Algorithms for Game Design Exercise #1

This is completely optional. Do it after seeing Session 01. Or, don't do it. Either way.

---

1) Primitives

Examine each of the files I've given you. Make sure you understand:

1. How to draw individual dots and lines (`draw_notepaper.py`)
2. How to draw arcs, lines, individual lines, circles (`draw_python.py`)
    a) How annoying it is to find individual points for the control points on your lines and arcs
3. How Rects work (`draw_checkerboard.py`)
    1. How to modify them with `move_ip`, `inflate` them, etc
    2. How to draw them
    3. How to use them to draw ellipses
4. How to load an image and use it as a background (`draw_background.py`)
5. How to use the `blit` method to draw from one surface into another (`draw_blit.py`)
6. How to use `blit` to draw an avatar in reaction to key commands (`draw_mario.py`)

---

2) Drop!

I've given you a simple Pygame game, called `simple_drop.py`.

It is structured much like much Pygame code that you'll find on various tutorials on the web -- that is, not much thought has been given to structure.

Rewrite it! Or, refactor it! That is, I want you to move the code around, while still keeping the same functionality

You should probably have a separate file for the model and for the controller. Maybe the view as well?

Add Classes and Objects where you see fit. I have put the Platform into a class, just to give you an example of one. You may feel that it is inappropriate (perhaps there should be a PlatformManager class that handles all the platforms). Feel free to move code around.

Your game loop should be very clear.

3) Pygame Exercise

For this exercise, you will write some pygame code. Use the **`template_pygame.py`** file that I gave you as a starting point.

Each of you should have been given a number for this exercise. You will write pygame code to solve the problem associated with your number. Feel free to write code for more than just your number, if you like.

0. Draw a 50x100 pixel rectangle in the center of a larger black window. The rectangle's color should start out red, but as time goes on it should change color (you figure out exactly how). At the very least, the color should go through all the colors of the rainbow.

   If the user presses the plus key, the rectangle should get bigger. If the user presses the minus key, the rectangle should get smaller

   Quit when the user presses ESC or Q

1. Draw a line between the points (10,300) and (500, 20). Each frame, the endpoints should move, with the first point moving up one pixel and right two pixels. The second point will move down two pixels and left one pixel. If either point runs into the edge of the window, it should bounce.

   If the user presses the plus key, the line should get wider. If the user presses the minus key, the line should get narrower. Make sure the line width never makes the line disappear.

   Quit when the user presses ESC or Q

2. Draw an analog clock, with an hour-hand, minute-hand and seconds hand. Each should be a simple line from the center of the window. Make them different widths to differentiate them.

   The lines should move around the window, as a regular clock would. Or, for fun, make the clock run backwards.

   Quit when the user presses ESC or Q

3. Modify the logo bouncer so that it bounces a five-pointed star instead. The star should be about 100pixels in diameter and bright green. As time goes on, the star should start to fade -- to get more and more transparent, until it disappears. Then, it should get more and more opaque until it is back to fully bright green. It should take about 10 seconds to go from bright green to transparent and back to bright green.

   If the user presses the plus key, the transparency should speed up, getting about .5 seconds faster with each press.

   If the user presses the minus key, the transparency should slow down, getting about .5 seconds longer with each press.

   Quit when the user presses ESC or Q.

4. Draw a Pac-man and animate the mouth. Do so with Pygame's arc / line / etc primitives, not with images. Take a look online to see how fast Pac-man's mouth takes to open and close, as well as the geometry (for instance, the mouth doesn't open in an arc all the way to the center of the circle).

   If the user presses the left-arrow key, Pac-man should face left. If the user presses the right-arrow key, Pac-man should face right

   Quit when the user presses ESC or Q.

4) Boss Mode

Add a "Boss Mode" to your code from Part 3. That is, whenever the user presses the B key, the window should display a picture that looks like something you would actually be working on in an office -- a spreadsheet or word processor document. Keep the business picture until the user presses the B key again.

---

5) Textbook

If you haven't already, read Chapter 0: Games and Chapter 1: Introduction to How Games Work.

Then, read Chapter 2: Graphics and Images