CS 184 Final Report: Team Shaders

ARTHUR POMMERSHEIM, NICOLAS VEGA, TONY LI, and MYA THANEGI SOE

Fig. 1. Team Shader Logo

This project involves creating a new collection of GLSL fractal shaders. It highlights the vast potential of shader programming, illustrating how intricate visuals can be generated from simple geometric primitives.

Additional Key Words and Phrases: shaders

ACM Reference Format:
Arthur Pommersheim, Nicolas Vega, Tony Li, and Mya Thanegi Soe. 2024. CS 184 Final Report: Team Shaders. 1, 1 (May 2024), 10 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Shaders provide a method for rendering realistic and captivating imagery. Shaders have continued to evolve, starting as simple programs for rudimentary rendering to complex porgrams that can render and simulate physical phenomena. This project seeks to explore the capabilities of shaders.

Authors' address: Arthur Pommersheim, arthurpomm@berkeley.edu; Nicolas Vega, nvegab99@berkeley.edu; Tony Li, litony396@berkeley.edu; Mya Thanegi Soe, myasoe@berkeley.edu.

Manuscript submitted to ACM

By adjusting geometric properties, surface materials, and lighting techniques, our team has successfully developed a variety of visually distinct fractal shaders. Initially, our shaders were designed to render basic geometries using flat colors and straightforward lighting. Building on this foundation, we have expanded our scope to incorporate complex elements like the Mandelbulb set, employing advanced concepts such as HSV color models and Blinn-Phong lighting. This progression has allowed us to achieve richer, more intricate visual outputs.

2 TECHNICAL APPROACH

2.1 Mandelbrot and Mandelbulb

The Mandelbrot set, named after mathematician Benoit Mandelbrot, is a set of points in the complex plane, the boundary of which forms a fractal. Formally, it is defined as the set of complex numbers c for which the function

$$z_{i+1} = z_i^2 + c, \quad z_0 = 0$$

does not diverge when iterated from $z = 0$ i.e. the sequence remains bounded in absolute value. In order to render, we set c as the position of the screen where the x-axis is treated as the real component and y-axis is treated as the imaginary component of the complex number and do a set number of iterations, stopping early if the equation diverges. The color is determined by taking a linear interpolation between two colors using the ratio of the number of iterations done to the max number of iterations. The beauty of the Mandelbrot set lies in its intricate boundary that reveals progressively finer recursive detail at increasing magnifications.

This Mandelbrot function can be extended into 3D to create the Mandelbulb by using Nylander's formula, which gives the nth power for a 3D point. This formula first converts the 3D point to spherical coordinates $[(x, y, z) \rightarrow (r, \theta, \phi)]$. Then, to take the nth power of the point, you calculate:

$$z^n = r^n \begin{bmatrix} \sin(n\theta)\cos(n\phi) \\ \sin(n\theta)\sin(n\phi) \\ \cos(n\theta) \end{bmatrix}$$

This method of applying an exponent is used the same way as in the Mandelbrot function, with the power n being varied to give different order Mandelbulb renders.

2.2 Raymarching

In order to render the Mandelbulb in 3D space in a time-efficient manner, the technique of raymarching is used. The basic setup is the same as standard raytracing, where a ray is sent out from the camera in a certain direction, looking for intersections with objects in the scene. Raymarching modifies the ray intersection process by instead following the path of the ray guided by the distance between the closest object in the scene and the current position along the ray.

Specifically, this distance is calculated using a signed distance function (SDF) which is made for each object in the scene. If this distance is less than a chosen small threshold, then this is counted as an intersection. If this distance is greater than a sufficiently large, chosen threshold, then the ray position is considered to be too far from any of the objects in the scene and counts as a ray miss. Otherwise, the current position along the ray is incremented by traveling along the ray the same amount as the distance given by the SDF and this process is repeated.

2.2.1 Mandelbulb SDF. In order to calculate the distance from the Mandelbulb given a 3D point, we first do a set number of iterations of the Mandelbrot function using the methods described earlier, ending iteration early if the point

diverges. During each iteration, we keep track of a running derivative of the mandelbrot function with respect to r (in spherical coordinates, the length of the vector z). This derivative is given by:

$$(dr)_i = nz_i^{(n-1)}(dr)_{i-1}$$

After iteration is terminated, we calculate the final distance using the formula as discussed by Quilez [3]:

$$d = \frac{|r| \ln |r|}{|dr|}$$

2.2.2 Nylander Formula Change. By altering the Nylander formula to the one below results in a "spikey" Mandlebrot set.

$$z^n = r^n \begin{bmatrix} \cos(n\theta)\cos(n\phi) \\ \sin(n\theta)\sin(n\phi) \\ \cos(n\theta) \end{bmatrix}$$

2.2.3 HSV to RGB Conversion. Our team implemented a function that converts a color from HSV (Hue, Saturation, Value) format to RGB (Red, Green, Blue) format [5]. The HSV representation of color was chosen in order to provide the ability to rotate hue around the color wheel.

The function begins by extracting the hue (H), saturation (S), and value (V) from the input HSV vector. It calculates the chroma (C), which represents the intensity of the color, and determines which sector of the color wheel the hue belongs to. We calculate an intermediate value X to determine the second largest component of this color. The conditional statements use the location in the color wheel to set the appropriate color value. We then adjust all three RGB variables by m to match the "value" portion.

2.3 Shading

The first step was to approximate the gradient at the point using finite difference method. The gradient is equivalent to the normal [1].

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$$

$$\frac{\partial f}{\partial x} \approx \frac{f(x+h,y,z) - f(x,y,z)}{h} \frac{\partial f}{\partial y} \approx \frac{f(x,y+h,z) - f(x,y,z)}{h} \frac{\partial f}{\partial z} \approx \frac{f(x,y,z+h) - f(x,y,z)}{h}$$

Next the group calculated the diffuse, specular, and ambient components as discussed in lecture 6 to produce blinn phong and diffuse images as shown in the results section.

2.4 Fog

The linear interpolation method based on distance for rendering fog involves gradually blending the color of objects with the fog color as their distance from the viewer increases. This technique uses a simple mathematical formula to compute the intensity of fog based on the object's distance from the camera. The further the object, the more it blends with the fog color, effectively simulating the effect of atmospheric scattering. The fog amount is then linear interpolated with the other objects in the scene.

$$\text{fogAmount} = 1 - e^{-t \cdot \text{fogDensity}}$$

2.5 Minor Effects: Texture Mapping + Glow + Shadow

We used the texture method in GLSL to apply a texture to anytime the raymarched rays exceed the max threshold and misses.

Glow effects in visualizations can be enhanced by adjusting the color intensity based on the number of ray steps used in rendering. Points that are closer to a fractal often require more ray steps to determine their visibility, even if they do not directly intersect with the fractal. This method ensures that pixels near the object exhibit a glow, subtly emphasizing their proximity to the fractal boundaries and enhancing the overall visual appeal of the scene.

The computation of soft shadows was achieved adjusting the rate at which shadow hardness changes based on the distance to obstructing objects. This results in shadows that soften gradually with distance, mimicking natural light behavior more closely. This calculation is done during the raymarching process.

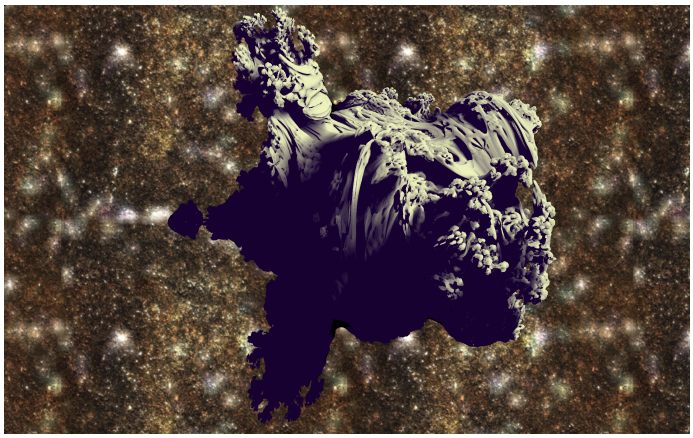

Fig. 2. Shadow and texture mapping

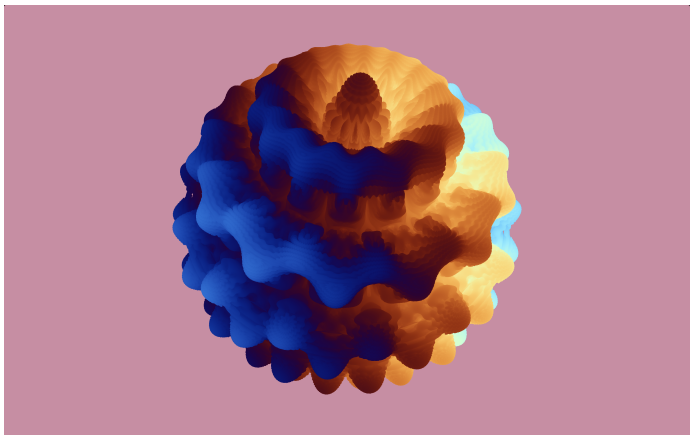

Fig. 3. Glow color palette application

2.6 Problems encountered and Lessons Learned

In implementing the Mandelbrot shader code, our team gleaned several valuable insights. First, this shader's iterative nature underscores the importance of setting appropriate thresholds for computing the set. We also learned about the `mix` function which performs a linear interpolation which is integral to how we color the Mandelbrot set shader.

Additionally, implementing the normalization of pixel coordinates taught us about maintaining aspect ratio correctness and visual accuracy across different screen resolutions. The incorporation of time-based dynamic transformations, such as rotation, added a layer of complexity and taught us more about animating shaders in response to time.

We also learned that zooming into the Mandelbrot shader can be very intensive on our computers. After our shader zoomed enough, our computers would get hot or fans would start to speed up. This is interesting and may be useful to look at through a ray-tracing perspective in order to find ways to make the shader more efficient.

One problem we encountered was the coloring of the Mandelbulb shader. We colored it based off the distance the ray had traveled from the camera position, but were unable to split this value up between 3 different numbers to make an RGB value. We only had one number to color with, so we switched to using HSV, since saturation and value can be set to constant values while we adjust hue based on the aforementioned distance value. This required us to write a function to convert from HSV to RGB as GLSL only takes in RGB. sectionResults

3 RESULTS

3.1 Mandelbrot Set

Our team successfully implemented shader code to visualize the Mandelbrot set, a complex fractal structure. The shader maps each pixel's coordinates to a point in the complex plane and iteratively applies the Mandelbrot equation to determine whether the point belongs to the fractal set. By setting a high iteration threshold and adjusting the coloring based on the number of iterations before the point escapes, our shader effectively highlights the boundaries and intricate details of the set. The color gradient, from deep blue for points far from the set to vibrant orange for those closer, provides a clear visual distinction of the fractal's complex behavior. Additionally, we introduced dynamic visual elements by enabling rotation based on the elapsed time, enhancing the interactive aspect of our visualization.

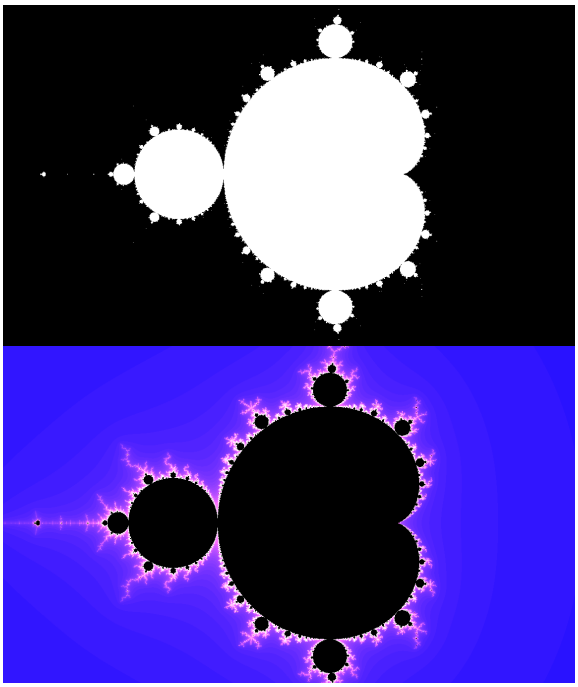

Fig. 4. The group's first two Mandelbrot set Shader results

3.2 Mandelbulb

Our team utilized the raymarching method to visualize the Mandelbulb, a three-dimensional fractal that extends the concept of the Mandelbrot set into higher dimensions. We rendered each point via iterative processing using a generalized version of the Mandelbrot formula and signed distance fucntion to determine their inclusion in the Mandelbulb structure. We set a high iteration limit to ensure precise detailing, with a dynamic color scale highlighting the fractal's intricate topology. The visualization is further enhanced by real-time rotation effects, which allow users to explore the fractal from multiple perspectives, making the experience interactive and engaging.

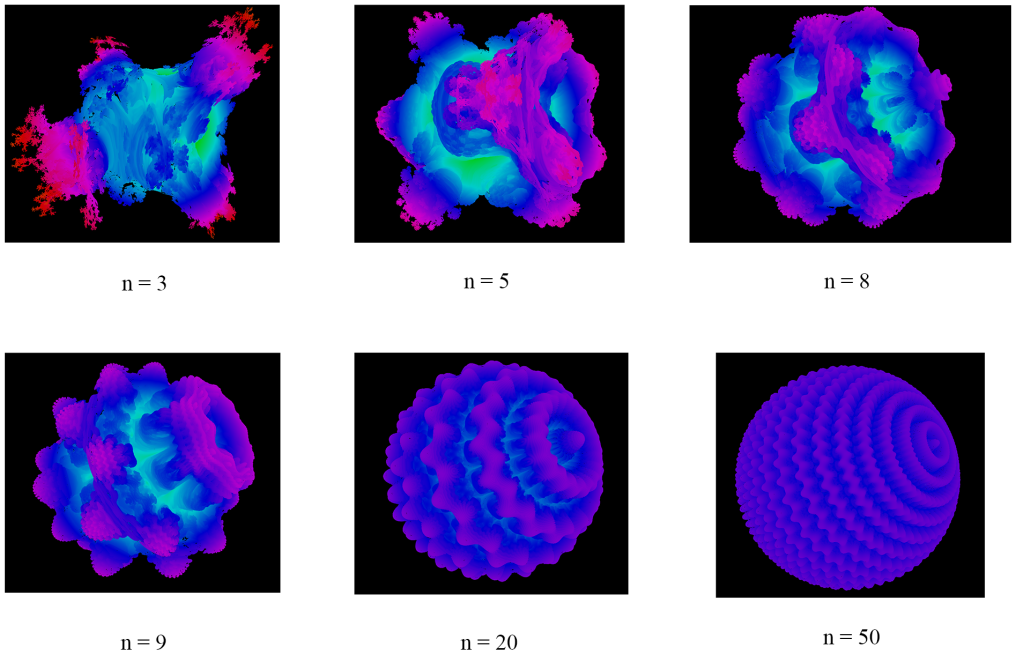

Fig. 5. Mandelbulb Shaders Using the Function $z_{i+1} = z_i^n + z_0$

3.3 Mandelbulb Modifications/Exploration

The Mandelbulb offered a canvas for creativity, and our team seized the opportunity to delve into and experiment with several of the concepts discussed in the class.

3.3.1 Altering the Nylanders formula. A simple change of sine to cosine in the formula and careful selection of color results in a central structure that resembles an inferno, with radiant orange and fiery red tones that emanate outward in an explosive array of sharp, spiky forms. [4]

Our team delved into the HSV color model as outlined in our lectures, which empowered us to create a dynamic Mandelbulb whose colors continuously evolve through the spectrum of the color wheel. This exploration not only deepened our understanding of color theory but also enhanced the visual impact of our fractal, showcasing a vibrant, ever-shifting display of hues.

3.3.2 Shading. We also added some lighting effects onto our Mandelbulb in order to add visual flair to the scene. We implemented diffuse shading, Blinn Phong shading, as well as fog.

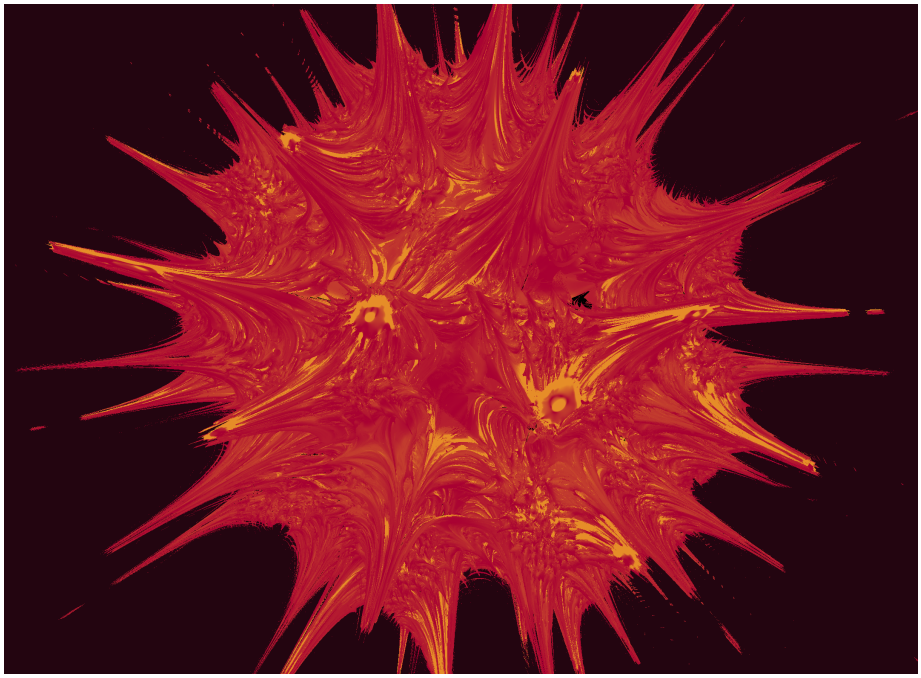

Fig. 6. Mandelbulb Shader with modification of Nylanders formula

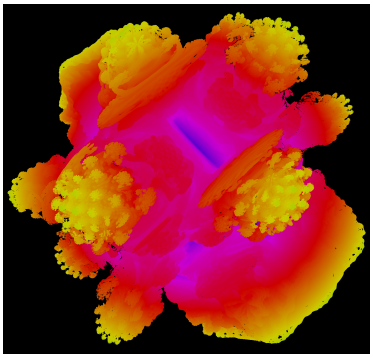

Fig. 7. Mandelbulb Shader Yellow, Red, Purple, Blue

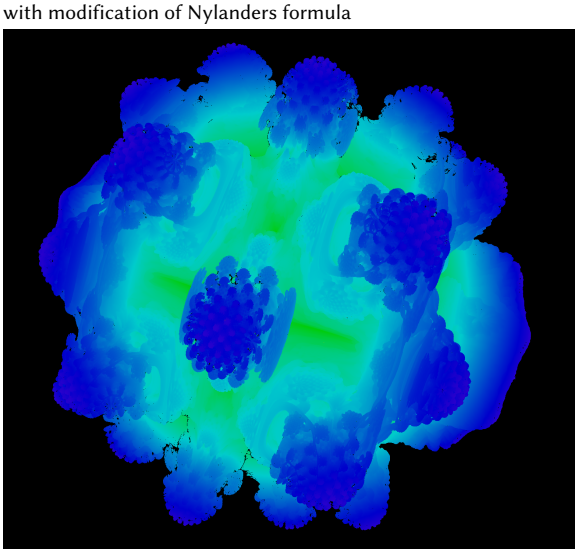

Fig. 8. Mandelbulb Shader Blue, Cyan, Green

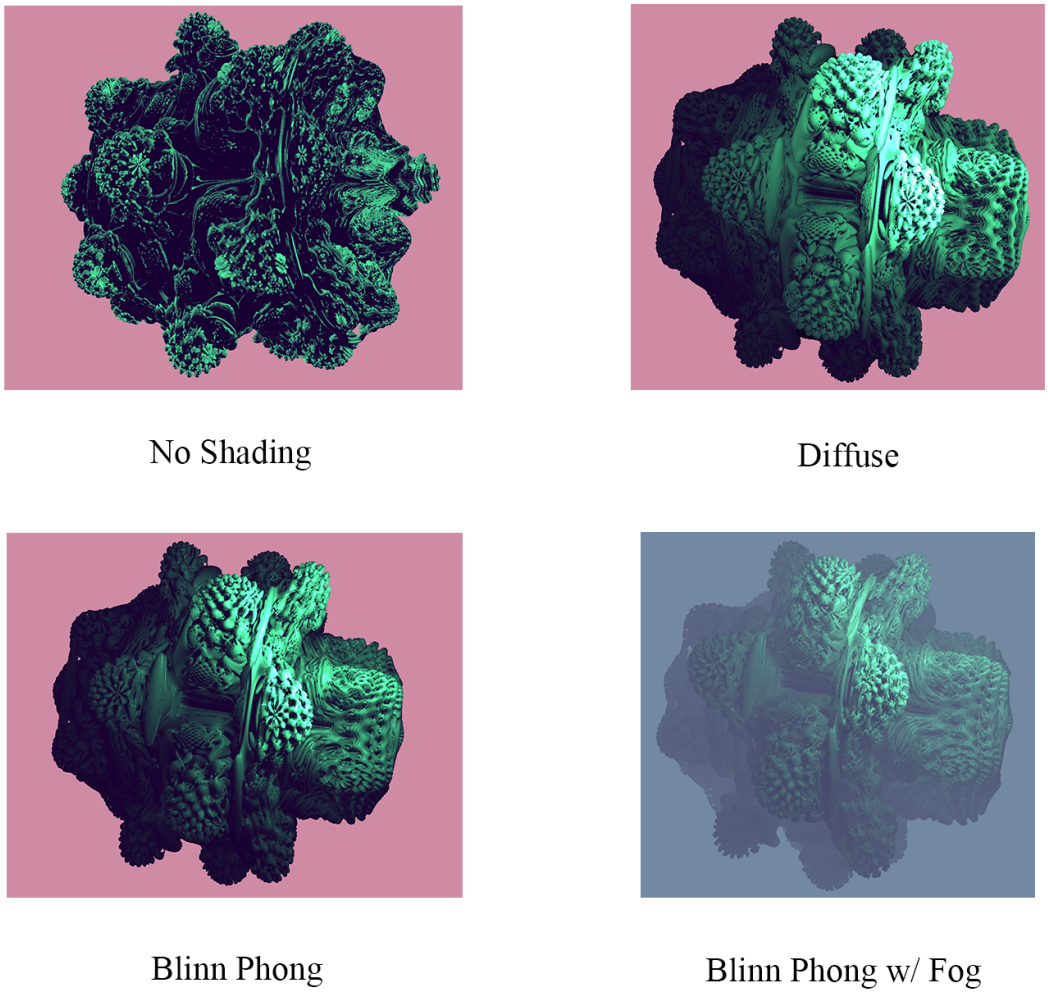

Fig. 9. Implemented Lighting Effects on an n = 8 Mandelbulb

4 OUTSTANDING WORK

The team explored other fractals such as the Quaternion Julia Set, a topic detailed by Crane in his work on ray tracing these complex structures on GPUs [2]. The Quaternion Julia Sets involve complex, four-dimensional shapes that are projected into 3-D. Although the team was not able to fully implement this in the time available, there is much more with Shaders to explore.

5 ACKNOWLEDGMENTS AND INDIVIDUAL CONTRIBUTIONS

Arthur Pommersheim

- Helped pick the topic that would meet 284 standards. Discussed topic choice with Professor Ren Ng.
- Wrote the summary and problem description of the project proposal.
- Wrote the Latex Milestone report.
- Wrote the Latex Final Report.
- HSV to RGB implementation
- Altered the Nylanders formula

Nicolas Vega

- Contributed to the schedule section of the project proposal.
- Wrote the Latex Milestone report.
- Wrote the powerpoint slides for final presentation.

Tony Li

- Created a test fractal based on the shader tutorial.
- Primary contributor for Mandelbrot and Mandlebulb set.
- Made milestone and final video.
- raymarcher

Mya Thanegi Soe

- Contributed to the schedule section of the project proposal.
- Made a basic 2D heart-shaped shader and a basic fractal using spheres.
- Made milestone PowerPoint.
- Made various shaders of fractals: Blinn-Phong, Glow, Fog.

ACKNOWLEDGMENTS

Ren Ng for running a sweet class and all the TAs who stay late.

REFERENCES

[1] 2010. Gradient: proof that it is perpendicular to level curves and surfaces. https://ocw.mit.edu/courses/18-02sc-multivariable-calculus-fall-2010/85c1d85363d9808505351b571d2750aa_MIT18_02SC_notes_19.pdf
[2] Keenan Crane. 2005. Ray Tracing Quaternion Julia Sets on the GPU. http://www.cs.cmu.edu/~kmcrane/Projects/QuaternionJulia/paper.pdf
[3] Inigo Quilez. 2004. Distance to Julia/Mandelbrot set. https://iquilezles.org/articles/distancefractals/
[4] Danial White. 2009. Distance Estimated 3D Fractals. http://blog.hvidtfeldts.net/index.php/2011/06/distance-estimated-3d-fractals-part-i/
[5] Wikipedia contributors. 2024. HSL and HSV — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=HSL_and_HSV&oldid=1218516932 [Online; accessed 26-April-2024].