**Reflection**

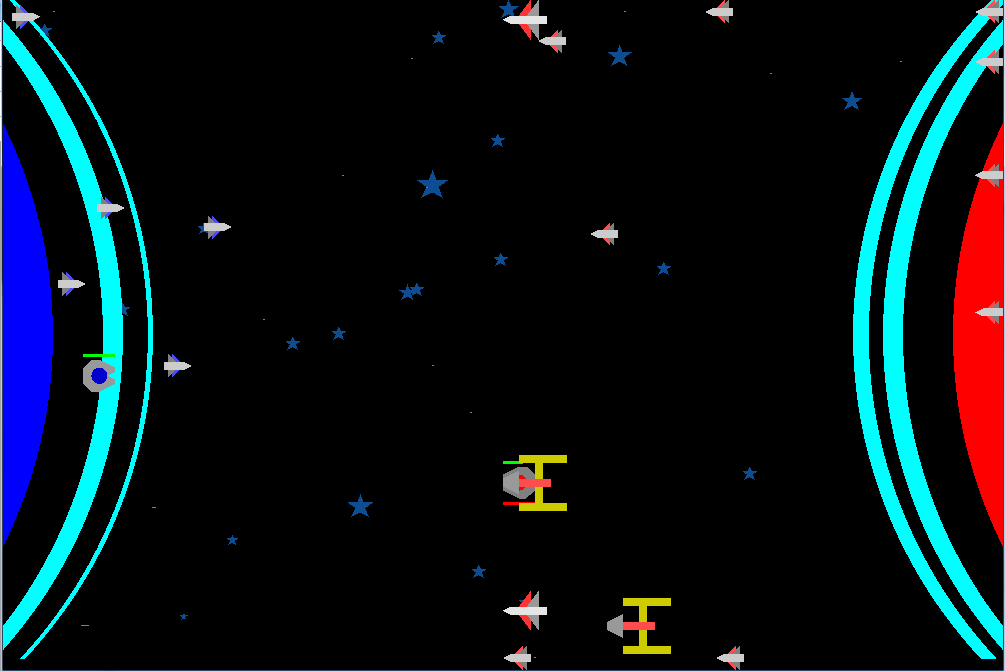
I would like to start by saying that this project was successful in the end, and achieved nearly all of the features I imagined it to have. The process of attaining this was not a simple one, however, since there were many obstacles that either slowed down the process of creating this program or prevented a feature from being included in the program. Most of all, though, is the time restraint put on this year’s project, as programming is always something that involves trial and error, no matter how much planning was involved beforehand.

The process to developing the project did not start as positively as it had in the previous years, since the knowledge required to make the program, the graphic programming for GLUT, was not taught for a very extensive amount of time, and even then was learnt through an independent study unit. The amount of material for GLUT was not troublesome, but it was the familiarity with the commands involved in the graphics programming for C++ that slowed down the process of producing this project in some parts. For example, drawing the planes and shields should not take an extensive amount of time, but because I was less familiar with the language, it took some time before I came up with the idea to use GLUT\_QUAD\_STRIP in conjunction with trigonometry and the unit circle to construct the shields, whose abnormal shape was not included in the GLUT language.

Besides the drawing of the different shapes and ships in the program taking up some time, the object-oriented structure of the program also took away some working hours; however, it was necessary and definitely a better choice considering the number of complex objects involved in the program. The spaceship object had to be capable of many functions such as firing, moving, knowing when to fire, when to move, and several other functions critical to the program, and the base object was also one that took a long time. I was happy having the experience of object-oriented programming last year so that this year’s programming structure was constructed with more ease and efficiency.

During the construction of this program, there were many problems I faced, both programming-wise and math-wise. I will list and describe the most prominent of each of the two categories mentioned.

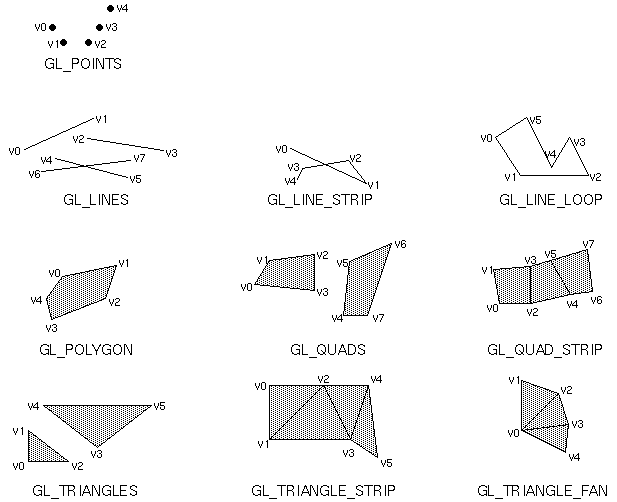
Shields



The shields for the planets were a challenge due to the fact that there was no command in GLUT that would draw such a shape. Furthermore, GLUT did not even have a command that would draw a circle. To solve this, I first thought that I can draw a polygon by drawing all of the points on the left border of each shield from top to bottom, and then draw all of the points on the right border of each shield from the bottom to the top.

Using the unit circle, I wrote the code that would do this, but it gave a shape that covered the entire region inside of the arc. After some research, I learned that it was due to the fact that GL\_POLYGON can only draw a convex polygon, and since the points that made up the arc did not form a convex polygon, it did not draw the correct shape.

I then did some more research, and found out that GL\_TRIANGLE\_STRIP could do this instead, since it can draw the shield one triangle at a time, and eventually the entire arc. However, after some trials, it became apparent that this was not feasible, and I tried the other alternative with GL\_QUAD\_STRIP.

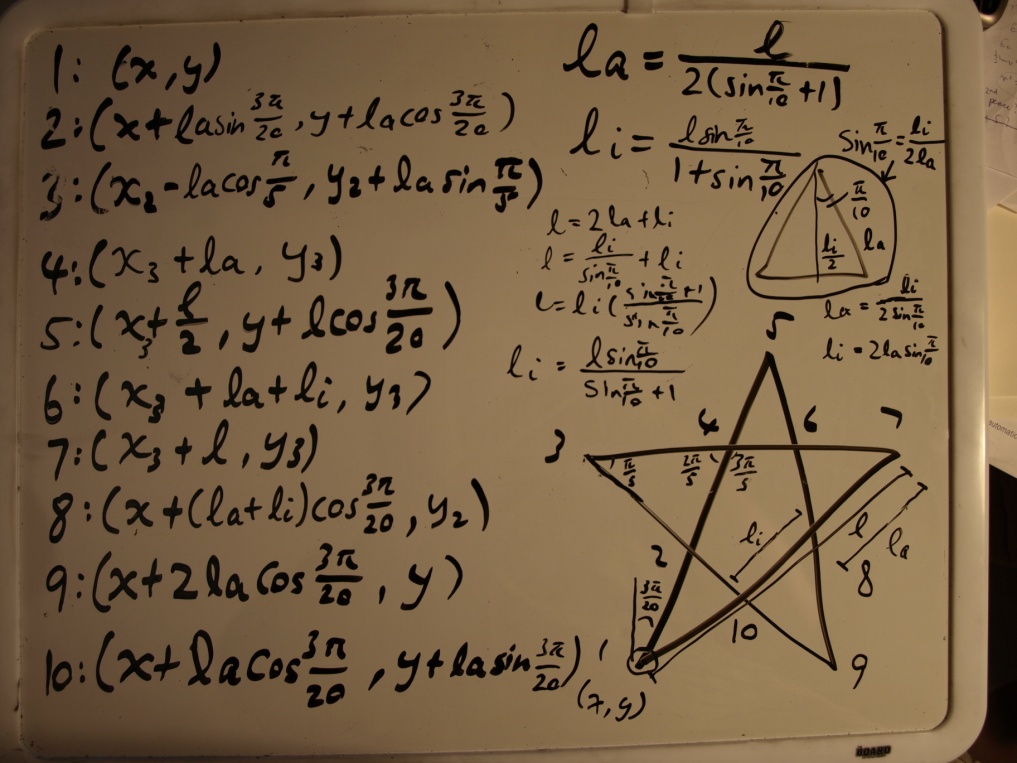


I drew each of the boundaries of the arc consecutively; when the top point of the left boundary is drawn, the top point of the right boundary is drawn, followed by the second point of the left boundary. This worked spectacularly, and the arc appeared like how I imagined it on the screen.

Stars

The most difficult math problem I came across was drawing the stars in the background. A five-pointed star is not a basic object, and I do not have the same built-in command of DrawStar like that of in Turing to use. Since I wanted to draw stars of different sizes, I should be able to specify a size for the length of the star that can then be used to calculate the location of the different points on the star.

To solve this, I dissected the dimensions of a five-pointed star.

I noticed that a star is comprised of a regular pentagon in the middle and five isosceles triangles emanating from the sides of the pentagon, so I saw that if I knew the lengths of these three variables, l, la, and li, I can construct a star from it. Below is a picture of how the value for each point was derived.

Although this program had almost everything I first envisioned it would have, I would make a few changes. First, I would like to add some pictures into the program. Because of the complexity and time it takes to actually putting a picture into a GLUT program, I did not decide to include one. Putting a picture background or planets would definitely make this game more visually appealing. Furthermore, I would also like to make the artificial intelligence in this game more human-like. Currently, the ships move in random directions, which makes them unpredictable, but also goalless and unable to play out intelligent tactics such as aiming for the other side’s ships. Lastly, I would try to make the game more efficient and not processing-intensive. When the number of ships or the number of bullets in the program increases, the speed drops almost exponentially due to the fact that collision detection has to run through nested for loops. I believe this is actually the reason why the school computers cannot run this program with the right ship colours, and rather makes all the ships red.

Given the amount of time allocated and the difficulty of the situation, I am very happy to be able to present a program that I have imagined from the start.