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)

РЕАЛІЗАЦІЯ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ЗА ДОПОМОГОЮ PYTHON У 3D-ПАКЕТІ BLENDER
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-3
Колісниченко М.В.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник ст. викл. Путятіна О.Є.
(посада, прізвище, ініціали)

Допускається до захисту

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

2024 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)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Колісниченко Марії В'ячеславівні
(прізвище, ім'я, по батькові)

1. Тема роботи Реалізація процедурної генерації за допомогою Python у 3D-пакеті Blender

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 27 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, 3D-пакет Blender.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області та огляд API Blender.
2. Огляд процедурної генерації по пармаметрах.
3. Розробка аддону процедурної генерації за допомогою Python у 3D-пакеті Blender.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-15.04.24	
3	Аналіз літератури з досліджуваної проблеми	16.04.24-18.04.24	
4	Аналіз технічних засобів	19.04.24-25.04.24	
5	Розробка методу	26.04.24-09.05.24	
6	Програмна реалізація	10.05.24-23.05.24	
7	Оформлення пояснювальної записки	24.05.24-26.05.24	
8	Перевірка на плагіат	28.05.24	
9	Рецензування	29.05.24	
10	Підготовка презентації та доповіді	30.05.24-05.06.24	
11	Занесення роботи в електронний архів	06.06.24	
12	Попередній захист кваліфікаційної роботи	06.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ ___ ст. викл. Путятіна О.Є.____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 65 с., 1 табл., 27 рис., 31 джерело.

ПРОГРАМУВАННЯ PYTHON, 3D-МОДЕЛЮВАННЯ, BLENDER, ПРОЦЕДУРНА ГЕНЕРАЦІЯ, ГЕОМЕТРИЧНІ СТРУКТУРИ, АДДОН BLENDER.

Об’єктом роботи є створення інтерактивного аддону для 3D-пакету Blender, який дозволить користувачам створювати та маніпулювати складними геометричними структурами.

Метою роботи є розробка інструменту, який спростить процес створення складних 3D-моделей у Blender за допомогою процедурної генерації та динамічних параметрів.

Тематикою кваліфікаційної роботи є дослідження можливостей використання Python у Blender для створення інтерактивних інструментів, аналіз методів процедурної генерації геометричних об’єктів, розробка та тестування аддону з використанням цих методів.

У результаті роботи здійснена розробка аддону для 3D-пакету Blender за допомогою мови програмування Python.

PYTHON PROGRAMMING, 3D-MODELLING, BLENDER, PROCEDURAL GENERATION, GEOMETRIC STRUCTURES, BLENDER ADD-ONS.

The object of the work is the creation of an interactive addon for the Blender 3D package, which will allow users to create and manipulate complex geometric structures.

The goal of the work is to develop a tool that will simplify the process of creating complex 3D models in Blender using procedural generation and dynamic parameters.

The topic of the qualification work is the study of the possibilities of using Python in Blender to create interactive tools, the analysis of methods of procedural generation of geometric objects, the development and testing of an addon using these methods.

As a result, an addon for the Blender 3D package was developed using the Python programming language.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface (набір визначень та протоколів, які визначають, як програми або компоненти програм та можуть взаємодіяти один з одним)

Рендеринг – це процес обчислення та створення зображення з тривимірної сцени або моделі за допомогою комп’ютерної графіки

Blender Add-on – це додатковий модуль або розширення для програмного забезпечення Blender

ПЗ – програмне забезпечення

ПО – предметна область

ВСТУП

У сучасному світі програмне забезпечення для роботи з 3D-графікою [1] є необхідним інструментом для багатьох професійних і творчих завдань. В цьому контексті особливою популярністю користується Blender – вільний та потужний інструмент, призначений для створення тривимірних моделей, анімації, візуалізації та рендерингу. Відмінною особливістю Blender є його відкритий код та активна спільнота розробників, які постійно розширюють його можливості.

Однією з ключових можливостей Blender є його API, який відкриває безліч можливостей для розширення функціональності програми. Використовуючи мову програмування Python, можна створювати додаткові модулі (аддони), що значно розширюють можливості Blender за допомогою нових інструментів, функцій та інтерфейсів.

Мета цієї кваліфікаційної роботи полягає в розробці функціонального аддону для Blender на мові програмування Python. Основне завдання – створення інструменту, який підвищить продуктивність та ефективність роботи з Blender. Для досягнення цієї мети будуть використані передові методи програмування, знання API Blender, а також розуміння потреб користувачів у зручних та потужних інструментах для створення 3D-графіки.

Актуальність даної роботи виявляється у поєднанні потужності Blender та гнучкості мови програмування Python. Створення функціонального аддону для Blender – це не тільки можливість збільшити його функціональність, але й дозвіл на створення індивідуальних інструментів для творчих і технічних завдань, що забезпечує більш гнучкий та ефективний підхід до роботи з програмою.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФУНКІОНАЛУ API BLENDER

1.1 Тривимірне моделювання та його важливість у сучасному світі

Тривимірне моделювання є комплексним процесом, що передбачає створення об’єктів і сцен у тривимірному просторі за допомогою спеціалізованих програм та технологій. Цей підхід знайшов широке застосування в різних сферах індустрії та культури, включаючи наступні:

– інженерія та архітектура: тривимірне моделювання дозволяє інженерам та архітекторам створювати віртуальні прототипи будівель, машин, транспорту та інших технічних об’єктів. Це сприяє вдосконаленню проєктів, зменшенню часу розробки і витрат, а також дозволяє виявляти можливі проблеми до початку фізичного виробництва;

– медицина: у медичній галузі тривимірне моделювання використовується для створення віртуальних моделей органів та систем організму. Це дозволяє лікарям здійснювати точнішу діагностику, планувати хірургічні втручання та навчати студентів медичних установ;

– дизайн та мистецтво: у творчих галузях тривимірне моделювання використовується для створення вражаючих візуальних ефектів, об’ємних композицій та артпроєктів. Від графічного дизайну до виробництва інтерактивних ігор, це інструмент для реалізації творчих ідей;

– розваги та ігрова індустрія: тривимірне моделювання є основою для створення візуально захоплюючих ігор, анімаційних фільмів та віртуальної реальності. Воно дозволяє створювати живі та реалістичні світи, які захоплюють глядачів та гравців;

– освіта: тривимірне моделювання використовується у навчальних закладах для навчання студентів архітектури, дизайну, інженерії та інших спеціальностей. Віртуальні моделі дозволяють краще зрозуміти принципи побудови об’єктів та їх взаємодії.

Застосування тривимірного моделювання у цих сферах відкриває безліч можливостей для покращення продуктивності, креативності та інновацій, що робить його важливим інструментом у сучасному світі.

Тривимірне моделювання, як концепція, виникло приблизно в середині XX-го століття разом з розвитком комп'ютерних технологій та комп'ютерної графіки. Найперші спроби створення тривимірних зображень відбувалися у 1960-х роках, коли вчені та інженери експериментували з різними методами візуалізації об'єктів у тривимірному просторі.

У 1963 році, наприклад, був розроблений перший комп'ютерний графічний об'єкт з тривимірною формою, яким став «Куб» на екрані комп'ютера (рис. 1.1). Ці початкові досягнення відкрили шлях до подальшого розвитку тривимірного моделювання [2].

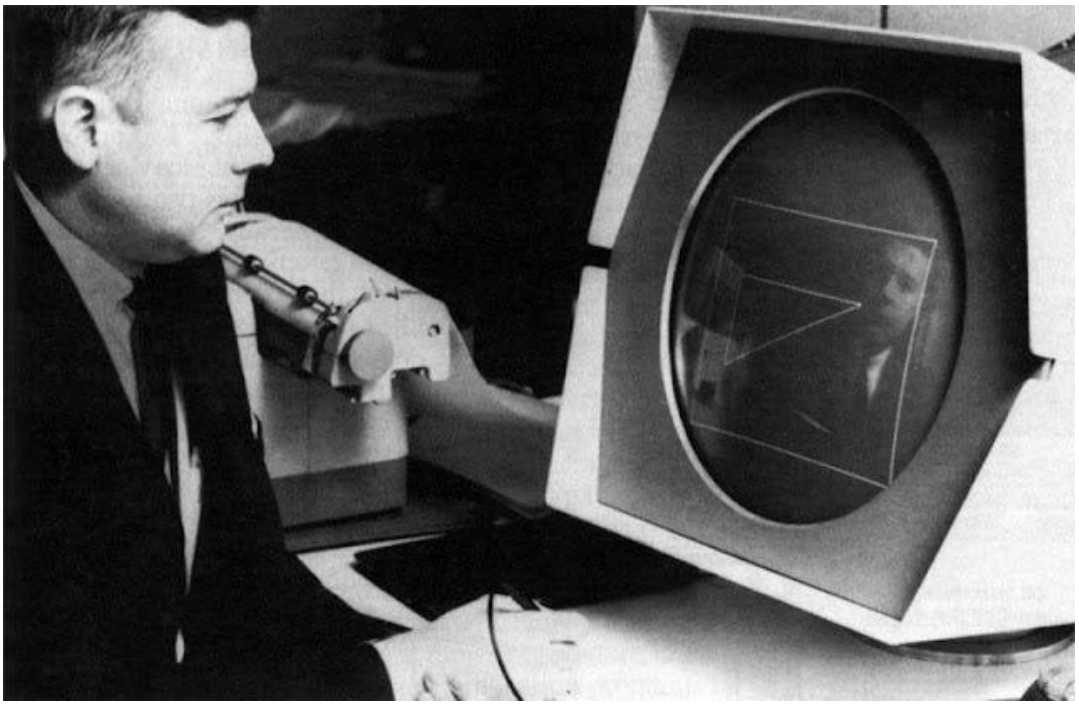

Рисунок 1.1 – Іван Сазерленд із Sketchpad на екрані

У наступні десятиліття тривимірне моделювання стало набирати популярність та широке застосування. З'явилися перші програми для

тривимірного моделювання, які дозволяли створювати складні об’єкти у тривимірному просторі та взаємодіяти з ними.

У 1990–2000 роках з’явилися потужні програмні засоби для тривимірного моделювання, які стали основою для роботи в різних галузях, таких як інженерія, архітектура, медицина, дизайн та ігрова індустрія. Сьогодні такі програми, як Autodesk 3ds Max, SolidWorks, Blender, надають користувачам широкий спектр інструментів для створення вражаючих тривимірних моделей і сцен, що знаходять широке застосування у сучасному світі [3, 4].

1.2 Blender 3D як програмний пакет для тривимірної графіки

Blender 3D – це відкрите програмне забезпечення для створення комп’ютерної графіки, анімації, відеоредакції та інших сфер комп’ютерного мистецтва. У цьому розділі ми розглянемо ключові аспекти Blender 3D, його функціональність, можливості та вплив на індустрію комп’ютерної графіки та анімації (рис. 1.2).

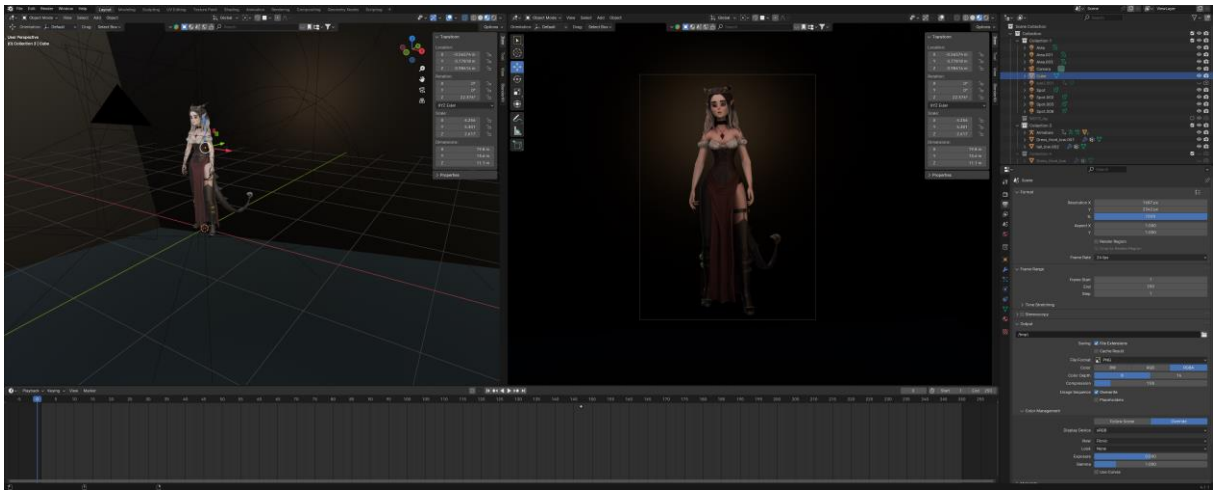

Рисунок 1.2 – Інтерфейс програми Blender

Blender почав свій шлях у 1994 році як внутрішній проєкт студії анімації NeoGeo в Нідерландах. Початково створений Тоном Розендалем, програма швидко стала популярною серед спеціалістів у сфері комп'ютерної графіки. У 2002 році Blender було випущено під вільною ліцензією GNU GPL, що дозволило розвивати його за участю глобальної спільноти розробників [5].

До особливостей Blender належать:

– моделювання: Blender пропонує різноманітні інструменти для моделювання, такі як скульптування, вершинне моделювання та моделювання за допомогою мешів. Завдяки цьому, користувачі можуть створювати складні 3D-моделі з нуля або редагувати існуючі моделі для власних потреб;

– анімація: у Blender існують різноманітні інструменти для створення анімації, включаючи ключові кадри, скелетну анімацію та анімацію шляхів. Це дозволяє аніматорам створювати рухи об'єктів та персонажів з високою ступенем контролю;

– рендерінг: Blender надає користувачам можливість вибирати між різними двигунами рендерингу, які відрізняються за швидкістю та якістю рендерингу. Наприклад, Cycles забезпечує фотореалістичні результати, в то час як Eevee надає швидкий попередній перегляд для швидкого відтворення;

– відеоредакція: Blender має інтегрований відеоінструментарій, який дозволяє монтаж відео, додавання спеціальних ефектів, корекцію кольору та багато іншого. Це робить його потужним інструментом для створення не лише 3D-анімації, а й редакції відеоматеріалів;

– спеціальні ефекти та симуляції: Blender також пропонує інструменти для створення різноманітних спеціальних ефектів, таких як симуляція частинок, рідин, вогню та диму.

Ці інструменти дозволяють створювати реалістичні та захоплюючі візуальні ефекти.

Blender змінив спосіб, яким люди підходять до створення комп'ютерної графіки та анімації. Його відкритий характер та безкоштовна ліцензія зробили

його доступним для широкого кола користувачів, що призвело до збільшення кількості творчих проєктів і розвитку спільноти користувачів [6–9].

Blender постійно розвивається, і його майбутнє виглядає яскравим. З покращенням інтерфейсу, новими функціями та підтримкою сучасних технологій, таких як віртуальна реальність та штучний інтелект, Blender залишається одним із провідних інструментів у сфері комп’ютерної графіки та анімації.

1.3 Застосування мови програмування Python у тривимірній графіці та огляд API Blender

Python використовується широко в області тривимірної графіки, особливо в контексті програмного пакету Blender. Він дозволяє користувачам автоматизувати рутинні завдання, розширювати функціональність та створювати скрипти для реалізації складних ідей.

Python у Blender надає гнучкість та зручність у розробці тривимірних проєктів. Користувачі можуть легко створювати сценарії для автоматизації рутинних завдань, таких як створення об’єктів або анімація. Він також дозволяє розробникам створювати власні додаткові інструменти та плагіни для розширення можливостей Blender.

Завдяки простому синтаксису та великій кількості ресурсів, Python є доступним для новачків у сфері тривимірної графіки. Його використання у поєднанні з Blender відкриває нові можливості для творчих індивідуумів, надаючи їм потужні інструменти для втілення своїх ідей у життя.

1.3.1 Мова програмування Python

Python – це високорівнева, інтерпретована мова програмування, яка займає велике місце в світі програмування. Відомий своєю простотою та ефективністю, Python став мовою вибору для багатьох розробників. Його зрозумілий синтаксис та широкий спектр функціональності роблять його ідеальним інструментом для початківців, що тільки вчаться програмувати, а також для досвідчених спеціалістів, які розвивають складні проєкти.

У сфері тривимірної графіки, Python став ключовим інструментом завдяки своєму застосуванню у програмному пакеті Blender. Blender – це потужне програмне забезпечення для тривимірного моделювання, анімації та візуалізації, яке знаходить застосування в різних галузях, від кіноіндустрії до ігрової розробки. Python дозволяє розробникам створювати скрипти та плагіни для автоматизації завдань, розширення функціональності та створення складних тривимірних сцен. Це робить Python незамінним інструментом для тих, хто працює у галузі тривимірної графіки.

Проте Python не обмежується лише тривимірною графікою. Він також широко використовується у веброзробці, де він служить основою для таких фреймворків, як Django та Flask. Python відомий своєю зручністю та ефективністю у розробці вебдодатків та вебсайтів будь-якої складності. Крім того, Python є популярним інструментом для аналізу даних та машинного навчання. Він має багато бібліотек, які дозволяють розробникам ефективно обробляти та аналізувати великі обсяги даних та створювати складні моделі машинного навчання.

У наукових дослідженнях Python використовується для проведення наукових обчислень та моделювання. Бібліотека SciPy, наприклад, надає широкий спектр інструментів для розв’язання наукових проблем та проведення складних обчислень. Усі ці можливості роблять Python однією з найбільш універсальних та потужних мов програмування на сьогоднішній

день. Він продовжує зберігати свою популярність і залишається незамінним інструментом для розробки різноманітних програмних продуктів у різних галузях індустрії.

1.3.2 Огляд API Blender

API Blender є ключовою складовою програмного пакету, яка надає розробникам широкі можливості для взаємодії з пакетом та створення різноманітних візуальних ефектів, анімацій та тривимірних сцен. Це набір інструментів та функцій, доступних для програмного забезпечення через зовнішні програми або скрипти, що відкриває нескінченні можливості для творчості та розширення функціональності Blender (рис. 1.3).

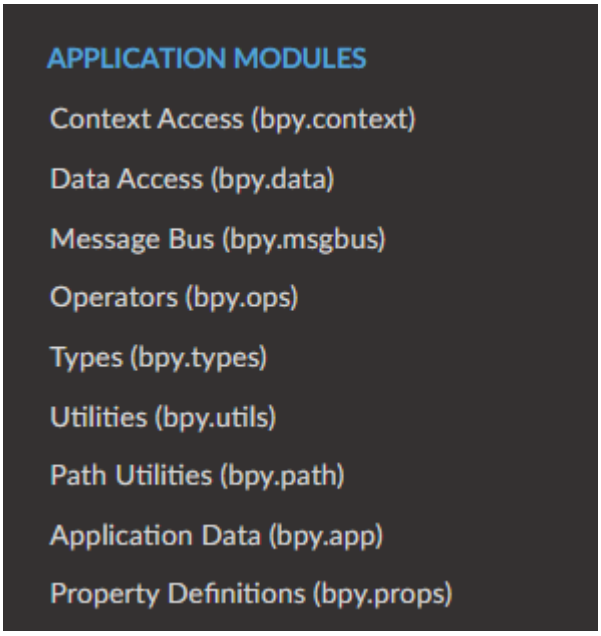

Рисунок 1.3 – Базові компоненти API Blender

API Blender надає доступ до всіх основних компонентів та функцій пакету, включаючи створення, редагування та анімацію об’єктів, керування матеріалами та текстурами, а також налаштування освітлення та камер. Ці

можливості дозволяють розробникам створювати складні тривимірні сцени, анімації та візуальні ефекти, які вражають глядачів та відкривають нові можливості для творчості.

Основною мовою програмування для взаємодії з API Blender є Python. Python є популярним вибором серед розробників завдяки своїй простоті та потужності. Він надає зручний та ефективний інтерфейс для роботи з Blender, що робить його ідеальним інструментом для створення скриптів та плагінів для розширення функціональності пакету. Python також надає розробникам можливість використовувати інші бібліотеки та інструменти для роботи з тривимірною графікою, що додає ще більше можливостей для творчості та розвитку проєктів.

API Blender можна класифікувати наступним чином:

– операції з об’єктами: це включає створення, редагування, переміщення та анімацію об’єктів у тривимірному просторі;

– робота з матеріалами та текстурами: API дозволяє налаштовувати матеріали та текстури об’єктів, задавати їх кольори, властивості та інші параметри;

– освітлення та камери: ця категорія охоплює функції для налаштування освітлення сцени, розміщення та налаштування камер для створення певної перспективи;

– анімація: API надає засоби для створення різноманітних анімаційних ефектів та управління анімаційними параметрами об’єктів;

– рендеринг та візуалізація: це включає функції для налаштування процесу рендерингу та візуалізації сцени у тривимірному просторі.

API Blender є дуже гнучким та розширюваним інструментом, який продовжує розвиватися та покращуватися з часом. Завдяки активній спільноті розробників та користувачів, нові можливості та функції постійно додаються до API, розширюючи його функціональність та відкриваючи нові можливості для творчості та інновацій у світі тривимірної графіки.

API Blender надає розробникам можливість повністю керувати всіма аспектами програми. Нижче наведено приклад коду (рис. 1.4), який ілюструє створення анімації з трьох кубів у Blender.

```
import bpy
import random

# Визначення функції для створення куба з довільними координатами та розмірами
def create_cube():
    bpy.ops.mesh.primitive_cube_add(size=1, location=(random.randint(-5, 5),
                                                      random.randint(-5, 5),
                                                      random.randint(0, 5)))

# Створення 3 кубів за допомогою функції create_cube
for i in range(3):
    create_cube()

# Встановлення ключових кадрів для анімації
bpy.context.scene.frame_start = 0
bpy.context.scene.frame_end = 50

# Визначення функції для зміни координат куба на кожному кадрі
def animate_cube(frame):
    for obj in bpy.context.scene.objects:
        if obj.name.startswith("Cube"):
            obj.location[2] += 0.1

# Запуск анімації
for i in range(50):
    animate_cube(i)
    bpy.context.scene.frame_set(i)
    bpy.ops.anim.keyframe_insert(type='BUILTIN_KSI_LocRotScale')
```

Рисунок 1.4 – Приклад коду створення анімації кубів

Цей код створює три куби з рандомними розмірами та розташуванням, а потім на кожному кадрі збільшує їх висоту на 0,1 одиницю. Як результат, отримується анімація з 50 кадрів, де куби поступово піднімаються вгору. Завдяки Blender API розробники мають можливість повністю керувати усіма параметрами Blender, що робить цей інструмент потужним для розширення функціональності програми [10–12].

1.4 Розгляд процедурної генерації та ціль її використання у створенні аддону

Процедурна генерація відіграє важливу роль у сучасній комп’ютерній графіці та дизайні. Цей підхід дозволяє створювати складний візуальний контент за допомогою алгоритмів та процедур, замість ручного моделювання кожного елементу. У цьому розділі ми розглянемо сутність процедурної генерації та визначимо мету розробки аддону для програми Blender.

Процедурна генерація – це метод створення візуального контенту за допомогою алгоритмів та математичних процедур. Замість ручного моделювання кожного елементу, процедурна генерація використовує програмні скрипти та параметри для автоматизованого створення складних об’єктів та сцен. Цей підхід широко використовується в індустрії відеоігор, анімації, візуальних ефектів у кіно та інших галузях (рис. 1.5).

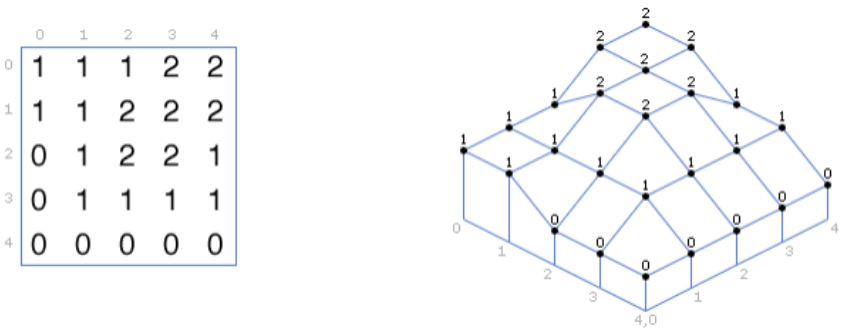

Рисунок 1.5 – Процедурна генерація по параметрам

Мета розробки аддону для програми Blender полягає в створенні інструментів, які допоможуть розширити можливості програми за допомогою процедурної генерації. Основні цілі розробки аддону включають:

– автоматизація процесу створення складних об’єктів: розробка інструментів, які дозволяють швидко та ефективно створювати складні 3D-моделі за допомогою параметрів та алгоритмів процедурної генерації;

– розширення можливостей користувача: створення нових функцій та інструментів, які дозволяють користувачам Blender працювати з процедурно згенерованими об’єктами та ефектами, що полегшує їхню роботу та збільшує творчий потенціал;

– підвищення ефективності та продуктивності: створення інструментів, які дозволяють швидко створювати та редагувати великі об’єкти та сцени, зменшуючи час та зусилля, потрібні для їхнього моделювання вручну.

Процедурна генерація відкриває нові можливості у створенні візуального контенту, а розробка аддонів для Blender дає змогу використовувати цей підхід у програмі для 3D-моделювання. Розглянувши основні аспекти процедурної генерації та визначивши мету розробки аддону, ми готові переходити до реалізації цього проєкту.

1.5 Постановка задачі

Тривимірне моделювання та його застосування у сучасному світі вимагають вдосконалення та нових підходів для покращення продуктивності та якості роботи. Основна мета даної кваліфікаційної роботи – розробка функціонального аддону для популярного 3D-пакету Blender, який дозволить розширити можливості програми у сфері тривимірного моделювання та рендерингу за допомогою мови програмування Python.

Об’єктом роботи є створення інтерактивного аддону для 3D-пакету Blender, який дозволить користувачам створювати та маніпулювати складними геометричними структурами.

Метою роботи є розробка інструменту, який спростить процес створення складних 3D-моделей у Blender за допомогою процедурної генерації та динамічних параметрів.

Для досягнення мети необхідно вирішити такі завдання:

– отримати фундаментальні навички управління 3D-пакетом Blender;

– проаналізувати наявні функції модифікаторів, які є основними параметрами процедурної генерації;

– ознайомитися з інтерфейсом програмування застосунків (API) Blender;

– розробити та втілити в життя алгоритми та масштабний спектр параметрів для процедурної генерації;

– створити каркас аддону та інтегрувати у нього вищезазначений функціонал.

2 ПОПЕРЕДНЄ МОДЕЛЮВАННЯ СИСТЕМИ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ПО ПАРАМЕТРАХ

2.1 Використання базових функцій API для створення процедур та параметричних модифікаторів

Blender API – це значущий засіб для створення різноманітних програм, які можуть працювати з Blender. Основним елементом Blender API є модулі аплікацій (application modules), що відкривають доступ до основних функцій та можливостей Blender [13].

Далі подано детальний опис кожного з модулів аплікацій та їх параметрів:

– bpy.app.handlers: цей модуль дозволяє реєструвати функції-обробники подій, які спрацьовують на певних етапах життєвого циклу Blender. У ці функції-обробники можна передати різні параметри, такі як контекст, подія та будь-які інші додаткові параметри, що були зазначені при реєстрації обробника подій;

Лістинг 2.1 Приклад використання модуля реєстрації функції-обробника:

```
import bpy

# Створення функції-обробника події
def my_handler(scene):
print("Подія сцени:", scene.name)

# Реєстрація функції-обробника на подію "завантаження сцени"
bpy.app.handlers.load_post.append(my_handler)
```

```
# Повідомлення про успішну реєстрацію
print("Функція-обробник успішно зареєстрована на подію 'завантаження сцени'.")
```

– bpy.app.version: цей модуль дозволяє отримати інформацію про версію Blender, яка використовується в поточному проєкті. Серед параметрів, які можна отримати з цього модулю, є номер версії, назва релізу та дата випуску програмного продукту;

Лістинг 2.2 Приклад використання модуля визначення версії Blender:

```
import bpy
# Отримання версії Blender
blender_version = bpy.app.version
print("Версія Blender:", blender_version)
```

– bpy.app.context: даний модуль містить інформацію про поточний контекст у Blender, включаючи видиму область та виділені об’єкти. Серед параметрів, які можна отримати з цього модулю, є інформація про активний об’єкт, активний матеріал та інші важливі дані, які стосуються поточного контексту роботи у Blender;

Лістинг 2.3 Приклад використання модулю інформації:

```
import bpy

# Отримання інформації про активний об’єкт
active_object = bpy.context.active_object
print("Активний об’єкт:", active_object.name)

# Отримання інформації про активний матеріал
active_material = active_object.active_material
```

print("Активний матеріал:", active_material.name)

 # Отримання інших параметрів
view_layer = bpy.context.view_layer.name
print("Видима область:", view_layer)

– bpy.app.data: цей інструмент дозволяє отримати доступ до різноманітних даних, що зберігаються в Blender, таких як інформація про сцени, об’єкти та матеріали. Його можна використовувати для отримання списків усіх об’єктів у сцені, усіх матеріалів та інших важливих параметрів;

Лістинг 2.4 Приклад використання модулю доступу:

import bpy

 # Отримання списку усіх сцен у проєкті
scenes = bpy.data.scenes
print("Список усіх сцен:", [scene.name for scene in scenes])

 # Отримання списку усіх об’єктів у поточній сцені
current_scene = bpy.context.scene
objects = current_scene.objects
print("Список усіх об’єктів у поточній сцені:", [obj.name for obj in objects])

 # Отримання списку усіх матеріалів у проєкті
materials = bpy.data.materials
print("Список усіх матеріалів:", [mat.name for mat in materials])

– bpy.ops: цей модуль відкриває можливість виконання різноманітних дій у Blender, таких як створення об’єктів та налаштування їх

характеристик. У функціях цього модулю можна передавати параметри для визначення контексту та параметрів виконуваної операції [14];

Лістинг 2.5 Приклад використання модулю доступу:

```
import bpy

# Створення нового куба
bpy.ops.mesh.primitive_cube_add(size=2, location=(0, 0, 0))

# Отримання доступу до створеного об’єкта
new_object = bpy.context.active_object

# Зміна назви об’єкта
new_object.name = "Мій Куб"
```

– bpy.app.ui: даний модуль у Blender дозволяє змінювати розташування елементів інтерфейсу користувача, таких як панелі та меню. Його функції дозволяють керувати розміщенням та виглядом цих елементів. Можливі параметри, які можна передати у функції цього модулю, охоплюють тип елемента інтерфейсу, такий як панель або меню, а також різні характеристики цього елемента, такі як розмір, розташування, кольорова схема та інші;

Лістинг 2.6 Приклад використання модулю змінення:

```
import bpy

# Приклад: зміна позиції та розміру панелі в Blender
def change_panel_properties():
# Отримання доступу до панелі по імені
```

```
panel = bpy.context.workspace.tools.from_space_view3d(mode='OBJECT').layout.prop_panel

# Зміна позиції панелі
panel.location = (10, 10)  # Нова позиція (x, y) панелі

# Зміна розміру панелі
panel.scale = (2, 1)  # Новий розмір (ширина, висота) панелі
```

– bpy.app.debug: цей інструмент у Blender дозволяє користувачам встановлювати різні параметри для налагодження, такі як рівень журналу та поведінка під час налагодження. Можна настроїти, яка інформація буде записуватися у журналі відлагодження та як програма буде вести себе під час виконання коду під час відлагодження.

Лістинг 2.7 Приклад використання модулю налаштування:

```
import bpy

# Приклад: налаштування рівня журналу відлагодження
def set_debug_level():
    bpy.app.debug_value = 2  # Встановлення рівня журналу відлагодження на високий рівень
```

Модуль bpy.utilities в Blender API призначений для надання корисних утиліт, які спрощують розробку скриптів та розширень для Blender. Основна його мета – забезпечити розробникам зручні та ефективні інструменти для роботи зі списками об’єктів, обробки файлів та роботи зі строками. Далі

представлена таблиця (табл. 2.1) із переліком функцій та класів зазначеного модуля та їх параметрами.

Таблиця 2.1 – Список функцій та класів модуля bpy.utilities

Функція/Клас	**Параметри**	**Повертає**	**Опис**
1	**2**	**3**	**4**
bpy.utils.register_module()	__name__: str	1 flipped name	Реєструє модуль для подальшого використання у інших скриптах.
bpy.utils.unregister_module()	__name__: str	1 un-escaped string	Видаляє реєстрацію модуля.
bpy.utils.previews	filepath: str, preview_id: str	1 resource path	Завантажує та зберігає попередній огляд об’єкта.
bpy.utils.scene_state_compare()	scene1: bpy.types.Scene, scene2: bpy.types.Scene	1 string	Порівнює декілька сцен та повертає їх різницю.

Продовження таблиці 2.1

1	2	3	4
bpy.utils.benchmark()	func: Callable, *args: Any, **kwargs: Any	1 list	Заміряє час виконання функцій.
bpy.utils.string_paths()	string: str	1 string	Перетворює рядок зі шляхами на список.
bpy.utils.blend_paths()	string: str	1 string	Перетворює рядок зі списком шляхів на список.

Ця таблиця містить детальну інформацію про функції та класи, що доступні в модулі bpy.utilities у Blender API. Для кожного елемента вказано параметри, які вони приймають, і наданий короткий опис їхнього призначення. Використання наданої інформації допоможе розробникам краще зрозуміти призначення цих утиліт та ефективність їх використання для написання програм [15–17].

2.2 Параметричні модифікатори як основа параметрів

Модифікатори у Blender – це інструменти, які дозволяють змінювати властивості об’єктів без безпосередньої модифікації їх базових полігонів або вершин. Застосовуючи модифікатори, можна змінювати форму, розмір, текстуру, розподіл вершин та інші характеристики об’єктів шляхом додавання

спеціальних ефектів або операцій. Вони дозволяють швидко створювати складні ефекти та трансформації без необхідності.

Створений аддон для процедурної генерації використовує модифікатори. Нижче буде наведено опис використовуємих модифікаторів.

Normal Edit Modifier – це інструмент, який дозволяє змінювати нормалі (вектори, що вказують напрямок поверхні) об’єктів. Він використовується для коригування або редагування нормалей, що може впливати на освітлення та візуальний вигляд об’єктів у сцені (рис. 2.1).

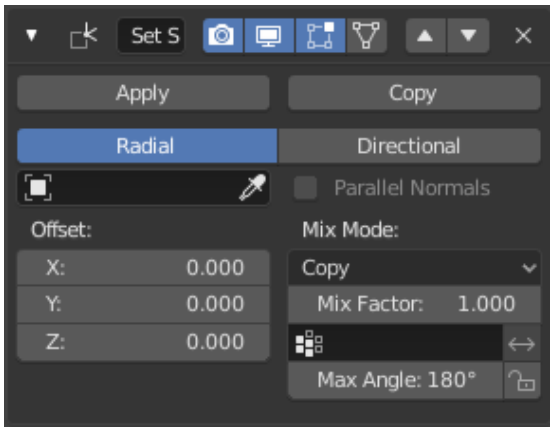

Рисунок 2.1 – Налаштування модифікатора Normal Edit

Даний модифікатор можна використовувати для швидкого створення радіальних нормалей для низькополігональних дерев або для виправлення ефекту мультико-подібного рендерингу, подібного до того, що показано на рисунку 2.2.

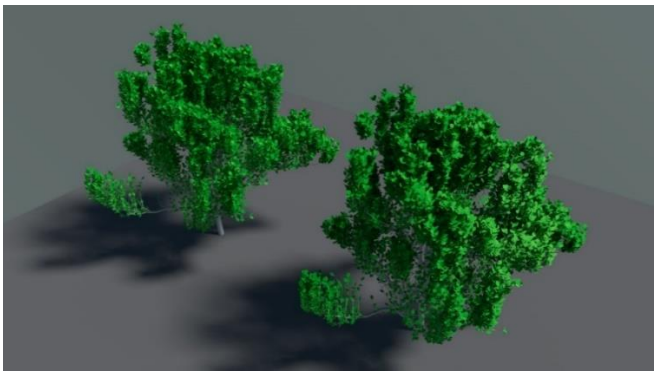

Рисунок 2.2 – Приклад модифікації нормалей

Subdivision Surface Modifier (підрозділення поверхні) в Blender – це інструмент, що застосовується для підвищення роздільної здатності (кількості полігонів) об’єктів, що дозволяє створювати більш плавні та деталізовані форми. Це, в свою чергу, дозволяє покращити вигляд моделі та зробити її більш реалістичною при рендерингу [18].

Формула для модифікатора Subdivision Surface в основному описує, як кожен полігон ділиться на більшу кількість дрібних полігонів:

$$V_{new} = (1 - \alpha) * V_{orig} + \alpha * V_{smooth}, \quad (2.1)$$

де *Vnew* – нове положення вершини після підрозділу;

Vorig – початкове положення вершини;

Vsmooth – плавне положення вершини;

α – коефіцієнт плавності, який визначає, наскільки далеко нова вершина розташовується між початковою та плавною позиціями.

Ця формула використовується для кожної вершини об’єкта під час використання модифікатора Subdivision Surface для додаткового розділення геометрії [19].

Нижче надано приклад застосування модифікатора Subdivision Surface (рис. 2.3).

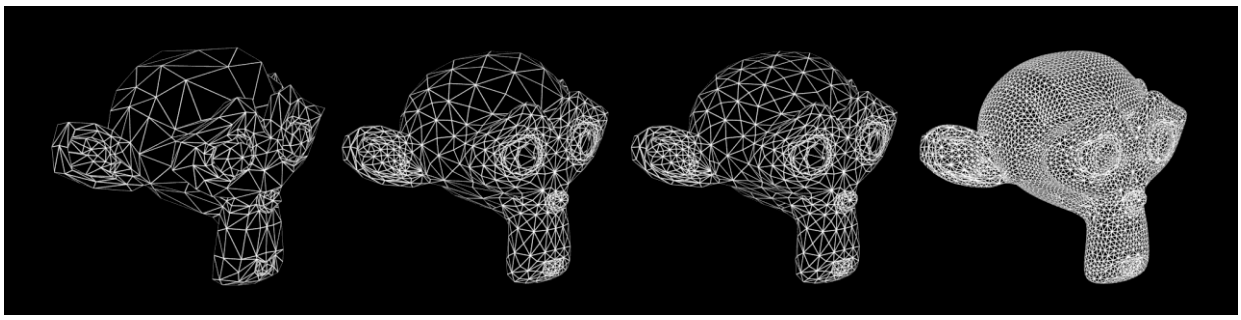

Рисунок 2.3 – Приклад роботи модифікатора Subdivision Surface

Bevel Modifier у Blender використовується для створення скошених країв або вигинів на об’єктах. Він заокруглює кути і перетини, що дозволяє створювати більш реалістичні та естетичні моделі (рис. 2.4).

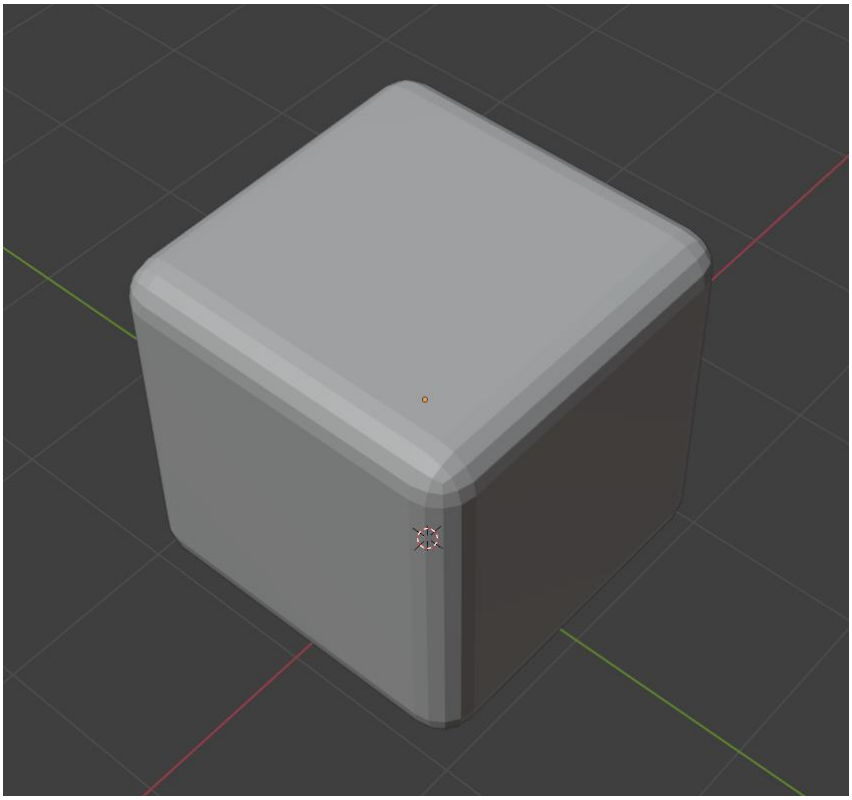

Рисунок 2.4 – Приклад роботи модифікатора Bevel на кубі

Формула для модифікатора Bevel може виглядати наступним чином:

$$V_{out} = V_{in} + \frac{N+S}{d}, \tag{2.2}$$

де *Vout* – нова позиція вершини після застосування Bevel;

Vin – початкова позиція вершини;

N – вектор нормалі до поверхні вершини;

S – ширина Bevel (величина зміщення);

d – відстань від початкової вершини до краю Bevel.

Boolean Modifier у Blender використовується для виконання операцій булевої логіки над об’єктами. Він дозволяє об’єднувати, віднімати або перетинати об’єкти, що дає можливість створювати більш складні форми. Наприклад, з його допомогою можна об’єднати два об’єктів, щоб створити єдину форму, або видалити об’єкт з іншого об’єкта, створюючи отвір або поглиблення.

Цей модифікатор базується на алгоритмах булевої алгебри і використовується для створення складних геометричних форм у Blender без необхідності вручну редагувати об’єкти. Нижче наведено приклад застосування операції вичітання циліндра з куба (рис. 2.5).

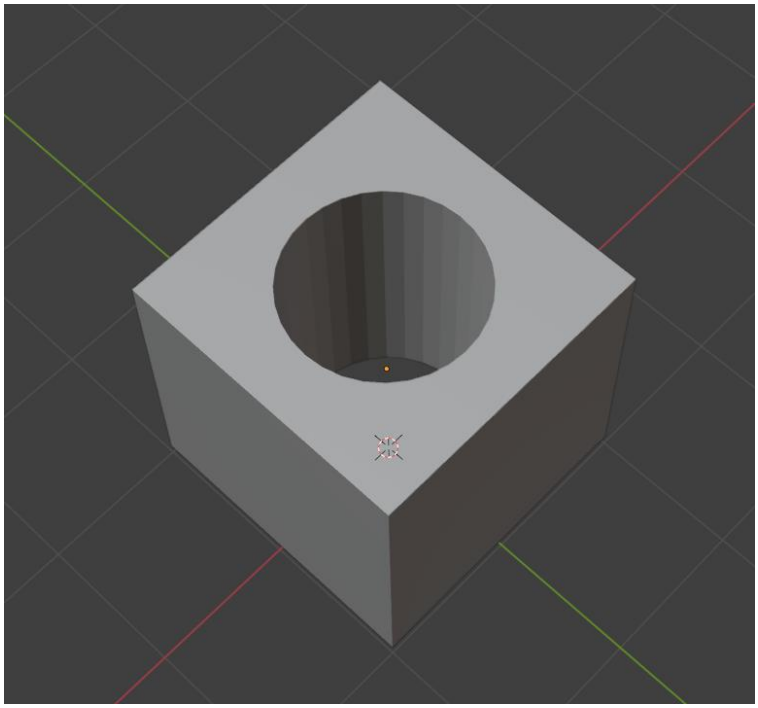

Рисунок 2.5 – Приклад роботи модифікатора Boolean

Операції булевої логіки в модифікаторі boolean відповідають наступній системі формул:

– об’єднання (union): результуючий об’єкт містить області, які належать хоча б одному з вихідних об’єктів. Це відповідає логічному оператору «OR»;

– віднімання (difference): результуючий об’єкт містить тільки ті області, які належать першому вихідному об’єкту, але не належать другому. Це відповідає логічному оператору «AND NOT»;

– перетин (intersect): результуючий об’єкт містить тільки ті області, які належать обом вихідним об’єктам одночасно. Це відповідає логічному оператору «AND».

Отже, модифікатор Boolean у Blender є потужним інструментом для виконання операцій булевої логіки над об’єктами. Його використання дозволяє розширити можливості моделювання та створити більш складні форми, об’єднуючи, віднімаючи або перетинаючи об’єкти. Модифікатор базується на алгоритмах булевої алгебри, що дозволяє виконувати операції об’єднання, віднімання та перетину з використанням відповідних логічних операторів. Цей інструмент забезпечує ефективний та швидкий спосіб створення складних геометричних форм без необхідності вручну редагувати об’єкти, що робить його невід’ємною частиною процесу моделювання у Blender.

2.3 Параметри процедурної генерації

Аддон для розширення можливостей Blender використовує вказані функції, процедури та модифікатори для створення об’єктів процедурної генерації на основі заданих параметрів. Давайте розглянемо, як саме ці параметри впливають на створені процедури та функції, які були описані раніше [20]. Аддон спрямований на тему наукової фантастики та космічних кораблів, тому в ньому можна виділити три основні компоненти:

– mesh scramble – редактор об’єктів по параметрам;

– generate spaceship base – меню, що процедурно генерує космічний корабель випадкової форми;

– generate sci-fi material – меню, що процедурно генерує текстуру для раніше створеного космічного корабля.

Детально розглянемо параметри генерування об’єкта на прикладі компонента «Mesh Scramble» (рис. 2.6).

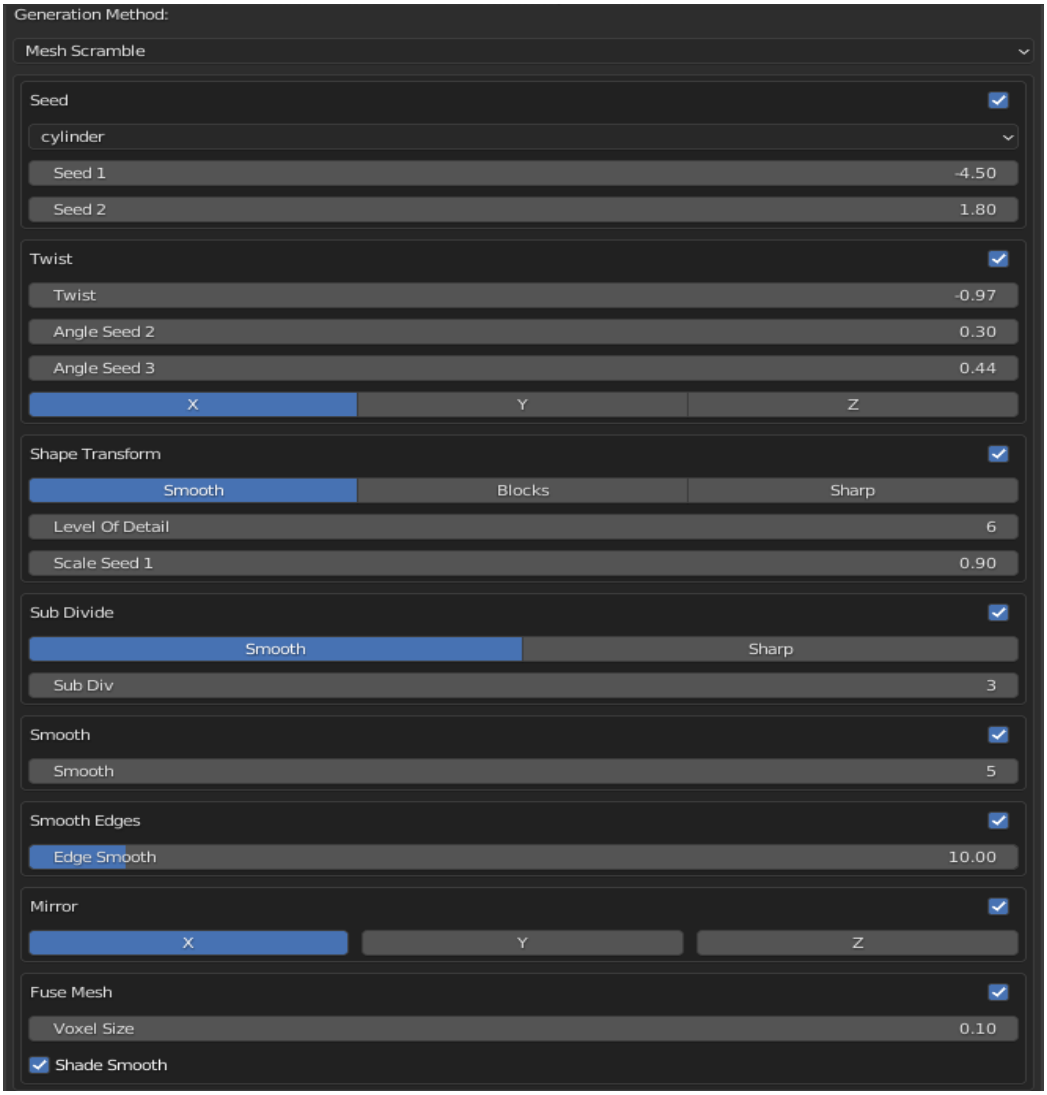

Рисунок 2.6 – Параметри генерування Mesh Scramble

Меню має наступни параметри:

– twist – дає можливість задавати значення вигину для кінцевого об’єкта та встановлювати кут вигину;

– shape Transform – спершу встановлюється режим трансформації форми об’єкта, а далі встановлюється значення цієї трансформації;

– sub divide – задає форму об’єкту за допомогою двох режимів: із заокругленими або гострими кутами;

– smooth – встановлює рівень гладкості об’єкта, якщо раніше був активований режим «Smooth»;

– smoth edges – налаштовує рівень гладкості ребер у об’єктів;

– mirror – визначає вісь, по якій буде відзеркалено об’єкт;

– fuse mesh – ця опція дозволяє визначити, наскільки близько полігони будуть приєднані один до одного.

Далі буде розглянуто пункт меню, який відповідає за процедурну генерацію базової форми космічного корабля під назвою «Generate Spaceship Base» (рис. 2.7).

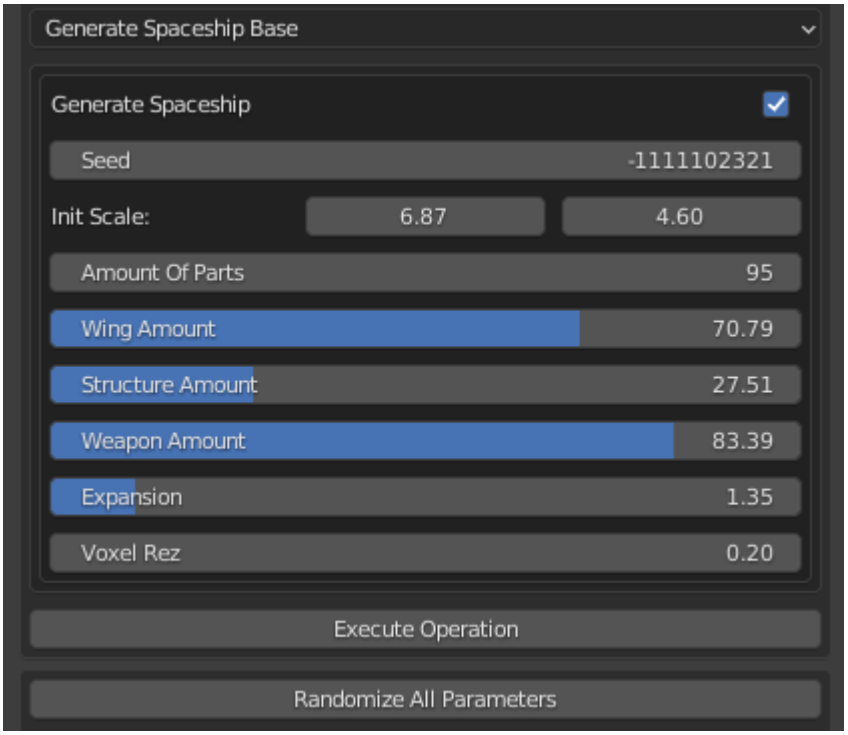

Рисунок 2.7 – Параметри для створення бази космічного корабля

Меню має наступни параметри:

– amount of parts – кількість унікальних об’єктів, які будуть з’єднані між собою;

– wing amount – показник відсоткового співвідношення кількості крил до загальної кількості частин космічного корабля;

– structure amount – кількість різних та унікальних об’єктів, які використовуються для створення однієї моделі;

– weapon amount – визначає, скільки об’єктів у формі космічної зброї буде використано в моделі;

– expansion – значення вказує, які розміри буде мати об’єкт, який буде використовуватися як загальний розмір у моделі;

– voxel rez – параметр встановлює, наскільки детально кожний полігон в об’єкті буде представлений у візуалізації;

– execute operation – кнопка використовується для застосування всіх раніше встановлених параметрів до об’єкта;

– randomize all parameters – кнопка призначена для встановлення випадкових значень для раніше вказаних параметрів, дозволяючи створити різноманітні та випадкові об’єкти або сцени [21, 22].

Наступним кроком буде огляд меню для автоматичного створення матеріалів або текстур для корабля, який вже було створено (рис. 2.8).

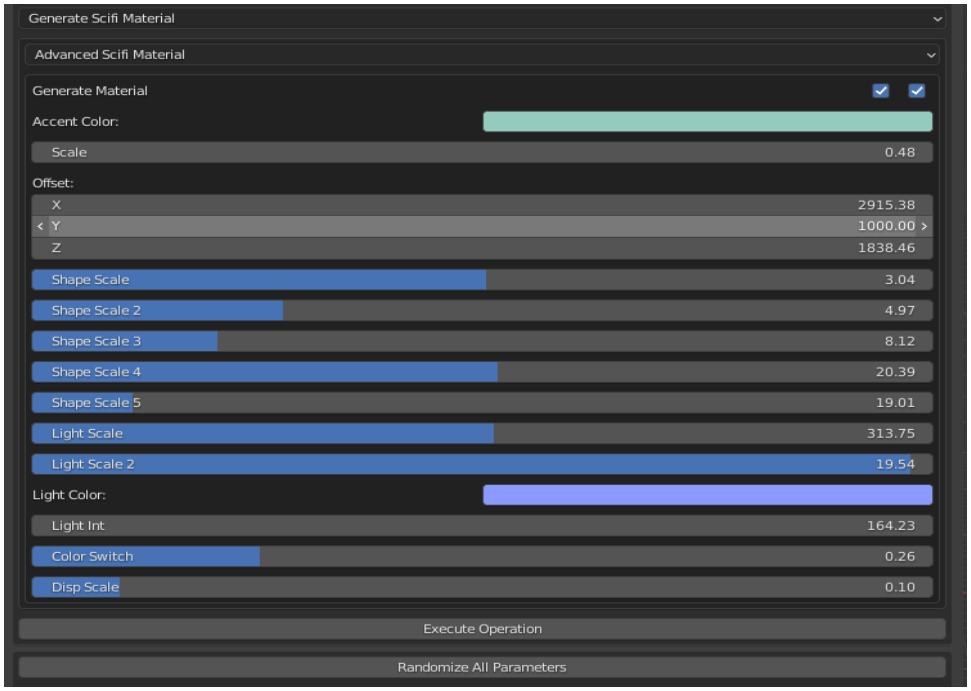

Рисунок 2.8 – Параметри для створення матеріалу космічного корабля

Меню має наступни параметри:

– accent color – параметр визначає колір, який буде використовуватися для частинок матеріалу;

– offset – параметр визначає величину зміщення текстури по всім трьом осям (*X*, *Y*, *Z*);

– shape scale – параметр визначає розмір та випуклість текстур, які застосовуються до об’єкта;

– shape scale 2, 3, … – параметр визначає розмір та випуклість кожного типу текстур, які застосовуються до об’єкта;

– light scale – параметр визначає розмір часточок з підкреслюючим кольором, які виступають в ролі джерел світла;

– color switch – парметр регулює співвідношення між насиченістю підкреслюючого кольору та кольором часточок, які є джерелами світла;

– disp scale – парметр задає розмір часточок, що мають підкреслюючий колір та виступають джерелами світла.

3 РЕАЛІЗАЦІЯ АДДОНУ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ, ОГЛЯД ПРОГРАМНОГО СЕРЕДОВИЩА ТА ПРОЦЕС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Аналіз робочого середовища для створення програмного продукту

В ході кваліфікаційної роботи створено Python-застосунок для 3D-пакету Blender, який використовує процедурну генерацію матеріалів, текстур і об’єктів. Для цього використовувалися інструменти розробника в Blender, які розширюють можливості цієї програми і допомагають розробникам працювати більш ефективно. Для активації усіх функцій розробника в Blender необхідно перейти до основних налаштувань інтерфейсу програми та увімкнути два пункти, як показано на рисунку 3.1.

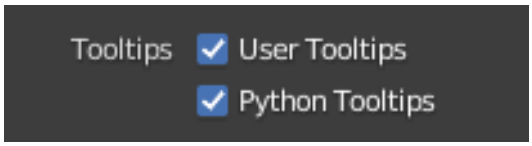

Рисунок 3.1 – Увімкнення функцій розробника

У Blender, поза основним набором інструментів, також існує застосунок, який можна вважати «секретною зброєю» для більш глибоких можливостей. Це вбудований текстовий редактор, доступний через вкладку «Text Editor» (текстовий редактор) у верхньому меню Blender.

Використовуючи цей редактор, користувачі можуть створювати скрипти на мові програмування Python, що відкриває безмежні можливості для автоматизації різних операцій та створення складних анімацій та 3D-моделей. Такі скрипти можна виконувати прямо в середовищі Blender або використовувати як розширення для додаткової функціональності програми.

Користування текстовим редактором у Blender має свої значущі переваги:

– автоматизація процесів: створення скриптів дозволяє автоматизувати багато рутинних завдань у Blender, таких як створення об’єктів, налаштування параметрів, анімація тощо;

– гнучкість та контроль: користувач може мати більший контроль над процесами та параметрами, ніж це можливо за допомогою стандартного інтерфейсу. Скрипти дають можливість створювати складніше та точніше визначені ефекти та дії;

– розширення можливостей: текстовий редактор розширює можливості Blender, оскільки дозволяє використовувати сторонні бібліотеки Python та створювати розширення та плагіни для розширення функціональності програми [23–26];

– навчання та спільнота: користувачі можуть навчитися програмувати на Python та розвивати свої навички через розробку скриптів для Blender. Також існує широка спільнота користувачів, яка готова допомогти у вирішенні проблем та надати поради;

– індивідуальні налаштування: за допомогою скриптів можна створювати індивідуальні інструменти та функції, які відповідають конкретним потребам користувача;

– швидкість розробки: використання скриптів дозволяє швидко реалізовувати та тестувати нові ідеї та концепції без необхідності повторного створення деяких процесів вручну.

Тож, текстовий редактор у Blender дозволяє більш гнучко працювати зі скриптами на мові програмування Python. Він дозволяє автоматизувати багато рутинних завдань та створювати нові інструменти, що розширює функціональність програми. Редагування скриптів у текстовому редакторі може бути швидшим та зручнішим способом роботи, особливо для досвідчених користувачів. Використання текстового редактора також допомагає вчитися програмуванню на Python та розвивати навички для роботи з іншими програмами і інструментами. Він також забезпечує більший

контроль над кодом та дозволяє легко редагувати та вдосконалювати скрипти для досягнення бажаних результатів. Також їм можна користуватися у окремому вікні, що дозволяє відразу бачити результат, як на рисунку 3.2.

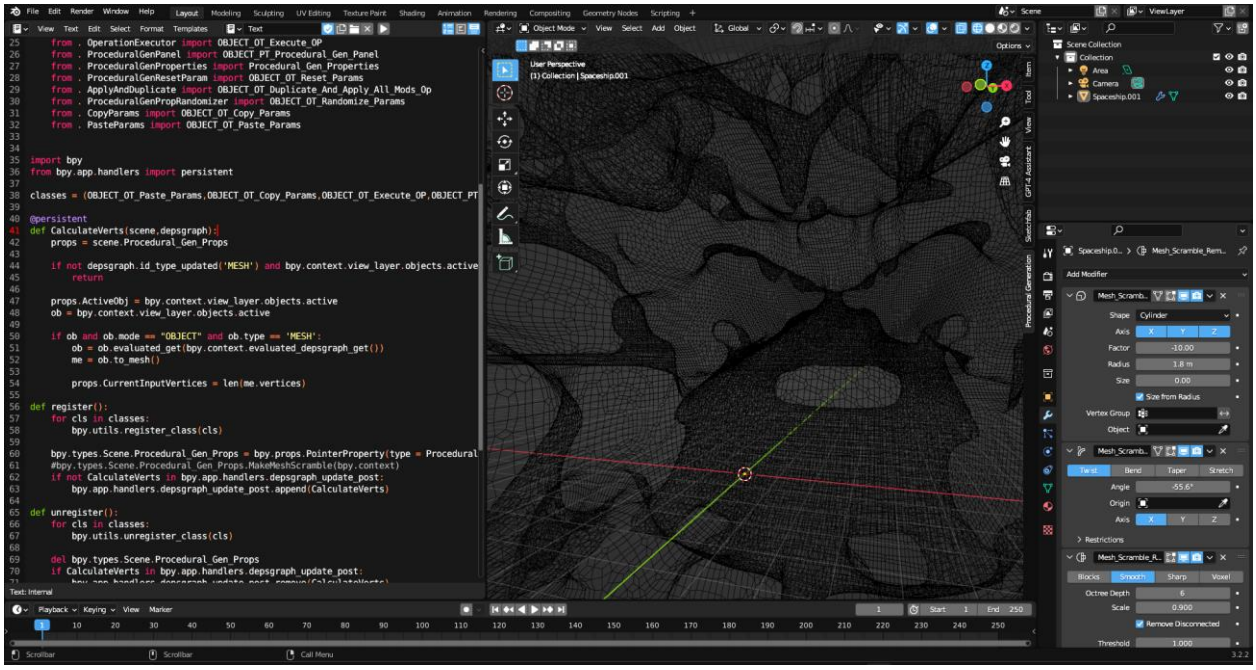

Рисунок 3.2 – Текстовий редактор та сцена

Отже, текстовий редактор у Blender є потужним інструментом для створення, редагування та виконання скриптів на мові Python. Він забезпечує зручну робочу область для програмістів та розробників, де можна легко створювати складні програми та автоматизувати рутинні завдання.

3.2 Розробка програмного забезпечення

Для коректної роботи доповнення було розроблено тринадцять файлів мовою Python. У кожного файлу своє призначення: деякі відповідають за деякий функціонал програми, деякі за методи або функції або за графічну складову оболонки доповнення. Нижче наведено детальний опис кожного файлу:

– init.py – головний файл плагіна виконує важливі завдання, зокрема, він відповідає за ініціалізацію та реєстрацію функцій плагіна при завантаженні у Blender. Цей файл має ключове значення, оскільки він починає виконання плагіна та забезпечує правильне функціонування його функціоналу;

– action.py – у цьому файлі знаходиться клас «Action», який управляє різними діями під час процесу процедурної генерації. Наприклад, він може створювати нові об’єкти, змінювати наявні або додавати нові текстури до об’єктів. Цей клас відповідає за виконання конкретних завдань у межах створення моделей або редагування їх параметрів в Blender;

– addModifier.py – у даному файлі знаходиться функція «AddModifier», що призначена для додавання нового модифікатора до об’єкта. Модифікатори – це інструменти, які змінюють форму, структуру або зовнішній вигляд об’єкта без фізичного редагування його полігонів або вершин. Використання цієї функції дає можливість вносити різні модифікації до об’єкта, щоб досягти бажаного результату в процесі моделювання;

– applyAndDuplicate.py – цей файл включає функцію «ApplyAndDuplicate», яка виконує два дії одночасно: застосовує зміни до об’єкту та створює нову копію з врахуванням цих змінених параметрів. Це корисна функція при моделюванні, коли потрібно зберегти оригінальний об’єкт і мати окрему копію зі зміненими налаштуваннями або модифікаціями;

– copyParams.py – цей файл включає в себе функцію «CopyParams», яка призначена для копіювання параметрів об’єкта з одного екземпляра на інший. Це стає вельми зручним, оскільки дозволяє швидко та ефективно передавати налаштування та характеристики об’єктів між різними елементами сцени або навіть між різними проєктами без необхідності вручну налаштовувати кожен елемент окремо;

– operation.py – у цей файл включено клас «Operation», який відповідає за всі види операцій, які можна виконати під час процесу

процедурної генерації. Тут реалізовані функції, такі як об’єднання об’єктів, створення нових матеріалів або зміна текстур. Цей клас надає можливість виконувати різні дії з об’єктами та їх характеристиками у межах плагіну, спрощуючи роботу з редагуванням та створенням елементів у Blender;

– operationExecutor.py – у даному файлі знаходиться клас «OperationExecutor», який відповідає за виконання операцій в контексті процедурної генерації. Цей клас дозволяє запускати операції та передавати їм необхідні параметри для виконання конкретних завдань. Він відіграє ключову роль у взаємодії з іншими частинами плагіну, забезпечуючи правильне виконання операцій з обробки об’єктів і їх атрибутів у середовищі Blender;

– param.py – у даному файлі знаходиться клас «Param», який відповідає за параметри об’єкта в контексті процедурної генерації. Цей клас забезпечує можливість встановлення та зміни різноманітних характеристик об’єкта, таких як розмір, текстура чи матеріал. Його функціонал дозволяє гнучко налаштовувати властивості об’єктів, що дозволяє створювати різноманітні та унікальні моделі в середовищі Blender;

– pasteParams.py – у даному файлі знаходиться функція «PasteParams», що відповідає за вставлення скопійованих параметрів об’єкта на інший об’єкт. Ця функція забезпечує зручний спосіб копіювання та вставлення різноманітних налаштувань об’єктів у середовищі Blender, що дозволяє ефективно керувати параметрами та створювати різноманітні моделі;

– proceduralGenPanel.py – цей файл включає клас «ProceduralGenPanel», який відповідає за відображення панелі налаштувань плагіну. Це вікно дозволяє користувачеві маніпулювати параметрами процедурної генерації, щоб змінювати поведінку та вигляд створюваних об’єктів. Користувач може змінювати різні налаштування, щоб отримати бажаний результат в роботі плагіну;

– proceduralGenProperties.py – цей файл включає клас «ProceduralGenProperties», який дозволяє користувачам налаштовувати

параметри для процедурної генерації. У цьому класі зберігаються всі налаштування, які можуть впливати на процес створення об’єктів у плагіні;

– proceduralGenPropRandomizer.py – цей файл включає клас «ProceduralGenPropRandomizer», який відповідає за можливість випадкового змінювання параметрів процедурної генерації. За допомогою цього класу можна створювати випадкові об’єкти з різними характеристиками та параметрами;

– proceduralGenResetParam.py – цей файл містить функцію «ProceduralGenResetParam», яка дозволяє повернутися до початкових значень параметрів процедурної генерації. Це дійсно зручна функція для відновлення вихідних налаштувань у разі необхідності [27].

Файл «init.py» є головним для плагіну, він завантажує та реєструє основний клас плагіну в Blender для доступу до панелі інструментів «Procedural Generation».

У «ProceduralGenPanel.py» міститься клас для створення панелі інструментів з усіма налаштуваннями для процедурного генерування, такими як мітка панелі та її розташування.

Файл «ProceduralGenProperties.py» містить класи для зберігання параметрів процедурного генерування, включаючи тип операції, налаштування параметрів операції, інформацію про вершини вхідного об’єкту та максимальну кількість вершин.

«Operation.py» має клас для кожної окремої операції в процедурному генеруванні, кожна з яких має свої власні параметри, що визначаються через «Param.py».

«OperationExecutor.py» використовується для виконання операцій, викликаючи відповідні функції для генерації 3D-моделей із заданими параметрами [28].

У файлі «ProceduralGenProperties.py» також визначаються класи для змінних параметрів, що можуть бути змінені через інтерфейс користувача, а також визначення параметрів за замовчуванням для цих властивостей.

«ProceduralGenPropRandomizer.py» відповідає за генерацію випадкових значень для властивостей після натискання кнопки «Randomize».

Так, «ProceduralGenPanel.py» дійсно виявляється ключовим файлом у контексті процедурного генерування. Його важливість полягає у тому, що він визначає, як саме користувач може взаємодіяти з плагіном через інтерфейс. Цей файл визначає, які елементи керування будуть доступні на панелі, як вони будуть відображені та як вони будуть взаємодіяти між собою. Завдяки «ProceduralGenPanel.py» користувач може змінювати параметри генерування та виконувати різноманітні дії, що робить його одним з ключових елементів плагіну.

Правильно розроблений аддон для процедурного генерування 3D об’єктів буде включати функції та параметри, які визначаються написаним кодом. Наприклад, це може включати створення нових форм, застосування випадкових трансформацій до існуючих об’єктів, генерацію текстур та інші можливості. Крім того, інтерфейс користувача залежатиме від властивостей, доданих у «ProceduralGenProperties.py», та від розташування елементів у «ProceduralGenPanel.py».

Загалом, кожен файл у цьому плагіні має свою важливу роль у функціоналі. Вони спільно створюють зручний та ефективний інструмент для генерації складних 3D об’єктів, а їхні можливості визначаються тим, які функції та параметри були включені в коді, а також які властивості були додані в інтерфейс користувача.

Файл «init.py» у плагіні має ключове значення, оскільки він містить скрипт для реєстрації та відміни реєстрації класів, які використовуються у плагіні для Blender. На початку цього файлу відбувається перевірка імпортованих класів, і якщо вони вже були імпортовані, то вони перезавантажуються. Якщо ж класи ще не були імпортовані, то вони імпортуються з інших файлів плагіну. Потім імпортується модуль bpy для доступу до функціональності Blender через Python.

У файлі знаходиться функція «CalculateVerts», яка збирає інформацію про кількість вершин у меші активного об’єкту відповідно до параметрів, встановлених користувачем у панелі «ProceduralGenPanel». Якщо кількість вершин змінюється, функція оновлює відповідні значення властивостей «Procedural_Gen_Props».

Функція «register» виконує реєстрацію набору класів, визначених у інших файлах плагіну, та створює нову властивість «Procedural_Gen_Props» для об’єкту сцени. Крім цього, вона додає функцію «CalculateVerts» до списку функцій, які автоматично викликаються під час оновлення завантаженого проєкту в Blender.

Функція «unregister» відміняє реєстрацію класів та видаляє властивість «Procedural_Gen_Props» для об’єкту сцени. Також вона видаляє функцію «CalculateVerts» зі списку функцій, які автоматично викликаються під час оновлення завантаженого проєкту в Blender.

На рисунку 3.3 представлений код файлу «Action.py», який містить клас «Action».

```
class Action:
    def __init__(self,UIName,enabeled, liveUpdate = False, liveUpdateParam="", dropDownTag = None):
        self.params = []
        self.actions = []
        self.UIName = UIName
        self.enabeled = enabeled
        self.liveUpdate = liveUpdate
        self.liveUpdateParam = liveUpdateParam
        self.dropDownTag = dropDownTag


    def AddParam(self, param):
        self.params.append(param)

    def GetParamByName(self, name):
        return next(filter(lambda x :  x[0] == name, self.params)[1],None)

    def AddAction(self,action):
        self.actions.append(action)

    def Execute(self,context, liveUpdate = False):
        for action in self.actions:
            action(self,context,liveUpdate)
        return True
```

Рисунок 3.3 – Код файлу «Action.py»

Цей клас відповідає за виконання дій в рамках процедурного генерування. Наприклад, він може виконувати дії, такі як створення нового об’єкту, модифікація наявного об’єкту або додавання нової текстури.

У цьому файлі можна очікувати реалізацію методів, які дозволяють здійснювати різноманітні дії з об’єктами та текстурами. Наприклад:

– метод для створення нового об’єкту з заданими параметрами;

– метод для модифікації властивостей наявного об’єкту, таких як розмір, форма, колір тощо;

– метод для додавання нової текстури до об’єкту з вказаними характеристиками (наприклад, тип текстури, масштаб, поворот);

– метод «GetParamByName» в класі «Action» використовується для повернення параметра зі списку «params», вказавши його назву. Цей метод дозволяє отримати доступ до конкретного параметра за його іменем, що спрощує роботу з параметрами об’єктів у процесі процедурного генерування;

– метод «AddAction» в класі «Action» використовується для додавання нової дії до списку «actions». Цей метод дозволяє динамічно розширювати список дій, що виконуються в рамках процедурного генерування, і додає нову дію до цього списку для подальшого виконання;

– метод «Execute» виконує всі дії зі списку «actions», які передані у параметрі. Якщо виконання дій завершується успішно, метод повертає значення «True». Параметр «context» передається усім діям, що додаються до списку «actions», і містить інформацію про контекст роботи у Blender. Наприклад, це може бути інформація про об’єкти на сцені, налаштування плагіну та інше. Параметр «liveUpdate» вказує, чи необхідне пряме оновлення під час виконання дій. Це означає, що при встановленні значення «True», зміни, які відбуваються під час виконання дій, будуть негайно відображені на екрані користувача [29].

Розглянемо код файлу «AddModifier.by» (рис. 3.4).

```
class AddModifier(Action):
    def __init__(self, modName, modType, UIName,enabeled, liveUpdate = False, liveUpdateParam=""):
        self.modName = modName
        self.modType = modType
        self.UIName = UIName
        self.params = []
        self.enabeled = enabeled
        self.liveUpdate = liveUpdate
        self.liveUpdateParam = liveUpdateParam

    def Execute(self,context):

        scene = context.scene
        props = scene.Procedural_Gen_Props

        obj = bpy.context.active_object
        mod = obj.modifiers.get(self.modName) if obj.modifiers.get(self.modName) != None else obj.modifiers.new(name=self.modName, type=self.modType
        mod.show_viewport = getattr(props,self.enabeled)
        mod.show_render = getattr(props,self.enabeled)
        mod.show_in_editmode = False

        for param in self.params:
            if not param.useArr:
                setattr(mod,param.paramName,getattr(props,param.VarName))
            else:
                arr = getattr(mod,param.paramName)
                arr[param.arrIndex] = getattr(props,param.VarName)
                setattr(mod,param.paramName,arr)
        return True
```

Рисунок 3.4 – Код файлу «AddModifier»

Це клас «AddModifier», який виконує роль підкласу «Action» у програмному коді. При створенні екземпляру цього класу, передаються деякі важливі параметри, такі як назва модифікатора, його тип, назва для відображення у користувацькому інтерфейсі, параметр для керування включенням/виключенням модифікатора, логічне значення для оновлення модифікатора в режимі реального часу, та параметр, який викликає оновлення модифікатора.

У класі «AddModifier» є метод «Execute», який використовується для додавання модифікатора до об’єкта у поточному контексті. Це включає в себе кілька кроків:

– отримання активного об’єкта;

– створення модифікатора з вказаними параметрами (якщо він ще не існує);

– встановлення значень параметрів модифікатора відповідно до переданих параметрів;

– оновлення модифікатора відповідно до змінених параметрів.

Метод «Execute» повертає «True», якщо операція виконана успішно.

Тепер перейдемо до розгляду коду файлу «CopyParams» та того, які функції він використовує.

Лістинг 3.1 Частина коду файлу «CopyParams»:

```
class OBJECT_OT_Copy_Params(bpy.types.Operator):
bl_idname = "object.procedural_gen_copy_parameters"
bl_label = "Copy Seed"
bl_description = "Copy The Seed Values To Share"
def execute(self,context):
scene = context.scene
props = scene.Procedural_Gen_Props
data = DataSingleton()
outs = ""
for actions in data.ActiveOperation.actions:
outs = outs + actions.enabeled + "*" + str(getattr(props,actions.enabeled))
+ "\n"
for p in actions.params:
if not p.Share:
continue
PropProps =
bpy.context.scene.Procedural_Gen_Props.bl_rna.properties[p.VarName]
if hasattr(PropProps, 'array_length') and PropProps.array_length>1:
VecotrProp = getattr(props,p.VarName)
TmpOut = "("
for i in range(PropProps.array_length):
TmpOut = TmpOut + str(VecotrProp[i])+","
outs = outs + p.VarName+ "*" + TmpOut[0:-1] + ")"+ "\n"
else:
t = base64.b64encode(zlib.compress(outs.encode()))
```

bpy.context.window_manager.clipboard = t
return {‘FINISHED’}

Оператор «OBJECT_OT_Copy_Params» є частиною системи управління об’єктами в Blender. Його основне призначення полягає у копіюванні параметрів об’єктів в межах сцени.

Коли користувач викликає цей оператор, він спрацьовує і виконує наступні дії:

– отримання об’єкту та параметрів: оператор спочатку отримує посилання на об’єкт в поточній сцені та параметри, які потрібно скопіювати. Це може включати різні властивості об’єкта, такі як позиція, обертання, масштаб, матеріали, модифікатори тощо;

– створення нового об’єкту або вибір існуючого: залежно від потреби, оператор може створити новий об’єкт або вибрати вже існуючий об’єкт, до якого будуть копіюватися параметри;

– копіювання параметрів: після визначення об’єкту та параметрів оператор починає процес копіювання. Він копіює значення вказаних параметрів з одного об’єкта на інший. Це може бути корисним для швидкого копіювання налаштувань з одного об’єкта на інший без необхідності ручного введення;

– додаткові дії (необов’язково): оператор може також включати додаткову логіку або функціональність, таку як обробка параметрів перед копіюванням або виконання додаткових дій після копіювання.

Цей оператор може бути викликаний з інтерфейсу користувача Blender або з консолі Python, щоб швидко та ефективно копіювати параметри об’єктів.

Розглянемо вміст файлу «Operation.py» та його функціональність більш детально (рис. 3.5).

```python
class Operation():
    def __init__(self, OperationName,InputObjNeeded=True, maxVert = 0, dropDownName= None):
        self.name = OperationName
        self.MaxVert = maxVert
        self.InputObjNeeded = InputObjNeeded
        self.actions = []
        self.dropDownName = dropDownName


    def AddAction(self,action):
        self.actions.append(action)

    def Execute(self,context):
        scene = context.scene
        props = scene.Procedural_Gen_Props
        for action in self.actions:
            if self.dropDownName and action.dropDownTag != getattr(props,self.dropDownName):
                continue
            action.Execute(context)
        return True

    def AutoUpdate(self,context,Props):
        for action in self.actions:
            if self.dropDownName and action.dropDownTag != getattr(Props,self.dropDownName):
                continue
            if action.liveUpdate and getattr(Props,action.liveUpdateParam):
                action.Execute(context,True)
        return True
```

Рисунок 3.5 – Код файлу «Operation.py»

Клас «Operation» є ключовим компонентом в плагінах програмного коду, так як програма працює з процедурним генеруванням. Цей клас відповідає за виконання різноманітних операцій у межах процедурної генерації. У класі «Operation» зазвичай визначаються різні методи, які реалізують конкретні функції. Цей клас є ключовим у взаємодії з користувачем, оскільки він дозволяє виконувати конкретні дії, визначені у плагіні.

Конструктор класу «Operation» визначається для ініціалізації нового екземпляру класу та встановлення початкових значень його властивостей.

Методи класу «Operation» визначаються для виконання конкретних дій з об’єктами цього класу.

Метод «AddAction» дозволяє додавати нову дію до списку дій, які пов’язані з операцією.

Метод «Execute» виконується у контексті поточної сцени у Blender. Він перевіряє, чи вибір користувача у випадаючому списку співпадає з визначеним

значенням властивості «dropDownName». Якщо таке співпадіння відбувається, то метод виконує відповідні дії операції. У випадку успішного виконання операції, метод повертає значення «True», щоб підтвердити виконання операції. Якщо ж співпадіння не виявлено, операція не виконується, і метод повертає «False» або не повертає нічого для позначення невдалого виконання операції.

Метод «AutoUpdate» відповідає за автоматичне оновлення операції залежно від змін у властивостях. При наявності назви випадаючого списку, він перевіряє, чи вибір користувача в цьому списку співпадає зі значенням властивості «dropDownName». Якщо співпадіння не виявлено, оновлення не виконується. Крім того, якщо властивість «liveUpdate» має значення «True» і властивість «liveUpdateParam» зазнає змін, то метод автоматично виконує відповідні дії оновлення. Наприклад, якщо змінюється значення «liveUpdateParam», яке впливає на операцію, метод здійснить відповідне оновлення. Після виконання оновлення, метод повертає «True», щоб підтвердити успішне виконання оновлення. Це дозволяє визначити, чи було виконано оновлення операції успішно [30].

Файл «ProceduralGenPanel.py» є ключовим при ініціалізації аддону, оскільки він відповідає за генерацію графічної панелі та каркасу аддону. Цей файл містить реалізацію інтерфейсу користувача для застосунку, що був розроблений у Blender (рис. 3.6).

Клас «OBJECT_PT_Procedural_Gen_Panel» є підкласом класу «bpy.types.Panel» і включає різні методи для створення та керування вмістом панелі. Один з основних методів у цьому класі – «draw(self, context)» – відповідає за генерацію вмісту панелі. У цьому методі відбувається вивід назви та іконки панелі, а також розміщення кнопок та інших елементів інтерфейсу, необхідних для взаємодії з користувачем.

```
class OBJECT_PT_Procedural_Gen_Panel(bpy.types.Panel):
    bl_idname = "object.PROCEDURALGEN_PT_procedural_gen_panel"
    bl_label = "Procedural Generation Panel"
    bl_category = "Procedural Generation"
    bl_space_type = "VIEW_3D"
    bl_region_type = "UI"

    def draw(self, context):
        layout = self.layout

        scene = context.scene
        props = scene.Procedural_Gen_Props
        data = DataSingleton()

        if not data.IsInitialized:
            props.CreateOperations(context)


        row = layout.row()
        row.operator('object.procedural_gen_reset_parameters')
        box = layout.box()
        box.label(text="Procedural Generation", icon='GHOST_ENABLED')
        row = box.row()
        row.label(text="Generation Method:")
        row = box.row()
        row.prop(props, "OperationsType",text = "")

        if data.ActiveOperation != None:
            box2 = box.box()
            if data.ActiveOperation.dropDownName:
                row = box2.row()
                row.prop(props, data.ActiveOperation.dropDownName)
            for a in data.ActiveOperation.actions:
                if data.ActiveOperation.dropDownName and a.dropDownTag != getattr(props, data.ActiveOperation.dropDownName):
                    continue
                box3 = box2.box()
                row = box3.row()
                row.label(text=a.UIName)
                if a.liveUpdate:
```

Рисунок 3.6 – Код файлу «ProceduralGenPanel»

Цей код відповідає за створення інтерфейсу панелі для користувача у Blender. Клас «OBJECT_PT_Procedural_Gen_Panel», який є підкласом «bpy.types.Panel», містить різні методи, що дозволяють керувати вмістом цієї панелі.

У методі «draw(self, context)» спершу отримується поточний стан сцени та дані про параметри генерації з об’єкту «Procedural_Gen_Props». Якщо дані про операції генерації не ініціалізовані, виконується метод «CreateOperations(context)», який створює список операцій та додає до нього відповідні дії.

Наступним кроком є створення різних елементів інтерфейсу. Кнопка «Reset Parameters» дає можливість скинути параметри генерації. Після цього

формується блок з назвою «Procedural Generation» та списком випадаючих елементів для вибору методу генерації.

Якщо обрана операція має список випадаючих елементів, то вони автоматично відображаються. Крім того, параметри дій, що входять до складу активної операції, відображаються за допомогою відповідних елементів інтерфейсу, таких як «row.prop» та «box3.column».

Цей код файлу «ProceduralGenPanel.py» містить набір параметрів та методів, які використовуються для створення та керування вмістом графічної панелі інтерфейсу користувача у Blender. Давайте розглянемо кілька ключових елементів з цього коду:

– layout = self.layout: цей параметр дає зручний доступ до створення розмітки на панелі. Він дозволяє організувати елементи інтерфейсу у зручному способі;

– scene = context.scene: тут отримуємо посилання на поточну сцену у Blender, яка відображається у вікні редактора;

– props = scene.Procedural_Gen_Props: цей параметр містить посилання на властивості об’єкта «Procedural_Gen_Props», який зберігає всі параметри для генерації об’єктів;

– data = DataSingleton(): тут створюється об’єкт DataSingleton, який відповідає за поточний стан генерації об’єктів у сцені;

– if not data.IsInitialized: цей рядок перевіряє, чи ініціалізований об’єкт «DataSingleton», щоб переконатися, що усі потрібні дані наявні для роботи;

– props.CreateOperations(context): якщо об’єкт не ініціалізований, то цей метод створює необхідні об’єкти «Operation», які визначають різні операції для генерації об’єктів;

– row = layout.row(): цей рядок створює новий рядок у розмітці панелі для розміщення елементів;

– row.operator('object.procedural_gen_reset_parameters'): тут додається кнопка, яка дозволяє скинути всі параметри генерації, що є зручною функцією для користувача.

Це лише декілька прикладів з коду, які допомагають збудувати та організувати інтерфейс панелі для користувача у Blender.

Далі у коді відбувається ітерація через всі дії вибраної операції за допомогою циклу «for a in data.ActiveOperation.actions:». Цей цикл відповідає за обробку кожного екшена або дії, яка визначена в рамках обраної операції.

Потім ми бачимо блок коду з вкладеними умовами «if» та циклами «for», які відповідають за генерацію відповідних полів для кожної дії. У цьому блоку коду ми перевіряємо, чи необхідно створити «dropdown» поле для конкретної дії за допомогою умови «if data.ActiveOperation.dropDownName». Якщо така необхідність існує, то ми створюємо «dropdown» поле використовуючи «row.prop(props, data.ActiveOperation.dropDownName)».

Потім ми відображаємо UI елементи для кожного параметра дії за допомогою циклу «for p in a.params:». У цьому циклі генеруються відповідні UI елементи для кожного параметра дії, враховуючи різні налаштування, такі як розміщення в стовпчик (if p.Column), відображення лейблів (elif p.UseLabel), та інші.

Далі, якщо поточна операція вимагає введення об’єкту та має обмеження на кількість вершин, ми генеруємо поле для введення максимальної кількості вершин за допомогою «row.prop(props, "MaxInputVertices")».

Завершающим кроком є створення кнопок у нижній частині панелі.

Ці кроки дозволяють розширити можливості звичайних художників за допомогою функціонального коду аддону [31].

3.3 Інструкція з використання розробленого аддону

Перш за все, перед використанням аддону вам необхідно мати встановлений 3D-пакет Blender версії з 3.0 по 3.3, оскільки застосунок був розроблений для цих версій. Після цього вам треба завантажити архів з розробленим аддоном, посилання на який буде надано разом з кваліфікаційною роботою.

Після завантаження архіву з аддоном ви можете підключити його до Blender. Це можна зробити шляхом переходу до налаштувань Blender і переходу до розділу, що відповідає за управління аддонами. Там вам буде запропоновано завантажити або вибрати архів з аддоном з вашого пристрою. Після встановлення аддон буде готовий до використання у Blender (рис. 3.7).

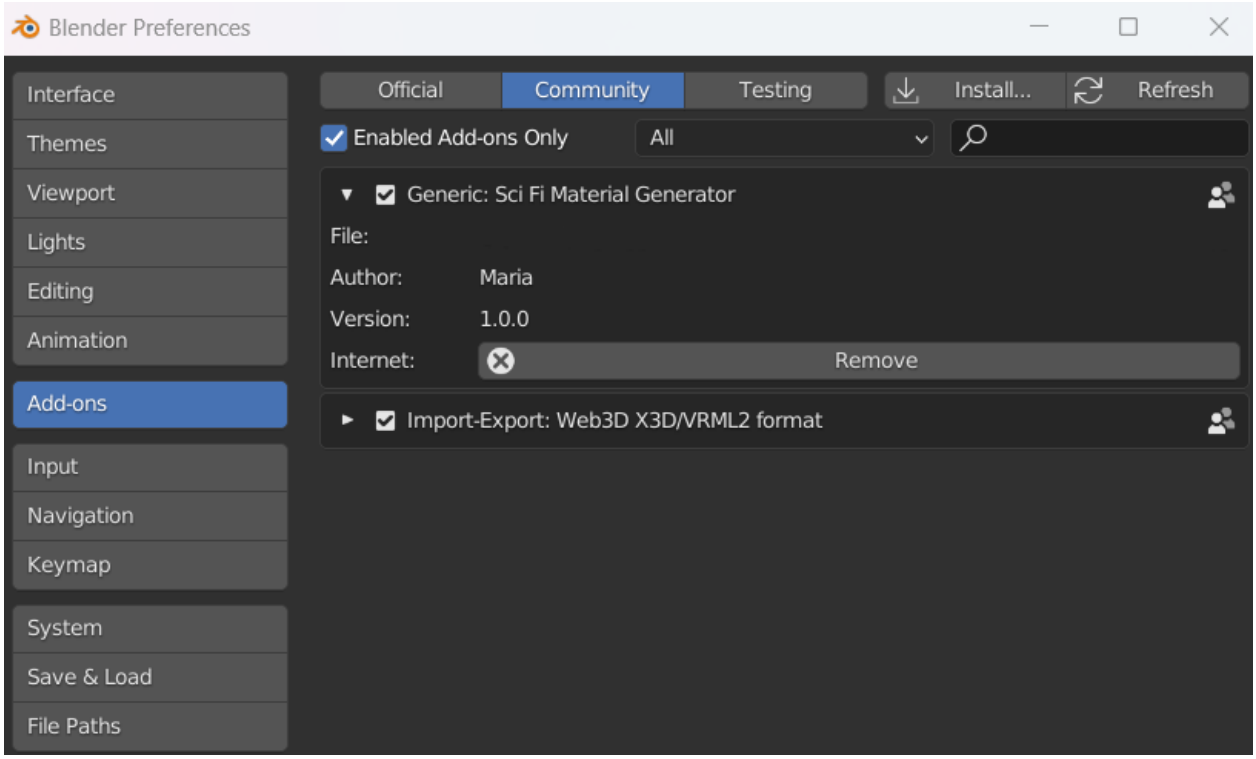

Рисунок 3.7 – Встановлення аддону

Після успішного підключення аддону у налаштуваннях Blender, ви можете відкрити його меню, використовуючи клавішу «N». Ця клавіша відкриває панель усіх активних на даний момент аддонів. Потім вам потрібно

знайти розроблений «Procedural Generation» у цьому списку та вибрати його для відкриття та подальшої роботи з аддоном (рис. 3.8).

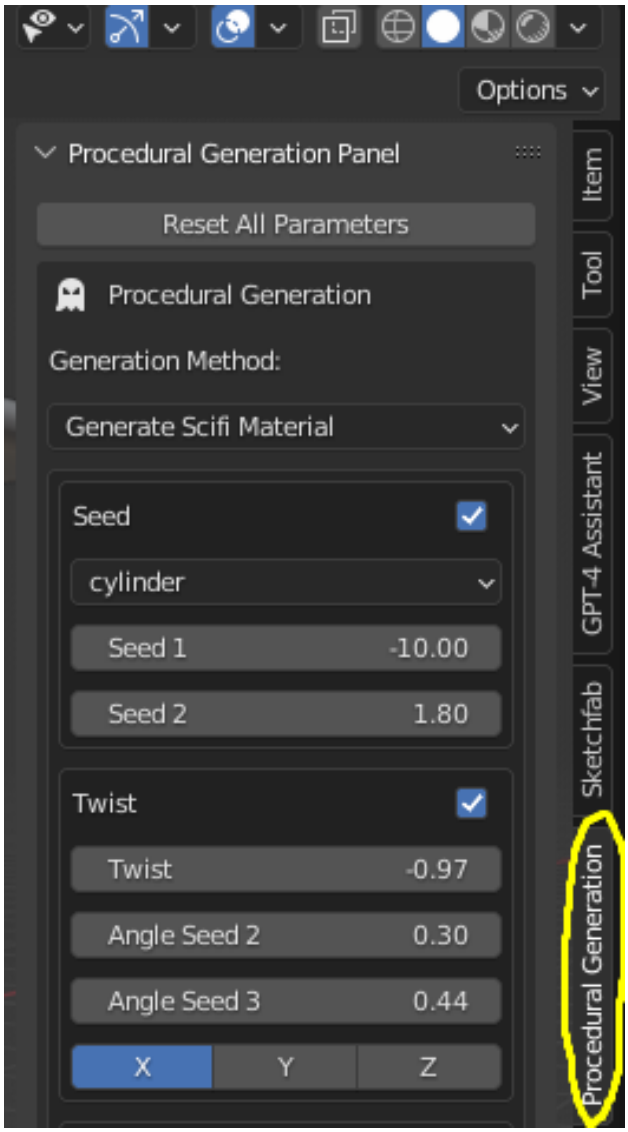

Рисунок 3.8 – Аддон «Procedural Generation»

Після активації аддону відкривається основне вікно аддону, де ви можете вибрати одну з трьох активних пунктів меню, що були описані раніше. Для створення початкового каркасу космічного корабля, до якого потім буде застосована процедурна генерація текстури, слід вибрати пункт «Generate Spaceship Base» у цьому меню. Після цього ви можете налаштувати параметри вручну або вибрати «Randomize All Parameters», особливо рекомендоване для новачків у Blender. Після вибору випадкових параметрів слід натиснути

кнопку «Execute Operation», щоб розпочати процес створення 3D-моделі космічного корабля, який показаний на рисунку 3.9.

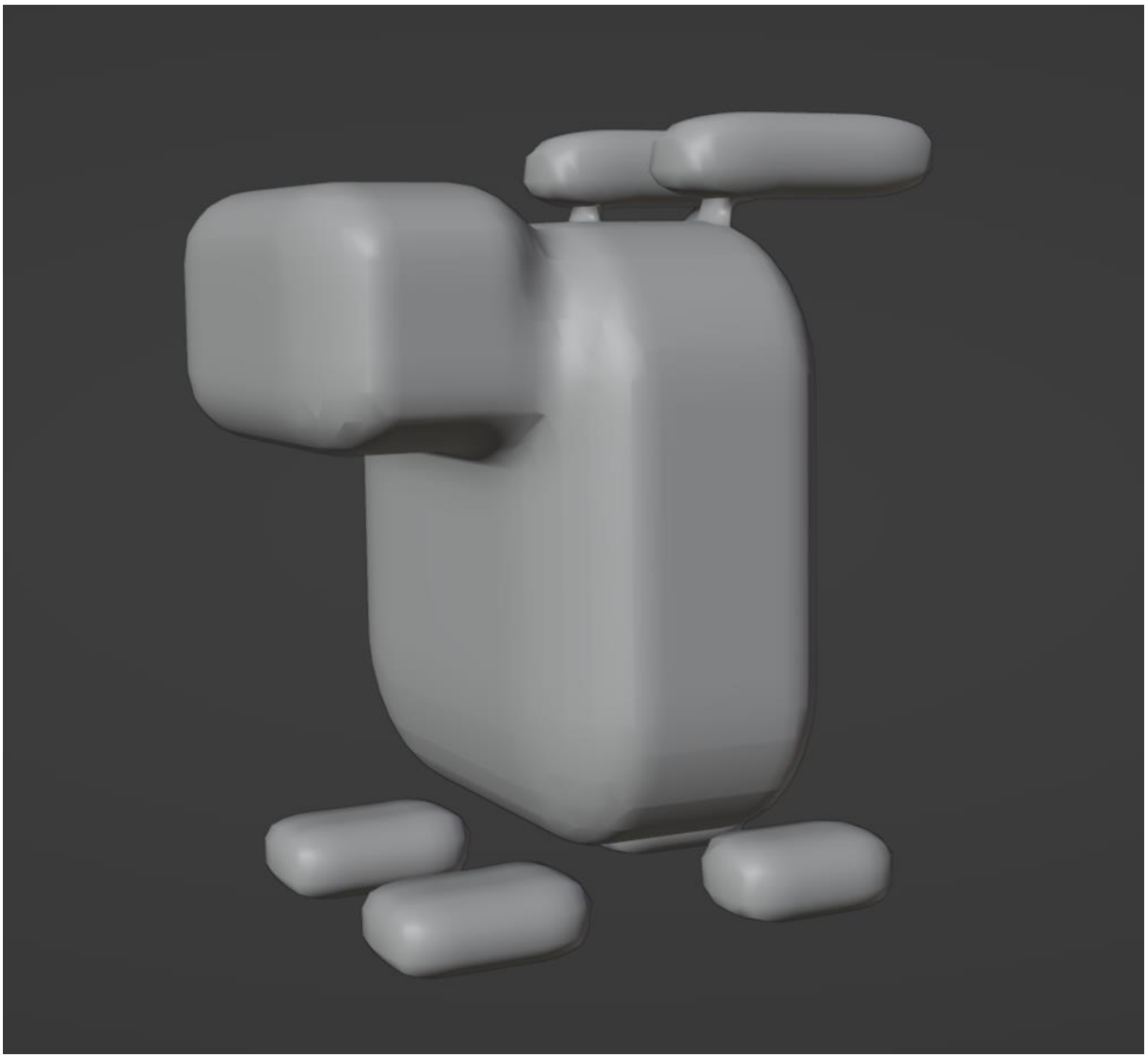

Рисунок 3.9 – Створена модель корабля

Після створення 3D-об’єкту за допомогою процедурної генерації, ми відкриваємо меню для генерації матеріалу для цього об’єкту. Текстура для цього матеріалу також буде створена процедурно.

У меню для генерації матеріалу є багато параметрів, таких як колір акценту, який підкреслить деталі текстури, і колір світла, який відобразиться на частинах текстури, які мають світлові джерела. Інші налаштування дозволяють більш детально налаштувати форму, розмір та яскравість джерел

світла. Щоб отримати результат, схожий на приклад на рисунку 3.10, потрібно вибрати в меню «Generate Sci-fi Material» та натиснути «Randomize All Parameters», а потім «Execute Operation».

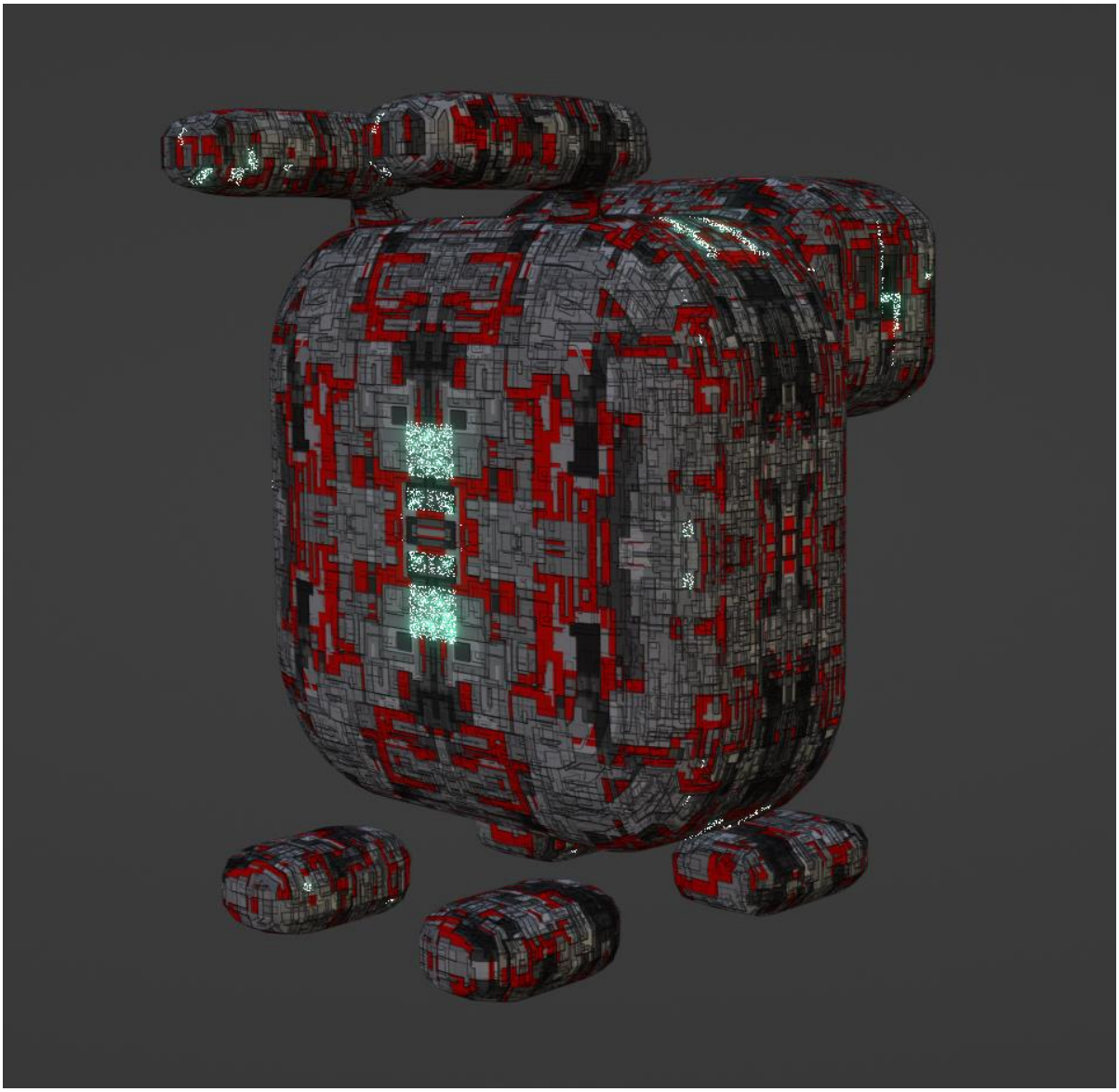

Рисунок 3.10 – Створена модель з текстурами

Після завершення рендерингу можна завантажити наявний результат у різних форматах.

3.4 Тестування та перевірка ефективності ітерацій процедурної генерації у реальних умовах

Давайте розглянемо деякі ітерації в процесі рендерингу з використанням різних параметрів у меню «Generate Sci-fi Material» та «Generate Spaceship Base». Ми будемо змінювати параметри процедурної генерації як вручну, так і за допомогою опції «Randomize All Parameters», а також будемо експериментувати з параметрами Sharp та Smooth. Приклади таких ітерацій наведені нижче (рис. 3.11).

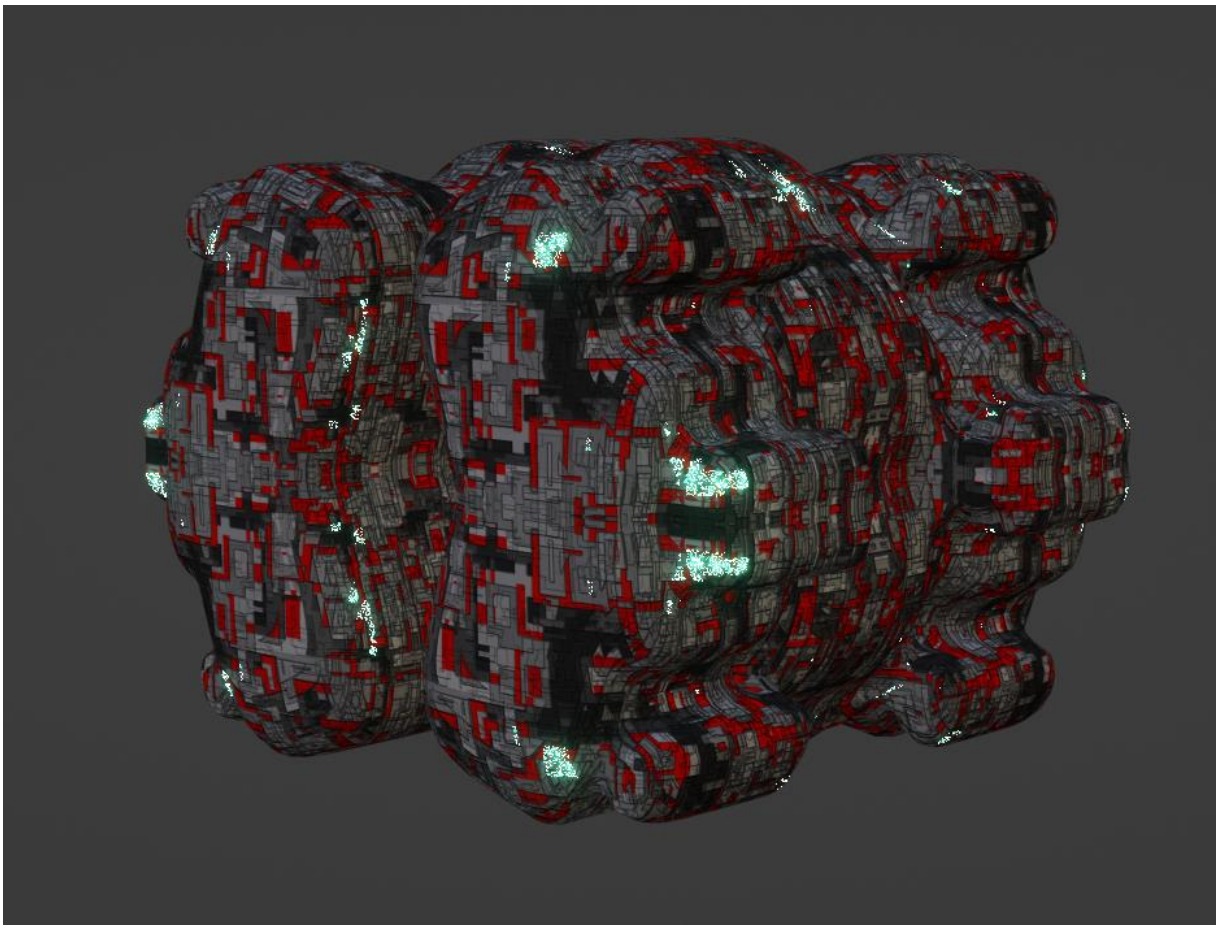

Рисунок 3.11 – Встановлення параметрів «Sharp» з відзеркаленням

Коли ви встановлюєте параметр Sharp з відображенням у вашому програмному застосунку, це дозволяє створити чіткі та виразні грані, які вирізняються на фоні плоскої текстури моделі. Цей параметр дозволяє

контролювати рівень виразності текстури в певних координатах, таких як *X*, *Y* або *Z*. І що цікаво, встановлення параметра відображення дозволяє розширити цю виразність не лише в одній області, а й відобразити її на інших координатах, які ви вказуєте для відображення.

Кожен параметр у вашому програмному застосунку розроблений для того, щоб забезпечити унікальність кожної згенерованої моделі при процедурній генерації. Навіть якщо ви встановите однакові налаштування, кожна нова згенерована модель буде відрізнятися деталями. Це основна концепція процедурної генерації, де кожен параметр впливає на кінцевий вигляд моделі.

Рисунки 3.12 – 3.14 наочно показують, як кожен з цих параметрів впливає на кінцевий результат створення нової моделі при використанні процедурної генерації.

Рисунок 3.12 – Встановлення довільних параметрів «Sharp» з віддзеркаленням тільки по координаті *Z*

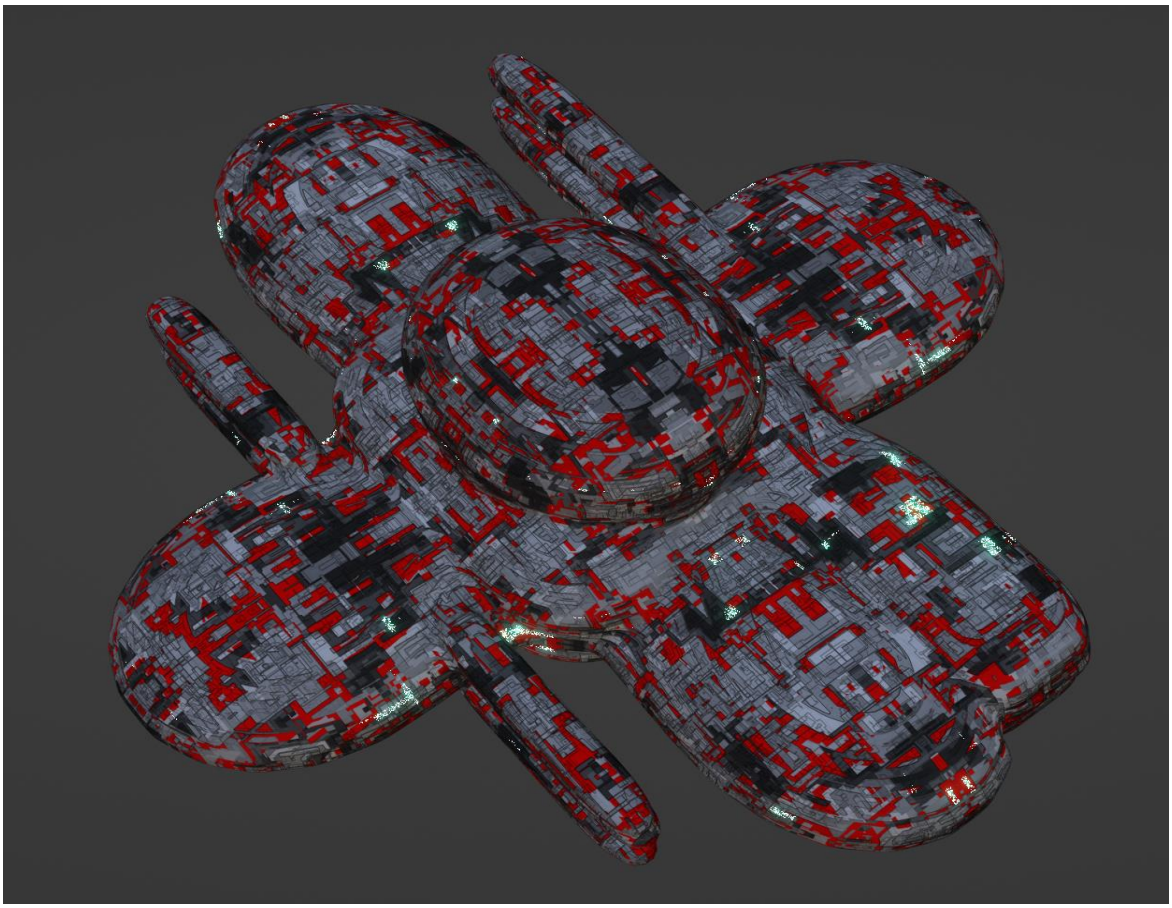

Рисунок 3.13 – Параметр «Randomize All Parameters» та «Smooth»

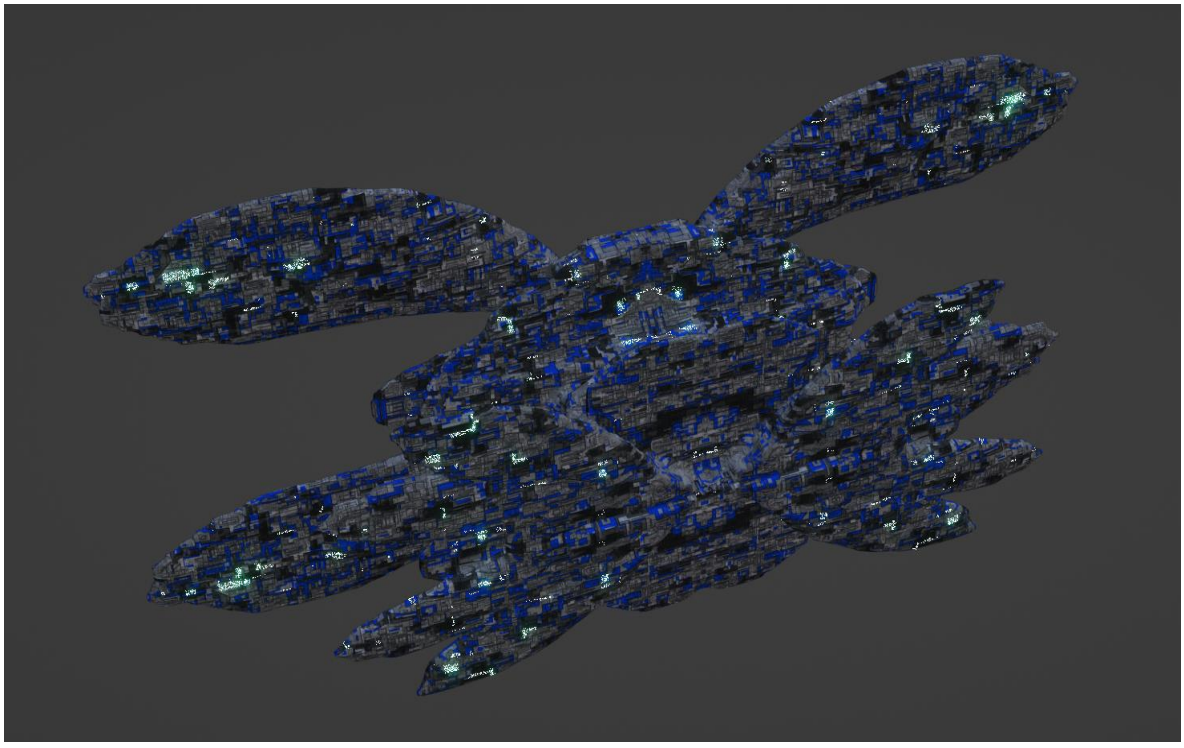

Рисунок 3.14 – Зміна «Accent Color» та застосування довільних параметрів процедурної генерації

Можна зробити висновок, що параметри процедурної генерації мають значний вплив на остаточний результат моделі. Наприклад, встановлення параметрів «Sharp» та «Smooth» змінює вигляд граней і випуклості текстур. Параметри, які відповідають за кольори і освітлення (наприклад, «Accent Color» та «Light Color»), також мають велике значення у визначенні зовнішнього вигляду об’єктів. Крім того, можна відзначити параметри, що дозволяють задавати розміри та форму елементів, такі як «Shape Scale» та «Light Scale», які допомагають створити більш деталізовані та реалістичні моделі.

Важливо також зазначити, що використання параметра відзеркалення дозволяє створювати більш складені інтересні текстури, продовжуючи ефекти там, де це потрібно. Кожен параметр має свої унікальні можливості для налаштування моделі, що дозволяє забезпечити велику різноманітність та індивідуальність у кожній згенерованій моделі.

Загалом, правильне встановлення параметрів процедурної генерації важливо для досягнення бажаного результату і створення унікальних та ефективних 3D-моделей.

ВИСНОВОК

У рамках кваліфікаційної роботи було розроблено аддон для 3D-пакету Blender на мові програмування Python, який використовує методи процедурної генерації.

Ця кваліфікаційна робота засвідчує важливість та перспективність використання технології процедурної генерації у галузі 3D-моделювання та графічного дизайну. Основною метою роботи було розроблення аддону на мові програмування Python для 3D-пакету Blender, що використовує процедурну генерацію для створення складних 3D-моделей, матеріалів та текстур.

Процедурна генерація – це підхід до створення об’єктів та ефектів у графіці, що базується на алгоритмах та параметрах, а не на ручному моделюванні. Вона дозволяє автоматизувати процес створення шляхом визначення правил та параметрів, що вказують на характеристики об’єктів чи текстур.

Розроблений аддон відкриває нові можливості для художників та дизайнерів, дозволяючи їм створювати унікальні та цікаві 3D-моделі з мінімальними зусиллями. Він надає широкий спектр параметрів для налаштування, таких як форма, текстура, колір, освітлення тощо, що дозволяє створювати різноманітні та реалістичні об’єкти.

Важливо зазначити, що процедурна генерація спрощує та прискорює процес розробки, особливо для великих та складних проєктів. Вона також дозволяє зберігати час і ресурси, а також забезпечує консистентність та варіативність результатів.

Отже, аддон на основі процедурної генерації у 3D-пакеті Blender є актуальним інструментом для сучасних художників та дизайнерів, що дозволяє їм швидше та ефективніше реалізувати свої творчі ідеї та забезпечує нові можливості для візуалізації та дизайну. Процедурна генерація є потужним

інструментом для автоматизації процесу створення складних 3D-моделей та анімацій. Її застосовують у багатьох галузях, включаючи відеоігри, фільми, архітектуру та дизайн.

У результаті роботи досягнуто поставленої мети, яка полягала у створенні аддону процедурної генерації, що би спрощувало роботу 3D художникам.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Грабченко, А. І., & Доброскок, В. Л. (2009). Теорія 3D моделювання.

2. Uahistory. URL: https://uahistory.co/pidruchniki/bilenko-technology-10(11)-class-2018/9.php#google_vignette (дата звернення 15.04.2024).

3. Putyatina, O. (2019). An Information Model for Pension Fund Management.

4. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

5. Wikipedia. URL: https://uk.wikipedia.org/wiki/Blender (дата звернення 15.04.2024).

6. Blender Manual. URL: https://docs.blender.org/manual/uk/dev/advanced/scripting/introduction.html (дата звернення 15.04.2024).

7. Sites Google. URL: https://sites.google.com/site/modeluvanna3d/home/aki-najvidomisi-sposobi-stvorenna-3d-modeli (дата звернення 16.04.2024).

8. Kyrychenko, O. S. (2017). Критерії формування готовності до професійної діяльності інженерів на основі 3D-моделювання. Освітологічний дискурс, 296-308.

9. Мартин, Є., & Гончаренко, М. О. (2022). КОМП'ЮТЕРНЕ 3Д–МОДЕЛЮВАННЯ У СЕРЕДОВИЩАХ 3DS MAX ТА AUTOCAD. Прикладна геометрія, інженерна графіка та об'єкти інтелектуальної власності, 1(11), 65-70.

10. Рижавський, К. Є., & Мартин, Є. В. (2020). Розробка концепції навчального онлайн ресурсу для курсу «основи 3д моделювання» (Doctoral dissertation).

11. Фахріян, Д. Ф. (2020). 3Д-моделювання будівель (Bachelor's thesis, КПІ ім. Ігоря Сікорського).

12. Gudvil. URL: https://gudvil.com.ua/ua/blog/scho-take-3d-rendering-6-grundlegende-prinzipien/ (дата звернення 16.04.2024).

13. Addtive. URL: https://addtive.com.ua/ua/shcho_take_3d_modelyuvannya/ (дата звернення 16.04.2024).

14. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, IEEE Access, 10, pp. 124738-124746.

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

16. Тарасов, Н. А., Васюта, С. П., & Хамула, О. Г. (2022). Інформаційні технології 3Д моделювання.

17. Mosiyuk, O. (2018). Особливості вивчення 3D моделювання у процесі професійної підготовки майбутніх учителів інформатики. Науковий вісник Ужгородського університету. Серія:«Педагогіка. Соціальна робота», (2 (43)), 182-186.

18. Свобода, Д. Г. (2012). Міський простір як основа для 3д моделювання каркасних систем міста. Містобудування та територіальне планування, (46), 492-497.

19. Zuo, Z. H., & Xie, Y. M. (2015). A simple and compact Python code for complex 3D topology optimization. Advances in Engineering Software, 85, 1-11.

20. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). *Bionic image segmentation of cytology samples method*. In 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET) (pp. 665-670). IEEE.

21. Arxiv. URL: https://arxiv.org/ftp/arxiv/papers/1807/1807.07824.pdf (дата звернення 19.04.2024).

22. Pradal, C., Boudon, F., Nouguier, C., Chopard, J., & Godin, C. (2019). PlantGL: a Python-based geometric library for 3D plant modelling at different scales. Graphical models, 71(1), 1-21.

23. Sullivan, C., & Kaszynski, A. (2019). PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). Journal of Open Source Software, 4(37), 1450.

24. Andreux, M., Angles, T., Exarchakisgeo, G., Leonardu, R., Rochette, G., Thiry, L., ... & Eickenberg, M. (2020). Kymatio: Scattering transforms in python. The Journal of Machine Learning Research, 21(1), 2256-2261.

25. Lee, G., Gommers, R., Waselewski, F., Wohlfahrt, K., & O'Leary, A. (2019). PyWavelets: A Python package for wavelet analysis. Journal of Open Source Software, 4(36), 1237.

26. Ari, N., & Ustazhanov, M. (2020, September). Matplotlib in python. In 2014 11th International Conference on Electronics, Computer and Computation (ICECCO) (pp. 1-6). IEEE.

27. Ramachandran, P., & Varoquaux, G. (2020). Mayavi: 3D visualization of scientific data. Computing in Science & Engineering, 13(2), 40-51.

28. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, Advances in Electrical and Electronic Engineering, 21(1), pp. 19-27.

29. Zhou, Q. Y., Park, J., & Koltun, V. (2018). Open3D: A modern library for 3D data processing. arXiv preprint arXiv:1801.09847.

30. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, Сучасні інформаційні системи, 6(3), С. 12.

31. Millman, K. J., & Brett, M. (2019). Analysis of functional magnetic resonance imaging in Python. Computing in Science & Engineering, 9(3), 52-55.