УДК 515.2:528.71

ДОСЛІДЖЕННЯ СПОСОБІВ ВІЗУАЛІЗАЦІЇ ТРИВИМІРНИХ СЦЕН У РЕАЛЬНОМУ ЧАСІ

Мірошниченко М.Ю., к.т.н. *e-mail*: *mykola.miroshnychenko@tsatu.edu.ua*
Таврійський державний агротехнологічний університет імені Дмитра Моторного

Актуальність та постановка проблеми. Візуалізація - це завершальний і, безумовно, найвідповідальніший етап створення тривимірної сцени. Редактор тривимірної графіки прораховує зображення, враховуючи геометрію об'єктів, властивості матеріалів, з яких вони виготовлені, розташування та параметри джерел світла тощо. Невдало виконана візуалізація може звести нанівець усі багатоденні зусилля щодо моделювання, освітлення та текстурування сцени. Створення сцени з реалістичним освітленням — це ще одне завдання, яке вирішують для того, щоб надати кінцевому зображенню більшої реалістичності.

У реальному світі світлові промені багаторазово відбиваються і переломлюються в об'єктах, внаслідок чого тіні, що відкидаються об'єктами, здебільшого мають нечіткі, розмиті межі. За якість відображення тіней, в основному відповідає апарат візуалізації. До тіней, що відкидають у сцені, пред'являються окремі вимоги. Тінь, що відкидається від об'єкта, може сказати багато про що — як високо він знаходиться над землею, яка структура поверхні, на яку падає тінь, яким джерелом освітлений об'єкт і т.д. Крім цього, тінь може підкреслити контраст між переднім та заднім планом, а також «видати» об'єкт, який не потрапив у поле зору об'єктива віртуальної камери. І тут глядачеві дається можливість самому домислити навколишнє оточення сцени.

Основна проблема побудови тіньового проектування полягає в тому, що апаратна частина для опису поверхні використовує в якості універсальної одиниці трикутники. Трикутники є елементарною поверхнею, для побудови якої використовується мінімальне число вершин. Описувані алгоритми побудови поверхонь апаратною частиною виділяються у загальну групу триангулятивних (що використовують у своїй основі універсальні одиниці трикутники). Сучасні графічні прискорювачі працюють на рівні не вище за триангулятивні алгоритми і нічого не знають про наявність сцени, тому тіні ними безпосередньо не підтримуються і вся підготовча робота лягає на плечі програміста. Проте в найпоширеніших алгоритмах практично всі стадії побудови тіні реалізуються саме через графічний прискорювач. Крім того, за останні роки були внесені значні зміни як до обсягу, так і до складових динамічних бібліотек, openGL, Direct3D тощо. у напрямку опису тіні. Поліпшення алгоритмів побудови тіней сприяли високим результатам реалістичності тривимірної візуалізації. Поява таких функцій і понять, як Decay, ShadowBias, ShadowType, Near/Far Attenuation і т.д., свідчать про збільшення гнучкості в керуванні тінню, що відкидається об'єктами.

Метою даної статті є дослідження питань вибору алгоритму побудови тіні, що є важливим етапом тривимірної візуалізації сцени.

Основні матеріали дослідження. Тіні бувають чіткі (*hard shadows*) та м'які (*soft shadows*). Точні тіні виходять, коли є точкове джерело або джерело спрямованого паралельного світла. Згідно з геометричною оптикою Френеля, у разі наявності протяжного джерела світла, від об'єкта виходить не одна тінь, а ціла серія тіней, які накладаються один на одного і утворюють більш-менш

затемнені області, які становлять основну тінь (*umbra*) і область півтіні (*penumbra*). Усі існуючі алгоритми побудови тіней дозволяють будувати чіткі тіні, але з них дозволяють розмити отриману тінь, створюючи псевдо-м'яку тінь.

Існує кілька основних підходів до побудови тіней:

- Перетворення моделі «на землю» та малювання її як тіні.
- Побудова тіньової маски об'єкта та проективне накладення її на інші об'єкти.
- Тіньові обсяги.
- Використання інформації про глибину (*depth buffer*).

1. Проєціювання (перетворення "на землю"). Фактично це перший алгоритм побудови тіні, який був застосований в іграх. Він відрізняється простотою реалізації та гарною якістю одержуваної тіні. Цей алгоритм був вперше описаний Д. Бліном [1], який описав рівняння для проектування полігону «на землю», тобто. на площину $z = 0$, у напрямку джерела світла.

Розглянемо два випадки:

1) Джерело на нескінченності.

2) Локальне джерело (точкове джерело неподалік об'єкта).

При цьому використовується геометричне взаємовідношення джерела світла та полігону для обчислення проекції кожного полігону моделі «на землю».

«Тіньові полігони» мають бути розраховані кожному із джерел світла, тобто, якщо об'єкт висвітлюється N джерелами світла, необхідно розрахувати N його «тіньових проекцій».

Розглянемо випадок, коли джерело світла розташовано на нескінченності відносно до об'єкта. У разі нескінченно віддаленого джерела світла ми припускаємо, що промені світла, що приходять до об'єкта, є паралельними (рис. 1). Це дозволить нам розв'язати рівняння проекції лише раз і застосовувати отримане рішення до всіх вершин об'єкта.

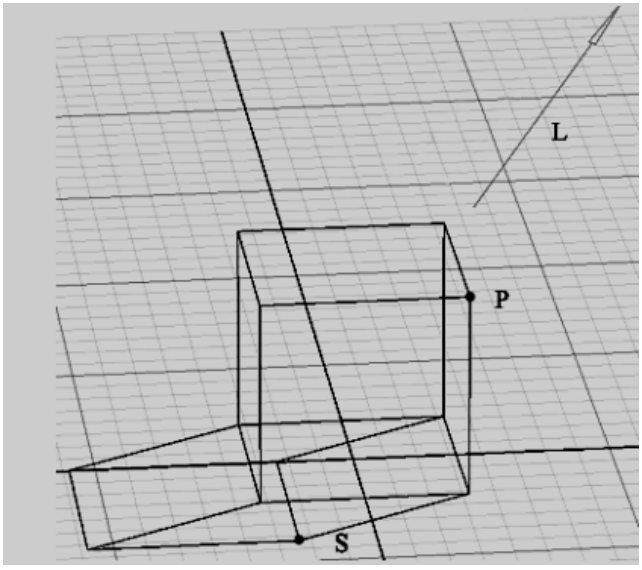

Рисунок 1. Схема розташування об'єкта та джерела світла
при нескінченній відстані між ними

Сформулюємо загальну постановку задачі: маючи координати точки джерела світла $(x_l; y_l; z_l)$ і координати вершини об'єкта $(x_p; y_p; z_p)$ необхідно

отримати координати проекції вершини об'єкта на площину $z = 0$, тобто точку тіні $(x_s; y_s; z_s)$.

З подібних трикутників отримуємо трикутників (рис. 1) отримуємо:

$$\frac{x_p - x_s}{z_p - z_s} = \frac{x_l - x_p}{z_l - z_p} \tag{1}$$

Розв'язавши це рівняння щодо x_s, отримуємо:

$$x_s = x_p - (z_p - z_s) \cdot \left(\frac{x_l - x_p}{z_l - z_p} \right) \tag{2}$$

Якщо прийняти, що L це вектор із точки P до джерела світла, то точку S можна висловити як

$$S = P - \alpha \cdot L. \tag{3}$$

Оскільки ми проектуємо на площину $z = 0$, то рівняння (3) можна переписати у такому вигляді:

$$0 = z_p - \alpha \cdot z_l \quad \text{або} \quad \alpha = \frac{z_p}{z_l}. \tag{4}$$

Розв'язавши рівняння (3) відносно x_s и y_s отримаємо

$$x_s = x_p - \frac{z_p}{z_l} \cdot x_l \quad \text{та} \quad y_s = y_p - \frac{y_p}{y_l} \cdot y_l. \tag{5}$$

або у матричному вигляді

$$M_s = \begin{vmatrix} 1 & 0 & -x_l / z_l & 0 \\ 0 & 1 & -y_l / z_l & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}. \tag{6}$$

Тепер, маючи координати точки P у світовому координатному просторі, можна отримати її проекцію на площину $z = 0$ просто шляхом множення на матрицю M_s

$$S = M_s \cdot P. \tag{7}$$

Рівняння (5) для віддаленого джерела світла може бути узагальнено для випадку, коли джерело світла знаходиться на кінцевій відстані від об'єкта (локальне джерело). І тут нам знадобляться додаткові обчислення для кожної вершину, так як кожна вершина має власний напрям на джерело світла. Тим не менш, у цьому випадку ми теж можемо перенести більшу частину обчислень у

матрицю M_s. Якщо L — точка розташування джерела світла, то рівняння (3) набуває вигляду:

$$S = P + \alpha \cdot (P - L). \tag{8}$$

Необхідно зробити проекцію на площину

$$\alpha = \frac{-z_p}{z_p - z_l}. \tag{9}$$

Отримуємо координати

$$x_s = \frac{x_l z_p - x_p z_l}{z_p - z_l} \quad \text{та} \quad y_s = \frac{y_l z_p - y_p z_l}{z_p - z_l}. \tag{10}$$

Якщо застосувати гомогенізацію після перетворення, то (10) можна записати у вигляді матриці

$$M_{sh} = \begin{vmatrix} -z_l & 0 & x_l & 0 \\ 0 & z_l & 0 & 0 \\ 0 & 0 & y_l & 0 \\ 0 & 0 & 1 & -z_l \end{vmatrix}. \tag{11}$$

Тоді

$$S_h = M_{sh} \cdot P. \tag{12}$$

Після цього после чего необхідно провести гомогенізацію точки S_h для получения проекции точки P на площину $z = 0$.

Побудова тіньової маски та проєктивне накладання. Цей метод є логічним продовженням попереднього, але, на відміну від нього, має цілу низку переваг. Об'єкт, який відкидає тінь (*shadow caster*) у той чи інший спосіб будується з погляду джерела світла чорним кольором в білу текстуру. Розмір текстури залежить від того, наскільки дрібні елементи об'єкта ми хочемо бачити в тіні. Уявімо собі площину, перпендикулярну напряму на джерело світла і розташовану відразу за об'єктом по променю світла - це і є текстура, в яку проводиться побудова об'єкта. Щоб отримати проєкцію об'єкта на цю текстуру, застосуємо елементи стандартного конвеєра трансформації.

Припустимо, що текстура, яку нам необхідно побудувати об'єкт, це екран. Тоді послідовність перетворень стає просто очевидною (стандартний конвеєр перетворення): *простір об'єкта => світовий простір => простір камери (у нас це джерело світла) => проективний простір => нормалізований проективний простір => екранний простір*. Для наших цілей необхідно модифікувати матриці переходів у простір камери, проектний простір та екранний простір.

Простір камери – це практично звичайна матриця камери, але тільки розташована в точці знаходження джерела світла і спрямована на *shadow caster*. У разі нескінченно віддаленого джерела світла віртуальна камера може розташовуватися в будь-якій точці на прямій між *shadow caster* та джерелом світла (зазвичай її вибирають близько до *shadow caster*).

Проектний простір. Ця матриця суттєво залежить від того, чи джерело світла є локальним або нескінченно віддаленим. Для нескінченно віддаленого джерела світла ця матриця має вигляд:

$$M_{pj} = \begin{vmatrix} \frac{2}{width} & 0 & 0 & 0 \\ 0 & \frac{2}{height} & 0 & 0 \\ 0 & 0 & -\frac{1}{depth} & 0 \\ 0 & 0 & 1 & 1 \end{vmatrix}. \quad (13)$$

де *width, height* та *depth* – ширина, висота та глибина проетивного обєма. В цьому випадку не потрібна гомогенізація.

Для локального джерела світла потрібна матриця перспективного перетворення

$$M_{pj} = \begin{vmatrix} \cos\left(\varphi/2\right) & 0 & 0 & 0 \\ 0 & aspect \cdot \cos\left(\varphi/2\right) & 0 & 0 \\ 0 & 0 & \frac{\sin\left(\varphi/2\right) \cdot far}{far - near} & \sin\left(\varphi/2\right) \\ 0 & 0 & -\frac{\sin\left(\varphi/2\right) \cdot far \cdot near}{far - near} & 0 \end{vmatrix},$$

де φ – кут зору, *aspect* - відношення ширини до висоти екрану (при побудові в квадратну текстуру *aspect = 1*), *far* і *near* - дальня та ближня стінки проективного об'єму відповідно. Після перспективного перетворення потрібна гомогенізація, як і у разі локального джерела при перетворенні «на землю».

Екранний простір. Модель із нормалізованого проективного простору має лягти у квадратну текстуру з мінімальними зазорами. Після того, як *shadow caster* промальований у текстурі, її необхідно накласти на затінені об'єкти (*shadow receiver*). Для цього можна скористатися технологією проектування текстур. Щоб уникнути повторення тіньової текстури, необхідно виставити *clamp* або *border color*. В результаті ми отримуємо тінь від одного об'єкта (shadow caster) на іншому (*shadow receiver*). Далі можна зробити з цією тінню:

1. Розмити (отримати псевдо-м'яку тінь). Зробити це можна кількома способами:

- «Вручну» шляхом накладання якогось *blur*-фільтра. Такий підхід зручний при програмному малюванні самої тіньової текстури.
- Використовувати "залізо". Можна відмалювати текстуру саму на себе зі зміщенням, або згенерувати *mip-level*.

2. Вводити тінь у прозорість на основі відстані до джерела світла. На жаль, реалізувати це без *vertex shader* за допомогою апаратної частини неможливо.

Єдине, що можна зробити для поліпшення виду тіні, так це зробити її не вугільно-чорною, а рівномірно напівпрозорою.

3. Мінімізувати витрати на побудову тіні. Це досягається при використанні технології LOD.

Тіньові об'єми. При освітленні сцени в тіні виявляються об'єкти, які потрапляють всередину тіньового об'єму [2]. Для того, щоб точка була затінена, промінь світла повинен увійти в тіньові обсяги більше разів, ніж вийти з них. Для спрощення завдання можна прийняти, що не промінь світла від джерела повинен пройти цей шлях, а промінь від спостерігача. За такого спрощення виникає кілька неприємних випадків, які розглянемо окремо.

Алгоритм можна розділити на дві явні дії: побудова «тіньової маски» (маски освітлених та затінених областей 2-мірної картинки) та відтворення сцени з використанням тіньової маски. Пропонується реалізація, яку можна розбити на три стадії.

Перша стадія. На цій стадії просто малюємо всю сцену таким чином, ніби вона затінена. Усі наступні стадії служать додавання освітлення в картинку. Ділянки картинки, які так і не будуть освітлені у наступних проходах, залишаться затіненими.

Друга стадія. На цій стадії будуємо тіньову маску, використовуючи інформацію про тривимірні тіньові об'єми. Для створення та зберігання тіньової маски нам знадобиться буфер шаблонів (*stencil buffer*). Побудова оптимальних тіньових об'ємів є досить складним завданням, яке виходить за рамки даної роботи. Опис одного з алгоритмів побудови силуету, необхідний для побудови оптимального тіньового обсягу, можна знайти, наприклад, в [3]. Отже, нам необхідно підрахувати для кожної точки різницю того, скільки разів промінь від спостерігача перетнув позитивно та негативно орієнтовані полігони тіньового об'єму. Це робиться за два проходи. На першому проході будуємо всі тіньові об'єми з нормальним (для нашого застосування) відкиданням тіні поверхні, алгоритм *back face culling*. У процесі виконання алгоритмів тінеутворення, наприклад, *z-buffer* для прискорення і спрощення прорахунку модель зберігається в триангулятивному вигляді. Тому часто йдеться про відкидання тіні набору трикутних поверхонь. Кожна точка, яка проходить тест по глибині, збільшує значення в буфері шаблонів на 1. На другому проході міняємо *back face culling* на протилежний – малюватимуться задні стінки тіньових об'ємів. Тепер кожна точка, яка проходить тест по глибині, зменшуватиме значення в буфері шаблонів на 1. У результаті після цих двох проходів ми маємо в буфері шаблонів нульові значення для освітлених областей і значення більше за нуль для затінених.

Третя стадія. На цій стадії вже проведено всі попередні приготування - є і затінена сцена, і тіньова маска. Залишається відмалювати ще раз всю сцену освітлення, використовуючи перевірку по буферу шаблонів.

Використання інформації про глибину. Метод базується на ідеї про те, що затінені точки – це ті, що «приховані» від джерела світла. Інакше кажучи, це «*hidden surfaces*» з погляду джерела світла. Щоб визначити невидимі поверхні з погляду джерела світла, необхідно відмалювати всю сцену з погляду джерела світла в буфер глибини. Далі необхідно відмалювати всю сцену з погляду спостерігача, проробляючи таку перевірку: якщо глибина точки, переведена в систему координат джерела світла, більше значення, яке ми зберегли з буфера глибини попередньої стадії, отже, точка затінена. В іншому випадку точка освітлена.

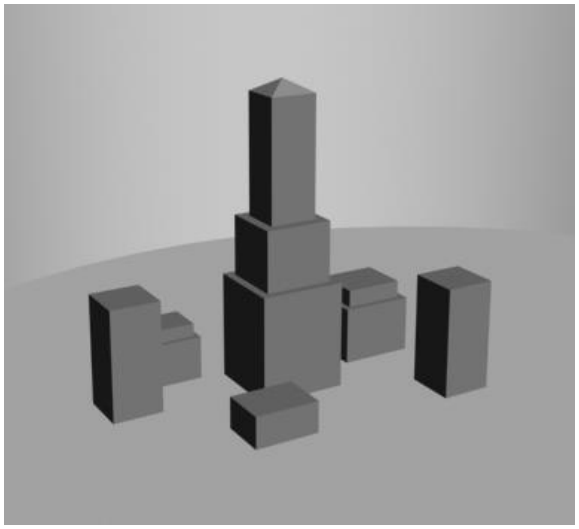

Рисунок 2. Сцена, побудована з погляду джерел світла

Якщо «залізо» не підтримує специфічних функцій (збереження та доступу до буфера глибини, порівняння глибини зі збереженим значенням тощо), то реалізувати його напряму практично неможливо. Але через те, що метод дуже привабливий, необхідно «обійти» апаратну частину. Припустимо, що апаратна частина не дає доступу до буфера глибини, у нього немає жодних можливостей порівнювати значення, крім порівняння по альфа-каналу. Єдине, що потрібно від апаратної частини, так це можливість зміни колірного буфера для поточного відмальовування (*render target*) та 2-текстурний растеризатор. На рис. 2. зображена сцена з погляду джерел світла, але на рис. 3. - сцена з погляду спостерігача з побудованими тінями.

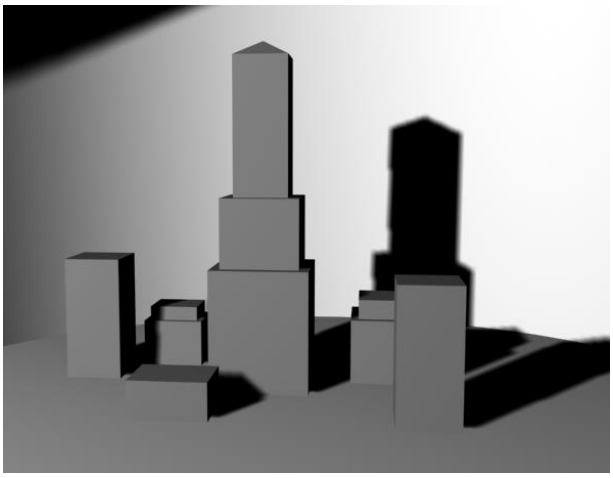

Рисунок 3. Сцена з побудованими тінями, зображена з боку спостерігача

Як і в попередньому методі, алгоритм можна розділити на дві явні дії: побудова тіньової маски та відтворення сцени з використанням тіньової маски.

Для отримання глибини ми скористаємося текстурою 256x1x32, в якій значення розміщені від 0 до 255 в альфа-каналі (колірні значення нас не цікавлять). Щоб текстура лягла необхідним чином, згенеруємо текстурні координати з подальшою трансформацією

$$S_h = \left(M_{pj} * M_{sel}\right) \cdot P_{cam}. \tag{14}$$

де M_{pj} – це матриця проективного перетворення, а M_{sel} – матриця вибору

$$M_{sel} = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{vmatrix}. \tag{15}$$

Після такої побудови, у текстурі ми маємо образ глибини сцени з дискретом 1/256. Далі створюємо тіньову маску, базуючись на порівнянні глибини. Для цього ми накладемо першу текстуру (256x1x32) на об'єкт способом, подібним до першої стадії, і накладемо другу текстуру, отриману на першій стадії, проективним методом. Після чого, у кожній точці віднімемо значення альфа-каналу першої текстури з другої та побудуємо в тіньову маску з перевіркою по альфа-каналу. Точки, які пройдуть цю перевірку, запишемо в тіньову маску як 1, а які не пройдуть залишимо в 0. Після цього залишається лише побудувати сцену, проставляючи значення у буфер шаблонів (рис. 4).

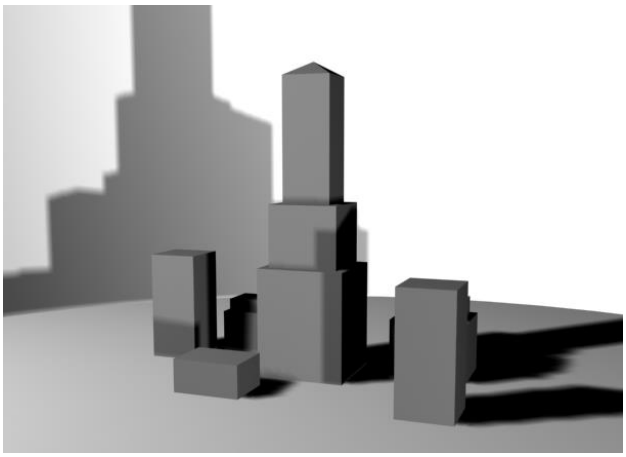

Рисунок 4. Результат створення тіньової маски

Висновки. У статті розглянуті та проаналізовані основні принципи і алгоритми побудови тіней в тривимірній графіці. Представлені алгоритми були розділені на дві явних дії: побудова тіньової маски та побудова сцени з використанням тіньової маски, і розділені на три стадії. Досліджена проблема побудови тіней в тривимірних сценах, в реальному часі. Перспективою подальших досліджень є розробка способу усунення помилкового самозатінення, яке виникає через недостатню точність уявлення глибини через альфа-канал.

Список використаних джерел:

1. Blinn James. Me and my (fake) shadow. *IEEE Computer Graphics and Applications*. 1988. 8(1). P. 82-86.
2. Franklin C. Crow. Shadow algorithms for computer graphics. In Computer Graphics (SIGGRAPH '77 Proceedings). 1977. P. 242-248.
3. Sander P., Gu X., Gortler S., Hoppe H., Snyder J. Silhouette Clipping, Computer Graphics (SIGGRAPH 2000 Proceedings). 2000. P. 327-334.