Inf-1100, Assignment 3

Submission: 23:59 October 08th, 2021

Overview

In this assignment you will implement C code for drawing filled triangles in a window on the screen.

Description

Your task is to complete the functions listed below. Together these functions provide functionality to draw filled triangles on the screen. In addition, the functions enable you to scale the size of a triangle and move (translate) a triangle to a given position on the screen.

Each triangle is described by a triangle_t data structure. All the functions below accept a pointer to a triangle data structure as input. Each function needs to access and modify the fields of the data structure as appropriate.

- **ScaleTriangle** - Scale the size of the triangle. The scaling factor is specified in the scale field in the triangle data structure. The scaling factor is a floating point number. A factor less than 1.0 should decrease the size of the triangle, while a factor greater than 1.0 should increase the size.
- **TranslateTriangle** - Move the triangle to a specific position on the screen. The position is specified in the triangle data structure in the tx and ty fields.
- **CalculateTriangleBoundingBox** - Calculate the size of a rectangle that is just large enough to contain the triangle. The bx, by, bw, and bh fields of the triangle data structure should be initialized with the appropriate values.
- **FillTriangle** - Fill the triangle with a color. The fillcolor field in the triangle data structure specifies the fill color. There exist many different approaches for filling triangles and polygons in general. See the Resources section below for some inspiration.
- **DrawTriangle** - Draw a scaled, translated, and filled triangle on the screen.

In this assignment, there should be no need to define functions beyond those already defined in the files we provide. We recommend starting with code to draw a wireframe version of the triangle on the screen (by use of the DrawLine function).

Then, either proceed with the code to scale the triangle, or the code to move the triangle to a given position on the screen. Note that triangle scale is specified as a floating-point number, while the triangle coordinates are integer variables. You will need to think about conversions between integer and floating-point numbers to make the scaling function work properly. When scaling, translation, and drawing works, embark on the code for filling the triangles.

Once you have working code for drawing triangles, you can test it by drawing a colorful teapot on the screen, using a triangle-based model that we provide. When rendered on the screen, the teapot should look like this:

The teapot model is represented as an array of pre-initialized triangle data structures. The model is contained within a 1000x1000 box, with coordinates ranging from -500 to 500 on both the x and y axis. To properly draw this model, you must translate it to the middle of the screen. Note that in this assignment we set the resolution of the screen to 1024x768 pixels. If translated to the middle of the screen, the teapot model does not have y coordinates that exceed the resolution of the screen. Thus there is no need to scale the teapot before drawing it.

C solution

Your starting point is the following set of files:

- drawline.h - Specifies the interface of the drawline function. Do not modify this file.
- drawline.c - Implements the DrawLine function.
- triangle.h - Specifies the triangle data structure and the interface to the DrawTriangle function. Do not modify this file.
- triangle.c - Empty stubs for DrawTriangle and other functions. You will be editing this file.
- teapot_data.h - Coordinates for the classic teapot model.
- main.c - Contains the main function, calls to initialize SDL, and calls to draw some example triangles.
- Makefile - A Makefile for compiling the code.

We've bundled all of these files in a zip file. The zip file is located in the same folder as this document. Use this zip file as a starting point, as it also includes the necessary SDL files.

Getting started

When the program compiles, try running it, then start gradually making changes to triangle.c and see what effect it has.

Resources

Makefile and make:

- Tutorial

A few approaches to filling algorithms for polygons. Note that most of the approaches described below are more general in that they assume arbitrary polygons. Your code only has to deal with triangles.

- Efficient polygon-filling algorithms for raster displays
- Polygon Area Filling.

Report

Describe the implementation of the programs you made. Do not include the source code in the text (it should be submitted as separate files). However, the source code should be readable enough, both in structure and the way it is commented, for another programmer to verify its correctness. In other words, your report together with the source code needs to convince us that your programs work, not execution of the actual programs!

Include a 'discussion section' in which you note your observations on this assignment. Maybe you observed something interesting (or just surprising to you). If so, write about it in your report.

Also report any particular assistance you got from course staff or your fellow students that has been crucial to the fulfillment of the assignment. You are strongly advised to write documentation/notes along the way.

Do not refer to the C code as self-explanatory. We encourage use of figures to illustrate data structures and pseudo-code to illustrate algorithms. We do not expect the report to contain any formal proofs of correctness, but we do expect that you argue why your program is correct. For example, explain all the corner cases of triangle filling and describe how your algorithm handles them.

Submission

You should submit the report ('assignment3.pdf') and your code no later than October 8^{h} 23:59. The submission should be done using Canvas.

You should submit two files, the report and the C source code. We only accept the report as a single file in the PDF format (not scanned). The C source code should be packed in a 'zip' file.