CS 106 Winter 2018

Lab 05: User Interfaces

Due: Wednesday, February 6th, 11:59pm

This lab will allow you to practice User Interfaces using Direct Manipulation and ControlP5. Each question is on a separate page.

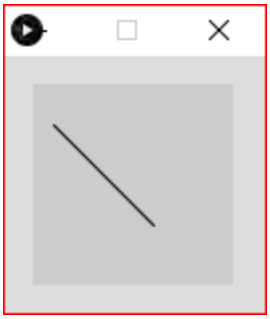
SAVE each sketch as “L05Q01”, “L05Q02, etc.

QUESTION ONE

Write a function called `line()` that behaves exactly like the built-in `line()` function, except that it takes two `PVectors` as parameters instead of four `floats`, and draws the line connecting the points represented by those `PVectors`. The body of your function should call the built-in version of `line()`, and you shouldn't return a value. Note that Processing won't get confused by the two versions of `line()`—it can tell them apart by their different argument types.

```
void setup() {
  PVector point1 = new PVector();
  PVector point2 = new PVector();

  point1.x = 10;
  point1.y = 20;
  point2.x = 60;
  point2.y = 70;

  line(point1, point2);
}
```

With the above starter code and your function `line()`, the result should look like the following.



QUESTION TWO

Write a function `doubler()` that takes an array of integers as input and returns a new array twice as long in which each element of the original array is given twice. That is, if you pass in the array `{1, 2, 3}` you would get back the array `{1, 1, 2, 2, 3, 3}`.

```
void setup() {

  int[] arr = {1, 2, 3, 4};
  int[] result;

  result = doubler(arr);
  printArray(result);
}
```

With the above starter code and your function `doubler()`, the result should look like the following.

```
[0] 1
[1] 1
[2] 2
[3] 2
[4] 3
[5] 3
[6] 4
[7] 4
```

QUESTION THREE: Direct Manipulation of an Image

In this exercise you will practice creating a direct manipulation interface by adding code to a sketch that allows you to move and scale an image.

1. The starter code is on the next page. Don't bother running it yet— it'll just crash.
2. Add any image you want to the sketch. The only constraints are that the image's width and height each be at least 100, and no more than 400. Now locate the `loadImage()` line in the sketch, and change it so that it loads the image you just added. *Now* you should be able to run the sketch and see your image.
3. Add code to the sketch that allows you to move the image by clicking and dragging anywhere inside its bounds. You'll need to change the code in a few places:
    - Add a global variable to keep track of whether or not the image is currently being dragged.
    - Now add a trio of hooks, `mousePressed()`, `mouseDragged()`, and `mouseReleased()`. These functions work together to move the image as the mouse is dragged, much like the demo code `SquareEllipse2.pde`. Make sure that the image moves only when you initiate a drag by clicking *inside* the image (by using the correct hit test in `mousePressed()`).
4. Notice that the starter code draws a small disc in the lower-right corner of the image. We'll use that as a handle to initiate scaling. When you click in the disc and drag, you move the disc around the screen. As that happens, the top-left corner of the image stays fixed but the image scales so that its bottom-right corner sits underneath the handle.

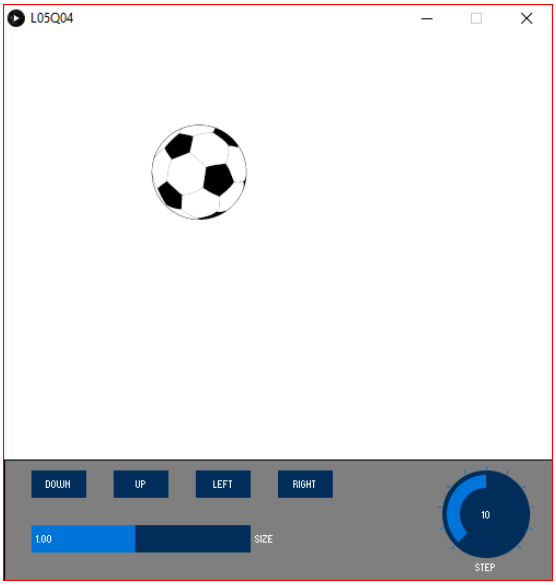
    You can add code very similar to the code for moving the image: a second global variable, and changes to the three hook functions. But you'll be hit testing against a small circle this time, instead of a big rectangle.
5. Your sketch must support *both* interactions. You should give precedence to the scaling handle over the image rectangle. That is, if the user clicks in the tiny region where the handle overlaps the image, it should initiate a scaling operation and not a move. (Therefore, hit test against the disc first, and the image second.)

```
PImage img;
Boolean imageActive = false;
Boolean handleActive = false;
int handleR = 50;

// The x and y position of the top-left corner of the image
float x;
float y;

// The width and height of the image
float w;
float h;

void setup()
{
  size( 500, 500 );

  // TODO: replace "ball.png" with an image of your choosing,
  // loaded into the data folder.
  img = loadImage( "ball.png" );

  // Start at the top-left corner of the sketch
  // with the original size.
  x = 0;
  y = 0;
  w = img.width;
  h = img.height;
}

void draw()
{
  // You probably do not need to change anything here.
  background( 255 );
  image( img, x, y, w, h );

  // Draw a handle for scaling the image.
  ellipse( x + w, y + h, handleR, handleR );
}
```

QUESTION FOUR: Simple Image Editor

For this question you will create a simple Image Editor using ControlP5. This question is similar to the previous question except the previous question used Direct Manipulation while this question uses ControlP5. The demo Code BusyBox.pde contains useful examples for ControlP5.

You are to mimic what is shown in the YouTube video: https://youtu.be/2457iROwrjQ



There is no starter code. You may want to use some code from Question Three above.

1. Add any image you want to the sketch using `loadImage()`.
2. You are to create a user interface using Control P5 with the controls shown.
3. The slider changes the size of the image (see the video).
4. The DOWN/UP/LEFT/RIGHT buttons move the image. The default is that pressing DOWN/UP/LEFT/RIGHT moves the image 10 pixels.
5. The dial changes the how many pixels the ball moves each time the user presses DOWN/UP/RIGHT/LEFT. This varies from a minimum of 1 to a maximum of 20.

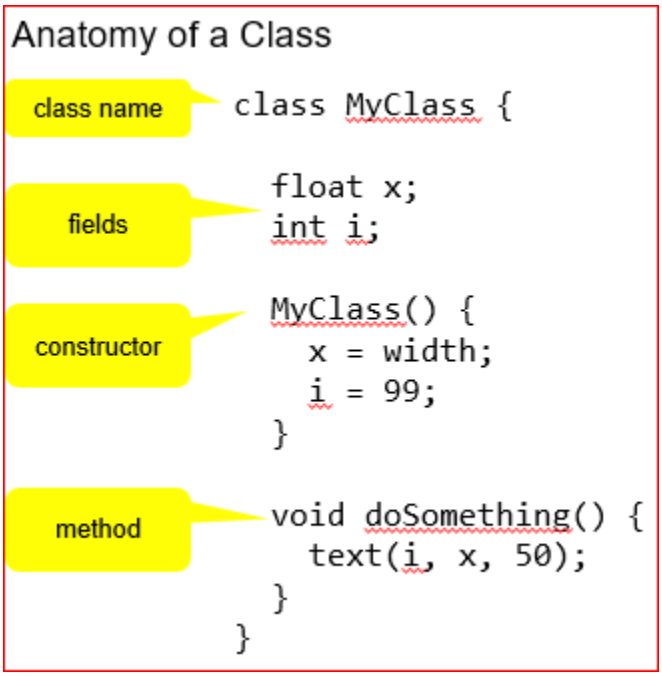
QUESTION FIVE: Practice for Assignment Five

In Assignment Five there is a Class called Word. This lab question is to familiarize you with that Class.

Cut and paste the following two segments of code into Processing. It should run without error. It displays word1 and word2 on top of one another so that you cannot see word1.

Change the Class called Word so that word1 and word2 are displayed randomly on the sketch window (i.e. they are not on top of one another). This is all you need to hand in for L05Q05.

Go through the Word code thoroughly so that you understand it. Doing this in the Lab will help a lot in doing Assignment 5. Remember the following slide from week 1 when we did a Recap of CS105.



As you go through the Class called Word, identify for yourself the class name, fields, constructor, and methods.

Then look to see how each is used in the main program paying particular attention to lines such as:

```
Word word1;
…

word1 = new Word("one");
…

rect(word1.getRectLeft(),
   word1.getRectTop(),
   word1.getRectWidth(),
   word1.getRectHeight());
```

Don't start Assignment 5 until you have a good understanding of the Word class. Get help in the lab or during office hours if you need to.

```
// CS106: Practice using the class Word.

import controlP5.*;
ControlP5 cp5;

Word word1;
Word word2;

void setup()
{
  size( 800, 300 );
  cp5 = new ControlP5( this );

  textFont( createFont( "Times", 18 ) );
  textAlign( CENTER );

  word1 = new Word("one");
  word2 = new Word("two");
}

void draw()
{
  background(127);
  // Draw word1.
  fill(255);
  rect(word1.getRectLeft(),
    word1.getRectTop(),
    word1.getRectWidth(),
    word1.getRectHeight());
  fill(1);
  text(word1.txt, word1.x, word1.y);
  // Draw word2.
  fill(255);
  rect(word2.getRectLeft(),
    word2.getRectTop(),
    word2.getRectWidth(),
    word2.getRectHeight());
  fill(1);
  text(word2.txt, word2.x, word2.y);
}
```

```
// Define a helper class that contains information about a single
// word, together with its position in the sketch
// window.
//  Written by Professor Craig Kaplan.
class Word
{
  String txt;
  float x;
  float y;

  // A basic constructor that stores one word passed in as a
  // parameter.
  Word( String txtIn )
  {
    txt = txtIn;

    // TODO: set x and y to random initial locations.
    x = width / 2;
    y = height / 2;
  }

  // The next four methods will make it easier to draw the
  // word and hit test it, by helping you calculate the
  // bounds of the rectangle enclosing the word.

  float getRectLeft()
  {
    return x - textWidth( txt ) / 2 - 5;
  }

  float getRectTop()
  {
    return y - textAscent() - 5;
  }

  float getRectWidth()
  {
    return textWidth( txt ) + 10;
  }

  float getRectHeight()
  {
    return textAscent() + textDescent() + 10;
  }

}
```

Submission

Submit all sketch directories from this lab as one ZIP file called L05.zip to the lab dropbox on Learn.

It is your responsibility to submit to the correct dropbox with the correct files before the deadline. Otherwise you will receive a mark of 0.