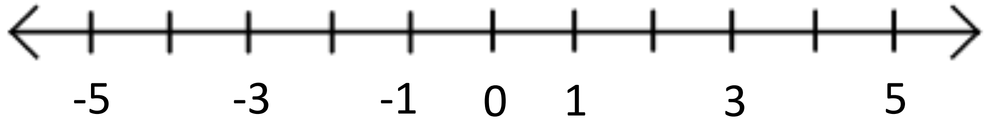
Vectors

Let’s start smaller…

- Scalars
  - One dimensional values
  - Can be real or complex

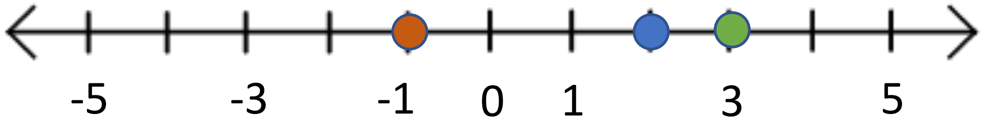


Let’s start smaller…

- Scalars
  - One dimensional values
  - Can be real or complex

Add:

-1 + 3 = 2

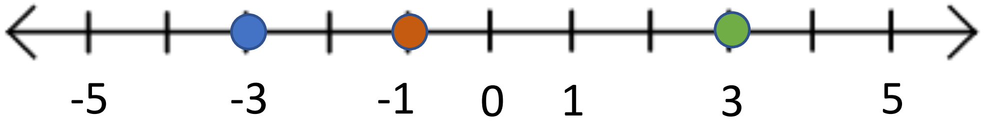


Let’s start smaller…

- Scalars
  - One dimensional values
  - Can be real or complex

Multiply:

-1 * 3 = -3

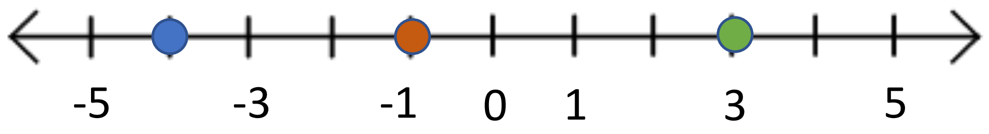


Let’s start smaller…

- Scalars
  - One dimensional values
  - Can be real or complex

Subtract:

-1 - 3 = -4



Let’s start smaller…

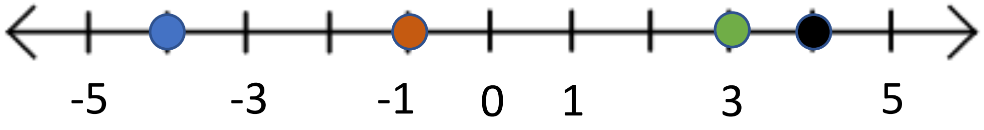
- Scalars
  - One dimensional values
  - Can be real or complex

Subtract:

$-1 - 3 = -4$

Subtract:

$3 - -1 = 4$



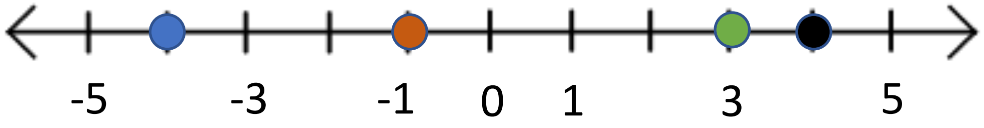
Let’s start smaller…

- Scalars
  - One dimensional values
  - Can be real or complex

Subtract:
-1 - 3 = -4

!=

Subtract:
3 - -1 = 4

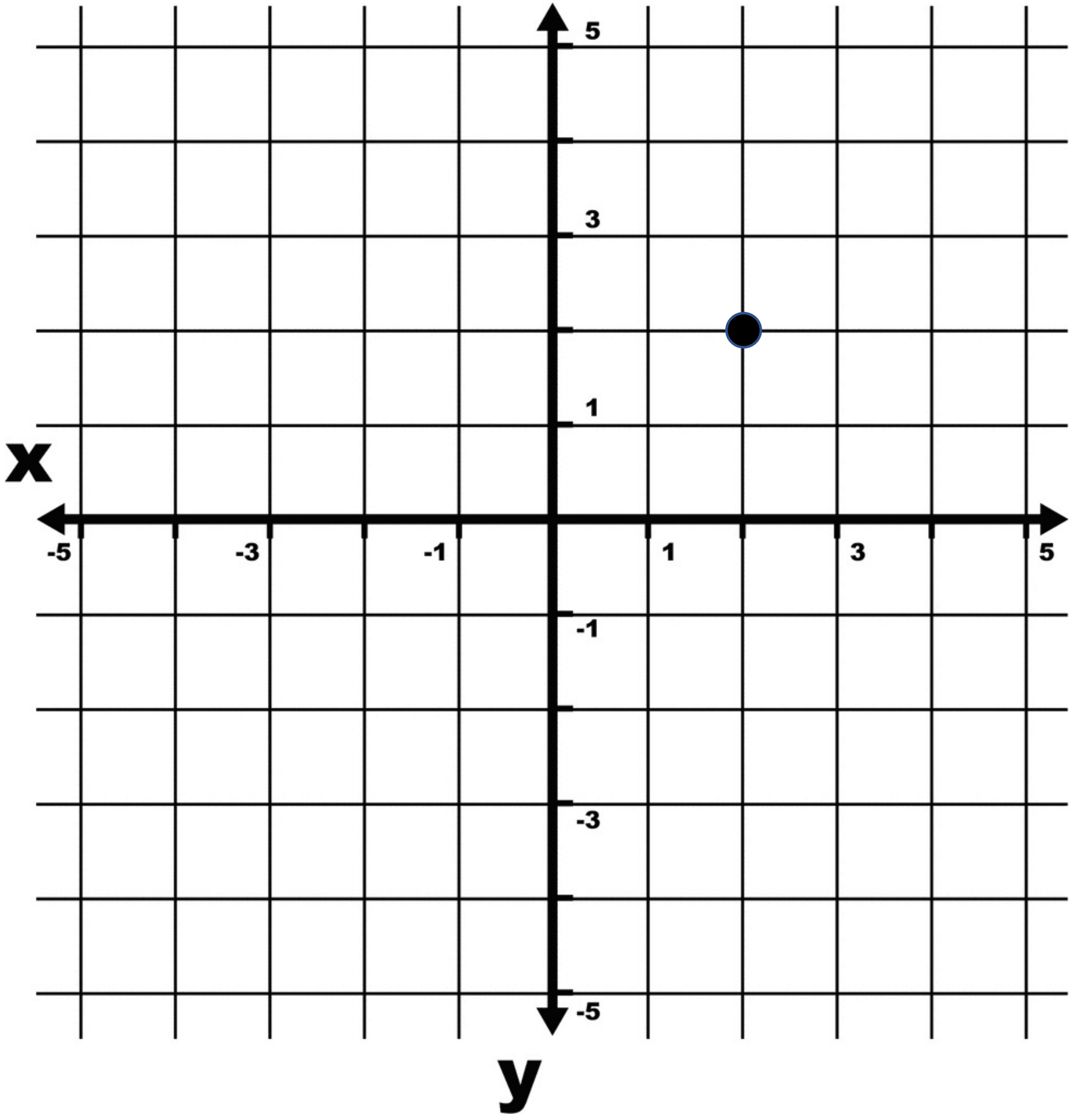


Points

- A location in space
- No size
- No direction

Points

- A location in space
- No size
- No direction
- We prefer them with respect to some other context…(more on that later)

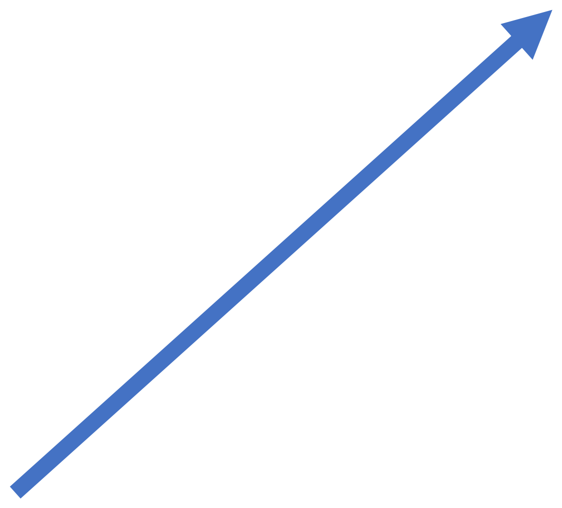


Points

- A location in space
- No size
- No direction
- We prefer them with respect to some other context…(more on that later)
- But for now, we'll ignore that

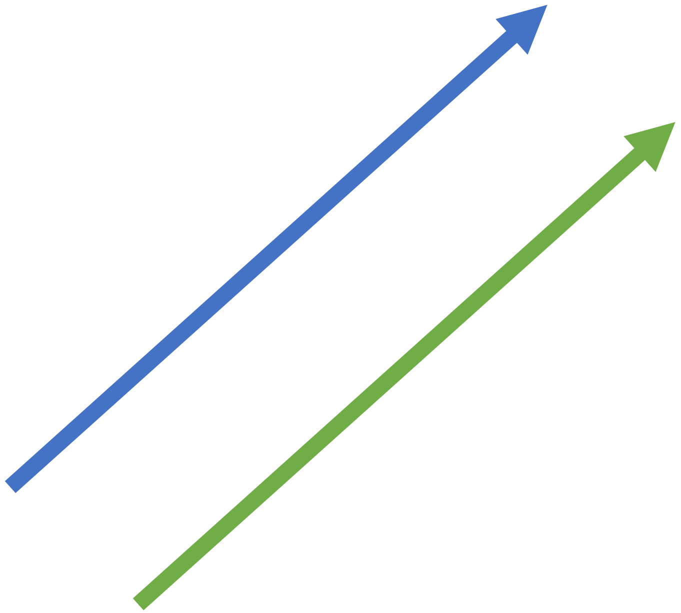
Vectors

- Have a direction
- Have magnitude
- A few basic concepts:



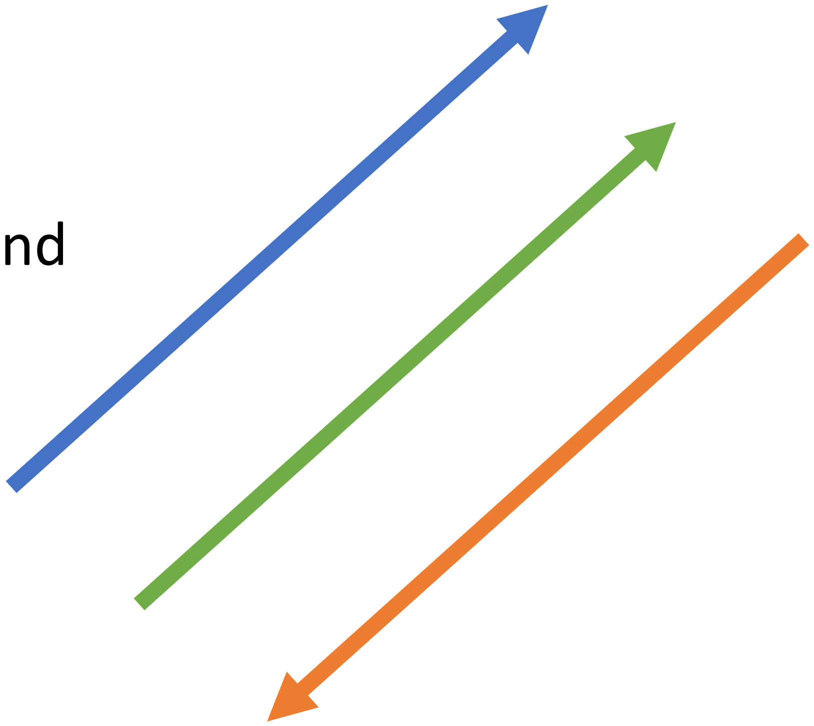
Vectors

- Have a direction
- Have magnitude
- A few basic concepts:
  - The blue and green vector are the same



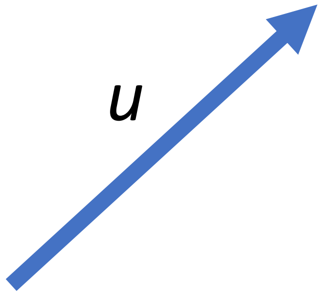
Vectors

- Have a direction
- Have magnitude
- A few basic concepts:
  - The blue and green vector are the same
  - The orange arrow is an inverse to the blue and green



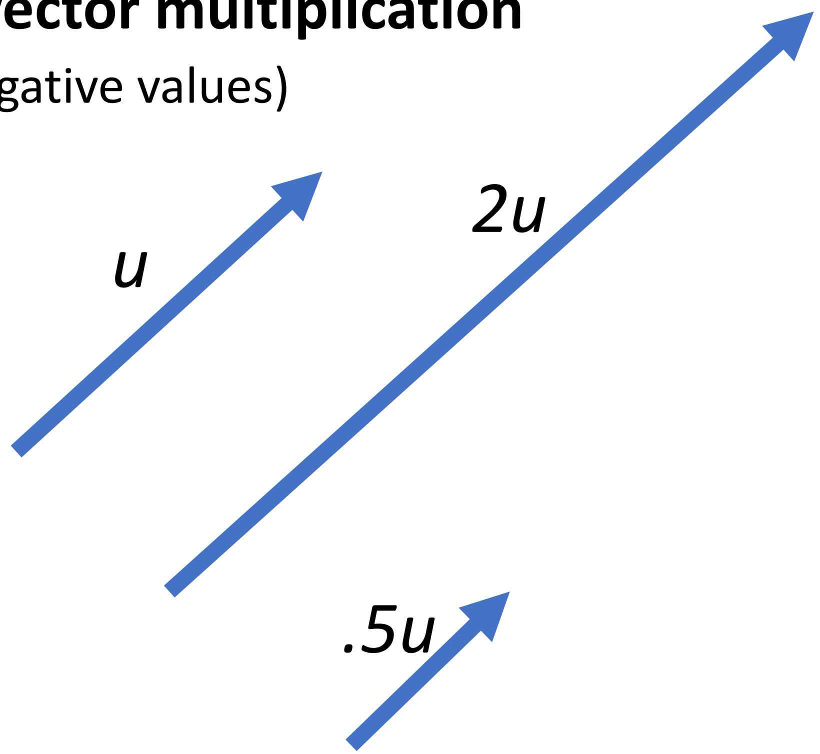
Vector Space

- We don’t have the concept of locations or distance
- What do we know without that context?



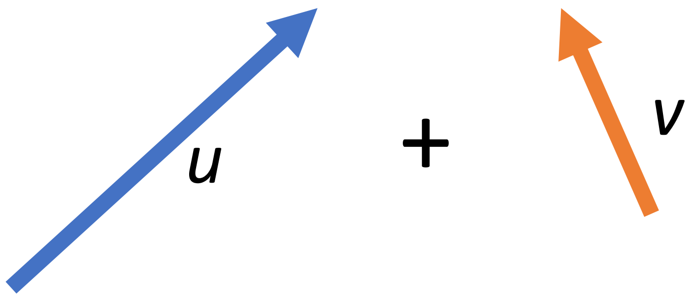
Vector Space

- We don't have the concept of locations or distance
- What do we know without that context?
    - We can create new vectors using **scalar-vector multiplication**
        - Has a scaling effect (and inverse if we use negative values)



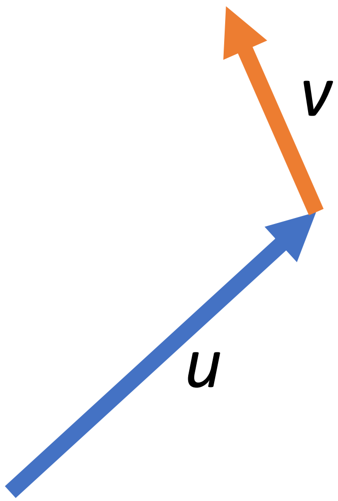
Vector Space

- We don’t have the concept of locations or distance
- What do we know without that context?
  - We can create new vectors using **scalar-vector multiplication**
    - Has a scaling effect (and inverse if we use negative values)
  - We can create new vectors using **vector-vector addition**
    - Head-tail axiom



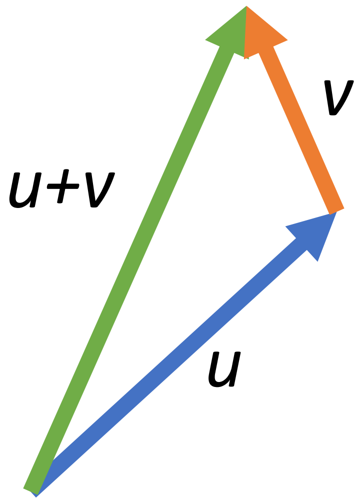
Vector Space

- We don’t have the concept of locations or distance
- What do we know without that context?
  - We can create new vectors using **scalar-vector multiplication**
    - Has a scaling effect (and inverse if we use negative values)
  - We can create new vectors using **vector-vector addition**
    - Head-tail axiom



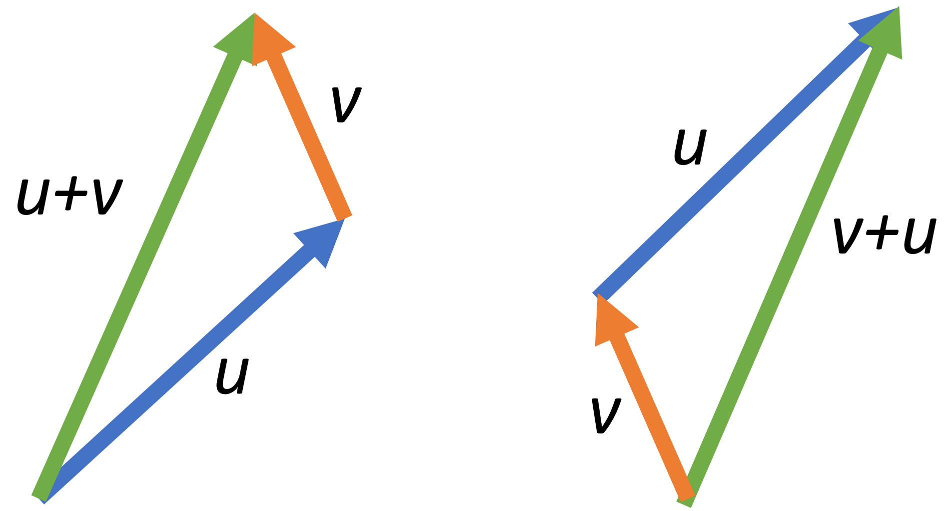
Vector Space

- We don’t have the concept of locations or distance
- What do we know without that context?
  - We can create new vectors using **scalar-vector multiplication**
    - Has a scaling effect (and inverse if we use negative values)
  - We can create new vectors using **vector-vector addition**
    - Head-tail axiom



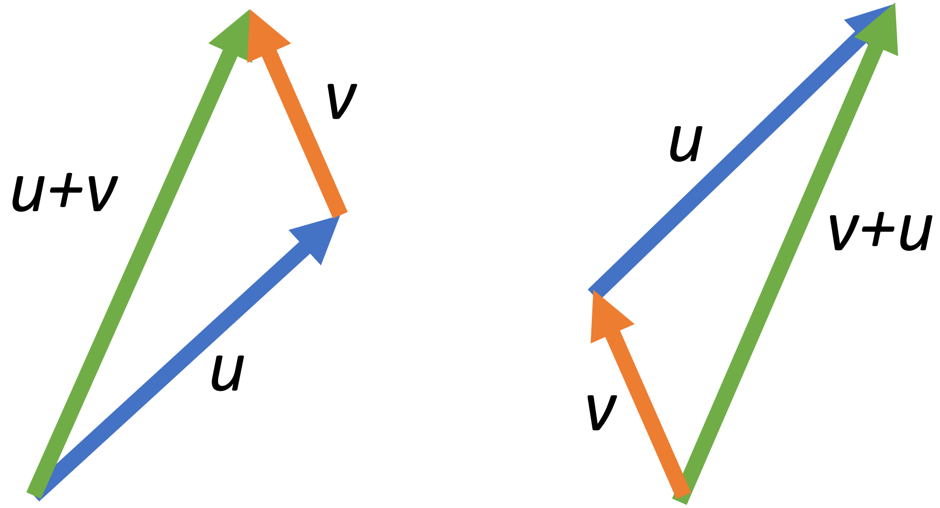
Vector Space

- We don’t have the concept of locations or distance
- What do we know without that context?
  - We can create new vectors using **scalar-vector multiplication**
    - Has a scaling effect (and inverse if we use negative values)
  - We can create new vectors using **vector-vector addition**
    - Head-tail axiom



Vector Space

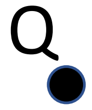
- We don’t have the concept of locations or distance
- What do we know without that context?
  - We can create new vectors using **scalar-vector multiplication**
    - Has a scaling effect (and inverse if we use negative values)
  - We can create new vectors using **vector-vector addition**
    - Head-tail axiom



$u+v == v+u$

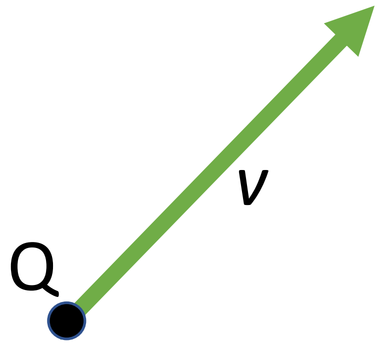
Affine Space

- Now…we add the point to vector space!
- With our new object we can perform additional operations:
  - **Point-vector addition** where the vector can displace a point and arrive at a new point.



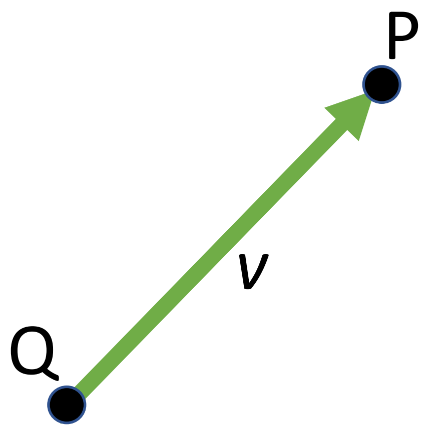
Affine Space

- Now…we add the point to vector space!
- With our new object we can perform additional operations:
  - **Point-vector addition** where the vector can displace a point and arrive at a new point.



Affine Space

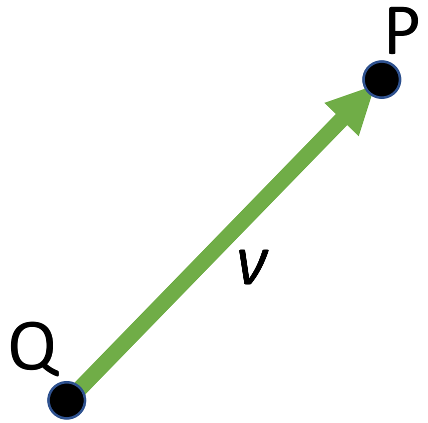
- Now…we add the point to vector space!
- With our new object we can perform additional operations:
  - **Point-vector addition** where the vector can displace a point and arrive at a new point.



Affine Space

- Now…we add the point to vector space!
- With our new object we can perform additional operations:
  - **Point-vector addition** where the vector can displace a point and arrive at a new point.

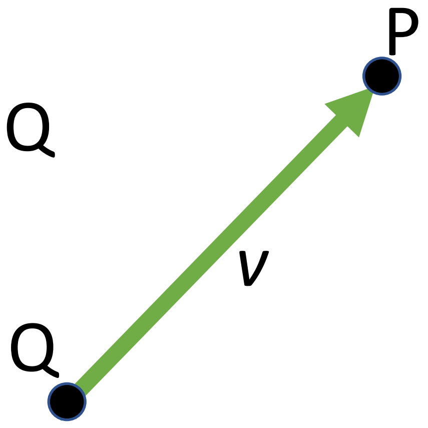
$$P = Q + v$$



Affine Space

- Now…we add the point to vector space!
- With our new object we can perform additional operations:
  - **Point-vector addition** where the vector can displace a point and arrive at a new point.
  - **Point-point subtraction** can provide a vector between two points

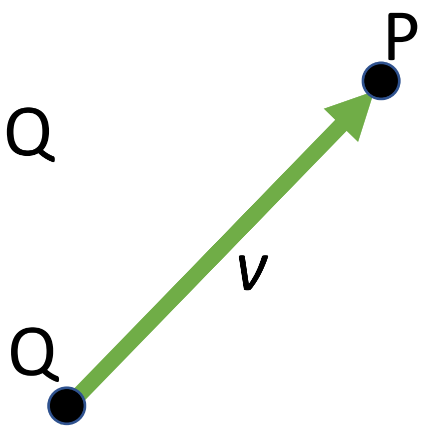
$v = P - Q$

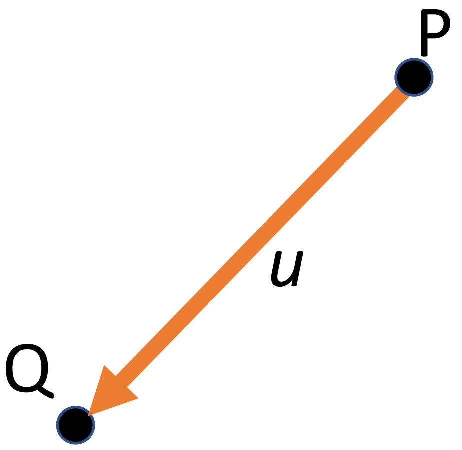


Affine Space

- Now…we add the point to vector space!
- With our new object we can perform additional operations:
  - **Point-vector addition** where the vector can displace a point and arrive at a new point.
  - **Point-point subtraction** can provide a vector between two points

$v = P - Q$



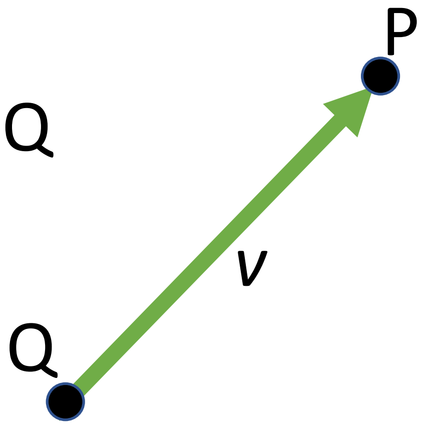


$u = Q - P$

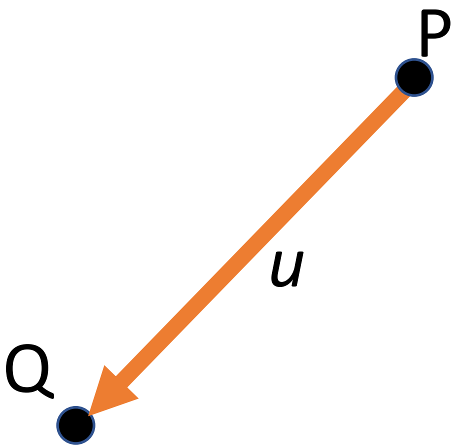
Affine Space

- Now…we add the point to vector space!
- With our new object we can perform additional operations:
  - **Point-vector addition** where the vector can displace a point and arrive at a new point.
  - **Point-point subtraction** can provide a vector between two points
    - **Careful!** The direction of the vector depends on the order of the subtraction.

$v = P - Q$



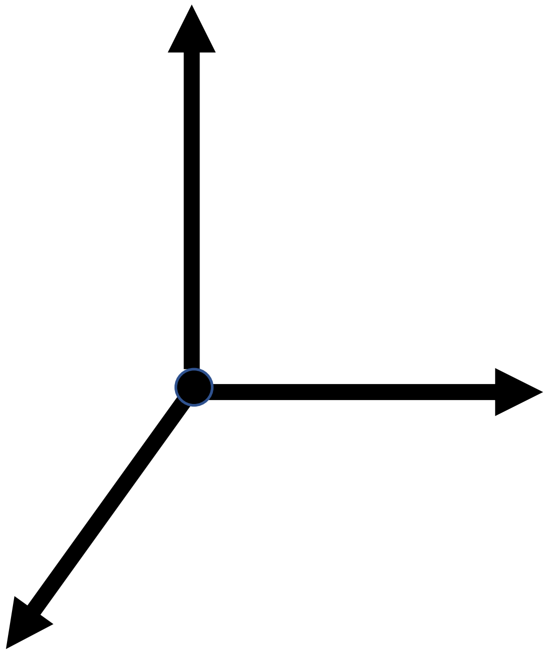
$v \neq u$



$u = Q - P$

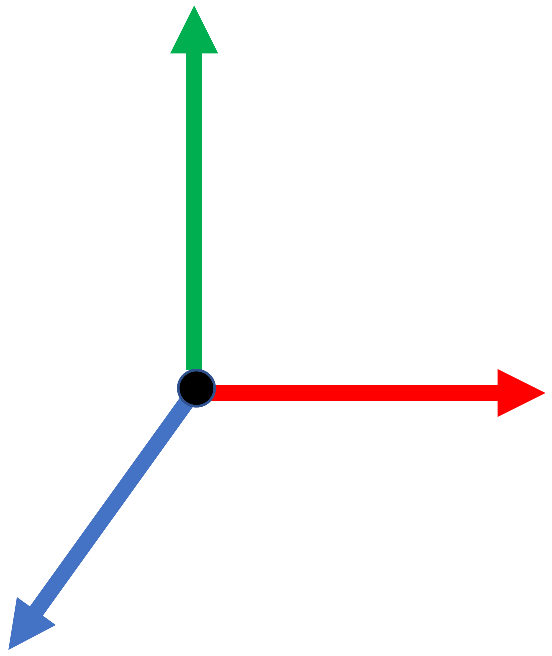
Getting a **Frame** of Reference

- Our point can serve as an origin to a set of vectors



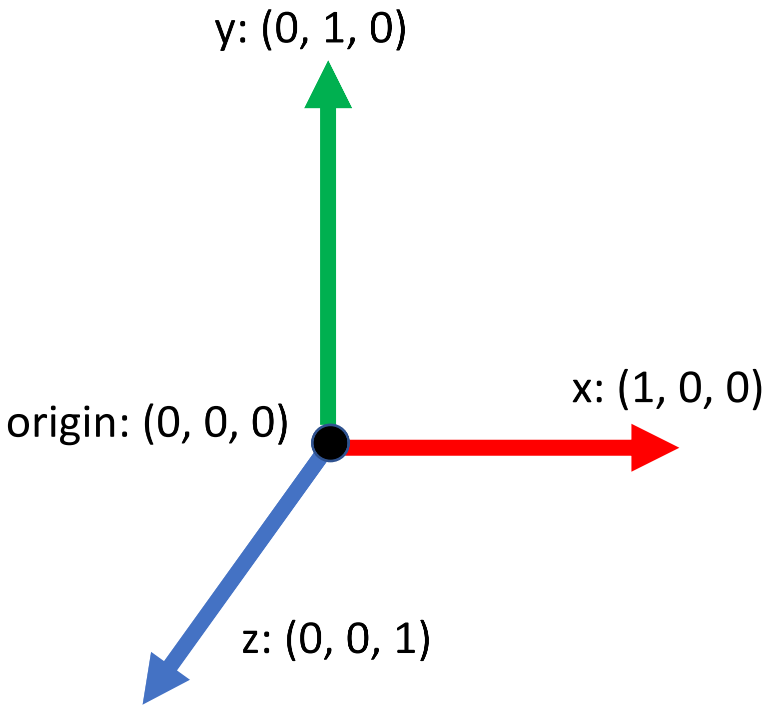
Getting a **Frame** of Reference

- Our point can serve as an origin to a set of vectors
- These basis vectors serve as our coordinate **frame**



Getting a **Frame** of Reference

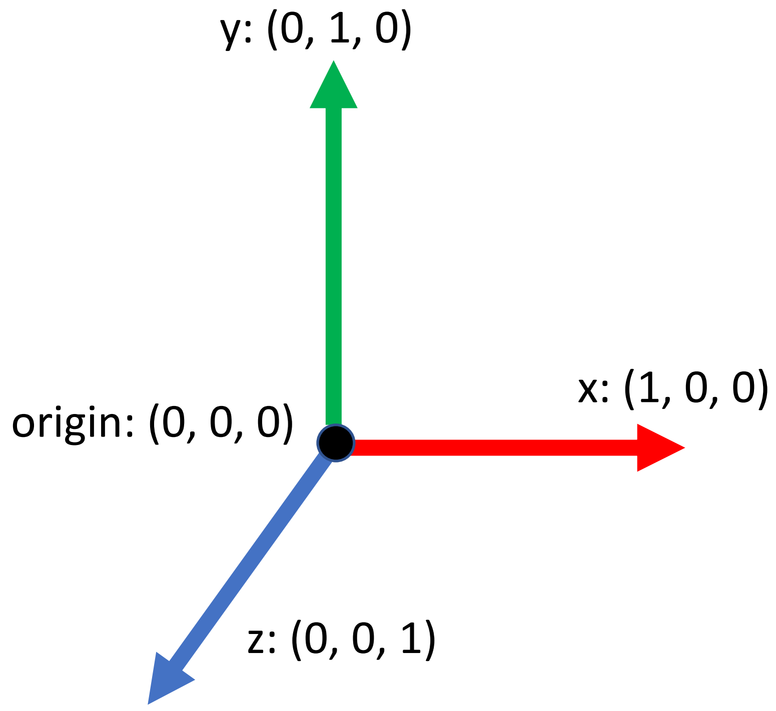
- Our point can serve as an origin to a set of vectors
- These basis vectors serve as our coordinate **frame**
- Adding in some Euclidean coordinates…



Getting a **Frame** of Reference

- Our point can serve as an origin to a set of vectors
- These basis vectors serve as our coordinate **frame**
- Adding in some Euclidean coordinates…
- We can represent vectors using three scalar values
    - Each vector has a x, y, and z component
    - These components are multiplied by each base vector

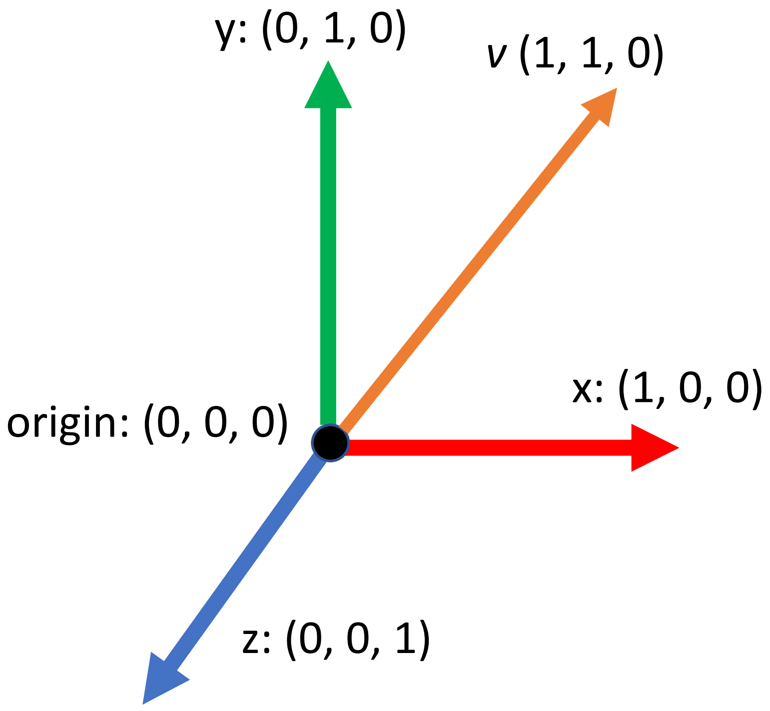
$$v = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \begin{array}{l} \text{x component} \\ \text{y component} \\ \text{z component} \end{array}$$



Getting a **Frame** of Reference

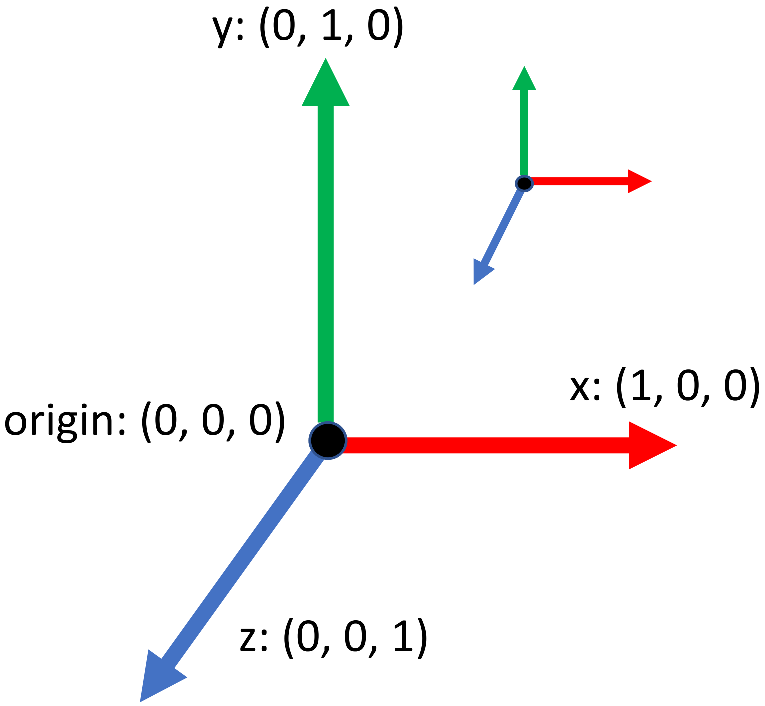
- Our point can serve as an origin to a set of vectors
- These basis vectors serve as our coordinate **frame**
- Adding in some Euclidean coordinates…
- We can represent vectors using three scalar values
  - Each vector has a x, y, and z component
  - These components are multiplied by each base vector
  - The result is a new vector originating from the origin

$$v = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \begin{array}{l} \text{x component} \\ \text{y component} \\ \text{z component} \end{array} \qquad v^T = (\ 1, 1, 0\ )$$



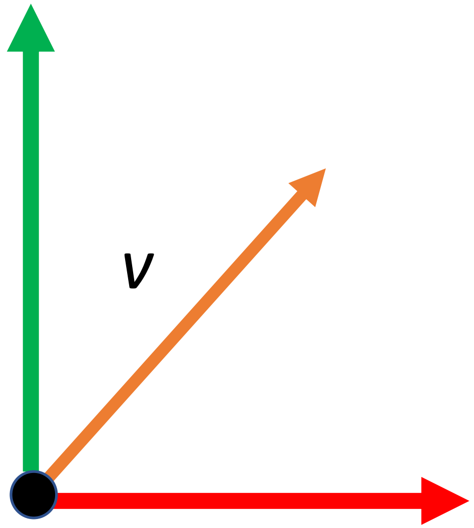
Frames Representations

- This frame representation is of the clip coordinates we have been using in class
- However, we can have multiple independent frames
- Each frame can have its own coordinates
- You CANNOT work with vectors in different coordinate systems unless they are translated to the same coordinate space
- We'll get deeper into this when we talk about matrices



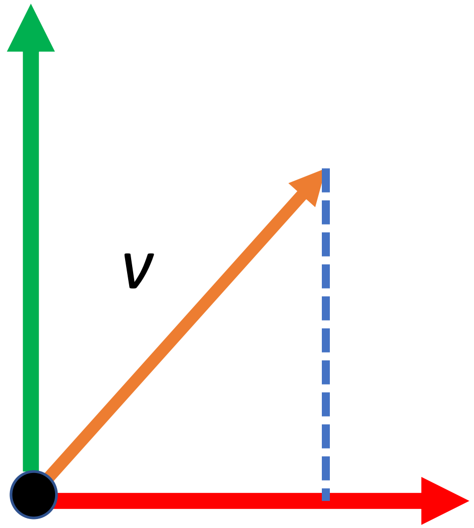
Vector Math - Magnitude

- Represented as |v| in the book
- The magnitude is the length of the vector



Vector Math - Magnitude

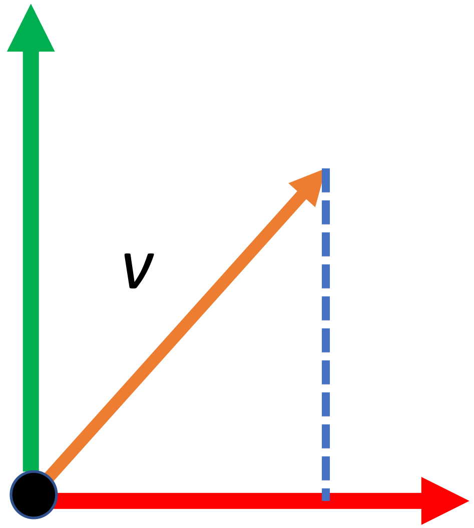
- Represented as |v| in the book
- The magnitude is the length of the vector
- Visually we can see it forms a very familiar shape



Vector Math - Magnitude

- Represented as |v| in the book
- The magnitude is the length of the vector
- Visually we can see it forms a very familiar shape
- We can use the Pythagorean Theorem!

In 2D: $|v| = \sqrt{{v_x}^2 + {v_y}^2}$

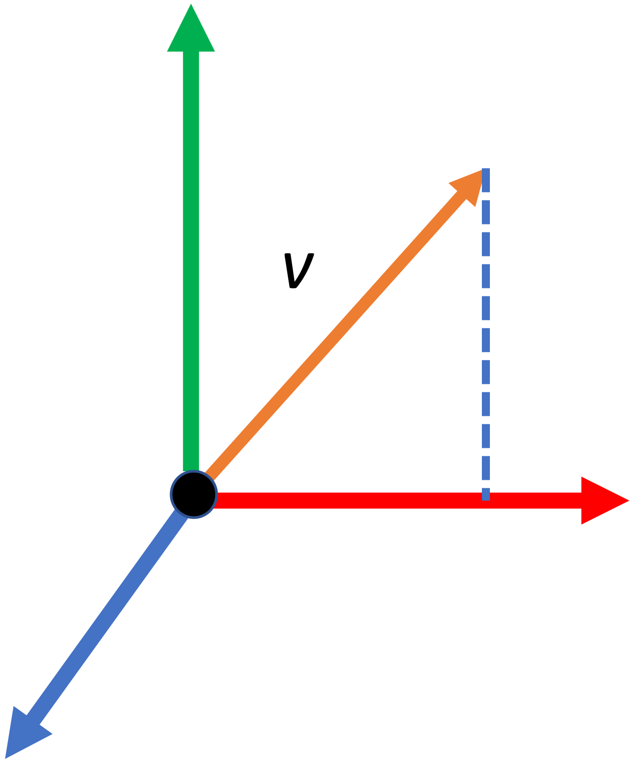


Vector Math - Magnitude

- Represented as |v| in the book
- The magnitude is the length of the vector
- Visually we can see it forms a very familiar shape
- We can use the Pythagorean Theorem!

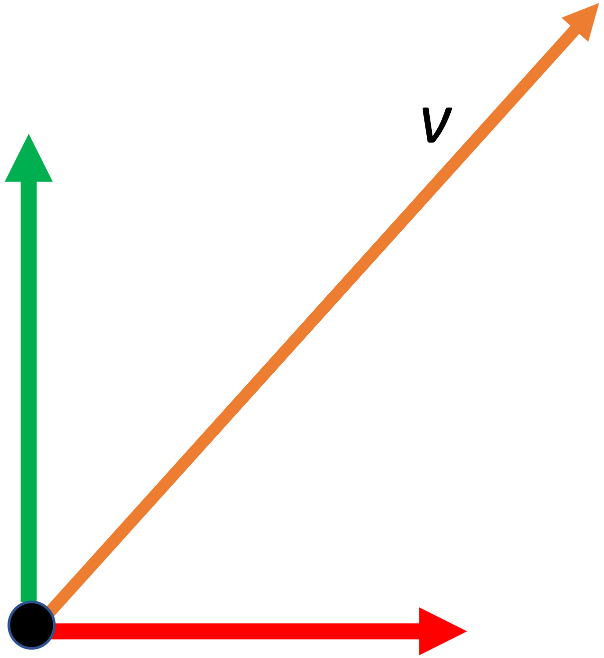
In 2D: $|v| = \sqrt{v_x{}^2 + v_y{}^2}$

In 3D: $|v| = \sqrt{v_x{}^2 + v_y{}^2 + v_z{}^2}$



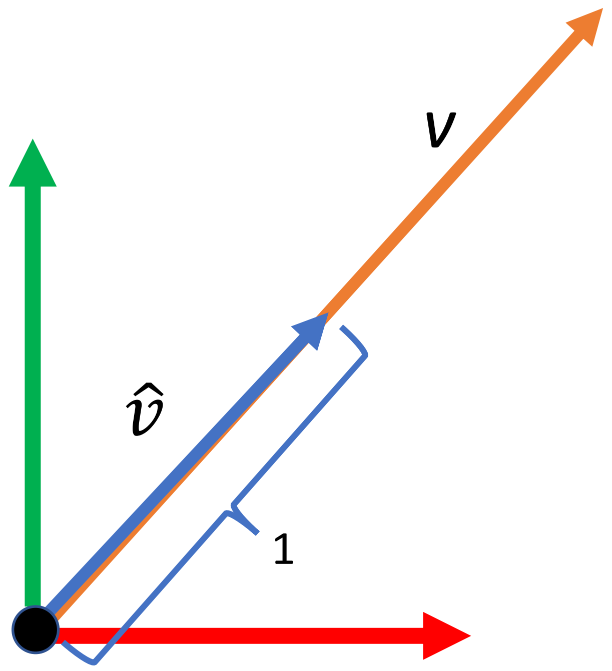
Vector Math - Normalization

- Normalizing a vector create a **unit vector** meaning the length/magnitude of the vector is 1
  - Sometimes referred to as a **direction vector**



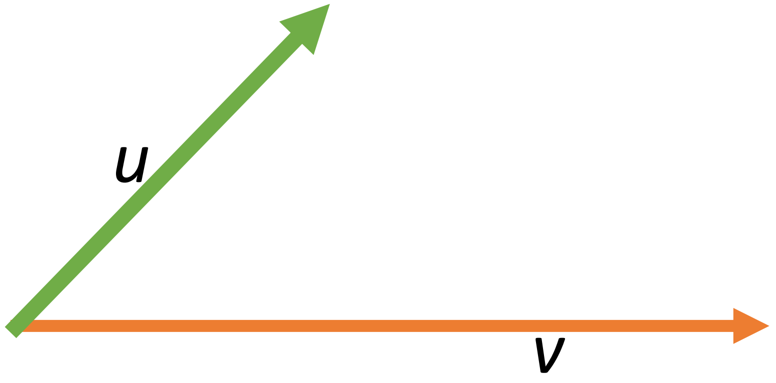
Vector Math - Normalization

- Normalizing a vector create a **unit vector** meaning the length/magnitude of the vector is 1
  - Sometimes referred to as a **direction vector**
- Represented as $\hat{v} = v \mathbin{/} |v|$
  - Note that $|v|$ is a scalar value
  - Scalar-vector multiplication
  - $\hat{v} = v \ * (1 \mathbin{/} |v|)$



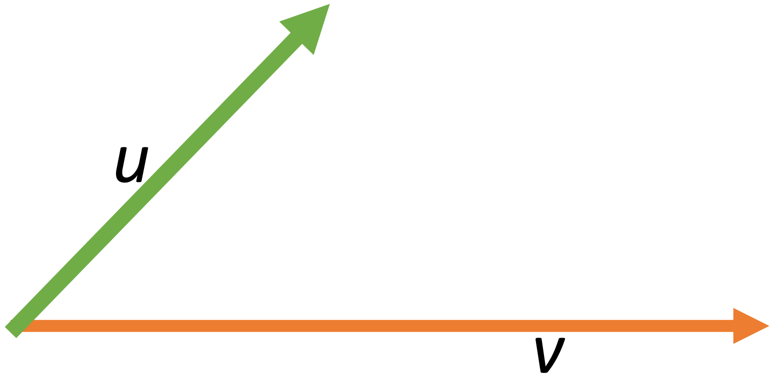
Vector Math – Dot Product

- One form of vector multiplication
- $u \cdot v = u_x v_x + u_y v_y + u_z v_z$



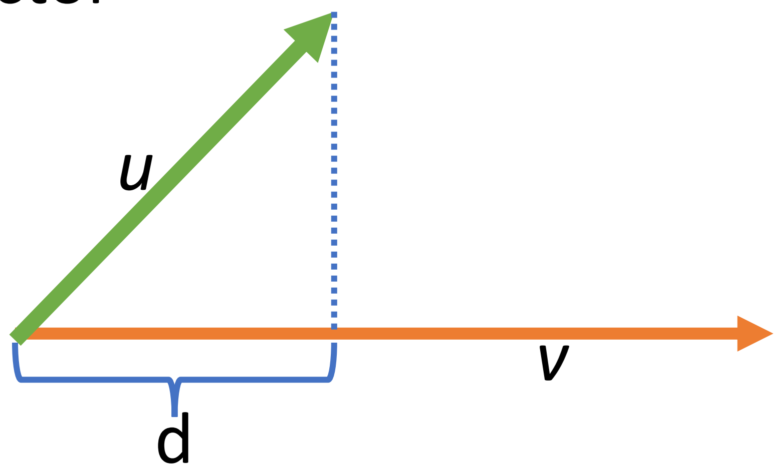
Vector Math – Dot Product

- One form of vector multiplication
- $u \cdot v = u_x v_x + u_y v_y + u_z v_z$
  - $u \cdot v$ will be a scalar value
- Note: $u \cdot u$ is the squared length of the vector



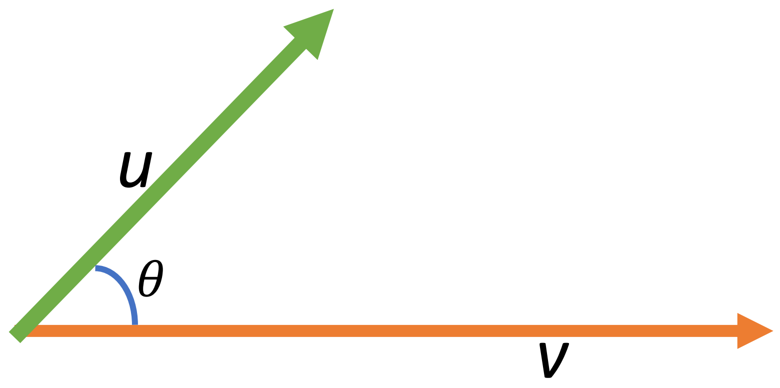
Vector Math – Dot Product

- One form of vector multiplication
- $u \cdot v = u_x v_x + u_y v_y + u_z v_z$
    - $u \cdot v$ will be a scalar value
- Note: $u \cdot u$ is the squared length of the vector
- Can be used to find the length of the projection of u onto v
    - If $|v| = 1$, $d = u \cdot v$
    - else $d = (u \cdot v) / |v|$



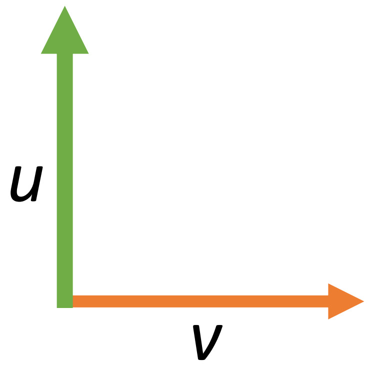
Vector Math – Dot Product

- Can be calculated a second way
- $u \cdot v = |u||v|\cos\theta$
- Note if u and v are both unit vectors this simplifies to:
  - $u \cdot v = \cos\theta$



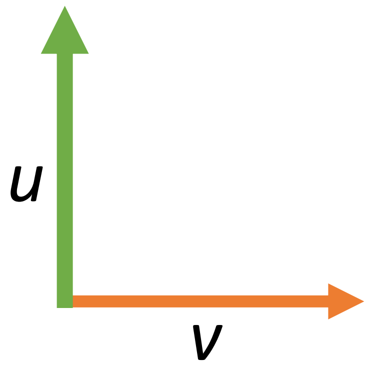
Vector Math – Dot Product Properties

- If two vectors are:
  - orthogonal (perpendicular): $u \cdot v = 0$



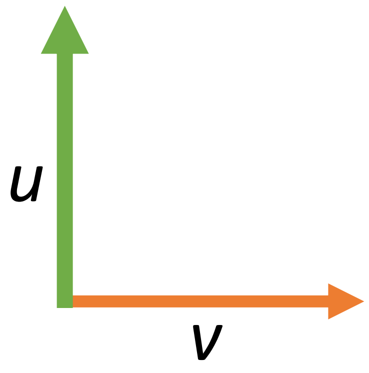
Vector Math – Dot Product Properties

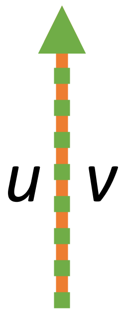
- If two vectors are:
  - orthogonal (perpendicular): $u \cdot v = 0$
    - Careful! There is a 0 vector…



Vector Math – Dot Product Properties

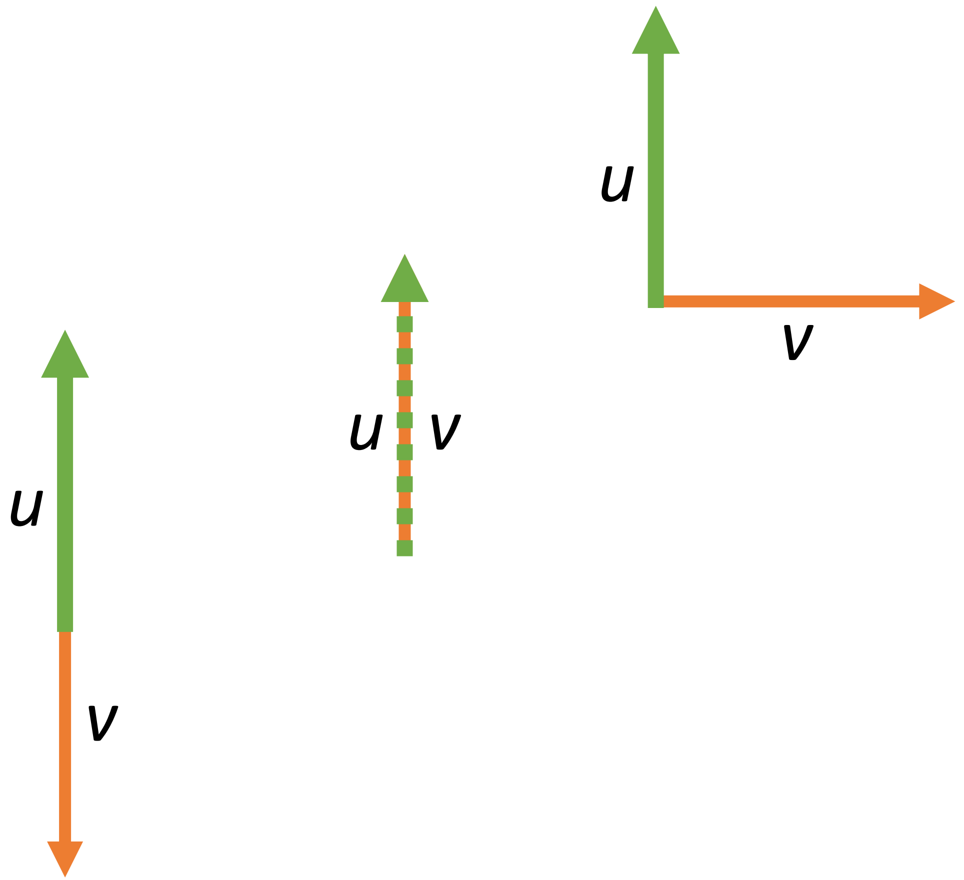
- If two vectors are:
  - orthogonal (perpendicular): $u \cdot v = 0$
    - Careful! There is a 0 vector…
  - same direction: $u \cdot v = 1$





Vector Math – Dot Product Properties

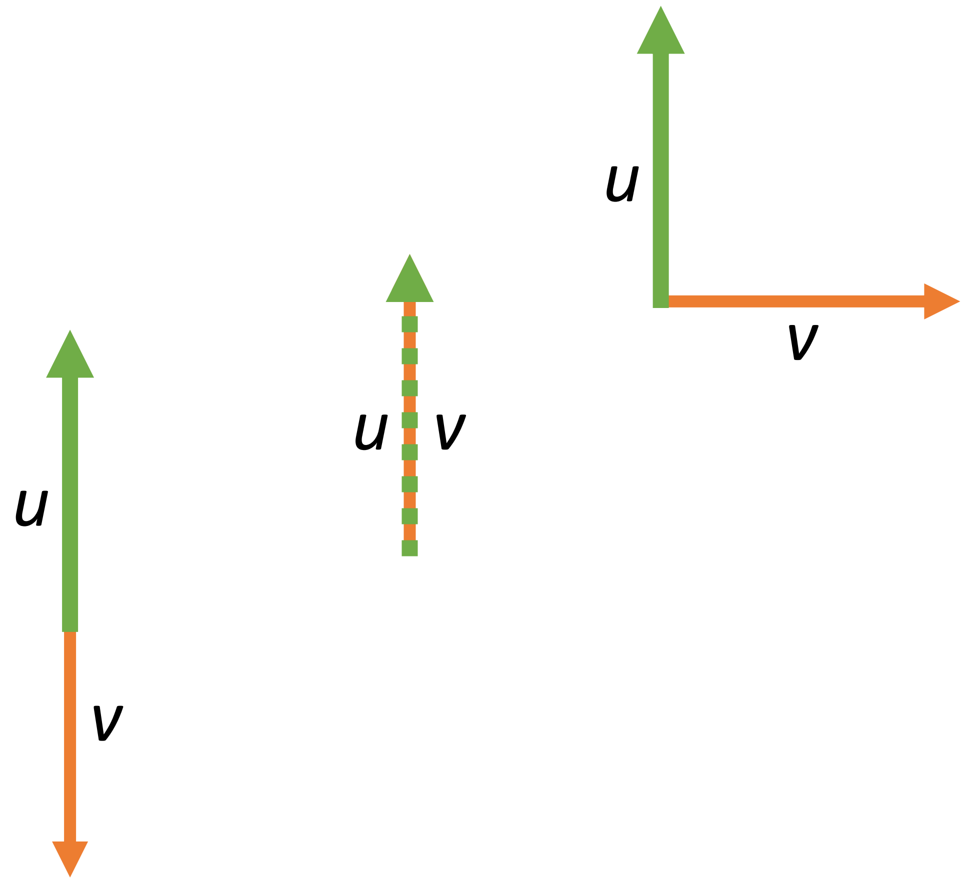
- If two vectors are:
  - orthogonal (perpendicular): $u \cdot v = 0$
    - Careful! There is a 0 vector…
  - same direction: $u \cdot v = 1$
  - opposite directions: $u \cdot v = -1$



Visualization: https://twitter.com/freyaholmer/status/1200807790580768768?lang=en

Vector Math – Dot Product Properties

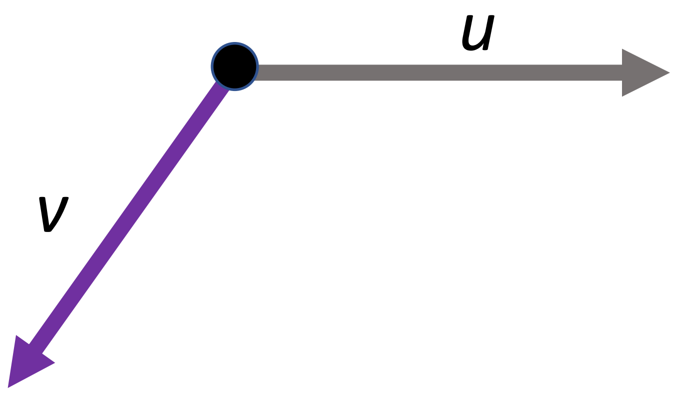
- If two vectors are:
  - orthogonal (perpendicular): $u \cdot v = 0$
    - Careful! There is a 0 vector…
  - same direction: $u \cdot v = 1$
  - opposite directions: $u \cdot v = -1$



- Commutative
  - $u \cdot v = v \cdot \mathrm{u}$
- Associative
  - $u \cdot (v + q) = \mathrm{u} \cdot v + \mathrm{u} \cdot q$
- Distributive
  - $\alpha u \cdot v = v \cdot \alpha u = \alpha(u \cdot v)$

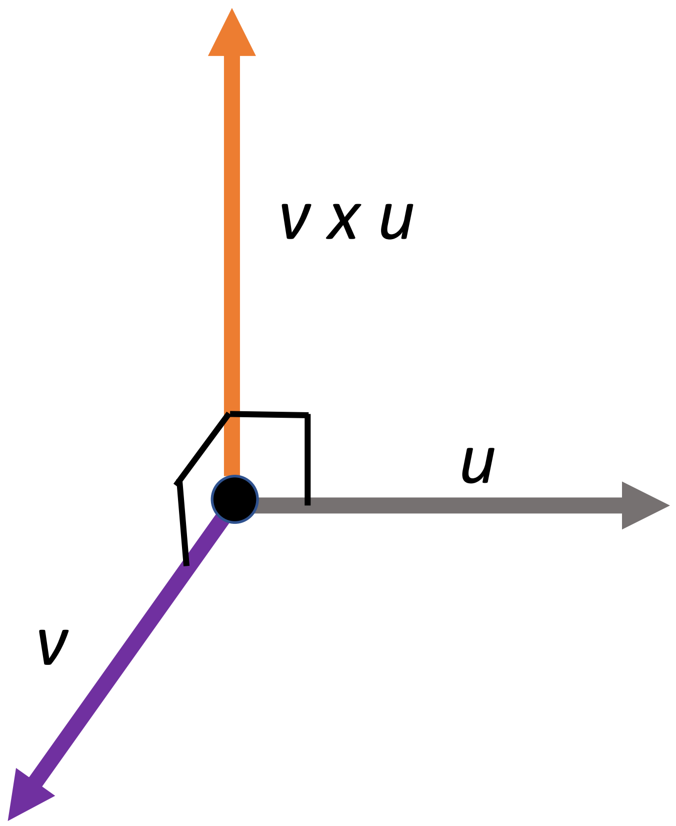
Visualization: https://twitter.com/freyaholmer/status/1200807790580768768?lang=en

Cross Product Visually



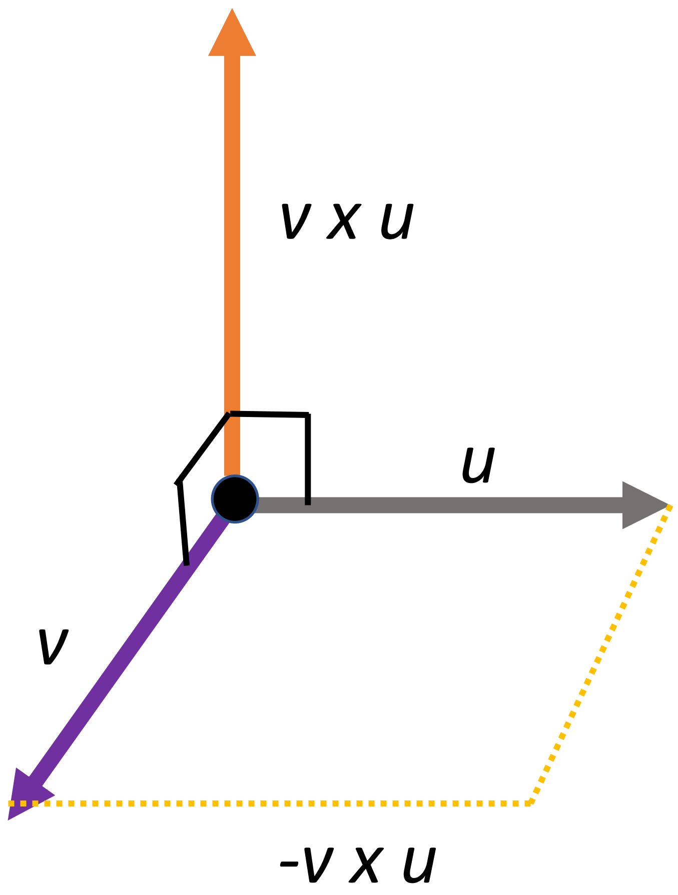
Cross Product Visually

- *u* and *v* do not need to be orthogonal but *v* x *u* (*v* cross *u*) will produce a vector orthogonal to both



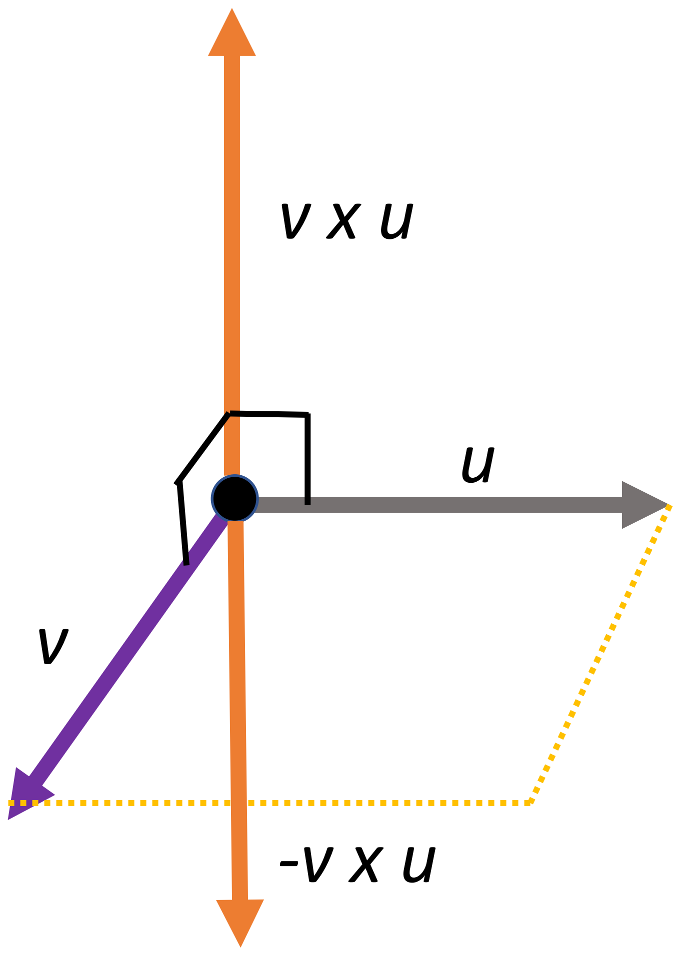
Cross Product Visually

- *u* and *v* do not need to be orthogonal but *v* x *u* (*v* cross *u*) will produce a vector orthogonal to both
- In 3D, the length of the cross product vector will also be the area of the parallelogram



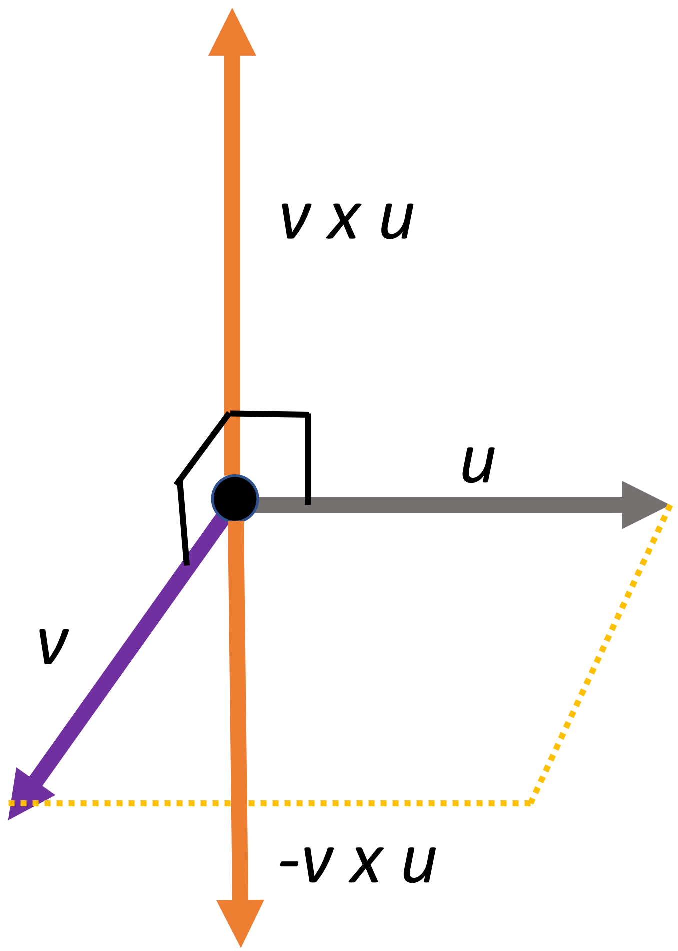
Cross Product Visually

- *u* and *v* do not need to be orthogonal but *v* x *u* (*v* cross *u*) will produce a vector orthogonal to both
- In 3D, the length of the cross product vector will also be the area of the parallelogram
- A vector that points outward from a plane is called a **normal**



Cross Product Visually

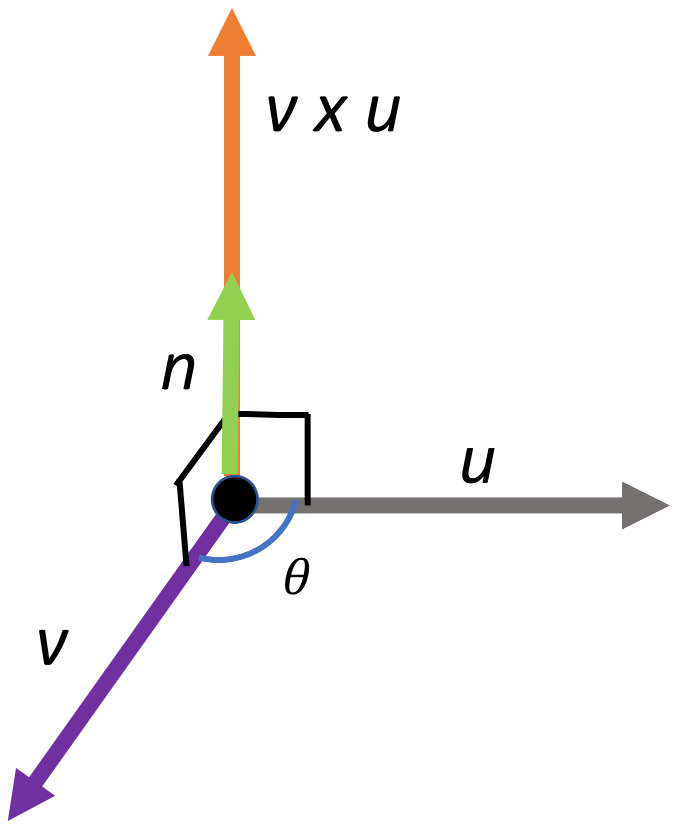
- *u* and *v* do not need to be orthogonal but *v* x *u* (*v* cross *u*) will produce a vector orthogonal to both
- In 3D, the length of the cross product vector will also be the area of the parallelogram
- A vector that points outward from a plane is called a **normal**
- If u and v are pointing in the same direction (regardless of length) the cross product is a 0 vector



Visualization: https://twitter.com/freyaholmer/status/1203421083371749376?lang=en

Calculating the Cross Product

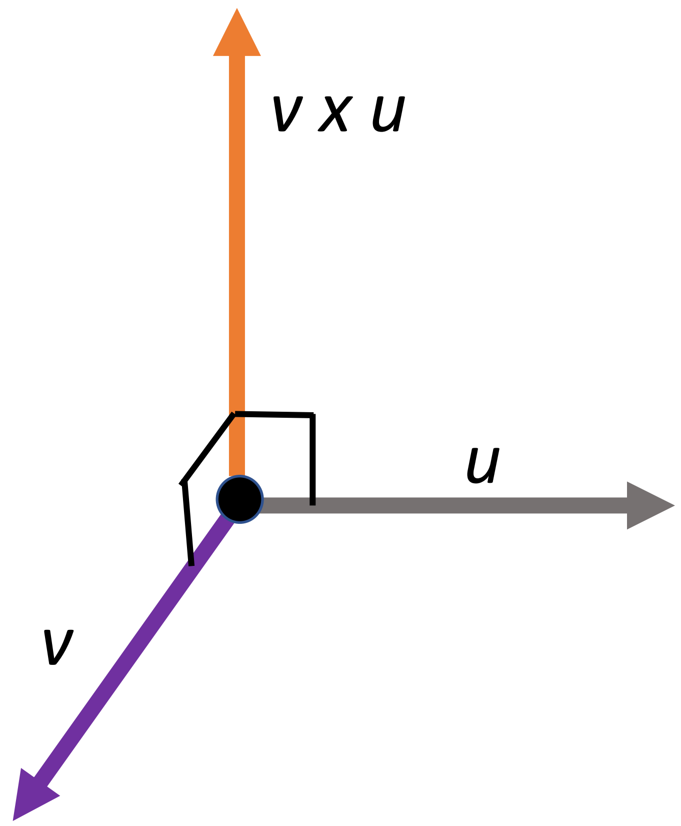
- Method #1
- $v \times u = |v||u| \sin(\theta)\, n$
  - n is a perpendicular unit vector



Calculating the Cross Product

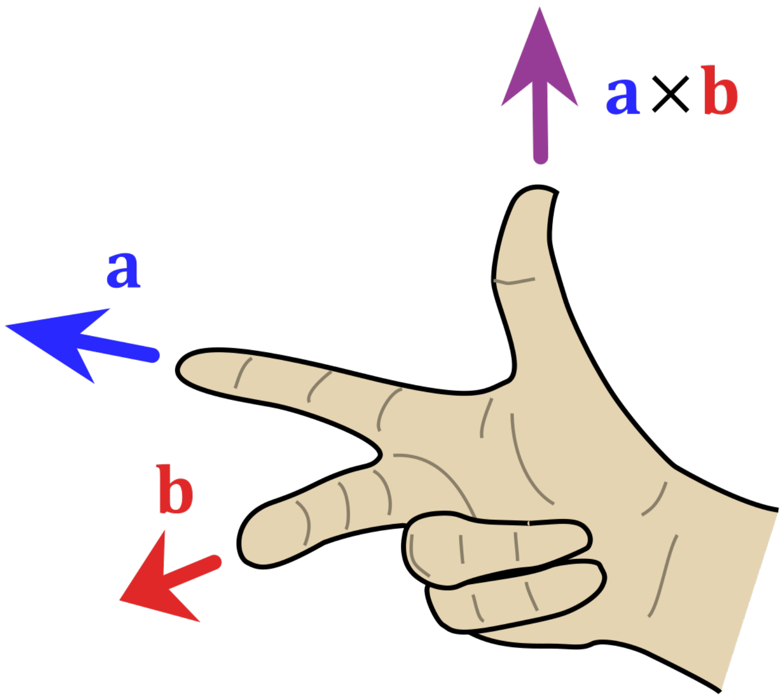
- Method #1
- $v \text{ x } u = |v||u| \sin(\theta)\, n$
  - n is a perpendicular unit vector

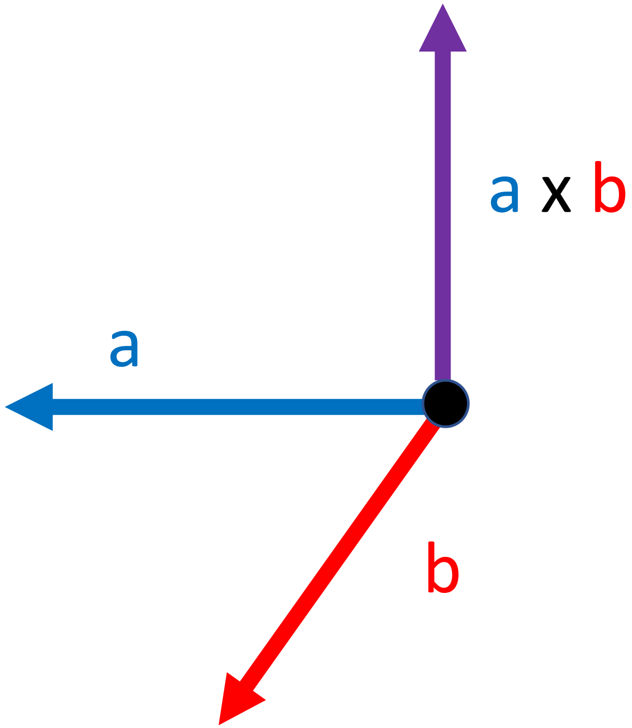
- Method #2
- $v \text{ x } u = (c_x, c_y, c_z)$
  - $c_x = v_y u_z – v_z u_y$
  - $c_y = v_z u_x – v_x u_z$
  - $c_z = v_x u_y - v_y u_x$



Cross Product “Handedness”

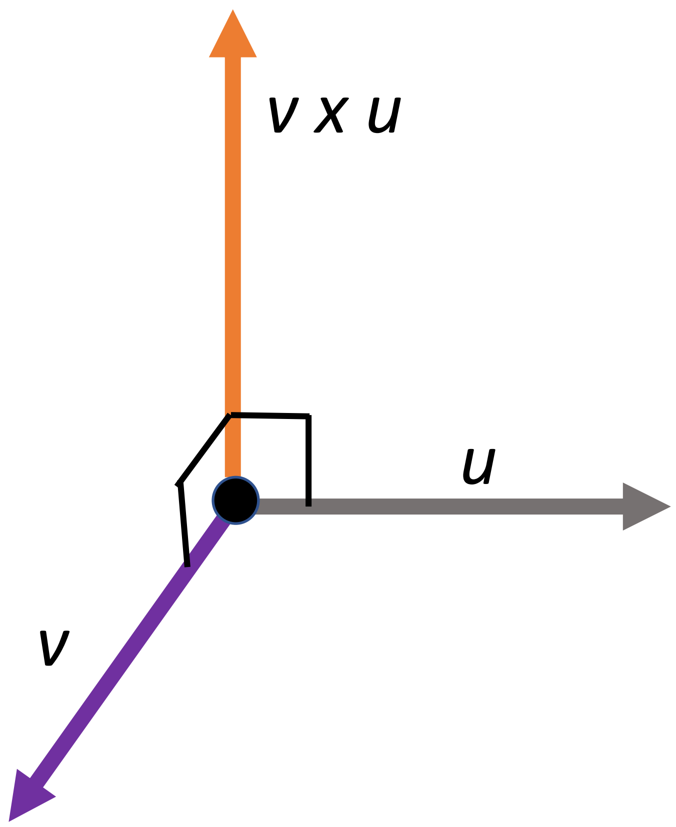
- Use the “right-hand rule” to determine if a x b will be point up or down





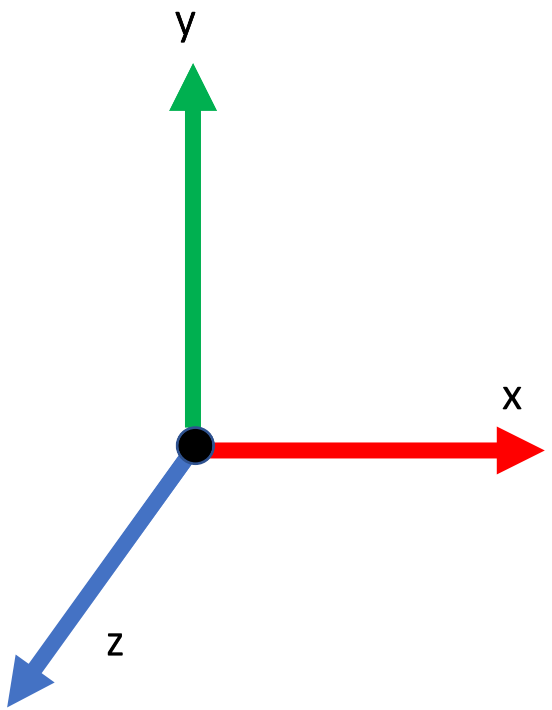
Cross Product Properties

- $v \times u = -(v \times u)$
- $(\alpha v) \times u = \alpha(v \times u)$
- $v \times (u + q) = v \times u + v \times q$



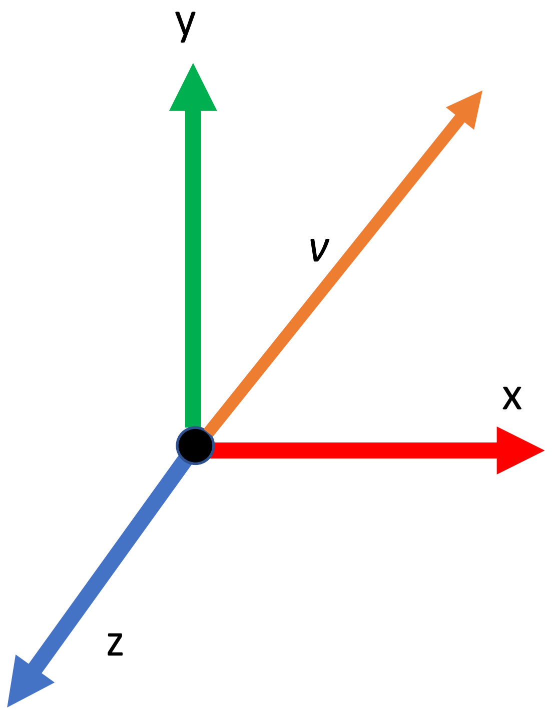
Vectors vs Points

- The distinction between vectors and points is obvious from a mathematical perspective



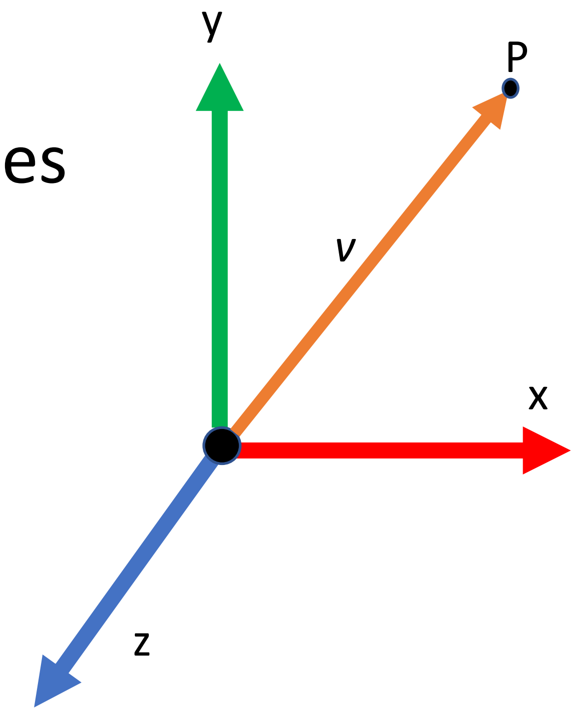
Vectors vs Points

- The distinction between vectors and points is obvious from a mathematical perspective
- All vectors can be represented as a series of scalar values multiplied by the basis vectors
  - v = $\alpha_1 \mathrm{x} + \alpha_2 \mathrm{y} + \alpha_3 \mathrm{z}$



Vectors vs Points

- The distinction between vectors and points is obvious from a mathematical perspective
- All vectors can be represented as a series of scalar values multiplied by the basis vectors
  - v = $\alpha_1 x + \alpha_2 y + \alpha_3 z$
- All points can be represented as a series of scalar values multiplied by the basis vectors AND knowledge of the origin
  - P = P0 + β_1x + β_2y + β_3z
- This can lead to some confusion in representation



Homogeneous Coordinates

- To remove the ambiguity of representation
- In 3D we extend the representation of vectors and points to 4D
- Vectors are represented as:
  - $w = \begin{bmatrix} \delta \\ \delta \\ \delta \\ \delta \\ 0 \end{bmatrix}$
- Points are represented as:
  - $P = \begin{bmatrix} \alpha \\ \alpha \\ \alpha \\ \alpha \\ 1 \end{bmatrix}$