IMT2112
Tarea 4

Elwin van 't Wout

November 10, 2020

Introducción

El conjunto de Mandelbrot es uno de los ejemplos más famosos de un fractal y es definido como todos los números complejos c tal que la serie $x_0 = 0$, $x_n = x_{n-1}^2 + c$, $n = 1, 2, \ldots$ permanece acotada en su valor absoluto. Debido al carácter fractal del conjunto, no es simple calcular su área de forma analítica. Por lo tanto, métodos numéricos son normalmente usados para aproximar su superficie. Uno de los cuales se llama *pixel counting*, lo cual es basado en simulaciones de Monte Carlo. Es decir, se puede generar K números complejos c_k y revisar para cada uno si la serie diverge o no. En seguido, la proporción de los números complejos adentro del conjunto de Mandelbrot es una aproximación de su superficie.

Actividades

Dado que el Mazinger no tiene un nodo de GPU, pueden usar tu propio computador. Primero, corre el código `opencl-devices.cpp` disponible en Canvas y copia el resultado al informe.

1. A pesar que existen librerías de C++ para la aritmética compleja, en este tarea usaremos una representación bidimensional para los números complejos. Es decir, cada número complejo debe ser representado por dos números de punto flotante, cada uno representa el parte real o imaginario del número complejo. Hay que programar las operaciones complejas a través de esta representación.

2. Expliquen tu algoritmo para calcular el area del fractal.

   (a) Mencionan como eligen los puntos c.

   (b) Mencionan como se puede determinar que un punto está adentro o afuera el fractal.

   (c) Se puede usar un máximo al número de iteraciones x_n, elegido de forma razonable.

3. Genera los puntos c de forma aleatoria.

   (a) En C++ se puede generar números aleatorios con la biblioteca `cstdlib`. La función `rand()` genera un número aleatorio entre cero y el valor máximo dado por el constante global `RAND_MAX`. Además se puede especificar el *seed* con `srand(0)`, lo cual genera resultados reproducibles.

   (b) Expliquen por qué no se puede paralelizar la generación de números aleatorios y, por lo tanto, no será eficiente hacerlo en la GPU. *Sugerencia:* la función es un '*pseudo* random number generator'.

4. Programa el algoritmo en OpenCL.

   (a) El cálculo de los valores x_n debe ser ejecutado en la GPU.

   (b) Calcular el superficie, es decir, la proporción de puntos adentro del fractal, debe ser ejecutado en la GPU. Expliquen como programaron esta operación de reducción.

   (c) Expliquen como programaron los *if-else statements* y si es eficiente hacerlo cada iteración.

5. Calcula el error de la aproximación. El valor esperado es 1,50659177.... Intenten de reducir el error lo más posible.

Evaluación

Entreguen el código de C++ y un informe corto, elaborado en LaTeX, con las respuestas a las preguntas.

Los reglamientos del curso se puede encontrar en Canvas.