

Game Engineering

CS420/686-2016S-04

Intro to 3D Math

David Galles

Department of Computer Science

University of San Francisco

04-0: Right-Handed vs. Left-Handed

- Hold out your left hand (really, do it!):
 - Thumb to the right
 - Index finger up
 - Middle finger straight ahead
- This forms a basis for a 3D coordinate system

04-1: Right-Handed vs. Left-Handed

- Hold out your left hand (really, do it!):
 - Thumb to the right (+ x)
 - Index finger up (+ y)
 - Middle finger straight ahead (+ z)
- This forms a basis for a 3D coordinate system –
Left-Handed Coordinate system

04-2: Right-Handed vs. Left-Handed

- Now, Hold out your *right* hand (yes, really do it!):
 - Thumb to the left (+ x)
 - Index finger up (+ y)
 - Middle finger straight ahead (+ z)
- This forms the other basis for 3D coordinate system – Right-Handed Coordinate system

04-3: Right-Handed vs. Left-Handed

- Any basis can be rotated to be either left-handed or right-handed
- Swap between systems by flipping any one axis
- Flipping *two* axes leaves handedness unchanged (why?)
 - What about flipping all 3?

04-4: Right-Handed vs. Left-Handed

- Computer Graphics typically uses Left-Handed coordinate system
 - Book does, too
- “Pure” linear algebra often uses Right-Handed coordinate system
 - Ogre also uses a Right-Handed coordinate system
 - Easy transformation, just invert the sign of one axis

04-5: Multiple Coordinate Systems

- OK, so we've decided on a right-handed coordinate system (given Ogre), with y pointing "Up"
- Pick an arbitrary location for the origin
 - Often in the middle of the world
 - Can place it off in some corner
- Not quite done – can use multiple coordinate systems!

04-6: Multiple Coordinate Systems

- World Space
- Object Space
- Camera Space (Special case of Object Space)
- Intertial Space

04-7: World Space

- Assume that the origin of the world is the middle of the field between SI and K Hall
 - 2130 Fulton, the official University address is there
 - $+x$ is East (along Fulton), $+y$ is straight up, $+z$ is North
- What direction is “forward” from me in world space?
- What is the point 5 feet in front of me in world space?
 - What if I rotate 15 deg. to the left?

04-8: Object Space

- Define a new coordinate system
- Origin is at my center
- +x to my right
- +y is up through my head
- +z is straight ahead

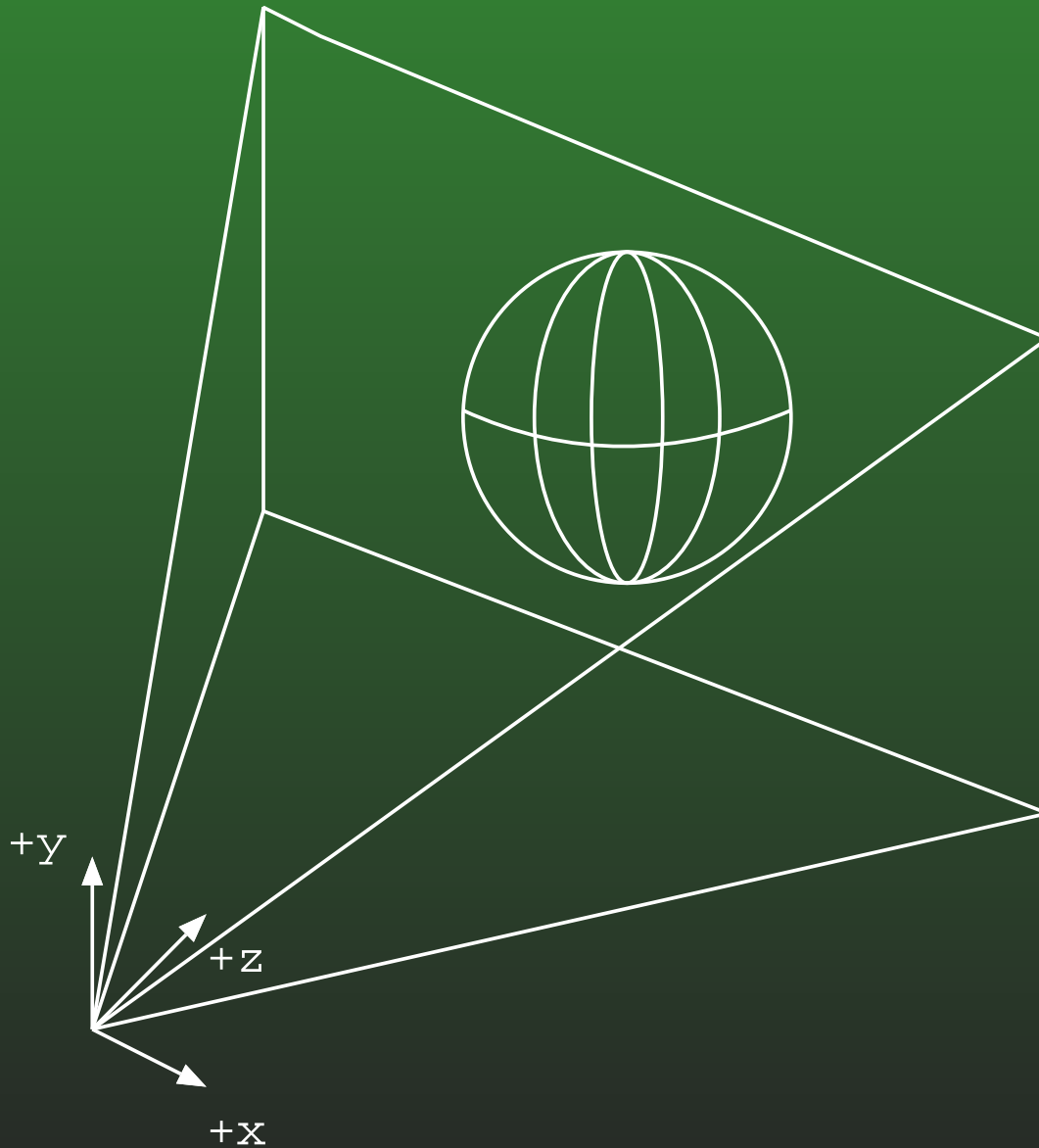
04-9: Object Space

- In my object space, finding a point right ahead of me is trivial
- Given a coordinate in my object space, determining where I have to look (to aim, for instance) is trivial
- Of course, we will need a way to translate between world space and object space
 - Say tuned!
- Define an “Object Space” for each object in our world

04-10: Camera Space

- Camera Space is a special case of object space
 - Object is the camera
- We'll use left-handed coordinates (+z into the screen), swapping to right-hand is easy (invert Z)
- Why is camera space useful?

04-11: Camera Space



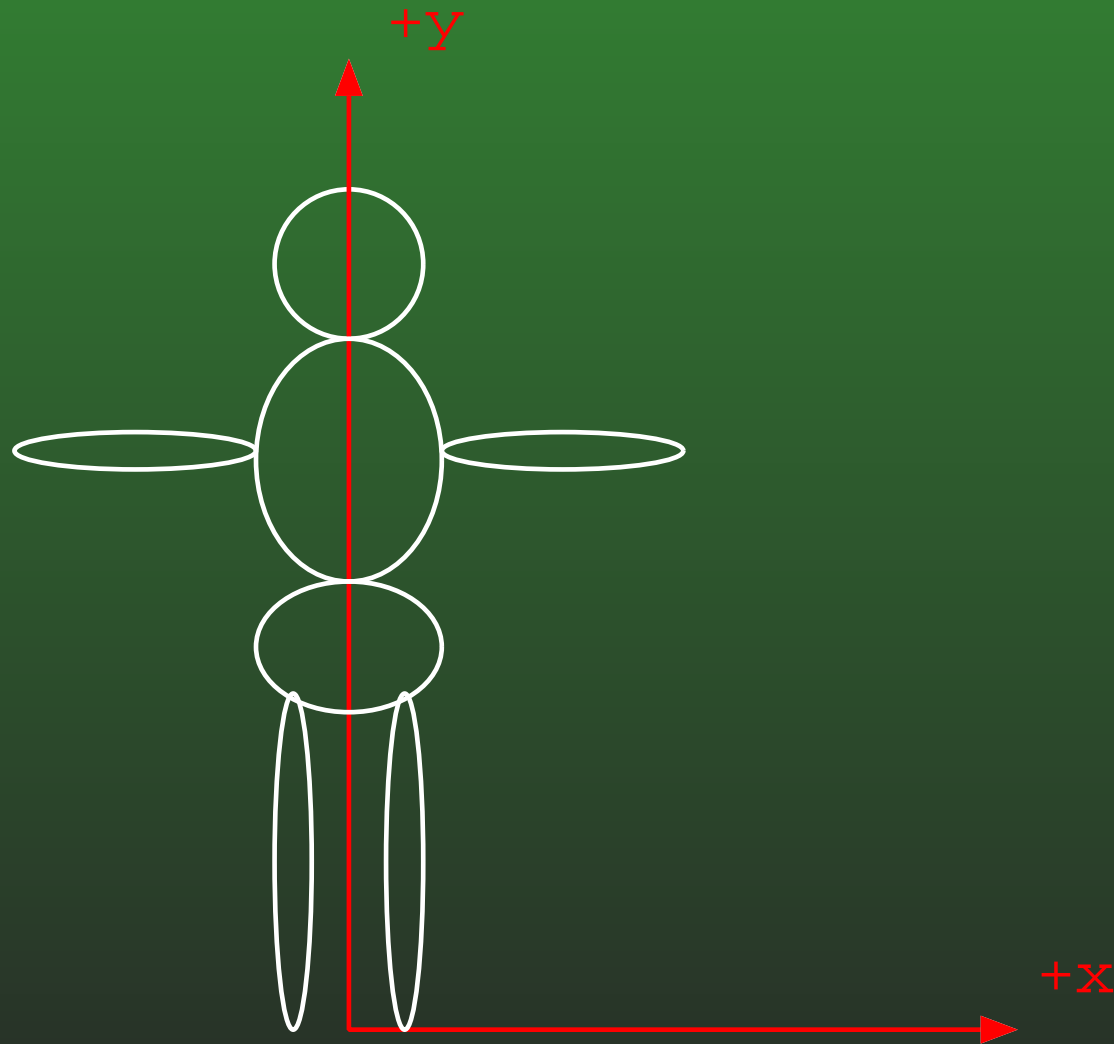
04-12: Camera Space

- Is an object within the camera's frustum?
- Is object A in front of object B, or vice-versa?
- Is an object close enough to the camera to render?
- ... etc

04-13: **Intertial Space**

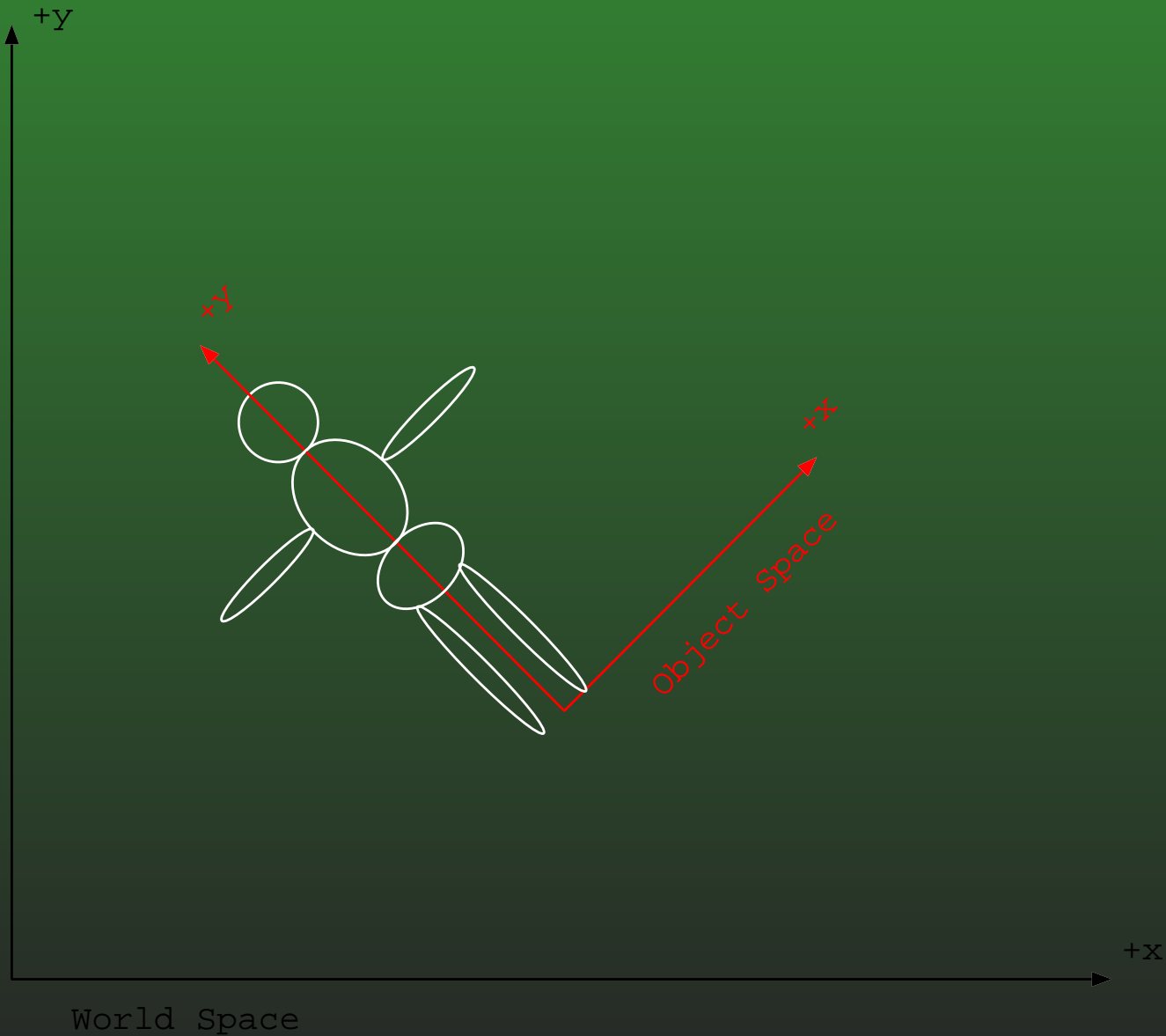
- Halfway between object space and world space
- Axes parallel to world space
- Origin same as object space

04-14: Inertial Space

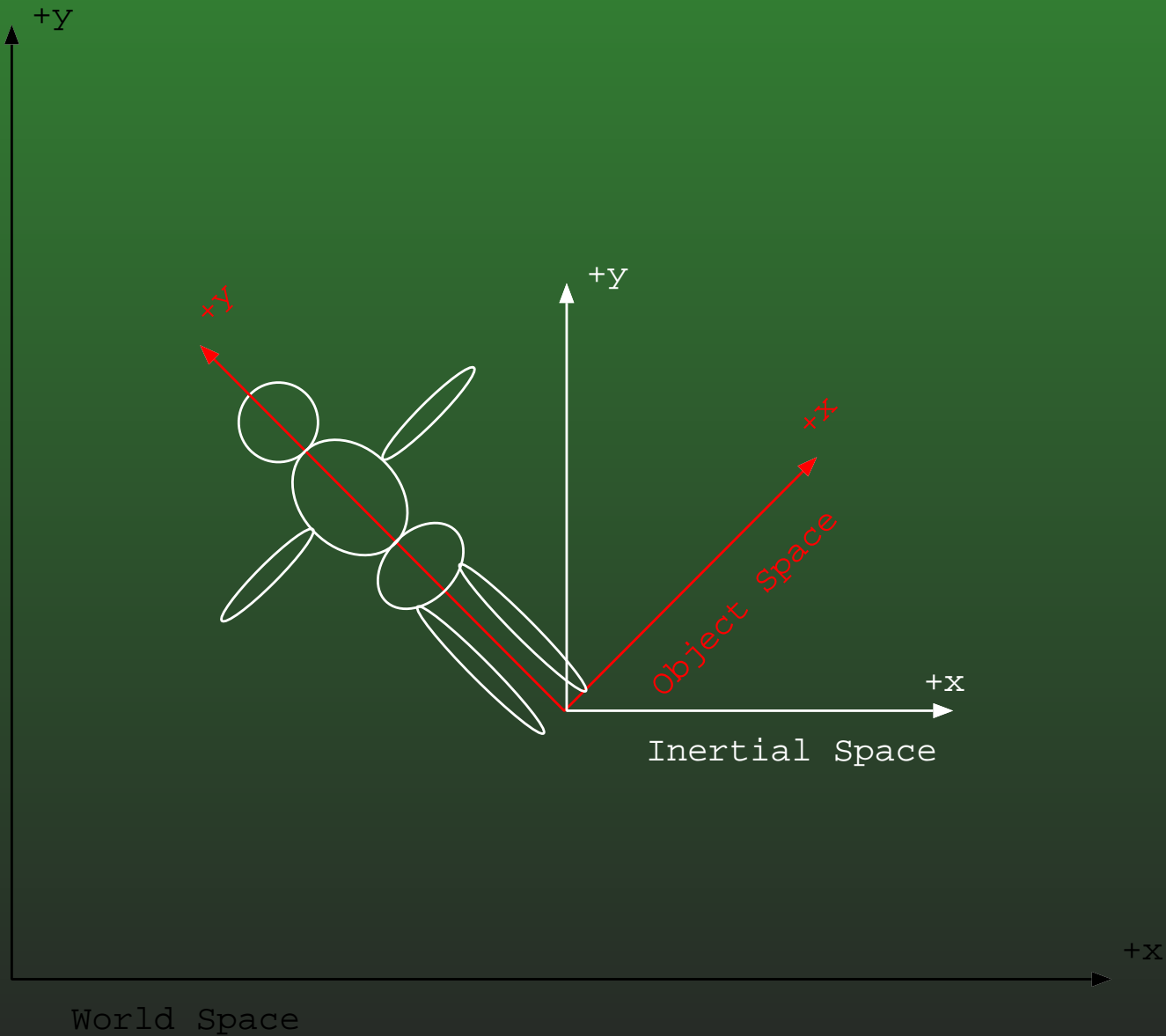


Object Space

04-15: Inertial Space



04-16: Inertial Space

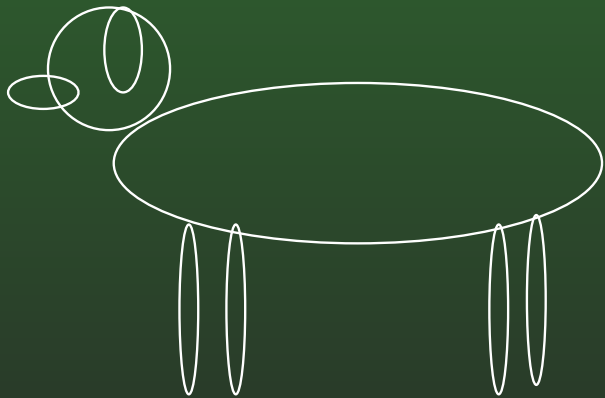


04-17: Nested Coordinate Spaces

- Each object needs to be oriented in world space
- That is, the axes for the local space of the object need to be oriented in world space.
- We could use a different object's local space instead of global space
 - Easiest to see with an example

04-18: Nested Coordinate Spaces

- Assume that we have a dog, which has a head and ears
 - The head can wag back and forth (in relation to the body)
 - The ears can flap up and down (in relation to the head)



- We don't want to describe the position of the ears in world space, or even in dog space
 - Head space is much more convenient

04-19: Nested Coordinate Spaces

- Dog's ears are described in head space
 - Up and down in relation to the head
- Dog's head is described in dog space
 - Back and forth in relation to the dog
- Dog's position is described in world space

04-20: Nested Coordinate Spaces

- To render the dog (with ears!)
 - Translate the ear location from head space to dog space
 - Translate the ear location (and the head location) from dog space to world space
 - Translate all the dog from world space to camera space
 - Project the objects from 3-space to a plane

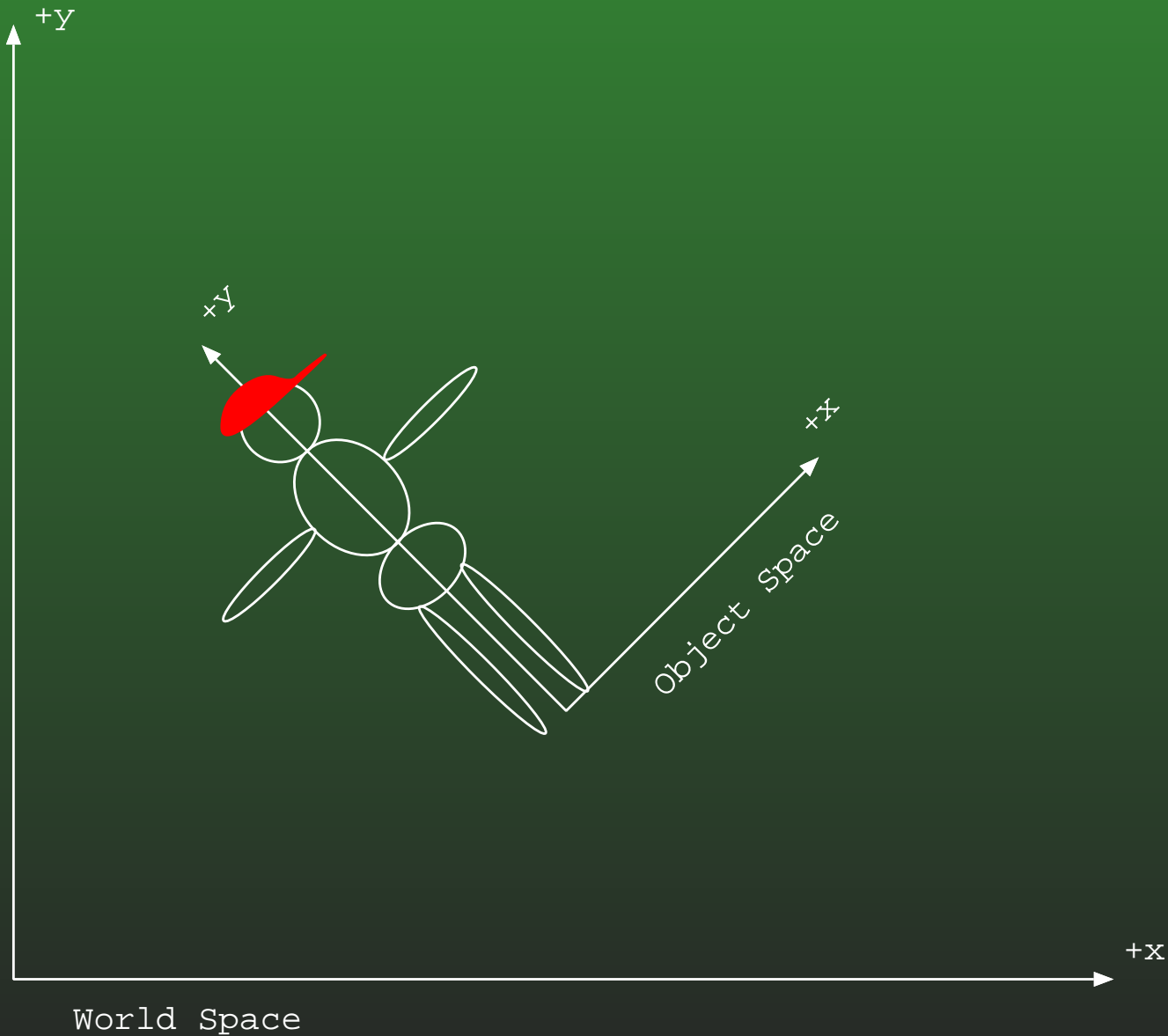
04-21: Nested Coordinate Spaces

- The Head space is a child of the Dog space
 - The Dog space is the parent of the Head space
- The Ear space is a child of the head space
 - The Head space is the parent of the ear space
- We could also dynamically parent and unparent objects

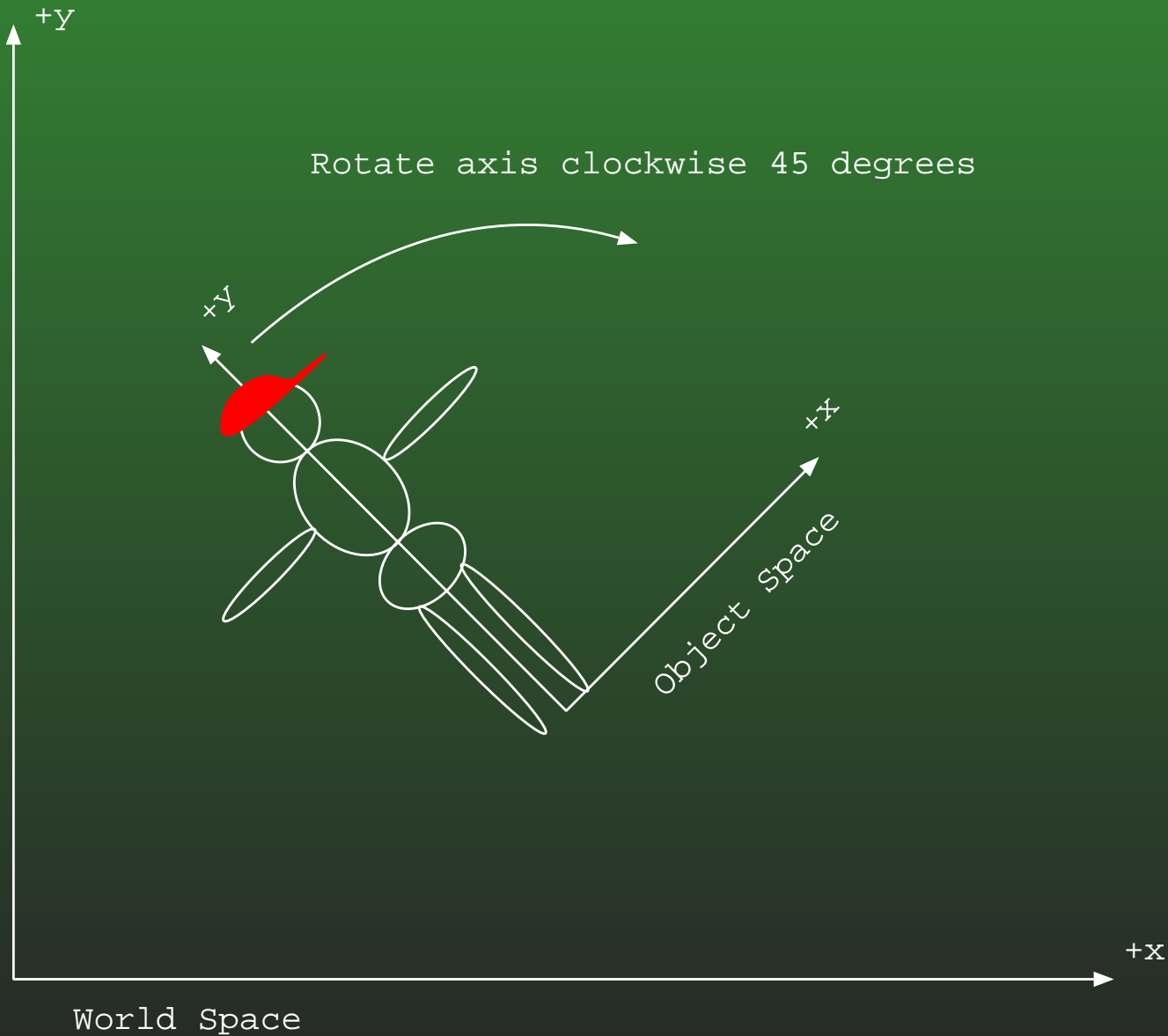
04-22: Changing Coordinate Spaces

- 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

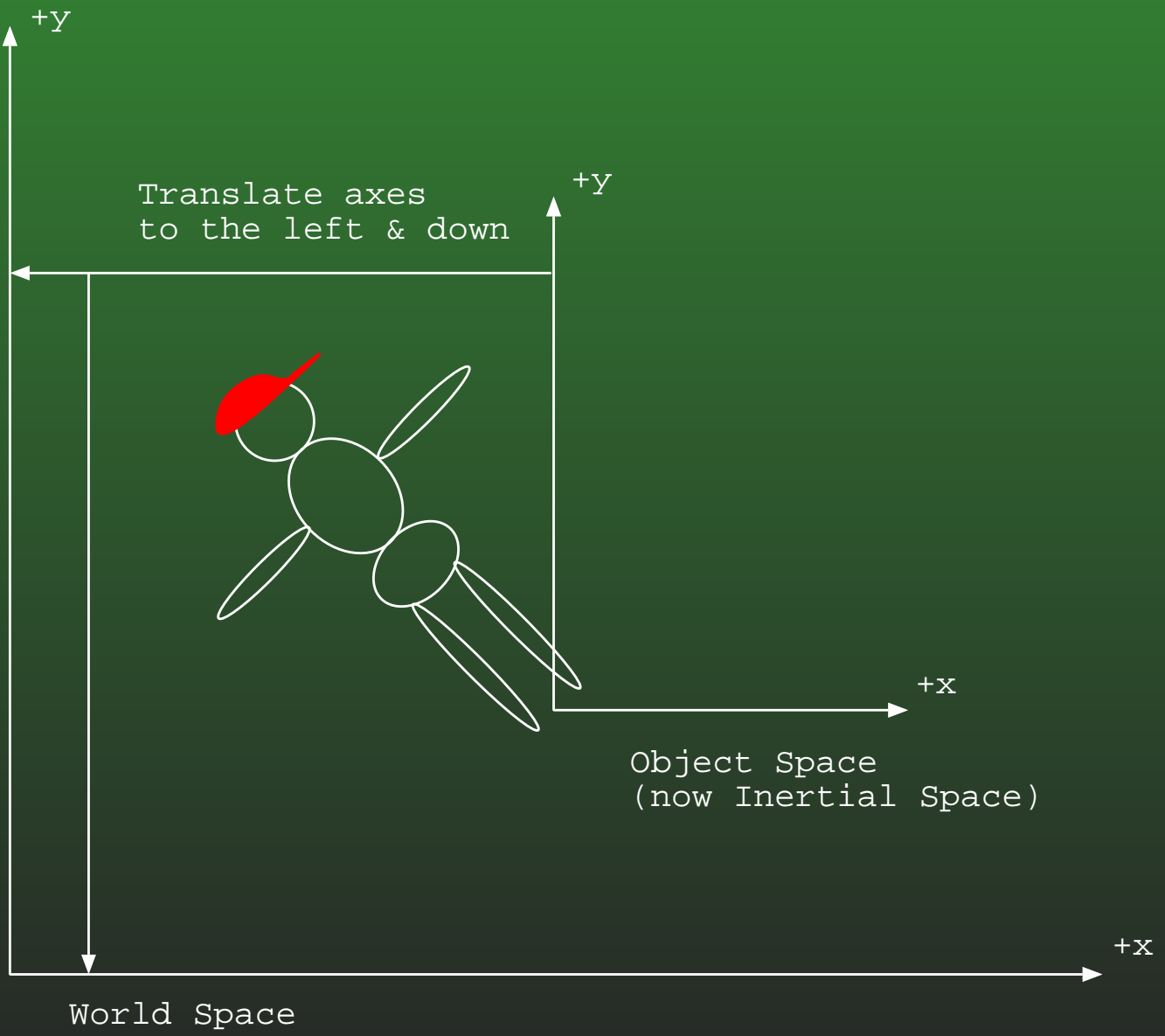
04-23: Changing Coordinate Spaces



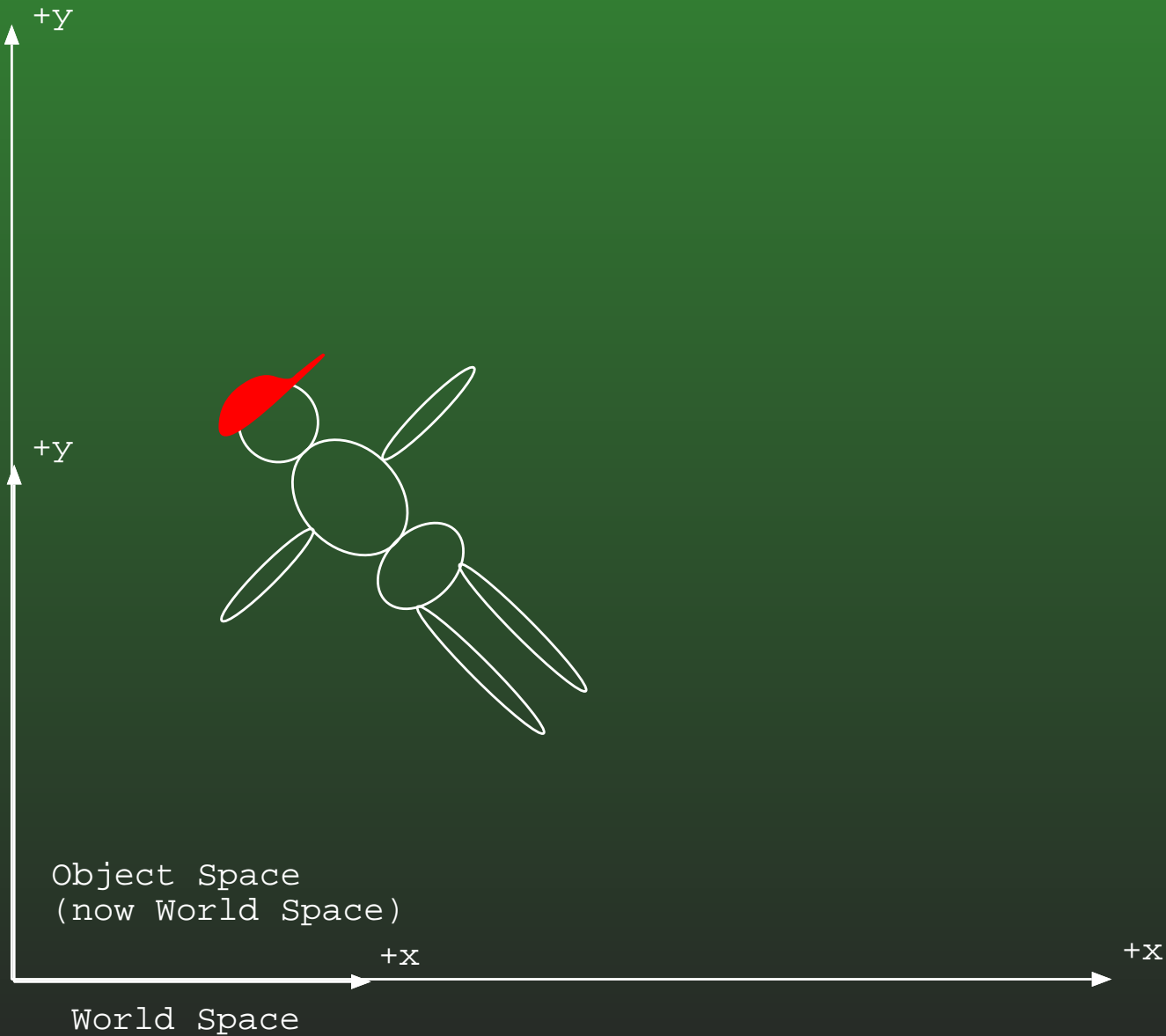
04-24: Changing Coordinate Spaces



04-25: Changing Coordinate Spaces



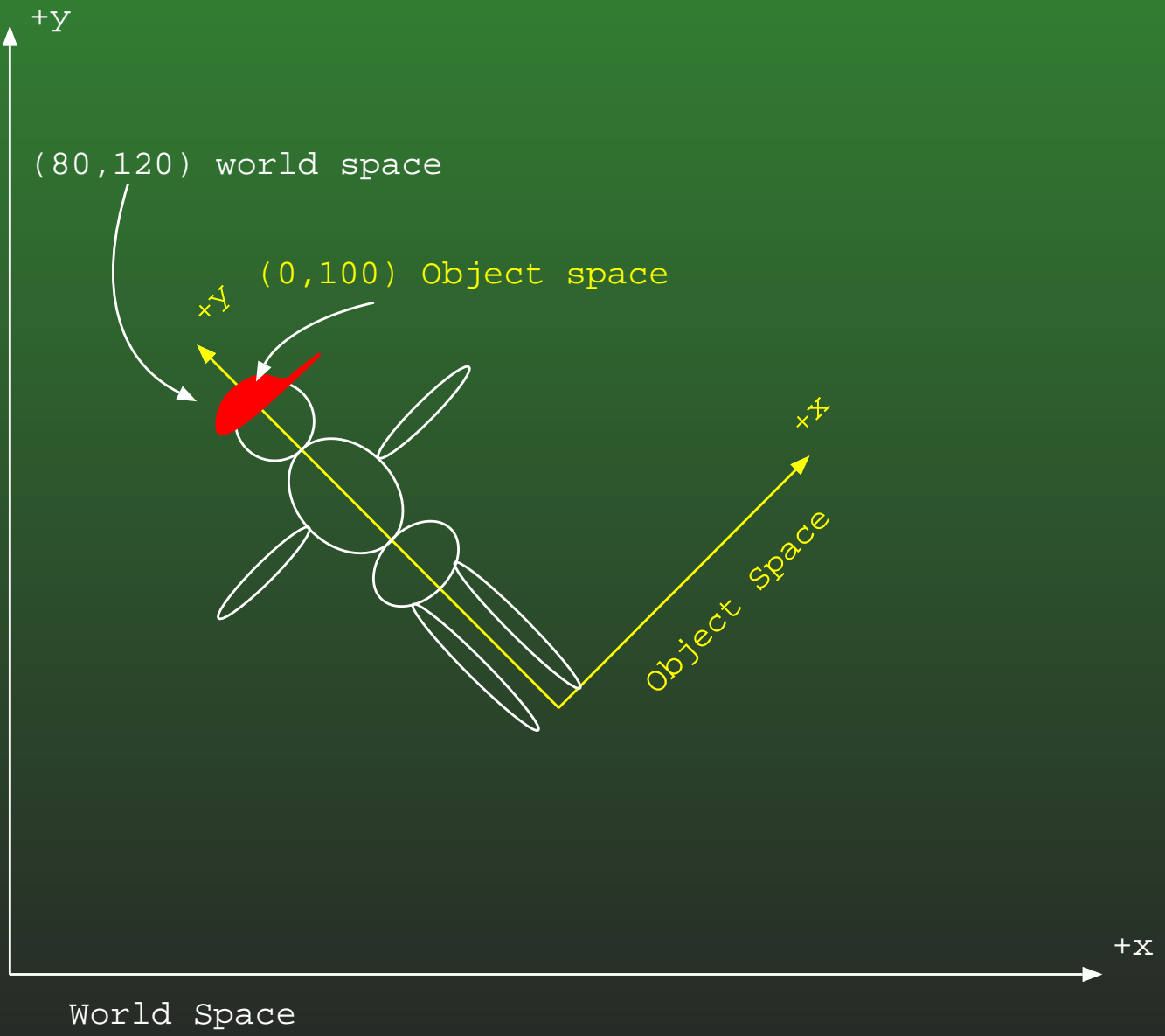
04-26: Changing Coordinate Spaces



04-27: Changing Coordinate Spaces

- Rotate axes to the right 45 degrees
 - Hat rotates the the left 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 $(80, 120)$
- We'll see how to do those rotations using matrices later ...

04-28: Changing Coordinate Spaces



04-29: Back to Basics

- A Vector is a displacement
- Vector has both *direction* and *length*
- Can also think of a vector as a position (just a displacement from the origin)
- Can be written as a row or column vector
 - Difference can be important for multiplication

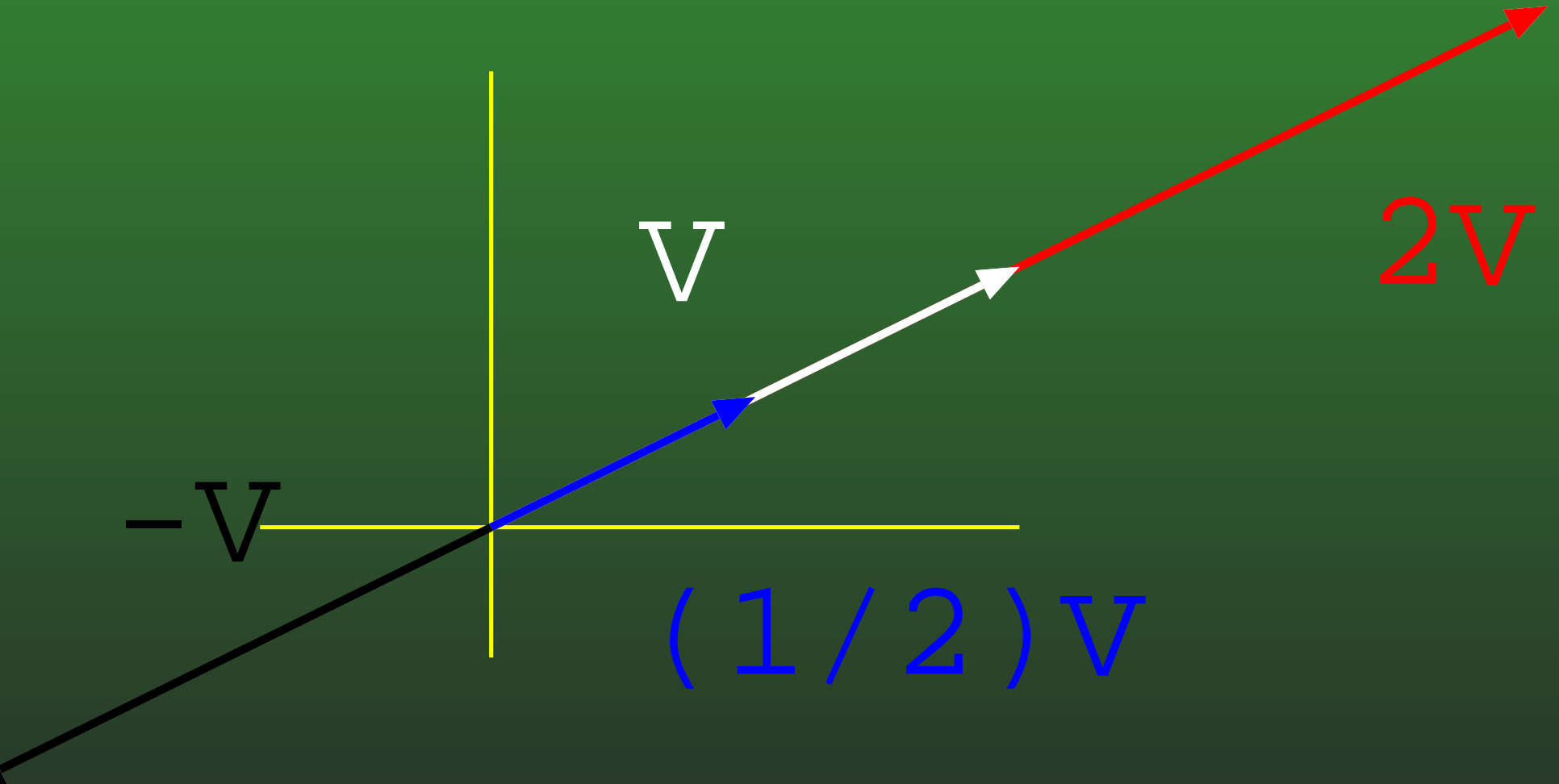
04-30: Vector Operations

- Multiplying by a scalar
 - To multiply a vector \mathbf{v} by a scalar s , multiply each component of the vector by s
 - Effect is scaling the vector – multiplying by 2 maintains the direction of the vector, but makes the length twice as long
 - Works the same for 2D and 3D vectors (and higher dimension vectors, too, for that matter)

04-31: Vector Operations

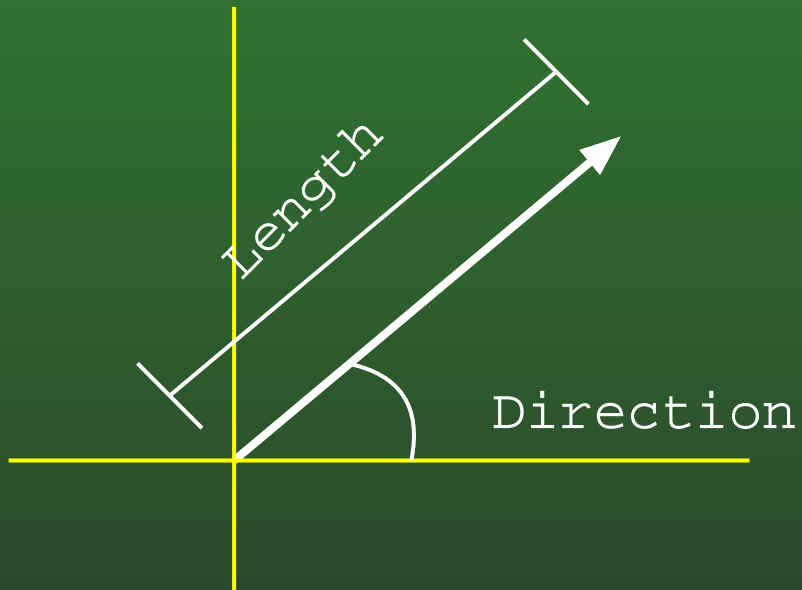
- Multiplying by a scalar
 - Multiplying a vector by -1 flips the direction of the vector
 - Works for 2D and 3D
 - Multiplying a vector by -2 both flips the direction, and scales the vector

04-32: Scaling a Vector



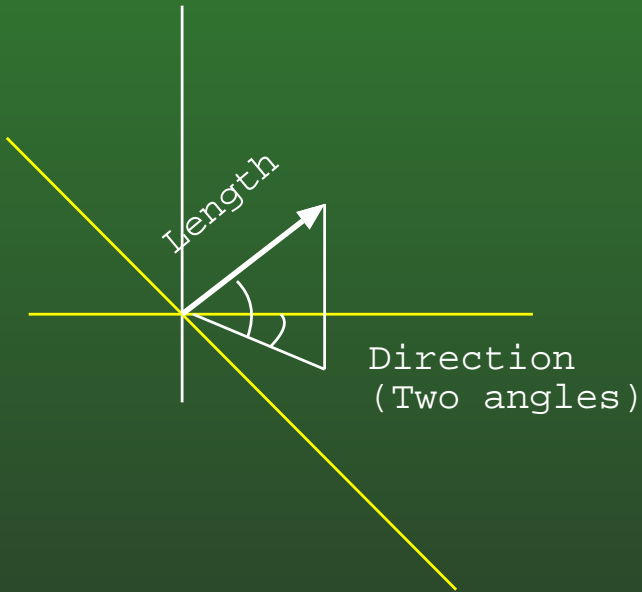
04-33: Length

- Vector has both direction and length



04-34: Length

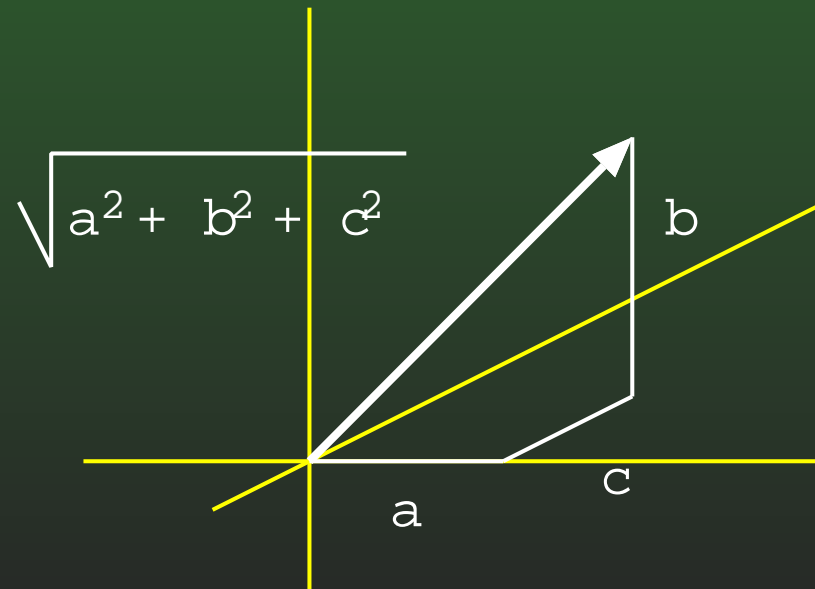
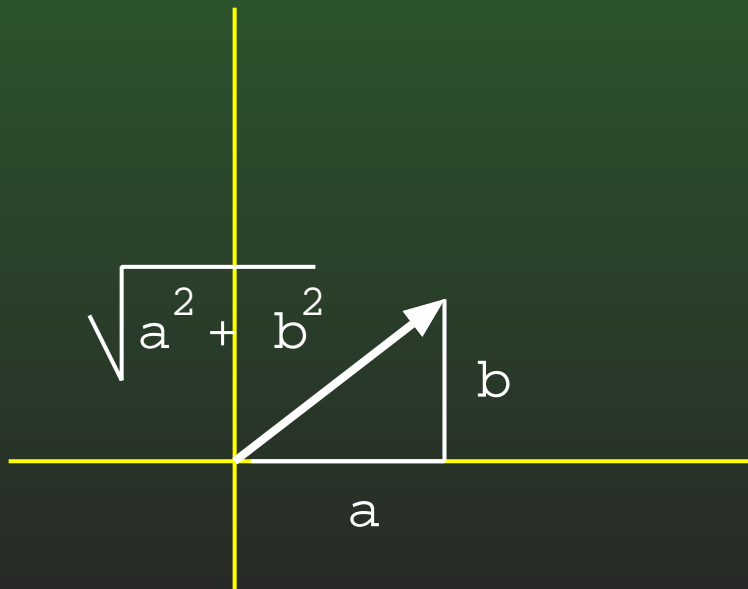
- Vector has both direction and length



04-35: Length

- Vector $\mathbf{v} = [v_1, v_2, \dots, v_n]$
- Length of \mathbf{v} :

$$\|\mathbf{v}\| = \sqrt{\sum_{i=1}^n v_i^2}$$

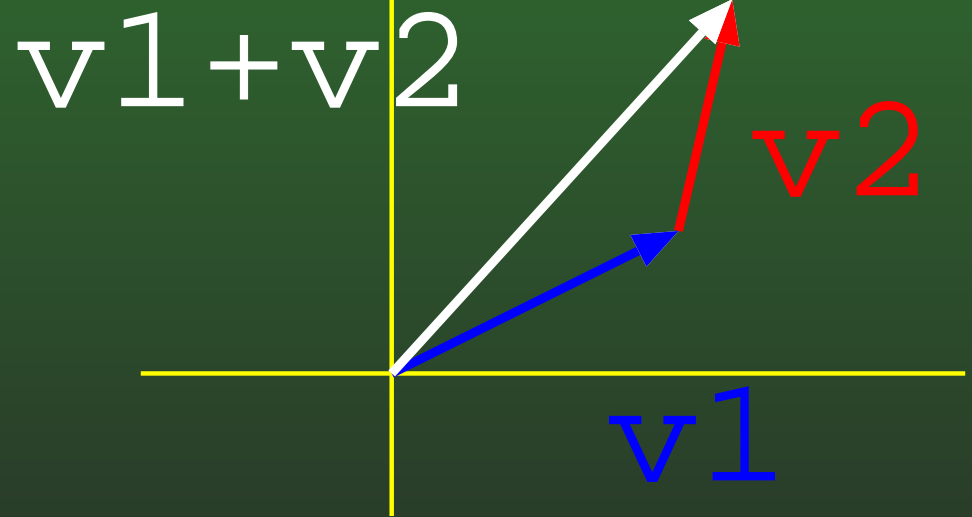
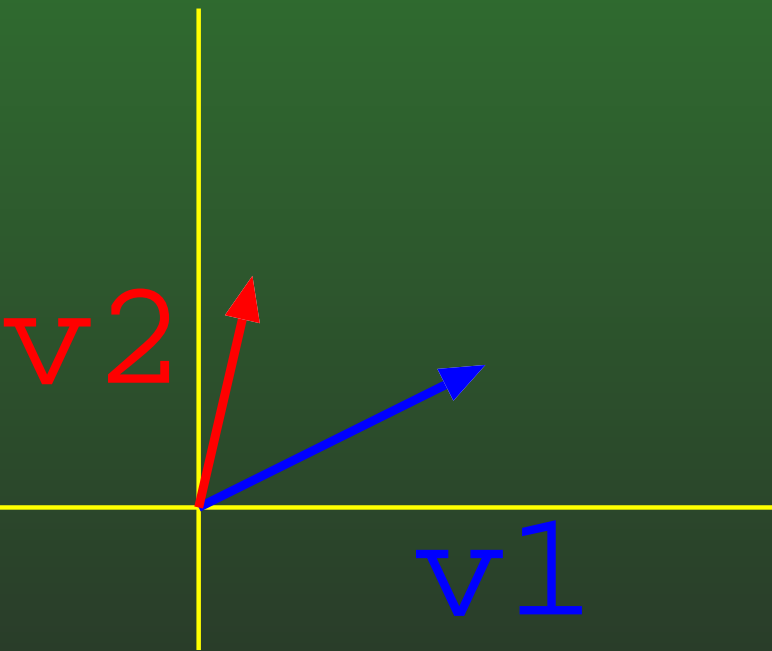


04-36: Normalizing a Vector

- Normalize a vector by setting its length to 1, but maintaining its direction.
- Multiply by $1/\text{length}$
- $\mathbf{v}_{norm} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$
- Of course, v can't be the zero vector
 - Zero vector is the only vector without a direction

04-37: Vector Addition

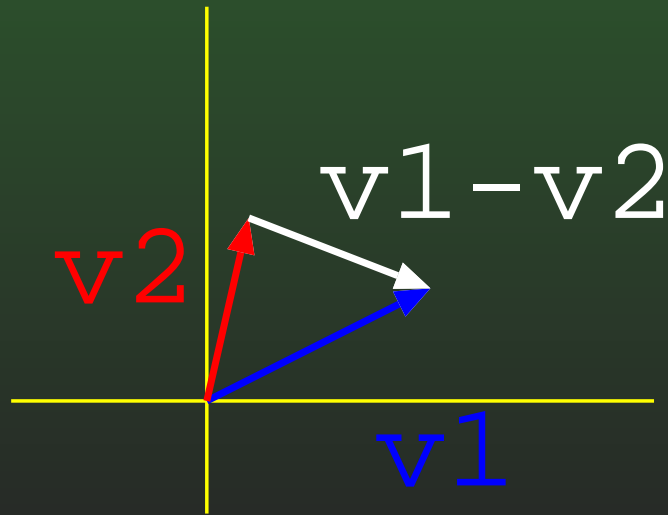
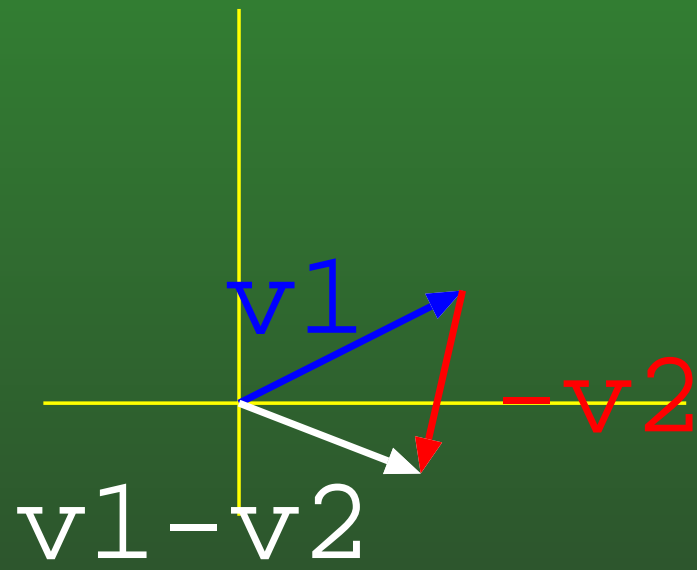
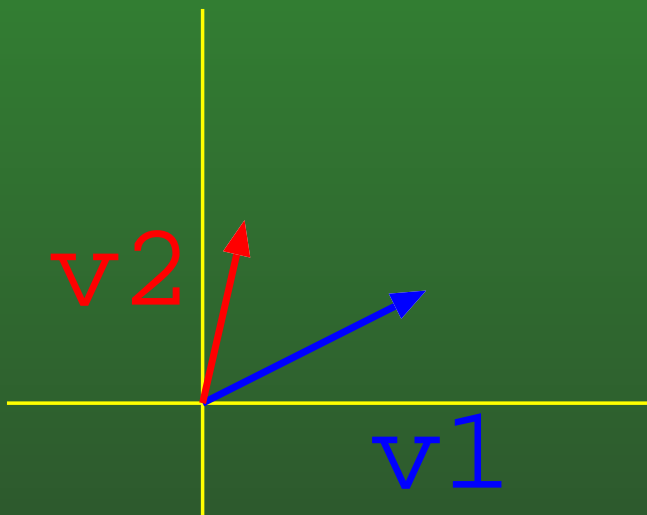
- Add two vectors by adding their components
- $[u_1, u_2, u_3] + [v_1, v_2, v_3] = [u_1 + v_1, u_2 + v_2, u_3 + v + 3]$



04-38: Vector Subtraction

- Vector subtraction is the same as multiplying by -1 and adding
- $\mathbf{v}_1 - \mathbf{v}_2$ is the displacement from the point at \mathbf{v}_2 to the point at \mathbf{v}_1
 - *not* the displacement from \mathbf{v}_1 to \mathbf{v}_2

04-39: Vector Subtraction



04-40: Point Distance

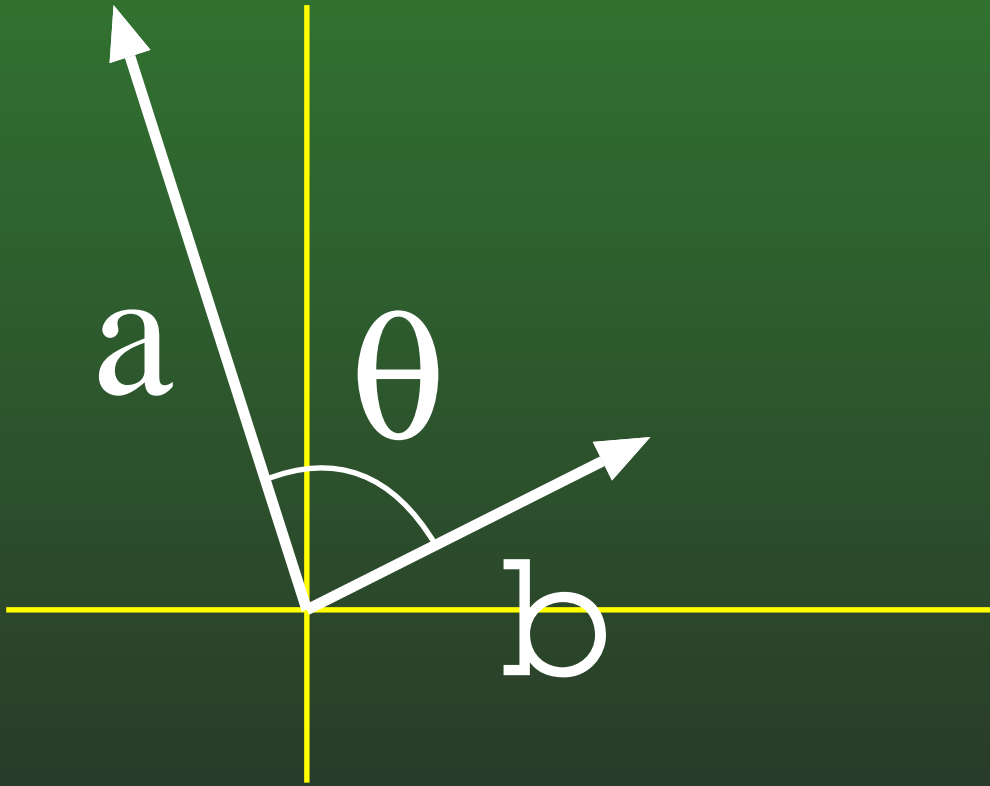
- We can use subtraction and length to find the distance between two points
- Represent points as vectors – displacement from the origin
- Distance from \mathbf{v} to \mathbf{u} is $\|\mathbf{v} - \mathbf{u}\| = \|\mathbf{u} - \mathbf{v}\|$
 - Where $\|\mathbf{v}\|$ is the length of the vector \mathbf{v} .

04-41: Dot Product

- $a = [a_1, a_2, \dots, a_n]$
- $b = [b_1, b_2, \dots, b_n]$
- $a \cdot b = \sum_{i=1}^n a_i b_i$
- $v_1 = [x_1, y_1, z_1], v_2 = [x_2, y_2, z_2]$
- $v_1 \cdot v_2 = x_1 x_2 + y_1 y_2 + z_1 z_2$

04-42: Dot Product

$$a \cdot b = \|a\| * \|b\| * \cos \theta$$



04-43: Dot Product

$$\theta = \arccos \left(\frac{a \cdot b}{\|a\| \|b\|} \right)$$

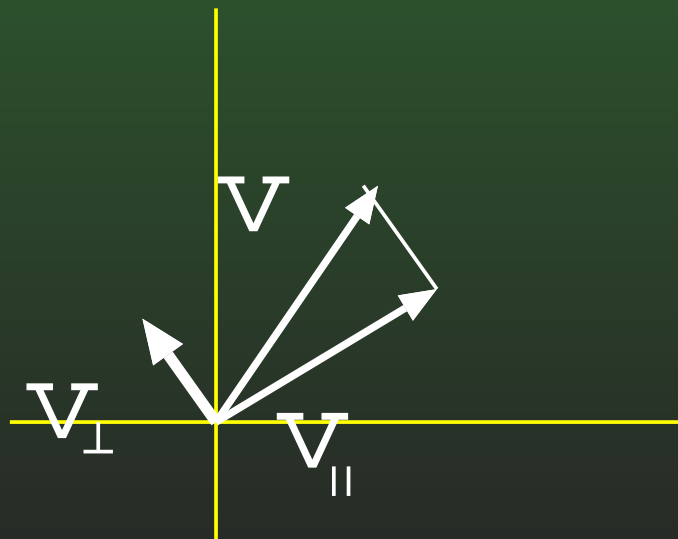
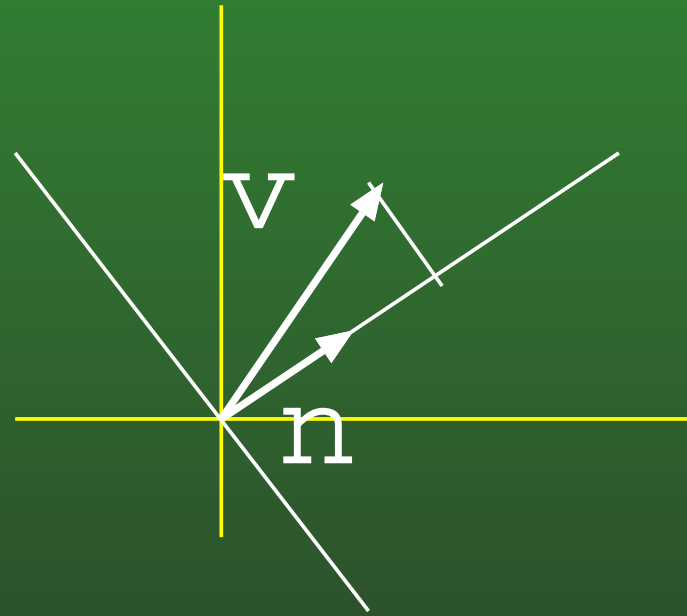
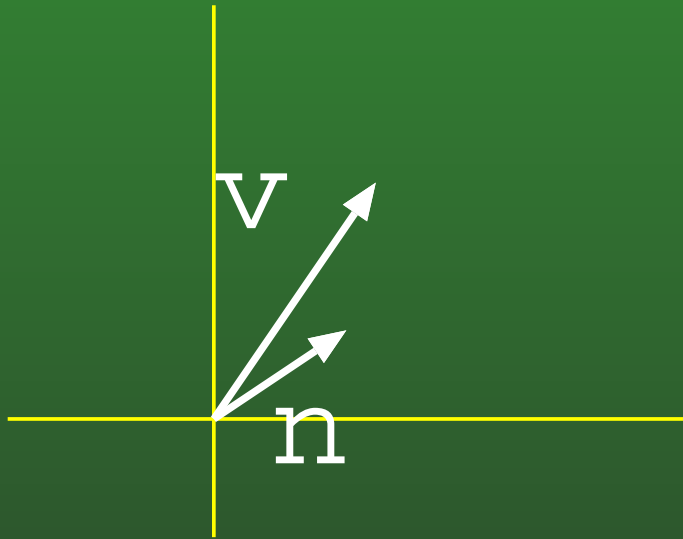
If a and b are unit vectors:

$$\theta = \arccos (a \cdot b)$$

04-44: Dot Product

- If we don't need the exact angle, we can just use the sign
 - If $\theta < 90$, $\cos \theta > 0$
 - If $\theta = 90$, $\cos \theta = 0$
 - If $90 < \theta < 180$, $\cos \theta < 0$
- Since $a \cdot b = \|a\| \|b\| \cos \theta$:
 - If $a \cdot b > 0$, $\theta < 90(\frac{\pi}{2})$
 - If $a \cdot b = 0$, $\theta = 90(\frac{\pi}{2})$
 - If $a \cdot b < 0$, $90 < \theta < 180$
 - $\frac{\pi}{2} < \theta < \pi$

04-45: Projecting Vectors



04-46: Projecting Vectors

- Given a vector v and n , we want to decompose v into two vectors, v_{\parallel} (parallel to n) and v_{\perp} (perpendicular to n)
- $v_{\parallel} = n \frac{\|v_{\parallel}\|}{\|n\|}$
 - So all we need is $\|v_{\parallel}\|$

$$\cos \theta = \frac{\|v_{\parallel}\|}{\|v\|}$$

$$\|v_{\parallel}\| = \cos \theta \|v\|$$

04-47: Projecting Vectors

- $v_{\parallel} = n \frac{\|v_{\parallel}\|}{\|n\|}$
- $\|v_{\parallel}\| = \cos \theta \|v\|$

$$\begin{aligned}v_{\parallel} &= n \frac{\|v_{\parallel}\|}{\|n\|} \\&= n \frac{\cos \theta \|v\|}{\|n\|} \\&= n \frac{\cos \theta \|v\| \|n\|}{\|n\|^2} \\&= n \frac{v \cdot n}{\|n\|^2}\end{aligned}$$

04-48: Projecting Vectors

- Once we have v_{\parallel} , finding v_{\perp} is easy, since
$$v = v_{\parallel} + v_{\perp}$$

$$v_{\parallel} + v_{\perp} = v$$

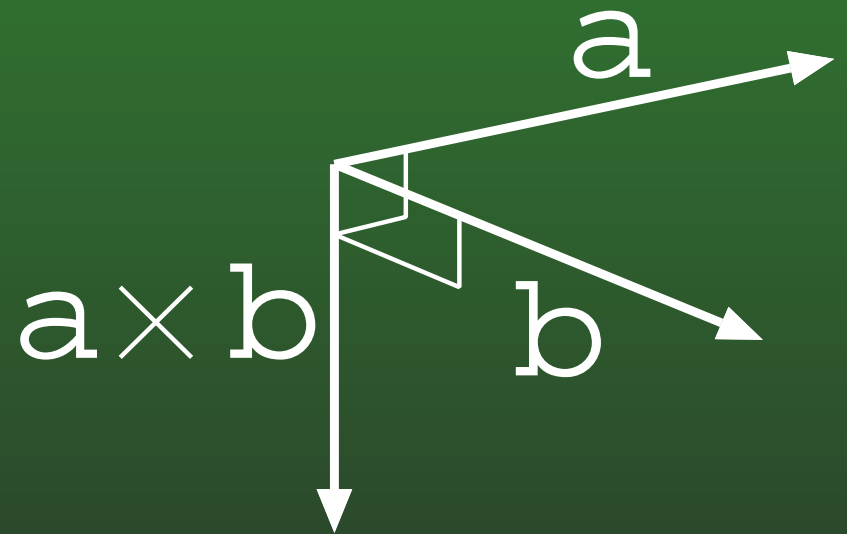
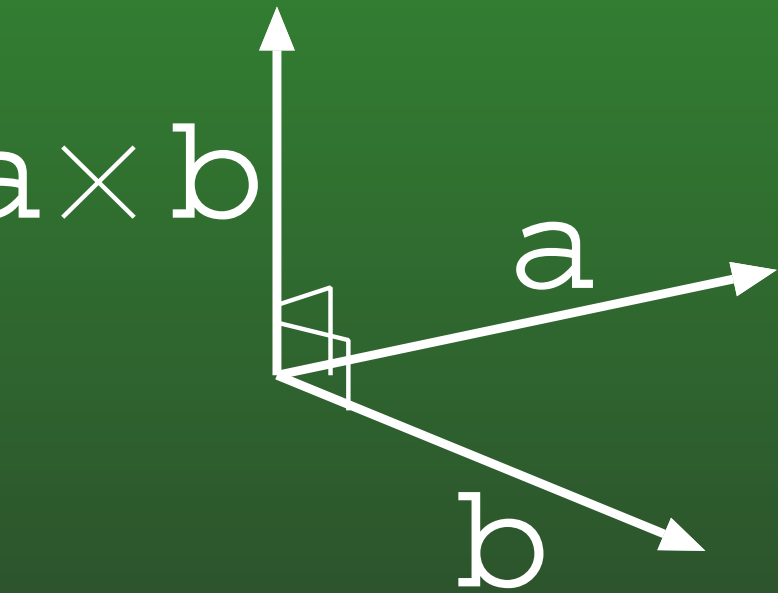
$$v_{\perp} = v - v_{\parallel}$$

$$v_{\perp} = v - n \frac{v \cdot n}{\|n\|^2}$$

04-49: Cross Product

- $v_1 = [x_1, y_1, z_1], v_2 = [x_2, y_2, z_2]$
- $v_1 \times v_2 = [y_1z_2 - z_1y_2, z_1x_2 - x_1z_2, x_1y_2 - y_1x_2]$
- Cross product of two vectors is a new vector perpendicular to the other two vectors

04-50: Cross Product

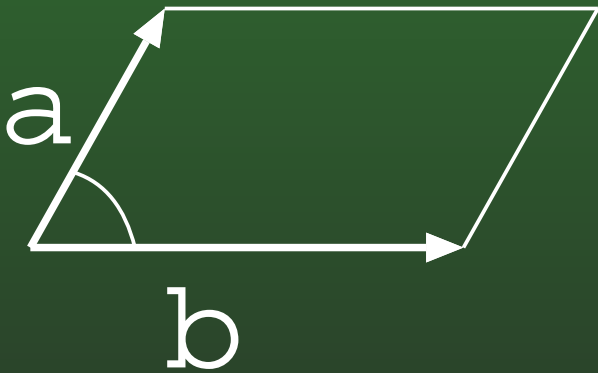


04-51: Cross Product

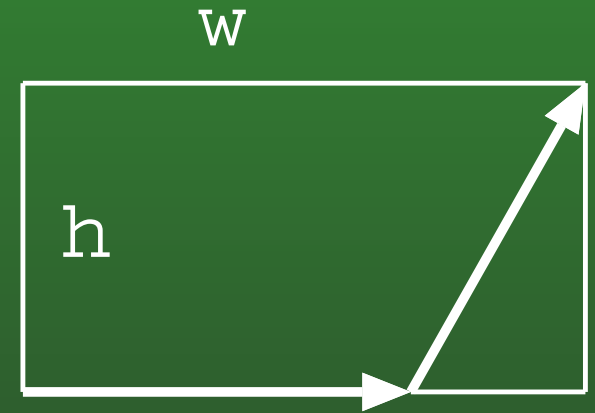
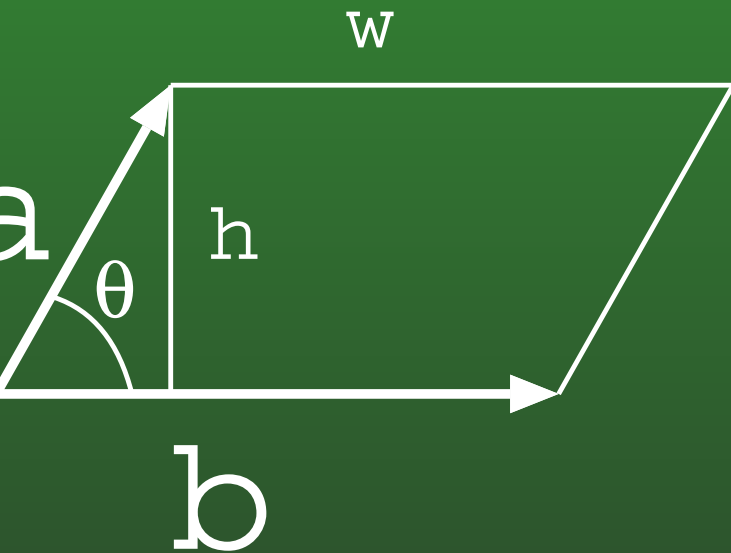
- Which way does the cross product $a \times b$ point?
 - It depends upon your coordinate system – right-handed vs. left-handed
- For right-handed coordinate systems, take your right hand, move your fingers from a to b – thumb points along $a \times b$
- For left-handed coordinate systems, take your left hand, move your fingers from a to b – thumb points along $a \times b$

04-52: Cross Product

- Magnitude of cross product:
 - $\|a \times b\| = \|a\| \|b\| \sin \theta$
- Same as the area of the parallelogram defined by a and b



04-53: Cross Product



- Area of parallelogram = $w * h$
- $w = ||b||$
- $\sin \theta = h / ||a||$, $h = ||a|| \sin \theta$
- $w * h = ||a|| ||b|| \sin \theta = ||a \times b||$

04-54: Matrices

- A 4x3 matrix M :

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \\ m_{41} & m_{42} & m_{43} \end{bmatrix}$$

04-55: Matrices

- A Square matrix has the same width and height

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

- A diagonal matrix is a square matrix with non-diagonal elements equal to zero

$$\mathbf{M} = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & 0 \\ 0 & 0 & m_{33} \end{bmatrix}$$

04-56: Matrices

- The *Identity Matrix* is a diagonal matrix with all diagonal elements = 1

$$\mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

04-57: Matrices

- Matrices and vectors
 - Vectors are a special case of matrices
 - Row vectors (as we've seen so far) $[x, y, z]$
 - Column vectors : $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$

04-58: Matrices

- Transpose
 - Written M^T
 - Exchange rows and columns

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{bmatrix}^T = \begin{bmatrix} a & d & g & j \\ b & e & h & k \\ c & f & i & l \end{bmatrix}$$

04-59: Transpose

- The transpose of a row vector is a column vector
- For any matrix M , $(M^T)^T = M$
- For a diagonal matrix D , $D^T = ?$

04-60: **Transpose**

- The transpose of a row vector is a column vector
- For any matrix M , $(M^T)^T = M$
- For a diagonal matrix D , $D^T = D$
 - True for any matrix that is symmetric along the diagonal

04-61: Matrix Multiplication

- Multiplying a Matrix by a scalar
 - Multiply each element in the Matrix by the scalar
 - Just like multiplying a vector by a scalar

$$k\mathbf{M} = k \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \\ m_{41} & m_{42} & m_{43} \end{bmatrix} = \begin{bmatrix} km_{11} & km_{12} & km_{13} \\ km_{21} & km_{22} & km_{23} \\ km_{31} & km_{32} & km_{33} \\ km_{41} & km_{42} & km_{43} \end{bmatrix}$$

04-62: Matrix Multiplication

- Multiplying two matrices A and B
- A dimensions $n \times m$, B dimensions $m \times p$
- $C = AB$
 - C dimensions $n \times p$

$$c_{ij} = \sum_{k=1}^m a_{ik} b_{kj}$$

04-63: Matrix Multiplication

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

04-64: Matrix Multiplication

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix}$$

$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}$$

04-65: Matrix Multiplication

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix}$$

$$c_{21} = a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31}$$

04-66: Matrix Multiplication

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix}$$

$$c_{12} = a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32}$$

04-67: Matrix Multiplication

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix}$$

$$c_{22} = a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32}$$

04-68: Matrix Multiplication

- Vectors are special cases of matrices
- Multiplying a vector and a matrix is just like multiplying two matrices

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

$$\begin{bmatrix} xm_{11} + ym_{21} + zm_{31} & xm_{12} + ym_{22} + zm_{32} & xm_{13} + ym_{23} + zm_{33} \end{bmatrix}$$

04-69: Matrix Multiplication

- Vectors are special cases of matrices
- Multiplying a vector and a matrix is just like multiplying two matrices

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} xm_{11} + ym_{12} + zm_{13} \\ xm_{21} + ym_{22} + zm_{23} \\ xm_{31} + ym_{32} + zm_{33} \end{bmatrix}$$

04-70: Matrix Multiplication

- Note that the following multiplications are not legal:

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

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

04-71: Matrix Multiplication

- Matrix Multiplication is not commutative: $AB \neq BA$ (at least not for all A and B – is it true for at least one A and B ?)
- Matrix Multiplication is associative: $(AB)C = A(BC)$
- Transposing product is the same as the product of the transpose, in reverse order: $(AB)^T = B^T A^T$

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

04-72: Matrix Multiplication

- Matrix Multiplication is not commutative: $AB \neq BA$ (at least not for all A and B – is it true for at least one A and B ?)
- Matrix Multiplication is associative: $(AB)C = A(BC)$
- Transposing product is the same as the product of the transpose, in reverse order: $(AB)^T = B^T A^T$

$$\left(\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right)^T = \begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \end{bmatrix}$$

04-73: Matrix Multiplication

- Identity Matrix I :
 - $AI = A$ (for appropriate I)
 - $IA = A$ (for appropriate I)

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

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

04-74: Matrix Multiplication

- Identity Matrix I :
 - $AI = A$ (for appropriate I)
 - $IA = A$ (for appropriate I)

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

04-75: Matrix Multiplication

- Identity Matrix I :
 - $AI = A$ (for appropriate I)
 - $IA = A$ (for appropriate I)

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

04-76: Matrix Multiplication

- Identity Matrix I :
 - $AI = A$ (for appropriate I)
 - $IA = A$ (for appropriate I)

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

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix}$$

04-77: Row vs. Column Vectors

- A vector can be represented as a row vector or a column vector
- This makes a difference when using matrices
 - Row: vA , Column $A v$
- It gets even more fun when using matrices to do several transformations of a vector:
 - Row $vABC$, Column $CBA v$ (note that to get the same transformation, you need to take the transpose of A , B , and C when swapping between row and column vectors)

04-78: Row vs. Column Vectors

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix}$$

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$$

$$= \begin{bmatrix} ax + by + cz & dx + ey + fz & gx + hy + iz \end{bmatrix}$$

04-79: Row vs. Column Vectors

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} =$$

$$\begin{bmatrix} xa + yc & xb + yd \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} =$$

$$\begin{bmatrix} (xa + yc)e + (xb + yd)g & (xa + yc)f + (xb + yd)h \end{bmatrix}$$

$$\begin{bmatrix} e & g \\ f & h \end{bmatrix} \begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} =$$

$$\begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} ax + cy \\ bx + dy \end{bmatrix} =$$

$$\begin{bmatrix} e(ax + cy) + g(ax + cy) \\ f(ax + cy) + h(ax + dy) \end{bmatrix}$$

04-80: Row vs. Column Vectors

- DirectX and the text use row vectors
- OpenGL and Ogre use column vectors
 - Ogre has a back end for both OpenGL and Direct3D
 - Ogre transposes matrices before sending them to D3D libraries
- Lecture will use both
 - This is on purpose
 - I want you to really understand what's going on, not just memorize formulas

04-81: More Matrices

- Consider the vector $[x, y, z]$

$$V = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ y \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ z \end{bmatrix}$$

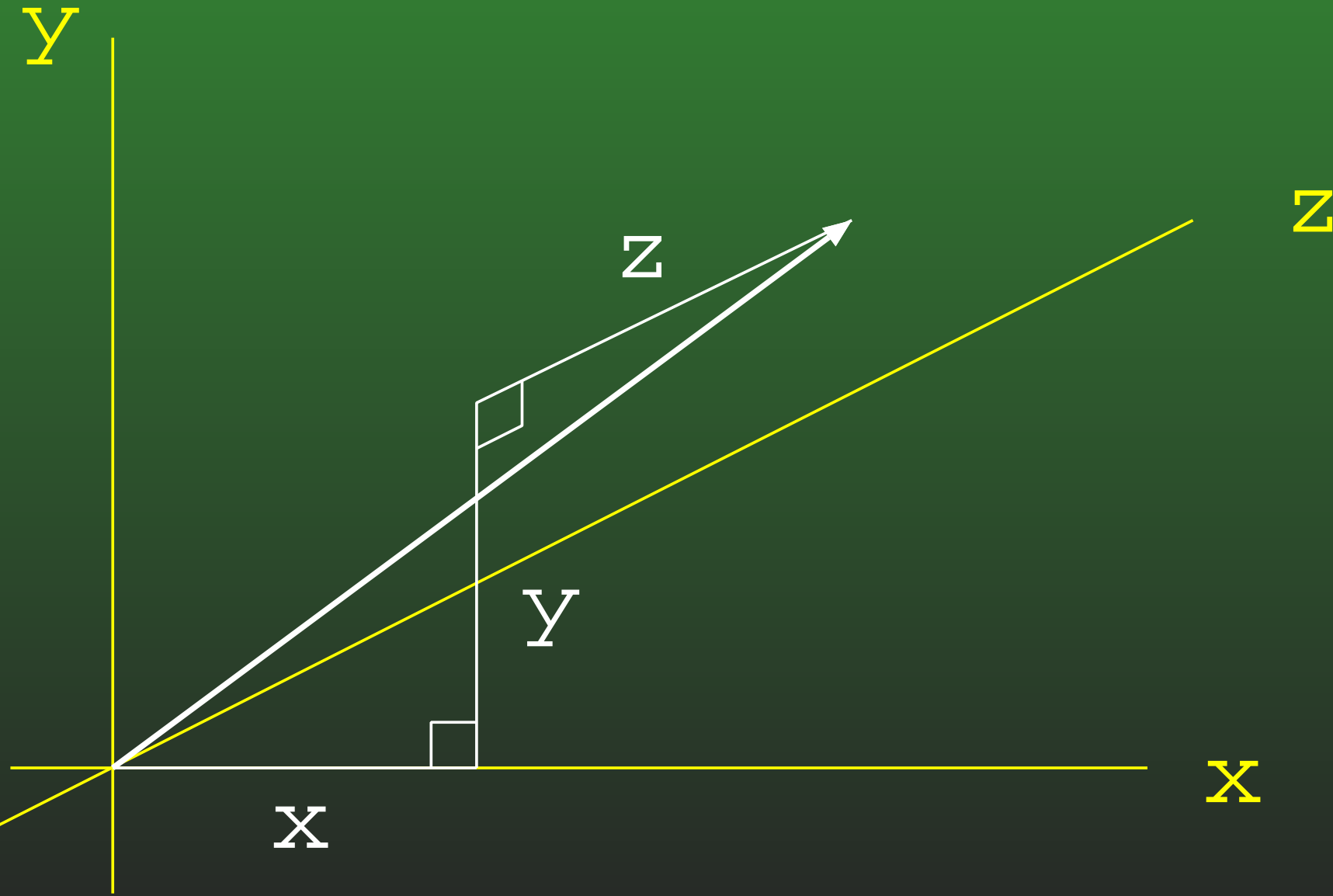
- Rewrite as:

$$V = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = x \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + y \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + z \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

04-82: More Matrices

- let \mathbf{p} , \mathbf{q} , \mathbf{r} be unit vectors for $+x$, $+y$ and $+z$
- $\mathbf{v} = x\mathbf{p} + y\mathbf{q} + z\mathbf{r}$
- We have defined v as a linear combination of \mathbf{p} , \mathbf{q} and \mathbf{r} .
 - \mathbf{p} , \mathbf{q} , and \mathbf{r} are *basis vectors*

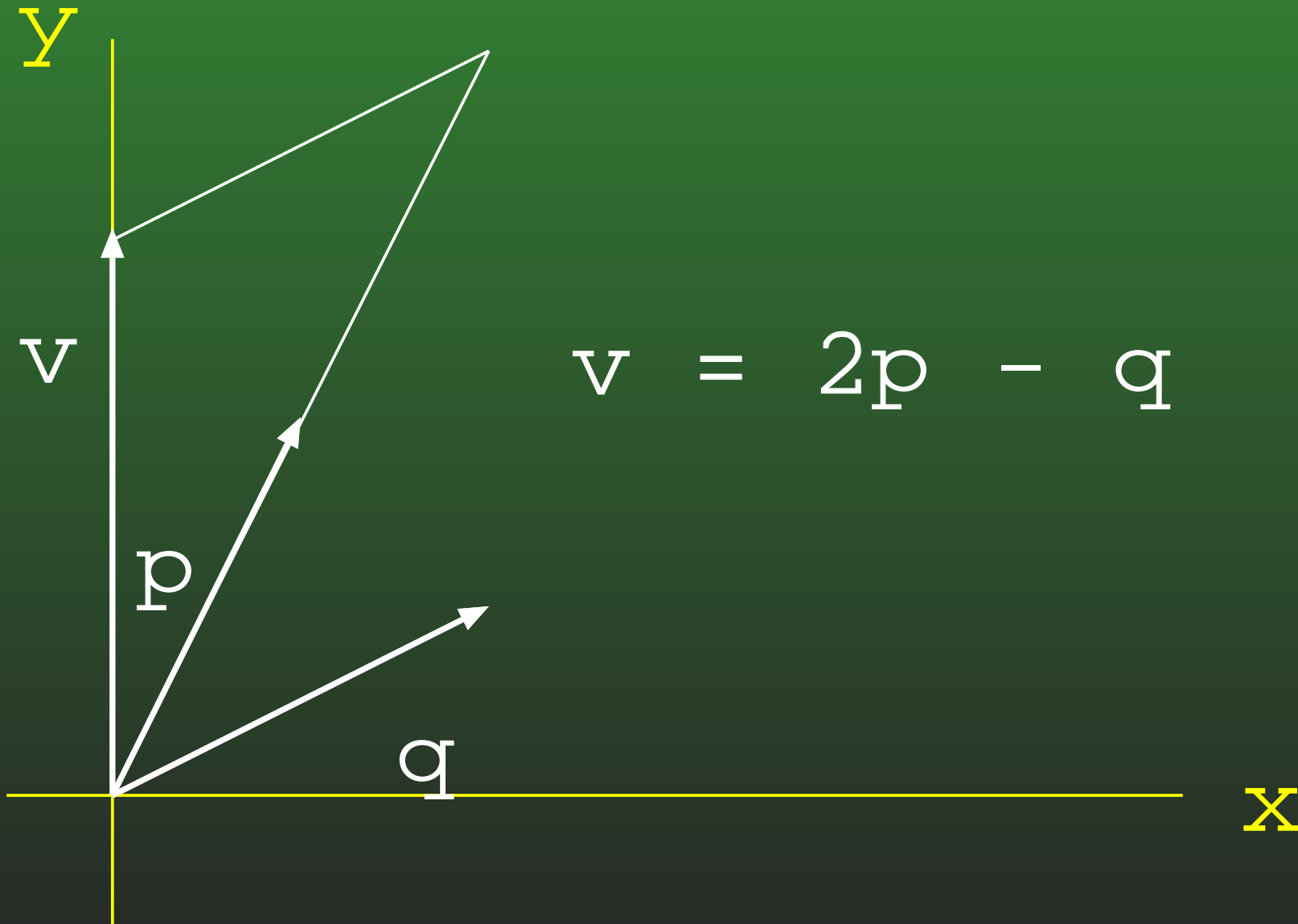
04-83: Basis Vectors



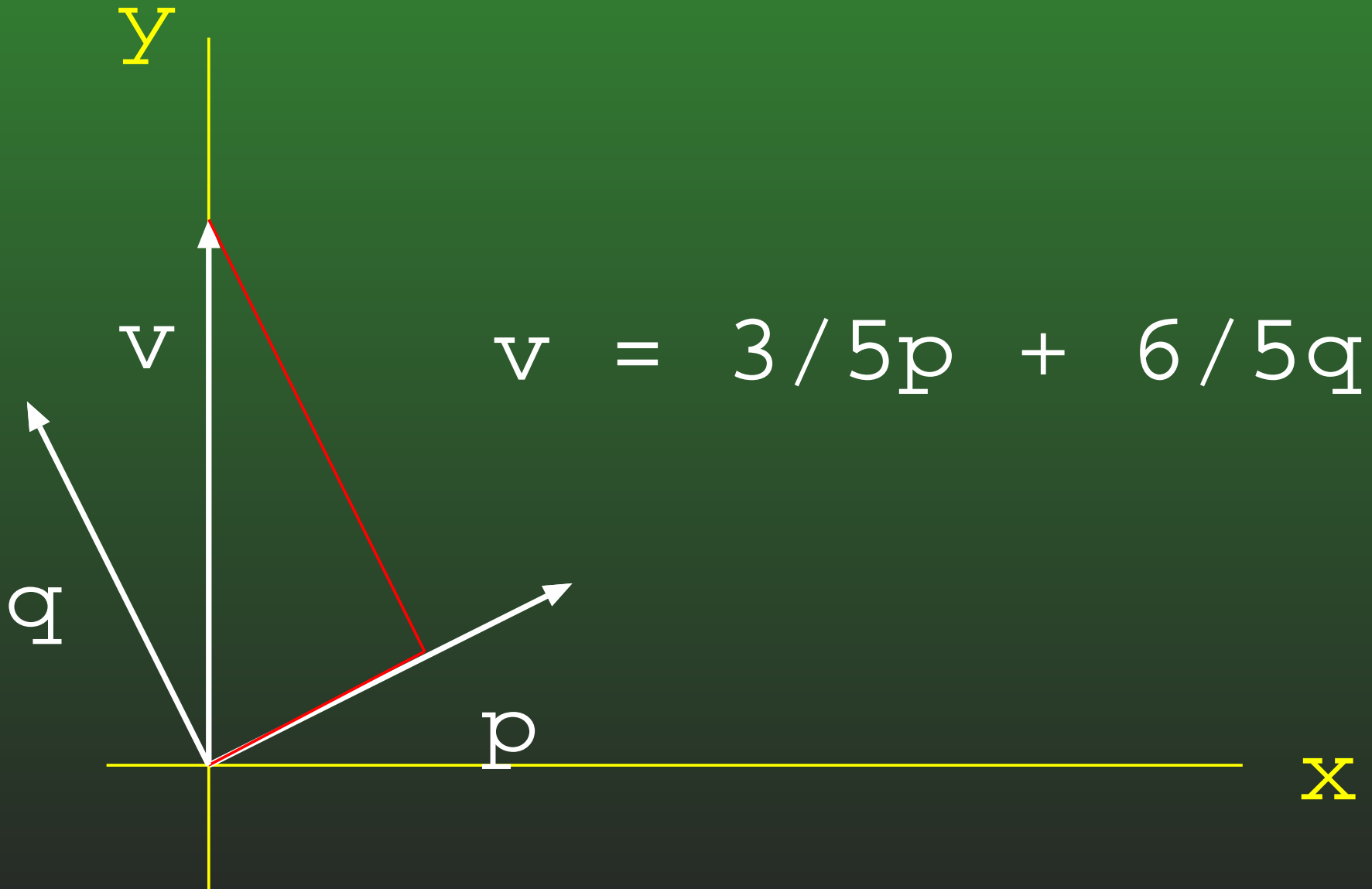
04-84: Basis Vectors

- p , q , and r are unit vectors along the X , Y and Z axes – we're used to seeing vectors decomposed this way
- Technically, *any* 3 linearly-independent vectors could be used as basis vectors
- Typically, mutually perpendicular vertices are used as basis vectors
- Basis vectors not aligned with axes: Object space rotated from world space

04-85: Non-Perpendicular Basis



04-86: Perpendicular Basis



04-87: Matrices & Basis

- Look back at our basis vectors \mathbf{p} , \mathbf{q} and \mathbf{r} .
- Create a 3x3 matrix M using \mathbf{p} , \mathbf{q} and \mathbf{r} as rows:

$$M = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{r} \end{bmatrix} = \begin{bmatrix} p_x & p_y & p_z \\ q_x & q_y & q_z \\ r_x & r_y & r_z \end{bmatrix}$$

- Multiply a vector by this matrix:

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} p_x & p_y & p_z \\ q_x & q_y & q_z \\ r_x & r_y & r_z \end{bmatrix} =$$

04-88: Matrices & Basis

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} p_x & p_y & p_z \\ q_x & q_y & q_z \\ r_x & r_y & r_z \end{bmatrix} =$$

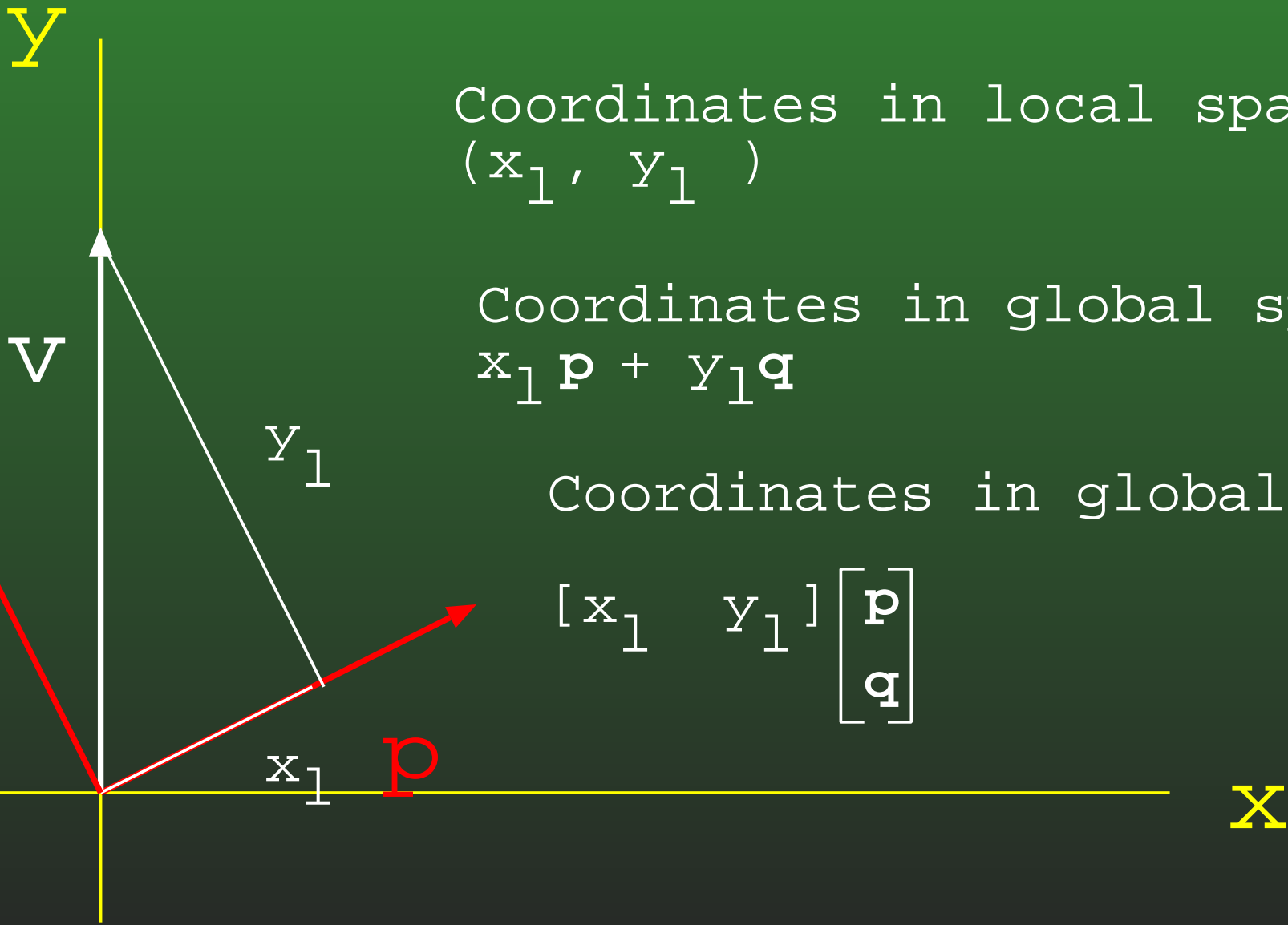
$$\begin{bmatrix} xp_x + yq_x + zr_x & xp_y + yq_y + zr_y & xp_z + yq_z + zr_z \end{bmatrix} =$$

$$x\mathbf{p} + y\mathbf{q} + z\mathbf{r}$$

04-89: Matrices & Basis

- This is really cool. Why?
 - Take a local space, defined by 3 basis vectors
 - Rotation only (no translation)
 - Create a matrix with these vectors as rows (or cols)
 - Matrix transforms from local space into global space

04-90: Matrices & Basis



Coordinates in local space
 (x_1, y_1)

Coordinates in global space
 $x_1 \mathbf{p} + y_1 \mathbf{q}$

Coordinates in global space

$$[x_1 \quad y_1] \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix}$$

04-91: Matrices as Transforms

- A 3x3 matrix is a transform
 - Transforms a vector
 - Since a 3D model is just a series of points, can also transform a model
 - Transforming each point in the model
- What does the transformation look like?
- Can you look at the matrix, and see what the transformation will be?

04-92: Matrices as Transforms

- Let's look at what happens when we multiply the basis vectors $[1, 0, 0]$, $[0, 1, 0]$ and $[0, 0, 1]$ by an arbitrary matrix:

04-93: Matrices as Transforms

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

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

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

04-94: Matrices as Transforms

- Each row of the matrix is a basis vector after transformation
 - (Or each column of the matrix, if we're using column vectors)
- Let's look at an example in 2D:

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

- What happens when we transform a vector (or a 2D polygon) using this matrix?
- Assume row vectors for the moment ...

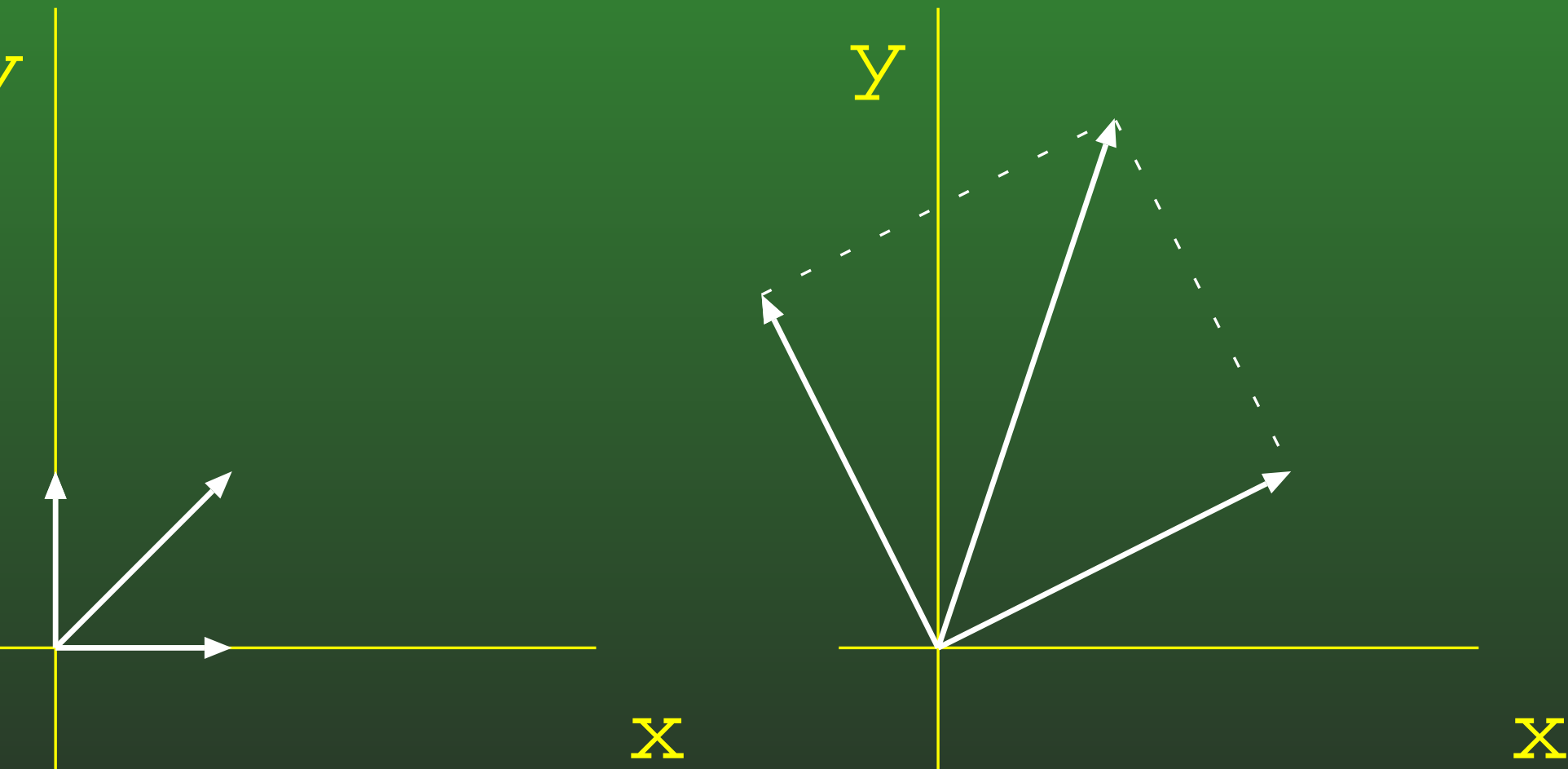
04-95: Matrices as Transforms

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

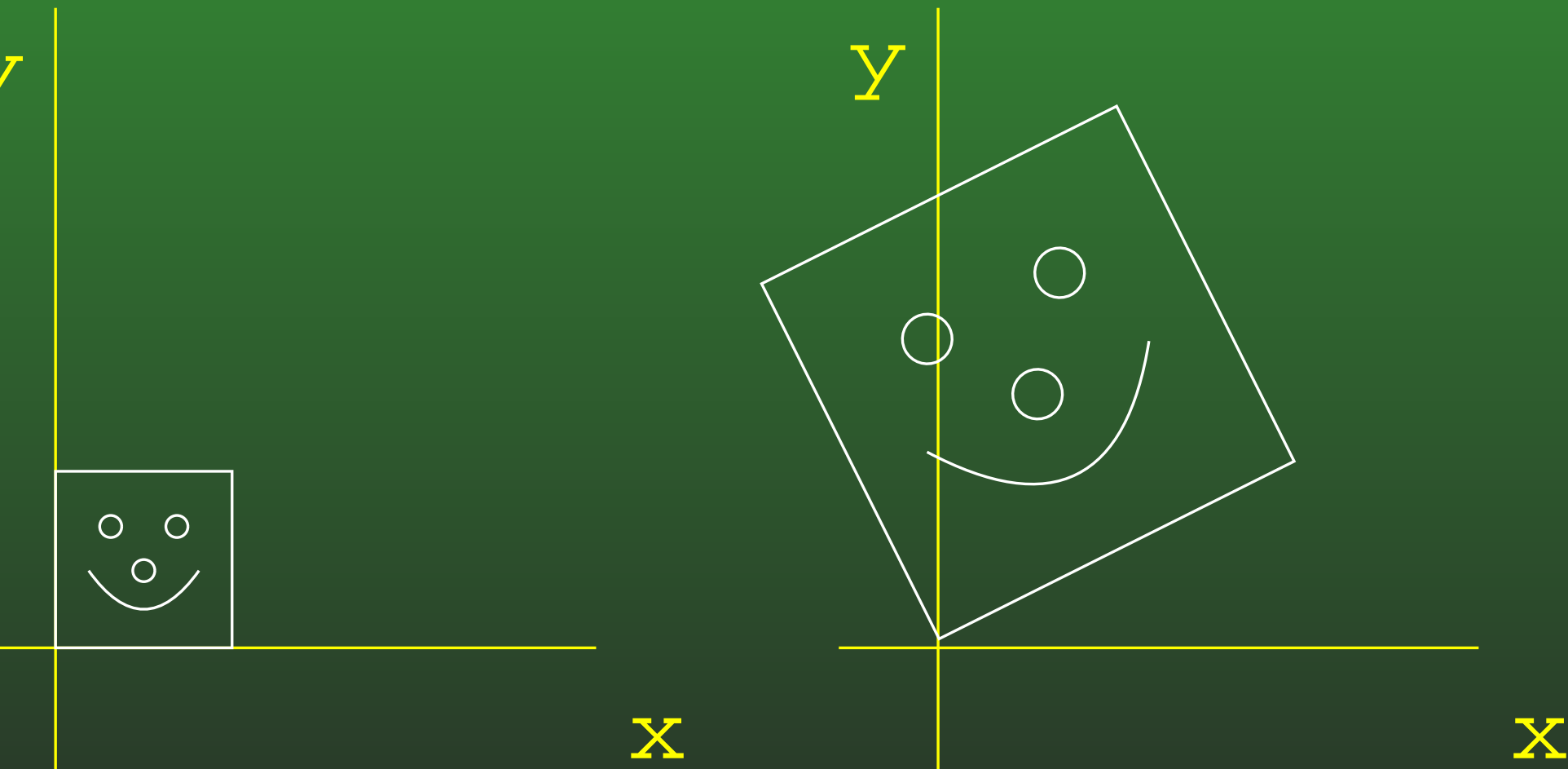
$$\begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} -1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 3 \end{bmatrix}$$

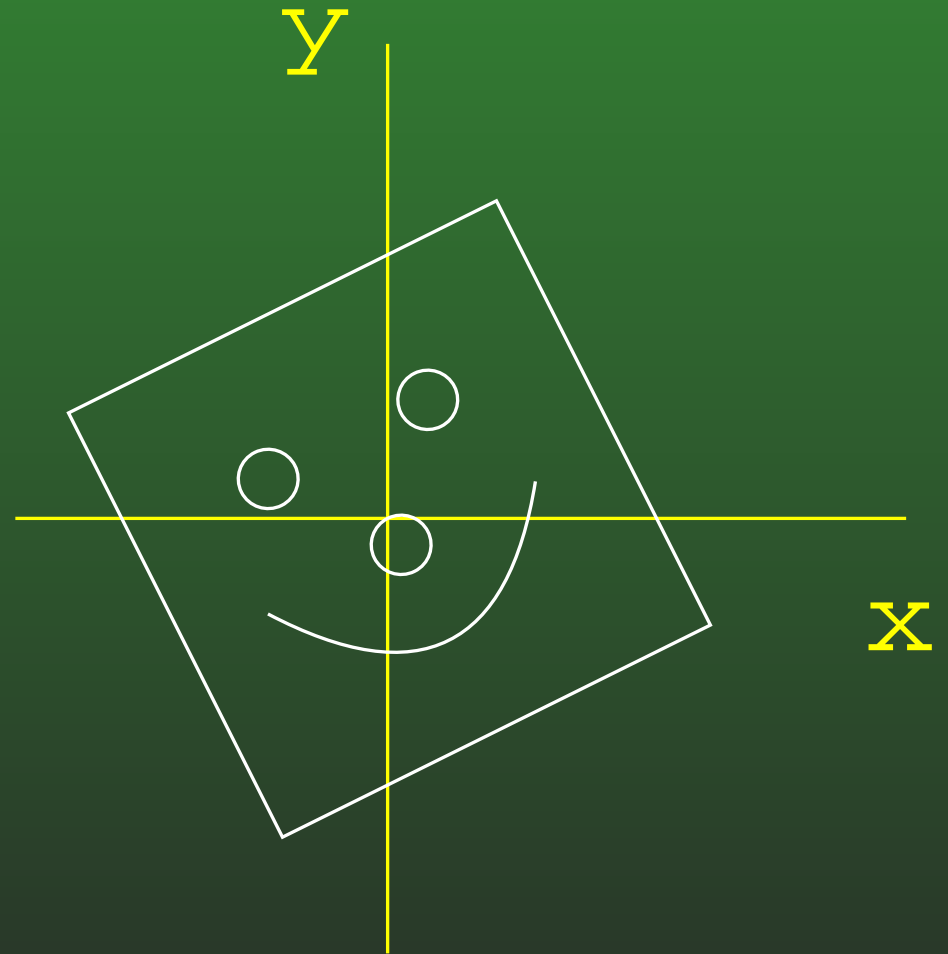
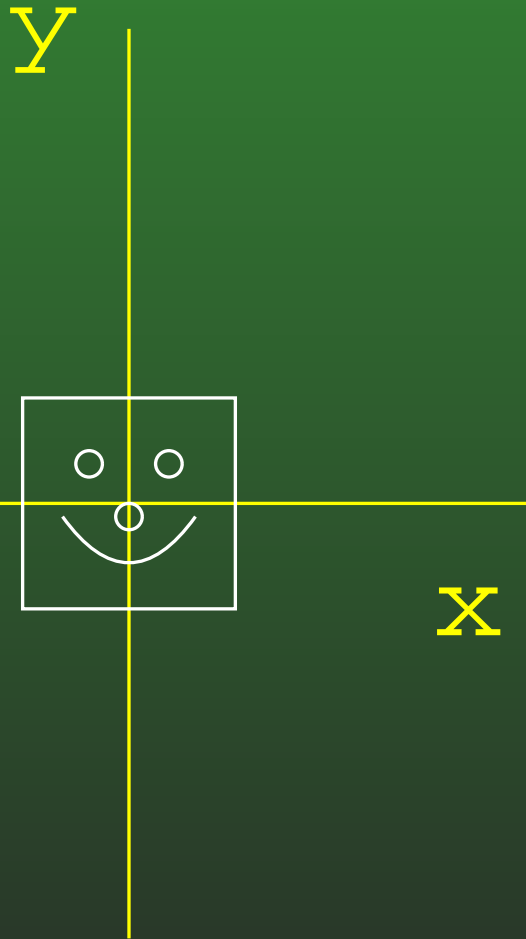
04-96: Matrices as Transforms



04-97: Matrices as Transforms



04-98: Matrices as Transforms



04-99: Matrices as Transforms

- The matrix:

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

both scaled and rotated a 2D image

- It is possible, of course for a matrix to just scale, or just rotate an image as well

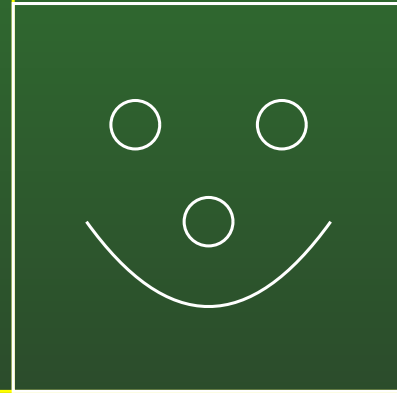
04-100: Matrices as Transforms

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



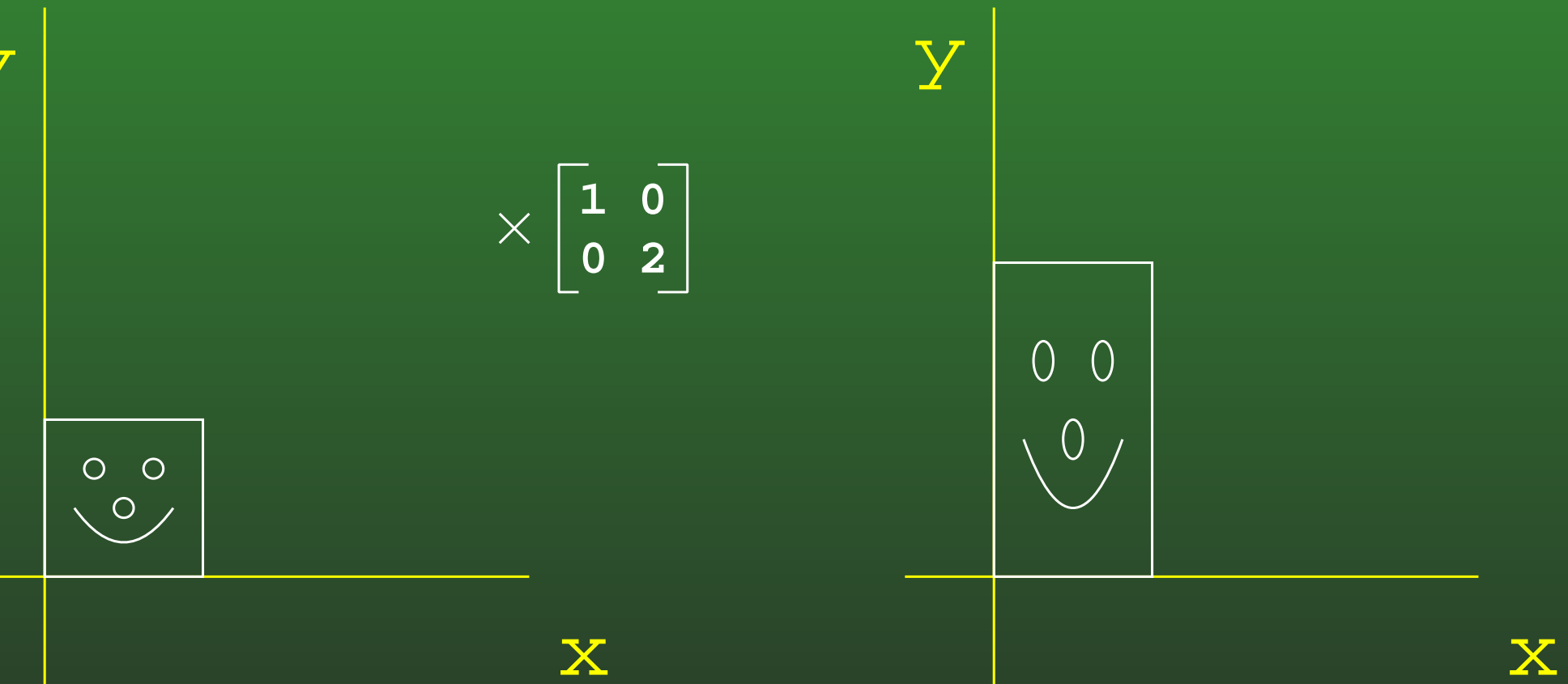
x

y



x

04-101: Matrices as Transforms



04-102: Matrices as Transforms

- Can a matrix do something other than scale and rotate?

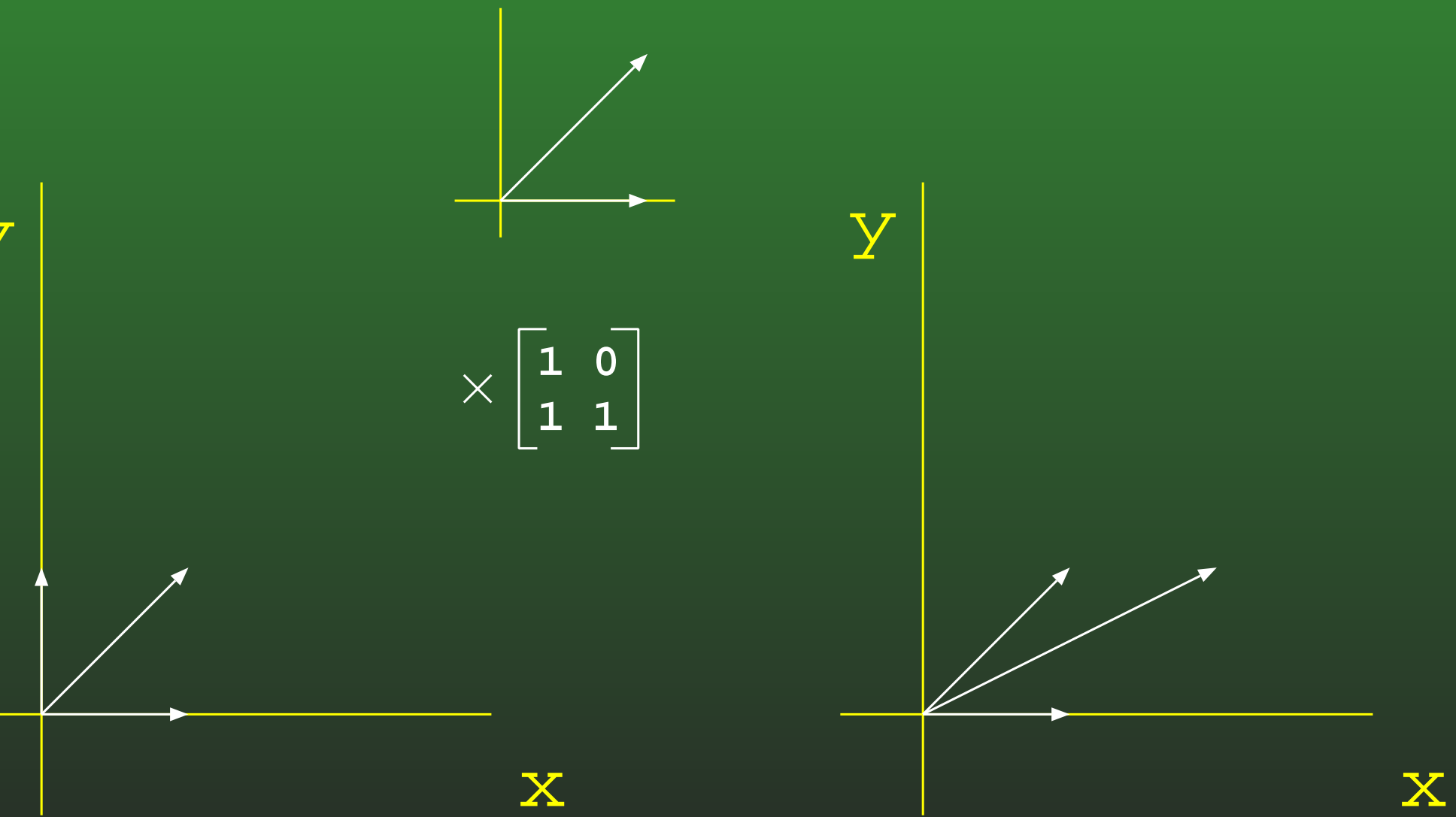
04-103: Matrices as Transforms

- Can a matrix do something other than scale and rotate?
 - Yes!
- What would a matrix that did something other than scale or rotate look like? (stay 2D, for the moment)

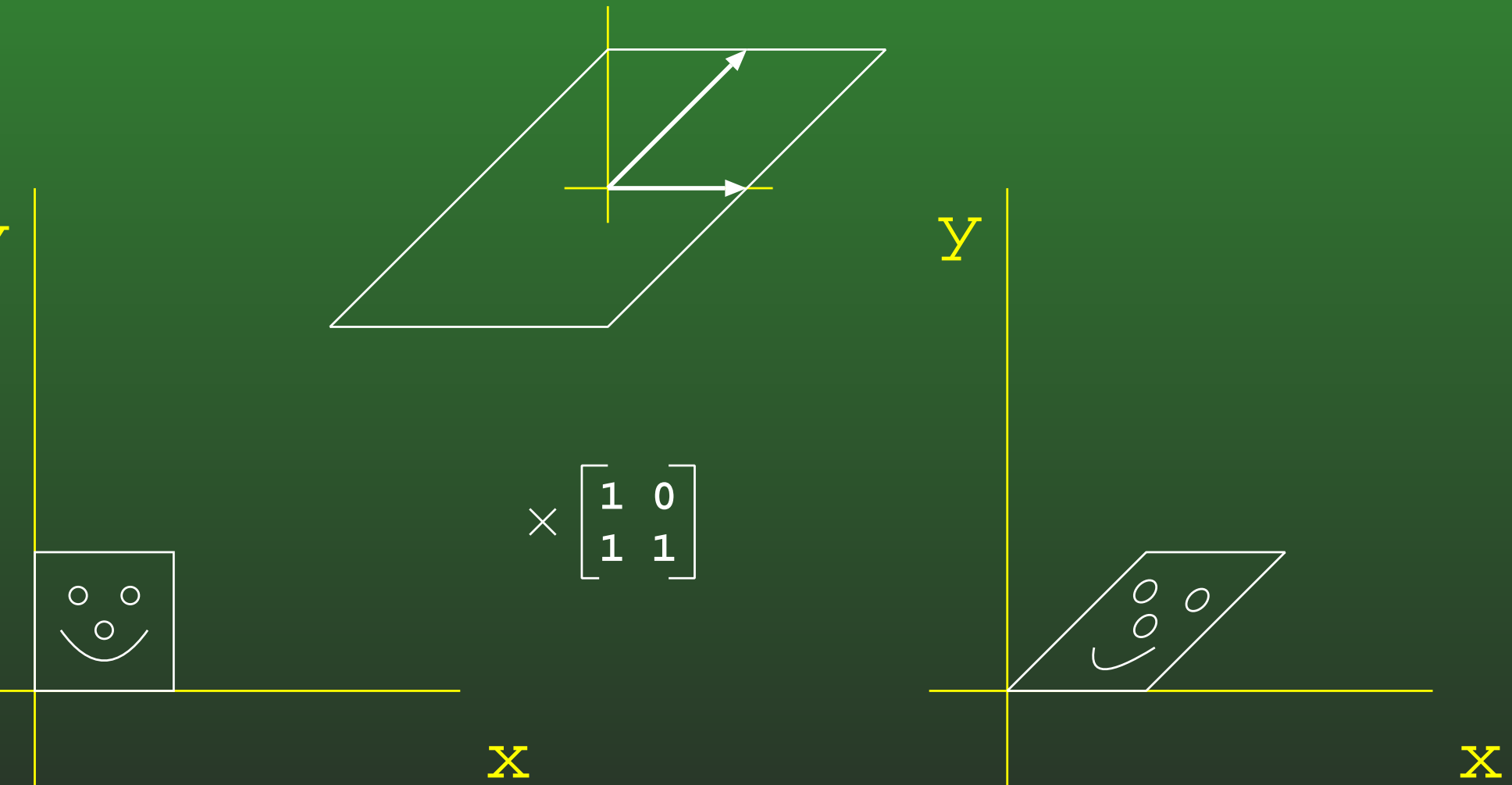
04-104: Matrices as Transforms

- Can a matrix do something other than scale and rotate?
 - Yes!
- What would a matrix that did something other than scale or rotate look like? (stay 2D, for the moment)
 - “Basis vectors” in matrix non-orthogonal

04-105: Matrices as Transforms



04-106: Matrices as Transforms



04-107: Matrices as Transforms

- This translates (reasonably) easily into 3D
- Instead of stretching, rotating, or skewing part of a plane, stretch, rotate, or skew a cube

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

No transformation (or identity transformation)

04-108: Matrices as Transforms

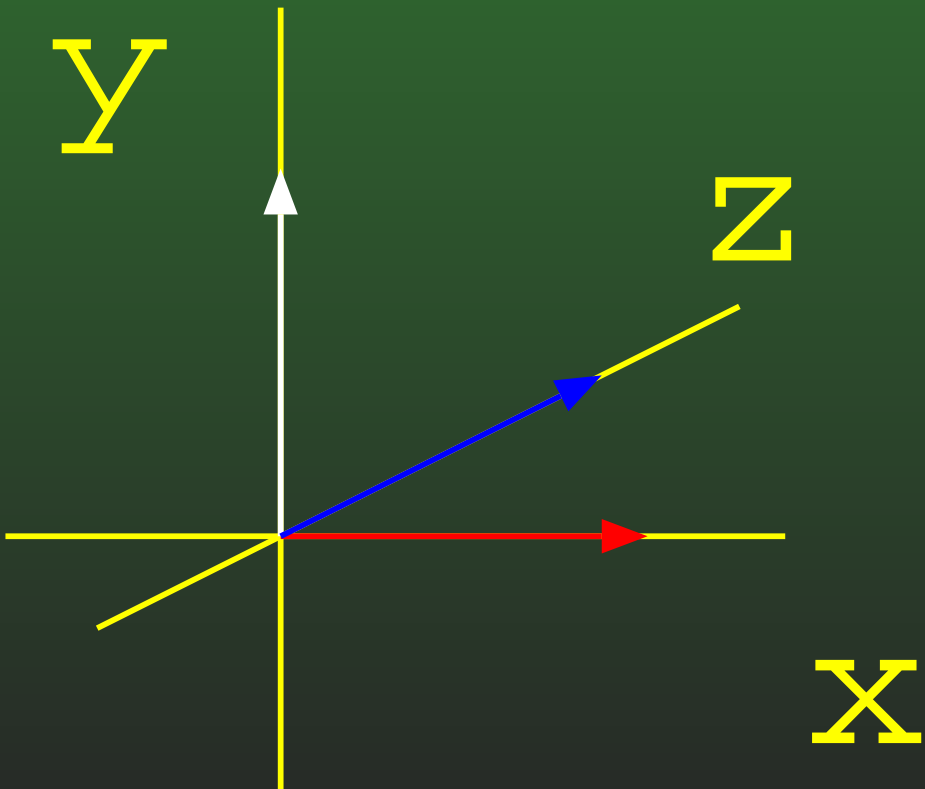
- This translates (reasonably) easily into 3D
- Instead of stretching, rotating, or skewing part of a plane, stretch, rotate, or skew a cube

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

What is this?

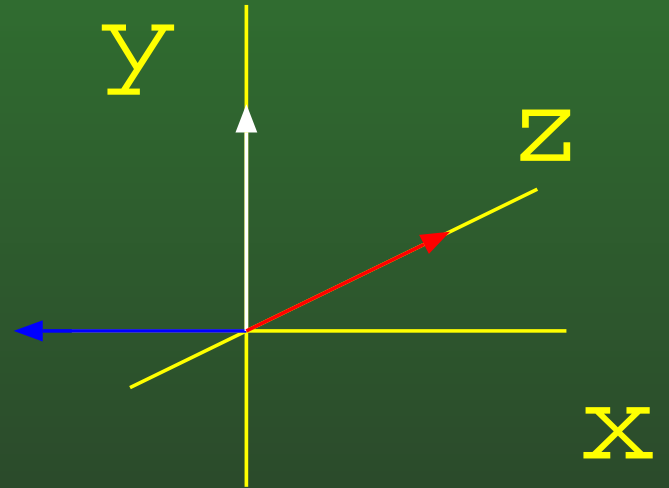
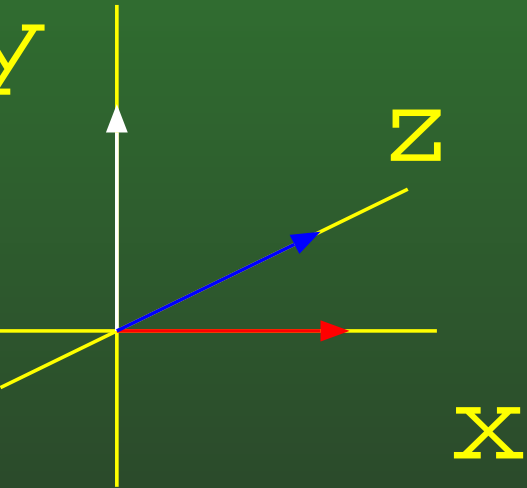
04-109: Matrices as Transforms

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



04-110: Matrices as Transforms

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



04-111: Matrices as Transforms

- This translates (reasonably) easily into 3D
- Instead of stretching, rotating, or skewing part of a plane, stretch, rotate, or skew a cube

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

Rotation about the Y axis, $\frac{\pi}{2}$ (90 degrees)