КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття освітнього рівня бакалавра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

РОЗРОБКА 3D ГРИ З ВИКОРИСТАННЯМ ІГРОВОГО РУШІЯ UNREAL ENGINE

Виконав студент 4-го курсу
Олексій ТКАЧУК _____
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Лариса КАТЕРИНИЧ _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«29» травня 2023р.,
протокол № 11

Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

КИЇВ – 2023

РЕФЕРАТ

Обсяг роботи 41 сторінки, 7 ілюстрацій, 18 джерел посилань.

Ключові слова: ВІДЕОГРА, СТРАТЕГІЯ В РЕАЛЬНОМУ ЧАСІ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, 3D, UNREAL ENGINE, BVH.

Об’єктом роботи є ознайомлення з використанням 3D графіки в прикладному застосуванні та ознайомленні з процесом розробки відеоігор. Предметом роботи є відеогра жанру стратегія в реальному часі.

Метою роботи є аналіз методів загального застосування 3D графіки, розробки 3D відеоігор з використанням рушія Unreal Engine 5 на практичній розробці власної гри, визначення методів застосування інструментарію рушія для створення інтерактивних віртуальних світів і визначити можливу сферу його застосування.

Методи розроблення: рушій Unreal Engine 5.1, інтегроване середовище розробки Visual Studio 2022, мови програмування C++, Blueprint, Python, графічний 3D редактор Blender, бібліотека RLlib.

Результат роботи: реалізовано гру жанру стратегія в реальному часі, проаналізовано переваги рушія Unreal Engine в розробці відеоігор з 3D графікою, продемонстровано приклад оптимізації відеоігор, а саме використання дерева ієрархії обмеженого об’єму для покращення швидкодії гри та також зроблено аналіз у виборі між використанням класичного алгоритмічного підходу і застосуванням навчання з підкріпленням для розробки ботів у відеоіграх.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта розробки. Люди є живими істотами і потребують розваг. З плином розвитку суспільства в життя людей входили нові види розваг, такі як спорт, музика, подорожі, читання, фільми. В другій половині XX століття, з розвитком технологій електронно обчислювальних машин почались створюватись перші відеоігри. Розвиток обчислювальної потужності сучасної техніки не перестає зупинятись, а разом з ним рівень якості комп’ютерної графіки. На сьогодні тяжко уявити наше життя без застосування 3D графіки. Вона використовується не тільки в відеоіграх, а й в кінематографі, конструкторському моделюванні, фізичних симуляціях. Так, наприклад, у 2022 розмір ринку відеоігор зайняв рекордні 196.8$ доларів [1], більша частина якого складають ігри з застосуванням 3D графіки. Тому прикладне застосування 3D графіки є однією з найбільш актуальних сьогодень.

Актуальність роботи та підстави для її виконання. Відеоігровий ринок зростає з кожним роком. Через це постійно зростає попит на розробку нових ігор і кваліфікованих розробників. Звичайно процес розробки ігор залежить не тільки від програмістів, а й від ігрових дизайнерів, сценаристів, 3D дизайнерів, звукових дизайнерів і менеджерів проєкту. Проте на руках програмістів лежить повна відповідальність за технічну частину розробки гри. Використання неоптимальних алгоритмів може призвести до низької швидкодії гри навіть на самих актуальних платформах, що своєю чергою призведе репутаційних втрат компанії розробника і до нижчого рівня продажу. Тому вибір оптимальних технологій у сфері 3D графіки та правильне використання інструментарію для розробки ігор є важливим питанням в сьогодення.

Мета й завдання роботи. Метою роботи є розробка відеогри жанру стратегія в реальному часі з застосуванням 3D графіки на основі рушія Unreal Engine 5. В ході також буде виконано оцінку ефективності рушія для поставленої задачі. Для досягнення мети поставлено наступні завдання:

- Розглянути загальні принципи 3D графіки

- Розібрати базовий набір інструментарію, які надає рушій Unreal Engine 5
- Розробити ігровий дизайн, чіткий процес ігрової логіки, правила і ціль гри.
- Розробити гру та описати процес її створення

Об'єкт, методи й засоби розроблення.

Рушієм для розробки відеогри було вибрано Unreal Engine 5. Причиною його вибору є його висока надійність, широкий спектр інструментарію для створення реалістичного віртуального світу, реалістичної фізики та звуку. Також з основних переваг є його кросплатформеність: Unreal Engine дозволяє розробляти ігри та програми для різних платформ, включаючи ПК, ігрові консолі, мобільні пристрої та віртуальної реальності [2].

Unreal Engine використовує мови програмування C++ або Blueprint. Для роботи з мовою програмування C++ Unreal Engine використовує інтегроване середовище розробки Visual Studio.

Можливі сфери застосування. Реалізована гра жанру стратегія в реальному часі може стати повноцінним комерційним продуктом, за наявності повноцінної команди розробників, шляхом розширення і покращення ігрових механік, створення нових ігрових класів, використанням більш деталізованих і стилізованих 3D моделей, доданням більшої кількості анімацій і введенням звукового дизайну.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

Цифрова дистрибуція – сучасний метод швидкого поширення легального цифрового медіаконтенту, такого як аудіо, фільми, відеоігри та інше, за допомогою Інтернету без використання фізичних носіїв.

Steam – сервіс цифрової дистрибуції компанії Valve

Google Play – платформа цифрової дистрибуції компанії Google.

App Store – платформа цифрової дистрибуції компанії Apple.

GUI – graphical user interface (укр. графічний інтерфейс користувача)

VBH – bounding volume hierarchy (укр. ієрархія обмеженого об’єму)

AABB – axis-aligned bounding box (укр. мінімальна обмежувальна коробка паралельна осям)

API - Application Programming Interface (укр. інтерфейс програмування застосунків)

PPO - Proximal Policy Optimization (укр. оптимізація проксимальної політики)

РОЗДІЛ 1 ІСТОРІЯ І ОСНОВНІ ПРИНЦИПИ КОМП’ЮТЕРНОЇ 3D ГРАФІКИ

1.1 Історія розвитку комп’ютерної графіки

Історія розвитку комп'ютерної графіки починається з 1960-х років, коли перші комп'ютери були розроблені та застосовувалися в наукових дослідженнях та військових цілях. В період того часу було створено перші програми з застосуванням комп’ютерної графіки, яка складалась з простих ліній та точок. «Spacewar!» (рис. 1.1) – одна з перших відомих цифрових комп’ютерних ігор.

Рисунок 1.1 «Космічна війна!» 1962 рік [3]

У 1970-х роках розвиток комп'ютерної графіки був пов'язаний з розвитком програмного забезпечення для створення й обробки векторної графіки, що дозволило спростити процес створення геометричних об'єктів. Також були створені перші програми для створення тривимірних моделей.

У 1990-х роках комп'ютерна графіка стала доступнішою для користувачів персональних комп’ютерів завдяки виходу на ринок графічних прискорювачів з підтримкою 3D графіки. Також з'явилися перші ігрові консолі з підтримкою 3D графіки.

У 2000-х роках комп'ютерна графіка продовжує розвиватися зі збільшенням деталізації графічних об'єктів. Також на масштабний ринок виходять першій шоломи віртуальної реальності, що дозволяють користувачам ще більше поринати

у віртуальний світ, але ціновий клас і сильний вплив на здоров’я не дав змогу стати йому пристроєм масового використання по сьогоднішній день.

Сьогодні комп'ютерна графіка стала невід'ємною частиною нашого життя, її застосовують в іграх, фільмах, архітектурному дизайні, медицині, науці та багатьох інших галузях. Розвиток комп'ютерної графіки не припиняється, з'являються нові технології та алгоритми, які дозволяють створювати ще більш деталізовані та реалістичні об'єкти. Прикладом є тенденція розвитку технологій штучного інтелекту і застосування їх у створені реалістичних віртуальних середовищ і об’єктів. Також прикладом є технологія трасування променів, яка широко набула популярність в останні декілька років. Ця технологія дозволяє створювати реалістичні тіні і відображення об’єктів шляхом швидкої обрахунки променів від світлових об’єктів в реальному часі.

1.2 Моделювання геометрії

Моделювання геометрії є одним з основних етапів створення тривимірних об'єктів в 3D графіці. Цей процес передбачає створення форми об'єкта за допомогою спеціальних інструментів в програмному забезпеченні для 3D моделювання.

Програмне забезпечення для 3D-моделювання – це програмне забезпечення для 3D-комп’ютерної графіки, яке використовується для створення 3D-моделей. 3D-моделювання починається з опису 3 моделей відображення: точок малювання, малювання ліній і малювання трикутників та інших полігональних фрагментів.

3D-моделери дозволяють користувачам створювати та змінювати моделі за допомогою 3D-сітки. Користувачі можуть додавати, віднімати, розтягувати та іншим чином змінювати сітку за своїм бажанням. Моделі можна розглядати з різних кутів. Розробники 3D-моделей можуть експортувати свої моделі у файли, які потім можна імпортувати в інші програми, якщо метадані сумісні.

При моделюванні геометрії важливо враховувати бажану кількість деталей та бажаний зовнішній вигляд об'єкта. Надмірна деталізація може призвести до

зменшення продуктивності програмного забезпечення та збільшення часу рендерингу, тоді як недостатня деталізація може призвести до втрати реалістичності об'єкта.

Крім того, під час моделювання важливо враховувати фізичні аспекти об'єкта, такі як його масштаб, матеріал та освітлення. Це допоможе створити реалістичний вигляд об'єкта в 3D просторі.

Загалом, моделювання геометрії є складним та творчим процесом, який вимагає багато досвіду роботи з програмним забезпеченням для 3D моделювання. Однак, з використанням правильних технік та інструментів, можна створювати вражаючі та реалістичні 3D об'єкти.

1.3 Матеріали та текстури

Матеріали та текстури є важливими компонентами 3D графіки, які допомагають надати реалістичний вигляд для об'єкта. Матеріал описує фізичні властивості поверхні об'єкта, такі як колір, блиск, прозорість та інші характеристики. Текстура використовується для накладання зображення на поверхню об'єкта [4].

Існує безліч матеріалів та текстур, які можуть бути використані в 3D графіці. Одним з найпоширеніших матеріалів є матові матеріали, які мають однорідну поверхню без блиску. Блискучі матеріали мають сильний блиск, що робить їх ідеальним для імітації металів або скла. Прозорі матеріали використовується для імітації прозорих матеріалів. Також, існують й інші типи матеріалів для імітації різних природних властивостей вигляду об’єктів.

Текстури можуть бути використані для створення різних ефектів на поверхні об'єкта, таких як рельєфність, структура, кольорові відтінки та інші характеристики. Текстури можуть бути створені з фотографій або малюнків, або можуть бути створені з нуля за допомогою використання спеціального програмного забезпечення.

Для створення реалістичних матеріалів та текстур, важливо враховувати фізичні властивості поверхні, такі як рефлексія світла та теплопроникність. Також важливо враховувати освітлення в сцені, оскільки воно може впливати на те, як матеріали та текстури виглядають в різних умовах освітлення. Деякі програми для 3D моделювання мають вбудовані бібліотеки матеріалів та текстур, які можуть бути використані для швидкого створення реалістичних об'єктів, але створення власних матеріалів та текстур може дати більш унікальний вигляд сцені.

Одним з важливих аспектів використання матеріалів та текстур є оптимізація їх використання для досягнення оптимальної продуктивності та швидкості рендерингу. Занадто складні матеріали та текстури можуть зменшити швидкість рендерингу та спричинити затримки, тому важливо зберігати баланс між реалістичністю та продуктивністю.

1.4 Освітлення

Освітлення є важливим аспектом 3D графіки, оскільки воно може суттєво впливати на вигляд сцени та об'єктів в ній. В 3D графіці використовуються різні методи освітлення, такі як точкове, неспрямоване та навколишнє освітлення [4].

Основні складові, які впливають на освітлення в 3D графіці, це колір світла, його інтенсивність та напрямок. Колір світла може варіюватися від холодного (білого) до теплого (жовтого), а його інтенсивність визначає, наскільки яскравим буде освітлення. Напрямок світла також дуже важливий, оскільки він визначає напрям тіні та контрастність в сцені.

Щоб створити реалістичне освітлення в 3D графіці, часто використовуються техніки, які моделюють фізичні принципи світла, такі як розсіювання світла, відбиття та переломлення. Ці техніки можуть бути достатньо ресурсозатратними для обробки в реальному часі, тому в при розробці ігор частіше використовують більш спрощені моделі обробки світла на сцені.

1.5 Анімація

Анімація в 3D графіці полягає в створенні рухомих об’єктів на сцені, що включають об'єкти, світло, тіні, текстури та інші ефекти. Основна ідея полягає в тому, що на відміну від статичного зображення, анімація створює враження руху, що дозволяє передавати більше інформації та емоцій.

Створення анімації в 3D графіці може бути досить складним та вимагати багато часу та зусиль. Однак, зараз на ринку існує велика кількість програмного забезпечення, що дозволяє створювати анімацію в 3D графіці, таких як Maya, 3ds Max, Blender та інші.

Основні кроки при створенні анімації в 3D графіці:

1. Створення об'єкта, який буде рухатися в анімації.
2. Налаштування анімаційних ключів для визначення руху та позицій об'єкта на різних етапах анімації.
3. Зміна освітлення та інших ефектів, які допоможуть зробити анімацію більш реалістичною.
4. Рендеринг анімації - це процес створення зображень для кожного кадру анімації.
5. Об'єднання кадрів у відео ряд, що створить враження живого руху.

Основною перевагою 3D анімації є можливість створити реалістичних та деталізованих відеорядів. Вона може бути використана для рекламних роликів, фільмів, відеоігор та інших проєктів. Застосування 3D анімації також дозволяє створювати відеоряди, які були б не по силі для звичайної зйомки. Наприклад, можна створити анімацію, де об'єкти піддаються фізичним взаємодіям, які в реальному житті дуже складно відтворити. Також, 3D анімація надає змогу створити сцени та ефекти, які були б дуже небезпечними для здійснення в реальному житті.

Проте, створення високоякісної 3D анімації може бути дуже складним та вимагати великої кількості часу та ресурсів. Від того, наскільки реалістичним має

бути кадр, залежить кількість ресурсів, які необхідні для створення анімації, а також час, необхідний для рендерингу кожного з кадрів.

1.6 Рендеринг

Рендеринг - це процес створення зображення з 3D середовища, який містити в собі процес обчислення кольорів та освітлення кожного пікселя зображення на основі властивостей матеріалів, текстур, освітлення та позиції й налаштування камери [4].

У 3D графіці існує кілька різних методів рендерингу, включаючи трасування променів, растровий та гібридний рендеринг.

Трасування променів - це метод рендерингу, який прослідковує шлях світла від джерела освітлення до кожного пікселя зображення. Він використовує алгоритми, які симулюють взаємодію світла з матеріалами та поверхнями об'єктів, щоб визначити кольори та тіні на зображенні.

Растровий рендеринг - це метод рендерингу, який використовується для швидкої генерації зображень. Він розбиває зображення на пікселі та обчислює кожен піксель окремо на основі його положення та інших властивостей. Растровий рендеринг зазвичай використовується для рендерингу реального часу в комп'ютерних іграх та інтерактивних додатках.

Гібридний рендеринг - це комбінація трасування променів та растрового рендерингу. Він використовує растровий рендеринг для швидкої генерації зображень та трасування променів для детальної обробки окремих об'єктів на зображенні.

Для рендерингу в 3D графіці використовуються спеціальне програмне забезпечення, таке як Autodesk Maya, Blender, Cinema 4D та інші. Ці програми мають вбудовані алгоритми рендерингу та інструменти для створення реалістичних зображень та анімації. Окрім того, використовуються також спеціалізовані рушії, які можуть бути інтегровані з програмами для 3D моделювання. Ці рушії можуть використовуватися для рендерингу

фотореалістичних зображень та відео, а також для створення спеціальних ефектів та інших візуальних елементів.

1.7 Оптимізація

Оптимізація є важливою складовою в 3D графіці, оскільки шляхом її застосування можна досягнути оптимальної продуктивності та швидкодії в роботі. Однією з основних стратегій оптимізації 3D моделей є зменшення кількості полігонів в ній. Це можна зробити шляхом застосування технік, таких як згладжування, зменшення використання вбудованих засобів автоматичної оптимізації вбудованих в програмне забезпечення для роботи з 3D моделями, заміни складних геометричних форм на простіші тощо. Також важливо враховувати оптимізацію при роботі з текстурами та матеріалами. Наприклад, використання текстур з меншою роздільною здатністю зменшить обсяг використовуваної пам'яті.

При розробці відеоігор варто враховувати значну затрату ресурсів при рендерингу кадрів. При цьому окрім рендерингу комп’ютер має виконувати ігрову логіку і сценарії, виконання яких теж витрачає частину обчислювальної потужності, тому розробникам ігор необхідно використовувати оптимальні алгоритми при їх розробці.

РОЗДІЛ 2 ЕТАПИ РОЗРОБКИ І ВИКОРИСТАННЯ РУШІЯ UNREAL ENGINE 5 В СТВОРЕННІ ВІДЕОІГОР З 3D ГРАФІКОЮ

2.1 Бізнес-аналітика розробки відеоігор

Бізнес-аналітика розробки відеоігор - це процес збору та аналізу даних, що допомагає розробникам та видавцям відеоігор приймати рішення, які стосуються фінансових інвестицій, маркетингових кампаній, стратегій розвитку та інших аспектів галузі.

Один з основних завдань бізнес-аналітика полягає в зборі та аналізі даних про ринок відеоігор, зокрема про продажі, поведінку користувачів, конкуренцію та інші фактори, які впливають на успішність відеоігор. Це дозволяє розробникам та видавцям приймати рішення про те, які ігри потрібно розробляти, як рекламувати та продавати їх, яку цінову політику приймати та інші аспекти.

Окрім того, бізнес-аналітика також займається аналізом внутрішніх процесів розробки відеоігор, зокрема витрат на розробку, час на розробку, кількість розробників та інші фактори, що впливають на фінансові показники та результативність розробки. Це дозволяє розробникам та видавцям приймати рішення про те, як оптимізувати внутрішні процеси та забезпечити більш ефективну розробку відеоігор.

Бізнес-аналітика також займається аналізом поведінки користувачів, які грають у відеоігри. Це включає аналіз даних про те, як користувачі взаємодіють з іграми, які рівні були пройдені, скільки часу користувачі проводять в грі, які функції використовуються найбільше та інші аспекти. Цей аналіз допомагає розробникам та видавцям розуміти, як їхні ігри використовуються та як можна покращити їх користувацький досвід.

Іншим важливим аспектом бізнес-аналітики відеоігор є монетизація. Бізнес-аналітики займаються аналізом різних способів монетизації відеоігор, таких як продажі, мікротранзакції, підписки та реклама. Вони досліджують, які способи монетизації найбільш ефективні для різних типів ігор та для різних ринків.

Крім того, бізнес-аналітика також допомагає визначити цільову аудиторію для відеоігор та встановити стратегії маркетингу, що максимально відповідають інтересам цієї аудиторії. Вони вивчають, які канали маркетингу та реклами є найбільш ефективними, які канали взаємодії з користувачами можна використовувати та які інші методи можна використовувати для підвищення свідомості про відеоігри та привертання більшої уваги до них.

У цілому, бізнес-аналітика розробки відеоігор є невід'ємною складовою успішної розробки та продажу відеоігор. Вона допомагає розробникам та видавцям приймати обґрунтовані рішення та оптимізувати різні аспекти розробки, маркетингу та монетизації відеоігор.

2.2 Стадії розробки відеоігор

Розробка відеоігор зазвичай складається з кількох етапів, кожен з яких вимагає відповідної експертизи та уваги до деталей. Основні стадії розробки відеоігор включають наступні кроки [5]:

1. Концептуальна стадія: ця стадія передбачає створення концепту гри, включаючи її історію, механіку та характеристики персонажів. На цій стадії важливо визначити цільову аудиторію, ринок та збори, які очікуються від гри.
2. Прототипування: на цій стадії розробники створюють прототип гри, щоб перевірити механіку та функціональність гри, а також для оцінки його ігрового процесу. Це дозволяє розробникам виявити та виправити можливі проблеми гри до того, як вона стане готовою до випуску.
3. Продукційна стадія: ця стадія передбачає активну розробку гри, включаючи моделювання персонажів, створення рівнів, анімації та візуальних ефектів, звукового дизайну. На цій стадії важливо забезпечити якість та функціональність гри.
4. Тестування: на цій стадії відбувається тестування гри, щоб виявити можливі проблеми з функціональністю та взаємодією з користувачами. Тестування

проводять як у внутрішній команді розробників, так і на зовнішніх тестових групах.

5. Випуск: після успішного тестування гра готова до випуску на ринку. Це може включати різні шляхи розповсюдження ігор, таких як роздрібний продаж, випуск на консолях або сервісів онлайн дистрибуції відеоігор(Steam, Google Play, App Store та інші).
6. Підтримка та оновлення: після випуску гри розробники зазвичай продовжують працювати над оновленням та підтримкою гри. Оновлення можуть включати виправлення помилок, додавання нових рівнів, персонажів та функцій або навіть додавання нового контенту, який продається за реальні гроші.

Важливо зазначити, що розробка відеоігор є складним процесом, який може займати довгий час та потребувати значних витрат. Від успіху гри залежить відповідне виконання кожної стадії розробки та взаємодія між розробниками та іншими фахівцями, що беруть участь у розробці проєкту.

2.3 Опису функціоналу і можливостей рушія UE5

Unreal Engine — це потужний і дуже універсальний ігровий рушій, розроблений компанією Epic Games. Він надає розробникам повний набір інструментів для створення захоплюючих 3D середовищ. Він широко використовується в ігровій індустрії, але також знаходить застосування у кінематографі, віртуальній реальності й доповненій реальності.

Серед списку видатних ігор реалізованих на рушії Unreal Engine є: Hogwarts Legacy(рис. 2.1) (укр. Спадщина Гоґвортсу), Star Wars Jedi: Fallen Order (укр. Зоряні війни. Джедаї: Полеглий Орден), Sea of Thieves (укр. Море злодіїв) та багато інших ігор [9].

Рисунок 2.1 Спадщина Гоґвортсу (2023 рік, рушій Unreal Engine 4) [6]

Однією з можливостей Unreal Engine є система візуальних сценаріїв Blueprint. Вона надає змогу розробникам створювати ігрові механіки, поведінку «штучного інтелекту», користувацькі інтерфейси та інші компоненти за допомогою візуального користувацького інтерфейсу на основі вузлів і блоків, зменшуючи потребу в традиційних знаннях програмування. Однак Unreal Engine також надає можливість використовувати C++ в якості мови програмування для розробки проєктів. Розробка на C++ надає більше контролю розробникам, дає змогу проводити гнучкішу оптимізацію і загалом має більшу швидкість роботи. Більшість розробників на Unreal Engine використовують комбінацію Blueprint і C++ у своїх проєктах.

Рушій має кросплатформну підтримку, що дозволяє розробникам створювати продукти для різних платформ, включаючи ПК, консолі, мобільні пристрої, VR і AR пристрої. Це дає змогу ширшого охоплення аудиторії та успішного проникнення продукту на масовий ринок.

Unreal Engine містить систему симуляції фізики, яка забезпечує реалістичну взаємодію у віртуальному середовищі. Вона підтримує виявлення зіткнень, динамічність твердих тіла, симуляцію тканини та ефектів руйнування. Фізична

система дозволяє розробникам створювати захопливі та інтерактивні світи з реалістичною поведінкою об’єктів.

Можливості аудіо та звукового дизайну в Unreal Engine забезпечують насичене та захопливе звучання. Він підтримує такі функції, як просторове аудіо, ефекти навколишнього середовища, динамічне мікшування звуку та інтерактивні музичні системи, покращуючи загальну звукову атмосферу в іграх і програмах.

Мережевий функціонал надає розробникам створювати багатокористувацькі ігри. Unreal Engine підтримує архітектуру клієнт-сервер, реплікацію стану гри між мережевими клієнтами та розміщення виділеного сервера. Це дає змогу ефективної розробки багатокористувацьких онлайн-ігор.

Спільнота Unreal Engine є дуже активною. Рушій має якісну документацію, навчальні посібники, має власний форум та безліч онлайн-ресурсів. Крім того, Unreal Marketplace пропонує величезну колекцію наборів матеріалів, плагінів та інструментів, які розробники можуть використовувати для створення своїх проєктів. Розробник Epic Games регулярно оновлює Unreal Engine, надаючи нові функції й роблячи введення покращень та виправлень помилок в наявному функціоналі [8]. Рушій постійно отримує переваги від якісної підтримки, що гарантує розробникам доступ до найновіших інструментів і технологій.

2.4 Переваги та недоліки в використанні C++ і Blueprint в Unreal Engine

Blueprint і C++ — це мови програмування, які використовуються для розробки відеоігор та інших інтерактивних програм у рушії Unreal Engine. Однак вони значну різницю між один одним і пропонують різні рівні гнучкості [7].

Blueprint — це мова візуальних сценаріїв, розроблена для людей які не мають досвіду в написанні коду або тих, хто має обмежений досвід програмування. Мова візуальних сценаріїв дозволяє розробникам створювати логіку гри та поведінку, з’єднуючи вузли в графоподібному інтерфейсі. За допомогою Blueprint можна створювати складні системи, визначати взаємодії між компонентами та впроваджувати ігрові механіки без написання традиційних рядків коду. Він забезпечує більш доступний та інтуїтивно зрозумілий спосіб розробки елементів ігрового процесу.

З іншого боку у розробника є вибір у застосуванні C++ – це мова програмування загального призначення, яка широко використовується в розробці програмного забезпечення, включаючи розробку ігор. C++ надає більше гнучкості та контролю порівняно з Blueprint. Використання C++ дає низькорівневий доступ до системи рушія, що дозволяє розробникам оптимізувати продуктивність, реалізувати складні алгоритми та створювати власні структури даних.

Хоча Blueprint дозволяє швидко створювати прототипи та швидку ітерацію, C++ забезпечує більшу продуктивність і масштабованість. В Unreal Engine Blueprint і C++ можуть використовуватись разом в одному проєкті, причому реалізовані класи та методи мовою програмування C++ можуть бути використані та викликані в скриптах Blueprint. Це надає можливість розробникам поєднувати простоту використання Blueprint із потужністю та гнучкістю C++.

Вибір між Blueprint і C++ залежить від різних факторів, таких як ваш досвід програмування, складність проєкту та вимоги до продуктивності. Початківці без досвіду програмування можуть почати з Blueprint, щоб створити базові механіки ігрового процесу, тоді як досвідчені програмісти можуть обрати C++ для реалізації критично важливих до продуктивності компонентів або спеціального функціонала. У більшості випадків для досягнення балансу між швидкістю розробки та продуктивністю продукту використовується комбінація обох мов.

2.5 Створення віртуального світу в Unreal Engine

Створення віртуального світу в Unreal Engine - це творчий та комплексний процес, що включає розробку ландшафту, налаштування освітлення та розміщення об'єктів на карті.

У створенні віртуального світу важливу роль відіграє ландшафт. За допомогою різноманітних інструментів та редактору рельєфу формується поверхня ландшафту, включаючи гори, долини та водні об'єкти. Щоб зробити ландшафт

реалістичним можна застосовувати текстури та матеріали, що дають різні види ґрунту та рослинності.

Освітлення впливає на атмосферу та настрій віртуального світу. Для роботи з освітленням необхідно розмістити джерела світла, налаштувати їх інтенсивність, колір та напрямок, щоб задати необхідний напрямок тіні. Освітлення може бути природним, наприклад від Сонця або Місяця, або штучним, від ліхтарів, світлофорів та інших джерел світла.

Розміщення об'єктів на рівні додає життєвості віртуальному світу. Використовуючи 3D моделі можна додати будівлі, рослинність, транспортні засоби та інші об'єкти на карту. Це дає можливість створювати міські ландшафти, природні області, підводні світи або будь-яку інші сцени.

Вся ця робота проводиться в Unreal Engine з використанням інструментів, редакторів та налаштувань, що забезпечують зручність розробки та можливості втілення творчої уяви в реальність.

2.6 Розробка графічного користувацького інтерфейсу в Unreal Engine

Щоб створити GUI в Unreal Engine розробникам надається можливість використати вбудовану систему UMG (Unreal Motion Graphics UI Designer). Ця система надає можливість розробляти графічні користувацькі інтерфейси для ігор чи програмного забезпечення за допомогою візуального редактора. Для початку роботи з UMG необхідно створити об’єкт типу Widget Blueprint.

Зайшовши в режим редагування створеного об'єкта відобразиться редактор з полотном, на якому можна створити графічний інтерфейс користувача. Для його створення необхідно перетягувати набір доступних віджетів з палітри на полотно, такі як кнопки, текстові поля, зображення тощо. Розробники мають можливість налаштувати зовнішній вигляд і поведінку своїх віджетів, змінювати кольори, розміри, шрифти та інші візуальні атрибути.

Щоб додати інтерактивності до графічного інтерфейсу користувача, необхідно створити необхідний набір подій для певних дій, таких як натискань кнопок, введення тексту чи інших дій користувача. Для цього треба вибравши віджет і перейшовши до розділу «Події» на панелі конфігурації натиснути кнопку

«+» поруч із подією, яку потрібно обробляти. Зробивши це буде автоматично здійснений перехід у вікно скриптових подій для об'єкта користувацького інтерфейсу і додано відповідний блок з функцією входу обробки цієї події.

Рисунок 2.2 Приклад демонстрації реалізованого графічного користувацького інтерфейсу на Unreal Engine

2.7 Розробка інтерактивного віртуального в UE5 з використанням C++

Ієрархія класів Unreal Engine у мові програмування C++ організована за допомогою системи під назвою Gameplay Framework [11]. Ця система надає набір базових класів та інтерфейсів, які формують основу для створення ігрових об’єктів і систем в Unreal Engine (рис. 2.3).

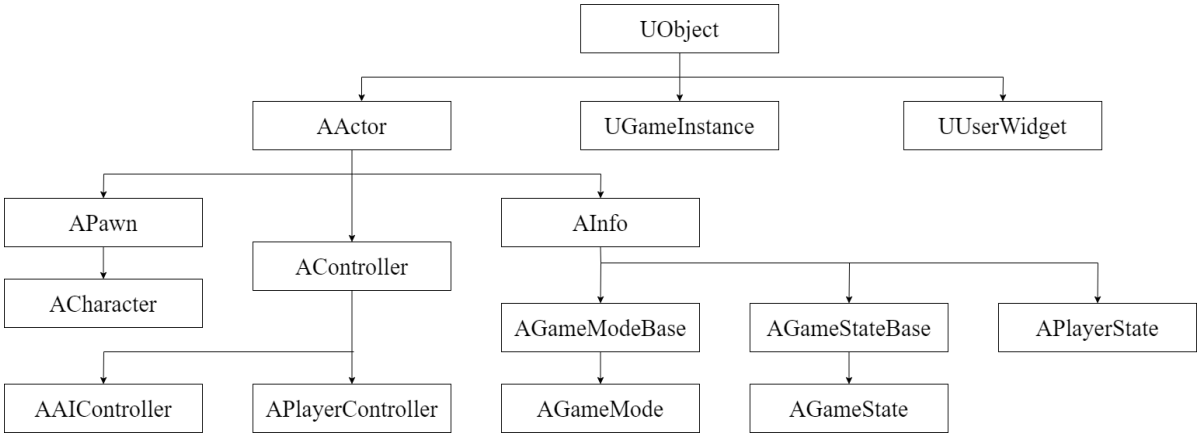

Рисунок 2.3 Схема ієрархії найбільш вживаних класів в Unreal Engine 5

На вершині ієрархії класів знаходиться клас UObject, який є базовим класом для більшості об’єктів Unreal Engine. Клас UObject забезпечує такі фундаментальні функції, як керування пам’яттю, відображення та серіалізація об’єктів.

Безпосередньо під класом UObject ви знайдете клас AActor. Цей клас представляє базу для всіх ігрових об’єктів, які можна розмістити на рівні, таких як персонажі та інтерактивні елементи. AActor надає функціональні можливості для управління розташування, обертання і масштабуванням об’єкта в ігровому світі.

Похідними від AActor [11] є різні спеціалізовані класи, які представляють різні типи акторів. Наприклад:

- APawn: використовується для персонажів або об’єктів, якими можуть керувати гравці або «штучний інтелект» [12].
- ACharacter: базується на APawn і надає додаткові функції, характерні для ігрових персонажів, таких як анімації та рух.
- AController: клас для управління об’єктами з зовнішнього контролю.
- APlayerController: клас відповідає за обробку введених гравцем дій і керування аспектами взаємодії гравця з ігровим світом. У мережевих іграх існує на серверній і на клієнтській частині [13]
- AAIController: суть класу полягає в тому, щоб спостерігати за навколишнім світом, приймати рішення та відповідно виконувати дії без прямого введення з боку людини-гравця.
- AInfo: є базовим класом, який представляє неінтерактивні об’єкти в ігровому світі, які надають інформацію або діють як точки інтересу. Він походить від

класу AActor і служить загальним контейнером для даних і функцій, які не пов’язані з ігровими діями чи взаємодіями. На практиці майже не застосовується.

- AGameModeBase: клас, який використовується для представлення правил та ігрової логіки в рушії. Він діє як центр для керування різними аспектами гри, такими як поява гравців, підрахунок очок, налаштування тощо.
- AGameMode: підклас AGameModeBase, який представляє поведінку багатокористувацької гри. Він має більш розширений функціонал для мережевих ігор основаних на сесії у вигляді матча [14].
- AGameStateBase: клас, який керує глобальним станом гри й створюється класом AGameModeBase. Він існує як на клієнтській, так і на серверній частині й повністю реплікується між ними.
- AGameState: підклас AGameStateBase, використовується для керування станом багатокористувацьких ігор заснована на матчах.
- APlayerState: створюється для кожного гравця на сервері. APlayerState реплікується на всі клієнти та містить інформацію гравця, яка стосується мережевої гри, таких як ім’я гравця, рахунок тощо.

UGameInstance це клас глобального об’єкта гри. Кожна запущена гра має свій екземпляр UGameInstance, який існує протягом всього часу виконання процесу гри. Це надає розробникам можливість зберігати дані при переході між рівнями.

UUserWidget використовується для створення користувацьких інтерфейсів в іграх чи додатках. Він надає засоби для створення і керуванням користувацькими елементами інтерфейсу, такими як кнопки, текстові поля, зображення та інші віджети.

2.8 Аналіз загальних методів розробки ботів і методів їх реалізації в UE5

Під час розробки відеоігор розробникам часто доводиться створювати віртуальних гравців – ботів. Вони використовуються як союзники чи противники людей-гравців, для створення атмосфери виклику або наданні можливості

співпрацювати з напарником в одиночній грі. Боти застосовуються як і в одиночних, так і в багатокористувацьких іграх. В масовій культурі ботів часто називають штучним інтелектом, хоча з технічної точки зору це є не зовсім вірно, оскільки бот не претендує на виконання творчих функцій людини або проходження тесту Тюрінга. Його ціль є вузькоспеціалізована – вибрати дію на основі теперішнього ігрового стану. Проте термін «штучний інтелект» в лапках може використовуватись в цій кваліфікаційній роботі як синонім до слова бот.

Є два підходи розробки ботів: алгоритмічний і оснований на навчанні з підкріпленням. В першому методі програміст має чітко задати правила вибору дії базуючись на поточному стані гри. Наприклад, розробник може задати боту правило, при якому бот виконує певний набір дій якщо виконується умова нестачі певної категорії ресурсів. У випадку застосування навчання з підкріпленням ігрове середовище розглядається, як марковський процес вирішування [15], що є п’ятіркою $(S, A, P_\alpha(s, s'), R_\alpha(s, s'), \gamma)$, де

- S – скінченна множина станів
- A – скінченна множина дій
- $P_\alpha(s, s')$ – ймовірність того, що дія α в стані s призведе до стану s'
- $R_\alpha(s, s')$ – функція нагороди за перехід з стану s до стану s'
- $\gamma \in [0, 1]$ – коефіцієнт знецінювання, який відображає важливість цінності майбутніх нагород відносно поточних

Навчання з підкріпленням є окремим об’єктом для дослідження відносно цієї кваліфікаційної роботи, тому в рамках цієї роботи буде доцільно навести короткий опис принципу його роботи. Навчання з підкріпленням - є методом машинного навчання, в якому агент (бот) взаємодіє з навколишнім середовищем і вчиться приймати послідовні дії для максимізації отриманої нагороди. Агент отримує зворотний зв'язок у вигляді позитивних або негативних нагород за кожну взаємодію з середовищем. В процесі навчання агент вдосконалює свою стратегію, щоб зробити більш оптимальні дії, враховуючи отримані нагороди. Цей цикл

взаємодії й навчання повторюється доти, поки агент не досягне оптимальної стратегії дій у даному середовищі.

Навчання з підкріпленням є проривною технологією, яка широко використовується в області робототехніки, автопілотів машин, сфери реклами та медицини. Проте, вона не набула масштабного застосування у сфері розробки ігор через ряд певних причин. Перша причина, що стоїть на заваді є велика ресурсозатратність. Для реалізації моделі навчання з підкріпленням необхідно витрати більше часу і зусиль для розробників ігор, в порівнянні з реалізацією заданого алгоритму дій. Затратність відображається не тільки в людському ресурсі, а й в ресурсі обчислювальної потужності – успішне навчання може вимагати значних обчислювальних ресурсів, які можуть бути обмеженими під час розробки гри. По-друге, навчання з підкріпленням може надати непередбачувані результати – агент може навчитися не тільки бажаного результату, але вивчити стратегію, яка була не передбачена розробниками. Але цей недолік може бути використаний як перевага, оскільки розробники можуть побачити знайдені недоліки, і вирішити їх для покращення якості гри. Останньою перевагою у виборі алгоритмічного підходу є баланс бота. Навчання з підкріпленням може призвести до створення дуже сильних агентів, що призведе до нерівноцінного балансу у співвідношенні до людини-гравця і стати причиною негативних емоцій у нього.

Для створення боту в Unreal Engine необхідно створити окремий клас контролеру для цього боту. Використовуючи мову програмування C++ або Blueprint можна задати відповідну логіку поведінки бота. Наприклад, можна налаштувати поведінку бота для руху до певної точки на карті, стрільбу у ворога або патрулювання певної території. При розробці боту на Blueprint надається інструмент Behavior Tree (Дерево поведінки), що надає можливість задати ігрову логіку бота у візуальному представленні дерева, що дає змогу іншим розробникам простіше зрозуміти задум логіки дій створеного бота.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ГРИ ЖАНРУ СТРАТЕГІЯ В РЕАЛЬНОМУ ЧАСІ

3.1 Введення в жанри відеоігор

Жанр відеоігри надають змогу класифікувати ігри на основі їх загальних елементів, дозволяючи гравцям визначити тип досвіду, який вони можуть очікувати від гри. Жанри відеоігор можуть охоплювати широкий спектр факторів, зокрема ігрові механіки, тип подачі сюжету та цільову аудиторію. Різні жанри підкреслюють різні аспекти ігрового процесу та пропонують різний ступінь виклику, занурення та інтерактивності.

Деякі поширені жанри відеоігор включають стратегії, шутери, спортивні ігри, симулятори, головоломки, ігри з відкритим світом та багато інших. У кожному жанрі можуть існувати піджанри або гібридні жанри, які поєднують елементи з кількох категорій.

Важливо зазначити, що жанри відеоігор не мають чіткого визначення, і багато ігор можуть включати елементи з кількох жанрів, що ускладнює включення їх до однієї категорії. Крім того, жанри можуть розвиватися та змінюватися з часом, коли в індустрії з’являються нові ігрові механізми та тенденції. Загалом, жанри відеоігор служать корисною основою для розуміння типів ігор, допомагаючи гравцям робити усвідомлення щодо досвіду, який вони можуть отримати у віртуальному світі. Нижче наведений список найбільш популярних жанрів відеоігор:

- Шутер: ігри цього жанру характеризуються на відтворенні бойових дій та механіках стрільби з вогнепальної зброї. Основна мета гравця – знищити ворогів за допомогою набору зброї різного типу.
- Рольові ігри: рольові ігри дають гравцям взяти на себе роль персонажа в віртуальному світі. Гравці розвивають свого персонажа та приймають рішення, що формують його історію. Рольові ігри часто мають насичений сюжет і широкий набір механік налаштування персонажів.

- Стратегія: ігри цього жанру вимагають від гравців стратегічного мислення та прийняття рішень, які впливають на результат. Вони можуть бути покроковими або в режимі реального часу, і зазвичай передбачають управління ресурсами, будівництвом структур на карті для досягнення конкретних цілей.
- Спортивні ігри: імітують реальні види спорту, дозволяючи гравцям брати участь у різноманітних віртуальних спортивних змаганнях, таких як футбол, баскетбол та інші.
- Симулятори: вони спрямовані на відтворення середовища з реального життя. Вони можуть варіюватися від симуляторів польоту до симулятора ферми. Категорія симулятор може відноситись не тільки до ігор, а і до програмного забезпечення для практичної підготовки працівників, шляхом їх тренування в віртуальному середовищі.
- Головоломки: ігри цього жанру дають гравцям можливість розгадувати головоломки чи ребуси.
- Відкритий світ: ігри з відкритим світом надають гравцям величезний віртуальне середовище, яке можна вільно досліджувати. Ігри цього жанру мають нелінійну структуру ігрового сюжету, що дозволяє гравцям вільно брати участь у різних видах діяльності, виконувати місії та взаємодіяти зі світом.

3.2 Жанр стратегія в реальному часу

Жанр гри визначає шлях взаємодії гравця і специфіку оформлення віртуального світу. Так, наприклад, жанр стратегія характеризується тим, що гравець має продумувати вибір свої рішення для досягнення перемоги. В цьому жанрі гравець зазвичай наглядає за навколишнім світом шляхом вільного переміщення камери по карті, для того, щоб отримувати загальну картину поточного стану ігрового процесу і на основі його аналізу приймати певне рішення.

Гравець своїми рішеннями може керувати певними об’єктами на карті, наприклад такими як будівлі, персонажі та інші.

Кожен жанр має свій розподіл на піджанри. Так, найпопулярнішими піджанрами стратегій є стратегії в реальному часі та покрокові стратегії. Різницю між ними можна побачити у їх назвах. У покрокових стратегіях гравець після виконання певної кількості дій передає іншим гравцям чергу і доки черга не повернеться до гравця, то він не може виконувати дії. Одним з найвидатніших покрокових стратегій є серія відеоігор Sid Meier’s Civilization, і серія Heroes of Might and Magic [16]. Стратегії в реальному часі в цьому плані є протилежними до покрокових – гравці можуть вільно робити дії незалежно одне від одного. Такий підхід створює баланс між ігровою динамічністю і залежністю від продуманих кроків, що є привабливішим для гравців, які віддають перевагу більш активному ігровому процесу.

Жанр стратегія в реальному часі був обраний мною для демонстрації розробки гри в рамках кваліфікаційної роботи, оскільки процес її розробки розкриває загальні концепти розробки ігор і демонстрацією можливості використання 3D графіки в практичному застосуванні.

3.3 Опис ігрових правил і необхідні для реалізації компоненти для гри

Ціль гри є перемога над опонентом. На початку гри камера гравця розміщена на його ратуші – основній будівлі, в якій можна переглянути ресурси гравця (деревина, дошки, залізо, каміння, зброя). Умова перемоги – це завоювання ворожої ратуші. Реалізована гра є одиночною. В ролі противника в грі виступає бот. У гравця є можливість будувати будівлі певних категорій, кожна з який має власну функцію:

- Хатина лісоруба – при побудові вирубує дерево на карті з певним інтервалом і збільшує кількість деревини у гравця
- Лісопильня – робить обробку деревини для перетворення її в дошки, які необхідні для побудови нових будівель

- Шахта – при побудові, дає гравцю по одній одиниці каміння і заліза. Каміння необхідне для побудови будівель, а залізо для створення зброї.
- Казарма – розширює кордон при побудові. В ній можна збільшити військову міць за наявності зброї у гравця. Військова міць необхідна для оборони або завоювання ворожої казарми. Завоювання ворожої казарми розширює кордон.
- Кузня – виконує обробку заліза для створення зброї.

Для реалізації гри необхідно створити набір певних компонентів. Перший компонент це карта, на якій гравець і бот зможуть будувати нові будівлі. Другим компонентом є камера з видом зверху. Основним компонентом є система будівель. Необхідно створити класи акторів для кожної з будівлі, які мають наслідувати клас AActor. Актори будівель, які зв’язані з ресурсами, мають виконувати дію з заданим інтервалом, наприклад як добування каміння і заліза в шахті. Також, необхідно налаштувати контролер гравця, щоб при натисненні лівою кнопкою миші на ратушу і казарми відображався відповідний графічний інтерфейс для взаємодії з цими будівлями.

3.4 Створення карти

Для реалізації гри необхідно створити карту. В нашому випадку для робочої версії гри буде достатньо зробити лише одну карту (рис. 3.1), проте в реальних проєктах класичних стратегій в реальному часі необхідно робити декілька карт. Гравцям буде приємніше бачити різне візуальне оформлення і буде приносити більше зацікавленості шляхом застосувань різних стратегій на різних картах. Структура карти є основою для ігрового процесу – відштовхуючись від структури ландшафту і розміщення ресурсів, гравець має вибирати стратегію для перемоги. При побудові карти необхідно враховувати визначені ігрові правила. Так в нашому випадку для побудови нових будівель нам необхідні такі ресурси як дошки та каміння. Вони отримуються шляхом вирубки дерев і добування з шахт. Тому при розробці рівня варто враховувати що біля ратуші гравця має бути достатньо дерев

і мають бути клаптики території на яких доступна побудова шахт, оскільки в іншому випадку гравець застряне на місці – в нього не буде вистачати матеріалів для побудови казарм і без розширення територій він не зможе отримувати доступ до ресурсів.

Для створення ландшафту рушій Unreal Engine є компонент з назвою Landscape. Для роботи з цим компонентом необхідно додати його на карту і перейти в режим Landscape Mode. В цьому режимі є 3 інструменти: Manage, Sculpt і Paint. В першому інструменті Manage відбувається задання параметрів розмірності ландшафту, в ньому можна додавати, видаляти шматочки клаптиків і змінювати їх розмірності. В режимі Sculpt створюється рельєф. При створенні об'єкта Landscape він являє собою велику горизонтальну плоску поверхню. Для надання рельєфності цій поверхні використовується режим Sculpt. Цей режим надає набір інструментів для деформації поверхні, що дає змогу створити нерівності, гори й впадини. Для завершення створення ландшафту необхідно створити багатошаровий матеріал з необхідними текстурами для поверхні. В режимі Paint за допомогою пензлика виділяються області на які треба накладати відповідну текстуру.

Рисунок 3.1 Створена карта з ландшафтом і освітленням

Для створення рослинності на ландшафті рушій Unreal Engine надає режим Foliage. Цей режим надає «малювати» об’єктами на поверхні нашого ландшафту. Задавши набір необхідних 3D моделей дерев і виставивши необхідну густину розташування моделей можемо водити пензликом на карті та швидко додати дерева на ландшафт. Це значно економить час для розробників, оскільки їм не доведеться вручну розставляти кожне дерево на ландшафті. Для моделей дерев в реалізованій грі було використано безплатний набір високодеталізованої рослинності Black Alder, отриманого з вбудованого в рушій магазину матеріалів для розробки Unreal Engine Marketplace.

Також варто врахувати, що створюваний для нашої гри ландшафт є обмеженим клаптиком землі на краях якого видно прірву. Якщо гравець буде бачити цю прірву в безодню, то в нього складеться не найкраще враження про створену гру. Тому варто обмежити пересування камери, щоб її поле зору не виходило за межі карти. Але також можна ввести покращення у вигляді обведення

всіх країв карти водою. Це створить ефект більш реалістичного обмежування карти, що позначиться на більш комфортному ігровому процесі.

3.5 Налаштування камери

Більшість класичних стратегій в реальному часі мають камеру з видом зверху. В створюваній грі це також не буде виключенням. Для створення камери необхідно створити новий клас UCameraPawn, який успадковує клас APawn. Камера контролюється за допомогою курсора миші – при пересуванні курсора до краю екрану камера має рухатись у відповідному напрямку. Це дасть змогу гравцю повністю оглядати карту. Також варто враховувати, що карта є обмеженим шматком території, і тому камера не має виходи за її межі. В основі розробки камери є створення і прив’язка класу UCameraComponent до класу UCameraPawn. Виставивши необхідну висоту, кут нагляду і реалізувавши необхідну логіку ми отримаємо необхідну для грального процесу камеру (рис. 3.2).

Рисунок 3.2 Реалізована камера з видом зверху

3.6 Розробка системи побудови будівель на карті

Для реалізації гри потрібно реалізувати наступні класи будівель: ратуша, шахта, кузня, хатина лісоруба, лісопильня і казарма. Кожна будівля має свій певний робочий функціонал. Для ратуші буде достатньо зробити графічний інтерфейс з переглядом ресурсів гравця для власної ратуші, а для ворожої – інтерфейс з можливістю нападу. Створивши класи акторів для будівель, які відповідають за видобуток, або обробку ресурсів, необхідно вказати відповідні інтервали спрацьовування функціоналу будівлі. Для власних казарм необхідно створити графічний інтерфейс для збільшення кількості військової сили в ній. Для ворожої казарми інтерфейс має мати вибір кількості сил для нападу і кнопку початку атаки.

Графічний інтерфейс гри має мати кнопку відкриття меню зі списком вибору будівлі для побудови (рис. 2.2). Гравець вибирає позицію для будівлі шляхом переміщення курсора на карті. Система побудови будівель має враховувати рельєфність ландшафту, дерева і кордон гравця. Будівля має розташовуватись на рівній поверхні та не перетинати дерева. Також будівля не може буде розміщена поза межами кордонів гравця. При неможливості побудови на певній позиції будівля має підсвічуватись червоним ефектом.

3.7 Розробка боту

Для створення противника для гравця необхідно створити бота. Для цього треба створити окремий контролер для бота. Він має виконувати певну дію за заданим інтервалом. Для вибору дії бот має аналізувати кількість наявних у нього ресурсів. Наприклад, при малій кількості дощок у бота в пріоритеті будівництво хатини лісоруба і лісопильні, оскільки якщо бот витратить всі дошки, то потім він не зможе побудувати будівлі для їх отримання, що призведе до того що бот не зможе побудувати жодної будівлі. Також бот має поповнювати військову силу в казармах і при наявній їй кількості має робити напад на військові будівлі людини-гравця. Розставивши відповідні пріоритети вибору дій для бота, ми отримаємо

базового противника для людини-гравця і створить атмосферу суперництва в грі при відсутності мережевого режиму.

Також для демонстрації буде реалізовано базову версію бота основаного на навчанні з підкріпленням. Як було зазначено в загальному описі процесу розробки ботів для відеоігор, для застосування навчання з підкріпленням, ігрове середовище необхідно розглянути як марковський процес вирішування, першим елементом якого є скінченна множина станів S. В рамках нашої гри множиною станів можна вважати множину можливих варіантів інформації про кількість ресурсів гравців, розташування і клас будівель на карті, кількість військової сили в бараках і ратушах. Необхідність враховувати координати позицій будівель на карті призведе до значної часової витрати під час машинного навчання. Спрощення представлення інформації будівель лише до простого списку без координат їх місцезнаходження на карті не є можливим, оскільки агент має враховувати місцезнаходження власних і ворожих військових на карті будівель для вибору дії.

Наступним елементом марковського процесу є множина дій A. В рамках гри у гравця є набір трьох наступних дій: вибір класу будівлі та локації її розташування, збільшення військової сили у вибраній військовій будівлі, напад на ворожу військову будівлю. Знову виникає проблема аналогічна до скінченної множини станів S – навчання агента має враховувати великий набір множини можливих координат для вибору місця побудови будівлі. Однак для цих двох проблем є рішення, яке значно пришвидшить процес навчання агента. Для цього необхідно зробити відображення координат ігрової карти в набір обмежених прямокутних шматків, які теж мають свої координати розташування, однак вони будуть задані цілими, а не дійсними числами, матимуть менший діапазон варіацій значень, що дасть значний приріст швидкості навчання агента.

Також необхідно підібрати функцію винагороди R, яка має надавати агенту винагороду відповідну до його вибраної дії в певному стані. Складнішою проблемою є необхідність створення мультиагентного середовища, оскільки в реалізованій грі є 2 гравці й роль кожного гравця мають виконувати паралельно два

агенти. Створення даного типу середовища збільшує складність реалізації в порівнянні з простим одно агентним середовищем.

В рушії Unreal Engine немає вбудованого інструментарію для розробки ботів на основі навчання з підкріпленням. Тому доведеться скористатись сторонніми рішеннями. Інструментом для навчання з підкріпленням буде використано бібліотеку RLlib на мові програмування Python. RLlib – це бібліотека з відкритим вихідним кодом для навчання з підкріпленням, яка надає можливості створення продуктів виробничого рівня з навчанням з підкріпленням, при цьому застосовуючи прості та уніфіковані інтерфейси [18]. Оскільки рушій використовує мову програмування C++, то необхідно реалізувати просту між процесну взаємодію для передачі даних про ігровий стан з процесу самої відеогри до процесу з запущеною бібліотекою RLlib.

Для використання бібліотеки RLlib для машинного навчання необхідно створити середовище Gymnasium, яке має містити інформацію про ігровий стан і можливий набір кроків у агенту. Gymnasium – це бібліотека на мові програмування Python, що надає стандартизований інтерфейс для взаємодії з середовищами для навчання з підкріпленням. Бібліотека RLlib надає широкий набір реалізованих алгоритмів для машинного навчання. Алгоритмом для навчання з підкріпленням вибрано алгоритм PPO.

Реалізований агент має ряд певних недоліків які були описані в розділі 2.8, а саме непередбачуваність поведінки. Версія боту на машинному навчанні має недолік у вигляді не самої задовільної поведінки у виборі дій. Для його покращення необхідно робити кращий підбір функції винагороди, налаштування і вибір алгоритму машинного навчання. Проте для демонстраційного прикладу отриманий результат задовольняє поставлені цілі.

3.8 Оптимізація гри за допомогою використання BVH-дерева

При переході в режим вибору місця побудови будівлі відбувається незначна втрата швидкодії гри. Причиною цього є те, що при кожній зміні положення

курсора миші відбувається перевірка на перетин будівлі з деревами, оскільки нові будівлі не мають бути створені на місці де стоїть дерево. Однак на відміну від людини в комп'ютерному представленні візуальне відображення позиції дерева не має жодного значення. При примітивній перевірці на перетин має відбутися повний перебір координат всіх дерев на карті, що має лінійну складність $O(n)$. В нашому випадку це не має дуже великого впливу на швидкодію, проте при розробці більш комплексної версії гри, кожна нова компонента вносить певне навантаження і якщо не придавати значення оптимізації кожній з них, то це може призвести до низької продуктивності гри. Тому при розробці гри варто робити глибокий аналіз ігрових механік і шукати оптимальний алгоритм для їх реалізації.

При пошуку шляху реалізації більш оптимального алгоритму на перевірку перетину об'єкта з іншими об'єктами, було знайдено структуру даних під назвою BVH-дерево (англ. bounding volume hierarchy, укр. ієрархія обмеженого об'єму) – це деревовидна структура даних на множині геометричних об'єктів, листки якого є обмежувальними об'ємами для об'єктів цієї множини [17]. Основна ідея алгоритму цієї структури даних це рекурсивний розбив множини об'єктів на дві підмножини (половини) за координатами об'єктів (рис. 3.3). При перевірці на перетин з об'єктом ми будемо робити рекурсивну перевірку на перетин лише з двома половинами поточної множини, що дає змогу відкидати перевірку половини об'єктів при кожній ітерації, що дає логарифмічну складність $O(log\ (n)\ )$.

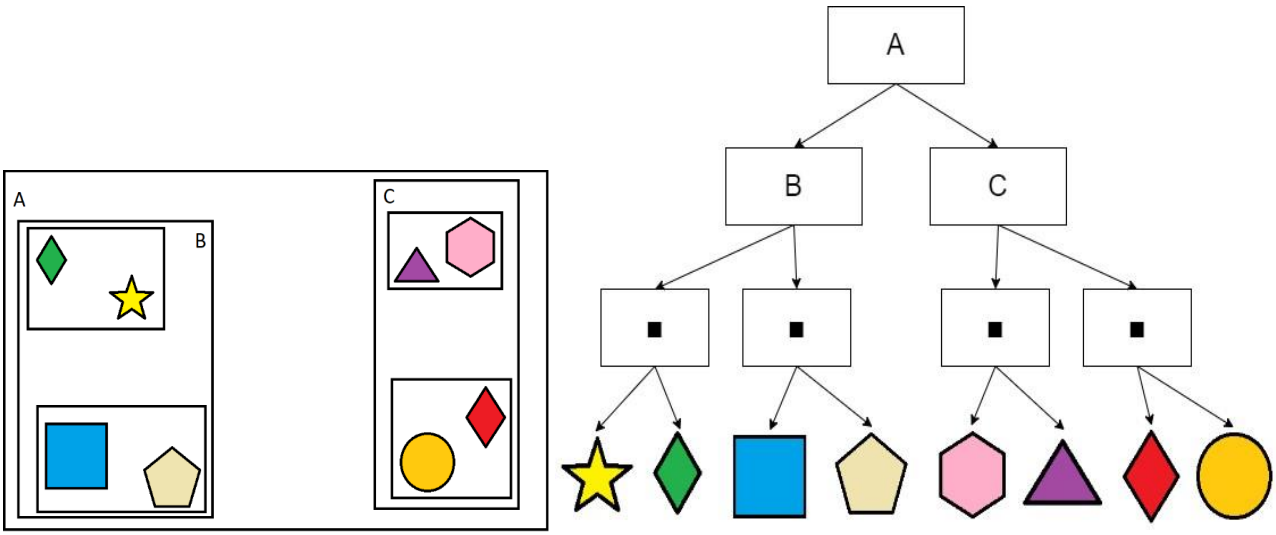

Рисунок 3.3 Візуалізація ієрархії обмежувальних об'ємів з формою AABB

Алгоритм побудови BVH-дерева для об’єктів у двовимірному просторі:

1. Обчислення обмежувального об’єму: формула для обчислення обмежувального об’єму залежить від форми об’єму. Простішою формою для реалізації є AABB (англ. axis-aligned minimum bounding box, укр. мінімальна обмежувальна коробка паралельна осям) – це прямокутна призма, вирівняна за осями координат. Нехай задано набір точок $P = \{(x_1, y_1), (x_2, y_2) ..., (x_n, y_n)\}$, то AABB набору точок P визначається як:

$$\begin{aligned} x_{min} &= min(x_i) \\ x_{max} &= max(x_i) \\ y_{min} &= min(y_i) \\ y_{max} &= max(y_i) \end{aligned} \tag{3.1}$$

2. Обчислення площини розбиття: під час рекурсивного поділу обмежувальних об’ємів площина розбиття використовується для поділу об’єму на два під об'єми. Формула для обчислення площини розбиття залежить від форми обмежувального об’єму та обраної евристики розбиття. Наприклад, у випадку AABB загальною евристичною схемою є розбиття об’єму вздовж його найдовшої осі, тому площина розбиття буде задана як:

$$\begin{aligned} x_s &= \frac{x_{min} + x_{max}}{2} \\ y_s &= \frac{y_{min} + y_{max}}{2} \end{aligned} \tag{3.2}$$

Перевірка на перетин двох обмежувальних об’ємів, залежить від форми об’ємів. Наприклад, у випадку AABB два об’єми перетинаються тоді та тільки тоді, коли їхні проєкції на осі x, у перекриваються: Нехай $AABB^1$ обмежувальний об’єм з координатами $(x^1_{min}, x^1_{max}, y^1_{min}, y^1_{max})$, $AABB^2$ обмежувальний об’єм з координатами $(x^2_{min}, x^2_{max}, y^2_{min}, y^2_{max})$. $AABB^1$ перетинає $AABB^2$, якщо координати двох обмежувальних об’ємів задовольняють системі нерівностей:

$$\begin{cases} x^1_{max} \le x^2_{max} \\ x^1_{max} \ge x^2_{min} \\ y^1_{min} \le y^2_{max} \\ y^1_{max} \ge y^2_{min} \end{cases} \tag{3.3}$$

Якщо розробникам необхідно зробити перевірку точного перетину складних геометричних форм, то вони можуть використати двох етапну методику для її перевірки. В першому кроці можна використати просту примітивну форму, як AABB. У випадку виявлення перетину простих форм можна перейти до наступного кроку, в якому відбувається перевірка на основі реальних об’ємів геометричних об’єктів. Така методика дає значний приріст у швидкодії у порівнянні з простим виявленням перетину лише базуючись на об’ємах моделей. Також варто враховувати, що BVH-дерево знаходить застосування не тільки для оптимізації визначення перетину геометричних об’єктів, а й може використовуватись для трасування променів.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було реалізовано гру жанру стратегія в реальному часі. В процесі її розробки було розглянуто і проаналізовано загальні концепції застосування 3D графіки і розробки відеоігор. Шляхом розробки було продемонстровано процес створення власного рівня (карти), створення інтерактивних ігрових об’єктів, реалізацію оброки вводу гравця-людини, створення ігрового бота і було наведено практичний приклад оптимізації відеогри шляхом застосування швидкодійної структури даних для геометричних об’єктів. Додатково було проведено аналіз переваг і недоліків між використання алгоритмічного підходу і застосування машинного навчання під час розробки бота для відеоігор.

Також в даній роботі було розглянуто застосування рушія Unreal Engine для розробки відеоігор. Базуючись на досвіді отриманому в процесі розробки, можна підсумувати, що рушій Unreal Engine є потужним інструментом для створення ігор і додатків з застосуванням 3D графіки. Рушій надає можливість розробникам концентруватись безпосередньо на самій розробці ігрової логіки, а не створювати базові речі з нуля, тому Unreal Engine ідеально підходить для невеликих і середніх ігрових студій, в яких обмежені ресурси для розробки. Проте, великі студії також можуть їм користуватись у випадку бажання зменшити матеріальні й часові витрати на процес розробки. За рахунок постійних оновлень рушій надає набір найактуальніших графічних технологій, що дозволяють розробникам з легкістю створювати віртуальні світи з реалістичною графікою. За рахунок наявності на борту рушія Unreal Engine двох мов програмування Blueprint і C++, що надає йому можливість для використання людям без досвіду в написанні програмного коду, так і для досвідчених програмістів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The Games Market in 2022: The Year in Numbers [Електронний ресурс] – Режим доступу до ресурсу: https://newzoo.com/resources/blog/the-games-market-in-2022-the-year-in-numbers
2. Unreal Engine [Електронний ресурс] – Режим доступу до ресурсу: https://www.unrealengine.com/en-US
3. The History of Spacewar: The First Computer Game – Режим доступу до ресурсу: https://www.thoughtco.com/history-of-spacewar-1992412
4. John H. Computer Graphics: Principles and Practice / H. John, D. Andries, S. David – Addison-Wesley Professional., 2013. – 1264 с.
5. Mark J. Encyclopedia of Video Games / J. Mark, P. Wolf – Greenwood
6. Hogwarts Legacy у Steam [Електронний ресурс] – Режим доступу до ресурсу: https://store.steampowered.com/app/990080/Hogwarts_Legacy
7. Blueprints vs. C++ [Електронний ресурс] – Режим доступу до ресурсу: https://awforsythe.com/unreal/blueprints_vs_cpp
8. Epic Games [Електронний ресурс] – Режим доступу до ресурсу: https://www.epicgames.com/site/en-US/home
9. List of Unreal Engine games [Електронний ресурс] – Режим доступу до ресурсу: https://thecoolestvideogames.fandom.com/wiki/List_of_Unreal_Engine_games
10. Gameplay Framework in Unreal Engine – Режим доступу до ресурсу: https://docs.unrealengine.com/5.1/en-US/gameplay-framework-in-unreal-engine
11. Actors in Unreal Engine [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unrealengine.com/5.1/en-US/actors-in-unreal-engine
12. Pawn in Unreal Engine [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unrealengine.com/5.1/en-US/pawn-in-unreal-engine
13. Controllers in Unreal Engine [Електронний ресурс] – Режим доступу до ресурсу:

https://docs.unrealengine.com/5.1/en-US/controllers-in-unreal-engine

14. Game Mode and Game State in Unreal Engine [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unrealengine.com/5.1/en-US/game-mode-and-game-state-in-unreal-engine
15. Laura G. Foundation of deep reinforcement learning / G. Laura, L. K, Wah – Addison Wesley Data & Analytics Series., 2020. – 416 с.
16. The best strategy games in 2023 [Електронний ресурс] – Режим доступу до ресурсу: https://www.pcgamer.com/the-best-strategy-games
17. Christer E. Real-Time Collision Detection / E. Christer – Morgan Kaufmann Publishers., 2004. – 632 с.
18. RLlib: Industry-Grade Reinforcement Learning [Електронний ресурс] – Режим доступу до ресурсу: https://docs.ray.io/en/latest/rllib/index.html