GYMNASIUM JANA KEPLERA

Parléřova 2/118, 169 00 Praha 6

Visualizing The Relationship Between Quadratic Julia Sets And Quadratic Mandelbrot Sets

High school graduation thesis

Author: Mikuláš Vanoušek

Class: 4.A

School year: 2020/2021

Subject: Informatics

Thesis supervisor: Šimon Schierreich

Prague, 2021

GYMNASIUM JANA KEPLERA
Kabinet informatiky

ZADÁNÍ MATURITNÍ PRÁCE

Student:	**Mikuláš Vanoušek**
Třída:	**4.A**
Školní rok:	**2020/2021**
Platnost zadání:	**27.10.2020**
Vedoucí práce:	**Šimon Schierreich**
Název práce:	**Třídimenzionální průřezy čtyřdimenzionálního fraktálu.**

Pokyny pro vypracování:

Cílem práce je vytvořit nativní aplikaci pro operační systém Linux, která umožní rendrovat 3-dimenzionální průřez 4-dimenzionálního fraktálu. Uživatel bude mít možnost pohybovat se ve všech čtyřech dimenzích. Také bude mít možnost fraktál „otočit" a změnit dimenzi, podél které jsou průřezy vedeny a která je tím pádem skrytá.

Doporučená literatura:

[1] BRENNAN, Kyle J., ed. Handbook on the Classification and Application of Fractals. London: Nova Science, 2012. ISBN 978-1613241981.

[2] MARTIN, Robert C. Design Principles and Design Patterns. www.objectmentor.com, 2000. Dostupné z: https://fi.ort.edu.uy/innovaportal/file/2032/1/design_principles.pdf.

[3] FOWLER, Martin. Patterns of enterprise application architecture. Boston: Addison-Wesley Professional, 2003. ISBN 978-0321127426.

URL repozitáře:

https://github.com/MikulasVanousek/4D-fractal

vedoucí práce *student*

V Praze dne 29. 10. 2020

Authorship Declaration

I hereby declare that the thesis submitted is my unaided work. All direct or indirect sources used are acknowledged as references. I am aware that the thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the thesis as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources, I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future theses submitted. Further rights of reproduction and usage, however, are not granted here. This paper was not previously presented to another examination board and has not been published.

In Prague, April 7, 2021 Mikuláš Vanoušek

Acknowledgements

I would hereby like to thank my thesis advisor Šimon Schierreich for all the advice he has provided. I would like to thank Jaroslav Žukov for his help. Last but not least I would like to express immense gratitude towards Pavel Vanoušek and Iva Vanoušková, who had given me the expensive gift of life and provided me with the environment and support that has allowed me to write this thesis.

Abstrakt

Kvadratické Julia sety a kvadratické Mandelbrot sety jsou na sebe kolmé průřezy funkce s dvěma komplexními parametry. Představit si, jak spolu souvisí může být velmi obtížné. Má práce si klade za úkol vizualizovat jejich vztah.

Klíčová slova

Mandelbrot set, Julia set, Komplexní analýza, OpenGL, vykreslování fraktálů

Abstract

Quadratic Julia sets and quadratic Mandelbrot sets are orthogonal cross-sections of one function with two complex parameters. It can be hard to grasp how exactly those two are related. My thesis aims to visualize the connection and help people understand how Julia sets and Mandelbrot sets are related.

Keywords

Mandelbrot set, Julia set, Complex Analysis, OpenGL, Fractal rendering

Contents

1. Theoretical part

In the first chapter of my thesis, I will lay down the theoretical foundations of my thesis. I will explain what Mandelbrot sets and Julia sets are, how we can visualize them, how they are part of one 4-dimensional object and how we can visualize this relationship.

1.1 Complex analysis

Complex analysis investigates the behavior of functions with complex parameters and complex values. The function in question is:

$$f(z, c) = z^n + c$$

$$z,\ c,\ f(z,c) \in \mathbb{C}$$

For given z, c we study the behaviour of this function, as it is iterated infinitely many times.

$$z_0 = z$$

$$z_{n+1} = f(z_n, c)$$

$$F(z, c) = z_\infty = \lim_{n\to\infty} z_n$$

For some z, c the value of $F(z, c)$ is bounded.

$$F(z,c) < k$$

$$k \in \mathbb{R}$$

For other z, c the value of $F(z, c)$ is unbounded - it escapes to infinity.

$$F(z,c) = \infty$$

For unbounded $F(z,c)$ we measure how fast they are growing with the function $g(z, c) \in [0; 1]$. [3]

1.2 Approximating $F(z, c)$

A way of computing exact value of $F(z, c)$ for arbitrary z,c has not been found. We have to approximate it. We choose a number of maximum iterations I_m and some escape radius $r \gg 0$. Then we iterate the function $f(z,c)$ i-times, until one of these conditions is met:

- $i = I_m$ - we approximate $F(z,c)$ to be bounded.
- $|z_i| > r$ - we approximate $F(z,c)$ to be unbounded and $g(z, c)$ is approximated some function $g'(i, I_m)$.

1.3 Visualizing complex functions

On an image with resolution R_x, R_Y, we can associate a pixel with n pixels before it in the horizontal direction and m pixels before it in the vertical direction (origin on the bottom left) with a pair of Cartesian coordinates X, Y.

$$X = 2n/R_x - 1$$

$$Y = 2m/R_y - 1$$

Here, $[0;0]$ is in the middle of the screen, $[-1;-1]$ is in the bottom left corner and $[1;1]$ is the top right corner. If we want to visualize different parts of the function, we can create adjusted coordinates of a pixel x,y by multiplying by scale s and adding offsets o_x, o_y.

$$x = o_x + X/s$$

$$y = o_y + Y/s$$

We can associate each pixel with a complex number p.

$$p = [x,y] = x + iy$$

We can then visualize a complex function $h(a)$; $a, h(a) \in \mathbb{C}$ by creating an image and coloring each pixel based on the value of $h(p)$.

1.4 Quadratic Mandelbrot sets

Quadratic Mandelbrot set M_z is the set of all c, for which $F(z, c)$ is bounded, z is held constant. [5] We can visualize an M_z by creating an image and coloring each pixel black if its associated complex number p belongs to the set ($F(z, p)$ is bounded), white if it does not ($F(z,p)$ is unbounded) - see figure 1.1.

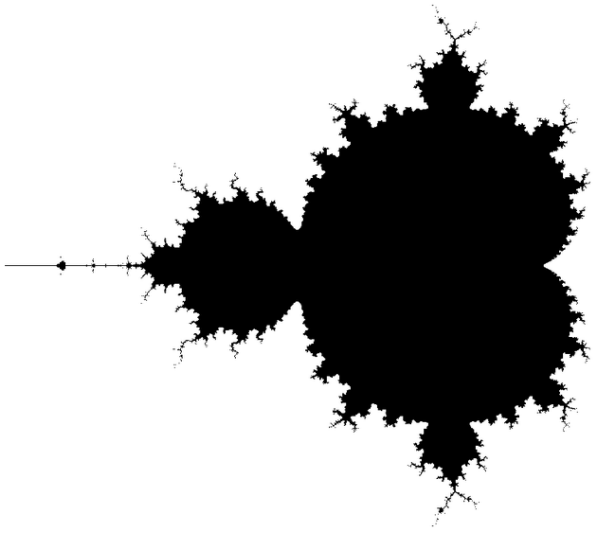

Figure 1.1: Two color version of the quadratic Mandelbrot set $z = 0$. [11]

This method can be improved by coloring each pixel, whose p is not present in the set using the value of $g(p, c)$. This will uncover a great amount of detail and expose the amazing complexity of the Mandelbrot set - see figure 1.2.

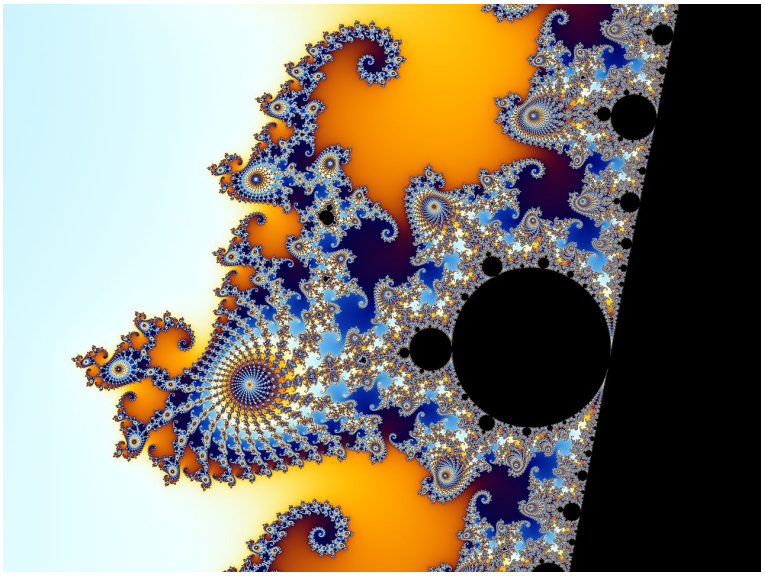

Figure 1.2: M_0 colored based on the value of $g(p,c)$. [2]

1.5 Julia Sets Of Quadratic Polynomials

Julia set of a quadratic polynomial J_c is the set of all z for which $F(z, c)$ is bounded.[4][9] We can visualize J_c analogously to Mandelbrot set, we just use the complex number p associated with the pixel as the fist argument - $F(c, p)$.

1.6 The Connection

Since the function $F(z, c)$ has 2 complex parameters ($z = [z_a, z_b], c = [c_a, c_b]$), we can associate each point in 4-dimensional space $P[z_a, z_b, c_a, c_b]$ with a value of $F(z, c)$. Both M_z and J_c are then cross sections of this object. M_z is parallel to the two z axes and J_c to the two c axes. We can, however also plot cross sections of this space that are not parallel to the axes. Our plane is rotated from z_a to c_a with the angel α and from z_b to c_b with the angel β.

$$F_r(x, y, \alpha, \beta) = F(\cos(\alpha) \cdot x, \cos(\beta) \cdot y, \sin(\alpha) \cdot x, \sin(\beta) \cdot y,$$

2. Implementation

In the second chapter of my Thesis, I will write about the decisions I had to make during the development, why I made the ones I made, If I think they were the right ones and how I would approach the project with the benefit of hindsight.

2.1 2-Dimensional slices

Shortly after turning in the official assignment worksheet, I realized making the slices 3-dimensional was not a good idea. Making 3-dimensional slices of a 4-dimensional object is not more difficult than making 2-dimensional slices. The problem is with rendering them. All of the 3-dimensional slices would have no outside border, many of them would have no inside border either. On top of that, the object I was rendering had a lot of inner complexity, and to expose it, the slices would have to be 2-dimensional.

2.2 Unity Post-Processing Shader

Since this task is computationally difficult - the function f is evaluated up to $R_x \cdot R_y \cdot I_m$ times. I needed to leverage GPU acceleration in order to run the rendering program in real-time. I had some very limited experience with the game engine Unity and I knew it featured shaders - programs that run parallelly on the many GPU cores for every pixel on the screen - so I decided to use it. I rendered an empty scene and added the fractal as a post-processing shader written in High-Level Shading Language. It worked, but it was not an ideal way to approach the problem. Unity post-processing shaders are meant for a very specific use case and were not build to be used this way.

2.3 OpenGL Rewrite

After the first consultation, I decided to rewrite the whole thing using C++ and OpenGL. I was not satisfied with the Unity HLSL documentation and I felt like OpenGL would provide me with more flexibility. While OpenGL has turned out to be more difficult to use than I anticipated, I still believe this was the right decision.

2.4 Double-Shader Architecture

To show more detail, I wanted the fractal to use the whole color spectrum, even if all the values visible on the screen were very close together. But since the whole point of using a shader is parallelization, it was impossible to know other values on the screen before the whole shader program has finished. And so I had to convert my program architecture into a more complex one. First, in the compute shader I calculate the value of $g(z,c)$ for every pixel on the screen. At this step, I also

save the lowest and highest values recorded and save it together with all the values in a Buffer on the GPU. Then in the fragment shader, I color each pixel based on where its associated value lays in the interval from the lowest value to the highest one. I do not believe this would be possible if I carried on with the Unity approach.

2.5 Performance Issues

The OpenGL application runs very slowly on my laptop, which is not surprising - my laptop has no dedicated GPU. But it runs even slower on my PC with GTX 1650. I have spent many hours profiling, debugging, researching, and even posted a question to Stack Overflow. I tried rewriting the application to use different buffers for writing and reading values. In the end, I was not able to improve the performance.

2.6 Mysterious Bugs

In the development of the application, I have encountered some bugs I just could not wrap my mind around. As an example, I noticed that after moving around in the fractal and stopping, the fractal colors were shivering for a while, then settled. I checked and double-checked every variable involved in the equation that determined the color of every pixel - none of them changed after I stopped touching the keyboard. Yet the colors on the screen changed. After a few hours, I just gave up. I accepted the fractal will keep on shivering from time to time. A week later, I noticed the same shivering when I opened a picture on my computer. The monitor was adjusting it is dynamic range - slowly so it would not be as noticeable.

2.7 The Benefit Of Hindsight

I have learned a lot in the development process. If you divide the time I have spent on the project by the number of lines of code I have produced, it is by far the most time-dense project of mine. If I were to take on the project today, with the experience I gained in the process, I would approach it very differently. I would choose to create an application that does not run in real-time, but instead saves the rendered images and creates a video of the movement through the fractal. This would be possible to do on the CPU. I would not have to spend so much time navigating technologies I was not at all familiar with and would allow me to invest more time into developing additional features.

3. Technical Documentation

In the last chapter of my thesis, I will provide information that will allow you to run and use the application.

3.1 Running The Application

First, download the code. Then extract the archive contents. The archive contains a 4D-fractal-master folder - inside you will find the source code.

3.1.1 Windows

Inside 4D-fractal-master/Windows/Release you will find fractal-cpp.exe. You should be able to run it - if not, see section 3.1.3.

3.1.2 Linux And Mac OS

On operating systems other than Windows, you will have to compile the application yourself. This can be a bit tricky to do, so I will provide you with some guidance.

- You might be able to import the solution into visual studio. If you do, you should be able to compile it and run the app that will be created in the Release folder. (You might still have to copy the shader files.)
- If you are not able to import the project, you can create a project from existing code and link all the libraries necessary - GLEW (static version), GLFW, linmath. To link the libraries, go to project properties and do the following:
  - Add all the include directories.
  - Add all the additional library directories (32-bit versions).
  - Add all the additional dependencies.
  - Add GLEW_STATIC preprocessor definition.
  - For more details see [8].
- You also need to make sure the shader files (vs.shader, cs.shader, fr.shader) will be copied to the output directory. This can be done by setting up the IDE to do this for you when the project is compiled or you can do this manually.

3.1.3 Installing OpenGL Driver

For the application to run, you need to have OpenGL 4.5.0 or a newer driver installed on your GPU. If your GPU driver is up to date, OpenGL 4.5.0 should be included. The version of OpenGL installed can be found in the console output of the application. If your version is lower than 4.5.0 or no OpenGL driver is installed, download the newest driver from the website of your GPU manufacturer and follow their installment instructions.

3.2 Controls

The controls work best with QWERTY keyboard layout.

- Use D (+) A (-) to mover your view along x - the horizontal axis of the slice.
- Use W (+) S (-) to mover your view along y - the vertical axis of the slice.
- Use R (+) F (-) to move along a - the first perpendicular axis to the slice.
- Use T (+) G (-) to move along b - the second perpendicular axis to the screen.
- Use Y (+) H (-) to rotate the slice $\frac{\pi}{2}$ along the y and b axes - y and b still point in the same direction.
- Use U (+) J (-) to rotate the slice $\frac{\pi}{2}$ along the x and a axes - x and a still point in the same direction.
- Use L-Shift (+) L-Ctrl (-) to control zoom.
- Use R-Shift ($\times 2$) R-Ctrl ($\times \frac{1}{2}$) to control the maximum number of iterations.
- Use Esc to exit the application.

Conclusion

The original goal of my work was to create a Linux application visualizing 3-d slices of the fractal. Unfortunately, I did not do enough research before writing the specification. I chose a very difficult assignment and therefore even in the midst of my work, I was not sure how it is going to turn out. Due to the severe lack of experience in the field, I had to pivot many times during the development process, which prevented me from fulfilling some of the originally laid out goals. Visualizing 3-dimensional slices turned out not only impractical but borderline impossible. The application was originally meant to be for Linux. When I replaced Ubuntu with Windows on my laptop, I relied on the Unity cross-platform compilation. After the OpenGL rewrite, I lost this option - the application is for Windows only.

Nevertheless, I believe I succeeded in my goal to visualize the relationship between quadratic Mandelbrot sets and quadratic Julia sets. Rotating the slice of the 4-dimensional function allows the user to see other slices of the fractal than the ones parallel to the axes. This gives the user much more freedom in the exploration of the 4-dimensional object and it allows him to understand how quadratic Mandelbrot sets and quadratic Julia sets are the object's orthogonal slices.

I gained a lot of experience working on this thesis. I realized how important the initial choice of technologies is and how mistakes in the onset of a project caused by the lack of expertise can be very costly. I also experienced firsthand the complexity associated with programming in the lower abstraction levels and working directly with the hardware. This helped me narrow down the domain of IT I want to specialize in and identify some areas of IT I will try to avoid in the future.

The performance issues, while agonizing for me, luckily are not severe enough to prevent the user from gaining the insight this thesis is supposed to provide. If I were to take on the project today, with the experience I gained in the process, I would approach it very differently - instead of creating a real-time application running on the GPU, I would use the CPU to render a 60 FPS video of the movement through the fractal. I would not have to spend so much time navigating technologies I was not at all familiar with, which would allow me to invest more time into developing additional features.

Citations

[1] Anonymous. *GLFW Documentation*. Accessed: 23.03.2021. URL: https://www.glfw.org/documentation.

[2] Wolfgang Beyer. *Figure 1.2*. Accessed: 23.03.2021. URL: https://commons.wikimedia.org/wiki/File:Mandel_zoom_03_seehorse.jpg.

[3] Wikipedia The Free Encyclopedia. *Complex analysis*. Accessed: 23.03.2021. URL: https://en.wikipedia.org/wiki/Complex_analysis.

[4] Wikipedia The Free Encyclopedia. *Julia set*. Accessed: 23.03.2021. URL: https://en.wikipedia.org/wiki/Juila_set.

[5] Wikipedia The Free Encyclopedia. *Mandelbrot set*. Accessed: 23.03.2021. URL: https://en.wikipedia.org/wiki/Mandelbrot_set.

[6] Wikipedia The Free Encyclopedia. *Plotting algorithms for the Mandelbrot set*. Accessed: 23.03.2021. URL: https://en.wikipedia.org/wiki/Plotting_algorithms_for_the_Mandelbrot_set#Escape_time_algorithm.

[7] Anton Gerdelan. *"Hello Triangle" OpenGL 4 Up and Running*. Accessed: 23.03.2021. URL: https://antongerdelan.net/opengl/hellotriangle.html.

[8] The Cherno. *OpenGL playlist*. Accessed: 23.03.2021. URL: https://www.youtube.com/playlist?list=PLlrATfBNZ98foTJPJ_Ev03o2oq3-GGOS2.

[9] Wolfram Mathworld. *Julia set*. Accessed: 23.03.2021. URL: https://mathworld.wolfram.com/JuliaSet.html.

[10] MorganS42, Hovercraft Full Of Eels, and Spektre. *Can't find a way to color the Mandelbrot-set the way i'm aiming for*. Accessed: 23.03.2021. URL: https://stackoverflow.com/questions/56195735/cant-find-a-way-to-color-the-mandelbrot-set-the-way-im-aiming-for.

[11] Protious. *Figure 1.1*. Accessed: 23.03.2021. URL: https://en.wikipedia.org/wiki/File:Mandelbrot_black_itr20.png.

[12] user8108238 and Spektre. *Mandelbrot Set - Color Spectrum Suggestions?* Accessed: 23.03.2021. URL: https://stackoverflow.com/questions/53658296/mandelbrot-set-color-spectrum-suggestions/53666890#53666890.

[13] Mikuláš Vanoušek. *How to write an OpenGL fragment shader in two stages?* Accessed: 23.03.2021. URL: https://stackoverflow.com/questions/66430125/how-to-write-an-opengl-fragment-shader-in-two-stages/66598188#66598188.

List of Figures