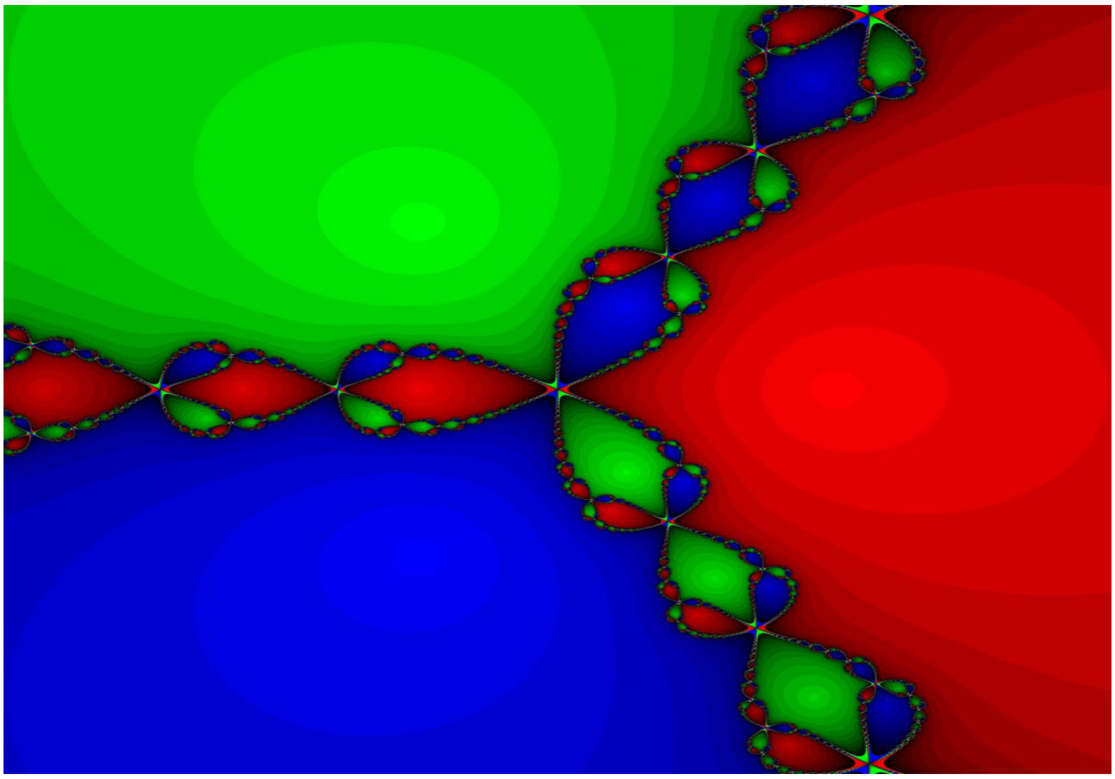
Assignment 1 – Newton fractal



Overview

In this assignment you will construct **a Newton method fractal for cubic equation** on the fragment shader.

- You will upload the shader the parameters it needs as uniform variables.
- In this assignment you will get familiar with the graphics engine of the course.
- You will write a shader in the first time, try to break the implementation to small parts you can check separately.
- Assuming we want to find the roots of $f(z)$ over the complex plane.

$$f(z) = az^3 + bz^2 + cz + d$$

Where $a, b, c\ and\ d$ are real numbers. $z = x + iy$ where $x\ and\ y$ are the coordinates on the screen.

- Upload $a, b, c\ and\ d$ as a **uniform vec4 coeffs** and **IterationNum** as a **uniform int**
- Newton- Raphson method for finding roots:

$$g(z) = z - \frac{f(z)}{f\prime(z)}$$

- Algorithm:

$z_0 = x + iy$

for(int k=0; k < **IterationNum**; k++)

$$z_{k+1} = z_k - \frac{f(z_k)}{f'(z_k)}$$

- We will apply the Newton method for each pixel repeatedly Each pixel on the screen will be colored according to the root newton method converts to (you will have 3 different colors, try to choose colors that are not close to each other).

Key input

Change `key_callback` function in inputManager.h

a. 1,2,3,4 for choosing a, b, c, d coefficient respectively
b. 'up arrow' press state for increasing the coefficient value by 0.01.
c. 'down arrow' press state for decreasing the coefficient value by 0.01
d. ‘left arrow’ press state for increment **iterationNum**
e. ‘right arrow’ press state for decrement **iterationNum** (until a minimum of 1)

Mouse input

Change `mouse_callback, cursor_position_callback, scroll_callback` functions in inputManager.h

The user can move the camera (eye) up, down, left and right when pressing mouse left button. Move the camera closer and farther from the fractal when the user scroll up and down. Print the width of each pixel when zoom in and out.

Some general hints:

- Shading Language summary https://www.youtube.com/watch?v=uOErsQljpHs
- Before you start watch Three blue one brown video about newton fractal: https://www.youtube.com/watch?v=-RdOwhmqP5s
- Before plotting a pixel, make sure that its color does not exceed the range of 0-1 for every color channel (you can use max, min or clamp functions).
- You can debug shaders by coloring pixels differently or you can implement problematic parts in c++ and then copy the implementation to the fragment shader with the relevant changes).
- You can implement the shader on the CPU using glm.h and pass the result as a texture to the shader before you program on the shader.
- You can use texture coordinates to get the **x** and **y** values of the fragment shader.
- If you want to make it nicer you can change the intensity of the pixel color according to the number of steps it takes for the newton method to convert from the pixel.

Submission:

Submission is in pairs.

Submit inputManager.h ,main.cpp, Assignment1.h and Assignment2.cpp. Submit your shaders (also the vertex shader). **Don’t change engine files that don't mention here**. Zip everything to <ID1> _<ID2>.zip file.