CSE120: Computer Science: Principles

Lab Exercise: Pair Programming

Goals

The point of this exercise is simply to get some hands-on experience with Processing. You should note that it is a convenient and reasonably forgiving environment, and at the same time it is a legitimate symbolic language (Java).

Find the Processing application and open it.

Exercise 1: Snow Angel

This is the first program students see. The text for it is here:

```
// Draw a snow angel on blue snow

void setup() {
  size(500,500);         //define canvas size
  background(0,0,255);   //define canvas color
  stroke(255,255,255);   //define line color
}

void draw() {
  //Draw line from (150,150) to the mouse position
  line(150,150, mouseX, mouseY); // Note caps! Essential
}
```

You can copy / paste this text into a processing window, but I suggest for this one that you type it in just to experience what the students see the first moment they touch Processing.

Suggested practice:

Easy: Flip color so the white line (stroke) on blue background becomes a blue line (rgb (0,0,255)) on a white (rgb (255,255,255)) background.

Medium: Get rid of the annoying line from the corner which is created because at program initialization the mouse is set to be a (0,0). This is a 2 step operation:

a) Set the stroke's initial value in `setup( )` to be the same as the background, disguising it.

b) Then as the last instruction the `draw( )` function, add this if-statement.

```
if (mousePressed){ // if the mouse is ever clicked,
     stroke(255);  // reset line color
}
```

Note cap in `mousePressed`. Be sure the reset line color is your contrasting color, and don't forget the closing brace at the end!

Exercise 2: Maggie's pacifier.

The Simpson's code is given below; copy/paste into new window. Add a pacifier for the baby Maggie. This is a circle – created by an `ellipse( )` call in Processing:

```
ellipse( x-posish, y-posish, 10, 10);
```

Want some documentation? Nav Help > Reference > 2D Primitives > Ellipse. Of course, the pacifier should be filled with the red color (rgb (255,0,0)).

```
void setup ( ) {
  size(900,800);
  background(255, 245, 220);
  noStroke();
  frameRate(5);
}

void draw ( ) {
  background(255, 245, 220);

// Homer
  fill(250,193,35);      //yellow
  rect(300,300,40,40);
  fill(245,245,245);
  rect(300,340,40, 50);  //white
  fill(89,79,217);
  rect(300,390,40,45);   //blue
// Marge
  fill(167,203,60);      //olive
  rect(360,340,40,95);
  fill(250,193,35);      //yellow
  rect(360,305,40,35);
  fill(41,82,240);       //light blue
  rect(360,210,40,95);
// Bart
  fill(89,79,217);       //blue
  rect(420,405,40,30);
  fill(240,71,41);       //red
  rect(420,375,40,30);
  fill(250,193,35);      //yellow
  rect(420,345,40,30);
// Lisa
  fill(229,77,35);       //red orange
  rect(480,380,40,55);
  fill(250, 193, 35);    //yellow
  rect(480, 350, 40, 30);
// Maggie
  fill(147,189,255);     //blue
  rect(540, 395,40, 40);
  fill(250, 193, 35);    //Yellow
  rect(540, 370, 40, 25);
}
```

The task is to get the positioning right. Students are prompted to estimate-try-test.

Exercise 3: Mike's Eye

The code for Mike from Monster's Inc is given below; copy and paste this into a new Processing window. Run the program. Notice his eye motion as the mouse sweeps across the canvas. Also, try clicking three times.

Notice that there are two if-statements at the start of the `draw( )` function, which control his eye. We would like the eye to look straight forward, but the logic of these two if-statements isn't enough. Add a third if-statement testing whether the `mouseX` value is both more that 180 AND less than 220, and if so, reset the variable `idx` to 0. (We write AND in Processing as `&&`) If you do this right, his eye should look straight ahead when the mouse is along his centerline.

```
float idx = 0;
int times = 0;
int amt = 0;
int step = 7;
int lim = -50;

void setup( ) {
  size(500,500);
  background(230);
  noStroke( );
}

void draw( ) {
  background(230);
  if (mouseX > 220 ) {
    idx = min(10, idx+0.5);
  }
  if (mouseX < 180 ) {
    idx = max(-10, idx-0.5);
  }
// **** need code here ****
  fill(95,165,35);
  ellipse(200,200+amt, 100, 120);
  ellipse(200,220+amt, 110, 110);
  rect(180,min(270+amt,330),10,80);
  rect(218,min(270+amt,330),10,80);
  fill(230);
  ellipse(200,180+amt, 60, 50);
  rect(200,230+amt, 10, 10);
  rect(186,230+amt, 10, 10);
  fill(35,99,165);
  ellipse(200+idx,180+amt, 27, 27);
  fill(0);
  ellipse(200+idx, 180+amt,16,16);
  strokeWeight(4);
  stroke(230);
  noFill();
  arc(200, 200+amt, 100, 60, HALF_PI-QUARTER_PI, HALF_PI+QUARTER_PI);
  noStroke( );
  if (amt > 100) {
```

```
    step = -5;
  }
  if (amt < lim) {
    step = 5;
  }
  amt = amt + step;
  if (times > 2) {
    lim = -400;
  }
}

void mousePressed( ) {
  times = times + 1;
}
```

Challenge: Note the code at the end of `draw( )` … make Mike's jump limit as high as -800 if the user presses the mouse four `times`. And, of course, there are a lot of other things one can play around with.