Assignment 1: Koch curves - Fractal generation using WebGL:

Due Date: October 26th. 10:30.

This assignment will NOT be done in groups. Every individual student should do his/her homework.

Demo will be at class hour (10.30-12.20 October 26th.) on Zoom. More details will be provided later.

Grade Value: 15 %

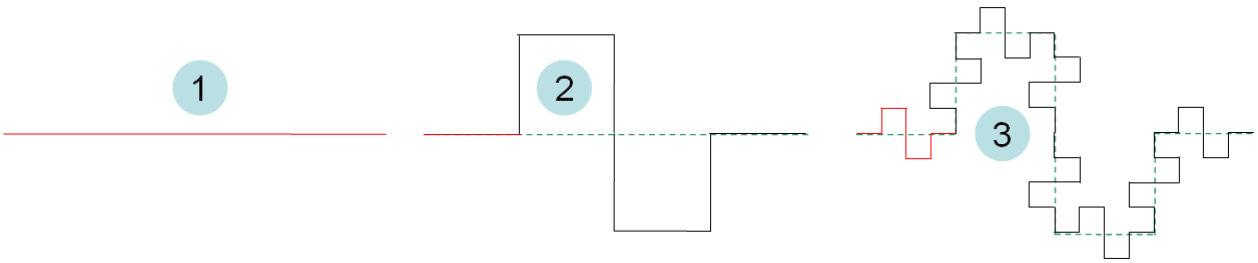
Important Notice

You will demonstrate your programs to Aytek Aman in the scheduled time. Demo details will be announced later. Demonstrations are a kind of exam. You will be asked questions to understand your contribution to the assignments in the demos.

You must also submit your source codes (including online documentation (comments and a README.txt file explaining how it is compiled and run) to the TA. You should email (with the Subject CS465 Assignment #1) to Aytek Aman (aytek.aman@cs) a single zip file named as LastName_Name_CS465_Asst1.zip, containing one directory called Source and a README.txt file describing how to compile and run your assignment.

Requirements

Koch curves are interesting geometrical structures. In the limit, they have infinite length, but they are always within a finite rectangle and they never cross themselves. Consider the rule below which replaces a single line with eight smaller lines.



In this assignment, you are required to write a shader-based WebGL program which will iteratively apply this rule to line segments. Your program should have the following capabilities:

- User should be able to interactively draw an arbitrary convex polygon by clicking with the mouse at vertex locations in order. You can assume that the user will draw a convex polygon.
- User should be able to specify the number of iterations before your program starts applying the rule.
- With a menu selection or keyboard command, the application of the rule starts and final approximate curve is displayed.
- All colors (the background, polygon, and space filling curve) should be changeable.
- There should be an option to fill the polygon.
- Your program should be able to read in and write out text files, so that you can open an existing polygon or Koch curve or save those. You can do polygon saving by the vertex coordinates. Koch curve saving can be done by keeping the number of iterations as well as the vertex coordinates so that

you can re-apply the rule iteratively. You also need to save color values, etc. to re-display the same image.

Important: Always comment your code. The code will also be checked during the demos.

Grading Criteria

- Does your program meet the requirements listed above? Your programs will be graded with respect to the number of the requirements they satisfy.
- You should construct an instructive user interface for your program. The user interface should be easy to use. There should not be any menus without any information. For example, having an empty window without any information but opening a menu when a mouse button is clicked is not a good user interface. The functionality of the user interface is a part of the grading criteria.

Tips

Please make sure your program runs in preparation for the demos. You might want to test your program on a PC to prepare for your demo.

You should use dragging appropriately to draw polygon edges. For example, one edge will be shown after the first click of the mouse button which will follow the mouse cursor and the next mouse click will finalize this edge and start a new one. The polygon will be finalized when user clicks near the first vertex. During this drawing procedure, the edge which is not finalized should also be drawn.

The user should be able to change the color for the lines to be drawn.

To change the brush color, you could use three sliders (easiest), or pick from a few colored patches, or display a color chart and let the user point and pick.

You could use simple HTML UI elements or JavaScript (jQuery, AngularJS, jQWidgets, and so on) UI widget libraries.

There are vast number of WebGL examples on the web. **(Do not use these codes directly)**. If you use some code for user interface generation or some other purpose with modifications, you must state so at the beginning of that code segment (e.g., method header) and properly state how you modified that code segment.

Also DO NOT use the third party wrappers like three.js. Use WebGL from scratch.

Extra

If you complete all of the above, consider zooming in/out with the mouse. An interesting property of space filling curves is that if you adapt the number of iterations to the zoom level an endless repeating detail appears no matter how much you zoom in (See fractal animations for Koch snowflake or Mandelbrot set on web).