Midterm

If you are in here, you had better step up your game.
Announcements

- No sections next Monday. Go to sections today and tomorrow if you can.

- **Creativity** in *Processing*
  - Due next Tuesday. Should have started by now.
  - *Get a programming partner! Go Crazy!*

- **Today:** *Functions, For Loops, Some creativity stuff*

- **Next week**
HW9: Creativity in Processing

Assignment. Write four programs to do whatever you want (but don’t copy the examples above), and try to make them clever or interesting or cute or have some property that would interest a viewer. You should try to use those you have learned in your former Processing homework, because those are the basics and one goal of this assignment is to practice the basics. But, if you need some other feature of processing that you find in the reference page, go ahead and use it. The goal is creativity … but don’t spend forever on it either.
```c
void draw() {
    background(0);
    stroke(255, 255, 0);
    line(50, 170, 302, 170);
    line(302, 170, 300, 300)
    line(333, 170, 450, 170)
    line(333, 170, 333, 300)
    stroke(255, 0, 0);
    ellipse(100+x, 100, 20, 20);
    line(100+x, 110, 100+x, 140);
    line(100+x, 125, 80+x, 110);
    line(100+x, 125, 120+x, 110);
    if (walk == 0) {
        stepl = stepl + 2;
    } else {
        stepr = stepr + 2;
    }
    line(100+x, 140, 90+stepl, 170);
    line(100+x, 140, 110+stepr, 170);
    if (abs(stepl-stepr) >= 15) {
        walk = 1-walk;
    }
}
```

// stick man is red
// head
// body
// left arm
// right arm
// which leg is moving
// left, go faster
// right, go faster
// actually move left
// actually move right
// stride over with?
// yes, switch legs
Why Pair Programming?

- Having a partner gives you a built-in helper
- Many people find computers more fun if they get to work with someone else
- Learning to work with computers is like learning a foreign language, lots of new words and codes
  - => Easier to learn a language if you have some to ‘converse with’ in the language
- Working with a partner helps you develop good teamwork skills
- Most programming projects in real life done in teams
- Employers look for people who are good at teamwork
A creativity program from last year

Source code: clickin_charlie
Built with Processing
What can we do with Processing we haven’t done yet?
If we generate it with processing, is it art?

Mondrian, Pollack, Albers are stars …
Art In A Click

- Computer art, that is, art generated by computers, not art created by people using computers, leads to some fun Web sites.

- Google “piet mondrian”. He was a cubist and created pictures that look like this

  http://www.google.com/search?client=safari&rls=en&q="Piet +Mondrian"&ie=UTF-8&oe=UTF-8

- Could we generate this in processing?
How about something a little easier first?

void draw() {
    mid_outer = lerpColor(outer, inner, .33);
    mid_inner = lerpColor(outer, inner, .66);
    fill(outer);
    rect(60, 60, 480, 480);
    fill(mid_outer);
    rect(100, 100, 400, 400);
    fill(mid_inner);
    rect(165, 200, 270, 260);
    fill(inner);
    rect(210, 260, 180, 160);
}

void mousePressed() {
    r = random(0, 255);
    g = random(0, 255);
    b = random(0, 255);
    outer = color(r, g, b);
    r = random(0, 255);
    g = random(0, 255);
    b = random(0, 255);
    inner = color(r, g, b);
}
What would we do in Processing? What does Random do?

```java
void draw() {
    mid_outer = lerpColor(outer, inner, .33);
    mid_inner = lerpColor(outer, inner, .66);
    fill(outer);
    rect(60, 60, 480, 480);
    fill(mid_outer);
    rect(100, 100, 400, 400);
    fill(mid_inner);
    rect(165, 200, 270, 260);
    fill(inner);
    rect(210, 260, 180, 160);
}

void mousePressed() {
    r = random(0, 255);
    g = random(0, 255);
    b = random(0, 255);
    outer = color(r, g, b);
    r = random(0, 255);
    g = random(0, 255);
    b = random(0, 255);
    inner = color(r, g, b);
}
```
Random Numbers

- Random numbers should be called random number sequences, because the definition requires that no matter how many numbers you already know in the sequence, it’s not possible to predict the next one. A non-random sequence is 2, 4, 6, 8, 10, …

- Computers cannot produce random numbers (because computers are completely predictable), but they can produce a sequence of numbers that passes all of the tests for randomness. These are called pseudo-random numbers, but everyone drops the “pseudo” part.

- To generate a random number in Processing we write:
  - random(<smallest possible number>, <largest possible number>).
  - We get back a number – we can’t predict which – between the two limits, including the end points.
  - To generate a random number between 0 and 255, write random(0, 255).
  - To generate a number between 0 and 1, write random(0, 1).
There is a very interesting function called `lerpColor()`. It uses a mathematical idea (that we don’t need to know about) called *linear interpolation* to pick intermediate colors.

Give it two colors, say gold and purple, and it finds a color in between.

Where in between? We also give it a fraction (.33, .66) between 0 and 1 that tells.

Program fills in the black region in the middle with two colors between gold and purple.
Remember how colors work

Pure red is 255

255 – 15 => 240
11111111 – 00001111 = 11110000

128 + 64 = 192
10000000 + 01000000 = 11000000
color colortomod = color(3, 7, 15);

void setup() {
  //size(120,120);
  size(520,520);
  textSize(36);
  textAlign(CENTER);
  frameRate(2);
}

void draw() {
  background(0);
  text(binary(int(red(colortomod)),8), 260, 210);
  text(binary(4*int(red(colortomod)),8), 260, 240);
  text(binary(64*int(red(colortomod)),8), 260, 270);
  text(binary(int(red(colortomod))%4,8), 260, 300);
  text(binary(int(red(colortomod))%3,8), 260, 330);
  //text(key%4, 60,80);
}
What does linear interpolation do to the color binary representations?

Give it two colors, say gold and purple, and it finds a color in between.
void setup() {
    size(400, 400);
}

void draw() {
    fill(255-abs(mouseX-pmouseX));
    rect(pmouseX, pmouseY, mouseX, mouseY);
}
System Variables (Processing)

- mouseX, mouseY
- pmouseX, pmouseY
- width, height
- frameCount
void setup() {
    size(400, 400);
    ________________ // position A
}

_______________ // position B

void draw() {
    ________________ // position C
    fill(255-abs(mouseX-pmouseX));
    rect(pmouseX, pmouseY, mouseX, mouseY);
    ________________ // position D
}

Where should the line “background(255);” be placed so that the sketch shows just a single moving rectangle?
Choose option E if it would work with either C or D.
void setup() {
  size(400, 400);
  //background(255);
}

// background(255);

void draw() {
  background(255, 0, 0);
  //background(255);
  fill(255-abs(mouseX-pmouseX));
  rect(pmouseX, pmouseY, mouseX, mouseY);
  //background(255);
}
pmouseX

// Move the mouse quickly to see the difference
// between the current and previous position
void draw() {
    background(204);
    line(mouseX, 20, pmouseX, 80);
    println(mouseX + " : " + pmouseX);
}

The system variable pmouseX always contains the horizontal position of the mouse in the frame previous to the current frame.
Arty version of Placement. arty_placement

```java
float r,g,b;
color newColor;

void setup() {
    size(400, 400);
    background(0);
    frameRate(20);
}

void draw() {
    r = random(0,255);
    g = random(0,255);
    b = random(0,255);
    newColor = color(r,g,b);
    fill(newColor-abs(mouseX-pmouseX));
    rect(pmouseX, pmouseY, mouseX, mouseY);
}
```
float r, g, b;
color newColor, oldColor;

void setup() {
  size(400, 400);
  background(0);
  frameRate(2);
}

void draw() {
  r = random(0, 255);
  g = random(0, 255);
  b = random(0, 255);
  newColor = color(r, g, b);
  newColor = lerpColor(newColor, oldColor, .33);
  fill(newColor - abs(mouseX - pmouseX));
  rect(pmouseX, pmouseY, mouseX, mouseY);
  oldColor = newColor;
  // background(255);
}
int ballX, ballDia = 50;
void setup() {
    size(400,400);
    ballX = -ballDia/2;
}
void draw() {
    background(120);
    if (ballX > width+ballDia/2)
        ballX = -ballDia/2;
    ellipse(ballX, height/2, ballDia, ballDia);
    ballX = ballX + 1;
}
arty_ball_click5

```cpp
void draw() {
    // background(0);
    if (ballX > width + ballDia/2)
    {
        r = random(0, 255);
        g = random(0, 255);
        b = random(0, 255);
        newColor = color(r, g, b);
        ballX = -ballDia/2;
    }
    // ballX = 0;
    fill(newColor);
    ellipse(ballX, height/2, ballDia, ballDia);
    ballX = ballX + 1;
}
```

A. Ball moves across jumping back to left edge and changing color as soon as it touches the right edge.

B. Ball moves across as one color until it moves off the right edge then changes color and moves back in from the left edge.

C. Ball moves across until half way off the right edge (showing just a half circle) then changes color.

D. Ball moves across then disappears and doesn’t come back.
void draw() {
    //background(0);
    if (ballX > width+ballDia/2)
    {
        r = random(0,255);
        g = random(0,255);
        b = random(0,255);
        newColor = color(r,g,b);
        ballX = -ballDia/2;
    }
    //ballX = 0;
    fill(newColor);
    ellipse(ballX, height/2, ballDia, ballDia);
    ballX = ballX + 1;
}
POINTILISM

- http://www.processing.org/learning/pixels/
Transparency, Tints and More!

http://www.processing.org/learning/pixels/
How about Jackson Pollock?

- Google “jackson pollock”.
- He was an abstract expressionist and created paintings that look like this!

Circa 1950
Computer generated Jackson Pollock?

- I made this on this website.
Can we do that in processing? jackson_pollack

color(255,98,4), color(255,229,7), color(0),};
color[] palette2= { color (138,163,185),

void setup(){

    //load image
    splatter1 = loadImage("pink_splatter.p
    //splatter2 = loadImage("purple_splatt
    splatter2 = loadImage("green_splatter.
    //splatter1 = loadImage("colima_dog.jp
    //splatter2 = loadImage("Colima_Dog_2.

    size(600,400);
    smooth();
    stroke(0);
    background(255,253,240);
}

void draw(){
    //draw when mouse pressed
    if (mousePressed == true){

        if (mouseButton == LEFT){
            image(splatter1,mouseX-10,mouseY-10,(splatter1.width/shrink),(splatter1.height/shrink));
void setup() {
    size(500, 500);
}

int diameter = 0;

void draw() {
    ellipse(width/2, height/2, diameter, diameter);
    diameter = diameter + 5;
}

What does this display?
A) many nested circles (like a target) growing in size
B) a white circle growing in size
C) a black circle growing in size
D) a pulsing images of circles growing and shrinking
E) nothing (a circle with diameter 0)
```java
int diameter = 0;
float r,g,b;
color newColor;

void setup() {
    size(500,500);
    //noFill();
    //fill(255,0,0);
    frameRate(5);
}

void draw() {
    r = random(0,255);
    g = random(0,255);
    b = random(0,255);
    newColor = color(r,g,b);
    fill(newColor);
    ellipse(width/2, height/2, diameter,diameter);
    diameter = diameter + 10;
}
```

What does this display?

A) many nested circles (like a target) changing color and growing in size

B) a white circle growing in size

C) a black circle growing in size

D) pulsing images of circles growing and changing color each time
```cpp
int diameter = 0;
float r, g, b;
color newColor;

void setup() {
    size(500, 500);
    // noFill();
    // fill(255, 0, 0);
    frameRate(5);
}

void draw() {
    r = random(0, 255);
    g = random(0, 255);
    b = random(0, 255);
    newColor = color(r, g, b);
    fill(newColor);
    ellipse(width/2, height/2, diameter, diameter);
    diameter = diameter + 10;
}
```
Redraw.

The redraw() function makes draw() execute once. In this example, draw() is executed once every time the mouse is clicked.

```java
float y;

// The statements in the setup() function
// execute once when the program begins
void setup() {
    size(640, 360);  // Size should be the first statement
    stroke(255);     // Set line drawing color to white
    noLoop();
    y = height * 0.5;
}

// The statements in draw() are executed until the
// program is stopped. Each statement is executed in
// sequence and after the last line is read, the first
// line is executed again.
void draw() {
    background(0);   // Set the background to black
    y = y - 4;
    if (y < 0) { y = height; }
    line(0, y, width, y);
}

void mousePressed() {
    redraw();
}
```
What does this display?

A) many nested circles (like a target) changing color and growing in size

B) a white circle growing in size

C) Circle changes color and grows as well at each mouse click

D) Pulsing images of circles changing color and growing at the frame rate
Don’t forget about Open Processing
How about if we want to repeat the same thing over and over (with variations)?
Two ways

- For Loops
- Functions
Homework 10: Functions

- void cell(int x, int y, int s, color tinto)
- void triple(int x, int y, int s, color tinto)
- Also functions for
  - block()
  - row()
  - cellarray()
For loops (Repetition)

- Repeating commands is powerful:
  - Lightbot 2.0 used recursion, a function calling itself
  - Symbolic Lightbot prefixed a number, 2:Step
- Processing uses a for loop:

```java
void setup() {
  size(500,200);
  background(0);
  noStroke();
  smooth();
  fill(255);
  for (int i=0; i < 16; i++) {
    ellipse(100+25*i, 100, 15, 15);
  }
}
```
Repetition, the Picture

- A **for loop** has several parts, all required ...

```
for (int i=0; i < 16; i++) {
    ellipse(100+25*i, 100, 15, 15);
}
```

The result of this statement is 16 copies of the stuff to be repeated. 16 Pacman pills
For loops: what about with redraw?

```cpp
int mv=6;
int ybase=10;

void setup() {
    size(500,200);
    background(0);
    noStroke();
    noLoop();
    smooth();
    fill(255);
}

void draw() {
    for (int i=0; i < 16; i++) {
        ellipse(100+25*i, ybase, 15, 15);
    }
}

void mousePressed() {
    ybase = ybase+20;
    redraw();
}
```

What does this display?

A) Lots of little pills going off the left and right side of the screen

B) Rows of little pills repeating with every mouse click

C) A row of little pills that change color with every mouse click

D) Pulsing images of circles changing color and growing at the frame rate
For loops. What about with redraw?
Or how about a bullseye?

Note the loop variable must be declared ... but could do it in loop itself like we did for pacman pills:

```java
for (int i = 0; ...)
```
Repetition: Effect of strokeWeight??

```cpp
int i;

size(200,200);
background(0);
fill(0,0,255);
//fill(255,0,0);
//fill(0,255,0);
strokeWeight(4);
for (i = 0; i < 5; i = i + 1) {
    fill(0, 0, 180 + 20*i);
    ellipse(100,100, 100-(20*i),100-(20*i));
}
```
Repetition: Effect of stroke

```
int i;

size(200,200);
background(0);
fill(0,0,255);
//fill(255,0,0);
//fill(0,255,0);
strokeWeight(4);
for (i = 0; i < 5; i = i + 1) {
    fill(0, 0, 180 + 20*i);
    ellipse(100,100, 100-(20*i),100-(20*i));
}
```
What would functions let us do?
Functions.

The `drawTarget()` function makes it easy to specify the position, size, and number of rings.

```java
void setup() {
  size(640, 360);
  background(51);
  noStroke();
  noLoop();
}

void draw() {
  drawTarget(width*0.25, height*0.4, 200, 4);
  drawTarget(width*0.5, height*0.5, 300, 10);
  drawTarget(width*0.75, height*0.3, 120, 6);
}

void drawTarget(float xloc, float yloc, int size, int num) {
  float grayvalues = 255/num;
  float steps = size/num;
  for (int i = 0; i < num; i++) {
    fill(i*grayvalues);
    ellipse(xloc, yloc, size - i*steps, size - i*steps);
  }
}
Instructions Formed of Simpler Instructions

- Check out this screen shot of the Lightbot
- It is partway through an instruction ... its beacon is lit, but not the tile
- To a programmer the instruction is monolithic (one thing)
- To an agent each instruction is a series of steps

An Instruction *abstracts* those steps
Abstracting

- Collecting the operations together and giving them a name is *functional abstraction*
  - The group of operations perform some function but we ignore all of the details
  - Giving it a name is *functional* abstraction
  - This is AMAZINGLY powerful
  - What makes it powerful, is we can forget about the operations and think only about the function they do; more about this later

- Let’s do some functional abstraction
Example: Abstraction in Everyday life

- Get Dressed
  - Dress Bottom Half
    - Put on Sox
  - Dress Top Half
    - Put on Pants
    - Put on Shirt
Functions abstract by packaging Computation

- F1( ) packages actions: E.G. “process a riser”
The Function Becomes A Concept

- Because $F_1()$ “processes a riser,” I think of the programming task as

<table>
<thead>
<tr>
<th>Process a riser</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move to next riser</td>
<td>F1</td>
</tr>
<tr>
<td>Process a riser</td>
<td>F1</td>
</tr>
<tr>
<td>Move to next riser</td>
<td>F1</td>
</tr>
<tr>
<td>Process a riser</td>
<td>F1</td>
</tr>
</tbody>
</table>

- With $F_1()$ as a concept, I simplify the programming to just 5 steps rather than 21
- It also suggests another concept:
  - Move_to_next_riser()
Functions.

The `drawTarget()` function makes it easy to specifies the position, size, and number of rings.

```java
void setup() {
  size(640, 360);
  background(51);
  noStroke();
  noLoop();
}

void draw() {
  drawTarget(width*0.25, height*0.4, 200, 4);
  drawTarget(width*0.5, height*0.5, 300, 10);
  drawTarget(width*0.75, height*0.3, 120, 6);
}

void drawTarget(float xloc, float yloc, int size, int num) {
  float grayvalues = 255/num;
  float steps = size/num;
  for (int i = 0; i < num; i++) {
    fill(i*grayvalues);
    ellipse(xloc, yloc, size - i*steps, size - i*steps);
  }
}
void draw() {
    drawTarget(width*0.25, height*0.4, 200, 4);
    //drawTarget(width*0.5, height*0.5, 300, 10);
    drawTarget(width*0.75, height*0.3, 120, 6);
}

What happens when I comment out the middle line of code?
void draw() {
  drawTarget(width*0.25, height*0.4, 200, 4);
  //drawTarget(width*0.5, height*0.5, 300, 10);
  drawTarget(width*0.75, height*0.3, 120, 6);
}
Functions In Processing

- Form of function definition in Processing

\[
\text{<return type> <name> ( <param list> ) \{} \\
\text{ \hspace{1cm} <body> \} }
\]

as in

```java
void draw_a_box (int x_pos, int y_pos) {
    rect(x_pos, y_pos, 20, 20);
} 

color pink ( ) {
}
```

or

```java
return color(255, 200, 200);
} 
```
- Functions that do something, but do not return a value, have **void** as their `<return type>`

- Functions that return a value must say its type

```java
void draw_a_box (int x_pos, int y_pos) {
    rect(x_pos, y_pos, 20, 20);
}

color pink () {
    return color(255, 200, 200);
}
```
Functions In Processing: Params

- Parameters are the values used as input to the function; parameters are not required, but the parentheses are.
- The type of each parameter must be given.

```java
void draw_a_box (int x_pos, int y_pos) {
    rect(x_pos, y_pos, 20, 20);
}

color pink ( ) {
    return color(255, 200, 200);
}
```
More On Parameters ...

**Parameters**: Customize each function call to a specific situation – they are the input to the function

- *Parameters* are the names of the input values used inside of the procedure body
- *Arguments* are the values from outside to be used for each of the parameters
Arguments Become Parameters

- Notice that if the DEFINITION has $n$ parameters, the CALL needs $n$ arguments
- The parameters and arguments correspond

Inside of the function, the parameter, e.g. xloc, is declared and initialized to the corresponding argument (width*.25). Then, the definition uses it, e.g.

```cpp
void draw() {
    drawTarget(width*0.25, height*0.4, 200, 4);
    //drawTarget(width*0.5, height*0.5, 300, 10);
    drawTarget(width*0.75, height*0.3, 120, 6);
}

void drawTarget(float xloc, float yloc, int size, int num) {
    float grayvalues = 255/num;
    float steps = size/num;
    for (int i = 0; i < num; i++) {
        fill(i*grayvalues);
        ellipse(xloc, yloc, size - i*steps, size - i*steps);
    }
```
Functions In Processing: Return

- A function returns its value with the `return` statement ... the stuff following return is the result
- The function is done when it reaches return

```cpp
void draw_a_box (int x_pos, int y_pos) {
  rect(x_pos, y_pos, 20, 20);
}

color pink ( ) {
  return color(255, 200, 200);
}
```
Recursive Functions. See examples on Open Processing and in processing.org tutorials
Sierpinski: function call when mouse pressed

```java
int size = 600;
int border = 50;
float a = sqrt(size*size - (size/2)*(size/2));
int depth = 3;

void setup()
{
    size(size, (int)h);
    smooth();
    noLoop();
    fill(0);
    noStroke();
    background(255);
}

void draw()
{
    background(255);
    drawTris(0, depth, new PVector(border, h-border), new PVector(width/2, border), new PVector(width-border, h-border));
}

void drawTris(int level, int maxlevels, PVector left, PVector top, PVector right)
{
    level++;
    if(level >= maxLevels) {
        triangle(left.x, left.y, top.x, top.y, right.x, right.y);
        return;
    }
    else {
        PVector a = PVector.add(left, PVector.div(PVector.sub(top, left), 2));
        PVector b = PVector.add(right, PVector.div(PVector.sub(top, right), 2));
        PVector c = PVector.add(left, PVector.div(PVector.sub(right, left), 2));
        drawTris(level, maxLevels, a, top, b);
        drawTris(level, maxLevels, left, a, c);
        drawTris(level, maxLevels, c, b, right);
    }
}

void mousePressed()
{
    switch(mouseButton) {
        case LEFT: depth++; break;
        case RIGHT: depth = max(depth-1, 0); break;
    }
    redraw();
}
```
Processing Basics: Recursive Circles

A demonstration of recursion, which means functions call themselves. Notice how the `drawCircle()` function calls itself at the end of its block. It continues to do this until the variable "level" is equal to 1.

```java
void setup() {
  size(640, 360);
  noStroke();
  noLoop();
}
void draw() {
  drawCircle(width/2, 280, 6);
}
void drawCircle(int x, int radius, int level) {
  float tt = 126 * level/4.0;
  fill(tt);
  ellipse(x, height/2, radius*2, radius*2);
  if (level > 1) {
    level = level - 1;
    drawCircle(x - radius/2, radius/2, level);
    drawCircle(x + radius/2, radius/2, level);
  }
}
```
Functional Abstraction Reduces Complexity

Layering: Building Functions out of Functions. Next week.
Homework 10: Functions

- void cell(int x, int y, int s, color tinto)
- void triple(int x, int y, int s, color tinto)
- Also functions for
  - block()
  - row()
  - cellarray()