GRAPHICS PROGRAMMING, LAB 3

STARTING A SIMPLE VECTOR ANIMATION

NAME: ____

Introduction:

In previous classes we have talked about the difference between vector and bitmap images and vector and bitmap animations. In the homework assignment, you designed a pair of simple vector images that when shown sequentially should create the illusion of movement. Today you will create a new function *simpleAnimatedImage()* which will be used to create a simple, vector based, interactive graphics program.

Your ultimate goal in the final project will be to create an Interactive Graphics Animation**.** That is, a program that uses graphics, is interactive, and has at least one element that changes shape, look or position on its own. This lab will help you create an interactive graphics program, with rudimentary animation.

1. Application Template

I have created a Processing application template for you. It is available online (and in the lab folder), contains all of the required elements and is divided into sections (using comments) to help you organize your project II program. For this lab, you will need a copy of the application template. It will look like this:

```
/** ********************
   Class:
    Your Name:

   Note: Fill in the appropriate   sections to make your program.
******************** */

// Quick comment.
/* Longer comment */
/**
   Very long comment.
*/

/** ********************
  Variables:
  Variables provide a way to save information within your sketch and are used to
  control the size, position, shape, etc. of what   you are drawing.

******************** */

/** ********************
  setup():
  Use setup() to specify things that need to   be done once, when the sketch first opens.
******************** */
void setup() {

}

/** ********************
  Use draw() to specify things that you want   to have done repeatedly.
  NOTE: draw() must be present in your program,  even if it is left empty.
******************** */
void draw() {

}

/** ********************
  Event Listeners:
  Use event-listeners like keyPressed() to allow
  users of your program to cause things to happen.
******************** */

/** ********************
  Custom Functions:
  Functions are sections of code that you create and name.
******************** */

/** ********************
  Classes:
  Used to create objects.
  This section is for advanced students only.
******************** */
```

2. void *simpleAnimatedImage()*

In the homework assignment you were asked to write down the commands needed to create a pair of images, including the colors and the points you chose for those images. Add the ENTIRE function below (`simpleAnimatedImage()`) to the "Custom Functions" section of the template program. When you are done the "Custom Functions" should look like this:

```
/** ********************
  Custom Functions:
  Functions are sections of code that you create and name.
******************** */

void simpleAnimatedImage(float x, float y, float r, float s, int i) {
      pushMatrix();     // Save the state of the world
      translate(x,y);   // Move the world in order to move what is drawn
      rotate( radians(r) );   // Change angle of the world
      scale(s);               // Change world size by a percent ( 2.0 = 200%)

      if(i == 0) {
            // draw first image





      } else if(i == 1) {
            // draw second image





      }
      popMatrix();    // Reset the world
}
```

Remember that a function is just a section of code that has been grouped together and given a name. When we want to use a function, we "call" that function (using its name). The name of our function is `simpleAnimatedImage()` and we will "call it" in section 4. The word *void* which appears at the beginning of the function indicates that this particular function won't send any information back, when we call it.

3. Adding your images to the `simpleAnimatedImage()` function

In the homework assignment you were asked to write down the commands needed to create a pair of images, including the colors. The commands you created should be in sections 2 and 4 of the homework assignment. Type the commands from section 2 of the homework assignment into the area of the `simpleAnimatedImage()` function just below the comment:

```
        // draw first image
```

Then type the commands from section 4 of the homework assignment into the area of the function just below the comment:

```
        // draw second image
```

When you are done your `simpleAnimatedImage()` function should look like the code below, except that the commands (in bold) will be different.

```
void simpleAnimatedImage(float x, float y, float r, float s, int i) {
      pushMatrix();     // Save the state of the world
      translate(x,y);   // Move the world in order to move what is drawn
      rotate( radians(r) );   // Change angle of the world
      scale(s);               // Change world size by a percent ( 2.0 = 200%)

      if(i == 0) {
            // draw first image

            stroke(#000000);
            fill(#000000);
            quad( 0,-10, 55,10, 55,0, -35,20);
            line(0,-10, 40,-20);


      } else if(i == 1) {
            // draw second image

            stroke(#000000);
            fill(#000000);
            quad( 0,-10, 55,10, 55,0, -35,20);
            line(0,-10, -40,0);

      }
      popMatrix();    // Reset the world
}
```

4. Adding variables and calling the function.

If you run your processing program at this point, you will notice that nothing happens. We have added the function to our program, but we never "call" that function, in order to make it run. Go to the draw() function in your processing program and add the following line of code INSIDE the draw function:

```
simpleAnimatedImage (xP, yP, rA, sP, imageNumber);
```

Then go to the variables section of your program and add the following lines of code beneath the comment that indicates where the variables section begins:

```
float xP=0;                // X position
float yP=0;                // Y position
float rA=0;                // Rotation Angle
float sP=1;                // Size as a percent
imageNumber = 0;           // Which image to draw
```

When you are done the top part of your program should look like this:

```
/** ********************
  Variables:
  Variables provide a way to save information   within your sketch and use it to control
  the size, position, shape, etc. of what   you are drawing.
******************** */
float xP=0;                  // X position
float yP=0;                  // Y position
float rA=0;                  // Rotation Angle
float sP=1;                  // Size as a percent
int imageNumber = 0;                 // Which image to draw

/** ********************
  setup():
  Use setup() to specify things that need to   be done once, when the sketch first opens.
******************** */
void setup() {

}

/** ********************
  Use draw() to specify things that you want   to have done repeatedly.
  NOTE: draw() must be present in your program,  even if it is left empty.
******************** */
void draw() {
    simpleAnimatedImage (xP, yP, rA, sP, imageNumber);
}
```

Run your program! What happens? What do you see?

5. Getting your image on screen

When you created your image, you should have (as instructed) chosen the center of your image to be the point (0,0). When we call the function `simpleAnimatedImage()` we are sending in xP and yP as the offset for the CENTER of the world. If the center of the world is in the top left corner, your image will only be partially on screen. Go the to the event listener section of your program. Add the `mouseclicked()` event listener to your program and modify it so that the variables xP and yP are set to mouseX and mouseY when a mouse click is detected:

```
void mouseClicked() {
        xP = mouseX;
        yP = mouseY;
}
```

Run your program. What happens when you click (a click is a press and a release) the mouse inside the processing window? Why does it happen?

6. Adding more event listeners

Your image is a vector image, and one of the advantages to vector images is that they can change size with little distortion. Create a keyboard event listener (like you did in lab_1_1) in the event listener section. Within the `keypressed()` event listener add code that will change the value of sP when certain keys are detected. Specifically, do the following:

- If the keys 'l' or "L" are detected change the value of the variable sP so that it is larger.
    - Example:
```
if ( key == 'l' || key=='L' ) {
      sP = sP + 0.1;
}
```
- If the keys 's' or "S" are detected change the value of the variable sP so that it is smaller.

Run the program. Use the mouse to click in different places on the screen and press the 'S' and 'L' keys. What happens? Why?

7. Erasing old images

You will notice that your program doesn't erase the old images as your program runs. Modify your program so that at the top of the draw() function you reapply the background. Example:

```
void draw() {
        background(#FFFFFF);
        simpleAnimatedImage(xP, yP, rA, sP, imageNumber);
}
```

8. Animating your image

Animation for our purposes is movement (or change is shape, color, etc.) that happens without needing input from the user. The program you have now IS NOT an animation.

8.1. Movement (changing position and rotating automatically)

The variables xP and yP control where your image is drawn and we use them by sending them into the `simpleAnimatedImage()` function.

Go to the draw() function of your program and make it look like this:

```
void draw() {
   background(#FFFFFF);
   simpleAnimatedImage(xP, yP, rA, sP, imageNumber);

   xP = xP + 1;
   yP = yP + 1;
}
```

Run the program. Remember you can use the mouse to click in different places on the screen and reset the image. What happens? Why?

Go to the draw() function of your program and add the following line, below the xP and yP lines:

```
   rA = rA + 1;
```

Run the program. Now, what is happening? Why?

8.2. Changing Shape

Movement is one type of animation but changing shape is another. We are going to modify your program so that you can draw one or more different images! First we need to fix some parts of your program:

- The human eye can't detect many kinds of changes that happen faster than 30 frames per second. The default frame rate for processing is 60 frames per second. We will want to slow down the frame rate in order to see your pictures changing.
- Your screen is a bit small at the moment. We need to make it bigger.
- The changes you made in section 7.1 need to be undone so your image will stop automatically moving and rotating.

Make your the setup() and draw() sections of your program look like the following:

```
/** ********************
  setup():
  Use setup() to specify things that need to be done once, when sketch opens.
******************** */
void setup() {
    size(500,500);
    frameRate(20);
}

/** ********************
  Use draw() to specify things that you want to have done repeatedly.
  NOTE: draw() must be present in your program,  even if it is left empty.
******************** */
void draw() {
   background(#FFFFFF);
   simpleAnimatedImage(xP, yP, rA, sP);

   // xP = xP + 1;
   // yP = yP + 1;
   // rA = rA + 1;
}
```

Your program works by repeatedly calling the `simpleAnimatedImage()` function. When it calls the function it tells it where (XPos, yPos) to draw the image, how to rotate and scale the image (rA, sP) and WHICH IMAGE TO DRAW (imageNumber). Right now the image to be drawn NEVER changes. Look at the code below:

```
if( imageNumber ==  0) {
        imageNumber = imageNumber + 1;
} else if(imageNumber == 1) {
        imageNumber = 0;
}
```

- What does the code above do?

Add the previously shown block of code to the bottom of the draw function. When you are done your draw function should look like this:

```
void draw() {
   background(#FFFFFF);
   simpleAnimatedImage(xP, yP, rA, sP, imageNumber);

   // xP = xP + 1;
   // yP = yP + 1;
   // rA = rA + 1;

   if( imageNumber ==  0) {
         imageNumber = imageNumber + 1;
   } else if(imageNumber == 1) {
         imageNumber = 0;
   }
}
```

- The result is that the `simpleAnimatedImage()` function will alternate between drawing the two different images.
- NOTE: We aren't limited to just two images! We can make our if-else statement as long as we want and have it draw multiple different images in succession. But this will only work if our `simpleAnimatedImage()` function has more images to draw.