Programming Assignment #1
2D Graphics in OpenGL

Due Thursday, September 27 by 11:59pm

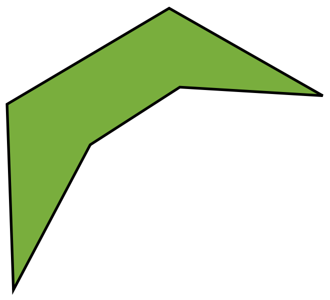
(The grade for this assignment is out of 100 points, but you can obtain up to a maximum of 125 points by doing more optional components. Additional points beyond 100 will be applied to your final score for all programming assignments.)

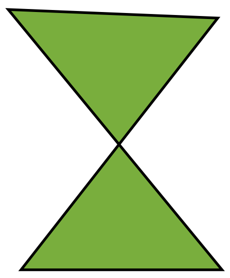
Required Components: (50 pts)

(1) Set up a working OpenGL programming environment. (10 pts)

(2) Using OpenGL programming, draw an image that includes at least 2 lines, 2 polygons, and 2 points, each with different attributes (*i.e.* different colors, different polygon types, dashed vs. thick lines, etc.) (5 pts)

(3) Write an OpenGL program that allows interactive creation of polygons via mouse input. That is, I click to create new vertices until I like the polygon, then make some indication (double click, right click, press a key, click a button; your choice) that I'm done, and the polygon is created. (20 pts)
NOTE: OpenGL does not handle concave or self-intersecting polygons properly. You do not need to worry about handling these either. (For example, if I click to create an hourglass figure like the one below, your program can draw a rectangle instead.)


Concave


Self-intersection

(4) Write an OpenGL program that allows interactive translation/rotation/scaling of polygons via user input. (This can be implemented as a standalone program, or as an extension to the program described in (3). The latter might prove useful for other parts of the assignment, but it's your choice.) You can implement this purely by keyboard input (*i.e.* the arrow keys translate by some preset amount) or by mouse input with a mode select (*i.e.* press 't' to put it into translate mode, then you can move the polygon by dragging it around the screen). (15 pts total; 5 pts for each transform type)
NOTE: Assume all rotation is just of the object itself; that is, the object is rotating around it's center. (Remember how we did translation about a non-origin point.)

Optional Components: (50-75 pts)

(1) Implement a user interface for the program in parts (3) and (4) above that can display what mode the user is in, lists the different available modes and how to use them, and provides any other information you feel is useful. (0-15 pts, depending on quality)

(2) Draw an interesting image using nothing but 2D primitives. (5-25 pts, depending on quality and complexity of the scene)

(3) Implement *Pong* (up to 25 pts total)

- **(1)** Have a moving "ball" that bounces properly off the walls. (15 pts)
- **(2)** Have 2 moving paddles, controlled by keyboard or mouse input. These paddles must not be able to leave the screen. (5 pts)
- **(3)** Solve for paddle/ball collisions, and change the path of the ball accordingly (10 pts)

(4) Implement *Asteroids* (up to 60 points total)

- **(1)** Draw the ship. (I must be able to easily tell which end of the ship is forward.) (5 pts)
- **(2)** Control the ship's rotation via user input. (5 pts)
- **(3)** Make the ship shoot via user input. (5 pts)
- **(4)** Control the ship's velocity by user input. User can accelerate/decelerate along the axis defined by the front and back of the ship. (5 pts)
- **(5)** Make the screen "wrap" like in the game. That is, if the ship moves off the right side of the screen, it should reappear on the left, and vice versa. Similarly for the top and bottom of the screen. (10 pts)
- **(6)** Add moving asteroids that can collide with the ship, destroying it and ending the player's "life", and can be hit by the shots fired by the ship. The asteroids can blow up when shot only once if you prefer, or they can split some number of times (as in the original game). (20 pts for asteroids that are destroyed when shot once, 30 pts for asteroids that split into multiple asteroids when shot the first time)

(5) Implement some other game of your choice. (Please propose it to me before starting work on it.) Some options would include *Missile Command*, *Break-Out*, or other games of that era. (Points negotiable)