Real-Time Rendering: Final Project Report
Rendering the Gas Giant Jupiter Using Noise

Xue Yu (Zoe YUU), Trinity College Dublin, MSc. Computer Science – R and VR

Abstract

Inspired by SIGGRAPH paper *A physically-based night sky model* [2], experiments have been done in this project to render the "gas giant" Jupiter using Simplex noise, Fractal noise, Ridge noise, as well as other experimental attempts to represent the Great Red Spot of Jupiter.

Background

In the study conducted in 2001, a physically-based model of the night sky has been presented. Using astronomical data, Researchers have modelled the sight of the Moon, the visible stars, the milky way and other nebulae, as well as the illumination from the Moon, the stars, the zodiacal light, the galactic light and the airglow. The model has been placed in a Monte Carlo ray tracer and a variety of night scenes with or without clouds have been rendered [2].

This paper inspired me to research for methods to render celestial bodies (without using the ray tracing approach). The experience of two dissatisfying attempts to render the sun introduced the concept of noise. Since the algorithm of Perlin noise has been invented in early 1980's, noise is widely used to render the textures with random patterns in nature, such as the terrain, the skin of a tree etc. [1,5]. In the "star blogs", game designers of Seed of Andromeda used noise to render Jupiter in real time [5]. Following their approaches, in this project, attempts have been conducted to render the "Gas Giant" Jupiter in the real-time environment.

Implementation details

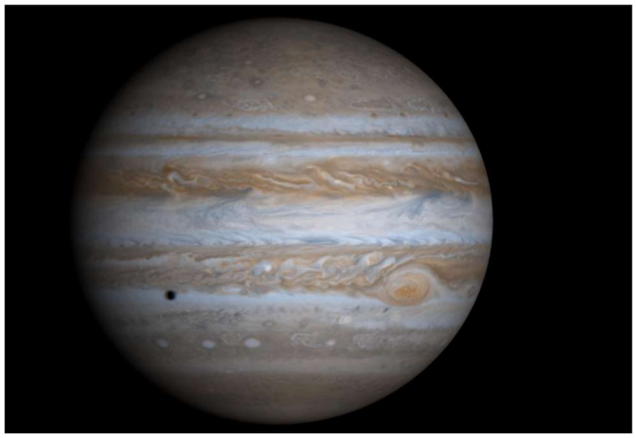

Figure 1. Nasa. Jupiter image captured by Hubble [4]

1. The real picture of Jupiter captured by Hubble shows that the gas giant can be resembled by a sphere with horizontal strips and circle formed storms. In step one, Jupiter has been rendered as a sphere using figure 2 as texture and illustrated by the diffuse light

placed behind the camera[1].

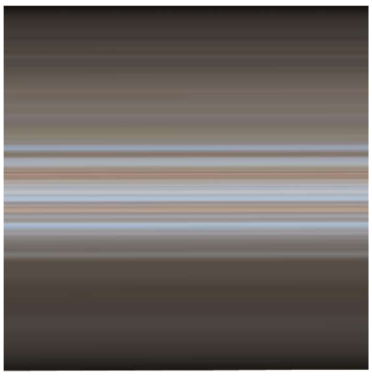

Figure 2. Texture of Jupiter

Figure 3. Shader 1, texture only

2. In the next step, the Simplex noise has been added to the coordination of the texture. A 3D Simplex noise function implemented by Ian McEwan has been used[2] [3], taken in the sum of the vertices' normals and a uniform time_factor (updated every frame), output a float noise from -1 to 1. The result is figure 4[3]:

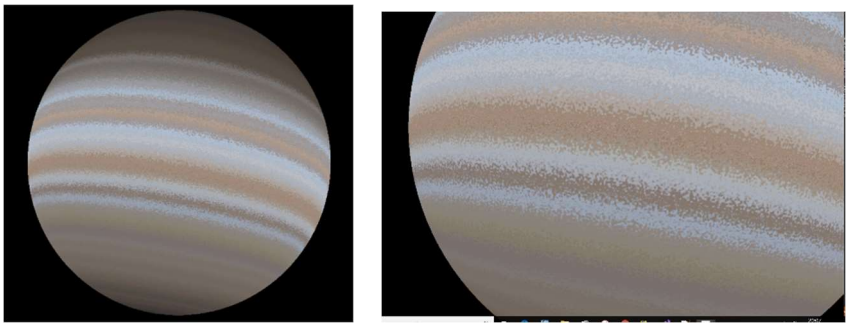

Figure 4. Simplex noise and a close look

3. To make the details looks the same at every scale, even after zoomed in, fractal noise (multi-octave noise) – which usually used to simulate the natural phenomenon – has been applied. 6 passes have been calculated upon the Simplex noise function. To limit the suffering of the performance, the octave number has been set as 6[4]; Later, Ridged fractal noise – commonly used to render mountains on planet surface – has also been added into the equation [5]. The sum of both noises has been used to manipulate the coordinates of the texture[5].

[1] The fragment shader used in this example is "JupiFS1.glsl".
[2] Details can be found in the shader code.
[3] The vertex shader used in every example remains the same. The fragment shader used in this example is "JupiFS2.glsl".
[4] The fragment shader used in this example is "JupiFS3.glsl".
[5] The fragment shader used in this example is "JupiFS4.glsl".

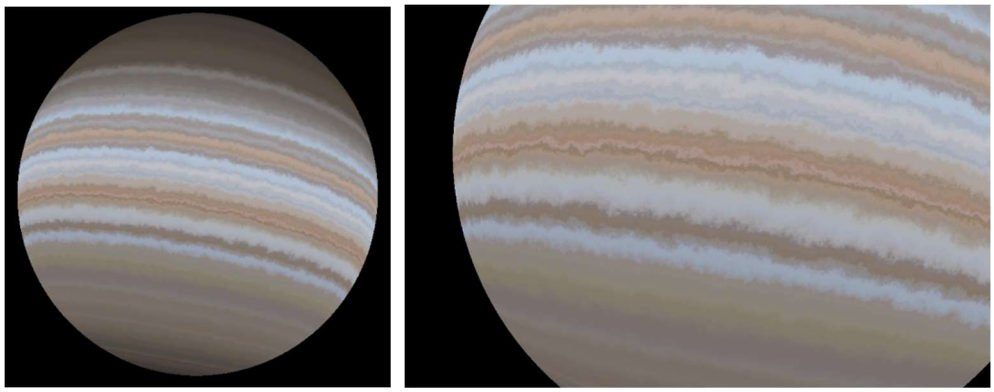

Figure 5. Fractal noise and a closed look

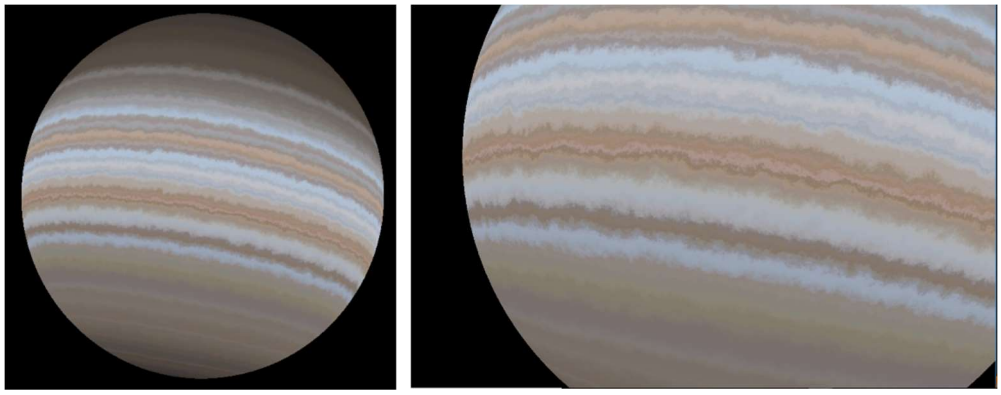

Figure 6. Ridged fractal noise and a closed look

4. The trademark of Jupiter is its storms in the atmosphere. To simulate this, the method of thresholds has been implemented: only noises with certain frequency and amplitude will be selected in this layer. Furthermore, 3 thresholds equations have been calculated and multiplied together to give the storm a form of a circle [5]. This threshold noise has been added to the previous result to get the imitation of Jupiter with real-time storms[6].

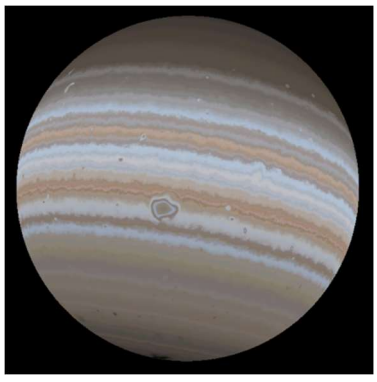

Figure 7. Thresholds noise

5. To make the spiral of the "Great Red Spot" clearer, texture coordinates have been rotated[7] and then translated[8] by the noise value.

---

[6] The fragment shader used in this example is "JupiFS5.glsl".

[7] The fragment shader used in this example is "JupiFS6.glsl".

[8] The fragment shader used in this example is "JupiFS7.glsl".

Figure 8. Rotation

Figure 9. Translation

6. To ensure that "Great Red Spot" has a reddish color, a layer with color scaled by the sum of noise has been added to the final color. Finally, a skybox with the universe as background texture has been added in the final demo[9]. For the first demo[10] uploaded to YouTube, all 8 shaders have been loaded in and applied to the model sequentially.

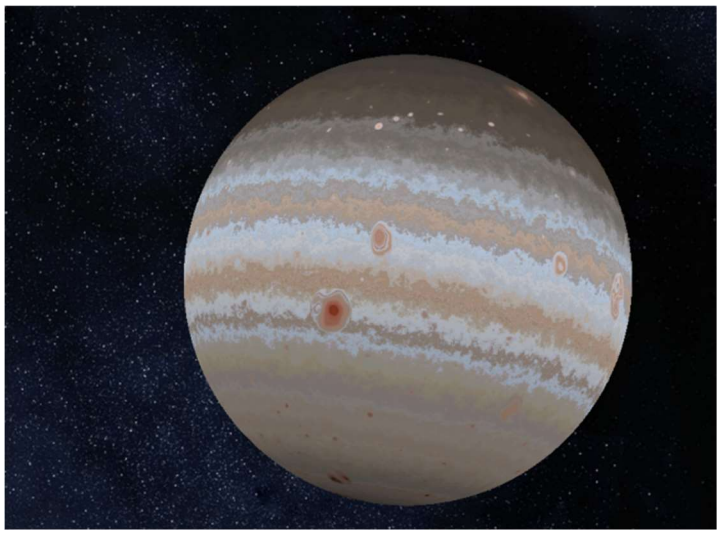

Figure 10. Reddish color on storm

Results and evaluation

Experiments have been run on a Surface 3, with Win10 as OS, Intel(R) Atom(TM) x7-Z8700 as CPU (1.60 GHz), and Intel(R) HD Graphics as GPU. As the program was running, CPU has been 100% used, GPU 22% used, and memory 85% used.

Attempts for improvements and limitations

Despite the attempts made to give "Great Red Spot" a larger shape, the end result still looks rather small and circle like.

The approaches haven't been added to the end result are[11]:

---

[9] The fragment shader used in this example is "JupiFS0.glsl".

[10] Details and used assets can be found in codes "main.cpp".

[11] Although useless to render Jupiter, those patterns are interesting nonetheless and might be useful one

1. Simplex noise has been added to the end result, with frequency and amplitude adjusted to simulate noise at different wavelengths. The results are following[12]:

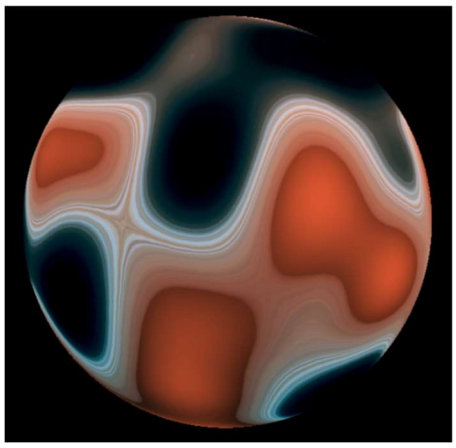

Figure 11. Shader 9

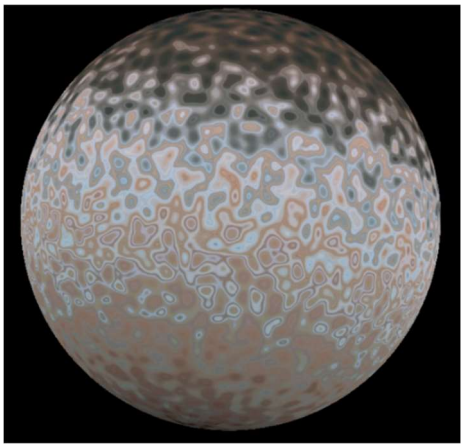

Figure 12. Shader 10

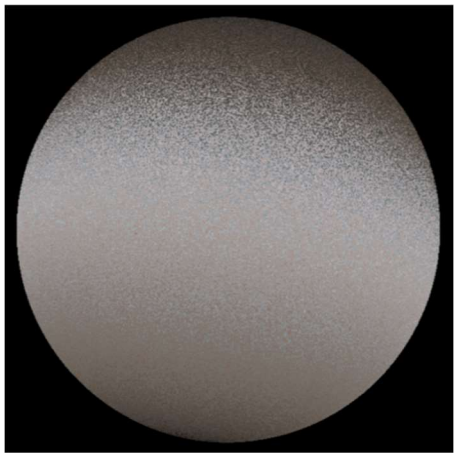

Figure 13. Shader 11

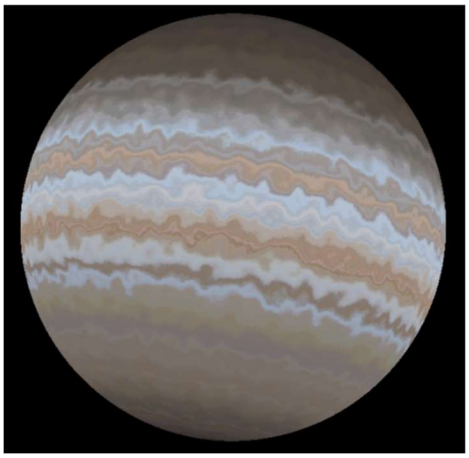

Figure 14. Shader 12

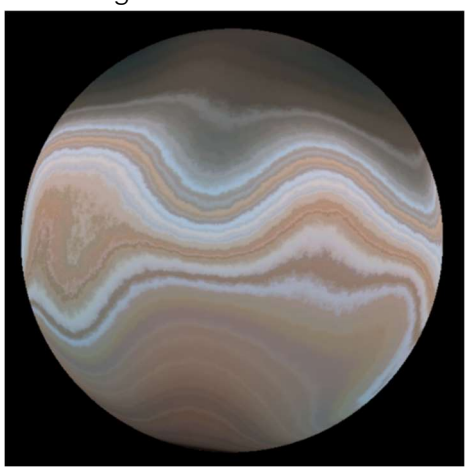

Figure 14. Shader 13

2. To change storms' size, an attempt has been made to firstly apply translation, then

day to render other things, such as patterns on stones, sands, even lava or the sun.

[12] The fragment shaders used in these are "JupiFS9.glsl", "JupiFS10.glsl", "JupiFS11.glsl", "JupiFS12.glsl", "JupiFS13.glsl".

apply rotation upon translation. However, the result gained from this attempt is nearly identical to the one applied rotation firstly and the translation next.

3. Four or two thresholds instead of three have been tried, in attempts to change the storm's shape, the results from those attempts are curious: instead of Jupiter, the texture looks more like the sun[13].

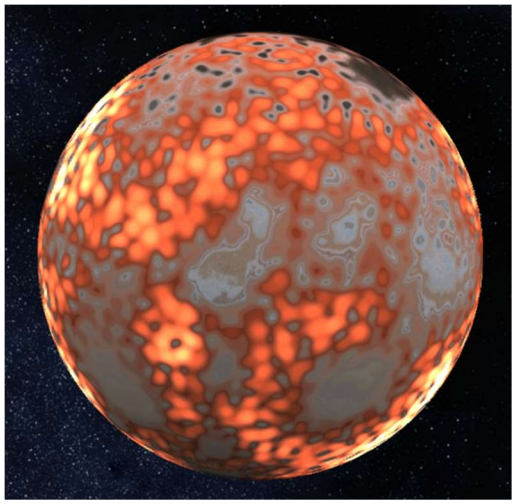

Figure 15. Shader 14 4 thresholds

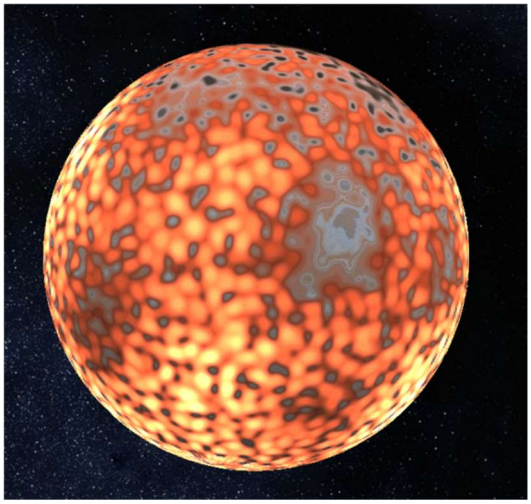

Figure 16. Shader 16, 2 thresholds

Final video on YouTube

Jupiter with different shaders: https://youtu.be/9O9ZB8cl4Ho

(in which shaders from "JupiFS1.glsl" to "JupiFS7.glsl" haven been displayed in sequence, the camera zoomed in when using Simplex noise, Fractal noise and Ridge noise to give viewer a closer look of the detailed patterns, in the end of the video, the final result with shader "JupiFS0.glsl" has been displayed)

Jupiter (End Result) https://youtu.be/dzozw945zmQ

(in which the final result with shader "JupiFS0.glsl" has been displayed and given time for the noise-generated storms to develop themselves and change shapes.)

Reference

[1] GONZALES, P. AND LOWE, J. Book of shader. https://thebookofshaders.com/11/

[2] JENSEN, H. DURAND, F., DORSEY J., STARK, M., SHIRLEY, P. AND PREMOZE, S. A physically-based night sky model. SIGGRAPH '01 Proceedings of the 28th annual conference on Computer graphics and interactive techniques (2001) PP. 399-408

[3] MCEWAN, I. Simplex noise. https://github.com/ashima/webgl-noise

[4] NASA, ESA, and SIMON, A. http://www.spacetelescope.org/images/heic1410a/

[5] SEED OF ANDROMEDA. Star blog. https://www.seedofandromeda.com/blogs/49-procedural-gas-giant-rendering-with-gpu-noise

[13] The fragment shaders used in these are "JupiFS14.glsl", "JupiFS16.glsl".