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)

РОЗРОБЛЕННЯ ФУНКЦІОНАЛЬНОГО АДДОНУ ДЛЯ 3D-ПАКЕТУ BLENDER
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-1
Герасимов Н.М.
(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)

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 Герасимову Нікіті Максимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення функціонального аддону для 3D-пакету Blender

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 офіційна документація щодо інтерфейсу програмування застосунків API, набір методів та функцій Blender, генеративні результати генерацій.

4. Перелік питань, що потрібно опрацювати в роботі
1. Ознайомлення з поняттям 3D-графіки та моделюванням.
2. Огляд можливостей програмної документації Blender API.
3. Створення функціонального аддону для генерації об'єктів.
4. Результати генерацій.

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	07.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

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

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 59 с., 1 табл., 28 рис., 31 джерело.

РОЗРОБКА АДДОНУ, API BLENDER, ІНТЕГРАЦІЯ PYTHON, РЕНДЕРИНГ ПО ПАРАМЕТРАМ, МОДЕЛЮВАННЯ ВЛАСНОГО GUI, РОЗШИРЕННЯ ФУНКЦІОНАЛЬНОСТІ 3D-ПАКЕТУ.

Об’єктом роботи є функціональний аддон та офіційна документація щодо інтерфейсу програмування застосунків API.

Метою роботи є розробка функіонального аддону для 3D-пакету Blender, для подальшого розширення його можливостей та функцій рендерингу, за допомогою мови програмування Python.

Для тематики кваліфікаційної роботи було розглянуто та отримано навички роботи з графічним інтерфейсом 3D-пакету Blender, проаналізовано методи використання мови програмування для розширення можливостей програми, розглянуто метод алгоритмічного створення даних за допомогою комбінацій алгоритмів.

У результаті роботи був розроблений аддон для 3D-пакету Blender.

ADDON DEVELOPMENT, BLENDER API, PYTHON INTEGRATION, PARAMETER RENDERING, OWN GUI MODELING, 3D PACKAGE FUNCTIONALITY EXPANSION.

The object of the work is a functional addon and official documentation for the API application programming interface.

The purpose of the work is to develop a functional addon for the Blender 3D package, to further expand its capabilities and rendering functions, using the Python programming language.

For the subject of the qualification work, the skills of working with the graphical interface of the Blender 3D package were considered and acquired, the methods of using the programming language to expand the program’s capabilities were analyzed, and the method of algorithmic data creation using combinations of algorithms was considered.

As a result of the work, an addon was developed for the Blender 3D package.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface (набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення)

ПЗ – програмне забезпечення

Рендеринг – це процес створення фінального зображення або послідовності із зображень на основі двомірних або тривимірних даних

ПО – предметна область

ВСТУП

У сучасному світі велику роль відіграють програмні продукти для роботи з 3D-графікою. Blender – це один з найпопулярніших вільних програмних продуктів для створення 3D-моделей, анімації, рендерингу та композитингу. Використовуючи API Blender та мову програмування Python, можна створювати різноманітні аддони, що дозволяють розширювати функціональність Blender за допомогою власних інструментів.

Ця кваліфікаційна робота присвячена розробці функціонального аддону для 3D-пакету Blender на мові програмування Python. Основна мета роботи – створити інструмент, який дозволить збільшити продуктивність та ефективність роботи з Blender. В процесі розробки будуть використані знання з мови програмування Python, API Blender, розробки плагінів та розширення інтерфейсу користувача.

Під час написання цієї роботи будуть розглянуті різні аспекти розробки аддонів для Blender. У результаті роботи буде створено функціональний аддон, що дасть можливість розширити функціональність Blender та забезпечить користувачам зручний та ефективний інструмент для роботи з 3D-графікою.

Актуальність даної роботи полягає в тому, що Blender є одним з найпопулярніших інструментів для створення 3D-моделей, анімації, рендерингу та композитингу. Створення аддонів для Blender на мові програмування Python є важливим завданням, яке дозволяє збільшити функціональність та продуктивність роботи з цим програмним продуктом. Аддони можуть включати в себе різноманітні інструменти, які забезпечують користувачам нові можливості, спрощують рутинні операції та покращують якість та швидкість роботи з Blender.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА МОЖЛИВОСТІ API BLENDER

1.1 Тривимірне моделювання та його особливості у сучасному світі

Тривимірне моделювання (3D-моделювання) – це процес створення 3D-моделей об’єктів та сцен для візуалізації, анімації, виробництва або інших цілей. У порівнянні з двомірним моделюванням, тривимірне моделювання дозволяє більш точно та реалістично відтворювати об’єкти та їхні характеристики у просторі. Існує декілька видів тривимірного моделювання, такі як:

– полігональне моделювання – об’єкти складаються з множини полігонів, таких як трикутники, чотирикутники, п’ятикутники тощо;

– NURBS моделювання – використовується для створення більш складних об’єктів, таких як автомобілі або інструменти, які потребують більш точного моделювання;

– скульптування – використовується для створення моделей шляхом додавання та видалення матеріалу відносно скульптури.

Тривимірне моделювання застосовується в багатьох галузях, таких як:

– ігрова індустрія – 3D-моделювання використовується для створення персонажів, ігрових об’єктів та ігрових світів;

– архітектура та будівництво – 3D-моделі використовуються для візуалізації будівель, квартир та інших приміщень;

– медична індустрія – 3D-моделі використовуються для візуалізації анатомічних структур та для планування хірургічних втручань [1].

Перша 3D-модель була створена в 1963 році Айвеном Сазерлендом на комп’ютері IBM TX-2. Це був простий дротовий фрейм-модель голови, який створювався за допомогою введення координат вручну (рис. 1.1). Пізніше, у 1970-х роках, з’явилися перші системи тривимірного моделювання, такі як Sketchpad та Ivan Sutherland.

Рисунок 1.1 – Дротовий та перший фрейм голови

Сучасне тривимірне моделювання має багато особливостей, що дозволяють створювати більш складні та деталізовані моделі. Наприклад, інструменти для текстурування та освітлення дозволяють створювати більш реалістичні об’єкти, а анімаційні інструменти дозволяють створювати рухомі сцени та персонажів [2].

Тривимірне моделювання також широко використовується в сучасній медіа-індустрії, зокрема в кіно та відеоіграх. Багато крупних студій використовують Blender для створення 3D-анімації, візуалізації та спецефектів. Також, тривимірне моделювання дозволяє створювати віртуальні світи для віртуальної та доповненої реальності.

У сучасному світі тривимірне моделювання має великий потенціал в різних галузях, таких як медицина, наука, архітектура та інженерія. Тривимірні моделі дозволяють більш детально досліджувати об’єкти, прогнозувати їх поведінку та ефективно планувати різноманітні процеси та проєкти.

1.2 Програмний пакет Blender 3D

Blender – це вільна та відкрита програма для тривимірного моделювання, яка забезпечує користувачів усіма необхідними інструментами для створення різних типів 3D-моделей (рис. 1.2).

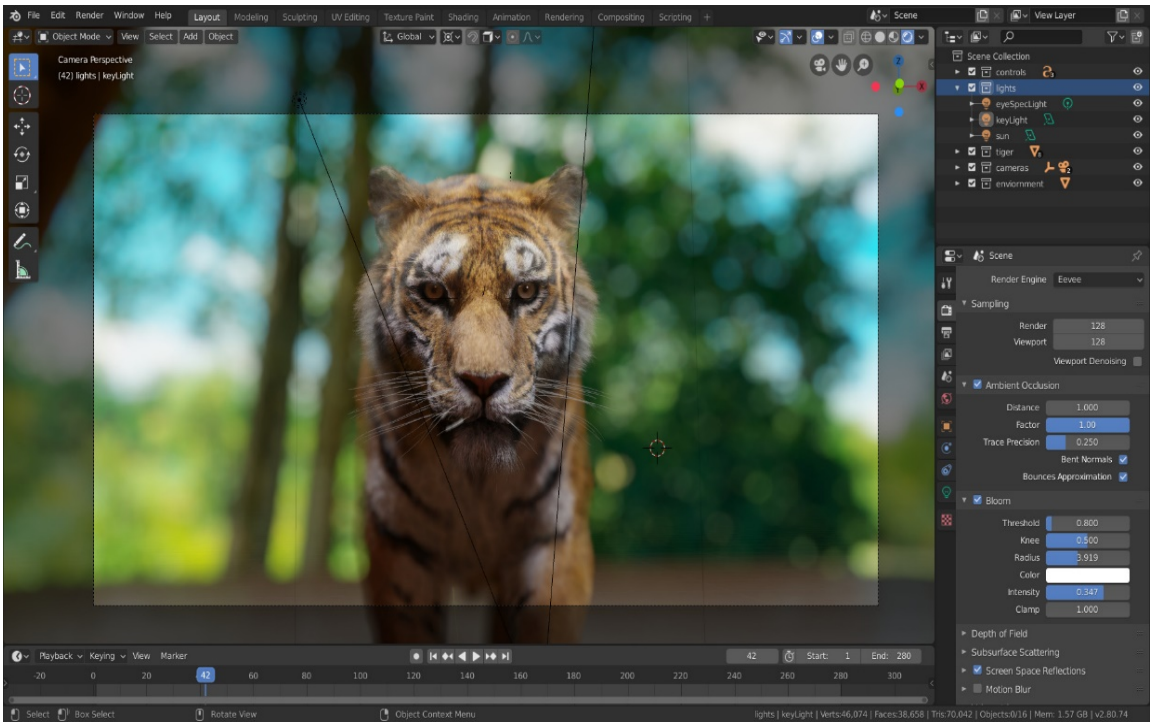

Рисунок 1.2 – Інтерфейс програми Blender

Blender є дуже популярною серед художників, дизайнерів та інших професіоналів, які займаються розробкою 3D-об’єктів для різних цілей, включаючи ігрову та відеоіндустрії, архітектуру, дизайн та інші [3].

У Blender є багато функцій, що дозволяють створювати різні типи 3D-моделей, від малих об’єктів до складних архітектурних моделей та ігрових світів. Деякі з функцій, що надає Blender, включають наступне:

– моделювання: Blender дозволяє користувачам створювати 3D-моделі з нуля за допомогою різних інструментів, таких як скульптурування, булеві операції, поверхневе моделювання, моделювання паттернів та інших;

– текстурування: Blender дозволяє створювати текстури для 3D-моделей та редагувати існуючі. Текстурування включає такі інструменти, як розміщення текстур, блендування текстур, використання фотографій для текстур та інші;

– анімація: Blender має досить потужний редактор анімації, який дозволяє створювати складні анімаційні ефекти для 3D-моделей та сцен;

– візуалізація: Blender дозволяє створювати різні типи візуалізацій, такі як 3D-рендеринг, анімаційні візуалізації та інші.

Окрім базового 3D-моделювання, Blender також має вбудовані функції для створення анімацій, відео- та аудіо-продакшну, а також візуалізації даних. У програмі є різноманітні інструменти для роботи з текстом, зображеннями, планування сценаріїв та інші корисні функції. Крім того, Blender має відкритий код, що дозволяє користувачам модифікувати програму та додавати нові функції за потреби.

Blender є дуже популярною програмою для 3D-моделювання та використовується в різних галузях, таких як графічний дизайн, архітектура, інтерактивна візуалізація, фільми, телебачення та ігри. Вона може бути корисною для художників, дизайнерів, інженерів, архітекторів та багатьох інших фахівців. У програмі є різноманітні інструменти для роботи з текстом, зображеннями, планування сценаріїв, розробка анімації та інші корисні функції.

Цікавим фактом є те, що програма була розроблена як внутрішній проєкт нідерландської студії «NeoGeo» у 1995 році, але в 2002 році була зроблена відкритою та перейшла під ліцензію GNU GPL [4]. У програмі є різноманітні інструменти для роботи з текстом, зображеннями, планування сценаріїв та інші корисні функції. Це означає, що програма є безкоштовною для використання та розповсюдження, і її відкритий код дозволяє спільноті програмістів та користувачів допомагати у розвитку та покращенні програми.

1.3 Python та його застосування в області тривимірної графіки

Python є досить популярною мовою програмування в області тривимірної графіки. Вона використовується для створення скриптів та застосунків для 3D-пакетів, таких як Blender. Python дозволяє зробити роботу з тривимірними об’єктами більш ефективною та продуктивною, завдяки своїм потужним бібліотекам та модулям.

Наприклад, бібліотека PyOpenGL дозволяє створювати візуалізації та анімації на основі тривимірних моделей. Також, Python використовується для обробки та аналізу даних, що забезпечує можливість створення більш складних та ефективних 3D-застосунків. Загалом, використання Python в області тривимірної графіки дозволяє зробити роботу з 3D-об’єктами більш доступною та ефективною. Але розглянемо використання мови програмування Python як інструмент розширення можливостей 3D-пакету Blender.

1.3.1 Мова програмування Python

Python – це високорівнева інтерпретована мова програмування, яка використовується для різноманітних задач, включаючи наукові дослідження, веб-розробку, розробку застосунків та багато іншого.

Однією з основних переваг Python перед іншими мовами програмування є простота синтаксису та лаконічність коду, що дозволяє розробникам швидко та ефективно створювати програми з меншими зусиллями. Крім того, Python є відкритою мовою програмування, що означає, що всі бібліотеки та інструменти для неї є безкоштовними та відкритими для користувачів.

Python також добре підходить для роботи з тривимірною графікою та анімацією, що робить його популярним в області розробки 3D-пакетів, таких як Blender. Завдяки підтримці Python в Blender, розробники можуть легко

створювати застосунки та скрипти для програми, що дозволяє автоматизувати робочий процес та зробити роботу з 3D-об’єктами більш продуктивною та ефективною [5-8].

Python є однією з основних мов програмування, яку використовують для розробки аддонів для програми Blender. Blender API, або програмний інтерфейс, дозволяє програмістам взаємодіяти з Blender через Python. Завдяки цьому інтерфейсу, програмісти можуть розширювати можливості Blender за допомогою власних скриптів на мові Python. Це дозволяє створювати власні інструменти та функції, які покращують продуктивність та зручність користування програмою.

Blender API забезпечує доступ до різноманітних функцій, таких як робота з об’єктами, матеріалами, текстурами, освітленням, камерами та іншими елементами, що дозволяє змінювати їх властивості та додавати нові функції. Python також має велику кількість бібліотек, які можуть бути використані для роботи з 3D-графікою, такі як NumPy, SciPy, PyOpenGL тощо.

Таким чином, засобами Python та Blender API можна створювати складні аддони та інструменти, які значно полегшують процес роботи з 3D-графікою.

1.3.2 Огляд API Blender

Blender API – це набір інструментів програмування, що надається для розширення можливостей 3D-пакету Blender. Цей API (рис. 1.3) дає можливість розробникам створювати свої власні плагіни та скрипти, що можуть бути використані для автоматизації рутинних завдань, додавання нових функцій та покращення робочого процесу.

Blender API використовує мову програмування Python. Python був вибраний як основна мова для Blender API через свою простоту та зручність використання, що дозволяє швидко створювати скрипти та плагіни [9].

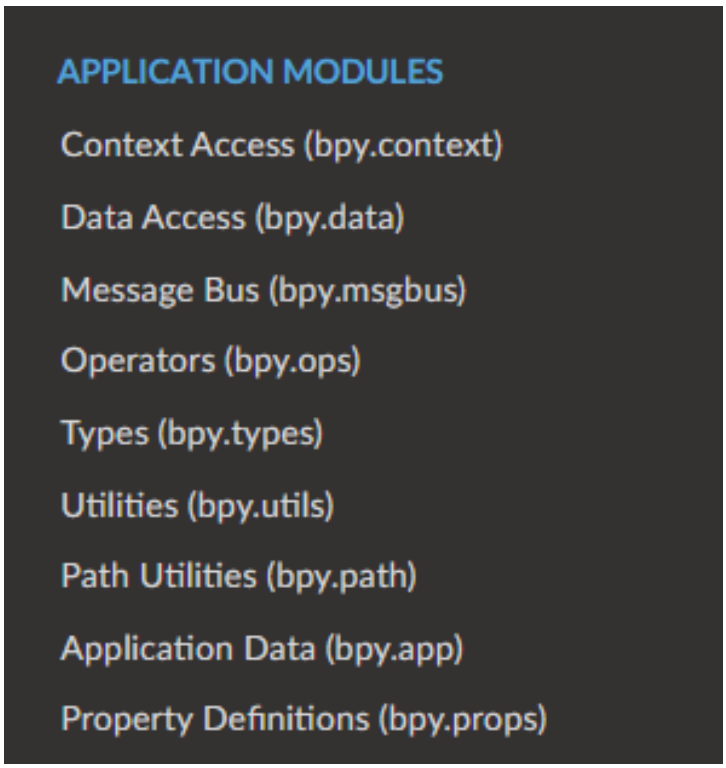

Рисунок 1.3 – Базові компоненти API Blender

Основні методи Blender API включають роботу з об’єктами, матеріалами, текстурами, світлом, анімацією та багато іншого. Розробники можуть використовувати ці методи для створення нових об’єктів, зміни параметрів наявних об’єктів та взаємодії з користувачем.

Команди Blender API поділяються на наступні частини:

– bpy.types: визначення типів об’єктів в Blender;

– bpy.props: визначення властивостей об’єктів в Blender;

– bpy.context: отримання та зміна контексту, в якому виконуються команди;

– bpy.ops: виконання операцій на об’єктах в Blender;

– bpy.data: отримання доступу до даних проєкту в Blender, таких як об’єкти, матеріали та інші.

Деякі приклади команд Blender API:

– bpy.ops.mesh.primitive_cube_add(): створює новий куб у сцені;

– bpy.data.objects[‘Cube’].location = (0, 0, 2): переміщує об’єкт «Куб» до нової позиції у просторі;

– bpy.context.scene.render.filepath = «//output.png»: встановлює шлях до файлу зображення, що буде створений при рендерингу сцени;

– bpy.data.materials[‘Material’].diffuse_color = (1, 0, 0): встановлює колір матеріалу «Material» на червоний.

Blender API дозволяє розробникам налаштовувати та контролювати всі аспекти Blender, нижче наведений приклад коду (рис. 1.4) для створення довільної анімації 3 кубів в 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 довільної анімації

Цей код створює 3 куби з випадковими координатами та розмірами, а потім на кожному кадрі збільшує їх висоту на 0,1. Результатом буде анімація з 50 кадрів, на яких куби будуть підніматись вгору.

Blender API дозволяє розробникам налаштовувати та контролювати всі аспекти Blender, що робить його потужним інструментом для розширення функціоналу програми [10-12].

1.4 Процедурна генерація та мета розробки аддону

Процедурна генерація – це метод створення візуального контенту, який базується на алгоритмічних процедурах, а не на ручному моделюванні. Такий підхід дозволяє швидко та ефективно створювати складні візуальні ефекти та об’єкти, які можуть бути складними для ручного моделювання. Процедурна генерація використовується в багатьох галузях, включаючи відеоігри, кіно, архітектуру, дизайн та інші. Вона дозволяє створювати складні та великі об’єкти та сцени швидко та ефективно, що дозволяє зберігати час та кошти.

Процедурна генерація пов’язана з 3D-моделями тим, що дозволяє створювати об’єкти та сцени за допомогою математичних формул, алгоритмів (рис. 1.5) та інших процедур. Це означає, що об’єкти можуть бути створені шляхом задання параметрів, які визначають їхні форми та текстури, замість ручного моделювання.

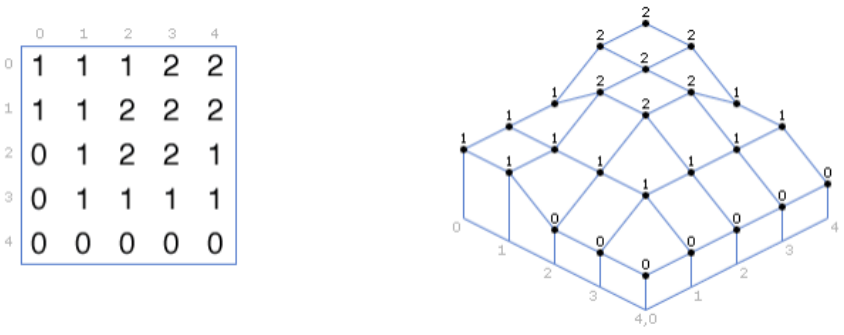

Рисунок 1.5 – Процедурна генерація по параметрам

У програмі Blender процедурна генерація може бути реалізована за допомогою різних інструментів, включаючи плагіни та скрипти. Наприклад, можна використовувати мову програмування Python та її бібліотеки, щоб створювати процедурно згенеровані об’єкти та сцени. Іншим інструментом, який може бути використаний для процедурної генерації в Blender, є вбудовані генератори, такі як Solidify, Bevel, Array та інші.

Процедурна генерація може бути використана для створення різних ефектів та об’єктів в Blender. Наприклад, можна створити процедурно згенеровані ландшафти, дерева, та матеріали – саме їх буде створювати наш плагін. Що ж, після розуміння того, що таке процедурна генерація, можемо розглянути, як можна використовувати її для розробки плагінів для Blender.

За допомогою процедурної генерації можна побудувати плагін для Blender, який автоматично генерує складні 3D-моделі з параметрами, заданими користувачем, як на прикладі нашого плагіну (рис. 1.6). Наприклад, плагін може створювати складні пейзажі з гір, долин і річок, або генерувати реалістичні створіння з тваринних та рослинних елементів.

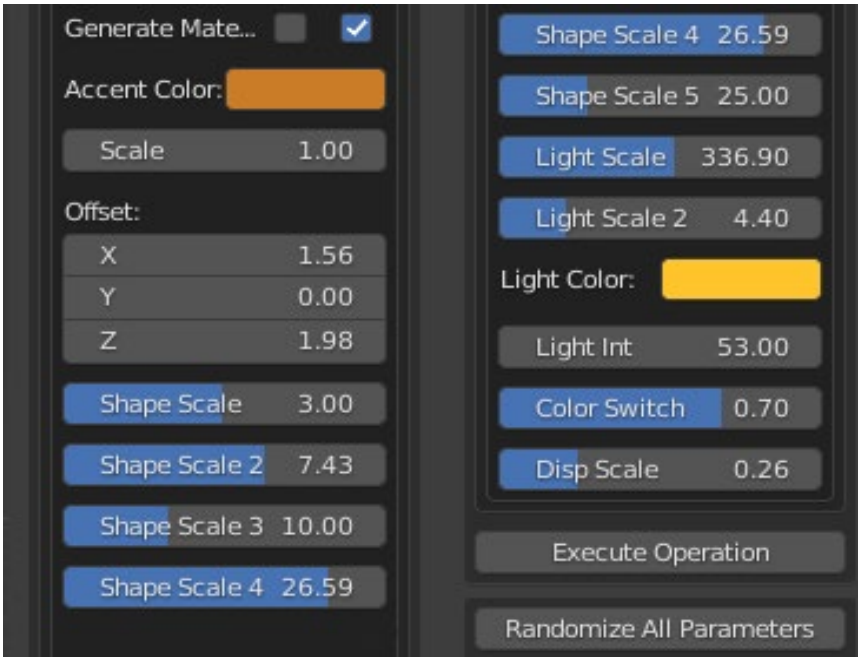

Рисунок 1.6 – Параметри процедурної генерації розробленого аддону

Такий плагін може бути корисним для художників-ілюстраторів, які швидко хочуть створити велику кількість різних 3D-моделей з різними параметрами. Він може також допомогти при розробці ігор, де потрібно генерувати складні середовища і персонажів.

Програмування такого плагіну зазвичай виконується з використанням мови Python та Blender API. Користувач може вказати параметри для генерації 3D-моделі через графічний інтерфейс, а плагін буде використовувати ці параметри для створення 3D-моделі за допомогою алгоритмів процедурної генерації.

Отже, побудова плагіну з використанням процедурної генерації може допомогти значно збільшити продуктивність і ефективність процесу розробки 3D-моделей, забезпечуючи автоматизовану генерацію складних 3D-моделей з параметрами, заданими користувачем.

1.5 Постановка задачі

Після огляду вищеописаних тем, можемо зробити підсумок, що розробка на мові Python для 3D-пакету Blender аддону, який використовує процедурну генерацію матеріалів, текстур чи об’єктів є актуальною темою та надає 3D-дизайнерам більш ефективну, зручну та швидку роботу з 3D-сценами.

Об’єктом роботи є функціональний аддон та офіційна документація щодо інтерфейсу програмування застосунків API.

Метою роботи є розробка функіонального аддону для 3D-пакету Blender, для подальшого розширення його можливостей та функцій рендерингу, за допомогою мови програмування Python.

Для досягнення мети необхідно вирішити такі завдання:

– отримати базові навики управління 3D-пакетом Blender;

– проаналізувати існуючі функції модифікаторів, які є базовими параметрами процедурної генерації;

– ознайомитися з інтерфейсом програмування застосунків API;

– реалізувати алгоритми та шкалу параметрів для процедурної генерації;

– розробити каркас аддону та додати до нього вищеперелічений функціонал.

2 ПОПЕРЕДНЄ МОДЕЛЮВАННЯ СИСТЕМИ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ПО ПАРАМЕТРАМ

2.1 Базові функції API для побудови процедур

Blender API є потужним інструментом для розробки різноманітних застосунків, які можуть взаємодіяти з Blender. Одним з ключових компонентів Blender API є application modules, які надають доступ до основних функцій та функціональності Blender [13].

Нижче наведено докладний опис кожного з application modules та їх параметрів:

– bpy.app.handlers.

Цей модуль дозволяє реєструвати функції-обробники подій, які викликаються в певних точках життєвого циклу Blender. Параметри, які можна передати у функції-обробники, включають контекст, подію та будь-які додаткові параметри, що були зареєстровані під час реєстрації обробника подій.

Лістинг 2.1 Приклад застосування модулю реєстрації функції-обробника:

```
import bpy
def my_handler(scene):
print("Scene", scene.name, "changed")
bpy.app.handlers.scene_update_pre.append(my_handler)
```

– bpy.app.version.

Цей модуль надає інформацію про версію Blender, що використовується. Параметри, які можна отримати з цього модулю, включають номер версії, назву релізу та дату випуску.

– bpy.app.context.

Цей модуль містить інформацію про поточний контекст Blender, такий як видима область та виділені об’єкти. Параметри, які можна отримати з цього модулю, включають активний об’єкт, активний матеріал та інші.

Лістинг 2.2 Приклад застосування модулю інформації:

```
import bpy
print("Active object:", bpy.context.active_object.name)
```

– bpy.app.data.

Цей модуль дозволяє отримати доступ до різних даних, які зберігаються в Blender, таких як сцени, об’єкти та матеріали. Параметри, які можна отримати з цього модулю, включають список усіх об’єктів у сцені, список усіх матеріалів та інші.

Лістинг 2.3 Приклад застосування модулю доступу:

```
import bpy
for obj in bpy.data.objects:
print(obj.name)
```

– bpy.app.ops.

Цей модуль надає доступ до різноманітних операцій, які можна виконати в Blender, таких як створення об’єктів та налаштування їх властивостей. Параметри, які можна передати у функції-операції, включають контекст та параметри операції [14].

Лістинг 2.4 Приклад застосування модулю доступу:

```
screen = bpy.context.screen
import bpy
```

bpy.ops.mesh.primitive_cube_add(size=2, location=(0,0,0))

– bpy.app.ui.

Цей модуль дозволяє змінювати розміщення елементів інтерфейсу користувача в Blender, таких як панелі та меню. Параметри, які можна передати у функції, включають тип елемента інтерфейсу та його параметри.

Лістинг 2.5 Приклад застосування модулю змінення:

```
import bpy
screen = bpy.context.screen
areas = [area for area in screen.areas if area.type == 'VIEW_3D']
```

– bpy.app.debug.

Цей модуль дозволяє налаштовувати різні параметри відлагодження в Blender, такі як рівень журналу та поведінка відлагоджування. Параметри, які можна настроїти за допомогою цього модулю, включають рівень журналу, інформацію про виконання коду та інші.

Лістинг 2.6 Приклад застосування модулю налаштування:

```
import bpy
bpy.app.debug_value = True
bpy.app.debug_log_level = 'DEBUG'
areas = [area for area in screen.areas if area.type == 'VIEW_3D']
```

Модуль bpy.utilities в Blender API містить різні утиліти, що можуть допомогти у розробці скриптів для Blender. Цей модуль містить функції та класи, які допомагають здійснювати різні завдання, такі як робота зі списками об'єктів, обробка файлів та робота зі строками. Наступним прикладом буде таблиця (табл. 2.1) зі списком функцій та класів модуля bpy.utilities у Blender API 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 у Blender для використання з інших скриптів.
bpy.utils.unregister_module()	__name__: str	1 un-escaped string	Скасовує реєстрацію модуля у Blender.
bpy.utils.previews	filepath: str, preview_id: str	1 resource path	Завантажує та зберігає попередній перегляд для об’єкта у Blender.
bpy.utils.scene_state_compare()	scene1: bpy.types.Scene, scene2: bpy.types.Scene	1 string	Порівнює дві сцени та повертає різницю між ними.

Продовження таблиці 2.1

1	2	3	4
bpy.utils.benchmark()	func: Callable, *args: Any, **kwargs: Any	1 list	Вимірює час виконання функції у Blender.
bpy.utils.string_paths( )	string: str	1 string	Перетворює рядок шляхів у список шляхів.
bpy.utils.blend_paths()	string: str	1 string	Перетворює рядок зі списком шляхів у форматі Blender у список шляхів.

Ця таблиця містить інформацію про кожну функцію та клас, які містить модуль bpy.utilities у Blender API. Для кожної функції та класу вказані параметри, які вони приймають, а також короткий опис того, що робить функція або клас. Використання цих параметрів допоможе розробникам краще розуміти, як працюють ці утиліти та як їх можна використовувати у своїх програмах [15-17].

2.2 Параметричні модифікатори як основа параметрів

Модифікатори у Blender це інструменти, які дозволяють змінювати топологію об’єктів шляхом додавання спеціальних ефектів, які впливають на їх форму та текстуру. Вони дозволяють здійснювати різноманітні операції з об’єктами без необхідності вручну редагувати кожен окремий полігон.

Розроблений аддон процедурної генерації використовує наступні модифікатори.

Normal Edit Modifier в Blender – це модифікатор, який дозволяє змінювати напрямок нормалей поверхні об’єкта. Нормалі – це вектори, які вказують напрямок від поверхні об’єкта. Їх можна змінювати за потребою як на рисунку 2.1.

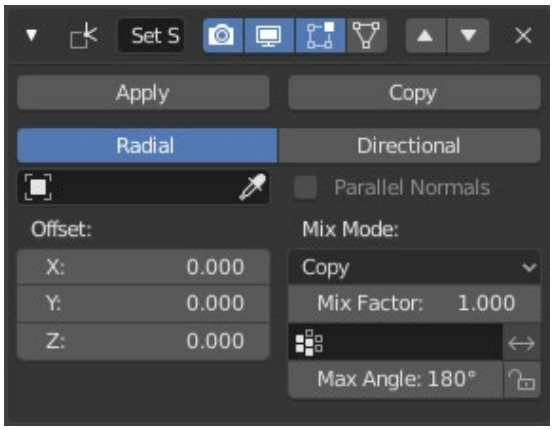

Рисунок 2.1 – Опції модифікатора

Цей модифікатор може використовуватися для швидкого генерування радіальних нормалей для листя низько-полігонального дерева або «виправлення» відтінювання мультико-подібного рендерингу як на рисунку 2.2.

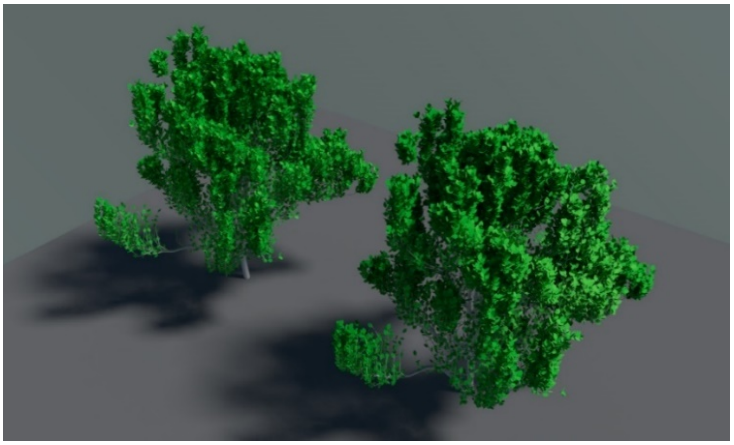

Рисунок 2.2 – Приклад редагування кастомних нормалей

Модифікатор Subdivision Surface – це один з найбільш популярних і потужних модифікаторів у Blender. Він дозволяє збільшити кількість полігонів у меші, зберігаючи її форму і деталі [18].

При застосуванні цього модифікатора кожен полігон розділяється на більшу кількість більш маленьких полігонів, що дозволяє отримати більш плавні форми та зменшити видимість візуальних артефактів, таких як кути або видимі шви.

Існує математична формула для модифікатора Subdivision Surface, який ділить кожен полігон на більшу кількість більш маленьких полігонів:

$$P_{new} = \frac{(P_1+P_2+P_3+...+P_n)}{n}, \quad (2.1)$$

де *P1, P2, P3, ..., Pn* – це координати вершин початкового полігона;

Pnew – координати вершин нових полігонів, що створюються в результаті підрозділу;

n – кількість вершин у нових полігонах.

Ця формула дозволяє визначити нові координати вершин, що створюються внаслідок підрозділу, за допомогою координат вершин початкового полігона [19].

Наступним буде показано приклад реалізації модифікатора Subdivision Surface (рис. 2.3).

Рисунок 2.3 – Приклад згладження об’єкта Subdivision Surface

Модифікатор Bevel у Blender дозволяє автоматично закруглювати ребра об’єктів (рис. 2.4), що дозволяє створювати більш органічні та гладкі форми. Цей модифікатор дуже корисний для моделювання меблів, техніки, будівель та інших об’єктів з округлими краями.

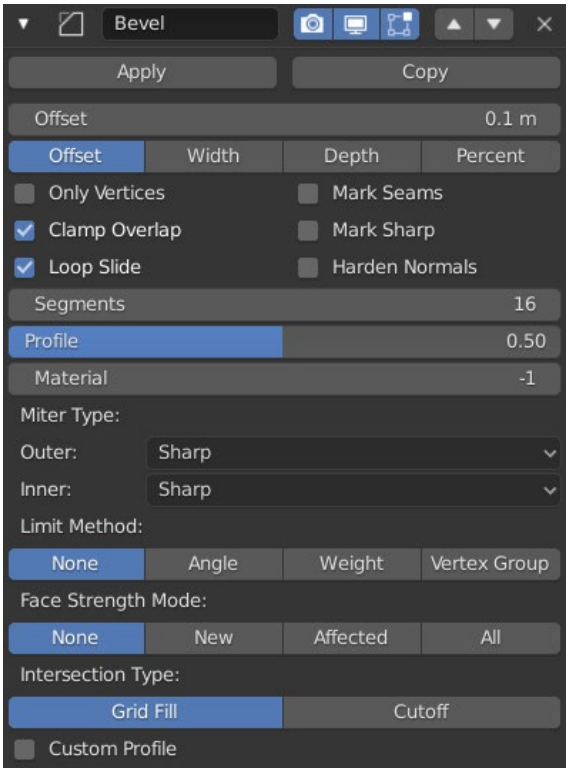

Рисунок 2.4 – Опції модифікатора Bevel

Формули для модифікатора Bevel можуть змінюватися в залежності від вибраних параметрів, а також можуть використовувати різні методи розрахунку нових координат вершин та ребер. Наступні формули є прикладом:

$$Nc = (1-w) * Oc + w, \tag{2.2}$$

$$B = amount * L, \tag{2.3}$$

$$P(x) = 1-(1-x)^n, \tag{2.4}$$

де Nc – нова координата вершини;

w – вага;

Oc – стара координата вершини;

lt – зсув вершини;

B – відсоток від довжини ребра;

L – довжина ребра;

P – профіль;

X – відстань вздовж профілю.

Модифікатор boolean в Blender дозволяє об’єднувати, віднімати або перетинати меші між собою на основі логічних операцій *AND*, *OR* і *NOT*.

Для застосування модифікатора boolean потрібно вказати дві меші, які потрібно об’єднати або відняти. Результат операції залежить від типу логічної операції, яку було обрано (рис. 2.5).

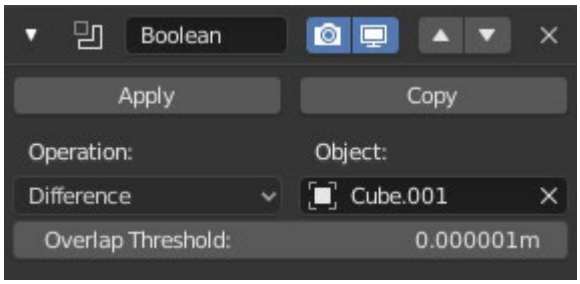

Рисунок 2.5 – Опції модифікатора Boolean

Операції булевої логіки в модифікаторі boolean відповідають наступній системі формул:

$$A \vee B = 1, \tag{2.5}$$

$$A \wedge B = 1, \tag{2.6}$$

$$A \wedge notB = 1, \tag{2.7}$$

$$AxorB = 1, \tag{2.8}$$

де (2.5) – об’єднання (Union);

(2.6) – перетин (Intersection);

(2.7) – різниця (Difference);

(2.8) – симетрична різниця (Symmetric Difference).

У цих формулах *A* та *B* відповідають відповідним об’єктам, а «not» використовується для зворотнього значення (інверсії) логічної змінної.

Роблячи висновок можемо сказати, що модифікатор boolean дуже корисний для створення складних форм та геометричних об’єктів. Однак, він може бути вимогливий до ресурсів комп’ютера, тому слід бути обережним при використанні його на складних мешах.

2.3 Процедурно-генеративні параметри

Розроблений аддон для розширення функціоналу Blender застосовує вищеперераховані функії, процедури та модифікатори для створення об’єктів процедурної генерації по параметрам, давайте розглянемо, які саме параметри утворюють процедури та функції, які було розглянуто [20]. Тематика розробленого аддону пов’язана з науковою фантастикою та космічними кораблями, саме тому в аддоні є три основні частини:

– mesh scramble – звичайний параметризований редактор об’єктів;

– generate spaceship base – меню для процедурної генерації довільної форми космічного корабля;

– generate sci-fi material – меню для процедурної генерації матеріалу (текстури) попередньо створеного космічного корабля.

Розглянемо параметри кожної частини більш детально, як на прикладі Mesh Scramble (рис. 2.6).

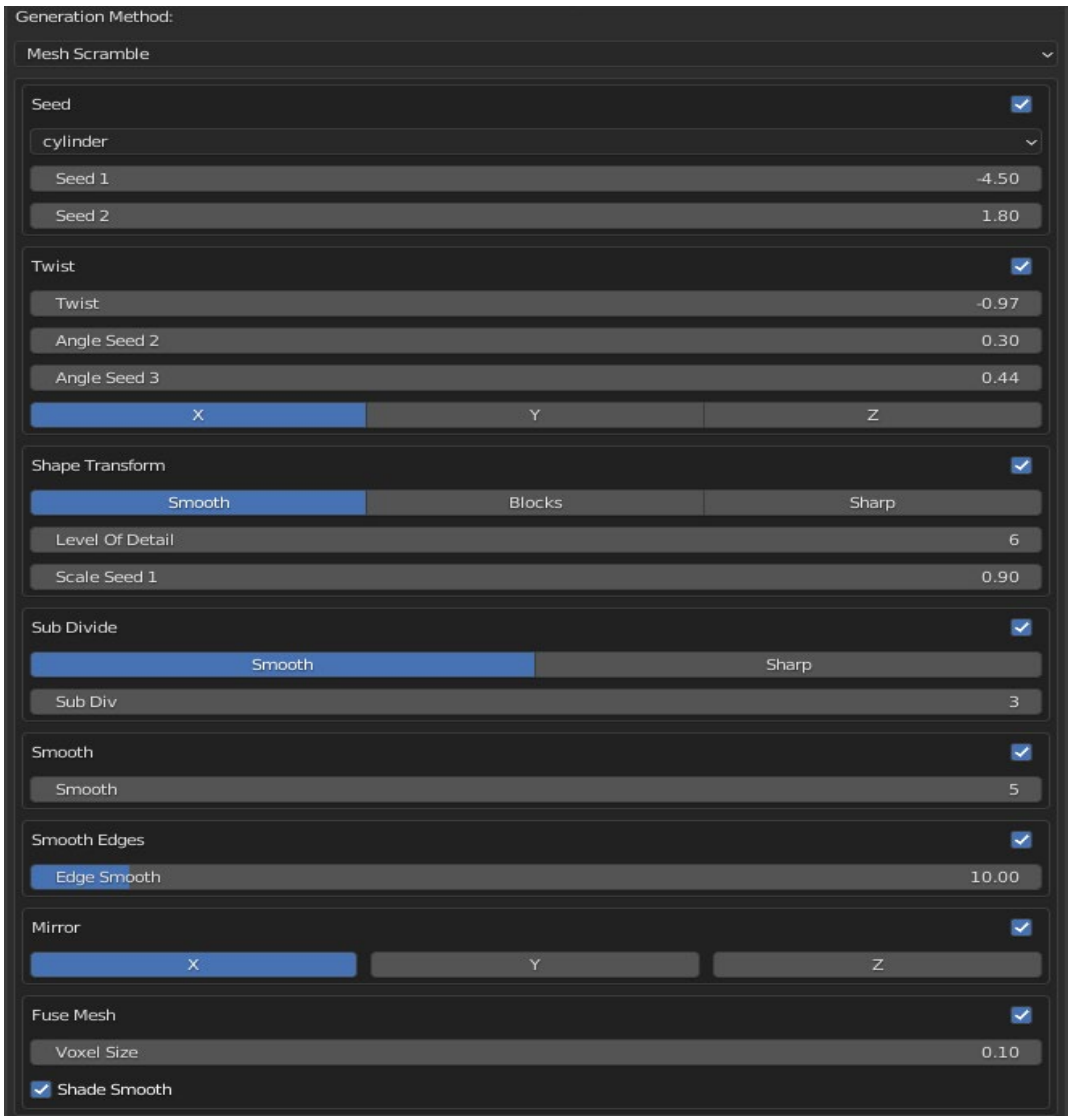

Рисунок 2.6 – Параметри генерування Mesh Scramble

Значення параметрів:

– twist – встановлює значення вигину кінцевого об’єкта, також можна встановити значення кута вигину;

– shape Transform – встановлює спочатку режим перетворення форми об’єкта, потім встановлює значення цього перетворення;

– smooth – встановлює значення гладкості об’єкту, якщо перед цим був вибраний режим Smooth;

– smoth edges – встановлює значення гладкості ребер об’єктів;

– mirror – встановлює по якій осі повинна бути відзеркалена інша половина моделі;

– fuse mesh – встановлює параметр склеювання полігонів між собою.

Далі буде розглянуто меню для процедурної генерації довільної форми космічного корабля, а саме Generate Spaceship Base (рис. 2.7).

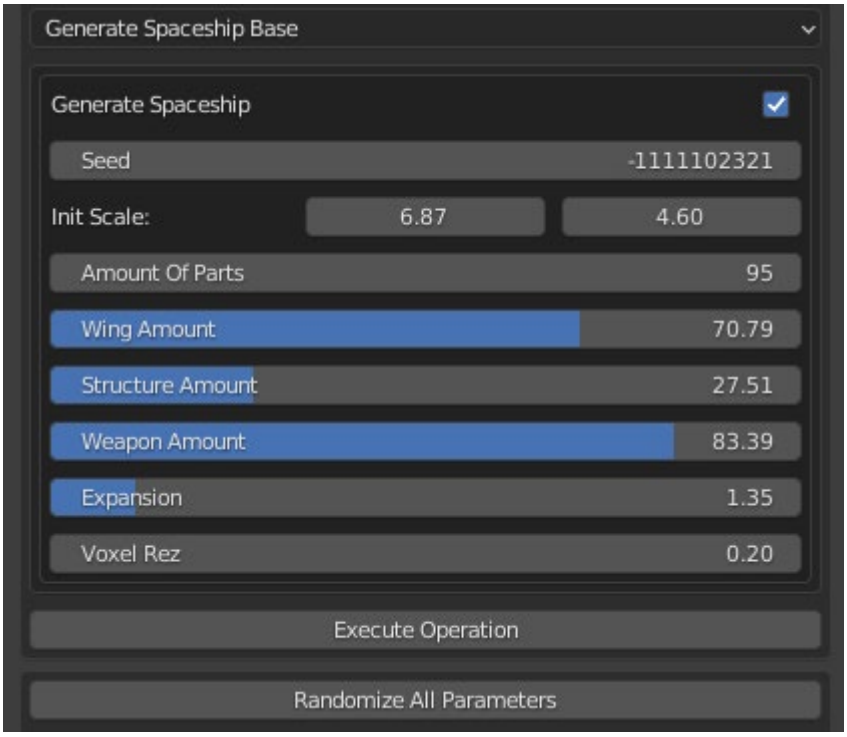

Рисунок 2.7 – Параметри генерування Generate Spaceship Base

Значення параметрів:

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

– wing amount – процентна кількість крил для космічного корабля;

– structure amount – кількість унікальних об’єктів для однієї моделі;

– weapon amount – кількість об’єктів довільної форми які будуть нагадувати космічну зброю;

– voxel rez – параметр роздільної здатності кожного полігону;

– execute operation – кнопка для відтворення усіх встановлених вище параметрів;

– randomize all parameters – кнопка для встановлення випадкових вищевказаних параметрів [21].

Далі буде розглянуто меню для процедурної генерації матеріалу (текстури) попередньо створеного космічного корабля (рис. 2.8).

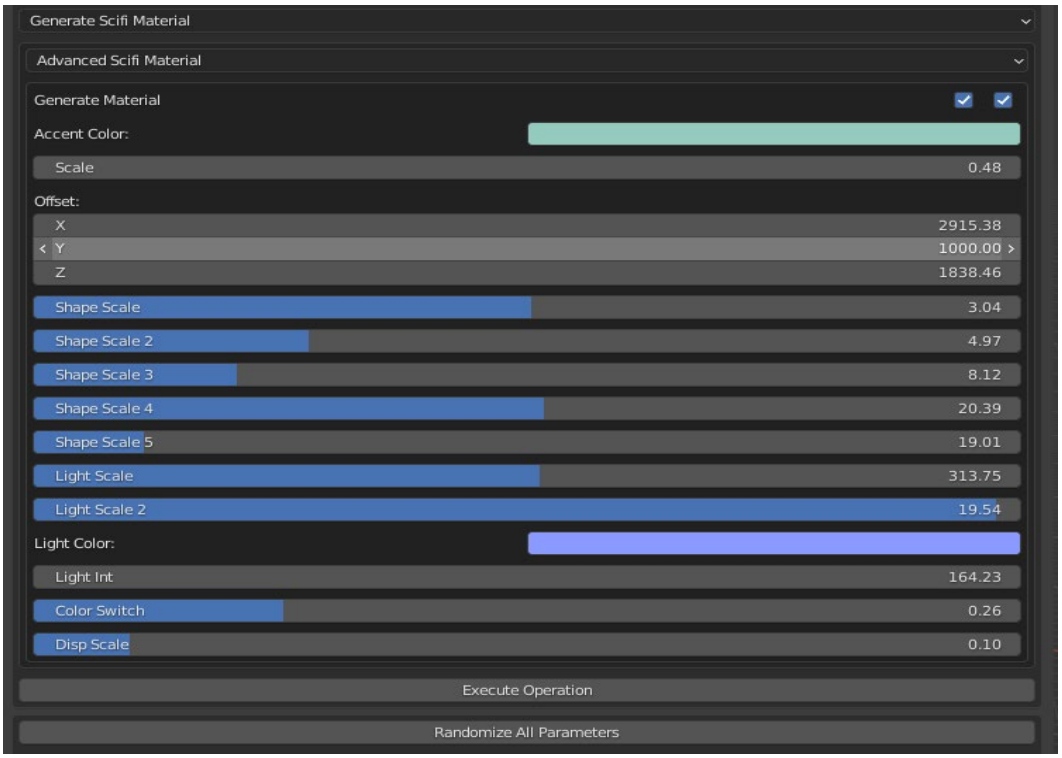

Рисунок 2.8 – Параметри генерування Generate Sci-Fi Material

Значення параметрів:

– accent color – колір частинок матеріала;

– offset – встановлюється значення зсуву текстури по всім 3 осям;

– shape scale – встановлюється загальний розмір та випуклість текстур;

– shape scale 2, 3, … – встановлюється загальний розмір та випуклість кожного виду текстур;

– light scale – встановлюється розмір частинок з акцентним кольором які є джерелами світла;

– color switch – встановлює баланс між акцентним кольором та кольором частинок з джерелом світла;

– disp scale – встановлюється розмір частинок з акцентним кольором які є джерелами світла.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ АДДОНУ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ

3.1 Огляд середовища програмної реалізації

У рамках кваліфікаційної роботи було розроблено аддон на мові Python для 3D-пакету Blender, який використовує процедурну генерацію матеріалів, текстур та об’єктів. Для реалізації були використані інструменти розробника у програмі Blender, які розширюють його функціонал та допомогають розробникам більш ефективніше працювати з цим 3D-пакетом [22]. Для того щоб активувати усі функції розробника в Blender, треба перейти в основні налаштування інтерфейсу програми та увімкнути два пункти як на рисунку 3.1.

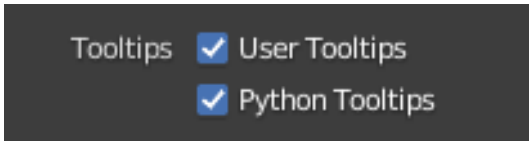

Рисунок 3.1 – Налаштування інтерфейсу для початку розробки

У Blender є вбудований текстовий редактор, який дозволяє користувачам писати скрипти на мові Python. Цей редактор можна знайти вкладку «Text Editor» (Текстовий редактор) у верхній частині інтерфейсу Blender.

З допомогою текстового редактора Blender можна створювати скрипти, які виконують певні завдання, такі як автоматизація процесу моделювання, анімації та інших операцій в Blender. Ці скрипти можуть бути виконані шляхом введення їх в консоль Blender або використання їх як додаткових застосунків.

Text Editor в Blender має кілька переваг:

– підсвічування синтаксису: Text Editor використовує підсвічування синтаксису Python для полегшення читання та редагування скриптів;

– інтеграція з Blender: Text Editor працює в межах інтерфейсу Blender, що дозволяє користувачам легко переключатися між текстовим редактором та іншими вкладками Blender;

– автодоповнення: Text Editor має функцію автодоповнення, яка дозволяє користувачам швидко вводити код Python та отримувати пропозиції автодоповнення для функцій та методів;

– збереження та завантаження скриптів: Text Editor дозволяє користувачам зберігати та завантажувати скрипти з легкістю;

– плагіни: Text Editor можна розширити за допомогою плагінів, що дозволяє додавати функціональність до текстового редактора [23-26].

Зручне використання Text Editor в Blender дозволяє використовувати його разом з іншими вікнами відкритими у цій програмі, тобто інтерфейс програми поділяється на частини і розробник має змогу бачити на екрані одразу і текстовий редактор з кодом, і 3D-модель та інші параметри програми Blender, як на рисунку 3.2.

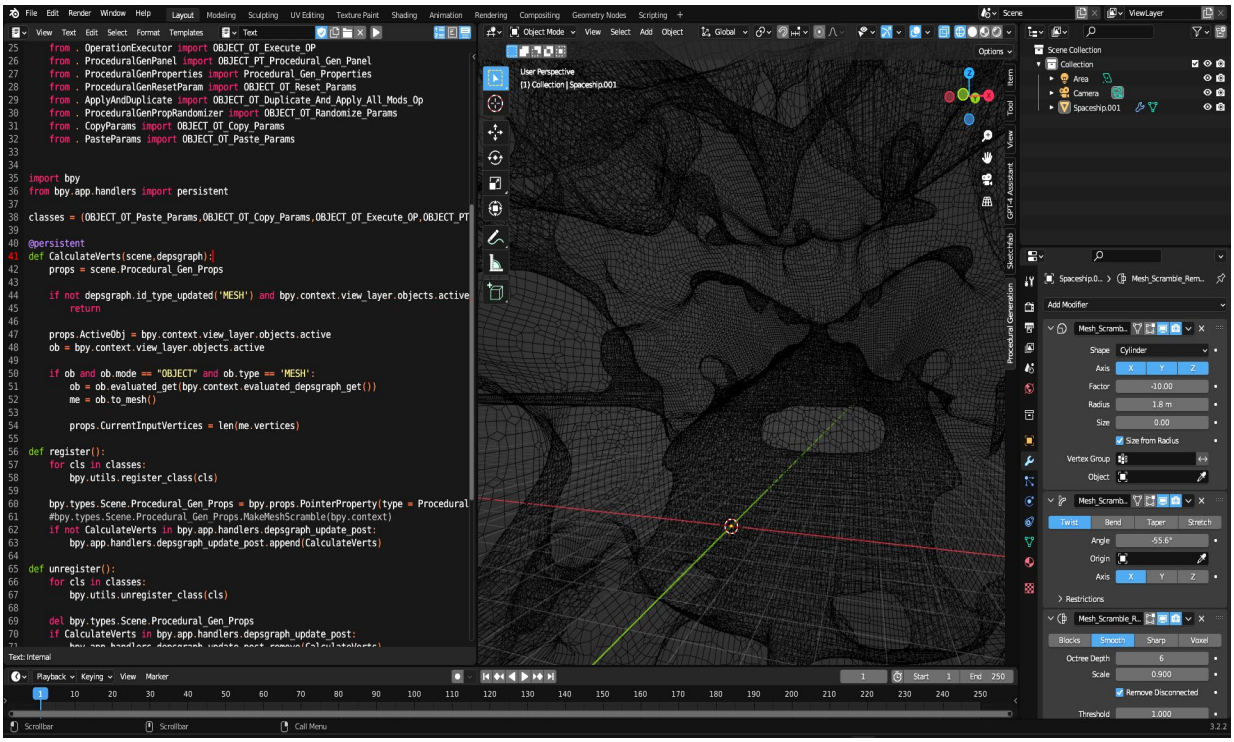

Рисунок 3.2 – Інтерфейс Blender поділений на частини

Узагалі, Text Editor в Blender є потужним інструментом, який дозволяє користувачам швидко писати та виконувати скрипти на мові Python в межах інтерфейсу Blender. Це робить процес створення скриптів та автоматизації задач у Blender більш доступним та ефективним.

3.2 Програмна реалізація

Для реалізації аддону було створено 13 файлів на мові Python, кожен файл відповідає за певний функціонал програми, за методи функції та графічну оболочку аддону. Розглянемо кожен з цих файлів:

– _init_.py – цей файл є головним файлом плагіну, який запускається при завантаженні плагіну в Blender. В цьому файлі виконується ініціалізація плагіну та реєстрація його функцій;

– action.py – цей файл містить клас Action, який відповідає за виконання дії в рамках процедурної генерації. Наприклад, створення нового об’єкту, модифікація наявного об’єкту або додавання нової текстури;

– addModifier.py – цей файл містить функцію AddModifier, яка дозволяє додати новий модифікатор до об’єкта. Модифікатори змінюють форму або зовнішній вигляд об’єкта;

– applyAndDuplicate.py – цей файл містить функцію ApplyAndDuplicate, яка дозволяє застосувати зміни до об’єкту та створити нову копію зі зміненими параметрами;

– copyParams.py – цей файл містить функцію CopyParams, яка дозволяє скопіювати параметри об’єкта з одного об’єкта на інший;

– operation.py – цей файл містить клас Operation, який відповідає за операції в рамках процедурної генерації. Наприклад, об’єднання двох об’єктів, створення нового матеріалу або модифікація текстури;

– operationExecutor.py – цей файл містить клас OperationExecutor, який відповідає за виконання операцій в рамках процедурної генерації. Цей клас дозволяє запускати операції та передавати їм параметри;

– param.py – цей файл містить клас Param, який відповідає за параметри об’єкта в рамках процедурної генерації. Наприклад, розмір об’єкта, його текстура або матеріал;

– pasteParams.py – цей файл містить функцію PasteParams, яка дозволяє вставити скопійовані параметри об’єкта на інший об’єкт;

– proceduralGenPanel.py – цей файл містить клас ProceduralGenPanel, який відповідає за вікно панелі налаштувань плагіну. В цьому вікні користувач може змінювати параметри процедурної генерації;

– proceduralGenProperties.py – цей файл містить клас ProceduralGenProperties, який відповідає за налаштування параметрів процедурної генерації. В цьому класі визначені всі параметри, які можуть бути змінені користувачем;

– proceduralGenPropRandomizer.py – цей файл містить клас ProceduralGenPropRandomizer, який відповідає за функцію випадкового змінення параметрів процедурної генерації. Ця функція дозволяє створювати випадкові об’єкти з різними параметрами;

– proceduralGenResetParam.py – цей файл містить функцію ProceduralGenResetParam, яка дозволяє скинути параметри процедурної генерації до початкових значень [27].

Файл «init.py» є основним файлом, який використовується для запуску плагіну. Він визначає основний клас плагіну і реєструє його в Blender, щоб можна було відкрити панель інструментів Procedural Generation.

Файл «ProceduralGenPanel.py» містить клас, який відповідає за створення панелі інструментів, яка містить усі налаштування для процедурного генерування. У цьому файлі задаються основні параметри, такі як мітка панелі, простір, в якому панель повинна бути розташована, і розташування панелі на екрані.

Файл «ProceduralGenProperties.py» містить класи, які відповідають за зберігання параметрів для процедурного генерування. Вони зберігають параметри, такі як тип операції, налаштування параметрів операції, поточну кількість вершин вхідного об’єкту і максимальну кількість вершин для вхідного об’єкту.

Файл «Operation.py» містить клас, який відповідає за кожну окрему операцію, яку можна використовувати для процедурного генерування. Кожна операція має свої власні параметри, які визначаються за допомогою класу «Param.py».

Файл «OperationExecutor.py» містить клас, який відповідає за виконання операції. Він викликає функції, які генерують 3D-модель за допомогою заданих параметрів.

Що стосується файлу ProceduralGenProperties.py, він містить класи, які відповідають за властивості, які можуть бути змінені в інтерфейсі користувача. Цей файл також містить визначення параметрів за замовчуванням для цих властивостей. Файл ProceduralGenPropRandomizer.py відповідає за генерацію випадкових значень для властивостей, коли користувач натискає кнопку «Randomize».

Проте, на мою думку, найбільш важливим файлом є ProceduralGenPanel.py, оскільки він містить головний клас, який створює панель з інтерфейсом користувача. Цей файл визначає, які кнопки та елементи керування будуть відображені на панелі, і як вони пов'язані між собою. За допомогою цього файлу користувач може взаємодіяти з плагіном та змінювати значення параметрів.

Після написання аддону для процедурного генерування 3D об’єктів, можливі функції та параметри будуть залежати від того, які функції були реалізовані в коді. Наприклад, можуть бути додані функції для створення нових форм, застосування випадкових трансформацій до існуючих об’єктів, генерації текстур, тощо. Крім того, можливості користувача будуть залежати

від того, які властивості були додані в ProceduralGenProperties.py та які елементи керування були розміщені на ProceduralGenPanel.py.

У загальному, важливість кожного файлу у цьому плагіні можна визначити за його внеском у функціонал аддону. Кожен файл має свою роль та функцію, і вони всі разом створюють можливість користувачеві генерувати складні та вражаючі 3D об’єкти.

Розглянемо більш детально кожен файл на наявність в ньому функцій та методів, як на прикладі файла _init_.py

Цей файл містить скрипт, який реєструє (або відміняє реєстрацію) низки класів, які використовуються в плагіні для Blender [28].

Перші декілька рядків цього файлу перевіряють, чи вже були імпортовані деякі класи (за допомогою оператора in locals()), і якщо так, то вони перезавантажуються (reload) за допомогою модуля importlib. Якщо ж класи ще не імпортовані, то вони імпортуються з інших файлів плагіну.

Далі імпортується модуль bpy (Blender Python API), який дає доступ до функціональності Blender через Python.

Функція CalculateVerts збирає відомості про кількість вершин (vertices) в меші (mesh) активного об’єкту (active object) у відповідності з параметрами, встановленими користувачем у панелі ProceduralGenPanel. Якщо кількість вершин змінюється, функція оновлює відповідні значення властивостей Procedural_Gen_Props.

Функція register реєструє у Blender набір класів (classes), визначених в інших файлах плагіну, та створює нову властивість Procedural_Gen_Props для об’єкту сцени. Крім того, вона додає функцію CalculateVerts до списку функцій, що викликаються автоматично в процесі оновлення завантаженого проєкту в Blender.

Функція unregister відміняє реєстрацію класів та видаляє властивість Procedural_Gen_Props для об’єкту сцени. Крім того, вона видаляє функцію CalculateVerts зі списку функцій, які викликаються автоматично в процесі оновлення завантаженого проєкту в Blender.

Розглянемо застосування функцій та методів у файлі Action.py як на рисунку 3.3.

```
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 файла Action.py

Цей код представляє клас Action, який містить методи та атрибути для виконання дій в Blender.

У методі __init__ класу Action визначаються наступні атрибути:

– UIName: назва для елементу інтерфейсу користувача;

– enabeled: логічне значення, що вказує, чи ввімкнена дія;

– liveUpdate: логічне значення, що вказує, чи має бути включена можливість прямого оновлення;

– liveUpdateParam: параметр, який використовується для прямого оновлення;

– dropDownTag: тег, який використовується для випадаючого списку.

У методі AddParam додається новий параметр до списку.

Метод GetParamByName повертає параметр зі списку params за назвою.

Метод AddAction додає нову дію до списку actions у класі Action.

У методі Execute виконуються всі дії зі списку actions. Якщо виконання дій успішне, метод повертає True. Параметр context передається усім діям, що додаються до списку actions, і містить інформацію про контекст Blender. Параметр liveUpdate вказує, чи необхідне пряме оновлення [29].

Розглянемо наступний код файлу AddModifier.by як на рисунку 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 – Огляд функцій файлу AddModifier

Цей код є класом AddModifier, який є підкласом класу Action.

При створенні об’єкту цього класу передаються наступні аргументи:

– modName: ім’я модифікатора, який потрібно додати;

– modType: тип модифікатора, який потрібно додати;

– UIName: назва, яка буде відображатися у вікні інтерфейсу;

– enabeled: назва параметру, який відповідає за включення/виключення модифікатора;

– liveUpdate: логічне значення, яке вказує, чи потрібно виконувати оновлення модифікатора в реальному часі;

– liveUpdateParam: назва параметру, зміна якого викликає оновлення модифікатора в реальному часі.

Цей клас має метод Execute, який додає модифікатор до об’єкта в контексті context. Для цього використовуються наступні кроки:

– отримується активний об’єкт у поточному контексті context;

– створюється модифікатор з ім’ям modName і типом modType для цього об’єкта, якщо він ще не існує;

– встановлюється параметр show_viewport, show_render модифікатора на значення, що відповідає стану параметру enabeled у props, тобто якщо параметр enabeled у props дорівнює True, то show_viewport та show_render буде встановлено в True;

– параметр show_in_editmode модифікатора встановлюється в False, тобто він не буде відображатися під час редагування об’єкта в режимі редагування;

– для кожного параметру param зі списку params встановлюється значення відповідного параметру в mod, що збігається з VarName у props.

Метод Execute повертає значення True, якщо операція успішно виконана. Розглянемо код файлу CopyParams та які функції він використовує.

Лістинг 3.1 Частина коду файлу CopyP:

```
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’}
```

Цей код містить клас OBJECT_OT_Copy_Params, який є оператором Blender. Цей оператор призначений для копіювання значень налаштувань, які можуть бути використані для додаткових ефектів генерації об’єктів у 3D-сцені.

У функції execute оператора виконується наступне:

– отримується посилання на поточний сценарій та налаштування Procedural_Gen_Props;

– створюється об’єкт DataSingleton, що містить список дій для генерації об’єктів;

– проходиться циклом по кожній дії зі списку ActiveOperation;

– збираються значення налаштувань, які дієві для даної дії, та записуються у змінну outs;

– далі перевіряється, чи налаштування підтримують обмін (Share), і якщо так, то значення додається до змінної outs;

– якщо налаштування є вектором, то значення розділяються комою та обгортаються у дужки;

– значення outs кодуються у формат base64 та стискуються у форматі zlib;

– остаточно, закодоване значення записується у буфер обміну.

Цей оператор може бути корисним для копіювання параметрів об’єктів у випадку, коли потрібно згенерувати багато об’єктів з однаковими налаштуваннями.

Наступним чином розглянемо код файлу Operation.py (рис. 3.5).

```
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

Цей код містить клас Operation, який використовується для опису різноманітних операцій, які можуть бути виконані з об’єктами в Blender.

У конструкторі класу Operation встановлюються параметри операції: назва, необхідність вхідного об’єкту (InputObjNeeded), максимальна кількість вершин (MaxVert), назва випадаючого списку (dropDownName), якщо такий є.

Метод AddAction додає дію до списку дій операції.

Метод Execute виконує всі дії операції в контексті поточної сцени Blender. Якщо задано назву випадаючого списку, метод перевіряє, чи збігається вибір користувача зі значенням властивості dropDownName. Якщо ні, то дія не виконується. Метод повертає True, якщо операція успішно виконана.

Метод AutoUpdate виконує автоматичне оновлення операції в залежності від змін властивостей. Якщо задано назву випадаючого списку, метод перевіряє, чи збігається вибір користувача зі значенням властивості dropDownName. Якщо ні, то оновлення не виконується. Якщо властивість liveUpdate дорівнює True і змінилися значення властивості liveUpdateParam, то виконується відповідна дія. Метод повертає True, якщо оновлення успішно виконано [30].

Розглянемо код файлу ProceduralGenPanel.py на наявність в ньому методів та процедур, саме цей файл є одним з найголовніших при ініціації цього аддону, він генерує всю графічну панель та каркас аддону (рис. 3.6).

Цей код представляє собою реалізацію панелі інтерфейсу користувача для застосунку, який був розроблений у Blender.

Клас OBJECT_PT_Procedural_Gen_Panel є нащадком класу bpy.types.Panel і містить різні методи для створення та керування вмістом панелі.

У методі draw (self, context) виконується створення вмісту панелі. Зокрема, виконується вивід назви та іконки панелі, а також кнопок та інших елементів інтерфейсу.

```
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 представляє собою реалізацію панелі інтерфейсу користувача для застосунку, який був розроблений у Blender.

Клас OBJECT_PT_Procedural_Gen_Panel є нащадком класу bpy.types.Panel і містить різні методи для створення та керування вмістом панелі.

У методі draw(self, context) виконується створення вмісту панелі. Зокрема, виконується вивід назви та іконки панелі, а також кнопок та інших елементів інтерфейсу.

У першому рядку методу draw (self, context) відбувається отримання поточного стану сцени, де виконується процедурна генерація моделей, та отримання даних про властивості генерації з об’єкту Procedural_Gen_Props за допомогою змінної props.

Далі виконується перевірка наявності ініціалізації даних про операції процедурної генерації. Якщо ці дані не ініціалізовані, то виконується метод CreateOperations (context), який створює список операцій та додає до нього відповідні дії.

Далі виконується створення елементів інтерфейсу. Спочатку створюється кнопка «Reset Parameters» для скидання параметрів генерації. Далі відбувається створення блоку з назвою «Procedural Generation» та вибір методу генерації за допомогою списку випадаючих елементів.

Якщо активна операція має список випадаючих елементів, то відображається відповідний елемент інтерфейсу. Далі відображаються параметри дій, що входять до складу активної операції, за допомогою відповідних елементів інтерфейсу, таких як row.prop та box3.column.

Огляд параметрів у файлі:

– layout = self.layout – зберігає зручний доступ до створення розмітки на цьому панелі;

– scene = context.scene – зберігає посилання на сцену, на якій відображається вміст редактора;

– props = scene.Procedural_Gen_Props – зберігає посилання на властивості об’єкта Procedural_Gen_Props, який містить всі параметри, що використовуються для створення процедурних об’єктів;

– data = DataSingleton() – створює об’єкт DataSingleton, який містить поточний стан генерування об’єктів;

– if not data.IsInitialized: – перевіряє, чи ініціалізований об’єкт DataSingleton;

– props.CreateOperations(context) – якщо об’єкт не ініціалізований, то створює нові об’єкти Operation за допомогою методу CreateOperations властивості props;

– row = layout.row() – створює новий рядок в розмітці;

– row.operator(‘object.procedural_gen_reset_parameters’) – додає кнопку, яка дозволяє скинути всі параметри генерації;

– box = layout.box() – створює нову коробку для групування елементів в розмітці;

– row = box.row() – створює новий рядок в коробці;

– row.label(text="Generation Method:") – додає текстову мітку до рядка з назвою "Generation Method:";

– row.prop(props, "OperationsType",text = "") – додає список випадаючих елементів зі списком методів генерації, вказаних у властивості OperationsType;

– if data.ActiveOperation != None: – перевіряє, чи встановлено активний метод генерації.

Далі в коді маємо цикл for a in data.ActiveOperation.actions:, який проходиться по всіх екшенах(діях) вибраної операції.

Далі маємо блок коду з вкладеним у нього if та for операторами, які відповідають за генерацію відповідних полів для кожного екшена (дії).

У цьому блоку коду перевіряємо чи необхідно згенерувати dropdown поле для відповідної дії за допомогою умови if data.ActiveOperation.dropDownName. Якщо так, то генеруємо dropdown поле за допомогою row.prop(props, data.ActiveOperation.dropDownName).

Далі відображаємо UI елементи для кожного параметру дії за допомогою наступного циклу for p in a.params:. У цьому циклі генеруємо відповідні UI елементи для кожного параметру дії за допомогою row.prop(props, row.prop(props, data.ActiveOperation.dropDownName) p.VarName, slider=p.slider,expand=p.expand,toggle=p.toggle). Умова if p.Column: відповідає за те, щоб параметри розміщувалися в стовпчик, якщо це необхідно. Умова elif p.UseLable: відповідає за те, щоб відображати лейбли для параметрів.

Умова elif p.ShowInUI: відповідає за те, щоб відображати параметри, які не належать до жодного з попередніх категорій.

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

Останнім кроком у цьому коді є генерація кнопок у нижній частині панелі. Таким чином було розібрано основний функціональний код аддону який розширюю можливості звичайних художників [31].

3.3 Інструкція використання розробленого аддону

Перед використанням аддону треба встановити 3D-пакет Blender однієї з розбіжності версій. Застосунок було написано для версій Blender з 3.0 до 3.3. Після цього треба завантажити архів з розробленим аддоном, посилання на нього буде разом з кваліфікаційною роботою. Після виконаних вище описаних дій треба підключити аддон у налаштуваннях Blender (рис. 3.7).

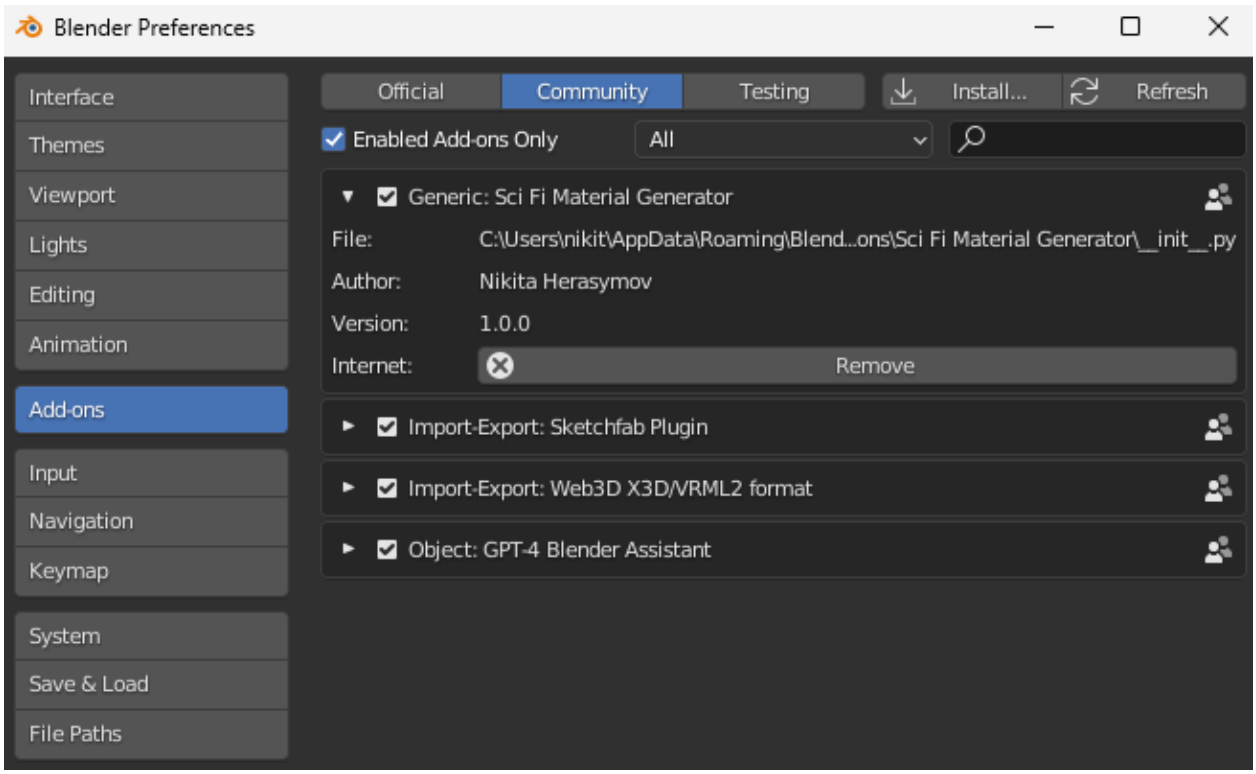

Рисунок 3.7 – Підключення аддону

Після підключення аддону повинні зайти у його меню, для цього клавішою «N» відкриваємо панель усіх активних на даний момент аддонів та вибираємо розроблений «Procedural Generation» (рис. 3.8).

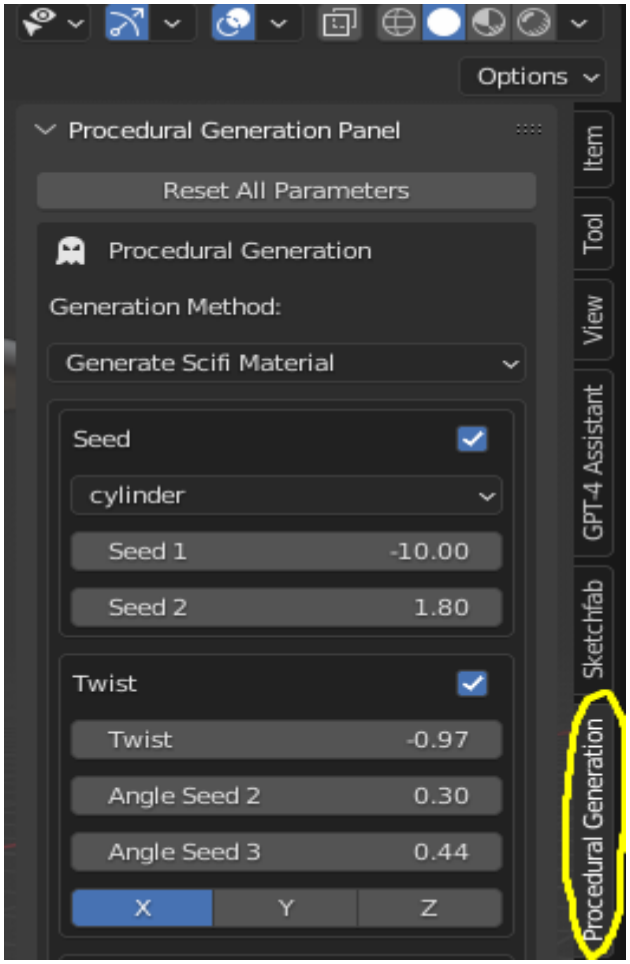

Рисунок 3.8 – Активний аддон Procedural Generation

Після активації аддону бачимо головне вікно аддону та вибір 3 активних позицій меню, які були розглянуті на початку. Щоб створити перший каркас космічного корабля для якого потім буде застосована текстура процедурної генерації треба вибрати у меню позицію «Generate Spaceship Base», після цього можна змінити власноруч параметри, або вибрати позицію «Randomize All Parameters», яка більш рекомендована для нового користувача, який ще не має досвіду у 3D-пакеті Blender. Після вибору встановлення випадкових параметрів треба натиснути на кнопку «Execute Operation», яка саме ініціалізує

процесс створення 3D-моделі космічного корабля, результат вищеперерахованих дій показано на рисунку 3.9.

Рисунок 3.9 – Сгенерований процедурною генерацією космічний корабель

Після генерації 3D-об’єкту довільної фігури – схожий процес буде проведений з матеріалом цього космічного корабля, тобто текстура поверх цього 3D-об’єкта буде теж згенерована за допомогою процедурної генерації.

Для цього потрібно вибрати нову позицію в головному меню – а саме «Generate Sci-fi Material», при виборі цього меню відкривається великий вибір параметрів процедурної генерації, Accent Color, який розглядали раніше потрібен для встановлення кольору який буде відображатися більш всього на частинках текстури, тобто акцентний кольор. Light Color встановлюємо для частинок які мають джерела світла. Усі інші налаштування потрібні для більш тонкого налаштування, наприклад як Shape Scale, або Light Scale, використання цих параметрів буде розглянуто у підрозділі 3.4. У даному випадку треба зробити кроки як для меню «Generate Spaceship Base», тобто

знову вибрати «Randomize All Parameters», та натиснути Execute Operation, результат буде як на рисунку 3.10.

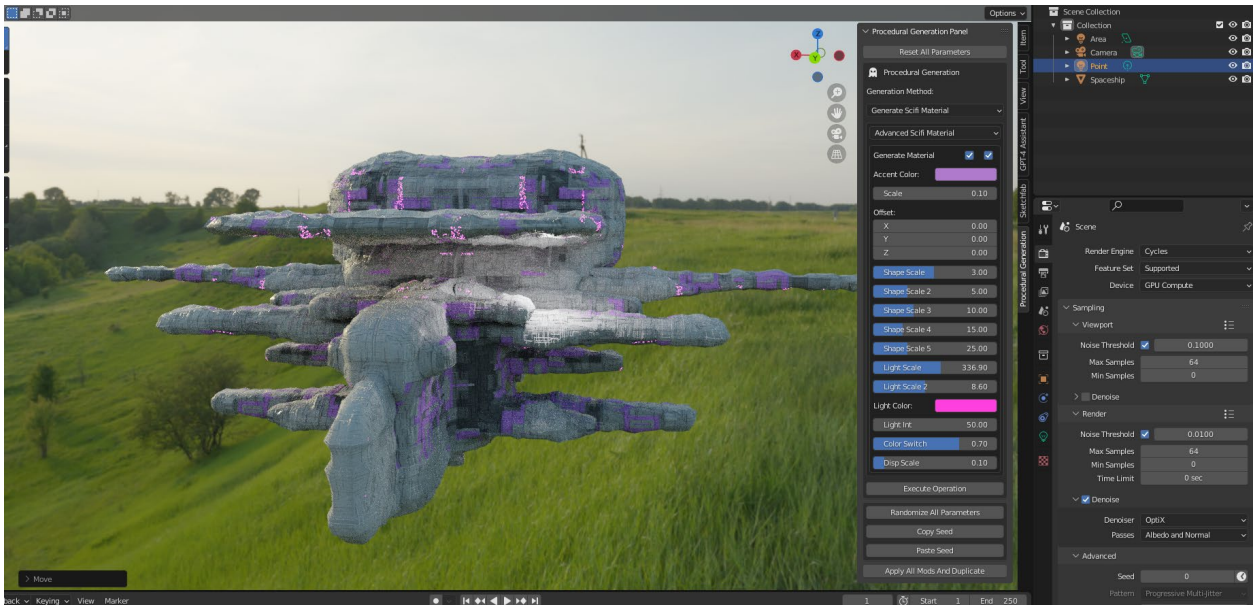

Рисунок 3.10 – Результат фінальної процедурної генерації

Після процесу рендерингу можна зберегти отриманий результат, як рисунок у різних відповідних форматах.

3.4 Тестування ітерацій процедурної генерації

Для більшого розуміння процедурної генерації, розглянемо деякі її ітерації в процесі рендерингу, з використанням різних параметрів як у меню «Generate Sci-fi Material» так і «Generate Spaceship Base», параметри процедурної генерації будуть змінені як в ручну так і з використанням опціїі Randomize All Parameters, та зі зміною параметрів Sharp та Smooth, приклади наведені нижче, (рис. 3.11) і т.д. Встановлення параметру відзеркалення дає можливість продовжити випуклість текстури не тільки в одному місці, а й по тим координатам які будуть вказані для відзеркалення.

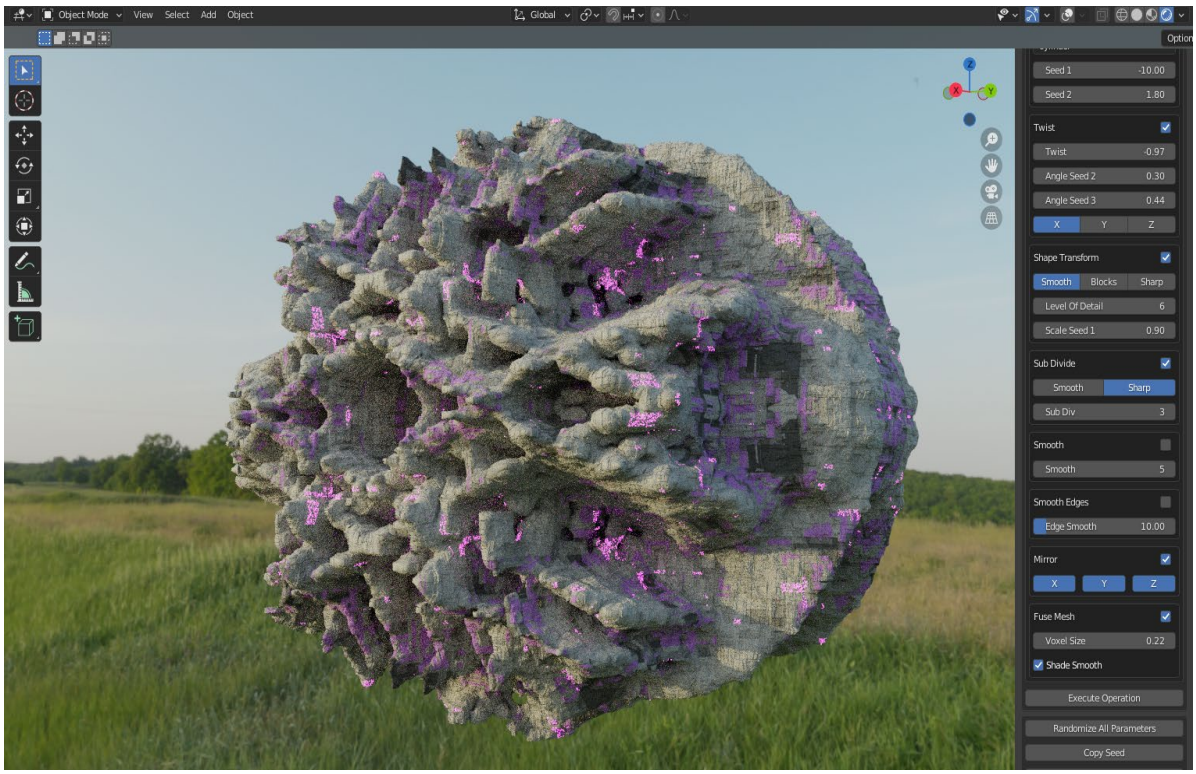

Рисунок 3.11 – Встановлення параметрів Sharp з відзеркаленням

Після встановлення у розробленому програмному застосунку параметрів Sharp з відзеркаленням можна побачити досить помітні гострі та випуклі грані, які випирають з плоскої текстури моделі, таким чином цим параметром можна задавати загальний розмір випуклості текстури у певній координаті, *X*, *Y* або *Z*. Задача кожного параметру була розроблена для максимально унікального створення кожної наступної моделі процедурної генерації. Навіть при виставленні однакових параметрівк кожна наступна згенерована модель буде відрізнятися однією з деталей, в цьому і полягає ідея процедурної генерації, наступні рисунки 3.12 – 3.14 дозволяють зрозуміти те наскільки кожен параметр впливає на остаточний результат кожної нової згенерованої моделі за допомогою процедурної генерації.

Після огляду встановлення різних параметрів процедурної генерації та виводу їх результату, можемо зробити підсумок, що розробка на мові Python для 3D-пакету Blender аддону, який використовує процедурну генерацію матеріалів, текстур чи об’єктів є актуальною темою та надає 3D-дизайнерам більш ефективну, зручну та швидку роботу з 3D-сценами.

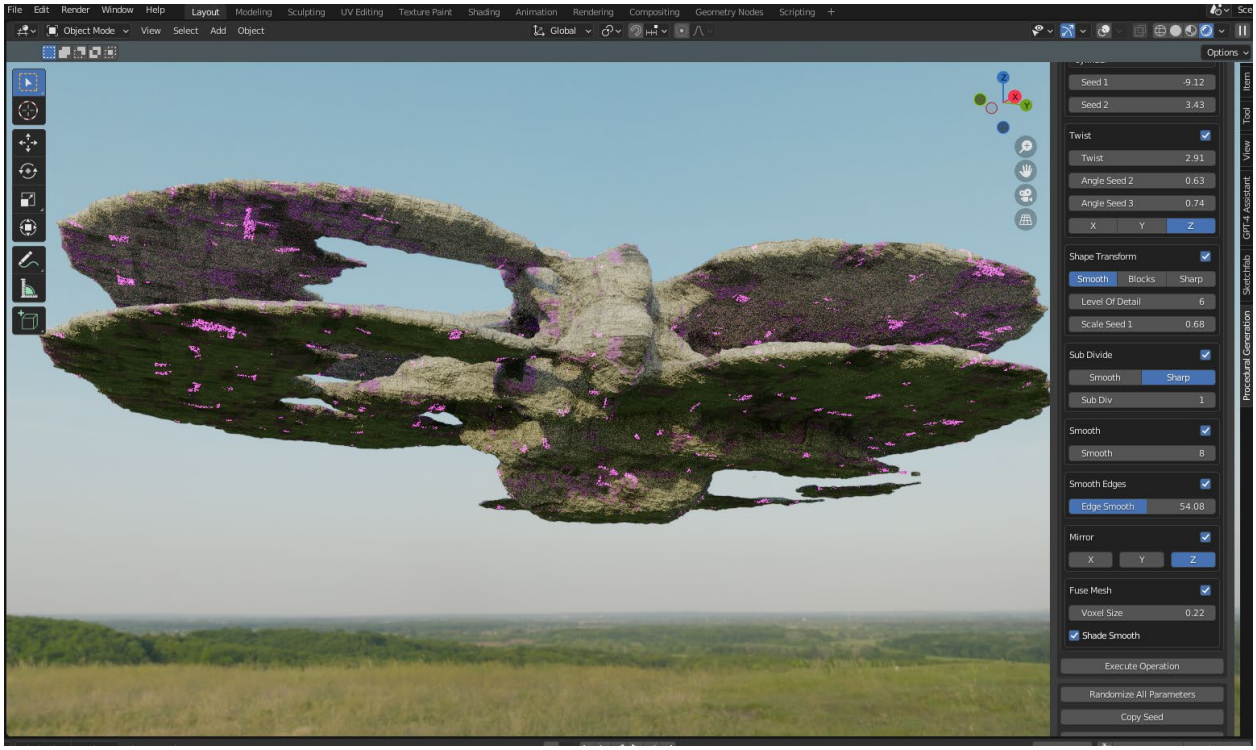

Рисунок 3.12 – Встановлення довільних параметрів Sharp з віддзеркаленням тільки по координаті *Z*

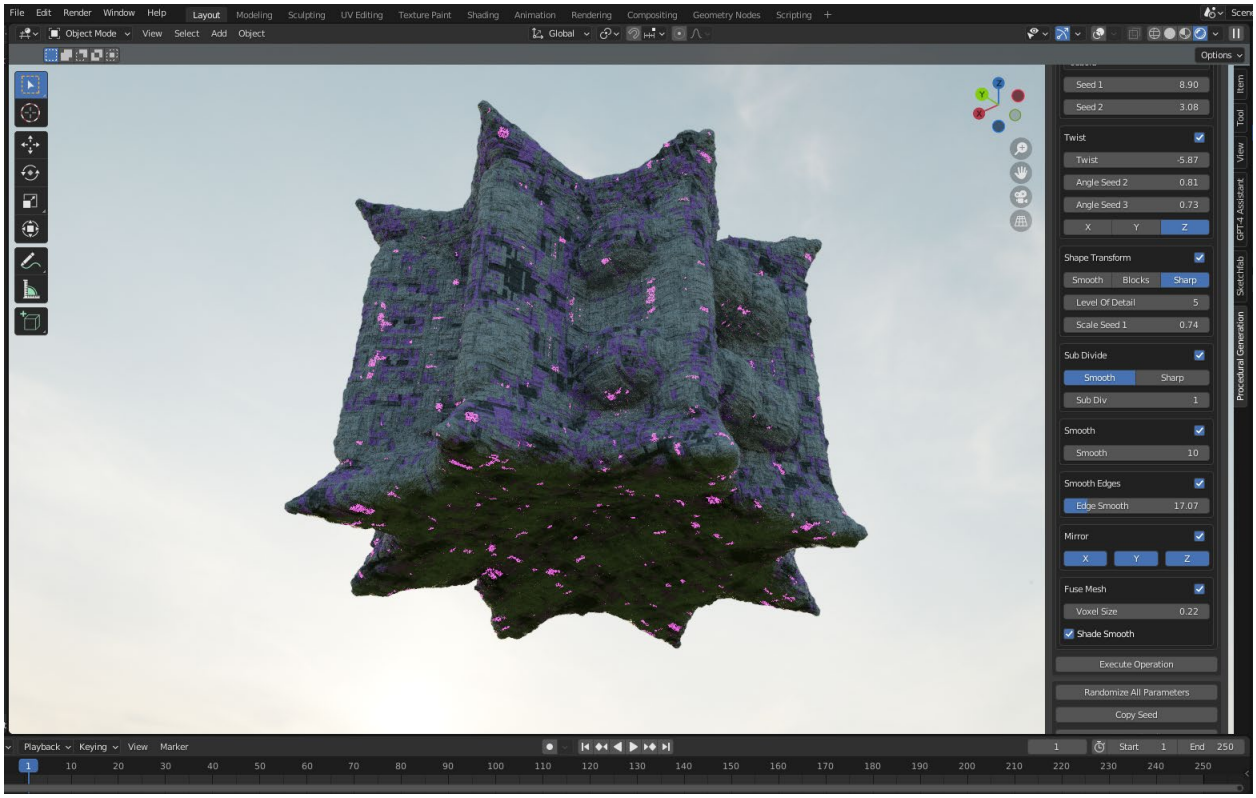

Рисунок 3.13 – Параметр Randomize All Parameters та Smoth

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

Встановлення навіть одного зміненого параметру дає можливість 3D-розробнику створити унікальну модель використовуючи застосунок для побудови моделей процедурної генерації. Аддон дозволяє користувачам створювати складні 3D-моделі та анімації шляхом визначення параметрів та правил, за якими будуть генеруватись об’єкти.

ВИСНОВОК

У рамках кваліфікаційної роботи було розроблено аддон на мові програмування Python для 3D-пакету Blender, який використовує технологію процедурної генерації та розширює можливості художників, які використовують цей 3D-пакет.

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.

Аддон дозволяє користувачам створювати складні 3D-моделі та анімації шляхом визначення параметрів та правил, за якими будуть генеруватись об’єкти. Це дозволяє художникам більш ефективно використовувати свій час та створювати більш складні та цікаві композиції.

Розроблений аддон був успішно інтегрований в Blender та пройшов відповідні тести на працездатність.

Було розглянуто теми, що стосуються 3D-пакету Blender, мови програмування Python, було проаналізовано методи та функції API Blender. Також було сформульовано математичні формули модифікаторів Subdivision Surface, Bevel та інших, які використовуються при створенні 3D-об’єктів за допомогою процедурної генерації.

Інструкція користувача та результати роботи ітерацій кожного сгенерованого об’єкта було додану у кваліфікаційну роботу.

В результаті цієї роботи, було досягнуто поставленої мети та ідеї, розроблений аддон дозволяє значно розширити можливості художників, які працюють з Blender.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

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

2. Researchgate. URL: https://www.researchgate.net/figure/The-rst-Head-Mounted-Display-proposed-by-Sutherland-1968_fig3_234836152 (дата звернення 08.04.2023).

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/%D0%9F%D1%80%D0%BE%%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%B0_%D0%B3%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%86%D1%96%D1%8F (дата звернення 12.04.2023).

6. Blender Manual. URL: https://docs.blender.org/manual/uk/dev/advanced/scripting/introduction.html (дата звернення 12.04.2023).

7. Sites Google. URL: https://sites.google.com/site/modeluvanna3d/home/aki-najvidomisi-sposobi-stvorenna-3d-modeli (дата звернення 13.04.2023).

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 16.04.2023).

13. Addtive. URL: https://addtive.com.ua/ua/shcho_take_3d_modelyuvannya/ (дата звернення 17.04.2023).

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. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Tools for Fast Metric Data Search in Structural Methods for Image Classification. *IEEE Access*, *10*, 124738-124746.

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. Гороховатський, В. О., & Творошенко, І. С. (2022). Аналіз багатовимірних даних за описом у формі множини компонент.

19. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Tools for Fast Metric Data Search in Structural Methods for Image Classification. *IEEE Access*, *10*, 124738-124746.

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 23.04.2023).

22. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Tools for Fast Metric Data Search in Structural Methods for Image Classification. *IEEE Access*, *10*, 124738-124746.

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. Gadetska, S., Gorokhovatskyi, V., Stiahlyk, N., & Vlasenko, N. (2022). Aggregate Parametric Representation of Image Structural Description in Statistical Classification Methods.

26. Gorokhovatskyi, V. O. (2021). Statistical data analysis tools in image classification methods based on the description as a set of binary descriptors of key points.

27. 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).

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. Ibrahim, D. Y., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Cluster representation of the structural description of images for effective classification.

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.