Pygame Assignment 2

For this assignment, you will build on the work that you have completed in assignment 1. This time, you will create a triangle that rotates and shoots projectiles.

At this point, you have already made your triangle rotate. You will create a new triangle of different dimensions and initial orientation. You will also simulate projectiles coming from the tip of the triangle and moving across the screen. You may reuse the routines in **triangleutils.py** from assignment 1 for this task.

Here are the files for this assignment:

shooting_triangle_game.py	The main program containing Pygame code
triangleutils.py	Routines for geometric manipulation of the triangle
test_triangleutils.py	A unit test file to test geometric manipulation routines
shootingtriangle.py	A class that represents the triangle
projectile.py	A class that represents the projectiles
shoot.wav	A sound file that plays whenever a new projectile in the game shoots off

Task Description

It may be best to start with the **shootingtriangle.py** file and the ShootingTriangle class within it. Here are the tasks to be completed for this file:

1. Set the values of **CLOCKWISE** and **ANTICLOCKWISE** in the class to appropriate values that indicate a contrast between these values. These

constants are used in the main program to indicate that we want to rotate in one direction or the other. See lines 91 and 93 of **shooting_triangle_game.py**.

2. Modify the constructor so that you have class members with appropriate data that may be used in other methods of the class. Initialize the polygon that represents the triangle here.
3. Modify the **draw** method to draw the triangle onto the display surface when called.
4. Modify the **rotate** method to compute and store the current rotation amount that the triangle will be rotated by. This rotation amount will be used when the draw function is called. Each call to this rotate method should change the amount of rotation by an increment equal to the value of the **ROTATION_AMOUNT** member. Use this member to set the value of the rotation increment. Please note the direction parameter. The triangle should rotate in the appropriate direction based on the direction parameter. The values passed to the direction parameter will be either **CLOCKWISE** or **ANTICLOCKWISE**. See lines 91 and 93 of **shooting_triangle_game.py**. ***Please note that this method does not do any drawing***. It simply sets the appropriate values so that when the draw method is called, the triangle will be drawn in its new angular orientation.
5. Modify the **get_front_location** to return a tuple containing the x and y coordinates of the “front” of the triangle. The front of the triangle is the point furthest from the center of the triangle.
6. Modify the **get_angle** method to return the current angular orientation of the triangle in degrees. Obviously, the angular orientation would need to be initialized and updated elsewhere.

In completing these tasks, you should have a rotating triangle just as before but packaged in a class structure instead.

Next, work on the **projectile.py** file and the Projectile class within it. You must complete the following:

1. Modify the constructor of the projectile class to capture relevant member variables that will support other methods in the class. Note that some code is already written for you. A Pygame rectangle is initialized and centered at the provided starting points passed into the constructor.
2. Modify the **draw** method to move the projectile across the screen. Use the **VELOCITY_MAGNITUDE** member of this class for the magnitude of the velocity. Here, it will be useful to think about the role of vectors.

    1. Any moving object can be seen as a vector of a particular magnitude with a component along the x axis and a component along the y axis.
    2. If the object is moving along the x axis at a constant speed and the height is not changing then the object has only an x component and zero y component to its velocity. The opposite is true for objects moving along the y axis but with no component velocity along the x axis.
    3. An object moving at an angle has non-zero x and y components to its velocity. Apply these ideas in this draw method to determine the values of increments for the x and y values to simulate motion at an angle. You may want to start by making your code move a projectile along only one axis or the other to start.
3. Modify the **get_location** method to return the current location of the projectile as a tuple with the x and y coordinates (x, y).

All other code is complete for you. If you finish these tasks then you should have a “shooting” triangle.

Good Luck!!! Have Fun!!!!