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)

МОДЕЛЮВАННЯ ТА РЕАЛІЗАЦІЯ ТРИВИМІРНОЇ АНІМАЦІЇ ІЗ ВПРОВАДЖЕННЯМ АВТОМАТИЗАЦІЇ У СЕРЕДОВИЩІ BLENDER НА ПРИКЛАДІ ЗАМКУ ГОҐВОРТС
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-2
Зінченко А. Я.
(прізвище, ініціали)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Інформатика
(повна назва освітньої програми)
Керівник асист. Кириченко І. Ю.
(посада, прізвище, ініціали)

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Зінченко Анастасії Ярославівні
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання та реалізація тривимірної анімації із впровадженням автоматизації у середовищі Blender на прикладі замку Гоґвортс

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 30 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, офіційна документація Blender, публікації щодо алгоритмів процедурної генерації, зображення та візуальні матеріали замку Гоґвортс, зібрані з відкритих джерел, матеріали онлайн-ресурсів, присвячених дизайну тривимірних сцен.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз варіантів архітектури Гоґвортсу та вибір версії замку для відтворення.
2. Огляд можливостей Blender для створення складних тривимірних сцен.
3. Вивчення принципів процедурної генерації моделей з використанням Python.
4. Реалізація Blender-адону для автоматизованого створення дерев.
5. Побудова сцени з урахуванням ландшафту, освітлення та створення анімації.
6. Візуалізація результатів та оцінка можливостей подальшого використання.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) 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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів	15.04.25-20.04.25	
5	Розробка методу	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____асист. Кириченко І. Ю._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 73 с., 31 рис., 31 джерело.

BLENDER, PYTHON, ПРОЦЕДУРНЕ МОДЕЛЮВАННЯ, ТРИВИМІРНЕ МОДЕЛЮВАННЯ, ЯЛИНА, АДОН, ГЕНЕРАЦІЯ, КРИВА БЕЗЬЄ, ГОҐВОРТС, ВІЗУАЛІЗАЦІЯ.

Об’єктом роботи є тривимірна сцена, що поєднує архітектурну модель замку Гоґвортс із автоматично згенерованими реалістичними ялинами.

Метою роботи є створення Blender-адону для генерації ялин на основі кривих Безьє, математичних залежностей і керованої варіативності, а також побудова сцени для подальшого рендеру й створення анімованого відео.

У ході роботи було реалізовано процедурну генерацію стовбура, гілок і площин-хвої мовою Python, впроваджено параметризацію, оптимізацію геометрії та зручний інтерфейс у Blender. Додатково вручну змодельовано замок з використанням модифікаторів, текстур, освітлення та інструментів анімації.

У результаті створено інструмент для генерації ялин, побудовано тривимірну сцену та сформовано фінальне відео, яке демонструє поєднання автоматизованого та ручного моделювання в єдиній композиції.

BLENDER, PYTHON, PROCEDURAL MODELING, 3D MODELING, PINE TREE, ADD-ON, GENERATION, BÉZIER CURVE, HOGWARTS, VISUALIZATION.

The object of the work is a three-dimensional scene combining an architectural model of the Hogwarts castle with procedurally generated realistic pine trees.

The aim of the work is to create a Blender add-on for generating pine trees based on Bézier curves, mathematical dependencies, and controlled variability, as well as to construct a scene for further rendering and the creation of an animated video.

The work includes the procedural generation of a trunk, branches, and needle-like planes using Python, along with parameterization, geometry optimization, and a user-friendly interface in Blender. Additionally, the castle was manually modeled using modifiers, textures, lighting, and animation tools.

As a result, a pine tree generator was developed, a full 3D scene was assembled, and a final video was produced to demonstrate the integration of automated and manual modeling in a single composition.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Адон – додатковий модуль, що розширює функціональність програми

Рендеринг – процес візуалізації сцени; отримання фінального зображення з тривимірної моделі із застосуванням матеріалів, освітлення й камер

Рендер – фінальне зображення або відео, отримане в результаті процесу рендерингу

API – Application Programming Interface (інтерфейс прикладного програмування)

API – набір інструментів, які дозволяють керувати програмними об’єктами, викликати функції та взаємодіяти з внутрішніми компонентами середовища

HDRI – High Dynamic Range Image (зображення з високим динамічним діапазоном)

HDRI – формат зображення, що зберігає розширену інформацію про освітлення, використовується для точного відтворення світла й тіней у сцені

ВСТУП

Тривимірне моделювання вже давно перестало бути винятковою сферою великих студій. Завдяки програмам з відкритим кодом, як-от Blender, воно стало доступним кожному – від художника до розробника. Це особливо відчутно там, де ручне моделювання поєднується з автоматизованими методами створення однотипних об’єктів, як-от дерева чи інші природні елементи.

Цей проєкт об’єднує два підходи: з одного боку – детальне ручне моделювання архітектурного об’єкта (замку Гоґвортс), а з іншого – розробка адону на Python для генерації ялин, які автоматично розміщуються в сцені. Уся робота виконувалась у середовищі Blender: тут створювалися моделі, налаштовувалися матеріали, розставлялося світло і відбувався рендер. Завдяки цьому не було потреби переходити між кількома програмами – усе зручно зосереджено в одному місці.

Замок моделювався з простих форм і поступово ускладнювався за допомогою модифікаторів. Після цього було сформовано ландшафт: рельєф місцевості, підвищення, озеро – усе, що створює логічне оточення навколо будівлі. Для поверхні використовувалися реалістичні матеріали, зібрані на основі фізично достовірних текстур, що підкреслюють масштабність сцени. Кожен об’єкт доводився до фінального вигляду вручну, після чого сцена доповнювалась автоматично створеними ялинами.

У підсумку було отримано повністю зібрану сцену: архітектуру, рельєф, дерева, освітлення, анімацію. Усе це об’єднано в завершену композицію, яка природно виглядає і готова до використання для створення фінального відео.

Актуальність проєкту зумовлена кількома факторами. По-перше, замок Гоґвортс – візуальний символ, який давно вийшов за межі книжкової чи кінематографічної історії. Його впізнають у грі, на постері, у відео. І навіть зараз інтерес не згасає: очікується запуск нового серіалу, а це майже напевно означає хвилю уваги до фанатського контенту, візуалізацій і реконструкцій

знайомих сцен. Створення власної сцени з архітектурою замку є не лише творчим завданням, а й способом долучитися до цього візуального світу.

По-друге, важливим є і технічний аспект. Побудова великих тривимірних сцен потребує чимало часу, особливо коли йдеться про оточення: дерева, рельєф, природні деталі. І саме тут автоматизоване створення об’єктів, наприклад, ялин, дає велику перевагу. Замість тривалого ручного моделювання – математична логіка, параметри, які можна змінювати в кілька кліків, і результат, що адаптується до будь-якої сцени. Це зручно не лише у творчих проєктах, а й у реальних завданнях геймдизайну, архітектурної візуалізації чи анімації.

По-третє, об’єднання процедурного та ручного підходів відкриває більше можливостей. Архітектура, побудована вручну, і ялини, згенеровані за допомогою Python-адону, не конкурують, а доповнюють один одного. Так формується сцена, яка виглядає завершеною: з деталями, масштабами, настроєм. Її можна рендерити, анімувати, переробляти під інші задачі або використовувати як основу для нових проєктів.

1 ОГЛЯД ОСНОВНИХ ПІДХОДІВ ДО СТВОРЕННЯ ТРИВИМІРНИХ СЦЕН

1.1 Огляд програмних засобів для тривимірного моделювання

Тривимірне моделювання – це процес побудови об’єктів, які мають не лише висоту та ширину, а й глибину. Такий підхід дозволяє отримати результат, який наближений до реального світу або ж зовсім від нього відходить – усе залежить від мети. Тривимірні моделі використовуються у фільмах, комп’ютерних іграх, архітектурних проєктах, а також у сфері науки, освіти та віртуальної реальності [1, 2]. І з кожним роком ця галузь стає популярнішою. Це є цілком логічним, адже за допомогою цифрових моделей можна передати те, що важко або неможливо зафіксувати на камеру.

На сьогодні існує багато програм, які дозволяють працювати з тривимірними об’єктами. Наприклад, Autodesk Maya відома у сфері кіноанімації, але складна у вивченні. Autodesk 3ds Max часто використовують архітектори. ZBrush обирають для скульптингу, коли потрібно створити детальні фігури з м’якими або фантазійними формами. Cinema 4D зручна для motion-дизайну, тоді як SketchUp орієнтована на створення простих моделей для концептів. Серед мобільних додатків можна виділити Nomad Sculpt – він простий, але має обмеження, якщо мова йде про складні сцени, анімацію чи освітлення.

Серед цих програм Blender займає окреме місце. Він є універсальним, адже у ньому можна створити об’єкт, нанести текстуру, додати анімацію, налаштувати фізику, світло, зробити рендер – і все це в одному середовищі. Ця програма дозволяє працювати над проєктом без потреби відкривати ще декілька інших застосунків.

Окремо варто згадати інтерфейс. Blender не виглядає перевантаженим, навіть якщо в ньому багато функцій. Все розміщено логічно, а дії можна

виконувати за допомогою гарячих клавіш, що є зручним у використанні. Панелі, вікна та їхній вигляд також легко можна налаштувати під себе.

На рисунку 1.1 наведено приклад інтерфейсу Blender.

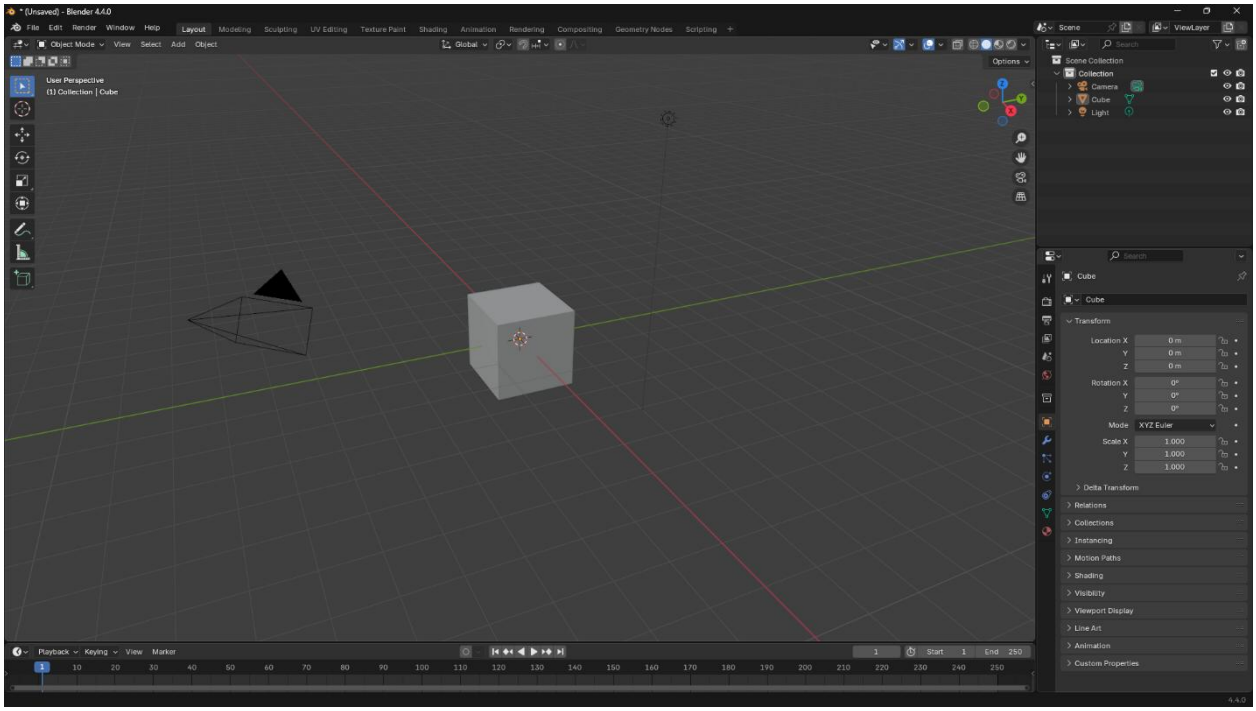

Рисунок 1.1 – Інтерфейс Blender 4.4.0

Ще один плюс – безкоштовна ліцензія. Blender відкритий, його можна використовувати в навчанні, для комерційних проєктів, для досліджень або хобі. Програма постійно оновлюється: розробники додають нові можливості, виправляють помилки, а ще користувачі зі всього світу створюють адони – додаткові інструменти під свої потреби.

Окремо варто згадати анімаційний фільм «Потік. Останній кіт на Землі», створений у 2024 році. Він був зроблений у Blender з дуже скромним бюджетом і отримав «Оскар». Це ще раз доводить, що навіть із безкоштовною програмою можна досягти рівня, який визнали на світовому рівні.

Також у Blender дуже потужна спільнота. В інтернеті є безліч відео, статей, моделей, HDRI і прикладів. Це все допомагає швидше навчатися і працювати. Якщо не вистачає інструментарію, то завжди можна знайти адон, який це додає.

Отже, з усього розмаїття програм для тривимірного моделювання саме Blender був обраний як найкращий варіант. Він доступний, сучасний, багатофункціональний і зручний. А ще – перевірений не тільки користувачами, а й часом.

1.2 Інтеграція мови програмування Python у середовище Blender

Blender – це не просто програма для тривимірного моделювання. У ньому є також можливість автоматизувати багато дій за допомогою мови програмування Python. Завдяки цьому користувач може не обмежуватися лише наявним функціоналом. У Blender можна написати скрипт, створити свій інструмент, змінити логіку поведінки або додати нову панель керування, що відкриває широкі можливості для розширення вже наявних можливостей.

Усе це створюється за допомогою так званих адонів – спеціальних додатків, які вбудовуються у середовище Blender. Вони допомагають автоматизувати повторювані завдання, створювати нові функції, додавати кнопки, слайдери, списки, що значно полегшує та пришвидшує роботу. Це особливо зручно, коли потрібно виконувати технічно складні або рутинні дії, які вручну вимагають багато часу.

Завдяки мові Python відкривається доступ до внутрішнього API програми. Це означає, що можна взяти будь-який об’єкт у сцені – модель, текстуру, анімацію, освітлення, камеру, і змінити його параметри через код. Наприклад, можна створити новий об’єкт, задати йому розміри, перемістити на потрібну координату або застосувати до нього матеріал – і все це без натискання жодної кнопки вручну.

Окремо варто зазначити те, що за допомогою Python можна створювати свої графічні інтерфейси. Це можуть бути вікна з полями для вводу, списки або вкладки – і виглядатимуть вони так, ніби їх розробили разом з програмою.

Завдяки цьому користувач взаємодіє з інструментами без необхідності відкривати код.

Отже, робота з Blender не обмежується стандартними можливостями. Завдяки інтеграції мови Python можна адаптувати програму під власні задачі, що дозволяє створювати прості та зручні адони, які не просто доповнюють вже наявний функціонал, а й точно вирішують конкретні завдання.

1.3 Переваги використання мови програмування Python

Мова програмування Python є основною для створення адонів у Blender. Вона проста у використанні та зрозуміла навіть тим, хто не має великих знань у програмуванні, але водночас її достатньо, щоб реалізувати складну логіку. Саме тому її було обрано як засіб для розширення функціональності Blender.

Python також має зрозумілу та лаконічну структуру, що забезпечує логічну послідовність у написанні коду. Це значно знижує поріг входу для новачків. Тому навіть ті, хто не є професійним програмістом, можуть швидко почати працювати з нею. Для Blender це важливо, адже його користувачі – це не лише досвідчені розробники, але й дизайнери, художники та початківці, які працюють у сфері візуалізації.

Ще однією перевагою Python є його підтримка великою кількістю користувачів. Ця мова програмування постійно оновлюється, а також має стабільну документацію та активну спільноту. Саме завдяки цьому в інтернеті можна знайти велику кількість навчальних матеріалів, прикладів готових програм, пояснень і туторіалів. Це можна використовувати як для загального вивчення мови, так і для роботи з нею у Blender.

Крім того, Python широко використовується в галузі машинного навчання, де особливо цінується за доступність синтаксису та потужні бібліотеки. Навіть ті посібники, що формально присвячені алгоритмам або аналізу даних, часто містять корисні пояснення щодо побудови програм

мовою Python. Саме тому такі джерела теж можна залучати як додаткову підтримку при роботі з кодом або структурою даних [3–8].

Також Python у Blender не потребує окремого встановлення – він уже інтегрований у середовище. Це зручно, адже будь-який скрипт можна запускати одразу після відкриття програми. Окрім того, його використання не обмежується якоюсь конкретною операційною системою, адже він працює однаково стабільно як на Windows, так і на macOS чи Linux.

Отже, Python є не лише зручним у використанні сам по собі, але й як інструмент, який можна використовувати для роботи з Blender. Саме завдяки йому можна створити необхідні адони, які будуть відповідати власним потребам. Саме поєднання простоти, універсальності й підтримки з боку спільноти робить Python важливим інструментом для автоматизації у Blender.

1.4 Аналіз адону Sapling Tree Gen та варіантів його удосконалення

У Blender є вбудовані інструменти, які дозволяють створювати дерева буквально в кілька кліків. Один із них – адон Sapling Tree Gen [9]. Його часто використовують для генерації дерев у пейзажах, лісах або будь-яких сценах на відкритому повітрі. У цьому адоні можна змінювати висоту дерева, форму його стовбура, густоту крони, кількість гілок, їх нахили, розгалуження і не тільки це. Він підтримує базові налаштування для широкого кола типів дерев, включно з листяними та хвойними видами, і дає змогу отримати детальну структуру вже на перших етапах роботи.

З одного боку, це добре, адже в результаті можна отримати найрізноманітніші результати. Але водночас ці можливості мають мінуси. Для того, щоб зробити одне конкретне дерево, треба використовувати та змінювати багато різних налаштувань. Також інтерфейс розділений на вкладки, які впливають одна на одну. Навіть у разі попереднього досвіду користування цим адоном, можна не одразу згадати, як досягти бажаного

результату. З технічної точки зору, кожен параметр впливає на кілька аспектів одночасно – тому спроби змінити одну характеристику часто призводять до небажаних візуальних змін в інших частинах дерева.

Для ілюстрації цього можна подивитись на рисунок 1.2, який демонструє початковий вигляд адону Sapling Tree Gen у Blender. Вже з перших кроків користувача зустрічає велика кількість налаштувань, що згруповані у різних меню. Наприклад, у розділі Settings зібрано основні параметри генерації, які можуть змінювати форму дерева, тип гілок, кількість листя тощо.

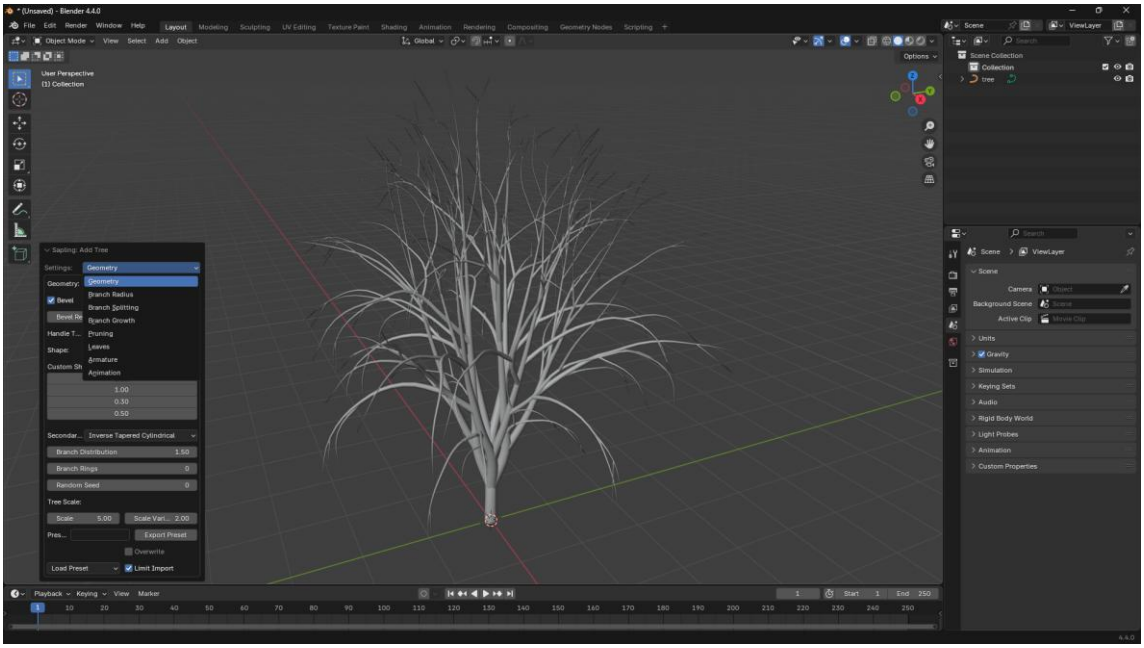

Рисунок 1.2 – Початковий вигляд дерева та інтерфейс адону Sapling Tree Gen у Blender з відкритим меню Settings

Через велику кількість взаємопов’язаних параметрів усі зміни потрібно вносити обережно. Наприклад, у вкладці «Branch Splitting» є чотири параметри «Split Angle» для різних рівнів гілок. Якщо значення другого або третього з них зробити надто великим, гілки починають скручуватися у спіраль і виглядають неприродно. Приклад такого результату наведено на рисунку 1.3.

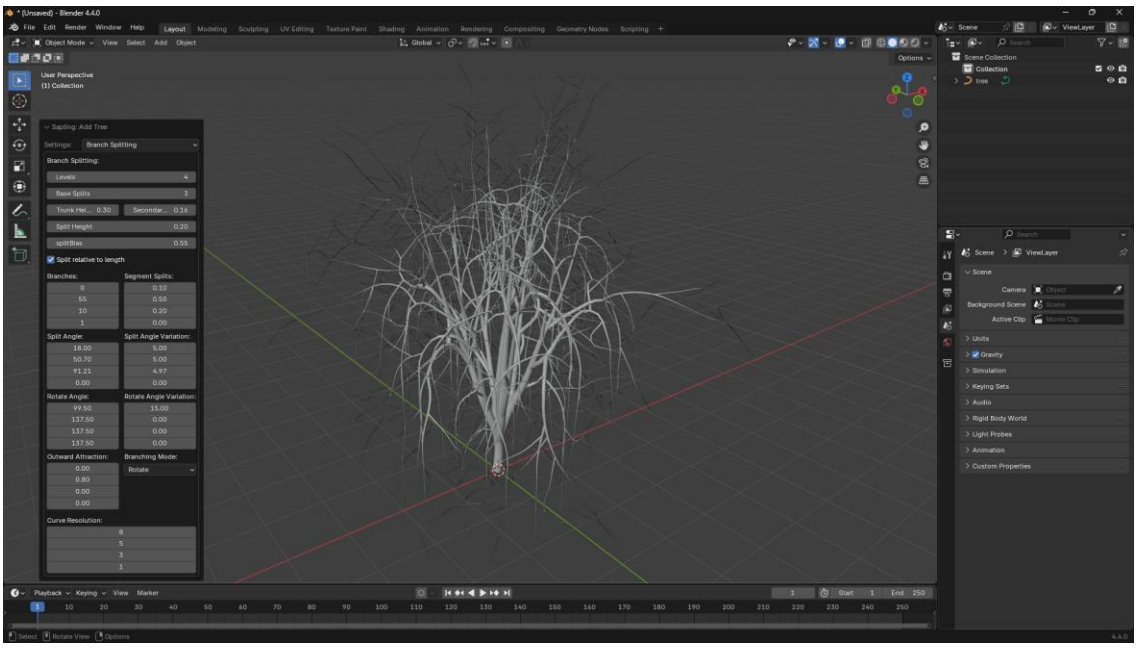

Рисунок 1.3 – Приклад дерева при великих значеннях параметра Split Angle

Проте значною перевагою цього адону є його універсальність. За допомогою Sapling Tree Gen можна створювати різні види дерев: дуби, сосни, клени, щось умовне чи стилізоване. Це зручно для тих, хто хоче мати все в одному місці. Але якщо потрібно працювати тільки з одним типом дерева, наприклад, з ялинами – така багатофункціональність більше заважає. Адже є забагато зайвого, що може відволікати від головного.

Іноді треба, щоб отримання результату було простим та швидким. Можливість мати менше параметрів, але більше контролю саме над тим, що необхідно для конкретно завдання, може бути більш важливим. Замість того щоб кожного разу згадувати, який саме параметр відповідає, наприклад, за вигин гілки або висоту стовбура, краще мати окремий інструмент, який створений під конкретний тип дерева. Він буде простішим, але неперевантаженим та швидким для використання.

Тому, не дивлячись на великі можливості і гнучкість Sapling Tree Gen, його складність і багатофункціональність не завжди є зручними у використанні. У деяких випадках доречніше створити простий і вузько спеціалізований інструмент, який буде легким у використанні. Хоча він буде

мати менше параметрів для налаштування, але він буде створений саме під ту задачу, яку необхідно виконати.

1.5 Аналіз результатів генерації тривимірних ялин за допомогою штучного інтелекту

Із розвитком інструментів на основі штучного інтелекту з'являється дедалі більше сервісів, що обіцяють автоматичну генерацію тривимірних об'єктів. Для деяких це є новими можливостями, адже такий спосіб дозволяє швидко створювати об'єкти, візуальні ідеї чи навіть цілі сцени. Для інших – питання естетики, контролю та унікальності залишаються принциповими. Попри скепсис до такого підходу, було вирішено протестувати кілька популярних сервісів – не для подальшого використання, а для порівняння з процедурною моделлю.

Такі сервіси можуть бути корисними на етапі концептуального дизайну або для заповнення другорядного фону, але рідко дають достатній рівень деталізації для професійної сцени. Крім того, користувач обмежений у точному контролі над структурою моделі, що важливо у складних або стилістично витриманих проєктах.

Першим прикладом став сервіс Ludo AI [10]. Він генерує одне тривимірне дерево, яке далі можна використовувати у сцені. Його результат виглядав доволі декоративно: модель мала загальний силует, була заповнена об'єктами, що нагадують хвою, але при ближчому розгляді було зрозуміло, що вся структура надто абстрактна. Гілки виглядають умовно, форма стовбура не має логіки, а крона більше схожа на згусток випадкових елементів. У такій формі дерево може бути доречним лише як частина умовного або стилізованого середовища, але не в реалістичній природній сцені (рис. 1.4).

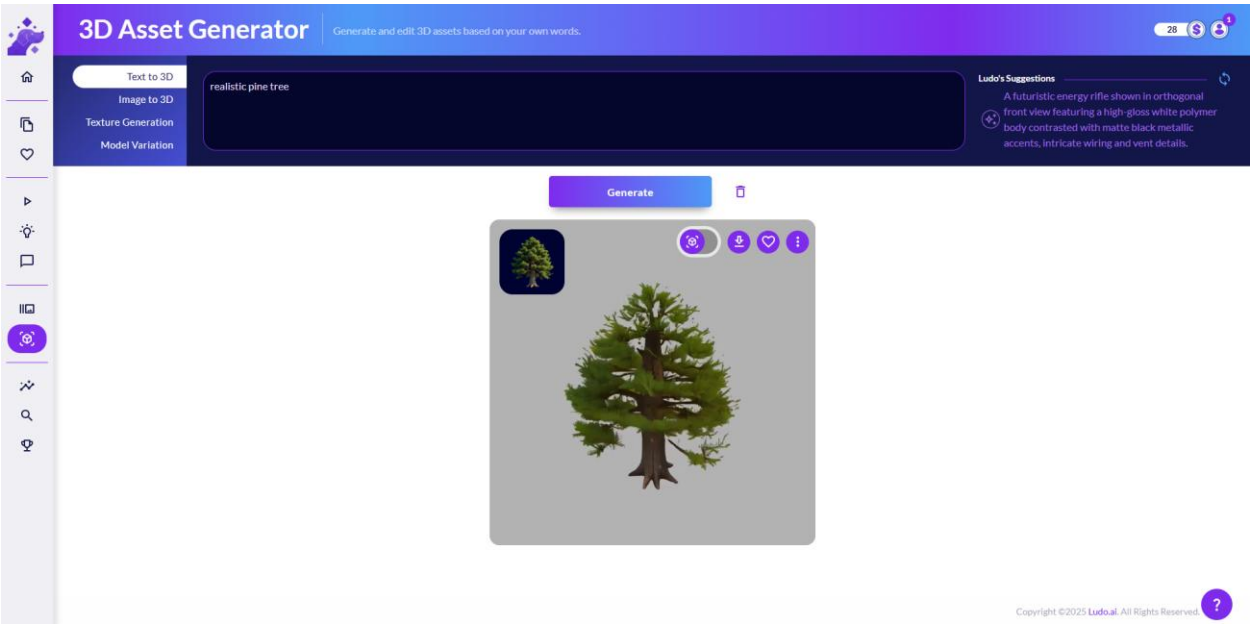

Рисунок 1.4 – Тривимірне дерево, згенероване у Ludo AI

Щоб переконатись, як така модель виглядає у реальному середовищі, вона була завантажена до Blender, де її було освітлено за допомогою HDRI й відрендерено для оцінки. Попри всі налаштування, результат залишався стилізованим і неприродним (рис. 1.5).

Рисунок 1.5 – Рендер дерева з Ludo AI у сцені з HDRI-освітленням

Другим прикладом став сервіс Tripo3D AI [11], який працює за схожим принципом, але одразу генерує кілька варіантів результату. У відповідь на запит про реалістичну тривимірну ялину було отримано чотири варіанти моделей: у першому було чотири, а в усіх наступних – по три. Вони відрізнялися між собою формою, пропорціями, густотою крони та загальним силуетом. На етапі попереднього перегляду всі вони виглядали доволі переконливо – як мінімум, геометрія була пропорційною, форма дерев чіткою, а симетричність надавала їм декоративної завершеності (рис. 1.6).

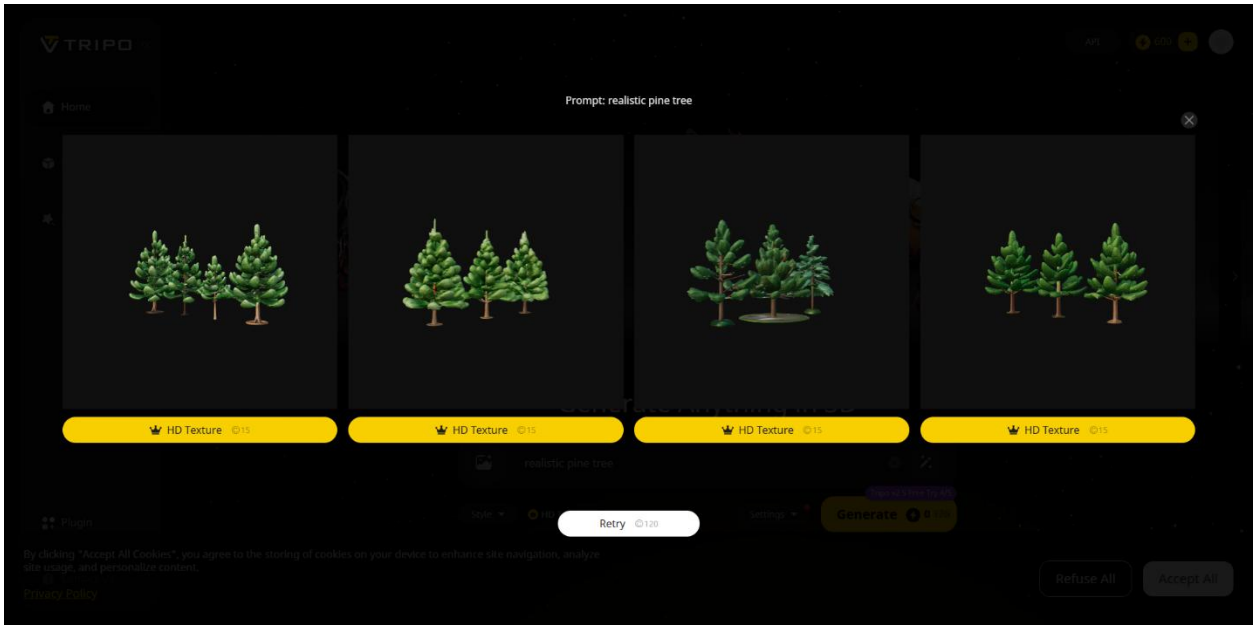

Рисунок 1.6 – Набір ялин, згенерованих Tripo3D AI

Серед них було обрано перший варіант – найбільш візуально привабливий. До того ж саме в ньому було чотири дерева замість трьох, як у решті, що дало більше варіантів для оцінки. Його загальна модель теж була завантажена до Blender і відрендерена. Навіть після освітлення і налаштування сцени результат залишався декоративним: силует був доволі впізнаваний, але структура була надто спрощеною. Дерево виглядало стилізовано, без природної складної будови, гілки не мали органічного розгалуження, а хвоя виглядала як абстрактна маса з повторюваних форм (рис. 1.7).

Рисунок 1.7 – Рендер вибраного набору ялин з Tripo3D AI

Усі ці тести показали одне: сучасні сервіси на основі штучного інтелекту поки що не можуть створити справді реалістичну тривимірну модель дерева. Вони не можуть відтворити природність реальних дерев. Це може бути зручним для концепт-артів, ігор або стилізованих візуалізацій, але не для сцени, де кожен елемент повинен виглядати реалістично.

1.6 Аналіз існуючих моделей замку Гоґвортс

Перш ніж почати роботу над власною моделлю замку, варто було з’ясувати, чи існує вже щось подібне. На перший погляд, здавалося б, тема давно вичерпана – фільми вийшли багато років тому, є мільйони фанатів, а інструменти доступні. Але після глибшого пошуку з’ясувалося, що ситуація складніша, ніж можна було уявити.

Одним із найкращих джерел інформації став блог Hogwarts 4D [12]. Його автор детально досліджував всі варіації замку, які використовувались у фільмах, збирав схеми, порівнював плани, описував різні аспекти Гоґвортсу й

хронологію його змін. Він також створював власні моделі замків для кожного фільму, але самих файлів з моделями на сайті немає. Отже, використати його проєкт безпосередньо як основу – неможливо. Водночас блог виявився унікальним не лише як джерело зображень, а як платформа, що систематизує різні варіації архітектури замку в часі. І хоча з нього не можна було безпосередньо імпортувати модель, концептуально він задав напрям роботи з візуальними джерелами.

Далі були спроби знайти моделі на відеохостингових платформах. Один із найпомітніших прикладів – відео Pietro Chiovaro в YouTube [13]. Його замок виглядає атмосферно, адже має нічне освітлення, ліхтарі, легкий туман і гарну композицію. Але це радше візуальна презентація, ніж повноцінний робочий проєкт. Сама модель не публікується, структура сцени базується більше на накладених текстурах, а не на геометрії. Відтворити це без доступу до деталізованих фотографій майже неможливо.

Ще один приклад – проєкт у стилі low poly [14, 15]. Такі моделі легкі, стилізовані, добре оптимізовані. Їх часто використовують у мобільних іграх, VR-сценах або при створенні графіки мультиплікаційних фільмів. Але у випадку з відтворенням Гоґвортсу був потрібний інший рівень деталізації. Такий low poly стиль не відповідає реальному вигляду замку, а є лише стилізацією його основних частин. Він не дозволяє передати архітектурні особливості, складну геометрію й деталізацію. Тому такий спосіб теж неможливо використати в такому проєкті.

Також окремо варто згадати платформи з готовими моделями. Наприклад, Sketchfab або CGTrader. Там можна знайти різні варіанти, які є як дуже простими, так і доволі деталізованими. Частина з них доступна безкоштовно, інші – лише за оплату. Але майже завжди є нюанс – або модель зроблена «на око», або не містить повної сцени, або створена під рендер конкретного кадру. У більшості випадків – це частини інтер’єру, окремі вежі або стилизовані проєкти без логіки простору. Крім того, в окремих випадках сцени виявлялися лише частково функціональними: відсутні матеріали,

некоректна топологія або неправильне масштабування. Тому навіть при формальному доступі до файлів, реальне використання вимагало б їх майже повної переробки.

Аналіз показав: проблема не в тому, що моделей немає взагалі, а в тому, що немає саме такої, яка підходила б і структурно, і композиційно, і технічно. Усі наявні приклади або вигладють не так, як треба, або створені для іншої мети, або не передбачають подальшої інтеграції в повноцінну тривимірну сцену з середовищем та анімацією.

Отже, можна зробити висновок, що готової сцени, яка би відповідала всім вимогам – немає. А ті, що є, або недоступні, або створені з іншою метою. В такому випадку краще зробити власну модель повністю з нуля. Це складніше, але й ефективніше. Адже тоді є повний контроль над формами, логікою будови, розміщенням деталей. А ще – свобода адаптації під сценарій, освітлення й анімацію. Такий підхід дозволяє не лише візуально контролювати результат, а й забезпечити інтеграцію з іншими компонентами сцени, наприклад, ландшафтом і деревами.

1.7 Аналіз архітектури Гоґвортсу у фільмах та вибір версії для відтворення

На перший погляд може здатися, що Гоґвортс у фільмах виглядав завжди однаково. Зазвичай, замок можна впізнати з першого кадру. Але насправді його форма змінювалася постійно. Іноді це були непомітні перестановки, але часом – масштабні зміни, які кардинально змінювали структуру будівлі. Причини були різні: зміна режисерів, розвиток технологій, побажання операторів або просто сценарні необхідності.

Це добре працює в кіно, де кожен кадр створюється з урахуванням ракурсу, композиції й драматургії. Але для тривимірного моделювання така варіативність створює серйозну проблему, бо немає єдиної, канонічної версії

замку – є лише багато різних варіантів, які суперечать один одному. Кожен з них має власну логіку композиції, масштаби, розташування ключових елементів. Об’єднати їх у єдину структуру практично неможливо без втрати достовірності, тому необхідний чіткий критерій вибору. Тож, щоб отримати узгоджену тривимірну модель, доводиться обирати одну версію як базову й ігнорувати решту – навіть якщо в інших фільмах були цікаві архітектурні рішення.

У першому фільмі замок виглядав просто. Багато сцен знімали в реальних локаціях: замку Алнік, соборі Дарема й університетських коридорах. Це створювало атмосферу старовинної магії, але ускладнювало логіку побудови. Деякі вежі залишались ізольованими, а частини двору або не з’являлися в кадрі, або були представлені окремими епізодами без загальної зв’язності.

У третьому фільмі з’явилась нова башта з годинником. Також змінився рельєф – замок опинився серед гір. Доріжки стали крутішими, хижу Геґріда перенесли на інше місце. У п’ятому фільмі реконструювали дворик перед астрономічною вежею. Його збільшили, щоб зручно знімати масові сцени. У шостому з’явилася нова Астрономічна вежа – з іншими формами, іншим дахом, новим обрамленням. А у восьмому – віадук, який з’єднував частини замку, став окремою структурою. Він розтягнувся і почав виходити з боку скелі. Двір перед Великою залою став ще більшим.

На екрані всі ці зміни виглядають гармонійно та природно, тому глядач сприймає замок як цілісне місце. Проте при спробі відтворити його у вигляді тривимірної моделі можуть виникнути певні труднощі, адже різні варіанти замків не узгоджуються між собою.

Саме тому для роботи була обрана конкретна версія – та, що з’являється у шостій частині, «Гаррі Поттер і напівкровний Принц». Її створювали для студії Warner Bros. як повноцінний фізичний макет, і саме вона збереглася в музеї, де її досі можуть побачити відвідувачі. Вона деталізована, стабільна, вважається найбільш канонічною та найкраще показує структуру замку.

Головне – її часто фотографували з різних ракурсів, тому фотографії цієї версії найпростіше знайти.

Отже, створення універсальної моделі Гоґвортсу є складним не лише з технічної точки зору, а й з аналітичної. Для цього потрібно враховувати різні інтерпретації, порівнювати деталі та зберігати цілісність образу. Сам вибір версії стає ключовим кроком у моделюванні – саме від нього залежить послідовність усіх подальших рішень. Вибір Гоґвортсу з шостого фільму був зумовлений доступністю фотографій замку, стабільністю та рівнем деталізації, що робить цю версію найбільш придатною для моделювання сцени.

1.8 Постановка задачі

Таким чином, створення реалістичних тривимірних сцен з великою кількістю об’єктів є актуальним завданням у сфері візуалізації. Особливо це стосується сцен, де треба деталізувати як архітектурні моделі, так і складні природні елементи, зокрема дерева. Тому ставиться завдання побудови тривимірної моделі замку Гоґвортс з реалістичним ландшафтом та розробки адону для автоматичного створення ялин у середовищі Blender.

Об’єктом роботи є тривимірна сцена, що поєднує архітектурну модель замку Гоґвортс із автоматично згенерованими реалістичними ялинами.

Метою роботи є створення Blender-адону для генерації ялин на основі кривих Безьє, математичних залежностей і керованої варіативності, а також побудова сцени для подальшого рендеру й створення анімованого відео.

Для досягнення мети необхідно вирішити такі завдання:

– проаналізувати наявні тривимірні моделі замку Гоґвортс і визначити ту, яка буде найбільше підходити для відтворення;

– дослідити інструменти програмного середовища Blender, зокрема можливості скриптової автоматизації за допомогою Python;

– реалізувати адон для генерації реалістичних ялин на основі математичних кривих і параметричних залежностей;

– створити вручну тривимірну модель замку на основі зображень та фотоматеріалів;

– побудувати повноцінну сцену з деревами, освітленням і камерою;

– змонтувати відео з анімованим польотом камери та візуалізованою композицією.

2 МАТЕМАТИЧНІ МОДЕЛІ ФІЛЬТРАЦІЇ ЗОБРАЖЕНЬ

2.1 Загальна структура та логіка роботи адону

Процедурне моделювання – це спосіб створення об’єктів на основі математичних залежностей, а не ручного створення кожного елементу. Такий підхід дозволяє отримати складні форми за допомогою коду, керованого змінними [16]. У випадку з деревами це особливо зручно, адже гілки, стовбур, листя мають свою закономірність, яку можна описати формулами. А якщо ще додати трохи варіативності, то кожне нове дерево буде виглядати унікально, але зберігатиме природну форму.

Саме цю ідею було покладено в основу розробки адону для Blender, що автоматично генерує реалістичні ялини. Структура побудована навколо трьох основних елементів: центрального стовбура, гілок, що спіраллю розходяться по висоті, і дрібних підгілок у вигляді плоских площин. Усі частини створюються за допомогою кривих Безьє та площин, розміщених у просторі згідно з заданими правилами. Крім цього, враховується висота, рівень деталізації, контрольовані відхилення та розумна система масштабування.

Кожен об’єкт генерується по черзі. Спочатку створюється стовбур. Потім, на основі його довжини, обчислюється положення гілок. Далі до них додаються площини, що імітують хвою. Вся логіка побудована так, щоб дерево залишалося природним незалежно від параметрів. Форма буде перебудовуватися автоматично, а дерево все одно буде виглядати пропорційно.

Також не треба взаємодіяти з кодом безпосередньо, адже усі основні параметри винесено в окрему панель. При використанні адону можна змінити висоту дерева, товщину стовбура, випадковість, кількість гілок і площин. Отже, натиснувши лише одну кнопку, можна отримати повну модель, яка вже буде з оптимізованою геометрією.

Усі об’єкти, які створюються в процесі, наприкінці автоматично об’єднуються. Це не просто зручно, адже це також знімає навантаження з Blender, особливо якщо дерев у сцені багато. Тобто, у результаті користувач отримує готовий об’єкт, який зручно масштабувати, анімувати або використовувати у великій композиції – наприклад, на задньому плані сцени з Гоґвортсом.

2.2 Побудова стовбура ялини

Уся генерація дерева починається зі стовбура. Він є основою, від якої залежить розміщення гілок, їх масштаб і пропорції. Якщо зробити його непропорційним або недостатньо плавним, усе дерево виглядатиме неприродно. Тому до цієї частини моделі підхід є особливо ретельним.

Стовбур створюється за допомогою кривої Безьє, яка дозволяє формувати складну лінію лише кількома контрольними точками. Цей тип кривої плавний за замовчуванням, легко масштабується, і при цьому дає повний контроль над формою, адже можна змінити розташування однієї точки, і вся крива миттєво перебудується. Саме ця властивість і зробила її зручною для генерації дерев. Адже, як зазначає Samuel Buss, дизайнер може контролювати всю криву, змінюючи лише кілька точок [17], що особливо важливо для автоматизованого створення природних об’єктів.

У документації Blender вказано, що криві Безьє редагуються трансформуванням керувальних точок і їх держаків [18], що забезпечує контроль гладкості. Також завдяки цьому їх доволі легко редагувати.

Крива будується вертикально, вздовж осі *Z*. Кожна точка на ній розташована одна над одною. Їх кількість визначається змінною, яка дорівнює 7. Для кожної точки обчислюється висота та радіус, які задають форму майбутнього стовбура. Висота точок розраховується рівномірно – від основи до верхівки. Формула виглядає таким чином:

$$z_i = h \times \frac{i}{n-i}, \tag{2.1}$$

де h – загальна висота дерева;

n – кількість точок;

i – індекс поточної точки (починаючи з нуля).

Таким чином, усі точки рівномірно розподіляються вздовж вертикальної осі. Але форма дерева не може бути циліндричною – тому кожна точка додатково отримує радіус, який поступово зменшується до верху. Це досягається за допомогою експоненційного зменшення:

$$r_i = r \times 0{,}76^i, \tag{2.2}$$

де r – початковий радіус біля основи.

Завдяки цьому нижня частина стовбура залишається масивною, а верхня буде тонкою й витонченою. Для останньої точки радіус задається вручну і дорівнює 0,001. Це робиться для того, щоб кінець стовбура виглядав як природна верхівка, що сходить нанівець.

Після побудови кривої визначається форма її згладження. Усі ручки встановлюються у режим *AUTO*, тобто Blender самостійно згладжує вигини між точками. Це важливо, бо завдяки такому підходу крива не має різких зламів, а лінія стовбура набуває природного, м’якого вигляду – ніби її справді створила природа, а не машина.

На цьому етапі крива – це лише лінія. Щоб вона стала об’ємною, у Blender використовується параметр *bevel depth*. Це є товщиною трубки, яку створює програма навколо кривої. Значення параметру розраховується як частка від початкового радіуса:

$$bevel\ depth = r \times 0{,}4. \tag{2.3}$$

Це значення підібрано так, щоб об’єм був переконливим, але не надто товстим, і не створював зайвих полігонів. Важливо й те, що товщина автоматично масштабується, якщо змінити висоту дерева – бо всі залежності задані через змінні. У результаті змінити геометрію стовбура можна в будь-який момент, не перебудовуючи криву вручну.

У такий спосіб із простої вертикальної кривої формується повноцінна геометрична основа дерева. Вона легка для обчислення, не перевантажує сцену, але водночас виглядає переконливо. До того ж, ця основа – не лише візуальний елемент. Вона служить каркасом, від якого відштовхуються гілки та всі наступні рівні деталізації.

2.3 Створення гілок

Після побудови стовбура наступним етапом є формування гілок. Саме вони задають характер дерева, його силует і густоту. Як і сам стовбур, гілки створюються на основі кривих Безьє, які згодом також перетворюються в об’ємні об’єкти. Основна ідея полягає в тому, що кожна гілка є зменшеною копією стовбура, але з власною висотою, радіусом і унікальним розташуванням у просторі.

Створення починається з визначення точки, в якій має з’явитися гілка. Для цього спочатку обчислюється висота головного стовбура – її потрібно знати, щоб правильно розмістити всі гілки. Проте гілки не ростуть прямо від основи, тому задається діапазон – від 40 % до 99 % висоти. Цей інтервал найбільш наближений до того, як це відбувається в реальних хвойних деревах: нижня частина без гілок, а на верхівці їх уже теж майже немає.

Далі розраховується положення кожної гілки по вертикалі. Вони рівномірно розташовуються в межах обраного діапазону. Такий підхід дозволяє зберегти логіку розподілу гілок, наближену до природного розташування. Щоб отримати такі координати, використовується формула:

$$z_i = z_{min} + (z_{max} - z_{min}) \times \frac{1}{n}, \quad (2.4)$$

де $z_{min} = h \times 0{,}4$;

$z_{max} = h \times 0{,}99$;

h – загальна висота стовбура;

n – кількість гілок;

i – індекс гілки.

Кожна точка розташування надає стартову позицію для окремої гілки. Але цього недостатньо, бо гілки в природі не однакові. Їх форма залежить від висоти: внизу вони ще тільки з’являються, короткі та тоненькі; у середній частині досягають найбільшого розвитку – стають товстішими й довшими; а ближче до вершини знову зменшуються, створюючи природне загострення крони. Щоб реалізувати цю закономірність, використовуються умовні масштабні коефіцієнти. Їх значення змінюється в залежності від того, на якій висоті розміщена гілка:

– якщо $z < 0{,}4$:

1) висота $h = s \times 0{,}5$;

2) висота $r = s \times 0{,}05$;

– якщо $z \leq 0{,}4 \leq 0{,}7$:

1) $h = s \times (0{,}41 + 0{,}412 \times \frac{1{,}2z - 0{,}4}{0{,}3})$;

2) $r = s \times (0{,}08 + 0{,}1 \times \frac{z - 0{,}4}{0{,}3})$;

– якщо $z > 0{,}7$:

1) $h = s \times (0{,}7 - 0{,}74 \times \frac{1{,}2z - 0{,}4}{0{,}3})$;

2) $r = s \times (0{,}2 - 0{,}24 \times \frac{z - 0{,}7}{0{,}29})$,

де s – загальний масштаб, який дорівнює 1 у даній реалізації.

Завдяки цьому гілки змінюються плавно, без різких переходів, і вся структура дерева виглядає гармонійно. Але не менш важливо – розміщення в просторі. Гілки не просто зростають вгору чи вниз – вони розходяться по спіралі. Це досягається за допомогою поступового повороту кожної наступної гілки відносно попередньої. Щоб уникнути повторюваності, кут обертання вибирається випадково з діапазону від 75° до 90°. У результаті виходить природна спіралевидна структура, яку часто можна побачити у справжніх хвойних деревах.

Окрім цього, щоб уникнути механічної симетрії, у моделі реалізовано вигин кінців гілок. Але не довільно, а згідно з автоматичною логікою, що залежить від висоти. Зсув відбувається у трьох останніх точках кривої – ті змінюють своє положення по осі X. При цьому важливо пам’ятати, що зміщення задається у локальній системі координат гілки, а сама гілка згодом повертається навколо стовбура. Тобто вигини не в один бік, а рівномірно розходяться по колу.

У середній зоні дерева (приблизно від 40 % до 70 % висоти) кінці гілок відхиляються сильніше – у цій частині вони найдовші, тож і мають прогинатися під власною вагою більше. У верхній зоні (від 70 % 99 %) відхилення менше і в зворотному напрямку – щоб верхівка дерева не була надто розкиданою. Таким чином створюється ефект легкого прогину, знайомий із реальних дерев.

Сама деформація обраховується за формулою:

$$\Delta x_{\text{кін.}} = \pm(0{,}1 \pm 0{,}1 \times f), \tag{2.5}$$

де f – коефіцієнт деформації, що залежить від висоти гілки.

Завдяки цьому методу гілки не виглядають штучно вирівняними – у кожної з них є свій власний напрямок, нахил і ритм, але вся структура зберігає логіку й рівновагу.

Окрім самих пропорцій, також враховується загальна висота дерева. Це важливо, якщо дерево масштабується, його гілки мають змінюватися разом із ним. У протилежному випадку виникне дисбаланс – або гілки виглядатимуть надто дрібними для великого дерева, або надто масивними для маленького. Саме тому в реалізації передбачено додаткові глобальні коефіцієнти, які масштабують і довжину, і товщину гілок залежно від розміру всього дерева. Таким чином, структура залишається цілісною – навіть якщо висоту сильно змінити, усе виглядатиме гармонійно і збалансовано.

Таким чином, за кожною гілкою стоїть цілий набір розрахунків – від координат до вигину. І вся складність лишається в коді, а результат – природний, м’який і переконливий.

2.4 Додавання площин

Після генерації основного стовбура та гілок до дерева додаються площини, які імітують хвою. Саме ці площини задають обсяг і пишність крони, візуально наповнюють структуру, роблячи її більш природною. Вони не мають складної геометрії – лише стандартні полігони типу «площина». Але завдяки правильному розташуванню, орієнтації та масштабу, саме вони створюють відчуття реальної хвої.

Кожна площина створюється в одній із точок уже згенерованих гілок. Для цього випадковим чином вибирається одна з трьох точок Безьє – друга, третя або четверта. Потім обчислюються координати цієї точки, але не в локальній системі, а в глобальній – тобто з урахуванням обертання та положення гілки в сцені. Завдяки цьому кожна площина органічно вписується в просторову структуру гілки.

Далі визначається масштаб кожної площини, бо усі вони не однакові. І щоб зберегти логіку росту дерева, розмір площини залежить від її висоти. У середній частині дерева – між 40 % і 70 % – хвої найбільше, вона густа, тому

площини мають бути більші, що й треба врахувати під час обчислень. Для цього застосовується така формула:

$$s = 1 + 0{,}5 \times \frac{\frac{z}{h} - 0{,}4}{0{,}3}, \tag{2.6}$$

де s – коефіцієнт масштабу;

z – висота площини;

h – повна висота дерева.

Це поступове збільшення масштабу створює ефект обсягу по центру. А от ближче до вершини дерева, у зоні від 70 % до 99 %, навпаки – площини стають меншими. Там формується візуальне звуження крони, і щоб воно не виглядало штучно, застосовується інша формула:

$$s = 1 - 0{,}5 \times \frac{\frac{z}{h} - 0{,}7}{0{,}29}. \tag{2.7}$$

Це дозволяє плавно перейти від розлогої середини до тонкої верхівки. Завдяки такому простому, але продуманому підходу, крона не виглядає рівномірною або механічною. Вона набуває природного силуету: щільного всередині, але легшого зверху. І водночас – адаптивного: якщо змінюється загальна висота дерева, масштаб площин буде автоматично підлаштовуватися. Усе оновлюється без потреби в ручному втручанні, зберігаючи пропорції незалежно від налаштувань.

Окрім масштабу, кожна площина отримує свою орієнтацію, бо вони не розташовуються вертикально чи горизонтально. Щоб уникнути симетрії та регулярності, що псує враження, для кожної площини генерується невелике випадкове обертання навколо трьох осей – X, Y і Z. Завдяки цьому площини розгортаються в різних напрямках, і вся крона виглядає хаотичною, як це й буває в природі.

Однак площин дуже багато – іноді тисячі. Якщо залишити їх окремими об’єктами, Blender просто не впорається з навантаженням. Тому реалізовано так звану пакетну генерацію: після створення певної кількості площин вони автоматично об’єднуються в один об’єкт. Це значно полегшує навігацію сценою, пришвидшує рендеринг і зменшує кількість об’єктів, які потрібно обробляти.

На фінальному етапі всі площини – вже згруповані, обернуті, масштабовані – стають частиною однієї суцільної структури. Разом вони формують об’ємну, деталізовану, але при цьому легку для обчислення крону дерева. Завдяки такому підходу адон дає змогу створити ялину, яка виглядає переконливо навіть на середньому або ближньому плані – без перенавантаження сцени.

2.5 Обґрунтування вибраних засобів тривимірного моделювання

2.5.1 Обґрунтування вибору типів геометричних об’єктів

У будь-якій тривимірній сцені геометрія починається з форм. І хоча в Blender їх доступно багато, основні завжди ті самі: куб, коло, площина, крива. Від того, яку з них обрати з самого початку, будуть залежити і швидкість роботи, і чистота геометрії, і можливість точного редагування. Тому рішення про використання того чи іншого об’єкта приймалося не інтуїтивно, а виходячи з логіки форми й задачі, яку він мав виконувати [19, 20].

Там, де потрібно було чітко прямокутну будівлю – з рівними фасадами, гострими кутами і лінійними пропорціями – найзручніше було починати з куба. Його структура одразу співпадає з координатною сіткою, і будь-яке масштабування, розрізання або вирізання відбувається просто й передбачувано. Особливо це стало в пригоді при створенні частин замку з симетричними фасадами, дахами або масивними стінами. Адже редагування кожної площини вручну не потребувало додаткових перетворень.

Якщо форма була округлою, наприклад, як у більшості башт, то використовувалось коло. Але і тут усе залежало від ситуації. У простих об’єктах достатньо було восьми точок, щоб задати базову форму, яку далі можна деталізувати. А в складніших – як-от у Мармуровій башті – кількість точок збільшувалась одразу при створенні, щоб уникнути ламаних країв. Це дозволяло контролювати плавність і точність вигину без зайвих модифікаторів або додаткового згладжування.

Окремо варто згадати криві. Вони використовувались не часто, але саме там, де інші об’єкти не справлялись. Наприклад, аркові прикраси, вигнуті балки, декоративні елементи або складні бічні елементи проходу дерев’яного мосту, які мали повторювати складну лінію. У таких випадках крива давала можливість одразу задати потрібний вигляд. Це значно полегшувало побудову, а головне – гарантувало рівномірність форми. Адже вручну домалювати симетричну арку з ідеальним згином – майже неможливо, а крива зробить це автоматично. До того ж, у подальшому до неї легко застосовується об’єм через параметр товщини, що дозволяє перетворити лінію на повноцінну геометричну форму.

Тож у результаті тип об’єкта визначався не за зручністю, а за відповідністю до задачі. Якщо форма була прямою і геометрично чіткою – використовувався куб. Якщо округла – коло. Якщо складна і вигнута – крива. Це дозволяло не лише прискорити моделювання, а й зменшити кількість зайвих точок, уникнути перевантаження сцени і зробити кожен об’єкт керованим. Завдяки цьому легше далі додавати деталі, змінювати масштаб або інтегрувати елемент у складнішу структуру. Власне, тому жоден об’єкт у сцені не був випадковим – форма завжди відповідала кінцевому результату.

2.5.2 Обґрунтування використання модифікаторів під час моделювання

Коли сцена складається з великої кількості веж, фасадів, вікон, арок і дрібних елементів, то зберігати однаковість вручну практично неможливо. Навіть не через брак часу, а через обмеження самого процесу. І саме тому використання модифікаторів також було необхідним. Вони дозволили зменшити кількість повторюваних дій і зберігати природність.

Одним із найчастіше вживаних був модифікатор дзеркального відображення. У багатьох частинах сцени симетрія була передбачена самою архітектурою – особливо це стосується вікон, декоративних арок або інших елементів, які мали однакову форму з обох боків будівлі. Але вручну дублювати такі елементи значило ризикувати мікровідхиленнями, що зіпсують загальний вигляд. Замість цього, досить створити лише один об’єкт, а далі треба застосувати віддзеркалення. Таким чином, досягається повна симетрія, але з можливістю змінювати лише одну сторону. І що важливо – вона працює навіть після застосування інших модифікаторів. Це давало змогу, наприклад, автоматично дублювати вікна на всі сторони веж.

Іншим ключовим модифікатором був булевий. Саме через нього виконувалась більшість вирізань у геометрії – від вікон до проходів. Він дозволяє об’єднувати, вирізати або перетинати об’єкти, створюючи складні форми без ручного моделювання. Але найважливіше – можливість комбінувати різні типи операцій. В одному випадку вікно потрібно було саме вирізати – у буквальному сенсі зробити отвір у стіні. А в іншому – навпаки: об’єднати, як у випадку зі сходами, де замість того щоб витісняти форми, нові елементи додавалися вручну й об’єднувались у складну, єдину геометрію, яку далі можна було редагувати. І там, і там – один і той самий модифікатор, але з різними режимами. Це давало змогу працювати логічно, залежно від ситуації, а не шукати інші складніші варіанти.

У випадках, коли елементів було багато – наприклад, фасади, що повторюються, або сходи – найкраще працював модифікатор масиву. Він

дозволяє задати кількість копій і відстань між ними – і все інше створиться автоматично. Це особливо зручно там, де важлива точність – скажімо, для повторів вікон або однакових частин стін. І тут знову – не тільки економія часу, а й гарантія, що всі елементи залишаться на своїх місцях, навіть якщо об’єкт буде змінено пізніше.

У специфічних випадках застосовувався модифікатор кривої. Його призначення – розташовувати об’єкти вздовж заданої кривої. Звучить просто, але в роботі має величезне значення. Адже виставити вручну, наприклад, вікна вздовж вежі – це багато часу редагування і все одно буде помітна неточність. А з кривою об’єкти самі підлаштовуються під вигин. І це особливо цінно там, де форма має бути не просто плавною, а рівномірно побудованою. Наприклад, у Мармуровій башті – всі вікна стоять на однаковій відстані одне від одного і точно слідують за контуром.

Саме тому модифікатори в цьому проєкті не були просто зручністю. Вони забезпечували точність, зменшували ризик помилок, дозволяли зберігати гнучкість у редагуванні, а також оптимізували сцену. У більшості випадків застосовувались кілька модифікаторів одночасно – і тільки завдяки цьому вдалося досягти балансу між складністю і правильністю моделі.

2.5.3 Обґрунтування підходів створення матеріалів і текстур

У тривимірному моделюванні поверхня об’єкта не менш важлива за його форму. Адже саме текстура визначає, з чого зроблений об’єкт – з каменю, металу, дерева чи скла. А ще – наскільки він старий, мокрий, новий чи запилений. Проте для того, щоб досягти цього ефекту, мало просто накласти картинку. Потрібно обрати метод, який дозволить цій поверхні поводитись природно під будь-яким кутом, при будь-якому освітленні й незалежно від масштабу.

Тому під час проєктування матеріалів у Blender зазвичай використовуються два підходи: процедурні текстури та текстури на основі зображень. Кожен з них має свої переваги і використовується залежно від задачі.

Процедурні текстури – це генеративні матеріали, створені всередині Blender за допомогою вузлів. Їх перевага в тому, що вони гнучкі. Вони не прив’язані до конкретної текстурної розгортки, тобто автоматично адаптуються до будь-якої форми. Також їх неможливо розтягнути чи пікселізувати, бо вони не мають фіксованого розміру. Це особливо корисно там, де потрібно покрити велику кількість об’єктів із різною топологією або забезпечити безшовність. Наприклад, для самого замку таким матеріалом була цегла. Крім того, параметри таких матеріалів легко змінювати: від кольору до масштабу деталей, що робить їх зручними для повторного використання в межах однієї сцени.

Текстури на основі зображень застосовуються тоді, коли важлива унікальність і деталізація [21]. Наприклад, при побудові природного рельєфу гір або складного покриття трави, яке складно згенерувати вручну. Такі текстури створюються на наборах фізично достовірних текстур, де кожна карта (кольору, рельєфу, шорсткості, зміщення) відповідає за окрему властивість матеріалу. Це дозволяє точно моделювати поведінку світла на поверхні – як воно розсіюється, відбивається чи поглинається.

Рішення між цими двома підходами завжди є компромісом. Процедурні матеріали забезпечують універсальність і легкість у використанні, а зображення – реалістичність і контроль. Тому в складних сценах обидва методи часто комбінуються. Важливо й те, що кожне рішення приймається не інтуїтивно, а виходячи з логіки моделювання: яка поверхня, в якому масштабі вона буде показана, наскільки важлива точність її вигляду.

Саме це дозволяє побудувати сцену, яка не просто виглядає гарно, а й поводиться очікувано у різних випадках: при зміні освітлення, при наближенні камери, при анімації. У результаті матеріал стає не просто розмальовкою

об’єктів, а повноцінною частиною моделі, яка взаємодіє з простором так, як би це робив справжній об’єкт.

2.5.4 Освітлення сцени

Для освітлення сцени використовувалась HDRI, яка одночасно виконувала роль і джерела світла, і фону. Це було найзручніше рішення, бо не вимагало створення окремих джерел світла чи налаштування для неба. У сцені передбачався природний ландшафт і відкрите небо, тому HDRI дозволяло одразу досягти реалістичного освітлення, без додаткових ламп і складних налаштувань.

Основна перевагою такого підходу є рівномірне і природне світло, що одразу правильно падає на всі об’єкти. Це важливо, особливо в великих сценах, де багато веж, переходів і виступів. З HDRI не потрібно вручну підлаштовувати тіні чи яскравість – вони з’являються автоматично, залежно від положення об’єкта. До того ж, зображення неба працює як фон, тож не виникає потреби створювати окрему геометрію або матеріал для заднього плану.

Тому саме HDRI було обрано як найбільш оптимальний варіант, адже воно дає одночасно освітлення, тіні, відбиття і фон, що значно полегшує роботу й дозволяє одразу бачити майже фінальний результат без зайвої постобробки.

2.5.5 Обґрунтування вибору рушія рендерингу

Щоб сцена виглядала переконливо, замало правильної геометрії й текстур – усе вирішує світло. А отже, критичним стає вибір рушія рендерингу. У Blender їх кілька, але кожен має своє призначення. Workbench – це

найпростіший варіант, який підходить лише для перегляду форми. Eevee – швидкий і зручний, особливо якщо сцена проста або розрахована на реальний час. Проте у випадках, коли потрібен реалізм – детальні тіні, відбиття, об’єм і глибина – саме Cycles дає той результат, який виглядає природно.

Цей рушій працює за фізичними принципами: враховує, як світло поводиться в реальності, як воно розсіюється, заломлюється чи відбивається від різних матеріалів. Саме тому текстури, мапи з імітацією рельєфу й прозорі поверхні поводяться у сцені очікувано – як у житті. Не треба довго налаштовувати кожен параметр, бо більшість речей просто працює автоматично. Якщо матеріал має рельєф – тіні самі його підкреслять. Якщо поверхня блискуча – вона буде блищати лише там, де треба.

Ще один великий плюс – можливість використовувати HDRI. Як вже було зазначено раніше, вони дають природне освітлення і фон, і саме Cycles правильно їх обробляє. Інші рушії не мають такої функціональності. Але з Cycles усе виглядає так, ніби сцена справді освітлюється навколишнім середовищем – із плавними переходами, глибокими тінями й правильними кольорами.

Так, такий рендер займає більше часу. Проте результат того вартий. Бо сцена виглядає як частина реального світу, а не як штучна модель. І саме тому для цього проєкту Cycles був не просто найкращим, а єдино логічним вибором.

2.5.6 Анімація сцени

Анімація у візуалізації є не просто рухом камери, а способом показати не лише форму, а й атмосферу сцени. Те, що в статичному рендері залишається на задньому плані, у русі може стати головним – залежно від того, як побудовано кадр. Тому ще на етапі проєктування враховується не лише композиція, а й те, як вона буде сприйматися у динаміці.

Щоб передати об’єм і масштаб сцени, використовуються три основні плани: загальний, середній та детальний. Загальний дозволяє побачити усе – розташування замку, ландшафт, структуру композиції. Середній – зосереджує увагу на окремих частинах: фасад, група веж, підвісний міст. А детальний – показує текстури, матеріали, окремі архітектурні елементи. Поділ на ці плани формує ритм, допомагає керувати увагою глядача і не дає сцені виглядати статичною. Кожен план виконує свою роль, створюючи глибину й підтримуючи цілісне сприйняття. Завдяки цьому анімація виглядає послідовною й логічною.

Крім того, завдяки цьому підходу можна демонструвати і масштаб, і детальність одночасно. Адже великі об’єкти без крупних планів здаються порожніми, а дрібні деталі – втрачаються без загальної картини.

З технічного боку, анімація базується на ключових кадрах. Кожен рух – це зміна положення, обертання або масштабу об’єкта, яку Blender автоматично інтерполює. Завдяки цьому можна створити плавне наближення, обертання навколо вежі або спокійний проліт через арку. І хоча сцена велика, керувати нею не складно – бо камера працює з тими ж інструментами, що й об’єкти.

Саме тому анімація є логічним завершення візуалізації. Вона дозволяє не тільки показати модель, а й зробити це так, щоб глядач повірив у її реальність – бодай на кілька секунд.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ СЦЕНИ ГОҐВОРТСА З АВТОМАТИЧНОЮ ГЕНЕРАЦІЄЮ ЯЛИН

3.1 Обґрунтування вибору середовища Blender

Як вже зазначалося у першому розділі, середовище Blender має все необхідне для створення як тривимірних моделей, так і для автоматизації роботи з ними. Саме завдяки цьому воно було обрано як єдине середовище для реалізації проєкту. І це є цілком логічним вибором, якщо врахувати, що тут можна не лише створити об’єкт, задати текстуру, налаштувати світло і сцену, а й написати власний адон із параметрами, які повністю адаптовані під власні потреби.

Ще однією перевагою є можливість написання коду безпосередньо всередині Blender. У ньому є вбудоване середовище для роботи з Python – простий текстовий редактор із доступом до внутрішнього API, який відкривається через вкладку «Text Editor». Завдяки цьому можна швидко тестувати зміни, коригувати логіку і миттєво бачити, як саме все працює у сцені. Це також особливо зручно під час налаштування інтерфейсу адону, коли важливо одразу перевірити, як реагує скрипт на введені параметри.

Окрім того, Blender дозволяє поєднувати ручне моделювання з процедурним. Це дає змогу створити сцену, де частина об’єктів змодельована вручну (наприклад, замок Гоґвортс), а інша частина – згенерована автоматично через адон (наприклад, ялини). Усе це створюється в єдиному файлі, в єдиній сцені, без конвертацій, імпорту чи інших складних рішень.

Такий підхід не лише спрощує розробку, а й економить час. Завдяки Blender можна зосередитися на логіці та композиції, не відволікаючись на технічні обмеження. Тому саме це середовище виявилось найбільш зручним і гнучким для реалізації проєкту.

3.2 Структура та логіка реалізації генератора ялин

Робота адону починається з опису службової інформації, яка міститься у спеціальній змінній *bl_info*. У ній зазначається назва адону, сумісна версія Blender, його розташування в інтерфейсі, версія, ім’я автора та короткий опис функціональності. Такий формат дозволяє Blender розпізнати адон та правильно його відобразити його.

Лістинг 3.1 Опис службової інформації Blender-адону:

```
bl_info = {
    "name": "Realistic Pine Tree Generator",
    "blender": (4, 0, 0),
    "category": "Add Mesh",
    "author": "Anastasiia Zinchenko",
    "version": (1, 0),
    "location": "View3D > Sidebar > Pine Tree Gen",
    "description": "Generates a realistic procedural pine tree with branches using curves",
    }
```

Після ініціалізації викликається функція створення стовбура, яка формує криву Безьє із заданою кількістю точок, обчислює для кожної з них координати по осі Z та відповідний радіус. Ці значення беруть участь у створенні плавного звуження стовбура до верхівки, а саме формуються за допомогою (2.1) та (2.2), описаних у підрозділі 2.2. У коді ці обчислення реалізовано таким чином:

Лістинг 3.2 Обчислення координат точок стовбура та їх радіусів:

```
radii = [trunk_radius * (0.76 ** i) for i in range(num_points - 1)] + [0.001]
z_positions = [height * (i / (num_points - 1)) for i in range(num_points)]
```

Після генерації стовбура створюються гілки. Їхнє розміщення залежить від висоти, і для цього обчислюється набір координат у межах від 40 % до 99 % від повної висоти дерева. Розподіл є рівномірним, що забезпечує гармонійне розміщення гілок уздовж стовбура. Кожна гілка створюється як окрема крива, подібна до стовбура, але з іншими масштабами та радіусами, які залежать від того, на якій висоті розміщена гілка.

Кут нахилу, а також орієнтація гілок по колу реалізовані через обертання навколо осі *Z*. Кожна наступна гілка повертається на дещо інший кут, щоб створити не тільки ефект спіралі, а й природну нерівномірність. Це видно у фрагменті коду нижче.

Лістинг 3.3 Обертання гілок навколо осі *Z* з випадковим кутом:

```
rotation_increment = math.radians(random.uniform(75, 90))
prev_rotation += rotation_increment
branch.rotation_euler[2] = prev_rotation
```

Після розміщення основних гілок на них додаються площини, які імітують хвою. Це прості квадратні об’єкти, які розташовуються у випадкових точках на гілках, а саме у другій, третій або четвертій точці Безьє-кривої. Їх координати переводяться у глобальний простір, і саме в цих точках додаються площини.

Для кожної площини визначається масштаб. Він залежить від висоти її розміщення. У середній частині дерева (від 40 % до 70 % від повної висоти) площини збільшуються, у верхній (від 70 % до 99 %) – зменшуються. Це дозволяє візуально створити ефект щільної крони з більш пухнастими гілками в центрі й тоншими – ближче до вершини.

Після цього запускається процес об’єднання площин. Замість того щоб залишати їх окремими об’єктами, вони формуються групами по 100 штук. Кожна група об’єднується через функцію join_objects(), яка всередині використовує оператор bpy.ops.object.join(), а після завершення генерації всі

площини збираються в єдину структуру, що дозволяє значно зменшити кількість об’єктів у сцені й пришвидшити майбутній рендеринг.

Лістинг 3.4 Внутрішній виклик об’єднання у функції join_objects():

```
bpy.ops.object.join()
bpy.context.object.name = name
```

На завершальному етапі усі частини дерева об’єднуються у фінальну модель. Саме вона надалі буде використовуватися в сцені або експортується як готовий об’єкт. Завдяки параметричному підходу дерево можна швидко змінювати, регулюючи його висоту, кількість гілок, густоту хвої й інші параметри без редагування геометрії вручну.

3.3 Інструкція користувача

Щоб скористатися адоном для генерації ялин, його потрібно спершу встановити у середовище Blender. Це треба зробити один раз і займає всього кілька хвилин, але дозволяє надалі запускати генерацію напряму з інтерфейсу без зайвих дій. Після встановлення адон інтегрується у вікно перегляду й додає окрему вкладку з початковим елементом керування. На цьому етапі у панелі відображається лише одна кнопка, яка служить для початку створення ялин за допомогою адону.

Для початку необхідно відкрити вікно налаштувань через обрання у верхньому меню пункту «Edit», а потім треба клікнути на «Preferences». У новому вікні, що відкриється, слід перейти до розділу «Add-ons». Він буде розташований у стовпчику зліва, серед інших параметрів персоналізації Blender.

У правому верхньому куті знаходиться маленька стрілка вниз, яка відкриває додаткове меню. Потрібно натиснути на неї, а далі обрати пункт

«Install from Disk...». Після цього відкриється вікно вибору файлу, де слід обрати необхідний файл скрипта адону. Як тільки файл буде обрано, адон з’явиться в загальному списку. Потім залишиться лише поставити галочку навпроти його назви для активації (рис. 3.1).

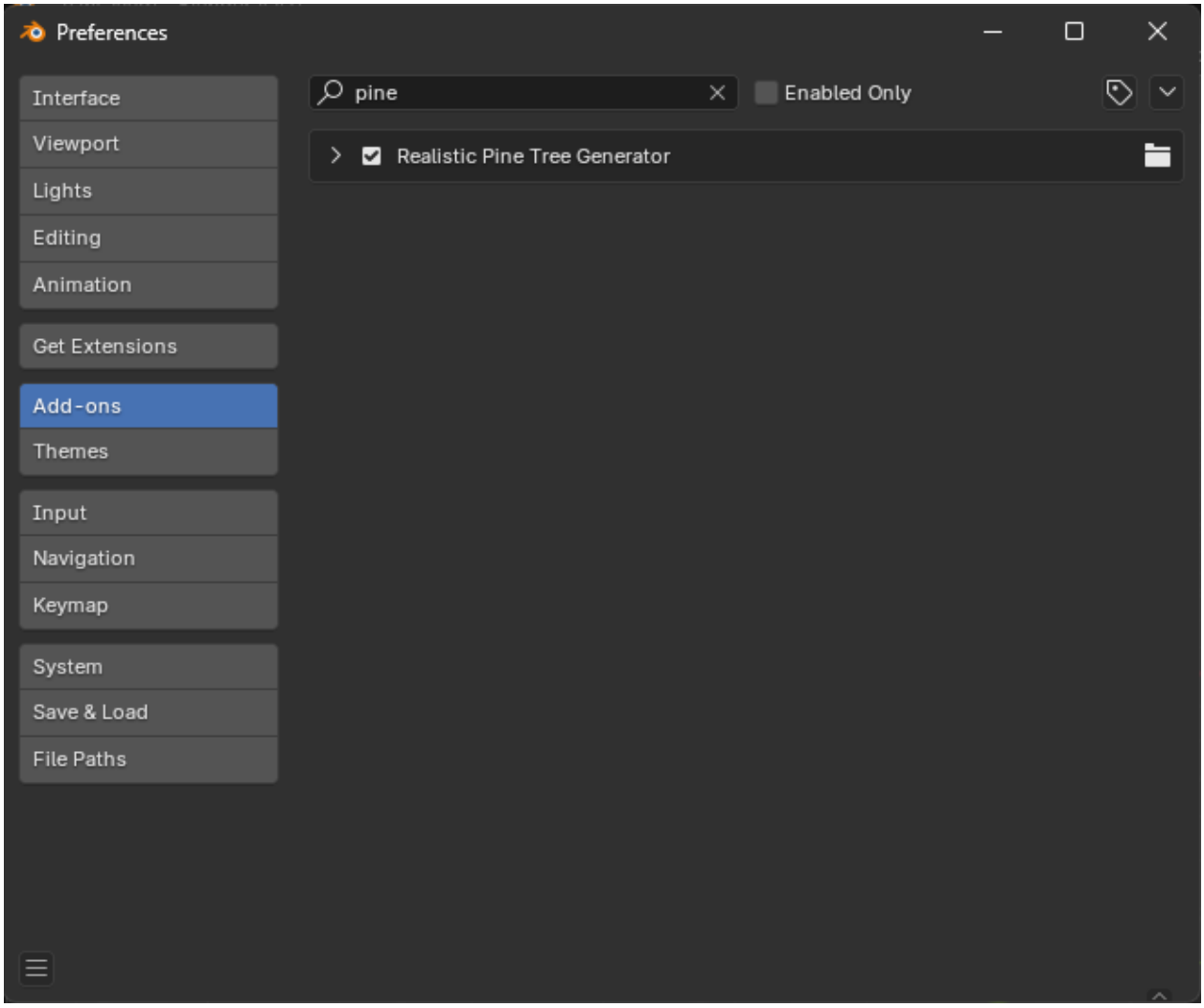

Рисунок 3.1 – Встановлення адону Realistic Pine Tree Generator у Blender 4.4.0

Після активації в інтерфейсі Blender з’являється додаткова вкладка під назвою «Pine Tree Gen». Вона розташована у правій панелі тривимірного вікна перегляду, яку можна викликати клавішею *N*. Всередині вкладки можна побачити кнопку «Generate Realistic Pine Tree» (рис. 3.2).

Рисунок 3.2 – Панель адону Realistic Pine Tree Generator у тривимірному вікні перегляду

Для того, щоб створити дерево, достатньо натиснути кнопку генерації. У сцені з’явиться готова модель ялини – з повноцінним стовбуром, гілками і площинами, що імітують хвою. Усі частини створюються автоматично, відповідно до заданих математичних формул. Це відбувається швидко, без затримок і без необхідності попередньо підготовлювати сцену (рис. 3.3).

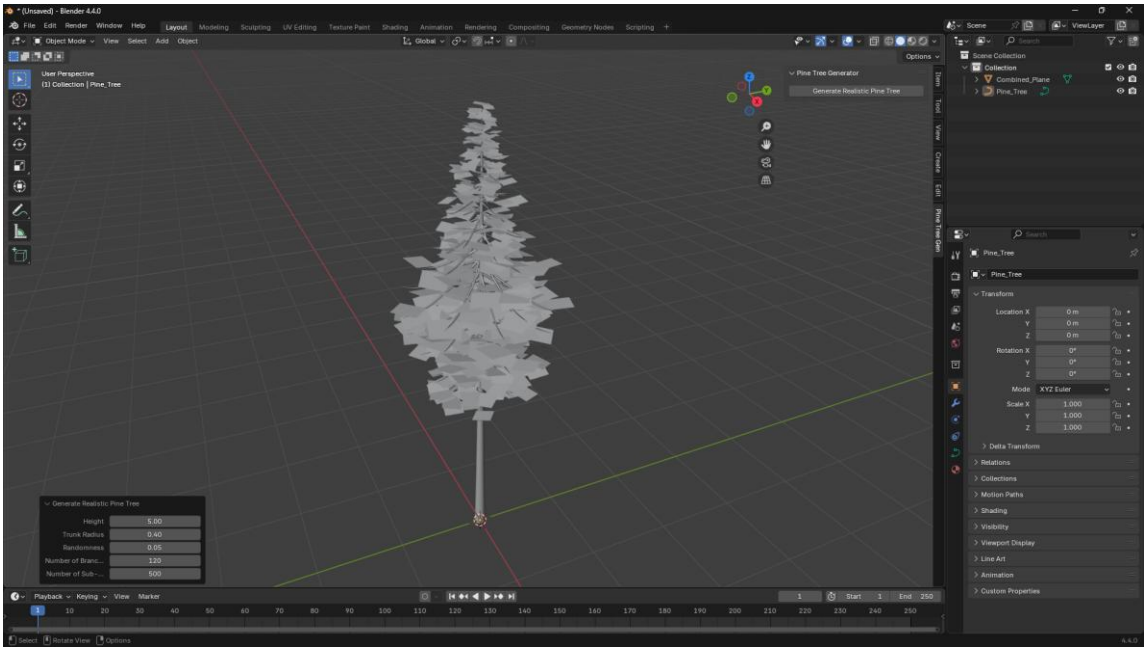

Рисунок 3.3 – Панель адону Realistic Pine Tree Generator у тривимірному вікні перегляду

За потреби можна одразу змінити параметри (висота дерева, товщина стовбура, кількість гілок, рівень випадковості та кількість площин) і нове дерево буде згенеровано з урахуванням оновлених значень. Завдяки цьому можна експериментувати з формою, розмірами та густотою моделі без жодного ручного редагування геометрії.

Для того, щоб побачити модель у готовому вигляді, потрібно лише увімкнути камеру, джерело світла та перейти до режиму рендерингу. Завдяки цьому можна оцінити, як виглядатиме дерево у фінальній сцені (рис. 3.4).

Рисунок 3.4 – Рендер результату генерації дерева

Отже, цей адон є легким у використанні, має можливість налаштовувати параметри прямо в сцені, що робить його корисним при створенні сцен з ялинами. Тому його можна використовувати, наприклад, для невеликих проєктів, ігор, анімацій чи візуалізації великих архітектурних композицій.

3.4 Візуалізація результату генерації

Після завершення генерації ялини усі її компоненти сформували повноцінну тривимірну модель. Однак лише геометрії недостатньо для досягнення переконливого результату. Щоб дерево сприймалося як частина реального середовища, потрібна додаткова обробка і дерева, і середовища. Для того, щоб модель виглядала природно, на неї накладаються матеріали та до сцени додається освітлення, після чого виконується фінальний рендеринг сцени.

Стовбур отримав процедурний матеріал кори, який містить базовий рельєф і варіації кольору, які відтворюють фактуру деревини. Для хвої було використано зображення гілки з прозорим фоном, накладене на площини. Такий підхід дозволяє зберегти деталізацію й текстуру без перевантаження геометрії. Завдяки цьому зберігся природний вигляд навіть при невеликій кількості полігонів.

Для візуалізації сцени застосовано HDRI, що імітує природне зовнішнє світло. Для цього було використано зображення Goegap Road з відкритої бібліотеки Poly Haven [22], яке забезпечує м’які тіні, хорошу видимість деталей і реалістичну глибину сцени. Камера була розташована під кутом, що дозволяє повністю оцінити структуру дерева з характерним силуетом ялини.

На рисунку 3.5 зображено один із отриманих рендерів моделі. Він демонструє, як виглядає дерево у готовому вигляді після застосування текстур, освітлення та рендерингу.

Рисунок 3.5 – Рендер однієї згенерованої ялини з матеріалами та освітленням

Для перевірки варіативності було згенеровано кілька дерев із різними параметрами – зміною висоти, кількості гілок і густоти хвої. Це дозволило оцінити, як змінюється силует моделі залежно від вхідних даних.

Завдяки цьому стало можливим візуально порівняти вплив зміни налаштувань і переконатися в стабільності генерації навіть при різних значеннях.

Результати такого рендеру наведено на рисунку 3.6.

Рисунок 3.6 – Встановлення адону Realistic Pine Tree Generator у Blender 4.4.0

Кожна з моделей була створена в межах однієї сцени без необхідності очищення або оновлення проєкту. Це стало можливим завдяки тому, що генерація виконується як окрема операція, а результат автоматично групується та перейменовується. Такий підхід дозволяє швидко створювати кілька варіантів без втрати контролю над структурою сцени. Таким чином, рендери демонструють не тільки коректність роботи адону, а також зручність його використання.

3.5 Моделювання архітектурних елементів

Основна частина сцени – замок Гоґвортс – моделювалася вручну. Для побудови використовувалися прості об’єкти, модифікатори й логіка, що дозволяє поступово складати складні форми з окремих елементів. Робота йшла поетапно: кожну вежу, стіну чи будівлю спершу було створено окремо, налаштовувано, а вже потім додано до загальної композиції. Такий підхід забезпечує гнучкість на кожному етапі, дозволяючи швидко вносити зміни або адаптувати частини сцени під загальний вигляд.

Першим прикладом є фрагмент внутрішнього дворика замку. Його модель починалася зі звичайного куба, з якого поступово витягувались архітектурні виступи, видалялися зайві площини і формувались аркові прорізи. Окремо до стіни додавались балки для вікон, які будувалися за допомогою кривих – це давало можливість створити правильну геометрію з мінімальними витратами часу. Ліворуч у композиції видно вертикальний об’єкт, що також починався з куба, але згодом був об’єднаний із рештою частин – так утворювався один цілісний модуль стіни (рис. 3.7).

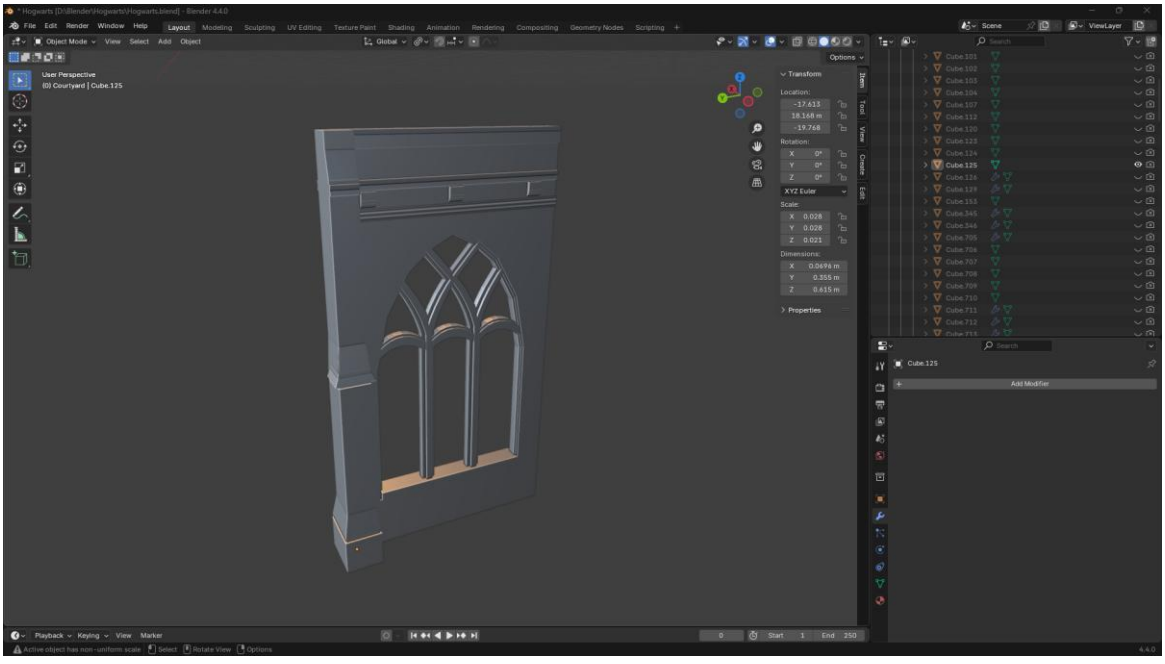

Рисунок 3.7 – Фрагмент внутрішнього дворика замку

Далі ці окремі частини об’єднувались у повноцінну структуру дворика. Щоб не повторювати одну й ту ж дію десятки разів, застосовувалися модифікатори масиву, які автоматично розмножували секції вздовж потрібної осі. Де потрібно було симетрію – використовувався модифікатор дзеркальності, що дозволяв будувати лише одну половину, а другу – автоматично відображати. В результаті формувався цілий двір із чіткою, структурою, що повторюється, який при цьому залишався легким у редагуванні й точним у симетрії (рис. 3.8).

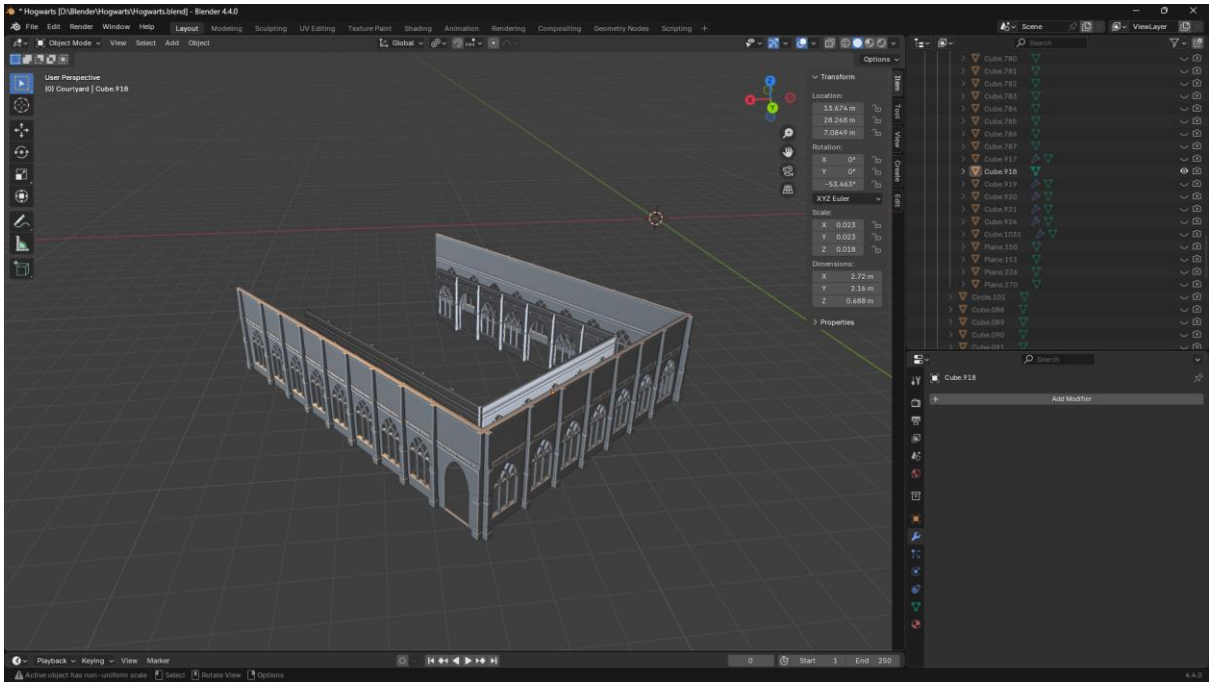

Рисунок 3.8 – Готова модель внутрішнього дворика

Найскладнішим етапом усього процесу було моделювання сходів, які вели від внутрішньої частини до верхнього рівня замку. Вони складалися з 21 окремої частини, кожна з яких створювалась вручну – на основі кубів і кіл із різними модифікаціями. Окремі елементи з’єднувались між собою за допомогою булевого модифікатора, який дозволяв правильно обробляти перетини. Це було важливо, щоб не залишалося геометричних помилок на стиках, і щоб структура сходів залишалася єдиною. На початковому етапі ці

елементи ще не мали точного розміщення і лише формували загальний напрямок (рис. 3.9).

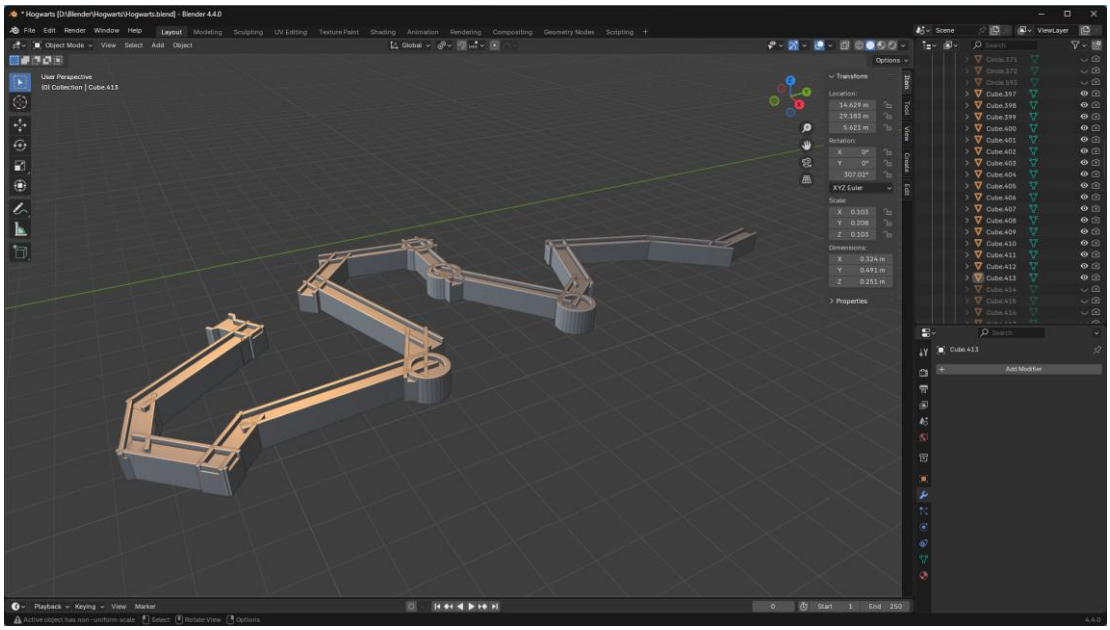

Рисунок 3.9 – Початкове компонування модулів сходів із окремих об’єктів

Після цього всі модулі поступово зміщувалися вниз, з урахуванням реальної висоти кожного елемента. Це дозволило досягти правильної послідовності та логіки розміщення частин сходів (рис. 3.10).

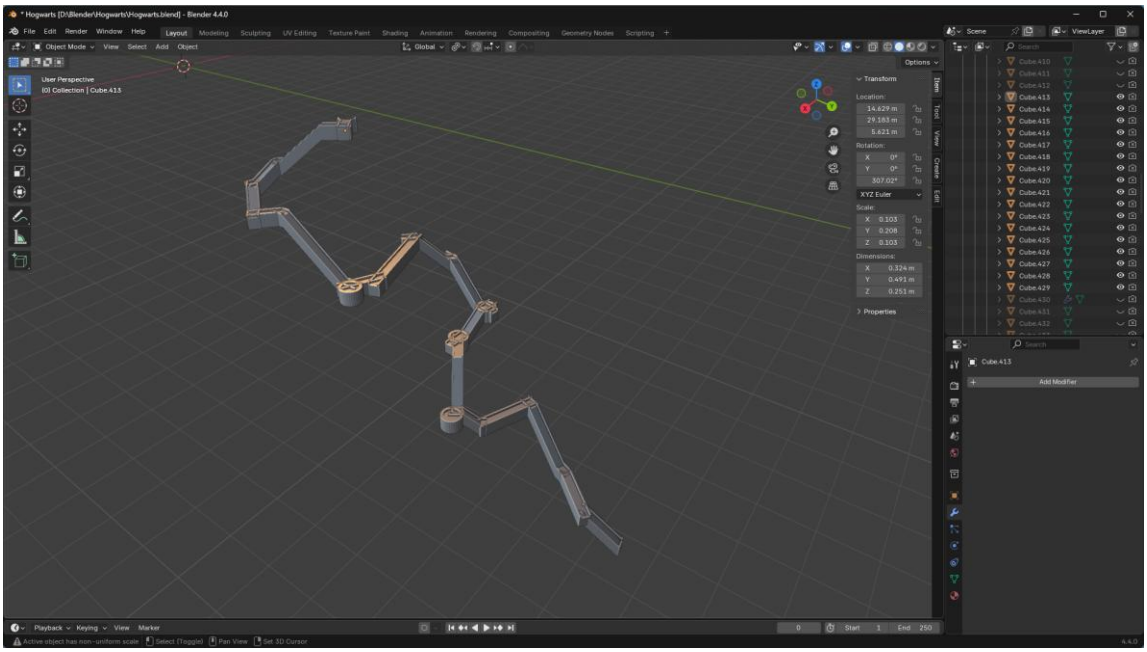

Рисунок 3.10 – Коригування висоти модулів сходів перед об’єднанням

У фінальному варіанті сходи були об’єднані в один об’єкт за допомогою булевого модифікатора, після чого вручну було видалено зайву геометрію та очищено внутрішні площини. Цей етап включав зміни в режимі редагування, під час якого були видалені зайві площини після об’єднання. Після цього до конструкції були додані окремі сходинки, що завершило формування структури. У такому вигляді весь об’єкт був готовий до подальшого використання (рис. 3.11).

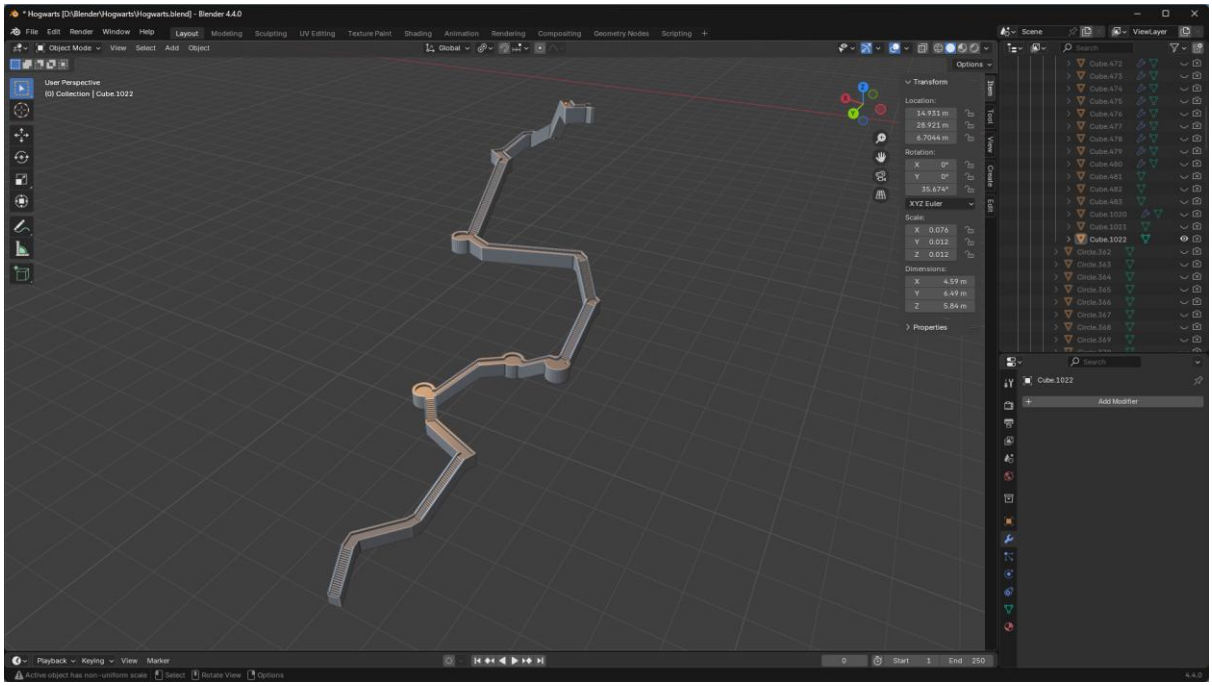

Рисунок 3.11 – Завершена модель сходів після очищення геометрії та додавання сходинок

Після завершення окремих елементів почалося поетапне об’єднання всіх частин замку. До фінальної сцени увійшли усі раніше підготовлені об’єкти: вежі, дворики, дахові елементи, допоміжні архітектурні частини. Кожен компонент попередньо був перевірений окремо, що дозволило уникнути помилок під час збирання і пришвидшити загальне об’єднання. Усі вони зібрані з урахуванням реальної композиції, симетрії та масштабів, наближених до макета Гоґвортса, який використовувався як приклад (рис. 3.12).

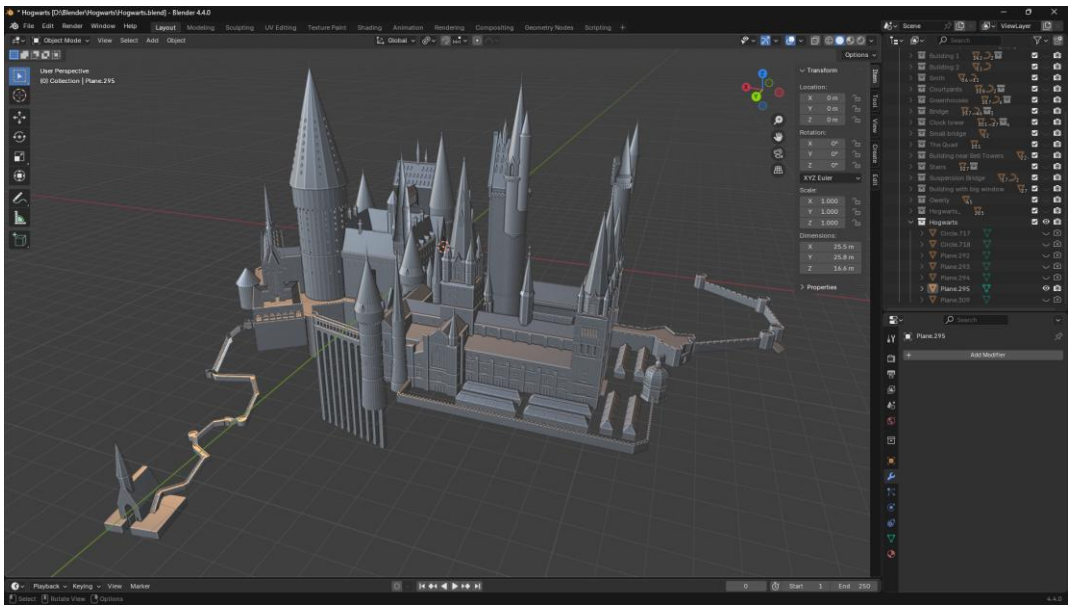

Рисунок 3.12 – Загальний вигляд сцени замку після об’єднання всіх архітектурних елементів

3.6 Створення ландшафту сцени

Після завершення моделювання архітектури постало завдання створити ландшафт, що її оточує. У цій частині сцени важливо було не просто додати умовні пагорби, рівнини чи схили, а відтворити середовище, яке буде максимально наближене до візуального образу у фільмах. За основу локального ландшафту було взято схему розміщення з офіційного видання Harry Potter: Page to Screen [23], де детально зображено розташування замку, рельєфу, доріжок і прилеглих територій. Візуальна копія цієї схеми була знайдена на сайті Hogwarts 4D [24], що дозволило працювати з орієнтирами, максимально наближеними до кінообразу. Завдяки цьому було зрозуміло, де має бути рівнина, а де – спуск [25].

Початковою формою для ландшафту слугував звичайний об’єкт площини. Щоб мати достатньо геометрії для подальшої роботи, до нього застосовувалася операція підрозділення – вона поділяє кожну грань на кілька менших, збільшуючи кількість полігонів. Наприклад, якщо площина

складається з одного квадрата, після одного підрозділення вона матиме чотири, після другого – шістнадцять, і так далі. У кожному кроці кількість полігонів зростає в чотири рази. У цьому проєкті було застосовано підрозділення 105 разів. У результаті площина мала досить щільну структуру для подальшого скульптингу з плавними деталями, але не була дуже перевантаженою.

На наступному етапі, у режимі редагування, вручну було витягнуто основні об’єми – височини, де має стояти замок, і загальні контури ландшафту. Це робилося для того, щоб спростити подальший скульптинг і вже на початку мати наближену форму. Власне формування деталей ландшафту відбувалося у режимі скульптингу. Подібна логіка нагадує задачу відновлення рельєфу за стереознімками, яка потребує точного розміщення відповідних координатних елементів [26]. Для цього використовувалися інструменти глиняних смуг та згладжування. Перший дозволяв швидко робити об’єм у вигляді м’яких мас, а за допомогою другого відбувалося згладжування переходів, щоб поверхня не виглядала рваною. У результаті було створено реалістичну місцевість навколо замку: із рівнинами, схилами, що ведуть до води, і природними підвищеннями, які відповідають географії у фільмі (рис. 3.13).

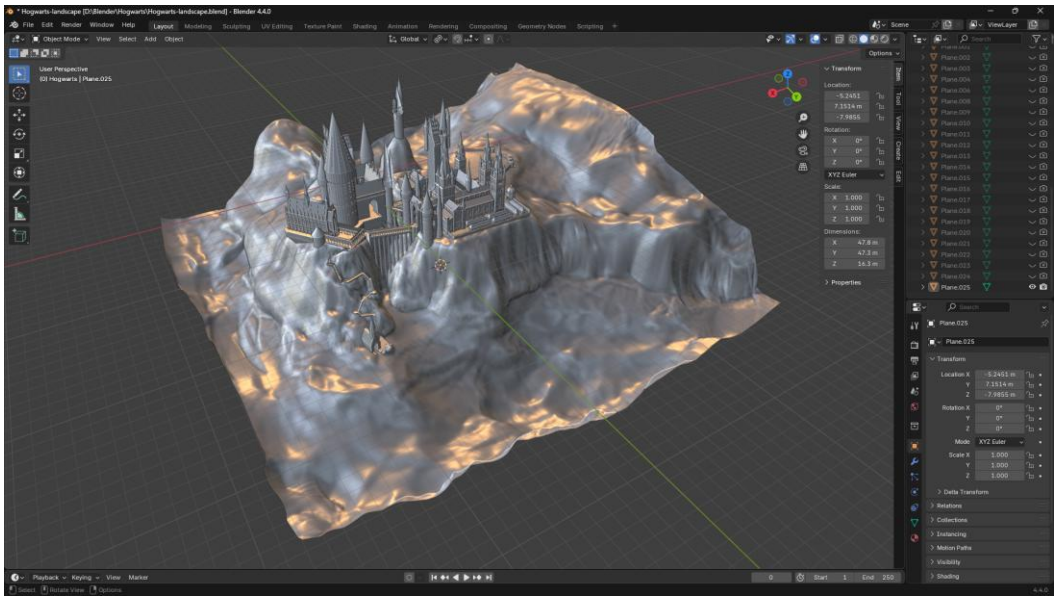

Рисунок 3.13 – Модель локального ландшафту навколо замку, створена у режимі скульптингу

Після завершення локальної зони моделювався загальний ландшафт сцени, тобто той, що оточує простір на далекому плані. Тут уже не використовувалися точні креслення – замість них орієнтиром слугували кадри з фільму, де можна побачити розташування гір, долин, озера. Створення такого пейзажу виконувалося через генеративний інструмент ландшафтів, вбудований у Blender. Усього було створено 11 окремих гірських масивів, кожен із яких мав свою форму, висоту, напрям. Вони розміщувалися довкола архітектурної частини, формуючи природне оточення й додаючи глибини сцені.

Рельєф для озера – що у сцені відповідає Чорному озеру – також формувався через інструмент ландшафтів [27]. Його параметри були підібрані так, щоб утворити заглиблення в центрі та плавний перехід до берегів. Геометрія цієї частини сцени була максимально простою, щоби полегшити рендер, але водночас достатньо складною, щоб не виглядати пласкою чи умовною (рис. 3.14).

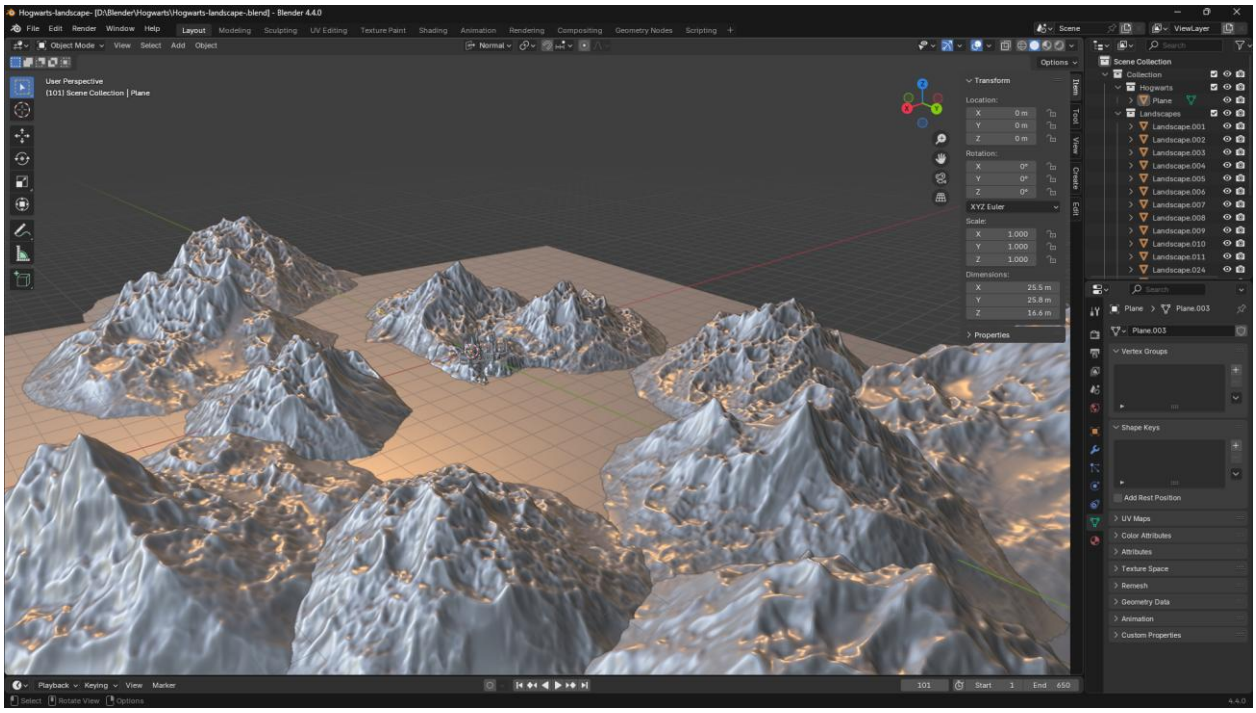

Рисунок 3.14 – Повний ландшафт сцени з гірським оточенням та рельєфом озера

Таким чином, уся місцевість сцени поєднувала точність для ближніх елементів, і умовність для віддалених. Це дозволило уникнути порожніх ділянок та створити повноцінне середовище, у якому замок виглядає природно і масштабно.

3.7 Створення матеріалів для архітектури й ландшафту

Щоб сцена справді ожила, однієї лише геометрії недостатньо. Поверхня має бути шорсткою або глянцевою, однотонною чи складною. Як це досягається – вже згадувалося раніше. Реалістичність значною мірою залежить саме від матеріалів, які відповідають за візуальні враження глядача. Нижче буде про те, як це було реалізовано безпосередньо в цьому проєкті.

Основна частина – замок і його архітектурні елементи – моделювалася під детальне спостереження. Камера часто наближалась до веж, дахів, мостів. Тому текстури повинні були бути гнучкими, масштабованими і без повторів. Було обрано процедурний підхід, без використання зовнішніх зображень. Це давало свободу, адже не треба було робити текстурну розгортку, а текстура автоматично підлаштовувалась під форму і не втрачала чіткості при зміні масштабу. Звичайні зображення могли б дати неправильне накладання або пікселізацію, особливо при сильному збільшенні.

Матеріал цегли будувався через вузол текстури цегляної кладки, у якому задавалися параметри ширини, висоти та шва. Колір модулювався за допомогою шумової текстури і коригувався через кольорову шкалу. У результаті кожна цеглина виглядала трохи по-своєму – як у справжній кладці. Додатково використовувалися два вузли імітації рельєфу, щоб надати об’єм як самим швам, так і загальній поверхні. Уся ця система збиралася у головний вузол матеріалу, що відповідає за візуальне відображення властивостей поверхні (рис. 3.15).

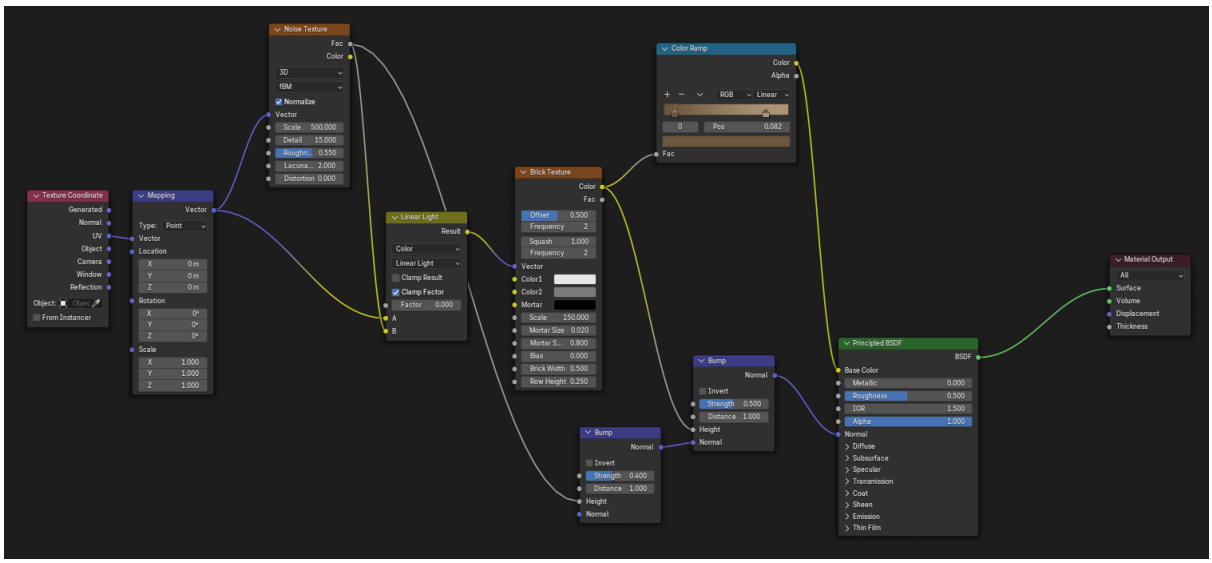

Рисунок 3.15 – Налаштування процедурного матеріалу цегляної кладки

Натомість ландшафт вимагає іншого підходу. Тут масштаби інші, і якщо на цеглину глядач може дивитись із пів метра, то рельєф схилу – здалеку. Через це важливо передати загальне враження від поверхні, а не дрібні деталі. У таких випадках найкраще працюють текстури, що імітують фізичну поведінку поверхні. Для цього використовувалися повні набори фізично достовірних текстур – готові карти кольору, шорсткості, мікрорельєфу, нормалей тощо. Як вже було зазначено раніше, такі матеріали забезпечують правильну взаємодію зі світлом і камерою, зберігаючи реалістичний вигляд навіть під час руху. Для створення матеріалу поверхні скелі було застосовано набір Rock 030 із сайту AmbientCG [28], а текстура трави Grass Lawn була взята з бібліотеки BlenderKit [29].

Але просто накласти зображення – замало. Адже може бути ситуація, коли присутній один камінь і зовсім немає трави. Тому треба, щоб поєднання між матеріалами мало природну межу, а ці дві поверхні з’являлися за висотою. Для цього у вузловому редакторі матеріалу використано логіку: чим нижче поверхня – тим більше трави, чим вище – тим більше каміння. Перехід між ними контролюється градієнтом за висотою, що створюється через координати геометрії та шкалу кольорів (рис. 3.16).

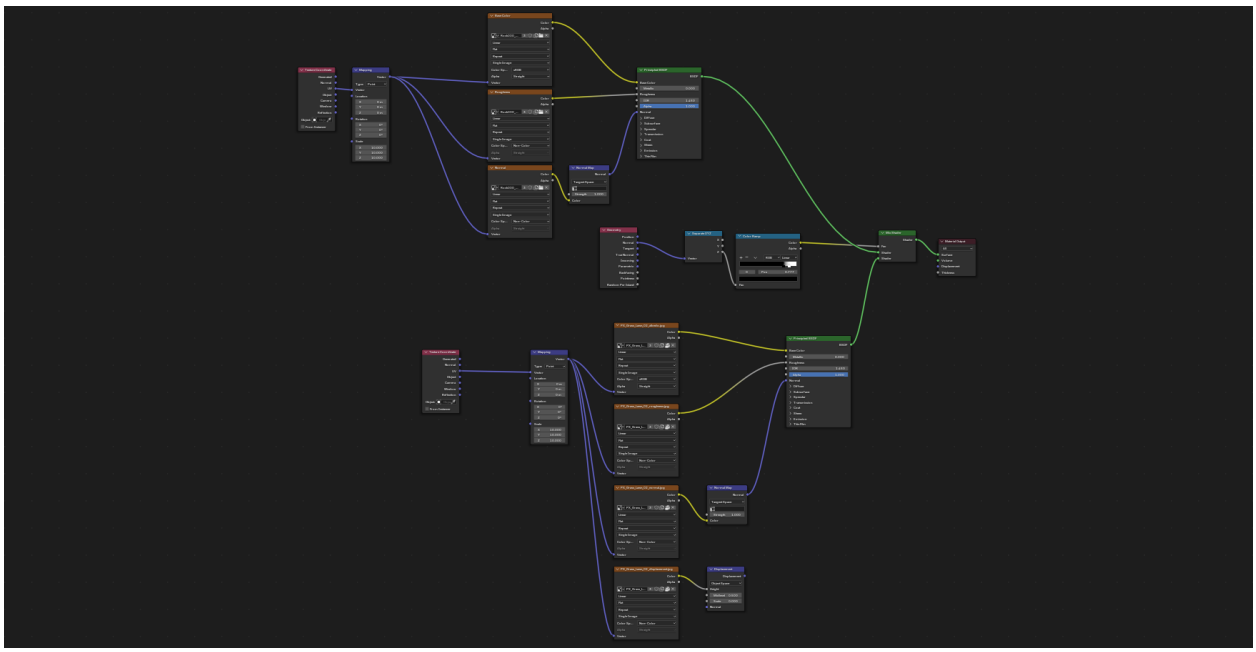

Рисунок 3.16 – Повна вузлова структура матеріалу ландшафту

Такий підхід дозволяє автоматично визначати, де має бути яка текстура, без жодного ручного розфарбовування або призначення вершинних груп. І головне – усе адаптивне: можна змінити форму рельєфу, і матеріал одразу підлаштується.

Також у сцені були й інші матеріали: скло – для вікон, мідна платина – для декоративних елементів на дахах, блакитна черепиця – для покрівлі, бетон, дошки, іржавий метал. Більшість з них створювалась на основі вузлів, інші – за допомогою наборів фізично достовірних текстур. Проте принцип був один: кожна поверхня повинна відповідати своїй функції, вписуватись у загальну стилістику й не виглядати випадково. Інакше – навіть найретельніше змодельована сцена втратить переконливість та достовірність.

Загалом, матеріали у сцені – це частина загальної атмосфери. Вони не існують окремо, а взаємодіють зі світлом, масштабом і контекстом. І саме в поєднанні – камінь стін, трава, скло у вікнах – усе це створює відчуття завершеної сцени. На зображенні нижче можна побачити загальний вигляд сцени вже з усіма застосованими матеріалами, безпосередньо у фінальному оточенні (рис. 3.17).

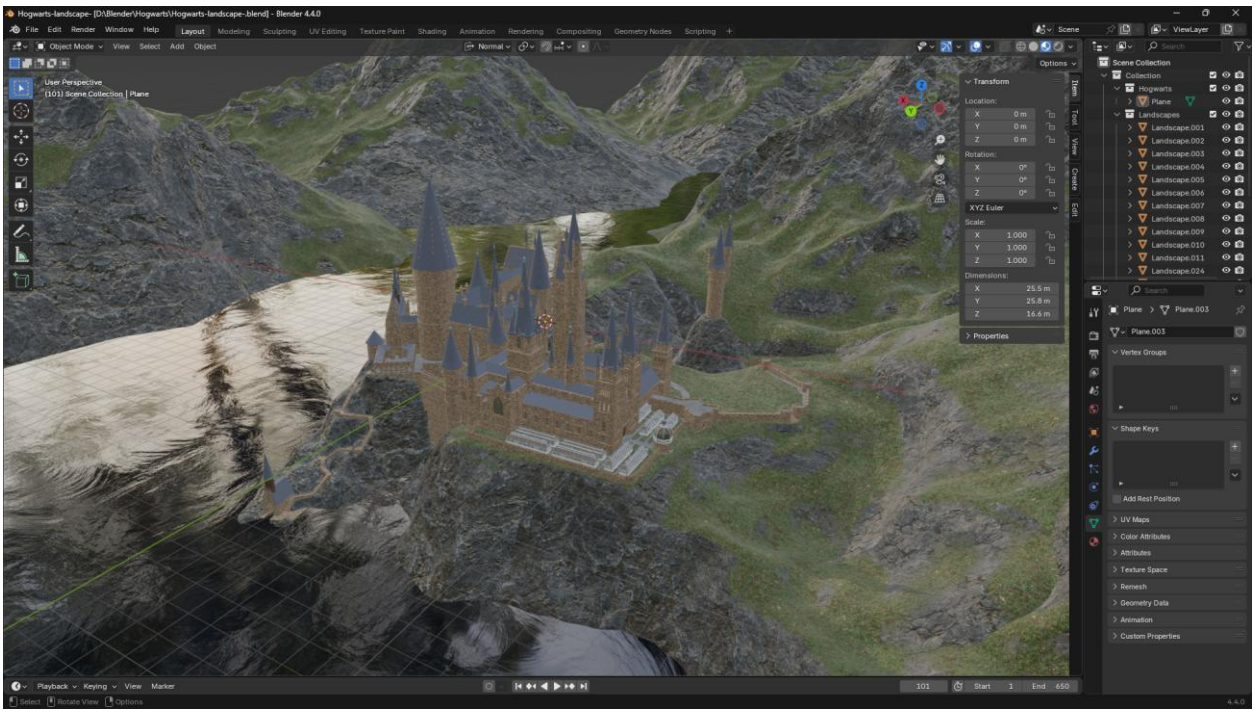

Рисунок 3.17 – Поєднання матеріалів у фінальній сцені

3.8 Розміщення ялин за допомогою Geometry Nodes

У сцені з великим ландшафтом вручну розставити всі ялини є нераціональним та довгим процесом. У випадку з проєктом, де сотні ялин мають органічно заповнити схили, долини та відкриті ділянки навколо замку, потрібне було рішення, яке дозволить зробити це швидко й гнучко. Тому для розміщення ялин було використано вузли геометрії – систему, яка дозволяє автоматично розставити об’єкти по поверхні, зберігаючи контроль над їх кількістю, положенням і виглядом.

У центрі всієї системи – модифікатор вузлів геометрії, який накладається на сам ландшафт. Саме він стає тією основою, куди додаються всі інші елементи. Ландшафт використовувався для того, щоб показати, де можуть з’явитися дерева. Наприклад, ялини не повинні зростати на камінні, на дорогах чи занадто крутих схилах. І щоб це забезпечити, вся логіка була зібрана у вузловій схемі (рис. 3.18).

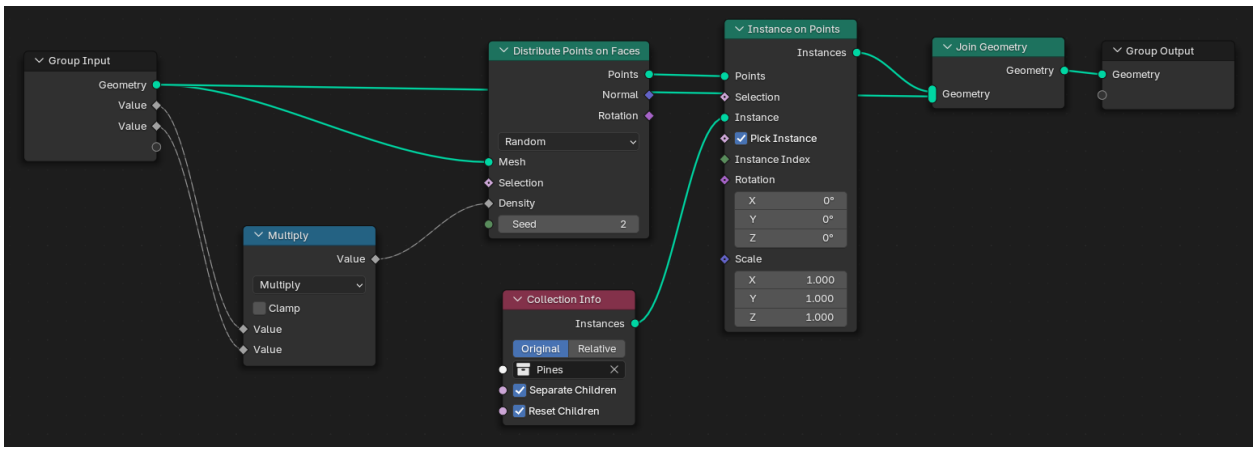

Рисунок 3.18 – Вузлова система автоматичного розміщення ялин

Перший крок – генерація випадкових точок на поверхні. Скільки точок, тобто потенційних дерев, буде – визначається вручну. Є спеціальний параметр щільності, який можна збільшувати або зменшувати за потреби. Це особливо зручно, коли потрібно адаптувати сцену під рендер: десь зробити більше ялин, десь – менше, або прибрати зайве. І не треба нічого рухати вручну – достатньо змінити значення у полі. Це значно спрощує підготовку великого середовища і дає змогу швидко знаходити потрібний баланс в розташуванні дерев. Та й параметр щільності можна змінити в будь-який момент, якщо дерев, наприклад, мало або забагато.

Далі в кожну точку вставляється об’єкт – у цьому випадку одна з ялин із попередньо зібраної колекції. Система випадково обирає, яка саме з п’яти варіантів буде вставлена. Це створює ефект природного середовища: дерева різні, не повторюються і не виглядають клонованими. Усі ці ялини – результат роботи адону, тож ідеально підходять до стилю сцени. А різноманітність моделей зменшує візуальну повторюваність, роблячи рендер більш реалістичним. До того ж, щоб усе працювало правильно, було увімкнено окреме позиціювання кожного дерева – без загальних масштабів чи зміщень.

Але найважливіше – це контроль. Дерева не повинні з’являтись там, де їх не може бути. І тут використовується ще один інструмент – карта ваги. З її допомогою позначається, де дозволено розміщення, а де – заборонено. Наприклад, ближче до підніжжя замку чи навколо мосту – зони без дерев. Усе

це задається вручну, пензлем, прямо на об’єкті. Процес малювання карти не потребує складних інструментів – достатньо базового пензля й розмітки кольором. А потім підключається до вузлів як маска (рис. 3.19).

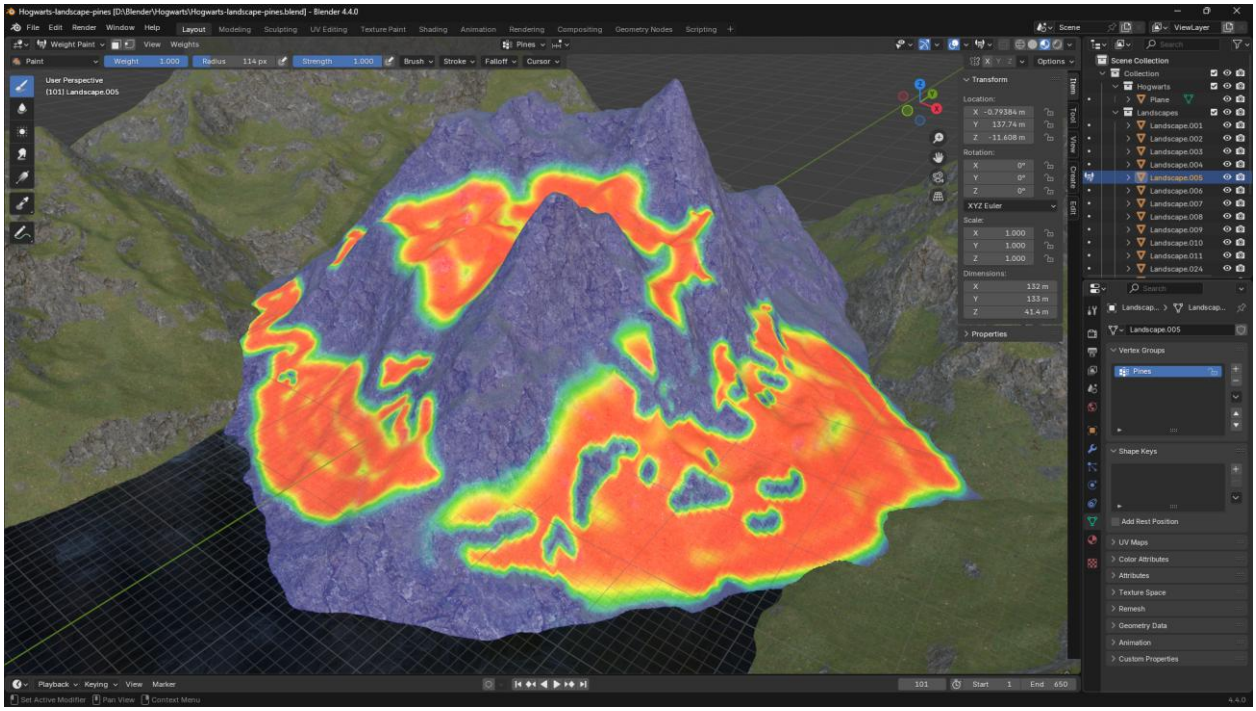

Рисунок 3.19 – Приклад карти ваги для розподілу ялин

У результаті було отримано точне, чисте, вивірене розміщення, яке при цьому не вимагає годин ручної роботи. І найкраще – це завжди можна змінити. У будь-який момент достатньо підправити пензлем маску, і весь розподіл миттєво оновиться. Це робить сцену динамічною у редагуванні та дозволяє швидко реагувати на зміну композиції чи завдання.

На завершення всі частини – і сам ландшафт, і вставлені дерева – є частиною одного об’єкту. Тобто немає тисячі окремих об’єктів, немає хаосу в списку сцен. Усе було зібрано в одному, але виглядає природно й точно. Об’єднання забезпечує не лише порядок, а й стабільність сцени під час рендерингу великих кадрів або анімацій. Це суттєво полегшує сам рендеринг, прискорює роботу й дозволяє зосередитися на художньому результаті, а не на технічних деталях (рис. 3.20).

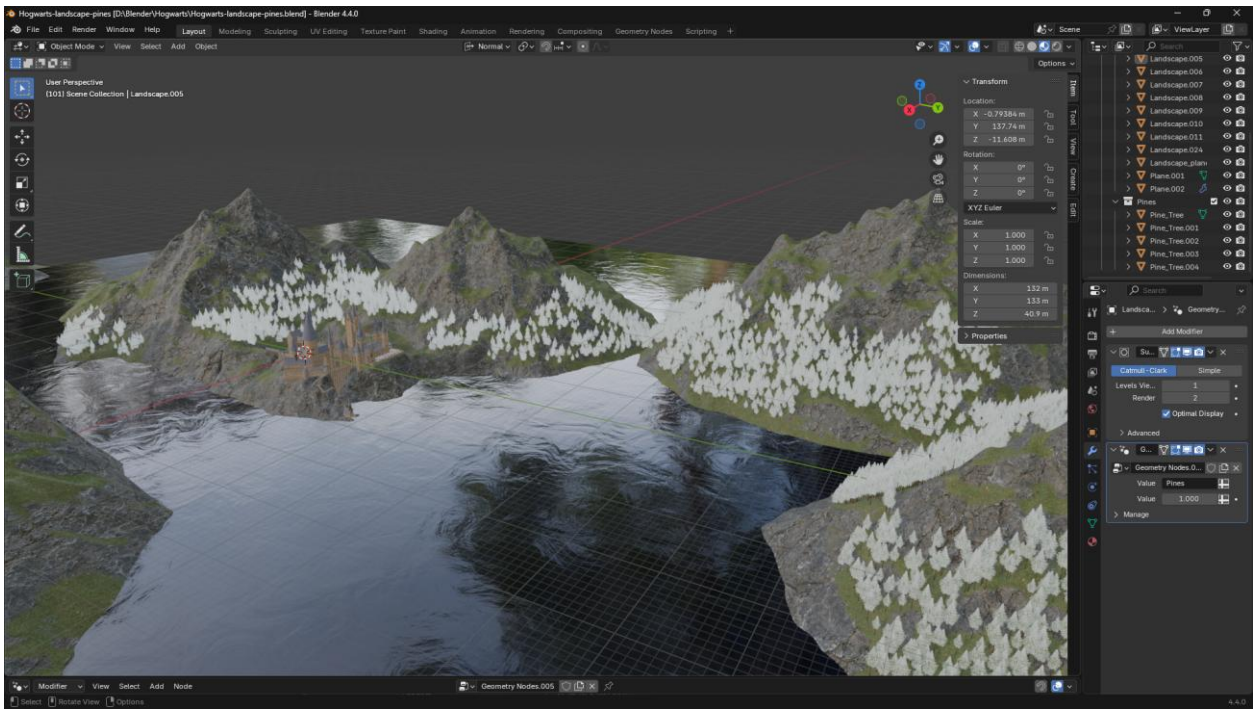

Рисунок 3.20 – Ялини, розміщені на ландшафті сцени

3.9 Створення анімації сцени

Щоб завершити роботу над сценою не лише візуально, а й композиційно, було створено повноцінну анімацію. Її завданням було не просто продемонструвати модель, а передати атмосферу – так, ніби камера справді летить навколо замку в різний час доби, спостерігаючи за його змінами.

Усього було використано п’ять окремих камер, кожна з яких відповідала за свою частину анімації. Перша повільно наближується до замку спереду, передаючи масштаб і монументальність будівлі. Друга – пролітає з фасаду та під дерев’яним мостом, відкриваючи нижній рівень сцени. Третя здійснює півколовий обліт навколо замку та залітає через внутрішній двір. Цей етап демонструє архітектурну композицію з різних боків. Четверта не рухається, але саме вона відповідає за зміну доби: від денного освітлення до нічного. П’ята – завершує композицію нічною сценою з плавним обльотом замку у підсвіченому вигляді.

Чотири з цих камер було прив’язано до кривих через вузол слідкування за кривою. Це дозволило не просто задати загальну траєкторію, а повністю контролювати рух: від швидкості до точного кута огляду. Ключові кадри задавалися вручну – як для координат, так і для обертання. Завдяки цьому вдалося досягти виразних переходів: в одному випадку – швидкого наближення, в іншому – м’якого уповільнення, яке давало змогу зупинитись на деталях. Для тонкого налаштування анімації було використано редактор графіків: там було зручно керувати кривими, пришвидшувати або згладжувати рух, налаштовувати інтервали. Це дало змогу, наприклад, сповільнити підйом камери в першій частині, щоб краще розкрити силует замку.

На рисунку 3.21 показано часову шкалу з ключовими кадрами третьої камери. Добре видно точки, в яких задавались нові положення або повороти камери.

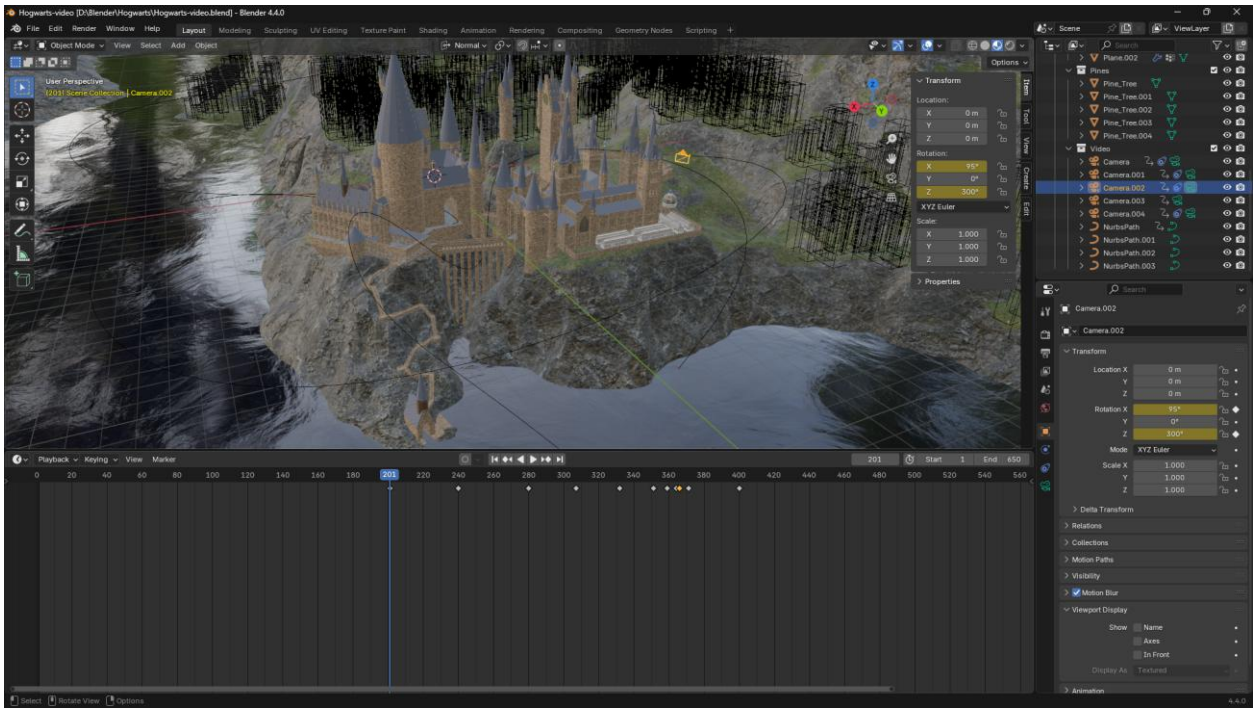

Рисунок 3.21 – Ключові кадри руху однієї з камер у редакторі часової шкали

Окремо увагу приділено налаштуванням рендерингу. Для досягнення високої якості сцени відео було рендерено у роздільній здатності 3840×2160 пікселів із 150 зразків для згладжування шуму. Це дозволило

зберегти деталізацію навіть при великій кількості тіней, світлових ефектів і дрібних об’єктів у кадрі. Важливо і те, що вся анімація базувалася на рушії Cycles, оскільки, як вже було зазначено раніше, саме він коректно відображає роботу світла, прозорості, емісії та взаємодії матеріалів.

Для створення ефекту зміни доби використовувалася HDRI Sunflowers Pure Sky з ресурсу Poly Haven [30]. Вона забезпечила реалістичне м’яке денне освітлення. Упродовж сцени його інтенсивність поступово зменшувалась, а HDRI також поступово поверталася, щоб зобразити рух неба протягом часу, а вікна замку починали світитись – цей ефект створювався за допомогою світловипромінювальних матеріалів, які активувались через ключі. Усе це разом формувало плавний перехід від раннього вечора до нічного середовища, без різких стрибків, зі збереженням глибини й атмосфери.

На рисунках 3.22–3.24 показано окремі кадри з анімації, які передають різні фази сцени. Один із моментів – це початкова частина, коли камера повільно підлітає до замку. Інший – коли зйомка переходить до тилу. Нарешті, останній кадр – нічна сцена, в якій замок освітлюється через вікна, а зовнішній простір лишається в темряві.

Рисунок 3.22 – Камера наближається до головного фасаду замку

Рисунок 3.23 – Вид з тилу замку

Рисунок 3.24 – Нічна сцена з підсвічуванням вікон

ВИСНОВКИ

У рамках кваліфікаційної роботи було створено повноцінну тривимірну сцену, яка об’єднує дві частини: архітектурну модель замку Гоґвортс і розроблений Python-адон для автоматичної генерації ялин. Обидві складові реалізовано в Blender, що дозволило зосередити всі етапи роботи в єдиному середовищі – від моделювання до анімації та фінального рендеру.

Модель замку було створено вручну, з нуля. Кожна деталь – від башт до дрібних архітектурних елементів – проєктувалася окремо, з урахуванням масштабу, симетрії та візуальної логіки. Основу геометрії складали прості об’єкти, які трансформувалися й доповнювалися модифікаторами. Уся архітектура поступово складалася в єдину композицію, зберігаючи стиль, композиційну побудову й атмосферу оригінальної локації. Для сцени також моделювався ландшафт із рельєфом, озером і гірськими масивами, а матеріали створювалися на основі процедурних вузлів і текстур, що забезпечували деталізацію без перевантаження моделі.

Python-адон був реалізований як окремий інструмент усередині Blender. Він дозволяє автоматично генерувати 3D-моделі ялин, використовуючи криві Безьє, математичні формули та параметри, які керують висотою, кутами гілок, розподілом хвої та масштабами площин. Основною метою було створення легких, але реалістичних дерев, які можна швидко змінювати й адаптувати під будь-яку сцену. Адон має графічний інтерфейс і дозволяє користувачеві генерувати об’єкти без потреби в додаткових налаштуваннях у коді. Автоматичне об’єднання площин і гілок дає змогу оптимізувати сцену, зменшити кількість полігонів і пришвидшити рендеринг.

У результаті було створено комплексну сцену, яка поєднує ручне та процедурне моделювання. Замок і оточення гармонійно взаємодіють, утворюючи завершене середовище. Було реалізовано анімацію з кількома камерами, зміною часу доби, денним і нічним освітленням, що дозволило повністю розкрити потенціал сцени.

Розроблений адон можна використовувати у візуалізаціях для архітектури, геймдизайну, анімаційних проєктів або концептуальних робіт. Його гнучкість і швидкість дозволяють створювати великі сцени без зайвих витрат часу. Те саме стосується і моделі замку: її можна адаптувати під інші задачі, розширювати або використовувати як базу для майбутніх робіт, наприклад, для фанатських візуалізацій або ігрових сцен. Обидва елементи – і замок, і адон – можна легко масштабувати, змінювати та інтегрувати у більші проєкти.

Таким чином, робота продемонструвала, як поєднання різних підходів у тривимірному моделюванні – ручного й автоматизованого – може дати не лише технічно якісний, а й візуально завершений і природний результат. Проєкт охоплює як творчі, так і технічні аспекти, і є прикладом того, як за допомогою одного програмного середовища можна реалізувати складну візуальну сцену з усіма її елементами.

Результати роботи апробовано у вигляді тез доповіді під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [31].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кобилін О. А., Творошенко І. С. (2021) Методи цифрової обробки зображень: навч. посібник. Харків: ХНУРЕ, 124 с.

2. Творошенко І. С., Кочура, Л. О. (2019) Про можливості геоінформаційних тривимірних моделей під час управління територіями. Економіко-управлінські та інформаційно-аналітичні новації в будівництві (Київ, 23-24 квітня 2019 р.). Київ, 2019. С. 43-44.

3. Mashtalir, S. V., & Nikolenko, O. V. (2024, April). Tree-Based Classification of the Technical Ukrainian Texts. In *International Conference on Computer Science, Engineering and Education Applications* (pp. 339-348). Cham: Springer Nature Switzerland.

4. Гороховатський, В. О., & Творошенко, І. С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

5. А. Шафроненко Є. Бодянський, І. Плісс (2022). Нечіткі методи інтелектуального аналізу даних. GlobeEdit.

6. Shafronenko, A. Y., Bodyanskiy, Y. V., & Holovin, O. O. (2023). CLUSTERIZATION OF DATA ARRAYS BASED ON THE MODIFIED GRAY WOLF ALGORITHM. *SCIENCE OF THE TOTAL ENVIRONMENT*, *873*, 73-79.

7. Гороховатський, В. О., & Гадецька, С. В. (2020). Статистичне оброблення та аналіз даних у структурних методах класифікації зображень.

8. Shafronenko, A. Y., & Rudenko, D. A. (2020). ONLINE RECURRENT METHOD OF CREDIBILISTIC FUZZY CLUSTERING. *BBK*, *91*, 37.

9. Sapling Tree Gen – Blender Manual. *Blender Documentation - blender.org*. URL: https://docs.blender.org/manual/uk/2.82/addons/add_curve/sapling.html (дата звернення 20.04.2025).

10. Create hit games with the power of AI with Ludo. *Ludo.ai*. URL: https://ludo.ai/ (дата звернення 26.04.2025).

11. Tripo AI - Create Your First 3D Model with Text and Image in Seconds. *Tripo AI - Create Your First 3D Model with Text and Image in Seconds*. URL: https://www.tripo3d.ai/ (дата звернення 27.04.2025).

12. Hogwarts 4D. *Hogwarts 4D*. URL: https://hogwarts4d.home.blog/ (дата звернення 10.04.2025).

13. Pietro Chiovaro. Recreating Hogwarts Castle in Blender, 2022. *YouTube*. URL: https://www.youtube.com/watch?v=h0o_WaL2YtI (дата звернення 10.04.2025).

14. Polygon Runway. Hogwarts Castle Part 1 in Blender 2.8 - Low Poly 3D Modeling Timelapse Tutorial, 2019. *YouTube*. URL: https://www.youtube.com/watch?v=Waz2lcbE5WM (дата звернення 10.04.2025).

15. Polygon Runway. Hogwarts Castle Part 2 in Blender 2.8 - Low Poly 3D Modeling Timelapse Tutorial, 2019. *YouTube*. URL: https://www.youtube.com/watch?v=BYZCtcQq37k (дата звернення 10.04.2025).

16. Tvoroshenko, I., & Almakaieva, A. (2020). Application of procedural generation of game content using software algorithms.

17. Buss S. R. 3D Computer Graphics: A Mathematical Introduction with OpenGL. Cambridge University Press, 2003.

18. Curve – Blender Manual. *Blender Documentation - blender.org*. URL: https://docs.blender.org/manual/en/3.3/modeling/curves/editing/curve.html (дата звернення 03.03.2025).

19. Творошенко, І. С. (2021). Технології прийняття рішень в інформаційних системах: навч. посібник. Харків: ХНУРЕ.

20. Кучеренко, Є. І., Творошенко, І. С., & Анопрієнко, Т. В. (2016). Моделювання та оцінювання станів складних об’єктів із застосуванням формальної логіки. *Системи обробки інформації*, (2), 76-82.

21. Yakovleva, O., Nebeský, L., & Liakhov, P. (2023). Research methods of texture image analysis to solve the texture search problem. In *Proceedings of the IV International Scientific and Practical Conference. Vienna, Austria* (pp. 252-261).

22. Goegap Road HDRI • Poly Haven. *Poly Haven*. URL: https://polyhaven.com/a/goegap_road (дата звернення 24.04.2025).

23. Harry Potter Page to Screen : Updated Edition: The Complete Filmmaking Journey. Harper Design, 2018. 540 p.

24. floor plan – Hogwarts 4D. *Hogwarts 4D*. URL: https://hogwarts4d.home.blog/tag/floor-plan/ (дата звернення 05.05.2025).

25. Творошенко, І. С. (2015). Конспект лекцій з дисципліни «Геоінформаційні системи в управлінні територіями»(для студентів 5 курсу денної форми навчання спеціальностей 7.08010105–Геоінформаційні системи та технології, 8.08010105–Геоінформаційні системи та технології та студентів 6 курсу заочної форми навчання спеціальності 7.08010105–Геоінформаційні системи та технології).

26. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

27. Творошенко І. С. (2019) Про особливості тривимірного подання просторових об’єктів засобами геоінформаційних технологій. Радіоелектроніка і молодь у XXI столітті: тези доповідей 23-го Міжнародного молодіжного форуму (Харків, 16-18 квітня 2019 р.). Харків: ХНУРЕ, 2019. Т. 7. С. 85-86.

28. Rock 030 on ambientCG. *ambientCG - Free Textures, HDRIs and Models*. URL: https://ambientcg.com/view?id=Rock030 (дата звернення: 07.05.2025).

29. Grass Lawn by PBRPX - Blender material | BlenderKit. *BlenderKit*. URL: https://www.blenderkit.com/asset-gallery-detail/2097a1c6-c8d8-4e5c-95cf-6030473391f0/ (дата звернення 07.05.2025).

30. Sunflowers (Pure Sky) HDRI • Poly Haven. *Poly Haven*. URL: https://polyhaven.com/a/sunflowers_puresky (дата звернення 10.05.2025).

31. Зінченко А. Я. (2025) Автоматична генерація реалістичних ялин за допомогою математичних моделей у середовищі Blender. *Радіоелектроніка і молодь у XXI столітті: тези доповідей 29-го Міжнародного молодіжного форуму (Харків, 16-19 квітня 2025 р.).* Харків: ХНУРЕ, 2025. Т. 7. С. 52-54.