Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona (FIB)

Trabajo de Fin de Grado

Jordi Gil González

Análisis del método de renderizado Path Tracing en CPU y GPU

Entrega 2: Planificación temporal

Departament de Ciències de la Computació

Director del Trabajo de Fin de Grado:	Chica Calaf, Antoni
Tutor de GEP:	Barrabes Naval, Fernando
Programa de estudio:	Computación
Especialización:	Computación Gráfica

Curso Académico 2019/2020

26 de septiembre de 2019

Índice general

*

CAPÍTULO 1

PLANIFICACIÓN TEMPORAL

1.1. Planificación y programación

Este capítulo trataremos la planificación con el objetivo de describir las tareas que serán llevadas a cabo con el fin de realizar el presente proyecto, brindando de esta forma una plan de acción que defina las diferentes acciones para el cumplimiento del proyecto en el tiempo estimado. Veremos también los recursos utilizados y se tendrán también en cuenta posibles obstáculos que puedan modificar la planificación.

El proyecto inició a principios de Julio de 2019 con fecha límite el 13 de Enero de 2020.

1.2. Descripción de tareas y recursos utilizados

1.2.1. Descripción de tareas

Estudio de conceptos

Antes de comenzar el proyecto, fue necesario familiarizarse con los conceptos básicos que rigen nuestro proyecto. Durante el semestre previo hubo un para de reuniones con el director del proyecto para definir el tema en el cual se centraría y poder así encaminarlo. También, el autor del presente proyecto se matriculó de la asignatura de Tarjetas Gráficas y Aceleradores (TGA por sus siglas) para introducirse en el entorno de `CUDA` y así agilizar

el proceso de desarrollo del proyecto en el momento de su inicio. Dado que esta tarea se realiza fuera del proyecto y es difícil de determinar una estimación en horas, no se tendrá en cuenta.

Configuración del sistema

Antes de entrar en el desarrollo en si del proyecto, debemos configurar las herramientas necesarias y probar que funcionan correctamente.

Para poder desarrollar el proyecto será necesario tener instalado en nuestros sistemas la API de `CUDA` con tal de poder crear el programa principal de nuestra aplicación. Las librerías de `OpenMP` vienen instaladas por defecto en los sistemas Linux por lo que no será necesaria una instalación, pero si comprobar que funcionan correctamente. Por último paras uso de una plataforma, *TexMaker*, que nos permita compilar el código LaTeXcon tal de generar la documentación.

Esta configuración debe hacerse tanto en el computador de sobremesa como en el portátil. La parte de *CUDA* en el cluster de docencia no ser'a necesario de configurar debido a que ya est'a previamente configurado por el DAC (Departament d'Arquitectura de Computadors).

Planificación del proyecto

Esta es la tarea que se está desarrollando actualmente. Esencialmente trata sobre todo el contenido cubierto por el curso de GEP. La podemos dividir en tres sub-tareas:

1. Contextualización y alcance.
2. Planificación temporal.
3. Presupuesto y sostenibilidad.

Desarrollo

Esta tarea es la más importante de todo el proyecto. Cubre tanto el desarrollo de la aplicación, experimentación y documentación de los resultados de cada una de las partes. Podemos dividir el desarrollo en tres etapas:

- **Desarrollo versión secuencial:** Se trata de la versión más básica de todas. Sirve como base para desarrollar tanto la versión paralela en CPU (`OpenMP`), como la versión paralela en GPU (`CUDA`).

- **Desarrollo versión paralela CPU:** Una vez implementada la versión secuencial pasaremos a implementar una versión paralela de esta haciendo uso de la librería de `OpenMP`.
- **Dearrollo versión paralela GPU:** Una vez implementada la versión secuencia pasaremos a implementar una versión paralela en `CUDA`.
- **Experimentación:** Crearemos la misma escena en cada una de las versiones comentadas en los puntos anteriores y analizaremos el rendimiento de ambas, teniendo en cuenta el tiempo de creación requerido para generar la imagen final.

Estas etapas no son secuenciales. Es decir, a medida que vayamos desarrollando la aplicación secuencial, antes de añadir características importantes, se implementará tanto la versión de CPU como la versión de GPU. Es por eso que cada una de las etapas se realizarán más de una vez en el curso del desarrollo del proyecto.

Cada vez que implementemos una nueva versión de cada una de las aplicaciones, éstas serán testeadas y comparadas entre si. De esta forma podremos ir haciendo los experimentos pertinentes y ver así como es el rendimiento de nuestras aplicaciones con las distintas características que iremos añadiendo. Un ejemplo de ello es comparar como afecta al rendimiento el uso de estructura de datos aceleradoras (en todas las versiones), frente a una versión en la cual no se hace uso de este tipo de estructuras de datos y así justificar el uso de ellas.

Las dependencias entre ellas son fáciles de describir. Hasta no tener la primera versión secuencial no podremos empezar a programar las versiones paralelas. Tampoco podremos añadir mejoras a la versión secuencial sin tener implementadas las versiones paralelas pertinentes.

Etapa final

En esta etapa realizaremos toda la documentación de las etapas anteriores y preparación de la presentación de la defensa del presente proyecto.

1.2.2. Tabla resumen

1.2.3. Recursos utilizados

Podemos dividir los recursos necesitados para la realización de las tareas en: hardware y software. A continuación tenemos ambas listas y las tareas en las cuales son utilizadas.

Tarea	Tiempo empleado (horas)
Configuración del sistema	10
Planificación del proyecto	90
Desarrollo	380
Etapa final	50
Total	530

Tarea	Tiempo empleado (horas)
Desarrollo secuencial	80
Desarrollo CPU	90
Desarrollo GPU	105
Experimentos	105
Total	380

Software

- Linux, usado en todas las tareas.
- LaTeX, usado para realizar la documentación.
- `C++`,`OpenMP` y `CUDA`, usado en el desarrollo de la aplicación.
- `git`, usado para el control de versiones y compartir de código entre diferentes entornos.

Hardware

- PC Sobremesa con las siguientes características:
  - i7 7700 3.6 GHz
  - NVIDIA GeForce RTX 2080 SUPER
  - 24GB RAM
  - 250 SSD, 1TB SSD, 1TB HDD
- PC Portátil con las siguientes características:
  - i7 7700HQ 2.8GHz
  - NVIDIA GeForce 1050 Mobile
  - 8GB RAM

    - 500GB SSD
- Cluster Docencia
    - Intel Xeon E5-2620 v2 2.10GHz x2
    - NVIDIA Tesla K40c x4
    - 64GB RAM
    - 1TB x2

1.3. Diagrama de Gannt y Estimación

1.4. Gestión de riesgos: Planes alternativos y obstáculos

BIBLIOGRAFÍA

[1] R. Beau Lotto, S. Mark Williams, and Dale Purves. Mach bands as empirically derived associations. *Proceedings of the National Academy of Sciences of the United States of America*, 1999.

[2] Gouraud Henri. Continuous Shading of Curved Surfaces. *IEEE Transactions on Computers*, 1971.

[3] Bui Tuong Phong. Illumination for Computer Generated Pictures. *Communications of the ACM*, 18(6):311—-317, 1975.

[4] Turner Whitted. An Improved Illumination Model for Shaded Display. *Communications of the ACM*, 1980.

[5] Arthur Appel. Some techniques for shading machine renderings of solids. 1968.

[6] James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1986*, 1986.

[7] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1986*, 1986.

[8] Eric P. Lafortune and Yves D. Willems. Bi-Directional Path Tracing. *Proc. SIGGRAPH*, 1993.

[9] Henrik Wann Jensen. Global Illumination using Photon Maps. 1996.

[10] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997*, 1997.

[11] Peter Shirley. *Ray Tracing in One Weekend*. 2018.

[12] Peter Shirley. *Ray Tracing : The Next Week*. 2018.

[13] Peter Shirley. *Ray Tracing: The Rest of Your Life*. 2018.

[14] Steven M. Rubin and Turner Whitted. A 3-dimensional representation for fast rendering of complex scenes. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1980*, 1980.

[15] Tero Karras. Maximizing parallelism in the construction of bvhs, octrees, and k-d trees. In *High-Performance Graphics 2012, HPG 2012 - ACM SIGGRAPH / Eurographics Symposium Proceedings*, 2012.

[16] Tero Karras and Timo Aila. Fast parallel construction of high-quality bounding volume hierarchies. In *Proceedings - High-Performance Graphics 2013, HPG 2013*, 2013.

[17] James F. Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1977*, 1977.

[18] Warren Hunt and William R. Mark. Ray-specialized acceleration structures for ray tracing. In *RT'08 - IEEE/EG Symposium on Interactive Ray Tracing 2008, Proceedings*, pages 3–10, 2008.