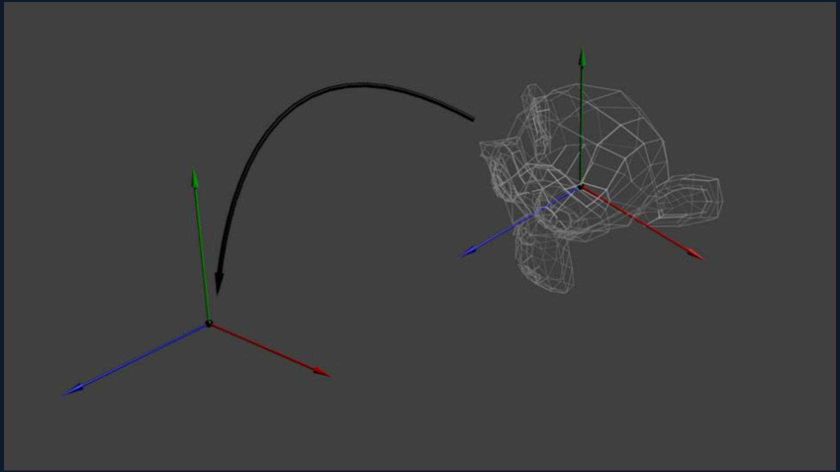
Matrix Transformations



Canberra Institute of Technology

aie

SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Lecture Contents

- Introduction to Matrix Transformations
- Coordinate Systems/Spaces
- Review of Matrix Multiplication
- Structure of a transformation Matrix
- How a transformation works
  - Translation
  - Rotation
  - Scale
- Concatenating Matrices
- Why use transformations?
  - Instancing
  - Parenting
  - Cameras

Canberra Institute of Technology

SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

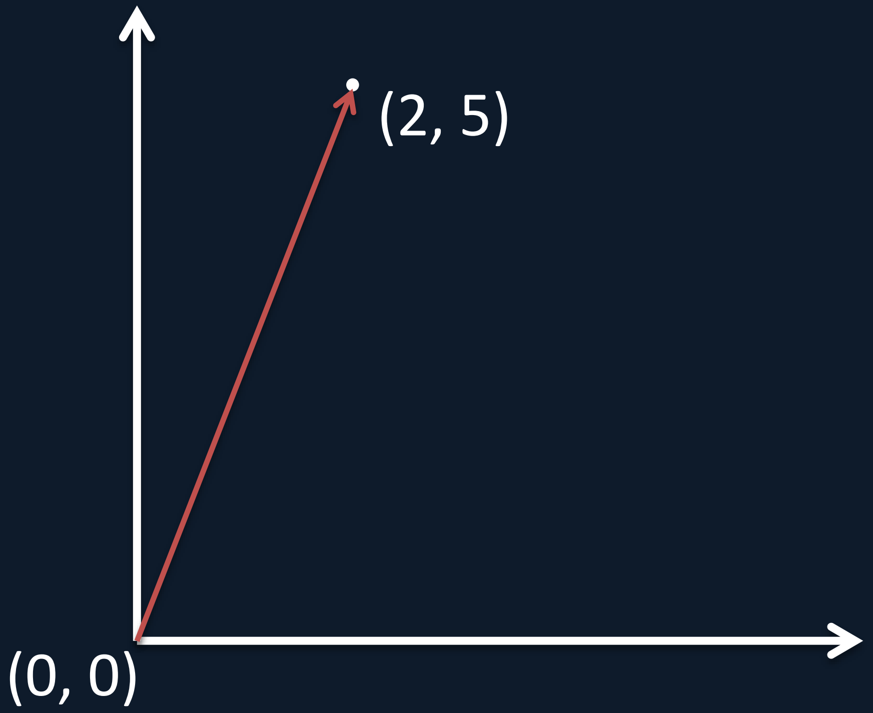
What is a matrix transformation?

- A matrix transformation is the process of multiplying a matrix by a vector in order to change where that vector is pointing.
- Can be used to move the vector in a straight line, change the magnitude of the vector or to rotate the vector around the current origin.
- Normally a combination of all 3

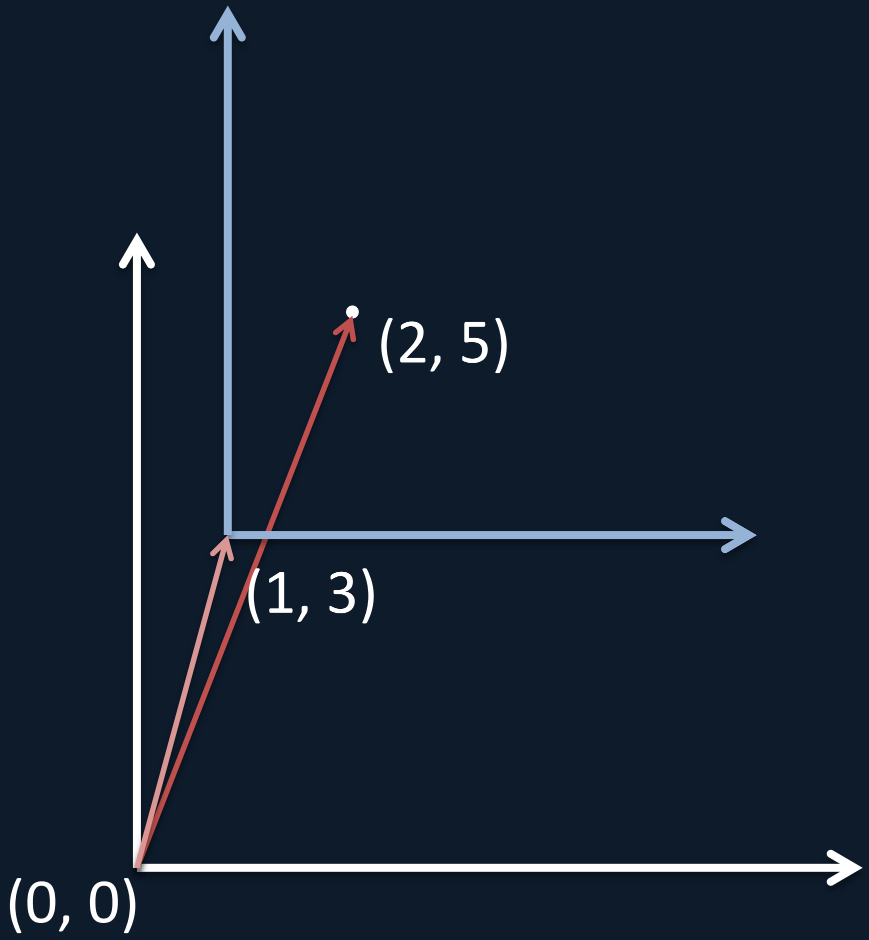
Coordinate spaces

- Storing vectors in different frames of reference.
- Essentially, moving the origin.

- Moving the origin? How would that work?



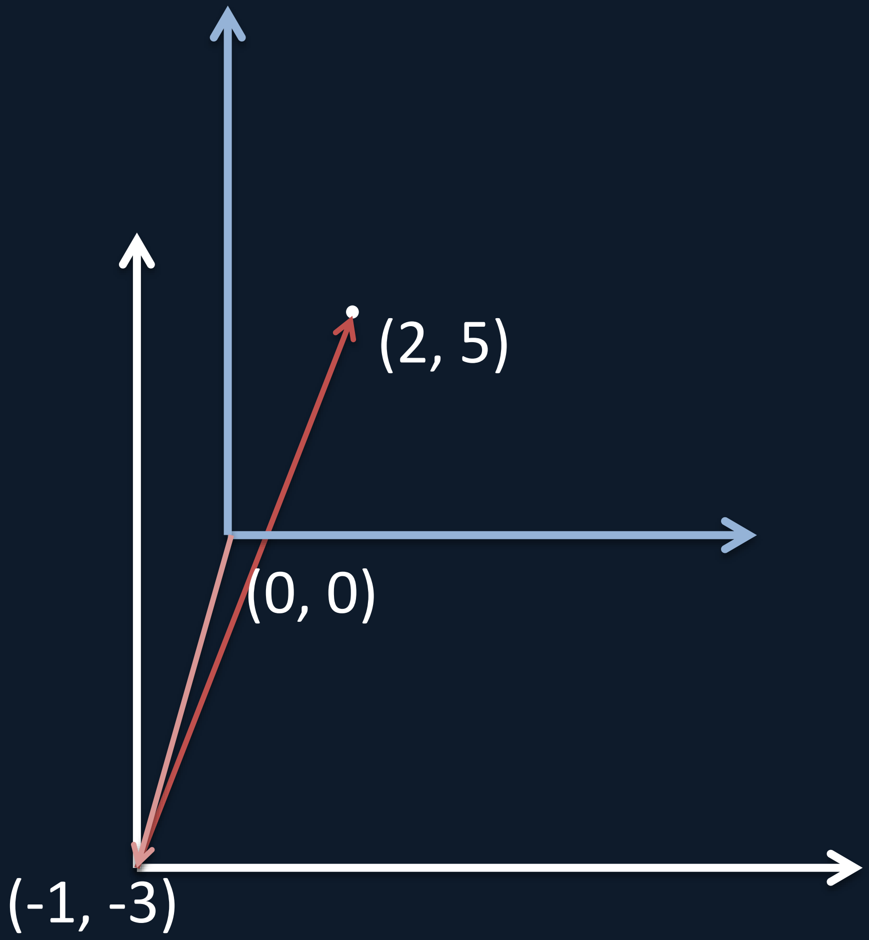
Here we have a coordinate system with a single point represented by the vector (2, 5)



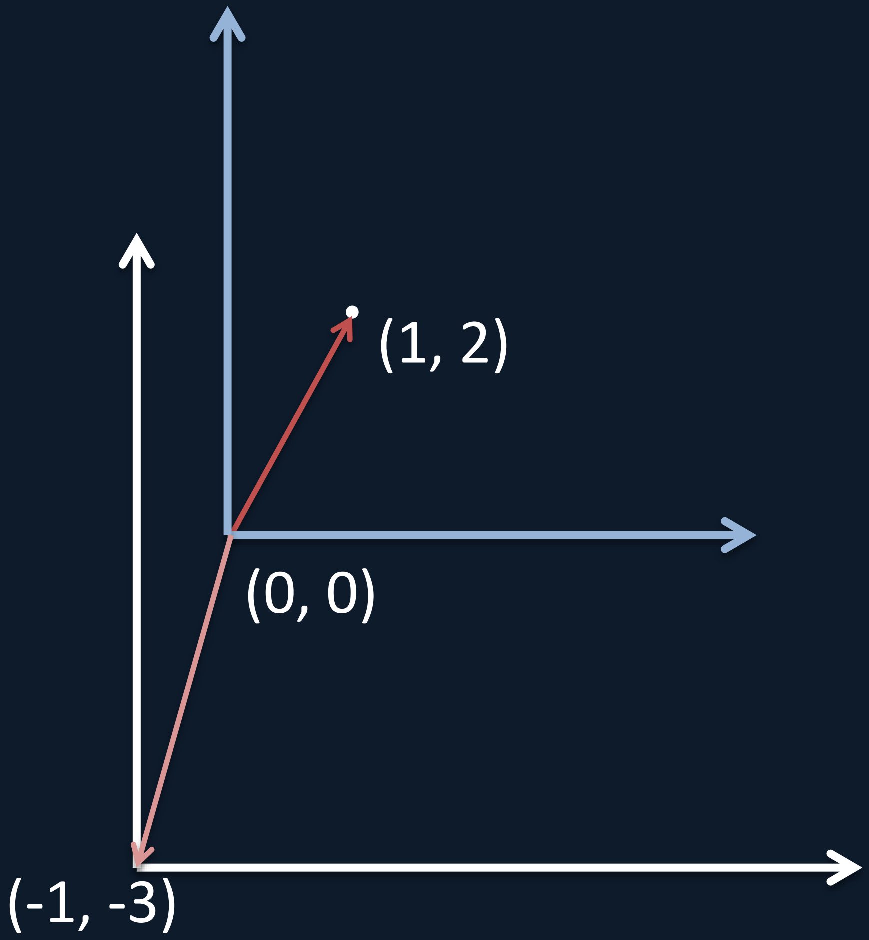
Now we have a second coordinate system, with its own origin. The position of the new system ***relative to the old one*** is the vector (1, 3)

Canberra Institute of Technology

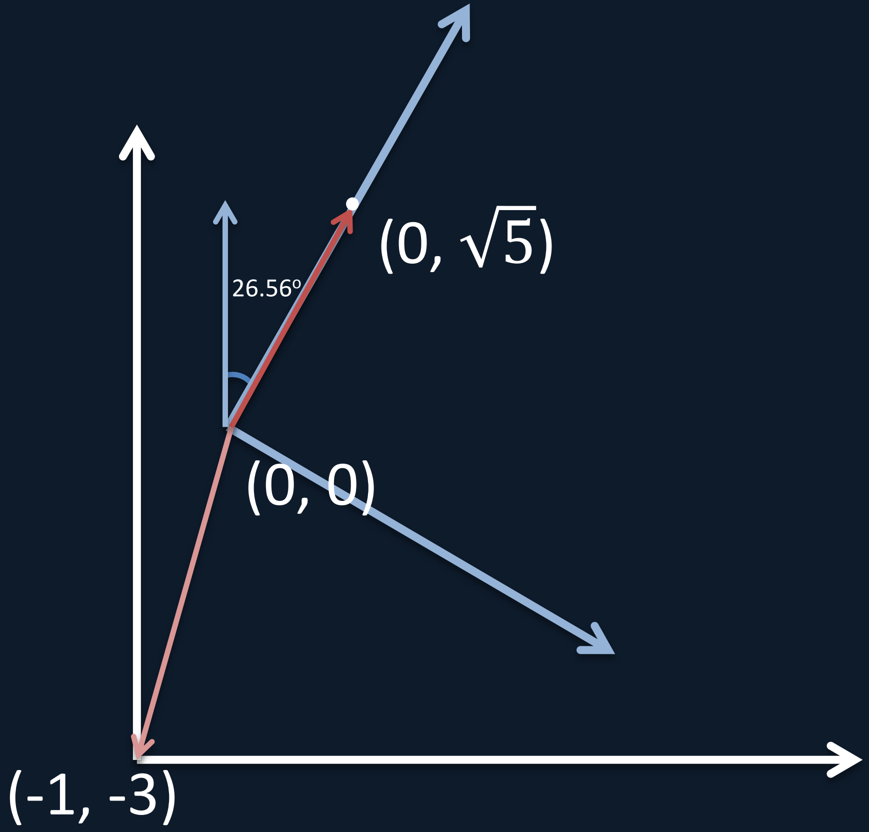
aie SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX



Conversely, the position of the old system ***relative to the new one*** is the vector (-1, -3)



And the position of our point in the new coordinate system is the vector (1, 2).



Now to get really crazy, we could rotate our new coordinate system. Again this rotation is ***<u>relative to the old coordinate system</u>***, our new system has been rotated clockwise 26.56º.

What does this have to do with matrices?

- Transformation matrices are an efficient and compact way to store the information about where one coordinate system is situated relative to another.

- It’s also really easy to use a transformation matrix to transform a vector between coordinate systems.

Review of matrix multiplication

- All the matrices we will be looking at in these slides are square and of equal size.
- In this case the final matrix will be the same size as the two input matrices.
- Each number in the final matrix is the dot product of the matching row in the first matrix and the matching column in the second.

Structure of a Transformation Matrix

- Homogeneous 2D
- 3 x 3 matrix
- $$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

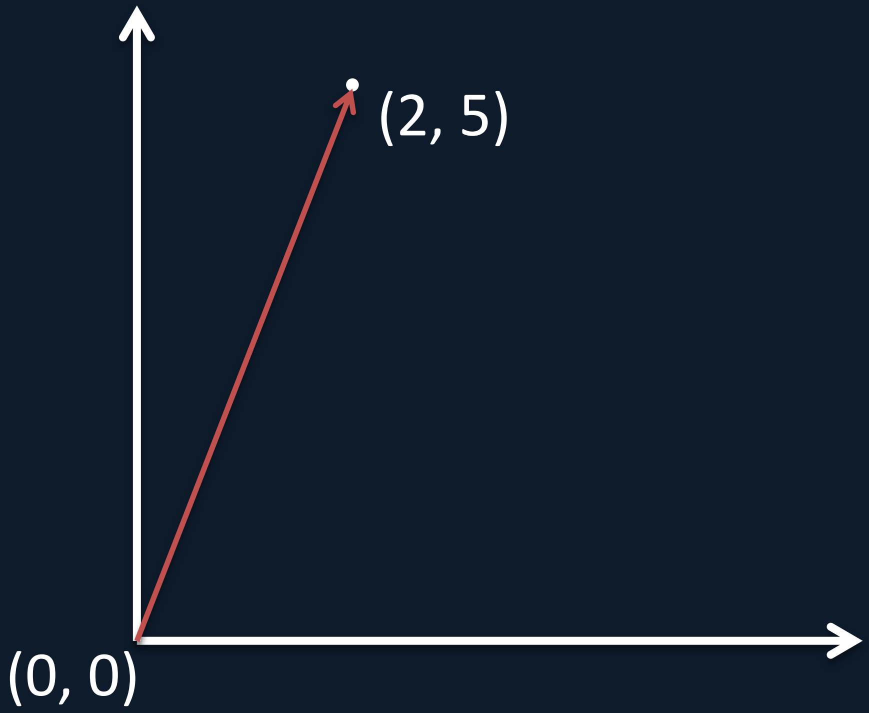
Structure of a Transformation Matrix

- Homogeneous 2D
- 3 x 3 matrix
- $$\begin{array}{cc} \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} & \begin{array}{c}X\\ Y\\ W\end{array} \\ \begin{array}{ccc}\text{X axis} & \text{Y axis} & \text{Translation}\end{array} & \end{array}$$

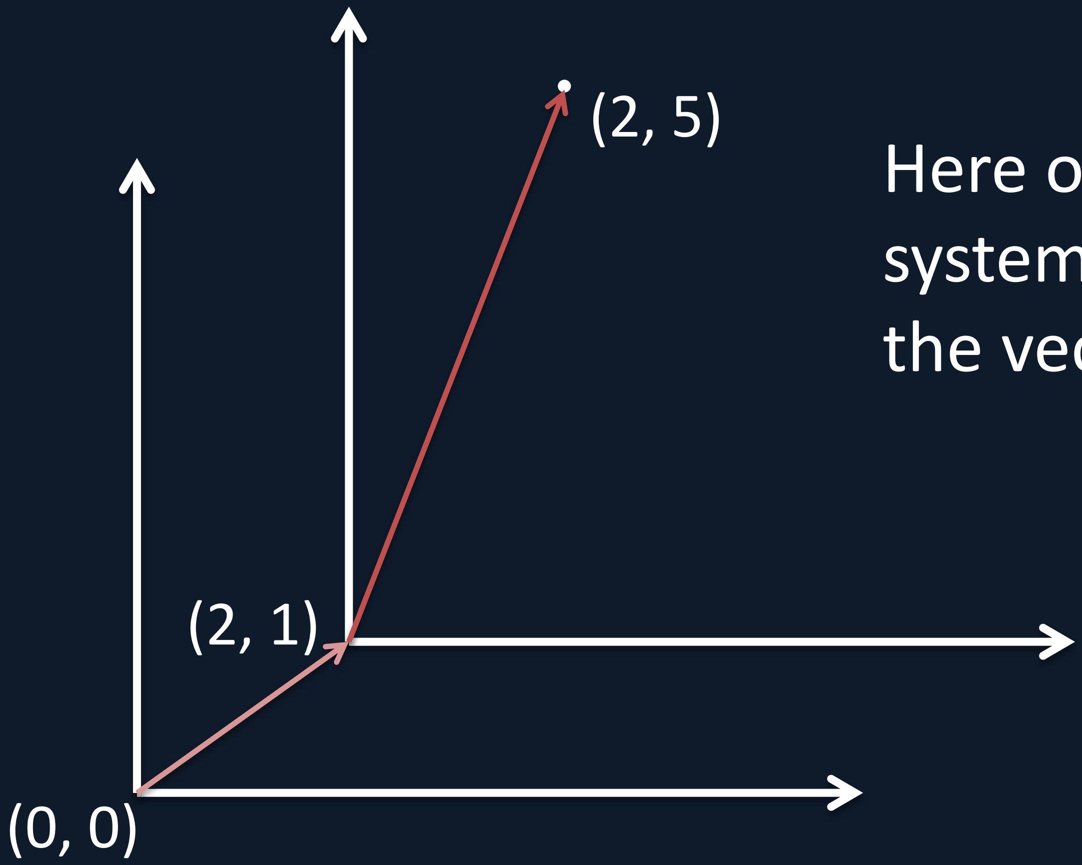
How a transformation works?

- There are 3 main kinds of transformation
  - Translation
  - Rotation
  - Scale

- Lets go back to our coordinate system.



The most basic kind of transformation is translation.



Here our coordinate system is translated by the vector (2, 1)

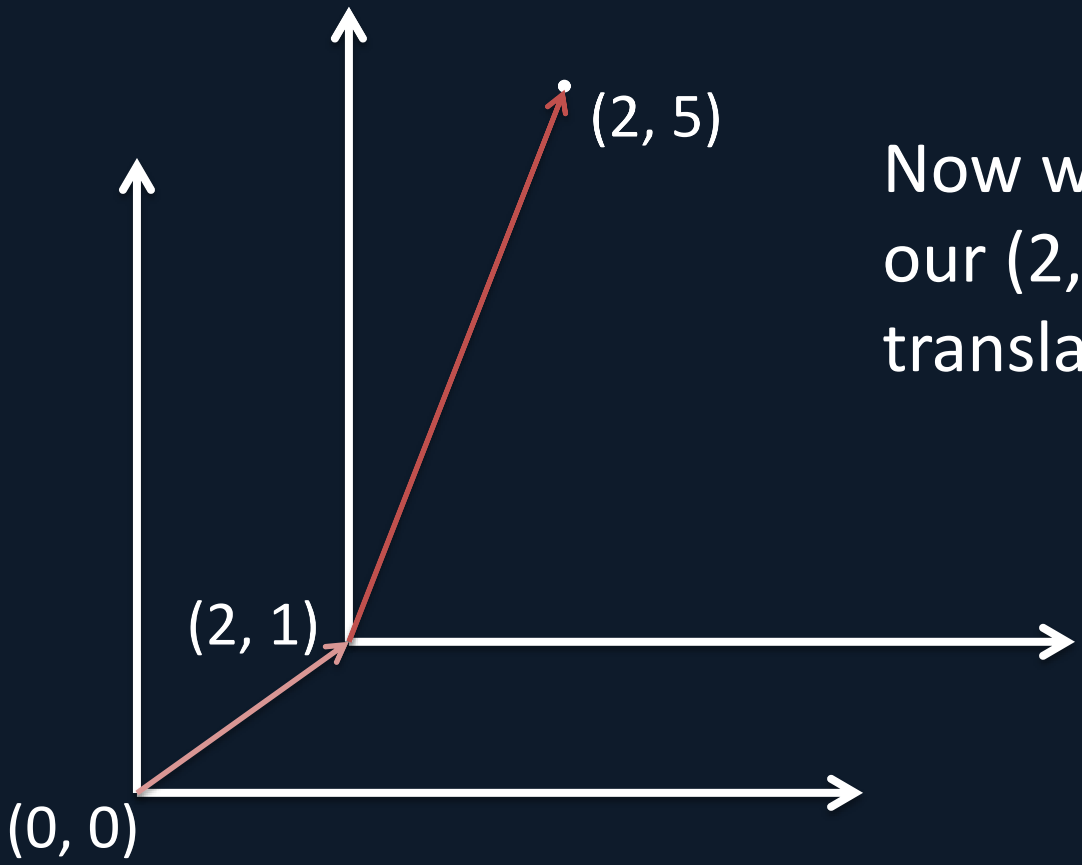
Translation

- $$\begin{array}{ccc|c} \left[\begin{matrix}1 \\ 0 \\ 0\end{matrix}\right. & \begin{matrix}0 \\ 1 \\ 0\end{matrix} & \left.\begin{matrix}0 \\ 0 \\ 1\end{matrix}\right] & \begin{matrix}\text{X} \\ \text{Y} \\ \text{W}\end{matrix} \\ \text{X axis} & \text{Y axis} & \text{Translation} & \end{array}$$
- Looking at our matrix, the right most column is the translation.
- To build a translation matrix we just put the X and Y translation we want into the X and Y rows

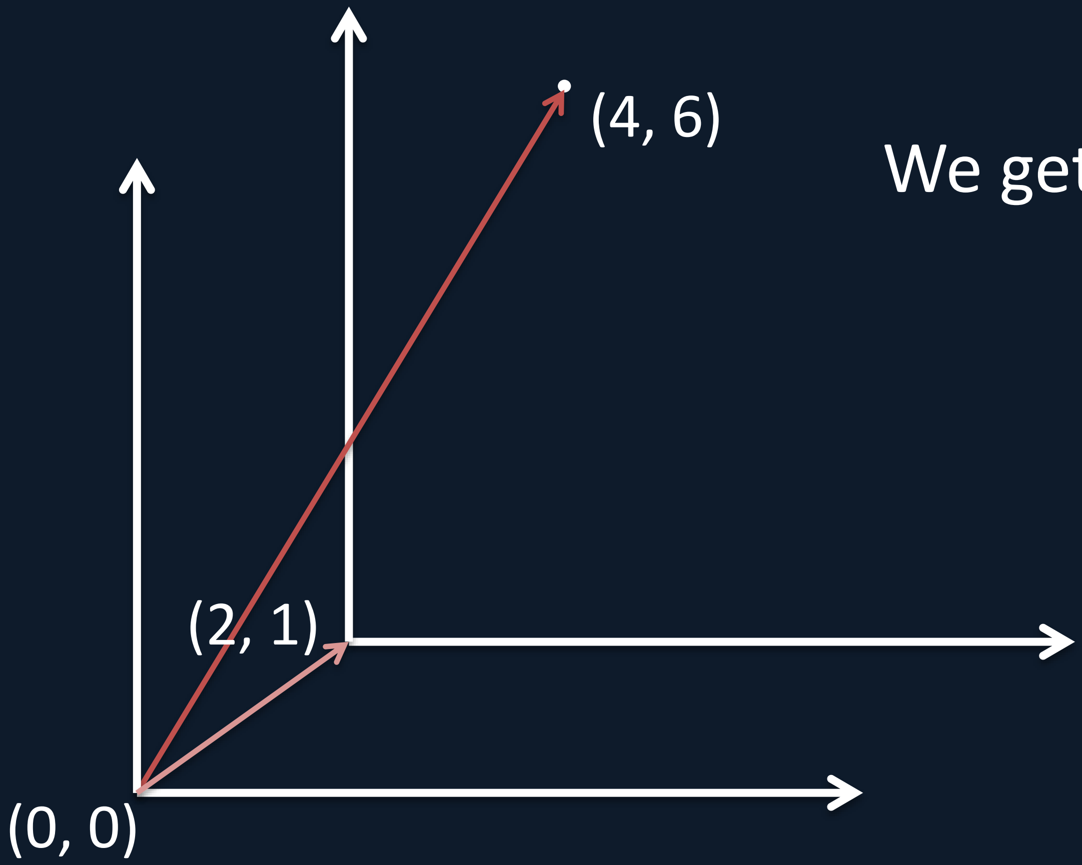
Translation

- $$\begin{array}{cccc} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} & \begin{array}{c} \text{X} \\ \text{Y} \\ \text{W} \end{array} \\ \begin{array}{ccc} \text{X axis} & \text{Y axis} & \text{Translation} \end{array} & \end{array}$$

- Looking at our matrix, the right most column is the translation.
- To build a translation matrix we just put the X and Y translation we want into the X and Y rows



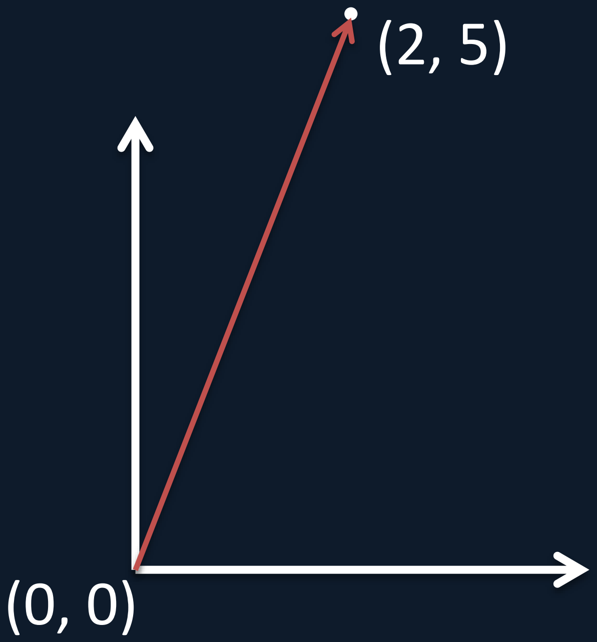
Now when we multiply our (2, 5) vector by our translation matrix…



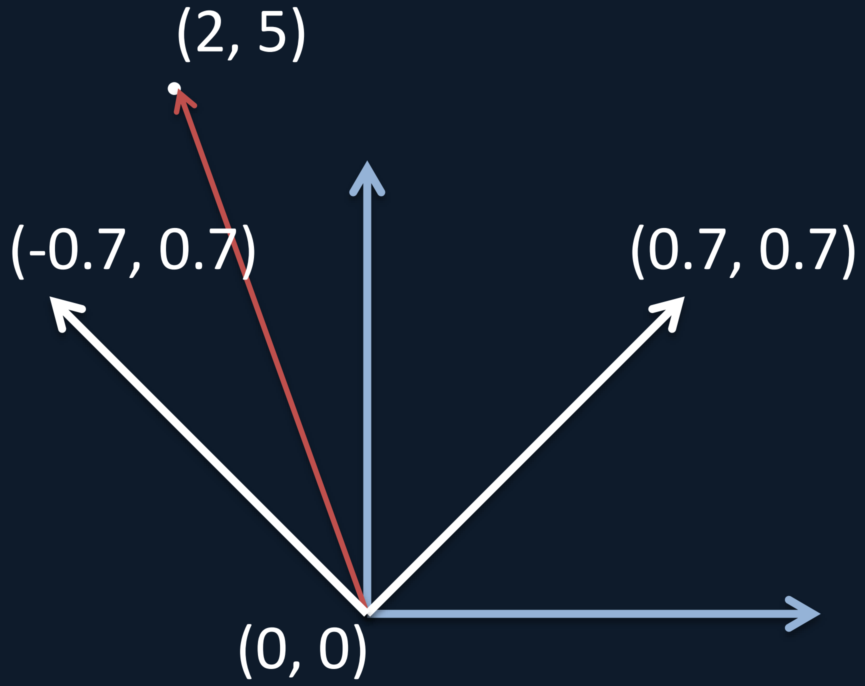
We get the vector (4, 6)

Rotations

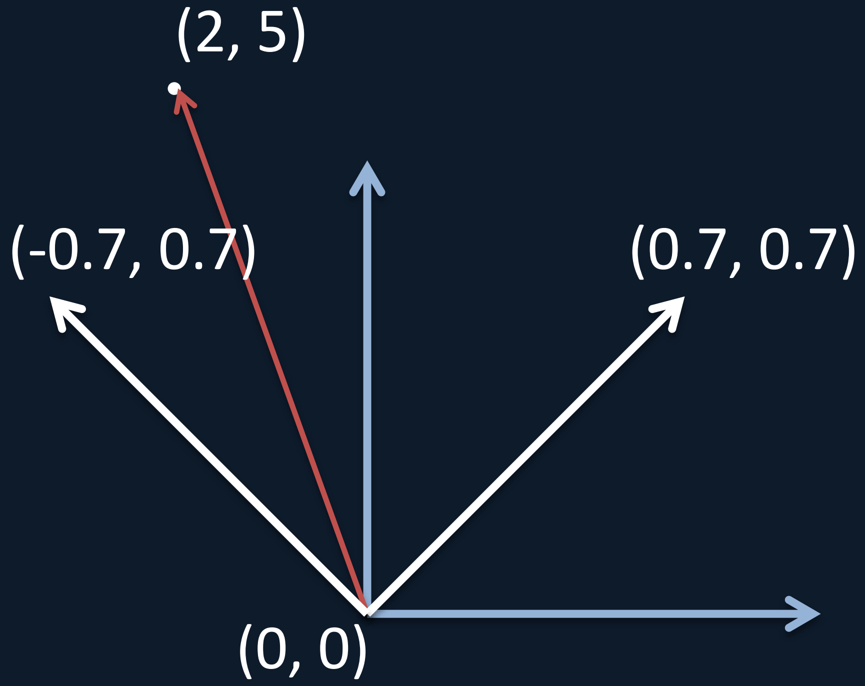
- Rotation matrices rotate vectors around the origin.



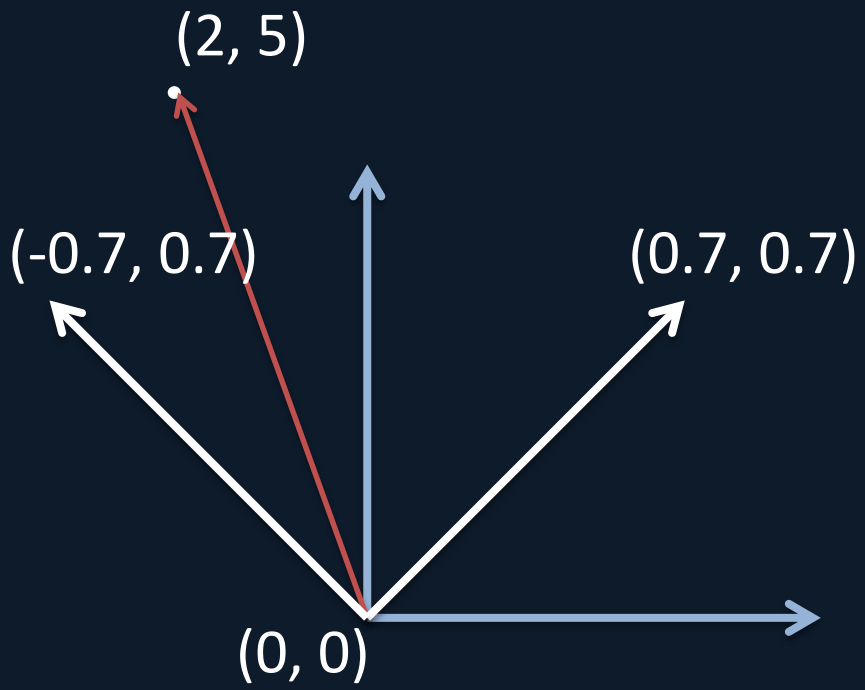
When a coordinate system is by itself, the X axis, *by definition*, points towards (1, 0) and the Y axis towards (0, 1)



But when another coordinate space exists, we can define them to point in directions relative to the X and Y axies of that space



The original axies are now pointing in new directions relative to the new system. The X axis is now pointing towards (0.7, 0.7) and the Y axis now points towards (-0.7, 0.7)



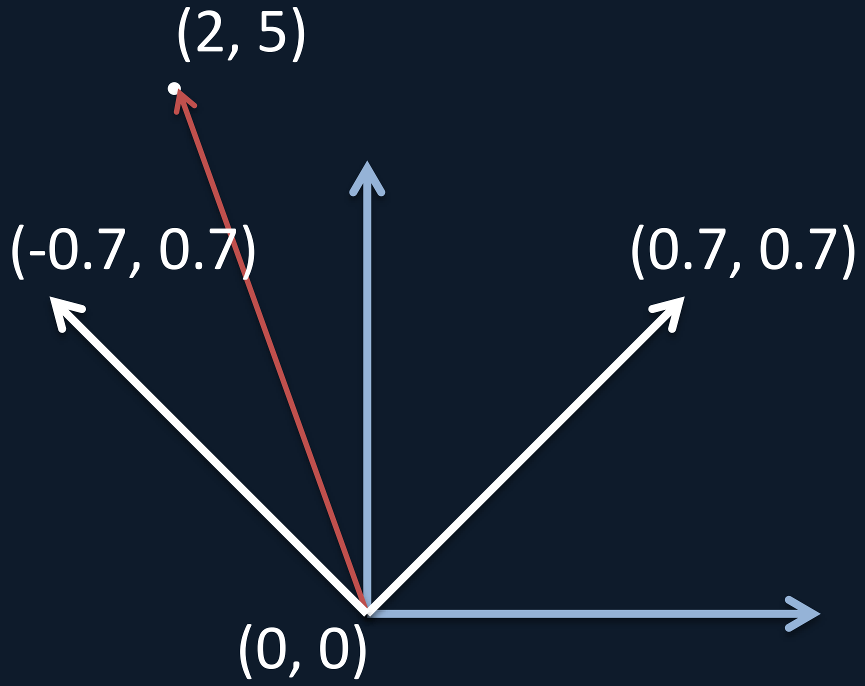
They’ve been rotated courter-clockwise by 45º

Rotation

- $$\begin{array}{ccc|c} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & W \\ \text{X axis} & \text{Y axis} & \text{Translation} & \end{array}$$
- Looking at our matrix, we have a column for both the X and the Y axis.
- To build a rotation matrix we put the new vectors for the X and Y axies into the matrix

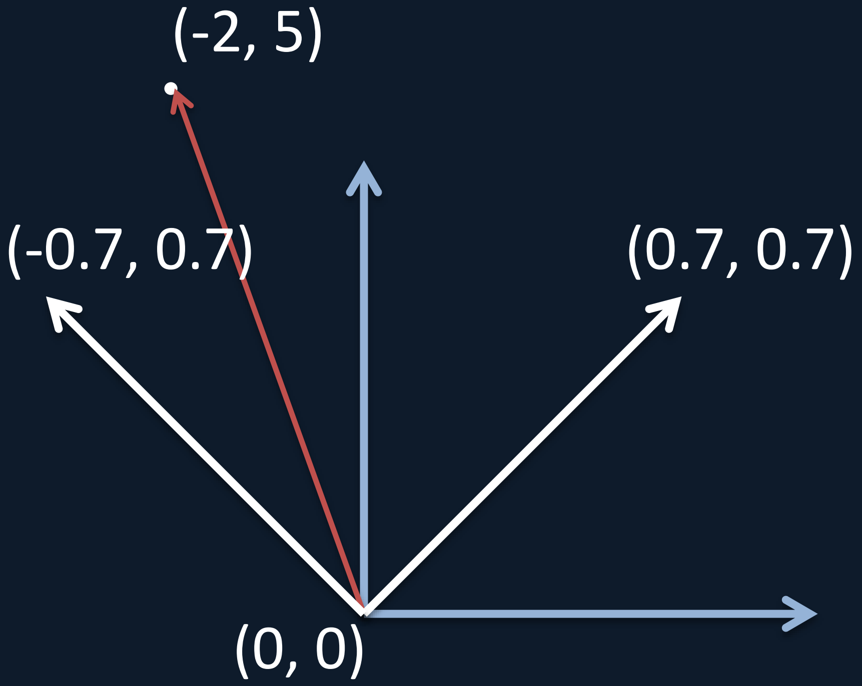
Rotation

- $$\begin{array}{ccc|c} 0.7 & -0.7 & 0 & \text{X} \\ 0.7 & 0.7 & 0 & \text{Y} \\ 0 & 0 & 1 & \text{W} \\ \hline \text{X axis} & \text{Y axis} & \text{Translation} & \end{array}$$
- Looking at our matrix, we have a column for both the X and the Y axis.
- To build a rotation matrix we put the new vectors for the X and Y axies into the matrix



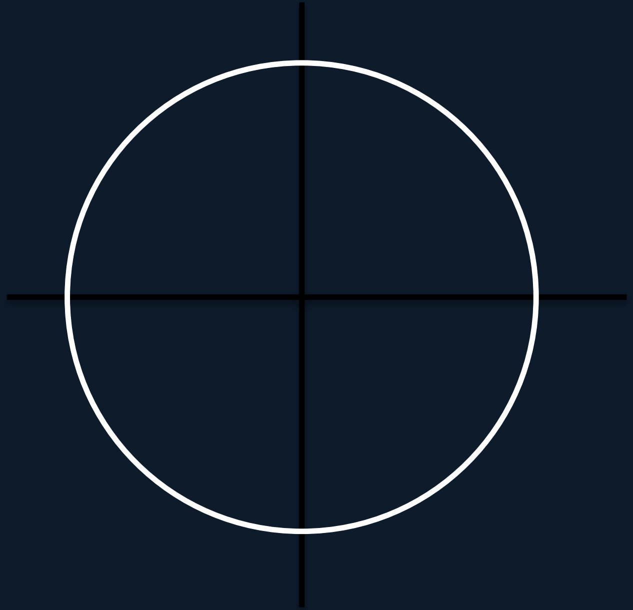
Now when we multiply our (2, 5) vector by our rotation matrix, we get…

The vector (-2, 5)



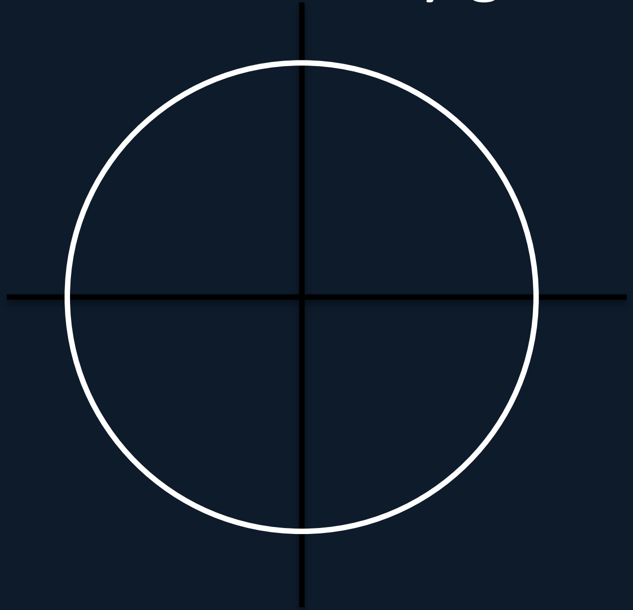
Rotation

- If you think back to basic trigonometry from high school, you should remember the unit circle.



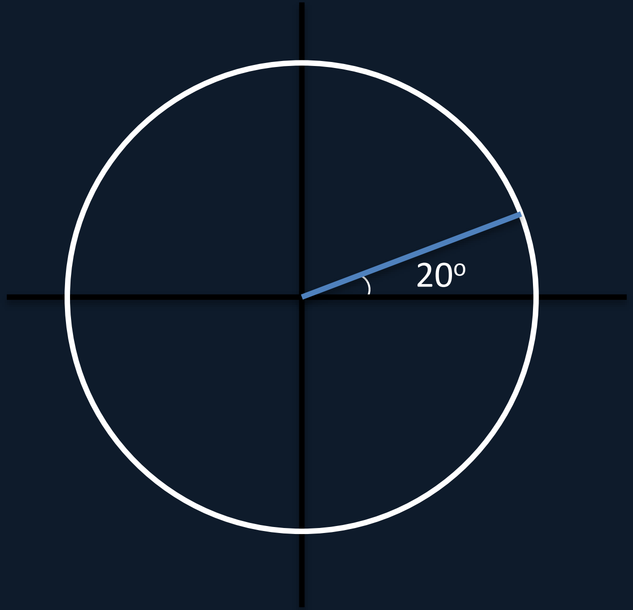
Rotation

- We can use the unit circle to figure out how to build a 2D rotation matrix from any given angle.



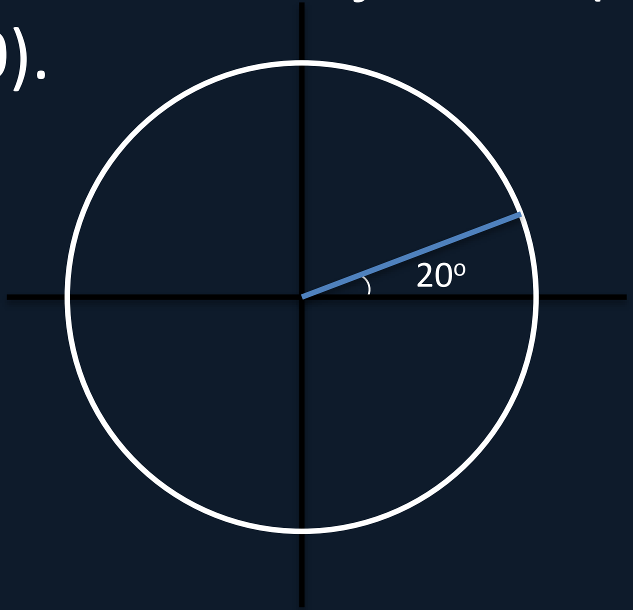
Rotation

- Say we want to rotate the X axis counter-clockwise by 20º.



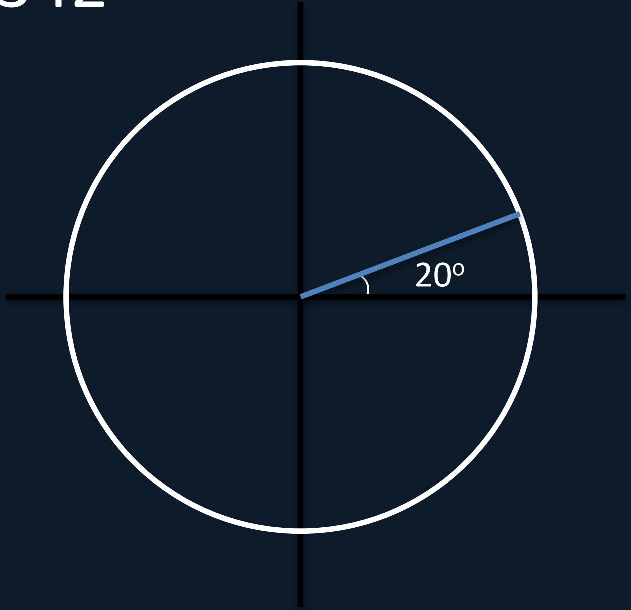
Rotation

- Thanks to the unit circle having a radius of one, the X length of the blue line is just cos(20) and the Y length is sin(20).



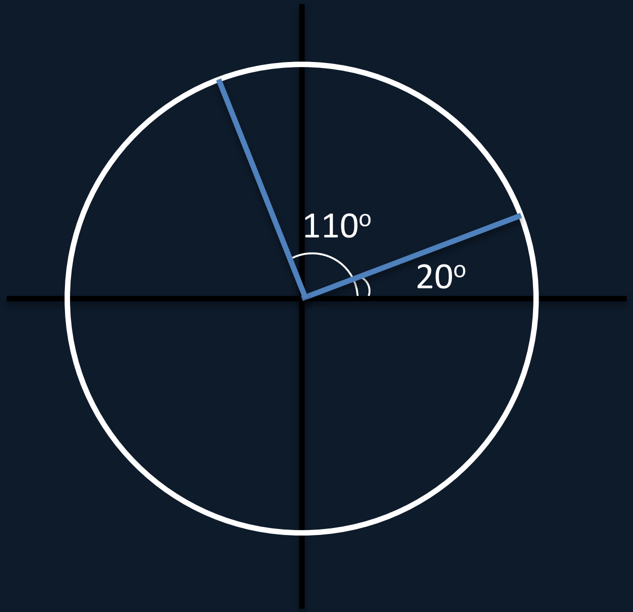
Rotation

- $X = \cos(20) = 0.939$
- $Y = \sin(20) = 0.342$



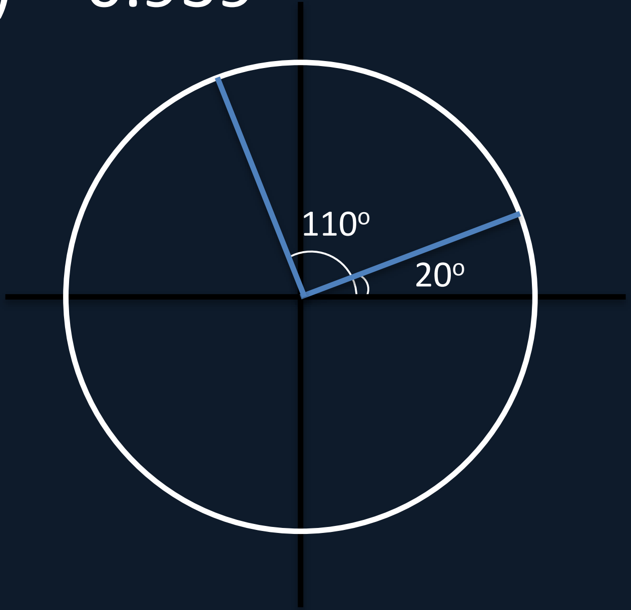
Rotation

- For the Y axis, we need 90º further then the X. So for the Y axis we have…



Rotation

- $X = \cos(20 + 90) = -0.342$
- $Y = \sin(20 + 90) = 0.939$



Rotation

- Our final rotation matrix for 20^o is then:
- $$\begin{bmatrix} 0.939 & -0.342 & 0 \\ 0.342 & 0.939 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation

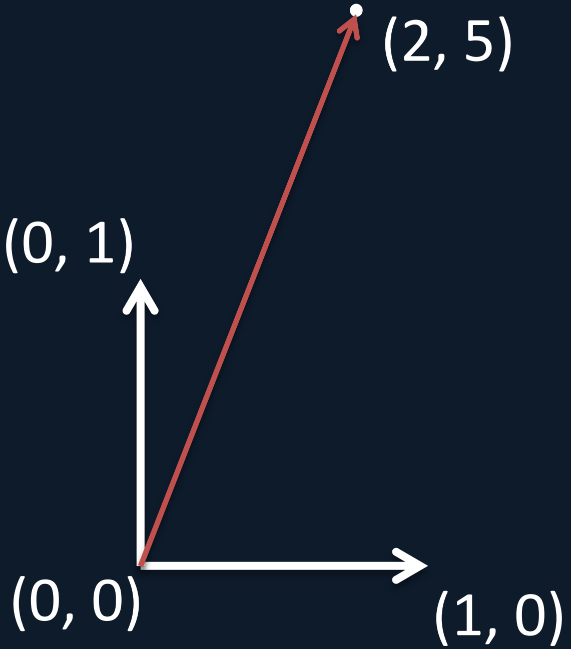
- For calculating a rotation matrix from an arbitrary angle we can just use the formula:
- $$\begin{bmatrix} \cos(\theta) & \cos(\theta + 90) & 0 \\ \sin(\theta) & \sin(\theta + 90) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
- Now if you *really* remember high school trig you should know
- $\cos(\theta+90) = -\sin(\theta)$
- $\sin(\theta+90) = \cos(\theta)$

Rotation

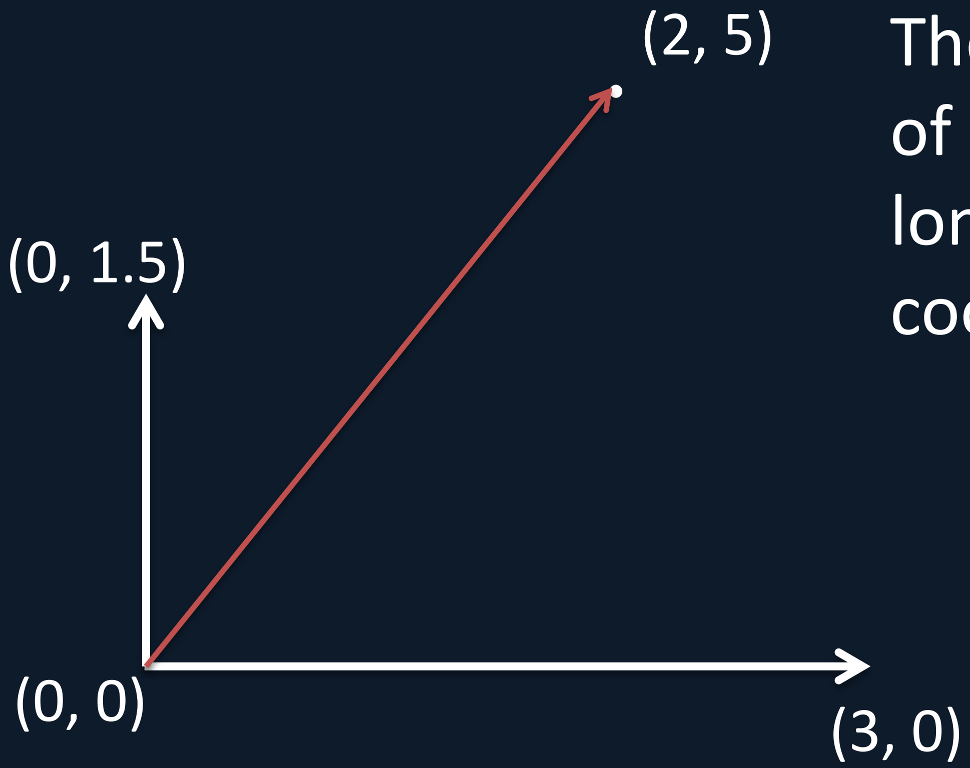
- So the final formula is
- $$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scale

- Using a scale matrix changes the magnitude of a vector.



As we said before, by definition the X axis is (1, 0) and the Y axis is (0, 1)



They both have a length of 1. If we make them longer, we scale the coordinate system.

Scale

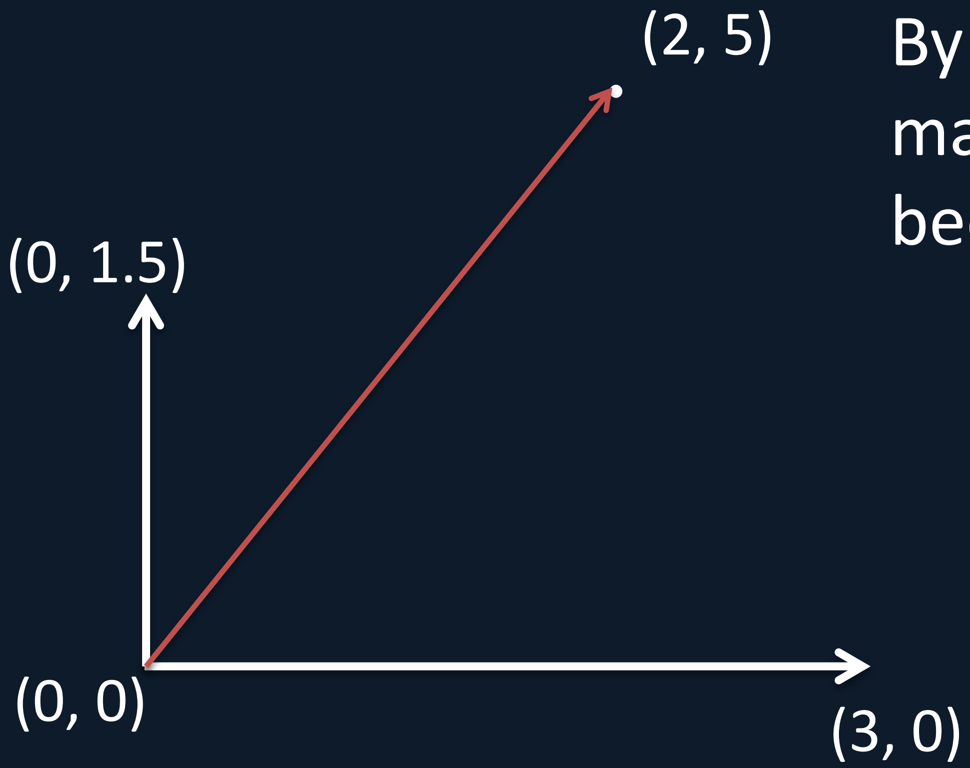
- $$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- To build a scale matrix all we do is multiply the X and the Y vectors by how much we want to scale each axis

Scale

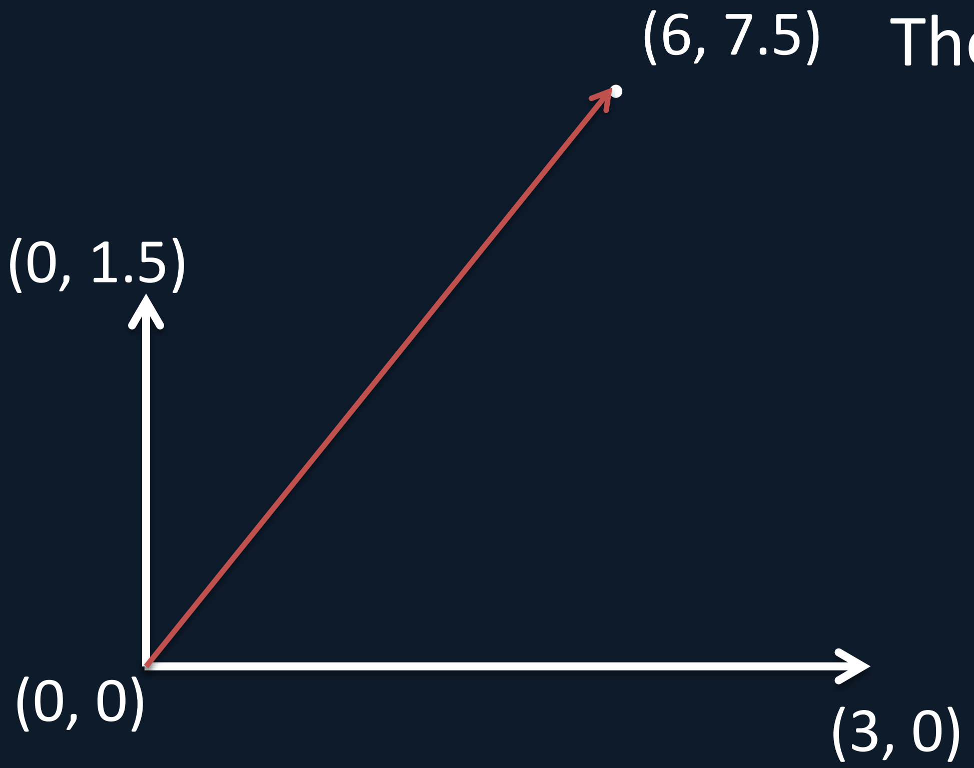
- $$\begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- To build a scale matrix all we do is multiply the X and the Y vectors by how much we want to scale each axis



By multiplying or scale matrix with our vector, it becomes…

The vector (6, 7.5)



Transformation Matrices are Orthogonal

- $$\begin{array}{cccc} 0.7 & -0.7 & 0 & \text{X} \\ 0.7 & 0.7 & 0 & \text{Y} \\ 0 & 0 & 1 & \text{W} \\ \text{X axis} & \text{Y axis} & \text{Translation} & \end{array}$$
- Orthogonal matrices are matrices where each of the column vectors within the matrix are at right angels to each other.
- It should be clear that any matrix representing the

  axies of a coordinate system will be orthogonal.

Concatenating Matrices

- It’s really simple to perform multiple operations such as both rotating and scaling, or translating and rotating.
- Simply multiply the matrices together.
- The resulting matrix, when multiplied with the vector will transform it by all of the operations.

Concatenating Matrices

- Its important to keep track of what order you multiply the matrices.
- Rotating around the origin by 65º, then translating by (7, -10) will lead to a very different final position then translating, then rotating the translated vector around the origin.
- As a general rule the order should be:
- Scale * Rotation * Translation

Is any of this useful, anyway?

- YES
- In most games, transformation matrices are the main way of storing the position/scale/rotation data for all the objects in the world.

Instancing

- Lets say you have a complex set of points representing the mesh for an object and you want to have multiple copies of the mesh in the world.
- Instead of keeping track of multiple copies of the points all moved around the world
- We keep one copy of the points stored around a local origin and a then a Matrix representing the orientation and translation of each copy.

Parenting

- Instead of using a static world origin as the coordinate system you are transforming to you can use the one defined by another objects transform.
- This is called parenting and is done by multiplying the two full transformation matrices together.
- The affect is that the second transform will move with the first as though it is stuck to it.

Cameras

- If you store a matrix representing the position and rotation of the camera, by multiplying the objects you want to render by the inverse of that matrix, you can treat the camera as the origin.

- This is very helpful when rendering, when you want to treat the upper left corner of the screen as the origin.

Conclusion

- We’ve covered the basics of matrix transformations.
- Converting from one coordinate system to another
- Translating, rotating, scaling
- What these transforms are actually used for.