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 рельєфу в системах візуалізації для авіаційних тренажерів

(тема)

Виконав:
студент II курсу, групи КСМм-19-1
Остапенко Т. О.
(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: ст. викл. Мовсесян Я. 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 Остапенку Тимофію Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання рельєфу 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 компютерна графіка.

Метод зворотнього трасування.

Фінітні функції.

4. Перелік питань, що потрібно опрацювати в роботі

Огляд літератури, постановка завдання дослідження.

Розробка спостерігача на основі метода зворотнього трасування.

Розробка моделі рельєфу представлений у вигляді карт висот, з використанням фінітних функцій та на основі метода зворотнього трасування.

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	Розробка спостерігача на основі метода	11.11.20 – 20.11.20	
	зворотнього трасування		
3	Розробка моделі рельєфу за допомогою карт	21.11.20 – 30.11.20	
	висот та на основі фінітних функцій		
4	Оформлення пояснювальної записки та	01.12.20 – 08.12.20	
	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: 75 с., 20 рис., 3 дод., 12 джерел.

МОДЕЛЬ РЕЛЬЄФУ, СИСТЕМИ ВІЗУАЛІЗАЦІЇ, АВІАЦІЙНІ ТРЕНАЖЕРИ, СИНТЕЗ ЗОБРАЖЕННЯ, РЕЙТРЕЙСІНГ.

Метою атестаційної роботи є підвищення реалістичності рельєфу в системах візуалізації для авіаційних тренажерів.

У ході виконання атестаційної роботи на водяться моделювання земної поверхні, поданої у вигляді цифрової карти висот, за допомогою фінітних функцій. Даний опис дозволяє одержати високу реалістичність зображення і має невелику обчислювальну складність. Синтез поверхні, таким способом, виконується зворотнім трасуванням. Побудова зображення в реальному часі методом зворотного трасування проводиться шляхом покрапкового обчислення центральної проекції на поверхню об'єкта сцени елемента екрана, що сканується в даний момент в ході растрової розгортки, що надає високу реалістичність.

ABSTRACT

Master’s thesis: 75 pages, 20 figures, 3 appendices, 12 sources.

RELIEF MODEL, VISUALIZATION SYSTEMS, AVIATION SIMULATORS, IMAGE SYNTHESIS, RAY TRACING.

The major goal of this thesis is increasing the realism of the terrain in visualization systems for flight simulators.

During the attestation work, the modeling of the earth's surface, presented in the form of a digital map of altitudes, using finite functions. This description allows to obtain a high realism of the image and has low computational complexity. The synthesis of the surface, in this way, is performed by reverse tracing. The construction of a real-time image by the method of reverse tracing is performed by spot calculation of the central projection on the surface of the object of the scene of the screen element, which is currently scanned during raster scanning, which provides high realism.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

2D – двовимірна графіка

3D – тривимірна графіка

ЕЛТ – електро-променева трубка

ПП – проекційний промінь

РГ – растрова графіка

СВ – система візуалізації

СЗ – синтез зображення

ТП – тріангулірувана поверхня

RTX – рейтресінг

ВСТУП

Програмні забезпечення обчислювального типу на сьогоднішній день займають далеко не останнє місце у різноманітних сферах наукової та практичної діяльності. Вони невпинно розвиваються, вдосконалюються, розширяються їх можливості та спектр впливу. Особливо інтенсивно розвивається напрямок комп'ютерного синтезу зображень, таких як комп'ютерна графіка. Її можна визначити як науку про математичне моделювання геометричних форм та вигляду об'єктів, а також методів їх візуалізації.

Одним з новіших напрямків комп'ютерної графіки є розробка принципів, та методів формування реалістичних зображень. Потреба в створенні подібних зображень виникає в таких областях, як дизайн, машинобудівне та архітектурне проектування, реклама і т.д. Найчастіше реалістичність використовується для підвищення емоційного впливу зображення, як, наприклад, в рекламі, в інших же галузях – як засіб оцінки якості обраних рішень, що забезпечує реалістичну обстановку наприклад, відеотренажерів, систем розпізнавання образів. У системах формування реалістичних зображень повинна забезпечуватися передача всієї сукупності образотворчих властивостей: об'ємність, розташування предметів в сюжеті, колір, текстура поверхні.

Стрімкий розвиток сучасних технологій також тягне за собою розвиток сфери комп'ютерної графіки. Розвиваються системи, які забезпечують відображення динамічних сюжетів (в яких зображення поступово змінюють один одне). З таких систем можна відзначити три групи:

- системи графічного моделювання для наочного уявлення процесів в хімії, медицині, астрономії;
- системи імітації динамічних ситуацій таких як відеотренажери;
- системи отримання 2D та 3D зображень для телебачення й кіно.

Саме в розвитку цих систем найбільше проявляються труднощі та проблеми тривимірної машинної графіки. Для них потрібно не тільки висока точність моделей, а й надзвичайно висока продуктивність обчислювальних засобів. Створення та впровадження в практику обчислювальних систем створюють в даний час реальні перспективи подальшого розвитку можливостей комп'ютерної графіки.

У тривимірної комп'ютерної графіки реалістичних зображень провідну роль займає метод трасування променів, в основі якого лежить відтворення в математичній формі ходу променів в реальних пристроях формування зображень. Якщо синтезується реалістичне зображення, то надлишкові витрати бувають вельми не великими. Вони великі лише тоді, коли синтезується умовно-об'ємне, стилізоване зображення. Поряд з цим метод зворотного трасування променів має безперечні переваги: універсальність, простота його фізичного трактування і, що дуже важливо, – можливість розпаралелювання обчислень. Це практично дозволяє проводити синтез для кожної точки зображення незалежно від інших.

У системах візуалізації тренажерів транспортних засобів виникає необхідність синтезувати та відображати на екрані земну поверхню. Авіатренажер – ціла система проекторів та моніторів, встановлена навколо кабіни, яка створює простір віртуальної реальності для пілота. Завдяки цьому можна побачити навколо себе те саме, що в реальному польоті. Авіасимулятор дозволяє відтворити грозу, польоти в горах, над морем та сушею, а також випробувати десятки різних літаків. Частіше при цьому до системи пред'являються одночасно вимоги високої точності, реалістичності та велике відображення площі. Існуюча практика синтезу зображення земної поверхні на основі різних полігональних методів не володіє високою реалістичністю. Такі методи, як параметричний, сплайн-інтерполяція є обчислювально-об’ємними та використовувати їх в системах реального часу важко. Тому розробка методів описаної поверхні та метод побудови тривимірних моделей являється актуальним.

1 ОГЛЯД ЛІТЕРАТУРИ ТА ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ

1.1 Комп'ютерна графіка та області застосування

Графіка визначається як будь-який ескіз, креслення або рисунок. Комп’ютерна графіка використовується там, де набором зображень потрібно маніпулювати або створювати зображення у вигляді пікселів та відображається на комп’ютері. Комп’ютерна графіка може бути використана у цифровій фотографії, кіно, розвагах, електронних гаджетах та у всіх інших основних необхідних технологіях. Це величезна тема та область у галузі інформатики[1,2].

Із розвитком комп'ютерних технологій розвивався і напрямок комп'ютерної графіки.

У 1950 роках, проекти Whirlwind та SAGE представили ЕЛТ (електронно-променеву трубку) як життєздатний інтерфейс для відображення та взаємодії та представили світлову ручку як пристрій введення.

У 1960 роках, Вільям Феттер, графічний дизайнер Boeing, ввів фразу «комп'ютерна графіка». Іван Сазерленд винайшов перший головний дисплей, керований комп'ютером (HMD). Його назвали Дамокловим мечем через апаратне забезпечення, необхідне для підтримки, воно відображало два окремі зображення каркасів, по одному на кожне око. Це дозволило глядачеві побачити комп'ютерну сцену в стереоскопічному 3D.

У 1970 роках, Джон Уорнок був першим піонером, який згодом заснував Adobe Systems і створив програмне забезпечення для редагування фотографій в Adobe Photoshop. Аркади для відеоігор народилися з першими аркадними іграми з використанням 2D-графіки спрайтів у режимі реального часу. Деякі з найвідоміших аркадних ігор включають Pong, Space Invaders та Speed Race.

У 1980 роках, виникає модернізація та комерціалізація комп’ютерної

графіки. У міру поширення домашнього комп’ютера кількість розробників комп’ютерної графіки значно зросла. Наявність розрядних та 16-розрядних мікропроцесорів почала революцію в терміналах комп'ютерної графіки з високою роздільною здатністю, які тепер все більше стають інтелектуальними, напівзалежними та автономними.

Непереборною нотою 1990-х стало поява масового 3D-моделювання та загальне підвищення якості CGI.

У 1999 році Nvidia випустила основну GeForce 256, першу домашню відеокарту, що виставляється в якості блоку обробки графіки або графічного процесора, який, за його власними словами, містив «інтегровані механізми перетворення, освітлення та рендеринга».

У 1999 році, CGI всерйоз став повсюдним у цю епоху. Відеоігри та кінотеатри CGI поширили обсяги комп’ютерної графіки на загальнодоступну тематику наприкінці 1990-х і продовжували робити це прискореними темпами в 2000-х. Постійне зростання та вдосконалення графічного процесора були вирішальними для цього десятиліття, і можливості 3D-рендерингу стали стандартною особливістю, оскільки графічні процесори для 3D-графіки стали вважатися необхідністю виробників настільних комп'ютерів[2].

На початку половини 2010-х CGI майже повсюдно зустрічається у відео, попередньо відтворена графіка є майже науково фотореалістичною, а графіка в режимі реального часу на відповідній висококласній системі може імітувати фотореалізм для нетренованого ока. У відеоіграх Xbox One від Microsoft, Sony Playstation 4 та Nintendo Wii U в даний час домінують у домашньому просторі та всі вони здатні до високотехнологічної 3D-графіки, але ПК з Windows досі є однією з найактивніших ігрових платформ.

Основні області застосування.

Комп’ютери відіграють важливу роль у всіх сферах життя. Вони використовуються в будинках, бізнесі, навчальних закладах, дослідницьких організаціях, медичній галузі, державних установах, розважальних закладах тощо.

Медична галузь. Комп’ютери використовуються в лікарнях для ведення бази даних історії хвороби, діагностики, рентгенівських знімків, моніторингу пацієнтів у реальному часі тощо. В наш час хірурги використовують роботизовані хірургічні пристрої для виконання делікатних операцій та дистанційного проведення операцій. Технології віртуальної реальності також використовуються для навчальних цілей. Це також допомагає спостерігати за плодом всередині утроби матері.

Розваги. Комп’ютери допомагають дивитися фільми в Інтернеті, грати в відеоігри в Інтернеті; виступати в ролі віртуального артиста під час гри, прослуховування музики тощо. Інструменти MIDI дуже допомагають людям в індустрії розваг у записі музики на штучних інструментах. Відео можна передавати з комп'ютерів на повноекранні телевізори. Фоторедактори доступні з неймовірними функціями.

Промисловість. Комп’ютери використовуються для виконання кількох завдань у таких галузях, як управління запасами, проектування цілей, створення віртуальних зразків продуктів, дизайн інтер’єру, відеоконференції тощо. Інтернет-маркетинг зазнав великої революції у своїй здатності продавати різні товари в важкодоступні куточки, такі як інтер’єр або сільські райони райони. Фондові ринки бачили феноменальну участь різних рівнів людей завдяки використанню комп’ютерів.

Освіта. Комп’ютери використовуються в освітньому секторі за допомогою онлайн-класів, онлайн-іспитів, електронних книг. Вони допомагають збільшити використання аудіо-візуальних засобів у освітній галузі.

Уряд. У державних секторах комп’ютери використовуються для обробки даних, ведення бази даних громадян та підтримки безпаперового середовища. Оборонні організації країни отримали значну вигоду від комп'ютерів, які використовуються для розробки ракет, супутників, запусків ракет.

Банківська справа. У банківському секторі комп’ютери

використовуються для зберігання даних про клієнтів та проведення операцій, таких як зняття та внесення грошей через банкомати. Банки значною мірою зменшили помилки та витрати, пов’язані з ручною працею, завдяки широкому використанню комп’ютерів.

Бізнес. У наш час комп’ютери повністю інтегровані у бізнес. Головною метою бізнесу є обробка транзакцій, яка включає операції з постачальниками, працівниками або замовниками. Комп’ютери можуть зробити ці операції легкими та точними. Люди можуть аналізувати інвестиції, продажі, витрати, ринки та інші аспекти бізнесу за допомогою комп’ютерів.

Навчання. Багато організацій використовують дистанційне навчання для навчання своїх співробітників, для економії грошей та підвищення ефективності роботи. Відеоконференції за допомогою комп’ютерів дозволяють заощадити час та дорожні витрати завдяки можливості зв’язку людей у різних місцях[3].

Мистецтво. Комп’ютери широко використовуються в танцях, фотографії, мистецтві та культурі. Плавний рух танцю можна показати в прямому ефірі за допомогою анімації. Фотографії можна оцифрувати за допомогою комп’ютерів.

Наука та техніка. Комп’ютери з високою продуктивністю використовуються для стимулювання динамічних процесів у науці та техніці. Суперкомп'ютери мають численні програми в галузі досліджень та розробок (НДДКР). Топографічні зображення можна створювати за допомогою комп’ютерів. Для кращого розуміння землетрусів вчені використовують комп’ютери для побудови та аналізу даних.

Види комп'ютерної графіки.

Двовимірна комп'ютерна графіка класифікується за типом представлення зображень, і випливаючими з цього алгоритмами обробки графічної інформації.

Перш ніж об’єкт можна буде показати на моніторі комп’ютера або принтері, потрібна модель, що описує геометрію об’єкта, якщо об’єктом не є саме зображення. Моделювання геометричних об'єктів зазвичай проводиться

в рамках векторно-орієнтованої або векторної графіки. Більш складний об'єкт моделюється як поєднання елементарних об'єктів, таких як лінії, прямокутники, кола, еліпси або дуги. Кожен з цих елементарних об'єктів може бути визначений кількома координатами, що описують місце розташування об'єкта, та деякими параметрами, такими як радіус кола. Дуже простий опис будинку на рисунку 1.2 з точки зору векторної графіки наведено на рисуснок 1.3. Будинок можна визначити як послідовність точок або векторів. Також слід вказати в послідовності точок, чи слід поєднувати дві сусідні точки лінією чи ні. Пунктирні лінії на рисисунок 2 відносяться до точок у послідовності, які не повинні з'єднуватися лінією.

Орієнтований на векторну графіку опис об’єктів безпосередньо не підходить для подання тільки на піксельних пристроях, таких як РК-монітор або принтер. З теоретичної точки зору можна було б відображати векторну графіку безпосередньо на моніторі CRT1, проводячи катодний промінь – або, у випадку кольорового відображення, три катодні промені – по лініях, визначених послідовністю точок, і перемикачем промінь увімкнено або вимкнено, залежно від того, чи слід провести відповідну сполучну лінію.

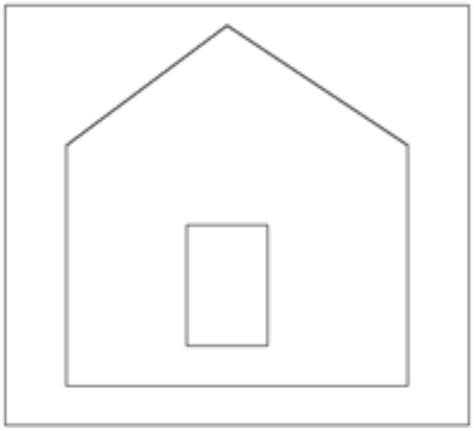

Рисунок 1.1 – Оригінальне зображення

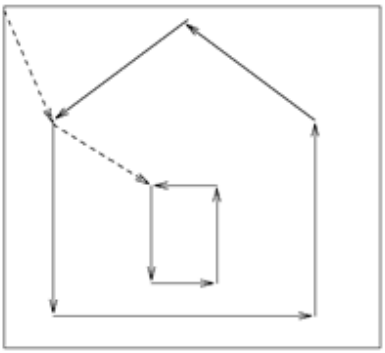

Рисунок 1.2 – Векторне зображення

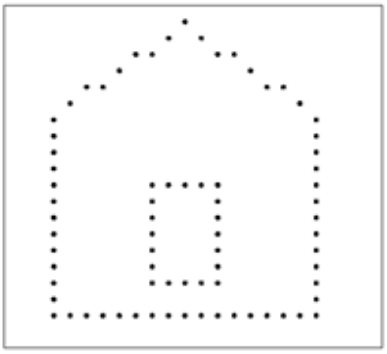

Рисунок 1.3 – Піксельне зображення

У цьому випадку монітор може більше не мерехтіти, оскільки катодному променю може знадобитися занадто багато часу, щоб оновити екран для отримання більш складного зображення у векторній графіці, так що флуоресцентні плями на екрані можуть зникнути до повернення катодного променя. Монітори без мерехтіння повинні мати частоту оновлення 60 Гц. Якщо катодний промінь повинен проходити вздовж контурних ліній об'єктів, представлених у векторній графіці, швидкість оновлення залежала б від того, скільки рядків містять об'єкти, так що в цьому робочому режимі не може бути гарантована досить швидка частота оновлення. Отже, катодний промінь сканує лінію екрану за лінією, що приводить до гарантованої та постійної частоти оновлення, незалежно від зображення, яке потрібно намалювати.

Комп’ютерні монітори, принтери, а також різні формати для зберігання зображень, таких як растрові зображення або JPEG, базуються на растровій або растрово-орієнтованій графіці, яку також називають піксельною або піксельно-орієнтованою графікою. Растрова графіка використовує піксельну матрицю фіксованого розміру. Кожному пікселю растру можна призначити колір. У найпростішому випадку чорно-білого зображення піксель приймає одне з двох значень чорний або білий.

Для відображення векторної графіки у вигляді растрової графіки всі геометричні фігури повинні бути перетворені в пікселі. Ця процедура називається перетворенням сканування. З одного боку, це може призвести до великих обчислювальних зусиль. Стандартний монітор має більше одного мільйона пікселів. Для кожного з них потрібно вирішити, який колір йому призначити для кожного зображення. З іншого боку, небажані ефекти згладжування виникають у вигляді нерівних країв, відомих як виїмки або сходи. Термін ефект згладжування походить з області обробки сигналів і стосується артефактів, тобто поверхневих небажаних ефектів, які можуть виникнути, коли для вимірювання безперервного сигналу використовується дискретна частота дискретизації. Зображення в масштабі сірого можна розглядати як двовимірний сигнал. Кольорове зображення на основі трьох кольорів - червоного, зеленого та синього - це не що інше, як три двовимірні сигнали, по одному для кожного кольору.

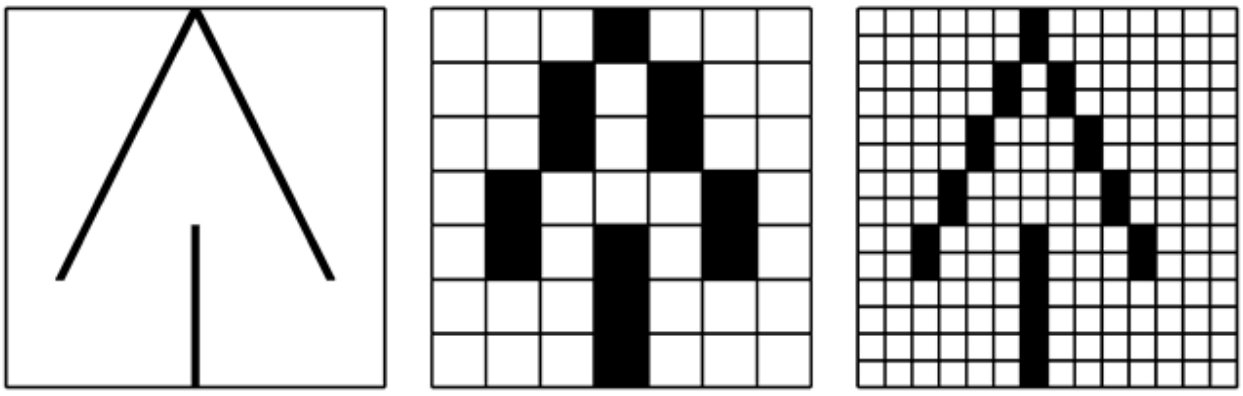

Рисунок 1.4 – Кінчик стрілки, зображений у вигляді растрової графіки у двох різних роздільних здатностях

Навіть якщо зображення буде відображатися з точки зору растрово-орієнтованої графіки, воно все одно має переваги у моделюванні та зберіганні у векторно-орієнтованому форматі. Растрова графіка (РГ) прив’язана до певної роздільної здатності. Як тільки роздільна здатність буде зафіксована, повна інформація, що міститься у векторно-орієнтованому зображенні, більше не може бути відновлена, що призводить до серйозних недоліків, коли зображення відображається на пристрої з іншою роздільною здатністю або коли зображення потрібно збільшити або зменшити. На рисунку 1.4 показано кінчик стрілки та її зображення у вигляді растрової графіки для двох різних роздільних здатностей.

1.2 Тривимірна комп’ютерна графіка та її застосування

Тривимірна графіка або 3D графіка - це сфера комп’ютерної графіки, сукупність прийомів та інструментів, що дозволяють створювати тривимірні об’єкти за допомогою фігур та кольорів. Він відрізняється від двовимірних зображень створенням геометричної проекції тривимірної моделі сцени (віртуального простору) у 2D. Це виконується за допомогою спеціалізованого програмного забезпечення. Отримана модель може бути такою ж, як реальні об'єкти (наприклад, будівля, людина, машина, астероїд), або бути абсолютно абстрактною (проекція чотиривимірного фракталу)[2,4].

Сьогодні тривимірна графіка міцно увійшла в багато сфер нашого життя:

- будівництво, візуалізація об'ємних зображень архітектурних будівель, об'єктів, інтер'єру та екстер'єру;
- виробництво, об'єктне моделювання;
- телевізор, імітація зображень у глянцевих журналах, відеокліпи, спецефекти у фільмі;
- ігрова індустрія, 3D-анімація та віртуальні світи, розробка комп’ютерних ігор;

- поліграфія, створення друкованої продукції;
- комерційні, такі як електронні презентації та каталоги, білборди.

3D-графіка є одним з найефективніших інструментів реклами, що дозволяє збільшити вплив на потенційного клієнта та підвищити якість реклами, що представляє як реальний, так і віртуальний світ.

Застосування.

Тривимірна графіка активно використовується для створення зображень на плоскому екрані або аркуші друкованих матеріалів у науці та промисловості. Найширше її використовують в багатьох сучасних комп’ютерних іграх, а також як елемент кіноіндустрії, телебачення та друкованої продукції.

Тривимірна графіка зазвичай має справу з віртуальним, уявним тривимірним простором, який повинен відображатися на плоскій, двовимірній поверхні або папері. На сьогоднішній день відомо кілька способів відображення тривимірної інформації в об’ємній формі, хоча більшість із них представляють тривимірні характеристики з багатьма умовами, оскільки вони працюють зі стереозображеннями. Стерео окуляри, віртуальні шоломи, 3D-дисплеї, здатні продемонструвати тривимірні зображення[4].

1.3 Підвищення реалістичності тривимірной графіки

В даний час зображення тривимірних сцен для систем візуалізації синтезуються в основному двома методами: прямим та зворотнім трасуванням променів. На сучасному етапі розвитку комп'ютерної графіки найбільш розвинений метод прямого трасування. Метод зворотного трасування, хоча і вимагає великих обчислювальних та апаратних витрат для своєї реалізації, але має ряд істотних переваг: обробка динамічних джерел світла та тіней, розрахунок вторинного освітлення, заломлення та

віддзеркалення променів світла. Все це дозволяє даним методом синтезувати більш реалістичне зображення. Тому розробка підсистем та вузлів СВ, що працює по зворотному методу, актуальна. Зараз цей напрямок активно досліджується як в нашій країні, так і за кордоном. Є розробки в області синтезу зображень хмар, рельєфу, об'єктів, що складаються з сукупності поверхонь другого порядку та площин; розглядаються питання візуалізації джерел світла.

У сучасній комп'ютерній графіці через переважання методу прямого трасування уявлення 3D-об'єктів в вигляді тріангулірованої поверхні (ТП) є стандартом. До переваг ТП відносяться: простота та універсальність її математичного уявлення, зручність побудови, трансформації та перетворення; можливість отримання в тріангулярному вигляді геометрії багатьох об'єктів реального світу. Існує безліч бібліотек ТП для вільного використання, наприклад, в Стенфордському університеті (The Stanford 3D Scanning Repository). Рішення завдання ефективного синтезу зображень ТП методом зворотного трасування є актуальним, так як дозволить зберегти для цього методу спадкоємність раніше отриманих результатів, зменшити час на підготовку або зміну 3D сцени і моделей об'єктів в ній. Класичний підхід до синтезу зображень ТП зворотним методом має на увазі пошук точки перетину проекційного променя (ПП) з трикутником і інтерполяцію освітленості. Широко відомий алгоритм пошуку перетину проекційного променя з трикутником, а також його модифікації для оптимальної обробки ТП. Дані алгоритми вважаються найбільш прийнятними для апаратної реалізації в порівнянні з іншими класичними методами, але не дозволяють досягти швидкодії порівнянного з прямим методом. При цьому синтезовані зображення нарівні з достоїнствами зворотного методу мають всі недоліки зображень, синтезованих прямим методом. Основним недоліком слід вважати відсутність взаємозв'язку кутової похибки СВ та геометричних

спотворень, що виникають в процесі синтезу зображення поверхні. В цьому випадку з певних ракурсів видно плоскі грані трикутників поверхні, виникають неточності в процесі розрахунку освітленості. Очевидно, що при класичному підході до синтезу зображень збільшення кількості трикутників для більш точного уявлення геометрії поверхні веде до істотного зростання обсягу обчислень, особливо при високій роздільній здатності екрану. Введення різних класифікаційних описів геометрії тривимірних об'єктів, побудова вісімкових дерев (octree) зменшують залежність часу синтезу зображення від кількості трикутників, але різко ускладнюють апаратну реалізацію.

Компанія Nvidia не так недавно продемонструвала технологію RTX, яка повинна нарешті принести метод трасування променів в ігри. Трасування променів (рейтрейсінг) – метод далеко не новий. Стосовно до ігор про нього говорили ще років 20 назад, а сам термін щодо комп'ютерної графіки виник в 1982 році, і з тих пір в іграх метод так і не з'явився.

Трасування променів – це метод побудови тривимірних моделей, в якому використовується принцип, аналогічний реальним фізичним процесам. Тобто для побудови об'єкта система відстежує траєкторію віртуального променя від екрану до цього самого об'єкту.

В реальності ми бачимо не об'єкти самі по собі, а відбите від них світло. Винятком є об'єкти, які самі служать джерелами світла. Метод трасування променів використовує приблизно ті ж принципи стосовно віртуальному середовищі[6].

Проблема в тому, що такий метод виходить вкрай витратним з точки зору вимог до апаратних ресурсів. Якщо при використанні звичних методів рендеринга той же колір або прозорість матеріалу об'єкта задаються спочатку, а відображення та тіні емулюються в тому числі за допомогою шейдерів і інших хитрувань, то у випадку з рейтрейсінгом ці характеристики

визначаються саме в процесі взаємодії тих самих віртуальних променів з об'єктом, як і в реальності. Це вимагає просто колосальних витрат з боку GPU навіть в разі якихось окремих об'єктів, не кажучи вже про використання рейтрейсінга в іграх в якості основного методу побудови об'єктів.

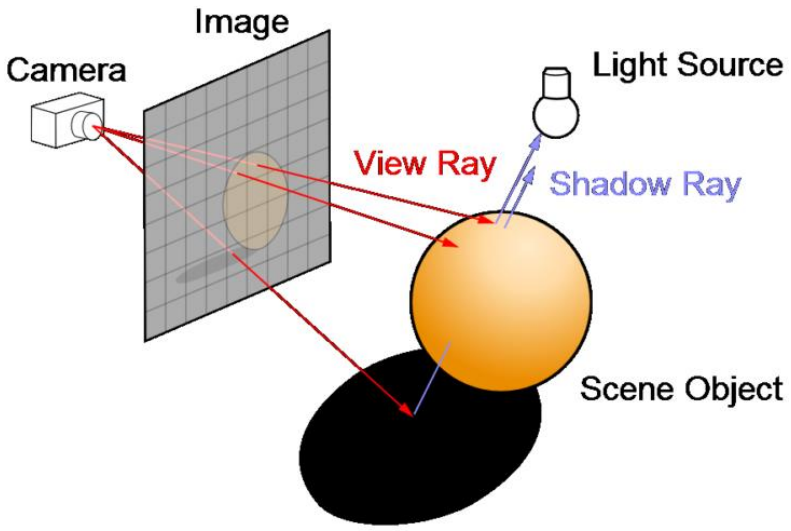

Рисунок 1.5 – Трасування променів

Для побудови зображення роздільною здатністю 1024 x 768 пікселів за допомогою рейтрейсінга необхідно сформувати 768 432 променя. При цьому кожен промінь може як відбиватися, так і переломлюватися, що в підсумку призводить до збільшення кількості трасуємих променів в кілька разів. І якщо у випадку звичайних методів рендеринга необхідні полігони просто потрібно встигнути відмалювати, то в разі рейтрейсінга кожен промінь вимагає постійних математичних розрахунків, починаючи з моменту його випускання. Саме з цієї причини рейтрейсінг вже давно використовується там, де не потрібно робота методу в режимі реального часу. Nvidia стверджує, що раніше не існувало графічних адаптерів, що мають достатню продуктивність для такої роботи. А тепер ніби є Volta, яка повинна апаратно прискорювати трасування променів. Правда, ніяких подробиць щодо цього немає. Поки абсолютно неясно, наскільки активно розробники ігор зможуть використовувати рейтрейсінг, і що гравці отримають на виході.

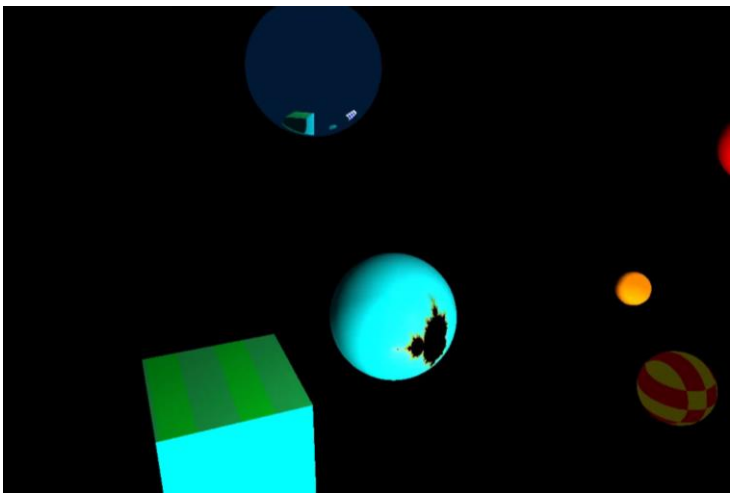

Рисунок 1.6 – Рейтресінг.

Очікувана багатьма Metro Exodus. Вона буле першою грою AAA-класу, що використовує технологію RTX. Насправді ж рейтрейсінг в Metro буде використовуватися для деяких ефектів глобального освітлення. Якщо точніше, для моделі затінення ambient occlusion та для непрямого освітлення indirect lighting. При цьому класична растеризация нікуди не дінеться.

1.4. Постановка завдання дослідження

Проведений вище огляд дозволяє зробити наступні висновки:

- для синтезу зображень в реальному часі добре опрацьований метод прямого трасування, однак він має низьку реалістичність;
- метод зворотного трасування дозволяє синтезувати зображення з високою реалістичністю, проте він недостатньо добре опрацьований для роботи в реальному часі;
- існуюча практика синтезу зображення земної поверхні на основі різних полігональних методів не володіє високою реалістичністю.

У зв'язку з вищевикладеним визначимо такі основні завдання, вирішенню яких присвячена дана робота:

- дослідження рельєфу в системах візуалізації для авіаційних тренажерів орієнтованих на метод зворотного трасування та за допомогою фінітних фукнцій;
- виконати моделювання досліджуваних моделей.

2 ФОРМУВАННЯ РЕЛЬЄФУ В СИСТЕМАХ ВІЗУАЛІЗАЦІЇ

2.1 Математична модель геометричних перетворень для спецпроцесорів растрової графіки

Розглянуто математичну модель центропроективних геометричних перетворень об'ємної сцени. Показана доцільність її використання при побудові спецпроцесорів растрової графіки реального масштабу та часу.

Побудова зображення в реальному часі методом зворотнього трасування проводитися шляхом покрапкового обчислення центральної проекції на поверхню об'єкта сцени елемента екрана (пікселя), що сканується в ході растрової розгортки та може використовуватися, наприклад, в системах візуалізації (СВ) тренажерів різного призначення [7].

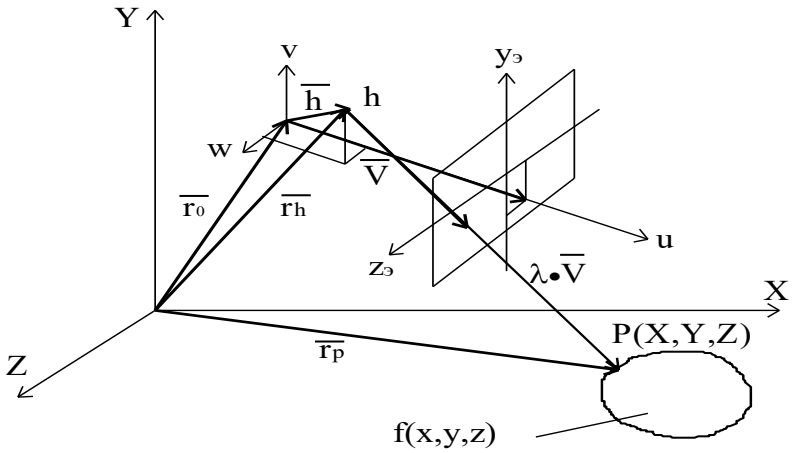

Рисунок 2.1 – Основні геометричні елементи представленої задачі

Перейдемо до опису параметризації центропроективного перетворення. На рисунку 2.1 представлені основні геометричні елементи даної задачі. Введемо праву «земну» систему координат X, Y, Z (g-система) яка пов'язана з спостерігачем системи координат u, v, w (v-система), початок якої

розміщено в центрі ваги переміщающогося апарату (літака, машини і т.д.). Точка спостереження h (центр проекції) задана щодо початку координат v-системи радіус-вектором h (Uh, Vh, Wh).

Введемо позначення проекційного променя $\lambda \vec{V} = \vec{V}_p$.

Екран (площини проекції) розташований перпендикулярно осі U, так що його центр Oe в v-системі в загальному випадку має координати (Ue + Uh), Ve, We. У площині екрана введена система координат Ye, Ze з початком в центрі екрану. Введено вектор спостереження $\vec{V}$, який йде від точки спостереження h до центру поточного пікселя екрана. Проекцію точки екрану, яка визначається цим вектором, що відображається на поверхні f (x, y, z) = 0 або $f(\vec{r}) = 0$ шукаємо на проекційному промені $\lambda \vec{V}$, проведеному з центру проекції h в напрямку вектора $\vec{V}$ до перетину з поверхнею, що відображається, в точці P. Визначаємо параметр λ так, щоб виконувалися векторні рівняння:

$$\vec{r}_p = \vec{r}_h + \lambda \vec{V} \ , \qquad (2.1)$$

де λ – скаляр.

Тоді маємо:

$$f(\vec{r}_p) = f\left[(X_h + \lambda V_x), (Y_h + \lambda V_x), (Z_h + \lambda V_z)\right] = 0. \qquad (2.2)$$

У загальному випадку проекційний промінь $\lambda \vec{V}$ перетинає поверхню в декількох місцях. Однак, на екрані буде видно найближчу непрозору точку I, отже:

$$\lambda = \text{Argmin} \left[f(\vec{r}) = 0 \right] \qquad (2.3)$$

$$\lambda > 0$$

Умова $\lambda > 0$ природно відділяє потрібний напрямок проекційного променя на екрані. У цьому рівнянні компоненти векторів $\vec{r}_h$ та $\vec{V}$ повинні бути задані в g-системі.

Розглянемо практично важливі приклади вирішення рівняння (2.3).

Приклад 1. Нехай поверхня f (x, y, z) = 0 є сферою в площинах XZ g-системи торкається поверхні сфери (Землі). Рівняння сфери у векторній формі має вигляд:

$$\left| \vec{r} - \vec{r}_1 \right| = R, \qquad (2.4)$$

де $\vec{r}$ – радіус-вектор (в g-системі) до будь-якої точки P поверхні сфери радіуса R;

$\vec{r}_1 = r1 * \vec{n} = (D + R) * \vec{n}$ - радіус-вектор (в g-системі) до центра сфери;

$\vec{n}$ – одиничний вектор;

d – відстань до поверхні сфери в напрямку радіус-вектора.

Перепишемо (2.4) у вигляді:

$$\vec{r} + \vec{R}_p - \vec{r}_1 = 0, \qquad (2.5)$$

де $\vec{R}_p$ – вектор, проведений з точки P до центру сфери.

Помножимо ліву та праву частини (2.5) на $\vec{n}$:

$$\vec{r} \cdot \vec{n} + \vec{R}_p * \vec{n} - \vec{r}_1 \cdot \vec{n} = 0.$$

Тоді маємо:

$$\vec{r} \cdot \vec{n} + R \cdot \cos\alpha - d - R = 0; \ \vec{r} \cdot \vec{n} - (d + dc) = 0, \qquad (2.6)$$

де $dc = R \cdot 1 - \cos\alpha$);

α – кут між векторами $\vec{R}_p$ та $\vec{n}$.

Для випадків, що розглядаються в прикладі, d = 0, тоді

$$\vec{r} \cdot \vec{n} - dc = 0, \tag{2.7}$$

де $dc = 2R*\cos 2\beta$;

β – кут між векторами $\vec{r}$ та $\vec{r}_1$.

З (2.1), (2.3) і (2.7) знайдемо рішння для λ:

$$\lambda = -\frac{\vec{n} \cdot \vec{r}_h - d_c}{\vec{n} \cdot \vec{V}}, \tag{2.8}$$

або в більш докладній формі запису:

$$\lambda = -\frac{n_x X_h + n_y Y_h + n_z Z_h - d_c}{n_x V_x + n_y V_y + n_z V_z}. \tag{2.9}$$

Приклад 2. Поверхня f (x, y, z) при синтезі зображення може бути задана як кусочно-плоска. Знайдемо рішення для λ в разі перетину вектора $\vec{r}_p$ з площиною, що є елементом поверхні f (x, y, z), і задається векторним рівнянням

$$\vec{r} \cdot \vec{n} - d = 0, \tag{2.10}$$

де $\vec{r}$ – радіус-вектор в g-системі до точки на поверхні площини;

$\vec{n}$ – одиничний вектор нормалі до площини;

d – параметр площини (найкоротша відстань від початку координат g-системи до площини).

Підставляючи (2.10) в (2.3) з урахуванням (2.1) маємо:

$$\lambda = -\frac{\vec{n} \cdot \vec{r}_h - d}{\vec{n} \cdot \vec{V}} \ , \qquad (2.11)$$

або:

$$\lambda = -\frac{n_x X_h + n_y Y_h + n_z Z_h - d}{n_x V_x + n_y V_y + n_z V_z} \ . \qquad (2.12)$$

Приклад 3. Розглянемо найбільш простий випадок, коли f (X, Y, Z) = y, тобто площина паралельна площині XZ g-системи.

Тоді в співвідношенні (2.12) маємо nx = nz = 0; ny = 1; d = dy і, отже,

$$\lambda = - (Y_h - dy) / V_y. \qquad (2.13)$$

У разі, якщо f (x, y, z) є площина XZ g-системи, маємо dy = 0 і

$$\lambda = - Y_h / V_y. \qquad (2.14)$$

З отриманих рішень (2.8), (2.11), (2.13) випливає, що в загальному вигляді вираз для λ можна узагальнити так:

$$\lambda = F / G, \qquad (2.15)$$

де функція F пов'язує параметри точки спостереження і параметри поверхні, а функція G – скалярний добуток вектора спостереження V з одиничним вектором n.

Розпишемо векторне рівняння (2.1) з урахуванням всіх векторних складових:

$$\vec{r}_h = \vec{r}_0 + \vec{h} + \lambda \vec{V}. \qquad (2.16)$$

Для знаходження координат точки перетину проекційного променя з відображеною поверхнею в g-системі необхідно попередньо виконати перетворення компонентів всіх векторів, що входять в (2.15), (2.16) з v-системи в g-систему. До таких векторів відносяться вектори $\vec{h}$ і $\vec{V}$, так як їх компоненти задані в v-системі. Взаємне розташування v-системи щодо g-системи задається радіус-вектором $\vec{r}_0$ і орієнтацією системи, яку в загальному випадку можна задати по різному. Скористаємося прийнятим в авіації способом задання орієнтації v-системи за допомогою кута крену γ, кута тангажу υ і кута рискання (курсу) ψ . Для виконання перетворень необхідно побудувати матрицю поворотів A за прийнятими кутами обертання ψ, υ, γ. Результуюча матриця поворотів A виходить шляхом перемноження елементарних матриць. Оскільки множення матриць в загальному випадку не задовольняє властивості комутативності, то задамо наступний порядок множення матриць[11]:

- Ay_ψ – поворот на кут ψ навколо осі Y;
- Az_υ – поворот на кут υ навколо осі Z;
- Ax_γ – поворот на кут γ навколо осі X.

Тоді для обчислення результуючої матриці маємо:

$$A\psi,\upsilon,\gamma = A\psi \cdot A\upsilon \cdot A\gamma. \qquad (2.17)$$

У загальному випадку результуюча матриця А має вигляд:

$$A = \begin{pmatrix} a_{xu} & a_{xv} & a_{xw} \\ a_{yu} & a_{yv} & a_{yw} \\ a_{zu} & a_{zv} & a_{zw} \end{pmatrix}, \qquad (2.18)$$

де елементи матриці являють собою напрямні косинуси.

Введемо позначення рядків в матриці А відповідно Ax, Ay, Az. Тоді, враховуючи ортогональність систем, для довільного вектора $\vec{R}$ (U, v, w), заданого в v-системі, отримаємо компоненти цього вектора в g-системі з наступних матричних співвідношень:

$$\begin{aligned} Rx &= (Ax) \cdot (Ru, Rv, Rw)\ T \\ Ry &= (Ay) \cdot (Ru, Rv, Rw)\ T \\ Rz &= (Az) \cdot (Ru, Rv, Rw)\ T, \end{aligned} \qquad (2.19)$$

де верхній індекс T позначає операцію транспонування.

Використовуючи співвідношення (2.15), (2.16), (2.19) знайдемо в загальному випадку значення координат точки P (X, Y, Z) (Рис. 2.1):

$$\begin{aligned} X &= r0x + hx + \lambda Vx \\ Y &= r0y + hy + \lambda Vy \\ Z &= r0z + hz + \lambda Vz, \end{aligned} \qquad (2.20)$$

де компоненти векторів $\vec{h}$ і $\vec{V}$ в матричної записи мають вигляд:

$$\begin{aligned} h\ (X, Y, Z) &= (Ax, y, z)\ (hu, hv, hw)\ T \\ V\ (X, Y, Z) &= (Ax, y, z)\ (Vu, Vv, Vw)\ T. \end{aligned} \qquad (2.21)$$

Аналіз співвідношень (2.20) і (2.21) дозволяє оцінити для кожного пікселя необхідний обсяг обчислень, який суттєво перевищує обчислювальні можливості універсальних ПЕОМ.

Остаточно відзначимо, що для реалізації растрової графіки методом зворотного трасування в реальному масштабі часу (приблизно ≤ 50 нс / піксель) необхідно для кожного пікселя в темпі його промальовування виконати обчислення співвідношень (2.15), (2.20), (2.21), що вимагає розробки спеціальної обчислювальної техніки, спецпроцесорів растрової графіки.

2.2. Математична модель геометричної обробки зображення на площині для растрових систем візуалізації

На рисунку 2.2 представлені основні геометричні елементи даної задачі.

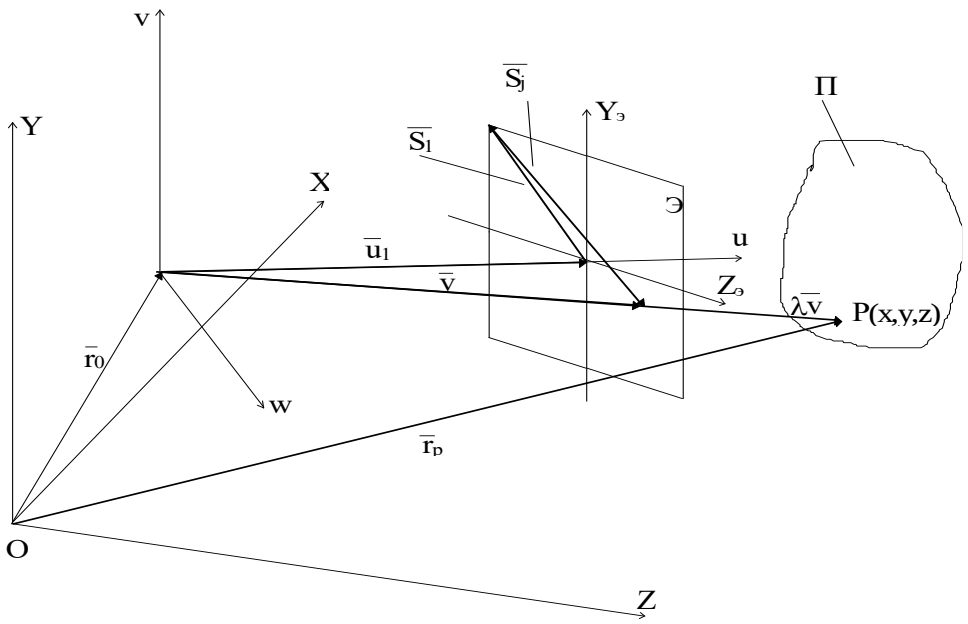

Рисунок 2.2 – Основні геометричні елементи

Показана права декартова v-система відліку (u, v, w), на початку координат якої знаходиться центр проекції (точка спостереження). Перпендикулярно до осі u розташована площина проекції (екран – е) зі своєю системою координат (с / к) Ye,Ze. Ось u проходить через центр екрану. Осі v і w паралельні осях Ye, Ze. Вважаємо, що предметна площина П, в якій розташовується плоске зображення, поєднана з площиною XZ системи відліку (X, Y, Z), в подальшому g-система[8].

На рисунку 2.2 прийняті позначення:

- u1 – видалення площини екрану від центру проекції;
- $\vec{r}_0$ – радіус-вектор центра проекції в с / к (X, Y, Z);
- $\vec{v}$ – вектор спостереження (вектор з центру проекції до j, k-ому пікселу на екрані);
- $\lambda\vec{v}$ – проекційний промінь, проведений з центру проекції в напрямку V до перетину з площиною П в точці Р;
- множник λ – скалярна величина;
- $\vec{s}_1$ – вектор з центру с / к Ye, Ze в лівий верхній кут екрану;
- $\vec{s}_{jk}$ – вектор з лівого верхнього кута екрану до j, k-ому пікселю;
- $\vec{r}_p$ – радіус-вектор точки Р.

З рисунку 2.2 випливає:

$$\vec{r}_p = \vec{r}_0 + \lambda \cdot \vec{v}. \qquad (2.22)$$

Рівняння площини у векторній формі має вигляд:

$$\vec{r} \cdot \vec{n} - d = 0, \qquad (2.23)$$

де $\vec{n}$ – одиничний вектор нормалі до площини;

d – параметр, що задає відстань площині від початку відліку в с / к (X, Y, Z).

У загальному випадку, з урахуванням (2.22) і (2.23) співвідношення для λ має вигляд:

$$\lambda = -\frac{\vec{r}_0 \cdot \vec{n} - d}{\vec{n} \cdot \vec{v}}. \qquad (2.24)$$

Так як в нашому випадку площина П збігається з XZ, то маємо:

$$d = 0,\ nx = nz = 0,\ ny = 1. \qquad (2.25)$$

Отже:

$$\lambda = -Y0 / Vy. \qquad (2.26)$$

З (2.22) і (2.26) знайдемо координати точки P в площині X, Z:

$$\begin{aligned} x &= X_0 - Y_0 \frac{V_x}{V_y} \\ z &= Z_0 - Y_0 \frac{V_z}{V_y} \end{aligned}, \qquad (2.27)$$

Для того, щоб знайти координати (2.27) в g-системі, виконаємо перетворення компонент вектора $\vec{v}$ з v-системи в g-систему за допомогою матриці обертання [1].

Нехай матриця перетворення має вигляд:

$$A = \begin{pmatrix} a_{xu} & a_{xv} & a_{xw} \\ a_{yu} & a_{yv} & a_{yw} \\ a_{zu} & a_{zv} & a_{zw} \end{pmatrix}. \qquad (2.28)$$

Позначимо рядки матриці A:

$$\begin{aligned} A_x &= \begin{pmatrix} a_{xu} & a_{xv} & a_{xw} \end{pmatrix}, \\ A_y &= \begin{pmatrix} a_{yu} & a_{yv} & a_{yw} \end{pmatrix}, \\ A_z &= \begin{pmatrix} a_{zu} & a_{zv} & a_{zw} \end{pmatrix}. \end{aligned} \tag{2.29}$$

Використовуючи співвідношення (2.29), виконаємо перетворення вектора $\vec{v}$:

$$\begin{aligned} V_x &= \begin{pmatrix} A_x \end{pmatrix} \cdot \begin{pmatrix} V_u & V_v & V_w \end{pmatrix}^T = a_{xu} \cdot V_u + a_{xv} \cdot V_v + a_{xw} \cdot V_w, \\ V_y &= \begin{pmatrix} A_y \end{pmatrix} \cdot \begin{pmatrix} V_u & V_v & V_w \end{pmatrix}^T = a_{yu} \cdot V_u + a_{yv} \cdot V_v + a_{yw} \cdot V_w, \\ V_z &= \begin{pmatrix} A_z \end{pmatrix} \cdot \begin{pmatrix} V_u & V_v & V_w \end{pmatrix}^T = a_{zu} \cdot V_u + a_{zv} \cdot V_v + a_{zw} \cdot V_w. \end{aligned} \tag{2.30}$$

Компоненти вектора $\vec{v}$ в v-системі знайдемо із співвідношення:

$$\vec{v} = \vec{u}_1 + \vec{s}_1 + \vec{s}_{j,k}. \tag{2.31}$$

Введемо ще деякі позначення:

- Ny, Nz – півширина екрану в пікселях;
- δ_y, δ_z – лінійний розмір пікселя.

Тоді:

$$\begin{aligned} \vec{s}_1 &= jNy\delta_y - kNz\delta_z, \\ \vec{s}_{j,k} &= -jNj\delta_y + kNk\delta_z. \end{aligned} \tag{2.32}$$

де Nj і Nk відповідно координати j, k-го пікселя.

Підставимо (2.22) в (2.21):

$$\vec{v} = \lambda u1 + j\,(Ny - Nj)\delta_y + K\,(Nk - Nz)\delta_z. \tag{2.33}$$

Остаточно маємо компоненти вектора $\vec{v}$:

$$\begin{aligned} Vu &= u1, \\ Vv &= (Ny - Nj)\delta_y , \\ Vw &= (Nk - Nz)\delta_z . \end{aligned} \tag{2.34}$$

Введемо позначення:

$$Ny\delta_y = Ye0;\ Nz\delta_z = Ze0;\ Nj\delta_y = Ye;\ Nk\delta_z = Ze, \tag{2.35}$$

де Ye і Ze – поточні координати екрану, що формуються в процесі растрової розгортки.

Підставимо (2.34) і (2.35) в (2.10):

$$\begin{aligned} Vx &= axu\ U1 + axvYe0 - axvYe + axwZe - axwZe0, \\ Vy &= ayu\ U1 + ayvYe0 - ayvYe + aywZe - aywZe0, \\ Vz &= azu\ U1 + azvYe0 - azvYe + azwZe - azwZe0. \end{aligned} \tag{2.36}$$

Співвідношення (2.26) для проекцій вектора $\vec{v}$ перетворимо до виду:

$$\begin{aligned} V_z &= c_{11}z_э - c_{12}y_э + c_{13}, \\ V_x &= c_{21}z_э - c_{22}y_э + c_{23}, \\ V_y &= c_{31}z_э - c_{32}y_э + c_{33}, \end{aligned} \tag{2.37}$$

де:

$$\begin{aligned} c_{11} &= a_{zw} = \cos\psi \cdot \cos\gamma - \sin\psi \cdot \sin\vartheta \cdot \sin\gamma; \\ c_{12} &= a_{zv} = \cos\psi \cdot \sin\gamma + \sin\psi\sin\vartheta\cos\gamma; \\ c_{13} &= a_{zv}Y_{э0} - a_{zw}Z_{э0} + a_{zu}U_1; a_{zu} = -\sin\psi\cos\vartheta; \\ c_{21} &= a_{xw} = \sin\psi\cos\gamma + \cos\psi\sin\vartheta\sin\gamma; \\ c_{22} &= a_{xv} = \sin\psi\sin\gamma - \cos\psi\sin\vartheta\cos\gamma; \\ c_{23} &= a_{xv}Y_{э0} - a_{xw}Z_{э0} + a_{xu}U_1; a_{xu} = \cos\psi\cos\vartheta; \\ c_{31} &= a_{yw} = -\cos\vartheta\sin\gamma; \\ c_{32} &= a_{yv} = \cos\vartheta\cos\gamma; \\ c_{33} &= a_{yv}Y_{э0} - a_{yw}Z_{э0} + a_{yu}U_1; a_{yu} = \sin\vartheta. \end{aligned} \tag{2.38}$$

У співвідношеннях (2.38) наведені формули для елементів матриці обертання, де ψ, υ, γ– кути обертання відповідно навколо осей v, w, u. З урахуванням (2.37) співвідношення (2.17) приймають вигляд:

$$z = Z_0 - Y_0 \frac{c_{11}z_э - c_{12}y_э + c_{13}}{c_{31}z_э - c_{32}y_э + c_{33}},$$
$$x = X_0 - Y_0 \frac{c_{21}z_э - c_{22}y_э + c_{23}}{c_{31}z_э - c_{32}y_э + c_{33}}, \quad (2.39)$$

Отримані співвідношення (2.39) являють собою формули центропроективного перетворення, записані в формі, яка враховує специфіку формування зображення в растровій графіці.

Такий запис дозволяє:

- отримати наочні і природньо випливаючи із суті завдання обмеження по чіткості обчислень;
- побудувати ефективну схему обчислень з мінімальним набором арифметичних операцій, що виключає операції множення і ділення.

2.3 Опис земної поверхні в системі візуалізації фінітними функциями

Нехай на площині (X, Z) задана фіксована прямокутна сітка з одиничним кроком. Вважаємо, що земна поверхня задана у вигляді цифрової карти висот матрицею $H = [h_{ij}]$, де h_{ij} – висоти в вузлах (i, j) цієї сітки. Необхідно за цими параметрами знайти висоту $y(x, z)$ в будь-якій точці (x, z) синтезуємої поверхні[9].

Уявімо $y(x, z)$ у вигляді суми фінітних функцій:

$$y(x, z) = \sum_{i,j} h_{ij} f(x - i, z - j), \quad (2.40)$$

де $f(x,z)$- фінітна функція двох змінних, яку запишемо у вигляді твору:

$$f(x,z) = f(x) \cdot f(z), \tag{2.41}$$

де $f(t)$- фінітна функція однієї змінної.

Вимагатимемо, щоб вона мала безперервну першу похідну і мала властивості:

$$f(0) = 1, \tag{2.42}$$

$$f(t) = f(-t) \text{- властивість парності,} \tag{2.43}$$

$$f(t) = 0, \forall t : 1 \le |t| < \infty \text{ – властивість фінітності,} \tag{2.44}$$

$$f(t) + f(t-1) = 1, \forall t : t \in [0,1], \tag{2.45}$$

$$f'(0) = f'(-1) = f'(1) = 0. \tag{2.46}$$

До таких засобів можуть бути:

$$f(t) = \begin{cases} (\cos(\pi t)+1)/2, \text{при } |t| < 1 \\ 0, \text{при } |t| \ge 1 \end{cases},$$

$$f(t) = \begin{cases} 1-2t^2, \text{при } |t| < 0.5 \\ 2(1-|t|)^2, \text{при } 0.5 \le |t| < 1, \\ 0, \text{при } |t| \ge 1 \end{cases}$$

$$f(t) = \begin{cases} 2^{-(2t)^2}, \text{при } |t| < 0.5 \\ 1-2^{-(2-2|t|)^2}, \text{при } 0.5 \le |t| < 1 \\ 0, \text{при } |t| \ge 1 \end{cases}$$

З огляду на фінітність $f(x,z)$ під знаком підсумовування в (2.40) можна залишити 4 доданки:

$$y(x,z) = h_1 f(x_д, z_д) + h_2 f(x_д, z_д - 1) + h_3 f(x_д - 1, z_д) + h_4 f(x_д - 1, z_д - 1), \quad (2.47)$$

де:

$$h_1 = h(x_ц, z_ц),\ h_2 = h(x_ц, z_ц + 1),\ h_3 = h(x_ц + 1, z_ц),\ h_4 = h(x_ц + 1, z_ц + 1). \quad (2.48)$$

Тут $x_ц, z_ц$ – ціла частина, а $x_д, z_д$ – дрібна частина координат:

$$x_ц = \lfloor x \rfloor,\ x_д = x - x_ц,\ z_ц = \lfloor z \rfloor,\ z_д = z - z_ц, \quad (2.49)$$

де $\lfloor x \rfloor$ означає найбільше ціле, яке не перевищує x.

Підставимо (2.33) в (2.35) і, з огляду на, що $x_д \in [0,1), z_д \in [0,1)$, Використовуємо (2.32):

$$\begin{aligned} y(x,z) &= h_1 f(x_д) f(z_д) + h_2 f(x_д)(1 - f(z_д)) + \\ &+ h_3 (1 - f(x_д)) f(z_д) + h_4 (1 - f(x_д))(1 - f(z_д)). \end{aligned} \quad (2.50)$$

Остаточно представимо $y(x,z)$ в наступному вигляді, при якому кількість множень зменшується з восьми до трьох:

$$y(x,z) = B + (A - B) f(x_д), \quad (2.51)$$

де:

$$A = h_2 + (h_1 - h_2) f(z_д),\ B = h_4 + (h_3 - h_4) f(z_д). \quad (2.52)$$

Таким чином, земна поверхня може бути інтерпольована за допомогою виразів (2.48), (2.49), (2.51, 2.52), які визначають алгоритм обчислення висоти y в будь-якій точці (x, z).

Для візуалізації поверхні потрібно значення освітленості в даній точці. Для цього потрібно обчислювати нормаль до будь-якої точки поверхні.

Приведемо (2.48) до виду $F(x, y, z) = 0$ і визначимо нормаль як градієнт до функції F в точці (x, y, z):

$$\vec{N}(x,z) = \begin{bmatrix} \partial F/\partial x \\ \partial F/\partial y \\ \partial F/\partial z \end{bmatrix} = \begin{bmatrix} f'(x)\left[h_4 - h_2 - (h_1 - h_2 - h_3 + h_4)f(z)\right] \\ 1 \\ f'(z)\left[h_4 - h_3 - (h_1 - h_2 - h_3 + h_4)f(x)\right] \end{bmatrix}.$$

Синтез поверхні, описаної запропонованим способом, виконаний зворотнім трасуванням, з використанням ітераційного методу.

Представлений метод опису земної поверхні має такі особливості:

- гладка інтерполяція (немає зламів на стиках суміжних поверхонь);
- метод не вимагає попередніх обчислень. Обчислення йде безпосередньо по висотам;
- універсальність методу. Як фінітної функції $f(t)$ при розрахунку висоти і нормалі може виступати будь-яка функція, яка задовольнить властивостями, що не змінює алгоритм обробки;
- у базі даних зберігається мінімум інформації, що дозволяє компактно зберігати і швидко обробляти великі простори земної поверхні;
- мала обчислювальна складність співвідношень.

2.4. Методи деформації фінітних функцій у задачах синтезу зображення для систем візуалізації

Однією з дуже складних завдань при формуванні зображень в системах візуалізації реального часу є відображення рельєфу земної поверхні на

великих просторах. Необхідність побудови таких систем візуалізації виникає при розробці, наприклад, авіаційних або космічних тренажерів. При використанні тріангуляції необхідна підготовка даних по рельєфу. В разі попередньої підготовки та проміжного зберігання бази даних великих просторів земної поверхні потрібне збільшення обсягу пам'яті. У разі підготовки цих даних на етапі розрахунку кадру збільшується час рахунку.[10]

В даний час активно формується база даних по рельєфу земної поверхні у вигляді карт висот. Ці карти, типу Digital Elevation Model, представляють собою набір висот над базовою площиною земної поверхні, розташованих у вузлах прямокутної сітки. Основна перевага такого підходу при вирішенні даного завдання полягає в невеликому обсязі обчислень без необхідності попередньої обробки (підготовки) та проміжного зберігання бази даних, тобто в процесі «польоту» на тренажері.

Фінітні функції задовольняють наступним умовам:

$$f(0)=1, \tag{2.53}$$

$$f(x)=f(-x)-\text{властивість парності}, \tag{2.54}$$

$$f(t)=0,\forall t:1\le|t|<\infty-\text{властивість фінітності}, \tag{2.55}$$

$$f(x)+f(x-1)=1,\forall x:x\in[0,1], \tag{2.56}$$

$$f'(0)=f'(-1)=f'(1)=0. \tag{2.57}$$

При інтерполяції за допомогою даного класу фінітних функцій, наприклад, одновимірних функцій на довгих ділянках монотонних змін утворюються «сходинки» (рис. 2.3). Ці сходинки при інтерполяції двовимірних функцій утворюють спотворення з регулярною структурою «сходинки» (рис.2.3), що призводить до зниження реалістичності синтезованого зображення. У зв'язку з цим виникає необхідність подальшого дослідження можливостей фінітних функцій та розширення їх класу з метою

підвищення реалістичності відображення рельєфу земної поверхні, а також інших природних та штучних об'єктів.

Методи деформації.

Є різні варіанти деформації фінітних функцій для апроксимації функцій однієї та двох змінних. Позначимо деформовану функцію через f^k, де k – номер варіанта.

Апроксимація функції однієї змінної. Нехай відомі значення функції в рівновіддалених вузлах: $y_i = y(i)$, $i = 0,...,n$. Наближающую функцію зазвичай задають у вигляді зваженої суми фінітних функцій:

$$\tilde{y}(x) = \sum_{i=0}^{n} y_i f^k(x - i), \tag{2.58}$$

Варіант 1.

Для зменшення спотворень типу «тераси» будемо апроксимувати нахил в даній точці, враховуючи внесок сусідніх висот. Це досягається шляхом розширення носія $supp(f)$ фінітної функції $f(x)$. Але зі збільшенням носія збільшується обсяг обчислень. Компромісом можна вважати $supp(f^1) = (-3/2, 3/2)$. Таким чином, пропонується функція $f(x)$, Що задовольняє умовам (2.57), деформувати наступним чином:

$$f^1(x) = 2/3 \cdot f(2/3 \cdot x). \tag{2.59}$$

При цьому множник 2/3 при x розширює носій фінітної функції, а множник при функції дозволяє домогтися умови $f^1(x-1) + f^1(x) + f^1(x+1) = 1, \forall x : x \in [-1/2, 1/2]$. Дана умова звужує клас функцій $f(x)$, Використованих в (2.57).

Варіант 2.

Функцію $f^2(x)$ визначимо як зважене середнє значення деяких двох фінітних функцій $f_1(x)$ і $f_2(x)$:

$$f^2(x) = f_1(x) \cdot a + f_2(x) \cdot (1 - a), \tag{2.60}$$

при цьому

$$\text{supp}(f^2) = (-1,1).$$

Для спрощення обчислень приведимо вираз $f^2(x)$ до виду:

$$f^2(x) = f_2(x) + a \cdot (f_1(x) - f_2(x)), \tag{2.61}$$

де $0 < a < 1-$ ваговий коефіцієнт. Раціонально цей коефіцієнт вибрати у вигляді $a = 2^{-d}, d = 0,1,2,3,\ldots$, Що спрощує апаратну реалізацію обчислення (2.61).

Нижче на рисунку 2.3 показані спотворення – «сходинки», які виникли при апроксимації по відрахункам за допомогою фінітної функції класу (2.56) виду:

$$f(x) = \begin{cases} (\cos(\pi x) + 1)/2, \text{при} |x| < 1 \\ 0, \text{при} |x| \geq 1 \end{cases}. \tag{2.62}$$

На рисунку 2.4 апроксимація виконана з використанням деформації функції (2.57) відповідно до (2.55). На рис. 2.5 апроксимація виконана відповідно до (2.56), причому $a = 0.5$, $f_1(x)$ відповідає (2.57), а

$f_2(x) = \begin{cases} 1 - |x|, \text{при} |x| < 1 \\ 0, \text{при} |x| \geq 1 \end{cases}$. Як видно з рис. 2.4 та рисунку 2.5, при використанні деформованих фінітних функцій «сходинки» зменшуються.

Рисунок 2.3 – Спотвореня

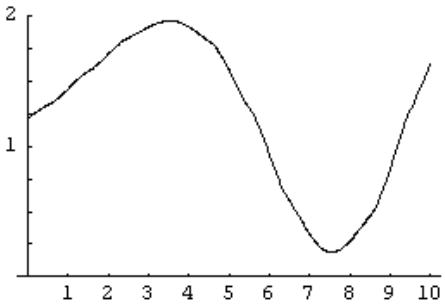

Рисунок 2.4 – Варіант 1

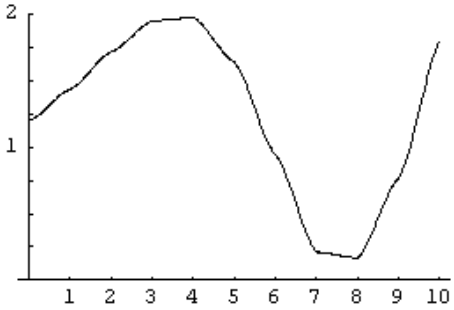

Рисунок 2.5 – Варіант 2

Апроксимація функції двох змінних. Нехай відомі значення функції в рівновіддалених вузлах $y_{ij} = y(i, j)$, $i = 0,...,n;\ j = 0,...,m$. Наближаючою функцію зазвичай задають у вигляді зваженої суми фінітних функцій:

$$\tilde{y}(x,z) = \sum_{i=0}^{n}\sum_{j=0}^{m} y_{ij} f^{k}(x-i, z-j) = \sum_{i=0}^{n}\sum_{j=0}^{m} y_{ij} f^{k}(x-i)\cdot f^{k}(z-j), \qquad (2.63)$$

При розгляді варіантів 1 і 2 залишимо в (2.63) під знаком підсумовування тільки ті функції, які не рівні нулю в точці (x,z).

Варіант 1.

$$\tilde{y}(x,z) = \sum_{i=\bar{x}-1}^{\bar{x}+1}\sum_{j=\bar{z}-1}^{\bar{z}+1} y_{ij}\cdot f^{1}(x-i)\cdot f^{1}(z-j) = F^{1}(x_p)\hat{Y}^{1}\left(F^{1}(z_p)\right)^{T},$$

де $\hat{x} = \lfloor x + \frac{1}{2} \rfloor$, $\hat{z} = \lfloor z + \frac{1}{2} \rfloor$, $x_p = x - \hat{x}$, $z_p = z - \hat{z}$, $x_p, z_p \in \left[-\frac{1}{2}, \frac{1}{2}\right)$,

$F^{1}(t) = \left(f^{1}(t+1) \quad f^{1}(t) \quad f^{1}(t-1)\right)$ – вектор, $\hat{Y}^{1} = \begin{pmatrix} y_{\bar{x}-1,\bar{z}-1} & y_{\bar{x}-1,\bar{z}} & y_{\bar{x}-1,\bar{z}+1} \\ y_{\bar{x},\bar{z}-1} & y_{\bar{x},\bar{z}} & y_{\bar{x},\bar{z}+1} \\ y_{\bar{x}+1,\bar{z}-1} & y_{\bar{x}+1,\bar{z}} & y_{\bar{x}+1,\bar{z}+1} \end{pmatrix}$ - матриця.

Варіант 2.

$$\tilde{y}(x,z) = \sum_{i=x_ц}^{x_ц+1}\sum_{j=z_ц}^{z_ц+1} y_{ij}\cdot f^{2}(x-i)\cdot f^{2}(z-j) = F^{2}(x_д)\hat{Y}^{2}\left(F^{2}(z_д)\right)^{T},$$

де $x_ц = \lfloor x \rfloor$, $z_ц = \lfloor z \rfloor$, $x_д = x - x_ц$, $z_д = z - z_ц$, $x_д, z_д \in [0,1)$

$$F^{2}(t) = \left(f^{2}(t) \quad f^{2}(t-1)\right),$$

$$\hat{Y}^{2} = \begin{pmatrix} y_{x_ц,z_ц} & y_{x_ц,z_ц+1} \\ y_{x_ц+1,z_ц} & y_{x_ц+1,z_ц+1} \end{pmatrix}.$$

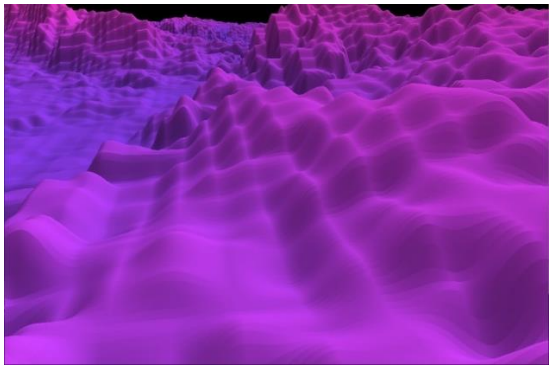

Рисунок 2.6 – Спотворення

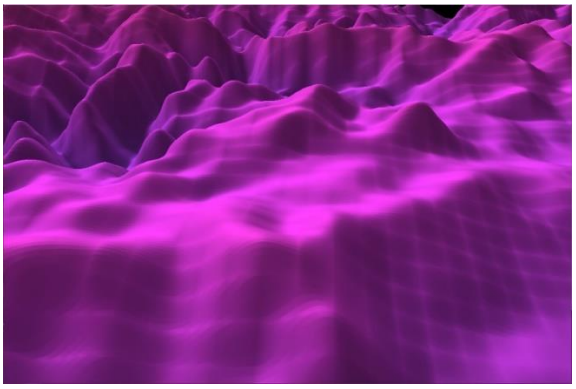

Рисунок 2.7 – Варіант 1

На рисуснку 2.6 показані спотворення типу «сходинки», які виникли при апроксимації двовимірним варіантом функції (2.57). На рисунку 2.7 представлено варіант 1. Розглянуті методи деформації фінітних функцій для опису рельєфу мають такі особливості:

- з'являється можливість корекції спотворень типу «сходинки»;
- деформовані фінітні функції володіють тим же набором позитивних властивостей.

3 МОДЕЛЮВАННЯ РЕЛЬЄФУ В СИСТЕМАХ ВІЗУАЛІЗАЦІЇ

3.1 Опис моделі рельєфа

У процесі дослідження була розроблена модель рельєфу з використанням фінітних функцій та метода зворотнього трасування, рисунок 3.1.

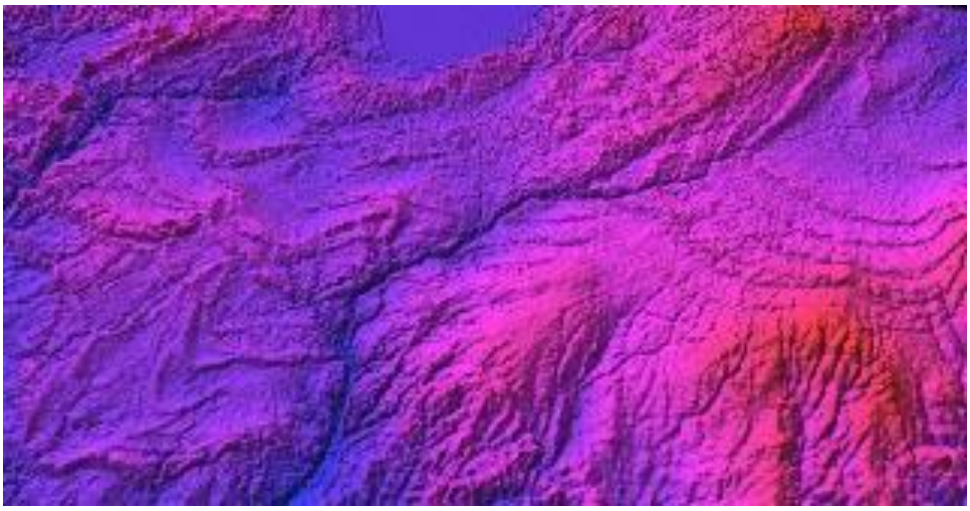

Рисунок 3.1 – Модель рельєфу

Модель рельєфу завантажується у програму у вигляді карти висот в форматі PNG-картинки. Після того як карта висот завантажена, програма почне завантажувати і компілювати GLSL-шейдерри. Для кожного променя итеративно відбувається переміщення точки, на яку він прикладений уздовж його направлення на відстань *dt*. *dt* залежить від положення камери над картою висот. Коли промінь досягає карти висот, відбувається розрахунок інтерпольованого за допомогою значення фінітних функцій. Якщо поточна позиція точки променя знаходиться під картою, то це зараховується як перетин. Залежно від розташування по осі *y* (вісь висоти) точки перетину обирається колір від синього до червоного. Після цього відбувається обчислення нормалі поверхні. Для цього проводиться 4 обчислення для

отримання висоти в сусідніх від точки перетину точках. Для цього для осей *x* і *y* додається і віднімається значення NORMAL_EPSILON (рівне 0.004) і знаходиться різниця між ними, що дає приблизний градієнт для двомірної фігури, яка є перетином карти висот і паралельної їй площини, що знаходиться на тому ж *y*, що і точка перетину.

3.2 Принцип роботи програмного забезпечення

Программа реалізована на язиці програмування C++ та GLSL. На рисунку 3.2 зображено блок схему алгоритму програми.

Рисунок 3.2 – Блок схема програми

Код C++. Программа після запуска завантажує PNG-картинку в структуру HeightMap_t, на рисунку 3.3 зображено карту висот.

```
typedef struct {
s32 w,h;
float* heights;
} HeightMap_t;
```

Приклад 3.1 – Завантаження картинки (Файл main.cpp)

Для завантаження використовується функція HeightMap_t LoadHeightMap(std::string path) яка завантажує PNG-картинку з path и повертає карту висот, де значення висоти від 0.0 до 1.0 виходить шляхом нормалізації червоної компоненти пікселя (ділеться на 255.0).

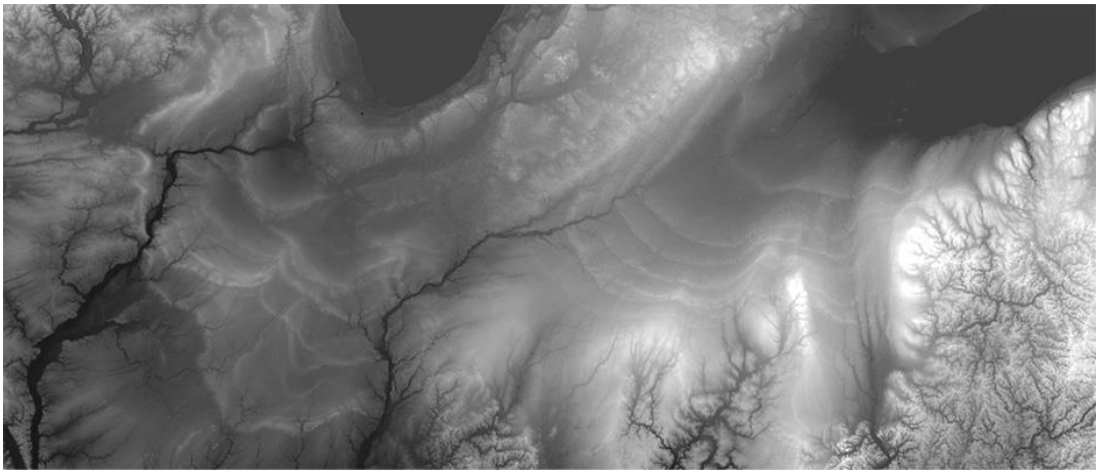

Рисунок 3.3 – Карта висот

Потім в нескінченному циклі викликаються функції draw, upr та деякі допоміжні, які керують усім процесом програми. Функція draw активує шейдерну програму і пересилає в неї GLSL uniforms, приклад 1.2.

```
mat4 camera
mat4 camera_notrans
vec3 camera_pos
Sampler2D HeightMap
int HeightMapWidth
int HeightMapHeight
```

Приклад 3.2 – Активація шейдерної програми (Файл fhm.dlsl)

Після чого, малює прямокутник на весь екран, в якому для кожного пікселя виповнюється функція main з GLSL-коду fhm.glsl. У функції upr відбувається обробка подій SDL2, де від подій миші (переміщення + натискання клавіш + коліщатка) переміщається камера. Камера – це матриця 4x4, яка переміщається шляхом множення її на кілька трансформаційних матриц. Спочатку на матрицю трансляції (translate), де вона переміщається по осі Z на змінну mode1r, що позначає відстань від деякої віртуальної точки, приклад 3.3.

```
camera = TranslateMatrix(camera,0,0,mode1r)
```

Приклад 3.3 – Матриця трасляції (Файл main.cpp)

Потім вона обертається навколо осі X на змінну angle_pitch, яка обчислюється за допомогою y-значення курсора миші на екрані, приклад 1.4.

```
camera = RotateMatrixX(camera,DegToRad(angle_pitch))
```

Приклад 3.4 – Обертання по осі X (Файл main.cpp)

Далі так само, але навколо осі Y на змінну angle_yaw, яка обчислюється за допомогою x-значення курсора миші, приклад 3.5.

```
camera = RotateMatrixY(camera,DegToRad(angle_yaw))
```

Приклад 3.5 – Обертання по осі Y (Файл main.cpp)

Та в кінці транслюється на нинішню позицію віртуальної точки, приклад 3.6.

```
camera = TranslateMatrix(camera,mode1pos.x,mode1pos.y,mode1pos.z);
```

Приклад 3.6 – Транслювання матриці (Файл main.cpp)

GLSL код. Функція main GLSL-коду це по функція зворотного трасування для окремого пікселя на екрані. В ній створюються змінні x і y, які дорівнюють нинішньому положення пікселя на екрані, приклад 3.7.

```
float x = gl_FragCoord.x;
float y = gl_FragCoord.y;
```

Приклад 3.7 – Нинішні положення пікселя (Файл main.cpp)

Після чого створюється сам промінь, який буде пересланий в функцію RaytraceObjects. Для нього обчислюються тривимірні вектора pos і dir, приклад 3.8.

```
vec3 pos = (camera*vec4(0.0,0.0,0.0,1.0)).xyz;
vec3 dir = (camera_notrans*normalize(vec4((x -
WIDTH/2)/WIDTH*tan(FOV/2)*WIDTH/HEIGHT,(y -
HEIGHT/2)/HEIGHT*tan(FOV/2),-1.0,1.0))).xyz;
```

Приклад 3.8 – Обчислення тривимірного вектора (Файл main.cpp)

Матриця камери pos помножена на 4-вимірної вектор, в якому x, y, z рівні 0, а w = 1.0. І від нього взято тільки xyz-компоненти. А dir обчислюється в залежності від FOV, ширини вікна і висоти вікна. На рисунку 3.4 зображена рейтресінг камера. Після чого вони передаються в функцію, приклад 3.9.

```
vec3 RaytraceObjects(vec3 pos,vec3 dir)
```

Приклад 3.9 – Функція рейтресінга (Файл fhm.glsl)

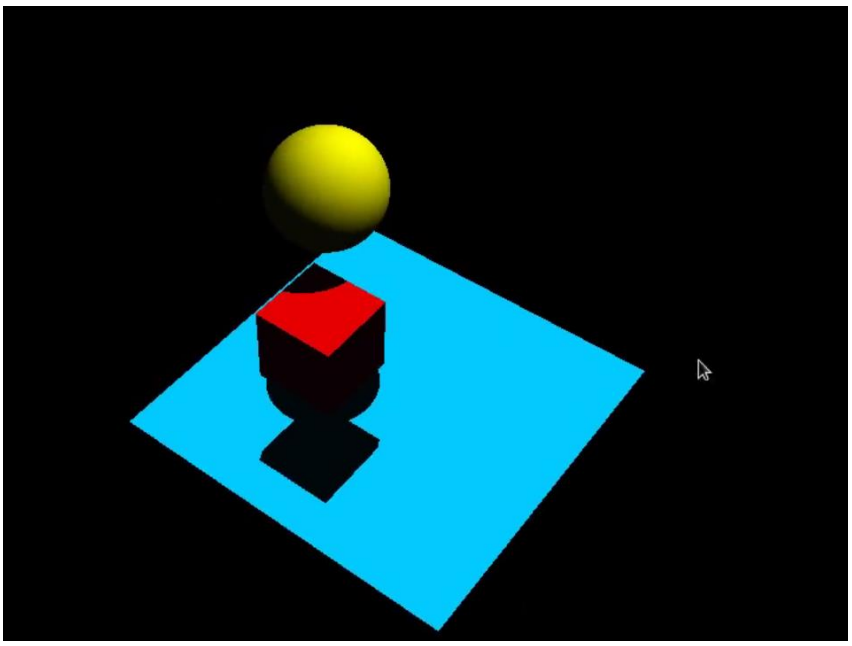

Рисунок 3.4 – Рейтресінг камера

Вона повертає тривимірний вектор, який містить RGB-значення кольору. Там відбувається перевірка, перетинає промінь карту висот і якщо перетинає, то отримати точку перетину і нормаль. Для цього потрібна функція отримання висоти за координатами, на рисунку 3.5 зображено модель рельєфу без деформаціїї рельєфу, приклад 3.10.

```
float HeightByCoord(float x,float z) {
float h = 0;
int sx = int(x + 0.5);
int sz = int(z + 0.5);
for (int ii = -1; ii <= 1; ii++) {
      for (int jj = -1; jj <= 1; jj++) {
            int i = sx + ii;
            int j = sz + jj;
            float                              tuth                              =
texture(HeightMap,vec2(float(i)/HeightMapWidth,float(j)/HeightMapHeight)
).r*255.0*SCALE;
            h += tuth*finit_cos_deform(x - i)*finit_cos_deform(z - j);
      }
}
return h;
}
```

Приклад 3.10 – Функція отримання координат (Файл fhm.glsl)

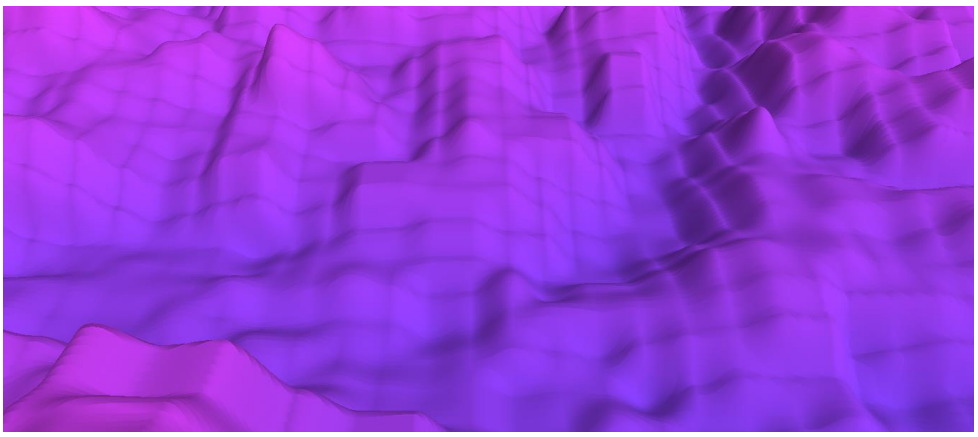

Рисунок 3.5 – Модель рельєфу з деформацією

Вона використовує GLSL-ву функцію texture для отримання значення з Sampler2D HeightMap, який був пересланий як uniform, через деякі координати. Вона отримує 9 сусідніх значень і за допомогою деформованої фінітної функції отримує згладжену висоту, рисунок 3.6.

Рисунок 3.6 – Модель рельєфу без деформацій

На прикладі 3.11 показан звичайна фінітна функція.

```
float finit_cos(float t) {
return (cos(M_PI*t) + 1.0)/2.0*int(t < 1.0);
}
```

Приклад 3.11 – Фінітна функція (Файл fhm.glsl)

На прикладі 3.12 показана деформована фінітна функція.

```
float finit_cos_deform(float t) {
return 2.0/3.0*finit_cos(t*2.0/3.0);
}
```

Приклад 3.12 – Деформована фінітна функція (Файл fhm.glsl)

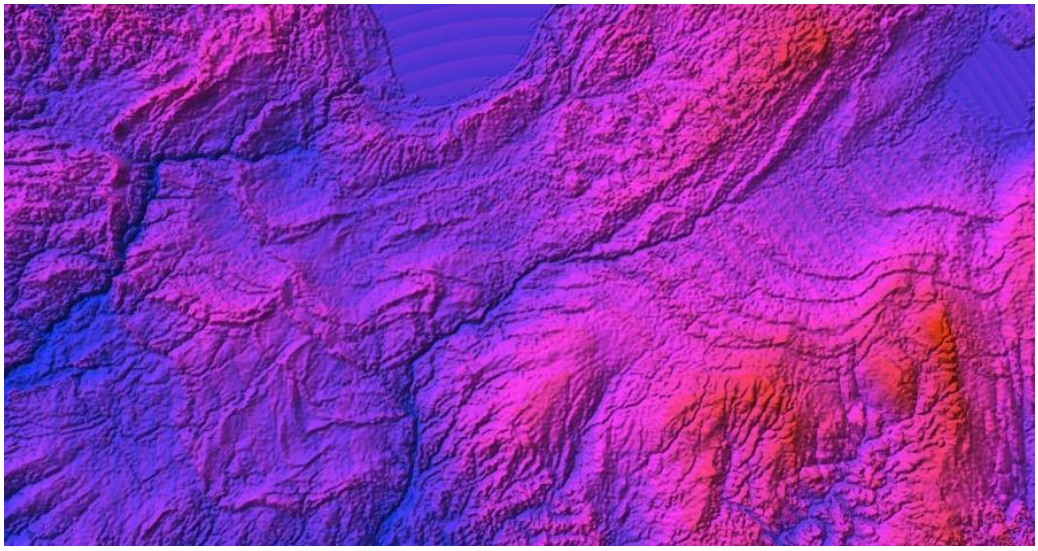

Рисунок 3.7 – Модель рельєфу

Побудована модель на основі метода зворотнього трасування та з використанням фінітних функцій, рисунок 3.7, має наступні особливості:

- гладка інтерполяція;
- обчислення цде безпосередньо по висотам;
- у базі даних зберігається мінімум інформації, що дозволяє компактно зберігати і швидко обробляти великі простори земної поверхні.

ВИСНОВКИ

В ході виконання атестаційної була побудована модель рельєфу в системах візуалізації. Досліджено що для синтезу зображень в реальному часі добре опрацьований метод прямого трасування. Докладно розглянутий метод зворотного трасування що дозволяє синтезувати зображення з високою реалістичністю. Розглядено існуюча практика синтезу зображення земної поверхні на основі різних методів. Досліджено рельєф в системах візуалізації для авіаційних тренажерів орієнтованих на метод зворотного трасування та за допомогою фінітних фукнцій. Модель показала що немає зламів на стиках суміжних поверхонь, побудова не вимагає попередніх обчислень, у базі даних зберігається мінімум інформації, що дозволяє компактно зберігати і швидко обробляти великі простори земної поверхні.

ПЕРЛІК ДЖЕРЕЛ ПОСИЛАННЯ

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

2. Трехмерная компьютерная графика. [Текст] / Иванов В.П., Батраков А.С. – М. : Радио и связь, 1995. – 222 с.

3. History Timeline [Electronic resource] // btxproject, – 2015, – Режим доступа: https://sites.google.com/site/btxprojectcomputergraphics/history-t

4. What is 3d graphics? [Electronic resource] // rendering blog, – 2015, – Режим доступа: https://vrender.com/what-is-3d-graphis/

5. Basic Applications of Computer [Electronic resource] // rendering blog, – 2016, – Режим доступа: https://vrender.com/what-is-3d-grapics/

6. Что такое трассировка лучей и нужна ли она нам в играх? [Електроний ресурс] // – 2017, – Режим доступа: https://keddr.com/2018/03/chto-takoe-trassirovka-luchey-i-nuza-li-ona-nam-v-igrh/

7. Алгоритм геометричних перетворень зображення в системах візуалізації тренажерів транспортних засобів [Текст] / Гусятин В.М. – М. : Авіаційно-космічна техніка і технологія. Праці ХАІ ім. Н.Є. Жуковського, 1997, – 471.

8. Математическая модель геометрической обработки изображения на плоскости для растровых систем визуализации [Текст] / В. М. Гусятин

9. Описание земной поверхности в системах визуализации [Текст]: довідник / Гусятин В.М., Чаговец Я. В. – К: Системний аналіз, управління і інформаційні технології: Вісник ХДПУ, 2000, – 47.

10. Методы деформации финитных функций в задачах синтеза изображений для систем визуализации [Текст] / В. М. Гусятин, Я. В. Чаговец., – М: Библиогр., –28.

11. Підхід до візуалізації тріангулірованних поверхонь методом зворотного трасування променів. [Текст] / В.М. Гусятін, А.Е. Громенко.

12. Моделювання рельєфу в системах візуалізації для авіаційних тренажерів. [Текст] / Т.О. Остапенко, – М: Інформаційне сусупільство: технологічні, економічні, та технічн