Inf-1100, Assignment 3

Submission: 23:59 October 14th, 2022

Overview

In this assignment you will implement C code for drawing filled triangles in a window on the screen. You should work in groups of two.

Description

Your task is to complete the functions listed below. Together these functions provide functionality to draw filled triangles on the screen. There is also a function to 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.

These functions should be implemented collaboratively

- **translate_triangle** - 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.
- **calculate_triangle_bounding_box** - 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.

These functions should be implemented by each member of the group individually.

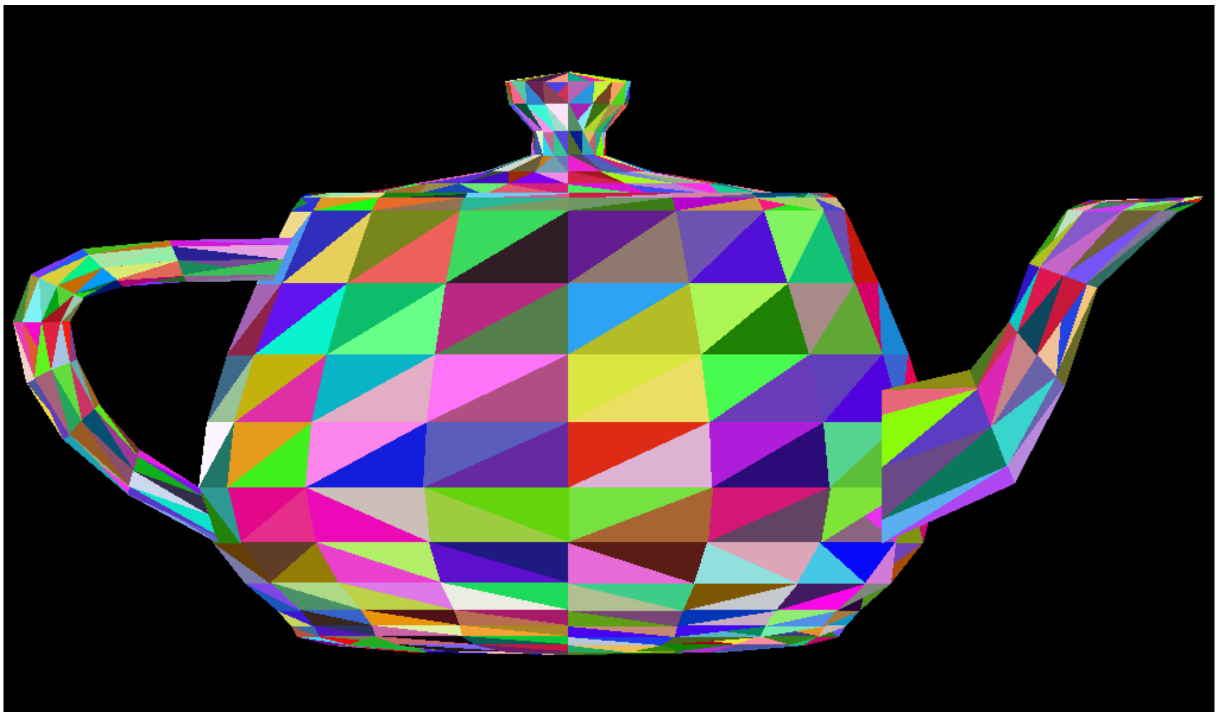
- **a_fill_triangle/b_fill_triangle** - Fill the triangle with a color. The fillcolor field in the triangle data structure specifies the fill color. There exist many different approaches for filling triangles and polygons in general. See the Resources section below for some inspiration.
- **a_draw_triangle/b_draw_triangle** - Draw a translated and filled triangle on the screen.

The implementations of the filling of the triangle shall follow two different algorithms (for example: one implementation using getpixel, and one calculating the fill mathematically), to be implemented in the a/b functions mentioned above.

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 draw_line function).

Then proceed with the code to move the triangle to a given position on the screen. When drawing and translation works, start 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. We set the resolution of the screen to 1024x768 pixels. If translated to the middle of the screen, the teapot should fit within the bounds of this window.

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 two draw triangle functions. Do not modify this file.
- fill_triangle.h – specifies interfaces to the common functions. Do not modify this either.
- triangle_common.c - Empty stubs for the functions you should solve in common and some filled-in debug functions. You will be editing this file.
- a_fill_triangle.c – Empty stub for one of the fill algorithms. You must complete this.
- b_fill_triangle.c – see above.
- 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 these files in a zip file that 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 and see what effect they have.

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
- "Filled Triangles" chapter from Computer Graphics from scratch
- Point in triangle test

Report

There should only be one report for each group. 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 14th 23:59. The submission should be done using Canvas. The submission should be identical (except the filename for the submission) for both members of the group.

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 compressed archive, such as 'zip', 'rar' or 'tar.gz', in the following way:

- The archive contains a folder with your username and the assignment number as its name, eg. eho033-assignment3/. This folder contains your other files.
- The archive itself has the same name, eg. eho033-assignment3.zip

We ask you to do this so that the people looking at your submissions can keep track of which files belong to which student, which otherwise is a lot of extra work for them. Alternate formats will not be accepted.