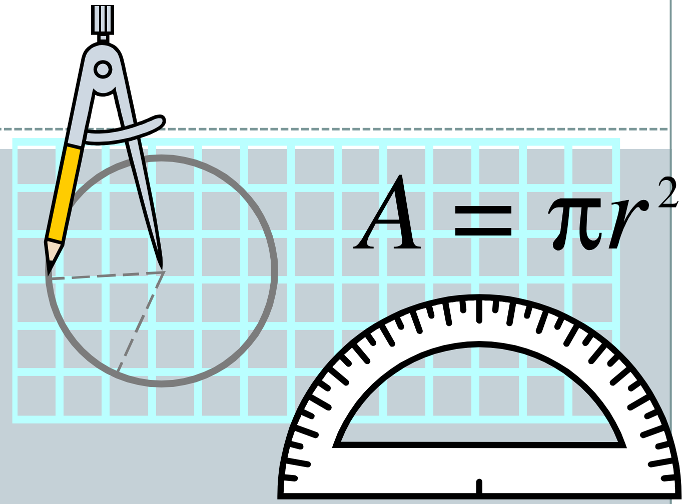
Trigonometry and Applications

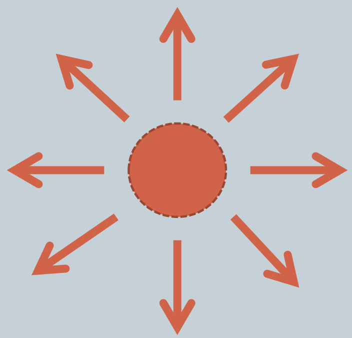


REFERENCES:

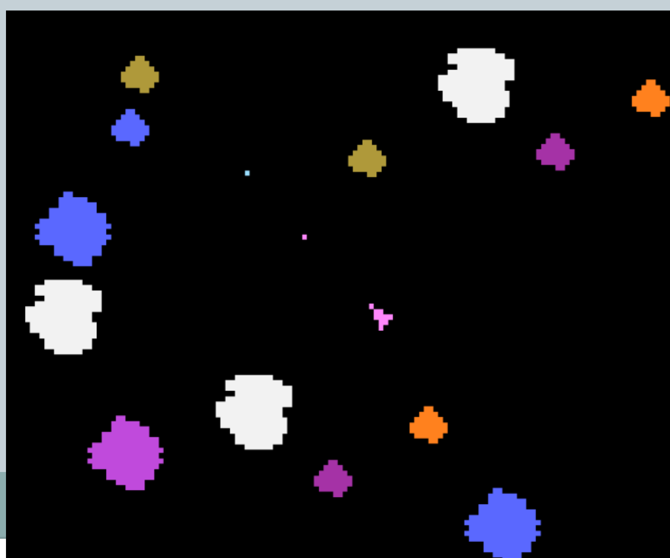
- HTTP://EN.WIKIPEDIA.ORG/WIKI/TRIGONOMETRY

Motivation

- Our character movements so far:



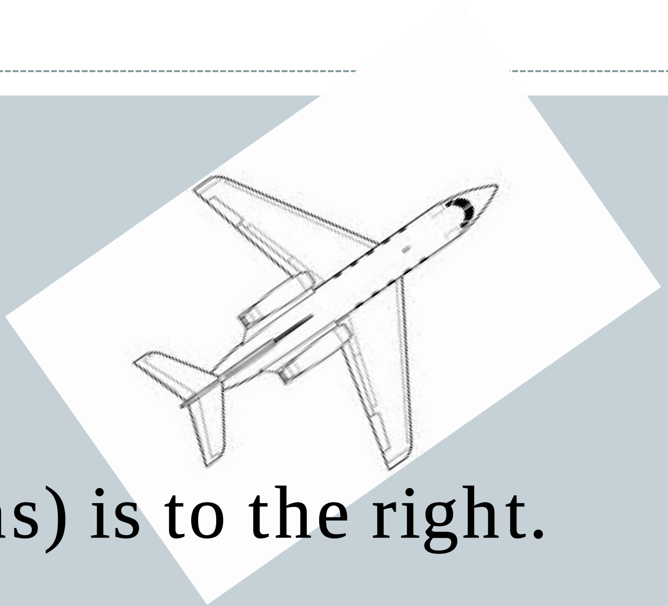
- Other types of movement:



The "new" type of movement

- Move **n** pixels (a distance) in this **direction**
- Does pygame have a secret function?
  - No! We need to devise a conversion.
- How do we represent a direction?
  - In 2d...an angle.

Angles (2D)



Two common systems:

- Degrees
- Radians

By convention, 0 (degrees / radians) is to the right.

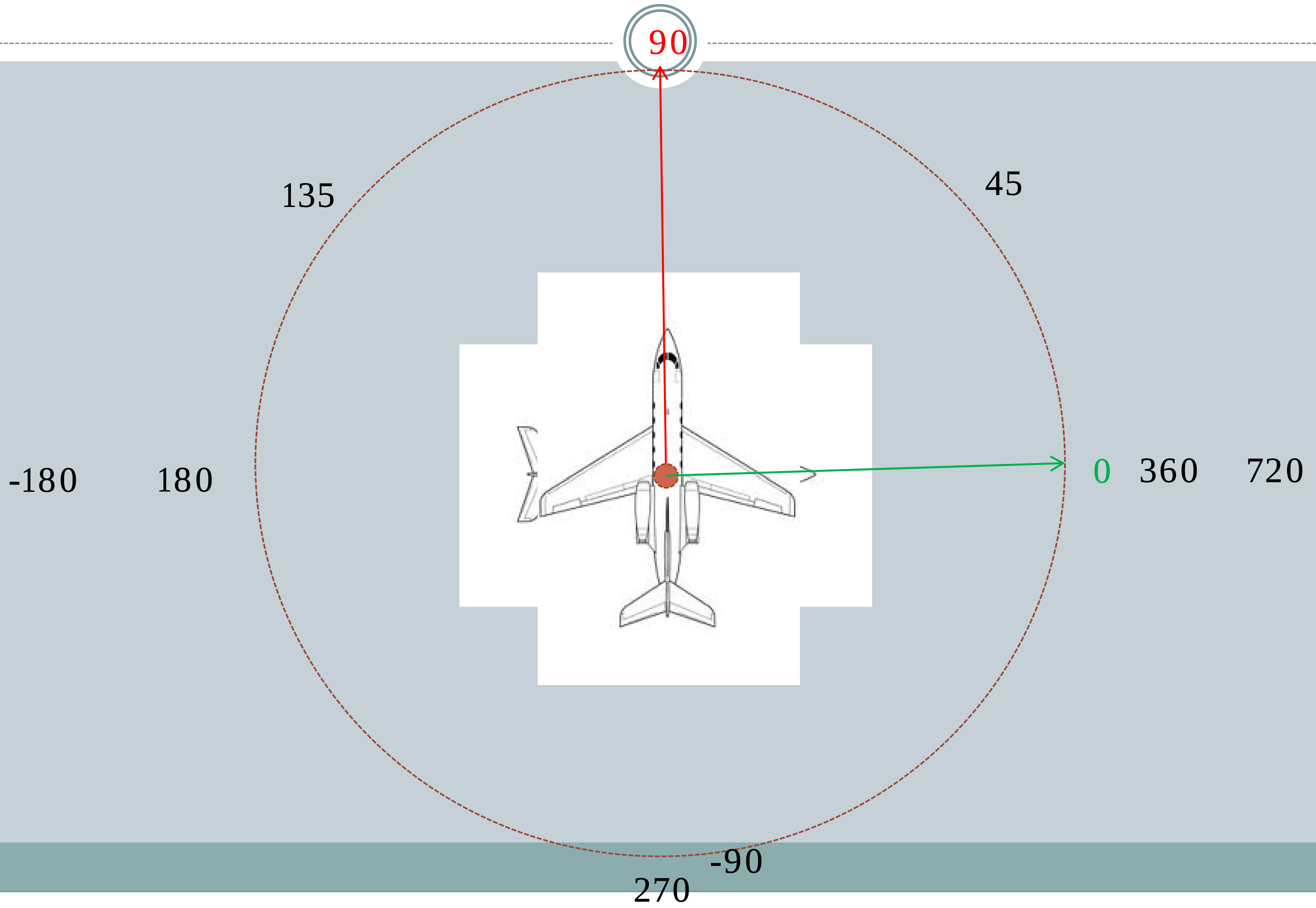
A measure of **rotation**:

- Negative is clockwise (by convention)
- Positive is counter-clockwise (by convention)

Also a description of **orientation**:

- How much we've rotate from the 0 (right) position

Angles (2D) Degrees



Angles (2D) degrees, cont.

- The number 360 is sort-of arbitrary
  - Evenly divisible by a lot of numbers (2, 4, 8, ...)
  - Loosely based on #days/yr
  - Babylonians used a **sexagesimal** number system (60-based instead of our 10-based system)
- In the radians system, the number has a physical meaning...

Angles (2D) radians

- What is π?
  - Common answer: 3.14159…
  - But what does it represent???
- Definition of π…

Circumference = 6.283"
Diameter = 2"

$$\frac{6.283}{2} = 3.14159\ldots$$

Circumference = 1.57"
Diameter = 0.5"

$$\frac{1.57}{0.5} = 3.14159\ldots$$

Angles (2D) radians, cont.

- So the <u>definition</u> of π is the **ratio** of <u>any</u> circle's circumference to its diameter.

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

- This is where this more familiar equation comes from.

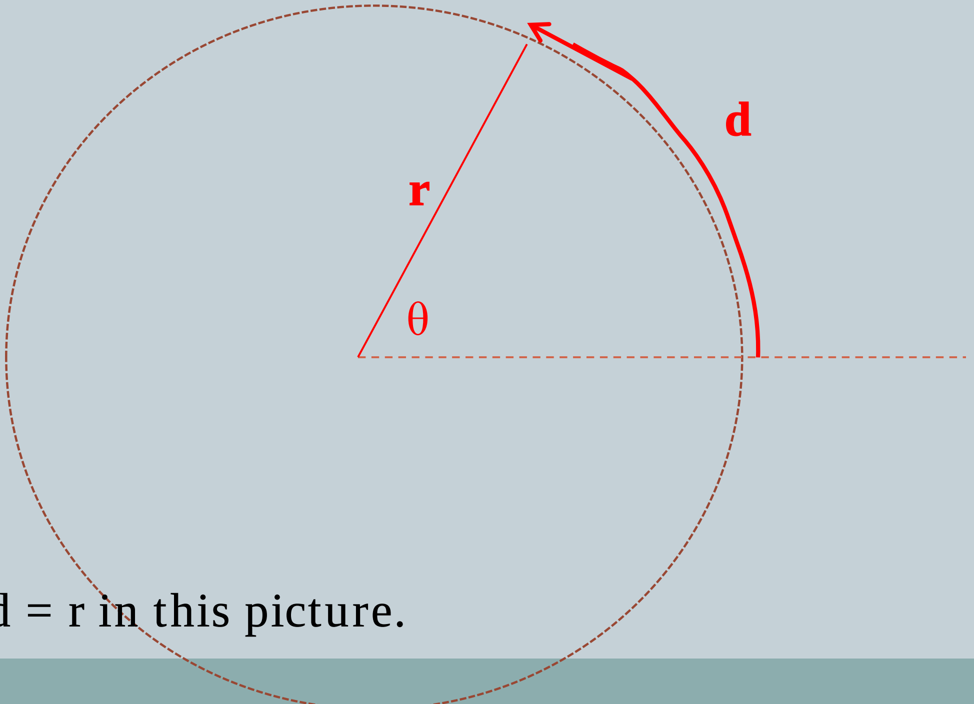
$$C = \pi D$$

Angles (2D) radians, cont.

- A *Radian* angle, θ is defined as

$$\theta = \frac{d}{r}$$

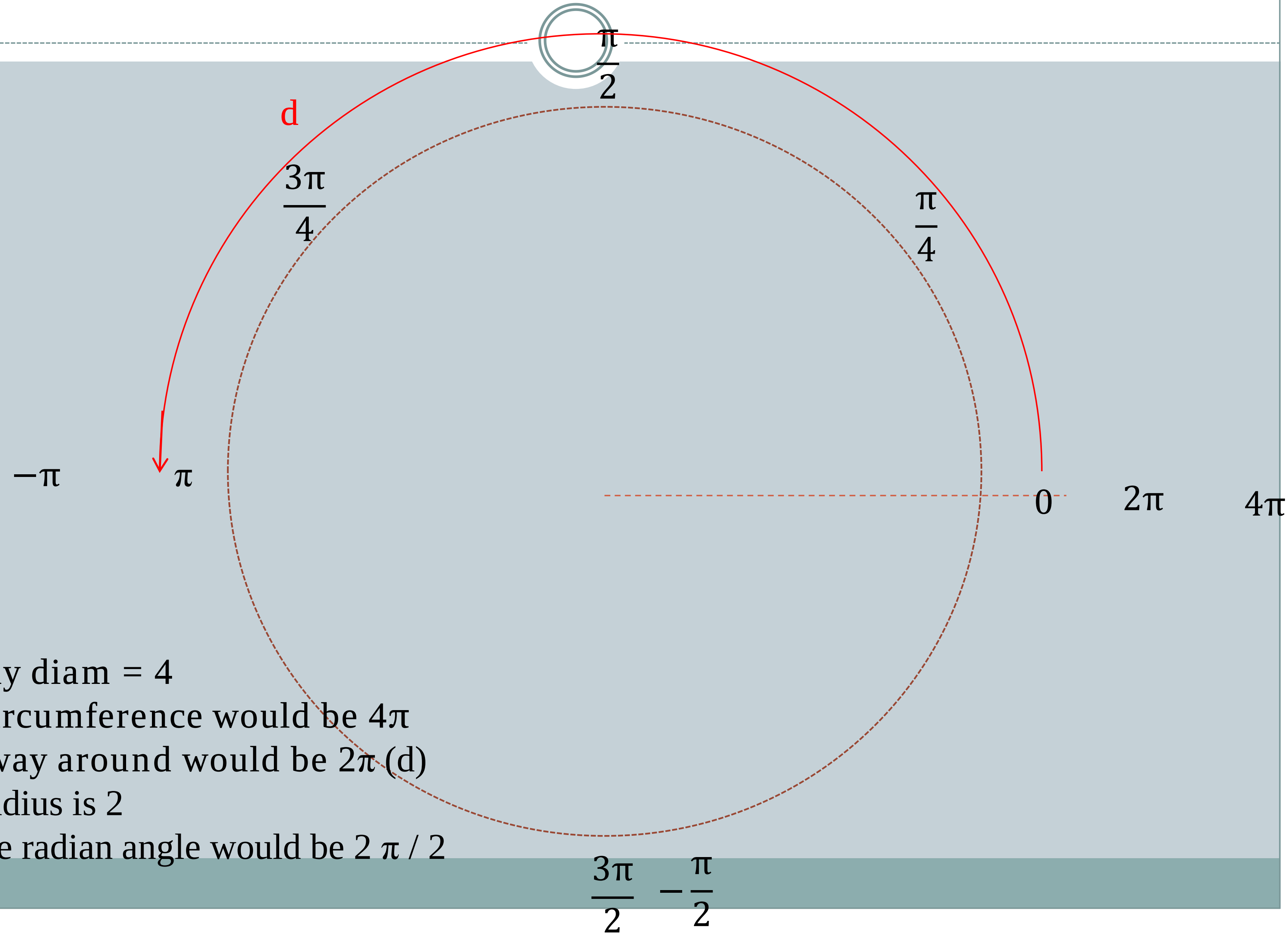
  - **d** is the the (partial) distance around a circle
  - **r** is the radius



What is one radian?

A: The angle such that d = r in this picture.

Angles (2D) radians, cont.



Let's say diam = 4

...the circumference would be 4π

... halfway around would be 2π (d)

…the radius is 2

…So the radian angle would be $2\pi / 2$

… π

Conversions

- Using ratios:

$$\frac{Degrees}{Radians} = \frac{180}{\pi}$$

- Solve for the unknown

49 degrees

$$\frac{49}{Radians} = \frac{180}{\pi}$$

$$49 = \frac{180 * Radians}{\pi}$$

$$49 * \pi = 180 * Radians$$

$$\frac{49 * \pi}{180} = Radians = 0.8552$$

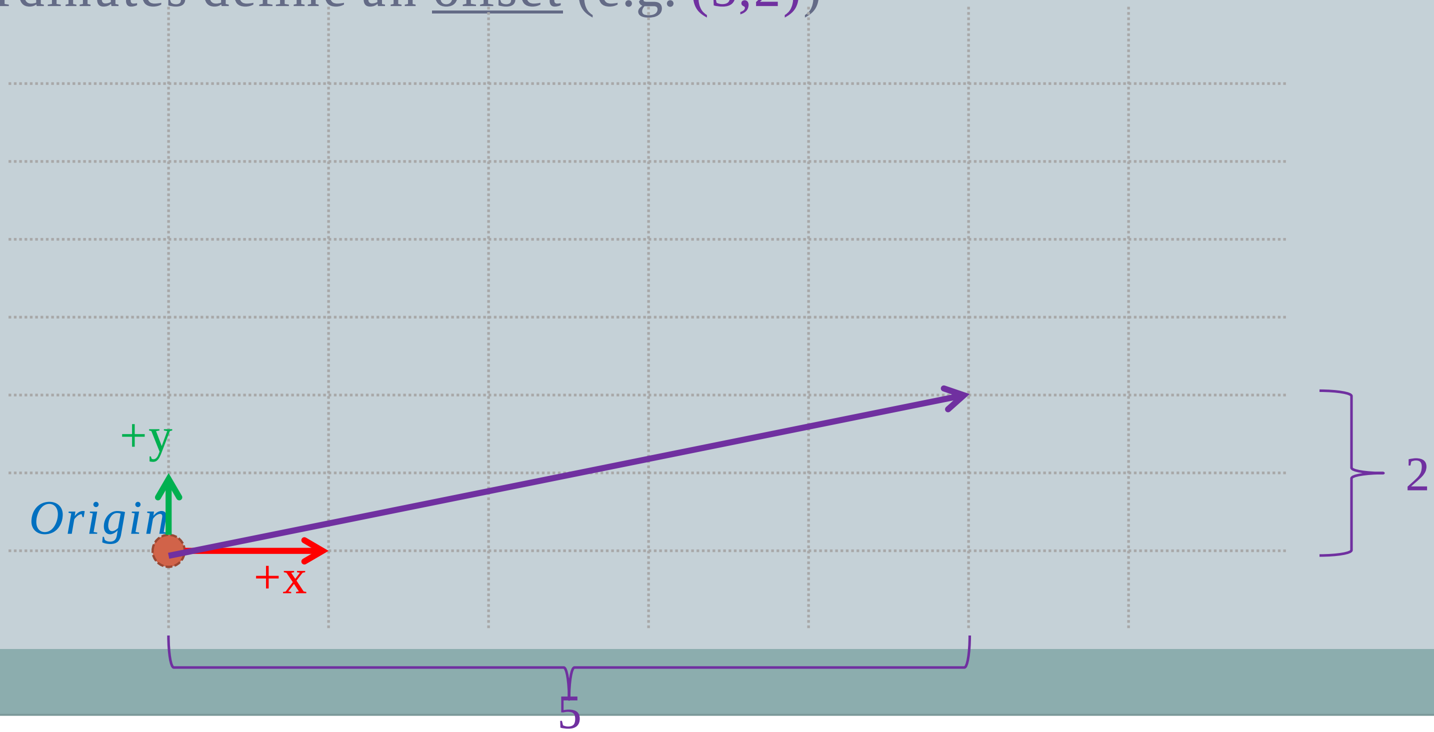
- Converting to Degrees is very similar.

Conversions, cont.

- ...Or just use the math functions.
  - math.radians(num)
    - Converts num (assumed to be in degrees) to radians
  - math.degrees(num)
    - Converts num (assumed to be in radians) to degrees
- Caution:
  - If you see the number 45.0 in your code, is it in radians or degrees?
    - You can't tell – neither can python.
  - Comments are very important!

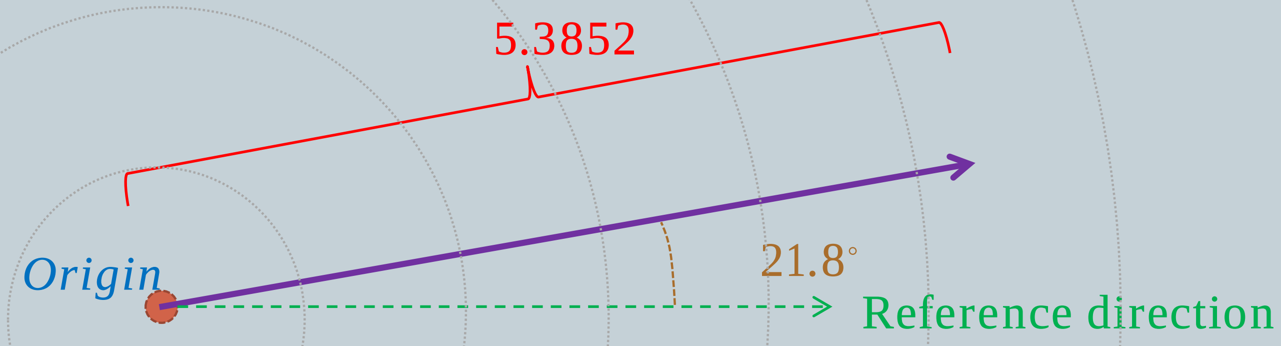
Cartesian (aka Euclidean, Rectangular) coordinates

- What we've been using so far.
  - Coordinate *System*
    - Origin
    - Axes (for 2d, 2 of them: conventionally named x and y)
      - With an associated scale (indicated by the gray dotted lines)
  - Coordinates define an <u>offset</u> (e.g. (5,2))



Polar coordinates

- A new way to specify an offset.
  - Coordinate System
    - Origin
    - A "reference" direction (conventionally right)
      - Scale in *any* direction is indicated by the gray dotted lines.
  - Coordinates define an offset. Now, they are:
    - A direction (an angle, relative to the "reference" direction)
    - A distance to move in that direction
    - Example: (5.3852, 21.8°)



Polar / Cartesian coordinates

- So...
  - (5,2) in Cartesian coordinates and
  - (5.3852, 21.8°) in Polar coordinates
- ...Represent the same offset!
  - (Flip back / forth on the previous slides – look at the purple arrow)
  - The first is useful for "Pacman-style" offsets
  - The second is useful for "Asteroids-style" offsets
- Problem: Pygame only uses Cartesian coordinates
- Solution: We need a way to convert
  - Initially from Polar=>Cartesian
  - We'll see later that Cartesian=>Polar has uses too.

Polar => Cartesian conversion

Initial assumption: distance is 1.0

	Cartesian	**Coordinates**
Angle (degrees)	x	y
0	1	0
90	0	1
180	-1	0
270	0	-1
15.4	??	??

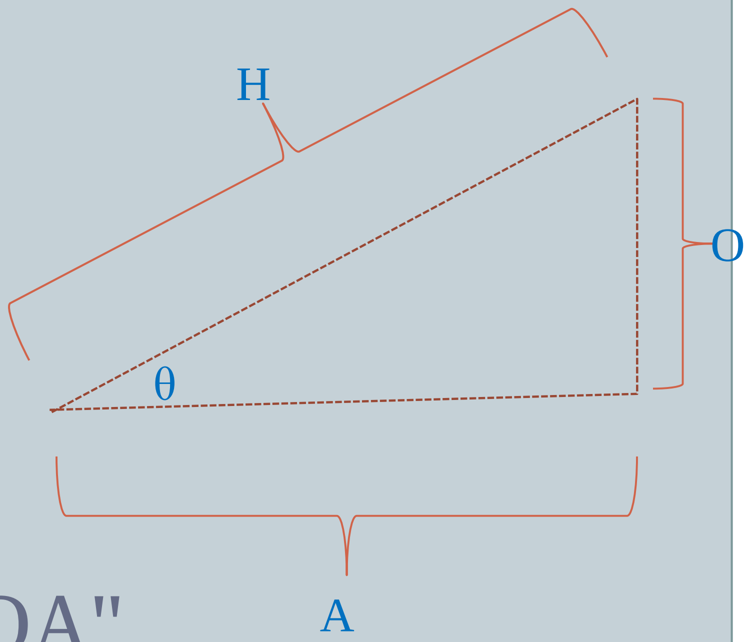
Trig to the rescue!

- Trigonometric function *definitions*:
  - Trigonometry in Greek means "Triangle measuring"

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

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

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



- Mnemonic: "SOH-CAH-TOA"

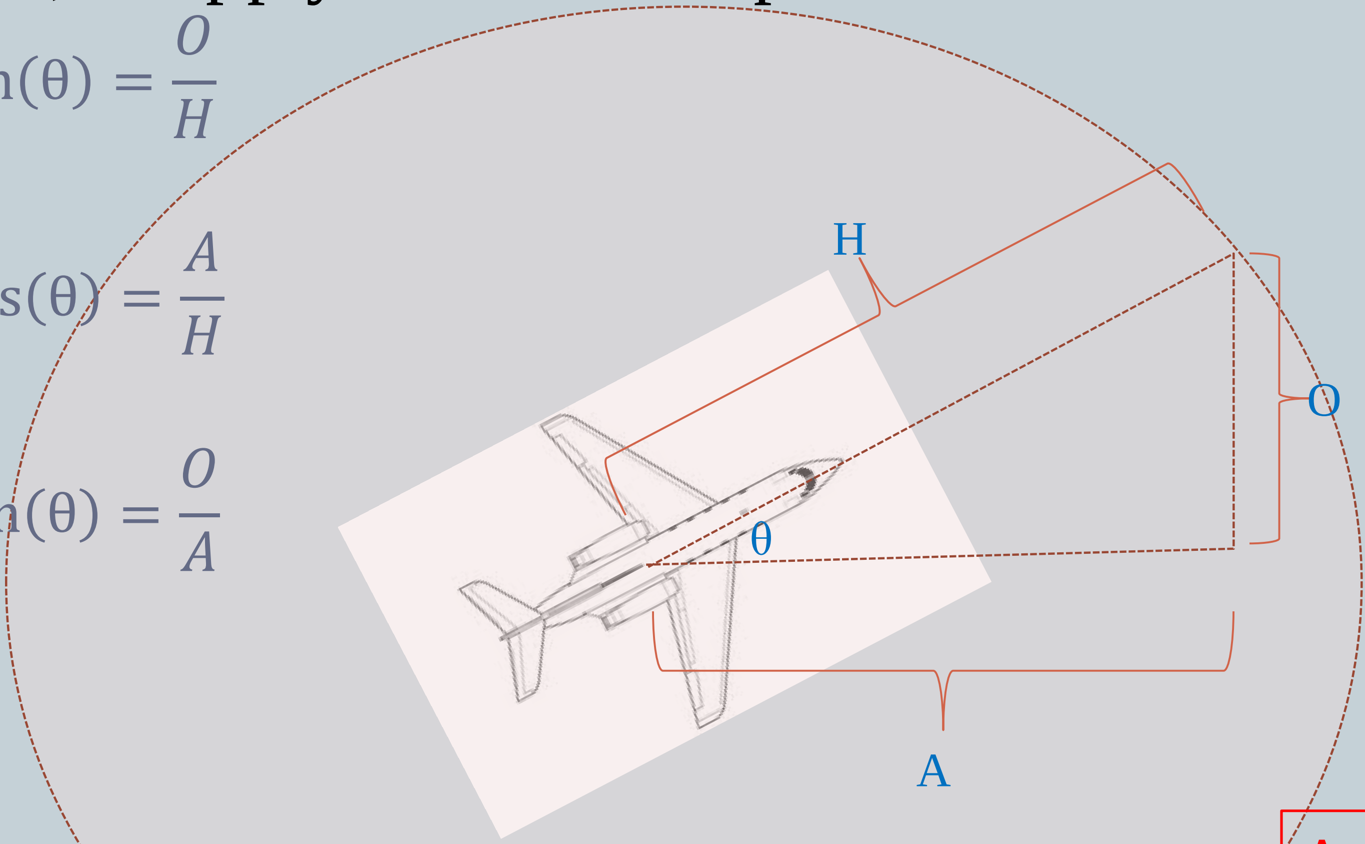
Trig functions

- Now, to apply these to our problem:

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

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

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

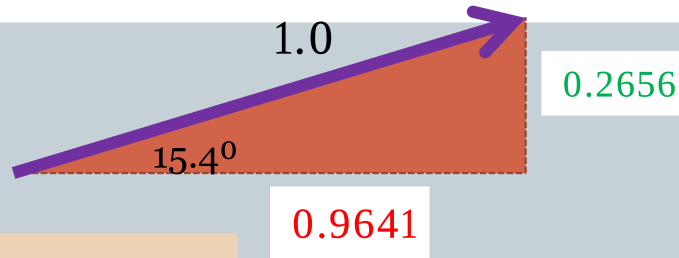


A=H*cos(θ)
O=H*sin(θ)

H is the distance we want to move forward
A is the amount to add to our x-position
O is the amount to add to our y-position (note pygame's y axis)
(A,O) is the Cartesian equivalent of (H, θ) in polar coordinate.

Polar => Cartesian conversion

Back to our original problem...
Initial assumption: distance is 1.0
This is the **hypotenuse's length**



Angle (degrees)	Cartesian x	Coordinates y
0	1	0
90	0	1
180	-1	0
270	0	-1
15.4	0.9641	0.2656

The length of the adjacent side's length (which we don't know)...
...but we can calculate

A = H * cos(angle)
= 1.0 * cos(15.4)
= 0.9641

The opposite side's length this time

O = H * sin(angle)
= 1.0 * sin(15.4)
= 0.2656

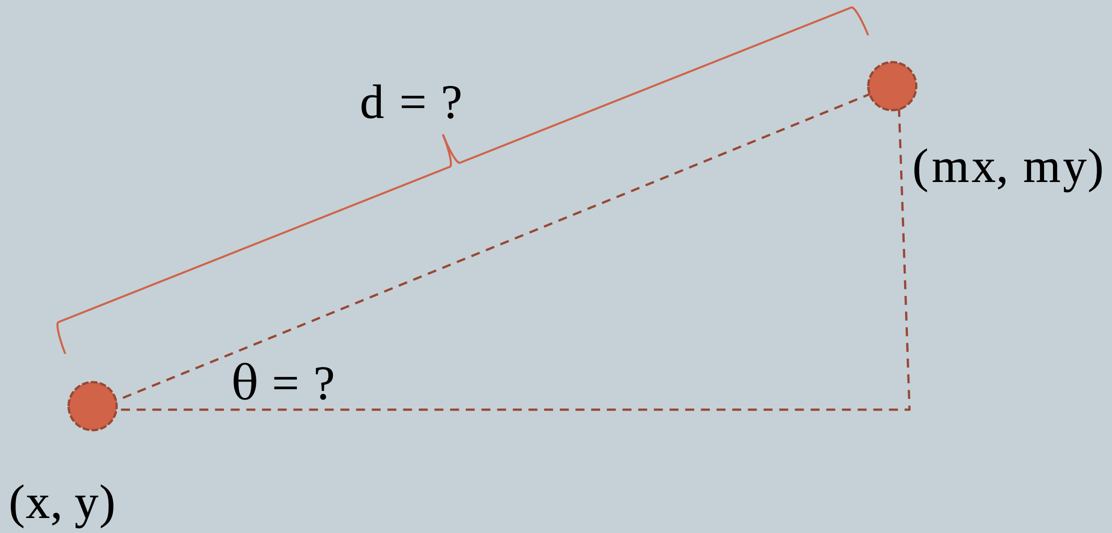
Example

- [Moving object, firing projectiles]
  - [Add a "rotate-able" object]

Cartesian => Polar Conversion

- Why?
- Sample Problem:
  - Character is at (x, y)
  - Mouse is at (mx, my)
  - Calculate the distance and angle towards mouse
  - [Picture]

Cartesian => Polar, cont.



[Develop solution together]

Cartesian => Polar

- [Make the “tank” point towards the mouse]