МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

Навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки

“До захисту допущений
Зав. Кафедри комп'ютерних наук
та прикладної математики
_____
«___» _____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

Розробка візуальної частини карткової гри з використанням NFC-карток

Виконав: Кучковський Богдан Андрійович

студент навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки

групи КН-41. _____

Підпис

Керівник: асистент Белозерова О. Д.

(науковий ступінь, вчене звання, посада, прізвище, ініціали) _____

Підпис

Рівне – 2024

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 40 сторінок, 17 малюнків, 7 джерел.

Об’єктом дослідження кваліфікаційної роботи є новітні розробки та засоби для ігрових програмних продуктів.

Предметом дослідження роботи є комп’ютерна карткова гра, що використовує NFC-картки.

Мета роботи: реалізація візуальної частини карткової гри з використанням NFC-карток, котрий стане фундаментом для подальшого забезпечення ігрової логіки. Головними цілями роботи є:

1. Створити 3D моделі ігрового поля, його середовища (зовнішнього та внутрішнього) та карток;
2. Створити зображення для карток;
3. Забезпечити збереження даних у базі даних;
4. Реалізувати анімацію.

Ключові слова: РОЗРОБКА, МОДЕЛЮВАННЯ, UNITY, BLENDER, CINEMA 4D, КОЛЕКЦІЙНІ КАРТКОВІ ІГРИ.

ВСТУП

Після своєї появи, через десятки років індустрія комп’ютерних ігор зайняла свою нішу на ринку, будучи поряд з іншими видами розваг сфери мультимедіа: мультиплікація, музика, кіно тощо. Згідно статистики Rock Paper Shotgun, у 2019 року, на платформі Steam вийшло 8 112 ігор, що значно перевищує показник 2017 року з результатом 7 672 гри. Що найцікавіше, кількість випусків ігор зростає з помітним стрибком. Протягом 2020 року було випущено 9 717 ігор, у 2021 – 11 341 гра, у 2022 – 12 476 ігор, а у 2023 – 14 544, що станом на сьогодні є піком за всю історію існування Steam. І ось, до середини 2024 року, платформа поповнила свою бібліотеку новими 7 577 іграми. Причиною різкого зростання пов’язане із запровадженням Steam Direct, що значно полегшило розробникам публікацію ігор без суворого процесу затвердження.

Зараз ігри охоплюють неабияк велику аудиторію, використовуючи найрізноманітніші ігрові пристрої. Що цікаво, на думку Філа Спенсера, ігри приносять неабияку користь, тому що ігри "…дають змогу людям стати рівними." Як не дивно, це є істина: усі ми граємо в ігри, незалежно від статі, раси, релігії, віку, політичних переконань національності чи здібностей. Ігри зводять людей разом, об'єднуючи гравців за допомогою унікальної, проте зрозумілої усім, мови – задоволення.

Люди використовують ігри не стільки щоб відпочити та розслабитись, скільки а як можливість гарно провести час у грі, котра дарує незабутній ігровий досвід, дозволяє пірнути у інший світ, відволікаючись від проблем реального світу. Ігри надають змогу гравцеві отримати незабутні емоції, а також минути через емоційні переживання. Мушу додати, що ті проекти, котрі відкрили портал між реальністю та ігровим світом, різко створили явище з назвою "культова гра".

Для того, щоб справді звернути увагу гравця, відеогра повинна включати в себе не тільки захоплюючий сюжет, логічну послідовність подій та багатий

функціонал, а й досконалу графіку. Згідно результатів дослідження Play To Earn Games та ESA, приблизно 67% геймерів вважають графіку важливим фактором під час купівлі відеоігор. Ця перевага підкреслює значну роль, котру відіграє візуальна привабливість і, як правило, чим кращою є графіка гри, тим ефективніше відбуватиметься процес занурення та задоволення.

Таким чином метою цієї бакалаврської роботи стала розробка візуальної частини карткової гри, що включала в себе не тільки створення 3D-об’єктів, а й налаштування та кодування анімацій. Даний проект буде створено у середовищі розробки Unity з використанням однойменного рушія, що є однією з найбільш популярних платформ, котра надає усі необхідні інструменти для створення відеоігор. Сам Unity дає можливість працювати зі зручним інтерфейсом, потужними засобами програмування та візуалізації для реалізації різних цікавих ідей та дослідження можливості процедурної генерації в ігровому середовищі.

РОЗДІЛ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз ігрових середовищ

Станом на 2024 рік, геймери мають великий вибір серед ігрових платформ для того, щоб насолодитися проведенням часу в улюблених відео-іграх. До списку ігрових платформ можна віднести ПК та ноутбуки, мобільні пристрої, консолі, а також VR (Virtual Reality) та AR (Augmented reality) середовища.

Дані платформи дозволяють гравцям користуватися великими бібліотеками ігор, котрі охоплюють як популярні жанри (шутери, пісочниці, рольові та інші), так і до менш популярних жанрів (мовні, словесні чи азартні ігри).

На жаль, важко чітко сказати, котре серед ігрових середовищ є найбільш популярним, оскільки, наприклад, консольні геймери використовують одну або кілька платформ. Згідно статистики DFC Intelligence стверджується, що 8% із 3,1 мільярда геймерів вважають себе винятково консольними геймерами, 48% використовують ПК та/або ноутбуки, а решта – мобільні пристрої [1]. Як зображено на Рис. 1.1., домінантами на ігровому ринку залишаються саме комп’ютерні ігри, проте, насправді переважна частина геймерів є гравцями кількох пристроїв і мають великий вибір з приводу того як проводити власну ігрову сесію.

Варто також відзначити, що хмарні ігри можуть досягти 13,5 мільярдів доларів річного доходу за п’ять років. Неважко зрозуміти, що кожне з ігрових середовищ має власний вибір платформ (наприклад, для ПК та ноутбуків це Windows, macOS та Linux). Тож, аби зрозуміти котре з ігрових середовищ найбільш використовуване, варто ознайомитися з кожним середовищем окремо.

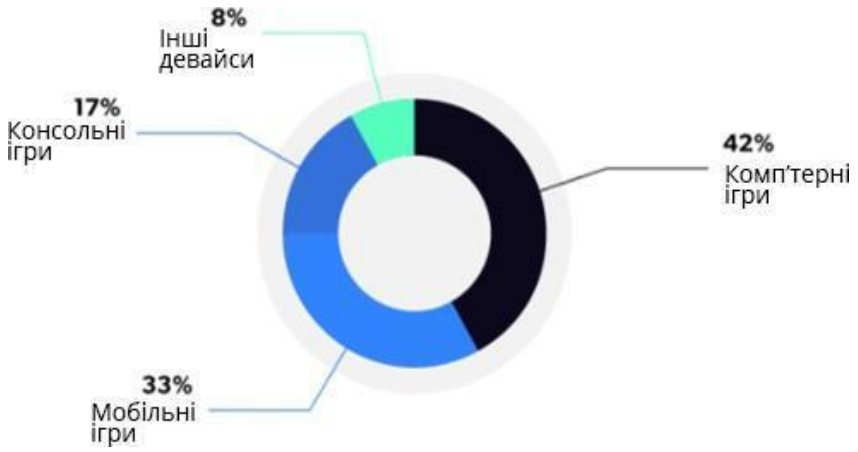


Рис. 1.1. Статистика популярності ігор та їх середовищ

1.1.1. Консолі

Варто почати з консолей та їх платформ, оскільки 8% усіх геймерів вважають себе консольними гравцями та найбільше витрачають. Найбільшими розробниками ігрових консолей варто вважати Microsoft, Nintendo та Sony. Вони мають власне апаратне та програмне забезпечення, таким чином даючи змогу геймерам переходити між платформами.

Програмне забезпечення ігор доступне або компактних дисках, або на DVD-дисках, але раніше використовувалися картриджі з чіпами пам'яті. Аби відтворювати зображення з консолей, гравцю необхідно мати телевізор або монітор.

Основною відмінністю від настільних ПК є те, що консолі працюють на операційних системах та процесорах. Ігри для консолей не можуть бути сумісними з іншими ігровими системами, такими як приставки чи ПК, хоча розробники можуть створювати ігри для кількох платформ одночасно.

Що стосується популярності, серед даних платформ останніми роками відбулися великі зміни. Якщо PlayStation 4 від Sony була домінантом упродовж багатьох років, Nintendo Switch стала бестселером у 2020 році, продавши майже 15,6 млн одиниць по всьому світу. Нова PlayStation 5 від Sony, котра вважалася найкращою ігровою консоллю 2021 року, розійшлася тиражем близько 4.4 мільйона одиниць за перші кілька місяців з дати запуску, в той час як Switch від Nintendo до кінця 2021 року було продано в обсязі 100

млн одиниць. Однак варто зазначити, що ринок консолей є дуже нестабільним, і станом на 2023 рік Sony PS5 повернув лідерство, здійснивши тираж 50 млн одиниць.

1.1.2. Комп’ютери та ноутбуки

Найбільша кількість геймерів (близько 1,77 мільйонів гравців) припадають на настільні ПК. Візуальні елементи, як правило, вищої якості саме на ПК, ніж на консолях. Крім того, ПК більш універсальні за рахунок більшої кількості взаємодій з точки зору гравців. Одним з прикладів є те, що гравці можуть використовувати клавіатуру та мишу, маючи можливість використовувати й інші пристрої, такі як джойстики, бездротові контролери, а у випадках з перегонами навіть кермо. Також варте уваги те, що комп’ютерні геймери мають можливість грати як у програмні ігри, так і в браузерні.

Також геймери на ПК мають набагато більший вибір серед ігор за рахунок більшої кількості платформ, таких як Valve, Ubisoft, EA тощо, однак попри це, варто зважати на те, що велика кількість відомих проектів, як-от «World of Warcraft», «Dota 2», «The Witcher 3: Wild Hunt» та інші, доступні лише на ПК. Ще однією ключовою перевагою, котру мають комп’ютерні геймери, полягає у здатності оновлення та покращення персонального комп’ютера чи ноутбука, за рахунок оновлення графічного процесора, оперативної пам’яті, центрального процесора тощо, замість того, щоб купувати оновлені консолі.

Що найбільше відрізняє ПК – це платформи, на яких вони працюють. Це операційні системи, як Windows (Microsoft), macOS (Apple), Linus та Chrome OS (Google), з частками ринку 87,65%, 9,54% і 0,41% відповідно (див. Рис. 1.2). Стосовно ігор, найбільша кількість проектів підтримується Windows, таким чином зміщуючи кількість цифр у її бік. На відміну від macOS набагато дорожче купувати та оновлювати систему, оскільки їм не вистачає підтримки багатьох популярних проектів. Саме тому, Windows безсумнівно є

найпопулярнішою платформою для ПК, тоді як категорія ПК з Windows є, мабуть, чи не найпопулярнішим ігровим середовищем у всьому світі.

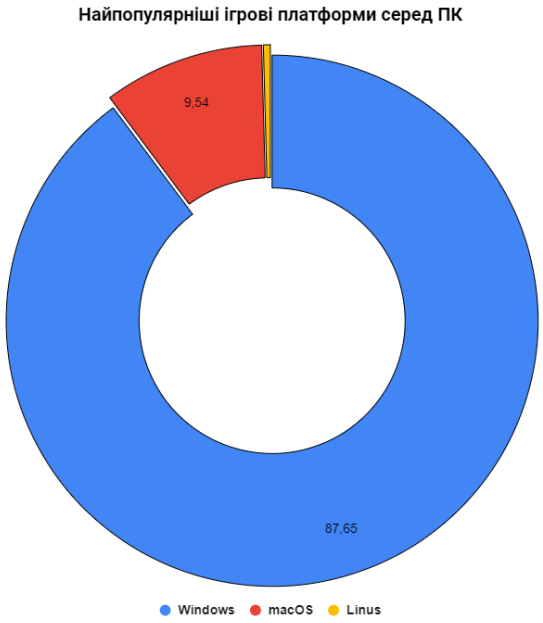


Рис. 1.2. Статистика популярності ігрових ПК платформ

1.1.3. Мобільні пристрої

Після ПК та консолей мобільні ігри стали рушійною силою світового ринку, зібравши понад 90 мільйонів доларів доходу. Оскільки понад шість мільярдів людей мають смартфони, не дивно, що мобільні ігри складають понад 53% ринку.

Загальним фактом є те, що більшість людей носять свої мобільні пристрої з собою, і кожен телефон є потенційним ігровим пристроєм. Переважну кількість мобільних ігор можна грати безкоштовно, тому доступ до мобільних ігор є значно більшим, порівняно з іграми на ПК та консолях. Через це серед людей почалися великі зміни, адже ті, хто раніше не вважав себе геймерами, тепер грають в ігри. Інший аспект цього явища полягає у тому, що мобільні ігри набагато дешевші та простіші у виробництві, ніж комп’ютерні та консольні ігри вищого класу. Це і є причиною того, чому мобільних ігор набагато більше.

Згідно поточної статистики, у світі налічується понад 2,2 мільярдів мобільних ігор, хоча ступінь збігу мобільного, ПК та консольного ігрового програмного забезпечення є невідомим. Найбільшими у світі мобільними платформами є Android, iOS та Samsung, які ділять між собою 72,44%, 26,75% і 0,41% світового ринку (див. Рис. 1.3). Крім того, прогнозується, що мобільні ігри будуть домінувати в найближчому майбутньому.

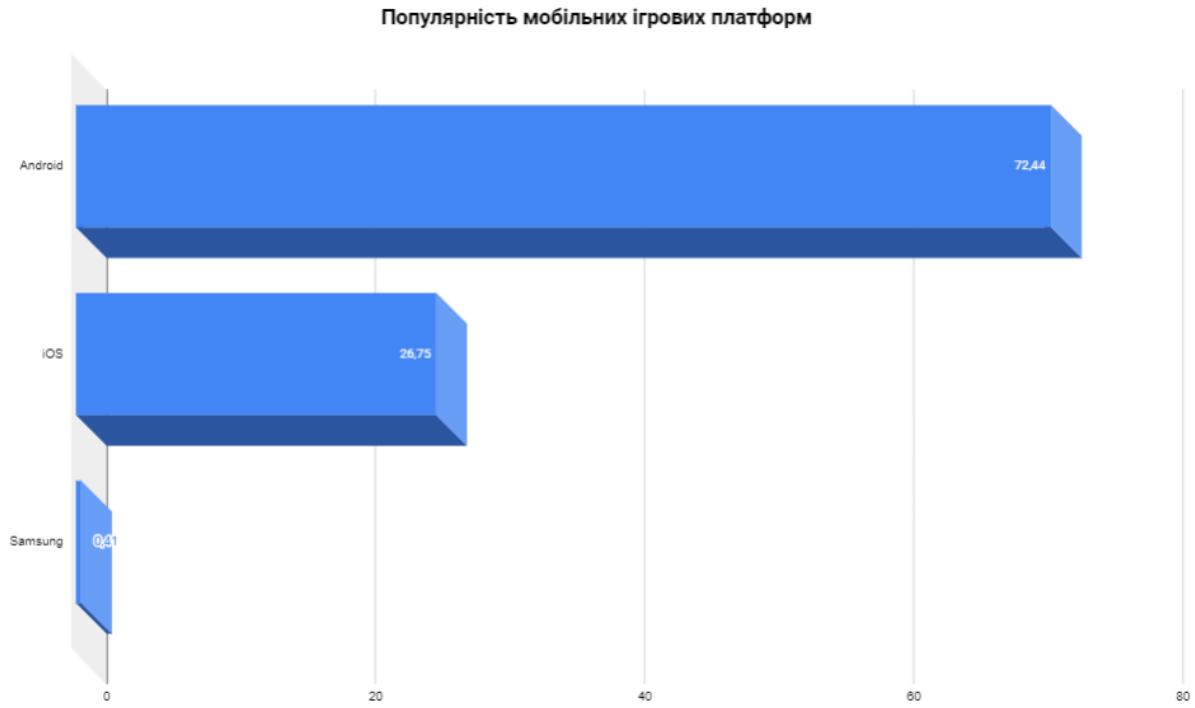


Рис. 1.3. Статистика популярності ігрових мобільних платформ

Разом зі зростанням мобільних ігор з’являються проблеми щодо безпеки, особливо щодо конфіденційної інформації та платіжних методів. Проте, на щастя, розробникам відомо про це, через що вони впроваджують підвищення безпеки та пропонують безпечні способи оплати, зокрема криптовалютою, адже її популярність вже увійшла в ігрову індустрію.

1.1.4. VR/AR середовища

ПК, консолі та мобільні пристрої вже довгий час присутні на ігровому ринку. Важко озвучити загальну кількість ігрових проектів, котрі поширилися на дані середовища. Однак, починаючи з 2012 року, з виходом першого

прототипу Oculus Rift, у світ ігрової індустрії різко увійшло середовище VR (ВР, або «віртуальна реальність») та AR (ДР, або «доповнена реальність»).

З точки зору технічної, PS VR поступається пропозиціям персональних комп'ютерів за рахунок застарілої технології відстеження. Крім цього, ігор для PS VR значно менше, у зв'язку з чим він є менш привабливим продуктом в загальному. Фінальним недоліком даного середовища є те, що для використання VR необхідно використовувати більш потужний ПК.

1.1.5. Висновок щодо вибору ігрового середовища

Ознайомившись з усіма ігровими середовищами, важко визначити остаточного переможця, котрий буде безпосередньо кращим у всьому. Усе залежить від очікувань та вимог геймера. Якщо потрібно мати доступ до ігор, перебуваючи вдома чи деінде, мобільні пристрої підійдуть найкраще. Якщо користувач бажає зіграти в ігри з максимальними параметрами графіки, найкращим рішенням стане ігровий ПК. Якщо ж користувачу потрібна спеціальна ігрова машина, котра буде простою у використанні та матиме доступ до великої бібліотеки ігор, включно з ексклюзивними ігровим програмним забезпеченням, варто звернути увагу на ігрову консоль.

1.2. 3D моделювання у відео-іграх

Моделювання при створенні відеогри має неабияк важливе значення, адже ігрові проекти можуть мати великий рівень складності та деталізації. Графічні ефекти, звукові ефекти, музичний супровід та діалоги – усе це необхідне для того, аби створити деяку уявну лінію між реальністю та ігровим світом, покращуючи ігровий досвід геймера. Деякі з ігор також мають деяку лінію прогресу у вигляді нагород та досягнень, котрі впливають на гравця, стимулюючи його до досягнення нових рівнів, ігрового розвитку, а також покращення своїх ігрових навичок.

Моделювання є однією з тих важливих завдань, котрі забезпечують зацікавленість гравця, демонструючи реалістичні чи фантастичні світи,

персонажів, механік гри. Загалом моделювання включає в себе такі ключові аспекти, як фізичне моделювання, графічне моделювання, штучний інтелект, моделювання ігрової механіки та процедурне моделювання.

Фізичне моделювання включає в себе забезпечення фізичних законів, як-от гравітація, тертя чи інерція, а також створення правдоподібних рухів персонажів та об’єктів. Це важливо для тих ігор, у яких повинен бути високий рівень реалістичності. Крім того, фізичне моделювання використовується для створення реалістичних водних поверхонь.

Графічне моделювання використовується для створення тривимірних моделей та персонажів, об’єктів та середовищ. Воно включає у себе створення полігональних моделей, використання текстур, а також анімацію. Також графічне моделювання використовується для забезпечення реалістичного освітлення та тіней. Це досягається за рахунок використання різних технік, таких як рейтрейсинг (трасування променів).

Штучний інтелект дозволяє створювати поведінку неігрових персонажів (так званих NPC), котрі по-різному реагують на гравця та його дії, а також і на навколишнє середовище. Це допомагає додати в гру такі аспекти, як планування, прийняття рішень та навчання.

Моделювання ігрової механіки передбачає налаштування фізичну взаємодію об’єктів як-от, наприклад, зіткнення, руйнування та інші маніпуляції. Що цікаво, ігри можуть моделювати складні економічні системи, котрі дають змогу гравцям керувати ресурсами, торгувати або взаємодіяти з іншими персонажами, будь це NPC чи інший гравець.

Процедурне моделювання включає в себе генерацію контенту та еволюцію ігрових світів. Процедурні алгоритми використовуються для створення великих ігрових світів та рівнів, котрі змінюються та розвиваються в реальному часі, залежно від дій гравця, а також квестів. Це значно зменшує ручну працю розробників, а також забезпечує різноманітність контенту відеогри.

Загалом, моделювання у відеоіграх є комплексним та багатогранним процес, котрий потребує синергії між різними технічними та творчими аспектами. Завдяки постійному розвитку технологій, ігри стають більш реалістичними та захоплюючими, даючи геймерам незабутні враження та цінний ігровий досвід.

1.3. Ігри жанру колекційна карткова гра

Ігрова індустрія славиться великим різноманіттям ігрових жанрів, котрі не просто відрізняються своєю унікальністю, а й надають можливість гравцеві знайти щось, що приглянеться його душі. Як не дивно, неможливо дуже чітко класифікувати жанри ігор, адже одна і та ж гра може легко ставитися до різних жанрів. Проте така класифікація є неабияк корисною, тому що гравці вважають за краще певний вид ігор і, після анонсу конкретного жанру від розробника, можуть легко визначити буде їм цікава гра чи ні. Аби краще зрозуміти таку стратегію поділу, можна взяти за приклад ігри під загальним жанром стратегії. Здавалося б, стратегія є стратегія – ціль гравця ухвалювати зважені рішення, котрі вплинуть на кінцевий результат. Так, безсумнівно, це і є особливістю даного жанру. Проте не варто забувати, що стратегії є також різних жанрів: економічні стратегії («Jurassic World Evolution», «SimCity», «Castle Clash» тощо), ігри типу «захисні башти» («Plants vs. Zombies», «Orcs Must Die! Unchained», «Kingdom Rush» і подібні), військові стратегії («Blitzkrieg», «World War II: Frontline Command», «Men of War» і т. п.), карткові стратегії («Hearthstone», «Magic: The Gathering», «Yu-Gi-Oh!») тощо. Саме зважаючи на ці піджанри гравцеві вдається знайти ту гру, котра ідеально підійде його вимогам. Оскільки метою бакалаврської роботи було створення карткової гри, пропоную зупинитися та ознайомитися з цим видом гри детальніше.

Свій початок жанр TCG (Trading Card Game, або ККІ – колекційна карткова гра) бере починаючи з 1990-х років, оскільки саме тоді з’явилися найперші карткові логічні ігри [3]. Родоначальницею цього жанру ігор стала

гра «Magic: The Gathering», котра стала найбільш успішною та впізнаваною і зберегла свою актуальність навіть до сьогодні. Після неї, починаючи з 2010 року, в хід пішли добре відомі підліткам, зокрема, хлопчакам, такі колекції карток, як «Супергонки», «Людина-Павук: Герої та злодії», «Черепашки ніндзя: Бойова четвірка» та інші (див. Рис. 1.4).

Рис. 1.4. Колекційна карткова гра «Супергонки»

Здавалося б, що після стрімкого просування гаджетів даний жанр міг би зникнути, оскільки молодь надала перевагу технологіям, однак, як і будь-який інший жанр ігор, ККІ стали цифровими, подарувавши любителям даного жанру такі незрівнянні проекти, як «Hearthstone», «Gwent: The Witcher Card Game», «Yu-Gi-Oh!» та інші (див. Рис. 1.5).

Рис. 1.5. Цифрова колекційна карткова гра «Gwent: The Witcher Card Game»

Ознайомившись з коротким описом колекційних карткових ігор, варто також розуміти основні характеристики та механіки даного виду відеоігор. Перш за все, це, як не дивно, колекціонування. Гравці поповнюють свою колекцію карток шляхом покупок бустер-паків, обміну карток з іншими гравцями або виграючи унікальні картки у турнірах. Крім того, картки можуть мати різну рідкість (звичайні, рідкісні, дуже рідкісні тощо), що додає гравцям почуття азарту та унікальності.

Другою важливою характеристикою варто вважати побудову власної ігрової колоди. Гравці створюють свою унікальну колоду карт зі своєї колекції, налаштовуючи її під свої вимоги і тим самим створюючи свій власний стиль гри. Створення колоди вимагає стратегічного мислення, оскільки важливо враховувати баланс карток, їх спеціальних здібностей, синергії між ними.

Третім, ключовим, аспектом є саме механіка гри. Картки можуть мати різні типи: наприклад, істоти, заклинання, предмети тощо, – і це вносить у процес гри неабиякого різноманіття. Крім того, стратегії гри можуть

враховувати побудову комбінацій карт, контроль над полем, агресивну гру чи захисну тактику.

Четвертою унікальністю колекційних карткових ігор є ігровий процес. Зазвичай гра відбувається у форматі поєдинку між двома та більше гравцями, де кожен гравець використовує карти зі своїх колод для виконання атак, захисту, застосування спец здібностей та інший ігровий дій. Мета гри може варіюватися, проте зазвичай переможець повинен або зменшити очки життя противника до нуля, або залишитися з більшою чи меншою кількості карток у руці, або набрати більше очок в одному чи кількох раундах.

Як і будь-які ігри, ККІ також розвивалися, перейшовши на цифрові платформи, таким чином давши змогу гравцям змагатися один з одним з будь-якої точки світу. Крім того, на рівні з іграми жанру MOBA (Multiplayer Online Battle Arena або “Багатокористувацька онлайнова бойова арена”) або шутерів типу «Conter Strike 2» чи «PUBG», багато колекційних карткових ігор мають організовані турніри та чемпіонати, що стимулюють активний розвиток спільноти. Проте, зрештою, чи мали би шанс на існування колекційні карткові ігри, не маючи своєї спільноти чи фан-клубів? На мою думку, навряд, адже наявність великої кількості прихильників даного виду відеоігор, також відіграє суттєву роль у розвитку та процвітанні даного жанру.

Таким чином можна з упевненістю сказати, що ігрова галузь колекційний карткових ігор буде й надалі розвиватися, пропонуючи ще більше проектів з новими механіками, сетингами і можливостями для гравців. Адже вона залишається популярною завдяки своїй активній спільноті, чудовій стратегії, постійному оновленню контенту та іншими важливими аспектами.

РОЗДІЛ 2. ПРОГРАМНІ ЗАСОБИ ДЛЯ РЕАЛІЗАЦІЇ ВІЗУАЛЬНОЇ ЧАСТИНИ ГРИ

2.1. Опис карткової гри та постановка задачі

Основною ідеєю карткової гри є дуель між гравцями, котрі використовуватимуть картки професій, кожна з яких має свої параметри, такі як сила, витривалість, інтелект, рівень доходів та соціальність – головний параметр, за допомогою якого картки різних професій будуть взаємодіяти один з одним.

Метою гравця, задля отримання перемоги у партії, є збереження на своїй руці якнайбільшої кількості карток. Дана механіка передбачає те, що гравцям необхідно буде зважено ухвалювати рішення щодо того, котру картку потрібно буде викласти під час свого наступного ходу. Зважаючи на цей принцип, стає очевидним, що сильніша картка може перемогти одну або кілька слабших.

Усі події гри відбуватимуться на ігровому полі, котре матиме вигляд великого нічного міста-мільйонника: на верхньому, лівому та правому кордонах арени будуть розташовані хмарочоси, у вікнах котрих у випадковому порядку вмикатиметься та вимикатиметься світло, для створення ефекту живого міста. На нижній границі арени буде розташовано дисплей, котрий відображатиме основну інформацію під час гри. Середина ігрового поля буде зображено у вигляді дороги, на протилежних сторонах якої знаходитимуться тунелі, з яких будуть виїжджати автомобілі. Також всередині поля будуть присутні неонові лінії, котрі виконуватимуть роль освітлення. Що найцікавіше, ігрове поле буде розташоване на столі таверни, котра буде освітлюватися рожевим світлом.

Для успішної реалізації логічної частини гри було необхідним забезпечити відповідні умови, котрі складаються з ігрового поля – середовища гри, – та карток – об’єктів, з якими взаємодіятимуть гравці.

2.2. Інструменти для створення тривимірних об'єктів

2.2.1. Blender

Під час вибору програмного забезпечення для створення арени та гральних карток, було прийнято рішення обрати Blender, оскільки даний програмний пакет славиться не тільки широким спектром інструментів для тривимірного моделювання, а й здатністю до адаптації програми для виконання різноманітних задач.

Оскільки Blender є безкоштовним потужним програмним пакетом для створення тривимірних об’єктів, анімацій, рендерингу та створення візуальних ефектів, дане середовище набуло популярності серед аніматорів, дизайнерів та розробників завдяки своїм функціональним здібностям та активній спільноті користувачів, котрі часто пропонують ідеї щодо нововведень.

До списку основного функціоналу входять:

1. Моделювання:
    a. Полігональне моделювання призначене для створення моделей вершин, ребер та граней;
    b. Скульптинг: набір інструментів для структурування деталей 3D-моделей;
    c. NURBS (Non-uniform rational B-spline або "*Неоднорідний раціональний B-сплайн*") та Bezier криві для створення гладких органічних форм.
2. Анімація:
    a. Ключові кадри призначені для стандартної системи анімацій;
    b. Скелетна анімація: здатність створювати анімацій за допомогою кісткової структури (rigging);

c. Фізичні симуляції: динаміка твердих тіл, рідин, диму, волосся, тканин та інших поверхонь.

3. Рендеринг:
    a. Cycles: реалістичний рендер-рушій на основі визначення маршруту променів;
    b. Eevee: рендер-рушій реального часу, котрий дає змогу миттєво переглядати результати.
4. Матеріали та текстури:
    a. Node-based system для створення складних матеріалів текстур;
    b. UV-mapping для нанесення текстур на моделі.
5. Композиція: Вузлова система для пост-обробки рендерів, враховуючи ефекти, корекцію кольорів та композитинг.
6. Розширення та скрипти: Python API дозволяє автоматизувати завдання, створювати нові інструменти та розширення.
7. Ігровий двигун: Blender Game Engine (або BGE): двигун, котрий підтримував створення інтерактивних 3D-ігор. Однак, починаючи з версії 2.8 даний двигун було видалено, хоча, станом на сьогодні, існують сторонні розширення, призначені для інтеграції з іншими ігровими двигунами, включно з Unity, Unreal Engine, Armory3D та інших.

Окрім гнучкого та багато функціоналу даної платформи, не буде зайвим згадати про переваги Blender, до списку якого варто внести:

1. Безкоштовний доступ з відкритим кодом: Blender надає змогу користувачам вносити модифікації у програму, налаштовуючи її до власних потреб;
2. Повний інструментарій, котрий Blender пропонує не тільки 3D-моделювання, а й для текстурування, рендерингу, анімацій, симуляцій фізики, композитингу та багато іншого. Саме великий спектр інструментів робить його універсальним рішенням для різноманітних творчих завдань;

3. Висока якість рендерингу, котра включає в себе як Cycles (фотореалістичний рендер на основі трасування променів), так і Eevee (реалістичний рендер у реальному часі). Це дозволяє створювати високоякісні зображення та анімації моделей;
4. Велика кількість плагінів та доповнень, що дозволяють розширити функціональний потенціал та адаптувати програму до виконання конкретно поставлених завдань. Що цікаво, переважна частина скриптів була створена саме спільнотою Blender і доступні для використання безкоштовно;
5. Інтеграція з іншими програмами дозволяє підтримувати імпорт та експорт багатьох відомих 3D-форматів;
6. Гнучкість та налаштування інтерфейсу та інструментів Blender до необхідних потреб забезпечує високу продуктивність. Крім того, як згадувалося вище, користувачі мають змогу створювати власні скрипти за допомогою мови програмування Python, дозволяючи автоматизувати велику кількість процесів.

Враховуючи вище наведені функціональні можливості та переваги, стає очевидним той факт, чому саме Blender є одним з найбільш поширених програмних забезпечень та затребуваний серед розробників відео-ігор.

2.2.2. Cinema 4D

Після створення основних моделей гри, наступним кроком для створення візуальної частини гри стала реалізація ефектів. Для цього було обрано програмний пакет **Cinema 4D (або C4D)** – німецьке якісне та популярне програмне забезпечення для створення 3D-моделей та маніпуляцій з ними, такі як рендеринг, анімація тощо. Дане ПЗ відоме своєю зручністю, стабільністю та великим інструментарієм, що робить його популярним, зокрема, серед аніматорів та художників VFX (Visual effects або "*Візуальні*

ефекти"). На відміну від Blender, C4D особливо широко використовується в таких напрямках, як телебачення, реклама, дизайн та візуалізація.

Подібно до Blender, C4D має потужний функціонал для роботи з 3D-моделями, проте спостерігаються деякі значні відмінності, серед яких можна виділити:

1. Анімацію:
    a. Character Animation: пакет інструментів, розроблений конкретно для анімації персонажів, враховуючи rigging, IK/FK та morph targets;
    b. MoGraph: набір інструментів для створення процедурних анімацій та генеративного дизайну.
2. Рендеринг:
    a. Physical Render: фізично коректний рендер для отримання високоякісних зображень;
    b. ProRender: інтегрований GPU-рендер, котрий використовує технологію процесору AMD;
    c. Підтримка сторонніх рендерів, серед яких V-Ray, OctaneRender та Arnold.
3. Композицію та постробку:
    a. Інтеграція з After Effects надає змогу експортування проектів After Effects для подальшої оборбки;
    b. Інструментарій для композиції надає базові пакети інструментів для обробки рендерів C4D.
4. Симуляції:
    a. Динаміка твердих тіл: симуляція фізики для створення реалістичних рухів об'єктів;
    b. Симуляції рідин та тканин включають у себе інструменти для створення реалістичних рідин та деформацій тканин;

   c. Hair and Fur: інструменти для створення та анімації волосся та хутра.
5. Розширення та інтеграцію:
   a. Підтримка великої кількості плагінів надає змогу розширити функціональності C4D;
   b. Інтеграція з іншими програмами: C4D демонструє хорошу сумісність з Adobe Creative Suite, CAD-програмами та іншими 3D-пакетами.

Після розгляду відмінностей двох конкурентних ПЗ та ознайомлення з функціоналом C4D, варто було би також визначити переваги даного пакету, а саме:

1. Інтуїтивний інтерфейс, котрий робить програму доступною для новачків, а також забезпечує швидке та ефективне робоче середовище для досвідчених розробників;
2. Потужний інструментарій дозволяє використовувати набір інструментів для полігонального та процедурного моделювання, а для створення моделей складної геометрії та поверхонь;
3. MoGraph інструменти, що допомагають створювати складні анімації та ефекти на основі модулів та клонів. Потрібно зазначити, що це є особливо корисним для створення анімаційних графік та візуальних ефектів, присутніх на сцені;
4. Cinema 4D славиться своєю стабільністю та високим рівнем продуктивності, що робить дане ПЗ надійним вибором для великих проектів;
5. Активна підтримка та навчальні матеріали, що створюють Maxon, розробники Cinema 4D, забезпечує регулярні оновлення, виправлення помилок та нововведення. Крім того, існує велика кількість навчальних ресурсів для роботи з Cinema 4D, зокрема офіційні підручники, курси і навіть онлайн-спільноти.

Аналізуючи функціональність та переваги Cinema 4D, можна дійти висновку, що дане програмне забезпечення є більш поширеним, оскільки функціонал програми дозволяє працювати не тільки в сфері ігрової індустрій, а й у кіно, телебаченні, рекламі тощо.

2.2.3. Порівняння та аналіз програм для створення тривимірних об’єктів

Під час ознайомлення та роботи з програмними пакетами для тривимірного моделювання Blender та Cinema 4D, було проведено аналіз, котрий допоміг мені порівняти два конкурентних середовища розробки та звернути увагу не тільки їх переваги, а й провести деяку паралель для визначення їх ключових відмінностей.

Перш за все, для зручності порівняння, потрібно зауважити, що відмінності між Blender та Cinema 4D можна поділити два основні типи: функціональні та нефункціональні. Як правило, функціональні відмінності стосуються конкретного функціоналу, котрий забезпечує роботоздатність програми. Вони описують поведінку системи, її функціональність та взаємодію з користувачем чи іншими системами. До функціональних відмінностей належать наступні явища: функції та можливості (дії, виконувані програмним забезпеченням), бізнес-правила (правила та логіка, що використовуються у бізнес-процесах), інтерфейс користувача (спосіб взаємодії з системою користувачем), інтеграція (спосіб взаємодії з системою іншими системами та/або службами, наприклад API). Зі свого боку нефункціональні відмінності стосуються загальних характеристик системи, що відповідають за її якість і ніяк не пов’язані з конкретними функціями, проте впливають на її експлуатацію. Тому до нефункціональних відмінностей належать наступні характеристики: продуктивність (швидкість роботи), надійність (стабільність роботи), масштабованість (ефективність роботи програми при збільшенні навантаження), безпека (забезпечення

конфіденційності), зручність використання (легкість роботи), підтримуваність (рівень обслуговування програмного забезпечення), сумісність (здатність програми працювати на різних платформах та взаємодіяти з іншими системами) тощо.

Таким чином, збагнувши основну різницю між функціональними та нефункціональними відмінностями (або вимогами), визначити розбіжності між Blender та C4D буде простіше.

Перш за все, пропонується розглянути узагальнення щодо функціональні та нефункціональні особливостей Blender. Варто почати з моделювання: Blender надає дуже потужний набір інструментів для моделювання з великою кількістю функцій, враховуючи скульптинг, ретопологію, інструменти для роботи з геометрією та багато інших. Наступним в черзі буде рендеринг: Blender має два різних вбудованих рендер-двигунів: Cycles, котрий забезпечує високоякісні рендери, проте може бути повільним, та Eevee, що пропонує швидкі результати з високою якістю. Третьою особливістю є процес анімації: Blender пропонує повний набір інструментів для анімації, зокрема ріггінг, кінематику, анімацію частинок та фізику [2]. Крім того, Blender також має потужні можливості для роботи з кістками та контрольними об’єктами, що є вагомою перевагою під час створення анімацій ігрових персонажів. Останнім в черзі функціональної частини Blender можна вважати скриптинг та розширюваність: Blender використовує лише бібліотеки Python для скриптингу, що дозволяє створювати власні інструменти та автоматизувати робочі процеси, адаптуючи середовище для власних потреб.

Що стосується нефункціональних вимог, варто почати з інтерфейсу ПЗ: Інтерфейс Blender може здатися складним для новачків за рахунок великої кількості різноманітних функцій, налаштованих для виконання окремих завдань, проте це компенсується високою гнучкістю і налаштовуванням під потреби користувача. Наступним пунктом є спільнота та підтримка: Blender

має велику спільноту з безліччю безкоштовних ресурсів, уроків та форумів, доступних у вільному доступі, таким чином даючи змогу новим користувачам пристосуватися до роботи з цим програмним забезпеченням. Останньою, ключовою відмінністю програмного середовища є його ціна: Blender – це абсолютно безкоштовний та відкритий продукт з відкритим кодом. Це є великою перевагою, за рахунок якої Blender можна вважати одним з найпоширеніших ПЗ для створення тривимірних об’єктів.

Тепер пропоную узагальнити функціональні та нефункціональні особливості Cinema 4D. Для реалізації моделювання дане ПЗ також має розвинені інструменти, але більше орієнтований на простоту виконання і швидкість створення моделей. Порівняно з Blender, Cinema 4D має менш потужні інструменти для скульптингу. Також, Cinema 4D, на відміну від Blender, має лише один власний рендер-двигун, хоча також реалізована підтримка сторонніх двигунів як Redshift, Octane та інші. Зважаючи на сильну інтеграцію з Redshift, процес рендерингу забезпечує високоякісні та швидкі рендери, що помітно спрощує та пришвидшує процес розробки. Крім того, Cinema 4D більш сильний в області моушн-графіки за наявності зручних інструментів для створення анімаційних ефектів та титрів. C4D також підтримує ріггінг та кінематику, але часто вважається менш потужним для складних анімацій персонажів, ніж Blender. Останньою відмінністю від Blender є той факт, що Cinema 4D використовує як бібліотеки Python, так і бібліотеки C++ для розширюваності, що дозволяє створювати потужні плагіни та інструменти, ефективно адаптуючи ПЗ для виконання конкретних вимог та завдань розробника.

Після визначення функціональних розбіжностей варто також звернути увагу і на нефункціональні відмінності C4D. Варто почати з користувацького інтерфейсу. На відміну від Blender, Cinema 4D неабияк відомий своїм інтуїтивно зрозумілим, чітким та дружнім інтерфейсом, що відображається на його популярності. Також ПЗ підтримується комерційною компанією Maxon,

що надає офіційну технічну підтримку. Проте C4D також володіє власною активною спільнотою користувачів, проте помітно менших масштабів. Що стосується ціна, Cinema 4D – це комерційне програмне забезпечення, для досконалої роботи з яким потрібно оформлювати підписку або здійснити одноразову покупку, що може бути сильно вдарити по кишені користувачу. І, нарешті, Cinema 4D на фоні Blender частіше вважається легшим для вивчення та використання через простий та дружній інтерфейс.

Таким чином, провівши аналіз та порівняння між Blender та Cinema 4D, можна дійти простого та дуже очевидного висновку: обидва програмних пакета є сильними за рахунок функціональної частини, що включає в собі усі необхідні інструменти для виконання тих чи інших вимог розробника. Як і завжди, вибір між двома однаковими ПЗ дуже часто залежить від конкретних потреб проекту, бюджету, а також власних вподобань, однак, на мою думку, Blender краще підійде для тих, хто бажає опанувати мистецтво 3D-розробки, а Cinema 4D – для тих, хто бажаю заглибитися у сферу кіно індустрії або стати досконалим аніматором.

2.3. Ігровий рушій Unity

Останнім етапом, що стосувався розробки візуальної частини карткової гри було перенесення готових ресурсів у Unity – середовище для розробки ігор та інтерактивних додатків на основі однойменного унікального рушія, підтримуючи такі мови програмування як C++ та C Sharp (C#). Даний рушій є популярним як серед маленьких інді-компаній, так і серед великих студій для створення ігор для різних ігрових платформ.

Дане програмне середовище було обране через гнучкість та простий, проте ефективний набір функцій, котрі забезпечують розробника наступним функціоналом:

1. Мультиплатформенність: Unity дозволяє розробляти ігри та додатки для великої кількості платформ, зокрема Windows, macOS, Linux,

iOS, Android, ігрові консолі, такі як PlayStation, Xbox, Nintendo Switch, а також пристрої віртуальної та доповненої реальності (VR/AR);

2. Потужний редактор, який спрощує процес створення та налаштовування сцен, об’єктів, анімацій, матеріалів. Особливо зручним редактором Unity робить візуальне програмування, а також інтеграція з багатьма сторонніми інструментами;
3. Використання системи компонентів дозволяє легко додавати функціональність до об’єктів, забезпечуючи гнучкість та модульність під час розробки;
4. Підтримка 2D та 3D розробки, пропонуючи конкретні інструменти для створення анімацій, фізики та графіки;
5. Фізичні рушії, що підтримуються Unity (а саме Nvidia PhysX для 3D-розробки і Box2D для 2D-розробки), та інструменти для анімації дозволяють створювати складні рухи та симуляції;
6. Unity Asset Store – бібліотека ресурсів, котра доповнюється як розробниками Unity, так і членами спільноти – пропонує велику кількість готових шаблонів та ресурсів, зокрема моделей, текстур, звуків, скриптів чи плагінів, котрі можна використовувати для пришвидшення розробки проекту;
7. Скриптування за допомогою мов програмування C# та C++ забезпечує великі можливості для розробки логіки гри та поведінки об’єктів на полі;
8. Інструменти для інтеграції з хмарними сервісами, як от Unity Cloud Build, Unity Analytics та Unity Multiplayer, що дозволяє ефективно створювати багатокористувацькі ігри та збирати дані про використання додатків.

Зважаючи на функціонал рушія Unity, можна виокремити великий ряд переваг, котрий включатиме:

1. Легкість освоєння: Завдяки великій кількості навчальних матеріалів та простому інтерфейсу, новачки у сфері розробки ігор можуть швидше освоїти Unity та почати створення власних проектів;
2. Гнучкість та модульність: Оскільки система компонентів дозволяє розробнику створювати модульні ігри, це спрощує процес розробки та тестування проектів;
3. Масштабованість: Як згадувалося раніше, даний рушій ідеально підходить не тільки для маленьких інді-компаній, а й для велетнів ринку ігрової індустрії, забезпечуючи масштабованість та стабільність;
4. Велика спільнота розробників, доступ до форумів, підручників та інших навчальних ресурсів забезпечують підтримку і допомогу у вирішенні будь-яких проблем, котрі можуть виникнути під час вивчення та практики. Крім цього, заважаючи на останні події в Україні, не буде зайвим зазначити, що Unity підтримала Україну під час агресії РФ, про що повідомила на своєму офіційному форумі;
5. Регулярні оновлення, котрі часто випускає Unity Technologies, додаючи нові функції та покращуючи раніше готовий функціонал, робить платформу актуальною та сучасною;
6. Чудова сумісність проектів Unity з проектами інших відомих інтегрованих середовищ розробки, таких як Photoshop, Blender, Visual Studio та інших.

Важливими аспектами Unity, під час дослідження та виконання поставлених задач, було зауважено велику кількість матеріалу для вивчення та вдосконалення раніше здобутих знань, а також чудова сумісність іншого використаного програмного забезпечення Blender, хоч і виникали деякі труднощі на певних етапах реалізації гри.

2.4. Використання штучного інтелекту для генерації зображень

Наступним аспектом візуальної частини гри були зображення, котрі будуть вказувати на ту чи іншу професію. Спочатку було важливим задати основний зовнішній вигляд зображення, спираючись на стиль та тематику гри. Оскільки стиль гри повинен відображати світ майбутнього, було прийнято рішення використовувати кольорову гаму з рожевих та синіх відтінків з неоновим акцентом спереду та темними відтінками на задньому фоні. Також варто зазначити, що картки повинні мати не векторну графіку, як це прийнято сьогодні, а піксельну. Це повинно надати грі атмосфери старих піксельних ігор на аркадних ігрових автоматах, що широко використовувалися та набули пік своєї популярності з кінця 1970-х – середини 1980-х років. Таким чином зображення мусило відображати поєднання майбутнього та минулого.

Після створення кількох зразків зображень вручну, використовуючи графічний планшет, з’явилася ідея щодо використання штучного інтелекту, зокрема ChatGPT, для генерації зображень. Наявність вже готових прикладів вплинула на швидке реагування нейромережі на вказаний запит, внаслідок чого ШІ зміг згенерувати зображення, подібні за стилем на зарання створенні картинки (див. Рис. 2.1).

Рис. 2.1. Приклад зображення картки, що відображає професію «Суддя»

2.5. Використання бази даних для збереження відомостей карток

Останньої вимогою, котру було необхідно було забезпечити – збереження даних карток, враховуючи їх показники характеристик та зображення. Таким чином було створено просту базу даних, котра містила у собі дві основні таблиці: Card та NfcCard.

Таблиця з назвою Card зберігає у собі відомості про конкретну картку, а саме її ідентифікатор, назву та параметри: силу, витривалість, інтелект, дохід та соціальність.

Основною інформацією, котру містить таблиця NfcCard, є власний ідентифікатор, ідентифікатор ігрової картки, та параметр UID – ідентифікатор NFC-картки. Створення даної таблиці було необхідним для того, щоб "прив’язати" до NFC-картки відомості про ту чи іншу картку в таблиці, використовуючи її в майбутньому без повторного "прив’язування" (див. Рис. 2.2).

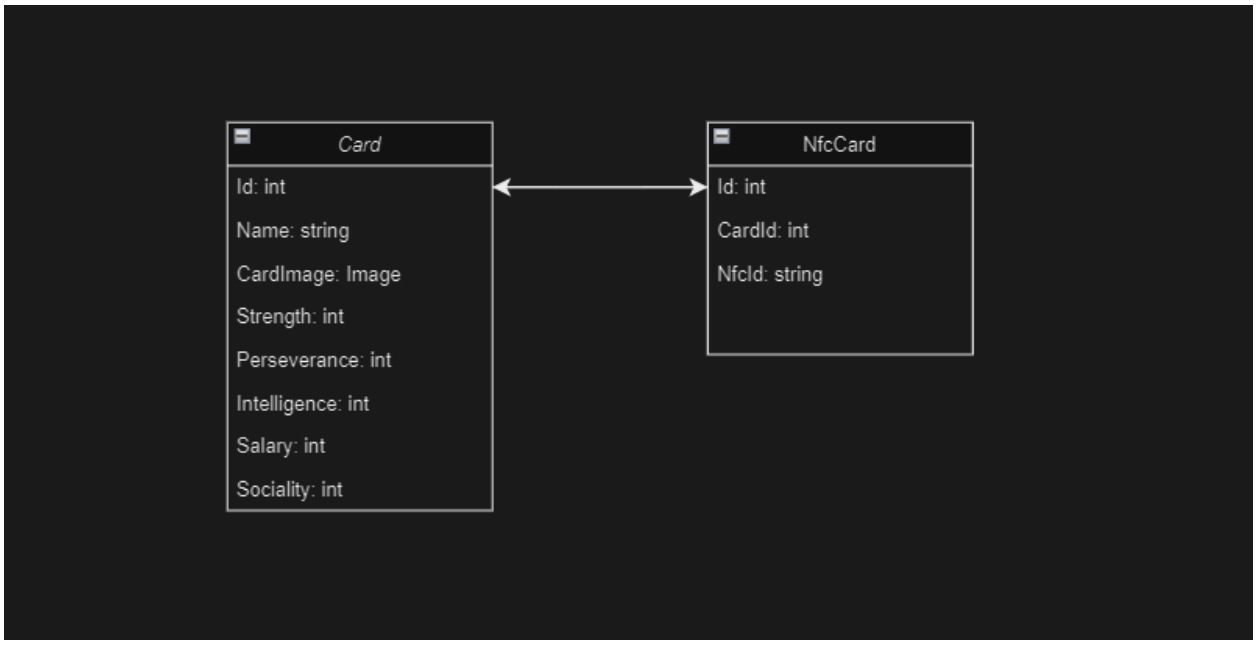


Рис. 2.2. Діаграма бази даних проекту

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РОЗРОБКА ВІЗУАЛЬНОЇ ЧАСТИНИ ГРИ

3.1. Реалізація моделей для гри

Для реалізації функціональної частини гри було важливим спочатку створити основні елементи ігрового середовища, тобто ресурси, з котрими в подальшому можна було б працювати. Тому, першим елементом, котрий необхідно було створити, стала сцена для гри.

Даний об’єкт було створено з куба, котрому спочатку було задано форму у вигляді поля. На ніжній стороні поля було вбудовано місце для дисплею, котрий в майбутньому зобов’язаний виводити деяку інформацію під час гри (див. Рис. 3.1).

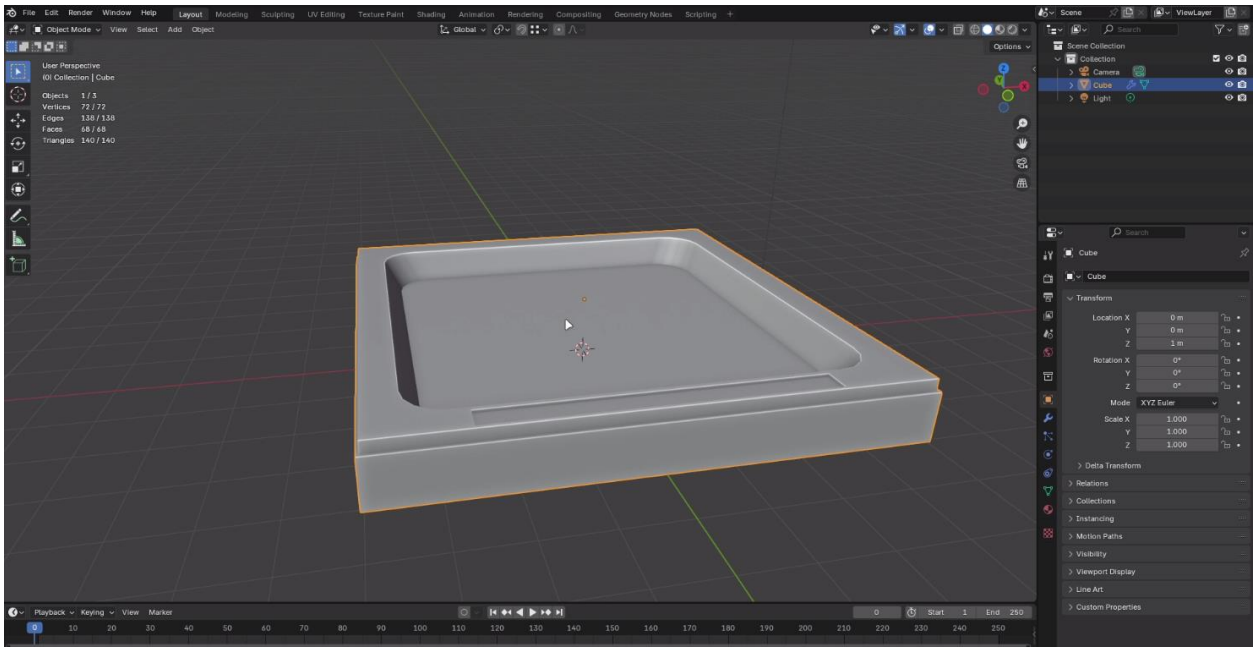


Рис. 3.1. Початковий вигляд ігрового поля

Після створення каркасу поля, наступним кроком стало текстурування сцени. Даний етап був дещо важчим, оскільки потрібно було не тільки зробити UV розгортку – процес розгортання 3D-об’єкта на 2D площину, з метою надання та налаштування майбутніх текстур. На рисунку 3.2. зображено початкову розгортку. Як можна побачити, вона є недосконалою, оскільки

середовище Blender сприйняло сцену, як куб. Тому, аби зробити розгортку ігрового поля досконалою, у Blender потрібно було додати аддон – додатковий модуль – з назвою TexTools, котрий дає змогу користувачу налаштувати не тільки розгортку об’єкта, а й його деталізацію під час накладання текстур. Таким чином було створено ручну розгортку (див. Рис. 3.3).

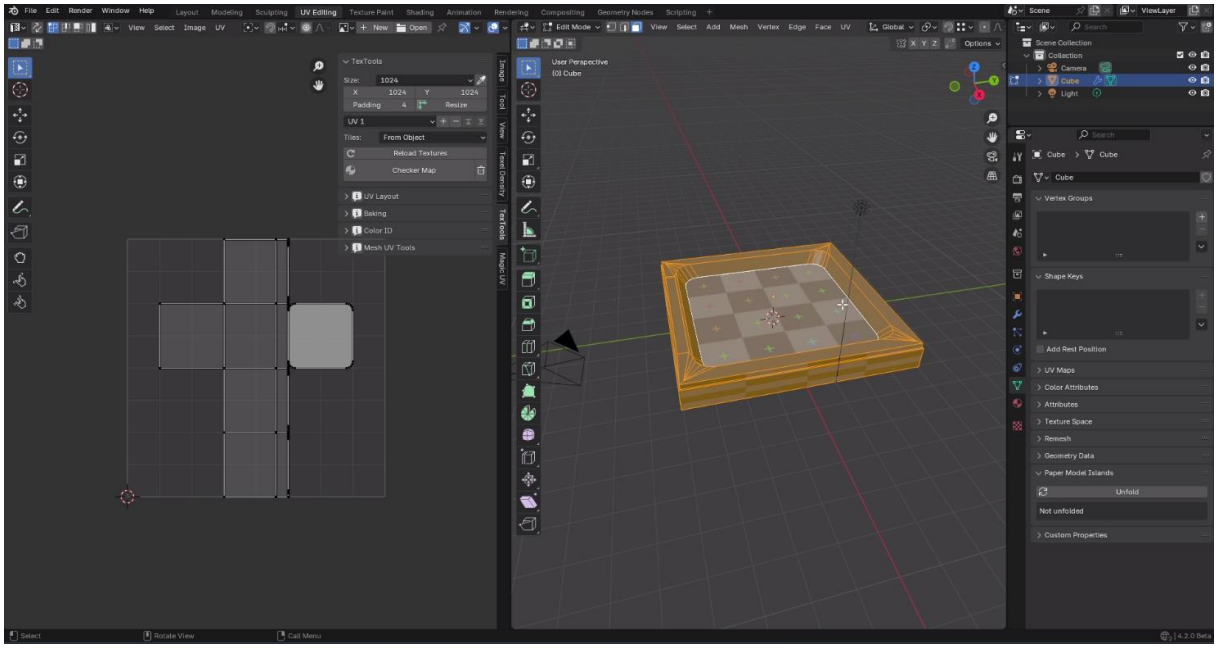


Рис. 3.2. Початкова розгортка об’єкту-сцени

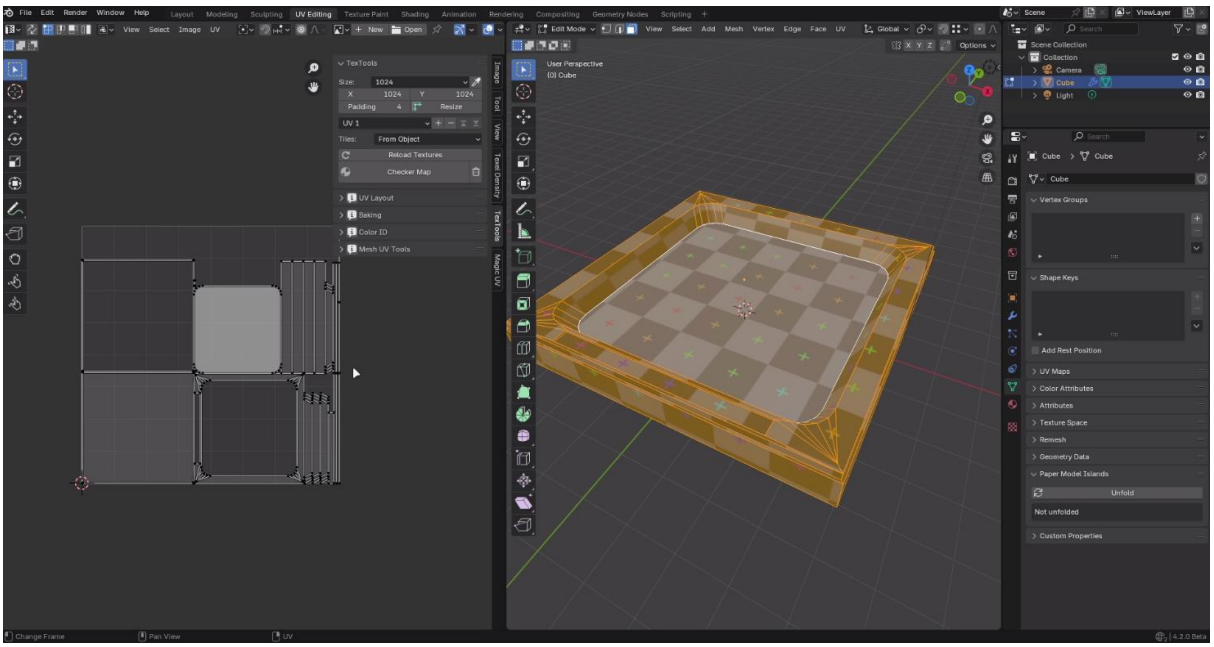


Рис. 3.3. Результат розгортки об’єкту вручну

Наступним кроком після виконання розгортки об’єкту стало безумовно його текстурування. Для того, щоб зробити текстури реалістичними та досконалими, було необхідно встановити ще один пакет доповнень – TexelDensity. Даний пакет надає можливість користувачу працювати з текселями – найменшими одиницями інформації, з котрої складається текстура [4]. Фактично, тексель виконує ту саму роль, що і піксель, однак між ними є деяка різниця – один піксель монітору може бути визначений багатьма текселями і навпаки, один тексель може покривати декілька пікселів. Порівняно з результатом автоматичної деталізації об’єкту (див. Рис. 3.3), після роботи з текселями кількість полігонів на ігровому полі помітно збільшилася. Дана процедура надала змогу зробити текстуру об’єкту більш досконалою та реалістичною (див. Рис. 3.4).

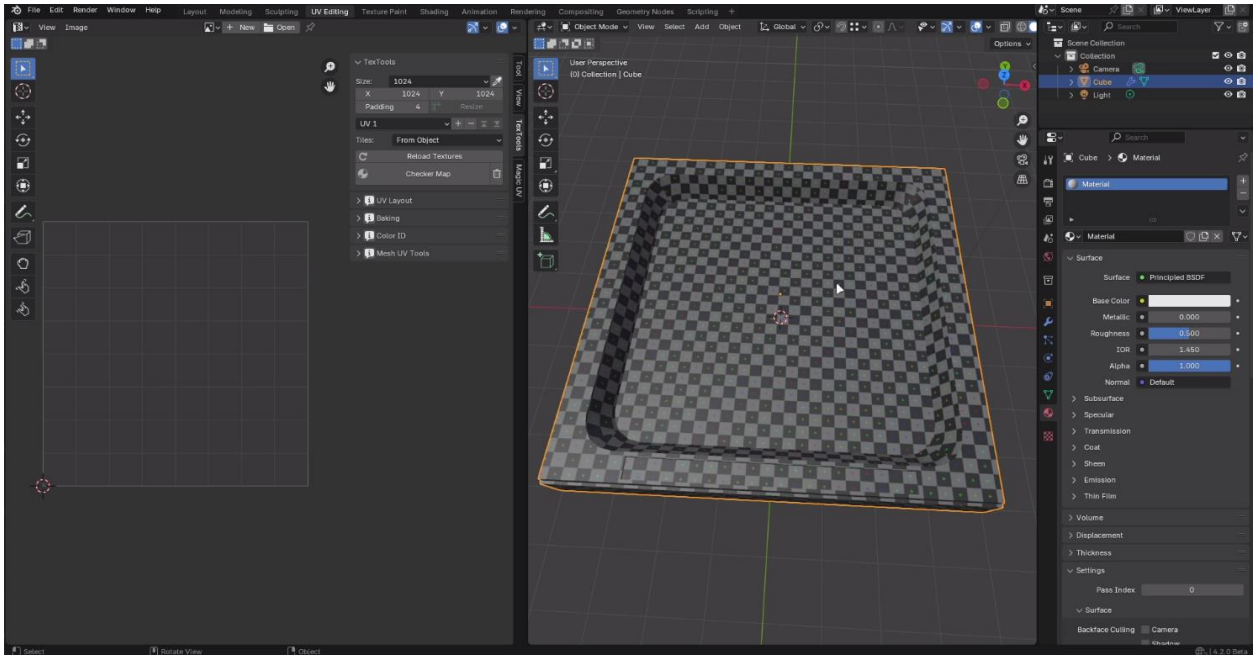


Рис. 3.4. Налаштування деталізації об’єкту

Для того щоб накласти текстуру на об’єкт рівномірно, необхідним кроком став його поділ по центру. Після цього, в центрі поля було накладено текстуру дороги. Однак більша частина сцени залишилася без текстури, що потрібно було виправити. Рішенням даної проблеми стало накладання текстур

асфальту та кам’яної поверхні. Два окремих матеріали було об’єднано в один, за допомогою візуальної скриптової мови – Blueprints (див. Рис. 3.5).

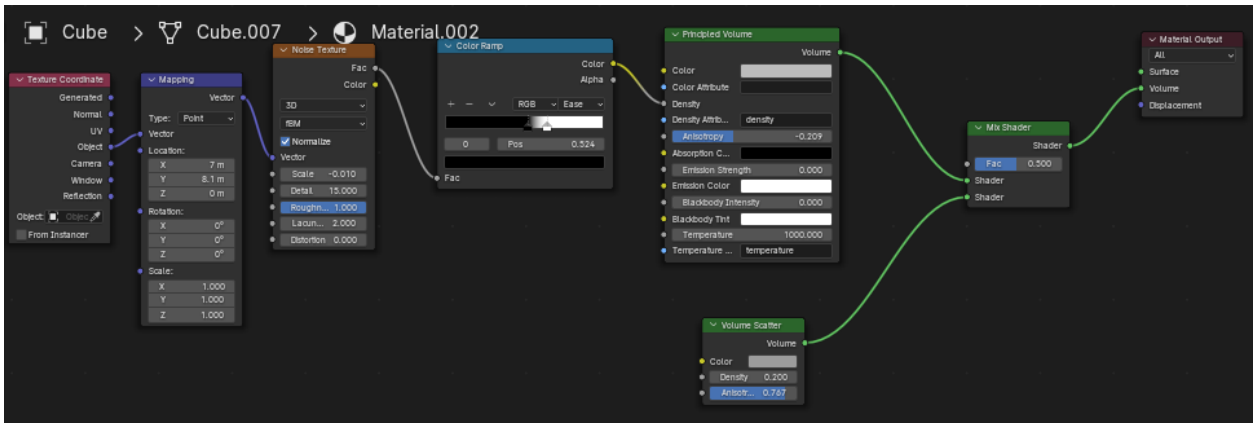


Рис. 3.5. Приклад Blueprints в програмному середовищі Blender

Після завершення текстурування ігрового поля, його необхідно було заповнити другорядними об’єктами, котрими стали будинки. Building assets pack – це пакет об’єктів, котрий було створено спільнотою Blender для некомерційного використання. Головною проблемою даних об’єктів стало те, що вони складалися з великої кількості полігонів, що негативно впливало на рендеринг проекту, тобто можливі “зависання” під час гри. Дану проблему було вирішено так званим “запіканням” фігур. Моделі будинків пізніше було додано на ігрове поле у випадковому порядку за допомогою функції Particle System – системи частинок, котрі пізніше налаштовувалися для генерації об’єктів, а наприкінці – налаштовувалися вручну.

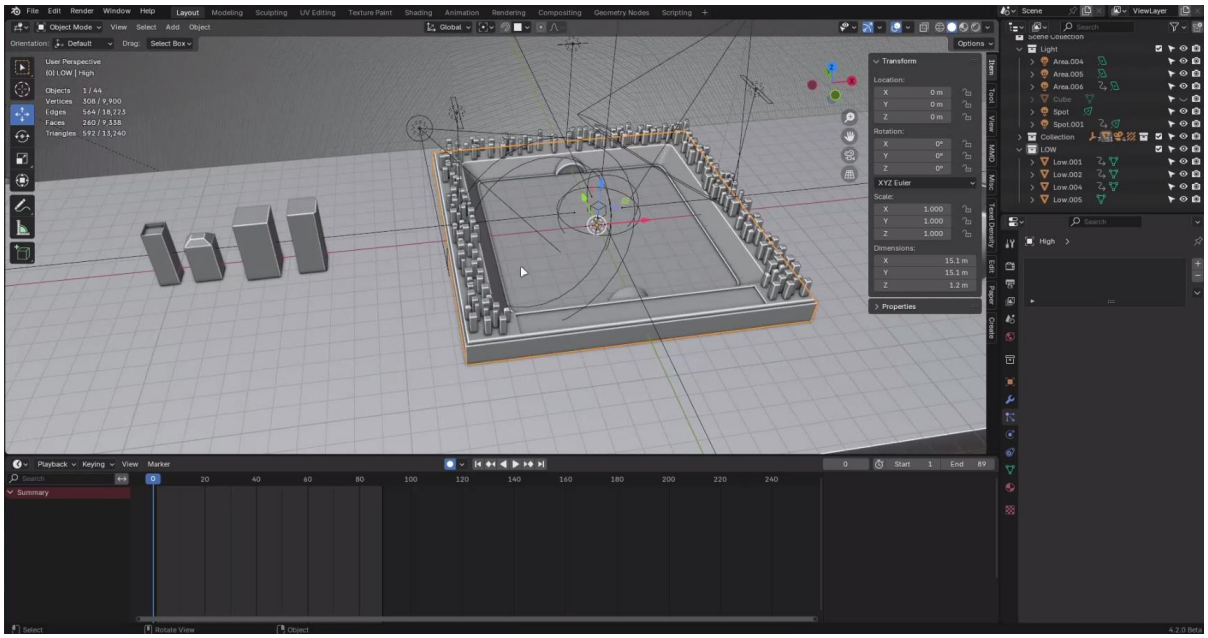


Рис. 3.6. Результат генерації будинків на ігровому полі.

Останньою частиною створення ігрового поля стала реалізація HDRI та освітлення. HDRI (High Dynamic Range Image) – це панорамна фотографія, котра охоплює всі кути зору з однієї точки і містить велику кількість даних, який може бути для освітлення сцени.[5] Її було створено в Blender заздалегідь, використовуючи іншу HDRI з додаванням на неї інших об’єктів, таких як бочки, чаші, стіл тощо, для створення ефекту перебування у таверні. Крім того, було застосовано ще один матеріал у вигляді кола, щоб гравець міг бачити середовище гри у 3D-вигляді. Після налаштування HDRI також було додано освітлення зверху, а також неонове світло по кругу ігрового поля. Фінальний вигляд загальної ігрової сцени та картки, готових до перенесення з Blender у Unity, зображено на рисунках 3.7. та 3.8.

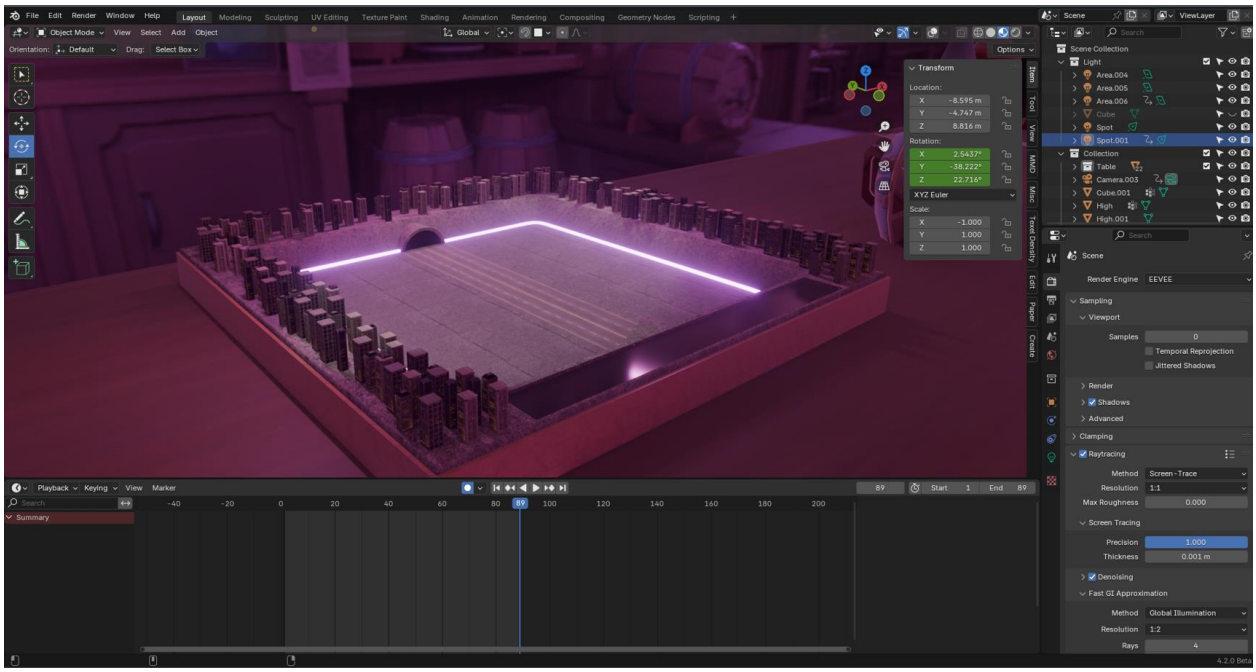


Рис. 3.7. Вигляд завершеної ігрової сцени

Рис. 3.8. Вигляд створеної картки, котра перебуває на сцені

3.2. Перенесення ресурсів з Blender у Unity

Під час експортування моделей у Unity було помічено дві основних проблеми: неправильне відображення текстур та їх зникнення, а також ефект “розпаду” об’єкта. Проблема з розсипанням об’єкту була зумовлена тим, що

наш об’єкт має велику кількість полігонів. Оскільки Unity робить більшими за розмірами файли розширення .blend, необхідно було або змінити тип об’єкту, або об’єднати його в одне ціле. Аби якісно здійснити експорт моделей з Blender на платформу Unity, було важливим знову його “запекти”, тим самим зменшивши кількість полігонів.

Загалом, перенесення об’єктів з Blender у Unity проходить у два етапи: перенесення об’єкту та перенесення текстур. Якщо з перенесення об’єкту усе стає очевидним, залишається лише правильно налаштувати текстури. Зважаючи на те, що під час перенесення об’єкту можуть бути відсутні матеріали, буде доцільно створити нові матеріали у самому середовищі Unity.

Важливо розуміти й те, що Unity може імпортувати в себе не всі моделі. До списку того, що розробник може імпортувати у Unity з інших проектів входять:

1. Усі зв’язки зі станами, обертаннями та масштабом;
2. Меші з вершинами, полігонами, трикутниками, UV та нормалями;
3. Кістки (під час створення персонажів);
4. Skinned меші (меші з прив’язкою до кісток);
5. Анімації, реалізовані у будь-якому іншому середовищі.

Щоправда, якщо моделі переносяться з самого Blender, варто зважати й на умови, передбачені Unity:

1. Версія Blender повинна бути не старішою, ніж 2.60, адже в деяких з більш ранішніх версій було зламано експорт FBX;
2. Текстури та колір diffuse неповинні назначатися автоматично. Їх потрібно назначити вручну, перетягуючи текстуру на меш у вікні Scene в Unity.

Зважаючи на вимоги, зазначені вище, та розуміння проблем, котрі можуть виникнути в ході процесу експортування, користувач зможе легко перенести усі необхідні об’єкти як з Blender у Unity, так і навпаки.

3.3. Реалізація анімацій в Unity

Система анімацій в Unity дозволяє створити чудово анімоване середовище. Вона підтримує блендинг, мікшуваня, складання анімацій, синхронізацію циклу, анімовані шари, а також засновані на фізиці rag-dolls. Створення анімованого об’єкту включає в себе два аспекти – переміщення у просторі сцени та відповідна анімація.

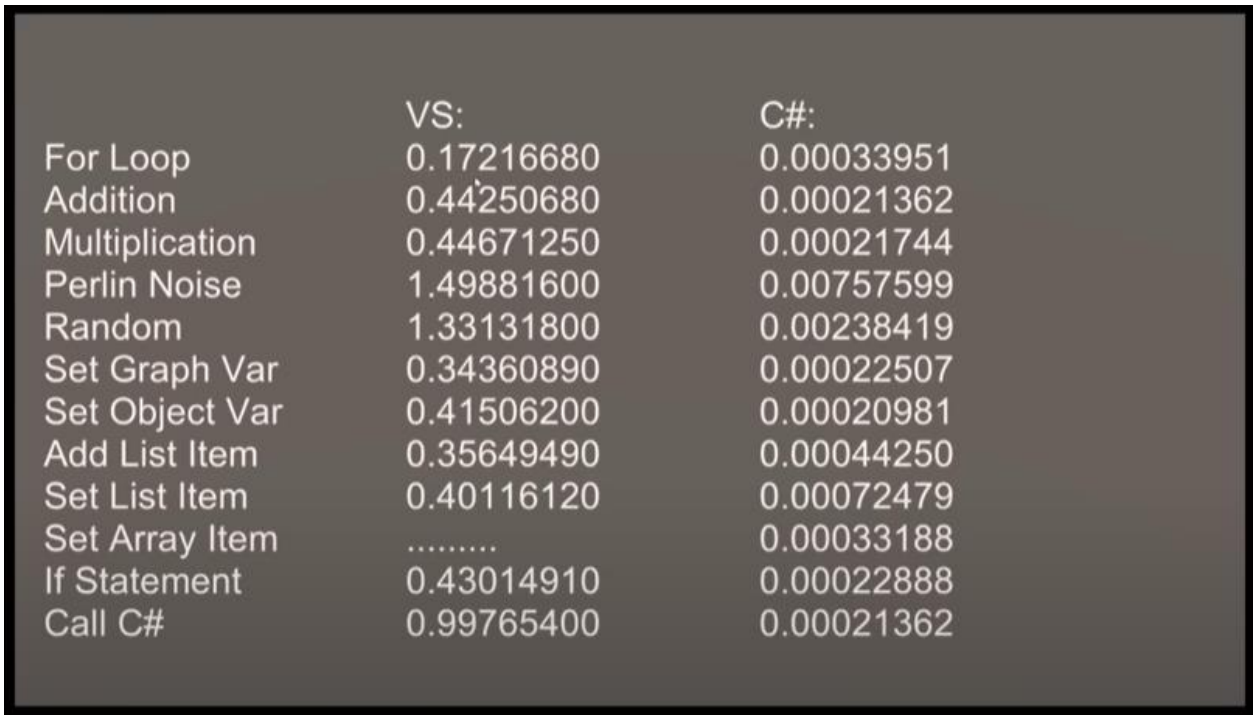
Створення анімацій за допомогою коду може бути виконане за допомогою системи “Animator” та компоненту “AnimatorController”. Алгоритм створення анімацій виглядає наступним чином:

1. створити анімаційного контролера;
2. реалізація анімації;
3. налаштування анімаційних перехідних станів (додавання станів та переходів);
4. реалізація параметрів для контролю анімацій;
5. додавання коду для контролю анімацій, написаного мовою програмування C#;
6. налаштування сцени, додаючи моделі та прикріплюючи скрипт до потрібного об’єкту.

Згідно даного алгоритму постановки задач, було створено кілька анімацій: появи картки на ігровому полі у вигляді вислизання її “з руки” гравця, підрахунок очків сильнішої картки по карті та інше.

Окрім прописання скриптів, розробник може також скористуватися візуальним аніматором, що містить у собі наперед заготовленні функції. Це значно спрощує процес розробки логіки гри. Однак даний спосіб має свої недоліки, серед яких, перш за все, варто згадати відсутність оптимізації, а також великий набір з’єднань між блоками, котрі відповідають за той чи інший функціонал. В деяких випадках для реалізації логіки, котру можна описати двома-трьома рядками коду, потрібно створити окреме дерево з десятків

блоків, що не тільки ускладнює процес реалізації функціональної частини гри, а й ускладнює пошук проблем, якщо вони виникають. Варто загадати також про те, що функціональні блоки налаштовані не повністю, через що, під час тестування продукту, випливають деякі недоліки. Що стосується оптимізації, на рис. 3.9 показано таблицю, в якій порівнюється швидкість виконання візуального аніматора та скрипту. Як можна помітити, скрипти прописані мовою C#, у нашому випадку, працюють значно швидше, ніж анімація за допомогою візуального аніматора.



	VS:	C#:
For Loop	0.17216680	0.00033951
Addition	0.44250680	0.00021362
Multiplication	0.44671250	0.00021744
Perlin Noise	1.49881600	0.00757599
Random	1.33131800	0.00238419
Set Graph Var	0.34360890	0.00022507
Set Object Var	0.41506200	0.00020981
Add List Item	0.35649490	0.00044250
Set List Item	0.40116120	0.00072479
Set Array Item	.....	0.00033188
If Statement	0.43014910	0.00022888
Call C#	0.99765400	0.00021362

Рис. 3.9. Таблиця для порівняння швидкості роботи візуального аніматора та скриптів, реалізованих на мові програмування C#

3.4. Створення базових скриптів

Для остаточної реалізації візуальної частини карткової гри було запропоновано створити наступні анімації: ефект вислизання карти на ігрове поле, підрахунок та порівняння очків сильнішої картки.

На рис. 3.10 зображено скрипт, котрий відповідає за анімацію появи картки на ігровому полі. Крім того, як можна побачити, у коді реалізовано

метод, що передбачає ініціалізацію картки за допомогою NFC-читача. Тобто у випадку, якщо зчитувач не ініціалізує картку, анімації не відбудеться.

```
using UnityEngine;
using System.IO.Ports;

public class CardController : MonoBehaviour
{
    private Animator animator;
    private bool isSlidingOut = false;
    private SerialPort nfcReader;

    void Start()
    {
        //Прикріплення компоненту Animator до картки
        animator = GetComponent<Animator>();

        // Ініціалізація серійного порту для NFC-читача
        nfcReader = new SerialPort("COM3", 9600);
        nfcReader.Open();
        nfcReader.DataReceived += OnNFCDataReceived;
    }

    void OnDestroy()
    {
        // Закриття серійного порта при знищенні об'єкта
        if (nfcReader != null && nfcReader.IsOpen)
        {
            nfcReader.Close();
        }
    }

    private void OnNFCDataReceived(object sender, SerialDataReceivedEventArgs e)
    {
        string nfcData = nfcReader.ReadLine();
        if (!isSlidingOut)
        {
            // Запуск анімації "SlideOut"
            animator.SetTrigger("SlideOutTrigger");
            isSlidingOut = true;
        }
    }

    void Update()
    {
        // Опціональний контроль через клавішу пробіл
        if (Input.GetKeyDown(KeyCode.Space) && !isSlidingOut)
        {
            // Запуск анімації "SlideOut"
            animator.SetTrigger("SlideOutTrigger");
            isSlidingOut = true;
        }
    }
}
```

Рис. 3.10. Скрипт анімації появи картки на ігровому полі “з руки” гравця

Після створення анімації вислизання картки, потрібно було забезпечити ефект живого міста на ігровому полі. Для цього було прийнято рішення реалізувати анімацію для моделі автомобілів, котрі будуть їхати дорогою з одного краю сцени, до іншого. Оскільки на ігровому полі з двох боків було створено два тунелі – в’їзд та виїзд – перед написанням анімації потрібно було

реалізувати модель автомобіля. Одну з моделей автомобілів було безкоштовно завантажено у Unity Assets Store, після чого було створено скрипт, у котрому була прописана деяка логіка: автомобіль повинен був з'являтися на початку шляху (точка А), котрим став тунель з боку гравця, та мусив зникати, як тільки він торкнеться кінця шляху (точка Б), котрим був тунель навпроти. Таким чином було реалізовано три зміни: точка А, точка Б та швидкість руху моделі.

Скрипт даної анімації був нескладним, оскільки усе, що було необхідним – оновлювати місцезнаходження моделі на площині. Дану логіку було налаштовано у функції Update(), а задання початкового місцезнаходження автомобіля, тобто його початок шляху, – у методі Start (). Щоразу, як тільки модель зрушувалася з місця, за допомогою умовного блоку if було реалізовано перевірку того, чи не досяг автомобіль кінцевої точки. Проте, так як одного автомобіля на карті буде мало, стало важливим реалізувати респавн моделі на її початковій точці. Готовий результат скрипту можна побачити на Рис. 3.11.

```
using UnityEngine;

public class CarMovement : MonoBehaviour
{
    public Transform pointA;
    public Transform pointB;
    public float speed = 5.0f;

    private Vector3 targetPosition;

    void Start()
    {
        // Задання початкової точки (перший тунель)
        transform.position = pointA.position;
        targetPosition = pointB.position;
    }

    void Update()
    {
        // Рух автомобіля до кінцевої точки (другий тунель)
        transform.position = Vector3.MoveTowards(transform.position, targetPosition, speed * Time.deltaTime);

        // Перевірка, чи досяг автомобіль точки Б
        if (Vector3.Distance(transform.position, targetPosition) < 0.1f)
        {
            // Зникнення автомобіля при досягненні точки Б
            gameObject.SetActive(false);

            // Респаун автомобіля на точці А (опціонально)
            // transform.position = pointA.position;
            // gameObject.SetActive(true);
        }
    }
}
```

Рис. 3.11. Скрипт анімації руху автомобіля на ігровому полі.

Оскільки ігрове поле було повністю готове, залишалося створити анімацію для карток, а саме анімацію порівнювання параметрів. Було прийняте рішення створити просту анімацію спаду чисельних значень карток (від 99 до 0, залежно від параметрів карток).

Першим, що потрібно було забезпечити – ініціалізація картки та її параметрів, при чому дані повинні були підтягуватися зі заздалегідь приготованої бази даних. Було створено клас Card, атрибутами якого стали параметри карток – сила, витривалість, кмітливість, рівень доходів та соціальність. Після цього було реалізовано метод AnimateParameterDecrease () – головну функцію даної анімації. Всередині методу було описано функціонал, що відповідав за оновлення параметрів картки, а також їх перевірка на від'ємні числа (див. Рис. 3.12).

```
19          // Оновлення параметрів карточки
20          switch (parameterName)
21          {
22              case "strength":
23                  strength -= decreaseAmount;
24                  break;
25              case "perseverance":
26                  perseverance -= decreaseAmount;
27                  break;
28              case "intelligence":
29                  intelligence -= decreaseAmount;
30                  break;
31              case "salary":
32                  salary -= decreaseAmount;
33                  break;
34              case "sociality":
35                  sociality -= decreaseAmount;
36                  break;
37          }
38
39          if (strength < 0) strength = 0;
40          if (perseverance < 0) perseverance = 0;
41          if (intelligence < 0) intelligence = 0;
42          if (salary < 0) salary = 0;
43          if (sociality < 0) sociality = 0;
```

Рис. 3.12. Реалізація функціоналу перед порівнянням карток

Після налаштування необхідного методу, котрий відповідав за ініціалізацію картки, було прописано метод, у котрому порівнювалися дані карток – CompareParameters (). Даний метод передбачав ініціалізацію картки

гравця та опонента, після чого параметри обох карток порівнювалися між собою. Під час порівняння кожного з параметрів відбувалася анімація спадання остаточних чисел, за допомогою яких гравці можуть визначити переможця.

ВИСНОВОК

Під час виконання кваліфікаційної роботи було розглянуто вид ігор жанру колекційні карткові ігри, а також розробку візуальної частини гри з використанням NFC-карток. Також було проаналізовано ринок ігор та досліджено наявні методи розробки відеоігор.

Основною перевагою даного проекту є використання Unity, оскільки воно забезпечує можливість легкої інтеграції гри до різних платформ, зокрема консолей та мобільних пристроїв. Важливою соціальною цінністю даної гри можна вважати той факт, що гравці можуть колекціонувати картки, використовуючи сучасні технології. Розробка візуальної частини карткової гри з використанням NFC-карток виявилася успішним проектом, котрий об’єднав в себе традиційні настільні ігри та сучасні технології. В рамках даного дослідження було визначено ключові етапи розробки, зокрема дизайн інтерфейсу, створення графічних елементів гри та інтеграцію NFC-технологій.

До списку висновків, зроблених в процесі роботи можна віднести дизайн ігрових карток, інтеграцію NFC, оптимізація, а також користувацький досвід. Створення привабливого та інтуїтивно зрозумілого дизайну є ключовим аспектом для залучення користувачів. Була розроблена серія графічних шаблонів, що підкреслюють тематику гри та забезпечують візуальну ідентичність гри. Використання NFC-карток дало змогу вивести цифрові ігри жанру ККІ на новий рівень взаємодії. Процес зчитування карток додає елемент інтерактивності та вносить помітні корективи до ігрового процесу. Оптимізація графічних елементів та зручності користувацького інтерфейсу забезпечує плавне відображення та захоплюючий ігровий досвід гравцям. Значну увагу було надано зручності. Інтуїтивно зрозумілий дизайн та легко зрозумілі правила гри сприяють позитивному сприйняттю та приємному проведенню часу у грі.

Таким чином, розробка візуальної частини карткової гри з використанням NFC-карток демонструє значний потенціал для створення інноваційного ігрового продукту. Отримані результати та досвід можуть бути використані для майбутніх розробок продуктів у сфері інтерактивних настільних ігор.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DFC Intelligence. A Leading Provider of Industry Analysis and Market Forecasting for the Video Game Sector [Електронний ресурс] (дата звернення: 08.06.2024). URL: https://www.dfcint.com
2. School of Motion. Blender vs Cinema 4D [Електронний ресурс] (дата звернення: 11.06.2024). URL: https://www.schoolofmotion.com/blog/blender-vs-cinema-4d
3. Wikipedia Колекційна карткова гра [Електронний ресурс] (дата звернення: 08.06.2024). URL: https://uk.wikipedia.org/wiki/Колекційна_карткова_гра
4. Офіційна документація Blender 3D TexelDensity [Електронний ресурс] (дата звернення: 10.06.2024). URL: https://blender3d.com.ua/texel-density/
5. 3DAS. Блог про 3Д моделювання. [Електронний ресурс] (дата звернення: 10.06.2024). URL: http://3das.com.ua/shho-take-hdri/
6. Mike Geig Unity Game Development in 24 Hours: Sams Publishing, 2013 p., 39 c.
7. Sue Blackman, Jenny Wang Unity for Absolute Beginners, 2014 p., 132 c.
8. Janine Suvak Learn Unity3D Programming with UnityScript, 2014 p., 56 c.
9. Офіційна документація Unity [Електронний ресурс] (дата звернення: 07.05.2024). URL: https://docs.unity3d.com/Manual/index.html