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

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!

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?

Two Related Terms ...

Terms: Parameters are the names for the positions; arguments are their values

Making Things Work For Ourselves

- 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

```
int mv = 0;

void setup() {
  size(500, 300);
  background(0);
  stroke(0);
  fill(125);
  for (int i = 0; i < 10; i++) {
    ellipse(200 + 25*i, 100, 15, 15);
  }
}

void draw() {
  fill(125, 255, 0);
  arc(xv, 180, 0, 0, radians(45), radians(360));
  mv = mv + 1;
}
```

Make the pills!

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...

```
void draw() {
  fill(0);
  arc(xv, 180, 0, 0, radians(45), radians(360));
  fill(125, 255, 0);
  arc(xv, 180, 0, 0, radians(45), radians(360));
  mv = mv + 1;
}
```

Add one to mv after each time the image is drawn

So, what happened?

Moving \( mv = mv + 1 \) Helps ... Some

- The improved code rids us of nearly everything

```
void draw() {
  fill(0);
  arc(xv, 180, 0, 0, radians(45), radians(360));
  mv = mv + 1;
  fill(255, 255, 0);
  arc(xv, 180, 0, 0, radians(45), radians(360));
}
```

Now What To Do?

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

```cpp
int mv = 45;
void main() {
    size(500, 300);
    background(0);
    strides(8);
    smooth(3);
    fill(255);
    for (int i = 90; i < 180; i++) {
        ellipse(100+i*5, 100, 15, 15);
    }
    void draw() {
        fill(0);
        arc(100, 80, 80, radians(45), radians(315));
        arc(255, 255, 80, radians(50-2*i*1025));
        arc(310, 240, radians(50-2*i*1025));
    }
}
```

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 `arc(mv,100,80,radians(45),radians(315))` function, $45 \rightarrow 0$ and $315 \rightarrow 360$

- How much should we change them by?
  - $45 \rightarrow 0$ 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
  - $45 - 5*0 = 45$
  - $45 - 5*1 = 40$
  - $45 - 5*2 = 35$
  - $45 - 5*3 = 30$
  - ...
  - $45 - 5*9 = 0$

- So, `radians(45-5*<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 twice with 2 leftover
  - $7\%4$ is 3, because 4 goes into 7 twice 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 mv

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

- We need to abstract
  - $45 - 5*0 = 45$
  - $45 - 5*1 = 40$
  - $45 - 5*2 = 35$
  - $45 - 5*3 = 30$
  - ...
  - $45 - 5*9 = 0$

Using Mod with mv

- 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...
  ```c
  int mv = 60;
  void setup() {
    size(180, 180);
    background(0);
    stroke(0);
    smooth();
    fill(255);
    for (int i=0; i < 10; ++i) {
      ellipse(100+2*i, 100, 15, 15);
    }
  }
  void draw() {
    ellipse(180, 180, 100, 15, 15);
    for (int i=0; i < 10; ++i) {
      ellipse(100+2*i, 100, 15, 15);
    }
  }
  ```

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

The Code V I

```c
int mv = 60;
void setup() {
  size(180, 180);
  background(0);
  stroke(0);
  smooth();
  fill(255);
  for (int i=0; i < 10; ++i) {
    ellipse(100+2*i, 100, 15, 15);
  }
}
void draw() {
  ellipse(180, 180, 100, 15, 15);
  for (int i=0; i < 10; ++i) {
    ellipse(100+2*i, 180, 15, 15);
  }
}
```