DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Final

Planeta Generado Proceduralmente

16 de febrero de 2022

Fundamentos de la Computación Gráfica

Integrante	LU	Correo electrónico
Imperiale, Luca	436/15	luca.imperiale95@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
https://exactas.uba.ar

Índice

1. Introducción

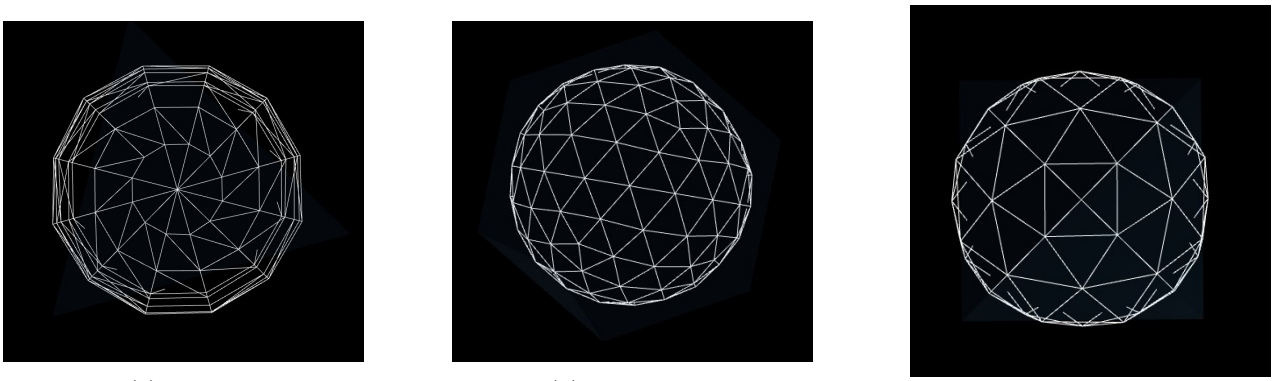
Para este trabajo práctico final, elegí entre las opciones propuestas, la de renderizar un planeta generado proceduralmente.

Usé ThreeJS para no complicarme con tener que programar los shaders manualmente, y poder enfocarme en la parametrización del planeta, y en como se generaban los colores.

2. Geometría del planeta

La idea básica para conseguir la superficie del planeta fue, desde una esfera base, iterar por todos los puntos y modificar la coordenada del radio de acuerdo a algún ruido.

El primer problema que encontré, fue que si partía de una esfera, los vértices no formaban planos uniformes. El problema se hacía particularmente malo en los polos de la esfera. En la figura 1, se ve la comparación con otras geometrías de la librería utilizada, y la que parece ser mas regular es el icosaedro. Éstas dos ultimas, están basadas en poliedros de 20 y 8 caras respectivamente, pero subdividiendo cada cara hasta formar una esfera (2)

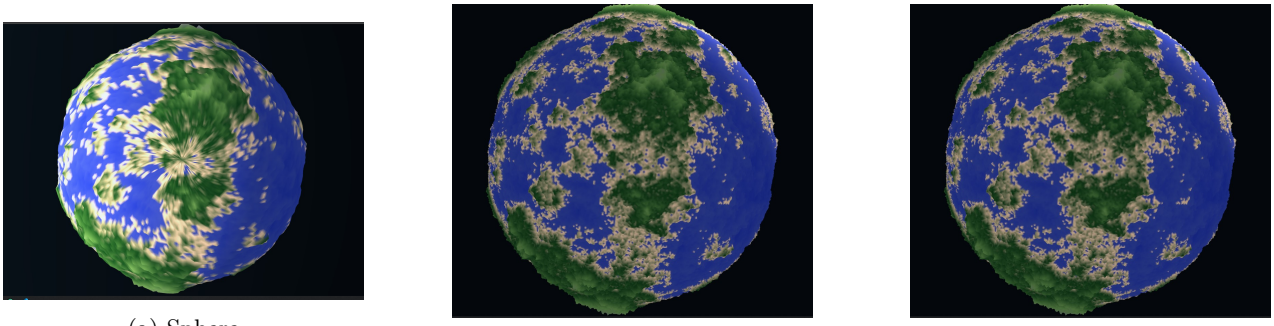


(a) Sphere (b) Icosahedron (c) Octahedron

Figura 1: Vista superior de las distintas mallas base

Figura 2: Subdivisión de un icosaedro

En la figura 3 se ve la comparación de como quedan los polos ya pintados. En la esfera se nota mucho los triángulos estirados, y entre las otras 2 no parece haber mucha diferencia en radios y niveles de detalle altos, aunque si mirándolo en detalle se notan las simetrías en el octaedro que no se presentan el segundo.



(a) Sphere (b) Icosahedron (c) Octahedron

Figura 3: Vista superior de las distintas mallas base, con los colores ya aplicados

2.1. Ruido

En cuanto a la elección del ruido a utilizar, experimenté con varios.

La primera opción y la más simple fue utilizar un random. Pero como vimos en clase, esto nos dejaba con una distribución no continua de puntos, por lo que se generaban "saltos"en la superficie del planeta que se veían muy mal.

Luego probé con el Simplex Noise de ThreeJS. La superficie era continua pero no logré parametrizarlo de ninguna manera para que, independientemente de la *seed*, genere un resultado que se viera bien. Por cuestiones puramente estéticas decidí seguir buscando.

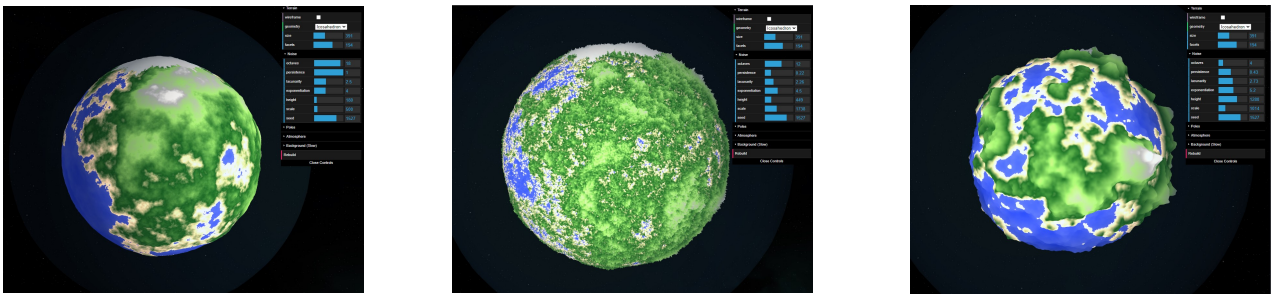
Finalmente terminé utilizando una implementación de Perlin Noise[1], que podía usar fácilmente tanto para ruidos 1D como para 3D, era fácil parametrizar la *seed* y generaba mejores resultados.

Por otro lado, al valor generado por el ruido debía modificarlo con ciertos parámetros, que respondan a la generación de una superficie, como por ejemplo la altura o que tan plana quiero que sea.

Para esto, utilizando como referencia el trabajo de SimonDev, en una serie de videos que explica como generar mundos proceduralmente, con un *scope* muy por encima de este trabajo, usé *Fractional Brownian Motion*. Esto es una forma de "superponer" capas de ruido de acuerdo a parámetros de una función determinada, que lo genera en definitiva es darle un "impulso"al ruido, o osea, que sea parametrizable cuanto un ruido va a tender a cambiar de dirección.

Estos parámetros permiten generar distintos tipos de planetas, pudiendo ser mas realistas o mas *cartoonish*. Son los siguientes:

- Scale: cantidad de picos que tiene el terreno
- Height: altura máxima de picos
- Exponentiation: aplana todo de acuerdo al factor, por defecto 4
- Octaves: le agrega mas o menos textura "agrietada"
- Persistance: inversamente proporcional a la cantidad de detalles
- Lacunarity: que tan "en punta" terminan los picos



(a) (b) (c)

Figura 4: Planetas generados con distintas configuraciones, misma *seed*

[1] https://github.com/alterebro/perlin-noise-3d

3. Coloreo

A este punto tenemos todos los vértices de la mesh del planeta bien ubicados, nos falta darles color a cada uno de ellos.

Esto se hace muy similar a la altura, iterando por cada punto. Pero en vez de modificar las coordenadas, se le asigna un color a cada vértice dependiendo de su coordenada de radio.

En la primera implementación, se subdividía la diferencia entre el punto mas bajo y el mas alto en *buckets*. Y, dependiendo de en qué *bucket* entraba la altura se le asignaba un color.

Los resultados de esto no eran muy buenos, ya que se generaban grandes sectores donde el color era exactamente el mismo, lo que le sacaba textura al render final.

Por lo tanto, lo siguiente que hice fue armar un spline lineal con lo que eran anteriormente los extremos de los *buckets* como puntos. Esto genera que para cada vértice, se interpolen linealmente los 2 colores, según donde caiga el punto.

Los resultados fueron muchos mejores y terminé quedándome con esta idea.

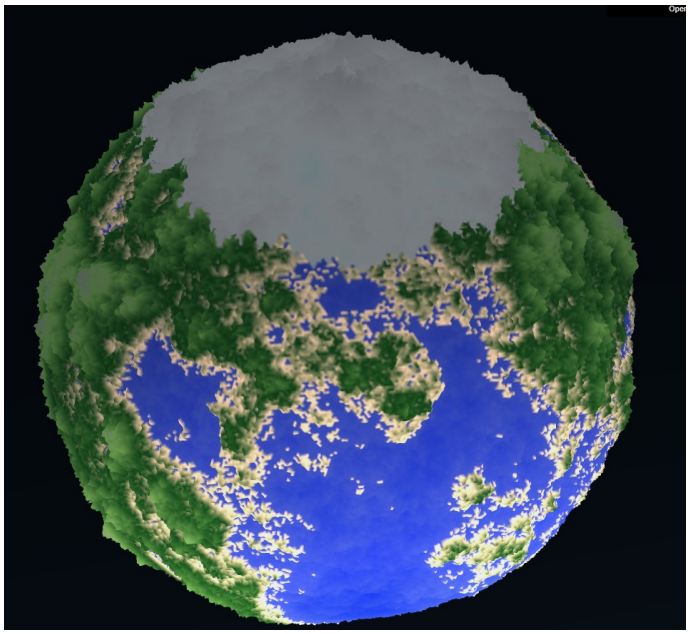


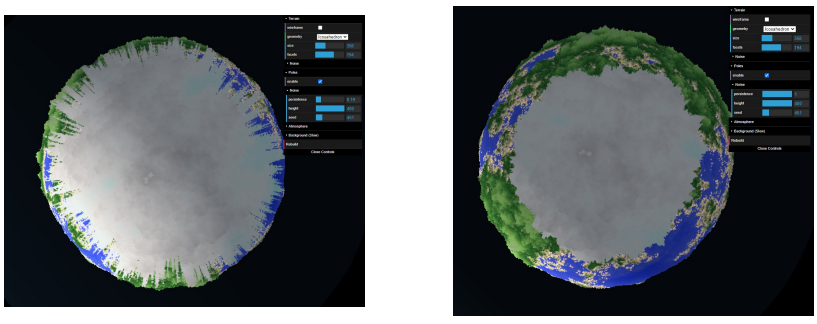
Figura 5: Foto en la que se ven los colores de cada spline(océano, polos y tierra)

3.1. Polos

Algo para mencionar aparte son los polos. Para conseguir el color usan la misma técnica que el resto de la superficie, solo que otro spline(al igual que los océanos).

La diferencia es como se arma el borde. Obviamente no quería definir una latitud límite tanto norte como sur para empezar a colorear con el spline de polos, ya que el limite quedaría muy demarcado.

Lo que terminé haciendo es usando el mismo ruido mas el *FBM* que para la altura, pero en una dimensión. Entonces, ahora el parámetro es la longitud, y el ruido indica hasta que latitud cubre el polo. Algunos parámetros quedaron fijos, y otros se pueden modificar para conseguir distintos tipos de polos, como la altura o la persistencia.



(a) (b)

Figura 6: Polos generados con distintas configuraciones, misma *seed*

4. Atmósfera, luces y fondo

Ya llegando a los detalles, agregué una atmósfera alrededor del planeta. Se genera con el mismo método que la superficie, para que tenga una cierta textura, pero con parámetros fijos. El color también es fijo y el material es casi transparente.

Para las luces agregué una luz global tenue, como para que lo no iluminado se siga viendo. Y también simule el sol con una luz que gira en torno al planeta(con una ligera oscilación vertical para que los polos queden iluminados).

Para el fondo utilice un cubo de texturas generadas, generadas en una herramienta llamada *space-3d*. Por defecto no están habilitadas, porque tardan mucho en cargar.

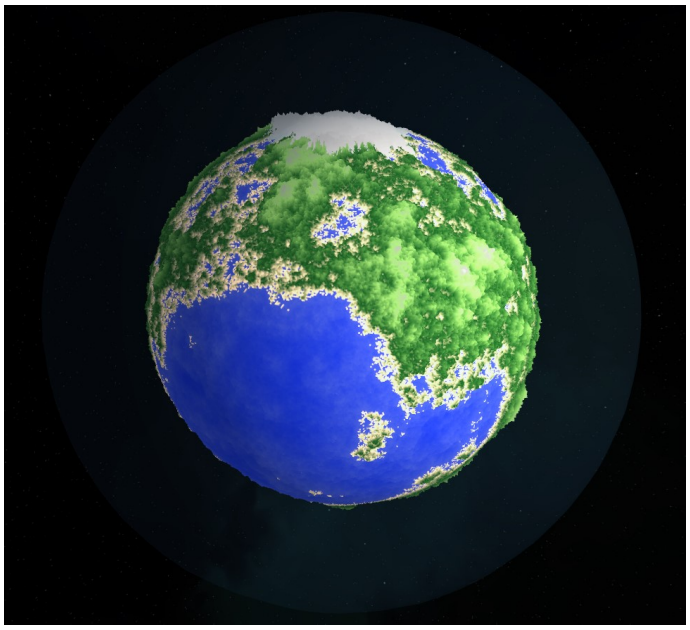


Figura 7: Foto con atmósfera y fondo activados

5. Conclusiones

Me quedé con las ganas de agregarle mas cosas al planeta, como por ejemplo objetos en la superficie, nubes, la posibilidad de colorear por biomas u otro objeto orbitando. Algunas de estas probaron ser mas difíciles que otras. Con las nubes me estuve peleando bastante tiempo para conseguir que se vieran bien, pero al final termine descartando todo. Y otro objeto orbitando es sencillo de hacer, solo que me quedé sin tiempo.

Como conclusión, puedo decir que el trabajo fue, al igual que la materia en general, un gran primer acercamiento a la computación gráfica. Me topé con bastantes problemas de trabajar con librerías y lenguajes que nunca había usado, pero el producto final terminó convenciéndome.

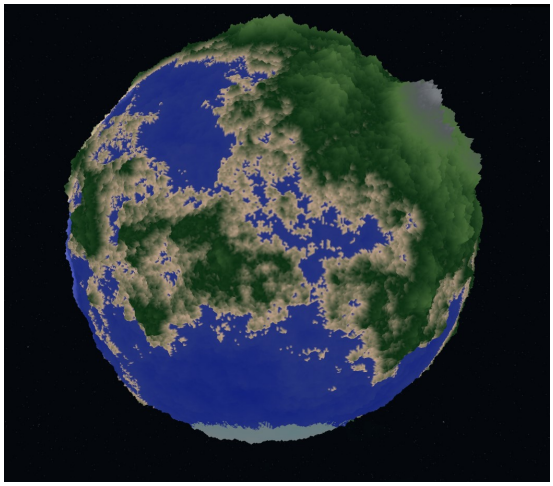


Figura 8: Ultima foto porque quedaba un espacio blanco