Game Physics in C#

Lecture 2 Tales of the Triangle

(aka Trigonometry)

Image: http://communities.ptc.com/people/DanMarotta

Lecture overview

- What is trigonometry?
- What do we use it for?
- Angles and polar coordinates
- Polar to cartesian mapping and back
- Practical applications
- Assignment 2

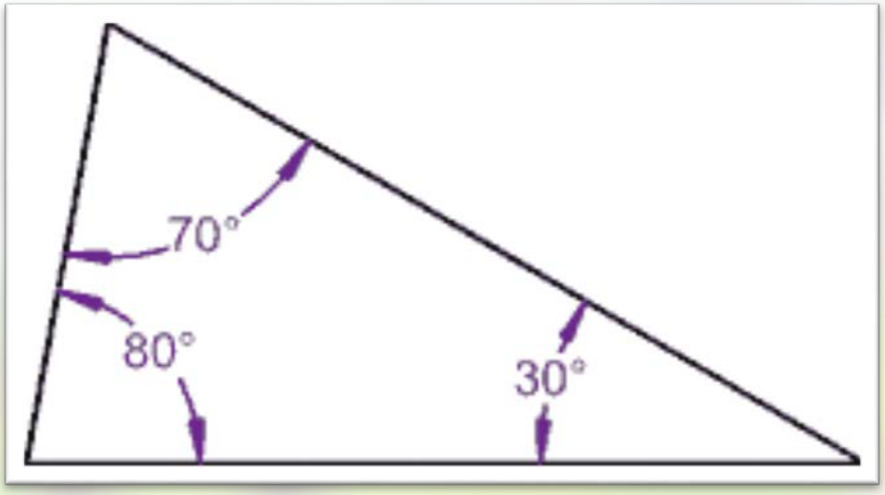
What is Trigonometry?

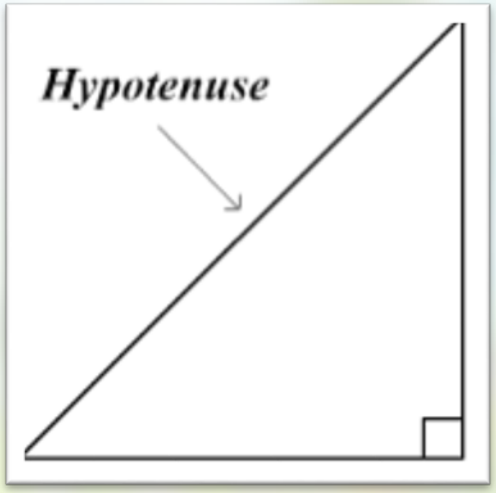
Tri – go? – No! – Me? – Tree!

Try repeating that 5 times really quickly!

Trigonometry

- Math that deals with:
  - triangles
  - ratio's between angles and side lengths in a triangle
  - mostly right-sided triangles





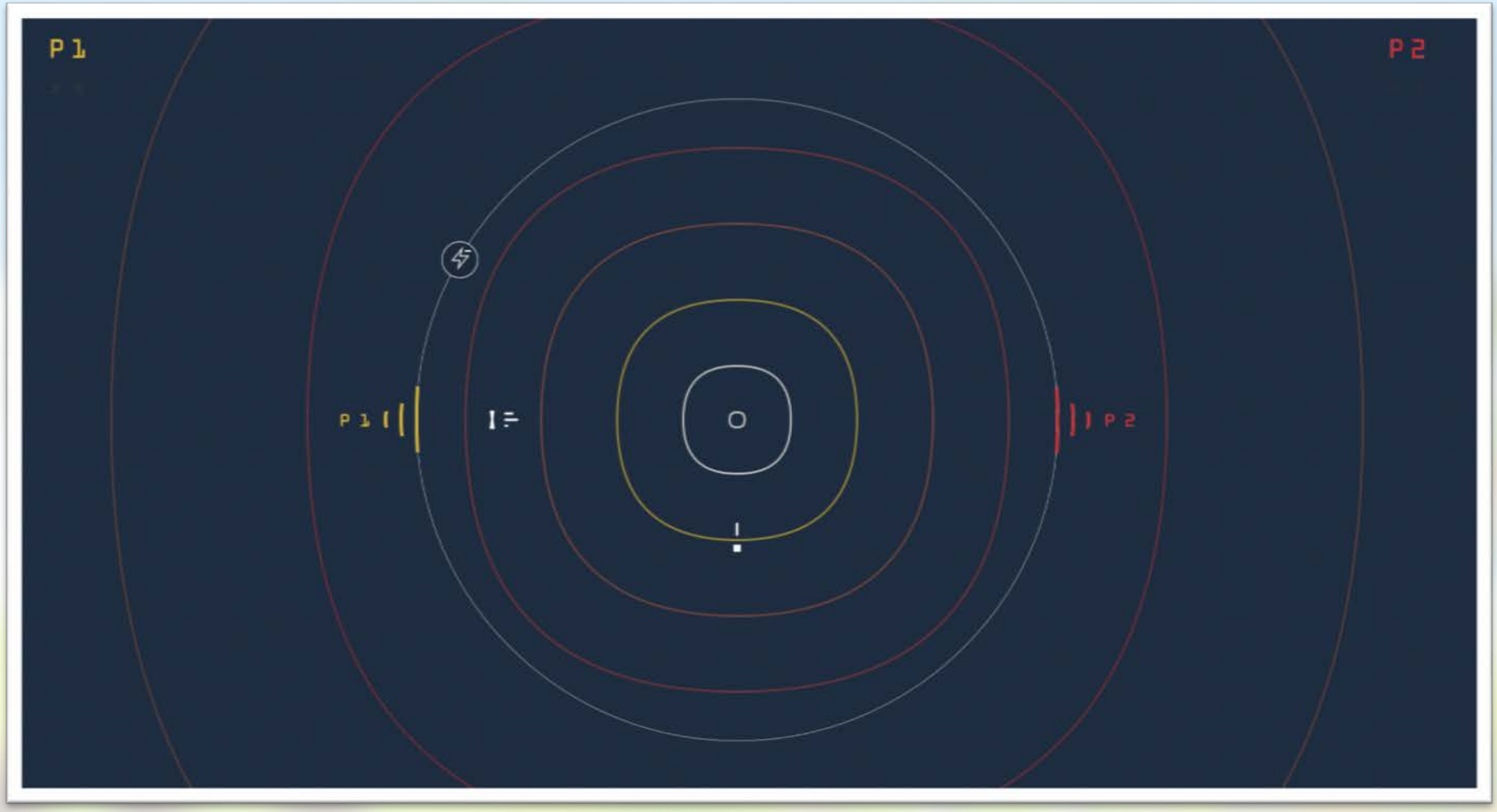
Practical Trigonometry

What can you do with it?

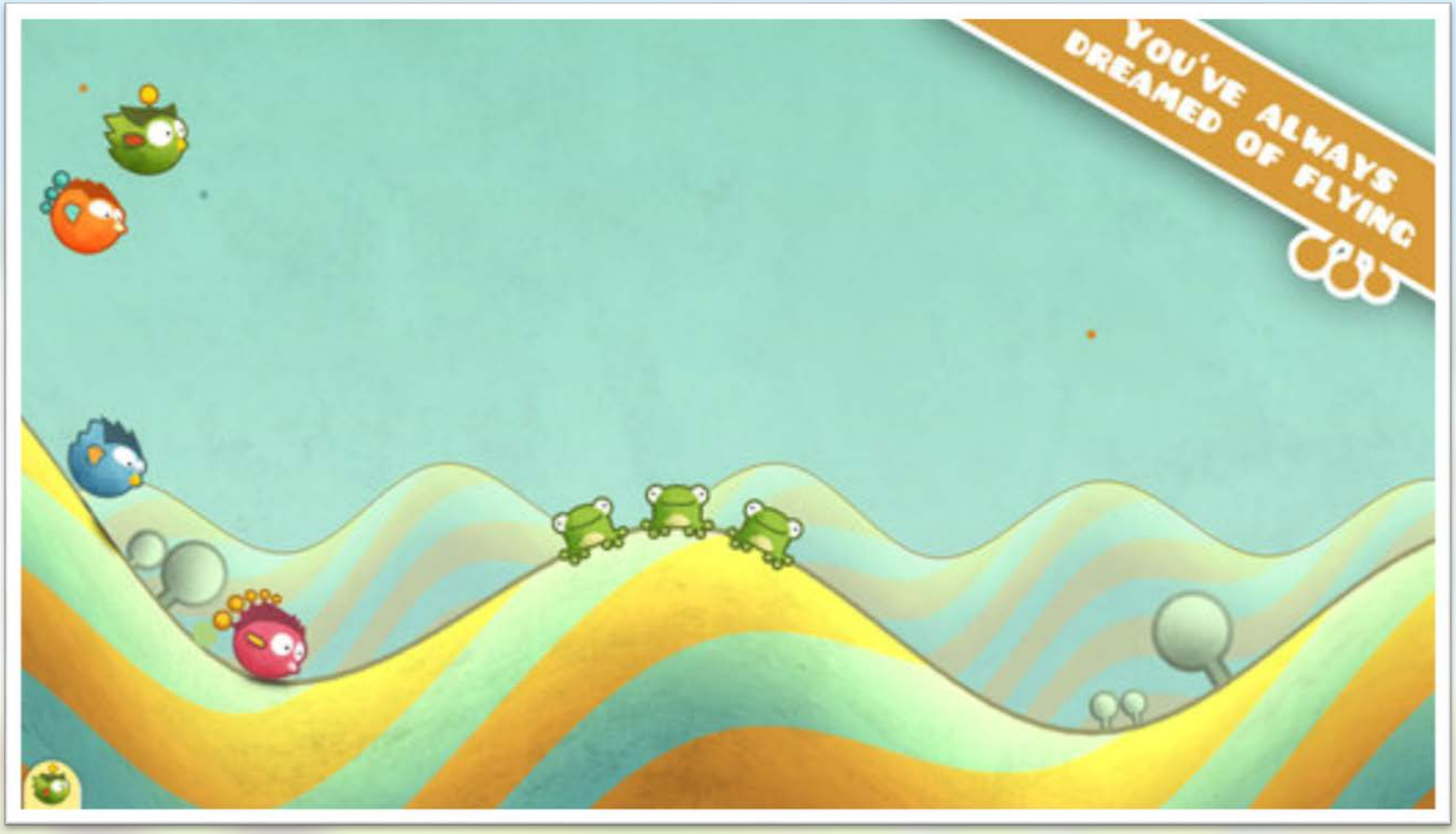
Move in a rotated direction

Rotate towards a target

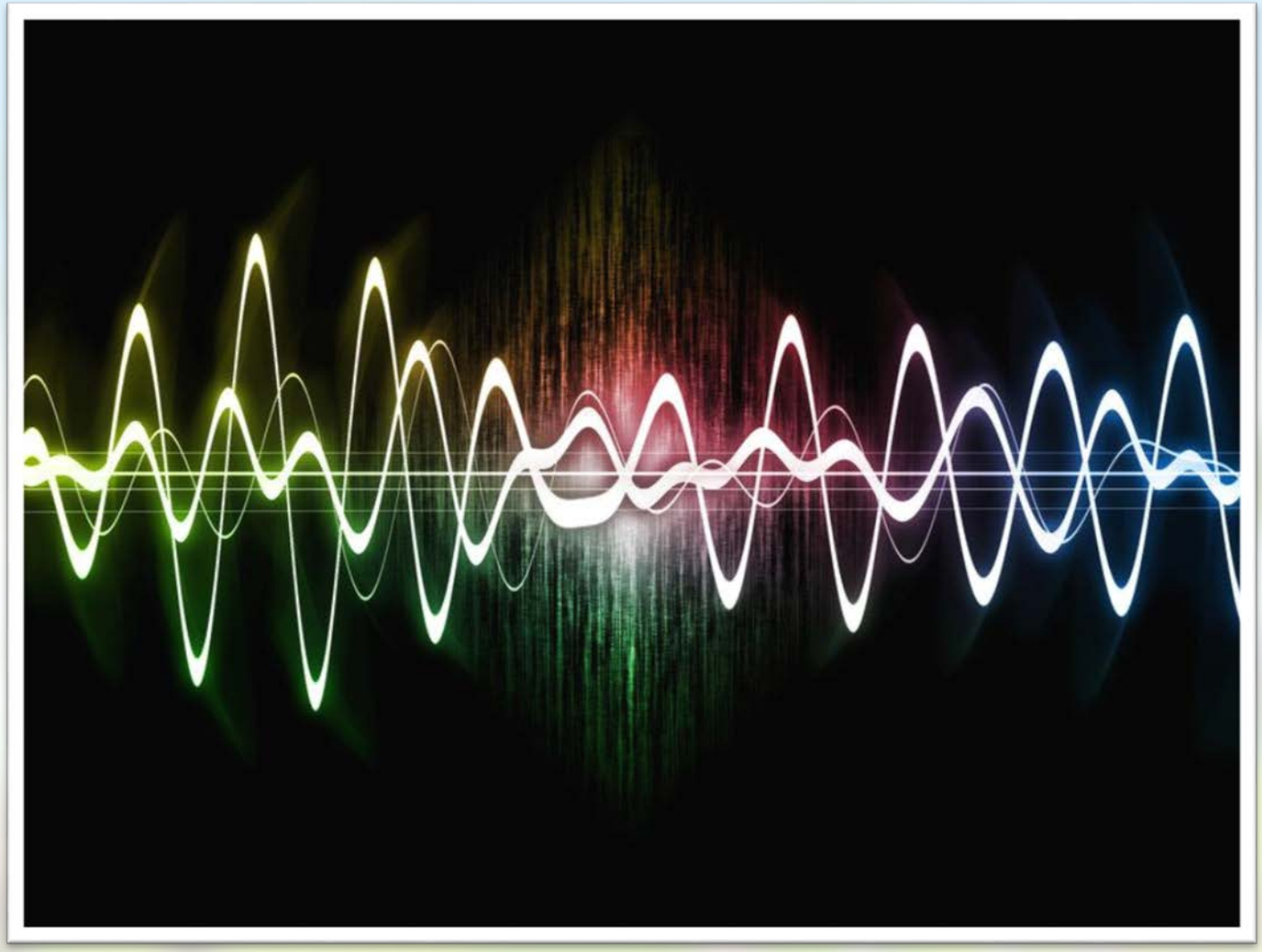
Orbit around a target



Landscape generation



Audio waves

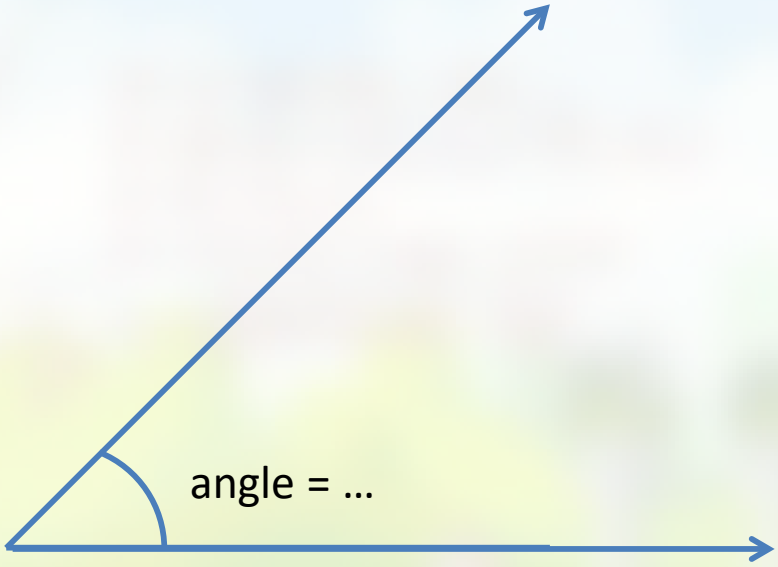


Angles

The corner stones of tri-angles

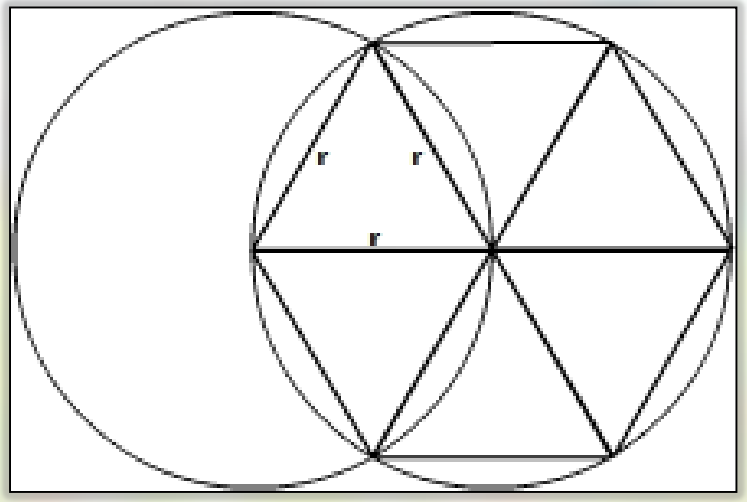
Trigonometry is about angles

- Angles are used to express directions
- Define amount of “turn” from **one** direction **to** another
- Two different measurements: degrees & radians



Angles in **degrees**

- Circles can be divided in **360 degrees**.
- 360 was “arbitrarily” chosen by man:
  - Babylonians counted to 60 on their fingers instead of 10
  - Circle divides nicely into 6 triangles with equal sides
  - 360 has a lot of factors (1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 18, 20, 24, 30, 36, 40, 45, 60, 72, 90, 120, 180, 360)



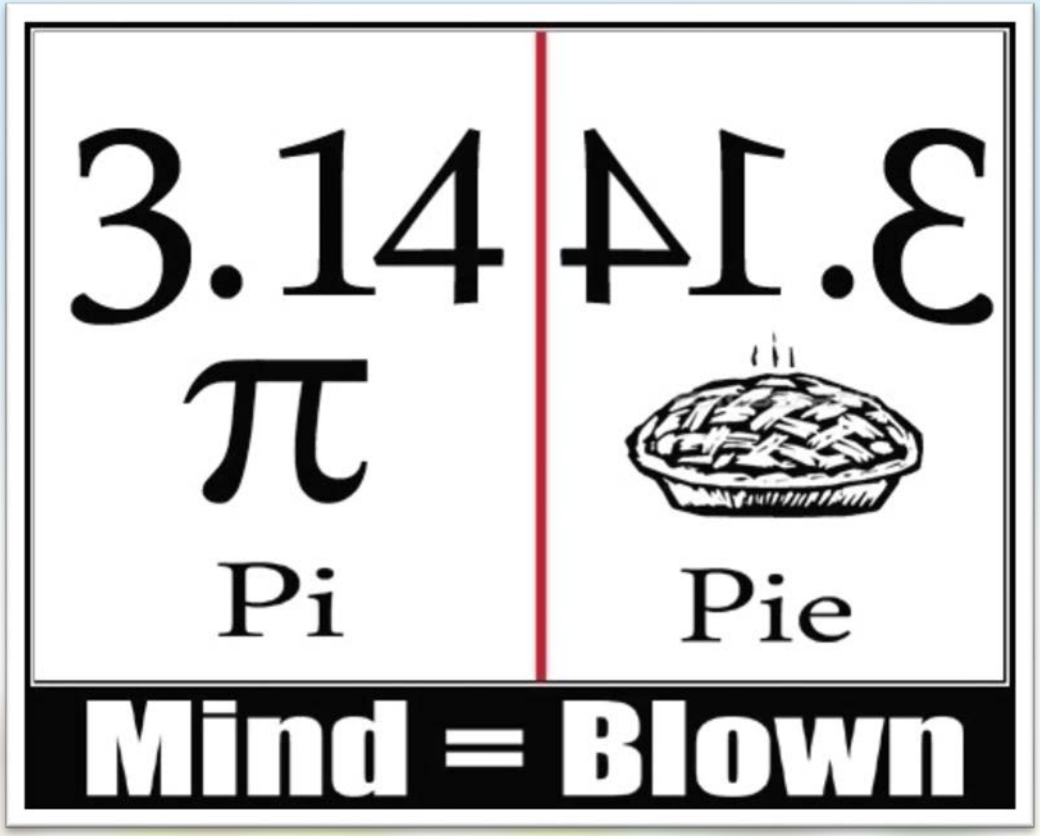
Angles in **radians**

- Circles can be divided in **2π radians**
- Radians are not arbitrarily chosen
- Based on ratio between diameter and circumference of a circle = C/d $\approx 3.14159265\ldots$ (unending)
- Cumbersome to pronounce so we use π, in C# Math.PI

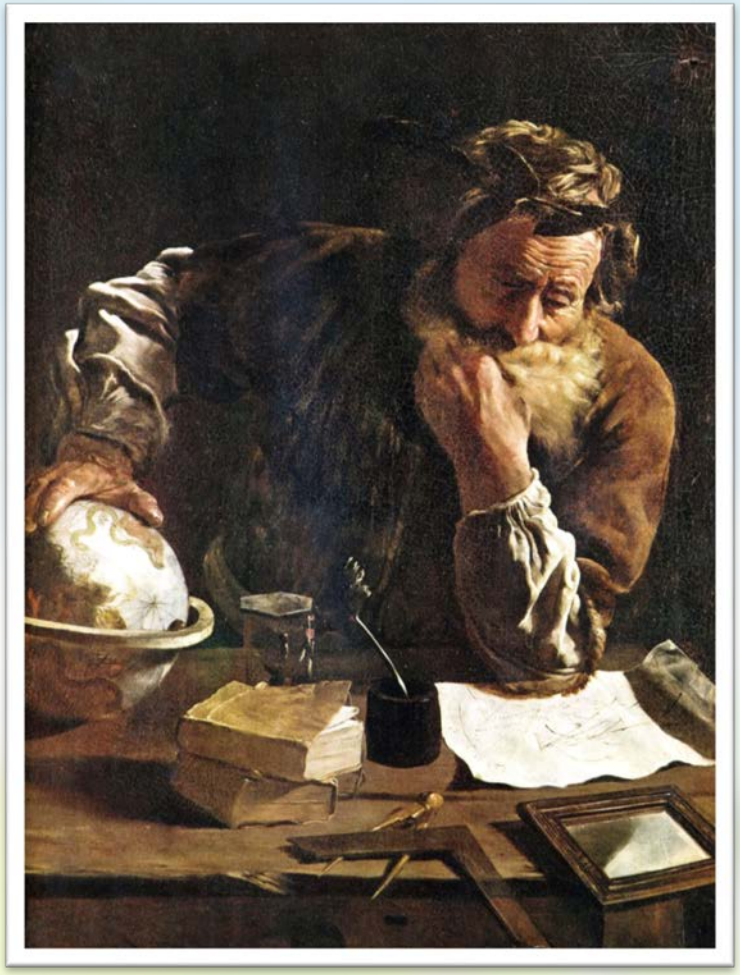
$$\frac{C}{d} = \pi$$

$$C = d\pi$$
$$C = 2\pi r$$

Radians

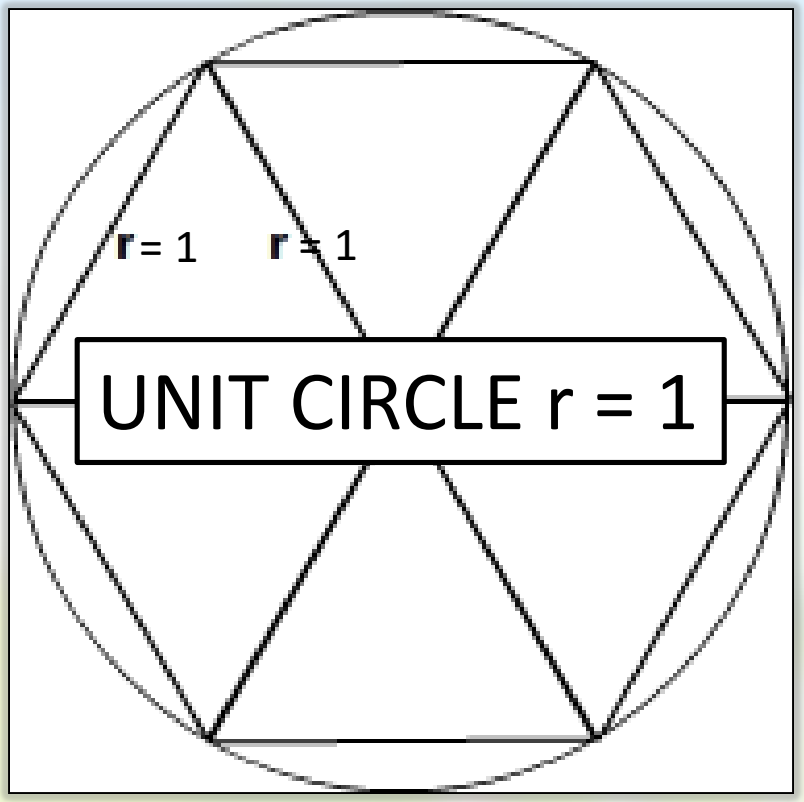


Archimedes (287 BC – 212 BC)



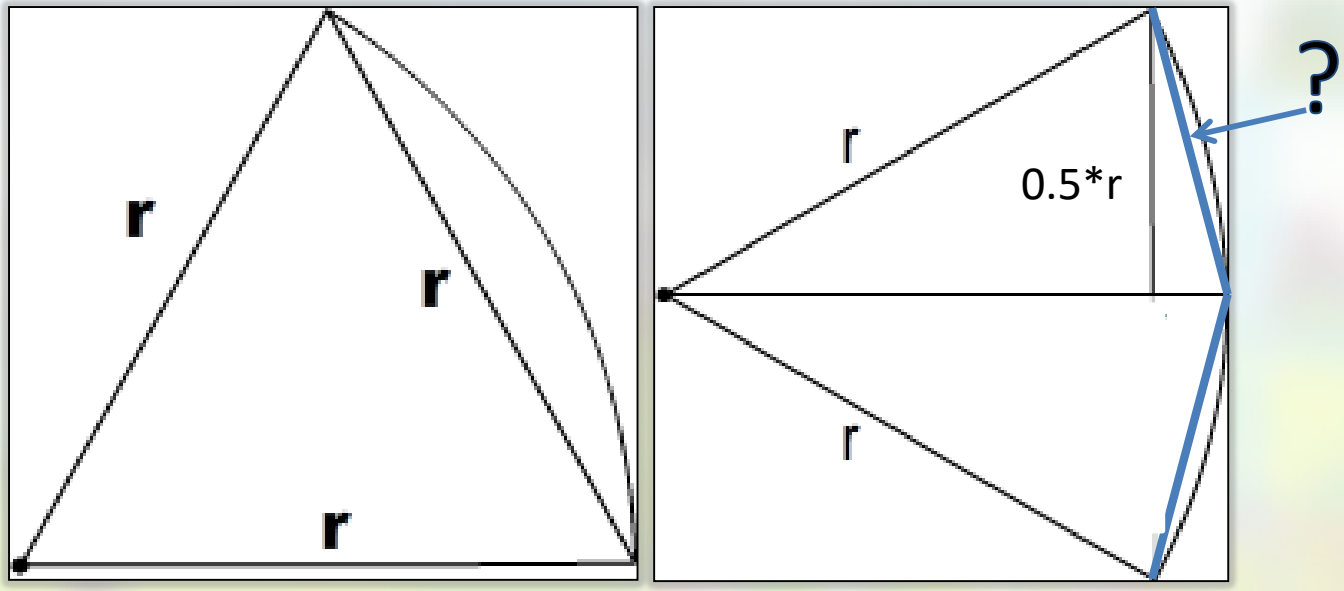
Archimedes' Method (287 BC)

- Observation:
  - **If** Circle radius 1 **then** circumference ≈ 6



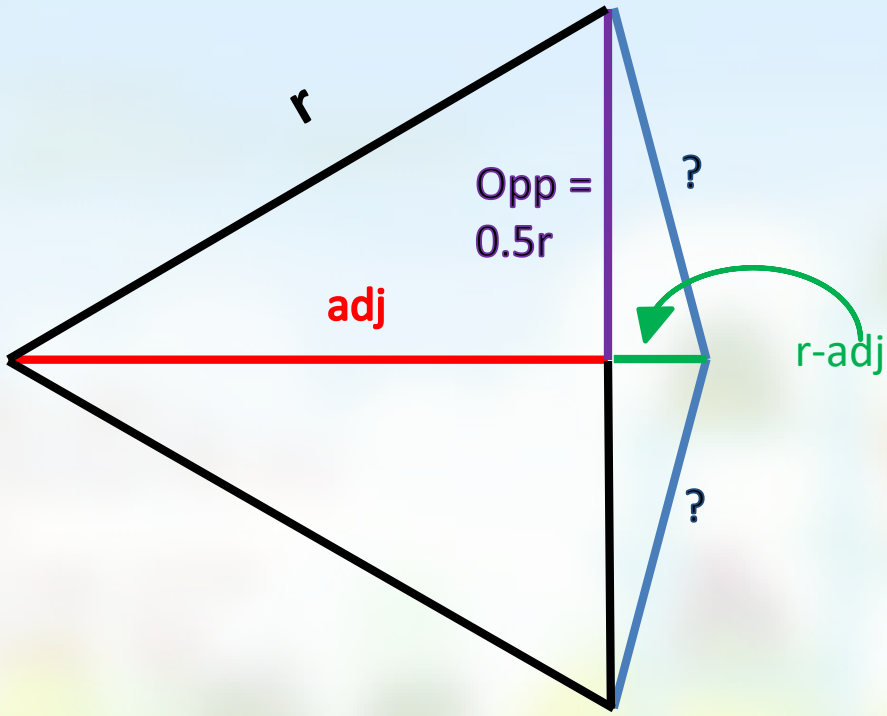
Archimedes' Method (287 BC)

- "If I take six sides of length 1, I get 6*1 = 6"
- "How can I improve this estimate?"
- (6 * r) vs (12 * ?) → better estimate?



Archimedes' Method (287 BC)

- "Look ma, I'm using Pythagoras!"
  - Calculate adj
  - Calculate r-adj
  - Calculate ?
- 6 * r ➔ 12 * ?



Archimedes' Method (287 BC)

- Doing this in excel shows us that the ratio of diameter vs circumference is 2 : 2π (2π ≈ 6.2831)
- In other words π ≈ 3.1415
- Try and repeat this process in excel if you are reaally bored...

#	r	opp	adj	leftover	new side length	2 * PI
12	1	0.5	0.866025404	0.133975	0.51763809	6.211657082
24	1	0.258819	0.965925826	0.034074	0.261052384	6.265257227
48	1	0.130526	0.991444861	0.008555	0.130806258	6.278700406
96	1	0.065403	0.997858923	0.002141	0.065438166	6.282063902
192	1	0.032719	0.999464587	0.000535	0.032723463	6.282904945
384	1	0.016362	0.999866138	0.000134	0.016362279	6.283115216
768	1	0.008181	0.999966534	3.35E-05	0.008181208	6.283167784
1536	1	0.004091	0.999991633	8.37E-06	0.004090613	6.283180926
3072	1	0.002045	0.999997908	2.09E-06	0.002045307	6.283184212

Angles

- In conclusion: 2 units of measurement for angles, based on:
  - 360 slices: degrees
  - the 2π circumference of a unit circle: radians
- And of course these are related: two sides of same coin
- Game engines often use both types: conversion required!

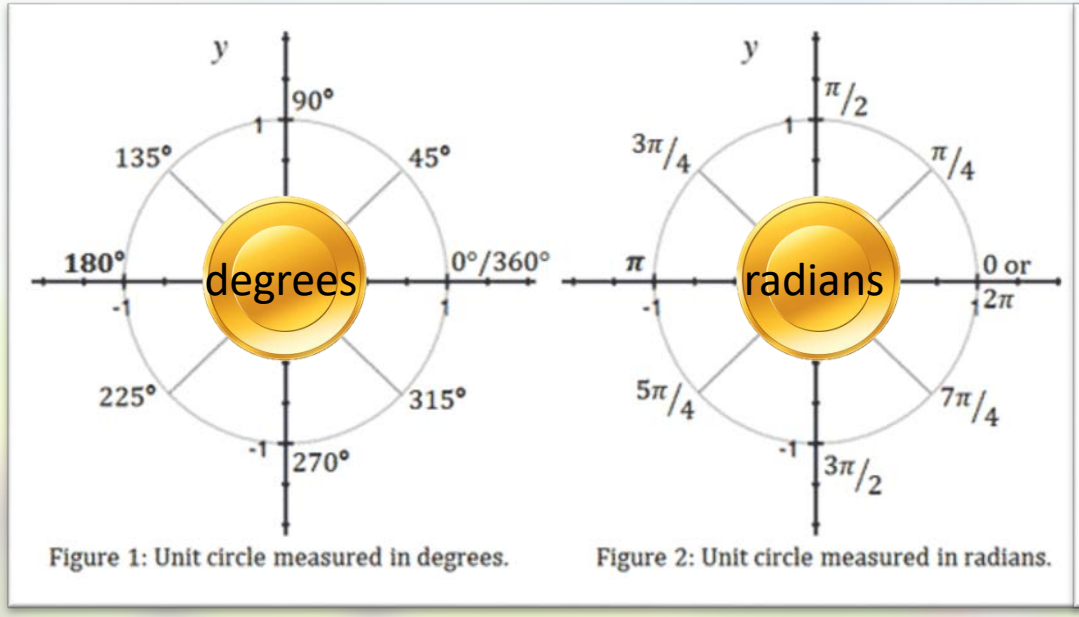
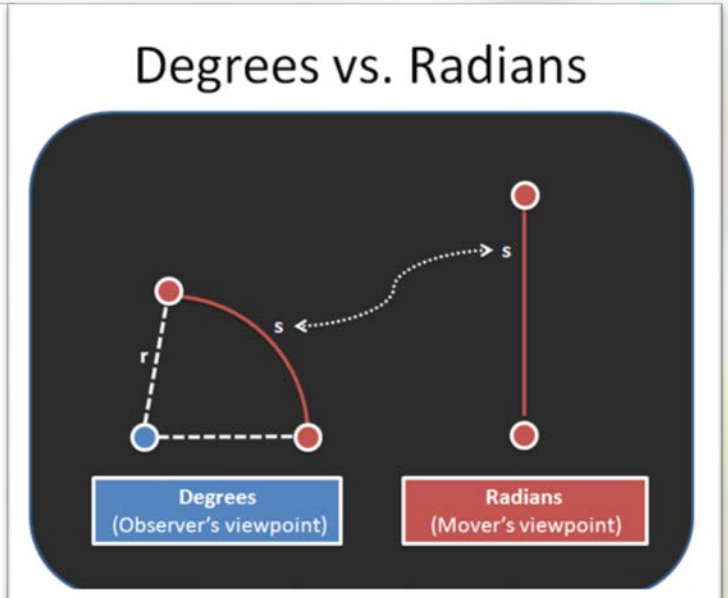


Figure 1: Unit circle measured in degrees.

Figure 2: Unit circle measured in radians.



Angle conversion

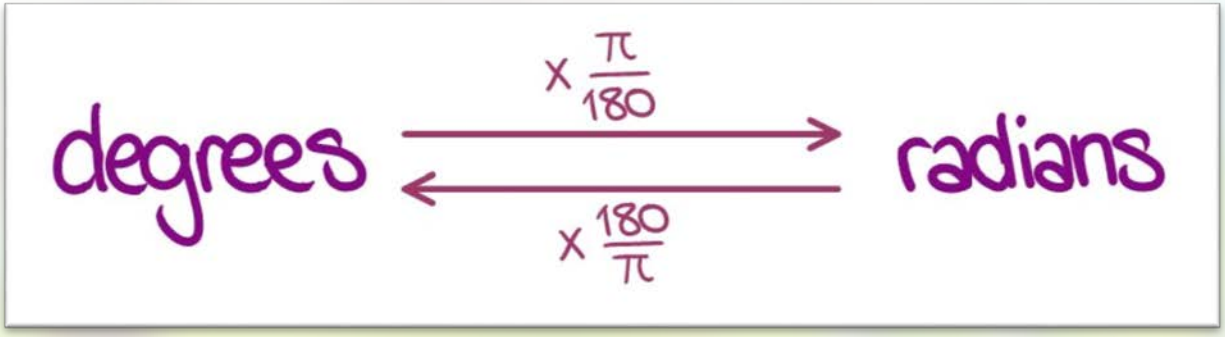
Angle Conversion ➔ $360° = 2\pi^{rad}$

- From degrees to radians:

$$1° \equiv \frac{\pi}{180°}$$

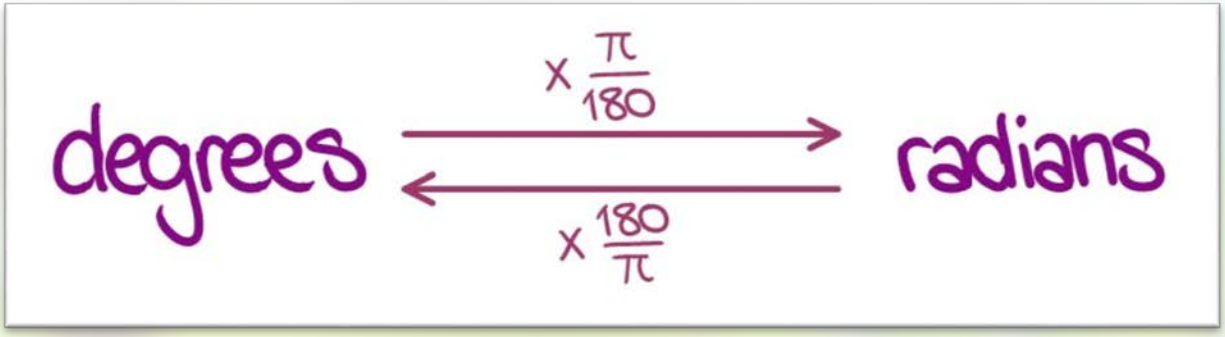
- From radians to degrees:

$$1^{rad} = \frac{180°}{\pi}$$



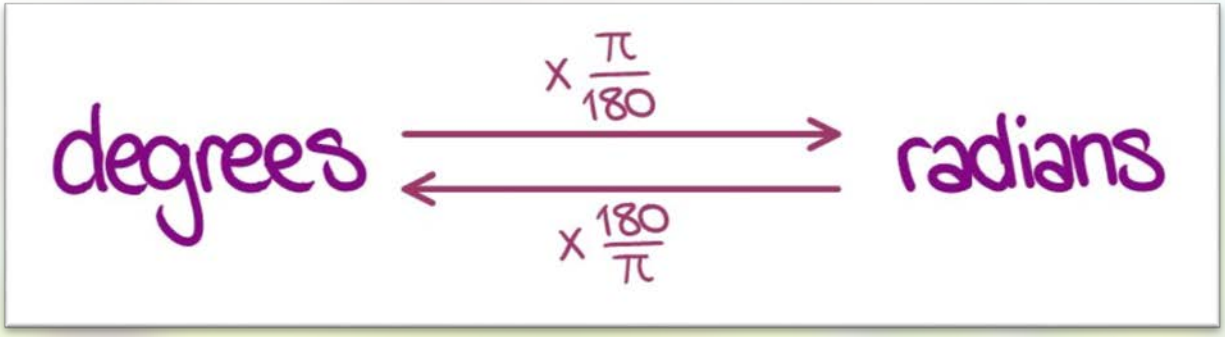
Angle Conversion Reminder Tip

- It's always either $*\pi/180$ or $*180/\pi$
- Degrees to radians:
  - Removing degrees, add radians → /degrees → * radians
- Radians to degrees:
  - Removing radians, add degrees → /radians → * degrees



Exercises

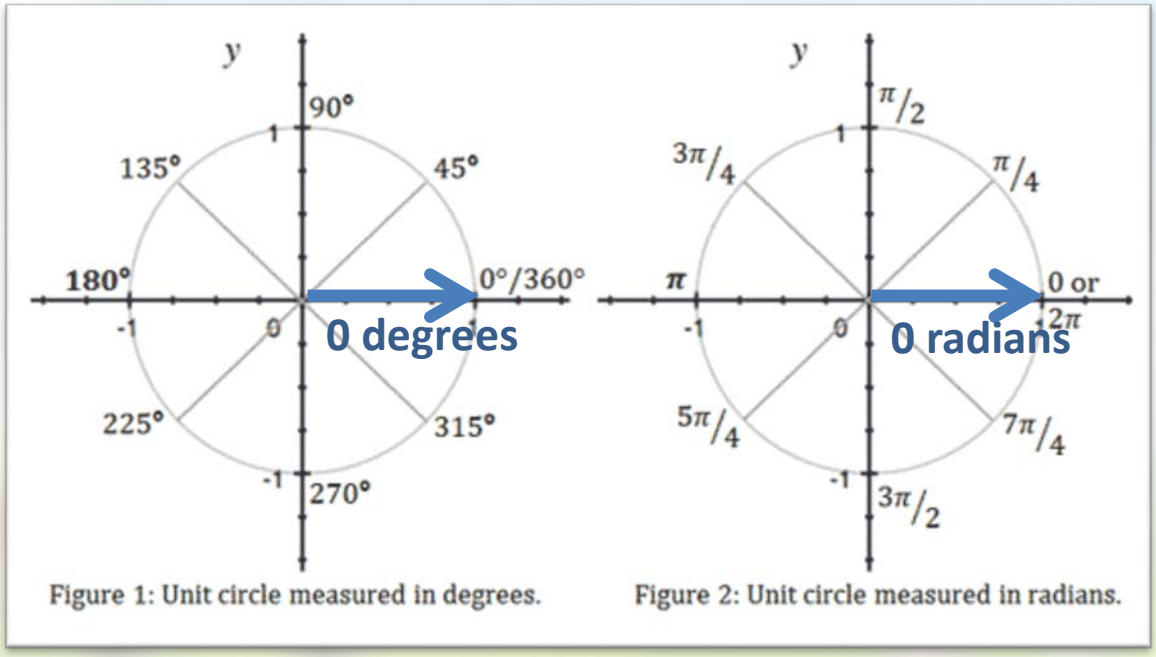
- Calculate circumference of a circle with d = 4
- Calculate circumference of a circle with r = 4
- Convert 45 degrees to radians
- Convert 135 degrees to radians
- Convert 1.75 Pi to degrees



Angle conventions

Angle conventions: angle 0

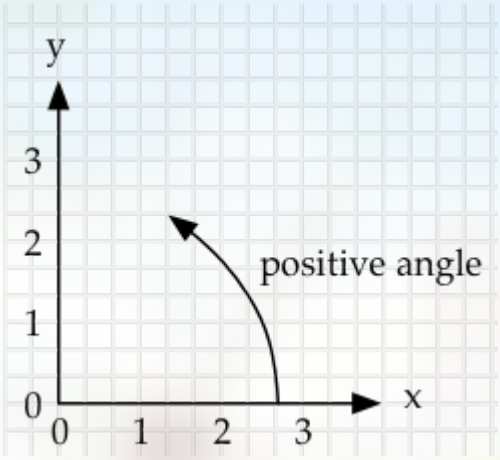
- For example: mySprite.rotation = 0; (degrees)
- Example: +001_samples\MovingSpaceship
- Only use/accept **correctly** aligned GFX!



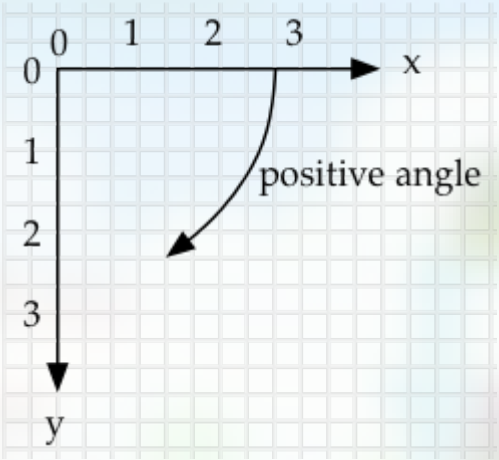
Angle conventions: positive angle

- A + rotation angle, rotates from +x to +y

+ angle when +y is up = CCW



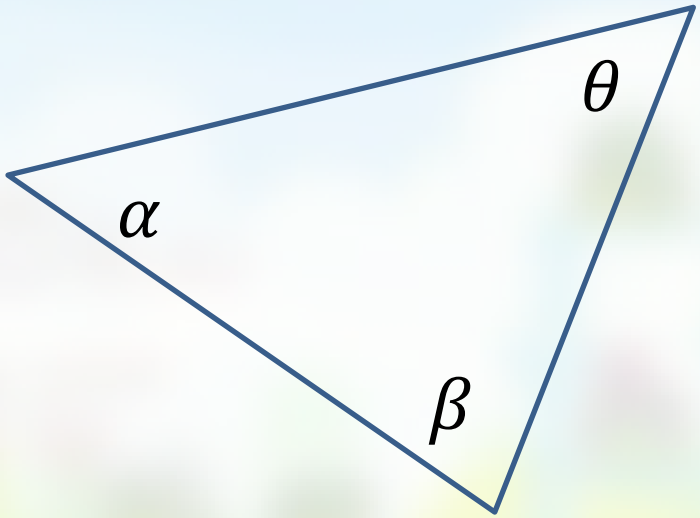
+ angle when +y is down = CW



- GXPEngine? +001_samples\RotatingSpaceship

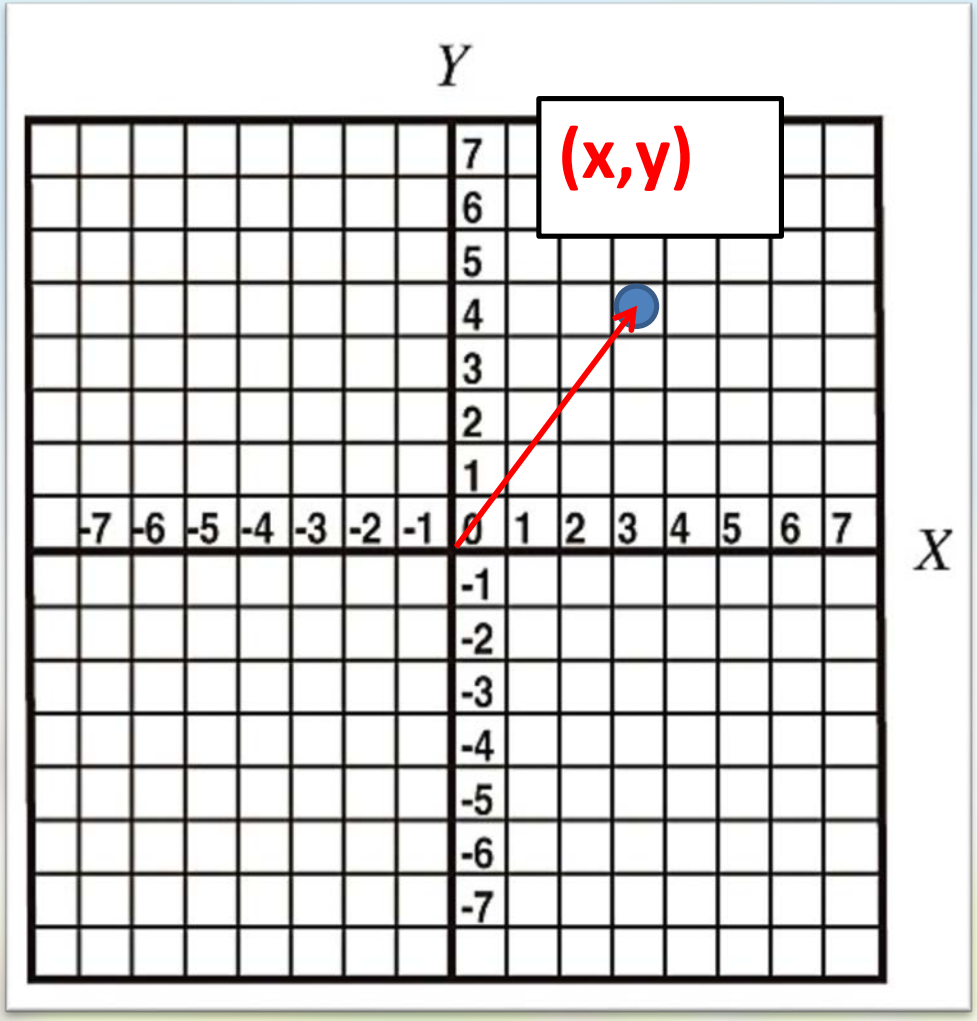
Angle conventions: angle names

- Angles are denoted by using Greek letters:
  - α alpha
  - β beta
  - θ theta
  - … etcetera

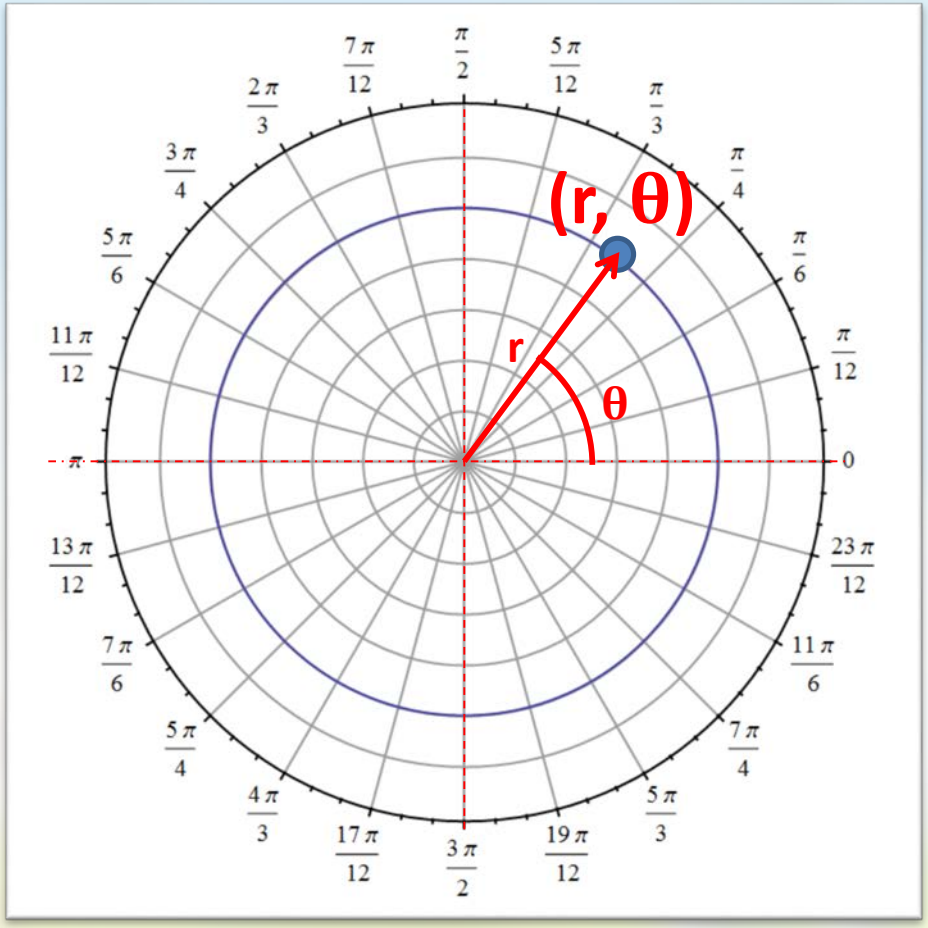


Polar Coordinate System

Cartesian Coordinates

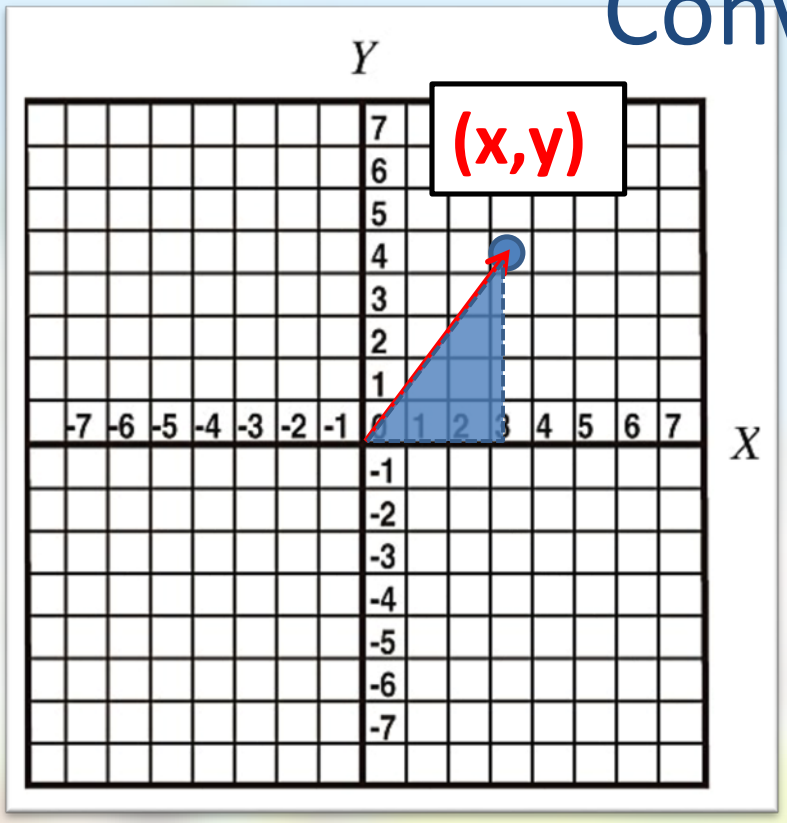


Polar Coordinate System

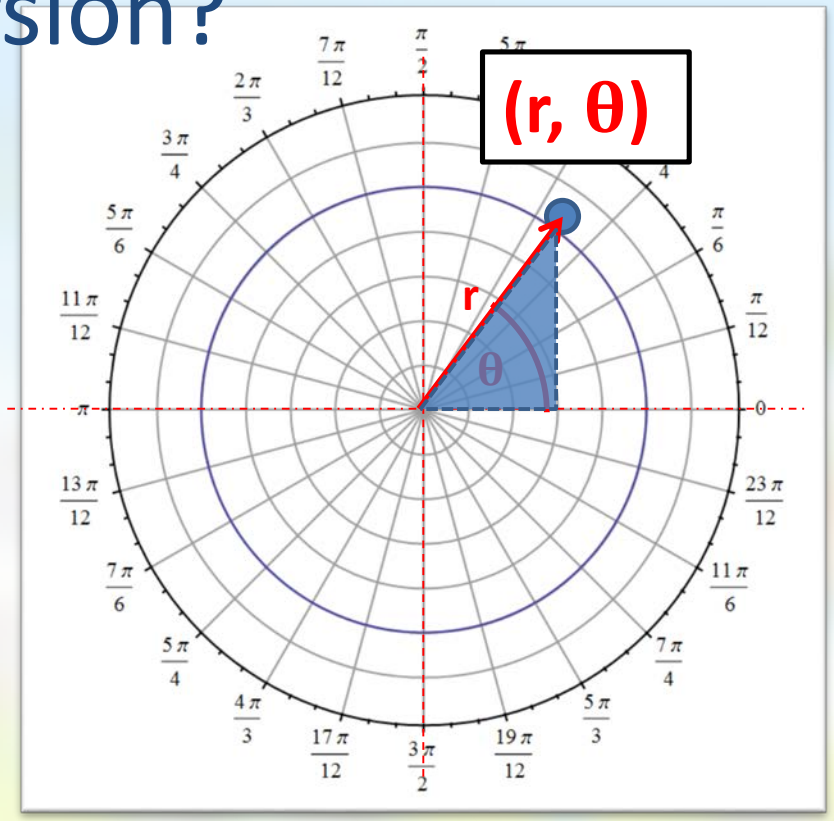


Coordinate System Conversion

Conversion?



Cartesian Coordinate System



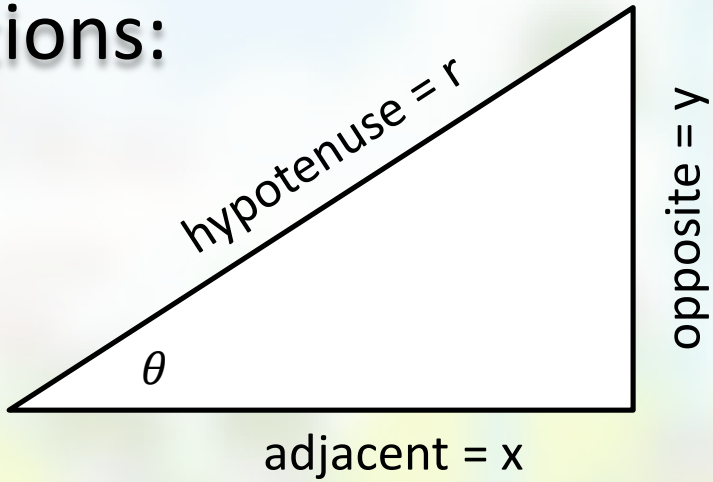
Polar Coordinate System

Trigonometric functions

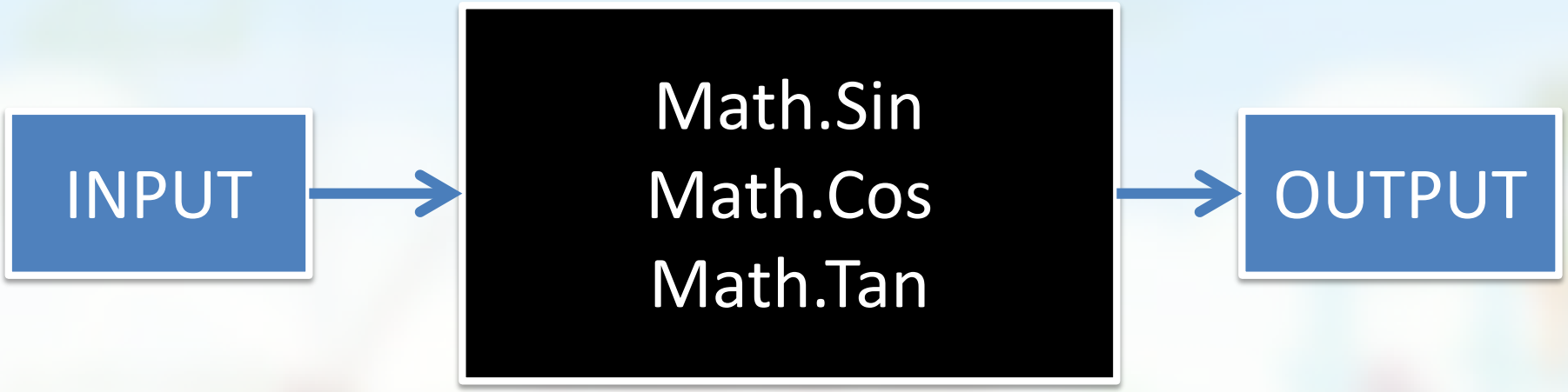
Trigonometric functions

- Describe relations between **the angles** in a right sided triangle and **the length** of that triangle's sides

- Most well known trig functions:
  - $\boldsymbol{sin\,\theta}$ **= opp/hyp =** $\boldsymbol{y/r}$
  - $\boldsymbol{cos\,\theta}$ **= adj /hyp =** $\boldsymbol{x/r}$
  - $tan\,\theta$ = opp/adj $= y/x$
  - θ is always in **radians !**

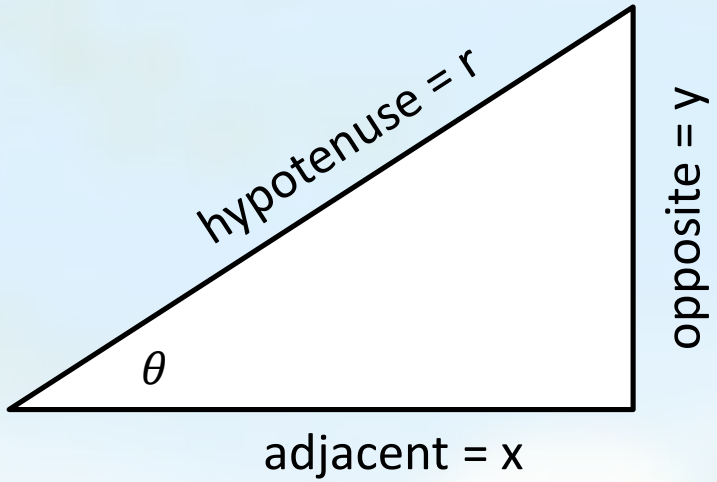


Trig functions are black box functions



Memorizing trig functions

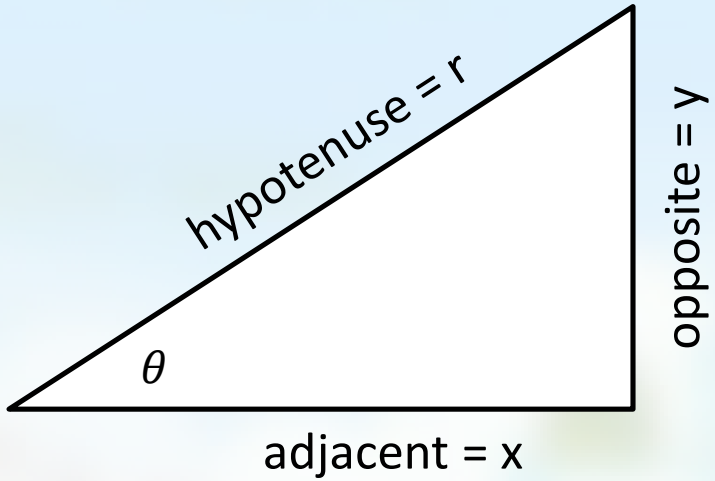
- $\boldsymbol{sin}\,\theta = \textbf{o}\text{pp} / \textbf{h}\text{yp} = y/r$
- $\boldsymbol{cos}\,\theta = \textbf{a}\text{dj} / \textbf{h}\text{yp} = x/r$
- $\boldsymbol{tan}\,\theta = \textbf{o}\text{pp}/ \textbf{a}\text{dj} = y/x$



From Polar to Cartesian

If we know that:

$sin\,\theta =$ opp/hyp = y/r
$cos\,\theta =$ adj/hyp = x/r
$tan\,\theta =$ opp/adj = y/x

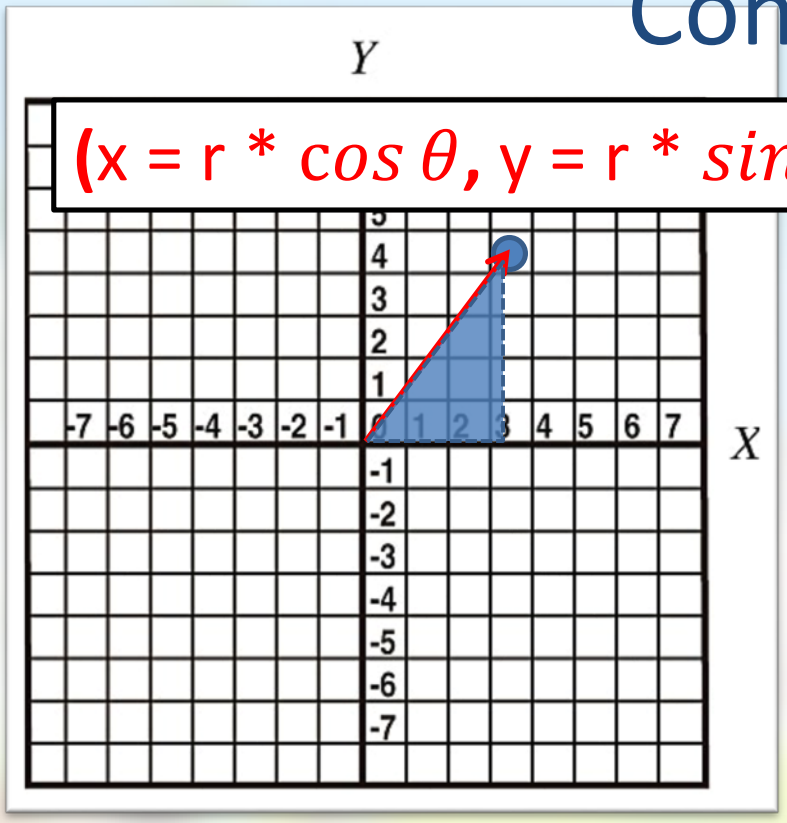


then, given **(r, θ)** what is **(x, y)** ?

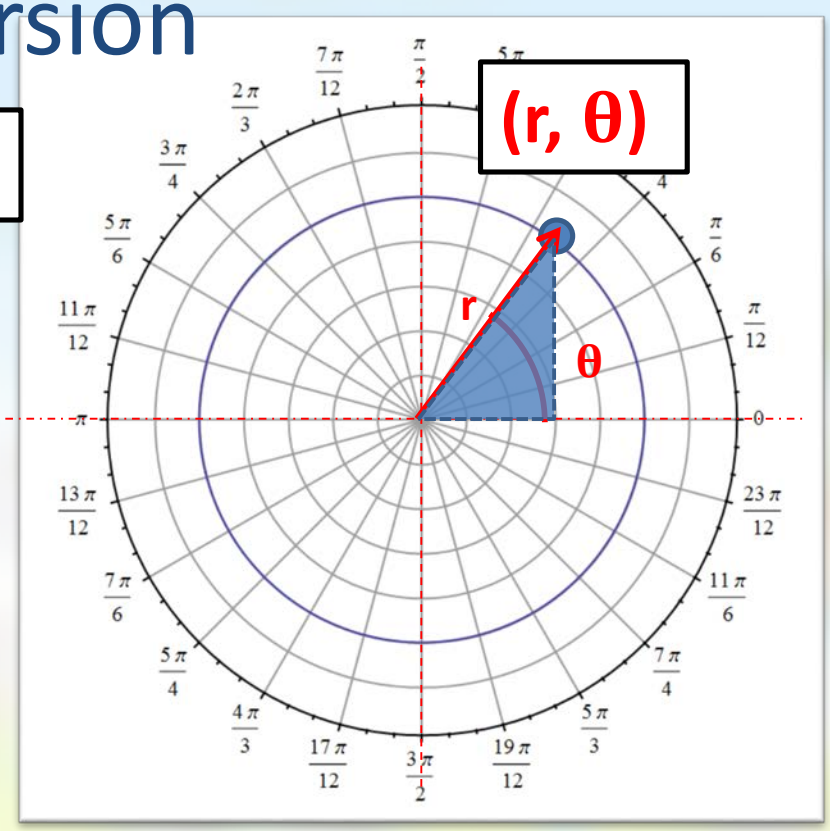
(r, θ) → x = r * $cos\,\theta$
(r, θ) → y = r * $sin\,\theta$

Coordinate System Conversion

Conversion

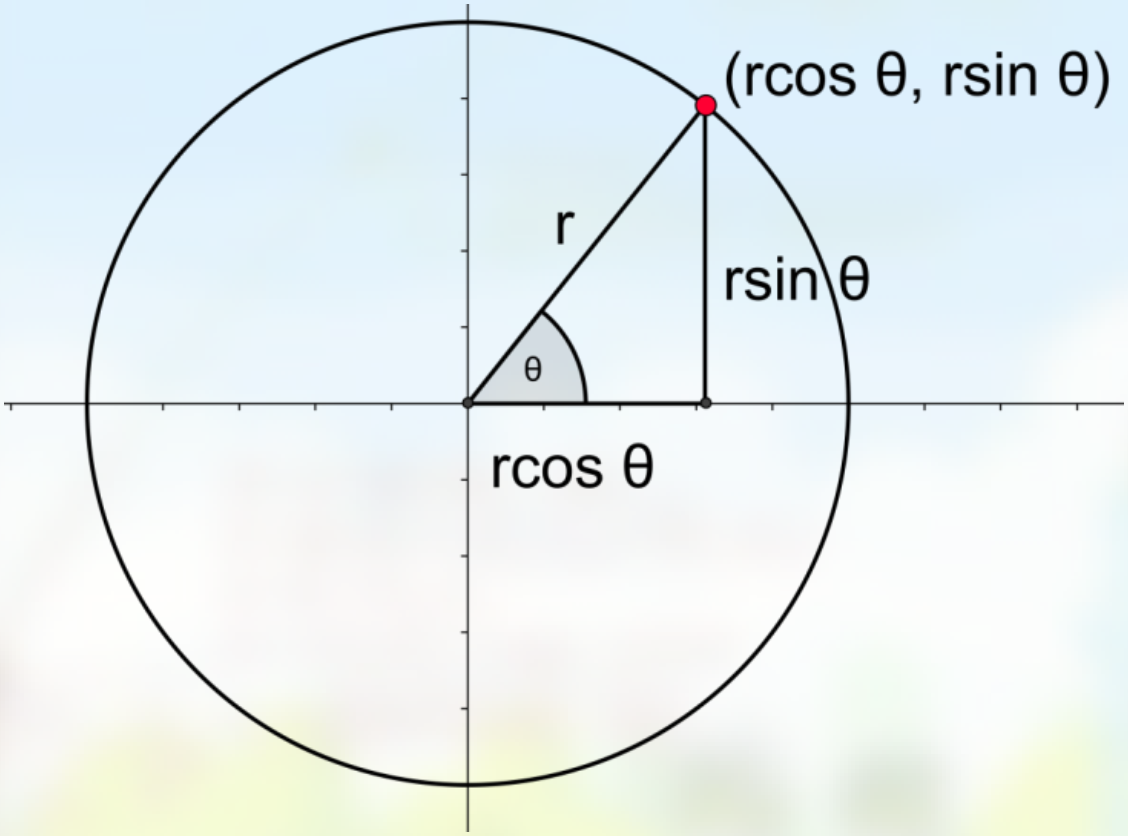


Cartesian Coordinate System

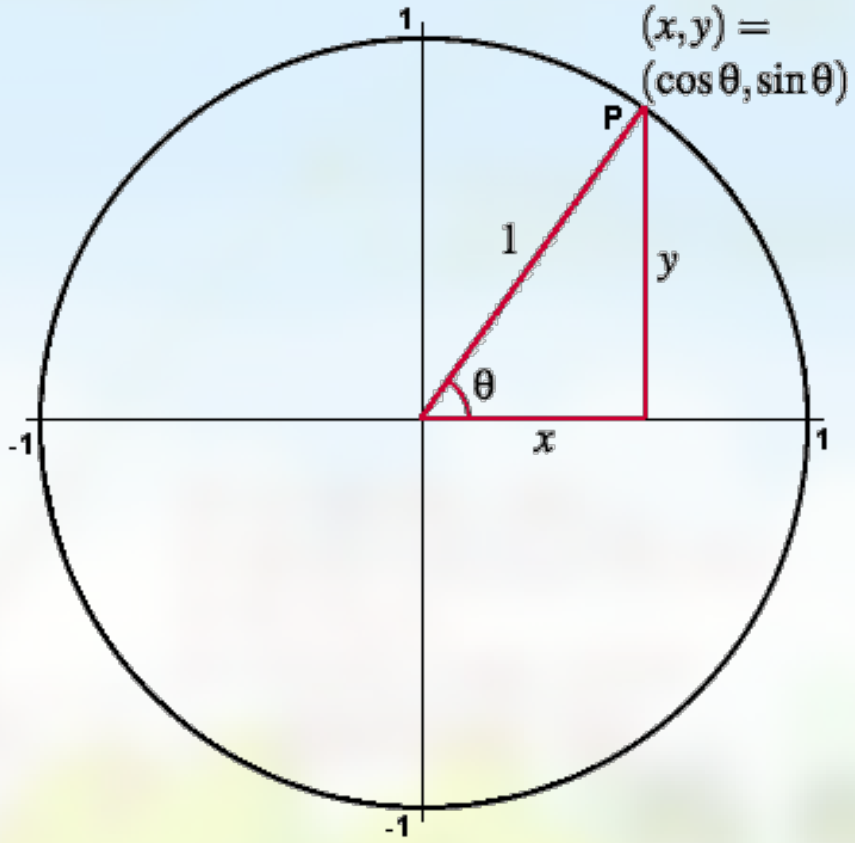


Polar Coordinate System

Cos/Sin describe circular motion



Unit circle, r = 1

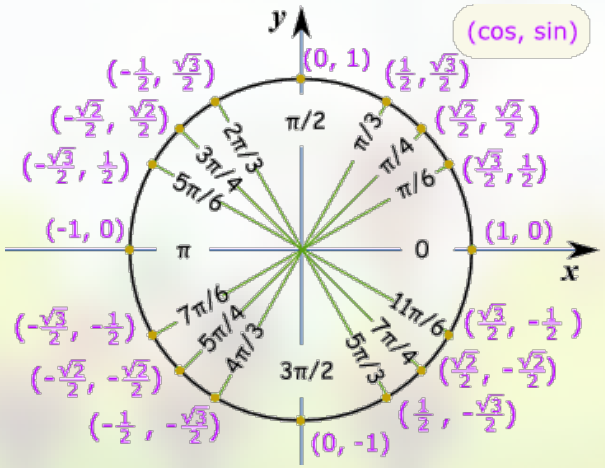


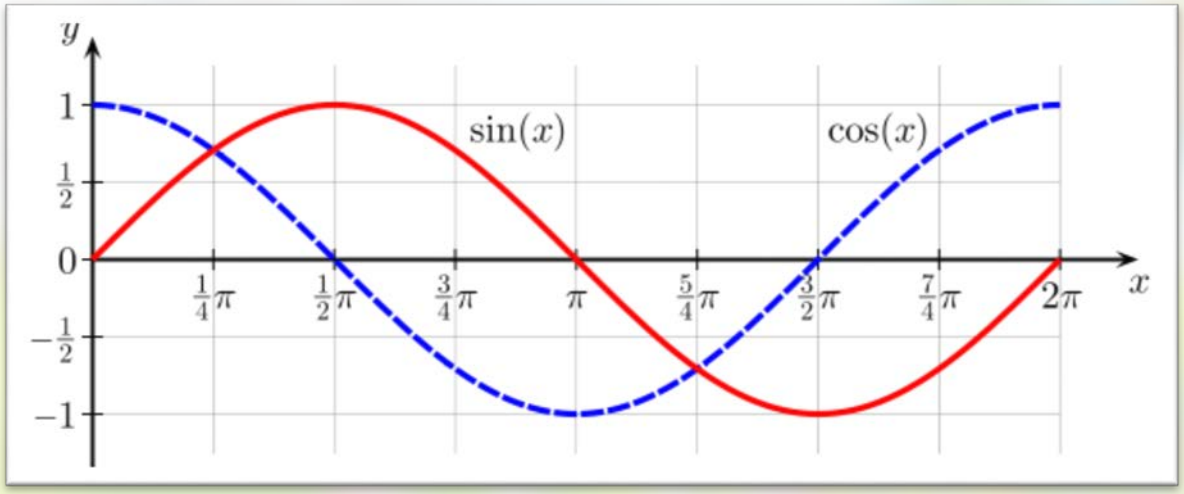
What is the length of each (x,y) vector on this unit circle?

$$1 = \sqrt{x^2 + y^2} = \sqrt{\cos(\theta)^2 + \sin(\theta)^2}$$

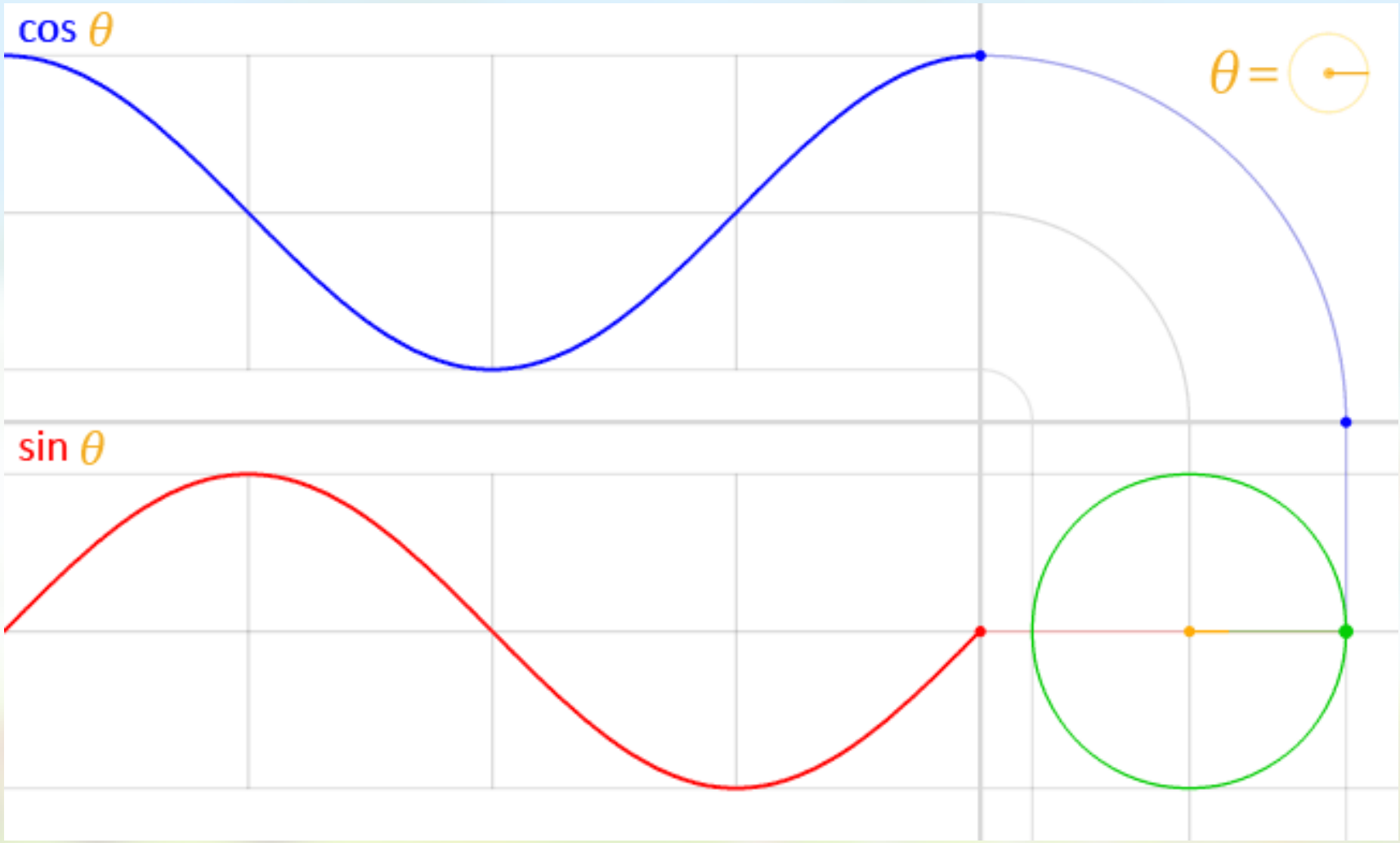
(Co)sine in/output

- Input in radians % 2π, also called phase
- Output between -1 and 1 which repeats every 2π, called period
- Cos/sin → same movement, with a phase shift of $\frac{1}{2}\pi$:
  - $\cos(\alpha) = \sin(\alpha + \frac{1}{2}\pi)$ & $\sin(\alpha) = \cos(\alpha - \frac{1}{2}\pi)$
- Converts input in radians to "wobbling" between -1 & 1



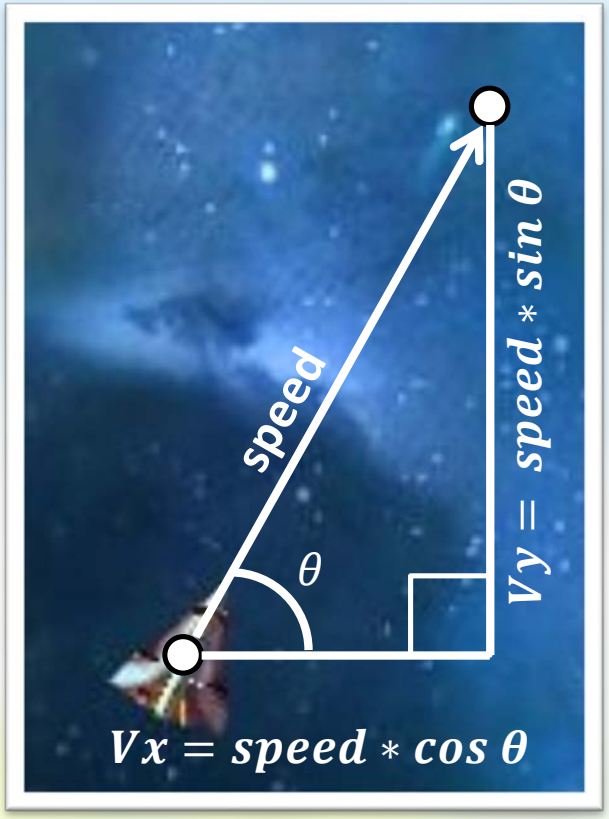


“Realtime” demo of cos/sin



Applications of Co(sine)

Move in a direction



Move in a direction

- Different ways (direct conversion):
  - velocity.x = speed * cos (angle)
  - velocity.y = speed * sin (angle)
- Or (generate unit vector and scale):
  - velocity.SetXY (cos(angle), sin(angle))
  - velocity.Scale (speed);
- Or (implement helper method to encapsulate cos/sin):
  - velocity.OnUnitCircle(angle).Scale (speed);

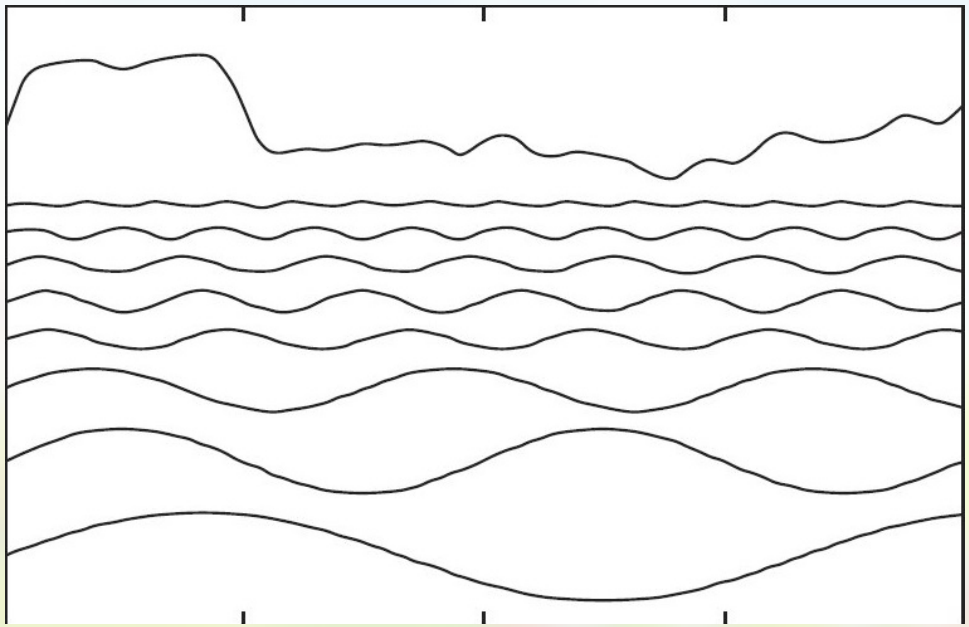
Can't we just use the Sprite Move(x,y) method for this ??

- There is a catch:
  - angle is stored in sprite.rotation, which is in degrees
  - all trig functions only take radians => conversion required

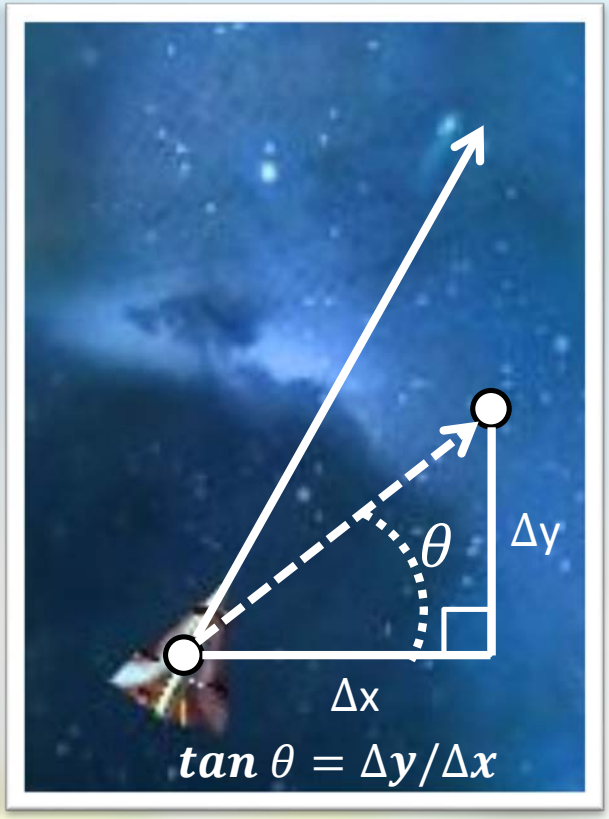
- Example +002_moving_in_a_direction

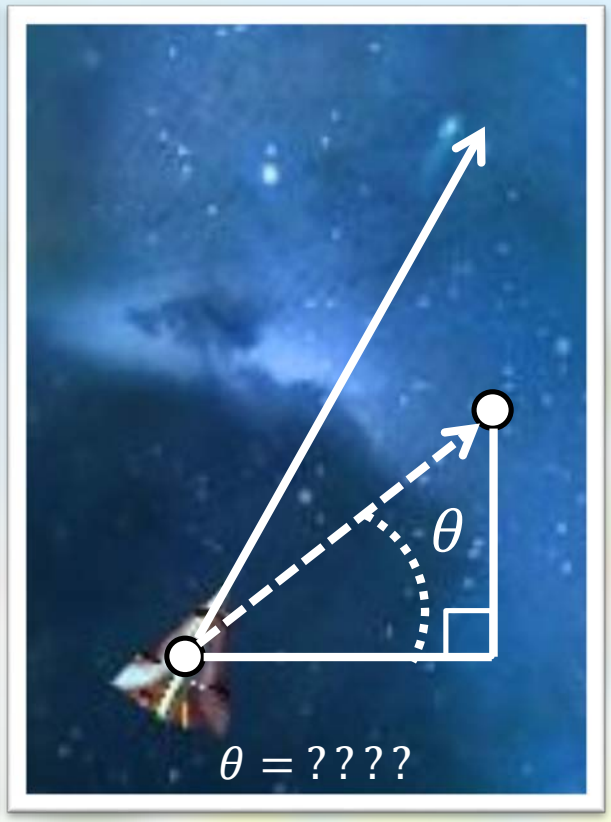
Other using of trig methods

- Cos/sin are just functions, mapping values
- Check out 003_samples, demonstrates:
  - Positions, orbiting, scaling, colors, slashing blades, wave generation, etc



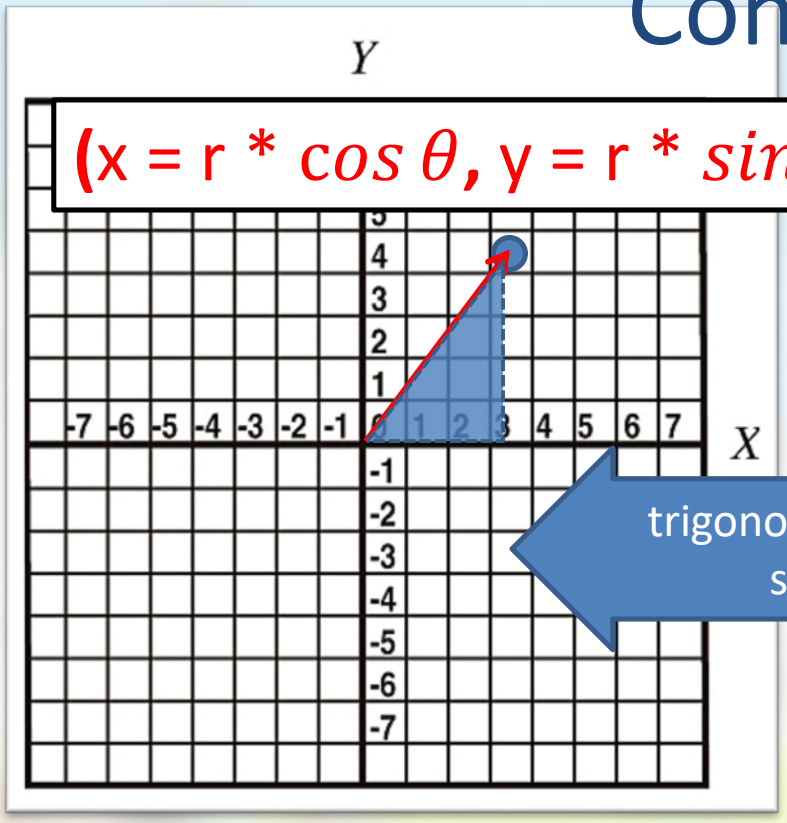
Rotate to a target: Δx Δy to angle?

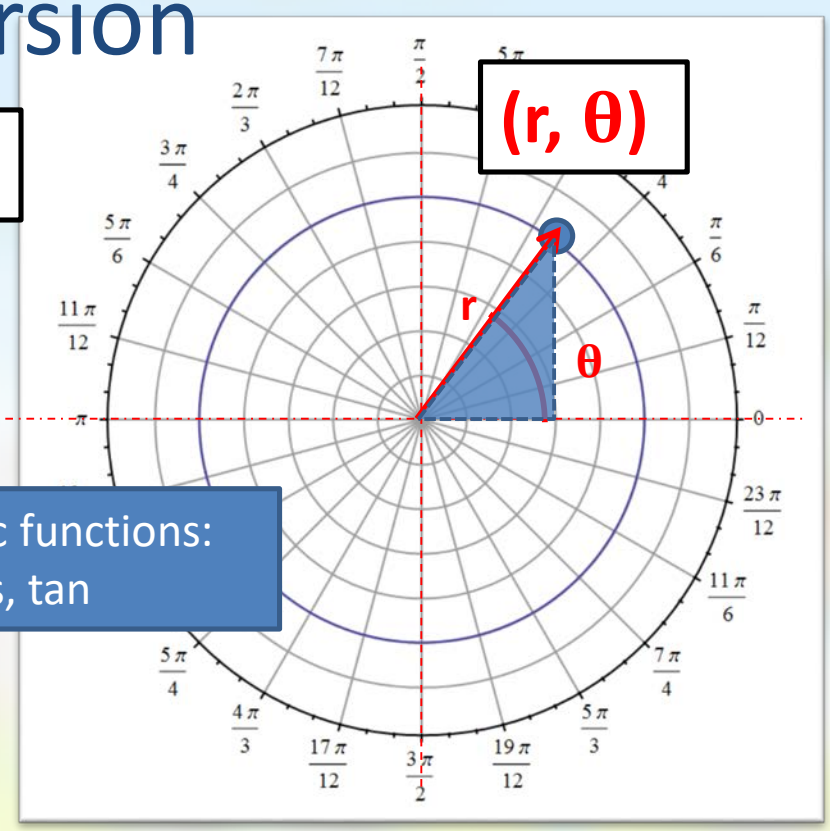




Coordinate System Conversion

Conversion





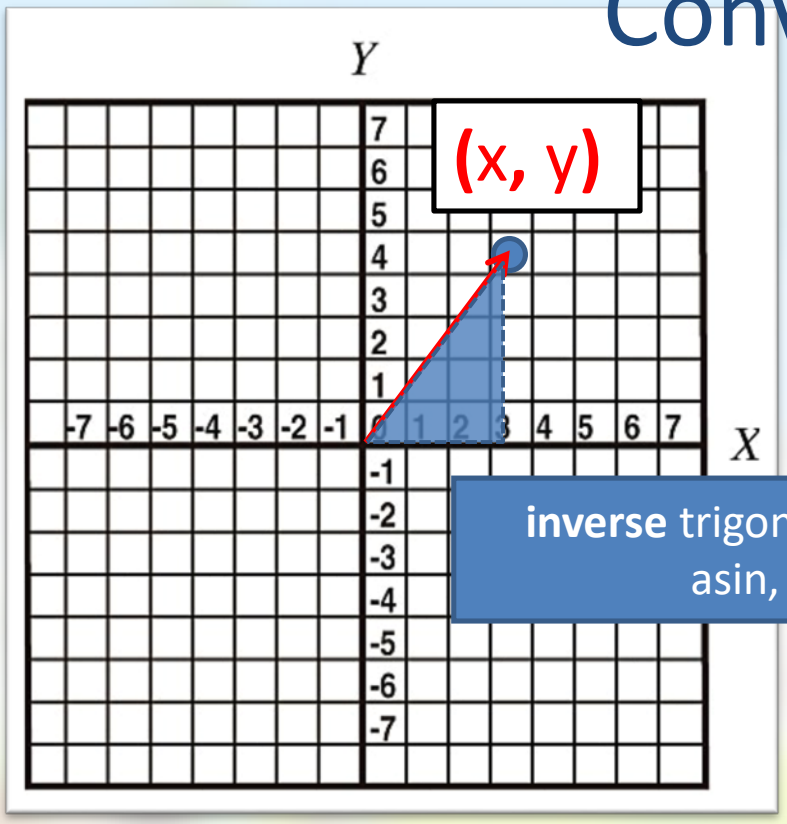
trigonometric functions:
sin, cos, tan

Cartesian Coordinate System

Polar Coordinate System

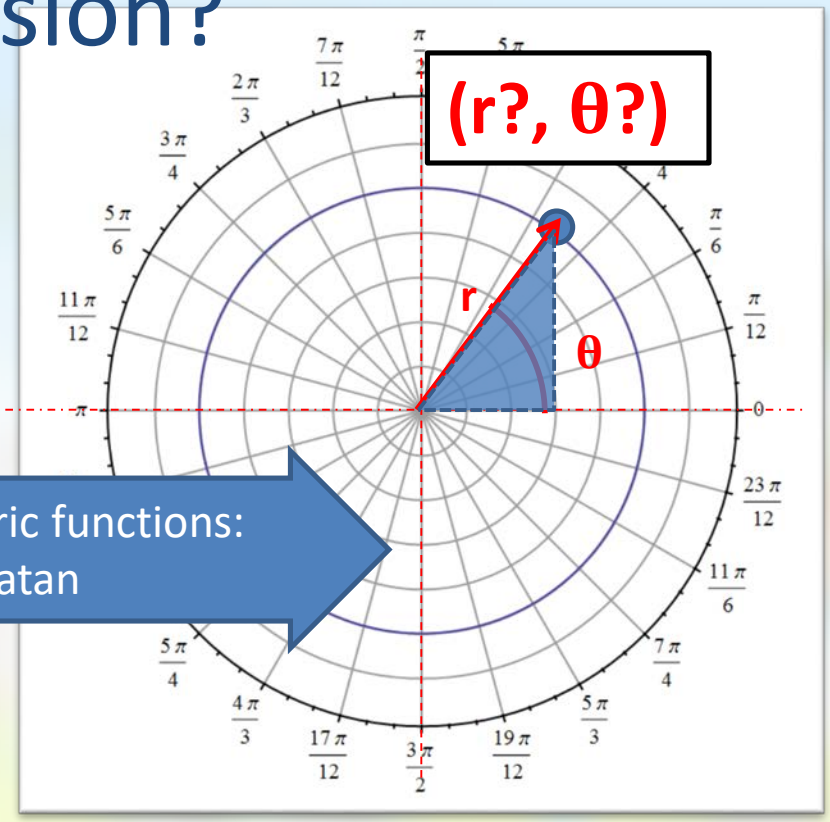
Coordinate System Conversion

Conversion?



Cartesian Coordinate System

inverse trigonometric functions:
asin, acos, atan



Polar Coordinate System

Inverse Trigonometric functions

Inverse trigonometric functions

- Inverse "undoes" the result of the non-inverse:
  - asin (sin θ) = θ, acos (cos θ) = θ, atan (tan θ) = θ

$$\sin\theta = y/r \quad \cos\theta = x/r \quad \boldsymbol{\tan\theta = y/x}$$

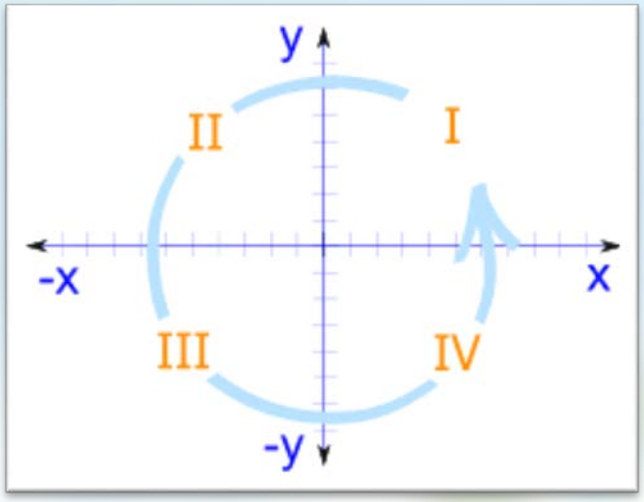
$$\arcsin(\sin\theta) = \arcsin(y/r) \quad \arccos(\cos\theta) = \arccos(x/r) \quad \boldsymbol{\arctan(\tan\theta) = \arctan(y/x)}$$

$$\theta = \arcsin(y/r) \quad \theta = \arccos(x/r) \quad \boldsymbol{\theta = \arctan(y/x)}$$

  - Another way to write $asin$ is: $arcsin$, sin^{-1}
  - It's called arc because it results in an arc (angle)
  - Math.Asin, Math.Acos, Math.Atan, Math.Atan2 (**?!**)

Atan (y/x) to calculate the angle

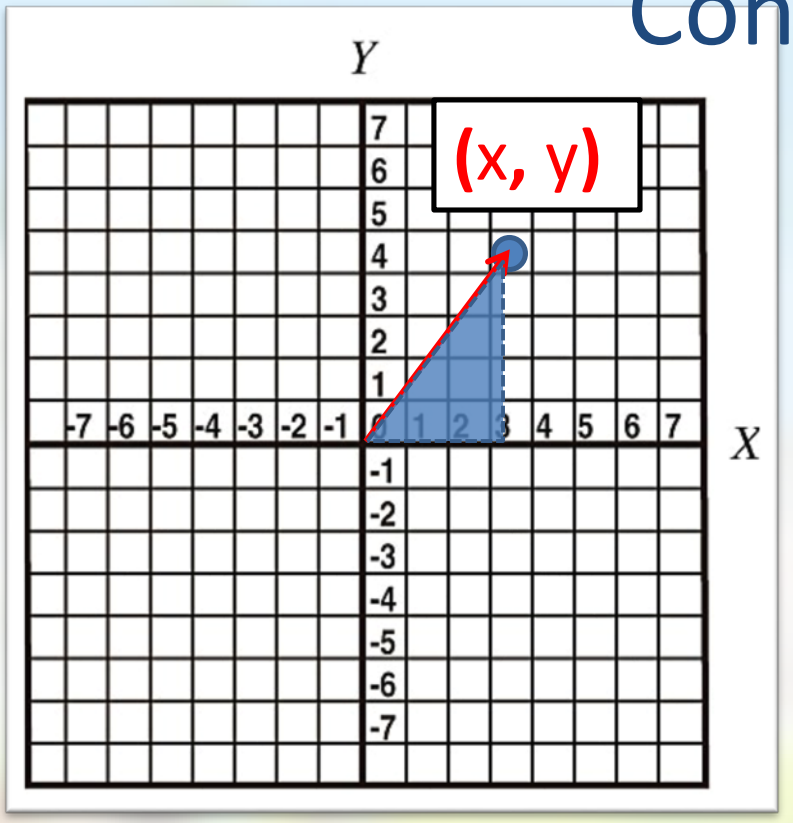
- y/x means:
  - hole in the universe when x ➔ 0
  - y/x == -y/-x and -y/x == y/-x ➔ which quadrant??



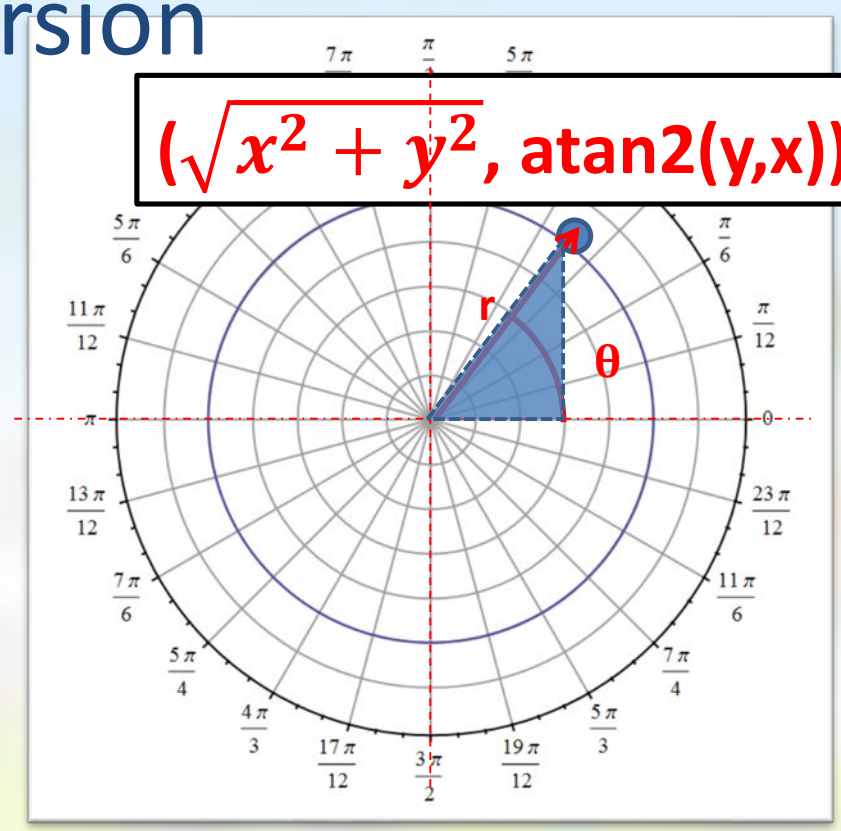
- Conclusion: converting y/x back to an angle is doable but cumbersome with Atan
- **Math.Atan2 (y,x)** takes care of these details
- **Warning**: result is in **radians**

Coordinate System Conversion

Conversion

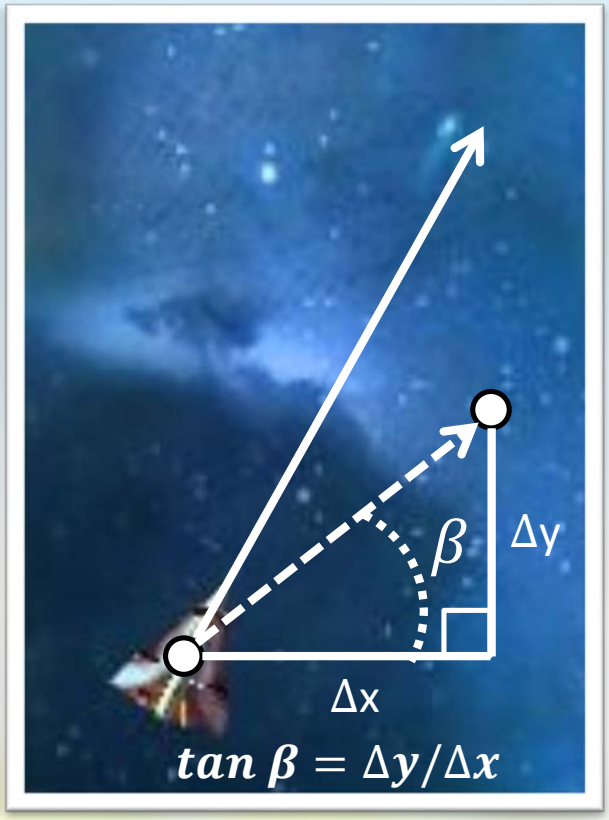


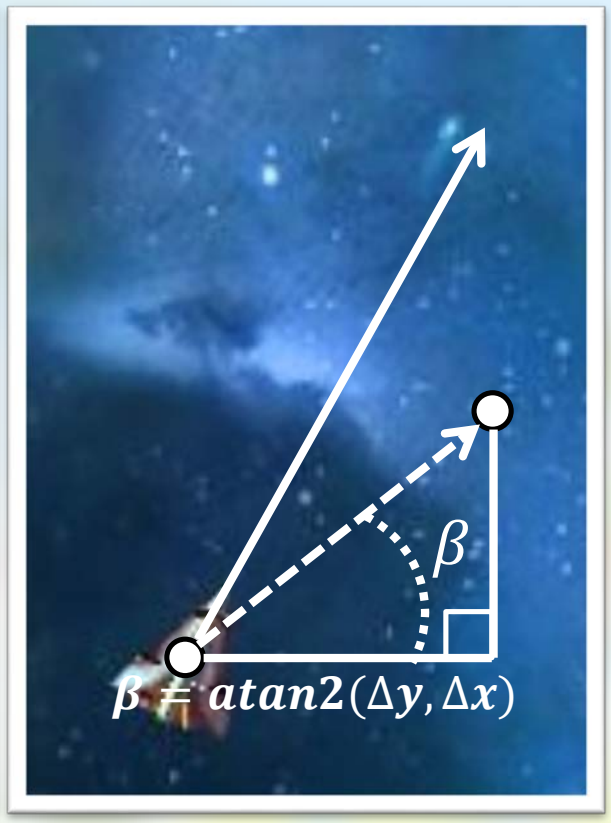
Cartesian Coordinate System



Polar Coordinate System

Rotate to a target: Δx Δy to angle



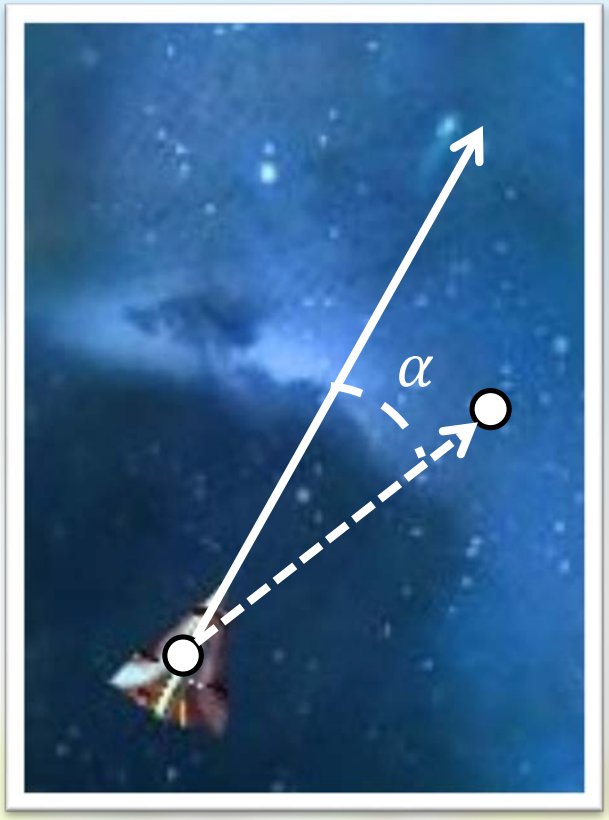


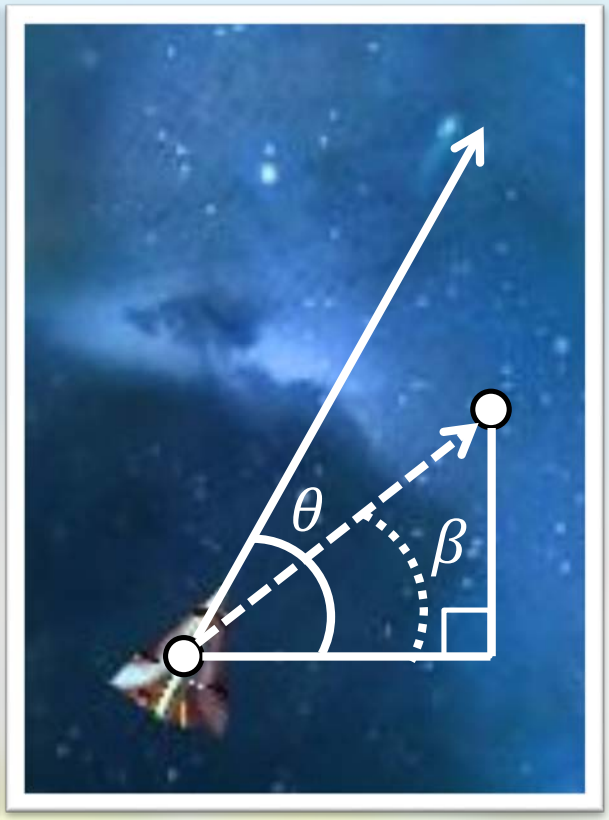
+004_rotating_in_a_direction

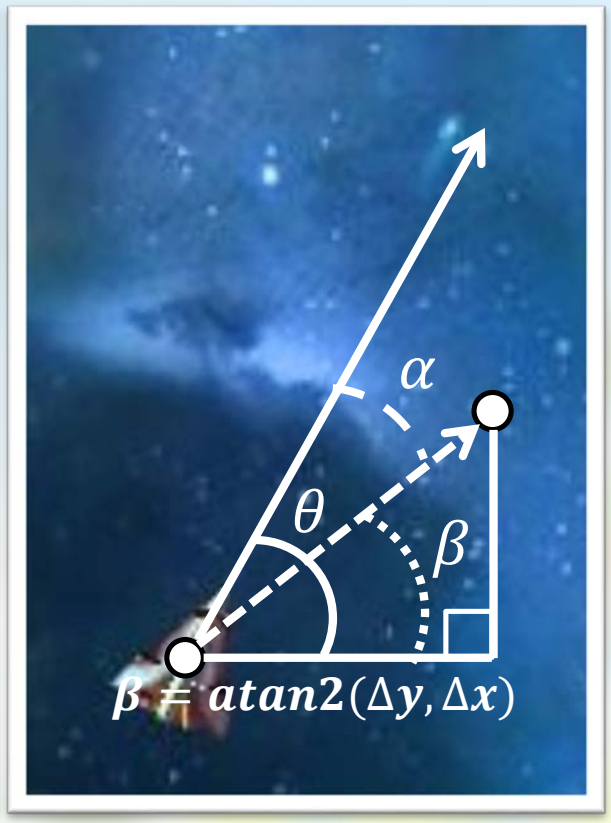
Following a target revisited

- Last time we talked about following a target
- Now we can add orientation:
  - Example -001_following_a_target

Rotation **towards** a target?





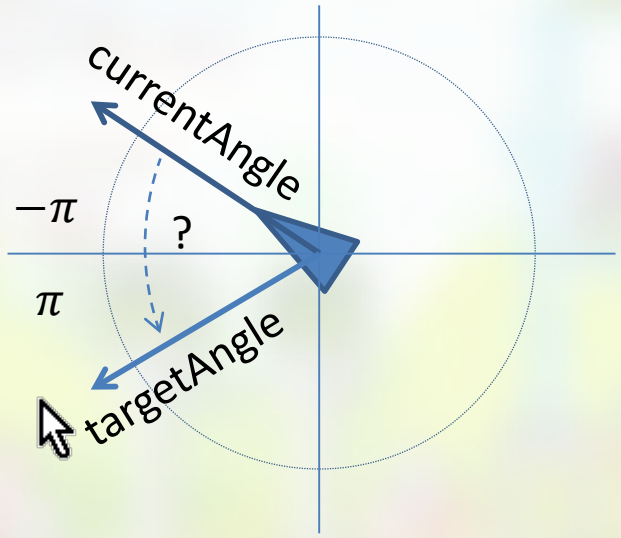


Much harder than it seems…

+005_rotating_towards_a_direction

Rotating **towards** vs **to**

- Rotating towards "eases" to target rotation instead of just setting it, in pseudo:
  - if (targetAngle > currentAngle) currentAngle++;
  - if (targetAngle < currentAngle) currentAngle--;
- Issue is that atan output range is from $(-\pi, \pi)$
- Fix is part of optional assignment

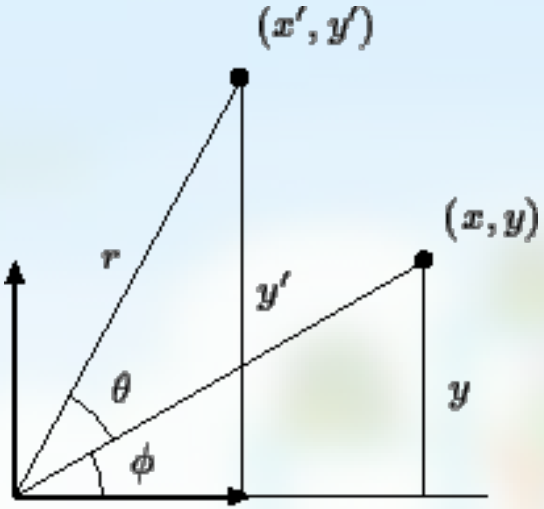


2D rotation

Using trigonometric functions to derive a 2d rotation formula

2D Rotation, naive "wrong" approach

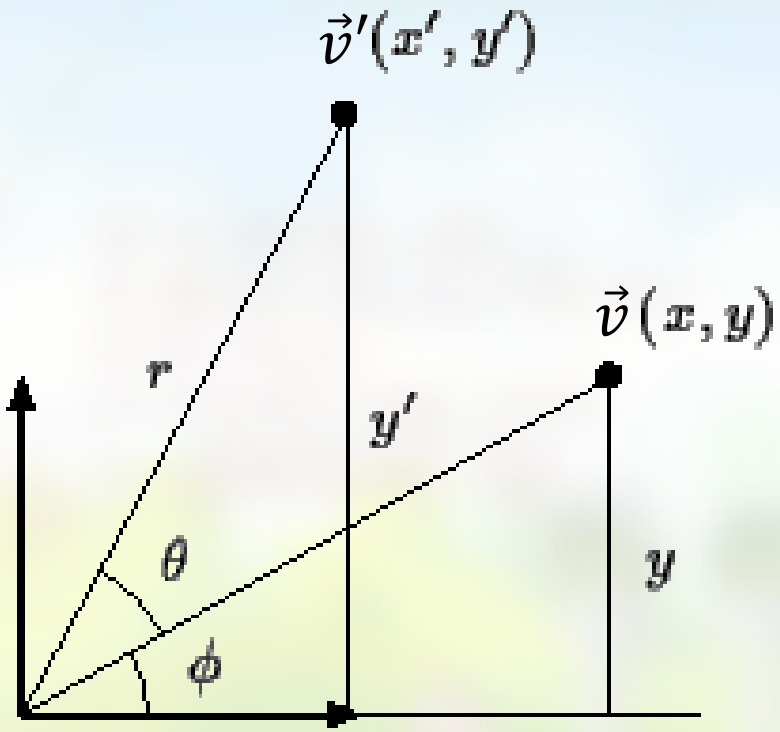
- The slow performance-intensitive option:
  - $currentAngle = Atan2(y, x)$
  - $r = \sqrt{x^2 + y^2}$
  - $newAngle = currentAngle + \theta$
  - $x' = r * \cos(newAngle)$
  - $y' = r * \sin(newAngle)$



3 trigonometric functions, two squares and one square root.
Has to be **repeated** for each point that you want to rotate.
This is **not** the way to do it… although it works (try it for practice)

A better 2D Rotation formula?

- Given $\vec{v}$ and angle of rotation θ, can we derive a better/faster 2d rotation formula for $\vec{v}'$?

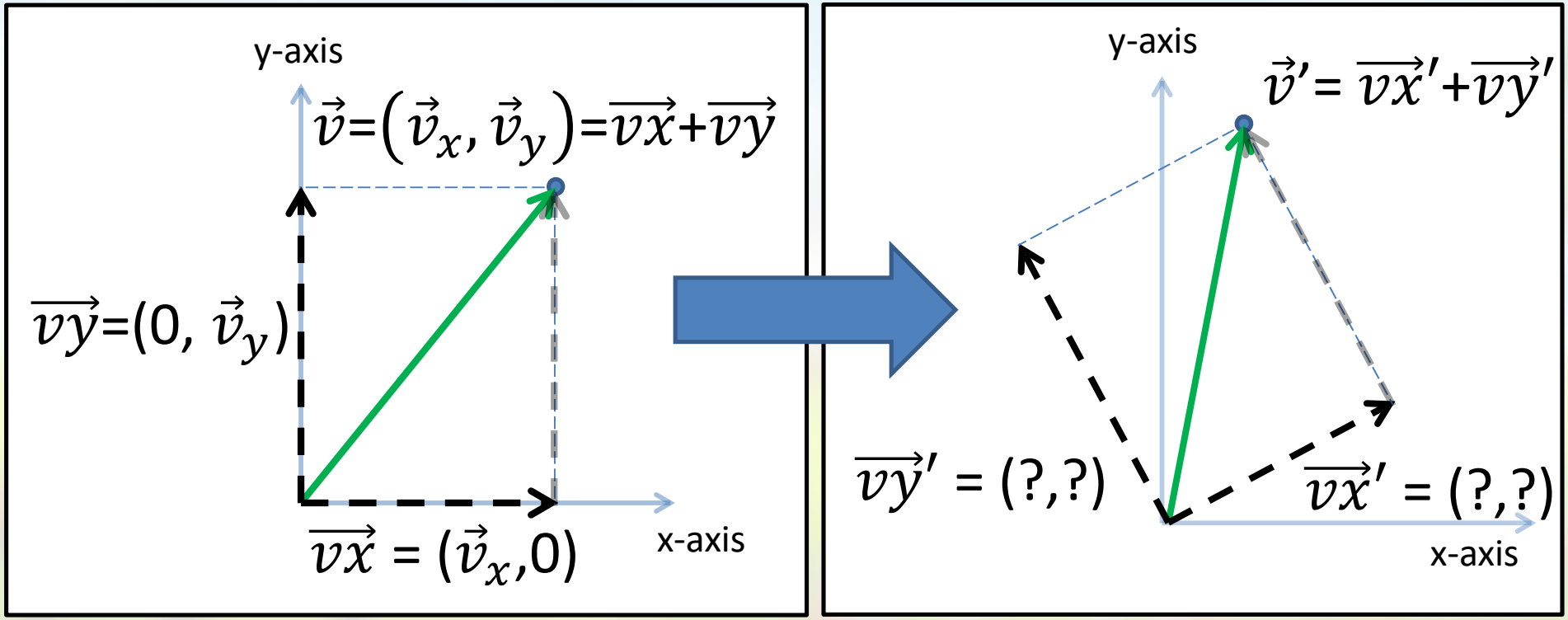


The idea

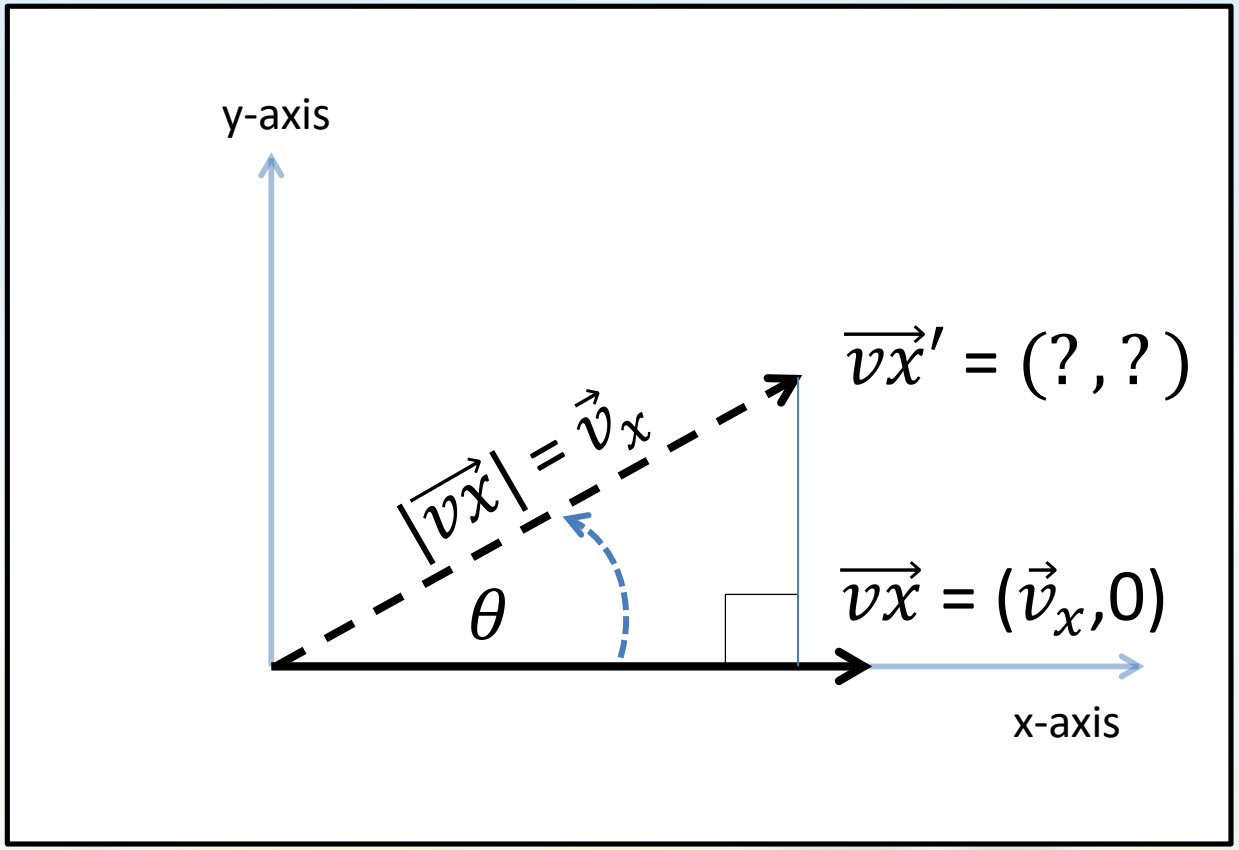
Decompose $\vec{v}$ into $\overrightarrow{vx}$ and $\overrightarrow{vy}$

Rotate $\overrightarrow{vx}$ and $\overrightarrow{vy}$ to $\overrightarrow{vx}'$ and $\overrightarrow{vy}'$

Add $\overrightarrow{vx}'$ and $\overrightarrow{vy}'$ back together to get $\vec{v}'$

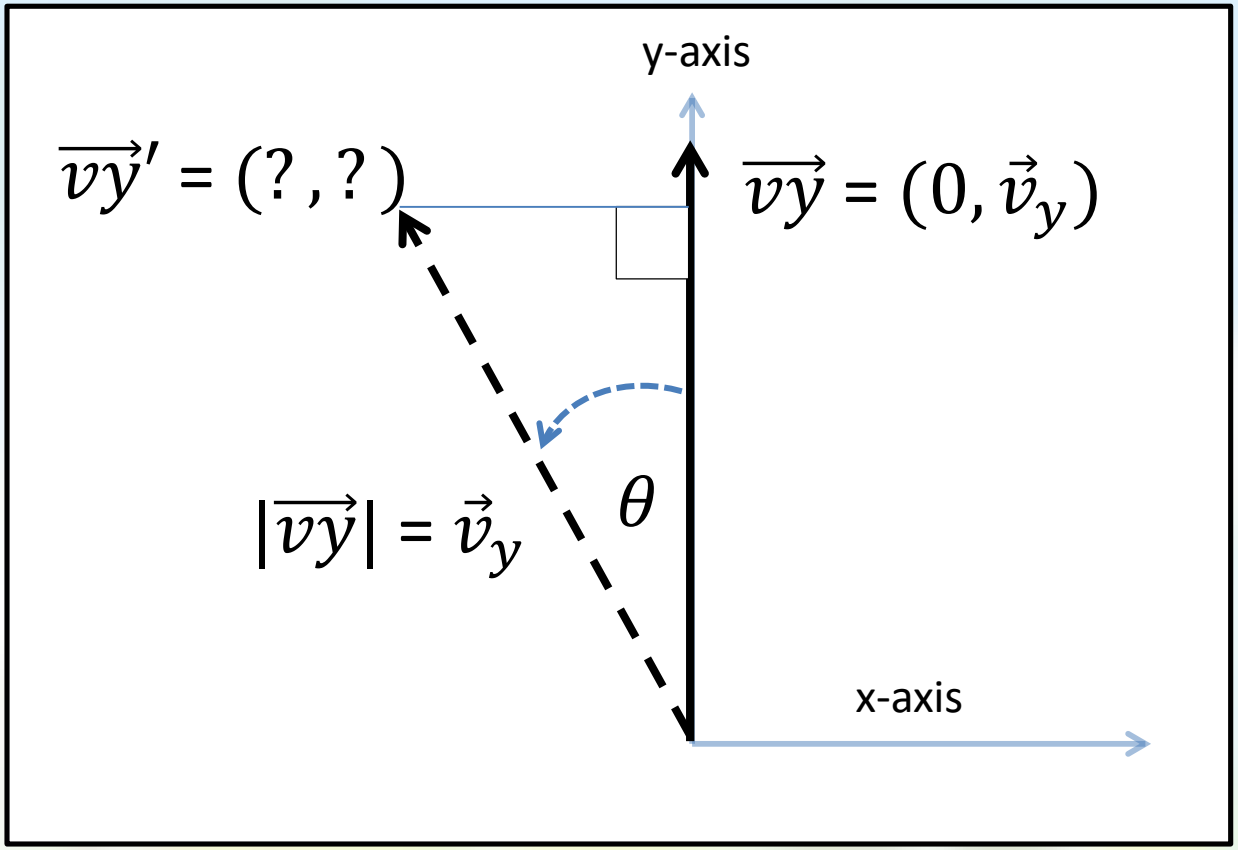


Calculating $\overrightarrow{vx'}$



$$\overrightarrow{vx'} = (\vec{v}_x * \cos\theta\,, \vec{v}_x * \sin\theta)$$

Calculating $\overrightarrow{vy}'$



$$\overrightarrow{vy}' = (-\vec{v}_y * \sin\theta\,, \vec{v}_y * \cos\theta)$$

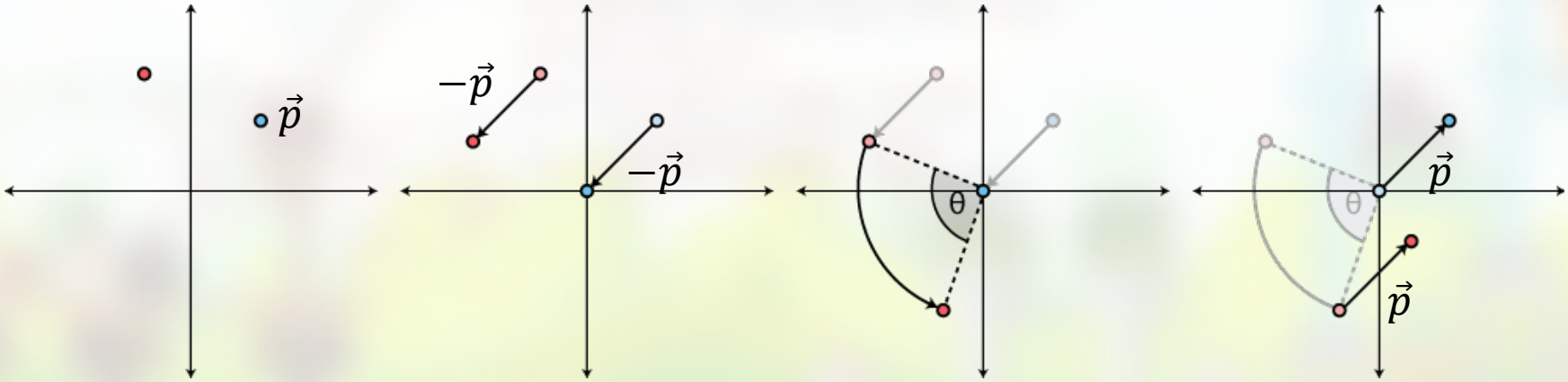
2D rotation formula for $\vec{v}$ and angle θ

- $\vec{v}' = \overrightarrow{vx}' + \overrightarrow{vy}'$ where,
  - $\overrightarrow{vx}' = (\vec{v}_x * \cos\theta\ , \vec{v}_x * \sin\theta)$
  - $\overrightarrow{vy}' = (-\vec{v}_y * \sin\theta\ , \vec{v}_y * \cos\theta)$
- gives:
  - $\vec{v}' = (\vec{v}_x * \cos\theta - \vec{v}_y * \sin\theta\ , \vec{v}_x * \sin\theta + \vec{v}_y * \cos\theta)$

- We have to calculate $\cos\theta$ / $\sin\theta$ only once!

Rotation around an arbitrary point?

- Rotation around point $\vec{p}$:
  - Translate everything over $-\vec{p}$
  - $\overrightarrow{p'}$ is now at (0,0)
  - Apply standard 2d rotation formula
  - Translate back over P



Rotation around an arbitrary point

- Usage:
  - Strafing/Orbiting NPC's?
  - Rotating a vector relatively
  - Assignment 2

Assignment 2

- Knowledge test
- Extending Vec2 even further…
- Building a tank and driving it…
  - Assets have been provided
  - Specific requirements/limitations

Divide and Conquer

We have no movable tank

We have a movable tank

Divide and Conquer

We have no movable tank

First requirement:

“forward/back accelerates/decelerates the tank in the direction the tank is facing, up to a maximum speed”

Implement a number which increases on Key.UP (print it!)
+ number reaches a max value
+ we can decrease it as well using Key.DOWN
+ number goes back to zero slowly on key release
+move tank in direction of current rotation using this speed

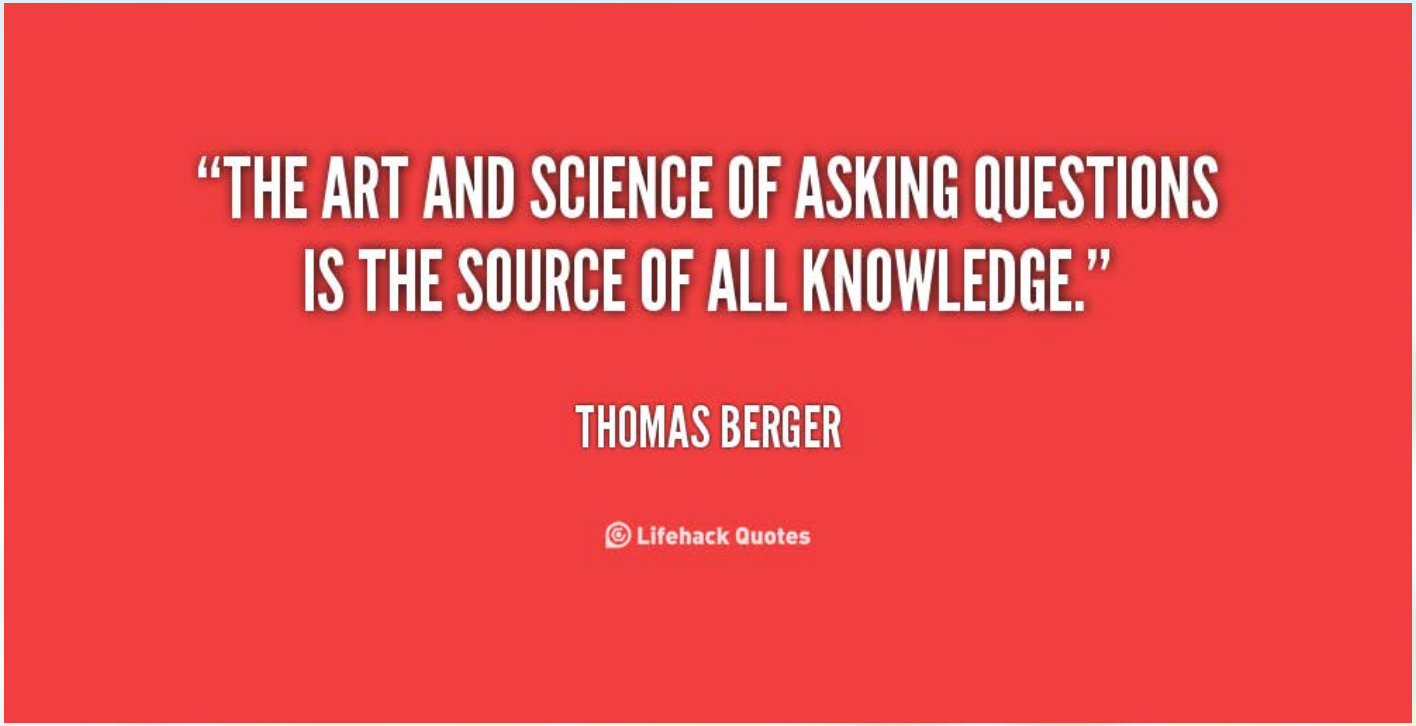
We have a movable tank

Divide and Conquer

- The number of steps to get somewhere is different for everyone
- What is your first step?
- What is your next step?
- Keep stepping

Next lecture

Newton’s Laws of Motion

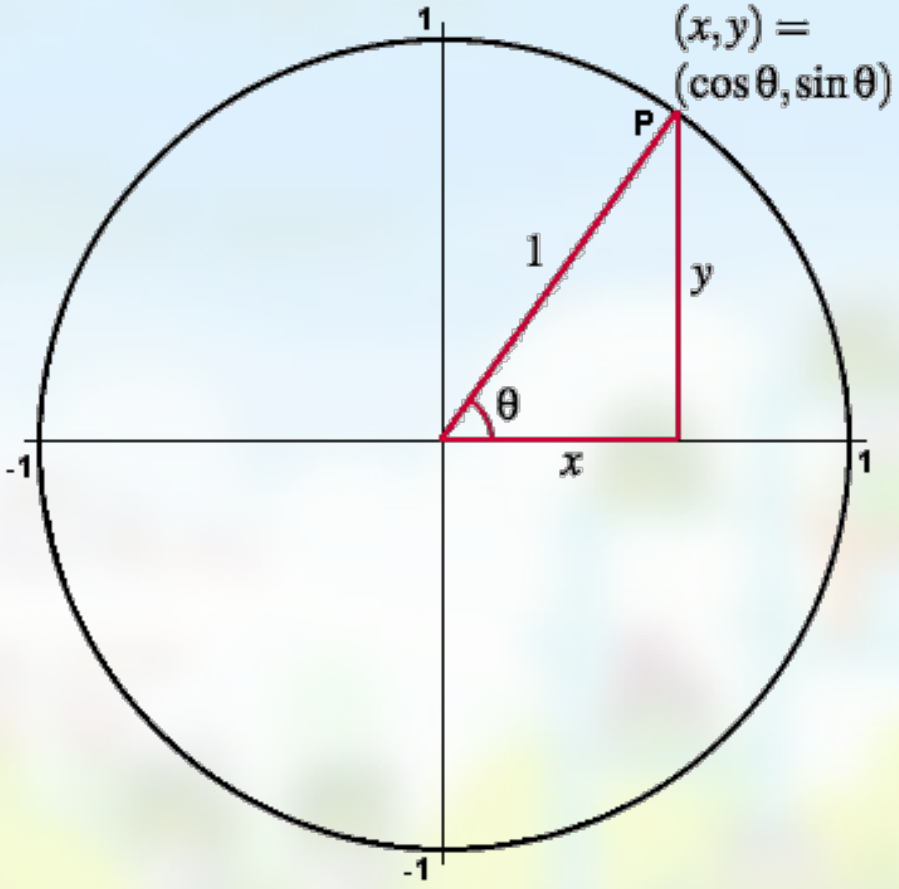

“THE ART AND SCIENCE OF ASKING QUESTIONS
IS THE SOURCE OF ALL KNOWLEDGE.”
THOMAS BERGER
© Lifehack Quotes

Detailed info

You should not read on, if you are already overwhelmed…

Cos and Sin equalities on Unit Circle

- $a^2 + b^2 = c^2$
- $x^2 + y^2 = 1^2$
- $\cos(\theta)^2 + \sin(\theta)^2 = 1$
- $\sin(\theta)^2 = 1 - \cos(\theta)^2$
- $\cos(\theta)^2 = 1 - sin(\theta)^2$
- sin (-θ) == - sin (θ)
- cos (-θ) == cos (θ)



Proofs

- Can we prove that if we calculate $\vec{v}$ as:

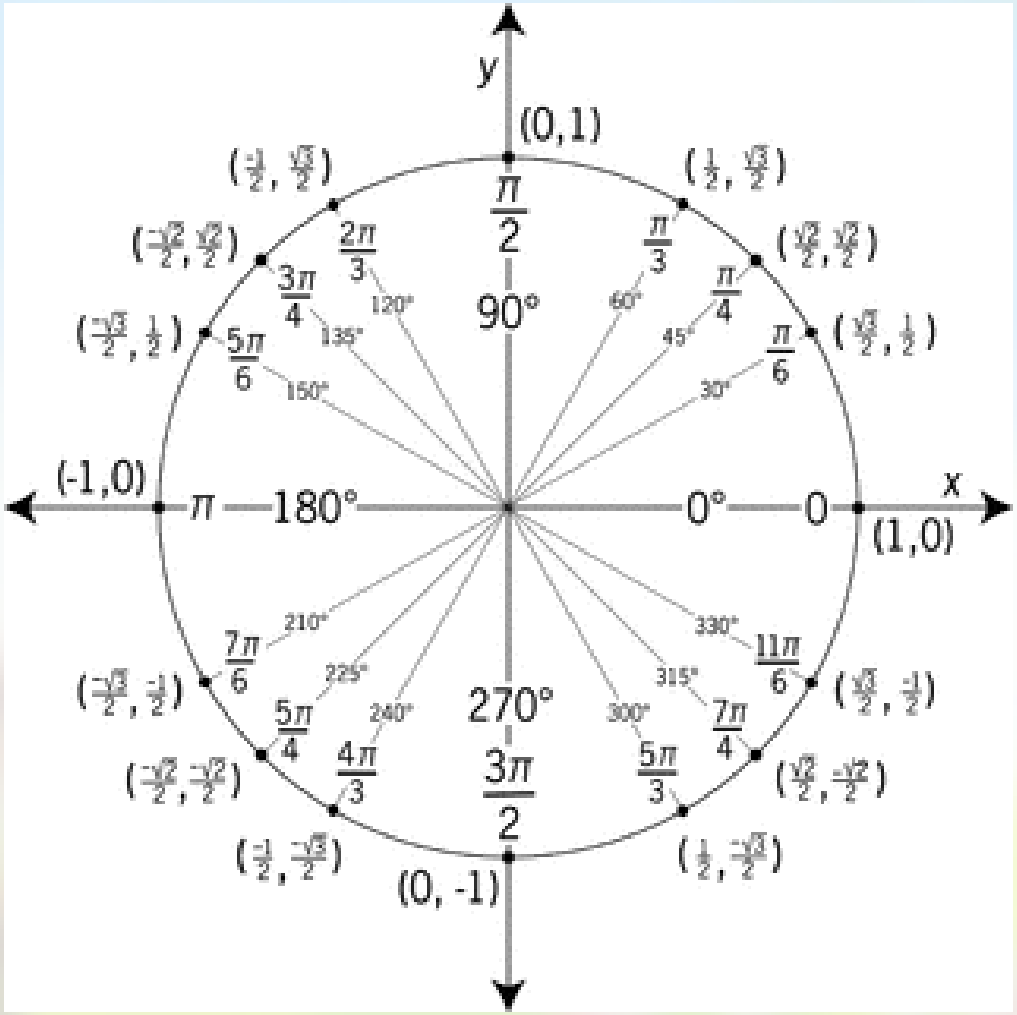
  (speed * cos(α), speed * sin(α))

  that the length of $\vec{v}$ is actually speed?
- Length of (speed * cos(α), speed * sin(α)) ==
- $\sqrt{(\text{speed} * \cos(\alpha))^2 + (\text{speed} * \sin(\alpha))^2}$ ==
- $\sqrt{speed^2 * (\cos(\alpha)^2 + \sin(\alpha)^2)}$ ==
- $speed * \sqrt{(\cos(\alpha)^2 + \sin(\alpha)^2)}$ ==
- speed * 1 == speed

Algebraic 2d rotation proof

- https://www.youtube.com/watch?v=a0LvqflQMx4
- http://www.siggraph.org/education/materials/HyperGraph/modeling/mod_tran/2drota.htm

Degrees / radians / Trig cheatsheet



Number bases

Short demo of how number bases work:

In $base_x$	$base_{10}$	
101_{10}	$1 * 10^2 + 0 * 10^1 + 1 * 10^0$	(= 101)
101_2	$1 * 2^2 + 0 * 2^1 + 1 * 2^0$	= 5
101_{16}	$1 * 16^2 + 0 * 16^1 + 1 * 16^0$	= 257
101_{60}	$1 * 60^2 + 0 * 60^1 + 1 * 60^0$	= 3660

π Boss