НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Факультет прикладної математики
Кафедра прикладної математики

«До захисту допущено»
Завідувач кафедри ПМА
_____ О. Р. Чертов
«____» _____ 2015 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності 6.040301 «Прикладна математика»

на тему: «Програмний модуль для візуалізації складних дзеркальних поверхонь у просторі»

Виконав: студент IV курсу, групи КМ-12
Байков Ігор Сергійович _____

Керівник асистент Дрозденко О.М. _____

Консультант старший викладач
з нормоконтролю Мальчиков В. В. _____

Рецензент к.т.н., доцент
Заболотня Т. М. _____

Засвідчую, що в цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент _____

Київ – 2015 року

Національний технічний університет України

«Київський політехнічний інститут»

Факультет прикладної математики

Кафедра прикладної математики

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 6.040301 «Прикладна математика»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМА
_____ О. Р. Чертов
«____» _____ 2015

ЗАВДАННЯ

на дипломну роботу студенту

Байкову Ігорю Сергійовичу

1. Тема роботи: «Програмний модуль для візуалізації складних дзеркальних поверхонь у просторі»

керівник роботи Дрозденко О. М.,

затверджені наказом по університету від «14» травня 2015 р. № 1039-С.

2. Термін подання студентом роботи: «15» червня 2015 р.

3. Вихідні дані до роботи:

- опис сцени що включає усі обєкти і їх групи
- камера

4. Перелік завдань, які потрібно розробити:

- аналіз існуючих методів;
- розробка алгоритму;
- програмна реалізація демонстрації;

5. Перелік ілюстративного матеріалу:

- блок-схема методу
- архітектура програми
- екранні знімки технологічної демонстрації
- презентація

6. Дата видачі завдання: «01» жовтня 2014 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання теми та завдання на виконання роботи	14.12.2014	
2	Ознайомлення з літературою, огляд існуючих реалізацій	10.02.2015-12.04.2015	
3	Огляд існуючих методів	14.04.2015-03.05.2015	
4	Розробка алгоритму	05.05.2015-09.05.2015	
5	Проектування та реалізація технологічної демонстрації	12.05.2015-24.05.2015	
6	Тестування техно-демоверсії	25.05.2015-27.05.2015	
7	Написання пояснювальної записки, оптимізація	28.05.2015-11.06.2015	
8	Підготовка матеріалів до захисту	12.06.2015	
9	Вдосконалення дипломної роботи	13.06.2015-18.06.2015	

Студент _____ Байков І.С.

Керівник роботи _____ Дрозденко О. М.

АНОТАЦІЯ

Робота присвячена розробці програмного модуля візуалізації сцени із складними дзеркальними поверхнями. Даний модуль призначений для роботи у режимі реального часу і повинен підтримувати високу якість візуалізації кривих та плоских дзеркал із урахуванням особливостей напівдзеркальних матеріалів. Також модуль повинен підтримувати множинне відбиття із як завгодно великою глибиною рекурсії.

У роботі наведено огляд існуючих рішень та виконано їх порівняльний аналіз. Із числа розглянутих рішень було обрано CubeMap Reflection та Stencil Buffer, які найкраще підходять для реалізації поставленої задачі зі встановленими вимогами. Роботу програмного модуля представлено у вигляді технологічної демонстрації.

Робота складається з вступу, 7 розділів та висновків, налічує 38 сторінок. Містить 20 ілюстративних матеріалів, 4 таблиці, 3 додатки та посилається на 11 літературних джерел.

Ключові слова: модуль візуалізації, дзеркало, режим реального часу, OpenGL, Stencil Buffer, CubeMap Reflection.

ABSTRACT

This work is dedicated to development of software module that can render a scene with a complex mirror surfaces. This module is designed to work in real-time and must maintain a high quality visualization of curves and flat mirrors allowing for the half-mirror materials. The module must support multiple reflection with arbitrarily large recursion depth.

The paper provides an overview of existing solutions and made a comparative analysis. Among solutions discussed were selected CubeMap Reflection and Stencil Buffer, which are best suited to accomplish the task with the requirements. The work program module is presented in the form of a technological demonstration.

The work consists of an introduction, 7 sections, includes conclusions and 38 pages, 20 illustrative materials, 4 tables, 3 appendices and has 11 references.

Keywords: visualization module, mirror real-time mode, OpenGL, Stencil Buffer, CubeMap Reflection.

Зміст

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

AABB — (з англ. axis-aligned bounding box) це обмежена область в просторі у вигляді прямокутного паралелепіпеда, зі сторонами, паралельними осям світової системи координат

FPS — (з англ. frames per second) кількість обрахованих кадрів за секунду

GPU — (з англ. graphics processing unit) графічний процесор, що знаходиться на відеадаптері

OpenGL — (з англ. Open Graphics Library) відкрита графічна бібліотека

Рендеринг — (англ. rendering) процес отримання зображення за моделлю з допомогою комп'ютерної програми

Тексель — (англ. Texture element) фундаментальна одиниця текстурного простору

Текстура — (англ. Texture) растрове зображення, що накладається на поверхню, масив текселів

Текстурна карта — (англ. texture-mapping) — спосіб нанесення на поверхню матеріалу

Технологічна демонстрація — (також технологічна демонстраційна версія, техно-демоверсія) прототип, наближений приклад чи неповна версія продукту, створена з метою продемонструвати ідею, продуктивність, метод або особливості якогось продукту або методу

Шейдер — (англ. Shader) це програма для одного із ступенів графічного конвеєра, що використовується в тривимірній графіці для визначення остаточних параметрів об'єкта чи зображення

ВСТУП

На сьогоднішній день в різних сферах людської діяльності все частіше виникає потреба у комп'ютерній графіці. Прикладами є комп'ютерне моделювання фізичних, погодних, сейсмічних явищ із наглядною візуалізацією, кіно-індустрія, індустрія відеоігор і навчальних симуляторів тощо.

В комп'ютерній графіці механізм побудови зображення (рендеринг) завжди грунтується на фізичній моделі, але через апаратні обмеження доводиться замінювати точні фізичні моделі на наближення, які простіші з точки зору обчислювальної складності. Один із найбільш точних методів побудови зображення грунтується на рівнянні рендерингу. Більшість існуючих технологій являють собою яке-небудь наближення до цього рівняння. Тобто, чим точніше передано всі фізичні закони - тим більш реалістичним буде здаватися зображення. Досі не існує оптимального методу, який би враховував всі складові рівняння рендерингу.

Метою даної роботи є розробка методу візуалізації складних дзеркальних поверхонь у просторі в реальному часі, який призначений для більшості сучасних платформ, що базуються на GPU (в тому числі і потужних мобільних платформ, які сьогодні є дуже розповсюджені і доступні).

1 ПОСТАНОВКА ЗАДАЧІ

Розробити програмний модуль для правдоподібного відображення дзеркальних поверхонь в просторі.

Результатом роботи є технологічна демонстрація.

Основні вимоги:

а) фотореалістичність відбиття (максимальне наближення до рівняння рендерингу)

б) підтримка великих плоских дзеркальних і напів-дзеркальних поверхонь

в) підтримка кривих дзеркальних поверхонь

г) підтримка множинного самовідбивання на довільну глибину

д) адаптивність для роботи на GPU (в реальному часі)

Вимоги до демонстрації:

а) робота на всіх платформах, що підтримують OpenGL 3.3 або OpenGL ES 2.0;

б) робота в реальному часі зі стабільним FPS>30

в) стабільна робота 720p/1080p

2 ОСНОВНА ЧАСТИНА

2.1 Фізична модель дзеркала

Світло

Світло - дуже складна система, щоб моделювати її повністю. Саме тому ми рідко можемо бачити створені комп'ютером тривимірні зображення, які були б по-справжньому фотореалістичними. У всіх випадках, чим складніше і реалістичніше створювана віртуальна сцена, тим більше обчислень, і тим повільніше вона буде виводитись на екран.

Світло складається з фотонів.

При зіткненні фотона з зовнішніми об'єктами може відбутися:

- відбиття (reflection) - фотон відскакує від поверхні
- поглинання (absorption) - фотон поглинається і віддає свою енергію об'єкту
- переломлення (refraction) - фотон проходить крізь об'єкт і змінює напрямок руху залежно від властивостей об'єкта та оточення
- відхилення (diffraction) - фотон може відхилитися і змінити напрямок у випадку, коли він проходить на дуже близькій відстані від поверхні об'єкту.

Оберненоквадратичне згасання світла (Inverse-square Light Falloff) означає, що зі збільшенням у два рази відстані від джерела світла до об'єкта, яскравість світла зменшується в 4 рази (формула 2.1).

$$Brightness = -\frac{k}{r^2} \tag{2.1}$$

Саме взаємодія світлового потоку з навколишніми об'єктами (предметами) дозволяє нам бачити їх. Зазвичай комп'ютери оперують зі світлом у вигляді величин, що визначають кількість червоного, зеленого і синього кольорів (RGB Model).

Дзеркало, відбиття

Дзеркало - гладка поверхня, яка відбиває світло і дозволяє отримати зображення предмета.

Коли світло відбивається від полірованої плоскої поверхні, кут падіння (від нормалі до поверхні) дорівнює куту відбиття (рис. 2.1), причому відбитий промінь, нормаль і падаючий промінь лежать в одній площині. Якщо на плоске дзеркало падає світловий пучок, то при відображенні форма пучка не змінюється; він лише поширюється в іншому напрямку. Тому, дивлячись у дзеркало, можна бачити зображення джерела світла (або освітленого предмета), причому зображення здається таким же, як і вихідний об'єкт, але перебувають за дзеркалом на відстані від об'єкта до дзеркала.

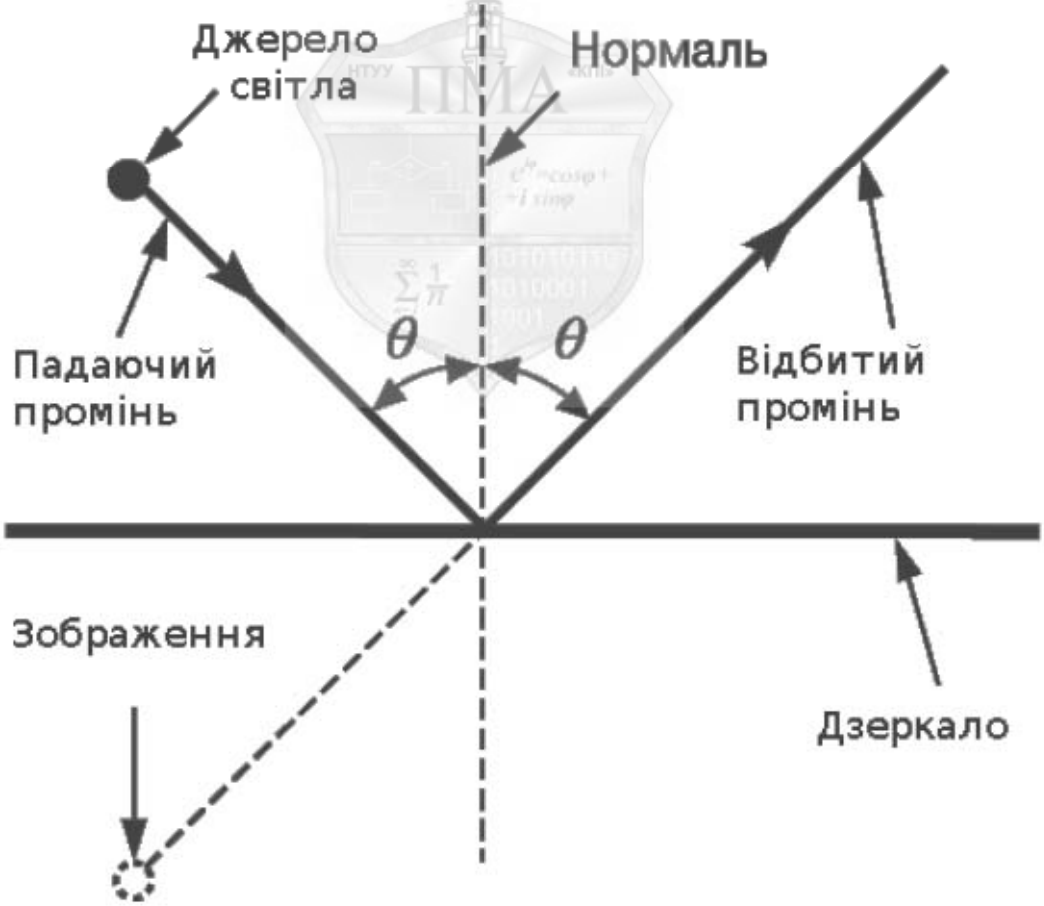

Рисунок 2.1 – Закон відбиття

Множинне відбиття

Коли два дзеркала повернуті одне до одного, зображення, що виникає в одному з них, відбивається в іншому, і виходить цілий ряд зображень, число яких залежить від взаємного розташування дзеркал. У разі двох паралельних дзеркал, коли об'єкт поміщається між ними, виходить нескінченна послідовність зображень, розташованих на прямій, перпендикулярній обом дзеркалам. Якщо два плоских дзеркала утворюють прямий кут, то кожне з двох первинних зображень відбивається в другому дзеркалі.

Відбиття від кривих поверхонь

Віддзеркалення від кривих поверхонь відбувається за тими ж законами, що і від прямих, причому нормаль в точці відображення проводиться перпендикулярно дотичній площині в цій точці. Найпростіший, але найважливіший випадок - відображення від сферичних поверхонь. У цьому випадку нормалі збігаються з радіусами [1].

Рівняння рендерингу

Основне рівння рендерингу (формула 2.2) — інтегральне рівняння, яке визначає кількість світлового випромінювання в певному напрямку як суму власного і відбитого випромінювання.

$$L_0(x, \omega_0, \lambda, t) = L_e(x, \omega_o, \lambda, t) + \int_\Omega f_r(x, \omega_i, \omega_o, \lambda, t) \cdot L_i(x, \omega_i, \lambda, t) \cdot (\omega_i \cdot n) d\omega_i \tag{2.2}$$

де:

- λ — довжина хвилі світла
- t — час
- x — розташування в просторі
- ω_o — напрямок вихідного світла
- ω_i — протилежний напрямок падаючого світла
- $L_0(x, \omega_o, \lambda, t)$ — випромінювання заданої довжини хвилі λ вздовж напрямку ω за час t із заданой точки x
- $L_e(x, \omega_o, \lambda, t)$ — випромінюване світло
- $\int_\Omega \cdots d\omega_i$ — інтеграл по полусфері вхідних направлень
- $f_r(x, \omega_i, \omega_o, \lambda, t)$ — двонаправлена функція розподілу параметру відбиття поверхні, частка світла відбивається від ω_i до ω_o
- $L_i(x, \omega_i, \lambda, t)$ — кількість енергії світла, яке прийшло до точки x з напрямку ω_i
- $(\omega_i \cdot n)$ — послаблення внутрішнього випромінювання згідно падаючого кута

Фізичною основою рівняння є закон збереження енергії.

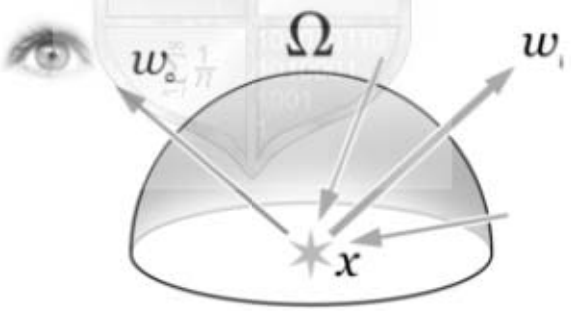

Рисунок 2.2 – Опис усього світлового потоку в заданій системі

Більш детальна інформація по цій темі [2].

Але вирішення рівняння в його прямому вигляді навіть при сьогоднішніх потужностях все ще складна задача. Тому, використовуються спрощені моделі та методи.

2.2 Огляд існуючих методів

2.2.1 Ray tracing

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, проте має дуже високу обчислювальну складність.

Алгоритм трасування променів дозволяє природнім чином отримати такі ефекти, які для інших алгоритмів рендеринга складають значну складність. Серед них правильне затінення, дзеркальні поверхні, заломлення світла. Завдяки цьому реалістичність сцен, обрахованих методом трасування променів, інколи сягає "фотографічної".

Також, на відміну від поширених полігональних алгоритмів, він дозволяє обробляти у сцені об'єкти фактично довільних геометричних форм, що можуть бути виражені математичним записом - справжні (а не апроксимованих 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.

Головним недоліком на теперішній час є надмірна часова складність алгоритму, яка перевищує можливості сучасних настільних і портативних комп'ютерів. Тому на сьогоднішній день рейтрейсинг не годиться на графіки в реальному часі (realtime) [3].

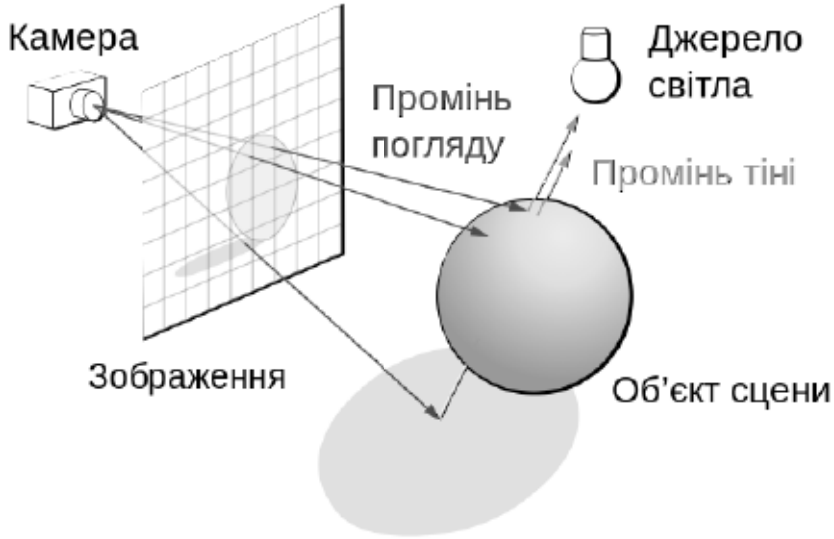

Рисунок 2.3 – Схема роботи алгоритму ray tracing

2.2.2 Voxel-Based Technology

Технології на основі вокселів уже давно використовуються в комп'ютерній графіці, проте їх використання обмежено через серйозні вимоги до апаратної частини.

Якщо полігональну сцену замінити на воксельну модель (наприклад, із використанням Sparse Voxel Octree [4]), то можна значно прискорити трасування променів. Недоліком є те, що для достатньо точного представлення сцени (особливо для точності до пікселя) потрібно багато памяті і є проблема представлення динамічних сцен.

Сьогодні воксельні технології застосовують для візуалізації низькочастотних ефектів, таких як наприклад Global Illumination [5].

2.2.3 CubeMap Reflection

Суть методу полягає у використанні кубічної карти для розрахунку відбитого світла текселя при обчисленні відбиття середовища на поверхні об'єкту.

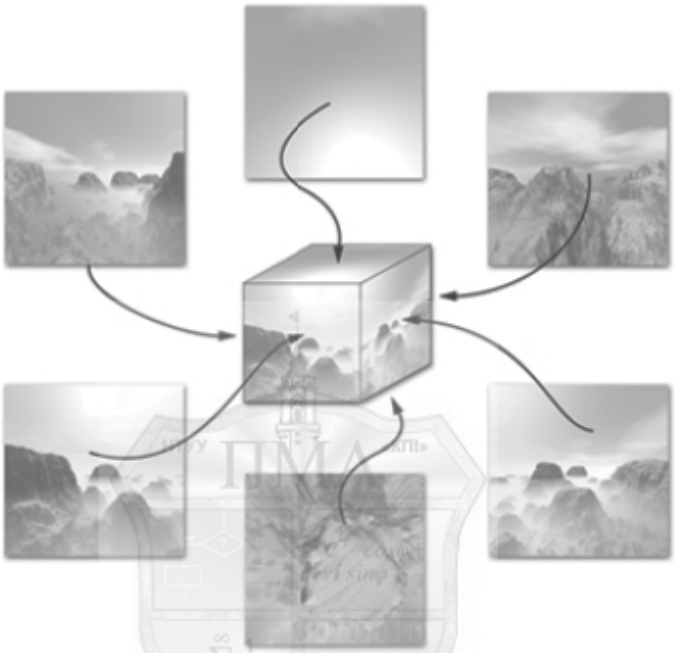

Рисунок 2.4 – Представлення текстури кубічної карти

Кубічна карта являє собою розгортку шести граней куба, кожна грань якого містить відповідну текстуру. Кожна текстура відображає вид оточення, яке видно з однієї точки зору в шести напрямках. Текстурна координата є вектором, який визначає, як дивитися з центру куба, щоб отримати бажаний тексель [6].

2.2.4 Stencil Buffer

Stencil buffer (буфер трафарету) - використовується для попіксельних ефектів. З його допомогою можна, наприклад, виводити пікселі довільного зображення лише в межах потрібного силуету (області). Ідеально підходить для відсікання зображення при малюванні плоских дзеркал по складному силуету, а також використовується у технології Stencil Shadows (побудова тіней, наприклад у двіжку id Tech 4).

Суть методу в тому, що для кожного пікселя зображення окрім даних про колір і глибину, вводять ще додаткові дані про кількість біт для трафарету. Ці додаткові дані використовуються для арифметичних і логічних розрахунків при візуалізації поверх відповідного пікселя. Так, можна, наприклад, не малювати ті пікселі, де значення трафарету буде рівне 0 і малювати там, де значення -1.

На рис. 2.5 зліва представлено плоске дзеркало та об’єкти. Зправа - відзеркалені об’єкти сцени.

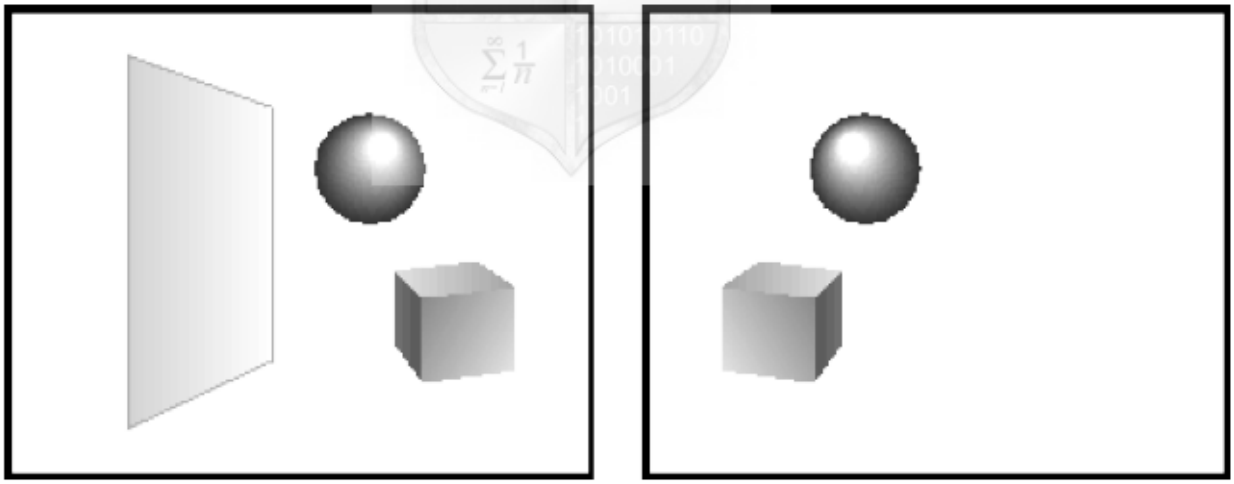

Рисунок 2.5 – Сцена та відзеркалені об’єкти

Далі (на рис. 2.5) представлено ідею роботи Stencil Buffer.

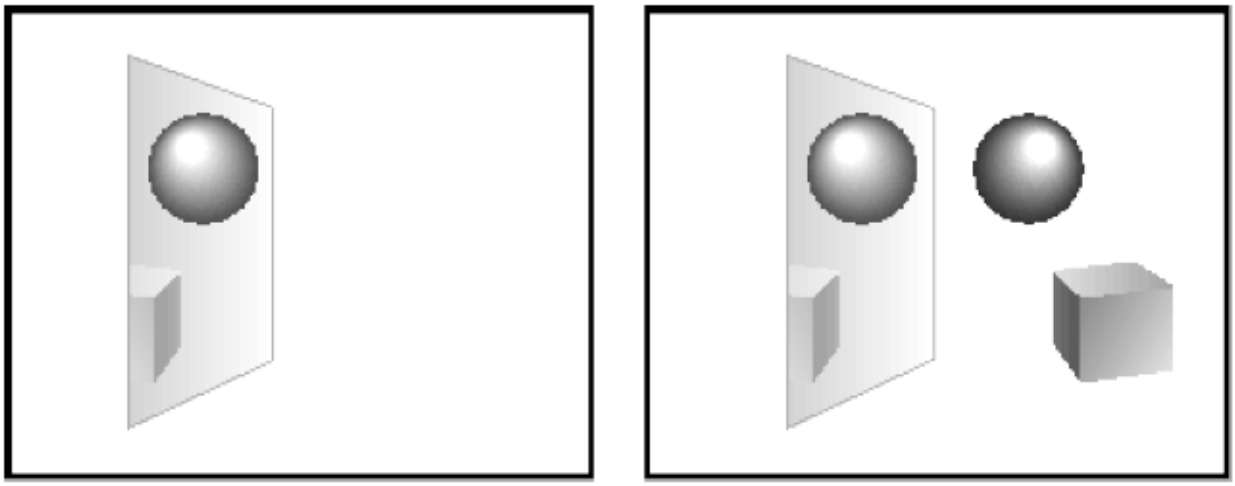

Рисунок 2.6 – Використання Stencil Buffer

2.2.5 Висновок

Таблиця 2.1 – Порівняння методів

	Робота realtime	Плоскі дзеркала	Криві дзеркала	Динамічні сцени
Ray tracing	-	+	+	+/-
Voxel-Based Technology	-	+/-	+/-	-
CubeMap Reflection	+	+	+	+
Stencil Buffer	+	+	+	+

З таблиці 2.1 видно, що методи **CubeMap Reflection** та **Stencil Buffer** найкраще підходять для кривих та плоских дзеркал, задовільняючи основним вимогам: швидкість та правдоподібність.

2.3 Побудова зображення сцени із дзеркалами

На рис. 2.7 показано схему побудови сцени із дзеркальними об'єктами:

Рисунок 2.7 – Побудова зображення сцени із дзеркалами

Вхідні дані:

- сцена
- камера
- матриця перетворення
- площина відсікання (фруструм)
- поточна глибина
- максимальна глибина

Вихідні дані:

Вихідними даними є 2-вимірне зображення.

2.4 Математична модель

Оператор переміщення всіх точок математичного об'єкта відповідно його дзеркального відображення:

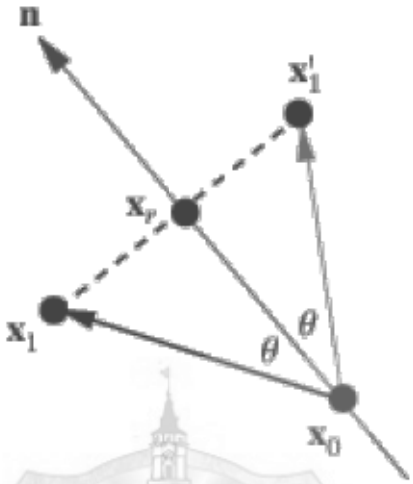

Рисунок 2.8 – Приклад відзеркалення 1

Розглянемо геометрію на рисунку 2.8 в якому точка x_1 відбивається в дзеркалі (синя лінія). Тоді:

$$x_r = x_0 + \hat{n}[(x_1 - x_0) \cdot \hat{n}]. \tag{2.3}$$

Знаходимо відбиття x_1':

$$x_1' = -x_1 + 2x_0 + 2\hat{n}[(x_1 - x_0) \cdot \hat{n}]. \tag{2.4}$$

Відображення може стосуватись відбиття м'яча, променя світла тощо. Як показано в на рисунку 2.9, відображення точки x_1 від стіни з вектором нормалі n задовольняє рівнянню:

$$x_1' - x_0 = v - 2(v \cdot \hat{n}) \cdot \hat{n}. \tag{2.5}$$

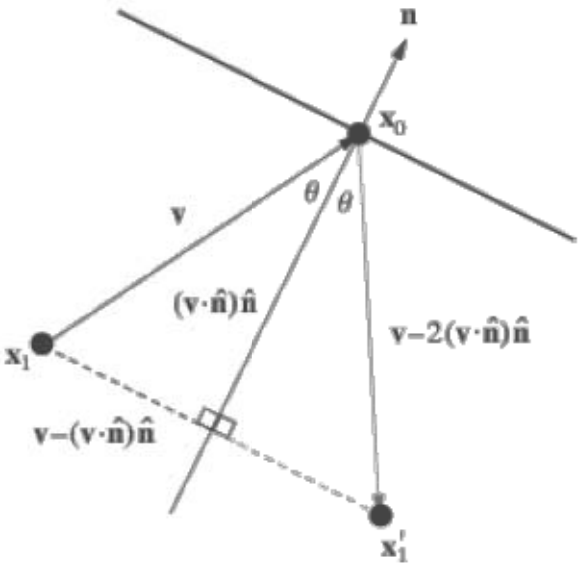

Рисунок 2.9 – Приклад відзеркалення 2

Щоб відобразити точку через площину $ax+by+cz=0$ (яка проходить через початок координат), можна використати формулу $A = I - 2NN^T$, де I є одиничною матрицею 3×3, а N є тривимірним одиничним вектором для вектора нормалі до площини. Якщо $L2$ норма a, b, c дорівнює одиниці, то матриця перетворення може бути виражена як:

$$A = \begin{bmatrix} 1-2a^2 & -2ab & -2ac \\ -2ab & 1-2b^2 & -2bc \\ -2ac & -2bc & 1-2c^2 \end{bmatrix} \tag{2.6}$$

Для того, щоб відобразити зображення плоского дзеркала, достатньо відобразити ту саму сцену, лише дзеркально відображену по площині дзеркала і обрізану по силуету плоского дзеркалала. Приклад - рис. 2.10.

Трансформація вектора виконується згідно формули:

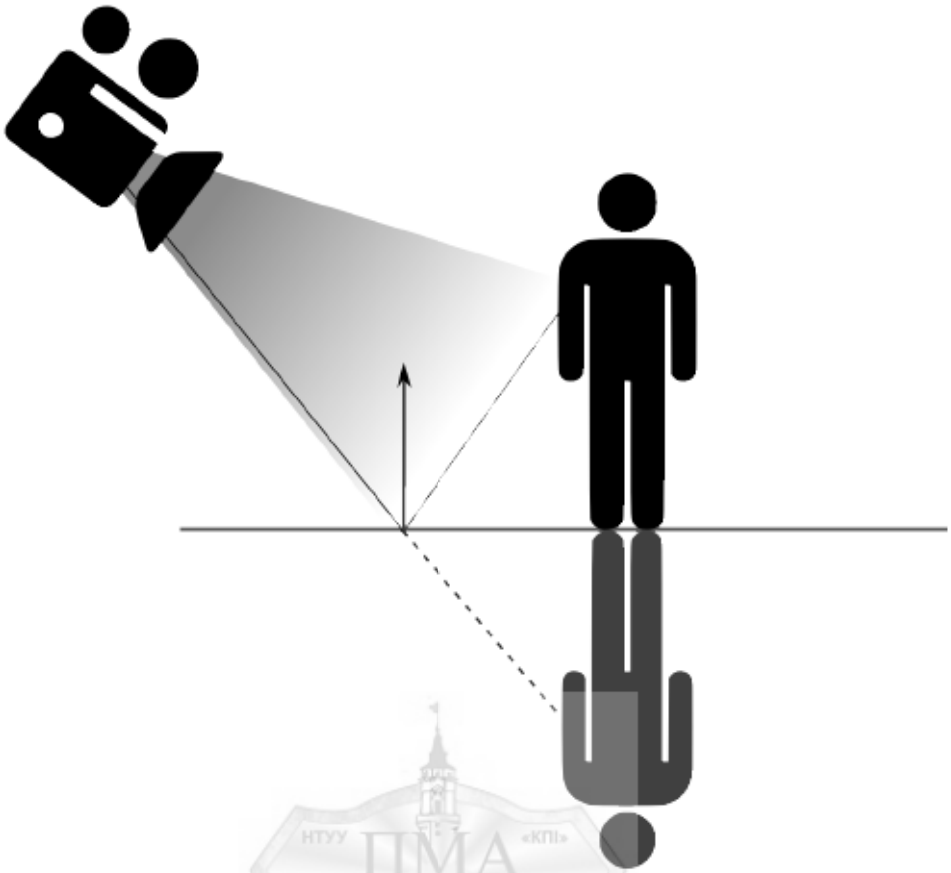

Рисунок 2.10 – Приклад віддзеркалення

$$TransformedVector = TranslationMatrix \cdot RotationMatrix \cdot \\ \cdot ScaleMatrix \cdot OriginalVector \tag{2.7}$$

Формула 2.7 показує, що спочатку виконується масштабування, потім поворот і тільки в останню чергу виконується перенос.

3D сцена - віртуальне оточення, являє собою вхід графічного конвеєру. Сцена складається з геометрії, матеріалу та освітлення.

Перехід "координати моделі (Model Coordinates) $\rightarrow$ світові координати (World Coordinates) $\rightarrow$ координати камери (Camera Coordinates) $\rightarrow$ однорідні координати (Homogeneous Coordinates)"відбувається за допомогою MVP матриці.

MVP - добуток матриць моделі (Model matrix), виду (View matrix) та

проекції (Projection matrix). Рисунок 2.11 ілюструє перетворення матриці.

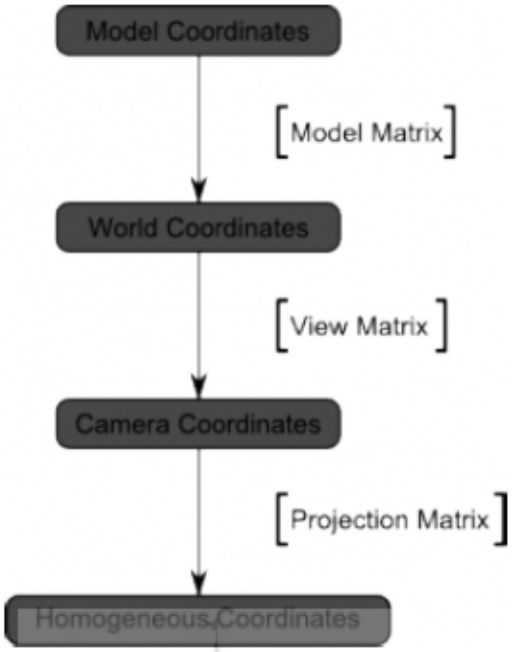

Рисунок 2.11 – Схема перетворень MVP матриці

Тобто, MVP матриця обчислюється за формулою:

$$MVPmatrix = projection \cdot view \cdot model \tag{2.8}$$

Використання MVP матриці:

$$transformed\_vertex = MVP \cdot in\_vertex; \tag{2.9}$$

2.5 Програмна реалізація

Програмний модуль повинен відповідати високим вимогам (розділ 1). Для написання основної частини програми було обрано мову C++ через:

- високу швидкодію

- підтримку всіх основних платформ

Огляд графічних бібліотек:

Таблиця 2.2 – Порівняння графічних бібліотек

	Мультиплатформенність	Актуальність	Швидкодія
OpenGL	+	+	+
Direct3D	±	+	+
Mantle	-	+	+

Обрано OpenGL через універсальність та роботу на багатьох платформах (включно з мобільними). Шейдерна мова — GLSL 330 (OpenGL 3.3 — актуальна та стабільна версія).

Для підтримки матриць, векторів, кватерніонів (базової математики) було написано свої класи:

- vec2
- vec3
- mat3x3
- mat4x4
- Quaternion

Додаткові бібліотеки, які використовувались для написання технологічної демонстрації:

- tinyXML — завантаження моделей (зчитування xml);
- DevIL — підтримка деяких форматів зображень та музики
- SDL2.0 — створення вікна

2.5.1 Структура програми

На рисунку 2.12 наведено UML діаграму технологічної демонстрації.

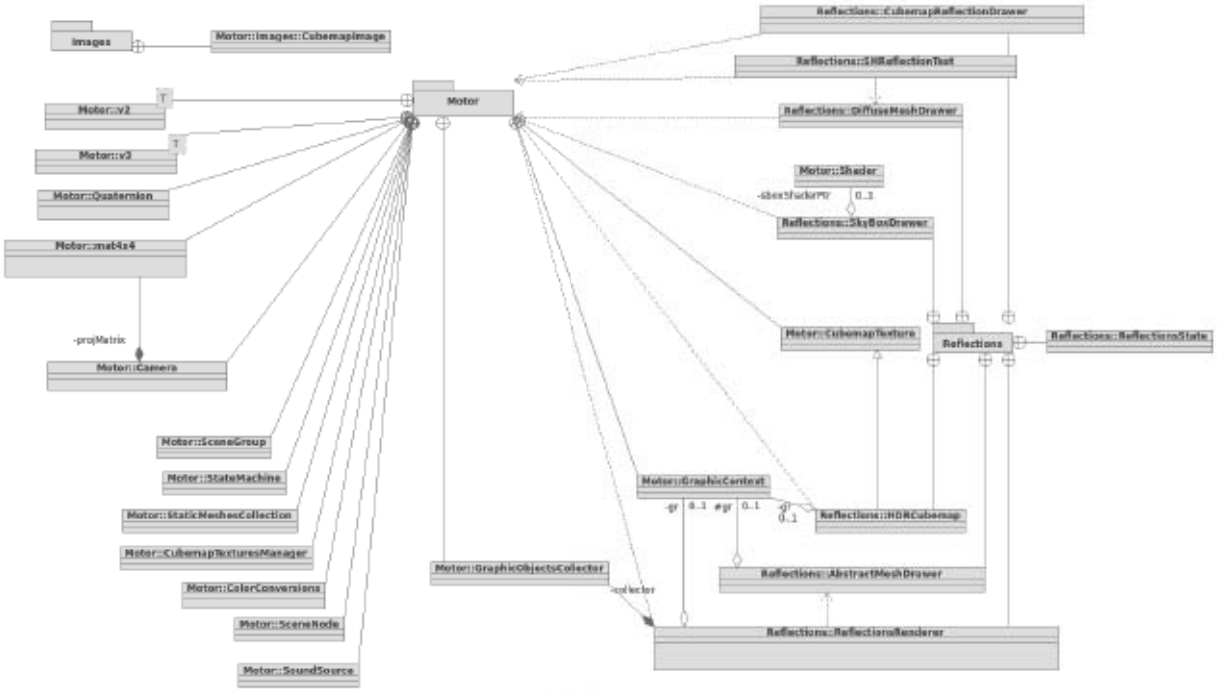

Рисунок 2.12 – UML діаграма програми

2.5.2 Карта нормалей

Карта нормалей (Normal Map) — це текстура, яка контролює напрямок відбиття світла. Використовується дана технологія для створення високодеталізованих низькополігональних моделей.

Для створення карти нормалей необхідно мати високополігональну і низькополігональну модель. Під час створення текстури інформація з високополігональної моделі інтерпритується в RGB складову кольору, де [r, g, b] співвідносяться з осями [x, y, z].

2.5.3 Граф сцени

View frustum (піраміда видимості) - це частина простору, в якій знаходяться всі об’єкти, видимі з даної точки в даний момент. Вона визначається

6 гранями усіченої піраміди (тобто піраміди зі зрізаною вершиною). Якщо якась точка знаходиться всередині піраміди видимості, її видно. Якщо поза піраміди, значить, цю точку не видно.

Незважаючи на те, що перевірка і відсікання здійснюється на рівні графічного API (як правило, апаратно), на більш високому рівні програміст може використовувати знання про структуру сцени для перевірок і відсікання великих груп вершин, розташованих локально. Для цього, як правило, на видимість перевіряються тільки нескладні геометричні фігури, що складаються з декількох вершин, і, якщо вся фігура виявляється поза видимості, то частина сцени, укладена в неї, може бути відкинута. У ролі таких фігур можуть виступати сфери, паралелепіпеди (OOB, AABB [9]) тощо.

При візуалізації досить великих полігональних об'єктів далеко не всі вершини реально видно на екрані. Не відображуючи ці вершини можна сильно заощадити на швидкості рендеринга. Важливим кроком оптимізації є представлення сцени у вигляді деревовидної структури, як показано на рис. 2.13. Вершини можуть бути 2 типів: групи, та об'єкти сцени. Кожна вершина дерева має свою позицію і орієнтацію, а також обмежуючий об'єкт (сферу).

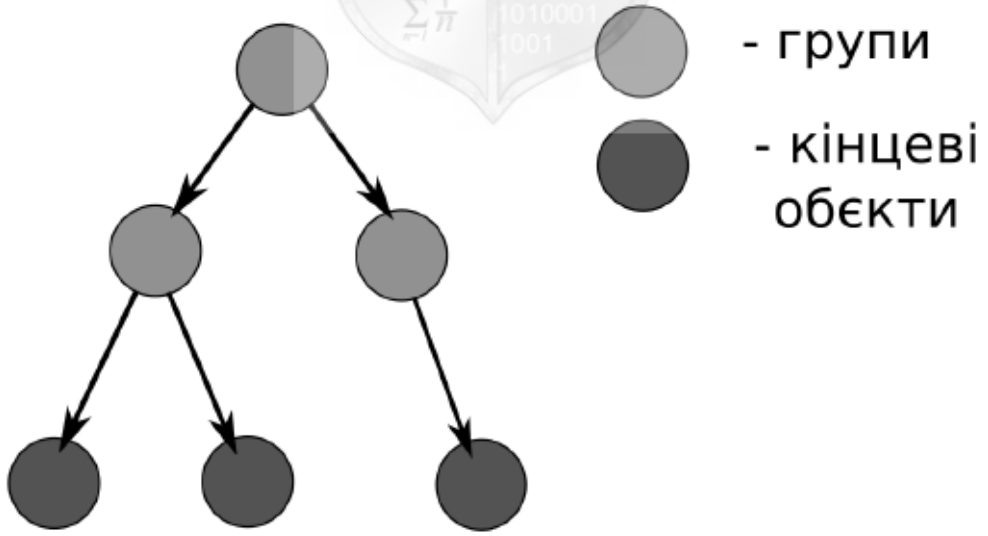

Рисунок 2.13 – Схема графу

Під час обходу сцени всі дочірні обєкти трансформуються із урахуванням перетворення батьківського обєкту. Головна властивість такого дерева

- якщо в камеру не видно групу - то не видно і всі елементи групи, тому їх можна не малювати [9]. Це дозволяє ефективно відсікати невидиму геометрію, як це показано на рис. 2.14.

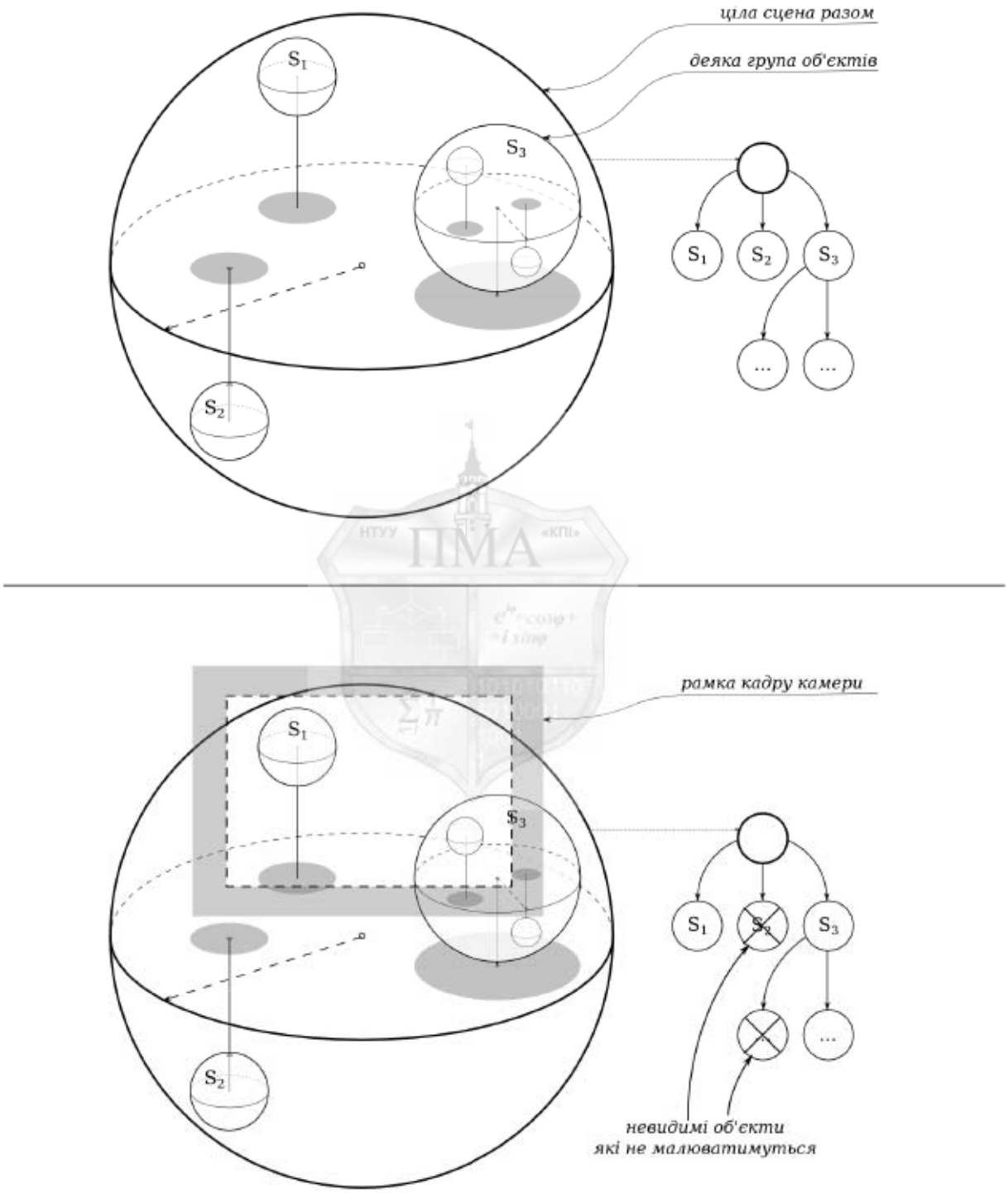

Рисунок 2.14 – Відсікання гілок графу

2.6 Висновок

Було розглянуто фізичну основу світла, його поведінку в просторі, взаємодію з об'єктами різної форми та матеріалів. Вибрано спрощені моделі даних фізичних процесів для кривих та плоских дзеркал, з врахуванням дзеркальних та напівдзеркальних поверхонь. Складено метод для візуалізації сцени із дзеркалами. Описано математичну модель, та основні матричні перетворення. Розглянуто критичні моменти в реалізації технічної демонстрації.

3 ТЕСТУВАННЯ ТА РЕЗУЛЬТАТИ

Оскільки, всі основні операції виконуються на відеокарті і в комп'ютерах потужності комплектуючих співставлено (не виникає ефекту bottleneck), приводиться детальна таблиця порівняння результатів на доступних відеокартах. Технологічну демострацію вдалось протестувати на 3 відеоприскорювачах, всі від компанії NVIDIA. ОС — GNU/Linux (Ubuntu 14.04, x64).

Таблиця 3.1 – Таблиця результатів

	min FPS	max FPS	середній FPS
GTX 770 (1080p)	25	185	85
GTX 770 (720p)	32	212	120
GTX 335M (720p)	15	97	48
GT 9600M (720p)	13	108	55

Як бачимо з таблиці, програмний модуль працює з прийнятною частотою як на відносно нових (GTX 770) так і на доволі старих мобільних відеоприскорювачах середнього сегменту (GTX 335M та GT 9600M). Найскладніша сцена - калейдоскоп. На ній і було отримано мінімальні показники FPS. Тестування проводилось з драйвером версії 340.76 (NVIDIA).

На рис. 3.1 показано множинне відбиття. Дзеркала розташовані навпроти. Можна помітити, що поточне налаштування максималнього відбиття - 10. Параметр можна змінювати в залежності від потужностей відеокарти та складності сцени.

Перед порівнянням можливостей технологічної демонстрації Chirality та гри Portal варто вказати основну відмінність:

В Chirality дзеркала - результат дії матриці відзеркалення. Звязані портали у грі Portal працюють на матрицях повороту та переміщення. Варто

Рисунок 3.1 – Chirality, множинне відбиття

Рисунок 3.2 – Зв'язані портали з гри Portal

зазначити, що Chirality теж можна розшири до відображення зв'язних потралів.

Таблиця 3.2 – Порівняння можливостей

	Portal	Chirality
Відкритість методу	-	+
Підтримка кривих поверхонь	-	+
Поверхня матеріалу	-	+
Кінематика	+	-

Зв'язні портали є дуже оптимізованим і вузьконаправленим методом, орієнованим тільки на відображення порталу та збереження імпульсу об'єкта при переміщенні крізь портал. Також, не враховується матеріал поверхні.

3.1 Висновок

Розроблений модуль відзеркалення є універсальним методом, який можна використовувати для підвищення реалістичності зображення в довільному візуалізаторі (інтеграція) та розширити додатковим функціоналом. В таблиці 3.2 показано, що розроблений модуль має переваги над вже існуючими схожими методами.

Метод в повній мірі відповідає поставленим вимогам. Варто виділити швидкодію програмного модуля на слабких та актуальних відеоприскорювачах (табл. 3.1), враховуючи задовільно-реалістичні ефекти віддзеркалення.

ВИСНОВКИ

Візуалізація дзеркальних поверхонь в середовищі у режимі реального часу - це складна і невичерпна задача. При її розв'язанні завжди доводиться робити спрощення реальних фізичних моделей і замінювати їх на більш простіші, з точки зору обчислювальної складності, моделі. При моделюванні такого роду процесів завжди приходиться шукати золоту середину між точністю і простотою математичних моделей. А інтегральний характер головних рівнянь завжди спонукає до детального аналізу задачі, для того, щоб можна було максимально спростити математичну модель і при цьому не втратити її адекватність і отримати реалістичний результат.

У даній роботі було написано програмний модуль для візуалізації складних дзеркальних поверхонь в середовищі для роботи у режимі реального часу. Розглянуто фізичні властивості світла та дзеркальних і напівдзеркальних поверхонь, та вибрано спрощені моделі даних фізичних процесів. При розгляді методів та існуючих реалізацій відображення графіки з використанням дзеркал, виявлено, що даний ефект майже не використовується і тому він зазвичай або не підтримується візуалізаторами або ж є погано оптимізованим і тому складні випадки виводяться із помітними помилками. Це можна вважати важливою причиною розробки представленого модуля, який є орієнтованим на візуалізацію складних дзеркальних поверхонь у просторі. Серед розглянутих існуючих методів швидкої візуалізації дзеркал було обрано алгоритми CubeMap Reflection та Stencil Buffer, які найкраще підходять для кривих та плоских дзеркал, легко реалізуються, є дуже швидкими та правдоподібними.

Розроблений програмний модуль показав хороші результати і задовольнив усі поставлені вимоги. Виділено переваги, недоліки, шляхи покращення та розширення модуля. У додатках даної роботи наведено лістинги вихідних кодів основного модуля (додаток Б), та ілюстративні матеріали (додатки А та В).

ПЕРЕЛІК ПОСИЛАНЬ

1. Ландсберг Г. С. Оптика. — М.: Мир, 1976. - 720 с.
2. Kajiya, James T. The rendering equation. — Siggraph 1986. - 143 с.
3. Jason Gregory Game Engine Archtecture — Boston: TBooks. - 2012. - 520 с.
4. Laine, Samuli; Tero Karras Efficient Sparse Voxel Octrees – Analysis, Extensions, and Implementation — NVIDIA Corporation, 2010
5. Alexey Panteleev. PRACTICAL REAL TIME VOXEL BASED GLOBAL ILLUMINATION FOR CURRENT GPUS — NVIDIA — Режим доступу: http://on-demand.gputechconf.com/gtc/2014/presentations/S4552-rt-voxel-based-global-illumination-gpus.pdf
6. The Cg Tutorial, Chapter 7 — NVIDIA Corporation. [Електронний ресурс]. — Режим доступу: developer.nvidia.com/CgTutorial/cg_tutorial_chapter07.html
7. Richard S. Wright, Nicholas Haemel, Graham Sellers. OpenGL SuperBible Comprehensive Tutorial and Reference (5th Edition) — USA: E. Wesly. - 2010 - 674 с.
8. Philip J. Schneider David H. Eberly Geometric tools for computer graphics. — 2002. - 438
9. Efficient Collision Detection of Complex Deformable Models using AABB Trees — Gino van Bergen — Режим доступу: http://www.cs.cmu.edu/djames/pbmis/etc/jgt98deform_AABB.pdf
10. Stefan Zerbst with Oliver. Duvel 3D game engine programming. — Germany: GSky. - 2006 - 367 с.
11. John Lekner. Theory of reflection : of electromagnetic and particle waves — USA : Distributors for the U.S. and Canada, Kluwer Academic Publishers, 1987. - 742 с.