FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

Grau en Enginyeria Informàtica

Especialització en Computació

Anàlisi del mètode de renderització Path Tracing en CPU i GPU

Entrega 1: Context i abast del projecte

Director:
Chica Calaf, Antoni
DEPARTAMENT DE CIÈNCIES DE LA COMPUTACIÓ

Autor:
Gil González, Jordi

Tutor de GEP:
Barrabes, Fernando

Any Acadèmic 2019/2020

21 de setembre de 2019

Índex

*

1 Introducció

En l'actualitat les imatges generades per ordinador són molt presents en el nostre dia a dia, ja sigui en un entorn de treball o un de lúdic. La creació d'imatges realistes mitjançant l'ús de computadores s'ha convertit en una necessitat. Indústries com el cinema o els videojocs requereixen d'algorismes capaços de poder reproduir el món real en un entorn virtual i en el menor temps possible.

El principal propòsit d'aquest projecte es veure com podem explotar de forma efectiva el hardware que tenim i fer un anàlisis comparatiu del rendiment d'un algorisme capaç de genera imatges realistes entre una CPU i una GPU.

En computació gràfica hi ha un conjunt de mètodes que permeten generar imatges realistes: *Ray Tracing* [1], *Path Tracing* [2], *Bidirectional Path Tracing* [3], *Photon Mapping* [4], *Metropolis light Transport* [5], entre d'altres. A excepció del *Ray Tracing*, la resta de mètodes llistats tracten d'aproximar la *Rendering Equation* presentada per Kajiya et al. [2].

En aquest projecte ens centrarem en la implementació del *Path Tracing*.

Ens plantegem tres grans reptes:

1. Cerca de tècniques d'optimització
2. Implementació per CPU i GPU de les tècniques vistes al punt 1
3. Anàlisi i comparació dels resultats obtinguts

2 Contextualització

2.1 Context

La renderització d'imatges realistes és una àrea de gran interès en el camp de la computació gràfica. Un dels principals objectius d'aquesta és ser capaços de renderitzar imatges que siguin indistingibles de les del món real, com ara fotografies. Per això és molt important poder reproduir el comportament de la llum en un entorn virtual. Per tal d'obtenir aquest realisme és necessària la il·luminació global, és a dir, hem de ser capaços de poder simular la llum directa (llum que incideix de forma directa a un objecte des del focus de llum) i la llum indirecta (llum que incideix a un objecte dels rebots d'altres objectes de l'escena).

L'algorisme de *Ray Tracing* és un dels mètodes més populars d'aquest conjunt. És tracta d'una tècnica que consisteix en traçar rajos des del focus de llum (*Forward Ray Tracing*) als objectes de l'escena, o des de l'ull, o càmera virutal, (*Backward Ray Tracing*) als objectes de l'escena. Aquest algorisme és molt sensible al nombre de polígons d'una escena, a més complexa és una escena més ineficient és l'algorisme. Tot i proporcionar un alt grau de realisme, no és capaç d'aconseguir un efecte foto-realista degut a que no té en compta la il·luminació indirecta. Per tal d'aconseguir aquest efecte de foto-realisme hem de tenir presenta tant la il·luminació directa com la indirecta, per això és va presentar la *Rendering Equation* [2].

L'algorisme de *Path Tracing* és una millora del *Ray Tracing*. L'objectiu d'aquest, és donar una aproximació de l'equació de renderització per tal de resoldre la il·luminació global és per això que l'algorisme per mitjà de la integració de Montecarlo tracta d'aproximar la *Rendering Equation*. Gràcies a resoldre la *RE* és capaç de forma implícita reproduir efectes naturals com ombres suaus, *motion blur*, *ambient oclussion* i il·luminació indirecta entre d'altres sense necessitat de fer-ho manualment nosaltres. Aquest algorisme és indiferent al nombre de polígons de l'escena però per a que l'algorisme convergeixi el nombre de mostres per cada píxel és molt elevat i això provoca que aquest algorisme no es pugui utilitzar en aplicacions en temps real com ara videojocs.

Degut a que cada píxel és independent, tenim un algorisme amb una alta capacitat de paral·lelització. Gràcies això podem fer ús dels diferents cores que componen una CPU i una GPU per calcular més d'un pixel a la vegada i així millorar el rendiment de la nostra aplicació.

2.2 Stakeholders

En aquest secció presentarem els diversos actors implicats en un projecte, els quals descriurem en les següents seccions.

2.2.1 Desenvolupador

Aquest actor és l'encarregat de realitzar el pla del projecte, recerca de la informació, documentació, desenvolupament del software requerit, solució de possibles problemes/obstacles, i les proves i anàlisis dels experiments. Aquest actor ha de treballar conjuntament amb el director, i codirector i/o ponent en el cas d'haver-hi i és la última persona encarregada de complir els terminis establerts.

2.2.2 Director del projecte

Aquest actor és l'encarregat de guiar al desenvolupador en cas de dificultats així com en l'assessorament de possibles solucions.

2.2.3 Usuaris beneficiats

Tot i que aquest projecte no té la intenció de crear un producte, no vol dir que no hi hagin beneficiaris. Optimitzar i adaptar un algorisme per veure el seu rendiment en diverses plataformes pot ser útil per investigadors en el camp de la computació gràfica.

3 Justificació

4 Abast del projecte

Per tal de solucionar el problema presentat en el nostre projecte necessitem un programa que sigui capaç de generar imatges realistes. A [6], [7] i [8] es presenten les bases per crear un Path Tracing. A partir d'aquesta base estendrem la nostra aplicació a un versió paral·lela en CPU i GPU. S'aplicaran també diverses tècniques d'optimització utilitzades en aquests tipus de mètodes com ara l'ús d'estructures de dades acceleradores [9]. L'estructura acceleradora que utilitzarem per representar internament la nostra escena serà en una *Bounding Volume Hierarchy*. La BVH és una estructura de tipus arbre on tots els objectes de la escena estan representats per la seva capsa englobant a les fulles de l'arbre. Cada node intermedi

4.1 Abast

En aquest apartat es presenten els diferents objectius i possibles obstacles del projecte.

4.2 Objectius

L'objectiu principal d'aquest projecte és implementar una aplicació que donada una escena, renderitzar imatges mitjançant *Path Tracing* i analitzar el rendiment que ens dóna en una aplicació paral·lela fent ús de la CPU i la GPU i fer un anàlisis sobre aquest.

La gestió de la memòria en una aplicació paral·lela, ja sigui en CPU o en GPU és molt important i tenir una mala gestió d'aquest pot fer que el rendiment de la nostre aplicació no sigui l'esperat. Per tal de garantir una bona gestió de memòria, definirem un altre objectiu secundari on s'estudiaran quines són les millors pràctiques.

Com a sub-objectius tenim:

1. Implementar de forma eficient l'algorisme presentat a [6] i [10]
2.
3.

4.3 Obstacles i riscos del projecte

Els temes principals que tractem al projecte com ara la implementació del *Path Tracing* estan molt estudiats i desenvolupats. No obstant, això no implica que el desenvolupament del projecte sigui un camí senzill, son molts els obstacles als quals podem enfrontar-nos.

4.3.1 Programa principal

Com bé hem comentat en la secció d'objectius, la gestió de la memòria és un tret molt important. Una mala gestió de la memòria pot provar errors en la nostre aplicació, fent que no sigui possible generar les imatges representades i això és un greu problem

4.3.2 Algorisme utilitzat

L'algorisme que utilitzem calcula el color a partir de la intersecció dels rajos amb els objectes de l'escena i aquest és un punt molt important. El càlcul d'intersecció dels rajos amb les diferents superfícies que conformen els objecte han d'estar implementats de la forma més eficient possible degut a que són operacions que s'han de dur a terme milers de milions de vegades en la creació d'una imatge. Tenir una pobre implementació pot afectar negativament en el rendiment de l'aplicació i tenir un impacte en el temps per fer les proves.

5 Metodologia i rigor

5.1 Eines de desenvolupament

El desenvolupament de la nostra aplicació es durà a terme utilitzant C++, *OpenMP* i *CUDA*. *OpenMP* és una API dissenyada per afegir concurrència a programes escrits en C, C++ i Fortran. El principal avantatge d'utilitzar aquesta API és que és multiplataforma i per tant el mateix codi ens serveix per *Linux*, *Microsft* o *MacOS* i ens permet crear fàcilment aplicacions paral·leles en una CPU. *CUDA* és una plataforma de computació paral·lela i una API desenvolupada per NVIDIA que ens permet accedir al conjunt d'instruccions i elements paral·lels de còmput de les GPU de NVIDIA per a l'execució de *kernels*. Un gran avantatge d'aquesta API és la seva accessibilitat en contraposició d'altres APIs com per exemple *OpenGL* o *DirectX*.

5.2 Validació de resultats

Referències

[1] Turner Whitted. An Improved Illumination Model for Shaded Display. *Communications of the ACM*, 1980. ISSN 15577317. doi: 10.1145/358876.358882.

[2] James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1986*, 1986. ISBN 0897911962. doi: 10.1145/15922.15902.

[3] Eric P. Lafortune and Yves D. Willems. Bi-Directional Path Tracing. *Proc. SIGGRAPH*, 1993. ISSN 1098-6596. doi: 10.1017/CBO9781107415324.004.

[4] Henrik Wann Jensen. Global Illumination using Photon Maps. 1996. doi: 10.1007/978-3-7091-7484-5_3.

[5] 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. ISBN 0897918967. doi: 10.1145/258734.258775.

[6] Peter Shirley. *Ray Tracing in One Weekend*. 2018. URL `https://github.com/RayTracing/InOneWeekend`.

[7] Peter Shirley. *Ray Tracing : The Next Week*. 2018. URL `https://github.com/petershirley/raytracingthenextweek`.

[8] Peter Shirley. *Ray Tracing: The Rest of Your Life*. 2018. URL `https://github.com/RayTracing/TheRestOfYourLife`.

[9] 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. ISBN 9781424427413. doi: 10.1109/RT.2008.4634613.

[10] 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. ISBN 9783905674415. doi: 10.2312/EGGH/HPG12/033-037.