Game Engineering

CS420-2016S-05

Linear Transforms

David Galles

Department of Computer Science
University of San Francisco

05-0: Matrices as Transforms

- Recall that Matrices are transforms
  - Transform vectors by rotating, scaling, shearing
  - Transform objects as well
    - Transforming every vertex in the object

05-1: Calculating Transformations

- What happens when we transform [1,0,0], [0,1,0], and [0,0,1] by

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

05-2: Calculating Transformations

- What happens when we transform [1,0,0], [0,1,0], and [0,0,1]:

$$[1, 0, 0]\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = [m_{11}, m_{12}, m_{13}]$$

$$[0, 1, 0]\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = [m_{21}, m_{22}, m_{23}]$$

$$[0, 0, 1]\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = [m_{31}, m_{32}, m_{33}]$$

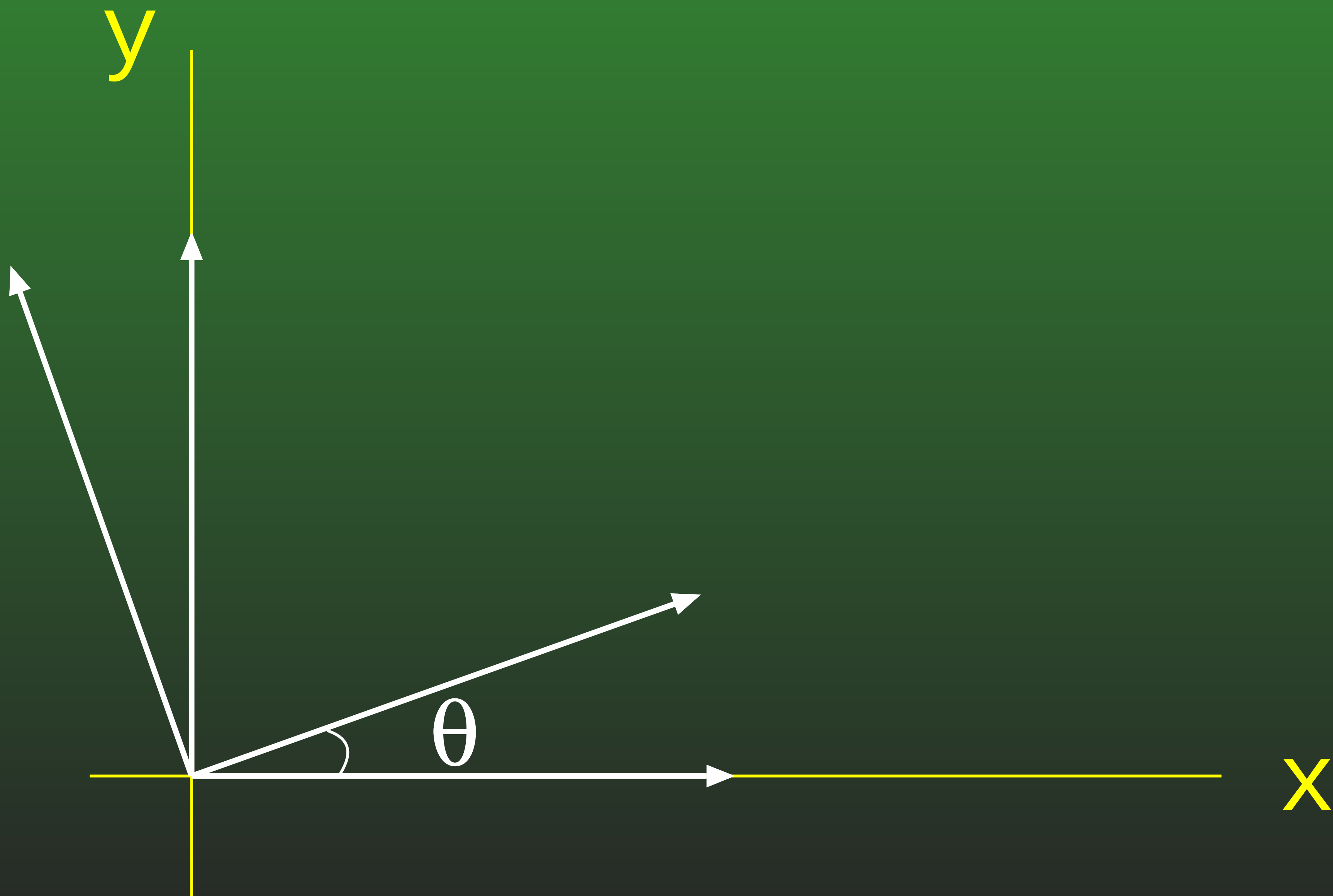
05-3: Calculating Transformations

- So, we want to make a transformation matrix
  - Matrix that, when multiplied by a vector, transforms the vector
  - (also transforms a model – just a series of points)
- To create the matrix
  - Decide what the basis vectors should look like after the transformation
  - Fill in the matrix with the new basis vectors

05-4: Rotations

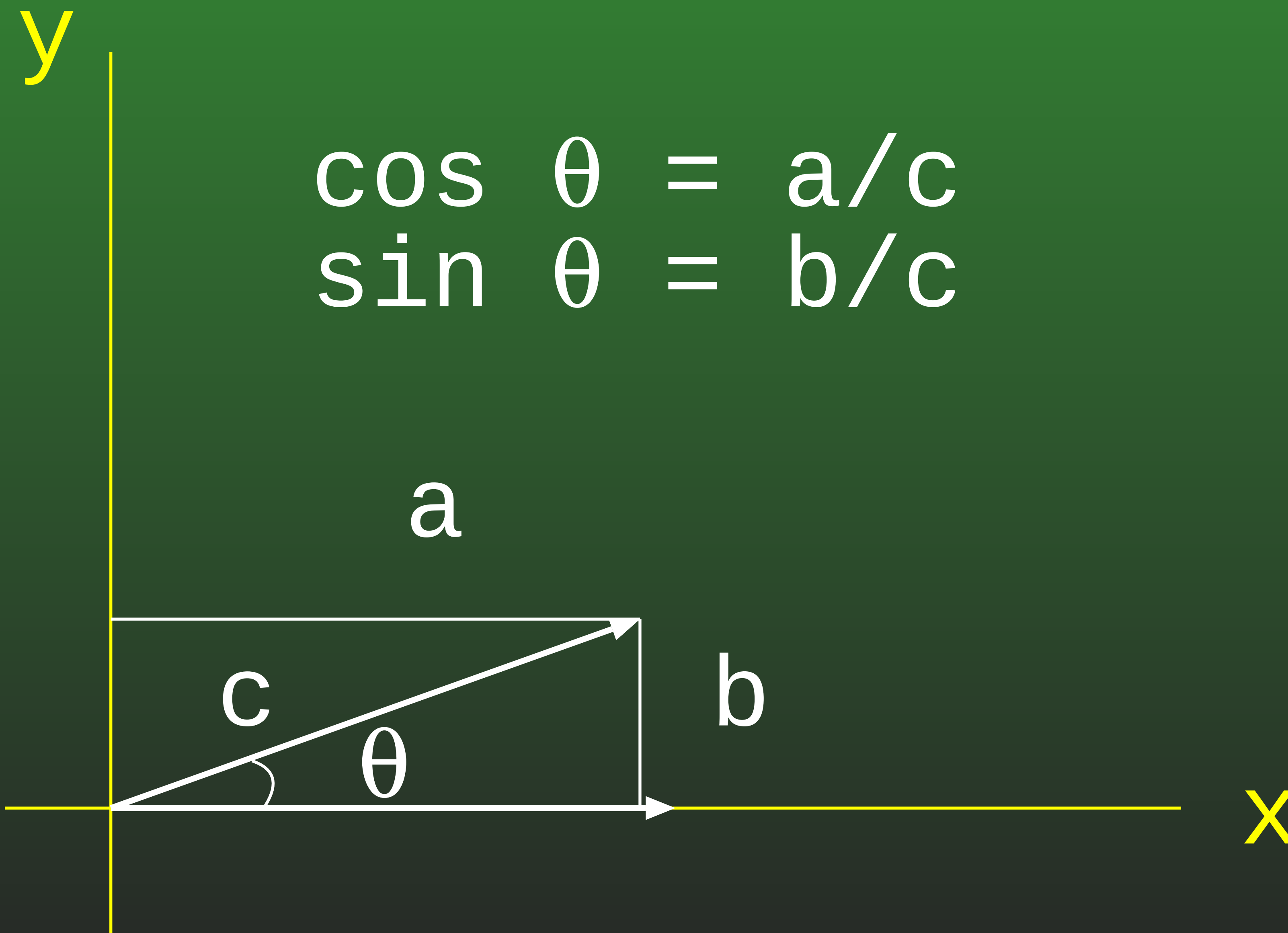
- Start with the 2D case
  - Rotate a vector θ degrees counter-clockwise
  - What do the basis vectors look like after the rotation?
  - That’s the transformation matrix!

05-5: Rotations 2D



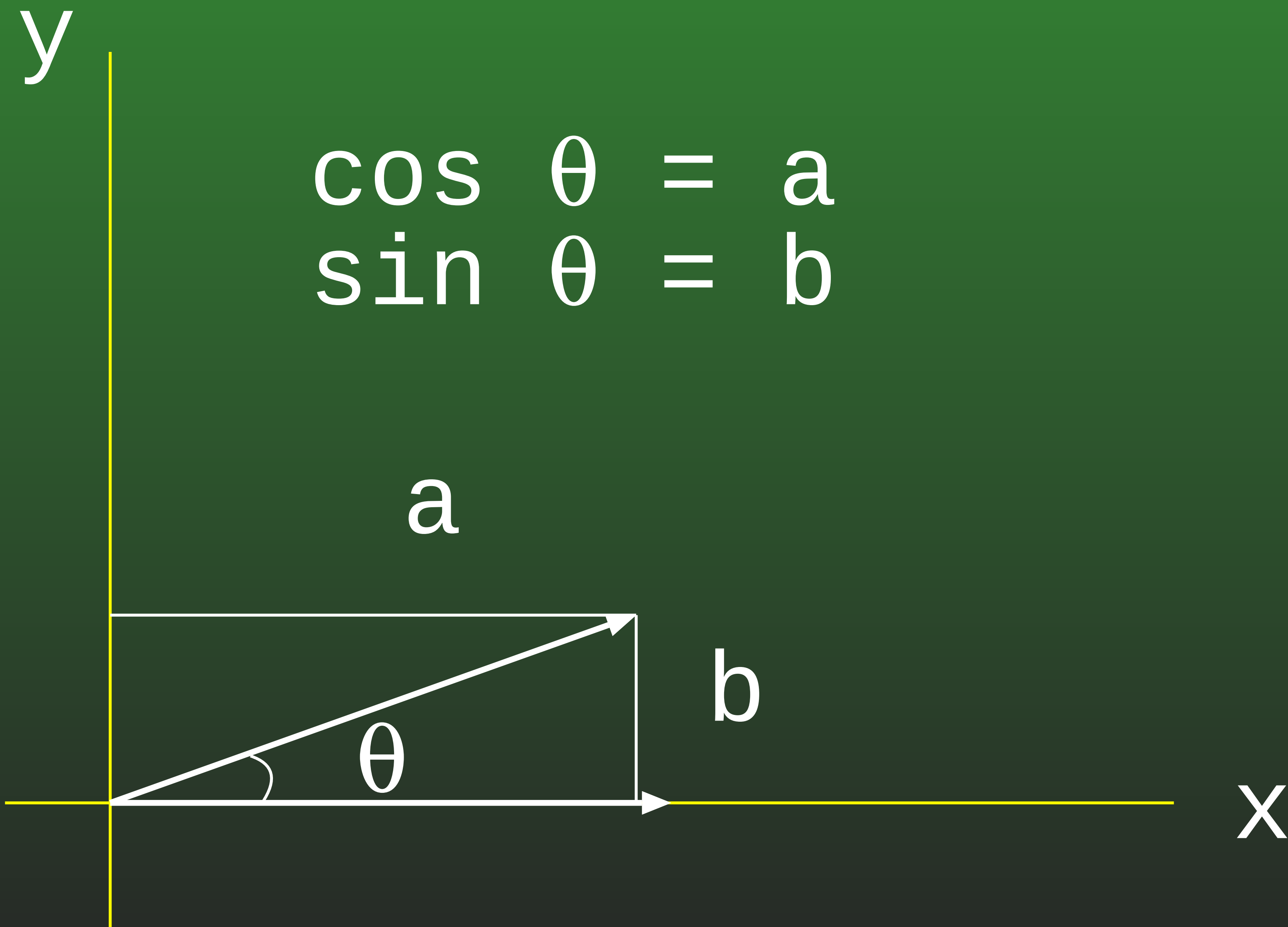
05-6: Rotations 2D

$$\cos\theta = a/c$$
$$\sin\theta = b/c$$



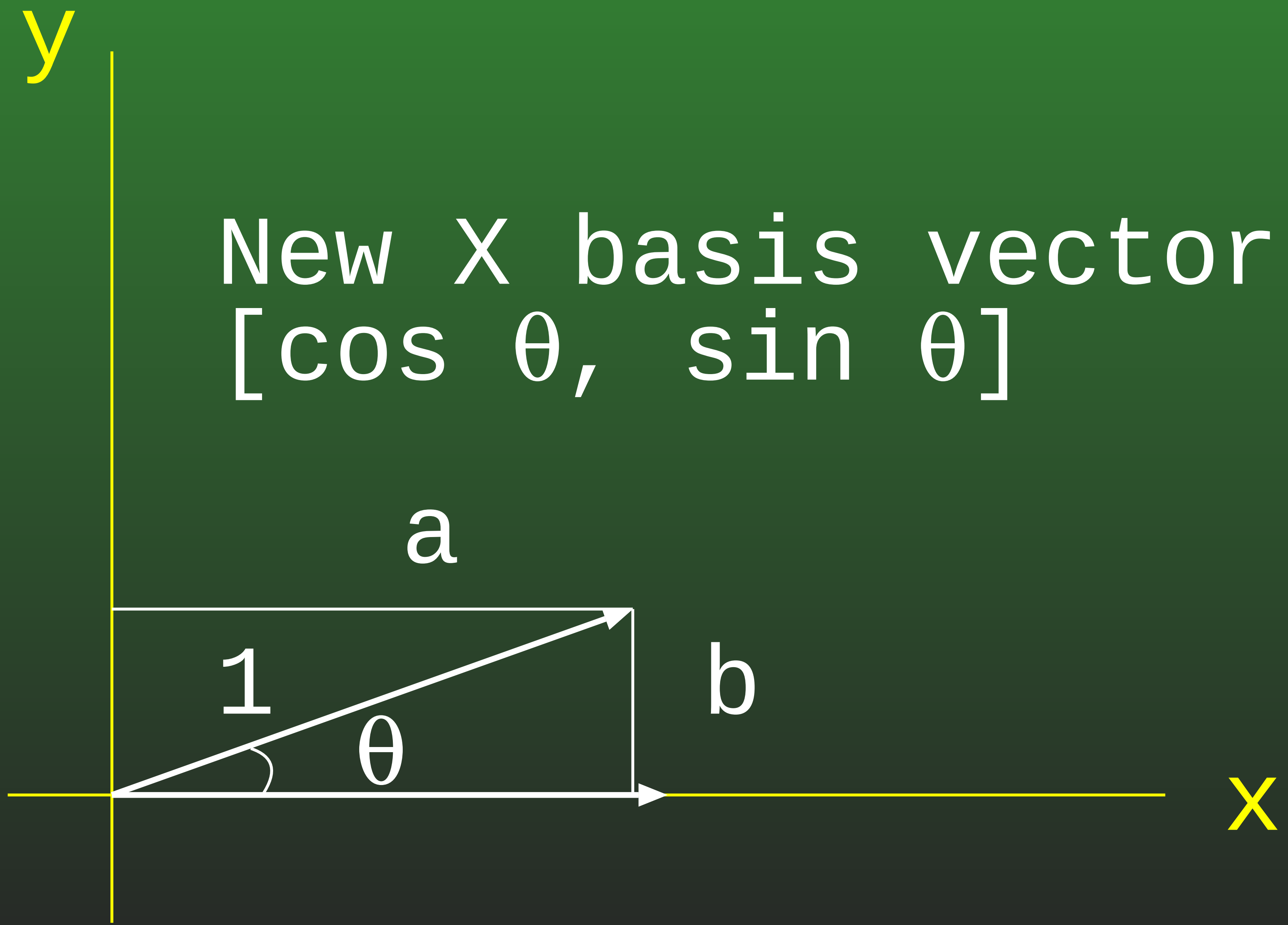
05-7: Rotations 2D

$$\cos\theta = a$$
$$\sin\theta = b$$

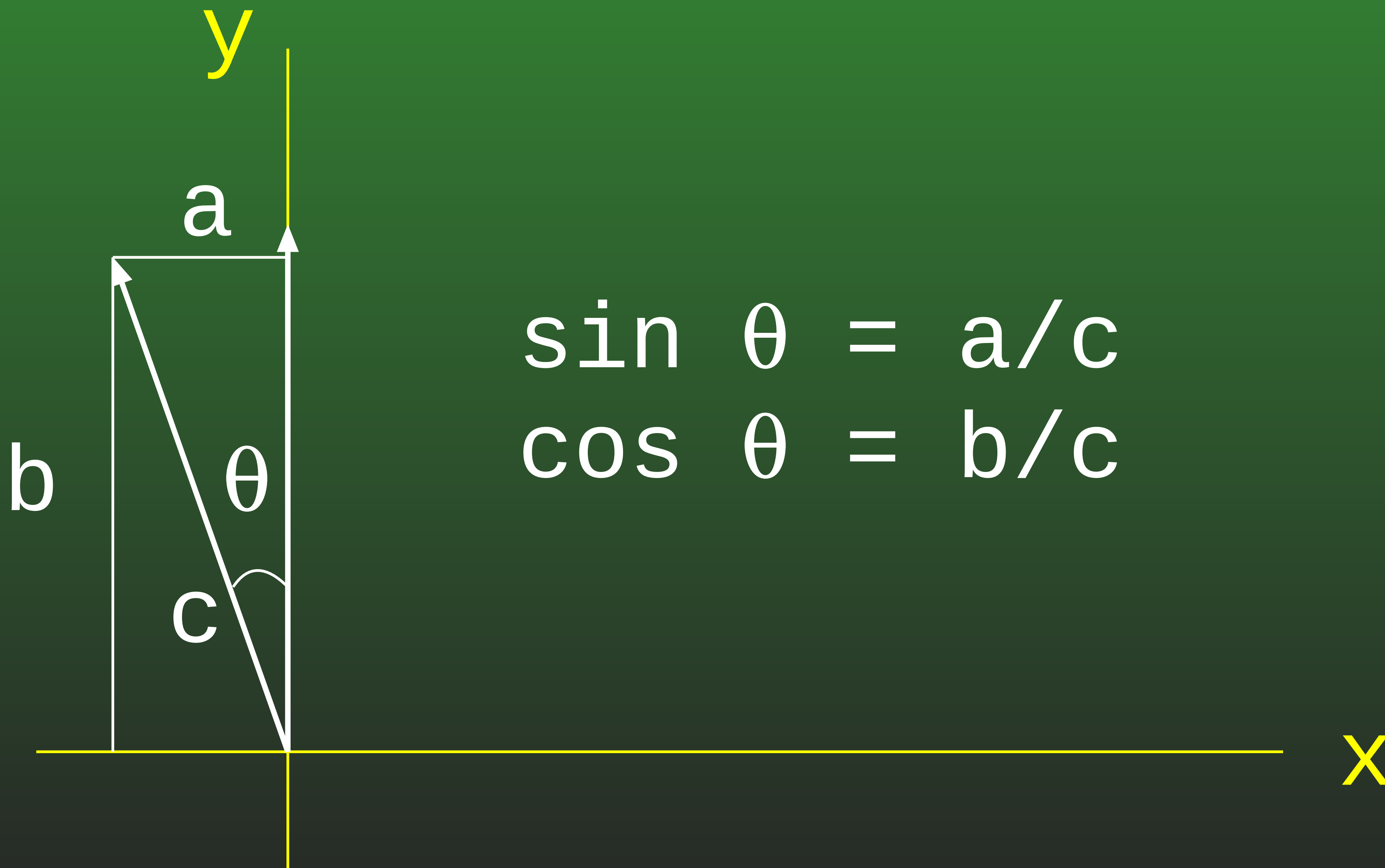


Rotations 2D

New X basis vector:
[cos θ, sin θ]



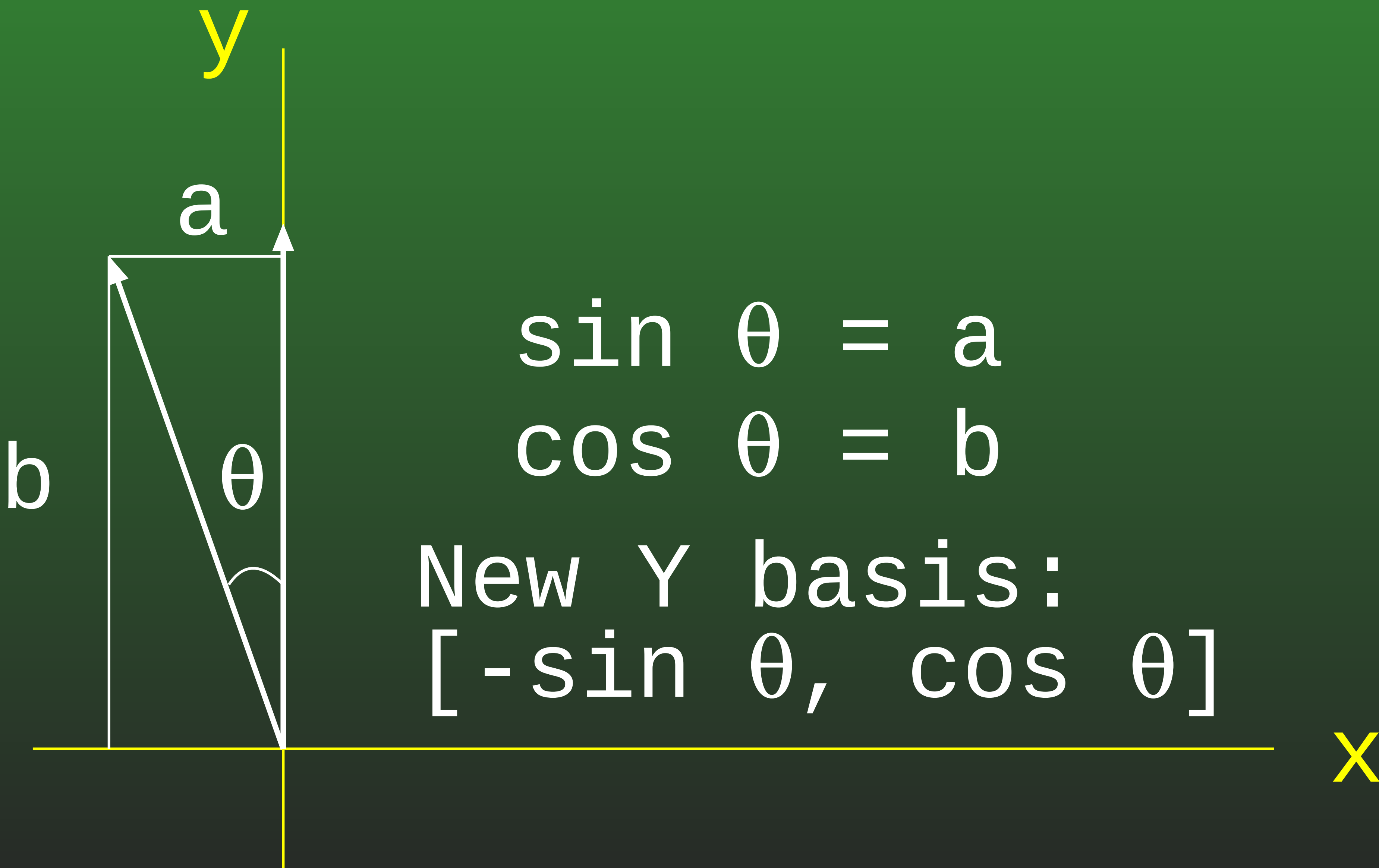
Rotations 2D



$\sin \theta = a/c$

$\cos \theta = b/c$

05-10: Rotations 2D



$$\sin\theta = a$$

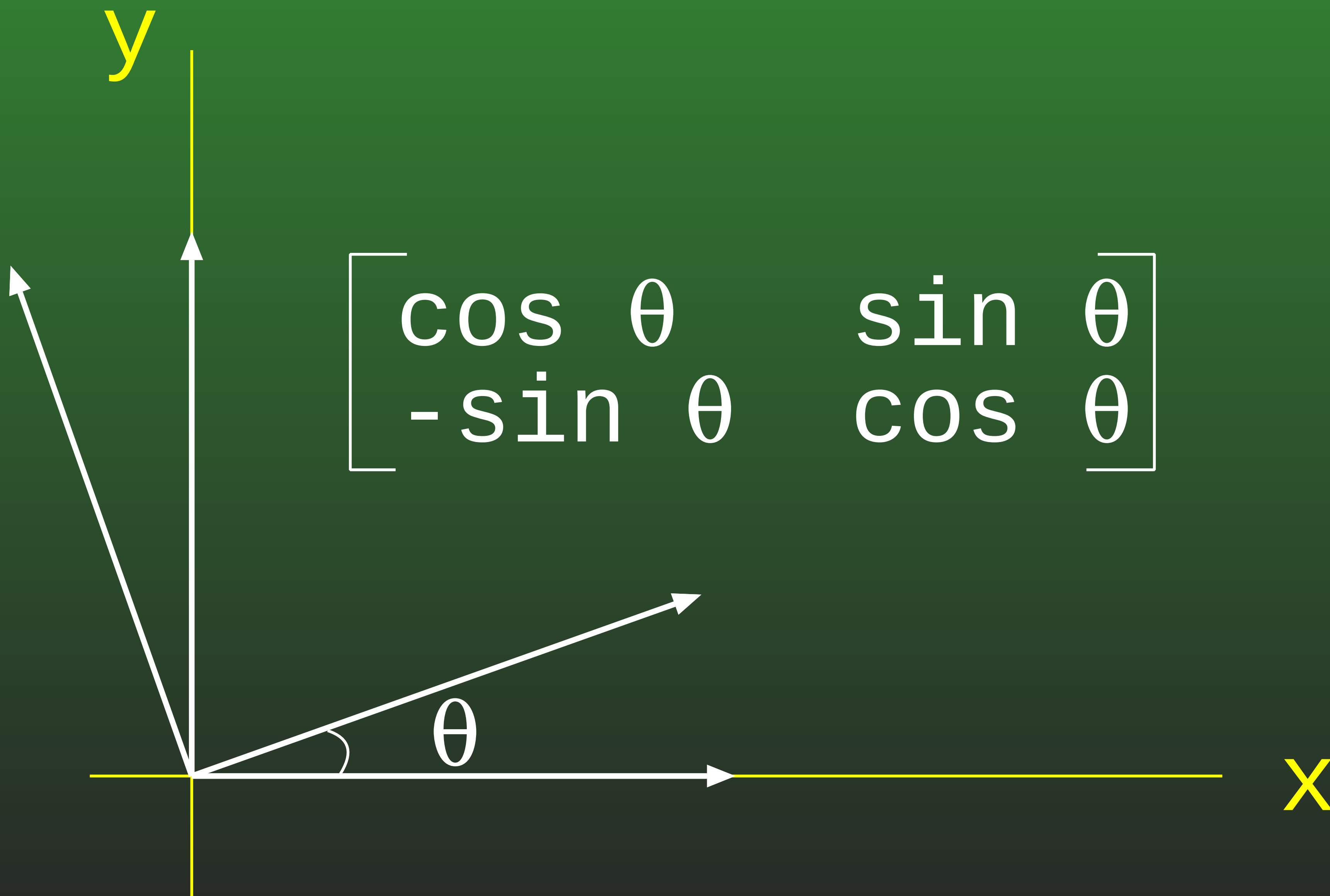
$$\cos\theta = b$$

New Y basis:

$[-\sin\theta, \cos\theta]$

05-11: Rotations 2D

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



05-12: Rotations 3D

- For rotations in 3 dimensions, we need to define:
  - The axis we are rotating around
  - The direction that we are rotating
- Can’t just use “counter-clockwise” anymore - “counter-clockwise” in relation to what?

05-13: Rotations 3D

- Rotation around the z axis
- Which direction to rotate depends upon whether you are using right-handed or left-handed coordinate system
- Select appropriate hand (right- or left-)
- Point thumb along the positive axis around which you are rotating
- Fingers curl in direction of θ

05-14: Rotations 3D

- Rotations in 3D work just like rotations in 2D
  - Determine how the basis vectors will change under the rotation
    - Need to consider 3 vectors instead of 2
  - Create a matrix using the new basis vectors
    - 3x3 instead of 2x3

05-15: Rotations 3D

- Rotating θ degrees around the z axis
  - How do the z coordinates of a vector change in this rotation?
  - In other words, what happens to the z-basis vector when rotating around the z axis?

05-16: Rotations 3D

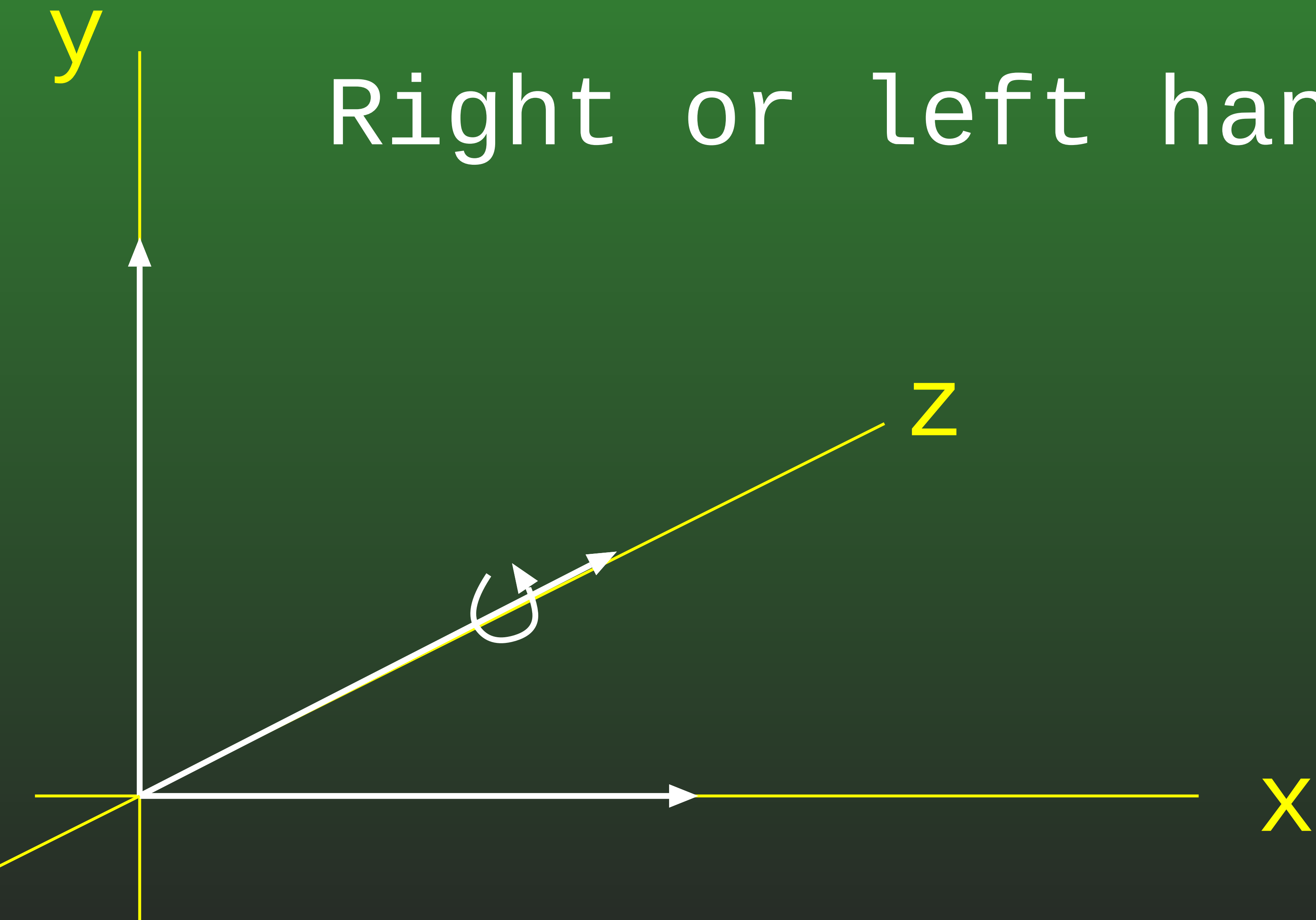
- Rotating θ degrees around the z axis
  - How do the z coordinates of a vector change in this rotation?
    - They don't!
  - In other words, what happens to the z-basis vector when rotating around the z axis?
    - It doesn't move!

05-17: Rotations 3D

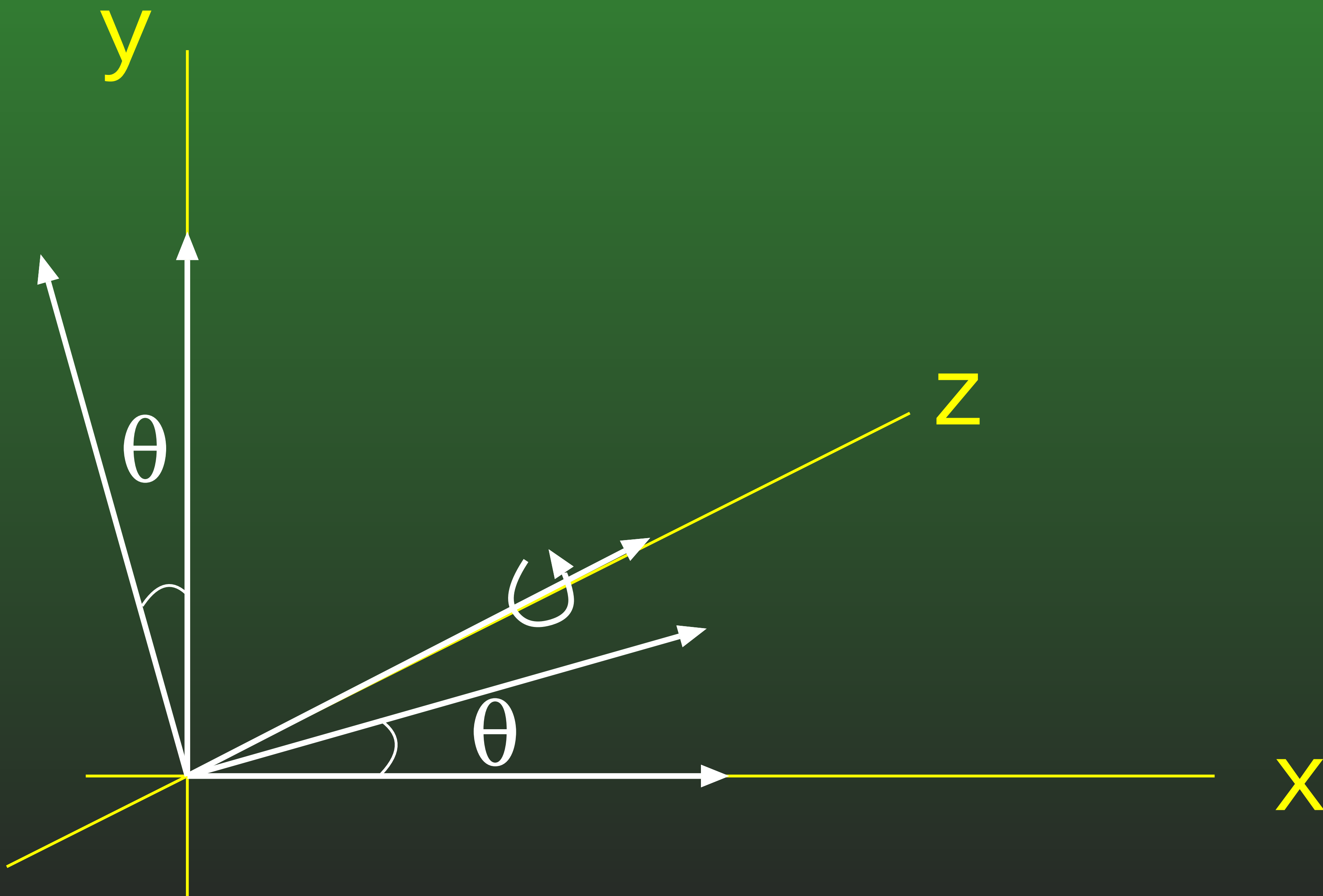
- What about the x basis vector – how does it change?

05-18: Rotations 3D

Right or left handed?



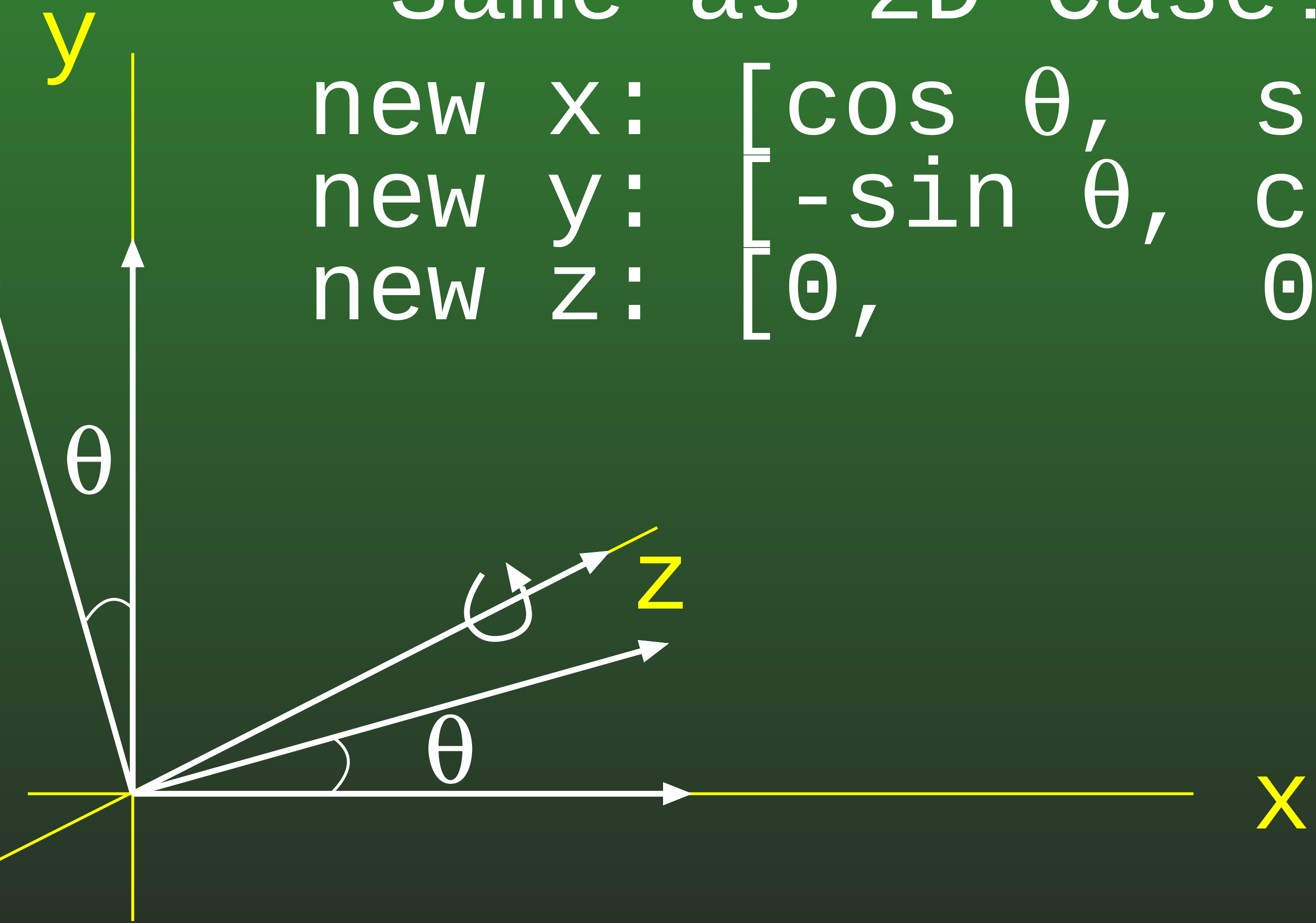
05-19: Rotations 3D



05-20: Rotations 3D

Same as 2D Case!

```
new x: [cos θ,  sin θ, 0]
new y: [-sin θ, cos θ, 0]
new z: [0,      0,     1]
```



05-21: Rotations 3D

- What about rotating around a different axis?
  - Works the same way
  - Axis being rotated around doesn't change
  - Other two axes are the 2D case

05-22: Rotations 3D

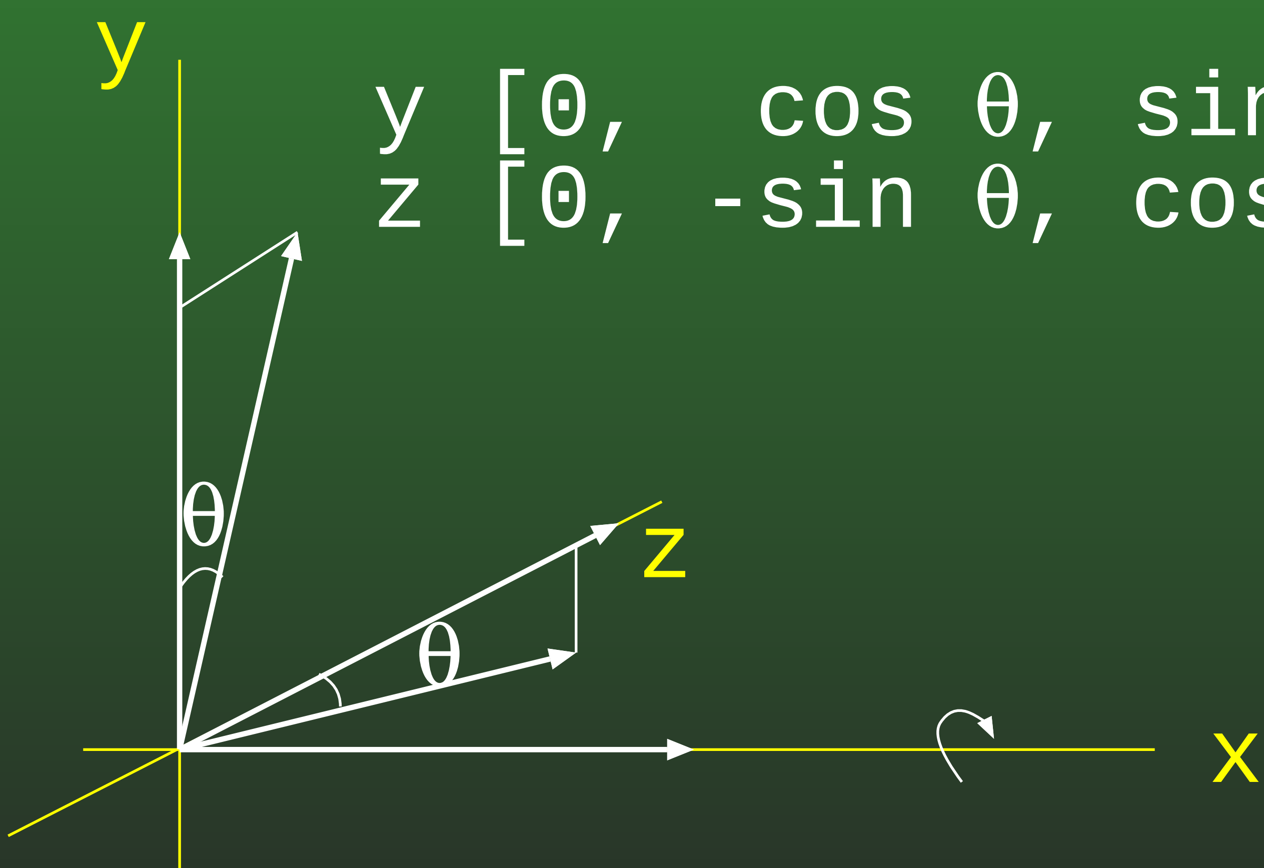
- Rotate θ degrees around the z-axis:

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

05-23: Rotations 3D

- Rotate θ degrees around the x-axis:

```
y [0,  cos θ, sin θ]
z [0, -sin θ, cos θ]
```



05-24: Rotations 3D

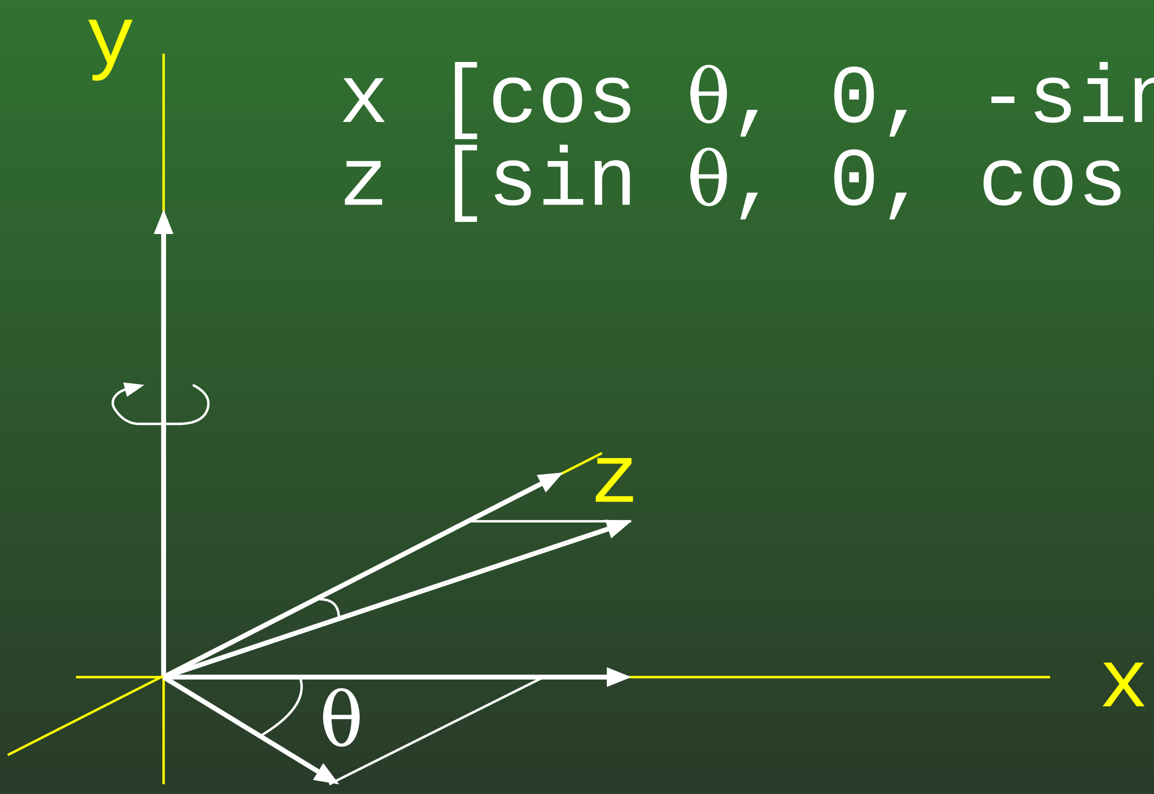
- Rotate θ degrees around the x-axis:

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

Rotations 3D

- Rotate θ degrees around the y-axis:

```
x [cos θ, 0, -sin θ]
z [sin θ, 0, cos θ]
```



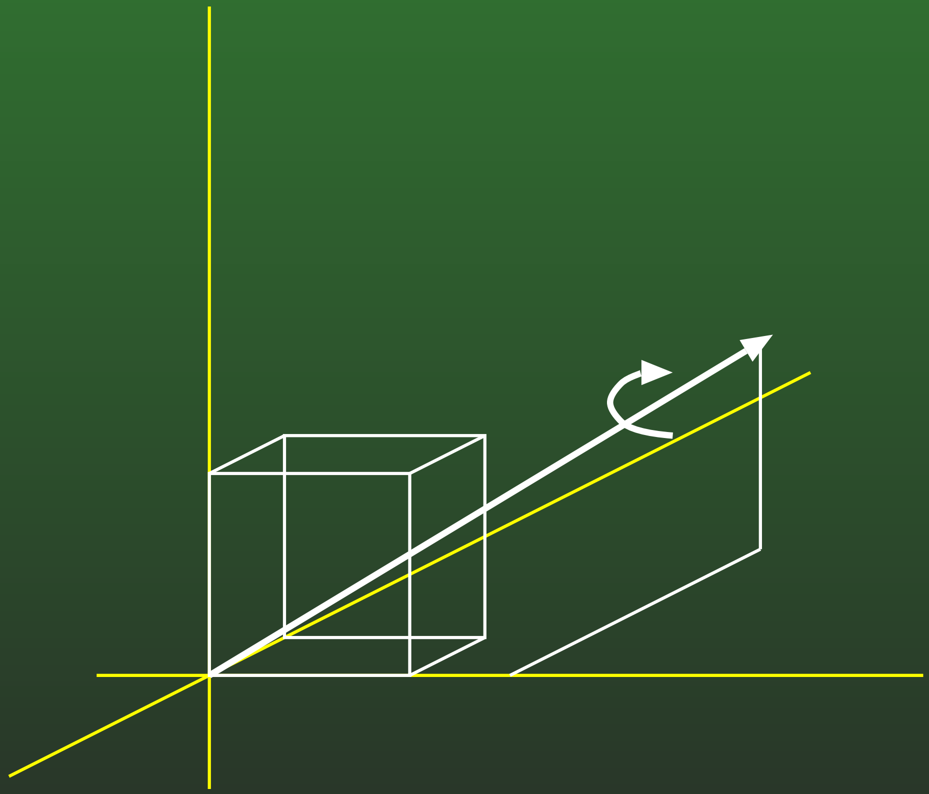
05-26: Rotations 3D

- Rotate θ degrees around the y-axis:

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

05-27: Arbitrary Axis Rotation

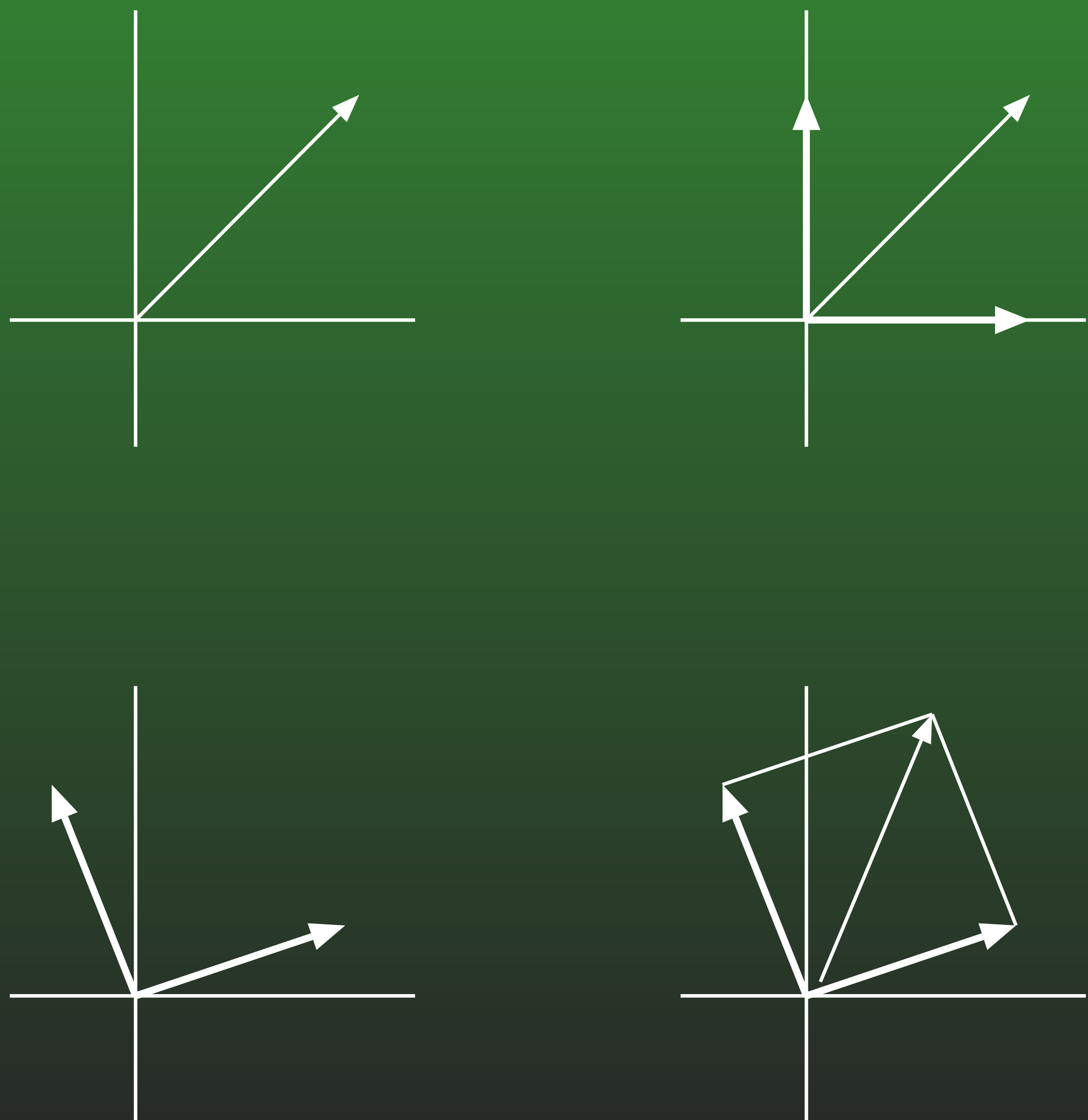
- What if we want to rotate about something other than a main axis?



05-28: Arbitrary Axis Rotation

- Use this trick to rotate a vector about aribitrary axis
  - Break the vector into two component vectors
  - Rotate the component vectors
  - Add them back together to get rotated vector
- The trick will be picking component vectors that are easy to rotate ...

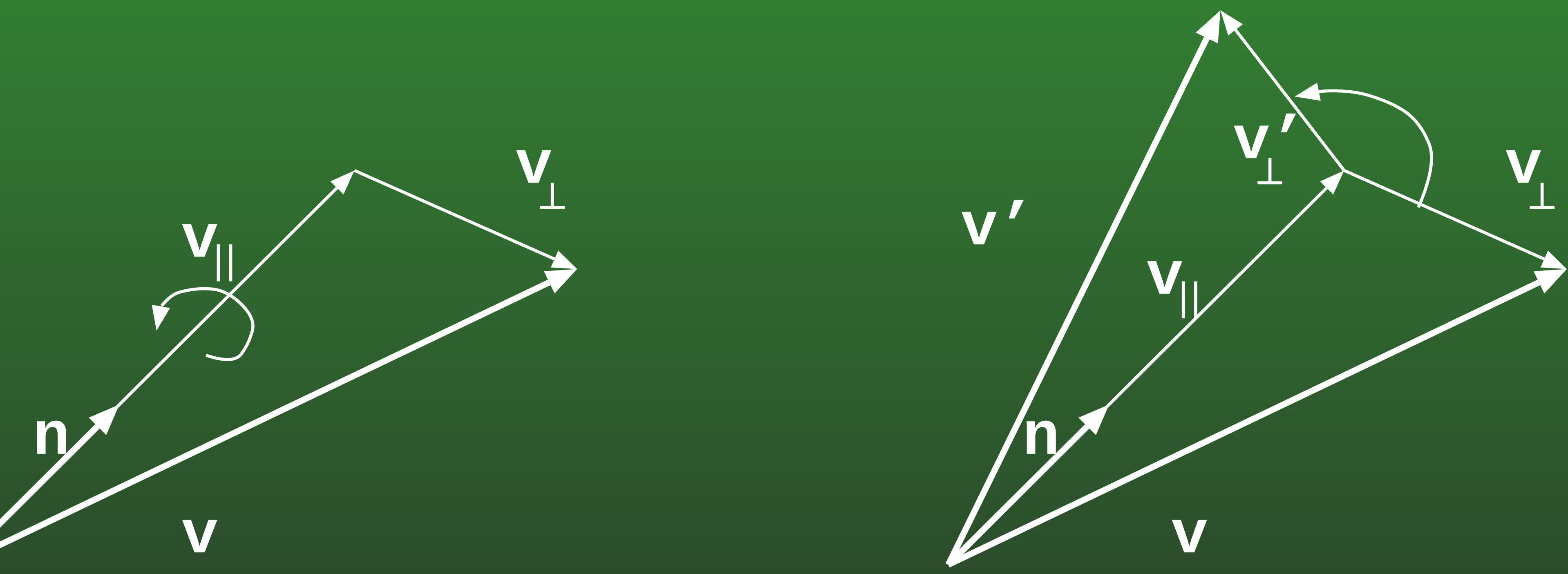
05-29: Arbitrary Axis Rotation



05-30: Arbitrary Axis Rotation

- $\mathbf{v}$ is the vector we want to rotate
- $\mathbf{n}$ is the vector we want to rotate around (assume n is a unit vector)
- Break $\mathbf{v}$ into $v_{\|}$ and $v_{\perp}$
- Rotate $v_{\|}$ and $v_{\perp}$ around n
- Add them back together to get rotated $\mathbf{v}$

05-31: Arbitrary Axis Rotation



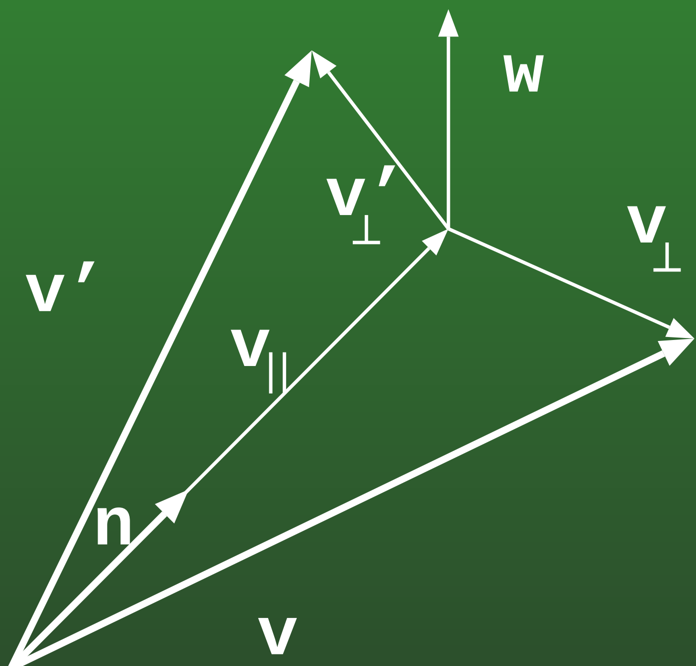
05-32: Arbitrary Axis Rotation

- $\mathbf{v}$ is the vector we want to rotate
- $\mathbf{n}$ is the vector we want to rotate around (assume n is a unit vector)
- Break $\mathbf{v}$ into $\mathbf{v}_{\|}$ and $\mathbf{v}_{\perp}$
- What is the result of rotating $\mathbf{v}_{\|}$ around $\mathbf{n}$?

Arbitrary Axis Rotation

- $\mathbf{v}$ is the vector we want to rotate
- $\mathbf{n}$ is the vector we want to rotate around (assume n is a unit vector)
- Break $\mathbf{v}$ into $\mathbf{v}_{\|}$ and $\mathbf{v}_{\perp}$
- What is the result of rotating $\mathbf{v}_{\|}$ around $\mathbf{n}$?
  - $v_{\|}$ doesn't change!

05-34: Arbitrary Axis Rotation



- Create $\mathbf{w}$, perpendicular to both $\mathbf{v}_{\|}$ and $\mathbf{v}_{\perp}$
  - $\mathbf{w}$ is the same length as $\mathbf{v}_{\perp}$
  - $\mathbf{w}$ perpendicular to $\mathbf{n}$
  - $\mathbf{w}$, $\mathbf{v}_{\perp}$ and $\mathbf{v}'_{\perp}$ ($\mathbf{v}_{\perp}$ after rotation) are all in the same plane.

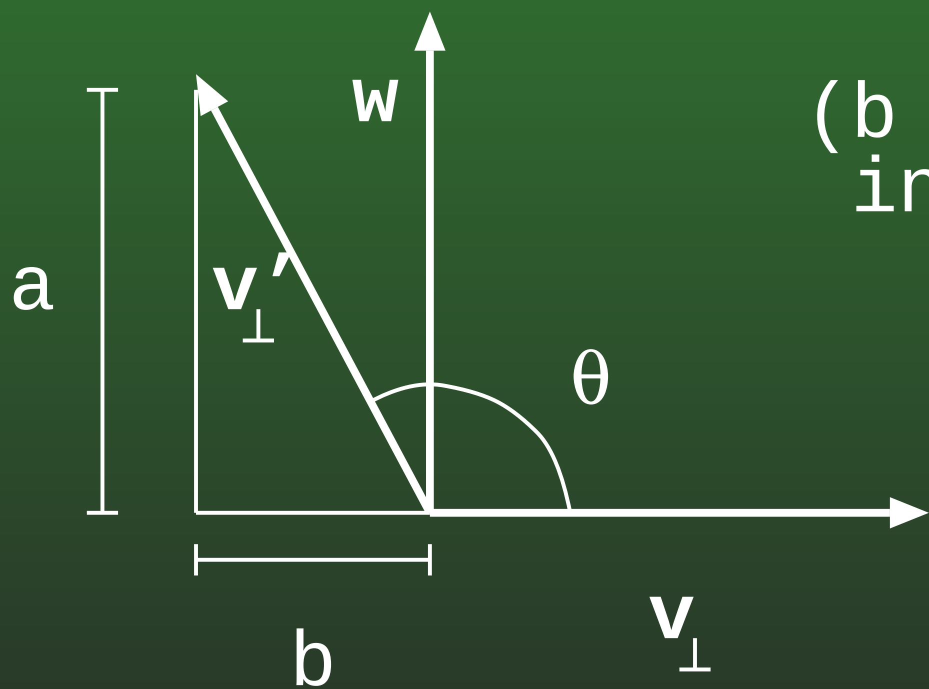
05-35: Arbitrary Axis Rotation

- Vector $v_\perp$ is rotating through the plane containing $\mathbf{w}$
- Since rotation is constrained to this one plane, back in the 2D case!

05-36: Arbitrary Axis Rotation

$$\sin \theta = a / ||v'_\perp|| = a / ||w||$$

$$\cos \theta = b / ||v'_\perp|| = b / ||v_\perp||$$



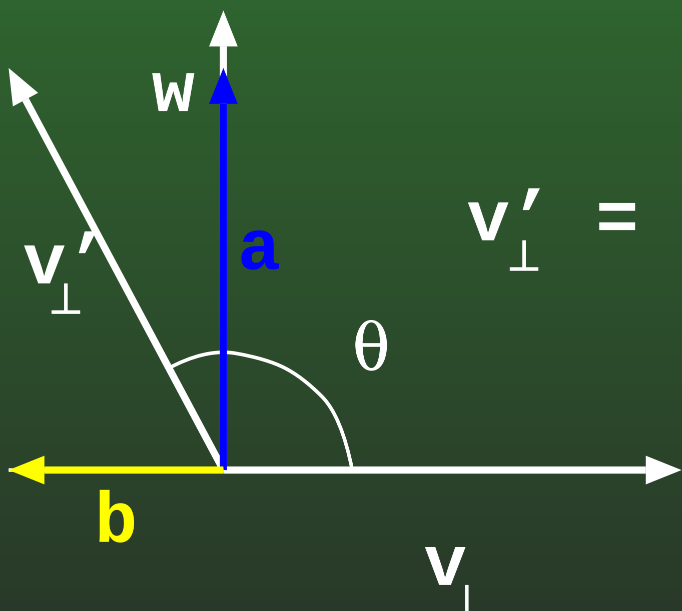
(b is negative in ths example)

05-37: Arbitrary Axis Rotation

$$\mathbf{v}'_{\perp} = \mathbf{a} + \mathbf{b}$$

$$\mathbf{a} = \sin\theta * \mathbf{w}$$

$$\mathbf{b} = \cos\theta * \mathbf{v}_{\perp}$$

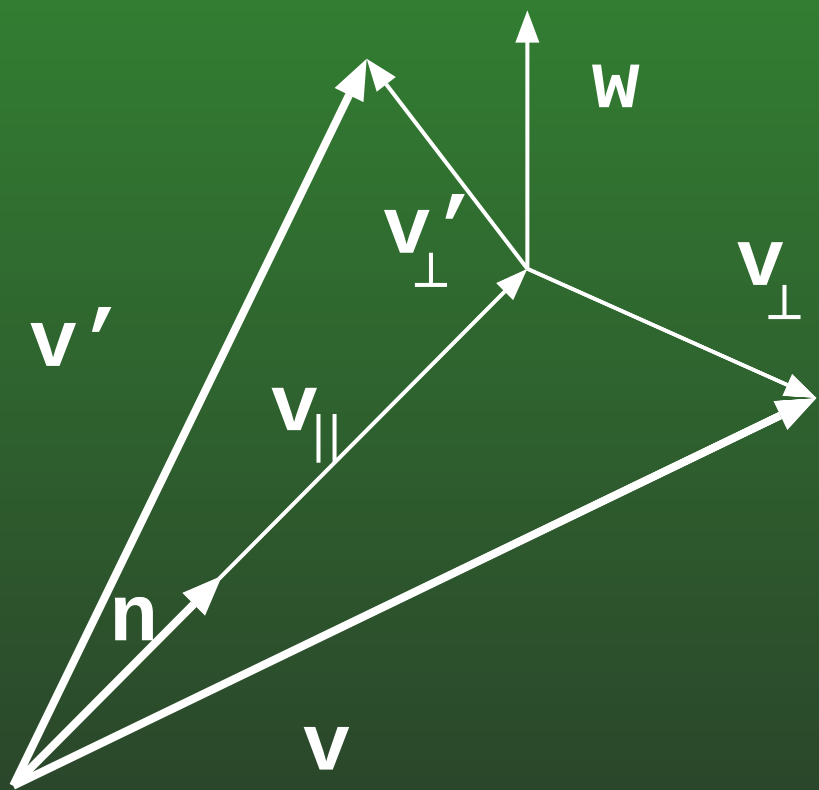


$$\mathbf{v}'_{\perp} = \cos\theta * \mathbf{v}_{\perp} + \sin\theta * \mathbf{w}$$

05-38: Arbitrary Axis Rotation

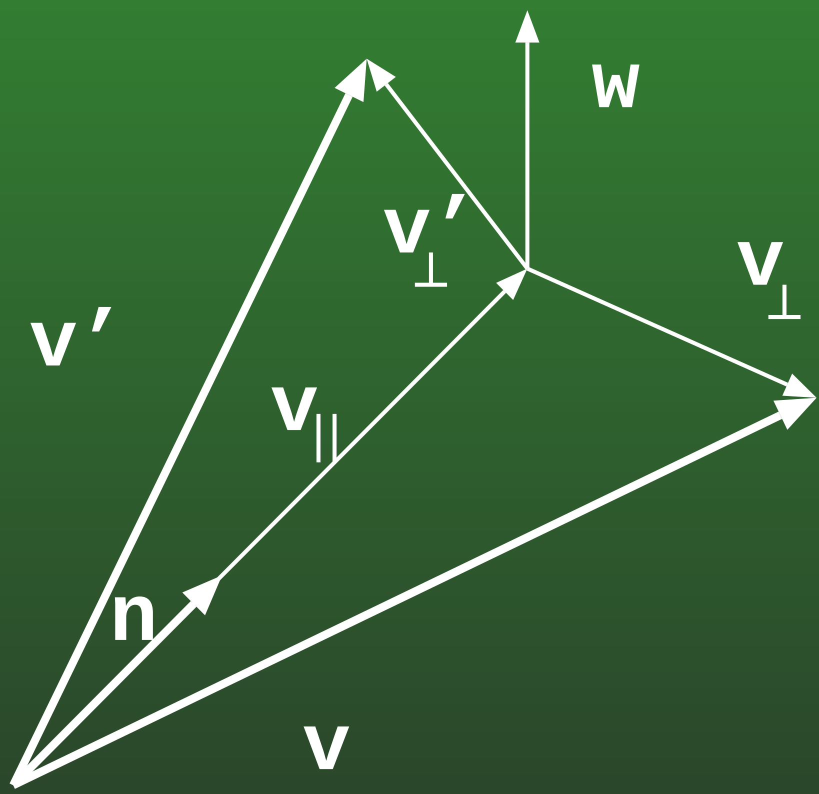
- So, we have:
  - $\mathbf{v}' = \mathbf{v}'_{\parallel} + \mathbf{v}'_{\perp}$
  - $\mathbf{v}'_{\parallel} = \mathbf{v}_{\parallel}$
  - $\mathbf{v}'_{\perp} = \cos\theta \mathbf{v}_{\perp} + \sin\theta \mathbf{w}$
- All we need to do now is find $\mathbf{v}_{\parallel}, \mathbf{v}_{\perp}$ and $\mathbf{w}$.

05-39: Arbitrary Axis Rotation



- What is $\mathbf{v}_{\|}$?
  - That is, the projection of $\mathbf{v}$ onto $\mathbf{n}$?

05-40: Arbitrary Axis Rotation



- What is $\mathbf{v}_{\|}$?
- $\mathbf{v}_{\|} = (\mathbf{v} \cdot \mathbf{n})\mathbf{n}$

05-41: Arbitrary Axis Rotation

- Once we have $\mathbf{v}_{\|}$, finding $\mathbf{v}_{\perp}$ is easy. Why?

05-42: Arbitrary Axis Rotation

- Once we have $\mathbf{v}_{\parallel}$, finding $\mathbf{v}_{\perp}$ is easy.
  - $\mathbf{v} = \mathbf{v}_{\parallel} + \mathbf{v}_{\perp}$
  - $\mathbf{v}_{\perp} = \mathbf{v} - \mathbf{v}_{\parallel}$

05-43: Arbitrary Axis Rotation

- $\mathbf{w}$ is perpendicular to both $\mathbf{v}_\perp$ and $\mathbf{n}$
- $\mathbf{n}$ is a unit vector
- $\mathbf{w}$ has the same magnitude as $\mathbf{v}_\perp$
- What is $\mathbf{w}$?

05-44: Arbitrary Axis Rotation

- $\mathbf{w}$ is perpendicular to both $\mathbf{v}_\perp$ and $\mathbf{n}$
- $\mathbf{n}$ is a unit vector
- $\mathbf{w}$ has the same magnitude as $\mathbf{v}_\perp$
- What is $\mathbf{w}$?
  - $\mathbf{n} \times \mathbf{v}_\perp$
  - Mutually perpendicular (left-handed system in diagrams)
  - $||\mathbf{n} \times \mathbf{v}_\perp|| = ||\mathbf{n}||||\mathbf{v}_\perp|| \sin\theta = ||\mathbf{v}_\perp||$

05-45: Arbitrary Axis Rotation

- $\mathbf{v}' = \mathbf{v}'_{\parallel} + \mathbf{v}'_{\perp}$
- $\mathbf{v}'_{\parallel} = (\mathbf{v} \cdot \mathbf{n})\mathbf{n}$
- $\mathbf{v}'_{\perp} = \cos\theta \mathbf{v}_{\perp} + \sin\theta \mathbf{w}$
- $\mathbf{v}_{\perp} = \mathbf{v} - \mathbf{v}_{\parallel}$
- $\mathbf{w} = \mathbf{n} \times \mathbf{v}_{\perp}$
- $\mathbf{v}' = \cos\theta(\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n}) + \sin\theta(\mathbf{n} \times \mathbf{v}) + (\mathbf{v} \cdot \mathbf{n})\mathbf{n}$ (whew!)

05-46: Arbitrary Axis Rotation

- OK, so we've found out how to rotate a single vector around an arbitrary axis.
- How do we create a rotation matrix that will do this rotation?
    - In general, how do we create a rotation matrix – or any transformation matrix, for that matter

05-47: Arbitrary Axis Rotation

- How to create a transformation matrix:
  - Transform each of the axis vectors
  - Put them together into a matrix (either as rows or columns, depending upon whether you are using row- or column transformation matricies)
- So, for $v = [1, 0, 0], [0, 1, 0]$ and $[0, 0, 1]$, calculate:

$$\cos\theta(\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n}) + \sin\theta(\mathbf{n} \times \mathbf{v}) + (\mathbf{v} \cdot \mathbf{n})\mathbf{n}$$

05-48: Arbitrary Axis Rotation

- $\mathbf{v} = [1, 0, 0]$
- $\mathbf{v}' = \cos\theta(\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n}) + \sin\theta(\mathbf{n} \times \mathbf{v}) + (\mathbf{v} \cdot \mathbf{n})\mathbf{n}$
  - $\cos\theta([1, 0, 0] - ([1, 0, 0] \cdot [n_x, n_y, n_z])[n_x, n_y, n_z])$
  - $\cos\theta([1, 0, 0] - (n_x)[n_x, n_y, n_z])$
  - $\cos\theta([1 - n_x^2, -n_x n_y, -n_x n_z])$

05-49: Arbitrary Axis Rotation

- $\mathbf{v} = [1, 0, 0]$
- $\mathbf{v}' = \cos\theta(\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n}) + \sin\theta(\mathbf{n} \times \mathbf{v}) + (\mathbf{v} \cdot \mathbf{n})\mathbf{n}$
  - $\sin\theta(\mathbf{n} \times \mathbf{v})$
  - $\sin\theta([n_x, n_y, n_z] \times [1, 0, 0])$
  - $\sin\theta([0, n_z, -n_z]$

05-50: Arbitrary Axis Rotation

- $\mathbf{v} = [1, 0, 0]$
- $\mathbf{v}' = \cos\theta(\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n}) + \sin\theta(\mathbf{n} \times \mathbf{v}) + (\mathbf{v} \cdot \mathbf{n})\mathbf{n}$
    - $(\mathbf{v} \cdot \mathbf{n})\mathbf{n}$
    - $([1, 0, 0] \cdot [n_x, n_y, n_z])[n_x, n_y, n_z]$
    - $n_x[n_x, n_y, n_z]$
    - $[n_x^2, n_x n_y, n_x n_z]$

05-51: Arbitrary Axis Rotation

- Add them all up, and simplify, to get

$$[n_x^2(1-\cos\theta)+\cos\theta, n_x n_y(1-\cos\theta)+n_z\sin\theta, n_x n_z(1-\cos\theta)-n_y\sin\theta]$$

05-52: Arbitrary Axis Rotation

- Do the same thing for the other two basis vectors, and get:
- y basis vector

  $[n_x n_y(1-\cos\theta) - n_z \sin\theta, n_y^2(1-\cos\theta) + \cos\theta, n_y n_z(1-\cos\theta) + n_x \sin\theta]$
- z basis vector

  $[n_x n_z(1-\cos\theta) + n_y \sin\theta, n_y n_z(1-\cos\theta) - n_x \sin\theta, n_z^2(1-\cos\theta) + \cos\theta]$

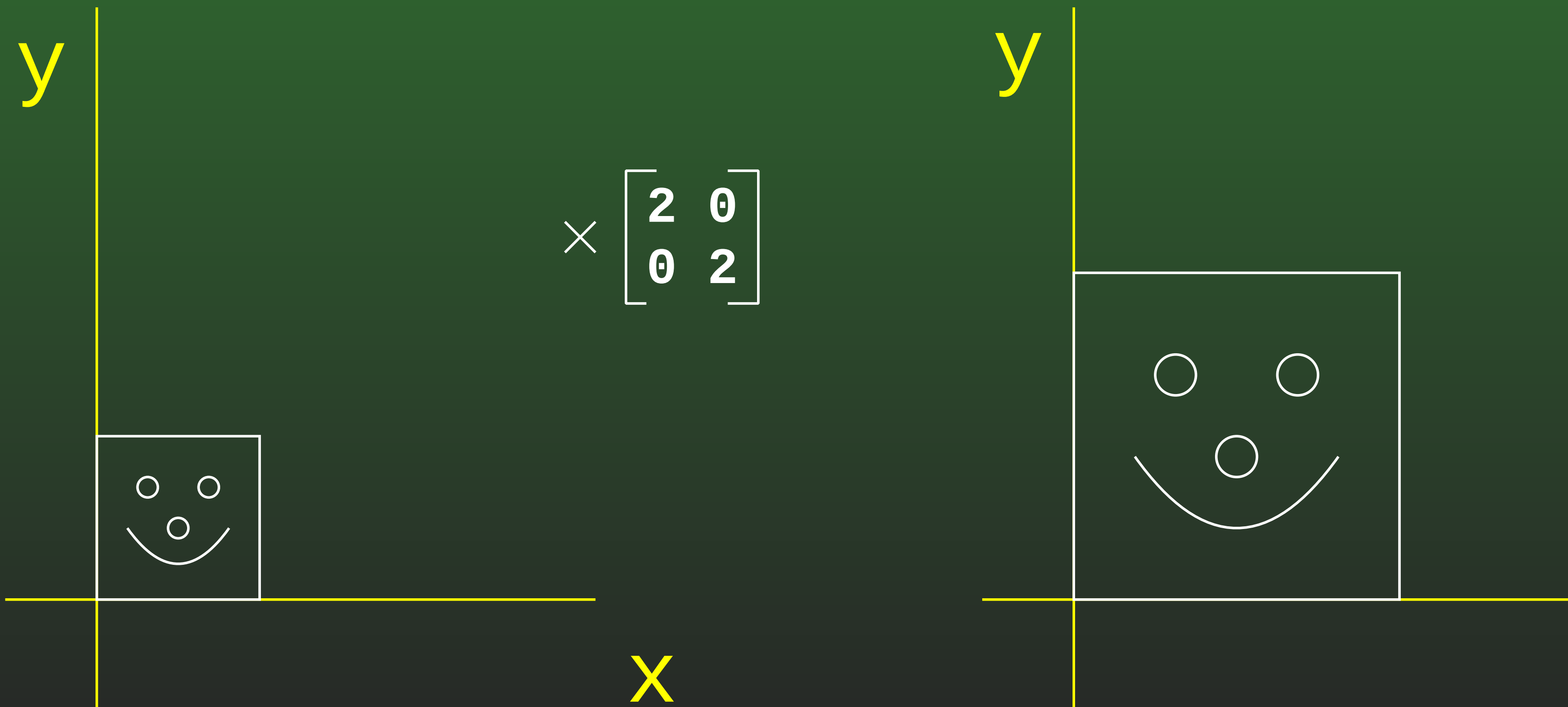
05-53: Arbitrary Axis Rotation

- Giving the final matrix:

$$\begin{bmatrix} n_x^2(1-\cos\theta)+\cos\theta & n_xn_y(1-\cos\theta)+n_z\sin\theta & n_xn_z(1-\cos\theta)-n_y\sin\theta \\ n_xn_y(1-\cos\theta)-n_z\sin\theta & n_y^2(1-\cos\theta)+\cos\theta & n_yn_z(1-\cos\theta)+n_x\sin\theta \\ n_xn_z(1-\cos\theta)+n_y\sin\theta & n_yn_z(1-\cos\theta)-n_x\sin\theta & n_z^2(1-\cos\theta)+\cos\theta \end{bmatrix}$$

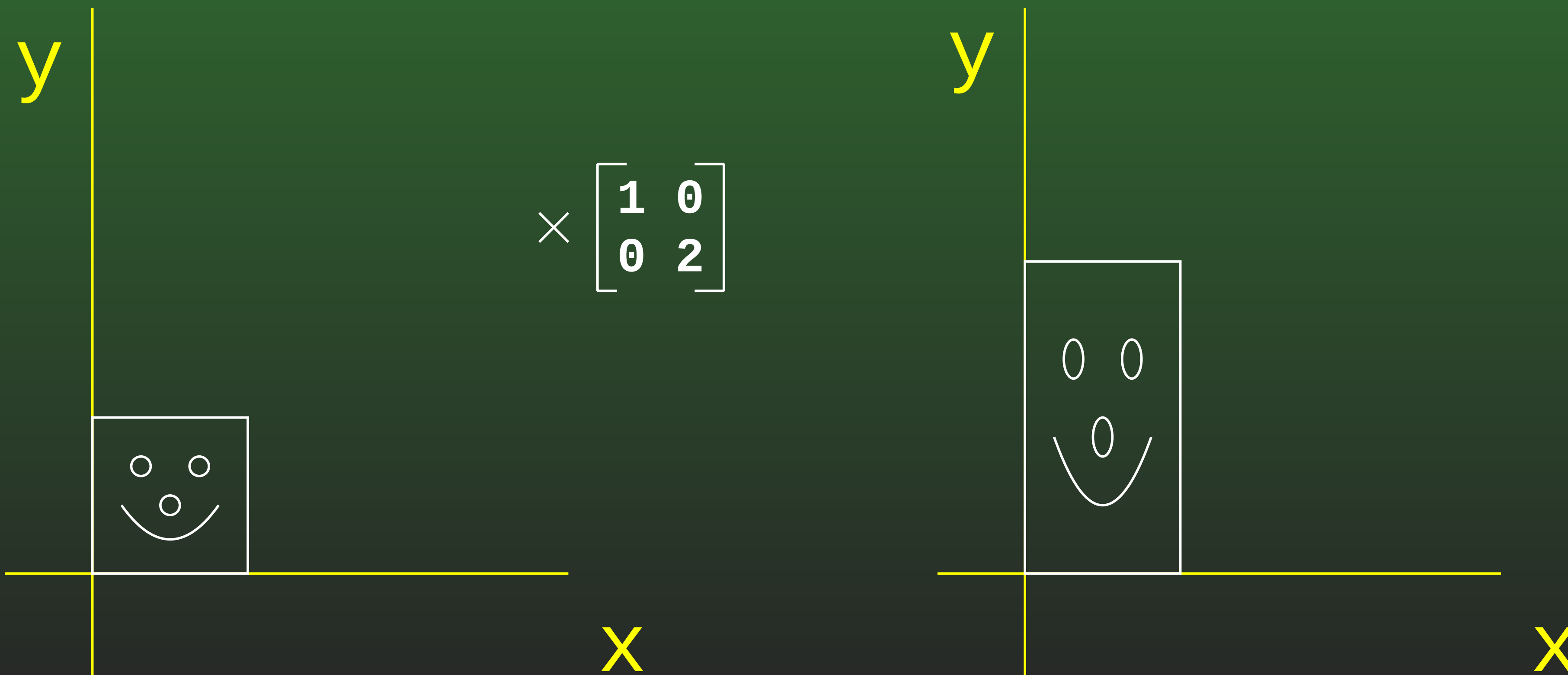
05-54: Scaling

- **Uniform Scaling** occurs when we scale an object uniformly in all directions
- Uniform scaling preserves angles, but not areas or volumes



Scaling

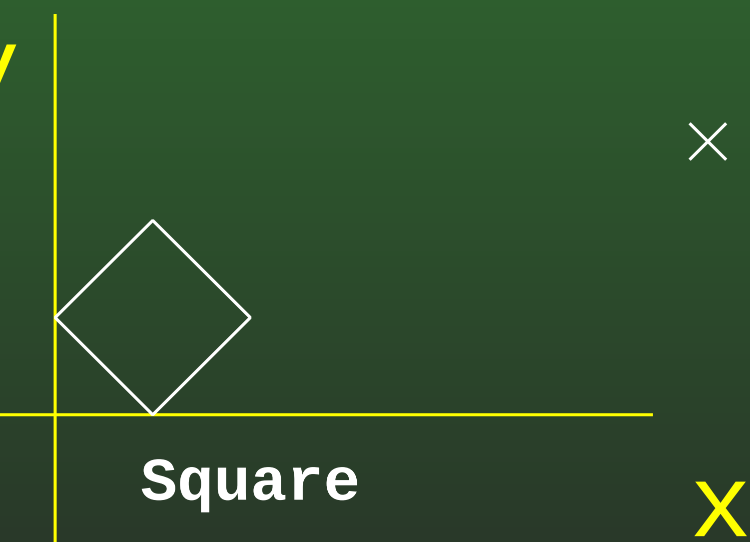
- **Non-Uniform Scaling** occurs when we scale an object by different amounts in different dimensions
- Non-uniform scaling does not preserve angles, areas, or volumes



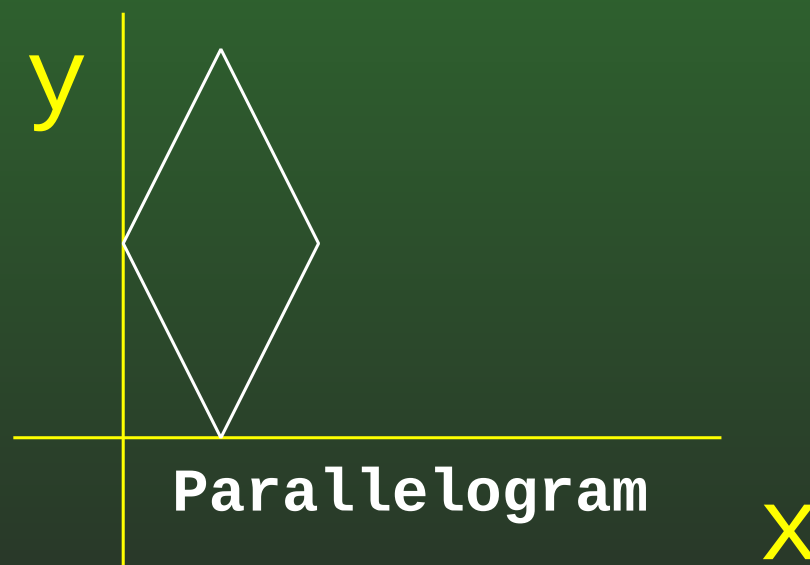
$$\times \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

05-56: Scaling

- **Non-Uniform Scaling** occurs when we scale an object by different amounts in different dimensions
- Non-uniform scaling does not preserve angles, areas, or volumes



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



05-57: Scaling in 3D

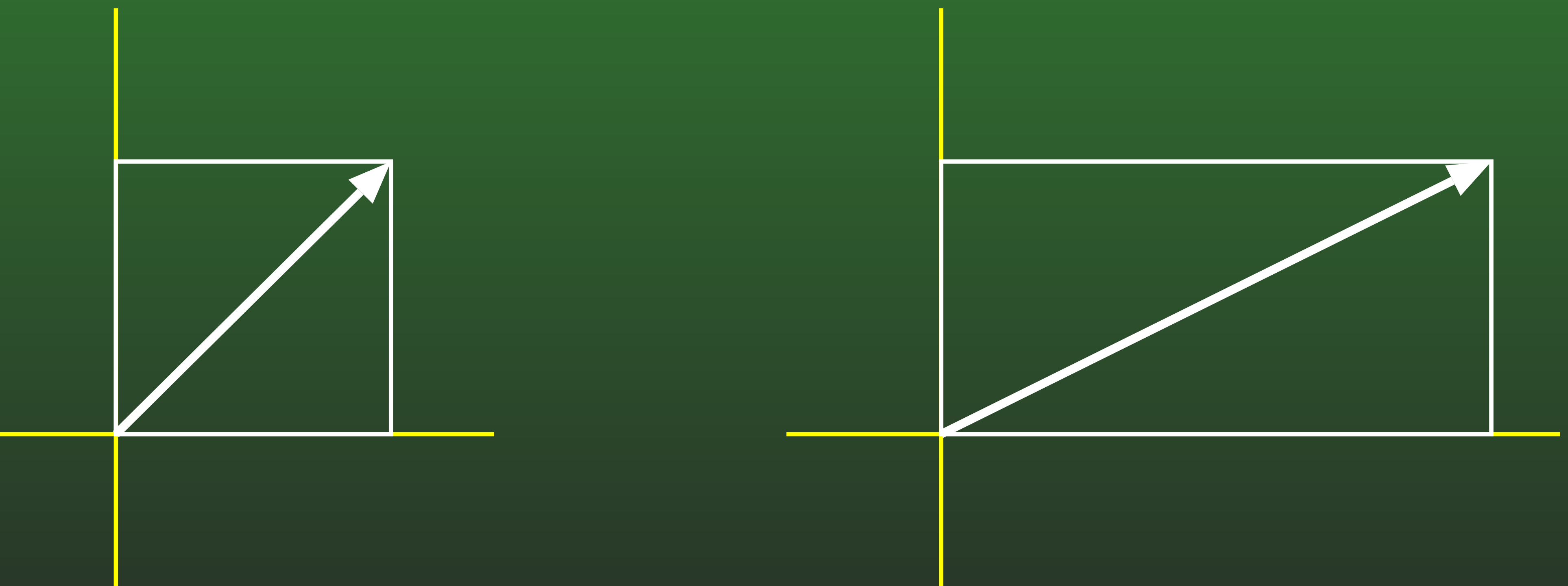
- The transformation matrix for scaling (both uniform and non-uniform) is straightforward:

$$\mathbf{S}(k_x, k_y, k_z) = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix}$$

- s_x, s_y, and s_z are the scaling factors for x, y and z
- if $s_x = s_y = s_z$, then we have uniform scaling

05-58: Scaling Along a Vector

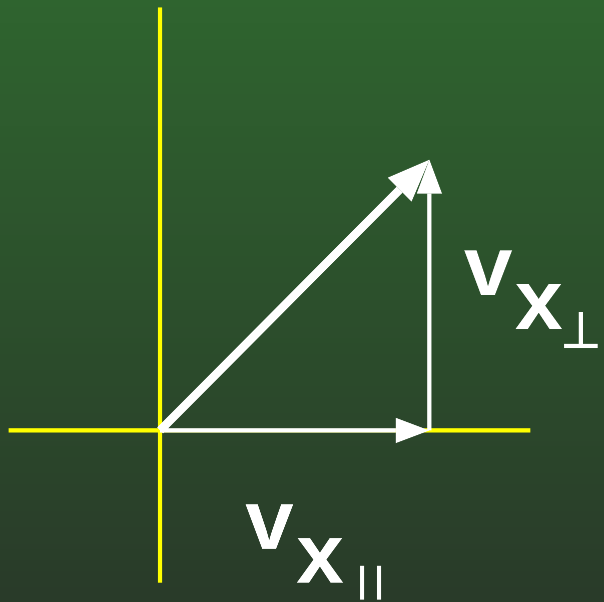
Scale by 2 along x axis

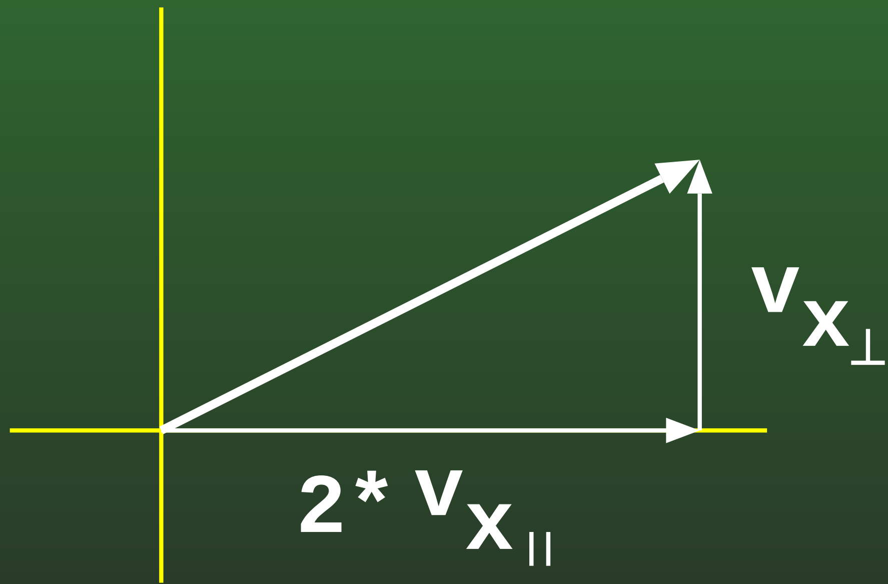


05-59: Scaling Along a Vector

Scale by 2 along x axis

Before Scale: $v = v_{x_{||}} + v_{x_{\perp}}$



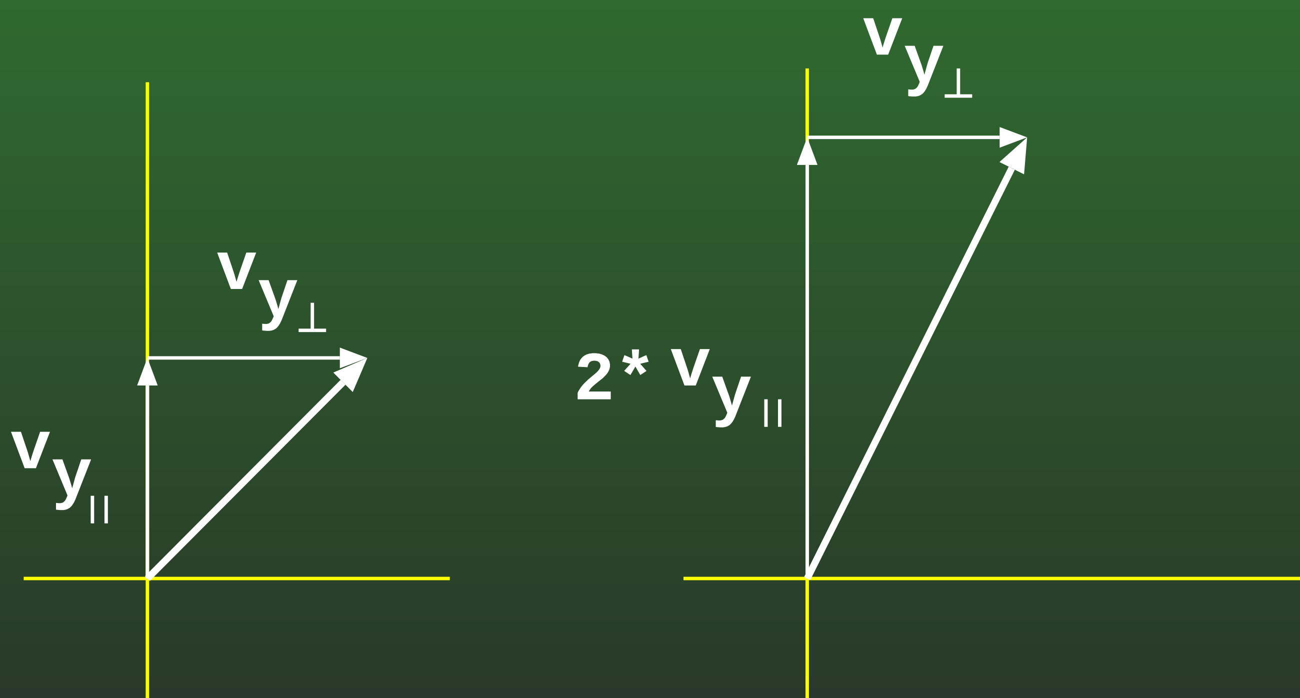


After Scale: $v = 2 * v_{x_{||}} + v_{x_{\perp}}$

05-60: Scaling Along a Vector

Scale by 2 along y axis

Before Scale: $v = v_{y_{\parallel}} + v_{y_{\perp}}$



After Scale: $v = 2 * v_{y_{\parallel}} + v_{y_{\perp}}$

05-61: Scaling Along a Vector

- To scale a vector along an axis:
  - Divide the vector into a component parallel to the axis, and perpendicular to the axis
  - Scale the component parallel to the axis
  - Leave the component perpendicular to the axis alone

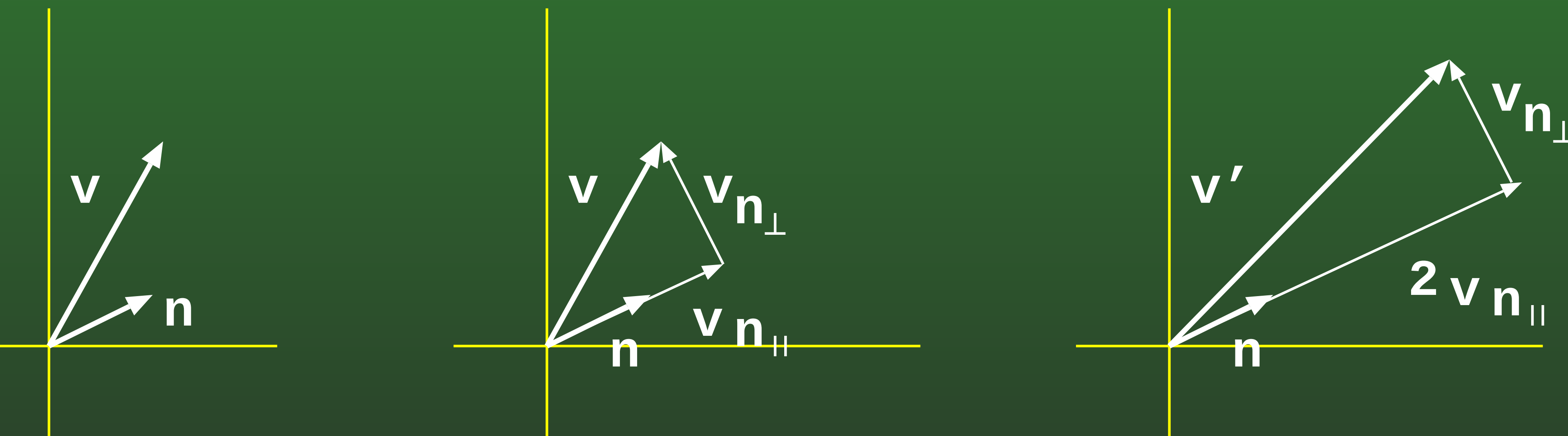
05-62: Scaling Along a Vector

- We can use the same technique to scale a vector $\mathbf{v}$ along an arbitrary vector $\mathbf{n}$
  - Divide $\mathbf{v}$ into a component parallel to $\mathbf{n}$, and a component perpendicular to $\mathbf{n}$
  - Scale the component parallel $\mathbf{n}$
  - Leave the component perpendicular to $\mathbf{n}$ alone

05-63: Scaling Along a Vector

Scale v by 2 along n

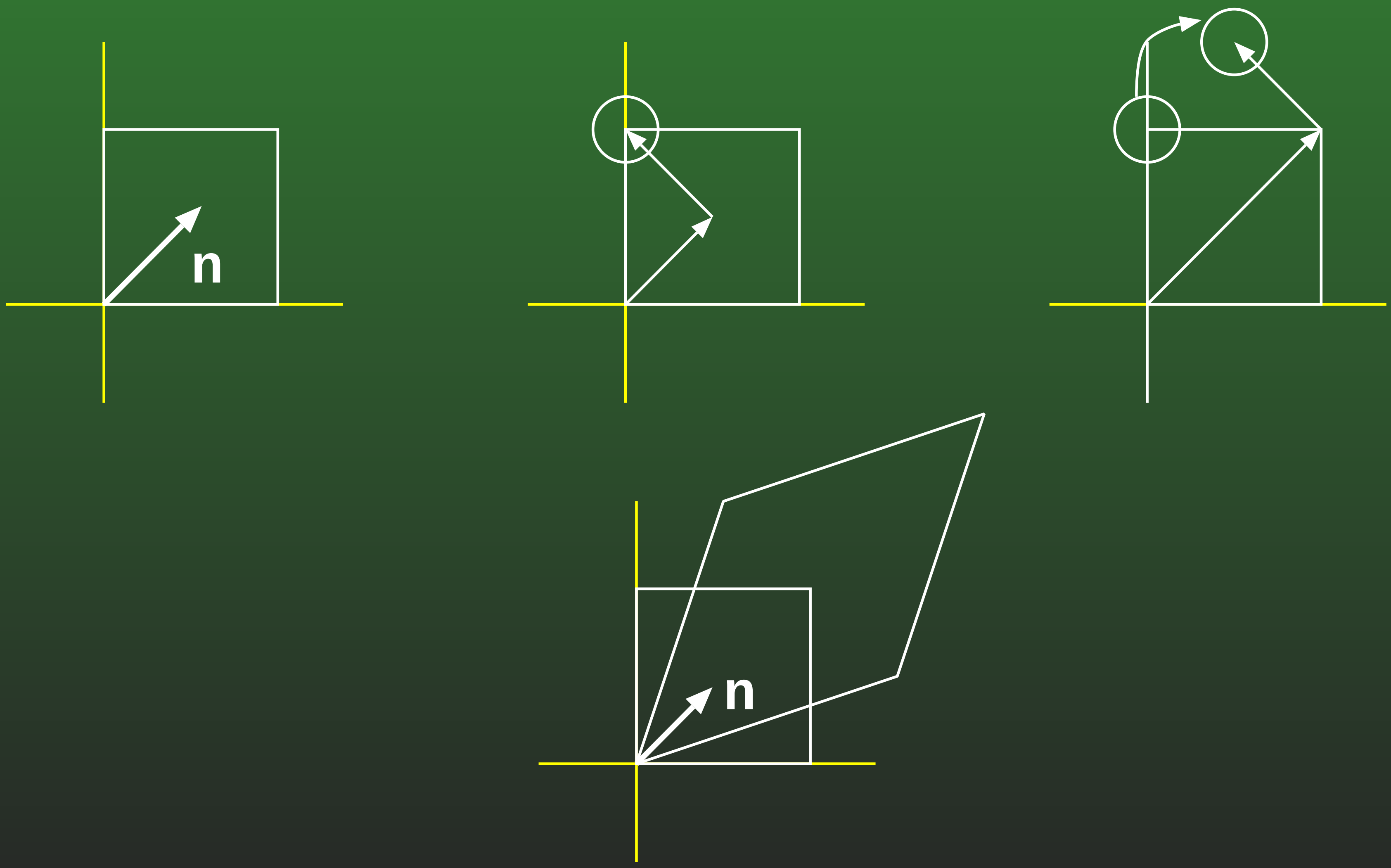
Decompse v into: $v = v_{n_{\parallel}} + v_{n_{\perp}}$



After Scale: $v' = 2 * v_{y_{\parallel}} + v_{y_{\perp}}$

05-64: Scaling Along a Vector

Scale box by 2 along n



05-65: Scaling Along a Vector

- Scaling a vector $\mathbf{v}$ by k along unit vector $\mathbf{n}$
  - Break $\mathbf{v}$ into $\mathbf{v}_{\parallel}$ and $\mathbf{v}_{\perp}$
  - $\mathbf{v} = \mathbf{v}_{\parallel} + \mathbf{v}_{\perp}$
  - $\mathbf{v}' = k * \mathbf{v}_{\parallel} + \mathbf{v}_{\perp}$
  - $\mathbf{v}_{\parallel} = ?$, $\mathbf{v}_{\perp} = ?$

Scaling Along a Vector

- Scaling a vector $\mathbf{v}$ by k along unit vector $\mathbf{n}$
  - Break $\mathbf{v}$ into $\mathbf{v}_{\parallel}$ and $\mathbf{v}_{\perp}$
  - $\mathbf{v} = \mathbf{v}_{\parallel} + \mathbf{v}_{\perp}$
  - $\mathbf{v}' = k * \mathbf{v}_{\parallel} + \mathbf{v}_{\perp}$
  - $\mathbf{v}_{\parallel} = (\mathbf{v} \cdot \mathbf{n}) * \mathbf{n}$
  - $\mathbf{v}_{\perp} = \mathbf{v} - \mathbf{v}_{\parallel}$

05-67: Scaling Along a Vector

- $\mathbf{v}_{\parallel} = (\mathbf{v} \cdot \mathbf{n}) * \mathbf{n}$
- $\mathbf{v}_{\perp} = \mathbf{v} - \mathbf{v}_{\parallel}$
- $\mathbf{v}' = k * \mathbf{v}_{\parallel} + \mathbf{v}_{\perp}$
- $\mathbf{v}' = k * \mathbf{v}_{\parallel} + \mathbf{v} - \mathbf{v}_{\parallel}$
- $\mathbf{v}' = (k - 1) * \mathbf{v}_{\parallel} + \mathbf{v}$
- $\mathbf{v}' = (k - 1) * (\mathbf{v} \cdot \mathbf{n}) * \mathbf{n} + \mathbf{v}$

05-68: Scaling Along a Vector

- Now that we know how to scale a vector along a different vector, how do we create the transformaion matrix?

05-69: Scaling Along a Vector

- Now that we know how to scale a vector along a different vector, how do we create the transformaion matrix?
  - Transform each of the axes
  - Fill in rows (columns, when using column vectors) of matrix

05-70: Scaling Along a Vector

- $\mathbf{v}' = (k-1) * (\mathbf{v} \cdot \mathbf{n}) * \mathbf{n} + \mathbf{v}$
- x-axis:

$(k-1)([1,0,0] \cdot [n_x, n_y, n_z]) * [n_x, n_y, n_z] + [1,0,0] =$

$(k-1)(n_x) * [n_x, n_y, n_z] + [1,0,0] = [(k-1)n_x^2 + 1, (k-1)n_x n_y, (k-1)n_x n_z]$

05-71: Scaling Along a Vector

- $\mathbf{v}' = (k-1) * (\mathbf{v} \cdot \mathbf{n}) * \mathbf{n} + \mathbf{v}$
- y-axis:

$$(k-1)([0,1,0] \cdot [n_x, n_y, n_z]) * [n_x, n_y, n_z] + [0,1,0] =$$

$$(k-1)(n_y) * [n_x, n_y, n_z] + [0,1,0] = [(k-1)n_x n_y, (k-1)n_y^2 + 1, (k-1)n_x n_z]$$

05-72: Scaling Along a Vector

- $\mathbf{v}' = (k-1) * (\mathbf{v} \cdot \mathbf{n}) * \mathbf{n} + \mathbf{v}$
- z-axis:

$$(k-1)([0,0,1] \cdot [n_x, n_y, n_z]) * [n_x, n_y, n_z] + [0,0,1] = (k-1)(n_z) * [n_x, n_y, n_z] + [0,0,1] =$$

$$[(k-1)n_x n_z, (k-1)n_y n_z, (k-1)n_z^2 + 1]$$

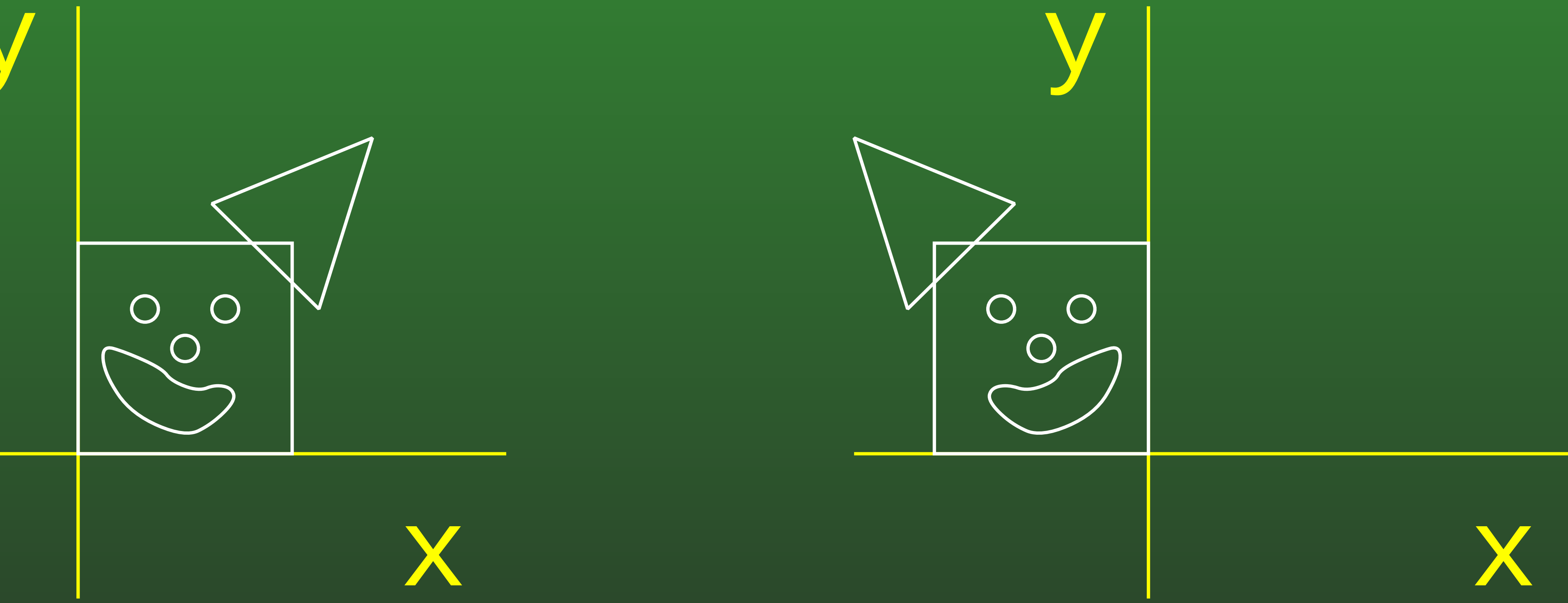
05-73: Scaling Along a Vector

$$\mathbf{S}(\mathbf{n}, k) = \begin{bmatrix} (k-1)n_x^2 + 1 & (k-1)n_x n_y & (k-1)n_x n_z \\ (k-1)n_x n_y & (k-1)n_y^2 + 1 & (k-1)n_x n_z \\ (k-1)n_x n_z & (k-1)n_y n_z & (k-1)n_z^2 + 1 \end{bmatrix}$$

05-74: Reflections 2D

- Another transformation that we can do with matrices is reflections
- Carndinal axes are easy to reflect around

 # Reflections 2D



05-76: Reflections 2D

- Another transformation that we can do with matrices is reflections
- Carndinal axes are easy to reflect around
  - How does the y basis vector change when reflecting around the y axis?
  - How does the x basis vector change when reflecting around the y axis?

05-77: Reflections 2D

- Another transformation that we can do with matrices is reflections
- Carndinal axes are easy to reflect around
  - How does the y basis vector change when reflecting around the y axis?
    - It doesn't!
  - How does the x basis vector change when reflecting around the y axis?
    - Multiplied by -1

Reflections 2D

- Reflecting around the y axis is the same as scaling the x axis by -1

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

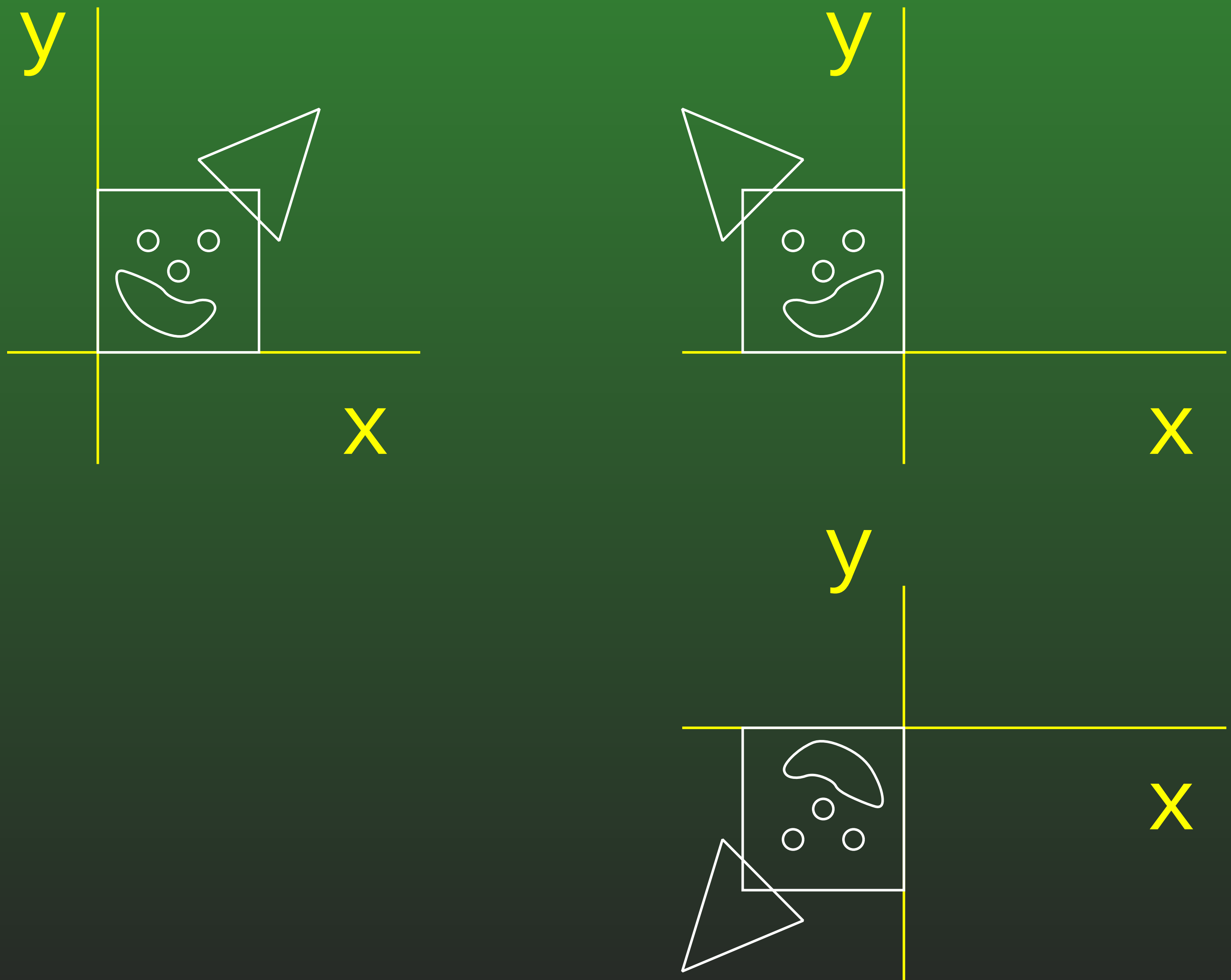
05-79: Reflections 2D

- To reflect along the x axis, we scale y by -1

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

- What happens when we reflect around the y axis, and then reflect around the y axis?
- Is this equivalent to doing some other operation?

05-80: Reflections 2D



05-81: Reflections 2D

- Let's say that we took a vector, then reflected it around the y axis, and then reflected it around the x axis:

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

05-82: Reflections 2D

- Let's say that we took a vector, then reflected it around the y axis, and then reflected it around the x axis
- Matrix Multiplication is associative

$$\begin{bmatrix} x & y \end{bmatrix} \left(\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \right)$$

05-83: Reflections 2D

- Let's say that we took a vector, then reflected it around the y axis, and then reflected it around the x axis
- Matrix Multiplication is associative

$$\begin{bmatrix} x & y \end{bmatrix} \left(\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \right)$$

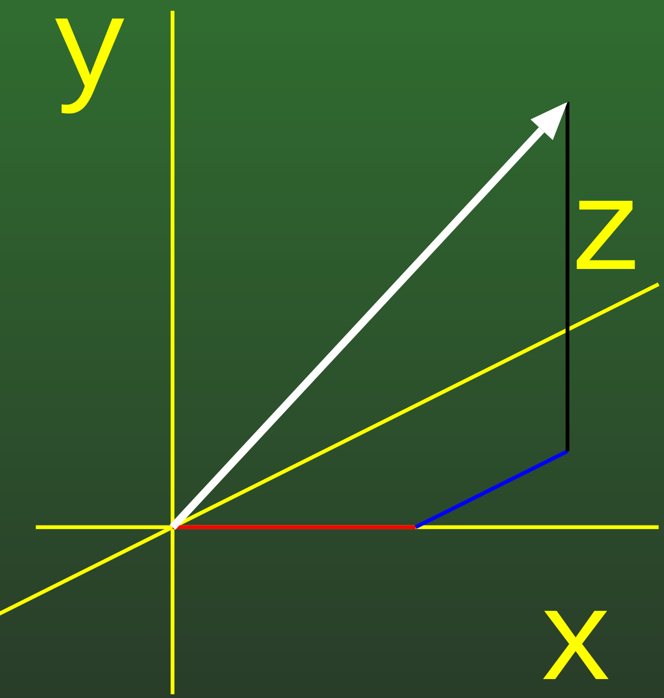
05-84: Reflections 2D

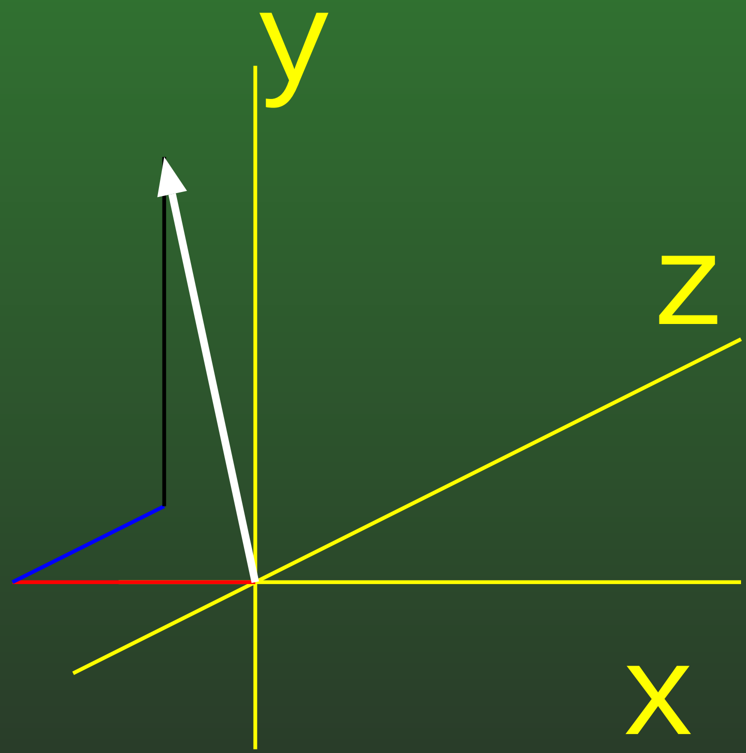
- Let's say that we took a vector, then reflected it around the y axis, and then reflected it around the x axis
- Equivalent to 180 degree (π radians) rotation

$$\begin{bmatrix} x & y \end{bmatrix} \left(\begin{bmatrix} \cos \pi & \sin \pi \\ -\sin \pi & \cos \pi \end{bmatrix} \right)$$

05-85: Reflections 3D

- What about reflecting around the yz-plane?





Reflections 3D

- To reflect around the yz plane, scale x by -1
- To reflect around the xy plane, scale z by -1
- To reflect around the xz plane, scale y by -1

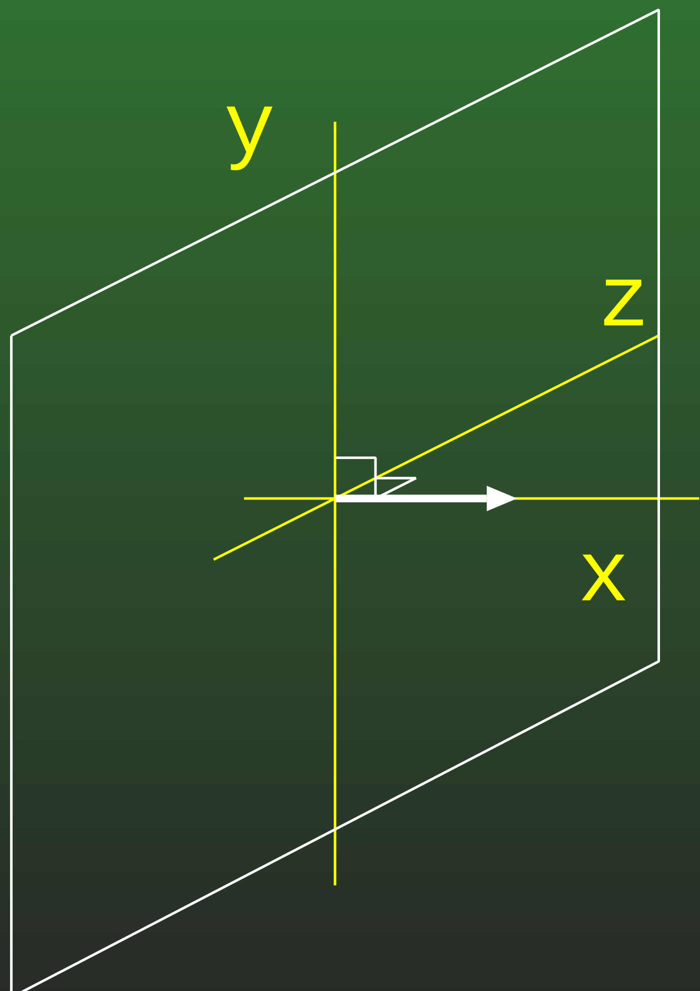
05-87: Reflections 3D

- To reflect around any plane
  - Find the normal of the plane (there are 2 – doesn’t matter which one)
  - Scale around this normal, with magnitude of -1

05-88: Reflections 3D

Reflect vector around yz-plane

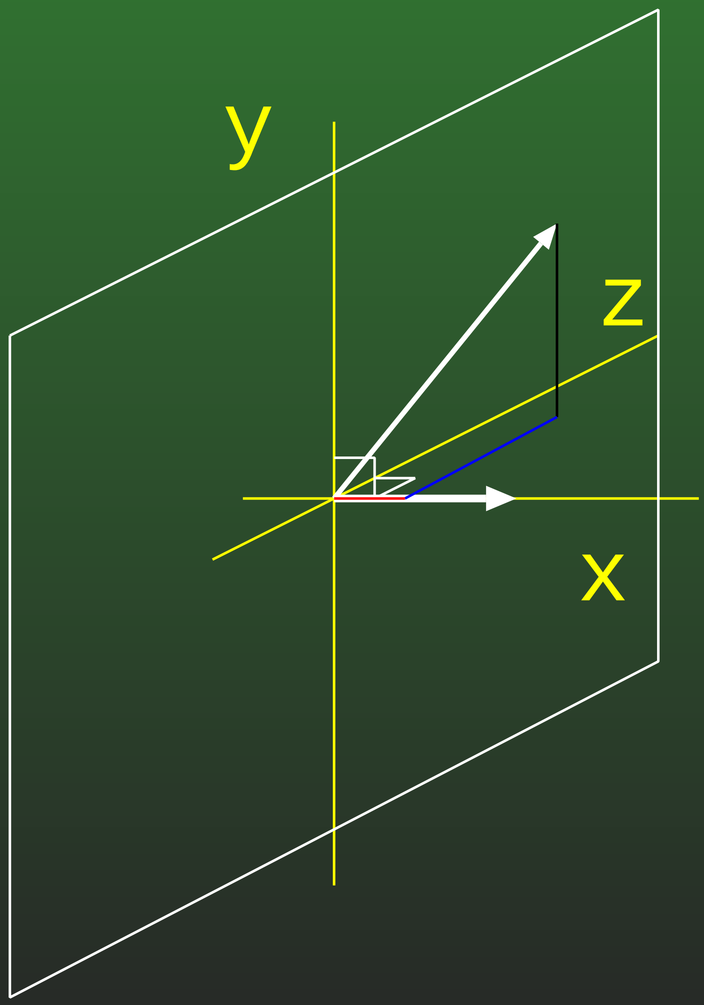
Scale by -1 along normal to plane

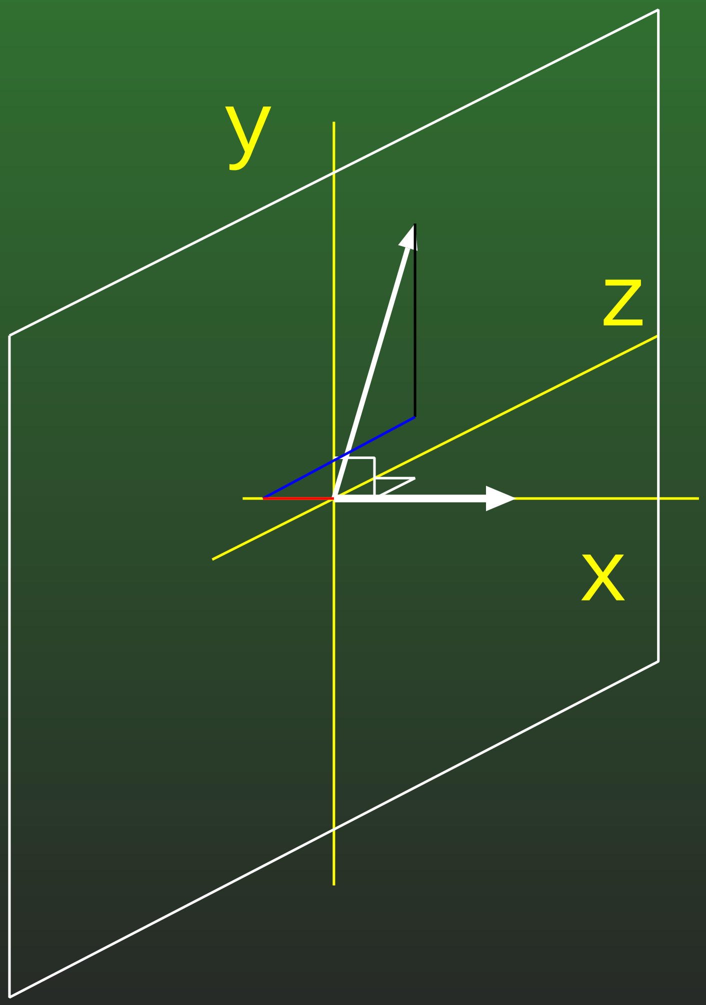


05-89: Reflections 3D

Reflect vector around yz-plane

Scale by -1 along normal to plane

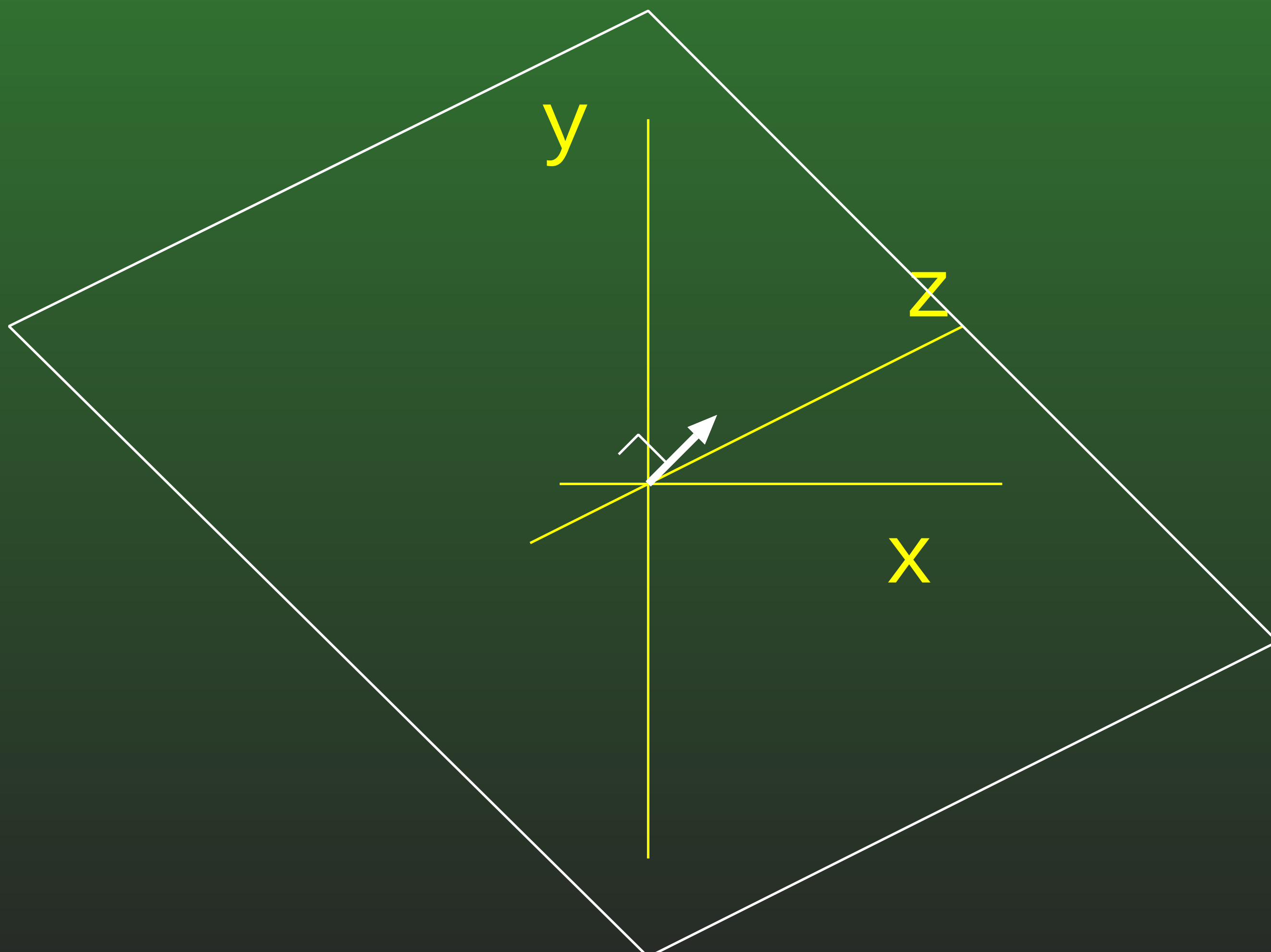




05-90: Reflections 3D

Reflect vector around any plane

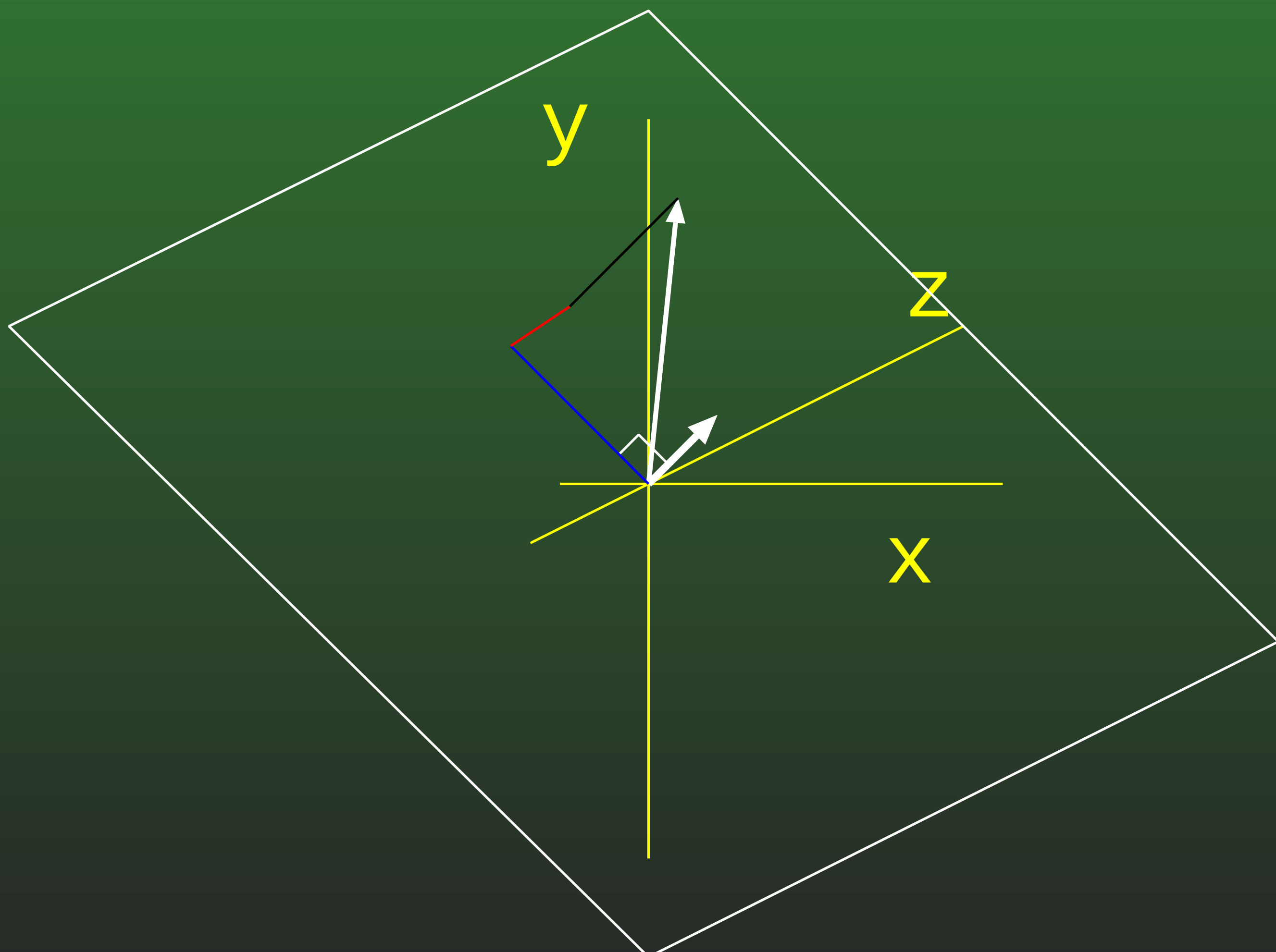
Scale by -1 along normal to plane



05-91: Reflections 3D

Reflect vector around any plane

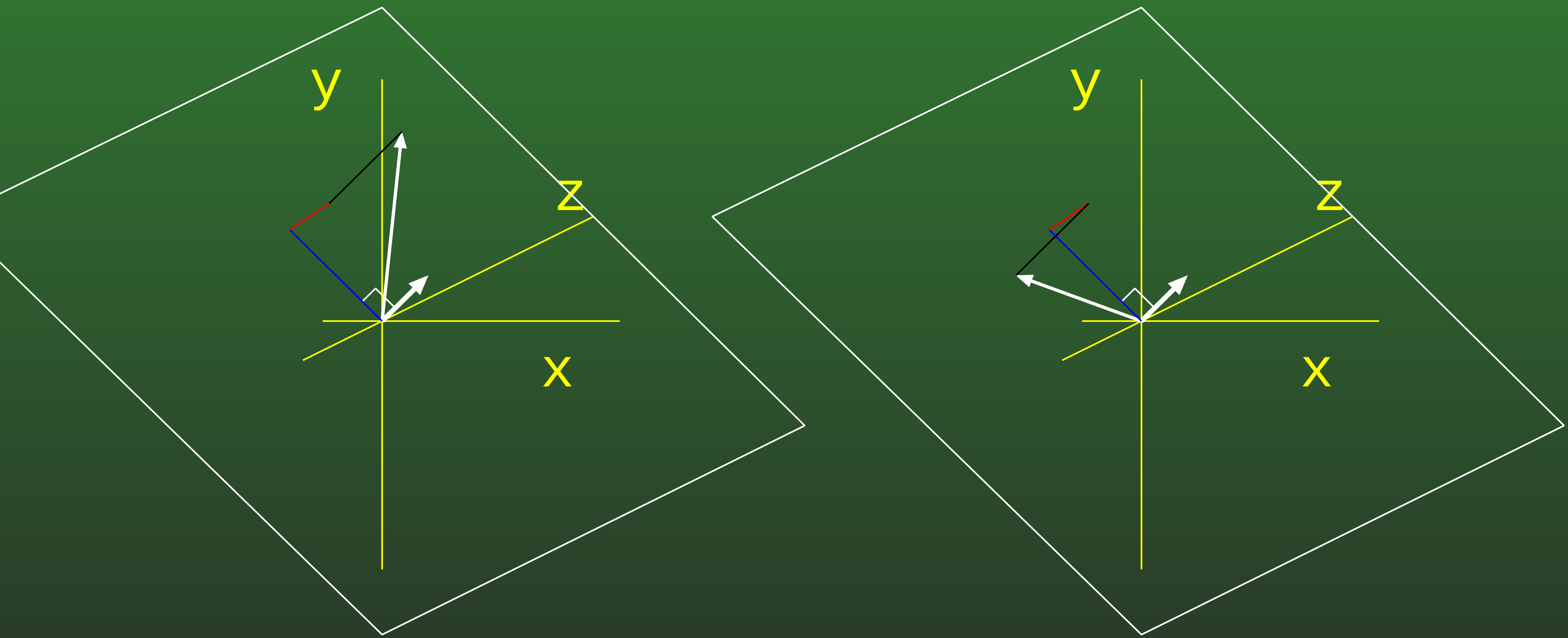
Scale by -1 along normal to plane



05-92: Reflections 3D

Reflect vector around any plane

Scale by -1 along normal to plane



05-93: Reflections 3D

- To reflect around any plane
  - Find the normal of the plane (there are 2 – doesn't matter which one)
  - Scale along this normal, with magnitude of -1
- If only we had some way of scaling along the normal
- ... can we scale along an arbitrary vector?

05-94: Reflection in 3D

- To scale along an arbitrary vector $\mathbf{n}$ by a scaling factor of k:

$$\mathbf{S}(\mathbf{n}, k) = \begin{bmatrix} (k-1)n_x^2 + 1 & (k-1)n_x n_y & (k-1)n_x n_z \\ (k-1)n_x n_y & (k-1)n_y^2 + 1 & (k-1)n_x n_z \\ (k-1)n_x n_z & (k-1)n_y n_z & (k-1)n_z^2 + 1 \end{bmatrix}$$

- Just need to set $k = -1$

05-95: Reflection in 3D

- To reflect around the plane normal to vector $\mathbf{n}$:

$$\mathbf{R}(\mathbf{n}) = \mathbf{S}(\mathbf{n}, -1) = \begin{bmatrix} -2n_x^2 + 1 & (-2)n_x n_y & -2n_x n_z \\ -2n_x n_y & -2n_y^2 + 1 & -2n_x n_z \\ -2n_x n_z & -2n_y n_z & -2n_z^2 + 1 \end{bmatrix}$$

05-96: Reflections

- Any two reflections are equivalent to a single rotation
  - Doesn’t matter what axes (2D) or planes (3D) we’re reflecting around
  - Reflect around *any* plane, then reflect around *any other* plane, still just a rotation
- First reflection flips model “inside out”, second reflection flips model “right-side out”
- A reflection around any axis is equivalent to a reflection around a cardinal axis, followed by a rotation

05-97: Shearing

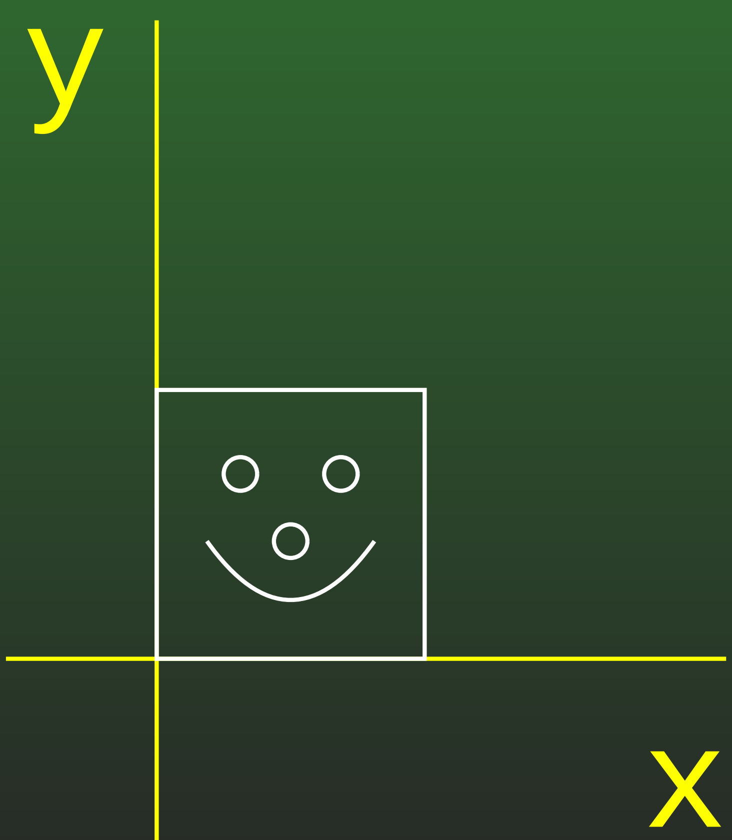
- A two-dimensional shear transform adds a multiple of x to y (while leaving x alone), or adds a multiple of y to x (while leaving y alone)
  - $[x, y] \Rightarrow [x + sy, y]$
  - $[x, y] \Rightarrow [x, y + sx]$
- Result is to “tilt” the object / image

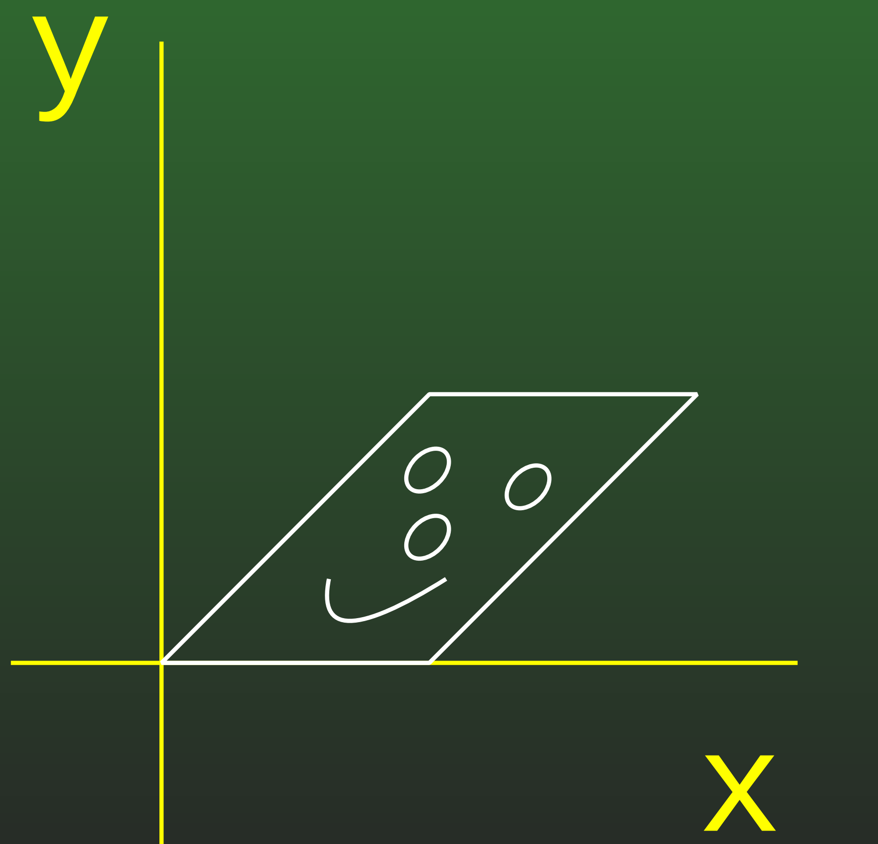
Shearing

Shearing along x in 2D

y' = y (unchagned)

x' = x + sy





05-99: Shearing

- Shearing along x axis by s:
  - $[x, y] \Rightarrow [x + sy, y]$
- What should the matrix be?

05-100: Shearing

- Shearing along x axis by s:
  - $[x, y] \Rightarrow [x + sy, y]$
- What should the matrix be?

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix}$$

05-101: Shearing

- Shearing along y axis by s:
  - $[x, y] \Rightarrow [x, y + sx]$

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}$$

05-102: Shearing

- We can extend shearing to 3 dimensions
  - Add a multiple of x to y, leaving x and y unchanged
  - Matrix?

05-103: Shearing

- We can extend shearing to 3 dimensions
  - Add a multiple of y to x, leaving y and z unchanged

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ s & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shearing

- We can extend shearing to 3 dimensions
  - Add a multiple s of z to x, and a multiple t of z to y, leaving z unchanged
  - Matrix?

Shearing

- We can extend shearing to 3 dimensions
  - Add a multiple s of z to x, and a multiple t of z to y, leaving z unchanged

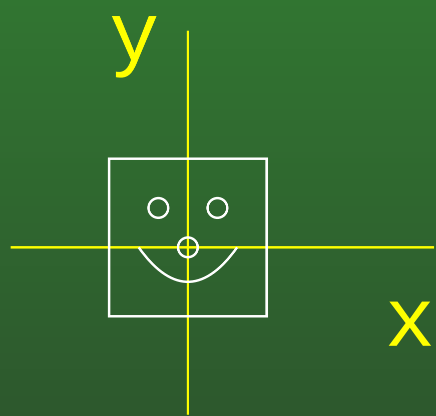
$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ s & t & 1 \end{bmatrix}$$

  - Other shears? (adding a multiple s of x to y, and a multiple t of x to z, for instance)

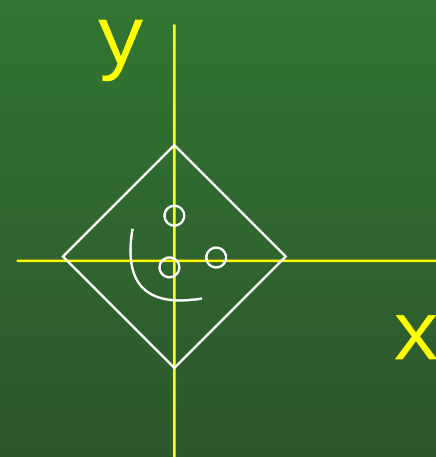
Shearing

- Shearing is equivalent to rotation and non-uniform scale
  - Technically, rotation and non-uniform scale gives a sheared shape
  - Need to rotate back to get the same orientation

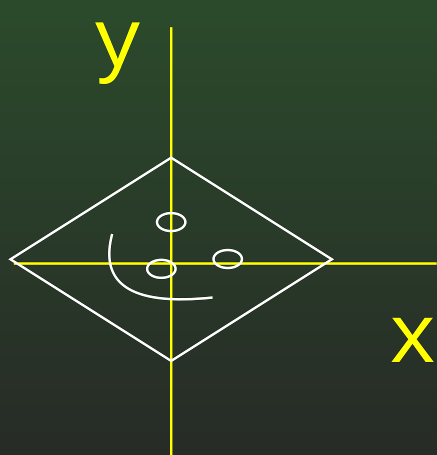
05-107: Shearing



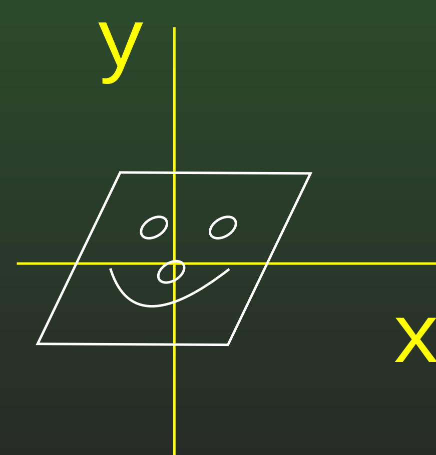
Rotate clockwise 45



Non-uniform scale (strech x, shrink y)



Rotate counter-clockwise (~32)



05-108: **Shearing**

- When shearing, angles are not preserved
- Areas (volumes) *are* preserved
- Parallel lines remain parallel

05-109: Combining Transforms

- A series of operations on a vector (model) is just a series of matrix multiplications
  - Rotate, scale, rotate (as above)
  - $((\mathbf{v}M_{rot})M_{scale})M_{rot}$
- Matrix multiplication is associative (but *not* communative!)

$$\begin{aligned} ((\mathbf{v}M_{rot})M_{scale})M_{rot} &= \mathbf{v}((M_{rot})(M_{scale}M_{rot})) \\ &= \mathbf{v}M' \end{aligned}$$

- We can create one matrix that does all transformations at once

05-110: Linear Transforms

- A transfomation is *Linear* if:
  - $\mathbf{F}(\mathbf{a} + \mathbf{b}) = \mathbf{F}(\mathbf{a}) + \mathbf{F}(\mathbf{b})$
  - $\mathbf{F}(k\mathbf{a}) = k\mathbf{F}(\mathbf{a})$
- That is:
  - Transforming two vectors and then adding them is the same as adding them, and then transforming
  - Scaling a vector and then transforming it is the same as transforming a vector, and then scaling it

05-111: Linear Transforms

- All transformations that can be represented by matrix multiplication are linear

$$\begin{aligned} \mathbf{F}(\mathbf{a}+\mathbf{b}) &= (\mathbf{a}+\mathbf{b})\mathbf{M} \\ &= \mathbf{aM}+\mathbf{bM} \\ &= \mathbf{F}(\mathbf{a})+\mathbf{F}(\mathbf{b}) \end{aligned}$$

$$\begin{aligned} \mathbf{F}(k\mathbf{a}) &= (k\mathbf{a})\mathbf{M} \\ &= k(\mathbf{aM}) \\ &= k\mathbf{F}(\mathbf{a}) \end{aligned}$$

05-112: Linear Transforms

- Rotation, scale (both uniform and non-uniform), reflection, and shearing are all linear transforms
- Is translation a linear transform?

05-113: Linear Transforms

- All linear transforms need to map the zero vector to the zero vector
  - Why?

05-114: Linear Transforms

- All linear transforms need to map the zero vector to the zero vector
  - Assume that $F(\mathbf{0}) = \mathbf{v}$
  - $F(k\mathbf{0}) = F(\mathbf{0}) = \mathbf{v}$
  - $\mathbf{F}(k\mathbf{a}) = k\mathbf{F}(\mathbf{a})$
  - Thus, $\mathbf{v} = k\mathbf{v}$ for all k, only true if $\mathbf{v}$ is the zero vector

05-115: Linear Transforms

- All linear transforms need to map the zero vector to the zero vector
- Translations do not map the zero vector to the zero vector
- Translations are not linear
  - Can't represent translations using matrix multiplication
  - (We will use matricies to represent translations later, but we will need to use highter dimensions ...)

05-116: Linear Transforms

- In a linear transformation, parallel lines remain parallel after translation
  - Angles may or may not be preserved
  - Areas / volumes may or may not be preserved

05-117: Affine Transforms

- An *Affine Transformation* is a linear transformation followed by a translation
- Any transform of form $\mathbf{F}(\mathbf{v}) = \mathbf{v}\mathbf{M} + \mathbf{b}$ is affine
- We will only concern ourselves with affine transforms in this class

05-118: Angle-Preserving Transforms

- A transform is angle preserving if angles are preserved.
- Which transformations are angle preserving?

05-119: Angle-Preserving Transforms

- A transform is angle preserving if angles are preserved.
- Which transformations are angle preserving?
  - Translations
  - Rotation
  - Uniform Scale
- Why not reflection?

05-120: Rigid Body Transforms

- Rigid body transforms change only:
  - Orientaton of an object
  - Position of an object
- Only translation and rotation are rigid-body transforms
- Reflection is not rigid body
- Also known as “proper” transformations