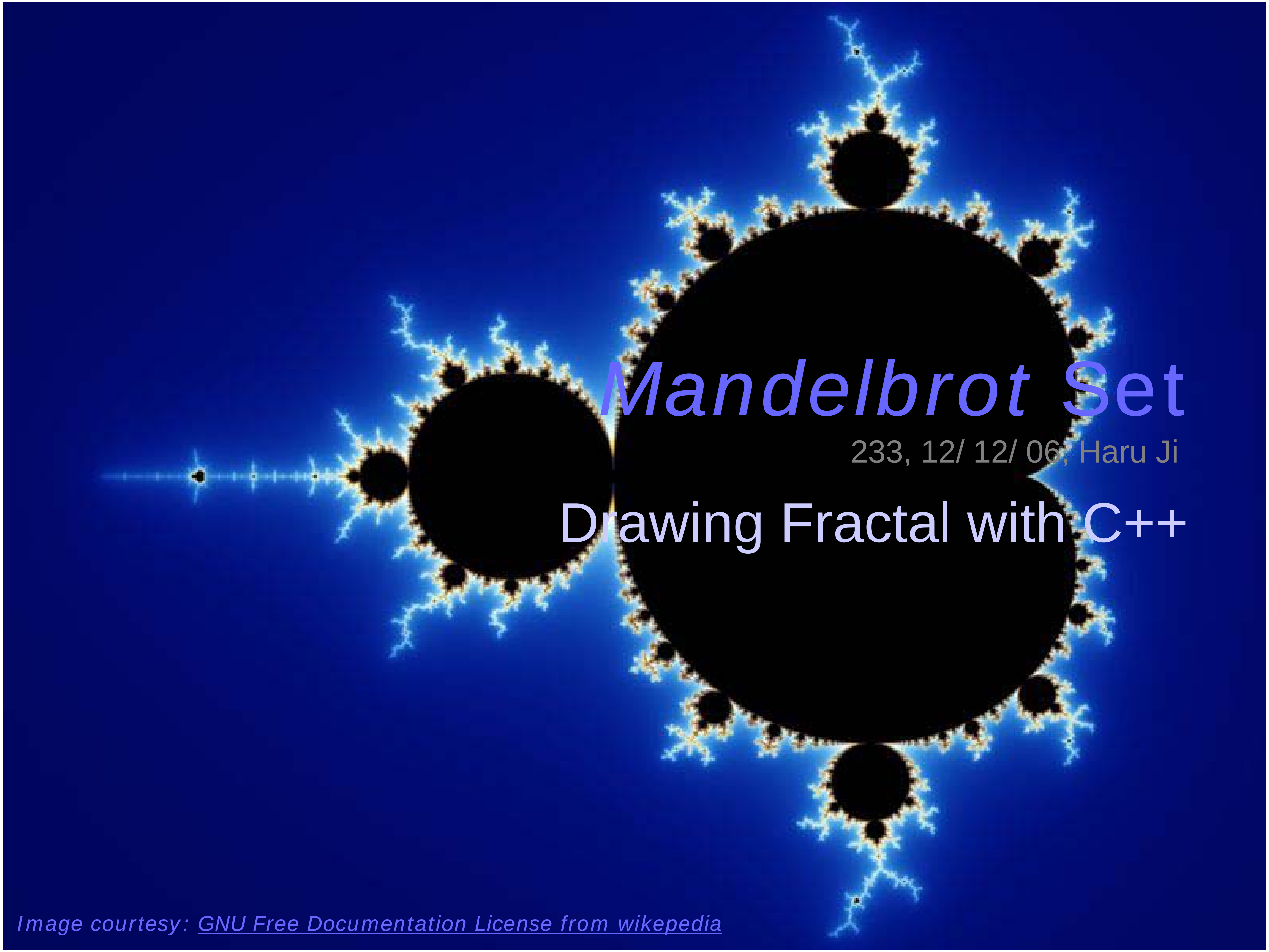


Mandelbrot Set

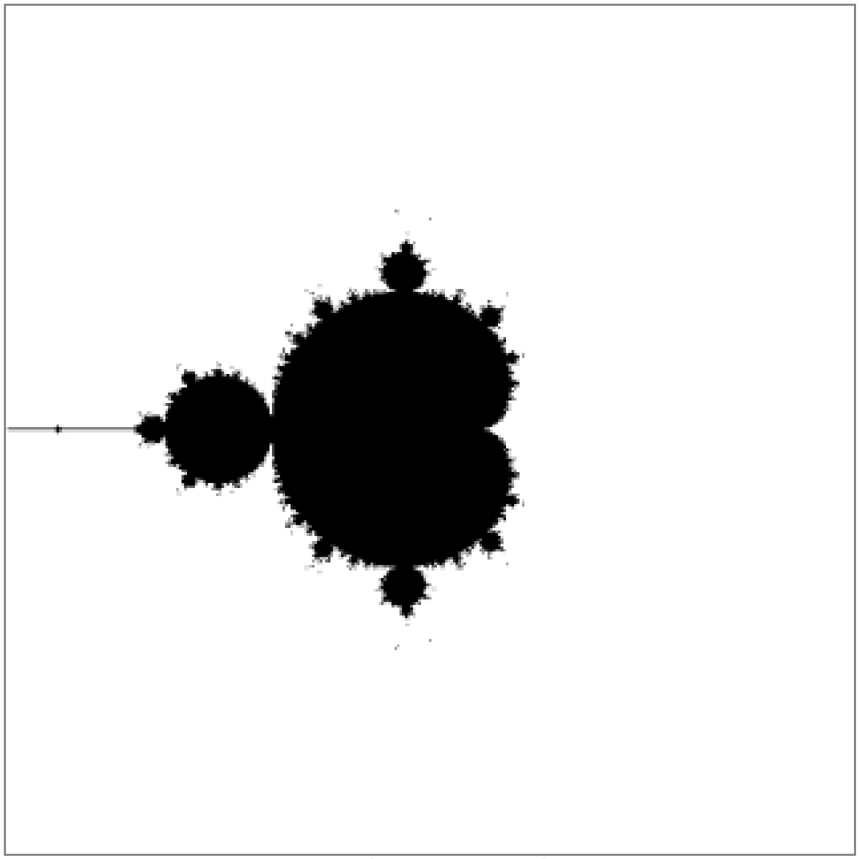
233, 12/ 12/ 06, Haru Ji

Drawing Fractal with C++

Image courtesy: GNU Free Documentation License from wikepedia

Concept : *Approaches to Infinity*

- To visualize infinite patterns such as a fractal of *the Mandelbrot Set*
- My proposal title came from its chapter title of the book titled ‘Computer Graphics using open GL-second edition’, F. S. Hill, JR. chapter 9,



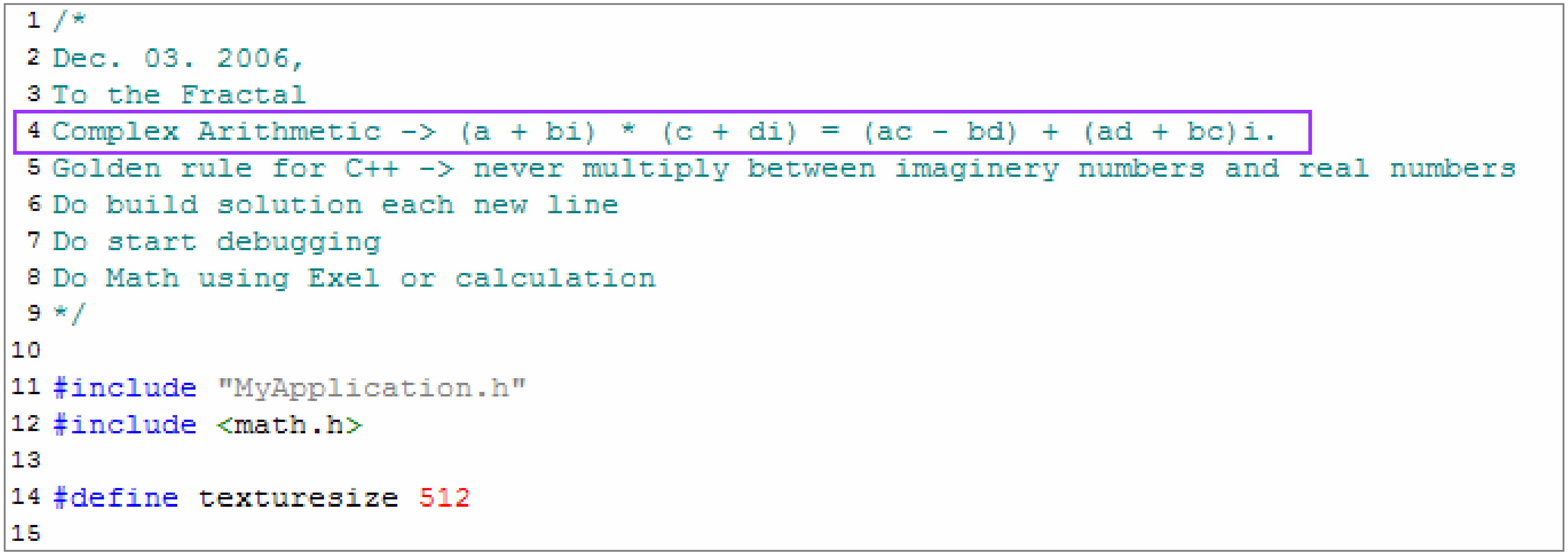
My First Drawing

Technical Approach

- C++ using open GL library
- Research mostly from the internet
- What is a Mandelbrot set?

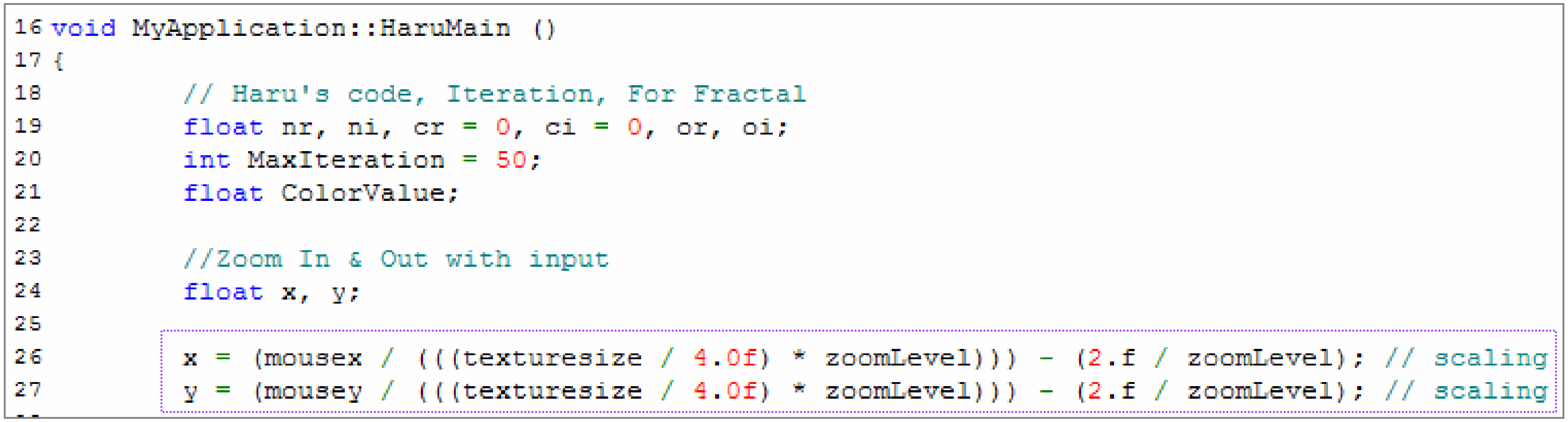
$$f_c : \mathbb{C} \to \mathbb{C}; z \mapsto z^2 + c.$$

C++ Using Open GL _ *Source Code 01*



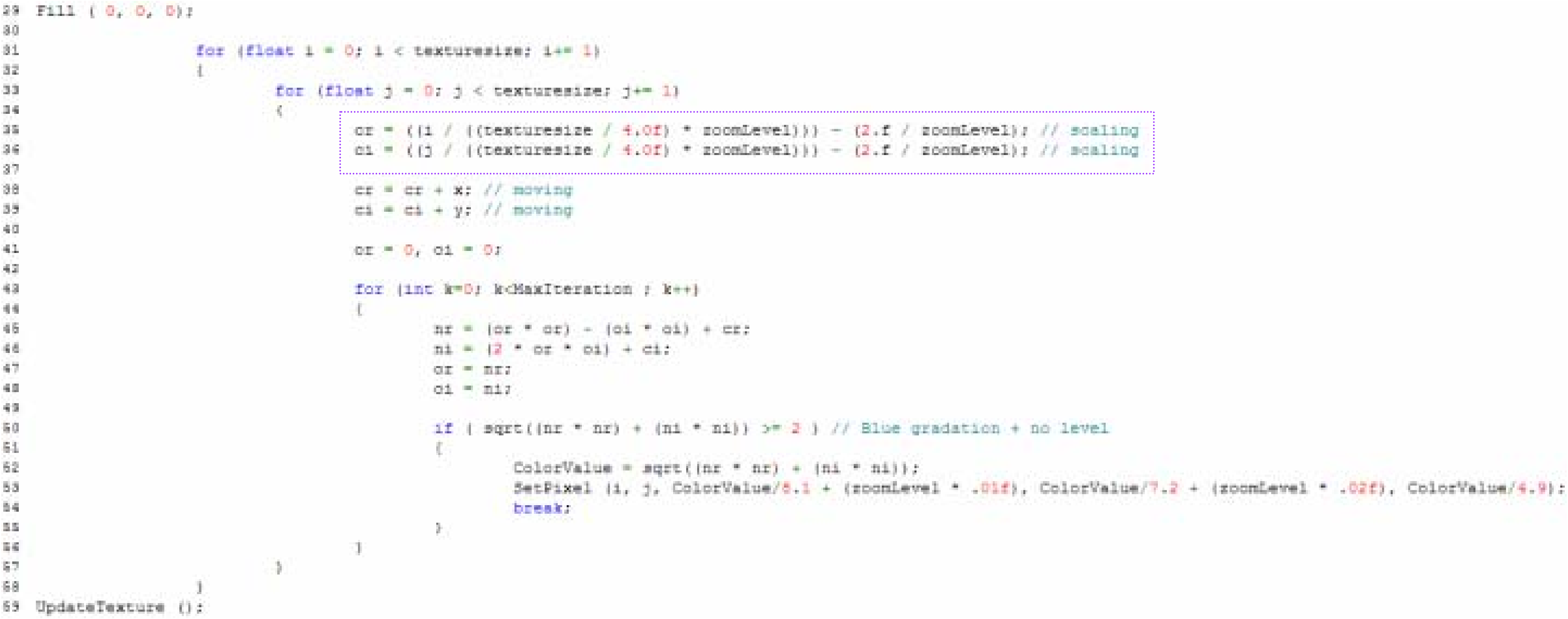
```
/*
Dec. 03. 2006,
To the Fractal
Complex Arithmetic -> (a + bi) * (c + di) = (ac - bd) + (ad + bc)i.
Golden rule for C++ -> never multiply between imaginery numbers and real numbers
Do build solution each new line
Do start debugging
Do Math using Exel or calculation
*/

#include "MyApplication.h"
#include <math.h>

#define texturesize 512

```

C++ Using Open GL _ *Source Code 02*



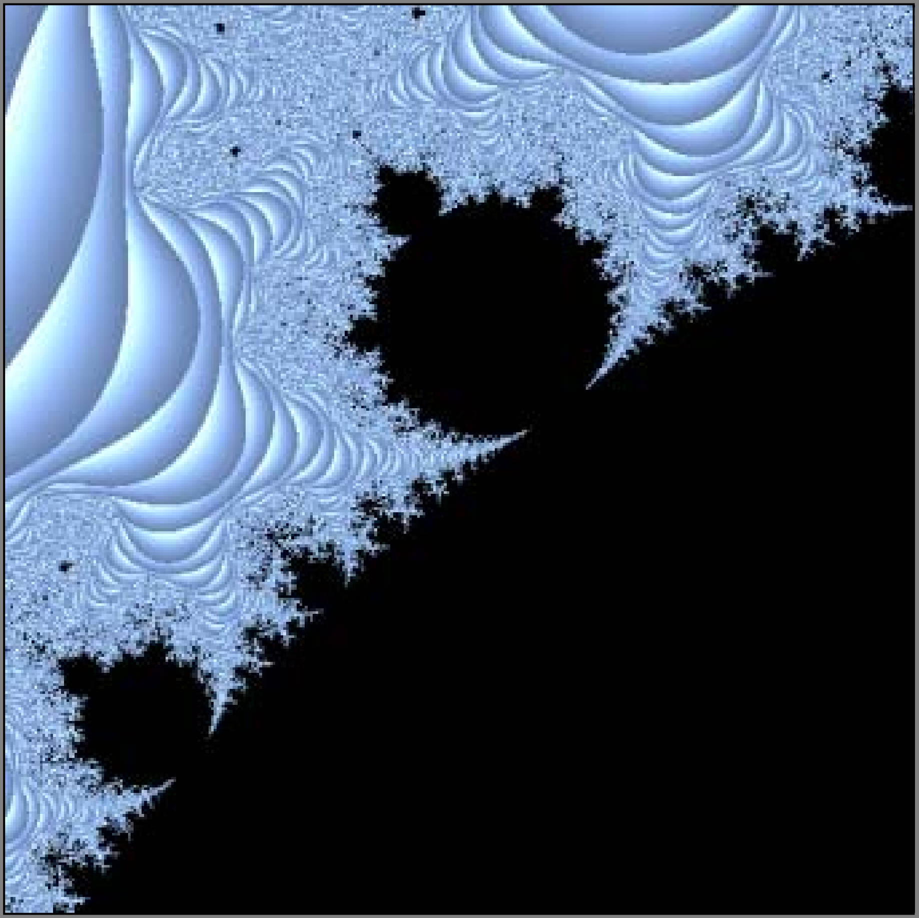
```
void MyApplication::HaruMain ()
{
        // Haru's code, Iteration, For Fractal
        float nr, ni, cr = 0, ci = 0, or, oi;
        int MaxIteration = 50;
        float ColorValue;

        //Zoom In & Out with input
        float x, y;

        x = (mousex / (((texturesize / 4.0f) * zoomLevel))) - (2.f / zoomLevel); // scaling
        y = (mousey / (((texturesize / 4.0f) * zoomLevel))) - (2.f / zoomLevel); // scaling
```

C++ Using Open GL _ *Source Code 03*



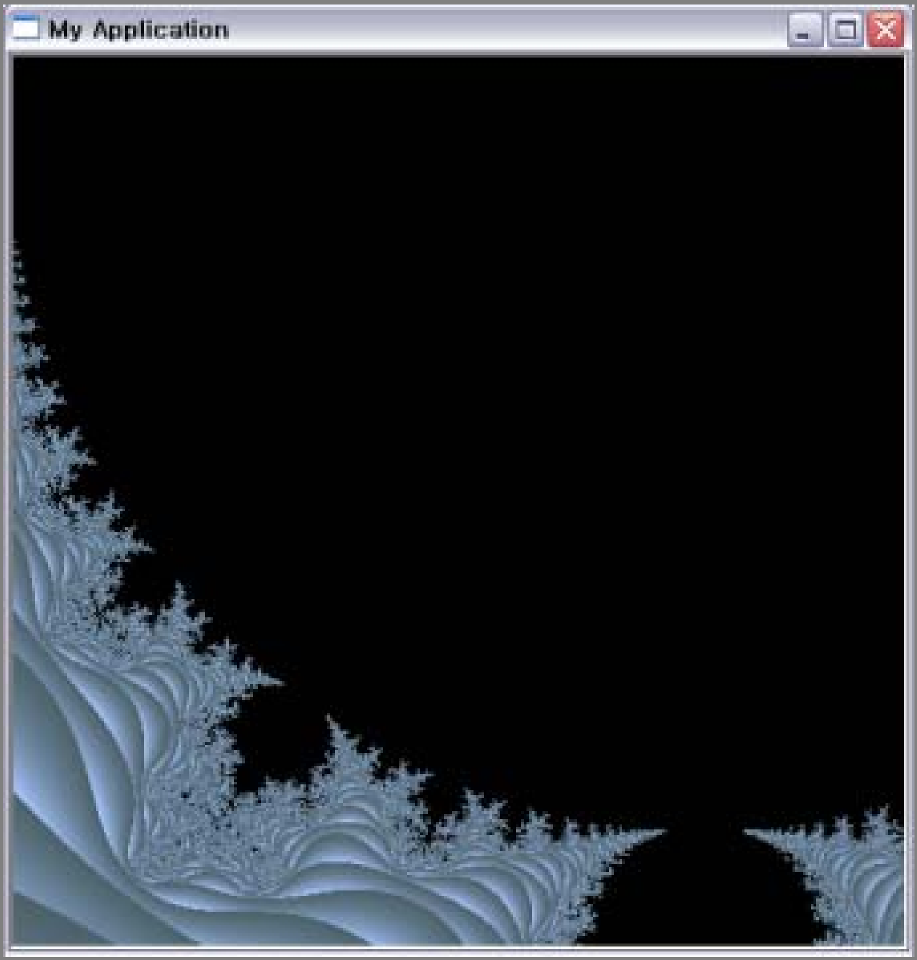
```
Fill ( 0, 0, 0);

        for (float i = 0; i < texturesize; i+= 1)
        {
            for (float j = 0; j < texturesize; j+= 1)
            {
                cr = ((i / ((texturesize / 4.0f) * zoomLevel))) - (2.f / zoomLevel); // scaling
                ci = ((j / ((texturesize / 4.0f) * zoomLevel))) - (2.f / zoomLevel); // scaling

                cr = cr + x; // moving
                ci = ci + y; // moving

                or = 0, oi = 0;

                for (int k=0; k<MaxIteration ; k++)
                {
                    nr = (or * or) - (oi * oi) + cr;
                    ni = (2 * or * oi) + ci;
                    or = nr;
                    oi = ni;

                    if ( sqrt((nr * nr) + (ni * ni)) >= 2 ) // Blue gradation + no level
                    {
                        ColorValue = sqrt((nr * nr) + (ni * ni));
                        SetPixel (i, j, ColorValue/8.1 + (zoomLevel * .01f), ColorValue/7.2 + (zoomLevel * .02f), ColorValue/4.9);
                        break;
                    }
                }
            }
        }
UpdateTexture ();
```

Screenshot _ 01



Max Iteration = 100;

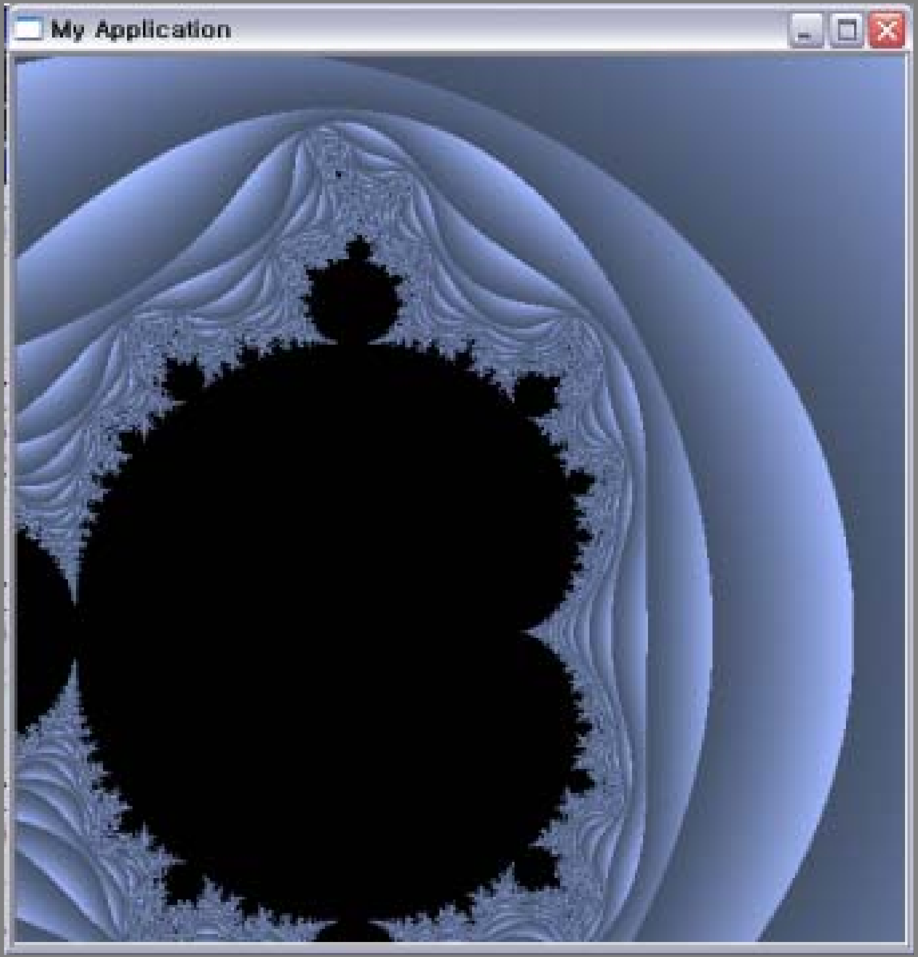
Screenshot _02



Max Iteration = 50;

Zoom In Level = 4;

Screenshot _03



Max Iteration = 50;

Zoom In Level = 2;