Announcements

- **Creativity in Processing**
  - HW9: Due in one week

- **Get a programming partner! Go Crazy!**
  - [https://courses.soe.ucsc.edu/courses/cmps10/Winter12/01/pages/syllabus](https://courses.soe.ucsc.edu/courses/cmps10/Winter12/01/pages/syllabus)

- Next week:
  - Steganography
  - Lecture on Processing concepts
  - Plus Review for Midterm (Feb 13th!!)

- We are working on a ‘practice midterm’ for you
- I like your privacy essays
Homework 7 Review: Here’s Chao!
Homework 7
Making Blinky Work

Zhichao Hu
zhu@soe.ucsc.edu
Data types

• Primitive data types
  ◦ int: numbers without a decimal point.
  ◦ boolean: data type for the boolean values true or false.

• Composite data types
  ◦ color: storing color values
  ◦ Function color(R, G, B); creates a color that can be stored in variable of the color data type

Sample Code:

```
color colorBlinky = color(255, 0, 0);
```
HW 7 answer part 1:

```cpp
int x=0;
int look=0; // 0 for left, 1 for middle, 2 for right
int dir=1; // 1 for moving right, -1 for right, 0 for middle
int colorFlag=0; // 0 for red, 1 for yellow
color colorBlinky=color(255, 0, 0);

void setup() {
  size(600, 200);
  background(0, 0, 0);
  noStroke();
}
```

A flag is a way to tell you whether you should go this way or that way in your program. (Combines with if-else)
if - else

- Expression (test) - true or false (boolean).
- Statements – executable: call a function, assign a value, etc.
- **if**: expression inside the parentheses is true, execute the statements enclosed within the curly brackets.
- **else**: extends the if() structure, when the expression in if() is false, execute the statements enclosed within the curly brackets.

Sample syntax:
```java
if (expression) {
    statements
}
else {
    statements
}
```

Returns a boolean value (true/false)
Relational Operators

- For true/false expressions (or boolean values)

- == (equality)
- > (greater than)
- >= (greater than or equal to)
- != (inequality)
- < (less than)
- <= (less than or equal to)

Sample Code:
```java
int n=0;
if(n>=0){
    n=n+1;
}
else{
    n=n-1;
}
```
Logical Operators

- Connect two or more true/false expressions (or boolean values)

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

Sample code using “&&”:
```c
int n=0;
if((n>=0) && (n<=6)) {
    n=n+1;
}
```

Sample code using “||”:
```c
if((n==0) || (n<6)) {
    n=n+1;
}
```

Sample code using “||”:
```c
if(! (n<6)) {
    n=n+1;
}
```
mouseX, mousePressed()

• `mouseX`: this system variable always contains the current horizontal coordinate of the mouse

• `mousePressed()`: this function is called automatically by the system once after every time a mouse button is pressed.

Sample syntax:

```java
void mousePressed() {
    statements
}
```
void mousePressed() {
    if (colorFlag==0) {
        colorBlinky=color(255, 255, 0);
        colorFlag=1; // yellow
    }
    else {
        colorBlinky=color(255, 0, 0);
        colorFlag=0; // red
    }
}
void draw() {
    background(0);
    // Blinky's body
    fill(colorBlinky);
    rect(50+x, 70, 10, 90);//1
    rect(60+x, 40, 10, 130);//2
    // Blinky's white of the eyes
    fill(255);
    rect(60+x+look*10, 50, 10, 30);
    rect(70+x+look*10, 40, 10, 50);
    // Blinky's pupils
    fill(0, 0, 255);
    rect(60+x+look*20, 60, 20, 20);
    rect(120+x+look*20, 60, 20, 20);
    // Change Blinky's moving and looking directions
    if ((mouseX>=110+x)&(mouseX<=130+x)) {
        dir=0;
        look=1;
    }
    if (mouseX>130+x) {
        dir=1;
        look=2;
    }
    if (mouseX<110+x) {
        dir=-1;
        look=0;
    }
    x=x+dir;
}
This can work too (notice the "else if"):

```cpp
if (mouseX>130+x) {
    dir=1;
    look=2;
}
else if (mouseX<110+x) {
    dir=-1;
    look=0;
}
else{
    dir=0;
    look=1;
}
```

This will **not** work, Blinky will not move right:

```cpp
if (mouseX>130+x) {
    dir=1;
    look=2;
}
else if (mouseX<110+x) {
    dir=-1;
    look=0;
}
else{
    dir=0;
    look=1;
}
```

Can you figure out why?
QUESTIONS??
Creative Assignment HW9
PAIR PROGRAMMING. COME UP AFTER CLASS TO GET A PAIR. WE STOP EARLY.
Huffman Coding, Logic Gates, Truth Tables

DO YOU WANT ME TO DO THEM?
OR WAS IT TOO EASY?
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.
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
- \(\Rightarrow\) 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
What is the PPP? Pair Programming Protocol!!

- In THE SAME PLACE AND TIME
- There's a DRIVER and a NAVIGATOR
- The driver places the keyboard and mouse in a good ergonomically correct position
- Partners position the monitor where they can both easily see it
- The Navigator has the instructional and design materials such as a notebook, assignment, Reas and Fry book
- Partners identify GOAL of the session.
  - What they were doing at end of last session
  - What they need to focus on during this work session
Continued: What is the PPP? RULES

- The Driver operates the keyboard and the mouse and the navigator does not GRAB or TOUCH them.
- The Navigator describes what the driver needs to do in reference to the goals, shows the instructions or whatever to the driver, but does not give the driver any of the materials.
- Partners negotiate and make mutual decisions.
- Partners are good-natured about mistakes.
- Partners try to encourage each other.
- When it is time to SWITCH ROLES, do it without drama.
- As switching, Be respectful about the role the other just played.
- To switch: either get up and switch places or move the keyboard and mouse.
What do you do if you get a lousy partner?

- Navigator who grabs the keyboard or mouse off of you
- Navigator not engaged, spaces out, talks to someone else, walks away to talk to someone else
- Partner who argues about everything, will not negotiate and compromise
- Partner who makes unilateral decisions or does stuff when the other one isn’t there (maybe cause the other person didn’t show up?)
- Insults/put downs! Even joking can undermine collaboration
- Refusing to switch roles
- Both partners off task, talk about nonproject stuff
Making an example of Pacman

Getting Into Processing
Being Successful with Computers

- Two good habits:
  - #1 When things go wrong, try to figure out what's wrong *yourself* ...
    - You will learn more if you invest more
    - But then ask or go to section when all else fails
  - #2 Persistence ... stay with it until you get it;
    - Most things that are worth mastering take effort ...
      - Violin, piano, tennis, soccer, singing, writing
    - When you get it, it's really satisfying!
    - Getting the computer to draw something you want, that is all your own, MASTERY!
Drawing arcs … a less intuitive operation as an example of working out the details.

### Drawing Curves

This tutorial is for Processing version 1.1+. If you see any errors or have comments, please let us know.

Download the files from this tutorial.

This tutorial introduces you to the three types of curves in Processing: arcs, spline curves, and Bézier curves.

#### Arcs

Arcs are the simplest curves to draw. Processing defines an arc as a section of an ellipse. You call the function with these parameters:

\[
\text{arc}(x, y, \text{width}, \text{height}, \text{start}, \text{stop})
\]

The first four parameters are the same as the ones for \texttt{ellipse}(); they define the boundary box for your arc. The last two parameters are the starting and ending angle for the arc. These angles, as with all other angles in Processing, are given in radians. Remember that angles are measured clockwise, with zero degrees pointing east. Using the fact that \(\pi\) radians equals 180°, here are some example arcs.

```java
void setup()
{
  size(200, 200);
  background(255);
  smooth();

  rectMode(CENTER); // show bounding boxes

  stroke(128);
  rect(35, 35, 50, 50);
  rect(105, 35, 50, 50);
  rect(175, 35, 50, 50);
  rect(105, 105, 100, 50);

  stroke(0);
  arc(35, 35, 50, 50, 0, PI / 2.0); // lower quarter circle
  arc(105, 35, 50, 50, -PI, 0); // upper half of circle
  arc(175, 35, 50, 50, PI / 6, PI / 6); // 60 degrees
  arc(105, 105, 100, 50, PI / 2, 3 * PI / 2); // 180 degrees
}
```
Getting Started with Processing: 20-21

- Drawing arcs ... a less intuitive operation as an example of working out the details
- Can we draw a figure that looks like Pacman?
- What does the tutorial say?

```java
arc(50, 55, 50, 50, 0, PI/2);
noFill();
arc(50, 55, 60, 60, PI/2, PI);
arc(50, 55, 70, 70, PI, TWO_PI-PI/2);
arc(50, 55, 80, 80, TWO_PI-PI/2, TWO_PI);
```
What The Parameters Mean ...

```
arc(50, 55, 50, 50, 0, PI/2);
noFill();
arc(50, 55, 60, 60, PI/2, PI);
arc(50, 55, 70, 70, PI, TWO_PI-PI/2);
arc(50, 55, 80, 80, TWO_PI-PI/2, TWO_PI);
```

<table>
<thead>
<tr>
<th>Syntax</th>
<th>arc(x, y, width, height, start, stop)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>x int or float: x-coordinate of the arc's ellipse</td>
</tr>
<tr>
<td></td>
<td>y int or float: y-coordinate of the arc's ellipse</td>
</tr>
<tr>
<td></td>
<td>width int or float: width of the arc's ellipse</td>
</tr>
<tr>
<td></td>
<td>height int or float: height of the arc's ellipse</td>
</tr>
<tr>
<td></td>
<td>start int or float: angle to start the arc, specified in radians</td>
</tr>
<tr>
<td></td>
<td>stop int or float: angle to stop the arc, specified in radians</td>
</tr>
</tbody>
</table>
Two Related Terms ...

Terms: **Parameters** are the names for the positions; **arguments** are their values.
What do we need to make Pacman?

arc(100, 100, 80, 80, radians(45), radians(315));

smooth();
The Basic Part is Defined …

- Now it’s time to make it move!
  - For that, define the program so it is active

```cpp
int mv = 60;

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);
    }
}

void draw() {
    fill(255, 255, 0);
    arc(mv, 100, 80, 80, radians(45), radians(315));
    mv = mv + 1;
}
```

Add one to mv after each time the image is drawn

Make the pills!
The Basic Part is Defined …

- Now it’s time to make it move!
  - For that, define the program so it is active

```cpp
int mv = 60;

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);
  }
}

void draw() {  
  fill(255, 255, 0);
  arc(mv, 100, 80, 80, radians(45), radians(315));
  mv = mv + 1;
}
```

Add one to `mv` after each time the image is drawn.
We’ve seen this before!

- We repeatedly redraw Pacman one position to the right, but he needs to be erased first.
  - A common trick – we’ve used it before – is to redraw the (black) background which rids us of Pacman, but it also rids us of the “pills”
  - We could redraw both, but then we need to figure out how many he’s eaten so we draw only those ahead of him
  - Or, we could leave the pills there, and just erase him
Following Tactic #3

- We first draw Pacman with black fill to obliterate him...

```java
void draw() {
    fill(0);
    arc(mv, 100, 80, 80, radians(45), radians(315));
    fill(255, 255, 0);
    arc(mv, 100, 80, 80, radians(45), radians(315));
    mv = mv + 1;
}
```

So, what happened?
Moving \( mv = mv + 1 \) Helps ... Some

- The improved code rids us of nearly everything

```cpp
void draw() {
    fill(0);
    arc(mv, 100, 80, 80, radians(45), radians(315));
    mv = mv + 1;
    fill(255, 255, 0);
    arc(mv, 100, 80, 80, radians(45), radians(315));
}
```
Now What To Do?

• Ideas?
The Code To This Point … He Works!

```c
int mv = 60;

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

void draw() {
    fill(0);
    arc(mv, 100, 80, 80, radians(50-2*(1+mv%25)), radians(310+2*(1+mv%25)));
    mv++;
    fill(255,255,0);
    arc(mv, 100, 80, 80, radians(50-2*(1+mv%25)), radians(310+2*(1+mv%25)));
}```
Planning the “Chomp”

- Clearly, to make Pacman chomp on the pills, we need to draw arcs that get closer and closer to being circles ... so in the

  \[ \text{arc(mv,100,80,80,\text{radians}(45),\text{radians}(315))} \]

  function, 45 \to 0 and 315 \to 360

- How much should we change them by?
  - 45 \to 0 by at several rates
    - by 15 in 3 steps
    - by 9 in 5 steps
    - by 5 in 9 steps
    - by 3 in 15 steps
Consider 9 Chomps of 5 Each

- We need to abstract
  - \(45 - 5 \times 0 = 45\)
  - \(45 - 5 \times 1 = 40\)
  - \(45 - 5 \times 2 = 35\)
  - \(45 - 5 \times 3 = 30\)
  - ...
  - \(45 - 5 \times 9 = 0\)

- So, \(\text{radians}(45-5 \times <\text{number in 0 to 9}>)\)
Mod – A Very Clever Operator

- Mod – short for modulo – is a programming operator like divide and is written with %
  - Mod gives the remainder from a division ...
  - 4%4 is 0, because 4 divides into 4 evenly, i.e. 0 left
  - 5%4 is 1, because 4 goes into 5 once with 1 leftover
  - 6%4 is 2, because 4 goes into 6 once with 2 leftover
  - 7%4 is 3, because 4 goes into 7 once with 3 leftover
  - 8%4 is 0 again, because 4 divides 8 evenly
  - 9%4 is 1 again, because 4 goes into 9 twice w/ 1 left
  - ...

mod is a very useful operator
Using Mod with \texttt{mv}

- So, to get the numbers 0 through 9 for our equation, we need to write \texttt{mv \% <what?>}

- We need to abstract
  - $45 - 5 \times 0 = 45$
  - $45 - 5 \times 1 = 40$
  - $45 - 5 \times 2 = 35$
  - $45 - 5 \times 3 = 30$
  - ...
  - $45 - 5 \times 9 = 0$
Using Mod with \( \text{mv} \)

- So, to get the numbers 0 through 9 for our equation, we need to write \( \text{mv} \% <\text{what}?> \)
- If you said “9” you were WRONG!
- \( x \% n \) gives numbers 0 through \( n-1 \)
- If you want numbers 0 through 9, you want 10 numbers or \( \text{mv} \% 10 \)

We need to abstract

- \( 45 - 5*0 \) = 45
- \( 45 - 5*1 \) = 40
- \( 45 - 5*2 \) = 35
- \( 45 - 5*3 \) = 30
- ... 
- \( 45 - 5*9 \) = 0

If you missed it, not to worry; EVERY programmer has, too
Ready To Chomp

- The code ...

```java
int mv = 60;

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

void draw( ) {
    fill(0);
    arc(mv, 100, 80, 80, radians(45-5*(mv%10)), radians(315+5*(mv%10)));
    mv++;
    fill(255,255,0);
    arc(mv, 100, 80, 80, radians(45-5*(mv%10)), radians(315+5*(mv%10)));
}
```

Try it after class
Try Revising it

- Perhaps slower is better ... need more frames, that is, repetitions of draw( ) to close mouth
- A rate of 3 in 15 steps would be somewhat better ... meaning that the amount subtracted is 3*mv%

... experimenting, I thought 25 frames was about right

Just Try It!
int mv = 60;

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

void draw() {
    fill(0);
    arc(mv, 100, 80, 80, radians(50-2*(mv%25)), radians(310+2*(mv%25)));
    mv++;
    fill(255, 255, 0);
    arc(mv, 100, 80, 80, radians(50-2*(mv%25)), radians(310+2*(mv%25)));
int mv = 60;

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

void draw() {
    fill(0);
    arc(mv, 100, 80, 80, radians(50-2*(1+mv%25)), radians(310+2*(1+mv%25)));
    mv++;
    fill(255, 255, 0);
    arc(mv, 100, 80, 80, radians(50-2*(1+mv%25)), radians(310+2*(1+mv%25)));
}
ANYONE WHO NEEDS A PAIR PROGRAMMING PARTNER COME UP NOW.