



## DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

# Trabajo Práctico Final: Mundos Procedurales

30 de noviembre de 2021

Fundamentos de la computación gráfica  
Emmanuel Iarussi

Integrante	LU	Correo electrónico
Alejandro Cortez	576/17	alejandro17198@gmail.com
Matías Waehner	294/17	matiaswaehner@gmail.com



**Facultad de Ciencias Exactas y Naturales**  
Universidad de Buenos Aires

Ciudad Universitaria - Pabellón I

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Argentina

Tel/Fax: (54 11) 4576-3359

<http://exactas.uba.ar>

# Índice

<b>1. Introducción</b>	<b>3</b>
<b>2. Procedimiento</b>	<b>4</b>
2.1. Creación de la esfera . . . . .	4
2.2. <i>Ruido de Perlin</i> para crear un terreno realista . . . . .	5
2.2.1. Fractional Brownian motion . . . . .	6
2.3. Coloración y retoques finales . . . . .	6
<b>3. Resultado Final</b>	<b>7</b>

# 1. Introducción

Decidimos trabajar en el proyecto de generación de mundos procedurales. Construimos una aplicación web con una simple interfaz para parametrizar la generación.

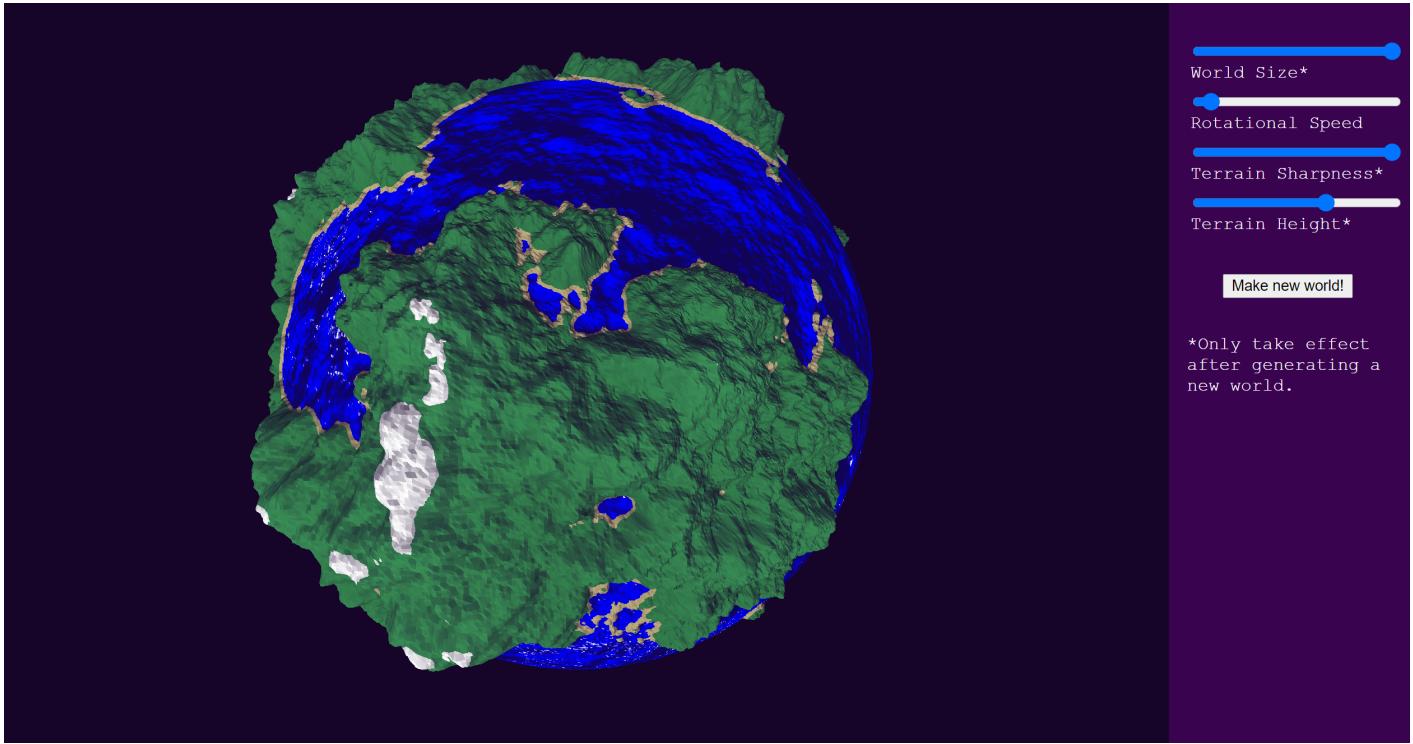


Figura 1: Mundo procedural que creamos y su interfaz

El usuario puede parametrizar la creación del mundo procedural mediante:

- *World Size*, que indica el tamaño del mundo.
- *Terrain Sharpness*, que parametriza que tan irregular va a ser el terreno.
- *Terrain Height*, para modificar la altura media del terreno.

Vamos a ir explicando estos parámetros en la siguiente sección.

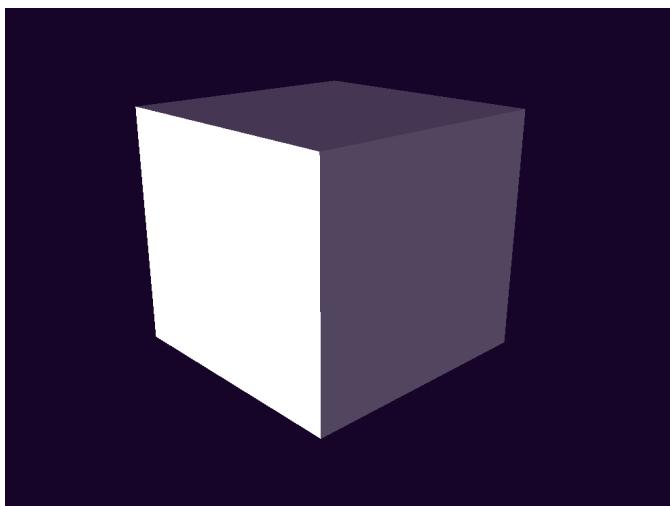
## 2. Procedimiento

Para crear un mundo que se asemeje a uno real (o por lo menos, a nuestro planeta), realizamos lo siguiente:

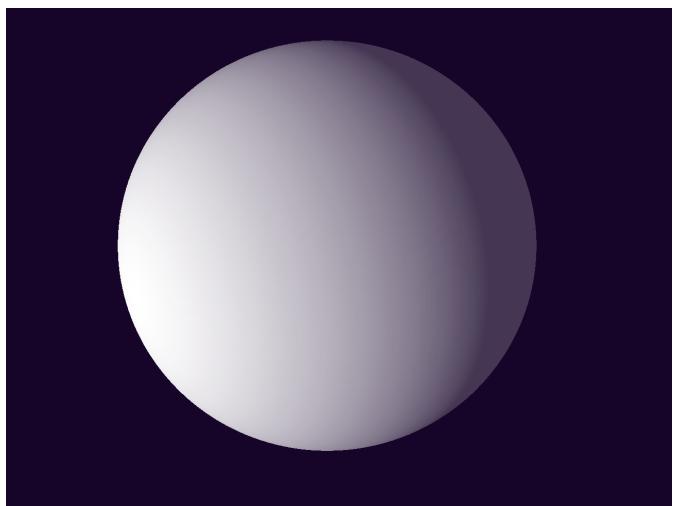
1. Creamos una esfera
2. Agregamos ruido al terreno. Para hacer esto, usamos *Ruido de Perlin*, ya que no cualquier ruido aleatorio construye un terreno realista.
3. Coloreamos el mundo de acuerdo a la altura del terreno.
4. Traemos todos los vértices correspondientes al agua al mismo nivel.
5. Agregamos iluminación y reflexión.

### 2.1. Creación de la esfera

Para crear una esfera, decidimos crear una cubo, y luego normalizar la distancia de todos los vértices, haciendo que tome forma esférica.



(a) Cubo



(b) Esfera

Originalmente habíamos decidido hacer una esfera directamente, iterando a través de las coordenadas esféricas de la misma, y siguiendo el tutorial citado [1]. Sin embargo, hacer esto nos trajo el problema de que la creación de ruido se torna confusa cerca de los polos, como podemos ver en la siguiente imagen, que corresponde a una de las primeras instancias de nuestro trabajo.

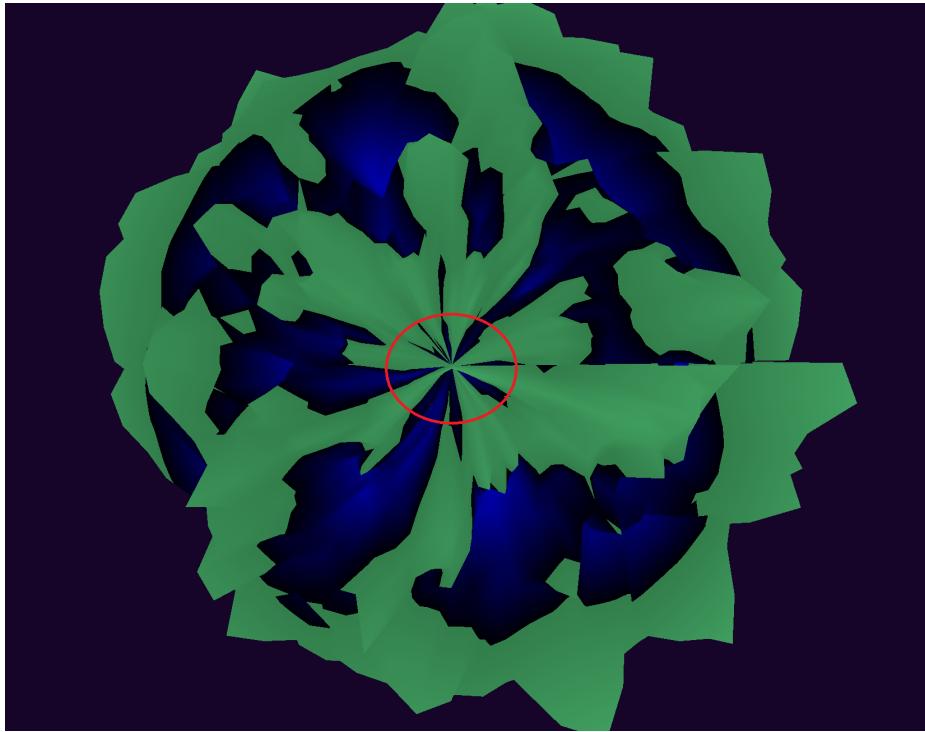


Figura 3: Deformación del ruido cerca de los polos de una instancia descartada de nuestro trabajo.

Si observamos los vértices de ambas geometrías en la siguiente imagen sacada de internet, podemos ver que esto tiene sentido, ya que la densidad de puntos en una esfera tradicional es mucho mayor cerca de los polos.

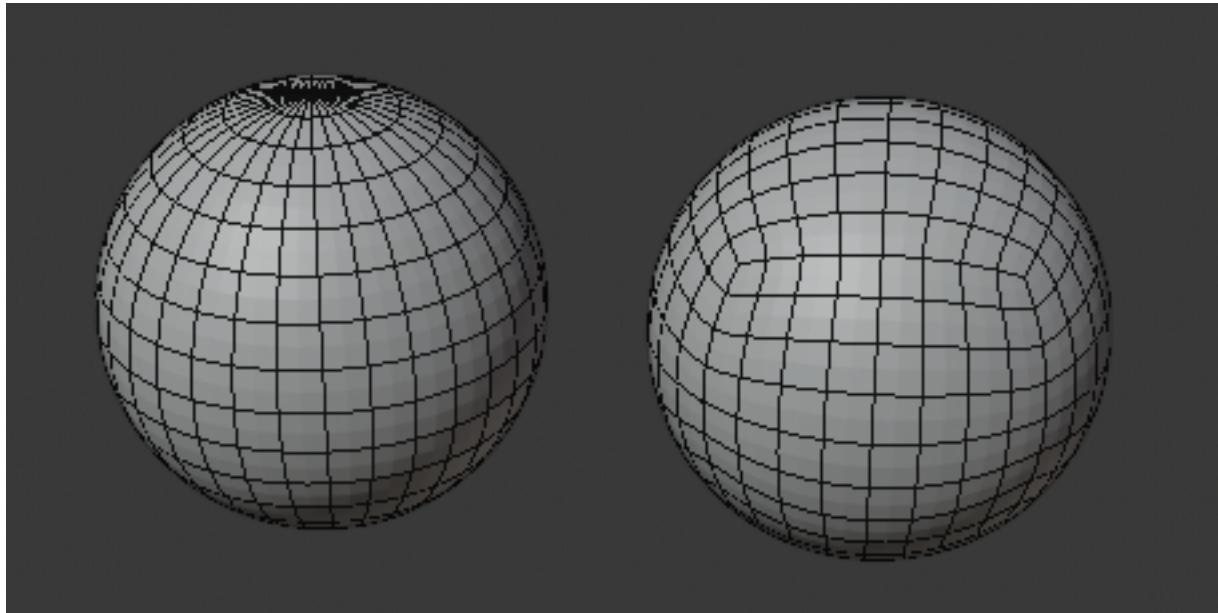


Figura 4: Topología de los vértices en cada geometría

El parámetro de la interfaz **World Size** corresponde a la cantidad de segmentos de cada cara del cubo construido, que se traslada directamente en una mayor cantidad de vértices y en consiguiente en una geometría con más apariencia esférica al normalizarlos.

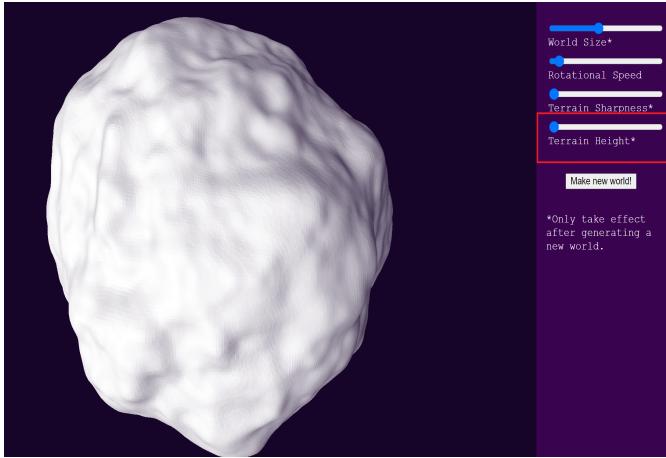
## 2.2. *Ruido de Perlin* para crear un terreno realista

El *Ruido de Perlin* es un tipo de ruido utilizado siempre que se quiera recrear algo aleatorio pero sin perder continuidad. Nosotros utilizamos el módulo de javascript Perlin Noise 3D<sup>1</sup>. Consultamos un recurso

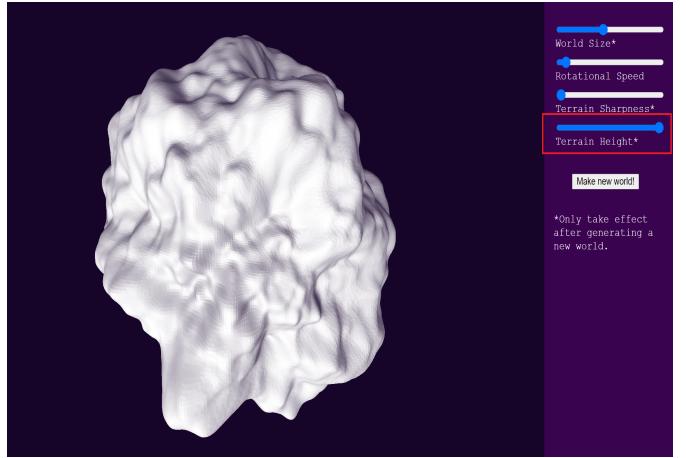
<sup>1</sup><https://www.npmjs.com/package/perlin-noise-3d>

para intentar una implementación bidimensional [2], pero nos pareció muy complicado hacerlo tridimensional (que era lo que necesitábamos) así que decidimos utilizar algo ya hecho.

En este punto introdujimos el parámetro **Terrain Height** en la interfaz para escalar el ruido obtenido, permitiendo la creación de terrenos más elevados.



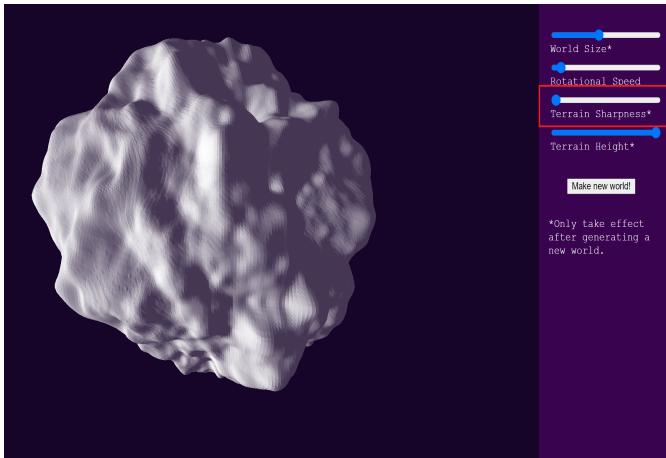
(a) Mundo con Terrain Height menor



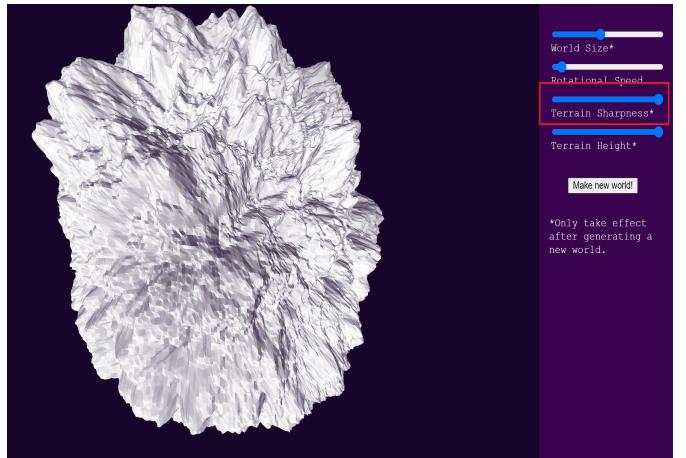
(b) Mundo con Terrain Height mayor

### 2.2.1. Fractional Brownian motion

El algoritmo de *Ruido de Perlin* suele ir acompañado de otro llamado *Fractional Brownian Motion* [2], que permite darle al terreno un aspecto más irregular, creando pequeños picos y valles pronunciados, pero respetando la continuidad general del algoritmo de Perlin. Hicimos una pequeña implementación propia, y controlamos la cantidad de iteraciones mediante el parámetro de **Terrain Sharpness**.



(a) Mundo con Terrain Sharpness menor



(b) Mundo con Terrain Sharpness mayor

### 2.3. Coloración y retoques finales

Decidimos crear un océano normalizando todos los vértices con altura menor a la altura del agua (que se determina aleatoriamente), pero dejando sus normales intactas para darle un aspecto de oleaje al océano. Esto lo realizamos en el *Vertex Shader*.

Aprovechamos el *Fragment Shader* para colorear los vértices de acuerdo a su altura, de menor a mayor: agua, arena, tierra y nieve. También creamos un punto único de iluminación (para que simule ser el Sol), y utilizamos en el agua y en la nieve reflexión de Blinn-Phong [3].

Además permitimos fijar una velocidad de rotación para el mundo, basándonos en lo hecho en un trabajo anterior de la materia.

### 3. Resultado Final

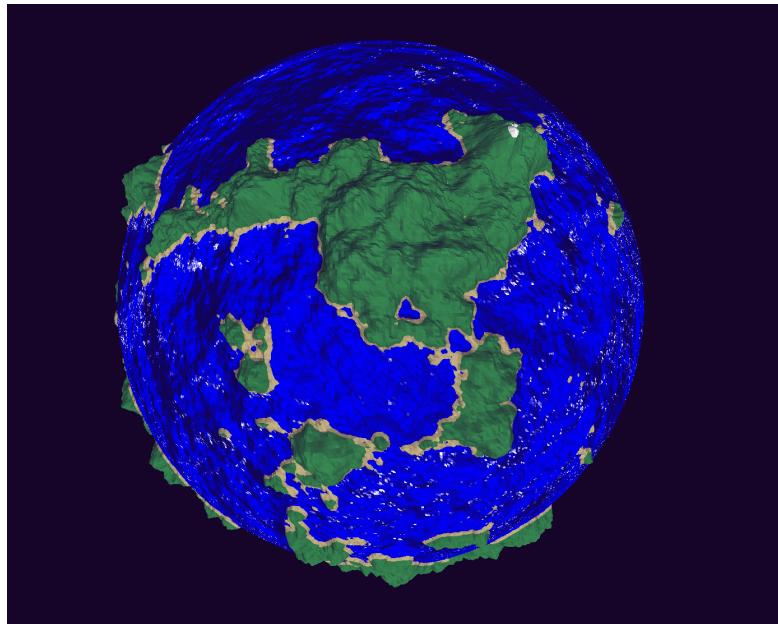


Figura 7: Resultado final.

## Referencias

- [1] Amir Emamjomeh. *Triangle List Generator for Sphere.*  
<https://www.codeproject.com/Articles/699565/Triangle-List-Generator-for-Sphere>
- [2] Raouf's Blog. *Perlin Noise: A Procedural Generation Algorithm.*  
<https://rtouti.github.io/graphics/perlin-noise-algorithm>
- [3] *Blinn-Phong reflection model.*  
[https://en.wikipedia.org/wiki/Blinn%20Phong\\_reflection\\_model](https://en.wikipedia.org/wiki/Blinn%20Phong_reflection_model)