Visualizing Texture of Fractal Arts in Unity3D Framework

2023 International Conference on Computer Science, Information Technology and Engineering (ICCoSITE) | 979-8-3503-2095-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCoSITE57641.2023.10127738

Benny Hansen Lifindra
Department of Informatics Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
6025221011@mhs.its.ac.id

Riyanarto Sarno
Department of Informatics Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
riyanarto@if.its.ac.id

***Abstract*—An image is a visual representation of a person or art. There exists a type of image called fractal art. It generates a disorganized but precise shape in any calibration. Fractal art often appears in digital images resulting from fractal calculation. The study of this paper focuses on visualizing fractal art with texture within the Unity3D framework. This study proposes a way to apply UV coordinates with Julia set iteration. It manifests in the form of a Unity3D shader code. The proposed code generates the Mandelbrot set and some Julia sets. It is used in conjunction with a batik color palette to generate batik fractal art to be analyzed. The code generates a fractal visual with another added detail. It shows how mapped texture adds more motifs based on the iterated texture coordinates. The fractal visualization is also affected by other factors. Change in iterated UV coordinates either by increment or logarithm remarkably transform generated motif. Textures with different compositions can slightly modify the generated motif other than changing colors. Texture wrap mode provided by Unity3D also influences how the texture is sampled when it is out of bounds. Unity3D automatically handles any calculation error. The result of this study is hoped to be beneficial in visual effects creation.**

Keywords— Fractal, Art, Image, Unity3D, Shader

I. INTRODUCTION

An image can be described as a visual representation of anything. It can be the form of a person or everything in art. An image does not have to utilize the entire visual system. An example of this is an image represented in black and white visual. Image is a means to bring information through our visual system.

Image has been used in fields other than art. In the medical field, 3D volumetric images can be rendered with Digital Imaging and Communications in Medicine (DICOM) images [2]. These 3D images help observe the interior of an object. They help improve the perception of a doctor in diagnosing patients or even planning surgery. Another example of a beneficial image in this field is Globus Pallidus segmentation. It can be used to help diagnose Parkinson's Disease. Various methods exist to improve the segmentation of Globus Pallidus with specific characteristics [9].

However, there exists a type of image with a disorganized shape yet detailed in any sequence or position. This kind of image is called fractal art. A fractal can be described as a geometric shape with thorough details at any small calibration. Fractal art produces chaotic yet defined shapes. They produce a surprising treat for human eyes.

An example of fractal art application is ceramic product design. Fractal arts can be implemented in teapots, bowls, and vases. Xingrui and Wenming commented that it blends modern science with traditional technology [12]. Another example of fractal art is batik. It had existed before fractal was conceptualized. Batik is a dyeing technique that originated from the island of Java in Indonesia. With Fourier Transformation, the fractal characteristics can be found within batik [4]. Romadiastri described several ways to produce a batik fractal [7]. It can be made from a fractal simulated with batik semblance. For instance, the L-System algorithm can be implemented to help construct batik design [8]. Batik fractal can also be drawn by combining both batik and fractal or computing pure fractal calculation. There exists a software called jBatik that helps produce batik patterns utilizing fractal principles [14].

The result of this study will contribute to fractal art application. It is hoped that this study can give literature exposure for fractal art visualization using Unity3D. The findings of this study can be beneficial for game developers who use Unity3D. The software also costs no money to be installed. Anyone with interest or hobbyists can use this study as a source of useful information. It is also hoped that this study can be resourceful for future work in this field.

The following section will review the literature related to this study. This section is then followed by details on the methodology used in this study. The section will explain how fractal art is implemented in Unity3D. After that, the results and analyses of this study will be shared. A conclusion is then drawn to conclude this study.

II. LITERATURE REVIEW

In this section, relevant literature will be explained. This section will define what fractal is as well as fractal art. Related literature to the Julia set, and the Mandelbrot set will be included. As the aim of this study is to visualize fractal art, fundamentals of texture mapping will be added.

A. Fractal

The term fractal was first coined by Benoit B. Mandelbrot in 1975. He then defined fractal as "a set for which the Hausdorff Besicovitch dimension strictly exceeds the topological dimension." [5] The Hausdorff Besicovitch dimension is called the fractal dimension. The name itself is derived from the Latin word *fractus*. The word can be translated as "broken" or "fractured." Mandelbrot noted that *fractus* could also mean irregular because its meaning is contained in a fragment.

However, Mandelbrot's definition of a fractal is inadequate. Falconer believed that the best way to describe a fractal is to list its characteristics [3]. The most common feature found within fractals is that it has exceptional structure at any scale. This means that fractals have fine details even on an arbitrarily small scope. As such, a fractal cannot be illustrated with traditional geometrical language. Fractals also have a form of self-similarity in themselves. Most of the time,

a fractal has its "fractal dimension" greater than its topological dimension. Fractals can be defined in a simple, possibly recursive, manner.

B. Fractal Art

There are some studies related to fractal art applications. Nurjanah et al. described jBatik as software that creates batik motifs using fractal principles [14]. This software first transforms the batik pattern into a fractal mathematical formula. The result will then be modified to produce a more complex formula. Xingrui and Wenming studied how fractals can be integrated into ceramic products [12]. These implementations evoke aesthetics that cannot be expressed by traditional technology. Malishevsky added that much research has been done studying the fractal properties of music for classification [13]. These studies show that fractal art can be digital images and music produced from fractal mathematical calculations.

C. Julia Sets

Falconer outlined how the simple process of Julia sets can construct highly complex sets [3]. Simple functions within complex plane C can create fractals with a striking appearance. Julia sets appear with the iteration of a complex variable function. All points within this plane are denoted with the complex number $c \in \mathrm{C}$. This complex number is written in the form of $c = a + bi$ where $a, b \in \mathrm{R}$ and $i = \sqrt{-1}$. Julia sets are commonly formed by taking a complex number $z = x + yi$ and repeatedly updating z with $z = z^2 + c$ [10]. Zou et al. also added that the Julia set has an escape radius threshold that holds a pivotal role in generating the set [15]. Figure 1 displays several Julia sets depicted by Falconer [3].

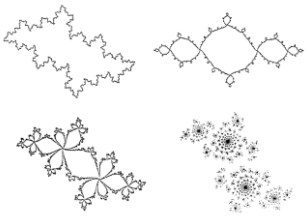

Figure 1. Julia Sets

D. Mandelbrot Set

Zou et al. explained that the Mandelbrot set could be defined as the collection of all points in the complex plane [15]. These points form a connected Julia set of f_c [3]. A point $c \in \mathrm{C}$ belongs to the Mandelbrot set if and only if $z_{n+1} = z_n^2 + c$ does not diverge to infinity when iterated from $z = 0$. In other words, the Mandelbrot set maps all Julia sets, as different c is used in each location [10]. Like Julia set, an escape radius threshold is used to generate the set. Figure 2 illustrates the appearance of a Mandelbrot set.

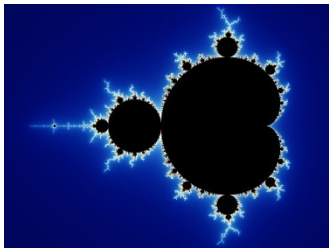

Figure 2. The Mandelbrot Set

E. Coloring Mandelbrot Set

The most used algorithm in coloring the Mandelbrot set is the escape time algorithm. This algorithm is based on the value of n, which is the number of iterations before z_n is outside a certain radius. The escape time algorithm is quite simple to be implemented. However, its simplicity creates bands of color. These bands appear because some neighboring pixels usually have the same number of iterations. This makes them have the same color. A way to solve this is by smoothing possible values so neighboring points do not share the same value. Vepstas provides a revised algorithm that can be used for smooth coloring [11]. Figure 3 depicts another pseudocode representing the algorithm with smooth iteration.

```
   int iter_count = 0;
   float escape_radius = 20.0;
   complex Z, C;
   loop (forever) {
      Z = Z*Z +C;
      iter_count ++;
      float modulus = sqrt (ReZ*ReZ + ImZ*ImZ);
      if (modulus > escape_radius) goto stop;
      if (iter_count > maxiter) goto stop;
   }

stop:
   Z = Z*Z +C; iter_count ++;    // a couple of extra iterations helps
   Z = Z*Z +C; iter_count ++;    // decrease the size of the error term.
   float modulus = sqrt (ReZ*ReZ + ImZ*ImZ);
   float mu = iter_count - (log (log (modulus)))/ log (2.0);
   color_value = colormap_lookup (mu);
   draw_pixel (C, color_value);
```

Figure 3. Revised Pseudocode of the Mandelbrot Set

F. Texture Mapping

Texture mapping can be defined as a resampling process [6]. To get a color, the computer shader figures a location that matches the shading point. This texture lookup picks the color at that point in the image and returns the texture sample. Texture mapping offers many benefits, such as rendering shadows.

Unity3D can import many standard two-dimensional image formats, such as JPEG and PNG, as textures. Unity3D's material then uses these textures as bitmap images. A texture is given coordinates within the range of 0 to 1. These coordinates are references to the location on the imported image [1]. They are referred to as UV coordinates. Within Unity3D, texture mapping is the list of UV coordinates mapped into 3D vertices. This is how Unity3D projects images within a scene. Figure 4 features an example of texture as well as how it is given coordinates in Unity3D.

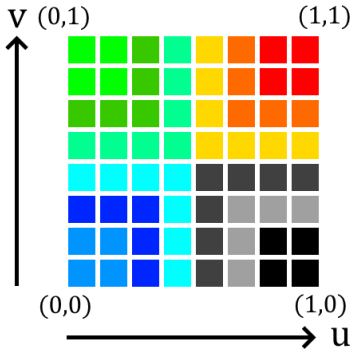

Figure 4. UV Coordinates

Although the range of texture coordinates is between 0 to 1, it is possible to try referencing outside its boundary. Unity3D has a feature to wrap its textures. It determines how the texture is sampled when the coordinates are out of range.

The study of this paper proposes fractal art visualization with texture in the Unity3D framework. This software is a powerful tool for game development. It is a helpful game engine with a variety of features, such as animation and visual scripting. Illustrating fractal art is possible for software of Unity3D caliber. However, there is more to be explored than ordinarily applying a fractal formula.

III. Methodology

In this section, the methodology used in this study will be detailed. This section will first depict the flow of the research conducted in this study. It will then exhibit the framework used and then the utilized source code. Finally, this section will explain the pictures used as textures to visualize the color of the generated fractals.

A. Research Design

Several steps are taken in this study. A literature review is done first to learn and understand the implementation of a fractal. The study will then move on to prepare the framework for visualizing fractals. The fractals are implemented using the shader provided by the framework. The value of several variables within the shader will be altered for analysis purposes. The observed result is then written in this paper. Figure 5 showcases the flow of the research in this study.

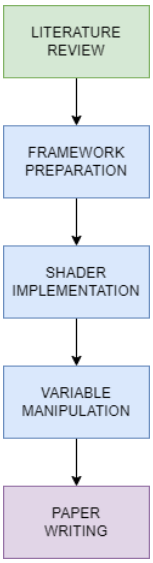

Figure 5. Research Flow

B. Framework

Unity3D will be used as the framework to visualize fractal art in this study. It is a game engine developed by Unity Technologies. The application is free to be installed. Figure 6 shows the interface of the Unity3D 2021 version when a new project is created.

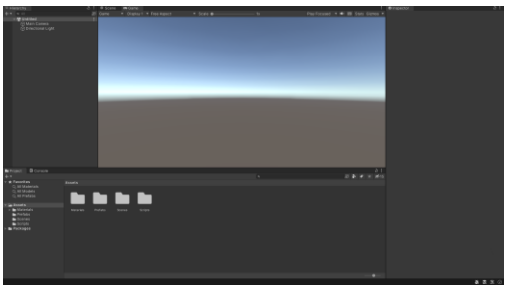

Figure 6. Unity3D

C. Shader

In this study, the Unity3D shader will be used to help visualize the fractal arts with textures. Figure 7 depicts the code used for the shader. The code is mostly copied from the pseudocode presented in Figure 3. The difference is the implementation of a texture sample instead of a normal color value. The tex2D() function is used to help return texture data. The first parameter is the reference to the texture assigned to the material. The second parameter is a two-dimensional variable defining the texture coordinate. The value assigned to the second parameter is defined as $m \times z$. The value of m represents the color value, while z represents the modified UV coordinates.

```
fixed4 frag (v2f i) : SV_Target
{
    float2 uv = _Position.xy + (i.uv - 0.5) * _Resolution.xy;
    float2 c = _C.xy;
    float2 z = uv;
    float r = 20;
    float iter = 0;
    float maxIter = 255;

    for(iter=0; iter<maxIter; iter++){
        z = float2(z.x * z.x - z.y * z.y, 2 * z.x * z.y) + c;
        float modulus = length(z);
        if(modulus > r)
            break;
    }

    if(iter >= maxIter)
        return 0;

    float modulus = length(z);
    float mu = iter - log2( log(modulus) );
    float m = mu / maxIter;

    fixed4 col = tex2D(_MainTex, m * z);

    return col;
}
```

Figure 7. Shader Code

D. Texture

Some textures are prepared for this shader to use. Figure 8 shows the texture used to help visualize the fractal art. It is a color palette associated with batik. Some modifications are also made to the texture. This is to observe how alteration to the texture influences the visualization of the generated fractal. Texture Figure 9 is a texture modified from Figure 8. These textures are also given different wrap modes to be compared together.

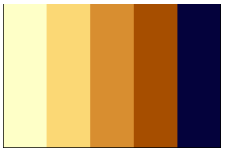

Figure 8. Batik Color Palette

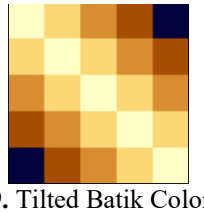

Figure 9. Tilted Batik Color Palette

IV. Result

In this section, the result of the study will be presented. Several fractal arts are generated and analyzed. The results are grouped into four separate subsections. The first subsection analyzes the general use of texture in visualizing fractals with the Mandelbrot set. The second subsection focuses on the comparison between each texture used. The third subsection evaluates the difference between each wrap mode. The fourth subsection examines the visualization of fractals other than the Mandelbrot set.

A. Visualizing Fractals with Texture

Figure 10 visualizes a Mandelbrot set with texture from Figure 8. It is produced by initializing the UV coordinates as constant c. The generated fractal from Figure 10 is colored with an identical pattern in each iteration. They are drawn in the form of ellipses of several colors. Inner iteration produced a visual effect akin to a ripple. The texture produced a pleasant-looking fractal for human eyes.

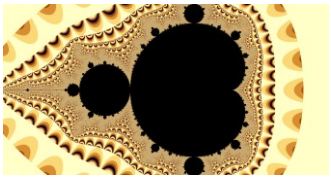

Figure 10. Mandelbrot Set with Texture From Figure 8

However, the "ripple" gets squeezed the closer it is to the set. Figure 11 shows a closer inspection of the generated fractal from Figure 10. The variable z influences texture mapping. As the number of iterations increases, the value of z also increases quadratically. This should give a more detailed texture mapping for those iterations. However, it instead decreases the detail quality on a closer look. This is because the area of later iterations is small enough to fit everything.

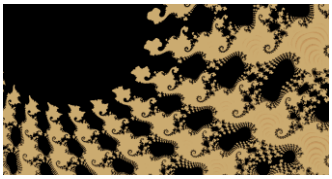

Figure 11. Closer Inspection on Figure 10

The texture can be mapped differently by modifying the value of z. Figure 12 pictures Figure 11 when the second parameter of tex2D() function from Figure 7 is changed to $m \times \log(z)$. Figure 12 shows better detail than Figure 11. Hewing the value of z with log() function also produces a motif like a scorpion tail. It protrudes from the outer iteration into the Mandelbrot set. With a further look at a different position, this theming is visualized in Figure 13.

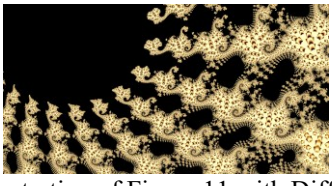

Figure 12. Illustration of Figure 11 with Different Z Value

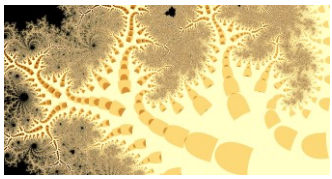

Figure 13. Further Look at Figure 12

A different result can be gained by giving an offset to the value within the log() function. Figure 14 visualizes the result. The scorpion tail motif is replaced with thick color mapping instead. The fractal resembles a neural system but is colored with a batik palette.

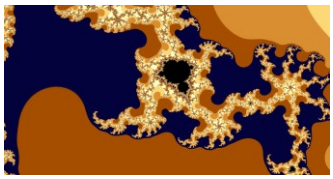

Figure 14. The Mandelbrot Set Visual with Thick Color Mapping

These visualizations show how important the value of z is. Initially, it is given reference to the UV coordinates. Over time, the next value of z is assigned with $z^2 + c$. This process modifies z to become UV the coordinates of the fractal. They provide information to the shader about necessary texture mapping. This method can integrate a different design into the image. This adds more aesthetic value to the fractal.

B. Visualizing Fractals with a Different Texture

Figure 15 visualizes the Mandelbrot set with the texture of Figure 9. The motif painted from texture mapping in Figure 15 is similar to the motif from Figure 10. However, each iteration in Figure 15 forms a ring of intersecting ellipses of several colors. Even so, it also suffers from decreased rendering quality when the set is zoomed in.

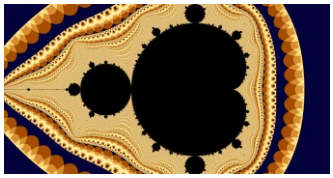

Figure 15. Mandelbrot Set with Texture from Figure 9

To further compare both textures, illustrations made in both Figure 13 and Figure 14 will be replicated with texture from Figure 9. Figure 16 and Figure 17 visualize these. Figure 16 appears heavier in coloring. The scorpion tail motif looks thicker. It resembles the root of a tree instead. On the other hand, Figure 17 only differs in color when compared with Figure 14. Although the obvious difference is color, modified texture design can produce a slight distinction from the produced design.

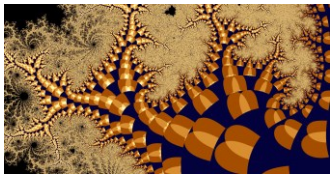

Figure 16. Illustration of Figure 13 with Different Texture

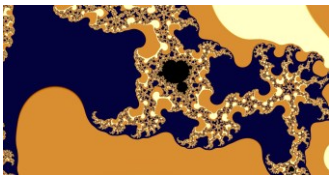

Figure 17. Illustration of Figure 14 with Different Texture

C. *Wrap Mode in Visualizing Fractal with Texture*

Unity3D provides a feature called texture wrap mode. Visualization in Figure 10 is achieved with "Mirror" mode. The texture is tiled, producing a repeating pattern by mirroring the texture whenever a boundary is crossed. Figure 18, Figure 19, and Figure 20 provide an illustration of the same Mandelbrot set with different wrap modes. Figure 18 uses the "Repeat" mode. Figure 19 utilizes the "Clamp" mode. Figure 20 applies the "Mirror Once" mode.

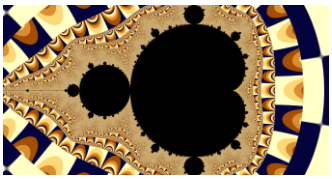

Figure 18. Mandelbrot Set with Repeat Wrap Mode

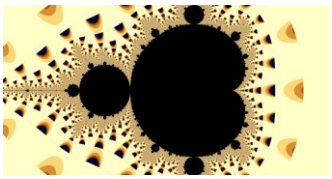

Figure 19. Mandelbrot Set with Clamp Wrap Mode

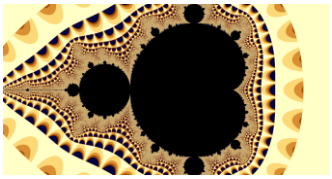

Figure 20. Mandelbrot Set with Mirror Once Wrap Mode

D. *Visualizing Julia Sets with Texture*

Fractal is not limited to just the Mandelbrot set. It is said that the Mandelbrot set is a collection of all Julia sets. Figure 21 attempts to visualize a Julia set. This set uses the constant $c = 0.66i$. The result shows that Figure 21 is also affected by the same problem as Figure 10. The earlier iteration is colored beautifully, but the later iteration is drawn in harsh quality.

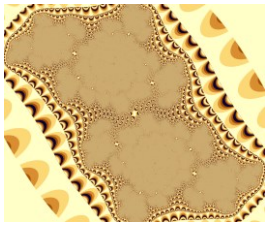

Figure 21. Disconnected Julia Set

Figure 22 recolors Figure 21 with a different return value. It will use the same texture mapping calculation as both Figure 12 and Figure 13. The produced fractal loses visible boundary in earlier iterations. However, the islands are colored in better quality.

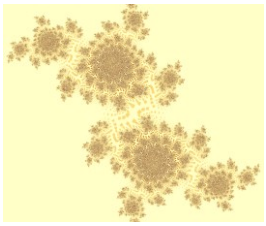

Figure 22. Illustration of Figure 21 with Different Z Value

If the fractal in Figure 22 is given different a constant c, a different fractal can be visualized while retaining the texture mapping calculation. With constant $c = -0.6 + 0.6i$, Figure 23 illustrates a different fractal art. When compared with previous illustrations, Figure 23 does not use many colors. It is then revealed that there is an error within the computation. Variable z is a two-dimensional variable that is repeatedly updated. It is possible within an iteration either value of z is negative. The negative values forced the log() function to return an error. Visualization in Figure 24 tries to mend the error from Figure 23 by making the z variable absolute. The image produced in Figure 24 confirms the error.

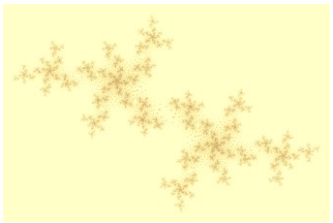

Figure 23. Julia Set of Figure 22 with Different Constant

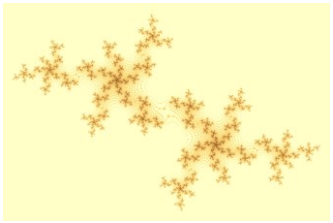

Figure 24. Altered Illustration of Figure 23

These findings show Unity3D's capability in rendering images. Despite the error that happened when computing the logarithm, it continues to map texture regardless. It is also shown how visualizing Julia sets with texture does not differ from the Mandelbrot set. Both are ruled by UV coordinates in variable z. Like the Mandelbrot set, Julia set visuals within Unity3D can also be influenced by assigned texture and texture wrap mode.

Computationally, this proposed method of fractal visualization does not differ too much from other common implementations. The shader code in Figure 7 performs a repeating calculation for each vertex within the plot area. Given that there are $u \times v$ vertices and at most i iterations, then the running time is $O(u \times v \times i)$. In this study, the number of iterations is limited to 255 times. Increasing the limit will certainly yield better visual detail, although with longer computational time.

V. Conclusion

This study proposes fractal art visualization with texture in the Unity3D framework. Each texture within Unity3D is given coordinates which are then used to map themselves into 3D

vertices. These coordinates are often called UV coordinates. They can be iterated repeatedly with Julia sets iteration. These iterations radically spread the coordinate values. These values can then be used to map texture into the fractal image.

Other factors adjust the visual of a fractal with texture in the Unity3D framework. Different images can be imported as textures, providing more texture choices to be mapped. Although the most visible change is color, specific textures can produce a slightly different motif. Unity3D also provides texture wrap mode to help sample textures when the coordinates are out of bounds. Loaded texture can be a repeating pattern or be clamped around its edges. Unity3D is also not prone to errors in calculation. The framework uses error value as texture coordinates when it happens.

The result of this study is hoped to be resourceful for anyone interested in this field. Unity3D is a powerful framework for game development. This proposed method of fractal visualization adds options in creating visual effects within this game engine. It is also hoped that this visualization can help increase creativity in producing batik motifs. Future works are suggested to study visualizing textures on the different fractal methods, such as the L-System algorithm. Future works can also focus on other features of Unity3D that may influence the illustration of a fractal.

ACKNOWLEDGMENT

This research was funded by Lembaga Pengelola Dana Pendidikan (LPDP) under Riset Inovatif-Produktif (RISPRO) Program; the Indonesian Ministry of Education and Culture under Penelitian Terapan Unggulan Perguruan Tinggi (PTUPT) Program; and Institut Teknologi Sepuluh Nopember (ITS) under project scheme of the Publication Writing and IPR Incentive Program (PPHKI).

REFERENCES

[1] "Materials, Shaders, Textures, and UVs," 9 November 2017. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.probuilder@4.3/manual/workflow-texture-mapping.html. [Accessed 30 November 2022]

[2] A. Fajar, D. W. Santoso, Z. R. Bahen, R. Sarno and C. Fatichah, "Color Mapping for Volume Rendering Using Digital Imaging and Communications in Medicine Images," in *2021 International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*, 2021.

[3] K. Falconer, Fractal Geometry: Mathematical Foundations and Applications 2nd Edition, John Wiley &Sons, 2003.

[4] M. Lukman, Y. Hariadi and A. H. Destiarmand, "Batik Fractal: From Traditional Art to Modern Complexity," in *Proceeding Generative Art X Milan Italia*, 2007.

[5] B. B. Mandelbrot, The Fractal Geometry of Nature, W. H. Freeman and Co., 1982.

[6] S. Marschner and P. Shirley, Fundamentals of Computer Graphics Fourth Edition, Boca Raton: CRC Press, 2015.

[7] Romadiastri, "Batik fraktal: perkembangan aplikasi geomteri fraktal," *Delta : Jurnal Ilmiah Pendidikan Matematika,* vol. 1, no. 2, pp. 158-164, 2013.

[8] Saefurrohman and D. H. U. Ningsih, "Desain Motif Batik Dengan Metode Fraktal Dan Algoritma L-System untuk Membangun Pustaka Batik Wali," *Dinamik,* vol. 21, no. 1, pp. 42-51, 2016.

[9] Y. Setiawan, C. Fatichah and R. Sarno, "A New Local Gaussian Variational Level Set for Globus Pallidus Segmentation," *International Journal of Intelligent Engineering & Systems,* vol. 13, no. 5, pp. 317-326, 2020.

[10] K. Sims, "Understanding Julia and Mandelbrot Sets," [Online]. Available: https://www.karlsims.com/julia.html. [Accessed 30 November 2022].

[11] L. Vepstas, "Renormalizing the Mandelbrot Escape," 1 June 1997. [Online]. Available: http://linas.org/art-gallery/escape/escape.html. [Accessed 30 November 2022].

[12] L. Xingrui and L. Wenming, "The Application of Fractal Art in Ceramic Product Design," in *7th International Forum on Industrial Design 17–19 May 2019*, Luoyang, 2019.

[13] A. Malishevsky, "Applications of Fractal Analysis in Science, Technology, and Art: A Case Study on Geography of Ukraine," in *2020 IEEE 2nd International Conference on System Analysis & Intelligent Computing (SAIC)*, Kyiv, 2020.

[14] I. Nurjanah, A. G. Abdullah and I. Widiaty, "The Application of jBatik Software for Batik Products," in *IOP Conference Series: Materials Science and Engineering*, 2020.

[15] C. Zou, A. A. Shahid, A. Tassaddiq, A. Khan and M. Ahmad, "Mandelbrot Sets and Julia Sets in Picard-Mann Orbit," *IEEE Access,* vol. 8, pp. 64411-64421, 2020.