

|  |
| --- |
| Generación procedural de terrenos 3D |
| Carlos Sánchez y Ángel Romero |
| Tutor: Samir Genaim |
| Trabajo de Fin de Grado  Grado en desarrollo de Videojuegos – Universidad Complutense de Madrid  2018/2019 |
|  |



# 

# Agradecimientos

# Resumen

# Palabras clave y siglas

# Índice

[Agradecimientos II](#_Toc530565441)

[Resumen II](#_Toc530565442)

[Palabras clave y siglas III](#_Toc530565443)

[1. Índice 1](#_Toc530565444)

[2. Introducción 1](#_Toc530565445)

[2.1 Breve introducción 1](#_Toc530565446)

[2.2 Objetivos 1](#_Toc530565447)

[2.3 Planteamiento 1](#_Toc530565448)

[2.4 Metodología 1](#_Toc530565449)

[3. Desarrollo 1](#_Toc530565450)

[4. Conclusiones 2](#_Toc530565451)

[5. Bibliografía 2](#_Toc530565452)

[6. Anexos 2](#_Toc530565453)

# Introducción

## 2.1 Antecedentes

## 2.2 Objetivos

## 2.3 Planteamiento

## 2.4 Metodología

# Desarrollo

 Esta parte se estructurará en aquellos capítulos o epígrafes necesarios para profundizar en el objeto de estudio y alcanzar los objetivos fijados.

## Algoritmo *midpoint* displacement

Este algoritmo es el primero en el que decidimos trabajar al comienzo del desarrollo dado que es, a priori, el más fácil de implementar.

[Explicación del algoritmo]

[Imagen representativa]

Experiencia en el desarrollo: es un algoritmo fácil de entender y el cuál se puede implementar en un tiempo considerablemente corto. Su primera implementación no nos dio muchos problemas. Los principales obstáculos con los que nos topamos surgieron dada la inexperiencia usando mapas de altura (heightmaps):

* Debíamos tomar la decisión del rango de valores que podía tomar un punto dentro del heightmap. En principio, decidimos que el punto más bajo del terreno seria 0, mientras que el más alto seria 1. Por lo tanto, el rango sería [0.0, 1.0]. Así, los valores estarían guardados en variables de punto flotante.
* Una vez completado el algoritmo, obtuvimos una matriz de float que representaba el mapa de altura. Ahora había que guardar estos datos en un archivo de extensión *.raw*. La primera idea era volcar toda la matriz de una vez en el archivo. Pero al hacer esto, nos dimos cuenta de que el archivo pesaba más bytes de los que debía, por lo que nos dimos cuenta de que no todos los valores estaban guardados de forma contigua, y esto provocaba el desfase de tamaño. La primera solución fue guardar los datos valor a valor.
* Cuando ya teníamos el archivo *.raw* con nuestro heightmap, buscamos la manera de dibujarlo. Unity pide que el archivo tenga una profundidad de 16 bit, lo que quiere decir que cada valor del mapa de altura debe ser de 2 bytes. Sin embargo, nuestros float eran de 4 bytes. Dado este problema, tuvimos que transformar los valores en coma flotante a enteros cortos (*short int)* que pesan 16 bits para que nuestro archivo fuese legible por Unity.

## Algoritmo Perlin Noise

[Introducción del algoritmo y creador]

Fue el segundo algoritmo que implementamos. Desde nuestra experiencia podemos apuntar varias cosas:

* Es un algoritmo que tiene varios años y del que existen 2 implementaciones, siendo la más moderna (*improved versión)* la que hemos decidido implementar por varias ventajas
* La mayoría de la documentación que existen en internet está en inglés, hay muy poca en español y si la hay es un tanto difusa.
* Vamos a explicar paso por paso el desarrollo del algoritmo ya que a nosotros nos costó entender el por qué de algunos pasos, para que así quede claro

Hablar del fractal brownian motion, que una modificación del perlin noise pero aplicándolo varias veces sobre un terreno pero con amplitudes y frecuencia distintas, haciéndolo así, más manejable.

Comentar que también, la mejor opción y uso del algoritmo es la combinación de múltiples perlin noise, con una mezcle de 3 ondas se consigue una decente representación de un paisaje.

[Explicación del algoritmo]

[Imagen representativa]

Experiencia en el desarrollo: como hemos dicho, es un algoritmo que nos parece complejo, ya que tiene diferencias sustanciales en cuanto al algoritmo de “midpoint displacement” y es, en principio, costoso de entender. Algunos de los problemas que nos surgieron:

* El proceso de entendimiento y comprensión del algoritmo fue más del que nos esperábamos, entre otras cosas por la introducción de conceptos como el ruido o técnicas como la interpolación.

## Algoritmo Voronoi

[Introducción del algoritmo y creador]

La siguiente implementación fue el diagrama de Voronoi. Existen diferentes algoritmos, pero el primero utilizado por nosotros ha sido el denominado como *fuerza bruta*.

Experiencia en el desarrollo: como muchos de los algoritmos que hemos utilizado, la documentación no es muy abundante y en su mayoría está en inglés. Nos servimos de varias fuentes (apuntadas en la solución de Visual Studio) para confluir conocimientos y llegar a dar con la implementación del algoritmo.

[Explicación de las diferentes implementaciones]

[Imagen representativa]

## Algoritmo de cortes

[Introducción del algoritmo]

Decidimos implementar este algoritmo simplemente para comprobar por nuestra propia cuenta que es mucho más lento que los demás y que el resultado no es lo suficientemente útil para utilizarse autónomamente. No obstante, nos sirve para añadir rasgos diferentes al heightmap.

Experiencia en el desarrollo: el punto más complicado fue decidir de qué forma (matemáticamente hablando) un punto en el plano está por encima o por debajo del corte que divide el plano en 2. Esto era importante porque afectaría al tiempo de ejecución del algoritmo. Las 2 formas con las que dimos fuero:

1. Calcular la distancia del punto a la recta, y ver si ésta distancia es negativa o positiva para evaluar la altura de ese punto.
2. Con la pendiente de la recta que corta el plano, encontrar la coordenada Y del punto de la recta con la misma coordenada X que el punto origen. (¿?)

[Explicación]

[Imagen representativa]

# Conclusiones

 En esta parte se presenta de manera concisa los resultados obtenidos después del estudio, elaboración y redacción del TFG.

# Bibliografía

 Se especificará la bibliografía utilizada. El TFG contará con un sistema uniforme de citación y con una bibliografía al final del escrito, ambos siguiendo el modelo Harvard (el utilizado por la Revista de la Facultad, Política y Sociedad).

# Anexos

 en su caso. material utilizado en el trabajo, demasiado extenso para ser incluido en nota, pero necesario para justificar resultados y conclusiones. Puede tratarse de tablas estadísticas, un conjunto de gráficos, legislación, cronología, etc.