INF-1100: Introduction to programing and computer behavior

Assignment 3 report

Amund H. Strøm

October 08, 2021

Introduction

In this assignment we were tasked to draw a teapot using triangles and filling them with different colors. The coordinates, color and functions necessary to draw the teapot was already given to us, so our task was to complete the empty functions that would create the teapot. In this report I am going to talk about how I solved the different functions, some problems I encountered on this path, and some things I found interesting.

Technical Background

In this assignment the focus was on getting an understanding of the use of “structs” and algorithms. “Structs” are used to create a list of different variables under a single name and makes the variables available via a single “pointer” or by using the declared name of the “struct”. This was used to declare all the variables needed to create triangles for the teapot. Algorithms are precise instructions to complete a task or solving a problem. For the case in this assignment an algorithm was used to fill triangles with different colors, but more on that later. There was also a focus on understanding pre-written code, since the assignment gave us a bunch of code that we had to understand before being able to complete the rest of the assignment.

Canvas

Something that is important to note is that the canvas for this assignment has the lowest value in the top left corner of the screen. Meaning that it is like a normal coordinate system, just flipped upside down. It would look something like this.

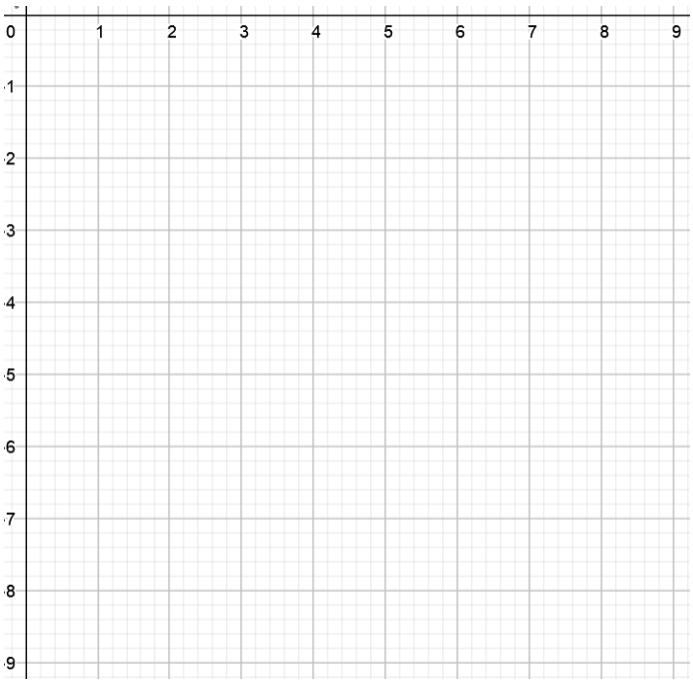

Figure 1: Illustration of the canvas

Design and implementation

This assignment gave us a lot of pre-written code, but I am not going to talk about this since I didn't write it and we was promised that it worked, which it did. I am instead going to talk about the 5 different functions we were tasked to complete. I will talk about each of them individually, and in the order I chose to complete them.

First, I drew a wireframe of a triangle by completing the "DrawTriangle" function. To draw a triangle you need 3 corners, and a corner consists of 1 x- and y-axis coordinate. This means that there are 3 unique x- and y-coordinates, summing up to 6 different coordinates. With this information we simply draw 3 different lines, sharing these coordinates, ending up with a closed triangle.

The Second function I completed was the “TranslateTriangle” which moves the triangle to the middle of the screen. This was done by dividing the value of the x- and y-axis for the screen by 2, and then adding those numbers to the same coordinates used to create the wireframe of the triangle.

Third function to complete was the “ScaleTriangle” function, which shrinks or grows the triangle without changing other properties of the triangle. To do this I multiplied the coordinates of the triangle with a scaling number. This number can either be less than 1.0 which means that the triangle will shrink, or it can be greater than 1.0 which means that the triangle will grow.

The use for the fourth function “CalculateTriangleBoundingBox” was to create a square that perfectly fit the triangle inside it. To do this we compare the 3 different x-axis coordinates and see which of them has the lowest and the highest value. We do the same with the y-axis coordinates. Then we make a point out of the lowest values of the x- and y-coordinates, this will result with a point that is in the top left of the triangle. We also subtract the highest x-coordinate with the lowest x-coordinate, this will result in the width of the triangle. We do the same with the y-coordinates and this will result with the height of the triangle. With this information we can create the bounding box for the triangle. To check if I got the right values, I drew the bounding box using 4 lines and using the above information. This resulted in a box that perfectly fitted the triangle inside it.

Now for the fifth and final function, “FillTriangle”. This function uses an algorithm to fill a triangle with a given color. My approach to write this algorithm was with the use of the already given function, “get_pixel”. What this function does is that it checks what color a pixel has, and if the pixel is inside the surface of the canvas. With this function I checked every pixel from left to right, and top to bottom, inside the bounding box for every triangle. And if the function noticed a pixel with the same color as the triangle it is currently filling, it would save this coordinate, and continue checking from left to right. Eventually it will notice another pixel that matched the correct color and draw a line between these two points. When it reached the end form left to right, it will go down 1 pixel, and do the same with this axis. In the end, it will result with a filled triangle.

Discussion

When I was doing the “CalculateTriangleBoundingBox” and “FillTriangle” function, I collaborated with two fellow students, Morten Jansen and Tarek Lein. We used most of our time struggling with the creation of these functions, and both functions ended up having the same problem and solution. Which I found very interesting. In the “CalculateTriangleBoundingBox” function we are comparing each value to check what integer has the lowest and highest value. We did this using the “greater then” or “less than” symbol. But instead of using these symbols, we were supposed to use “greater than or equal to” and “less than or equal to” symbols. Which makes sense, since in some cases two integers can have the same value, for example in a right-angled triangle. We encountered the same problem in the “FillTriangle” function, inside the “for loops” that counts every x- and y-axis coordinates. Here the integer is supposed to be “less or equal to” the height or width of the bounding box. I understand that in some cases the edge of the triangle can be at the same location as the edge of the bounding box. But what I don’t understand is how this simple chance can make such a huge difference.

Figure 2: teapot with “less or equal to” symbol

Figure 3: teapot with “less than” symbol

Conclusion

To sum up, the task of this assignment was to create a teapot using triangles. We got a lot of pre-written code that was necessary to understand before tackling the assignment. Our task was to complete 5 different functions that would create the teapot. The major problem that I spent most of my time figuring out was the use of “less or equal to” and “greater or equal to” symbols.

References

Making a teapot in C with triangles. (2013, October 5). Hentet fra cprograming.com: https://cboard.cprogramming.com/c-programming/159716-making-teapot-c-triangles.html

Wikepedia. (2021, 10 04). *struct (C programming language)*. Hentet fra Wikepedia: https://en.wikipedia.org/wiki/Struct_(C_programming_language)