Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Моделювання хмарних об'єктів в системах візуалізації для авіаційних тренажерів

(тема)

Виконав:
студент II курсу, групи КСМм-19-1
Журавльов Д.Є.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерні системи та мережі
(повна назва освітньої програми)

Керівник: доц. Ляшенко О.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп’ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Журавльову Дмитру Євгеновичу
(прізвище, ім’я, по батькові)

1. Тема роботи Моделювання хмарних об'єктів в системах візуалізації для авіаційних тренажерів

затверджена наказом по університету від “ 30 ” жовтня 2020 р. № 1487 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 14 грудня 2020 р.

3. Вхідні дані до роботи

Тривимірна комп’ютерна графіка.

Метод трасування променів.

Алгоритм моделі хмар.

4. Перелік питань, що потрібно опрацювати в роботі

Огляд літератури, постановка завдання дослідження.

Розробка моделі хмар.

Розробка спостерігача на основі метода трасування променів.

Реалізація алгоритму обробки метеоумов.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд презентація – 11 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз літератури що до розвитку та	03.11.20 - 10.11.20	
	застосування комп'ютерної графіки		
2	Аналіз та розробка моделі хмар	11.11.20 - 20.11.20	
3	Розробка спостерігача на основі метода	21.11.20 - 30.11.20	
	трасування променів		
4	Оформлення пояснювальної записки та	01.12.20 - 11.12.20	
	демонстраційних матеріалів		

Дата видачі завдання 02 листопада 2020 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Гусятін В.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 69 с., 16 рис., 1 дод., 15 джерел.

ВІЗУАЛІЗАЦІЯ, МОДЕЛЮВАННЯ, ХМАРА, ТРАСУВАННЯ, ПРОМІНЬ, КОМП'ЮТЕРНА ГРАФІКА, СВІТЛО, РЕАЛІЗМ, МЕТАСФЕРА.

Метою атестаційної роботи є моделювання хмарних об’єктів для систем візуалізації і підвищення їх реалістичності.

У ході виконання атестаційної роботи був проведений аналіз існуючих методів візуалізації, розроблена модель та алгоритм, що дозволяє прискорити роботу вже існуючих методів синтезу зображень і навіть підвищити їх реалістичність.

Кінцевим результатом є швидка, візуально реалістична система, яка дозволяє в реальному часі проводити моделювання хмарних об’єктів.

ABSTRACT

Master’s thesis: 69 pages, 16 figures, 1 appendice, 15 sources.

RENDERING, SIMULATION, CLOUD, TRACING, RAY, COMPUTER GRAPHICS, LIGHT, REALISM, METASPHERE.

The purpose of the certification work is to model cloud objects for visualization systems and increase their realism.

During the attestation work the analysis of the existing methods of visualization was carried out, the model and the algorithm which allows to accelerate work of already existing methods of synthesis of images and even to increase their realism were developed.

The end result is a fast, visually realistic system that allows real-time modeling of cloud objects.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЕОМ – електронна обчислювальна машина

МДВ – метеорологічна дальність видимості

СВ – система візуалізації

GLSL – мова шейдерів графічної бібліотеки (англ., Graphics Library Shader Language)

GPU – графічний процесор (англ., Graphics Processing Unit)

OpenGL – відкрита графічна бібліотека (англ. Open Graphics Library)

2D – двомірна (англ., two-dimensional)

3D – тривимірна (англ., three-dimensional)

ВСТУП

Хмари – це всюдисуще, вражаюче та постійно мінливе явище. Вони є невід'ємним фактором погодних систем Землі та показником погодних моделей. Їх наявність та властивості важливі для польотів та польотних тренувань. Хмари є важливою складовою візуального моделювання будь-якої сцени на відкритому повітрі, але складність утворення хмари, динаміки та взаємодії світла ускладнює моделювання та надання хмарам реалістичного виду в реальному часі. В інтерактивному моделюванні польоту користувачі хотіли б літати в реальних об'ємних хмарах та навколо них, а також бачити, як інші літаки пролітають всередині них і позаду. В ідеалі імітовані хмари могли б рости і розсіюватися, як справжні хмари, і рухатися у відповідь на вітер та інші сили. Імітовані хмари повинні бути реалістично освітлені прямими сонячними променями, внутрішнім розсіюванням і відблисками від неба і землі знизу. Існуючі методи моделювання реального часу не надають користувачам такого досвіду.

Інтерактивні 3D-графічні середовища, такі як ігри та імітатори польотів, стають дедалі візуально реалістичнішими, зокрема завдяки потужності графічного обладнання. Швидкість і паралельність графічного обладнання дозволяють моделювати динаміку хмар в режимі реального часу. Однак у цих додатках часто бракує насичених динамічних явищ, таких як рідина, хмари та дим, які є поширеними в реальному світі. Великий обсяг роботи з комп'ютерній графіці був присвячений моделюванню природної динаміки. Однією з цілей є прискорення таких фізичних явищ, виконуючи обчислення на графічному обладнанні. Графічне обладнання – це ефективний процесор, він може використовувати текстурні зображення як вхідні дані, а зображення виводити за допомогою візуалізації.

Хмари поглинають дуже мало світлової енергії. Натомість кожна крапля води відбиває або розсіює майже все падаюче світло. Хмари складаються з

мільйонів цих крихітних крапель води. Майже кожен фотон, який потрапляє в хмару, розсіюється багато разів, перш ніж він покине хмару. Світло, що виходить із хмари, досягає очей і тому відповідає за формування її зовнішнього вигляду. Точне формування зображень хмар вимагає моделювання багаторазового розсіювання світла, що відбувається всередині них. Складність розсіювання унеможливлює вичерпне моделювання. Натомість для зменшення складності моделювання необхідно використовувати апроксимацію, тобто наближення.

Після ефективного обчислення динаміки та освітленості хмар залишається завдання створення хмарного зображення. Напівпрозора природа хмар означає, що їх не можна представити у вигляді простих геометричних тіл, як багатокутні моделі, які зазвичай використовуються в комп'ютерній графіці. Натомість потрібно використовувати об'ємне подання для фіксації змін щільності в хмарі. Візуалізація таких об'ємних моделей вимагає значних обчислень для кожного пікселя зображення. Це обчислення може призвести до надмірного часу візуалізації для кожного кадру.

Системи синтезу візуальної обстановки повинні забезпечувати формування високореалістічного зображення. Відображення таких природних явищ, як вода, туман, вогонь, хмари, значно підвищують реалістичність синтезованого зображення. Відпрацювання туману і хмарного шару необхідно в тренажерах літальних апаратів, де політ в умовах поганої видимості є одним з елементів програми навчання пілотів. В роботі пропонується метод формування тривимірної моделі хмари, що дозволяє при мінімальній кількості параметрів формувати реалістичне зображення хмарного шару.

Об'ємний рендеринг хмарного шару в реальному часі є привабливою функцією для мультимедійних додатків, таких як комп'ютерні ігри, СВ для авіаційних тренажерів та різні сценарії на відкритому повітрі. Досягнення реалістичних результатів вимагає передового програмного забезпечення і масивного багатоядерного графічного обладнання, яке використовується в

індустрії анімації, але є дорогим і часто не працює в режимі реального часу. Пропонується підхід, що забезпечує ефективну реалізацію об'ємного хмарного рендерингу в реальному часі за рахунок абстрагування суті математики та фізичної складності атмосферних моделей і використання переваг парадигм розширеного паралельного програмування GPU.

Візуалізація об'ємних хмар – це процес, що вимагає великого обсягу обчислень, що ускладнює його використання в додатках реального часу. У той же час необхідність в об'ємних хмарах очевидна, оскільки розробники шукають нові способи, щоб підвищити реалістичність. Скайбокси і плоскі текстури добре працюють в 3D-сценах, де передбачається, що камера знаходиться далеко від хмар і не переміщається на великі відстані. Але в реалізаціях з відкритим світом, де неможливо припустити положення камери, скайбокси створюють статичне зображення. В роботі досліджуються методи економії часу обчислень при реалізації об'ємних хмар для рендерингу в реальному часі.

1 КОМП’ЮТЕРНА ГРАФІКА ТА ЇЇ ВИКОРИСТАННЯ

1.1 Поняття комп’ютерної графіки

Комп’ютерна графіка – це галузь інформатики, яка займається створенням зображень за допомогою комп’ютерів. Сьогодні комп’ютерна графіка є основною технологією цифрової фотографії, кіно, відеоігор, мобільних телефонів, комп’ютерних дисплеїв та багатьох спеціалізованих програм. Розроблено велику кількість спеціалізованого обладнання та програмного забезпечення, причому дисплеї більшості пристроїв керуються апаратним забезпеченням комп’ютерної графіки. Це величезна і не так давно розроблена область інформатики.

Деякі розділи комп’ютерної графіки включають дизайн інтерфейсу користувача, графіку спрайтів, візуалізацію, трасування променів, обробку геометрії, комп’ютерну анімацію, векторну графіку, тривимірне моделювання, шейдери, дизайн графічного процесора, неявну візуалізацію поверхні, обробку зображень, обчислювальну фотографію, наукову візуалізацію, обчислювальну геометрію та комп'ютерний зір, тощо. Загальна методологія сильно залежить від основних наук геометрії, оптики, фізики [1].

Комп’ютерна графіка відповідає за ефективне відображення даних зображень для споживачів. Вона також використовується для обробки даних зображень, отриманих з фізичного світу, таких як фото та відео. Розвиток комп’ютерної графіки справив значний вплив на більшість засобів масової інформації та зробив революцію в анімації, фільмах, рекламі, відеоіграх та графічному дизайні загалом.

Для візуалізації даних розроблено багато інструментів. Комп’ютерні зображення можна класифікувати на кілька різних типів: двовимірні (2D), тривимірні (3D) та анімовані графічні зображення. По мірі вдосконалення технологій 3D-комп’ютерна графіка стала більш поширеною, але 2D-

комп’ютерна графіка все ще широко використовується. Комп’ютерна графіка виникла як підгалузь інформатики, яка вивчає методи цифрового синтезу та маніпулювання візуальним контентом. За останнє десятиліття були розроблені інші спеціалізовані галузі, такі як візуалізація інформації та наукова візуалізація, яка більше є візуалізацією тривимірних явищ архітектурних, метеорологічних, медичних, біологічних, де акцент робиться на реалістичному відображенні об'ємів , поверхні, джерела освітлення тощо.

1.2 Види комп’ютерної графіки

Існує чотири види комп'ютерної графіки, які відрізняються принципами зберігання і формування зображення: растрова, векторна, фрактальна, тривимірна.

У растровій графіці зображення зберігається у вигляді мозаїки з точок, де кожна точка має свій колір. Растровими зображеннями є цифрові фотографії, скановані ілюстрації. Такі зображення рідко створюються «з нуля». Тому програми-редактори растрової графіки орієнтовані не на створення зображень, а на їх обробку.

Перевагами растрової графіки є:

- растрова графіка дозволяє створити рисунок будь-якої складності;
- складні зображення обробляються швидко, якщо вони не вимагають масштабування;
- растровий формат є природним для більшості пристроїв ввіду-виводу (моніторів, принтерів, сканерів), так як зображення на цих пристроях теж формується з пікселів.

Але растрова графіка також має свої недоліки:

- навіть просте зображення буде мати великий розмір файлу;
- масштабування погіршує якість зображення, що відображено на рисунку 1.1;
- неможливий вивід на окремі пристрої друку.

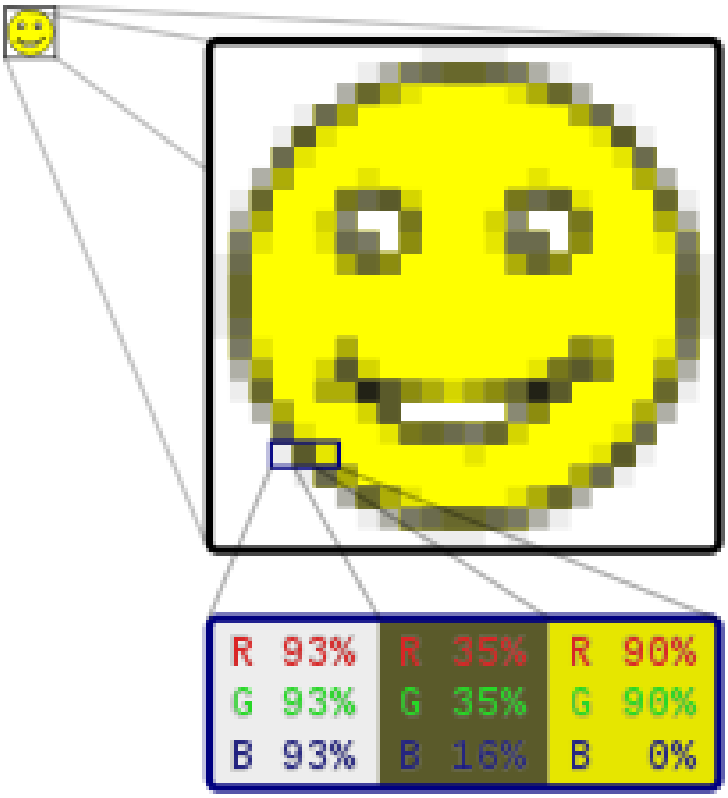

Рисунок 1.1 – Приклад растрового зображення

З растровими зображеннями працюють такі графічні редактори як Adobe Photoshop, GraphicsMagick, ImageMagick. Растрові зображення зберігають в стислому вигляді. Існує два типи стиснення: стиснення без втрат і стиснення з втратами. Стиснення з втратами передбачає деяку втрату якості при відновленні після стиснення. Однак, передбачається, що ця втрата якості повинна знаходитися в деяких допустимих межах. Людське око не повинно бачити суттєвої різниці зображення до і після стиснення.

Векторна графіка представляє зображення у вигляді сукупності дуже простих геометричних об'єктів. Такі об'єкти є базовими для побудови зображення і називаються примітивами. Примітивами можуть бути відрізки, маленькі дуги, кола, сплайни і т.д. Графіка називається векторною тому, що набір примітивів, які формують даний графічний об'єкт, називається вектором.

Векторна графіка широко використовується, наприклад, для малювання популярних в мережевому спілкуванні смайлів.

Основні переваги векторної графіки:

- масштабування зображення не викликає спотворень у відмінності від растрового зображення, що показано на рисунку 1.2;
- обсяг графічного файлу невеликий;
- частини зображення можна редагувати незалежно один від одного;
- висока точність прорисовування.

Недоліки векторної графіки:

- невелика кількість реальних предметів, які можна відобразити за допомгою такої графіки;
- зображення мають не достатньо реалістичний вигляд.

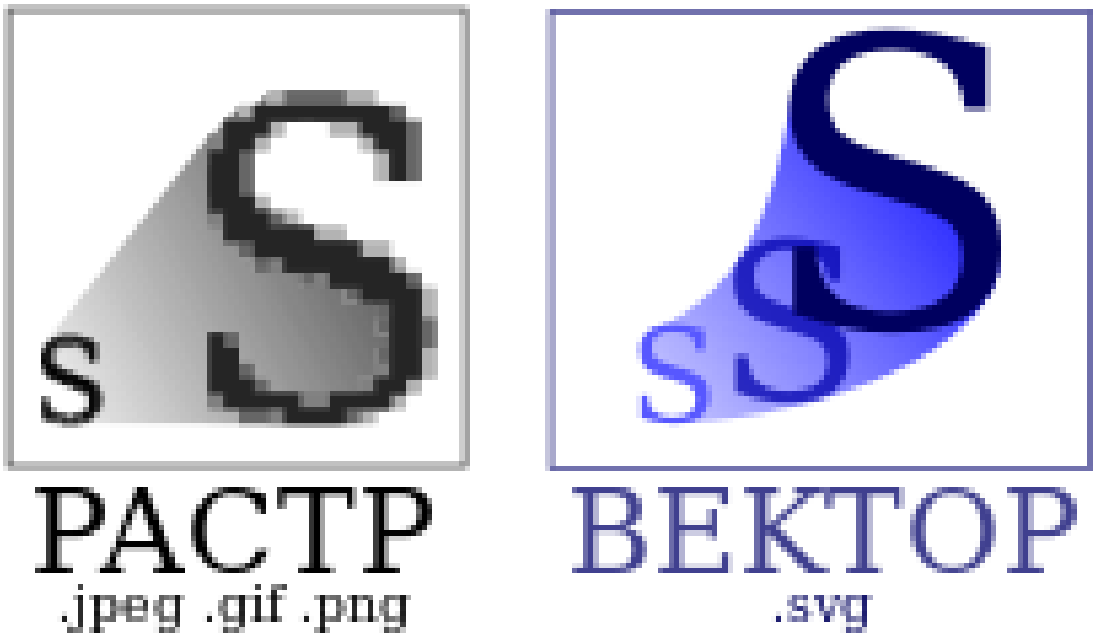

Рисунок 1.2 – Порівняння растрового та векторного зображення

Векторні зображення можна створювати в таких редакторах як CorelDraw, InkScape.

Фрактальна графіка є одним з перспективних напрямків комп'ютерної

графіки. Вона заснована на розділі математики – фрактальній геометрії. Термін фрактал ввів французький математик Бенуа Мандельброт. Цим терміном він назвав геометричну фігуру, яка складається з частин, подібних цілій фігурі.

Таким чином, головна властивість фракталів – це самоподібність. У фракталів збільшені частини фігури подібні всій фігурі і один одному. Таким чином, навіть якщо взяти невелику частину фігури, то по ній можна отримати всі зображення виходячи з міркувань подібності. На рисунку 1.3 показано послідовна побудова відомого фрактала «Крива Коха» по невеликому фрагменту.

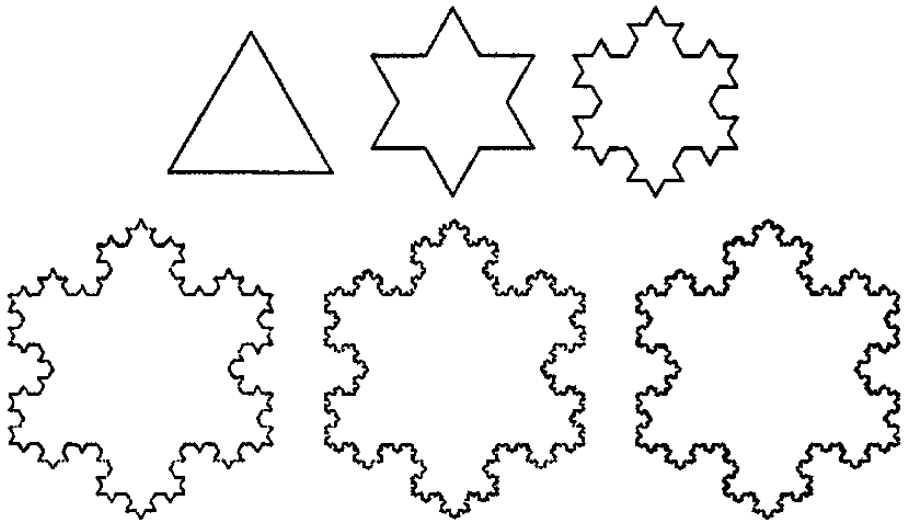

Рисунок 1.3 – Приклад фрактального об'єкта

Фрактальна графіка дозволяє створювати дуже красиві і складні абстрактні композиції. Крім абстрактних зображень фрактальная графіка незамінна при створенні зображень різних поверхонь: вода, гори, хмари. Для створення фрактальних зображень використовуються редактори Art Dabbler, Fractal Explorer, Chaos Pro, Apophysis, Mystica [2].

Тривимірна графіка оперує з об'єктами в тривимірному просторі. Для побудови зображення, яке виглядає як об'ємне, використовується так зване

полігональне моделювання. Для цього поверхню об'єкта представляють у вигляді простих двовимірних геометричних фігур. Вони називаються полігонами. Слово polygon в перекладі з англійської означає багатокутник. У комп'ютерних іграх в якості полігонів найчастіше використовуються трикутники, так як саме трикутники обробляються з найвищою швидкістю. Для інших цілей використовуються інші багатокутники.

1.3 Тривимірна комп’ютерна графіка

3D-комп’ютерна графіка або тривимірна комп’ютерна графіка – це графіка, що використовує тривимірне представлення геометричних даних (часто декартових), які зберігаються в комп’ютері для проведення обчислень та візуалізації 2D зображень. Отримані зображення можна зберігати для подальшого перегляду , наприклад як анімацію, або відображати в режимі реального часу. На відміну від тримірного кіно та подібних прийомів, результат є двовимірним, без ілюзії твердості.

Тривимірна комп’ютерна графіка опирається на ті ж самі алгоритми, що і 2D комп’ютерна векторна графіка в каркасній моделі, і 2D комп’ютерна растрова графіка на кінцевому візуалізованому об'єкті. У програмному забезпеченні для комп’ютерної графіки програми 2D графіки можуть використовувати тривимірні технології для досягнення таких ефектів, як освітлення, і аналогічним чином 3D можуть використовувати деякі методи 2D візуалізації [3].

Об'єкти в 3D-комп'ютерній графіці часто називають 3D-моделями. На відміну від візуалізованого об’єкту, дані моделі містяться у графічному файлі даних. 3D-модель – це математичне зображення будь-якого тривимірного об’єкта, модель технічно не є графікою, поки вона не відображається. Модель може бути відображена візуально у вигляді двовимірного зображення за допомогою процесу, який називається 3D-рендеринг або тривимірна комп’ютерна візуалізація, або може бути використана в неграфічних

комп'ютерних моделюваннях та розрахунках. За допомогою 3D-друку моделі перетворюються на фактичне 3D-представлення себе, з деякими обмеженнями щодо того, наскільки точно фізична модель може відповідати віртуальній моделі [4].

1.4 Етапи створення та основні властивості тривимірної графіки

Створення тривимірної комп’ютерної графіки поділяється на три основні етапи:

- 3D-моделювання – процес формування комп’ютерної моделі форми об’єкта;
- макет та анімація – розміщення та переміщення об’єктів у межах сцени;
- 3D-рендеринг – комп’ютерні розрахунки, які на основі розміщення світла, типів поверхні та інших властивостей генерують зображення.

Модель описує процес формування форми предмета. Два найпоширеніші джерела тривимірних моделей – це ті, що художник чи інженер бере на комп'ютері за допомогою якогось інструменту тривимірного моделювання, і моделі, відскановані в комп’ютер із реальних об’єктів: полігональне моделювання, моделювання патчів та моделювання NURBS – деякі популярні інструменти, що використовуються при тривимірному моделюванні. Моделі також можна виготовляти процедурно або за допомогою фізичного моделювання. В основному 3D-модель формується з точок, що називаються вершинами, які визначають форму і утворюють багатокутники. Багатокутник – це площа, утворена щонайменше з трьох вершин (трикутників). Багатокутник з n точок є n-кутником. Загальна цілісність моделі та її придатність для використання в анімації залежать від структури багатокутників.

Матеріали та текстури – це властивості, які механізм візуалізації використовує для візуалізації моделі. Можна надати моделі властивості

матеріалів, щоб механізм візуалізації зміг згідно до наданих властивостей обробляти світло при попаданні на поверхню. Текстури використовуються для завдання кольору матеріалу за допомогою кольорової карти або карти альбедо, або надання поверхневих властивостей за допомогою рельєфного або нормального текстурування. Він також може бути використаний для деформації самої моделі за допомогою карти зміщення.

Перш ніж візуалізовати зображення, об'єкти слід розкласти на сцені. Це визначає просторові взаємозв'язки між об'єктами, включаючи розташування та розмір. Анімація стосується часового опису об'єкта, тобто того, як він рухається та деформується з часом. Популярні методи включають в себе створення кадрів, зворотну кінематику та захоплення руху. Ці прийоми часто застосовуються в поєднанні. Як і анімація, фізичне моделювання також визначає рух.

Рендеринг перетворює модель у зображення або імітує трасування променів для отримання фотореалістичних зображень, або застосовує художній стиль, як у нефореалістичній візуалізації. Дві основні операції в реалістичній візуалізації – це трасування, тобто скільки світла надходить з одного місця в інше і розсіювання, що визначає як різні поверхні взаємодіють зі світлом. Зазвичай цей крок виконується за допомогою програмного забезпечення для комп’ютерної графіки 3D або API 3D-графіки. Зміна сцени у відповідну форму для рендерингу також передбачає тривимірну проекцію, яка відображає тривимірне зображення у двох вимірах. Хоча програмне забезпечення для 3D-моделювання може також виконувати тривимірну візуалізацію, наприклад, Autodesk 3ds Max або Blender. Існує також ексклюзивне програмне забезпечення для 3D-рендерингу.

1.5 Особливості програм обробки тривимірних зображень

Програмне забезпечення для комп’ютерної графіки створює зображення, згенеровані комп’ютером за допомогою 3D-моделювання та 3D-

рендерингу та створює 3D-моделі для аналітичних, наукових та промислових цілей.

Існує безліч різновидів файлів, що підтримують 3D-графіку, наприклад, файли .obj Wavefront та файли .x DirectX з власною унікальною структурою даних.

Кожен формат файлу можна отримати через їх власні програми, такі як DirectX Files та Quake, або через незалежну програму, або навіть шляхом декомпіляції вручну.

3D-моделятори дозволяють користувачам створювати та змінювати моделі за допомогою їхньої 3D-сітки. Користувачі можуть додавати, віднімати, розтягувати та іншим чином змінювати сітку за своїм бажанням. Моделі можна розглядати з різних сторін, як правило, одночасно. Моделі можна обертати, а вигляд можна збільшувати та зменшувати.

3D-моделятори можуть експортувати свої моделі у файли, які потім можна імпортувати в інші програми, якщо метадані сумісні. Багато моделей дозволяють підключати імпортерів та експортерів, щоб вони могли читати та записувати дані у власних форматах інших програм.

Більшість 3D-моделей містять ряд супроводжуючих функцій, таких як індикатори променів та інші альтернативні варіанти візуалізації та засоби відображення текстур. Деякі також містять функції, які підтримують або дозволяють анімацію моделей. Деякі можуть створити повноформатне відео із серії відтворених сцен, тобто анімація.

Програмне забезпечення для автоматизованого проектування може використовувати тіж основні методи, що і програмне забезпечення для 3D-моделювання, але їх мета відрізняється. Вони використовуються в автоматизованій техніці, автоматизованому виробництві, аналізі кінцевих елементів, управлінні життєвим циклом продукції, 3D-друці та автоматизованому архітектурному дизайні.

Після створення відео студії редагують або складають відео за допомогою таких програм, як Adobe Premiere Pro або Final Cut Pro на

середньому рівні, або Autodesk Combustion, Digital Fusion, Shake у висококласному. Програмне забезпечення використовується у поєднанні зазвичай для узгодження відео в реальному часі з відео, створеним комп’ютером, підтримуючи синхронізацію двох, коли камера рухається.

Використання двигунів комп’ютерної графіки в реальному часі для створення кінематографічного виробництва називається машиніма.

1.6 Особливості авіаційних тренажерів

Авіаційний (пілотажний) тренажер – симулятор польоту, призначений для наземної підготовки пілотів. В авіаційному тренажері за допомогою апаратно-програмного комплексу імітується динаміка польоту і робота систем повітряного судна за допомогою спеціальних моделей, реалізованих в програмному забезпеченні обчислювального комплексу тренажера.

Підготовка пілотів на авіаційному тренажері – один з найважливіших елементів забезпечення безпечної експлуатації. Вона дозволяє мінімізувати негативний вплив людського фактора, тобто дозволяє звести до мінімуму можливість помилкових дій екіпажу транспортного засобу. Актуальність тренажерної підготовки має стійку тенденцію до зростання в зв'язку з тим, що людський фактор продовжує залишатися основною причиною авіаційних катастроф. Крім цього, бурхливе зростання обчислювальних потужностей ЕОМ дозволило довести сучасні авіаційні тренажери до такого рівня розвитку, що підготовка пілотів на тренажерах стала більш ефективною, ніж підготовка на реальному повітряному засобі. Така ефективність авіаційних тренажерів обумовлена їх можливостями до забезпечення високої інтенсивності підготовки. Так, якщо в реальному польоті екіпаж змушений приділяти значний час виконанню рутинних операцій, не пов'язаних з виконанням конкретних завдань навчання, наприклад, виконання тривалого «польоту по коробочці», набору висоти, польоту в зону і т. д., то на тренажері спеціальне програмне забезпечення дозволяє миттєво змінювати умови польоту, погоду,

географічне положення, зупиняти виконання завдання для розбору і повтору і т. д. Також на тренажері можна без обмежень виконувати відпрацювання дій в нештатних ситуаціях, деякі з яких або небезпечні для відпрацювання в реальному польоті, або взагалі їх відпрацювання в реальному польоті заборонено. Крім цього, підготовка пілотів на авіатренажёрах вигідна з економічної точки зору, не дивлячись на високу вартість сучасних тренажерів, що наближається до вартості самого повітряного засобу.

Незважаючи на те, що необхідність тренажерної підготовки загальновизнана, вона несе потенційну небезпеку, пов'язану з можливістю прищеплення хибних навичок через недостатню адекватність моделей засобів. Прикладом прищеплення помилкової навички на тренажері, що призвів до авіакатастрофи, є катастрофа лайнера А300 в Нью-Йорку. Як показало розслідування цієї катастрофи, пілот цієї авіакомпанії демонстрував на тренажері енергійну роботу педалями керма напряму, що призвело в реальному польоті при попаданні в зону турбулентності до розгойдування літака по нишпоренню з подальшим відділенням вертикального оперення від фюзеляжу. При цьому подібні дії на тренажері не приводили до виходу літака за межі експлуатаційних обмежень.

Авіаційні тренажери можна розділити на три основні групи:

- тактичні тренажери (англ. Full Mission Simulator);
- комплексні тренажери (англ. Full flight simulator);
- процедурні тренажери (англ. Flight Procedures Training Device).

У сучасній практиці підготовки пілотів цивільної авіації найбільшого поширення набули комплексні і процедурні тренажери.

Процедурні тренажери призначені для відпрацювання екіпажем процедур підготовки та виконання польоту.

У тренажерах такого призначення пульти, прилади і органи управління зазвичай імітуються за допомогою сенсорних моніторів. Для зручності окремі пульти і органи управління можуть бути представлені у вигляді повнорозмірних макетів. Зазвичай це імітатори бічних ручок управління

повітряного засобу, імітатори пульта управління автопілотом, імітатори лицьових панелей системи літаководіння.

Процедурні тренажери не призначені для придбання навичок пілотування. Тому вони зазвичай не обладнуються системою візуалізації.

Під комплексними тренажерами розуміють авіаційні тренажери, що забезпечують підготовку екіпажів в повному обсязі їх функціональних обов'язків з льотної експлуатації повітряного судна конкретного типу.

Комплексні тренажери – це тренажери найвищого рівня. Як правило, вони мають систему руху. Кабіна комплексного тренажера виконується у вигляді повної репліки реальної кабіни повітряного судна. На комплексні тренажери встановлюються передові системи візуалізації.

Якщо тренажери цивільних літаків практично досягли стелі свого розвитку для сучасного рівня елементної бази, то тактичні тренажери продовжують мати практично необмежені можливості для свого вдосконалення. Тактичні тренажери призначені для відпрацювання групових бойових дій. Вони об'єднані в єдину мережу за допомогою інтерфейс, який дозволяє поєднувати різнорідні тренажери – авіаційні, танкові, артилерійські та ін.

1.7 Система візуалізації авіаційних тренажерів

Сучасні системи візуалізації бувають двох типів – проекційні і колімаційні. У системах візуалізації обох типів зображення проектується за допомогою проекторів на сферичних або циліндричних екранах. Проектування зображення на екранах, розташованих в безпосередній близькості від кабіни тренажера, призводить до того, що лінія візування віддалених об'єктів залежить від положення очей пілотів. Кут цієї помилки – паралакс, що можна оцінити за допомогою формули 1.1:

$$\alpha = \text{arctg}(D/L), \tag{1.1}$$

де D – відстань від голови пілота до центру настройки системи візуалізації;

L – відстань від центру настройки системи візуалізації до екрану, що зображено на рисунку 1.4.

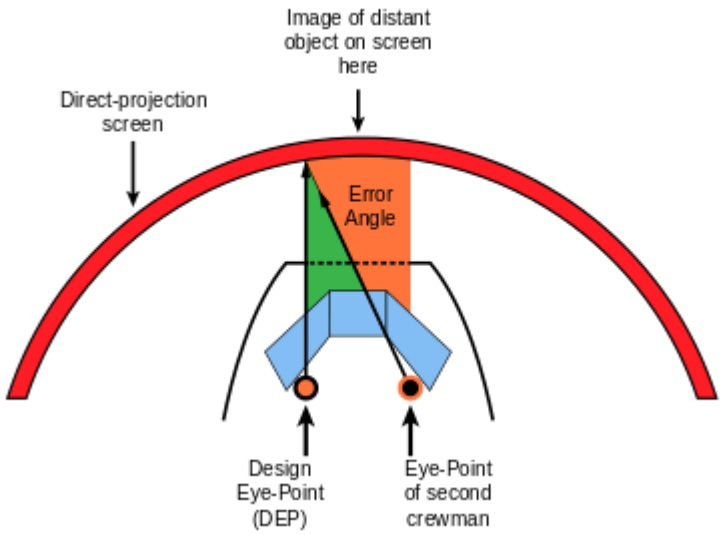

Рисунок 1.4 – Кут помилки лінії візування проекційної системи

Наявність паралакса – недолік властивий саме проекційним системам візуалізації. В кабіні тренажера з проекційною системою візуалізації існує тільки одна точка, в якій паралакс дорівнює нулю. При проектуванні системи візуалізації за цю точку приймають місце пілота. Так як в екіпажі може бути як лівий, так і правий пілот, то в цьому випадку в системі візуалізації передбачають дві точки нульової помилки з можливістю перемикання з одного місця на інше.

Причиною паралакса є близько розташований екран, а також властивість світла розсіюватися при відображенні від негладкої поверхні екрану. Якщо йде від проекторів світло колімованим, тобто проектувати таким чином, щоб промені світла об'єкта були паралельні один одному, то явище паралакса буде усунуто. На цьому принципі заснована робота колімаційної системи візуалізації. У колімаційній системі світло від проекторів пропускають через

спеціальну оптичну систему – екран зворотної проекції на сферичне дзеркало. Таким чином створюється ілюзія об'єктів віддалених на велику відстань, що зображено на рисунку 1.5.

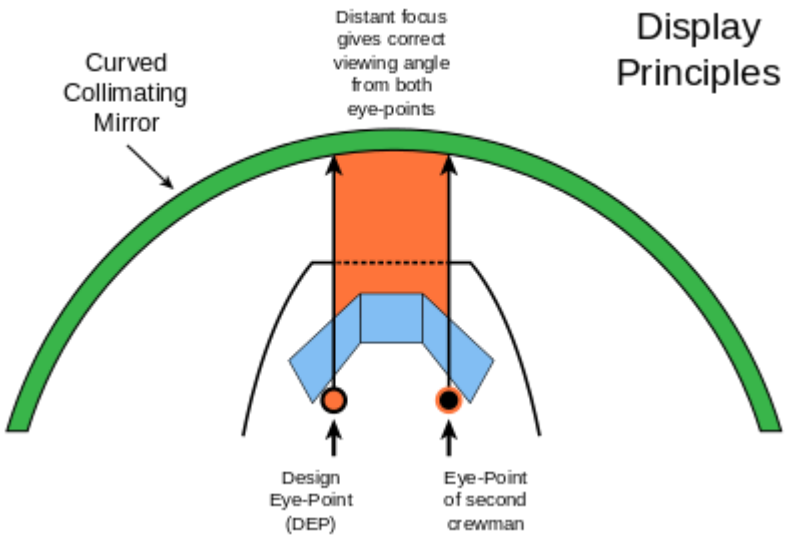

Рисунок 1.5 – Колімаційна система візуалізації

Вартість колімаційної системи візуалізації перевищує один мільйон долларів, але тільки вона дозволяє відпрацьовувати на тренажері навички візуальної посадки. Колімаційна системи встановлюються на комплексні тренажери FFS і тренажери FTD Level 2 (Level 2 по JAR-FSTD).

Важливим елементом системи візуалізації є відеопроектори. В сучасних тренажерах застосовуються DLP-проектори. У комплексних тренажерах більш досконалі LCOS-проектори або DLP-проектори на світлодіодах.

2 СИНТЕЗ ЗОБРАЖЕНЬ

2.1 Поняття візуалізації

Візуалізація або синтез зображень – це процес створення фотореалістичного або нефореалістичного зображення з 2D або 3D моделі за допомогою комп'ютерної програми. Отримане зображення називається візуалізацією. Кілька моделей можна визначити у файлі сцени, що містить об’єкти суворо визначеною мовою або структурою даних. Файл сцени містить геометрію, точку зору, текстуру, освітлення та інформацію про затінення, що описують віртуальну сцену. Дані, що містяться у файлі сцени, потім передаються програмі візуалізації для обробки і виводяться у файл цифрового зображенн. Термін візуалізація використовується для опису процесу обчислення ефектів у програмі відеомонтажу для отримання кінцевого відеовиходу.

Візуалізація є однією з основних підтем комп'ютерної тривимірної графіки і на практиці вона завжди пов'язана з іншими. Це останній великий крок у графічному конвеєрі, який надає моделям та анімації остаточного вигляду. З ростом витонченості комп’ютерної графіки з 1970-х років вона стає все більш виразною.

Візуалізація використовується в архітектурі, відеоіграх, симуляторах, візуальних ефектах фільмів та телевізорів та візуалізації дизайну, кожен із яких використовує різний баланс функцій та технік. Доступний широкий вибір різноманітних візуалізаторів. Деякі з них інтегровані у великі пакети моделювання та анімації, інші є самостійними, а деякі – безкоштовними проектами з відкритим кодом. З внутрішньої сторони візуалізатор – це ретельно розроблена програма, заснована на багатьох дисциплінах, включаючи фізику світла, зорове сприйняття, математику та розробку програмного забезпечення [5].

Хоча технічні деталі методів візуалізації різняться, загальні проблеми, які потрібно подолати при створенні двовимірного зображення на екрані із тривимірного, що зберігається у файлі сцени, обробляються графічним конвеєром у пристрої візуалізації, такому як графічний процесор. GPU – це спеціально розроблений пристрій, який допомагає центральному процесору у виконанні складних обчислень візуалізації. Якщо сцена виглядає відносно реалістичною та передбачуваною при віртуальному освітленні, програмне забезпечення для рендерингу має вирішити рівняння. Рівняння рендерингу не враховує всіх явищ освітлення, але натомість діє як загальна модель освітлення для комп'ютерних зображень.

У випадку 3D-графіки сцени можуть бути попередньо відтворені або створені в режимі реального часу. Попередній рендеринг – це повільний обчислювальний процес, який зазвичай використовується для створення фільмів, де сцени можуть створюватися заздалегідь, тоді як рендеринг у реальному часі часто робиться для 3D-ігор та інших програм, які повинні динамічно створювати сцени. 3D апаратні прискорювачі можуть покращити ефективність візуалізації в режимі реального часу.

Коли попереднє зображення, як правило ескіз каркасу, завершується, використовується візуалізація, яка додає в растрові текстури або процедурні текстури, індикатори, відображення нерівностей та відносне розташування до інших об’єктів. Результат – завершене зображення, яке бачить споживач або передбачуваний глядач.

Для анімації фільмів потрібно відтворити кілька зображень (кадрів) і зшити їх у програмі, здатній зробити анімацію такого роду. Більшість програм для редагування 3D-зображень можуть це зробити.

Багато алгоритмів візуалізації досліджено і програмне забезпечення, що використовується для рендерингу, може використовувати низку різних методів для отримання кінцевого зображення.

Візуалізація є завершальним процесом створення власне 2D-зображення або анімації з підготовленої сцени. Це можна порівняти з фотозйомкою або

зйомкою сцени після завершення налаштування в реальному житті. Було розроблено кілька різних і часто спеціалізованих методів візуалізації. Вони варіюються від відверто нереалістичного каркасного рендерингу через рендеринг на основі багатокутників до більш прогресивних методів, таких як: рендеринг лінії сканування та трасування променів. Візуалізація може зайняти від частки секунди до днів для одного зображення або кадру. Загалом, різні методи краще підходять або для фотореалістичного візуалізації, або для рендерингу в реальному часі.

Візуалізація для інтерактивних медіа, таких як ігри та симуляції, обчислюється та відображається в реальному часі зі швидкістю приблизно від 20 до 120 кадрів в секунду. При отриманні в реальному часі мета полягає в тому, щоб показати якомога більше інформації, яку око може обробити за частку секунди. У разі анімації 30 кадрів в секунду кадр охоплює одну тридцяту секунди.

Основною метою є досягнення якомога більш високого ступеня фотореалізму при допустимій мінімальній швидкості візуалізації. Зазвичай це 24 кадри в секунду, оскільки це мінімум, який має бачити людське око, щоб успішно створити ілюзію руху.

Програмне забезпечення для рендерингу може імітувати такі візуальні ефекти, як відблиски об'єктива, глибина різкості або розмиття в русі. Це спроби імітувати зорові явища, що виникають внаслідок оптичних характеристик камер та людського ока. Ці ефекти можуть надати елемент реалізму сцені, навіть якщо ефект є лише імітованим артефактом камери.

2.2 Основні властивості візуалізованого зображення

Візуалізоване зображення може бути описане низкою видимих ознак. Дослідження та розробка в основному мотивуються пошуком шляхів для їх ефективного моделювання. Деякі пов’язані безпосередньо з певними алгоритмами та техніками, тоді як інші створюються разом.

Властивості візуалізованого тривимірного об'єкту:

- затінення – як колір і яскравість поверхні змінюються залежно від освітлення;
- текстура – метод нанесення деталей на поверхні;
- рельєфне текстурування – метод імітації дрібних нерівностей на поверхнях;
- ефект туману – як світло тьмяніє, проходячи через неясну атмосферу або повітря;
- тіні – ефект перешкоджання світлу;
- м'які тіні – мінлива темрява, спричинена частково затемненими джерелами світла;
- відблиск – дзеркальні або глянцеві відблиски;
- прозорість (оптика), прозорість (графічна) або непрозорість – різка передача світла через тверді предмети;
- прозорість – сильно розсіяне пропускання світла через тверді предмети;
- заломлення – вигин світла, пов'язаний з прозорістю;
- дифракція – вигин, розповсюдження та перешкоди світла, що проходить повз об'єкт або отвір, що призводить до змізення променю;
- непряме освітлення – поверхні, освітлені променями, відбитими від інших поверхонь, а не безпосередньо від джерела світла (також відомого як загальне освітлення);
- каустика (форма непрямого освітлення) – відбиття світла від блискучого об'єкта або фокусування світла через прозорий об'єкт для отримання яскравих відблисків на іншому об'єкті;
- глибина різкості – предмети здаються розмитими або поза фокусом, якщо знаходяться занадто далеко перед або поза фокусом;
- розмиття в русі – предмети здаються розмитими внаслідок швидкісного руху або руху камери;
- нефореалістичний візуалізація – візуалізація сцен у художньому стилі.

2.3 Візуалізація в реальному часі

Швидке збільшення обчислювальної потужності комп’ютера дозволило поступово підвищити ступінь реалізму навіть для рендерингу в режимі реального часу, включаючи такі методи, як рендеринг HDR. Візуалізація в режимі реального часу часто є полігональною і допомагає GPU комп'ютера.

Анімація для неінтерактивних засобів масової інформації, таких як художні фільми та відео, може зайняти набагато більше часу для рендерингу. Візуалізація в режимі реального часу дозволяє використовувати обмежену обробну потужність для отримання високої якості зображення. Час рендерінгу для окремих кадрів може змінюватись від декількох секунд до декількох днів для складних сцен. Візуалізовані кадри зберігаються на жорсткому диску, а потім передаються на інші носії. Потім ці кадри відображаються послідовно з високою частотою кадрів, як правило, 24, 25 або 30 кадрів в секунду для досягнення ілюзії руху.

Процес візуалізації є обчислювально дорогим, враховуючи складне різноманіття фізичних процесів, що моделюються. Потужність комп’ютерної обробки швидко зростала за ці роки, що дозволило поступово підвищити ступінь реалістичної візуалізації. Кіностудії, які виробляють комп’ютерну анімацію, зазвичай використовують ферму або сервер рендерингу для своєчасного створення зображень. Однак падіння апаратних витрат означає, що цілком можливо створити невелику кількість 3D-анімації в системі домашнього комп'ютера. Результат рендерингу часто використовується як лише одна невелика частина завершеної сцени кінофільму. Багато шарів матеріалу можуть бути зроблені окремо та інтегровані в остаточний знімок за допомогою програмного забезпечення для композитування.

Відстеження кожної частинки світла в сцені майже завжди абсолютно непрактично і забирає колосальний час. Навіть відстеження частини, достатньо великої для створення зображення, займає надзвичайно багато часу, якщо вибірка не розумно обмежена.

Тому існують більш ефективні методи моделювання:

- растеризація, включаючи візуалізацію лінії сканування, геометрично проектує об’єкти сцени на площину зображення, без вдосконалених оптичних ефектів;

- кидання променів або ray casting розглядає сцену як спостережену з конкретної точки зору, обчислюючи спостережуване зображення, базуючись лише на геометрії та основних оптичних законах інтенсивності відбиття, і, можливо, використовуючи методи Монте-Карло для зменшення артефактів;

- трасування променів схоже на кидання променів, але воно використовує більш вдосконалене оптичне моделювання і зазвичай використовує методи Монте-Карло для отримання більш реалістичних результатів зі швидкістю, яка часто на порядок швидша.

Найсучасніше програмне забезпечення поєднує два або більше методів для отримання достатньо хороших результатів за розумну ціну.

Інша відмінність полягає в алгоритмах впорядкування зображень, які здійснюють ітерацію над пікселями площини зображення, та алгоритмах впорядкування об’єктів, які здійснюють ітерацію над об’єктами на сцені. Як правило, порядок об'єктів є більш ефективним, оскільки в сцені зазвичай менше об'єктів, ніж пікселів.

2.4 Растеризація

Растеризація – це процес, в якому описану у векторному графічному форматі фігуру перетворюють в його растрове зображення: серію пікселів, крапок або ліній, які при спільному відображенні створюють зображення, яке було представлене через фігури. Потім растризоване зображення може відображатися на дисплеї комп'ютера, відеодисплеї або принтері або зберігатися у форматі растрових файлів. Растеризація може стосуватися техніки малювання тривимірних моделей або перетворення 2D-примітивів візуалізації, таких як багатокутники, сегменти рядків, у растровий формат.

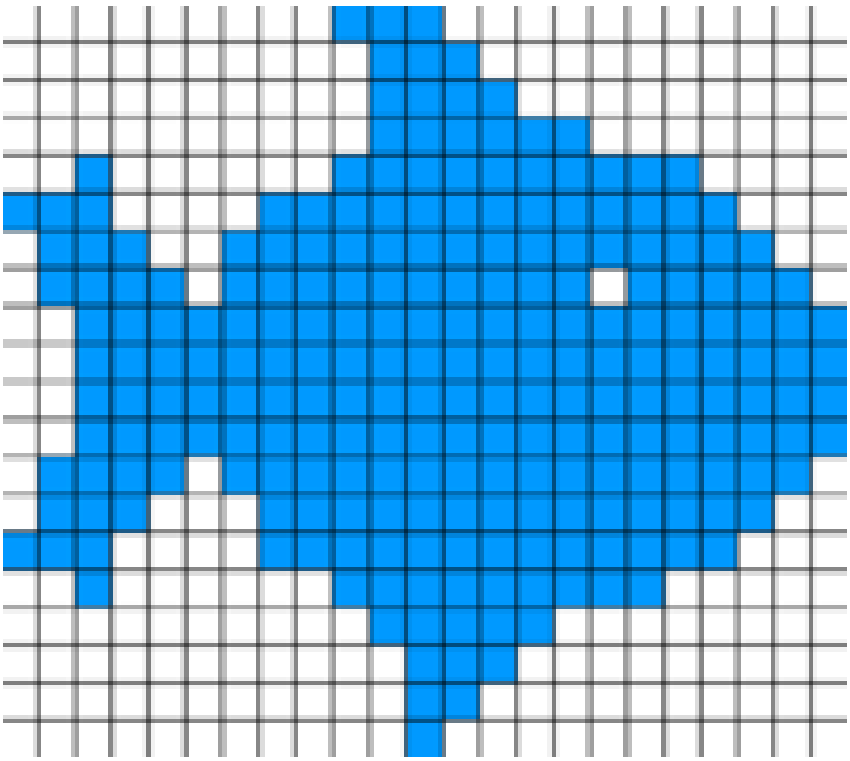

Рисунок 2.1 – Приклад растрового графічного зображення

Растеризація – одна із типових технік візуалізації 3D-моделей. Порівняно з іншими методами візуалізації, такими як трасування променів, растеризація надзвичайно швидка і тому використовується в більшості 3D-движків реального часу. Однак растеризація – це просто процес обчислення відображення від геометрії сцени до пікселів і не передбачає певного способу обчислення кольору цих пікселів. Конкретний колір кожного пікселя призначається затіненням, що в сучасних графічних процесорах повністю програмоване. Затінення може базуватися на фізичних законах, їх наближеннях або суто художньому задумі.

Процес растеризації 3D-моделей на 2D-площині для відображення на екрані комп'ютера часто здійснюється за допомогою функціонального (непрограмованого) обладнання в графічному конвеєрі. Це пояснюється тим, що не існує мотивації для модифікації методів растеризації, що використовуються під час рендерингу, а спеціальна система забезпечує високу ефективність.

Растеризація трикутниками поширене представлення цифрових 3D-моделей. Перед растеризацією окремі багатокутники розбиваються на трикутники, тому типовою проблемою для вирішення при 3D-растеризації є растеризація трикутника. Властивості, які зазвичай вимагаються від алгоритмів растеризації трикутників, полягають у тому, щоб растеризувати два сусідні трикутники, тобто ті, що мають спільний край:

- не залишає отворів (нерастеризованих пікселів) між трикутниками, так що растеризована область повністю заповнена (як і поверхня сусідніх трикутників);
- жоден піксель не раструється більше одного разу, тобто растризовані трикутники не перекриваються. Це гарантує, що результат не залежить від порядку, в якому трикутники растеризовані. Перевитрата пікселів може також означати витрачання обчислювальної потужності на пікселі, які будуть перезаписані.

Це призводить до встановлення правил растеризації, щоб гарантувати вищезазначені умови. Один набір таких правил називається правилом у верхньому лівому куті, яке говорить, що піксель растеризується тоді і тільки тоді, коли:

- його центр лежить повністю всередині трикутника;
- його центр лежить точно на краю трикутника (або декількох ребрах у випадку кутів), що є (або декілька країв у випадку кутів) або верхнім, або лівим краєм.

Верхнє ребро – це ребро, яке є точно горизонтальним і лежить над іншими краями, а ліве ребро – це негоризонтальне ребро, яке знаходиться з лівої сторони трикутника.

Це правило реалізується в Direct3D та багатьох специфікаціях OpenGL, хоча специфікація цього не визначає і вимагає лише послідовного правила.

Якість растеризації можна поліпшити за допомогою згладжування, яке створює гладкі краї. Субпіксельна точність – це метод, який враховує позиції в більш тонкому масштабі, ніж піксельна сітка, і може дати різні результати,

навіть якщо кінцеві точки примітиву потрапляють в однакові піксельні координати, створюючи більш плавні анімаційні рухи. Простому або старішому обладнанню, такому як PlayStation 1, не вистачало точності субпікселів у 3D-растеризації.

2.5 Кидання променів

Кидання променів є основним з багатьох алгоритмів візуалізації комп’ютерної графіки, що використовують геометричний алгоритм трасування променів. Алгоритми рендеринга на основі трасування променів працюють для того, щоб візуалізувати тривимірні сцени у двовимірні зображення. Геометричні промені простежуються від ока спостерігача для відбору світла, що рухається до спостерігача з напрямку променя. Швидкість і простота кидання променів походить від обчислення кольору світла без рекурсивного відстеження додаткових променів, які відбирають проміння, що падає на точку, в яку потрапив промінь. Це виключає можливість точного надання відображень, заломлень або природного падіння тіней, однак усі ці елементи можна підробити до певної міри шляхом творчого використання текстурних карт чи інших методів. Висока швидкість обчислення робить кидання проміння зручним методом візуалізації у ранніх 3D-іграх у реальному часі.

Ідея кидання променів полягає в тому, щоб простежити промені від ока, по одному на піксель, і знайти найближчий об’єкт, що перешкоджає шляху променя. Використовуючи властивості матеріалу та ефект світла у сцені, цей алгоритм може визначити затінення цього об’єкта. Якщо поверхня звернена до світла, світло буде досягати цієї поверхні і не буде заблокованим або в тіні. Затінення поверхні обчислюється за допомогою традиційних моделей затінення комп'ютерної графіки 3D. Однією з важливих переваг кидання променів, запропонованих у порівнянні зі старими алгоритмами сканлайн-ліній, була здатність легко оброблювати тверді тіла, такі як конуси та сфери.

Якщо математичну поверхню можна перетинати променем – її можна відтворити за допомогою проміння. Опрацьовані об'єкти можуть бути створені за допомогою моделювання твердих тіл та легко відтворені [6].

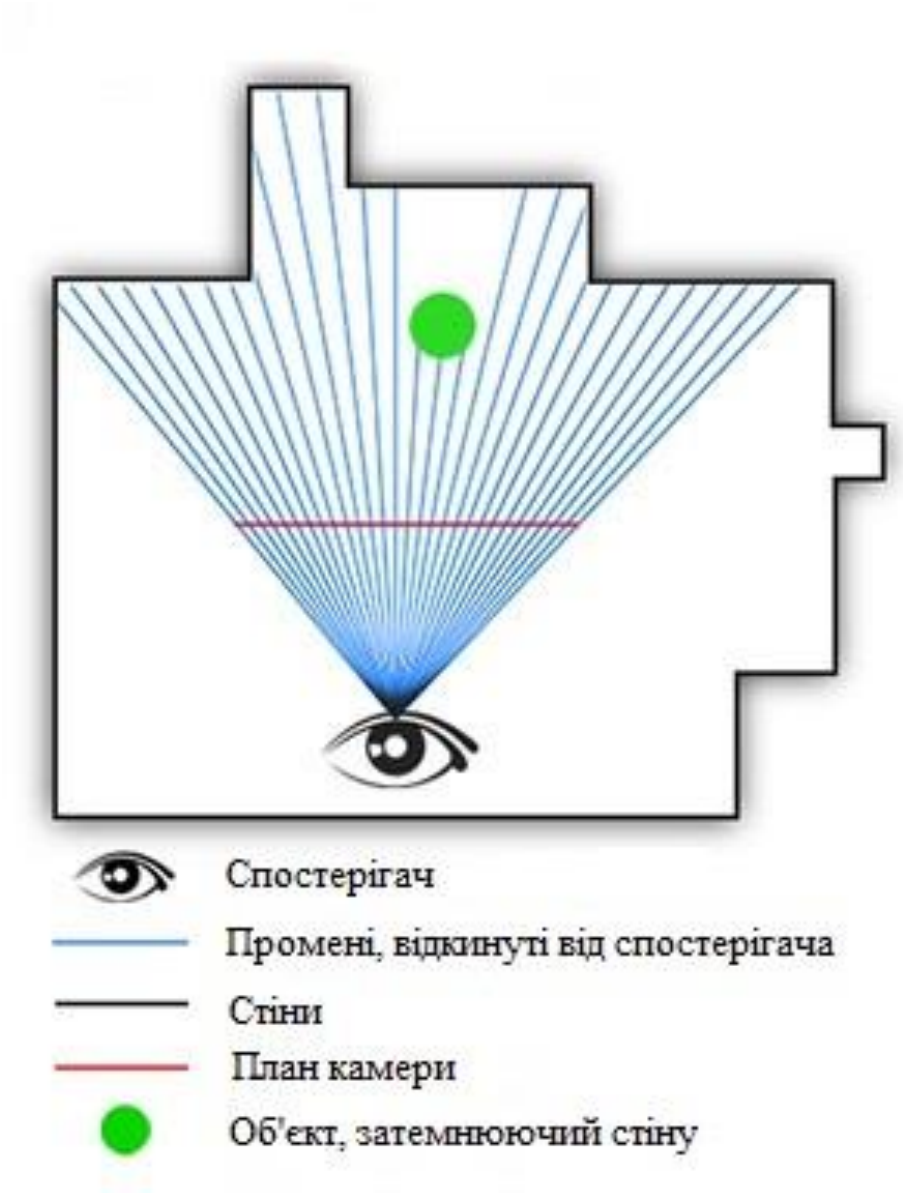

Рисунок 2.2 – Метод кидання променів

2.6 Трасування променів

Трасування променів – це техніка візуалізації для створення зображення шляхом відстеження шляху світла як пікселів у площині зображення та моделювання ефектів його зустрічей з віртуальними об'єктами. Ця техніка здатна надати високий ступінь візуального реалізму, набагато більший, ніж типові методи рендеринга ліній сканування, але за більших обчислювальних витрат. Це робить трасування променів найкращим чином придатним для додатків, де можна терпіти відносно тривалий час для рендерингу, наприклад, у нерухомих комп'ютерних зображеннях та візуальних ефектах у фільмах та телебаченні.

Трасування променів здатне імітувати різноманітні оптичні ефекти, такі як відбиття та заломлення, розсіювання та дисперсійні явища ,наприклад, хроматичні аберації.

Трасування шляху – це форма трасування променів, яке може створювати м’які тіні, глибину різкості, розмиття руху, їдкість, оклюзію навколишнього середовища та непряме освітлення. Трасування шляху – це неупереджений метод візуалізації, але для отримання високоякісних еталонних зображень без шумних артефактів потрібно відстежувати велику кількість променів.

Оптичне трасування променів описує метод для створення візуальних зображень, побудованих у середовищах комп’ютерної графіки 3D, з більшою фотореалістичністю, ніж метод кидання променів. Метод працює шляхом трасування шляху від уявного ока через кожен піксель на віртуальному екрані та обчислення кольору видимого через нього об’єкта.

Сцени з трасуванням променів математично описуються програмістом або візуальним художником, як правило, за допомогою посередницьких інструментів. Сцени також можуть включати дані із зображень та моделей, знятих такими засобами, як цифрова фотографія.

Як правило, кожен промінь повинен бути перевірений на перетин з деякою підмножиною всіх об'єктів сцени. Після визначення найближчого об’єкта алгоритм оцінить чи надходить світло у точці перетину, з'ясує властивості матеріалу об’єкта та об’єднає цю інформацію для обчислення кінцевого кольору пікселя. Деякі алгоритми освітлення та світловідбиваючі або напівпрозорі матеріали можуть вимагати потрапляння більшої кількості променів на сцену [7].

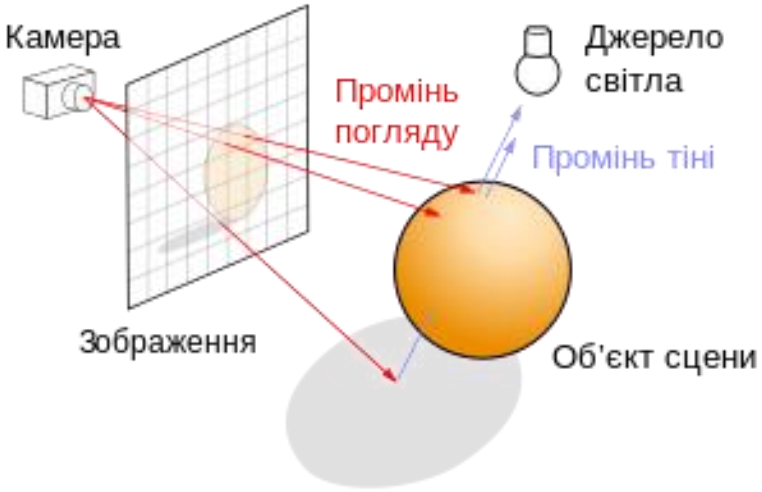

Рисунок 2.3 – Схема роботи алгоритму трасування променів

На рисунку 2.3, спочатку, може здатися неінтуїтивним направляти промені подалі від камери, а не в неї, як це робить справжнє світло в дійсності, але це набагато ефективніше. Оскільки переважна більшість світлових променів від даного джерела світла не потрапляє безпосередньо в око глядача, симуляція «вперед» може потенційно витратити величезну кількість обчислень на світлових шляхах, які ніколи не реєструються.

2.7 Трасування шляху

Трасування шляху – методика візуалізації в комп'ютерній графіці, яка прагне симулювати фізичне поведінки світла настільки близько до реального,

наскільки це можливо. Трасування шляху є окремим випадком традиційного трасування променів. Якість зображень, одержуваних за допомогою методу трасування шляху, як правило, краще, ніж якість зображень, отриманих іншими методами рендерингу, проте трасування шляху вимагає набагато більших витрат обчислювальних ресурсів.

Трасування шляху є найбільш простим, найбільш точним з фізичної сторони і найбільш повільним по продуктивності методом візуалізації. Трасування шляху природним способом відтворює безліч оптичних ефектів, які важко досяжні або взагалі недосяжні іншими методиками рендерингу:

- побудова тіней;
- глибина різкості;
- розмитість;
- каустика;
- непряме освітлення.

Здійснення цих оптичних ефектів за допомогою трасування шляху набагато простіше, ніж за допомогою інших методів.

Виходячи з точності і відсутності апроксимацій і припущень, трасування шляху використовується для генерації зображень, які потім використовуються як порівняльні зразки для оцінки якості рендерингу інших алгоритмів. Для того, щоб одержати високу якість зображень, згенерованих за допомогою трасування шляху, потрібно провести трасування дуже великої кількості променів, в іншому випадку з'являться графічні артефакти у вигляді шуму.

У реальному світі безліч невеликих порцій світла випромінюються джерелами світла і поширюються по прямих лініях у вигляді променів крізь середу і від об'єкта до об'єкта, змінюючи свій колір і інтенсивність. Ця подорож триває до тих пір, поки промені не будуть поглинені об'єктами, включаючи такі об'єкти, як людське око або камеру. Цей процес поширення променів симулюється трасуванням шляху, за винятком того, що промені трасуються навпаки, від віртуальної камери, тобто спостерігача, до джерела світла. Це зроблено через те, що з тих променів, які виходять з джерела світла,

лише дуже мала частина потрапляє на об'єктив віртуальної камери, тому розрахунок переважної більшості променів ніяк не впливає на одержуване віртуальною камерою зображення.

Трасування шляху не є простим трасуванням променів з необмеженою кількістю віддзеркалень променів. У традиційному трасуванні променів світло обчислюється в момент безпосереднього перетину променя з дифузійною поверхнею. При трасуванні шляху новий промінь генерується випадковим чином всередині півсфери об'єкта і потім трасується до тих пір, поки не перетнеться з джерелом світла, що може і не статися. При трасуванні шляху промінь може перетнутися з безліччю дифузних поверхонь до того, як перетнеться з джерелом світла.

Псевдокод, який реалізує трасування шляху показано в прикладі 1.1. Якщо кожна поверхня закритого простору випромінила і відбила (0.5,0.5,0.5), то кожен піксель в зображенні матиме білий колір.

```
      Color TracePath(Ray r,depth) {
          if(depth == MaxDepth)
            return Black;  // bounced enough times

          r.FindNearestObject();
          if(r.hitSomething == false)
            return Black;  // nothing was hit

          Material m = r.thingHit->material;
          Color emittance = m.emittance;

          // pick a random direction from here and keep going
          Ray newRay;
          newRay.origin = r.pointWhereObjWasHit;
          newRay.direction                                                    =
RandomUnitVectorInHemisphereOf(r.normalWhereObjWasHit);
          float cos_t = DotProduct(newRay.direction, r.normalWhereObjWasHit);

          Color BRDF = m.reflectance/PI;
          float scale = 1.0*PI;
          Color reflected = TracePath(newRay,depth+1);

          // return emittance + ( BRDF * scale * cos_t * reflected );
          return emittance + ( BRDF * scale * reflected );
        }
```

Приклад 2.1 – Реалізація трасування шляху

Вибирати інтеграл для точки можна за допомогою двох незалежних методів:

- обстріл променями з джерел світла і створення шляху в сцені. Шлях переривається випадковим числом «кроків-відскоків» променя. Потім світло прямує на проектований піксель кінцевого зображення. Під час цього методу візуалізації мільйони шляхів створюються і на зображенні запам'ятовуються результати прорахунку шляхів;

- збір променів з точки на поверхні. Промінь вистрілюється крізь пікселі зображення і скаче по сцені, поки не зустріне на своєму шляху джерело світла. Світло з джерела надсилається у напрямку пікселів зображення. Процес створення шляху називається семплінги. Одна точка поверхні зазвичай отримує від восьмисот семплів до трьох тисяч. Кінцеве зображення переводиться за допомогою арифметичних дій.

Двонаправлене трасування променів комбінує обстріл і збір в одному алгоритмі і це дає більшу збіжність зображенню, тобто менше шуму. Ці два методи генерації шляхів трасуються незалежно і потім початок обстріляного шляху з'єднується з хвостом шляху збору променів. Враховується згасання світла при кожному відскоку променя і запам'ятовується в пікселях зображення. Ця техніка здається на перший погляд парадоксально повільною. На практиці ж – навпаки, додаткова швидкість збіжності зображення компенсує уповільнення, що виникають по необхідності випускати нові і нові промені.

Для того, щоб прискорити конвергенцію або збіжність зображень, двонаправлені алгоритми трасують шлях в обох напрямках. У прямому напрямку промені трасуються від джерела світла до тих пір, поки вони не стануть настільки слабкими, що їх не можна буде побачити, чи поки не потраплять на об'єктив віртуальної камери. У зворотному промені трасуються від віртуальної камери до тих пір, поки вони не зіткнуться з джерелом світла, або поки кількість їх відображень не перевищить певну межу. Такий підхід зазвичай приводить до такого зображення, яке сходиться набагато швидше.

Трасування шляху постійно проводить вибірку пікселів зображення. Зображення стає розрізняти тільки тоді, коли проведено кілька вибірок на один піксель, аж до ста вибірок на піксель. Як правило, для звичайних зображень і для зменшення цифрового шуму до прийнятного рівня роблять близько п’яти тисяч вибірок. Однак для патологічних випадків кількість вибірок стає набагато більше. Процес візуалізації може зайняти години і дні в залежності від складності сцени і продуктивності апаратного і програмного забезпечення. Сучасні реалізації графічних процесорів обіцяють від одного до десяти мільйонів вибірок в секунду, що робить можливим згенерувати безшумне зображення прийнятної якості протягом декількох секунд або хвилин. Цифровий шум створює особливу проблему для анімації, створюючи, як правило, небажаний ефект зернистості зображення.-

Досить важко справедливо оцінити рівень продуктивності рендерера. Один підхід полягає в підрахунку вибірок (семплів) за секунду, інший підраховує кількість шляхів, які можуть бути відтрасовувані і додані до зображення за секунду. Результати цих методів значно варіюються в залежності від сцени і залежать від глибини шляху, тобто від того, скільки разів променю дозволяється відбитися від об'єкта, перш ніж він буде зупинений. Результат вимірювання продуктивності також сильно залежить від використовуваного апаратного забезпечення. Один рендерер може виробляти багато низькоякісних вибірок, тоді як інший може вивести фінальне зображення швидше, використовуючи меншу кількість більш високоякісних вибірок.

3 МЕТОДИ ТА РЕАЛІЗАЦІЯ ХМАРНОГО ШАРУ

3.1 Алгоритм побудови моделі хмари

Існуючі методи створення моделей хмар можна розділити на дві групи. До першої групи належать методи, засновані на моделюванні фізичних процесів. Ці методи потребують завдання великого числа вхідних даних, а також значних витрат обчислювальних ресурсів. Другу групу складають методи обчислювального моделювання форми хмар. Такі методи використовують, коли необхідно отримати зображення довільного хмарного об'єкту або схожого на оригінал, заданий будь-яким способом. Вони, як правило, мають більш прості алгоритми обчислення, проте також дозволяють формувати реалістичне тривимірне зображення хмарного шару. Пропонований метод відноситься до другої групи. В результаті виконання алгоритму виходить тривимірна модель хмари, задана набором метасфер. Для візуалізації такої моделі зручно використовувати метод зворотного трасування [7].

Вхідними даними в цій моделі є параметри еліпсоїда, що обмежує хмару для формування – значення розмірів піввісь a, b, c і координати його центру x, y, z. У процесі формування моделі хмари вихідний еліпсоїд розбивається на п'ять подібних йому фігур, розміри головних піввісь яких визначаються за формулою:

$$a_i = k_i a,\ b_i = k_i b,\ c_i = k_i c, \tag{3.1}$$

де $i \in \{1,2,...,5\}$ – номер фігури, k – випадкова величина в діапазоні від 0 до 1.

Координати центрів отриманих еліпсоїдів визначається наступним чином:

$$
\begin{aligned}
x_1 &= x + a(1\text{-}k_i),\ y_1 = y,\ z_1 = z,\\
x_2 &= x - a(1\text{-}k_i),\ y_2 = y,\ z_2 = z,\\
x_3 &= x,\ y_3 = y,\ z_3 = z + c(1\text{-}k_i),\\
x_4 &= x,\ y_4 = y,\ z_4 = z - c(1\text{-}k_i),\\
x_5 &= x,\ y_5 = y + b(1\text{-}k_i),\ z_5 = z.
\end{aligned}
\tag{3.2}
$$

На наступному кроці алгоритму кожен з отриманих еліпсоїдів також розбивається на п'ять фігур відповідно до формул 3.1 і 3.2. Цей процес триває до тих пір, поки не буде досягнутий заданий рівень деталізації. На останньому кроці кожен еліпсоїд розбивається на п'ять метасфер. Координати центрів знаходяться за допомогою формули 3.2, враховуючи, що $R = a = b = c$, а радіуси визначаються за формулою:

$$R_1 = k_1 a,\ R_2 = k_2 a,\ R_3 = k_3 c,\ R_4 = k_4 c,\ R_5 = k_5 b. \tag{3.3}$$

Значення коефіцієнта пропускання в точці перетину променя спостереження з метасферою визначається співвідношенням:

$$T_i = t_i\, f(d_i, R_i), \tag{3.4}$$

де i – номер метасфери, t – коефіцієнт пропускання в центрі метасфери, f – функція розподілу щільності, R – радіус метасфери, d – відстань від проекційного променя до центру метасфери.

Вираз для t_i можна записати в наступному вигляді:

$$t_i = \exp(-\alpha R_i), \tag{3.5}$$

де α – коефіцієнт, що характеризує прозорість всередині хмарного шару.

В якості f пропонується використовувати одну з наступних функцій:

$$f(d,R)=\begin{cases}cos^2\left(\frac{d}{R}\right), & \text{d} \le \text{R};\\ 0, & \text{d} > \text{R},\end{cases} \tag{3.6}$$

або

$$f(d,R)=\begin{cases}\left(1-\frac{d}{R}\right)cos\left(\frac{d}{R}\right), & \text{d} \le \text{R};\\ 0, & \text{d} > \text{R}.\end{cases} \tag{3.7}$$

Така модель дозволяє легко здійснити анімацію хмарних утворень. Анімація хмар, в загальному випадку, здійснюється шляхом зміни параметрів моделі хмари. У даній моделі змінюються радіуси метасфер і координати їх центрів.

Для анімації формується послідовність з декількох реалізацій моделей хмар, отриманих описаним вище методом. Оскільки номера метасфер в одній реалізації моделі відповідають номерам в інших реалізаціях, то легко отримати відповідність між двома реалізаціями. Потім здійснюється лінійна інтерполяція параметрів між двома моделями. Оскільки побудова моделі хмари даним методом не вимагає значних витрат обчислювальних ресурсів, то можливо виконання процесу анімації в реальному масштабі часу на універсальній ЕОМ [8].

На рисунку 3.1 наведено реалізацію хмарного утвореня, отриманого при використанні випадкових функцій з різним розподілом щільності ймовірності.

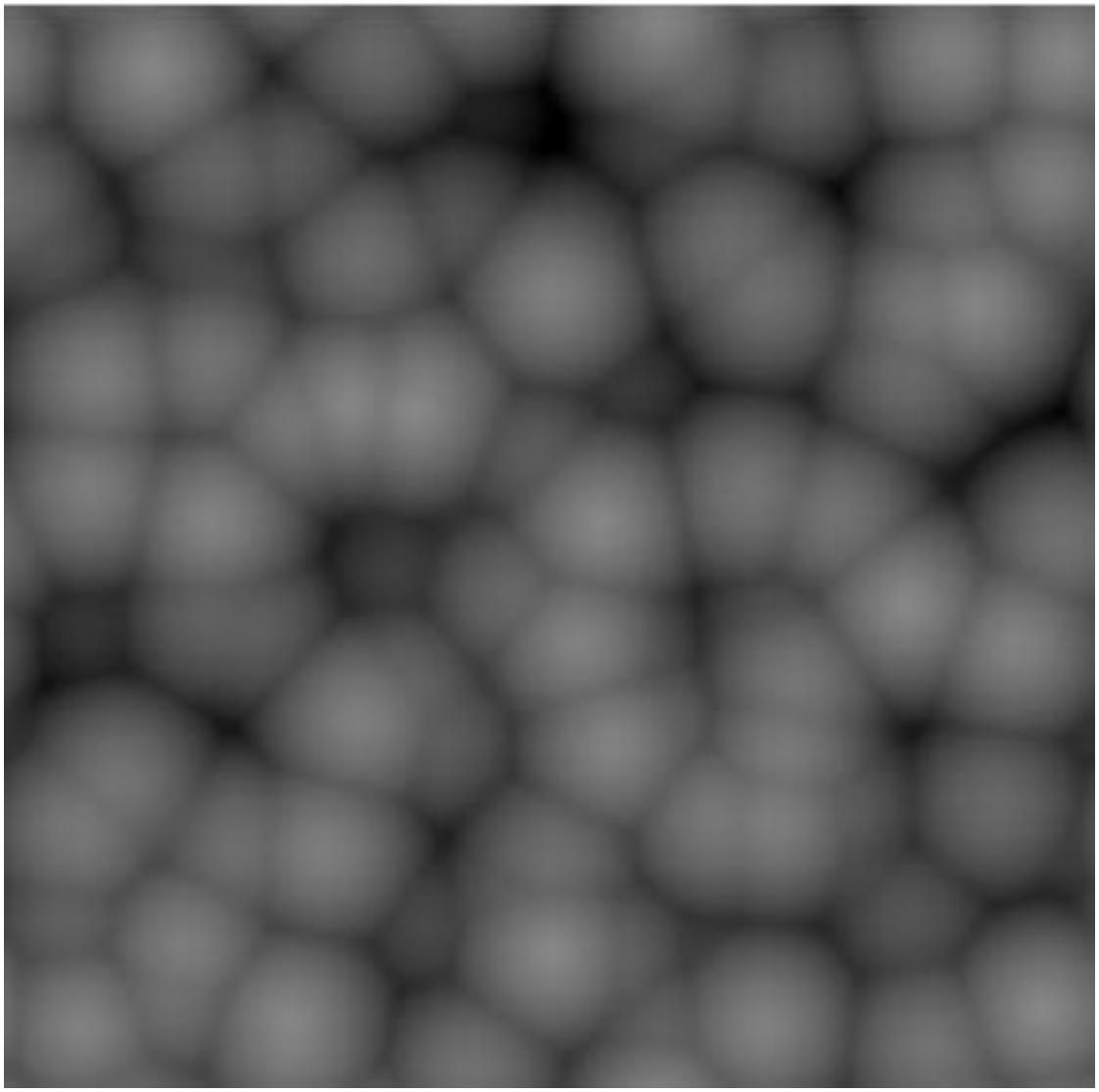

Рисунок 3.1 – Хмарне утвореня з невеликою кількістю метасфер

При збільшенні на порядок кількості метасфери в реалізації моделі, можна досягти заданного значення якості зображення. На рисунку 3.2 наведено реалізацію утвореня з більшою кількістю метасфер, що підвищує реалістичність такої реалізації, але збільшується кількість необхідних обчислень, що за допомогою методу трасування променів ніяк не позначиться на швидкості роботи програмного засобу.

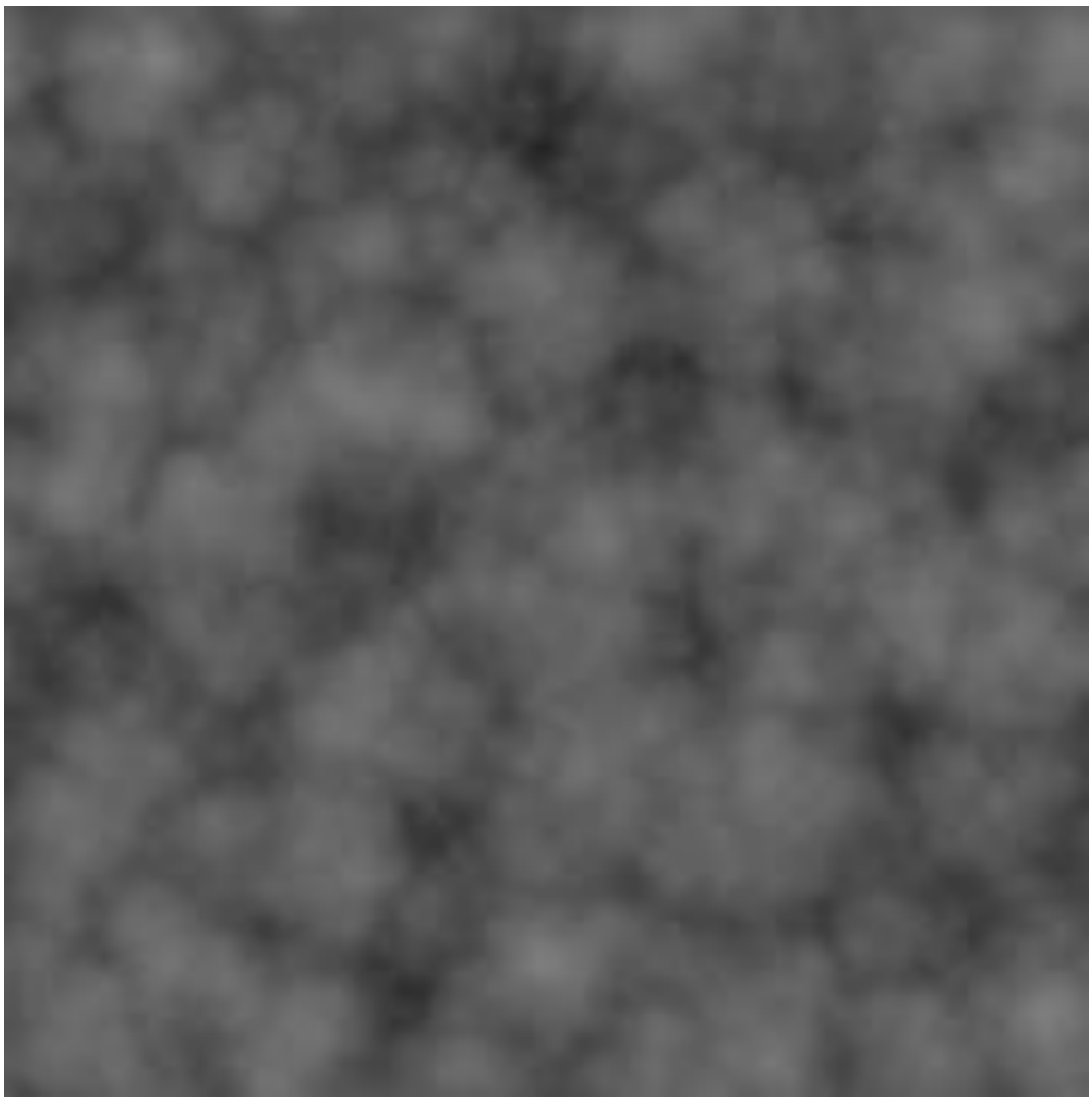

Рисунок 3.2 – Більш реалістичне хмарне утворення

Запропонований метод призначений для побудови моделей окремих купчастих хмар. Змінюючи закон розподілу випадкових чисел в формулах 3.1 і 3.2, а також вибираючи відповідну функцію розподілу щільності, можна впливати на форму одержуваних хмар. Хмарний шар великої протяжності можна отримати комбінацією окремих хмар. Завдяки тому, що метасфери в одній реалізації моделі відповідають метасферам з такими ж номерами в інший реалізації хмар, можна легко здійснити анімацію хмарного шару, як показано на рисунку 3.3

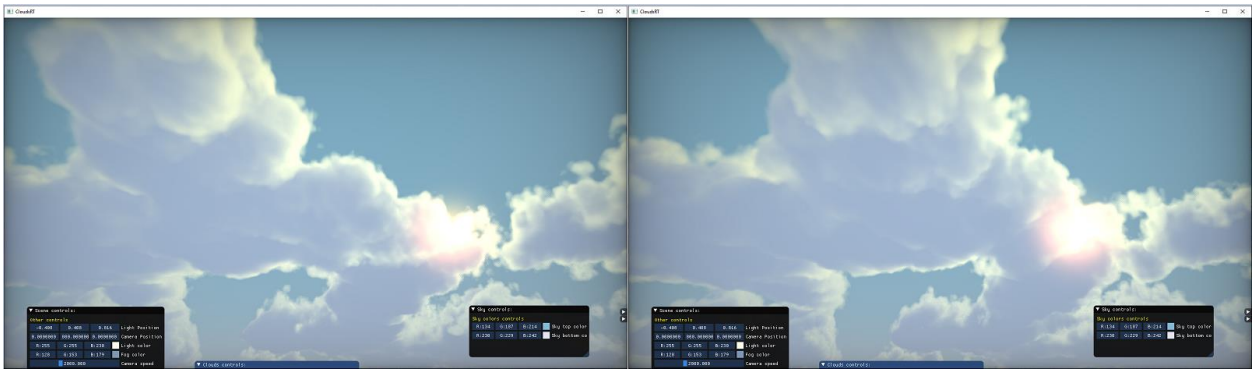

Рисунок 3.3 – Переміщення масиву хмарних утворень

3.2 Обчислення кольору пікселя зображення

Обчислення кольору пікселя здійснюється у відповідності з наступним алгоритмом:

- визначення номерів метасфер, що перетинаються променем спостереження і їх сортування в порядку зростання відстані до центру проекції;

- обчислення інтенсивності відбитого світла в кожній точці перетину проекційного променя і метасфери:

$$I'_j = I_{sun} [0{,}125 (5 + \cos \gamma_j) + 0{,}25 \cos \beta_j], \quad (3.8)$$

де $j \in \{1,...,N\}$ – номер метасфери;

γ – кут між нормаллю до поверхні метасфери в точці перетину з проекційним променем і віссю Y в базовій системі координат;

β – кут між напрямком на Сонце і нормаллю до поверхні метасфери в точці перетину з проекційним променем;

- обчислення поглинання світла атмосферою для кожної точки перетину:

$$I_j = I'_j \exp(-\alpha l), \quad (3.9)$$

де l – відстань від центру проекції до точки перетину проекційного променя з метасфер;

- обчислення кольору з урахуванням всіх точок перетину проекційного променя з метасферами:

$$I = \left(I_b T_N + I_N(1\text{-}T_N)\right)T_N\text{-}1 + I_N - \text{-}1(1\text{-}T_N\text{-}1)) \ \ldots\ )T_1 + I_1(1 - T_1), \quad (3.10)$$

де I_b – колір фону;

T_i – значення коефіцієнта пропускання метасфери, отримане з формули 3.4.

На цьому етапі алгоритм обчислення кольору пікселя завершується [9].

В прикладі 3.1 показано функцію обчислення кінцевого кольора пікселя.

```
      vec4 calculateFinalColor(vec3 cameraPos, vec3 cameraRayDir, float
AAIndex)
      {
          //init
          vec3 finalColor = vec3(0.0);
          float absorbMul = 1.0;
          vec3 rayStartPos = cameraPos;
          vec3 rayDir = cameraRayDir;
          float firstHitRayLength = -1.0;
          for(int i = 0; i < MAX_BOUNCE; i++)
          {
              HitData h = AllObjectsRayTest(rayStartPos + rayDir *
0.0001,rayDir);//+0.0001 to prevent ray already hit @ start pos

                      firstHitRayLength = firstHitRayLength < 0.0 ? h.rayLength
: firstHitRayLength;
              if(h.rayLength > 99999.0)
              {
                  vec3 skyColor = vec3(0.7,0.85,1.0);//hit nothing = hit sky
color
                  finalColor = skyColor * absorbMul;
              }
            absorbMul *= 0.8;
            rayStartPos = rayStartPos + rayDir * h.rayLength;
            rayDir = normalize(reflect(rayDir,h.normal) +
randomInsideUnitSphere(rayDir,rayStartPos,AAIndex) * rougness);
          }
          return vec4(finalColor,firstHitRayLength);//alpha nly for CineShader,
to show depth
      }
```

Приклад 3.1 – Обчислення кінцевого кольору пікселя

3.3 Математична модель метеоумов для систем візуалізації

При проектуванні СВ тренажерів розробники прагнуть до формування зображення, найбільш відповідного реальній картині, що спостерігається з кабіни транспортного засобу. Можливість обробки в СВ атмосферної димки, туману, хмар підвищує фізичну реалістичність синтезованого зображення і розширює перелік вправ для відпрацювання навичок керування ТЗ в екстремальних ситуаціях.

З деяким наближенням можна вважати, що ослаблення світла атмосферою викликається тільки його розсіюванням, що створює так звану димку. Повітряна димка має яскравість, так як зважені в повітрі частки пилу, кристали льоду, краплі, водяна пара і самі молекули повітря розсіюють світловий потік, що падає на них. Наявність димки в атмосфері призводить до зниження контрасту між спостережуваним об'єктом і фоном. Цьому сприяють два фактори. Перший фактор – це зменшення істинної яскравості об'єкта внаслідок ослаблення світла шаром атмосфери. Другий – це позірне перетворення кольорового об'єкта в ахроматичний через зменшення насиченості кольору внаслідок накладання яскравості атмосферної димки на яскравість об'єкта [10].

Таким чином зміна видимості об'єкта днем може бути проімітована в СВ шляхом зміни насиченості і яскравості кольору об'єкта в залежності від відстані між об'єктом і наглядачем.

Для обчислення уявної яскравості об'єкта, що знаходиться від спостерігача на відстані l, скористаємося співвідношенням:

$$L = L_0\tau(l) + L_\infty[1 - \tau(l)], \qquad (3.11)$$

де L_0 – початкова яскравість кольору об'єкта;

L_∞ – коефіцієнт, що характеризує яскравість димки при $l \to \infty$, τ – коефіцієнт пропускання атмосфери.

У наведеному рівнянні 3.11 перший доданок відповідно до закону Бугера показує ослаблення яскравості кольору об'єкта, що знаходиться на відстані l від спостерігача. Другий доданок є світло-повітряним рівнянням Кошмидера, і описує зміну яскравості димки в залежності від товщини атмосферного шару. Зменшення насиченості кольору внаслідок накладення яскравості атмосферної димки на яскравість об'єкта охарактеризуємо наступним співвідношенням:

$$S = S_0\tau(l), \tag{3.12}$$

де S_0 – початкова насиченість кольору об'єкту.

У разі оптично однорідного шару атмосфери товщиною, що дорівнює одиниці довжини, τ визначається як питома прозорість атмосфери τ_1. Знаючи питому прозорість і товщину однорідного шару атмосфери, можна визначити коефіцієнт пропускання:

$$\tau = \tau_1^{\,l}, \tag{3.13}$$

Зазвичай $\tau_1 = e^{-\alpha}$ або $\tau_1 = 10^{-\delta}$, де показники α і δ характеризують ступінь розсіювання (ослаблення) світла атмосферою. Для спрощення подальших обчислень вводиться показникова двійкова функція:

$$\tau = 2^{-\beta} \tag{3.14}$$

$$\beta = 1/l^0 \tag{3.15}$$

де l^0 – товщина однорідного атмосферного шару заданої прозорості, на котрій коефіцієнт пропускання зменшується в два рази.

З формули 3.14 та 3.15 маємо:

$$l^0 = -l/\log_2 \tau_1. \tag{3.16}$$

Для однакової оцінки прозорості атмосфери прийнята міжнародна шкала видимості, в якій питома прозорість атмосфери τ_1 пов'язана з МДВ. Таким чином, запроваджений параметр l^0 може бути однозначно прив'язаний до МДВ, що дозволяє задавати і обробляти метеоумови в СВ відповідно до прийнятої практики.

У разі неоднорідності шарів атмосфери коефіцієнт пропускання визначимо наступним чином:

$$\tau = 2^{-\sum_{i=1}^{n} \frac{l_i}{l_i^0}}, \tag{3.17}$$

де n – число ділянок (шарів) різної прозорості на шляху проходження проекційного променя в атмосфері;

l_i – довжина i-ї ділянки шляху проекційного променя в атмосфері з заданим однорідним шаром прозорості;

l_i^0 – параметр прозорості i-го шару атмосфери.

Скористаємося моделлю кольорів TSL (T – тон кольору, L - яскравість, S - насиченість):

$$\begin{gathered} R = L - \frac{S(\cos T + \sqrt{3}\sin T)}{3}, \\ B = L - S(\cos T - \sqrt{3}\sin T)/3, \\ B = L - S(\cos T - \sqrt{3}\sin T)/3. \end{gathered} \tag{3.18}$$

Підставимо вирази для L і S з формул 3.11 і 3.12:

$$\begin{gathered} R = L_0\tau + L_\infty(1-\tau) - \frac{S_0\tau(\cos T_0 + \sqrt{3}\sin T_0)}{3}, \\ G = L_0\tau + L_\infty(1-\tau) + 2S_0\tau\cos T_0\,/3, \\ B = L_0\tau + L_\infty(1-\tau) - S_0\tau(\cos T_0 - \sqrt{3}\sin T_0)/3. \end{gathered} \tag{3.19}$$

Виконаємо заміни відповідно до рівняння 3.18:

$$\begin{aligned} R &= R_0\tau + L_\infty(1-\tau), \\ G &= G_0\tau + L_\infty(1-\tau), \\ B &= B_0\tau + L_\infty(1-\tau). \end{aligned} \tag{3.20}$$

Так як $0 \le \tau \le 1$, спростимо це співвідношення:

$$\begin{aligned} R &= R_0\tau + L_\infty\overline{\tau}, \\ G &= G_0\tau + L_\infty\overline{\tau}, \\ B &= B_0\tau + L_\infty\overline{\tau}, \end{aligned} \tag{3.21}$$

де $\overline{\tau}$ – інверсія від τ.

Для знаходження довжини і-ї ділянки шляху проекційного променя в атмосфері застосуємо математичну модель геометричних перетворень зображення:

$$l_i = \frac{VH_i}{V_y}, \tag{3.22}$$

де V – модуль вектора спостереження;

V_y – проекція вектора V на вісь Y;

H_i – проекція l_i на вісь Y.

Розділимо в формулі 3.22 ліву і праву частину на l_i^0 і підставимо в формулу 3.17:

$$\tau = 2^{-\frac{V}{V_y}\sum_{i=1}^{n}\frac{H_i}{l_i^0}}. \tag{3.23}$$

Таким чином, співвідношення 3.21 і 3.23 дозволяють обчислити для кожного пікселя екрану з урахуванням прозорості ділянок атмосфери складові кольору RGB об'єкта, на який потрапив проекційний промінь [11].

3.4 Алгоритм обробки метеоумов

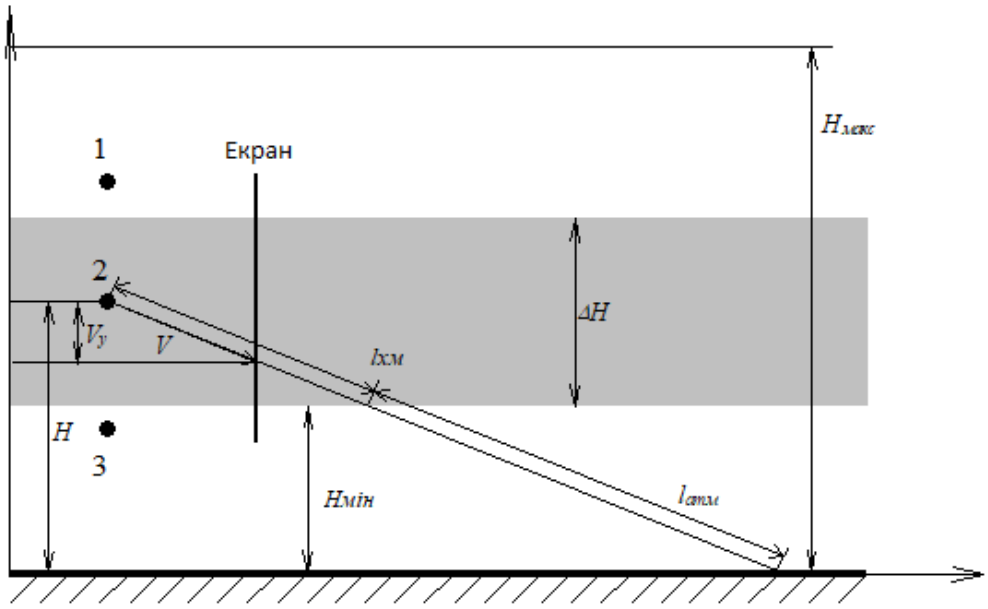

Рисунок 3.4 – Розміщення точки спостереження

Розглянемо приклад, представлений на рисунку 3.4. Точки 1,2,3 визначають три можливі випадки положення центру проекції на висоті H по співвідношенню до поверхні Землі. На малюнку прийняті наступні позначення: V – відстань від точки спостереження до екрану монітора, V_y – проекція вектора V на вісь Y, H – висота точки спостереження, $H_{мін}$ – висота шару хмар, ΔH – товщина шару хмар, $H_{макс}$ – граничне значення висоти.

Для спрощення обчислень скористаємося логарифмічною системою числення. Позначимо показник в формулі 3.23 через ρ. Тоді маємо:

$$\log_2 \rho = \log_2 (V \sum_{i=1}^{n} \frac{H_i}{l_i^0}) - \log_2 V_y. \qquad (3.24)$$

У правій частині співвідношення 3.24 зменшуване можна вважати незмінним протягом кадру і тому необхідно обчислювати його один раз на кожен кадр. Від'ємник в правій частині рівняння обчислюється для кожного пікселя екрану

Відповідно до формули 3.24 розпишемо три випадки, представлених на рисунку, з урахуванням можливого положення проекційних променів.

1. $H \geq H_{хм} + \Delta H$, точка 1:

- нижче лінії горизонту:

$$\log_2 \frac{l}{l_0} = \log_2 \left[\left(\frac{H - \Delta H}{l_{атм_0}} + \frac{\Delta H}{l_{хм_0}} \right) V \right] - \log_2 V_y; \quad (3.25)$$

- вище лінії:

$$\log_2 \frac{l}{l_0} = \log_2 \left(\frac{H_{макс} - H}{l_{атм_0}} V \right) - \log_2 V_y. \quad (3.26)$$

2. $H_{хм} < H < H_{хм} + \Delta H$, точка 2:

- нижче лінії горизонту:

$$\log_2 \frac{l}{l_0} = \log_2 \left[\left(\frac{H_{мін}}{l_{атм_0}} + \frac{H - H_{мін}}{l_{обл_0}} \right) V \right] - \log_2 V_y. \quad (3.27)$$

- вище лінії:

$$\log_2 \frac{l}{l_0} = \log_2 \left[\left(\frac{H_{макс} - \Delta H - H_{мин}}{l_{атм_0}} + \frac{H_{мин} + \Delta H - H}{l_{хм_0}} \right) V \right] - \log_2 V_y. \quad (3.28)$$

3. $H < H_{хм}$, точка 3:

- нижче лінії горизонту:

$$\log_2 \frac{l}{l_0} = \log_2 \frac{HV}{l_{атм_0}} - \log_2 V_y; \quad (3.29)$$

- вище лінії:

$$\log_2 \frac{l}{l_0} = \log_2 \left[\left(\frac{H_{макс} - H - \Delta H}{l_{атм_0}} + \frac{\Delta H}{l_{хм_0}} \right) V \right] - \log_2 V_y. \quad (3.30)$$

Перелічимо основні етапи обробки.

1. Для кожного кадру зображення обчислюється пара зменшуваних в співвідношеннях в залежності від випадків 1, 2, 3, які заносяться в підсистему обробки метеоумов, де зберігаються протягом обробки кадру зображення. У підсистему заноситься також параметр L_∞.

2. На кожному такті роботи конвеєра в підсистему обробки метеоумов з підсистеми графічної обробки надходить $\log_2 V_y$.

3. Обчислюється $\log_2 \rho$ в відповідності до виразів 3.25-3.30.

4. Табличним методом обчислюється коефіцієнт пропускання τ.

5. Одночасно обчислюються складові кольору. Множення в формулі 3.21 виконуються табличним методом [12].

3.5 Паралельність обчислень

Використання графічного процесора для моделювання звільняє центральний процесор для інших обчислень та дозволяє здійснювати загально швидше моделювання. Обчислення на графічних процесорах нещодавно стали активною областю досліджень в галузі комп'ютерної графіки.

Центральні процесори призначені для обчислень із низькою затримкою, тоді як графічні процесори оптимізовані для високої пропускної здатності фрагментів. Низька затримка в додатках із великою пам’яттю, як правило,

вимагає більшого об’єму кеш пам’яті, які використовують велику кількість кремнію. Додаткові транзистори використовуються з більшим ефектом в архітектурах графічних процесорів, оскільки вони застосовуються до додаткових процесорів і функціональних блоків, що збільшують пропускну здатність. Інтерактивні програми для комп’ютерної графіки мають багато компонентів, які змагаються за час обробки. Часто важко ефективно виконувати одночасно моделювання, рендеринг та інші обчислювальні завдання без падіння продуктивності. Оскільки на меті візуальне моделювання, візуалізація є важливою частиною будь-якого рішення. Таким чином, методи динамічного моделювання на графічному процесорі забезпечують додатковий інструмент для балансування навантаження в складних інтерактивних додатках .

Існують різні API для взаємодії з графічним процесором. Деякі з найбільш відомих – DirectX, OpenGL та Vulkan. В роботі використовується OpenGL, щоб отримати доступ до графічного процесора. OpenGL також пропонує власну мову шейдерів – GLSL. Він використовується для написання програм шейдерів, які є програмованими етапами в конвеєрі візуалізації. GLSL – це мова, схожа на C, яка надає доступ до функцій, явно створених для роботи з графікою. Програми шейдерів компілюються на графічному процесорі з використанням драйверів, наданих вибраним API [14].

Щоб візуалізувати реалістичні хмари, потрібно були би відслідковувати кожен фотон, що випромінюється сонцем, простежити їх траєкторію, коли вони розсіюються через хмари, і зареєструвати ті кілька фотонів, які потрапляють в камеру. Це, звичайно, вимагає дуже великих обчислювальних ресурсів, і не потрібно відстежувати всі фотони, які ніколи не потрапляють в поле зору глядача. Тому алгоритм візуалізації хмар побудований на техніці трасування променів.

```
vec4 path_ray(vec3 origin, vec3 dir) {
vec4 value = vec4(0.0);
float delta = 1.0;
float start = gl_DepthRange.near;
```

```
float end = 500.0;
for (float t = start; t < end; t += delta) {
      sample_point = origin + dir * t;
      if (value.a == 1.0) {
            break;
      }

      value.a += cloud_sampling(sample_point, delta);
      value.a = clamp(value.a, 0.0, 1.0);
      ...
      value.rgb += ... * alpha;
}
}
return value;
}
```

Приклад 3.2 – використання GLSL

У прикладі 3.2 показано сильно стиснуту версія трасування променів, реалізована у GLSL. path_ray () відбиває промінь від початку у напрямку dir. Тут довжина кроку називається дельта і встановлюється в 1 одиницю. Верхня межа для проміжного маршу встановлена на 500 одиниць. Розміщується другий марш променів для розрахунку світлових ефектів [15].

3.6 Параметри візуалізації програмного додатку

Рисунок 3.5 – Контроль візуалізації

Реалізований додаток дозволяє змінювати та моделювати основні параметри хмарного шару в реальному часі. На рисунку 3.5 показано вікно програмного додатку, що відповідає за контроль затінення та візуалізації.

Основними параметрами є:

- Render – використовується для зупинки або запуску візуалізації;
- Exposure – встановлює експозицію зображення;
- Ray Depth – глибина відбиття променя для об'єктів на сцені;
- Volume Depth – глибина об'ємного променя, яка визначає, чи дозволено променю відбиватися в об'ємі кілька разів. На рисунку 3.6 показано різницю між об'ємною глибиною променів 1 і 50;

Рисунок 3.6 – різниця об’ємної глибини та глибини відбиття

- Phase g1 – це параметр анізотропії. Він приймає значення від -1 до 1 і визначає ймовірність напрямку, який може взяти промінь, що входить в об’єм;
- Sun Color – змінює колір сонячного світла;
- Light position – використовується для визначення положення сонця на небі, що також впливає на зміну кольору сонця та хмар, як показано на рисунку 3.7;

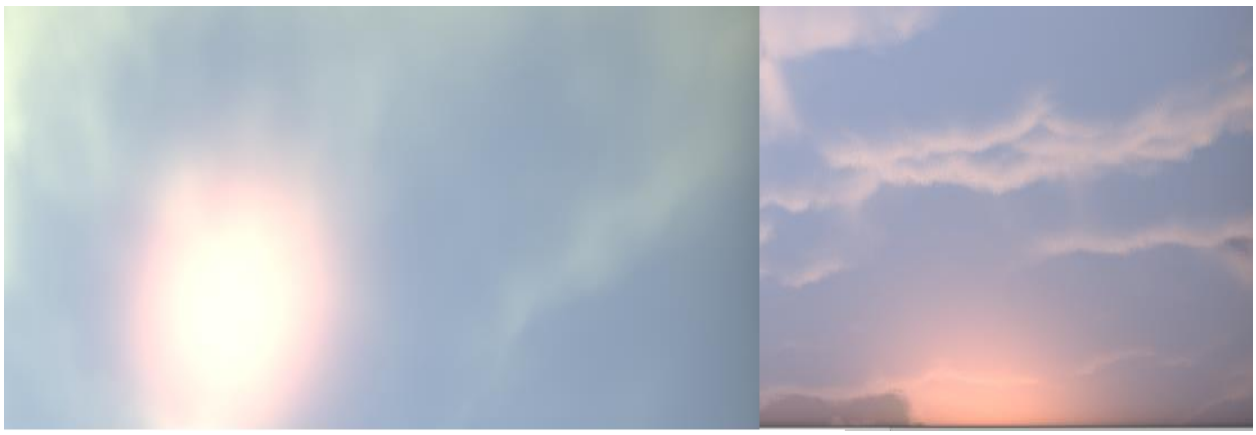

Рисунок 3.7 – Зміна положення сонця в небі

- Camera position – встановлює положення камери в залежності від заданих координат, також є можливість змінювати положення за допомогою комп'ютерної миші.

- Coverage – змінює об’єм хмарного шару за рахунок змін основних параметрів метасфер. На рисунку 3.8 наведено однакову хмару з різним розміром.

Рисунок 3.8 – Зміна параметру Coverage

- Speed – зміна параметру дозволяє прискорити анімації, тим самим змоделювати швидкість переміщення хмар;

- Light absorption – встановлює параметр поглинання світла для хмар;

- Clouds bottom height / clouds top height – схожі на параметр Coverage, але змінюється тільки розмір хмар по висоті, тобто вісь с.

ВИСНОВКИ

В процесі виконання роботи був розроблений програмний додаток, що дозволяє моделювати хмарний шар для систем візуалізації. Для візуалізації використовується метод зворотного трасування променів, що дозволяє підвищити реалістичність такої реалізації, але він також потребує великої кількості обчислень. Для отримання реалістичного зображення потрібно було б розробити велику кількість готових моделей та текстур хмар, що потребує затрат пам’яті СВ та позбавляє можливості їх ефективної анімації, тому для підвищення швидкості роботи розроблений та реалізований алгоритм побудови моделі хмар, що дозволяє виконувати моделювання переміщення хмар, тобто анімацію, без використання великої кількості окремих готових моделей. Також були представлені алгоритми обробки метеоумов, що підвищують ефективність роботи методу трасування променів для такої реалізації, а саме визначення коректного кольору для багатьох об’єктів та їх обробку в процесі трасування в залежності від розміщення спостерігача, тобто кабіни пілота, в просторі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hughes, F. John. Computer Graphics (principles and practice) / Andries van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley // by Addison–Wesley Publishing Company, Inc. 2014, – 1209p.

2. Introduction to Computer Graphics [Електронний ресурс] – Режим доступу: www/ URL: https://www.geeksforgeeks.org/introduction-to-computer-graphics/ – Загол. з екрану.

3. Иванов В.П., Батраков А.С. Трехмерная компьютерная графика // Под ред. Г.М. Полищука. – М. : Радио и связь, 1995. – 224 с.: ил.

4. 3D computer graphics [Електронний ресурс] – Режим доступу: www/ URL: https://en.wikipedia.org/wiki/3D_computer_graphics – Загол. з екрану.

5. Rendering: definition, types and visualization techniques [Електронний ресурс] – Режим доступу: www/ https://biblus.accasoftware.com/en/rendering-definition-types-and-visualization-techniques/ – Загол. з екрану.

6. What’s the Difference Between Ray Tracing, Ray Casting, and Ray Charles? [Електронний ресурс] – Режим доступу: www/ URL: https://www.electronicdesign.com/technologies/displays/article/21801219/whats-the-difference-between-ray-tracing-ray-casting-and-ray-charles – Загол. з екрану.

7. Real Time Ray Tracing [Електронний ресурс] – Режим доступу: www/ URL: https://alain.xyz/blog/real-time-ray-tracing – Загол. з екрану.

8. Гусятин В.М. Итерационный алгоритм синтеза изображения в растровой графике реального масштаба времени. // Харьков, ХТУРЭ, Радиоэлектроника и информатика. -1998. -№3. -С.81-83.

9. Гусятин, В.М. Метод уменьшения итераций в алгоритмах синтеза изображений реального масштаба времени / В.М Гусятин // Радиоэлектроника и информатика. – 2001.– №1. – С.99-100.

10. Гусятин В.М. Алгоритм геометрических преобразований изображения в системах визуализации тренажеров транспортных средств //

Авиационно-космическая техника и технология. Труды ХАИ им. Н.Е. Жуковского за 1997, с.467-471.

11. Гусятин В.М., Остроушко А.П. Математическая модель и алгоритм обработки метеоусловий для систем визуализации // Автоматизированные системы управления и приборы автоматики: Сб. научн. трудов. Выпуск 111. – Харьков: ХТУРЭ, 1999. – СХХ-ХХ.

12. Остроушко А.П., Гусятин В.М. Синтез изображений отдельных облачных образований // Радиоэлектроника и информатика. – 2000. – № 2. – С.79-81.

13. Five steps to adding ray tracing to an OpenGL ES-based deferred lighting system [Електронний ресурс] – Режим доступу: www/ URL: https://www.imaginationtech.com/blog/five-steps-to-adding-ray-tracing-to-an-opengl-es-based-deferred-lighting-system/ – Загол. з екрану.

14. Efficient Algorithms for Real-Time GPU Volumetric Cloud Rendering with Enhanced Geometry [Електронний ресурс] – Режим доступу: www/ URL: https://www.mdpi.com/2073-8994/10/4/125/htm – Загол. з екрану.

15. Nimbus SDK: A Tiny Framework for C++ Real-Time Volumetric Cloud Rendering, Animation and Morphing [Електронний ресурс] – Режим доступу: www/ URL: https://www.codeproject.com/Articles/5269907/Nimbus-SDK-A-Tiny-Framework-for-Cplusplus-Real-Tim – Загол. з екрану.