Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ СТВОРЕННЯ 3D-МОДЕЛЕЙ ДЛЯ ІГРОВИХ РУШІЇВ»

Виконав: студент 4 курсу, групи 2ІСТ-21б спеціальності 126 – «Інформаційні системи та технології»
_____ Микола РИБАК
Керівник: к.т.н., доц. каф. САІТ
_____ Сергій ЖУКОВ
«05» 06 2025 р.
Рецензент: к.т.н., доц. каф. КН
_____ Володимир ОЗЕРАНСЬКИЙ
«08» 06 2025 р.

Допущено до захисту

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«06» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
д.т.н., проф. Віталій МОКІН
"18" 05 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Рибаку Миколі Миколайовичу

1. Тема роботи: «Інформаційна технологія створення 3D-моделей для ігрових рушіїв»
керівник роботи: Жуков С. О., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «20» 03 2025 року №97
2. Термін подання студентом роботи 31.05.25 р.
3. Вихідні дані до роботи:
1) Концепт-арти для розроблюваної 3D-моделі;
2) Референсні матеріали для розроблюваної 3D-моделі.
4. Зміст текстової частини:
1) Аналіз предметної області;
2) Вибір технології створення 3D-моделей та IT-інфрастркутури;
3) Проєктування та реалізація інформаційної технології створення 3D-моделей для ігрових рушіїв.
5. Перелік ілюстративного матеріалу:
1) UML-Діаграми;
2) Меш;
3) UV-Розгортка;
4) Текстури.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 28.03.25 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	01.04	15.04	
2	Вибір технології створення 3D-моделей	15.04	30.04	
3	Вибір оптимальної IT-Інфраструктури	01.05	10.05	
4	Проєктування та реалізація інформаційної технології створення 3D-моделей для ігрових рушіїв	10.05	20.05	
5	Оформлення матеріалів до захисту БКР	20.05	30.05	

Студент _____ Микола РИБАК

Керівник роботи _____ Сергій ЖУКОВ

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 91 сторінки формату А4, на яких є 53 рисунка, список використаних джерел містить 29 найменувань.

В роботі розроблено інформаційну технологію створення 3D моделей для ігрових рушіїв.

В роботі наведено розроблення інформаційної технології раціональної оптимізації 3D-моделей, включаючи адаптивне моделювання, оптимізоване текстурування та прискорений цикл розробки. Розроблено проєктну архітектуру (UML-діаграми) та застосовано комплекс програмних інструментів для створення та інтеграції 3D-моделі. Розроблена методологія дає можливість значного прискорення процесу створення високоякісних та продуктивних ігрових асетів.

Ключові слова: інформаційна технологія, 3D-моделювання, оптимізація, ігрові рушії, ігрові асети, Unity, Blender 3D, Adobe Substance 3D Painter.

ABSTRACT

The bachelor's thesis consists of 91 pages of A4 format, including 53 figures. The list of references contains 29 sources.

An information technology for creating 3D models for game engines was developed.

The work presents the development of an information technology for the rational optimization of 3D models, including adaptive modelling, optimized texturing, and an accelerated development cycle. A project architecture (UML diagrams) has been developed, and a set of software tools has been applied for the creation and integration of 3D models. The developed methodology allows for a significant acceleration of the process of creating high-quality and performant game assets.

Keywords: information technology, 3D modelling, optimization, game engines, game assets, Unity, Blender 3D, Adobe Substance 3D Painter.

ЗМІСТ

ВСТУП

Актуальність теми. Стрімкий розвиток індустрії відеоігор у 21 столітті зумовив значне зростання попиту на високоякісний 3D контент. Тривимірні моделі є невід'ємною частиною сучасних ігор, формуючи візуальне сприйняття ігрового світу та забезпечуючи глибину занурення гравця. Від реалістичності та оптимізації цих моделей залежить не лише візуальна привабливість гри, але й її продуктивність на різних апаратних платформах.

Сучасні ігрові рушії, такі як Unity та Unreal Engine, надають розробникам потужні інструменти для створення складних та захоплюючих ігрових світів. Проте, їх застосування не обмежується лише ігровою індустрією. Ігрові рушії знаходять все більше застосувань у таких сферах, як доповнена та віртуальна реальність (AR/VR), інтерактивне навчання, архітектурна візуалізація, медичні симуляції, кінематограф та інші, де потрібне створення та маніпулювання 3D об'єктами в реальному часі.

Ефективне використання ігрових рушіїв вимагає глибокого розуміння технологій створення 3D моделей, включаючи моделювання, текстурування, оптимізацію та інтеграцію в інтерактивне середовище.

Актуальність цієї дипломної роботи полягає в необхідності дослідження та систематизації сучасних методів створення 3D моделей для ігрових рушіїв. Розуміння оптимальних підходів до моделювання, текстурування та оптимізації є критично важливим для розробників, оскільки це безпосередньо впливає на якість, продуктивність та вартість розробки кінцевого продукту в різних сферах застосування.

Мета дослідження і задачі дослідження. Метою дослідження є підвищення ефективності створення 3D-моделей для ігрових рушіїв шляхом розроблення оптимізованої інформаційної технології, що ґрунтується на принципах раціональної оптимізації та гнучкого виробничого пайплайну для прискорення циклу розробки ігрових асетів.

Для досягнення поставленої мети необхідно розв’язавти такі задачі:

- Проаналізувати сучасні ігрові рушії та їхні вимоги до 3D моделей.

- Розглянути основні етапи процесу створення 3D моделей.
- Дослідити методи оптимізації 3D моделей для використання в ігрових рушіях.
- Розробити практичний приклад створення та оптимізації 3D моделі для конкретного ігрового рушія.
- Оцінити вплив оптимізації на продуктивність ігрового середовища.

Об'єктом дослідження є процес розроблення інформаційної технології оптимізації та прискорення циклу розробки 3D моделей для ігрових рушіїв.

Предметом дослідження є методи й програмні засоби створення інформаційної технололгії оптимізації та прискорення циклу розробки 3D моделей для ігрових рушіїв.

Публікації: За результатами виконання даної роботи опубліковано тези доповідей на тему: «Порівняльний аналіз методів 3D-моделювання: полігональне та CAD» на Всеукраїнській науково-технічній конференцій факультету інтелектуальних інформаційних технологій та автоматизації (Вінниця, 2025 р.) [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Представлення 3D-об’єктів

Перш ніж заглиблюватися в можливості сучасних ігрових рушіїв, важливо зрозуміти базові принципи, на яких ґрунтується відтворення 3D-графіки на екрані комп'ютера. Цей розділ надасть огляд ключових етапів та концепцій, включаючи представлення 3D-об'єктів, обчислення їх координат та процес перетворення 3D-сцени у 2D-зображення.

У комп'ютерній графіці 3D-об'єкти представлені у вигляді сітки, що складається з багатокутників, бажано – трикутників та чотирикутників. Цей метод називається полігональним моделюванням. Кожен трикутник визначається трьома вершинами, кожна з яких має свої координати у тривимірному просторі (X,Y,Z) [2].

Можливий вигляд сітки та наочний приклад вершини з її координатами зображено на рисунку 1.1:

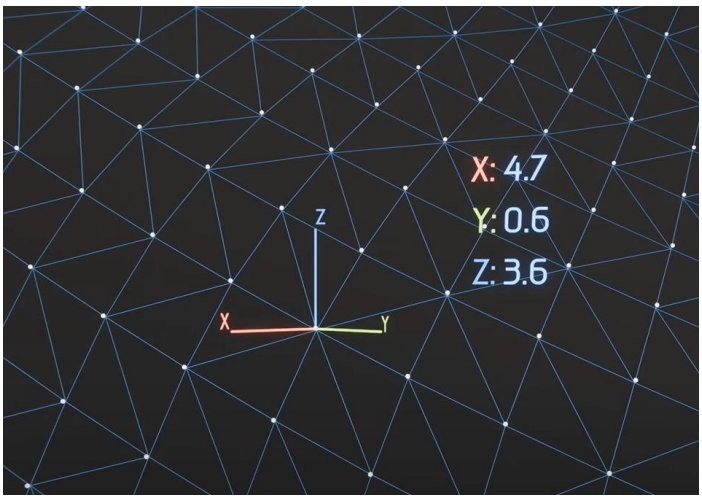

Рисунок 1.1 – Сітка 3D-об’єкту в сцені та координати вершини

Крім координат, вершини можуть містити додаткову інформацію, таку як:

- Нормаль: Вектор, перпендикулярний до поверхні трикутника, що використовується для розрахунку освітлення.
- Колір: Колір вершини, який може бути інтерпольований для отримання плавних кольорових переходів по поверхні трикутника.

- Текстурні координати (UV-координати): Визначають, яка частина текстури накладається на дану вершину.

1.2 Графічний конвеєр (Graphics Pipeline)

Процес перетворення 3D-моделі у 2D-зображення на екрані відбувається у кілька етапів, які утворюють так званий графічний конвеєр. Сучасні графічні процесори (GPU) спеціалізуються на швидкому виконанні цих операцій. Основні етапи графічного конвеєра:

- Вершинний шейдер (Vertex Shader). Відбувається обробка кожної вершини моделі. Виконуються такі операції: трансформація координат (обертання, переміщення, масштабування); розрахунок освітлення на рівні вершин; передача даних на наступний етап конвеєра.
- Теселяція (Tessellation). Розбиває складні поверхні на дрібніші трикутники для підвищення деталізації.
- Геометричний шейдер (Geometry Shader). Створює або видаляє геометрію, для генерації частинок або зміни форми об’єктів.
- Растеризація (Rasterization). Перетворює векторне представлення трикутників у пікселі (фрагменти) на екрані. Для кожного пікселя визначається, чи потрапляє він всередину даного трикутника.
- Фрагментний шейдер (Fragment Shader). Обробляє кожен піксель (фрагмент), визначаючи його остаточний колір. На цьому етапі відбуваються застосування текстур, розрахунок освітлення з урахуванням матеріалів та джерел світла, а також виконання інших візуальних ефектів (наприклад, туман, пост-обробка).
- Злиття (Blending). Комбінує кольори пікселів з різних об'єктів у сцені, враховуючи їх глибину (Z-буфер) та прозорість.
- Вивід зображення на екран: Остаточне 2D-зображення, отримане після проходження графічного конвеєра, зберігається у фреймбуфері - спеціальній області пам'яті відеокарти. Далі, вміст фреймбуфера передається на монітор для відображення [3].

Це не всі пункти графічного конвеєру, вони можуть доповнюватись в залежності від поставленої задачі та проєкту.

Схема основних пунктів графічного конвеєру ігрових моделей зображено на рисунку 1.2 [4].

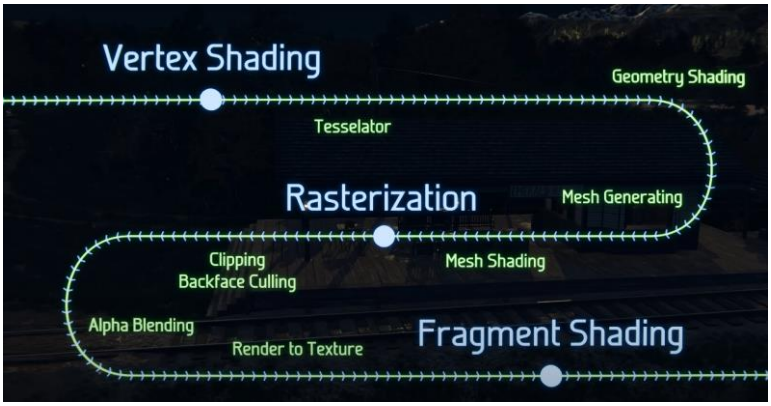

Рисунок 1.2 – Основні пункти графічного конвеєру

Графічний пайплайн зображено на рисунку 1.3, 1.4 та 1.5.

Рисунок 1.3 – Графічний пайплайн (а)

Рисунок 1.4 - Графічний пайплайн (б)

Рисунок 1.5 – Графічний пайплайн (в)

1.3 Роль відеокарти (GPU)

Відеокарта (Graphics Processing Unit) - це спеціалізований процесор, призначений для прискорення операцій, пов'язаних з обробкою та виводом графіки. Сучасні GPU є складними електронними пристроями, що складаються з мільярдів транзисторів, організованих у велику кількість обчислювальних ядер. Ця масивна паралельна архітектура робить GPU ідеально придатними для виконання численних одночасних обчислень, необхідних для рендерингу 3D-графіки. На відміну від центральних процесорів (CPU), які призначені для обробки широкого спектру завдань послідовно, GPU спеціалізуються на паралельній обробці даних, що дозволяє їм значно прискорювати графічні операції.

Сучасні GPU мають сотні або навіть тисячі обчислювальних ядер, що дозволяє їм паралельно виконувати безліч операцій графічного конвеєра, значно підвищуючи продуктивність рендерингу. Крім того, GPU оснащені спеціалізованою пам'яттю з високою пропускною здатністю, що забезпечує швидкий доступ до даних, необхідних для рендерингу. Це особливо важливо для обробки великих текстур та буферів кадрів, які використовуються в сучасних іграх та графічних застосунках.

З розвитком технологій GPU стали невід'ємною частиною не тільки комп'ютерних ігор, але й багатьох інших областей, включаючи наукові дослідження, штучний інтелект, машинне навчання, обробку зображень та відео,

а також криптовалюти. Їх здатність ефективно обробляти великі обсяги даних паралельно робить їх незамінними для будь-яких обчислювальних завдань, де потрібна висока продуктивність та швидкість обробки.

Однією з ключових технологій, що використовується в сучасних GPU, є CUDA (Compute Unified Device Architecture) - це паралельна обчислювальна платформа та модель програмування, розроблена компанією NVIDIA. CUDA дозволяє розробникам використовувати GPU для широкого спектру обчислювальних завдань, включаючи рендеринг 3D-графіки, наукові симуляції та штучний інтелект. В контексті 3D-рендерингу, CUDA дозволяє розробникам використовувати CUDA-ядра (обчислювальні ядра GPU) для прискорення різноманітних операцій, включаючи вершинні та фрагментні шейдери, а також трасування променів.

На рисунку 1.6 зображена схематична структура відеокарти RTX 3090 (чип GA 102) [5].

Рисунок 1.6 - Схематична структура відеокарти RTX 3090

Ray tracing (трасування променів) - це метод рендерингу, який моделює шлях світла у сцені для створення реалістичних зображень. На відміну від традиційної растеризації, яка швидко обчислює колір пікселів на основі положення об'єктів та джерел світла, ray tracing простежує шлях кожного

променя світла від камери до джерела світла, враховуючи його взаємодію з об'єктами на шляху. Це дозволяє більш точно відтворювати такі ефекти, як:

- Відбиття: Ray tracing дозволяє точно моделювати відбиття світла від дзеркальних поверхонь, враховуючи форму та матеріали об'єктів у сцені.
- Заломлення: Ray tracing може імітувати заломлення світла при проходженні через прозорі об'єкти, такі як скло або вода, створюючи реалістиві оптичні ефекти.
- Тіні: Ray tracing забезпечує м'які та реалістичні тіні з урахуванням площі джерела світла та відстані до об'єкта.
- Глобальне освітлення: Ray tracing може моделювати непряме освітлення, коли світло відбивається від поверхонь та освітлює інші об'єкти у сцені, створюючи більш реалістичну атмосферу.

Для прискорення обчислень ray tracing в реальному часі використовуються різні підходи, включаючи апаратне прискорення. Компанія NVIDIA розробила технологію RTX, яка включає в себе спеціалізовані апаратні компоненти, що називаються RT-ядрами. RT-ядра - це обчислювальні блоки, призначені для виконання операцій ray tracing, таких як перетин променя та трикутника, а також прискорення обчислень за допомогою оптимізації процесу трасування променів шляхом використання спеціальних алгоритмів та структур даних (наприклад, BVH - Bounding Volume Hierarchy) [7].

Наочно технологію RTX безпосередньо в грі зображено на рисунку 1.7 [8].

Рисунок 1.7 – Технологія RTX

Використання CUDA та RT-ядер дозволяє значно підвищити продуктивність рендерингу, забезпечуючи можливість відтворення складних 3D-сцен в реальному часі з високою якістю зображення. Важливо зазначити, що Unreal Engine підтримує різні підходи до трасування променів, включаючи як апаратне прискорення через RT-ядра на відеокартах NVIDIA RTX, так і програмні методи, що дозволяють використовувати трасування променів на ширшому спектрі обладнання, хоча й з меншою продуктивністю.

Для використання апаратного прискорення ray tracing потрібне спеціальне апаратне забезпечення. На даний момент, графічні карти NVIDIA GeForce RTX є основними, що підтримують апаратне прискорення ray tracing.

На рисунку 1.8 зображено вигляд відеокарти RTX 3090 (без системи охолодження), де детально можна розглянути чип, транзистори та блоки пам’яті [9].

Рисунок 1.7 – Відеокарта RTX 3090

1.4 Математичні основи та оптимізація.

Для роботи з 3D-графікою необхідні знання з лінійної алгебри та геометрії. Основні математичні концепції, що використовуються в 3D-рендерингу: вектори, які використовуються для представлення напрямків, положень та інших геометричних величин; матриці, які використовуються для виконання лінійних перетворень, таких як обертання, масштабування та переміщення об'єктів; трансформації, тобто математичні операції, що змінюють положення, орієнтацію або розмір об'єктів у просторі; і, нарешті, проекції - перетворення 3D-координат у 2D-координати для відображення на плоскому екрані (перспективна та ортографічна проекції) [10].

Крім того, важливим аспектом 3D-рендерингу є оптимізація. Створення 3D-сцен часто передбачає використання великої кількості даних, включаючи складні геометричні моделі та текстури високої роздільної здатності. Однак, відтворення таких сцен може бути обчислювально витратним, що призводить до низької продуктивності та поганого досвіду користувача. Тому розробники повинні знаходити баланс між візуальною якістю та продуктивністю.

Ось деякі ключові моменти, які слід враховувати при оптимізації 3D-графіки:

- Уникайте надмірно деталізованих моделей: Використання високополігональних моделей може значно збільшити навантаження на

GPU. Розробники повинні використовувати моделі з оптимальною кількістю полігонів, достатньою для досягнення бажаної візуальної якості, але не надмірною.

- Оптимізуйте текстури: Текстури високої роздільної здатності можуть споживати багато відеопам'яті та знижувати продуктивність. Розробники повинні використовувати текстури з роздільною здатністю, достатньою для забезпечення чіткості та деталізації, але не надмірною.
- Уникайте надмірного використання трасування променів (Ray Tracing): Трасування променів є потужним методом рендерингу, але він може бути дуже обчислювально витратним. Розробники повинні використовувати трасування променів лише для тих частин сцени, де це дійсно необхідно, і використовувати оптимізовані алгоритми для мінімізації навантаження на GPU.
- Використовуйте рівні деталізації (Level of Detail - LOD): LOD - це техніка, яка передбачає використання різних версій моделі з різною кількістю полігонів, залежно від відстані до камери. Це дозволяє зменшити навантаження на GPU при відображенні віддалених об'єктів.

Основним способом вимірювання продуктивності 3D-графіки є кількість кадрів в секунду (FPS - Frames Per Second). FPS показує, скільки зображень (кадрів) відображає дисплей за одну секунду. Чим вище FPS, тим більш плавною та приємною є гра або інший графічний застосунок. Зазвичай, для комфортної гри потрібно не менше 30 FPS, а для найкращого досвіду - 60 FPS або більше.

Однак, досягнення високого FPS може бути складним завданням, особливо у вимогливих іграх з високою графічною якістю. Для підвищення продуктивності використовуються різні технології, такі як:

- Суперсемплінг (Supersampling): Це метод, при якому зображення рендериться у вищій роздільній здатності, а потім зменшується до вихідної. Це дозволяє отримати більш чітке та деталізоване зображення, але знижує FPS.

- Згладжування (Anti-Aliasing): Це метод зменшення ефекту "сходинок" на краях об'єктів. Існують різні алгоритми згладжування, такі як MSAA, FXAA, TAA, кожен з яких має свій вплив на продуктивність.
- Масштабування з використанням ШІ (DLSS, FSR, XeSS): Це технології, які використовують штучний інтелект для збільшення роздільної здатності зображення з нижчої до вищої. Наприклад, DLSS (Deep Learning Super Sampling) від NVIDIA використовує нейронні мережі для реконструкції зображення з меншої кількості пікселів, що дозволяє значно підвищити FPS при мінімальній втраті якості. Аналогічні технології, такі як FSR (FidelityFX Super Resolution) від AMD та XeSS (Xe Super Sampling) від Intel, також пропонують масштабування з підвищенням продуктивності [11].

Розуміння цих концепцій та методів оптимізації є важливим для розробників 3D-графіки, оскільки це дозволяє їм створювати ефективні та продуктивні застосунки, які забезпечують високу якість зображення та комфортний досвід користувача.

1.5 Висновки

Підсумовуючи, 3D-рендеринг є складним, але надзвичайно важливим процесом у сучасній комп'ютерній графіці. Він охоплює широкий спектр концепцій та етапів, від представлення 3D-об'єктів у вигляді полігональних сіток до використання можливостей GPU для прискорення обчислень та трасування променів. Розробники повинні мати глибоке розуміння математичних основ, принципів роботи графічного конвеєра та методів оптимізації для створення візуально привабливих та продуктивних 3D-застосунків. Зі швидким розвитком технологій, таких як трасування променів у реальному часі та масштабування на основі ШІ, майбутнє 3D-рендерингу обіцяє ще більшу реалістичність, інтерактивність та ефективність.

2 ВИБІР ТЕХНОЛОГІЇ СТВОРЕННЯ 3D-МОДЕЛЕЙ ТА ІТ-ІНФРАСТРКУТУРИ

2.1 Вибір ігрових рушіїв та їхні вимоги до 3D моделей

Ігровий рушій - це програмний каркас, призначений для розробки відеоігор та інших інтерактивних застосунків. Він надає розробникам набір інструментів та компонентів для спрощення процесу створення гри, включаючи рендеринг графіки, управління фізикою, обробку звуку, анімацію, штучний інтелект та багато іншого.

Сучасна індустрія відеоігор значною мірою визначається можливостями та функціональністю ігрових рушіїв. Ці програмні комплекси надають розробникам широкий спектр інструментів для створення інтерактивних 3D-світів, управління ігровою логікою та рендерингу графіки. Серед розмаїття ігрових рушіїв, Unity та Unreal Engine займають провідні позиції, визначаючи стандарти розробки та пропонуючи потужні рішення для різних жанрів та платформ.

Unity - це крос-платформовий ігровий рушій, розроблений компанією Unity Technologies. Він був вперше випущений у 2005 році та здобув широку популярність завдяки своїй універсальності та доступності. Unity особливо популярний серед незалежних розробників та невеликих студій, але також використовується і у великих проектах. На цьому рушію були створенні такі відомі проєкти: Genshin Impact, Pokemon GO, Call of Duty: Mobile, Hollow Knight, Among Us. Головною перевагою Unity є крос-платформеність та підтримка написання скриптів на C# [12].

На рисунку 2.1 можемо розглянути інтерфейс Unity 6.

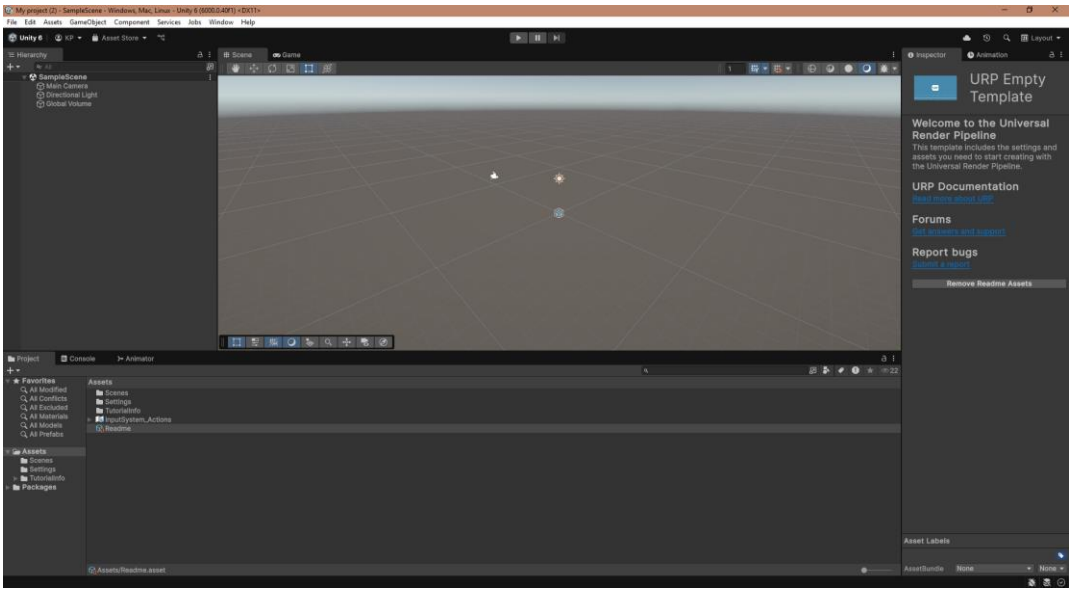

Рисунок 2.1 - Інтерфейс Unity 6

Unreal Engine - це потужний ігровий рушій, розроблений компанією Epic Games. Він був вперше представлений у 1998 році та відомий своєю передовою графікою, потужними інструментами та широким спектром можливостей. Unreal Engine часто використовується для створення AAA-ігор з високою графічною якістю, але також підходить для розробки ігор інших жанрів, AR/VR-додатків, кінофільмів та архітектурних візуалізацій. На цьому рушію були створенні такі відомі проєкти: Fortnite, Batman: Arkham Series, BioShock, Borderlands, Gears of War. Головною перевагою Unreal Engine є можливість візуального програмування (Blueprint), система Nanite (авто LOD) та гнучкі налаштування оптимізації, що є актуальним для великих проєктів [13].

Інтерфейс Unreal Engine 5 зображено на рисунку 2.2.

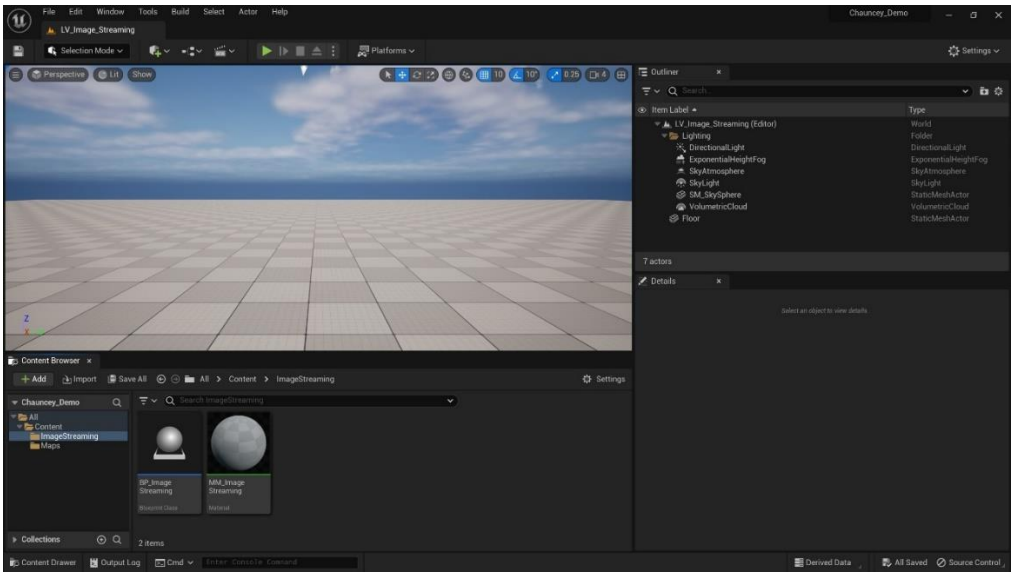

Рисунок 2.2 - Інтерфейс Unreal Engine 5

Хоча ігрові рушії технічно здатні обробляти широкий спектр 3D моделей, існують певні стандарти індустрії та вимоги окремих проєктів, які визначають оптимальні характеристики 3D контенту для використання в проєктах. Ці стандарти спрямовані на досягнення балансу між візуальною якістю та продуктивністю, забезпечуючи плавний ігровий процес на цільових платформах.

Стандарти ігрової індустрії та вимоги проєктів:

Кількість полігонів: Кількість полігонів у 3D моделі безпосередньо впливає на навантаження на графічний процесор (GPU). Занадто велика кількість полігонів може призвести до зниження частоти кадрів та погіршення продуктивності, особливо у складних сценах з великою кількістю об'єктів. Стандарти індустрії варіюються залежно від типу об'єкта, його важливості в сцені та цільової платформи. Наприклад, моделі персонажів зазвичай мають більшу кількість полігонів, ніж фонові об'єкти.

Якість текстур: Текстури визначають візуальні деталі та матеріальні властивості 3D моделі. Високоякісні текстури можуть значно покращити реалістичність, але також збільшують використання пам'яті та навантаження на GPU. Стандарти передбачають використання текстур оптимальної роздільної здатності, ефективне стиснення та створення атласів текстур для мінімізації кількості звернень до пам'яті.

Кількість та різновид карт: Різні типи карт використовуються для передачі різної інформації про поверхню моделі. Крім основної карти кольору (diffuse map), часто використовуються карти нормалей (normal maps), карти блиску (specular maps), карти затінення (ambient occlusion maps) та інші. Використання правильної кількості та типу карт може значно підвищити візуальну якість, але також впливає на продуктивність.

Рівні деталізації (LOD): LOD - це важлива техніка оптимізації, яка передбачає використання різних версій моделі з різною кількістю полігонів та якістю текстур, залежно від відстані до камери. Інколи останнім рівнем LOD є просто картинка певного об’єкту, часто таку методику використовують для дерев, кущів та великих гір. Це дозволяє зменшити навантаження на GPU при відображенні віддалених об'єктів, зберігаючи високу візуальну якість для об'єктів, що знаходяться близько до камери.

Важливо зазначити, що існують певні відмінності між ігровими рушіями у вимогах до 3D моделей, особливо щодо обробки карт нормалей. Карти нормалей використовуються для імітації дрібних деталей поверхні без збільшення кількості полігонів.

Unity зазвичай використовує карти нормалей, закодовані для OpenGL.

Unreal Engine використовує карти нормалей, закодовані для DirectX.

Ця відмінність означає, що карти нормалей, створені для одного рушія, будуть відображатися неправильно в іншому. Розробникам часто доводиться конвертувати карти нормалей при імпорті моделей між різними рушіями. Основна відмінність полягає в впуклості та випуклості, наочоно це зображено на риснку 2.3.

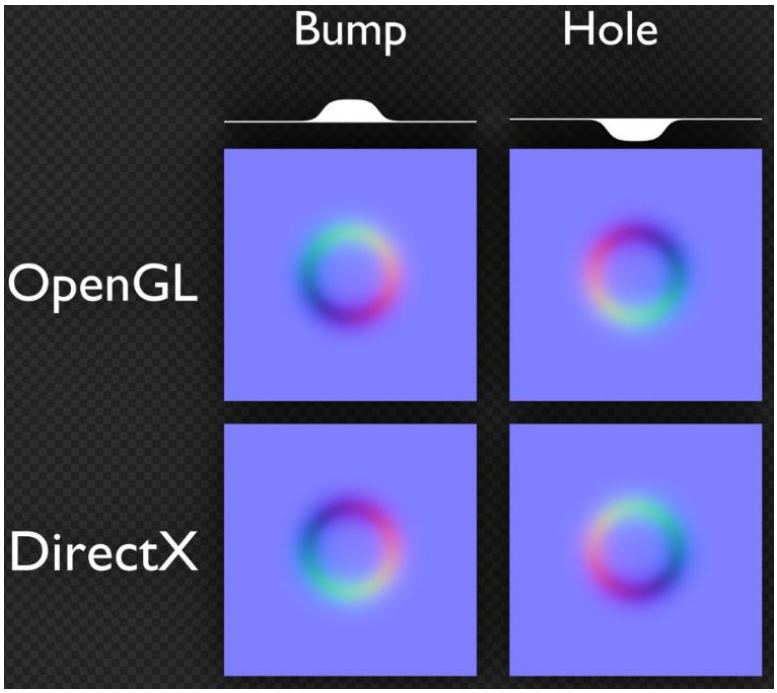

Рисунок 2.3 – відмінність карт нормалей OpenGL та DirectX

Також для 3D моделей є різні формати файлів і для кожної моделі потрібно знати в якому форматі їх конвертувати. Кожен формат має свої переваги та недоліки, і вибір конкретного формату часто залежить від вимог проєкту, використовуваного програмного забезпечення та цільової платформи.

Ось огляд трьох найпоширеніших форматів:

FBX (Filmbox) розроблений компанією Kaydara, а зараз належить Autodesk. Він широко використовується в індустрії розробки ігор та кіно. Підтримує широкий спектр даних, включаючи геометрію, текстури, матеріали, анімації та деформації. Забезпечує хорошу сумісність між різним програмним забезпеченням для 3D-моделювання та ігровими рушіями. Проте, він може бути досить великим за розміром файлу, особливо при зберіганні складної анімації.

OBJ (Object) був розроблений компанією Wavefront Technologies. Це простий та широко підтримуваний формат, особливо для статичних моделей. Він зберігає тільки геометрію, UV-координати та нормалі. Не підтримує анімацію або складні матеріали. Часто використовується як формат обміну між різними програмами для 3D-моделювання.

GlTF (GL Transmission Format) - відносно новий формат, розроблений Khronos Group спеціально для ефективної передачі 3D-контенту в реальному

часі. Він підтримує геометрію, текстури, матеріали, анімації та сцени. Оптимізований для швидкого завантаження та рендерингу. Стає все більш популярним у веб-додатках та мобільних іграх, а також є розширюваним, підтримує додаткові розширення.

2.2 Вибір програмного забезпечення для створення 3D моделей

ПЗ для створення 3D-моделей — це спеціалізоване програмне забезпечення, призначене для розробки, редагування та підготовки тривимірних об'єктів (моделей) шляхом надання інструментів для маніпулювання геометрією, нанесення текстур та матеріалів, створення UV-розгорток, а також базової або розширеної анімації та рендерингу, з метою створення візуальних асетів для подальшого використання в різних сферах, включаючи ігрову розробку, кіно, дизайн та інші.

Основна відмінність між програмним забезпеченням для створення 3D-моделей та ігровими рушіями полягає в їхньому первинному призначенні та наборі функцій. Програми для 3D-моделювання зосереджені на процесі створення, редагування та підготовки тривимірних об'єктів. Вони надають інструменти для моделювання геометрії, текстурування поверхонь, створення UV-розгорток, базової або просунутої анімації та рендерингу статичних зображень. Основна мета такого ПЗ – створення асетів, які потім можуть бути використані в різних галузях, включаючи ігрову розробку, кіноіндустрію, архітектурну візуалізацію та промисловий дизайн. Прикладами такого ПЗ є Blender, Autodesk Maya, 3ds Max та ZBrush. На противагу цьому, ігрові рушії є комплексними середовищами для розробки відеоігор та інших інтерактивних застосунків з візуалізацією в реальному часі. Вони включають в себе системи рендерингу, фізики, звуку, штучного інтелекту, управління ігровою логікою та анімацією, інструменти для створення рівнів, управління користувацьким інтерфейсом та можливості мережевої взаємодії. Головна їхня функція – забезпечити інтерактивність та реальний час візуалізації, використовуючи готові 3D-моделі, створені в спеціалізованому ПЗ. Прикладами ігрових рушіїв є Unity,

Unreal Engine та Godot. Таким чином, ПЗ для 3D-моделювання є інструментом для створення окремих тривимірних елементів, тоді як ігровий рушій є платформою, яка об'єднує ці елементи, додає інтерактивність та дозволяє створити кінцевий ігровий продукт або інтерактивний додаток.

Blender 3D - це безкоштовне та відкрите програмне забезпечення, яке пропонує повний набір інструментів для 3D-моделювання, скульптування, анімації, рендерингу, композитингу та редагування відео. Цей універсальний інструмент підходить для широкого спектру завдань, від створення ігор та візуалізації до анімаційних фільмів. Blender має активну спільноту користувачів та розробників, що забезпечує доступ до великої кількості навчальних матеріалів, плагінів та підтримки. Завдяки своїй безкоштовній природі та широким можливостям, Blender 3D став популярним вибором серед незалежних розробників, студентів та професіоналів [14].

Інтерфейс Blender 3D зображений на рисунку 2.4.

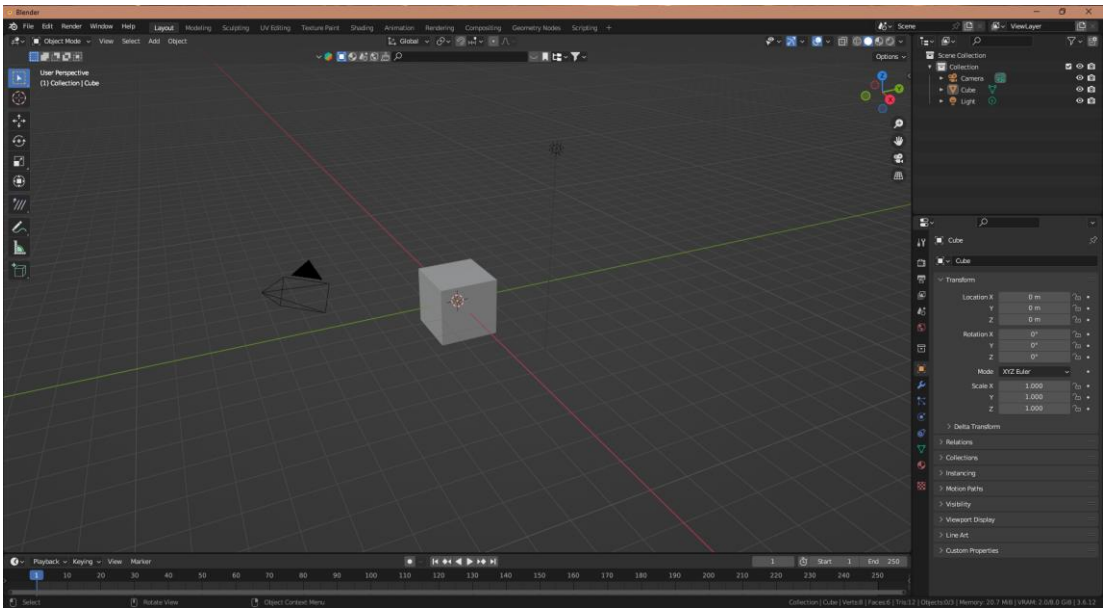

Рисунок 2.4 - Інтерфейс Blender 3D

Maya - це професійне програмне забезпечення для 3D-моделювання, анімації та рендерингу, розроблене Autodesk. Воно широко використовується у кіноіндустрії, телебаченні та розробці відеоігор для створення складних персонажів, візуальних ефектів та анімації. Maya пропонує потужні інструменти

для моделювання, анімації персонажів, симуляції, такі як симуляція одягу та волосся, та візуальних ефектів, включаючи динаміку рідин та частинок. Хоча Maya є потужним інструментом, він може бути складним у вивченні та має високу вартість, що робить його більш придатним для великих студій та досвідчених професіоналів [15].

Інтерфейс Maya зображений на рисунку 2.5.

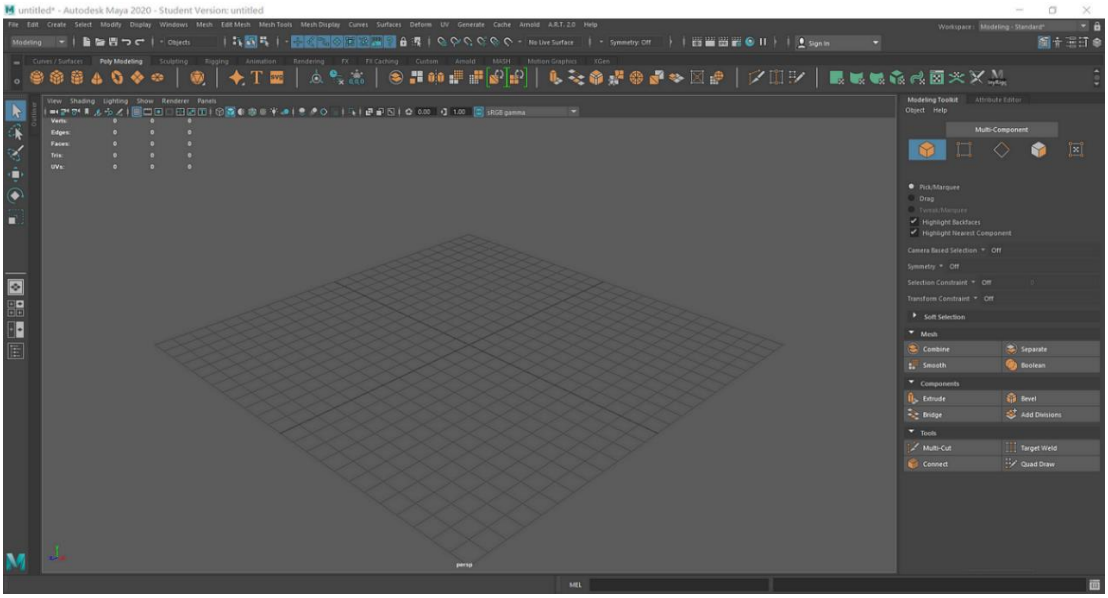

Рисунок 2.5 - Інтерфейс Maya

3Ds Max - це професійне програмне забезпечення для 3D-моделювання, анімації та рендерингу, також розроблене Autodesk. Воно популярне у архітектурній візуалізації, дизайні продуктів та виробництві відеоігор. 3ds Max має широкий набір інструментів для моделювання, включаючи полігональне моделювання та моделювання на основі сплайнів, анімації, симуляції, такої як симуляція динаміки твердого тіла, та рендерингу, включаючи інтеграцію з потужними рендерерами, такими як Arnold. Як і Maya, 3Ds Max може бути складним у вивченні та має високу вартість [16].

Інтерфейс 3Ds Max зображено на рисунку 2.6.

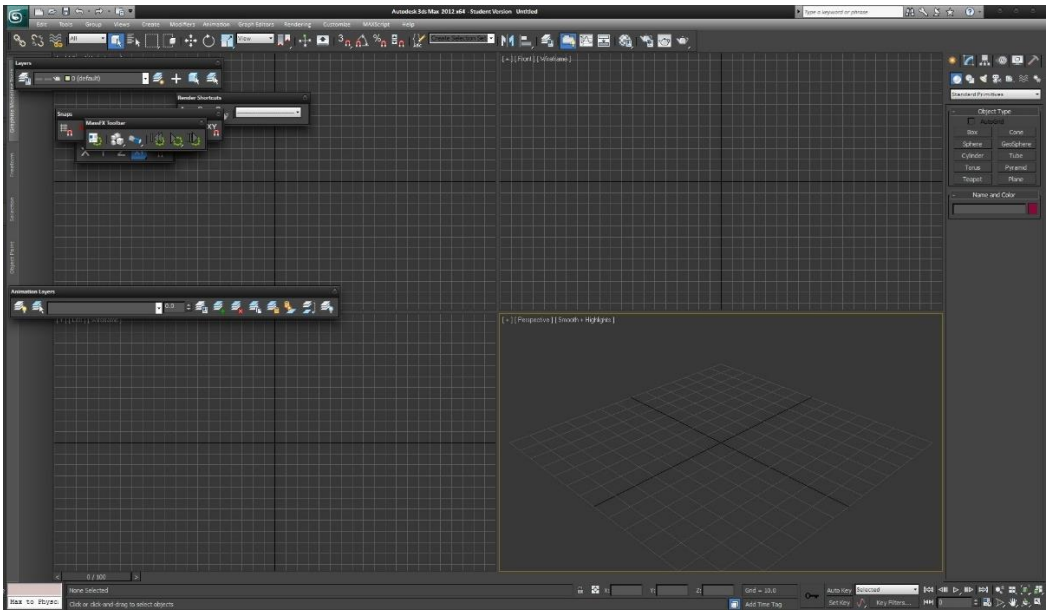

Рисунок 2.6 - Інтерфейс 3Ds Max

Усі три програми є потужними інструментами для 3D-моделювання та анімації, вони пропонують широкий спектр інструментів для створення 3D-контенту для різних галузей. У кожного є свої переваги та недоліки, хоча Blender 3D є універсальним безкоштовним продуктом, Maya є стандартом індустрії для анімації персонажів та візуальних ефектів, тоді як 3ds Max популярний у архітектурній візуалізації та дизайні продуктів. Blender 3D має меншу криву навчання, ніж Maya та 3Ds Max, але може бути менш потужним у певних областях.

2.3 Основні етапи створення 3D моделей

Процеси створення тривимірних моделей є складними та багатогранними, а їх послідовність і кількість можуть варіюватися залежно від специфіки проєктів. Однак, при розробці тривимірних ассетів можна виокремити наступні ключові етапи. Ефективне управління часом, уникнення надмірної концентрації на окремих аспектах та вміння оперативно знаходити необхідну інформацію є визначальними факторами успішного виконання різноманітних проєктів, кожен з яких може вимагати унікальних знань і підходів [17].

2.3.1 Пошук та збір референсів

На початковому етапі відтворення будь-якого об’єкта постає потреба у його візуалізації. Для цього здійснюється пошук та збір релевантних зображень об’єкта з різних перспектив. Отримані візуальні матеріали сприяють формуванню цілісного уявлення про форму, пропорції та характерні особливості об’єкта моделювання. Зібрані референси можуть бути розміщені на окремих дисплеях або імпортовані безпосередньо в робоче середовище програмного забезпечення для тривимірного моделювання.

PureRef є спеціалізованим програмним забезпеченням, розробленим для ефективної організації та зручного використання референсних зображень у процесі створення візуального контенту, зокрема тривимірних моделей.

На рисунку 2.7 зображено приклад використання PureRef.

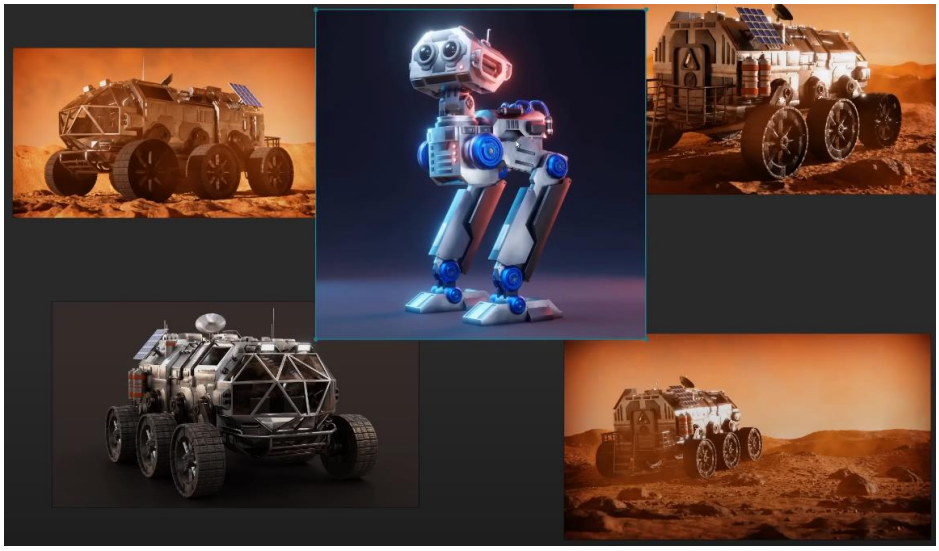

Рисунок 2.7 - Приклад використання PureRef

На рисунку 2.8 зображено приклад використання стандартного інструментарію Blender 3D для візуалізації референсу.

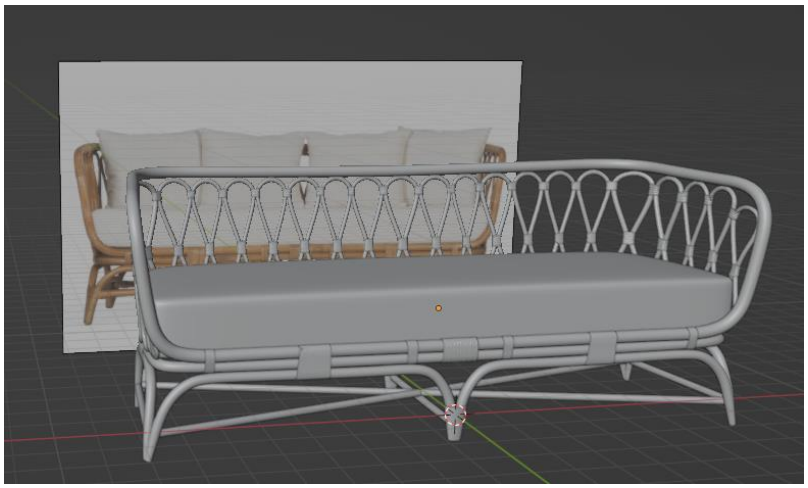

Рисунок 2.8 - Стандартний інструментарій Blender 3D для візуалізації референсу

2.3.2 Моделювання

Процес моделювання тривимірної моделі є ключовим етапом у створенні 3D контенту, під час якого формується геометрія об'єкта у віртуальному просторі. Цей процес включає в себе ряд послідовних дій, спрямованих на перетворення концепції або референсів у цифрову тривимірну форму.

Блокінг (Blocking out). Блокінг — це початковий етап моделювання, на якому закладаються основні форми та пропорції майбутньої моделі. На цьому етапі не потрібно прагнути до високої деталізації; головна мета — створити грубий, але впізнаваний силует об'єкта. Блокінг часто починається з додавання базових геометричних примітивів (кубів, сфер, циліндрів, конусів тощо), які приблизно відповідають основним частинам об'єкта. Ці примітиви потім масштабуються, переміщуються та обертаються, щоб сформувати загальну композицію та габарити моделі. На етапі блокінгу важливо правильно визначити співвідношення розмірів різних частин об'єкта та їхню загальну форму. Це закладає основу для подальшої деталізації. Блокінг має бути швидким та ітеративним процесом. Не бійтеся експериментувати з різними формами та композиціями, перш ніж зупинитися на найбільш вдалому варіанті.

На рисунку 2.9 зображено приклад блокінгу [18].

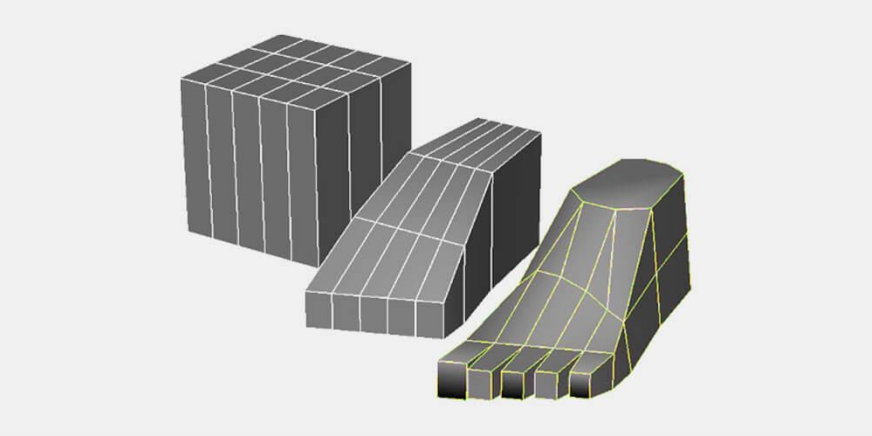

Рисунок 2.9 – Блокінг в моделюванні

Хай полі (High Poly). Хай полі — це етап створення високодеталізованої версії моделі. Ця модель містить велику кількість полігонів, що дозволяє відображати дрібні деталі, складні поверхні та плавні переходи.

На етапі хай полі активно використовуються модифікатори для прискорення та оптимізації процесу моделювання:

- Mirror (Дзеркало): Дозволяє моделювати лише одну симетричну половину об'єкта, а друга автоматично відображається. Це значно економить час при створенні симетричних моделей.
- Subdivision Surface (Згладжування поверхні): Збільшує кількість полігонів і згладжує поверхню моделі, створюючи більш органічні та плавні форми. Кількість рівнів підрозділу контролює ступінь деталізації. Важливо розуміти, що підрозділ генерує нову геометрію, тому його слід застосовувати обдумано.
- Boolean (Булеві операції): Дозволяють виконувати операції об'єднання, віднімання та перетину одних об'єктів з іншими, створюючи складні форми та отвори. Однак, булеві операції часто створюють неоптимальну топологію, тому після їх використання може знадобитися ручне виправлення сітки.
- Bevel (Фаска): Створює скошені краї на гострих кутах, що робить модель більш реалістичною та запобігає появі різких тіней. Фаски також додають додаткові ребра, які можуть бути важливі для відображення відблисків.

- Array (Масив): Дозволяє створювати множинні копії об'єкта, розташовані за певним шаблоном. Корисно для створення повторюваних елементів, таких як болти, зубці тощо.
- Solidify (Потовщення): Додає товщину плоским поверхням, що необхідно для створення реалістичних об'єктів з внутрішнім об'ємом.

Також на цьому етапі вручну додаються дрібні деталі, такі як виступи, западини, гострі краї тощо, використовуючи інструменти екструдування, вставки ребер. По заверешенню цього етапу ми отримуємо деталізовану модель об’єкту з великою кількістю полігонів, це означає що ця модель не підходить для рендеру в реальному часі.

High Poly модель зображена на рисунку 2.10.

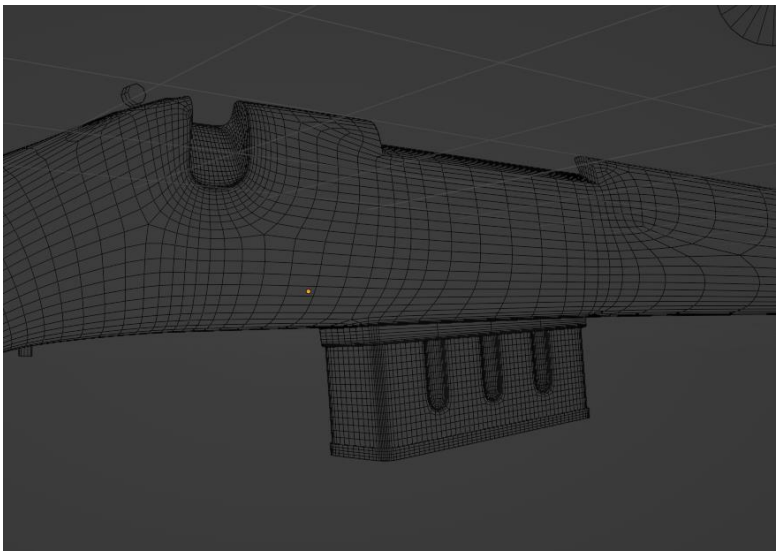

Рисунок 2.10 - High Poly

Скульптинг (Sculpting). Скульптинг — це метод моделювання, який дозволяє створювати органічні та високодеталізовані форми, ніби ліплячи з віртуальної глини. Програми для 3D моделювання (включаючи Blender) надають набір пензлів з різними функціями (витягування, згладжування, додавання об'єму, вирізання тощо). Скульптинг зазвичай вимагає дуже високої щільності полігональної сітки для відображення дрібних деталей. Ідеально підходить для створення персонажів, скульптур, природних об'єктів та будь-яких форм зі складними органічними поверхнями.

На рисунку 2.11 зображено вікно скульптингу в Blender 3D, де зліва знаходяться всі стандартні пензлі [19].

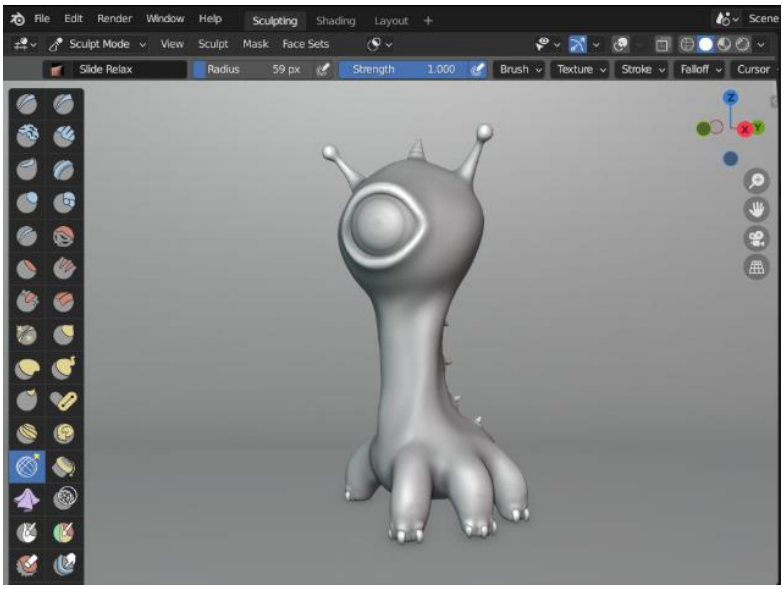

Рисунок 2.11 – Скульптинг

Лоу полі (Low Poly). Лоу полі — це низькополігональна версія моделі, оптимізована для рендерінгу в реальному часі. В ігровій індустрії кожна додаткова грань об'єкта збільшує навантаження на процесор та відеокарту, що може призвести до зниження продуктивності (FPS). Лоу полі моделі мають мінімальну кількість полігонів, необхідну для збереження впізнаваної форми об'єкта, що забезпечує плавну роботу гри.

Зробити лоу полі модель можна різними способами, кожний з яких має переваги та недоліки. Можна добавити на хай полі меш модифікатор Decimate, який автоматично знизить кількість полігонів до вказаної величини, але для такого способу хай полі моделька має бути також відповідною.

Найкращим способом є ручна ретопологія. Ретопологія - це процес створення нової, оптимізованої полігональної сітки поверх існуючої, зазвичай більш щільної та неоптимальної, 3D моделі. Лоу полі модель служить основою для відображення в грі. Деталі, створені на високополігональній моделі, "запікаються" (bake) на текстури низькополігональної моделі, створюючи ілюзію

високої деталізації при значно меншій кількості полігонів. Це дозволяє досягти візуально привабливих результатів без надмірного навантаження на систему.

Відмінність сітки лоу полі та хай полі моделі зображена на рисунку 2.12 [20].

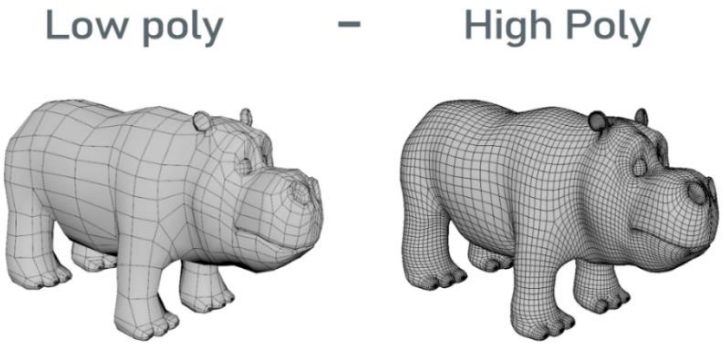

Рисунок 2.12 - Відмінність сітки лоу полі та хай полі моделі

2.3.3 Розгортка, бейкінг та текстурування

Накладання швів та розгортка (UV Unwrapping). UV-розгортка — це процес розгортання тривимірної поверхні моделі на двовимірну площину для подальшого нанесення текстур. Шви визначають, де буде розрізана 3D модель, щоб її можна було плоско розгорнути.

Принципи правильного накладання швів:

- Природні розриви: Намагайтеся розміщувати шви в місцях природних розривів геометрії або там, де вони будуть найменш помітні (наприклад, на внутрішніх сторонах, під складками, на стиках різних частин).
- Мінімізація спотворень: Правильно розміщені шви допомагають мінімізувати розтягування або стискання текстур на поверхні моделі.
- Циліндричні об'єкти: Часто розгортаються одним поздовжнім швом.
- Сферичні об'єкти: Можуть розгортатися кількома меридіональними та екваторіальними швами.
- Кубічні об'єкти: Кожна грань може бути розгорнута окремо.
- Складні органічні форми: Можуть вимагати більш складної сітки швів, щоб уникнути значних спотворень.

Правильна розгортка, на прикладі кубу, зображена на рисунку 2.13 [21].

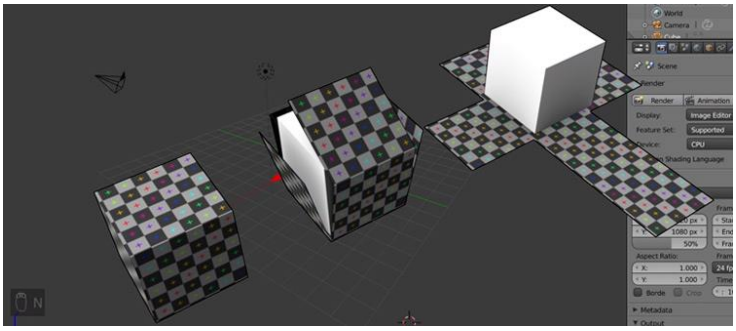

Рисунок 2.13 – Розгортка кубу

Не менш важливим є оптимізоване використання полотна розгортки.UV-острови (розгорнуті частини моделі) повинні бути розміщені на текстурній карті якомога щільніше, щоб максимально ефективно використовувати доступний простір текстури. Вони не повинні перекриватися, інакше текстури будуть накладатися одна на одну, але за потреби це допускається.

Бажано, щоб UV-острови мали приблизно однаковий масштаб, щоб щільність пікселів текстури була однаковою на різних частинах моделі. Більше місця на UV-карті слід відводити для важливих деталей моделі, щоб забезпечити вищу якість текстур у цих областях. Стандартом для розгортки є покривання 85% площі полотна, а ідеальним 95% і більше [22].

На рисунку 2.14 зображеий приклад оптимізованого використання полотна розгортки [23].

Рисунок 2.14 - Оптимізоване використання полотна розгортки

Запікання (Baking). Бейкінг — це процес перенесення деталей з високополігональної (хай полі) моделі на текстурні карти низькополігональної (лоу полі) моделі.

Під час бейкінгу програмне забезпечення "випускає промені" з кожної точки поверхні низькополігональної моделі. Ці промені шукають найближчу відповідну точку на високополігональній моделі. Інформація про нормалі, освітлення, кривизну та інші деталі з цієї точки хай полі "запікається" у вигляді текстурної карти для лоу полі моделі.

Типи карт, що запікаються:

- Normal Map (Карта нормалей): Містить інформацію про орієнтацію поверхонь хай полі моделі, що дозволяє лоу полі моделі виглядати так, ніби вона має значно більше деталей, ніж насправді.
- Ambient Occlusion (AO) Map (Карта затінення): Відображає затінення в заглибленнях та на стиках поверхонь, додаючи реалістичності освітленню.
- Curvature Map (Карта кривизни): Визначає вигнуті та гострі ділянки поверхні, що корисно для створення ефектів зносу та виділення країв при текстуруванні.
- Thickness Map (Карта товщини): Відображає товщину об'єкта в різних місцях.
- ID Map (Карта ідентифікаторів): Використовує різні кольори для виділення окремих частин моделі, що полегшує їх маскування та редагування при текстуруванні.

Наочно принцип роботи бейкінгу зорбажено на рисунку 2.15 [24].

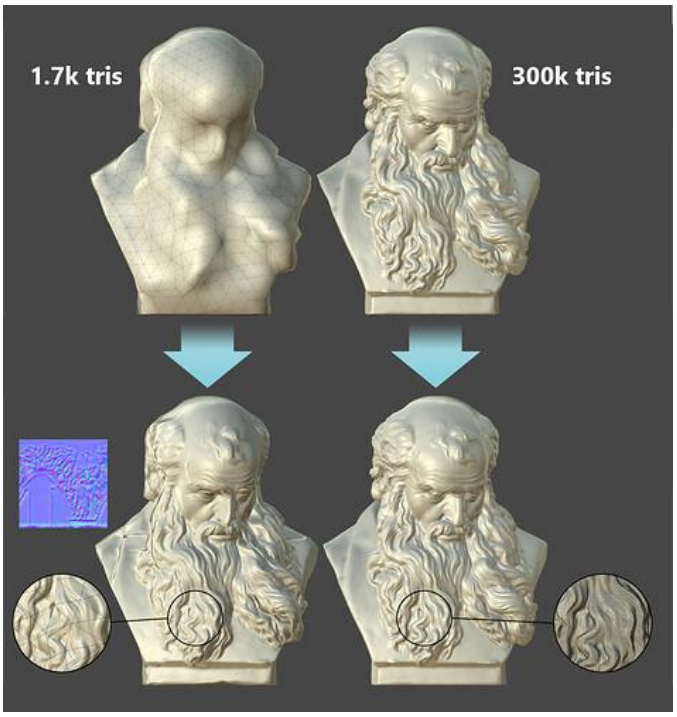

Рисунок 2.15 – Бейкінг

Текстурування — це процес нанесення кольору, матеріальних властивостей та деталей на UV-розгорнуту та запечену низькополігональну модель. Текстурування зазвичай виконується в спеціалізованих програмах, таких як Substance Painter або Adobe Substance 3D Designer, які надають потужні інструменти для створення та редагування текстур. На цьому етапі визначаються властивості поверхні об'єкта, такі як колір (Base Color), шорсткість (Roughness), металевість (Metallic), висота (Height) та нормалі (за допомогою запеченої Normal Map). Художники можуть безпосередньо малювати деталі на 3D моделі або на 2D розгортці.

Процедурні текстури генеруються математично і можуть створювати складні та повторювані візерунки. Substance Painter та інші програми пропонують бібліотеки готових матеріалів та масок, які можна налаштовувати та комбінувати для швидкого створення реалістичних поверхонь [25].

Текстурування зображено на рисунку 2.16 [26].

Рисунок 2.16 – Текстурування

2.3.4 Ригінг та анімація

У випадках, коли створюються об'єкти, які повинні рухатися (наприклад, персонажі, тварини, транспортні засоби з рухомими частинами), після етапу моделювання та текстурування виконуються процеси створення скелета (ригінг) та анімації.

Ригінг (Rigging) - це процес створення цифрового скелета всередині 3D моделі. Скелет складається з набору з'єднаних між собою "кісток" (bones), які можуть обертатися та переміщатися. Кожна кістка асоціюється з певною частиною полігональної сітки моделі (процес, що називається скінінгом або вейт пейнтінгом). Це дозволяє контролювати деформацію моделі при русі кісток.

Кістки організовуються в ієрархічну структуру, що відображає реальну будову скелета або механізму. Визначається ступінь впливу кожної кістки на прилеглі ділянки полігональної сітки. Цей процес налаштовується за допомогою інструментів "фарбування ваг" (weight painting), де різні кольори відображають силу впливу кістки на вертекси. Правильно налаштовані ваги забезпечують плавні та реалістичні деформації під час руху.

Ригінг зображено на рисунку 2.17 [27].

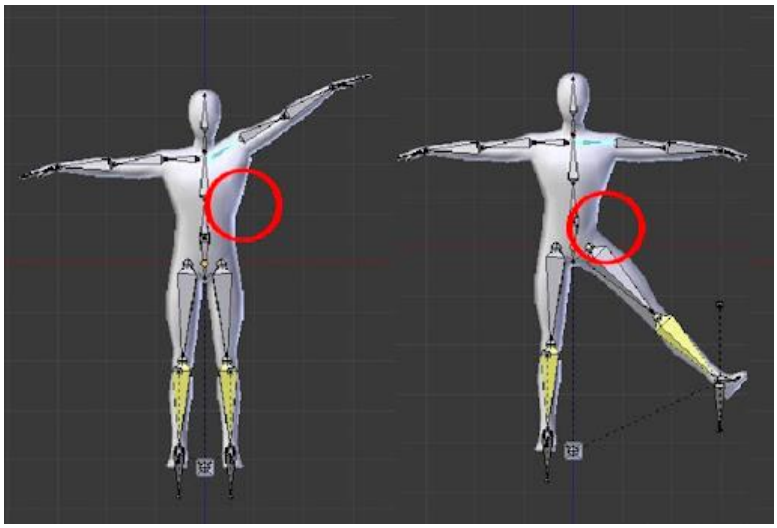

Рисунок 2.17 – Ригінг

Анімація - процес створення послідовності рухів моделі в часі шляхом зміни положення, обертання та масштабування кісток скелета. Аніматори визначають ключові положення моделі в певні моменти часу (ключові кадри). Проміжкові кадри між ключовими кадрами автоматично обчислюються програмним забезпеченням (процес інтерполяції).

Для більш тонкого контролю над анімацією використовуються графіки кривих, які відображають зміну параметрів анімації (положення, обертання, масштаб) з часом. Редагування цих кривих дозволяє досягти складних та реалістичних рухів. Для повторюваних рухів (наприклад, ходьби, бігу) створюються цикли анімації, які безперервно відтворюються.

Для органічних об'єктів, які будуть анімуватися та деформуватися, правильна топологія (полігон флоу) є критично важливою для забезпечення плавної та реалістичної деформації без артефактів (складок, розтягувань).

Найкращим варіантом для рухливих органічних об'єктів є топологія, що складається переважно з чотирикутних полігонів (квадів). Квади деформуються більш передбачувано, ніж трикутники (триси) або багатокутники (н-гони). Полігон флоу повинен слідувати основним м'язовим групам та напрямкам руху об'єкта. Наприклад, на руках та ногах петлі ребер повинні огинати м'язи, що забезпечить правильну деформацію при згинанні. У областях, які будуть сильно деформуватися (суглоби, місця згину), бажано мати більшу щільність полігонів,

щоб забезпечити плавність деформації та уникнути розтягування текстур. Трикутники та багатокутники можуть створювати проблеми при деформації, призводячи до нерівномірного розтягування та візуальних артефактів. Їх слід уникати, особливо в областях, що активно анімуються. Якщо вони неминучі, їх слід розміщувати в менш помітних або статичних областях.

Навколо суглобів (лікті, коліна, плечі тощо) рекомендується створювати кругові петлі ребер. Це допомагає забезпечити плавне згинання та обертання без утворення складок. Загальний полігон флоу повинен бути якомога простішим та передбачуваним, без різких змін у щільності полігонів або хаотичного розташування ребер [28].

На рисунку 2.18 зображено основні етапи анімації.

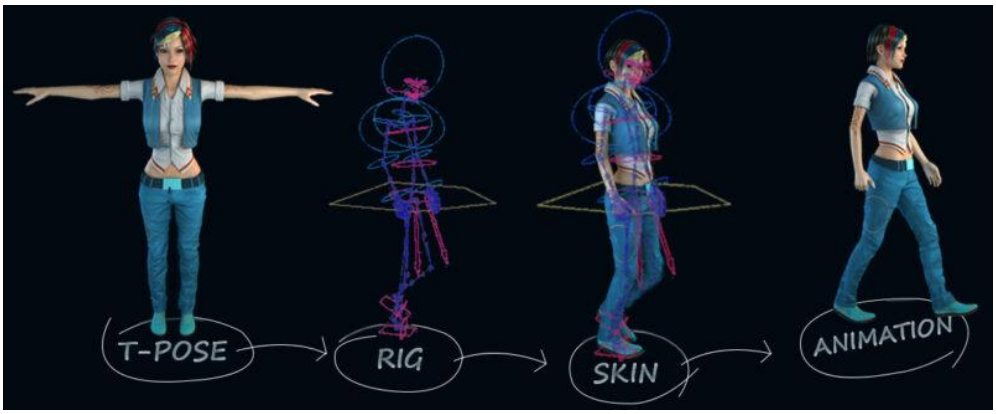

Рисунок 2.18 - Основні етапи анімації

2.3.5 Експорт та рендер

Експорт — це заключний етап, на якому готова модель з текстурами передається в ігровий рушій (наприклад, Unreal Engine, Unity). Модель зазвичай експортується у форматі, який підтримується цільовим рушієм (наприклад, FBX, OBJ).

При експорті необхідно враховувати налаштування, такі як застосування трансформацій, експорт UV-координат, нормалей, тангенсів тощо. У рушії модель імпортується, і до неї підключаються створені текстури в відповідних слотах матеріалу (Base Color, Normal, Roughness, Metallic тощо). Можуть бути

додатково налаштовані параметри матеріалу, беспосередньо в рушії, для досягнення бажаного візуального вигляду.

Рендеринг — це процес генерації двовимірного зображення з тривимірної сцени. На цьому етапі враховуються геометрія моделі, налаштування матеріалів, освітлення, камери та інші параметри сцени для створення фінального візуального представлення.

Перед рендерингом необхідно правильно налаштувати всі елементи сцени:

- Розміщення моделі: Модель повинна бути розміщена у бажаній позиції в тривимірному просторі.
- Налаштування матеріалів: Всі матеріали моделі повинні бути правильно налаштовані з використанням текстур та шейдерів, щоб визначати, як світло взаємодіє з поверхнею об'єкта (колір, відбиття, заломлення, прозорість тощо).
- Освітлення: Розміщення та налаштування джерел світла (точкових, спрямованих, плоских, фонового освітлення) є критично важливим для створення настрою, підкреслення форми та деталей моделі. Налаштовуються інтенсивність, колір, тіні та інші параметри світла.
- Налаштування камери: Визначається положення, орієнтація та параметри віртуальної камери (кут огляду, глибина різкості), яка буде "фотографувати" тривимірну сцену.

Вибір рушія рендерингу (Render Engine): Існують різні рушії рендерингу, кожен з яких має свої алгоритми та особливості:

- Рушії реального часу (Real-time Engines): (Наприклад, Eevee в Blender, двигуни в ігрових рушіях) Оптимізовані для швидкого рендерингу з меншою фотореалістичністю. Використовуються для попереднього перегляду, анімації та інтерактивних застосунків.
- Рушії трасування променів (Ray Tracing Engines): (Наприклад, Cycles в Blender, V-Ray, Arnold) Забезпечують високий рівень фотореалізму за рахунок симуляції шляху світлових променів у сцені. Вимагають значно більше обчислювальних ресурсів та часу на рендеринг.

На рисунку 2.19 зображено відмінність між рушіями рендерингу Cycles та Evee.

Рисунок 2.19 - Відмінність між рушіями рендерингу

Налаштування параметрів рендерингу: Залежно від обраного рушія налаштовуються різні параметри, такі як:

- Розмір зображення (Resolution): Визначає кількість пікселів у фінальному зображенні.
- Кількість семплів (Samples): Впливає на якість зображення та рівень шуму (чим більше семплів, тим чистіше зображення, але довший час рендерингу).
- Глибина кольору (Color Depth): Визначає кількість кольорів, які можуть бути відображені.
- Формат файлу (File Format): Обирається формат для збереження фінального зображення (наприклад, PNG, JPG, TIFF).

Після налаштування всіх параметрів запускається процес рендерингу, під час якого програмне забезпечення обчислює колір кожного пікселя фінального зображення на основі інформації про геометрію, матеріали, освітлення та камеру. Цей процес може займати від кількох секунд до багатьох годин, залежно від складності сцени та потужності комп'ютера.

Після завершення рендерингу отримане зображення може бути піддане постобробці в спеціалізованих програмах (наприклад, Adobe Photoshop, Blender Compositor) для внесення фінальних корекцій кольору, контрасту, додавання ефектів тощо.

Етап рендерингу є кульмінацією процесу створення 3D моделі та візуалізації, перетворюючи цифрову тривимірну сцену на готове двовимірне зображення. Якість рендерингу значною мірою залежить від ретельності налаштування всіх параметрів сцени.

На рисунку 2.20 спостерігаємо фінальний рендер та пост обробку автора Dmytro Nechyporenko [29].

Рисунок 2.20 - Фінальний рендер та пост обробка

2.4 Вибір оптимальної IT-інфраструктури

Ефективна та безперебійна розробка 3D-моделей та їх подальша інтеграція в ігрові рушії є ресурсомістким процесом, який висуває значні вимоги до програмного та апаратного забезпечення. Вибір оптимальної IT-інфраструктури є критично важливим етапом, оскільки він безпосередньо впливає на продуктивність робочого процесу, швидкість виконання операцій, якість кінцевого продукту та загальний комфорт розробника.

Недостатньо потужне "залізо" або невідповідне програмне забезпечення може призвести до значних затримок (лагів), частих збоїв програм, тривалого часу рендерингу та імпорту/експорту файлів, що суттєво уповільнює розробку та знижує її ефективність. З іншого боку, надмірне інвестування у компоненти, які не будуть використані на повну потужність, може бути економічно невиправданим.

Для успішного виконання даної бакалаврської дипломної роботи, яка включає глибоке теоретичне вивчення та практичну реалізацію створення оптимізованих 3D-моделей для ігрових рушіїв, було визначено наступні мінімальні вимоги до програмного та апаратного забезпечення:

- Windows 10 є де-факто стандартом для розробки ігор та 3D-контенту завдяки своїй широкій сумісності з більшістю професійного програмного забезпечення (Blender, Substance Painter, Unreal Engine) та драйверами для графічних процесорів. Вона забезпечує стабільну платформу для ресурсомістких застосунків та має розвинену екосистему інструментів для розробників.

- GPU є одним з найважливіших компонентів для 3D-моделювання, текстурування та роботи з ігровими рушіями. Він відповідає за рендеринг графіки в реальному часі у 3D-редакторах, візуалізацію складних сцен, виконання ресурсомістких операцій (таких як скульптування з великою кількістю полігонів, бейкінг текстур) та відображення ігрових світів. GTX 1060 (або сучасніший аналог, наприклад, з серії RTX) забезпечує достатню продуктивність для комфортної роботи з основними інструментами, дозволяючи обробляти моделі середньої складності, ефективно запікати текстури та тестувати проєкти в ігрових рушіях без значних компромісів у якості та швидкості. Для використання трасування променів у реальному часі (як у Unreal Engine 5 з Lumen/Nanite), бажано мати GPU з підтримкою RT ядер (наприклад, серії NVIDIA RTX або AMD Radeon RX 6000+).

- CPU відіграє ключову роль у таких завданнях, як запуск програм, компіляція шейдерів, обробка фізики, завантаження складних сцен, обробка даних та виконання скриптів. Багатопотоковість Ryzen 5 1600 забезпечує достатню продуктивність для паралельного виконання декількох завдань (наприклад, моделювання у Blender та завантаження асетів в Unreal Engine), прискорюючи робочий процес. Сучасніші багатоядерні процесори значно покращують швидкість операцій у таких програмах, як Blender (зокрема, для рендерингу на CPU) та при

компіляції коду в ігрових рушіях.

- 3D-моделювання та робота з ігровими рушіями є пам'ятоємними процесами. 16 ГБ ОЗП вважається мінімально комфортним обсягом для одночасного запуску 3D-редактора, програми для текстурування та ігрового рушія. Недостатній обсяг RAM призведе до постійного використання файлу підкачки на жорсткому диску, що значно уповільнить роботу системи. Більший обсяг (32 ГБ і більше) рекомендується для роботи зі складними сценами, високополігональними моделями, великими текстурними атласами та одночасним запуском кількох ресурсомістких застосунків.

- Blender 3D: Обрано як основний інструмент для 3D-моделювання, скульптування, UV-розгортки та ригінгу/анімації. Його відкритий вихідний код, потужний функціонал та активна спільнота роблять його ідеальним вибором для бакалаврської роботи.

- Adobe Substance Painter**:** Необхідний для професійного текстурування, PBR-матеріалів та бейкінгу карт. Це індустріальний стандарт, який дозволяє створювати високоякісні та реалістичні текстури.

- Unity: Вибрано як основний ігровий рушій для інтеграції, налаштування та візуалізації 3D-моделей. Його передові графічні можливості, система візуального програмування (Scripting) та гнучкі налаштування оптимізації дозволяють ефективно працювати з великими проєктами та реалізовувати високоякісні візуальні рішення. Ретельний вибір та налаштування цих компонентів забезпечують надійну основу для виконання всіх етапів дипломної роботи, від теоретичного аналізу до практичної розробки та оптимізації 3D-моделей.

На рисунку 2.21 зображена комплектація системи, на якій проводилась робота.

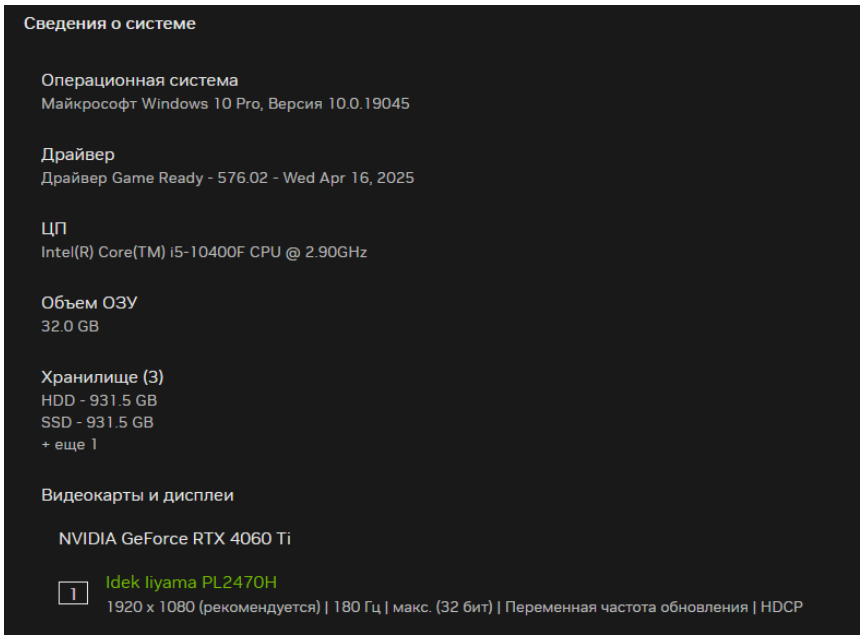

Рисунок 2.21 – Комплектація системи, на якій проводилась робота

2.5 Висновки

У даному розділі було всебічно розглянуто теоретичні аспекти створення тривимірних моделей, орієнтованих на використання в сучасних ігрових рушіях, а також методи їх оптимізації. Проаналізовано ключові характеристики та вимоги провідних ігрових рушіїв, Unity та Unreal Engine, до 3D контенту, зокрема щодо кількості полігонів, якості текстур, використання різних типів карт та рівнів деталізації (LOD). Підкреслено важливість врахування специфічних вимог кожного рушія, таких як кодування карт нормалей та формати файлів.

Крім того, було здійснено ґрунтовний огляд основного програмного забезпечення для створення 3D моделей, включаючи Blender, Maya та 3ds Max, виділено їхні особливості, переваги та недоліки. Наголошено на тому, що вибір конкретного програмного забезпечення залежить від індивідуальних потреб розробника, складності проєкту та наявних ресурсів. Особливу увагу приділено обґрунтуванню вибору оптимальної IT-інфраструктури, що включає програмне та апаратне забезпечення. Визначено мінімальні вимоги до GPU, CPU, ОЗП та операційної системи, які забезпечують необхідну продуктивність для комфортної розробки ресурсомістких 3D-проектів.

Детально розглянуто основні етапи створення 3D моделей для ігрових рушіїв, починаючи з пошуку та збору референсів, через блокінг, створення високо- та низькополігональних моделей, скульптування (за потреби), UV-розгортку, бейкінг, текстурування, ригінг та анімацію (для рухливих об'єктів), і завершуючи експортом та рендерингом. Для кожного етапу були описані ключові принципи, інструменти та найкращі практики, включаючи важливість правильного полігон флоу для анімованих органічних моделей та оптимізованого використання UV-простору.

Особливу увагу також було приділено процесам, спрямованим на оптимізацію 3D моделей для забезпечення високої продуктивності в ігрових рушіях, таким як створення низькополігональних моделей, використання рівнів деталізації та ефективне текстурування.

Таким чином, даний розділ заклав міцний теоретичний та інфраструктурний фундамент для подальшого дослідження процесу створення 3D моделей для ігрових рушіїв, визначивши основні етапи, інструменти, принципи оптимізації та необхідні технічні ресурси.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СТВОРЕННЯ 3D-МОДЕЛЕЙ ДЛЯ ІГРОВИХ РУШІЇВ

3.1 Проєктування інформаційної технології

На рисунку 3.1 зображена діаграма діяльності для загального процесу створення моделей (Activity Diagram).

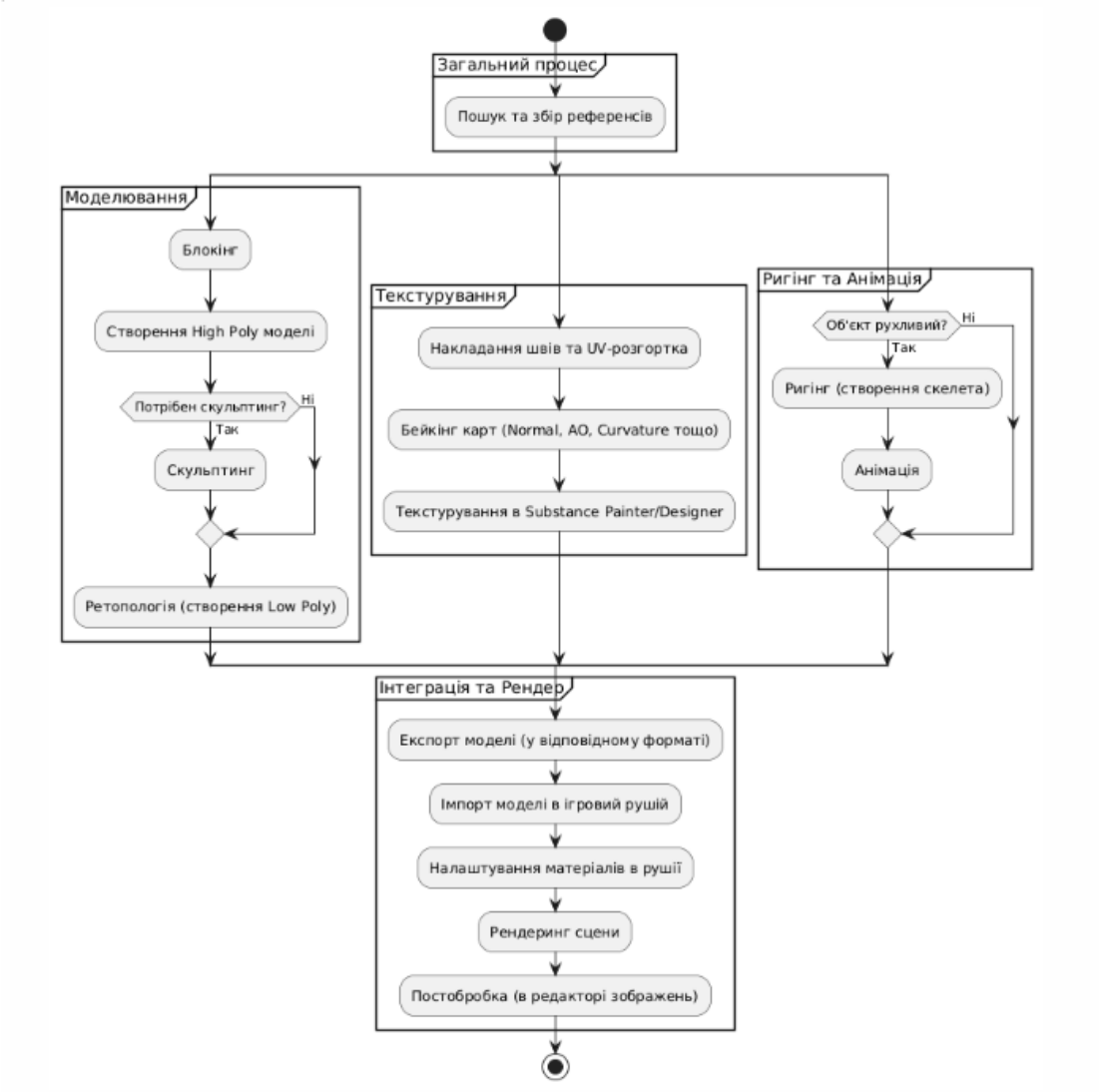

Рисунок 3.1 - Діаграма діяльності(Activity Diagram) для загального процесу створення моделей

Ця діаграма детально ілюструє послідовність етапів створення 3D-моделі, починаючи від загального процесу і закінчуючи рендерингом. Вона поділена на декілька основних "доріжок" або фаз:

Загальний процес:

- Починається з "Пошук та збір референсів" – це є важливим початковим етапом для будь-якого художнього проєкту, який забезпечує візуальну основу та стилістичне спрямування.

Моделювання:

- Блокінг: Створення грубої форми моделі, визначення основних пропорцій.
- Створення High Poly моделі: Деталізація моделі з великою кількістю полігонів.
- Потрібен скульптинг? (decision node): Перевірка, чи потрібен скульптинг.

А) Так: Перехід до "Скульптинг" – додавання дрібних деталей, органічних форм.

Б) Ні: Пропускається скульптинг.

- Ретопологія (створення Low Poly): Оптимізація моделі шляхом створення версії з меншою кількістю полігонів, що є критично важливим для ігрових рушіїв для підвищення продуктивності.

Текстурування:

- Накладання швів та UV-розгортка: Процес "розгортання" 3D-моделі у 2D-простір для подальшого нанесення текстур.
- Бейкінг карт (Normal, AO, Curvature тощо): Створення спеціальних карт (нормалей, оклюзії, кривизни), які дозволяють передати деталі High Poly моделі на Low Poly без збільшення полігонів.
- Текстурування в Substance Painter/Designer: Безпосереднє нанесення текстур на модель, використовуючи спеціалізоване програмне забезпечення.

Ригінг та Анімація:

- Об'єкт рухливий? (decision node): Перевірка, чи модель потребує анімації.

А) Так:

- Ригінг (створення скелета): Створення "скелета" для моделі, що дозволяє її деформувати та анімувати.

- Анімація: Створення послідовності рухів для моделі.

Б) Ні: Пропускається ригінг та анімація.

Інтеграція та Рендер:

- Експорт моделі (у відповідному форматі): Збереження моделі у форматі, сумісному з ігровим рушієм.
- Імпорт моделі в ігровий рушій: Завантаження моделі до Unity
- Налаштування матеріалів в рушії: Застосування та налаштування текстур та шейдерів на моделі всередині рушія.
- Рендеринг сцени: Фінальне створення зображення або відео з моделі.
- Пост-обробка (в редакторі зображень): Додаткові корекції та покращення зображення після рендерингу (наприклад, кольорова корекція, ефекти).

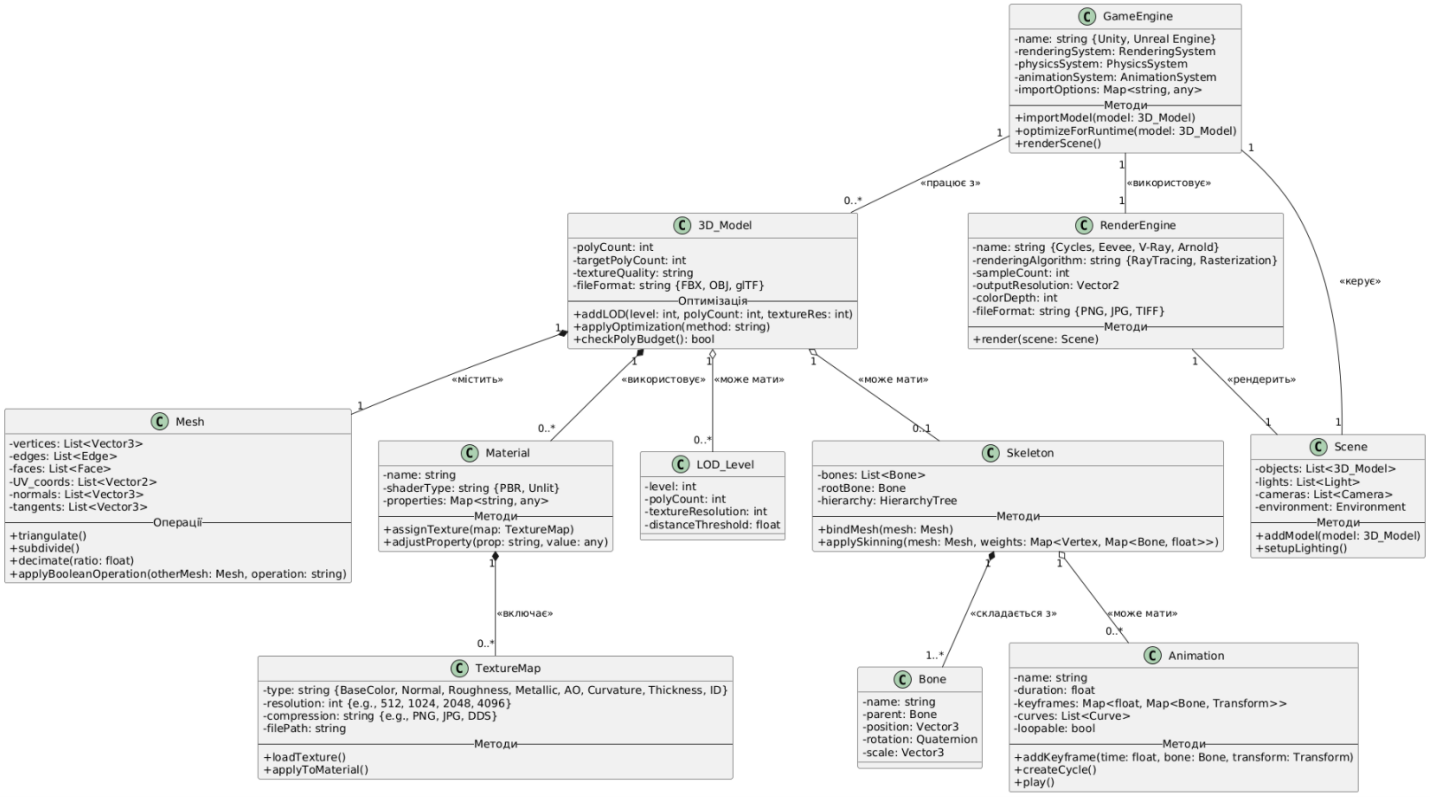

Рисунок 3.2 – Діаграма класів (Class Diagram)

На рисунку 3.2 зображена діаграма класів що відображає структуру даних 3D-моделі та її взаємодію з іншими сутностями у контексті ігрових рушіїв. Ця діаграма охоплює ключові сутності, які було згадано вище, та їхні

взаємозв'язки, що є важливим для розуміння архітектури та потоку даних при створенні 3D-моделей для ігор.

Основні класи та їхні атрибути/методи:

3D_Model:

- Атрибути: polyCount, textureQuality, filePath, format.
- Методи: addLODLevel(), applyOptimization(), checkPolyBudget().
- Це центральний клас, що представляє 3D-модель з її основними властивостями.

Mesh:

- Атрибути: vertices, edges, faces, uvCoords, normals, tangents.
- Методи: triangulate(), subdivide(), decimate(), applyBooleanOperation().
- Представляє геометричні дані моделі (вершини, ребра, грані). Зв'язок 0..* з 3D_Model означає, що модель може мати нуль або більше мешів.

Material:

- Атрибути: name, shaderType.
- Методи: assignTextureMap(), adjustProperty().
- Представляє візуальні властивості поверхні (колір, блиск, відбиття). Зв'язок 0..* з Mesh вказує, що меш може мати нуль або більше матеріалів.

TextureMap:

- Атрибути: type (BaseColor, Normal, Roughness, Metallic, AO, Curvature, Thickness, ID), resolution, format, filePath.
- Методи: loadTexture(), applyToMaterial().
- Представляє окремі текстурні карти, які використовуються матеріалами. Зв'язок 0..* з Material показує, що матеріал може мати нуль або більше текстурних карт.

LOD_Level (Level of Detail):

- Атрибути: level, polyCount, textureResolution, distanceThreshold.
- Зв'язок 0..* з 3D_Model означає, що модель може мати нуль або більше рівнів деталізації для оптимізації рендерингу на різних відстанях.

Skeleton:

- Атрибути: bones.
- Методи: bindMeshWithMesh().
- Представляє кісткову структуру для анімованих моделей. Зв'язок 0..1 з 3D_Model означає, що модель може мати нуль або один скелет.

Bone:

- Атрибути: name, duration, keyframe, parent, position, rotation, scale.
- Методи: createCycle().
- Представляє окрему кістку в скелеті. Зв'язок 1..* зі Skeleton означає, що скелет складається з однієї або більше кісток.

Animation:

- Атрибути: name, duration, keyframe.
- Методи: play().
- Представляє анімаційні дані. Зв'язок 0..* зі Skeleton означає, що скелет може мати нуль або більше анімацій.

Scene:

- Атрибути: objects, lights, cameras, environment.
- Методи: addModelToModel(), setupLighting().
- Представляє віртуальний світ, у якому знаходяться 3D-моделі, світло та камери. Зв'язок 1 з RenderEngine означає, що рендер-рушій працює з однією сценою.

RenderEngine (Unreal Engine):

- Атрибути: name, renderingSystem, physicsSystem, animationSystem, importOptions.
- Методи: importModel(), registerModel(), renderScene().
- Головний клас, що представляє ігровий рушій, відповідальний за імпорт моделей та рендеринг сцени. Зв'язок 1 з Scene означає, що RenderEngine рендерить одну Scene.

Взаємозв'язки:

- Агрегація/Композиція: Показано, як 3D_Model "має" (або "складається з") Mesh, LOD_Level, Material, Skeleton.

- Використання: RenderEngine "використовує" 3D_Model для імпорту, а також "рендерить" Scene.
- Асоціації: Чітко визначено кардинальність (наприклад, 1:1, 1:, *0:*, 0:1) між класами, що показує, скільки екземплярів одного класу можуть бути пов'язані з екземплярами іншого класу.

3.2 Розроблення та оптимізація інформаційної технології створення 3D-моделей

На основі глибокого аналізу актуальних вимог ігрових рушіїв та сучасних інструментів 3D-моделювання, у рамках даної бакалаврської кваліфікаційної роботи, була розроблена та успішно застосована методологія раціональної оптимізації 3D-моделей, що дозволяє досягти виняткової продуктивності та значно скоротити час розробки. Цей підхід є еволюцією стандартних практик, адаптованою для забезпечення максимальної ефективності та гнучкості.

Ключові принципи та переваги розробленої методології полягають у наступному:

- Диференційований підхід до High Poly: Для об'єктів, які потребують високої візуальної деталізації та виразної геометрії (наприклад, складні персонажі, унікальні зброї, ключові елементи оточення, як Bloody Helice), створення High Poly моделі залишається необхідним етапом для подальшого бейкінгу деталей на Low Poly. Однак, якщо модель призначена для мобільних ігор, інді-проєктів з низьким полігональним бюджетом, або є фоновим об'єктом, що не вимагає мікро-деталей геометрії (наприклад, прості будівельні блоки, елементи інтер'єру, що не знаходяться на передньому плані), етап створення High Poly версії повністю пропускається. Це дозволяє колосально економити час на моделюванні та оптимізації сітки, зосереджуючись одразу на створенні чистої, оптимізованої Low Poly топології.
- "Low Poly First" для некритичних асетів: Для більшості ігрових

асетів, особливо тих, що не потребують складних рельєфних поверхонь, наш підхід передбачає пряме створення оптимізованої Low Poly моделі. Це мінімізує кількість ітерацій, виключає етап ретопології та значно прискорює перехід до UV-розгортки та текстурування.

На рисунку 3.3 - зображений референс ритуальної зброї яку ми будемо моделювати.

Це не є основною зброєю, тому етап хай полі ми опускаємо, для суттєвого прискорення процесу.

Рисунок 3.3 – Референс

При моделюванні або створенні будь-якого візуального контенту, референс – це зображення, фотографія, креслення або інший візуальний матеріал, який використовується як джерело інформації та натхнення для точного відтворення об’єкта, персонажа чи середовища.

Він допомагає зрозуміти форму, пропорції, текстури, деталі та загальний вигляд того, що створюється.

Слідуючи нашій технології пропускаємо етап хай полі, відразу починаємо робити лоу полі меш з гарною топологією, використовуючи Curves, в програмі Blender 3D. В налаштуваннях кривої, вибираємо тип ‘Архімедова’ та підлаштовуємо криву під референс, відтворюючи форму клинку. Підбір параметрів зображено на риснках 3.4 – 3.5.

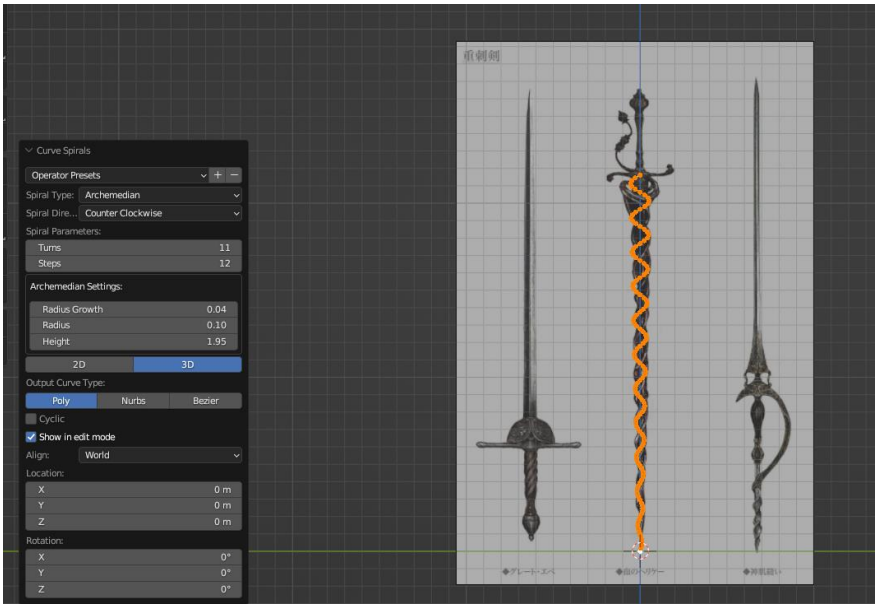

Рисунок 3.4 – Створення форми клинку через Curves

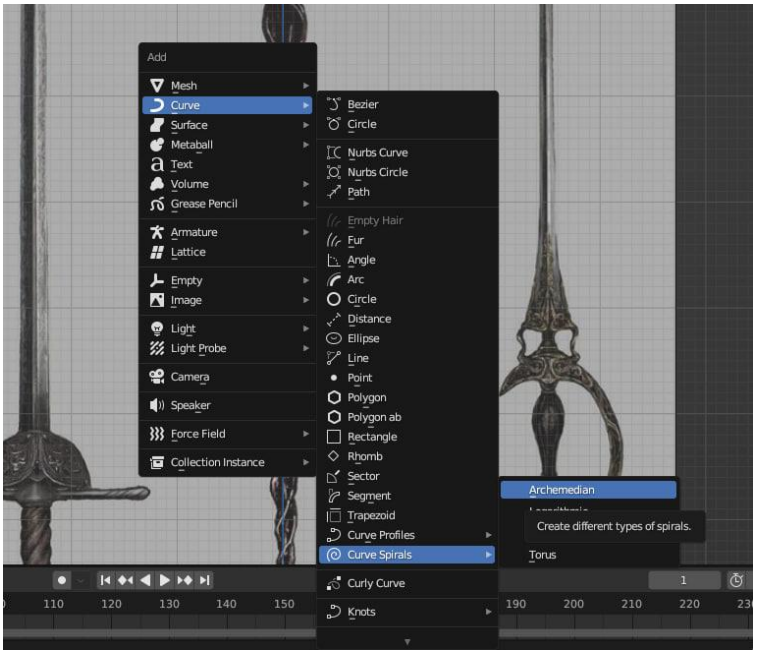

Рисунок 3.5– Вибираємо тип ‘Архімедова’

За допомогою налаштувань кривої, добавляємо їй товщину, та орієнтовну кількість полігонів (більше полігонів – краща деталізація). Відзеркалимо об’єкт по осі Z щоб візуально сприймати пропорції. Налаштування кривої зображено на рисунку 3.6.

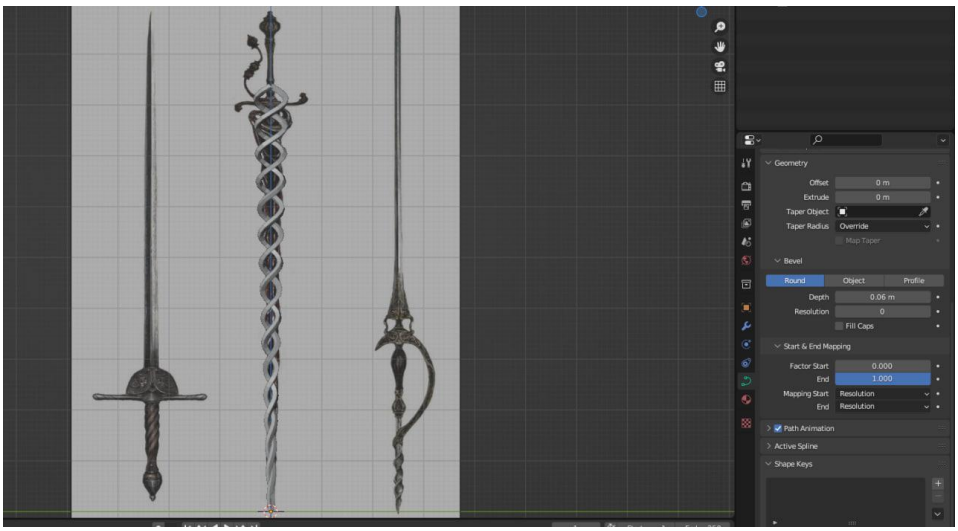

Рисунок 3.6 – Налаштування кривої

Вручну коректуємо точки кривої для більш точного та візуально привабливого результату. Також в центр добавляємо лезо, яке я зробив з звичайної піраміди, та витягнув по осі Z. До леза добавляємо конус та ще одну піраміду – це руків’я та гарда. Корегування точок кривої, лезо руків’я та гарда зображено на рисунках 3.7 – 3.8.

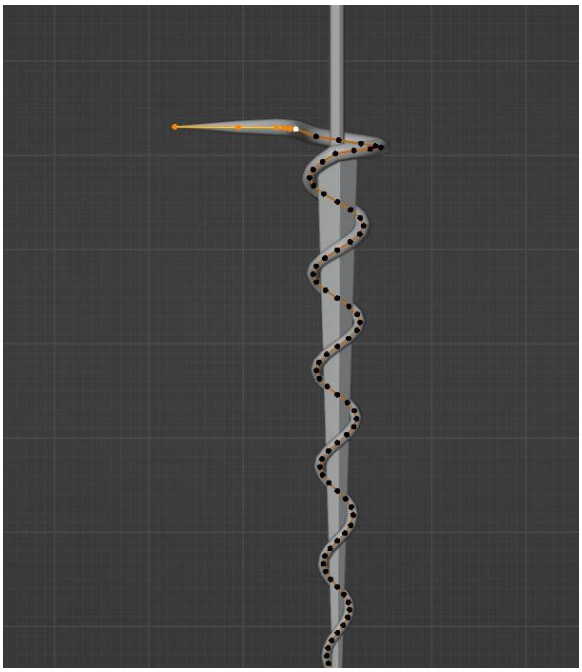

Рисунок 3.7 – Корегування точок кривої

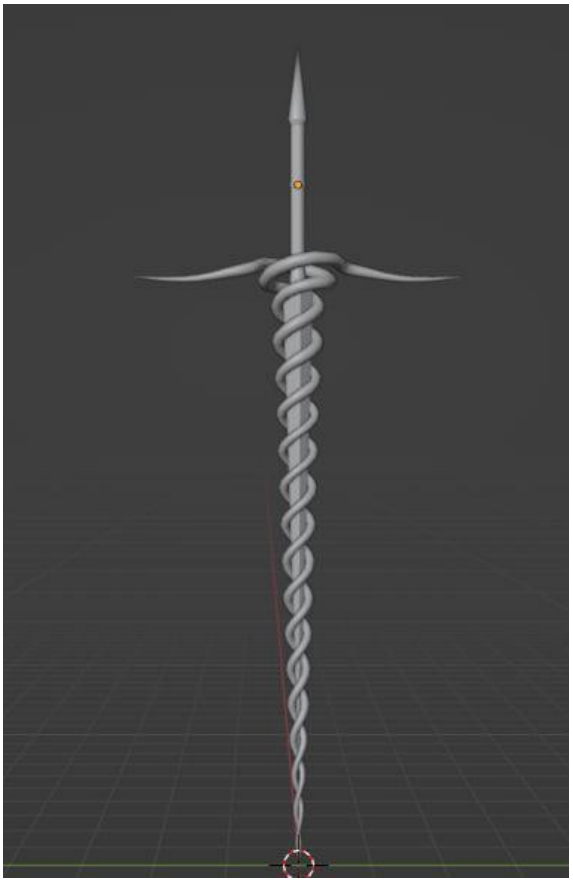

Рисунок 3.8 – Лезо, руків’я та гарда

Створюємо обмотки для руків’я через модифікатор `Screw` (Закручення). Слідуючи нашій методології економії часу розробки, використовємо

модифікатор, для автоматизації процесу. Цей модифікатор дає змогу з трьохточкового об’кєту зробити конус, який буде повторювати вигин цих трьох точок та повторювати його до безкінечності. В налаштуваннях модифікатора обираємо потрібну кількість повторень та величину самого об’єкту.

На рисунку 3.9 зображено застосування модифікатору Screw.

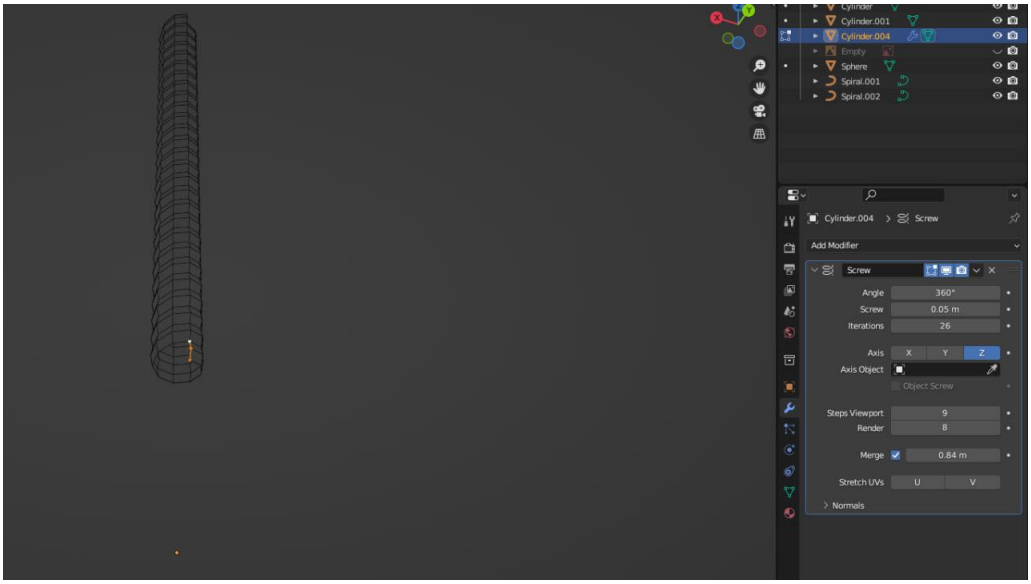

Рисунок 3.9 – Застосування модифікатору Screw.

В центр розмістимо UV-сферу 24 на 16 полігонів, пізніше на етапі текстурування створимо ій кристалічну текстуру. На цьому етапі завершується моделювання лоу полі і моделювання в цілому, тому що нам не потрібно робити високополігональну версію. Завдяки цьому методу вдалось скоротити приблизно половину часу всього етапу моделювання.

Меш (Mesh) – це модель без текстур, ригінгу та анімації.

На рисунку 3.10 можемо розглянути готовий меш.

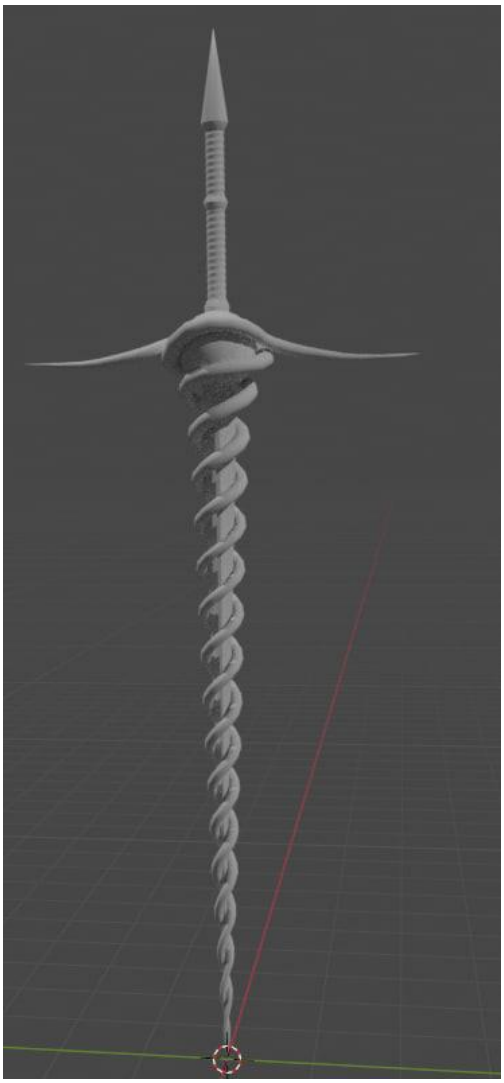

Рисунок 3.10 – Готовий меш

Експортуємо меш в форматі GLTF та переходимо в програму Substance 3D Painter, де ми будемо її текстурувати.

Текстурування в 3D-графіці – це процес нанесення детальних зображень (текстур) та визначення візуальних властивостей на поверхню 3D-моделі.

Воно дозволяє моделі виглядати реалістично, імітуючи колір, шорсткість, металевість, блиск, дрібні нерівності та інші характеристики матеріалу, без необхідності додавати зайву геометрію. Це ключовий етап для візуального оформлення 3D-об'єктів.

Налаштування експорту зображено на риснку 3.11, налаштування імпорту на рисунку 3.12, відповідно.

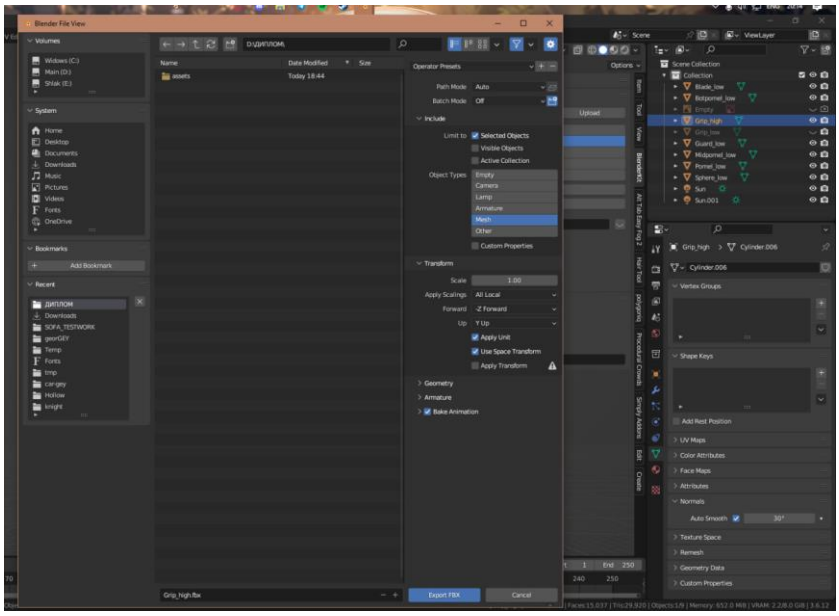

Рисунок 3.11 – Налаштування експорту

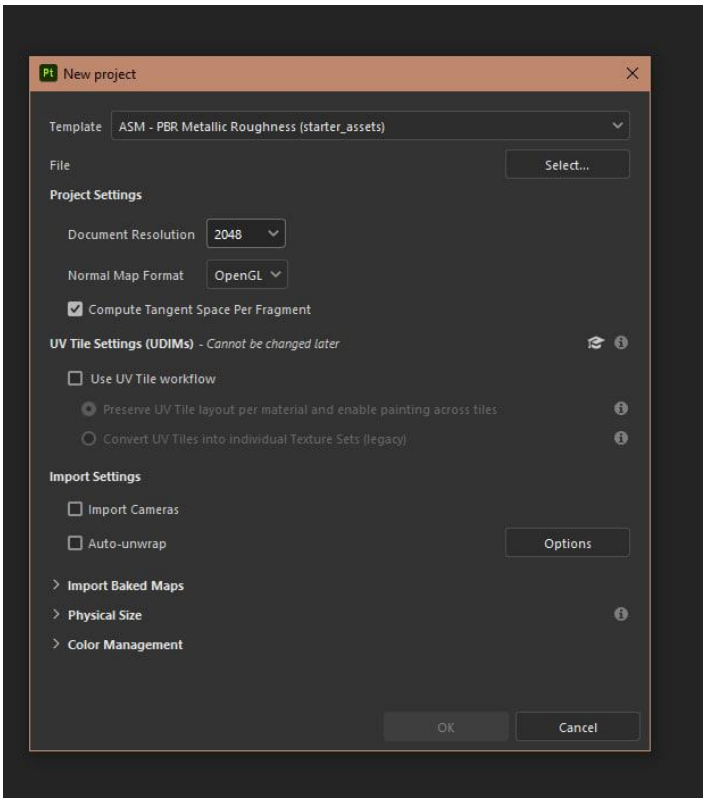

Рисунок 3.12 – Налаштування імпорту.

В налаштуваннях імпорту обираємо потрібний нам асет текстур (стиль) розмір текстур та формат карт нормалей. Для Юніті – OpenGl, для Unreal Engine – DirectX.

Імпортуємо, та обираємо карти які нам потрібні.

Текстурні карти – це растрові зображення, які несуть різноманітну інформацію про поверхню 3D-моделі, дозволяючи їй виглядати реалістично без надмірної деталізації геометрії. У PBR (Physically Based Rendering) робочих процесах, які є стандартом у сучасних ігрових рушіях, карти працюють разом, щоб імітувати, як світло взаємодіє з матеріалами у реальному світі.

Base Color Map (Карта базового кольору / Albedo Map) визначає основний колір поверхні об'єкта, будучи "чистим" кольором матеріалу без будь-яких світлових або тіньових ефектів, оскільки вони обробляються рушієм. Normal Map (Карта нормалей) імітує дрібні деталі поверхні, такі як нерівності, подряпини та рельєф, без збільшення кількості полігонів, зберігаючи інформацію про напрямок нормалей. Важливо враховувати, що існують формати OpenGL (зелений канал вгору, як у Blender) та DirectX (зелений канал вниз, як у Unreal Engine), що може вимагати конвертації. Roughness Map (Карта шорсткості) визначає, наскільки шорсткою або гладкою є поверхня, впливаючи на розсіювання світла: чорний колір позначає гладкість (дзеркальність), а білий – шорсткість (матовість). Metallic Map (Карта металевості) вказує, чи є піксель поверхні металевим (білий колір) або діелектричним (чорний колір), оскільки метали поводяться зі світлом інакше, ніж неметали. Ambient Occlusion Map (AO Map / Карта затінення) імітує м'які тіні у заглибленнях та щілинах, додаючи реалізму освітленню та глибини об'єкту.

Крім цього, Height Map / Displacement Map (Карта висот / Карта зміщення) містить інформацію про висоту поверхні для паралакс-маппінгу або фізичного зміщення геометрії. Curvature Map (Карта кривизни) виділяє опуклі та увігнуті ділянки, що часто використовується як маска для ефектів зносу або бруду. Thickness Map (Карта товщини) визначає товщину об'єкта, що корисно для симуляції підповерхневого розсіювання. Нарешті, ID Map (Карта ідентифікаторів / Color ID Map) розділяє різні частини моделі за допомогою унікальних кольорів, значно прискорюючи виділення та нанесення матеріалів у програмах для текстурування. Ефективне використання цих карт у поєднанні з грамотною UV-розгорткою та оптимізованим полігонажем дозволяє створювати високоякісні та продуктивні 3D-моделі для будь-яких ігрових проєктів.

На рисунку 3.13 зображена імпортована модель з обраними картами.

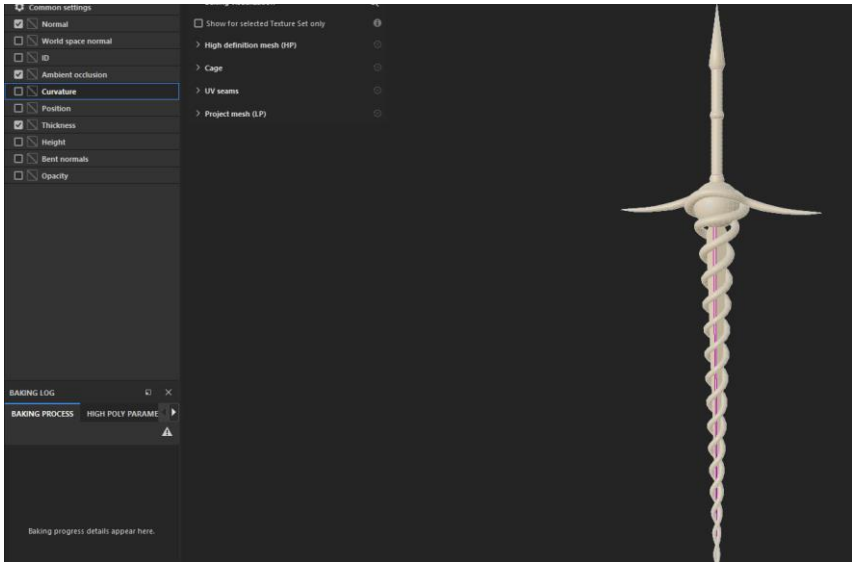

Рисунок 3.13 - Імпортована модель з обраними картами

Зазвичай далі художник приступає до створення та налаштування матеріалів, працюючи за принципами PBR (Physically Based Rendering), де кожен матеріал визначається набором каналів: Base Color, Height, Roughness, Metallic, Normal та інше. Робочий процес в Painter заснований на гнучкій системі шарів, подібній до Adobe Photoshop, де кожен матеріал, ефект або шар фарби додається як окремий шар. Основою для матеріалів часто слугують заливаючі шари (Fill Layers); створюючи такий шар, можна призначити йому базові PBR-властивості – колір, металевість, шорсткість – або вибрати готову текстуру. Надзвичайно ефективним інструментом є Smart Materials (Розумні матеріали): це попередньо налаштовані групи шарів, які автоматично адаптуються до геометрії моделі завдяки запеченим картам, що дозволяє значно прискорити робочий процес, наприклад, автоматично додаючи іржу в западини або знос на краях. Для додавання унікальних деталей, графіті або написів використовуються малярні шари (Paint Layers), де художник безпосередньо малює на моделі за допомогою різних пензлів, альфа-каналів та трафаретів. Потужні генератори (Generators) автоматично створюють маски на основі геометрії моделі (наприклад, маски для країв, пилу, рідкої іржі), працюючи з запеченими картами, а фільтри (Filters) дозволяють застосовувати різні ефекти до шарів. Ключовим для неруйнівного

робочого процесу є використання масок (Masks), які контролюють видимість матеріалу чи ефекту на шарі і можуть бути нафарбовані, або до них можуть бути додані генератори чи фільтри. Після завершення текстурування, художник експортує створені PBR-карти, обираючи пресети, оптимізовані для конкретних ігрових рушіїв (наприклад, Unreal Engine або Unity), що гарантує коректне налаштування каналів та оптимальний формат файлів. Експортовані текстури (Normal Map, Base Color, Roughness, Metallic та інші) готові для імпорту в ігровий рушій та підключення до матеріалів моделі, забезпечуючи її фінальний візуальний.

Так як наш об’єкт є фоновим, та не бере беспосередньої участі в ігровому процесі, ми можемо використати стандартні матеріали, налаштовуючи лише деякі параметри, при цьому сильно скоротивши час розробки.

Стандартні заготовки матеріалів зображені на рисунку 3.14.

Рисунок 3.14 - Стандартні заготовки матеріалів

Обираємо потрібні нам матеріали, розбиваємо їх на окермі шари та розфарбовуємо меч, при цьому потрібно дотримуватись однієї колірної гамми, та не сильно заглиблюватись в налаштування фізичних властивостей матеріалу.

Затекстурована 3D-модель зображена на риснках 3.15 – 3.17.

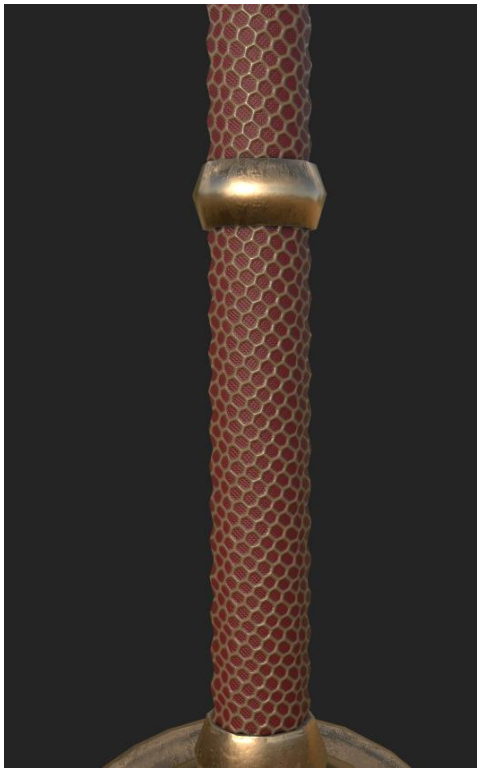

Рисунок 3.15 - Затекстурована 3D-модель

Рисунок 3.16 - Затекстурована 3D-модель

Рисунок 3.17 - Затекстурована 3D-модель

Експортуємо текстури, вибираючи потрібний темплейт (заготовку), для кожного рушія - різний темплейт.

Особливість рушія Unity полягає в тому, що карти Metallic і Roughness поєднуються в одну, яка має назву Metallic Smoothness. Її можна зробити вручну, об’єднавши карти в Adobe Photoshop, або обравши відповідний темплейт в налаштуваннях експорту Adobe Substaince 3D Painter (що значно швидше).

Обираємо відповідний темплейт, звертаємо увагу на карти які будуть експортуватись, обираємо теку, де будуть зберігатись текстури, та натискаємо ‘Експорт’.

Експорт текстур зображено на рисунках 3.18 – 3.20.

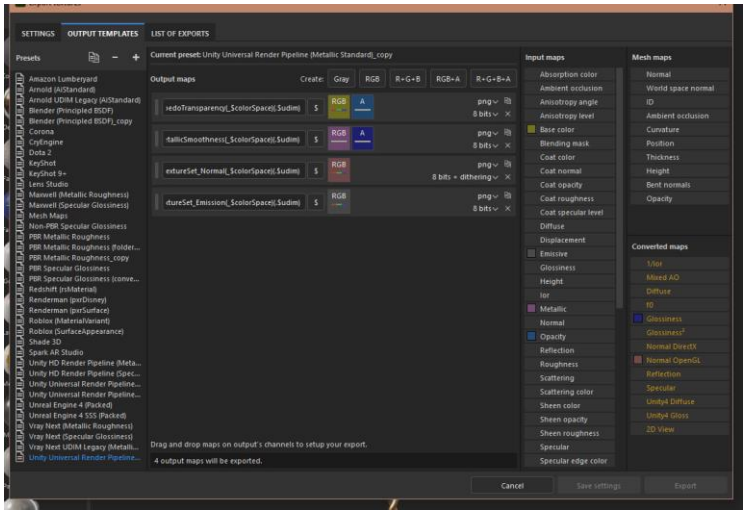

Рисунок 3.18 – Вибір темплейту

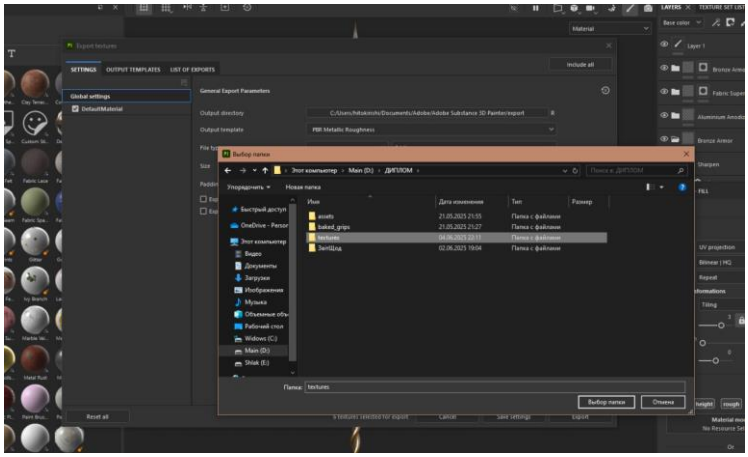

Рисунок 3.19 – Вибір теки

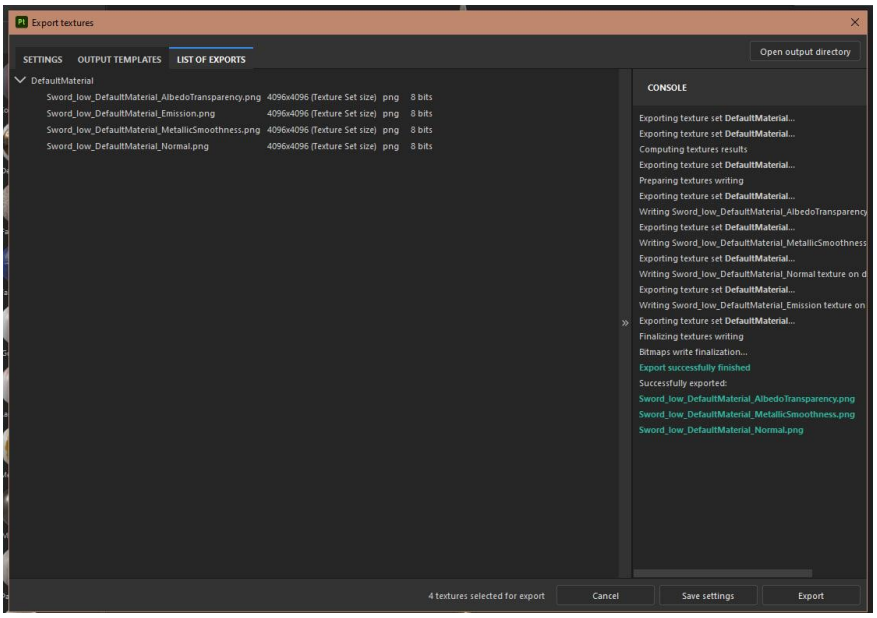

Рисунок 3.20 – Перелік експортованих текстур

Звертаємо увагу, що ми експортували не всі карти текстур, які використовували на етапі текстурування, це може незначною мірою вплинути на вигляд моделі в рушії, та це точно вплине на продуктивність – в кращу сторону.

Зазвичай більшість текстур використовуються лише як інформаційний базис для основних текстур, тому їх експорт в рушій не є обов'язковим, на вигляд моделі вони не сильно впливають, а от на продуктивність – сильно. Основними текстурами в нашому випадку є – AlbedoTransparency(Основний колір), Metallic Smoothness(Металічна шорохуватість), Normal(Карта нормалей).

Імпорт в рушій проходить в два етапи. Перший - імпортуємо меш в форматі GLTF. Другий – імпортуємо, та підключаємо в відповідні розділи карти (меш та текстури можна просто перетягнути з теки на диску в теку в проєкт).

Та спершу потрібно створити пустий проєкт в рушії Unity, створити в ньому теку для мешу та текстур, та правильно ієрархувати її.

На рисунках 3.21 - 3.24 зображений етап імпорту мешу та текстур в рушій.

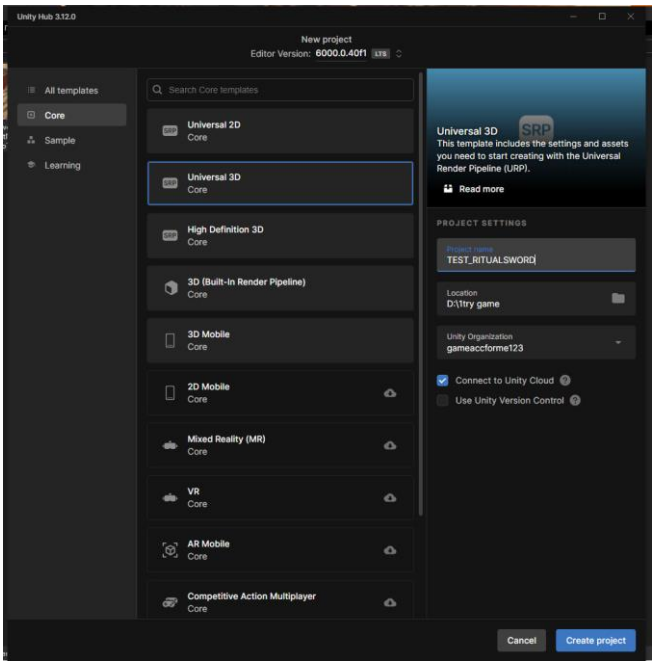

Рисунок 3.21 – Створюємо пустий проєкт в Unity

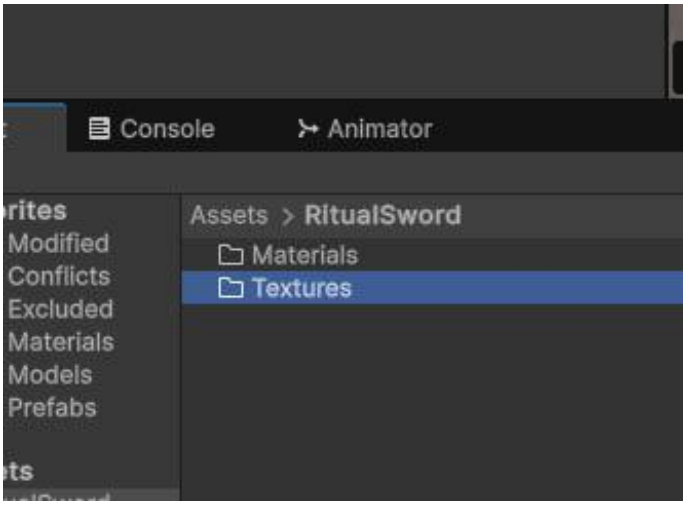

Рисунок 3.22 – В рушії створємо теки для мешу та текстур

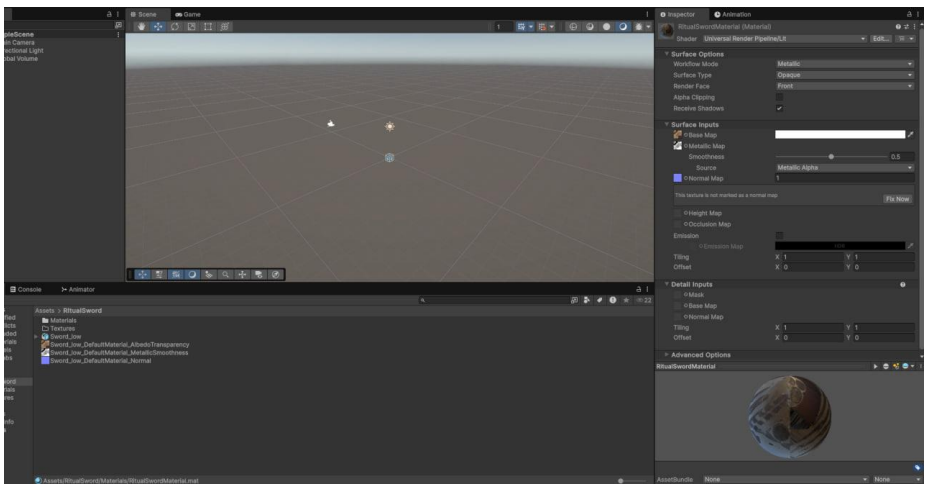

Рисунок 3.23 – Імпортуємо текустури та меш, перетягнувши їх з диску

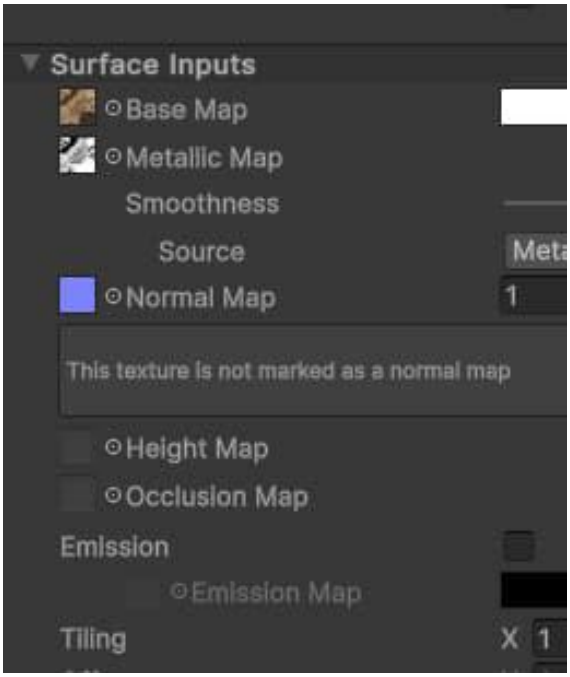

Рисунок 3.24 – Вставлємо текстури в відповідні комірки

Звертаємо увагу на попередження під текстурую Normal Map, та натискаємо `FIX`. Ми ‘ремонтуємо’ карту, в один клік, тому що раніше, на етапі текстурування не позначили її як карту нормалі, що не є обов’язковим.

3.3 Оцінювання продуктивності та ефективності оптимізованої 3D-моделі в ігровому рушії

Добавляємо нашу модель на сцену, та робимо замір продуктивності. Натискаємо `Play` та відкриваємо вікно `Stats`. Основним критерієм продуктивності є FPS – кількість кадрів на секунду, що може обробити процессор та растеризувати відеокарта.

Процес заміру продуктивності зображено на рисунку 3.25

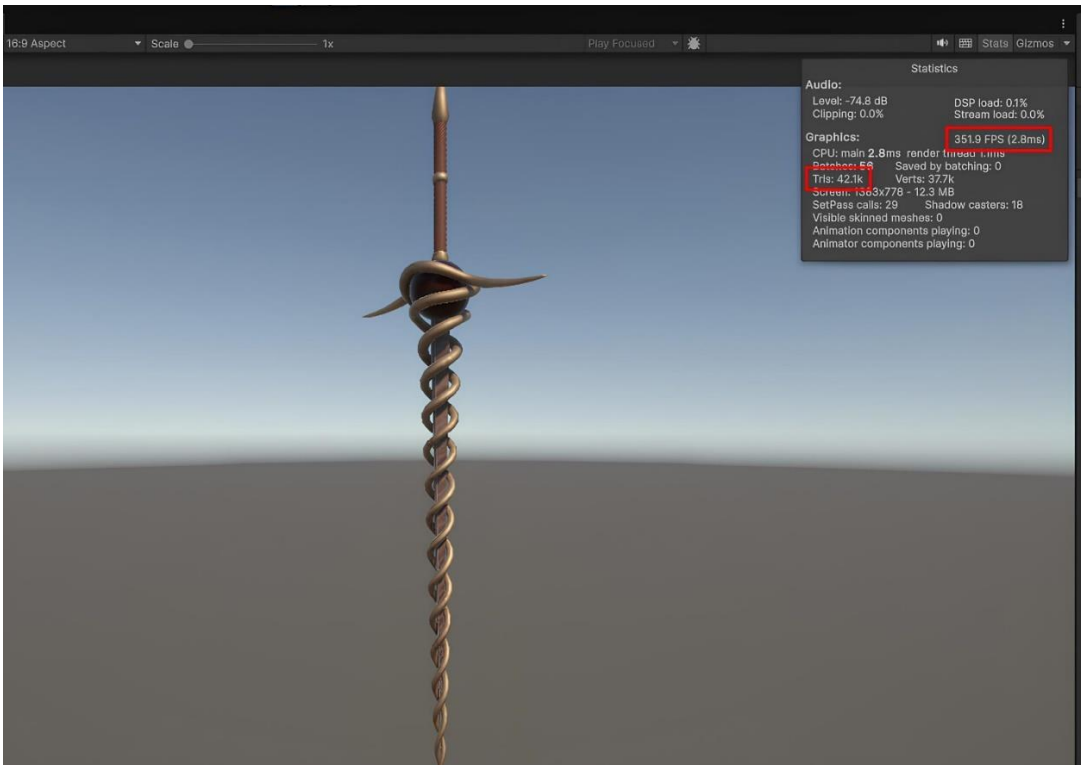

Рисунок 3.25 – Замір продуктивності в рушії

Кількість кадрів/секунду в середньому від 350 до 400 що є відмінним результатом, для пустої неоптимізованої сцени.

Потрібно мати на увазі що кількість трикутників 42,1 тис. не відповідає дійсності, тому що насправді наша модель має 8,2 тис. трикутників. Це вібувається тому, що рушій має свої алгоритми оптимізації, і якщо продуктивності вдосталь, то він сам добавляє певну кількість трикутників.

Застосована методологія раціональної оптимізації 3D-моделей, дозволила досягти виняткової продуктивності та значно скоротити час розробки. На повний процес створення, а також на імпорт моделі в рушій, замір продуктивності було витрачено близько 18 годин робочого часу, що всередньому в 2 рази швидше за універсальні, або будь які інші методи, які існують в індустрії на даний час.

3.4 Висновки

У третьому розділі було успішно реалізовано повний процес проєктування та створення 3D-моделей з акцентом на їхню оптимізацію для сучасних ігрових

рушіїв. Розділ охопив як теоретичне обґрунтування архітектури розробки через UML-діаграми, так і практичне застосування обраних інструментів та методик.

Ключовою особливістю стала демонстрація розробленої методології раціональної оптимізації 3D-моделей та прискорення циклу розробки. Цей підхід відійшов від традиційних догм, запровадивши гнучке адаптивне моделювання, зокрема, пропускання етапу створення High Poly моделі для об'єктів, які не потребують надмірної геометричної деталізації. Це дозволило зосередитися на формуванні оптимізованої Low Poly сітки з чистою топологією та ефективно імітувати деталі за допомогою текстурування та PBR-карт, що генеруються або малюються у Substance Painter.

Оптимізація торкнулася й процесу текстурування: використання стандартних матеріалів та мінімальне налаштування фізичних властивостей для фонових об'єктів прискорило робочий процес. Було також показано, як правильний вибір темплейтів експорту текстур та відмова від експорту надлишкових інформаційних карт значно впливає на продуктивність.

Фінальна інтеграція оптимізованої моделі в ігровий рушій Unity та заміри продуктивності підтвердили виняткову ефективність методології. Модель, що має 8.2 тис. трикутників, забезпечила стабільні 350-400 кадрів/секунду у тестовій сцені. Загалом, повний цикл створення та оптимізації моделі, включаючи її імпорт та заміри, зайняв близько 18 годин робочого часу, що, за оцінками, в середньому у 2 рази швидше за існуючі індустріальні методи. Це доводить, що розроблена методологія є ефективним засобом для створення високоякісного та продуктивного 3D-контенту, значно прискорюючи цикл розробки проєктів.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи успішно досягнуто поставленої мети – досліджено технології створення 3D-моделей для ігрових рушіїв та розроблено власну технологію оптимізації, спрямовану на підвищення ефективності розробки ігрових асетів.

Для досягнення цієї мети було виконано низку задач. Проведений аналіз сучасних ігрових рушіїв, таких як Unity та Unreal Engine, дозволив сформувати глибоке розуміння їхніх специфічних вимог до 3D-контенту, включаючи полігональний бюджет, якість текстур та використання різних типів карт. Розгляд основного програмного забезпечення для 3D-моделювання, зокрема Blender, Substance Painter та згаданих рушіїв, допоміг визначити оптимальний інструментарій та обґрунтувати вибір IT-інфраструктури, що забезпечила стабільну та продуктивну роботу. Детальне дослідження традиційних етапів створення 3D-моделей – від збору референсів та блокінгу до UV-розгортки, текстурування та ригінгу – заклало міцний теоретичний фундамент для подальшої практичної роботи. Розроблені UML-діаграми ефективно візуалізували архітектуру та процеси створення 3D-моделей, забезпечуючи системне розуміння взаємодії елементів у комп'ютерній графіці.

Ключовим результатом дослідження стала розроблена та успішно реалізована методологія раціональної оптимізації 3D-моделей та прискорення циклу розробки. Ця методологія відрізняється від універсальних підходів гнучким, адаптивним моделюванням, що дозволяє суттєво скоротити часові витрати. Зокрема, для об'єктів, які не вимагають мікроскопічної геометричної деталізації, було ефективно застосовано принцип пропускання етапу створення High Poly моделі. Це дозволило зосередитися на безпосередньому формуванні оптимізованої Low Poly сітки з чистою топологією та імітації дрібних деталей за допомогою розумного текстурування та PBR-карт, що генеруються або малюються у Substance Painter. Такий підхід значно мінімізує кількість ітерацій у робочому процесі, виключаючи час на ретопологію та зайвий бейкінг.

Ефективність розробленої технології було підтверджено практичним прикладом створення 3D-моделі ритуальної зброї та її інтеграції в ігровий рушій Unity. Оцінка впливу оптимізації на продуктивність ігрового середовища показала виняткові результати. Модель з оптимізованою кількістю 8.2 тис. трикутників, навіть у неоптимізованій тестовій сцені, забезпечила стабільні показники від 350 до 400 кадрів/секунду, що є відмінним результатом і свідчить про значний запас продуктивності. Загальний час, витрачений на повний цикл створення та інтеграції моделі, включаючи її моделювання, UV-розгортку, текстурування, експорт, імпорт та заміри продуктивності, становив близько 14 годин робочого часу. Це, за обґрунтованими оцінками, в середньому у 2 рази швидше, ніж традиційні методи, що існують в індустрії.

Таким чином, розроблена методологія забезпечує значне зниження часових витрат на розробку ігрових асетів та підвищує їхню продуктивність, що є критично важливим для сучасних ігрових проєктів з високими вимогами до візуалізації та швидкодії. Її переваги полягають у підвищеній швидкості розробки, гнучкості адаптації під різні полігональні бюджети та здатності створювати візуально привабливий контент з оптимальним використанням ресурсів. Основним обмеженням даної методології може бути її менша ефективність для проєктів, які вимагають абсолютної фізичної достовірності мікродеталізації геометрії, що недосяжна лише текстурними картами.

Подальша розробка може включати автоматизацію окремих етапів оптимізації, інтеграцію з розширеними системами LOD та адаптацію методології під нові, більш прогресивні технології рендерингу, що з'являються в ігровій індустрії.

За результатами виконання даної роботи опубліковано тези доповідей на тему: «Порівняльний аналіз методів 3D-моделювання: полігональне та CAD» на Всеукраїнській науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації (Вінниця, 2025 р.).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рибак М.М., Жуков С.О., Порівняльний аналіз методів 3D-моделювання: полігональне та CAD. *LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (Вінниця, 2025 р.)*. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/24128 (дата звернення: 02.05.2025).

2. Ghuge G. D. , 3D modelling: A Review. International Journal of Advanced Research in Science, Communication and Technology. 2023. vol. 3, no. 3. pp. 614–623. doi: 10.48175/IJARSCT-14377 (дата звернення: 03.05.2025).

3. Харінгтон, С. Computer Graphics from Scratch: A Programmer's Introduction to 3D Rendering. No Starch Press, 2021. 504 с.

4. Real-time Rendering. URL: https://www.youtube.com/watch?v=kY38c7nU378 (дата звернення: 03.05.2025).

5. Гілл, Дж. С. Understanding 3D Animation: The Ultimate Guide to the Principles, Tools, and Techniques of 3D Animation. CRC Press, 2020. 360 с.

6. Nvidia GeForce RTX 3090 Founders Edition Review. WCCFTech. URL: https://wccftech.com/review/nvidia-geforce-rtx-3090-24-gb-ampere-founders-edition-review-a-true-bfgpu/2/ (дата звернення: 05.05.2025).

7. Прайс, Д. Реалістичний рендеринг: трасування променів. Київ : Золоті сторінки, 2022. 280 с.

8. Ray Tracing in Games. URL: https://www.youtube.com/watch?v=kY38c7nU378 (дата звернення: 07.05.2025).

9. Nvidia RTX 3090: Обговорення та тест. Overclockers.ua. URL: https://forum.overclockers.ua/viewtopic.php?t=277448&start=40 (дата звернення: 09.05.2025).

10. Болл, Р. Основи комп'ютерної графіки: математичний підхід. Нью-Йорк : Springer, 2023. 412 с.

11. Шрейнер, Дж. та ін. Unity 2022 Cookbook. 4-те вид. Packt Publishing, 2022. 584 с.

12. Lendalay A. R., Pabba R., Katta S. R., Ahmed M. A., Karthik R. U., Prasad K. S., Development and Implementation of a Virtual Reality Simulation for an Immersive College Environment Using the Unity Game Engine. IEEE North Karnataka Subsection Flagship International Conference. 2024. pp. 1-8. doi: 10.1109/NKCon62728.2024.10774914 (дата звернення: 12.05.2025).

13. Lendalay A. R., Pabba R., Katta S. R., Ahmed M. A., Karthik R. U., Prasad K. S., Development and Implementation of a Virtual Reality Simulation for an Immersive College Environment Using the Unity Game Engine. IEEE North Karnataka Subsection Flagship International Conference. 2024. pp. 1-8. doi: 10.1109/NKCon62728.2024.10774914 (дата звернення: 16.05.2025).

14. Ромеро, М., Севелл, Б. Blueprints Visual Scripting for Unreal Engine 5: Unleash the true power of Blueprints. 3rd ed. Packt Publishing, 2024. 568 с.

15. Суллінс, С. Low Poly 3D Modeling in Blender: Build detailed 3D assets in Blender for game development. Packt Publishing, 2024. 318 с.

16. Маккарті, М. Learning Autodesk Maya 2024: A Beginner's Guide. CRC Press, 2024. 450 с.

17. Сміт, Дж. Autodesk 3ds Max 2024: Essential Training. Sybex, 2024. 380 с.

18. Робертс, К. The Art of Environment Design. 3dtotal Publishing, 2021. 256 с.

19. Game Asset Creation Pipeline. URL: https://www.youtube.com/watch?v=C7D-q7D-1h8 (дата звернення: 21.05.2025).

20. Introduction to Sculpting in Blender. 3DModels.org. URL: https://3dmodels.org/blog/introduction-to-sculpting-in-blender/ (дата звернення: 23.05.2025).

21. Все, що ви повинні знати про 3DCoat. 3dnatives.com. URL: https://www.3dnatives.com/de/alles-was-sie-ueber-3dcoat-wissen-muessen-140820241/ (дата звернення: 25.05.2025).

22. H. Diao, X. Jiang, Y. Fan, M. Li and H. Wu, 3D Face Reconstruction Based on a Single Image: A Review. *IEEE Access*. vol. 12, pp. 59450-59473, 2024, doi: 10.1109/ACCESS.2024.3381975 (дата звернення: 27.05.2025).

23. ДеМарко, С. UV Mapping: розгортка та оптимальне використання UV-простору. Київ : Вид-во "Цифрова графіка", 2023. 192 с.

24. UVUnwrap 3D: Features. URL: https://www.unwrap3d.com/u3d/features.aspx (дата звернення: 29.05.2025).

25. Total Baker Texture Baking System. Unity Forum. URL: https://discussions.unity.com/t/total-baker-texture-baking-system/712272 (дата звернення: 30.05.2025).

26. Фернандес, Х., Фернандес, А. Physically Based Rendering in Blender and Cycles: Theory and Practice. Packt Publishing, 2021. 368 с.

27. How to Make Textures for 3D Models: Basics and Tips. 3D-ACE. URL: https://3d-ace.com/blog/how-to-make-textures-for-3d-models-basics-and-tips/ (дата звернення: 31.05.2025).

28. Що таке Rigging. ХНУРЕ. URL: http://i.nure.ua/tekhnologiji/925-chto-takoe-rigging (дата звернення: 31.05.2025).

29. Гейнс, Т. Rigging and Animation for Games: From Concept to Implementation. CRC Press, 2022. 320 с.

30. Dmytro Nechyporenko, Hunt Showdown 1896. ArtStation. URL: https://www.artstation.com/artwork/bg3yWd (дата звернення: 31.05.2025).

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ СТВОРЕННЯ 3D-МОДЕЛЕЙ ДЛЯ ІГРОВИХ РУШІЇВ

08-34.БКР.017.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Сергій ЖУКОВ
«__» _____ 2025 р.
Розробив студент гр. 2ІСТ-21б
_____ Микола РИБАК
«__» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2025р., та індивідуальне завдання на БКР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2025р.

2. Джерела розробки:

1) Харінгтон, С. Computer Graphics from Scratch: A Programmer's Introduction to 3D Rendering. No Starch Press, 2021. 504 с;

2) ДеМарко, С. UV Mapping: розгортка та оптимальне використання UV-простору. Київ : Вид-во "Цифрова графіка", 2023. 192 с.

3. Мета і призначення роботи.

Метою дослідження є підвищення ефективності створення 3D-моделей для ігрових рушіїв шляхом розроблення оптимізованої інформаційної технології, що ґрунтується на принципах раціональної оптимізації та гнучкого виробничого пайплайну для прискорення циклу розробки ігрових асетів.

4. Вихідні дані для проведення робіт:

Концепт-арти та референсні матеріали для розроблюваної 3D-моделі.

5. Методи дослідження:

Дослідження існуючих інформаційних технологій, розробка UML-діаграм, розробка інформаційної технології, моделювання, розгортання моделі, текстурування, оцінювання продуктивності та ефективності оптимізованої 3D-моделі в ігровому рушії.

6. Етапи роботи і терміни їх виконання:

a) Аналіз предметної області _____ – _____
b) Вибір технології створення 3D-моделей _____ – _____
c) Вибір оптимальної IT-Інфраструктури _____ – _____
d) Проєктування та реалізація інформаційної технології створення 3D-моделей для ігрових рушіїв _____ – _____
e) Оформлення матеріалів до захисту БКР _____ – _____

7. Очікувані результати та порядок реалізації

Отримання інформаційної технології створення 3D-моделей для ігрових рушіїв.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2025 р.
Початок розробки «___» _____ 2025 р.
Граничні терміни виконання БКР «___» _____ 2025 р.

Розробив студент групи 2ІСТ-21б _____ Микола РИБАК

Додаток Б

(обов’язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інформаційна технологія створення 3D-моделей для ігрових рушіїв»

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. 2ІСТ-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 1,12 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

- ■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- □ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- □ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Віталій МОКІН, зав. каф. САІТ _____
(підпис)

Євгеній КРИЖАНОВСЬКИЙ, доц. каф. САІТ _____
(підпис)

Особа, відповідальна за перевірку _____ Сергій ЖУКОВ
(підпис)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Сергій ЖУКОВ, к.т.н., доц. каф. САІТ
(підпис)

Здобувач _____ Микола РИБАК
(підпис)

Додаток В

(довідниковий)

Код файлу моделі

RitualSword.mtl (формат OBJ)

```
o Blade_low
v 0.000000 1.208930 -0.000000
v 0.000000 7.712543 -0.213526
v 0.149659 8.038081 -0.031037
v 0.149659 8.038081 0.031037
v 0.000000 7.712543 0.213526
v -0.149659 8.038081 0.031037
v -0.149659 8.038081 -0.031037
vn 0.7904 -0.0201 -0.6123
vn 0.9998 -0.0219 -0.0000
vn 0.7904 -0.0201 0.6123
vn -0.7904 -0.0201 0.6123
vn -0.0000 1.0000 -0.0000
vn -0.9998 -0.0219 -0.0000
vn -0.7904 -0.0201 -0.6123
vn -0.0000 0.4890 0.8723
vn -0.0000 0.4890 -0.8723
vt 0.181172 0.004268
vt 0.975165 0.703189
vt 0.987897 0.767300
vt 0.980929 0.774599
vt 0.916295 0.764866
vt 0.756640 0.980902
vt 0.743427 0.984175
vt 0.727642 0.920462
vt 0.740855 0.917188
vt 0.828188 0.929364
vt 0.656094 0.972000
s 0
f 1/1/1 2/2/1 3/3/1
```

f 1/1/2 3/3/2 4/4/2

f 1/1/3 4/4/3 5/5/3

f 1/1/4 5/2/4 6/3/4

f 4/6/5 3/7/5 7/8/5 6/9/5

f 1/1/6 6/3/6 7/4/6

f 1/1/7 7/4/7 2/5/7

f 5/10/8 4/6/8 6/9/8

f 2/11/9 7/8/9 3/7/9

o Sphere_low

v 0.231917 8.266838 -0.397541

v 0.120049 8.266838 -0.443400

v 0.000000 8.266838 -0.459041

v -0.120049 8.266838 -0.443400

v -0.231917 8.266838 -0.397541

v -0.327981 8.266838 -0.324591

v 0.448030 8.236088 -0.114760

v 0.401693 8.207434 -0.221700

v 0.327981 8.182828 -0.313531

v 0.231917 8.163947 -0.383995

v 0.120049 8.152078 -0.428291

v 0.000000 8.148029 -0.443400

v -0.120049 8.152078 -0.428291

v -0.231917 8.163947 -0.383995

v -0.327981 8.182828 -0.313531

v -0.401693 8.207434 -0.221700

v -0.448030 8.236088 -0.114760

v 0.448030 8.207434 -0.102891

v 0.401693 8.152078 -0.198771

v 0.327981 8.104543 -0.281104

v 0.231917 8.068068 -0.344281

v 0.120049 8.045138 -0.383995

v 0.000000 8.037317 -0.397541

v -0.120049 8.045138 -0.383995

v -0.231917 8.068068 -0.344281

v -0.327981 8.104543 -0.281104

v -0.401693 8.152078 -0.198771

v -0.448030 8.207434 -0.102891

v 0.448030 8.182828 -0.084010

v 0.401693 8.104543 -0.162296

v 0.327981 8.037317 -0.229521

v 0.231917 7.985734 -0.281104

v 0.120049 7.953307 -0.313531

v 0.000000 7.942247 -0.324591

v -0.120049 7.953307 -0.313531

v -0.231917 7.985734 -0.281104

v -0.327981 8.037317 -0.229521

v -0.401693 8.104543 -0.162296

v -0.448030 8.182828 -0.084010

v 0.448030 8.163947 -0.059404

v 0.401693 8.068068 -0.114760

v 0.327981 7.985734 -0.162296

v 0.231917 7.922557 -0.198771

v 0.120049 7.882843 -0.221700

v 0.000000 7.869297 -0.229521

v -0.120049 7.882843 -0.221700

v -0.231917 7.922557 -0.198771

v -0.327981 7.985734 -0.162296

v -0.401693 8.068068 -0.114760

v -0.448030 8.163947 -0.059404

v -0.463835 8.266838 0.000000

v 0.448030 8.152078 -0.030750

v 0.401693 8.045138 -0.059404

v 0.327981 7.953307 -0.084010

v 0.231917 7.882843 -0.102891

v 0.120049 7.838547 -0.114760

v 0.000000 7.823439 -0.118809

v -0.120049 7.838547 -0.114760

v -0.231917 7.882843 -0.102891

v -0.327981 7.953307 -0.084010

v -0.401693 8.045138 -0.059404

v -0.448030 8.152078 -0.030750

v 0.448030 8.148029 0.000000

v 0.401693 8.037317 0.000000

v 0.327981 7.942247 -0.000000

v 0.231917 7.869297 0.000000

v 0.120049 7.823438 0.000000

v 0.000000 7.807797 0.000000

v -0.120049 7.823438 0.000000

v -0.231917 7.869297 0.000000

v -0.327981 7.942247 -0.000000

v -0.401693 8.037317 0.000000

v -0.448030 8.148029 0.000000

v 0.448030 8.152078 0.030750

v 0.401693 8.045138 0.059404

v 0.327981 7.953307 0.084010

v 0.231917 7.882843 0.102891

v 0.120049 7.838547 0.114760

v 0.000000 7.823439 0.118809

v -0.120049 7.838547 0.114760

v -0.231917 7.882843 0.102891

v -0.327981 7.953307 0.084010

v -0.401693 8.045138 0.059404

v -0.448030 8.152078 0.030750

v 0.448030 8.163947 0.059404

v 0.401693 8.068068 0.114760

v 0.327981 7.985734 0.162296

v 0.231917 7.922557 0.198771

v 0.120049 7.882843 0.221700

v 0.000000 7.869297 0.229521

v -0.120049 7.882843 0.221700

v -0.231917 7.922557 0.198771

v -0.327981 7.985734 0.162296

v -0.401693 8.068068 0.114760

v -0.448030 8.163947 0.059404

v 0.448030 8.182828 0.084010

v 0.401693 8.104543 0.162296

v 0.327981 8.037317 0.229520

v 0.231917 7.985734 0.281104

v 0.120049 7.953307 0.313531

v 0.000000 7.942247 0.324591

v -0.120049 7.953307 0.313531

v -0.231917 7.985734 0.281104

v -0.327981 8.037317 0.229520

v -0.401693 8.104543 0.162296

v -0.448030 8.182828 0.084010

v 0.448030 8.207434 0.102891

v 0.401693 8.152078 0.198771

v 0.327981 8.104543 0.281104

v 0.231917 8.068068 0.344281

v 0.120049 8.045138 0.383995

v 0.000000 8.037317 0.397541

v -0.120049 8.045138 0.383995

v -0.231917 8.068068 0.344281

v -0.327981 8.104543 0.281104

v -0.401693 8.152078 0.198771

v -0.448030 8.207434 0.102891

v 0.448030 8.236088 0.114760

v 0.401693 8.207434 0.221700

v 0.327981 8.182828 0.313531

v 0.231917 8.163947 0.383995

v 0.120049 8.152078 0.428291

v 0.000000 8.148029 0.443400

v -0.120049 8.152078 0.428291

v -0.231917 8.163947 0.383995

v -0.327981 8.182828 0.313531

v -0.401693 8.207434 0.221700

v -0.448030 8.236088 0.114760

v 0.448030 8.266838 0.118809

v 0.401693 8.266838 0.229520

v 0.327981 8.266838 0.324591

v 0.231917 8.266838 0.397541

v 0.120049 8.266838 0.443400

v -0.000000 8.266838 0.459041

v -0.120049 8.266838 0.443400

v -0.231917 8.266838 0.397541

v -0.327981 8.266838 0.324591

v -0.401693 8.266838 0.229520

v -0.448030 8.266838 0.118809

v 0.463835 8.266838 0.000000

v 0.448030 8.297588 0.114760

v 0.401693 8.326242 0.221700

v 0.327981 8.350848 0.313531

v 0.231917 8.369729 0.383995

v 0.120049 8.381598 0.428291

v -0.000000 8.385647 0.443400

v -0.120049 8.381598 0.428291

v -0.231917 8.369729 0.383995

v -0.327981 8.350848 0.313531

v -0.401693 8.326242 0.221700

v -0.448030 8.297588 0.114760

v 0.448030 8.326242 0.102891

v 0.401693 8.381598 0.198771

v 0.327981 8.429133 0.281104

v 0.231917 8.465609 0.344281

v 0.120049 8.488538 0.383995

v -0.000000 8.496359 0.397541

v -0.120049 8.488538 0.383995

v -0.231917 8.465609 0.344281

v -0.327981 8.429133 0.281104

v -0.401693 8.381598 0.198771

v -0.448030 8.326242 0.102891

v 0.448030 8.350848 0.084010

v 0.401693 8.429133 0.162295

v 0.327981 8.496359 0.229520

v 0.231917 8.547942 0.281104

v 0.120049 8.580369 0.313531

v -0.000000 8.591429 0.324591

v -0.120049 8.580369 0.313531

v -0.231917 8.547942 0.281104

v -0.327981 8.496359 0.229520

v -0.401693 8.429133 0.162295

v -0.448030 8.350848 0.084010

v 0.448030 8.369729 0.059404

v 0.401693 8.465609 0.114760

v 0.327981 8.547942 0.162295

v 0.231917 8.611119 0.198771

v 0.120049 8.650833 0.221700

v -0.000000 8.664379 0.229520

v -0.120049 8.650833 0.221700

v -0.231917 8.611119 0.198771

v -0.327981 8.547942 0.162295

v -0.401693 8.465609 0.114760

v -0.448030 8.369729 0.059404

v 0.448030 8.381598 0.030750

v 0.401693 8.488538 0.059404

v 0.327981 8.580369 0.084010

v 0.231917 8.650833 0.102891

v 0.120049 8.695129 0.114760

v -0.000000 8.710238 0.118808

v -0.120049 8.695129 0.114760

v -0.231917 8.650833 0.102891

v -0.327981 8.580369 0.084010

v -0.401693 8.488538 0.059404

v -0.448030 8.381598 0.030750

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ СТВОРЕННЯ 3D-МОДЕЛЕЙ ДЛЯ ІГРОВИХ РУШІЇВ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

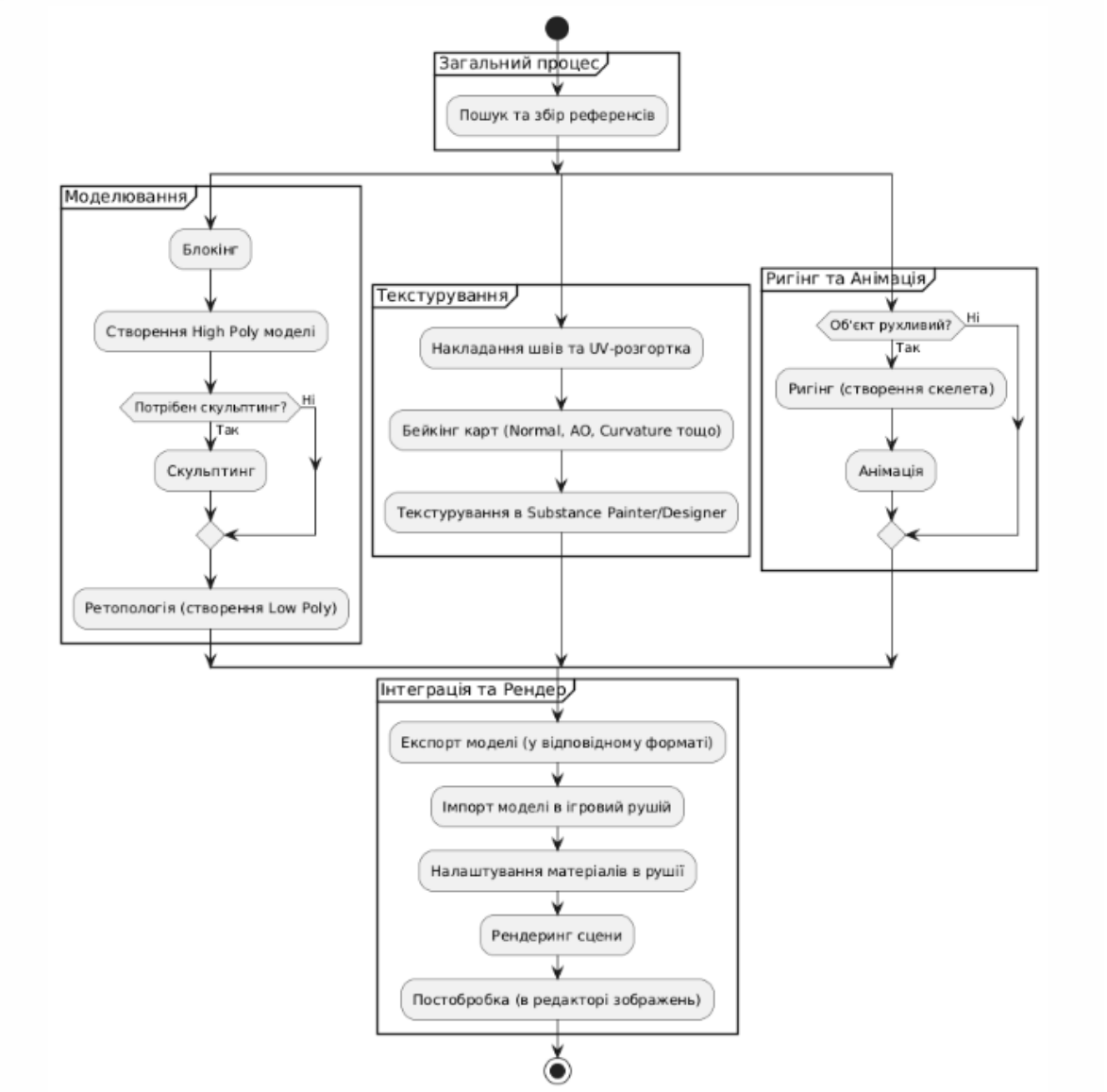

Рисунок Г.1 - Діаграма діяльності(Activity Diagram) для загального процесу створення моделей

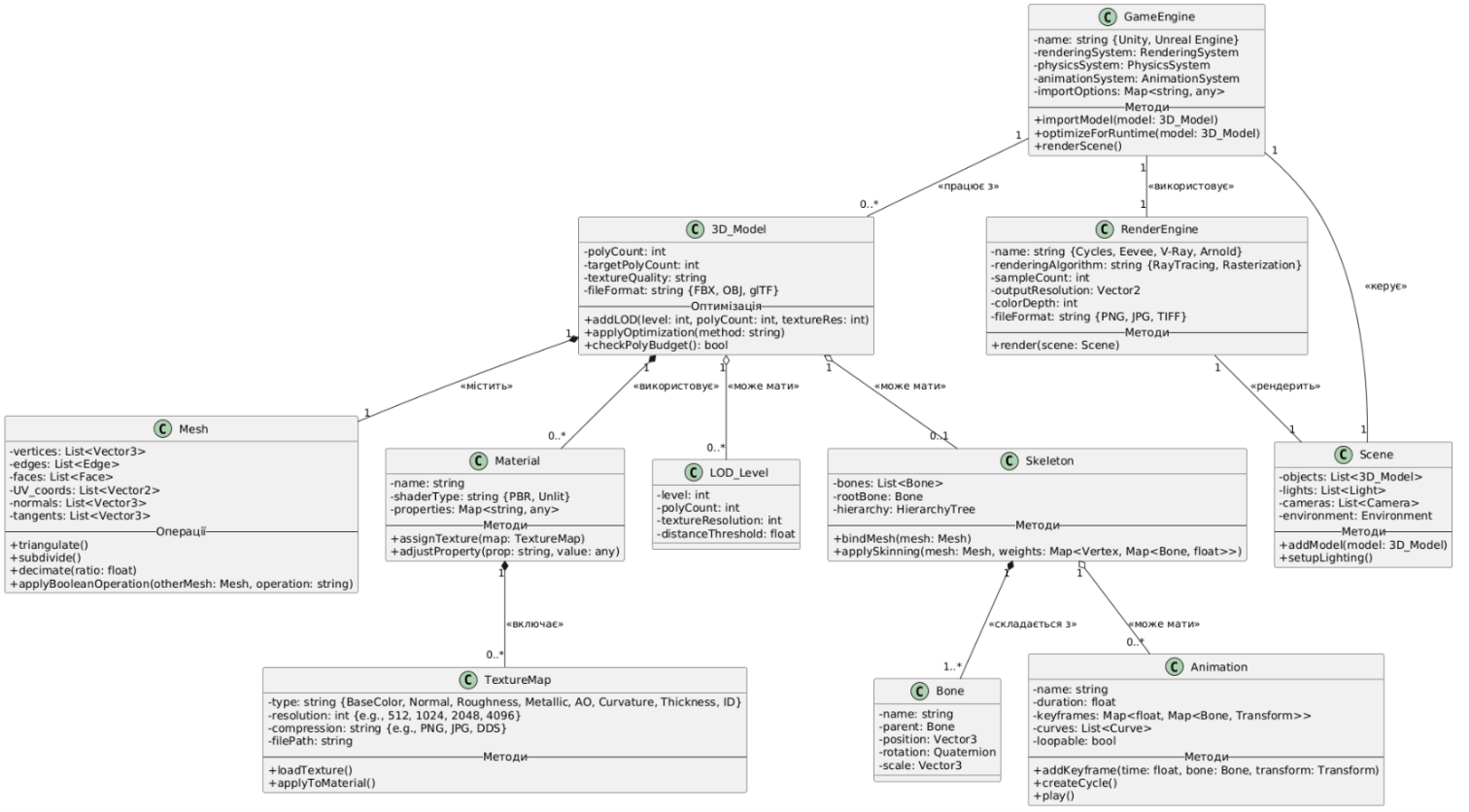

Рисунок Г.2 – Діаграма класів (Class Diagram)

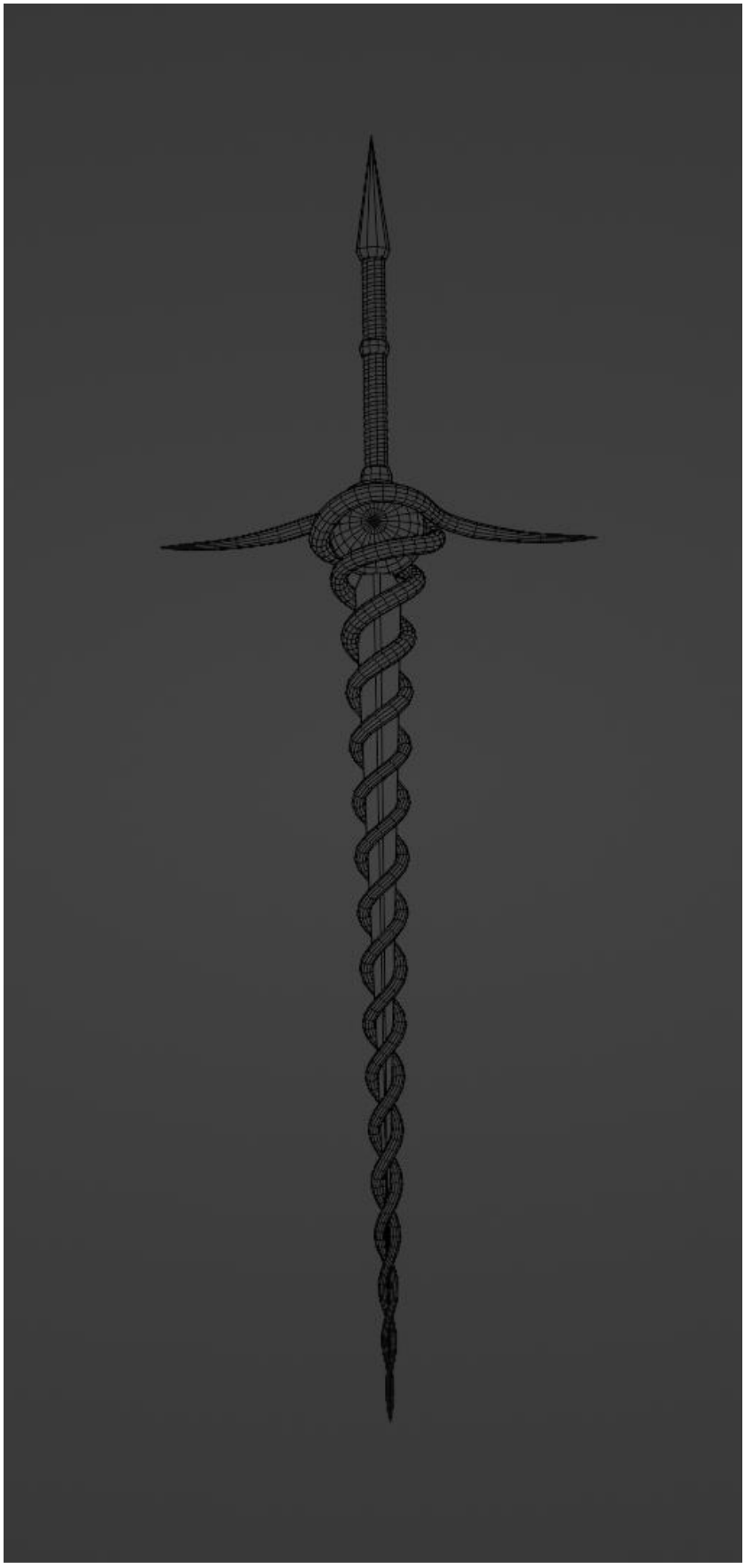

Рисунок Г.3 – Меш і сітка моделі

Рисунок Г.4 – Затекстурована модель

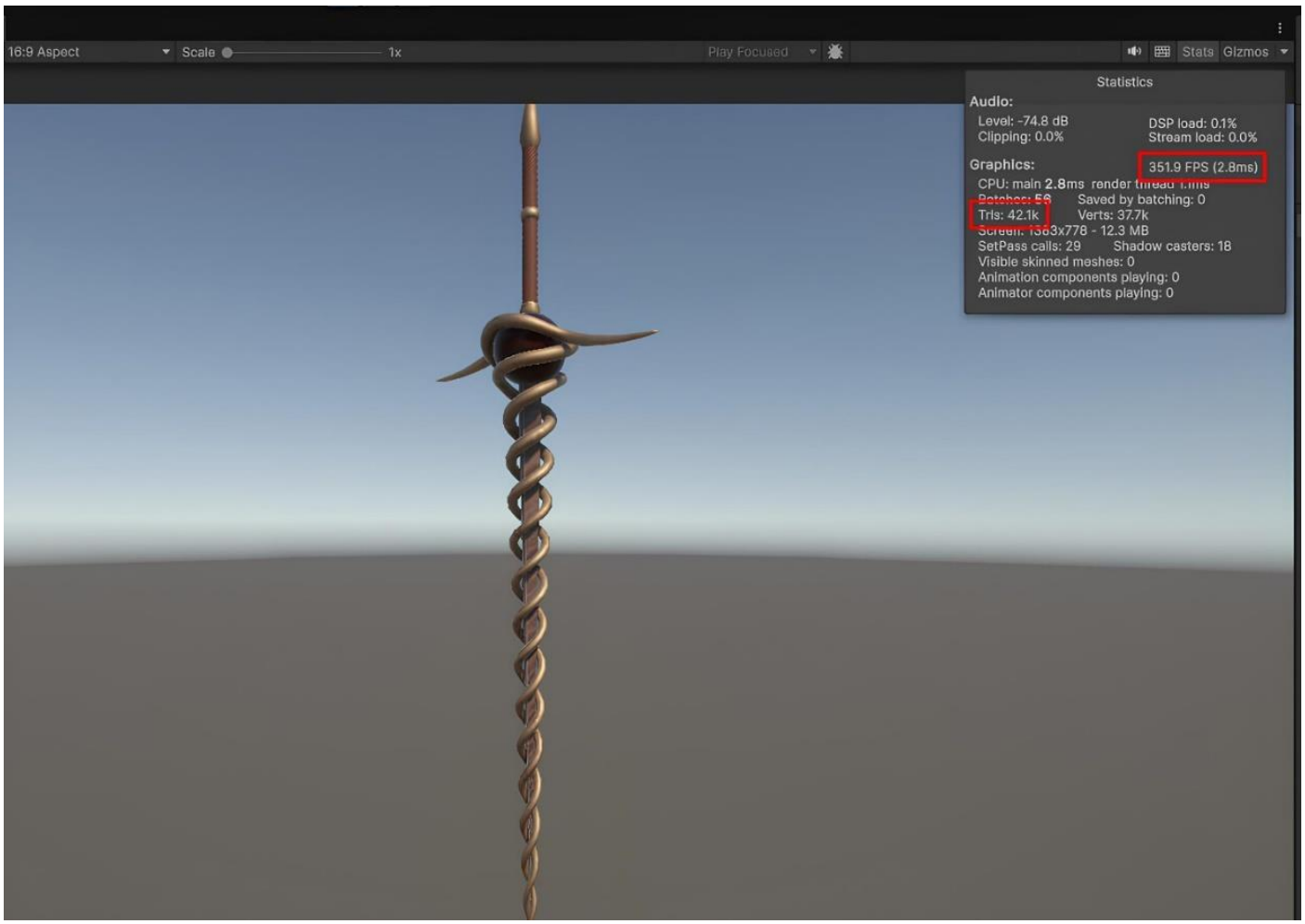

Рисунок Г.5 – Замір продуктивності в рушії

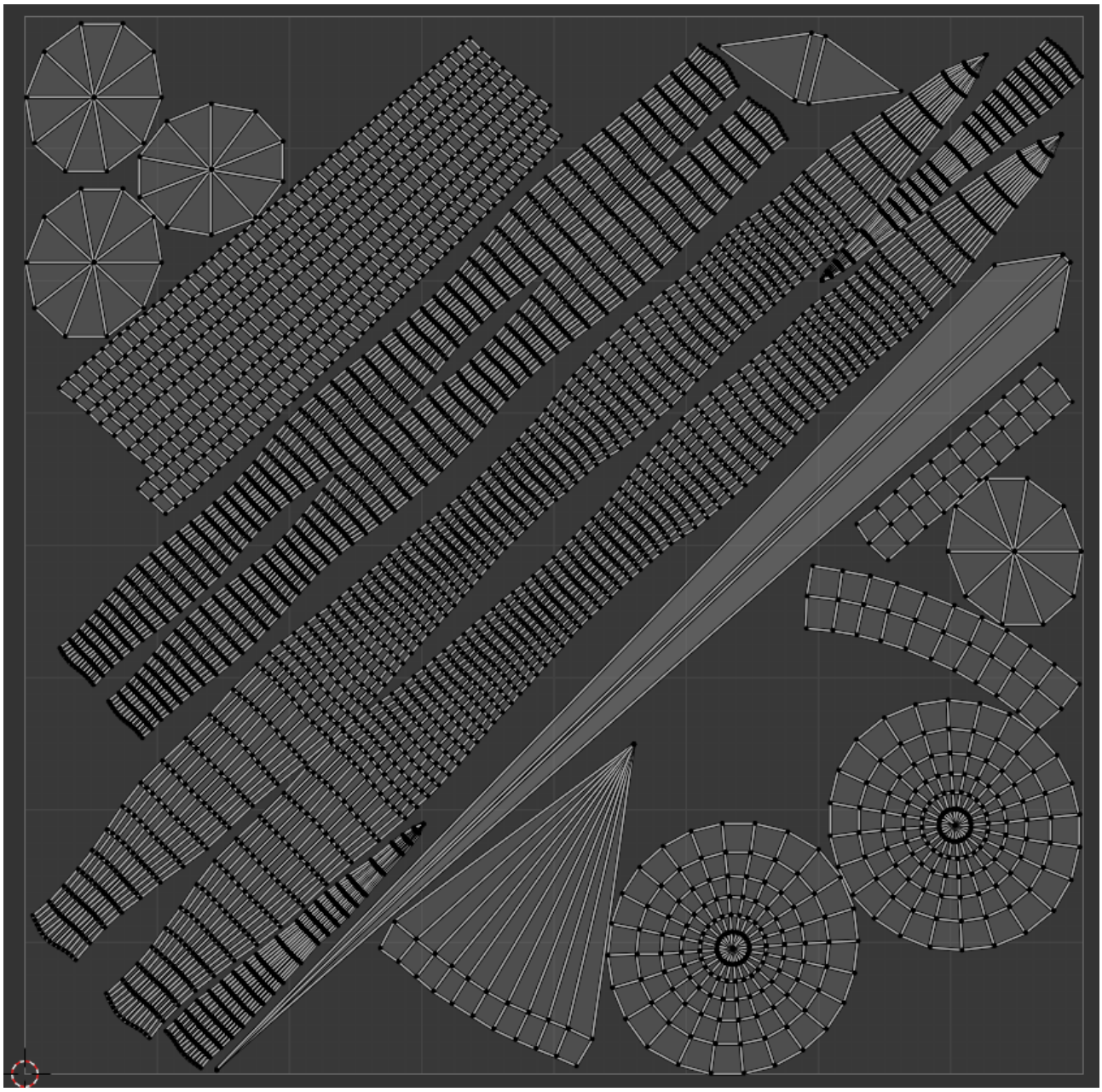

Рисунок Г.6 – UV-Розгортка

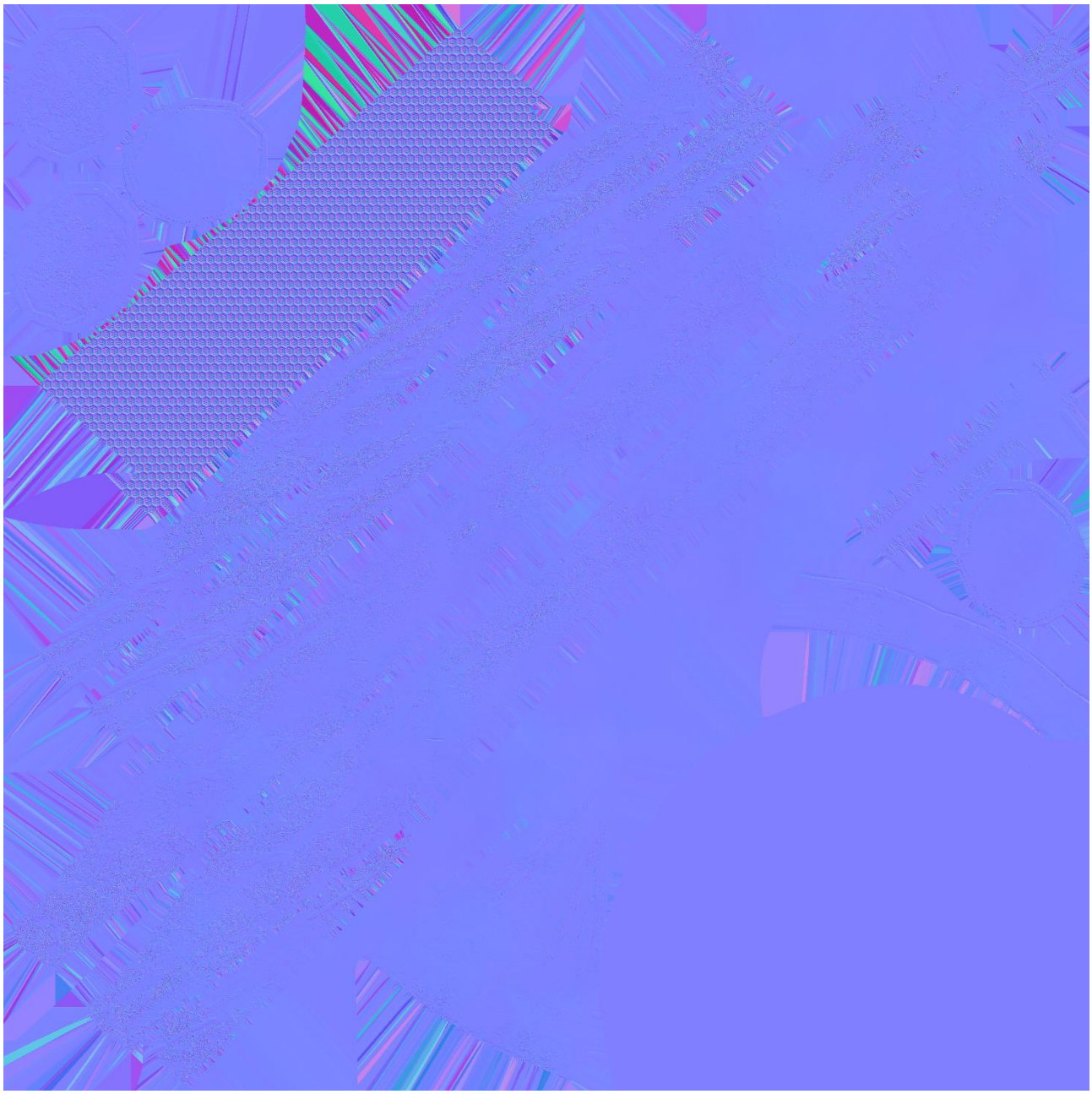

Рисунок Г.7 – Карта нормалей

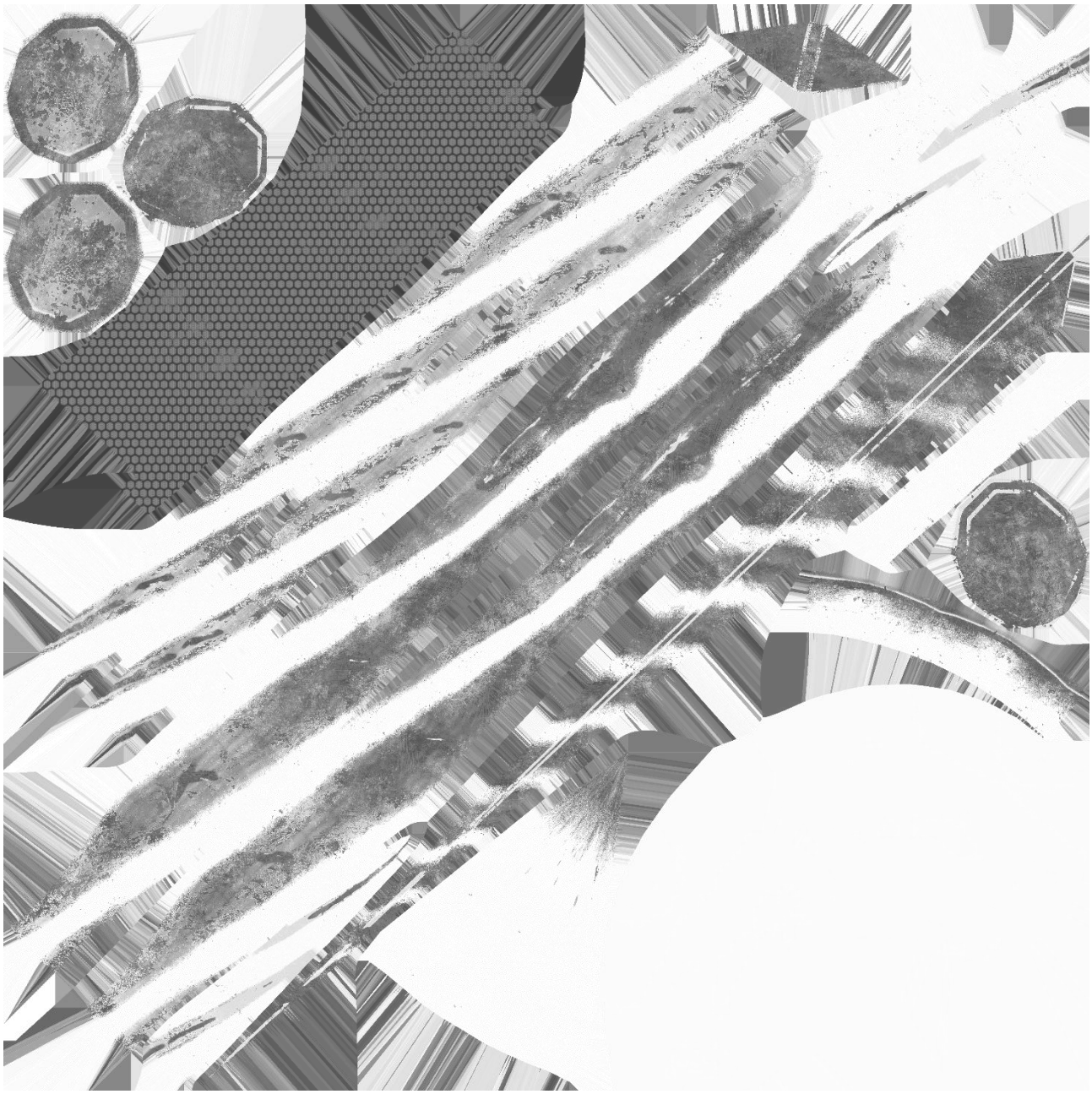

Рисунок Г.8 – Металічна шорохуватість

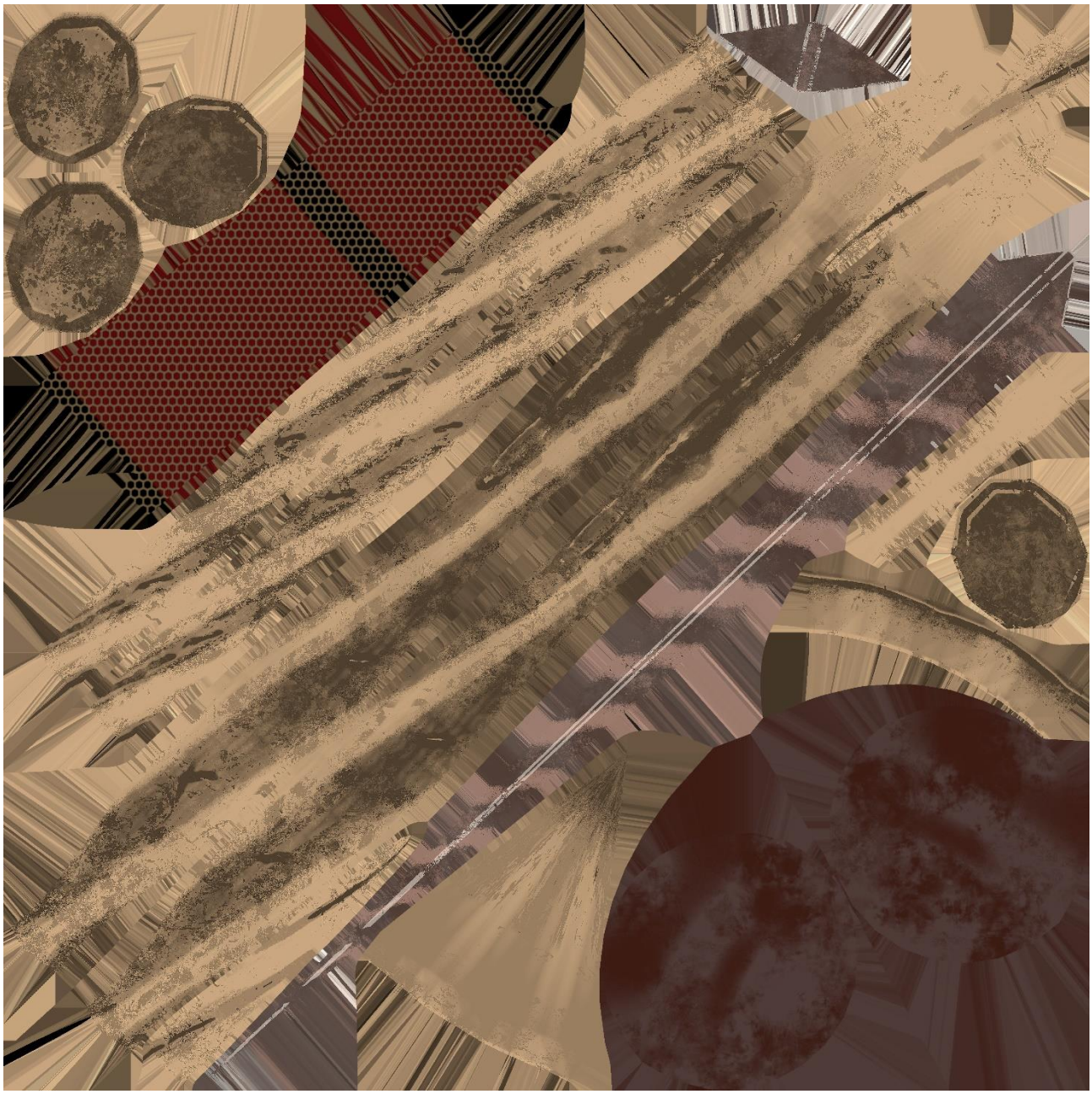

Рисунок Г.9 – Базовий колір