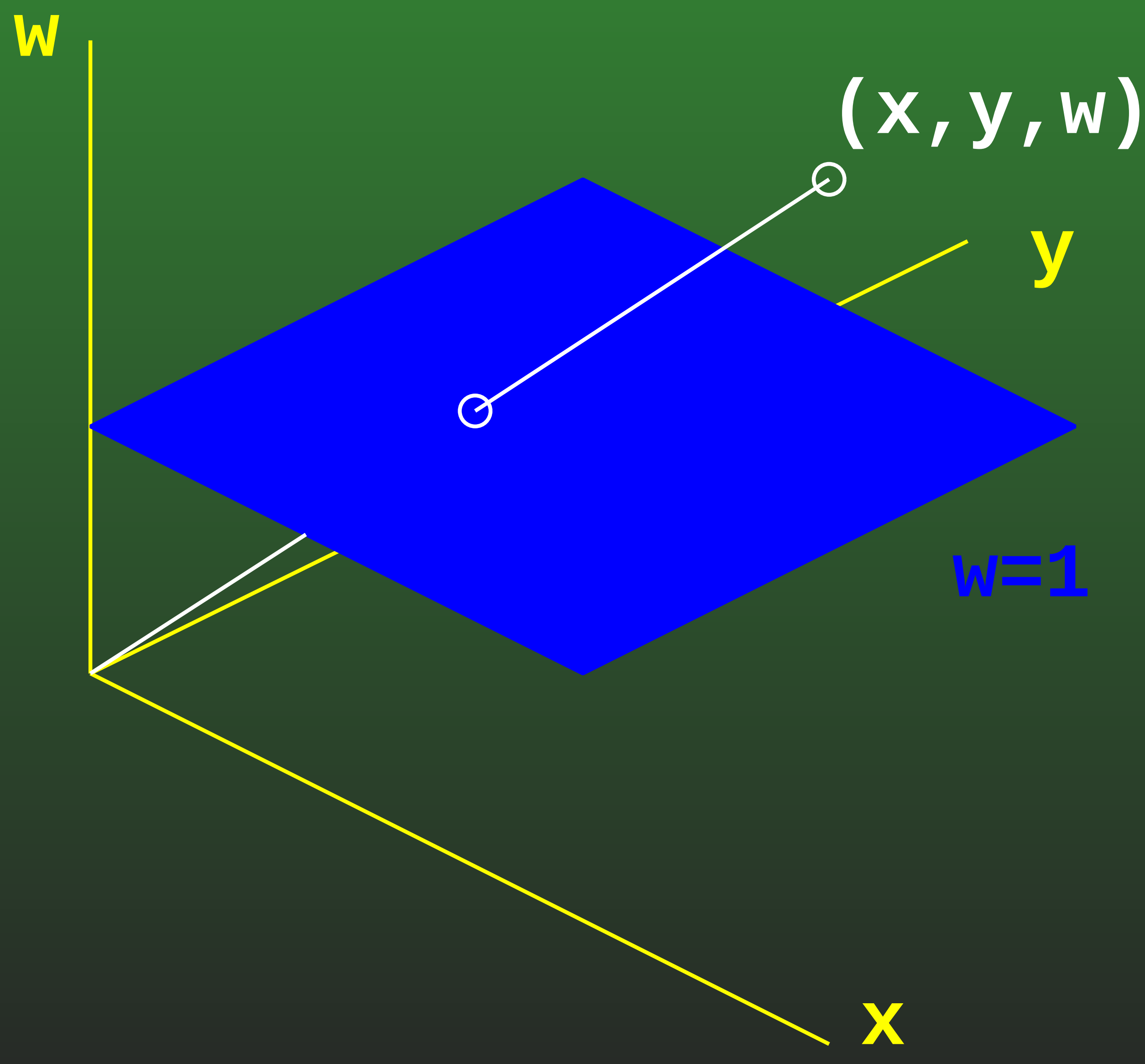
Homogeneous Space

- We are no longer working in 2D, we are now working in 3D
  - Extra 3rd parameter, that is always == 1
- We can extend our definition to allow the 3rd parameter to be some value other than 1
  - Need to be able to convert back to 2D space

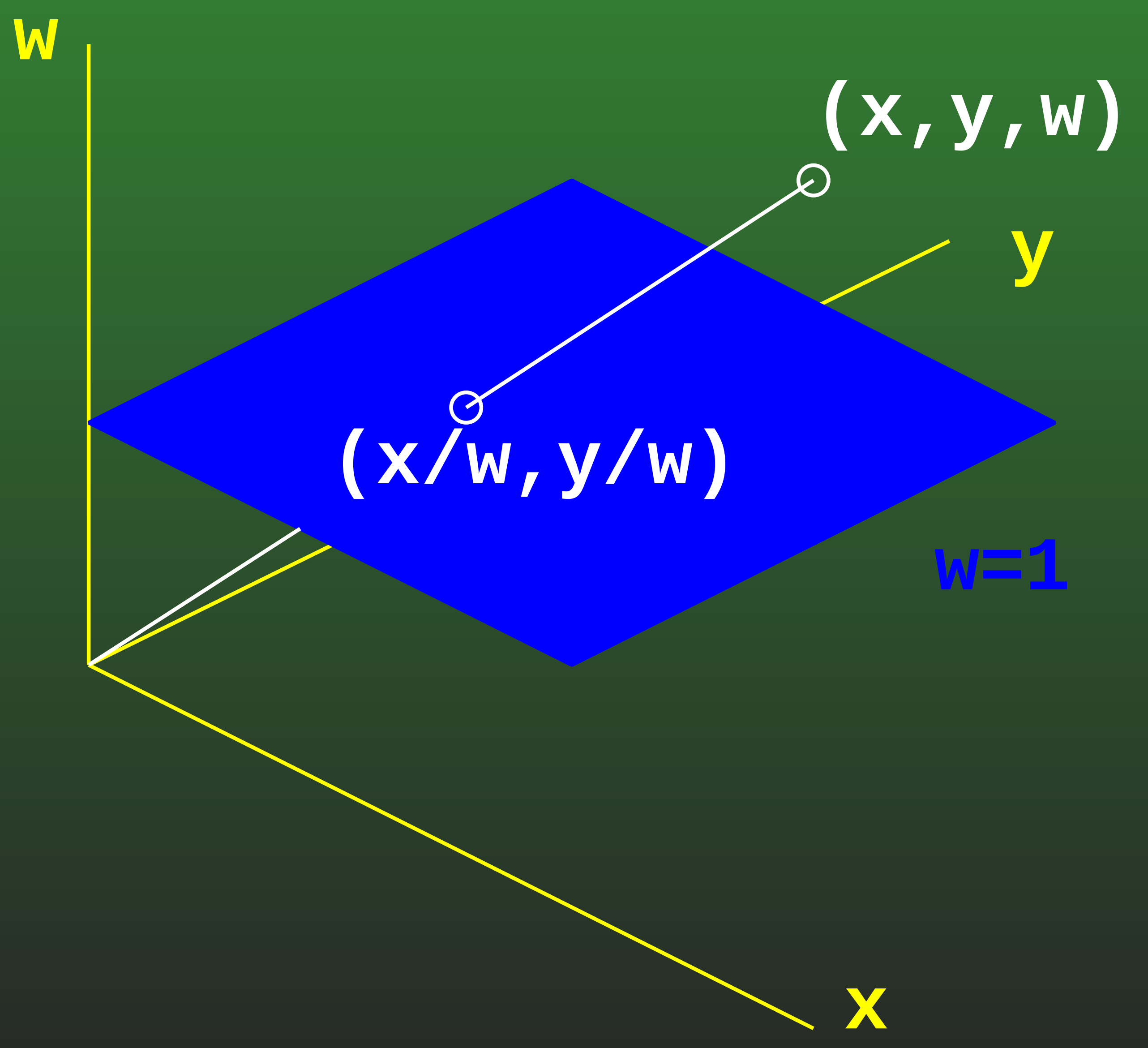
3D Homogeneous Space

- To convert a point (x, y, w) in 3D Homogeneous space into 2D (x, y) space:
  - Place a plane at $w = 1$
  - (x, y, w) maps to the (x, y) position on the plane where the ray (x, y, w) intersects the plane

07-152: 3D Homogeneous Space



3D Homogeneous Space



07-154: 3D Homogeneous Space

- Converting from a point in 3D homogeneous space to 2D space is easy
  - Divide the x and y coordinates by w
  - What happens when $w = 0$?

3D Homogeneous Space

- Converting from a point in 3D homogeneous space to 2D space is easy
  - Divide the x and y coordinates by w
  - What happens when $w = 0$?
    - “Point at infinity”
    - Direction, but not a magnitude

07-156: 3D Homogeneous Space

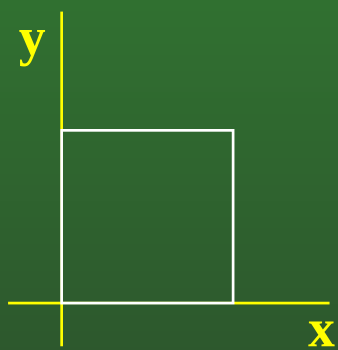
- For a given (x, y, w) point in 3D Homogeneous space, there is a single corresponding point in "standard" 2D space
  - Though when $w = 0$, we are in a bit of a special case
- For a single point in "standard" 2D space, there are an infinite number of corresponding points in 3D Homogeneous space

Translation

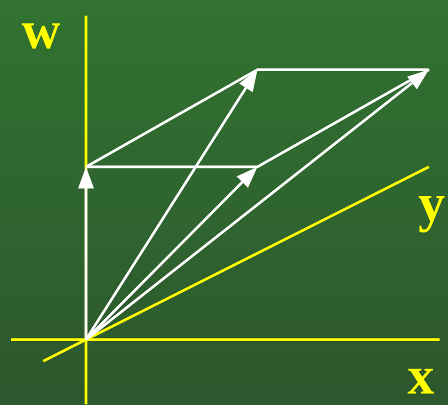
- We are still doing a linear trasformation of the 3D vector
- We are *shearing* the 3D space
- The resulting projection back to 2D is seen as a translation

07-158: Translation

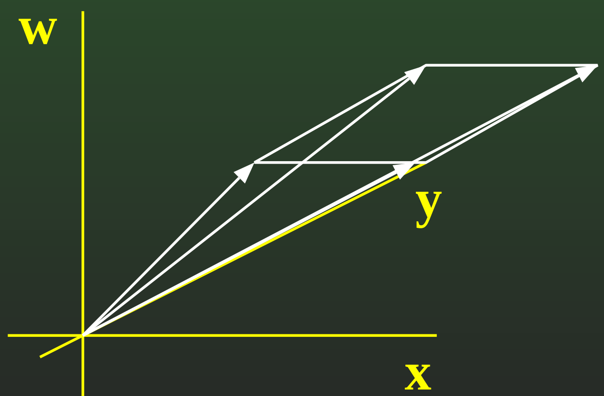
2D Shape



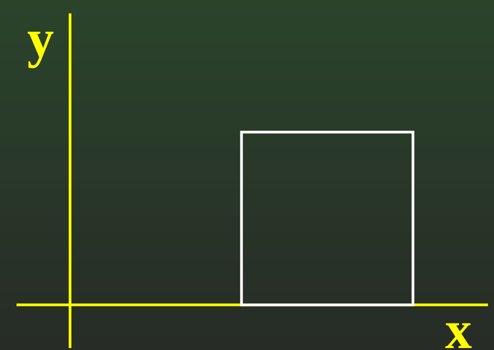
Transform to 3D Homogenous Space



Shear operation in 3D space



Back to 2D



07-159: Inverse

- Finding inverse of 2x2 matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$A^{-1} = \frac{1}{\det A} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

07-160: Inverse

- A matrix is signular if it does not have an inverse
  - determinant = 0
- Why does this make sense, geometrically?

07-161: Orthogonal Matrices

- A matrix M is orthogonal if:
  - $\mathrm{M}\mathrm{M}^T = \mathbf{I}$
  - $\mathrm{M}^T = \mathrm{M}^{-1}$
- Orthogonal matrices are handy, because they are easy to invert
- Is there a geometric interpretation of orthogonality?

07-162: Orthogonal Matrices

$$\mathbf{M}\mathbf{M}^T = \mathbf{I}$$

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} m_{11} & m_{21} \\ m_{12} & m_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Do all the multiplications ...

07-163: Orthogonal Matrices

$$\begin{aligned} m_{11}m_{11} + m_{12}m_{12} &= 1 \\ m_{11}m_{21} + m_{12}m_{22} &= 0 \\ m_{21}m_{11} + m_{22}m_{12} &= 0 \\ m_{21}m_{21} + m_{22}m_{22} &= 1 \end{aligned}$$

- Hmmm... that doesn't seem to help much

Orthogonal Matrices

- Recall that rows of matrix are basis after rotation
  - $\mathbf{v}_x = [m_{11}, m_{12}]$
  - $\mathbf{v}_y = [m_{21}, m_{22}]$
- Let's rewrite the previous equations in terms of $\mathbf{v}_x$ and $\mathbf{v}_y$ ...

07-165: Orthogonal Matrices

$$
\begin{aligned}
m_{11}m_{11} + m_{12}m_{12} &= 1 & \mathbf{v}_x \cdot \mathbf{v}_x = 1 \\
m_{11}m_{21} + m_{12}m_{22} &= 0 & \mathbf{v}_x \cdot \mathbf{v}_y = 0 \\
m_{21}m_{11} + m_{22}m_{12} &= 0 & \mathbf{v}_y \cdot \mathbf{v}_x = 0 \\
m_{21}m_{21} + m_{22}m_{22} &= 1 & \mathbf{v}_y \cdot \mathbf{v}_y = 1
\end{aligned}
$$

07-166: Orthogonal Matrices

- What does it mean if $\mathbf{u} \cdot \mathbf{v} = 0$?
  - (assuming both $\mathbf{u}$ and $\mathbf{v}$ are non-zero)
- What does it mean if $\mathbf{v} \cdot \mathbf{v} = 1$?

Orthogonal Matrices

- What does it mean if $\mathbf{u} \cdot \mathbf{v} = 0$?
  - (assuming both $\mathbf{u}$ and $\mathbf{v}$ are non-zero)
  - $\mathbf{u}$ and $\mathbf{v}$ are perpendicular to each other (orthogonal)
- What does it mean if $\mathbf{v} \cdot \mathbf{v} = 1$?
  - $||\mathbf{v}|| = 1$
- So, transformed basis vectors must be mutually perpendicular unit vectors

07-168: Orthogonal Matrices

- If a transformation matrix is orthogonal,
  - Transformed basis vectors are mutually perpendicular unit vectors
- What kind of transformations are done by orthogonal matrices?

07-169: Orthogonal Matrices

- If a transformation matrix is orthogonal,
  - Transformed basis vectors are mutually perpendicular unit vectors
- What kind of transformations are done by orthogonal matrices?
  - Rotations & Reflections

07-170: Orthogonal Matrices

- Sanity check: Rotational matrices are orthoginal:

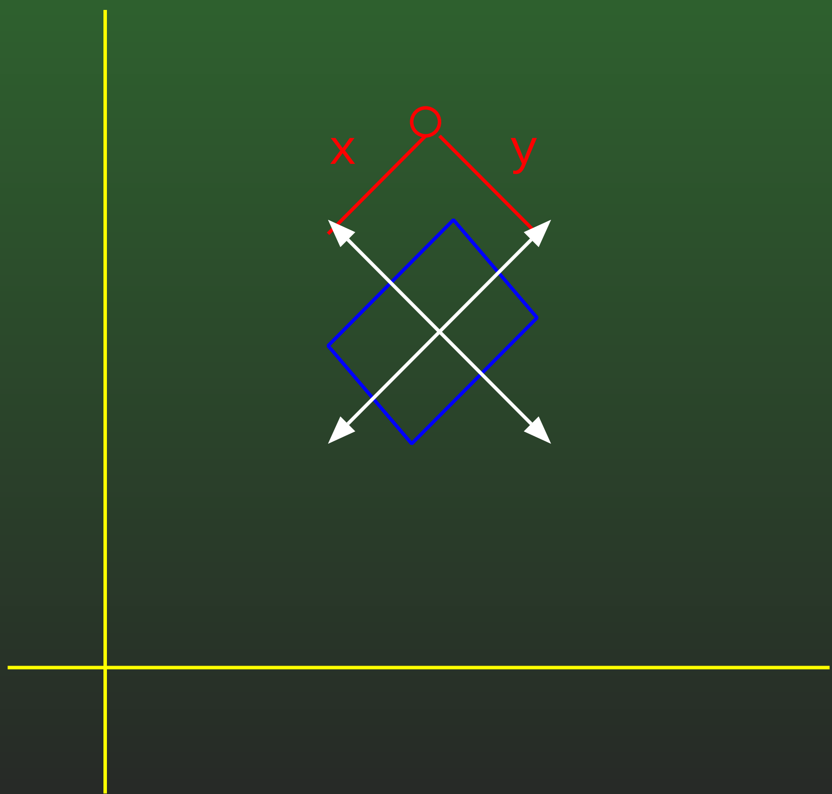
$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, A^{-1} = \frac{1}{\det A} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

$$A = \begin{bmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{bmatrix}, A^{-1} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix}$$

$$= \begin{bmatrix} \cos(-\Theta) & \sin(-\Theta) \\ -\sin(-\Theta) & \cos(-\Theta) \end{bmatrix}$$

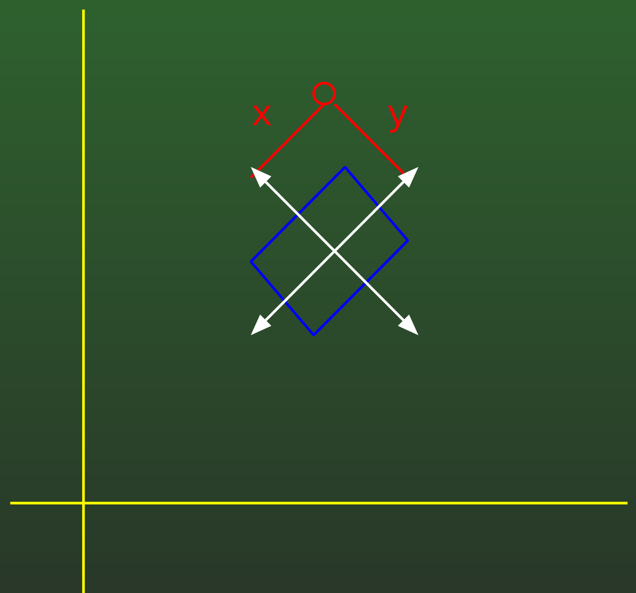
07-171: Examples

- An object's position has a rotational matrix M and a position p. A point $o_1 = [o_{1x}, o_{1y}]$ is in *object space* for the object, What is the position of the point in world space?



07-172: Examples

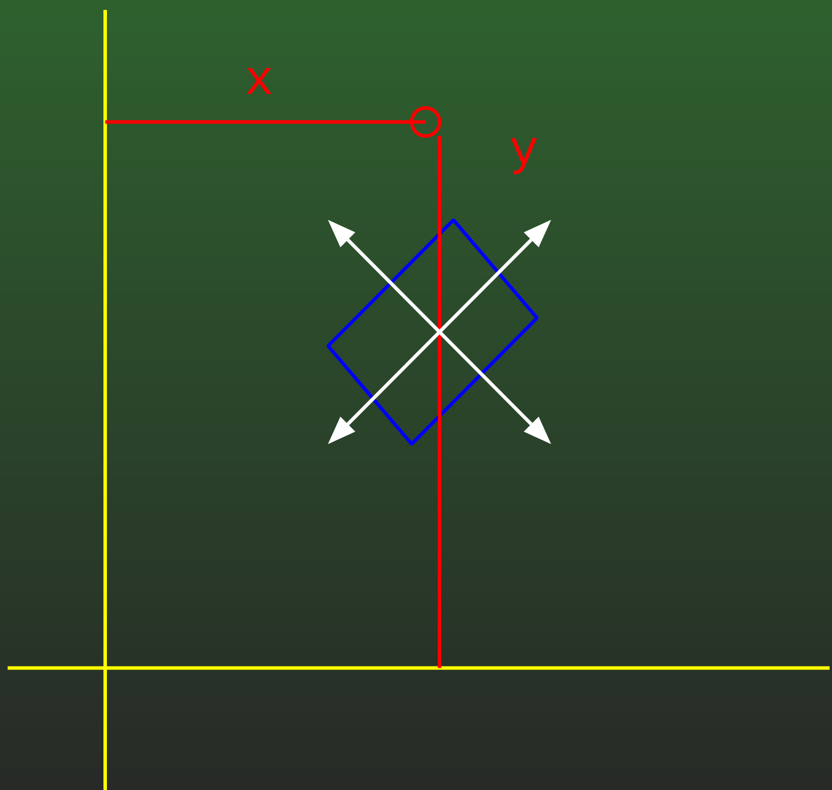
- An object's position has a rotational matrix M and a position p. A point $o_1 = [o_{1x}, o_{1y}]$ is in *object space* for the object, What is the position of the point in world space?



Position in world space: $o_1 M + p$

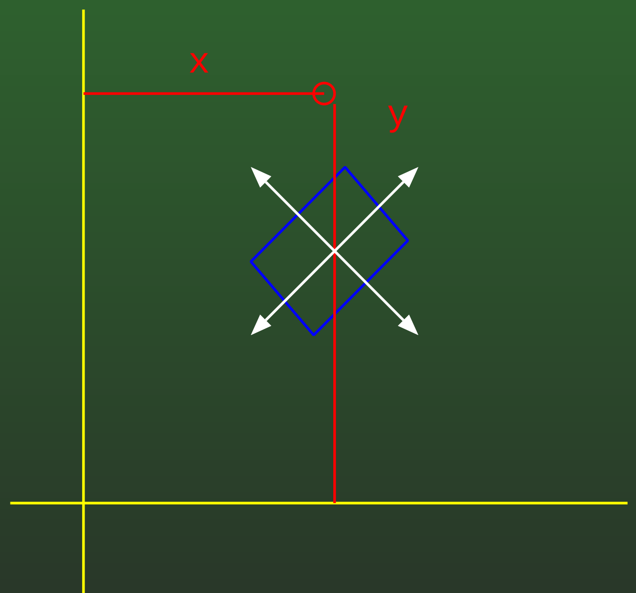
07-173: Examples

- An object's position has a rotational matrix M and a position p. A point $w_1 = [w_{1_x}, w_{1_y}]$ is in *world space.* What is the position of the point in objet space



Examples

- An object's position has a rotational matrix M and a position p. A point $w_1 = [w_{1_x}, w_{1_y}]$ is in *world space.* What is the position of the point in objet space

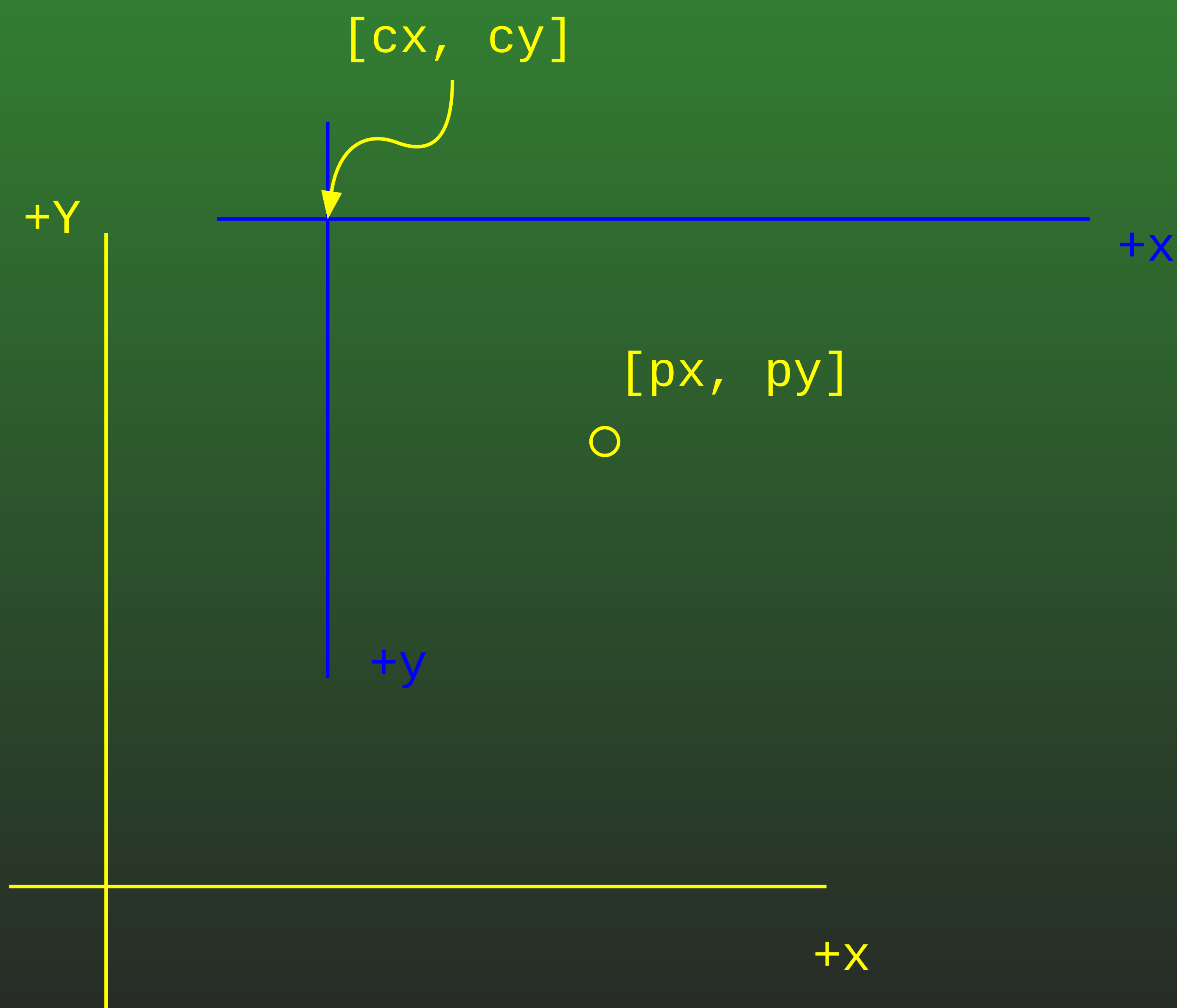


Position in object space: $(w_1 - p)M^T$

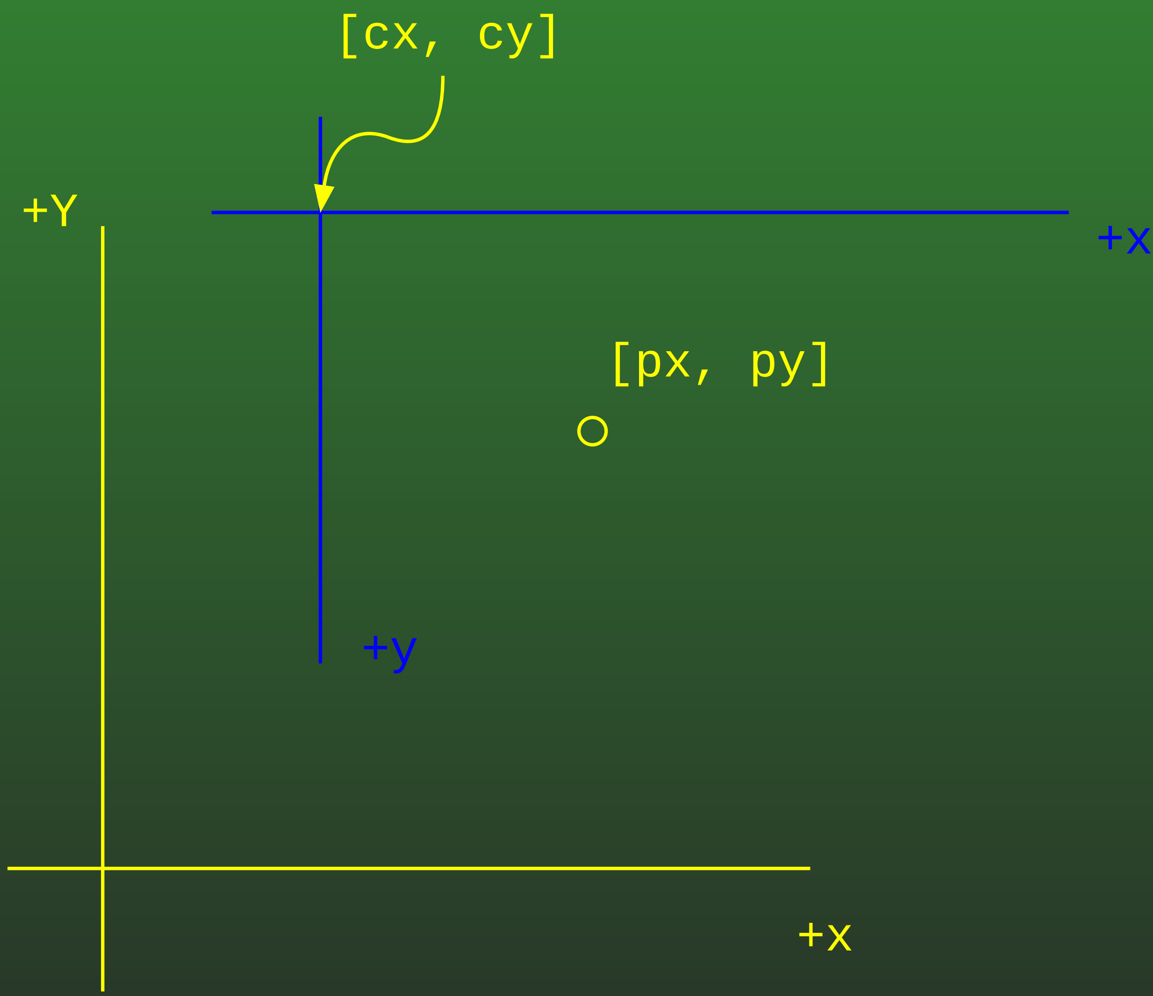
07-175: Examples

- Origin of screen is at position $[c_x, c_y]$ in *world space*
- Object is a point $[p_x, p_y]$ in world space
- World has $+x$ to right, $+y$ up
- Screen has $+x$ to right, $+y$ down
- What is the position of p in screen space?

07-176: Examples

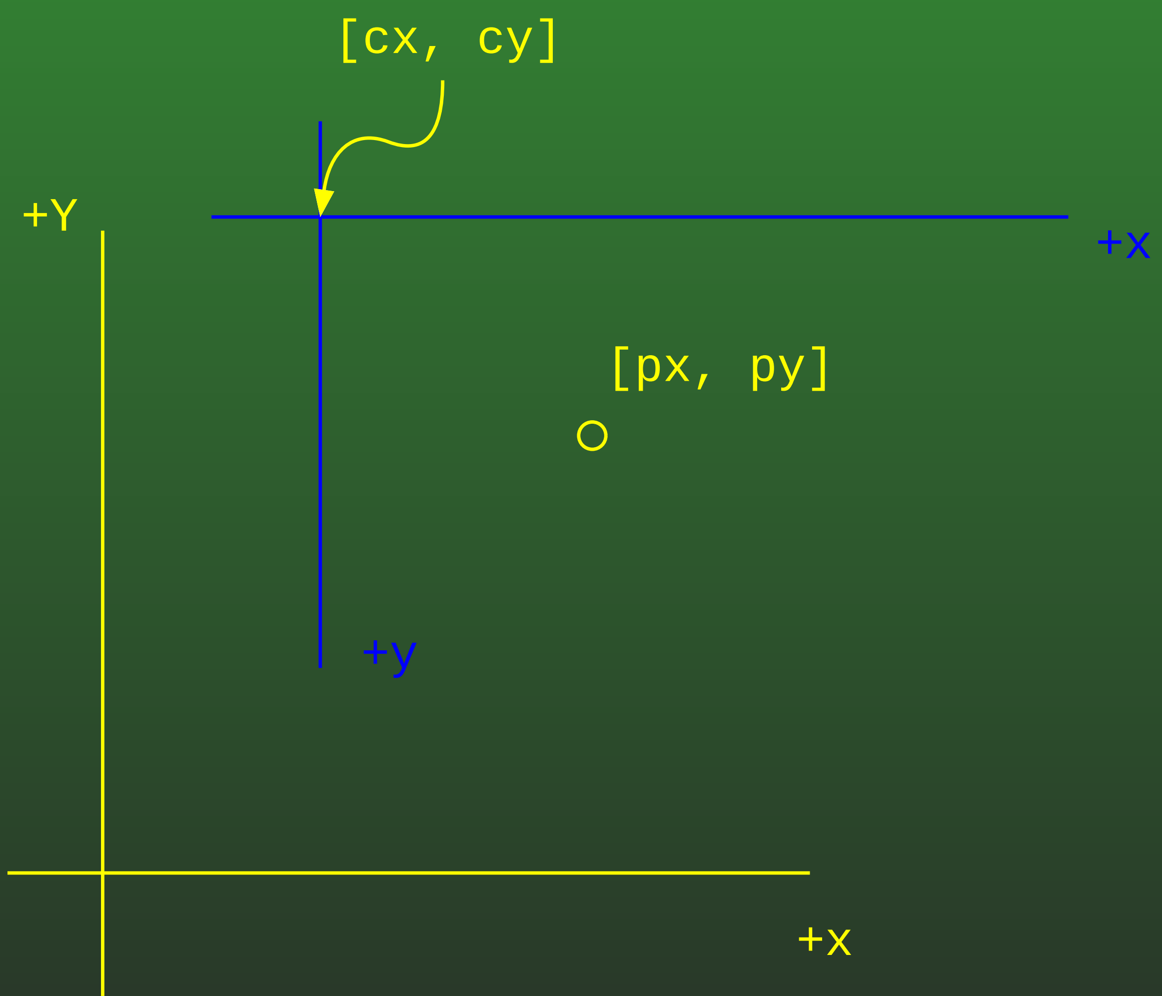


07-177: Examples



$$[p_x, p_y] - [c_x, c_y] \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

07-178: Examples



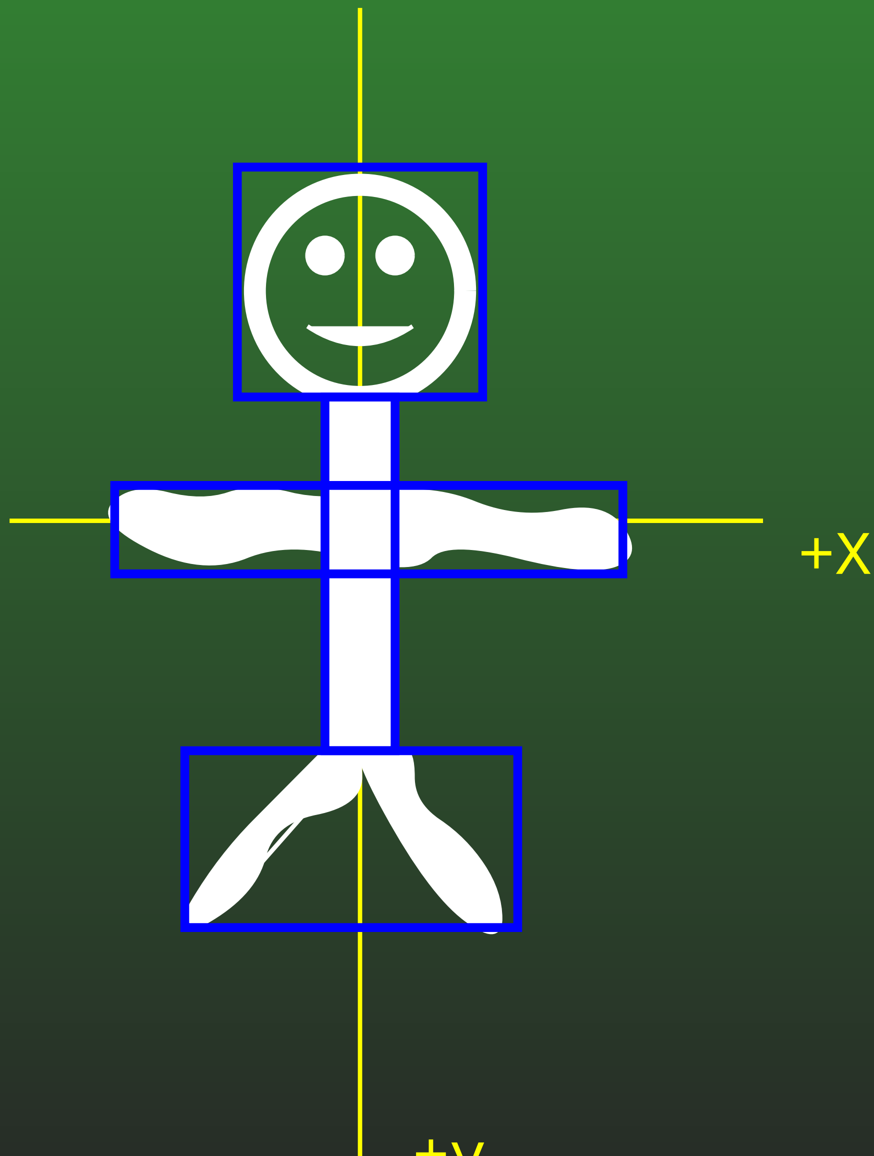
$$[p_x - c_x, c_y - p_y]$$

07-179: Examples

- Coversion from +y up to +y down
  - Points and rectangles easy to convert
  - Sprites would require a reflection
    - Can use SpriteEffects to reflect sprites
  - Best to stay in “reflected” space

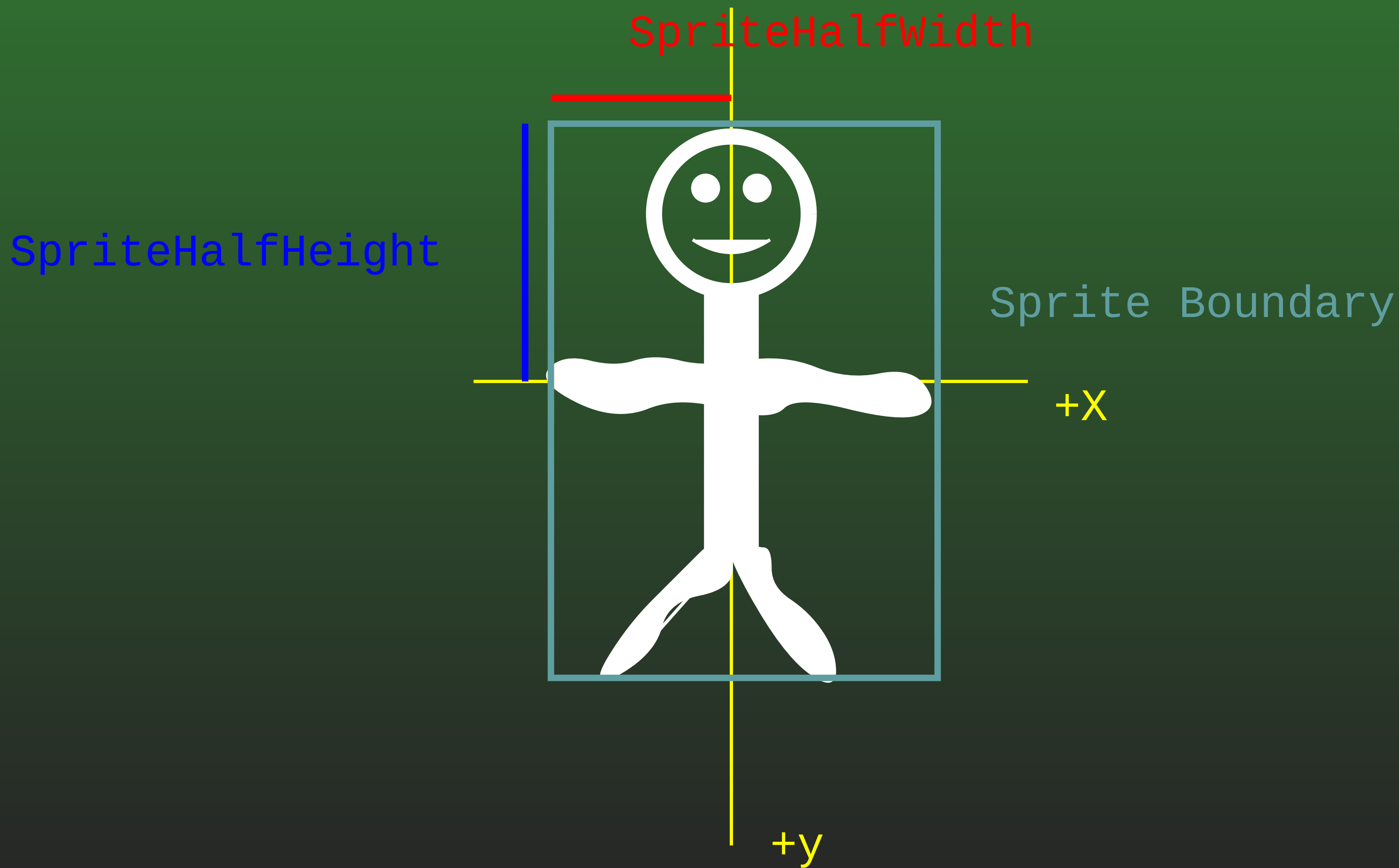
07-180: Objects with Sprites

07-181: Objects with Sprites



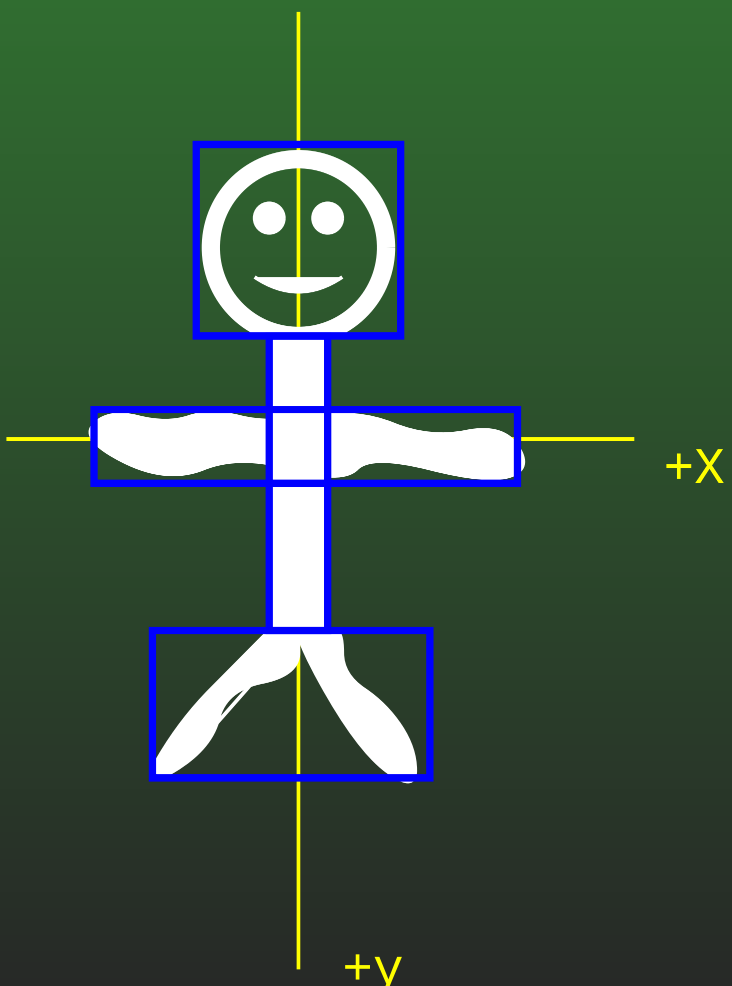
07-182: Objects with Sprites

- Center of the sprite is at [SpriteHalfWidth,SpriteHalfHeight] from top left corner of sprite



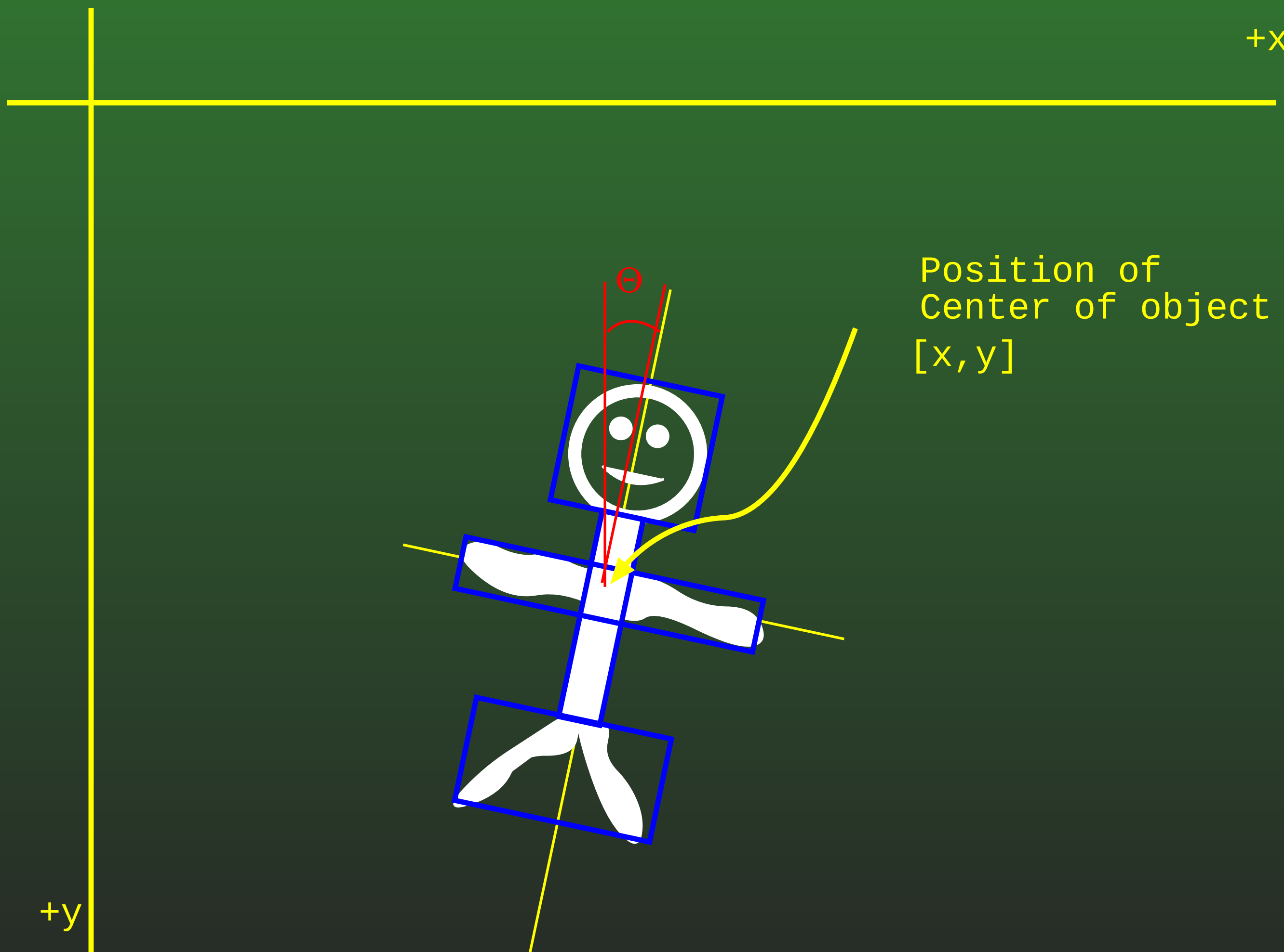
07-183: Objects with Sprites

- Boundary box locations are stored as edge points of each rectangle (4 points per)



07-184: Objects with Sprites

- Object is located at position $[x, y]$ and has rotation Θ *clockwise* (since +x is right, +y is down)



07-185: Objects with Sprites

- When dealing with the object in game logic (collistions, etc), we need to know the positions of each of the points of each rectangle in world space.
- If a vertex has position p in local (object) space, what is its position in world space?

07-186: Objects with Sprites

- When dealing with the object in game logic (collistions, etc), we need to know the positions of each of the points of each rectangle in world space.
- If a vertex has position p in local (object) space, what is its position in world space?
  - $p \begin{bmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{bmatrix} + [x, y]$

07-187: Objects with Sprites

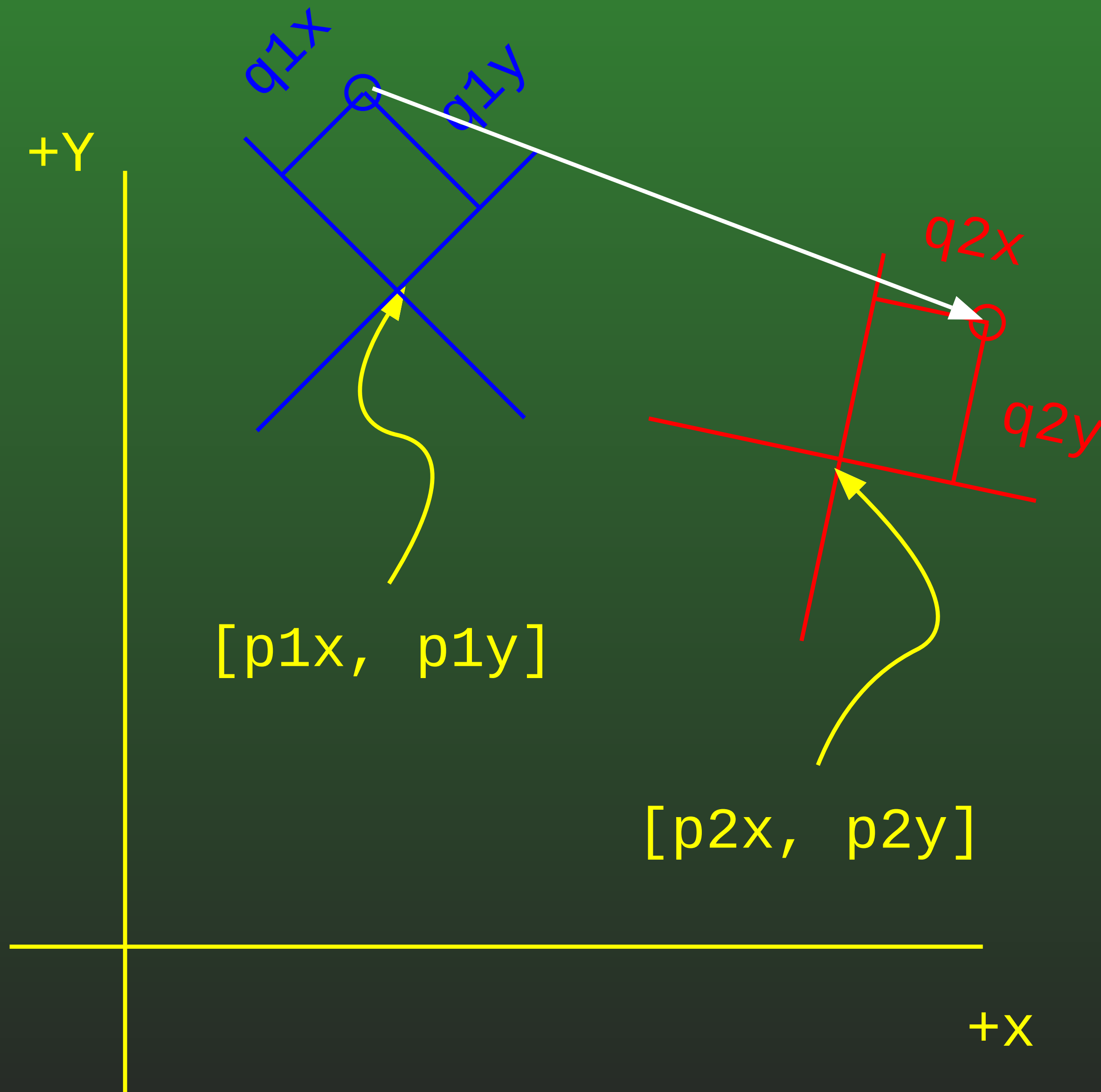
- To draw the sprite on the screen:

```
SpriteBatch sb
Vector2 pos = new Vector2(x,y);
Vector2 center = new Vector2(SpriteHalfWidh, SpriteHalfHeihgt);
sb.Draw(texture, pos, null, Color.White, theta, center, 1.0f,
        SpriteEffects.None, 0.0f);
```

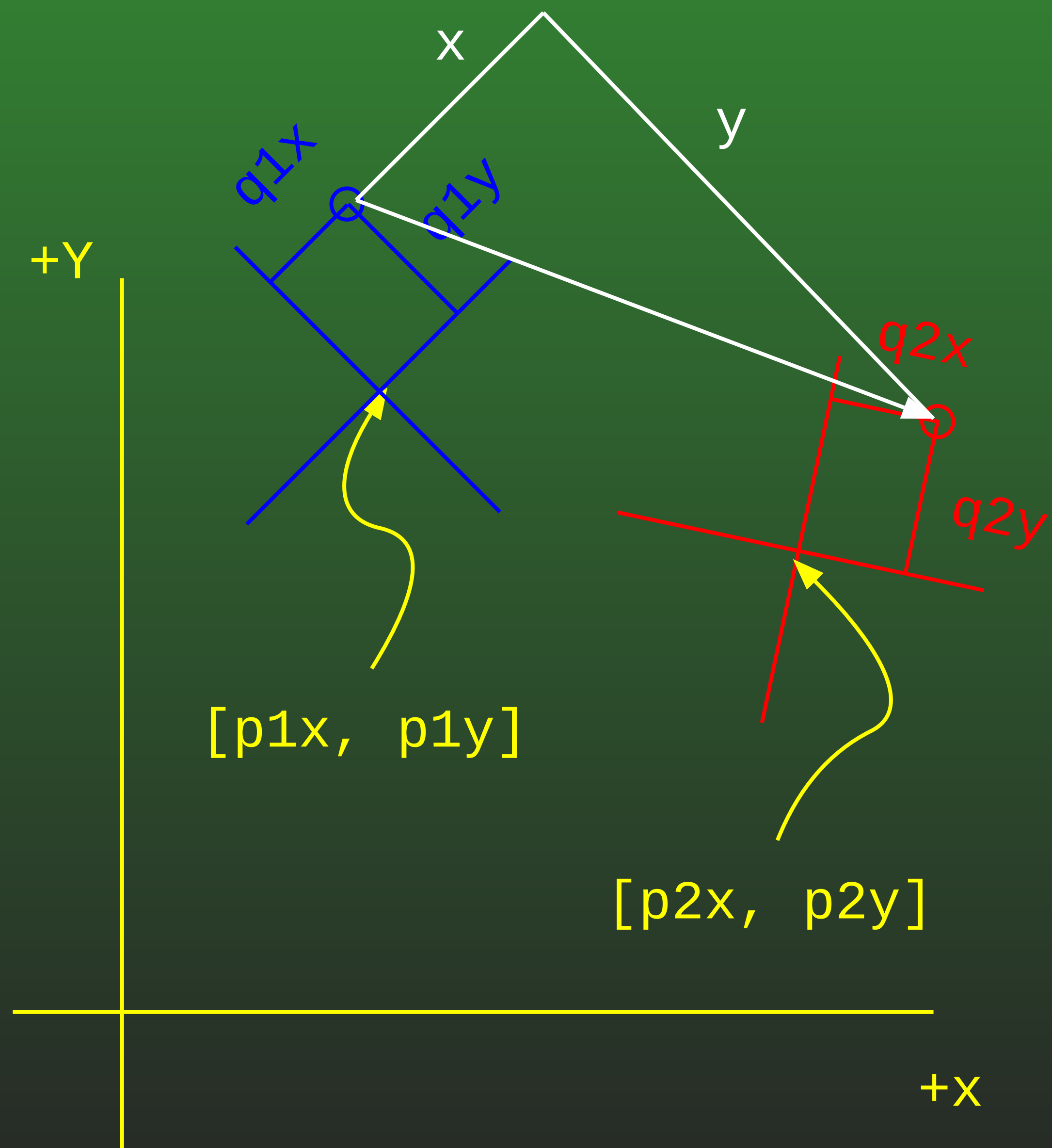
Examples

- Object1 has rotational matrix M_1 and position p_1
- Object2 has rotational matrix M_2 and position p_2
- Point q_1 is at postion $[q_{1_x}, q_{1_y}]$ in the object space of Object1
- Point q_2 is at positio $[q_{2_x}, q_{2_y}]$ in the object space of Object 2
- What is the vector from q_1 to q_2 in the object space of q_1?

Examples



07-190: Examples



Examples

- First, find the position of point q_2 in the local space of Object1.
- What's the best way to do this?

Examples

- First, find the position of point q_2 in the local space of Object1.
- What's the best way to do this?
  - Go through the world space
  - Find the position of q_2 in world space, translate to object space

Examples

- Position of q_2 in world space:
  $q_2 M_2 + p_2$
- Position of q_2 in Object1's local space
  $(q_2(global) - p_1)M_1^T = (q_2 M_2 + p_2 - p_1)M_1^T$
- Vector from q_1 to q_2 in local space of Object 1:

07-194: Examples

- Position of q_2 in world space:
  $q_2 M_2 + p_2$
- Position of q_2 in Object1's local space
  $(q_2(global) - p_1)M_1^T = (q_2 M_2 + p_2 - p_1)M_1^T$
- Vector from q_1 to q_2 in local space of Object 1:
  $(q_2 M_2 + p_2 - p_1)M_1^T - q_1$

07-195: Examples

- Given a transformation matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- How can we determine if this is a "pure rotation" – no scale, shear, reflection

07-196: Examples

- Matrix is pure rotation if:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- a = d
- -c = b
- sin(arccos(a)) = b
- What if we don't have access to arccos, sin?

Examples

- Matrix is pure rotation if:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- ac + bd = 0 [a,b] and [c,d] are perpendicular
- a*a + b*b = 1 [a,b] is unit vector
- c*c + d*d = 1 [c,d] is unit vector
- a*d - c*b = 1 No reflection

07-198: Examples

- Spaceship, in local space looks down x axis
- Position $[p_x, p_y]$
- Orientation: $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$
- Place an enemy 10 units directly in front of spaceship, pointing straight back at it
- What is position and orientation of the new spaceship?

Examples

- Spaceship, in local space looks down x axis
- Position $[p_x, p_y]$
- Orientation: $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$
- Place an enemy 10 units directly in front of spaceship, pointing straight back at it
- What is position and orientation of the new spaceship?
  - Position = $[p_x, p_y] + 10 * [a, b]$
  - Orientation: $\begin{bmatrix} -a & -b \\ -c & -d \end{bmatrix}$

Examples