Introduction to 3D game programming

Module 6:

- Trigonometry.
- Vectors in 1, 2 and 3-space.
- Operations on Vectors

Today’s Workshop:

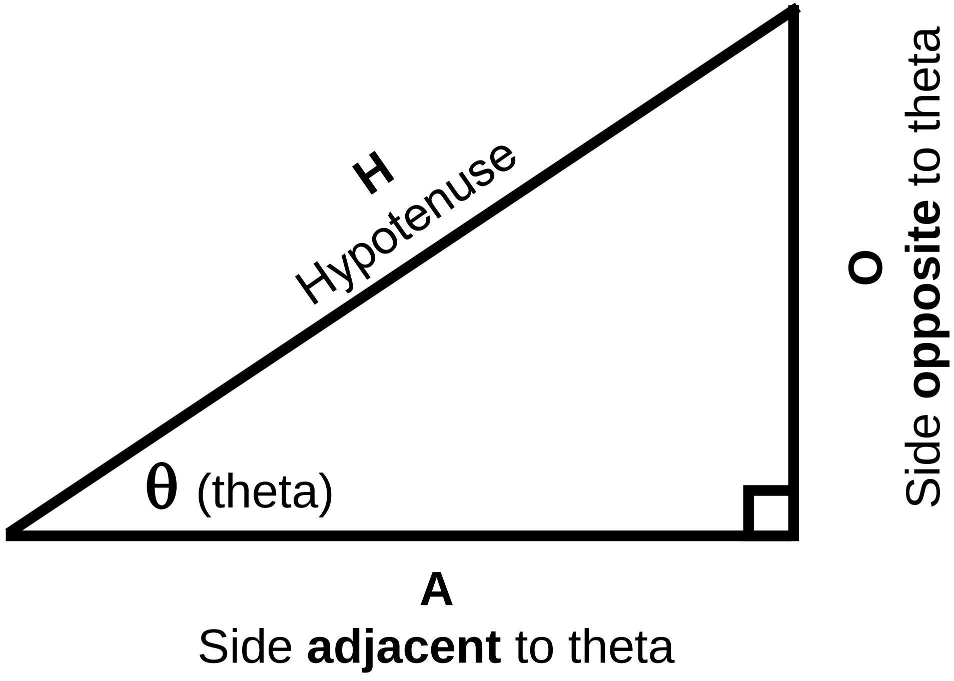
- Workshop 6 - vectors.




Working with Triangles

Right Triangles

- The properties of right triangles (one angle is 90º) are very useful for analysing vectors and points in 2D and also 3D.



Right Triangles

- The properties of right triangles (one angle is 90º) are very useful for analysing vectors and points in 2D and also 3D.
- Pythagoras:

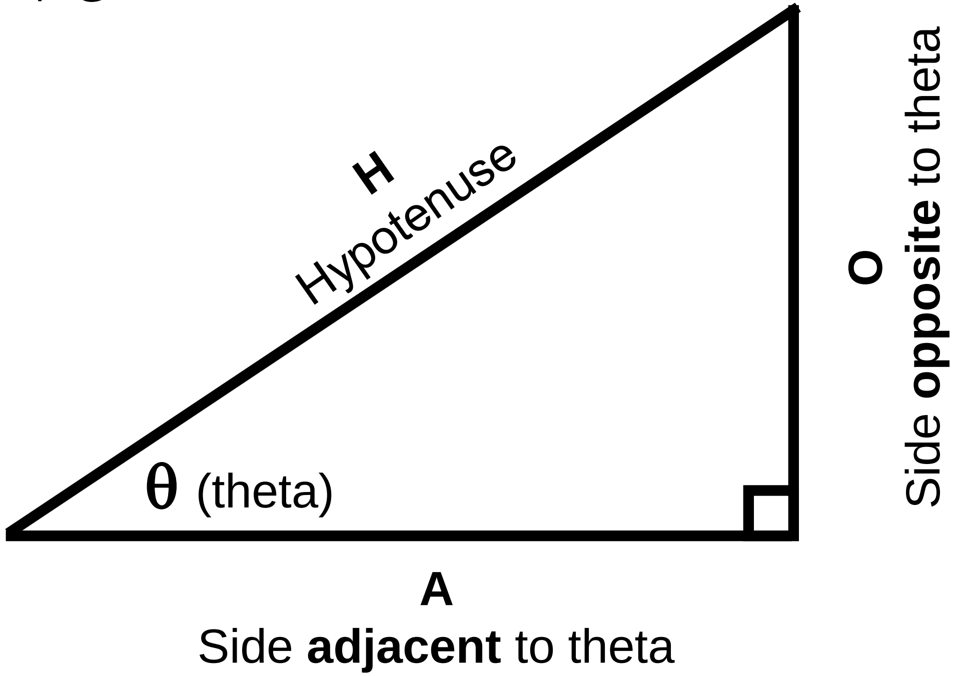
$$H = \sqrt{A^2 + O^2}$$

- Trigonometric Identities:

$$\cos(\theta) = \frac{A}{H}$$

$$\sin(\theta) = \frac{O}{H}$$

$$\tan(\theta) = \frac{O}{A}$$



Measuring Angles

- There are two standards for measuring units of angles – Degrees (^{o}) and Radians (R).
- 1 full circle rotation is
  - 360^o
  - $(2 * pi)^R$, where pi = $\pi \approx 3.1415927$
- Both systems are common – so we have to know how to convert between the two
  - C/C++ functions sin, cos, tan, asin, acos, atan work with radians.
  - OpenGL glRotatef function takes an angle in degrees.

Converting Angle Units

$360^o = 2pi^R$, or $180^o = pi^R$

So there are 180/pi degrees in 1 radian.

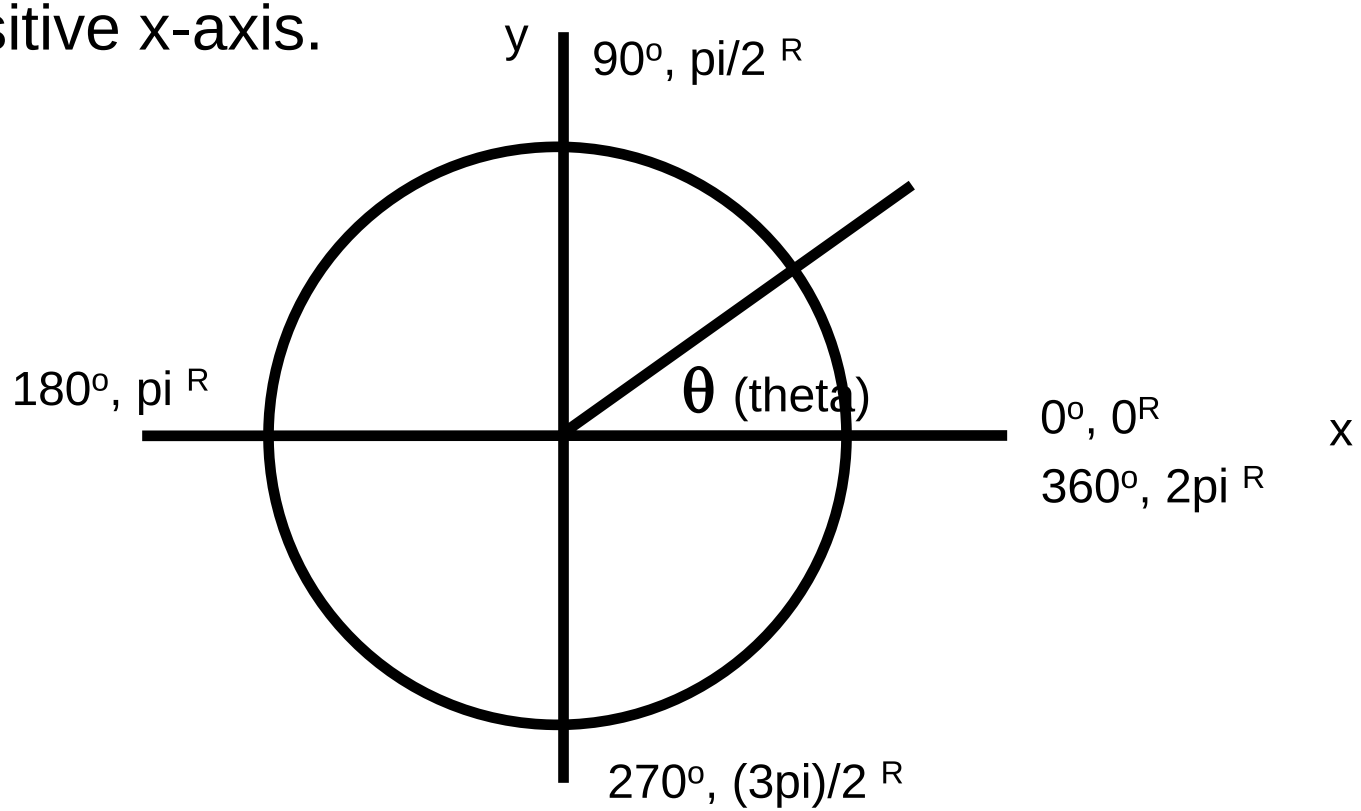
So:

Angle in Degrees = 180(Angle in Radians)/pi

Angle in Radians = pi(Angle in Degrees)/180

Measuring Angles

- Angles ‘wrap around’ so 360^o is the same as 0^o
- By convention, angle increases anti-clockwise from positive x-axis.



Working with Angles in Code

- C/C++ defines sin, cos, tan and their inverses asin, acos, atan the the math.h library. These work with angles in radians.
- Rather than performing the complete conversion calculation between degrees and radians each time – its faster to define a conversion ratio.

```
const float pi = 3.1415927;
const float DegToRad = pi/180;
const float RadToDeg = 180/pi;
```

  - To convert an angle measured in one system to the other, just multiply by the appropriate constant.

Working with Angles in Code

- The sin, cos, and tan functions can be computationally expensive if you need to call them often. If this becomes a problem there are several solutions.
  - Use the compiler to optimise the code (see #pragma intrinsic in MSDN), **usually the best option**.
  - Create a lookup table.

```
const float pi = 3.1415927; // declare constants
const float DegToRad = pi/180;
float getSin[360]; // create the lookup table and fill in
for (int i=0; i<360; i++)
   {
   getSin[i] = sin(DegToRad*i);
   }
// later that very same program…
float IwantSin90degrees = getSin[90];
```

Vectors




Scalars and Vectors

- A scalar quantity indicates a magnitude only
  - 100 metres, 110 Km per hour.
- A vector quantity indicates both magnitude and direction
  - 100 metres south, 110 Km per hour East.

Examples of Vectors

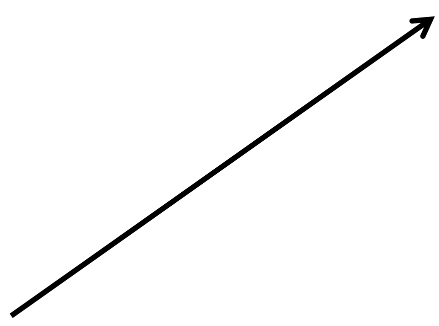
- Displacement – distance with direction
- Velocity – speed with direction
- Acceleration – change in velocity
- Force – eg. friction, gravity
- Momentum
- … and a lot more

Uses of Vectors

- Motion
- Collision Detection
- Collision Response
- Physics
- Light models

Vector Notation

- Vectors are represented geometrically/graphically as directed line segments with line segment length representing magnitude and arrow representing direction (polar form), or as a set of Cartesian coordinates.
- Vectors **do not represent a position in space**, rather a magnitude and direction of some quantity.

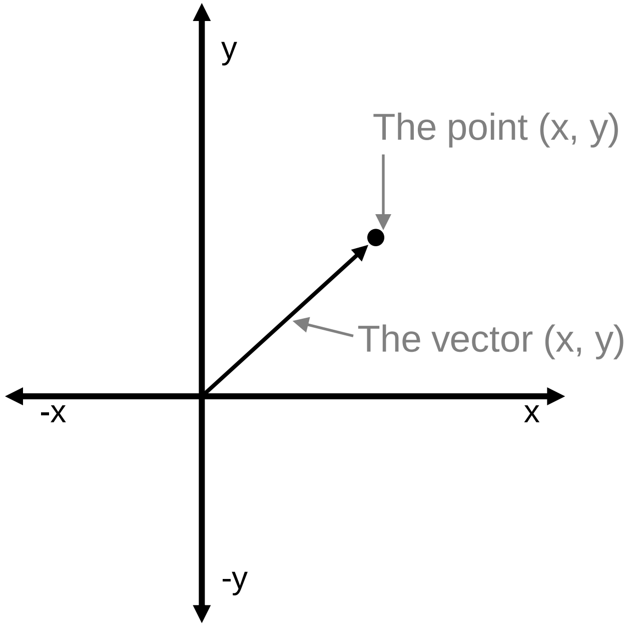


Vector Notation

- In math books, often represented in variable form by bold type face and a lower case letter:
  - i.e. **a** = (1, 3, 9), **b** = (2, 4, 8)
- In Cartesian form often denoted in brackets,
  - i.e. (1, 5, 7, 8, 9).
- Vector indices
  - a component of a vector is generally referenced via subscript
  - if $\mathbf{v} = (6, 8, 10)$
  - then $\mathbf{v}_1 = 6$, $\mathbf{v}_2 = 8$ and $\mathbf{v}_3 = 10$
  - can also use x,y, and z, or 0,1, and 2 to identify the components.
    - vector[0] = 6; vector[1] = 8; vector[2] = 10;

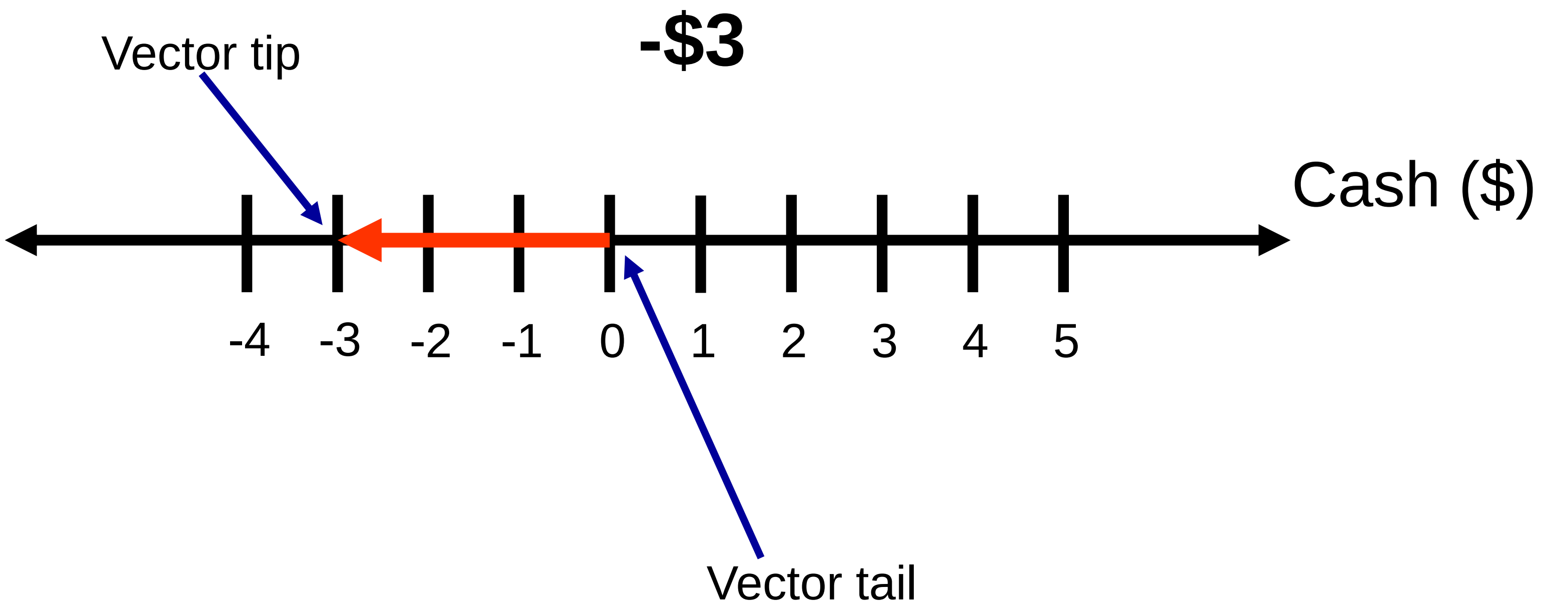
Point or Vector?

- What is the difference between a point and a vector?



Vectors in 1D

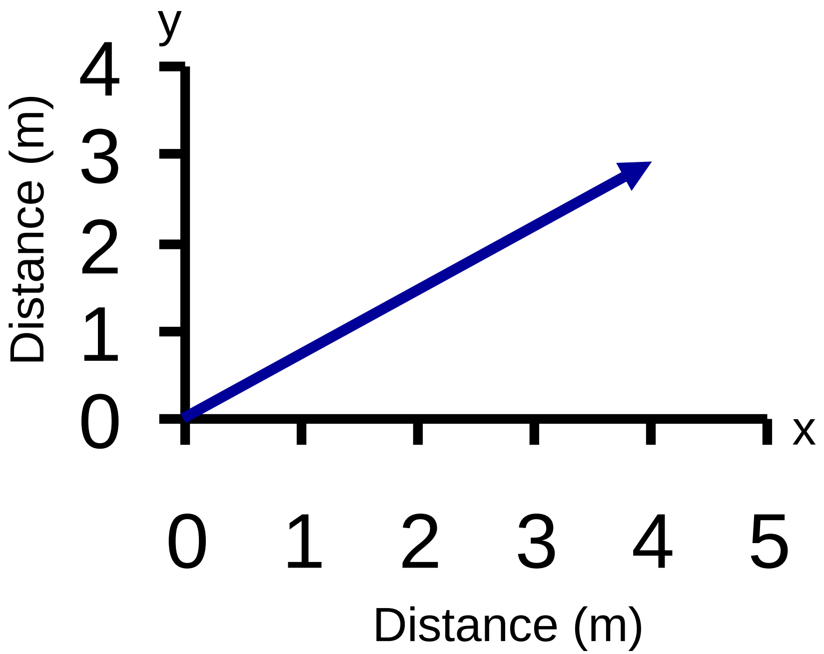
- In 1D there are only two directions, negative and positive, so the sign of a 1D number is enough to indicate direction.



Vectors in 2D

- In 2D – a magnitude and direction of a vector can be represented on a graph as an angle and distance from origin.

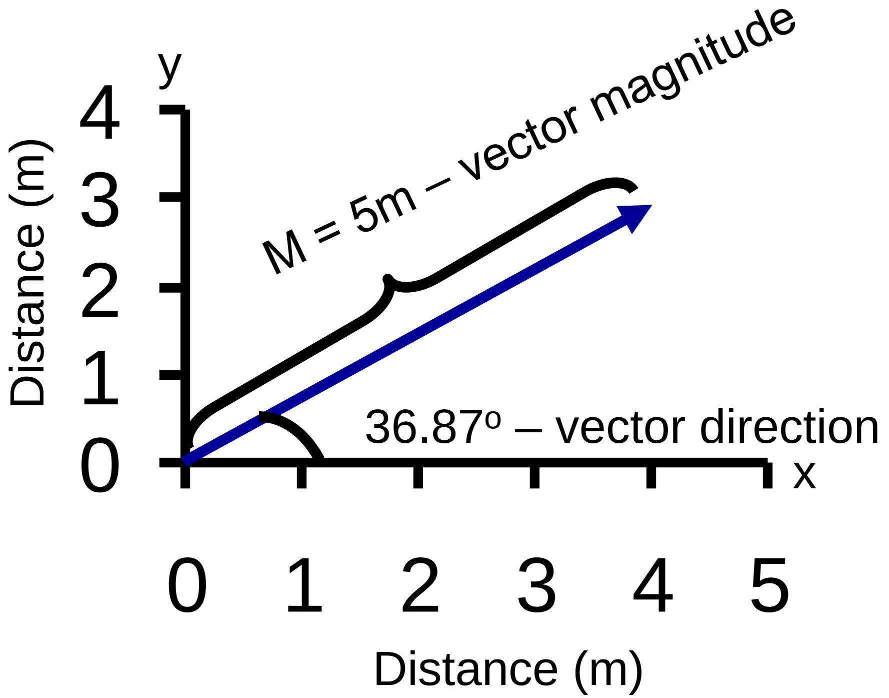
eg. Displacement of magnitude (M) 5 metres at 36.87^{o} to the x-axis



Vectors in 2D

- In 2D – a magnitude and direction of a vector can be represented on a graph as an angle and distance from origin.

eg. Displacement of magnitude (M) 5 metres at 36.87^{o} to the x-axis, written as $5 \angle 36.87^{o}$



Polar V’s Cartesian Form

- Describing a vector in terms of magnitude and angle is known as polar form.
- The same vector can also be defined as a translation in Cartesian space, that is the movement from any point by (X, Y)
  - We simply refer to this as (ΔX, ΔY) or (X, Y)
    - Δ, pronounced “delta” represents “change in”
- Conversion between polar and Cartesian form is done using the trigonometric identities.

Polar to Cartesian Form

$$\cos(\theta) = \frac{A}{H}$$

$$\cos(\theta) = \frac{x}{M}$$

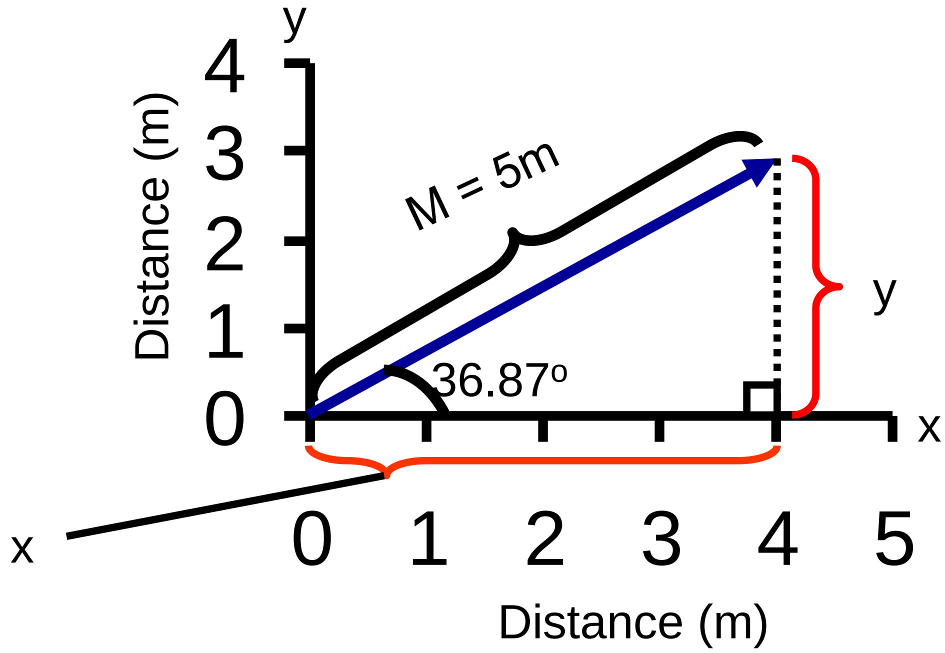
$$M\cos(\theta) = x$$

$$x = 5\cos(36.87^o)$$

$$x = 4m$$

$$y = M\sin(\theta)$$

$$y = 3m$$



Cartesian to Polar Form

Using Pythagoras:

$$M = \sqrt{x^2 + y^2}$$

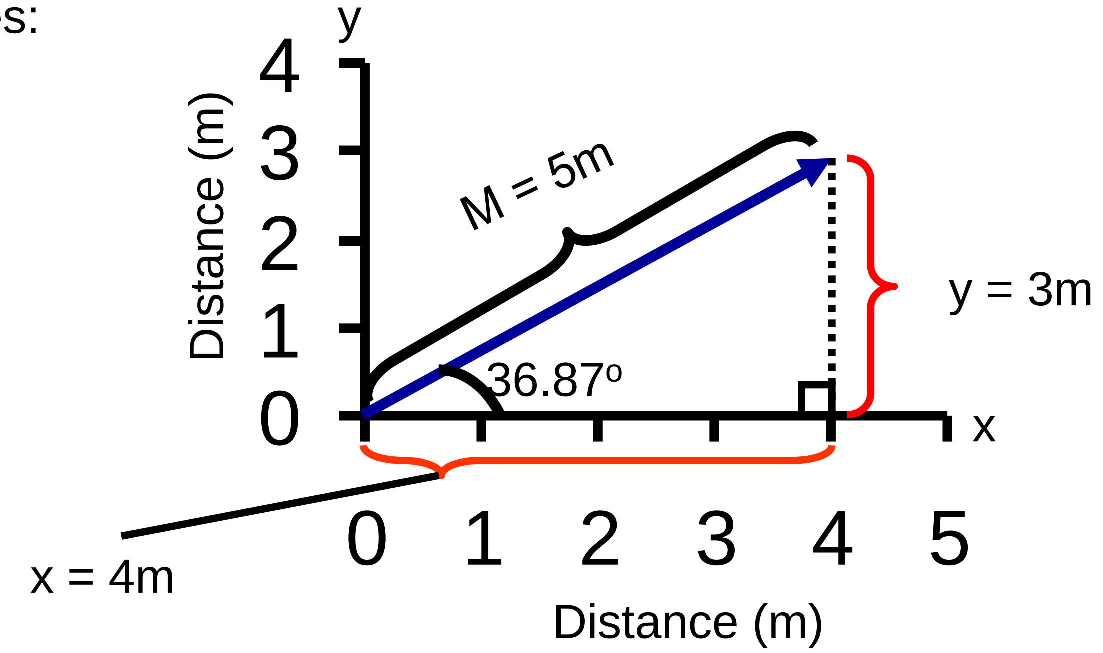
$$M = 5m$$

Using Trigonometric Identities:

$$\cos(\theta) = \frac{x}{M}$$

$$\theta = \arccos(\frac{x}{M})$$

$$\theta = 36.87^o$$



Cartesian to Polar Form

- In summary, to change Polar coordinates, M $\angle\theta$, into Cartesian coordinates (x,y) several trig identities can be used, including:

$$x = M\cos(\theta)$$

$$y = M\sin(\theta)$$

$$M = \sqrt{x^2 + y^2}$$

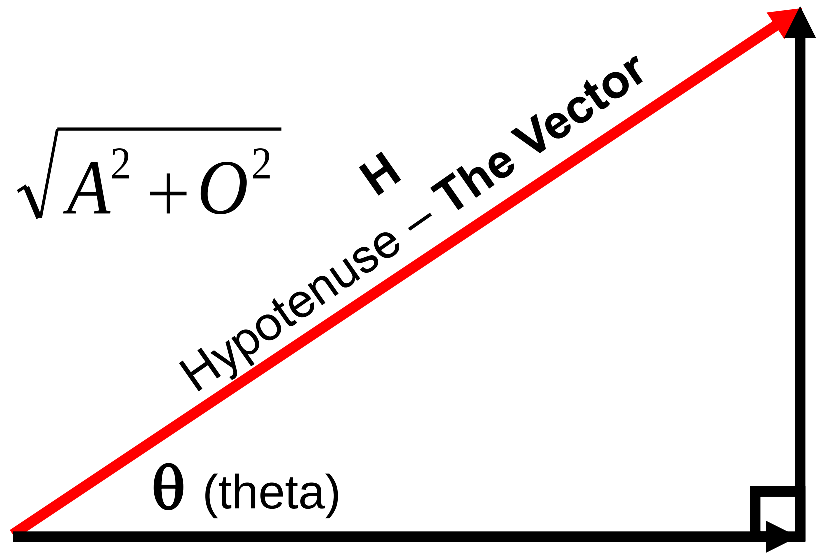
$$\theta = \arcsin(y / M)$$

$$\theta = \arccos(x / M)$$

What do all these triangles mean?

- HCos(θ) gives us the length of the projection of a vector of length H onto an axis that is at an angle of θ to the vector.
- HSin(θ) gives us the length of the projection of the H length vector onto an axis that is **orthogonal** (at 90º) from the previously mentioned axis
- These projections give us the magnitude of the vector along the axes.

$$H = \sqrt{A^2 + O^2}$$



O

Side **opposite** to theta – **The part of the vector acting in the Y direction**

$$O = H\sin(\theta)$$

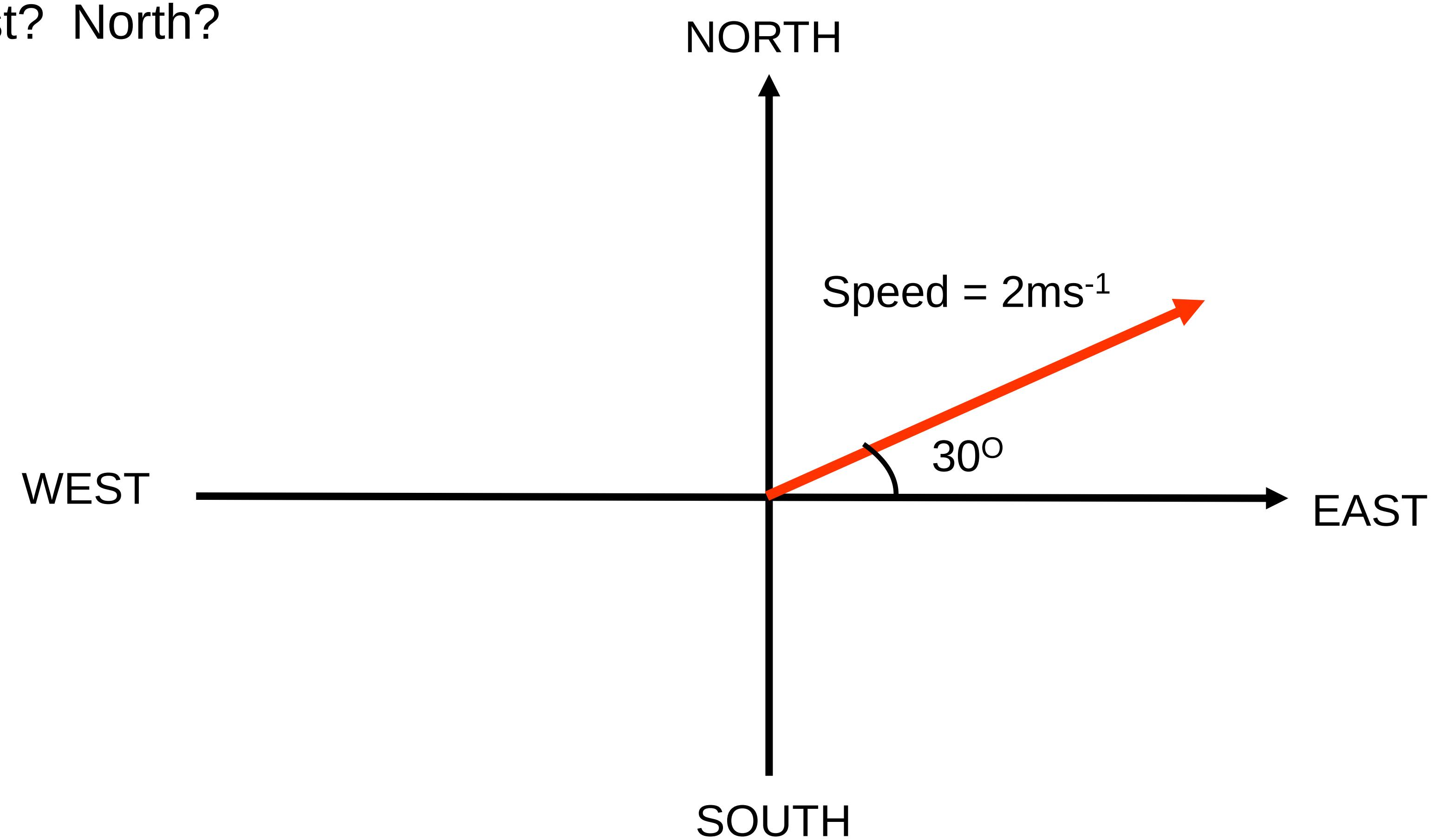
A

Side **adjacent** to theta – **The part of the vector acting in the X direction**

$$A = H\cos(\theta)$$

Example

- A frightened student is running away from the CSP2307 exam at a velocity of 2 metres per second in a direction that goes at 30º from the East-pointing axis of the equator. How fast is the student going East? North?

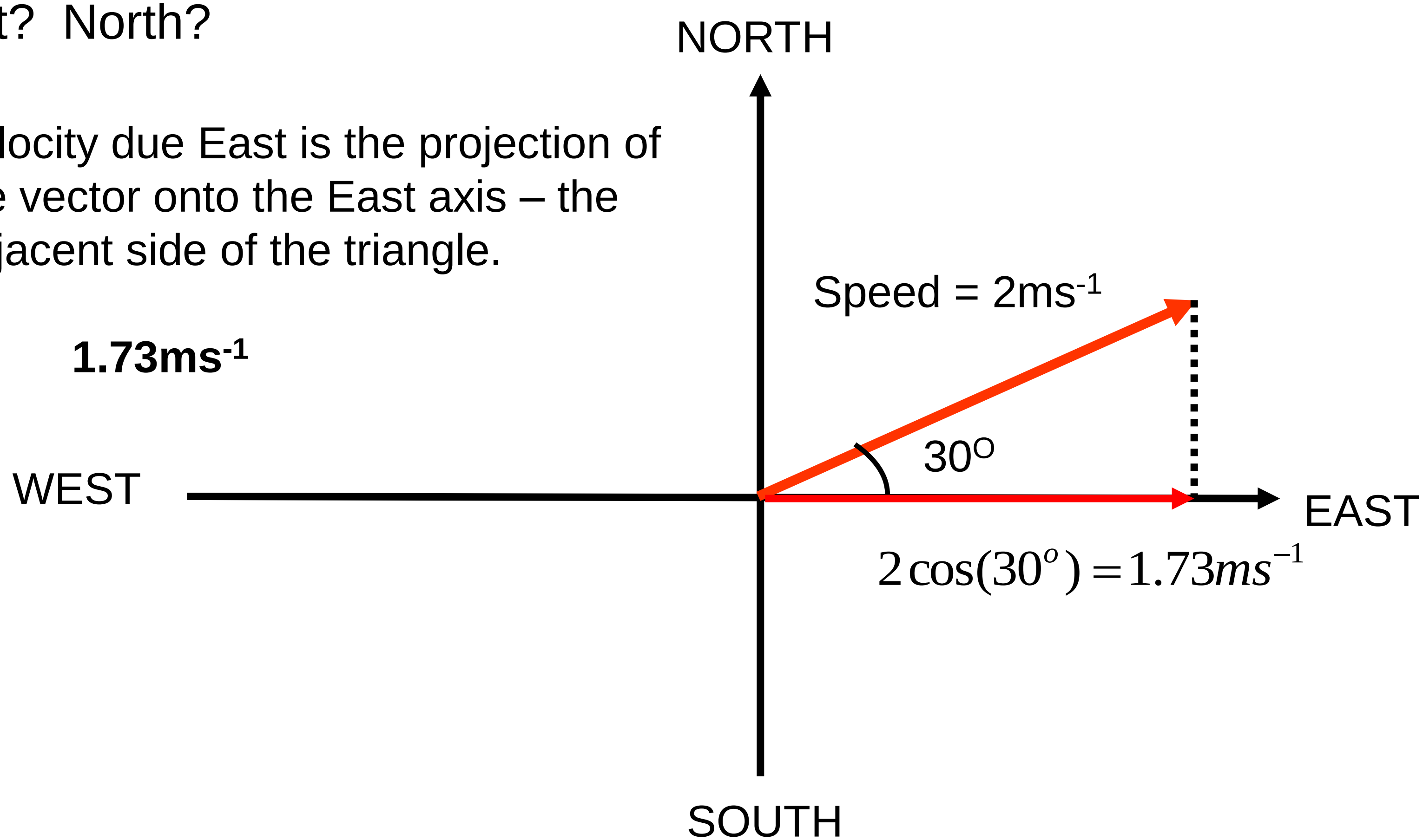


Example

- A frightened student is running away from the CSP2306 exam at a velocity of 2 metres per second in a direction that goes at 30° from the East-pointing axis of the equator. How fast is the student going East? North?

  - Velocity due East is the projection of the vector onto the East axis – the adjacent side of the triangle.

1.73ms^{-1}

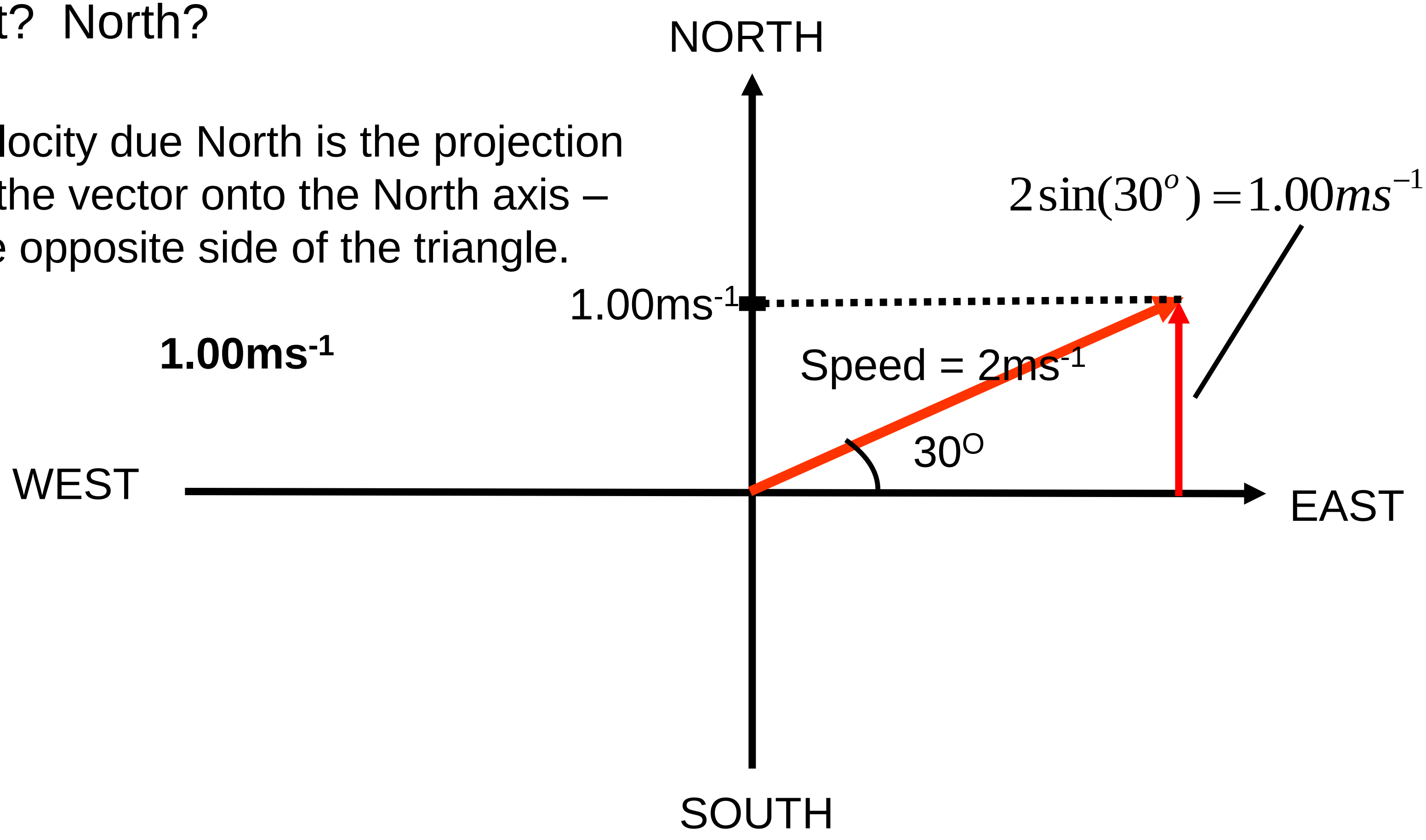


Example

- A frightened student is running away from the CSP2306 exam at a velocity of 2 metres per second in a direction that goes at 30º from the East-pointing axis of the equator. How fast is the student going East? North?

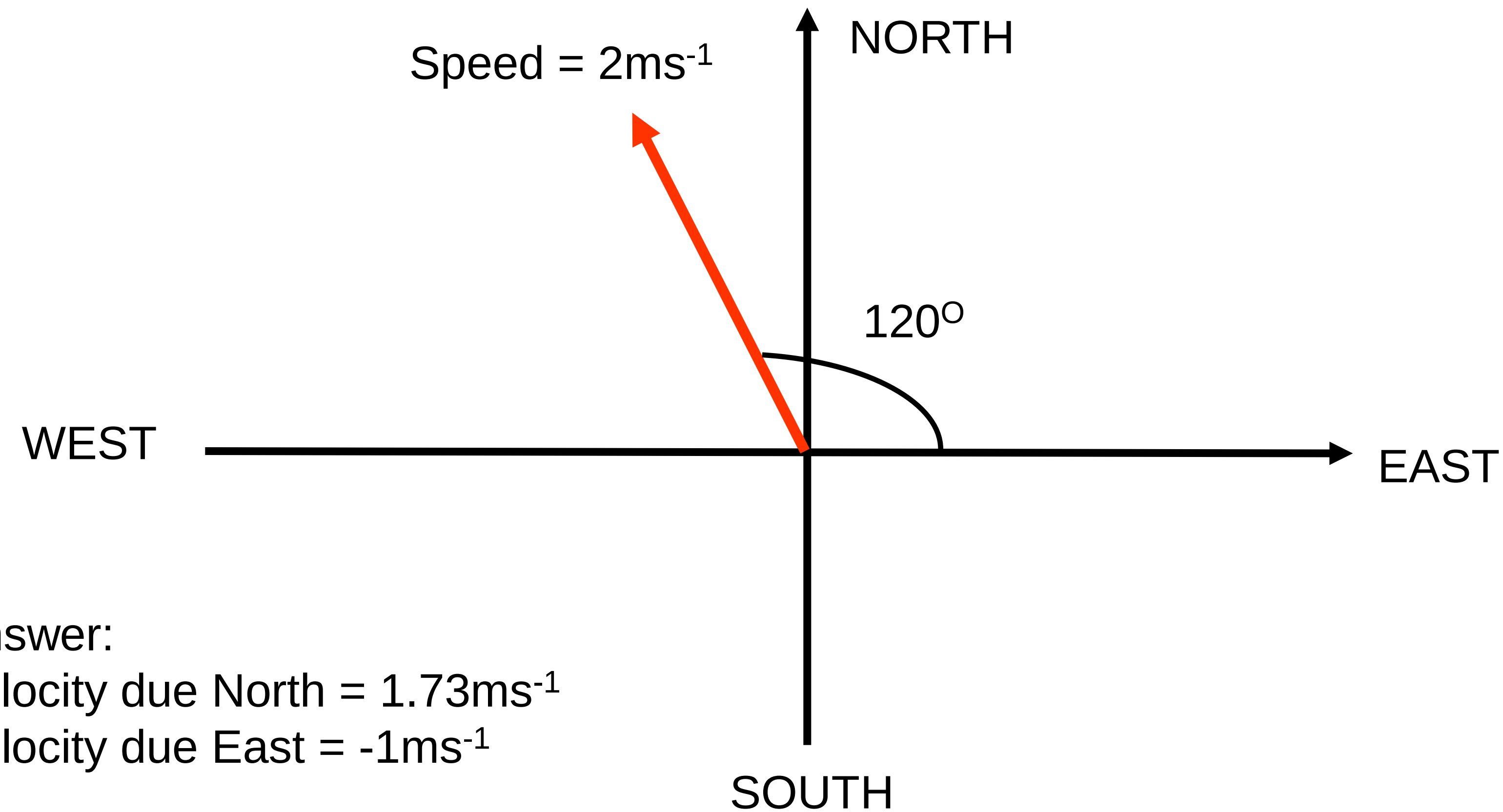
  - Velocity due North is the projection of the vector onto the North axis – the opposite side of the triangle.

1.00ms⁻¹



Exercise

- What if the student keeps the same speed, but changes direction to 120^{o} to the East axis. What is the velocity North, and East?



Answer:
Velocity due North = 1.73ms^{-1}
Velocity due East = -1ms^{-1}

Exercise

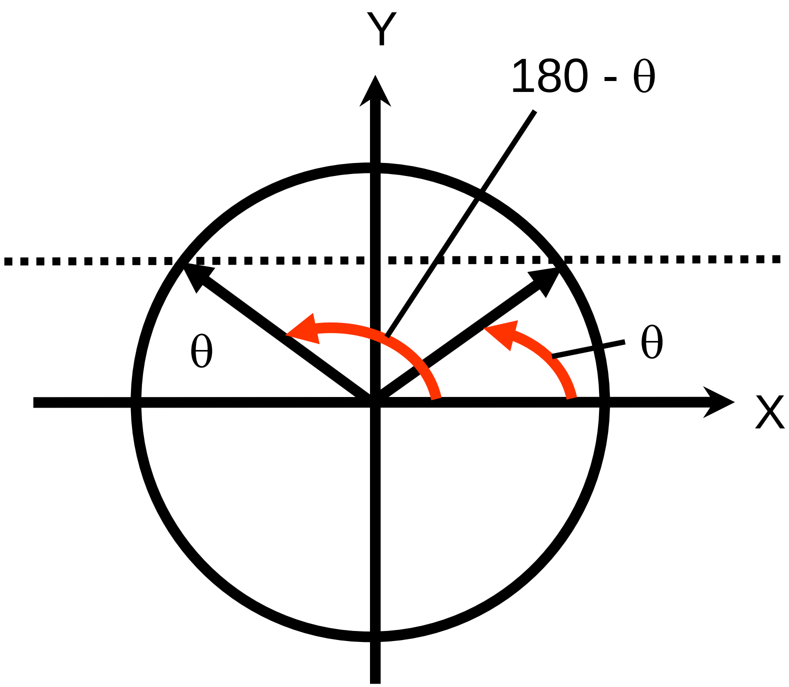
- An examiner is looking for students. Every second the examiner's vehicle travels -20 metres East and 30 metres North. What is the velocity vector of the vehicle in Cartesian and Polar coordinates?

  Answer:
  Cartesian Velocity vector (East, North) = (-20, 30) metres/s
  Polar Velocity vector (Magnitude, angle to East axis) = 36.06 m/s $\angle 123.69^{o}$

Cartesian to Polar Form

- Be careful with arcsin, arccos, and arctan:

$$\theta = \arcsin(y / M)$$

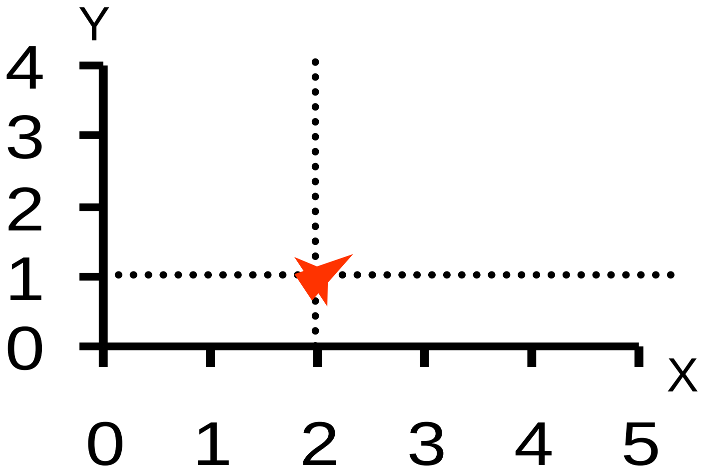


Two vectors, same y-coordinate, same magnitude, so same θ from arcsin, but different angle from origin! Why? (where is the triangle?).

Need to compensate for this by checking sign of x and y coordinate to determine the correct angle from origin.

Example

- The players character in a 2D game is at location (x,y) = (2,1), rotated at 35^o from the X-axis. The user presses the ‘up’ arrow once, wanting to move the character forward. Each press ‘up’ moves the character forward by 2 units. What is the characters new (x,y) position?



Example

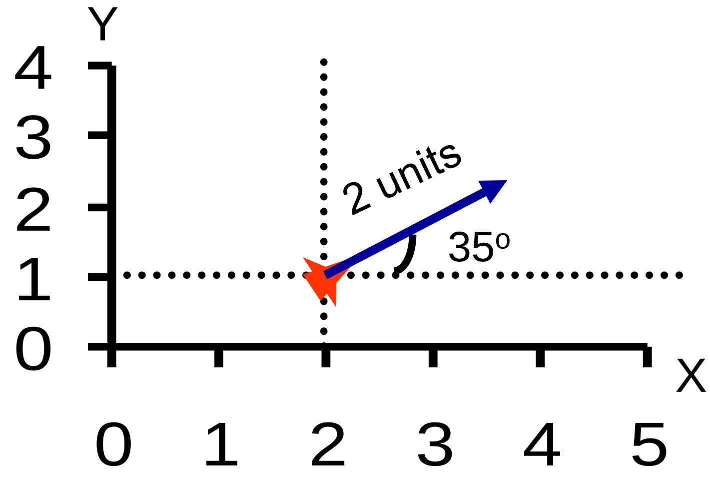
- The players character in a 2D game is at location (x,y) = (2,1), rotated at 35° from the X-axis. The user presses the 'up' arrow once wanting to move the character forward. Each press 'up' moves the character forward by 2 units. What is the characters new (x,y) position?

New position = Old position + Displacement

- Displacement (change in position) is a vector quantity $2\angle 35^o$ – magnitude = 2 units, direction = 35° to X-axis.
- Displacement is in polar form – must convert to Cartesian

$$x = M\cos(\theta)$$

$$y = M\sin(\theta)$$



Example

$$x = M\cos(\theta)$$

$$y = M\sin(\theta)$$

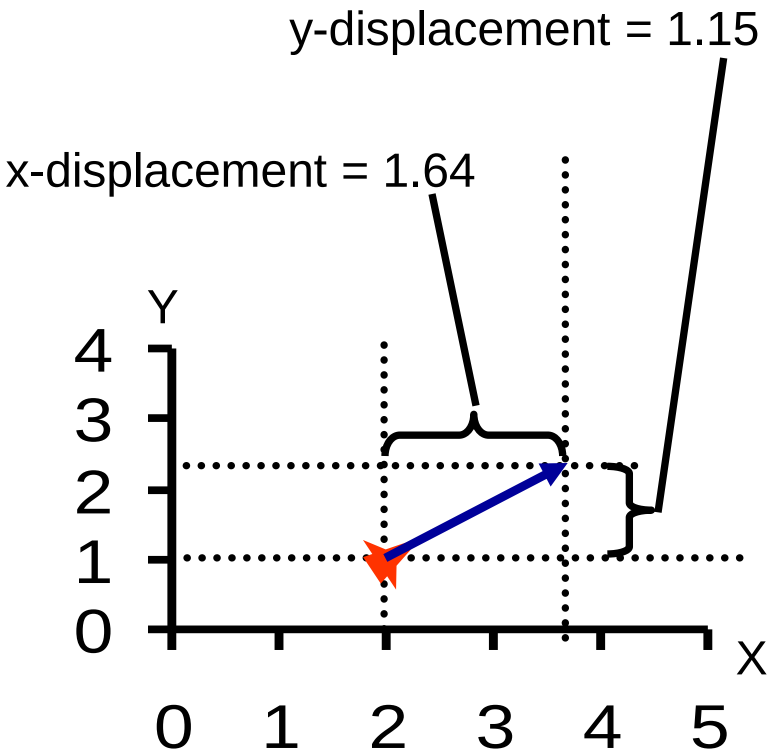
So, Cartesian form displacement is:

- x = 2 cos(35º), y = 2sin(35º)
- (x,y) = (1.64, 1.15)

New Position = Old Position + Displacement

= (2, 1) + (1.64, 1.15)

= (3.64, 2.15)



Example

$$x = M\cos(\theta)$$

$$y = M\sin(\theta)$$

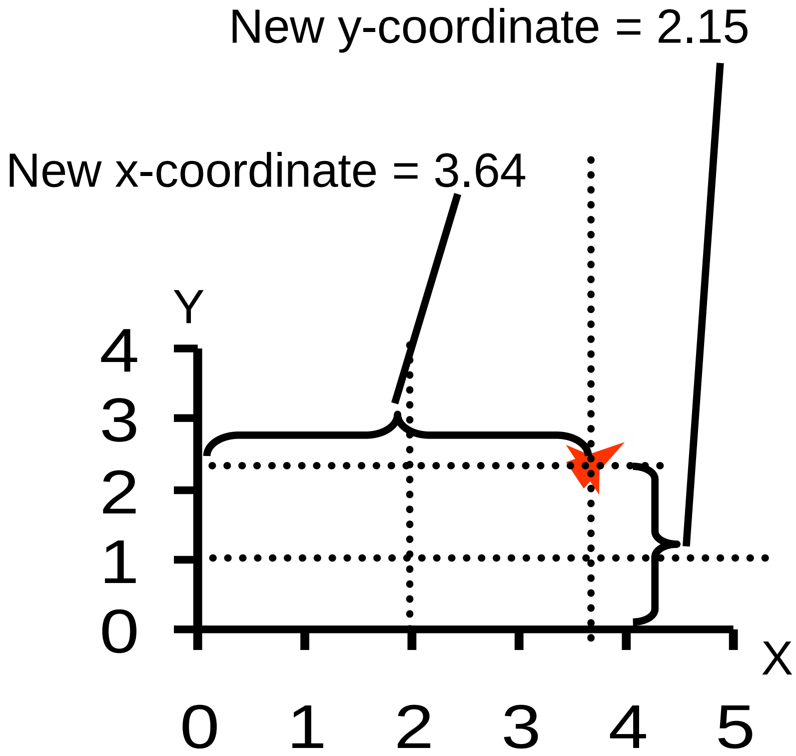
So, Cartesian form displacement is:

- $x = 2\cos(35^o)$, $y = 2\sin(35^o)$
- $(x,y) = (1.64, 1.15)$

New Position = Old Position + Displacement

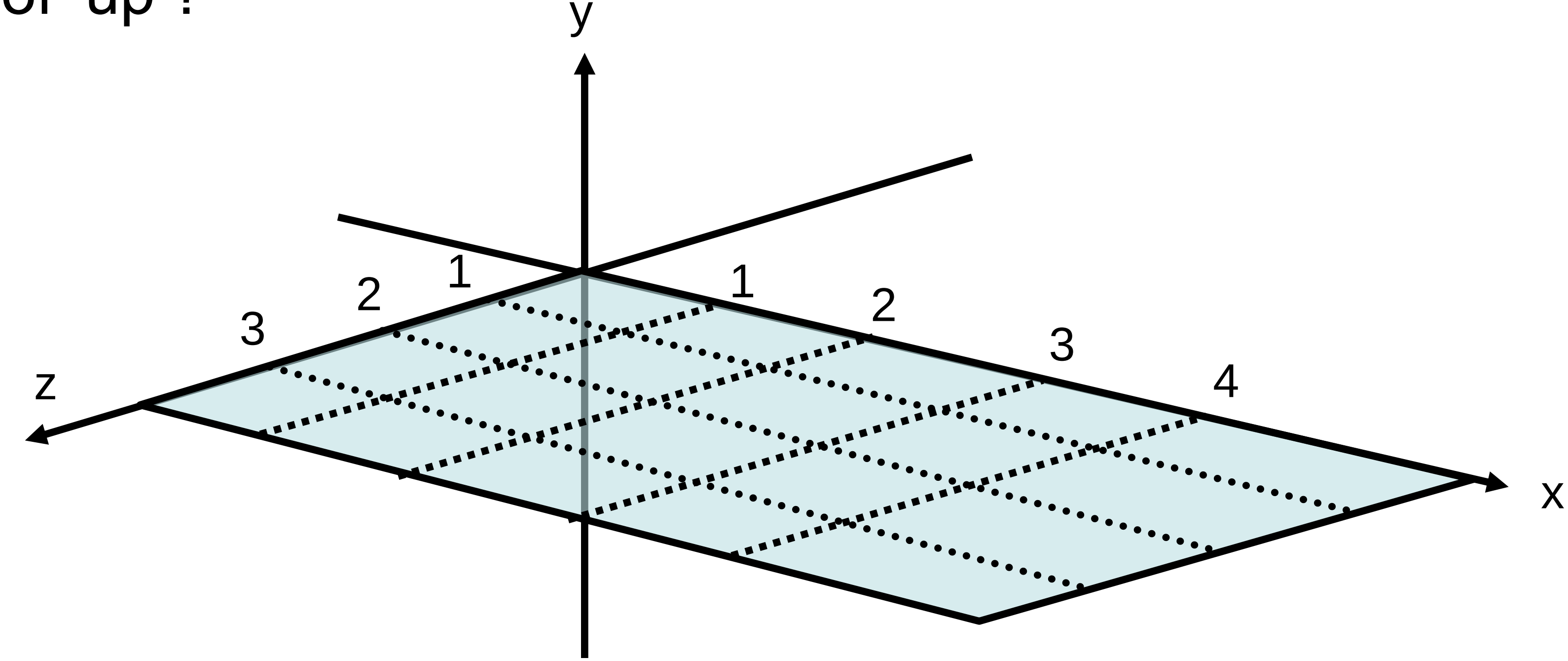
= (2, 1) + (1.64, 1.15)

= (3.64, 2.15)



Example 2

- A character in a 3D game moves along a 2D ground in the X-Z plane, The character is located at position (x_1,z_1), with angle away from the x-axis of θ^o. Each press of the ‘up’ arrow **displaces** the character M units forward. What is the new position (x_2, z_2)after N presses of ‘up’?



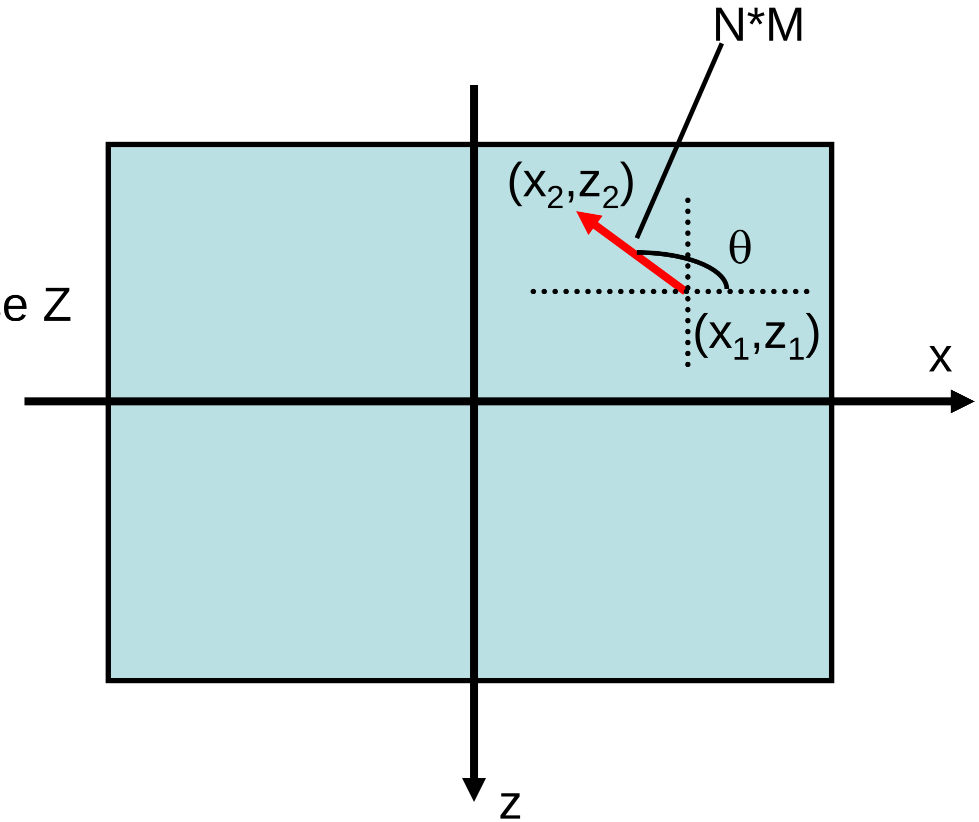
Example 2

Answer: Looking down at the X-Z plane

$$x_2 = x_1 + NM\cos(\theta)$$

$$z_2 = z_1 - NM\sin(\theta)$$

- –ve in the expression for z_2 because Z increases down the axis



Advantage of Cartesian Vectors

- The advantage of Cartesian form is that the vector is split into individual components along each axis.
- The components can be treated independently, reducing a problem with N-Dimensional vectors to N 1-Dimentional problems.

Difference Between Points and Vectors

Vectors

- Vectors are used to represent magnitude and direction of some property – **but not location**.
- Represented using Cartesian coordinates (x,y,z), to mean (Δx, Δy, Δz)

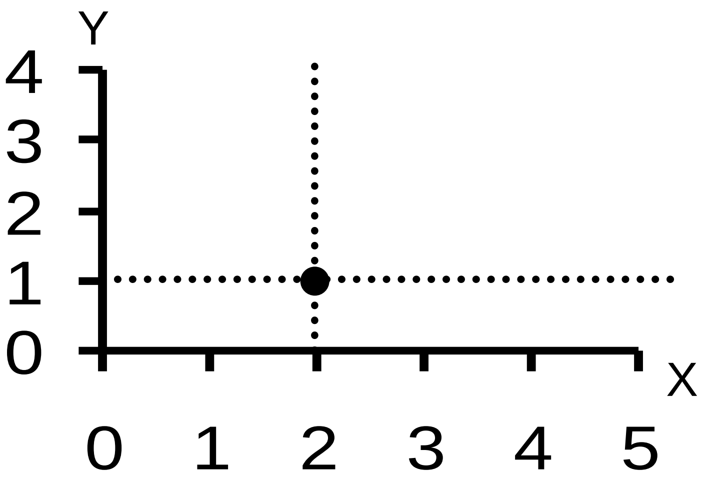
Points

- Points are physical locations in space (2-D, 3-D, N-D).
- In creating a vector class to hold three numbers (X, Y and Z) we can use the same class to represent a **point** in three dimensional Cartesian space. In doing this, some operations defined on vectors will not make sense for points.
  - eg. adding two displacement vectors gives total displacement. Adding two vertex locations of a cube gives us ????

Difference Between Points and Vectors

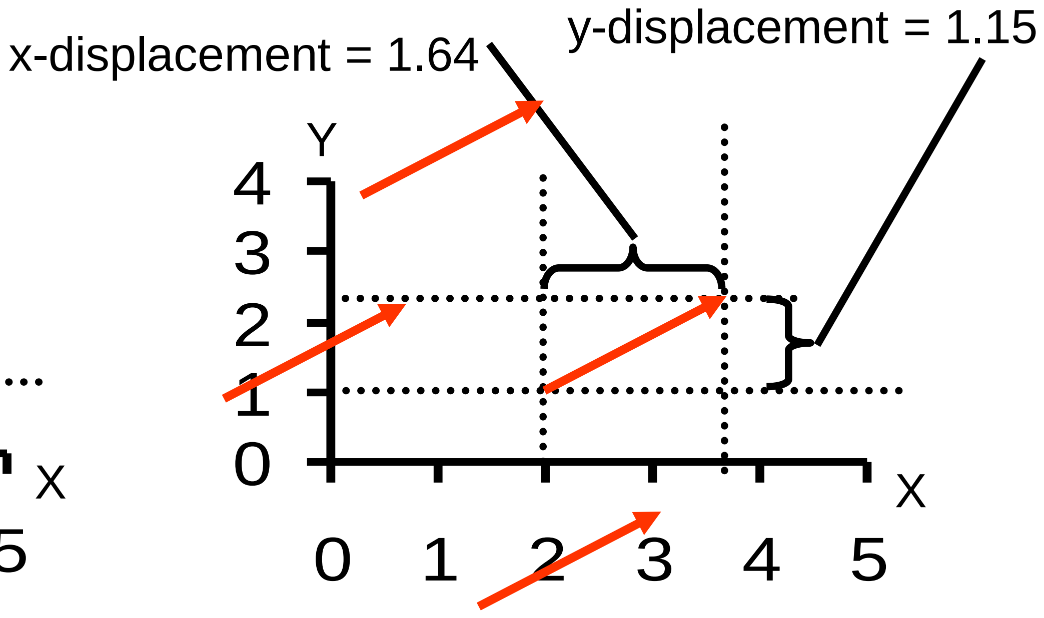
Point

- Unique location in space
- This point is (2,1)



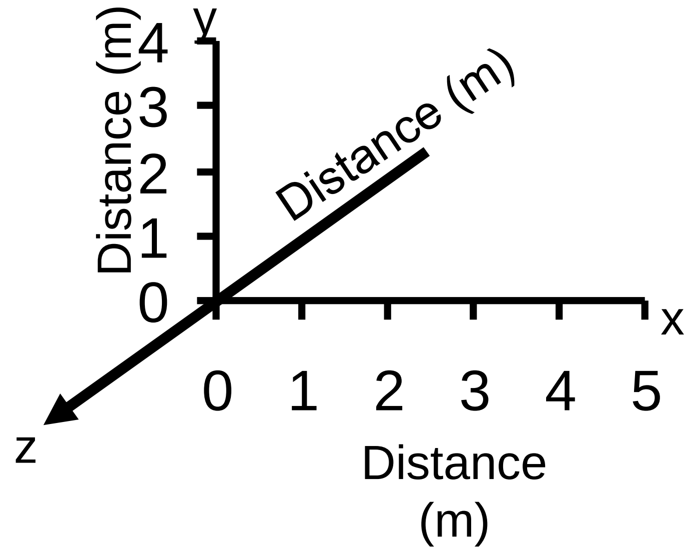
Vector

- Magnitude and direction of something.
- Does not represent location, we put it where its convenient for calculation (as in previous examples).
- All of these are (1.64, 1.15)



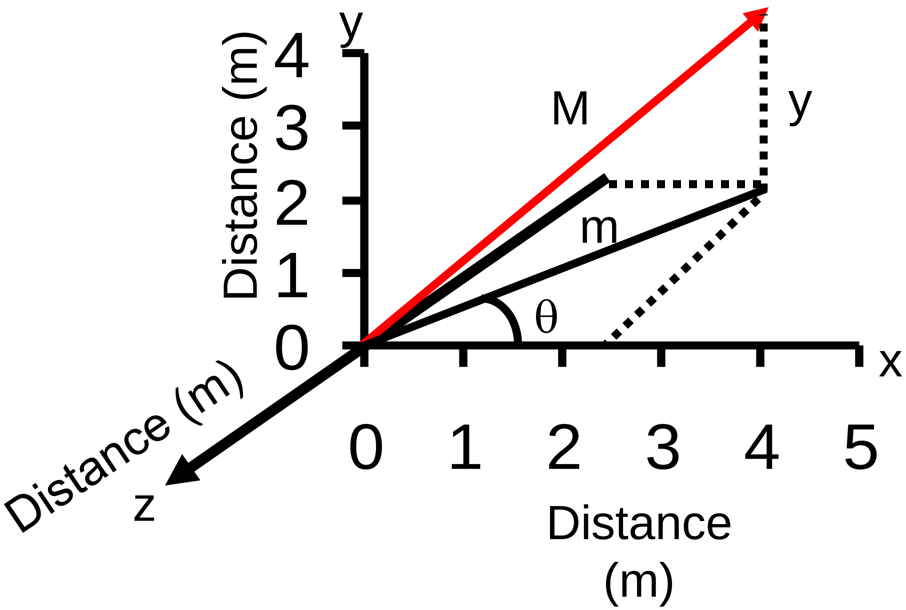
Vectors in 3D

- In 3D – add another axis
  - Cartesian form becomes (x,y,z).
  - Polar form becomes more complicated
    - Cylindrical coordinates
    - Spherical coordinates



Cylindrical Coordinates

- Keep polar coordinates for one axis plane (eg. x-z plane),
  - $m \angle \theta$
- Add a Cartesian coordinate for the remaining axis (eg. y-axis).
  - y



NOTE: m here is not the vector magnitude, its only the magnitude along the x-z plane

Cylindrical Coordinates

Cylindrical to Cartesian

$$x = m\cos(\theta)$$

$$z = m\sin(\theta)$$

$$y = y$$

Vector Magnitude

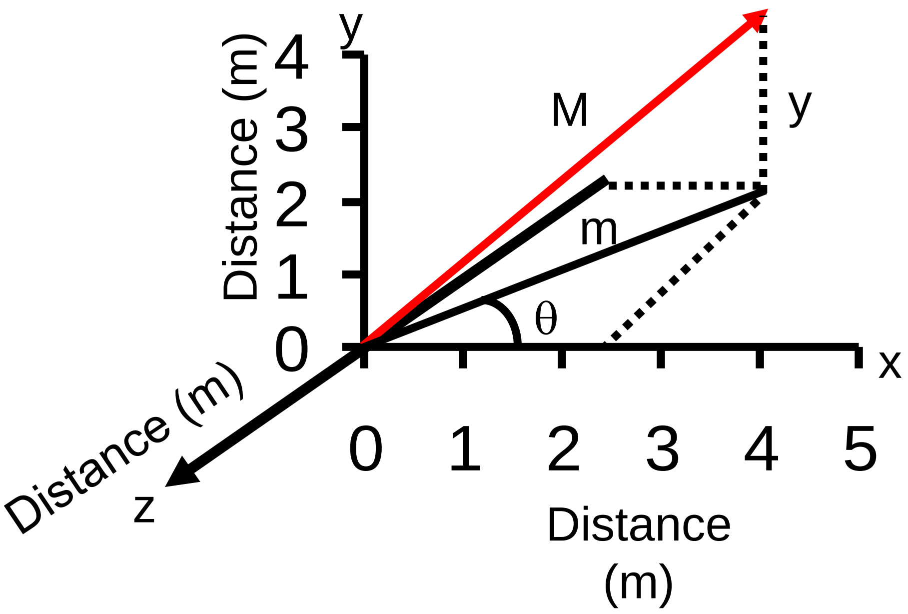
$$M = \sqrt{x^2 + y^2 + z^2}$$

Cartesian to Cylindrical

$$m = \sqrt{x^2 + z^2}$$

$$\theta = \arcsin(z/m)$$

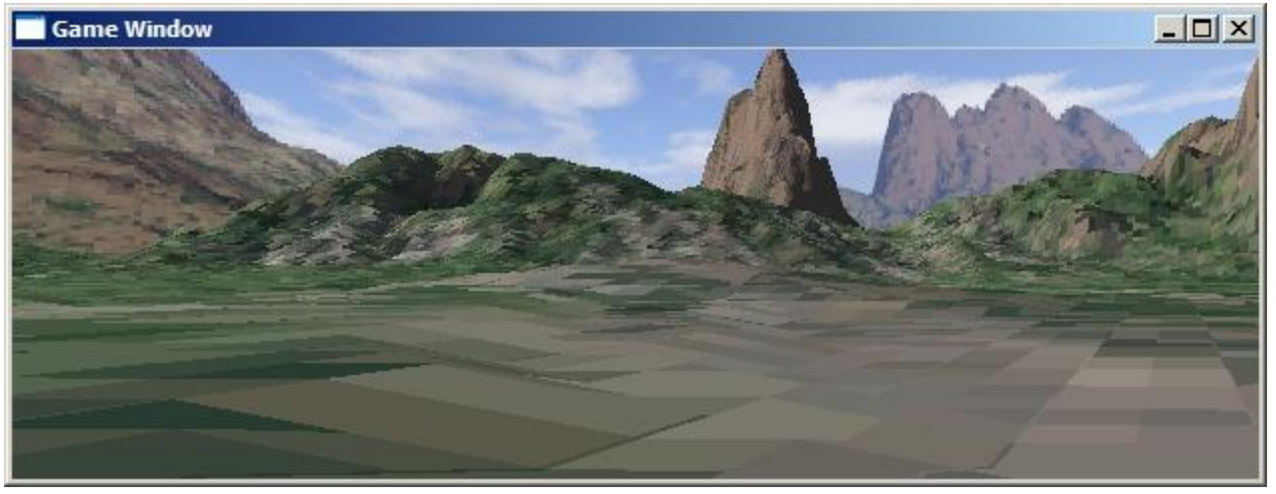
$$y = y$$



NOTE: m here is not the vector magnitude, its only the magnitude along the x-z plane

Cylindrical Coordinates

- Working in cylindrical coordinates is useful for games where the character travels along a mostly flat surface.
  - Keep a data structure (array, matrix…) of the height of the surface at each (X,Z) coordinate.
  - Keep track of the speed and angle of an object in the x-z plane.
  - For each movement determine the new Cartesian (X,Z) position from the polar speed-angle vector.
  - For the determined X,Z position, read in a height from the data structure to use as the Y coordinate.



Spherical Coordinates

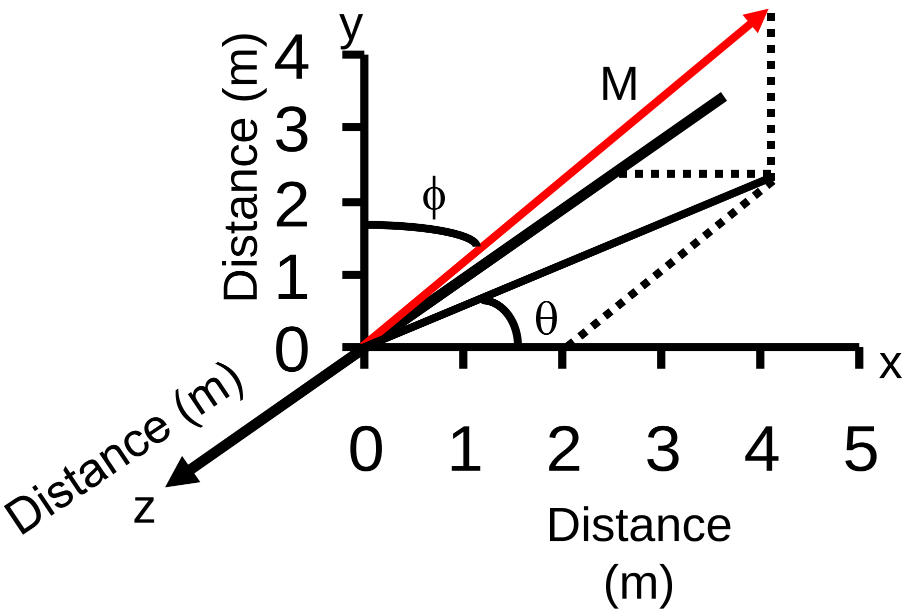
- Express the vector in terms of the magnitude, M, and two angles - M $\angle\theta$ $\angle\phi$
  - –Angle of the vectors projection on the x-z plane from the x-axis (θ).
  - –Angle of the vector from the y axis (ϕ).

Spherical to Cartesian

$$y = M\cos(\phi)$$

$$x = M\sin(\phi)\cos(\theta)$$

$$z = M\sin(\phi)\sin(\theta)$$



NOTE: M here is the vector magnitude

Spherical Coordinates

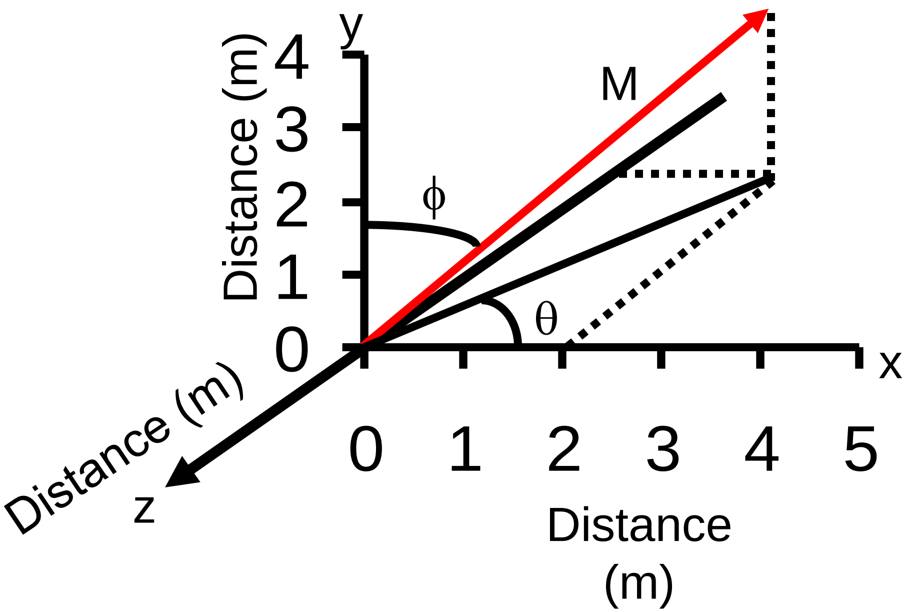
- Express the vector in terms of the magnitude, M, and two angles - M $\angle\theta$ $\angle\phi$
  - –Angle of the vectors projection on the x-z plane from the x-axis (θ).
  - –Angle of the vector from the y axis (ϕ).

Cartesian to Spherical

$$M = \sqrt{x^2 + y^2 + z^2}$$

$$\phi = \arccos(y/M)$$

$$\theta = \arccos(x/\sqrt{x^2 + y^2})$$



NOTE: M here is the vector magnitude

Operations on Vectors




Vector Norm (length, or magnitude)

- The magnitude of a vector is called its **norm**. We have seen norms already in our conversion from Cartesian to Polar/Spherical coordinates in 2D/3D
- Denoted in math books by double bracket $\|\mathbf{v}\|$
- Calculated using Pythagoras theorem – valid for vectors of any dimension n.

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

```
sum = 0;
for (i=0 to (n-1))
{
sum += v[i] * v[i];
}
length = sqrt(sum);
```

  - Equivalent to the vector Magnitude (M) we used before
  - Equivalent to the hypotenuse of the triangle (H) we used before
  - Squaring means we end up with a positive number
  - In the case of a three dimensional vector **v**.

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + v_3^2}$$

Vector Normalisation

- Unit vector
  - Any vector **v** whose **norm** (magnitude) $\|\mathbf{v}\|$ = 1
  - **Used when we are interested in direction only, not magnitude**
  - A length of exactly 1 is not always possible on vectors stored using IEEE floating point formats, but 32 bit floating point precision is sufficient for most purposes.
  - mathematically convenient for many operations

- Normalised vectors
  - A vector scaled to have a norm of 1 is said to be **normalised**
  - $\mathbf{v}_{\text{norm}} = \mathbf{v} / \|\mathbf{v}\|$
  - This does not allow for the zero vector to be normalised
  - If your code does not check for the zero vector before normalising a vector you will get a divide by zero error.

Norm, Normalised, Normal

- Can be confusing
  - **Norm** = length of a vector
  - **Normalised** vector = one whose Norm = 1
  - **Normal**: A vector is said to be normal to another vector or surface if it runs at 90^o to it (orthogonal).
    - Often convenient to normalise the normal to give it a norm of 1. ☺

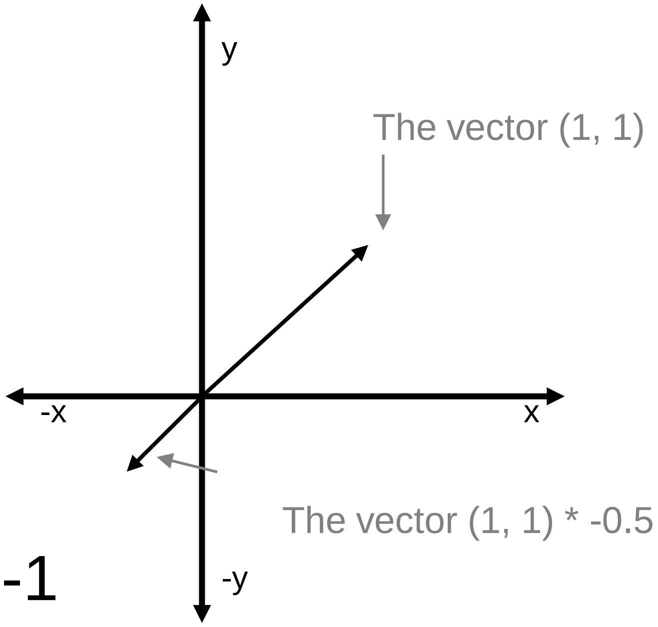
Multiplication by a Scalar

- Vectors can be scaled by a (scalar) value. Multiply all components by the scalar, producing another vector.
  - (x, y, z) * a = (x * a, y * a, z * a)
  - (x, y) / a = (x/a, y/a) = (x, y) * (1/a)

  eg,

  (5,10) * 5 = (25, 50)

  (-3, 8) * -2 = (6, -16)



- Vectors are negated by scaling of -1

  -**v** = **v** * -1
- Scaling changes magnitude, but not direction (direction can only be negated)

Multiplication by a Scalar

Application

- Momentum, **p**, (how hard it is to stop something) is defined as velocity * mass. If 'something' weighs 48Kg, and is travelling with velocity vector (3,4,5)m/s. What is the momentum vector?

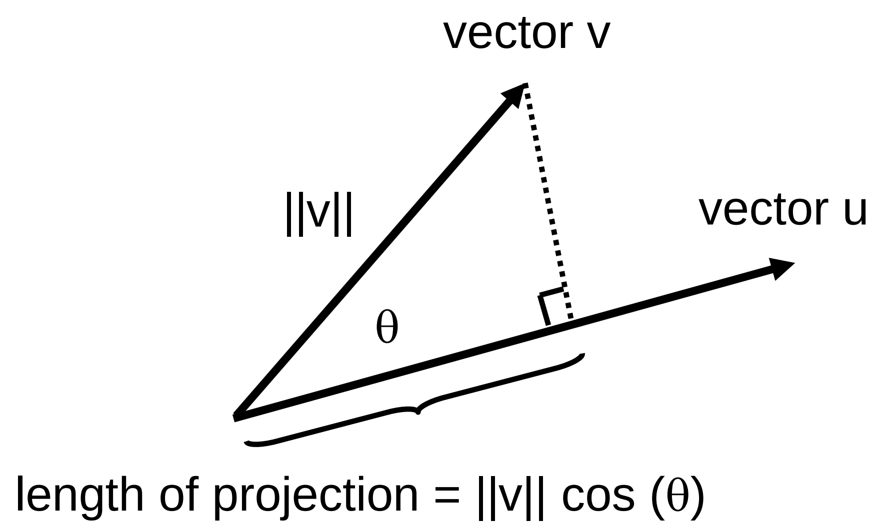
  p = (3*48, 4*48, 5*48)Kgm/s

  – Momentum is conserved in collisions (it gets transferred between objects so total momentum remains the same). This can be used to work out at what speed you fly back when such an object hits you.

Multiplication by a Scalar

Application

- From our discussion of triangles, we can work out that the **length (not a vector)** of the projection of a vector **v** onto vector **u** is given by the cos function, but what is the vector corresponding to this projection?
- The projection vector is in the same direction as vector **u**, with magnitude = $||v||\cos(\theta)$

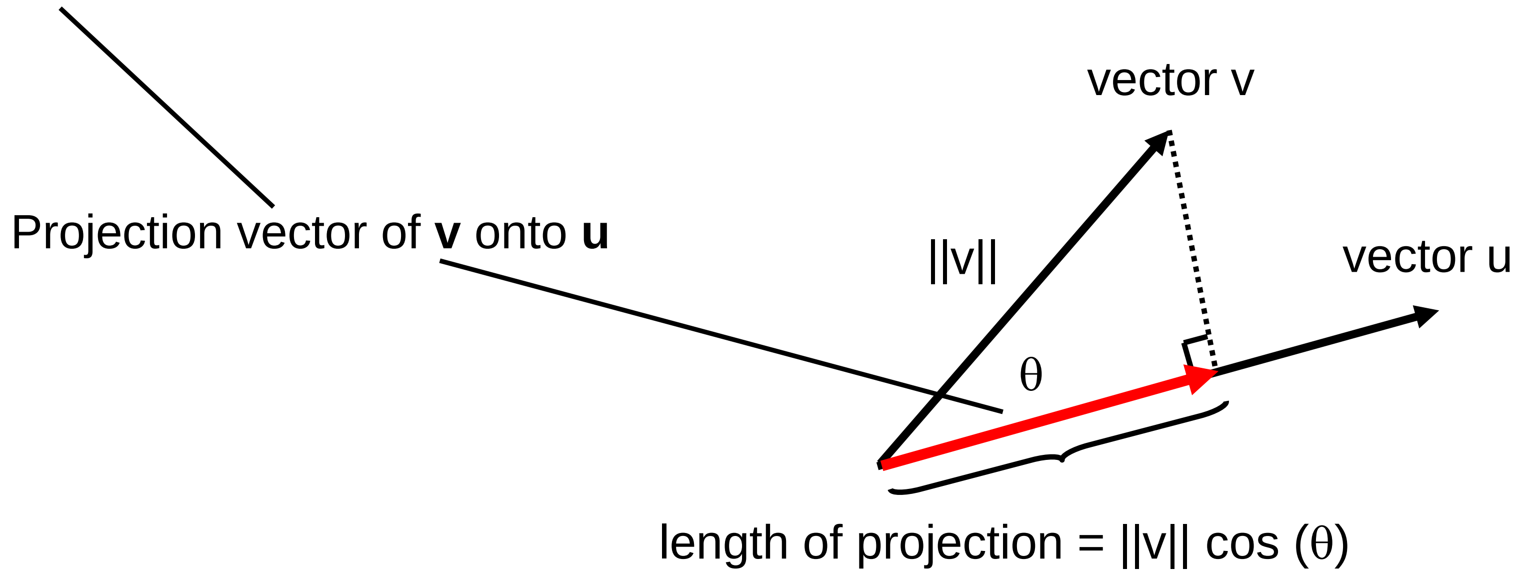


Multiplication by a Scalar

Solution

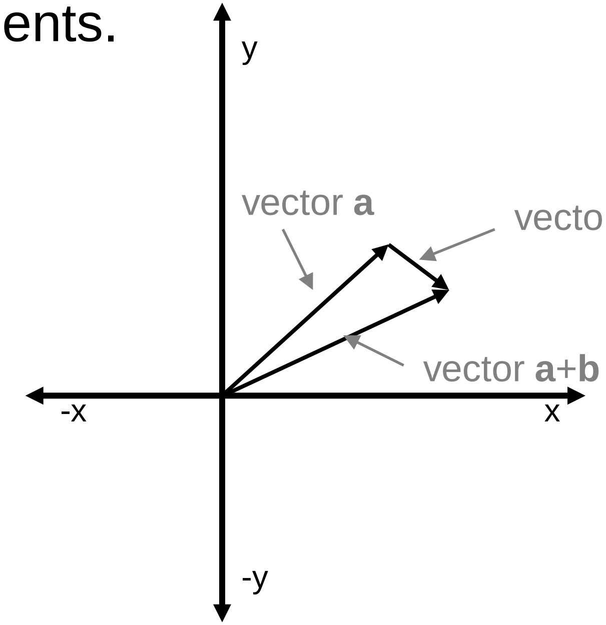
- Normalise vector **u** to give it a magnitude of 1
- Multiply the normalised **u** vector by the magnitude of the projection length to give it the length of $||v||\cos(\theta)$

- So $\mathrm{Proj}_{\mathbf{u}}\mathbf{v} = ||\mathbf{v}||\cos(\theta) * (\mathbf{u} / ||\mathbf{u}||)$



Vector Addition

- To add vectors, add individual components.
  - (x1, y1) + (x2, y2) = (x1+x2, y1+y2)
    - **v** + **w** = **w** + **v**



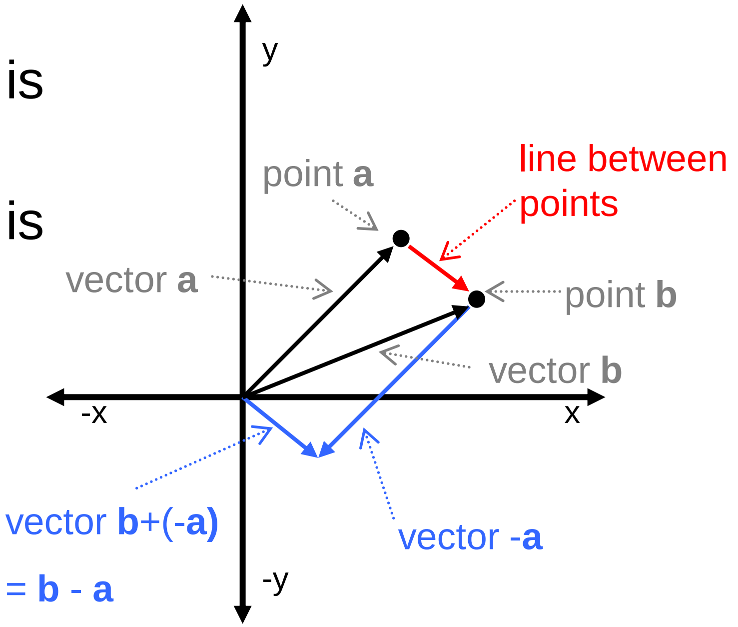
  - eg, an object is displaced, first by (5,10)m, then by (3,2)m. What is the total displacement from the objects starting point?

    (5, 10) + (3, 2) = (8, 12) metres
  - eg. Two forces are acting on an object (5, 10) N and (-3,-2) N, what is the total force?

    (5, 10) + (-3, -2) = (2, 8) Newtons

Vector Subtraction

- Use same function as addition – but means something different.
- (x1, y1) - (x2, y2) = (x1-x2, y1-y2)
    - **v** – **w** = -(**w** – **v**)
- Vector between two points
    - A vector from point a to point b is given by **b** – **a**.
    - A vector from point b to point a is given by **a** – **b**.



- This formula is extremely important when creating 3D graphics software!

Vector Subtraction Example

- In animation, Key Frames are used to outline the animation by specifying 'important' frames at certain points in time. The in-between frames are then filled in by:
  – Junior animators in the case of hand drawn animation.
  – A program in the case of digital animation.
- In one key frame, a vertex is at position **v** = (4,3), in the next key frame, 5 seconds later, the vertex is at position **w** = (5,2). What is the displacement vector, **d** ?
    - **d** = **w** – **v** = (1, -1)
- What is the average velocity vector between the two positions?
    - average velocity = displacement / time (this is division of vector by a scalar)
    - average velocity = (1/5, -1/5);
- Where should the vertex be 2 seconds after the first key frame assuming constant velocity between positions?
    - displacement = velocity * time
    - displacement = (2/5, -2/5)
    - newPosition = originalPosition + displacement
    - newPosition = (4+2/5, 3+ (-2/5))

Distance Between Two Points

- Calculating the distance between two points
  - Given the line between two points we can calculate the distance between two points by finding the length of that line

  - Distance between point a and b is the length of the vector from a to b.

$$\|(\mathbf{a} - \mathbf{b})\| = \|(\mathbf{b} - \mathbf{a})\|$$

  - By applying the vector length formula we derive

$$\|(\mathbf{a} - \mathbf{b})\| = \sqrt{\sum_{i=1}^{n} (\mathbf{a}_i - \mathbf{b}_i)^2}$$

Dot Product

- Also called inner product
- a Multiplication of two vectors giving a scalar – a single number.
- Denoted a "**.**", i.e. **a.b**
- the sum of the products of corresponding vector components.

Where vector **a** = (a_1, a_2,…, a_n), and vector **b** = (b_1, b_2,…, b_n)

a.b = ($a_1 \times b_1$) + ($a_2 \times b_2$) … + ($a_n \times b_n$)

or

$$\mathbf{a}.\mathbf{b} = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i$$

Dot Product

- The dot product is related to the angle between two vectors
- In three dimensions this angle is measured in the plane that contains both vectors

  – *Given vectors a and b, and θ as the angle between them*
  $\mathbf{a.b} = \|\mathbf{a}\| \times \|\mathbf{b}\| \times \cos\theta$

  or

  $\theta = acos\left(\dfrac{\boldsymbol{a.b}}{\|\mathbf{a}\| \times \|\mathbf{b}\|}\right)$ (this gives us the angle between two vectors)

Dot Product

- Quick information without using trig functions.

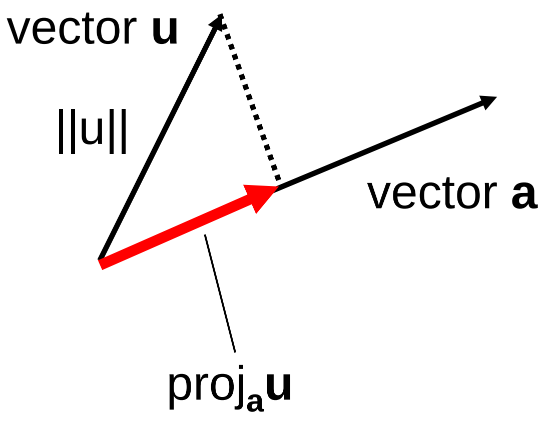
a.b	θ	Angle
< 0	$90° < \theta < 180°$	obtuse
0	$90°$	right
> 0	$0° < \theta < 90°$	acute

Dot Product – Properties

- The dot product is a great way to check if two vectors are **orthogonal** (perpendicular, angle between them is 90^o)
  - $\theta = 90^o$ if and only if $\mathbf{u.v} = 0$
- If **u**, **v**, and **w** are vectors and k is a scalar:
  - $\mathbf{u.v} = \mathbf{v.u}$
  - $\mathbf{u}.(\mathbf{v}+\mathbf{w}) = \mathbf{u}.\mathbf{v} + \mathbf{u}.\mathbf{w}$
  - $k(\mathbf{u.v}) = (k\mathbf{u}).\mathbf{v} = \mathbf{u}.(k\mathbf{v})$
  - $\mathbf{v.v} = 0$ if $\mathbf{v} = 0$, otherwise $\mathbf{v.v} > 0$

Dot Product - Projections

- The dot product can be used to obtain the projection of one vector onto another.
- Vector component of **u** along **a**:



$$proj_{\mathbf{a}}\mathbf{u} = \frac{\mathbf{u} \bullet \mathbf{a}}{\|\mathbf{a}\|^2}\mathbf{a}$$

- If **a** has previously been normalised (||a||=1) this becomes:

$$proj_{\mathbf{a}}\mathbf{u} = (\mathbf{u} \bullet \mathbf{a})\mathbf{a}$$

- From the diagram, component of u orthogonal to a (at 90^{o}) is **u** – proj$_{\mathbf{a}}$**u** (its just the vector between the two end points – see vector subtraction)

Dot Product - Projections

- Why is this useful? We can get the length of the projection (a scalar) using trigonometry as: $||\mathbf{u}||\cos(\theta)$, and get the vector corresponding to that length by multiplying it by the normalised **a** – which is what the equation on the previous slide is equivalent to.
- The nice thing about using the dot product is we can get the dot product, and hence the projection, using only the Cartesian vector components, without explicitly finding the angle between the vectors.
- Projections are useful for finding at what angle ‘things’ bounce off a surface after collision, things such as:
  - Rays of light – OpenGL works this out, all we specify is a normal to be projected onto.
  - Balls bouncing off a wall.

Cross Product

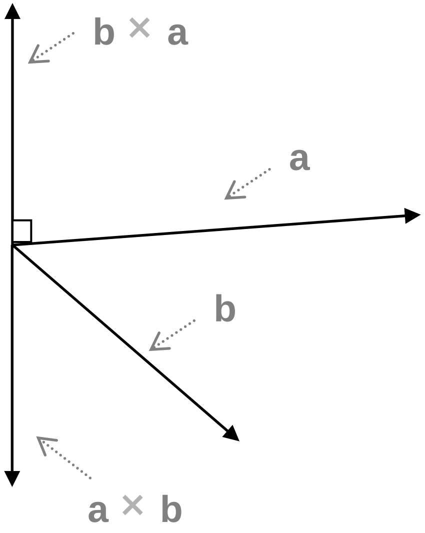
- Cross Product
  - Also called outer product or vector product
  - **Gives a vector perpendicular to two vectors**
  - Only valid for 3 dimensional vectors
  - Denoted as **a × b**
  - Orientation of the vector is determined by the right hand rule

Formula

- for **a** = (a_1, a_2, a_3) and **b** = (b_1, b_2, b_3)

a × b = ($a_2b_3 - a_3b_2$, $a_3b_1 - a_1b_3$, $a_1b_2 - a_2b_1$)

- this is a vector (3 components) orthogonal to the a-b plane.



Cross Product Properties

- Produces a vector orthogonal to both vectors so:
  - $\mathbf{u}.(\mathbf{u} \times \mathbf{v}) = 0$
  - $\mathbf{v}.(\mathbf{u} \times \mathbf{v}) = 0$

- The length of $\mathbf{a} \times \mathbf{b}$:

  $$\|\mathbf{a}\times\mathbf{b}\| = \|\mathbf{a}\| * \|\mathbf{b}\| * \sin\theta$$

- Identities of the Cross Product
  - $\mathbf{a}\times\mathbf{b} = -\mathbf{b}\times\mathbf{a}$
  - $\mathbf{a}\times(\mathbf{b}+\mathbf{c}) = \mathbf{a} \times \mathbf{b} + \mathbf{a} \times \mathbf{c}$
  - $(t\,\mathbf{a}) \times \mathbf{b} = t\,(\mathbf{a} \times \mathbf{b})$, where t is a scalar

Example

Find the surface normal to the x-y plane.

- A plane can be described by two vectors within it, lets just take two vectors along x-axis and y-axis:
  - u = (1,0,0), v = (0,5,0)
- A vector orthogonal to these (and to the plane) is given by **w**, where:
  - w = u × v
  - w = (0*0 – 0*5, 0*0 – 1*0, 1*5 – 0*0)
  - w = (0,0,5), this is the orthogonal vector (surface normal) – proves z-axis is at 90^{o} to both x and y axis.

Exercise

- Find the surface normal of a triangle described by 3 vertices v1 = (3,4,2), v2 = (3,5,6), and v3 = (5,3,1), and normalise it.
  - Form any two vectors given the triangle points:
    - eg. **u** = (v1 – v2), **v** = (v1-v3), or **w** = (v2-v3)
  - Perform the cross product, and divide by magnitude to normalise.

Answer:

- **u** = (0, -1, -4), **v** = (-2, 1, 1), **w** = (-2, 2, 5)
- u × v = (3, 8, -2)
- u × w = (3, 8, -2)
- v × w = (3, 8, -2)
- Normalised: $(3/\sqrt{77}, 8/\sqrt{77}, -2/\sqrt{77})$

Summary

- Vectors represent a magnitude and direction of something.
- Defined in Cartesian Coordinates as a list of numbers
- We have operations to:
  - Add Vectors
  - Multiply Vectors by a Scalar (a single number)
  - Dot Product
    - Find angle between two vectors in Cartesian form
    - Find the projection of one vector onto another
  - Cross Product
    - Find a vector perpendicular to a plane
    - Also can be used to find the angle between vectors.

Questions

- **In this lecture we have covered :**
  - Vector Operations
- **Extra reading:**
  - **2 and 3D vector section of a Linear Algebra book**

- Any Questions?

References

- 3D Math Primer For Graphics And Game Development. F. Dunn, I. Parberry
- Anton, H. and Rorres, C. (1991) Elementary Linear Algebra – Applications Version. 6th Ed. Singapore:Wiley & Sons.
- Beginning Math and Physics for Game Programmers – Wendy Stahler.