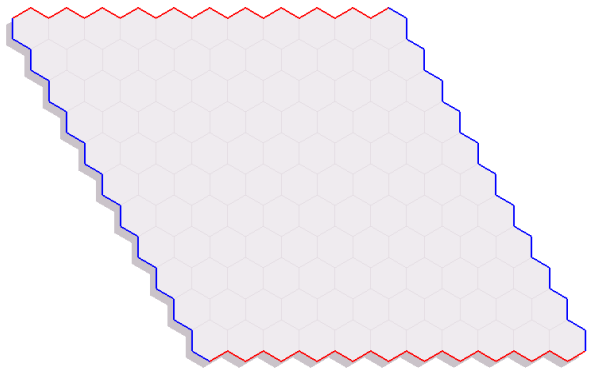
109 Extra Credit Write Up

Talon Baker • taebaker • 1515497

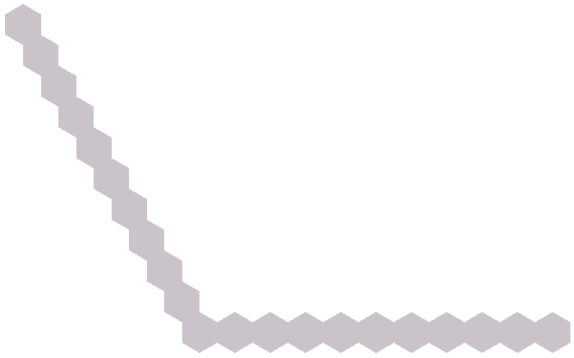
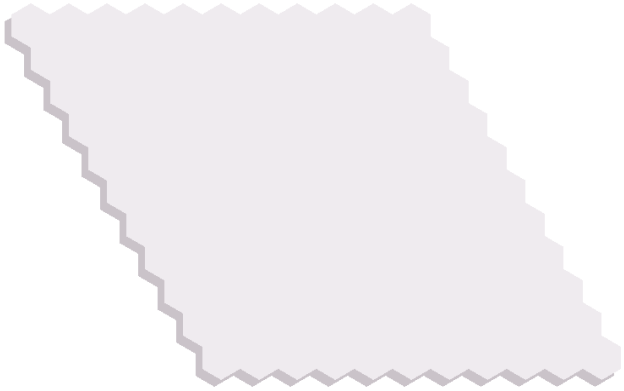
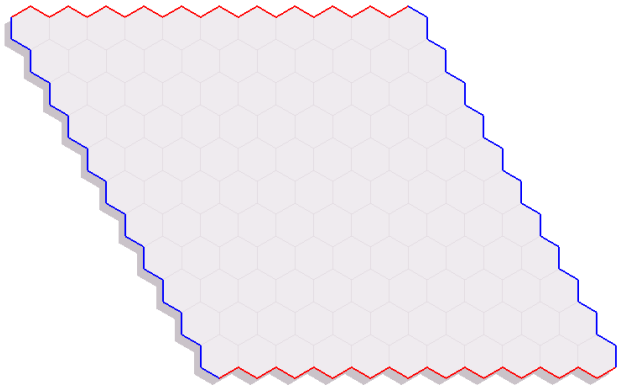
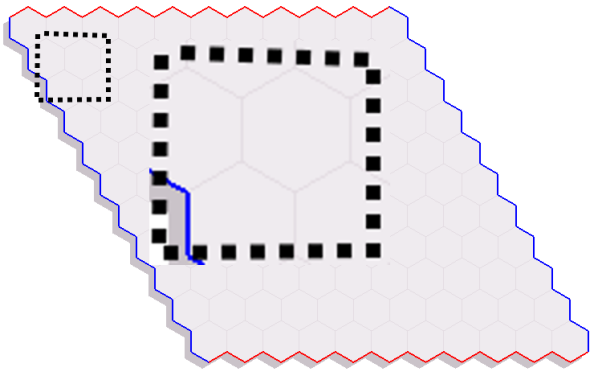
For this extra credit assignment I worked with **OpenGL** to create a graphical representation of the Hex Board for the player. This was done using the OpenGL utility toolkit **freeglut** and the **Eclipse IDE** in C++

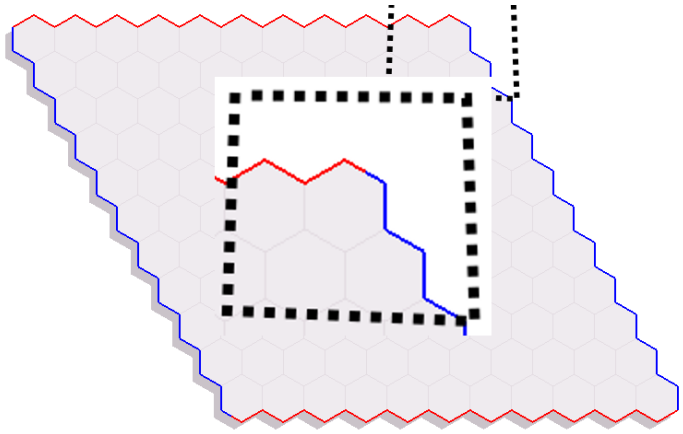
	This is the final product after all the steps are completed. There were 6 steps in total to output the completed Hex Board.

To draw all the hexagons, I used a starting location, in the upper left of quadrant II, and a unit hexagon. I coded all the math required to render one hexagon at a time using a GL_POLYGON line. Some of the general code/math is shown below:

```
179
180     currX = MASTER_X;
181         y = MASTER_Y;
182         x = currX;      //currX for line position when new line is started
183
184         //creating the background of the board
185         for(int i = 0; i < BOARD_SIZE; i++) {
186             for(int j = 0; j < BOARD_SIZE; j++) {
187                 glBegin(GL_POLYGON);                //starting drawing polygon
188
189                     //line RGB light gray
190                     glColor3f(0.937f, 0.921f, 0.937f);
191                     //adding all points to hexagon starting with north-most point
192                     glVertex2f(x + R * 0.0,           y + R * 1.0);
193                     glVertex2f(x - R * (sqrt(3) / 2), y + R * 0.5);
194                     glVertex2f(x - R * (sqrt(3) / 2), y - R * 0.5);
195                     glVertex2f(x + R * 0.0,           y - R * 1.0);
196                     glVertex2f(x + R * (sqrt(3) / 2), y - R * 0.5);
197                     glVertex2f(x + R * (sqrt(3) / 2), y + R * 0.5);
198
199                 glEnd();                            //ending drawing polygon
200                 //incrementing x by X_OFFSET
201                 x += 2 * X_OFFSET;
202             }//end - inner for
203             //resetting x, y for new row
204             currX += X_OFFSET;
205             x = currX;
206             y -= Y_OFFSET;
207         }//end - outer for
208
209     currX = MASTER_X;
210     y     = MASTER_Y;
211     x     = currX;      //currX for line position when new line is started
```

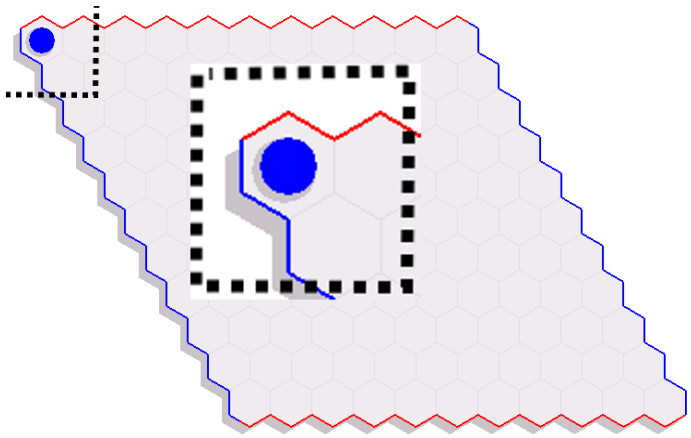
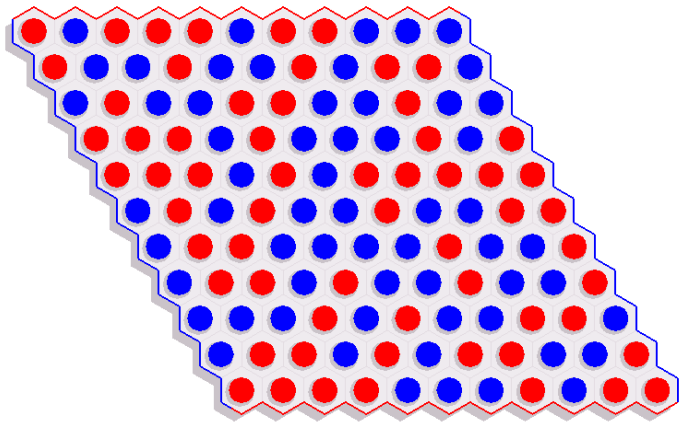
The following details the steps involved in drawing the Hex Game Board to the screen

	Step 1: A dark gray shadow is drawn to give the board some depth
	Step 2: A complete 11 x 11 board is drawn. The code is general enough to allow for the drawing of an n x n board.
	Step 3, 4: A red border is drawn on top and bottom of Hex Board and a Blue border is drawn on the left and right of Hex Board I basically just *walked* a line along each wall
	Step 5: Light lines are drawn to break up the board and make an easier time to see available spaces

	Step 6: The NE and SW corners of the map were broken up to include half red / half blue lines This step was surprisingly hard. This is the last step as it was simply drawn over the top of the finished map rather than included at red/blue line-drawn time

One of the sources I used was a Hexagonal Grids blog post by Red Blog Games (found here: https://www.redblobgames.com/grids/hexagons/). This illuminated many of the methods that I was trying to implement in my outputting of a Hex Board.

Pieces:

	Placing a piece: Placing a piece includes drawing a shadow for the piece, then checking what color of piece was placed and drawing a simple circle of that color using a GL_POLYGON at that position
	Completed board filled with randomly placed pieces

Post Mortem (or what I've learned over the last month):

I underestimated the complexity of OpenGL when I started this project. I thought that, due to OpenGL being so well-established and well-documented for C++ graphics, it would be relatively very easy to implement a GUI system for this Hex Game. This, however, was not the case.

What I should have done was use an IDE for GUI creation such as QT Creator. This would have saved me an immense amount of time and stopped me from calculating some of the math and debugging the output errors by hand. I tried to implement some kind of simple core game loop that would allow players to use the mouse to select which position to place a piece but my lack of OpenGL / freeglut library made this a very difficult thing to do by hand. An IDE specializing in GUI creation would have made this issue trivial.

A task that shouldn't have been as difficult as it was was figuring out how to pass the state of my board to OpenGL for piece rendering. I toyed with the idea of making my drawing functions methods of my HexBoard class, or even making a new *DrawingTools* class that was friends with the HexBoard class in order to access current board data, both these options seemed to cause conflicts with freeglut, however.

Another idea I had was storing the board data as a global list variable I called masterList using the extern keyword. I used extern because simply declaring the masterList variable in my main file seemed to be causing scoping or *not defined here* conflicts in my other cpp/header files.

In the end I ended up placing a global declaration of masterList in my hexBoard class header file along with the rest of the methods of the class. I included the HexBoard header in my drawing tools cpp file and this seemed to solve the access problem for everyone.

If I were to do this project over again I would do research into a modern library alternative to OpenGL. I mention this because, as part of my research for this project, I came across lots of discussions about whether OpenGL was worth learning over, say, a newer graphical library such as Vulkan, DirectX or WebGL.
I would also use a more user friendly alternative to simply text coding everything with an IDE like QT Creator.