**Problem Statement**

The goal of this assignment is to create an OpenGL program that displays a series of golden rectangles in a screen window of 600 pixels wide by 400 pixels high. The program must calculate and display the largest golden rectangle that will fit in the previous rectangle drawn until the size of the rectangle is one pixel high.

**Proposed Solution**

In the design phase of the assignment we knew that we needed to perform the calculations to draw the first golden rectangle. This would be our starting point in the assignment. This rectangle needed to be centered on the screen. However, the screen size did not lend itself very well to a perfect golden rectangle. After the first rectangle was drawn, we knew that we could use the same technique to draw the other remaining rectangles. With this concept in mind we were able to complete the program as required. In the sections below we will outline the steps we took to complete our program.

**Strategies and Algorithms**

One of the first issues we faced in this assignment was centering the first rectangle on the screen. To solve this we created a function that simply drew the first rectangle. Passed into this function was the screen’s height and width. This allowed the function to calculate the exact location to draw the rectangle. We drew the rectangle using the glRect\*() function. This allowed us to draw rectangle using two points: one start point and an adjacent endpoint.

After the first rectangle was drawn, we saved the endpoint of the rectangle into a GLintPoint variable. It was now time to call another function to draw the next rectangle. We passed in three parameters: the GLintPoint, the last rectangles height, and the last rectangles width. With the previous rectangles height and width, we calculated the largest golden rectangle that we could draw. We then used the endpoint from the previous rectangle and used this as our starting point of the next rectangle. We again drew this rectangle using the glRect\*() function. Like before we then saved the endpoint of this newest rectangle and called our next function passing in the GLintPoint, height, and width. This process was repeated over and over until the rectangle’s calculated height was less than one pixel high.

The last item we needed to implement was the aspect ratio control. This allows the rectangles to be redrawn in the correct aspect ratio when the screen is resized. This used the glutReshapeFunc() function to run our callback function to redraw the rectangles. Our callback function sets the appropriate viewport size and resets the gluOrtho2d() window to the new size. The results look great, and the rectangles are redrawn to the correct aspect ratio.

**Mathematical Techniques**

Throughout the program we had to base our calculations on the ratio of the golden rectangle. This ratio is 1 / or 1 / 1.618033989. This was straight forward and the mathematical operations were not that difficult.

**Program Output/Results**

Our program uses the logic we just described and displays a series of golden rectangles until the size of the rectangles is less than one pixel. Each of the respective rectangles are different colors to makes then easier to see.

**Summary**

This assignment combines all of the information that we learned in class along with the information covered in the book. As a team, this program was pretty straight forward. The calculations required to draw the golden rectangles was easy to picture visually. This allowed us to work through each of the functions in the program. Finally, our program fully meets the requirements as specified by the assignment.