Game Physics

GPR350, Fall 2019
Daniel S. Buckstein

Math Review
Week 1

License

Math for Games Review

- Review of vectors and points
- Review of matrices
- Transformations
- Some new and exciting stuff

Vectors & Points

- A ***vector*** is a quantity with a *direction* and a *magnitude* (length)
- Most general form:

$$\vec{v} = (v_0, v_1, v_2, \dots, v_n)$$

where $v_0, v_1, v_2, \dots, v_n$ are scalars describing shift along each axis

- 2D, 3D, 4D… *n*-dimensional vectors, all have the same behaviours and functions

Vectors & Points

- Different ways to express a vector

- 2D vectors usually generalized like this:

$$\vec{v} = (x, y)$$

This is called an *ordered pair*, where x and y are scalars representing the shifts along their respective axes

Vectors & Points

- Different ways to express a vector

- 3D vectors usually generalized like this:

$$\vec{v} = (x, y, z)$$

This is called an *ordered triplet*, where x, y and z are scalars representing the shifts along their respective axes

Vectors & Points

- Different ways to express a vector

- Can be written as a *column vector*, which is the same as a 3x1 matrix:

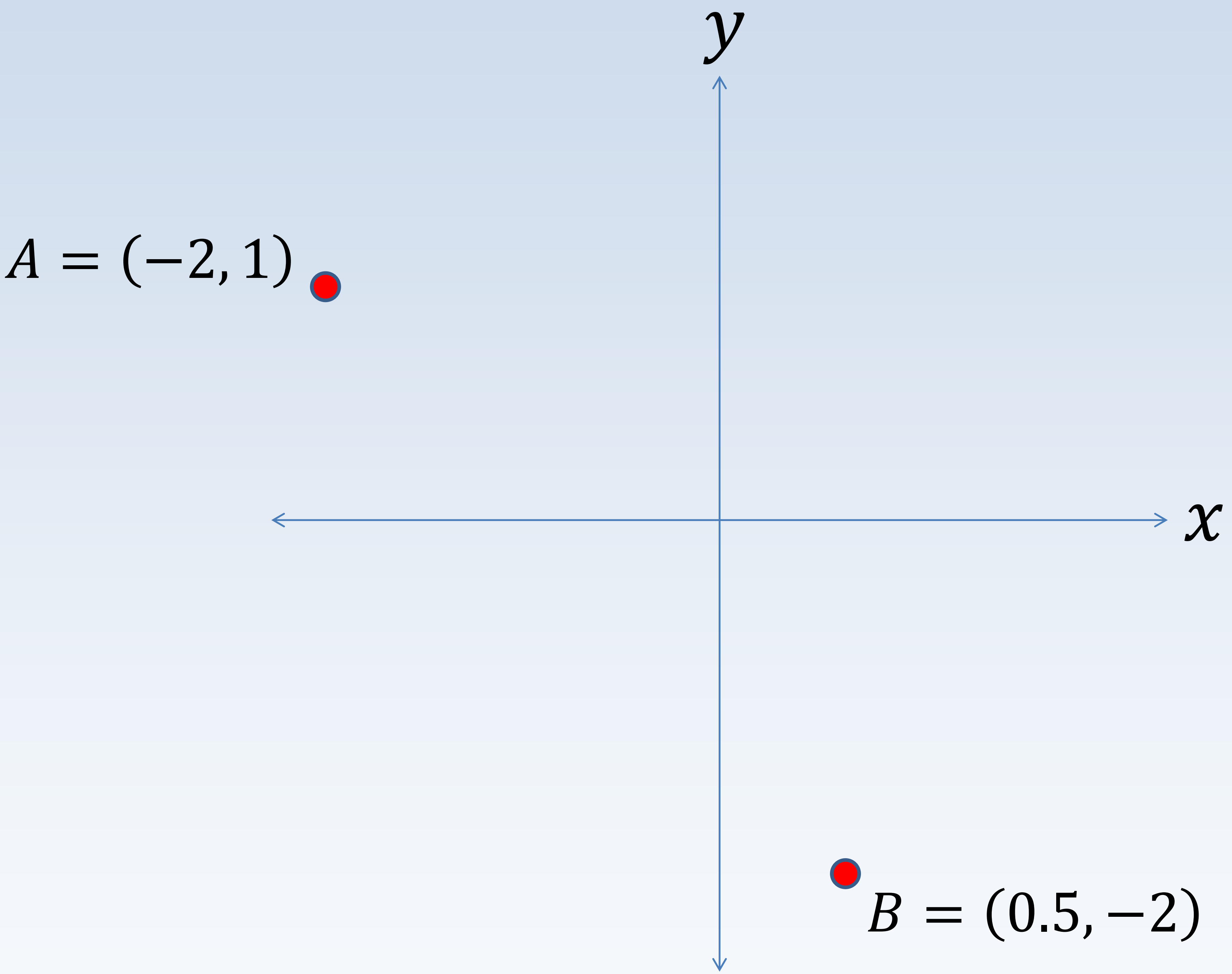
$$\vec{v} = [x, y, z]^T = [x \;\; y \;\; z]^T = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Vectors & Points

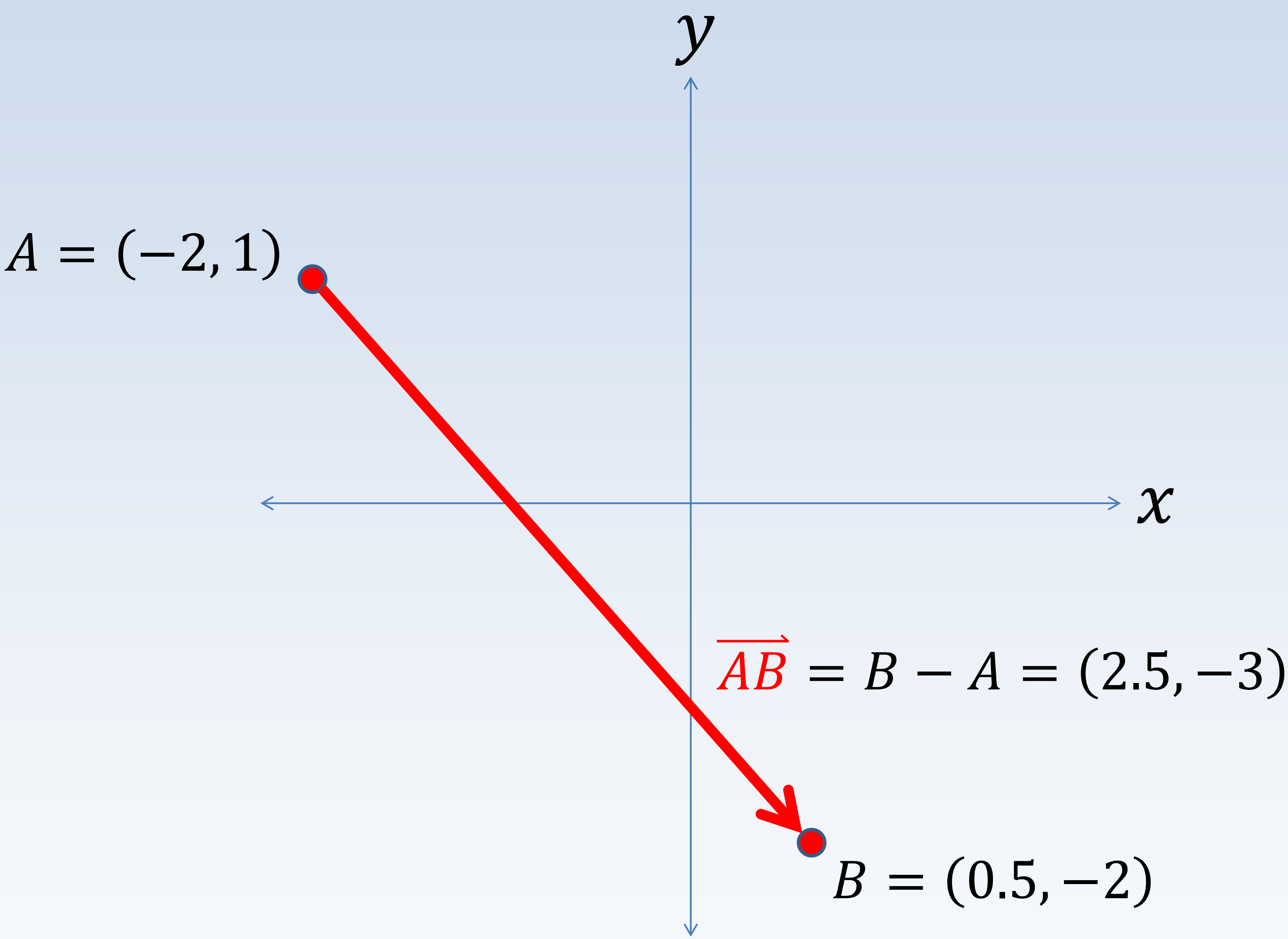
- A ***point*** is a *location in space*
- A vector represents a *displacement in space*, or the difference between two points

- Analogy: a specific *moment* or *instance* in time would be a *point*, and some *duration* or *time measurement* would be a *vector*

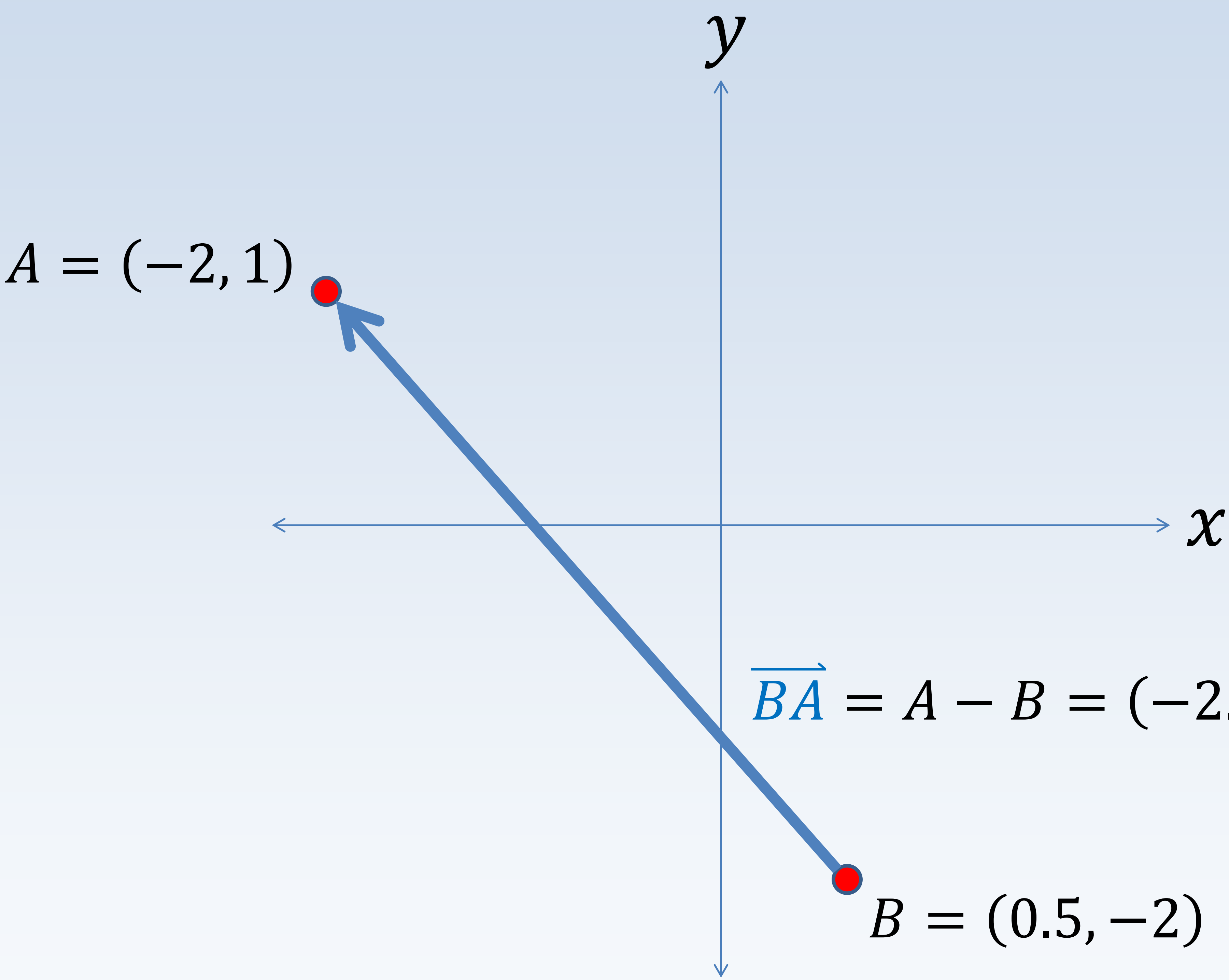
Vectors & Points



Vectors & Points



Vectors & Points



$$\overrightarrow{BA} = A - B = (-2.5, 3) = -\overrightarrow{AB}$$

Vector Operations

- Addition and subtraction
- Perform operation on each component of the vector:

If $v_0 = (x_0, y_0, z_0)$

and $v_1 = (x_1, y_1, z_1)$

then

$$v_0 + v_1 = (x_0 + x_1, y_0 + y_1, z_0 + z_1)$$
$$v_0 - v_1 = (x_0 - x_1, y_0 - y_1, z_0 - z_1)$$

Vector Operations

- Scalar multiplication
- Multiply each component in the vector by scalar:

If $v = (x, y, z)$,

then $sv = (sx, sy, sz)$

Vector Operations

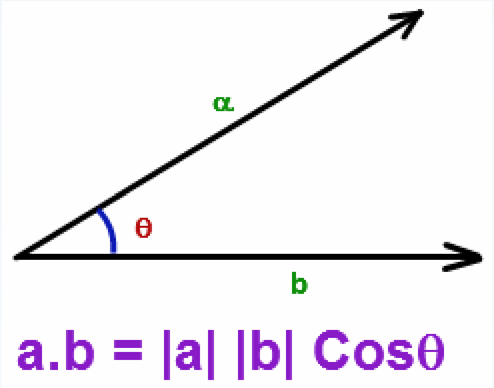
- Dot product
- Represents the relative alignment of two vectors
- Can be used to determine the angle separating the vectors
- To take the dot product of two vectors, multiply the respective components, and add the products together

Vector Operations

- Dot product (result is a ***scalar***)

$$v_0 = (x_0, y_0, z_0)$$
$$v_1 = (x_1, y_1, z_1)$$

$$v_0 \cdot v_1 = x_0 x_1 + y_0 y_1 + z_0 z_1$$



Vector Operations

- Cross product
- Used to find a vector perpendicular to both input vectors
- Careful: vectors that are perfectly aligned or opposite will return a cross product of $\vec{\mathbf{0}}$
- Useful mnemonic for solving cross product:

xyzzy: “***x easy***” – describes the variable order

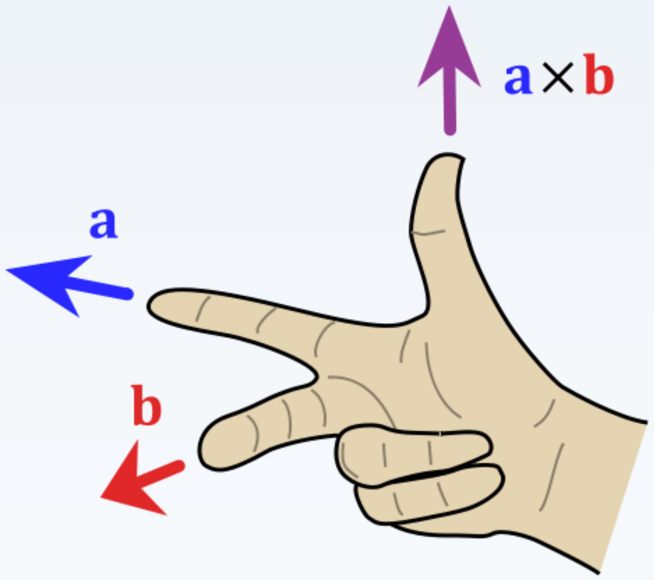
Vector Operations

- Cross product (result is a ***vector***)

$$v_0 = (x_0, y_0, z_0)$$
$$v_1 = (x_1, y_1, z_1)$$

$$v_0 \times v_1 = (y_0 z_1 - z_0 y_1,$$
$$z_0 x_1 - x_0 z_1,$$
$$x_0 y_1 - y_0 x_1)$$

x=yzzy



Vector Operations

- Magnitude
- Describes the *length* of the vector or the *distance* covered by the vector
- Derived from Pythagorean theorem:

$$v = (x, y, z)$$
$$\|v\| = \sqrt{x^2 + y^2 + z^2}$$

Vector Operations

- Magnitude
- Pro tip: does the formula for *magnitude squared* look familiar???

$$\|v\|^2 = \left(\sqrt{x^2 + y^2 + z^2}\right)^2$$

$$\|v\|^2 = x^2 + y^2 + z^2$$

- How about this?

$$\|v\|^2 = x \cdot x + y \cdot y + z \cdot z$$

Vector Operations

- Magnitude
- Pro tip: *magnitude* can be expressed as a function of the vector's dot product with itself!!!

(isn't math neat?!)

$$\|v\|^2 = v \cdot v$$
$$\|v\| = \sqrt{v \cdot v}$$

Vector Operations

- Normalize
- A ***normalized*** vector has a magnitude of **1**
- Also known as ***direction vector*** or ***unit vector***
- To normalize a vector or find its direction, divide the vector by its magnitude:

$$v = (x, y, z)$$

$$\hat{v} = \frac{v}{\|v\|}$$

Vector Operations

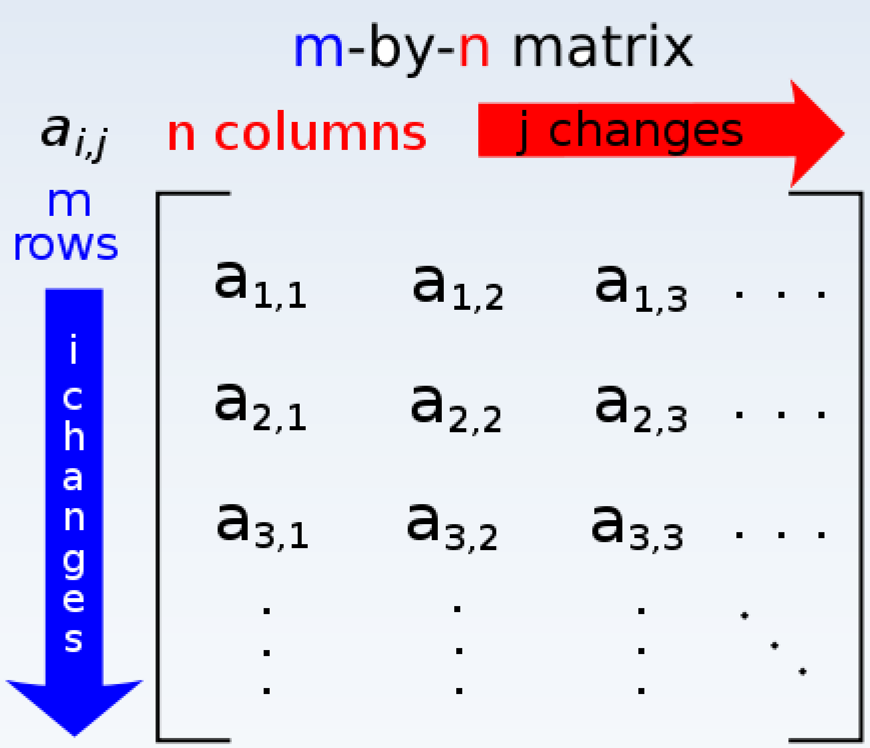
- Projection
- Handy function for figuring out what a vector looks like projected onto another vector:

$$\mathrm{proj}_b a = \frac{a \cdot b}{||b||^2} b$$

- Used in graphics, collision detection, hull construction…

Matrices

- A ***matrix*** is a rectangular array of values
- ***m-by-n*** matrix has *m* rows and *n* columns
- Single element of matrix denoted as $a_{i,j}$



Matrices

- 3x3 matrices can be used to represent *rotations in 3D*

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Most relevant to 2D sprite-based games ;)

Matrix Operations

- Concatenation
- Also known as ***matrix multiplication***
- ***Non-commutative***: written order matters!

$$AB \neq BA$$

- ***Associative***: chaining more than 2 matrices, does not matter which concatenation happens first

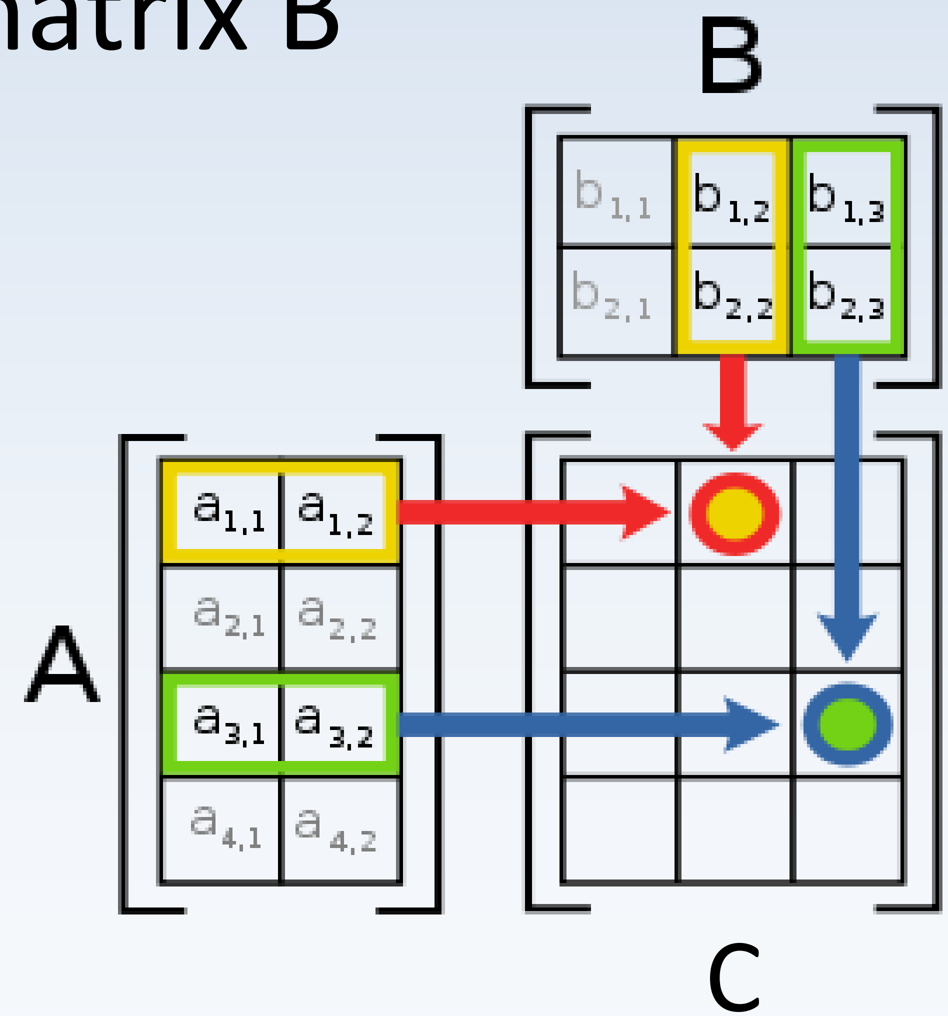
$$(AB)C = A(BC)$$

Matrix Operations

- Concatenation
- Number of ***columns*** in the ***left matrix*** must equal the number of ***rows*** in the ***right matrix***
- Resultant vector will have the number of rows of the left and the number of columns of the right matrix
- E.g. $A_{4\times 2}\, B_{2\times 3} = C_{4\times 3}$

Matrix Operations

- Concatenation
- Graphical method of finding the product of matrix A times matrix B
- (assuming they are compatible)



$$C = AB$$

Matrix Operations

- Concatenation
- When describing *rotations*, the first rotation is written on the *right*

- E.g. $R = R_0 R_1$

- In this example, the rotation R_1 will occur first

Matrix Operations

- Transpose
- Flip the values along the diagonal

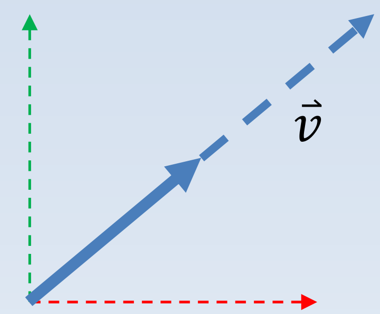
$$A = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}, \qquad A^T = \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

- PRO TIP: For *rotation matrices*, the transpose is also the inverse!!! $R^{-1} = R^T$

Transformations

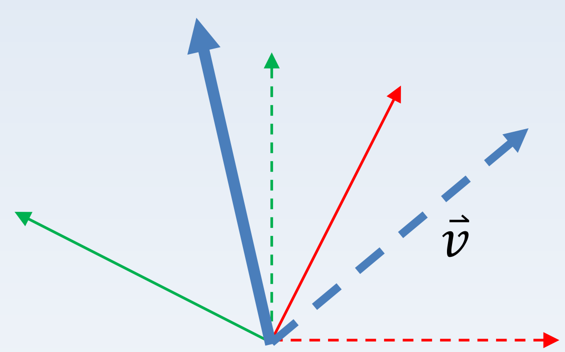
- A vector can be mathematically *transformed*:
- *Scale*: change size

$$\vec{v}_{\text{scaled}} = S\,\vec{v}$$



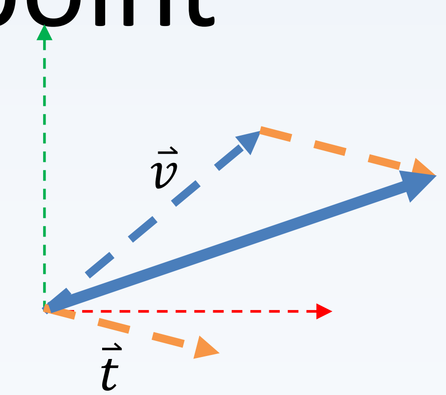
- *Rotation*: change direction

$$\vec{v}_{\text{rotated}} = R\,\vec{v}$$



- *Translation*: change position/endpoint

$$\vec{v}_{\text{translated}} = \vec{v} + \vec{t}$$



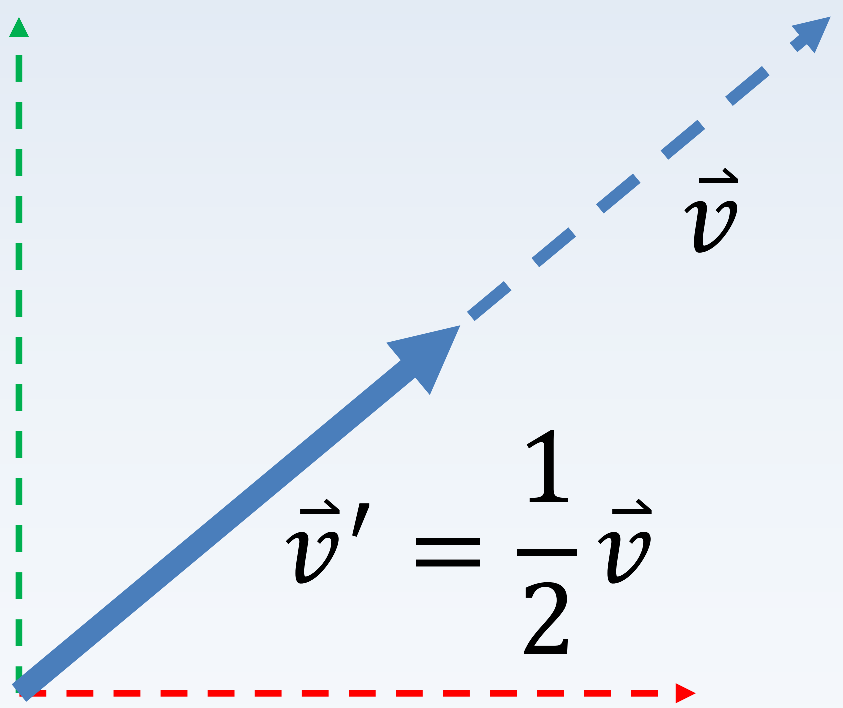
Transformations

- A vector can be mathematically *transformed*:
- *Scale*: change magnitude

$$\vec{v}_{\text{scaled}} = S\,\vec{v}$$

'S' can be a *scalar*
("uniform scale")
e.g. 2, 0.5, ½, 3.14159

or a *diagonal matrix*
("non-uniform scale")
e.g. $\begin{bmatrix} 2.5 & 0 \\ 0 & -1 \end{bmatrix}$

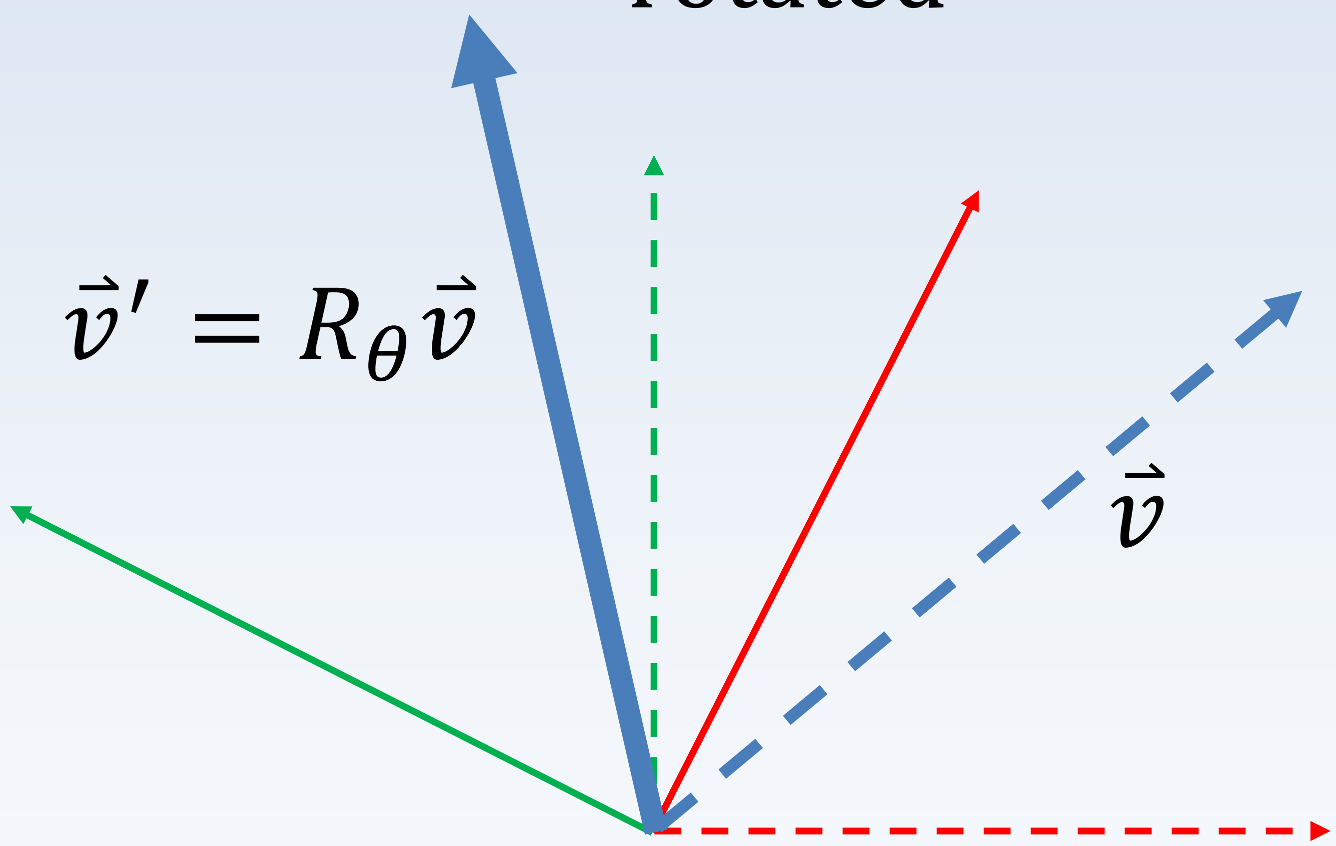


Transformations

- A vector can be mathematically *transformed*:
- *Rotation*: change direction

$$\vec{v}_{\text{rotated}} = R\,\vec{v}$$

$$\vec{v}\,' = R_\theta \vec{v}$$



Pro tip: positive values for θ are *counter-clockwise*

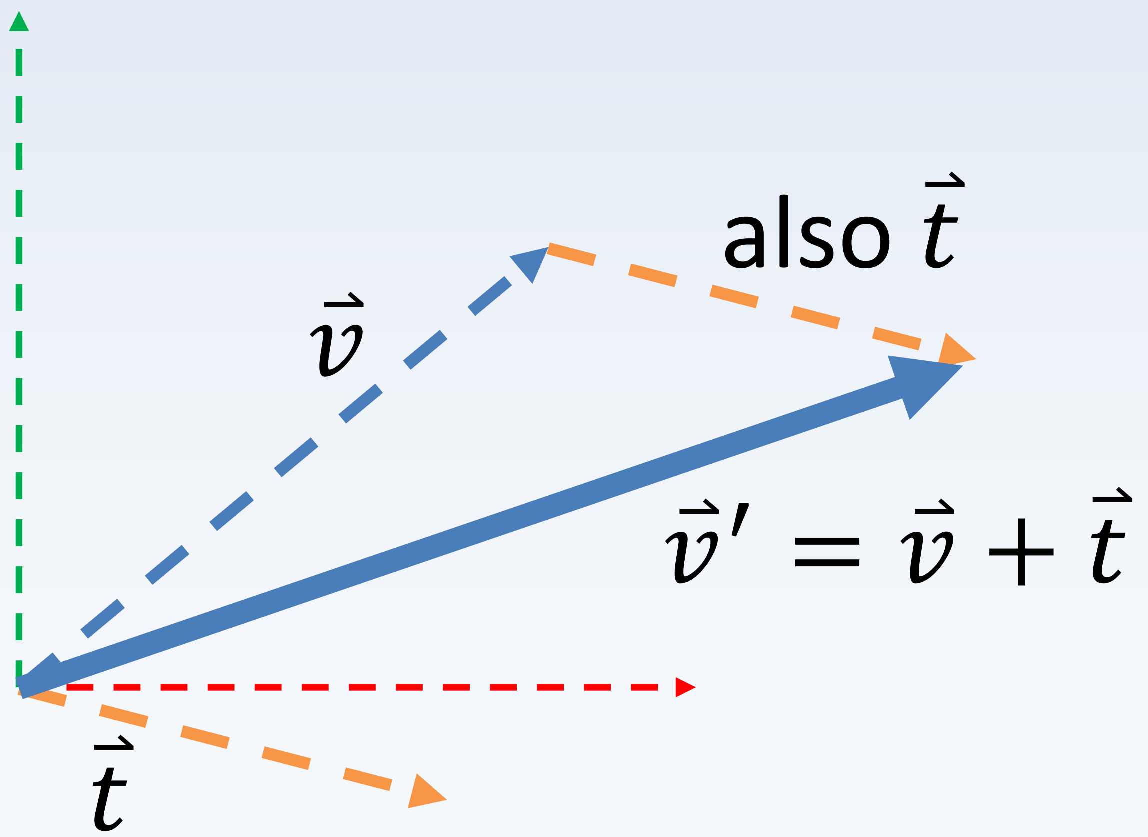
Transformations

- A vector can be mathematically *transformed*:
- *Translation*: change position/endpoint

$$\vec{v}_{\text{translated}} = \vec{v} + \vec{t}$$

Pro tip: translation vector $\vec{t}$ is *just another vector…*

…end-to-end gives it meaning!



Transformations

- When applying a scale and/or rotation, *and* a translation to a vector, we use this formula:

$$\vec{v}' = R\,S\,\vec{v} + \vec{t}$$

- Ultimately applies a scale, *then* a rotation to a vector and *then* adds a translation…
- Why is it in this order???

Transformations

- ***Homogeneous Transformations:*** a compact format combining scale, rotate, & translate
- Transform as a 4x4 matrix:

$$T_{4\times4} = \begin{bmatrix} \begin{bmatrix} \\ R_{3\times3} \\ \\ \end{bmatrix} & \begin{bmatrix} \\ \vec{t}_{3\times1} \\ \\ \end{bmatrix} \\ \begin{bmatrix} \vec{\mathbf{0}}_{1\times3} \end{bmatrix} & \begin{bmatrix} 1_{1\times1} \end{bmatrix} \end{bmatrix}$$

Transformations

- Can be used to perform the same transformations as the formula:

$$\vec{v}' = R\,\vec{v} + \vec{t}$$

Transformed 3D vector as output/result:
$\vec{v}' = (x', y', z')$
$= [x' \;\; y' \;\; z']^T$

3D vector as input:
$\vec{v} = (x, y, z) = [x \;\; y \;\; z]^T$

This operation is OpenGL's lifeblood.

$$\begin{bmatrix} \vec{v}' \\ 1 \end{bmatrix} = T_{4\times4} \begin{bmatrix} \vec{v} \\ 1 \end{bmatrix} = \begin{bmatrix} R_{3\times3} & \vec{t}_{3\times1} \\ \vec{\mathbf{0}}_{1\times3} & 1_{1\times1} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformations

- Transformation inverse:
- Inverse as a 4x4 matrix:
- This is also a valid transformation on its own!

$$T^{-1} = \begin{bmatrix} R^{-1} & -R^{-1}\,\vec{t} \\ \vec{0} & 1 \end{bmatrix}$$

Transformations

- H-transforms are ***matrices***!!! Concatenation, commutativity and associativity rules apply!!!

$$T_0 T_1 \neq \mathrm{T}_1 \mathrm{T}_0$$

$$T_2(T_1 T_0) = (T_2 T_1) T_0$$

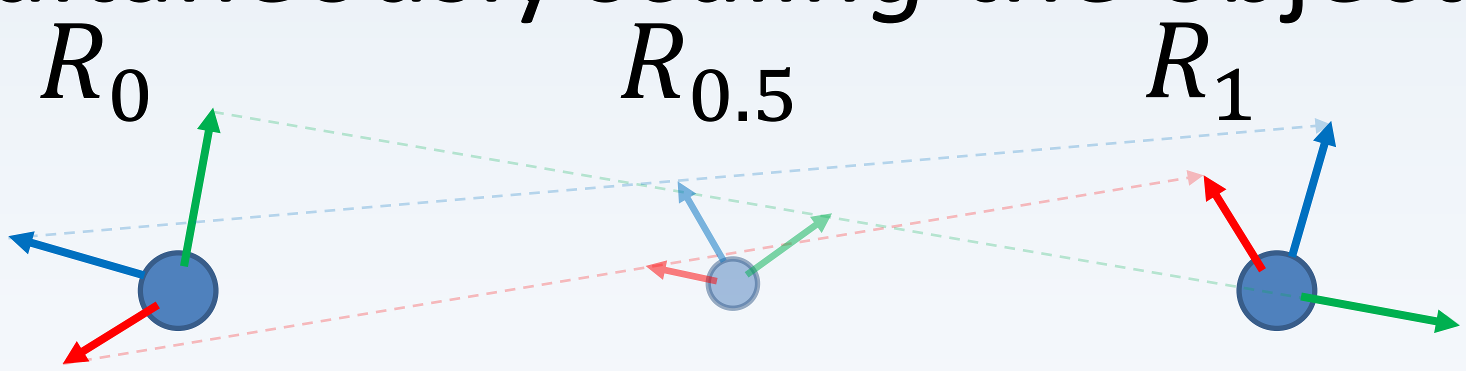
- Multiplying ***any*** transformation by its own inverse yields the identity transform:

$$T T^{-1} = T^{-1} T = I$$

$$R R^{-1} = R^{-1} R = I$$

Transformations

- Problem: transformation matrices do not *animate* very nicely ☹
- The translation part is just a vector, so it can *interpolate* easily (we'll get to that)
- Interpolating a rotation matrix has the effect of simultaneously *scaling* the object!!!



$$R_t = \mathrm{lerp}(R_0, R_1, t)$$

Transformations

- Quaternions: not as scary as they sound
  - (your faithful instructor is good with them… scary good)
    - (he can see in 4 dimensions)
      - (he's actually from the 5th dimension)
- We'll be dealing with these later
- They share many properties with vectors…
- …including physical ones!

The end.

- Questions? Comments? Concerns?