НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
"___" _____2023_ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою

«Системне програмування та спеціалізовані комп'ютерні системи»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Алгоритмічне та програмне забезпечення для відтворення тривимірної графіки
_____
_____

Виконав:

студент IV курсу, групи КВ-91

Гончар Михайло Віталійович_____ _____
(підпис)

Керівник: старший викладач к.т.н. Наливайчук М. В. _____ _____
(підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(підпис)

Рецензент к. т. н., доцент каф. І та ПМ ФІОТ Олійник Ю. О. ____ _____
(підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2023 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма

«Системне програмування та спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____20___ р.

ЗАВДАННЯ
на дипломний проєкт студента

Гончара Михайла Віталійовича _____
(прізвище, ім'я, по батькові)

1. Тема проєкту: Алгоритмічне та програмне забезпечення для відтворення тривимірної графіки,
керівник проєкту старший викладач, к. т. н. Наливайчук М. В.

затверджені наказом по університету від «31» травня 2023 р. № 2107-с

2. Термін подання студентом проєкту 13 червня 2023 р.

3. Вихідні дані до проєкту: технічна документація, теоретичні дані

4. Зміст пояснювальної записки:

- сучасний стан розвитку 3d-моделювання;
- основні алгоритми створення 3d-об'єктів;
- розробка алгоритмів для зображення 3d моделі;
- висновки.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо):

– Процедура зчитування полігонів

– Алгоритм класичного RayTracing

– Алгоритм оптимізованого RayTracing

– Представлення сцени

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	к.т.н. доцент Клятченко Я. М.		

8. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
	Видача завдання на дипломне проектування	04.11.2022	
	Вивчення літератури за тематикою роботи	12.11.2022	
	Розроблення та узгодження технічного завдання	06.02.2023	
	Розроблення структури додатку	12.02.2023	
	Розроблення дизайну та графічних елементів	14.04.2023	
	Програмна реалізація додатку	06.05.2023	
	Тестування додатку	10.05.2023	
	Підготовка матеріалів текстової частини проекту	11.05.2023	
	Підготовка матеріалів графічної частини проекту	20.05.2023	
	Оформлення технічної документації проекту	28.05.2023	

Студент _____ Михайло ГОНЧАР
(підпис) (Ім'я та ПРІЗВИЩЕ)

Керівник проєкту _____ Микола НАЛИВАЙЧУК
(підпис) (Ім'я та ПРІЗВИЩЕ)

* Консультантом не може бути зазначено керівника дипломного проєкту.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку об’ємом в 65 сторінок.

Об’єкт розробки – розробка та оптимізація алгоритмів представлення тривимірної графіки.

Оптимізація алгоритму дозволяє: досягати зображення тривимірного об’єкту за меншу кількість часу; заощаджувати роботу процесору системи; заощаджувати на роботі операційної пам’яті системи; регулювати якість зображення. В процесі розробки були використані такі алгоритми та допоміжні технології зображення тривимірної графіки, як Ray casting, Scanline rendering, NF, Ray marching.

В ході розробки:

- проведено аналіз існуючих методів комп’ютерних систем зображення тривимірного об’єкту;
- сформульовані вимоги до комп’ютерної системи зображення тривимірного об’єкту;
- розроблена структура програмного забезпечення системи зображення тривимірного об’єкту;
- розроблено користувацький додаток для управління;
- створено відповідні виміри

Оптимізація алгоритмів дозволить зменшити вартість машинного забезпечення, необхідного для зображення 3D-об’єктів, що дозволить зменшити витрати для його рендерингу, що в свою чергу збільшить продуктивність системи, яка займається обробкою подібних процесів.

Ключові слова:
КОМП’ЮТЕРНА ГРАФІКА, 3D, ТРИВИМІРНА ГРАФІКА, ОПТИМІЗАЦІЯ, АЛГОРИТМ

ABSTRACT

The qualification work includes an explanatory note of 65 pages.

The object of development is the development and optimization of algorithms for the presentation of three-dimensional graphics.

Optimization of the algorithm allows: to achieve the image of a three-dimensional object in less time; save the work of the system processor; save on system operating memory; adjust image quality. In the development process, such algorithms and auxiliary technologies of three-dimensional graphics were used, such as Ray casting, Scanline rendering, NF, Ray marching.

In the course of development:

- the analysis of existing methods of computer systems for the image of a three-dimensional object was carried out;
- formulated requirements for the computer system of the image of a three-dimensional object;
- the software structure of the three-dimensional object image system was developed;
- developed a custom application for management;
- appropriate dimensions have been created;

Optimizing of algorithms will allow to reduce the cost of hardware necessary for the image of 3D objects, which will allow to reduce costs for its rendering, which in turn will increase the performance of the system that handles such processes.

Keywords:

COMPUTER GRAPHICS, 3D, THREE-DIMENSIONAL GRAPHICS, OPTIMIZATION, ALGORITHM

Формат	Зона	Поз.	Позначення	Найменування	Кільк. арк.	Прим.
				Документація загальна		
				розроблена		
А4			ІАЛЦ.466538.002 ТЗ	Алгоритмічне та	6	
				програмне забезпечення		
				для відтворення		
				тривимірної графіки		
				Технічне завдання.		
А4			ІАЛЦ.466538.003 ТП	Алгоритмічне та	1	
				програмне забезпечення		
				для відтворення		
				тривимірної графіки		
				Відомість технічного		
				проекту		
А4			ІАЛЦ.466538.004 ПЗ	Алгоритмічне та	53	
				програмне забезпечення		
				для відтворення		
				тривимірної графіки		
				Пояснювальна записка		
А1			ІАЛЦ.466538.005 Д1	Алгоритм класичного	1	
				RayTracing		
				Схема алгоритму.		
А1			ІАЛЦ.466538.006 Д2	Алгоритм оптимізованого	1	
				RayTracing		
				Схема алгоритму.		
А1			ІАЛЦ.466538.007 Д3	Процес зчитування	1	
				полігонів		
				Схема алгоритму.		
А1			ІАЛЦ.466538.008 Д4	Представлення сцени	1	
				Схема структурна.		
			Диск CD-R	Матеріали роботи		

Зм.	Арк	№ докум.	Підпис	Дата	ІАЛЦ.466538.001ОА		
Розроб.		Гончар.			Алгоритмічне та програмне забезпечення для відтворення тривимірної графіки. Опис альбому.	Літ.	Арк. 1 / Арк-ів 1
Перевір.		Наливайчук М.В.					
Нор.кон.		Плахотний М.В.				НТУУ «КПІ», ФПМ КВ-91	
Затв.		Тарасенко В.П.					

Зміст

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Розробка та оптимізація алгоритмів представлення тривимірної графіки».

Галузь застосування: створення додатків та об’єктів з меншими витратами на машинні ресурси.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Оптимізація алгоритмів представлення тривимірної графіки може мати численні користі. Вона сприяє підвищенню продуктивності шляхом збільшення швидкості обробки тривимірних даних і зниження затримок при зображенні. Крім того, оптимізовані алгоритми дозволяють економити ресурси, такі як процесорний час, пам'ять та енергію, що особливо важливо при роботі на менш потужних пристроях. Також, оптимізовані алгоритми підтримують різні розміри проектів та дозволяють ефективно працювати з великими обсягами тривимірних даних, забезпечуючи масштабованість системи. Вони також підвищують точність обчислень і можуть знизити вартість розробки тривимірних графічних додатків.

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення оптимізованого алгоритму для представлення тривимірної графіки на основі вже існуючих.

4. ДЖЕРЕЛА РОЗРОБКИ

1) "Optimization Techniques for 3D Rendering Algorithms": John Smith, Sarah

Johnson, 2022

2) "Advanced Topics in 3D Graphics and Rendering": David Wilson, 2019

3) "Efficient 3D Algorithm Optimization Strategies": Emily Davis, 2020

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з будь якою операційною системою (IOS,Android,WP);
- можливість управління процесом обробки заявок;
- можливість управління заявками;
- можливість додавати нові заявки;
- можливість редагування заявок;
- наявність зручної системи сповіщень ініціатора та виконавця;
- наявність системи контакту з ініціатором;
- прив'язки файлів ініціатора та виконавця до заявок;
- зберігання заявок;
- аналітичні можливості.

5.2. Вимоги до апаратного забезпечення

- Процесор: 2,4-ядерний, MediaTek, Snapdragon, Kirin, Intel Atom;
- Оперативна пам’ять: 2 Гб;
- Наявність доступу до мережі Internet (GPRS, EDGE, 3G, 4G);

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows Phone 8, Android, iOS;

6. ЕТАПИ РОЗРОБКИ

/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
.	Вивчення літератури за тематикою проекту	15.04.2019
.	Розроблення та узгодження технічного завдання	30.04.2019
.	Аналіз існуючих рішень	05.05.2019
.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
.	Підготовка графічної частини дипломного проекту	20.05.2019
.	Оформлення документації дипломного проекту	25.05.2019
.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Формат	Зона	Поз.	Позначення	Найменування	Кільк. арк.	Прим.
				Документація загальна		
				розроблена		
А4			ІАЛЦ.466538.004 ПЗ	Алгоритмічне та	53	
				програмне забезпечення		
				для відтворення		
				тривимірної графіки		
				Пояснювальна записка		
А1			ІАЛЦ.466538.005 Д1	Алгоритм класичного	1	
				RayTracing		
				Схема алгоритму.		
А1			ІАЛЦ.466538.006 Д2	Алгоритм оптимізованого	1	
				RayTracing		
				Схема алгоритму.		
А1			ІАЛЦ.466538.007 Д3	Процес зчитування	1	
				полігонів		
				Схема алгоритму.		
А1			ІАЛЦ.466538.008 Д4	Представлення сцени	1	
				Схема структурна.		
			Диск CD-R	Матеріали роботи		

Зм.	Арк	№ докум.	Підпис	Дата	ІАЛЦ.466538.001ОА		
Розроб.		Гончар.			Алгоритмічне та програмне забезпечення для відтворення тривимірної графіки. Опис альбому.	Літ.	Арк. 1 / Арк-ів 1
Перевір.		Наливайчук М.В.					
Нор.кон.		Плахотний М.В.				НТУУ «КПІ», ФПМ	

Затв.	Тарасенко В.П.				КВ-91

ЗМІСТ

Додаток 1. Копії графічного матеріалу

ІАЛЦ.467200.005 Д1 Вхід в систему. Схема алгоритму.

ІАЛЦ.467200.006 Д2 Видалення страви. Схема алгоритму.

ІАЛЦ.467200.007 Д3 Резервація столика. Схема алгоритму.

ІАЛЦ.467200.008 Д4 Замовлення страви. Схема алгоритму.

Додаток 2. Фрагменти програмного коду

Додаток 3. Презентація

ВСТУП

На сьогодні 3D моделювання є широко розповсюдженим елементом найрізноманітніших сфер людської діяльності. Присутність тривимірної графіки можна віднайти фактично в будь-якій галузі, яка не обов'язково буде медійною. Часто 3D моделі виступають в ролі зручного інструменту, необхідного для візуалізації тих, чи інших даних.

Еволюція індустрій та розвиток технологічних рішень дозволяють та потребують осучаснення таких процедур, як 3D моделювання. Саме через це ми бачимо регулярну появу нових рішень в сфері тривимірної графіки, націлених на надання нових можливостей користувачам та розробникам. Наразі такі рішення дозволяють досягати неймовірних результатів зображення.

Актуальність дослідження.

За статистикою, основними галузями, що мають постійну необхідність в 3В моделюванні, є: кіноіндустрія, відео-ігрова індустрія та цифрове образотворче мистецтво. Також дане моделювання часто можна зустріти в якості допоміжного інструмента в проектуванні, наприклад в сфері архітектури, ландшафтного дизайну, медицини, тощо. При чому, враховуючи що попит на дані сфери діяльності активно зростає останнім часом, можна зробити відповідний висновок, що зростатиме попит і на відповідних фахівців та технології. Підсумовуючи, можна спостерігати наявну актуальність даної теми роботи.

Створення алгоритмів тривимірної графіки має підвищений рівень складності, так як дана сфера інженерії потребує регулярних іноваційних рішень. Ключовим спільним моментом таких алгоритмів є процес представлення наборів математичних даних у такий набір, що зможе бути перетворений у тривимірне зображення. Інсуюче програмне забезпечення, що використовує подібні алгоритми, дозволяє значно скоротити час, необхідний на створення картинки.

Представлене дослідження присвячене оптимізації та розробці алгоритмів представлення тривимірної графіки.

Мета дослідження. Алгоритми представлення тривимірної графіки та їхня оптимізація.

Об'єктом дослідження є оптимізація та розробка алгоритмів.

Предметом дослідження є алгоритмічні рішення, що дозволяють досягти оптимального розподілу ресурсів машини під час 3D моделювання.

Методи дослідження: проведення детального розгляду наявної технічної літератури та існуючих рішень. Основними методами дослідження були статистичне порівняння розроблених алгоритмів, аналіз їхніх переваг та недоліків. Методи оптимізації полягають у застосуванні математично та програмно легших операціях та арифметичній оптимізації алгоритму. Методи дослідження не суперечать наявним.

При проведенні наукових досліджень, залежно від поставлених завдань було використано наступні наукові методи: ситуаційний підхід, методи статистичного аналізу, порівняння, прогнозування, графічний та табличний.

Теоретична та практична цінність роботи полягає в зібранні в єдиний документ наявних методів оптимізації та незначній їхній доробці. Відповідний до роботи медіа-файл дозволяє наочно переконатися в ефективності тих, чи інших технічних рішень та візуально побачити майже усі наявні переваги та недоліки алгоритмів. В результаті забрані разом наявні та розроблені рішення виступають в ролі не без вагомого джерела інформації.

Новизна дослідження полягає в підсумуванні наявних технічних рішень в одному документі, додавання власних розробок до них, та присутність наочної візуалізації цих рішень.

Структура роботи. Відповідно до посібника, структура дипломного проекту містить вступ, три розділи, висновки, список використаних джерел та додатки.

РОЗДІЛ 1.
СУЧАСНИЙ СТАН РОЗВИТКУ 3D-МОДЕЛЮВАННЯ

1.1. Історія розвитку 3D-моделювання

Поняття комп'ютерної графіки виник у 60-х роках. Творцем його був Уільям Феттер, який працював на той момент над проектами у фірмі Boeing. У 1964 році Уільям створив першу в світі модель людського тіла, яку пізніше назвали «Людина Боїнга» (рис. 1.1). Дану модель використовували для розробки авіа лайнерів та для інших досліджень [1].

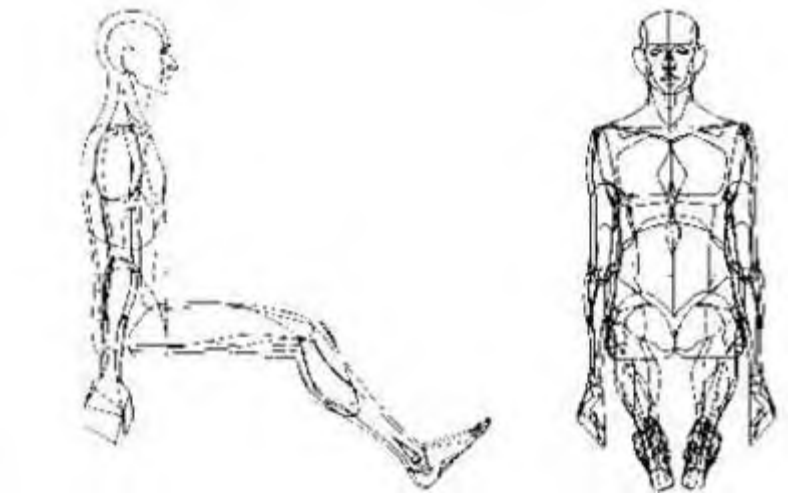


Рис. 1.1. Модель «Людина Боїнга»

Наступним вагомим кроком розвитку була подія, коли у той же час в Університеті Юти було відкрито першу в світі кафедру, присвячену дослідженню комп'ютерної графіки, яку очолювали з Айдан Сезерленд та Девід Еванс. Тривімірне проектування виконували, застосовуючи програмне забезпечення Sketchpad, яке по суті є аналогом сучасних 3D-редакторів. Дане програмне забезпечення дозволяло виконувати базові операції над тривимірних зображенням: створення, зображення та редагування. Користування програмою забезпечувалось аналогом сучасного стилуса – «світловим пером» (рис. 1.2).

Дана кафедра випустила чимало видатних фахівців, таких як Джим Блінн, Анрі Гуро (розробник алгоритму накладання текстури), БІ Тю Фонг (розробник алгоритму накладання тіней) та Ед Катмулл (майбутній тех-директор студії Pixar) [10].

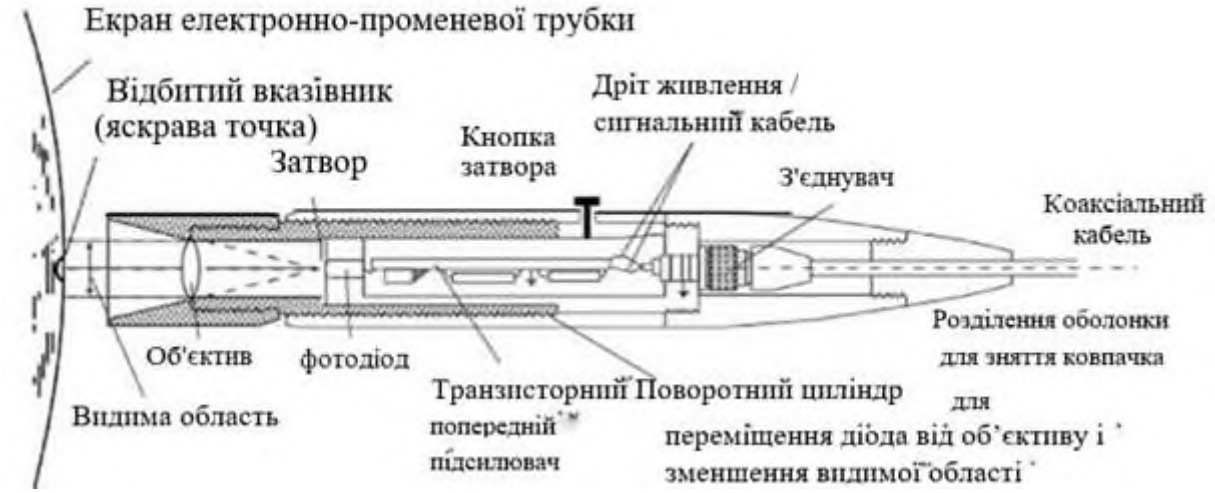


Рис. 1.2. Схема «світлого пера»

Активний попит 3D у кіноматографі з'явився після 1981 року. Комп'ютерну анімацію створили такі компанії, як Information International Inc (або Triple-I) і Mathematical Applications Group Inc (відома як MAGI/Synthavision). Дві інші компанії були найняті для роботи над деякими зі сценаріїв: Роберт Абель. & Associates і Digital Effects. MAGI розробила систему процедурного моделювання, що створює 3D-моделі шляхом поєднання геометричних фігур. Ця система моделювання є основою для програмного забезпечення Synthavision. Бібліотека програмного забезпечення містить 25 різних геометричних фігур, включаючи сфери, циліндри та піраміди. Для анімації моделей Synthavision використовує спеціальну «мову керування», яка дозволяє аніматору описувати шлях моделі, оцінювати результат на дисплеї та змінювати його, якщо необхідно.

Редактування руху тривимірного об’єкта в реальному часі надає можливість художникам MAGI ставити якісну анімовану каортинку в сценах, де наявна множина 3D моделей.

Процедура створення тривимірного заображення за допомогою ПЗ TRANEW виглядає так: ручне малювання 3D-об’єкту шляхом задання даних про полігони, перенесення за допомогою спеціальних оцифровуючих пристроїв на машину моделі Foonly F1. Програмне забезпечення TRANEW надавало та зображало текстуру до моделі, та освітлення. Створене зображення за якістю мало значні переваги над тими, що були створені ПЗ MAGI.

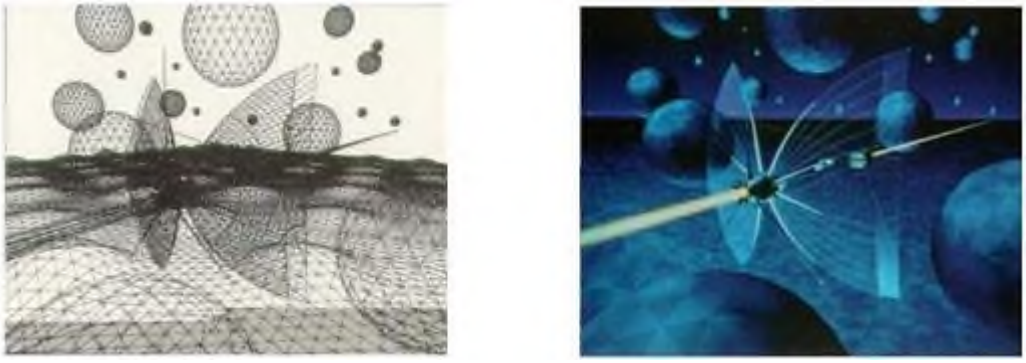


Рис. 1.3. Модель «Сонячний вітрильник» у ПЗ TRANEW

Черговий етап полягав в кольоруванні 3D об’єктів, оскільки ті після оцифровування мали вигляд монотонних фігур з контурною текстурою.

У середньому для зображення застосовували близько сотні обчислюваних машин корпорації Sun. Рендеринг одного кадру для всіх етапів займав близько від сорока п’яти хвилин до тридцяти годин.

1.2. Сучасний стан галузі 3D-моделювання

Технології тривимірного моделювання переживає активний розвивиток. Вони стали необхідними у множині галузей: кіноіндустрія, реклама, комп’ютерні ігри, дизайну.

На розвиток моделювання також впливають можливості апаратних і програмних ресурсів. За останнє десятиліття можливості технологій тільки зросли. Поява більш потужних комп'ютерів дозволила досягти складних

візуальних ефектів, підвищити якість анімаційних моделей, а також запустити процес рендерингу набагато швидше, ніж раніше.

Процес рендерингу тепер можна здійснювати набагато швидше, ніж раніше, а розробка нового програмного забезпечення і методів створення 3D-моделей і анімації дає можливість спростити роботу професіоналів без шкоди для якості готового продукту.

Трасування променів - це техніка створення зображень 3D-моделей шляхом відстеження зворотної траєкторії світлових променів у комп'ютерній програмі. Траєкторія світлових променів від освітлюваного об'єкта відстежується за допомогою спеціальних алгоритмів для створення симуляції його взаємодії з об'єктом, включаючи відбиття і заломлення. Фахівці просто задають джерело світла, його напрямки та відстані, регулюють температуру та інтенсивність світла, а все інше робить програма.

Затінення - це процес, що виконується за допомогою шейдерів, тобто програм, які використовуються в 3D-графіці для визначення кінцевих параметрів об'єкта або зображення. Шейдери можуть включати в себе як завгодно складні описи поглинання і розсіювання світла, накладання текстур, відбиття і заломлення, затемнення, зміщення поверхонь і ефекти пост-обробки. Програмовані шейдери є дуже ефективними і можуть візуалізувати складні поверхні за допомогою простих геометричних фігур [23].

Починаючи з введення графічної картки 3Dfx Voodoo 1 у 1996 році, стало можливим візуалізувати тривимірну графіку на настільних комп'ютерах. Раніше це завдання повністю покладалося на центральний процесор, який обчислював кінцеве зображення. Однак з того часу графічні прискорювачі, також відомі як графічні процесори, стрімко розвивалися в термінах продуктивності, функціональності та гнучкості. Це

призвело до того, що все більше завдань передається на GPU з метою підвищення загальної ефективності процесу візуалізації. На сьогоднішній день центральний процесор переважно використовується для підготовки даних для графічного процесора, що включає структурування та обробку даних моделі в оперативній пам'яті. Крім того, до цих моделей застосовуються фізичне моделювання та штучний інтелект [18].

Наприклад, розглянемо графічний прискорювач Geforce 2060 NVIDIA, який здатний до обчислення тривимірної графіки з наближенням до фотореалізму. На рисунку 1.2 наведено зразок зображення, створеного цим графічним процесором. Зображено фотореалістичне відображення моделі, реалізоване за допомогою графічного прискорювача Geforce 2080 NVIDIA.

1.3. Основні поняття тривимірної комп’ютерної графіки

Термін тривимірна графіка, або 3D-графіка зазвичай трактують, як розділ комп’ютерної графіки, що являє собою набір утиліт і методів, головною задачею яких є представлення об’ємних об’єктів. Частіше всього цей набір інструментів використовують для проектування об’єктів, які потім експортують в зображення, щоб потім зобразити на екрані чи інших пристроях зображення [8].

Зображення тривимірного об’єкту на площині має різницю від зображення тим, що має геометричну проекцію моделі (сцени) на площину використовуючи спеціалізовані програми. Модель, буває зображенням

як реальних об’єктів, так абстракцій (наприклад проекція фракталу).

Кроки, які потрібно виконати для отримання зображення тривимірного об’єкту:

Моделювання – це процес розробки сцени, що є тривимірною математичною моделлю та об’єктів, що в ній знаходяться.

Рендеринг (візуалізація) – процес розрахунку проекцій.

Зображення результату на пристрій відображення.

Наступним етапом є моделювання. Невід'ємною часткою моделювання є сцена. В неї включаються наступні категорії:

1. Геометрія (модель, що будується використовуючи різних технології створення тривимірних об'єктів, наприклад будівля).

Матеріали (так називають інформацію про характеристики моделі, це може бути колір, коефіцієнти заломлення та відбиття).

Джерела світла (керування напрямками, сила світла та спектр освітлення)

Головною задачею моделювання тривимірних об'єктів – максимально описати і розмістити об'єкти у сцені використовуючи геометричні перетворення щоб наблизитись до вимог поставлених для виведення зображення.

Рендеринг. Головною задачею цього етапу є перетворення просторового об'єкту на плоске двовимірне зображення. При створенні фільму потрібно зарендерити всі кадри, що являють собою послідовність зображень, які виникли в результаті перетворення. Структурно дані зображення представлені матрицею, що складається з точок, які мають інтенсивності. Тобто рендеринг це процес перетворення тривимірної векторної структури в набір пікселів, матрицю. Такий процес є обчислювально складним, що є помітно при обробці великих тривимірних моделей.

Часто використовують наступні технології візуалізації, які комбінуються разом. Наприклад: Z-буфер (цю технологію використовують в OpenGL та DirectX10).

Сканлайн (scanline) – або як ще називають Ray casting (цей алгоритм є спрощеним алгоритмом зворотного трасування променів) – обчислення кольору пікселя картинки за допомогою спрямування променю з точки, яку називають камерою [6].

Трасування променів – робота методу схожа на Ray casting, але розрахунок кольорів є більш точним в силу того, що аналізується більша кількість променів, в тому числі тих, що заломилися.

Глобальне освітлення – метод працює за рахунок розрахунку, як взаємодіють поверхонь та середовища між собою у видимому спектрі випромінювання, та обчислюється використовуючи інтегральні рівняння. В силу того, що при рендеренгу багато обчислень є однотипними, то він зазвичай розпаралелюється.

1.4. Типи представлення тривимірних об'єктів

3D-моделювання визначення комп'ютерної графіки зазвичай трактують як процес розробки математичного уявлення тривимірної поверхні об'єкту використовуючи спеціалізоване програмне забезпечення. В результаті моделювання на виході отримуємо SD-модель. Моделі зазвичай представлені в різних форматах, деякі з них описані нижче. Відповідно до формату модель можна відкрити за допомогою спеціальних програм відображення 3D – моделей. Ці моделі створюватися трьома способами: вручну, автоматично, або за допомогою сканера [5].

Формат VRML (мова моделювання віртуальної реальності). Файли VRML частіше всього мають розширення WRL. В ньому використовується формат кодування ASCII, тобто представлений як текстовий файл в якому описані об'єкти, що часто називають вузлами (nodes). Цими вузлами VRML 2.0 є 3D-

геометрія, коефіцієнти освітлення, що форматуються за допомогою VRML.

Самі вузли можуть складати ієрархічну структуру, що означає – деякі вузли можуть успадковувати характеристики інших вузлів, що знаходяться на вищих ієрархічних рівнях. Вузол має структуру, що полягає в типізації вузла, та даних вузла, що вказуються у фігурних дужках.

Типи даних:

Приватні значення поля, які захищені від зміни іншими вузлами. Також є тип полів, які, можуть змінюватись при дії на них іншими вузлами.

Поле задається наступними параметрами: тип поля; ім’я поля;

Вузли в VRML задаються в декартовій тривимірній системі координат.

Структура VRML-файлу така:

перший рядок це заголовок, що є обов’язковим #VRML V2.0 utf8. (починається коментар, його кінець є кінцем поточного рядка. Значення utf8 показує, що в VRML рядки записані в кодуванні стандарту UTF–8 ISO 10646);

за деякої кількості класів тривимірних або двовимірних об’єктів;

вузли;

Формат 3DS. Один з найпопулярніших форматів 3D-графіки. Файли такого формату були створені разом з програмою 3D Studio. Найчастіше цей формат використовувався для розробки ігор. Даний формат кодувався по наступному принципу: спочатку були координати точок, а потім каркасні сітки, які являли собою саму 3D-модель. [35]

Формати MDL, MD2, MD3, SMD. Такі формати використовуються для представлення анімованих моделей, найчастіше анімації людей. Ці формати є досить популярними, адже для роботи з ними можна використовувати безкоштовні програмні комплекси для редагування тривимірних моделей. Ця анімована модель має досить великий дисковий об’єм. Адже кадри анімації зберігаються кожен окремо, тому відповідно кількість моделей в файлі залежить від кількості кадрів [4].

Формат X. Формат X- формат «Direct» для тривимірних моделей.

Direct3D: – це набір API що створений для написання програм зв’язаних з тривимірною графікою.

При наявності файлів у форматі 3DS, так само як і в форматі X, вводяться тексти в текстовому форматі. Сама структура (перетворена в формат BMP). Відображення формату X поля, в тому числі, для отримання додаткової інформації в 3dsconv, Який може бути конвертований в популярний формат 3DS в форматі X. Ви можете легко модернізувати в Kinetix 3D Studio MAX, Використовуючи для цього формат 3DS.

Формат OBJ. Формат OBJ – це формат файлу в якому описано геометрію об’єкту, вперше. Цей фал не кодується, тобто можна в текстовому редакторі побачити всі характеристики 3D- моделі. Типовий файл OBJ виглядає приблизно як на рисунку 1.13:

```
# це коментар
# Список вершин заданих координатами (x,y,z)
v 0.123 0.234 0.345
v ...
...

#Текстурні координати (u,v).
vt 0.500 -1.352
vt ...
...

#Нормалі (x,y,z); нормалі можуть бути не нормалізовані.
vn 0.707 0.000 0.707
vn ...
..

#Кожна грань задається множиною трьох індексів кожен з яких відповідає за вершину/текстуру/нормаль
#координати яких записані в списках вище
#Тому f 1/1/1 2/2/2 3/3/3 це прикутник, що має текстурні координати та нормалі для
#всіх трьох вершин
#ЗАУВАЖЕННЯ: Списки нумеруються починаючи з одиниці.

f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3
f ...

# Чотирикутники, по яких многокутники задаються відповідною кількістю вершин
f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3 v4/vt4/vn4
f ...
...
```

Рис. 1.13. Типовий вигляд файлу OBJ

Формат PLY. PLY – формат опису геометрії тривимірних об’єктів, часто його називають Polygon File Format або Stanford Triangle Format. Його призначення полягає в зберіганні тривимірних даних які отримані з 3D-сканерів. В цьому форматі опис тривимірного об’єкту є відносно простим і представлений, як список плоских полігонів.

PLY файл може бути розширеним додатковими характеристиками моделі, а саме: значеннями кольору, коефіцієнтом прозорості, заданими нормалями до поверхні і т.д.

Початком файлу є заголовок, де описано які елементи потрібно використовувати при відображенні моделі. Найпростіша модель зазвичай має елементи полігональних сіток і їх типи. Елементами є вершини (координати) і грані (набір точок). Перший рядок заголовку повинен мати слово ply, це дає змогу програмі відображення зрозуміти, яким способом представлені дані. В другому рядку вказано до якого типу кодування належить даний файл,

наприклад format ascii 1.0. Для того, щоб записати коментар, потрібно почати рядок зі слова comment та після нього описати коментар. Відповідно коментар не буде братись до уваги програмою відображення. Якщо рядок починається зі слова element, то це означає, що в ньому буде описано деякий елемент, що потрібний для зображення моделі, а також скільки таких елементів є у файлі [3].

Після введення елементу, потрібно описати його властивості за допомогою слова property, в попередньому прикладі видно, що властивості вершини задаються, як координати. Типи властивостей задаються по аналогії з типізованими мовами програмування, а саме char uchar short ushort int uint float double або одним з int8 uint8 int16 uint16 int32 uint32 float32 float64. Частіше всього у фалі задаються грані. Вони, та їх властивості, зазвичай, представляють таким чином:

element face 10

property list uchar int vertex_indices

властивість list, тобто в перекладі список означає, що грані задаються списком значень, де перше число, означає кількість вершин грані, а після вказані індекси вершин, що були задані до граней. Кінець заголовку можна розпізнати по рядку: end_header.

РОЗДІЛ 2.
ОСНОВНІ АЛГОРИТМИ СТВОРЕННЯ 3D-ОБ’ЄКТІВ

2.1. Сутність та основні принципи полігонального моделювання

Наразі складні 3D-моделі, які були б неможливими ще пару років тому, можуть бути створені за припустимий проміжок часу, завдяки програмному забезпеченню, що описує тривимірну геометрію. Головний виклик полягає в оптимізації процедур, яка повинна бути одночасно швидкою і надійною.

3D-об'єкти можуть створюватися автоматично, або вручну завдяки деформації, наприклад, редагування вершин. Процес 3D-моделювання створює цифрові об'єкти, які можуть бути повністю анімовані і використовуватися для анімації об’єктів сцени [2].

Основою моделі є сіть, яку можна задати як набір точок у просторі. Ці точки представляються у вигляді 3D сітки і з'єднані між собою у вигляді багатокутників (зазвичай трикутників або чотирикутників). Кожна вершина має власне розміщення в просторі, і шляхом об'єднання цих точок у фігуру задається площина об'єкта.

Полігональне моделювання - це нижчий рівень моделювання, що дає можливість візуалізувати об'єкти за допомогою полігональних сіток. Полігональна сітка складається з простих фігур (полігонів). Багатокутники-цетри кутники або чотирикутники з вершинами, ребрами та гранями.

Моделі, створені методом полігонів, є фігурами, що складаються з багатокутників з різним ступенем спотворення, так що об'єкт має визначену постійну форму. Гладкість об'єкта можна регулювати, змінюючи кількість полігонів, з яких складається об'єкт. Для візуалізації, фігура з великою кількістю полігонів буде гладшою, ніж фігура з малою кількістю полігонів (Рис.2.1).

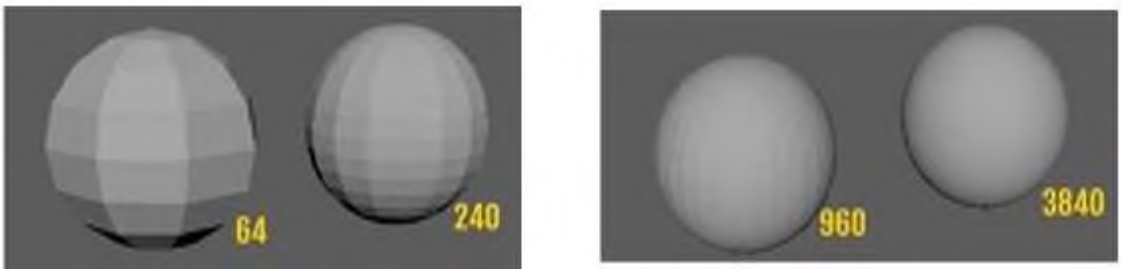


Рис. 2.1. Сфери, що складаються з різної кількості полігонів

В залежності від кількості полігонів у тривимірній моделі їх ділять на наступні види [24]:

високополігональні (high-poly) – з великою кількістю полігонів;

середньополігональні (mid-poly) – з середньою кількістю полігонів;

низькополігональні (low-poly) – з малою кількістю полігонів.

Моделі з високою кількістю полігонів є фотореалістичнішими і тому можуть використовуватися для виробництва анімаційних фільмів. Однак цей тип моделей вимагає багато витрат у вигляді програмних і апаратних ресурсів. Крім того, оскільки кожен кадр прораховується попіксельно, рендеринг займає багато часу.

Високополігональні моделі рідко використовуються в комп'ютерних іграх через повільне завантаження кадрів. Щоб уникнути перевантаження програми та відеокарти комп'ютера, розробники оптимізують візуальну частину гри відповідно до можливостей комп’ютера користувача.

У випадку з високополігональними моделями, низькополігональні моделі створюються з трикутних полігонів. Для створення низькополігональної моделі залишаються тільки ті полігони, які впливають на форму силуету. Невидимі елементи видаляються [33]. До моделі додаються джерела світла та коригуються світлі та темні ділянки (затінення). Потім аналізується напрямок нормалей обох моделей. Нормалі - це вектори, які використовуються для визначення того, як світло відбивається від поверхні.

Побудова променів програмою виконуєтсья в напрямку нормальних векторів низькополігональної моделі. При зіткненні з високополігональною поверхнею, вібувається обчислення про те, як зображення цих променів виглядатме вздовж нормальних векторів високополігональної моделі. Дані про нормальні вектори зберігається у структурті “мапа нормалей”. Таку процедуру називають “запіканням” (рис. 2.2).

Рис. 2.2. Процедура «запікання»

Карта нормалей потім накладається на модель з низьким полі. Оскільки полігональне моделювання не пов’язане з реальними одиницями вимірювання, цей метод моделювання не підходить для створення моделей із точними розмірами, наприклад на основі креслень або будівельних планів. Уніфікованість менш важлива у створенні візуального контенту, де важливішу роль відіграє художня візуалізація об’єктів.

Для моделювання багатокутників зазвичай використовуються трикутники, які називаються трикутниками, і чотирикутники, які називаються чотирикутниками; Багатокутники з більш ніж чотирма сторонами також можуть бути створені і часто використовуються в моделюванні.

Окремі багатокутники зазвичай називаються «гранями» і визначаються як області, обмежені трьома або більше вершинами та пов’язаними з ними ребрами. Коли багато граней з’єднується разом, створюється мережа граней, яка називається багатокутною сіткою (також

називається набором багатокутників або багатокутним об’єктом). Таким чином, 3D-моделі створюються за допомогою багатокутних сіток.

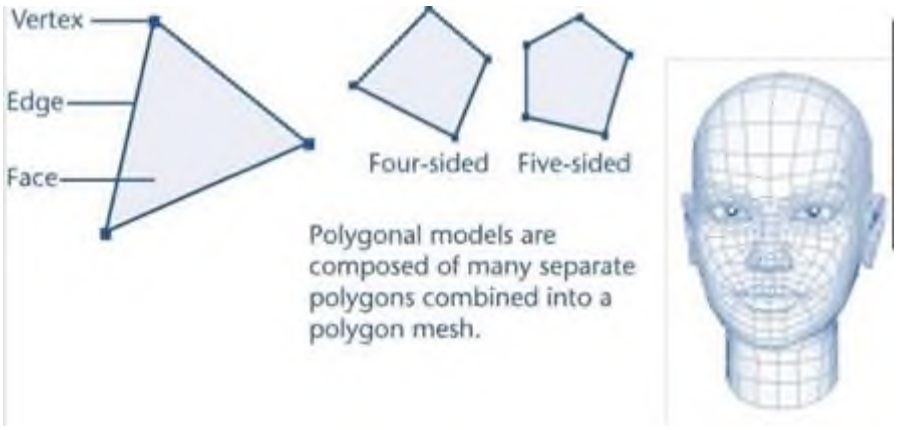


Рис. 2.3. Полігональне моделювання

Багато дизайнерів персонажів у фільмах роблять так, щоб поверхні їхніх моделей були гладкими і виглядали якомога природніше. У випадку, якщо оригінальний зразок має впорядковану структуру, існують спеціальні інструменти, які можуть перетворити моделі з низькою роздільною здатністю на моделі з високою роздільною здатністю майже автоматично. Такий метод поділу полігонів (стиснення кадру) називається згладжуванням [26].

Процес поділу полігональних поверхонь створює на зображенні нові елементи, які потрібно правильно розмістити. Утворення додаткових полігонів регулюється набором правил.

Декілька точок, які не належать одній прямій, перетворюються кривою вищого порядку, ніж дана пряма, що призводить до такого наближення поверхні.

Простішим є спосіб обчислити положення додаткових елементів - зрізати певні кути. Якщо гострий крок віртуальної моделі персонажа заокруглити за допомогою математичної формули, поверхня буде більш гладкою. Це можливо завдяки методу, який називається зрізанням кутів.

Він має середню точку або центр багатокутників, що утворюють поверхню об’єкта, з'єднує їх, створюючи проміжні багатокутники. На кожному кроці кути відсікаються, і об'єкт стає більш гладким.

Однією з цікавих властивостей цієї операції є те, що при використанні патча на основі B-сплайна можна отримати контур поверхні після нескінченної

кількості прогонів. Проаналізуємо математичну сутність найпростіших фігур що використовуються у моделюванні. Сфера задається рівнянням S (x) = (x-C; x-C) = r^2, де C-центр сфери, R-радіус, x-точка, що належить сфері. Перетин з променем розраховується наступним чином:

$$S(R(t)) = (R(t) - C, R(t) - C) = r^2;$$
$$(O + t \cdot D - C, O + t \cdot D - C) = r^2;$$
$$(D, D)t^2 + 2(D, CO)t + (CO, CO) - r^2 = 0, \text{ где } CO = O - C;$$
$$t^*_1 = -(D, OC) - \sqrt{(D, CO)^2 - (CO, CO) + r^2};$$
$$t^*_2 = -(D, OC) + \sqrt{(D, CO)^2 - (CO, CO) + r^2},$$

враховуючи, що D нормалізується.

Якщо discr = (d; OC) ^{2}- (OC; OC) + $r^2 \geq 0$ і хоча б один з t_1 або t_2 невід’ємний, значить, перетин є, вибираємо найменший з невід’ємних t, позначимо його за t∗, тоді точка перетину дорівнює R (t^*) = O + t^* * D. Тоді Нормаль дорівнює N = R (t∗) – C = O + t∗ · D–C.

Рис. 2.4. Рендер сфери а), площини б) та трикутника

Площина. Площина задається рівнянням P (x) = (p – p_0; N) = 0, де p_0 – точка, що належить прямій, N – нормаль площині. Тоді перетин променю R (t) = O + t · D знаходиться наступним чином:

$$P(R(t)) = (R(t) - p_0; N) = 0;$$

$$(O + t \cdot D - p_0; N) = 0;$$

$$t = (p_0 - O; N) / (D; N):$$

Якщо / , то вважаємо, що перетин є. Нормаль в точці перетину дорівнює нормалі площині.

Трикутник. Трикутник визначається трьома точками A; B; C. Шукати перетин з променем R (t) = O+T·D будемо методом Моллера-Трумбора [32]. Даний метод примітний тим, що він не вимагає попереднього обчислення рівняння площини, що містить трикутник. Обчислення перетину виглядає наступним чином. Розглянемо барицентричні координати точки всередині трикутника P = w · A + u · b + v · C. враховуючи, що u + v + w = 1, виразимо w = 1-u-v. Підставляючи в вираз точки P отримуємо:

$$P = (1 - u - v) \cdot A + u \cdot B + v \cdot C;$$
$$P = A + u \cdot AB + v \cdot AC, \text{ де } AB = B - A, AC = C - A;$$

Підставляючи R (t) = O + t · D в рівняння вище отримуємо:

$$O + t \cdot D = A + u \cdot AB + v \cdot AC;$$

$$AO = -t \cdot D + u \cdot AB + v \cdot AC; \text{ де } AO = O - A;$$

$$\begin{bmatrix} -D & AB & AC \end{bmatrix} \cdot \begin{bmatrix} t \\ u \\ v \end{bmatrix} = AO;$$

Використовуючи метод Крамера, ми отримуємо:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{\begin{vmatrix} -D & AB & AC \end{vmatrix}} \begin{bmatrix} \begin{vmatrix} AO & AB & AC \\ -D & AO & AC \\ -D & AB & AO \end{vmatrix} \end{bmatrix};$$

При цьому, якщо / , то це означає, що перетину немає.

Враховуючи, що $\begin{vmatrix} V_1 & V_2 & V_3 \end{vmatrix} = V_1 \cdot (V_2 \times V_3)$ і $AB \times AC = N$ - нормальний трикутника, отримуємо:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{(D,N)} \begin{bmatrix} (AO,N) \\ -D \cdot (AO \times AC) \\ -D \cdot (AB \times AO) \end{bmatrix}$$

Позначимо знайдене значення t для t^*, тоді трикутник і промінь має перетин в R (t^*), якщо $t^* \geq 0$; u + v <= 1 і u ≥ 0; v ≥ 0. Нормаль в точці перетину дорівнює N = AB × AC

2.2. Моделювання кривих nurbs та процедурне моделювання

NURBS–Non-Uniform Rational B-Splines (Неоднорідний раціональний B-сплайн) – це один тип геометрії, який можна використовувати для створення 3D-кривих та поверхонь [27].

Non-Uniform (неоднорідний) відноситься до параметризації кривої. Неоднорідні криві дозволяють наявність багато вузлів, які потрібні для представлення кривих Безьє.

Rational (раціональний) відноситься до основного математичного представлення. Ця властивість дозволяє NURBS представляти точні канонічні перерізи (такі як параболічні криві, кола та еліпси) на додаток до кривих вільної форми.

B-сплайни – це кускові поліноміальні криві, які мають параметричне зображення.

NURBS корисні для побудови багатьох видів органічних ЗБ-форм через плавний та мінімальний характер кривих, які вони використовують для побудови поверхонь. Типи поверхні NURBS широко використовуються в галузі анімації, ігор, наукової візуалізації та промислового дизайну.

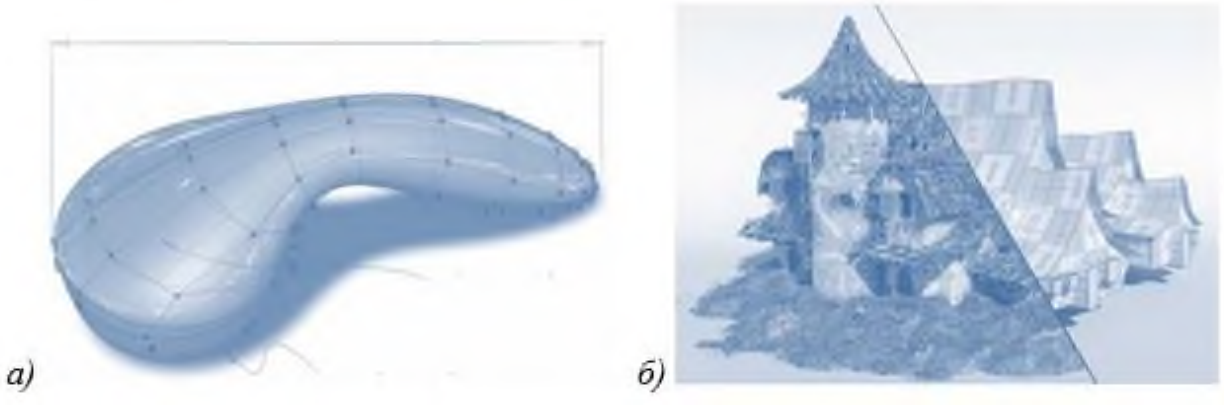


Рис. 2.5. а) Моделювання кривих NURBS б) процедурне моделювання

Процедурне моделювання буває різних форм і розмірів, є два типи моделювання. Перший – на інструментальній основі. Ми або хтось інший створили інструмент, призначений для процедурного генерування купи подібних об’єктів. Наприклад, у нас може бути генератор будівлі. Тоді ми могли б ввести купу таких параметрів, як кількість поверхів, високу стелю і яку форму має мати дах. Потім ми кілька разів запускаємо програму, і щоразу з’являється нова модель, яка відповідає нашим критеріям.

Наступний вид процедурного моделювання тісно пов’язаний із затіненням. Шейдер може мати зсув результату, і завдяки цьому зсуву беручи такі прості примітиви, як сфера чи площина, і використовуючи математичні формули для деформації поверхні, вони можуть стати складним об’єктом чи поверхнею. Це тенденція, яка зростає, коли все більше і більше інструментів стали доступними для витіснення геометрії за допомогою затінення.

Доступні як традиційні переміщення, що працюють на осі вгору і вниз, так і переміщення вектору.

2.3. Сплайнове моделювання

Сплайни (Spline – шматково-поліноміальна функція) – це двовимірні геометричні об’єкти. З них можуть бути створені більш складні тримірні тіла, але також вони можуть бути абсолютно самостійними. Сплайнами є

різноманітні лінії, форма лінії визначається типом вершин, через які вона проходить [28].

Сплайни це як і найпростіші геометричні фігури (прямокутники, зірки, еліпси і ін.), так і складні ламані або криві, а також контури текстових символів. Спочатку будується сплайновий каркас, для створення моделі за допомогою тривимірних кривих, а далі на основі цього каркасу огинаюча тривимірна геометрична поверхня. Гладкість кривої визначає набір тривимірних контрольних точок, які задають тримірні криві. Крім того, в сплайновому моделюванні використовуються сплайнові примітиви (лінії, дуги, спіралі, кола, кільця, еліпси, прямокутники, багатокутники). Об’єкти, створені сплайновим моделюванням, є гнучкі до редагування та зміни їх форми.

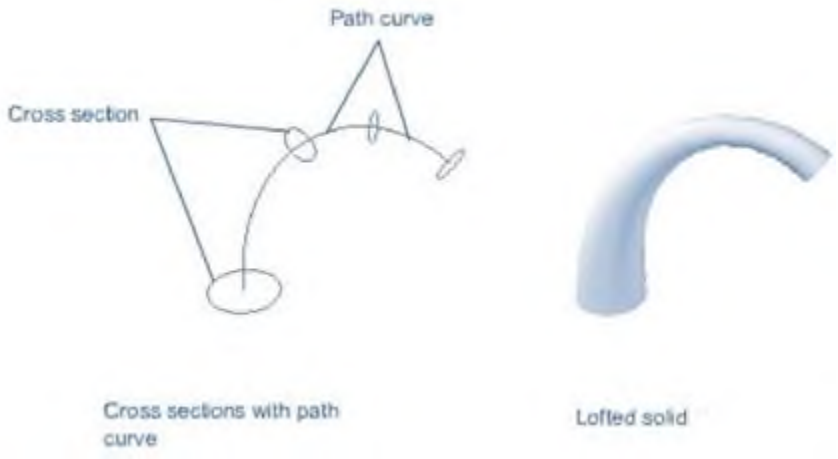


Рис. 2.6. Сплайнове моделювання

2.4. Метод кидання променів (ray casting) та Scanline rendering

Метод трасування променів являє собою метод генерації зображення шляхом відстеження, або трасування, шляху світла і моделювання ефектів його зіткнення з об’єктами [29]. Вперше в масовій індустрії даний алгоритм був застосований Джоном Кармаком в грі Wolfenstein 3D

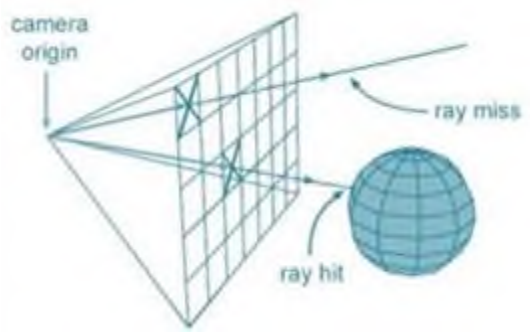


Рис. 2.7. Операція кидання променів

Він відомий своєю гнучкістю, як правило, більш досяжним реалізмом і масштабованістю, однак, як правило, досить трудомісткий. Даний метод випускає тільки первинні промені, що робить його нерекурсивним. Метод трасування променів (ray tracing) ж є рекурсивним алгоритмом, що робить його більш витратним в обчислювальному плані алгоритмом.

Однак, завдяки тому, що даний метод випускає не тільки первинні промені, але і вторинні, робить даний метод помітно більш реалістичним і дозволяє реалізувати багато ефекти, які недоступні методу кидання променів, таких як зображення, заломлення, м'які тіні та інші. Однак в даному методі присутні деякі припущення, що робить даний метод менш реалістичним, ніж його вдосконалена версія.

Більш вдосконалений метод називається методом трасування шляху (path tracing). Даний метод ґрунтується на так званому рівнянні рендеринга, запропонованому в статті Джеймса КаджWF-Sp800nii в 1982 році, і його вирішенні методом Монте-Карло. Даний алгоритм є методом без припущень (unbiased method) [30], що робить його найбільш реалістичним методом з трьох.

Трасування променів здатне імітувати широкий спектр оптичних ефектів, таких як явища відбиття та заломлення, розсіювання та дисперсії (наприклад, хроматична аберація). На рис. 2.7 можна побачити як сфера

демонструє ефект малої глибини поля, джерела світла та дифузійного відбиття за допомогою рекурсивного трасування променів.

Оптичне трасування променів описує спосіб отримання візуальних зображень, побудованих у середовищі 3D – комп'ютерної графіки, з більшою кількістю фотореалізму, ніж методики «Ray casting» або методів «Scanline rendering». Він працює, відстежуючи шлях від уявного ока через кожен піксель на віртуальному екрані та обчислює колір об'єкта, видимого через нього.

Сцени в трасування променів описуються математично програмістом або візуальним художником (як правило, використовуючи посередницькі інструменти). Сцени також можуть містити дані із зображень та моделей, знятих такими засобами, як цифрова фотографія.

Ray casting може стосуватися різноманітних проблем і прийомів:

загальна проблема визначення першого об'єкта, що перетинається променем, як при виявленні зіткнень.

методика прихованого видалення поверхні, заснована на знаходженні першого перетину променю, відкинутого від очей, через кожен піксель зображення.

алгоритм відтворення нерекурсивного трасування променів, який відкидає лише первинні промені.

об'ємне випромінювання променю, метод прямого об'єму, в якому промінь «проштовхується» через об'єкт, а 3D-скалярне поле, що цікавить, відбирається вздовж променю всередині об'єкта.

Хоча «Ray casting» та «Ray tracing» часто використовувались взаємозамінно в ранній літературі BD-комп'ютерної графіки, більш недавнє використання намагається розрізнити ці два, хоча і не дуже успішно. Основна відмінність полягає в тому, що термін «Ray casting» ніколи не застосовується, коли промені простежуються рекурсивно [31].

Ray casting – головний з багатьох алгоритмів візуалізації комп'ютерної графіки, які використовують геометричний алгоритм трасування променів. Алгоритми візуалізації на основі трасування функціонують у порядку зображення для відображення тривимірних сцен на двовимірних зображеннях. Геометричні промені простежуються від очей спостерігача для зразка світла (сяйва), що рухається до спостерігача з напрямку променю.

Швидкість і простота випромінювання «Ray casting» пояснюється обчисленням кольору світла, не рекурсивно відстежуючи додаткові промені, які відбирають випромінювання, що падає в точку, в яку потрапив промінь. Це виключає можливість точного відображення відбитків, заломлення або природного випадання тіней; однак усі ці елементи можуть бути підроблені певною мірою за допомогою творчого використання текстурних карт або інших методів. Висока швидкість обчислення зробила випромінювання зручним методом візуалізації у ранніх 3D-іграх у реальному часі.

У природі джерело світла випромінює промінь світла, який рухається, врешті-решт, до поверхні, яка перериває її прогрес. Про цей «промінь» можна

думати як про потік фотонів, що подорожує тією ж доріжкою. У цей момент з цим світловим променем може статися будь – яке поєднання трьох речей: поглинання, відбиття та заломлення. Поверхня може відображати весь або частину променю світла в одному або декількох напрямках. Він також може поглинати частину світлового променю, що призводить до втрати інтенсивності відбитого та / або заломленого світла. Якщо поверхня має якісь прозорі або напівпрозорі властивості, вона заломлює частину світлового променю в себе в іншому напрямку, поглинаючи деякий (або весь) спектр (і, можливо, змінюючи колір). Між поглинанням, відбиттям і заломленням все світло що повинно

враховуватися, і не більше. Наприклад, поверхня не може відображати 66% променів світла і заломлювати 50%, оскільки два склали б 116%. Звідси відбиті або заломлені промені можуть вражати інші поверхні, де їх поглинаючі, заломлюючі та відбиваючі властивості знову обчислюються на основі вхідних променів. Деякі з цих променів подорожують таким чином, що вони потрапляють у наше око, змушуючи нас бачити сцену і так сприяють остаточному виведеному зображенню. Спроба моделювати цей реальний процес відстеження світлових променів за допомогою комп'ютера може вважатися надзвичайно марнотратною, оскільки лише незначна частка променів у сцені насправді досягає очей [32].

Scanline rendering – це алгоритм визначення видимої поверхні в 3D – комп'ютерній графіці, який працює на основі рядків за рядком, а не на багатокутнику за полігоном або пікселем за пікселем. Усі багатокутники, які потрібно винести, спочатку сортуються за верхньою координатою Y, на якій вони спочатку з'являються, потім кожен рядок або лінія сканування зображення обчислюється за допомогою перетину сканування з полігонами на передній частині відсортованого списку, при цьому відсортований список оновлюється, щоб відкинути вже не видимі багатокутники, оскільки активна лінія сканування просувається вниз по малюнку.

Основна перевага цього методу полягає в тому, що сортування вершин вздовж нормальної площини сканування зменшує кількість порівнянь між ребрами. Ще одна перевага полягає в тому, що не потрібно переводити координати всіх вершин з основної пам'яті в робочу пам'ять – лише вершини, що визначають ребра, які перетинають поточну лінію сканування, повинні бути в активній пам'яті, і кожна вершина читається лише один раз. Основна пам'ять часто дуже повільна в порівнянні з зв'язком між центральним процесорним блоком і кеш-пам'яттю, і, таким

чином, уникнення повторного доступу до вершин в основній пам’яті може забезпечити значне прискорення.

Цей алгоритм може бути легко інтегрований з багатьма іншими графічними методами.

2.5. Використання рівняння дифузії для згладжування об’єктів

Головною проблемою використання 3D моделей є складність операцій над 3D об’єктами. Однією з найважливіших операцій є згладжування знімків, бо при скануванні 3D-сканером, або МРТ часто отримують результати з артефактами, які не властиві об’єкту який був сканований.

Є різні методи згладжування 3D об’єктів. В більшості з яких є велика складність обчислень. Що потребує великих обчислювальних потужностей. Тому постає потреба в розробці методів, що матимуть меншу складність. Завдяки сучасним пристроям збору даних, дані 3D-поверхні стали більш поширеними. При роботі з подібними даними, наприклад, отриманими лазерним сканером або системою стерео зору, завдання згладжування поверхні терміново виникає з різних причин.

По-перше, реальні дані мають тенденцію бути зашумленеми та зіпсованими сторонніми людьми. Таким чином, щоб проаналізувати та візуалізувати дані, першим кроком попередньої обробки доводиться боротися із видаленням шуму та перешкод.

Анізотропна дифузія, або як її ще називають дифузія Перона-Маліка – це метод, що використовується в обробці зображень та комп’ютерному баченні, ціллю якого є зменшення шуму вхідного об’єкту, при цьому залишаючи важливі частини об’єкту, зазвичай це дані, які є важливими при інтерпритації об’єкту.

Процес анізотропної дифузії схожий на таку послідовність дій, спочатку створюється масштабований простір, в якому генерується сім’я

розмитих об’єктів, за допомогою дифузійних процесів. Такий процес є лінійним та просторово-інваріантним до перетворень вихідного об’єкту. Анізотропна дифузія це узагальнення такого дифузійного процесу: використовуючи її створюється сімейство параметризованих об’єктів, але кожен отриманий об’єкт є поєднанням вихідного об’єкту та фільтру, що залежить від характеристик вихідного об’єкту. Тобто, анізотропна дифузія це нелінійна і просторово-варіантна трансформація вихідного об’єкту [30].

Формулювання, що представили Перон і Малік в 1987 році, просторово-варіантний фільтр – це ізотропія, що є залежною від вмісту об’єкта, адже вона наближається до імпульсної функції по краям та інших структурах, що повинні були збережені в об’єкті на різних ієрархічних рівнях у результаті масштабування простору. Таке формування терміну анізотропної дифузії називають дифузію Перона і Маліка чи нелінійною і неоднорідною дифузією.

Більш загальна рецептура дозволяє фільтру, пристосованому до початкових умов, нагадувати анізотропні об’єкти лінійної структури, такі як ребра або лінії: його орієнтація визначається такою структурою, вона витягнута вздовж структури і має вузький перетин. Такі прийоми називають згладжуванням, адаптованим до форм. В результаті отримані зображення зберігають лінійні структури і в той же час згладжування виконується уздовж цих структур. Обидва випадки можна описати узагальненням звичайного рівняння дифузії, де коефіцієнт дифузії замість того, щоб бути постійним скалярним, є функцією положення зображення і забезпечує матричні (або

тензорні) значення.

Анізотропна дифузія зазвичай реалізується шляхом наближення узагальненого рівняння дифузії: кожне нове зображення в сім’ї обчислюється, застосовуючи це рівняння до попереднього зображення.

Отже, анізотропна дифузія – це ітераційний процес, при якому для обчислення кожного наступного зображення в сім'ї використовується відносно простий набір обчислень, і цей процес триває до досягнення достатньої ступеня гладкості [4].

Формально нехай / позначає підмножину площини та $I(\cdot,t):\Omega\to$ – сім'я об'єктів. Тоді анізотропну дифузію можна представити

як $$\frac{\partial I}{\partial t} = div(c(x,y,t)\nabla I) = \nabla c\cdot\nabla I + c(x,y,t)\Delta I$$

де позначає оператор Лапласа, / позначає градієнт, div(...) це дивергенція оператора і с (x, y,t) – коефіцієнт дифузії. С (x, y,t) контролює швидкість дифузії. Перона і Малік при представленні анізотропної дифузії запропонували наступні функції для визначення коефіцієнта дифузії:

$$c(\|\Delta I\|) = e^{-(\|\Delta I\|/K)^2} \quad ma \quad c(\|\Delta I\|) = \frac{1}{1+-\left(\frac{\|\Delta I\|}{K}\right)^2}$$

Константа К визначає чутливість до країв і, як правило, визначається експериментально або ж як функція шуму на об'єкті. Нехай М позначає копію гладких об'єктів. Тоді дифузійні рівняння, представлені вище, можуть бути інтерпретовані як рівняння градієнтного спуску для мінімізації енергії $E:M\to$ визначеної як:

$$E[I] = \frac{1}{2}\int_\Omega g(\|\Delta I(x)\|^2)dx$$

де $g:\ \to$ є дійсною функцією і яка тісно пов'язана з коефіцієнтом дифузії. Тоді для будь-якої фінітної, нескінченно диференційованої тестової функції h, маємо:

$$\left.\frac{d}{dt}\right|_{t-0} E[I+th] = \left.\frac{d}{dt}\right|_{t=0} \frac{1}{2}\int_\Omega g(\|\nabla(I+th)(x)\|^2) = \int_\Omega g'(\|\nabla I(x)\|^2)\nabla I\cdot\nabla h dx =$$

$$= -\int_\Omega div(g'(\|\nabla I(x)\|^2)\nabla I)h dx$$

де останній рядок випливає з багатовимірного інтегрування частинами. Через / E позначимо градієнт E щодо L^2 (Ω,) прегільбетового простору оцінений в I. Це дає:

$$\nabla E_I = -div(g'(\|\Delta I(x)\|^2)\nabla I)$$

Таким чином, градієнтний спуск рівняння заданий як:

$$\frac{\partial I}{\partial t} = -\nabla E_I = div(g'(\|\Delta I(x)\|^2)\nabla I)$$

Тоді, взявши c g ми отримуємо анізотропне рівняння дифузії.

Регуляризація Перона-Маліка. При такому підході невідоме згортається з Гаусіаном для отримання модифікованих рівнянь Перона-Маліка.

$$\frac{\partial I}{\partial t} = div(c(|DG_\sigma * I|)\nabla I)$$

де $G_\sigma = C_\sigma^{-(1/2)} \exp(-|x|^2/4\sigma)$.

Коректність цього рівняння може бути досягнута шляхом регуляризації, а також використовуючи виведення ефектів розмитості, які є основними недоліками регуляризації. Також необхідно мати знання про рівень шуму, оскільки параметр регуляризації вибирається в залежності від рівня шуму.

Шейдерна мова високого рівня була введена Microsoft з DirectX 9.0 і називається HLSL. Як і у GLSL, HLSL пов’язаний з одним графічним API, а саме Direct3D.

На рис. 2.8 нижче показана програма вершин та фрагментів, написана HLSL для ефекту дифузії:

```
float4x4 matWorldViewProj;
float4x4 matWorld;
float4 vecLightDir;
struct VS_OUTPUT {
```

```
      float4 Pos: POSITION; 26
      float3 Light: TEXCOORD0;
      float3 Norm: TEXCOORD1; };
      // Вершинна программа
      VS_OUTPUT VS (float4 Pos: POSITION, float3 Normal: NORMAL){
            VS_OUTPUT Out = (VS_OUTPUT) 0;
            Out. Pos = mul (Pos, matWorldViewProj);
            Out. Light = vecLightDir;
            Out. Norm = normalize (mul (Normal, matWorld));
            return Out;}
      // Фрагментна програма
      float4 PS (float3 Light: TEXCOORD0, float3 Norm: TEXCOORD1):
COLOR
      {
            float4 diffuse = {1.0f, 0.0f, 0.0f, 1.0f};
            float4 ambient = {0.1f, 0.0f, 0.0f, 1.0f};
            return ambient + diffuse · saturate (dot (Light, Norm));
      }
```

Рис. 2.8. Вершинна та фрагментна програма для ефекту дифузії

Шейдерна мова OpenGL також відома як GLSL або glslang може використовуватися лише графічним OpenGL API. GLSL був визначений комітетом з архітектурного огляду під час специфікації OpenGL 2.0. На рис. 2.9 показана програма вершин та фрагментів, написана в GLSL для ефекту дифузії:

```
varying vec3 normal;
varying vec3 vertex_to_light_vector;

void main(){
      // Трансформація вектерної проекції на простір.
      gl_Position = gl_ModelViewProjectionMatrix · gl_Vertex;
      // Трансформація нормалів на модель огляду у просторі.
      normal = gl_NormalMatrix · gl_Normal;
      // Трансформація вертексної позиції на проекцію в просторі.
      vec4 vertex_in_modelview_space = gl_ModelViewMatrx · gl_Vertex;
      //Отримання вектору з вертексної позиції на позицію освітлення
      vertex_to_light_vector = vec3 (gl_LightSource [0]. position -
      vertex_in_modelview_space);
}

// Фрагментна програма
varying vec3 normal;
varying vec3 vertex_to_light_vector;
```

```
void main(){
// Визначення коефіцієнтів матеріалів.
      const vec4 AmbientColor = vec4 (0.1, 0.0, 0.0, 1.0);
      const vec4 DiffuseColor = vec4 (1.0, 0.0, 0.0, 1.0);
      vec3 normalized_normal = normalize (normal);
      vec3 normalized_vertex_to_light_vector =
      normalize (vertex_to_light_vector);
      float      DiffuseTerm      =      clamp      (dot      (normal,
vertex_to_light_vector), 0.0, 1.0);
      gl_FragColor = AmbientColor + DiffuseColor · DiffuseTerm;
}
```

Рис. 2.9. Вершинна та фрагментна програма на GLSL

Мова програмування Cg (Cg означає C для графіки) була розроблена NVIDIA і може використовуватися з обома основними графічними API: OpenGL та Direct3D. Cg має той самий синтаксис, що і HLSL. На рис. 2.10 У наведену просту програму вершин та фрагментів, написану на Cg:

```
struct appdata
{
      float4 Position: POSITION;
      float3 Normal: TEXCOORD0;
};
struct vfconn
{
      float4 HPOS: POSITION;
      float3 Normal: TEXCOORD0;
};

// Вершинна программа
vfconn main (appdata IN,
uniform float4x4 WorldViewMatrixIT,
uniform float4x4 WorldViewProjectionMatrix)
{
      vfconn OUT;
      OUT. HPOS = mul (WorldViewProjectionMatrix, IN. Position);
      OUT. Normal = mul (WorldViewMatrixIT, float4 (IN. Normal, 1)).
xyz;
      return OUT;
}
```

```
// Фрагмента программа
fragout main (vfconn IN)
{
      fragout OUT;
      float grey = clamp (dot (IN. Normal, float3 (0,0,1)), 0, 1);
      OUT. col = float4 (grey, grey, grey, 1);
      return OUT;
}
```

Рис. 2.10. Вершинна та фрагмента програма на Cg

На рис. 2.11 наведено приклад який визначає інтерфейс для всіх вогнів і дає точкову реалізацію цього інтерфейсу:

```
      // Інтерфейс для всіх вогнів.
interface ILight
{
      float3 illuminate (float3 p, out float3 L);
};
// Реалізація точкового світла інтерфейсу ILight.
struct PointLight: ILight
{
      float3 Plight, Clight;
      float3 illuminate (float3 P, out float3 L)
      {
            L = normalize (Plight – P);
            return Clight;
      }
};
```

Рис. 2.11. Приклад використання шейдерних інтерфейсів

Потім ці інтерфейси можуть бути реалізовані структурами, які можна перемикати під час виконання.

Для видалення шумів із цифрових об’єктів можна використовувати анізотропну дифузію, при цьому вона не розмиває країв, якщо коефіцієнт дифузії задається функцією. Якщо коефіцієнт дифузії є константою, тоді анізотропні рівняння дифузії зводяться до рівнянь теплопровідності, що є еквівалентними Гаусовому розмиттю. Це підходить для видалення шумів, але і розмиває краї теж. Перон і Маліка в своїх роботах задають коефіцієнт дифузії функцією, пошуку границь, тоді отримані рівняння згладжують

всередині областей. Тобто характеристики об’єкта можуть бути збережені при видаленні шуму об’єкта.

РОЗДІЛ 3.
РОЗРОБКА АЛГОРИТМІВ ДЛЯ РЕДАГУВАННЯ 3D МОДЕЛІ

3.1. Опис алгоритму згладжування

Для згладжування тривимірних об’єктів, було розроблено алгоритм, на основі методів дифузійного згладжування. Для згладжування тривимірного об’єкту було розроблено наступну послідовність дій:

1. формування з координат вершин матрицю;

знаходження площ всіх заданих трикутників;

знаходження кутів між всіма сусідніми вершинами;

дискретизація поверхні тривимірного об’єкту за допомогою оператору Лапласа-Бельтрамі;

використання перетворення Келі на дискретизованій поверхні;

отримання результату згладжування.

Рис. 3.1. Обчислення геометрії

Розглянемо детально дискретизацію поверхні тривимірного об’єкту за допомогою оператору Лапласа-Бельтрамі. Для виконання такої дискредитації потрібно підготувати дані тривимірного об’єкту. Щоб провести дискретизацію поверхні потрібно використати формули згортання з ядром Гауса:

$$g(x,t)=\frac{1}{4\pi kt}e^{\left(\frac{-x^2}{4kt}\right)}\cdot\ k\Delta u(x,t)=\partial_t u(x,t)$$

На діаграмі зображено процес дискретизації.

Рис. 3.2. Послідовність дій для дискретизації тривимірного об’єкту

3.2. Процес роботи з розробленим програмним забезпеченням

Спочатку опишемо процес зчитування тривимірного об’єкту. Для подання на вхід програмного забезпечення 3D модель повинна бути з розширенням *.ply. Адже зазвичай моделі в цьому форматі задаються в ASCII. На діаграмах зображена послідовність дій ПЗ для зчитування.

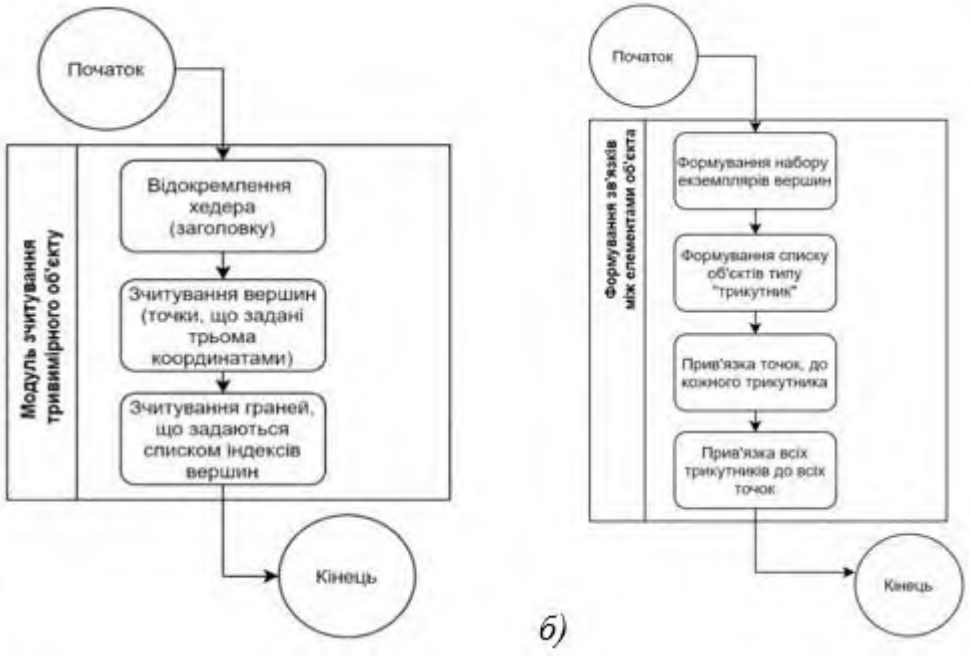


Рис. 3.3. Послідовність дій при зчитуванні тривимірного об’єкту а) та формування зв’язків між елементами тривимірного об’єкту б)

Більш детально розглянемо зчитування елементів 3D об’єкту, тобто вершин та граней – в нашому випадку трикутників. Зчитування елементів наведено в блок-схемі на рис. 3.4.

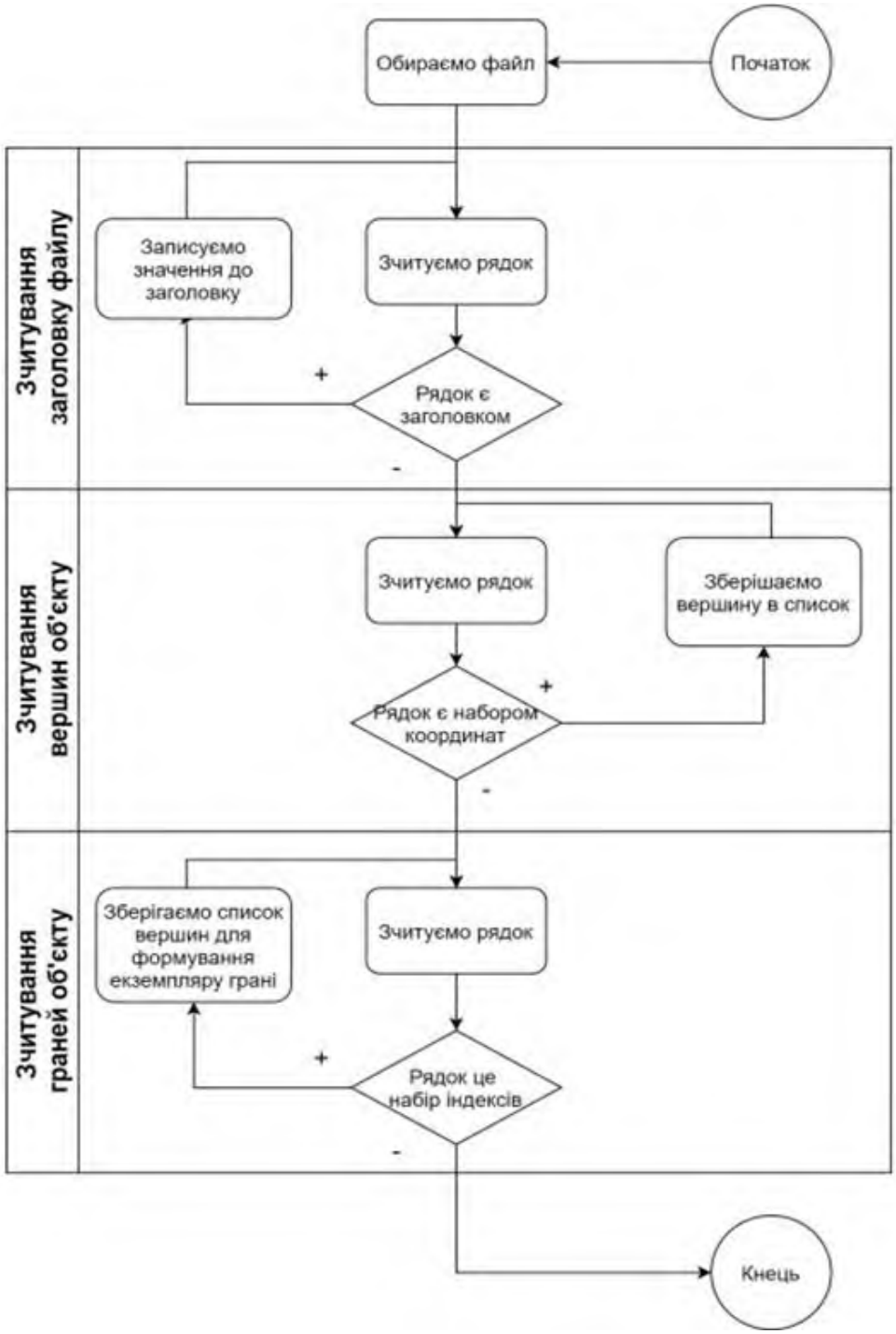


Рис. 3.4. Блок-схема алгоритму зчитування тривимірного об’єкту

Для більшої швидкодії обчислень дискретизації було прийнято рішення сформувати зв’язки між елементами тривимірного об’єкту. Загальну послідовність дій роботи розробленого програмного забезпечення можна побачити на рис. 3.5.

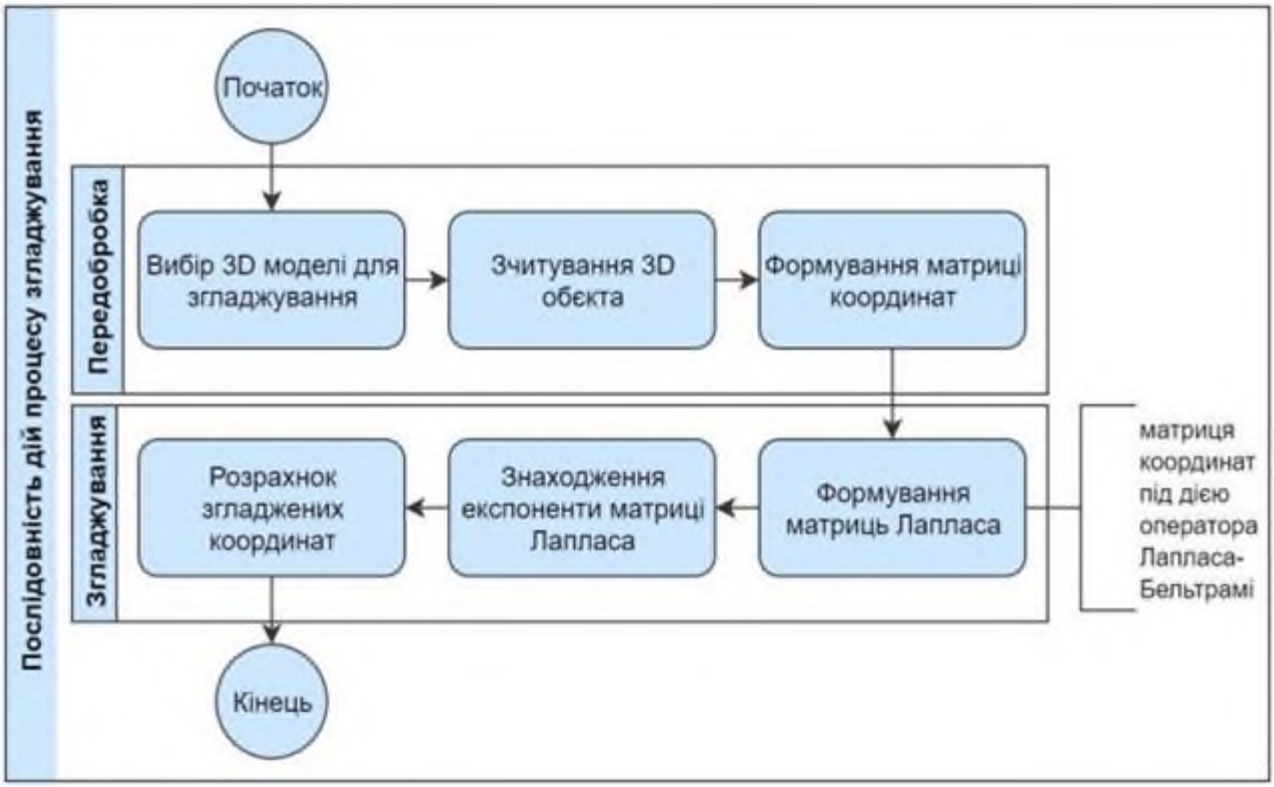


Рис. 3.5. Послідовність дій процесу згладжування

3.3. Основні вимоги до робочої станції для роботи з програмним забезпеченням

Платформа:. NET (C#).

Середовище розробки: Microsoft Visual Studio 2017.

Мінімальні системні вимоги:

64 – бітна операційна система Windows;

Microsoft. NET Framework 4.7;

70 КБ доступного простору на жорсткому диску;

Обсяг оперативної 8 ГБ;

3.4. Реалізація алгоритму «Ray casting»

Продемонстровано 3D реалізацію алгоритму «Ray Casting» за допомогою графічного рушія. Ray casting це алгоритм, який відображає 3D – світ на основі 2D – карти. Ray casting, як правило це кадри з низькою роздільною здатністю, це зроблено для того щоб запобігти відставанню картинки так як цей алгоритм в основному використовується на машинах з слабкою обчислювальною потужністю.

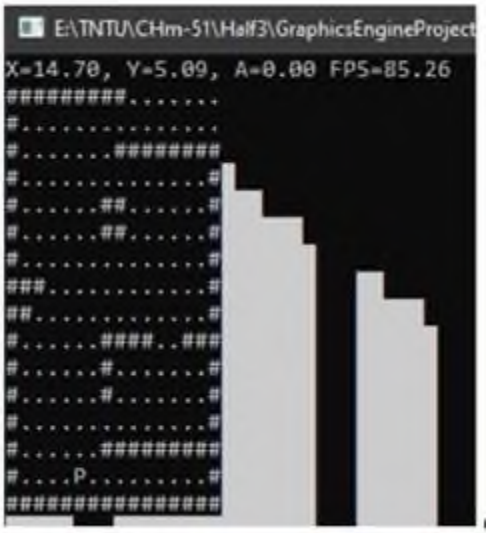
Ray casting не слід путати з ray tracing, який надає променям більшу фізичну точність, забезпечуючи відбиття та заломлення світлових променів, і відстежує промені у двох вимірах, а не в одному вимірі, як ray casting. Ray casting працює, викидаючи «промені», щоб виміряти відстань до найближчої стіни, звідси і термін «raycaster». Програма висилає промені, починаючи від початку камери, рухаючись вперед, поки камера не вдариться об стіну, і в цей момент камера проходить відстань, і намалює стовпчик на основі відстані. Чим ближче стіна, тим більша колона.

Промені надсилаються в різні боки, при цьому кут надсилання визначає, де колона буде намальована. Промінь, надісланий праворуч від точки зору гравця, намалює стовпчик у правій частині екрана, а промінь, надісланий ліворуч, намалює колонку зліва. Після відправлення всіх променів буде видно повну картину. Є два типи алгоритму ray casting, які можна запрограмувати: метод на основі спрайту та метод на основі списку.

В даній роботі реалізовано алгоритм на основі списку, на рис. 3.19 а наведено мапу сцени яка буде візуалізована.

```
// Карта створена з # блоків,
# = стіни, . = вільний простір.
wstring map;
map += L"#########.......";
map += L"#...............";
map += L"#.......########";
map += L"#..............#";
map += L"#......##......#";
map += L"#......##......#";
map += L"#..............#";
map += L"###............#";
map += L"##.............#";
map += L"#......####..###";
map += L"#......#.......#";
map += L"#......#.......#";
map += L"#..............#";
map += L"#......#########";
map += L"#..............#";
map += L"################";
```

а)



б)

Рис. 3.19. Характеристики мапи яка буде візуалізована а) та візуалізація мапи б)

На рис. 3.19 б можна побачити як карта та позиція камери відображаються у вікні візуалізації. На рис. 3.20 зображено як відображаються об’єкти які візуалізовано на відстані.

Рис. 3.20. Візуалізація об’єктів на відстані

На рис. 3.21 можна побачити як буде візуалізовуватися об’єкт праворуч який знаходиться на ближній дистанції до камери та як візуалізовується об’єкт що знаходиться у кінці мапи.

Рис. 3.21. Візуалізація об’єктів

На рис. 3.22 показано як алгоритм візуалізація об’єктів на ближній, середній на дальній дистанцій відповідно до напрямку погляду камери. У додатку наведено повний код програми.

Рис. 3.22. Візуалізація об’єктів на різній дистанції

На рис. 3.23 та 3.24 приведено 2D візуалізація алгоритму Ray Casting. Промені кидаються у всі сторони і відбиваються від об’єктиву на їхньому шляху, за рахунок чого відбувається прорахунок відстані до об’єкта та його рівень візуалізації, на даному рисунку промені кидаються у всі сторони так як в ньому продемонстровано погляд на об’єкти під кутом 365 градусів,

кожен об’єкт візуалізується відповідно до відстані. Повний код програми наведено у

додатку. Кожен з алгоритмів має свої переваги та недоліки відповідно до завдання яке ставиться перед ним. В даному підрозділі відображено швидкодію алгоритмів візуалізації та їх переваги.

Рис. 3.23. Часткова візуалізація алгоритму «Ray Casting»

Проаналізуємо переваги та недоліки алгоритму «Ray casting».

Переваги: легка реалізація; не потребує великої обчислювальної потужності.

Недоліки: невелика роздільна здатність картинки; повільний порівняно з іншими методами; багато артефактів у складних ситуаціях.

Рис. 3.24. Візуалізація алгоритму «Ray Casting»

Перелічимо переваги та недоліки алгоритму «Ray tracing».

Переваги: реалістичне відображення відбиття, заломлення, тіней; легко досягається згладжування ефектів.

Недоліки: потрібна велика обчислювальна потужність; важка реалізація; передові освітлювальні ефекти віжко відстежувати.

Назвемо переваги та недоліки алгоритму «Scanline».

Переваги: користується когерентністю, що призводить до збільшення швидкості; візуалізовує лише видимі пікселі.

Недоліки: складний до реалізації; потребує отримання всіх полігонів перед візуалізацією.

Алгоритм «Ray casting».

Переваги: легка реалізація; не потребує великої обчислювальної потужності.

На рис. 3.25 зображено графік який відображає швидкість візуалізації 60 кадрів кожним із алгоритмів.

Рис. 3.25. Порівняння швидкодії алгоритмів.

З графіка на рис. 3.25 видно, що для візуалізації 60 кадрів алгоритму «Ray Casting» потрібно 268 мілісекунд, алгоритму «Ray Tracing» потрібно 562 мілісекунд, алгоритму «Scanline» потрібно 361 мілісекунди, алгоритму «Ray

Casting – Oprimized» потрібно 162 мілісекунд. Найкращу якість візуалізації дає алгоритм «Ray Tracing», але за рахунок складність реалізації та потреб до високої обчислювальної потужності, його потрібно реалізовувати відповідно до завдань які ставляться перед розробником програмного забезпечення. На рис 3.26 наведено виміряні дані за допомогою програми «Intel Graphics Monitor».

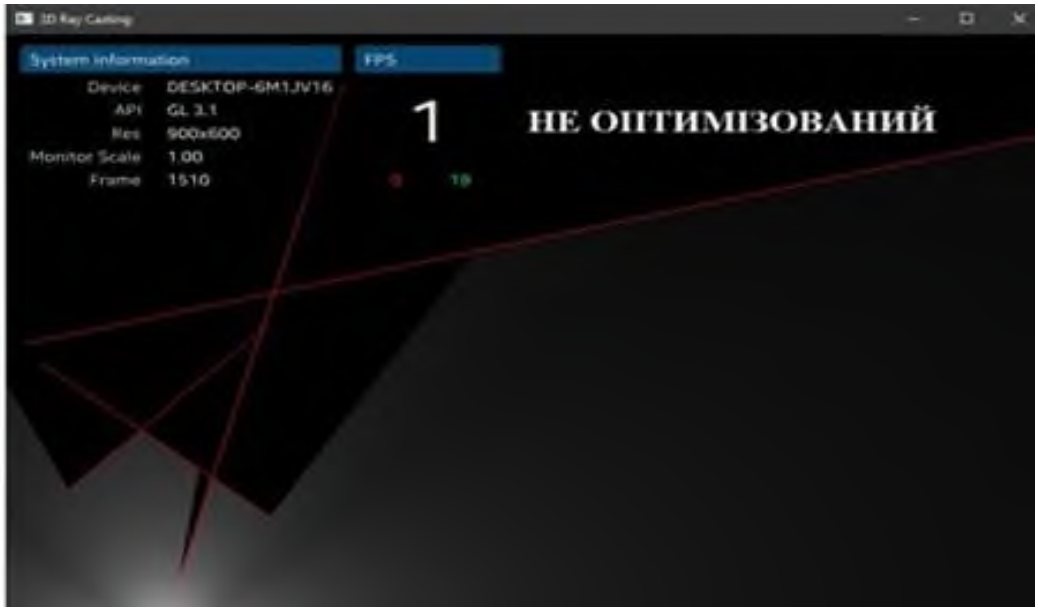


Рис. 3.26. Робота не оптимізованого алгоритму

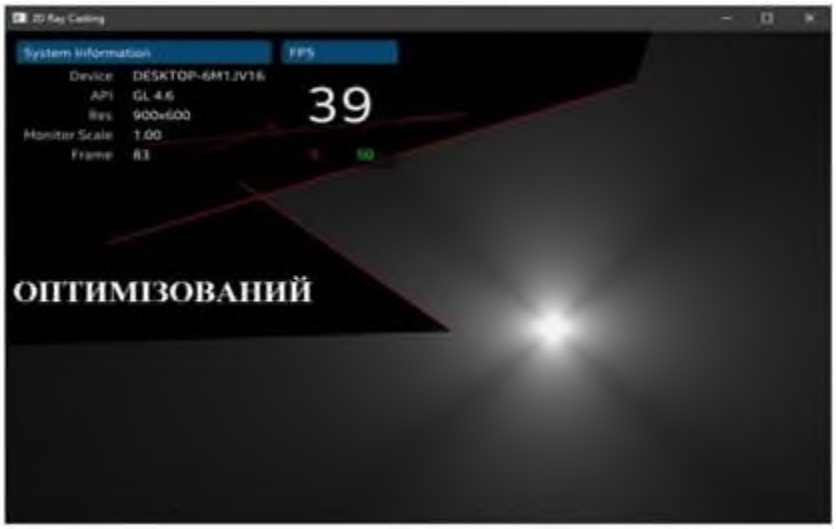


Рис. 3.27. Робота оптимізованого алгоритму

```
// Window dimensions
uniform vec2 iResolution;
// Camera position
uniform vec3 iRayOrigin;
// Camera facing direction


uniform vec3 iRayDir;
// Camera up direction
uniform vec3 iRayUp;
// Distance to viewing plane
uniform float iPlaneDist;
// ‘Texture’ coordinate of each
// vertex, interpolated across
// fragments (0,0) -> (1,1)
in vec2 texCoord;
vec3 getRayDir(
      vec3 CameraDirection,
      vec3 CameraUp,
      vec2 TexCoord)
{
      vec3 CameraSide = normalize(cross (CameraDirection,
CameraUp));
      vec2 p = 2.0 * TexCoord – 1.0;
      p. x *= iResolution.x / iResolution.y;
      return normalize(p.x * CameraSide + p.y * CameraUp +
iPlaneDist * camDir);
}

Hit TraceSphere (vec3 orig, vec3 dir, vec3 pos, float radius)
{
      float OdotD = dot (orig – pos, dir);
      float OdotO = dot (orig – pos, orig – pos);
```

```
        float base = OdotD * OdotD – OdotO + radius * radius;
        if (base >= 0) {
        float root = sqrt (base);
        float t1 = -OdotD + root;
        float t2 = -OdotD – root;
        if (t1 >= 0 || t2 >= 0)
        {
              float t = (t1 < t2 && t1 >= 0)? t1: t2;
              vec3 pt = orig + raydir * t;
              vec3 normal = normalize (pt – pos);
              return Hit (pt, normal, t);
        }
        return Hit (vec3 (0), vec3 (0), – 1);
  }
```

ВИСНОВКИ

1. На початку дослідження наголошено, що у сучасному світі 3D-моделювання займає лідерські позиції серед сфер людської діяльності. Важко уявити що-небудь сучасне, де б не використовувалася дана технологія, починаючи від підготовки якісного візуального контенту або створення мультфільму, закінчуючи виробництвом високотехнологічного обладнання. Розробка програмного забезпечення для представлення тривимірної графіки є одна з найскладніших областей інженерії програмного забезпечення через кількість та складність необхідних алгоритмів. Ці програмні проекти мають спільне те, що їм потрібно структурувати дані в основній пам’яті, обробляти їх та надсилати на графічний пристрій для ефективної візуалізації.

2. Розкрито історичні аспекти розвитку 3D-моделювання названо реалізовані приклади технічних засобів для візуалізації графіки. Оцінено сучасний стан галузі 3D-моделювання. Акцентовано на тому, що розробка нових програм та технологій для створення 3D-моделей та анімації спрощує роботу спеціалістів без втрати якості готового продукту. Дано короткий опис найбільш поширених інструментів для роботи з реалістичною комп’ютерної графікою.

3. Окремо досліджено сутність основних понять тривимірної комп’ютерної графіки та сказано, що головною задачею моделювання тривимірних об’єктів є максимально описати і розмістити об’єкти у сцені використовуючи геометричні перетворення щоб наблизитись до вимог поставлених для виведення зображення. В роботі також названо найбільш поширені типи представлення тривимірних об’єктів (3DS, MDL, MD2, MD3, SMD, VRML, X, OBJ, PLY).

4. Розкрито сутність та основні принципи полігонального моделювання. Зроблено висновок, що модель, яку створену методом

полігонального моделювання, можна назвати фігурою, яка складається з полігонів з різним

ступенем перспективного спотворення, за рахунок чого об’єкт має певну форму. Розкрито особливості моделювання кривих nurbs, процедурного та сплайнового моделювання. Досліджено можливості використання рівняння дифузії для згладжування об’єктів.

5. В практичній частині роботи дано опис алгоритму згладжування та проаналізовано черговість роботи з розробленим програмним забезпеченням. Приведено вигляд розробленого алгоритму та наведено програмний код.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бабич О., Семеніхіна О. До питання про співвідношення понять наочності і візуалізації. *Фізико-математична освіта*. 2014. № 2 (3). С. 47–53.
2. Бабенко Л. В. Комп'ютерна графіка [навчальний посібник для студентів вищих педагогічних навчальних закладів. Кіровоград: РВВ КДПУ ім. В. Винниченка, 2010. 250 с.
3. Безуглий Д. С. Візуалізація як сучасна стратегія навчання. *Фізико-математична освіта.* 2015. № 1 (7). С. 146.
4. Брянцева Г. В. Візуалізація навчального матеріалу з комп'ютерної графіки за допомогою асоціативних зображень-образів. *Освіта Донбасу.* 2011. № 6. С. 53–59.
5. Веселовська Г. В. Комп'ютерна графіка: навч. посібник для студентів вищих навчальних закладів / [Текст] за ред. В. Є. Ходакова. Херсон: ОЛДІ-плюс, 2011. 584 с.
6. Веселовська Г. В., Ходакова В. Є.: Компютерна графіка. Навч. пос. К.: Кондор, 2015. 584 с.
7. Ганжела С. І. Основи інформатики з елементами програмування та сучасні інформаційні технології навчання. Кропивницький: РВВ ЦДПУ ім. В. Винниченка, 2017. 61 с.
8. Жвалевський А., Гурська І, Гурський Ю. Комп'ютерна графіка: Photoshop CS3, CorelDRAW X3, Illustrator CS3. Трюки й ефекти. К.: Пітер, 2008. 992 с.
9. Ізонін І. В. Метод збільшення роздільної здатності зображень на основі штучних нейронних мереж. *Вісник Львівського державного університету безпеки життєдіяльності.* 2018. № 11. С. 47–56.