Partner Work 1 Reference Sheet: Processing

By Duke Hutchings for CSC 130

Variables

`mouseX` current *x*-coordinate of the mouse cursor (position from left-to-right)

`mouseY` current *y*-coordinate of the mouse cursor (position from top-to-bottom)

`width` the number of pixels from left to right in the drawing space

`height` the number of pixels from top to bottom in the drawing space

`PI` The mathematical constant pi (π), about 3.1415927.
Also equivalent to 180 degrees or half of a circle.

`TWO_PI` 2π, or 360 degrees

Functions – drawing stuff

`line(x1,y1,x2,y2)` draws a line using the two points (x1,y1), (x2,y2)

`triangle(x1,y1,x2,y2,x3,y3)` draws a triangle using the three points (x1,y1), (x2,y2), (x3,y3)

`ellipse(x,y,w,h)` draws an ellipse anchored at point (*x*,*y*) with width *w* and height *h*

`ellipseMode(CORNER)` when drawing ellipse, (*x*,*y*) refers to its upper-left corner

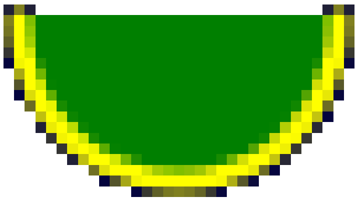
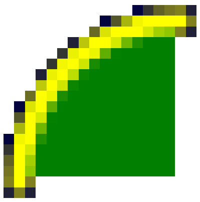
`ellipseMode(CENTER)` when drawing ellipse, (*x*,*y*) refers to its center point

`rect(x,y,w,h)` draws rectangle anchored at point (*x*,*y*) with width *w* and height *h*

`rectMode(CORNER)` when drawing rectangle, (*x*,*y*) refers to its upper-left corner

`rectMode(CENTER)` when drawing rectangle, (*x*,*y*) refers to its center point

`arc(x,y,w,h,b,e)` draws a portion of an ellipse centered at (*x*,*y*), width *w*, height *h*
beginning at radian value *b* and ending at radian value *e*

	
arc(600, 180, 30, 30, **0, PI**)	arc(599, 259, 30, 30, **PI, 3*PI/2**)

Functions – affecting the drawing of stuff

`background(r,g,b)`	sets the background color based on the combination of *r*, *g*, and *b*
`fill(r,g,b)`	when drawing shapes, the interior color will be based on *r*, *g*, and *b*
`fill(r,g,b,a)`	use the extra argument *a* to make the shape semi-transparent
`noFill()`	when drawing shapes, prevents drawing an interior color
`stroke(r,g,b)`	when drawing shapes, the border color will be based on *r*, *g*, and *b* this color will also be used for drawing single lines
`strokeWeight(w)`	when drawing shapes or lines, the thickness of the line will be *w* pixels
`noStroke()`	when drawing shapes, no border will be drawn

Functions – typically called in setup()

`size(w,h)`	sets the size of the drawing space to be *w* pixels wide and *h* pixels high
`smooth()`	draws smoother curves and diagonal lines

Some Math Examples

Sometimes you will want or need to use math to help draw something in a particular way in Processing, especially when using variables like *mouseX*, *mouseY*, *width*, and *height*. You can use the usual symbols of + and − for addition and subtraction. For multiplication and division, use `*` and `/`.

For example, the following two sets of code accomplish the exact same thing, pixel-for-pixel.

```
rectMode(CENTER);
rect(mouseX, mouseY, 32, 18);
```

```
rectMode(CORNER);
rect(mouseX-16, mouseY-9, 32, 18);
```

This next set of code will draw a box around the mouse that...
- is at least 20 pixels wide and 20 pixels tall
- slowly gets wider as the mouse moves to the right-hand area of the drawing space
- slowly gets taller as the mouse moves to the bottom area of the drawing space

Notice the use of parentheses in a manner similar to the normal usage in mathematics.

```
rectMode(CENTER);
rect(mouseX, mouseY, (mouseX/5)+20, (mouseY/5)+20);
```