**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ**

**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ДИПЛОМНА РАБОТА**

Тема: Реализация на RayTracer

Дипломант: Научен ръководител:

Николай Дионисов Светлин Младенов

СОФИЯ

2015

**Увод**

Рендерирането е фундаментален компонент от компютърната графика. Рендериране се нарича процеса по превръщане на описание на три дименсиална сцена в изображение. Алгоритмите за анимация, моделиране, текстуриране и други части от компютърната графика трябва да преминат през някакъв вид рендериращ процес за да могат да бъдат превърнати в видими изображения. Техниките за рендериране са разделени в две основни групи – растеризиране и тресиране на лъчи.

Растеризирането е метод използван в интерактивни приложения, който обхожда всички видими примитиви във сцената и геометрично ги прожектира върху платно. Растеризирането само по себе си не дава много добри резултати откъм фотореализъм, но е значително по-бързо от всички останали методи. Поради тази причина това е най-разпространенят метод в гейм индустрията. Видео картите имплементират работят по този начин.

Методите за синтез на изображение базирани на Тресирането на лъчи или „Ray tracing“ са вдъхновени от реалния физичен свят. Те са симулация на света около нас. Работата на рей трейсъра се изразява в проследяването на лъчи минаващи през виртуален филм. При прекъсването на лъча от обект във сцената, част от лъча рефлектира, а друга се абсорбира. Типично за разработката на рей трейсъри се използват снимки и компютърно пресъздадени сцени от снимките. След това резултатите от програмата се сравняват с оригиналните снимки. Основният проблем на този метод е дългото време на генериране на изображение. Рей трейсинга е предпочитаният метод за рендериране, там където има възможност по-голяма част от работата да се свърши предварително.

В тази дипломна работа е представена примерна имплементация на RayTracer, тип PathTracer.

**ПЪРВА ГЛАВА: МЕТОДИ И ТЕХНОЛОГИИ ЗА РЕАЛИЗИРАНЕ НА RAY TRACER**

1. **Основни принципи**

За синтезиране на изображения ни трябва някакъв модел на светлината. Предбладаващите модели на светлината са геометричните и физичните. Геометричната оптика моделира разпространението на светлина като лъчи, които се движат в прави линии и техните пътища са представени като серия от рефлекции и рефракции. Физиката гледа на разпространието на светлината като вълнов феномен, който е повлиян от ефекти на поляризация, смущения и дифракциии.

* 1. **Основни типове рей-трейсъри**

Класическият рей трейсър, използващ геометричния модел, взима проби от идващата светлина от сцената, като пуска лъчи през съответните точки на филма и ги проследява до източниците на светлина. Изображенията генерирани от имплементацията на класическият рей трейсър може да изглежда добре, но също и нереалистично. Причината за това е, че този алгоритъм пренебрегва голяма част от домейна на интегралното уравнение, което описва сложната интеракция на светлината с обектите.

Разпределеният рей трейсър, усъвършенства класическият рей трейсър като използва Монте Карло техники. Вместо да използва резултата от само един вторичен лъч, той разпределя няколко вторични лъча за да може по-точно да пресметне нужните интеграли. Този подход е много по-скъп от страна на процесорно време, но е нужен за ефекти като меки сенки, depth of field, motion blur и други. Въпреки подобренията, които разпределеният рей трейсър внася, те не решават напълно проблема за фотореалистична графика. [1]

За да имаме реалистични изображения, трябва да взимаме в предвид и светлината отразена от останалите обекти в сцената. Този ефект в компютърната графика се нарича глобална илюминация и се противопоставя на локалната илюминация, чиито представители са представените до тук типове рей трейсъри.

Пътищния рей трейсър е разработен като решение на рендериращото уравнение и е силно базирано на Монте Карло техники. Той разглежда целия домейн на уравнението, докато разпределеният рей трейсър – само част от него. Този метод пуска не един, а много първични лъчи прези всяка една точка на филма, след което пробаблистично решава в каква посока да го отрази. Рекурсията в този метод е ненужна. Основния проблем на Пътищния рей трейсър е, че разликите в крайния резултата се възприемат като шум. При ползването на повече лъчи през всеки пиксел, покриваме по голяма част от домейна, изичисляваме интеглала с по-малка грешка и шума нямалява. Това обаче струва много прецесорно време – трябва да увеличим пробите 4 пъти, за да намалим шума двойно.[2] Възможно е да се използва техника наречена „Photon mapping“ за намаляването на шума, но тя не се разглежда тук.

* 1. **RayTracer tasks**

За да извършат задачите си, рей-трейсърите типично имат следните компоненти:

1. Камери – Как и от къде се разглежда сцената? Камерите генерират първичните лъчи от изгледната точка към сцената.
2. Лъч-обект сечение – Трябва да знаем къде даден лъч ще пробие геометричен обект. Също така за тази точка трябва да знаем нормалата и материала на обекта.
3. Разпределение на светлината – Трябва да се моделира разпространението на светлината в сцената. Това включва позиците на източниците на светлина и по какъв начин се разпространява тяхната енергия в пространството.
4. Видимост – За да разберем дали даден източник на светлина достига дадена точка на повърхност, трябва да знаем дали между тях има непрекъснат път.
5. Материали – Всеки обект трябва да предоставя описание на своя външен вид като информация за това как светлината реагира с повърхността на обекна.
6. Рекурсивност – Понеже един лъч светлина може да е бил рефлектиран няколко пъти преди да достигне дадена точка. Понякога е нужда да бъдат преследени повече от един вторичен лъч за да се прихванат някои ефекти. Рефлектираните лъчи ще бъдат наричани вторични, а тези идващи директо от камерата - първични.
7. Разпространине на лъчите – Кога един светлинен лъч пътува в вакуум през пространството, неговата енергия остава константа на всяка една точка от него. Изображенията които типично бихме искали да занемем не са във вакуум и ни трябват модели по който да отнемаме от енергията за до създадем ефекти като мъгла, пушилка или просто атмосферата на земята. [3]
   1. **The rendering equation**

Цялата теорията върху която е изграден рей трейсингът е бязирана на т.н. “rendering equation”. Рендериращото уравнение, формулирано от Джеймс Каджия през 1986, описва движението на светлината от една точка върху повърхност до друга като сума на излъчен и рефлектиран радианс.

Където:

х е гочката на повърхността

посоката бягаща от повърхността

е излъчваният радианс

е рефлектираният радианс

Радиансът ни казва колко светлинна енергия пристига или напуска дадена повърхност през дадена единица време. В вакуум радианса е константа за всяка точка по даден лъч. Това прави рей трейсинга възможен.

Рефлектираната светлина може да се опише по следния начин:

Този интеграл взима в предвид цялата пристигаща светлина и изчислява рефлектинарата. Той взима в предвид и светлината отговаряща за глабална илюминация. Елементите в това уравнение са двупосочната рефлектираща функция, идващата свелина от дадена посока и косинусът от ъгъла между посоката от ноято идва текущия лъч и нормалата на повърхността. Двупосочната рефлектираща функция или „BRDF“ е функцията която дефинира материала на повърхността. Тя приема посоките на изходния и входния лъч и връща даден спектър – цветовете които ще бъдат отразени. Функцията може да бъде представена като 4 дименсиална функция от изходните и входните посоки.

[IMG]

За да може двупосочната рефлектираща функция да е физически възможна, тя трябва да спазва закона за запазвана на енергията и да се подчинява на резипрочния принцип на Хелмхолц[?]. Това значи че функцията винаги трябва да има резултат между 0 и 1. За да сме по-точни, диферециалната функция интегрирана върху хемисфера трябва да бъде по малка или равна на едно. Това означава, че не можем да рефлектираме повече светлина от колкото сме получили. Това може да бъде представено математически както следва:

[IMG]

Реципрочния принцип на Хелмхолц означава, че пробите от идващите и рефлектираните посоки от двупосочната рефлектираща функция могат да се обърнат и резултата ще остане същият.

[IMG]

Намирането на BRDF може да стане по няколко начина. Първият е чрез емпирически измервания и вместване на резултатите в математически функции. Премери за този метод са модели като Ламберт, Фонг и Блин-Фонг. Функцията за дифузен материал(ламберт) е просто константа.

Функцията Li описва пристигащия радианс в дадена точка. Тя може да бъде дълбоко рекурсивено интегрално уравнение, защото част от светлината може да идва индиректно - рефлектирана от други повърхностти. [4]

1. **Технологии за разработване на рей-трейсъри**

Технически най-предизвикателната част от рей-трейсъра е сечението на лъчи с обекти. Това се дължи на факта, че то изиства сложни структури за съхранение в пространтвото, рекурсивно обхождане и много процесорно време. Представени са няколко имплементации извършващи тази работа.

* 1. **Embree**

Това е колекция от високо производителни рей трейсинг кърнели разработени от Интел. Кърнелите са оптимизирани за фотореалистично рендериране на последните Интелски процесори с поддръжка на SSE, AVX и AVX2. Ембрее съдържа алгоритми за некохерентни и кохерентни лъчи. Поддържат се и динамични сцени благодарение на дву-стъпалните индексни конструиращи алгоритми. [5]

* 1. **OptiX**

Фреймуорк за създаване на рей трейсиг приложения вървящи върху видео картите на NVidia. OptiX не отраничава продуктите базирани на него само до рендериращи такива. Той позволява на разработчиците да свършат бързо всяка задача, която изиства рей трейсинг. [6]

* 1. **OpenCL**

Стандарт, дефиниращ програмен интерфейс за паралелни изчисления. Имплементиран е от всички водещи производители на хардуерни изчислителни компоненти. Код написан за OpenCL може да бъде изпълнен от прецесори, видео карти и други. Това е изключително полезно за рей трейсърите като едни от най-гладните за изчислителна мощност приложения. [7]

* 1. **OpenMP**

Рей трейсърите трябва да използват всички налични ресурси. Въпреки че OpenCL е изключителено мощен инструмент, той изисква начана инвестиция от имплементатора – да се запознае с стандарта, да създаде изчислителните кърнели и т.н. От друга страна е OpenMP, която е библиотека вградена във всички модерни C++ компилатори и платформи. Тя предоставя лесно използваеми примитиви за паралелизиране на кода върху всички процесорни ядра. [8]

1. **Съществуващи решения и реализации**
   1. **PBRT**

PBRT е примерна имплементация на рей трейсър идваща с книгата „Physically based rendering“. Въпреки учебният си характер, това е напълно функционален и използван рей трейсър. На него са базирани много други продукти сред които са LuxRender и Mitsuba. Рей трейсърът е силно обектно ориентиран, използва всички налични ядра, поддържа volume rendering и path tracing.

* 1. **POV-Ray**

Pov-Ray е един от първите рей трейсъри, развиващ се и до днес. Първоначалната имплементация е вървяла на Амига компютри и е била базирана на DKBTrace. В момента поддържа photon mapping, специфичен език за описване на сцените, няколко вида светлини, атмосферни ефекти, рефлекции, рефракции и много други. Примерното изображение от Janet Lowry, рендерирано чрез POV-Ray.

* 1. **YafaRay**

ВТОРА ГЛАВА

**ПРОЕКТИРАНЕ НА СТРУКТУРАТА НА RAY TRACER**

1. **Функционални изисквания към рей трейсъра**

Използвана литература

[1] <http://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/realistic_raytracing.html>

[2] 4 пъти – 2 по-малко

[3] Physically Based Rendering - http://www.pbrt.org/

[4] Рекурсивен equation

[5] <https://embree.github.io/>

[6] https://developer.nvidia.com/optix

[7] https://www.khronos.org/opencl/

Unsorted

Стандартният рей трейсър генерира по един начален лъч през всеки пиксел. При прекъсване на всеки един от лъчите, алгоритъма рекурсивно генерира нови 2 лъча. Първия е перфектно огледално рефлектираната част от лъча, а другия предадената от него част в самия обект. Посоката на вторият лъч обикновенно е събръзена с закона на Снел[\*]. Закона на Снел гласи n_1\sin\theta_i = n_2\sin\theta_t \quad където n1 и n2 са коефициентите на пречупване на сътветните среди, а sin0i sin0t са косинусите на ъглите под които падат лъчите.