Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційних радіотехнологій та технічного захисту інформації
(повна назва)

Кафедра медіаінженерії та інформаційних радіоелектронних систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Комплексна тема. Оптимізація 3D моделі для ігрового рушія. Unreal Engine 5
(тема)

Виконав:
студент 2 курсу, групи СТМм-22-1
Сергій РОГИНСЬКИЙ
(прізвище, ініціали)

Спеціальність 171 Електроніка
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Системи, технології і комп'ютерні засоби мультимедіа
(повна назва освітньої програми)

Керівник асист. Тетяна ВАСИЛЕНКО
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Володимир КАРТАШОВ
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій та технічного захисту інформації
Кафедра Медіаінженерії та інформаційних радіоелектронних систем

Рівень вищої освіти другий (магістерський)

Спеціальність 171 Електроніка
(код і повна назва)
Тип програми освітньо-професійна

Освітня програма "Системи, технології і комп'ютерні засоби мультимедіа"
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Рогинському Сергію Вячеславовичу

(прізвище, ім'я, по батькові)

1. Тема роботи Комплексна тема. Оптимізація 3Д моделі для ігрового рушія. Unreal Engine 5 затверджена наказом університету від 20.12.2023 р. № 1371 СТ
2. Термін подання студентом роботи до екзаменаційної комісії 08.01.2024 р.
3. Вихідні дані до роботи:
1. Дослідити сучасні програмні продукти ігрових рушіїв.
2. Створити і оптимізувати ігрову модель.
3. Створити ігровий рівень в ігровому рушії використовуючи створену оптимізовану 3D модель.
4. Перелік питань, що потрібно опрацювати в роботі
ВСТУП
1. ОБҐРУНТУВАННЯ ВИБОРУ ІГРОВОГО РУШІЯ
2. СТВОРЕННЯ І ОПТИМІЗАЦІЯ ІГРОВОЇ 3D МОДЕЛІ
3. СТВОРЕННЯ ІГРОВОГО РІВНЯ І РОБОТА З ОПТИМІЗОВАНОЮ 3D МОДЕЛЛЮ В UNREAL ENGINE 5
ВИСНОВКИ
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ
ДОДАТКИ
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій:
1. Візуалізація етапів моделювання в Blender. 2. Виконана і оптимізована UV розгортка моделі. 3. Оптимізована лоуполі модель в Blender. 4. Неоптимізована хайполі модель в Blender. 5. Фінальний результат текстурування в Substance Painter.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	ОБҐРУНТУВАННЯ ВИБОРУ ІГРОВОГО РУШІЯ	21.12.23-22.12.23	
2	СТВОРЕННЯ І ОПТИМІЗАЦІЯ ІГРОВОЇ 3D МОДЕЛІ	23.12.23-25.12.23	
3	СТВОРЕННЯ ІГРОВОГО РІВНЯ І РОБОТА З ОПТИМІЗОВАНОЮ 3D МОДЕЛЛЮ В UNREAL ENGINE 5	26.12.23-28.12.23	
4	Графічна частина проекту	29.12.23-31.12.23	
5	Перевірка керівником проекту	29.12.23-30.12.23	
6	Перевірка норм. контролем	02.01.24-03.01.24	
7	Перевірка на академічний плагіат	04.01.24-05.01.24	
8	Перевірка завідувачем кафедри, рецензування	06.01.24-07.01.24	

Дата видачі завдання: 20.12.2023 р.

Студент _____ Сергій РОГИНСЬКИЙ

(підпис)

Керівник роботи: асист.каф. Василенко Т.О.

(підпис) (посада, прізвище, ініціали)

.

РЕФЕРАТ

Пояснювальна записка до атестаційної роботи: 90 сторінок, 83 рисунки, 2 таблиці, 3 формули, 23 джерела.

UNREAL ENGINE, BLENDER, SUBSTANCE PAINTER, 3D, МОДЕЛЮВАННЯ, РОЗГОРТКА, ТЕКСТУРУВАННЯ, ОПТИМІЗАЦІЯ.

Об'єкт дослідження — оптимізація 3Д моделі для ігрового рушія. Unreal Engine 5.

Предмет дослідження — розробка оптимального робочого процесу створення графічного 3D контенту та ігрового рівня для ігрової індустрії.

Мета роботи — проаналізувати та реалізувати найефективніші техніки створення 3D контенту для використання в ігровому рушії, створити оптимізовану 3D модель за допомогою реалізованих технік, створити ігровий рівень в Unreal Engine 5.

В атестаційній роботі проведено теоретичний аналіз математичних моделей використаних у процесі моделювання.

У першому розділі проведено теоретичний аналіз програмних продуктів ігрових рушіїв та математичних моделей, які використовуються у процесі створення ігрових рівнів.

У другому розділі розроблено та реалізовано ефективні техніки створення та оптимізацію 3D контенту для використання в ігрових рушіях. Створення 3D моделі було виконано використовуючи програмне забезпечення Blender. Розгортка і пакування було виконано використовуючи програмне забезпечення RizomUV. Текстурування було виконано використовуючи програмне забезпечення Substance Painer.

У третьому розділі виконано створення ігрового рівня в ігровому рушії Unreal Engine 5. Було виконано порівняння продуктивності рівня на двох комп’ютерних системах з різним рівнем технічних характеристик.

ABSTRACT

Explanatory note to the attestation work: 90 pages, 83 images, 2 tables, 3 formulas, 23 sources.

BLENDER, SUBSTANCE PAINTER, 3D, MODELING, UNWRAPPING, TEXTURING, RENDERING, POLYGON.

The object of research is optimization of a 3D model for a game engine. Unreal Engine 5.

The subject of research is the development of an optimal workflow for creating graphic 3D content and a game level for the game industry.

The purpose of research is to analyze and implement the most effective techniques for creating 3D content for use in a game engine, to create an optimized 3D model using the implemented techniques, to create a game level in Unreal Engine 5.

In the certification work, a theoretical analysis of the mathematical models used in the modeling process was carried out.

In the first chapter, a theoretical analysis of software products of game engines and mathematical models, which are used in the process of creating game levels, is carried out.

In the second chapter, effective techniques for creating and optimizing 3D content for use in game engines are developed and implemented. The 3D model was created using Blender software. Unwrapping and packing was performed using RizomUV software. Texturing was done using Substance Painer software.

In the third chapter, the game level was created in the Unreal Engine 5 game engine. The performance of the level was compared on two computer systems with different levels of technical characteristics.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

2D (англ. two-dimensional) – двовимірний.

3D (англ. three-dimensional) – тривимірний.

AAA (англ. Triple-A) – це неофіційна класифікація, яка використовується для класифікації відеоігор, які виробляються та розповсюджуються середнім або великим видавцем, який зазвичай має вищі бюджети на розробку та маркетинг, ніж інші рівні ігор.

HP (англ. HighPoly) – високополігональний.

LP (англ. LowPoly) – низькополігональний.

NPC (англ. Non-Player Character) – неігровий персонаж.

ORM – комбінована карта Occlusion Roughness Metallic.

RGB (англ. Red, Green, Blue) – адитивна колірна модель, у якій червоний, зелений і синій основні кольори світла додаються різними способами для відтворення широкого спектру кольорів.

SSAA (англ. SuperSampling Anti-Aliasing) – метод просторового згладжування, який використовується для видалення аліасінгу (зубчастих і піксельних країв) із зображень, відтворених у комп’ютерних іграх.

UE4 – Unreal Engine 4

UE5 – Unreal Engine 5

VR (англ. Virtual Reality) – віртуальна реальність, симульований досвід, який використовує відстеження пози та 3D-дисплеї близько до очей, щоб дати користувачеві відчуття занурення у віртуальний світ.

ВСТУП

За останні роки, ігрова індустрія стала найприбутковішою галуззю в сфері розваг оціненою в $192 млрд, обігнавши книги і кінематограф з результатами в $120 млрд і $99 млрд відповідно. Через такий стрімкий зріст, кожного дня з'являються нові ігрові студії і створюються нові ігри. Але така популярність і перенасичення ринку викликали одну велику проблему – зниження якості фінального продукту.

Такий занепад може проявлятися в поганому геймплеї, частих багах і помилках, хижацьких практиках монетизації, тощо. Але найпоширенішою проблемою є оптимізація, а точніше – її відсутність.

Правильно оптимізована 3D модель – вирішальна частина позитивного ігрового досвіду для користувача бо від неї залежить не тільки кількість кадрів за секунду, а й коректність роботи фізичних симуляцій, відображення текстур та геймплей в цілому.

Для 3D художників, оптимізація є невідкладною частиною робочого пайплайну, але не кожен художник може отримати вільний доступ до уроків і технік правильної оптимізації. Дуже часто саме такі вирішальні знання знаходяться за регулярними підписками, дорогими курсами та платними відеоуроками.

Пайплайн (англ. Pipeline) – поетапний робочий процес для виконання специфічної задачі.

Саме через таку тенденцію монетизації просунутих технік оптимізації та через стрімке зростання популярності ігрової індустрії в світі, є актуальним створити ефективний пайплайн створення і оптимізації 3D моделі для використання в ігровому рушії для подальшого створення якісних відеоігор.

1 ОБҐРУНТУВАННЯ ВИБОРУ ІГРОВОГО РУШІЯ

1.1 Порівняння популярних ігрових рушіїв в ігровій індустрії

Коли мова йде про розробку ігор, вибір правильного ігрового рушія є одним із найважливіших рішень. Оскільки на ринку представлено досить різноманітні варіанти, може знадобитися багато часу, щоб знайти той, який найкраще відповідає вашим потребам.

Існує багато ігрових рушіїв, які можна використовувати для розробки ігор для ПК, консольних і мобільних ігор. Такі як:

- Unreal Engine;
- Unity;
- Godot;
- Ігрові рушії Amazon;
- CryEngine.

1.1.1 Unreal Engine

Unreal Engine це ігровий рушій, розроблений Epic Games з 1998 року.

Unreal Engine — це добре задокументований і простий у використанні ігровий рушій. Розробники можуть використовувати його для розробки будь-яких типів ігор — від консольних до мобільних (включаючи Android та iOS). Він навіть використовується в галузях за межами розробки ігор, включаючи автомобільну.

Це робить його найкращим ігровим рушієм як для новачків, так і для експертів. Ось чому всі, від розробників ігор AAA до інді-студій, використовують Unreal Engine, як показано на (рис.1.1).

Рисунок 1.1 — Приклад гри на рушії Unreal Engine 5 [1]

Сильні сторони:

- Інтуїтивно зрозуміла система планів для непрограмістів;
- Чудово підходить для високоякісної графіки;
- Підвищена продуктивність порівняно з іншими ігровими рушіями;
- Найкращий вибір для VR;
- Unreal Engine Marketplace, який містить безкоштовні ресурси.

Слабкі сторони:

- Краще для 3D-ігор, ніж для 2D і мобільних ігор;
- Високі системні вимоги для більш розширених функцій;
- Вищий поріг входу.

1.1.2 Unity

Unity це топовий ігровий рушій, розроблений Unity Technologies з 2005 року.Unity робить розробку ігор доступнішою завдяки помітній підтримці програми зчитування з екрана. Розробники можуть використовувати Unity для

розробки ігор на різних платформах, як показано на (рис.1.2). Спочатку він був запущений для MacOS, але зараз підтримує понад 25 платформ. Як і Unreal Engine, Unity зараз використовується в інших галузях, що окрім розробки ігор, включаючи архітектуру.

Рисунок 1.2 — Приклад гри на рушії Unity [2]

Це робить його найкращим ігровим рушієм для кросплатформних команд з упором в мобільну індустрію тому що він має гарну репутацію у розробці мобільних ігор на Android. Також Unity використовується для ПК, консолей, веб та ігор AR/VR.

Сильні сторони:

- Безкоштовно для початківців;
- Гнучкий і розширюваний ігровий рушій із готовими компонентами;
- Ідеально підходить як для 2D, так і для 3D ігор;
- Потужна підтримка розробки мобільних ігор;
- Доступні комплекти розробки програмного забезпечення (SDK) для VR і AR;

– Широкі кросплатформні можливості;
– Unity Asset Store із безкоштовними активами.

Слабкі сторони:

– Професійні ліцензії дорогі;
– Необхідність вивчити нову мову програмування;
– Немає доступу до вихідного коду для невеликих команд розробників;
– Високі системні вимоги для більш розширених функцій;
– Багато змін в інтерфейсі;
– Немає підтримки посилань на зовнішні бібліотеки;
– Громіздкий і повільний.

1.1.3 Godot

Створений в 2014 році, Godot це кросплатформний, безкоштовний ігровий рушій із відкритим кодом, випущений за ліцензією MIT.

Рисунок 1.3 — Приклад гри на рушії Godot [3]

Godot використовує унікальний підхід до архітектури вузлів і сцен, щоб представити певні ігрові функції, які деякі користувачі вважають більш простими та інтуїтивно зрозумілими у використанні, ніж деякі інші ігрові рушії, як показано на (рис.1.3). Крім того, Godot використовує власну мову для сценаріїв GDScript, схожу на Python.

Сильні сторони:

- Повністю безкоштовний ігровий рушій із відкритим кодом;
- Ідеально підходить як для 2D, так і для 3D ігор;
- Кросплатформні можливості;
- Унікальна архітектура для розробки ігор.

Слабкі сторони:

- Менше ресурсів і матеріалів, ніж інші ігрові рушії.
- Легка крива навчання.

1.1.4 Amazon Lumberyard

Amazon Lumberyard це безкоштовний кросплатформний ігровий рушій з відкритим кодом., розроблений Amazon, як показано на (рис.1.4). Заснований у 2016 році, вихідний код був спочатку ліцензований і перероблений з CryEngine (див. нижче).

Open3D є останньою розробкою від Amazon. Він будується на основі Lumberyard і створює більш модульний підхід до розробки ігор. Це легкий ігровий рушій для початку, який робить його доступним для стартапів або інді-студій, а також для студій розробки ігор AAA.

Рисунок 1.4 — Приклад гри на рушії Amazon Lumberyard [4]

Сильні сторони:

- Безкоштовно для одиночних ігор;
- Розширена підтримка VR;
- Інтеграція з AWS для багатокористувацьких функцій онлайн;
- Вбудована інтеграція з Twitch;
- Кросплатформні можливості;
- Надійне відтворення для зовнішнього середовища.

Слабкі сторони:

- Не підходить для менших проектів;
- Менше ресурсів і матеріалів, ніж інші ігрові рушії;

1.1.5 CryEngine

CryEngine — ігровий рушій, розроблений компанією CryTek у 2002 році, останнім рушієм є CryEngine V, як показано на (рис.1.5).

CryEngine є кросплатформним двигуном. Розробники можуть використовувати його для створення фотореалістичних ігор від першої особи, особливо

шутерів. CryEngine використовується у всіх іграх від CryTek, а також деяких інших студій.

CryEngine використовується в Hunt: Showdown, Crysis, Far Cry, а також Kingdom Come: Deliverance.

Рисунок 1.5 — Приклад гри на рушії Cryengine [5]

Сильні сторони:

- Функції за замовчуванням великі;
- Можливі високоякісні зображення;
- Швидкий процес ітерації;
- Потужний редактор ігор Sandbox;
- Підтримка VR;
- Легко навчатися.

Слабкі сторони:

- Крута крива навчання для початківців;
- Менше ресурсів і матеріалів, ніж інші ігрові рушії.

Отже, оглянувши найпопулярніші рушії в індустрії, є один ігровий рушій, який виділяється серед інших своєю доступністю, різнобічністю, гнучкістю і якістю фінального продукту, і це — Unreal Engine.

Unreal Engine — потужний і універсальний ігровий рушій, розроблений Epic Games. Він широко використовується для розробки ігор на різних платформах, включаючи ПК, мобільні пристрої та консолі. Давайте дослідимо переваги Unreal Engine для розробки ігор.

1.2 Унікальні особливості та переваги Unreal Engine 5

Unreal Engine 4 — це ігровий движок, випущений у 2014 році. Розробники ігор можуть використовувати UE4 для створення ігор та інших віртуальних ресурсів.

Порівняно з Unreal Engine 4, Unreal Engine 5 дозволяє розробникам легше створювати 3D-контент і ігровий простір у реальному часі. Крім того, UE5 містить оновлення редактора Unreal Editor і розширені інструменти анімації та розробки.

1.2.1 Особливості Unreal Engine 5

Unreal Engine є основним компонентом розробки ігор і віртуального виробництва. З UE5 команди мають змогу створювати великі проекти. У різних галузях творці можуть співпрацювати в режимі реального часу, щоб створювати виняткові візуальні ефекти для кінцевих користувачів.

Однією з найбільш значущих переваг Unreal Engine є його здатність створювати високоякісну графіку. Двигун має потужний рушій візуалізації (або рендерингу), який може відтворювати складні сцени з високою точністю. Він пропонує розширені функції освітлення та тіні, завдяки чому ваша гра виглядає більш реалістичною та захоплюючою. Крім того, Unreal Engine підтримує розширені ефекти постобробки, такі як розмивання, відблиски на лінзах і розмиття, що може ще більше покращити візуальну якість вашої гри.

Unreal Engine 5 (UE5) — остання версія Unreal Engine, одного з найпотужніших і найпопулярніших ігрових движків. Unreal містить такі функції:

Nanite

Напевно, найвидатнішою рисою UE5 є Nanite – революційна технологія, яка пропонує виняткову візуальну якість і масштабованість завдяки використанню віртуалізованої геометрії, як показано на (рис.1.6).

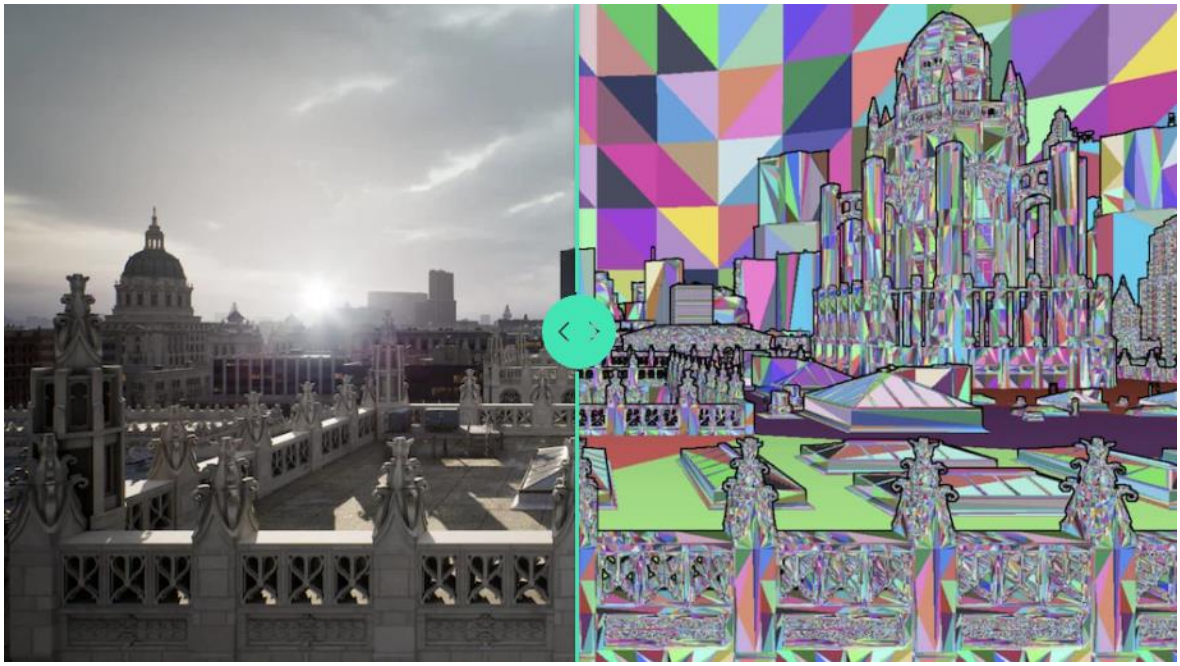


Рисунок 1.6 – Використання Nanite в Unreal Engine 5 [6]

Спочатку, він розбиває кожну сітку на кластери з різним рівнем деталізації. Враховуючи відстань і роздільну здатність екрана, Nanite генерує LOD (Levels Of Detail) для кожного кластера, а потім інтелектуально передає геометричні дані на основі того, що бачить камера в будь-який момент.

Nanite також використовує ефективну техніку вилучення екранного простору для ще більшої оптимізації продуктивності рендерингу. Аналізуючи вигляд камери, він визначає, які кластери є видимими та мають бути відрендерені, зменшуючи навантаження на GPU. Це дозволяє створювати неймовірно складні сцени з мільйонами полігонів для підтримки оптимальної частоти кадрів і продуктивності.

Полігон (англ. Polygon) – багатокутник, або замкнена ламана, що відображає площину між 3 і більше геометричними точками.

Lumen

Створення реалістичного віртуального світу зводиться до освітлення. В Unreal Engine 5 існує глобальне рішення для освітлення під назвою Lumen, як показано на (рис.1.7). Воно точно і реалістично імітує поведінку світла в сцені.

Завдяки повністю динамічному глобальному освітленню та відображенню Lumen дозволяє створювати непряме освітлення, яке реагує на пряме освітлення та геометрію.

Часи традиційних методів випікання світла на моделях минули. Lumen адаптується до змін навколишнього середовища в реальному часі, забезпечуючи більш реалістичне та динамічне освітлення, що потребує менше ручної роботи від художників.

Рисунок 1.7 – Використання Lumen в Unreal Engine 5 [7]

Наприклад, освітлення можна налаштувати відповідно до часу доби, нового джерела світла (наприклад, ліхтарик), раптового проміння, що потрапляє в кадр,

тощо. Lumen регулює освітлення від відкритих динамічних сцен до найдрібніших деталей.

Virtual Shadow Maps

Віртуальні карти тіней — це новий метод відображення тіней, який пропонує UE5, забезпечуючи узгоджене затінення з високою роздільною здатністю. Цей інструмент був розроблений для збільшення роздільної здатності тіні відповідно до нової високодеталізованої геометрії, запропонованої UE5.

UE5 також пропонує правдоподібні м’які тіні та забезпечує просте та зрозуміле рішення для створення тіней, усе з контрольованими витратами на продуктивність. Ідеально підходить для створення реалістичних сцен і захоплюючих ігрових просторів, як показано на (рис.1.8).

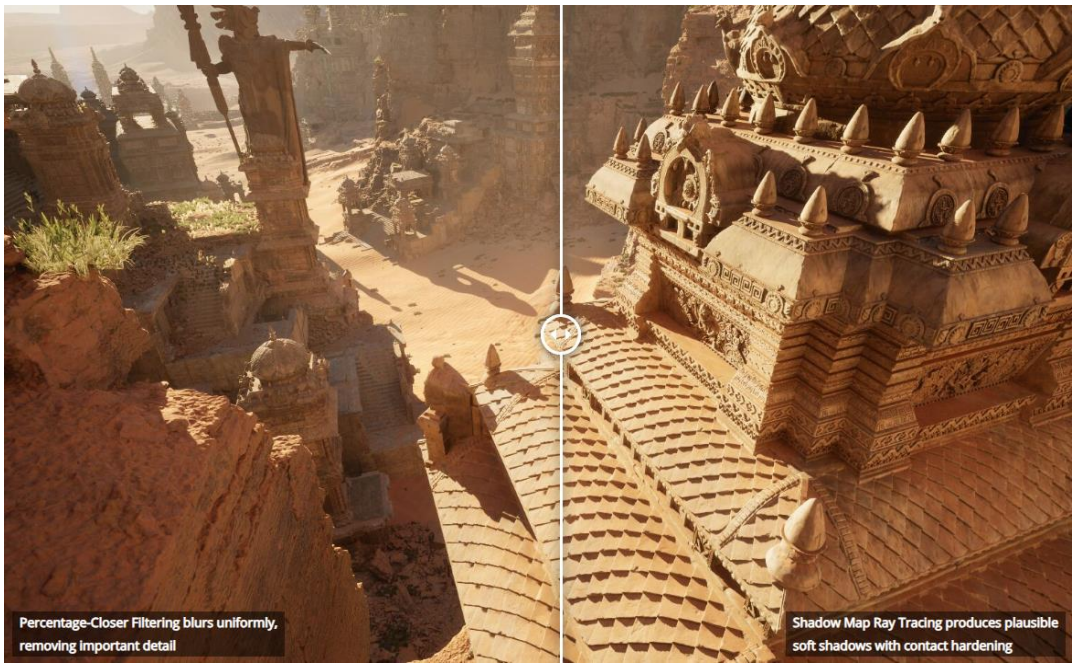


Рисунок 1.8 – Використання Virtual Shadow Maps в Unreal Engine 5 [8]

Chaos Physics

UE5 також надає нові можливості для керування фізикою та зіткненнями. Основною системою, відповідальною за це, є Chaos Physics. Використовуючи LOD і інтелектуальне виявлення зіткнень, Chaos Physics дозволяє проводити більш

реалістичні та інтерактивні симуляції на основі фізики в грі. Його функція Chaos Destruction дозволяє розробникам створювати динамічні та руйнівні середовища, які реагують на події в грі та дії гравців, як показано на (рис.1.9).

Chaos Physics також пропонує автоматичне коригування складності для фізичних об’єктів на основі їх відстані від камери, забезпечуючи високу продуктивність у фізико-навантажених сценах. Він легко інтегрується з іншими системами, такими як Niagara, Animation Blueprints і Control Rig, для динамічної взаємодії персонажів із середовищем і VFX.

Рисунок 1.9 – Використання Chaos Physics в Unreal Engine 5 [9]

World Partition

Unreal Engine 4 вже відомий тим, що дозволяє створювати відкриті світи. Але UE5 виводить це на наступний рівень, прискорюючи створення великомасштабного світу та спрощуючи взаємодії в ньому.

Коли мова йде про рендеринг середовищ відкритого світу, UE5 використовує всі доступні інструменти для потокової передачі. Окрім Nanite, Lumen і LODs, World Partition відіграє вирішальну роль. Він забезпечує ефективне використання пам’яті, завантажуючи та вивантажуючи секції ігрового світу залежно від місця розташування гравця, як показано на (рис.1.10).

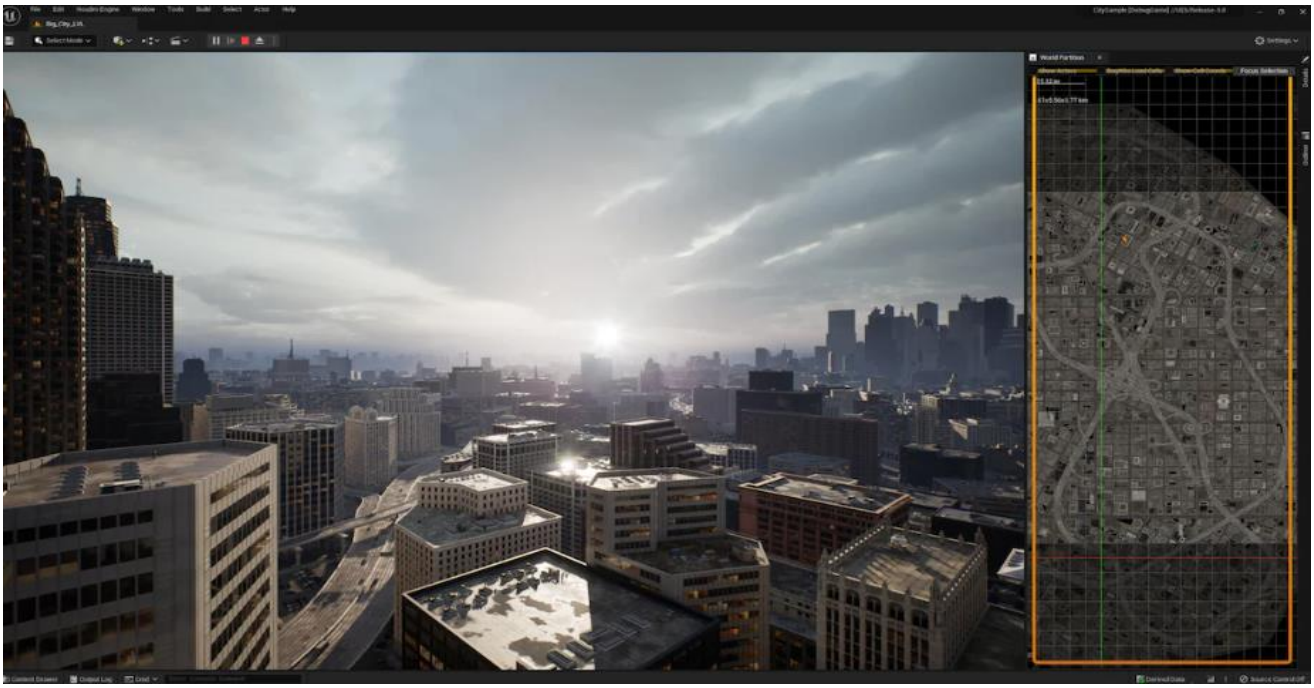


Рисунок 1.10 – Використання World Partition в Unreal Engine 5 [10]

Система World Partition використовує сітку для відображення підрівнів цілого всесвіту. Ви можете керувати складними рівнями, які завантажуються та розвантажуються, коли гравець йде по ландшафту. Крім того, система One File Per Actor допомагає командам працювати паралельно. Це зменшує кількість файлів в проекті, зберігаючи дані як зовнішні файли для кожного учасника.

Рушій також використовує алгоритми відсічення оклюзії, які ідентифікують видимі об’єкти для візуалізації, забезпечуючи відображення лише необхідних ассетів. Інші функції, такі як стиснення ассетів і потокова передача, ще більше покращують використання пам’яті та продуктивність. Ассет (англ. Asset) – створений і готовий для використання контент. Може бути як тривимірною моделлю, так і набором ігрових правил.

MetaHuman Character Creator

Постійно зростаюча реалістичність персонажів в іграх - це ретельний баланс, щоб уникнути ефекту “uncanny valley”. За допомогою нового інструменту MetaHuman і загальним удосконаленням систем анімації, персонажі, які населяють ігрові світи, також можуть зробити стрибок у покоління вперед, як показано на

(рис.1.11). Ці персонажі можуть слугувати чудовою основою для роботи художників по персонажах або допомагати заповнювати масу NPC у великих сценах.

Рисунок 1.11 – Використання MetaHuman в Unreal Engine 5 [11]

Нова технологія Unreal Engine 5 дозволяє дуже легко створювати реалістичних персонажів і містить безліч нових інструментів і оновлень, якими розробники можуть користуватися незалежно від типу гри, яку вони створюють. MetaHuman також допомагає оптимізувати етап анімації створення персонажа, що означає простіший і швидший конвеєр для всіх розробників.

Анімація

Є багато можливостей анімації, наданих UE4, які були перенесені або покращені в UE5: Animation Blueprints, Morph Targets, Retargeting, Root Motion та багато іншого. Але UE5 також пропонує нові інструменти, такі як Control Rig, Full-Body IK Solver і багато корисних вузлів, як показано на (рис.1.12).

Unreal Engine 5 розширив свій набір інструментів для анімації такими інструментами, як Control Rig, який дає змогу створювати та ділитися рігами між персонажами. Щоб створити більш природні рухи, ви можете зберегти та застосувати пози за допомогою програми Full-Body IK.

Control Rig — це універсальна система ріггінгу, яка дає змогу художникам і розробникам ефективно створювати розширені ріги персонажів і об’єктів для процедурної анімації та анімації на основі даних у рушії. Full-Body IK Solver може обробляти складну взаємодію між частинами тіла персонажа, полегшуючи створення анімацій, які адаптуються до різноманітних середовищ і ситуацій.

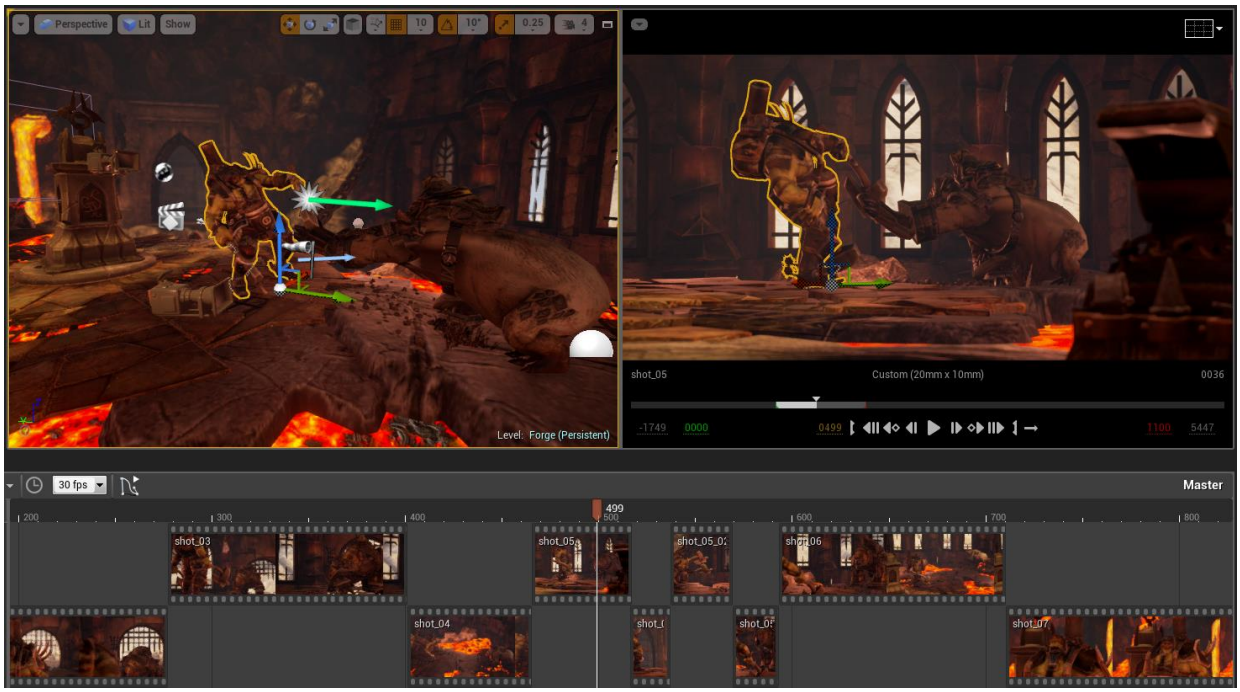


Рисунок 1.12 – Використання Sequencer в Unreal Engine 5 [12]

Іншою важливою функцією є Sequencer, який дає можливість створювати нелінійну анімацію та кінематографічні зображення. Хоча основні функції UE4 і UE5 залишаються подібними, покращення взаємодії з користувачем, продуктивності та інтеграції роблять версію UE5 більш надійним і спрощеним інструментом для створення анімації та кінематографії.

Також варто згадати багатошарову анімацію в Layered Animation. Ця функція має окремі доріжки в Animation Blueprint, якими можна індивідуально керувати та змішувати їх між собою. Кожен шар має певну анімацію або логіку, яка дозволяє зосередитися на різних частинах руху персонажа, наприклад, на верхній і нижній частині тіла.

Крім того, за допомогою таких інструментів, як Skeletal Editor і Panel Cloth Editor, аніматори можуть малювати ваги для скінінгу та додавати тканині більшої глибини, створюючи більш реалістичних персонажів та сцени.

MetaSounds

Що стосується звукового оформлення, UE5 представила нову аудіосистему MetaSounds. Вона використовує процедурний підхід на основі нодів, що дозволяє створювати складні динамічні звуки, які реагують на події в грі.

Крім того, MetaSounds також може синтезувати аудіо в реальному часі, що означає меншу залежність від попередньо записаних семплів і більше можливостей для процедурного та адаптивного аудіо, як показано на (рис.1.13). Крім того, UE5 має покращені функції просторового аудіо та акустичного моделювання, що робить ігровий аудіо-досвід ще більш захоплюючим і реалістичним.

Також, плагін Audio Synesthesia надає набір інструментів аналізу для вилучення інформації з аудіосигналів, які потім можна використовувати для керування ігровим процесом, візуальними ефектами чи іншими аспектами проекту.

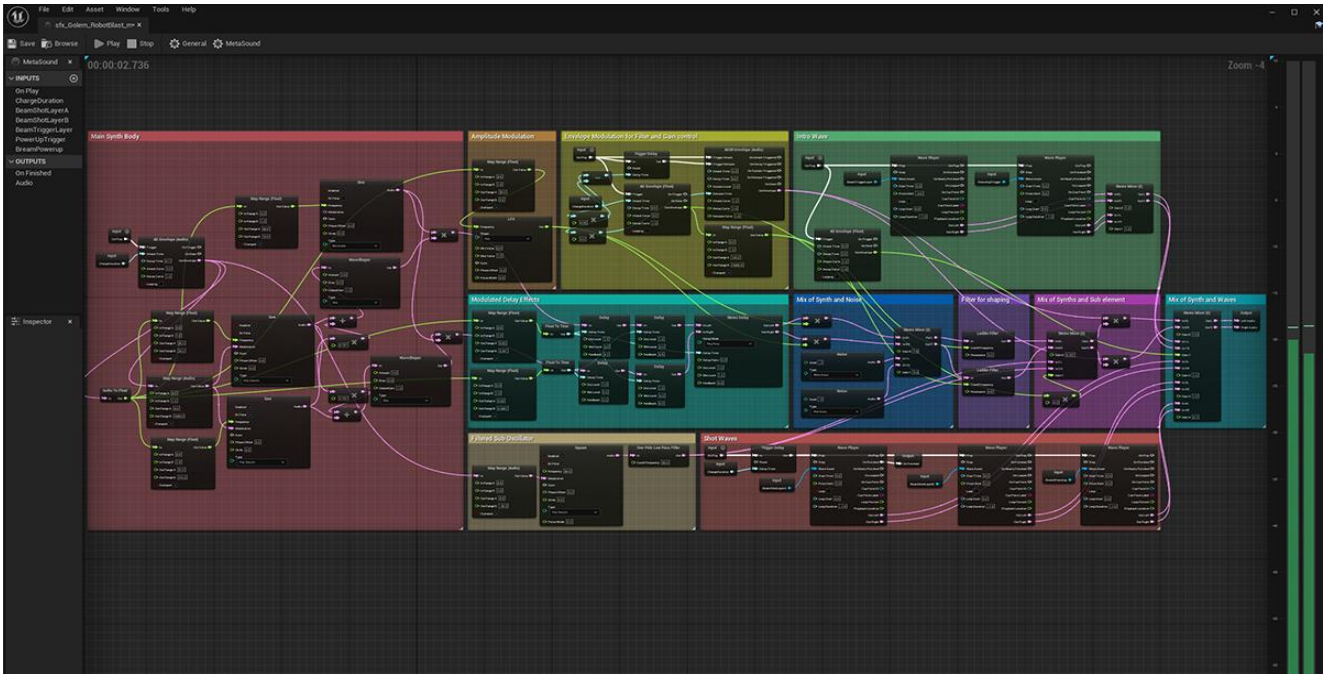


Рисунок 1.13 – Використання MetaSounds в Unreal Engine 5 [13]

1.2.2. Переваги Unreal Engine 5 над іншими рушіями

Технологія Unreal Engine 5 має вирішальне значення для консолей наступного покоління. UE5 також став критично важливим у створенні простих, потужних і масштабованих тривимірних світів для різноманітних галузей, включаючи архітектуру, трансляцію, події в прямому ефірі, автомобільну промисловість і транспорт, кіно та телебачення.

UE5 дозволяє файли більшого розміру

Unreal Engine 5 дозволяє збільшувати розміри файлів — і використовувати текстури 4K, 8K і 12K. Скажімо, гравець дивиться на печеру та бачить сталагміт. Він відтворюється в 4K. Але коли гравець збільшує зображення, роздільна здатність динамічно змінюється. І гравець бачить сталагміт у 12K.

Ця функція неймовірна для геймдеву і UE5 робить це можливим, обробляючи більші розміри файлів. Оскільки команди працюють над створенням ігор для консолей наступного покоління, таких як PlayStation 5 і Xbox Series S, їм знадобиться можливість обробляти великі файли.

Щоб реалізувати всі переваги Unreal Engine 5, розробникам і творцям ігор доведеться працювати з системою контролю версій, яка створює версії для будь-яких великих файлів і нескінченно масштабується.

UE5 підходить для виробництва фільмів і анімації

Технологічний прогрес Unreal Engine 5 також суттєво вплинув на індустрію кіно та розваг, зробивши внесок у понад 550 основних кінофільмів і телесеріалів.

Лише кілька років тому віртуальний продакшн використовували лише досвідчені режисери. Завдяки технології та можливостям Unreal Engine 5 віртуальний продакшн став доступним для більшої індустрії та викликав хвилю в анімаційних фільмах, телебаченні в стрімінгу, музичних заходах тощо.

Unreal Engine 5 надає творцям великий набір інструментів для віртуального продакшену, інструменти кінематографічного редагування та анімації під назвою

Sequencer, а також можливість створювати фінальний вихідний контент для свого продукту.

Кросплатформна розробка

Ще однією перевагою використання Unreal Engine є його підтримка кросплатформної розробки. Двигун дозволяє розробляти ігри для багатьох платформ, включаючи Windows, Mac, Linux, Android, iOS і консолі. Це може заощадити розробникам багато часу та зусиль, оскільки їм потрібно лише один раз написати код і розгорнути його на кількох платформах. Крім того, Unreal Engine пропонує вбудовану підтримку різних пристроїв введення, таких як геймпади, клавіатури та сенсорні екрани, що полегшує розробку ігор для різних пристроїв.

Зручний інтерфейс

Unreal Engine має зручний інтерфейс, який полегшує розробникам створення та зміну вмісту гри. Двигун поставляється з візуальною системою сценаріїв під назвою Blueprint, яка дозволяє розробникам створювати логіку гри без написання коду. Крім того, Unreal Engine пропонує потужний редактор рівнів, який дозволяє розробникам швидко й ефективно створювати та змінювати рівні гри. Редактор постачається з набором інструментів і функцій, таких як скульптурування ландшафту, малювання листя та редагування сітки, які допоможуть вам легко створювати складні ігрові середовища.

Візуальний сценарій Blueprint

Система Blueprint Visual Scripting в Unreal Engine надає розробникам ігор значну перевагу, дозволяючи їм створювати логіку гри без написання коду. Його інтерфейс перетягування дає змогу швидко створювати складну ігрову механіку, і він повністю інтегрований із рушієм, дозволяючи створювати все, від простих рухів гравця до складної поведінки ШІ.

Система базується на інтерфейсі на основі нодів у Unreal Editor і є повною системою сценаріїв ігрового процесу, як показано на (рис.1.14). Він функціонує

подібно до звичайних мов сценаріїв у визначенні об’єктно-орієнтованих класів або об’єктів у механізмі. Ця система є надзвичайно гнучкою та потужною, надаючи дизайнерам можливість використовувати майже повний спектр концепцій та інструментів, які зазвичай доступні лише програмістам.

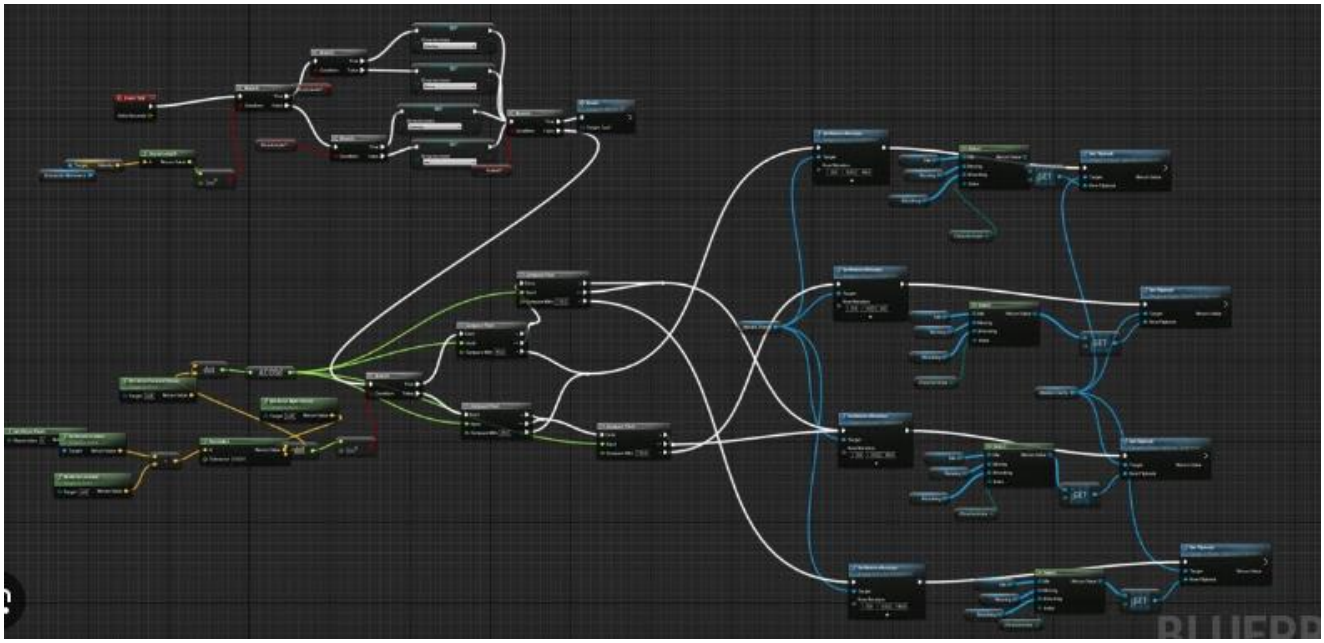


Рисунок 1.14 – Використання Blueprints в Unreal Engine 5 [14]

Крім того, реалізація Unreal Engine на C++ включає спеціальну розмітку Blueprint, що дозволяє програмістам створювати базові системи, які можуть бути розширені дизайнерами.

Підсумовуючи, система візуальних сценаріїв Blueprint у Unreal Engine є надзвичайно універсальним і потужним інструментом, який пропонує розробникам ігор швидший і простіший спосіб створювати складну ігрову механіку, не вимагаючи від них написання коду.

Unreal Engine Marketplace

Unreal Engine має магазин (або ринок), який надає розробникам ігор доступ до широкого спектру ассетів, інструментів і плагінів, як показано на (рис.1.15). Цей ринок пропонує різноманітні ассети, включаючи 3D-моделі, текстури, звукові

ефекти та анімацію, які допомагають розробникам швидко створити високоякісну гру. Крім того, ринок пропонує ряд інструментів і плагінів, таких як аналітика ігор, багатокористувацькі фреймворки та інструменти оптимізації продуктивності, які можуть допомогти розробникам створити кращу та успішнішу гру.

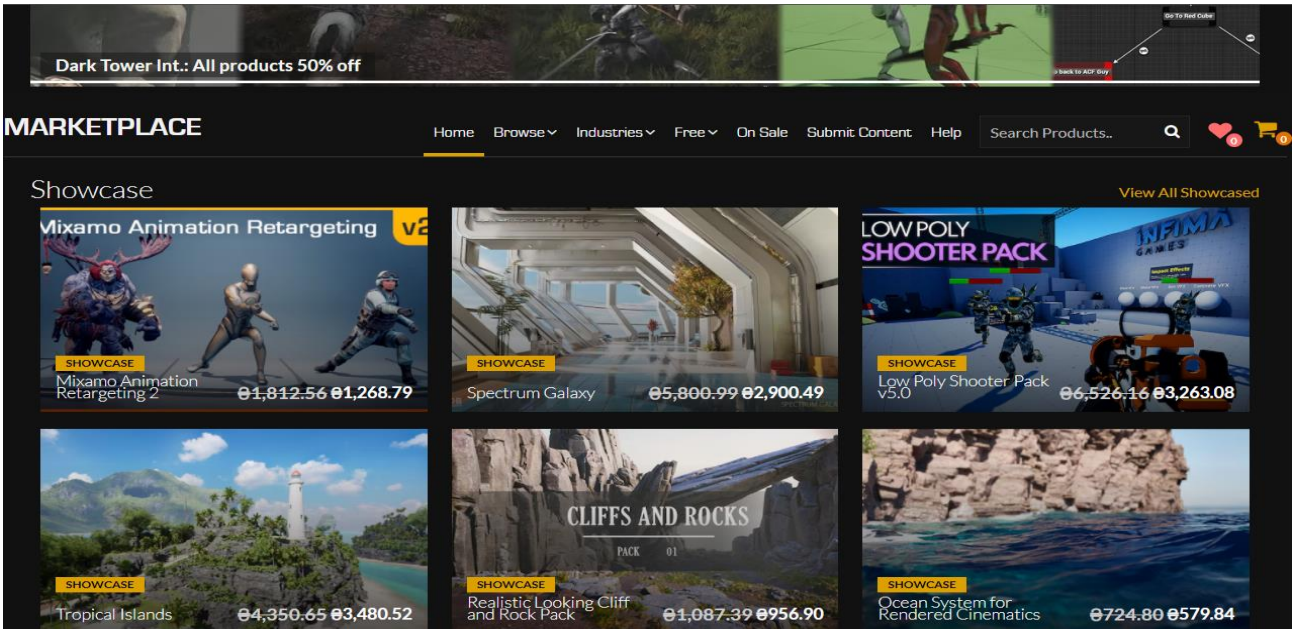


Рисунок 1.15 – Сторінка магазину Unreal Engine [15]

Висновки до розділу: Unreal Engine — це потужний ігровий рушій, який пропонує різноманітні переваги для розробників ігор. Він забезпечує високоякісну графіку, кросплатформну розробку, зручний інтерфейс, розширену документацію та підтримку спільноти, розширене моделювання фізики, швидке створення прототипів, масштабованість, візуальний сценарій Blueprint, ринок і розробку, керовану спільнотою[16].

Вміщаючи в собі такий революційний інструментал функцій, як Nanite, Lumen, Virtual Shadow Maps, Chaos Physics, World Partition, MetaHuman, MetaSounds та інші, Unreal Engine 5 має що запропонувати незалежно від того, чи ви любитель, незалежний розробник або велика AAA студія. Це відмінний вибір для тих, хто хоче створити якісну та успішну гру.

2 СТВОРЕННЯ І ОПТИМІЗАЦІЯ ІГРОВОЇ 3D МОДЕЛІ

Швидкі темпи розвитку графічних технологій та поява потужних графічних рушіїв, таких як Unreal Engine 5, створюють нові вимоги до оптимізації 3D моделей. Оптимізація - це необхідний етап у процесі розробки, оскільки вона визначає ефективність відтворення графіки та впливає на загальну продуктивність гри.

Оптимізація 3D моделей для Unreal Engine 5 є невід'ємною частиною розробки ігор, спрямованою на підвищення продуктивності та забезпечення високоякісного геймплею. Цей процес вимагає ретельного підходу та використання різноманітних технік з метою досягнення оптимального балансу між високою якістю графіки та ефективною роботою гри на різних пристроях.

Однією з ключових цілей оптимізації є забезпечення стабільної кадрової частоти та плавного геймплею. Зменшення кількості полігонів на моделі сприяє швидшому відтворенню в сцені та кращому досвіду гравця.

Оптимізація також дозволяє ефективніше використовувати оперативну та графічну пам'ять, що особливо важливо для гравців із менш потужними пристроями. Оптимізовані моделі дозволяють грі працювати ефективно на різних пристроях, включаючи консолі, ПК та мобільні платформи. Зменшення обсягу моделей сприяє швидшим переходам між сценами та завантаженню гри в цілому[17].

2.1 Обґрунтування вибору програмного пакету для створення 3D моделі.

У галузі геймдеву використовуються різноманітні програми для створення 3D моделей, серед яких важливими є Autodesk Maya, Blender, Autodesk 3ds Max, ZBrush та Cinema 4D. Кожна з цих програм має свої унікальні особливості та інструменти для моделювання, текстурування, анімації та інших аспектів роботи з тривимірною графікою.

Autodesk Maya традиційно визнана стандартом індустрії та забезпечує широкі можливості для створення складних анімованих об'єктів та персонажів.

ZBrush використовується для скульптурного моделювання та деталізації, тоді як Cinema 4D найбільше використовується в галузі візуальних ефектів та анімації.

Blender, як безкоштовна програма з відкритим кодом, стає популярним вибором завдяки своїй безкоштовності та активній спільноті.

В сучасній індустрії геймдеву, вибір програмного пакету для створення 3D-моделей є ключовим етапом розробки гри. Однією з найбільш захоплюючих та ефективних альтернатив для цієї мети стає Blender.

Однією з ключових переваг Blender є його безкоштовність та відкритість коду. Це робить його доступним для широкого кола користувачів, незалежно від їхнього бюджету. Геймдевелопери можуть використовувати Blender безкоштовно, що важливо, особливо для невеликих студій або початківців.

Blender вражає своєю потужністю та гнучкістю в галузі моделювання. Його інструменти дозволяють вам створювати складні 3D-моделі з високою деталізацією. Від полігонального моделювання до скульптурного режиму, Blender забезпечує широкі можливості для втілення творчих ідей та концепцій геймдевелоперів.

Blender також відзначається своїми розширеними можливостями текстурування та створення матеріалів. З інструментами, які дозволяють легко накладати текстури та створювати реалістичні матеріали, Blender стає важливим інструментом для створення об'єктів та персонажів у графіці гри.

Blender володіє потужними засобами для анімації об'єктів та персонажів, що робить його ідеальним інструментом для розробників ігор. Інтеграція з різними ігровими двигунами, такими як Unity чи Unreal Engine, дозволяє легко і ефективно використовувати створені в Blender моделі в ігрових проектах[18].

Blender має велику та активну спільноту користувачів, що сприяє обміну знаннями та розвитку програми. Постійні оновлення та вдосконалення забезпечують користувачам доступ до новітніх технологій та функцій.

Blender виявляється не лише потужним інструментом для створення 3D-моделей, але й доступним для широкого кола користувачів. Його безкоштовність, розширені можливості моделювання та текстурування, а також активна спільнота роблять його перспективним інструментом для геймдевелопменту.

Через регулярні оновлення з додаванням унікальних функцій, безліч безкоштовних аддонів та універсальність, Blender все частіше використовують, як маленькі інді-проекти, так і великі ААА студії. Тому можна сміливо сказати, що такими темпами саме Blender стане широковизнаним стандартом індустрії і замінить пакет Autodesk.

Цей програмний пакет не тільки задовольняє потреби досвідчених розробників, але і відкриває двері для нових талантів, що робить його найкращим вибором для розвитку та створення унікальних ігор.

2.2 Визначення технічних рамок.

Створення будь-якої моделі починається з ідеї та концепту. Перед тим, як почати моделювати, треба зрозуміти що саме ми будемо моделювати, в якому стилі це треба робити, де ця модель буде використовуватись та яких технічних параметрів потрібно дотримуватись. В даному випадку була поставлена задача створити ігрову модель середньої складності у фотореалістичному стилі.

У кожної моделі є свої бажані технічні параметри в залежності від використання. Це кількість полігонів або трикутників (polycount), розмір текстури, тексель та формат експорту текстур[19].

2.2.1 Кількість полігонів

Кількість полігонів або трикутників – перше, на що звертається увага під час аналізу і тестування моделі. Оптимальна кількість трикутників визначається для кожної моделі індивідуально[20]. Наприклад, для моделі оточення середнього розміру (1-2м в одній з осей) оптимальним є діапазон в 5000-10000 трикутників в залежності від складності і деталізації, як показано на (рис. 2.1).

Для дрібних моделей, наприклад, столових приборів або ламп, відводиться набагато менше трикутників, приблизно від сотні до тисячі в залежності від розміру моделі, як показано на (рис. 2.2).

Рисунок 2.1 – Приклад моделі в діапазоні 5-10к трикутників

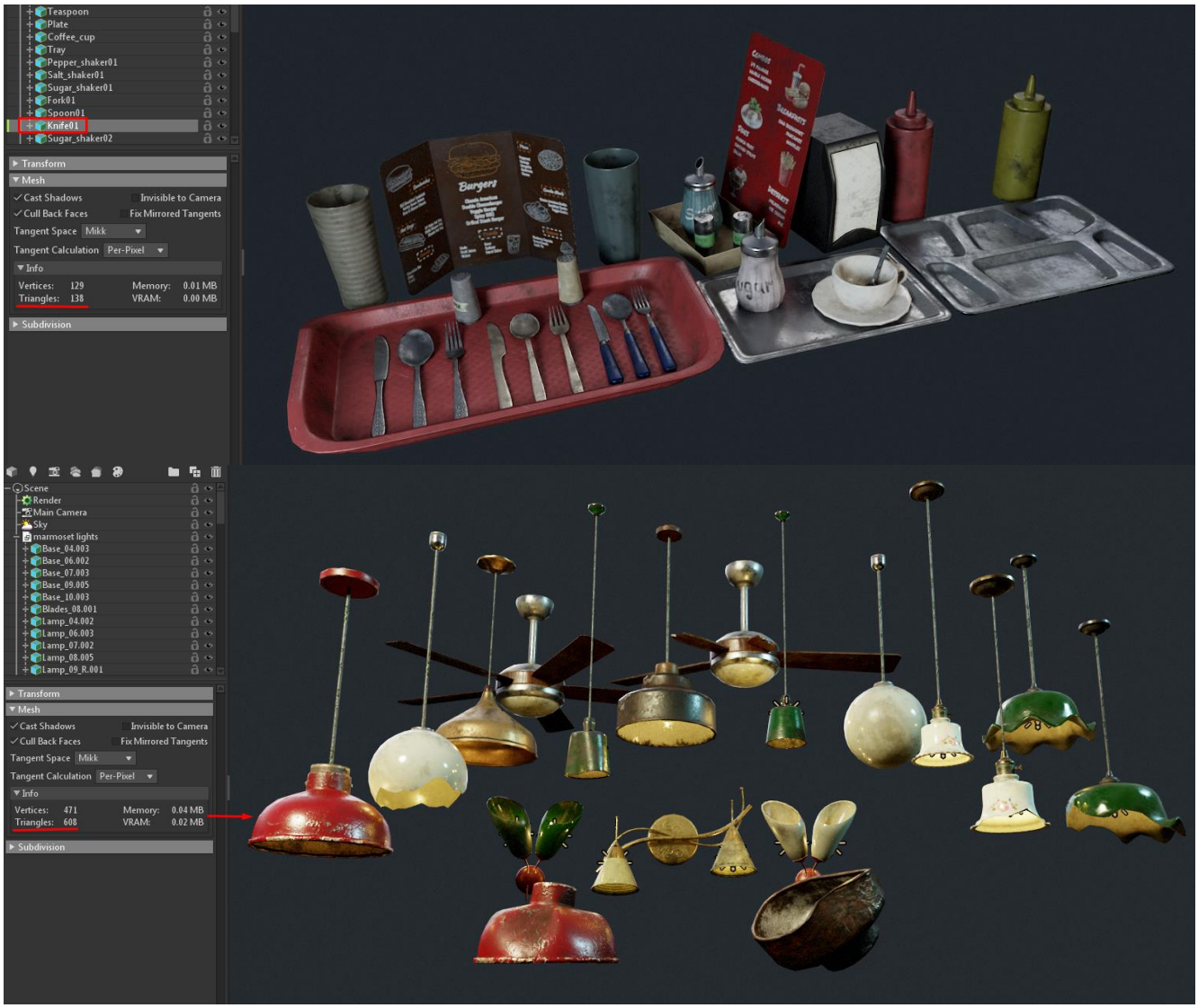


Рисунок 2.2 – Приклад моделей в діапазоні до 1000 трикутників

Якщо модель буде в руках у гравця, або гравець буде мати можливість роздивитись її з ретельно і з кожного кута, то таку модель називають Hero Prop, в такому випадку полікаунт дозволяють збільшити, поки модель не буде відповідати технічному завданню, як показано на (рис. 2.3). Полікаунт (англ. Polycount) – кількість полігонів на моделі. В ігровій індустрії, частіше – трикутників на моделі.

Рисунок 2.3 – Приклад Hero Prop моделі в діапазоні до 5000 трикутників

2.2.2 Розмір текстури

Розмір текстури визначає роздільну здатність фінальної карти текстури. Розміри текстури можуть бути будь-якими, але найчастіше використовують квадратні текстури з роздільною здатністю від 1024 (або 1К) і аж до 8192 пікселей (або 8К). В ААА проектах, 8К текстури використовують для дуже великих об’єктів, або для персонажів[21].

Для моделей оточення та пропсів використовують текстури 1К для маленьких моделей, 2К для середніх, як показано на (рис. 2.4) та 4К для великих моделей, як показано на (рис. 2.5) або для моделей, які персонаж буде тримати в руках. Якщо модель завелика для 2К, але замала для 4К текстури, то поширеною практикою є присвоєння моделі двох карт по 2К, як показано на (рис. 2.6).

Рисунок 2.4 – Приклад моделі з 2К текстурою

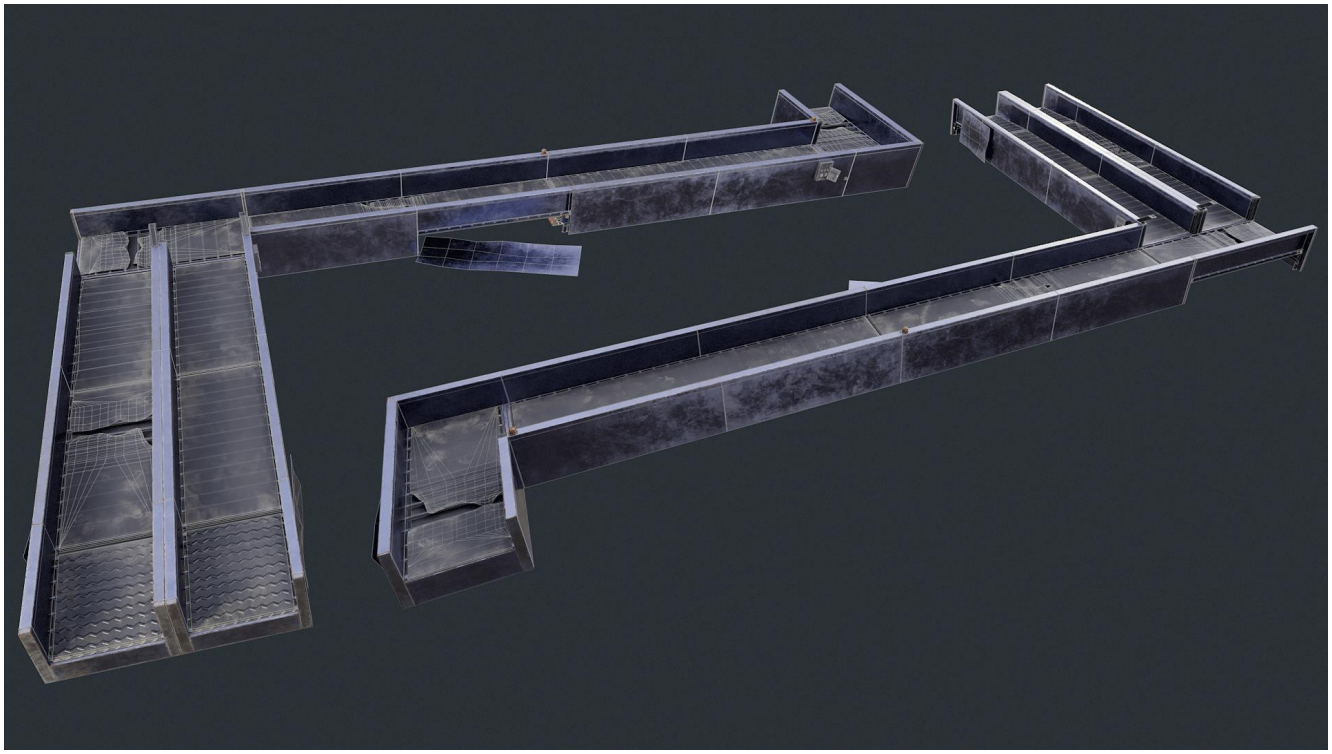


Рисунок 2.5 – Приклад моделі з 4К текстурою

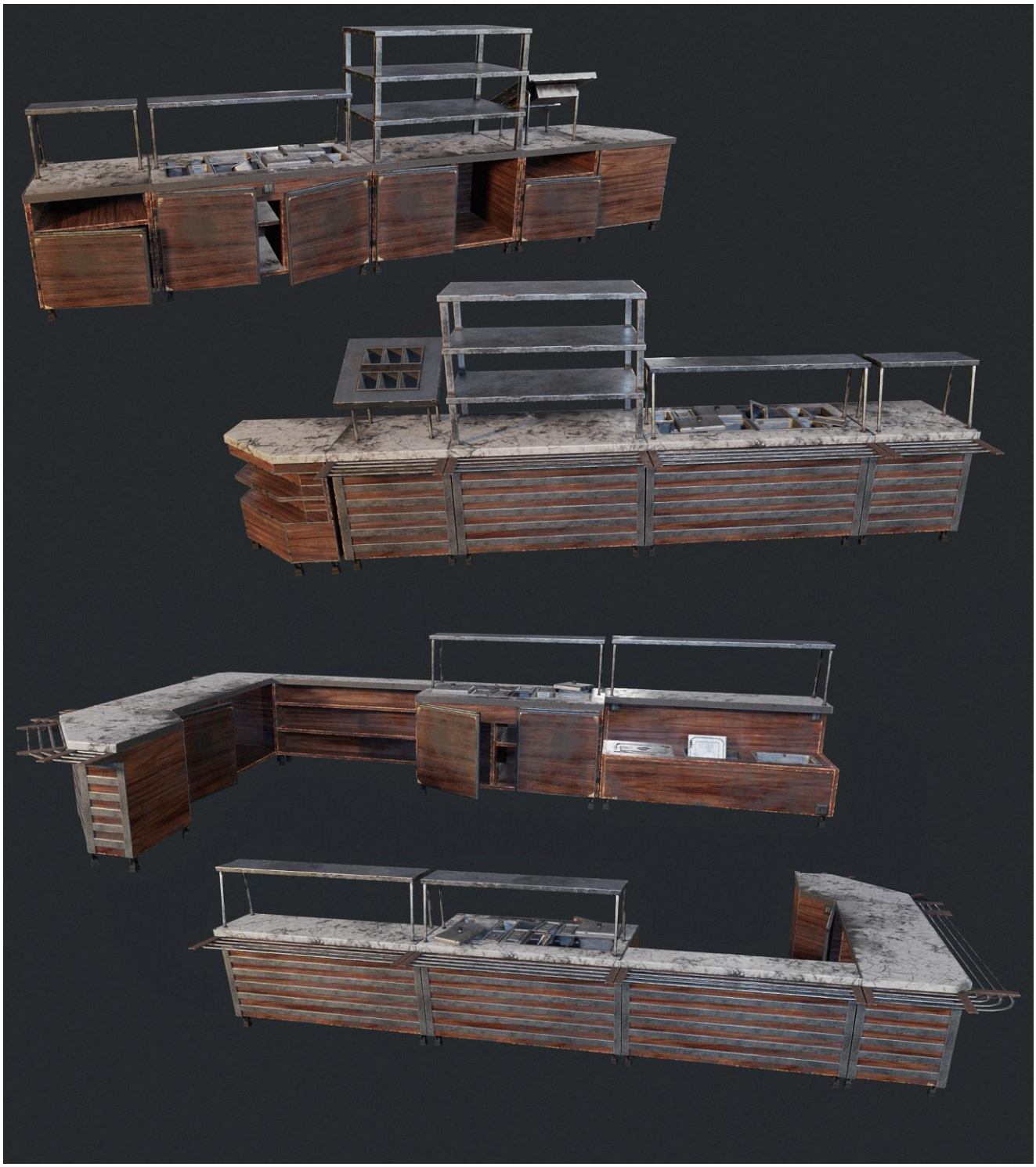


Рисунок 2.6 – Приклад моделей з двома 2К текстурами

2.2.3 Щільність текселю

Щільність текселю або texel density визначає кількість пікселів на метр на готовій моделі. Щільність текселю залежить від роздільної здатності (розміру) текстури та щільності пакування розгортки. Для кожного проекту щільність текселю

визначається індивідуально і ця щільність зберігається і застосовується для абсолютно всіх моделей в грі[22].

Існують тільки два виключення коли можна виходити за узгоджений тексель в проекті: якщо ділянку моделі не буде видно гравцю (дно бочки чи внутрішня частина дула пушки), тоді можна зменшити тексель щоб дати більше місця на розгортці більш важливим деталям. Або навпаки, якщо модель буде використовуватись гравцем близько до камери, її будуть брати в руки і розглядати, тоді тексель потрібно збільшувати в 1.5-2.5 разів.

2.2.4 Щільність пакування

Щільність пакування – це міра заповненності UV розгортки UV шелами моделі. Для оптимального використання текстури потрібно досягати мінімум 75% заповненності розгортки.

2.2.5 Формат експорту текстур

Формат експорту текстур визначає фінальний вигляд і кількість експортованих файлів текстур.

Під час текстурування виділяють такі основні текстури:

Diffuse – текстура, яка відображає основний колір моделі.

Metallic – передає металічність об’єкту в чорно білому спектрі.

Roughness – передає шорсткість моделі в чорно білому спектрі (наскільки гладкий або потертий об’єкт)

Normal – карта нормалей використовується для змоделювання деталей освітлення на поверхні об'єкта. Вона містить інформацію про напрямки нормалей (векторів, перпендикулярних до поверхні) у кожній точці поверхні.

Height – карта висот використовується для змоделювання висоти точок поверхні. Кожен піксель відображає висоту поверхні відносно базового рівня.

Ambient Occlusion – карта оклюзії використовується разом із зображенням для моделювання затінення та підвищення контрастності в областях, де поверхні зближені або де знаходяться у закутках. Це допомагає надати зображенню більш реалістичний вигляд, оскільки воно враховує більше фізичних особливостей освітлення та тіней.,

Opacity – карта передачі прозорості об’єкту. Наприклад, використовується щоб зробити прозоре скло або вирізати отвори в сітці чи паркані.

Emissive – визначає області текстури чи матеріалу, які випромінюють світло незалежно від освітлення сцени. Ця карта визначає, які частини об'єкта світяться самі по собі і не залежать від зовнішнього освітлення[23].

Під час експорту, можна обирати які саме карти зберегти та як саме їх зберегти. Популярним методом експорту карт в геймдеві є метод ORM пакування.

Пакування цих трьох карт у одну текстуру (ORM texture) має кілька переваг:

Один текстурний об'єкт замість трьох може зменшити використання пам'яті, що важливо для оптимізації продуктивності гри, особливо на мобільних пристроях та інших обмежених платформах.

У сучасних графічних двигунах, зокрема, у техніці Deferred Rendering, кількість текстурних зразків може впливати на продуктивність. Пакування карт у одну зменшує кількість зразків.

Використання одного текстурного об'єкта для ORM спрощує роботу художників та дизайнерів, оскільки їм потрібно керувати лише однією текстурою для визначення властивостей матеріалів.

ORM-текстури допомагають зберігати важливу інформацію про фізичні властивості матеріалів у більш ефективний спосіб, зменшуючи вплив на продуктивність і забезпечуючи високий рівень деталізації у візуальних ефектах.

А якщо така текстура буде використовуватися в десятках матеріалів - це полегшить роботу в цілому і знизить навантаження на оперативну пам’ять.

2.2.6 Обрані технічні рамки

Для даної моделі було обрано наступні технічні параметри:

Кількість трикутників: <8000

Розмір текстури: 2048*2048

Тексель: >500 px/m

Щільність пакування: >75%

Формат експорту текстур: ORM Packed.

2.3 Створення 3D моделі

Для визначення конкретного концепту, були використані сайти Google, Pinterest та ArtStation.

2.3.1 Вибір концепту

Було обрано зробити модель старовинної гармати з музею Хотинської фортеці, як показано на (рис.2.7).

Рисунок 2.7 – Обраний приклад гармати з реального життя [14]

Цей вибір обґрунтовується достатньою кількістю деталізації, цікавістю і різноманітністю текстур та можливістю використання моделі у патріотичній обстановці гри про українську історію.

2.3.2 Створення мудборду в PureRef.

Наступним етапом підготовки є створення мудборду. Мудборд (англ. Moodboard) – набір прикладів для роботи, організованих за спільними аспектами цільового продукту. Мудборди використовують для більш точнішого розуміння деталізації та текстури моделі.

Найпопулярніша програма для створення мудбордів – PureRef, як показано на (рис.2.8). Вона є безкоштовною та інтуїтивною, тому вона підходить як для досвідчених художників, так і для початківців.

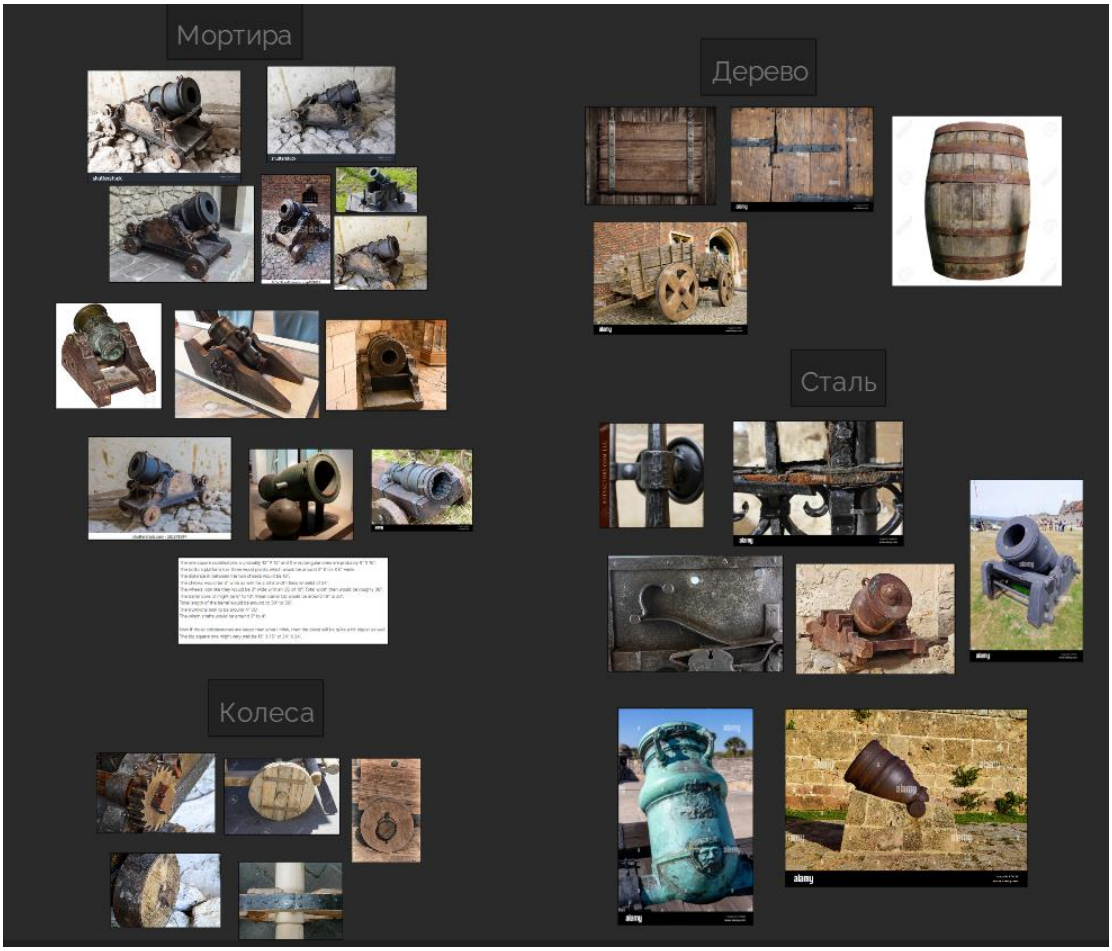


Рисунок 2.8 – Створений мудборд в PureRef

За допомогою пошукових ресурсів, було підібрано декілька референсів на кожен аспект майбутньої моделі: сама мортира, колеса та дрібні деталі, а також текстури дерева та сталі.

Референс (англ. Reference) – приклад аспекту цільового продукту. Використовується як приблизний орієнтир, або чітка ціль в роботі. Всі знайдені референси були зібрані в один мудборд в програмі PureRef.

2.3.3 Блокінг моделі в Blender

Наступний етап створення моделі – блокінг (blocking). Блокінг – це приблизний тривимірний “нарисок” майбутньої моделі. Блокінг роблять для виставлення і узгодження форми та розмірів моделі, як показано на (рис.2.9). Блокінг робиться в 3D редакторі, в якому будуть робитись наступні етапи моделювання, в нашому випадку – це Blender.

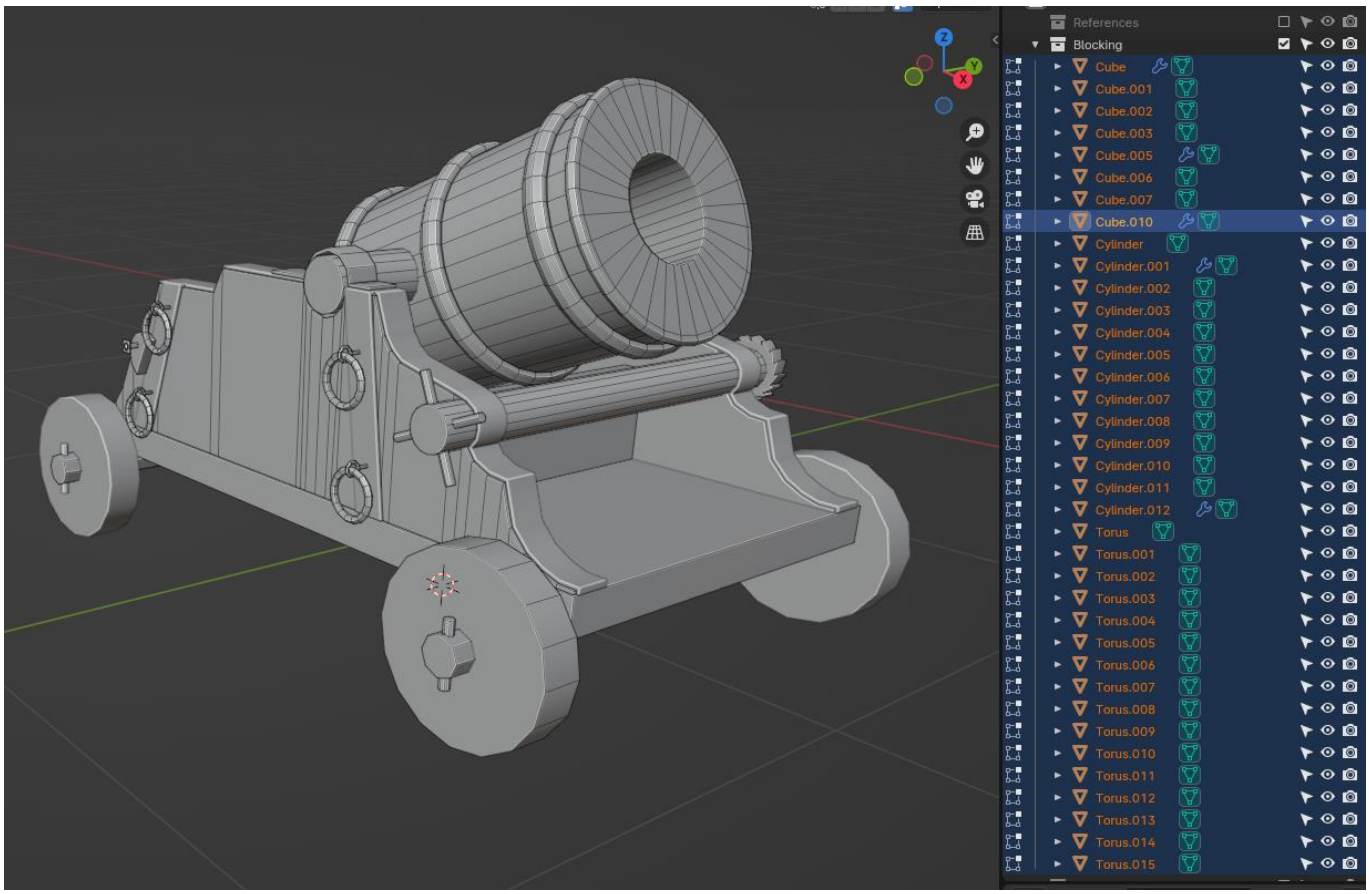


Рисунок 2.9 – Виконаний блокінг моделі

Було виконано приблизне відтворення неоптимізованої модель мортири. Блокінг не повинен мати в собі велику кількість деталей або оптимізовану полігональну сітку, його основна ціль – приблизно передати форму і концепт майбутньої моделі.

2.3.4 Лоуполі моделювання

Наступним етапом моделювання є створення низькополігональної або лоуполі (LP) моделі. На цьому етапі художник повинен із блокінгу зробити модель, готову до імпорту в ігровий рушій, як показано на (рис.2.10).

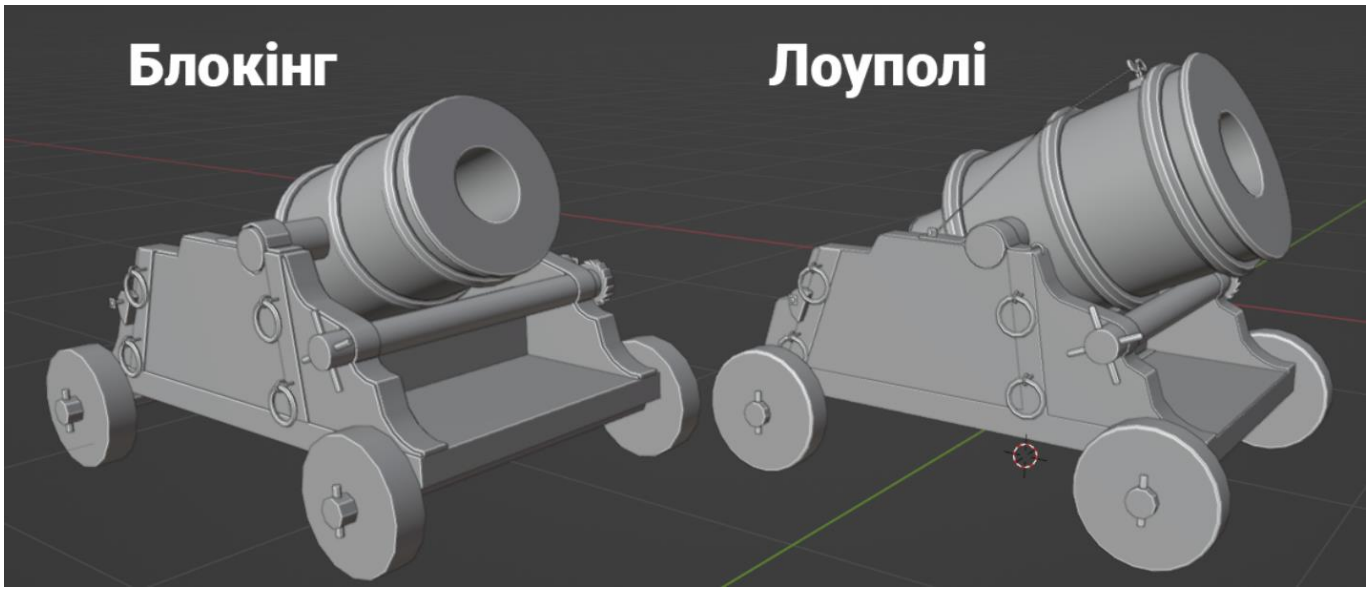


Рисунок 2.10 – Різниця між етапами блокінга і лоуполі

Тобто, на цьому етапі виправляються помилки блокінга та додається максимальна геометрична деталізація фінальної моделі (заклепки, мотузки, структурні елементи). Дуже важливо передати реалістичність моделі. Якщо модель має дві дошки прибиті одна до одної, то на моделі (або хоча би на текстурі) треба показати як саме вони тримаються разом, наприклад, гвіздками або планками з заклепками, як показано на (рис.2.11).

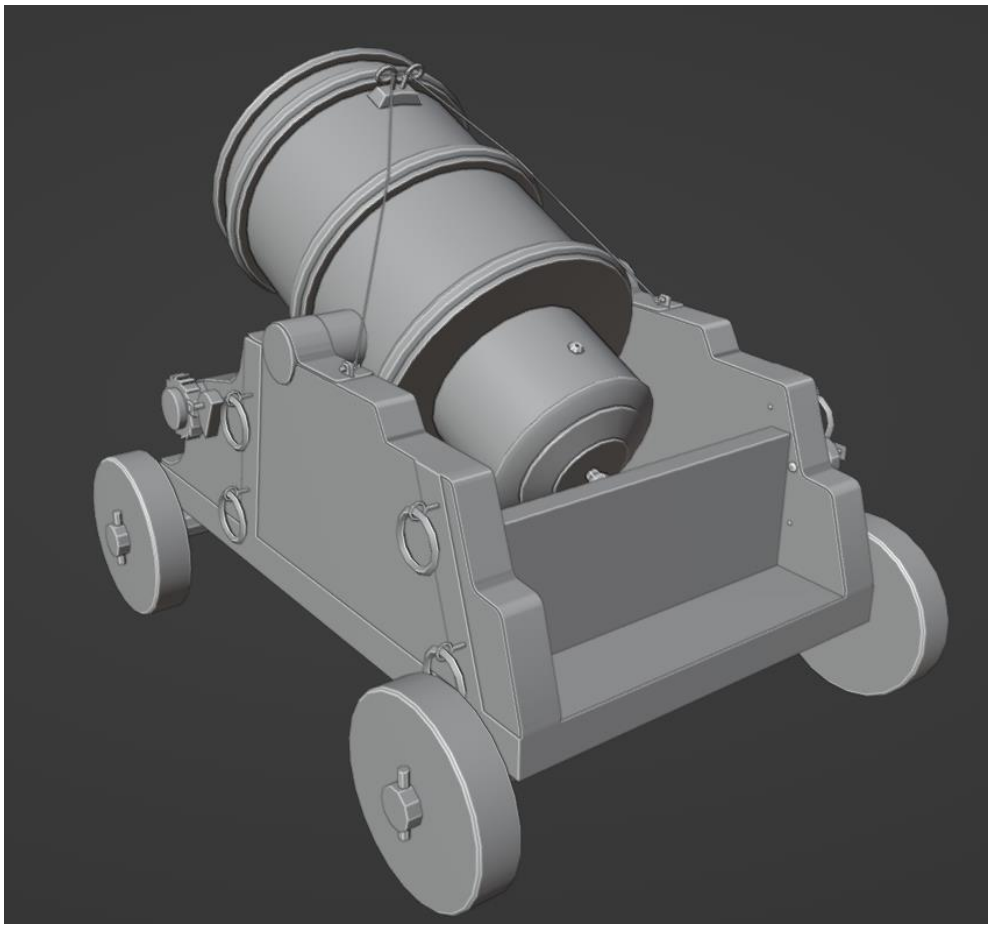


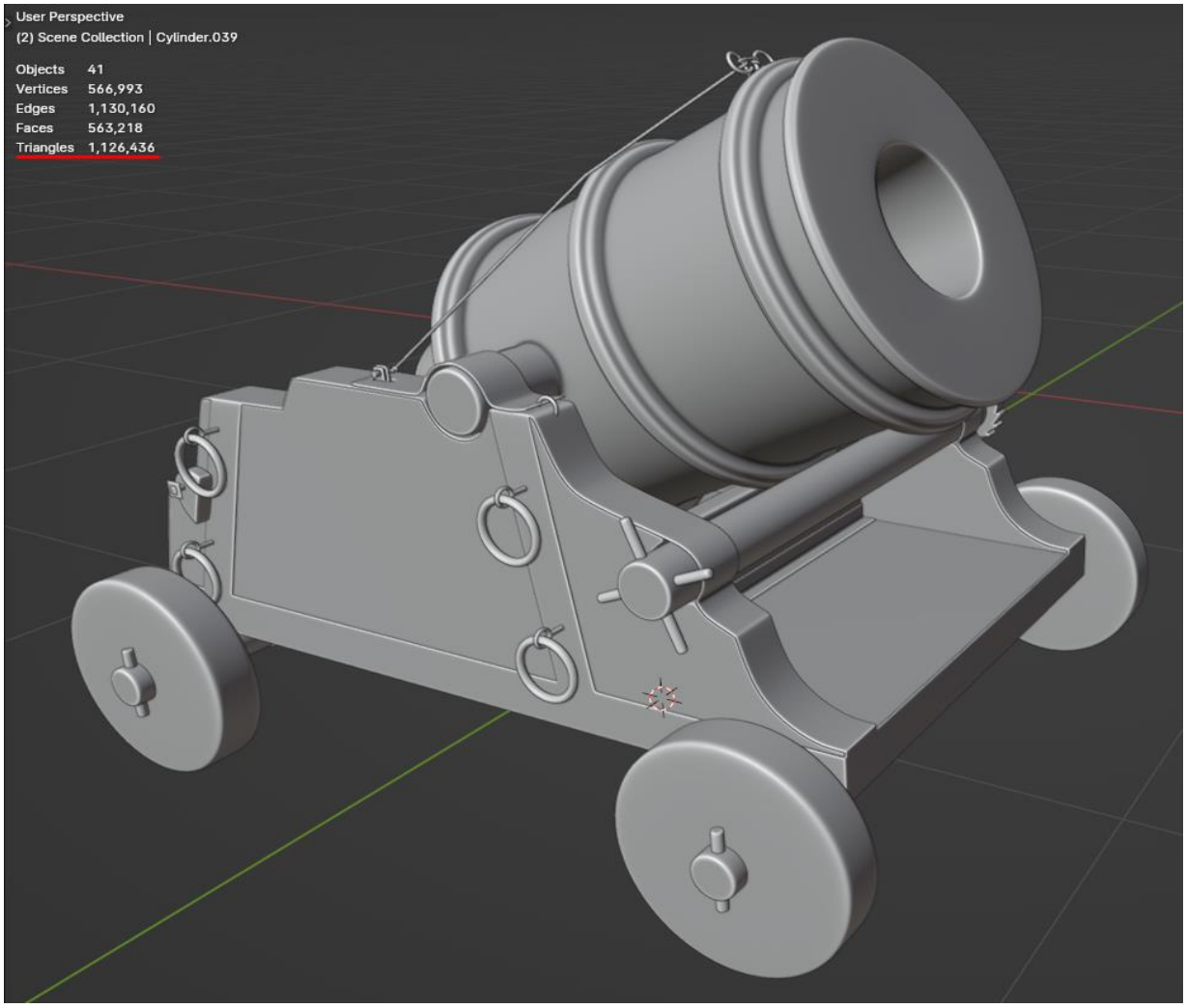
Рисунок 2.11 – Зворотня частина лоуполі моделі

2.3.5 Створення хайполі моделі

Highpoly модель (HP) створюється для того, щоб передати деталізацію на низькополігональну модель. Тому, під час створення хайполі, треба досягти максимально деталізованого, відпрацьованого вигляду (з м’якими кутами, всіма гвіздками і болтами, виїмками, тощо), як показано на (рис.2.12).

Отриману модель не можна використовувати в ігровому рушії по двом причинам. По-перше, її полікаунт зависокий, що погано відобразиться на продуктивності гри. По-друге, таку високополігональну модель буде надзвичайно складно і непрактично розгортати на UV, а потім імпортувати в Substance Painter.

Тому ця модель буде використовуватись лише для передачі своєї геометричної інформації на карту нормалей лоуполі моделі.



Риунок 2.12 – Створена хайполі модель з кількістю трикутників більше мільйона

2.3.6 Оптимізація лоуполі моделі

На цьому етапі буде створений меш для імпорту в Substance Painter та безпосереднього використовування в ігровому рушії, тому перед його створенням треба ретельно продумати способи його оптимізації.

Оптимізація моделі зазвичай ділиться на два етапи: оптимізація геометрії і оптимізація розгортки.

Оптимізація геометрії виконується за рахунок видалення, заміни або зведення непотрібних полігонів, які не несуть вирішальної ролі і не визначають форму об’єкту.

2.3.7 Основні техніки оптимізації геометрії

Зведення радіусів

Зведення радіусів – виконується на місцях, де коло великого радіусу веде до кола меншого радіусу. Задля збереження рівномірного вигляду на радіусах, зайві еджи слід зводити разом, щоб зберегти відносно однакову довжину фінального еджа , як показано на (рис.2.13). Едж (англ. Edge) – пряма лінія, що з’єднує дві точки. Три і більше еджа створюють фейс. Фейс (англ. Face)– назва примітиву багатокутника в програмному забезпеченні Blender.

Якщо це не робити, то, по-перше, на моделі залишаться зайві полігони, які не покращують загальний вигляд моделі, по-друге, фінальна модель буде мати нерівномірну щільність геометрії, що буде привертати більшу увагу до менш щільних областей моделі, як показано на (рис.2.14).

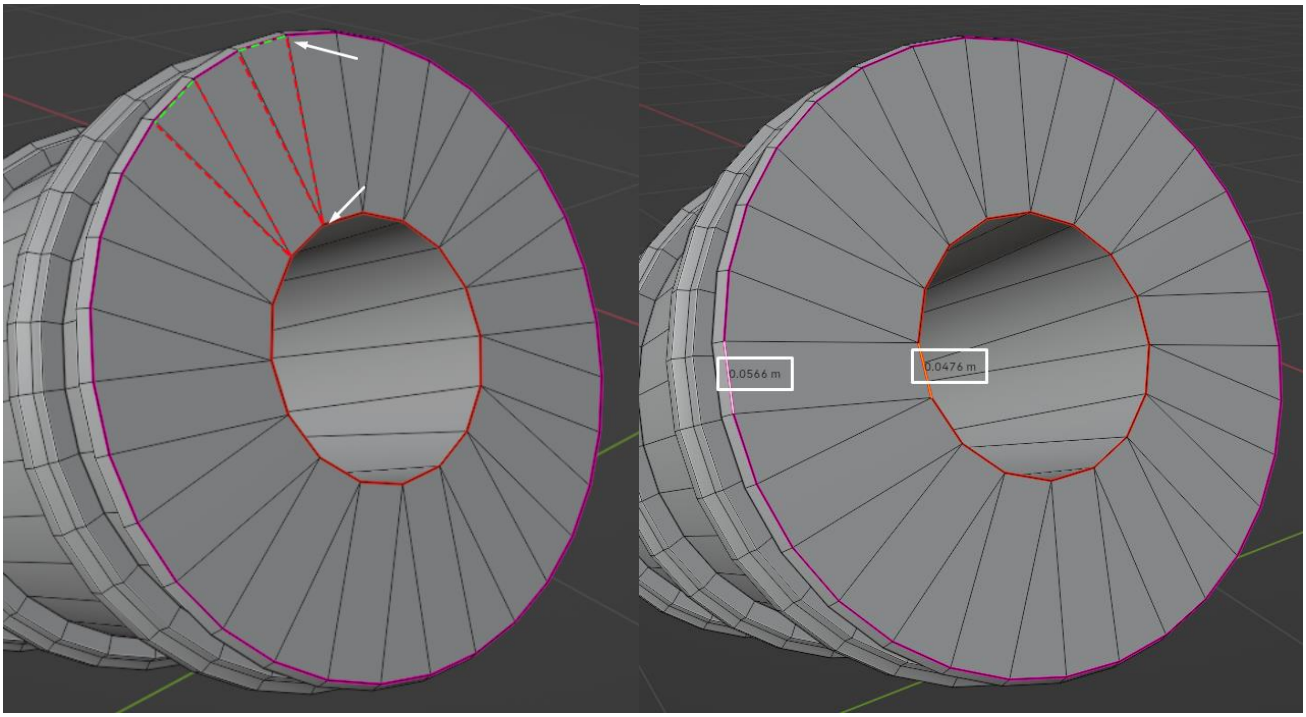


Рисунок 2.13 – Техніка зведення радіусів (зліва) з перевіркою довжини еджів (праворуч)

Зведення в одну точку на пласких фейсах

Зведення в одну точку на пласких фейсах – виконується на ділянках моделі з пласкими фейсами, які не будуть піддаватися деформації. Це поширена практика в індустрії в цілому, а у сфері hard-surface моделювання – правило оптимізації.

Ця техніка дуже ефективно позбувається зайвих трикутників, на нашому прикладі, лише ця деталь була оптимізована майже в два рази – зменшення від 100 до 61 трикутника , як показано на (рис.2.15).

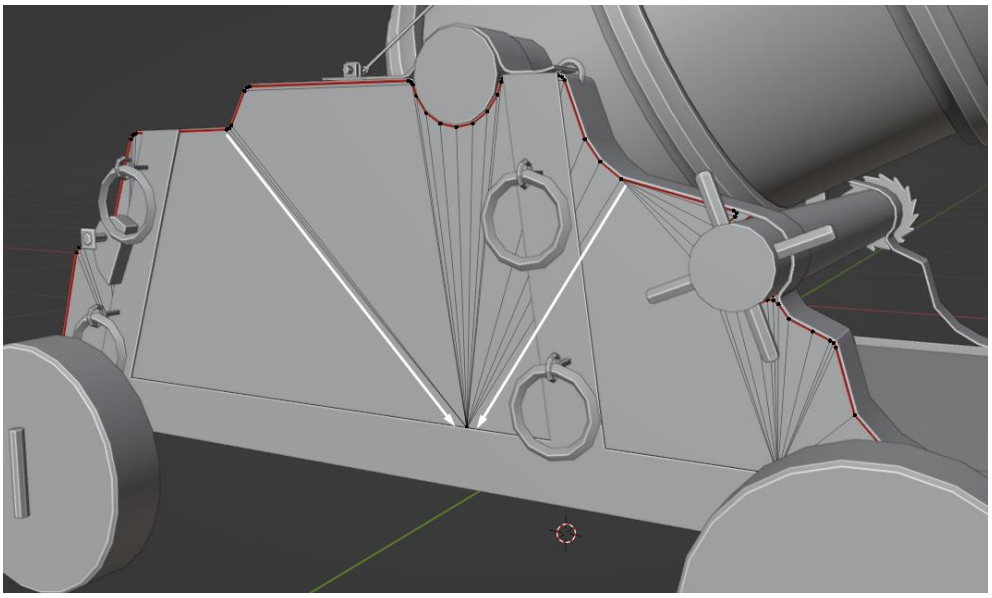


Рисунок 2.14 – Виконання зведення в одну точку на пласкому фейсі

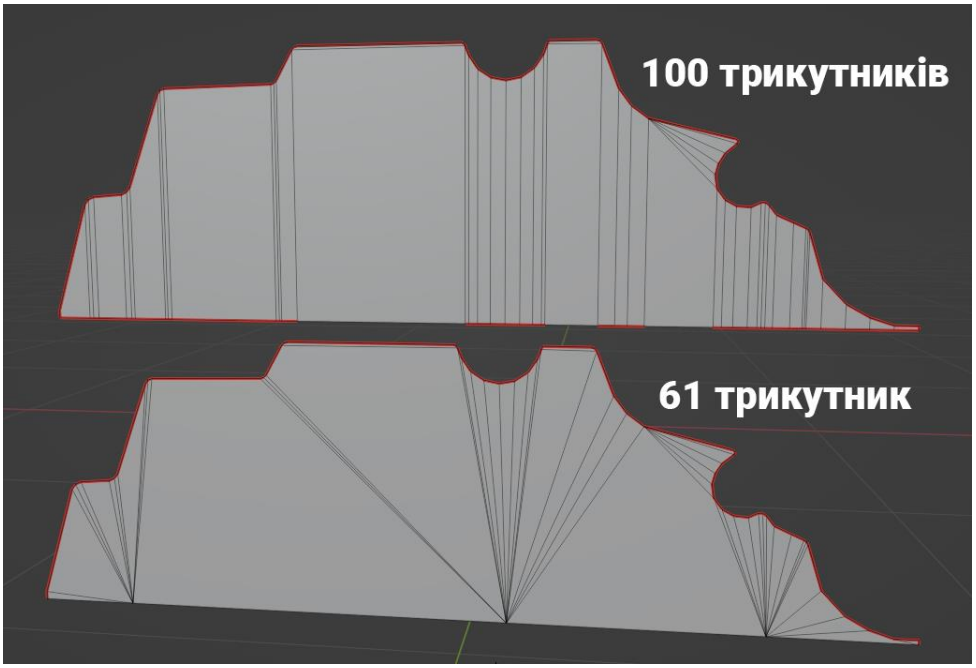


Рисунок 2.15 – Порівняння ефективності зведення в одну точку на пласкому фейсі

Розбиття на деталі

Розбиття на деталі – дозволяє розбити складну деталь на простіші малі деталі. Це дозволяє використовувати простішу топологію для кожної деталі, як показано на (рис.2.16), а не намагатись зробити все одним суцільним мешем.

Ця техніка – є основним правилом створення комплексної геометрії на моделях з великою кількістю деталізації, але попри це, її дуже рідко використовують художники початківці, чим і ускладнюють собі процес моделювання в декілька разів.

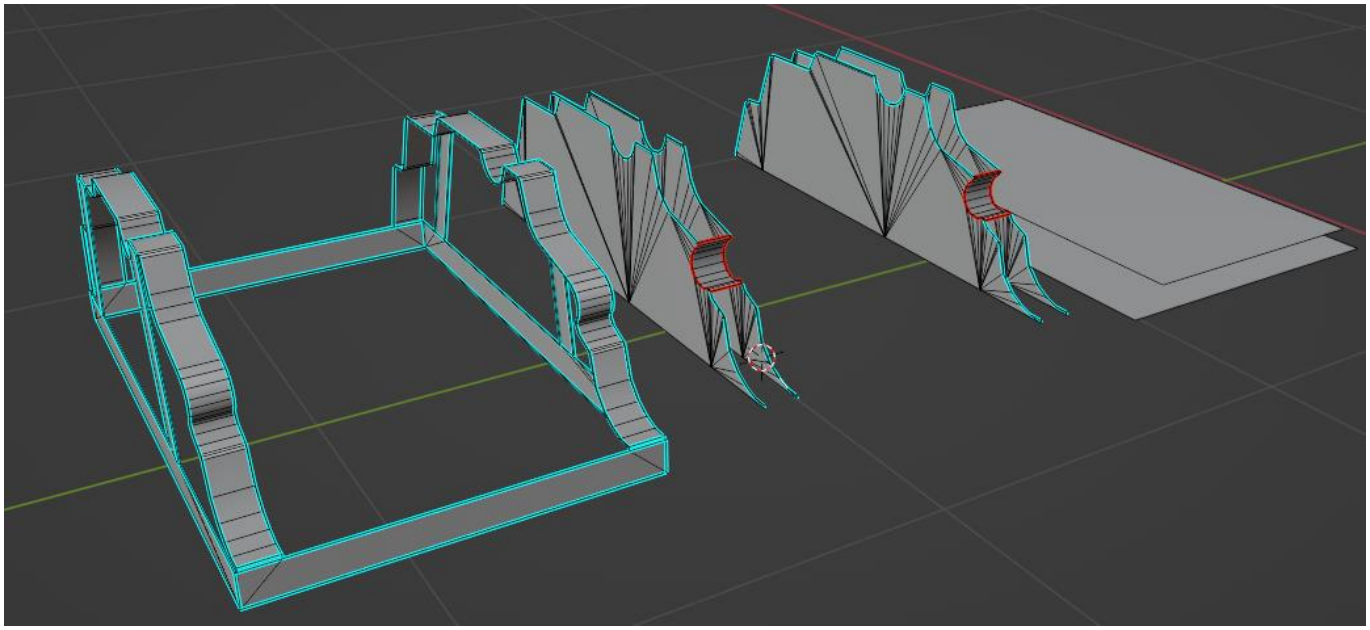


Рисунок 2.16 – Виконання розбиття моделі на деталі

Видалення задніх фейсів

Видалення задніх фейсів – техніка видалення полігонів, прихованих іншими полігонами. Інколи на моделях зустрічаються ділянки полігонів, які гравець не буде бачити на фінальній моделі, наприклад, сторони дошок, на стиках з іншими дошками, як показано на (рис.2.17, 2.18).

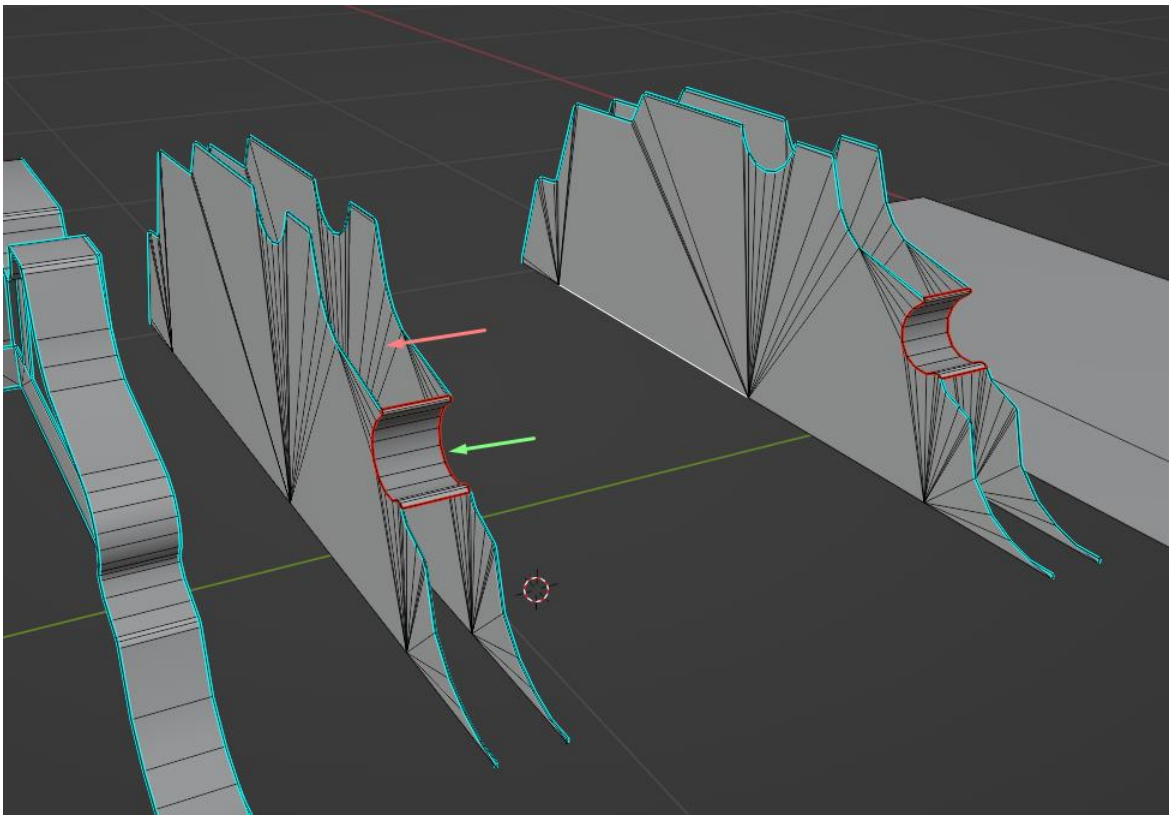


Рисунок 2.17 – Видалення задніх фейсів (червоним) і збереження видимих фейсів (зеленим)

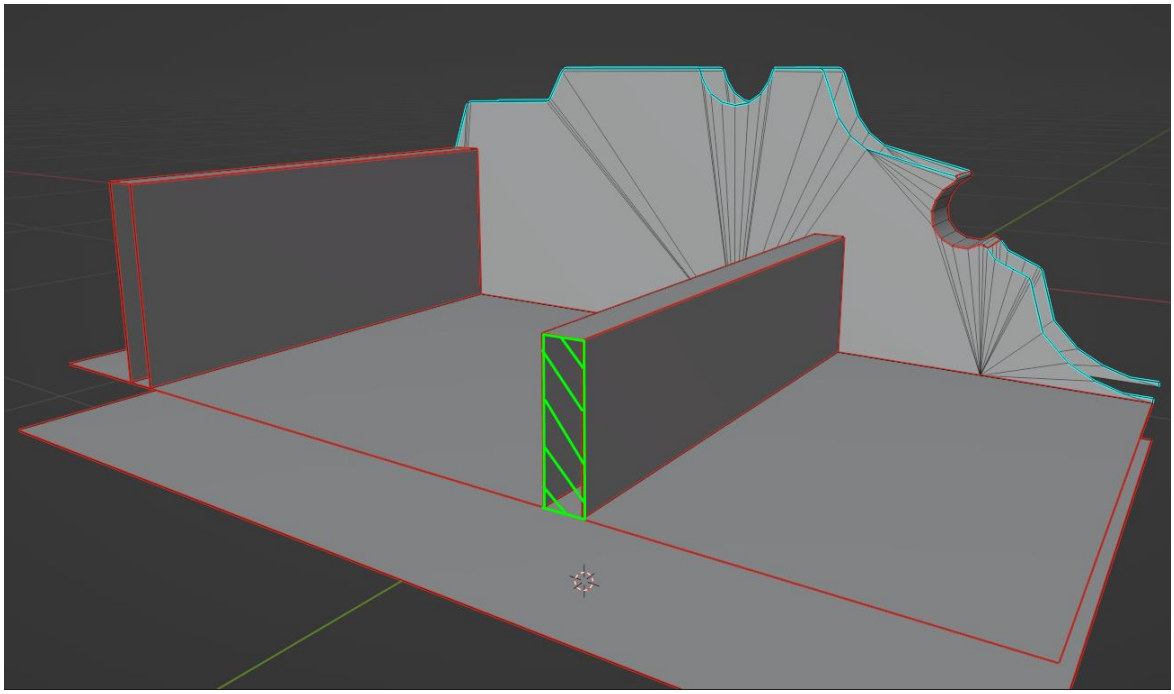


Рисунок 2.18 – Типовий приклад фейсу під видалення

Після застосування всіх технік оптимізації геометрії, фінальна модель має 7994 трикутника, як показано на (рис.2.19). Це входить в рамки технічного завдання, тому можна вважати етап моделювання виконаним успішно.

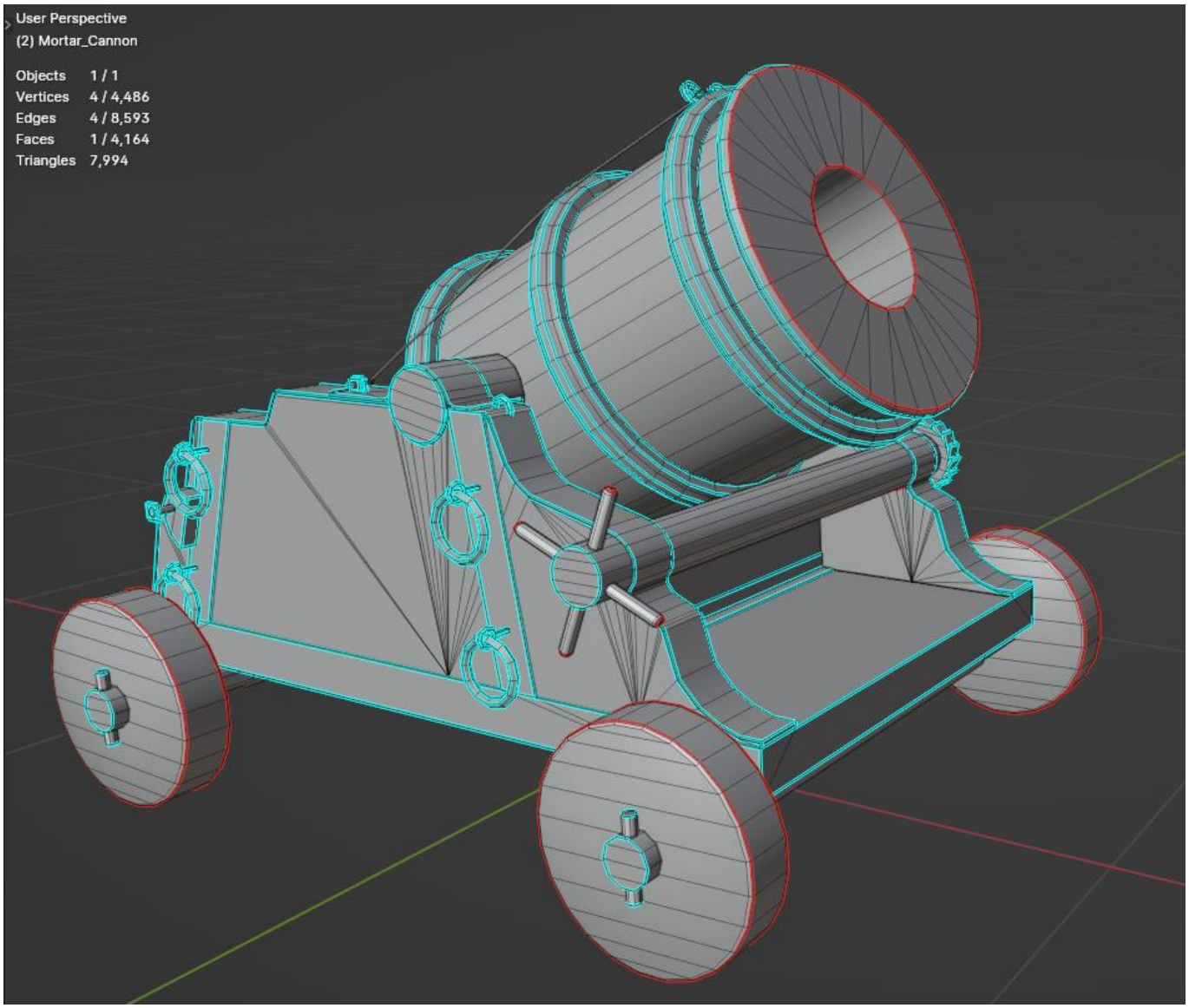


Рисунок 2.19 – Фінальна оптимізована лоуполі модель

2.3.8 Оптимізація розгортки

Наступним етапом підготовки лоуполі моделі до імпорту в Substance Painter є створення оптимізованої розгортки моделі.

UV-розгортка – це процес прив'язки текстур до геометричної моделі для того, щоб кожен піксель текстури визначався на моделі. У тривимірному моделюванні та комп'ютерній графіці 3D-об'єкти визначаються у тривимірному просторі, а текстури - у двовимірному просторі. Таким чином, потрібно встановити зв'язок між цими двома просторами.

UV-розгортка включає в себе визначення того, як координати текстур (U та V) відображаються на поверхні 3D-моделі. Кожній вершині моделі присвоюються координати UV, які вказують, який піксель текстури відповідає цій вершині. Таким чином, UV-розгортка генерує карту, яка визначає, які частини текстури будуть відображені на конкретних частинах моделі.

UV-розгортка є важливим етапом у створенні текстур для 3D-моделей, оскільки вона дозволяє ефективно наносити текстури на поверхні об'єктів і надає контроль над тим, як текстура відображається на моделі.

Для збереження простору на UV розгортці, використовують техніку під назвою Overlapping, як показано на (рис.2.20).

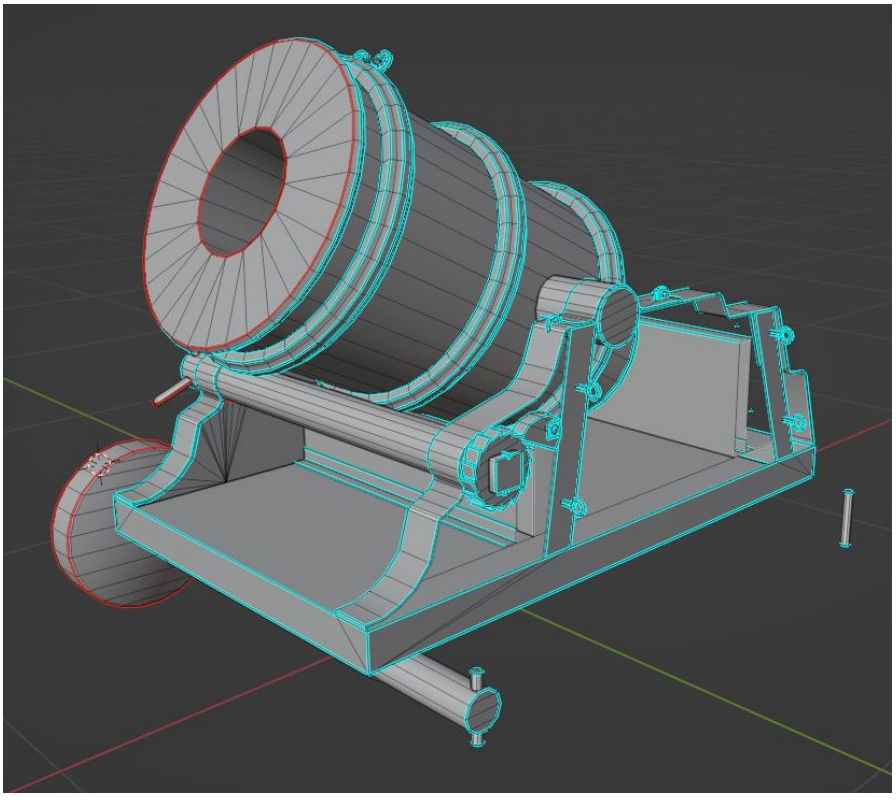


Рисунок 2.20 – Видалені деталі під Overlap

Ця техніка полягає в визначенні деталей моделі що повторюються, які не будуть видимі з одного і того ж ракурсу. Тобто, якщо у моделі з двох сторін однакова стіна, то одну із стін можна видалити і продублювати ту, що залишилась. Таким чином, можна зекономити велику частину UV розгортки.

Також рекомендується використовувати метод зменшення розміру другорядних деталей. Наприклад, якщо гравець не буде бачити дно підлоги, то йому можна відвести менше простору на UV розгортці. Це робиться вручну, зменшуючи тексель та розмір UV острова цією деталі.

Іншою важливою технікою оптимізації UV простору є випрямлення островів. Якщо на моделі є криві або окружні острови (зазвичай на циліндрах), то їх потрібно випрямити в прямокутники. Це дозволить більш компактно упакувати розгортку та запобігатиме появі графічних артефактів під час текстурування.

Якщо трикутник ріже піксель під кутом і проходить через центр пікселя, як показано на (рис. 2.21), то весь піксель зафарбовується кольором цього трикутника, що призводить до нерівних і піксельних країв, як показано на (рис. 2.22).. Щоб зменшити видимість цих нерівних країв, графічні процесори реалізують техніку, яка називається Super Sampling Anti-Aliasing.

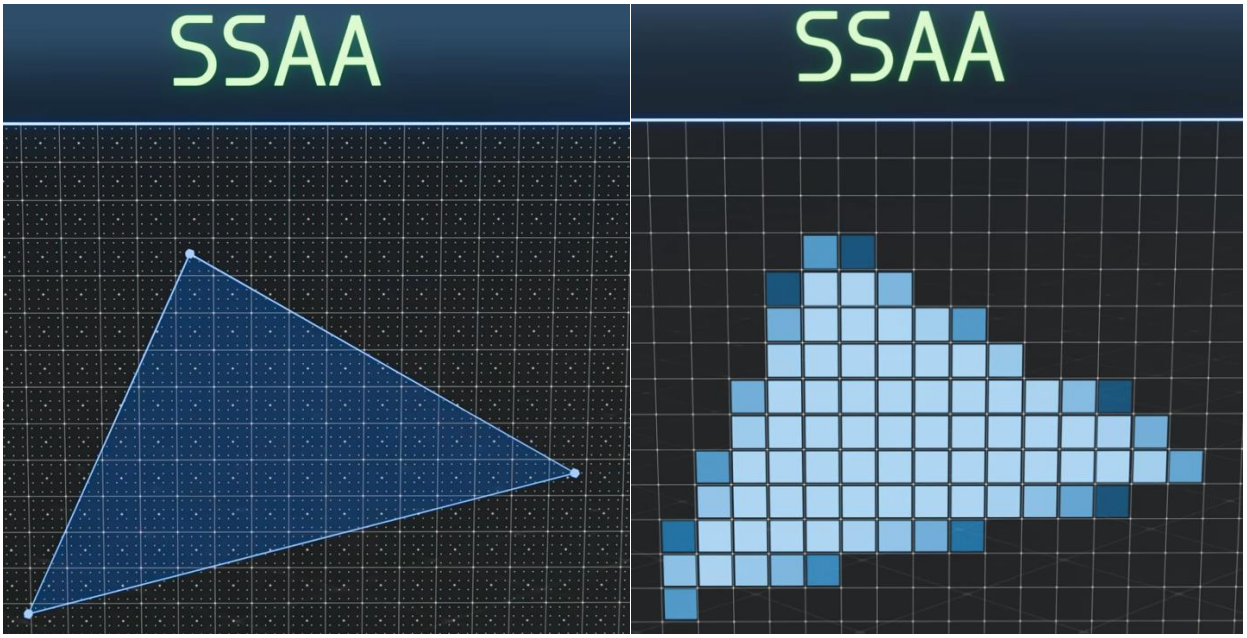


Рисунок 2.21 – Алгоритм роботи SSAA [8]

За допомогою SSAA 16 точок вибірки розподіляються по одному пікселю, і коли трикутник прорізає піксель, залежно від того, скільки з 16 точок вибірки охоплює трикутник, до пікселя застосовується відповідний дробовий відтінок цього кольору, що призводить до тьмяних країв зображення та значно менш помітної пікселізації, як показано на (рис. 2.23, 2.24).

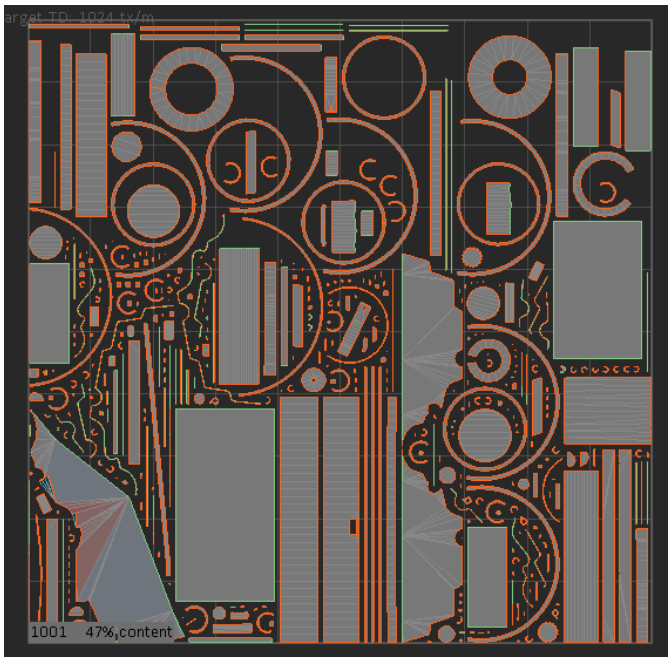


Рисунок 2.22 – Приклад неоптимізованої розгортки

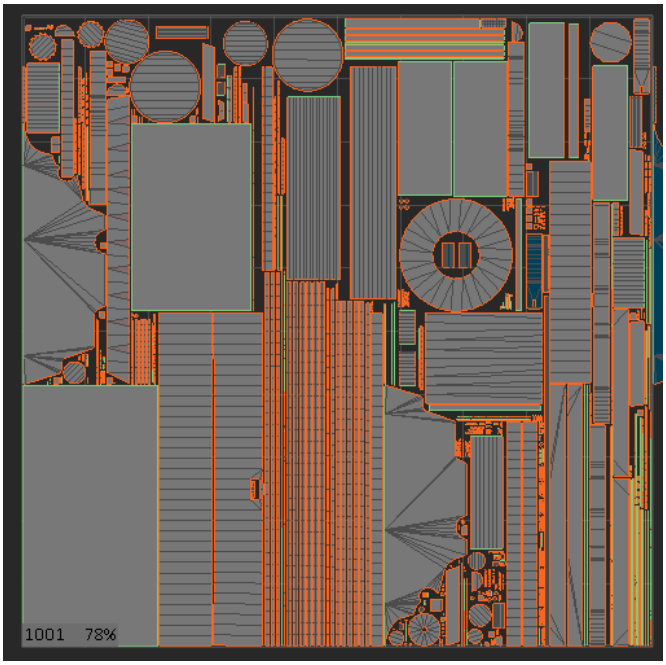


Рисунок 2.23 – Приклад оптимізованої розгортки

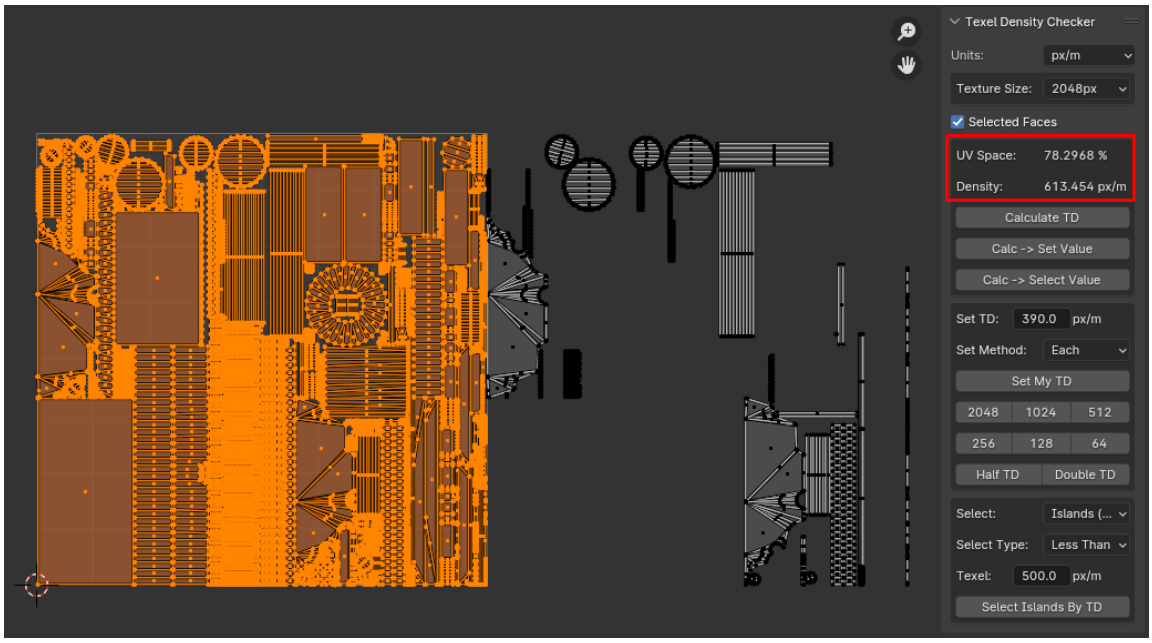


Рисунок 2.24 – Фінальний вигляд розгортки з заповненням в 78% і винесеними оверлапами

Після оптимізації, було отримано UV розгортку з заповненням UV простору в 78% щільності та текселем 613 px/m.

Це співпадає з технічними рамкам, тому можна вважати цей етап роботи виконаним вірно і якісно.

2.3.9 Шейдинг і кастом нормалі

Наступним етапом після розгортки є перевірка на артефакти шейдингу моделі і створення кастом нормалей. Шейдинг – це способ розрахунку освітлення і відбиття світла від об’єкта. Існують три режими шейдингу в Blender: Shade Smooth, Shade Auto Smooth, Shade Flat.

Для того, щоб розрахувати затінення трикутника, нам потрібно знати дві ключові деталі. По-перше, напрямок світла, а по-друге, напрямок, в якому спрямована поверхня трикутника. Напрямок, до якого звернений окремий трикутник, називається нормалем його поверхні, тобто просто напрямком, перпендикулярним до площини трикутника.

Щоб обчислити затінення трикутника Sh, ми беремо косинус кута між двома напрямками, як показано на (рис. 2.25).

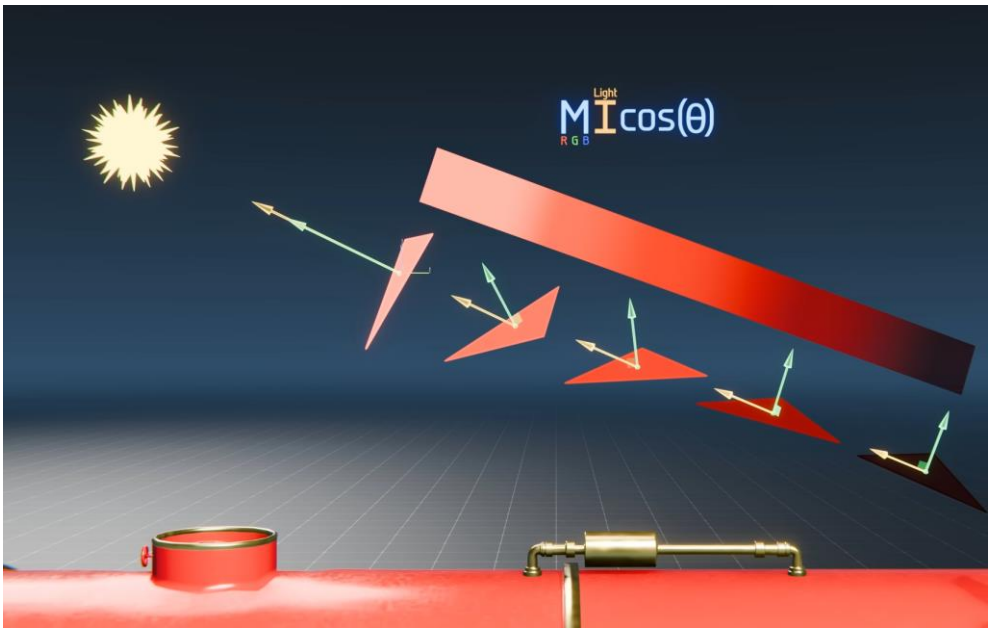


Рисунок 2.25 – Вплив кута θ на затемнення матеріалу [8]

Далі ми множимо тета косинуса на інтенсивність світла, а потім на колір матеріалу, щоб отримати правильно заштрихований колір цього трикутника. Цей процес регулює значення RGB трикутників, і в результаті ми отримуємо діапазон від світла до темряви поверхні залежно від того, як її окремі трикутники звернені до світла.

Однак, якщо поверхня перпендикулярна або звернена в бік, нам не потрібне значення тета косинуса 0 або від'ємне число, тому що це призведе до повністю чорної поверхні. Тому ми встановлюємо мінімальне значення 0 і додаємо інтенсивність навколишнього освітлення - A, помножену на колір поверхні - M, і налаштовуємо це навколишнє освітлення так, щоб воно було вищим у денних сценах і ближче до 0 вночі (1.1).

$$Sh_1 = MI * cos\,\theta + MA \qquad (1.1)$$

Коли в сцені є кілька джерел світла, ми виконуємо цей розрахунок кілька разів з різними напрямками та інтенсивністю світла, а потім додаємо разом для кожного джерела світла (1.2).

$$Sh_3 = M(I_1 \cos(\theta_1) + I_2 \cos(\theta_2) + I_3 \cos(\theta_3) + A) \qquad (1.2)$$

Наявність більш ніж кількох джерел світла є обчислювально інтенсивною для вашого графічного процесора, і, таким чином, сцени обмежують кількість окремих джерел світла, а іноді обмежують діапазон впливу світла, щоб трикутники ігнорували далеке світло.

Одна з ключових проблем полягає в тому, що трикутники всередині об'єкта мають лише одну нормаль, і, таким чином, кожен трикутник матиме однаковий колір по всій поверхні трикутника. Це називається плоским затіненням (Shade Flat) і є досить нереалістичним, якщо дивитися на вигнуті поверхні, такі як корпус цієї парової машини. Отже, для того, щоб отримати плавне затінення, замість використання нормалей поверхні, ми використовуємо по одній нормалі для кожної вершини, обчисленої з використанням середнього значення нормалей сусідніх трикутників.

Далі ми використовуємо метод, званий барицентричними координатами, щоб отримати плавний градієнт нормалей по поверхні трикутника. Візуально це схоже на змішування 3 різних кольорів у трикутнику, але замість цього ми використовуємо три нормальні напрямки вершин.

Для заданого фрагмента ми беремо центр кожного пікселя і використовуємо нормалі вершин і координати попередньо растеризованого трикутника, щоб обчислити барицентричну нормаль цього конкретного пікселя. Так само, як і змішування трьох кольорів у трикутнику, нормаль цього пікселя буде пропорційною сумішшю трьох нормалей (n_0, n_1, n_2)вершин трикутника (1.3).

$$n_{\text{піксель}} = \alpha n_0 + \beta n_1 + \gamma n_2 \qquad (1.3)$$

В результаті, коли набір трикутників використовується для формування криволінійної поверхні, кожен піксель буде частиною градієнта нормалей, що призводить до градієнта кутів, звернених до світла, з попіксельним забарвленням і плавним затіненням по всій поверхні.

Shade Auto Smooth – використовує інтерполяцію Vertex Normals для гладких і викривлених поверхонь, а Face Normals – для плоских поверхонь, які підпадають під обмеження кутом нахилу. Це означає, що всі фейси, кут нахилу яких більше ніж 35°, будуть згладжені за алгоритмом Shade Smooth, і навпаки.

Іноді, Shade Auto Smooth може викликати артефакти шейдингу , як показано на (рис. 2.26). Вони виглядяють як затемнені ділянки моделі на краях об’єкту. Такі артефакти зазвичай з’являються на гострих кутах між двома пласкими фейсами , як показано на (рис. 2.27). Якщо такі артефакти не виправити, то вони будуть некоректно відображати накладені текстури на ці ділянки.

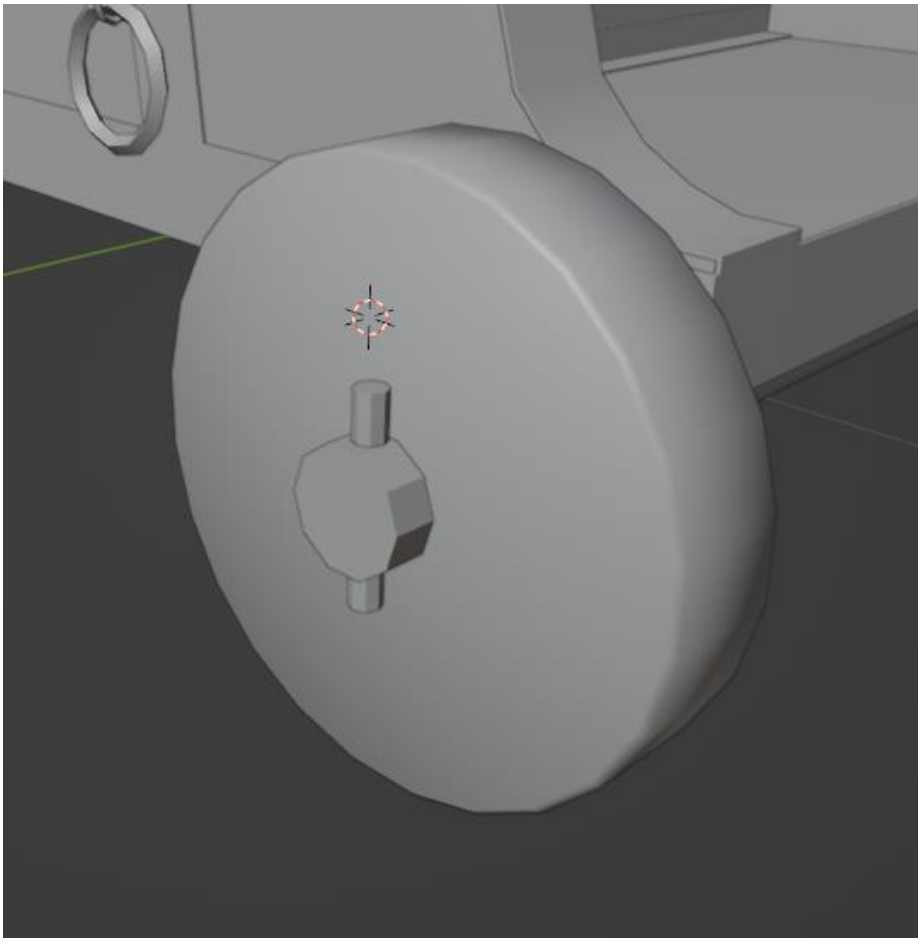


Рисунок 2.26 – Приклад артефакту шейдингу на краях колеса

Щоб це виправити, використовують спосіб створення Custom Normals. Цей спосіб створює нові (кастомні) нормалі на об’єкті, які можна змінювати вручну де це потрібно. Це дозволяє виправляти некоректне відображення шейдингу на проблемних ділянках.

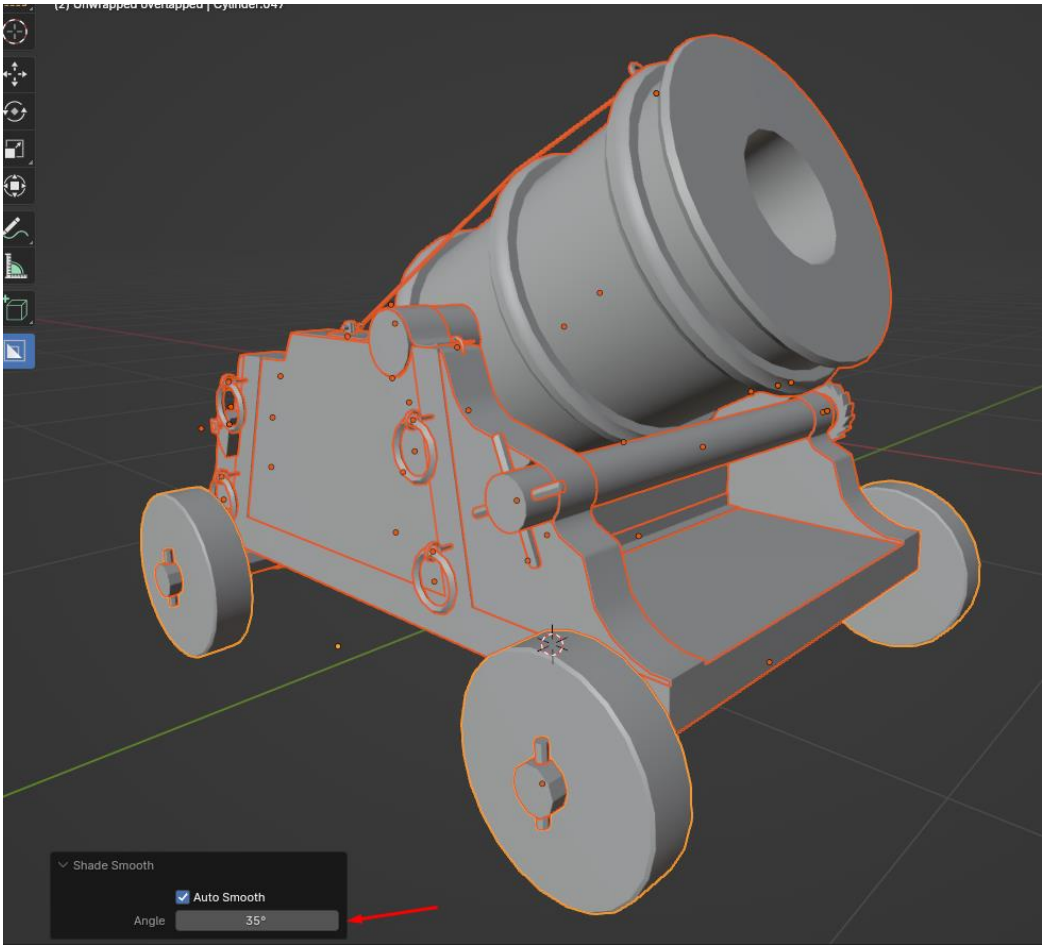


Рисунок 2.27 – Застосування Shade Auto Smooth на моделі

Для створення кастомних нормалей був використаний безкоштовний аддон TexTools. Функція Smooth by UV Islands згладжує UV острова і проставляє Sharp еджі на краях острова , як показано на (рис. 2.28). Це дозволяє уникнути артефактів згладжування, які з’являються саме на таких місцях.

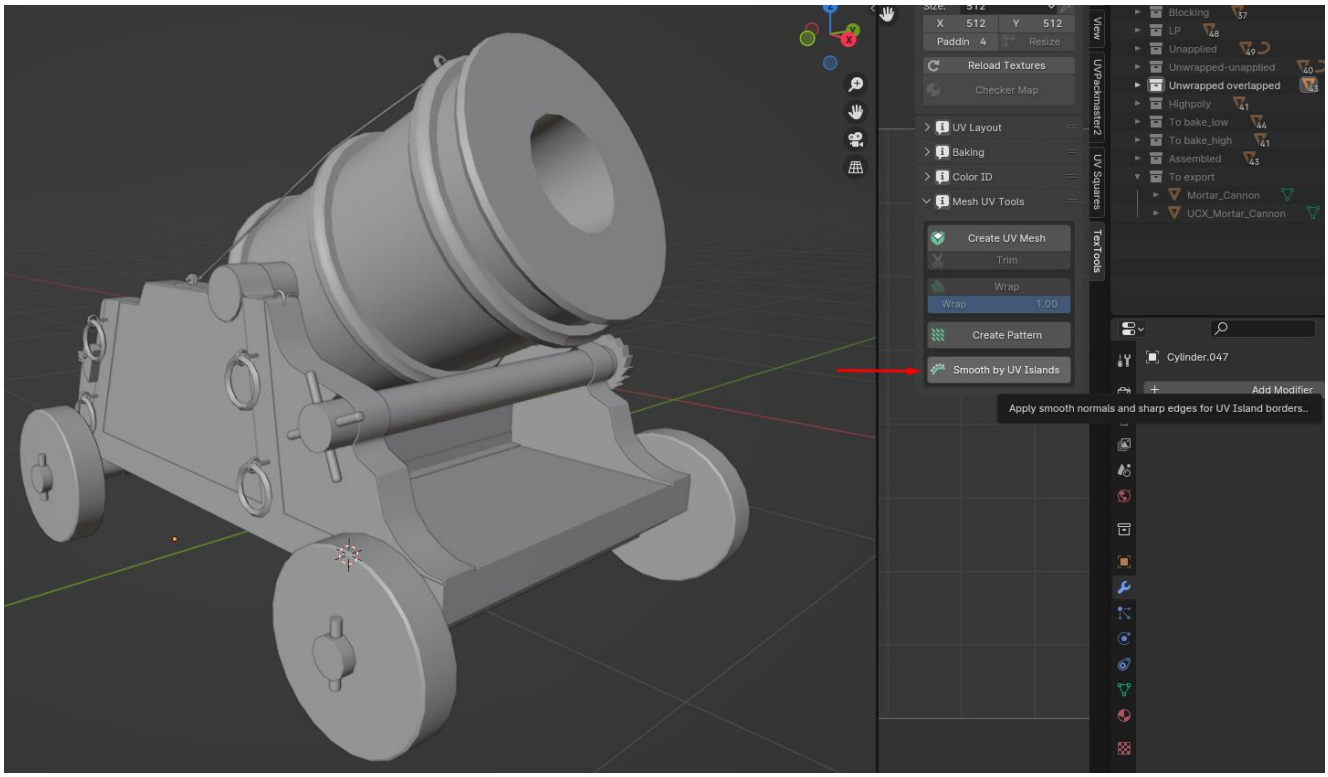


Рисунок 2.28 – Застосування функції Smooth by UV Islands

Через те, що цей спосіб створює гострі грані на краях UV островів, після його використання треба почистити зайві гострі еджі. Наприклад, на місцях розрізу циліндрів , як показано на (рис. 2.29).

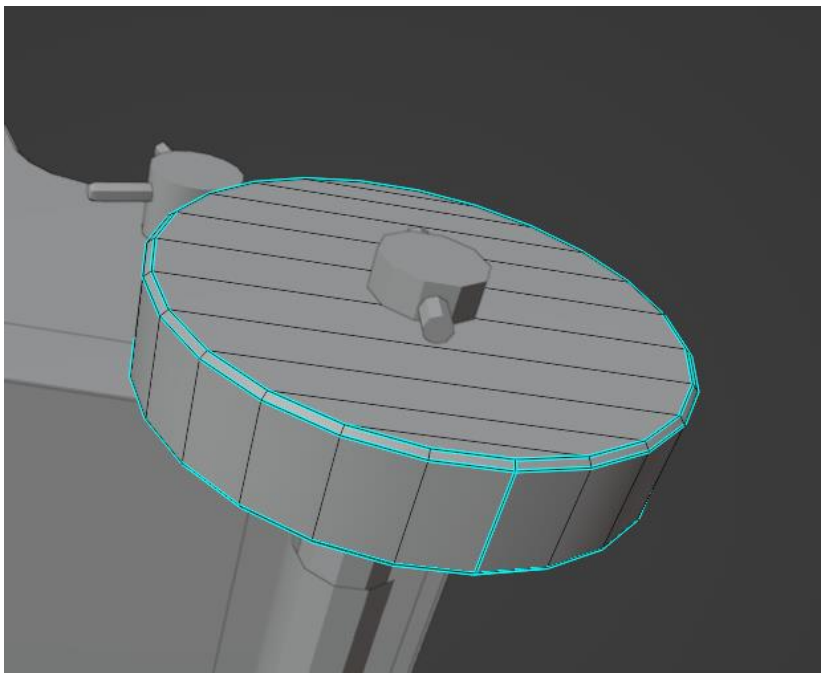


Рисунок 2.29 – Видалення зайвих Sharp еджів на краях UV островів

2.3.10 Запікання

Для запікання і подальшого текстурування була обрана програма Substance Painter. Вона є стандартом індустрії через свою багатофункціональність та зручність.

Для коректного запікання, потрібно “рознести” деталі моделі (на лоуполі, як показано на (рис. 2.30) і хайполі, як показано на (рис. 2.31)) одна від одної. Це можна зробити за допомогою анімації, або вручну. Це робиться для того, щоб уникнути некоректного запікання карт Ambient Occlusion та Normal.

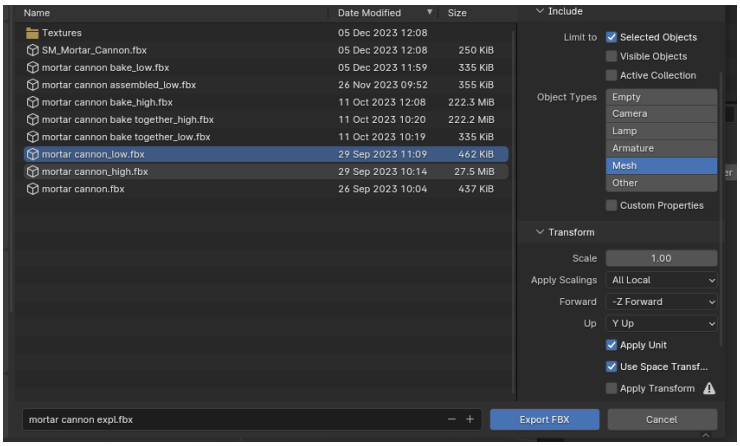


Рисунок 2.30 – Параметри експорту лоуполі моделі для роботи в Substance Painter

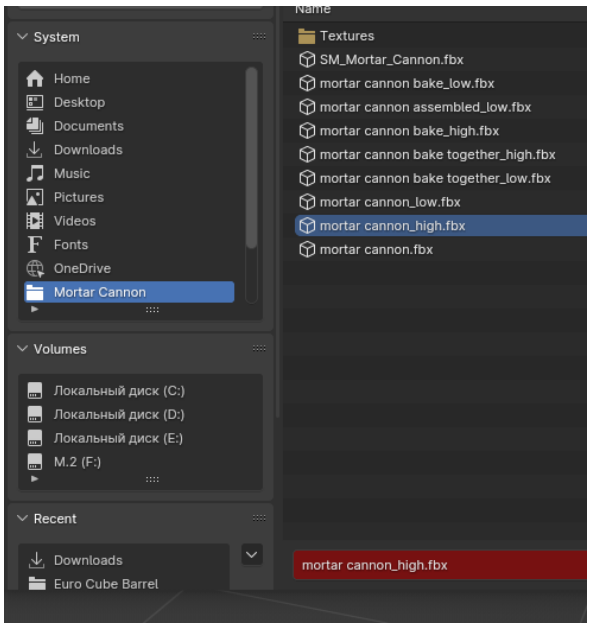


Рисунок 2.31 – Параметри експорту хайполі моделі для роботи в Substance Painter

Існує і інший спосіб експорту – Mesh Naming. Він полягає в призначенні однакового імені для однакових деталей на хайполі і лоуполі. Потім в налаштуваннях запікання Substance Painter треба увімкнути розділення запікання By Mesh Name. Тобто кожна індивідуальна деталь буде запікати AO і нормалі по відношенню до самої себе.

Але у цього метода є один мінус: через те, що деталь запікається сама на себе, то на неї не впливають інші деталі моделі. Тому на неї не запікається затінення решти моделі на карті AO.

В параметрах запікання меш мап, спочатку треба імпортувати хайполі модель. Потім потрібно вибрати коректний розмір карти запікання – 2048px , як показано на (рис. 2.32). Після цього, дивлячись на модель у меню перегляду, треба експериментальним чином обрати оптимальні Max Frontal і Max Rear дистанції. Від цього залежить яка кількість деталей, що випирають за лоуполі меш, запечеться на нормалі.

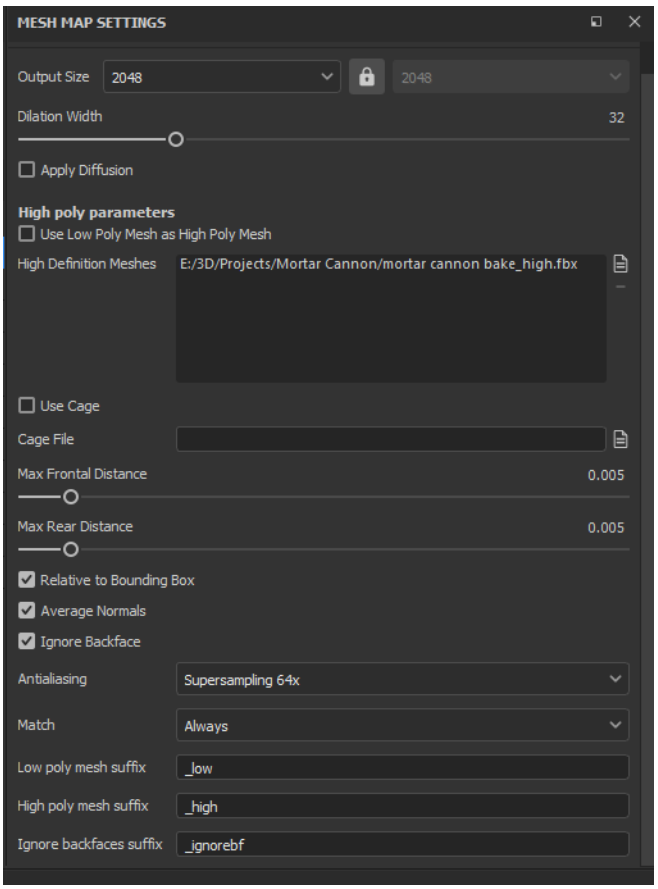


Рисунок 2.32 – Основні параметри запікання меш мап в Substance Painter

Далі треба обрати ступінь антіаліасінгу. Це дозволяє згладити гострі края на UV розгортці, але на це пійде більше часу при запіканні. Хоча наша UV розгортка і випрямлена, ступінь антіаліасінгу в 64х дозволить загладити гострі пікселі на краях циліндрів і непрямокутних островів. Далі треба запекти меш карти , як показано на (рис. 2.33).

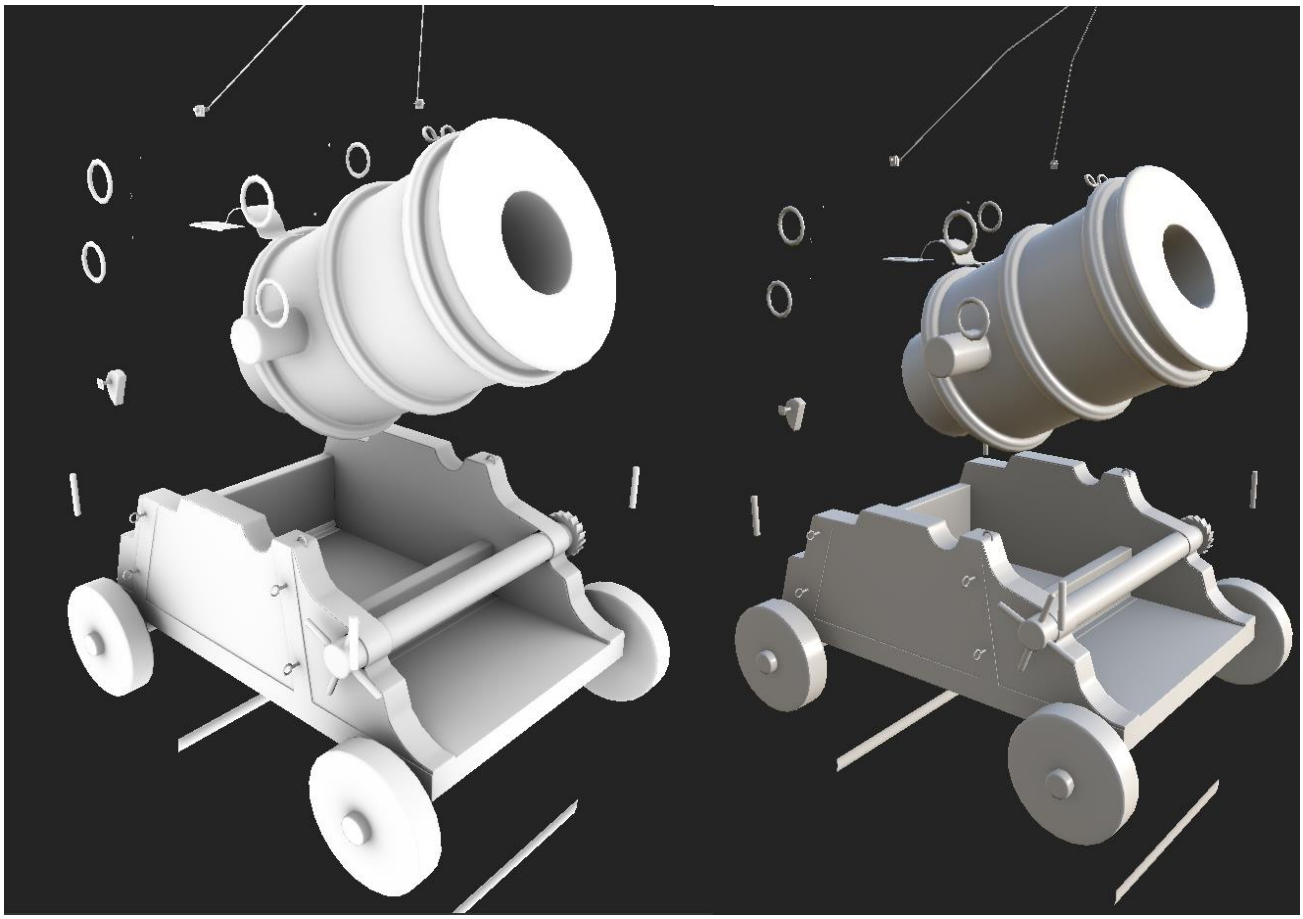


Рисунок 2.33 – Запечені карти Ambient Occlusion та Normals на лоуполі моделі

2.3.11 Текстурування

Наступним етапом в створенні ігрової моделі є текстурування. Це найкреативніша частина пайплайну і результат вже не обмежується рамками оптимізації. Тому на цьому етапі дуже важливо максимально точно передати стиль, стан, настрій і задум всієї моделі.

Для початку, потрібно визначитись з кількістю різних матеріалів на референсі. Після перегляду прикладів, було визначено п’ять основних матеріалів: дерево, мотузка, чавун, коване залізо і цвяхи.

Були створені окремі папки матеріалів і за допомогою чорної маски були обрані потрібні деталі моделі для кожного матеріалу , як показано на (рис. 2.34).

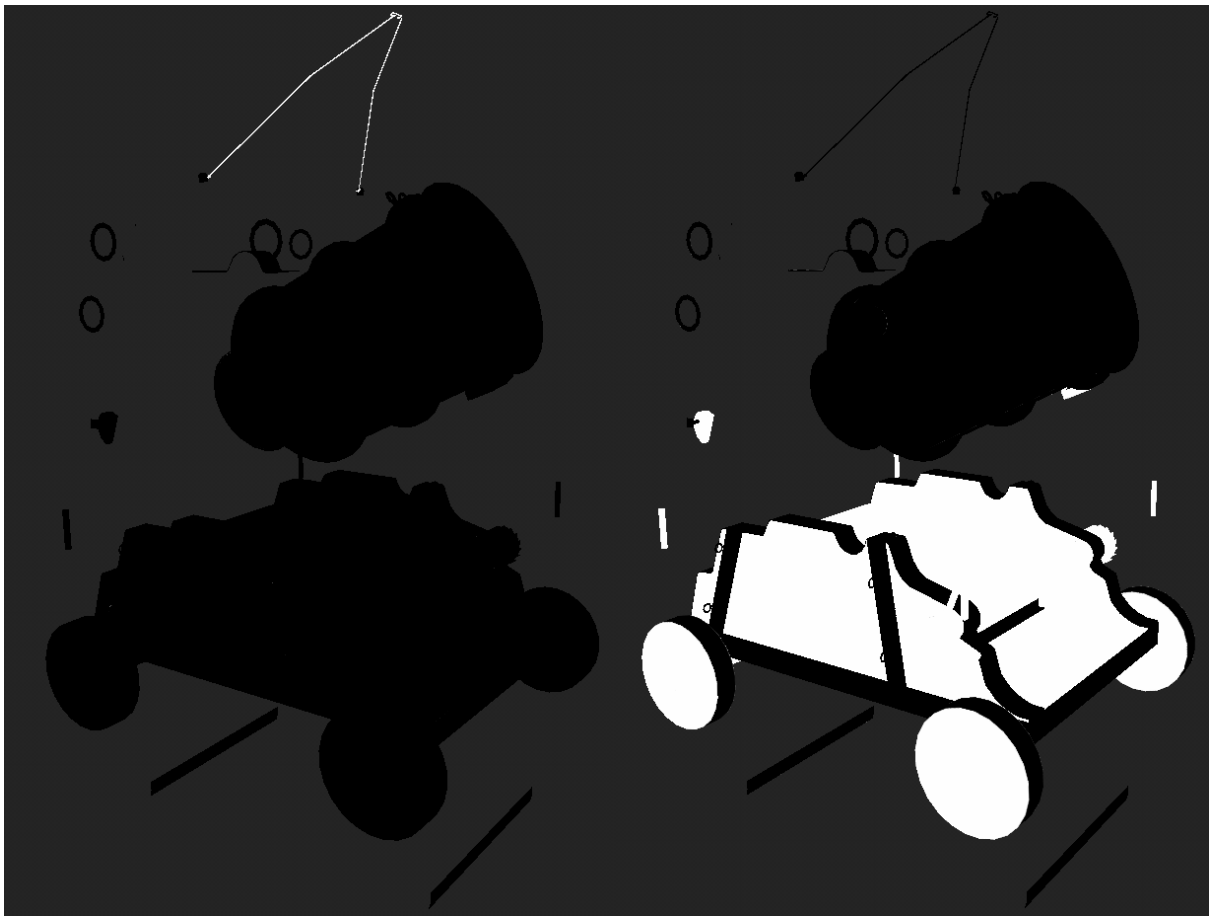


Рисунок 2.34 – Маскування окремих деталей моделі під окремі матеріали: мотузка (ліворуч) і дерево (праворуч)

Для створення маски цвяхів, був створений альфа-пензель у формі розташування цвяхів, як показано на (рис. 2.35). Потім на UV розгортці в панелі 2D, цим пензлем були проставлені цвяхи згідно з референсів, як показано на (рис. 2.36, 2.37).

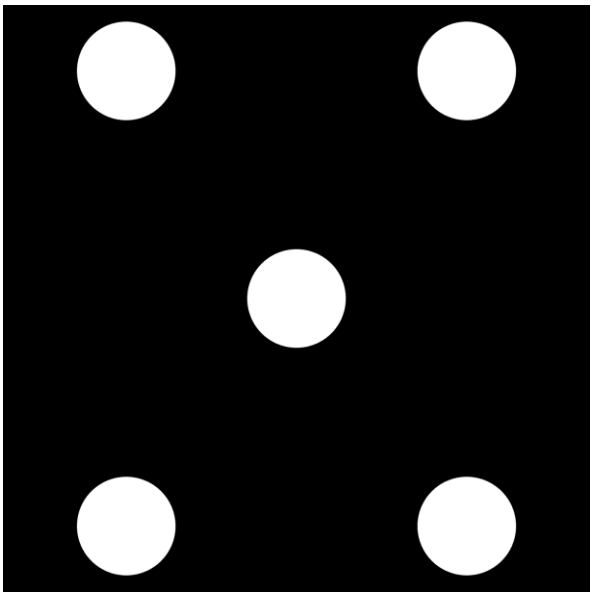


Рисунок 2.35 – Створений альфа пензлик для маскування цвяхів

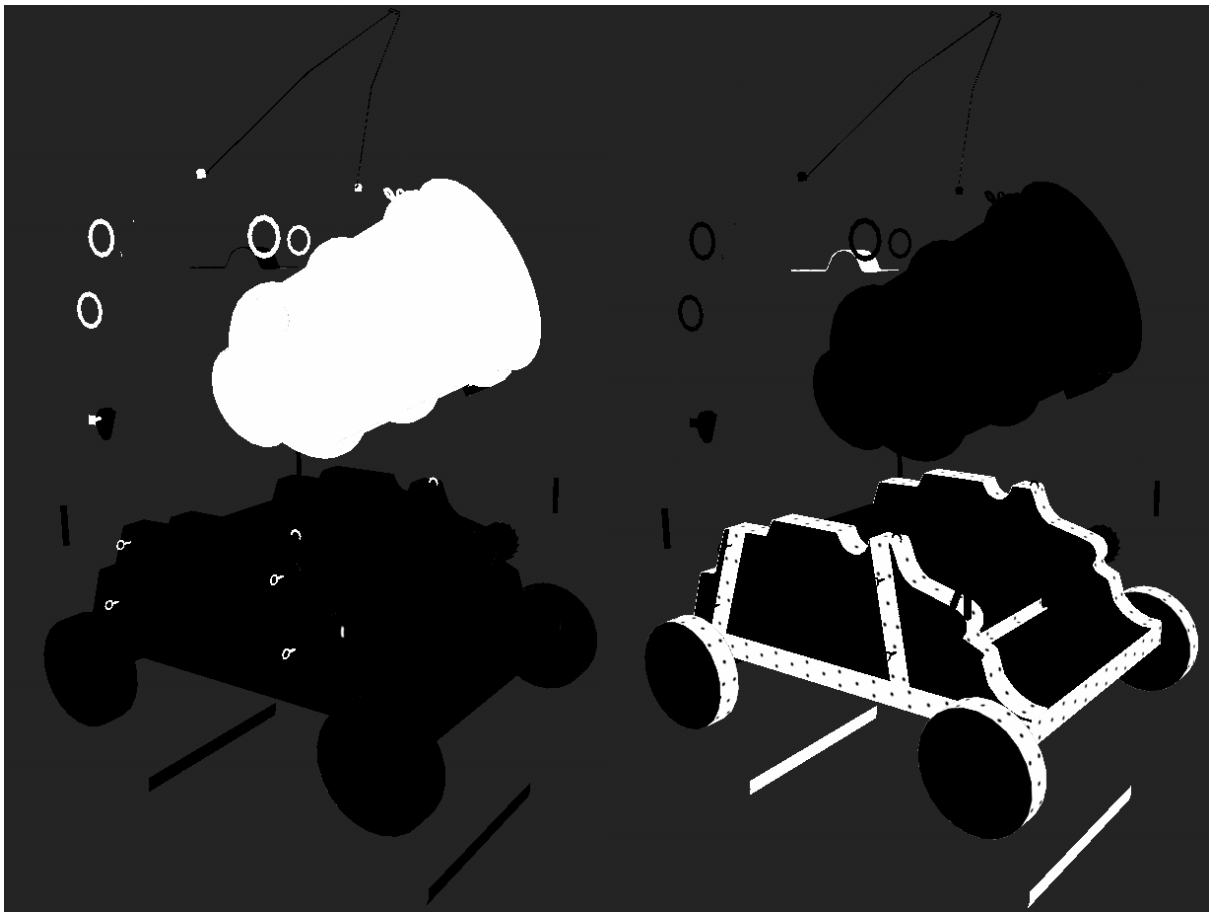


Рисунок 2.36 – Маскування окремих деталей моделі під окремі матеріали: чавун (ліворуч) і кован залізо (праворуч)

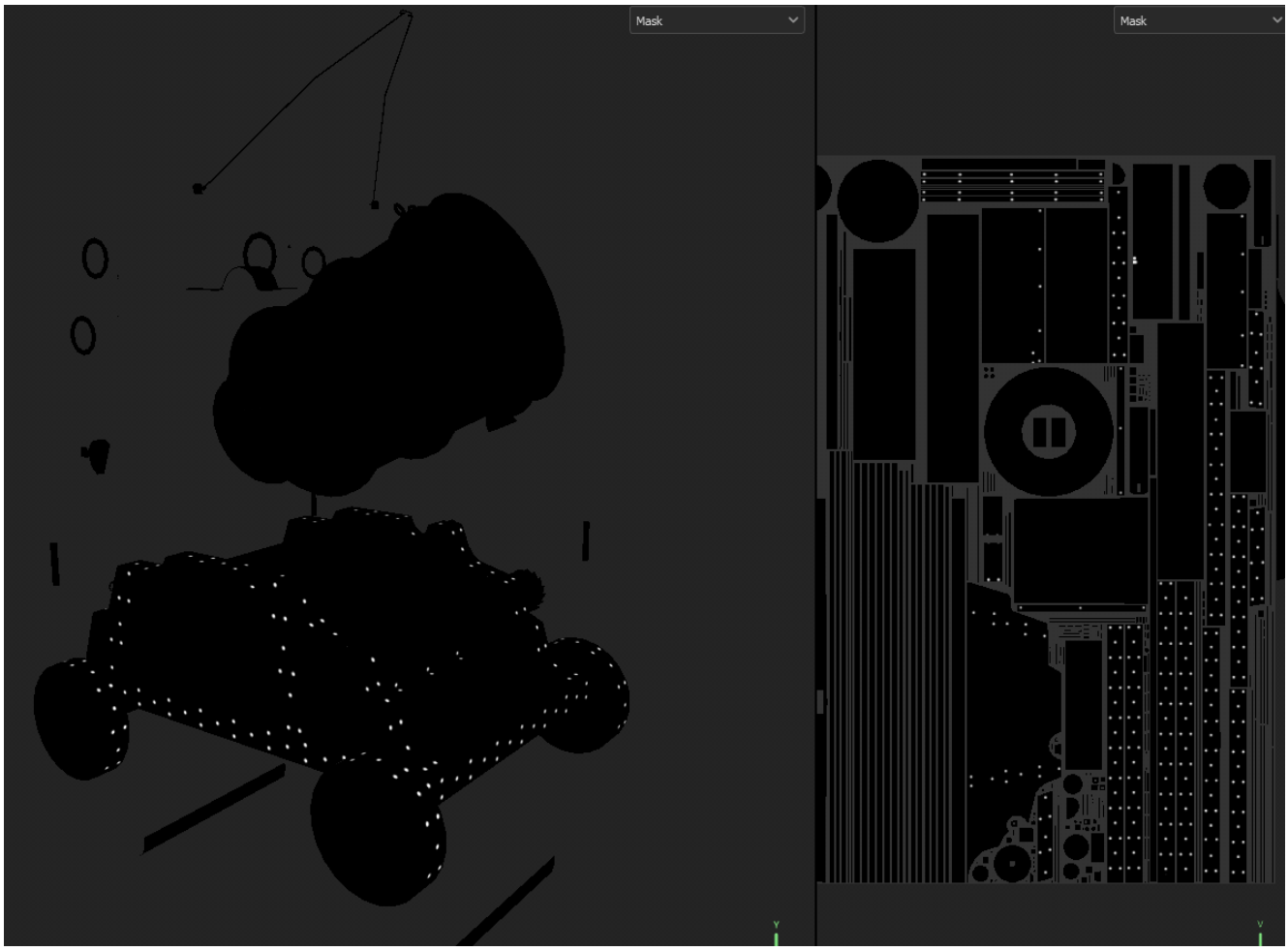


Рисунок 2.37 – Результат маскування за допомогою альфа-пензлика

Для кованого заліза був обраний матеріал старого заліза, кована текстура була створена процедурально за допомогою текстури Voronoi, а також були додані елементи складених країв на швах UV розгортки , як показано на (рис. 2.38).

Для цвяхів, був створений іржавий матеріал заліза. Вибірково було “зачищено” деякі ділянки, щоб текстура не була такою одноманітною , як показано на (рис. 2.39).

Чавунне дуло гармати було зроблено з комбінації двох матеріалів заліза для варіації кольору, була додана текстура подряпин і потертостей, а також сліди від пороху на дулі , як показано на (рис. 2.40).

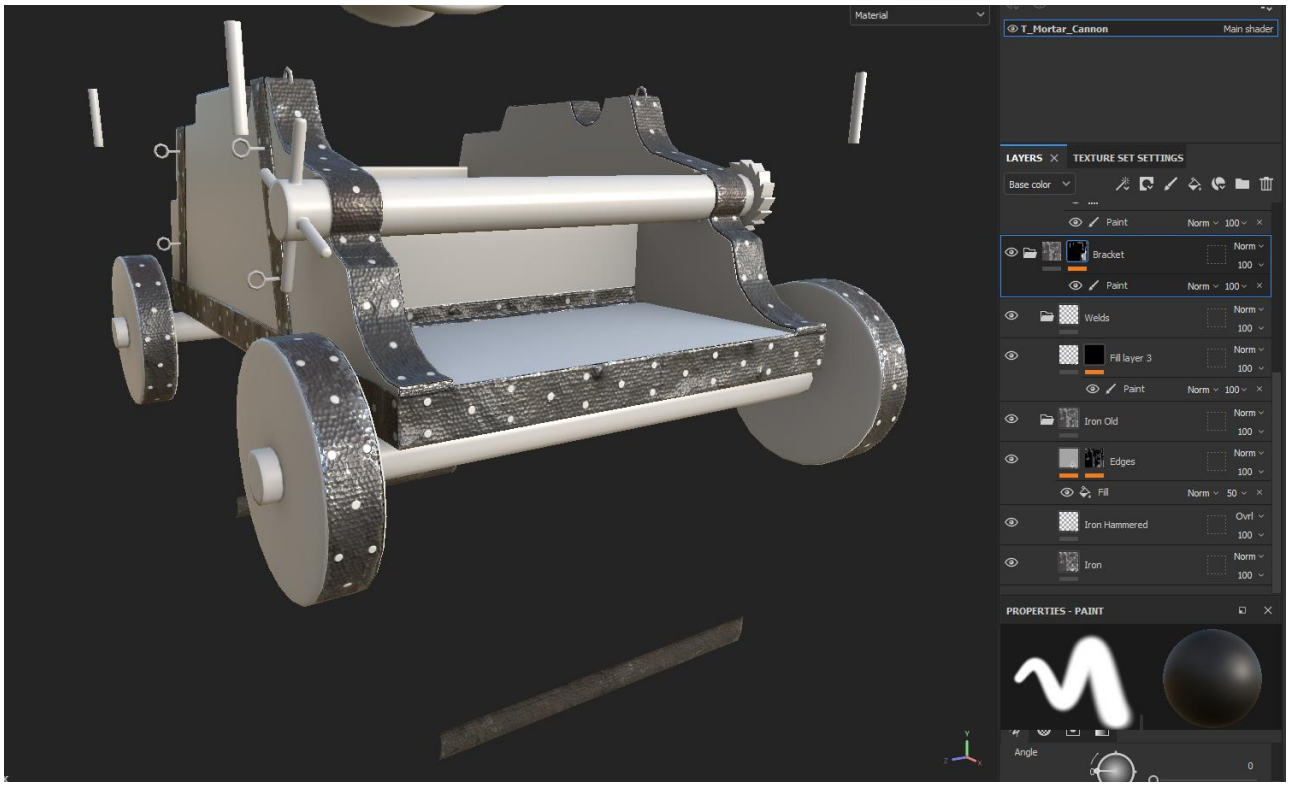


Рисунок 2.38 – Готовий матеріал кованого заліза

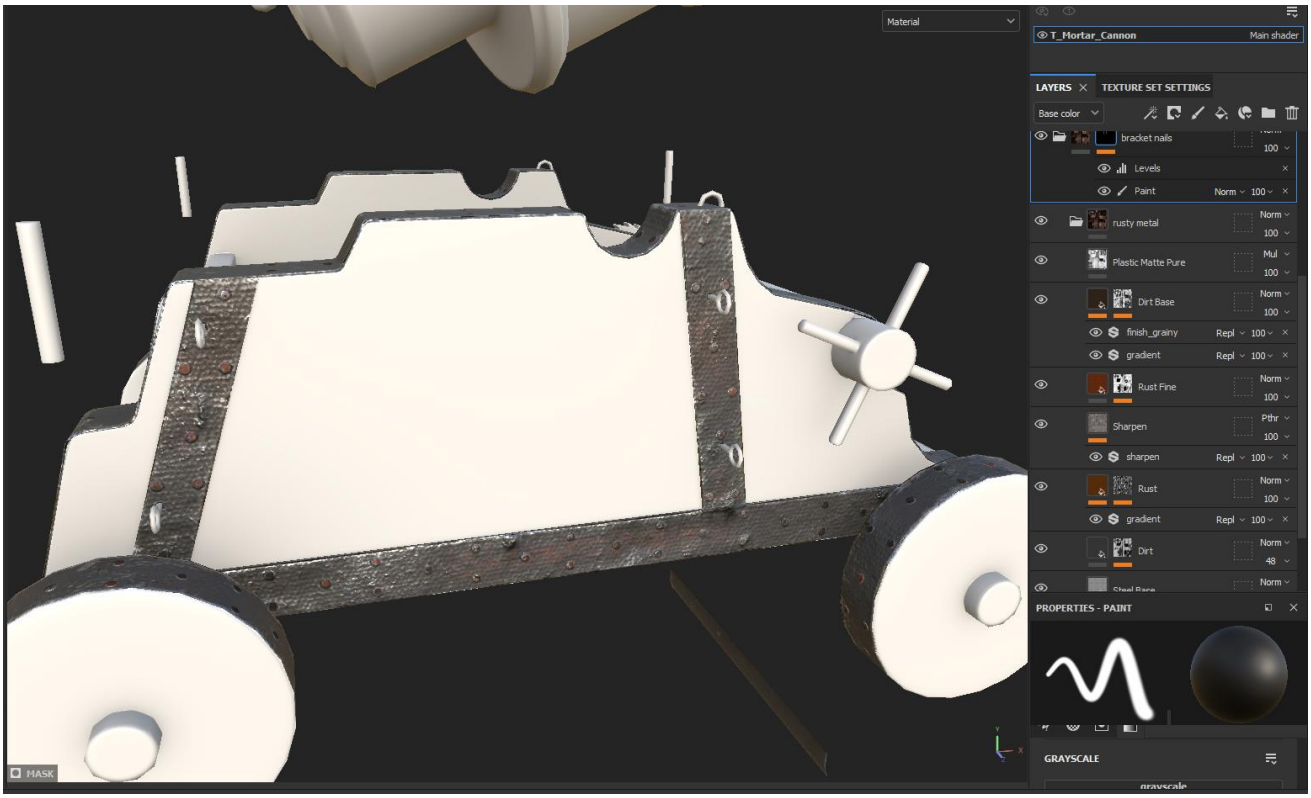


Рисунок 2.39 – Готовий матеріал цвяхів

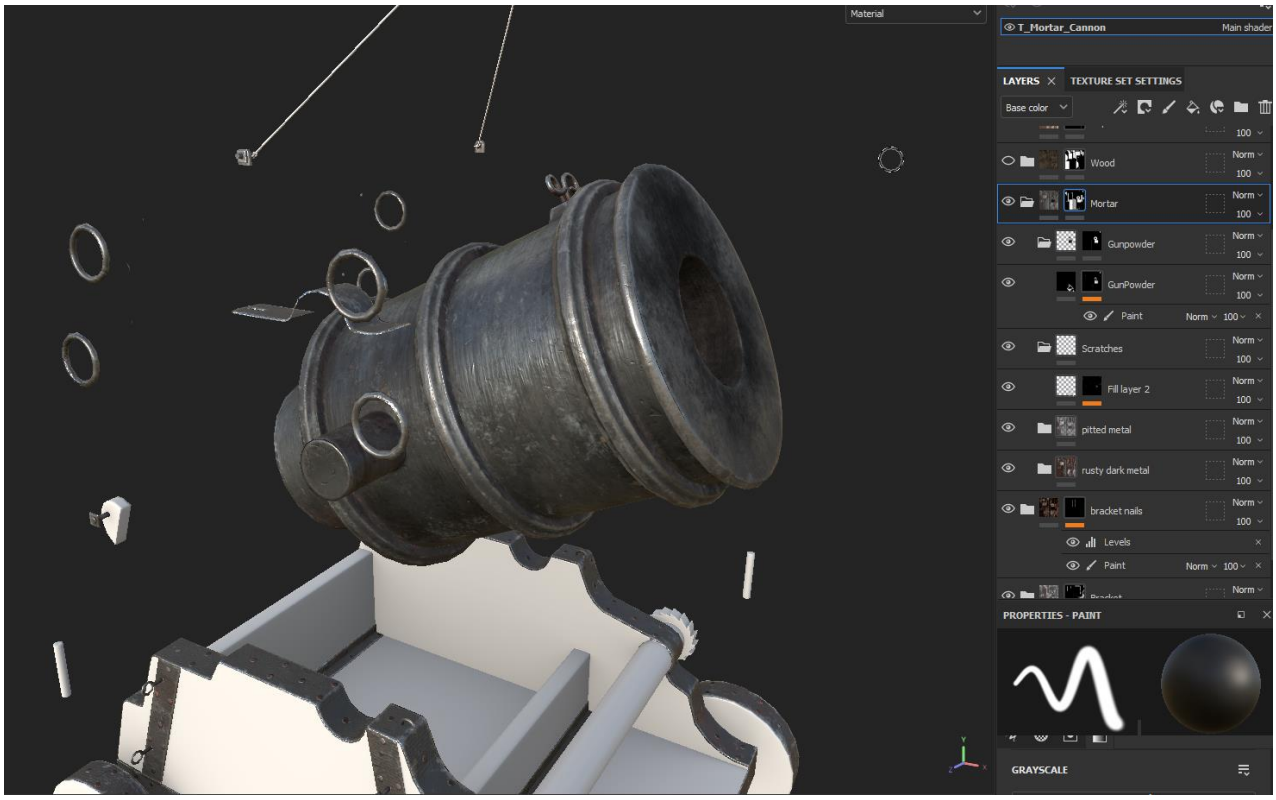


Рисунок 2.40 – Готовий матеріал чавунної гармати

Дерев’яний каркас був зроблений з текстури дошок накладених поверх матеріалу дерева. Також були додані відтінки зелених і жовтих кольорів для передачі занедбаності, як показано на (рис. 2.41).

Матеріал мотузки був зроблений з фотоскану (результат сканування реального об’єкту камерою з великою роздільною здатністю. Використовуються для створення гіперреалістичного стилю) реальної мотузки, також на фотоскан було накладено шар Roughness через функцію Anchor, як показано на (рис. 2.42).

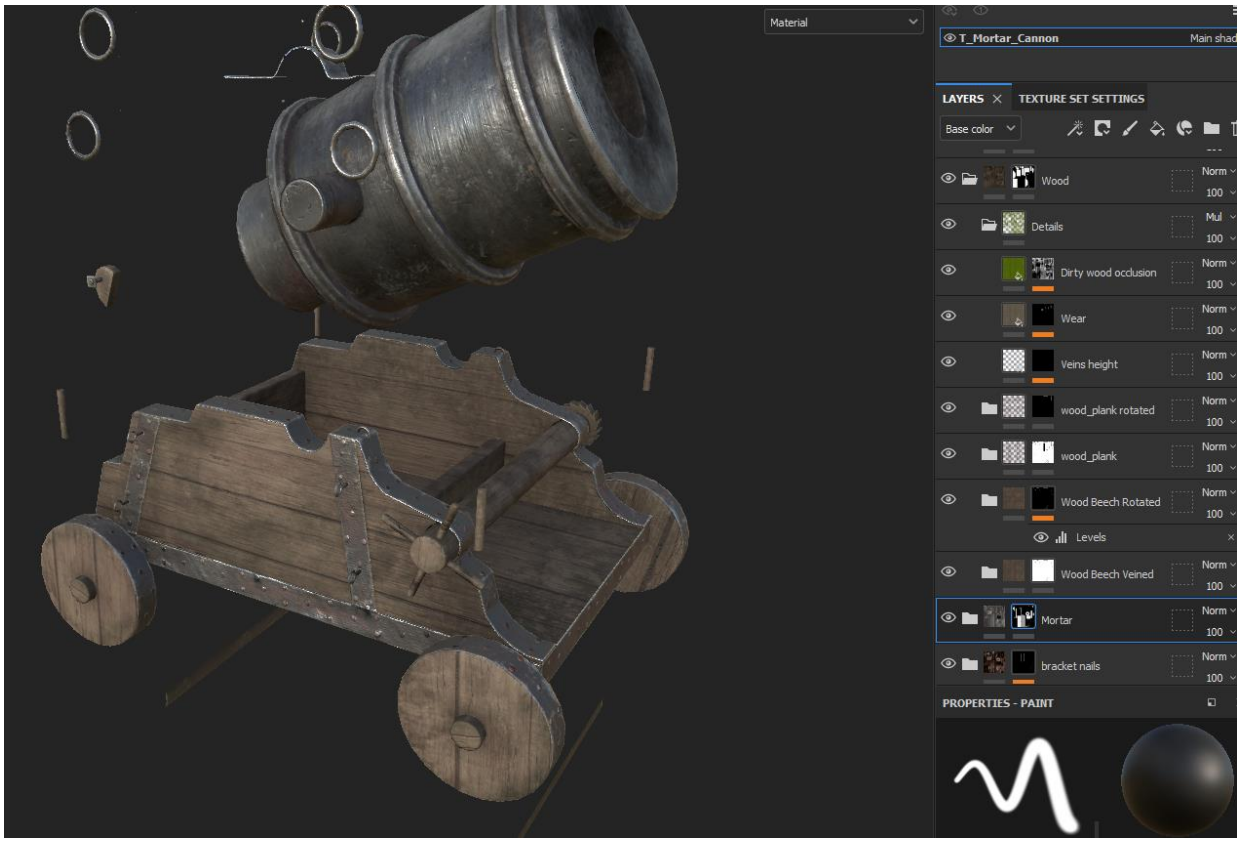


Рисунок 2.41 – Готовий матеріал дерева

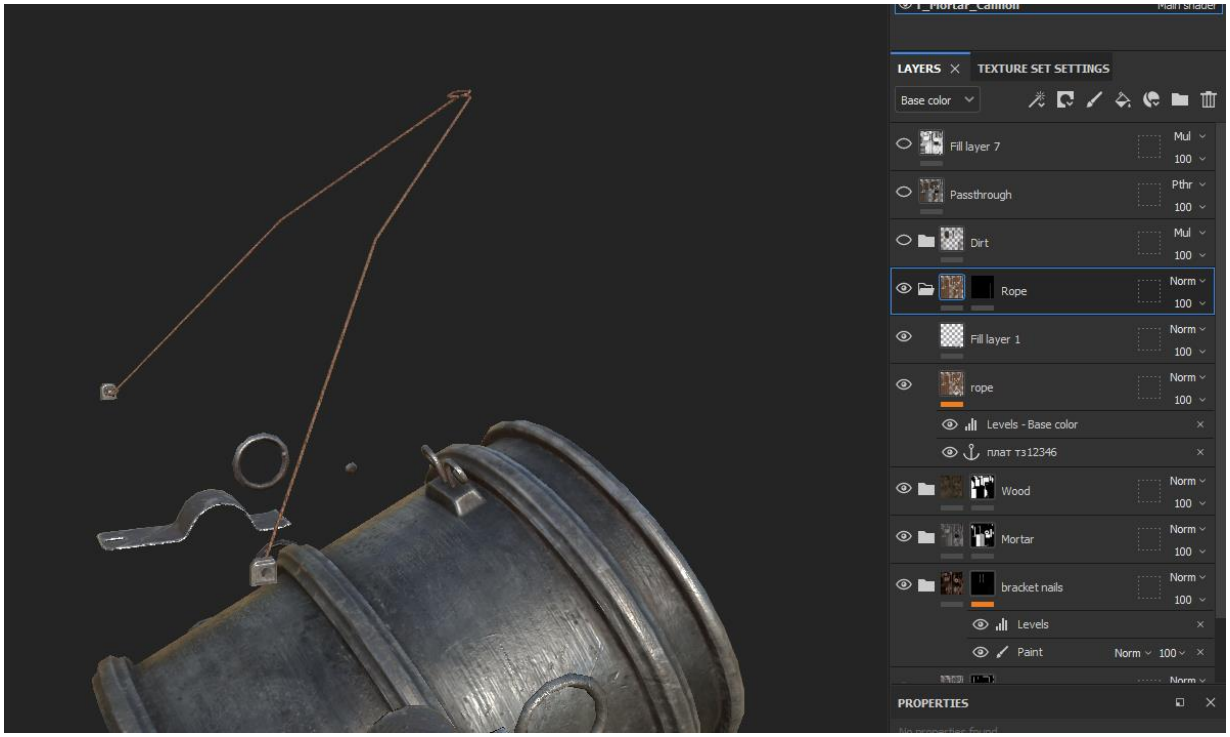


Рисунок 2.42 – Готовий матеріал мотузки

Зверху цих шарів було створено шари потертостей, бруду і пилу для передачі використаного і занедбаного стану, як показано на (рис. 2.43). Були використані як процедуральні генератори бруду по AO, так і техніки ручного малювання. Наступним шаром було створено мультиплікатор AO для поглибшення контрасту Ambient Occlusion. Це надає текстурам більш кінематографічного і цікавого вигляду, як показано на (рис. 2.44).

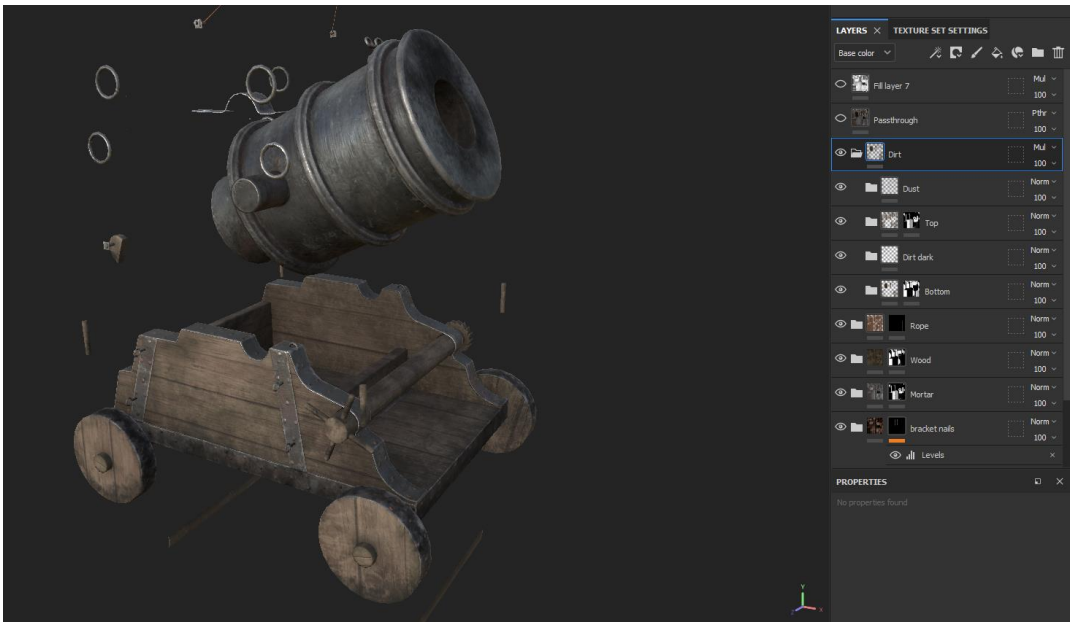


Рисунок 2.43 – Шар потертостей, бруду і пилу

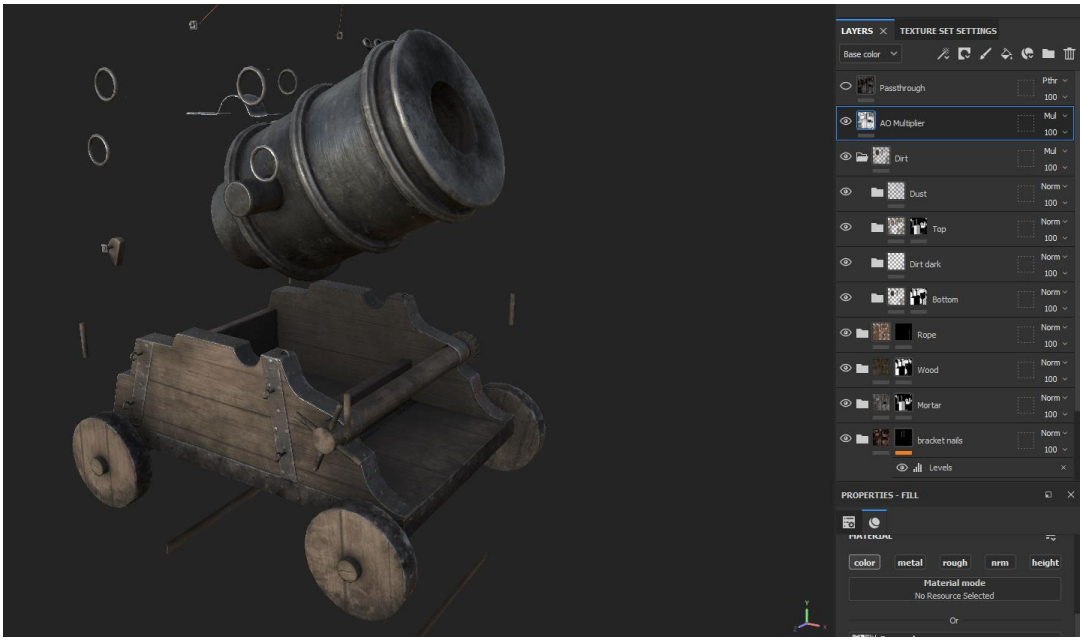


Рисунок 2.44 – Шар посилення AO

Останній шаром на кожному етапі текстурування повинен бути шар “затирання швів”. Це шар Passthrough, на якому вручну проходяться по всім помітним швам нормалей і “затирають” їх. Таким чином можна позбутися небажаних швів UV розгортки і зробити ілюзію непреривності текстури, як показано на (рис. 2.45).

Рисунок 2.45 – Результат затирання швів за допомогою техніки Passthrough

Експорт текстур робиться по кастомному пресету. Цей пресет був створений для експорту текстур ORM. Він включає в себе три текстури: DIF – основний колір текстури, ORM – Occlusion, Roughness та Metallic, та NOR – карта нормалей, як показано на (рис. 2.46).

Всі карти експортуються в роздільній здатності 2К та на бітовій глибині в 8 біт. Інколи карту нормалей експортують в 16 біт, але в такому випадку така карта може важити в десятки разів більше з незначним візуальним покращенням.

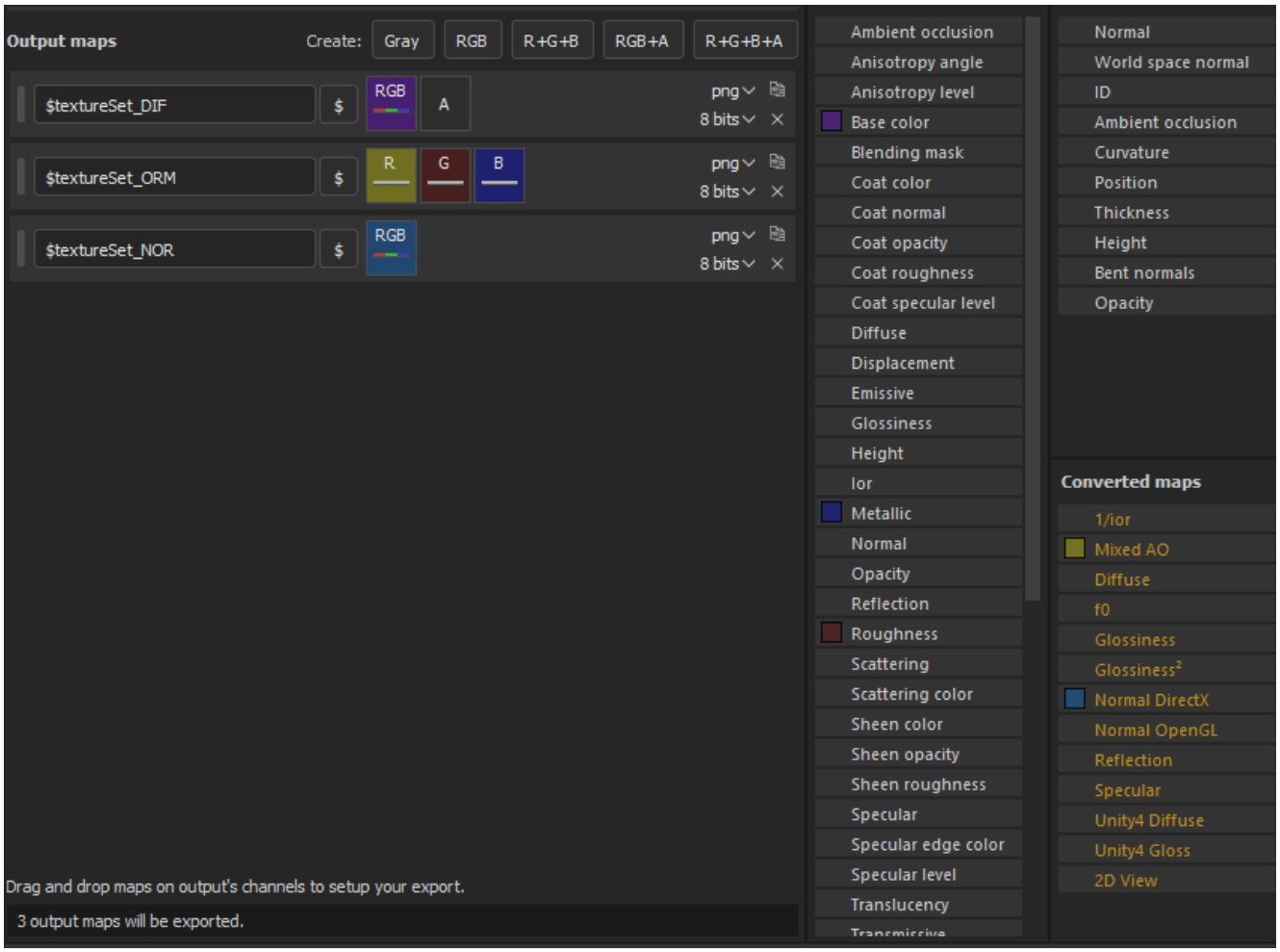


Рисунок 2.46 – Налаштування експорту ORM мап

2.3.12 Підготовка до ігрового рушія:

Перед імпортом в ігровий рушій, потрібно провести ще кілька важливих операцій з моделлю. Першою є об’єднання усіх деталей в один об’єкт, якщо ці деталі не плануються анімуватись чи розкладатись по сцені. Також потрібно призначити Origin point моделі був на “землі” у моделі, а сама модель стояла в “нулях” координат, як показано на (рис. 2.47).

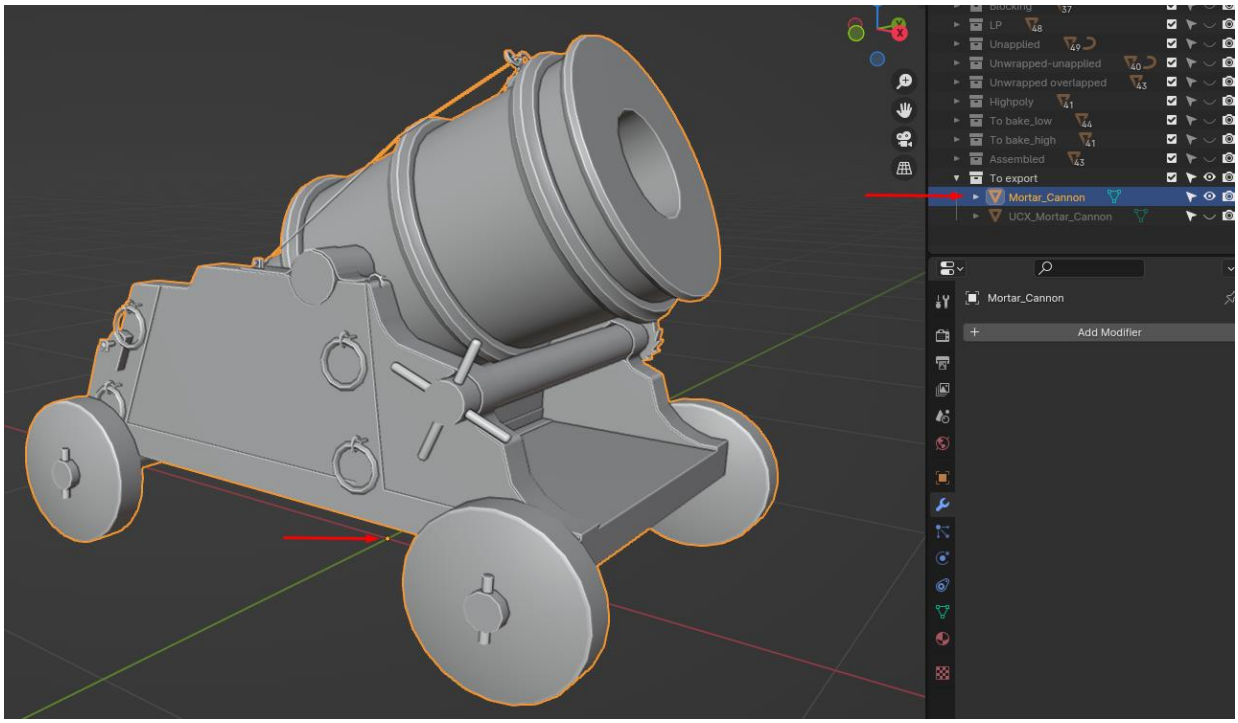


Рисунок 2.47 – Об’єднання моделі в один об’єкт і створення коректного неймінгу

Наступним етапом є створення колізії для моделі. Колізія – це невидима оболонка моделі, яка взаємодіє з іншими об’єктами та персонажами в сцені. Колізія визначає “хітбокс” моделі, як модель можна перестрибнути чи наступити персонажем, як показано на (рис. 2.48).

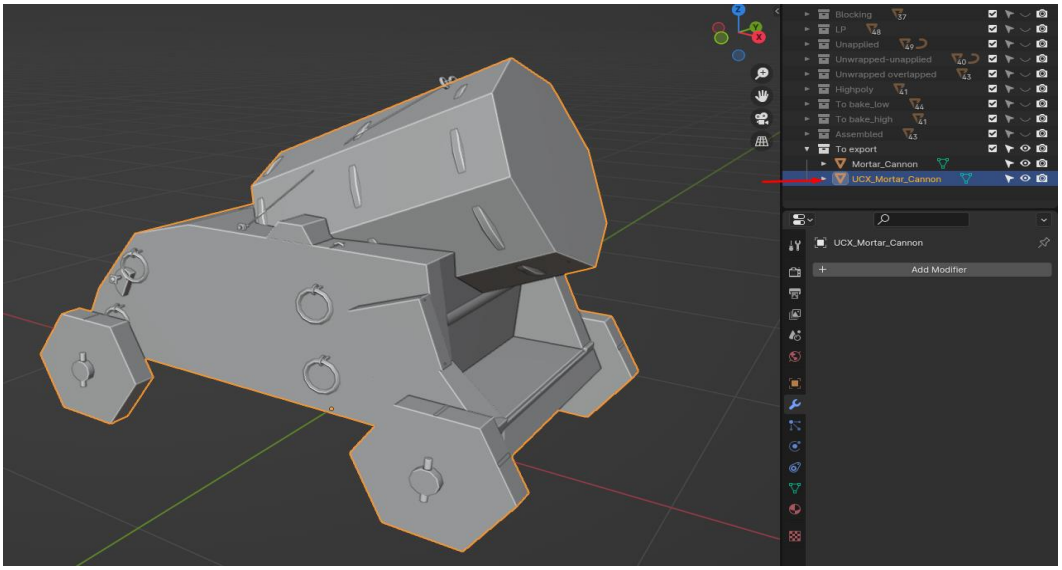


Рисунок 2.48 – Створення мешу колізії навколо моделі

Колізія повинна в загальних рисах передати форму основної моделі. Вона не повинна мати більше пари сотень трикутників. Також, потрібно дати колізії префікс UCX_, щоб Unreal Engine розпізнав її як колізію

Під час експорту, потрібно виділити модель і колізію і експортувати обидва об’єкти в один файл. Важливо дати ім’я файлу з префіксом SM_ для обозначення моделі як Static Mesh – статичний об’єкт, як показано на (рис. 2.49).

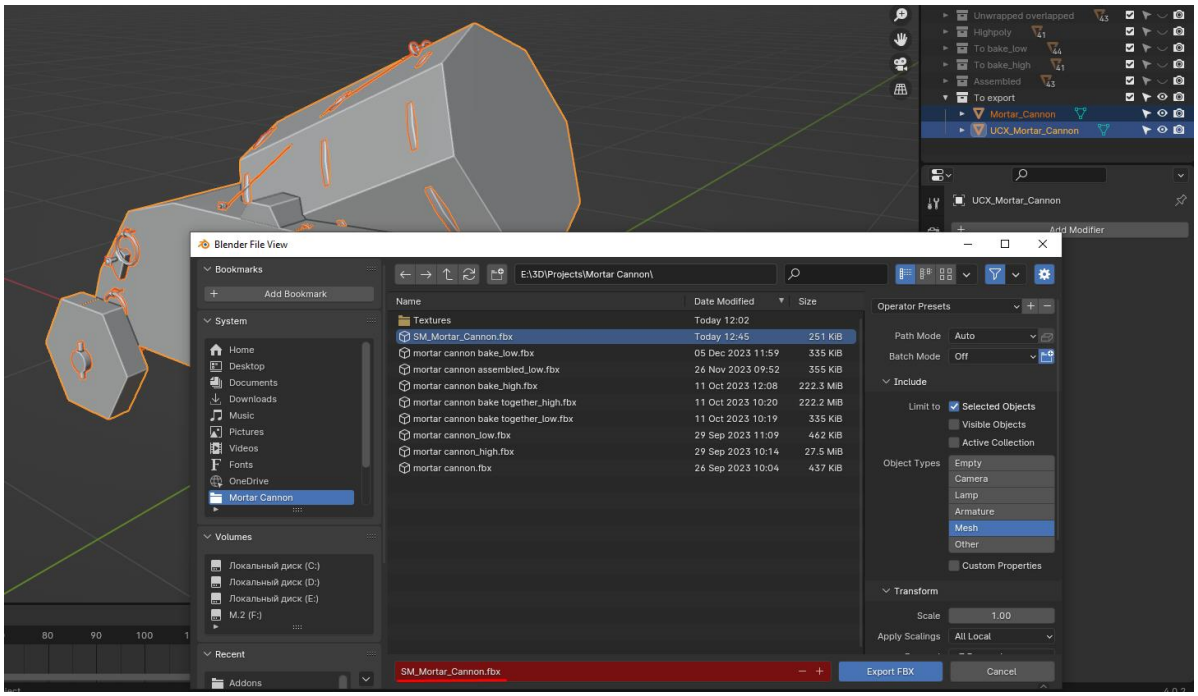


Рисунок 2.49 – Експорт моделі і мешу колізії для використання в Unreal Engine

Висновки до розділу: у цьому розділі було розглянуто основні техніки створення і оптимізації 3D моделі, які використовуються в ігровій індустрії. Було визначено цілі оптимізації моделі і переваги коректно оптимізованої моделі під час розробки відеоігор.

Було детально описано кожний етап створення 3D моделі та яких технік оптимізації слід дотримуватись на кожному з етапів пайплайну.

Як результат практичної частини другого розділу, було успішно створено оптимізовану модель старовинної гармати, готової до використання в ігровому рушії Unreal Engine 5.

3 СТВОРЕННЯ ІГРОВОГО РІВНЯ І РОБОТА З ОПТИМІЗОВАНОЮ 3D МОДЕЛЛЮ В UNREAL ENGINE 5

3.1 Створення ігрового рівня

Для початку роботи з ігровим рушієм Unreal Engine 5, потрібно визначитись з версією для роботи. Нами було обрано версію 5.2.1 через більшу кількість сумісних з нею наборів ассетів в магазині, як показано на (рис. 3.1).

Після визначення і загрузки потрібної версії рушія, потрібно створити новий проект. Для нашої сцени було використано пресет First Person Game, як показано на (рис. 3.2). Ці налаштування можна завжди змінити в самому рушії.

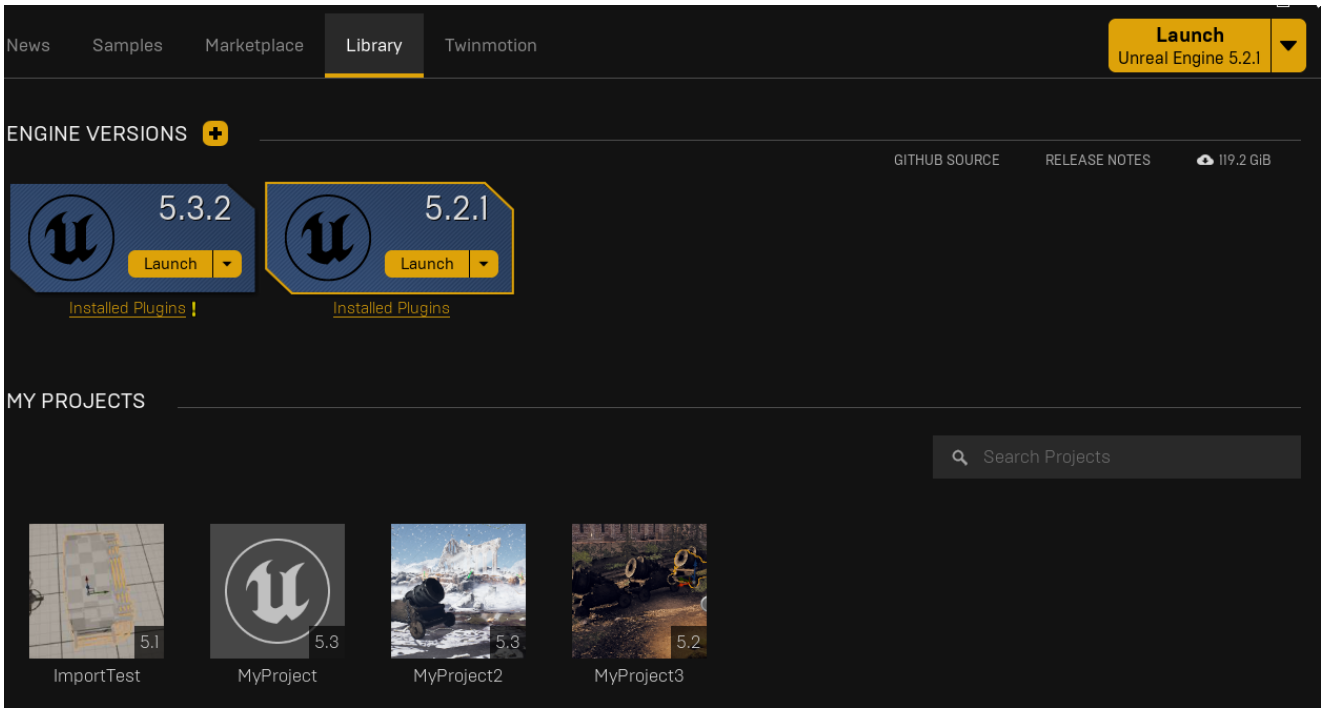


Рисунок 3.1 – Головна сторінка в The Unreal Engine Marketplace

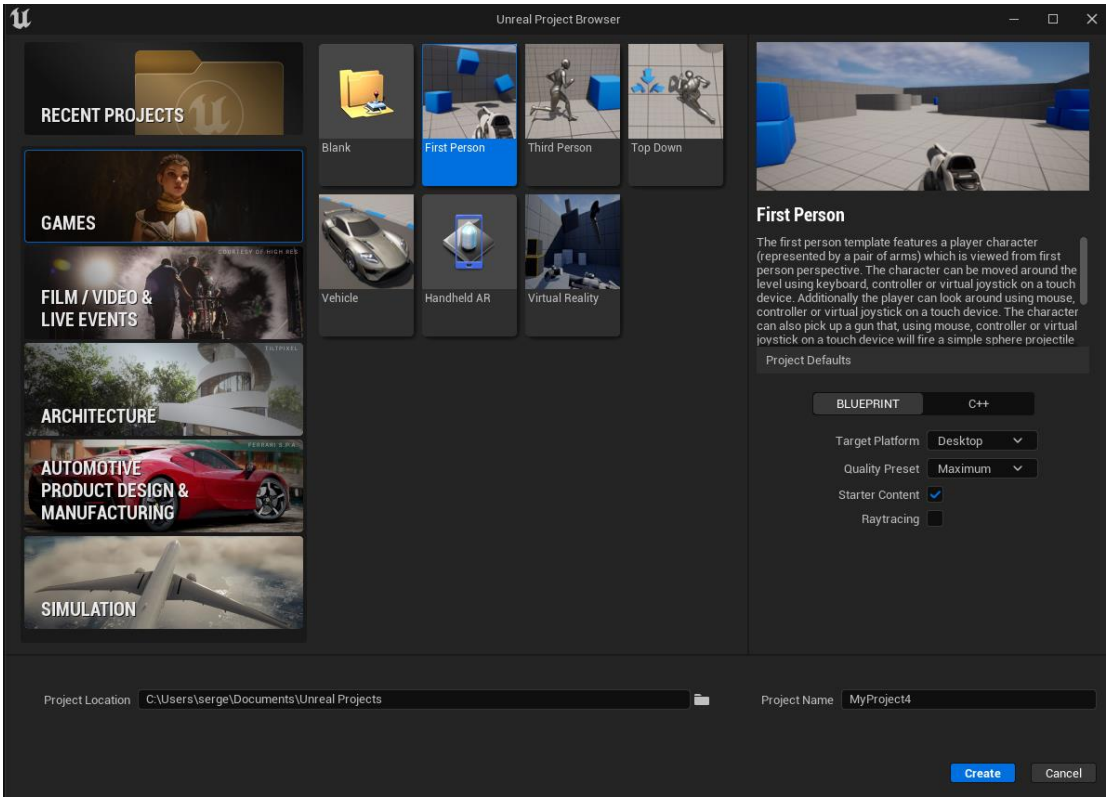


Рисунок 3.2 – Створення нового проекту в Unreal Engine 5

Після створення проекту, потрібно за бажанням додати набір ассетів з The Unreal Engine Marketplace. Було обрано набір “Sunset”, тому що в ньому були тематично сумісні моделі з нашою моделлю старовинної гармати, як показано на (рис. 3.3).

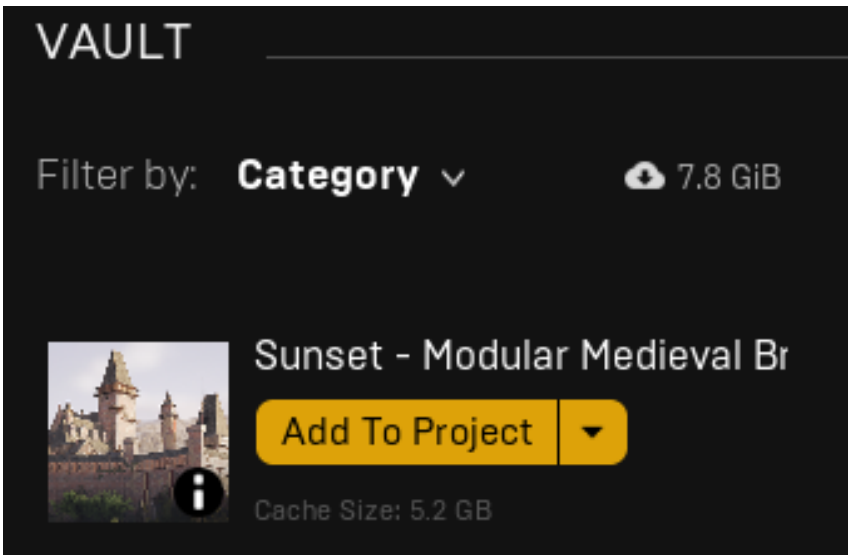


Рисунок 3.3 – Набір ассетів на сторінці The Unreal Engine Marketplace

Після імпорту набору ассетів, було оглянуто створений розробниками Showcase, або виставка моделей, присутніх в цьому наборі, як показано на (рис. 3.4).

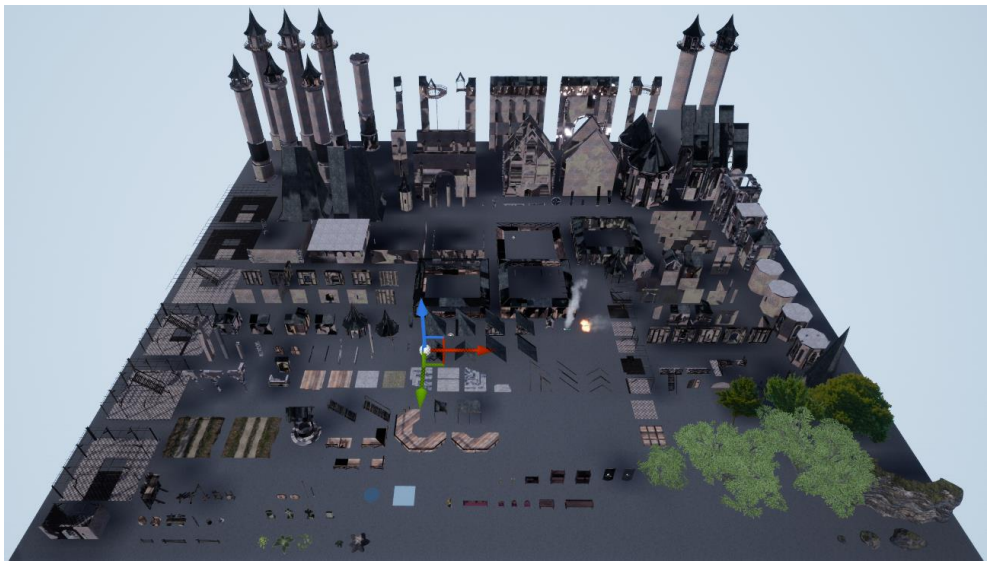


Рисунок 3.4 – Розкладений набір ассетів

Керуючись вже зробленими прикладами сцени, було створено підходящу сцену для розміщення моделі гармати. Було створено місце біля багаття поряд з військовими казармами замку, як показано на (рис. 3.5).

Рисунок 3.5 – Створена сцена з набору ассетів Sunset

3.2 Налаштування ігрового рівня і моделі

Для імпорту моделі, потрібно обрати файл і перетягнути у вікно рушія. Також треба використати функцію Build Nanite та налаштування імпорту текстур Do Not Create Material. Для створення матеріалу, потрібно приєднати карти текстур до відповідних вузлів матеріалу, як показано на (рис. 3.6). На карті нормалей потрібно вимкнути sRGB.

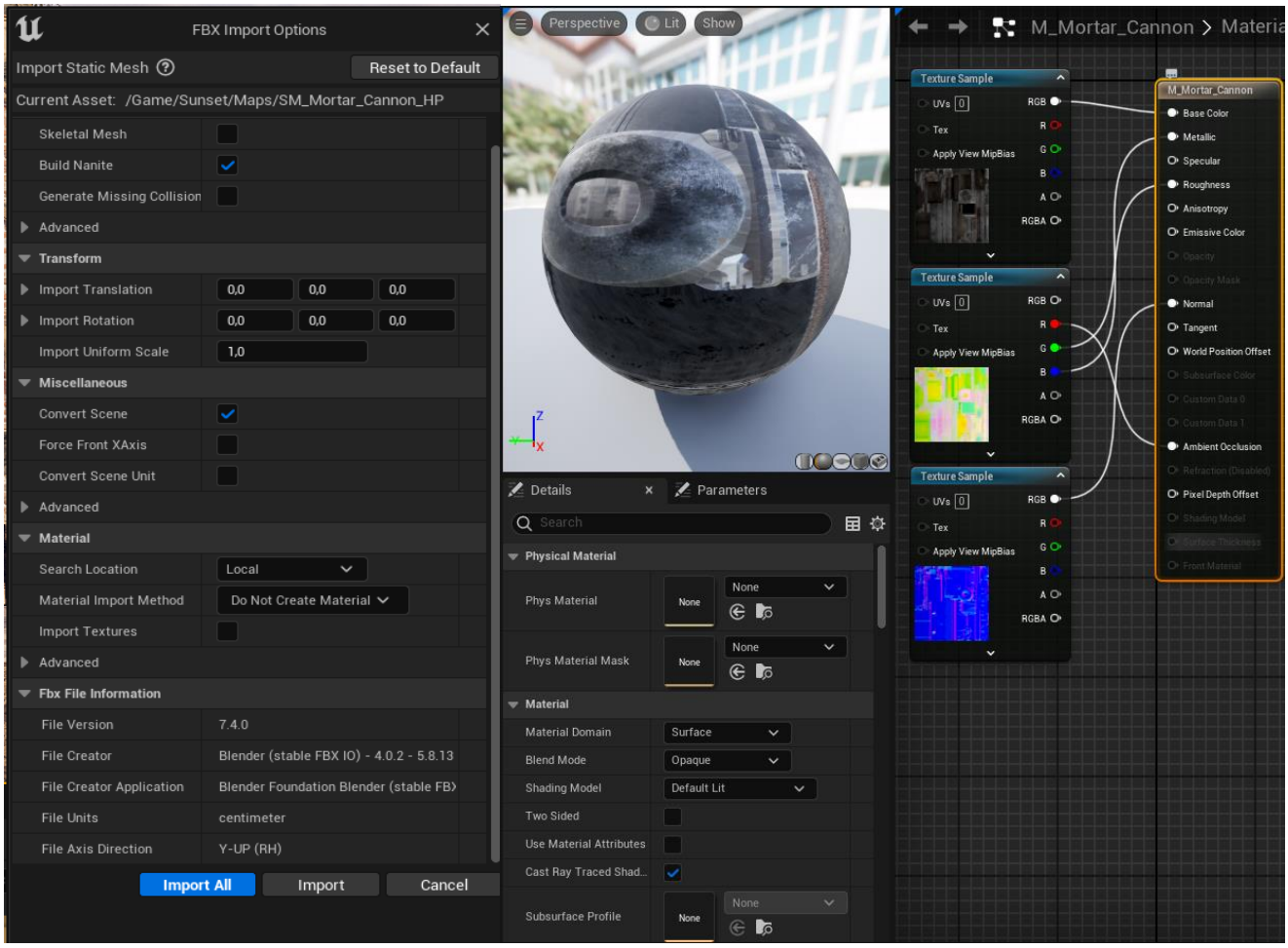


Рисунок 3.6 – Налаштування імпорту моделі і матеріалу в Unreal Engine 5

Імпортовану модель з призначеним матеріалом треба перетягнути з панелі Content Browser на ігровий рівень, як показано на (рис. 3.7).

Рисунок 3.7 – Імпортована модель в ігровому рівні

Щоб використати функцію Nanite для всіх Static Mesh, потрібно їх виділити у вікні Content Browser та включити налаштування Nanite. Це дозволить рушію спрощувати плотну геометрію моделей покращуючи продуктивність гри , як показано на (рис. 3.8).

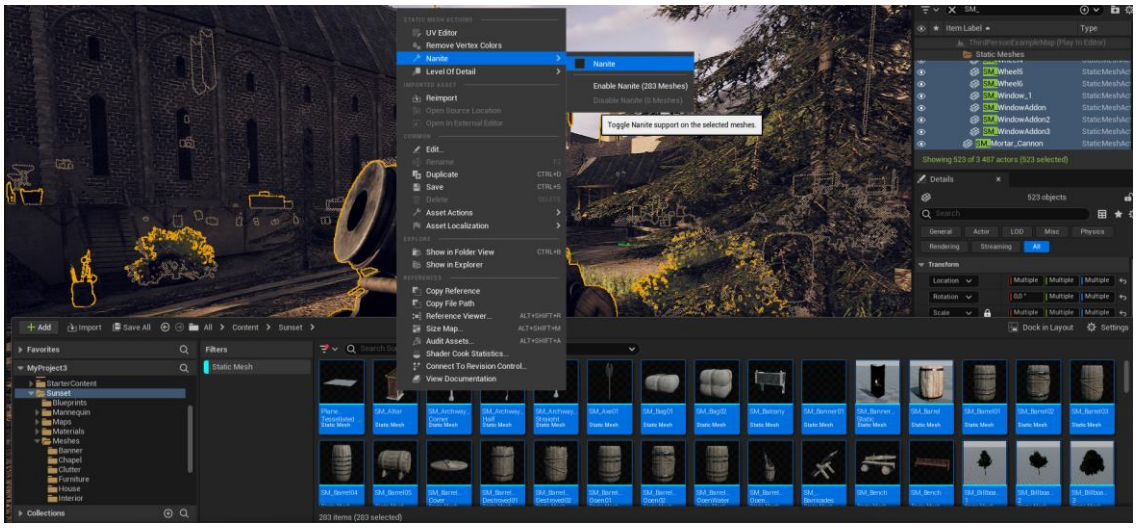


Рисунок 3.8 – Включення функції Nanite для всіх Static Mesh у сцені

Для подальшої оптимізації, потрібно побудувати карти світла Lightmaps і відбиття Reflections. Це запече статичне освітлення на всі моделі у сцені і зніме деяке навантаження з комп’ютеру гравця, як показано на (рис. 3.9).

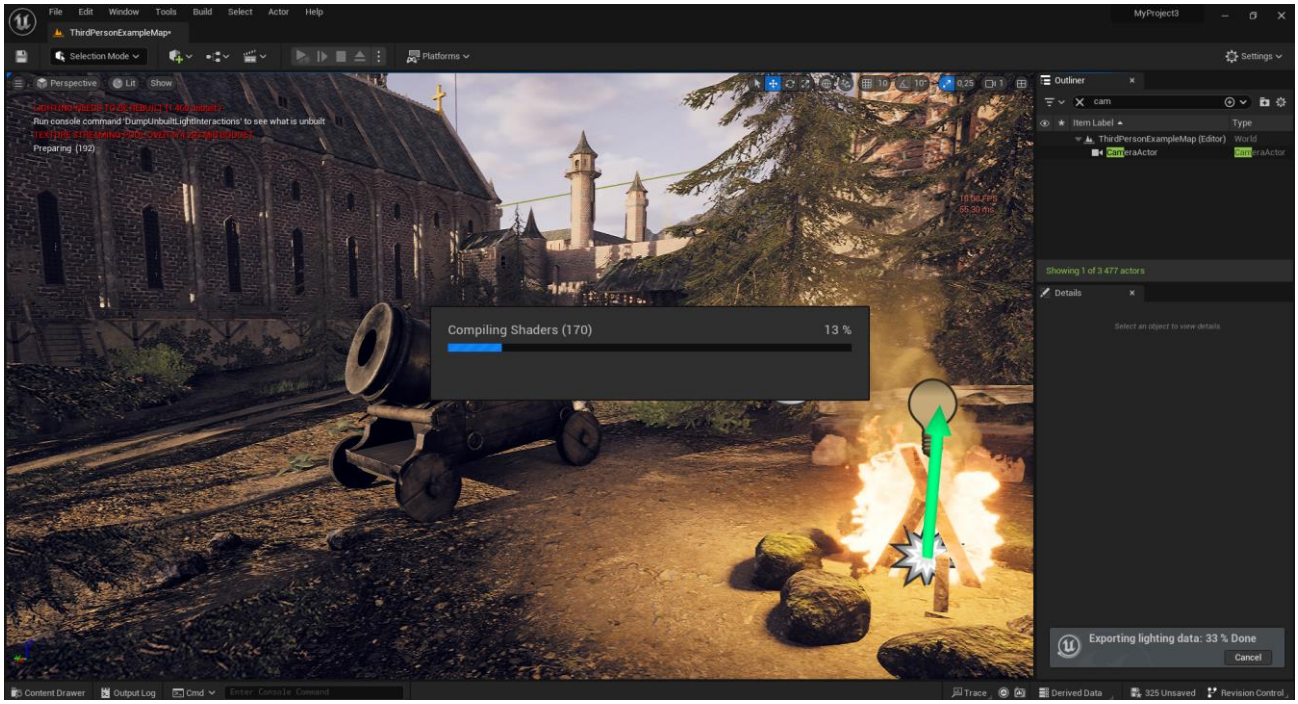


Рисунок 3.9 – Створення карт світла і відбиття для Lumen

3.3 Перевірка оптимізації моделі і ігрового рівня

Для перевірки оптимізації моделі і ігрового рівня, було використано дві комп’ютерні системи: середнього (С) і високого (В) технічного рівня. Обидві системи використовували монітор з роздільною здатністю 1920x1080p.

Таблиця 1 – Комп’ютерні системи для порівняння результатів

	Середній (С)	Високий (В)
Процесор	Ryzen 5 5600X, 4 ГГц	Ryzen 5 5600x, 4.4ГГц
Відеокарта	GTX 1660 Super 6 Гб	RTX 3080Ti 12гб
ОЗУ	32 Гб DDR4, 3600 МГц	32 Гб DDR4, 3600 МГц
Диск	1 Тб SSD	1 Тб SSD

3.3.1 Перевірка роботи на комп’ютері середнього технічного рівня “C”:

Ціллю експерименту є досягти стабільного значення FPS > 30, як показано на (рис. 3.10). Саме таке значення в індустрії є мінімальним прохідним значенням для деяких систем. Наприклад, для ігор на консолях, де упор робиться на візуальній частині, а не на швидкому геймплеї.

Рисунок 3.10 – Оптимізована модель у рушії на комп’ютері “C”

Результати порівняння різної кількості оптимізованих і неоптимізованих моделей було зображено на (рис. 3.11-3.19)

Рисунок 3.11 – Неоптимізована модель у рушії на комп’ютері “С”

Рисунок 3.12 – Неоптимізована модель у рушії на комп’ютері “С” з включеною функцією Nanite

Рисунок 3.13 – 18 оптимізованих моделей у рушії на комп’ютері “С”

Рисунок 3.14 – 5 неоптимізованих моделей у рушії на комп’ютері “С” з включеною функцією Nanite

Рисунок 3.15 – 5 неоптимізованих моделей у рушії на комп’ютері “С”

3.3.2 Перевірка роботи на комп’ютері високого технічного рівня “В”:

Рисунок 3.16 – Оптимізована модель у рушії на комп’ютері “В”

Рисунок 3.16 – Неоптимізована модель у рушії на комп’ютері “B”

Рисунок 3.17 – 5 оптимізованих моделей у рушії на комп’ютері “B”

Рисунок 3.18 – 5 неоптимізованих моделей у рушії на комп’ютері “B”

Рисунок 3.19 – 5 неоптимізованих моделей у рушії на комп’ютері “B” з включеною функцією Nanite

Таблиця 2 – Порівняння результатів симуляції на двох комп’ютерних системах з різними технічними характеристиками

	Середній (FPS)	Високий (FPS)
1 оптимізована	33	95
1 неоптимізована	24	72
5 оптимізованих	32	91
5 неоптимізованих	23	66
5 неоптимізованих з Nanite	40	90

Висновки до розділу: в третьому розділі було виконано практичну частину імплементації оптимізованої моделі в ігровий рушій Unreal Engine 5. Було налаштовано матеріал для моделі гармати за допомогою попередньо створених текстур. Було побудовано ігровий рівень і сцену для проведення симуляції.

Було створено 5 умов для експериментального порівняння продуктивності ігрового рівня на двох різних комп’ютерних системах з середнім і високим рівнем технічних характеристик.

Технічні характеристики порівнюваних систем та результати симуляцій були записані у формі двох таблиць.

ВИСНОВКИ

В атестаційній роботі було розглянуто найпопулярніші ігрові рушії в ігровій індустрії, такі як Unreal Engine, Unity, Godot, Lumberyard та Cryengine. Були висвітлені слабкі і сильні сторони кожного рушія і було зроблено висновки щодо оптимального рушія для створення фотореалістичних ігор.

Було детально розглянуто такі функції Unreal Engine, як Nanite, Lumen, Virtual Shadow Maps, Chaos Physics, World Partition, MetaHuman, MetaSounds та інші, що виокремлюють його з-поміж конкурентів в індустрії

Було розглянуто основні техніки створення і оптимізації 3D моделі, які використовуються в ігровій індустрії. Реалізуючи ці техніки створення і оптимізації, в практичній частині роботи, було успішно створено оптимізовану модель старовинної гармати, готової до використання в ігровому рушії Unreal Engine 5.

Використовуючи вбудовані функції Unreal Engine та ассети з The Unreal Engine Marketplace, було створено ігровий рівень та було створено сцену для проведення аналізу і порівняння продуктивності з оптимізованою моделлю.

Було проведено експериментальне порівняння продуктивності з оптимізованою та неоптимізованою моделлю в створеній сцені, а результати експерименту були записані в таблиці.

ПЕРЕЛІК ДЖЕРЕЛ, ПОСИЛАНЬ

1. Are You Not Entertained? [Інтернет ресурс]. — 2023 — Режим доступу до ресурсу: https://www.statista.com/chart/22392/global-revenue-of-selected-entertainment-industry-sectors/ (дата звернення: 20.11.2023)
2. Most Popular Game Engines [Інтернет ресурс]. — 2023 — Режим доступу до ресурсу:https://www.perforce.com/blog/vcs/most-popular-game-engines (дата звернення: 30.11.2023)
3. What Is Unreal Engine 5 (UE5)? [Інтернет ресурс]. — 2023 — Режим доступу до ресурсу: https://www.perforce.com/blog/vcs/unreal-engine-5#:~:text=Compared%20to%20Unreal%20Engine%204,enhanced%20animation%20and%20development%20tools.(дата звернення: 4.12.2023)
4. What Makes Unreal Engine So Good [Інтернет ресурс]. — 2023 — Режим доступу до ресурсу: https://bulldogjob.com/readme/unreal-engine-5 (дата звернення: 10.12.2023)
5. UNREAL GAME DEVELOPMENT: THE BENEFITS OF USING UNREAL ENGINE [Інтернет ресурс]. — 2023 — Режим доступу до ресурсу: https://agate.id/unreal-game-development-the-benefits-of-using-unreal-engine/ (дата звернення: 15.12.2023)
6. Unreal Engine 5 Features and How it Can Improve Gaming? [Інтернет ресурс]. — 2023 —Режим доступу до ресурсу: https://starloopstudios.com/unreal-engine-5-features-and-how-it-can-improve-gaming/ (дата звернення: 17.12.2023)
7. Building Virtual Worlds [Інтернет ресурс]. — 2023 — Режим доступу до ресурсу: https://docs.unrealengine.com/5.3/en-US/building-virtual-worlds-in-unreal-engine/ (дата звернення: 19.12.2023)
8. Branch Education [Інтернет ресурс]. — 2023 — Режим доступу до ресурсу: https://branch.education/ (дата звернення: 25.12.2023)

9. Карташов В.М., Коритцев І.В., Олейніков В.М., Зубков О.В., Шейко С.А., Бабкін С.І., Обробка сигналів при пеленгації та визначенні дальності до малорозмірних БПЛА в оптичному та інфрачервоному діапазон АХ // Радіотехніка. (Харків). - 2020. - Вип. 202. - С. 125-135. DOI:10.30837/rt.2020.3.202.13
10. Карташов В.М., Коритцев І.В., Олейніков В.М., Зубков О.В., Шейко С.А., Бабкін С.І. ОПТИКО-ЕЛЕКТРОННІ МЕТОДИ ОБНА-РУШЕННЯ ПОВІТРЯНИХ ОБ'ЄКТІВ І ВИМІРЮВАННЯ ЇХ КООРДИ-НАТ// Радіотехніка. (Харків). - 2020. - Вип. 202. - С. 153-59. DOI:10.30837/rt.2020.3.202.16
11. Карташов В.М., Коритцев І.В., Олейніков В.М., Зубков О.В., Шейко С.А., Бабкін С.І. // Радіотехніка. (Харків). - 2020. - Вип. 202. - С. 136-146. DOI:10.30837/rt.2020.3.202.14
12. Карташов В.М., Харченко О.І., Чумаков В.І. Використання ефекту стохастичного резонансу для аналізу спектрів акустичного випромінювання малих безпілотних літальних апаратів // Радіотехніка. (Харків). - 2019. - Вип. 197. - С. 100-106.
13. Карташов В.М., Сидоров Г.І., Толстих Є.Г., Шаповалов С.В. Акустичний вимірювач швидкості вітру в атмосферному прикордонному шарі// Радіотехніка. (Харків). - 2019. - Вип. 199. - С. 54-58.
14. Карташов В.М., Посошенко В.А., Цехмістро Р.І., Тимошенко Л.П., Колендовська М.М. Методи орієнтації, навігації та контролю мобільних робототехнічних платформ// Радіотехніка. (Харків). - 2019. - Вип. 199. - С. 38-44.
15. Олейніков В.М., Зубков О.В, Карташов В.М., Коритцев І.В., Бабкін С.І., Шейко С.А, Селезньов І.С. Експериментальна оцінка ефективності аглоритмів пеленгування безпілотних літальних апаратів з акустичного випромінювання// Радіотехніка. (Харків). - 2019. - Вип. 199. - С. 29-37.
16. Карташов В.М., Олейніков В.М., Колендовська М.М., Тимошенко Л.П., Капуста А.І., Рибніков Н.В. Комплексування зображень для виявлення безпілотних літальних апаратів // Радіотехніка. (Харків). - 2020. - Вип. 201. - С. -120-129.

17. Карташов В.М., Олейніков В.М., Воронін В.В., Рябуха В.П., Капуста А.І., Рибніков Н.В., Селезньов І.С. Методи комплексної обробки та інтерпретації радіолокаційних, акустичних, оптичних та інфрачервоних сигналів безпілотних літальних апаратів // Радіотехніка. (Харків). - 2020. - Вип. 202. - С. 173-182-. DOI:10.30837/rt.2020.3.202.19
18. V. M. Kartashov, V. N. Oleynikov, S. A. Sheyko, I. V. Koryttsev, S. I. Babkin, O. V. Zubkov, "Peculiarities of small unmanned aerial vehicles detection and recognition," Telecommunications and Radio Engineering, 2019, V. 78, Iss. 9, pp. 771–781. DOI: 10.1615/TelecomRadEng.v78.i9.30.
19. V. N. Oleynikov, O. V. Zubkov, V. M. Kartashov, I. V. Korytsev, S. I. Babkin, S.A. Sheiko, "Investigation of detection and recognition efficiency of small unmanned aerial vehicles on their acoustic radiation," Telecommunications and Radio Engineering, 2019, V. 78, Iss. 9, pp. 759–770. DOI: 10.1615/TelecomRadEng.v78.i9.20.
20. Kartashov, V.M., Sidorov, G.I., Sheiko, S.A., Kolendovska, M.M., Sergienko O.Yu. Principles of Construction and Assessment of technical Characteristics of multi-Frequency atmospheric Sodar in the Humidity Measurement Mode / Telecommunications and Radio Engineering.- New York. - 2020.- Vol. 79, №4.- P.323-333. (стаття). DOI: 10.1615/TelecomRadEng.v79.i4.50.
21. Kartashov, V.M., Oleynikov V.N, Zubkov, O.V., Korytsev I.V., Babkin, S. I., Sheiko, S.A.,. Kolendovskaya, M.M. Spatial-temporal Processing of acoustic Signals of Unmanned Aerial Vehicles; Telecommunications and Radio Engineering, 2020. Vol. 79, Iss, 9, pp.769-780.
22. V.M. Semenets, V.M. Kartashov, V.I. Leonidov. Features of Acoustic Noise of Small Unmanned Aerial Vehicles // Telecommunications and Radio Engineering.- New York. - 2020.- Vol. 79, №11.- P. 985-995. DOI: 10.1615/TelecomRadEng.v79.i11.80 (стаття).
23. Oleynikov V.N., Kartashov, V.M., Babkin, S. I., Zubkov, O.V., Korytsev I.V., Sheiko, S.A.,. Seleznov I.S. Structure and Parameter Unmanned Aerial Vehicles

Sound Fields/ Telecommunications and Radio Engineering.- New York. - 2020.- Vol. 79, №17.- P.1539-1550. DOI: 10.1615/TelecomRadEng.v79.i17.50 (стаття).