УДК 681.31

Денисюк В.О.

Вінницький національний аграрний університет

ЕТАПИ ГРАФІЧНОГО КОНВЕЄРУ

В статті розглядаються особливості основних етапів графічного конвеєру з точки зору вибору оптимальних програмних засобів апаратних спеціалізованих пристроїв машинної графіки відтворення складних картин і сцен для кожного із етапів графічного конвеєру.

In the article the features of the main stages of graphic pipeline are probed with sharpen sights of choice of optimal softwave and hardwave of reproducing of difficult graphics pictures and scene for every stage of graphic pipeline.

Ключові слова: комп'ютерна графіка, графічний конвеєр, відображення графічної інформації

Вступ

Важливим завданням комп'ютерної графіки є інформаційне обслуговування життєдіяльності людини, що виражається в широкому застосуванні у різноманітних галузях науково-практичної діяльності людини графічної інформації та використанню засобів відображення графічної інформації, які є невід'ємною частиною різних обчислювальних комплексів, систем і мереж.

Постановка задачі

Різноманітні системи відображення графічної інформації – це складна сукупність інтерактивних технічних і програмних засобів для подання інформації у графічній формі [2-5, 9-11, 15]. Інтерактивні системи при побудові графічних сцен дотримуються певної послідовності дій, які

складають графічний конвеєр. Метою дослідження є визначення основних показників графічного конвеєру по перетворенню об'єктів, що описуються у тривимірному просторі в матрицю комірок відеопам'яті растрового дисплея - у растрове зображення [4, 6, 13]. Актуальність дослідження обумовлена необхідністю вибору оптимальних програмних засобів та апаратних спеціалізованих пристроїв машинної графіки для відтворення складних картин і сцен на кожному етапі графічного конвеєру.

Аналіз особливостей графічного конвеєру

Сучасні інтерактивні графічні системи при побудові об'ємних сцен дотримуються певної послідовності дій, які в сукупності складають графічний тривимірній (3D) конвеєр. Групу операцій графічного конвеєра (ГК), що виконують відокремлені проміжні дії прийнято називати стадією або етапом графічного конвеєра. За виконання кожного з етапів відповідає своя підсистема: підсистема опису сцен у 3D-просторі, геометрична підсистема, що об'єднує теселяцію геометричних моделей, афінні перетворення, видові претворення та вибір моделі освітлення, підсистема рендерингу, яка містить у собі процеси видалення невидимих поверхонь, накладання текстур, зафарбовування та фінальне опрацювання сцени, підсистема кадрового буфера та підсистема візуалізації, що відповідають за кінцеве відображення об'ємної сцени на екрані [4, 5, 13].

Концептуально ГК необхідно розглядати, як деяку узагальнену модель виведення 3D графічної інформації [4, 11]. Виділяють такі основні етапи ГК: опис 3D зображення, геометричні перетворення та рендеринг [4, 5, 9, 13].

На етапі опису зображення визначають стан усіх складових об'єктів, їх взаємне розташування та подальша стратегія дій над об'єктами. Відбувається відсікання по об’єму реальних об’єктів з світової системи координат (ССК) та отримання усіченої світової системи координат

(УССК). Із реальних об’єктів у ССК отримуємо їх моделі. Моделі реальних об’єктів розглядаємо в УССК, які є зменшиними за множиною параметрів та властивостей у порівнянні з об’єктами ССК, але такими, що задовільняють вимогам та цілям задач їх використання. Обидві координатні системи (ССК та УССК) мають однакові розмірності та орієнтацію. У випадку моделювання 3D графічних об’єктів це – довжина, висота, глибина (X,Y,Z), які вимірюються у метрах (км, дм, см тощо) [2, 4, 5, 9, 13, 14].

На етапі геометричних перетворень у якості вхідних використовуються об’єкти УССК. На цьому етапі в залежності від області використання моделі, тобто точки зору на графічні об’єкти (центру проекції) відбувається проекція об’єктів на відповідні площини проекції – виконується видова операція. Над об’єктами проводиться декомпозиція, теселяція геометричних моделей, виконуються афінні та видові перетворення, визначається тип моделі освітлення і т.ін. Отримані дані використовуються для побудови об’єктів у нормованій координатній системі (НКС), яка, у загальному випадку, орієнтується на засоби подальшого використання моделі та має відносну розмірність не більше одиниці по кожній із координат. У випадку графічного нормування мова іде про зменшення вимірів до двох – довжина та висота (X та Y),- для представлення інформаціїї за допомогою традиційних засобів виведення графічної інформації, які мають пласку, двовимірну (2D), область виведення. Спосіб машинної організації НСК може бути довільним – лінійним, двовимірним, тривимірним, асоціативним і т.ін. [4, 13, 14].

На етапі кінцевої візуалізації (рендерингу) за даними про зображення, отриманими на етапі нормування, виконуються різноманітні геометричні перетворення, формуються видимі пікселі зображення, для яких визначаються екранні координати та кінцева інтенсивність кольору у координатному просторі пристроя відображення [13, 14]. Розмірність

координат пристроїв відображення на тепер не перевищує сотень мм, а у деяких випадках - одиниць м [3, 8].

Розглянемо етапи виведення 3D графічної інформації більш детально. Сам принцип конвеєрної обробки 3D-зображень є технологічним стандартом [14, 17]. За конвеєрним принципом працюють усі 3D програмні інтерфейси і графічні акселератори [1, 2, 5, 6, 11, 12, 17].

При описі зображення використовується тріангуляція зображення. При тріангуляції зображення розбивається на трикутники, оскільки: 1) трикутник є найпростішим полігоном, вершини якого однозначно задають площину; 2) будь-яку область можна гарантовано розбити на трикутники; 3) обчислювальна складність алгоритмів розбиття на трикутники набагато менша, ніж при використанні інших полігонів; 4) для трикутника легко визначити трьох його найближчих сусідів, які мають із ним спільні грані [5, 12, 13]. Тріангуляція залежитьвід об'єктів обробки. Якщо об'єкти обробки багатокутники (піраміда, призма тощо), то необхідно поділити їх грані на трикутники. Якщо об'єкти обробки криволінійні поверхні, то застосовуються складніші алгоритми, наприклад, метод Делоне, або спочатку об'єкт розбивається на багатокутники (виконується теселяція), а потім вже на трикутники. Для сучасних задач машинної графіки загальна кількість пікселів може досягати млрд. на один кадр [4, 20].

Всі операції рендерингу (кінцевої візуалізації) виконуються за багатокроковим механізмом - конвеєром рендерингу. Однією з основних та найбільш трудомістких процедур рендерингу є процедура зафарбовування, згідно з якою для кожної точки поверхні визначається інтенсивність кольору та екранні координати, що вимагає високої колірної здатність до 24 колірних площин (в окремих випадках до 48) [7, 9, 15, 20]. Спочатку при зафарбовуванні обирається модель освітлення, яка визначає взаємодію об'єкта зі світлом, яке на нього падає. При цьому враховується розташування джерела світла, його тип, оптичні властивості матеріалу поверхні об'єкту. За

оптичні характеристики об'єкта відповідає двопроменева дистрибутивна функція (BRDF) кількох змінних, які характеризують властивості матеріалу, з якого виготовлено об'єкт [12, 13].

Під час трансформації перетворюються координати об'єктів, до них застосовуються матриці трансформації. Перед наступним етапом можна реалізувати повершинне освітлення, коли кожна вершина отримує значення освітлення, а потім вони інтерполюються по поверхні полігону. Також на цьому етапі присутні вершинні шейдери, які дозволяють деформувати чи викривляти об'єкти при зміні координат вершин. Трансформація вигляду полягає у тому, що координати об'єктів, що займали місця в системі координат сцени транслюються в координати, що прив'язані до віртуальної камери (центр проекції), це робиться для спрощення наступного підкроку - трансформації проекції. На цьому кроці відбувається проекція 3D-об'єктів у 2D-площину. Те, що видно на цій площині, є результатом зйомки віртуальної камери. Для того, щоб на вхід наступних етапів конвеєра не поступала зайва інформація (об'єкти та їх фрагменти, які не потрапляють у поле видимості камери) застосовуються методи відсікання невидимих частин. Однак ця задача достатньо складна [13, 14]. Для відсікання використовується об'єм відсікання (clipping volume) - шістьма площинами за трьома координатами обмежується область сцени, яка гарантовано буде помітна на екрані. Потім застосовується backface culling - відкидання задніх граней (отримуємо об’єкт в УССК) [2]. У кожного полігона окрім координат вершин є важлива характеристика - нормаль. Це вектор, який лежить на перпендикулярі, що виходить із геометричного центра трикутника. У кожного полігона є дві сторони - лицьова і зворотна. Нормаль визначає його орієнтацію [2, 5, 9, 13, 14].

При растеризації 3D-зображення, спроектоване на площину, перетворюється у растровий формат, тобто визначаються значення результуючих пікселів [9, 13]. На етапі обробки окремим пікселям надаються значення

кольору, що були отримані інтерполяцією кольорів вершин, або вони замінюються чи до них додається колір текстури. Тут може діяти піксельний шейдер, який визначеним чином комбінує колір, глибину і позицію пікселя з текстурами або за спеціальними алгоритмами. Після проходження усіх стадій конвеєра зображення заноситься у буфер кадру, який має передній і задній шари. В задньому - новий кадр, в передньому - поточний. Коли приходить час наступного рендеринга, вміст цих шарів міняється місцями (swap), у результаті на екрані новий кадр, а старий надсилається у задній буфер, де замінюється наступним, щойно прорахованим [13,14].

У процесі рендеринга виконується етап накладання текстур, що робить віртуальний світ реалістичним. Текстура - це двовимірна бітова картка (картинка), яка накладається на полігон і зображує фактуру його поверхні [13, 14]. Те, що не можна змоделювати полігонами, можна намалювати, причому, якщо на поверхні, яку ми моделюємо текстурою, немає виразних чи неоднорідних деталей, вона буде виглядати реалістично. Якщо звичайне зображенням складається із пікселів, то текстура складається із текстелів. Текстура накладається строго за координатами. При роботі з текстурами виникає немало проблем, тому задача формулюється наступним чином: як визначити колір конкретного пікселя на екрані, якщо в сцені на нього приходиться менше або більше текстелів. Раніше дану задачу вирішували з методом Point Sampling. Від кожного пікселя на екрані опускається промінь вглиб сцени. Текстель, що найближчий до цього променя і накладається на екран. При цьому обов'язково виникають помилки. Якщо текстелів на кожен піксел забагато, то частина інформації просто втрачається, якщо ж на піксель менше одного текстеля (об'єкт близько до камери – центру проекції), пробіли заміщуються неіснуючими пік селами [9, 13, 14]. Такий метод сучасного користувача не влаштовує, тому використовують методи білінійної фільтрації (Bi-linear Filtering), їх суть полягає в тому, що колір пікселя отримується у результаті інтерполяції (усереднення) кольорів чотирьох

сусідніх текстелів. Якщо об'єкт розміщено далеко від камери, його текстура майже не спотворюється. А коли об'єкт недалеко від камери і текстелів не вистачає, інтерполяція створює розпливчасте зображення цієї області. Але білінійна фільтрація добре працює лише для полігонів, які паралельні або майже паралельні екрану, тому що чотири сусідніх текстелі, які беруться для інтерполяція, - це майже коло. Якщо площина нахиляється, коло перетворюється у еліпс, але інтерполюються текстелі по колу. Від цього постійно накопичуються невеликі помилки, які після визначеного кута нахилу стають помітними, текстура фільтрується геометричне невірно, а користувач спостерігає сильні спотворення. Для визначення кольору одного пікселя потрібно зчитувати кольори чотирьох тектселів, що збільшує навантаження на шину пам'яті. Тому, білінійну фільтрацію використовують досить рідко. Якщо користувач на екрані бачить текстуру значно далі або ближче, ніж передбачував розробник, то текстура досить сильно спотворюється фільтрацією, що стає помітним через появу "завад" зображення. Це можна вирішити за допомогою міп-меппінгу (MIP Mapping). Multum in Parvo (MIP) з латини – "багато в одному", - її сутність полягає у тому, що для однієї і тієї ж поверхні розробник створює декілька копій текстури із різним ступенем деталізації. Кожна наступна версія текстури більша або менша за попередню у 4 рази. Ця версія текстури називається міп-рівнем, а усі міп-рівні разом складають міп-каскад. Коли камера віддаляється від текстури, вона змінюється на міп-рівень з меншою роздільною здатністю, а коли наближається - з більшою. Переваги даної технології – це те, що незалежно від того, на якій відстані знаходиться спостерігач від об'єкта, текстура відображується без геометричних спотворень, а далеко розміщені текстури не забирають багато ресурсів акселератора. Недоліки - доводиться зберігати кілька копій однієї і тієї ж текстури, а переходи між міп-рівнями відбуваються досить різко [5, 14].

Сучасний 3D-конвеєр використовує у своїй роботі технологію шейдерів [1, 2, 5, 12, 15]. Шейдер - це програма одного з етапів ГК, що використовується у 3D-графіці для визначення кінцевих параметрів об'єкта чи зображення. Розрізняють вершинний, піксельний та потоковий шейдери. Вершинний шейдер оперує даними вершин полігону. До таких даних відносять координати вершини в просторі, текстурні координати, вектор бінормалі, вектор нормалі. Вершинний шейдер може бути використаний для видового чи перспективного перетворення вершин, генерації текстурних координат, розрахунку освітленості. Більшість графічних акселераторів (GeForce 9800GTX, Radeon HD 3870 X2) містять блок розрахунку трансформації і освітлення, що виконує фіксовані операції: встановлення параметрів рендерингу освітлення, текстур, матричні перетворення [7]. В графічному прискорювачі Radeon 8800 присутні 128 потокових процесорів (streaming processor), які є уніфікованими – можуть обробляти як піксельні, так і вершинні шейдери [12]. Технологія піксельних шейдерів дозволяє успішно вирішувати проблему змішування декількох текстур, у тому числі карт відображення, тіньових карт (освітлення), об'ємних текстур тощо. Тільки з появою піксельних шейдерів з'явилися текстури, що реалістично імітують воду і хмари [5]. Ускладнення та розширення функцій GPU вимагає орієнтування на 256-розрядну чи 512-розрядну архітектуру в арифметиці із плаваючою крапкою з використанням паралельних структур на базі використання кристалів від 700 млн. до 1,5 млрд. транзисторів, як правило за 55-90 нм технологією [2, 3, 6, 7, 11, 12, 19, 20].

В специфікації API DirectX намітилася тенденція зменшення кількість можливих станів ядра GPU за рахунок більшої уніфікації (у порівнянні з попередніми - DX8 SM1.X, DX9 SM2, DX9 SM3, DX10, DX10.1, DX11) [12, 17]. Найбільш важливі операції, які можна виконувати апаратними засобами графічного процесора (DirectX 10), такі: міжкадрова інтерполяція вершин (Key Frame Interpolation), що суттєво прискорює анімацію;

накладання вершин (Vertex Blending) з використанням більш ніж чотирьох матриць перетворення, що полегшує “скелетну анімацію” складних моделей без їх розбиття на декілька частин; процедурна геометрія (викривлення властивостей вершин параметричним об'єктом, наприклад, хвилі на водній поверхні); складні моделі освітлення, що враховують властивості матеріалу об'єктів (ray tracing). Технологія вершинних шейдерів є значним кроком вперед до фотореалістичної графіки. Вершинні шейдери рідко використовуються самостійно, у більшості операцій вони тісно пов'язані з піксельними шейдерами. Піксельний шейдер працює з пікселями зображення, кожному пікселю поставлений у відповідність деякий набір атрибутів, таких як колір, глибина, текстурні координати. Піксельний шейдер використовується на останньому етапі графічного конвеєра для формування видимих пікселів зображення. Одним із найбільш потужних нововведень API DirectX є геометричні шейдери (geometry shader, GS) в тому числі нова шейдерна модель Shader Model 5.0**.** Їх поява пов'язана із тим, що вершинні шейдери не здатні працювати із даними про зв'язки окремих вершин об'єкта, над яким працюють, тобто про геометричну топологію об'єкта, також вони не можуть втручатися в дані зв'язки, додавати і видаляти вершини [5, 17]. На відміну від них, геометричні шейдери працюють уже із цілими примітивами (точка, лінія, трикутник, патч) і їх зв'язками із сусідніми примітивами, але головне те, що вони можуть безпосередньо ними керувати, генеруючи на виході нові примітиви у будь-якій кількості або пропускаючи непотрібні, в залежності від вхідних даних і свого алгоритму. Також у стандартний графічний конвеєр API DirectX додано три нові стадії (DirectX 11): Hull Shader, Tessellator та Domain Shader. Зміни торкнуться і піксельного шейдера, що дозволить включити так звані обчислювальні шейдери (Compute Shader), які працюватимуть на загальні комп'ютерні застосування. Також в DirectX 11 додано деякі нові формати стиснення текстури, які забезпечуватимуть кращу якість зображення і підтримуватимуть великий

динамічний діапазон (ми розглянемо їх докладніше трохи пізніше) і маса невеликих функцій, більшості з яких не буде потрібно нове обладнання. Серед таких функцій - підтримка подвійної точності чисел з плаваючою комою (опциональная можливість, призначена для обчислювальних шейдеров), збільшення обмежень текстур до 16К і збільшення верхнього ліміту ресурсу до 2Гб [16, 18].

Такі широкі можливості в обробці геометричних об'єктів дозволяють перенести на графічний процесор (GPU) виконання багатьох операцій, які раніше виконувалися частково під керуванням центрального процесора (CPU), що знижувало швидкодію паралельної роботи процесора і відеокартки за рахунок частих блокувань 3D-ресурсів для виконання операцій над ними на CPU. Наприклад, можливо повністю перенести на GPU розрахунок тіньових об'ємів для певних алгоритмів динамічних тіней, реалізувати якісне сумісне перетворення об'єктів (displacement mapping) та деякі прогресивні техніки розмитості зображення (motion blur), візуальні ефекти (хутро, волосся, рослинність), однопрохідне створення кубічних карт (cubemap) [5, 14, 16-18].

Висновки

3D-сцену можна представити як набір окремих груп елементів: групи 3D-об'єктів, групи джерел освітлення, групи накладання текстурних карт, групи камер, які у результаті взаємодії одна із одною та їх обробки за допомогою спеціальних апаратних та програмних засобів можна організувати у єдиний процес, що систематизує правильну обробку тривимірних об'ємних сцен. Етапи графічного конвеєра, складені у відповідну послідовність операцій, не є жорстко заданими, але є загальноприйнятими у сучасних графічних підсистемах. Дослідження етапів графічного конвеєру дозволяє сформулювати особливості сучасних GPU, як засобів машинної графіки: 1) виконання обробки складних 2D і

3D-зображень згідно стандартів OpenGL чи API DirectX [14-17]; 2) відповідність високій швидкодії з метою створення динамічних картин у межах до 50 млрд. пікселів/с, що забезпечується використанням синхрочастот до 1,25 ГГц, із внутрішньою швидкодією до 900 ГФлопс і внутрішніми циклами в межах 0,4-0,5 нс [3, 16]; 3) апаратна реалізація функцій графіки; 4) активне використання технології шейдерів; 5) висока колірна роздільна здатність до 24, іноді - до 48; 6) можливість адресації відеопам'яті до одиниць Гбайт за стандартом GDDR; 7) орієнтація на 256-розрядну, а в окремих випадках на внутрішніх шинах, на 512-розрядну архітектуру в арифметиці із плаваючою крапкою з використанням паралельних структур; 8) орієнтацію на "класичні" алгоритмічні рішення [10, 13]; 9) реалізація GPU у вигляді окремих, або набору великих інтегральних схем, що представляють собою комплекти для побудови графічних систем, як з можливістю програмування, так і спеціалізованих (як правило на базі 40-90 нм технологій, на кристалах від 500 млн. до 2 млрд. транзисторів) [3, 16]; 10) можливість використання деяких GPU у якості загальноцільових високопродуктивних паралельних процесорів [1, 2, 6, 16, 17].

Література

1. Адинец А. Графический вызов суперкомпьютерам [Электронный ресурс] / А.Адинец, В.Воеводин // Открытые системы - 2008. - № 4. - Режим доступа: http://www.osp.ru/os/ 2008/04/5114497.html.
2. Денисюк В.О. Аналіз етапів графічного конвеєру/ Денисюк В.О., Боднар А.В., Ліщинський О.В. // Міжнародний науково-технічний журнал "Вимірювальна та обчислювальна техніка в технологічних процесах".- Хмельницький, 2009. - № 2 - С. 230-235.
3. Денисюк В.О., Цвілюк А.О., Чех О.І. Вибір процесору для графічної системи. Міжнародний науково-технічний журнал "Вимірювальна та

обчислювальна техніка в технологічних процесах”.- Хмельницький, 2009. - № 2 - С. 146-152.

4. Денисюк В.А. Исследование и разработка цифровых функциональных генераторов графических примитивов для устройств отображения информации: дис. … канд. техн. наук: 05.13.08 / Денисюк Валерий Александрович.- Винница.-1996.- 202 с.
5. Денисюк В. О. Основні етапи графічного конвеєру / Денисюк В. О., Нікітченко Т.П., Нікітченко Н.П. // Вісник Східноукраїнського національного університету ім. Володимира Даля, м. Луганськ, 2008. - №12(130). - Частина 2 - С.110-114.
6. Любке Д. Графические процессоры - не только для графики [Электронний ресурс]/ Любке Д. // Открытые системы. - М, 2007. - № 2. - Режим доступа: http://www.osp.ru/os/2007/02/ 4106864.
7. Набережный А. GeForce 9800GTX: шутки в сторону [Электронний ресурс] / Набережный А. // Мир ПК.- 2008.- № 04. - Режим доступа: http://www.osp.ru/pcworld/2008/04/ 5053483.
8. Нечай О. Тосихиро Сакамото: жизнь в высоком разрешении [Электронний ресурс]/ Нечай О. - М., 2008. - Режим доступа: http://www.terralab.ru/video/344394.
9. Палташев Т.Т. Технология визуализации в компьютерном синтезе реалистичных зображений / Палташев Т.Т., Климина С.И., Лях А.С. , Ю Вл. К. // Зарубежная радиоэлектроника. - № 6. - 1991. - С. 71, 96-108.
10. Пєтух А.М. Інтерполяція в задачах контурного формоутворення [Монографія] / Пєтух А.М., Обідник Д.Т., Романюк О.Н.- Вінниця: УНІВЕРСУМ-Вінниця, 2007.- 103 с.- ISBN 978-966-641-223-5.
11. Пєтух А. М. Швидкодійні цифрові функціональні генератори графічних примітивів: монографія / А.М.Пєтух, В.О.Денисюк, Д.Т.Обідник.- Вінниця: ВНТУ, 2010.- 148 с. ISBN 978-966-641-343-0.

12. Пугач Е. nVIDIA GeForce 8800: революция свершилась! Часть вторая: обзор GeForce 8800GTX / Пугач Е.- М, 2006.- Режим доступа: http://www.ferra.ru/online/video/s26691.
13. Роджерс Д. Алгоритмические основы машинной графики / Роджерс Д.; пер. с англ.- М.: Мир, 1989.- 512 с.- ISBN 5-03-000476-9.
14. Херн Д. Компьютерная графика и стандарт OpenGL / Д.Херн , М.Бейкер; пер. с англ. - М.: Издательский дом "Вильямс", 2005.- 1168 с. - ISBN 5-8459-0772-1.
15. Чеботарёв А. OpenGL: раскрой глаза на трехмерную графику [Электронный ресурс] / Чеботарёв А. Издательский Дом "КОМИЗДАТ".- М, 2004.- Режим доступа: http://www.comizdat.com.
16. ATI RADEON HD 5870 1024MB. В преддверии DirectX 11. Режим доступа: http://www.ixbt.com/ video3/cypress-part1.shtml
17. Blythe, D. The Direct3D 10 System [Electronic resource] / D. Blythe. - Microsoft Corporation, 2007.- Mode of access: http://download.microsoft. com/download/f/2/d/Direct3D10_web.pdf.
18. Microsoft DirectX 11 – новый виток эволюции игровой графики. Режим доступа: http://www.svideocards.ru/2009/11/15/microsoft-directx-11
19. Nvidia Tesla. GPU Computing Technical Brief [Electronic resource] / Nvidia, 2007.- v.1.0.0 - Mode of access: http:// www.nvidia.com.
20. Ultimate Display Technologies. IEEE Computer Society, August 2005.- V. 38, No. 8.