МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
Фізико-математичний факультет
Кафедра інформатики та прикладної математики

«Допущено до захисту»
В.о завідувача кафедри
_____ Семеріков С.О.
_____ 2025 р.

Реєстраційний № _____
«___» _____ 2025 р.

РОЗРОБКА НАБОРУ 3D-МОДЕЛЕЙ ДЛЯ ІГРОВОГО ОТОЧЕННЯ

Кваліфікаційна робота студента
групи І-21
ступінь вищої освіти «бакалавр»
спеціальності
014 Середня освіта (Інформатика)
Кучерова Богдана Олександровича
Керівник:
кандидат фізико-математичних наук,
доцент
Моісеєнко Наталя Володимирівна

Оцінка:
Національна шкала _____
Шкала ECTS_____ кількість балів___

Члени комісії
_____ _____
_____ _____
_____ _____

Кривий Ріг – 2025

ЗАПЕВНЕННЯ

Я, Кучеров Богдан Олександрович, розумію і підтримую політику Криворізького державного педагогічного університету з академічної доброчесності. Запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають покликання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Криворізького державного педагогічного університету ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасна ігрова індустрія демонструє стрімке зростання, постійно збільшуючи вимоги до якості та реалістичності ігрового контенту. За прогнозами, глобальний ринок відеоігор у 2025 році сягне понад 250 мільярдів доларів [1]. Водночас ключовим елементом привабливості та занурення в ігровий світ виступають саме 3D-моделі, що складають основу візуального сприйняття гравцем. Розробка високоякісного набору 3D-моделей для ігрового оточення є запорукою конкурентоспроможності проєктів та їхньої відповідності сучасним стандартам.

Ігрове оточення, як частина гейм-дизайну, значно впливає на ігровий досвід. Дослідження підтверджують, що якісні 3D-об’єкти підвищують рівень залученості користувача [2]. Популярність віртуальної та доповненої реальностей, мобільних платформ та ігор з відкритим світом ставить все нові вимоги до візуальної складової, вимагаючи підвищення якості та деталізації. Це, у свою чергу, значно ускладнює процес створення ігрових світів, які мають бути не тільки візуально вражаючими та реалістичними, але й максимально ефективними з огляду на обмежені ресурси комп'ютера або мобільного пристрою. Це ставить нові вимоги до 3D-моделей, зокрема, потребує адаптивності, оптимізованої геометрії та використання процедурних технік, що забезпечують створення складних сцен з мінімальним навантаженням на обчислювальну потужність.

Індустрія вимагає оптимізації моделей, адже високополігональні об’єкти можуть негативно впливати на продуктивність гри [3]. Тому розробка оптимізованих 3D-моделей, що зберігають високу деталізацію та відповідають технічним вимогам, є нагальною проблемою.

Важливо також відзначити, що доступ до якісного набору 3D-ресурсів полегшує процес прототипування та розробки ігор. Використання бібліотек

моделей або створення власних дозволяє розширювати креативні можливості та прискорювати вихід продукту на ринок [4].

Одночасно спостерігається тенденція до зростання різноманіття інструментів та методів, що використовуються для побудови моделей: полігональне, параметричне та процедурне моделювання, цифрове ліплення, фотограмметрія та інші. Такий широкий вибір програмних рішень ускладнює визначення найкращого варіанту для кожного окремого проєкту. Тому аналіз та порівняння сучасних інструментів та методів 3D-моделювання є надзвичайно важливим завданням, що має безпосереднє практичне значення для розробників ігор, дизайнерів та спеціалістів у галузі мультимедіа.

Також спостерігається зростання попиту на кваліфікованих спеціалістів, які мають навички роботи з сучасними інструментами створення тривимірного контенту. Майстерність у професійному програмному забезпеченні та глибоке розуміння принципів моделювання є ключовим елементом цифрової компетентності для фахівців IT, дизайну, кінематографу та галузі розробки відеоігор.

Таким чином, тема роботи «Розробка набору 3D-моделей для ігрового оточення» є актуальною у контексті зростання попиту на якісний візуальний контент, а також потреби оптимізації та адаптації моделей під вимоги сучасних ігрових рушіїв. Дослідження та створення таких наборів дозволяють не лише підвищити якість ігор, а й розширити професійні горизонти розробників у сфері 3D-моделювання.

Мета дослідження – створити набір тривимірних моделей, оптимізованих для використання в ігрових рушіях, обґрунтувавши вибір методів та інструментів 3D-моделювання. Для досягнення зазначеної мети були сформульовані такі завдання:

1. Провести аналіз сучасних методів тривимірного моделювання, з’ясувати їх переваги, недоліки та можливості застосування в розробці ігрових проєктів.
2. Оглянути та порівняти провідні програмні пакети для створення та текстурування 3D-моделей.
3. Проаналізувати можливості та вимоги сучасних ігрових рушіїв до тривимірних моделей.
4. Обрати найбільш доцільні та ефективні інструменти для реалізації практичної частини роботи.
5. Створити набір тривимірних моделей для ігрового оточення.
6. Розробити програму для демонстрації процесу інтеграції створених 3D-моделей до ігрового рушія.

Об’єкт дослідження – процес створення тривимірних моделей для застосування в ігрових проєктах.

Предмет дослідження – методики та програмні засоби тривимірного моделювання, текстурування та інтеграції 3D-моделей у сучасні ігрові рушії.

Методи дослідження. Для досягнення поставлених завдань у роботі застосовано такі методи:

- аналіз і узагальнення науково-технічної літератури, спеціалізованих ресурсів і документації з теми дослідження;
- порівняльний аналіз існуючих програмних рішень та методів створення 3D-моделей;
- класифікація та систематизація інформації для виявлення оптимальних інструментів та підходів до моделювання; практичне експериментування із використанням обраних програмних засобів для створення і тестування моделей в реальних умовах;
- емпіричні методи – тестування створених моделей на відповідність вимогам продуктивності та якості у вибраному ігровому рушії.

Структура роботи. Кваліфікаційна робота складається зі вступу, двох розділів, висновків до кожного розділу, загальних висновків, списку використаних джерел та додатків. Загальний обсяг кваліфікаційної роботи –62 сторінки. Обсяг основного тексту на 49 сторінках. Робота містить 22 рисунки, 3 таблиці.

РОЗДІЛ 1
СЕРЕДОВИЩА РОЗРОБКИ ІГРОВИХ ДОДАТКІВ ТА 3D-МОДЕЛЮВАННЯ

1.1 Основні методи та поняття тривимірного моделювання

Тривимірне (3D) моделювання – це процес створення цифрових тривимірних об’єктів у віртуальному просторі за допомогою спеціалізованого програмного забезпечення. Процес моделювання генерує математичне представлення об'єкта у формі полігональної сітки або поверхні, що відображає його форму та структуру. 3D-моделі широко використовуються в різних галузях, починаючи від розробки відеоігор та анімації, і закінчуючи архітектурною візуалізацією, промисловістю та медициною [5].

В сфері ігрової індустрії 3D-моделювання служить фундаментальною технологією для створення персонажів, локацій, предметів та інших компонентів, які формують ігровий світ [6]. Цей процес поєднує художній задум з точними математичними обчисленнями координат вершин та граней, що забезпечує високу точність та можливість редагування моделей. Існують різні методи тривимірного моделювання, кожен з яких має унікальні характеристики, переваги та найкраще підходить для певних завдань. Серед найпоширеніших методів виділяють такі: полігональне, NURBS-моделювання (поверхневе та параметричне), цифрове ліплення, процедурне, а також допоміжні технології, як-от бокс-моделювання, булеве та воксельне моделювання. Далі розглянемо особливості та приклади використання основних методів.

Полігональне моделювання. Полігональний метод, відомий як один з найстаріших і найпоширеніших методів в комп'ютерній графіці, базується на представленні об'єктів як сіток багатокутників [7]. Найчастіше використовуються трикутники або чотирикутники, які і складають поверхню об'єкта. Моделювання за допомогою полігонів дає можливість поступово

формувати форму об'єкта, маніпулюючи його вершинами, ребрами та гранями. Такий підхід забезпечує повну свободу в зміні топології моделі: можна додати деталі шляхом поділу сітки на менші полігони або спростити модель, видаляючи непотрібні полігони. Так, моделі персонажів та навколишнього середовища в відеоіграх зазвичай створюються у вигляді полігональних сіток з чітко продуманою структурою, що дозволяє досягти балансу між деталізацією та продуктивністю.

Параметричне та NURBS-моделювання. Параметричне моделювання працює з об’єктами, форма яких визначається математичними формулами та налаштуваннями. Одним з видів параметричного підходу є NURBS-моделювання, яке використовує нерівномірні раціональні B-сплайни (Non-Uniform Rational B-Splines) для створення криволінійних поверхонь [7]. NURBS-поверхні не будуються на полігонах, а на гладких кривих, завдяки чому можуть точно відображати ідеально гладкі форми (сфери, тороїди, складні криволінійні елементи) без будь-яких видів "фасетування". Цей підхід широко використовується в промисловому дизайні, механічній інженерії та CAD-системах, де необхідна висока точність, наприклад, при розробці автомобільних кузовів або побутової електроніки. Параметричні моделі зберігають історію створення: дизайнер може коригувати параметри (радіуси, відстані, кути) та автоматично оновлювати модель. Переваги NURBS-моделей полягають у точності та можливості редагування на рівні високих параметрів [8]. Однак, недоліками є складність їх реалізації в іграх (через необхідність конвертації в полігональну сітку для рендерингу) та обмежена інтерактивність складних NURBS-моделей в режимі реального часу. У розробці ігор NURBS-моделі все частіше застосовуються на стадії високополігонального моделювання та для створення основних форм, проте кінцевий ігровий контент конвертується в полігони.

Цифрове ліплення (Digital Sculpting). Цей метод моделювання нагадує процес ліплення з глини, але в цифровому просторі. Програми для цифрового ліплення, такі як ZBrush, Mudbox та Blender, дають змогу "виліплювати" форму високополігональної моделі за допомогою спеціальних інструментів: можна розтягувати, згладжувати, доповнювати об'єм та визначати деталі. Скульптинг особливо корисний для створення органічних форм — персонажів, міфічних істот, складних рельєфів. 3D-моделі, створені цифровим ліпленням, можуть включати мільйони полігонів і зберігати найдрібніші деталі (наприклад, зморшки на шкірі, витончені елементи одягу) [9]. Ключові переваги цифрового моделювання полягають у свободі творчого вираження та вражаючій деталізації; недоліками є необхідність додаткових етапів (ретопологія, запікання карт) для інтеграції результатів в ігри.

Процедурне моделювання. Процедурний підхід до створення моделей ґрунтується на використанні алгоритмів та правил, а не на ручному моделюванні. Художник налаштовує генеративну процедуру, яка автоматично створює 3D-геометрію. Прикладом процедурного моделювання є генерація ландшафтів за допомогою функцій шуму (Perlin noise, Voronoi, Wave, Cloud та ін) [10]. Одним із сучасних інструментів для процедурного моделювання є програма SideFX Houdini [11]. Її перевагою є те, що будь-які зміни параметрів автоматично оновлюють модель, що забезпечує високу гнучкість на пізніх етапах проекту. Перевагами процедурного підходу є висока продуктивність при створенні великого обсягу контенту, гнучкість та повторюваність; недоліками є крута "крива навчання", складність досягнення тонкого контролю над окремими елементами та час, який витрачається на розробку самих процедур.

Інші методи. Окрім зазначених, варто згадати булеве моделювання (побудова об’єктів шляхом виконання булевих операцій – об’єднання, віднімання, перетину примітивних форм), воксельне моделювання (представлення об’єктів у вигляді тривимірної сітки вокселів – кубічних

“пікселів”, характерний приклад – гра Minecraft [12]; фотограмметрію та 3D-сканування (отримання моделей реальних об’єктів шляхом обробки фотографій або лазерного сканування) [13]. Зазначені підходи часто виступають допоміжними інструментами: приміром, фотограмметрія активно використовується для створення ігрового оточення.

Для кожного конкретного завдання необхідно обирати метод, що забезпечить потрібний баланс між швидкістю створення, рівнем деталізації та оптимальністю моделі для подальшого використання. Наприклад, при розробці набору 3D-моделей для ігрового оточення доцільно комбінувати методи: базові об’єкти правильної форми можуть бути створені полігональним або параметричним моделюванням, складні органічні деталі – виліплені засобами цифрового скульптингу, повторювані елементи середовища – згенеровані процедурно, а реалістичні текстури – отримані через фотограмметрію.

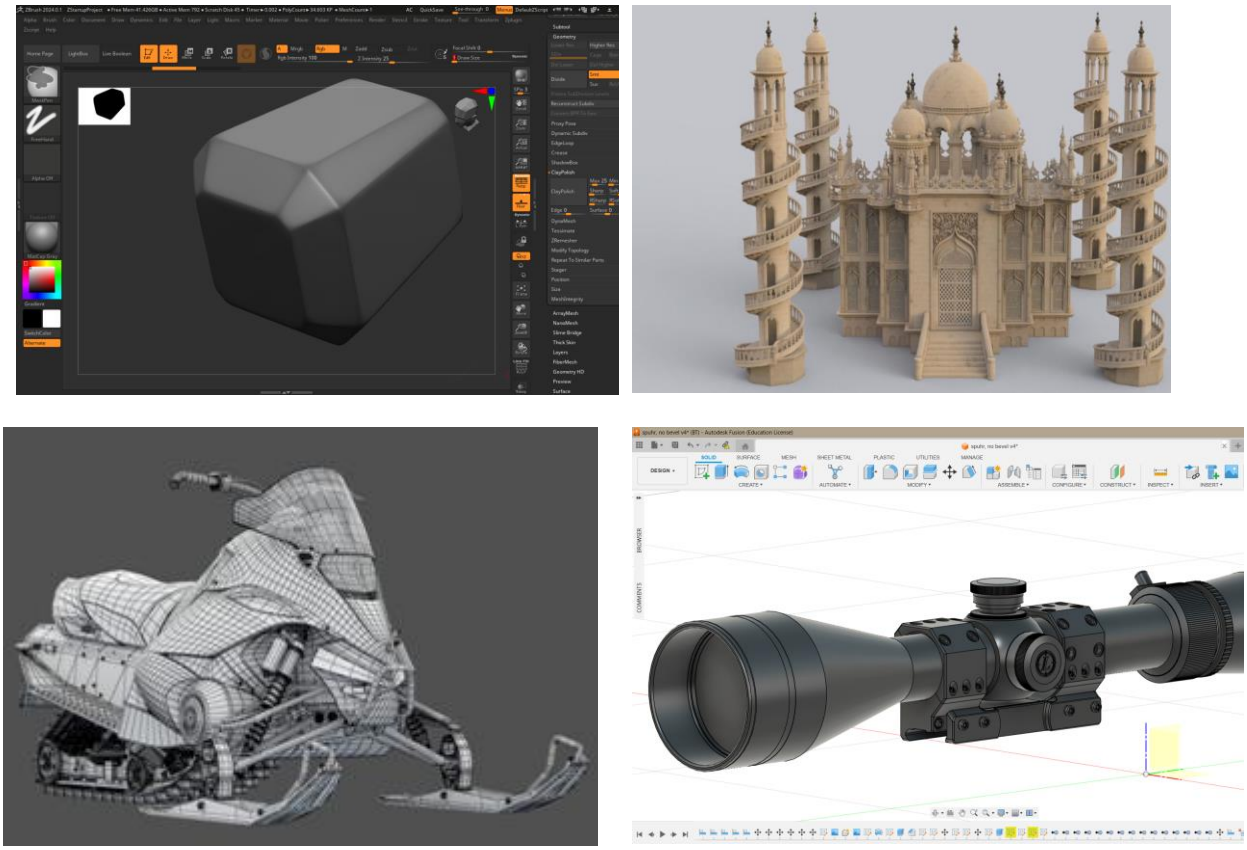


Рис. 1.1 Види моделювання.

1.2 Порівняльний аналіз пакетів для 3D-моделювання

Існує широкий спектр програмних пакетів, призначених для тривимірного моделювання. Їх умовно можна поділити на універсальні системи 3D-графіки (такі як Blender, 3ds Max, Maya), CAD/CAM-системи з упором на параметричне моделювання (Fusion 360, SolidWorks та ін.), вузькоспеціалізовані програми для цифрового ліплення (ZBrush, Mudbox) і процедурного моделювання (SideFX Houdini), а також новітні гібридні інструменти, що поєднують переваги різних підходів (наприклад, Plasticity, який комбінує NURBS і полігональне моделювання). Розглянемо характеристики та можливості найпопулярніших з них, приділивши особливу увагу таким, котрі згадані в темі дослідження.

Blender вільно поширюваний пакет з відкритим вихідним кодом для 3D-моделювання, анімації, рендерингу та композитингу. Він працює на різних платформах (Windows, macOS, Linux тощо) і розповсюджується під ліцензією GNU GPL, що робить його безкоштовним для будь-якого використання. Blender надає інструменти для повного циклу створення 3D-графіки: полігональне і поверхневе моделювання, скульптинг, UV-розгортка, текстурування, шейдинг, анімація, симуляція фізики, рендеринг (вбудовані рушії Eevee і Cycles) та відеомонтаж. Blender був успішно застосований при створенні анімаційних фільмів (наприклад, короткометражний фільм Flow, що отримав премію «Оскар» у 2024 р.) і впроваджується в освітніх програмах провідних університетів [14]. Основні переваги Blender – безкоштовність, крос-платформність, і велика користувацька спільнота, що забезпечує наявність навчальних матеріалів [15]. Відносним недоліком у минулому була менш зручна, ніж у комерційних аналогів, реалізація деяких задач (приміром, складні NURBS-моделі або високополігональний скульптинг), однак з версії 2.8 інтерфейс і можливості Blender істотно покращилися, і нині його розглядають як повноцінну альтернативу платним пакетам в індустрії розробки ігор.

Autodesk 3ds Max – професійний пакет для 3D-моделювання, рендерингу та анімації, орієнтований на платформу Windows. 3ds Max розробляється компанією Autodesk з 1990-х років і є одним із стандартів індустрії в галузі створення ігрової графіки, спецефектів та архітектурної візуалізації. Програма має потужні засоби полігонального і поверхневого моделювання, модуль інструментів для скульптингу та малювання прямо на моделі, а також багату систему плагінів, яка дозволяє розширювати функціональність під конкретні потреби [16]. Ліцензійна модель 3ds Max – комерційна: програму можна використовувати на основі підписки (щорічна або щомісячна оплата), надається безкоштовна пробна версія, а також існує спеціальна Indie-ліцензія за зниженим тарифом для малих розробників з доходом менше $100 тис. на рік. Сильні сторони 3ds Max: зрілий інтерфейс і робочий процес для полігонального моделювання, широке застосування у студіях, великі можливості з візуалізації (вбудований рендер Arnold, підтримка шейдерів, освітлення та ін.). Недоліки – висока вартість ліцензії, офіційна підтримка лише ОС Windows (що обмежує користувачів Mac/Linux), а також досить високі системні вимоги при роботі з важкими сценами**.**

Autodesk Maya – ще один продукт Autodesk, призначений для 3D-моделювання, анімації та візуальних ефектів. Від 3ds Max ця програма відрізняється кращою крос-платформністю (працює на Windows, Linux і macOS) [17]. За функціоналом у сфері моделювання Maya багато в чому подібна до 3ds Max: підтримує полігональне, NURBS і Subdivision-моделювання, має засоби скульптингу, а також розширюється плагінами. Ліцензійна модель Maya також передбачає підписку; існує Maya Indie для малих розробників, що надає повний функціонал за умови невеликого річного доходу. Проте обидва пакети є взаємозамінними до певної міри. Переваги Maya: потужні засоби анімації, підтримка багатьох ОС. Недоліки: висока ціна, складність для початківців –

опанування Maya вимагає часу, оскільки програма пропонує дуже багато функцій і налаштувань.

Autodesk Fusion 360 – хмарна CAD/CAE/CAM платформа, орієнтована на промислове та інженерне 3D-моделювання. Проте Fusion 360 заслуговує уваги як сучасний інструмент параметричного моделювання із простим інтерфейсом і доступною ліцензійною політикою [18]. Програма працює на Windows і macOS, зберігаючи дані моделей у хмарному сховищі Autodesk. Вона підтримує твердотільне моделювання (solid modeling) – побудову об'єктів шляхом операцій з базовими об'ємними примітивами та ескізами, що задають профілі і скетчі. Важлива особливість – наявність безкоштовної версії для персонального використання, що надається хобістам і стартапам з некомерційними цілями за умови доходу менше $1000 на рік. Ця безкоштовна версія дещо обмежена (наприклад, урізані можливості командної роботи, менше форматів експорту), але для більшості індивідуальних проєктів її достатньо. Наприклад, при розробці предметів ігрового оточення, Fusion 360 можна використати для моделювання складних механізмів, техніки, зброї. (Рис. 1.2)

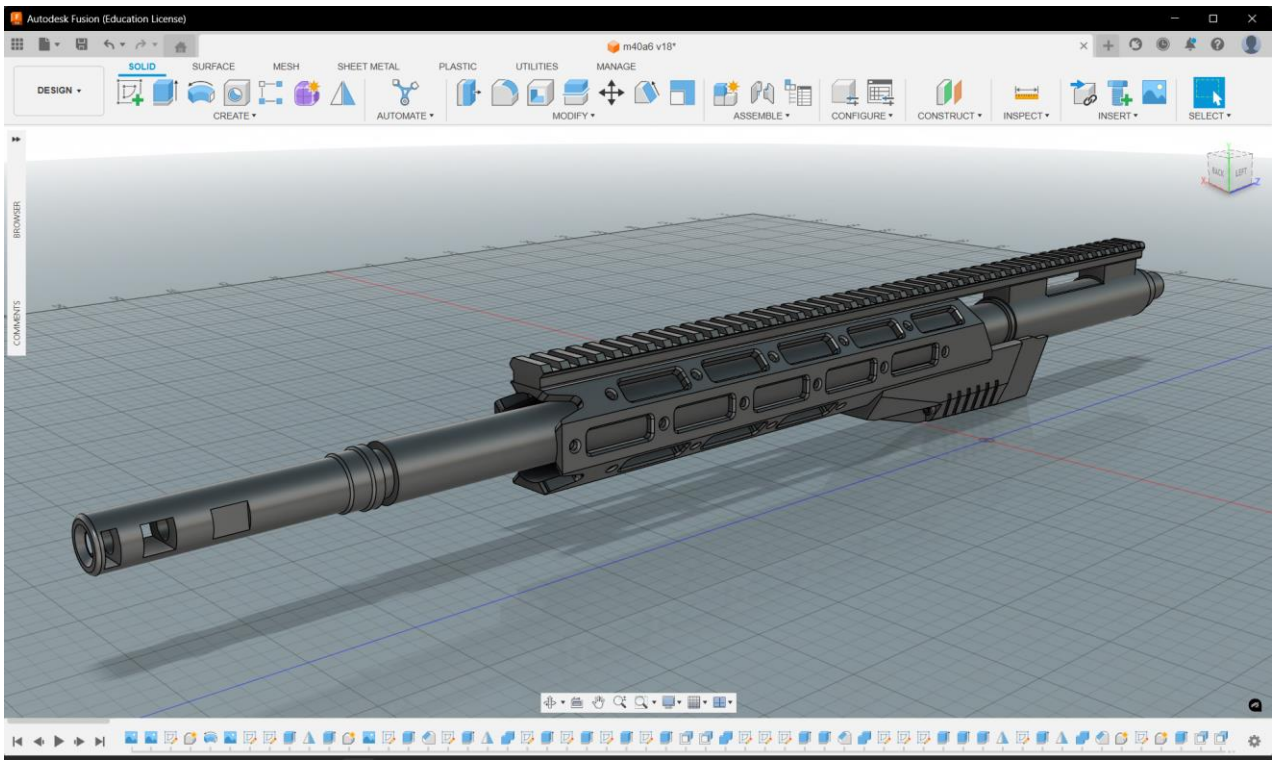


Рис. 1.2 Модель, виконана параметричними засобами моделювання Fusion 360.

Отримані CAD-моделі можна експортувати у полігональні формати (OBJ, FBX) для подальшого використання у грі. Переваги Fusion 360: параметричність, інтегрована єдина платформа для моделювання та інженерного аналізу, безкоштовність для любителів та студентів. Недоліки: вимога постійного підключення до Інтернету для роботи (хмарна архітектура), не такий широкий набір інструментів полігонального моделювання та текстурування, як у Blender/Max (Fusion більше про CAD, ніж про художню деталізацію), а також обмеження безкоштовної версії для великого бізнесу.

SideFX Houdini – високопрофесійне програмне забезпечення для процедурного 3D-моделювання, генерації візуальних ефектів (VFX) та анімації. Houdini вирізняється серед інших пакетів своїм вузловим підходом: практично будь-яка сцена або модель створюється як мережа вузлів, кожен з яких виконує окрему операцію (деформує геометрію, застосовує булеву операцію, розміщує об’єкти за алгоритмом тощо). За допомогою спеціального модуля Houdini Engine результати, створені в Houdini, можна імпортувати динамічно в інші програми, зокрема в ігрові рушії Unity та Unreal – тобто художник будує “цифровий актив” у вигляді вузлової системи, а потім рівень або гра можуть генерувати різні варіації цього активу на основі заданих параметрів. Houdini є комерційним програмним продуктом з досить високою вартістю повної ліцензії (Houdini FX). Втім, компанія SideFX пропонує доступніші варіанти: Houdini Indie – ліцензія для інді-розробників і малого бізнесу (умовно, якщо дохід < $100 тис. на рік), що надає повний функціонал Houdini FX за значно нижчою ціною, а також Houdini Apprentice – безкоштовна версія для навчання і некомерційних проєктів, яка містить майже всі можливості Houdini (обмеження стосуються лише комерційного використання і форматів експорту). Houdini працює на Windows, Linux, macOS. Основні переваги: безпрецедентні можливості процедурного моделювання і симуляції фізики, що дозволяють створювати те, що вручну було

б неможливо чи надто довго; можливість глибокої автоматизації і повторного використання розроблених процедур; гнучкість у внесенні змін на будь-якому етапі виробництва (завдяки вузловій історії побудови моделі). Недоліки: висока складність освоєння (поріг входження для художника значно вищий, ніж у традиційних DCC-пакетах), значні вимоги до ресурсів при генерації складних сцен, а також дещо менш зручне "ручне" полігональне моделювання – для простих форм традиційні інструменти можуть бути швидшими. У задачах розробки ігрового оточення Houdini раціонально застосовувати там, де потрібна генерація великої кількості контенту або складних структур (наприклад, мережа печер, процедурний міст із безліччю деталей, автогенерація різноманітних будівель із заданого набору модулів тощо). В інших випадках його можна поєднувати з класичним підходом: моделі створювати у Blender/Maya, а Houdini використовувати для окремих ефектів чи масової обробки (скажімо, розкласти по ігровій локації сотні каменів випадковим чином).

Plasticity – відносно новий 3D-редактор (реліз перших версій відбувся в 2022–2023 роках), який позиціонується як "CAD для художників" [19]. Plasticity намагається об'єднати найкраще з двох світів: точність і можливості NURBS-твердотільного моделювання (як у CAD) та гнучкість і швидкість полігонального workflow, до якого звикли 3D-художники. Програма дозволяє створювати і редагувати твердотільні об'єкти та має зручний інтерфейс: наприклад, філлет та скруглення ребер (fillets/chamfers) робляться інтерактивно й легко, а булеві операції виконуються швидко.

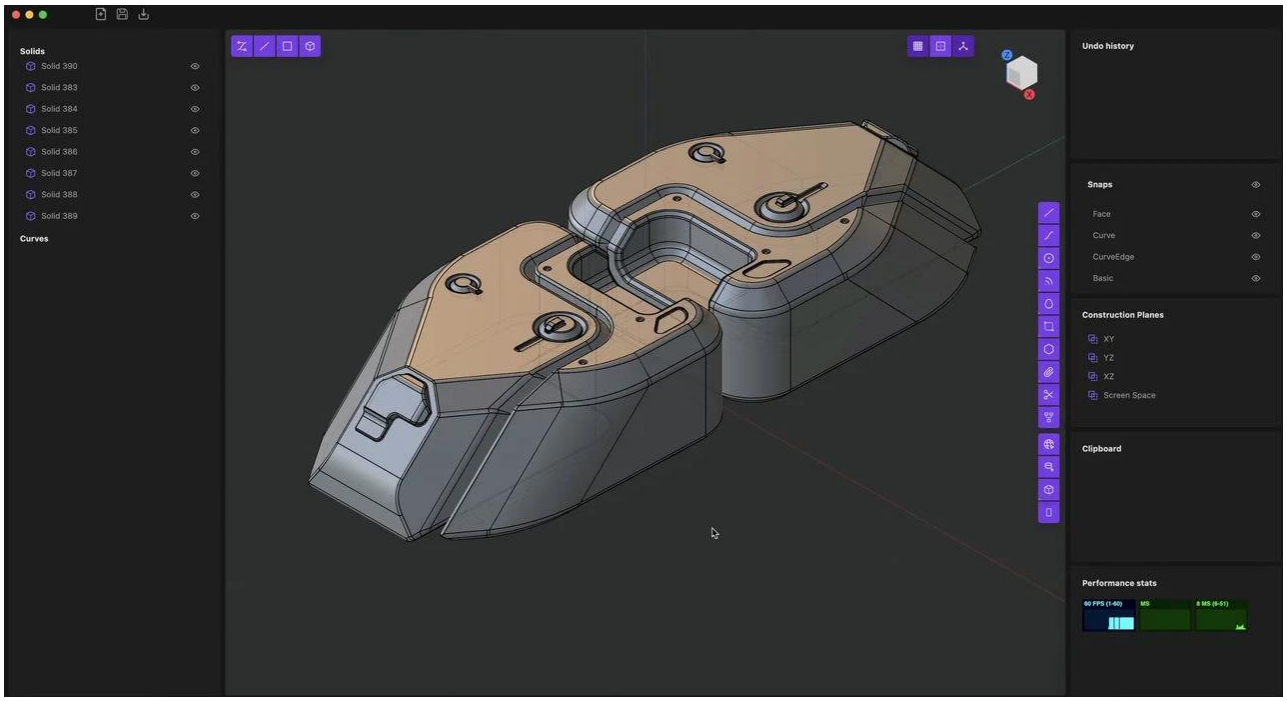


Рис. 1.2 Модель, виконана параметричними засобами моделювання Plasticity.

Plasticity використовує ядро Parasolid – геометричний рушій, що лежить в основі SolidWorks, Siemens NX та інших промислових CAD-систем. Водночас інтерфейс Plasticity наближений до інтерфейсу Blender: навігація, гарячі клавіші, взаємодія з об’єктами розраховані на дизайнерів, а не на інженерів, що значно спрощує опанування програми тими, хто мав досвід полігонального моделювання. Plasticity підтримує Windows, macOS і Linux, розповсюджується на умовах разової покупки (перпетуальна ліцензія, без підписки) за відносно невисокою ціною, має 30-денний безкоштовний пробний період. Переваги Plasticity: поєднання точності CAD та зручності DCC, швидкі булеві операції і філлети (Fillet), відсутність підписки (користувач фактично “володіє” придбаним ПЗ). Недоліки: програма ще молода, тому деякі функції наразі є неповними. Менша спільнота і менше ресурсів підтримки, відсутність засобів анімації чи рендерингу (Plasticity фокусується виключно на моделюванні). Основні характеристики розглянутих систем зведено в табл. 1.1, де зазначено їх тип ліцензії, платформені особливості та сфери застосування.

Таблиця 1.1

Порівняння програмних пакетів для 3D-моделювання

Програма	Тип моделювання	Ліцензія	Переваги	Недоліки
Blender	Полігональне, процедурне, цифрове ліплення	Безкоштовна, Open Source	Безкоштовний, багатофункціональний, велика спільнота	Складний інтерфейс для новачків
3ds Max	Полігональне, процедурне	Платна (підписка)	Широке використання у геймдеві, потужний інструментарій	Лише для Windows, висока ціна
Autodesk Maya	Полігональне, процедурне, цифрове ліплення	Платна (підписка)	Потужні інструменти анімації, стандарт у кіноіндустрії	Високий поріг входження, висока вартість
Fusion 360	Параметричне, CAD	Платна (підписка)	Чудово підходить для технічного моделювання, хмарні функції	Слабка підтримка полігонального моделювання
Houdini	Процедурне, полігональне	Безкоштовна (Apprentice), платна	Потужні процедурні інструменти, ідеальна для VFX	Високий рівень складності
Plasticity	Параметричне (NURBS)	Платна	Інтуїтивно зрозумілий для CAD/NURBS, швидка робота	Обмежена підтримка полігонального моделювання

1.3 Інструменти для текстурування 3D-моделей

Щоб об’єкт виглядав правдоподібно в грі, для нього необхідно створити текстури. Текстурування – процес нанесення зображень (текстурних карт) на поверхню 3D-моделі з метою визначити її колір, рельєф, рефлективні властивості та інші візуальні характеристики. Сучасний підхід до текстурування ігор базується на фізично коректних матеріалах (PBR – Physically Based Rendering). Для створення таких текстур застосовуються спеціалізовані програми, що дозволяють малювати безпосередньо по 3D-моделі, бачачи результат у режимі реального часу. Найбільш відомі інструменти для текстурування – Adobe Substance 3D Painter, 3DCoat (модуль 3DCoat Textura) та Marmoset Toolbag. Розглянемо їх стисло.

Adobe Substance 3D Painter стандарт індустрії 3D для професійного текстурування моделей. Цей застосунок дозволяє в режимі реального часу фарбувати модель, накладаючи шари матеріалів і масок [20]. Painter підтримує PBR-матеріали та працює з UV-розгорткою моделі: художник може завантажити модель (формати OBJ, FBX тощо), обрати готові матеріали з бібліотеки або створити власні та наносити їх пензлями або через процедурні маски. Ключова особливість – проекційне малювання по поверхні, коли користувач наносить штрихи ніби по тривимірному полотну, а програма автоматично розподіляє фарбу по відповідних текстурних координатах. Ліцензійно Substance Painter доступний через підписку Adobe (пакет Substance 3D Collection) або як окремий продукт, також є варіант придбання через Steam для індивідуальних користувачів [21]. Переваги: інтуїтивність, фізично коректні матеріали, велика спільнота і своя бібліотека матеріалів. Недоліки: відсутня повнофункціональна безкоштовна версія, лише платна або навчальна ліцензія для закладів освіти, доволі високі вимоги до пам’яті при роботі з 4K-8K текстурами.

3DCoatTextura багатофункціональне програмне забезпечення від української компанії Pilgway, яке поєднує засоби скульптингу, ретопології, UV-

розгортки і текстурування. 3DCoat позиціонується як повний цикл створення 3D-моделі з нуля: «містить всі інструменти, необхідні, щоб перетворити блок цифрової глини на готову до продакшну, повністю текстуровану органічну або твердотільну модель» [22]. Для тих, хто потребує лише малювання текстур, існує окрема версія 3DCoat Textura – в ній залишені тільки модулі UV та малювання, без скульптингу і ретопології, що здешевлює продукт. 3DCoat підтримує живопис PBR-матеріалами, схожий за підходом на Substance Painter: шари, маски, Smart Materials, режим перегляду під різні рушії. Його вирізняє надзвичайно зручний інструмент для ручного малювання швів UV-розгортки і можливість автоматичного розгортання з мінімальними відхиленнями, що економить час перед текстуруванням. Ліцензійна політика 3DCoat гнучка: можна купити постійну ліцензію (з безкоштовним оновленням впродовж першого року підписки, далі – платні оновлення за бажанням) або оформити передплату; для навчання й використання у некомерційних цілях передбачена безкоштовна версія 3DCoatPrint. Переваги: 3DCoat дозволяє на одному етапі і розгорнути, і розмалювати модель, має потужні засоби ручного доведення дрібних деталей пензлями. Крім того, 3DCoat підтримує технологію воксельного скульптингу (модель як масив вокселів), що спрощує створення і текстурування дуже складних рельєфів. Недоліки: інтерфейс дещо переобтяжений (через різноманітність модулів), менша кількість готових матеріалів у відкритому доступі (у порівнянні з Substance) – хоча розробники надають безкоштовну бібліотеку з 500+ PBR-матеріалів [23]. В цілому 3DCoat є гарною альтернативою для інді-розробників, особливо враховуючи лояльні умови для невеликих студій і постійну підтримку розробників (оновлення виходять регулярно, програма доступна 70 мовами світу, включно з українською).

Marmoset Toolbag універсальний інструмент “все в одному” для запікання текстурних карт, текстурування та рендерингу, популярний серед 3D-художників для створення портфоліо і презентації моделей [24]. Marmoset

пропонує тестовий безкоштовний період, а для студентів діють знижки. Переваги: простота отримання швидкого результату, доступний рендер у часі з високою якістю (підтримується трасування променів – ray tracing – для фотореалістичності), інтегрований workflow (bake → texture → render). Недоліки: менше інструментів малювання, порівняно з Substance, відсутність версії під Linux.

Порівняння основних властивостей цих програм представлено у табл. 1.2.

Таблиця 1.2

Порівняння програм для текстурування 3D-моделей

Програма	Платформа	Ліцензія	Переваги	Недоліки
Substance Painter	Windows, macOS	Платна (підписка)	Широкі можливості, стандарт у галузі	Висока ціна, складність для новачків
3DCoat	Windows, Linux, macOS	Платна	Простий інтерфейс, підтримка скульптингу	Менш поширена серед студій
Marmoset Toolbag	Windows, macOS	Платна	Висока якість візуалізації, простота у використанні	Обмежені можливості порівняно з Substance Painter

1.4 Ігрові рушії: вимоги та можливості

Після створення 3D-моделей та текстур наступним кроком є їх інтеграція в ігровий рушій – спеціальне програмне середовище, на базі якого розробляються інтерактивні програми та відеоігри. Ігровий рушій забезпечує відтворення тривимірної графіки у реальному часі, фізичну симуляцію, обробку

вводу гравця, звук тощо. Приклади сучасних рушіїв – Unity, Unreal Engine, CryEngine, Godot та ін. Розглянемо найбільш потужні з них, що застосовуються в комерційній розробці масштабних проєктів.

Unity популярний рушій від компанії Unity Technologies, відомий своєю зручністю для інді-розробників та крос-платформністю [25]. Unity дозволяє експортувати ігри на десятки платформ (Windows, Android, iOS, консолі, веб тощо) і має вбудовану середу розробки з підтримкою мови C# для написання ігрової логіки. В контексті 3D-графіки Unity пропонує два рендерингові рішення: Built-in Render Pipeline (традиційний) [26] та Scriptable Render Pipelines – Universal Render Pipeline (URP) і High Definition Render Pipeline (HDRP) [27]. URP оптимізований для середнього рівня якості та мобільних пристроїв, HDRP – для найвищої якості на потужних ПК/консолях (фізично коректне освітлення, ефекти глобального освітлення тощо). Вимоги до моделей в Unity стандартні для реального часу: кількість полігонів має бути зваженою, текстури – оптимізованими за роздільністю, бажано використовувати LOD (рівні деталізації). Unity надає зручний імпорт ресурсів із форматів FBX, OBJ та ін., підтримує анімацію (Mecanim анімаційна система) і має розвинену систему матеріалів (Shader Graph для створення шейдерів). Ліцензія Unity умовно безкоштовна: версія Unity Personal надається без оплати всім розробникам, чий річний дохід від проєктів на Unity не перевищує $200 тис.

При перевищенні порогу необхідно придбати Unity Pro (платна передплата), також існують спеціальні умови для навчальних закладів та великих підприємств.

Unity відомий невибагливістю до навичок розробника – поріг входження досить низький, тому цей рушій часто обирають для невеликих ігор, мобільних додатків та VR/AR проєктів, проте на ньому створено і чимало AAA-ігор. У нашій роботі, для інтеграції моделей в рушій, найпростіше обрати Unity як один з варіантів, зважаючи на його популярність та доступність.

Unreal Engine високопродуктивний рушій від Epic Games, який знаменитий фотореалістичною графікою та використанням у численних ААА-проєктах. Актуальна версія Unreal Engine 5 принесла технології Nanite (віртуалізована геометрія, що дозволяє рендерити мільйони трикутників, автоматично управляючи рівнями деталізації) та Lumen (динамічне глобальне освітлення), що значно спрощує вимоги до моделей – наприклад, Nanite дає змогу імпортувати в рушій навіть дуже високополігональні об’єкти без ручного створення LOD-сіток [28]. Проте для звичайних моделей також діють типові принципи оптимізації. Unreal Engine використовує мову C++ і візуальну скриптову систему Blueprints для ігрової логіки. Рушій безкоштовний у використанні для розробників, проте застосовується роялті-модель ліцензування: якщо гра випущена комерційно, то після перших $1 млн доходу розробник сплачує Epic Games 5% від подальших прибутків. Ця умова стосується ігор; для неігрових застосунків (візуалізації, архітектура) з 2024 р. запроваджено альтернативну схему – підписку для великих компаній, але для нас актуальна саме ігрова модель. Таким чином, незалежно від кількості членів команди, за використання Unreal не потрібно платити нічого, доки гра не почне приносити вагомий дохід. Unreal Engine є open-source (за ліцензією EULA) – його повний вихідний код доступний на GitHub для ліцензійних користувачів. У аспекті роботи з графічним контентом Unreal надає розвинуті інструменти імпорту (Datasmith для CAD-моделей, прямий імпорт з DCC через FBX/OBJ), власний редактор матеріалів і шейдерів, можливості створення ландшафтів, систем часток (Niagara) тощо. Для наших цілей Unreal Engine може бути використаний для фінальної презентації моделей у вигляді сцени з реалістичним освітленням.

CryEngine рушій, розроблений німецькою компанією Crytek, який став відомим завдяки грі Crysis та іншим проєктам з акцентом на передову графіку [29]. CryEngine славився чудовою якістю зображення і реалістичною поведінкою

освітлення та матеріалів, однак трохи втратив позиції останніми роками через конкуренцію Unity та Unreal. Нині CryEngine доступний розробникам за моделлю, подібною до Unreal: він є безкоштовним у завантаженні, але вимагає виплати роялті 5% з комерційних проєктів (при цьому перші $5 тис. на рік з кожної гри звільняються від роялті). CryEngine надає сильний графічний інструментарій, зокрема власний фотореалістичний рендерер, потужну систему побудови ландшафтів та рослинності (Voxel-based Terrain, Vegetation system), інтегровану фізику та розвинену систему штучного інтелекту. Сценарії в CryEngine можна писати на C++, а в новіших версіях додано підтримку C# для спрощення розробки. З точки зору роботи з моделями: історично CryEngine вимагав дуже оптимізованих ресурсів (відомий мем «чи запуститься Crysis?» саме про колишню вибагливість рушія), але сучасні версії здатні обробляти великі обсяги геометрії, хоча й не мають аналогів Nanite. CryEngine використовується менше спільнотою, проте його обирають деякі студії для шутерів, екшенів завдяки прекрасній якості картинки «з коробки». Для нашого проєкту достатньо усвідомити, що моделі, підготовлені за стандартами PBR для Unreal/Unity, будуть сумісні і з CryEngine – він також оперує базовими форматами FBX/OBJ і підтримує PBR-матеріали.

Наведемо коротке порівняння ключових аспектів трьох ігрових рушіїв у табл. 1.3.

Таблиця 1.3.

Порівняння популярних ігрових рушіїв

Ігровий рушій	Мови програмування	Платформа	Ліцензія	Переваги	Недоліки
Unity	C#	Windows, Linux, macOS, мобільні платформи	Безкоштовна, платна	Простота використання, велика спільнота, популярність	Менше графічних можливостей, ніж UE

Unreal Engine	C++, Blueprints	Windows, Linux, macOS, мобільні платформи	Безкоштовна (з роялті)	Потужна графіка, відкритий код, Blueprints	Складніший для новачків, висока вимогливість до ресурсів
CryEngine	C++, Flowgraph	Windows	Безкоштовна (з роялті)	Високоякісна графіка, потужні можливості для відкритих світів	Менше навчальних матеріалів, складний інтерфейс

Для 3D-моделей, що створюються у рамках нашої роботи, критично забезпечити їх сумісність із обраним рушієм. Як правило, це означає: експорт у формат FBX або OBJ зі збереженням ієрархії та нормалей, наявність розгортання UV без накладень (для уникнення артефактів при освітленні і текстуруванні), дотримання реальних розмірів (метричних) та масштабування, використання текстур у форматах, які підтримує рушій (PNG, JPEG, TGA для Unity/Unreal). Також бажано генерувати карти нормалей, ambient occlusion та інші допоміжні карти заздалегідь (наприклад, у Substance або Marmoset), щоб у рушії залишилося тільки налаштувати матеріали, а не виконувати важкі обчислення.

1.5 Вибір основних програмних інструментів

На основі результатів порівняльного аналізу сучасних програм для 3D-моделювання, текстурування та створення ігрових проєктів, та, враховуючи, що кожний з розглянутих програмних пакетів має сильні та слабкі сторони, було обрано комбінований підхід до вибору програмного забезпечення, при якому кожна програма відповідає за виконання певної спеціалізованої задачі.

Для створення 3D-моделей в рамках цього проєкту було обрано Fusion 360. Ця програма використовує параметричні операції та дає можливість досягти

високої точності, забезпечує структурну цілісність та дозволяє контролювати рівень деталізації моделей. Ще однією перевагою є наявність зручної системи управління історією створення моделі та можливість оперативного внесення будь-яких змін. Ці можливості сприяють досягненню поставлених задач на етапах концептуального та детального проєктування моделей.

Після завершення початкового моделювання, наступним кроком є підготовка моделей для текстурування та інтеграції в ігровий рушій. Для цього ми використаємо Blender. В ньому виконаємо оптимізацію високолігональних та низькополігональних моделей, виявимо та виправимо можливі помилки топології моделей, підготуємо їх до текстурвання, а також триангулюємо сітку.

Для створення текстур використаємо Adobe Substance 3D Painter. Це дозволить нам створити реалістичні та високодеталізовані текстури за допомогою шарів, масок, смарт-матеріалів та пензлів. Також тут присутні пресети експорту до різних ігрових рушіїв та стандартів рендерінгу. Що значно спрощує та прискорює процес інтеграції текстур до ігрового рушія.

Unity було обрано як платформу для подальшого тестування та візуалізації створених моделей. Цей ігровий рушій пропонує широкі можливості для інтеграції 3D-асетів, включаючи підтримку фізики, анімації, реалістичного освітлення та PBR-матеріалів. Також Unity широко застосовується для навчання та незалежними студіями розробки, що робить його оптимальним рішенням для реалізації навчальних проектів.

Висновки до розділу 1

Перший розділ роботи був присвячений розбору теоретичних та практичних засад тривимірного моделювання. У ньому розкрито сутність та класифіковано основні методи, такі як полігональний, параметричний, цифрове ліплення та процедурний. Встановлено, що кожен метод має свої переваги та

сфери застосування, хоча, в реальних проектах часто використовуються комбінації різних методів послідовно. Було проведено порівняльний аналіз шести 3D-пакетів (Blender, 3ds Max, Maya, Fusion 360, Houdini, Plasticity) за такими параметрами як ліцензійний доступ, операційна система, ключові функції, сильні та слабкі сторони. Враховуючи результати аналізу, визначено доцільність використання Blender як основного інструменту в цьому проекті, враховуючи його універсальність. Проведено аналіз інструментів для текстурування 3D-моделей. Substance Painter, 3DCoat та Marmoset Toolbag, та їхню інтеграцію в процес розробки ігрових ресурсів. Фізично коректне текстурування (PBR) є стандартом де-факто, а зазначені програми забезпечують ефективне створення необхідних карт матеріалів. Для виконання поставлених задач обрано підхід, де для реалізації складних матеріалів – застосовується Adobe Substance 3D Painter. Досліджено вимоги та особливості ігрових рушіїв Unity, Unreal Engine та CryEngine щодо імпорту та рендерингу 3D-моделей. Проаналізовано ліцензійні угоди, а також сумісні графічні технології разом з їх впливом на методи моделювання, особливо з урахуванням впровадження технології Nanite в Unreal Engine 5. Цей аналіз дозволив зрозуміти, як підготувати моделі для коректної роботи в кожному з рушіїв (формати, кількість полігонів, масштабування, текстури).

Вибір програмного забезпечення для виконання практичної частини другого розділу кваліфікаційної роботи було затверджено наступним чином: Fusion 360 буде використано для моделювання з максимальною деталізацією, Blender - для оптимізації та підготовки моделей до UV-розгортання та текстурування, Substance 3d Painter - для створення текстур високої якості. Цей вибір співпадає з сучасними тенденціями та є доступним в контексті виконання кваліфікаційної роботи, оскільки буде використана безкоштовна версія Fusion 360 для студентів та підписка на програмні продукти Adobe, що включає Adobe Substance 3D Painter, Adobe Photoshop та інші. Здобуті в цьому розділі висновки

стануть фундаментом для наступного етапу роботи – безпосереднього створення набору 3D-моделей для ігрового оточення. В результаті було сформовано чітке розуміння послідовності дій (pipeline) та інструментів, що будуть використані.

РОЗДІЛ 2
РОЗРОБКА ТА ІНТЕГРАЦІЯ 3D-МОДЕЛЕЙ ДЛЯ КОМП'ЮТЕРНОЇ ГРИ

2.1 Підготовка об'єктів для комп'ютерної гри

Перший етап – це концептуалізація. Має бути знайдено відповідні референсні зображення, на основі яких буде створено набір моделей.

Другий етап – моделювання. Спочатку створюється параметрична модель у програмі Fusion 360. Далі вона експортується як полігональна, через MoI3D. Після цього, в Blender, створюються дві версії моделі - низькополігональна та високополігональна. Високолігональна буде використана для запікання карти нормалей.

На третьому етапі створюються текстури для об'єкта за допомогою Substance 3D Painter, що передбачає попереднє розгортання моделі в RizomUV. Цей процес включає визначення і розміщення текстурних координат на поверхні моделі, а також створення текстур для визначення кольорів, характеристик поверхні та інших атрибутів.

Четвертий етап – риггінг або створення скелетної анімації. Тут розробляється скелет для анімації моделі та його прив'язка до полігонів моделі. Даний етап буде виконано в Blender та Unity3D.

Для демонстрації загальних підходів та методик моделювання була обрана конкретна модель холодної зброї, на якій реалізовано ці методи. А саме M9 Bayonet. Варто також зазначити, що кінцевий продукт моделювання не має бути точною копією предмета референсного зображення, тому що це не є метою даного процесу; основним завданням є передача загальних форм і пропорцій при збереженні художнього та технічного підходу до моделювання.

Отже, в рамках цього проєкту предметом моделювання була обрана низка видів холодної зброї, що демонструватиме варіативність і використання різних методик.

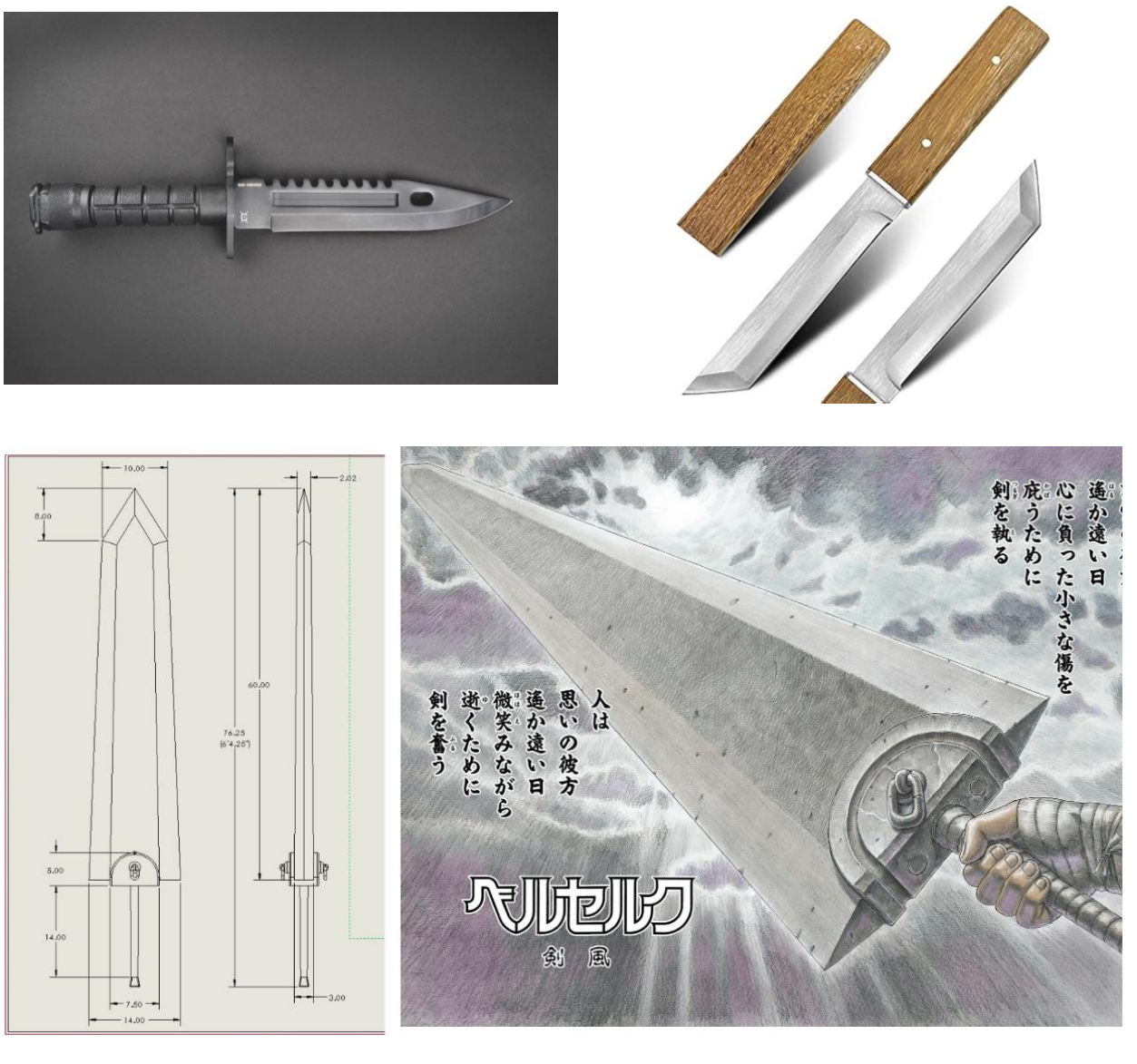


Рис 2.1 Обрані референси для моделювання

Виконуючи перший етап, переходимо до безпосереднього моделювання у середовищі **Autodesk Fusion 360**, де ми спочатку додаємо референсне зображення, яке допоможе точно дотримуватися пропорцій та форми при моделюванні. Це зображення розміщується на відповідній площині, після чого створюємо **скетч** — креслення, яке слугує основою для створення моделі (Рис. 2.2). На цьому етапі важливо дотримуватись масштабу, а також чітко визначити силует об'єкта.

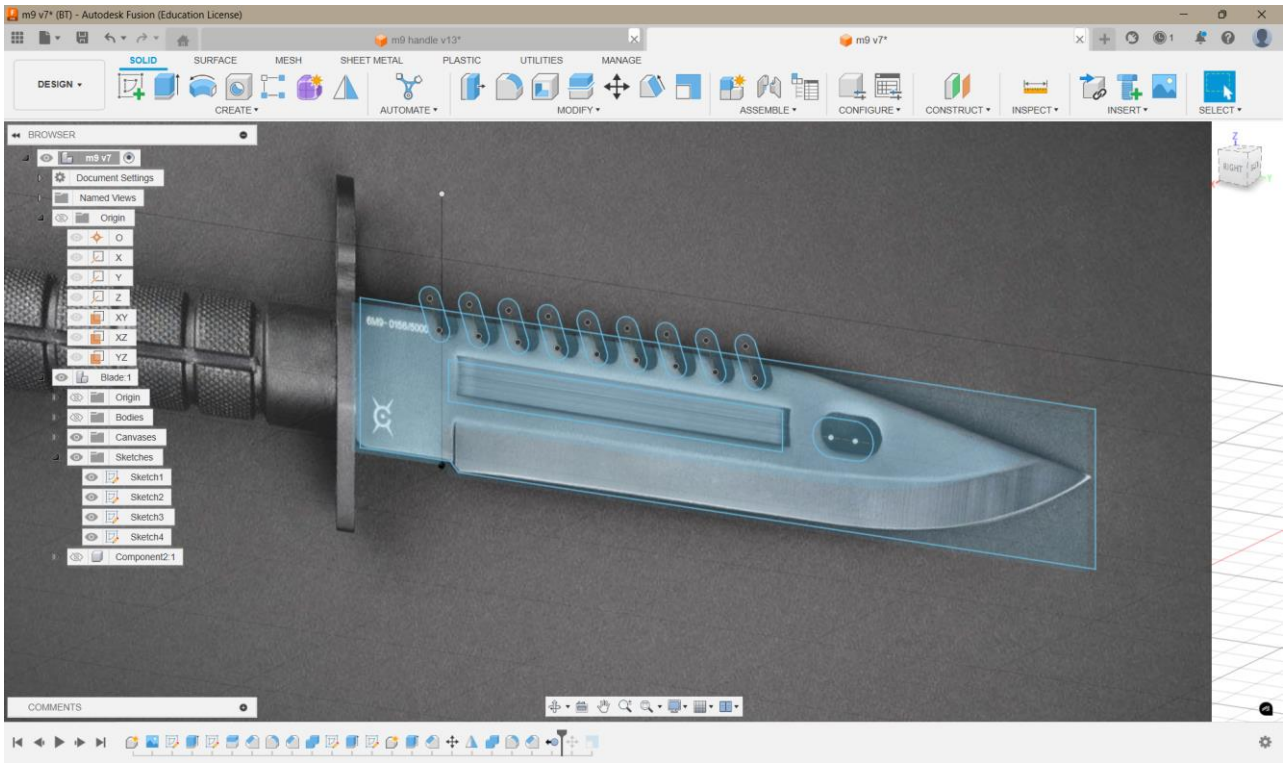
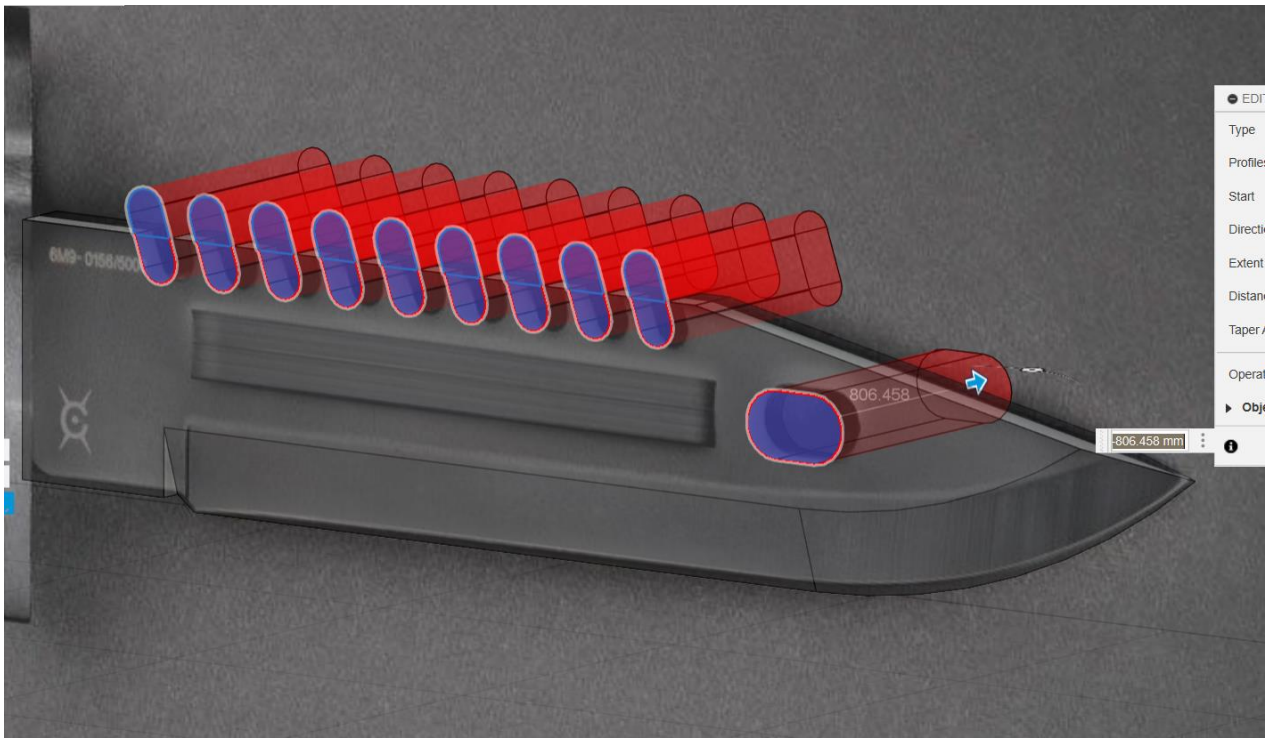
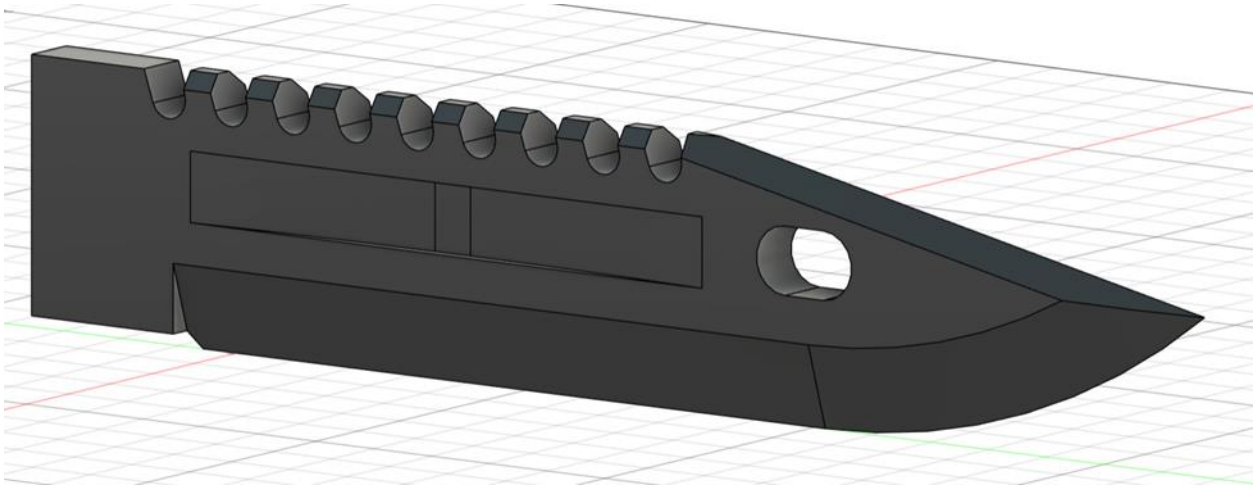


Рис 2.2 Створений скетч на фоні референсу

Після створення скетчу переходимо до використання операцій моделювання, таких як: extrude, combine, loft, fillet та chamfer (рис 2.3). Дані операції є фундаментальними при створенні моделей предметів, які в реальності були б виготовлені на станках з числовим програмним керуванням (ЧПК) [30].



а)



б)

Рис 2.3 а) виконання операції combine б) готове лезо ножа

Далі застосовуємо ті ж методи для моделювання рукояті та всіх інших елементів даного ножа (рис 2.4).

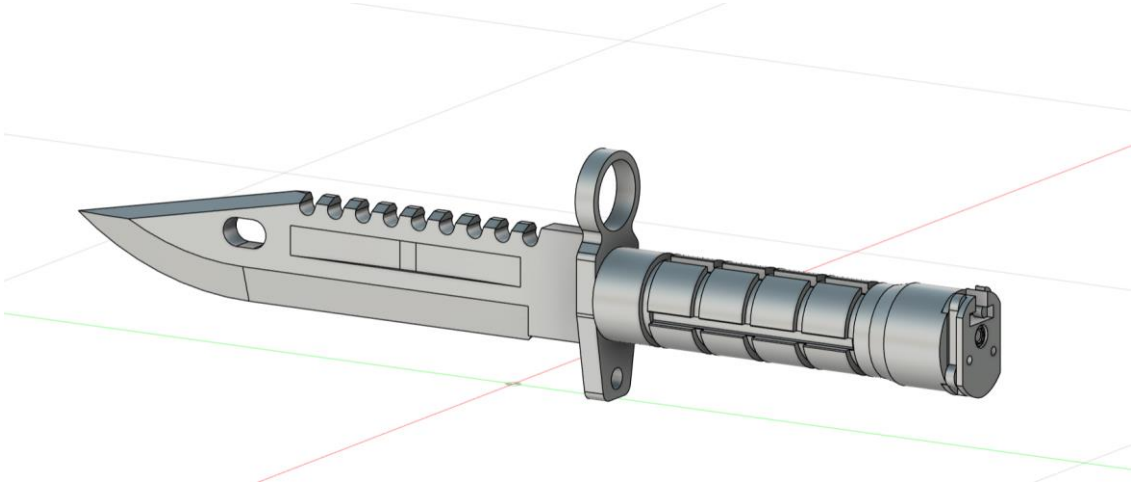
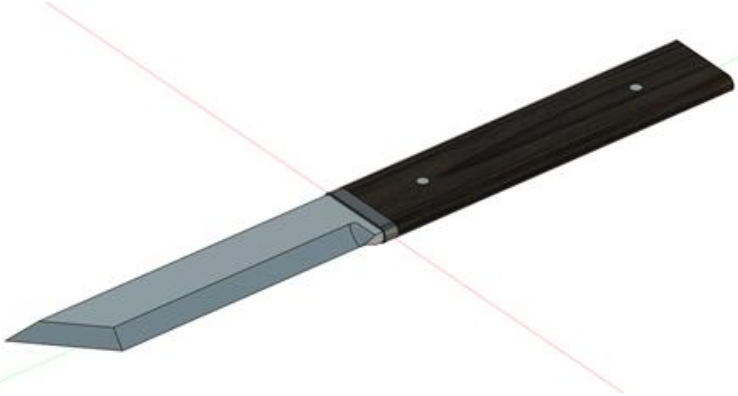
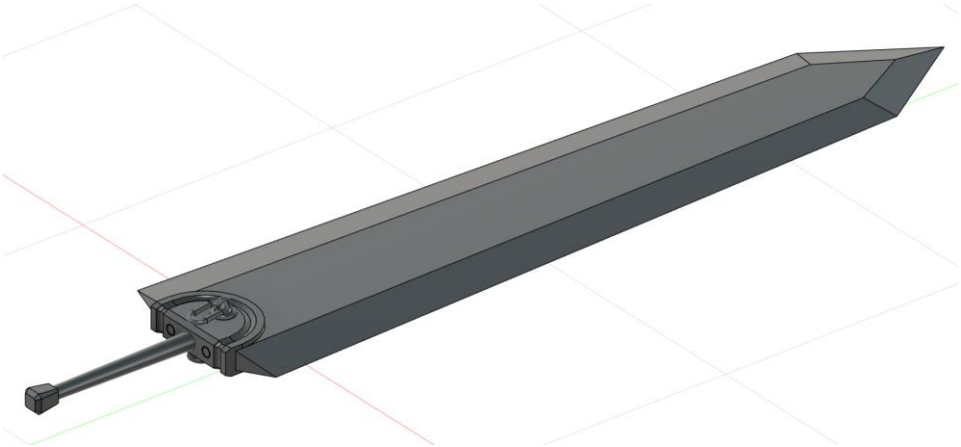


Рис 2.4. Готова модель ножа M9 Bayonet

В аналогічний спосіб створюємо моделі ножа та великого меча (рис 2.5).



а)



б)

Рис 2.5. Готові моделі: а) ножа б) великого меча

Наступним кроком експортуємо модель із **Fusion 360** до **MoI3D** у форматі **“.iges”** для подальшого налаштування якості геометрії, що дозволить більш гнучко контролювати щільність сітки при триангуляції [31]. Це є важливим підготовчим етапом перед імпортом у **Blender**, де ми будемо створювати високополігональну та низькополігональну версії моделі (рис. 2.6).

Обов’язково зберігаємо модель у двох варіантах роздільності – з високою щільністю сітки для запікання нормалей і з оптимізованою кількістю полігонів для використання у грі. Такий підхід дає змогу забезпечити якість візуалізації без надмірного навантаження на систему, особливо у реальному часі. Крім того, це полегшує подальші етапи текстурування та створення UV-розгортки, оскільки обидві версії моделі мають спільне походження і легко співставляються при запіканні карт у Substance 3D Painter або аналогічних інструментах.

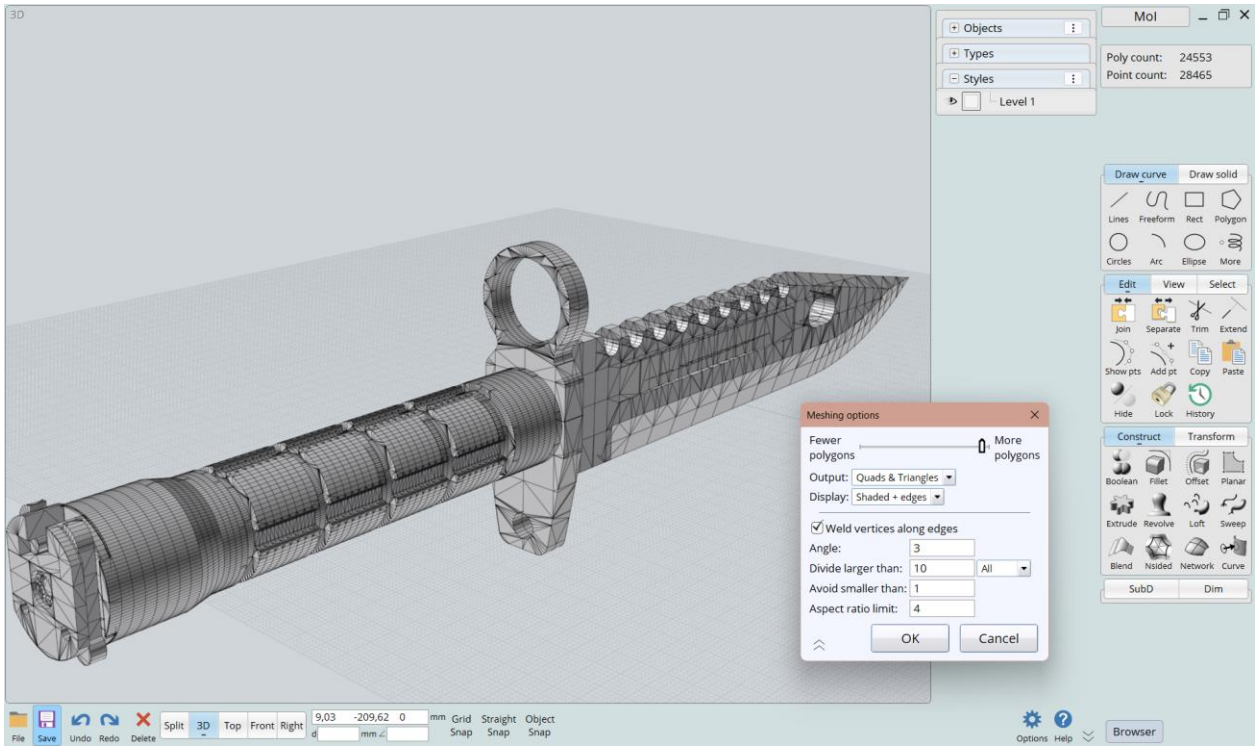


Рис 2.6. Налаштування якості моделі.

Тепер, маючи полігональне представлення моделі, можемо переходити до роботи в Blender. Імпортуємо файл у зручному для програми форматі, наприклад, **.fbx** або **.obj**, зберігаючи вихідну топологію та масштаб. Створимо відповідний стек модифікаторів для генерації високополігональної моделі (Рис. 2.7), зокрема застосуємо **Subdivision Surface** для згладжування геометрії, **Remesh** — для перебудови сітки моделі з рівномірним розподілом полігонів, та **Corrective Smooth** — для додаткового згладжування та усунення артефактів після ремешингу. За потреби також може бути використаний модифікатор **Decimate**, що дозволяє зменшити кількість полігонів, не втрачаючи при цьому загальну форму об’єкта, і покращити відгук програми на слабших системах.

Такий підхід дозволяє створити деталізовану версію моделі, яка слугуватиме джерелом при запіканні карти нормалей на оптимізований, низькополігональний варіант. Важливо слідкувати за тим, щоб усі модифікатори не призводили до деформацій або порушення первинних пропорцій, а також зберігали коректну топологію, необхідну для подальшої UV-розгортки, запікання карт та текстурування в зовнішніх програмах, таких як Substance 3D Painter.

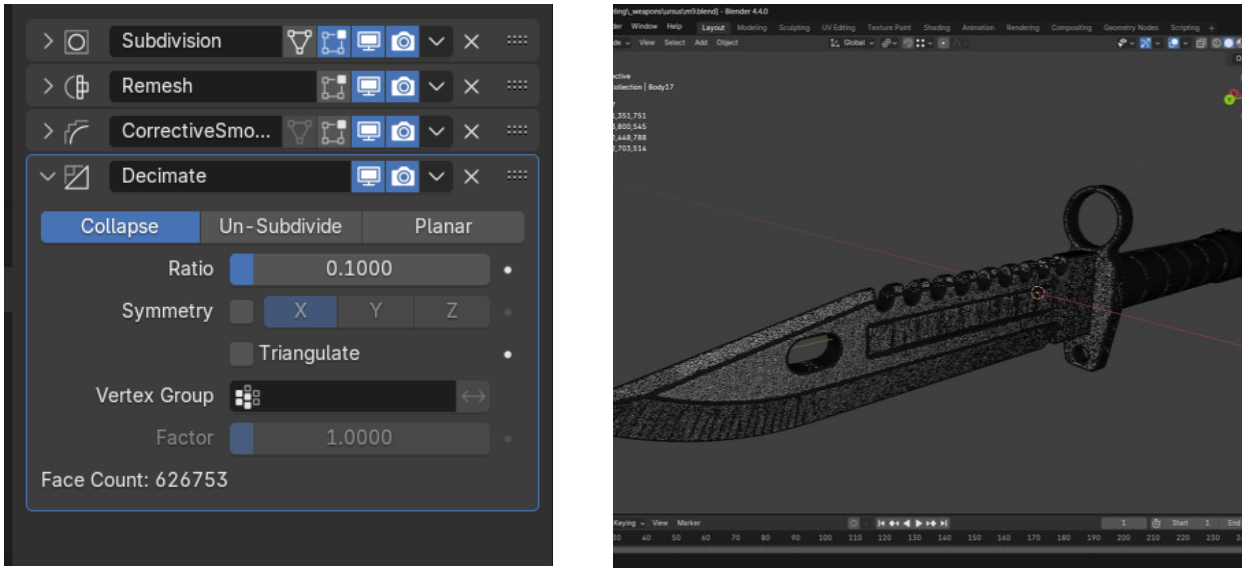


Рис 2.7. Стек модифікаторів. Готова високополігональна модель.

Далі створюємо низькополігональну модель. Для цього імпортуємо файл моделі з відповідною роздільністю, і виконуємо процес очищення та триангуляції сітки моделі (Рис 2.8).

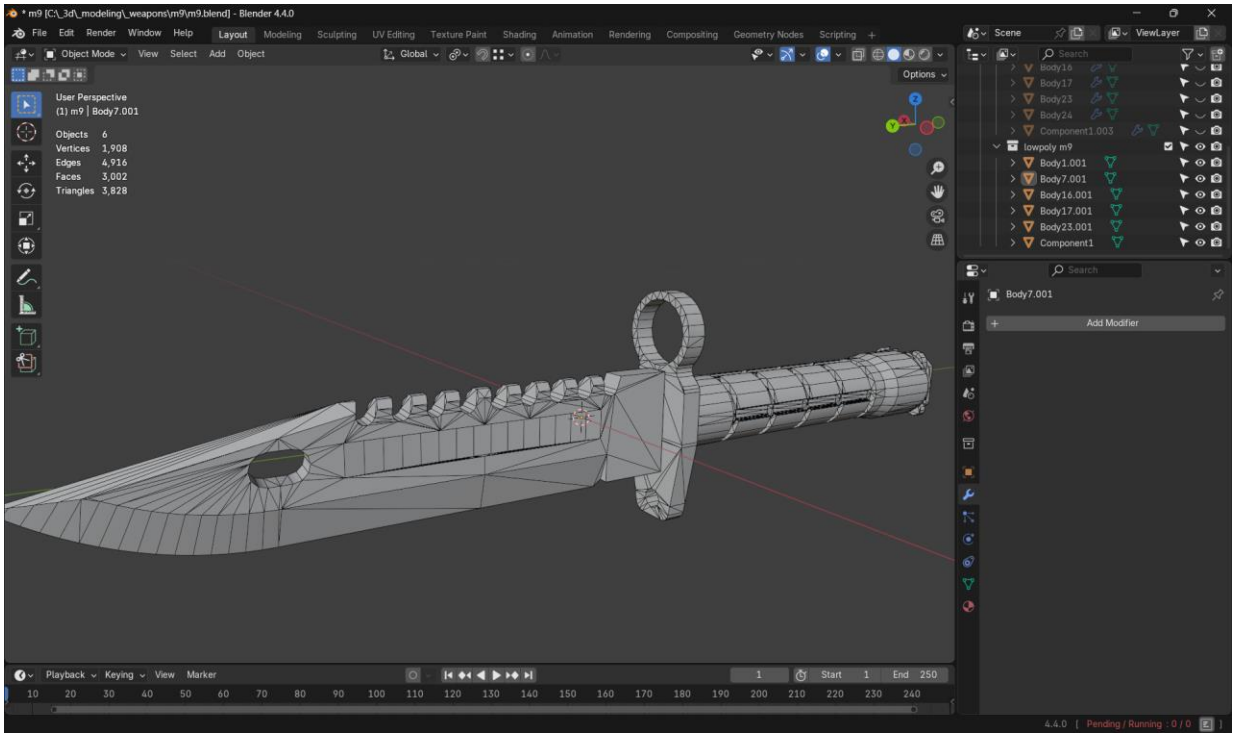


Рис 2.8. Низькополігональна модель.

Отже, у нас є все необхідне, щоб перейти до наступного етапу — текстурування. Перед цим ми створимо базову UV-розгортку в програмі **RizomUV**, яка дозволяє зручно розгортати 3D-моделі, оптимізуючи розміщення UV-островів [32]. Правильно підготовлена UV-розгортка є критично важливою для якісного запікання карт і подальшого нанесення текстур, оскільки вона визначає, як саме текстури(карти нормалей, roughness, albedo тощо) проєктуватимуться на поверхню моделі. Після експорту UV-розгортки ми зможемо перейти до етапу запікання текстур у **Substance 3D Painter** та детального текстурування моделі з урахуванням матеріалів, зношеності поверхні та художніх акцентів. (рис 2.9).

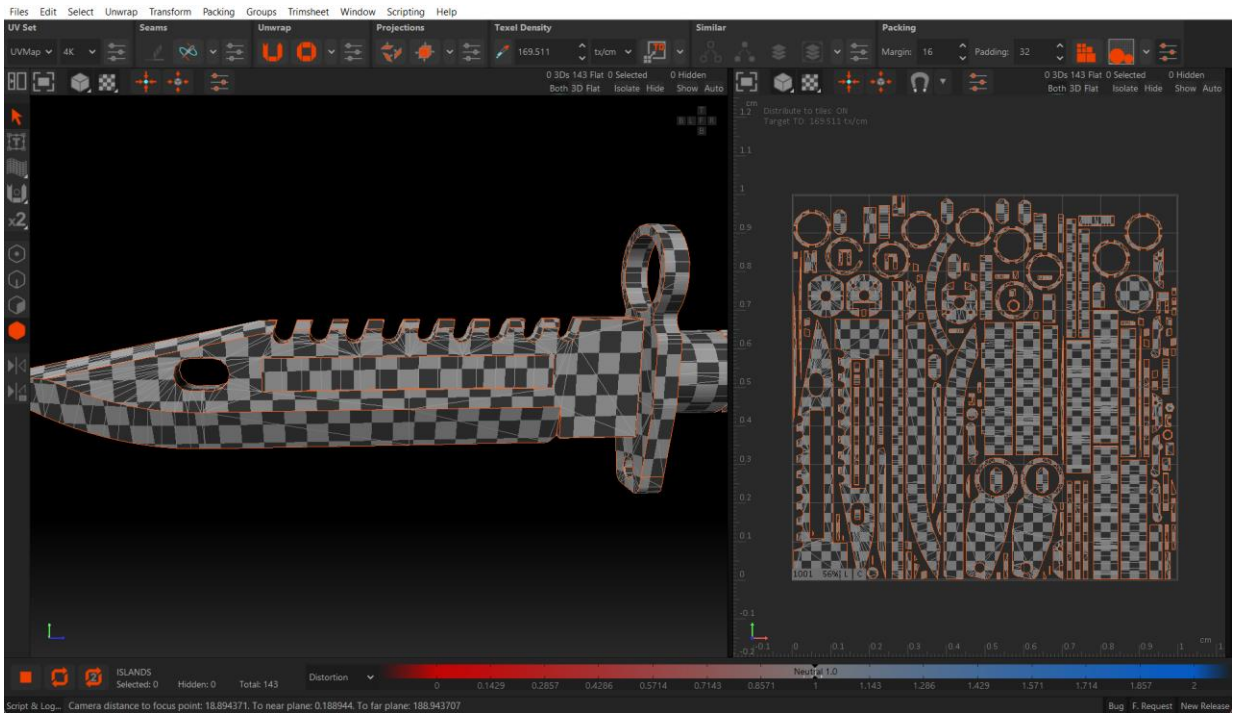


Рис 2.9. Фінальний варіант UV-розгортки в RizomUV.

Маючи готову UV-розгортку, можна перейти до етапу запікання карти нормалей та текстурування моделі. Це зумовлено тим, що саме UV-розгортка визначає, як текстури будуть проєктуватися на поверхню 3D-моделі, забезпечуючи коректне відображення деталей матеріалів і результатів запікання на її геометрії.

Імпортуємо низькополігональну модель у **Adobe Substance 3D Painter**, де розпочинаємо з етапу **запікання карт (baking)**. Переходимо в режим запікання й у відповідному вікні обираємо нашу **високополігональну модель** як **“High Definition Mesh”**, яка слугуватиме джерелом геометричної деталізації для перенесення на спрощену сітку. Обов’язково ставимо галочку навпроти **“Use Cage”**, що дозволяє контролювати межі запікання й уникати артефактів.

Далі налаштовуємо параметри **Max Frontal Distance** та **Max Rear Distance**, які визначають допустиму відстань від поверхні низькополігональної моделі до високополігональної в напрямках спереду та ззаду відповідно (Рис 2.10). Ці значення варто підібрати індивідуально залежно від масштабу моделі

— занадто малі значення можуть призвести до неповного переносу деталей, тоді як надто великі — до змішування небажаних елементів. Після налаштувань запускаємо процес запікання основних карт: **Normal**, **Ambient Occlusion**, **Curvature**, **World Space Normal** та ін., які в подальшому використовуватимуться для генерації матеріалів, ефектів зношеності та деталізації текстур.

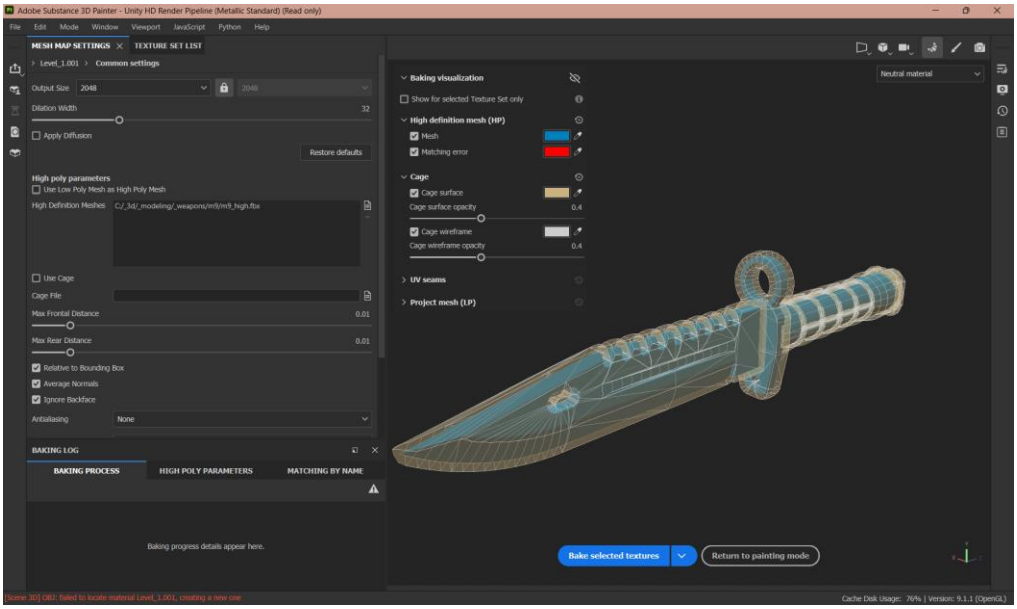


Рис 2.10 Налаштування cage та запікання текстурних карт

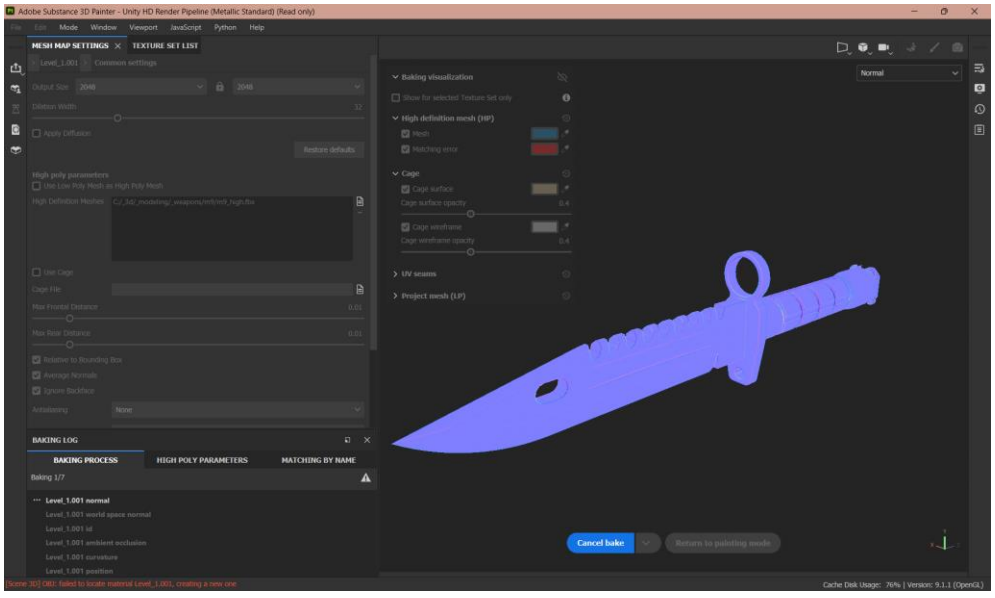


Рис 2.11 Результат запікання карти нормалей

Тепер переходимо в режим **малювання (Paint Mode)** та створюємо текстури для нашої моделі. Обираємо **базовий матеріал (Smart Material або стандартний Fill Material)** з бібліотеки Adobe Substance 3D Painter та накладаємо його на відповідну частину моделі.

Використовуємо інструмент **маскування (Mask)** — це дозволяє ізолювати окремі ділянки об’єкта, які потребують різного текстурного оформлення.

Далі детальніше налаштовуємо властивості матеріалу щоб досягти бажаного візуального ефекту. По мірі просування додаємо додаткові шари, ефекти зношення, подряпини або бруд для створення більш реалістичного зовнішнього вигляду. В результаті розробили текстури для моделі (Рис 2.12).

Рис 2.12 Модель ножа M9 з готовими текстурами

Тепер важливо правильно експортувати текстури, щоб вони були сумісні з обраним **render pipeline** в **Unity**. Залежно від того, чи використовується **Built-in Render Pipeline**, **Universal Render Pipeline (URP)** чи **High Definition Render Pipeline (HDRP)**, набір і формат текстур може дещо відрізнятись (Рис 2.13).

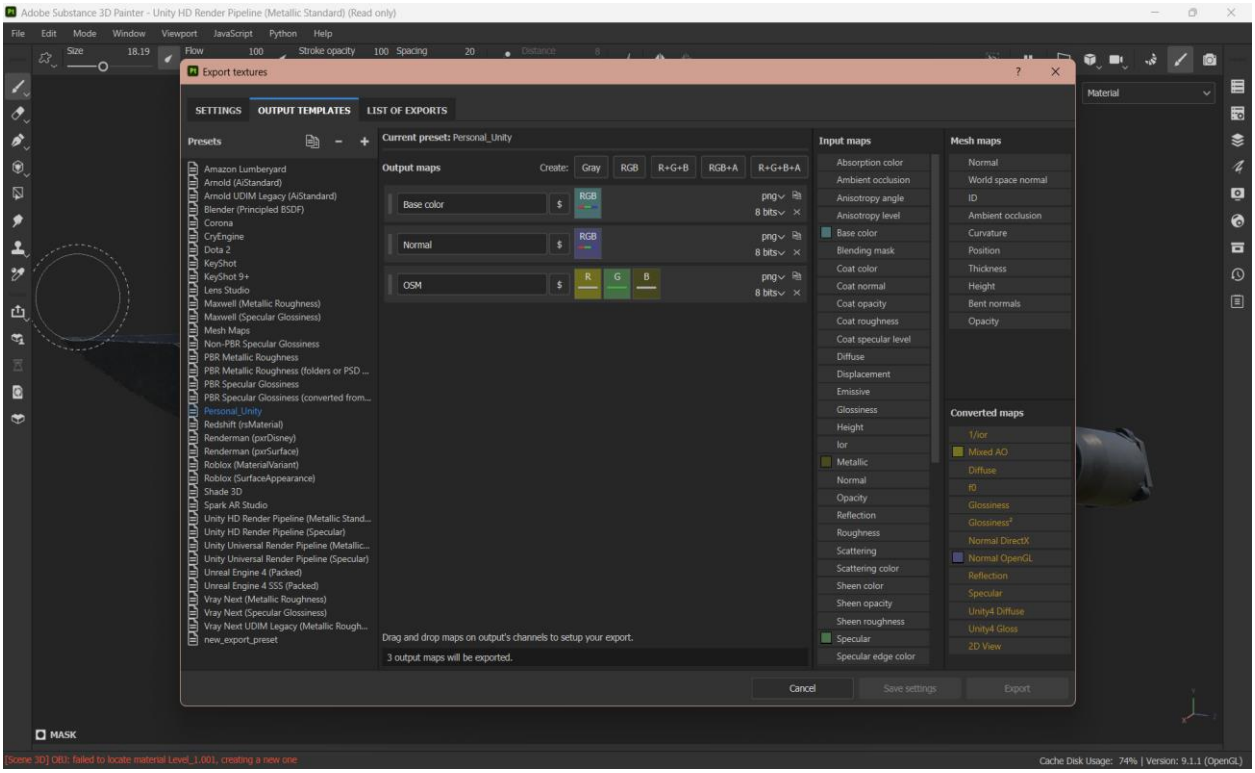


Рис 2.13 Приклад користувацьких налаштувань для середовища Unity

У **Adobe Substance 3D Painter** переходимо до вікна **Export Textures**, де в полі **Config** обираємо відповідний шаблон експорту, наприклад:

- **Unity 5 (Standard Metallic)** — для **Built-in RP**.
- **Unity URP** — для **Universal Render Pipeline**.
- **Unity HDRP** — для **High Definition Render Pipeline**.

Якщо потрібного шаблону немає, можна створити власний, вказавши правильні назви та призначення текстурних каналів. Для URP, наприклад, зазвичай експортуються такі карти:

- **Base Color**. Містить колір.
- **Mask Map (ORM)**. Містить Metallic (B), Occlusion (R), Roughness (B)
- **Normal Map.** Містить вектори нормалей.

Після вибору шаблону вказуємо роздільну здатність (наприклад, 2048x2048 або 4096x4096), формат файлів (**.png** або **.tga**) та цільову директорію експорту (Рис 2.14). Експортовані текстури потім імпортуються в Unity, де їх

необхідно правильно призначити відповідним слотам у матеріалі, створеному на основі обраного шейдера (наприклад, **Lit Shader** для URP). Це забезпечить коректне відображення матеріалів та збереження всіх візуальних властивостей моделі у фінальному середовищі.

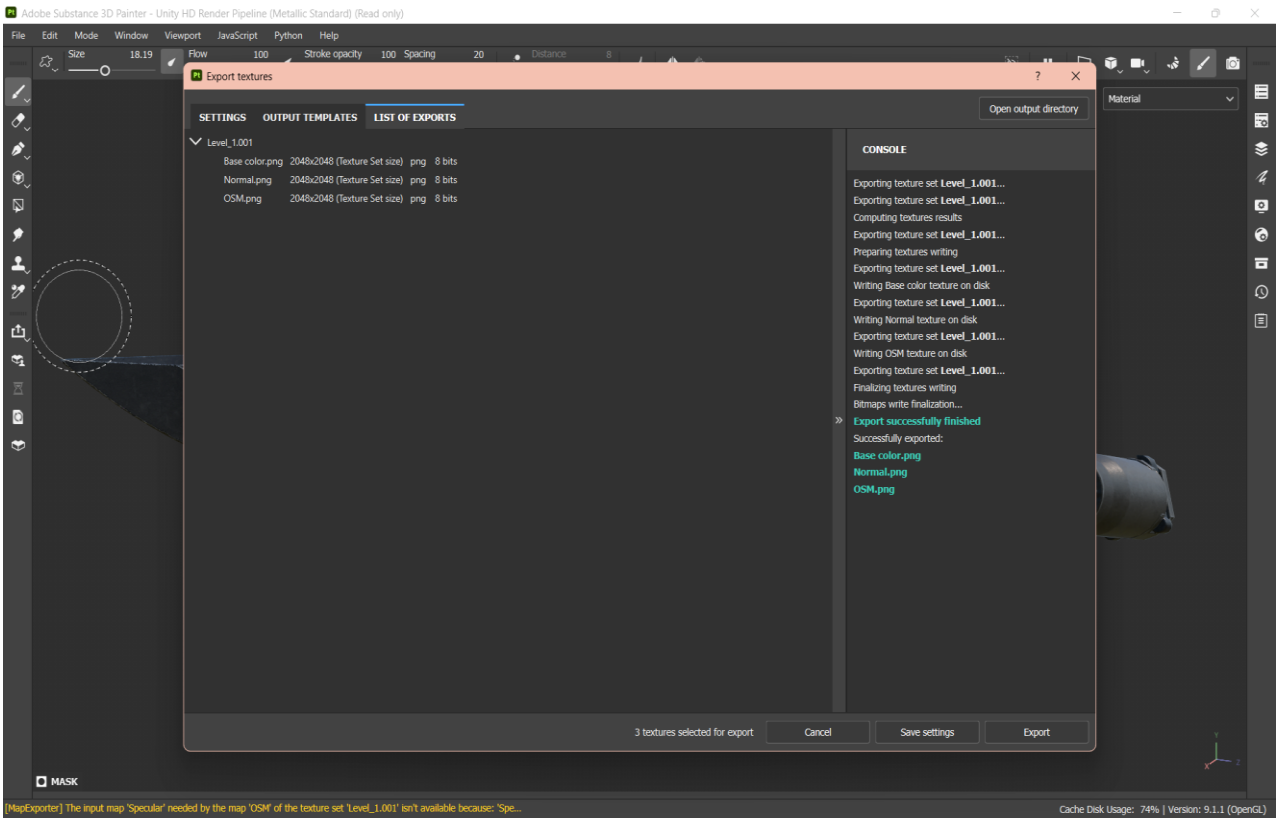


Рис 2.14 Перелік експортованих текстур

Таким чином, маємо повністю готову до використання в ігровому рушії модель ножа M9, яка включає оптимізовану геометрію, коректну UV-розгортку та набір текстур, сумісних із Unity.

Далі виконуємо аналогічні дії для підготовки інших двох моделей, повторюючи етапи оптимізації, UV-розгортки, створення високополігональної версії, запікання текстурних карт і текстурування в **Substance 3D Painter** (Рис. 2.15).

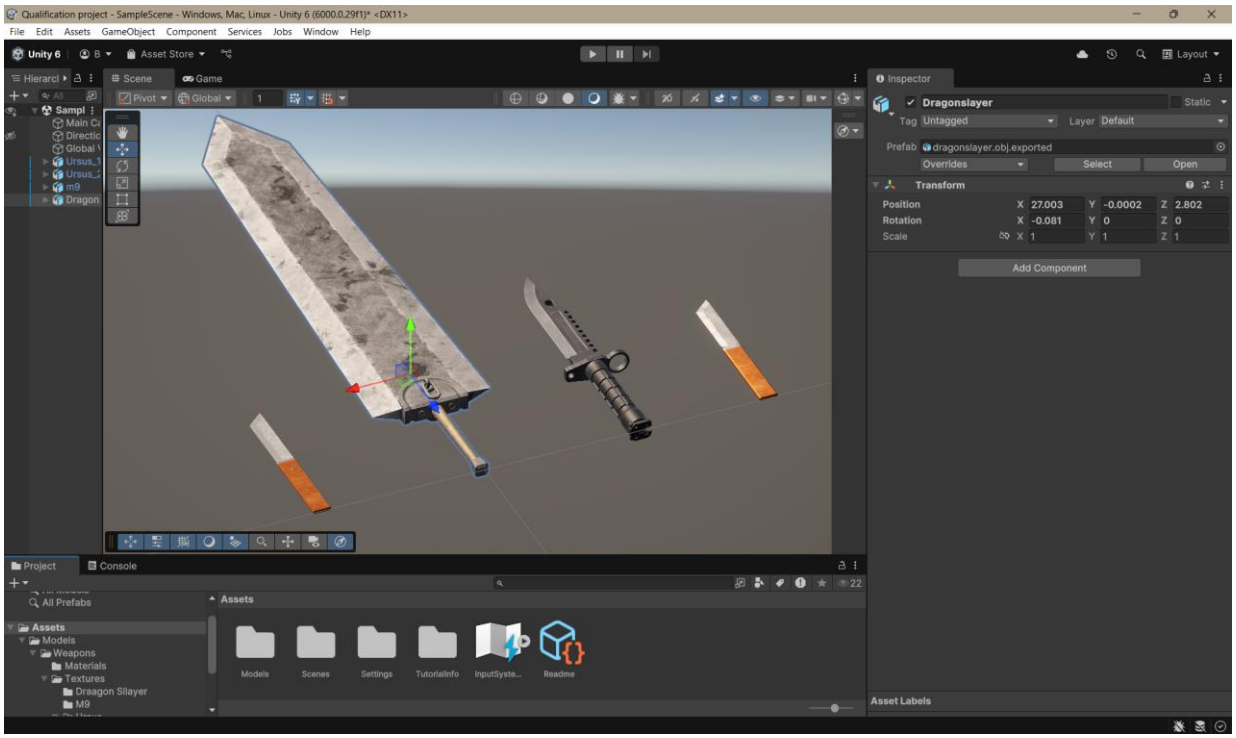


Рис 2.15. Набір моделей у вікні проєкту Unity

2.2. Інтеграція моделей у ігровий рушій

Щоб відповідним чином реалізувати розроблений набір моделей, було прийнято рішення створити інтерфейс інвентаря з функціональністю перетягування та екіпірування предметів, що забезпечує інтеграцію моделей безпосередньо в ігровий процес.

Отже, фінальний результат включатиме реалізацію ігрових механік, зокрема функціонального інтерфейсу інвентаря з можливістю екіпірування предметів шляхом їх перетягування (drag and drop) до відповідної області персонажа.

Для цього нам знадобиться зручна структура папок в проєкті Unity. В кореневій папці проєкту створимо папки: Icons, InventoryLogic, Models та Prefabs (Рис. 2.16).

- **Icons** — тут зберігаються спрайти та ефекти.
- **InventoryLogic** — для всіх скриптів.

- **Models** — для моделей, а також текстур і матеріалів.
- **Prefabs** — для префабів моделей.

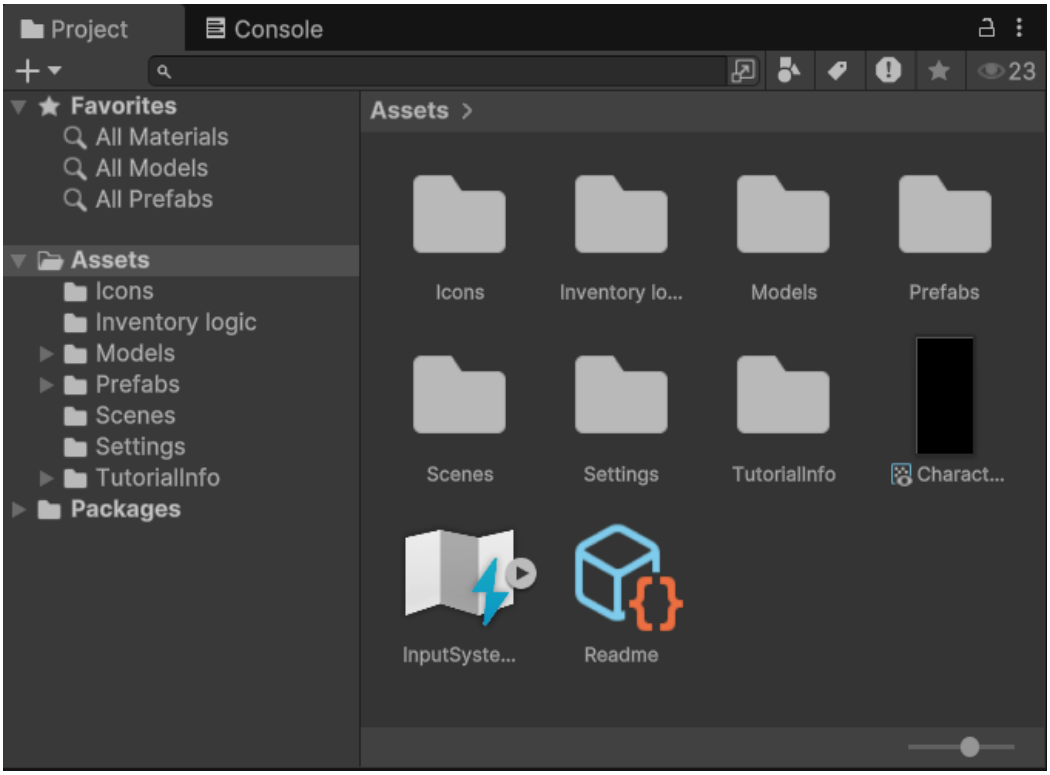


Рис 2.16 Файлова структура проєкту.

Розробка інтерфейсу користувача (UI) інвентаря почалась зі створення графічного макета та спрайтів предметів. Макет було реалізовано у зовнішньому графічному редакторі, а в якості спрайтів було використано рендери моделей з видом згори (Рис. 2.17). Даний макет є сіткою 20x25, та має розмір 625x500 пікселів (Рис. 2.18).

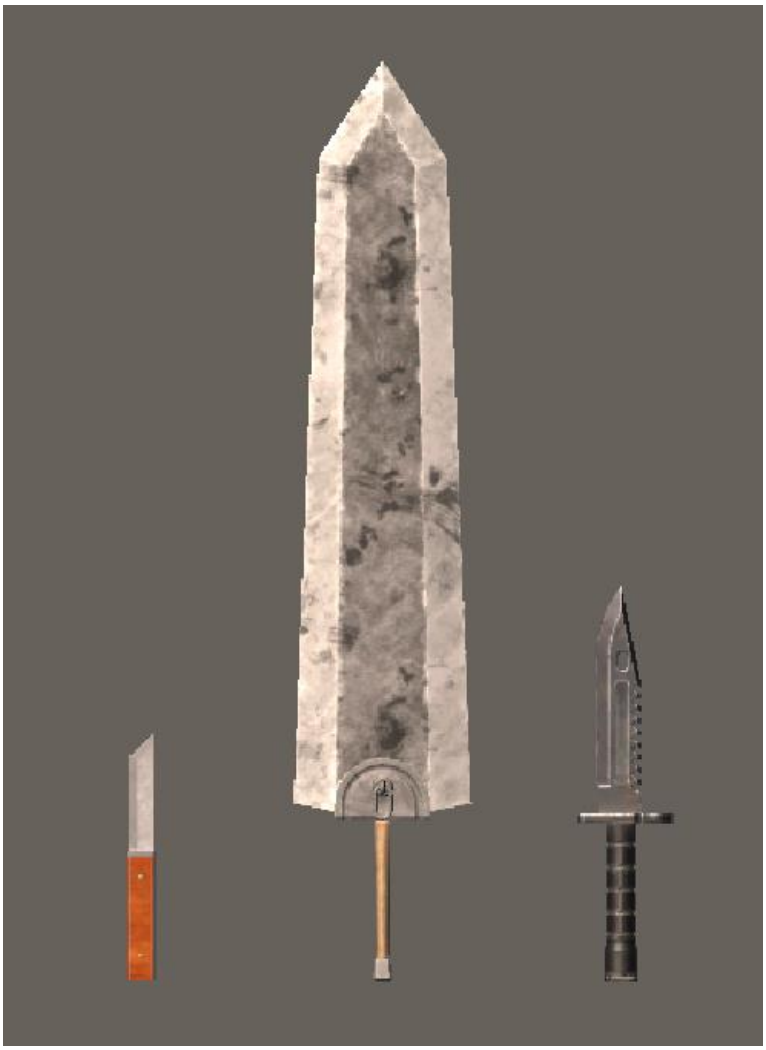


Рис 2.17. Рендери моделей з видом згори

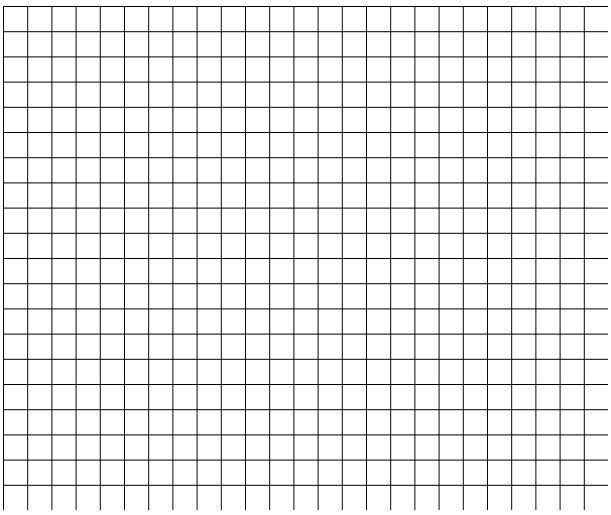


Рис 2.18 Сітка інвентаря.

Наступним етапом стало налаштування сітки у середовищі Unity. Для цього було створено скрипт **InventoryGrid**, який відповідає за управління комірками інвентаря (див. Додаток А).

Після цього було розроблено клас **InventoryItem**, що містить дані про предмети (назву, розмір, іконку, префаб моделі) (див. Додаток Б). Цей клас є основою для створення конкретних предметів інвентаря.

З метою забезпечення взаємодії користувача з предметами був створений скрипт **DraggableItem**, який реалізує інтерфейси Unity: IBeginDragHandler, IDragHandler та IEndDragHandler (див. Додаток В). Цей скрипт забезпечує перетягування предметів у межах інвентаря, перевірку допустимих меж розміщення та фіксацію предметів у комірках.

Для підсвітки комірок під час перетягування предметів було реалізовано скрипт **CellHighlighter** (див. Додаток Г). Цей скрипт визначає та підсвічує потенційні клітинки, у які можна розмістити предмет, спрощуючи користувачеві взаємодію з інтерфейсом. В якості кольору підсвічення було обрано зелений колір з середньою прозорістю.

Паралельно було створено систему екіпірування персонажа предметами з інвентаря. Для цього було реалізовано **EquipSlotUI** — скрипт, що приймає подію OnDrop та передає інформацію про предмет до менеджера екіпірування (див. Додаток Д). **EquipmentManager** відповідає за розміщення 3D-моделей предметів у руці персонажа (див. Додаток Е).

Для інтеграції персонажа в інтерфейс інвентаря було створено окрему камеру (UICamera), яка відображає 3D-модель персонажа на UI за допомогою RenderTexture та компонента RawImage (Рис. 2.19).

Останнім етапом було проведено тестування та інтеграцію всіх компонентів у єдину систему (Рис. 2.20). Під час цього етапу було виявлено та вирішено низку технічних проблем, пов'язаних з відображенням предметів у

межах інтерфейсу, взаємодією скриптів, налаштуванням масштабування та розміщення моделей у просторі Unity.

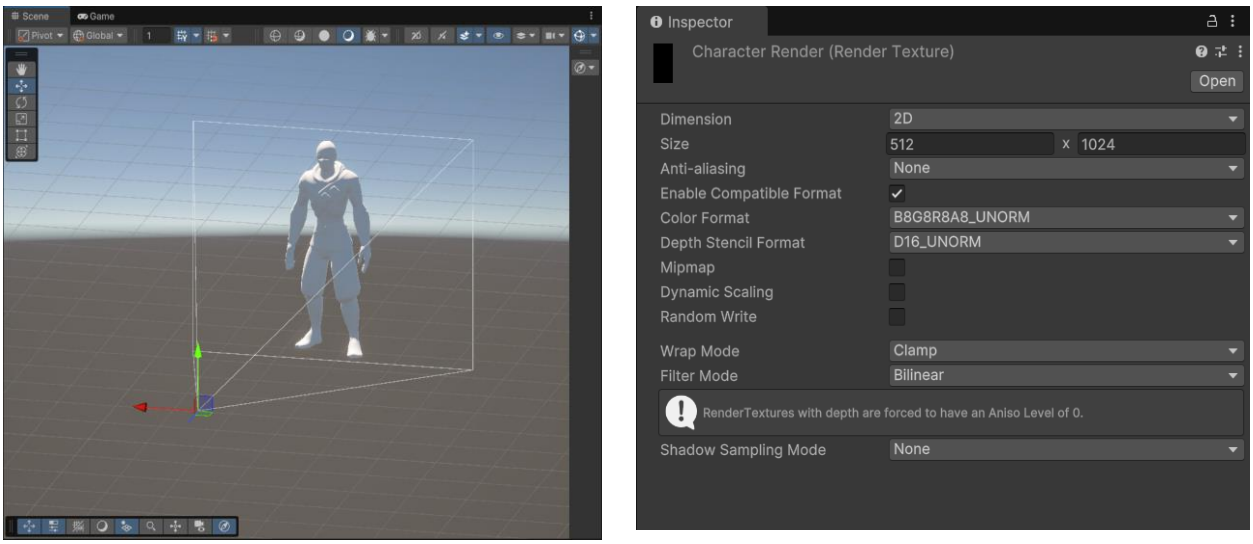


Рис. 2.19 Положення камери та елементу RenderTexture.

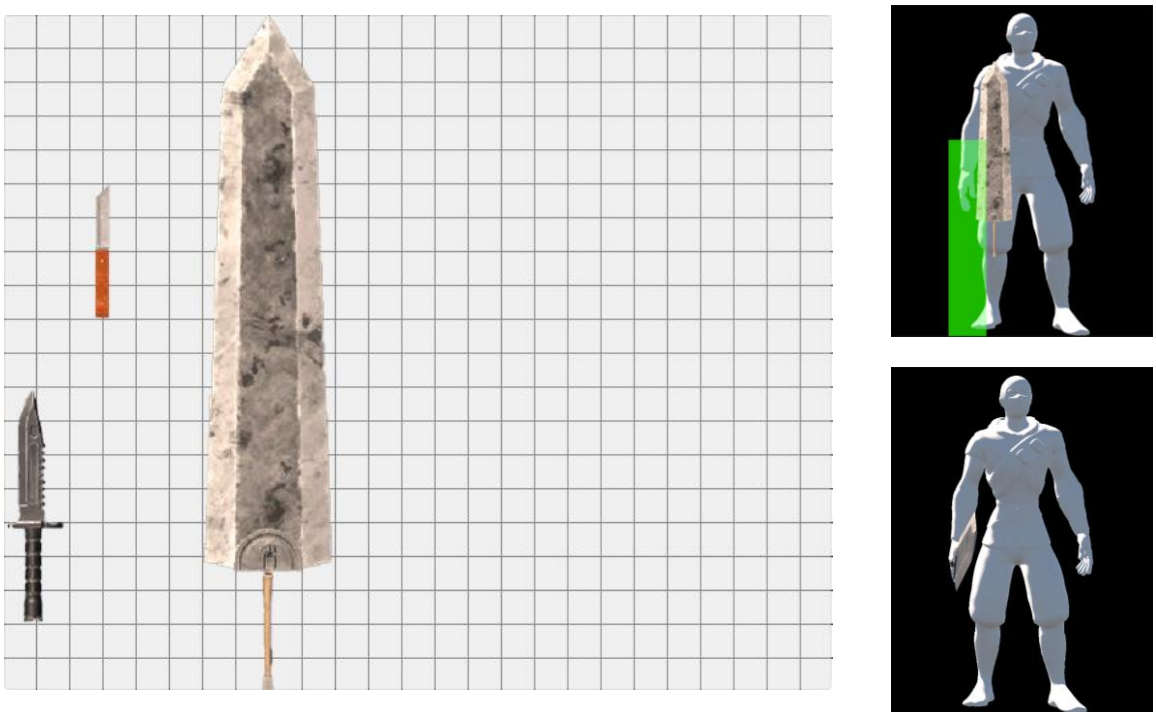


Рис 2.20 Демонстрація сітки інвентаря та функції екіпірування предмета.

Завдяки чіткій структурі класів та скриптів, а також послідовному підходу до налаштування компонентів UI, було створено надійну, гнучку та інтерактивну

систему інвентаря, що дозволяє легко управляти екіпіруванням персонажа в ігровому середовищі.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи розглянуто детальний процес створення та інтеграції 3D-моделей для комп’ютерної гри, що включає концептуалізацію, параметричне моделювання, оптимізацію геометрії, текстурування та реалізацію ігрових механік у рушії Unity.

Під час виконання поставлених завдань було здійснено:

концептуальний пошук референсних зображень, що став основою для створення відповідних 3D-моделей холодної зброї;

параметричне моделювання об'єктів у середовищі Fusion 360 із подальшою оптимізацією геометрії через MoI3D та Blender. Це дозволило створити високополігональні та низькополігональні версії моделей;

розроблено UV-розгортки в програмі RizomUV, що забезпечило правильне накладання текстурних карт на об’єкт;

виконано текстурування моделей у Substance 3D Painter з попереднім запіканням карт нормалей, Ambient Occlusion, Curvature та інших;

експортовано текстури відповідно до вимог рендер-пайплайну URP, що забезпечило коректне відображення матеріалів моделей у ігровому рушії Unity;

реалізовано систему інтеграції моделей в Unity, включаючи створення інтерфейсу інвентаря з функціональністю перетягування (drag-and-drop), екіпірування предметів, а також візуалізацію 3D-моделі персонажа через RenderTexture.

Завдяки системному підходу та використанню сучасних інструментів для створення 3D-моделей вдалося розробити зручну й гнучку систему, яку легко доповнювати новими об'єктами та ефективно використовувати в ігровому проєкті.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досягнуто основної мети – обґрунтовано вибір цільового ігрового рушія, методів, інструментів 3D-моделювання та розроблено набір тривимірних моделей, оптимізованих для використання у сучасних ігрових рушіях.

Під час дослідження проведено аналіз сучасних методів 3D-моделювання. З’ясовано переваги та недоліки полігонального, параметричного, цифрового ліплення та процедурного моделювання, встановлено їхню доцільність для різних етапів створення ігрових об’єктів. Зокрема, визначено, що для створення високоякісних моделей предметів ігрового оточення оптимальним є комбінований підхід із застосуванням параметричного моделювання, для дотримання точності форм та пропорцій, і полігонального – для подальшої оптимізації, та можливості інтеграції моделі в середовище розробки ігор .

Було проведено огляд та порівняння провідних програмних пакетів для моделювання (Blender, 3ds Max, Maya, Fusion 360, Houdini, Plasticity), текстурування (Adobe Substance 3D Painter, 3DCoat, Marmoset Toolbag) та ігрових рушіїв (Unity, Unreal Engine, CryEngine). Враховуючи критерії доступності, функціональності та ефективності, було обрано Fusion 360 для початкового етапу параметричного моделювання, Blender для оптимізації полігональної моделі, RizomUV для створення UV-розгортки, Adobe Substance 3D Painter для текстурування та Unity для інтеграції та візуалізації створених моделей.

Розроблено набір моделей холодної зброї. Виконано повний цикл розробки, що включав пошук референсів, параметричне моделювання у Fusion 360, оптимізацію геометрії у Blender, розгортання UV в RizomUV, запікання текстурних карт та текстурування у Substance 3D Painter.

Розроблені моделі були успішно інтегровані до ігрового рушія Unity. Реалізовано інтерактивний інтерфейс інвентаря з функцією перетягування (drag-

and-drop), що дозволяє екіпірувати створені моделі безпосередньо на персонажа. Ця система забезпечує не лише демонстрацію візуальної складової моделей, а й практичну взаємодію користувача з ігровими об’єктами у реальному часі.

Перспективами подальших досліджень є поглиблення інтеграції процедурного моделювання, автоматизація процесів створення та оптимізації геометрії, а також експерименти з іншими сучасними ігровими рушіями, такими як Unreal Engine 5, з використанням його новітніх технологій (Nanite, Lumen). Це дозволить покращити розуміння можливостей оптимізації етапів створення ігрового контенту та покращити візуальні та інтерактивні характеристики майбутніх проєктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Newzoo. Global Games Market Report 2024. URL: https://newzoo.com/insights/articles/global-games-market-reaches-250-billion-in-2025/ (дата звернення: 27.02.2025).
2. Lan Y. Development and Application of 3D Modeling in Game. *Academic Journal of Science and Technology*. 2023, 7(2), pp. 94–97. doi:10.54097/ajst.v7i2.11949.
3. Unity Technologies. Best practices for optimizing 3D models. URL: https://docs.unity3d.com/Manual/BestPracticeUnderstandingPerformanceInUnity.html (дата звернення: 12.10.2024).
4. Blender Manual. Asset Libraries. https://docs.blender.org/manual/en/latest/editors/asset_browser.html
5. *Adobe*. URL: https://www.adobe.com/ua/products/substance3d/discover/what-is-3d-modeling.html (дата звернення: 02.01.2025).
6. Василенко Я.П., Вольська І.П. Дидактичні аспекти вивчення 3D-моделювання з використанням середовища Blender у закладах загальної середньої освіти. *Сучасні цифрові технології та інноваційні методики навчання: досвід, тенденції, перспективи* : матеріали IX Міжнар. науковопракт. інтернет-конф., м. Тернопіль, 28 квітня 2022 р., 2022. С. 69-72.
7. Беседа А. Типи 3Д моделювання: полігональне, сплайнове та NURBS моделювання. *3D моделювання*. URL: https://rocketmen.com.ua/ua/article/nurbs (дата звернення: 31.01.2025).
8. Перейма В.В., Путіліна К.В. Використання комп'ютерних програм для створення тривимірних моделей у гуртковій роботі з паперкрафту. *Інформаційні технології в соціокультурній сфері, освіті та економіці*: матеріали IV Міжнародної науково-практичної конференції студентів і молодих учених. / М-во освіти і науки України; Київ. нац. ун-т культури і мистецтв.– Київ: Видавничий центр КНУКіМ, 2020. С. 126-128.

9. Денісов Р. В. «Особливості сучасних засобів створення анімаційних тривимірних сцен : Магістерська дисертація : 171. Київ, 2020. 97 с.
10. Косенко Д. Метод процедурної генерації об’єктів ігрового світу в жанрі “Tower Defense” на основі шумів : кваліфікаційна робота : 121. Київ, 2025. 76 с.
11. SideFX. Houdini. Version 20.0.745. Side Effects Software.
12. Mojang Studios. Minecraft. Bedrock Edition. Version 1.20. Xbox Game Studios, 2009.
13. Гривняк А. Що таке фотограмметрія, як вона допомагає зберегти українську спадщину і чим корисна для геймдеву. *GameDev спільнота*. URL: https://gamedev.dou.ua/blogs/skeiron-developer-about-photogrammetry/ (дата звернення: 01.02.2025).
14. Gints Zilbalodis. Flow, 2025. URL: https://www.youtube.com/watch?v=p17b1c8bJ5A (дата звернення: 02.02.2025).
15. Blender support – blender.org. URL: https://www.blender.org/support/ (дата звернення: 02.02.2025).
16. Autodesk. 3Ds MAX. URL: https://www.autodesk.com/products/3ds-max/overview (дата звернення: 03.02.2025).
17. Autodesk. Maya. URL: https://www.autodesk.com/eu/products/maya/overview (дата звернення: 03.02.2025).
18. Autodesk. Fusion 360. URL: https://www.autodesk.com/eu/products/fusion-360/overview (дата звернення: 03.02.2025).
19. Plasticity. URL: https://www.plasticity.xyz/ (дата звернення: 03.02.2025).
20. Adobe Substance 3D Painter. Adobe, 2023. URL: https://www.adobe.com/products/substance-3d-painter.html (дата звернення: 03.02.2025).

21. Adobe. Substance 3D Painter 2025. Adobe, 2025. URL: https://store.steampowered.com/app/3366290/Substance_3D_Painter_2025/ (дата звернення: 03.02.2025).
22. 3DCoatTextura. URL: https://pilgway.com/product/3dcoattextura (дата звернення: 03.02.2025).
23. 3DCoat PBR Material Library. URL: https://materials.3dcoat.com/scans (дата звернення: 03.02.2025).
24. Marmoset Toolbag 5. URL: https://marmoset.co/toolbag/ (дата звернення: 04.02.2025).
25. Unity Game Engine. URL: https://unity.com/ (дата звернення: 04.02.2025).
26. Render pipelines. URL: https://docs.unity3d.com/Manual/render-pipelines.html (дата звернення: 05.02.2025).
27. High Definition Render Pipeline. URL: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@17.1/manual/index.html (дата звернення: 05.02.2025).
28. Unreal Engine. URL: https://www.unrealengine.com (дата звернення: 05.02.2025).
29. CryTek. CryEngine. URL: https://www.cryengine.com (дата звернення: 06.02.2025).
30. Still T. Autodesk Fusion 360 Basics: 3D Modeling Made Easy. https://www.autodesk.com/products/fusion-360/blog/autodesk-fusion-360-basics-3d-modeling-made-easy/ (дата звернення: 08.03.2025).
31. MoI, 3D modeling for designers and artists. URL: http://moi3d.com/ (дата звернення: 07.02.2025).
32. RizomUV. URL: https://www.rizomuv.com/ / (дата звернення: 08.02.2025).

ДОДАТКИ

Додаток А

```
using UnityEngine;

public class InventoryGrid : MonoBehaviour
{
    public int width = 14;
    public int height = 22;
    public int cellSize = 25;

    private bool[,] cells;

    private void Awake()
    {
        cells = new bool[width, height];
    }

    public void ClearItemArea(int x, int y, int itemWidth, int
itemHeight)
    {
        for (int i = x; i < x + itemWidth; i++)
        {
            for (int j = y; j < y + itemHeight; j++)
            {
                cells[i, j] = false;
            }
        }
    }

    public bool CanPlaceItem(int x, int y, int itemWidth, int
itemHeight)
    {

        if (x < 0 || y < 0 || x + itemWidth > width || y + itemHeight
> height)
            return false;

        for (int i = x; i < x + itemWidth; i++)
        {
            for (int j = y; j < y + itemHeight; j++)
            {
                if (cells[i, j]) return false;
            }
```

```
        }

        return true;
    }

    public void PlaceItem(int x, int y, int itemWidth, int
itemHeight)
    {
        for (int i = x; i < x + itemWidth; i++)
        {
            for (int j = y; j < y + itemHeight; j++)
            {
                cells[i, j] = true;
            }
        }
    }

    public Vector2 GetCellPosition(int x, int y)
    {
        return new Vector2(x * cellSize, -y * cellSize);
    }
}
```

Додаток Б

```
using UnityEngine;

[CreateAssetMenu(fileName = "NewItem", menuName = "Inventory/Item")]
public class InventoryItem : ScriptableObject
{
    public string itemName;
    public int width;
    public int height;
    public Sprite icon;
    public GameObject modelPrefab; // 3D-модель
}
```

```
using UnityEngine;
using UnityEngine.EventSystems;

public class DraggableItem : MonoBehaviour, IBeginDragHandler,
IDragHandler, IEndDragHandler
{
    private RectTransform rectTransform;
    private CanvasGroup canvasGroup;
    private Vector2 originalPosition;

    private InventoryUI inventoryUI;
    private InventoryItem itemData;
    private InventoryGrid grid;
    private CellHighlighter highlighter;
    private RectTransform gridRectTransform;

    public InventoryItem ItemData => itemData;

    private void Awake()
    {
        rectTransform = GetComponent<RectTransform>();

        canvasGroup = GetComponent<CanvasGroup>();
        if (canvasGroup == null)
            canvasGroup = gameObject.AddComponent<CanvasGroup>();

        inventoryUI = Object.FindFirstObjectByType<InventoryUI>();
        if (inventoryUI == null)
        {
            Debug.LogError("InventoryUI not found");
            return;
        }

        grid = inventoryUI.grid;
        if (grid == null)
        {
            Debug.LogError("InventoryGrid not assigned in
InventoryUI");
            return;
        }

        gridRectTransform = grid.GetComponent<RectTransform>();
```

```
        highlighter = grid.GetComponent<CellHighlighter>();
        if (highlighter == null)
        {
            Debug.LogError("CellHighlighter not found on
InventoryPanel");
            return;
        }

        ItemUIReference refComponent =
GetComponent<ItemUIReference>();
        if (refComponent == null)
        {
            Debug.LogError("ItemUIReference component missing on
ItemUI");
            return;
        }

        itemData = refComponent.itemData;
        if (itemData == null)
        {
            Debug.LogError("ItemData not set in ItemUIReference");
        }
    }

    public void OnBeginDrag(PointerEventData eventData)
    {
        originalPosition = rectTransform.anchoredPosition;
        canvasGroup.blocksRaycasts = false;

        Vector2 gridRelative = originalPosition -
gridRectTransform.anchoredPosition;

        int oldX = Mathf.FloorToInt(gridRelative.x /
grid.cellSize);
        int oldY = Mathf.FloorToInt(-gridRelative.y /
grid.cellSize);

        grid.ClearItemArea(oldX, oldY, itemData.width,
itemData.height);
    }

    public void OnDrag(PointerEventData eventData)
    {
        rectTransform.anchoredPosition += eventData.delta;
```

```
        canvasGroup.blocksRaycasts = false;

        Vector2 gridRelative = rectTransform.anchoredPosition -
gridRectTransform.anchoredPosition;

        gridRelative.x -= (itemData.width * grid.cellSize) / 2f;
        gridRelative.y -= (itemData.height * grid.cellSize) / 2f;

        int x = Mathf.FloorToInt(gridRelative.x / grid.cellSize);
        int y = Mathf.FloorToInt(-gridRelative.y / grid.cellSize);

        highlighter.ShowHighlight(gridRectTransform, x, y,
itemData.width, itemData.height, grid.cellSize);
    }

    public void OnEndDrag(PointerEventData eventData)
    {
        canvasGroup.blocksRaycasts = true;
        highlighter.Clear();

        Vector2 gridRelative = rectTransform.anchoredPosition -
gridRectTransform.anchoredPosition;

        gridRelative.x -= (itemData.width * grid.cellSize) / 2f;
        gridRelative.y -= (itemData.height * grid.cellSize) / 2f;

        int x = Mathf.FloorToInt(gridRelative.x / grid.cellSize);
        int y = Mathf.FloorToInt(-gridRelative.y / grid.cellSize);

        if (x < 0 || y < 0 || x + itemData.width > grid.width || y
+ itemData.height > grid.height)
        {
            rectTransform.anchoredPosition = originalPosition;

            Vector2 oldRelative = originalPosition -
gridRectTransform.anchoredPosition;
            int ox = Mathf.FloorToInt(oldRelative.x /
grid.cellSize);
            int oy = Mathf.FloorToInt(-oldRelative.y /
grid.cellSize);

            grid.PlaceItem(ox, oy, itemData.width,
itemData.height);
```

```
            return;
        }

        if (grid.CanPlaceItem(x, y, itemData.width,
itemData.height))
        {
            rectTransform.anchoredPosition =
grid.GetCellPosition(x, y);
            grid.PlaceItem(x, y, itemData.width, itemData.height);
        }
        else
        {
            rectTransform.anchoredPosition = originalPosition;

            Vector2 oldRelative = originalPosition -
gridRectTransform.anchoredPosition;
            int ox = Mathf.FloorToInt(oldRelative.x /
grid.cellSize);
            int oy = Mathf.FloorToInt(-oldRelative.y /
grid.cellSize);

            grid.PlaceItem(ox, oy, itemData.width,
itemData.height);
        }
    }
}
```

```
using UnityEngine;
using System.Collections.Generic;

public class CellHighlighter : MonoBehaviour
{
    public GameObject highlightPrefab;
    private List<GameObject> activeHighlights = new
List<GameObject>();

    public void ShowHighlight(RectTransform parent, int x, int y,
int width, int height, int cellSize)
    {
        Clear();

        for (int i = x; i < x + width; i++)
        {
            for (int j = y; j < y + height; j++)
            {
                GameObject h = Instantiate(highlightPrefab,
parent);
                RectTransform rt = h.GetComponent<RectTransform>();
                rt.sizeDelta = new Vector2(cellSize, cellSize);


                rt.anchoredPosition = new Vector2(
                    i * cellSize,
                    -j * cellSize + cellSize
                );

                activeHighlights.Add(h);
            }
        }
    }



    public void Clear()
    {
        foreach (var h in activeHighlights)
            Destroy(h);
        activeHighlights.Clear();
    }
}
```

Додаток Д

```
using UnityEngine;
using UnityEngine.EventSystems;

public class EquipSlotUI : MonoBehaviour, IDropHandler
{
    public EquipmentManager equipManager;

    public void OnDrop(PointerEventData eventData)
    {
        DraggableItem dragged =
eventData.pointerDrag?.GetComponent<DraggableItem>();
        if (dragged != null)
        {
            Debug.Log("Item equipped: " + dragged.name);
            equipManager.Equip(dragged.ItemData);
        }
    }
}
```

Додаток Е

```
using UnityEngine;

public class EquipmentManager : MonoBehaviour
{
    public Transform rightHand;
    private GameObject currentWeapon;

    public void Equip(InventoryItem item)
    {
        if (item == null || item.modelPrefab == null)
        {
            Debug.LogWarning("Item or modelPrefab is null");
            return;
        }

        if (currentWeapon != null)
        {
            Destroy(currentWeapon);
        }

        currentWeapon = Instantiate(item.modelPrefab, rightHand);

        currentWeapon.transform.localPosition = Vector3.zero;
        currentWeapon.transform.localRotation =
Quaternion.identity;
        currentWeapon.transform.localScale = Vector3.one;

        Debug.Log("Equipped item: " + item.itemName);
    }
}
```