

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:

**«Візуальні компоненти для ігрового додатку
"Forbidden Forest", що ґрунтується на механіках та
естетиці жанру Rogue-lite»**

Виконав студент групи КН-20-2
Шевченко Максим Олегович

Керівник роботи:
МАРЧУК Галина Вікторівна

Рецензент:
ПЕТРОСЯН Руслан Валерікович

Житомир – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук
Марина ГРАФ
«25» січня 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)

Здобувач вищої освіти: **ШЕВЧЕНКО Максим Олегович**
Керівник роботи: **МАРЧУК Галина Вікторівна**
Тема роботи: **«Візуальні компоненти для ігрового додатку "Forbidden Forest", що ґрунтується на механіках та естетиці жанру Rogue-lite»,**
затверджена наказом закладу вищої освіти від **«25» січня 2024 р., №22/с**
Термін здачі закінченої роботи: **«11» червня 2024 р.**
Вихідні дані роботи (зазначається предмет і об'єкт дослідження): використання професійного пакету для розробки комп'ютерної графіки.

Консультанти кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв
1	Галина МАРЧУК	01.09.2023	01.11.2023
2	Галина МАРЧУК	01.11.2023	01.04.2024
3	Галина МАРЧУК	01.04.2024	01.05.2024

Календарний план

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	2 лютого 2024 – 5 березня 2024	Виконано
2	Проектування структури розробки графіки	6 березня 2024 – 25 березня 2024	Виконано
3	Розробка базових об'єктів	26 березня 2024 – 26 квітня 2024	Виконано
4	Накладання текстур	27 квітня 2024 – 14 травня 2024	Виконано
5	Оптимізація 3D об'єктів	15 травня 2024 – 1 червня 2024	
6	Попередній захист	1 червня 2024	Виконано

Здобувач вищої освіти **Максим ШЕВЧЕНКО**
Керівник **Галина МАРЧУК**

РЕФЕРАТ

Кваліфікаційна робота включає в себе процес створення комп'ютерної графіки та супровідну пояснювальну записку. Остання містить у собі вступ, три розділи, висновки, список використаної літератури та додаток.

Текстова частина викладена на 69 сторінках друкованого тексту.

Пояснювальна записка має 28 сторінок додатків. Список використаних джерел містить 20 найменувань і займає 2 сторінки. В роботі наведено 70 рисунків. Загальний обсяг роботи – 93 сторінок.

Ключові слова: UNITY, BLENDER, SUBSTANCE, PAINTER, 3D, TEXTURE, LOD, ADDON, ROUGE-LITE, UV, TRI-PLANAR.

					КН.КР.Б – 122 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Візуальні компоненти для ігрового додатку "Forbidden Forest", що ґрунтується на механіках та естетиці жанру Rogue-lite Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.	М. О. ШЕВЧЕНКО						3	99
Керівник	Г. В. МАРЧУК					Житомирська політехніка, група КН-20-2		
Рецензент	Р. В. ПЕТРОСЯН							
Н. контр.								
Зав. каф.	М. С. ГРАФ							

ABSTRACT

The qualification work includes the process of creating computer graphics and an accompanying explanatory note. The latter contains an introduction, three sections, conclusions, a list of references, and an appendix.

The textual part is presented on 69 pages of printed text.

The explanatory note consists of 28 pages of appendices. The list of used sources includes 20 titles and occupies 2 pages. The work includes 70 figures. The total volume of the work is 93 pages.

Keywords: UNITY, BLENDER, SUBSTANCE, PAINTER, 3D, TEXTURE, LOD, ADDON, ROUGE-LITE, UV, TRI-PLANAR.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	9
1.1 Постановка задачі.....	9
1.2. Аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи.....	11
1.3 Концепт-документ (Concept Document).....	20
Висновки до першого розділу	24
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА КОМП'ЮТЕРНОЇ ГРАФІКИ	25
2.1 Візуалізація процесів розробки графічних елементів комп'ютерної гри	25
2.2 Розробка 3D-моделі для комп'ютерної гри.....	28
2.3 Створення графічного матеріалу для комп'ютерної гри	43
2.4 Реалізація програмного коду для оптимізації геометрії	55
Висновки до другого розділу	59
РОЗДІЛ 3. ОПИС РОБОТИ З ВІЗУАЛЬНИМИ КОМПОНЕНТАМИ ТА ЇХ ТЕСТУВАННЯ	60
3.1 Порядок оптимізації системи рівнів деталізації та експорту в ігровий рушій Unity	60
3.2 Порядок встановлення, налаштування та тестування в ігровому рушії Unity	63
Висновки до третього розділу	67
Висновки	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	69
ДОДАТКИ.....	71

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ	Програмне забезпечення
UV	Текстурні координати
Tri-planar	Триплощинна проекція
LOD	Level of Detail – рівні деталізації моделі від кращої до гіршої
UML	(Unified Modeling Language) – це уніфікована мова моделювання, яка використовується для візуалізації
WIP	Work In Progress – малюнок або рендер незавершеної роботи

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми. Rogue-lite - це жанр комп'ютерних ігор, що зародилася у 1980-х роках і названа на честь комп'ютерної гри "Rogue" 1980 року випуску. Жанр був створений з метою перенесення складної взаємодії та пригод настільних рольових ігор, як-от "Dungeons & Dragons", у цифровий світ відеоігор.

Характерною рисою Rogue-lite є використання процедурної генерації для створення рівнів, що забезпечує унікальність кожної комп'ютерної гри з випадковим розташуванням елементів, ворогами та подіями, дозволяючи гравцям кожного разу відкривати щось нове.

Перманентна смерть є ще однією важливою особливістю, коли гравець після програшу починає гру заново, втрачаючи всі набуті досягнення, що додає іграм високий рівень складності та вимагає від гравців обережності та стратегічного підходу.

Завдяки широкому спектру ігор, що представляють жанр, від класичних до сучасних версій з передовою графікою та геймплеєм, Rogue-lite здобув значну популярність. Останнім часом спостерігається збільшення зацікавленості в цьому жанрі, як серед геймерів, так і серед розробників, особливо серед індустрій, які пропонують свої унікальні інтерпретації, сприяючи різноманітності та інноваціям. Активна громада, що обмінюється досвідом та стратегіями, відіграє ключову роль у популяризації та розвитку жанру.

Жанр Rogue-lite відрізняється від інших жанрів відеоігор своєю глибиною геймплею. У багатьох Rogue-lite існує велика кількість різних механік, які взаємодіють між собою. Наприклад, гравці можуть поєднувати предмети для створення нових, брати участь у виготовленні та вдосконаленні знарядь, а також поєднувати закляття, щоб отримати унікальні ефекти. Це надає безмежних можливостей для експериментів та стратегічного мислення, що робить жанр особливим порівняно з іншими жанрами відеоігор.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		7

Метою кваліфікаційної роботи є створення комп'ютерної графіки для використання в комп'ютерній грі жанру Rougelite.

Об'єктом дослідження є використання технологій для створення комп'ютерної графіки для подальшого використання в комп'ютерній грі.

Предметом дослідження є розробка комп'ютерної графіки для ігрового додатку в жанрі Rouge-lite з назвою "Forbidden Forest" в спеціалізованих пакетах Blender для створення 3D-моделей та Substance 3D Painter для текстрування.

В рамках теми кваліфікаційної роботи було опубліковано тези: Шевченко М. О., Марчук Г.В., Аналіз програмного забезпечення для 3D моделювання від компанії Autodesk. IV Всеукраїнська науково-технічна конференція. Інформаційні комп'ютерні технології - 2021. 25-26 листопад 2021 р. м. Житомир.

Шевченко М. О., Марчук Д.К., Використання PBR-текстур в ігровій індустрії. XIII Міжнародна науково-технічна конференція. Інформаційні комп'ютерні технології - 2023. 30-31 березень 2023 р. м. Житомир.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Постановка задачі

Метою кваліфікаційної роботи є створення комп'ютерної графіки для комп'ютерної гри в жанрі Rougelite.

Основною метою розробки стилізованої комп'ютерної графіки в жанрі Rougelite є створення атмосферного та привабливого візуального середовища, яке допомагає зануритися глибше в ігровий всесвіт, це також включає в себе:

1. Сприйняття комп'ютерної гри. Стилiзована графіка забезпечує комфортне сприйняття під час тривалих ігрових сесій. Вона використовує прості, але чіткі візуальні елементи, які дозволяють гравцям швидко орієнтуватися в ігровому процесі.

2. Ностальгiчний візуальний стиль. Чимало Rogue-lite-ігор приймають ретро-стиль або піксельну графіку, що нагадує класичні комп'ютерні ігри, створюючи унікальний візуальний стиль та викликаючи ностальгiчні почуття.

3. Гнучкість графіки. Стилiзована графіка легко масштабується для різних роздільних здатностей, роблячи комп'ютерні ігри доступними для гравців з різними технічними можливостями.

4. Атмосфера комп'ютерної гри. Візуальний стиль ігор може створити особливу атмосферу, яка відображає тематику та настрої комп'ютерної гри, наприклад, темна піксельна графіка може підсилити відчуття містики та загадковості.

5. Візуальна привабливість: Графічний дизайн є ключовим для естетики комп'ютерної гри, роблячи її більш привабливою та залучаючи гравців завдяки унікальному візуальному досвіду.

У рамках кваліфікаційної роботи передбачається створення унікальних асетів для комп'ютерної гри, які включатимуть широкий спектр несхожих між собою ігрових елементів. Для досягнення поставленої задачі потрібно виконати такі етапи розробки:

1. Розробка ігрових асетів:

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		9

- a. Розробка концепції стилізованих асетів.
 - b. Розробка ігрових асетів по концептам.
 - c. Додавання різних варіацій ігрового об'єкта (низька, середня та висока версії).
 - d. Розробка системи рівнів деталізації.
2. Тестування розроблених асетів:
- a. Перевірка ігрових асетів на поєднання між собою
 - b. Перевірка, чи є можливість щось покращити
3. Поліпшення асетів:
- a. виправлення графічних помилок та усунення недоліків
 - b. Додавання нових ігрових асетів та їх варіацій
4. Список моделей на розробку (Додаток В):
- a. Розробити три версії дерева дуба. AW_Broadleaf_1 – найменша за розміром версія, AW_Broadleaf_2 – найбільша за розміром версія, AW_Broadleaf_3 – середня за розміром версія.
 - b. Розробити три версії дерева берези: AW_Birch_1 – найменша за розміром версія, AW_Birch_2 – середня за розміром версія, AW_Birch_3 – найбільша за розміром версія.
 - c. Розробити три версії мертвого дерева: AW_DeadTree_1 – найменша за розміром версія, AW_DeadTree_2 – найбільша за розміром версія, AW_DeadTree_3 – середня за розміром версія.
 - d. Розробити три версії дерева ялинки: AW_Fir_1 – найменша за розміром версія, AW_Fir_2 – середня за розміром версія, AW_Fir_3 – найбільша за розміром версія.
 - e. Розробити три версії трави: AW_Grass_1 – середня за розміром версія, AW_Grass_2 – найбільша за розміром версія, AW_Grass_3 – найменша за розміром версія.
 - f. Розробити три версії кущів: AW_Bush_1 – найменша за розміром версія, AW_Bush_2 – середня за розміром версія, AW_Ivy_1 – найбільша за розміром версія.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		10

- g. Розробити дві версії паркану: AW_Fence_1 – повна версія, AW_Fence_2 – часткова версія.
- h. Розробити шість версій шмату землі: AW_Terrain_1, AW_Terrain_2, AW_Terrain_3, AW_Terrain_4, AW_Terrain_5, AW_Terrain_6
- i. Розробити інструмент для швидкої розробки LOD-рівнів.

1.2. Аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи.

Для аналізу було обрано декілька ігор зі схожим стилем графіки.

«Borderlands 2» [1] є відомою відеогрою у жанрі шутера з елементами відкритого світу та рольових ігор, створеною Gearbox Software. Випущена у 2012 році, гра здобула шанувальників завдяки своєму відмінному дизайну та ігровому процесу.

Головна концепція комп'ютерної гри - це пригоди гравців у просторому світі Пандори, де вони можуть збирати різноманітну зброю, виконувати місії та битися з ворогами, включно з епічними босами (рис.1.1).



Рисунок 1.1 – Заставка комп'ютерної гри «Borderlands 2»

Гра «Borderlands 2» має унікальний художній стиль із затінками, що робить її візуально відмінною від інших шутерів. Цей стиль надає грі естетику коміксів із виразними контурами та яскравими кольорами.

Ключові аспекти графіки «Borderlands 2»:

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		11

1. Деталізовані текстури (Detailed Textures). Гра «Borderlands 2» відома своїми деталізованими текстурами, які додають реалістичності та глибини ігровому світу. Навіть здалеку об'єкти виглядають реалістично завдяки детальним текстурам, що дозволяє гравцям краще бачити навколишній світ (рис.1.2).

2. Яскраві кольори (Vibrant Colors). Однією з найпомітніших особливостей «Borderlands 2» є її яскрава, багата колірна палітра. Від глибоких барвистих пустель до яскравих космічних мегаполісів гра пропонує гравцям барвистий світ, який справляє враження життя та енергії (рис.1.3).



Рисунок 1.2 – Фрагмент комп'ютерної гри «Borderlands 2», приклад деталізації текстур

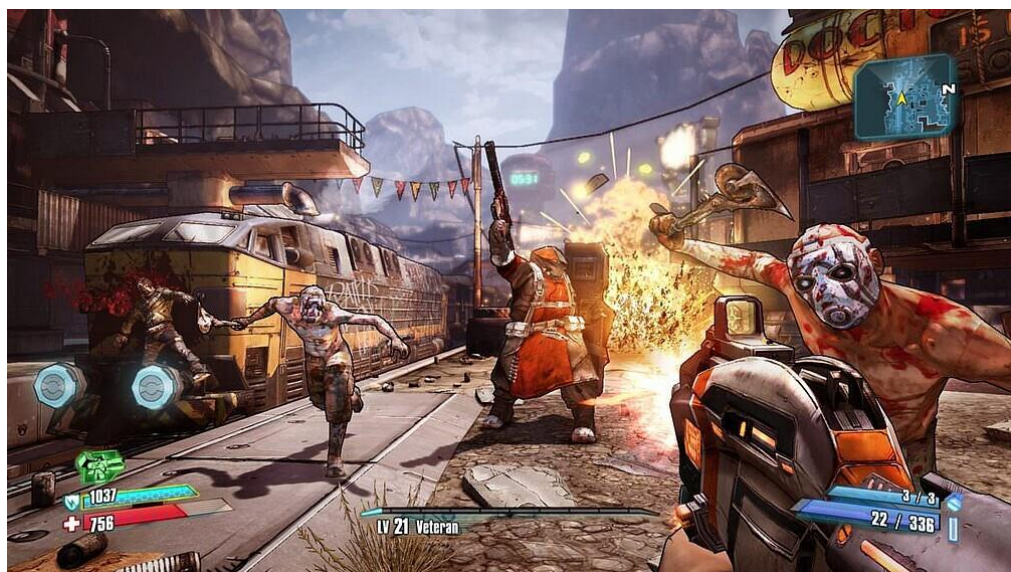


Рисунок 1.3 – фрагмент «Borderlands 2», приклад кольорової гамми

3. Перебільшені риси (Exaggerated Features). Персонажі та вороги в «Borderlands 2» мають перебільшені риси, які надають їм комічного

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		12

мультяшного вигляду. У них великі очі, виразні риси обличчя, розкішні костюми та аксесуари, які роблять їх унікальними та незабутніми. «Borderlands 2», безсумнівно, є культовою грою з візуально унікальним стилем, який поєднує в собі елементи коміксів і cel-shading (рис.1.4).

4. Сел-шейдинг (Cel-Shading). Ця техніка візуалізації використовується в «Borderlands 2» для створення живописного або мультиплікаційного вигляду. Додає текстурам глибину та виразність, завдяки чому об'єкти виглядають намальованими від руки. Це один із ключових елементів візуального стилю комп'ютерної гри (рис.1.5).



Рисунок 1.4 – Приклад реалізації текстур для обложки в комп'ютерній грі



Рисунок 1.5 – Приклад реалізації текстур

Однак, деякі гравці можуть вважати графіку недопрацьованою з кількох причин:

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		13

1. Пристрій з обмеженою функціональністю. Оскільки ця гра використовує відносно велику кількість деталей і спеціальних ефектів, вона може працювати менш ефективно на старих або менш потужних комп'ютерах і ігрових пристроях, що призведе до зниження продуктивності та частоти кадрів на екрані.

2. Ефекти затемнення. Деякі гравці вважають багато ефектів затемнення та відблисків неприємними, особливо в сценах із великою кількістю вибухів або швидким рухом камери. Це може призвести до поганої видимості та розмитості зображень для деяких користувачів.

3. Контрастність. Для деяких геймерів яскравість і насиченість кольорів можуть бути занадто високими або занадто насиченими, викликаючи дискомфорт або неприємні відчуття під час комп'ютерної гри.

Незважаючи на недоліки, для більшості гравців візуальний стиль та графіка «Borderlands 2» є одними з її найбільших переваг, які додають грі виразності та вражаючого вигляду.

«Roboquest» [2] - це швидкий і динамічний шутер від першої особи з елементами рогалика, розроблений студією RyseUp Studios. Гра вийшла в ранньому доступі у 2020 році та відзначається своєю швидкістю, динамічним геймплеєм та високоякісною графікою.

Основна мета комп'ютерної гри - битва з армією роботів у футуристичному світі. Гравці виступають у ролі операторів, які керують бойовими роботами та використовують різноманітну зброю та навички, щоб пройти через генеровані випадковим чином рівні, збирати ресурси та покращувати свої персонажів (рис.1.6).

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		14



Рисунок 1.6 – Заставка комп'ютерної гри «Roboquest»

Існує безліч інтерпретацій стилів графіки, особливо в ігровій індустрії. Проте, відповідно до візуальних матеріалів і відео, гра «Roboquest» може бути описана як Sci-Fi, або Футуристичний стиль. З високотехнологічними машинами, роботами та космічними елементами, графіка відображається у футуристичному ключі. Вона характеризується металевими поверхнями, яскравими кольорами, а також складними та деталізованими механізмами.

Ключові аспекти графіки у грі «Roboquest»:

1. Стиль і естетика. Графіка в «Roboquest» вражає своїм виразним стилем, який приваблює увагу гравців. Футуристичні елементи, яскраві кольори та динамічні об'єкти роблять світ комп'ютерної гри цікавим та привабливим (рис.1.7).



Рисунок 1.7 - Стиль і естетика комп'ютерної гри «Roboquest»

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				15
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Деталізація. Один із основних плюсів графіки у грі «Roboquest» - це висока деталізація об'єктів та оточення. Кожен елемент має багато дрібних деталей, що додає реалізму та поглиблює іммерсію гравця (рис.1.8).



Рисунок 1.8 – Деталізація текстур комп'ютерної гри «Roboquest»

3. Динамічність. Використання шейдерів та інших спеціальних ефектів дозволяє досягти вражаючої динаміки у графіці. Рухливість об'єктів, світлові ефекти та інші динамічні аспекти створюють відчуття живого світу (рис.1.9).



Рисунок 1.9 – Приклад динамічних текстур та спеціальних ефектів комп'ютерної гри «Roboquest»

Однак, деякі гравці можуть вважати графіку недопрацьованою з кількох причин:

1. Вимогливість. Якщо графіка в «Roboquest» і є однією з її переваг, то вона також може бути і недоліком. Для плавної роботи комп'ютерної гри потрібне потужне обладнання, і для деяких гравців це може стати проблемою.

2. Можливі проблеми оптимізації. Іноді гравці можуть зіткнутися з проблемами з оптимізацією комп'ютерної гри, особливо на менш потужних комп'ютерах або пристроях. Це може призвести до зниження кадрової частоти або відчуття нестабільності у геймплеї.

Загалом, гра «Roboquest» має вражаючу графіку, що додає глибини та реалізму її ігровому світу. Вона використовує сучасні технології для досягнення високого рівня візуального враження, але варто враховувати його вимоги до обладнання для оптимального досвіду комп'ютерної гри.

«Tiny Tina's Wonderlands» [3] - це фентезі-екшн гра з елементами рольової гри, розроблена компанією Gearbox Software. Гра є спін-оффом від серії ігор "Borderlands" і відзначається власним унікальним стилем та атмосферою.

У грі гравці відправляються у захоплюючий світ магії, чар і пригод, створений самою Tiny Tina - однією з головних персонажів серії "Borderlands". Гравці можуть насолоджуватися швидким та захоплюючим геймплеєм, виконуючи завдання, збираючи зброю та розвиваючи свого персонажа (рис.1.10).



Рисунок 1.10 – Заставка комп'ютерної гри «Tiny Tina's Wonderlands»

Ключові аспекти графіки «Tiny Tina's Wonderlands»:

1. Візуальний стиль. «Tiny Tina's Wonderlands» вражає своїм унікальним, казковим візуальним стилем. Гра використовує яскраві кольори, м'які форми та анімацію, щоб створити казковий світ, наповнений чарівністю і фантазією (рис.1.11).

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		17



Рисунок 1.11 - Візуальний стиль комп'ютерної гри «Tiny Tina's Wonderlands»

2. Деталізація. Ігровий світ "Tiny Tina's Wonderlands" дуже деталізований. Кожен елемент, починаючи від персонажів і закінчуючи оточенням, має багато дрібних деталей, які роблять гру більш живою та захоплюючою (рис.1.12).

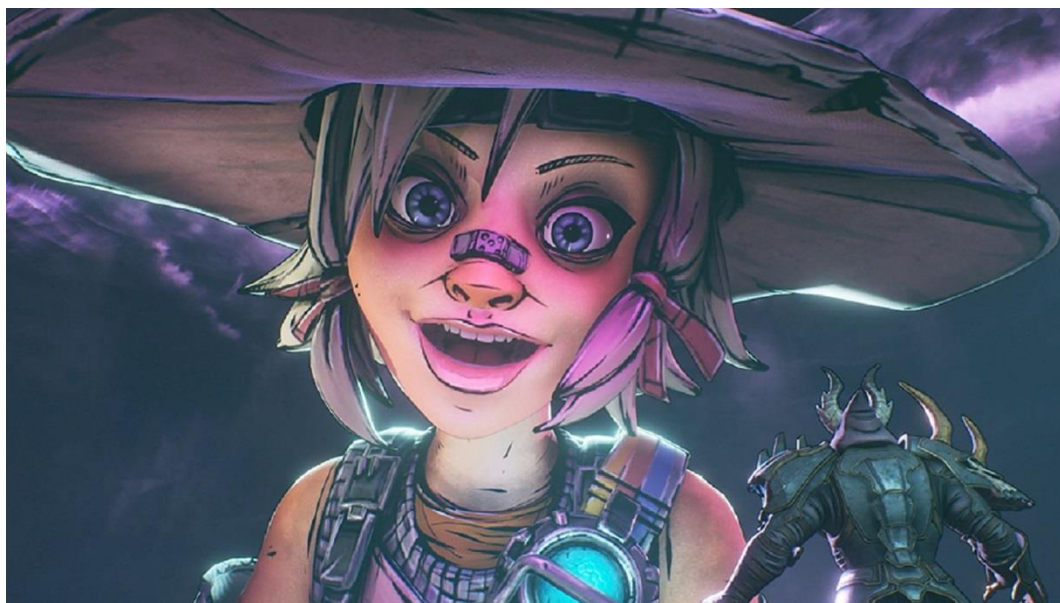


Рисунок 1.12 – Деталізація текстур комп'ютерної гри «Tiny Tina's Wonderlands»

3. Спеціальні ефекти. Гра використовує різноманітні спеціальні ефекти, такі як магичні вибухи, блискавичні частинки та вогняні вибухи, щоб зробити битви і пригоди у грі більш епічними та захоплюючими (рис.1.13).

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		18



Рисунок 1.13 – Деталізація графічних елементів комп'ютерної гри «Tiny Tina's Wonderlands»

Однак, деякі гравці можуть вважати графіку недопрацьованою з кількох причин:

1. Потреба у потужному обладнанні: Як і в багатьох сучасних іграх, «Tiny Tina's Wonderlands» може бути вимогливою до обладнання, що може стати проблемою для гравців з менш потужними комп'ютерами або консолями.

2. Можливість втоми для гравця. Хоча гра пропонує різноманіття в візуальному стилі, але за деяким часом, гравці можуть відчувати монотонність або втомитися від постійних яскравих кольорів та спецефектів.

Отже, було вирішено створити власну гру з стилізованою графікою. Важливо щоб гра відповідала таким графічним критеріям:

1. Візуальний стиль. Гра повинна мати чіткий, різноманітний та привабливий візуальний стиль, який відтворюється в унікальних арт-стилях, кольорних схемах та дизайні персонажів та оточення.

2. Естетика. Гра повинна мати привабливу естетику, яка привертає увагу гравців та залишається в їхній пам'яті. Це може включати в себе цікавий арт-дизайн, веселих або кумедних персонажів та унікальні фони та об'єкти.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		19

3. Сприйняття. Стилізований візуальний стиль повинен бути легко сприйманий гравцями, допомагаючи їм швидко розуміти світ комп'ютерної гри та його правила.

4. Унікальність. Хоча гра може використовувати стилізований візуальний стиль, важливо, щоб він був унікальним і відрізнявся від інших ігор, пропонуючи щось особливе та неповторне.

5. Відповідність тематиці. Візуальний стиль повинен підходити до тематики комп'ютерної гри та підкреслювати її ключові аспекти, створюючи атмосферу, яка відповідає концепції комп'ютерної гри.

1.3 Концепт-документ (Concept Document)

Жанр і цільова аудиторія.

Жанр – Rougelite.

У грі буде декілька рівнів складності, що зацікавить людей різного віку.

Легка складність розрахована які тільки починають приєднуватися до цього жанру ігор.

Середня – на людей які вже мали ігровий досвід в цьому жанрі.

Складна буде цікава людям які постійно грають в комп'ютерні ігри цього жанру.

Основні особливості комп'ютерної гри.

Гра буде вирізнятися серед інших авторським рівнем графіки, унікальним рівнем атмосфери, декількома рівнями складності та системою вибору предметів.

Геймплей.

Головний персонаж з'являється в барі, гравець може підійти персонажів до кожного з них та завести діалог, також гравець може підійти до дошки з завданнями та вибрати на якого ворога він хоче розпочати полювання. Після того як користувач вибрав завдання він може вийти з таверни та розпочати основний геймплей.

Гравець заходить на перший рівень та починає проходити рівні які при

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		20

кожному новому проходженні генеруються по новому. В кінці проходження арен буде знаходитися фінальна локація з боссом на якого велося полювання. Після перемоги над боссом гравцеві дається унікальний предмет в вигляді зброї босса або інший предмет нижчої якості.

Сюжет.

З давніх часів існував орден алхіміків, відомий своєю мудрістю та жагою до знань. Орден був створений людьми з усіх земель задля розвитку науки та служив обранцями миру.

Одного дня орден зробив відкриття що магія реальна, але не досяжна для звичайних людей. Натхненні амбіціями та жагою знань алхіміки віднайшли шлях як отримати бажане.

Під час проведення ритуалу сталася непередбачувана подія.

Розрив який стався під час проведення експерименту, вивів у світ нових невідомих істот та катастрофи по усьому світу. Мерці відроджувались, а загадкові істоти підкорювали собі землі, несучи за собою смерть та хаос. Серед тих хто залишився в новому світі, з'явилися особливі. Їм снилися сноведіння в яких, володарі потайбічного світу передавали їм дари чи то магії чи то надзвичайних особливостей. Вони були обрані для протистояння новим загрозам та вносили невизначеність у власне майбутнє через постійну загрозу своєму життю.

Хоча орден заплатив велику ціну за свої експерименти та жаги до знань, вони спробували обмежити поширення впливу потойбічного світу виконавши передсмертний ритуал. Відтоді світ став ареною де особливі мають протистояти новим змінам та мусять випробувати себе в ролі захисників від монстрів, мерців та невідомих сил, зберігаючи рівновагу між світами.

На шляху їх чекали тяжкі випробування, багато могутніх ворогів. Чи вдасться їм відновити рівновагу та знайти спосіб закрити розрив, відкритий своєю жадобою до магії?

Платформа.

Планується, що гра вийде тільки на РС-платформу.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		21

1.4 Обґрунтування вибору інструментальних засобів.

Популярні на даний час інструменти розробки комп'ютерної графіки це Blender, Autodesk Maya, Autodesk 3DS Max та програма для текстуровання Substance 3D Painter та інші.

Було проведено аналіз двох популярних програм для створення комп'ютерної графіки, Autodesk 3DS Max та Blender. Обидві програми є потужними інструментами для створення 3D-моделей, анімацій та візуалізації. Однак вони мають різні переваги та недоліки.

Переваги пакету для створення комп'ютерної графіки Autodesk 3DS Max:

1. Професійний інструмент. 3ds Max є стандартом у багатьох галузях, таких як архітектура, кіно та розробка комп'ютерних ігор.
2. Широкий функціонал. Програма має величезний набір інструментів для моделювання, анімації, візуалізації та обробки звуку.
3. Підтримка плагінів. Є багато розширень та плагінів, що розширюють функціонал програми, дозволяючи налаштовувати її під конкретні потреби користувача.

Недоліки пакету для створення комп'ютерної графіки Autodesk 3DS Max:

1. Вартість. 3ds Max є комерційною програмою і вимагає покупки ліцензії або підписки, що може бути дорогим для незалежних користувачів або початківців.
2. Системні вимоги. Для плавної роботи програми потрібне потужне обладнання, що може бути проблемою для користувачів з обмеженими ресурсами.

Переваги пакету для створення комп'ютерної графіки Blender:

1. Безкоштовність. Blender є відкритою програмою з вільною ліцензією GNU GPL, тому вона доступна для завантаження та використання безкоштовно.
2. Спільнота та підтримка. Blender має велику та активну спільноту користувачів, яка регулярно розробляє нові функції, плагіни та навчальний матеріал.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		22

3. Платформонезалежність. Blender підтримується на різних операційних системах, включаючи Windows, macOS та Linux.

Недоліки пакету для створення комп'ютерної графіки Blender:

1. Навчання та крутий старт. Blender має крутіший криву навчання порівняно з 3ds Max, оскільки інтерфейс та робочий процес можуть бути важкими для новачків.

2. Менша поширеність у деяких галузях. Незважаючи на широкий функціонал, у деяких галузях, таких як архітектура та деякі галузі ігрової розробки, 3ds Max залишається більш популярнішим інструментом розробки комп'ютерної графіки.

Щодо спільного у них, обидві програми надають користувачам можливість створювати вражаючі 3D-моделі, робити анімацію та візуалізацію, а також вони обидві активно розвиваються і підтримуються їхніми розробниками та спільнотами.

Хоча існують інші програми для текстуровання 3D-моделей, які включають Mari, Quixel Mixer, Substance Designer та вбудований Blender Texture Paint, програма Substance Painter виділяється як кращий вибір завдяки своїй потужній функціональності, інтуїтивно зрозумілому інтерфейсу та широкому спектру доступних матеріалів і текстур.

Було обрано Blender та Substance 3D Painter:

1. Легкий та зрозумілий для користувача інтерфейс.

2. Багатофункціональність. Blender дозволяє інтегрувати нові інструменти для розробки моделей, а Substance 3D Painter має досить широкий вибір інструментів для досягнення досить приємної стилізованої графіки.

3. Велика спільнота. по програмам Blender та Substance 3D Painter досить багато відеоуроків в інтернеті, також користувачі цих програм постійно вигадують нові інструменти для пришвидшення процесу розробки.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Висновки до першого розділу

Було встановлено ціль та сформульовано завдання для розробки. Проведено аналіз трьох відомих ігор у жанрі Rougelike (Borderlands 2, Roboquest, Tiny Tina's Wonderlands), виявлено їх сильні та слабкі сторони. Визначено необхідний графічний стиль для комп'ютерної гри.

Розроблено концептуальний документ, який окреслює жанр, цільову аудиторію, ключові особливості, геймплей, сюжет та платформу. Було проведено аналіз інструментів, таких як Blender та Substance 3D Painter. Було обґрунтовано їх використання для розробки моделей та графіки які використовуються в комп'ютерній грі з урахуванням їх переваг та недоліків.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		24

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА КОМП'ЮТЕРНОЇ ГРАФІКИ

2.1 Візуалізація процесів розробки графічних елементів комп'ютерної гри

Діаграми UML (англ. Unified Modeling Language) є універсальною мовою, яка зрозуміла розробникам, аналітикам та замовникам. Існують різні типи діаграм UML, кожна з яких використовується для моделювання певного аспекту. Діаграми станів описують поведінку об'єктів у системі або поведінку самої системи, або стани розробки графічних об'єктів. Діаграма станів (рис. 2.1) представляє етапи розробки концепції графіки для комп'ютерної гри.

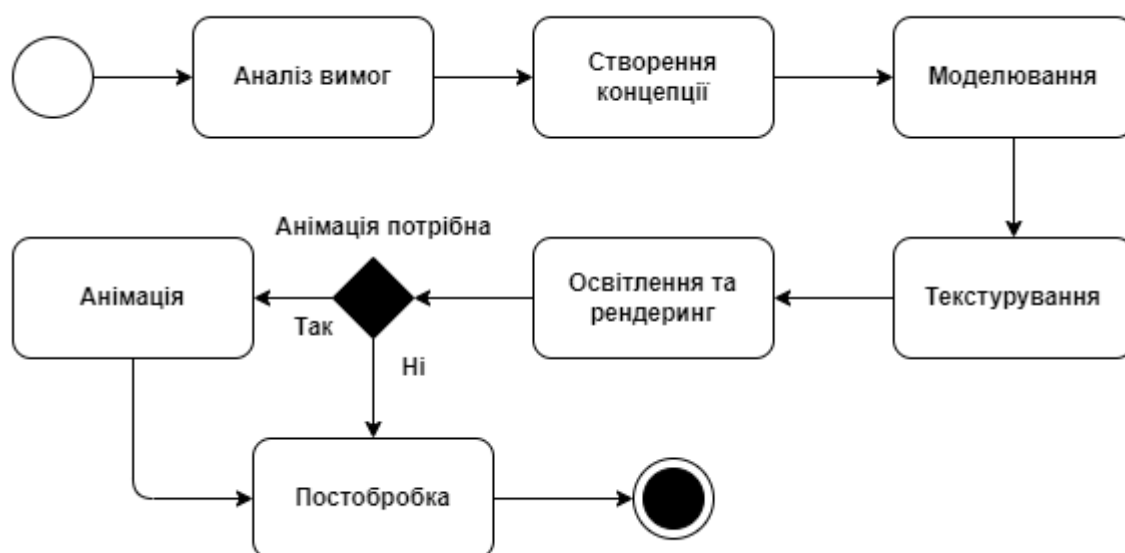


Рисунок 2.1 – Діаграма станів етапів розробки 3D-моделі

Процес створення 3D-моделі у Blender розпочинається з запуску програми. Blender - це відкрите програмне забезпечення для 3D-моделювання, анімації, рендерингу та композингу. Перш ніж розпочати моделювання, важливо чітко уявити концепцію 3D-моделі. Це може включати створення ескізів або концептуальних зображень (див. рис. 2.1).

Після визначення концепції, переходимо до моделювання основної форми. У Blender можна створити модель з нуля або скористатися готовими примітивами, такими як кулі або куби, і використовувати інструменти моделювання для їх зміни та редагування.

Після моделювання важливо перевірити коректність 3D-моделі та виправити будь-які помилки. Далі можна переходити до створення деталізованих елементів за допомогою інструментів скульптування або додавати додаткові деталі, такі як волосся або одяг.

Діаграму активності, що описує процес створення комп'ютерної 3D-моделі та текстур можна переглянути на (рис. 2.2), в ній коротко описанні кроки створення комп'ютерної графіки.

Після завершення моделювання наступним процесом є текстурювання, це потрібно щоб зробити модель більш реалістичною. Це включає створення текстур або використання готових текстур з онлайн - бібліотек.

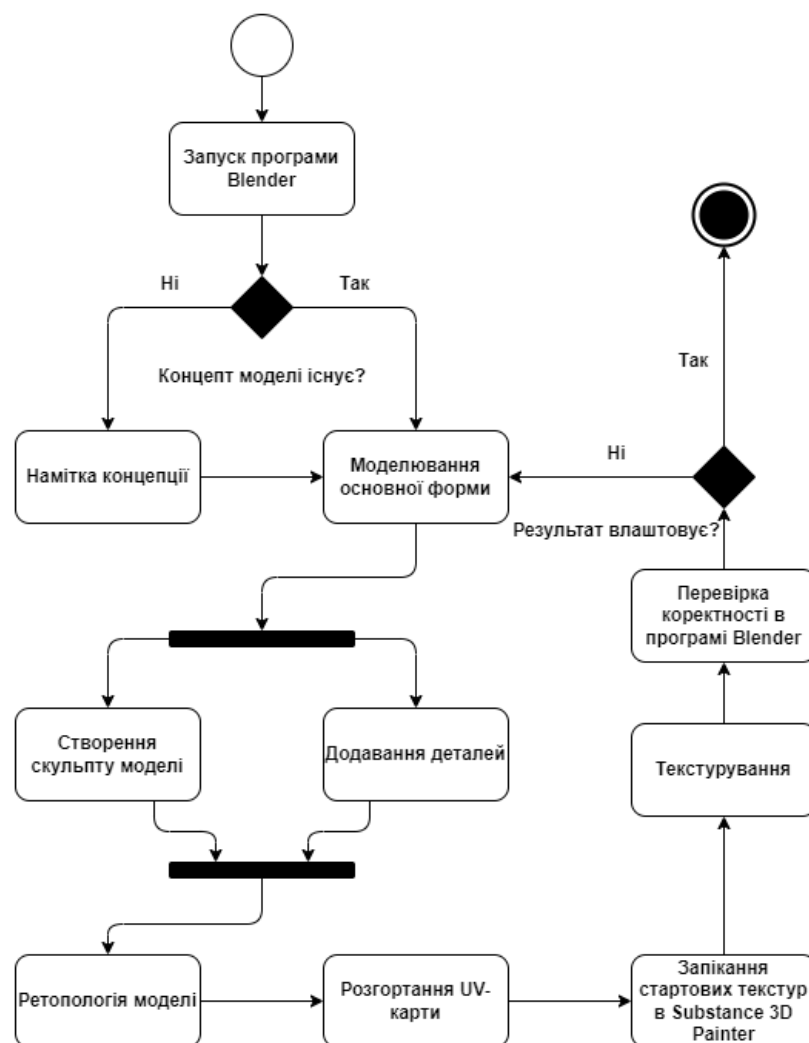


Рисунок 2.2 – Діаграма активності, що описує процес створення комп'ютерної 3D-моделі та текстур

Наступний процес це запікання текстур, що допоможе покращити візуальну якість 3D-моделі. Після цього потрібно виконати ретопологію 3D-моделі та розгорнути UV-карти для підготовки до рендерингу.

Діаграми послідовностей описує поведінку ігрового додатку, представляючи взаємодію між об'єктами в часі. Діаграма послідовності (рис. 2.3) описує послідовний процес розробки готової 3D-моделі та текстур. Насправді процес може бути більш складним і потребувати додаткових кроків, залежно від типу ігрового додатку та використовуваних технологій.

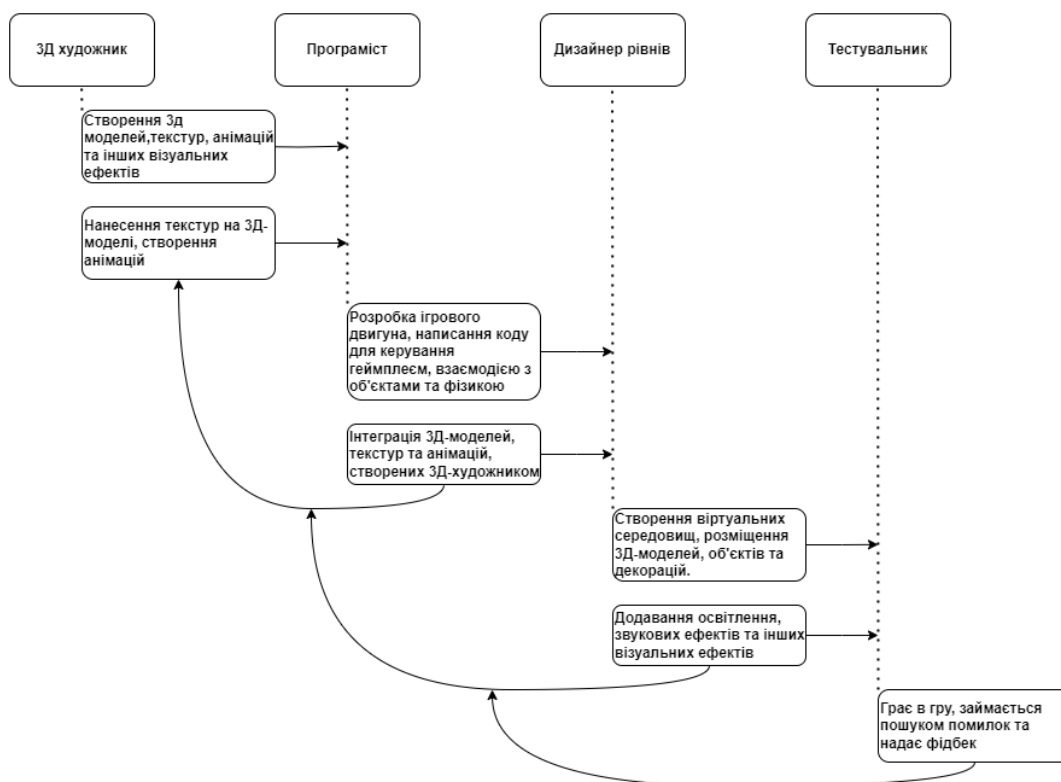


Рисунок 2.3 – Діаграма послідовності процесу розробки готової 3D-моделі та текстур

Процес розробки комп'ютерної графіки є ітеративним. 3D-художник, програміст, дизайнер рівнів та тестувальник постійно працюють разом, щоб покращити візуальну якість комп'ютерної гри та ігровий процес.

Діаграми варіантів використання описують функціональні вимоги до програмної системи, представляючи акторів, варіанти використання та відносини між ними. В даному випадку діаграма варіантів використання описує процес отримання та виконання роботи по створенню готового 3D-асету.

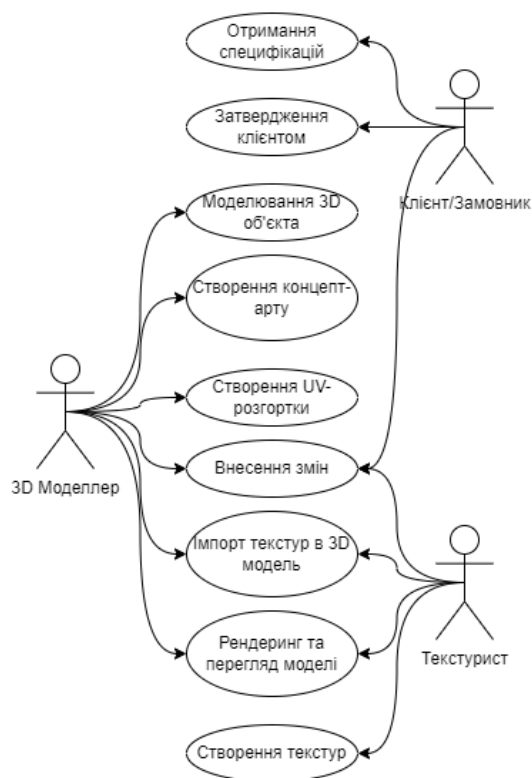


Рисунок 2.4 – Діаграма варіантів використання процесу отримання та виконання роботи по створенню готового 3D-аскету

2.2 Розробка 3D-моделі для комп'ютерної гри

Починаючи процедуру розробки, важливо чітко уявити концепцію комп'ютерної графіки для комп'ютерної гри, її функції та основні характеристики. Людина, яка створює графіку, повинна мати уявлення про те, як ця графіка буде виглядати в грі і які компоненти для неї потрібні.

Щоб створити комп'ютерну графіку, можна використовувати спеціалізоване програмне забезпечення для 3D-моделювання, наприклад Blender, яке передбачає розміщення примітивів у 3D-просторі. Типові примітиви можуть включати сфери, конуси або циліндри, де вносяться коригування для отримання певних форм — це допомагає визначити різні частини об'єкта легше, ніж за допомогою лише одного примітиву форми, розміщеного на порожній сцені.

Крім того, важливо врахувати технічні обмеження комп'ютерної гри, такі як розширення текстур, кількість полігонів, освітлення тощо. Наприклад, для

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		28

комп'ютерної гри, яка має працювати на мобільних пристроях, можуть бути встановлені обмеження щодо роздільної здатності текстур або кількості одночасно відображуваних об'єктів.

У процесі розробки графіки також важливо забезпечити зручний робочий процес для художника. Це може включати оптимізацію інтерфейсу програмного забезпечення, використання шаблонів або скриптів для автоматизації рутинних завдань, а також забезпечення можливості візуального попереднього перегляду графіки в контексті комп'ютерної гри

Нарешті, важливо пам'ятати, що розробка комп'ютерної графіки - це ітеративний процес. Під час розробки можуть з'явитися нові ідеї або вимоги, які потребуватимуть змін у вже створеній графіці. Тому важливо мати гнучкість і готовність до змін під час всього процесу розробки.

Для початку потрібно створити інструмент з допомогою якого буде зручно формувати форму дерева [18].

В програмі Blender потрібно затиснути кнопки Shift + A та з вападаючого списку в категорії Mesh вибрати пункт Single Vert та обрати одиничну вершину (Add Single Vert) (рис. 2.5).

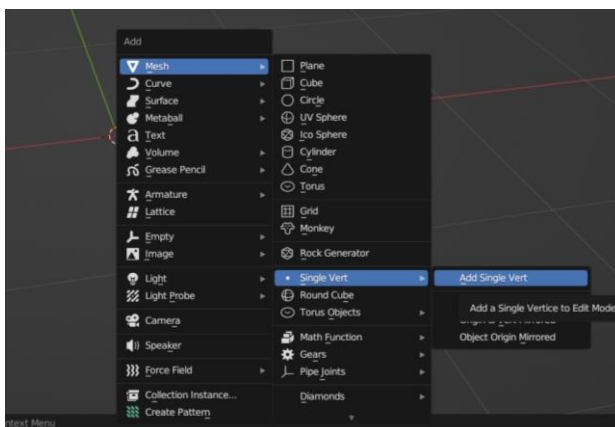


Рисунок 2.5 – Створення одиничної вершини (Add Single Vert)

Після створення одиничної вершини потрібно накласти на неї два модифікатори, Skin та Subdivision Surface.

Модифікатор Skin у програмі Blender використовується для автоматичного створення м'яких і кривих форм, таких як тіло людини або тварини. Цей

модифікатор дозволяє швидко створювати базові форми та регулювати їх, не вдаючись до складних і ручних налаштувань.

Алгоритми інтерполяції які використовує модифікатор для обчислення геометрії нових точок, використовуються для створення оболонки навколо контрольних точок кривої. Існує багато різних алгоритмів інтерполяції, кожен з яких має свої переваги та недоліки. Вибір алгоритму інтерполяції залежить від конкретної задачі та доступних даних. Найпоширеніші алгоритми інтерполяції які можна використати в Blender – це лінійна, константна, Безьє, В-сплайн, Рогова, Root.

Один з найпростіших і найпоширеніших алгоритмів - це лінійна інтерполяція, яка використовується для приблизного визначення значення функції між двома відомими точками.

$$f(x) = f(x_0) + (x - x_0) * (f(x_1) - f(x_0)) / (x_1 - x_0)$$

де $f(x)$ - шукане значення функції в точці x ; $f(x_0)$ - значення функції в точці x_0 ; $f(x_1)$ - значення функції в точці x_1 ; x_0 - відома точка з координатою x ; x_1 - відома точка з координатою x .

Рівняння прямої має вигляд:

$$y - f(x_0) / f(x_1) - f(x_0) = x - x_0 / x_1 - x_0$$

Звідси для $x \in [x_0, x_1]$ маємо:

$$f(x) \approx y = P_1(x) = f(x_0) + f(x_1) - f(x_0) / x_1 - x_0 * (x - x_0)$$

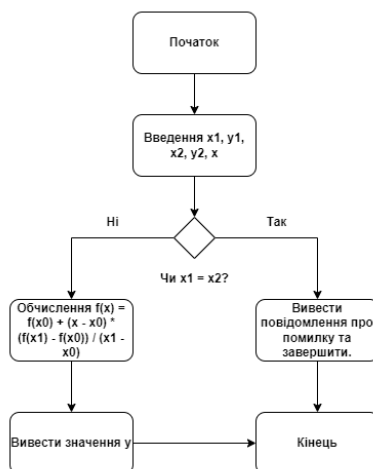


Рисунок 2.6 – Алгоритми лінійної інтерполяції для обчислення геометрії нових точок

Параметри, які можна налаштувати для модифікатора Skin, такі як радіус та кількість сегментів, також відображають математичні властивості. Зміна цих параметрів впливає на спосіб, якими обчислюються та розташовуються нові точки.

Щоб застосувати будь-який модифікатор на геометрію потрібно перейти в меню Modifiers яке знаходиться в правій частині програми. Кнопка цього меню має піктограму синього ключа (рис. 2.7).

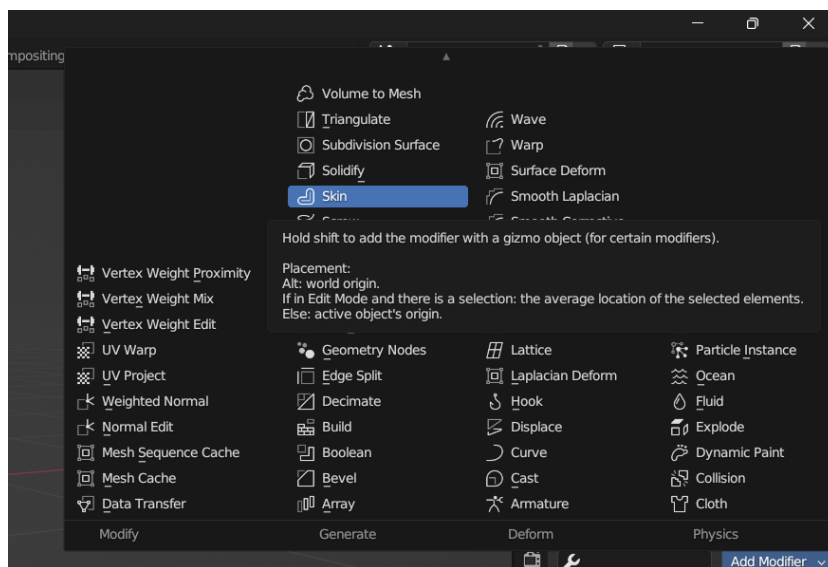


Рисунок 2.7 – Накладання модифікатору Skin на одиничну вершину

Модифікатор Subdivision Surface у програмі Blender використовується для згладжування поверхонь об'єктів. Він дозволяє створювати більш реалістичні та плавні форми, збільшуючи кількість полігонів (трикутників або чотирикутників) на поверхні 3D-моделі. Це особливо корисно при моделюванні органічних об'єктів, таких як тварини, люди або рослини, але використовується також для створення багатьох інших типів об'єктів.

Модифікатор використовує математичні алгоритми для згладжування геометрії об'єкта та додавання додаткових точок для підвищення роздільної здатності [9]. Ці алгоритми базуються на методах, таких як Catmull-Clark, які використовуються для обчислення нової геометрії з використанням існуючих вершин.

Під час застосування модифікатора Subdivision Surface, математичні алгоритми розгладжують кривизну об'єкта, збільшуючи кількість вершин та полігонів, що дозволяє створювати більш гладку та реалістичну поверхню.

Формула Catmull-Clark [10] – це метод, який перетворює грубу полігональну сітку в гладку з більш високою деталізацією. Процес виконується наступним чином:

Алгоритм діє таким чином:

1. Для кожної грані:
 - Обчислюється центр маси грані.
 - Обчислюються середні точки ребер грані.
 - Обчислюється центр ваги всіх точок грані, при цьому кожна точка враховується відповідно до її ступеня.
2. Для кожної вершини:
 - Обчислюється нове положення вершини як середнє з наступних точок:
 - $1/n$ старої вершини, де n - це ступінь вершини.
 - $(n - 3) / n$ центрів маси граней, до яких належить вершина.
 - $1/n$ середніх точок ребер, що сходяться у вершині.
3. Створюються нові грані:
 - Кожна грань старої сітки отримує нову грань, що складається з нових позицій вершин старої грані.
 - Кожне ребро старої сітки отримує нову грань, що складається з нового центру ваги ребра, нових середніх точок на ребрі та нових позицій вершин, які утворюють ребро.

Catmull-Clark та Loop subdivision - це два різні алгоритми субдивізії (subdivision), які використовуються для згладжування поверхонь у комп'ютерній графіці та моделюванні. Обидва алгоритми використовуються в різних програмах для створення більш гладких та деталізованих поверхонь з використанням меншої кількості контрольних точок чи вершин (рис. 2.8).

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		32

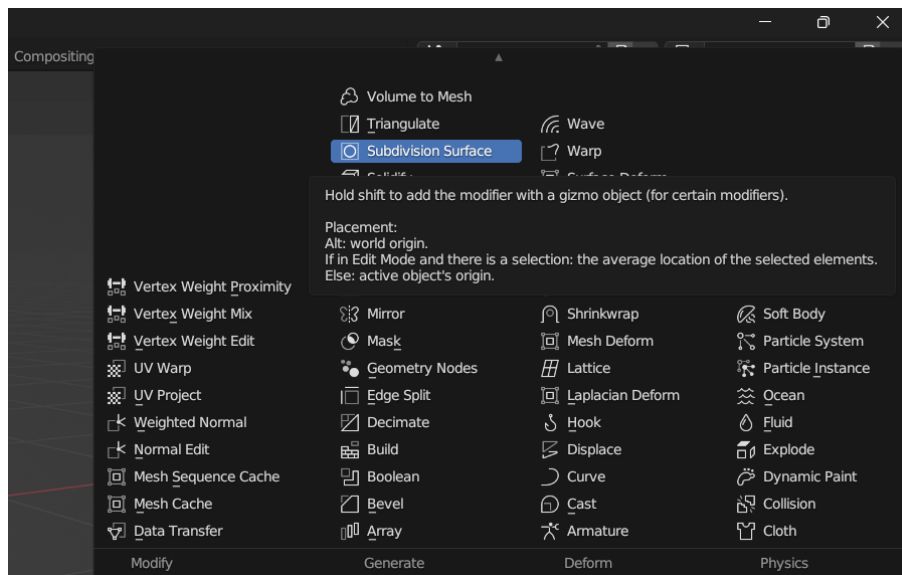


Рисунок 2.8 – Накладання модифікатору Subdivision Surface для згладжування геометрії 3D-моделі

Застосувавши всі необхідні модифікатори, потрібно натиснути на клавіатурі кнопку Tab яка відповідає за перехід до режиму редагування геометрії 3D-моделі, або можна на панелі в лівому верхньому куті вибрати з вападаючого меню замість режиму Object Mode перейти в режим Edit Mode (рис. 2.9).

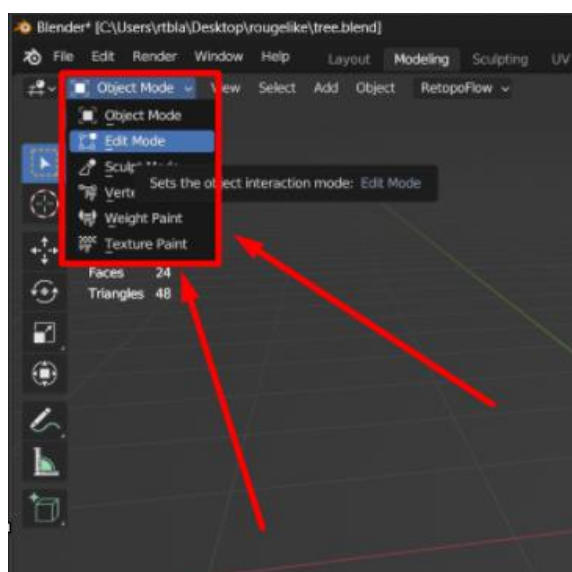


Рисунок 2.9 – Перехід в режим редагування геометрії

Після переходу в режим Edit Mode ми повинні натиснути на клавіатурі кнопку 1 яка переключає редагування геометрії в режим Vertex та маніпулювати вершинами в просторі. Щоб почати формувати довільну форму потрібно вибрати в сцені вершину та натиснути кнопку E яка відповідає за режим Extrude (рис. 2.10).

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		33

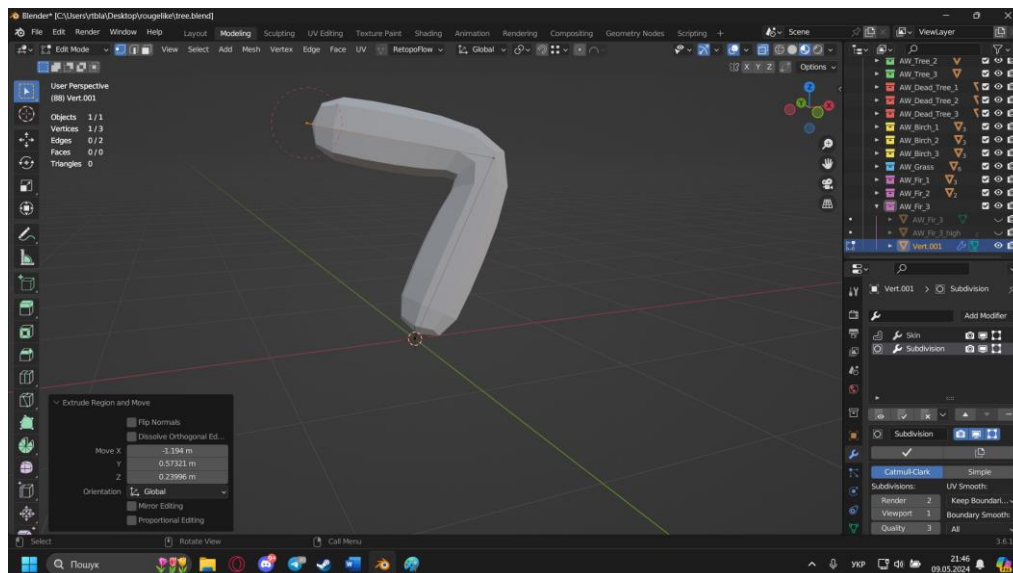


Рисунок 2.10 – Натискання кнопки Е та застосування функції Extrude та витягування довільної форми

Функція Extrude у програмі Blender використовується для створення нових геометричних форм шляхом "витягування" обраних вершин, ребер або обличчя об'єкта. Ця функція дозволяє розширювати об'єкти у тривимірному просторі, що дозволяє створювати складніші форми та додавати об'єм до 3D-моделі.

Додатково потрібно поміняти режим обробки світла в програмі. Функція Shade Auto Smooth у програмі Blender використовується для згладжування вигляду поверхні об'єкта, надаючи йому більш природний, м'який вигляд.

Ця функція особливо корисна для обробки органічних об'єктів, таких як персонажі, тварини або рослини, де різні частини повинні мати гладку поверхню. Вона також може використовуватися для створення різних способів освітлення та відображення на об'єктах, щоб забезпечити більш реалістичний вигляд. Щоб застосувати режим Shade Auto Smooth потрібно знаходитися в режимі Object Mode, вибрати модель в сцені та натиснути праву кнопку миші та вибрати відповідний режим (рис. 2.11). Для зручності також потрібно встановити градус при якому буде заломлюватися світло, оптимальним рішенням буде 46 градусів, аналогічно потрібно в модифікаторі Skin натиснути кнопку Smooth Shading.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		34

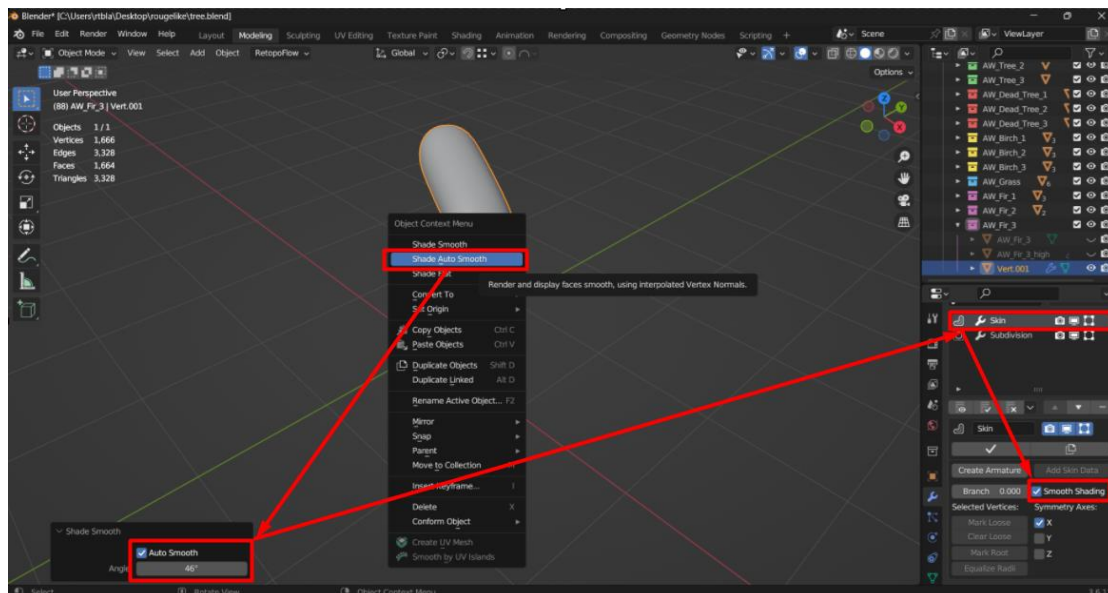


Рисунок 2.11 – Обрання режиму Shade Auto Smooth та натискання кнопки Smooth Shading в модифікаторі Skin

Після налаштування всіх необхідних параметрів можна створити довільну форму дерева. Щоб зменшити або збільшити товщину гілки або стовбура потрібно виділити вершину та затиснути одночасно дві кнопки Ctrl + A (рис. 2.12).

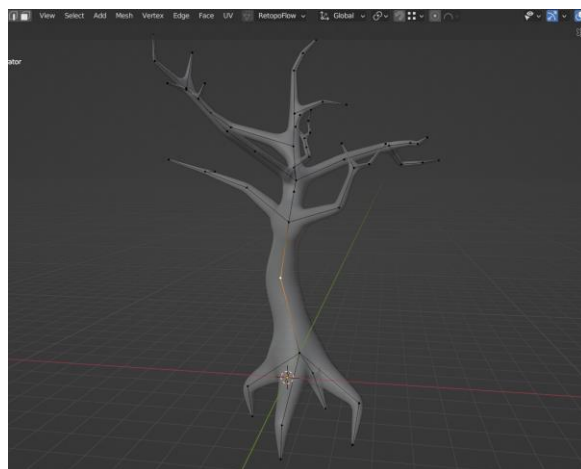


Рисунок 2.12 – Зміна товщини стовбура дерева

Після створення базової форми потрібно зберегти копію налаштованого дерева. Щоб здублювати модель потрібно виділити її та затиснути клавіші Shift + D та в списку моделей з правого боку перести її в колекцію для зберігання якщо потрібно буде створити модель іншої форми. Далі виділяємо здубльовану модель та в меню модифікаторів параметр Viewport модифікатора Subdivision Surface збільшуємо щоб на 3D-моделі було не менше ніж 800000 тисяч полігонів, це

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				35
Змн.	Арк.	№ докум.	Підпис	Дата		

потрібно для зручності в режимі Sculpting. Застосовуємо всі наявні на 3D-моделі модифікатори (рис. 2.13) та переходимо в режим Sculpting де потрібно додати деталей на модель такі як: форму кори, тріщини, відколоті частини та форму мертвих гілок (рис. 2.14).

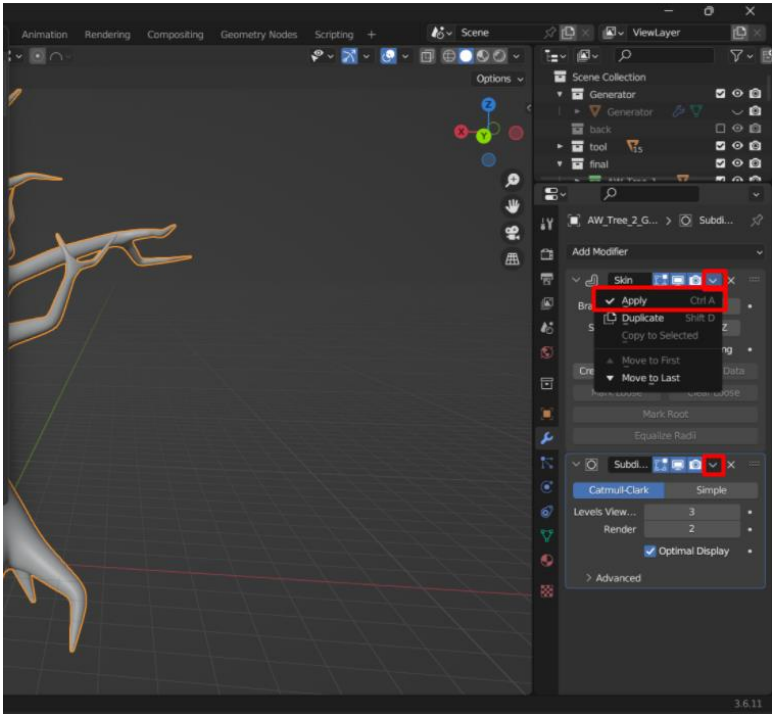


Рисунок 2.13 – Застосовування всіх накладених на модель модифікаторів

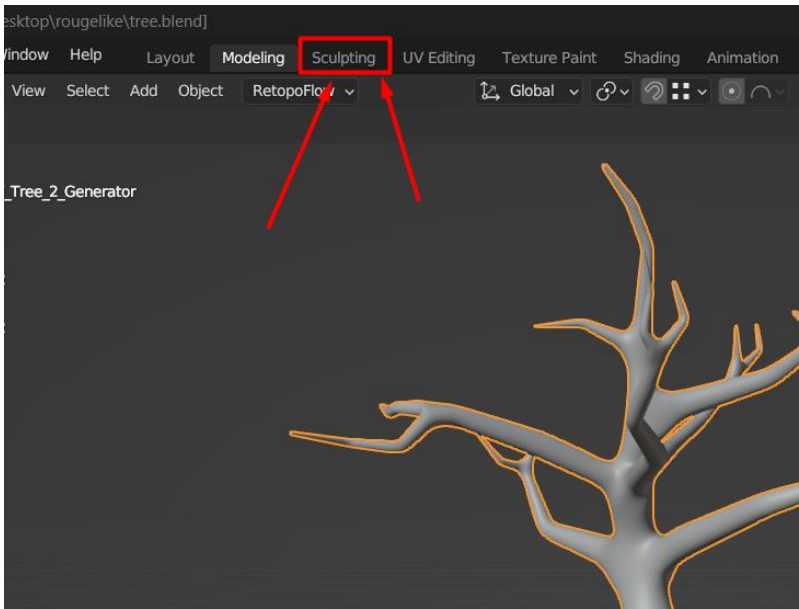


Рисунок 2.14 – Перехід в режим Sculpting для нанесення деталей

Перш ніж переключити режим з Modeling на режим Sculpting потрібно на модель дерева застосувати новий модифікатор під назвою Multiresolution, модифікатор (багаторівнева роздільна здатність) у програмі Blender

використовується для створення деталізованих моделей з використанням ієрархії роздільності (рівнів деталізації) (рис. 2.15).

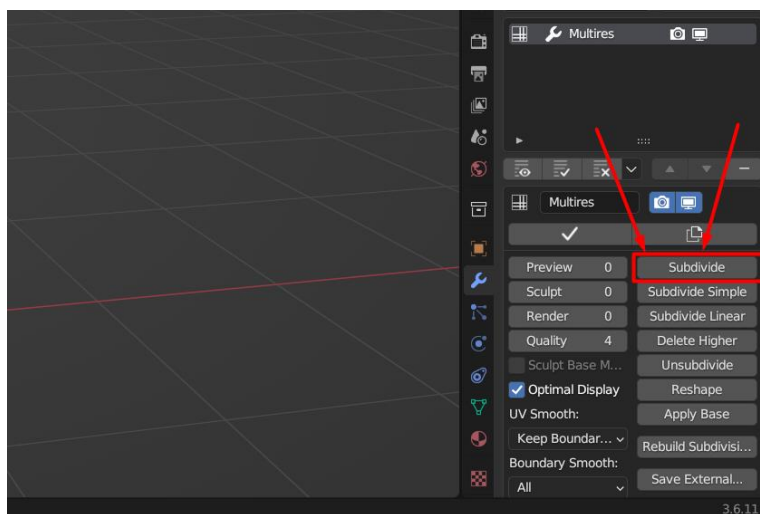


Рисунок 2.15 – Накладаємо модифікатор Multiresolution та додаємо декілька рівнів згладжування за допомогою кнопки Subdivide

Для нанесення необхідних деталей на високополігональну модель було використано пензлі такі як: Draw, Crease, Smooth, Scrape. Силу пензлів можна налаштувати затиснувши кнопки Ctrl + F та сам розмір затиснувши кнопку F.

Пензлем Draw потрібно створити рельєф стовбура по спіралі від гілок до коріння, гаряча клавіша пензлю – X. В місцях де це потрібно за допомогою затиснутої кнопки Ctrl ми можемо вдавину частину полігонів в середину щоб зобразити ями в стовбурі (рис. 2.16).

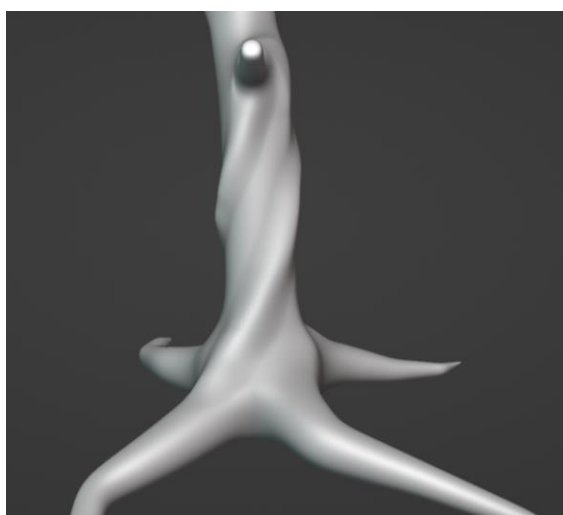


Рисунок 2.16 – Накладання спірального рельєфу

Після створення спірального рельєфу, потрібно кут між нарощеним слоєм та основним стовбуром зробити гострим, від чого буде більш приємніша тінь яку

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		37

забезпечить текстурна карта Ambient Occlusion, потрібно використати пензл Crease, гаряча клавіша Ctrl + C (рис. 2.17).



Рисунок 2.17 – Накладання спірального рельєфу

Далі потрібно використати пензл Scrape щоб створити прямі кути які зображають рельєф кори стовбура дерева (рис. 2.18). Де потрібно використовуємо пензель Smooth для згладжування затемнених ділянок та нерівностей геометрії, гаряча клавіша пензлю Shift + S (рис. 2.19).



Рисунок 2.18 – Створення рельєфу кори стовбура



Рисунок 2.19 – Згладження ділянка за допомогою пензлю Smooth

Після режиму Sculpting потрібно оптимізувати модель, це називається процесом Retopology.

Процес ретопології 3D-моделі є важливою частиною процесу моделювання у 3D-графіці, і він виконується з різних причин таких як:

1. Покращення топології: Ретопологія дозволяє переосмислити топологію 3D-моделі з метою зробити її більш оптимальною та ефективною. Це може включати створення чистої, легко читабельної сітки з оптимальним розміщенням вершин та ребер.

2. Оптимізація для анімації та рендерингу: Ретопологія дозволяє оптимізувати модель для подальшого використання у анімації та рендерингу. Чиста та оптимізована меша може покращити продуктивність при обробці та рендерингу, а також забезпечити кращі результати під час анімації.

3. Підготовка для деталізації: Ретопологія дозволяє підготувати модель для додавання деталей, таких як текстури, нормальні карти, або деталізовані модифікації. Це може бути корисно при створенні більш складних та реалістичних зображень.

4. Покращення управління полігонами: Ретопологія дозволяє керувати кількістю полігонів у 3D-моделі. Це важливо для забезпечення оптимальної роздільної здатності та продуктивності при роботі з моделлю.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		39

5. Покращення деталізації та форми: Ретопологія може допомогти покращити деталізацію та форму 3D-моделі, забезпечуючи більш гладкі переходи та кращу підтримку для деталей.

Для процесу ретопології в програмі Blender був використаний аддон RetopoFlow [4], який дозволяє швидко та якісно виконати процес ретопології за допомогою зручних інструментів та зрозумілого інтерфейсу.

Перед тим як почати працювати з аддоном його потрібно завантажити з сайту[4], щоб встановити потрібно в лівому верхньому куту програми Blender навести на кнопку Edit та з випадаючого списку вибрати пункт Preferences (рис. 2.20).

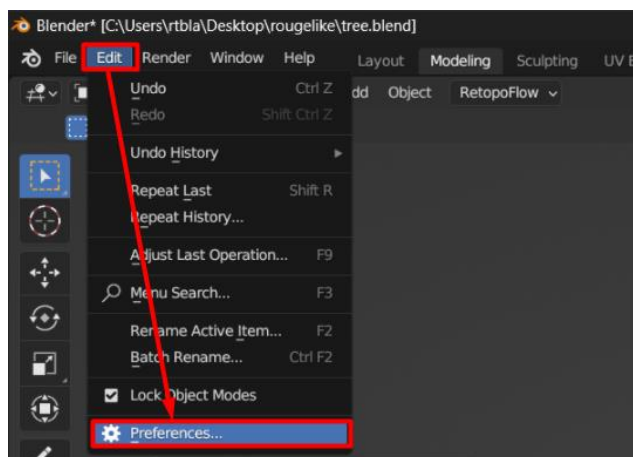


Рисунок 2.20 – Перехід до налаштувань програми для встановлення користувацького аддону

Після відкриття налаштувань програми потрібно перейти до пункту Addons та натиснути кнопку Install (рис. 2.21).

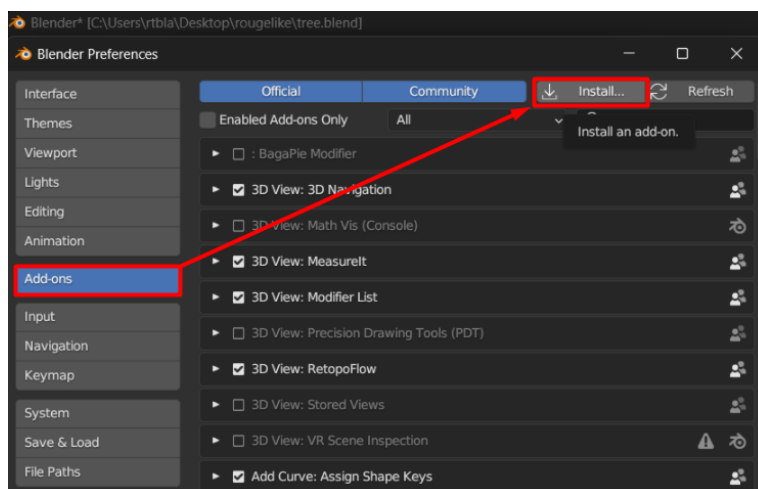


Рисунок 2.21 – Перехід до пункту Addons та натискання кнопки Install

Після натискання кнопки Install потрібно вибрати заархівований файл аддону, він обов'язково повинен мати розширення zip. Після того як ми встановили аддон, потрібно в Blender поставити галочку щоб він запрацював (рис. 2.22).

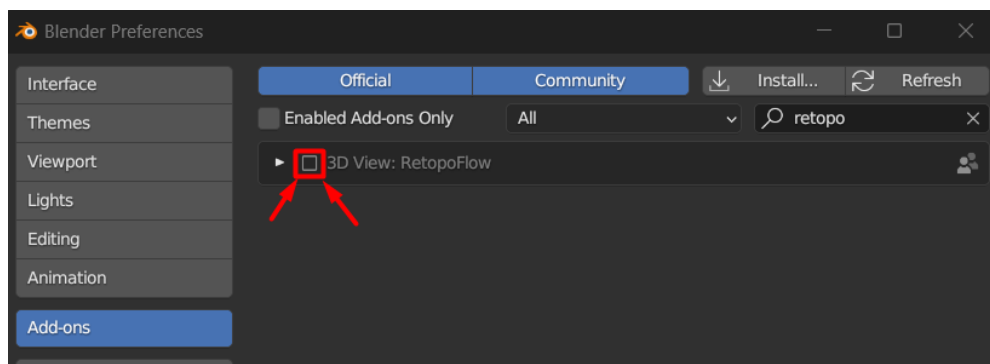


Рисунок 2.22 – Ввімкнення імпортованого аддону

Після ввімкнення аддону в налаштуваннях програми, потрібно вибрати модель на якій потрібно провести процес ретопології та тиснути кнопку з назвою RetopoFlow яка з'явилася біля меню Object. Після чого натискаємо кнопку + at active. Після її натискання вікно програми переключиться на вікно з інструментами аддону (рис. 2.23).

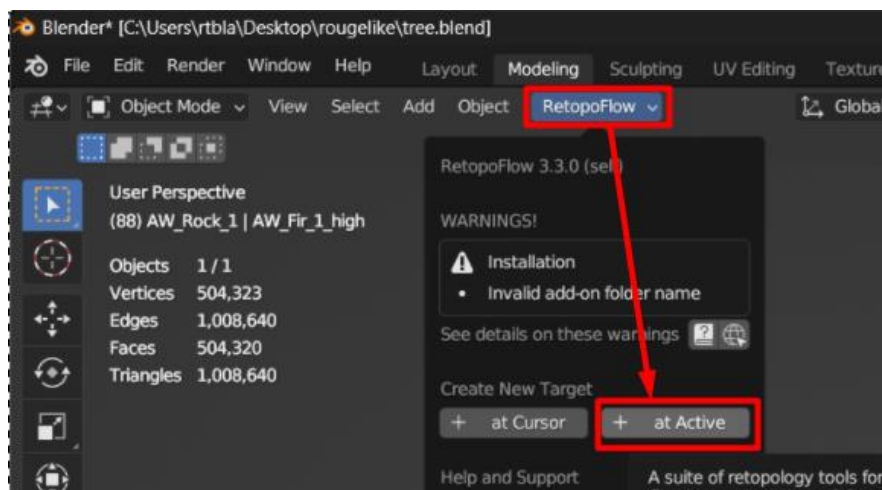


Рисунок 2.23 – Перехід в режим ретопології

Для виконання процесу ретопології було використано такі інструменти аддону як: Countours, PolyStrips, Patches та PolyPen.

1. За допомогою Contours можна швидко розміщувати та регулювати петлі навколо будь-якої циліндричної форми.

2. За допомогою PolyStrips можна швидко створювати ескізи. Потрібно утримувати Ctrl і затискати ліву кнопку миші, щоб намалювати смужку. Це може бути розширення або з'єднання між існуючою топологією або абсолютно новою смугою залежно від того, де можна почати та зупинити штрих.

3. Великі області можна заповнити одночасно за допомогою Patches. Потрібно вибрати граничний край і натиснути F, щоб заповнити. Можна перемикаати вибір, які вершини слід вважати кутами, тому він є достатньо гнучким, щоб працювати з набагато більше, ніж звичайними прямокутними формами.

4. PolyPen створено, щоб надати максимальний контроль над дрібними деталями. Виділіть край, утримуйте Ctrl і кладніть, щоб розширити виділення трикутниками, чотирикутниками або обома способами. PolyPen автоматично прив'язується до інших вершин і країв і навіть може використовуватися як ніж для виконання базових надрізів.

Після проведеного процесу ретопології повинна вийти добре оптимізована модель з лівого боку з сумарним полігонажем 965 полігонів або 1911 трикутників, а модель з якої робили процес ретопології мала в суммі 543 744 полігонна або 1 087 488 трикутників (рис. 2.24).

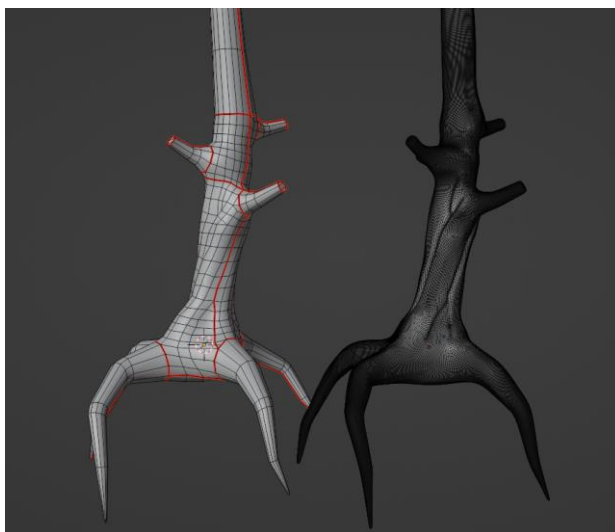


Рисунок 2.24 – Приклад оптимізованої сітки 3D-моделі після процесу ретопології

Наступним етапом буде створення оптимізованої UV-карти 3D-моделі. UV-карта 3D-моделі використовується для того, щоб призначити текстурі

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		42

частину поверхні 3D-об'єкта. Це дозволяє створювати детальні та складні текстури, які наносяться на модель для створення більш реалістичного вигляду.

UV-карта використовується для відображення та призначення текстур на 3D-модель. Кожній точці на UV-карті відповідає певна частина поверхні 3D-моделі, на яку наноситься текстура.

Завдяки UV-карті можна легко керувати розташуванням текстурних даних на 3D-моделі. Це дозволяє використовувати одну текстуру на кількох об'єктах чи створювати складні композиції текстур для покращення вигляду сцени.

Для 3D-моделі було створено оптимізовану UV-карту адже це є важливим кроком для забезпечення ефективного використання текстурних ресурсів [17], а також для підвищення якості та продуктивності роботи з 3D-моделями. Для її створення було використано декілька професійних аддонів таких як: TexTools [5], UV Toolkit [6] та UVPackmaster3 [7] (рис. 2.25).

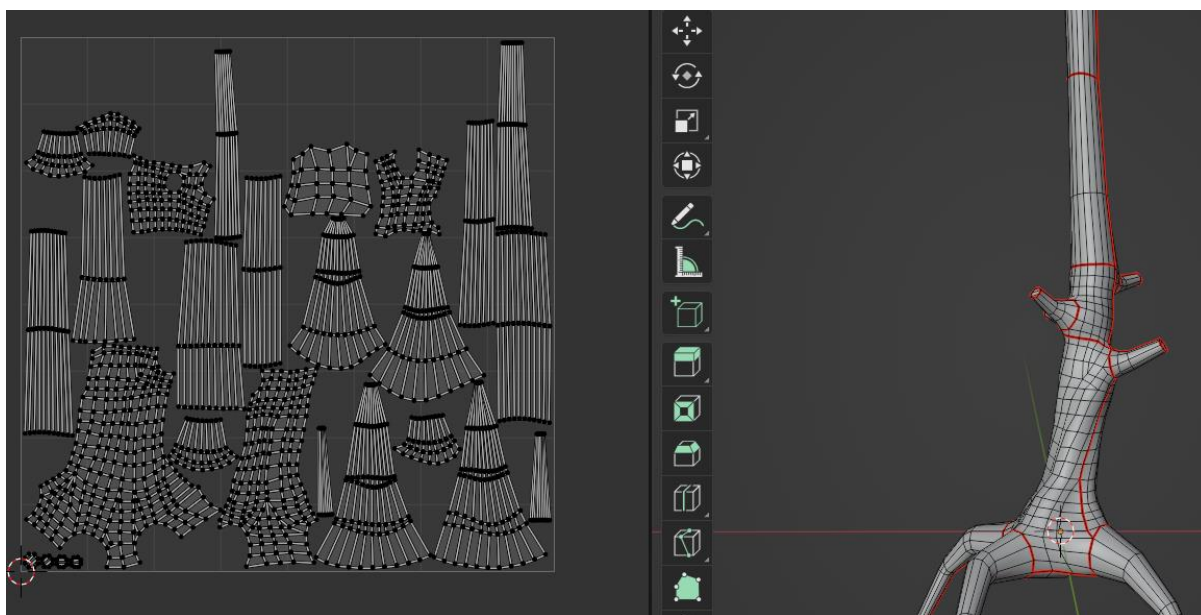


Рисунок 2.25 – Оптимізована UV-карта для запікання текстур та основного процесу текстуровання

2.3 Створення графічного матеріалу для комп'ютерної гри

Створення текстур в програмі Substance 3D Painter починається з підготовки готових моделей. Потрібно експортувати 3D-моделі з програми

Blender з спеціальними налаштуваннями які спростять налаштування в програмі для створення текстур та налаштуванні в ігровому рушії [11].

Потрібно виділити модель та дописати в кінець назви 3D-моделі припис _low, це потрібно для правильного налаштування в процесі запікання базових текстур (рис. 2.26).

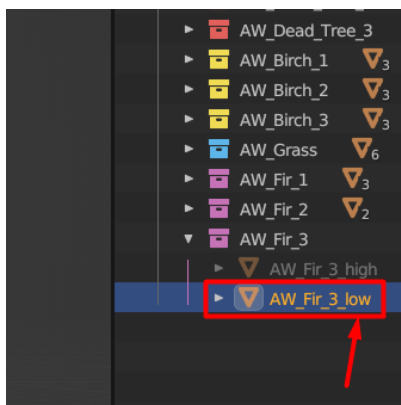


Рисунок 2.26 – Дописування в кінець назви 3D-моделі приписки _low

Аналогічним чином потрібно на раніше заскульпчену 3D-модель також дописати приписку, тільки вже потрібно приписати _high (рис. 2.27).

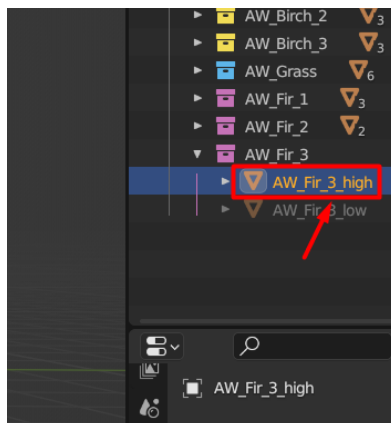


Рисунок 2.27 – Дописування в кінець назви 3D-моделі приписки _high

Запікання текстур - це процес перетворення різних типів інформації, таких як освітлення, тіні, нормалі, глобальне затемнення тощо, у текстурні карти. Ці текстурні карти потім можуть бути використані для покращення візуальної якості 3D-моделі під час рендерингу.

Щоб розпочати процес текстурювання потрібно відкрити програму Substance 3D Painter та в лівому верхньому куті навести курсор миші на кнопку File та вибрати пункт New, або скористатися комбінацією клавіш Ctrl + N (рис. 2.28).

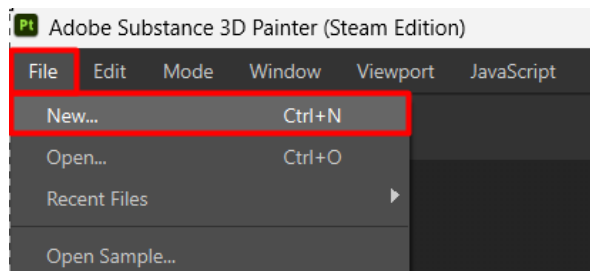


Рисунок 2.28 – Створення нового файлу для текстурвання

Після натискання кнопки New перед нами відкриється вікно з початковими налаштуваннями для імпорту 3D-моделі в програму. Потрібно натиснути кнопку Select для імпорту 3D-моделі з припискою _low, вибрати роздільну здатність текстур та тип відображення графіки, OpenGL або DirectX (рис. 2.29).

OpenGL та DirectX - це дві різні технології для відображення графіки у програмному забезпеченні, такому як комп'ютерні ігри, відеоредактори, 3D-моделювальні програми та інші. Обидва ці фреймворки дозволяють програмістам створювати складну графіку та використовувати можливості графічного прискорення, щоб підвищити продуктивність і якість візуалізації у програмах та іграх. Вибір між OpenGL та DirectX зазвичай залежить від потреб конкретного проекту, платформи, на якій він запускається, та особистих вподобань розробників.

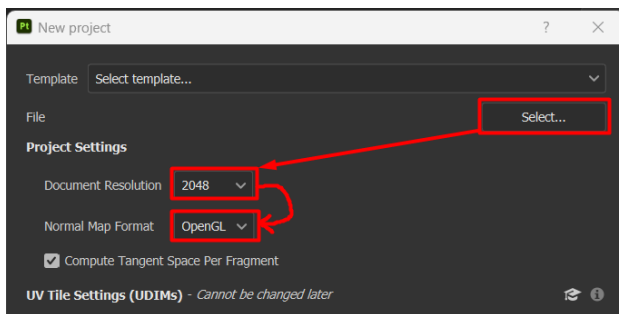


Рисунок 2.29 – Вікно налаштувань імпорту 3D-моделі в програму

Після налаштувань імпорту натискаємо кнопку ОК та перед нами з'являється оптимізована модель з налаштованою UV-картою (рис. 2.30).

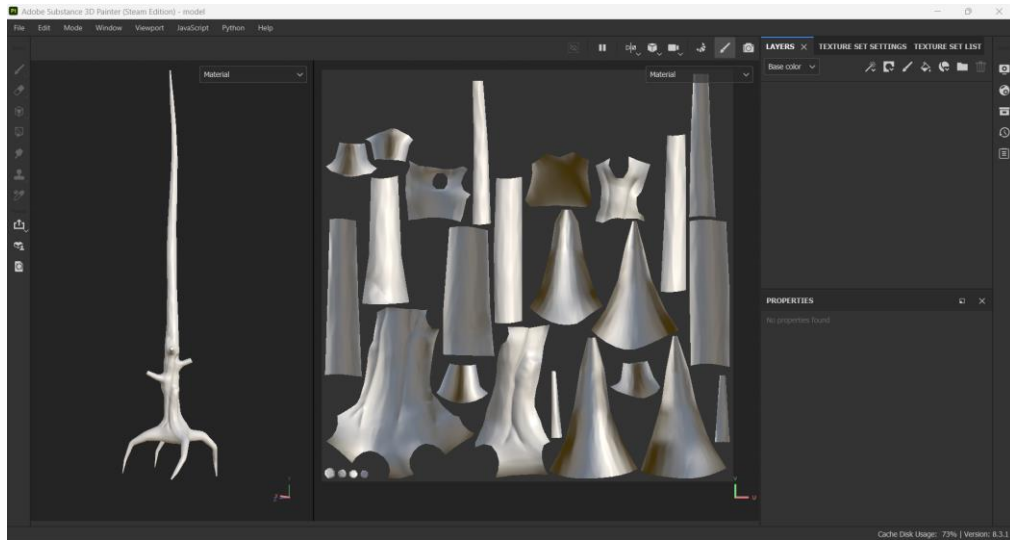


Рисунок 2.30 – Вікно програми з усіма налаштуваннями

Щоб запекти необхідні базові текстури потрібно натиснути кнопку з правого верхнього боку, кнопка яка має піктограму круасану (рис. 2.31).

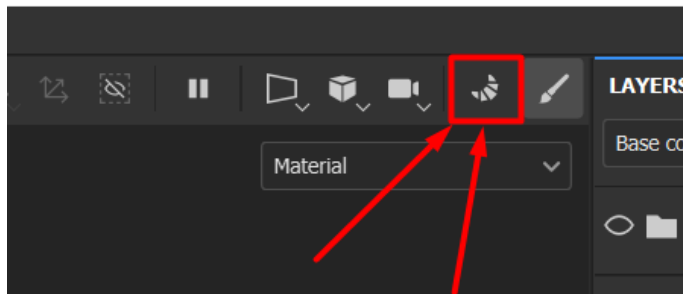


Рисунок 2.31 – Кнопка переходу в режим запікання текстур

Після натискання кнопки, з'являється меню запікання текстур, потрібно вибрати необхідну роздільну здатність текстур, вибір роздільної здатності залежить для текстури об'єкту залежить від технічного завдання. Також потрібно натиснути кнопку з назвою Apply Difusion, це потрібно щоб при запіканні текстур на місцях швів 3D-моделі не з'являлися артефакти запікання, до цієї кнопки також приєднане поле вибору Dilation Width, воно потрібно для ширину захоплення режиму Apply Difusion.

Далі потрібно в полі High Definition Meshes натиснути кнопку з піктограмою файлу та вибрати високо полігональну модель яка має приписку _high. З цієї 3D-моделі при запіканні буде переноситися світло та вся деталізація яку ми наносили в програмі Blender в режимі Sculpting.

Потрібно також виставити параметри Max Frontal Distance. Цей параметр відповідає за область навколо 3D-моделі з якої будуть вилітати промені які

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		46

будуть переносити світло з високо полігональної 3D-моделі на низько полігональну. Також потрібно виставити інший параметр, Max Rear Distance. Він потрібний для глибини запікання 3D-моделі, цей параметр не можна робити більшим по значеннях ніж параметр Max Frontal Distance.

На лівій панелі потрібно вибрати наступні необхідні карти які в подальшому будуть використовуватися в програмі: Normal, World Space Normal, ID, Ambient Occlusion, Curvature, Position, Thickness (рис. 2.32).

Normal Map (Карта нормалей). Використовується для створення ілюзії деталізації та глибини на поверхні 3D-моделі.

World Space Normal Map (Карта нормалей у світовому просторі). Зберігає інформацію про нормалі відносно глобального координатного простору, що полегшує взаємодію з іншими об'єктами у сцені.

ID Map (Карта ідентифікаторів). Використовується для розділення окремих частин 3D-моделі за допомогою унікальних кольорів для кожного елемента.

Ambient Occlusion Map (Карта амбієнтної оклюзії). Моделює темні області на поверхні, де світло має тенденцію не потрапляти, підсилюючи глибину та реалістичність.

Curvature Map (Карта кривизни). Відображає зміни кривизни на поверхні 3D-моделі, корисна для візуалізації деталей та текстур.

Position Map (Карта позиції). Зберігає глобальні координати кожної точки поверхні 3D-моделі, що може бути корисним для різних обчислень у програмі.

Thickness Map (Карта товщини). Визначає товщину об'єктів у сцені, що корисно для симуляцій та обробки ефектів просвічування.

Після виставлення всіх необхідних налаштувань потрібно натиснути в правому нижньому куті синю кнопку Bake selected textures.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		47

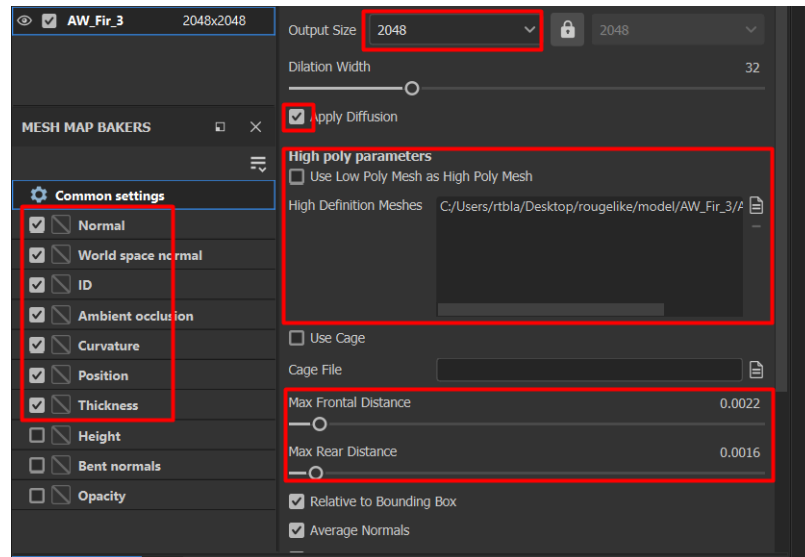


Рисунок 2.32 – Меню з необхідними вибраними параметрами

Після запікання всіх базових текстур потрібно натиснути натиснути кнопку Return to painting mode.

Був створений базовий матеріал з назвою Base, матеріал мав канали Base Color з значеннями # C09167 та Roughness з значенням 0.9458 (рис. 2.33).

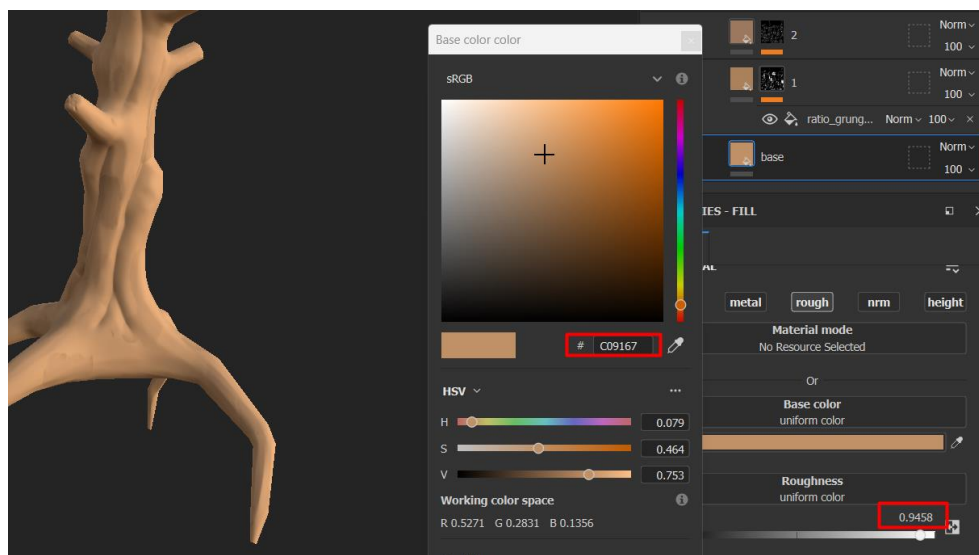


Рисунок 2.33 – Створено базовий матеріал

Наступним шаром потрібно створити плями бруду на дереві, для цього потрібно створити новий шар, також в нього ввімкнено канал Base Color з значеннями #AB825C та канал Roughness з значеннями 0.9583, щоб створити чорну маску на шару потрібно натиснути праву кнопку миші та вибрати пункт Add black mask. Це потрібно для того щоб шар з кольором накладався тільки в ті місця які буде накладати чорно-біла текстура. Щоб накласти її потрібно на чорну

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		48

маску натиснути правою клавішою миші та вибрати пункт Add fill та вибрати карту з назвою Grunge Map 004 (рис. 2.34).

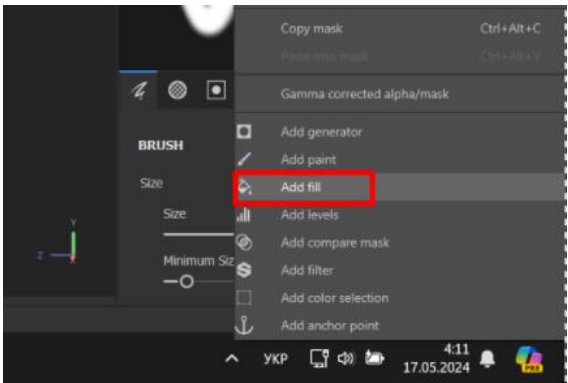


Рисунок 2.34 – Додавання заповнюючого шару

Аналогічним чином було створенно наступний шар з налаштуваннями вже для накладання чорних плям. Для цього шару налаштування Base Color дорівнює #9C785E та значення каналу Roughness дорівнює 0.9583. Також було створенно чорну маску та накладенно текстуру з назвою Grunge Map 004, також змінено канал накладання з UV prokection на Tri-planar projection. Це потрібно для того щоб текстура накладалася без врахунку UV-карти 3D-моделі а накладалося відносно 3D-моделі в просторі (рис. 2.35).

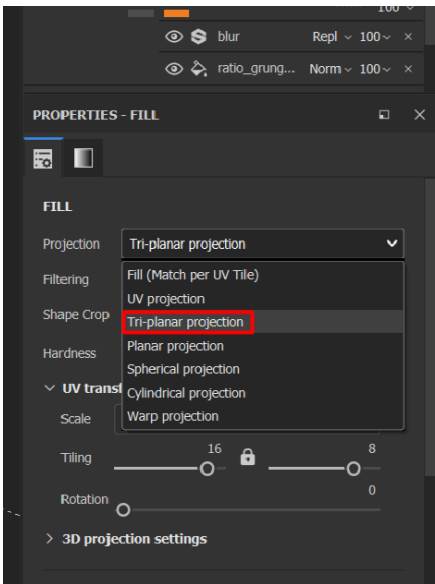


Рисунок 2.35 – Змінення каналу накладання з UV projection на Tri-planar projection

Наступним кроком було накладання фактурності кори дерева. Для цього шару налаштування Base Color дорівнює #7D604B та значення каналу Roughness дорівнює 0.9583. Також було створенно чорну маску та накладенно текстуру з

назвою Grunge Map 005 та поняти режим накладання з Normal на Multiply. Також потрібно створити генератор з назвою Position який накладає шар з ефектом затухання. Цей шар потрібно перенести в створену теку та на теку накласти білу маску, на білу маску накласти генератор UV Border Distance, він потрібний щоб відняти частину текстури від країв 3D-моделі. Це потрібно щоб створити безшовні текстури (рис. 2.36).

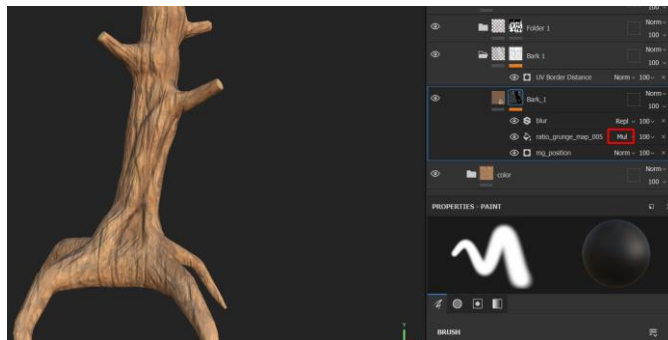


Рисунок 2.36 – Накладання фактурності кори

Фінальний шар відображає текстуру яка заповнює ті частини які генератор UV Border Distance відняв (рис 2.37).

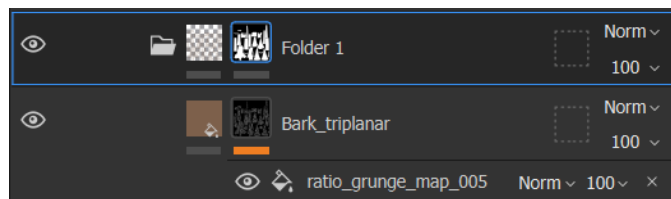


Рисунок 2.38 – Заповнення віднятого генератором простору новою текстурою

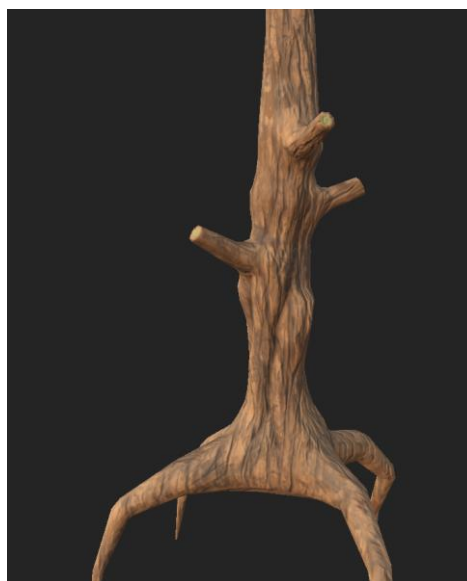


Рисунок 2.39 – Фінальний вигляд текстури в програмі

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				50
Змн.	Арк.	№ докум.	Підпис	Дата		

Щоб експортувати текстури з програми потрібно з лівого верхнього кута натиснути кнопку File та вибрати пункт Export Texture або затиснути комбінацію кнопок Ctrl + Shift + E (рис. 2.40).

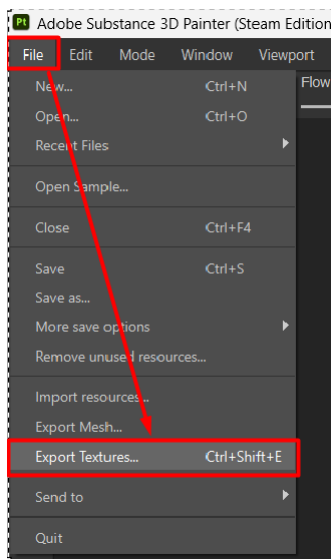


Рисунок 2.40 – Перехід в меню експорту готових текстур

Після переходу в меню експорту потрібно виставити наступні налаштування експорту для ігрового рушія Unity (рис. 2.41).

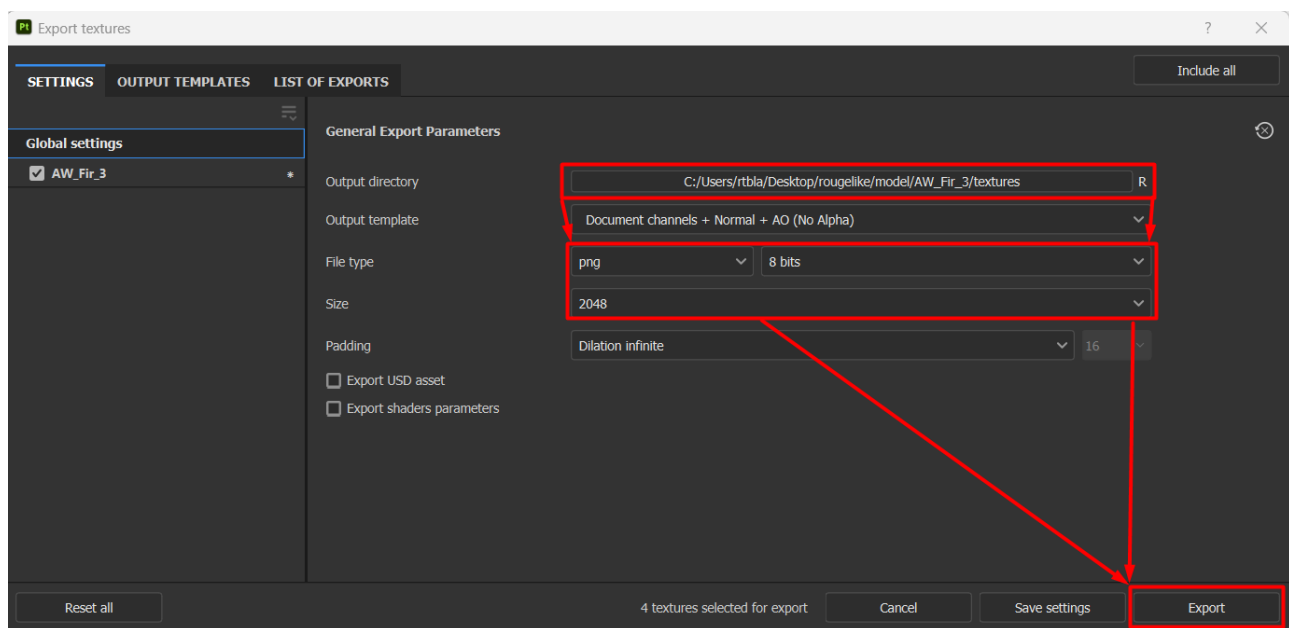


Рисунок 2.41 – Налаштування експорту для ігрового рушія Unity

Після експорту текстур з програми Substance 3D Painter їх потрібно перевірити в програмі Blender на коректність відображення. Для цього потрібно перейти до меню Shading яке відповідає за відображення текстур (рис. 2.42).

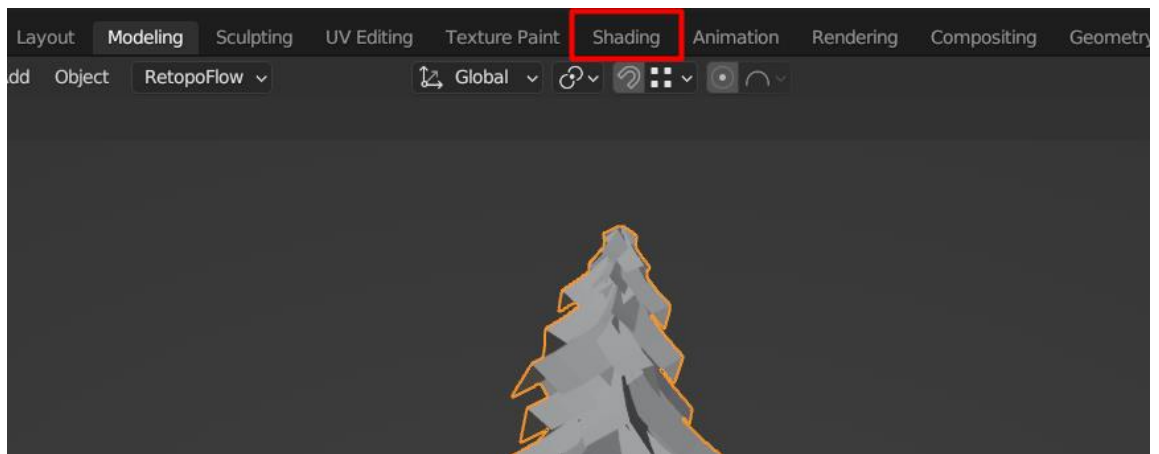


Рисунок 2.42 – Перехід до меню Shading для налаштування текстур

Для швидкої перевірки було використано вбудований в Blender аддон з назвою Node Wrangler [8]. Щоб підключити текстури потрібно затиснути на клавіатурі клавіші Ctrl + Shift + T, відкриється тека в якій потрібно вибрати експортовані текстури. Після підключення побудується сітка нодів з текстурами (рис. 2.43).

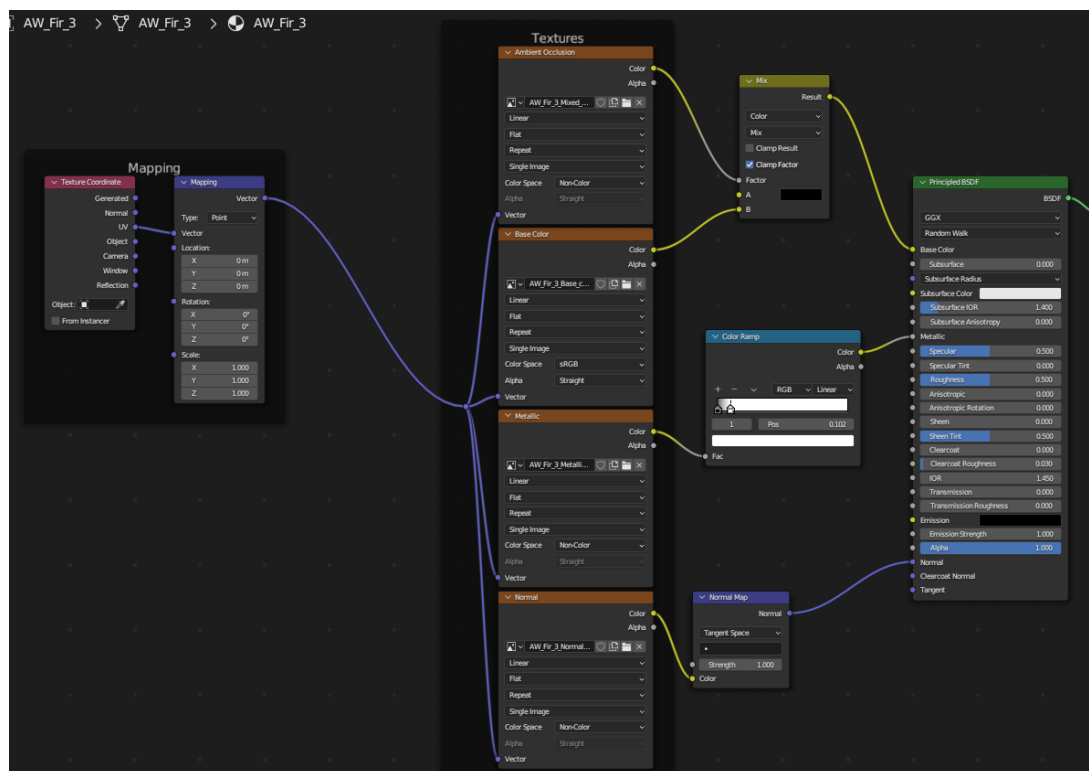


Рисунок 2.43 – Сітка нодів після підключення текстур

Texture Coordinate - генерує різні типи координат для текстурного мапінгу. Використовується для визначення способу розміщення текстури на поверхні об'єкта [12].

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		52

- Generated: Автоматично генеровані координати.
- Normal: Координати нормалі.
- UV: Координати UV-розгортки.
- Object: Координати об'єкта.
- Camera: Координати від камери.
- Window: Екранні координати.
- Reflection: Координати відображення.

Mapping - змінює координати текстур. Використовується для трансформації (зміни позиції, масштабу і обертання) текстурних координат.

- Location: Зміщення текстури.
- Rotation: Обертання текстури.
- Scale: Масштабування текстури.

Image Texture - додає зображення як текстуру. Використовується для застосування зображень на поверхню об'єкта.

- Image: Вибір зображення.
- Interpolation: Інтерполяція пікселів (наприклад, лінійна або найближча).
- Extension: Як обробляти текстуру поза межами UV-координат (повторення, дзеркало, затиснення).

Mix Color (MixRGB) - змішує два кольори або текстури. Використовується для створення комбінованих кольорів або для плавного переходу між текстурами.

- Mix: Змішування за допомогою фактору (наприклад, лінійне змішування).
- Add: Додавання кольорів. Subtract: Віднімання одного кольору від іншого.
- Multiply: Множення кольорів.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Color Ramp - перетворює значення в кольори за допомогою градієнта. Використовується для створення градієнтів або для мапування значень (наприклад, висоти або температури) на кольори.

- Ramp: Набір кольорів для градієнта.
- Interpolation: Спосіб інтерполяції між кольорами (лінійна, б-смуга, карта).

Normal Map (Додаток Б) - застосовує нормал-карту для зміни нормалей поверхні. Використовується для додавання деталей до поверхні без зміни геометрії, що створює ілюзію складної структури [13].

- Color: Вхід нормал-карти.
- Strength: Сила впливу нормал-карти.

Principled BSDF - універсальний шейдер для створення реалістичних матеріалів. Використовується для створення різних типів матеріалів (металевих, діелектричних, прозорих тощо) з використанням фізично коректного рендерингу.

- Base Color: Основний колір матеріалу.
- Subsurface: Інтенсивність підповерхневого розсіювання.
- Metallic: Металевість матеріалу.
- Specular: Інтенсивність відблиску.
- Roughness: Шорсткість поверхні.
- Anisotropic: Анізотропія поверхні.
- Sheen: Інтенсивність оксамитового блиску.
- Clearcoat: Інтенсивність верхнього шару покриття.
- Transmission: Прозорість матеріалу.

Ці ноди є основними будівельними блоками для створення складних і реалістичних матеріалів у Blender.

Таким же чином було створенно чорно-білу текстуру гілок для ялинки, це потрібно щоб в ігровому рушії можна було задавати різні кольори гілкам дерева. Також потрібно для коректності роботи гілкам дерева створити свій окремий матеріал та налаштувати для нього цю текстуру (рис. 2.44).

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		54

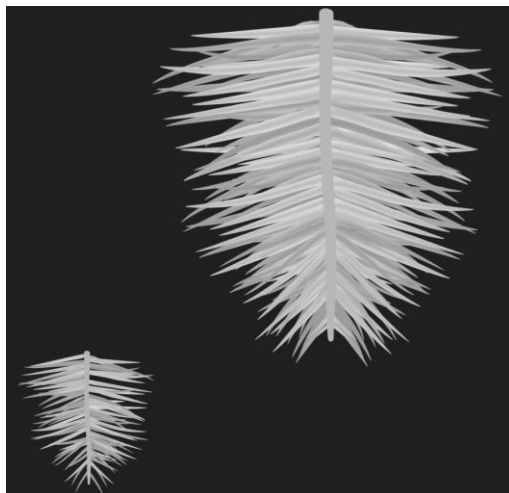


Рисунок 2.44 – Текстура для гілок дерева

2.4 Реалізація програмного коду для оптимізації геометрії

Для пришвидшення розробки комп'ютерних моделей було розроблено аддон для програми Blender з назвою GenerateLOD (Додаток А) який з допомогою налаштувань буде спрощувати геометрію 3D-моделі, тобто створювати LOD вибраної 3D-моделі [9, 20].

LOD (англ. Level of Detail) використовується в комп'ютерних іграх для оптимізації відображення об'єктів на різних відстанях від гравця. Замість того, щоб відображати деталізовану модель об'єкта навіть на великій відстані, LOD дозволяє використовувати менш деталізовані 3D-моделі або текстури, що дозволяє зменшити обчислювальні витрати і підвищити продуктивність комп'ютерної гри. Коли гравець зближується до об'єкта, LOD автоматично змінюється на більш деталізований варіант, щоб забезпечити більш реалістичний вигляд. Це допомагає підтримувати плавний геймплей і запобігає зайвому навантаженню на обчислювальні ресурси пристрою.

Для початку була створена структура аддону (рис. 2.45) в якому всі класи підпорядковуються файлу `__init__.py`.

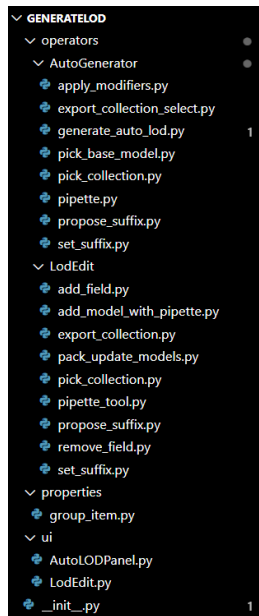


Рисунок 2.45 – Структура розробленого адону GenerateLOD

На рисунку 2.46 зображена структура класів та їх зв'язків у програмному забезпеченні. Діаграма відображає складну систему, в якій різні компоненти взаємодіють один з одним через методи та властивості, що дозволяє зрозуміти структуру програмного забезпечення та його взаємозв'язки.

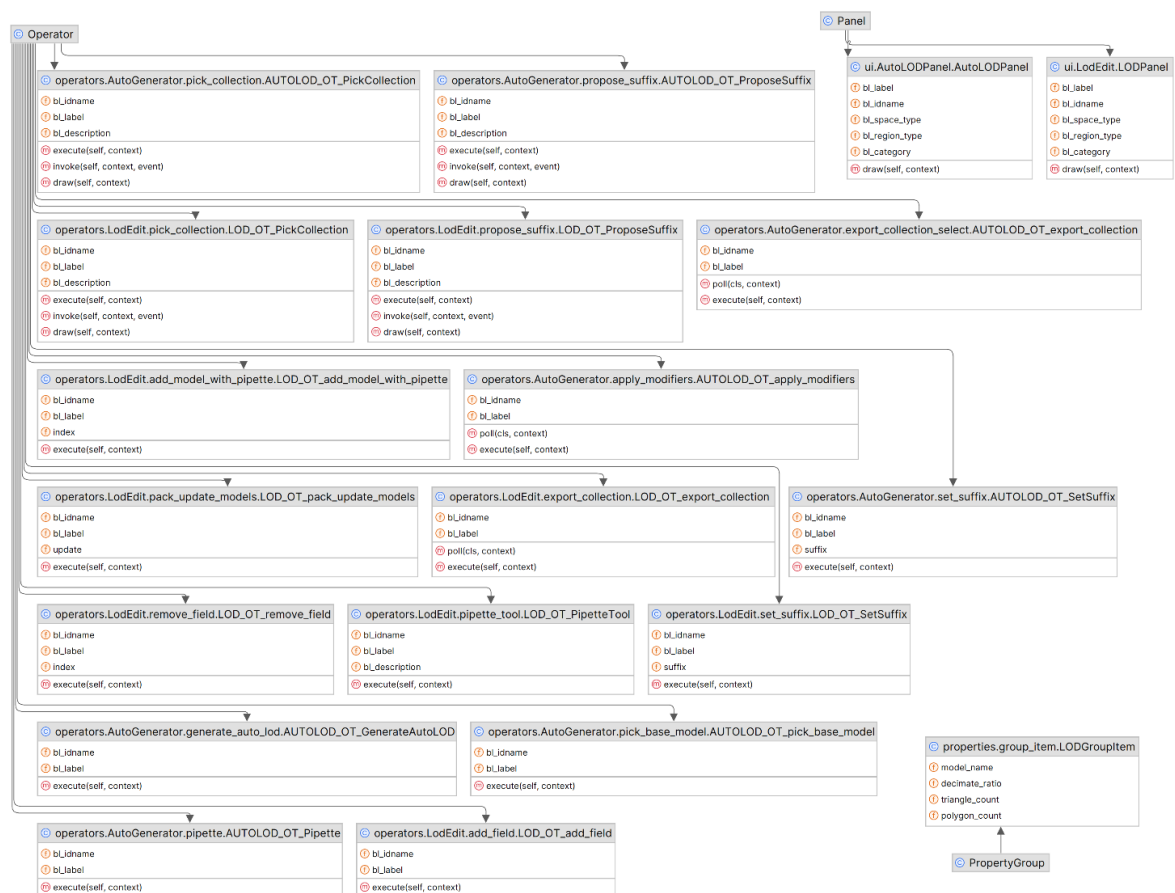


Рисунок 2.46 – Діаграма класів адону

Програма Blender використовує мову програмування Python для створення власних інструментів. Щоб програма ініціювала новий аддон використовується наступний код:

```
bl_info = {
    "name": "GenerateLOD",
    "author": "RitterBlack",
    "version": (1, 22),
    "blender": (2, 80, 0),
    "location": "View3D > N Panel",
    "description": "Create and update LOD groups with new models, applying suffixes on initial packing.",
    "category": "Development",
}
```

Щоб в програмі створити панель з інструментами потрібно написати код який створить на бічній панелі програми вікно для розміщення власних інструментів:

```
import bpy

class AutoLODPanel(bpy.types.Panel):
    bl_label = "LOD Generator"
    bl_idname = "SCENE_PT_auto_lod"
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = 'LOD edit'

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

        row = layout.row()
        row.prop(scene, "autolod_collection_name", text="Collection Name")
        row.operator("autolod.pick_collection", text="",
icon='COLLECTION_NEW')
```

Клас pick_collection використовується для пришвидшення вибору колекцій з доступних в програмі Blender, код для вибору колекції:

```
import bpy

class AUTOLOD_OT_PickCollection(bpy.types.Operator):
    """Operator to pick a collection for AutoLOD"""
    bl_idname = "autolod.pick_collection"
    bl_label = "Pick Collection for AutoLOD"
    bl_description = "Choose a collection for AutoLOD"

    def execute(self, context):
```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		57

```

        return {'FINISHED'}

    def invoke(self, context, event):
        wm = context.window_manager
        return wm.invoke_props_dialog(self, width=400)

    def draw(self, context):
        layout = self.layout
        layout.prop_search(context.scene, "autolod_collection_name",
bpy.data, "collections", text="Collection")

```

В аддоні розроблено функціонал який на основі базової 3D-моделі створює її копії тільки з меншою кількістю геометрії та переносить базову модель в колекцію для збереження, якщо такої колекції в проекті не існує аддон її створить, код для файлу generate_auto_lod.py:

```

backup_collection = bpy.data.collections.get("BACKUP") or
bpy.data.collections.new("BACKUP")
    if "BACKUP" not in bpy.context.scene.collection.children:
        bpy.context.scene.collection.children.link(backup_collection)
    if base_model.name not in backup_collection.objects:
        backup_collection.objects.link(base_model)

    for collection in base_model.users_collection:
        if collection != backup_collection:
            collection.objects.unlink(base_model)

    for i in range(0, lod_levels):
        model = base_model.copy()
        model.data = base_model.data.copy()
        model.name = f"{lod_base_model_name}{model_suffix}{i}" if
model_suffix else f"{lod_base_model_name}_{i}"

        target_collection = bpy.data.collections.get(collection_name) or
bpy.data.collections.new(collection_name)
        if collection_name not in bpy.context.scene.collection.children:
            bpy.context.scene.collection.children.link(target_collection
)
            target_collection.objects.link(model)

        decimate_modifier = model.modifiers.new(name="Decimate",
type='DECIMATE')
        decimate_modifier.ratio = max(0.1, 1 - i / lod_levels)

```

Також в файлі `__init__.py` потрібно ініціалізувати виклик класів, це робиться за допомогою такого коду:

```
def register():
    bpy.utils.register_class(AUTOLOD_OT_GenerateAutoLOD)
    bpy.utils.register_class(AutoLODPanel)
def unregister():
    bpy.utils.unregister_class(AUTOLOD_OT_GenerateAutoLOD)
    bpy.utils.unregister_class(AutoLODPanel)

if __name__ == "__main__":
    register()
```

Висновки до другого розділу

Продемонстровано діаграми процесу розробки графічних елементів для використання в ігровому додатку. Також описано процес розробки 3D-моделей та текстур для використання в ігровому додатку. Описано інструменти та аддони, які використовувалися для цього. Розроблено власний аддон на мові програмування Python для пришвидшення процесу створення оптимізації 3D-моделей.

Додатково, було проведено всебічне тестування та налагодження аддону для забезпечення його стабільної роботи та сумісності з іншими інструментами, що використовуються в процесі розробки. Аддон успішно інтегровано в робочий процес, що дозволило значно скоротити час на оптимізацію моделей без втрати якості. Було розроблено детальну документацію та інструкції з використання аддону, що полегшило його впровадження для членів команди.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		59

РОЗДІЛ 3. ОПИС РОБОТИ З ВІЗУАЛЬНИМИ КОМПОНЕНТАМИ ТА ЇХ ТЕСТУВАННЯ

3.1 Порядок оптимізації системи рівнів деталізації та експорту в ігровий рушій Unity

В поле Collection Name потрібно вписати або вибрати існуючу колекцію в яку буде створено моделі з зниженим рівнем деталізації, також потрібно вибрати ідентифікатор якості моделі, зазвичай використовують ідентифікатор LOD, за допомогою кнопки Propose Suffix можна швидко вибрати відповідний суфікс з запропонованих. В наступне поле потрібно за допомогою піпетки вибрати оригінальну моделі з максимальною якістю та вибрати яку кількість моделей потрібно створити. Після налаштування всіх параметрів панелі LOD Generator натискаємо кнопку AutoLOD, аддон перенесе оригінальну модель в колекцію BACKUP на зберігання та створить на її основі 3D-моделі з меншою якістю. Це потрібно для того щоб в грі з поступовим віддаленням від моделі знижувалася її якість що позитивно впливає на оптимізацію комп'ютерної гри (рис. 3.1).

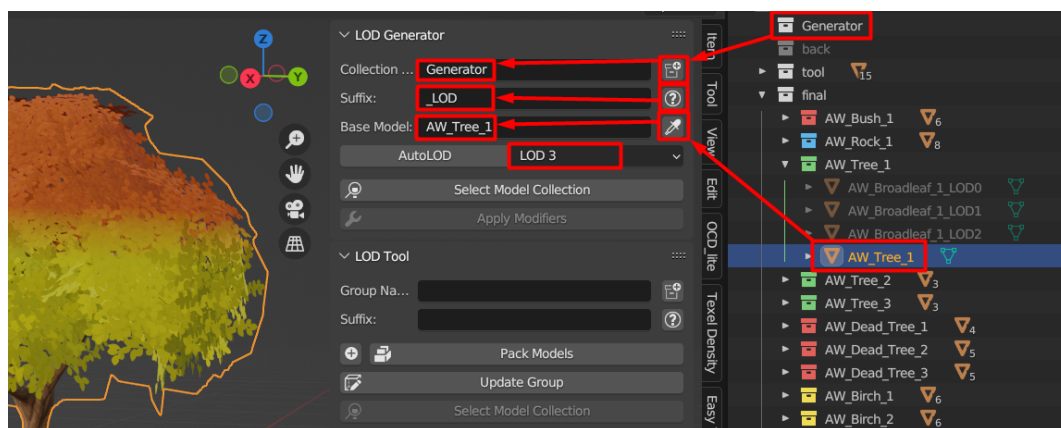


Рисунок 3.1 – Порядок налаштування панелі LOD Generator для створення рівнів деталізації

Наступним етапом буде налаштування створених об'єктів, це потрібно для того щоб прибрати можливі помилки алгоритму спрощення. Для цього потрібно на наступній панелі LOD Tool також вибрати колекцію в яку було створено моделі з зниженим рівнем деталізації. Після вибору колекції аддон автоматично створить поля з урахуванням порядку кожної моделі та налаштуваннями

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		60

важливим з яких є налаштування Decimate Ratio, також на кожному полі моделі є її статистика яка показує кількість полігонів та трикутників, це потрібно для візуального відображення різниці в рівнях деталізації (рис. 3.2).

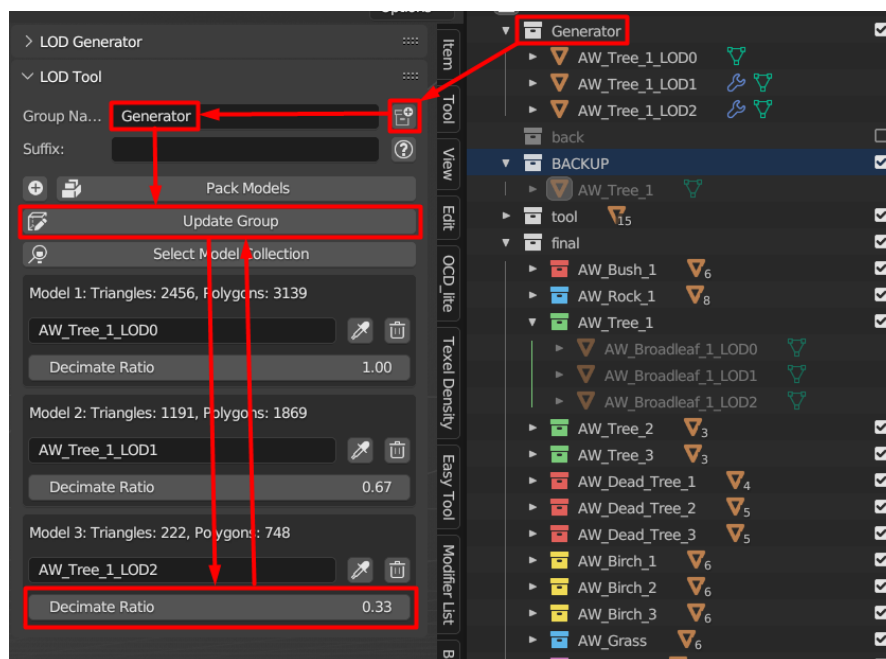


Рисунок 3.2 – Порядок налаштування панелі LOD Tool для внесення змін в створенні 3D-моделі

Налаштування Decimate Ratio базується на вбудованому модифікаторі Decimate та його режимі спрощення Collapse алгоритм роботи якого це притягування двох найближчих точок одна до одної.

Після налаштування всіх моделей які знаходяться в колекції потрібно застосувати модифікатори, для зручності на панелі LOD Generator існує кнопка Apply Modifiers яка застосує всі наявні на 3D-моделях модифікатори. Також для швидкості вибору моделей в групі існує кнопка Select Model Collection (рис.3.3).

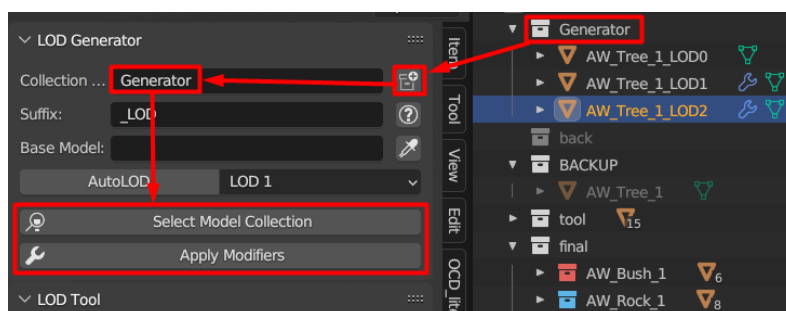


Рисунок 3.3 – Застосування всіх модифікаторів та швидкий вибір моделей в колекції

Після вибраних об'єктів потрібно в лівому верхньому куті натиснути кнопку File потім вибрати пункт Export та вибрати формат експорту. Потрібно вибрати формат FBX (.fbx) (рис. 3.4)

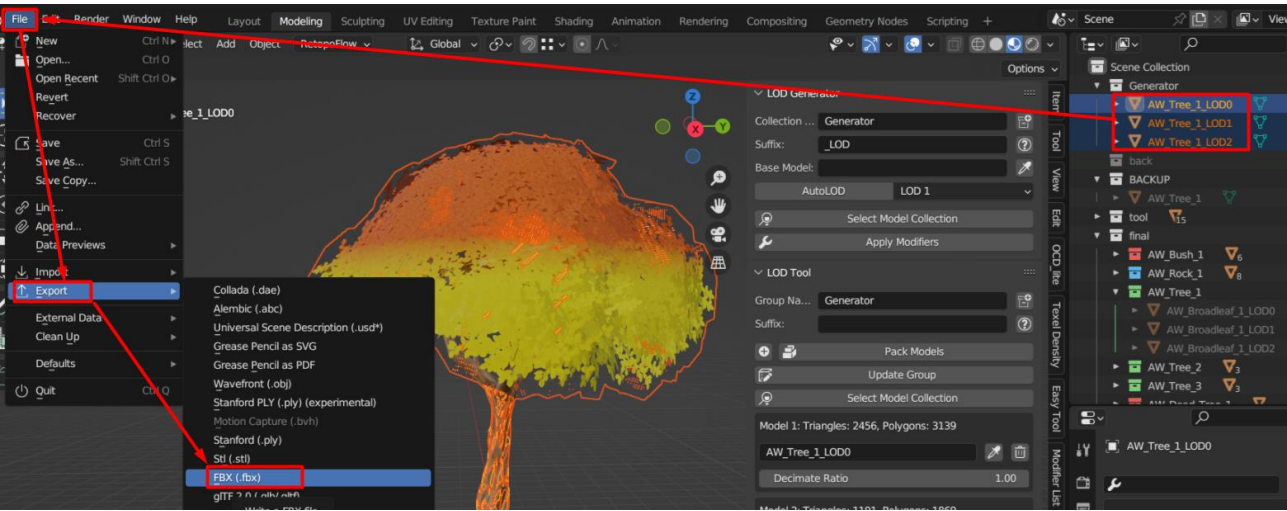


Рисунок 3.4 – Порядок дій експорту групи 3D-моделей

Після переходу до панелі експорту потрібно вибрати наступні параметри для коректної роботи в ігровому рушії Unity (рис. 3.5).

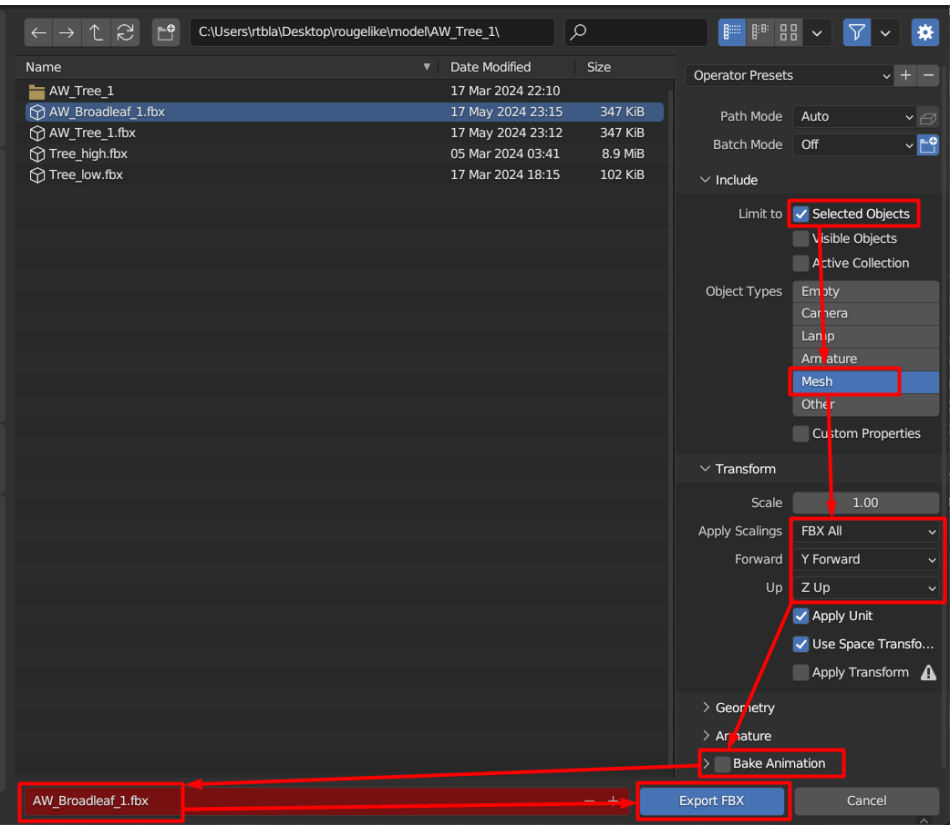


Рисунок 3.5 – Налаштування експорту для ігрового рушії Unity

3.2 Порядок встановлення, налаштування та тестування в ігровому рушії Unity

Система рівнів деталізації (англ. Level of Detail, LOD) використовується в ігровому рушії Unity для оптимізації продуктивності та збереження ресурсів обчислень, особливо у графічно насичених іграх та додатках. Основна мета LOD-системи полягає в тому, щоб динамічно змінювати деталізацію 3D-об'єктів залежно від їх відстані до камери [19].

Після налаштування рівнів деталізації буде різна кількість викликів текстур з ресурсів (Batches), чим більше викликів тим гірше оптимізація. Зменшити кількість викликів можна різними способами починаючи зменшенням кількості використовуваних текстур на карті закінчуючи програмним методом.

Перед імпортом моделей в ігровий рушій Unity було підготовлено декілька папок куди потрібно розсортувати файли для зручності [14]. Було створено батьківську папку з назвою AW_Fir_1 в яку було імпортовано ще декілька папок з назвою Materials, Model та Textures [15]. В папку Model було імпортовано 3D-модель в якій є налаштованні рівні деталізації (LOD) [16] (рис. 3.6).

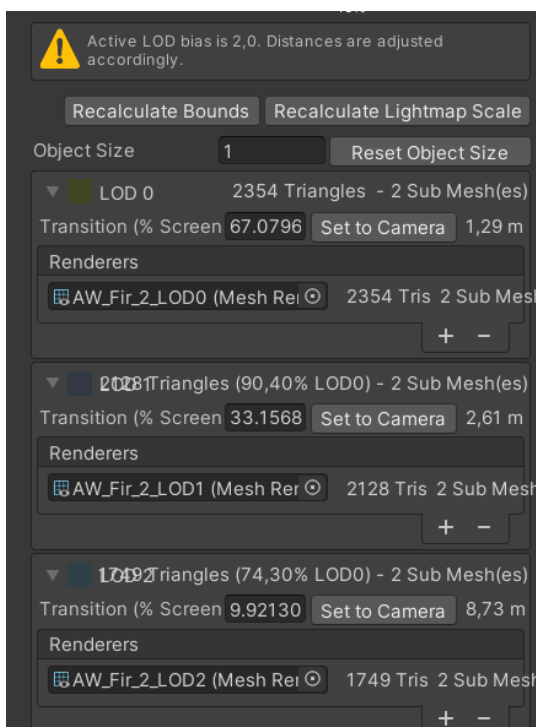


Рисунок 3.6 – Налаштованні рівні деталізації

Після налаштування рівнів деталізації необхідно встановити на шкалі (рис. 3.7) коректні значення дистанції віддалення від 3D-моделі, при якій буде змінюватися модель. Для цього потрібно враховувати розмір карти, густоту інших об'єктів, які знаходяться між камерою гравця та 3D-моделлю, а також туман, який можна використовувати для приховування переходу моделей з більш деталізованої на менш деталізовану.

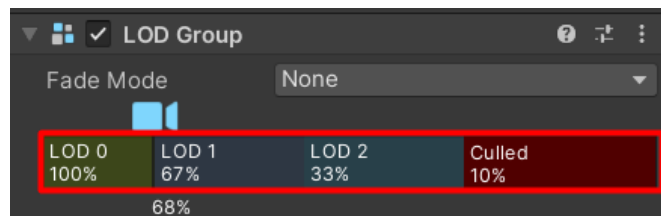


Рисунок 3.7 – Налаштування дальності переключення рівнів деталізації

Виставлені значення LOD об'єктів безпосередньо впливають на кількість Batches (пакетів обробки) в Unity, що, в свою чергу, впливає на продуктивність рендерингу.

LOD0 знаходиться на невеликій відстані від камери гравця, ігровий рушій Unity робить 17 викликів на відмальовування текстур та трикутників (рис. 3.8).



Рисунок 3.8 – Перевірка кількості викликів на відмальовування текстур

LOD0

Наступний LOD1 також робить 17 викликів на відмалювання але вже рендерить меншу кількість геометрії (рис. 3.9).

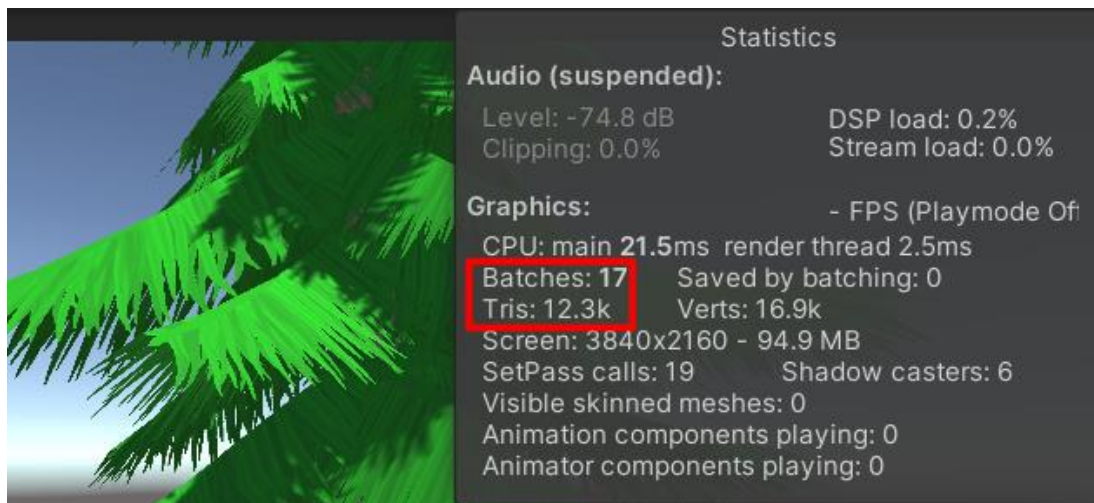


Рисунок 3.9 – Перевірка середнього по якості LOD1 на кількість викликів відмальовування текстур

Фінальний LOD2 робить 15 викликів на відмальовування текстур та геометрії (рис. 3.10).

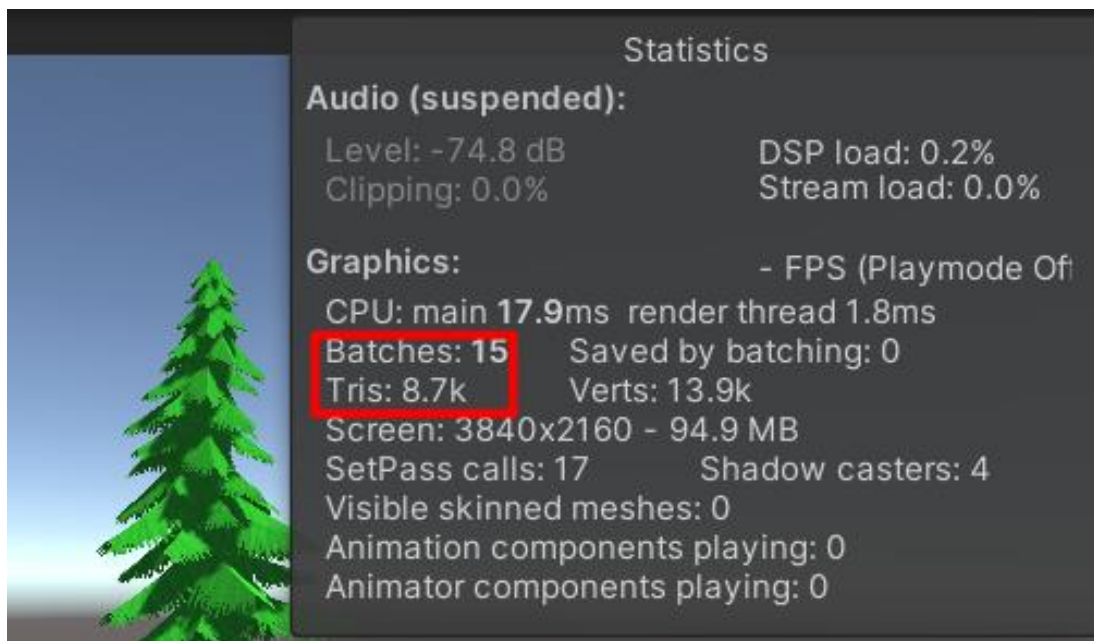


Рисунок 3.10 – Перевірка останнього по якості LOD2 на кількість викликів відмальовування текстур

Також при повному віддаленні від об'єкта він вимикається на сцені але постійно зберігається в пам'яті ігрового рушія для швидкого виставлення його сцену якщо гравець буде близько підходити до об'єкта (рис. 3.11).

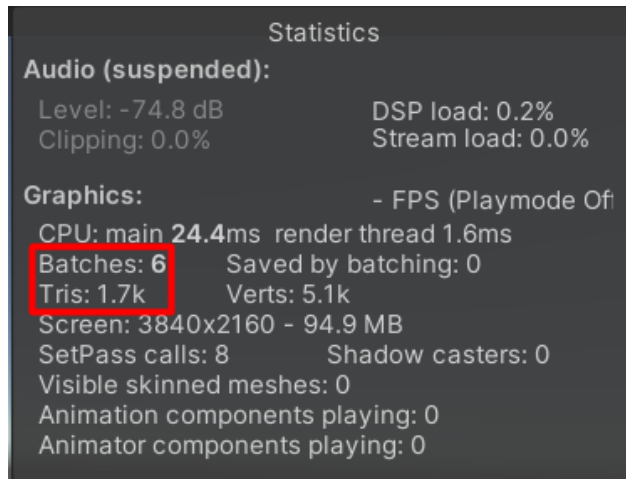


Рисунок 3.11 – Результат при повному віддаленні від об’єкта

Через вбудовану систему моніторингу використання ресурсів ігрового рушія Unity було проведено аналіз що дає детальну інформацію про різні аспекти під час тестування. Було отримано детальний графік рендерингу графіки та навантаження на центральний процесор (рис. 3.12).

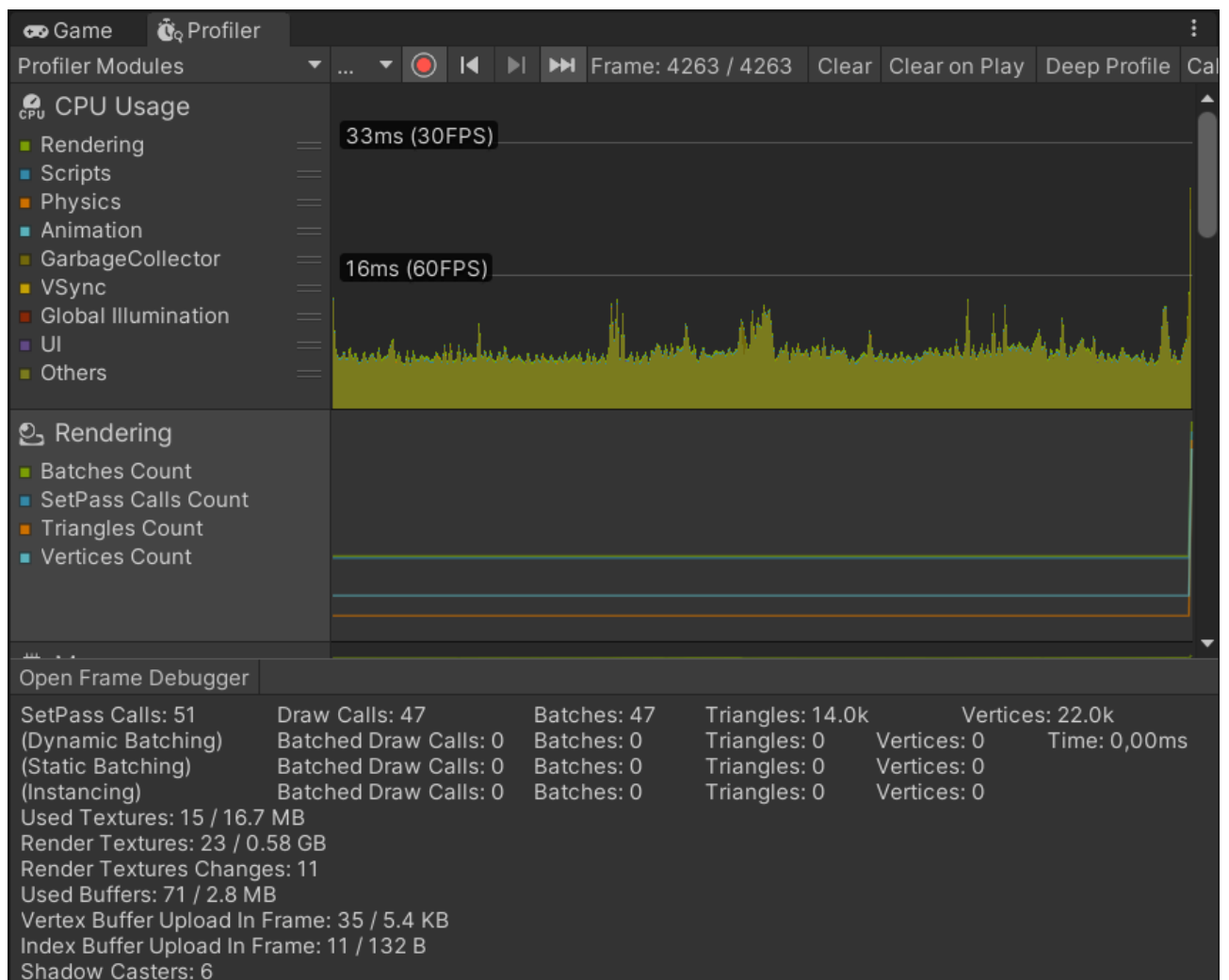


Рисунок 3.12 – Дані навантаження в режимі Profiler

Висновки до третього розділу

Продемонстровано процес оптимізації системи рівнів деталізації. Також показано процес експорту готових 3D-моделей з програми Blender та їх імпорт в ігровий рушій Unity. Показано процес налаштування LOD-system в ігровому рушії. Проведено аналітику використання ресурсів рушія за допомогою вбудованої системи моніторингу Profiler в Unity що в свою чергу показало що це є критично важливим етапом для забезпечення високої продуктивності ігрових проєктів. Profiler дозволяє детально аналізувати роботу рушія, виявляти вузькі місця та оптимізувати графіку, що сприяє створенню більш ефективних та якісних ігор.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		67

Висновки

У ході розробки кваліфікаційної роботи було успішно створено концептуальну та технічну основу для комп'ютерної гри в жанрі Rouge-lite. Аналіз трьох відомих ігор (Borderlands 2, Roboquest, Tiny Tina's Wonderlands) допоміг визначити сильні та слабкі сторони, що сприяло вдосконаленню комп'ютерної гри. Було розроблено концептуальний документ, який окреслив жанр, цільову аудиторію, ключові особливості, геймплей, сюжет та платформу для комп'ютерної гри.

Особлива увага приділялася вибору інструментів для створення графіки, зокрема Blender та Substance 3D Painter. Було розроблено діаграми процесу створення графічних елементів, 3D-моделей та текстур. Також було створено власний аддон на мові Python, що значно пришвидшив процес оптимізації 3D-моделей. Аддон успішно інтегровано у робочий процес, забезпечено його стабільну роботу та сумісність з іншими інструментами. Детальна документація полегшила впровадження аддону для команди.

Процес оптимізації системи рівнів деталізації включав експорт моделей з Blender, імпорт в Unity та налаштування LOD-системи. Використання системи моніторингу Profiler в Unity виявило критично важливі аспекти для забезпечення високої продуктивності в комп'ютерній грі. Також Profiler допоміг детально аналізувати роботу рушія, виявляти вузькі місця та оптимізувати графіку, що підвищило ефективність та якість гри.

Таким чином, кваліфікаційна робота продемонструвала ефективний підхід до розробки комп'ютерної гри в жанрі Rouge-lite. Ретельний аналіз існуючих ігор, вибір оптимальних інструментів, розробка та інтеграція власних аддонів, а також оптимізація графіки та ресурсів рушія Unity дозволили створити якісний продукт. Отримані знання та навички будуть корисними у подальших проектах, підвищуючи професійну підготовку та конкурентоспроможність на ринку ігрової індустрії.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		68

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Borderlands 2 [Електронний ресурс] // Valve Corporation. – 2012. – Режим доступу до ресурсу: <https://store.steampowered.com/agecheck/app/49520/>.
2. Roboquest [Електронний ресурс] // Steam. – 2023. – Режим доступу до ресурсу: <https://store.steampowered.com/app/692890/Roboquest/>.
3. Tiny Tina's Wonderlands [Електронний ресурс] // Steam. – 2022. – Режим доступу до ресурсу: https://store.steampowered.com/app/1286680/Tiny_Tinas_Wonderlands/.
4. RetopoFlow [Електронний ресурс] // BlenderMarket. – 2018. – Режим доступу до ресурсу: <https://blendermarket.com/products/retopoflow>.
5. TexTools [Електронний ресурс] // franMarz. – 2024. – Режим доступу до ресурсу: <https://github.com/franMarz/TexTools-Blender>.
6. UV Toolkit [Електронний ресурс] // oRazeD. – 2024. – Режим доступу до ресурсу: <https://github.com/oRazeD/UVToolkit>.
7. UVPackmaster3 [Електронний ресурс] // glukoz. – 2024. – Режим доступу до ресурсу: <https://blendermarket.com/products/uvpackmaster>.
8. Node Wrangler [Електронний ресурс] // Blender. – 2024. – Режим доступу до ресурсу: https://docs.blender.org/manual/en/latest/addons/node/node_wrangler.html.
9. Козуб, Д. П. "Використання скриптів в blender для автоматизації процесу створення 3d-моделей та анімації." (2023).
10. Stam, Jos. "Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values." Seminal Graphics Papers: Pushing the Boundaries, Volume 2. 2023. 139-148.
11. Real-Time Rendering [Електронний ресурс] // Tomas Akenine-Möller, Eric Haines, Naty Hoffman. – 2018. – Режим доступу до ресурсу: https://www.realtimerendering.com/Real-Time_Rendering_4th-TOC_Preface_Intro_Bib_Index.pdf.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		69

12. Blender GitHub [Електронний ресурс] // Blender Authors. – 2024. – Режим доступу до ресурсу: <https://github.com/blender/blender>.
13. Посилання на код ноди Normal Map [Електронний ресурс] // Blender Authors. – 2024. – Режим доступу до ресурсу: https://github.com/blender/blender/blob/main/source/blender/nodes/shader/nodes/node_shader_normal_map.cc.
14. Імпортування ассетів [Електронний ресурс] // Unity. – 2024. – Режим доступу до ресурсу: <https://docs.unity3d.com/Manual/ImportingAssets.html>.
15. Налаштування 3D-моделей в ігровому рушії Unity [Електронний ресурс] // Unity. – 2024. – Режим доступу до ресурсу: <https://docs.unity3d.com/Manual/mesh.html>.
16. Опис роботи з текстурами в ігровому рушії Unity [Електронний ресурс] // Unity. – 2024. – Режим доступу до ресурсу: <https://docs.unity3d.com/Manual/Textures.html>.
17. UVunwrapping [Електронний ресурс] // Blender Manual. – 2024. – Режим доступу до ресурсу: <https://docs.blender.org/manual/en/latest/modeling/meshes/uv/unwrapping/introduction.html>.
18. Категорія моделювання в Blender Manual [Електронний ресурс] // Blender Manual. – 2024. – Режим доступу до ресурсу: <https://docs.blender.org/manual/en/latest/modeling/index.html>.
19. Документація LOD-system [Електронний ресурс] // Unity. – 2024. – Режим доступу до ресурсу: <https://docs.unity3d.com/Manual/LevelOfDetail.html>.
20. Документація по створенню власних скриптів для додатку Blender [Електронний ресурс] // Blender Manual. – 2024. – Режим доступу до ресурсу: <https://docs.blender.org/manual/en/latest/advanced/scripting/index.html>.

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				70
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				71
Змн.	Арк.	№ докум.	Підпис	Дата		

__init__.py

```

bl_info = {
    "name": "GenerateLOD",
    "author": "RitterBlack",
    "version": (1, 22),
    "blender": (2, 80, 0),
    "location": "View3D > N Panel",
    "description": "Create and update LOD groups with new models, applying suffixes
on initial packing.",
    "category": "Development",
}

import bpy
from .properties.group_item import LODGroupItem

from .ui.LodEdit import LODPanel
from .ui.AutoLODPanel import AutoLODPanel

from .operators.LodEdit.add_model_with_pipette import LOD_OT_add_model_with_pipette
from .operators.LodEdit.add_field import LOD_OT_add_field
from .operators.LodEdit.remove_field import LOD_OT_remove_field
from .operators.LodEdit.pack_update_models import LOD_OT_pack_update_models
from .operators.LodEdit.pipette_tool import LOD_OT_PipetteTool
from .operators.LodEdit.pick_collection import LOD_OT_PickCollection
from .operators.LodEdit.export_collection import LOD_OT_export_collection
from .operators.LodEdit.propose_suffix import LOD_OT_ProposeSuffix
from .operators.LodEdit.set_suffix import LOD_OT_SetSuffix

from .operators.AutoGenerator.apply_modifiers import AUTOLOD_OT_apply_modifiers
from .operators.AutoGenerator.export_collection_select import
AUTOLOD_OT_export_collection
from .operators.AutoGenerator.generate_auto_lod import AUTOLOD_OT_GenerateAutoLOD
from .operators.AutoGenerator.pick_base_model import AUTOLOD_OT_pick_base_model
from .operators.AutoGenerator.pick_collection import AUTOLOD_OT_PickCollection
from .operators.AutoGenerator.pipette import AUTOLOD_OT_Pipette
from .operators.AutoGenerator.propose_suffix import AUTOLOD_OT_ProposeSuffix
from .operators.AutoGenerator.set_suffix import AUTOLOD_OT_SetSuffix

def get_lod_levels(self, context):
    return [(str(i), f"LOD {i}", f"Generate {i} additional LOD levels") for i in
range(1, 11)]

def register():
    bpy.utils.register_class(AUTOLOD_OT_Pipette)
    bpy.utils.register_class(AUTOLOD_OT_PickCollection)
    bpy.utils.register_class(AUTOLOD_OT_ProposeSuffix)
    bpy.utils.register_class(AUTOLOD_OT_SetSuffix)
    bpy.utils.register_class(AUTOLOD_OT_GenerateAutoLOD)

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

bpy.utils.register_class(AUTOLOD_OT_pick_base_model)
bpy.utils.register_class(AutoLODPanel)
bpy.utils.register_class(AUTOLOD_OT_export_collection)
bpy.utils.register_class(AUTOLOD_OT_apply_modifiers)
bpy.types.Scene.autolod_model_name = bpy.props.StringProperty(
    name="Model Name",
    description="Name of the model for AutoLOD"
)
bpy.types.Scene.autolod_collection_name = bpy.props.StringProperty(
    name="Collection Name",
    description="Name of the collection for AutoLOD"
)
bpy.types.Scene.autolod_model_suffix = bpy.props.StringProperty(
    name="AutoLOD Model Suffix",
    description="Suffix for AutoLOD models"
)
bpy.types.Scene.auto_lod_level = bpy.props.EnumProperty(
    name="Auto LOD Level",
    description="Choose the number of LOD levels to generate",
    items=get_lod_levels
)

bpy.utils.register_class(LOD_OT_ProposeSuffix)
bpy.utils.register_class(LOD_OT_SetSuffix)
bpy.utils.register_class(LODGroupItem)
bpy.types.Scene.lod_group_items =
bpy.props.CollectionProperty(type=LODGroupItem)
bpy.utils.register_class(LOD_OT_add_model_with_pipette)
bpy.utils.register_class(LOD_OT_add_field)
bpy.utils.register_class(LOD_OT_remove_field)
bpy.utils.register_class(LOD_OT_pack_update_models)
bpy.utils.register_class(LODPanel)
bpy.utils.register_class(LOD_OT_PickCollection)
bpy.utils.register_class(LOD_OT_export_collection)
bpy.types.Scene.lod_group_items =
bpy.props.CollectionProperty(type=LODGroupItem)
bpy.types.Scene.lod_group_name = bpy.props.StringProperty(name="Group Name")
bpy.types.Scene.model_suffix = bpy.props.StringProperty(name="Suffix")
bpy.types.Scene.collection_name = bpy.props.StringProperty(name="Collection
Name")
bpy.types.WindowManager.chosen_collection = bpy.props.StringProperty()

def unregister():
    bpy.utils.unregister_class(AUTOLOD_OT_Pipette)
    bpy.utils.unregister_class(AUTOLOD_OT_PickCollection)
    bpy.utils.unregister_class(AUTOLOD_OT_ProposeSuffix)
    bpy.utils.unregister_class(AUTOLOD_OT_SetSuffix)
    bpy.utils.unregister_class(AUTOLOD_OT_GenerateAutoLOD)
    bpy.utils.unregister_class(AUTOLOD_OT_pick_base_model)
    bpy.utils.unregister_class(AutoLODPanel)

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```

bpy.utils.unregister_class(AUTOLOD_OT_export_collection)
bpy.utils.unregister_class(AUTOLOD_OT_apply_modifiers)
del bpy.types.Scene.autolod_model_name
del bpy.types.Scene.autolod_collection_name
del bpy.types.Scene.autolod_model_suffix
del bpy.types.Scene.auto_lod_level

bpy.utils.unregister_class(LOD_OT_ProposeSuffix)
bpy.utils.unregister_class(LOD_OT_SetSuffix)
del bpy.types.Scene.lod_group_items
bpy.utils.unregister_class(LODGroupItem)
bpy.utils.unregister_class(LOD_OT_add_model_with_pipette)
bpy.utils.unregister_class(LOD_OT_add_field)
bpy.utils.unregister_class(LOD_OT_remove_field)
bpy.utils.unregister_class(LOD_OT_pack_update_models)
bpy.utils.unregister_class(LODPanel)
bpy.utils.unregister_class(LOD_OT_PickCollection)
bpy.utils.unregister_class(LOD_OT_export_collection)
del bpy.types.Scene.lod_group_items
del bpy.types.Scene.lod_group_name
del bpy.types.Scene.model_suffix
del bpy.types.Scene.collection_name
del bpy.types.WindowManager.chosen_collection

if __name__ == "__main__":
    register()

```

ui\LodEdit.py

```

import bpy

class LODPanel(bpy.types.Panel):
    bl_label = "LOD Tool"
    bl_idname = "SCENE_PT_lod"
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = 'LOD edit'

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

        row = layout.row()
        row.prop(scene, "lod_group_name")
        row.operator("lod.pick_collection", text="", icon="COLLECTION_NEW")

        row = layout.row()
        row.prop(scene, "model_suffix")
        row.operator("lod.propose_suffix", text="", icon="QUESTION")

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

row = layout.row(align=True)
row = layout.row(align=True)

split = row.split(align=True)

buttons_row = split.row(align=True)
buttons_row.operator("lod.add_field", text="", icon='PLUS')

buttons_row.separator(factor=1)

buttons_row.operator("lod.pack_update_models", text="Pack Models",
icon='PACKAGE').update = False
layout.operator("lod.pack_update_models", text="Update Group",
icon='MOD_LINEARTEXT').update = True

layout.operator("lod.export_collection_select", text="Select Model
Collection", icon='ZOOM_SELECTED')

for i, item in enumerate(scene.lod_group_items):
    box = layout.box()

    row = box.row()
    row.label(text=f"Model {i+1}: Triangles: {item.triangle_count},
Polygons: {item.polygon_count}")

    row = box.row()
    row.prop(item, "model_name", text="")
    row.operator("lod.add_model_with_pipette", text="",
icon='EYEDROPPER').index = i
    row.operator("lod.remove_field", text="", icon='TRASH').index = i

    box.prop(item, "decimate_ratio", text="Decimate Ratio")

```

ui\AutoLODPanel.py

```

import bpy

class AutoLODPanel(bpy.types.Panel):
    bl_label = "LOD Generator"
    bl_idname = "SCENE_PT_auto_lod"
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = 'LOD edit'

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

        row = layout.row()
        row.prop(scene, "autolod_collection_name", text="Collection Name")

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		75

```

row.operator("autolod.pick_collection", text="", icon='COLLECTION_NEW')

row = layout.row()
row.prop(scene, "autolod_model_suffix", text="Suffix")
row.operator("autolod.propose_suffix", text="", icon='QUESTION')

row = layout.row()
row.prop(scene, "autolod_model_name", text="Base Model")
row.operator("autolod.pipette", text="", icon='EYEDROPPER')

row = layout.row(align=True)
row.operator("autolod.generate_auto_lod", text="AutoLOD")
row.prop(scene, "auto_lod_level", text="")

row = layout.row(align=True)
layout.operator("autolod.export_collection_select", text="Select Model
Collection", icon='ZOOM_SELECTED')

layout.operator("autolod.apply_modifiers", text="Apply Modifiers",
icon='MODIFIER')

```

properties\group_item.py

```

import bpy

class LODGroupItem(bpy.types.PropertyGroup):
    model_name: bpy.props.StringProperty(name="Model Name")
    decimate_ratio: bpy.props.FloatProperty(name="Decimate Ratio", default=1.0,
min=0.0, max=1.0)

    triangle_count: bpy.props.IntProperty(name="Triangle Count", default=0)
    polygon_count: bpy.props.IntProperty(name="Polygon Count", default=0)

```

operators\LodEdit\set_suffix.py

```

import bpy

class LOD_OT_SetSuffix(bpy.types.Operator):
    bl_idname = "lod.set_suffix"
    bl_label = "Set Suffix"

    suffix: bpy.props.StringProperty()

    def execute(self, context):
        context.scene.model_suffix = self.suffix
        return {'FINISHED'}

```

operators\LodEdit\remove_field.py

```

import bpy

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

class LOD_OT_remove_field(bpy.types.Operator):
    bl_idname = "lod.remove_field"
    bl_label = "Remove Field"
    index: bpy.props.IntProperty()

    def execute(self, context):
        context.scene.lod_group_items.remove(self.index)
        return {'FINISHED'}

```

operators\LodEdit\propose_suffix.py

```

import bpy

class LOD_OT_ProposeSuffix(bpy.types.Operator):
    bl_idname = "lod.propose_suffix"
    bl_label = "Propose Suffix"
    bl_description = "Choose from proposed suffixes"

    def execute(self, context):
        return {'FINISHED'}

    def invoke(self, context, event):
        return context.window_manager.invoke_popup(self)

    def draw(self, context):
        layout = self.layout
        box = layout.box()
        row = box.row()
        row.label(text="Select a suffix:")
        proposed_suffixes = ["_LOD", "_low", "_med", "_high"]
        for suffix in proposed_suffixes:
            row.operator("lod.set_suffix", text=suffix).suffix = suffix

```

operators\LodEdit\pipette_tool.py

```

import bpy

class LOD_OT_PipetteTool(bpy.types.Operator):
    bl_idname = "lod.pipette_tool"
    bl_label = "Pipette Tool"
    bl_description = "Use pipette tool to pick model"

    def execute(self, context):
        bpy.ops.object.select_all(action='DESELECT')
        bpy.context.scene.tool_settings.use_transform_skip_children = True
        bpy.context.scene.tool_settings.transform_pivot_point = 'CURSOR'
        bpy.context.scene.tool_settings.use_transform_data_origin = False
        bpy.ops.object.select_by_type(extend=False, type='MESH')
        bpy.ops.view3d.snap_cursor_to_selected()
        return {'FINISHED'}

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		77

operators\LodEdit\pick_collection.py

```
import bpy

class LOD_OT_PickCollection(bpy.types.Operator):
    bl_idname = "lod.pick_collection"
    bl_label = "Pick Collection"
    bl_description = "Choose collection and populate models with decimate ratio if applicable"

    def execute(self, context):
        collection_name = context.scene.lod_group_name
        collection = bpy.data.collections.get(collection_name)

        if collection:
            context.scene.lod_group_items.clear()

            for obj in collection.objects:
                item = context.scene.lod_group_items.add()
                item.model_name = obj.name

                decimate_mod = next((mod for mod in obj.modifiers if mod.type ==
'DECIMATE'), None)
                if decimate_mod:
                    item.decimate_ratio = decimate_mod.ratio
                else:
                    item.decimate_ratio = 1.0

            return {'FINISHED'}

    def invoke(self, context, event):
        wm = context.window_manager
        return wm.invoke_props_dialog(self, width=400)

    def draw(self, context):
        layout = self.layout
        layout.prop_search(context.scene, "lod_group_name", bpy.data,
"collections", text="Collection")
```

operators\LodEdit\pack_update_models.py

```
import bpy
import re

class LOD_OT_pack_update_models(bpy.types.Operator):
    bl_idname = "lod.pack_update_models"
    bl_label = "Pack / Update Models"
    update: bpy.props.BoolProperty()
```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

def execute(self, context):
    group_name = context.scene.lod_group_name
    suffix = context.scene.model_suffix
    items = context.scene.lod_group_items
    collection = bpy.data.collections.get(group_name)

    if not collection:
        collection = bpy.data.collections.new(group_name)
        context.scene.collection.children.link(collection)

    existing_model_names = {item.model_name for item in items}
    for obj in collection.objects:
        if obj.name not in existing_model_names:
            new_item = items.add()
            new_item.model_name = obj.name
            new_item.decimate_ratio = 1.0

    for i, item in enumerate(items):
        original_name = re.sub(r'\.\d+$', '', item.model_name)
        proposed_name = f"{original_name}{suffix}{i}" if suffix else
f"{original_name}{i}"

        obj = bpy.data.objects.get(item.model_name)
        if obj:
            if obj.name not in {o.name for o in collection.objects} or not
obj.name.endswith(f"{suffix}{i}"):
                obj.name = proposed_name
                if obj.name not in {o.name for o in collection.objects}:
                    collection.objects.link(obj)
                item.model_name = obj.name

            decimate_modifier = obj.modifiers.get("Decimate")
            if item.decimate_ratio != 1.0:
                if not decimate_modifier:
                    decimate_modifier = obj.modifiers.new(name="Decimate",
type='DECIMATE')
                decimate_modifier.ratio = item.decimate_ratio
            else:
                if decimate_modifier:
                    obj.modifiers.remove(decimate_modifier)

            if obj.type == 'MESH':
                context.view_layer.update()

            evaluated_obj =
obj.evaluated_get(context.evaluated_depsgraph_get())
            mesh_data = evaluated_obj.to_mesh()

            item.triangle_count = sum(1 for poly in mesh_data.polygons if
len(poly.vertices) == 3)
            item.polygon_count = len(mesh_data.polygons)

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

        evaluated_obj.to_mesh_clear()

    return {'FINISHED'}

```

operators\LodEdit\export_collection.py

```

import bpy

class LOD_OT_export_collection(bpy.types.Operator):
    """Select all objects in the specified collection for export"""
    bl_idname = "lod.export_collection_select"
    bl_label = "Select Collection Objects"

    @classmethod
    def poll(cls, context):
        return context.scene.lod_group_name != ''

    def execute(self, context):
        collection_name = context.scene.lod_group_name
        collection = bpy.data.collections.get(collection_name)

        if collection is None:
            self.report({'ERROR'}, f"Collection '{collection_name}' not found.")
            return {'CANCELLED'}

        bpy.ops.object.select_all(action='DESELECT')

        for obj in collection.objects:
            obj.select_set(True)
            context.view_layer.objects.active = obj

        self.report({'INFO'}, f"All objects in '{collection_name}' are selected.")
        return {'FINISHED'}

```

operators\LodEdit\add_model_with_pipette.py

```

import bpy

class LOD_OT_add_model_with_pipette(bpy.types.Operator):
    bl_idname = "lod.add_model_with_pipette"
    bl_label = "Pick Model"
    index: bpy.props.IntProperty()

    def execute(self, context):
        lod_items = context.scene.lod_group_items
        active_obj = context.view_layer.objects.active
        if active_obj and active_obj.type == 'MESH':
            lod_item = lod_items[self.index]
            lod_item.model_name = active_obj.name

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

        mesh_data = active_obj.data
        lod_item.triangle_count = sum(1 for poly in mesh_data.polygons if
len(poly.vertices) == 3)
        lod_item.polygon_count = len(mesh_data.polygons)

    return {'FINISHED'}

```

operators\LodEdit\add_field.py

```

import bpy

class LOD_OT_add_field(bpy.types.Operator):
    bl_idname = "lod.add_field"
    bl_label = "Add Field"

    def execute(self, context):
        lod_items = context.scene.lod_group_items
        lod_items.add()

        if len(lod_items) > 1:
            decimate_ratio = lod_items[-2].decimate_ratio
            decimate_ratio = max(0.1, decimate_ratio)
        else:
            decimate_ratio = 1.0

        lod_items[-1].decimate_ratio = decimate_ratio

    return {'FINISHED'}

```

operators\AutoGenerator\set_suffix.py

```

import bpy

class AUTOLOD_OT_SetSuffix(bpy.types.Operator):
    bl_idname = "autolod.set_suffix"
    bl_label = "Set Suffix"
    suffix: bpy.props.StringProperty()

    def execute(self, context):
        context.scene.autolod_model_suffix = self.suffix
        return {'FINISHED'}

```

operators\AutoGenerator\propose_suffix.py

```

import bpy

class AUTOLOD_OT_ProposeSuffix(bpy.types.Operator):
    bl_idname = "autolod.propose_suffix"
    bl_label = "Propose Suffix"
    bl_description = "Choose from proposed suffixes for AutoLOD"

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

def execute(self, context):
    return {'FINISHED'}

def invoke(self, context, event):
    return context.window_manager.invoke_popup(self)

def draw(self, context):
    layout = self.layout
    box = layout.box()
    row = box.row()
    row.label(text="Select a suffix:")
    proposed_suffixes = ["_LOD", "_low", "_med", "_high"]
    for suffix in proposed_suffixes:
        row.operator("autolod.set_suffix", text=suffix).suffix = suffix

```

operators\AutoGenerator\pipette.py

```

import bpy

class AUTOLOD_OT_Pipette(bpy.types.Operator):
    bl_idname = "autolod.pipette"
    bl_label = "Pick Model with Pipette"

    def execute(self, context):
        active_obj = context.view_layer.objects.active
        if active_obj and active_obj.type == 'MESH':
            context.scene.autolod_model_name = active_obj.name
            self.report({'INFO'}, f"Model {active_obj.name} selected")
        else:
            self.report({'ERROR'}, "No active mesh object selected")
        return {'FINISHED'}

```

operators\AutoGenerator\pick_collection.py

```

import bpy

class AUTOLOD_OT_PickCollection(bpy.types.Operator):
    """Operator to pick a collection for AutoLOD"""
    bl_idname = "autolod.pick_collection"
    bl_label = "Pick Collection for AutoLOD"
    bl_description = "Choose a collection for AutoLOD"

    def execute(self, context):

        return {'FINISHED'}

    def invoke(self, context, event):
        wm = context.window_manager
        return wm.invoke_props_dialog(self, width=400)

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```

def draw(self, context):
    layout = self.layout
    layout.prop_search(context.scene, "autolod_collection_name", bpy.data,
"collections", text="Collection")

```

operators\AutoGenerator\pick_base_model.py

```

import bpy

class AUTOLOD_OT_pick_base_model(bpy.types.Operator):
    bl_idname = "autolod.pick_base_model"
    bl_label = "Pick Base Model"

    def execute(self, context):
        active_obj = context.view_layer.objects.active
        if active_obj and active_obj.type == 'MESH':
            context.scene.lod_base_model_name = active_obj.name
        else:
            self.report({'ERROR'}, "No mesh object active. Please select a mesh
object.")
            return {'CANCELLED'}

        return {'FINISHED'}

```

operators\AutoGenerator\propose_suffix.py

```

import bpy

class AUTOLOD_OT_GenerateAutoLOD(bpy.types.Operator):
    bl_idname = "autolod.generate_auto_lod"
    bl_label = "Generate Auto LOD"

    def execute(self, context):
        scene = context.scene
        lod_base_model_name = scene.autolod_model_name
        lod_levels = int(scene.auto_lod_level)
        model_suffix = scene.autolod_model_suffix
        collection_name = scene.autolod_collection_name

        base_model = bpy.data.objects.get(lod_base_model_name)
        if not base_model:
            self.report({'ERROR'}, "Base model not found.")
            return {'CANCELLED'}

        backup_collection = bpy.data.collections.get("BACKUP") or
bpy.data.collections.new("BACKUP")
        if "BACKUP" not in bpy.context.scene.collection.children:
            bpy.context.scene.collection.children.link(backup_collection)
        if base_model.name not in backup_collection.objects:

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

        backup_collection.objects.link(base_model)

    for collection in base_model.users_collection:
        if collection != backup_collection:
            collection.objects.unlink(base_model)

    for i in range(0, lod_levels):
        model = base_model.copy()
        model.data = base_model.data.copy()
        model.name = f"{lod_base_model_name}{model_suffix}{i}" if model_suffix
    else f"{lod_base_model_name}_{i}"

    target_collection = bpy.data.collections.get(collection_name) or
bpy.data.collections.new(collection_name)
    if collection_name not in bpy.context.scene.collection.children:
        bpy.context.scene.collection.children.link(target_collection)
    target_collection.objects.link(model)

    decimate_modifier = model.modifiers.new(name="Decimate",
type='DECIMATE')
    decimate_modifier.ratio = max(0.1, 1 - i / lod_levels)

    return {'FINISHED'}

```

operators\AutoGenerator\export_collection_select.py

```

import bpy

class AUTOLOD_OT_export_collection(bpy.types.Operator):
    """Select all objects in the specified collection for AutoLOD"""
    bl_idname = "autolod.export_collection_select"
    bl_label = "Select AutoLOD Collection Objects"

    @classmethod
    def poll(cls, context):
        return context.scene.autolod_collection_name != ''

    def execute(self, context):
        collection_name = context.scene.autolod_collection_name
        collection = bpy.data.collections.get(collection_name)

        if collection is None:
            self.report({'ERROR'}, f"Collection '{collection_name}' not found.")
            return {'CANCELLED'}

        bpy.ops.object.select_all(action='DESELECT')

        for obj in collection.objects:
            obj.select_set(True)
            context.view_layer.objects.active = obj

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		84

```
self.report({'INFO'}, f"All objects in '{collection_name}' are selected.")
return {'FINISHED'}
```

operators\AutoGenerator\apply_modifiers.py

```
import bpy

class AUTOLOD_OT_apply_modifiers(bpy.types.Operator):
    """Apply all modifiers for objects in the specified collection"""
    bl_idname = "autolod.apply_modifiers"
    bl_label = "Apply Modifiers in Collection"

    @classmethod
    def poll(cls, context):
        collection_name = context.scene.autolod_collection_name
        collection = bpy.data.collections.get(collection_name)
        return collection is not None and len(collection.objects) > 0

    def execute(self, context):
        collection_name = context.scene.autolod_collection_name
        collection = bpy.data.collections.get(collection_name)

        if collection is None:
            self.report({'ERROR'}, f"Collection '{collection_name}' not found.")
            return {'CANCELLED'}

        for obj in collection.objects:
            if obj.type == 'MESH':
                for modifier in obj.modifiers:
                    bpy.context.view_layer.objects.active = obj
                    bpy.ops.object.modifier_apply(modifier=modifier.name)

        self.report({'INFO'}, f"All modifiers in '{collection_name}' have been applied.")
        return {'FINISHED'}
```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		85

Код ноди Normal Map - node_shader_normal_map.c

```

/* SPDX-FileCopyrightText: 2005 Blender Authors
 *
 * SPDX-License-Identifier: GPL-2.0-or-later */

#include "node_shader_util.hh"
#include "node_util.hh"

#include "BKE_context.hh"
#include "BKE_node_runtime.hh"

#include "DEG_depsgraph_query.hh"

#include "RNA_access.hh"

#include "UI_interface.hh"
#include "UI_resources.hh"

namespace blender::nodes::node_shader_normal_map_cc {

static void node_declare(NodeDeclarationBuilder &b)
{
    b.add_input<decl::Float>("Strength")
        .default_value(1.0f)
        .min(0.0f)
        .max(10.0f)
        .description("Strength of the normal mapping effect");
    b.add_input<decl::Color>("Color")
        .default_value({0.5f, 0.5f, 1.0f, 1.0f})
        .description("Color that encodes the normal map in the specified space");
    b.add_output<decl::Vector>("Normal");
}

static void node_shader_but_normal_map(uiLayout *layout, bContext *C, PointerRNA
*ptr)
{
    uiItemR(layout, ptr, "space", UI_ITEM_R_SPLIT_EMPTY_NAME, "", ICON_NONE);

    if (RNA_enum_get(ptr, "space") == SHD_SPACE_TANGENT) {
        PointerRNA obptr = CTX_data_pointer_get(C, "active_object");

        if (obptr.data && RNA_enum_get(&obptr, "type") == OB_MESH) {
            PointerRNA eval_obptr;

            Depsgraph *depsgraph = CTX_data_ensure_evaluated_depsgraph(C);
            DEG_get_evaluated_rna_pointer(depsgraph, &obptr, &eval_obptr);
            PointerRNA dataptr = RNA_pointer_get(&eval_obptr, "data");
        }
    }
}

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

        uiItemPointerR(layout, ptr, "uv_map", &dataptr, "uv_layers", "",
ICON_GROUP_UVS);
    }
    else {
        uiItemR(layout, ptr, "uv_map", UI_ITEM_R_SPLIT_EMPTY_NAME, "", ICON_NONE);
    }
}
}

static void node_shader_init_normal_map(bNodeTree * /*ntree*/, bNode *node)
{
    NodeShaderNormalMap *attr = MEM_cnew<NodeShaderNormalMap>("NodeShaderNormalMap");
    node->storage = attr;
}

static int gpu_shader_normal_map(GPUMaterial *mat,
                                bNode *node,
                                bNodeExecData * /*execdata*/,
                                GPUNodeStack *in,
                                GPUNodeStack *out)
{
    NodeShaderNormalMap *nm = static_cast<NodeShaderNormalMap *>(node->storage);

    GPUNodeLink *strength;
    if (in[0].link) {
        strength = in[0].link;
    }
    else if (node->runtime->original) {
        bNodeSocket *socket = static_cast<bNodeSocket *>(
            BLI_findlink(&node->runtime->original->inputs, 0));
        bNodeSocketValueFloat *socket_data = static_cast<bNodeSocketValueFloat *>(
            socket->default_value);
        strength = GPU_uniform(&socket_data->value);
    }
    else {
        strength = GPU_constant(in[0].vec);
    }

    GPUNodeLink *newnormal;
    if (in[1].link) {
        newnormal = in[1].link;
    }
    else if (node->runtime->original) {
        bNodeSocket *socket = static_cast<bNodeSocket *>(
            BLI_findlink(&node->runtime->original->inputs, 1));
        bNodeSocketValueRGBA *socket_data = static_cast<bNodeSocketValueRGBA *>(socket-
>default_value);
        newnormal = GPU_uniform(socket_data->value);
    }
    else {
        newnormal = GPU_constant(in[1].vec);
    }
}

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		87

```

}

const char *color_to_normal_fnc_name = "color_to_normal_new_shading";
if (ELEM(nm->space, SHD_SPACE_BLENDER_OBJECT, SHD_SPACE_BLENDER_WORLD)) {
    color_to_normal_fnc_name = "color_to_blender_normal_new_shading";
}

GPU_link(mat, color_to_normal_fnc_name, newnormal, &newnormal);
switch (nm->space) {
    case SHD_SPACE_TANGENT:
        GPU_material_flag_set(mat, GPU_MATFLAG_OBJECT_INFO);
        /* We return directly from the node_normal_map as strength
         * has already been applied for the tangent case */
        GPU_link(mat,
            "node_normal_map",
            GPU_attribute(mat, CD_TANGENT, nm->uv_map),
            strength,
            newnormal,
            &out[0].link);
        return true;
    case SHD_SPACE_OBJECT:
    case SHD_SPACE_BLENDER_OBJECT:
        GPU_link(mat, "normal_transform_object_to_world", newnormal, &newnormal);
        break;
    case SHD_SPACE_WORLD:
    case SHD_SPACE_BLENDER_WORLD:
        /* Nothing to do. */
        break;
}

/* Final step - mix and apply strength for all other than tangent space. */
GPU_link(mat, "node_normal_map_mix", strength, newnormal, &out[0].link);

return true;
}

NODE_SHADER_MATERIALX_BEGIN
#ifdef WITH_MATERIALX
{
    NodeShaderNormalMap *normal_map_node = static_cast<NodeShaderNormalMap *>(node_
->storage);
    NodeItem color = get_input_value("Color", NodeItem::Type::Vector3);
    NodeItem strength = get_input_value("Strength", NodeItem::Type::Float);

    std::string space;
    switch (normal_map_node->space) {
        case SHD_SPACE_TANGENT:
            space = "tangent";
            break;
        case SHD_SPACE_OBJECT:
        case SHD_SPACE_BLENDER_OBJECT:

```

		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        space = "object";
        break;
    case SHD_SPACE_WORLD:
    case SHD_SPACE_BLENDER_WORLD:
        /* World isn't supported, tangent space will be used */
        space = "tangent";
        break;
    default:
        BLI_assert_unreachable();
}

return create_node("normalmap",
                  NodeItem::Type::Vector3,
                  {"in", color}, {"scale", strength}, {"space", val(space)}));
}
#endif
NODE_SHADER_MATERIALX_END

} // namespace blender::nodes::node_shader_normal_map_cc

/* node type definition */
void register_node_type_sh_normal_map()
{
    namespace file_ns = blender::nodes::node_shader_normal_map_cc;

    static blender::bke::bNodeType ntype;

    sh_node_type_base(&ntype, SH_NODE_NORMAL_MAP, "Normal Map",
NODE_CLASS_OP_VECTOR);
    ntype.declare = file_ns::node_declare;
    ntype.draw_buttons = file_ns::node_shader_but_normal_map;
    blender::bke::node_type_size_preset(&ntype,
blender::bke::eNodeSizePreset::Middle);
    ntype.initfunc = file_ns::node_shader_init_normal_map;
    blender::bke::node_type_storage(
        &ntype, "NodeShaderNormalMap", node_free_standard_storage,
node_copy_standard_storage);
    ntype.gpu_fn = file_ns::gpu_shader_normal_map;
    ntype.materialx_fn = file_ns::node_shader_materialx;

    blender::bke::nodeRegisterType(&ntype);
}

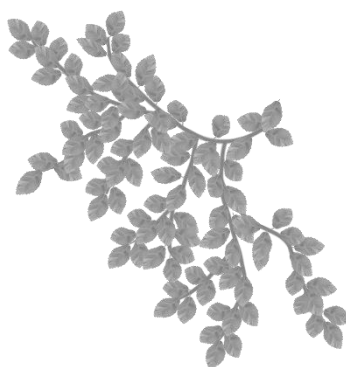
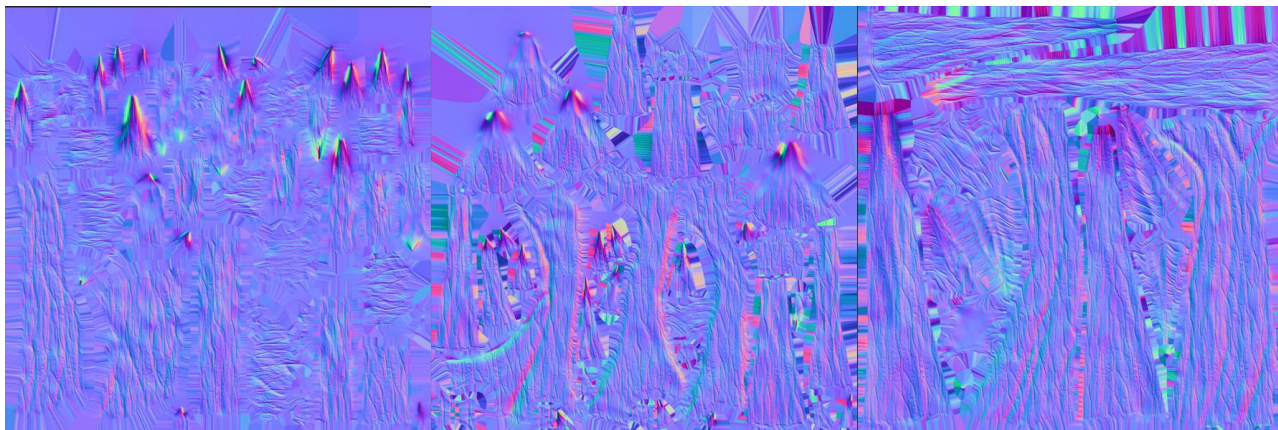
```

Моделі: AW_Broadleaf_3, AW_Broadleaf_2, AW_Broadleaf_1



Текстури для моделей: AW_Broadleaf_3, AW_Broadleaf_2, AW_Broadleaf_1



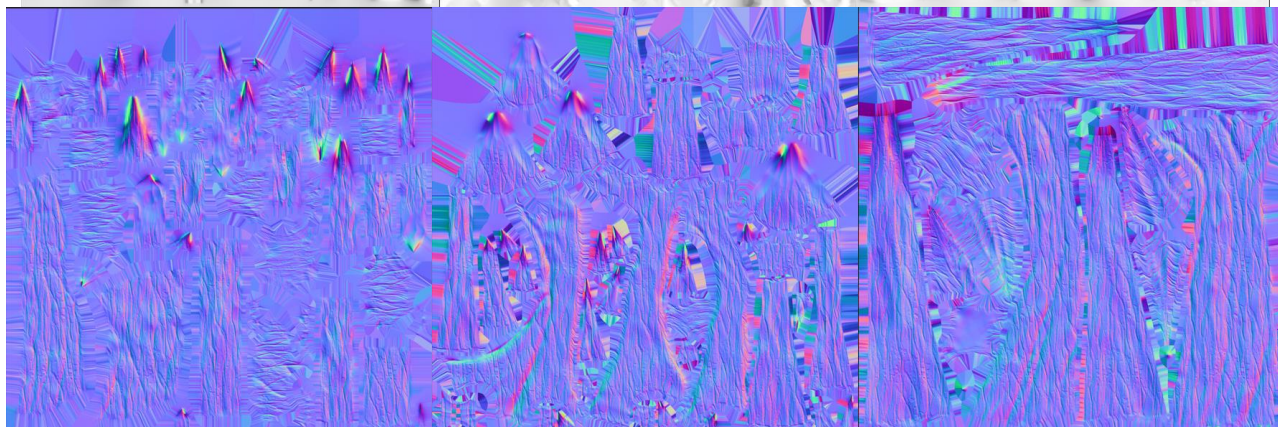
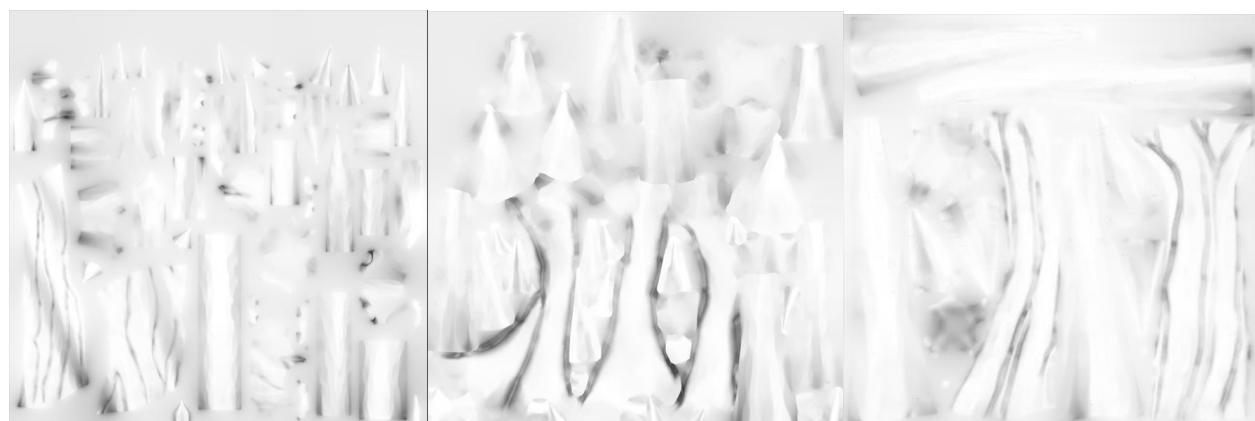
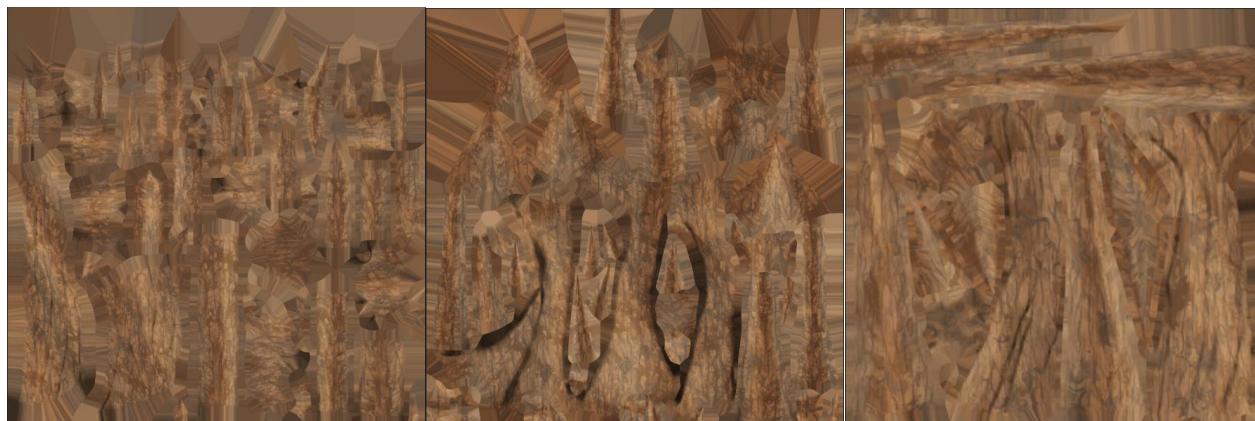


Моделі: AW_Dead_Tree_3, AW_Dead_Tree_2, AW_Dead_Tree_1



		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		91

Текстури для моделей: AW_Dead_Tree_3, AW_Dead_Tree_2, AW_Dead_Tree_1

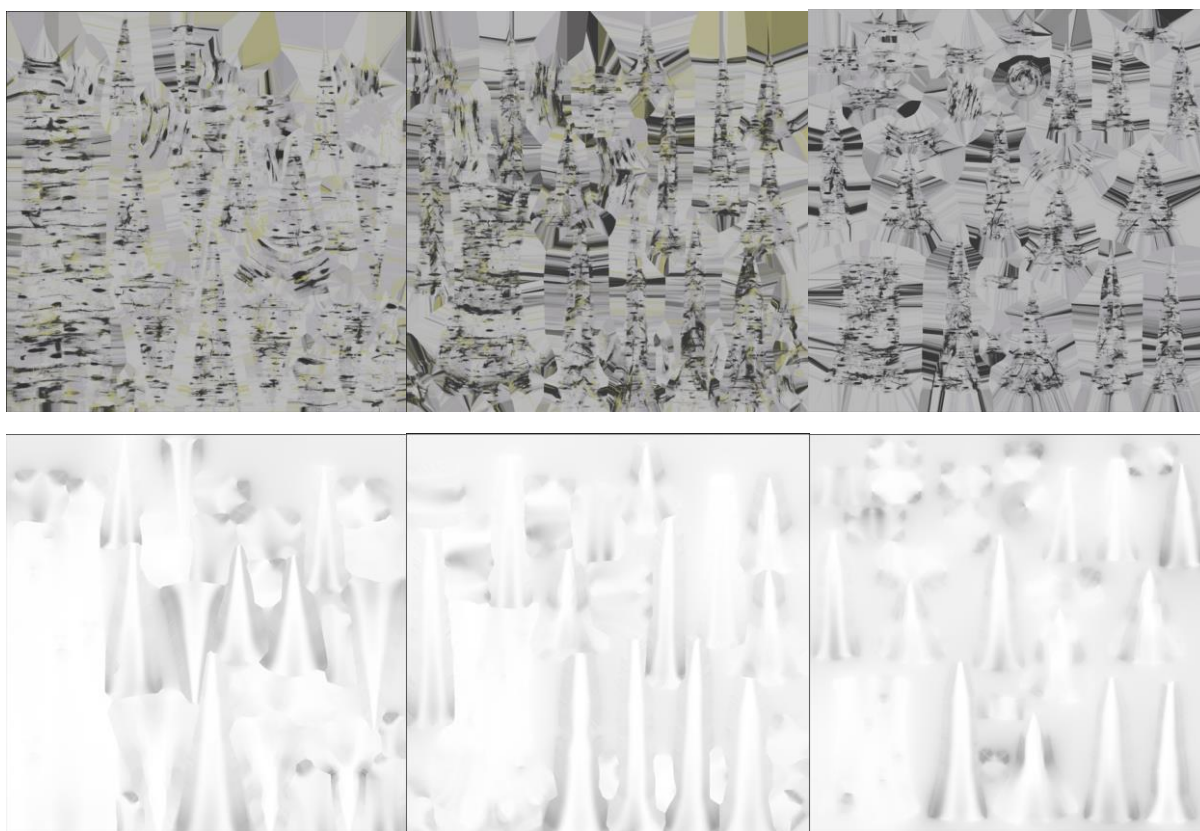


		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		92

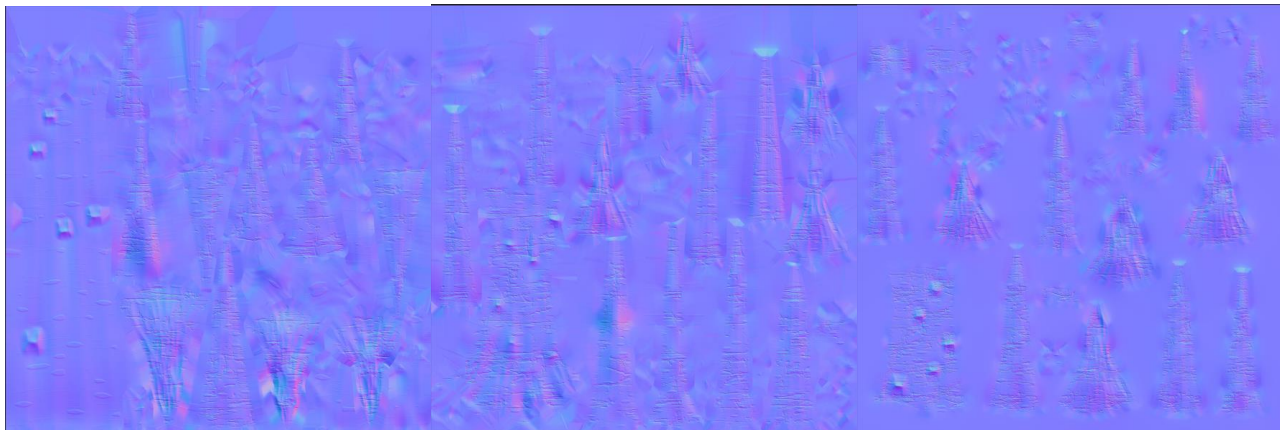
Моделі: AW_Birch_3, AW_Birch_2, AW_Birch_1



Текстури для моделей: AW_Birch_3, AW_Birch_2, AW_Birch_1



		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		93

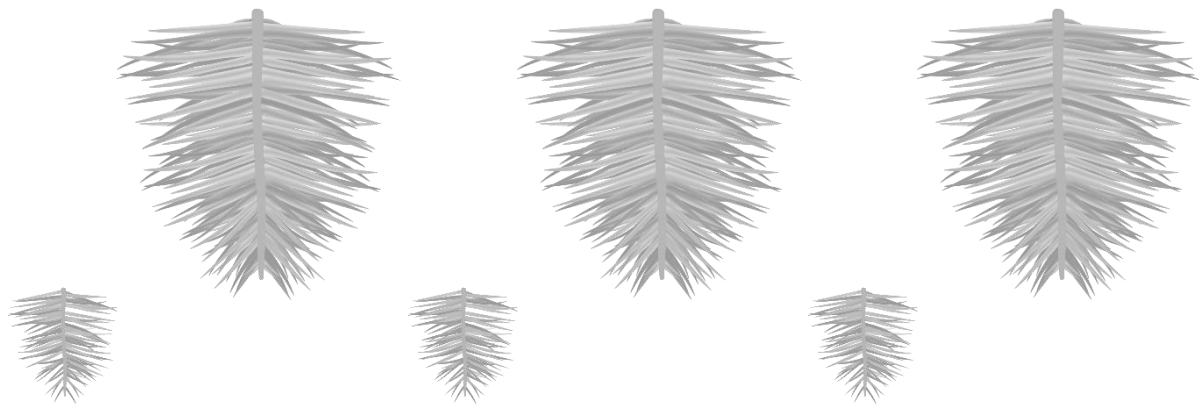
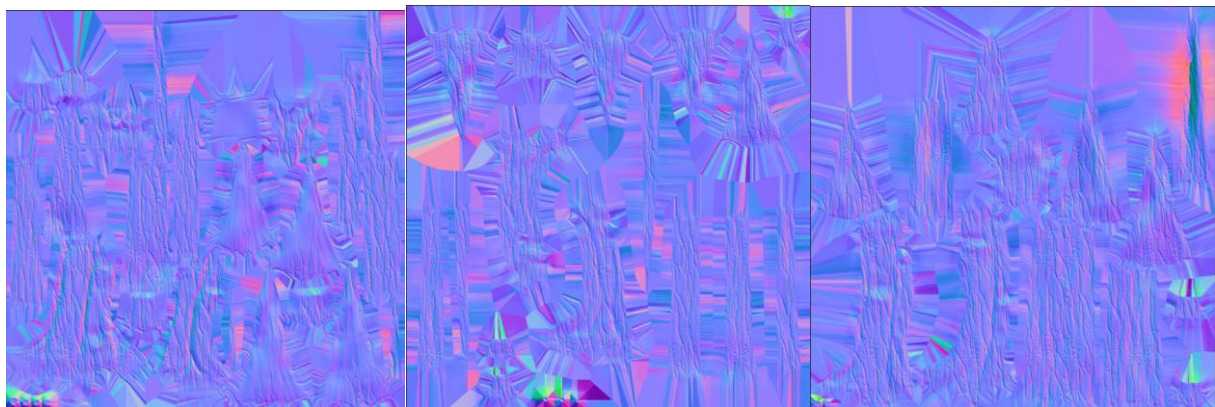
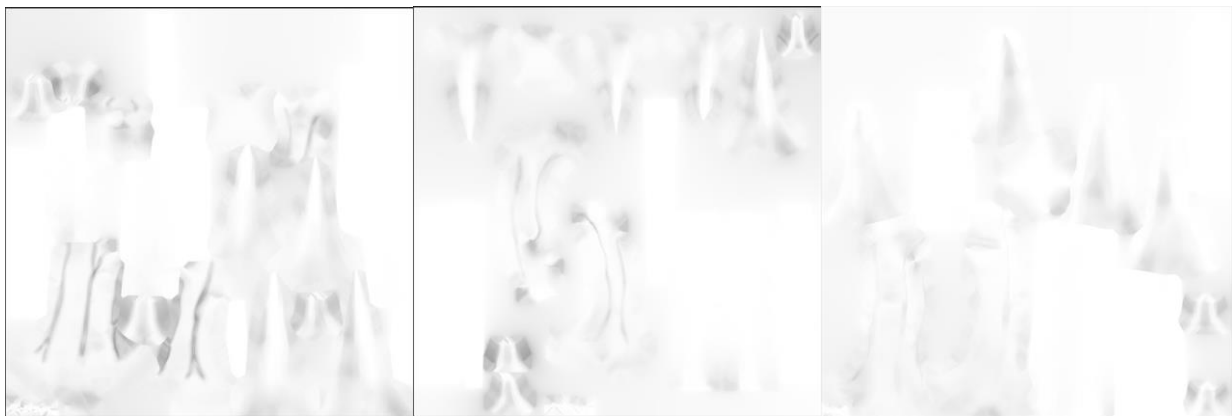


Моделі: AW_Fir_3, AW_Fir_2, AW_Fir_1

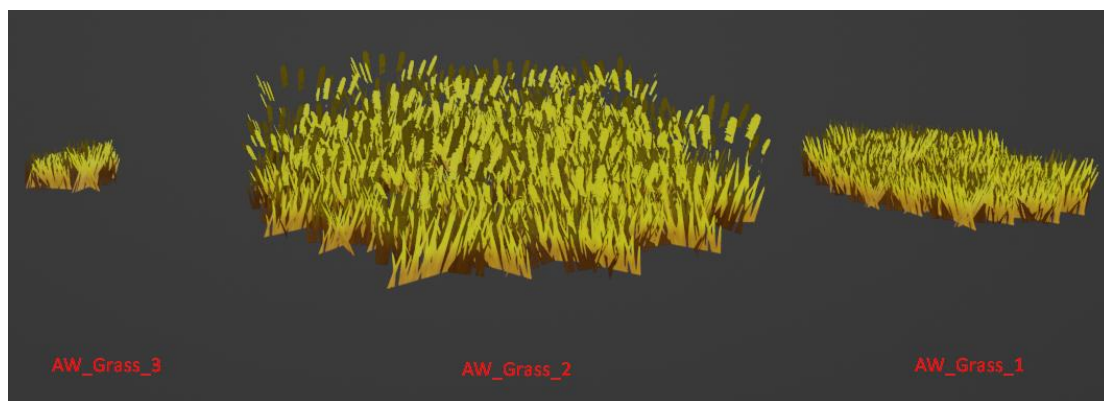


		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				94
Змн.	Арк.	№ докум.	Підпис	Дата		

Текстури для моделей: AW_Fir_3, AW_Fir_2, AW_Fir_1



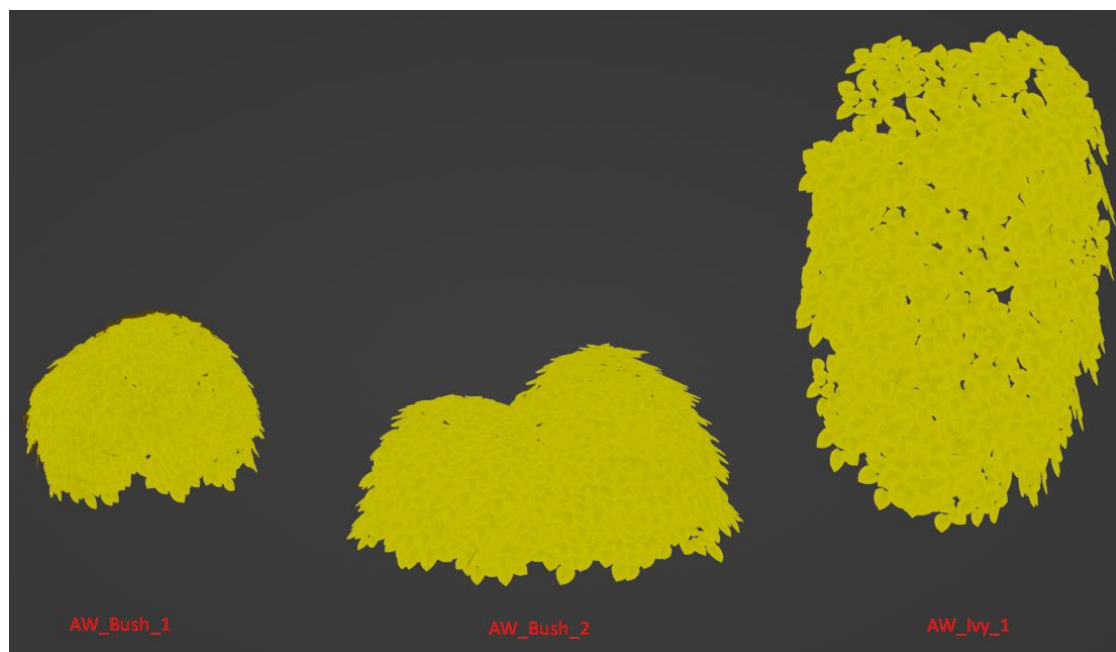
Моделі: AW_Grass_3, AW_Grass_2, AW_Grass_1



Текстури для моделей: AW_Grass_3, AW_Grass_2, AW_Grass_1



Моделі: AW_Bush_1, AW_Bush_2, AW_Ivy_1



		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		96

Текстури для моделей: AW_Bush_1, AW_Bush_2, AW_Ivy_1



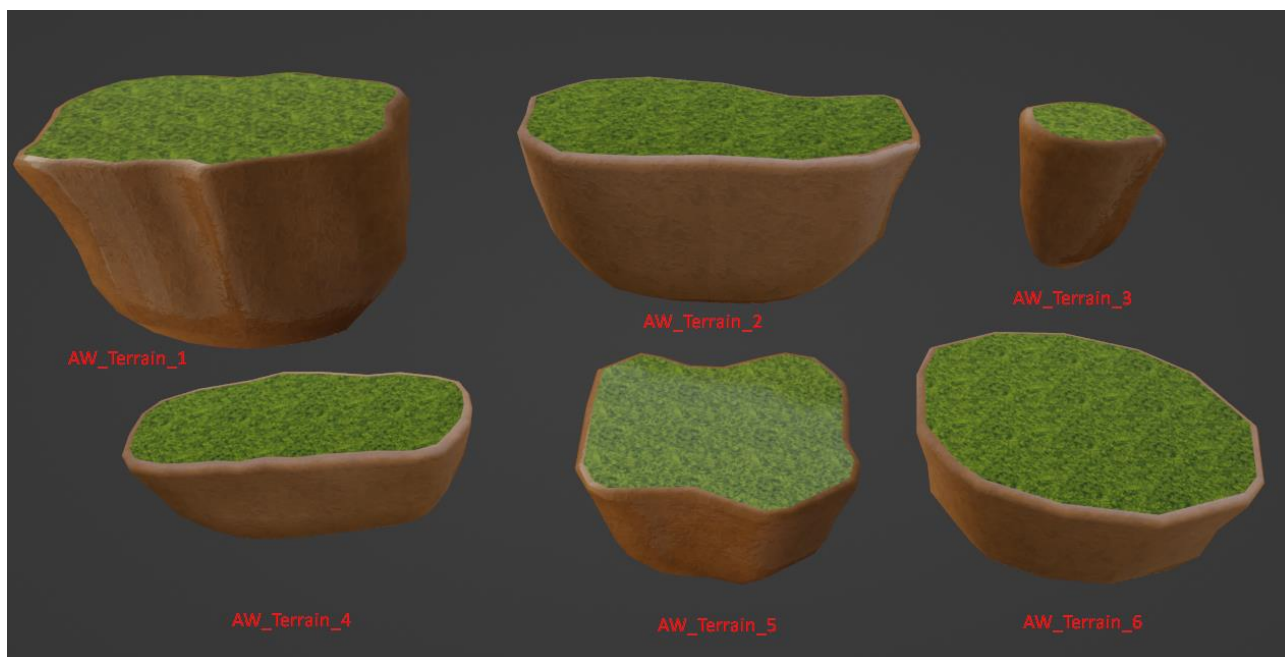
Моделі: AW_Fence_2, AW_Fence_1



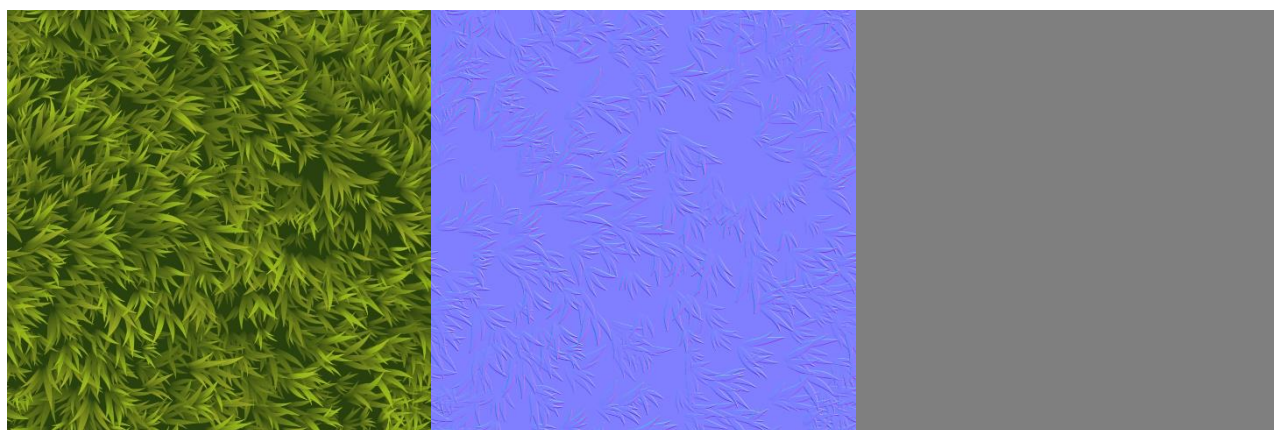
Текстури для моделей: AW_Fence_2, AW_Fence_1



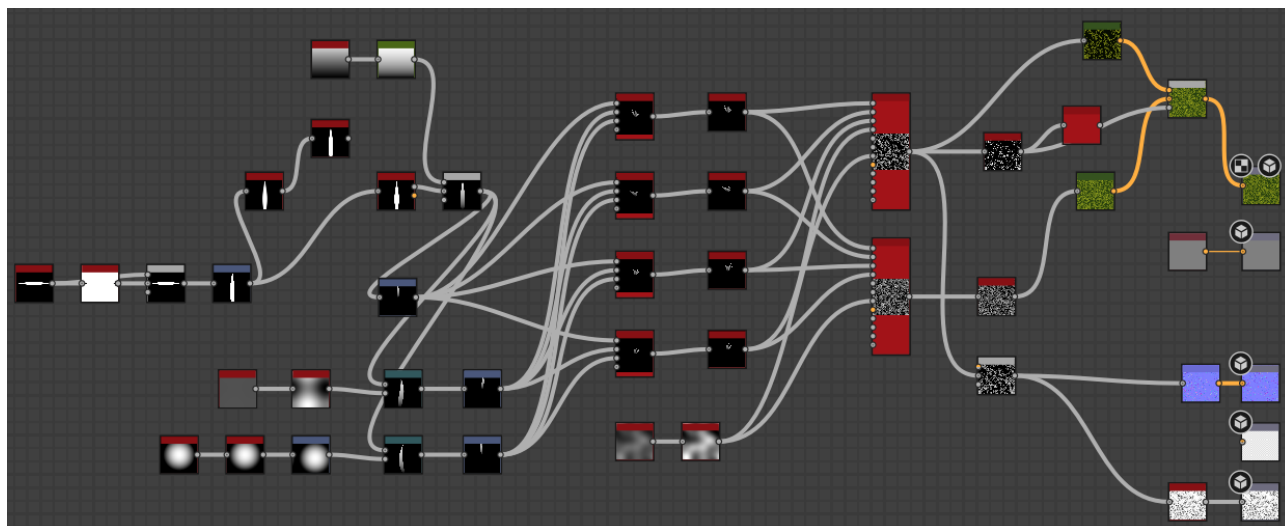
Моделі: AW_Terrain_1, AW_Terrain_2, AW_Terrain_3, AW_Terrain_4,
AW_Terrain_5, AW_Terrain_6



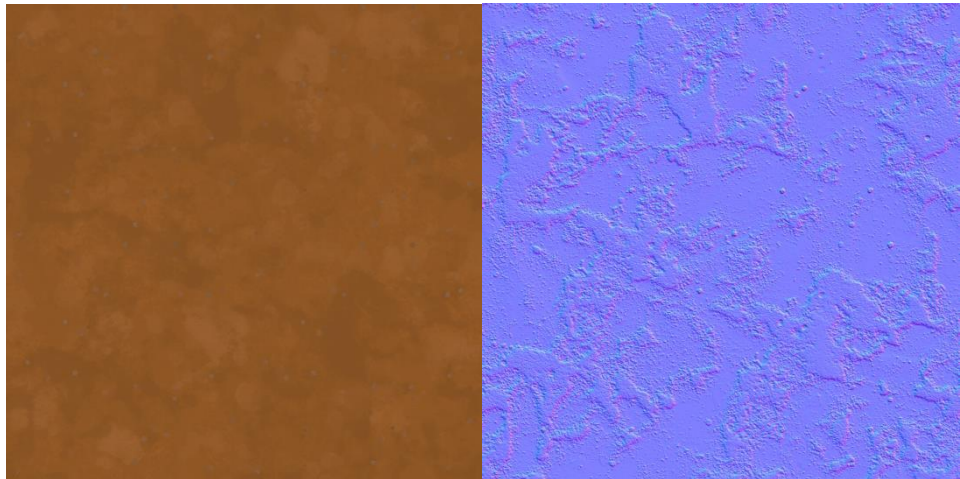
Текстура трави: AW_Grass_1



Нодова система в програмі Substance 3D Designer для текстури: AW_Grass_1



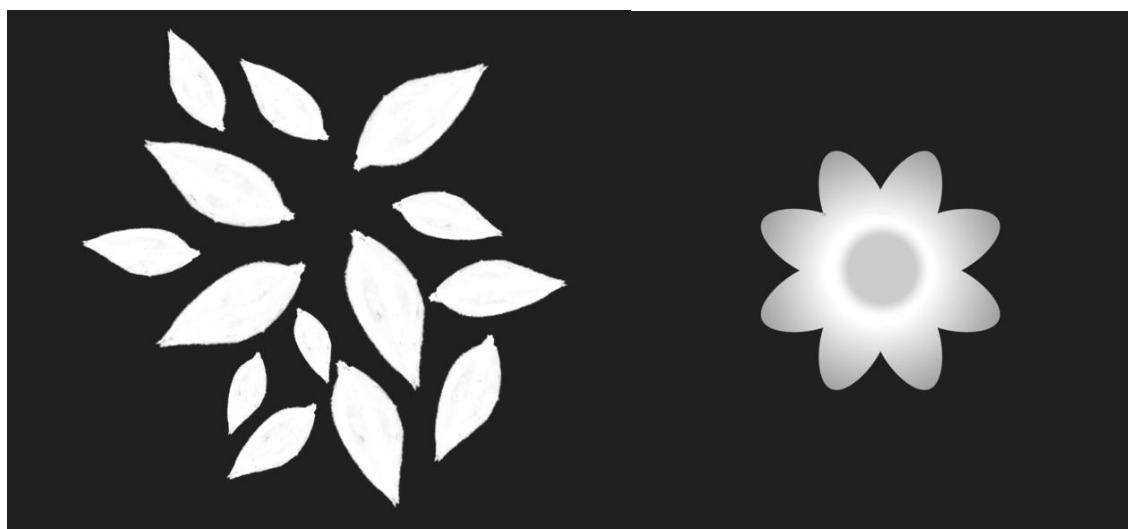
Текстура бруду: AW_Terrain_Dirt



Моделі: PP_Bush_3



Текстури для моделей: PP_Bush_3



		М. О. ШЕВЧЕНКО			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		Г. В. МАРЧУК				
Змн.	Арк.	№ докум.	Підпис	Дата		99