Lab Week 5. Functions

Last Week

- Events
- Animation
- IF statement – conditional branching
- User interaction
- Pong Game

Learning Objectives

- Using functions
- Implementing Functions
- Text command

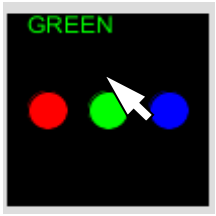
[Please open last week's portfolio exercise Pong Squash for marking (and any others)]

Resources

- Lecture Notes
- Processing website - reference
- http://www.cs.sfu.ca/CourseCentral/166/tjd/first_program.html
- http://codingbat.com/java/Logic-1

In a new Week 5 directory, save your code after each exercise

Ex1. Colour reporting function



We will consider a program that reports on the colour of the pixel at the current mouse location.

Download from moodle and try it out

- We've introduced some constants (int) to represent the colours (we'll look at **enum** in later weeks)
- Writen a **testColour function** that is given a **color**(R,G,B) and returns an **integer constant** pertaining to that colour.

```
int testColour(color c)
```

- Drawn some ellipses and written the correct piece of text to the screen
- Used the built in **get** function to provide the colour of the pixel as a **color(R,G,B)**

Code (on moodle)

```
//colour detecter
final int RED=0;  //constants :state why these are useful
final int GREEN=1;
final int BLUE=2;
final int OTHER=3;

int testColour(color c) {
//highlight the return type (one of the constants), the parameter
  if (c==color(255,0,0))      //testing the colour
     return RED;              //returning the correct constant value
  else if (c==color(0,255,0))
    return GREEN;
  else if (c==color(0,0,255))
    return BLUE;
  else
    return OTHER;
}

void setup(){
}

void draw(){
  color pixel;  //variable of type color (R,G,B)

  background(0);
  fill(255,0,0);              //drawing 3 circles red,green & blue
  ellipse(20,50,20,20);
  fill(0,255,0);
  ellipse(50,50,20,20);
  fill(0,0,255);
  ellipse(80,50,20,20);
       //calling get function: takes x,y parameters, returns color
  pixel = get(mouseX,mouseY);
  if (testColour(pixel)==RED){  //calling our function testColour
    fill(255,0,0);
    text("RED",10,10);    //writing text to the screen at 10,10
  }
  else if (testColour(pixel)==GREEN){
    fill(0,255,0);
    text("GREEN",10,10);
  }
   else if (testColour(pixel)==BLUE){
    fill(0,0,255);
    text("BLUE",10,10);
   }
   else {
     fill(255);
     text("OTHER",10,10);
   }
}
```

Add a yellow circle and update the code to report YELLOW when the mouse moves over the yellow circle.

Ex2. Using CodingBat website http://codingbat.com/java/Logic-1

Attempt to write the functions, from cigarparty up to and including teensum

Ex3. **TronLiteCycles**

This game involves a moving point with a constant speed that leaves a trail (of light) wherever it has been. At any time the point will be moving in one of the following directions UP,DOWN,LEFT or RIGHT (i.e. they can't move diagonally).

We can start to think about this game using the **EtchySketch** drawing program below

```
//EtchySketch
//arrow keys to draw
float x,y;

void setup(){
 size(500,500);
 background(255,255,255);  //background white
 x=200;
 y=200;
 stroke(0,0,0);  //pen black
}

void draw(){
  point(x,y);  //draw a point at current (x,y)
}

void keyPressed(){
 if(key == CODED)
  {
    if (keyCode == UP && y>=0){ //restrict to screen edge
      y=y-1;
    }
    else if(keyCode == DOWN && y<=500){
          y=y+1;
    }
    else if (keyCode == LEFT && x>=0){
      x=x-1;
    }
    else if (keyCode == RIGHT && x<=500){
      x=x+1;
    }
  }
}
```

1. Copy and paste (or download from moodle) the code above into a sketch and save as “etchySketch”. Ensure you understand how it works, play with it and read through the code events. Arrow keys to draw.

2. Alter the code so that the drawn point is continually moving in its current direction **deltaX** or **deltaY**. We used a similar idea with the bouncing ball. Here we need to ensure that if **deltaX** is +1 or -1, then **deltaY** is 0 and vice versa. Remember to use Procedures for any blocks of code which perform a single task.

```
 stroke(255,0,0); //pen colour red
 point(x,y);
//move cycle, calculate next x and next y
```

3. Add and make use of a crash detection function using **get** to check the pixel colour, a crash hass occurred if the pixel is not the background colour;

```
boolean crash(float x, float y)
{
 color col=get((int)x,(int)y);  //get colour of the next position
 if (col==color(0,0,0)) //check if colour is background colour, e.g. black

//complete the code
}
```

We need to call crash from our draw event BEFORE we draw the point – why?

What will happen if your point moves off the screen. Draw a box around the perimeter so that a crash will occur.

extension exercise Ex4. **Complete TronLiteCycles Game (- harder)**

Generally the game consists of two players, the objective being to cause your opponent to crash into your tail (or perhaps a wall – draw a box around screen edge).

Pressing a key should change the direction of the point by 90 degrees (a 180 degree turn would be a crash).

Collision detection can be accomplished by checking the colour of the new position of the point before it has been drawn. This will be the background colour if no crash has occurred.

```
color col = get( (int) x, (int) y); //get colour of pixel at (x,y)

if (col != color(0,0,0)) //if pixel colour is not black (background) then ...
```

(int) is called casting, in this case it creates a whole number – chopping off anything after the decimal point.

Variables required for one player are

```
float player1_x, player1_y, player1_deltaX, player1_deltaY;
```

where if player1_deltaX is not 0, then player1_deltaY must be **0** and vice versa (ensuring the correct movement).

Complete the game for two players, using functions (hint: a crash detection function, a restart function and other blocks of code will make your program easier to read and understand).

If a crash has occurred draw could (possibly) provide an explosion then restart the game – clear the screen and relocate players. Have fun.