Teach Pygame 1: Drawing and Animating

Most of the text below comes from a handout prepared for my *Programming Principles* class.

Introduction

Pygame is a Python module that contains a lot of code to help you add graphical output to your programs. We will use it for example programs, and those of you who are interested in doing so will eventually even be able to create animations and simple games with it.

One of the fun things about starting out with computer programming is that you often don't need to fully understand everything you're doing in order to make some cool things happen. A full understanding of how pygame works is well beyond where we're at right now, but there's still a lot you can do by simply copying pygame statements from a reference like this one and tweaking them a little bit to make them do what you want. Introducing pygame now will open up a lot of possibilities for demonstrating programming concepts later on.

The Template

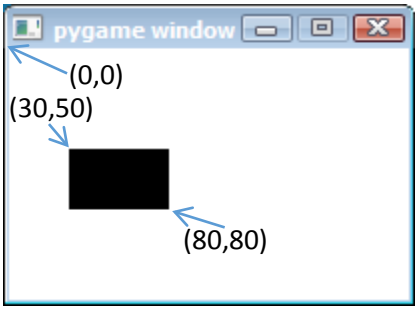
For now you should base your pygame programs on the **`pygame_template.py`** file from SLATE. This template imports the pygame module, initializes it, creates the drawing surface, and closes the drawing surface when the program is finished.

To use the template, copy it to a new file, give the file a name that describes what your program will do, and then change the docstring to contain a description of your program, your name and the date.

The Drawing Surface

The line below is probably the most important line in `pygame_template.py`:

```
surface = pygame.display.set_mode([800,450])
```



This line creates a drawing surface that is 800 pixels wide by 450 pixels high (feel free to change these numbers).

A drawing surface consists of a grid of pixels (points). Each pixel is identified by an x and y location that represents its horizontal and vertical distance from the top left of the surface.

In the picture above the dimensions of the drawing surface have been changed to `[200,125]`.

Rectangles and Colors

For many of the pygame drawing statements in the next section, you will have to set a rectangle (rect) and/or a color. The easiest way to do this is to include one or more of the two statements below just before the drawing statement. Make that you get the syntax, including capitalization, exactly right.

`rect = pygame.Rect(x, y, width, height)`

To make this work you have to replace *x*, *y*, *width* and *height* with numbers, where *x* and *y* are the location of the top left corner.

`color = pygame.Color("color name")`

The color name can be any HTML color name like "red", "white" or "blanchedalmond". See http://www.w3schools.com/html/html_colornames.asp for the full list. (If you know how rgb color values work you could also use **`color = pygame.Color(r, g, b)`** where *r*, *g* and *b* are replaced with integers in the range 0 to 255.)

Drawing Shapes

`pygame.display.flip()` ← VERY IMPORTANT !!!!

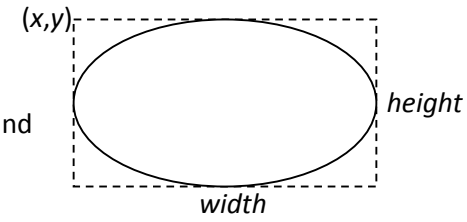
Each of the drawing statements below must be followed by this statement or they won't work.

`pygame.draw.rect(surface, color, rect)`

Draws the `rect` on the `surface` using the `color`.

`pygame.draw.ellipse(surface, color, rect)`

Draws an ellipse (oval) on the `surface` using the `color` and using `rect` as the bounding box, as shown on the right.



`pygame.draw.circle(surface, color, [x,y], r)`

Draws a circle on the `surface` using the `color`. The circle is centered at `[x,y]` and has a radius of *r*.

`pygame.draw.line(surface, color, [x1,y1], [x2,y2])`

Draws a line from the point `[x1,y1]` to the point `[x2,y2]`

`surface.fill(color)`

Fills the entire `surface` with the `color`.

Putting it All Together

The code below draws a rectangle on the screen:

```
color = pygame.Color("blanchedalmond")      # set the color
rect = pygame.Rect(50,50,100,100)           # set the rect
pygame.draw.rect(surface, color, rect)      # draw the rect
pygame.display.flip()                       # display the result
```

Do I Need 4 Lines For Every Shape? No. If you are drawing multiple shapes in the same color you only have to set `color` once and then use it multiple times. Similarly if you are drawing many shapes at once without pausing between them, you only have to call the `flip()` function once after you've finished.

Setting the Drawing Width

For rectangle, ellipse, circle and line drawing you can also specify one more numeric value inside the brackets. This value is the width of the line to use when drawing the outline of the shape. If you don't specify it, a line will be 1 pixel wide and a rect, ellipse or circle will be completely filled in.

In the example above if you change the rect command to the following…

```
pygame.draw.rect(surface, color, rect, 5)
```

…it will cause an aquamarine outline of a circle with radius 25, centered at the point [50,50] and with a line width of 5 pixels.

Drawing Text

Drawing text is a three stage process. First you create the font, then you create the text, then you draw it on the screen.

`font = pygame.font.SysFont("font name", font size)`

The font name must be in quotation marks. It can be any font on your computer. But three that every computer has are `"serif"`, `"sans-serif"` and `"monospace"`.

`text = font.render("your message", True, color)`

Creates a text object with your message in it using the given `color`.

`surface.blit(text, [x,y])`

This puts the text on the screen at the given location (*x* and *y* define the top left of the text). As with the other drawing commands, you must follow this with `pygame.display.flip()` or it will have no effect.

Sounds and Images

For instructions on playing sounds and drawing images, see my examples on line.

Animation

Animating means repeatedly redrawing a scene with minor changes. In the context of pygame, making a shape move from one place to another might look like this:

1. Initialize variables for shape position (e.g. `x` and `y`)
2. Clear screen and draw shape using the position variables.
3. Update the position variables (e.g. `x=x+1`, `y=y+1`).
4. Pause for 1/60 of a second (e.g. `time.sleep(1/60)`).
5. Go back to step 2.

This is the basic idea but you can adapt it to move multiple shapes at once, and to change shape size and color as well for fades and other transitions.

Allowing the User to Quit Early

The only problem with the above animation recipe is that while the loop is running, the user will not be able to close, move or minimize the window. To fix this, insert the following code between steps 4 and 5 of the loop above:

```
if pygame.event.peek(pygame.QUIT):
    pygame.display.quit()
    quit()
```

This code checks to see if there has been a QUIT event (triggered by the user attempting to close the window). If there has, it quits the program. Because you are briefly passing control back to the pygame event handler, it also allows it to deal with any move or minimize events that may have happened.

More Information

For more tutorials and complete documentation on pygame go to http://www.pygame.org/.