

A detailed fractal image of the Mandelbrot set. It features a large, solid black central disk with a highly complex, jagged, and fractal-like boundary. This central disk is surrounded by a series of smaller, similar black disks, each with its own intricate, fractal boundary. The entire structure is set against a deep blue background. The fractal boundaries are highlighted with a bright, glowing yellow and white light, giving it a starry or lightning-like appearance. The overall shape is roughly circular but with many protrusions and indentations.

Mandelbrot Set

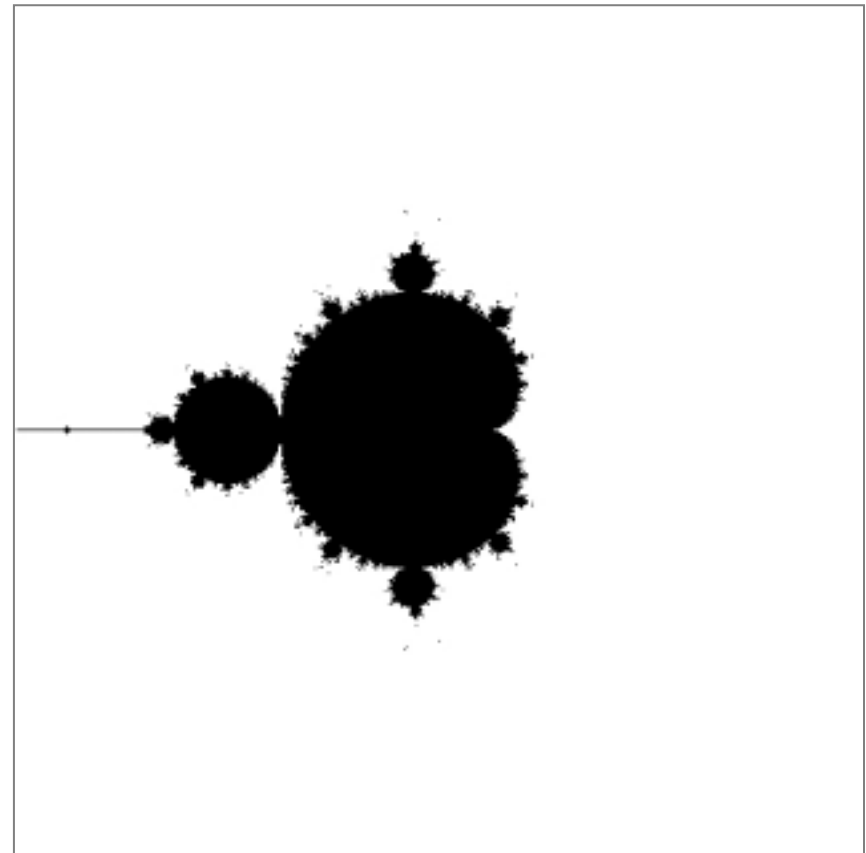
233, 12/ 12/ 06, Haru Ji

Drawing Fractal with C++

Image courtesy: [GNU Free Documentation License from wikipedia](#)

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?

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

C++ Using Open GL *_Source Code 01*

```
1 /*
2 Dec. 03. 2006,
3 To the Fractal
4 Complex Arithmetic ->  $(a + bi) * (c + di) = (ac - bd) + (ad + bc)i$ .
5 Golden rule for C++ -> never multiply between imaginary numbers and real numbers
6 Do build solution each new line
7 Do start debugging
8 Do Math using Exel or calculation
9 */
10
11 #include "MyApplication.h"
12 #include <math.h>
13
14 #define texturesize 512
15
```

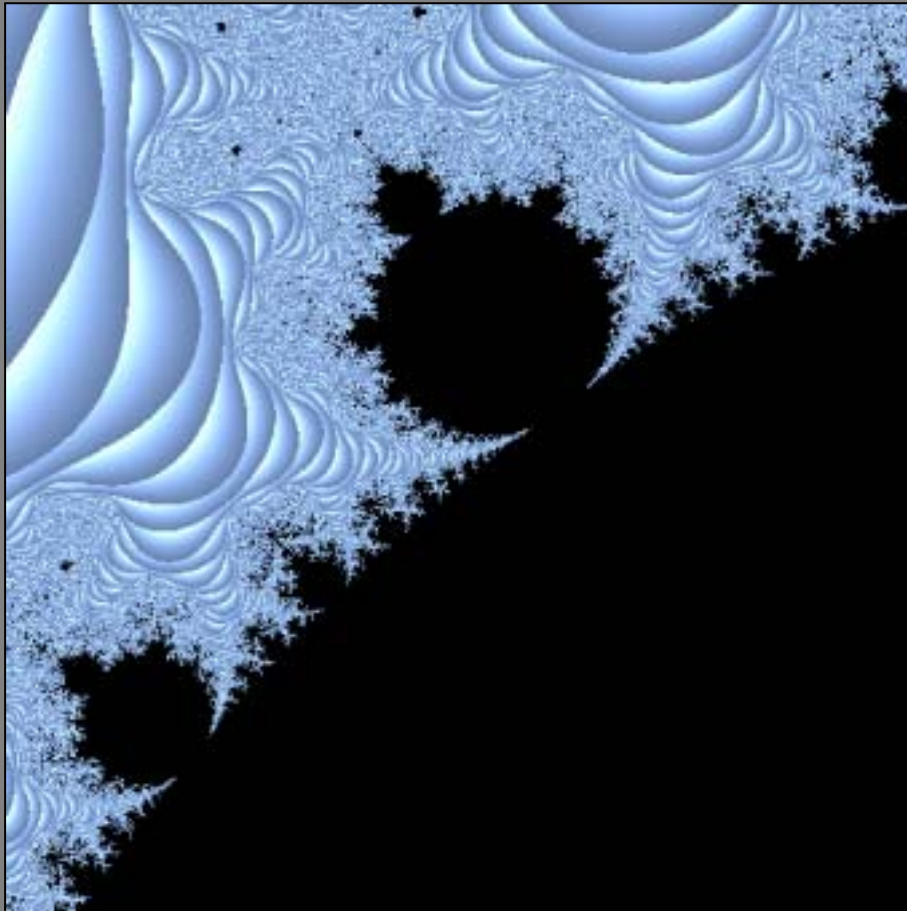
C++ Using Open GL *_Source Code 02*

```
16 void MyApplication::HaruMain ()
17 {
18     // Haru's code, Iteration, For Fractal
19     float nr, ni, cr = 0, ci = 0, or, oi;
20     int MaxIteration = 50;
21     float ColorValue;
22
23     //Zoom In & Out with input
24     float x, y;
25
26     x = (mousex / (((texturesize / 4.0f) * zoomLevel))) - (2.f / zoomLevel); // scaling
27     y = (mousey / (((texturesize / 4.0f) * zoomLevel))) - (2.f / zoomLevel); // scaling
28 }
```

C++ Using Open GL *_Source Code 03*

```
29 Fill ( 0, 0, 0);
30
31 for (float i = 0; i < textureSize; i+= 1)
32 {
33     for (float j = 0; j < textureSize; j+= 1)
34     {
35         cr = ((i / ((textureSize / 4.0f) * zoomLevel))) - (2.f / zoomLevel); // scaling
36         ci = ((j / ((textureSize / 4.0f) * zoomLevel))) - (2.f / zoomLevel); // scaling
37
38         cr = cr + x; // moving
39         ci = ci + y; // moving
40
41         cr = 0, ci = 0;
42
43         for (int k=0; k<MaxIteration ; k++)
44         {
45             nr = (cr * cr) - (ci * ci) + cr;
46             ni = (2 * cr * ci) + ci;
47             or = nr;
48             oi = ni;
49
50             if ( sqrt((nr * nr) + (ni * ni)) >= 2 ) // Blue gradation + no level
51             {
52                 ColorValue = sqrt((nr * nr) + (ni * ni));
53                 SetPixel (i, j, ColorValue/8.1 + (zoomLevel * .01f), ColorValue/7.2 + (zoomLevel * .02f), ColorValue/4.9);
54                 break;
55             }
56         }
57     }
58 }
59 UpdateTexture ();
```

Screenshot_01



Max Iteration = 100;

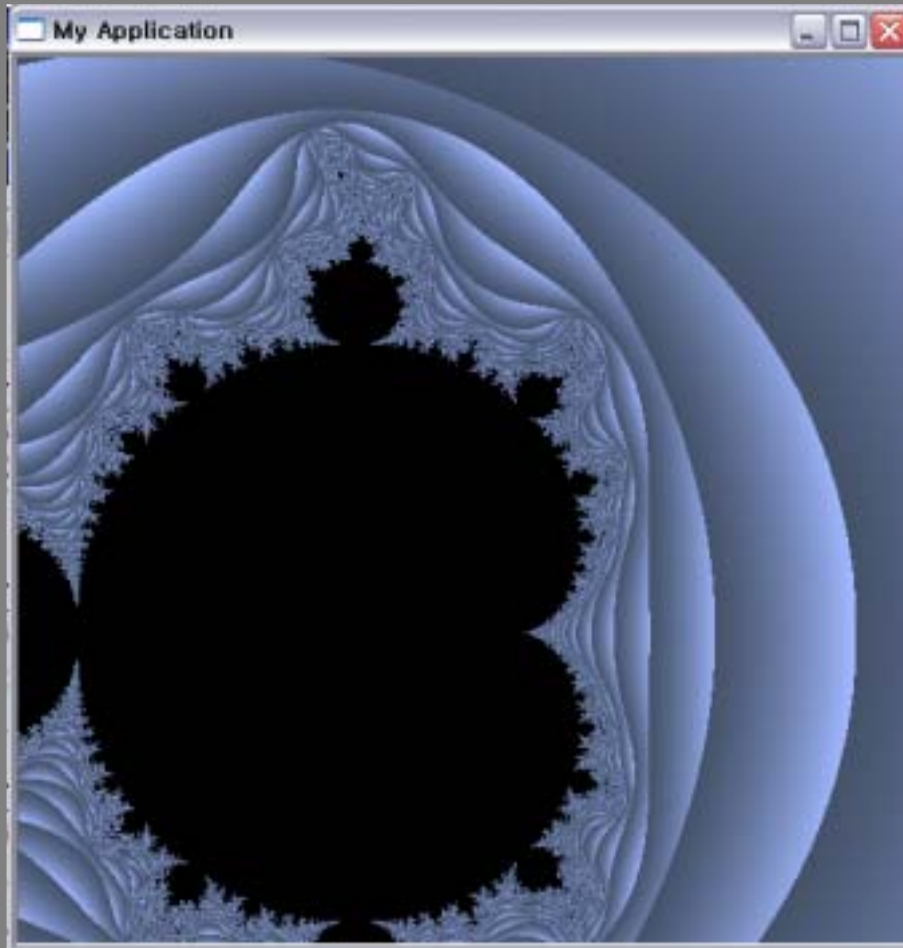
Screenshot_02



Max Iteration = 50;

Zoom In Level = 4;

Screenshot_03



Max Iteration = 50;
Zoom In Level = 2;