УДК 004.922, 004.925, 004.415.2
П.А.Грицан, Г.В.Кіт
Луцький інститут розвитку людини ВМУРоЛ «Україна»

ТРАСУВАННЯ ПРОМЕНІВ З ВИКОРИСТАННЯ ТЕХНОЛОГІЇ NVidia CUDA

В статті було проаналізовано можливість створення програми генерування зображень на базі підходу трасування променів з використанням технології NVidia CUDA.

Ключові слова: *трасування променів, CUDA, рендеринг, паралельні обчислення.*
Рис.2. Літ. 5.

Вступ

Трасування променів (англ. ray tracing) – один з найпотужніших методів комп'ютерної графіки для побудови зображень. Із його допомогою можна отримати безліч ефектів, таких як заломлення (refraction) і відображення (reflection), бамп-меппінг (bump-mapping), ефект розмиття глибини (depth of field) і багато інших.

Метод трасування променів математично складний, але простий концептуально. Що реально робить комп'ютер, так це відстежує промені світла, що йдуть від джерела в око. Коли випускаються промені наче з ока до джерела замість того, щоб йти від джерел до ока, то це називається зворотнім трасуванням (на відміну від першого - прямого). Зворотне трасування більш ефективне, так як гарантує, що число променів, що досягли очей спостерігача, буде точно таким же, як число пікселів зображення.

Основні переваги:

- можливість візуалізації гладких об'єктів без апроксимації їх полігональних поверхнями (наприклад, трикутниками);
- обчислювальна складність методу слабо залежить від складності сцени;
- висока алгоритмічна розпаралеленість обчислень – можливість паралельно і незалежно трасувати два і більше променів, розділяти ділянки (зони екрану) для трасування на різних вузлах кластера і т.д.;
- відсікання невидимих поверхонь, перспектива і коректне відображення поля зору є логічним наслідком алгоритму.

Серйозним недоліком методу трасування променів є продуктивність. Метод растеризації та сканування рядків використовує когерентність даних, щоб розподілити обчислення між пікселями. У той час як метод трасування променів кожен раз починає процес визначення кольору пікселя заново, розглядаючи кожен промінь спостереження окремо.

Постановка проблеми

Сучасні графічні процесори можуть паралельно виконувати тисячі потоків, забезпечуючи при цьому високу швидкість виконання кожного потоку. Саме тому вони чудовим інструментом для виконання трасування променів. При цьому виникає необхідність розробки таких алгоритмів та структур даних, які дозволять ефективно виконувати трасування променів на графічному процесорі, використовуючи усі особливості його архітектури.

Основним інструментом для роботи з графічним процесором є технологія CUDA (англ. Compute Unified Device Architecture).

Аналіз останніх досліджень і публікацій

Технологія CUDA перебуває в постійному розвитку [1]. На даний момент розробник пропонує використовувати п'яту її версію. Хоча сама технологія з'явилась в 2007 році, і отримала поширення в обчислювальних експериментах [2], в комп'ютерній графіці застосування її не надто поширене. Трасування променів використовується в комп'ютерній графіці достатньо давно [3], але, наприклад, в програмах реального режиму часу майже не використовується через високу ресурсоємність. Проте, використовуючи розробки [4], можна реалізовувати його в реальному режимі часу. Окремі практичні моменти реалізації з технологією CUDA є в [5].

Постановка завдання

Було поставлено завдання дослідити групу алгоритмів, які використовуються для побудови реалістичних тривимірних зображень методом трасування променів. Важливою вимогою є можливість побудови тривимірних зображень в реальному часі.

Детальніше необхідно дослідити:

- технологію паралельного програмування NVIDIA CUDA на мовах програмування Visual C++ та CUDA C;
- методи рендерингу тривимірних сцен;
- метод трасування променів;
- можливості розробки якісного програмного продукту, котрий у реальному часі створює фізично правильні та найбільш наближені до реальності рендери тривимірних сцен.

Основний матеріал дослідження

Технологія CUDA базується на швидкому виконанні графічним процесором порівняно простих паралельних обчислень.

Паралельні обчислення — це форма обчислень, в яких кілька дій проводяться одночасно. Ґрунтуються на тому, що великі задачі можна розділити на кілька менших, кожну з яких можна розв'язати незалежно від інших. Є кілька різних рівнів паралельних обчислень: бітовий, інструкцій, даних та паралелізм задач. Паралельні обчислення застосовуються вже протягом багатьох років, в основному в високопродуктивних обчисленнях, але зацікавлення ним зросло тільки недавно, через фізичні обмеження зростання частоти. Оскільки споживана потужність (і, відповідно, виділення тепла) комп'ютерами стало проблемою в останні роки, паралельне програмування стає домінуючою парадигмою в комп'ютерній архітектурі, переважно у формі багатоядерних процесорів.

Паралельні комп'ютери можуть бути грубо класифіковані згідно з рівнем, на якому апаратне забезпечення підтримує паралелізм: багатоядерність, багатопроцесорність — комп'ютери, що мають багато обчислювальних елементів в межах одної машини, а також кластери, MPP, та ґрід — системи що використовують багато комп'ютерів для роботи над одним завданням. Спеціалізовані паралельні архітектури іноді використовуються поряд з традиційними процесорами, для прискорення розв'язання особливих задач.

Програми для паралельних комп'ютерів писати значно складніше, ніж для послідовних, бо паралелізм додає кілька нових класів потенційних помилок, серед яких найпоширенішою є стан гонитви. Комунікація та синхронізація процесів — одна з найбільших перешкод для досягнення хорошої продуктивності паралельних програм.

Потенційне прискорення алгоритму при збільшенні числа процесорів задається законом Амдала, що вперше був сформульований у 1960-их роках.

Відображення зображення з допомогою трасування променів спирається на ряд тверджень, постулатів і властивостей. У природі джерело світла випромінює промінь світла, який мандрує, поки його поширенню не заважає якийсь об'єкт. «Промінь» можна уявляти собі як потік фотонів, що летять у одному напрямку. Якщо знехтувати релятивістськими ефектами, то у повному вакуумі такий потік буде поширюватися вздовж геометричної прямої лінії. У реальному середовищі з променем можуть трапитися різні ефекти, наприклад, поглинання, відбиття, заломлення, флуоресценція тощо. Поверхня може відбити світло у одному чи декількох напрямках, або розсіяти його. Світло може бути частково відбите, частково розсіяне, а частково — поглинуте поверхнею. Якщо ж матеріал поверхні є прозорим, то світло також буде поширюватися крізь матеріал, який може поглинати його у всьому чи частині спектру, змінюючи таким чином колір променя. При цьому світло буде поширюватися у іншому напрямку (заломлення). Рідше світло може бути частково поглинуте матеріалом і знову випромінене у вигляді флуоресценції світла довших хвиль (інший колір) у довільних напрямках. Сума відбитого, заломленого, поглинутого і випроміненого у вигляді флуоресценції світла має дорівнювати отриманому світлу. Надалі нові промені відбитого, заломленого чи випроміненого світла самі потрапляють на інші об'єкти, де вони теж зазнають вказаних ефектів. Деякі з цих променів досягають наших очей, і завдяки інформації про колір і силу світла, що в них міститься, ми маємо змогу бачити світ навколо нас.

Трасування променів (англ. ray tracing) у комп'ютерній графіці є способом створення зображення тривимірних об'єктів чи сцени за допомогою відстеження ходу променя світла крізь точку екрану і симуляції взаємодії цього променя з уявними об'єктами, що підлягають відображенню. Цей спосіб дозволяє створювати надзвичайно реалістичні зображення, зазвичай значно вищої якості, ніж дає типовий алгоритм Scanline або ж метод кидання променів (англ. Ray casting), проте має значно вищу обчислювальну складність. Із цієї причини алгоритми трасування променів використовуються там, де немає суттєвих обмежень часу рендерингу, наприклад у створенні нерухомих зображень чи для комп'ютерної графіки і спецефектів у фільмах, мультиплікації чи телебаченні, але в наш час є малопридатними для застосувань, що працюють в режимі реального часу, наприклад, відеоігор. Метод трасування променів здатний симулювати широкий набір оптичних ефектів, таких як відбиття променів, їх заломлення, розсіювання чи хроматичну аберацію.

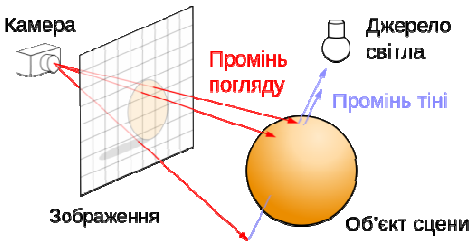

Рис. 1 Схема виконання алгоритму трасування променів
Авторська розробка

На рис.1 зображено загальну схему роботи алгоритму: уявні промені проходять крізь піксель екрана і йдуть у сцену, взаємодіючи з нею.

Метод трасування променів, як і будь-який інший метод, має свої певні переваги та недоліки. До переваг даного методу можна відвести наступні пункти:

- Алгоритм дозволяє природнім чином отримати такі ефекти, які для інших алгоритмів рендерингу складають значну складність. Серед них правильне затінення, дзеркальні поверхні, заломлення світла. Завдяки цьому реалістичність сцен, обрахованих методом трасування променів, інколи сягає «фотографічної».
- На відміну від поширених полігональних алгоритмів, він дозволяє обробляти у сцені об'єкти фактично довільних геометричних форм, що можуть бути виражені математичним описом — справжні (а не апроксимовані полігонами) полігони, сфери, тори, конуси, параболоїди, еліпсоїди тощо.
- Алгоритм також дозволяє природно розпаралелити обчислення між процесорами, адже обрахування ходу кожного променя у сцені відбувається незалежно.

А до вад методу трасування променів належать:

- Головним недоліком на теперішній час є надмірна часова складність алгоритму, яка перевищує можливості сучасних настільних і портативних комп'ютерів.
- Вадою є те, що хоча алгоритм точно враховує різні оптичні ефекти, традиційний метод трасування променів дає не завжди фотореалістичні зображення. Справжній фотореалізм можливий, лише коли рівняння рендерингу апроксимується доволі точно, адже це рівняння враховує кожен можливий ефект потоку світла. Однак це інколи потребує надзвичайно великих обчислювальних потужностей. Тож будь-який метод рендерингу є лише наближенням до цього рівняння, і метод трасування променів інколи не є найбільш близьким, а отже і реалістичним.

CUDA (англ. Compute Unified Device Architecture) — технологія GPGPU (англ. General-purpose computing on Graphics Processing Units), що дозволяє програмістам реалізовувати мовою програмування C алгоритми, що виконуватимуться на графічних процесорах Geforce восьмого покоління і вище (Geforce 8 Series, Geforce 9 Series, Geforce 200 Series), Nvidia Quadro і Tesla компанії Nvidia. Технологія CUDA розроблена компанією Nvidia.

Технологія CUDA — це середовище розробки на C, яка дозволяє програмістам і розробникам писати програмне забезпечення для вирішення складних обчислювальних завдань за менший час завдяки багатоядерній обчислювальній потужності графічних процесорів. Простіше кажучи, графічна підсистема комп'ютера з підтримкою CUDA може бути використана, як обчислювальна.

При використанні GPU розробнику доступно декілька видів пам'яті: регістри, локальна, розподілена, глобальна, константна і текстурна пам'ять. Кожен з цих типів пам'яті має певне призначення, яке обумовлюється її технічними параметрами (швидкість роботи, рівень доступу на читання і запис). Ієрархія типів пам'яті показана на рис.2. Уся пам'ять GPU поділяється на сітки (grids), які, у свою чергу, поділяються на блоки (blocks), кожен з яких ділиться на нитки (threads). Усі дані пам'ять GPU приймає з CPU, і після виконання своїх завдань повертає результати своєї роботи до CPU, де й проходить подальше опрацювання даних. Такий цикл може виконуватись багато разів.

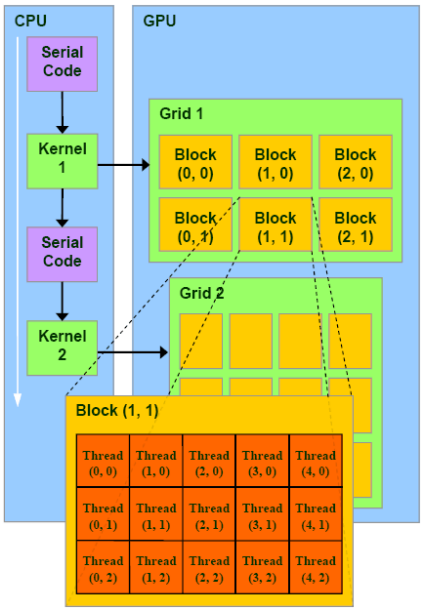

Рис. 2 Види пам'яті NVIDIA CUDA
Авторська розробка

CUDA дає розробникові можливість на свій розсуд організовувати доступ до набору інструкцій графічного прискорювача і управляти його пам'яттю, організовувати на ньому складні паралельні обчислення. Графічний процесор з підтримкою CUDA стає потужною програмованою відкритою архітектурою подібно до сьогоднішніх центральних процесорів.

Все це надає в розпорядження розробника низькорівневий, розподілюваний і високошвидкісний доступ до устаткування, роблячи CUDA необхідною основою при побудові серйозних високорівневих інструментів, таких як компілятори, математичні бібліотеки, програмні платформи.

CUDA використовує grid-модель пам'яті, кластерне моделювання потоків і SIMD інструкції. Застосовується в основному для високозатратних графічних обчислень і розробок nvidia-сумісного графічного API. Включена можливість підключення до застосунків, що використовують OpenGL 9 і Microsoft Direct3D.

У основі CUDA API лежить розширена мова C. Для успішної трансляції коду цією мовою, до складу CUDA SDK входить власний C-компілятор командного рядка nvcc компанії Nvidia.

GPGPU (англ. General-purpose graphics processing units — «GPU загального призначення») — техніка використання графічного процесора на відеокарті для проведення неграфічних розрахунків.

Загальний алгоритм роботи програми із використанням CUDA може бути таким:

- Скопіювати дані з CPU в пам'ять GPU.

- Запустити нитки обробки даних.
- Дочекатися завершення обробки даних.
- Скопіювати дані з пам'ять GPU в CPU.

Основні функції програми для побудови реалістичних тривимірних зображень методом трасування променів можуть бути такі:

- Створення реалістичних зображень.

Завдяки використаним алгоритмам та технологіям паралельного програмування, розроблений програмний продукт може у реальному часі створювати близькі до реальності та фізично коректні рендери тривимірних сцен.

- Можливість завантаження моделей у форматі obj.

Розроблена програма прийматиме на вхід лише файли з розширенням .obj. Такі файли легкі для сприйняття програмою, тож не потребують особливих затрат системних ресурсів, тому читання такого файлу потребуватиме порівняно мало часу.

- Збереження результатів рендерингу в різних графічних форматах.

У результаті виконання програми ми отримуватимемо звичайне зображення відрендереної сцени, тож необхідна можливість збереження вихідного зображення у багатьох графічних форматах.

Програмний продукт можна спроектувати та реалізувати з використанням методів опрацювання тривимірних зображень, основним реалізованим алгоритмом — алгоритм трасування променів. Код програми можна створити у Microsoft Visual Studio 2010 мовою програмування Visual C++ та CUDA C. Для створення користувацького інтерфейсу використати крос-платформовий інструментарій розробки програмного забезпечення QT. Також у програмній реалізації слід використати технологію паралельного програмування NVIDIA CUDA. При роботі з тривимірними зображеннями задіяти можливості бібліотеки функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення OpenCV.

У програмі зчитування даних проходитиме з файлу формату OBJ. OBJ (або .OBJ) — відкритий файловий формат опису геометрії, він був прийнятий іншими розробниками 3D редакторів як стандартний. Формат OBJ дуже простий, і задає тільки геометрію об'єкта, а якщо конкретно, то координати кожної вершини, її текстурні координати, нормалі та грані, що задаються списками вершин многокутників. Вершини многокутників за замовчуванням задаються проти годинникової стрілки, роблячи явне задання нормалей необов'язковим.

Дані про тривимірну сцену в OBJ-файлі записуються у такому вигляді:

- Список вершин заданих координатами (x,y,z);

Зразок: v 0.123 0.234 0.345.

- Текстурні координати (u,v);

Зразок: vt 0.500 -1.352.

- Нормалі (x,y,z); нормалі можуть бути не нормалізовані;

Зразок: vn 0.707 0.000 0.707.

- Кожна грань задається множиною трьох індексів, кожен з яких відповідає за вершину/текстуру/нормаль, координати яких записані в списках вище, тому f 1/1/1 2/2/2 3/3/3 це трикутник, що має текстурні координати та нормалі для всіх трьох вершин;
- Списки нумеруються починаючи з одиниці;

Зразок: f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3.

- Чотирикутники, та інші многокутники задаються відповідною кількістю вершин;

Зразок: f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3 v4/vt4/vn4.

Оскільки файли цього типу мають чітку структуру, то зчитування інформації не потребує великих затрат часу.

Алгоритм зворотного трасування променів дозволяє добитися максимальної реалістичності зображення. Він дозволяє змоделювати поширення світла у просторі, враховуючи закони геометричної оптики. Він дозволяє будувати якісні тіні з огляду на велику кількість джерел світла. Інші алгоритми не дозволяють так просто і якісно будувати тіні. Серед всіх алгоритмів оптимізації даний виявився найефективнішим.

Через кожен піксель з віртуального «ока» глядача проводиться промінь. Кожен промінь трасується. Спрощений алгоритм трасування виглядає так:

— шукаємо перетин променя з деякою поверхнею сцени;

— якщо не знайшли, то для цього пікселя встановлюємо деяку фонову освітленість;

— якщо ж перетин знайшовся, то:

а) поточну яскравість вважаємо фоновою;

б) для кожного точкового джерела світла випускаємо промінь зі знайденої точки перетину, шукаючи його перетин зі сценою;

в) якщо перетин знайдено, і він розташований між поверхнею і джерелом світла, то джерело світла не освітлює цю поверхню;

г) якщо джерело світла освітлює поверхню, то вираховуємо за рівнянням Фонга чи іншим освітленість точки поверхні та додаємо її до поточної;

д) якщо поверхня має властивість відбивати все або частину світла, то випускаємо відбитий промінь, трасуючи його, як і інші, а результат його обрахунків яскравості враховуємо для розрахунку яскравості початкового променя, беручи до уваги коефіцієнт відбиття та інші можливі властивості поверхні;

е) якщо матеріал поверхні пропускає і заломлює світло, то також випускаємо новий промінь у напрямку заломлення, який трасуємо. Можливе послаблення світла в процесі проходження крізь речовину, що теж враховується. Потім яскравість заломленого променя теж враховується для розрахунку сумарної яскравості початкового променя.

Користувацький інтерфейс можна розробити із використанням крос-платформового інструментарію розробки програмного забезпечення QT. Розробка інтерфейсу складається з двох основних етапів: візуального проектування користувацького інтерфейсу із використанням програми QT Designer та програмування його реакції на дії користувача. Розроблена програма дозволятиме отримати близьке до реальності зображення, також воно буде фізично правильне. Це забезпечить у загальному високу якість зображення.

Зображення такого типу можуть застосовуватися у презентаціях найрізноманітнішої продукції. Розробка може надати високоякісні та близькі до реальності зображення об'єкта, користуючись лише тривимірним зображенням моделі. Програма також забезпечуватиме відносно високу швидкість отримання вихідного зображення. Програму також можна використовувати дизайнерами для фізично правильної та швидкої візуалізації певної 3D-сцени.

Програми для виконання подібних завдань використовуються у сучасності для презентацій нових концептів автомобілів, комп'ютерів, взуття багатьма відомими виробниками. Не всі з них є фізично достовірними і не обов'язково вимагають мінімум машинних ресурсів. Розробка якраз і буде позбавлена подібних недоліків.

Висновки

В даному дослідженні було зроблено огляд методу трасування променів (англ. ray tracing), способу створення зображення тривимірних об'єктів чи сцени за допомогою відстеження ходу променя світла крізь точку екрану і симуляції взаємодії цього променя з уявними об'єктами, що підлягають відображенню. Цей спосіб дозволяє створювати надзвичайно реалістичні зображення, значно вищої якості, ніж дає типовий алгоритм Scanline або ж метод кидання променів (англ. Ray casting), проте має значно вищу обчислювальну складність. Метод трасування променів здатний симулювати широкий набір оптичних ефектів, таких як відбиття променів, їх заломлення, розсіювання чи хроматичну аберацію. Для подолання бар'єру малої обчислювальної потужності персональних комп'ютерів можна використовувати технологію NVidia CUDA, яка призначена для високошвидкісних паралельних обчислень. Це середовище розробки на C, яке дозволяє програмістам і розробникам створювати програмне забезпечення для вирішення складних обчислювальних завдань за менший час завдяки багатоядерній обчислювальній потужності графічних процесорів, графічна підсистема комп'ютера з підтримкою CUDA може бути використана, як повноцінна високопродуктивна обчислювальна система.

Перспективи подальших досліджень.

На даний момент трасування променів (в тому числі зворотне) не має широкого поширення в комп'ютерній графіці через високу ресурсоємність. Але, зважаючи на високу, часто

фотографічну, якість зображення і подальший ріст продуктивності комп'ютерів, можна сподіватись на подальший розвиток технології CUDA і розвиток програмного забезпечення з використанням цієї технології (точніше, подальший розвиток паралельних обчислень

1. Parallel Programming and Computing Platform | CUDA | NVIDIA. [Electronic resource] — Mode of access: http://www.nvidia.com/object/cuda_home_new.html. - Last access: 2013. – Title from the screen.
2. NVIDIA CUDA — неграфические вычисления на графических процесорах. [Електронний ресурс] — Режим доступу http://www.ixbt.com/video3/cuda-1.shtml. — Останній доступ: 2013. — Заголовок з екрану.
3. Проблемы трассировки лучей — из будущего в реальное время. [Електронний ресурс] — Режим доступу http://nvworld.ru/articles/ray_tracing/. — Останній доступ: 2013. — Заголовок з екрану.
4. Интерактивная трассировка лучей с использованием SIMD инструкций. [Електронний ресурс] — Режим доступу http://software.intel.com/ru-ru/articles/interactive-ray-tracing/. — Останній доступ: 2013. — Заголовок з екрану.
5. Stencil Buffer на CUDA. [Electronic resource] — Mode of access: http://www.ray-tracing.ru/articles233.html. - Last access: 2013. – Title from the screen.