

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

**«Створення комп'ютерної графіки для гри жанру
Rogue-lite»**

Виконав студент групи КНк-22-1
СЕМЕНЕЦЬ Віктор Дмитрович

Керівник роботи:
ПЕТРОСЯН Арсен Русланович

Рецензент:
ЛЕВКІВСЬКИЙ Віталій Леонідович

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

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

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

Здобувач вищої освіти: **СЕМЕНЕЦЬ Віктор Дмитрович**
Керівник роботи: **ПЕТРОСЯН Арсен Русланович**
Тема роботи: **«Створення комп'ютерної графіки для гри жанру Rogue-lite»**,
затверджена наказом закладу вищої освіти від **«25» січня 2024 р., №22/с**
Термін здачі закінченої роботи: **«27» червня 2024 р.**
Вихідні дані роботи (зазначається предмет і об'єкт дослідження): Об'єктом дослідження є використання технологій для створення комп'ютерної графіки для подальшого використання у грі "Forbidden Forest". Предметом дослідження є розробка комп'ютерної графіки для гри "Forbidden Forest" з допомогою спеціалізованого пакету програм Blender та Subtanse Painter.
Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв
1			
2			
3			

Календарний план

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

Здобувач вищої освіти **Віктор СЕМЕНЕЦЬ**
Керівник **Арсен ПЕТРОСЯН**

РЕФЕРАТ

Кваліфікаційна робота включає в себе процес створення комп'ютерної графіки та супровідну пояснювальну записку. Остання містить у собі вступ, три розділи, висновки, список використаної літератури та додаток. Текстова частина викладена на 60 сторінках друкованого тексту.

Пояснювальна записка має 12 сторінок додатків. Список використаних джерел містить 20 найменувань і займає 2 сторінки. В роботі наведено 69 рисунків. Загальний обсяг роботи – 79 сторінок.

Ключові слова: UNITY, ігровий додаток, програмне забезпечення, BLENDER, SUBSTANCE, PAINTER, 3D, TEXTURE, ADDON, ROUGELITE, UV.

					КН.КР.Б – 122 – 24 – ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата	Створення комп'ютерної графіки для гри жанру Rogue-lite Пояснювальна записка		
Розроб.	В. Д. СЕМЕНЕЦЬ						
Керівник	А. Р. ПЕТРОСЯН						
Рецензент	В.Л. ЛЕВКІВСЬКИЙ						
Н. контр.							
Зав. каф.	М.С. ГРАФ				Літ. Арк. Аркушів		
						3	79
					Житомирська політехніка, група КНк-22-1		

ABSTRACT

The graduation qualification work includes the process of creating computer graphics and an accompanying explanatory note. The latter contains an introduction, three sections, conclusions, a list of references, and an appendix. The textual part is presented on 60 pages of printed text.

The explanatory note consists of 12 pages of appendices. The list of used sources includes 20 titles and occupies 2 pages. The work includes 69 figures. The total volume of the work is 79 pages.

Keywords: UNITY, BLENDER, SUBSTANCE, PAINTER, 3D, TEXTURE, ADDON, ROUGELITE, UV.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	10
1.1 Постановка задачі.....	10
1.2. Аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи.....	10
1.3 Обґрунтування вибору інструментальних засобів.....	22
Висновки до першого розділу	23
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА КОМП'ЮТЕРНОЇ ГРАФІКИ	24
2.1 Візуалізація процесів розробки графічних елементів ігрового додатку	24
2.2 Розробка 3D-моделі для ігрового додатку.....	30
2.3 Створення графічного матеріалу для ігрового додатку	39
2.4 Реалізація програмного коду для оптимізації геометрії	48
Висновки до другого розділу	51
РОЗДІЛ 3. ОПИС ТЕСТУВАННЯ ГРАФІЧНОЇ СКЛАДОВОЇ ІГРОВОГО ДОДАТКУ	52
3.1 Основні процеси тестування графічної складової.....	52
Висновки до третього розділу	51
Висновки	62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	63
ДОДАТКИ.....	65

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ	Програмне забезпечення
UV	Текстурні координати
UML	(Unified Modeling Language) – це уніфікована мова моделювання, яка використовується для візуалізації
WIP	Work In Progress – малюнок або рендер незавершеної роботи

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми. Rogue-lite — це жанр відеоігор, який поєднує елементи класичних ігор жанру roguelike з деякими відступами від суворих правил цього жанру. Цей термін став популярним в останні роки і використовується для опису ігор, які зберігають основні характеристики roguelike, але при цьому додають або змінюють певні аспекти геймплею, роблячи їх більш доступними та різноманітними.

До основних характеристик, які визначають ігри жанру roguelite, належать:

Процедурна генерація рівнів: Як і в roguelike, у roguelite іграх рівні зазвичай генеруються випадковим чином, що забезпечує унікальність кожного проходження.

Перманентна смерть: Гравці втрачають значний прогрес після загибелі персонажа, але в roguelite часто залишаються певні елементи, які зберігаються між ігровими сесіями.

Розвиток персонажа: В іграх roguelite часто є система постійного прогресу, де гравці можуть покращувати свого персонажа чи відкривати нові можливості навіть після смерті.

Високий рівень складності: Roguelite ігри зазвичай відомі своїм викликом і вимогою до гравців проявляти навички та стратегічне мислення.

Метою кваліфікаційної роботи є створення оптимізація та тестування графічної складової для гри в жанрі Roguelike.

Приклади ігор жанру Roguelite:

Dead Cells: Ця гра поєднує елементи платформера і roguelite, пропонуючи гравцям процедурно згенеровані рівні, перманентну смерть, але з можливістю зберігати певні покращення між проходженнями.

Hades: Ігра від Supergiant Games, яка змішує roguelite з елементами історії та розвитку персонажа, де кожне проходження розкриває нові аспекти сюжету і надає нові можливості.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		7

The Binding of Isaac: Гра, що пропонує випадково згенеровані підземелля та різноманітні вдосконалення, які можна знаходити під час проходження, зберігаючи основні елементи roguelite.

Відмінності від класичних Roguelike:

Roguelike ігри мають суворі визначення, відомі як "Берлінська інтерпретація", що включає такі характеристики як процедурна генерація, перманентна смерть, покроковий геймплей та відсутність збережень під час гри. Roguelite ж дозволяє більше відступів від цих правил, надаючи більшу гнучкість в дизайні гри:

Більше збережень: у багатьох roguelite іграх можна зберігати прогрес або відкривати постійні вдосконалення, що полегшує гру після кожної спроби.

Більш різноманітний геймплей: Roguelite часто експериментує з різними стилями геймплею, поєднуючи елементи інших жанрів, таких як екшн, платформер чи RPG.

Меньше випадковості: хоча процедурна генерація залишається, roguelite часто містить більш контрольовану випадковість або певні передбачувані елементи, що допомагають гравцям стратегічно планувати свої дії.

Об'єктом дослідження є використання технологій для створення комп'ютерної графіки для подальшого використання у грі "Forbidden Forest".

Предметом дослідження є розробка комп'ютерної графіки для гри "Forbidden Forest" з допомогою спеціального пакету програм Blender та Substance Painter.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Постановка задачі

Метою кваліфікаційної роботи є створення та тестування комп'ютерної графіки для гри в жанрі Roguelike.

Основною метою розробки є створення стилізованої графіки для гри в жанрі Roguelite що дозволяє глибше зрозуміти та проникнутися ігровим всесвітом а також це включає в себе:

Універсальність стилізованої графіки: Roguelite – ігри часто використовують стилізовану графіку тому що вона поєднує в собі безліч корисних аспетів наприклад оптимізація що дозволяє гравцям з різними конфігураціями отримати повний досвід користування ігровим додатком.

Зовнішній вигляд: стилізована графіка навідміну від реалістичної графіки дає більший простір для експериментів та художньої інтерпритації автора що в свою чергу частіше використовується інді розробниками.

Простота графічної кладової: Стилiзована графіка сприяє швидкому та легкому зануренню в ігровий процес і допомагає орієнтуватися в ігровому середовищі.

Менша складність при розробці: навідміні від реалістичної графіки стилізована може допускати певні відмінності яких не може бути при розробці реалістиної графіки.

У рамках кваліфікаційної роботи передбачається розробка власних різноманітних ігрових моделей які в подальшому будуть використані в ігровому додатку.

Необхідні етапи розробки:

1. Створення моделей для подальшого використангня в ігровому додатку.
 - Розробка та аналіз концепт документа.
 - Планування розробки моделі.
 - Розробка унікальної моделі.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		9

- Диверсифікація типів моделей.
2. Перевірка цілісності моделі відносно стилю та концепта.
- Перевірка на стильову ідентичність.
 - Тестування моделі на придатність перед подальшими маніпуляціями.
3. Фінальні маніпуляції перед внесенням в ігрове середовище.
- Фінальна перевірка на відсутність технічних помилок.
 - Оптимізація ігрової моделі.
 - Створення високополігональної версії моделі для спрощення накладання текстур.
4. Приклад розроблених моделей:
- RG_gate – арка для переходу між аренами
 - RG_mug – кухоль який використовується в хаб локації
 - TR_Tabble_tawern – барна стійка в хаб локації
 - TR_roof_tiles – дах для таверни в хаб локації
 - TR_wall – одна зі стін з яких побудована хаб локація
 - TR_Torch – факел що освітлює приміщення
 - Big_door – двері для переходу між локаціями

1.2. Аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи.

Для порівняння підходять декілька ігр з художнім стилем графіки.

"Divinity: Original Sin" [1] — це рольова відеогра (RPG), розроблена і видана бельгійською компанією Larian Studios. Вона вийшла у 2014 році і здобула визнання критиків та гравців за глибокий сюжет, розгалужену систему бою та високу інтерактивність.

Гра розгортається у фантастичному світі Рівеллон, де гравці беруть на себе ролі Джерелошукачів — мисливців на магів Джерела (рис. 1.1).

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		10



Рисунок 1.1 - Скріншот з гри "Divinity: Original Sin"

Ігра має барвисту ізометричну графіку, що допомагає створити атмосферу фентезійного світу.

Основні аспекти графіки "Divinity: Original Sin":

Диз пропорціональні розміри та неприродня гладкість моделей оточення та персонажів занурюють гравця в атмосферу казковості та надають умовної чарівності всьому що з знаходиться на екрані у гравця (рис. 1.2).

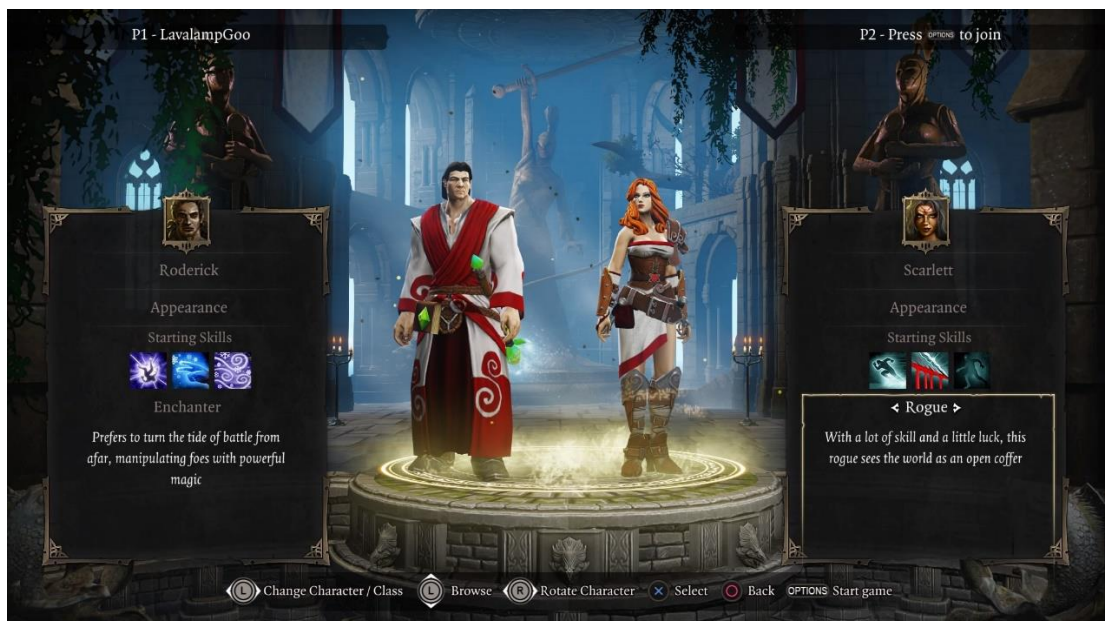


Рисунок 1.2 - Приклад графічного оздоблення "Divinity: Original Sin"

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		11

Різноманітні візуальні ефекти під час використання магії або ведення ближнього бою надають глибини ігровому процесу та підкреслюють казковість візуального стилю гри (рис 1.3).



Рисунок 1.3 - Приклад візуальних ефектів "Divinity: Original Sin"

Ізометрична перспектива: Гра використовує ізометричний ракурс, що забезпечує широкий огляд і дозволяє гравцям краще планувати свої дії а також дає насолодитися проробленістю та переднього та задніх планів ігрових локацій (рис. 1.4).



Рисунок 1.4 - Приклад багатошарової побудови локації "Divinity: Original Sin"

Різноманітність середовищ: Гра пропонує різноманітні локації, від міст і сіл до підземель, лісів та узбереж. Кожна локація має унікальний дизайн, що відображає її функції та історію.

Деталізація оточення: Об'єкти оточення, такі як рослинність, будівлі, елементи інтер'єру, ретельно розроблені і додають глибинності світу.

Екологічні взаємодії: Вода, вогонь, дим, лід та інші елементи можуть взаємодіяти один з одним, створюючи додаткові тактичні можливості та впливаючи на хід бою.

Графічні налаштування: Гра пропонує широкий спектр графічних налаштувань, що дозволяє гравцям налаштовувати якість графіки відповідно до можливостей їхнього обладнання.

Висока продуктивність: Гра добре оптимізована, що забезпечує плавний ігровий процес навіть на середніх конфігураціях комп'ютерів.

Художнє оформлення: Використання класичного фентезійного стилю з елементами магії та середньовічної архітектури додає грі автентичності та занурює гравця в світ (рис. 1.5).



Рисунок 1.5 - Приклад сетінгу "Divinity: Original Sin"

Хоча "Divinity: Original Sin" здобула багато позитивних відгуків за свої графічні рішення, у неї є кілька графічних проблем, з якими стикалися гравці. Ось деякі з основних графічних проблем, про які повідомляли користувачі:

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		13

Артефакти і візуальні глітчі:

Відображення текстур: У деяких випадках текстури можуть відображатися некоректно, бути низької якості або зовсім не завантажуватися.

Пропадання об'єктів: Об'єкти навколишнього середовища або персонажі можуть зникати або мерехтіти, що заважає ігровому процесу і порушує занурення у гру.

Проблеми з освітленням і тінями:

Неправильні тіні: Тіні можуть відображатися некоректно, з'являтися і зникати без причини або бути надто темними/світлими.

Динамічне освітлення: У деяких сценах динамічне освітлення може працювати некоректно, що призводить до нерівномірного освітлення або блимаючих ефектів.

Водні ефекти та відображення:

Відображення на воді: Відображення на водних поверхнях можуть виглядати спотвореними або неправильно відображати навколишнє середовище.

Якість водних ефектів: Водні ефекти можуть виглядати низькоякісними або неправильно взаємодіяти з іншими елементами гри.

Незважаючи на ці проблеми, більшість гравців все одно відзначають високу якість графіки "Divinity: Original Sin" та її здатність створювати захоплюючий ігровий досвід.

"Deer Rock Galactic" [2] – це кооперативна гра на виживання від першої особи, розроблена компанією Ghost Ship Games і видана Coffee Stain Publishing. Вона вийшла у ранньому доступі у лютому 2018 року, а повний реліз відбувся у травні 2020 року. Гра доступна на платформах Windows і Xbox One, а також була випущена для PlayStation 4 і PlayStation 5 у січні 2022 року.

У "Deer Rock Galactic" гравці спускаються в підземні печери для виконання різних місій. Кожна місія має свої цілі, наприклад, збір певних ресурсів або знищення конкретних ворогів. Гравці повинні співпрацювати, використовуючи унікальні здібності своїх класів, щоб успішно виконати завдання і повернутися на поверхню живими.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		14

"Deer Rock Galactic" має унікальний і впізнаваний стиль графіки, який відрізняється від багатьох інших ігор жанру. Ось основні графічні аспекти гри:

Низькополігональна графіка: Гра використовує стиль низької полігональної деталізації. Це означає, що об'єкти, персонажі та оточення мають чіткі, виразні форми з меншою кількістю деталей, що створює своєрідну художню естетику (рис. 1.6).



Рисунок 1.6 - Скріншот з гри "Deer Rock Galactic"

Яскраві кольори та контрастні палітри: Печери та підземні локації наповнені яскравими кольорами, які допомагають виділити різні елементи оточення та створюють вражаючу атмосферу. Це включає яскраво освітлені кристали, рідкісні ресурси та ворогів, які контрастують з темними, глухими коридорами печер.

Освітлення та тіні [16]: Важливий аспект графіки – це використання динамічного освітлення і тіней. Оскільки дії відбуваються під землею, освітлення відіграє ключову роль у створенні атмосфери та навігації. Гравці використовують ліхтарі та світлові гранати для освітлення темних ділянок, що додає напруженості і реалістичності (рис. 1.7).

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		15



Рисунок 1.7 - Приклад освітлення "Deer Rock Galactic"

Візуальні ефекти: У грі багато ефектів, таких як частинки пилу, вибухи, іскорки від інструментів і зброї. Ці ефекти додають динамічності та допомагають передати відчуття небезпеки та інтенсивності.

Динамічні ландшафти [20]: Печери та підземелля генеруються процедурно, що означає, що кожна місія пропонує нові, унікальні локації з різними ландшафтами. Це включає природні формування, тунелі, великі підземні зали та інші цікаві структури (рис. 1.8).

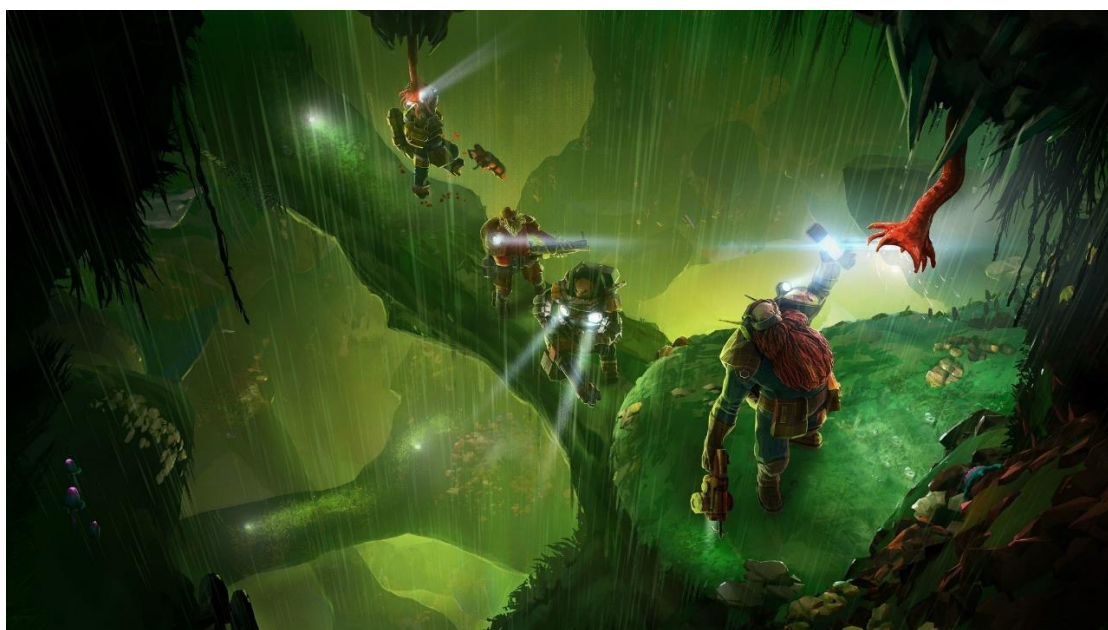


Рисунок 1.8 - Приклад генерації локації "Deer Rock Galactic"

		В.Д.Семенець		
		А.Р.Петросян		
Змн.	Арк.	№ докум.	Підпис	Дата

КН.КР.Б – 122 – 24 – ПЗ

Арк.

16

Стилізований дизайн ворогів: Вороги у грі мають виразний, стилізований вигляд, що підкреслює їхню загрозу та унікальність. Від величезних павуків до гігантських комах – кожен ворог має свій унікальний дизайн (рис. 1.9).



Рисунок 1.9 - Приклад стилізації ворогів "Deer Rock Galactic"

Загалом, графічний стиль "Deer Rock Galactic" добре доповнює її ігровий процес, створюючи виразну та захоплюючу атмосферу, яка підкреслює кооперативну природу гри та інтенсивність підземних пригод.

Хоча графічний стиль "Deer Rock Galactic" має багато переваг, деякі гравці можуть знайти його недоліки. Ось основні з них:

Низька деталізація: Низькополігональний стиль, хоч і естетично привабливий, може здатися спрощеним для тих, хто звик до високодеталізованих графічних ігор.

Обмежені текстури: Використання простих текстур може здаватися монотонним, особливо після тривалих сеансів гри. Деякі гравці можуть вважати, що різноманітність текстур і деталізація поверхонь могла б бути більш насиченою.

Одноманітність оточення: Незважаючи на процедурно згенеровані світи, деякі печери можуть здатися схожими одна на одну. Це може створити відчуття повторюваності і вплинути на різноманітність геймплею.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		17

Візуальний безлад: Під час інтенсивних боїв і великих сутичок кількість візуальних ефектів, таких як вибухи, частинки пилу та світлові ефекти, може бути надмірною. Це іноді призводить до того, що екран стає занадто захаращеним, ускладнюючи орієнтацію та сприйняття ситуації.

Стиль на любителя: Низькополігональний стиль і яскраві кольори можуть не сподобатися всім гравцям. Деякі можуть вважати цей стиль занадто мультяшним або несерйозним, що може вплинути на їхнє загальне сприйняття гри.

Незважаючи на ці недоліки, "Deer Rock Galactic" залишається популярною завдяки своїй кооперативній грі, унікальній атмосфері та захоплюючому геймплею. Багато гравців вважають, що її графічний стиль добре відповідає загальній концепції гри, хоча він може не сподобатися всім.

"Serious Sam 2" [3] – це шутер від першої особи, розроблений хорватською компанією Croteam і виданий 2K Games. Гра вийшла у жовтні 2005 року для платформ Windows і Xbox. Це продовження популярної серії ігор "Serious Sam", відомої своїм динамічним геймплеєм та ордами ворогів.

"Serious Sam 2" вирізняється яскравим і барвистим графічним стилем, який значно відрізняється від більш реалістичних шутерів того часу. Ось основні графічні аспекти гри:

Дизайн ворогів та персонажів: яскравий і карикатурний, що підкреслює гумористичний і абсурдний стиль гри. Вороги варіюються від маленьких і швидких істот до гігантських босів з вражаючими деталями (рис. 1.10).



Рисунок 1.10 - Приклад стилізації головного героя "Serious sam 2"

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Яскраві та насичені кольори, що створює живий і динамічний візуальний стиль. Кожен рівень має свій унікальний колірний стиль, що допомагає створити відчуття різноманітності та новизни (рис. 1.11).



Рисунок 1.11 - Скріншо з гри "Serious sam 2"

Різнноманітні локації, включаючи джунглі, пустелі, футуристичні міста, космічні станції та інші екзотичні місця. Кожен рівень має свої унікальні ландшафти та архітектурні особливості.

Ефекти частинок [17]: гра використовує різноманітні ефекти частинок для вибухів, стрільби, крові та інших динамічних подій. Ці ефекти додають інтенсивності бойовим сценам і роблять їх більш видовищними (рис. 1.12).



Рисунок 1.12 - Скраншо з гри "Serious sam 2"

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Використання динамічного освітлення та тіней додає глибини і реалістичності сценам. Освітлення змінюється в залежності від локацій, підкреслюючи атмосферу кожного рівня.

Гра багата на різноманітні об'єкти та декорації, які заповнюють ігрові рівні. Це включає рослинність, будівлі, технологічні об'єкти та інші деталі, що створюють атмосферу кожної локації.

Різнманітність стилів рівнів: кожна планета та місце має свій унікальний візуальний стиль, що відображає різноманітність ігрового світу. Це допомагає підтримувати інтерес гравця і робить кожен рівень унікальним (рис. 1.13).



Рисунок 1.13 - Постер на якому зображено основні локації "Serious sam 2"

Загалом, графічний стиль "Serious Sam 2" підкреслює веселий та динамічний характер гри, пропонуючи яскраву і захоплюючу візуальну подорож, що добре доповнює швидкий і насичений екшен-геймплей.

Незважаючи на багато позитивних аспектів графіки "Serious Sam 2", у грі є кілька недоліків, які варто відзначити:

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Застаріла графіка: на момент виходу, хоча графіка була покращена порівняно з попередніми частинами, вона все одно виглядала дещо застарілою порівняно з іншими сучасними шутерами. Деякі текстури можуть здаватися низькороздільними, а моделі персонажів і ворогів – простими.

Світлові ефекти та тіні: хоча гра використовує динамічне освітлення та тіні, у деяких місцях вони можуть виглядати недостатньо реалістично або некоректно. Це може вплинути на загальне враження від графіки.

Проблеми з текстурами: у деяких локаціях можна помітити проблеми з завантаженням текстур або їх недостатньою деталізацією. Це може негативно вплинути на візуальну привабливість гри.

Візуальний безлад: під час інтенсивних боїв з великою кількістю ворогів і вибухів на екрані може виникати візуальний безлад, що ускладнює орієнтацію і сприйняття бойової ситуації.

Попри ці недоліки, "Serious Sam 2" все одно залишається захоплюючим і веселим шутером, який багато гравців цінують за його динамічний геймплей і гумористичний стиль. Графічні недоліки не завжди сильно впливають на загальне враження від гри, особливо для тих, хто цінує її унікальну атмосферу та інтенсивні бойові сцени.

1.3 Обґрунтування вибору інструментальних засобів.

Станом на 2024 рік, ринок програмного забезпечення (ПЗ) для 3D-моделювання продовжує активно розвиватися, відображаючи потреби та тренди в різних галузях, таких як архітектура, дизайн, ігрова індустрія, анімація, інженерія та медицина.

Основні гравці на ринку та їх продукти:

3ds Max: широко використовується в архітектурі, дизайні інтер'єрів та в ігровій індустрії.

Maya: популярний серед аніматорів і студій, що займаються створенням відеоігор та кіно.

AutoCAD: використовується для створення 2D і 3D креслень в архітектурі та інженерії.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Blender: [4] безкоштовне та відкрите ПЗ, яке набуло популярності завдяки потужному набору інструментів для моделювання, анімації та рендерингу. Широко використовується як професіоналами, так і любителями.

SketchUp: відомий своєю простотою у використанні, популярний серед архітекторів і дизайнерів інтер'єрів. Використовується для швидкого створення концептуальних моделей.

SolidWorks: Використовується в інженерії та проектуванні продуктів, особливо в машинобудуванні.

CATIA: Використовується в аерокосмічній і автомобільній промисловості для складного моделювання і проектування.

Houdini: Популярний у візуальних ефектах та ігровій індустрії завдяки своїй процедурній природі та потужним інструментам для створення складних симуляцій.

Було обрано Blender та Substance 3D Painter [12]:

1. Гнучкий та простий в освоєнні інтерфейс
2. Відкритий код програмного забезпечення Blender надає можливість самостійно доповнювати програму необхідними та зручними для користувача функціями.
3. Підтримка численних плагінів та аддонів, які дозволяють розширити функціонал Blender відповідно до потреб конкретного проекту.
4. Blender підтримує різні формати файлів, що дозволяє легко інтегрувати його в існуючі робочі процеси та обмінюватися файлами з іншими програмами для 3D-моделювання та рендерингу.
5. Blender працює на різних операційних системах, включаючи Windows, macOS і Linux, що забезпечує гнучкість для користувачів.
6. Регулярні оновлення та нові функції завдяки внескам від спільноти та розробників.
7. Вбудований редактор відео (Video Sequence Editor) дозволяє редагувати відео та створювати композитні проекти безпосередньо в Blender.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				22
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до першого розділу

У ході проекту було проаналізовано 3 відомі гри ("Divinity: Original Sin", "Deer Rock Galactic", "Serious sam 2") що допомогло визначити основні відмінності стилізованої графіки від реалістичного стилю і з допомогою визначення цих відмінностей вдалося покращити візуальну частину ігрового продукту.

Велика увага була приділена вибору програмного забезпечення. Були вибрані 2 основні програми в яких буде вестися розробка моделей а саме Substance 3D Painter та Blender.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		23

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА КОМП'ЮТЕРНОЇ ГРАФІКИ

2.1 Візуалізація процесів розробки графічних елементів ігрового додатку

Діаграми UML (Unified Modeling Language) — це стандартний спосіб візуалізації дизайну системи. Вони використовуються для моделювання об'єктно-орієнтованого програмного забезпечення та бізнес-процесів. Діаграма станів що представляє етапи розробки графіки для комп'ютерної гри (рис. 2.1).

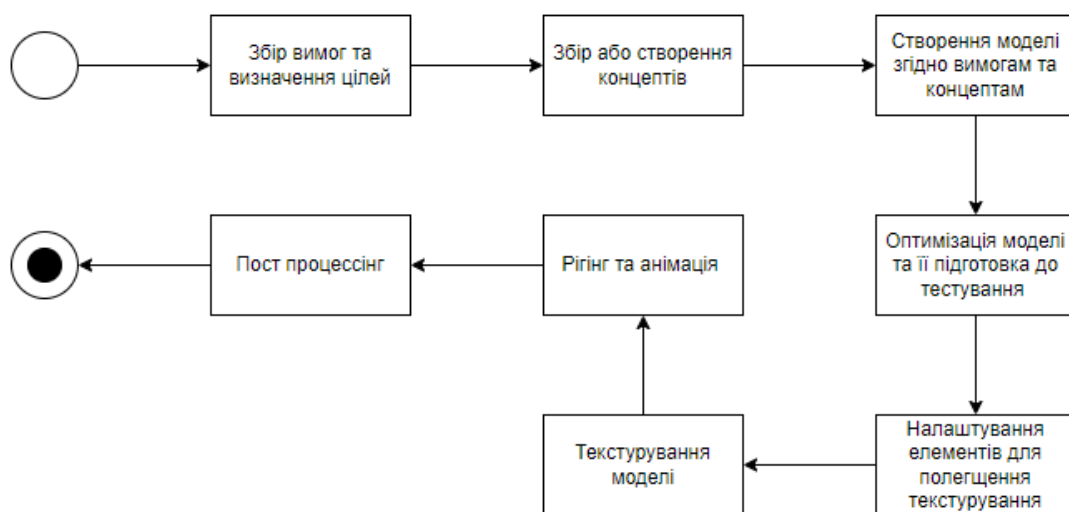


Рисунок 2.1 – Діаграма станів етапів розробки моделі

Для створення 3D-моделей в Blender потрібно чітко уявляти фінальний продукт та весь процес створення моделі покроково або оператися на вже існуючі концепти та ескізи, це допомагає проаналізувати весь пайплайн виробництва та передбачити можливі проблеми які можуть виникнути під час різних етапів роботи.

Після визначення стилю моделі та вибору концепта слідує один з початкових етапів розробки - створення основної форми. В Blender існують різноманітні заготовлені примітивні моделі наприклад куб та різні вузько спеціалізовані інструменти з допомогою яких можна простіше маніпулювати середовищем та полегшити і оптимізувати процес розробки 3D - моделі.

Після завершення розробки основної моделі слідує етап її перевірки на можливі помилки, графічні баги або так звані артефакти. Цей етап на мою думку є найважливішим тому, що від нього буде залежати подальша складність розробки та вигляд текстур на моделі, що на пряму впливає на стиль ігрового продукту.

Коли модель перевірена та розробник впевнений в її придатності до подальшої розробки починається етап текстуровання. Він включає в себе процес підготовки UV-розгортки та малювання або завантаження та експорт все готових текстур.

Після етапа текстуровання та фінальної перевірки моделі з текстурою готову модель з тектурами можна експортувати в середовище в якому будуть проводитися подальші маніпуляції (рис. 2.2).

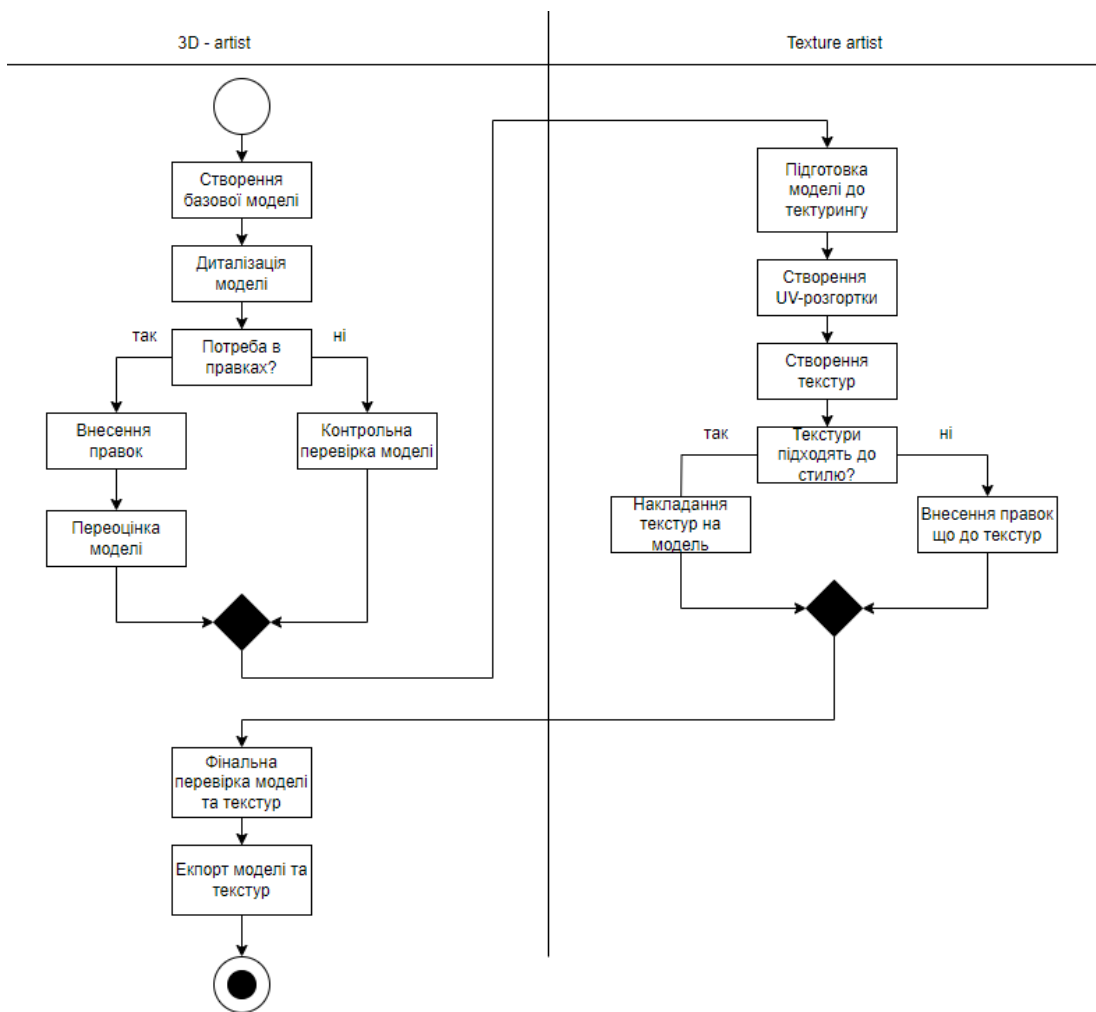


Рисунок 2.2 – Діаграма активності, що описує процес створення комп’ютерної 3D-моделі та текстур

Для покращення віальної складової моделі можна провести запікання текстур, це допоможе покращити вигляд моделі без додавання поліганажу.

Якщо це необхідно можна провести процес ретопології моделі. Ретопологія – це процес зменшення поліганажу на моделі шляхом накладання сітки поверх готової моделі що допомагає запобігти проблемам з оптимізацією.

Діаграми послідовностей що зображують поведінку ігрового додатку, демонструючи взаємодію між об'єктами протягом часу. Діаграма послідовності показує етапи процесу розробки готової 3D-моделі та текстур. Насправді цей процес може бути складнішим і вимагати додаткових кроків, залежно від типу ігрового додатку та використовуваних технологій (рис. 2.3).

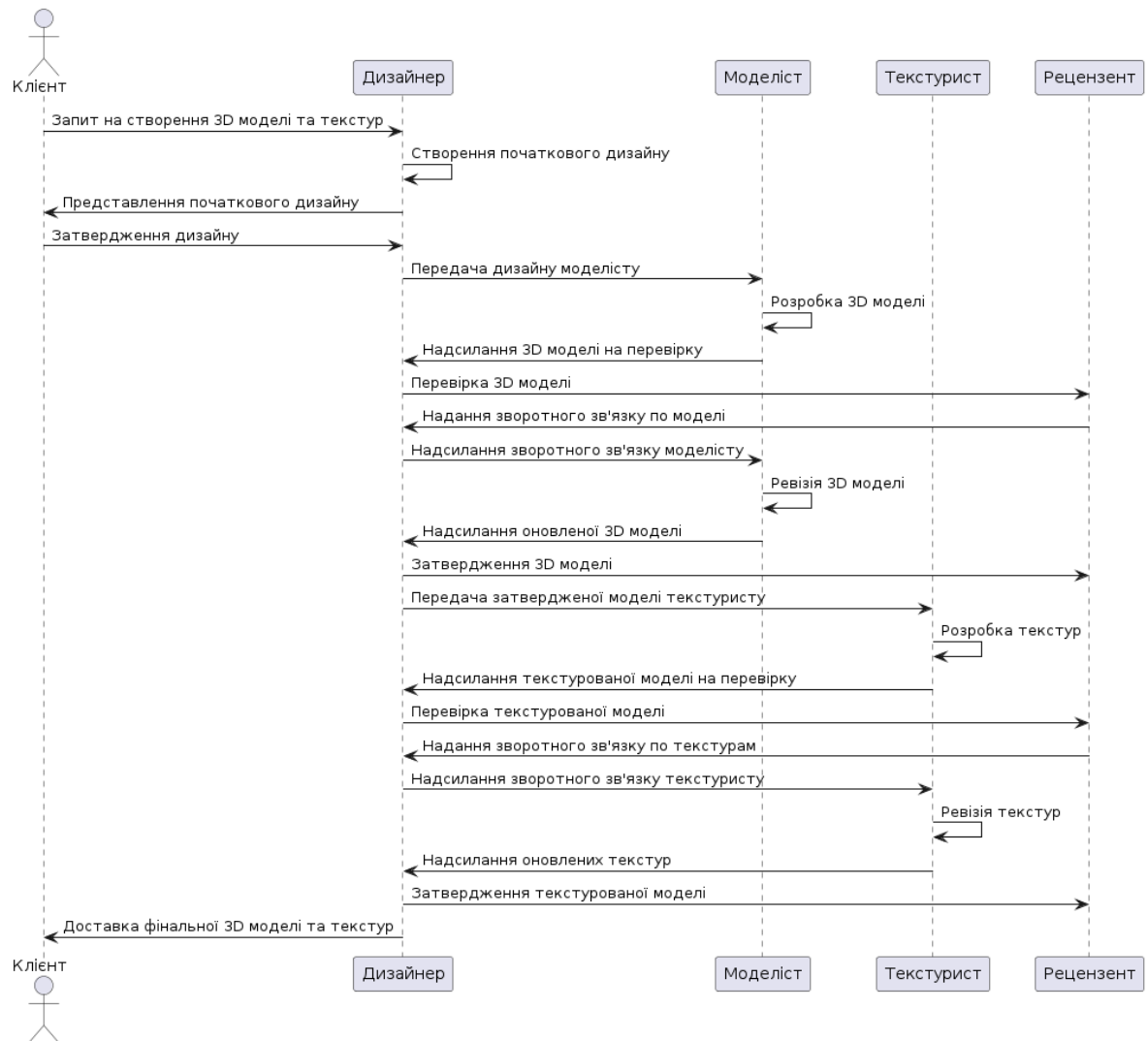


Рисунок 2.3 – Діаграма послідовності процесу розробки готової 3D-моделі та текстур

Діаграма варіантів використання (Use Case Diagram) - це тип діаграми в моделюванні програмного забезпечення, який використовується для візуалізації функціональних вимог до системи та взаємодії між її складовими частинами. У контексті процесу отримання та виконання роботи по створенню готового 3D-аскету, діаграма варіантів використання може відображати основні взаємодії між учасниками (акторами) та системою, які відбуваються під час цього процесу (рис. 2.4).

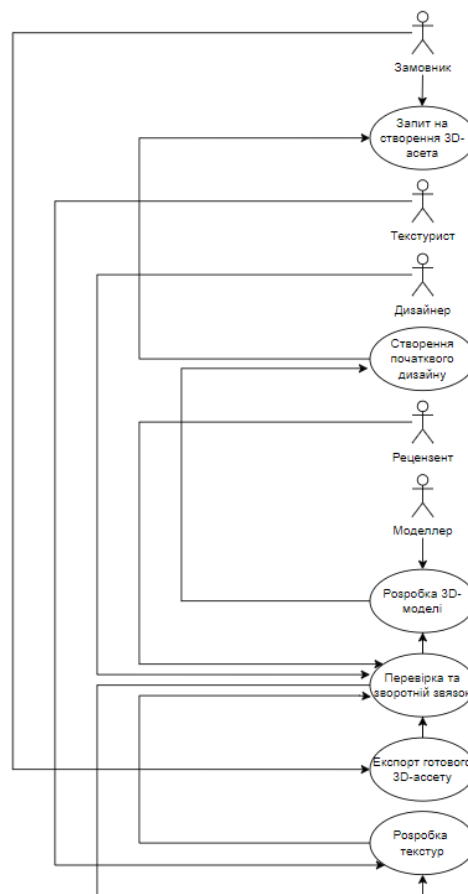


Рисунок 2.4 – Діаграма варіантів використання процесу отримання та виконання роботи по створенню готового 3D - асету

		<i>В.Д.Семенець</i>			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		<i>А.Р.Петросян</i>				27
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

2.2 Розробка 3D-моделі для ігрового додатку

Під час процедури розробки 3D-моделі потрібно тримати в голові багато важливих факторів таких як загальний графічний стиль ігрового продукту. Також потрібно чітко уявляти як розроблена модель буде виглядати в грі та функції які вона буде виконувати на ігровій мапі.

Для створення 3D-моделі можна використати один з багатьох існуючих спеціалізованих програмних пакетів, наприклад Blender який передбачає створення, розміщення та редагування примітивних форм таких як куб, циліндр або куля. З допомогою примітивних форм можна швидко та легко створити та редагувати початкову форму потрібної моделі, це сильно пришвидшує та спрощує початкові етапи розробки графічної складової ігрового додатку.

Також не слід забувати про вузько спеціалізовані інструменти які за замовчуванням встановлені в программі, вони значно спрощують створення моделей на всіх етапах циклу розробки.

Дуже важливо вчасно проаналізувати та врахувати технічні можливості ігрового рушія для полегшення подальшого опрацювання моделі.

Не треба забувати про те що розробка графічної складової та конкретної моделі напряму залежить від фізичного та морального стану моделлера, одією з найбільш імовірних ментальних проблем для робітника сфери 3D є емоційне вигорання.

Важливим є той факт що створення графічної складової для ігрового додатку є ітеративним процесом і слід бути готовим до неочікуваного внесення правок.

Для початку створення 3D-моделі [6] потрібно запустити програму Blender.

Після запуску ми потрапляємо в середовище розробки. Необхідно створити примітивну заготовку з якої ми будемо формувати основну форму графічного продукту.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		28

Щоб створити заготовку куб можна скористатися комбінацією кліш Shift + A. Це викличе невелику панель під назвою Add в якій можна обирати базові заготовки і не тільки (рис. 2.5).

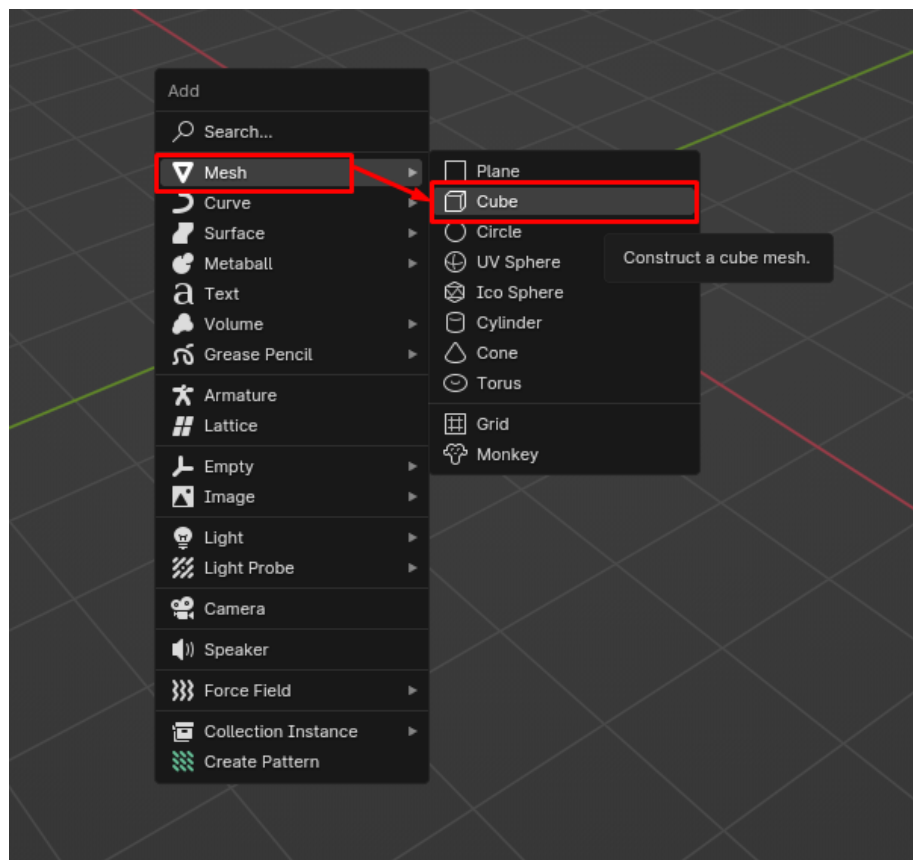


Рисунок 2.5 - Створення примітивного меша заготовки в програмі Blender

Подальша робота передбачає формування куба в певну фігуру або базову форму яка потрібна або яка будена найбільш зручною та універсальною – це називається базове моделювання (рис. 2.6), (рис В.1) по (рис В.17).

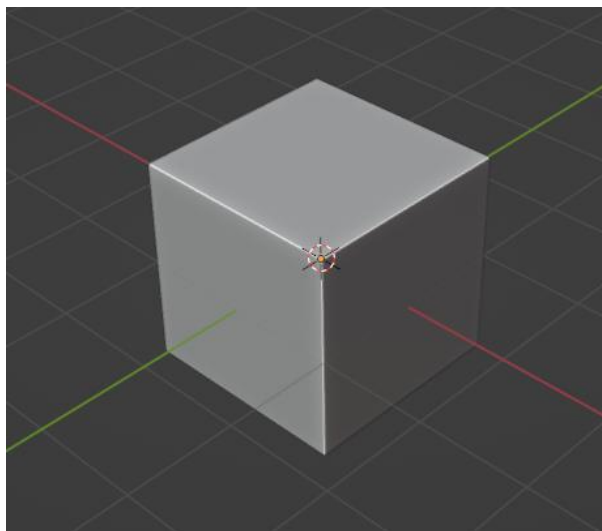


Рисунок 2.6 - Базова заготовка модель куба

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		29

Наприклад дошка для хаб локації на якій будуть вивішені відомості про завдання для гравця (рис. 2.7).

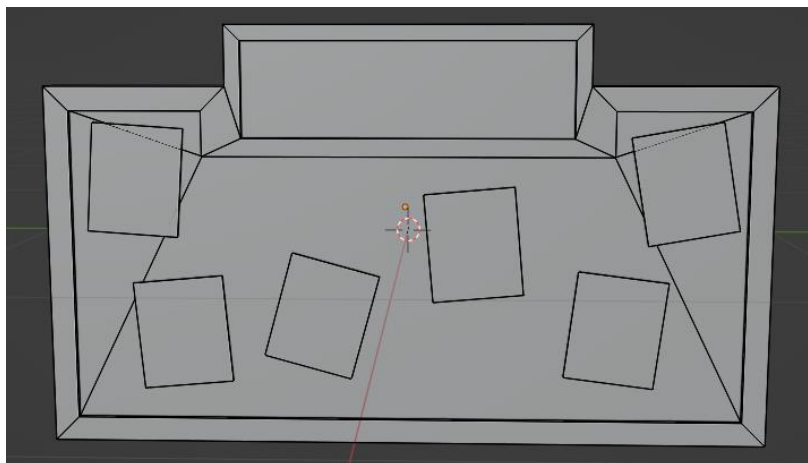


Рисунок 2.7 - Сформована базова модель дошки для хаб локації

Одночасно з цим буде корисно накласти декілька різнокольорових матеріалів аби не заплутатися в рінях глибини моделі та деталях які потім будуть нанесені на неї.

Необхідно перейти до вкладки матеріали на основній панелі з інструментами в программі Blender.

Після цього потрібно створити декілька новий матеріалів та накласти їх на модель таким чином щоб виділити основні перепади висот та деталей для подальшого, більш продуктивного створення UV-розгортки та полегшення інших маніпуляцій хоча потім ці матеріали доведеться видалити (рис. 2.8).

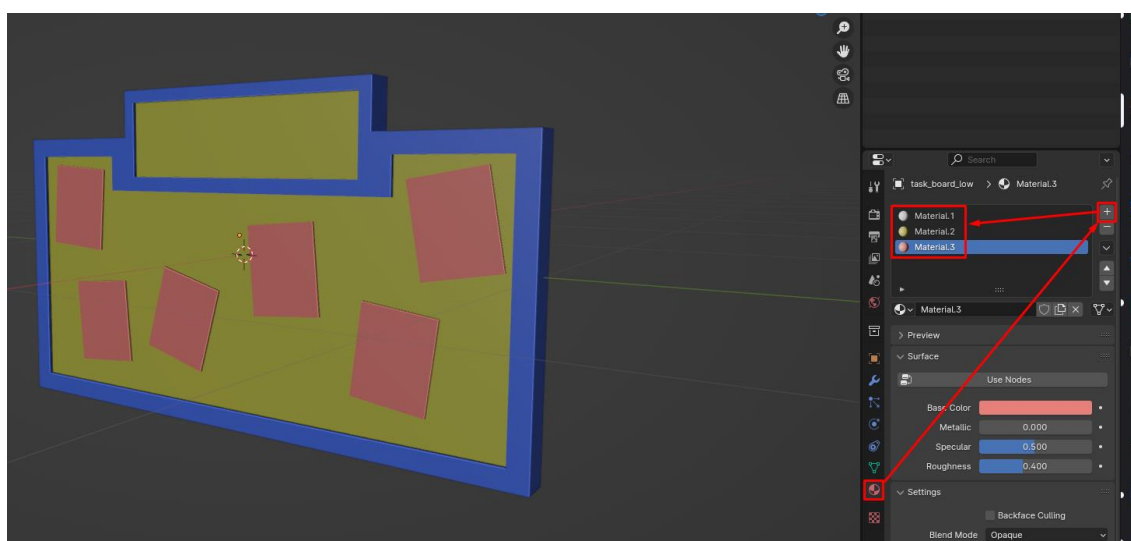


Рисунок 2.8 - Створення та накладання кольорового меша на базову модель в Blender

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		30

Наступним етапом створення 3D-моделі для ігрового додатку буде формування UV-розгортки для дошки з завданнями.

UV-розгортка (UV mapping) — це процес перетворення 3D-моделі на 2D-поверхню для подальшого нанесення текстур. Назва "UV" походить від координатної системи, де "U" та "V" представляють горизонтальні та вертикальні осі 2D-простору (на відміну від X, Y, Z у 3D-просторі).

Етапи процесу UV-розгортки:

На 3D-моделі визначаються шви, які будуть розрізами для створення розгортки. Вибір швів визначає, як модель буде розгортатися на плоску поверхню. Шви створюють там, де вони будуть найменш помітними або найзручнішими для розгортки.

Після створення швів 3D-модель "розрізається" і "розгортається" на плоску 2D-поверхню. Цей етап перетворює поверхню моделі в двовимірний макет, який можна текстурувати (рис. 2.9).

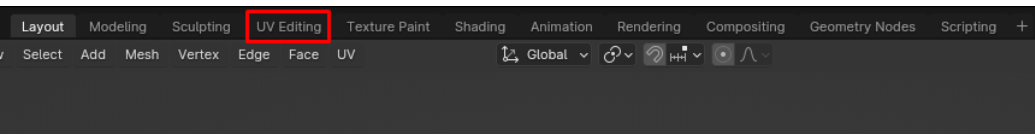


Рисунок 2.9 - Кнопка переходу до режиму UV-розгортки

Після розгортки UV-мапа може бути відкоригована для мінімізації розтягувань та спотворень. Вершини UV координат можуть бути переміщені, щоб забезпечити більш рівномірний розподіл текстури по поверхні моделі (рис. 2.9).

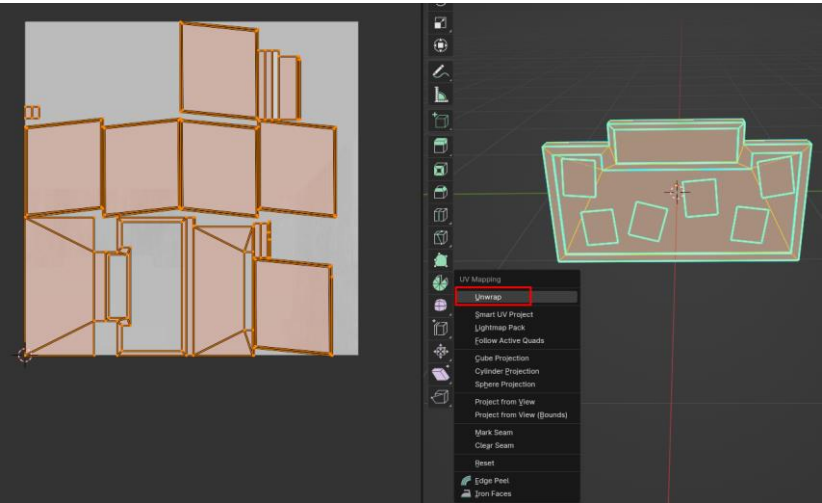


Рисунок 2.9 - Приклад UV-розгортки для базового меша дошки в Blender

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		31

Також для покращення UV-розгортки можна використати аддон UVPackmaster3 (рис 2.10).

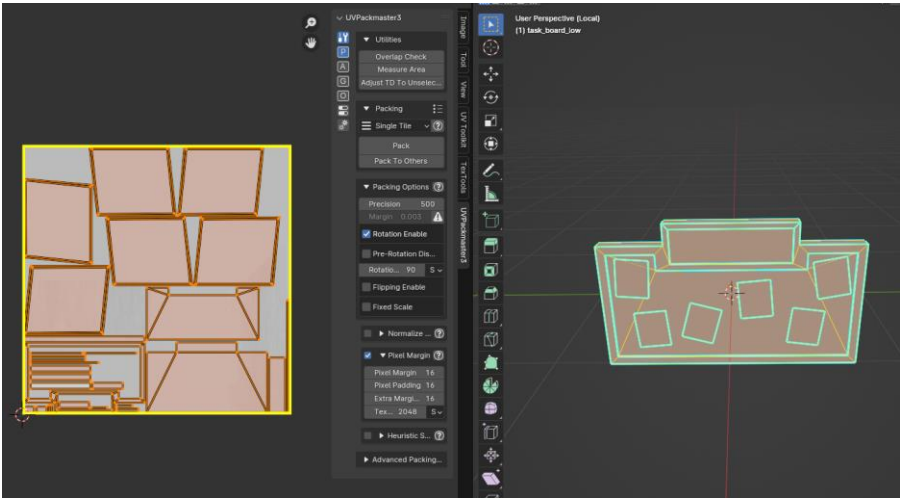


Рисунок 2.10 - Покращена UV-розгортка

Чому UV-розгортка важлива? UV-розгортка дозволяє точніше накладати текстури на 3D-моделі. Текстури — це зображення, які додають моделі кольори, деталі поверхні, такі як нерівності, реалістичні матеріали і багато іншого. Без правильної UV-розгортки текстури можуть виглядати спотвореними або неправильно накладеними на модель. UV-розгортка — це важливий крок у процесі створення якісних 3D-моделей, особливо для використання в іграх, анімаціях та візуалізаціях.

Процес створення High Poly моделі перед запіканням текстур:

Створення High Poly моделі є важливим етапом у 3D моделюванні, особливо для запікання текстур і нормалей на Low Poly моделі. High Poly моделі відрізняються високим рівнем деталізації, що дозволяє створювати реалістичні та детальні текстури для використання в іграх, фільмах та інших медіа.

Перед накладанням модифікаторів які потрібні для створення High Poly необхідно дублювати основну модель, щоб з її допомогою пізніше провести запікання текстур (рис 2.11).

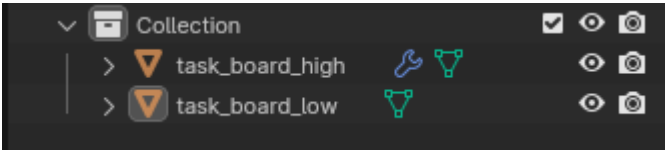


Рисунок 2.11 - Було створено 2 моделі high poly та low poly

Поділ поверхонь: Додавання додаткових ліній поділу (edge loops) для підвищення рівня деталізації (рис 2.12).

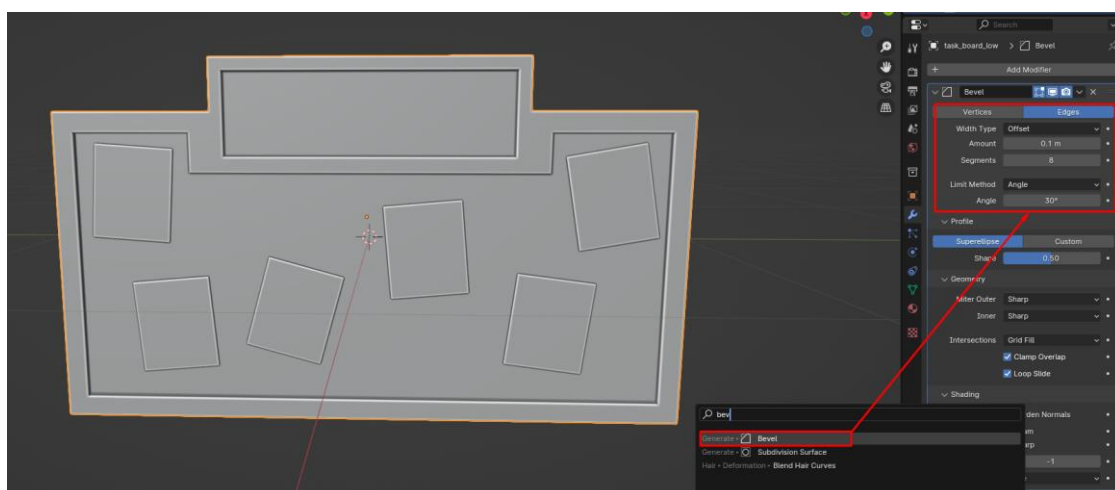


Рисунок 2.12 - Перший модифікатор для створення high poly моделі

Додавання дрібних деталей: Використання інструментів скульптингу, таких як ZBrush або Blender, для додавання дрібних деталей, таких як зморшки, тріщини, текстури шкіри тощо.

Модифікатори: Використання модифікаторів, таких як Subdivision Surface у Blender [10], для згладжування поверхонь і додавання більшої кількості полігонів (рис. 2.13).

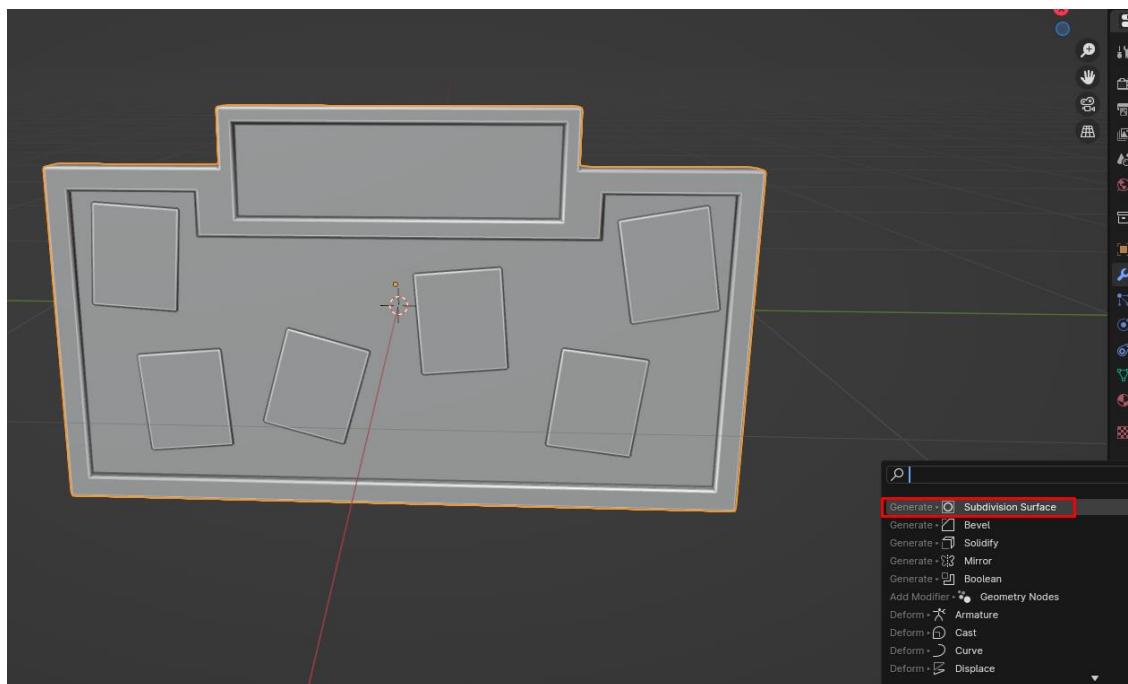


Рисунок 2.13 - Другий модифікатор для створення high poly моделі

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Давайте розглянемо один із основних інструментів у Blender — Subdivision Surface Modifier. Цей інструмент використовується для згладжування та підвищення деталізації моделей, розбиваючи поверхню на менші полігони.

Математична формула Subdivision Surface Modifier:

Subdivision Surface Modifier (simple Subdivision). Цей алгоритм перетворює полігональні моделі в більш гладкі поверхні шляхом багаторазового поділу поверхонь на менші квадрати. Основні кроки цього алгоритму такі:

Нова вершина на ребрі створюється як середнє арифметичне координат обох кінцевих вершин ребра та середнє арифметичне координат центрів обох фейсов, що містять це ребро.

Формула для нової вершини на ребрі

$$E_{\text{new}} = 1/4 (V_1 + V_2 + F_1 + F_2),$$

E_{new} — перша вершина ребра.

V_1 — перша вершина ребра.

V_2 — друга вершина ребра.

F_1 — центр першої грані, що примикає до ребра.

F_2 — центр другої грані, що примикає до ребра.

Створення нових вершин для кожного фейсу:

Нова вершина на фейсі створюється як середнє арифметичне координат всіх вершин, що утворюють фейс.

Формула для нової вершини на фейсі:

$$F_{\text{new}} = 1/n \sum_{i=1}^n V_i$$

Де - F_{new} — нова вершина на грані (центроїд грані).

n — кількість вершин, що утворюють грань.

Σ — сума.

i — індекс, що пробігає від 1 до n

V_i — вершина i – тої грані

Зміщення старих вершин:

Старі вершини зміщуються на нові позиції, які враховують усі нові вершини, що утворені навколо кожної старої вершини.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Формула для зміщення старої вершини:

$$V_{\text{new}} = 1/n (F_{\text{sum}} + 2E_{\text{sum}} + (n - 3)V)$$

V_{new} — нове положення старої вершини після зміщення.

n — кількість сусідніх вершин, які примикають до старої вершини.

F_{sum} — сума центрів граней, що примикають до старої вершини.

E_{sum} — сума середніх точок ребер, що примикають до старої вершини.

V — старе положення вершини.

Візуалізація алгоритму:

Початкова модель: Нехай у є куб.

Перша ітерація: Ребра куба діляться на менші сегменти, створюються нові вершини, центри фейсов зміщуються.

Друга ітерація: Повторення алгоритму для створення більш гладкої форми.

Наступні ітерації: Кожна наступна ітерація робить модель ще більш гладкою, наближаючи її до сфери (рис. 2.14).

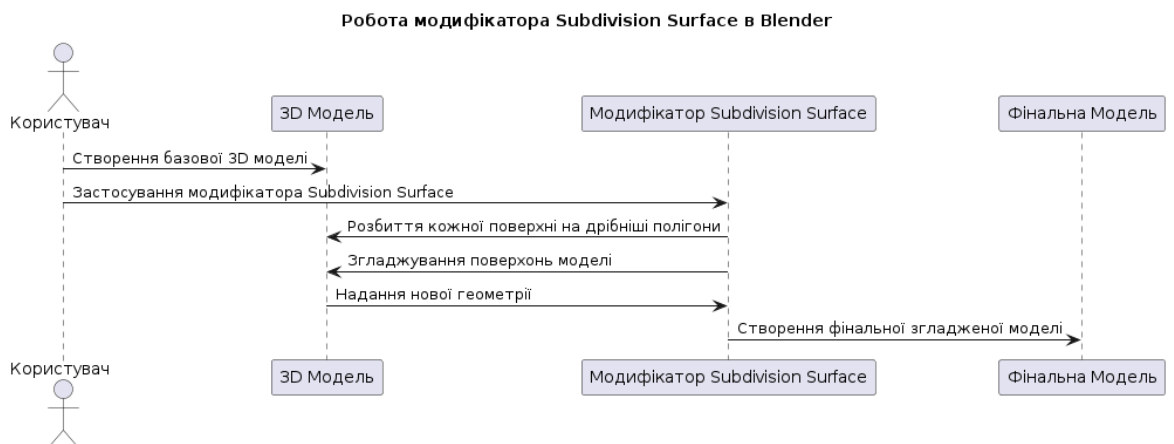


Рисунок 2.14 - Робота модифікатора Subdivision Surface в Blender

Після підготовки основної форми моделі, створення та налаштування оптимальної UV-розгортки а також створення додаткової high poli моделі для подальшого запікання текстур прийшов час експортувати готові моделі в середовище для текстуровання.

Експорт моделей з Blender в інше середовище є важливим етапом у 3D-роботі, оскільки дозволяє використовувати створені моделі в різних програмах та ігрових движках. Процес експорту включає кілька кроків, щоб забезпечити

сумісність і оптимальну якість моделі. Ось покроковий опис процесу експорту моделей з Blender:

Налаштування трансформацій:

Спочатку необхідно скинути масштаб моделі до значен 1,1,1, від значень масштабу залежить диталізація текстур на моделі та її реальний розмір при експорті в ігрове середовище (рис. 2.15).

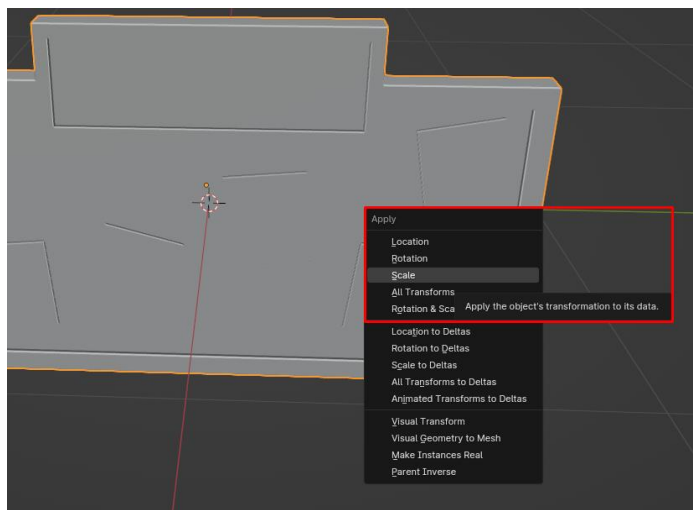


Рисунок 2.15 - Вікно налаштування трансформації та масштабу

Обнулення трансформацій: Переконайтесь, що масштабування, обертання та позиції об'єктів обнулені (Ctrl + A -> Scale, Rotation, Location) [11]. Це забезпечить коректне позиціонування моделі в іншому середовищі (рис. 2.16).

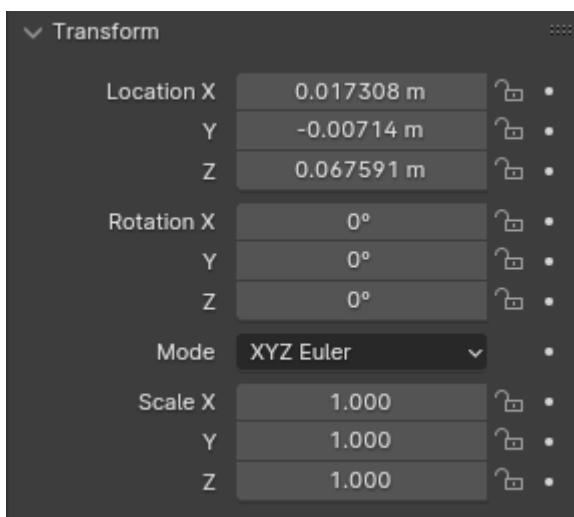


Рисунок 2.16 - Налаштування масштабу моделі в Blender

Blender підтримує різні формати експорту, серед яких найпоширеніші:

FBX: Популярний формат для імпорту в ігрові движки (Unity, Unreal Engine) та інші 3D-програми.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		36

OBJ: Широко використовується для передачі моделей між різними програмами, але має обмежену підтримку анімацій та матеріалів.

GLTF/GLB: Сучасний формат для веб-додатків та AR/VR з підтримкою матеріалів PBR.

Використовуємо формат FBX тому, що він найбільше підходить під вимоги подальшого середовища в якому буде знаходитися модель.

Експорт моделі в форматі FBX:

Вибір об'єкта: потрібно вибрати модель або групу об'єктів для експорту.

Файл -> Експорт -> FBX (.fbx) [11]: потрібно підкрити меню експорту та вибрати формат FBX (рис. 2.17).

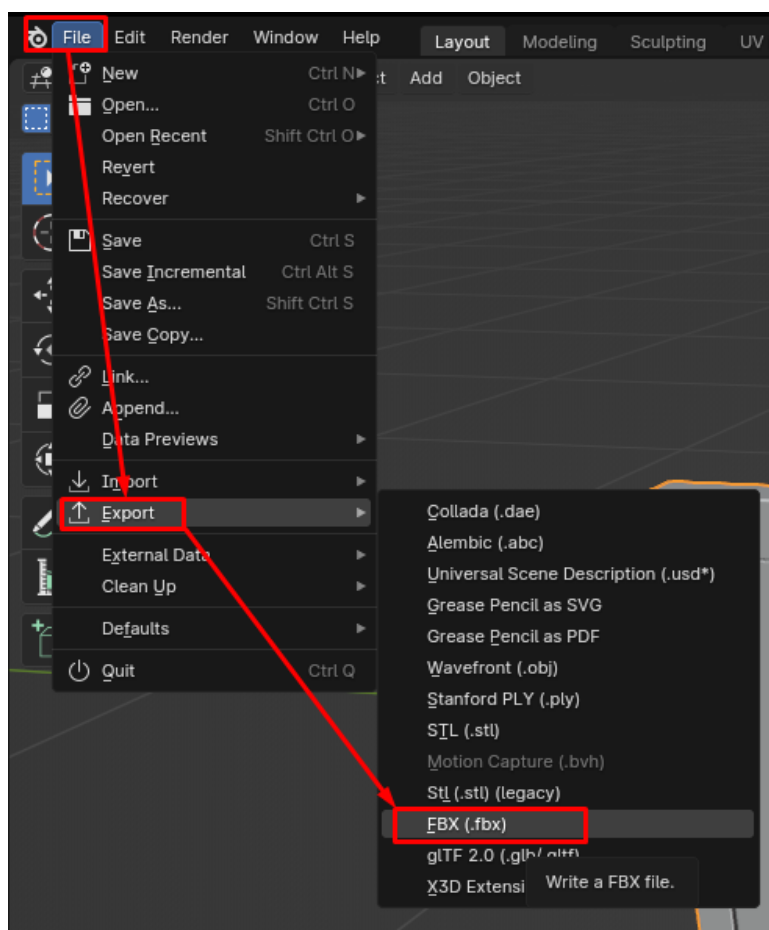


Рисунок 2.17 - Вікна експорту моделі

Налаштування експорту:

Scale: Встановіть масштаб (часто 1.00).

Include: потрібно вибрати, що експортувати (наприклад, Selected Objects).

Transform: Переконайтесь, що вибрані правильні опції для трансформацій (Apply Transform).

Geometry: Smoothing для згладжування нормалей.

Armature: Якщо модель має анімацію, потрібно вибрати опції для кістяка (Add Leaf Bones зазвичай вимикається).

Експорт: Натиснути Export FBX (рис. 2.18).

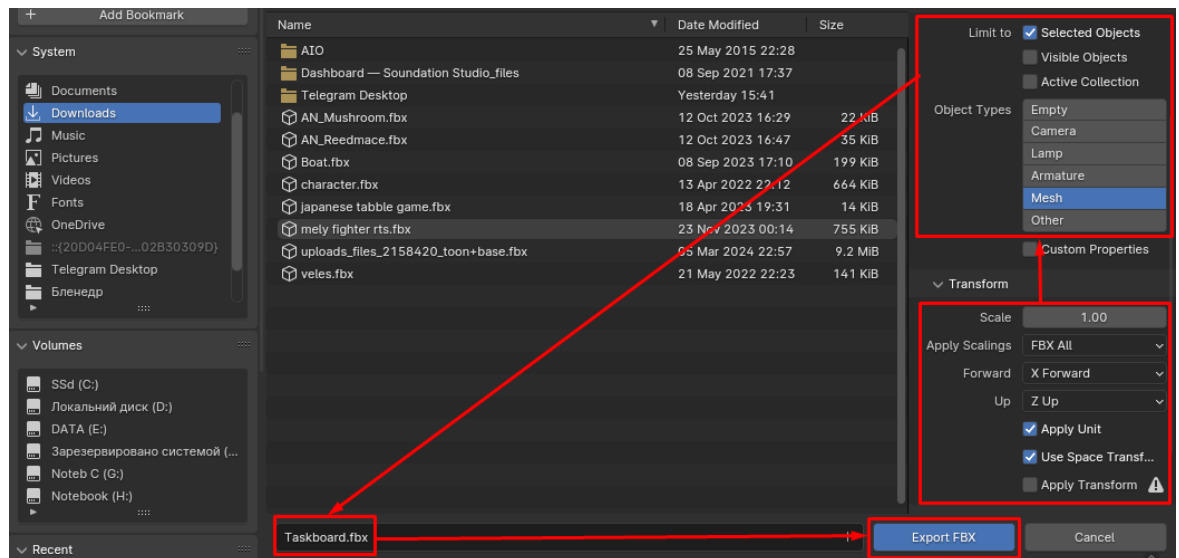


Рисунок 2.18 - Приклад експортування моделі з Blender а форматі FBX

2.3 Створення графічного матеріалу для ігрового додатку

Наступними етапами розробки 3D-моделі є імпорт моделей в потрібне середовище, запікання текстур їх накладання на модель та їх експорт.

Імпорт моделі в Substance Painter:

Створення нового проекту:

File -> New: Новий проект (рис. 2.19).

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				38
Змн.	Арк.	№ докум.	Підпис	Дата		

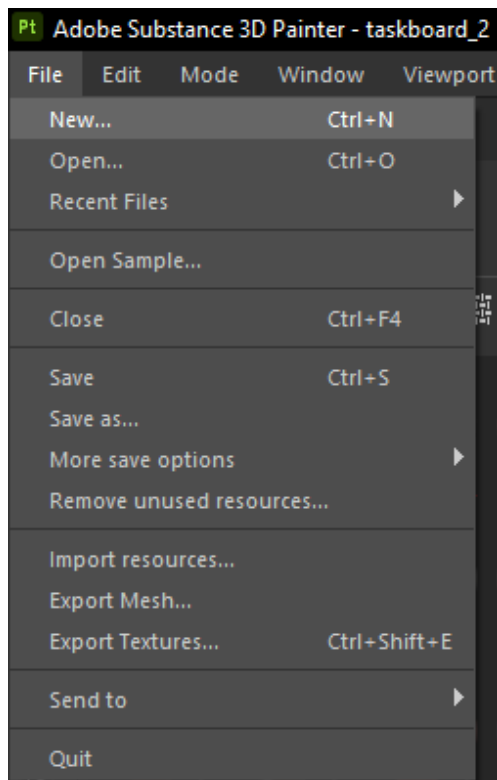


Рисунок 2.19 - Меню створення нового файлу в середовищі.

Select: потрібно вибрати імпортований файл FBX.

Template: треба вибрати шаблон для проекту, залежно від типу текстур і шейдерів (PBR - Metallic Roughness або Specular Glossiness) [12].

Налаштування імпорту:

Document resolution: треба вибрати роздільну здатність документа (наприклад, 2048 або 4096).

Normal Map Format: Вибрати формат карти нормалей (DirectX або OpenGL) (рис. 2.20).

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		39

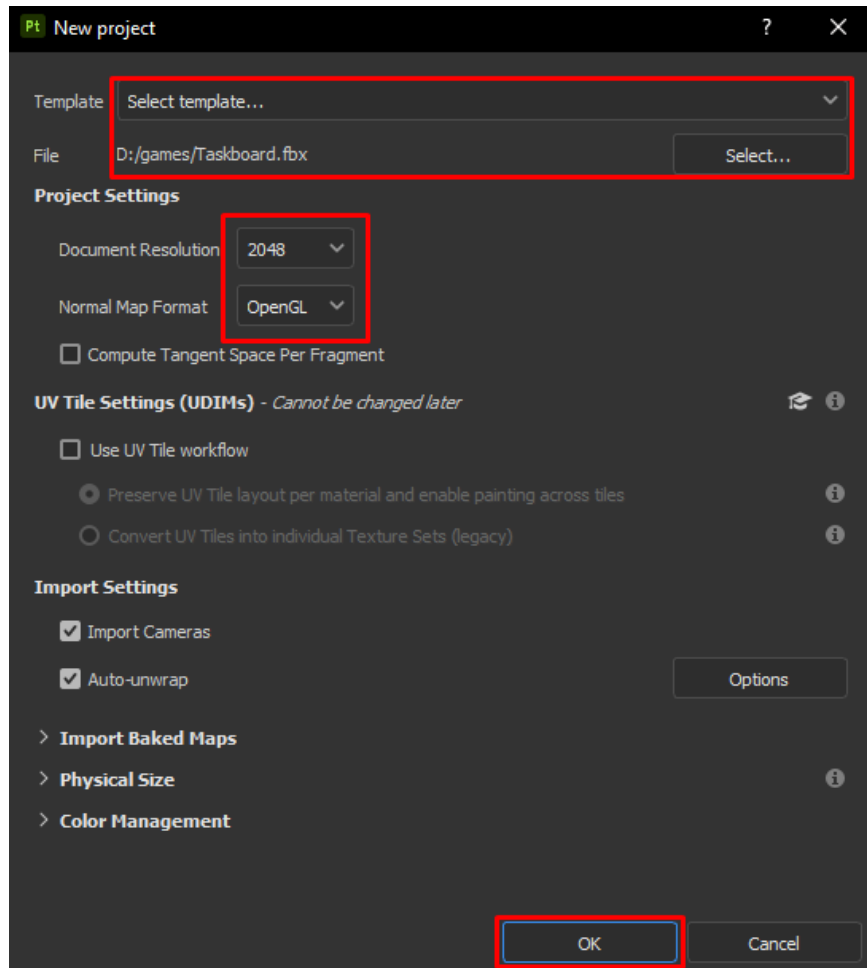


Рисунок 2.20 - Імпорт та налаштування моделі в середовищі

Слід сказати що інтерфейс програми Substance Painter достатньо простий але потребує певного часу на звикання або налаштування (рис. 2.21).

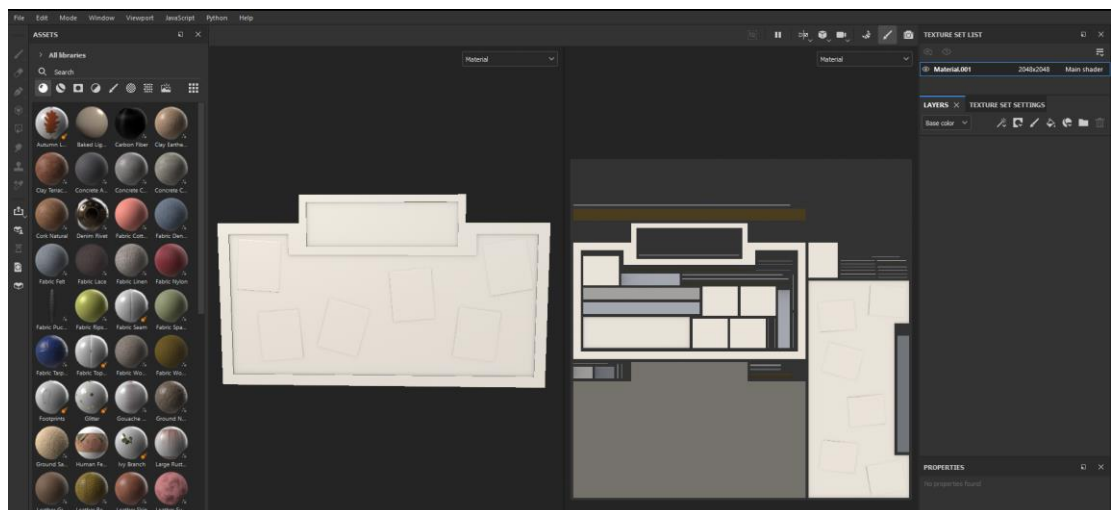


Рисунок 2.21 - Вікно з інтерфейсом програми

Bake Mesh Maps: Натиснути кнопку Bake Mesh Maps, щоб запекти карти з Low Poly моделі (рис. 2.22).

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				40
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 2.22 - Bake Mesh Maps – перехід в режим запікання текстур
Вибір карти, які потрібно запекти (Normal, AO, Curvature, etc.).
Встановлення роздільної здатності для карт (рис. 2.23) та (рис. 2.24).

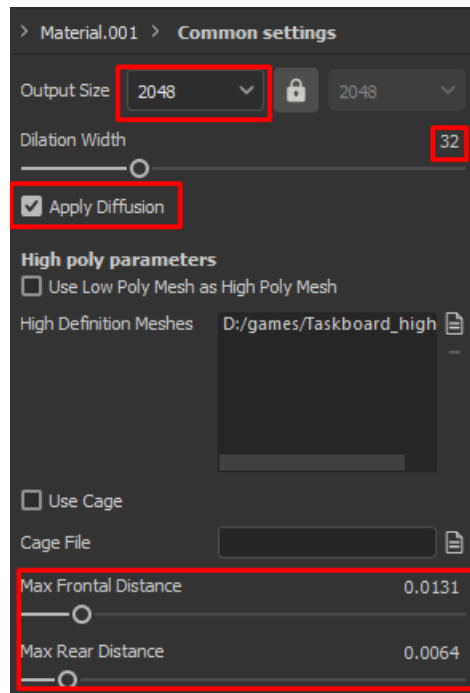


Рисунок 2.23 - Налаштування роздільної здатності для карт

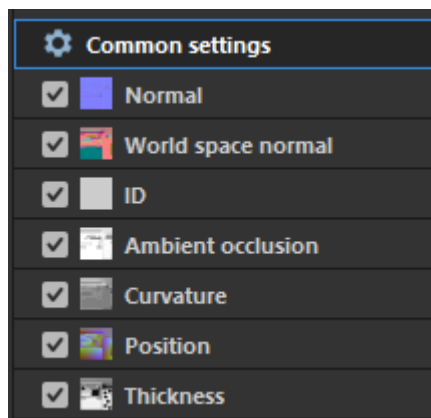


Рисунок 2.24 - Вибір розширення та запікання карт в Substance Painter
Текстурування в Substance Painter:
Base Layer: Створіть базовий шар з основним кольором та матеріалом.
Details Layer: Додавання шарів для деталей, як потертості та інші ефекти.
Використання Smart Materials і Smart Masks (рис. 2.25).

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				41
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 2.25 - Вибір кольору для однієї з текстур

Smart Materials: Використовуйте попередньо налаштовані матеріали, щоб швидко додати складні текстури.

Smart Masks: Використовуйте маски для створення реалістичних ефектів зношеності та інших деталей (рис. 2.26), (рис В.15) по (рис В.17).



Рисунок 2.26 - Модель з накладеними Smart Materials і Smart Masks в Substance Painter

Запікання та експорт текстур

Запікання текстур: Після завершення текстурування запечіть всі карти для отримання фінальних текстур.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				42
Змн.	Арк.	№ докум.	Підпис	Дата		

Експорт текстур: Використовуйте File -> Export Textures для експорту всіх необхідних текстур (рис. 2.27).

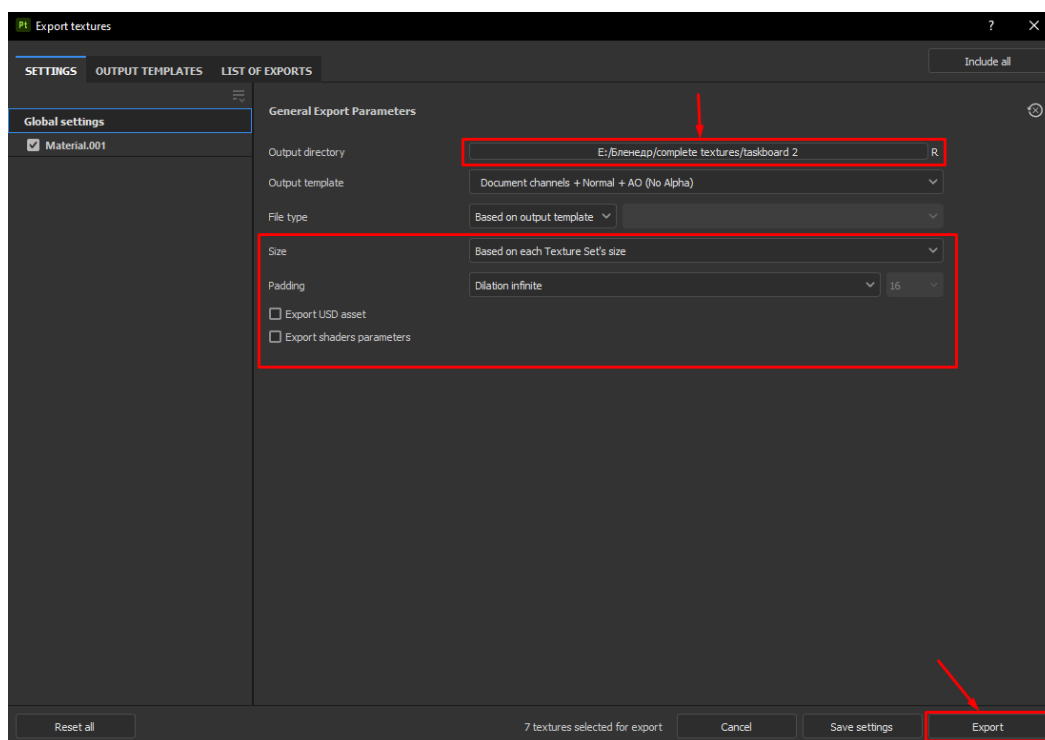


Рисунок 2.27 - Процес експорту текстур з Substance Painter

Шейдінг у Blender з використанням нодів — це потужний спосіб створення матеріалів і текстур для 3D моделей. Ноди дозволяють створювати складні матеріали за допомогою вузлів, які представляють різні функції, такі як змішування кольорів, застосування текстур, створення відображень і багато іншого.

Shader Editor: Спеціальне вікно, для роботи з нодами та створення матеріалів (рис. 2.28).

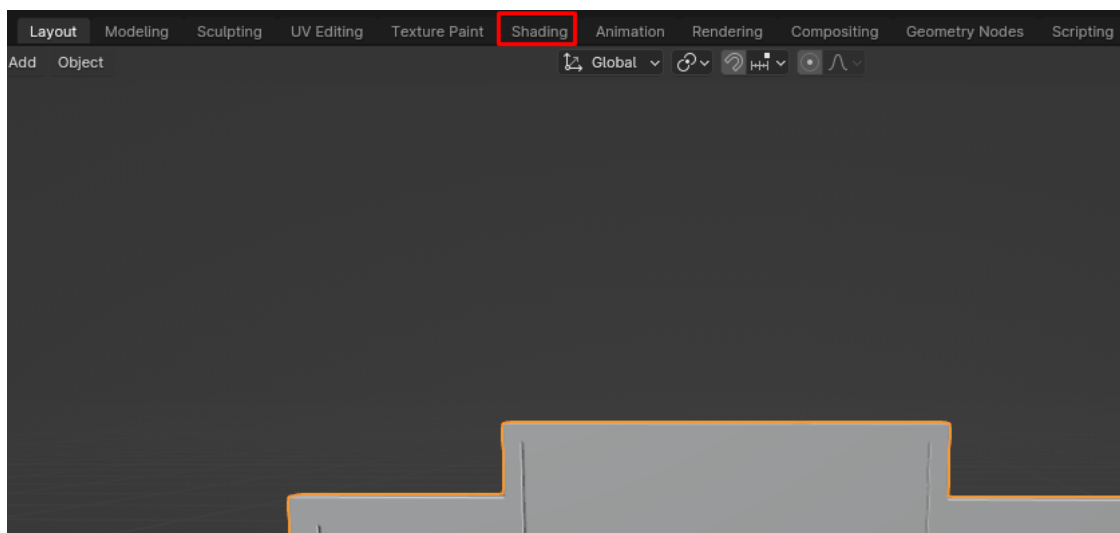


Рисунок 2.28 - Кнопка переходу в меню шейдерів

Основні поняття шейдінгу в Blender з нодами:

Матеріал: Набір властивостей, які визначають, як поверхня моделі взаємодіє зі світлом.

Ноди (Nodes)[4]: Блоки, які виконують різні операції з кольором, текстурою і світлом. Вони з'єднуються між собою для створення комплексних матеріалів.

Ноди Shader: Використовуються для визначення кінцевого вигляду матеріалу (наприклад, Diffuse, Glossy, Mix Shader).

Ноди Input: Вхідні дані для шейдерів, такі як кольори, текстури, координати UV та інше.

Основні типи нодів:

Shader Nodes: Визначають властивості поверхні, такі як:

Diffuse BSDF: Основний шейдер для непрозорих матеріалів.

Glossy BSDF: Шейдер для відображень та блиску.

Mix Shader: Змішує два різні шейдери.

Principled BSDF: Комплексний шейдер, що об'єднує багато властивостей в одному вузлі.

Texture Nodes: Відповідають за додавання текстур до матеріалів.

Image Texture: Використовує зовнішнє зображення як текстуру.

Noise Texture: Генерує шумову текстуру.

Voronoi Texture: Генерує текстуру на основі діаграми Вороного.

Input Nodes: Вхідні дані для матеріалів.

Texture Coordinate: Надає різні способи координат текстур (UV, Object, Generated).

Fresnel: Визначає ефект відображення залежно від кута зору.

Color Nodes: Виконують різні операції з кольорами.

RGB: Простий колірний вузол.

Mix RGB: Змішує два кольори.

Invert: Інвертує кольори.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				44
Змн.	Арк.	№ докум.	Підпис	Дата		

Vector Nodes: Маніпулюють векторними даними.

Bump: Додає ефект рельєфу.

Normal Map: Використовує карту нормалей для додавання деталей на поверхню.

Створення нового матеріалу:

Потрібно вибрати об'єкт.

Потрібно перейти у вкладку Material в Properties панелі.

Натиснути New для створення нового матеріалу.

Відкриття Shader Editor:

Необхідно перейти у вікно Shader Editor.

У правій частині екрана ви побачите новий матеріал з одним нодом Principled BSDF, підключеним до Material Output.

Додавання текстури:

Додавання нод Image Texture (Shift + A -> Texture -> Image Texture)[19].

Підключити Color вихід Image Texture до Base Color входу Principled BSDF.

Завантажте зображення текстури (рис. 2.29).

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		45

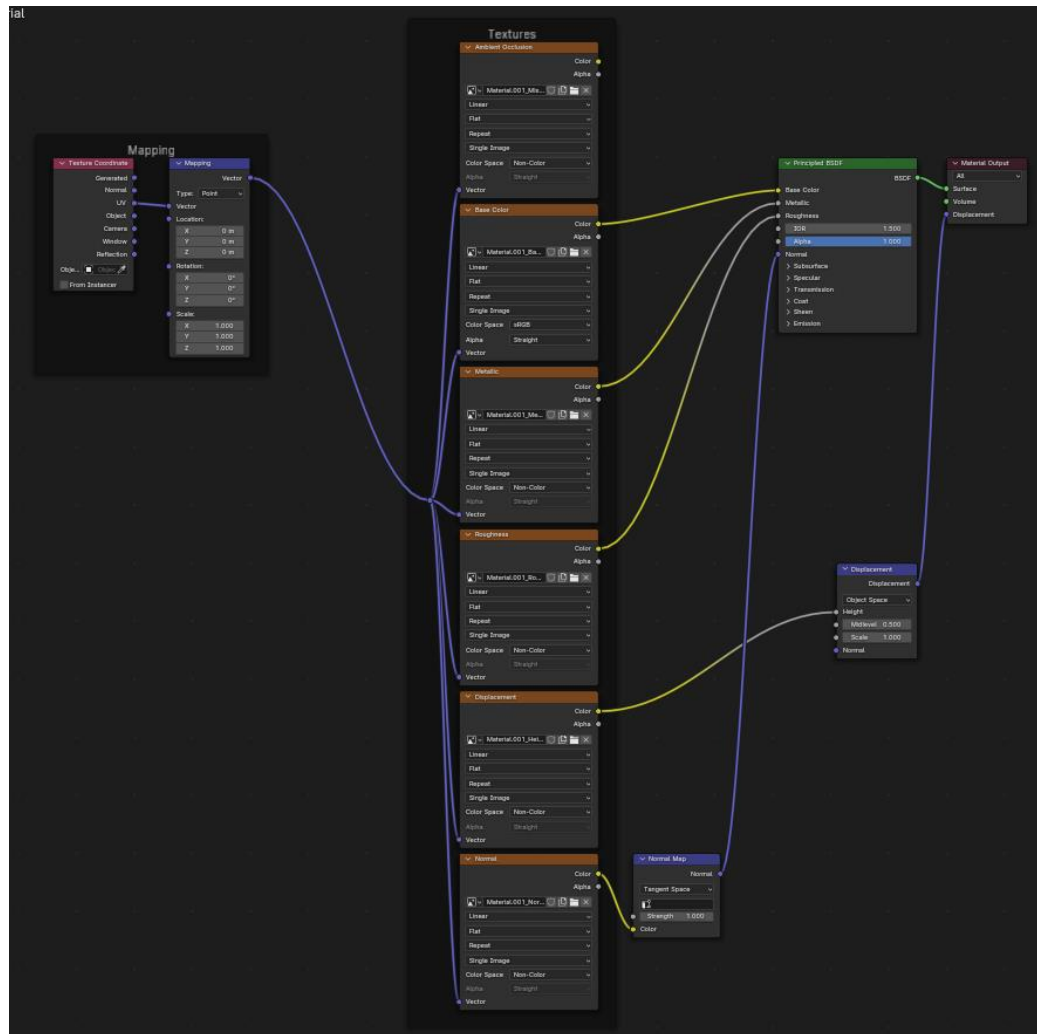


Рисунок 2.29 - Ноди для накладання текстур в Blender

Основні типи шейдерів які використовуються в Blender:

Diffuse BSDF (Розсіюючий шейдер):

Функція: Відповідає за основний колір об'єкта. Він визначає, як поверхня розсіює світло в усіх напрямках.

Використання: Основний шейдер для непрозорих матеріалів.

Glossy BSDF (Блискучий шейдер):

Функція: Відповідає за створення відблисків і відображень на поверхні об'єкта.

Використання: Використовується для матеріалів з відбиваючими поверхнями, таких як метали і поліровані поверхні.

Principled BSDF (Універсальний шейдер):

Функція: Об'єднує багато властивостей в одному вузлі, включаючи металевість, шорсткість, прозорість та інші.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		46

Використання: Універсальний шейдер для створення реалістичних матеріалів. Спрощує процес шейдінгу, об'єднуючи багато налаштувань в одному вузлі.

Glass BSDF (Шейдер скла):

Функція: Відповідає за створення прозорих матеріалів, які заломлюють світло, як скло.

Використання: Використовується для матеріалів, таких як скло, вода і інші прозорі об'єкти.

Emission (Шейдер випромінювання)[8]:

Функція: Випромінює світло з поверхні об'єкта.

Використання: Використовується для створення світлових джерел або самосвітних об'єктів.

Mix Shader (Змішувальний шейдер):

Функція: Змішує два різні шейдери на основі значення фактору змішування (Fac).

Використання: Використовується для комбінування властивостей двох матеріалів.

Transparent BSDF (Прозорий шейдер):

Функція: Створює ідеально прозорий матеріал.

Використання: Використовується для прозорих матеріалів без заломлення світла, таких як прозора плівка.

Anisotropic BSDF (Анізотропний шейдер):

Функція: Відповідає за створення анізотропних відблисків, які виникають на поверхнях з напрямною текстурою, таких як шліфований метал.

Використання: Використовується для матеріалів з анізотропними властивостями.

Шейдінг у Blender є потужним і гнучким інструментом для створення реалістичних і фантастичних матеріалів для 3D моделей. Використання нодів дозволяє точно налаштовувати властивості матеріалів і досягати високої деталізації.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		47

2.4 Реалізація програмного коду для оптимізації геометрії

Для пришвидшення та оптимізації процесу розробки 3D - моделей був застосований користувацький аддон для автоматичного зменшення полігонажу певної області або моделі в цілому (рис. 2.30).

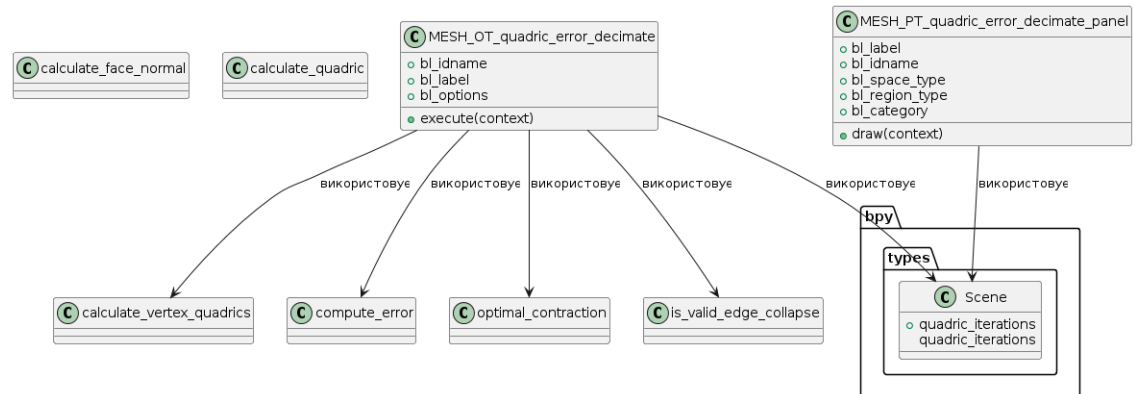


Рисунок 2.30 - Діаграма класів аддону для зменшення полігонажу

Частина коду аддону.

```
def calculate_face_normal(v1, v2, v3):
    normal = (v2 - v1).cross(v3 - v1)
    normal.normalize()
    return normal

def calculate_quadric(v1, v2, v3):
    normal = calculate_face_normal(v1, v2, v3)
    d = -normal.dot(v1)
    q = np.zeros((4, 4))
    normal_homogeneous = np.append(normal, d)
    for i in range(4):
        for j in range(4):
            q[i][j] = normal_homogeneous[i] * normal_homogeneous[j]
    return q

def calculate_vertex_quadrics(obj):
    bm = bmesh.new()
    bm.from_mesh(obj.data)
    vertex_quadrics = {v.index: np.zeros((4, 4)) for v in bm.verts}

    for face in bm.faces:
        v1, v2, v3 = face.verts[0], face.verts[1], face.verts[2]
        q = calculate_quadric(v1.co, v2.co, v3.co)
        for v in face.verts:
            vertex_quadrics[v.index] += q

    bm.free()
    return vertex_quadrics
```

Цей код обчислює квадрики помилок для кожної вершини об'єкта в 3D сцені. Квадрики помилок використовуються в алгоритмах спрощення мешів, таких як квадричне спрощення.

Призначення: Обчислює нормаль до грані, визначеної трьома вершинами $v1$, $v2$ і $v3$.

Вектор нормалі обчислюється як векторний добуток (cross product) між двома векторами, які утворюються з трьох точок.

Нормаль нормалізується до одиничної довжини.

Повертається нормалізований вектор нормалі.

Призначення: Обчислює квадрат для грані, визначеної трьома вершинами $v1$, $v2$ і $v3$.

Опис:

Викликає `calculate_face_normal` для отримання нормалі до грані.

Обчислює відстань d від початку координат до площини, визначеної нормаллю та однією з вершин.

Створює 4x4 матрицю q , заповнену нулями.

Розширює нормаль до однорідної координати, додаючи d .

Заповнює матрицю q як добуток зовнішнього добутку (outer product) однорідної нормалі з самою собою.

Повертає обчислений квадратик.

Призначення: Обчислює квадратики для всіх вершин заданого об'єкта.

Опис:

Створює новий об'єкт `bmesh`.

Ініціалізує `bmesh` з даних об'єкта `obj`.

Створює словник `vertex_quadrics`, де ключами є індекси вершин, а значеннями - нульові 4x4 матриці.

Для кожної грані об'єкта:

Визначає перші три вершини грані ($v1$, $v2$, $v3$).

Обчислює квадратик для грані за допомогою `calculate_quadric`.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		49

Додає обчислений квадрик до відповідних вершин у словнику vertex_quadrics.

Звільняє ресурси bmesh.

Повертає словник vertex_quadrics.

Узагальнений опис

Нормалі: Вектори, перпендикулярні до поверхні грані, що вказують напрямком зовні.

Квадрики: 4x4 матриці, які зберігають інформацію про площини, що проходять через вершини меша.

Процес:

Обчислення нормалі до кожної грані.

Використання нормалі для створення квадрика, який враховує геометричні властивості грані.

Агрегування квадриків для всіх граней, що прикріплені до кожної вершини.

Цей код є частиною алгоритму спрощення меша, який використовує квадрики помилок для визначення важливості кожної вершини і вибору тих, які можна видалити, щоб зберегти загальну форму меша.

Чим корисні користувацькі аддони в Blender:

Користувацькі аддони в Blender мають багато переваг і корисних функцій, які розширюють можливості програми і роблять робочий процес більш ефективним та зручним. Ось декілька основних переваг користувацьких аддонів у Blender:

Розширення функціональності:

Аддони дозволяють додати нові інструменти та функції, які відсутні в стандартній версії Blender. Наприклад, специфічні інструменти моделювання, рендерингу, анімації або скульптурингу.

Автоматизація завдань:

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		50

Багато аддонів автоматизують рутинні та повторювані завдання, що зменшує час, необхідний для виконання таких завдань, і мінімізує помилки, пов'язані з людським фактором.

Підвищення продуктивності:

Аддони можуть оптимізувати різні аспекти робочого процесу, що дозволяє працювати швидше і зручніше. Наприклад, аддони для управління сценами, швидкого доступу до інструментів та функцій або автоматичного налаштування матеріалів.

Спеціалізовані інструменти для індустрій[14]:

Деякі аддони розроблені для специфічних потреб певних індустрій, таких як архітектурна візуалізація, ігрова розробка, візуальні ефекти та анімація. Це дозволяє професіоналам використовувати спеціалізовані інструменти, розроблені саме для їхніх потреб.

Спільнота та підтримка:

Користувачі можуть знайти багато аддонів, розроблених спільнотою Blender. Це означає, що є доступ до великої кількості безкоштовних та платних аддонів, а також до підтримки та обміну досвідом з іншими користувачами.

Висновки до другого розділу

В другому розділі було продемонстровано основні етапи розробки 3D – моделі, такі як створення базової форми, проектування UV – розгортки її покращення, та використання в процесі накладання текстур.

Також була показана робота декількох спеціалізованих інструментів в середовищі Blender а саме Bevel та Subdivision surface.

Був продемонстрований поетапний процес експорту та імпорту моделі в різні середовища.

Також вдалося розповісти про корисність та функціональність передвстановлених та користувацьких аддонів в середовищі Blender.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		51

РОЗДІЛ 3. ОПИС ТЕСТУВАННЯ ГРАФІЧНОЇ СКЛАДОВОЇ ІГРОВОГО ДОДАТКУ

3.1 Основні процеси тестування графічної складової

Тестування графічної складової гри є складним і багатоетапним процесом, який вимагає глибокого розуміння графічних технологій, ретельного планування та виконання. Це дозволяє виявити й усунути будь-які візуальні дефекти, забезпечуючи таким чином високий рівень якості гри. Ось детальний опис цього процесу з додатковими деталями та поясненнями.

1. Планування тестування

Визначення вимог

Процес тестування графіки починається з розуміння графічних вимог гри. На цьому етапі потрібно зібрати інформацію про:

Візуальний стиль: Це включає в себе художній напрямок, наприклад, реалізм, мультяшність або футуризм.

Текстури та матеріали: Специфікації текстур (роздільна здатність, формати), типи матеріалів (металічні, дерев'яні, кам'яні) (рис 3.1).

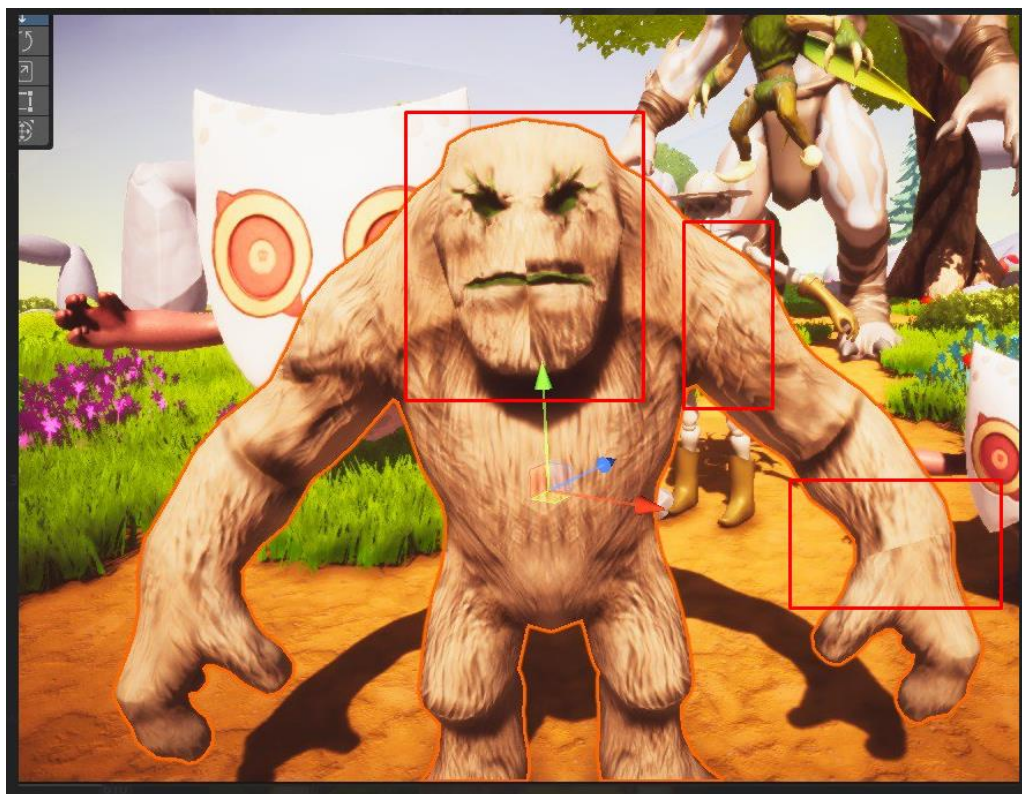


Рисунок 3.1 - Некоректна робота Normal Map

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		52

Освітлення та тіні: Типи освітлення (динамічне, статичне), поведінка тіней (рис 3.2).



Рисунок 3.2 - Некоректна робота освітлення

Анімації: Різновиди анімацій (персонажі, об'єкти, навколишнє середовище) (рис 3.3).



Рисунок 3.3 - Некоректна робота анімації одного з ворогів.

Візуальні ефекти: Ефекти, такі як дим, вогонь, вибухи, водні ефекти.

Частота кадрів (FPS)[18]: Цільова частота кадрів для плавного ігрового процесу.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Складання плану тестування

На основі зібраних вимог розробляється детальний план тестування, який включає:

Методи тестування: Вибір методів тестування, таких як ручне та автоматизоване тестування.

Обсяг тестів: Визначення того, які аспекти графіки потрібно протестувати.

Критерії прийняття: Встановлення критеріїв, за якими буде оцінюватися якість графіки.

Ресурси: Визначення необхідних ресурсів, таких як тестувальники, обладнання та програмне забезпечення.

Графік: Створення графіку тестування з термінами виконання кожного етапу.

2. Створення тестових сценаріїв

Аналіз гри

Глибокий аналіз документації гри, включаючи:

Технічні специфікації: Описання всіх технічних аспектів графіки.

Дизайнерські документи: Візуальні референси та приклади.

Попередні результати тестування: Виявлені раніше проблеми та способи їх вирішення.

Написання тестових сценаріїв

Розробка тестових сценаріїв для всіх графічних елементів, які потребують тестування:

Текстури: Перевірка якості та правильності застосування текстур.

Моделі: Оцінка полігональних моделей на наявність артефактів.

Анімації: Тестування плавності та коректності анімацій.

Освітлення: Оцінка правильності освітлення сцени.

Ефекти: Перевірка роботи візуальних ефектів (рис 3.4).

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		54



Рисунок 3.4 - Приклад багу стріли застрягли в повітрі

Створення тестових випадків

Тестові сценарії декомпонуються у конкретні тестові випадки з чіткими очікуваними результатами, що дозволяє систематично підходити до тестування кожного графічного аспекту гри.

3. Підготовка середовища тестування

Налаштування тестових середовищ

Підготовка різних конфігурацій системи для тестування, включаючи:

Різні платформи: ПК, консолі, мобільні пристрої.

Версії операційних систем: Windows, macOS, Linux, Android, iOS.

Апаратне забезпечення: Різні відеокарти, процесори, обсяги оперативної пам'яті.

Створення тестових даних

Підготовка збережених ігор, профілів користувачів, налаштувань та інших даних, необхідних для тестування:

Збережені ігри: Створення точок збереження для швидкого доступу до різних частин гри.

Профілі користувачів: Налаштування профілів з різними налаштуваннями графіки.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		55

Конфігурації: Застосування різних графічних налаштувань для тестування адаптивності гри.

4. Виконання тестування

4.1. Ручне тестування

Перевірка текстур та моделей

Перевірка текстур: Виявлення артефактів, неправильного мапування UV, розмитих або неправильно застосованих текстур.

Перевірка моделей: Перевірка полігональних моделей на наявність артефактів, дефектів та проблем з топологією.

Перевірка освітлення та тіней

Освітлення сцени: Оцінка якості освітлення в різних умовах (денне, нічне, штучне освітлення).

Динамічні тіні: Перевірка правильності відображення та руху тіней у динамічних сценах.

Перевірка анімацій

Плавність анімацій: Оцінка плавності та реалістичності анімацій персонажів, об'єктів та навколишнього середовища.

Синхронізація анімацій: Перевірка синхронізації анімацій з іншими візуальними та звуковими ефектами.

Перевірка візуальних ефектів

Ефекти частинок: Перевірка якості та коректності роботи ефектів частинок (вибухи, дим, дощ).

Спеціальні ефекти: Оцінка якості візуальних ефектів, таких як вода, вогонь, магичні ефекти.

Перевірка інтерфейсу користувача (UI)

Візуальне представлення UI: Оцінка якості та коректності відображення інтерфейсу.

Адаптивність UI: Перевірка роботи інтерфейсу на різних розширеннях екрану та пристроях.

4.2. Автоматизоване тестування

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		56

Розробка тестових скриптів

Створення автоматизованих тестів для перевірки графічних елементів гри:

Текстури: Скрипти для автоматичного виявлення артефактів у текстурах.

Моделі[11]: Скрипти для перевірки правильності топології та відсутності полігональних дефектів.

Виконання автоматизованих тестів

Регулярне виконання автоматизованих тестів для перевірки стабільності графічної складової та виявлення нових проблем:

Планування тестування: Налаштування автоматизованих тестів для регулярного виконання на різних етапах розробки.

Аналіз результатів: Оцінка результатів автоматизованих тестів для виявлення нових проблем та регресій.

5. Перформанс-тестування

Тестування частоти кадрів (FPS)

Перевірка стабільності частоти кадрів у різних сценах гри:

Стрес-тести: Виконання тестів у сценах з високим навантаженням для виявлення падінь FPS.

Аналіз причин падінь FPS: Виявлення та аналіз причин падінь частоти кадрів.

Тестування завантаження системи

Вимірювання навантаження на процесор, графічний процесор та оперативну пам'ять під час гри:

Моніторинг системних ресурсів: Використання інструментів для моніторингу навантаження на систему.

Оптимізація продуктивності: Внесення змін для оптимізації продуктивності гри на основі результатів тестів.

Приклади Візуальних Багів у Популярних AAA Іграх

1. Cyberpunk 2077

Cyberpunk 2077 від студії CD Projekt Red зіткнувся з численними візуальними багами під час свого релізу. Ось деякі з них:

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		57

Зникаючі текстури: В різних частинах гри текстури об'єктів могли зникати або не завантажуватися належним чином, що створювало дивні та неприродні сцени (рис. 3.5).



Рисунок 3.5 - Приклад невдалого завантаження текстур в Cyberpunk 2077

Поп-ап об'єкти: Об'єкти та персонажі могли раптово з'являтися або зникати на екрані, що створювало відчуття неузгодженості та порушувало занурення у гру.

Проблеми з тінями: Тіні могли відображатися неправильно або взагалі відсутні, що негативно впливало на освітлення та атмосферу гри.

Неправильна анімація персонажів: Персонажі могли рухатися неприродно або застрягати в об'єктах, що викликало смішні та іноді дратівливі сцени.

2. Assassin's Creed Unity

Assassin's Creed Unity, випущений Ubisoft, став відомий завдяки численним візуальним багам на момент релізу:

Зникання облич: Один із найбільш відомих багів полягав у тому, що обличчя персонажів могли зникати, залишаючи лише очі та зуби, що виглядало досить страшно (рис. 3.6).

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		58



Рисунок 3.6 - Приклад багу 3D-моделі

Поп-ап текстури: Текстури могли з'являтися із затримкою, що призводило до неповного відображення оточення на короткий час.

Проблеми з моделями персонажів: Персонажі могли провалюватися крізь підлогу або стіни, застрягати в об'єктах чи мати неправильну позу.

Освітлення: Неправильне відображення джерел світла, що призводило до дивних тіней і відблисків.

3. The Elder Scrolls V: Skyrim

Skyrim, розроблений Bethesda Game Studios, також мав свою частку візуальних багів (рис. 3.7).



Рисунок 3.7 - Приклад багів в The Elder Scrolls V: Skyrim

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		59

Провалювання крізь об'єкти: Персонажі та об'єкти могли провалюватися крізь текстури землі або будівель, що робило неможливим продовження гри без перезавантаження.

Неправильне відображення води[13]: Вода могла відображатися неправильно або зникати, створюючи дивні та неприродні ефекти.

Текстури низької якості: У деяких випадках текстури могли відображатися у низькій якості, навіть якщо налаштування гри були встановлені на високий рівень.

Проблеми з анімацією [8]: Деякі NPC могли рухатися неправильно або мати дивні пози через проблеми з анімацією.

4. Red Dead Redemption 2

Навіть такий технічно досконалий проект, як Red Dead Redemption 2 від Rockstar Games, мав свої візуальні баги:

Поп-ап об'єкти: У деяких випадках дерева, будівлі та інші об'єкти могли з'являтися занадто пізно, створюючи ефект їхнього миттєвого появи.

Проблеми з освітленням: Неправильне відображення джерел світла, що призводило до незвичайних тіней та відблисків.

Анімації персонажів: Персонажі могли застрягати в об'єктах або мати неприродні рухи через помилки в анімації (рис. 3.8).



Рисунок 3.8 - Приклад некоректної анімації в Red Dead Redemption 2

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		60

Неправильні тіні: Тіні могли відображатися некоректно або змінюватися під неприродними кутами.

Візуальні баги є невід'ємною частиною процесу розробки відеоігор, особливо в масштабних AAA проектах. Незважаючи на ретельне тестування та перевірку, деякі проблеми можуть залишатися невиявленими до моменту релізу або з'являтися лише в певних умовах. Важливо, що розробники постійно працюють над виправленням таких багів, випускаючи оновлення та патчі для покращення якості гри.

Висновки до третього розділу

Візуальні баги є поширеним явищем навіть у найпопулярніших AAA іграх. Вони можуть варіюватися від незначних графічних недоліків до серйозних помилок, які впливають на ігровий процес.

Було дитально описано всі етапи тестування графічної складової а також наведено приклад графічних багів з ігрового додатку та декількох популярних AAA ігор.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		61

Висновки

В ході розробки кваліфікаційної роботи було виконано необхідний об'єм роботи втконано всі поставлені задачі та проаналізовано 3 відомі гри ("Divinity: Original Sin", "Deep Rock Galactic", "Serious sam 2") що допомогло визначити основні відмінності стилізованої графіки від реалістичного стилю і з допомогою визначення цих відмінностей вдалося покращити візуальну частину ігрового продукту.

Основна увага була приділена вибору програмного забезпечення. Були вибрані 2 основні програми в яких буде вестися розробка моделей а саме Substance 3D Painter та Blender.

Було продемонстровано основні етапи розробки 3D – моделі, такі як створення базової форми, проектування UV – розгортки її покращення, та використання в процесі накладання текстур.

Також була показана робота декількох спеціалізованих інструментів в середовищі Blender а саме Bevel та Subdivision surface.

Був продемонстрований поетапний процес експорту та імпорту моделі в різні середовища.

Була доведена корисність та функціональність передвстановлених та користувацьких аддонів в середовищі Blender.

Були продемонстровані приклади візуальних багів які є поширеним явищем навіть у найпопулярніших AAA іграх. Вони можуть варіюватися від незначних графічних недоліків до серйозних помилок, які впливають на ігровий процес.

Було дитально описано всі етапи тестування графічної складової а також наведено приклад графічних багів з ігрового додатку та декількох популярних AAA ігор.

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		62

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Divinity: Original Sin [Електронний ресурс] // Steam - Режим доступу до ресурсу: https://store.steampowered.com/app/373420/Divinity_Original_Sin_Enhanced_Edition/
2. Deep Rock Galactic [Електронний ресурс] // Steam - Режим доступу до ресурсу: https://store.steampowered.com/app/548430/Deep_Rock_Galactic/
3. Serious Sam 2 [Електронний ресурс] // Steam - Режим доступу до ресурсу: https://store.steampowered.com/app/204340/Serious_Sam_2/
4. Blender nodes [Електронний ресурс] // Blender market - <https://blendermarket.com/categories/addons/browse>
5. Blender surfacing [Електронний ресурс] // Blender market - <https://blendermarket.com/categories/surfacing>
6. Blender video tutorials [Електронний ресурс] // Blender market - <https://blendermarket.com/categories/video-tutorials>
7. Blender e-books [Електронний ресурс] // Blender market - <https://blendermarket.com/categories/ebooks>
8. Blender animation [Електронний ресурс] // Blender market - <https://blendermarket.com/categories/animation-scripts-and-addons>
9. Blender materials and shading [Електронний ресурс] // Blender market - <https://blendermarket.com/categories/materials-and-shading-addons>
10. Blender tools [Електронний ресурс] // Blender market - <https://blendermarket.com/categories/tools>
11. Blender compositing presets [Електронний ресурс] // Blender market - <https://blendermarket.com/categories/compositing-presets>
12. Adobe substance 3D painter [Електронний ресурс] // Adobe Substance 3D Community - <https://substance3d.adobe.com/community-assets?assetType=painterSmartMaterial>
13. Unity render pipelines [Електронний ресурс] // Unity documentation - <https://docs.unity3d.com/Manual/render-pipelines.html>

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		63

14. Unity cameras [Електронний ресурс] // Unity documentation - <https://docs.unity3d.com/Manual/CamerasOverview.html>
15. Unity post-processing [Електронний ресурс] // Unity documentation - <https://docs.unity3d.com/Manual/PostProcessingOverview.html>
16. Unity lighting [Електронний ресурс] // Unity documentation - <https://docs.unity3d.com/Manual/LightingOverview.html>
17. Unity shaders [Електронний ресурс] // Unity documentation - <https://docs.unity3d.com/Manual/Shader.html>
18. Unity graphics performance [Електронний ресурс] // Unity documentation - <https://docs.unity3d.com/Manual/OptimizingGraphicsPerformance.html>
19. Unity color space [Електронний ресурс] // Unity documentation - <https://docs.unity3d.com/Manual/LinearLighting.html>
20. Unity world building [Електронний ресурс] // Unity documentation - <https://docs.unity3d.com/Manual/CreatingEnvironments.html>

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		64

ДОДАТКИ

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		65

Quadric Error Metrics Simplification

```

import bpy
import bmesh
from mathutils import Vector, Matrix
import numpy as np
from concurrent.futures import ThreadPoolExecutor

bl_info = {
    "name": "Quadric Error Metrics Decimator",
    "blender": (2, 80, 0),
    "category": "Object",
}

def calculate_face_normal(v1, v2, v3):
    normal = (v2 - v1).cross(v3 - v1)
    normal.normalize()
    return normal

def calculate_quadric(v1, v2, v3):
    normal = calculate_face_normal(v1, v2, v3)
    d = -normal.dot(v1)
    q = np.zeros((4, 4))
    normal_homogeneous = np.append(normal, d)
    for i in range(4):
        for j in range(4):
            q[i][j] = normal_homogeneous[i] * normal_homogeneous[j]
    return q

def calculate_vertex_quadrics(obj):
    bm = bmesh.new()
    bm.from_mesh(obj.data)
    vertex_quadrics = {v.index: np.zeros((4, 4)) for v in bm.verts}

    for face in bm.faces:
        v1, v2, v3 = face.verts[0], face.verts[1], face.verts[2]
        q = calculate_quadric(v1.co, v2.co, v3.co)
        for v in face.verts:
            vertex_quadrics[v.index] += q

    bm.free()
    return vertex_quadrics

def compute_error(v1_quadric, v2_quadric, v_new):
    q = v1_quadric + v2_quadric
    v_homogeneous = np.array([v_new.x, v_new.y, v_new.z, 1])
    error = v_homogeneous.dot(q).dot(v_homogeneous)
    return error

def optimal_contraction(v1_quadric, v2_quadric):

```

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		66

```

q = v1_quadric + v2_quadric
q_3x3 = q[:3, :3]
q_3x1 = q[:3, 3]
try:
    q_inv = np.linalg.inv(q_3x3)
    v_new = q_inv.dot(-q_3x1)
    return Vector(v_new)
except np.linalg.LinAlgError:
    return None

def is_valid_edge_collapse(v1, v2):
    for edge in v1.link_edges:
        if edge.other_vert(v1) != v2 and edge.other_vert(v1) in v2.link_edges:
            return False
    return True

class MESH_OT_quadric_error_decimate(bpy.types.Operator):
    bl_idname = "mesh.quadric_error_decimate"
    bl_label = "Quadric Error Metrics Decimate"
    bl_options = {'REGISTER', 'UNDO'}

    def execute(self, context):
        obj = context.object
        bpy.ops.object.mode_set(mode='EDIT')
        bpy.ops.mesh.select_all(action='SELECT')
        bpy.ops.mesh.remove_doubles(threshold=1e-6)
        bpy.ops.object.mode_set(mode='OBJECT')

        vertex_quadrics = calculate_vertex_quadrics(obj)
        target_ratio = context.scene.quadric_iterations

        bm = bmesh.new()
        bm.from_mesh(obj.data)
        original_edge_count = len(bm.edges)
        target_edge_count = int(original_edge_count * (1.0 - target_ratio))

        while len(bm.edges) > target_edge_count:
            edge_collapses = []

            def process_edge(edge):
                v1, v2 = edge.verts
                if v1.index not in vertex_quadrics or v2.index not in
vertex_quadrics:
                    return None
                if not is_valid_edge_collapse(v1, v2):
                    return None
                v1_quadric = vertex_quadrics[v1.index]
                v2_quadric = vertex_quadrics[v2.index]
                v_new = optimal_contraction(v1_quadric, v2_quadric)
                if v_new:
                    error = compute_error(v1_quadric, v2_quadric, v_new)

```

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		67

```

        return (error, v1, v2, v_new)
    return None

    with ThreadPoolExecutor() as executor:
        results = list(executor.map(process_edge, bm.edges))

    edge_collapses = [result for result in results if result]
    edge_collapses.sort()

    for error, v1, v2, v_new in edge_collapses:
        if len(bm.edges) <= target_edge_count:
            break
        if not v1.is_valid or not v2.is_valid:
            continue
        v1.co = v_new
        bmesh.ops.pointmerge(bm, verts=[v1, v2], merge_co=v_new)

    bm.to_mesh(obj.data)

    bm.free()
    return {'FINISHED'}

class MESH_PT_quadric_error_decimate_panel(bpy.types.Panel):
    bl_label = "Quadric Error Metrics Decimate"
    bl_idname = "MESH_PT_quadric_error_decimate_panel"
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = 'Tools'

    def draw(self, context):
        layout = self.layout
        layout.operator("mesh.quadric_error_decimate", text="Optimize")
        layout.prop(context.scene, "quadric_iterations", text="Reduction Ratio")

    def register():
        bpy.types.Scene.quadric_iterations = bpy.props.FloatProperty(
            name="Reduction Ratio",
            default=0.1,
            min=0.0,
            max=1.0,
            description="Target edge count reduction ratio"
        )
        bpy.utils.register_class(MESH_OT_quadric_error_decimate)
        bpy.utils.register_class(MESH_PT_quadric_error_decimate_panel)

    def unregister():
        del bpy.types.Scene.quadric_iterations
        bpy.utils.unregister_class(MESH_OT_quadric_error_decimate)
        bpy.utils.unregister_class(MESH_PT_quadric_error_decimate_panel)

if __name__ == "__main__":
    register()

```

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		68

Shader node emission

```

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

#include "node_shader_util.hh"

namespace blender::nodes::node_shader_emission_cc {

static void node_declare(NodeDeclarationBuilder &b)
{
    b.add_input<decl::Color>("Color").default_value({1.0f, 1.0f, 1.0f, 1.0f});
    b.add_input<decl::Float>("Strength").default_value(1.0f).min(0.0f).max(1000000.0f);
    b.add_input<decl::Float>("Weight").unavailable();
    b.add_output<decl::Shader>("Emission");
}

static int node_shader_gpu_emission(GPUMaterial *mat,
                                     bNode *node,
                                     bNodeExecData * /*execdata*/,
                                     GPUNodeStack *in,
                                     GPUNodeStack *out)
{
    GPU_material_flag_set(mat, GPU_MATFLAG_EMISSION);
    return GPU_stack_link(mat, node, "node_emission", in, out);
}

NODE_SHADER_MATERIALX_BEGIN
#ifdef WITH_MATERIALX
{
    if (to_type_ != NodeItem::Type::EDF) {
        return empty();
    }

    NodeItem color = get_input_value("Color", NodeItem::Type::Color3);
    NodeItem strength = get_input_value("Strength", NodeItem::Type::Float);

    return create_node("uniform_edf", NodeItem::Type::EDF, {"color", color *
    strength}));
}
#endif
NODE_SHADER_MATERIALX_END

} // namespace blender::nodes::node_shader_emission_cc

/* node type definition */
void register_node_type_sh_emission()

```

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

{
    namespace file_ns = blender::nodes::node_shader_emission_cc;

    static blender::bke::bNodeType ntype;

    sh_node_type_base(&ntype, SH_NODE_EMISSION, "Emission", NODE_CLASS_SHADER);
    ntype.declare = file_ns::node_declare;
    ntype.gpu_fn = file_ns::node_shader_gpu_emission;
    ntype.materialx_fn = file_ns::node_shader_materialx;

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

```

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				70
Змн.	Арк.	№ докум.	Підпис	Дата		

Декілька варіацій моделей візка

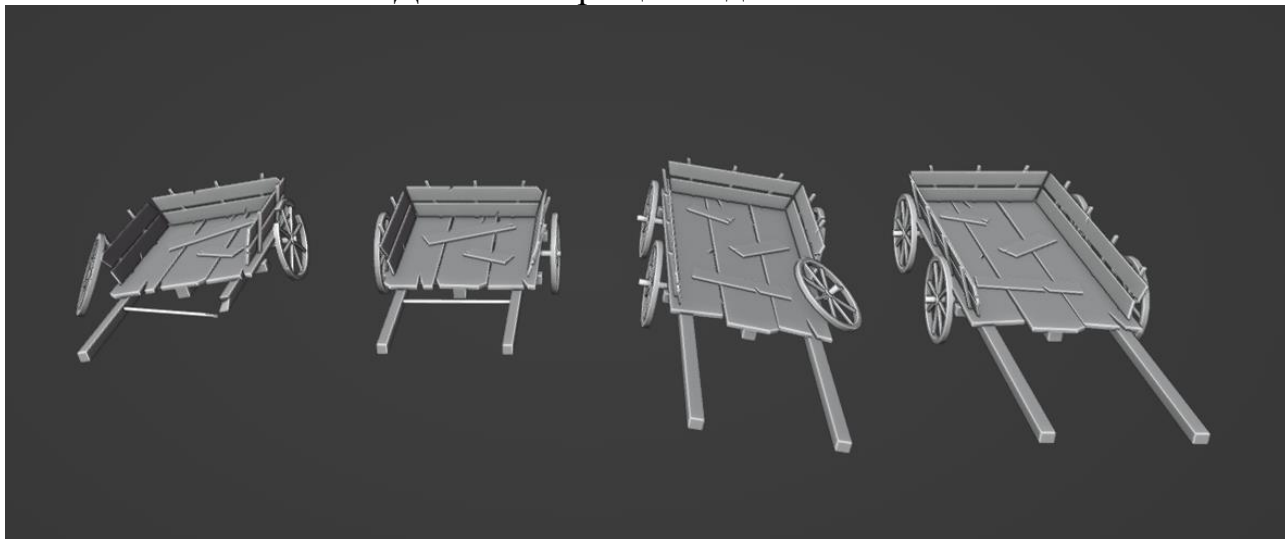


Рисунок В.1 - Базові моделі візка

Декілька варіацій моделей підсвічника

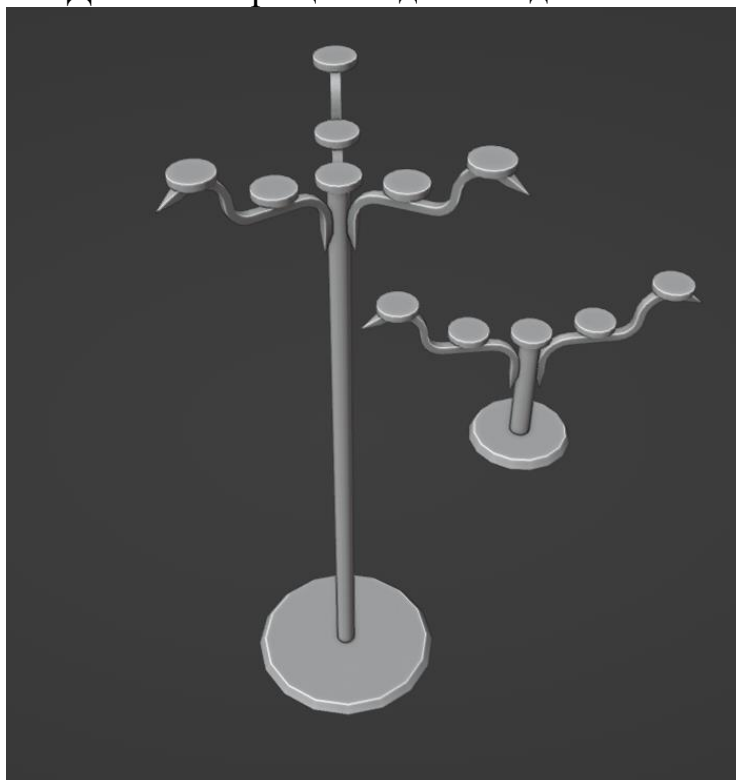


Рисунок В.2 - Базові моделі підсвічника

Декілька варіацій моделей паркану

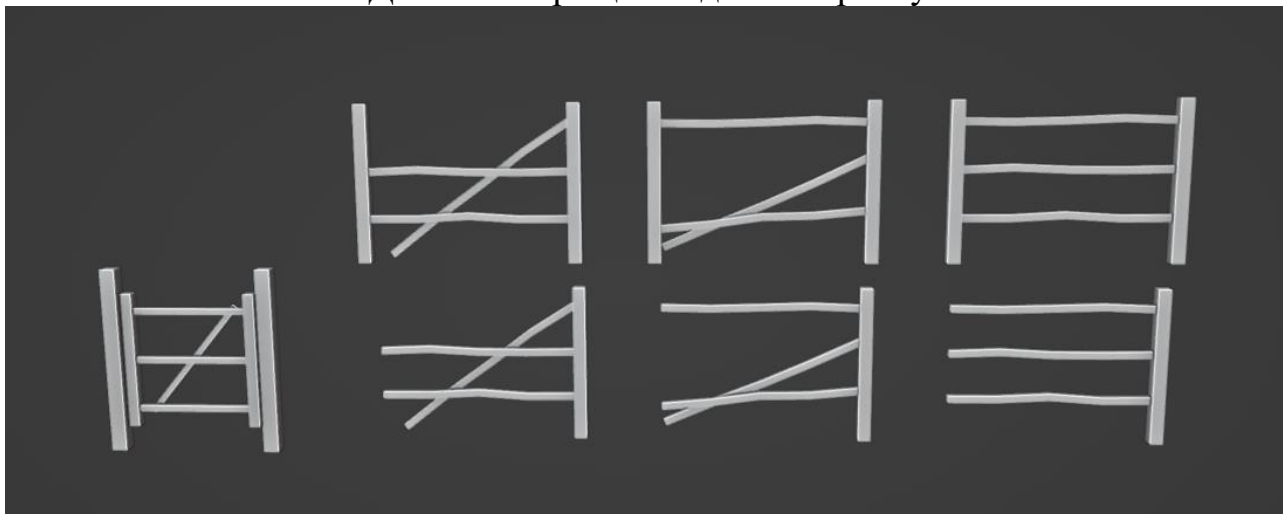


Рисунок В.3 - Базові моделі паркану

Декілька моделей бочка, табурет та ящик



Рисунок В.4 - Базові моделі придатні до сидіння

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		72

Модель великої ємності

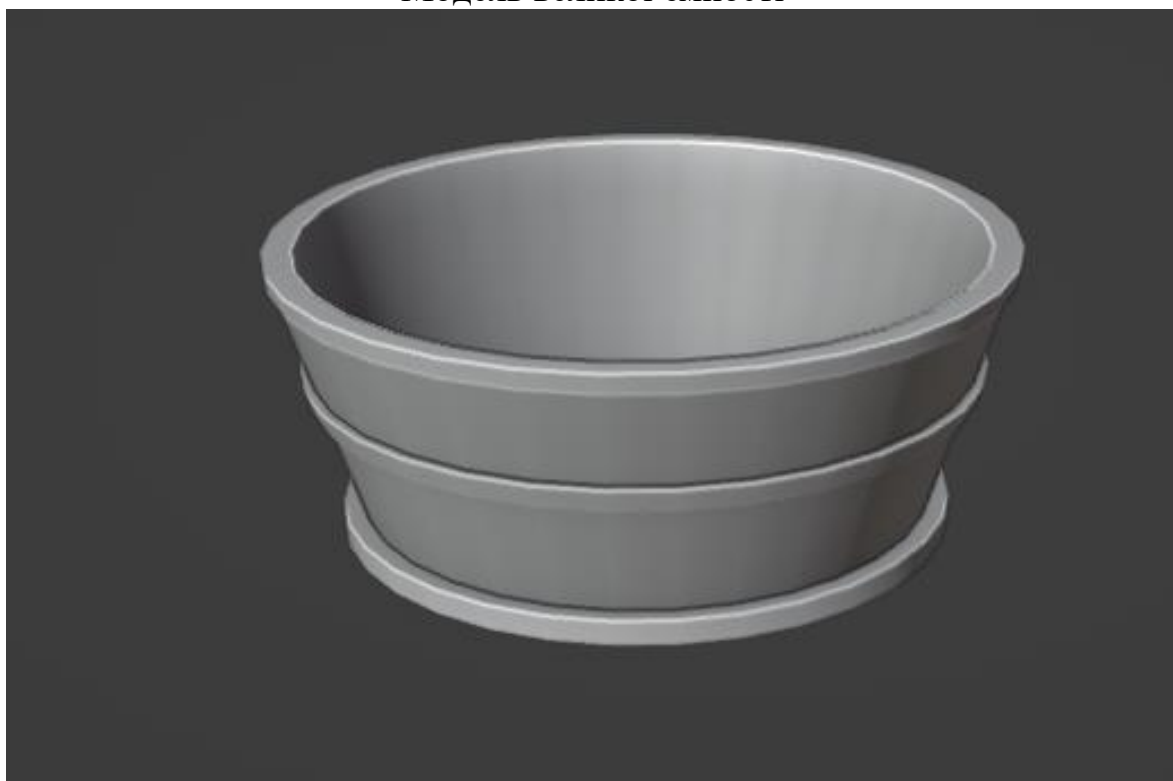


Рисунок В.5 - Базова модель великої ємності

Декілька варіацій моделі колодязя

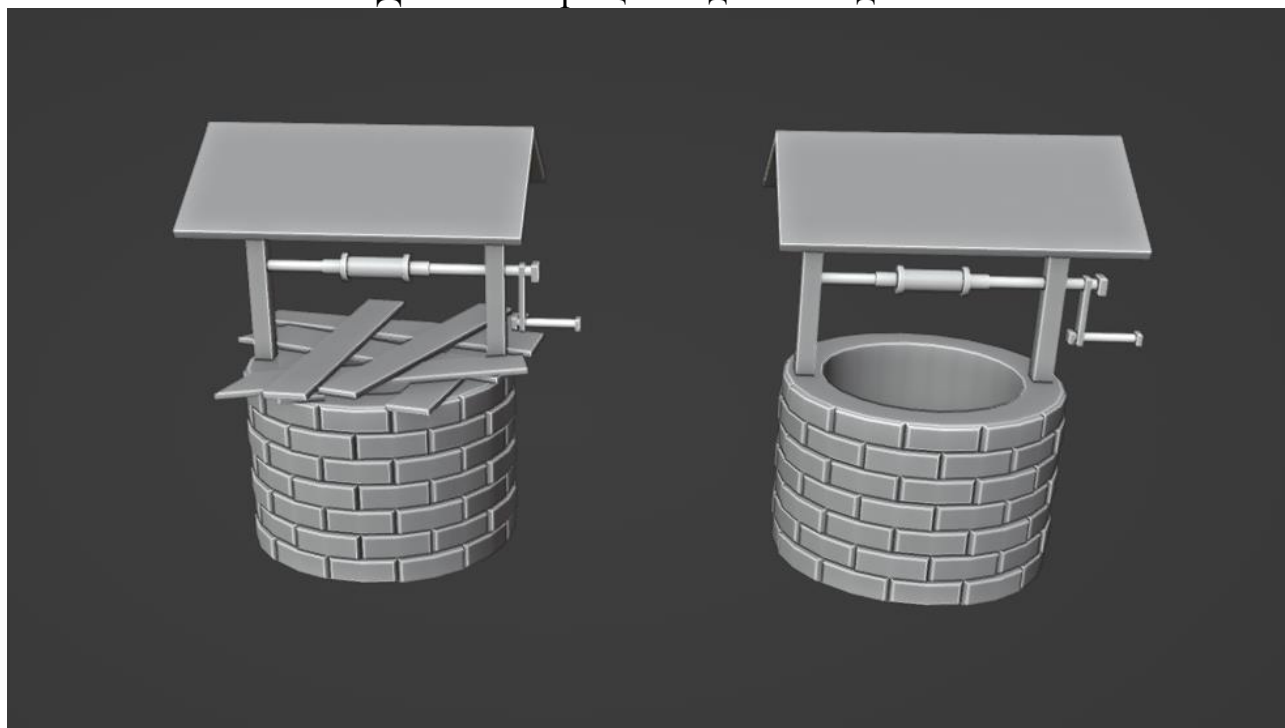


Рисунок В.6 - Базові моделі колодязя

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		73

Декілька варіацій моделей колон

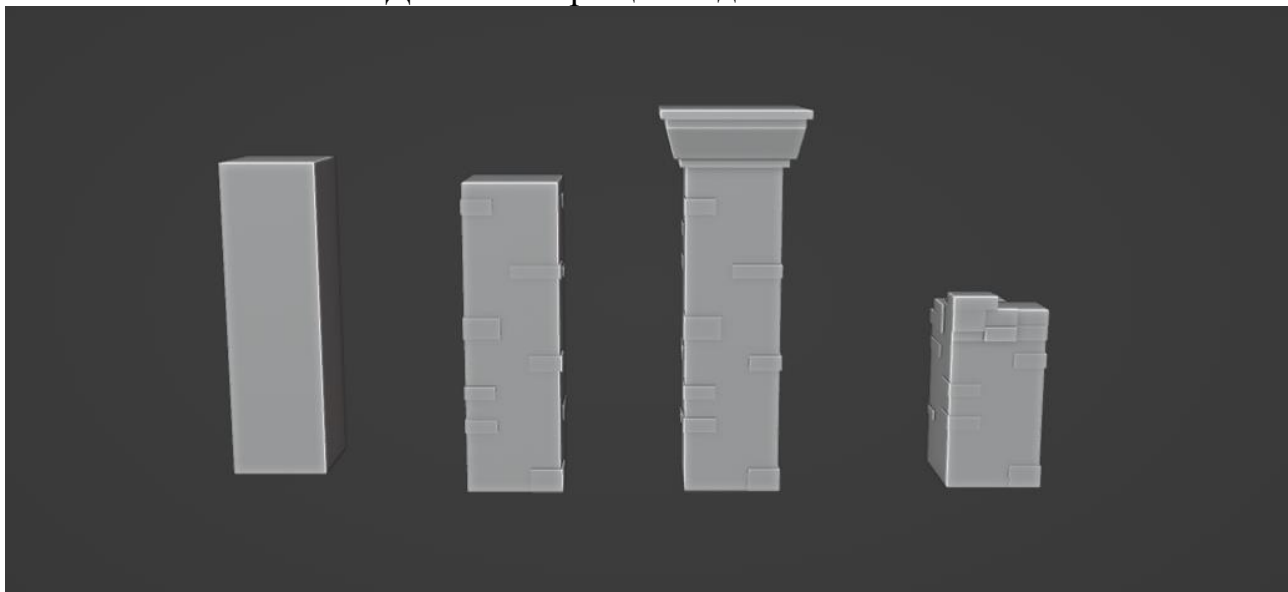


Рисунок В.7 - Базові моделі колон

Декілька варіацій моделей стіни

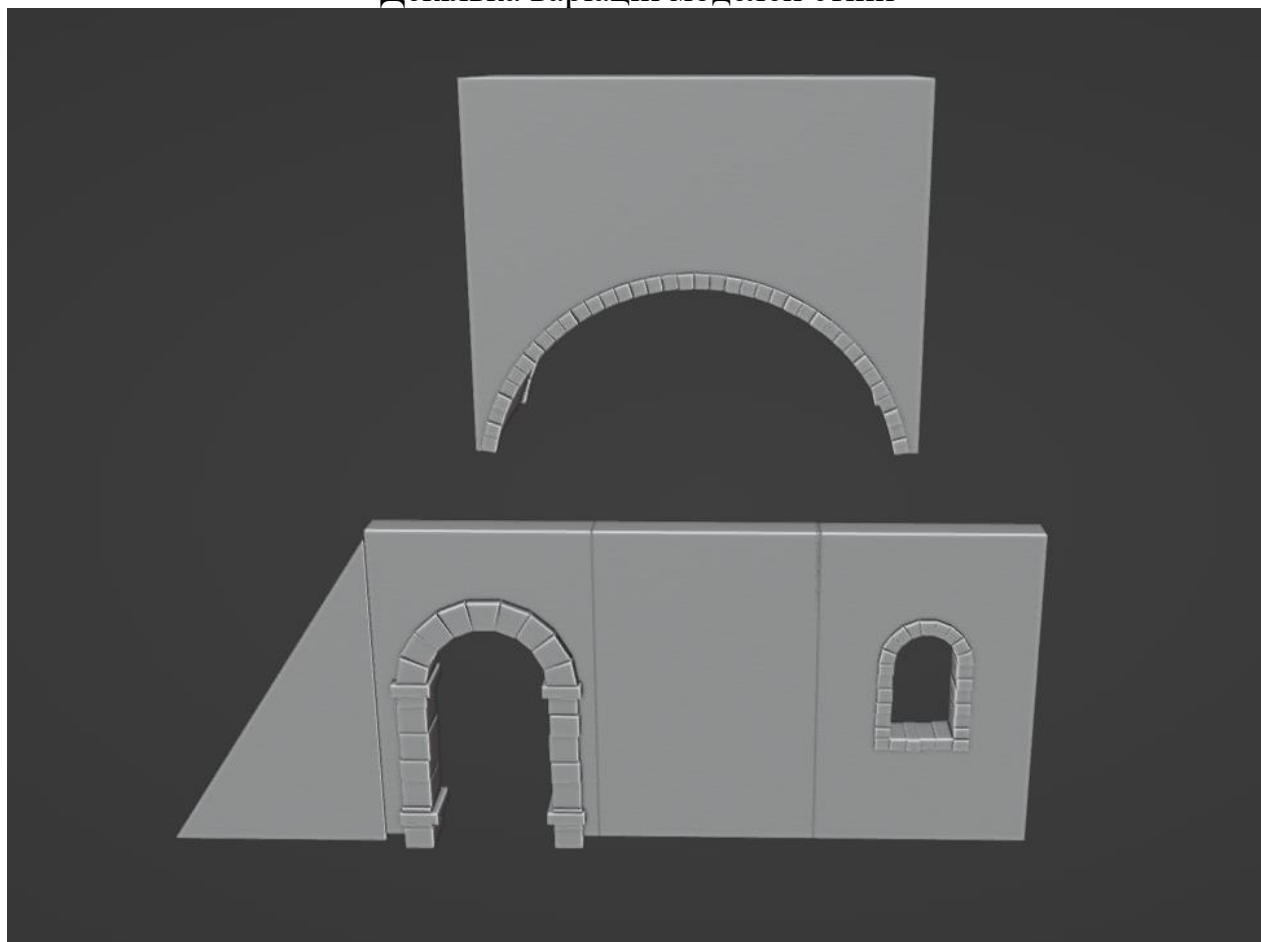


Рисунок В.8 - Базові моделі стін

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		74

Декілька варіацій моделі шторів

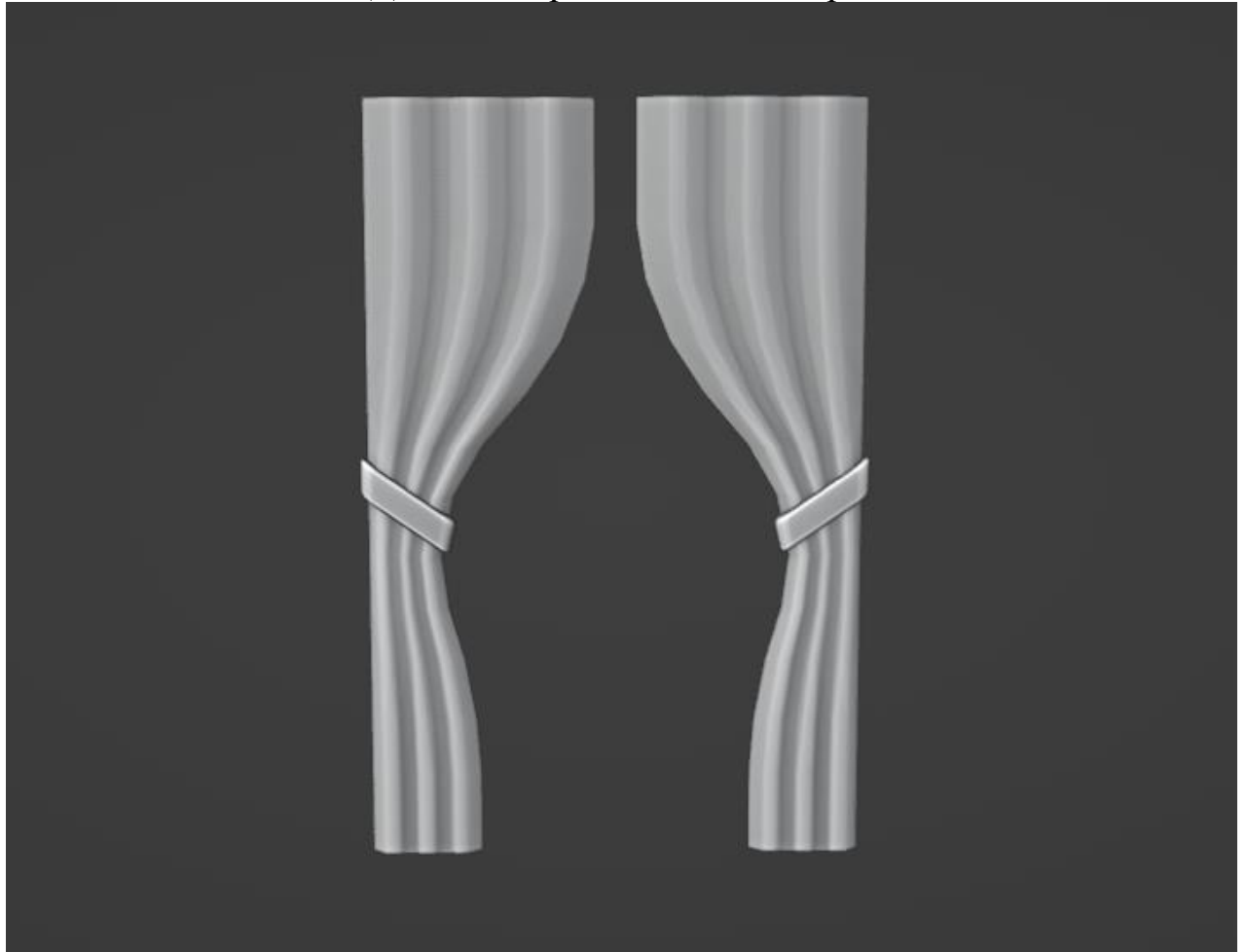


Рисунок В.9 - Базові моделі шторів

Декілька варіацій моделей будівничого сміття

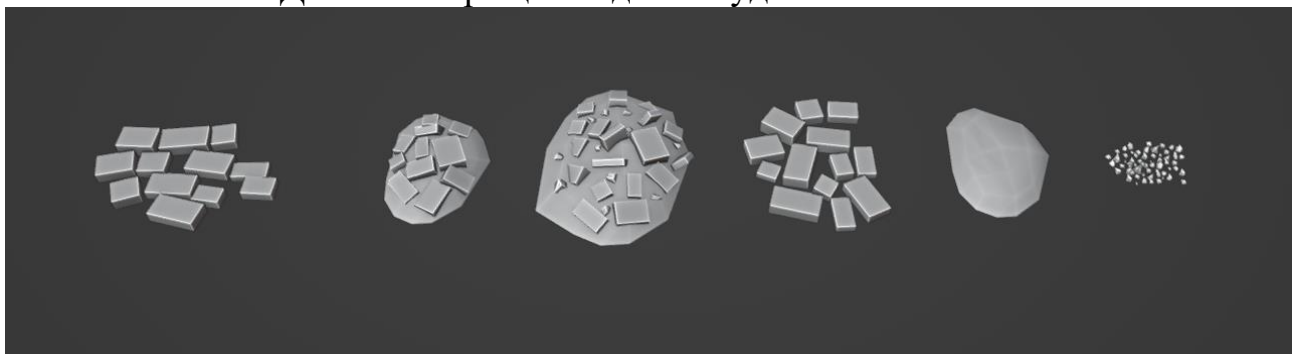


Рисунок В.10 - Базові моделі будівничого сміття

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		75

Модель свічкової люстри

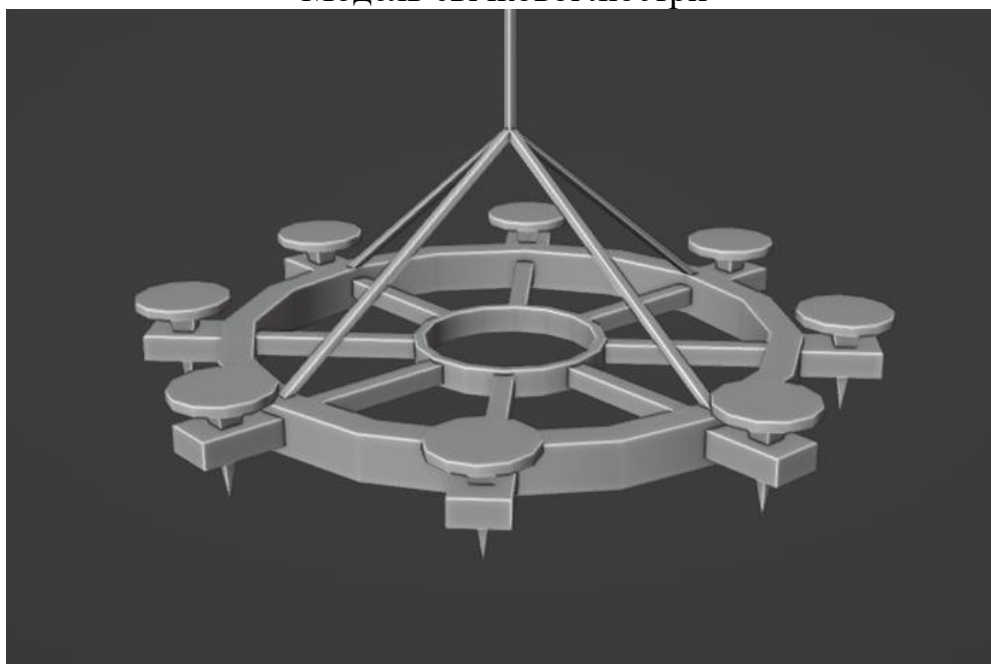


Рисунок В.11 - Базова модель свічкової люстри

Модель каміна



Рисунок В.12 - Базова модель каміна

Модель шафи



Рисунок В.13 - Базова модель шафи

Декілька варіацій зламаного даху

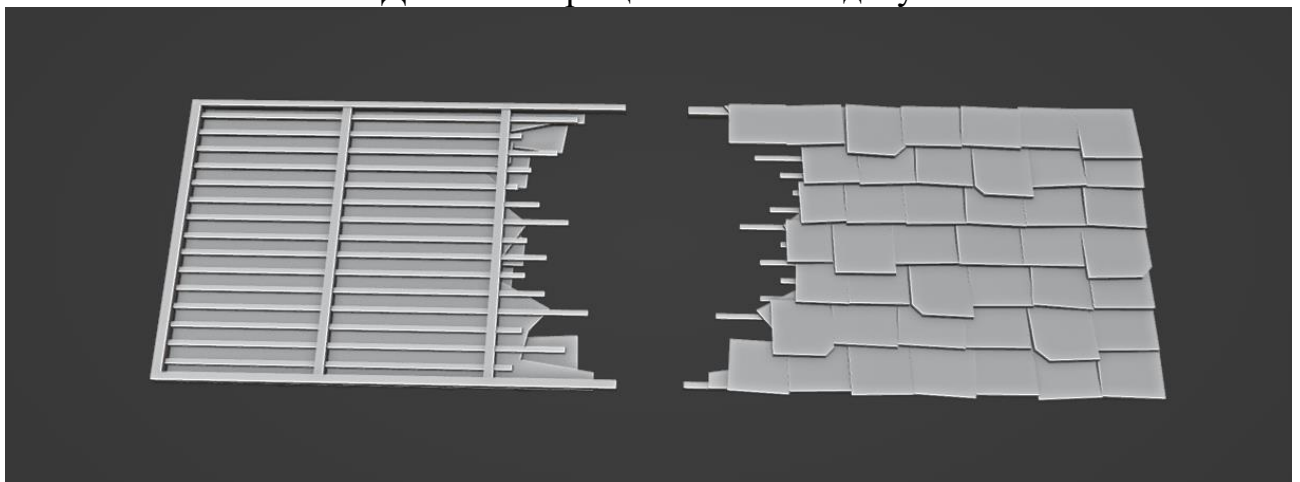


Рисунок В.14 – Базові моделі зламаного даху

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		77

Одна з варіацій ігрового арбалету



Рисунок В.15 – Запечена модель арбалета з накладеною текстурою

Одна з варіацій кружки



Рисунок В.16 – Запечена модель кружки з накладеною текстурою

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				
Змн.	Арк.	№ докум.	Підпис	Дата		78

Одна з варіацій бутилки



Рисунок В.17 – Запечена модель бутилки з накладеною текстурою

		В.Д.Семенець			КН.КР.Б – 122 – 24 – ПЗ	Арк.
		А.Р.Петросян				79
Змн.	Арк.	№ докум.	Підпис	Дата		