Assignment 1

COMP 3501

Date: 17 September 2014

Due: on <u>October 2, 2014 before 22:00 (10:00 PM)</u>
Submission: Electronic submission on webCT.

Assignment Objectives:
- a. Familiarization with OpenGL initialization
- b. Familiarization with call back functions.
- c. Understanding the classes and the relationships between the classes

2. Working with Visual C++
   - a. Setting openGL
   - b. Using a debugger

Grades:
1. Assignment total marks: 100%.
   .

Instructions
1. Log into the course CULearn account and download the zip file of the project
2. Compile and run the project.
3. Modify the project code according to the instructions below
4. Once the work is complete
   - a. Create a zip file of your project
   - b. Make sure that all required files are included in the zip file
     - i. Source code
     - ii. Images
     - iii. Any other files
   - c. Copy the zip file into a new directory (e.g., myTestDirectory)
   - d. Extract the files from the zipped file in the new directory (myTestDirectory)
   - e. Open the project
   - f. Rebuild the project
   - g. Test the project to ensure that it executes
5. If the project passed the test
   - a. Submit the zipped file using CULearn

Part I (60 points)

1. Task 1 – Render a triangle

Purpose:
Create a window and draw a single triangle in it.

To do:

1. Down load the project in assign1.zip into a local directory on your computer. Extract the files. You should see two folders:
   - 1.1. GL – this file includes all the required files for using OpenGL in windows: header files, library files and dll files.
   - 1.2. Assign1 – this directory contains the project file. The project currently renders and empty window. The project is Visual C++ 2010.
2. Open the project and modify code.
   - 2.1. Create a function that draws a single triangle. The function call is drawTri(v1, v2, v3, colour) where v1, v2, v3 are the three 2D vertices of the triangles and colour is the colour of the triangle. Use float as the vertex coordinate type. For this assignment you can use glBegin() and glEnd();
   - 2.2. Create a window titled "Single Triangle"
   - 2.3. Using the function in 1 render a single triangle in the window as shown in Figure 1.

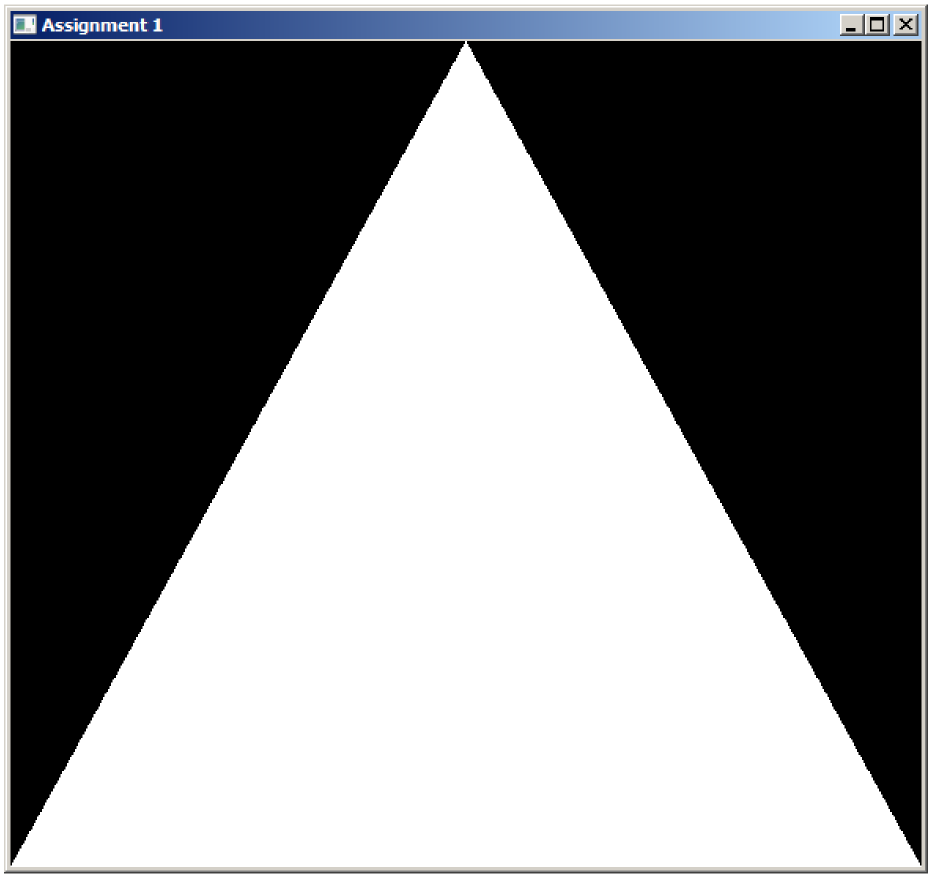

Figure 1: single triangle

2. Task 2 – Render triangles recursively

1. Create a second window titled “Recursive Triangle”
2. Create a recursive call that renders numerous triangles as depicted in Figure 2. The function call should be recursiveTri(v1, v2, v3, n) where v1, v2, v3 are the vertices of the first triangle and n is the number of recursive call. Namely, if n = 1 a single triangle is rendered, and if n == 2 three triangles are rendered. In a recursive call you will do the following:
   - 2.1. Compute the midpoint of each edge (v1v2, v2v3 and v1v3).
   - 2.2. At each intermediate step, use the six points do a recursive call to render three sub triangles recursively.
   - 2.3. When you reach the end of the recursion use your function from step one to draw the triangle.
   - 2.4. Use a different colour to render each of the three triangle. You can either use global or constant colours, or pass the colours using the recursion.

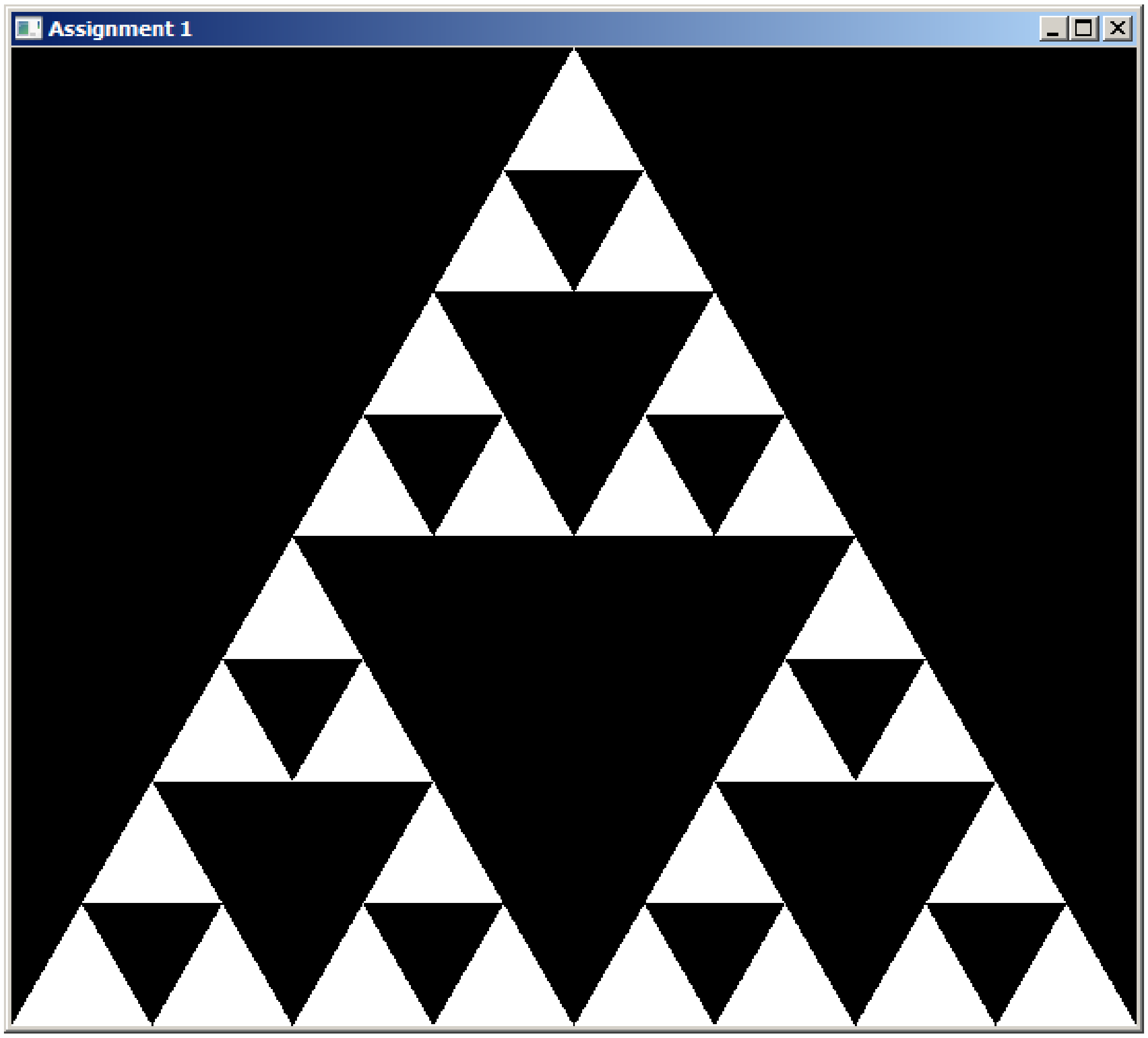

Figure 2: Recursive drawing of triangles. Note the rendered triangles are white and are all of the same size.

Part II (40 points)

Submit a separate program to complete Tasks 3 and 4.

3. Task 3 – Render a triangle using shaders

Repeat Task 1 but use shaders to draw a triangle

4. Task 4 – Render a triangle using shaders

Repeat Task 2 but use shaders to draw the triangles. In this task you will not use function in Task. Rather, you will do the following:

a. Using the recursive function you will create a single array containing all the vertices of the triangles.
b. You will pass the complete array to the GPU for rendering
c. You will use the shaders from Task 3 to render the triangles.

Part III – Colouring the triangles Bonus (15)

a. Add colours to the triangles generated in task 4. Again you will create a single buffer containing geometry and colours and transfer it to the GPU or two buffers (one for vertices and one for colours).

Part IV Bonus (10)

a. Add a displacement parameter "dispX" which will be incremented by the application at every time tick.
b. Pass the parameter dispX to the shader
c. In the vertex shader modify the x-coordinate of each vertex by adding the the dispX the value of each vertex, creating an effect of scrolling along the x-axis. Note, that when the triangles disappear from one side they will appear on the other side.

Part V Bonus (<=15)

a. Do other interesting effects with the shaders (e.g., blinking triangles). Explain what you have done. Make sure that it is your own work.
b. Decide how many points it is worth (no more than 15) and justify it.