COMP 1010- Summer 2015 (A01)

Jim (James) Young

young@cs.umanitoba.ca

jimyoung.ca

Hello!

James (Jim) Young
young@cs.umanitoba.ca
jimyoung.ca

office hours T / Th: 17:00 – 18:00
EITC-E2-582
(or by appointment, arrange by email)

Assignment posted

Drop box

Defining variables

You can define multiple variables at a time

type var1, var2, var3…;

Eg.,

int pointX, pointY;

floating point variables

```
size(500,500);
float percent = 0.3;
float targetX = percent*500;
line(0,250,targetX,250);
```

Trigonometry and processing

Processing uses **radians** not degrees

Circle goes from 0..2PI

PI?

-> 180 degrees

PI/4?

-> 45 degrees

Trigonometry and processing

New constant: PI (all caps)

New way to show you commands

type commandName(type parameter);

float sin(float radians)

float cos(float radians)

float tan(float radians)

More trigonometry

We have inverse functions-> $\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$

Also called arccos, arcsin, arctan.

float asin(float ratio)

float acos(float ratio)

float atan(float ratio)

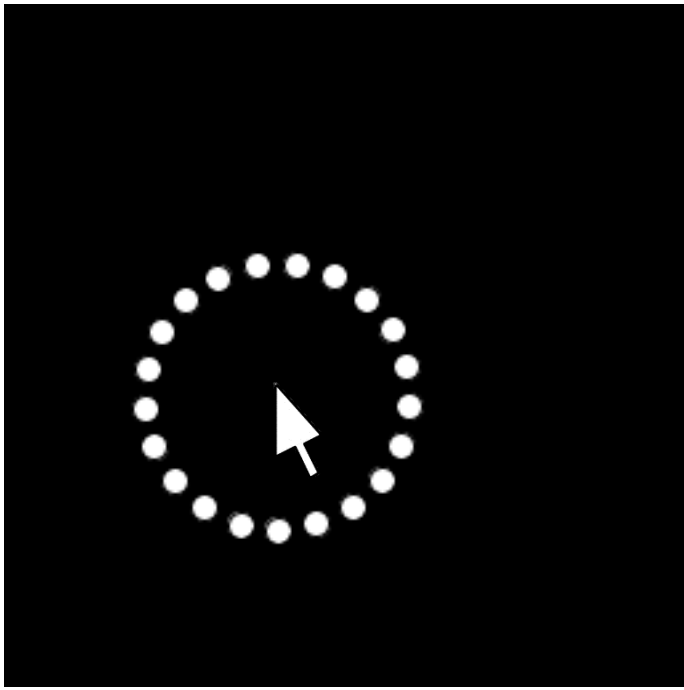
ADVANCED: atan2 (see notes)

Back to our mouse orbiter!

Start with simple static case

Let’s pick an angle, theta, and start at some point

We also need a radius – how far the ball will circle



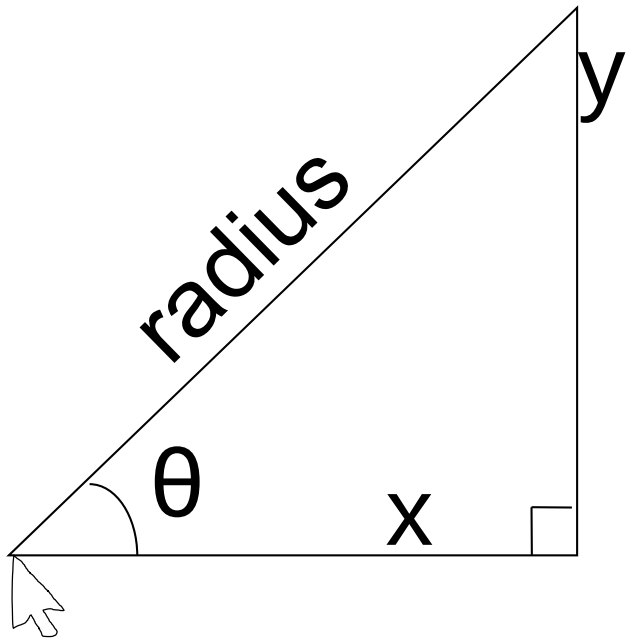
Given a radius and an angle, what is the x, and y?

$\sin(\theta) = y/\text{radius}$

$y = \sin(\theta) * \text{radius}$

$\cos(\theta) = x/\text{radius}$

$x = \cos(\theta) * \text{radius}$



It worked!! We can now place a ball at some distance and angle from the mouse

How to make it animate?

What did we do for mouse bubbles?

Every time we draw, adjust angle by a little delta

What about the angle getting too big?

mod and floating point is messy…

angles wrap around we're safe!

At home exercise:

Here is an extension you can try at home: expand this to having multiple planets orbiting the mouse at different speeds. Here are some helping steps:

- rename the variables to start with p1 for planet 1
- copy all your variables for each planet. Try three planets
- make each planet move at a different speed. What if one moves at a negative speed??

A closer look at random…

Let’s lookup random in the reference

Random **returns a float**. Means the data gives you is floating point

Can only be stored in a floating point variable

block moving randomly around a screen

A block that jumps around

Globals

- block position
- block size
- how fast it can move

Block moving randomly…

What is random? How much the block moves. Not the block position

```
float moveX = random(blockSpeed);
```

Then we add the movement to the position

```
blockX = blockX + moveX;
```

The block can only move right!

How can we also make it move negative?

```
random(blockSpeed*2)-blockSpeed;
if blockspeed is 5..
random(5*2)-5;
```

- Generate double the range .
- Subtract the range from it.
- If random gives us 0
  - 0-5 is -5
- If random gives us 5
  - 5-5 is 0
- If random gives us 9
  - 9-5 is 4

It moves off the edge of the screen…

Use min and max to cap it!

Make it ALSO move toward the mouse!!

X first…

How to get how much toward the mouse?

mouseX – blockX

if mouse is to the right, we get how many

if mouse is to the left, we get neg. how many

How to move toward the mouse?

add the difference to the block

Oh no! We just move TO the mouse

Move **toward** the mouse

Divide the difference by some number….

common basic arithmetic

Processing has a bunch of arithmetic shortcuts that makes your life easier… as long as you understand them

common basic arithmetic

it is very common in programming to increase, decrease, or multiply a variable by a certain amount:

variable = variable + 5;

variable = variable * 10;

variable = variable / 4;

combined operations

```
variable = variable + 5;
        variable += 5;
variable = variable * 10;
        variable *= 10;
variable = variable / 4;
        variable /= 4;
variable = variable – 5;
        variable -= 5;
```

```
variable = variable % 2;
        variable %= 2;
```

examples..

```
price = price * SALES_TAX;
price *= SALES_TAX;

posX = posX – moveX;
posX -= moveX;

payment = payment / NUMBER_OF_PEOPLE;
payment /= NUMBER_OF_PEOPLE;
```

another shorthand: incrementors and decrementors

straight forward: what does increment and decrement mean?

variable = variable + 1; // increment variable

variable = variable – 1; // decrement variable

one short hand: variable += 1; variable -= 1;

incrementor: variable++

decrementor: variable--

incrementor / decrementor example

```
int i = 5;
i++;
i++;
i--; // what does I equal?

int i = 5;
i++; // i now equals 6
i++; // i now equals 7
i--; //i now equals 6
```

trivia: why is c++ called c++?

a *very* popular programming language was called C, but it did not have object oriented programming.

some people updated C to include object oriented programming, and called it c++. it means c + 1 ☺

Boolean!!

(not bullion)

the **boolean** data type

note: the boolean data type stores only two different values: **true** and **false**.

boolean isStudentSleeping;

isStudentSleeping = true;

true, false

boolean jimIsRich = false; // :(

true,false are not numbers.. they're booleans.

but what use is this?? just storing true and false??

we need to learn two things to make booleans useful:

conditionals: code that only runs if a boolean is true or false, e.g.,

```
if (jimIsRich) { stealHisMoney(); }
```

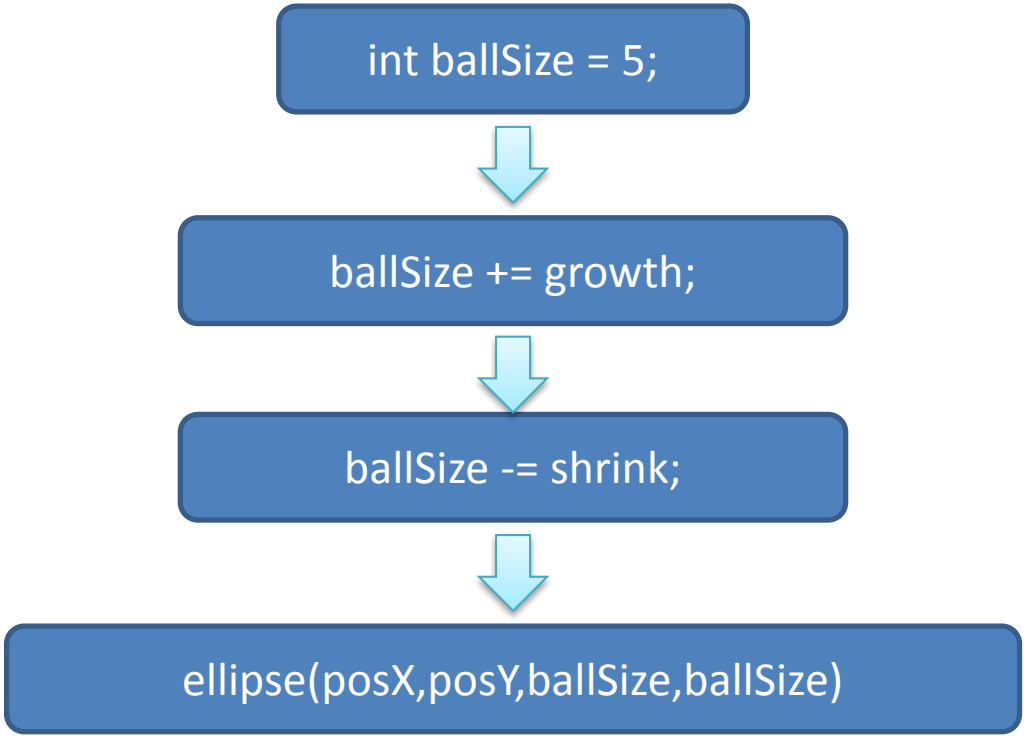
boolean operations : calculations that use and result in booleans!

the boolean data type is used in computer logic – it gets confusing fast

conditionals!!

using booleans to have code run
ONLY if certain conditions are met

so far, our programs run in a straight path:



start at the top

work down.

what if we only want to grow or shrink the ball, but not both? Depending on circumstances?

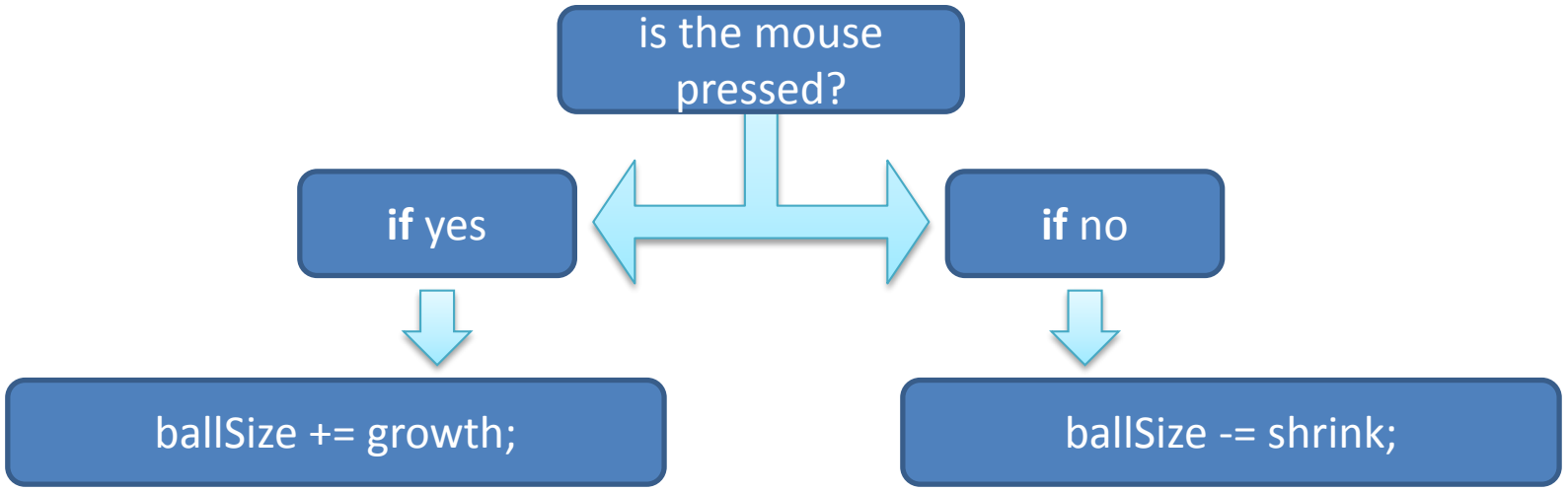
what if we only want to draw the ball if it is not too small?

a fork in the road!

we need a way to choose which lines of code are run and which are not run,

and under what conditions they are run

this is **conditional programming**, and achieved using **if** statements



the **if** statement!

Processing uses the **if** keyword, and a **boolean** test, and executes a block of code IF AND ONLY IF the boolean test is true.

```
// .. your program…
if (booleanValue)
{
   // then do this code
}
// more program…
```

this code is ONLY run if the booleanValue is true. otherwise, it is skipped completely!!

Processing has some great pre-defined booleans

```
boolean mousePressed;
boolean keyPressed;


e.g.,
if (mousePressed)
{
  // do stuff
}
```