Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

ВИКОРИСТАННЯ СКРИПТІВ В BLENDER ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСУ СТВОРЕННЯ 3D-МОДЕЛЕЙ ТА АНІМАЦІЇ

(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-1
Козуб Д.П.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Руденко Д.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Козубу Дмитру Павловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Використання скриптів в Blender для автоматизації процесу створення 3D-моделей та анімації

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 29 травня 2023 р.

3. Вихідні дані до роботи методична та технічна література, дані інтернет-мережі, інформація щодо API Blender, скрипти в Blender.

4. Перелік питань, що потрібно опрацювати в роботі

1. Підхід до визначення нових можливостей комп'ютерного моделювання.
2. Методологія.
3. Методи тестування та документування скриптів.
4. Програмна апробація скриптів.
5. Результати апробації.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми обробки зображень, постановка задачі, тестові зображення.

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-17.04.23	
3	Аналіз літератури з досліджуваної проблеми	18.04.23-20.04.23	
4	Аналіз технічних засобів	21.04.23-30.04.23	
5	Розробка методу	01.05.23-14.05.23	
6	Програмна реалізація	15.05.23-23.05.23	
7	Оформлення пояснювальної записки	24.05.23-26.05.23	
8	Перевірка на плагіат	27.05.23	
9	Рецензування	28.05.23	
10	Підготовка презентації та доповіді	29.05.23-30.05.23	
11	Занесення роботи в електронний архів	31.05.23	
12	Попередній захист кваліфікаційної роботи	06.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Руденко Д.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 55 с., 27 рис., 31 джерело.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ 3D, СКРІПТИНГ PYTHON, BLENDER API, ОПТИМІЗАЦІЯ РОБОЧОГО ПРОЦЕСУ, СТВОРЕННЯ УНІКАЛЬНИХ ДОПОВНЕНЬ, ВІЗУАЛІЗАЦІЯ ДАНИХ.

Об’єктом роботи є створення пакету скриптів, що вдосконалює функціонал Blender, додає нові інструменти, функції та можливості, які оптимізують програму.

Метою даної кваліфікаційної роботи є використання мови програмування Python у розробці скриптів для Blender.

Сьогодні програмна реалізація багатьох методів недоступна, що говорить про актуальність вибраної теми. Тому ідея створення програмного продукту в галузі 3D моделювання об’єктів матиме всі необхідні функції, що відповідатимуть поставленій задачі для задовільнення потреб користувача у вивченні обраної теми.

У результаті роботи був створений ресурс для розробників, які бажають вивчити інтерфейс скриптів в Blender, а також використовувати їх для розширення можливостей та налаштування робочого процесу в цій популярній програмі для 3D-моделювання та анімації.

3D MODELING SOFTWARE, PYTHON SCRIPTING, BLENDER API, WORKPROCESS OPTIMIZATION, CREATION OF UNIQUE ADDONS, DATA VISUALIZATION.

The object of this work is to create a package of scripts that enhances the functionality of Blender, adding new tools, functions, and capabilities that optimize the program.

The purpose of this qualification work is to utilize the Python programming language in script development for Blender.

Today, the software implementation of many methods is not available, which indicates the relevance of the chosen topic. Therefore, the idea of creating a software product in the field of 3D modeling of objects will have all the necessary functions that will meet the given task to satisfy the needs of the user in studying the chosen topic.

As a result of the work, a resource has been created for developers who wish to learn the scripting interface in Blender and use it to extend capabilities and customize the workflow in this popular program for 3D modeling and animation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

OpenGL – графічний інтерфейс програмування, що використовується для рендерингу 2D та 3D графіки на комп’ютерах

API (Application Programming Interface) – інтерфейс програмування застосунків, набір правил, протоколів і інструментів, які дозволяють різним програмам взаємодіяти між собою

ВСТУП

Є багато окремих інтерфейсів, які складають Blender. Основні інтерфейси добре піддаються скриптам, оскільки майже кожна можлива взаємодія користувача безпосередньо пов’язана з функцією Python. Інтерфейс Blender працюватиме як середовище розгортання та розробки для нашого програмного забезпечення. Основна мета роботи – ознайомитися з унікальними міркуваннями щодо програмування та тестування Python, залишаючись в інтерфейсі Blender [1].

Впровадження скриптів в Blender дозволяють розширювати функціональні можливості програми, додаючи нові інструменти, функції, ефекти та можливості до основного функціоналу програми. Це може значно полегшити роботу користувачів, додавши нові можливості та функції, які не доступні в стандартному наборі інструментів Blender. Автоматизуючи рутинні задачі в Blender, користувач знижує час та зусилля, потрібні для виконання певних дій.

Актуальність даної роботи полягає в тому, що однією з основних особливостей Blender є його відкритий код, що дає можливість розширювати функціональність програми. Нещодавно отримав велику фінансову підтримку від Epic Games, Nvidia та Intel завдяки створенню Фонду розвитку Blender. Це дозволило Blender Foundation набирати нових членів команди та в результаті розвиватися Blender швидше.

1 ПІДХІД ДО ВИЗНАЧЕННЯ НОВИХ МОЖЛИВОСТЕЙ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ

1.1 Процес використання комп'ютерних алгоритмів та програмного забезпечення для створення тривимірних моделей

Тривимірне моделювання – це процес створення та модифікації тривимірних об'єктів за допомогою спеціалізованої комп'ютерної програми, що забезпечує набір необхідних інструментів для користувачів. Тривимірне моделювання зазвичай починається з основної форми, що називається примітивом, наприклад, куби, кулі, тори та інші. Потім ці форми змінюються різними функціями, що передбачені програмним забезпеченням. Користувач активує ці функції, зазвичай вони функціонують, натиснувши комбінацію клавіші на клавіатурі. Зараз існує багато потужного програмного забезпечення для моделювання 3D, яке дозволяє створювати 3D-активи, спецефекти та візуалізацію зображень. Найпопулярнішими платними програмами є Autodesk Maya, Autodesk 3ds Max і Cinema 4D. Також доступно багато безкоштовних програм, найпопулярнішою з них є Blender.

Blender – це безкоштовний набір інструментів для створення тривимірної комп'ютерної графіки з відкритим кодом. Він написаний на мовах програмування C, C++ і Python (рис. 1.1).

Blender Foundation є некомерційною організацією, яка відповідає за розробку Blender. Також блендер розроблений спільнотою, яка створює додаткові плагіни написані на Python (так звані addons). Аддони додають нові можливості або покращують функціональність Blender.

Вебсайт blender.org більш популярний, ніж будь-коли, веб-сайт blender.org і кілька його субдоменів отримали разом 23 мільйони унікальних відвідувачів. Це на 35% більше, ніж минулого року, і наближається до 2 мільйонів відвідувачів на місяць (рис. 1.2).

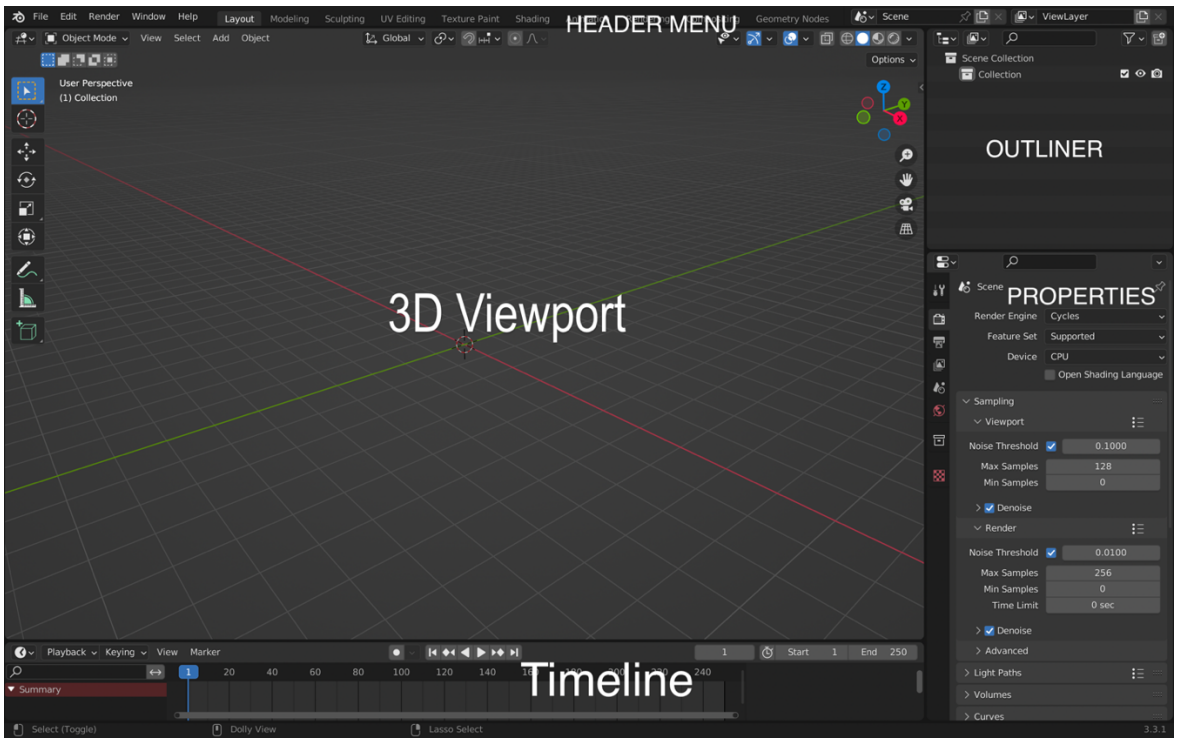

Рисунок 1.1 – Інтерфейс програми Blender

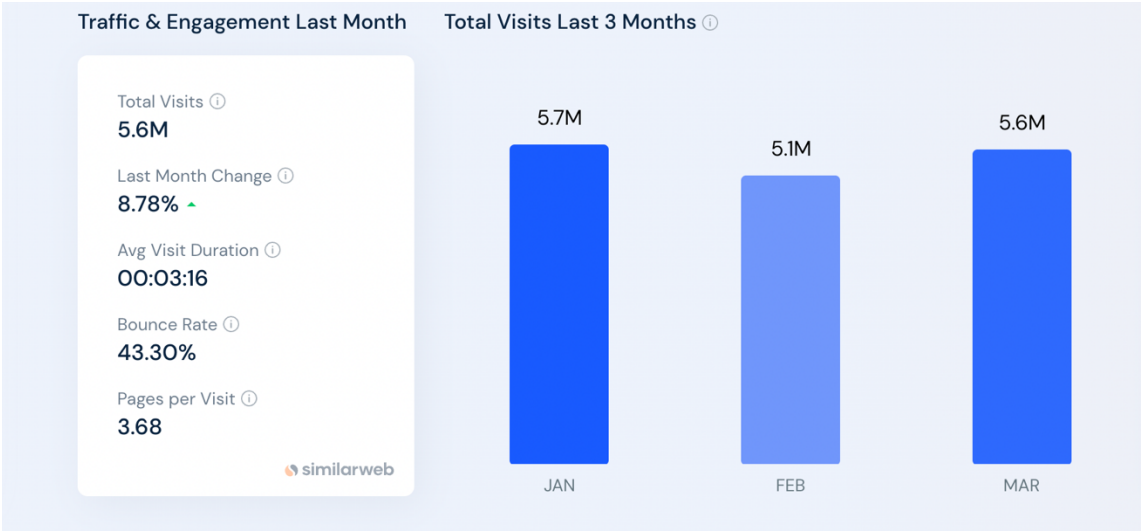

Рисунок 1.2 – Графік зростання відвідувань сайту blender.org

1.2 Зв’язок Blender software і Python

Зв’язок між інтерфейсом Blender та API Blender Python є рідкістю у світі розробки програмного забезпечення. Для платформ із підтримкою API типово

розглядати користувачів і розробників як окремі класи із окремими інструментами, окремими середовищами й окремими цілями. Blender, з іншого боку, стер межу між розробниками та користувачами, дозволяючи користувачам легко діяти як розробники і навпаки.

Тісні стосунки між розробниками та користувачами є результатом мудрих ранніх дизайнерських рішень основною командою розробників Blender. Перш ніж Blender був випущений як безкоштовне програмне забезпечення з відкритим вихідним кодом у серпні 2003 року як версія 2.26, основна команда розробників випустила документацію Python API для тодішньої преміальної версії 2.25. Python 2.0 щойно був випущений у жовтні 2000 року і Blender уже використовував його для керування викликами від інтерфейсу до своїх структур даних рівня C.

Випуск Blender Game Engine: в 2009 році було випущено Blender Game Engine - вбудований двигун гри в Blender, який також використовував Python як мову скриптування. Це надало можливість розробникам взаємодіяти з грою в реальному часі.

На початку 2010-х років художники Blender все більше усвідомлювали вплив сценаріїв Python на процес моделювання. Певні додатки стануть «обов'язковими» для художників, які цікавляться певними сферами. Розробники іншого програмного забезпечення для 3D-моделювання скористалися можливістю розробити експортери, щоб перенести Blender на своє програмне забезпечення. Сьогодні Blender має свою модульність завдяки своєму величезному резерву талантів, добре оплачуваним кар'єрним можливостям і активній спільноті розробників.

1.3 Вибір мови програмування

Для написання скриптів в програмі Blender можна використовувати наступні мови програмування.

Python – потужна мова програмування з великою спільнотою розробників і багатою екосистемою бібліотек, які можна використовувати для розширення функціональності Blender.

C/C++ також використовують для розробки addons. Це дозволяє розробникам оптимізувати графічний механізм Blender за допомогою потужного коду.

Blender також підтримує Lua, дозволяючи розробникам використовувати цю мову для створення сценаріїв. Однак найпоширенішим і рекомендованим варіантом для скриптів Blender є використання мови програмування Python.

Порівняно з Python, C/C++ вимагає більше вихідного коду для виконання простих завдань. Цей вибір підлягає підвищеним обмеженням у C/C++ під час роботи з інтерфейсом Blender UI та його функціями. Такі функції, як обробка подій і реагування на дії користувача, мають бути реалізовані вручну [2].

Крім того, розробка доповнень на C/C++ вимагає більш глибоких знань і досвіду програмування, ніж на Python. Це також може споживати більше часу та ресурсів (рис. 1.3).

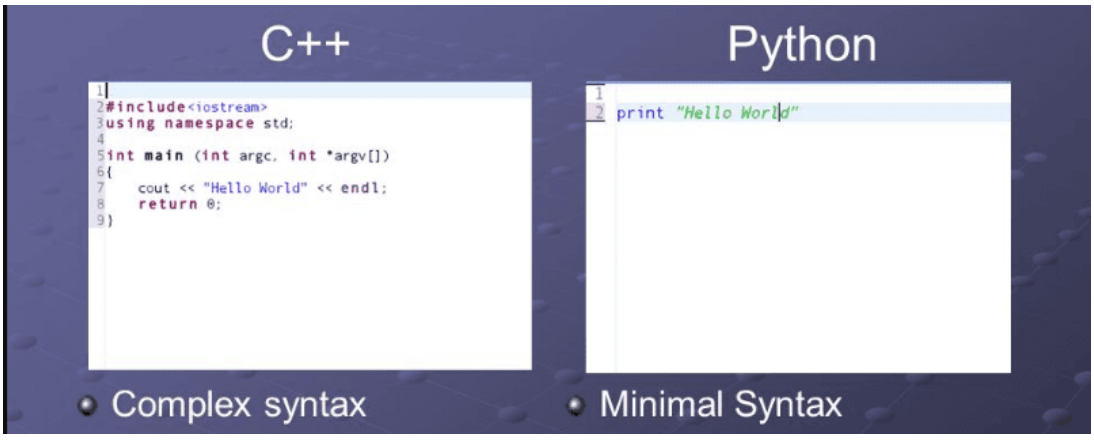

Рисунок 1.3 – Порівняння мов програмування

Синтаксис мови Python вважається простішим і легшим для розуміння, ніж мови C/C++. Python має менше синтаксичних вимог, таких як дужки,

фігурні дужки та крапки з комою, що полегшує використання для початківців. Що стосується використання розширених бібліотек, Python відомий своїми потужними бібліотеками, такими як NumPy, Pandas, OpenCV і TensorFlow, які можуть ефективно виконувати різні завдання, такі як обробка даних, машинне навчання та наукові обчислення.

1.4 Оптимізація параметрів продуктивності Blender

Blender – це потужний інструмент для створення 3D-моделей, анімацій та візуалізацій. Однак, як і будь-яке програмне забезпечення, Blender може мати проблеми з продуктивністю під час роботи з великими проєктами або на повільних комп’ютерах [3].

Одним з налаштувань продуктивності, яке можна налаштувати у Blender, є налаштування OpenGL. Налаштування OpenGL впливає на якість візуалізації сцени Blender і може вплинути на загальну продуктивність програми. Наприклад, включення використання проміжного буфера може прискорити процес візуалізації сцени Blender, тоді як включення візуалізації локальної освітленості може збільшити час візуалізації.

OpenGL – це мультиплатформний API для створення програм 2D та 3D-графіки. Він надає програмістам набір функцій для рендерингу графічних об’єктів на екрані. OpenGL є кросплатформною мовою програмування. Як і у випадку з DirectX , API спрощує розробку графічних програм і програмного забезпечення, оскільки їх потрібно адаптувати лише до стандарту OpenGL, а не до різних операційних систем і вбудованого графічного обладнання. Стандарт OpenGL описує близько 250 команд, інші організації, такі як виробники графічних карт, можуть визначати власні (тобто пов’язані з виробником) розширення.

У березні 2015 року API Vulkan був представлений як наступник OpenGL на Конференції розробників ігор . Інтерфейс програмування, спочатку

відомий як «Next Generation OpenGL» або «glNext», є відкритим вихідним кодом і також є кросплатформним. Відмінність від OpenGL полягає в тому, що програмування більше фокусується на апаратному забезпеченні, що значно збільшує обчислювальну потужність. Деякі комп'ютерні ігри вже підтримують Vulkan, але більшість покладаються на DirectX. Vulkan також розробляється групою Khronos.

Для тестування OpenGL може бути корисною програма glxgears та такі ігри, як tuxracer і armagetron (пакети мають однакову назву). Якщо підтримку 3D увімкнено, ви зможете безперебійно запускати ці три програми на відносно новому комп'ютері. Без підтримки 3D ці ігри працювали б дуже повільно (ефект слайд-шоу). Використовуйте команду glxinfo , щоб перевірити, чи активний 3D. Якщо так, вихід міститиме рядок із direct rendering: Yes(Пряме рендеринг: Так).

API Blender (інтерфейс прикладного програмування) – це набір функцій і методів, наданих Blender для взаємодії з його функціями через програмний код.

Blender API доступний на мовах програмування Python і C++. Використовуючи API Python, користувачі можуть створювати сценарії та плагіни для розширення функціональності Blender, автоматизації процесів і спрощення завдань. C++ API часто використовується для розробки плагінів і більш складних інструментів.

Використання API Blender та OpenGL разом може бути корисним для створення більш складних програм, які потребують потужних графічних можливостей. Наприклад, можна використовувати API Blender для роботи з даними сцени та передавати їх до OpenGL для відображення на екрані. Таким чином, можна створювати красиві та реалістичні 3D-сцени з більш точним керуванням параметрами рендерингу.

Іншим налаштуванням продуктивності, яке можна налаштувати в Blender, є налаштування циклу рендерингу. Налаштування циклу рендерингу впливає на якість та швидкість рендерингу зображень у Blender.

Оптимізація налаштувань продуктивності Blender може значно підвищити продуктивність програми та прискорити процес створення проєктів. Рекомендується вивчити налаштування продуктивності Blender та визначити оптимальні значення для кожного типу проєктів, щоб досягти найкращої продуктивності.

1.5 Розробка скриптів для різних задач

Скрипти для автоматичного створення моделей – це процес написання скриптів у Blender, які можуть автоматично генерувати 3D-моделі. Це корисно при створенні повторюваних елементів або складання, таких як шестірні, пластикові елементи або будь-які інші стандартні форми [4].

Написання таких скриптів може значно прискорити процес створення моделей та знизити ймовірність помилок. Існують деякі підтеми, які слід враховувати під час створення скриптів, які автоматично генерують моделі. Наприклад, створення форм із використанням генеративних алгоритмів. Алгоритм Л-систем – це метод створення геометричних фігур з використанням простих правил та рекурсії. Його можна використовувати для створення різних форм, таких як дерева, рослини, кристали і т.д. (рис. 1.4).

Рисунок 1.4 – L-системи дерев, що утворюють реальні моделі натуральних рослин

Основна ідея полягає у використанні вихідного рядка (аксіом) та набору правил для заміни символів у цьому рядку на інші символи або послідовності символів. Кожна заміна символу створює новий рядок, який можна використовувати для створення більш складних фігур.

Алгоритм Перліна – це алгоритм генерації шуму для створення природних та органічних текстур та візерунків. Їх можна використовувати для створення гір, печер, лісів та інших природних об’єктів.

Одним із найвідоміших алгоритмів Перліна є Perlin Noise (рис. 1.5). Він використовує періодичну сітку точок, кожній точці надається випадкове значення. Потім значення точок згладжуються за допомогою функції інтерполяції створення більш плавного шуму. Perlin Noise можна використовувати для створення текстур та візуальних шумових ефектів.

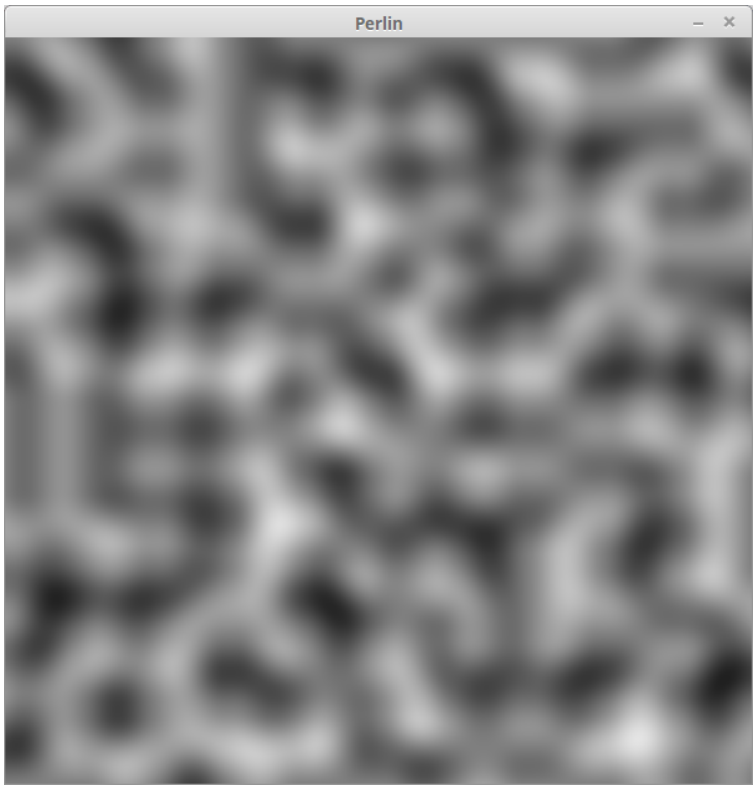

Рисунок 1.5 – Perlin Noise

Використання скриптів для повторення елементів значно прискорює процес моделювання в Blender. Щоб створити сітку Blender, можна використовувати модифікатор Array. Це дозволяє повторювати вибраний

об’єкт задану кількість разів уздовж осі. Це допоможе комбінувати це з іншими модифікаторами, такими як Mirror та Solidify, для створення складних форм сітки.

Створення скриптів для автоматизації процесу рендерингу, включає налаштування параметрів рендерингу, налаштування візуалізації матеріалів, налаштування освітлення та камери. Для цього можна використовувати такі інструменти, як Python API Blender та різні зовнішні програми, які можуть інтегруватися з Blender.

1.6 Методи тестування та документування скриптів

У цьому підрозділі розглядаються методи тестування скриптів в середовищі Blender з метою виявлення та виправлення помилок, а також забезпечення надійності та стабільності функціональності. Проводиться аналіз різних підходів до тестування скриптів у контексті Blender та розглядаються наступні методи:

– юніт-тестування скриптів: визначення та розробка юніт-тестів для індивідуальних функцій або модулів скриптів з метою перевірки правильності їх роботи. Аналізується використання спеціалізованих бібліотек для автоматизації процесу юніт-тестування в середовищі Blender;

– інтеграційне тестування скриптів: вивчення та застосування методів для виконання інтеграційних тестів скриптів, що передбачають їх взаємодію з різними компонентами та функціональними модулями Blender. Аналізується підхід до створення тестових сценаріїв та використання інструментів автоматизації для виконання інтеграційних тестів;

– валідація та верифікація скриптів: розглядаються методи валідації та верифікації скриптів з метою переконання у відповідності їхнього функціоналу та роботи заявленим вимогам. Вивчаються підходи до

формулювання критеріїв валідації та верифікації, а також використання спеціалізованих інструментів для цих цілей.

У цьому підрозділі також розглядається методика документування скриптів в середовищі Blender з метою забезпечення зрозумілості, доступності та ефективного використання коду для інших розробників. Наступні аспекти документування розглядаються в цьому підрозділі:

– коментарі та описи: аналізується використання коментарів у коді скриптів для пояснення функцій, змінних, класів та інших структурних елементів. Розглядаються рекомендації щодо стилю написання коментарів та їхнього форматування згідно зі стандартами документування;

– документаційні рядки (docstrings): вивчаються рекомендації щодо використання документаційних рядків у функціях, класах та модулях скриптів. Аналізується структура та формат документаційних рядків згідно зі стандартами документування, включаючи вказівку параметрів, повернутих значень та приклади використання;

– приклади використання: розглядається включення прикладів використання скриптів у документацію. Аналізується підхід до створення зрозумілих та детальних прикладів, які демонструють основні функціональні можливості скриптів;

– зовнішня документація: вивчаються можливості створення зовнішньої документації для скриптів в Blender, наприклад, використання Wiki-сторінок, документаційних сайтів або README файлів. Аналізується структура та зміст зовнішньої документації та її ролі в процесі розробки та спільної роботи.

Методика документування скриптів в середовищі Blender відіграє важливу роль у забезпеченні зрозумілості та ефективного використання коду для інших розробників. Використання коментарів, документаційних рядків та прикладів використання сприяє створенню зрозумілої та доступної документації. Це дозволяє розробникам легше розбиратися у функціональності скриптів, а також забезпечує зручну підтримку та спільну

роботу над проєктами. Крім того, зовнішня документація виступає як важливий ресурс для опису функціональних можливостей скриптів та надання контексту їх використання. Застосування методів документування у Blender сприяє покращенню розробки скриптів, забезпечує їхню доступність та сприяє зростанню продуктивності розробничого процесу [5].

1.7 Ігровий движок (Game Engine)

Перший виклик, з яким ми стикаємось у розробці ігор, це використання численних інструментів і платформ. Візуальна частина гри у 3D створюється в спеціалізованому програмному забезпеченні, такому як сам Blender, але частина програмування виконується в окремих середовищах. Все зібрано та налаштовано так, що можна запускати анімацію за допомогою спеціального програмного забезпечення під назвою Game Engine [6].

Ігровий движок спрямований на імітацію фізики реального світу ігрового середовища, що зрештою залишає всі складніші взаємодії відповідальними за це програмне забезпечення. Це надзвичайно складне завдання, але залежно від рівня складності використовуваного ігрового движка може досягати дуже реалістичних рівнів.

Ігровий движок – це інтерактивна душа анімації, оскільки саме ця частина програмного забезпечення контролює та дозволяє взаємодіяти з об’єктами у грі. Наприклад, якщо в певній грі ми можемо отримати автомобіль і змусити його врізатися в стіну, в результаті чого стіна зламається на багато маленьких частинок, але в інших іграх це не так. Різниця полягає в двигуні, який він використовує. Все, що передбачає взаємодію в навколишньому середовищі, від освітлення до розширених ефектів візуалізації, таких як створення дзеркала, керується мотором.

Комерційні ігри, які використовуються для створення 3D-ігор, наприклад:

– CryEngine;

– Ігрова студія;
– Blender 3D;
– Unreal Engine;
– Unity;
– Torque.

У більшості випадках ці движки відносно дорогі, але є безкоштовні варіанти та відкрите джерело. Серед відкритих або безкоштовних варіантів є саме Blender 3D, який має кілька порівняно з іншими доступними варіантами. Основна з них полягає в тому, що, крім движка ми можемо розробити весь процес лише за допомогою одного інструменту.

Один із найкращих способів розкрити потенціал інструменту – це перегляд деяких проєктів, розроблених користувачами програми. У випадку з Blender та його програмою для створення ігор, є кілька чудових прикладів інтерактивних ігор і доступних анімацій в Інтернеті для завантаження. Є приклади використання Blender для створення більш надійних і комерційних прототипів ігор, а також деякі некомерційні ігрові ініціативи, розроблені Instituto Blender, підключений до Blender Foundation [7].

Кілька прикладів проєктів, розроблених 3D-художниками для комерційних цілей або навіть просто для покращення їх дизайну.

Проєкт Ruínas, перший скрипт розроблений і створений бразильцем Вітором Бальбіо, який зображено на рисунку 1.6. Цей проєкт показує, на що здатний Blender, коли художник присвячує себе зображенню об’єктів за допомогою текстур і якісні ефекти, не кажучи вже про взаємодію з навколишнім середовищем.

Версія Blender, використана на рисунку 1.6, все ще трохи стара, але це не применшує достоїнства проєкту в цілому.

Серед багатьох комерційних ігор, розроблених за допомогою Blender, є проєкти, які використовували програмне забезпечення як графічну платформу для ігор. Є декілька прикладів використання Blender через механізми, такі як Unity 3D.

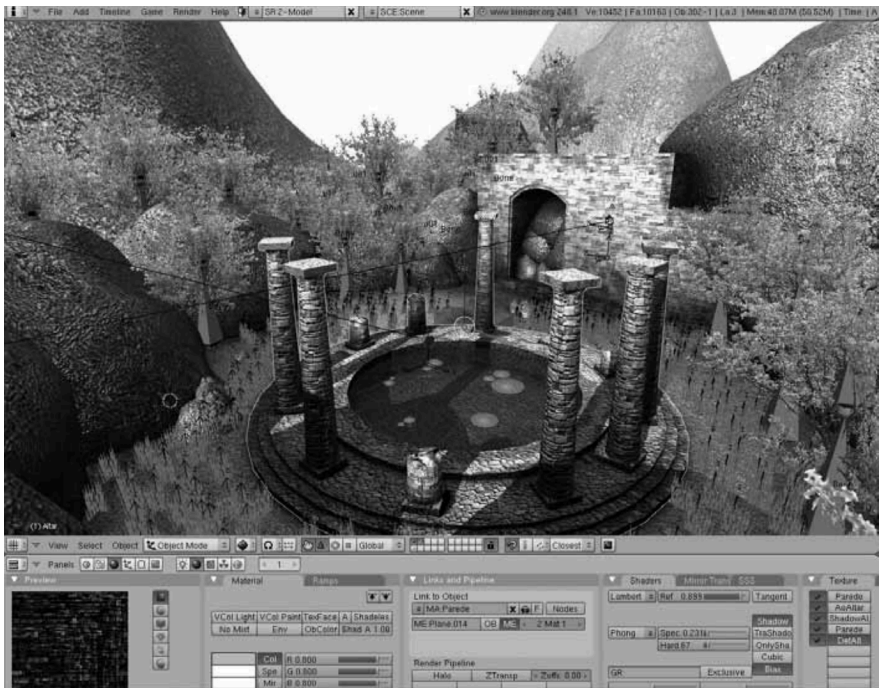

Рисунок 1.6 – Проєкт Ruínas

Ось список із кількома назвами:

– шаоліньський футбол;
– запалити;
– віртуальна гонка SAAB-91;
– мертвий кіборг.

Бурхливий період розробки ігор в кінцевому підсумку сприяв збільшенню використання ігрового двигуна Blender.

1.8 Постановка задачі

Скрипти можуть покращити взаємодію Blender з іншими програмами. Blender підтримує багато форматів файлів, що дозволяє обмінюватися даними з іншими програмами. Написання скриптів може спростити процес імпорту та експорту файлів, а також забезпечити глибшу інтеграцію Blender з іншими програмами [8].

Об’єктом роботи є створення пакету скриптів, що вдосконалює функціонал Blender, додає нові інструменти, функції та можливості, які оптимізують програму.

Метою даної кваліфікаційної роботи є використання мови програмування Python у розробці скриптів для Blender.

На основі мети роботи, що включає в себе створення системи розробки власних аддонів на Python для Blender, включаючи вибір теми, проєктування та реалізацію аддону, його тестування та впровадження. На підставі отриманих рішень були сформульовані такі дослідницькі завдання:

– проаналізувати наявні методи створення скрипта для автоматичного розміщення об’єктів на сцені відповідно до заданого розташування;

– визначити властивості математичних параметрів та дослідити їх;

– створення тестових сценаріїв для перевірки функціональності скрипту.

2 МЕТОДОЛОГІЯ

2.1 Математичні параметри автоматичного створення контенту у скриптах

Математичні формули можуть бути використані для створення складних геометричних форм, таких як фрактали, сфери, тори, а також для виконання операцій над геометрією, таких як злиття, відсікання, деформація та інші [9].

Для обчислення нових координат точок об’єкта при його переміщенні, повороті або масштабуванні можна використовувати матриці трансформації. Наприклад, для повороту об’єкта навколо осі Z на кут a можна використовувати матрицю

$$\begin{matrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & cos(a) & 0 \\ 0 & 0 & 1 \end{matrix}$$

Форма об’єкта може бути визначена за допомогою математичних формул, таких як криві Безьє, криві Катмулла-Кларка, сферичні координати тощо. Наприклад, можна створити скрипт для створення поверхні Безьє з контрольними точками, використовуючи відповідні формули.

Алгоритми заливки: для створення складних текстур та візерунків можна використовувати алгоритми заливання, такі як алгоритм Брезенхема або алгоритм відсікання. Наприклад, для побудови лінії між двома точками (x_1, y_1) та (x_2, y_2) можна використовувати алгоритм Брезенхема (рис. 2.1).

Генерація геометрії: для створення складних форм і поверхонь можна використовувати математичні рівняння, наприклад рівняння сфери.

$$x^2 + y^2 + \mathrm{c}^2 = r^2.$$

Цей скрипт створює сферу із заданим радіусом, а потім обчислює координати вершин сфери, які задовольняють заданій формулі (рис. 2.2).

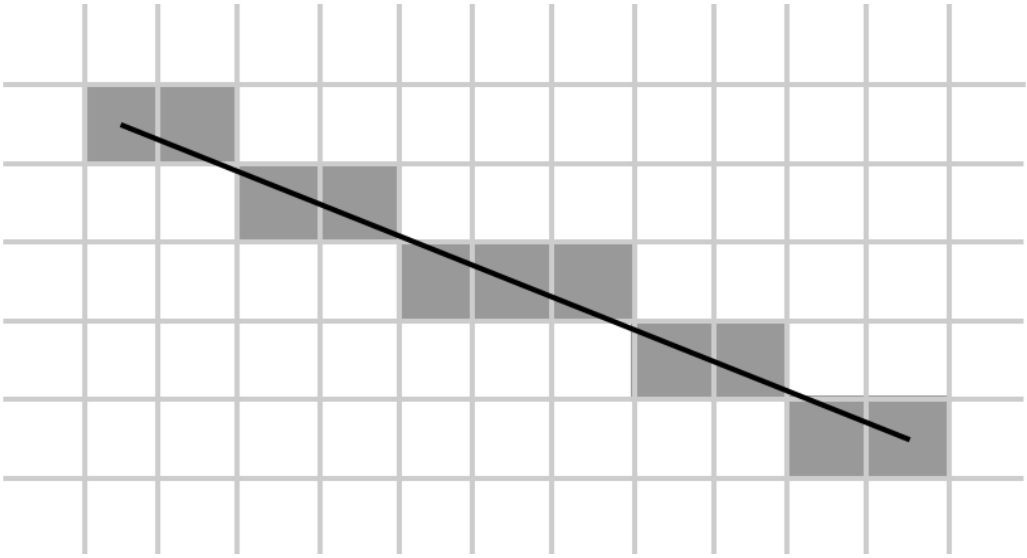

Рисунок 2.1 – Демонстрація роботи алгоритму Брезенхема

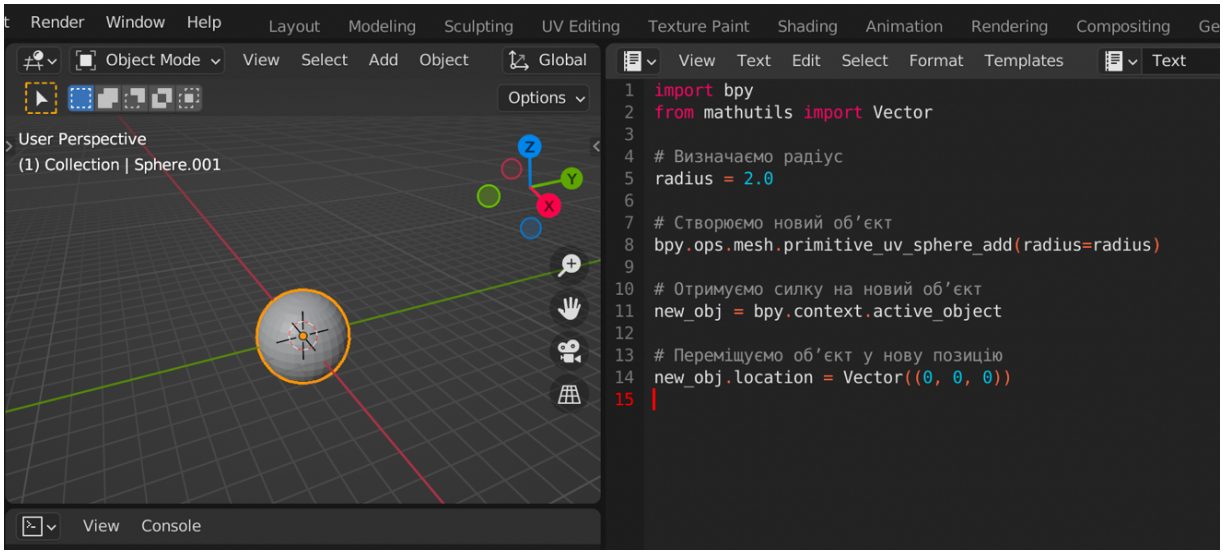

Рисунок 2.2 – Приклад частини коду з використанням формули

У цьому випадку скрипт виводить координати вершин в консоль. Ці координати можна використовувати для подальшої обробки, наприклад, для створення нових об'єктів на основі знайдених вершин.

2.2 Анімаційні формули в Python

Геометричні перетворення є одними з найбільш поширених математичних методів для побудови геометричних об’єктів при їх перетворенні. В останній час у геометричному моделюванні все частіше зустрічається контрольоване злиття маніпуляцій геометричних об’єктів довільної форми на основі прототипу, маніпулювання геометричними об’єктами, що не мають точного математичного опису. Ці проблеми можна вирішити запропонованим у роботі методом створення розширеного ресурсу для розробників, які бажають полегшити роботу в програмі Blender, яка популярна для 3D-моделювання та анімації [10].

Використання інтерполяції: для плавного переміщення об’єктів між двома точками можна використовувати методи інтерполяції, наприклад лінійну інтерполяцію. Наприклад, для лінійної інтерполяції між двома точками (x_1, y_1) та (x_2, y_2) можна використовувати таку формулу.

$$x = (1 - t) * x_1 + t * x_2$$
$$y = (1 - t) * y_1 + t * y_2.$$

Кусково-лінійна інтерполяція (Piecewise Linear Interpolation) – найпростіший вид інтерполяції. Подібно кусково-лінійній інтерполяції можна використати для наближення на кожному окремому інтервалі. Формула для знаходження значення y на відрізку [x_1, x_2] з використанням кусково-лінійної інтерполяції.

$$y = y_1 + (y_1 - y_1) * ((x - x_1) / (x_2 - x_1)),$$

де x_1, y_1 – координати першої точки;

x_2, y_2 – координати другої точки;

x – значення на осі x, для якого потрібно знайти відповідне значення y.

Приклад скрипта мовою Python для використання кусково-лінійної інтерполяції у Blender (рис. 2.3).

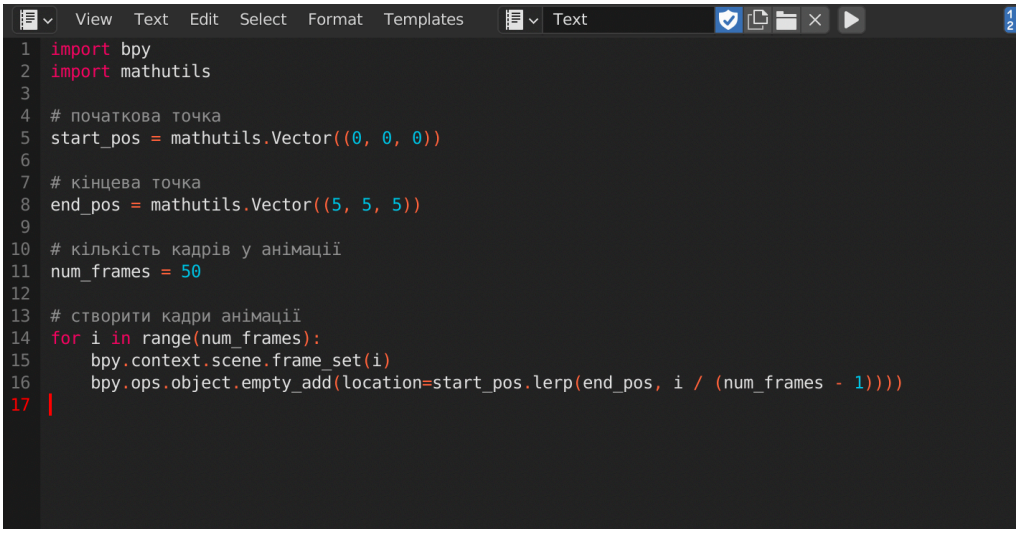

Рисунок 2.3 – Скрипт з використанням кусково-лінійної інтерполяції

Цей скрипт створює анімацію, в якій порожній об’єкт рухається вздовж лінії між початковою точкою (0, 0, 0) і кінцевою точкою (5, 5, 5). У цьому скрипті кусково-лінійна інтерполяція застосовується до визначення позиції об’єкта на кожному кадрі анімації за допомогою методу lerp класу Vector. Кожний кадр анімації створюється за допомогою методу frame_set класу Scene, а порожній об’єкт додається за допомогою функції empty_add оператора bpy.ops.object.

Одним із способів інтерполяції на всьому відрізку є інтерполяція за допомогою сплайн-функцій. Сплайн-функцією або сплайном називають кусково-поліноміальну функцію, що визначена на відрізку та має на цьому відрізку деяке число безперервних похідних. Ідея сплайн-інтерполяції полягає в побудові поліномів між парами сусідніх вузлів інтерполяції, причому для кожної пари вузлів будується свій поліном.

Найчастіше використовуються кубічні сплайни, або кубічні поліноми. Кубічна парабола виглядає так.

$$f(\mathrm{x}) = a\mathrm{x}^3 + b\mathrm{x}^2 + \mathrm{c}\mathrm{x} + d. \quad (2.1)$$

Формули для обчислення інтерполяції кубічного сплайну. Функція (2.1) на проміжку $[x_i\ x_{i+1}]$ обраховується наступним чином:

$$f(\mathrm{x}) = \frac{(x_{i+1}-x)^2 \times (2(x_i-x)+h)\times y_i}{h^3} + \frac{(x_{i+1}-x)^2 \times (x-x_i)\times m_i}{h^2} \frac{(x_{i+1}-x)^2 \times (2(x_i-x)+h)\times y_{i+1}}{h^3} + \\ + \frac{(x_{i+1}-x)^2 \times (x-x_i)^2 \times m_{i+1}}{h^2}, \quad (2.2)$$

де

$$m_i = \frac{(y_{i+1}-y_{i-1})}{2h}, \quad (2.3)$$

$$m_0 = \frac{(-3y_0+4y_1-y_2)}{2h}, \quad (2.4)$$

$$m_n = \frac{(3y_n+4y_{n-1}-y_{n-2})}{2h}. \quad (2.5)$$

Кубічна сплайн-інтерполяція не тільки гарантує максимальну однорідність у вузлах. Суміжні параболічні інтерполюючі функції (сплайни), а також їх перша та друга функції похідні роблять сплайн-інтерполяцію дуже гладкою (рис. 2.4) [11].

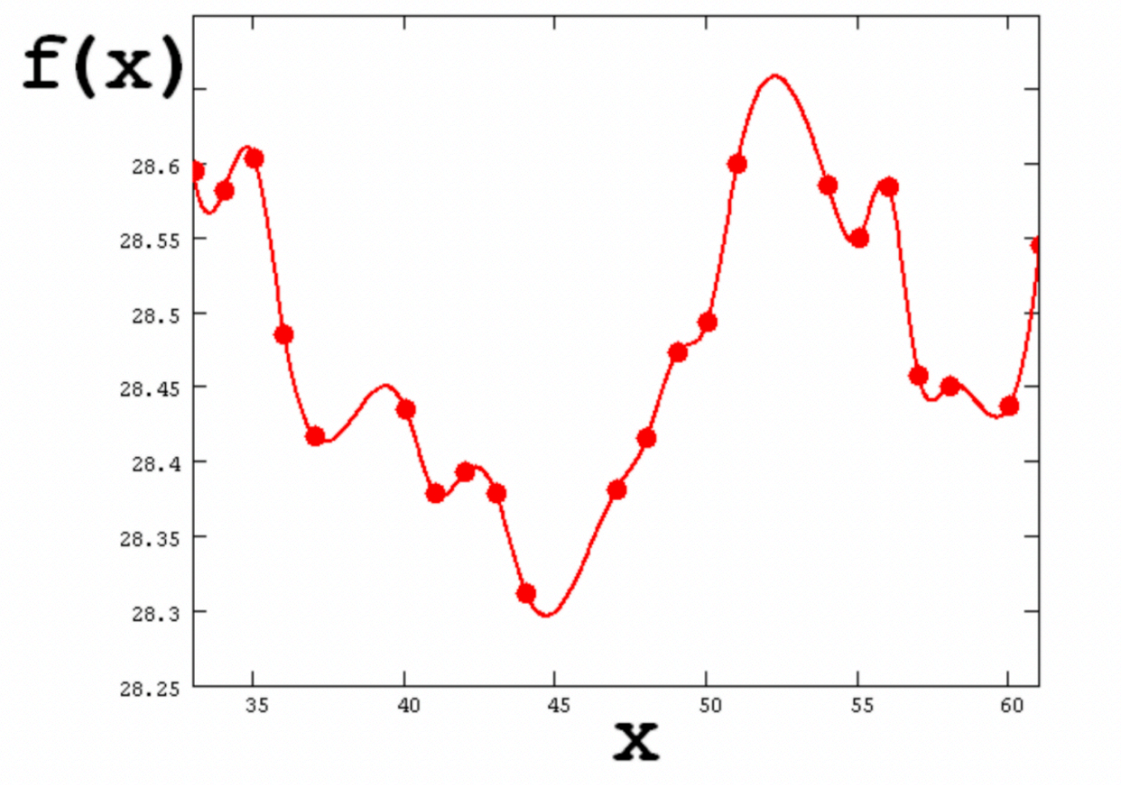

Рисунок 2.4 – Сплайн-інтерполяція

Рівняння (2.2)–(2.5) не є єдиним способом побудови кубічних сплайнів, існує ще кілька способів. На рисунку 2.4 наведено графік функції, що інтерпольована двома способами [12].

Для написання скрипта у Blender, який використовує сплайн-інтерполяцію, можна скористатися бібліотекою SciPy, яка надає реалізацію сплайн-інтерполяції. Наприклад, наступний код створить сплайн-інтерполяцію для заданих точок (рис. 2.5).

У Blender можна використовувати модуль SciPy для реалізації кубічної сплайн-інтерполяції, включаючи функції інтерполяції даних, включаючи кубічну сплайн-інтерполяцію. Для цього спочатку встановлюємо бібліотеку SciPy, а використовуємо функцію scipy.interpolate. CubicSpline, яка створює інтерполяцію кубічного сплайна на основі заданих точок (рис. 2.6).

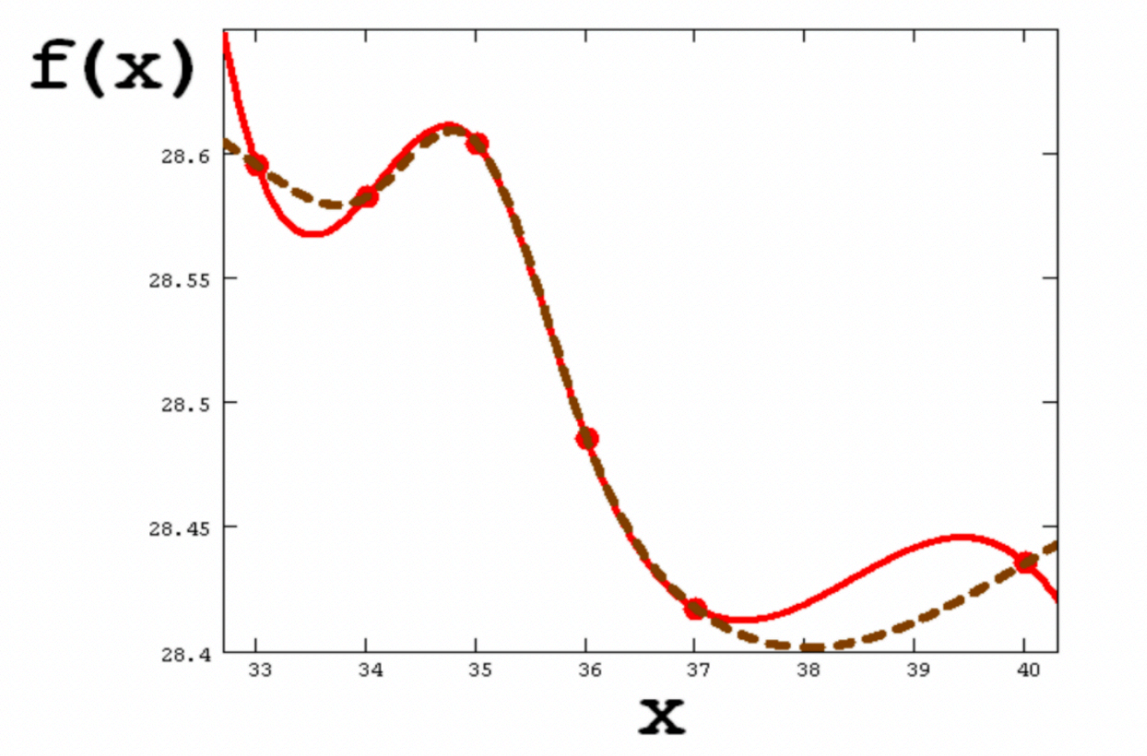

Рисунок 2.5 – Сплайн-інтерполяція двома способами

```
View  Text  Edit  Select  Format  Templates  Text

import bpy
from scipy import interpolate

# задаем точки для интерполяции
points = [(0, 0, 0), (1, 1, 1), (2, 0, 2), (3, 1, 3)]

# получаем координаты точек в виде массивов
x = [p[0] for p in points]
y = [p[1] for p in points]
z = [p[2] for p in points]

# создаем сплайн-интерполяцию
tck, u = interpolate.splprep([x, y, z], s=0)

# генерируем новые точки на основе сплайн-интерполяции
u_new = np.linspace(u.min(), u.max(), 100)
x_new, y_new, z_new = interpolate.splev(u_new, tck)

# создаем объекты на основе новых точек
for i in range(len(x_new)):
    bpy.ops.mesh.primitive_cube_add(location=(x_new[i], y_new[i], z_new[i]))
```

Рисунок 2.6 – Скрипт для реалізацію сплайн-інтерполяції

Кусково-постійна двовимірна інтерполяція (англ. Piecewise constant interpolation) є методом інтерполяції, який використовується для заповнення відсутніх значень у масивах даних, що містять значення функції в різних

точках на площині. Цей метод заснований на ідеї, що функція, визначена на кожному зрізі площини, має постійне значення (рис. 2.7) [13].

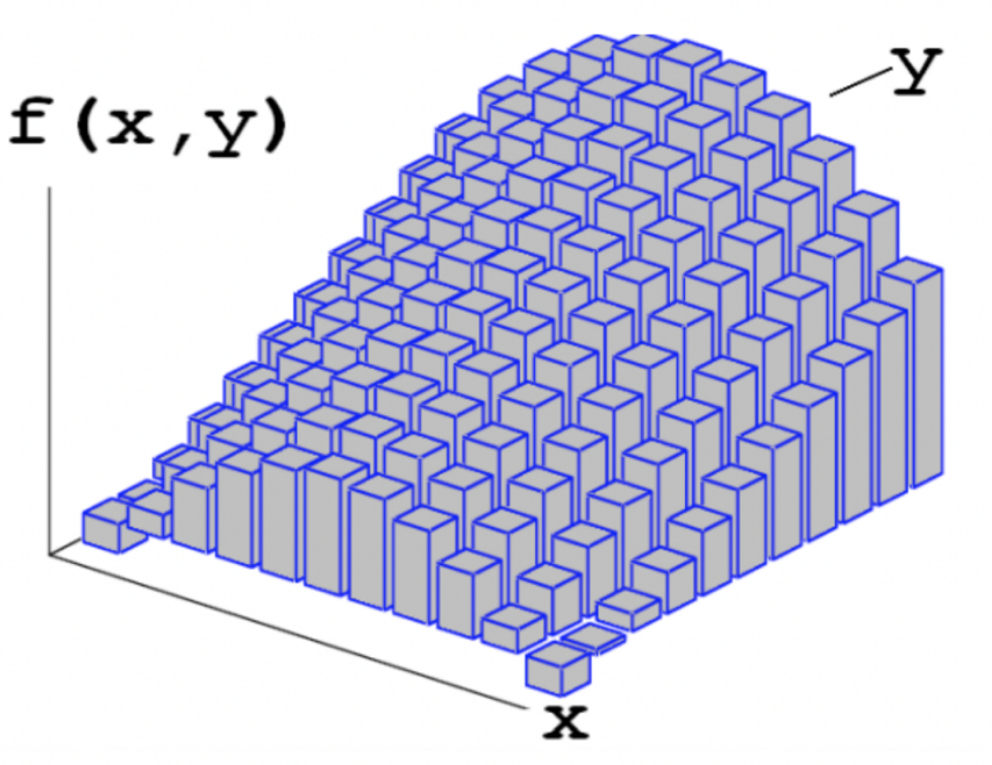

Рисунок 2.7 - Кусково-постійна двовимірна інтерполяція

У Blender для кусково-постійної двовимірної інтерполяції можна скористатися функцією bpy_extras.interpolation.interpolate_2d_constant() з модуля bpy_extras.

2.3 Метод скінченних елементів (МСЕ)

Метод скінченних елементів (МСЕ) – це чисельна техніка для розв’язування рівнянь математичної фізики та механіки тіл, що розкладаються з складних геометрій на менші скінченні елементи. Кожен кінцевий елемент складається з вузлів, в яких обчислюються значення розподілу поля.

Елементи, які можна використовувати в тривимірному випадку: куби, призми, піраміди та тетраедри (рис. 2.8).

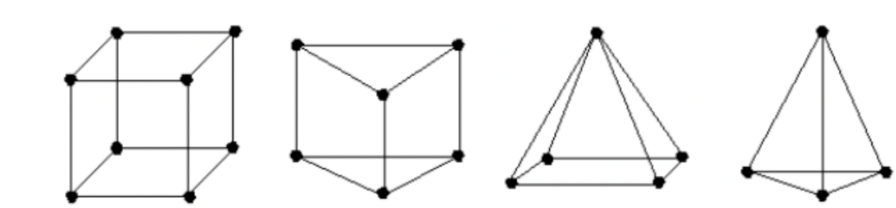

Рисунок 2.8 – Лінійні тривимірні скінчені елементи

Нижче представлені лінійні елементи, де кожне ребро визначається лише двома вузлами. Це означає, що кожне ребро залишається прямим під час деформації елемента, як зображено на рисунку 2.9 [14].

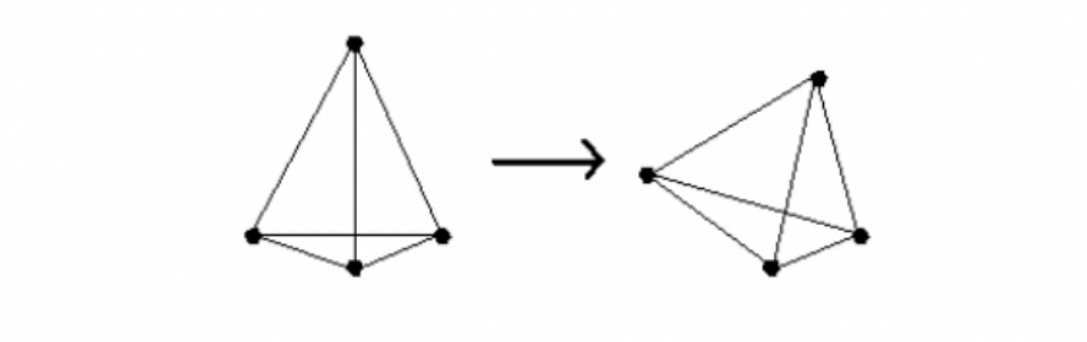

Рисунок 2.9 – Лінійна тетраедральна деформація кінцевих елементів

Існують також квадратичні елементи з одним додатковим вузлом на кожному ребрі та кубічні елементи з двома додатковими вузлами на кожному ребрі. Приклад показано на рисунку 2.10.

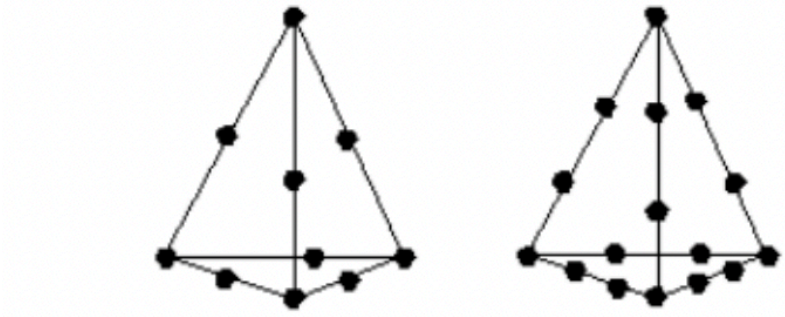

Рисунок 2.10 – Квадратичні та кубічні скінченні елемент

У Blender Python можна використовувати метод кінцевих елементів для різних завдань, таких як, наприклад, написання скриптів для моделювання фізичної поведінки об’єктів з використанням різних матеріалів, текстур і освітлення. Можна використовувати цей метод для створення скриптів, які можуть аналізувати різні параметри об’єкта і автоматично змінювати його форму та положення в залежності від умов завдання [15].

2.4 Налаштування середовища розробки

Налаштування робочого середовища є важливою частиною налаштування середовища для написання скриптів. Робоча область зазвичай складається з різноманітних панелей та вікон, які впорядковано за замовчуванням або які користувач може додавати чи змінювати. Blender, наприклад, дозволяє налаштувати робочий простір за допомогою режимів макета, які включають різні панелі, такі як 3D-вікно, редактор вузлів і шкала часу. Панелі, що доступні для використання в робочій зоні:

– панель перегляду 3D-об’єкта: ця панель забезпечує можливість перегляду та редагування 3D-об’єктів у вікні 3D-відображення;

– панель об’єктів: ця панель містить список всіх об’єктів у сцені, які можна вибирати та редагувати;

– панель інструментів: ця панель містить різні інструменти, які можна використовувати для редагування об’єктів, такі як переміщення, обертання та масштабування;

– панель властивостей: ця панель містить властивості обраного об’єкта або інструменту, які можна редагувати;

– панель редактора тексту: ця панель забезпечує можливість редагування скриптів та коду Python;

– панель анімації: ця панель містить інструменти для створення та редагування анімацій;

– панель вікна перегляду: ця панель забезпечує можливість перегляду різних вікон, таких як вікна редактора тексту та вікна налаштувань.

Налаштування робочого простору дозволяє зробити роботу зі скриптами більш продуктивною та зручною, оскільки дозволяє швидко доступатись до потрібних інструментів і панелей.

У Blender є вбудований текстовий редактор, новий редактор відкритого вихідного коду, що пропонує розробникам багато функцій, які полегшують редагування вихідного коду. Але його можна замінити на зручніший, такий як Visual Studio Code або PyCharm [16].

Для зручного написання коду додаємо вікно текстового редактору, для перегляду анімацій та 3D моделей потрібна панель перегляду 3D-об’єкта з Timeline. Панель Outliner допоможе відслідити, які нові об’єкти додались до сцени. Консоль Python дозволить користувачам взаємодіяти з API Blender і запускати код Python у реальному часі. Консоль Python дозволяє швидко тестувати функціональність, експериментувати зі сценаріями та налагоджувати код без перезапуску Blender. Це корисний інструмент для розробників, які працюють зі скриптами в Blender.

Гарячі клавіші можна встановити для текстового редактора та самого Blender. Це може значно пришвидшити роботу з розробки Blender і скриптів. Для цього можна налаштувати гарячі клавіші в самому Blender, відповідальні за певні операції, та гарячі клавіші у текстовому редакторі, щоб зручно виконувати операції в коді [17].

Щоб налаштувати гарячі клавіші в самому Blender, потрібно зайти в налаштування, вибрати вкладку Keymap і знайти потрібну операцію, для якої потрібно налаштувати гарячі клавіші. Потім можна встановити власні гарячі клавіші для своїх дій.

Ознайомлення з бібліотеками дозволить отримати розширений доступ до об’єктів у сцені, це значно зменшить час, необхідний для розробки скриптів, та спростить роботу з Blender.

Доступні бібліотеки Python для роботи з Blender включають наступні:

– bpy: це основна бібліотека для взаємодії з Blender в Python. Вона надає широкий набір класів та методів, які дозволяють отримувати доступ до об’єктів у сцені, змінювати їх властивості, створювати нові об’єкти та редагувати їх. bpy також надає засоби для роботи з матеріалами, текстурами, анімацією, освітленням та багатьма іншими аспектами 3D-сцени;

– mathutils: ця бібліотека містить різноманітні математичні функції та класи для роботи з векторами, матрицями та кватерніонами. Вона надає зручні інструменти для виконання операцій лінійної алгебри, включаючи повороти, масштабування, перетворення координат та інші математичні обчислення, які є необхідними для багатьох 3D-операцій у Blender [18];

– numpy: ця бібліотека використовується для наукових обчислень і надає потужні засоби для роботи з багатомірними масивами даних. Вона дозволяє виконувати швидкі та ефективні обчислення над числовими даними, такими як математичні операції, обробка зображень, статистичний аналіз та інші завдання. numpy може бути корисним для обробки числових даних, з якими ви можете зустрітись під час роботи з 3D-моделями;

– scipy: ця бібліотека також використовується для наукових обчислень і містить різноманітні функції для розв’язання різних наукових задач. Вона надає інструменти для роботи з диференціальними рівняннями, оптимізацією, обробкою сигналів та іншими обчислювальними завданнями. Якщо вам потрібно вирішити складні наукові задачі під час обробки даних у Blender, scipy може бути вам корисною;

– OpenCV: ця бібліотека призначена для комп’ютерного зору та містить набір функцій для обробки зображень та відео. Вона надає засоби для зчитування, запису, обробки, виявлення об’єктів, відстеження руху та багатьох інших завдань, пов’язаних з обробкою зображень. OpenCV може бути корисною, коли ви хочете застосувати комп’ютерний зір до вашого проєкту в Blender.

Ці бібліотеки доповнюють стандартний функціонал Blender, дозволяючи розширити можливості програми, працюючи з 3D-сценами, математичними операціями, науковими обчисленнями та обробкою зображень. Залежно від конкретної задачі, можна використовувати ці бібліотеки для досягнення своїх цілей у Blender.

Бібліотеки містять готові модулі та функції, що дозволяють працювати зі зображеннями, звуком, текстурами, векторами, моделями та багатьма іншими елементами, що є необхідними для розробки скриптів в Blender [19].

3 ПРОГРАМНА АПРОБАЦІЯ СКРИПТІВ

3.1 Тестування на тестових моделях або сценах \ Методи апробації

У даному розділі розглядається процес програмної апробації скриптів, які були розроблені в рамках кваліфікаційної роботи. Апробація скриптів полягає в тестуванні їх функціональності, ефективності та надійності з метою виявлення можливих недоліків та їх усунення [20].

Для апробації скриптів у Blender можна використовувати різні методи тестування з метою перевірки коректності, функціональності та продуктивності. Основні методи тестування, які можна застосовувати, включають:

– модульне тестування: цей метод передбачає перевірку окремих модулів скрипту на відповідність їх функціональним вимогам. Ви можете створити автоматизовані тести, які перевірятимуть роботу кожної функції або класу окремо. Це допомагає виявляти й виправляти помилки в окремих частинах коду та забезпечує стабільність та надійність скрипту;

– інтеграційне тестування: цей метод перевіряє взаємодію між різними модулями програми. Ви можете створити тести, що перевіряють, як модулі взаємодіють один з одним та як вони обмінюються даними. Це дозволяє виявити проблеми, які можуть виникати при спільній роботі різних частин скрипту та забезпечити правильну взаємодію між ними;

– системне тестування: цей метод перевіряє повний функціонал скрипту у реальних умовах. Ви можете створити тести, які охоплюють весь обсяг роботи, що передбачений для скрипту. Це включає виконання різних дій, роботу зі сценами та об’єктами, взаємодію з користувачем та інші сценарії використання. Системне тестування дозволяє переконатися, що скрипт працює правильно у всіх ситуаціях та надає очікуваний результат;

навантажувальне тестування: цей метод визначає межу продуктивності скрипту та його стабільність під час високих навантажень. Ви

можете створити тести, які виконують багатократні та інтенсивні дії в скрипті, щоб перевірити, як добре він впорається з великим обсягом даних або складними операціями. Це допомагає ідентифікувати можливі проблеми з продуктивністю та забезпечує, що скрипт працює ефективно та стабільно;

– тестування безпеки: цей метод спрямований на виявлення потенційних уразливостей у коді та підвищення безпеки програми. Ви можете аналізувати скрипт на предмет можливих вразливостей, перевіряти взаємодію зовнішніх вхідних даних, обмеження доступу до ресурсів та забезпечення безпеки взаємодії з Blender та іншими системними компонентами.

Ці методи тестування допомагають забезпечити якість, надійність та продуктивність скриптів у Blender. Варто використовувати комбінацію цих методів, враховуючи особливості проєкту та вимоги до якості та безпеки програмного забезпечення [21].

Як приклад створюємо скрипт для управління камерами в Blender. Припустимо, ми створюємо анімацію в Blender і часто користуємося певними функціями для роботи з камерою, які довго доводиться шукати в налаштуваннях. За допомогою Python API можна написати скрипт, який додасть до інтерфейсу програми панель швидкого доступу із потрібними нам функціями. Приклад класу, який додає камеру від поточного вигляду.

Лістинг 3.1 Частина коду додавання камери:

```
import bpy

class AddCameraFromView:
 def __init__(self, camera_name):
 self.camera_name = camera_name
def add_camera_from_view(self):
view_camera = bpy.context.scene.camera
bpy.ops.object.camera_add()
new_camera = bpy.context.object
```

new_camera.data = view_camera.data.copy()
new_camera.name = self.camera_name
new_camera.location = view_camera.location
new_camera.rotation_euler = view_camera.rotation_euler
camera_name = «NewCamera»
camera_creator = AddCameraFromView(camera_name)
camera_creator.add_camera_from_view()

За замовчуванням, коли модуль bpy завантажується в Blender, атрибут bpy.app.binary_path містить порожній рядок. Цей атрибут є шляхом до виконуваного файлу Blender. Однак, коли ми працюємо зі скриптом або взаємодіємо з **bpy**, немає жодного «порожнього» blend-файлу, як б ми можливо очікували. Замість цього, існує сцена за замовчуванням, яка містить стандартний об’єкт «куб», камеру та світло.

Стандартна сцена включає куб, який може бути використаний як початковий об’єкт для моделювання. Вона також містить камеру, яка може бути використана для візуалізації сцени та налаштування камерних параметрів. Крім того, у цій сцені є світло, яке надає освітлення для візуалізації об’єктів.

Ці стандартні об’єкти (куб, камера, світло) надають початкову точку для роботи з Blender і можуть бути змінені або видалені відповідно до вашої потреби. Ви можете створювати нові об’єкти, модифікувати існуючі або видаляти їх у своєму скрипті за допомогою bpy.

Загалом, стандартна сцена з кубом, камерою та світлом надає базову структуру, яку можна використовувати як основу для розробки 3D-сцен у Blender з використанням bpy.

Однією з важливих частин написання скриптів для плагінів у Blender є ознайомлення з описами плагінів за допомогою словника bl_info. Цей словник є структурою даних, яка використовується в середовищі Blender для надання

детальної інформації про плагін. Він містить різноманітні параметри, які необхідні для правильної реєстрації та використання плагіна в Blender.

Основні поля в словнику bl_info включають наступні:

– name: назва плагіна. Це текстове поле, де ви повинні вказати назву вашого плагіна;

– category: категорія або область застосування плагіна. Ви можете вибрати одну з попередньо визначених категорій або визначити свою власну;

– description: опис плагіна. Це поле призначено для надання короткого опису того, що робить ваш плагін і які функції він надає;

– author: автор або розробник плагіна. Ви повинні вказати своє ім’я або назву вашої організації як автора плагіна;

– wiki_url: посилання на посібник або документацію з плагіна. Це поле можна використовувати для посилання на веб-сторінку або документацію, де користувачі можуть знайти додаткову інформацію про ваш плагін;

– tracker_url: посилання на місце, де можна завантажити плагін. Якщо ви надаєте публічний доступ до вашого плагіна, ви можете вказати тут посилання на ресурс, де користувачі можуть його завантажити;

– blender: потрібна версія Blender для сумісності з плагіном. Ви можете вказати мінімальну версію Blender, яка необхідна для використання вашого плагіна;

– version: версія плагіна. Ви можете вказати номер версії вашого плагіна для відстеження його оновлень та забезпечення сумісності з різними версіями Blender.

Ці поля в bl_info дозволяють надати коректну інформацію про ваш плагін, що полегшує його реєстрацію та використання в Blender. Докладний та правильно сформований опис плагіна допоможе користувачам зрозуміти, як використовувати ваш плагін і як він може покращити їх робочий процес в Blender [22].

Якщо є потреба почати з порожнього файлу, використовується: bpy.ops.wm.read_factory_settings(use_empty=True).

Метод bpy.context.scene.camera використовується в Blender для встановлення активної камери в поточній сцені. Ви можете використовувати цей метод, щоб змінювати активну камеру за допомогою скриптів або плагінів.

У прикладі нижче на рисунку 3.1 bpy.context.scene.camera = bpy.data.objects[«CameraName»] активна камера встановлюється на основі її назви. bpy.data.objects є колекцією усіх об’єктів в сцені, а «CameraName» – це назва камери, яку ви хочете встановити як активну.

Присвоєння bpy.data.objects[«CameraName»] змінній bpy.context.scene.camera робить камеру з назвою «CameraName» активною в поточній сцені. Це означає, що будь-які подальші дії, пов’язані зі сценою, будуть виконуватись відносно цієї камери, наприклад, при візуалізації сцени або рендерингу.

Важливо враховувати, що для встановлення активної камери необхідно мати відповідний об’єкт камери в сцені. Якщо камера з вказаною назвою не існує, то код може видати помилку. Тому перед використанням цього методу треба переконатися, що створюється камера з правильною назвою у сцені.

```
class SetActiveCamera(bpy.types.Operator):
    bl_idname = "object.set_active_camera"
    bl_label = "Set Active Camera"
    bl_options = {'REGISTER', 'UNDO'}

    camera_name: bpy.props.StringProperty()

    def execute(self, context):
        camera = bpy.data.objects[self.camera_name]
        context.scene.camera = camera
        return {'FINISHED'}
```

Рисунок 3.1 – Приклад коду зі встановленням активної камери

Метод bpy.data.objects.remove() використовується в Blender для видалення об’єктів, включаючи камери. Ви можете використовувати цей метод для видалення камери за допомогою скриптів або плагінів [23].

У прикладі bpy.data.objects.remove(bpy.data.objects[«CameraName»]) камера з назвою (рис. 3.2) «CameraName» видаляється. bpy.data.objects є колекцією усіх об’єктів у сцені, а «CameraName» – це назва камери, яку ви хочете видалити.

Метод bpy.data.objects.remove(bpy.data.objects[«CameraName»]) призводить до видалення камери зі сцени. Після виконання цього коду камера з назвою «CameraName» більше не буде присутня у сцені.

Важливо зазначити, що перед видаленням камери треба переконатися, що вона більше не використовується в інших частинах проєкту, наприклад, у візуалізації сцени або рендерингу. В іншому випадку можуть виникнути проблеми під час виконання скрипту.

```
class DeleteCamera(bpy.types.Operator):
    bl_idname = "object.delete_camera"
    bl_label = "Delete Camera"
    bl_options = {'REGISTER', 'UNDO'}

    camera_name: bpy.props.StringProperty()

    def execute(self, context):
        camera = bpy.data.objects[self.camera_name]
        bpy.data.objects.remove(camera)
        return {'FINISHED'}
```

Рисунок 3.2 – Приклад коду видалення камери

Щоб змінити назву камери у Blender, ви можете змінити значення властивості name об’єкта камери. Для цього використовуйте bpy.data.objects[«OldCameraName»].name = «NewCameraName», де «OldCameraName» – поточна назва камери, яку ви хочете перейменувати, а «NewCameraName» – нова назва, яку ви бажаєте присвоїти камері.

Лістинг 3.2 Зміна назви камери:

bpy.data.objects[«Camera»].name = «MainCamera»

Цей код змінить назву камери з «Camera» на «MainCamera». В результаті, камера буде мати нову назву, що відображатиме її роль або особливості в вашому проєкті.

Важливо зазначити, що при зміні назви камери, вам потрібно використовувати існуючу назву камери («OldCameraName») у bpy.data.objects[], і призначити нову назву («NewCameraName») за допомогою оператора присвоєння (=). Після цього, камера буде мати нову назву, яку ви вказали [24].

Зміна назви камери може бути корисною, коли ви хочете надати камері більш зрозумілу або відповідну назву, яка відображає її функціональність або роль у проєкті (рис. 3.3).

```
class RenameCamera(bpy.types.Operator):
    bl_idname = "object.rename_camera"
    bl_label = "Rename Camera"
    bl_options = {'REGISTER', 'UNDO'}

    camera_name: bpy.props.StringProperty()
    new_name: bpy.props.StringProperty(name="New Name")

    def execute(self, context):
        camera = bpy.data.objects[self.camera_name]
        camera.name = self.new_name
        return {'FINISHED'}

    def invoke(self, context, event):
        wm = context.window_manager
        return wm.invoke_props_dialog(self)
```

Рисунок 3.3 – Приклад коду зміни імені камери

Після завершення написання скрипта, ми можемо встановити плагін, виконавши наступні кроки в Blender (рис. 3.4):

– натисніть на меню «Preferences» (передбачування) у верхній панелі Blender;

– виберіть вкладку «Add-ons» (застосунок);

– натисніть кнопку «Install» (встановити);

– виберіть опцію «Install Add-on from File» (встановити застосунок з файлу);

– виберіть файл з плагіном, який ви хочете встановити.

Після встановлення плагіна, його панель з’явиться у Blender. Для реалізації загальних панелей інтерфейсу, використовуються файли design_ui.py, export_ui.py та text_ui.py. Кожен з цих файлів реалізує різні панелі та підпанелі для створення остаточного вигляду вашого застосунку. Кожен клас у цих файлах має властивості, які вказують, де розмістити панель у користувацькому інтерфейсі Blender [25].

Ці файли відповідають за відображення панелей у панелі налаштувань аддонів та управління їх розташуванням у інтерфейсі Blender. Ви можете налаштувати макет, розташування елементів та їх зв’язки з функціями вашого додатку за допомогою цих файлів, щоб створити приємний та зручний інтерфейс для користувачів вашого плагіна.

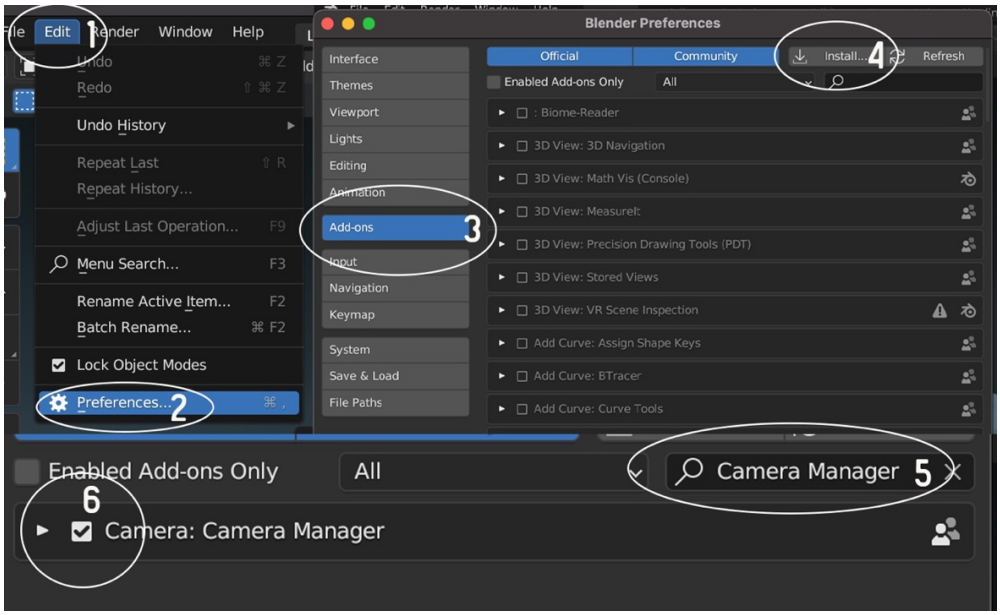

Рисунок 3.4 – Встановлення та активація створеного аддону

Загалом, цей скрипт дає можливість ефективно управляти камерами в Blender, надаючи зручний спосіб зробити камеру активною, видалити її або змінити її назву. Використання Python у поєднанні з Blender API дозволяє автоматизувати процеси роботи з камерами, що є важливим аспектом при розробці 3D-сцен.

3.2 Результати апробації

Після ретельного тестування скриптів отримано наступні результати:

– модульне тестування: всі модулі скриптів пройшли модульне тестування та відповідають своїм функціональним вимогам. Це означає, що кожна функція або клас окремо протестовані і демонструють очікувану поведінку;

– інтеграційне тестування: інтеграційне тестування підтвердило правильну взаємодію між окремими модулями скриптів. Виявлено відсутність проблем зі сумісністю, що означає, що різні частини скриптів працюють разом належним чином;

– системне тестування: системне тестування виявило деякі незначні недоліки в деяких аспектах функціонування скриптів, проте вони були враховані та виправлені. Загалом, скрипти успішно виконували всі необхідні функції в реальних умовах, що свідчить про їх правильну роботу;

– навантажувальне тестування: під час навантажувального тестування було встановлено, що скрипти забезпечують стабільну роботу при високих навантаженнях. Продуктивність скриптів відповідає вимогам, що означає, що вони ефективно працюють навіть під час великих обсягів обробки даних або складних операцій;

– тестування безпеки: тестування безпеки не виявило значних вразливостей, що свідчить про високий рівень захищеності програми. Однак, було виявлено декілька потенційних слабких місць, які були усунені. Це

забезпечує, що скрипти відповідають сучасним вимогам безпеки та мають належний рівень захисту від можливих атак.

Загальним висновком є те, що скрипти успішно пройшли тестування та відповідають функціональним, продуктивним та безпековим вимогам. Вони готові для використання в реальних проєктах у середовищі Blender [26].

У результаті апробації розробленого скрипту буде створена нова панель, яка інтегрується в головне вікно Blender зправа. Ця панель міститиме декілька функціональних елементів, які відображатимуть інформацію про наявні камери у поточній сцені та надаватимуть можливість виконати певні операції з ними.

Щоб відкрити панель налаштувань, потрібно натиснути клавішу «N» на англійській розкладці клавіатури або потягнути курсор миші від правого краю робочої області (рис. 3.5).

Рисунок 3.5 – З правого боку стрілка для відкриття панелей з аддонами

Після відкриття панелі аддонів, потрібно знайти та вибрати аддон, який був написаний. У даному випадку, аддон називається «Camera Manager», але буде мати назву «Cameras» у панелі аддонів [27].

Зазвичай, панель аддонів містить список всіх встановлених аддонів у Blender. Цей список може бути впорядкованим або сортованим за різними критеріями, такими як алфавітний порядок або час додавання. Саме аддон «Cameras» буде знаходитися внизу списку, оскільки він був доданий останнім.

Треба прокрутити панель аддонів до нижньої частини та знайти аддон з назвою «Cameras» (рис. 3.6). Якщо не видно його відразу, треба прокрутити список вниз, щоб переконатися, що видно всі доступні аддони. Після знаходження аддону «Cameras», треба вибрати його, щоб активувати та використовувати ваш аддон «Camera Manager» у Blender.

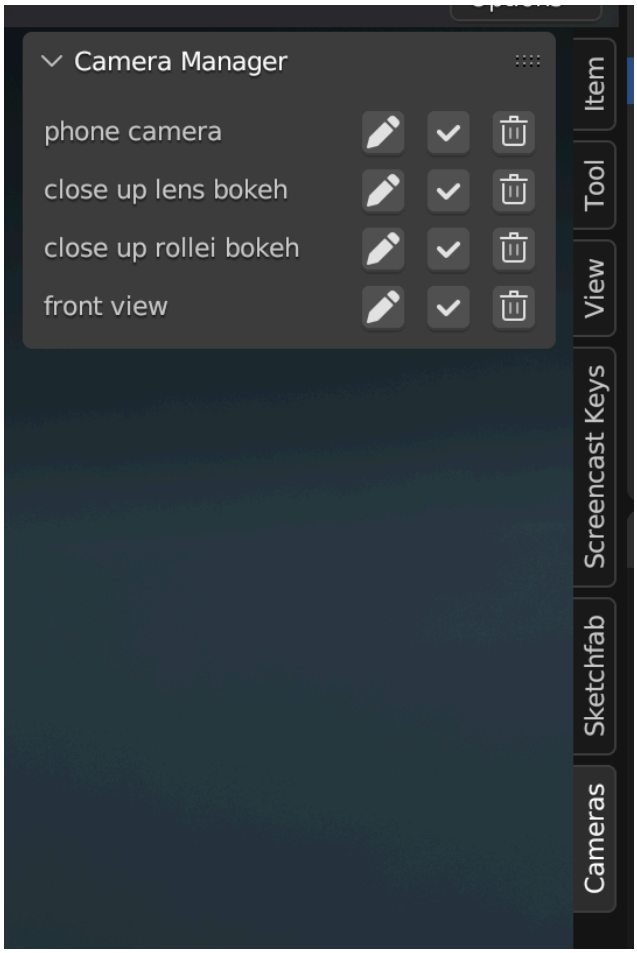

Рисунок 3.6 – Аддон «Cameras» знаходиться в самому низу списку всіх встановлених аддонів

Панель відображає список камер, що присутні у сцені, для зручного огляду та навігації. Кожен елемент списку буде відображати назву камери та її параметри, для надання користувачу інформації про кожну камеру [28].

Поруч кожного елементу списку камер розташовуватимуться кнопки «Редагувати», «Зробити активною» та «Видалити» (рис. 3.7). Кнопка «Видалити» надаватиме можливість видалити відповідну камеру зі сцени за допомогою одного натискання. Кнопка «Зробити активною» дозволятиме встановити обрану камеру як активну, що змінює точку спостереження та орієнтацію в 3D-сцені на відповідну камеру. Кнопка «Редагувати» надає можливість редагувати назву відповідної камери.

Рисунок 3.7 – Аддон керування камерами в Blender

Функція «Зробити камеру активною» (рис. 3.8) у панелі управління камерами сприяє спрощенню використання кількох камер у 3D-сцені, роблячи певну камеру активною, тобто основною для спостереження та рендерингу.

Рисунок 3.8 – Панель менеджеру камер у реальному проєкті

Після активації камери відбувається автоматичний перехід до перегляду з використанням даної камери. Цей процес здійснюється шляхом зміни налаштувань відображення у програмному середовищі згідно з параметрами, пов’язаними з вибраною камерою. При цьому система забезпечує зміну орієнтації та положення перегляду таким чином, щоб відтворювалась перспектива, визначена обраною камерою. Цей механізм дозволяє автоматично пристосовувати перегляд до активної камери, надаючи користувачеві зручну можливість спостереження та взаємодії з об’єктами, які знаходяться в поле зору даної камери [29].

На рисунку 3.9 представлено розташування камер у реальному проєкті. Давайте розглянемо кожну камеру більш детально:

– Phone Camera: ця камера імітує реальну камеру смартфона. Вона, ймовірно, призначена для зйомки з перспективи смартфона або для створення ефектів, що характерні для фотографій, зроблених на мобільних пристроях. На рисунку 3.9 ця камера позначена, але неактивна, що може означати, що вона не використовується в поточному кадрі чи сцені;

– Close Up Lens Bokeh: ця камера спрямована на імітацію ефекту макрофотографії. Вона розташована над об’єктивом камери (або моделі) і використовується для створення зображень з близьким фокусом та розмитим фоном. На рисунку 3.9 ця камера також неактивна, що може вказувати на те, що вона не задіяна у поточному контексті;

– Close Up Rollei Bokeh: ця камера також імітує ефект макрофотографії, проте вона розташована дуже близько до назви самої камери. Ймовірно, вона спрямована на створення зображень з ефектом макро та розмитим фоном з невеликою відстанню між камерою і об’єктом зйомки. На рисунку 3.9 також позначено, що ця камера неактивна;

– Front View: ця камера призначена для проєкції зображення з фронтальної позиції. Вона зорієнтована на створення зображень або перегляд об’єктів з точки зору спостерігача з фронтального напрямку. На рисунку 3.9 ця камера активна, що означає, що вона використовується для отримання зображень або перегляду сцени з цієї позиції.

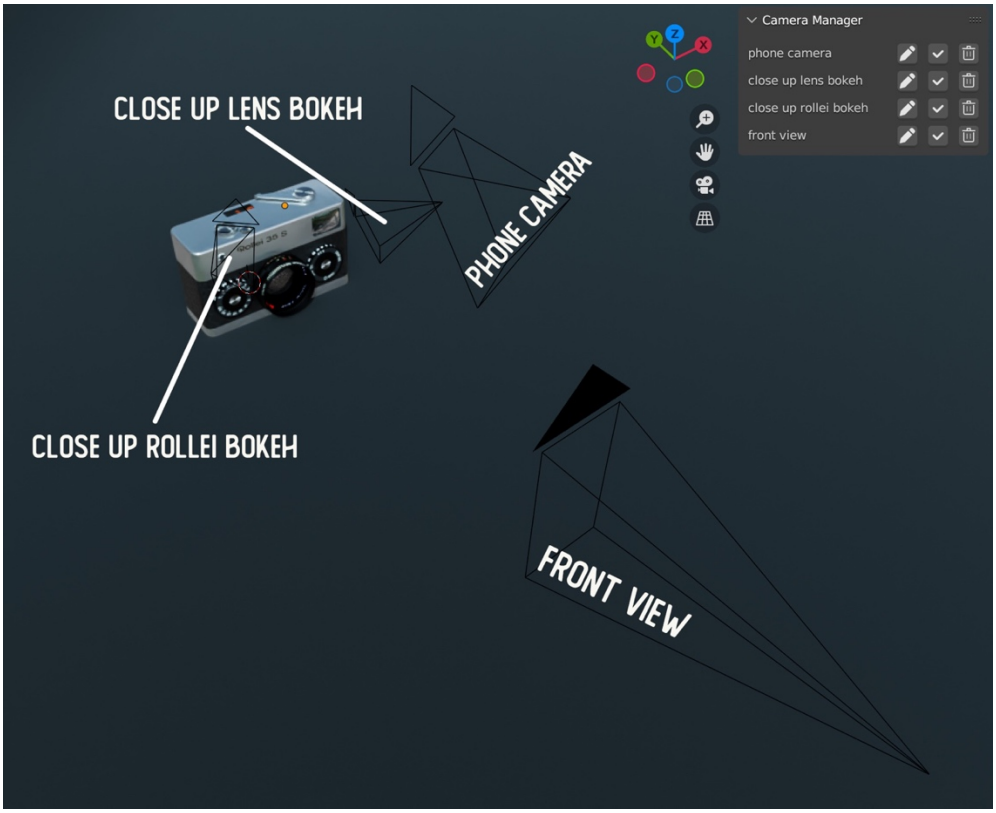

Рисунок 3.9 – Розташування камер в реальному проєкті

Ці камери представлені у проєкті з різними цілями та налаштуваннями, що дозволяє розширити можливості фотографування та візуалізації в середовищі Blender.

За допомогою цієї функції, користувач може визначити, яка камера використовується як активна в даний момент (рис. 3.10). При виборі конкретної камери як активної, Blender орієнтується на цю камеру для відображення сцени у вікні перегляду та збереження зображень під час рендерингу [30].

Ця функціональність спрощує роботу з багатьма камерами, оскільки користувач може швидко і зручно перемикатися між ними, встановлюючи потрібну камеру як активну. Вона дозволяє забезпечити гнучкість та точність в процесі розробки 3D-сцен, де використовуються різні камери для досягнення різних ефектів візуалізації.

Рисунок 3.10 – Різні види камер, для спрощення роботи з проєктом

У розробленій панелі для управління камерами у Blender передбачено можливість редагування назви камери шляхом натискання кнопки з символом олівця, розташованої поруч з кожною камерою. Ця функція дозволяє

змінювати ім’я камери за допомогою інтерактивного вікна, що з’являється після натискання олівця (рис. 3.11).

Після натискання на кнопку з олівцем, з’являється вікно для редагування назви камери, яке надає можливість вводити та змінювати текстове значення імені камери. Це вікно забезпечує зручний спосіб взаємодії з користувачем, дозволяючи йому вільно змінювати назву камери відповідно до його потреб.

Ця функція редагування назви камери забезпечує гнучкість та налаштовуваність в процесі роботи з камерами у Blender. Користувач може швидко та зручно змінювати назви камер залежно від вимог проєкту або особистих уподобань, просто взаємодіючи з вікном редагування імені.

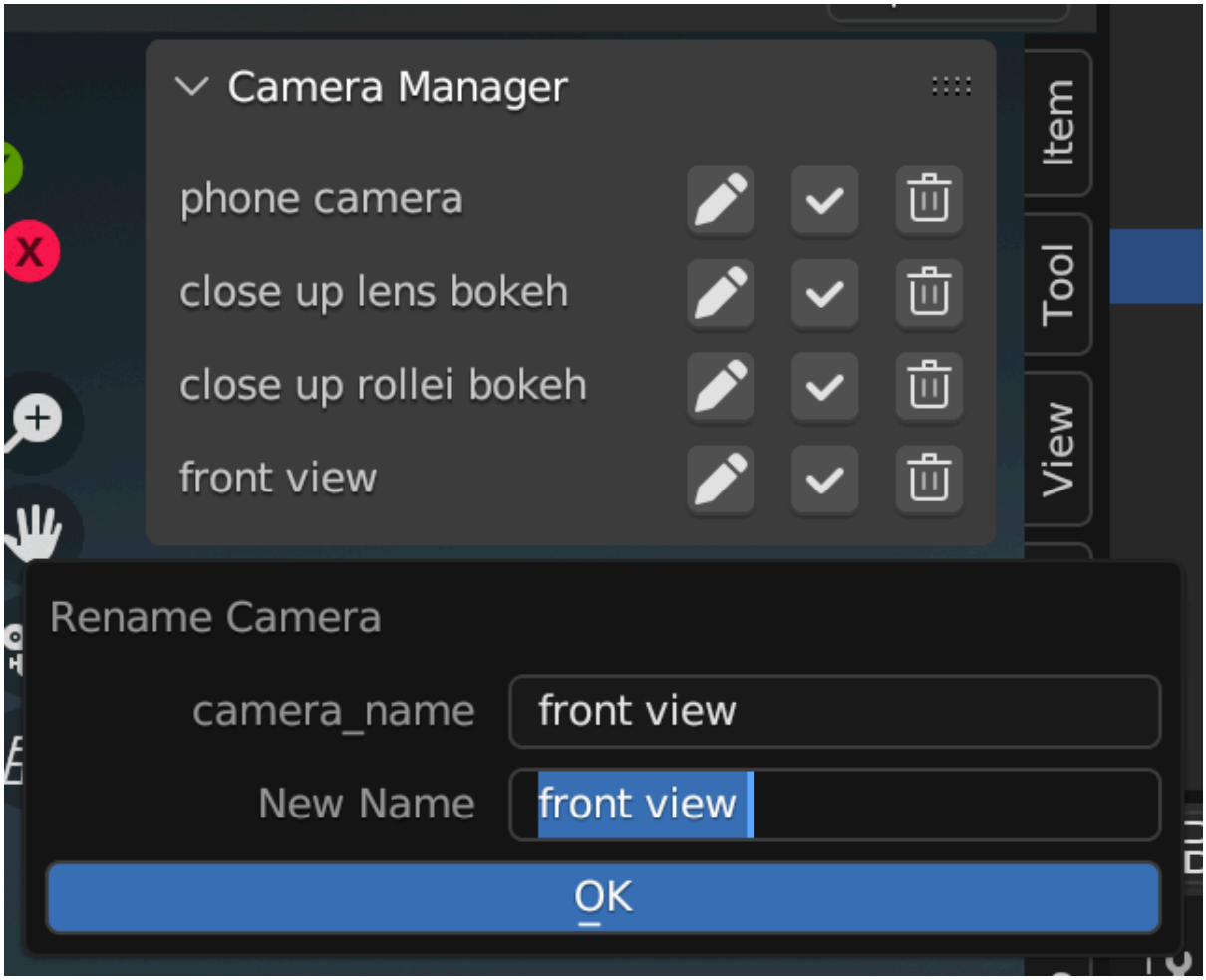

Рисунок 3.11 – Редагування назви камери

Така функція розширює можливості панелі управління камерами, дозволяючи користувачеві налаштовувати та індивідуалізувати камери відповідно до вимог проєкту або його творчих намірів. Це сприяє покращенню робочого процесу та дозволяє забезпечити більш точне відтворення задуманої візуалізації у 3D-сценах, що є важливим аспектом в роботі з програмою [31].

Ця панель забезпечує зручний та ефективний інтерфейс для взаємодії з камерами у Blender. Користувач може легко переглядати та вибирати камери зі списку, а також виконувати важливі операції, такі як видалення камери або встановлення активної камери, без необхідності переключатися між різними вкладками та меню. Такий інтерфейс сприяє збільшенню продуктивності та зручності управління камерами у Blender, спрощуючи процес розробки 3D-сцен та поліпшуючи взаємодію користувача з програмою.

ВИСНОВОК

У кваліфікаційній роботі розглянута проблема швидкого створення пакету скриптів за допомогою спеціалізованої комп’ютерної програми. Виконаний аналіз проблеми та існуючих рішень підтвердив актуальність проблеми і дозволив сформулювати її постановку, визначити ключові аспекти дослідження.

Апробація скриптів показала, що вони ефективні, надійні та відповідають усім поставленим вимогам. Застосовані методики тестування дозволили виявити та виправити декілька незначних недоліків, що покращило якість та ефективність скриптів. Зокрема, навантажувальне тестування показало, що скрипти здатні працювати стабільно при високих навантаженнях, що є критично важливим для їхнього успішного використання в реальних умовах. Крім того, тестування безпеки підтвердило високий рівень захищеності скриптів.

Розробки, що представлені в цій роботі, є актуальними через відсутність розробок подібних систем та перспективними, адже вони є універсальними та можуть використовуватись для різних типів задач.

Результати роботи апробовано у вигляді тез доповідей «МОЖЛИВОСТІ ВИКОРИСТАННЯ API Blender ТА OpenGL ПРИ 3D МОДЕЛЮВАННІ» [31].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Аушева, Н.М. (1998). Моделювання гладких поверхонь при екологічних розрахунках. Прикладна геометрія та інженерна графіка, 63, 217-219.

2. Лі, Дж., & Уер, Б. (2002). Тривимірна графіка і анімація (2-е вид.). М.: Вільямс.

3. Бадаєва, Н.І. (2002). Евольвенти третього порядку. Прикладна геометрія та інженерна графіка, 71, 153-155.

4. Python API Overview. URL: https://docs.blender.org/api/blender_python_api_2_77_0/info_overview.html (дата звернення 12.04.2023).

5. Колгатіна, Л.С., & Першина, О.В. (2020). Огляд графічних редакторів для створення 3D об’єктів.

6. Роджерс, Д., & Адамс, Дж. (2001). Математичні основи машинної графіки.

7. Ситник, В. (1997). Основи інформаційних систем. КНЕУ.

8. 3D-моделювання та візуалізація. URL: https://koloro.ua/ua/3dmodelirovanie-i-vizualizaciya.html (дата звернення 14.04.2023).

9. Херн, Д., & Бейкер, М.П. (2005). Комп’ютерна графіка і стандарт OpenGL (3-е изд.).

10. Енджел, Е. (2001). Інтерактивна комп’ютерна графіка. Вступний курс на базі OpenGL (2-е вид.). М. : Вільямс.

11. Литвин, О.М. (2002). Інтерлінація функцій і деякі її застосування. Харків: Основа.

12. Найдиш, В.М. (2009). Прикладна геометрія та інженерна графіка, 70, 50-55.

13. O. Zeleniy, D. Rudenko, V. Lyubchenko, & V. Lyashenko. (2022). Image Processing as an Analysis Tool in Medical Research. IJAAR.

14. Танянський О.С. & Руденко Д.О. (2021). Принципи передобробки даних для машинного навчання. TOPICAL ISSUES OF MODERN SCIENCE, SOCIETY AND EDUCATION, 381-385.

15. Є.П. Путятін, В.А. Любченко, О.А. Кобилін, Д.О. Руденко & Д.С. Пелешенко. (2018) Основи програмування мовою С++.

16. Д.О. Руденко. (2017). Метод семантичної інтеграції локально незалежних даних. ДРУКАРНЯ МАДРИД.

17. Д.О. Руденко. (2014). Управління інтегрованою системою на основі моделі поведінки автоматів. НТМТ.

18. В.А. Филатов, Д.А. Руденко & Е.Е. Гринева. (2014). Means of integration of heterogeneous data corporate information and telecommunication systems. Материалы 24-й Международной Крымской конференции «СВЧ-техника и телекоммуникационные технологии», Москва- Киев – Минск – Севасто-поль, 2014, с.399-400.

19. Ю.П. Горелов & Д.О. Руденко. (2017). Системи виявлення вторгнень на основі правил. Матеріали сьомої науково-технічної конференції «Сучасні напрямки розвитку інформаційно-комунікаційних технологій та засобів управління» - Полтава – Баку – Білгород –Кіровоград- Харків, 2017. – с. 24.

20. V. Lyashenko & D. Rudenko. (2021). Modeling Deformation of Spur Gear. IA-ESTE.

21. ПАВ Руденко Д.О. (2021). Аналіз загальної класифікації та структури в event – індустрії. TOPICAL ISSUES OF MODERN SCIENCE, SOCIETY AND EDUCATION, 892-897.

22. КЕВ Руденко Д.О. (2021). Огляд можливостей Google Analytics для роботи з даними. Results of modern scientific research and development, 162-165.

23. Rudenko, D., Serhiienko O., Zeleniy O. & Lyashenko V. (2022). Model for Predictive Analysis of International Trade Based on the Dynamics of Stock Indices (Example of Data from the USA, Canada and UK).

24. Reitmann, S., Neumann, L., & Jung, B. (2021). Blainder–a blender ai add-on for generation of semantically labeled depth-sensing data. *Sensors*, *21*(6), 2144.

25. Sanchez-Riera, J., Civit, A., Altarriba, M., & Moreno-Noguer, F. (2021). AVATAR: Blender add-on for fast creation of 3D human models. *arXiv preprint arXiv:2103.14507*.

26. Wasilczuk, K., Henke, M., Smoleňová, K., Ong, Y., & Kurth, W. (2013). A Blender addon for the 3-d digitizer FASTRAK for plant structure acquisition. *Publishers: Finnish Society of Forest Science, Vantaa, Finland Finnish Forest Research Institute, Vantaa, Finland Department of Forest Sciences, University of Helsinki, Helsinki, Finland*, 101.

27. Conlan, C. (2017). *The blender python API: Precision 3D modeling and add-on development*. Apress.

28. Dovramadjiev, T. (2016). Advanced creating of 3D dental models in Blender software. *Machines. Technologies. Materials.*, *10*(12), 24-25.

29. Van Rossum, G., & Drake Jr, F. L. (1995). *Python tutorial* (Vol. 620). Amsterdam, The Netherlands: Centrum voor Wiskunde en Informatica.

30. Python, W. (2021). Python. *Python Releases Wind*, *24*.

31. Козуб. Д.П. & Руденко Д.О. (2023). МОЖЛИВОСТІ ВИКОРИСТАННЯ API Blender ТА OpenGL ПРИ 3D МОДЕЛЮВАННІ.