



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TP Final: Generación de terrenos montañosos por elevación

Organización del computador II

Integrante	LU	Correo electrónico
Florencia Zanollo	934/11	florenciazanollo@gmail.com
Luis Toffoletti	827/11	luis.toffoletti@gmail.com



Índice

1. Introduccion	3
2. Implementación	4
2.1. Versión de C	4
2.2. Versión de ASM	4
3. Experimentación y Resultados	5
4. Conclusión y Mejoras	6

1. Introduccion

Para este trabajo práctico nos propusimos implementar el modelo para generación de terreno explicado en el paper "The Uplift Model Terrain Generator".¹

En él se propone la generación de terreno montañoso a partir de elevaciones o picos. El modelo se puede aplicar tanto en 2D como 3D. La idea general del algoritmo es generar picos de manera aleatoria y luego obtener la altura final de cada porción del terreno promediando las influencias provenientes de las elevaciones.

Nuestra meta es realizar dos implementaciones del modelo, una de ellas en C++ y la otra en ASM utilizando la tecnología SIMD. Para poder así demostrar cómo, utilizando SIMD, se puede acelerar el procesamiento.

¹<https://www.dropbox.com/s/q6brk3jqwppxrhx/upliftTerrainGenerator.pdf?dl=0>

2. Implementación

El método requiere de los siguientes datos de entrada:

divisions cantidad de divisiones sobre la cuál se colocarán los picos.

nroPeaks cantidad total de picos.

yMin altura mínima permitida de un pico.

yMax altura máxima permitida de un pico.

En nuestro caso también contamos con las siguientes entradas, las cuáles agregamos por razones de utilidad a la hora de experimentar y comparar las implementaciones de C y ASM.

seed permite setear una semilla particular para el random, esto es para lograr el mismo gráfico y poder experimentar con datos más certeros.

debugging si una semilla es proporcionada entonces se le puede decir al programa que entre en modo verbose; el cuál nos va a dar, además del gráfico, el valor numérico de cada posición del terreno final y la posición y tamaño original de cada pico.

Para poder llevar a cabo el método vamos a estar utilizando las siguientes estructuras:

peaksPos arreglo con **nroPeaks** posiciones, cada una contiene la posición dentro del terreno de dicho pico. Las posiciones van de 0 a **divisions-1**.

peaksSize arreglo con **nroPeaks** posiciones, cada una contiene la altura de dicho pico. Las alturas posible están entre **yMin** e **yMax** (inclusive en ambos casos).

terrain vector con los valores numéricos finales de cada posición del terreno.

Los picos son generados de manera aleatoria, tanto su posición como su altura (dentro de los límites explicados más arriba). Luego se hace lo siguiente:

```
for all positions do
  for all peaks do                                ▷ se calcula la influencia de cada pico para cada posición
    influencia ← altura del pico - distancia del pico a la posición actual * ruggness.
  end for
  if la posición no tiene influencia de nadie then
    valor final de la posición ← 0.
  else if la posición solo es influenciada por un pico then
    valor final de la posición ← influencia / 2.
  else if la posición es influenciada por dos o más picos then
    valor final de la posición ← influencia / cantidad de picos con influencia sobre ella.
  end if
end for
```

Quedando así en cada posición del terreno final el valor de 'promediar' todos los picos que tienen influencia sobre ella.

2.1. Versión de C

Está versión es muy sencilla ya que se puede mapear directamente el pseudocódigo a código, con pocas modificaciones. Luego basta con compilar utilizando las mejoras de sse para contar con las posibles mejoras SIMD que encuentre el compilador.

2.2. Versión de ASM

3. Experimentación y Resultados

4. Conclusión y Mejoras