today

Due: ex 1, critique
topics: variables, conditions
introduce ex 2: Free Patterns!
Student Presentation
Reading Ch 4, 5

Monday Jan 18 - no class.

Wednesday, Jan 20

due: Paper sketches for Ex 2
code: Free Patterns!
student presentations: see schedule online

review

flow with setup and draw

order of operation

command-t tidy up

iterate your design on unruled paper FIRST!

this is the same thumbnail practice you do in all your classes in design.

basic functions

```
void setup(){
  //runs only once
  //such as size, background
}

void draw(){
  //code that repeats forever
  //such as shapes and movements
}
```

variables

Variables store data for later use in your program.

variable (data) types

numbers: **`int (integer number), float(number with decimals)`**

numbers: **`byte, long, double`** - difference in bit size.

letters: **`char`**

words, including spaces: **`String`**

true or false: **`boolean`**

colors

positions

fonts

...and more

using variables

use variables to:

- save yourself from typing the same thing over and over again.

(example: three circles all have the same y position and diameter.)

```
size (480, 120);
int y=60;
int d=80;
ellipse(75, y, d, d);
ellipse(175, y, d, d);
ellipse(275, y, d, d);
```

- the changing nature of an element in your design.

(example: a circle moves from left to right => its x position changes.)

variable names

Case sensitive!

myContainer, **my_container**, **my_Container**

theScore, **the_score**, **the_Score**

Use only letters, numbers and underscores

Cannot start with a number

Must be unique names within your program

Cannot be a keyword (such as *rect, ellipse, setup* or *draw*)

keywords in Processing

indicated with color

note here: i just named y and d as my variables; it's in black. black is safe!

```
size (480, 120);
int y=60;
int d=80;
ellipse(75, y, d, d);
ellipse(175, y, d, d);
ellipse(275, y, d, d);
```

consistency

Be consistent in naming conventions:

can use: user_name

the convention: userName

easy on yourself and easy on others

design variable names so that you can understand in a flash

think of others who will be reading your code

try to make it shorter rather than longer

What is a good variable name for temperature?

t

temp

temperature

roomTemp

roomTemperature

follow the rules

(even though that makes me cringe...lol)

```
int temp;

int int;

boolean 1temp;

boolean heater;

int last temp;

int lastTemp;

int last_temp;

int last-temp;
```

```
int temp;

int int;              //can't use a keyword

boolean 1temp;        //can't start with number

boolean heater;

int last temp;        //can't use space

int lastTemp;

int last_temp;

int last-temp;        //can't use dash
```

variable declaration and initialization

If the entire program shares the same variable, we put it before setup() to declare what's called "global variables" - variables used throughout the program.

```
//declaration only
int x;

//initialization
x = 5;

//declaration and initialization
int x = 5;
```

```
int circleX=50;

void setup() {
  size(640, 360);
}

void draw() {
  background(50);

  fill(255);
  ellipse(circleX, 180, 24, 24); //draw circle's x position according to the circleX value.

}
```
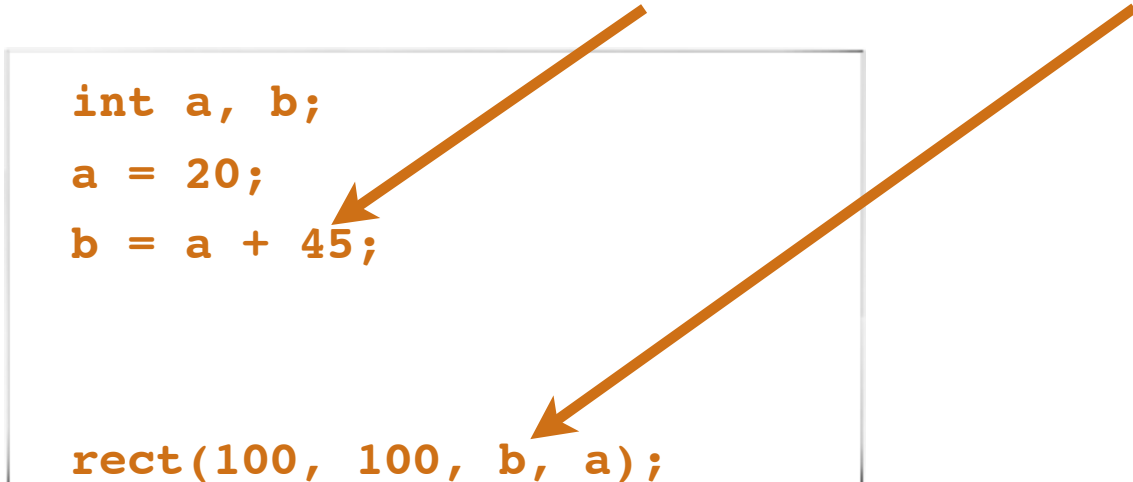
variable: assign values with =

```
int a;
a = 3;
a = 4*7/2; // can be an equation or expression
```

Variables can be used in expressions or as parameters

```
int a, b;
a = 20;
b = a + 45;


rect(100, 100, b, a);
```

let's try it!

sketch_4_Variables.pde

use variable to draw a circle.

User Defined Variables

step 1 declare the variable
1. specify data type (number, a string of characters, etc)
2. give a name
can be absolutely anything (blueberrypancake, wackyshoes, etc)
<u>rule of thumb:</u> name that defines purpose.
lower case is the convention

int circleX;

step 2 define its initial value
circleX=50; //This means, we want circleX to begin at 50 pix from the left.

step 3 use the variable
so far - our codes have been "function calls", example -
line(50, 100, 200, 100); //not using variable
ellipse(circleX, 180, 24, 24); //using variable
function(arguments)semicolon

now we are going to learn a new thing-
something=something; //This means, circleX=some math operation;
circleX=width/3; //width of the canvas divided by 3. here "width" is a keyword.

Using Variables to Create Animation

Animation:
think flip-book!
elements advance frame by frame

In Processing:
default frame rate is 60 fps
draws everything inside draw 60 times per second!

Use this property to create animation!
variables can also be a math operation
if i want to advance the x position of the circle by 1 pixel every time the
program draws, then I can animate the circle!

circleX=50; //first x position
.
.
.
circleX=circleX+1;

   50 +1 //second x position
   51+1   //third x position
   52+1   //fourth x position …. the program will add forever.

Incremental Variables - Speed of Animation

circleX=circleX+1;

      50 +1   //second x position

      51+1   //third x position

      52+1   //fourth x position …. the program will add forever.

      vary speed of the animation:

      1 pix per draw

      10 pix per draw

      50 pix per draw, etc.

```
ellipse(circleX, 180, 24, 24); //draw circle's x position according to the circleX value.
//CREATING very slow ANIMATION
circleX=circleX+0.33; //the x position is going to advance by 1 pixels every three draws.
//circleX=circleX+0.25; //the x position is going to advance by 1 pixels every four draws.
//circleX=circleX+0.1; //the x position is going to advance by 1 pixels every ten draws.
```

circleX=circleX+0.33; //the x position is going to advance by 1 pixels every three draws.

Processing reads it as:

initial value =50

50+0.33=50.33 ==>50

50.33+0.33=50.66 ==>50

50.66+0.33=50.99 ==>50

50.99+0.33=51.32 ==>51

Check out!

sketch_4_1_Incremental_Variables.pde

variable scope

When a variable is shared throughout the program, declare it above and outside **setup()** and **draw()**

This is called a *global variable*

variable scope

When a variable is created within a block of code, such as a function such as **setup()**, it can be used only within that block. In this case **draw()** will not know it.

This is called a *local variable*

system variables for dimensions

```
width // width in pixels of sketch size

height // height in pixels of sketch size

fullScreen // fullScreen
```

width and height are only available after
they have been declared using size();

increment and decrement shorthand

`x = x + 1;` is the same as `x++;` (increment by 1)

`x = x - 1;` is the same as `x--;` (decrement,increment by -1)

`x = x + 5;` is the same as `x+=5;` (increment by 5)

`x = x - 5;` is the same as `x-=5;` (decrement,increment by -5)

let's code - 6 variables!

sketch_4_2_six_variables.pde

```
float x=50; //x position
float d=20;  //diameter
float r=0;  //red channel
float g=0;  //green channel
float b=255;  //blue channel


void setup() {
  size(640, 360);
}

void draw() {
  background(50);
  float y=height/2;  //y position, to use the pre-defined valuable "height", it has to appear after size.

  noStroke();
  fill(r, g, b);
  r=r+10;
  g=g+2;
  b=abs(b-5);


  ellipse(x, y, d, d); //draw circle's x position according to the circleX value.
  x=x+5;
  d=d+1;
}
```

Boolean expressions

FYI- Link to Daniel Shiffman's video:

boolean expression - is only true or false, with no gray area what so ever.

variable in a boolean expression:

x >20

In this case, evaluation of the statement is needed.

For the purpose of evaluation:

relational operators

```
!= (not equal)
< (less than)
<= (less than or equal to)
== (equal)
> (greater than)
>= (greater than or equal to)
```

logical operators

```
! (logical NOT)
&& (logical AND)
|| (logical OR)
```

These operators are C's operators.

conditional statements

if

Only do one thing, when there are two possibilities:

Imagine you have 5 big apples and 5 small apples.

If it's a big apple, then put it in the basket.

if, else

Can do two things, when there are two possibilities:

Same scenario as above.

If it's a big apple, then put it in the basket, **else** <u>compost the apple.</u>

if, else if

More than two possibilities:

If it's a small apple, then compost,
else if it's a big red apple, then put it in the red basket,
else, put it in the black basket.

syntax

```
if (condition) {

    statements;

}
```

```
if (x > 5) {

    background (0);

}
```

If the condition inside the ( ) evaluates to be true, then, execute the code inside the { }. If not, do nothing.

sketch_5_0_Boolean.pde

```
//EXAMPLE 1 -- CHANGE BACKGROUND COLOR ACCORDING TO MOUSE POSITION
void setup() {
  size(640, 360);
}

void draw() {
  background(50);
  if (mouseX>200) {
    background(255, 100, 0);
  }
}
```

If the x position of the mouse is greater than 200, then the background color is orange.

Note: mouseX is in pink. This means "system variable". Must follow syntax.

sketch_5_0_Boolean.pde continued

```
//EXAMPLE 2 -- MOVE THE BALL BACK TO THE LEFT, IF THE BALL GOES OFF THE RIGHT EDGE.
float xPosition=0;

void setup() {
 size(460, 360);
}

void draw() {
 background(50);

 ellipse(xPosition, 100, 20, 20);
 xPosition=xPosition+1;

 if (xPosition==width) {
   xPosition=0;
 }
}
```

This code might run fine, but we are taking a chance.

xPosition is of the variable type of float (decimals).

The width of the canvas is set to an integer number of 360.

There is a chance that the program skips over the exact number of 360.

It's better to use xPosition>=width.

Try xPosition=xPosition+3 to see the program skipping over 360.

syntax for if else

```
if (condition) {

    statements

} else {

    statements

}
```

```
if (x > 5) {

    background (0);

} else {

    background (255);

}
```

If the condition inside the ( ) evaluates to be true, then, execute the code inside the { }. Else, execute the code inside the { } after the word "else".
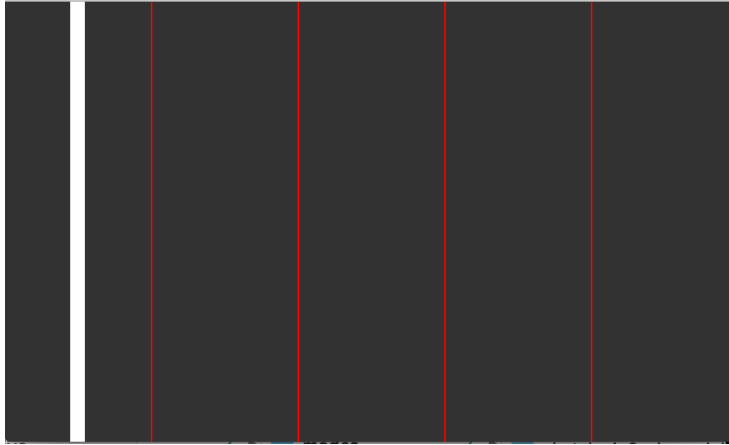
syntax for if else

```
if (condition) {

    statements

} else {

    statements

}
```

```
if (x > 5) {

    background (0);

} else {

    background (255);

}
```

If the condition inside the ( ) evaluates to be true, then, execute the code inside the { }. Else, execute the code inside the { } after the word "else".

syntax for AND OR

for AND, we use **ampersands**.

```
if (( )&&( )){

___

___

___

}
```

for OR, we use **pipes**.

```
if (( )||( )) {

___

___

___

}
```

sketch_5_1_if_else.pde

sketch_5_4_Bouncing_Ball.pde

sketch_5_2_and_or.pde

sketch_5_5_Boolean_true_false.pde

5.1 if, Else if, Else

https://www.youtube.com/watch?v=mVq7Ms01RjA&index=2&list=PLRqwX-V7Uu6YqykuLs00261JCqnL_NNZ_

5.2 AND, OR

https://www.youtube.com/watch?v=9857701OsDE&list=PLRqwX-V7Uu6YqykuLs00261JCqnL_NNZ_&index=3

Yeah!! Now you have all the tools to program!!! All you need is variables and conditions! Other things we learn in the future will help organize and make your code more efficient!

Congratulations!

Now, on to ex 2: Free Patterns!

Reference for ex 2:

How to indicate motion in a paper sketch?

https://www.google.com/search?q=onion+skin&espv=2&biw=1294&bih=722&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjrm6GSmKPKAhUEy2MKHWpvB5EQ_AUIBigB#tbm=isch&q=onion+skin+animation