Normalization of vectors

A <u>unit</u> vector is vector w/ length one

$\boxed{\text{EX}}$ Show that the vector $v = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}, 0\right)^T$ is a unit vector.

Compute the length $\|v\| = 1$

$$\|v\| = \left\|\left(\frac{1}{2}, \frac{\sqrt{3}}{2}, 0\right)^T\right\| = \sqrt{\left(\frac{1}{2}\right)^2 + \left(\frac{\sqrt{3}}{2}\right)^2 + 0^2} = \sqrt{\frac{1}{4} + \frac{3}{4}} = \sqrt{4/4} = 1$$

Normalization

We can re-scale any vector to length 1 by **normalizing** it, e.g. dividing the vector by its length.

$$\text{normalize}(v) = \frac{v}{\|v\|}$$

v ← n-component vector

$\|v\|$ ← scalar, the length of v

$\boxed{\text{EX}}$ Normalize the vector $v = (4,3,0)^T$

① Compute the length of v

$\|v\| = \|(4,3,0)^T\| = \sqrt{16+9} = \sqrt{25} = 5$

② Divide each component of v by $\|v\|$: $(4/5, 3/5, 0)^T$

Aside: Using GLM

graphics library math

important to use floats!

0.0f

vec3(1.0f) ← create vector having all components = 1

length(v) ← returns the length (aka magnitude) of v.

watch out! DO NOT call v.length() ← always returns 3 for a vec3

cross(u, v) ← u × v

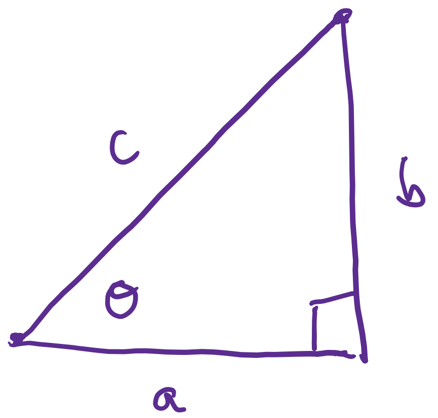
dot(u, v) ← u · v

normalize(v)

v − u

v + a

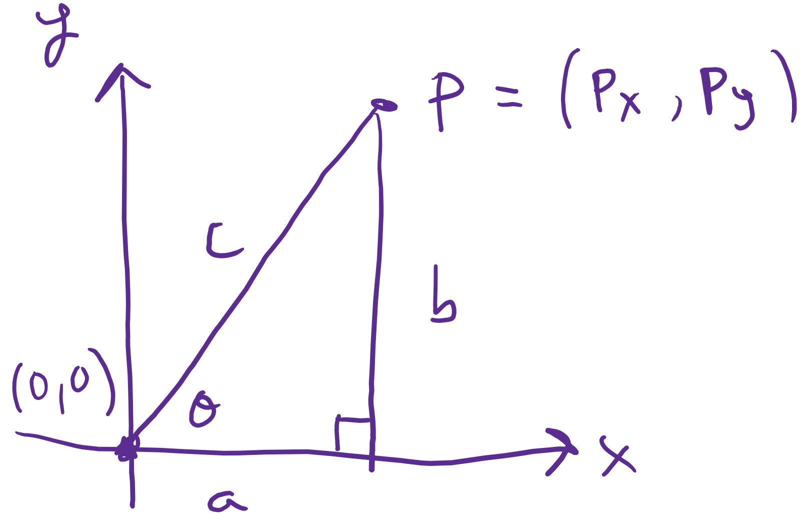
Review: Sin/Cos/Tan



$\sin\theta = b/c$

$\cos\theta = a/c$

$\tan\theta = b/a$



Pythagoream Rule: $a^2 + b^2 = c^2$

$P_x = a = c\cos\theta$

$P_y = b = c\sin\theta$

$c = \sqrt{a^2+b^2} = \sqrt{P_x^2 + P_y^2} = \|P\|$

Moving in a circle

Let's write an algorithm in terms of setup() & scene()

```
setup()
   theta = 0
   thetaRate = 0.1
   r = 250

scene()
   theta += thetaRate * dt()
   float px = r * cos(theta);
   float py = r * sin(theta);
   drawSphere(vec3(px, py, 0));
```

radians

Idea:

$p_x = r\cos\theta$

$p_y = r\sin\theta$

$p_z = 0$

formulas rotate around (0,0,0)

increase θ each frame:

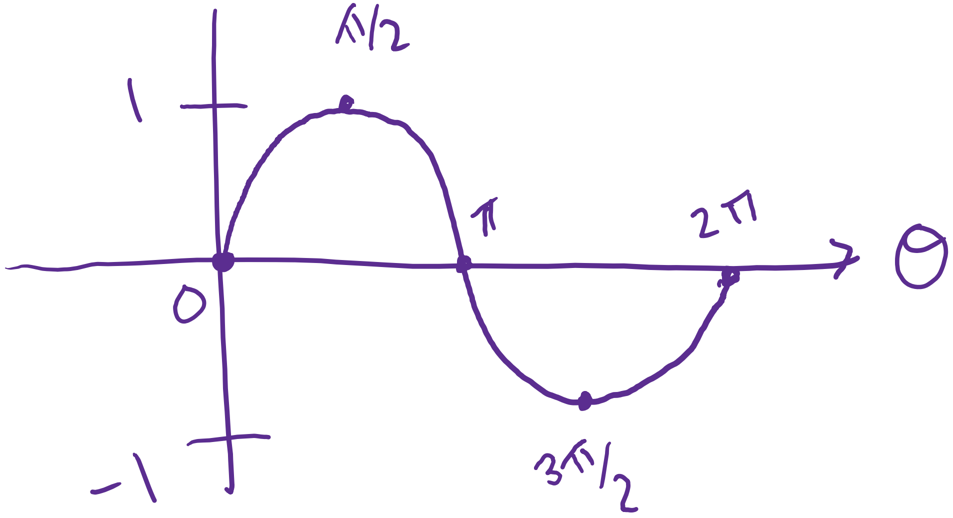
$\theta = \theta + dt() * \Delta\theta$

where $\Delta\theta$ = rate of rotation

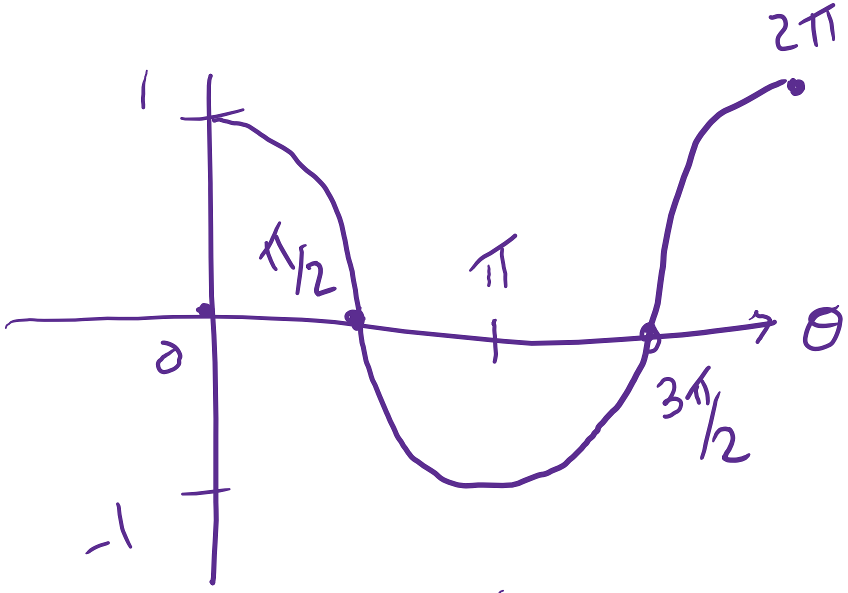
data
```
float theta;
float ThetaRate;
float r; // distance from (0,0,0)
```

Oscillating movement

Recall: the curves for sine & cosine



sine (θ)



Cosine (θ)

Oscillating movement

Let's animate the height of an object sine

Idea: each frame, update

$$y = 200 \sin(\text{theta})$$

$$x = 0$$

$$z = 0$$

setup:

```
thetaRate = 0.5
theta = 0
```

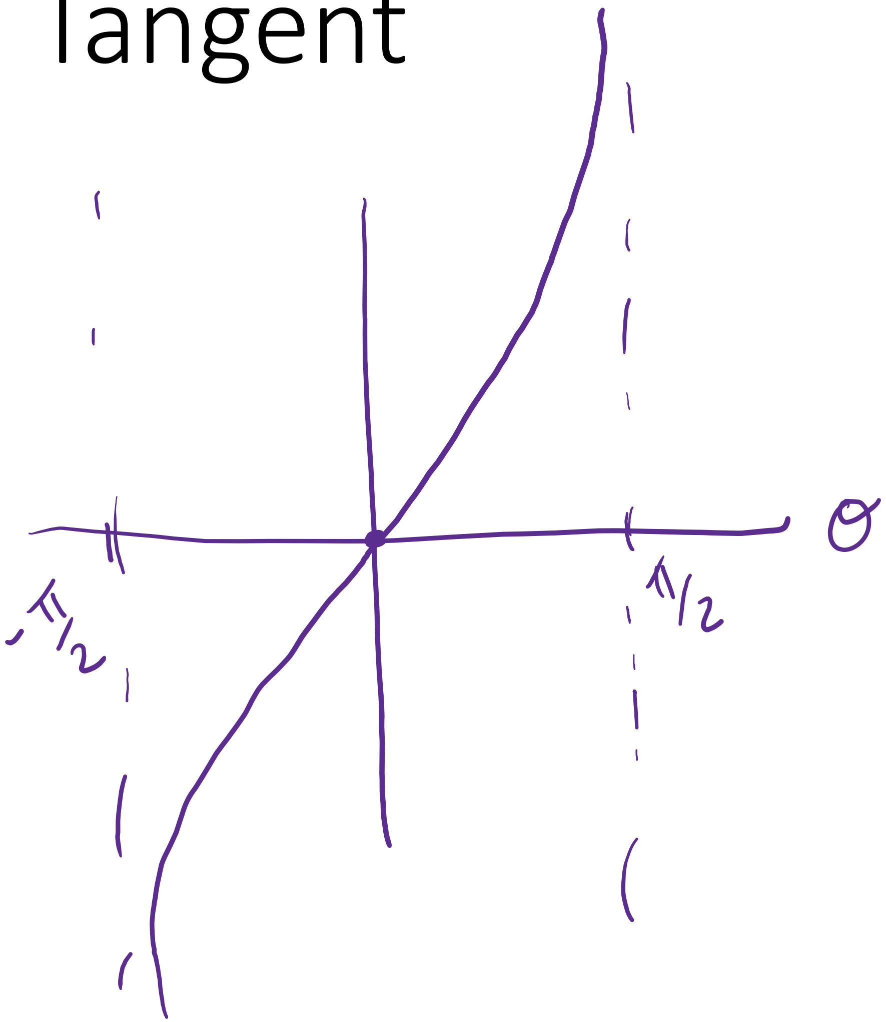
scene:

```
theta += thetaRate * dt()
float py = 200. * sin(theta);
float px = 0.0
drawSphere(vec3(px, py, 0), 100);
```

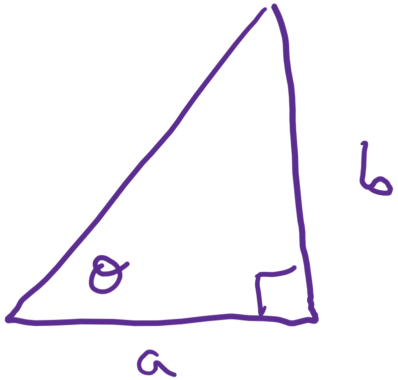
data

```
float theta
float thetaRate
```

Tangent



Recall: $\tan\theta = b/a$



Inverses of Sin/Cos/Tan

cos/sin take on angle $\in [-\infty, \infty]$ & outputs a values in range $[-1, 1]$

asin "arcsine" takes a value in range $[-1, 1]$ and returns an angle $\in [-\frac{\pi}{2}, \frac{\pi}{2}]$

e.g. has domain $[-1, 1]$

acos "arccosine" has domain $[-1, 1]$ & returns an angle $\in [0, 2\pi]$

watch out! these are undefined outside range $[-1, 1]$

→ NaN "not a number"

Arctangent

$\text{atan}(b/a)$

↳ outputs an angle in range $\in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$

domain is $[-\infty, \infty]$

better to use

$\text{atan2}(b, a)$

↳ outputs angle in range $\in [0, 2\pi]$

↳ uses the signs of a & b to deduce the quadrant of the angle