МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_д.т.н., доцент. Скарга-Бандурова І.С.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Дослідження та розробка системи для створення елементів комп'ютерної графіки

Освітньо-кваліфікаційний рівень "Магістр"

Спеціальність 123 - "Комп'ютерна інженерія"

Науковий керівник роботи:	(підпис)	О.І. Рязанцев (ініціали, прізвище)
Консультант з охорони праці:	(підпис)	Я.О. Критська (ініціали, прізвище)
Студент:	(підпис)	І.В. Нєскородєв (ініціали, прізвище)
Група:		КІ-17дм

Сєвєродонецьк 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень магістр
Напрям підготовки _____
(шифр і назва)
Спеціальність 123 Комп'ютерна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Завідувач кафедри комп'ютерних наук та інженерії
д.т.н., доцент І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Нєскродєв Іван Вікторович
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та розробка системи для створення елементів комп'ютерної графіки
керівник проекту (роботи) Рязанцев О.І. д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "18 " 10 2018 р. № ____
2. Строк подання студентом роботи _____
3. Вихідні дані до роботи Матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Огляд сучасних технологій та методів для вирішення задачі Аналіз методів досягнення завдань з використанням комп'ютерної геометрії Проектування та розробка системи для створення комп'ютерної графіки Питання охрани праці.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охрана праці	Критська Яна Олександрівна		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналітичний огляд літератури за темою роботи	01.09.18 – 1.10.18	
2	Огляд сучасних технологій для розробки комп'ютерної графіки	01.09.18 – 2.10.18	
3	Аналіз методів досягнення поставлених задач на основі комп'ютерної геометрії	03.10.18 – 09.10.18	
4	Проектування та розробка системи для створення комп'ютерної графіки	10.10.18 – 25.11.18	
5	Розгляд питань охрани праці та основних напрямків їх дотримання	13.11.18 – 15.12.18	
6	Оформлення пояснювальної записки	22.12.18 – 28.12.18	
7	Оформлення презентації роботи	29.12.18 – 07.01.19	

Студент _____ Нєскородєв І.В.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Рязанцев О.І.
(підпис) (прізвище та ініціали)

Анотація

Нєскородєв І.В. Дослідження та розробка системи для створення елементів комп'ютерної графіки.

Розглянуті різні види комп’ютерної графіки, методи і засоби розробки комп’ютерної графіки, проаналізовані сучасні методи для вирішення поставлених завдань. Було досліджено оптимальні алгоритми за допомого котрих створюються сучасні графічні програми та їх елементи. За допомогою досліджених джерел інформації було розроблено систему для створення комп’ютерної графіки, її елементів та складових.

Ключові слова: центральній процесор, відеоадаптер, комп’ютерна геометрія, стан виконання, синхронізація даних, менеджер станів, рушій, фреймворк, сцена, об’єкти, інтерфейс.

Аннотация

Нескородев И.В. Исследование и разработка системы для создания элементов компьютерной графики.

Рассмотрены различные виды компьютерной графики, методы и средства разработки компьютерной графики, проанализированы современные методы для решения поставленных задач. Было исследовано оптимальные алгоритмы с помощью которых создаются современные графические программы и их элементы. С помощью исследованных источников информации было разработано систему для создания компьютерной графики, ее элементов и составляющих.

Ключевые слова: центральной процессор, видеоадаптер, компьютерная геометрия, состояние выполнения, синхронизация данных, менеджер состояний, двигатель, фреймворк, сцена, объекты, интерфейс.

Annotation

Nieskorodiev I.V. Research and development the system for creating elements of computer graphics.

Considered various types of computer graphics, methods and tools for developing computer graphics, analyzed modern methods for solving problems. The optimal algorithms were investigated with an employer of which modern graphic programs and their elements are created. With the help of the studied sources of information, a system was designed to create computer graphics, its elements and components.

Keywords: central processor, video adapter, computer geometry, execution state, data synchronization, state manager, engine, framework, scene, objects, interface.

Зміст

Вступ

Актуальність теми. На сьогоднішній день складно уявити собі світ без комп’ютерної графіки. Вона знаходить собі застосування в багатьох сферах сучасного життя. Видавництва використовують графічні засоби під час створення ілюстрацій та обкладинок. Архітектори та дизайнери використовують графіку під час створення макетів будівель. Також графіка активно використовується для створення рекламних відео та логотипів, анімаційних фільмів та складних спецефектів у кіно. Але одна з найбільш розповсюджених сфер використання – це відеоігри.

Під час створення простих графічних сцен зазвичай не потрібно використання багатьох ресурсів комп’ютера. Але якщо потрібне створення тривимірних комп’ютерних ігор необхідно дуже уважно ставитись до написання графічних складових, так як подібні сфери використання графіки потребуюсь досить багато ресурсів комп’ютера.

З розвитком комп’ютерних технологій та створенням більш продуктивних комп’ютерних складових, постійно з’являються нові можливості для створення якісніших та більш продуктивних програм та систем. Комп’ютерна графіка не стала винятком. Постійно з’являються нові відео прискорювачі які надають більше потужності та ресурсів для розробки комп’ютерної графіки. Паралельно з ростом потужності комп’ютерних складових розвивались і графічні технології, що дозволяє створювати більш якісну та візуально кращу комп’ютерну графіку.

При використанні багатоядерних процесорів є необхідність у розробці системи на основі паралельної архітектури. Використання всіх процесорів системи - як графічного (ГП), так і центрального (ЦП) - відкриває набагато більше можливостей у порівнянні з однопоточні движком на базі тільки ГП. Наприклад, використовуючи більше ядер ЦП, можна поліпшити візуальні ефекти, збільшивши кількість фізичних об'єктів, які використовуються в грі, а також домогтися більш реалістичного поведінки персонажів за рахунок реалізації просунутого штучного інтелекту (ШІ).

В процесі розробки саме складних, ресурсоємких графічних складових часто нехтуюсь якістю продукту намагаючись як умова швидше закінчити розробку та отримати результат. Це є одна з проблем сучасної розробки графічних складових.

Також при розробці подібних систем слід більше уваги приділити саме проектуванню майбутньої системи. Від проектування залежить як и сам процес написання програмних елементів системи, так и продуктивність, якість, стабільність, та функціонал.

Мета і завдання дослідження.

Проектування та розробка системи, для створення елементів комп’ютерної графіки, яка може полегшити взаємодію розробника з ресурсами комп’ютера в процесі розробки. А також дозволяє скоріш досягнути необхідного результату.

Об’єкт дослідження – Дослідження методів полегшення розробки комп’ютерної графіки та підвищення її якості.

Предмет дослідження – Ефективне проектування та розробка системи, для досягнення ліпшого результату при створенні елементів комп’ютерної графіки.

РОЗДІЛ 1
ОГЛЯД СУЧАСНИХ ТЕХНОЛОГІЙ ТА МЕТОДІВ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕННОЇ ЗАДАЧІ

1.1 Огляд методів і засобів розробки комп’ютерної графіки

Один із розповсюджених напрямів використання персонального комп’ютера є комп’ютерна графіка. Зараз тривимірні зображення можна побачити скрізь, починаючи від комп'ютерних ігор і закінчуючи системами Моделювання в реальному часі. Раніше, коли тривимірна графіка існувала тільки на суперкомп'ютерах, не існувало єдиного стандарту в галузі графіки. Всі програми писалися з нуля або з використанням накопиченого досвіду, але в кожній програмі реалізовувалися свої методи для відображення графічної інформації. З приходом потужних процесорів і графічних прискорювачів тривимірна графіка стала реальністю для персональних комп'ютерів [20].

Не дивлячись на те, що для роботи з комп’ютерною графікою існує маса класів програмного забезпечення, розрізняють усього 3 види комп’ютерної графіки, це растрова графіка, векторна графіка і фрактальна графіка. Вони відрізняються принципами формування зображення при відображенні на екрані монітора або при друці на папері.

Растрову графіку використовувати при розробці електронних (мультимедійних) і поліграфічних видань. Ілюстрації, виконані засобами растрової графіки, рідко створюють вручну за допомогою комп’ютерних програм. Частіше для цієї мети використовують скановані ілюстрації, підготовлені художником на папері, або фотографії. В останній час для вводу растрових зображень в комп’ютер широко використовують цифрові фото і відеокамери.

Відповідно, більшість графічних редакторів, які призначені для роботи з растровими ілюстраціями, орієнтовані не стільки на створення зображення, скільки на їх обробку. В Інтернеті поки що використовується тільки растрові ілюстрації.

У растрового зображення є кілька характеристик. Для фото найважливішими є: дозвіл, розмір і колірна модель. Іноді розмір також називають дозволом і тому відбувається плутанина, щоб цього не відбувалося, потрібно чітко уявляти про що йде мова і «дивитися по контексту» - розмір вимірюється в Мп (мегапіксехах), а розширення - dpi або ppi.

Дозвіл - це кількість пікселів на дюйм (ppi - pixel per inch) для опису відображення на екрані або кількість точок на дюйм (dpi - dot per inch) для друку зображень. Існує кілька усталених правил: для публікації зображення в мережі Інтернет використовують дозвіл 72ppi, а для друку - 300dpi (ppi).

Розмір - загальна кількість пікселів в зображенні, зазвичай вимірюється в Мп (мегапікселя), це всього лише результат множення кількості пікселів по висоті на кількість пікселів по ширині зображення. Тобто, якщо величина фотографії 2000x1500, то її розміром буде 2000 * 1500 = 3 000 000 пікселів або 3Мп.

Колірна модель - характеристика зображення, що описує його подання на основі колірних каналів. Мені відомо 4 колірні моделі - RGB (червоний, зелений і синій канали), CMYK (блакитний, пурпурний, жовтий і чорний), LAB («светлота», червоно-зелений і синьо-жовтий) і Grayscale (Відтінки сірого).

Переваги растрової графіки:

– Можливість відтворення зображень будь-якого рівня складності. Кількість деталей, відтворених на зображенні багато в чому залежить від кількості пікселів.
– Точна передача колірних переходів.
– Наявність безлічі програм для відображення і редагування растрової графіки. Абсолютна більшість програм підтримують однакові формати файлів растрової графіки. Растрове уявлення, мабуть, самий «старий» спосіб зберігання цифрових зображень.

Недоліки растрової графіки:

– Великий розмір файлу. Фактично для кожного пікселя доводиться зберігати інформацію про його координатах і кольорі.
– Неможливість масштабування (в часності, збільшення) зображення без втрати якості.

Не дивлячись на уявну простоту представлення растрової графіки, її форматів існує «вагон і маленький візок»! І їх кількість продовжує змінюватися - якісь формати застарівають, якісь тільки починають розроблятися. Далі будуть описані декілька найрозповсюдженіших форматів:

GIF (Graphics Interchange Format) якщо не найпопулярніший, то вже точно другою за популярністю формат растрової графіки, що використовується для публікації зображень в WEB. Хоча деякі називають його «застарілим», альтернативи йому поки немає. Справа в тому, що це фактично єдиний растровий формат графіки, що підтримує анімацію. На даний момент існує ще 2 формату з анімацією - MNG і APNG, які, ймовірно і замінять GIF, але

поки їх підтримують далеко не всі браузери і графічні редактори. До переваг GIF можна віднести: невеликий розмір зображення, підтримка прозорості, можливість створення покадрової анімації. До недоліків можна віднести невелику кількість можливих кольорів (до 256) і відсутність передачі напівпрозорості. GIF зараз широко використовується для створення анімованих рекламних банерів.

PNG (Portable Network Graphics) - ще один формат растрової графіки, що підтримує прозорість, причому не тільки звичайної прозорості, як GIF, але і напівпрозорість - плавний перехід кольору в прозору область. Метою створення PNG якраз і була заміна GIF, так як компанія CompuServe - розробник формату GIF в 1995 році на 10 років запатентувала алгоритм стиснення, використаний при створенні gif-картинок, що унеможливлювало безкоштовне використання даного формату в комерційних проектах.

Переваги PNG:

- Можливість створення кольорового зображення з колірними переходами і півтонами.
- Збереження графічної інформації за допомогою алгоритму стиснення без втрат.
- Можливість використання альфа-каналів, тобто, просто кажучи, прозорості та, більш того - напівпрозорості, що дозволяє створювати плавні переходи кольору в прозору область.

Недоліки:

- Неможливість створення анімованого зображення
- Неоднозначне «розуміння» прозорості формату PNG інтернет-браузерами. Деякі браузери, в основному застарілі версії, відмовляються відображати прозорі області зображення формату PNG і зафарбовують їх сірим кольором.

JPEG (Joint Photographic Experts Group - назва розробника) - найпоширеніший формат растрової графіки (принаймні - в Інтернеті). JPEG - приклад використання алгоритмів стиснення «з втратами» або, по-іншому, «спотворює стиснення», він найбільш підходить для зберігання картин, фотографій та інших реалістичних зображень з плавними колірними переходами, але зате практично не придатний для креслень і схем, тобто для зображень з різкими переходами - алгоритм стиснення буде утворювати помітні артефакти в місцях різкого контрасту.

Не рекомендується зберігати в цьому форматі проміжні варіанти роботи - кожне «пересохраненіе» буде вести до незворотної втрати частини інформації. Алгоритм

стиснення, який використовується в цьому форматі (lossy compress) заснований на «усередненні» кольору поруч стоять пікселів.

JPEG не підтримує роботу з альфа-каналами, тобто не може містити прозорі пікселі, але дозволяє зберегти в файлі відсічний контур.

Крім загальних форматів растрової графіки (GIF, JPEG, PNG та інш.), Які «підтримуються» всіма графічними редакторами та переглядачами зображень, існують «рідні» формати майже кожного редактора, які можна відкрити тільки програмою, в якій вони були зроблені, наприклад, формат.PSD програми Adobe Photoshop. При обробці фотографій, растрових ілюстрацій і розробці дизайну, проміжні варіанти слід зберігати в таких форматах і тільки фінальні версії переводити JPEG. Це потрібно для того, щоб можна було зберігати результати роботи без втрати інформації та в будь-який момент внести зміни в зображення або проект.

Програмні засоби для роботи з векторною графікою призначені, в першу чергу, для створення ілюстрацій і в меншій мірі для їх обробки. Такі засоби широко використовують в рекламних агентствах, дизайнерських бюро, редакціях і виданнях. Оформлювальні роботи, основані на застосуванні шрифтів і простих геометричних елементів, вирішуються засобами векторної графіки набагато простіше. Існують приклади високохудожніх творів, створених засобами векторної графіки, але вони скоріше виключення, ніж правило, оскільки художня підготовка ілюстрацій засобами векторної графіки надзвичайно складна.

Для створення зображення векторного формату, відображуваного на растровому пристрої, використовуються перетворювачі математичного опису графічних примітивів в растрове зображення для відображення на матричних моніторах, ці перетворювачі або реалізовані програмно, або апаратні перетворювачі.

Переваги векторного способу створення графіки над растовим:

- Обсяг даних, яку він обіймав описовою частиною, не залежить від реальної величини об'єкта, що дозволяє, використовуючи мінімальну кількість інформації, описати як завгодно великий об'єкт файлом мінімального розміру. Наприклад, опис окружності довільного радіуса вимагає завдання тільки 3 чисел, не рахуючи атрибутів.
- У зв'язку з тим, що інформація про об'єкт зберігається в описовій формі, можна нескінченно збільшити графічний примітив при виведенні на графічний пристрій, наприклад, дугу кола, і вона залишиться при будь-якому збільшенні гладкою. З іншого боку, якщо крива представлена у вигляді ламаної лінії, збільшення покаже, що вона насправді не крива.

- Параметри об'єктів зберігаються і можуть бути легко змінені. Також це означає що переміщення, масштабування, обертання, заповнення та т. Д. Не погіршує якості малюнка. Більш того, зазвичай вказують розміри в апаратно-незалежних, які ведуть до найкращої можливої растеризації на растрових пристроях.
- При збільшенні або зменшенні об'єктів товщина ліній може бути задана постійною величиною, незалежною від реальної площі зображуваної фігури.

До основних недоліків векторної графіки можна віднести:

- Не кожна графічна сцена може бути легко зображена в векторному вигляді - для подібного оригінального зображення може знадобитися опис дуже великої кількості примітивів з високою складністю, що негативно впливає на кількість пам'яті, займаної зображенням і на час, необхідний для перетворення його в растровий формат для графічного виведення (відтворення або растеризації).
- Переклад векторної графіки в растрове зображення досить простий. Але зворотний шлях, як правило, складний - цей процес називають трасуванням растра, і часто вимагає значних обчислювальних потужностей і процесорного часу, і не завжди забезпечує високу якість отриманого векторного малюнка.
- При цьому специфікації векторних форматів (і, відповідно, рендерер векторної графіки) набагато складніше таких для растрової графіки.
- Перевага векторної картинки - масштабованість - пропадає, коли векторний формат відображається в растрове дозвіл з особливо малими дозволами графіки (наприклад, іконки 32 × 32 або 16 × 16).

Програмні засоби для роботи з фрактальною графікою призначені для автоматичної генерації зображення шляхом математичних розрахунків. Створення фрактальної художньої композиції полягає не в рисуванні чи оформленні, а в програмуванні. Фрактальну графіку рідко використовують для створення друкованих або електронних документів, але її часто використовують у розважальних програмах.

Якщо мова йде про високопродуктивну комп’ютерну графіку, з багатьма динамічними елементами такими як, сцена, моделі об’єктів, світло, тіні, камера та інші, для їх створення не достатньо графічного редактору. Для створення високопродуктивних графічних елементів потрібне написання програми на одній з мов, з використанням одної з технологій або систем для створення комп’ютерної графіки.

Сучасні системи для комп’ютерної графіки зазвичай складаються з цілих пакетів програмних продуктів, та дозволяють створювати великі проекти з використання

комп’ютерної графіки іноді навіть без написання коду, за допомогою тільки графічного інтересу редактора. Такі графічні редактори зазвичай потребують багатьох ресурсів комп’ютера як при створенні графіки, так и при її відображенні. Для створення графіки в подібних редакторах не завжди необхідно мати досвід в математиці або в програмуванні, це далеко не завжди добре. Продукти створенні в подібних редакторах без допомоги математики та програмування часто можуть буди не дуже якісними, та потребувати дуже багато ресурсів комп’ютера для відображення не великого змісту.

Якщо необхідно створити якісну, та продуктивну програму з використанням комп’ютерної графіки, то буде складно створити її без використання якоїсь мови програмування та хоча б одну з основних технологій для вирішення подібних задач. Деякими з основних API які допомагають створити якісну комп’ютерну графіку – є OpenGL, Vulkan, DirectX, Mantle, Metal. Але створення графічних елементів з використанням однієї з цих технологій не є простою задачею, та не кожен програміст може її вирішити, тому що ці технології написані на низкорівневих мовах програмування з більшим контролем над ресурсами комп’ютера ніж в прикладному програмуванні. Для вирішення цієї проблеми зазвичай створюються системи(API, Engine), які надають розробнику деякий набір інструментів, яких дозволяє йому створювати елементи комп’ютерної графіки на високому рівні програмування, та іноді навіть не знаючи як саме, або що саме роблять ті інструменти які він використовує.

Такий підхід розділяє задачу на маленькі під задачі, та дозволяє створювати елементи продукту свого різними розробниками. Хтось створює системи для розробки графічних елементів, інші люди використовують створену систему для розробки невеличких елементів, потім ці елементи об’єднуються на одній графічній сцені, далі їм задається якась поведінка, все це збирається в одну програму яка вже и буде продуктом.

1.2 Аналіз існуючих технологій, які використовуються при розробці систем для створення комп’ютерної графіки

З розвитком комп’ютерних технологій та створенням більш продуктивних комп’ютерних складових, постійно з’являються нові можливості для створення якісніших та більш продуктивних програм та систем. Комп’ютерна графіка не стала винятком. Постійно з’являються нові відео прискорювачі які надають більше потужності та ресурсів для розробки комп’ютерної графіки. Паралельно з ростом потужності комп’ютерних

складових розвивались і графічні технології, що дозволяє створювати більш якісну та візуально кращу комп’ютерну графіку. Основними з таких технологій є - OpenGL, Vulkan, Mantle, DirectX. Вони використовуються для різних завдань, та на різних платформах [21]. Далі ці графічні технології будуть розглянуті більш детально.

OpenGL (Open Graphics Library — відкрита графічна бібліотека) — специфікація, що визначає незалежний від мови програмування крос-платформовий програмний інтерфейс (API) для написання застосунків, що використовують 2D та 3D комп'ютерну графіку. Цей інтерфейс містить понад 250 функцій, які можуть використовуватися для малювання складних тривимірних сцен з простих примітивів. Широко застосовується індустрією комп'ютерних ігор і віртуальної реальності, у графічних інтерфейсах (Compiz, Clutter), при візуалізації наукових даних, в системах автоматизованого проектування тощо [21].

На базовому рівні, OpenGL — це всього лише специфікація, тобто, — просто документ, який описує набір функцій та їх точну поведінку. На основі цих специфікацій виробники апаратного забезпечення створюють реалізації — бібліотеки функцій, які відповідають заявленій в OpenGL специфікації. Ці реалізації проектуються для того, щоб при можливості використовувати можливості апаратного забезпечення. Коли апаратне прискорення не допускається, виконання функцій здійснюється за допомогою програмного забезпечення. Виробники повинні пройти спеціальні тести на відповідність, перш, ніж їхню реалізацію класифікуватимуть, як реалізацію OpenGL [16]. Таким чином, розробникам програмного забезпечення необхідно всього лиш навчитися використовувати описані у специфікації функції, і лишити їхню реалізацію за розробниками апаратного забезпечення.

Ефективні реалізації OpenGL існують для операційних систем Linux, MacOS X, Microsoft Windows та багатьох UNIX-подібних ОС, а також для таких ігрових боксів, як Sony PlayStation 3. Різні програмні реалізації OpenGL існують для платформ, виробники яких не підтримують дану специфікацію.

Vulkan - багатоплатформовий API для 2D і 3D-графіки, вперше представлений Khronos Group в рамках конференції GDC 2015.

Vulkan API спочатку був відомий як «нове покоління OpenGL» або просто «glNext», але після анонса компанія відмовилася від цих назв на користь назви Vulkan. Як і OpenGL [15], Vulkan дозволяє з високою продуктивністю відображати в реальному часі різні додатки з 3D-графікою, такі як ігри або інтерактивні книги на всіх платформах, а також забезпечує більш високу продуктивність і менше навантаження на процесор, аналогічно Direct3D 12, Metal і Mantle. Vulkan заснований на технологіях AMD в Mantle.

Метою Vulkan було перевершити інші API, включаючи його попередника OpenGL, в частині зниження накладних витрат, підвищення ступеня прямого контролю над GPU і зменшення навантаження на CPU. Vulkan має передбачувані переваги:

- OpenGL використовує мову для написання шейдеров GLSL. Це змушує кожного виробника OpenGL драйвера реалізувати свій власний компілятор для GLSL, який працює під час виконання графічного додатку, компілюючи шейдерниє програми в виконуваний код цільової платформи. Vulkan натомість пропонує проміжний двійковий формат SPIR-V (Standard Portable Intermediate Representation), аналогічний бінарного формату в який компілюються HLSL-шейдери на платформі DirectX. Це знімає тягар з постачальників драйверів, дозволяючи компілювати шейдери на етапі розробки. Також дозволяє розробникам додатків писати шейдери на інших мовах, крім GLSL.
- Багатоплатформовий API підтримується на мобільних пристроях і високопродуктивних відкритих.
- Покращена підтримка сучасних систем, що використовують багатопоточність.
- Зниження навантаження на центральний процесор в ситуаціях, коли він є недостатньо продуктивним, що дозволяє досягти більш високої пропускної здатності для GPU-обчислень і візуалізації.

Mantle - специфікація низкорівневого API, розроблена компанією AMD в якості альтернативи DirectX і OpenGL. В даний час підтримується лише графічними процесорами AMD архітектури GCN (Graphics Core Next, Наступне графічне ядро), хоча є припущення, що інші виробники GPU могли б реалізувати її в майбутньому.

Основні переваги цієї технології:

- AMD стверджує, що Mantle може обробляти до дев'яти разів більше запитів на відображення в секунду, ніж зіставні API, за рахунок зниження навантаження на процесор.
- Більш точний контроль над апаратними засобами.
- Всі апаратні можливості надаються через API.
- Можливі нові методи візуалізації.
- Прямий доступ до пам'яті GPU.
- Сумісність з DirectX HLSL для спрощення портування.
- Незалежність розробників ігор від існуючих драйверів GPU AMD.

- Спрощення розробки крос-платформних ігор для ПК і консолей (теоретично, всі низькорівневі запити, написані для ПК, будуть зрозумілі новим APU Playstation 4 і Xbox One).
- Приріст продуктивності в порівнянні з більш високорівневими API, такими як DirectX і OpenGL.ї

  Але як вже було сказано ця технологія підтримується лише графічними процесорами AMD архітектури GCN.

DirectX невідмінно від OpenGL та Vulkan є технологією яка має підтримку тільки у системі Windows. По суті DirectX є набір API, які дозволяють створювати розробника гри і мультимедіа. Крім цього, DirectX служить для обробки клавіатури, миші, джойстика, а також для мережевого повідомлення. По суті DirectX API це технологія, яка об’єднує в собі декілька модулів, основними з яких є:

- DirectX Graphics, набір інтерфейсів, раніше (до версії 8.0) ділилися на:
- DirectInput: інтерфейс, використовуваний для обробки даних, що надходять з клавіатури, миші, джойстика і інших ігрових контролерів.
- DirectPlay: інтерфейс мережевої комунікації ігор.
- DirectSound: інтерфейс низькорівневої роботи зі звуком (формату Wave)
- DirectMusic: інтерфейс відтворення музики у форматах Microsoft.
- DirectShow: інтерфейс, використовуваний для введення / виведення аудіо та / або відео даних.
- DirectSetup: частина, відповідальна за установку DirectX.
- DirectX Media Objects: реалізує функціональну підтримку потокових об'єктів (наприклад, кодувальники / декодувальник)
- Direct2D: інтерфейс виведення двомірної графіки

Багато сучасні пристрої мають DirectX-сумісні драйвери, іншими словами, користувач повинен встановити DirectX для використання всіх можливостей пристрою. На даний момент ця графічна технологія може використовуватись лише для створення графічного продукту для комп’ютерів з операційної системою Windows.

Таким чином, при розробці системи для створення елементів комп’ютерно графіки, з розглянутих технологій найбільш відповідна с точки зору продуктивності та мультиплатформеності є саме OpenGL та Vulkan технології. Завдяки їх розповсюдженості

та підтримці на всіх сучасних графічних прискорювачах вони дадуть можливість розробляти графічні прогарами, які зможуть виконуватися на багатьох пристроях.

Також серед переваг обраних технології слід звернути увагу на низкорівневість, завдяки контролю над багатьма елементами розробки, що надасть кращій контроль над ресурсами комп'ютера. Завдяки цьому з'явитеся можливість створювати високопродуктивну графічну складову.

1.3 Постановка наукової задачі та обґрунтування методики досліджень

Результати проведеного аналізу моделей, методів і інструментальних засобів було виявлено декілька схожих систем для створення елементів комп'ютерної графіки але в багатьох з них не вирішена проблема саме невеликого розміру самої системи, та її доступності. А саме багато подібних систем розробляються компаніями для власного використання, або для продажу таких систем для використання іншими особами.

У обробленій літературі не були розглянуті завдання, пов'язані з використання технологій OpenGL та Vulkan для розробки системи, яка допоможе створювати графічні елементи без глибоких пізнань елементів комп'ютера. Також в літературі не було приведено способів використання двох або більш технологій під час створення подібних систем.

Основними задачами магістерської роботи слід виділити наступні пункти:

1. Проектування та розробка системи, яка б дозволяла не поглиблюючись в пізнання структури комп'ютера та його складових, розробляти графічні елементи та продуктивні програми, які відображають ці елементи.
2. Розроблена система має співпадати з усіма нормами заданими для систем такого типу.
3. Для проведення експерименту з оцінки якості розробленої системи знадобиться розробити декілька тестів, за результатами котрих можна буде побачити ефективність розробленої системи.

РОЗДІЛ 2
АНАЛІЗ ПРЕДМЕТА ДОСЛІДЖЕННЯ ТА МЕТОДІВ ДОСЯГНЕНЯ ПОСТАВЛЕННИХ ЗАВДАНЬ

2.1 Пристрій відеоадаптеру

Відеоадаптер перетворює графічний образ, що зберігається як вміст пам'яті комп'ютера (або самого адаптера), у форму, придатну для подальшого виведення на екран монітора. Перші монітори, побудовані на електронно-променевих трубках, працювали по телевізійному принципом сканування екрану електронним променем, і для відображення був потрібний відеосигнал, що генерується відкритий [3].

Однак ця базова функція, залишаючись потрібної і затребуваною, пішла в тінь, переставши визначати рівень можливостей формування зображення - якість відеосигналу (чіткість зображення) дуже мало пов'язано з ціною і технічним рівнем сучасної відеокарти. В першу чергу, зараз під графічним адаптером розуміють пристрій з графічним процесором - графічний прискорювач, який і займається формуванням самого графічного образу. Сучасні відеокарти не обмежуються простим виведенням зображення, вони мають вбудований графічний процесор, який може виробляти додаткову обробку, знімаючи цю задачу з центрального процесора комп'ютера. Наприклад, всі сучасні відеокарти здійснюють рендеринг графічного конвеєра OpenGL і DirectX на апаратному рівні. Останнім часом також має місце тенденція використовувати обчислювальні можливості графічного процесора для вирішення неграфічних завдань [15].

Зазвичай відеокарта виконана у вигляді друкованої плати (плата розширення) і вставляється в роз'єм розширення, універсальний або спеціалізований (AGP, PCI Express). Також широко поширені і вбудовані (інтегровані) в системну плату відеокарти - як у вигляді окремого чіпа, так і в якості складової частини північного моста чіпсета або в процесор; в цьому випадку пристрій, строго кажучи, не може бути названо відеокартою.

Сучасна відеокарта складається з наступних частин:

Графічний процесор (Graphics processing unit (GPU)) займається розрахунками зображення, що виводиться, звільняючи від цього обов'язку центральний процесор, проводить розрахунки для обробки команд тривимірної графіки [5]. Є основою графічної плати, саме від нього залежать швидкодія і можливості всього пристрою. Сучасні графічні

процесори по складності мало чим поступаються центральному процесору комп'ютера, і часто перевершують його як по числу транзисторів, так і по обчислювальної потужності, завдяки великому числу універсальних обчислювальних блоків. Однак архітектура GPU минулого покоління зазвичай передбачає наявність декількох блоків обробки інформації, а саме: блок обробки 2D-графіки, блок обробки 3D-графіки, у свою чергу, зазвичай розділяється на геометричне ядро (плюс кеш вершин) і блок растеризації (плюс кеш текстур).

Видеоконтроллер, який відповідає за формування зображення в відеопам'яті, дає команди RAMDAC на формування сигналів розгортки для монітора і здійснює обробку запитів центрального процесора. Крім цього, зазвичай присутні контролер зовнішньої шини даних (наприклад, PCI або AGP), контроллер внутрішньої шини даних і контроллер відеопам'яті. Ширина внутрішньої шини і шини відеопам'яті зазвичай більше, ніж зовнішньої (64, 128 або 256 розрядів проти 16 або 32), в багато відеоконтролерів вбудовується ще і RAMDAC. Сучасні графічні адаптери (AMD, nVidia) зазвичай мають не менше двох видеоконтроллеров, що працюють незалежно один від одного і керуючих одночасно одним або декількома дисплеями кожен [10].

Відео-ПЗП (Video ROM) - постійний запам'ятовуючий пристрій (ПЗП), в яке записано BIOS відеокарти, екранні шрифти, службові таблиці. ПЗП не використовується видеоконтроллером безпосередньо - до нього звертається тільки центральний процесор. BIOS забезпечує ініціалізацію і роботу відеокарти до завантаження основної операційної системи, задає все низькорівневі параметри відеокарти, в тому числі робочі частоти і живлять напруги графічного процесора і відеопам'яті, таймінги пам'яті. Також VBIOS містить системні дані, які можуть читатися і інтерпретуватися відеодрайвером в процесі роботи (в залежності від застосовуваного методу поділу відповідальності між драйвером і BIOS). На багатьох сучасних картах встановлюються електрично перепрограмовані ПЗП (EEPROM, Flash ROM), що допускають перезапис відео-BIOS самим користувачем за допомогою спеціальної програми.

Відео-ОЗП. Відеопам'ять виконує функцію кадрового буфера, в якому зберігається зображення, що генерується і постійно змінюване графічним процесором і що виводиться на екран монітора (або декількох моніторів). У відеопам'яті зберігаються також проміжні невидимі на екрані елементи зображення і інші дані. Відеопам'ять буває декількох типів, що розрізняються за швидкістю доступу і робочій частоті. Сучасні відеокарти комплектуються пам'яттю типу DDR, GDDR2, GDDR3, GDDR4, GDDR5 і HBM. Слід також мати на увазі, що, крім відеопам'яті, що знаходиться на відеокарті, сучасні графічні

процесори зазвичай використовують у своїй роботі частину загальної системної пам'яті комп'ютера, прямий доступ до якої організовується драйвером відеоадаптера через шину AGP або PCIE. У разі використання архітектури Uniform Memory Access в якості відеопам'яті використовується частина системної пам'яті комп'ютера [17].

Цифро-аналоговий перетворювач (ЦАП; RAMDAC - Random Access Memory Digital-to-Analog Converter) служить для перетворення зображення, формованого видеоконтроллером, в рівні інтенсивності кольору, що подаються на аналоговий монітор. Можливий діапазон кольоровості зображення визначається тільки параметрами RAMDAC. Найчастіше RAMDAC має чотири основні блоки: три цифро-аналогових перетворювача, по одному на кожний колірний канал (червоний, зелений, синій - RGB), і SRAM для зберігання даних про гамма-корекції. Більшість ЦАП мають розрядність 8 біт на канал - виходить по 256 рівнів яскравості на кожен основний колір, що в сумі дає 16,7 млн кольорів (а за рахунок гамма-корекції є можливість відображати вихідні 16,7 млн квітів в набагато більшу колірний простір) . Деякі RAMDAC мають розрядність по кожному каналу 10 біт (1024 рівні яскравості), що дозволяє відразу відображати більше 1 млрд квітів, але ця можливість практично не використовується. Для підтримки другого монітора часто встановлюють другий ЦАП.

Відеоадаптери MDA, Hercules, EGA і CGA оснащувалися 9-контактним роз'ємом типу D-Sub. Зрідка також був присутній коаксіальний роз'єм Composite Video, що дозволяє вивести чорно-біле зображення на телевізійний приймач або монітор, оснащений НЧ-відеовходом. Відеоадаптери VGA і більш пізні зазвичай мали всього один роз'єм VGA (15-контактний D-Sub). Зрідка ранні версії VGA-адаптерів мали також роз'єм попереднього покоління (9-контактний) для сумісності зі старими моніторами. Вибір робочого виходу задавався перемикачами на платі відеоадаптера. В даний час плати оснащують роз'ємами DVI або HDMI, або DisplayPort в кількості від одного до трьох.

Порти DVI і HDMI є еволюційними стадіями розвитку стандарту передачі відеосигналу, тому для з'єднання пристроїв з цими типами портів можливе використання перехідників (роз'єм DVI до гнізда D-Sub - аналоговий сигнал, роз'єм HDMI до гнізда DVI-D - цифровий сигнал, який не підтримує технічні засоби захисту авторських прав, тому без можливості передачі багатоканального звуку і високоякісного зображення). Порт DVI-I також включає аналогові сигнали, що дозволяють підключити монітор через перехідник на старий роз'єм D-SUB (DVI-D не дозволяє цього зробити). DisplayPort дозволяє підключати до чотирьох пристроїв, в тому числі аудіо, USB-концентратори і інші пристрої введення-виведення.

Система охолодження призначена для збереження температурного режиму відеопроцесора і (найчастіше) відеопам'яті в допустимих межах. Також правильна і повнофункціональна робота сучасного графічного адаптера забезпечується за допомогою відеодрайвера - спеціального програмного забезпечення, що поставляється виробником відеокарти і завантажується в процесі запуску операційної системи. Відеодрайвер виконує функції інтерфейсу між системою з запущеними в ній додатками і відеоадаптером. Так само, як і відео-BIOS, відеодрайвер організовує і програмно контролює роботу всіх частин відеоадаптера через спеціальні регістри управління, доступ до яких відбувається через відповідну шину.

Відкрите для комп'ютерів існують в одному з двох розмірів. Деякі відеокарти нестандартного розміру, і, таким чином, класифіковані як низькопрофільні. Профілі відеокарт засновані тільки на ширині, низькопрофільні карти займають менше ширини щілини PCIe.

Основна перешкода до підвищення швидкодії відеосистеми - це інтерфейс передачі даних, до якого підключений відеоадаптер. Як би не був швидкий процесор відеоадаптера, велика частина його можливостей залишиться незадіяною, якщо не будуть забезпечені відповідні канали обміну інформацією між ним, центральним процесором, оперативною пам'яттю комп'ютера і додатковими відеопристроїв. Основним каналом передачі даних є, звичайно, інтерфейсна шина материнської плати, через яку забезпечується обмін даними з центральним процесором і оперативною пам'яттю.

2.2 Комп’ютерна геометрія

2.2.1 Двовимірні перетворення

Комп'ютерна геометрія є математичний апарат, покладений в основу комп'ютерної графіки. У свою чергу, основу комп'ютерної геометрії складають різні перетворення точок та ліній. При використанні машинної графіки можна за бажанням змінювати масштаб зображення, обертати його, зміщувати і трансформувати для поліпшення наочності перспективного зображення [1]. Всі ці перетворення можна виконати на основі математичних методів, які будуть розглядати далі.

Перетворення, як і комп'ютерну геометрію, поділяють на двовимірні (або перетворення на площині) і тривимірні (або просторові). Спочатку слід ознайомитись с перетворенням на площині [6].

Для початку слід зауважити, що точки на площині задаються за допомогою двох її координат. Таким чином, геометрично кожна точка задається значеннями координат вектора щодо обраної системи координат. Координати точок можна розглядати як елементи матриці [x, y]. У вигляді вектор-рядки або вектор-стовпці. Положенням цих точок керують шляхом перетворення матриці. Точки на площині xy можна перенести в нові позиції шляхом додавання до координат цих точок констант переносу:

$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} + \begin{bmatrix} a & b \end{bmatrix} = \begin{bmatrix} x+a & y+b \end{bmatrix} \tag{2.1}$$

Таким чином, для переміщення точки на площині треба до матриці її координат додати матрицю коефіцієнтів перетворення.

Результати матричного множення матриці [x, y], яка визначає точку P, і матриці перетворень 2x2 загального вигляду:

$$\begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} (ax+cy)(bx+dy) \end{bmatrix} = \begin{bmatrix} x^* & y^* \end{bmatrix} \tag{2.2}$$

Проведемо аналіз отриманих результатів, розглядаючи x * і y * як перетворені координати. Для цього досліджуємо кілька окремих випадків.

Випадок, коли a = d = 1 і c = b = 0. Матриця перетворень призводить до матриці, ідентичною вихідної:

$$\begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} (1x+0y)(0x+1y) \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} x^* & y^* \end{bmatrix} \tag{2.3}$$

При цьому змін координат точки P не відбувається.

Якщо тепер d = 1, b = c = 0, a = const, то:

$$\begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} (ax+0y)(0x+1y) \end{bmatrix} = \begin{bmatrix} ax & y \end{bmatrix} = \begin{bmatrix} x^* & y^* \end{bmatrix} \tag{2.4}$$

Як видно, це призводить до зміни масштабу в напрямку x, так як x * = ax. Отже, дане матричне перетворення еквівалентно переміщенню вихідної точки в напрямку x.

Тепер b = c = 0, тобто.

$$\begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} = \begin{bmatrix} (ax+0y)(0x+dy) \end{bmatrix} = \begin{bmatrix} ax & dy \end{bmatrix} = \begin{bmatrix} x^* & y^* \end{bmatrix} \quad (2.5)$$

В результаті було отримано зміну масштабів в напрямках x і y. Якщо a ≠ d, то переміщення уздовж осей неоднакові. Якщо a = d> 1, то має місце збільшення масштабу координат точки P. Якщо 0 <a = d <1, то буде мати місце зменшення масштабу координат точки P. Якщо a або (i) d негативні, то відбувається відображення координат точок. Тобто, поклавши b = c = 0; d = 1 і a = -1, тоді.

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} (-1x+0y)(0x+1y) \end{bmatrix} = \begin{bmatrix} -x & y \end{bmatrix} = \begin{bmatrix} x^* & y^* \end{bmatrix} \quad (2.6)$$

Сталося відображення точки відносно осі y. У разі b = c = 0, a = 1, d = -1, відображення відбувається відносно осі x. Якщо b = c = 0, a = d <0, то відображення відбуватиметься щодо початку координат.

Відображення і зміна масштабу викликають тільки діагональні елементи матриці перетворення.

Тепер випадок, коли a = d = 1, а з = 0, тобто.

$$\begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} x & (bx+y) \end{bmatrix} = \begin{bmatrix} x^* & y^* \end{bmatrix} \quad (2.7)$$

Координата x точки P не змінюється, в той час як y * лінійно залежить від початкових координат. Цей ефект називається зрушенням. Аналогічно, коли a = d = 1, b = 0, перетворення здійснює зрушення пропорційно координаті y.

Перетворення загального вигляду, застосоване до початку координат, не приведе до зміни координат точки (0,0). Отже, початок координат інваріантної при загальному перетворенні. Це обмеження долається за рахунок використання однорідних координат.

Якщо піддати загальному перетворенню різні геометричні фігури, то можна встановити, що паралельні прямі перетворюються в паралельні прямі, середина відрізка - до середини відрізка, паралелограм - в паралелограм, точка перетину двох ліній - в точку перетину перетвореної пари ліній.

Аналіз перетворення одиничного квадрату.

Чотири вектора положення точок одиничного квадрата з одним кутом на початку координат записуються у наступному вигляді.

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{matrix} \rightarrow A \\ \rightarrow B \\ \rightarrow C \\ \rightarrow D \end{matrix}$$

Застосування загального матричного перетворення.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

До одиничному квадрату приводить наступна матриця.

$$\begin{matrix} A \rightarrow \\ B \rightarrow \\ C \rightarrow \\ D \rightarrow \end{matrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ a & b \\ a+c & b+d \\ c & d \end{bmatrix} \begin{matrix} \leftarrow A^* \\ \leftarrow B^* \\ \leftarrow C^* \\ \leftarrow D^* \end{matrix}$$

На рис. 2.1 відображенні ітог перетворення одиничного квадрату.

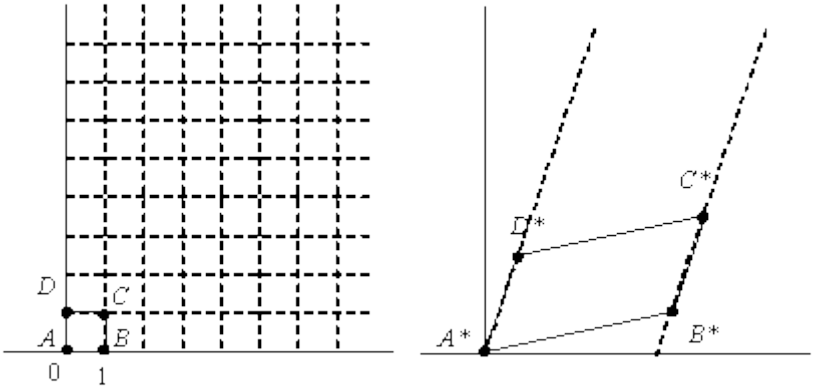

Рисунок 2.1 Перетворення одиничного квадрата

З отриманого співвідношення можна зробити висновок, що координати B * визначаються першим рядком матриці перетворення, а координати D * другим рядком цієї матриці. Таким чином, якщо координати точок B * і D * відомі, то загальна матриця перетворення визначена. Скористаємося цим властивістю для знаходження матриці перетворення для обертання на довільний кут [12].

Загальну матрицю 2x2, яка здійснює обертання фігури щодо початку координат, можна отримати з розгляду обертання одиничного квадрата навколо початку координат.

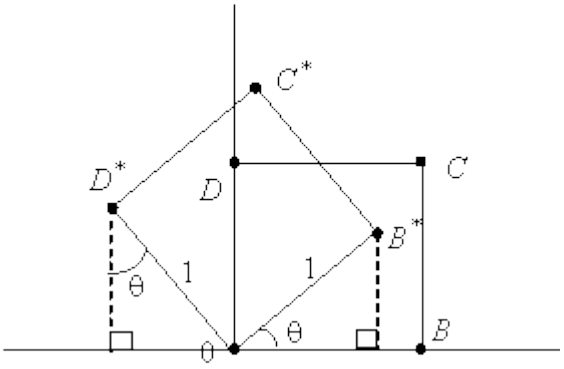

Рисунок 2.2. Обертання одиничного квадрата

Як випливає з рис. 2.2, точка B з координатами (1,0) перетворюється в точку B *, для якої x * = (1) cos q і y = (1) sin q, а точка D, що має координати (0,1) переходить в точку D * з координатами x * = (- 1) sin q і y * = (1) cos q.

Матриця перетворення загального вигляду записується наступним чином:

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Для окремих випадків. Поворот на 90 ° можна здійснити за допомогою матриці перетворення.

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Якщо використовувати матрицю координат вершин, то отримаємо, наприклад:

$$\begin{bmatrix} 3 & -1 \\ 4 & 1 \\ 2 & 1 \end{bmatrix} \bullet \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ -1 & 4 \\ -1 & 2 \end{bmatrix}$$

Поворот на 180 ° виходить за допомогою наступної матриці

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

У той час як чисте двовимірне обертання в площині xy здійснюється навколо осі, перпендикулярної до цієї площини, відображення визначається поворотом на 180 ° навколо осі, що лежить в площині xy.

Таке обертання навколо лінії y = x відбувається при використанні матриці

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Перетворені нові вирази визначаються співвідношенням

$$\begin{bmatrix} 8 & 1 \\ 7 & 3 \\ 6 & 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 8 \\ 3 & 7 \\ 2 & 6 \end{bmatrix}$$

Обертання навколо y = 0 виходить при використанні матриці.

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

2.2.2 Однорідні координати

Перетворення перенесення, масштабування і повороту записуються в матричної формі у вигляді

$$P^* = P + T$$

$$P^* = P \cdot S$$

$$P^* = P \cdot R$$

Очевидно, що перенесення, на відміну від масштабування і повороту, реалізується за допомогою складання. Це обумовлено тим, що вводити константи переносу всередину структури загальної матриці розміром 2x2 не представляється можливим. Бажаним є уявлення перетворень в єдиній формі - за допомогою множення матриць. Цю проблему можна вирішити за рахунок введення третьої компоненти в вектори точок [x y] та [x* y*]. Представляючи їх у вигляді [x y 1] і [x* y* 1]. Матриця перетворення після цього стає матрицею розміром 3x2:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ m & n \end{bmatrix}$$

Це необхідно, оскільки число стовпців в матриці, яка описує точку, має дорівнювати числу рядків в матриці перетворення для виконання операції множення матриць. Таким чином,

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ m & n \end{bmatrix} = \begin{bmatrix} x+m & y+n \end{bmatrix} = \begin{bmatrix} x^* & y^* \end{bmatrix}$$

звідки випливає, що константи т, п викликають зміщення відносно х і у. Оскільки матриця 3'2 не є квадратної, вона не має зворотної матриці. Ці труднощі можна обійти, доповнивши матрицю перетворення до квадратної розміром 3х3. наприклад,

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}$$

Слід зауважити, що третій компонент векторів положення точок не змінюється при додаванні третього стовпця до матриці перетворення. Використовуючи цю матрицю в співвідношенні, отримуємо перетворений вектор [х * у * 1]. Додавання третього елементу до вектору положення і третього стовпчика до матриці перетворення дозволяє виконати зміщення вектора положення. Третій елемент тут можна розглядати як додаткову координату вектора положення. Отже, вектор положення [х у 1] при впливі на нього матриці 3х3 стає вектором положення в загальному випадку виду [X Y H]. Представлене перетворення було виконано так, що

[X Y H] = [х * у * 1].

Перетворення, що має місце в тривимірному просторі, в нашому випадку обмежена площиною, оскільки Н = 1. Якщо, проте, третій стовпець

$$\begin{bmatrix} p \\ q \\ s \end{bmatrix}$$

матриці перетворення T розміру 3x3 відмінний від 0, то в результаті матричного перетворення отримаємо [x y 1] = [X Y H], де H ≠ 1. Площина, в якій тепер лежить перетворений вектор положення, знаходиться в тривимірному просторі. Однак зараз нас не цікавить те, що відбувається в тривимірному просторі. Отже, знайдені x * й y * отримані за допомогою пучка променів, що проходять через початок координат. Результат перетворень показаний на рис. 2.3.

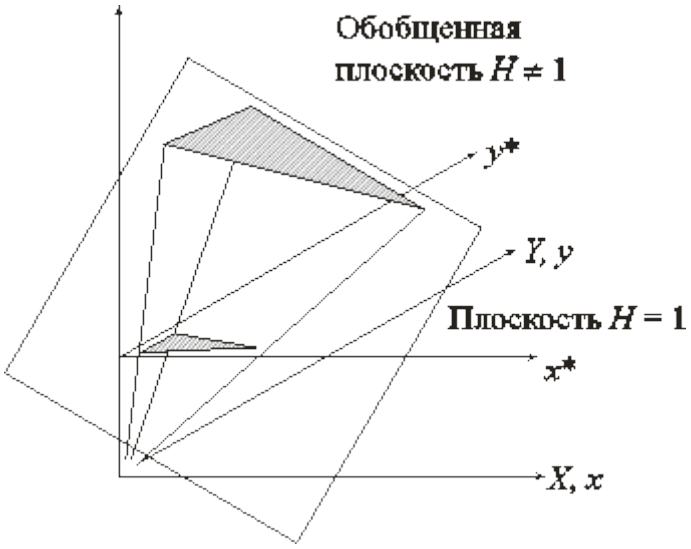

Рисунок 2.3. Геометричне уявлення однорідних координат

З розгляду подібних трикутників видно, що:

$$H / X = 1 / x^*,$$

$$H / Y = 1 / y^*$$

Розглядаючи три компоненти, запишемо це в вигляді:

$$\left[x^* \quad y^* \quad 1\right] = \left[\frac{X}{H} \quad \frac{Y}{H} \quad 1\right]$$

Подання двовимірного вектора тривимірним або в загальному випадку n-мірного вектора (п + 1) мірним називають однорідним координатним відтворенням. При однорідному координатном відтворенні n-мірного вектора воно виконується в (п + 1) мірному просторі, і кінцеві результати в n-вимірному просторі отримують за допомогою зворотного перетворення [13]. Таким чином, двовимірний вектор [x y] представляється

трикомпонентним вектором. Розділивши компоненти вектора на однорідну координату h, виходить що:

$$x = \frac{hx}{h} \quad y = \frac{hy}{h}$$

Не існує єдиного однорідного координатного представлення точки в двовимірному просторі. Наприклад, однорідні координати (12, 8, 4), (6, 4, 2) і (3, 2, 1) представляють вихідну точку [3 2]. Для простоти обчислень вибираємо [x y 1], щоб представити непреобразованную точку в двовимірних однорідних координатах. Перетворення

$$\begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

в додаткових координатах задається виразом в однорідних координатах у вигляді

$$\begin{bmatrix} X & Y & H \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Виконання зазначених вище перетворень показує, що X = x *, Y = y *, а H = 1. Рівність одиниці додаткової координати означає, що перетворені однорідні координати рівні вихідним координатам.

У загальному випадку H ¹ 1, і перетворені звичайні координати виходять за рахунок нормалізації однорідних координат,

$$x^* = \frac{X}{H} \quad y^* = \frac{Y}{H}$$

Геометрично все перетворення x і y відбуваються в площині H = 1 після нормалізації перетворених однорідних координат.

Перевага введення однорідних координат виявляється при використанні матриці перетворень загального вигляду порядку 3x3

$$\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$$

за допомогою якої можна виконувати і інші перетворення, такі як зміщення, операції зміни масштабу і зсуву, обумовлені матричними елементами a, b, c і d. Зазначені операції розглянуто раніше.

Щоб показати вплив третього стовпця матриці перетворень 3x3:

$$\begin{bmatrix} X & Y & H \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x & y & (px+qy+1) \end{bmatrix}$$

тут X = x, Y = y, а H = px + qy + 1. Змінна H, яка визначає площину, яка містить перетворені точки, представлені в однорідних координатах, тепер утворює рівняння площини в тривимірному просторі.

Це перетворення показано на рис. 2.4, де лінія AB, що лежить в площині xy, спроектована на лінію CD площині pX + qY H + 1 = 0.

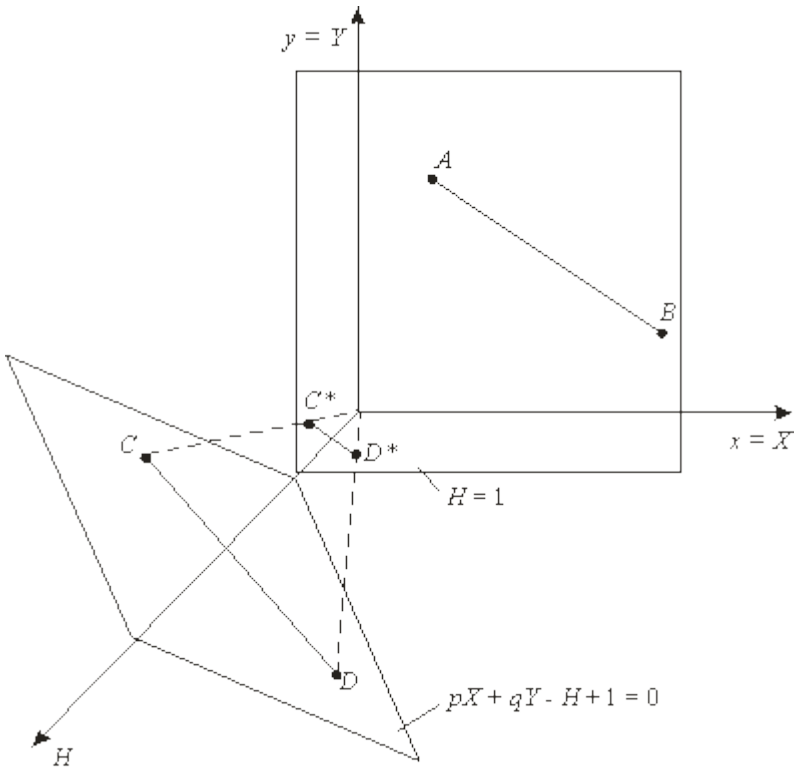

Рисунок 2.4. Перетворення лінії в однорідних координатах

На малюнку величина p = q = 1.Виполніте нормалізацію для того, щоб отримати звичайні координати:

$$x^* = \frac{X}{H} = \frac{X}{pX + qY + 1} \qquad y^* = \frac{Y}{H} = \frac{Y}{pX + qY + 1}$$

Вважаючи p = q = 1, для зображених на малюнку точок A і B з координатами відповідно (1, 3) і (4, 1) отримаємо

$$x^* = \frac{1}{1+3+1} = \frac{1}{5} \text{ та } y^* = \frac{3}{5}.$$

Результатом нормалізації є переклад тривимірної лінії CD в її проекцію C * D * на площину H = 1. Як показано на малюнку, центром проекції є початок координат.

Основна матриця перетворення розміром 3'3 для двовимірних однорідних координат може бути підрозділена на чотири частини:

$$\left[\begin{array}{cc|c} a & b & p \\ c & d & q \\ \hline m & n & s \end{array}\right].$$

Як результат, a, b, c і d здійснюють зміна масштабу, зрушення і обертання; т і п виконують зсув, а p і q - отримання проекцій. Частивна, що залишилася матриці, елемент s, виробляє повна зміна масштабу. Це перетворя можна побачити в цьому рівнянні

$$\begin{bmatrix} X & Y & H \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix} = \begin{bmatrix} x & y & s \end{bmatrix}.$$

Тут X = x, Y = y, а H = s. Це дає x * = x / s і y * == y / s. В результаті перетворення [x y 1] -> [x / s y / s 1] має місце однорідне зміна масштабу вектора положення. При s <1 відбувається збільшення, а при s> 1 - зменшення масштабу.

2.2.3 Двовимірне обертання навколо осі

Вище було розглянуто обертання зображення близько початку координат. Однорідні координати забезпечують поворот зображення навколо точок, відмінних від початку координат. У загальному випадку обертання близько довільної точки може бути виконано шляхом перенесення центру обертання в початок координат, поворотом щодо початку координат, а потім перенесенням точки обертання в початкове положення. Таким

чином, поворот вектора положення [x y 1] близько точки (т, п) на довільний кут може бути виконаний за допомогою перетворення

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m & -n & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix} = \begin{bmatrix} X & Y & H \end{bmatrix}$$

Виконавши дві операції множення матриць, можна записати

$$\begin{bmatrix} X & Y & H \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ -m(\cos\theta - 1) + n(\sin\theta) & -m(\sin\theta) - n(\cos\theta - 1) & 1 \end{bmatrix}$$

Припустимо, що центр зображення має координати (4, 3) і бажано повернути зображення на 90 ° проти годинникової стрілки навколо центральної його осі. Дія, виконане за допомогою матриці

$$\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

викликає обертання навколо початку координат, а не навколо осі. Як сказано вище, необхідно спочатку здійснити перенесення зображення таким чином, щоб бажаний центр обертання знаходився на початку координат. Це здійснюється за допомогою матриці переносу

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -4 & -3 & 1 \end{bmatrix}.$$

Потім слід застосувати матрицю обертання і, нарешті, привести результати до початку координат за допомогою оберненої матриці. Вся операція

$$\begin{bmatrix} X & Y & H \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -4 & -3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 4 & 3 & 1 \end{bmatrix}$$

може бути об'єднана в одну матричну операцію шляхом виконання матричних перетворень виду

$$[X \quad Y \quad H] = [x \quad y \quad 1] \cdot \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 7 & -1 & 1 \end{bmatrix}.$$

В результаті буде отримано x * = X / H і y * = Y / H. Двовимірні обертання біля кожної осі ортогональної системи представлені на рис. 2.4.

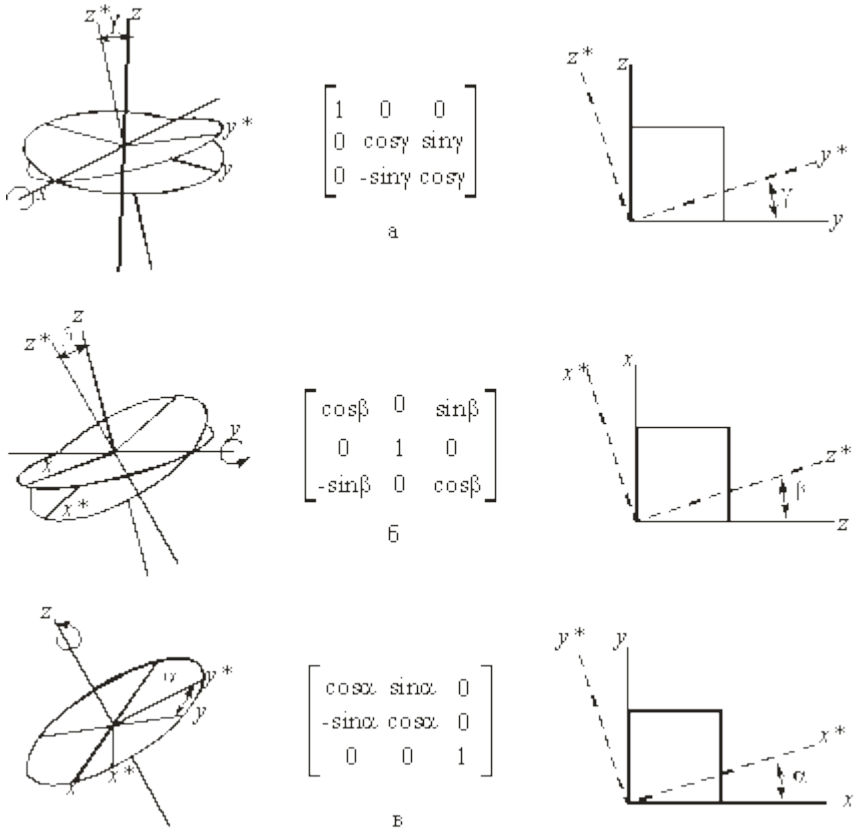

Рисунок 2.4. Обертання: а - навколо осі x; б - навколо осі y; в - навколо осі z

2.2.4 Тривимірні перетворення і проекції

Тривимірну декартову систему координат, яка є правобічної. Прийнято вважати позитивними такі повороти, при яких (якщо дивитися з кінця півосі в напрямку початку координат) поворот на 90 ° проти годинникової стрілки буде переводити одну піввісь в іншу. Будується наступна таблиця, яку можна використовувати як для правих, так і для лівих систем координат:

Таблиця 2.1

Якщо ось обертання	Напрямок повороту
X	Від y до z
Y	Від z до x
Z	Від x до y

На рис 2.5 це зображено більш наглядно.

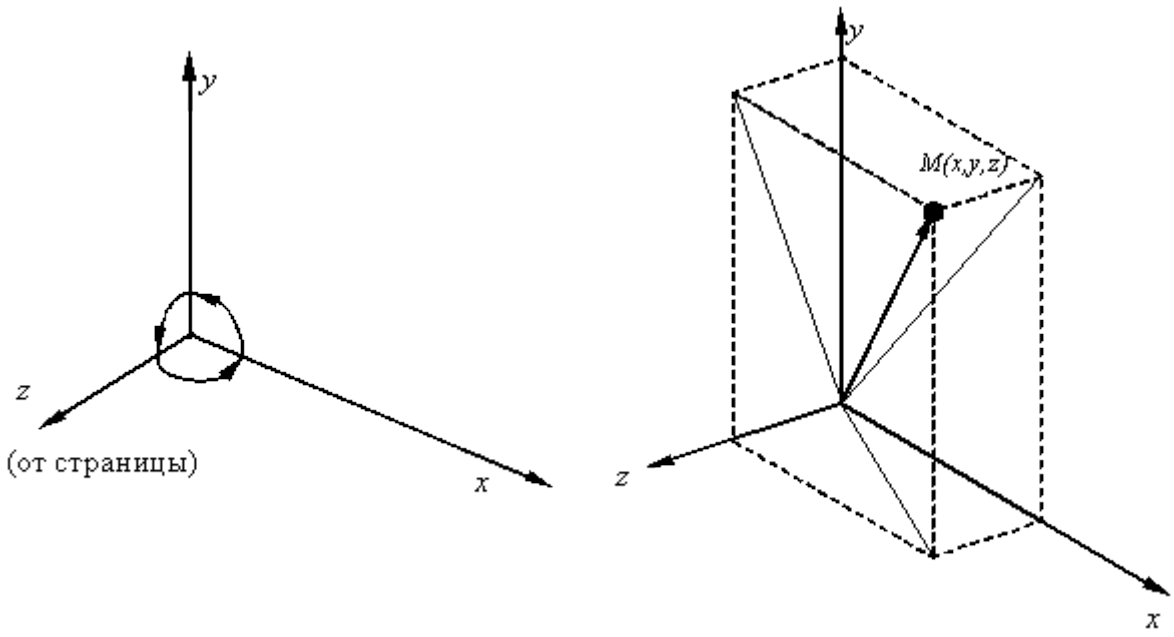

Рисунок 2.5. Тривимірна система координат

Аналогічно тому, як точка на площині описується вектором (x, y), точка в тривимірному просторі описується вектором (x, y, z).

Як і в двомірному випадку, для можливості реалізацій тривимірних перетворень за допомогою матриць перейдемо до однорідних координатах:

[X, y, x, 1] або [X, Y, Z, H]

[X *, y *, z * 1] = [], де H¹1, H ¹0.

Узагальнена матриця перетворення 4x4 для тривимірних однорідних координат має вигляд

$$\begin{bmatrix} a & b & c & p \\ d & e & f & q \\ h & i & j & r \\ l & m & n & s \end{bmatrix}$$

Ця матриця може бути представлена у вигляді чотирьох окремих частин:

$$\begin{bmatrix} 3\times3 & 3\times1 \\ 1\times3 & 1\times1 \end{bmatrix}$$

Тривимірний перенесення - є простим розширенням двовимірного:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Dx & Dy & Dz & 1 \end{bmatrix}$$

Тобто [x,y,z,1]*T(Dx,Dy,Dz)=[x+Dx,y+Dy,z+Dz,1].

Тривимірне зміна масштабу.

Часткова зміна масштабу. Вона реалізується в такий спосіб:

$$\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Тобто [x,y,z,1]*S(Sx,Sy,Sz)=[Sx*x,Sy*y,Sz*z,1].

Загальна зміна масштабу виходить за рахунок 4-го діагонального елемента, тобто.

$$[x\ y\ z\ 1] * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & S \end{bmatrix} = [x\ y\ z\ S] = [x^*\ y^*\ z^*\ \ 1] = [\ \frac{X}{S}, \frac{Y}{S}, \frac{Z}{S}, 1\ ].$$

Такий же результат можна отримати при рівних коефіцієнтах часткових змін масштабів. В цьому випадку матриця перетворення така:

$$\begin{bmatrix} \frac{1}{s_x} & 0 & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 & 0 \\ 0 & 0 & \frac{1}{s_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Тривимірний зрушення. Недіагональні елементи матриці 3'3 здійснюють зрушення в трьох вимірах, тобто.

$$[x\ y\ z\ 1] * \begin{bmatrix} 1 & b & c & 0 \\ d & 1 & f & 0 \\ h & i & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [x+yd+hz,\ bx+y+iz,\ cx+fy+z,\ 1].$$

Тривимірне обертання. Двомірний поворот, розглянутий раніше, є в той же час тривимірним поворотом навколо осі Z. У тривимірному просторі поворот навколо осі Z описується матрицею

$$\begin{bmatrix} \cos(\Theta) & \sin(\Theta) & 0 & 0 \\ -\sin(\Theta) & \cos(\Theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Матриця повороту навколо осі X має вигляд

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\Theta) & \sin(\Theta)0 & 0 \\ 0 & -\sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Матриця повороту навколо осі Y має вигляд

$$\begin{bmatrix} \cos(\Theta) & 0 & -\sin(\Theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\Theta) & 0 & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Результатом довільній послідовності поворотів навколо осей x, y, z є матриця

$$\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ h & i & j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Підматрицю 3x3 називають ортогональною, так як її стовпці є взаємно ортогональними одиничними векторами.

Матриці повороту зберігають довжину і кути, а матриці масштабування і зсуву немає.

2.3 Представлення просторових форм

У багатьох додатках машинної графіки виникає потреба в поданні тривимірних форм: при проектуванні літаків, при відновленні тривимірних тіл по зображеннях їхніх поперечних перерізів, побудованих за допомогою машинної томографії, при автоматичному складанні і в багатьох інших. же відомо, як зображуються просторові об'єкти, коли їх вдається представити у вигляді послідовності відрізків прямих, заданих в світових координатах. Сукупність відрізків не є адекватним описом об'єкта, оскільки відрізки самі по собі не визначають поверхонь [4]. У той же час інформація про поверхнях необхідна для проведення обчислень, пов'язаних зі стиранням прихованих частин зображення, для визначення обсягів. Таким чином, приходимо до висновку, що для опису тривимірних форм необхідні поверхні - примітиви більш високого рівня, ніж відрізки

Зупинимо увагу на двох широко поширених тривимірних уявленнях поверхонь в просторі: полігональних сітках і параметричних Бікубічеський шматках. Полігональної сіткою є сукупність пов'язаних між собою плоских багатокутників. Зовнішню форму більшості будівель можна легко і природно описати за допомогою полігональної сітки (так само, як меблі і кімнати). Полігональні сітки застосовуються також для представлення об'єктів, обмежених криволінійними поверхнями. Однак недоліком цього методу є його приблизність. Видимі помилки в такому поданні можна зробити як завгодно малими, використовуючи все більше число багатокутників для поліпшення кусочно-лінійної апроксимації об'єкта, але це призведе до додаткових витрат пам'яті і обчислювального часу для алгоритмів, що працюють з таким поданням.

Параметричні Бікубічені шматки описують координати точок на викривленій поверхні за допомогою трьох рівнянь (по одному для x, y і z). Кожне з рівнянь має дві змінні (два параметра), причому показники ступеня при них не вище третьої (звідси назва Бікубіческий). Кордонами шматків є параметричні кубічні криві. Для представлення поверхні із заданою точністю потрібна значно менша кількість Бікубіческий шматків, ніж при апроксимації полігональної сіткою. Однак алгоритми для роботи з Бікубіческий об'єктами істотно складніше алгоритмів, що мають справу з багатокутниками.

При використанні обох методів тривимірне тіло представляється у вигляді замкнутої поверхні.

Полігональна сітка являє собою сукупність ребер, вершин і багатокутників. Вершини з'єднуються ребрами, а багатокутники розглядаються як послідовності ребер або вершин. Сітку можна уявити кількома різними способами, кожен з них має свої переваги і недоліки. Для оцінки оптимальності уявлення використовують такі критерії:

- Обсяг необхідної пам'яті;
- Простота ідентифікації ребер, інцидентних вершині;
- Простота ідентифікації багатокутників, яким належить дане ребро;
- Простота процедури пошуку вершин, що утворюють ребро;
- Легкість визначення всіх ребер, що утворюють багатокутник;
- Простота отримання зображення полігональної сітки;
- Простота виявлення помилок в поданні (наприклад, відсутність ребра або вершини або багатокутника).

Далі буде більш детально розглянуто 3 способи опису полігональних сіток.

2.3.1 Явна завдання багатокутників

Кожен багатокутник можна представити у вигляді списку координат його вершин:

P = ((x1, y1, z1), (x2, y2, z2), ..., (xn, yn, zn))

Вершини запам'ятовуються в тому порядку, в якому вони зустрічаються при обході навколо багатокутника. При цьому всі послідовні вершини багатокутника (а також перша і остання) з'єднуються ребрами. Для кожного окремого багатокутника даний спосіб запису є

ефективним, проте для полігональної сітки дає втрати пам'яті внаслідок дублювання інформації про координати загальних вершин.

Полігональна сітка зображується шляхом креслення ребер кожного багатокутника, однак це призводить до того, що загальні ребра малюються двічі - по одному разу для кожного з багатокутників.

2.3.2 Завдання багатокутників за допомогою покажчиків в список вершин

При використанні цього подання кожен вузол полігональної сітки запам'ятовується лише один раз в списку вершин V = ((x1, y1, z1), ..., (xn, yn, zn)). Багатокутник визначається списком покажчиків (або індексів) в списку вершин. Багатокутник, складений з вершин 3, 5, 7 і 10 цього списку, представляється як P = (3, 5, 7, 10).

Таке уявлення має ряд переваг в порівнянні з явним завданням багатокутників. Оскільки кожна вершина багатокутника запам'ятовується тільки один раз, вдається заощадити значний обсяг пам'яті [9]. Крім того, координати вершини можна легко змінювати. Однак все ще не просто відшукувати багатокутники із загальними ребрами. Ребра при зображенні всієї полігональної фігури як і раніше малюються двічі. Ці дві проблеми можна вирішити, якщо описувати ребра в явному вигляді.

2.3.3 Явна завдання ребер

У цьому поданні є список вершин V, проте будемо розглядати тепер багатокутник як сукупність покажчиків на елементи списку ребер, в якому ребра зустрічаються лише один раз. Кожне ребро в списку ребер вказує на дві вершини в списку вершин, що визначають це ребро, а також на один або два багатокутника, яким це ребро належить [6]. Таким чином, опускається багатокутник як P = (E1, ..., E2), а ребро - як E = (V1, V2, P1, P2). Якщо ребро належить тільки одному многоугольнику, то або P1 або P2 - порожньо.

При явному завданні ребер полигональная сітка зображується шляхом креслення не всіх багатокутників, а всіх ребер. В результаті вдається уникнути багаторазового малювання загальних ребер [4]. Окремі багатокутники при цьому також зображуються досить просто.

У деяких додатках ребра полігональних сіток є спільними для більш ніж двох багатокутників. Це можна побачити в картографії, коли такі підрозділи, як округу, штати і т. Д., Описуються багатокутниками. Ребро (або послідовність ребер), що представляє частину кордону між двома штатами, є також кордоном округу в кожному штаті, а можливо, і міста. Таким чином, ребро може належати одночасно шести багатокутників. Якщо взяти до уваги розподіл міст на райони, виборчі округи і шкільні ділянки, то це число відповідно зросте. Для таких додатків опису ребер можуть бути розширені, щоб включити довільне число багатокутників: E = (V1, V2, P1, P2, ..., Pn).

Ні в одному з цих уявлень завдання визначення ребер, інцидентних вершині, не є простою: для її вирішення необхідно перебрати всі ребра. Звичайно, для визначення таких відносин можна безпосередньо використовувати додаткову інформацію.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ ДЛЯ СТВОРЕННЯ КОМП'ЮТЕРНОЇ ГРАФІКИ

3.1 Опис процесу розробки

Початком проекту можна назвати підготовчий етап. Мета цього етапу полягала в тому, щоб на основі розглянутих пропозицій створити деяку концепцію майбутньої системи. Підштовхуючись від неї провести деяку оцінку затребуваності і можливості реалізації проекту.

Потім проект, концепція якого виглядала задовільною для реалізації, виходить на етап розробки вимог. На цьому етапі виконавець розвинув перелік усіх явних і прихованих вимог. Мішенями цього етапу є виявлення всіх прихованих потреб, вирішення конфліктів вимог, формування монолітного технічного рішення та аналіз реалізованості підготовленого рішення.

Коли технічне рішення було виявлено, наступний етап полягав в розробці архітектури майбутньої системи. Мета цієї стадії – формування логічної і фізичної архітектури, яка цілком покриває всі вимоги. Під час розробки архітектури проводилося рецензування та уточнення концепції, вимог і попереднього технічного рішення, що дало можливість за стерегтися від найбільш небезпечних ризиків.

Після завершення проектування архітектури варто було знову провести інспектування основних параметрів проекту і вирішити, чи існує можливість закінчити проект. В момент коли баланс було знайдено, і вдалося створити допустиму архітектуру системи, можна переходити до реалізації і постачання системи. Також у цих умовах потрібно було ділити реалізацію на декілька стадій, щоб в кінці кожного розробник мав готовий до постачання продукт. При цьому найважливіший, основний елемент повинен був розроблятися на ранніх етапах, а надбудови, що функціонують поверх цих основних компонентів, слід реалізовувати останніми. У такому випадку найбільш ризиковані для системи помилки були виправлені на перших етапах, і ризик того, що прикладна функціональність системи буде реалізована на нестабільній основі, було зменшено.

Після доставки повністю завершеною системи проект зазвичай переходить до етапу експериментальної експлуатації. Мета цього етапу полягає у випробуванні якості роботи

розробленої системи в справжніх умовах експлуатації. Найбільш часто, на цьому етапі виконавець спільно з замовником проводить вимір числових метрик, що дозволяють розкрити якість створеної системи.

Цілковито налагоджена і налаштована система включається в промислову експлуатацію. Як правило, виконавець зобов'язаний супроводжувати систему впродовж терміну гарантії. Виявленні невідповідності повинні бути виправленими. Користувачі і обслуговуючий персонал замовника повинні одержувати оперативну консультативну підтримку.

Кожний проект, в кінцевому рахунку, приходить до свого логічного закінчення. Етап припинення проекту має на меті аналіз результатів, внесення змін в процес розробки на основі отриманого досвіду і доповнення бази знань розробників новими ефективними рішеннями, а також новими готовими компонентами, які можна буде використовувати в майбутніх проектах.

3.2 Базова архітектура системи створення комп’ютерної графіки

До загального принцип архітектури системи закладений так званий «принцип мікроядра». Подібний принцип набув широкого поширення в операційних системах. Класичні мікроядра надають лише дуже невеликий набір низькорівневих примітивів, або системних викликів, що реалізують базові сервіси операційної системи. Я постарався перенести цей принцип на архітектуру системи. І зробити так, що ядро реалізує лише мінімальний загальний для всіх підсистем функціонал і займається управлінням модулями, здійснює межмодульногї взаємодія [4]. Такий підхід забезпечує з одного боку простоту розробки ядра, а з іншого робить архітектуру системи в цілому більш прозорою, ще важливим достоїнством подібного підходу є те, що зависання або помилки будь-якого модуля системи не призводять до зависання всієї системи в цілому.

На схемі зображені основні елементи розробляємо системи (рис. 3.1.). Ця схема не є повною схемою усіх елементів системи, а лише зображує основні спроектовані модулі, та зв’язки між ними.

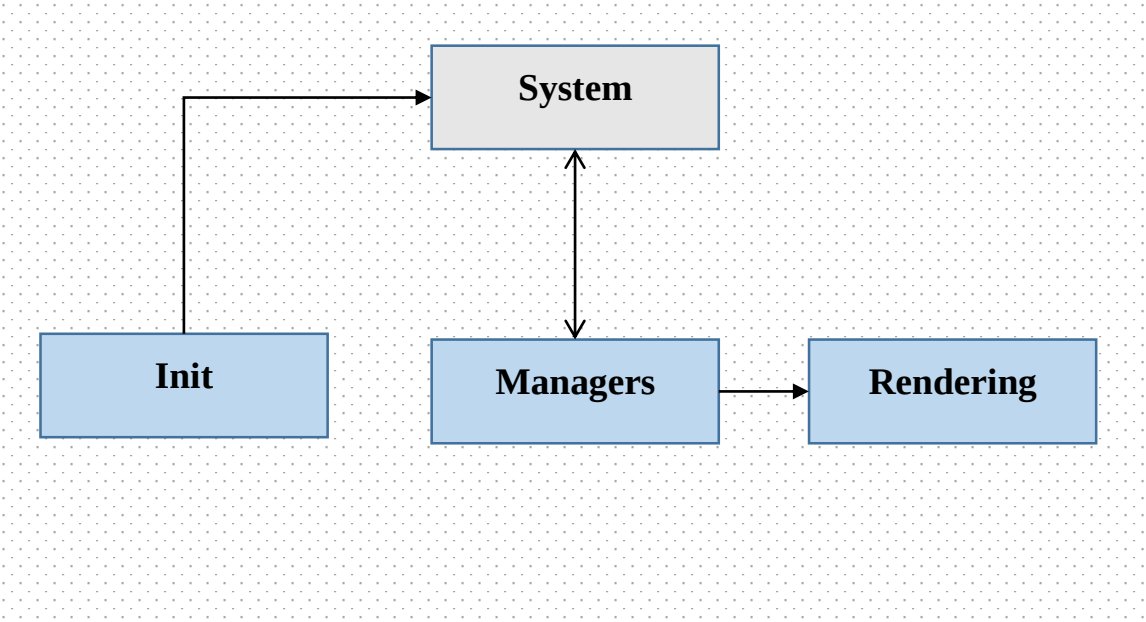

Рисунок 3.1 Схема базової архітектура графічної системи

Розробляємо система має бути модульного типу. Це рішення приводить до підвищення гнучкості розробляємої системи. А також дозволяє вносити необхідні правки в модулі, а також додавати нові модулі не впливаючи на інші структури системи [3].

В подальших розділах буде розглянуте більш детальне проектування основних модулів системи, та їх підсистем.

3.3 Проектування та розробка модулів ініціалізації

Модуль ініціалізації є одним з основних модулів системи [20]. Основними задачами цього модуля є створення вікна програми, на (рис. 3.2) зображено схему спроектованого модуля ініціалізації.

Модуль ініціалізації (Init_GLUT) - відповідає за створення, зміни розміру, та закриття вікна графічної програми. При створенні та функціонуванні вікна передає системі команду рендерінгу. Також цей модуль включає в себе декілька елементів, а саме:

а) WindowInfo – задає координати початкової позиції вікна програми на моніторі, його розмір (ширину та довжину), назву. Також він відповідає за подальші зміни формату вікна, та вносить необхідні правки. Код цього модуля можна побачити в додатку А.1.

б) ContextInfo – структура, яка зберігає в собі необхідну інформацію про контекст OpenGL (наприклад версію). Код модуля знаходиться в додатку А.2.

в) FrameBufferInfo - ініціалізує основні буфери, та зберігає інформацію шо до них. Додаток А.3.

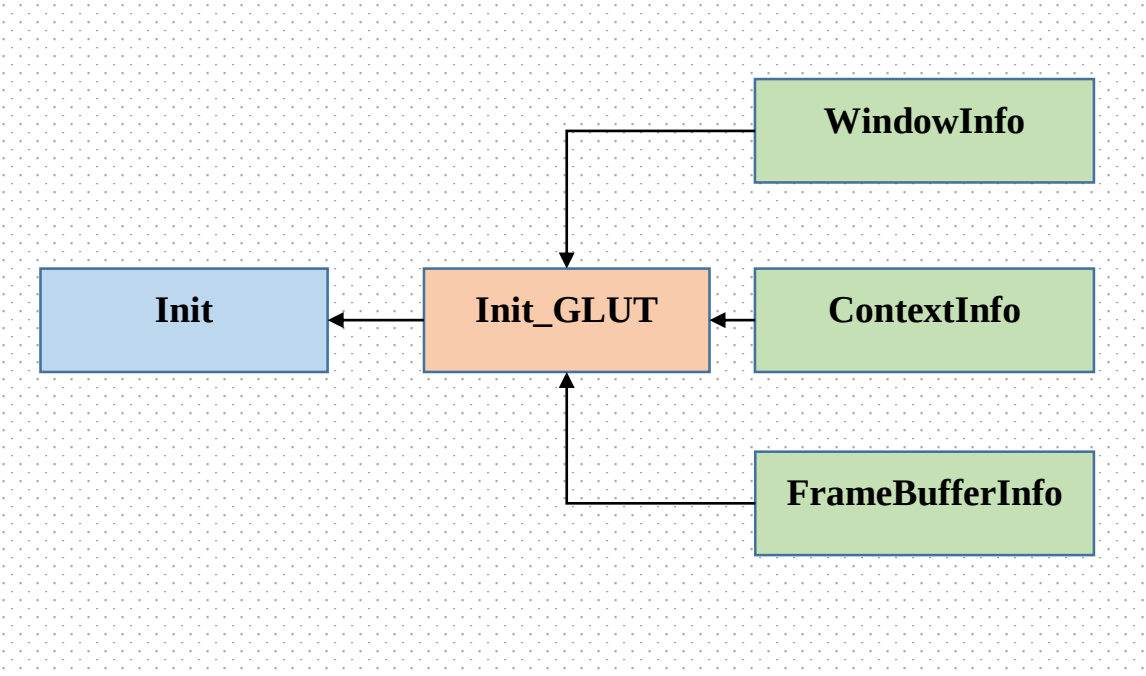

Рисунок 3.2 Структура модуля ініціалізації

Код Init_Glut модуля можна побачити у додатку А.4.

3.4 Розробка модуля менеджерів та його елементів

Менеджери керують роботою системи. Менеджер кожного типу доступний тільки в одному екземплярі. Це необхідно, оскільки дублювання ресурсів менеджерів неминуче призведе до надмірності і негативно позначиться на продуктивності. Крім того, менеджери відповідають за реалізацію загальних функцій для всіх систем.

Менеджер сцени (SceneManager) включає в себе всі графічні елементи, які додані до сцени. Також він займається контролем над всіма елементами сцени, змінює їх параметри за необхідністю

Менеджер моделей (ModelsManager) контролює всі 3д моделі сцени (об’єкти, камери, освітлення), та їх параметри (координати, розмір, колір, текстура, матеріал, тощо).

Менеджер шейдерів (ShaderManager) контролює викликання відображення елементів сцени. Всі графічні фігури створюються за допомогою виклику інструкцій шейдерним менеджером. Код менеджера шейдерів можна побачити у додатку Б.

На (рис 3.3) зображено схему менеджерів, та їх взаємодії.

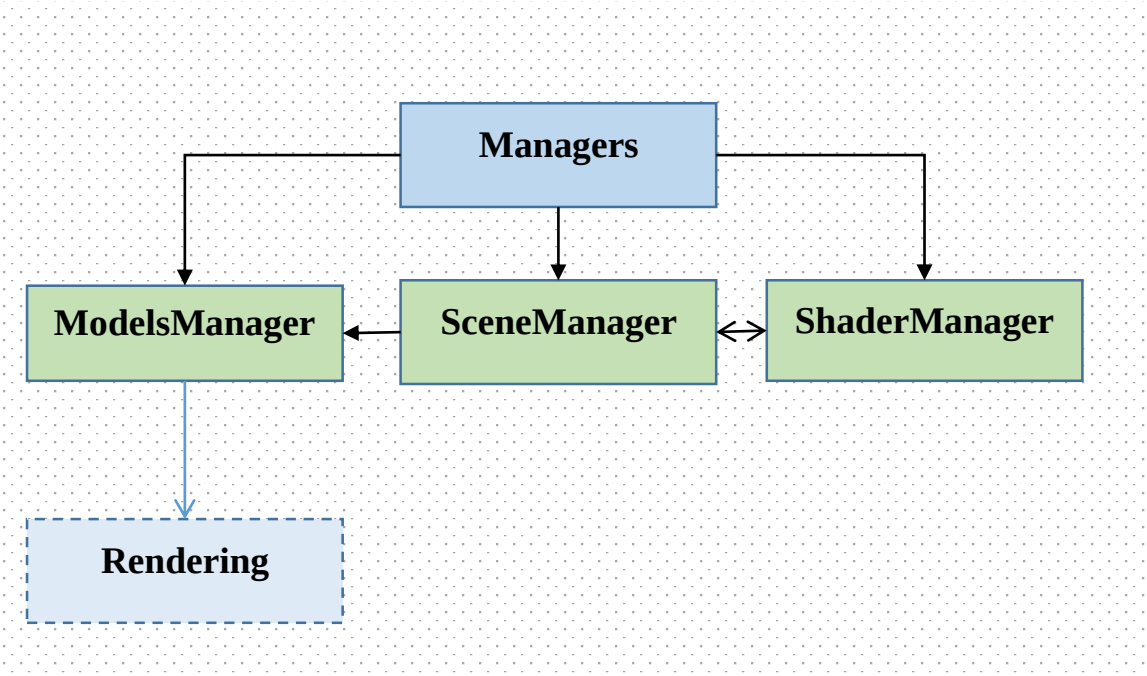

Рисунок 3.3 Схема менеджерів

3.4 Розробка модуля рендерінгу та його елементів

Візуалізація - термін в комп'ютерній графіці, що позначає процес отримання зображення по моделі за допомогою комп'ютерної програми.

Прикладом візуалізації можуть служити радарні космічні знімки, що представляють у вигляді зображення дані, отримані за допомогою радіолокаційного сканування поверхні космічного тіла, в діапазоні електромагнітних хвиль, невидимих людським оком. Часто в комп'ютерній графіці (художньої та технічної) під рендерингом (3D-рендерингом) розуміють створення плоскою картинки - цифрового растрового зображення - за розробленою 3D-сцені. Синонімом в даному контексті є візуалізація.

Візуалізація - один з найбільш важливих розділів в комп'ютерній графіці, і на практиці він тісно пов'язаний з іншими. Зазвичай програмні пакети тривимірного моделювання та анімації включають в себе також і функцію рендерингу.

До рендерінгу також входить процес растерізації зображення, тобто зміна векторної комп’ютерної графіки на растову для подальшого відображення на екрані пристрою. На рис 3.4 відображена схема модуля рендерінгу зображення.

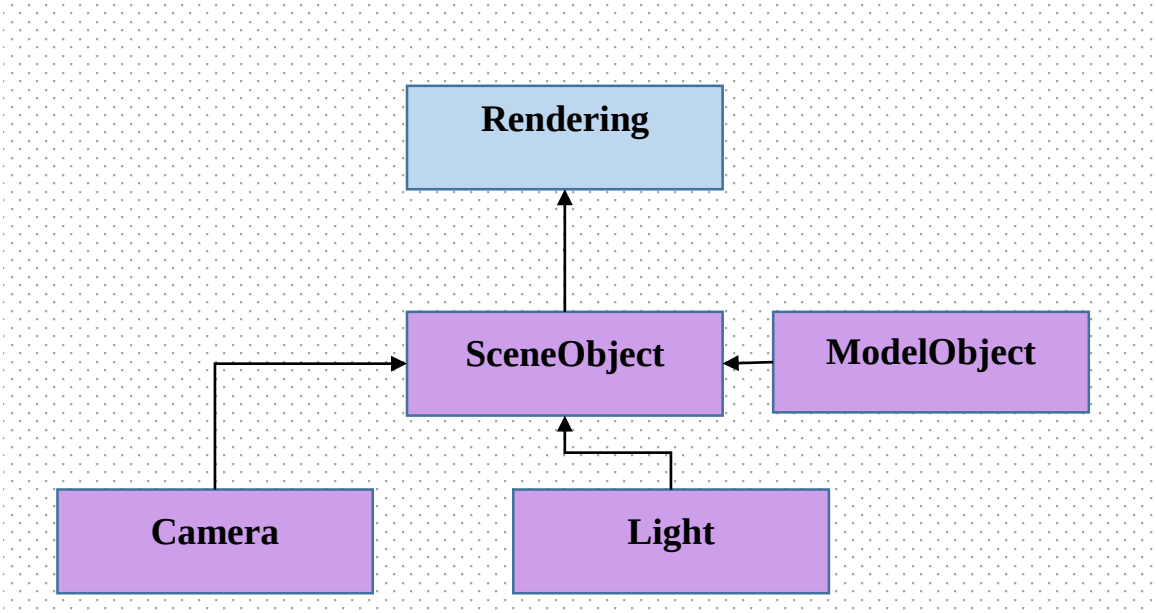

Рисунок 3.4 Схема модуля рендерінгу

Об’єкт сцени (SceneObject) - є головним класом, до якого додаються інші об’єкти. Він має свої параметри, та методи. По суті все що ми можемо бачити у вікні комп’ютерної графіки, виконується в межах сцени.

Об’єкт камери(Camera) - створює на сцені камеру за заданими координатами, значенням повороту, та кута огляду. Від параметрів камери залежить масштаб та під яким кутом зображення сцени та її об’єктів буде відображено на екрані пристрою.

Світло (Light) - створює на сцені джерело світла, яке має свої параметри. До параметрів відносяться: координати на плоскості, напрямок освітлення, кут освітлення, та тип світла.

Об’єкт моделі (ModelObject) – є класом від якого наслідуються всі додані до сцени моделі та фігури. Має такі параметри як: координати знаходження на сцені, масштаб, геометрію, матеріал, текстуру, та інші.

Сцена та всі розташовані на ній об’єкти передаються рендеру. Він відображає на екрані пристарою картинку враховуючи положення об’єктів у просторі, їх розмір, стан оберту, колір, матеріал, текстуру, тощо. Після закінчення відображення кадру, рендер викликається знову, завдяки цьому відбувається постійне (декілька разів в секунду) оновлення вмісту графічної сцени, та відбуваються анімації, якщо такі наявні.

3.5 Об’єднання розроблених модулів в систему

В попередніх підрозділах було спроектовано та розроблено модулі та елементи системи для створення комп’ютерної графіки на основі технології OpenGL. Ці модулі самі по собі нічого не роблять, тому для досягнення поставлених цілей необхідно з’єднати ці модулі зв’язками за допомогою в основному файлі в систему.

За допомогою певних методів було налаштовано зв’язки кожного елементу системи з його основним модулем як показано на схемі (рис. 3.5).

Розроблена система надає всі необхідні модулі для ініціалізації вікна програми. Спочатку модуль ініціалізації зчитує параметри вікна програми (координати стартової позиції вікна, висота вікна, ширина, та назва), потім перевіряє стан наявності OpenGL бібліотеки на комп’ютере. Вже потім створює вікно програми в якому розташована сцена та всі її елементи.

Для відображення елементів сцени, до неї додаються, за допомогою певних менеджерів: сцени, камери, світла, та графічних об’єктів. За допомогою методів певних об’єктів може бути реалізовано рух або анімація об’єктів сцени. Потім викликається рендерер, який і служить для створення картинки, та постійного її оновлення. Кількість оновлення картинки рендером залежить від сукупності декількох факторів:

а) Встановленого ліміту на кількість кадрів в секунду;
б) Кількість елементів на сцені;
в) Потужність комп’ютеру на котрому виконується програма створена за допомогою розробленої системи.

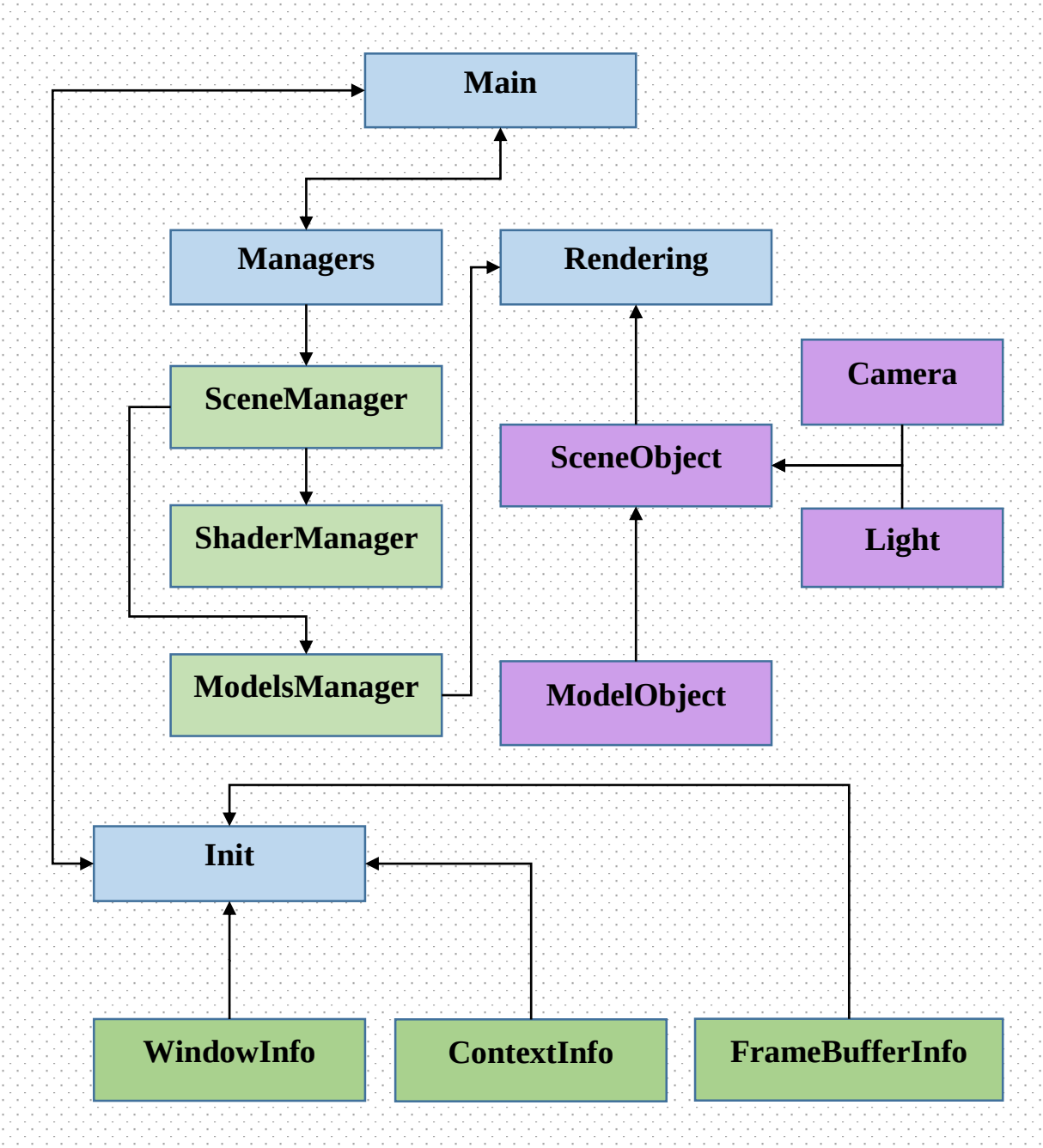

Рисунок 3.5 Загальна схема розробленої системи

3.6 Порівняння розробленої системи з аналогічними на ринку

За допомогою розробленої бібліотеки було створено невеликий графічних об’єкт для порівняння потреби у ресурсах комп’ютеру з проектами створеними за допомогою аналогічних рішень на ринку. Створений об’єкт є ідентичним з виду не зважаючи на те що було створено 2 варіанти за допомогою різних систем. Його можна побачити далі (рис 3.6). Створений демонстраційний приклад програми постійно обертається навколо осі Y, тим

самим створює анімацію. Швидкість обертання 90 градусів в секунду, що приблизно дорівнює – один повний оберт навколо осі Y за 4 секунди.

Рисунок 3.6 Приклад створеного об’єкту

Далі приведені результати тестування потреби у ресурсах комп’ютера при відображені та повільному обертанні об’єкта (рис 3.7). Для порівняння продуктивності з розробленою системою було обрано графічну систему – «Irrlicht Engine», результати продуктивності якого показані далі «рис 3.8).

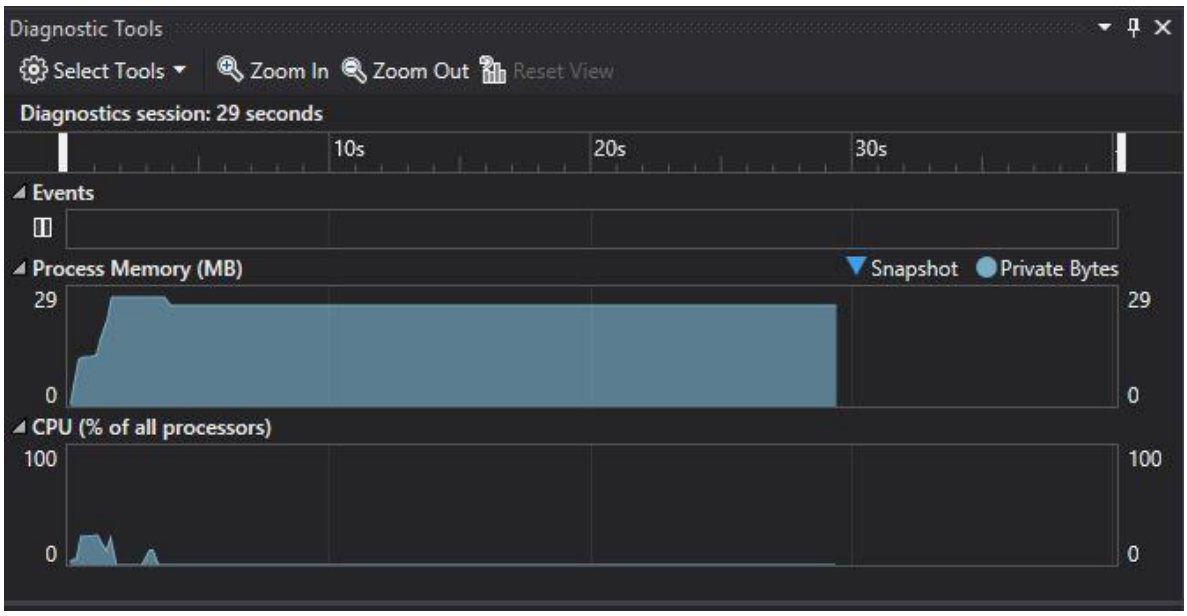

Рисунок 3.7 Програма створена за допомогою розробленої системи

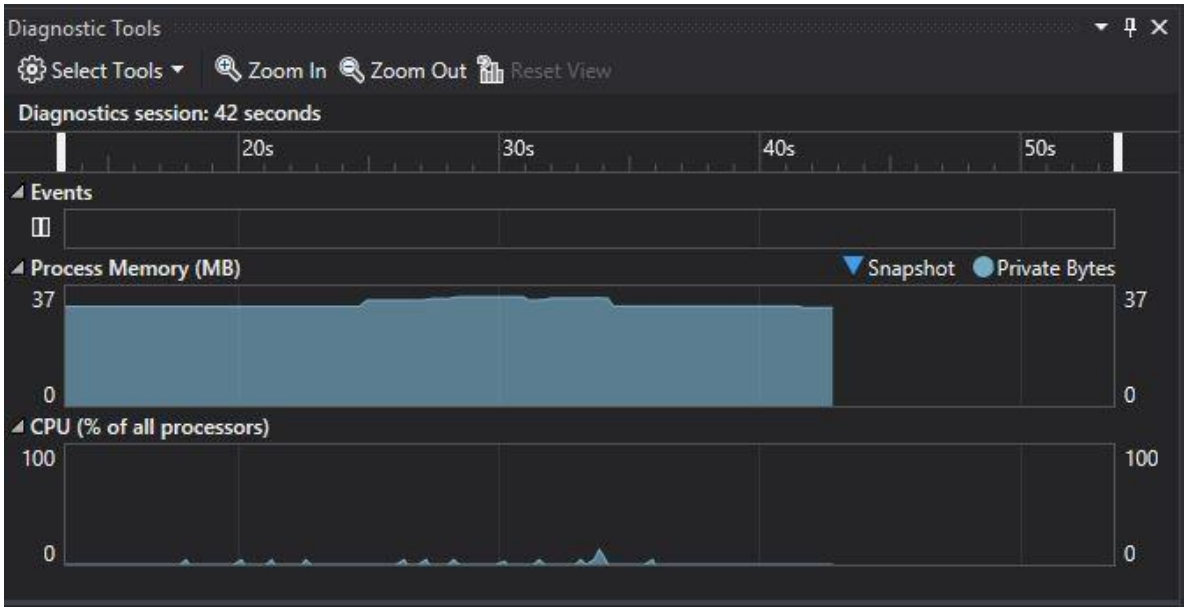

Рисунок 3.8 Програма створена за допомогою системи –«Irrlicht Engine»

Як можна побачити з результатів тестування, створена за допомогою розробленої системи програма потребує на 28% менше ресурсів пам’яті, а також демонструє кращу стабільність у роботі та використанні ресурсів центрального процесора. При цому на створення демонстраційного прикладу програми за допомогою розробленої системи пішло менше часу ніж при розробці з використанням аналогічного продукту.

3.7 Особливості та перспективи розробленої системи

Розроблена система дозволяє створювати графічний елементи комп'ютерної графіки з використання технології OpenGL. Розробка елементів виконується за допомогою розробленого інструментарію системи.

Під проведення експериментів та використання розробленої системи були виявлені наступні особливості:

а) Система непогано проявляє себе під час використання, а саме створення графічних елементів, але все ж таки існують деякі недоліки пов'язані з недостатнім функціоналом.
б) Існуючи системи для створення комп'ютерної графіки зазвичай надають більший функціонал, але при йому є деякі обмеження під час розробки. Розроблена система лише надає необхідний інструментарій для створення графічних елементів, але і надає можливості в реалізації власних ідей в організацію об'єктних структур під час використання розробленої системи.
в) Завдяки повній модульності розробленої системи програміст розробляти власні модулі та з легкість використовувати їх разом с розробленої системою.

До відмінностей даної системи від аналогічних систем на ринку можна віднести наступні пункти:

а) Система в порівнянні з багатьма подібними рішеннями є досить невеликою, але при цьому надає базові можливості для використання
б) Професійні системи для вирішення подібних задач зазвичай коштують чималих грошей.
в) Завдяки невеликому розміру її досить легко и швидко можна освоїти невідмінно від аналогічних великих систем.
г) Завдяки модульності системи, її функціонал можна з легкістю розширювати за допомогою сторонніх бібліотек.

Деякі з перспектив та планів розробленої системи:

а) Додання підтримки багатопотокових процесорів та інших оптимізації системи.
б) Реалізація фізики та взаємодії об'єктів сцени між собою.

в) Додання можливостей імпортувати розроблені в графічних редакторах 3д моделі до сцени, та можливостей взаємодії з ними.

г) Створення модулю звуку. Це наддасть можливостей для розширення функціоналу створюваємих графічних складових.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. ЕКОЛОГІЯ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даної магістерської роботи було використання інформаційних технологій для аналізу методів забезпечення кібербезпеки електроенергетичних мереж, оцінки ризиків вразливостей системи, прогнозування ймовірної структури моделювання архітектури, і як результат було розроблено математичну модель. За цією моделлю в подальшому розроблятиметься реальна система, яка значно полегшить процес виявлення вразливостей системи під час атак та запобігання їх в майбутньому. Так як в процесі проектування використовувалося ПК, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп'ютера на якому буде використовуватися розроблена об'єктна модель для оцінки ризиків та аналізу кібербезпеки під час атак, розрахунку вразливостей системи, визначення зв'язків атак і захистів та прогнозування властивостей системи.

4.1 Загальні питання з охорони праці

В законі України «Про охорону праці» визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

При роботі з обчислювальною технікою змінюються фізичні і хімічні фактори навколишнього середовища: виникає статична електрика, електромагнітне випромінювання, змінюється температура і вологість, рівень вміст кисню і озону в повітрі. Повітря забруднюється шкідливими хімічними речовинами антропогенного походження за рахунок деструкції полімерних матеріалів, які використовуються для обробки приміщень та обладнання. Неправильна організація робочого місця сприяє загальному і локальній напрузі м'язів шиї, тулуба, верхніх кінцівок, викривлення хребта і розвитку остеохондрозу. На всіх підприємствах, в установах, організаціях повинні створюватися безпечні і нешкідливі умови праці.

4.1.1 Правові та організаційні основи охорони праці

Основним організаційним напрямом у здійсненні управління в сфері охорони праці є усвідомлення пріоритету безпеки праці і підвищення соціальної відповідальності держави, і особистої відповідальності працівників.

Користувачі персональних комп'ютерів, для яких ця робота є головною, підлягають медичним оглядам: попереднім — під час влаштування на роботу і періодичним — протягом професійної діяльності раз на два роки. Жінок з часу встановлення вагітності та в період годування дитини грудьми до роботи з ПК не допускають.

Обов'язки працівників щодо додержання вимог нормативно-правових актів з охорони праці, відповідальність робітників всіх категорій за порушення вимог щодо охорони праці та структура організації/виробництв системи управління охорони праці визначені у [13].

4.1.2 Організаційно-технічні заходи з безпеки праці

В організації/підприємстві проводиться навчання і перевірка знань з питань охорони праці відповідно до вимог Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держнаглядохоронпраці України від 26.01.2005 N 15, зареєстрованого в Міністерстві юстиції України 15.02.2005 за N 231/10511 [23].

Також впроваджені організаційні заходи з пожежної безпеки - навчання і перевірку знань відповідно до вимог Типового положення про інструктажі, спеціальне навчання та перевірку знань з питань пожежної безпеки на підприємствах, в установах та організаціях України, затвердженого наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 29.09.2003 N 368, зареєстрованого в Міністерстві юстиції України 11.12.2003 за N 1148/8469 [24].

4.2 Аналіз стану умов праці

Робота над створенням об’єктної моделі забезпечення оцінки кібербезпеки, розрахунок уразливості системи і визначення зв'язків атак і захистів проходитиме в побутовому приміщенні. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп'ютером.

4.2.1 Вимоги до приміщень

Геометричні розміри приміщення зазначені в табл. 4.1.

Таблиця 4.1 - Розміри приміщення

Найменування	**Значення**
Довжина, м	3.5
Ширина, м	2,57
Висота, м	2,5
Площа, $м^2$	9
Об'єм, $м^3$	22,5

Згідно з [25] розмір площі для одного робочого місця оператора персонального комп'ютера має бути не менше 6 кв. м, а об'єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

Для зручності спільної роботи з іншими працівниками (обговорення ідей, з'ясування проблем і т. п.) в кімнаті є дивани і журнальний стіл, обставлені живими квітами. Також робочий процес пов'заний з багатьма документами, теками, журналами для чого приміщення облаштоване принтером і шафою для зручності. Задля дотримання визначеного рівня мікроклімату в будівлі встановлено систему опалення та кондиціонування.

Для забезпечення потрібного рівного освітленості кімната має вікно та систему загального рівномірного освітлення, що встановлена на стелі. Для дотримання вимог пожежної безпеки встановлено порошковий вогнегасник та систему автоматичної пожежної сигналізації.

4.2.2 Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за [26] (табл. 4.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 4.2 - Характеристики робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	720	680 ÷800
Висота простору для ніг, мм	700	не менше 600
Ширина простору для ніг, мм	600	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	450	400 ÷500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	500	не менше 300
Ширина опорної поверхні спинки, мм	500	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	700	700 ÷800

4.2.3 Навантаження та напруженість процесу праці

Під час виконання робіт використовують ПК та периферійні пристрої (лазерні та струменеві), що призводить до навантаження на окремі системи організму. Такі перекоси у напруженні різних систем організму, що трапляються під час роботи з ПК, зокрема, значна напруженість зорового аналізатора і довготривале малорухоме положення перед екраном, не тільки не зменшують загального напруження, а навпаки, призводять до його посилення і появи стресових реакцій.

Найбільшому ризику виникнення різноманітних порушень піддаються: органи зору, м’язово скелетна система, нервово-психічна діяльність, репродуктивна функція у жінок.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, а також інші профілактичні заходи на ведені в [26].

Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви:

- для операторів персональних комп’ютерів тривалістю 15 хв через дві години роботи;

4.3 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності

забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

4.3.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Роботу, пов'язану з ЕОМ з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання [27], які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне електрообладнання, зокрема персональні комп'ютери (ПК) та периферійні пристрої.

Основними робочими характеристиками персонального комп'ютера є:

– робоча напруга U=+220В +-5%;
– робочий струм I=2А;
– споживана потужність P=350 Вт.

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [26].

За умов роботи з ПК виникають наступні небезпечні та шкідливі чинники: несприятливі мікрокліматичні умови, освітлення, електромагнітні випромінювання, забруднення повітря шкідливими речовинами, шум, вібрація, електричний струм, електростатичне поле, напруженість трудового процесу та інше.

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 4.3).

Таблиця 4.3 - Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
фізичні			
- підвищена температура поверхонь обладнання	експлуатація ЕОМ, принтерів, сканерів чи/або серверного обладнання для роботи	2	ДСН 3.3.6.042-99 [25]

Продовження таблиці 4.3

- підвищений рівень шуму на робочому місці	-//-	2	ДСН 3.3.6.042-99 [25]
- підвищена або знижена вологість повітря	-//-	2	ДСН 3.3.6.042-99 [25]
- підвищена або знижена рухливість повітря	-//-	1	ДСН 3.3.6.042-99 [25]
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	ГОСТ 12.1.030-81 [28] ГОСТ 13109-97 [29]
- недостатність природного світла	порушення умов праці (вимог до приміщень)	2	ДБН В.2.5-28:2015 [30]
- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	3	ДБН В.2.5-28:2015 [30]
- підвищена яскравість світла	порушення умов праці (організації місця праці - налагодження моніторів)	1	ДСанПіН 3.3.2.007-98[26]
- понижена контрастність	-//-	1	ДСанПіН 3.3.2.007-98[26]
психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	НПАОП 0.00-1.28-10[27] ДСанПіН 3.3.2.007-98[26]
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці - сидіння користувача,) та організації робочого часу - безперервна робота)	2	НПАОП 0.00-1.28-10[27] ДСанПіН 3.3.2.007-98[26]

4.3.2 Пожежна безпека

Для гасіння пожеж в офісному приміщенні пропонується використовувати порошкові або вуглекислотні вогнегасники, так як вони є універсальними.

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), надійно захищені діелектричними щитками та/або сітками з метою недопущення потрапляння працівника під напругу.

В приміщенні наявна затверджена «План-схема евакуації з кабінету (приміщення)».

Пожежна безпека при застосуванні ЕОМ забезпечується:

1) системою запобігання пожежі,
2) системою протипожежного захисту,
3) організаційно-технічними заходами.

Згідно [31] таке приміщення, площею 9 м^2, відноситься до категорії "В" (пожежонебезпечної) та для протипожежного захисту в ньому проектом передбачено устаткування автоматичною пожежною сигналізацією із застосуванням датчиків-сповіщувачів РІД-1 (сповіщувач димовий ізоляційний) в кількості 1 шт., і застосуванням первинних засобів пожежогасіння. Відповідно до норм первинних засобів пожежогасінні пропонується використовувати:

- ручний вуглекислий вогнегасник ОУ-5 в кількості 1 шт.;
- повсть 1 1 м^2, кошму 2×1,5 м^2 або азбестове полотно 2×2 м^2 в кількості 1 шт.

Виникнення пожежі можливе, якщо на об'єкті є горючі речовини, окислювач і джерела запалювання. Вірогідність пожежної небезпеки приймається значною, якщо ймовірна взаємодія цих трьох чинників. Горючими компонентами є: будівельні матеріали для акустичної і естетичної обробки приміщень, перегородки, підлоги, двері, ізоляція силових, сигнальних кабелів і т.д.

4.3.3 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа,

шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК, укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

4.4 Гігієнічні вимоги до параметрів виробничого середовища

4.4.1 Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. Оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають [32] і наведені в табл. 4.4:

Таблиця 4.4 - Норми мікроклімату робочої зони об'єкту

Період року	**Категорія робіт**	**Температура C^0**	**Відносна вологість %**	**Швидкість руху повітря, м/с**
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

4.4.2 Освітлення

Збільшення освітленості сприяє поліпшенню працездатності навіть в тих випадках, коли процес праці практично не залежить від зорового сприйняття. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, виникає потенційна небезпека помилкових дій і нещасних випадків.

У проекті, що розробляється, передбачається використовувати суміщене освітлення. У світлий час доби використовуватиметься природне освітлення приміщення через віконні

отвори, в решту часу використовуватиметься штучне освітлення. Штучне освітлення створюється газорозрядними лампами.

Розрахунок освітлення.

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \cdot S_n, \quad (4.1)$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$S_n = a \cdot b = 3{,}5 \cdot 2{,}57 = 9$м^2,

$S_{вік} = 1/10 \cdot 9{,}45 = 0{,}9$ м^2.

Приймаємо 2 вікна площею $S = 0{,}9$ м^2 кожне.

Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 5 м, ширина 5 м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 5400 лм кожна.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (4.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \quad (4.2)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 9 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ;

Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.2), отримуємо:

$$n = \frac{300 \times 9 \times 1.1 \times 1.5}{5400 \times 0.575 \times 2} = 0,7 \approx 1$$

Приймаємо освітлювальну установку, яка складається з 1 світильника, який складається з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

4.5 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти), тобто при V приміщення > 40 м^3 на одного працюючого допускається природна вентиляція. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП.

Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

4.6 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Загальний опір захисного заземлення визначається за формулою:

$$R_{ççï} = \frac{R_ç \cdot R_n}{R_n \cdot n \cdot \eta_ç + R_ç \cdot \eta_n} \qquad (4.3)$$

де $R_з$ - опір заземлення, якими можуть бути труби, опори, кути і т.п., Ом;

R_n - опір опори, яка з'єднує заземлювачі, Ом;

n - кількість заземлювачів;

$\eta_з$ - коефіцієнт екранування заземлювача; приймається в межах 0,2 ÷ 0,9; $\eta_з = 0,7$

η_n - коефіцієнт екранування сполучної стійки; приймається в межах 0,1 ÷ 0,7; η_n = 0,5;

Опір заземлення визначається за формулою:

$$R_ç = \frac{\rho}{2\pi \cdot l}\left(\ln\frac{2 \cdot l}{d} + \frac{1}{2}\ln\frac{4 \cdot t + l}{4 \cdot t - l}\right) \quad (4.4)$$

де ρ - питомий опір ґрунту, залежить від типу грунту, Ом·м;

для піску - 400 ÷ 700 Ом·м; приймаємо ρ = 400 Ом·м;

l - довжина заземлювача, м; для труб - 2-3 м; l = 3 м;

d - діаметр заземлювача, м; для труб - 0,03-0,05 м; d = 0,05 м;

t - відстань від середини забитого в ґрунт заземлювача до рівня землі, м; t = 2 м.

$$R_з = \frac{400}{2 \cdot 3{,}14 \cdot 3}\left(\ln\frac{2 \cdot 3}{0{,}05} + \frac{1}{2}\ln\frac{4 \cdot 2 + 3}{4 \cdot 2 - 3}\right) = 110, \text{ Ом}$$

Опір смуги, що з'єднує заземлювачі, визначається за формулою:

$$R_n = \frac{\rho}{2\pi \cdot L} \cdot \ln\frac{2 \cdot L^2}{b \cdot t_l} \quad (4.5)$$

де L - довжина смуги, що з'єднує заземлювачі (м) і приблизно дорівнює периметру будівлі: $П_{буд.}$ = 42·2 + 38·2 = 160 м; L = 160 м;

b - ширина смуги, м; b = 0,03 м;

t_1 - глибина заземлення від рівня землі, м; t_1 = 0,5 м.

$$R_n = \frac{400}{2 \cdot 3{,}14 \cdot 160} \cdot \ln\frac{2 \cdot 160^2}{0{,}03 \cdot 0{,}5} = 5{,}99, \text{ Ом}$$

Кількість заземлювачів захисного заземлення визначається за формулою:

$$n = \frac{2 \cdot R_ç}{4 \cdot \eta_ç} = \frac{2 \cdot 110}{4 \cdot 0{,}7} = 79 \text{ øò} \quad (4.6)$$

де 4 - допустимий загальний опір, Ом;

2 - коефіцієнт сезонності.

Визначаємо загальний опір захисного заземлення:

$$R_{ззп} = \frac{110 \cdot 5{,}99}{5{,}99 \cdot 79 \cdot 0{,}7 + 110 \cdot 0{,}5} = 1{,}7 \text{ , Ом}$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{ззп} < 4$ Ом.

4.7 Охорона навколишнього природного середовища

4.7.1 Загальні дані з охорони навколишнього природного середовища

Діяльність за темою магістерської роботи, а саме: Методи забезпечення кібербезпеки систем релейного захисту та автоматики в процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища», Законом України «Про забезпечення санітарного та епідемічного благополуччя населення», Законом України «Про відходи», Законом України «Про охорону атмосферного повітря», Законом України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру», Водний кодекс України.

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на видалення (знешкодження), утилізацію, тощо в ІТ галузі.

В процесі діяльності розробника об’єктної моделі за допомогою ПК виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

Відпрацьовані люмінесцентні лампи - І клас небезпеки

Акумулятор для джерел безперебійного харчування -ІІІ клас небезпеки

Змінні носії інформації - IV клас небезпеки

Макулатура - IV клас небезпеки

Побутові відходи - IV клас небезпеки

4.7.2 Вимоги до збору, пакування та розміщення відходів ІТ галузі

Наводяться вимоги зберігання виявлених за своєю роботою відходів відповідно до вимог [33].

Відходи в міру їх накопичення збирають у тару, відповідну класу небезпеки, з дотриманням правил безпеки, після чого доставляють до місця тимчасового зберігання відходів відповідно до затвердженої схеми їх розміщення.

Не допускається зберігання відходів у невстановлених схемою місцях, а також перевищення норм тимчасового зберігання відходів.

Способи тимчасового зберігання відходів визначаються видом, агрегатним станом і класом небезпеки відходів:

- Відходи I класу небезпеки зберігаються в герметичній тарі (сталеві бочки, контейнери). У міру наповнення тару з відходами закривають герметично сталевий кришкою;

- Відходи II класу небезпеки в залежності від агрегатного стану зберігаються в поліети-ленових мішках, бочках, сховищах та інших видах тари, яка запобігає поширенню шкідливих речовин;

- Відходи III класу небезпеки зберігаються в тарі, яка забезпечує локалізацію зберігання, дозволяє виконувати вантажно-розвантажувальні і транспортні роботи і виключає поширення в ОС шкідливих речовин;

- Відходи IV класу небезпеки можуть зберігатися відкрито на промисловому майданчику у вигляді конусоподібної купи, звідки їх автонавантажувачем перевантажують у самоскид і до-ставляють на місце утилізації або захоронення;

4.7.3 Визначення впливу та заходів щодо поводження з відходами ІТ галузі

З метою визначення та прогнозування впливу відходів на навколишне середовище, своєчасного виявлення негативних наслідків, їх запобігання відповідно до Закону України «Про відходи» повинен здійснюватися моніторинг місць утворення, зберігання, і видалення відходів. Відомості про місце утворення та місце розташування відходів зазначаються та наводяться у таблиці 4.5.

Таблиця 4.5 - Відомості про місце утворення та місце розташування відходів

№ з/п	Код та найменування відходів за ДК -005-96	Технологічний процес або виробництво, де утворюються відходи / клас небезпеки	Місце розташування відходу, тара та її кількість, місткість, розміри у разі наявності майданчиків розташування відходів необхідно зазначити тип покриття та наявність даху)
1	2	3	4
1	7710.3.1.26 Лампи люмінесцентні, та відходи, які містять ртуть, інші зіпсовані або відпрацьовані (Відпрацьовані ртутьвмісні люмінесцентні лампи)	1	буд.75, кв. 59
2	7710.3.1.01 Макулатура паперова та картонна (Макулатура)		буд.75, кв. 59
3	Акумулятор для джерел безперебійного живлення	3	буд.75, кв. 59

Висновки

Метою магістерської роботи було дослідження основних методів та технологій для проектування системи, яка може використовуватись при розробці комп’ютерної графіки, та її складових.

За результатом дослідження сформовано наступні висновки:

1. Існує чимало готових графічних систем, але далеко не всі ці системи розумно спроектовані, та легкі у застосуванні. Не всі вони можуть демонструвати гарні показники якості та надійності. Тому є потреба в розробці системи, яка б могла надавати програмісту всі необхідні базові інструменти для розробки графічних складових, та була б надійною.
2. В ході виконання дипломної роботи було розглянуто чи мало літератури пов’язані с поставленими задачами. Було проаналізовані існуючи рішення для розробки комп’ютерної графіки. Були вивчені їх особливості та основні недоліки.
3. В результаті проведених досліджень та вивчених особливостей подібних систем були виявлені деякі принципи та методи, які дозволяють покращити процес розробки комп’ютерної графіки, та досягти ліпших результатів.
4. Були розглянуті деякі властивості ресурсів комп’ютера та способи використання їх з користю в процесі розробки.
5. Були проаналізовані оптимальні алгоритми створення графічних елементів за допомогою комп’ютерної геометрії.
6. Також я допомогою проектних моделей було розроблено систему для створення елементів комп’ютерної графіки.
7. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в дипломній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.
8. Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Були наведені розміри приміщення та значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп'ютері.
9. А також визначені основні екологічні аспекти впливу на навколишнє природнє середовище та зазначені заходи щодо поводження з ними.

Література

1. Dunn F. 3D Math Primer for Graphics and Game Development 2nd Edition. / F. Dunn - A K Peters/CRC Press, 2011.-846p.
2. Freeman E.. Head First Design Patterns: A Brain-Friendly Guide 1st Edition./ E.Freeman, O'Reilly Media, 2004. – 694p.
3. Goodwin S. Cross Platform Game Programming (Game Development)/ S. Goodwin - Charles River Media, 2005. - 460p.
4. Gregory J. Game Engine Architecture 2nd Edition. / J. Gregory - A K Peters/CRC Press, 2014. - 1052p.
5. Lengyel E. Foundations of Game Engine Development, Volume 1: Mathematics. / E. Lengyel - Terathon Software LLC, 2016. – 200p.
6. Lengyel E.. Mathematics for 3D Game Programming and Computer Graphics, Third Edition 3rd Edition./ E. Lengyel - Cengage Learning PTR, 2011. -624p.
7. Lippman S., C++ Primer(5th Edition). / S. Lippman, J. Lajoie ; Addison-Wesley Professional. 2012. - 976p.
8. Martin R. Agile Software Development, Principles, Patterns, and Practices 1st Edition / R. Martin - Pearson; 1st edition, 2002. -552p.
9. Martin R., Clean Code: A Handbook of Agile Software Craftsmanship 1st Edition./ R. Martin - Prentice Hall, 2008. – 464p.
10. McConnell S. Code Complete: A Practical Handbook of Software Construction, Second Edition./ S. McConnell, Microsoft Press, 2014. -960p.
11. McShaffry M., David Graham. Game Coding Complete, Fourth Edition / M.McShaffry, D. Graham - Cengage Learning PTR, 2012. – 960p.
12. Nystrom R. Game Programming Patterns. / R. Nystorm - Genever Benning, 2014. – 354p.
13. Reddy M. API Design for C++ 1st Edition / M.Reddy - Morgan Kaufmann, 2011. - 470p.
14. Schaling B., The Boost C++ Libraries./ B. Schaling - XML Press, 2011. – 262p.
15. Sellers G. Vulkan Programming Guide: The Official Guide to Learning Vulkan (OpenGL) 1st Edition. / G. Sellers, J. Kessenich - Addison-Wesley Professional, 2016. – 480p.
16. Stoustrup B. The C++ Programming Language, 4th Edition/ B. Stoustrup - Addison-Wesley Professional. 2013. - 1376 p.
17. Strang G. Computational Science and Engineering / G. Strang - Wellesley-Cambridge Press, 2007. - 750 p.
18. Van Verth J., Essential Mathematics for Games and Interactive Applications 3rd Edition. / J. Van Verth, L. Bishop - A K Peters/CRC Press, 2015. - 624p.

19. Wayne K. Algorithms (4th Edition)/ Wayne K., R. Sedgewick; Addison-Wesley Professional. 2011. - 976p.
20. Вольф Д. OpenGL 4. Язык шейдеров книга рецептов. / Д. Вольф - ДМК Пресс, 2015. - 368с.
21. Гинсбург Д.. OpenGL ES 3.0. Руководство разработчика. / Д. Гинсбург - ДМК Пресс 2014. – 448с.
22. НПАОП 0.00-6.03-93 «Порядок опрацювання та затвердження власником нормативних актів про охорону праці, що діють на підприємстві»
23. НПАОП 0.00-4.12-05 «Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці»
24. НАПБ Б.02.005-2003 «Типове положення про інструктажі, спеціальне навчання та перевірку знань з питань пожежної безпеки на підприємствах, в установах та організаціях України»
25. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень»
26. ДСанПіН 3.3.2.007-98 «Правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»
27. НПАОП 0.00.-1.28-10 «Правил охорони праці під час експлуатації електронно-обчислювальних машин»
28. ГОСТ 12.1.030-81 «ССБТ. Електробезпечність .Захисне заземлення. Занулення»
29. ГОСТ 13109-97 «Електрична енергія. Сумісність технічних засобів віелектромагнітних. Норми якості електроенергопостачання загального призначення »
30. ДБН В.2.5-28:2015 «Природнє і штучне освітлення»
31. НАПБ Б.03.002-2007 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою»
32. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень»
33. ДСанПіН 2.2.7.029 «Гігієнічні вимоги щодо поводження з промисловими відходами та визначення їх класу небезпеки для здоров'я населення».

Додаток А

1. Код модуля ContextInfo

```
namespace Core
{
	//OpenGL versions
	struct ContextInfo
	{
		int major_version, minor_version;
		bool core;
		ContextInfo()
		{
			major_version = 3;
			minor_version = 3;
			core = true;
		}
		ContextInfo(int major_version, int minor_version, bool core)
		{
			this->major_version = major_version;
			this->minor_version = minor_version;
			this->core = core;
		}
		void operator= (const ContextInfo &info)
		{
			major_version = info.major_version;
```

```
            minor_version = info.minor_version;
            core = info.core;
        }
    };
}
```

2. Код модуля WindowInfo

```
#pragma once
#include <string>

namespace Core
{
    struct WindowInfo
    {
        std::string name;
        int width, height;
        int position_x, position_y;
        bool isReshapable;

        WindowInfo()
        {
            name = "OpenGL tutorial";
            width = 800; height = 600;
            position_x = 300;
            position_y = 300;
            isReshapable = true;
        }

        WindowInfo(std::string name, int start_position_x, int start_position_y, int width, int height, bool is_reshapable)
        {
```

```
            this->name = name;
            this->position_x = start_position_x;
            this->position_y = start_position_y;
            this->width = width;
            this->height = height;
            this->isReshapable = is_reshapable;
        }

        WindowInfo(const WindowInfo& windowInfo)
        {
            name = windowInfo.name;
            position_x = windowInfo.position_x;
            position_y = windowInfo.position_y;
            width = windowInfo.width;
            height = windowInfo.height;
            isReshapable = windowInfo.isReshapable;
        }

        void operator=(const WindowInfo& windowInfo)
        {
            name = windowInfo.name;
            position_x = windowInfo.position_x;
            position_y = windowInfo.position_y;
            width = windowInfo.width;
            height = windowInfo.height;
            isReshapable = windowInfo.isReshapable;
        }
    };
};
```

3. Код модуля FrameBufferInfo

```
#pragma once

namespace Core
{
    struct FramebufferInfo
```

```
    {
        unsigned int flags;
        bool msaa;

        FramebufferInfo()
        {
            flags = GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH;
            msaa = false;
        }

        FramebufferInfo(bool color, bool depth, bool stencil, bool msaa)
        {
            flags = GLUT_DOUBLE;
            if (color)
                flags |= GLUT_RGBA | GLUT_ALPHA;
            if (depth)
                flags |= GLUT_DEPTH;
            if (stencil)
                flags |= GLUT_STENCIL;
            if (msaa)
                flags |= GLUT_MULTISAMPLE;
            this->msaa = msaa;
        }

        void operator=(const FramebufferInfo& info)
        {
            flags = info.flags;
            msaa = info.msaa;
        }
    };
}
```

4. Код модуля Init_GLUT

```
#pragma once
#include "ContextInfo.h"
#include "FrameBufferInfo.h"
#include "WindowInfo.h"
#include <iostream>

namespace Core
{
    namespace Init
    {
        class Init_GLUT
        {
        public:
            static void Init(const Core::WindowInfo& window,
```

```
                                                const Core::ContextInfo& context,
                                                const                Core::FramebufferInfo&
framebufferInfo);

        public:
                static void Run(void);
                static void Close();
                void EnterFullscreen();
                void ExitFullscreen();
                static void PrintOpenGLInfo(const Core::WindowInfo& windowInfo,
                                        const Core::ContextInfo& context);
        private:
                static void IdleCallback(void);
                static void DisplayCallback(void);
                static void ReshapeCallback(int width, int height);
                static void CloseCallback();
        private:
                static Core::IListener* listener;
                static Core::WindowInfo windowInformation;
        public:
                static void SetListener(Core::IListener*& iListener);
        };
    }
}
using namespace Core::Init;
Core::IListener* Init_GLUT::listener= NULL;
Core::WindowInfo Init_GLUT::windowInformation;

void Init_GLUT::Init(const Core::WindowInfo&  windowInfo,
            const Core::ContextInfo& contextInfo,
            const Core::FramebufferInfo& framebufferInfo)
  {
        int fakeargc = 1;
        char *fakeargv[] = { "fake", NULL };
        glutInit(&fakeargc, fakeargv);
        windowInformation = windowInfo;
        if (contextInfo.core)
        {
                glutInitContextVersion(contextInfo.major_version,
contextInfo.minor_version);
                glutInitContextProfile(GLUT_CORE_PROFILE);
        }
        else
        {
                glutInitContextProfile(GLUT_COMPATIBILITY_PROFILE);
        }
        glutInitDisplayMode(framebufferInfo.flags);
        glutInitWindowPosition(windowInfo.position_x, windowInfo.position_y);
```

```
		glutInitWindowSize(windowInfo.width, windowInfo.height);
		glutCreateWindow(windowInfo.name.c_str());
		std::cout << "GLUT:initialized" << std::endl;
		glutIdleFunc(IdleCallback);
		glutCloseFunc(CloseCallback);
		glutDisplayFunc(DisplayCallback);
		glutReshapeFunc(ReshapeCallback);
		Core::Init::Init_GLEW::Init();
		//cleanup
		glutSetOption(GLUT_ACTION_ON_WINDOW_CLOSE,
GLUT_ACTION_GLUTMAINLOOP_RETURNS);
		PrintOpenGLInfo(windowInfo, contextInfo);

}
void Init_GLUT::Run()
{
    std::cout << "GLUT:\t Start Running  " << std::endl;
    glutMainLoop();
}
void Init_GLUT::Close()
{
    std::cout << "GLUT:\t Finished" << std::endl;
    glutLeaveMainLoop();
}
void Init_GLUT::IdleCallback(void)
{
    glutPostRedisplay();
}
void Init_GLUT::DisplayCallback()
{
    if (listener)
    {
		listener->NotifyBeginFrame();
		listener->NotifyDisplayFrame();
		glutSwapBuffers();
		listener->NotifyEndFrame();
    }
}
void Init_GLUT::ReshapeCallback(int width, int height)
{
    if (windowInformation.isReshapable){
		if (listener)
			listener->NotifyReshape(width,	height,	windowInformation.width,
windowInformation.height);
		windowInformation.width = width;
		windowInformation.height = height;
    }
}
```

```
void Init_GLUT::CloseCallback()
{
    Close();
}
void Init_GLUT::EnterFullscreen()
{
    glutFullScreen();
}
void Init_GLUT::ExitFullscreen()
{
    glutLeaveFullScreen();
}
void Init_GLUT::PrintOpenGLInfo(const Core::WindowInfo& windowInfo, const
Core::ContextInfo& contextInfo)
{
    const unsigned char* renderer = glGetString(GL_RENDERER);
    const unsigned char* vendor   = glGetString(GL_VENDOR);
    const unsigned char* version  = glGetString(GL_VERSION);

    std::cout<<"*****************************************************************
*****************" << std::endl;
    std::cout << "GLUT:\tVendor : " << vendor << std::endl;
    std::cout << "GLUT:\tRenderer : " << renderer << std::endl;
    std::cout << "GLUT:\tOpenGl version: " << version << std::endl;
    std::cout << "GLUT:\tInitial window is '" << windowInfo.name << "', with dimensions ("
<< windowInfo.width
                << "X" << windowInfo.height;
    std::cout << ") starts at (" << windowInfo.position_x << "X" << windowInfo.position_y;
    std::cout << ") and " << ((windowInfo.isReshapable) ? "is" : "is not ") << "
redimensionable" << std::endl;
    std::cout << "GLUT:\tInitial Framebuffer contains double buffers for" << std::endl;

    std::cout << "GLUT:\t OpenGL context is " << contextInfo.major_version << "." <<
contextInfo.minor_version;
    std::cout << " and profile is " << ((contextInfo.core) ? "core" : "compatibility") <<
std::endl;
    std::cout<<"*****************************************************************
*****" << std::endl;
}
void Init_GLUT::SetListener(Core::IListener*& iListener)
{
    listener = iListener;
}
```

Додаток Б

Код модуля Shader_Manager

```
#pragma once
#include <fstream>
#include <iostream>

namespace Managers
{
	class Shader_Manager
	{

	public:
		Shader_Manager(void);
		~Shader_Manager(void);
		void CreateProgram(const std::string& shaderName,
					const std::string& VertexShaderFilename,
					const std::string&
FragmentShaderFilename);
		static const GLuint GetShader(const std::string&);
	private:
		std::string ReadShader(const std::string& filename);
		GLuint CreateShader(GLenum shaderType,
					const std::string& source,
					const std::string& shaderName);
		static std::map<std::string, GLuint> programs;
	};
}

using namespace Managers;

std::map<std::string, GLuint> Shader_Manager::programs;
Shader_Manager::Shader_Manager(void){}
Shader_Manager::~Shader_Manager(void)
{
	for (std::map<std::string, GLuint>::iterator i = programs.begin();
		i != programs.end(); ++i)
	{
		GLuint pr = i->second;
		glDeleteProgram(pr);
	}
	programs.clear();
}
std::string Shader_Manager::ReadShader(const std::string& filename)
{
```

```
    std::string shaderCode;
    std::ifstream file(filename, std::ios::in);
    if (!file.good()){
            std::cout << "Can't read file " << filename.c_str() << std::endl;
            std::terminate();
    }
    file.seekg(0, std::ios::end);
    shaderCode.resize((unsigned int)file.tellg());
    file.seekg(0, std::ios::beg);
    file.read(&shaderCode[0], shaderCode.size());
    file.close();
    return shaderCode;
}
GLuint Shader_Manager::CreateShader(GLenum shaderType, const std::string& source, const
std::string& shaderName)
{
    int compile_result = 0;
    GLuint shader = glCreateShader(shaderType);
    const char *shader_code_ptr = source.c_str();
    const int   shader_code_size = source.size();
    glShaderSource(shader, 1, &shader_code_ptr, &shader_code_size);
    glCompileShader(shader);
    glGetShaderiv(shader, GL_COMPILE_STATUS, &compile_result);
    //daca exista erori output la consola
    if (compile_result == GL_FALSE)
    {
            int info_log_length = 0;
            glGetShaderiv(shader, GL_INFO_LOG_LENGTH, &info_log_length);
            std::vector<char> shader_log(info_log_length);
            glGetShaderInfoLog(shader, info_log_length, NULL, &shader_log[0]);
            std::cout << "ERROR compiling shader: " << shaderName.c_str() << std::endl <<
&shader_log[0] << std::endl;
    }
    return shader;
}
void Shader_Manager::CreateProgram(const std::string& shaderName, const std::string&
vertexShaderFilename, const std::string& fragmentShaderFilename)
{
    //read the shader files and save the code
    std::string vertex_shader_code = ReadShader(vertexShaderFilename);
    std::string fragment_shader_code = ReadShader(fragmentShaderFilename);
    GLuint vertex_shader = CreateShader(GL_VERTEX_SHADER, vertex_shader_code,
"vertex shader");
    GLuint fragment_shader = CreateShader(GL_FRAGMENT_SHADER,
fragment_shader_code, "fragment shader");
    int  link_result = 0;
    //create the program handle, attatch the shaders and link it
    GLuint program = glCreateProgram();
```

```
    glAttachShader(program, vertex_shader);
    glAttachShader(program, fragment_shader);
    glLinkProgram(program);
    glGetProgramiv(program, GL_LINK_STATUS, &link_result);
    if (link_result == GL_FALSE)
    {
            int info_log_length = 0;
            glGetProgramiv(program, GL_INFO_LOG_LENGTH, &info_log_length);
            std::vector<char> program_log(info_log_length);
            glGetProgramInfoLog(program, info_log_length, NULL, &program_log[0]);
            std::cout << "Shader Loader : LINK ERROR" << std::endl << &program_log[0]
<< std::endl;
            return;
    }
    programs[shaderName] = program;
}
const GLuint Shader_Manager::GetShader(const std::string& shaderName)
{
    return programs.at(shaderName);

}
```

Роздруковані слайди комп'ютерної презентації