257 / 757
Programming in the Arts with Processing
Homework 5

Start: Tue 1.28.14
Due: in class Tue 2.4.14
Goal: Functions and Control

Course Web Site:
http://www.andrew.cmu.edu/course/60-257/

Reading:
Posted on the calendar web page available from the link shown above.

Assignment:
Define functions and use them as required below.

Specifications:

1. ____Define a **`figure( )`** function with arguments for its location. This function draws only one figure. It must not draw two. Other arguments are optional.
2. ____Declare two sets of variables. Each set will be used to locate and move one figure.
3. ____Define a function without arguments to prepare the screen. This should set the **`fill( )`** to a slightly transparent color and draw a rectangle that covers the entire window. Do not use **`background( )`** since it cannot be made transparent
4. ____Define a function (without arguments) to move two figures in the manner described in specification #8.
5. ____Define a function (without arguments) that calls your **`figure( )`** function two times to draw both figures.
6. ____Initialize the location variables to place the two figures at opposite corners of the window.
7. ____Initialize the movement variables to small values. The values for one figure must be slightly faster than those of the other figure.
8. ____Both figures follow the perimeter of the window when they as shown in class when this is assigned . One figure moves slightly faster than the other figure eventually catching up and passing.
9. Most of the body of the figures must always be visible – they may not move out of the window.
10. ____There is no user control or user input.
11. ____The **`draw ( )`** function must not have any API function calls. Your **`draw( )`** function must look very similar to this:

```
void draw( )
{
  prepWindow( );
  moveFigures( );
  drawFigures( );
}
```

where **`prepWindow( )`** draws a slightly transparent rectangle to give the visual illusion of movement, **`moveFigures( )`** alters the location and movement variables of the figures as needed, and **`drawFigures( )`** calls your **`figure( )`** function two times to draw the two different figures.

Advice:

1. Define the **`figure( )`** function first and ignore the animation. If your **`figure( )`** function does not work, you cannot animate it. Remember that it draws only one figure. To draw two figures, it must be called two times with different arguments.
2. Once you have the **`figure( )`** function working, add only enough code to move one figure along the top edge of the window.
3. Once the figure moves to the right side, add code to move it down the right side.
4. Next, add code to move the figure along the bottom edge.
5. Then add code to move the figure up the left side.
6. Repeat the process for the figure in the opposite corner.
7. There will be a lot of duplicate code in your program. If you have coded before and are familiar with arrays, you may use them but you may end up making the programming more difficult than is intended.
8. The **`draw( )`** function may not have any API function calls. Your **`draw( )`** function must look very similar to this:

```
void draw( )
{
   prepWindow( );
   moveFigures( );
   drawFigures( );
}
```

where **`prepWindow( )`** draws a slightly transparent rectangle, **`moveFigures( )`** alters the location and movement variables for both figures, and **`drawFigures( )`** calls the **`figure( )`** function two times.

Handin and Grading:

Turn in the entire folder to the handin folder for hw5 and then post the code on Open Processing.

Sample:

Very near the start of execution:	About 500 frames later and the faster figure is catching up:	About 700 frames after the start, the faster figure has passed the slower figure: