МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

“До захисту допущена”

Зав. Кафедри комп'ютерних наук та прикладної математики

д.т.н., професор Турбал Ю. В.

«___» _____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

ТЕМА:

«РОЗРОБКА БАЗОВОЇ ГРИ НА РУШІЇ UNREAL ENGINE 5»

Виконав: Павлюк Станіслав Сергійович

студент навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки

групи ІПЗ-33інт.

_____

Підпис

Керівник: Герус Володимир Андрійович

_____

Підпис

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 57 с., 22 малюнків, 12 джерел.

Актуальність теми: розробка базового функціоналу шутера симулятора є актуальною, оскільки вона відповідає потребам зростаючого інтересу до жанру шутерів, а саме виживання, поводження себе в тій чи іншій ситуації. Такий симулятор надає можливість експериментувати з концепціями карт, досліджувати вплив різних факторів та покращувати процес з точки зору функціональності та візуальності. Це може бути корисним для студентів, дослідників, архітекторів та місцевих урядів, що прагнуть що прагнуть провести чудово час.

Мета роботи: розробка базового функціоналу для гри шутера, який стане фундаментом або основою для подальшої розробки. Головними цілями роботи є:

1. Розробити базовий функціонал
2. Створити робочу гру
3. Прописати фізику
4. Та налагодити функціювання гри

Ключові слова: Unreal Engine 5, БАЗОВИЙ ФУНКЦІОНАЛ, СИСТЕМА BLUEPRINTS, UNITY 3D.

ВСТУП

Через десятки років після своєї появи індустрія відеоігор зайняла власну нішу на ринку, поряд з іншими видами розваг сфери мультимедіа, такими як кіно, мультиплікація, музика. За останнє десятиліття галузь розробки віртуальних розваг збільшилася в рази порівняно з минулим десятиліттям, через низького порогу входу для нових користувачів. За статистикою SteamSpy на платформі Steam у 2017 році вийшло 7 672 гри, порівняно з 2016 роком цей показник вищий на 2 427 ігор. Також портал опублікував статистику ігор, що вийшли з 2004 по 2017 рік, і в ній йдеться про те, що 40% ігор вийшло якраз у 2017 році.

Зараз ігри охоплюють велику аудиторію, з'являючись на різноманітних ігрових пристроях. Крім того, що ігри є сферою розваги, так вони ще й приносять користь. На думку Філа Спенсера - "Ігри дають змогу людям стати рівними. Усі ми граємо в них - незалежно від віку, раси, політичних переконань, релігії, національності та здібностей. Ігри при цьому зводять нас разом – вони об'єднують гравців за допомогою універсальної мови - задоволення.

Люди грають в ігри не тільки щоб розслабитись, скільки для гарного проведення часу, гра яка дає: захоплюючий прилив адреналіну, незабутні пригоди, виклик гравцю. Люди грають в ігри, щоб отримати незабутні емоції та пройти емоційні переживання, чи зможуть вони подолати ігрові виклики, шукаючи забуття від щоденної рутини.

Інформаційні технології, з одного боку, ускладнили життя зарашніх маркетологів, вводячи нові терміни, поняття та визначення, нові методи і сутність роботи, а з іншого – надали новий вектор можливостей, розширивши межі впливу компаній.

Актуальність даної дипломної роботи полягає в тому, що з часом інформаційні технології необхідно оновлювать в нові програмні продукти, що будуть давати змогу розвивати ігрову Індустрію та ринок розваг в цілому. На сьогоднішній день є велика кількість технічних можливостей для побудови

багатоплатформених ігор у різних ігрових девайсах, що дозволяє поринути у програмний продукт у сегменті ринку, що росте і стрімко розвивається у світі.

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

Предметом досліду є комп’ютерна гра.

Комп’ютерна гра – це тип відеоігор, в які грають на персональному комп’ютері (ПК), а не на спеціальний для цього консолі чи автоматі. Це зумовлено рядом причин, що будуть досліджуватись у дипломній роботі. Гра, як і будь-який програмний продукт проходить певні етапи від задумки до готового продукту та має свої якості щодо створення та реалізації.

Мета дипломної роботи: проектування ігрового процесу і виконання етапів розробки комп’ютерної гри.

Для досягнення мети дипломної роботи поставлено наступні завдання :

1. Дослідити ринок ігрової індустрії;
2. Дати основну характеристику програмним приладам для створення ігор;
3. Проаналізувати основні етапи створення комп’ютерної гри;
4. На основі даних аналізу, розробити комп’ютерну гру.

РОЗДІЛ 1. Дослідження предметної області

1.1. Аналіз стану ігрової індустрії на сьогоднішній день

Ігрова індустрія є однією з найбільших індустрій розваг у світі, нарахоуючи близько 3 мільярдів гравців у всьому світі. Розраховується, що в 2023 році ігрова індустрія отримає прибуток у розмірі близько 390 млрд доларів США.

Загальний обсяг світового ринку відеоігор оцінювався вп 213,58 мільярдів доларів США в 2019 році і очікується, що буде тільки рости в середньому темп зростання (CAGR) в 15,9% з 2017 по 2027 рік. Очікується, що технологічне розповсюдження та інновації що в апаратному, що в програмному забезпеченні будуть відігравати головні фактори росту. Зростаюче розповсюдження інтернет-послуг в поєднанні з легким доступом до ігор що знаходяться в Інтернеті по всьому світу також дає оптимістичні перспективи зростання ігрового ринку в майбутні роки. Розробники ігор завжди бімпровізують і перевершують технологічні можливості щодо рендерингу графіки в зарашньому часі в індустрії відеоігор, що, в майбутньому, буде сприяти її активному зростанню.

Рис. 1.1. Динаміка доходу на ринку відеоігор.

Зростаюча тенденція зміни від фізичних ігор до онлайн-ігор змусила шанувальників галузі зосередитися на апаратних можливостях та ефективності. Free2Play, Massively Multiplayer Online (MMO) і розраховані на велику аудиторію користувачів ігри поступово стають популярними, і очікується, що ця тенденція тільки збільшиться протягом наступних кількох років. Зростаючий рівень активного доходу веде до збільшення загальних витрат на ігрові проекти. Крім того, зміни уподобань гравців призвели до масового впровадження більш досконалих ігрових консолей, оснащених такими новими функціями, як запис і загальне використання, а також багатоплатформені ігрових процесів.

Тенденція ігор в загальних мережах матиме великий вплив на зростання ринку. Наприклад, великий відсоток світового населення використовує для ігор такі сайти соціальних мереж, як Facebook і Reddit. Очікується, що доступність різних жанрів ігор, таких як бойовик, рольові ігри, симулятори та стратегії, дасть більше клієнтів. Збільшена популярність кіберспортивних змагань і збільшення числа професійних гравців дасть поштовх до збільшення продажів відеоігор та аксесуарів, а також ігрового обладнання та програмного забезпечення.

Протягом останніх кількох років кіберспорт стає активно розвивається як з боку виграшу призів, так і з боку зору визнання у всьому світі. Хоча близько пару років тому це було нішею для маленької спільноти комп’ютерних фанатів, тепер все більше підлітків з нетерпінням чекають на можливість побудувати кар’єру в кіберспорті.

Наразі кіберспорт охоплює великий спектр дисциплін, починаючи від шутерів та стратегій до MMO, симуляторів та гоночних ігор. Варіантів для взяття участі багато, але весь час з’являються нові дисципліни. Наразі були великі зміни по жанрам Battle Royale яке набуде такої популярності, що на турнірах Fortnite, Apex Legends та PUBG будуть представлені призові фонди як грошові так і призові.

Кіберспорт в цілому – це вид спорту, який включає велику кількість дисциплін та з великими призовими фондами що дає змогу йому стати найбільш популярним джерелом розваг. Звичайних гравців усі ці фактори захоплюють, оскільки захоплення відеоіграми завжди була притаманна всім групам вікової категорії.

Рис. 1.2. Масштаби кіберспортивних змагань.

Кіберспорт вже називають майбутнім всіх видів спорту, особливо після спалаху COVID-19. Тепер цілком зрозуміло, що ця ніша – гарний вибір для всіх людей, захоплених іграми та популярними жанрами кіберспорту.

Ринок відеоігор має високий попит у різних сферах, таких як навчальні заклади та корпоративні підприємства. Використання ігор як навчального інструменту надає можливість для більш глибокого та пізнавального навчання. Поняття "ігри для навчання" існує досить давно. Однак реальний потенціал можливостей гейміфікації в області академічного середовища використовується лише нещодавно.

Гейміфікація в процесі навчання дає змогу використовувати наступних ігрових елементів, як підрахунок балів, конкуренція між гравцями, робота в команді, таблиці оцінок, щоб дати змогу залучати, допомагати учням засвоїти нову інформацію та перевірити свої знання. Вона дає змогу поширюватися на шкільні предмети, але також вона широко поширюється в додатках та курсах самоосвіти, доводячи, що переваги гейміфікації не зникають, навіть коли ми дорослі.

Технології оточують велику частину нашого звичайного життя , замінивши те, як люди живуть, робить покупки, працювать, грать, харчуваться, знайомляться та спілкуються з людьми. Політики починають розглядати потенційні переваги використання технологій для оптимізації робочого навантаження працівників.

Однак поширення фальсифікованої продукції через їх низькі ціни, особливо в таких країнах, як Китай та Індонезія, дещо стримуватиме зростання ринку. Очікується, що питання авторського права та піратства буде негативно впливати на досвід користувачів. Побоювання користувачів, пов'язані з шахрайством під час транзакцій з іграми, також стримуватимуть зростання ринку. Різке зростання проблем зі здоров'ям та проблем, пов'язаних з відеоіграми, – ще один фактор, який, як очікується, стримуватиме зростання індустрії [2].

Пандемія COVID-19 продовжує впливати на світову економіку, однак очікується, що індустрія відеоігор продемонструє значне зростання протягом наступних кількох місяців. Оскільки уряди у всьому світі обмежили перебування людей за межами дому для запобігання поширенню COVID-19, компанії стають свідками зростання кількості користувачів та зростання кількості годин, які користувачі витрачають на онлайн-ігри. Крім того, деякі компанії-розробники приймають рішення запускати свої онлайн-ігри для безкоштовного завантаження. Наприклад, у березні 2020 року компанія Activision Blizzard, Inc. випустила гру “Call of Duty: Warzone” для безкоштовного завантаження та отримала близько 6 мільйонів завантажень за один день.

Через пандемію різні компанії страждають від затримок та перерв у випуску своїх продуктів. Крім того, різні прес-конференції, на яких планувалося оголошення останніх анонсів ігор та трейлерів, також відкладаються. Наприклад, конференція розробників ігор, яка спочатку була запланована на червень 2020 року, була офіційно перенесена через зростання і поширення пандемії COVID-19. Пандемія також вплинула на виробництво обладнання та логістику поставок.

Багато з цих затримок пов'язані з закриттям різних виробничих потужностей у Китаї, де виробляється багато таких ігрових продуктів. Наприклад, у лютому 2020

року Nintendo Co., Ltd. оголосила про затримку випуску Nintendo Switch, який спочатку планувався 6 березня 2020 року, оскільки закрив свої виробничі потужності через COVID-19.

1.2. Особливості які вплинуть на розробку ігор в майбутньому

Розробка ігор у майбутньому буде суттєво впливати на різні фактори. Ось кілька ключових факторів, які можуть вплинути на розробку ігор в майбутньому:

1. Технологічний прогрес: Постійне вдосконалення технологій, таких як штучний інтелект (AI), віртуальна реальність (VR), розширена реальність (AR) та облачне обчислення, впливає на можливості ігрової індустрії. Розробники будуть мати доступ до потужних інструментів і технологій, що дозволить створити більш реалістичні та інноваційні ігрові досвіди.
2. Штучний інтелект (AI): Розвиток штучного інтелекту може призвести до створення більш розумних та реалістичних ворогів, союзників та НПС (некерованих персонажів). Штучний інтелект також може використовуватися для автоматичного створення контенту, такого як генерація рівнів або завдань, що дозволяє розробникам ефективно створювати великі ігрові світи.
3. Соціальна взаємодія та співробітництво: Зростаюча популярність мультиплеєрних та онлайн-ігор означає, що розробники будуть зосереджені на розширені можливості соціальної взаємодії та співробітництва гравців. Це можна включати нові способи спілкування, інтеграцію з соціальними медіа та обмін обміну контентом.
4. Гранична реалістичність: Розробники будуть продовжувати покращувати реалістичність графіки, фізики та звуку в іграх. Застосування нових технологій, таких як трасування променів (ray tracing), дозволяє створити деталізовані та іммерсивніші світи.
5. Переносність і кросплатформенність: Гравці все більше очікують, щоб їх ігровий

досвід був доступний на різних пристроях і платформах. Розробники будуть зосереджені на іграх, які можна грати на комп’ютерах, консолях, мобільних пристроях та інших пристроях зі збереженням створеного прогресу та кросплатформеною грою.

6. Екологічна стійкість: В останні роки зростає усвідомлення екологічних проблем. Розробники можуть звернути увагу на створення екологічно стійких ігрових продуктів, зменшення споживання енергії, використання відновлених матеріалів та інше для зменшення негативного впливу на навколишнє середовище.

Ці фактори кілька тільки прикладів того, які фактори можуть вплинути на розробку ігор в майбутньому. З інноваціями в технологіях та змінами в суспільстві можуть з'явитися нові можливості та виклики для галузі.

1.2.1. Віртуальна та доповнена реальність

Віртуальна реальність (VR) і розширена реальність (AR) є двома суміжними технологіями, які перетинають межі між реальним світом та віртуальними елементами, пропонуючи унікальний досвід користувачам. Ось що варто знати про VR і AR:

Віртуальна реальність (VR): VR – це технологія, яка створює іммерсивне враження від присутності у віртуальному середовищі. Користувач носить спеціальну гарнітуру VR, яка затамовує зовнішні подразники та замінює їх цифровими видимими та звуковими елементами. Гарнітур може мати вбудовані датчики, що відстежують рухи голови користувача, що забезпечує інтерактивність з віртуальним середовищем.

Основні пристрої VR включають Oculus Rift, HTC Vive, PlayStation VR та інші. VR використовується в багатьох галузях, включаючи геймінг, навчання, медицину, архітектуру та симуляцію.

Розширена реальність (AR): AR - це технологія, яка додає віртуальні об'єкти інтерактивно до реального світу. За допомогою AR-пристроїв, таких як смартфони, планшети або спеціальні AR-гарнітури, користувач може спостерігати реальне середовище, в яке додаються віртуальні об’єкти, які взаємодіють з ним.

Приклади AR-технологій включають популярні додатки, такі як Pokémon Go та Snapchat, а також промислові застосування в архітектурі, маркетингу та медицині. Деякі AR-пристрої, як відомо, включають Microsoft HoloLens і Google Glass.

Віртуальна та розширена реальність надає нові можливості для ігрової індустрії. Гравці можуть занурюватися у повністю віртуальні світи VR-ігор або взаємодіяти з віртуальними об’єктами та персонажами в реальному світі за допомогою AR. Ці технології забезпечують нові рівні іммерсії та взаємодії, що перетворюють спосіб, яким гравці взаємодіють з ігровими середовищами.

Рис. 1.3. Приклад використання технології додаткової реальності.

Вважається, що вихід даного ігрового двигуна Unreal Engine 5 (UE5) дасть змогу стати силою для всього технологічного процесу утворення ігор. AR / VR більше не буде рідкісною функцією ігор, а стане розповсюдженим форматом.

1.2.2. Хмарні технології

Хмарні технології в ігровій індустрії залишаються все більш популярними і впливовими. Ось декілька ключових аспектів хмарних технологій у контексті ігор:

1. Ігри на вимогу (Gaming-as-a-Service, GaaS): Хмарні технології дозволяють грати в ігри на вимогу, без необхідності завантажувати або встановлювати їх на власний

пристрій. Гравці можуть потоково отримувати графічно вимогливі ігри через Інтернет, запускаючи їх на серверах хмарних платформ. Це дозволяє грати в складні ігри навіть на пристроях з обмеженими технічними характеристиками.

2. Зберігання та синхронізація: завдяки хмарним технологіям, гравці можуть зберегти свій ігровий прогрес, досягнення та налаштування в хмарних сховищах. Це означає, що гравець може переходити від одного пристрою до іншого, а його ігровий прогрес буде автоматично синхронізований, дозволяючи продовжувати гру без перерви.
3. Мультиплеєр та спільна гра: Хмарні технології сприяють розширенню можливостей мультиплеєра та спільної гри. Гравці можуть з легкістю підключатися до онлайн-ігрових серверів через хмарні платформи, спілкуватися з іншими гравцями та взаємодіяти з ними у віртуальних світах.
4. Масштабованість та оновлення: Хмарні технології дозволяють розробникам ігор масштабувати серверні ресурси для вирішення збільшених потреб під час великих мультиплеєрних подій або релізів нових ігор. Вони також спрощують процес оновлення та патчінгу ігор, останні ці зміни можуть бути виконані на рівнях серверів хмарних платформ, які потребують внутрішнього завантаження оновлень гравцями.
5. Спільнота та стрімінг: Хмарні технології також дозволяють гравцям спілкуватися, спостерігати та спільно виконувати ігри через стрімінгові платформи. Це дає можливість поділитися своєю ігровою досвідом з глядачами та взаємодіяти з ними в реальному часі.

Загалом, хмарні технології змінюють підхід до розробки, розповсюдження та гри в ігри, надаючи більшу доступність, гнучкість та інноваційні можливості для гравців та розробників.

1.2.3. Новітні технології розробки ігор

Unreal Engine – ігровий двигун, що змінить ігрову індустрію. Unreal Engine 5 дебютував у 2021 році, і невдовзі це дасть йому змогу стати одним з основних інструментів, які дадуть змогу всій ігровій індустрії зробити крок уперед у якості графіки.

Як багатьом геймерам вже відомо, відеоігри складаються з тисяч полігонів, які відеокарта читає та інтерпретує. Unreal Engine 5 змінює спосіб взаємодії розробників з цими фігурами, запроваджуючи нову систему віртуалізованих полігонів.

Розробка гри є складним і творчим процесом, і новітні технології грають важливу роль у покращенні якості та ефективності цього процесу. Наукове обґрунтування використання новітніх технологій у розробці ігор можна побудувати навколо кількох ключових аргументів:

1. Покращена візуалізація: завдяки новим технологіям розробок, таким як рушії Unreal Engine 5 та Unity, розробники можуть створити більш реалістичні та вражаючі видимі ефекти. Технології, які базуються на фізичному моделюванні світу, деталях та глобальному освіті можуть досягти високої якості графіки та підвищити реалістичність оточуючого світу в іграх.
2. Поліпшені інтерактивність та геймплей: Новітні технології дозволяють розробникам створювати більш інтуїтивні та захоплюючі ігрові досвіди. Наприклад, використання віртуальної реальності (VR) та доповненої реальності (AR) дозволяє гравцям взаємодіяти з грою в більш імерсивний спосіб. Також розробка ігор з використанням штучного інтелекту та машинного навчання може покращити інтелектуальні можливості ворогів, персонажів та ігрових механізмів, забезпечуючи більш складні та цікаві виклики для гравців.
3. Підвищена продуктивність та ефективність: Використання новітніх технологій розробки дозволяє збільшити продуктивність розробничих команд і зменшити час, необхідний для розробки гри. Наприклад, використання інструментів розробки, таких як системи управління версіями, автоматизовані тести та інтегроване

середовище розробки (IDE), допоможе розробникам прискорити ітераційний процес та забезпечити високу якість гри.

4. Розширені можливості мультиплатформенності: завдяки новим технологіям розробки, розробники можуть створювати ігри, які працюють на різних платформах, таких як консолі, ПК, мобільні пристрої тощо. Це дає змогу досягти ширшої аудиторії та забезпечити кросплатформену взаємодію між гравцями.

Використання нових технологій у розробці ігор дозволяє розробникам створити більш вражаючий та захоплюючий ігровий досвід, підвищити продуктивність та забезпечити мультиплатформенність. Це сприяє розвитку ігрової індустрії та задоволенню потреб сучасних гравців.

1.3. Сегментація аудиторії при розробці гри

Сегментація аудиторії є етапом у проці розробці гри, поза програмою цільової групи гравців і забезпечує їм дізнатися про потреби цільової аудиторії та отримати належний досвід гри.[3]

Розглянемо кілька наступних факторів, які можна отримати при сегментації аудиторії:

1. Ігровий досвід та інтереси: Розробники також можуть отримати рівень ігрового досвіду та інтересів аудиторії. Це може включати схожі жанри ігор, типи геймплею, подібні платформи тощо. Наприклад, любителі стрілянина в першій особи можуть бути цікаві в іграх з багатокористувацьким режимом або екшен-іграх з великим арсеналом зброї.
2. Інтереси та хобі: Вивчення інтересів і хобі наявних гравців може допомогти в розробці гри, яка відповідає їхнім інтересам. Наприклад, гра, пов’язана з науково-фантастичними темами, може привернути гравців, які цікавляться наукою та фантастикою.
3. Психографічні характеристики: Психографічна сегментація зосереджується на

психологічних аспектах аудиторії, таких як особистість, цінність, інтереси та спосіб життя. Це може допомогти допомогти, які емоційні стимули та мотивації можуть впливати на гравців, а які елементи гри можуть бути привабливими для певних сегментів аудиторії.

4. Демографічні характеристики: Один із способів сегментації аудиторії - це за допомогою демографічних характеристик, таких як вік, стать, місце проживання, освіта та дохід. Ці фактори можуть впливати на ігрові вподобання, стиль гри та можливість придбання додаткових елементів або внутрішньоігрових предметів.
5. Географічні фактори: Культурні відмінності та географічні особливості можуть впливати на вибір тематики, стилістики та контенту гри. Наприклад, гра з орієнтованим на північноамериканський ринок змістом може вимагати інший підхід, ніж гра для азіатського ринку.

Кожен сегмент цільової аудиторії можна розписати за допомогою різних показників. Такий як розмір аудиторії; її частка конверсії; наскільки легко гравці можуть здолати перешкоди, щоб потрапити до ігрового проекту; ступінь лояльності (retention); можливість зберігати аудиторію (retention і sticky factor) — чисельних параметрів, які поділяються у кожного сегмента, досить багато. Ключовими характеристиками кожного сегмента є її ROI (return of investment), LTV (lifetime value) і розмір цього сегменту. Частково вони містять такі важливі критерії, як основний дохід з кожного гравця (ARPPU — average revenue per paying user), частка аудиторії, що платить у своєму сегменті (PU) і retention (відданість аудиторії).

1.4. Аналіз найпопулярніших ігрових пристроїв

На даний момент, 2023 році геймери мають широкий спектр вибору ігрових платформ для їхніх улюблених ігор. А саме мобільні,ПК,консолі.

На цих платформах геймери мають до величезних збірок бібліотек ігор. Від найпопулярніших відеоігор до менш популярних стилів гри, таких як, наприклад, мовних ігор та навіть програм для азартних ігор.

Кількість гравців у відеоігри зросла на 90% з 2019 по 2023 рік, частково через COVID-19, оскільки все більше людей залишаються вдома без роботи та проводять свій вільний час у відеоіграх.

Рис. 1.4. Статистика популярності ігрових пристроїв

1.4.1. Мобільні ігри

Кількість користувачів смартфонів у всьому світу продовжує без зупинок зростати і в 2023 році перевищила 4,5 мільярда. Усі ці люди мають доступ до світу ігор у своїй кишені і можуть грати в них практично скрізь, у будь-який час. Мова більше не йде лише про такі прості ігри, як Snake, незважаючи на те, що ця гра стала легендарною. Завдяки потужності та технологіям сучасних смартфонів, ігри на мобільних пристроях стають більш потужним та захоплюючим, ніж будь-коли раніше. Такі компанії, як Razer та Asus, навіть почали випускати спеціальні ігрові телефони з надзвичайними характеристиками та кнопками спеціально для ігор [4].

Мобільні ігри є одним із найпопулярніших сегментів в ігровій індустрії сьогодні. Вони на мобільних пристроях, таких як смартфони та планшети, мають широкий спектр жанрів, від казуальних ігор до складних доступних ігрових досвідом. Ось кілька ключових фактів та інформація про мобільні ігри:

1. Ринок мобільних ігор: Ринок мобільних ігор зростає швидкими темпами і має значний економічний вплив. Для даних дослідницьких компаній, таких як App Annie, вхід від мобільних ігор перевищує вхід від ігрових консолей і ПК.

Розробники і видавці активно працюють над створенням якісних ігор, щоб привернути увагу мобільних гравців.

2. Масова аудиторія: Мобільні ігри приваблюють широкий спектр аудиторії, включаючи як дорослих, так і дітей. Зручність гри на мобільних пристроях, доступність і велика кількість безкоштовних ігор зробили їх популярними серед різних груп користувачів.
3. Монетизація моделі: багато мобільних ігор вибирають модель "free-to-play" (безкоштовна гра з можливістю покупки в середній грі). Це дозволяє гравцям завантажувати та грати в гру безкоштовно, але заробляти гроші через мікротранзакції або рекламу в середині гри. Ця модель монетизації стала популярною і дозволяє розробникам отримувати прибуток, а гравцям володіти більш гнучкими варіантами.
4. Інновації та технології: Мобільні пристрої залишаються все потужнішими, що дозволяє розробникам використовувати передові технології в мобільних іграх. Наприклад, використання віртуальної реальності (VR) та доповненої реальності (AR) у мобільних іграх відкриває нові можливості для імерсивного геймплею та взаємодії.
5. Соціальна взаємодія: Багато мобільних ігор мають функції соціальної взаємодії, які можуть гравцям спілкуватися, змагатися та співпрацювати з іншими гравцями. Це створює соціальну складову гру та розширює її відтворюваність.

Мобільні ігри займають важливе місце в ігровій індустрії та продовжують рости.

Розробники виконують нові технології, створюють захоплюючі геймплеї та залучають різну аудиторію. Вони залишають джерелом розваги, соціальної взаємодії та навчання для мільйонів гравців по всьому світу.

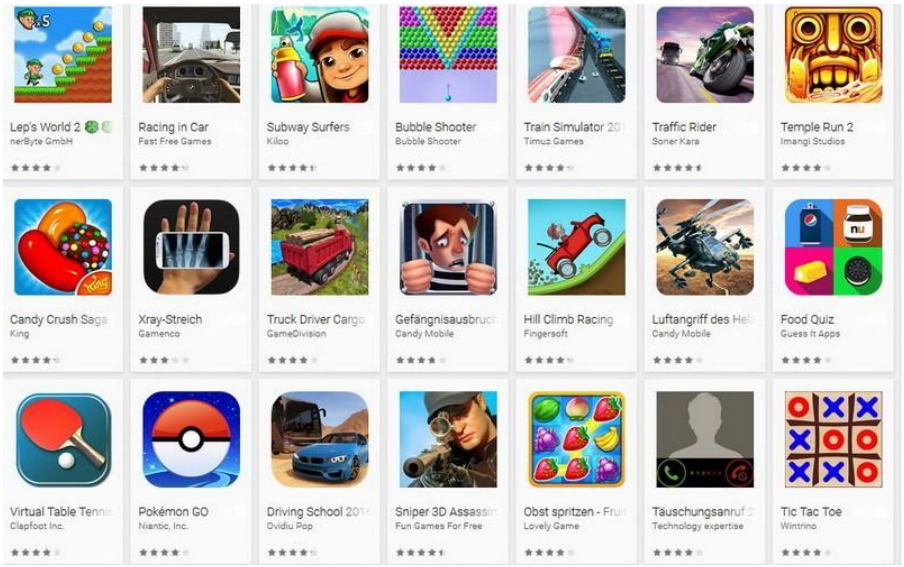


Рис. 1.5. Популярні мобільні ігри.

1.4.2. Комп'ютерні ігри

Основна кількість геймерів твердо переконані, що ПК – це єдиний правильний спосіб грати в ігри. Авжеж, це є не так, але ПК має досить значні підстави, щоб бути найкращою ігровою платформою в світі.

Ігри на ПК пропонують дуже зручний рівень налаштування, що є головною силою його популярності. Є можливість створити свій персональний комп’ютер, який би відповідав вашому колу бюджету та потреб. ПК також можна удосконалити, тому є можливість завжди іти в ногу з сучасними технологіями або модернізувати свій пристрій з часом.

Геймери можуть підняти свої ігрові навички на будь-який бажаний рівень. Можна почати роботу із звичайною установкою або зануритись у світ будівництва своєї збірки, налаштування та вдосконалення.

Крім чудових ігрових можливостей, також варто запам’ятати, що ПК можна також юзати для великої кількості програм, це не є просто пристроєм для ігор. Комп'ютер можна застосовувати як для роботи, так і для вирішення інших завдань. Поганою стороною в цьому порівнянні з консолями є те, що геймерам на ПК

доводиться мати завжди більше технічних знань і досвіду для побудови власної платформи, яка перевершить консоль з таким самим бюджетом.

Головною перевагою для геймерів на ПК є те, що ігри, взагалі завжди, трохи дешевші. Однак деякі ігри є ексклюзивними контентом створеним для консолей, тому ви не маєте можливості грати в них на ПК взагалі.

Переваги ігор на ПК:

- Найкраща доступна продуктивність;
- Ігри на ПК дешевші за консольні;
- Завжди можна налаштувати та оновити;
- Завжди є універсальний засіб застосування.

1.4.3. Консольні ігри

Консоль – це спеціалізований настільний комп’ютер, який ми використовуємо для відеоігор. Дві найпопулярніші консолі – це PlayStation від Sony та Xbox від Microsoft. Wii від Nintendo також є претендентами, які імітують фізичну участь у таких активностях, як боулінг та гра в теніс.

Ігрове програмне забезпечення доступне на компакт-дисках або DVD-дисках, хоча колись ігрові автомати використовували картриджі, що містять чіпи пам'яті (тільки для читання). Для відображення відеоігрових консолей потрібен телевізор або монітор.

Відеоігрові консолі зазвичай працюють від операційних систем і процесорів, що відрізняються від настільних ПК. Консолі керуються відповідними виробниками, а програмне забезпечення інтегроване до можливостей системи. Ігри не заміниш з інших ігрових систем такі як приставки або настільними комп’ютерами, хоча розробники програмного забезпечення можуть розробляти ігри для кількох платформ [5].

Розробка консольних ігор включає в себе використання різних наукових принципів та технологій. Ось кілька основних аспектів, які застосовуються в сучасній науково обґрунтованій розробці консольних ігор:

1. Графіка та комп'ютерна графіка: Розробники ігор вибирають принципи комп'ютерної графіки для створення реалістичних візуальних ефектів у грі. Вони використовують такі наукові принципи, як трасування змін (tray tracing), освітлення, моделювання фізичних матеріалів та багато іншого, для досягнення реалістичного вигляду гри.
2. Фізика: Багато ігор базуються на принципах фізики для створення реалістичної поведінки об'єктів у грі. Фізичні двигуни, такі як PhysX або Havok, не дозволяють розробникам моделювати рух, гравіт, столкнувшись з об'єктами та інші фізичні ефекти.
3. Штучний інтелект: Розробники ігор створення технології штучного інтелекту для створення поведінки комп'ютерних персонажів у грі. Штучний інтелект може включати алгоритми машинного навчання, які дозволяють персонажам реагувати на дії гравця, прийняти рішення та навчатися з досвідом.
4. Звуковий дизайн: Використання звукових ефектів та музики в іграх також базується на наукових принципах. Звукові інженери вивчають акустику, психоакустику та інші аспекти звукового дизайну, щоб створити ефективні звукові ефекти та імерсивний звуковий досвід для гравців.
5. Мережева технологія: Багато сучасних консольних ігор підтримують режими мережевої гри, що дозволяють гравцям грати разом у режимі онлайн. Це вимагає використання наукових принципів мережевих технологій, таких як маршрутизація, оптимізація пропускної здатності та безпеки, для забезпечення стабільного та безперебійного геймплею.

Важливо відзначити, що розробка ігоря по цьому науковий підхід з творчим процесом. Розробники виконують наукові принципи, але також враховують ігровий дизайн, естетику та інші аспекти, щоб створити захоплюючі та цікаві ігрові враження.

1.4.4. Портативні консольні ігри

Портативні відеоігри – це мініатюрні версії ігрових приставок і менш складні. Це повністю портативні автономні пристрої з батарейками та власними невеликими екранами.

Розробка портативних консольних ігор також базується на наукових принципах та технологіях. Ось кілька аспектів, що застосовуються в науково обґрунтованій розробці портативних консольних ігор:

1. Енергоефективність: Оскільки портативні консолі мають обмежену потужність акумулятора та ресурсів, розробники ігор вводяться до наукових методів для оптимізації продуктивності та енергоефективності гри. Вони використовують техніку стиснення даних, оптимізований рендеринг, керування ресурсами та інші підходи для забезпечення оптимальної продуктивності на портативних пристроях.
2. Мобільні технології: Розробка портативних ігор включає в себе використання мобільних технологій, таких як сенсорні екрани, акселерометри, гіроскопи та GPS. Розробники використовують ці технології для створення інтерактивного геймплею, додавання жестів керування та мобільних функцій у гру.
3. Оптимізований контент: Розробники портативних ігор звертаються до особливої уваги розміру та обсягу контенту. Вони використовують наукові методи для натискання текстур, звуків та інших медіа-елементів, щоб забезпечити найкращу якість гри при обмежених ресурсах пристрою.
4. Мережева гра та спільна гра: багато портативних консолей підтримують мережеві функції, які дозволяють гравцям грати одночасно в режимі онлайн або взаємодіяти з іншими користувачами. Розробники розробки наукових принципів мережевих технологій та алгоритмів забезпечення стабільної мережевої гри та оптимізації передачі даних.
5. Інтерфейс та взаємодія: Портативні консолі мають свої особливості щодо інтерфейсу та взаємодії з гравцем. Розробники вивчають наукові принципи людсько-комп'ютерного взаємодії, щоб створити зручний та інтуїтивно зрозумілий інтерфейс, який можливо відповідає портативним пристроям.

Рис. 1.6. Портативна консоль Playstation portable

1.4.5. Ігри з VR реальністю

Ігри VR – це найновіша тенденція на ринку. І консолі, і ПК мають свої позиції на цьому ринку. Серед консолей лише PS4 насправді пропонує гарнітуру VR, тоді як ПК має кілька можливих гарнітур, якими можна користуватися, хоча HTC Vive та Oculus Rift є найпопулярнішими та найпотужнішими.

З технічної точки зору PS VR поступається пропозиціям ПК, як через свою застарілу технологію відстеження, так і сам PS4 менш потужний. Більше того, ігор для PS VR менше, що робить його менш привабливим продуктом загалом. Для VR також рекомендується використовувати більш потужний ПК [6].

Не можна визначити одну ігрову платформу, яка буде абсолютно кращою. Це дійсно залежить від очікувань та цілей кінцевого користувача.

Якщо потрібно мати доступ до ігор, знаходячись поза домом, мобільні ігри підходять найкраще.

Якщо на меті є грати в ігри на максимальних потужностях пристрою, краще підійде ігровий ПК.

Якщо потрібна спеціальна ігрова машина, що буде простою у використанні, матиме багато ігор і навіть ексклюзивні ігри, імовірно, варто обрати к ігрову консоль.

РОЗДІЛ 2. Інструменти та методи реалізації модулів

Unreal Engine 5 містить кілька батьківських класів, за допомогою яких гру можна доповнити новими властивостями.

Допомогою яких гру можна доповнити новими властивостями. Від батьківських класів створюються об'єкти, що успадковують властивості відповідного класу. Налаштовані класи можуть бути використані в подальшому розробленні будь-якого іншого проєкту необхідні для того, щоб якнайшвидше прийти від ідеї до виходу в продакшн [7].

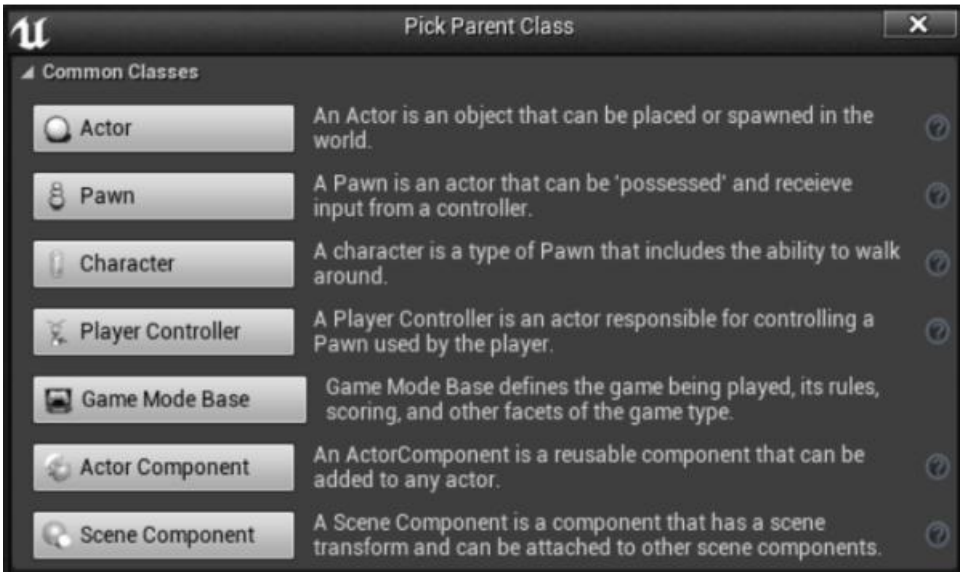


Рис. 2.1. Класи

Залежно від обраного батьківського класу, успадкованому об'єкту будуть притаманні відповідні характеристики [12]. Наприклад, які дії доступні об'єкту, зовнішній вигляд, відсутність або наявність скелета й анімації пересування, чи повинен управлятися об'єкт, ким і як він має управлятися, чи має об'єкт почуття - слух, зір. Також є можливість створювати Blueprints, які будуть успадковуватися від створених вами Blueprints. Таким чином, можна створити свій Blueprint із певним функціоналом, а потім використовувати його як батьківський клас для надання цієї функціональності іншим Blueprints

Класи в Unreal Engine 5 та їхній опис

Батьківські клас	**Опис призначення**
Actor (Актор)	Актори - їх можна створювати в грі або розміщувати в неї з ресурсів. неї з ресурсів. Це загальний клас, що підтримує 3D-перетворення (переміщення, обертання, масштаб). Можуть створюватися і видалятися програмним кодом (C++ або Blueprints)
Pawn (Пішак)	Актори, якими може керувати штучний інтелект або гравець. Може керуватися гравцем або штучним інтелектом (ШІ), в останньому випадку являючи собою т. зв. НІПа (не-ігрового персонажа, NPC).
Character (Характер)	Класи, які можуть ходити і можуть переміщатися по рівнях. Це клас пішака, але з додатковими функціями, такими як скелетна сітка, додана за замовчуванням.
PlayerController (Контролер управління гравцем)	Клас відповідає за контроль над Pawn, отримання команд і виконання певних дій.
Game Mode	Клас ігровий режим дає змогу встановлювати параметри ігрового світу, правила і тип гри.

2.1. Actor

Ймовірно, найбільш використовуваний клас в іграх. Актор є основою кожного об'єкта на рівні, включно з гравцями, керованими ворогами ШІ, дверима, стінами та об'єктами ігрового процесу. Актори створюються з використанням ActorComponents , таких як StaticMeshComponent, CharacterMovementComponent, ParticleComponent і багато інших. Навіть такі класи, як GameMode є акторами (хоча GameMode не має "реальної" позиції в світі). Давайте обговоримо кілька аспектів, які вам потрібно знати про акторів.

Actor - це клас, який можна реплікувати в мережі (у багатокористувацькому режимі). Це можна легко зробити за допомогою виклику в конструкторі SetReplicates (true). Щоб створювати ефективні мережеві розробники, потрібно враховувати безліч аспектів.Актори підтримують концепцію травми. Збиток може бути завдано безпосередньо на актора, використовуючи MyActor-> TakeDamage (...) або UGameplayStatics :: ApplyDamage (...). Варто враховувати, що є різновиди: PointDamage (наприклад, для зброї, з якої розраховується удар за трасуванням променів (hitcan)) і RadialDamage (наприклад, для вибуху). На офіційному сайті Unreal Engine є чудова вступна стаття Damage в UE5.

Ось деякі корисні функції, які використовуються в контексті

актора:BeginPlay // "Перша" функція, що викликається після створення і повної ініціалізації актора. Це зручне місце для встановлення базової логіки, годинника і внесення змін до властивостей, оскільки суб'єкт повністю ініціалізований і може виконувати запити до свого середовища.Вибрати // Викликати кожен кадр. Для більшості акторів ви можете відключити його з міркувань продуктивності, але він увімкнено за замовчуванням. Він чудово підходить для швидкого налаштування динамічної логіки та перевірки умов у кожному кадрі. Поступово ви почнете передавати все більше і більше коду, пов'язаного з логічними подіями, з таймерів у логіку, що працює на нижчих частотах.

EndPlay // Викликається, коли актор видаляється зі світу. Містить "EEndPlayReason", що вказує причину з'єднання.

GetComponentByClass // Знаходить один екземпляр компонента зазначеного класу. Це дуже корисно, коли ви не знаєте точний тип актора, але знаєте, що він повинен містити компонент певного типу. Існує також GetComponentsByClass, який повертає всі екземпляри класів, а не тільки перший знайдений.

GetActorLocation // І всі його варіанти - * Обертання, * Масштаб, включаючи SetActorLocation тощо.

NotifyActorBeginOverlap // Зручно для перевірки накладок, викликаних будь-яким із його компонентів. Таким чином, ви можете швидко налаштувати тригери гри.

GetOverlappingActors // Знаходить, які інші актори перетинаються з обраним. Існує також опція для компонентів: GetOverlappingComponents

Актор має величезний функціонал і безліч змінних - це основа гри в Unreal Engine, тому це не дивно. Для подальшого аналізу цього класу було б добре відкрити файл заголовка Actor.h у Visual Studio і подивитися, які функції він має. Нам ще багато чого належить розглянути в цій статті, тому давайте перейдемо до наступного класу в списку.

2.1.1. ActorComponent

Компоненти знаходяться всередині акторів, стандартними компонентами є StaticMeshComponent, CharacterMovementComponent, CameraComponent і SphereComponent. Кожен із цих компонентів виконує своє конкретне завдання, наприклад, рух, фізичну взаємодія (наприклад, кількість зіткнень для чіткої перевірки взаємодій акторів) або візуально відображає щось у світі, наприклад сітку гравця.

Підклас цього компонента - SceneComponent - це базовий клас для усього, що пов'язано з перетворенням (положення, обертання, масштаб), яке підтримує

закріплення. Наприклад, ми можемо прикріпити CameraComponent до SpringArmComponent, щоб налаштувати камеру третьої особи. І перетворення, і прикріплення необхідні для правильної встановлення відносного положення.

Найчастіше компоненти створюються в конструкторі актора, але вони також можуть бути створені та знищені під час виконання.

Ці елементи, поряд з Actor, мають вирішальне значення для створення ігор на C ++ і проєктів. Це ігрові кубики. Ви можете легко створювати свої власні компоненти для підтримки певних специфічних аспектів гри, таких як HealthComponent, який зберігає очки здоров'я і реагує на пошкодження. бали здоров'я і реагує на шкоду, завдану його батьком.

2.1.2. PlayerController

Це базовий клас для гравця, який отримує дані від користувача. Сам PlayerController не відображається візуально в середовищі, але керує екземпляром Pawn, який визначає візуальне та фізичне представлення цього гравця у світі. Під час гри у гравця може бути кілька різних фігур (наприклад, транспортний засіб або свіжа копія фігури під час переродження), і екземпляр контролера гравця залишається незмінним протягом усього рівня. Це важливо, тому що у деяких точках у гравця може не бути пішака. Це означає, що такі речі, як-от відкриття меню, мають бути додані в PlayerController, а не у клас Pawn.

У багатокористувацьких іграх PlayerController існує тільки на клієнті, якому він належить, і на сервері. Це означає, що в грі із 4 гравцями на сервері встановлено 4 ігрові контролери, а в кожного клієнта - тільки один. Дуже важливо розуміти, коли необхідно використовувати змінні; якщо всі гравці вимагають реплікації змінної гравця, вона має існувати не в PlayerController, а у вертикальному або навіть PlayerState.

2.1.3. Pawn

Це фізичне та візуальне представлення того, що контролює гравця (або ШІ). Це може бути машина, воїн, вежа або щось іще, що означає характер гри. Стандартний підклас Piona - це персонаж, який реалізує SkeletalMesh і, що більш важливо, CharacterMovementComponent із безліччю опцій для точного налаштування руху гравця відповідно до навколишнього середовища за допомогою звичайного стрілка.

У багатокористувацьких іграх кожен екземпляр Дивізіону копіюється на інших клієнтів. Це означає, що в грі є 4 екземпляри пішаків для 4 гравців як на сервері, так і на кожному клієнті. Досить часто екземпляр Pawn "вбивається", коли гравець помирає, а респаун створює новий екземпляр. Пам'ятайте про це, зберігаючи дані, які мають бути збережені після життя гравця (або повністю відмовитися від цієї схеми і постійно підтримувати екземпляр пішака).

2.1.4. GameModeBase

Базовий клас, який визначає, які класи використовувати (PlayerController, Pawn, HUD, GameState, PlayerState). Часто використовується для встановлення правил гри в таких режимах, як "Захоплення прапора"; може поводитися з прапорами або хвилями ворогів. Він підтримує інші важливі функції, такі як створення гравця.

GameMode - це підклас GameModeBase. Він містить у собі кілька додаткових функцій, які спочатку використовувалися в Unreal Tournament, таких як MatchState та інші функції шутера.

У мультиплеєрі клас GameMode існує тільки на сервері! Це означає, що жоден клієнт не має такого екземпляра. В одиночних іграх це не має жодного ефекту. Ви можете використовувати GameState, який існує на всіх клієнтах і спеціально розроблений для цієї мети, для реплікації функцій і зберігання даних, необхідних для GameMode.

2.1.5. HUD

Це клас користувацького інтерфейсу. Він містить багато коду Canvas, який являє собою користувацький інтерфейс, що відображає код, написаний до UMG. відображає код, написаний до появи UMG.

Сьогодні основна робота з відтворення користувацького інтерфейсу ведеться в UMG.Клас існує тільки в клієнті. Реплікація неможлива.

Є власником PlayerController.

Доступ до HUDPlayerController-> GetHUD () // Доступно в локальному

PlayerController.

Створення.

Він створюється за допомогою SpawnDefaultHUD (створює звичайнийAHUD) всередині PlayerController, який володіє HUD, а потімперезаписує GameModeBase з використанням класу HUD InitializeHUDForPlayer, зазначеного в GameModeBase.

2.1.6. PlayerState

Контейнер для змінних, що реплікуються між одним клієнтом - сервером гравця. У багатокористувацьких іграх він не призначений длявиконання логіки і просто контейнер для даних, оскільки PlayerController не доступний для всіх клієнтів, і пішак часто руйнуються, коли гра гравець помирає, тому не відноситься до даних, які повинні бути збережені після смерті

Доступ до PlayerState.

Поділ містить його як змінну Pawn-> PlayerState, також доступну в Controller-> PlayerState. Вертикальний статус гравця присвоюється тільки тоді, коли пішак є власником контролера, в іншому випадку це nullptr.

Список усіх доступних екземплярів PlayerState (наприклад, усіх гравців у матчі) можна отримати через GameState-> PlayerArray.

СтворенняКлас створення призначається в GameMode (PlayerStateClass) і створюється в AController :: InitPlayerState ()Рекомендується використовувати тільки під час роботи в багатокористувацьких іграх.

2.1.7. GameStateBase

Схоже на PlayerState, але надає клієнтам інформацію про GameMode. Оскільки екземпляр GameMode існує не на клієнтах, а тільки на сервері, цей клас є корисним контейнером для реплікації такої інформації, як кінець матчу, командні очки тощо.

Він має два різновиди - GameState і GameStateBase. GameState підтримує додаткові змінні, необхідні для GameMode (на відміну від GameModeBase)

Доступ до GameStateBase

World-> GetGameState <T> () // де T - клас, що викликається, наприклад GetGameState <AGameState> ()

MyGameMode-> GetGameState () // зберігається і доступний в екземплярі ігрового режиму (потрібен тільки на сервері, який володіє одним екземпляром GameMode); клієнти повинні використовувати вищевказане з'єднання.

Рекомендується використовувати GameStateBase замість GameState, тільки якщо gamemode не успадкований від GameMode замість GameModeBase.

Основні класи, що використовуються в розробці модуля:

- Actor - використаний для створення елементів, що поповнюють

здоров'я, предметів, що випадають після усунення супротивників, а також зброї, який тримає в руці персонаж.

- Character - з його допомогою створено персонажа, яким керує

гравець, а також противники, керовані комп'ютером.

- Game Mode - тут створено глобальні змінні.

Класи, що використовуються для створення об'єктів, утворюють дерево успадкування в такому порядку: Object, Actor, Pawn, Character.

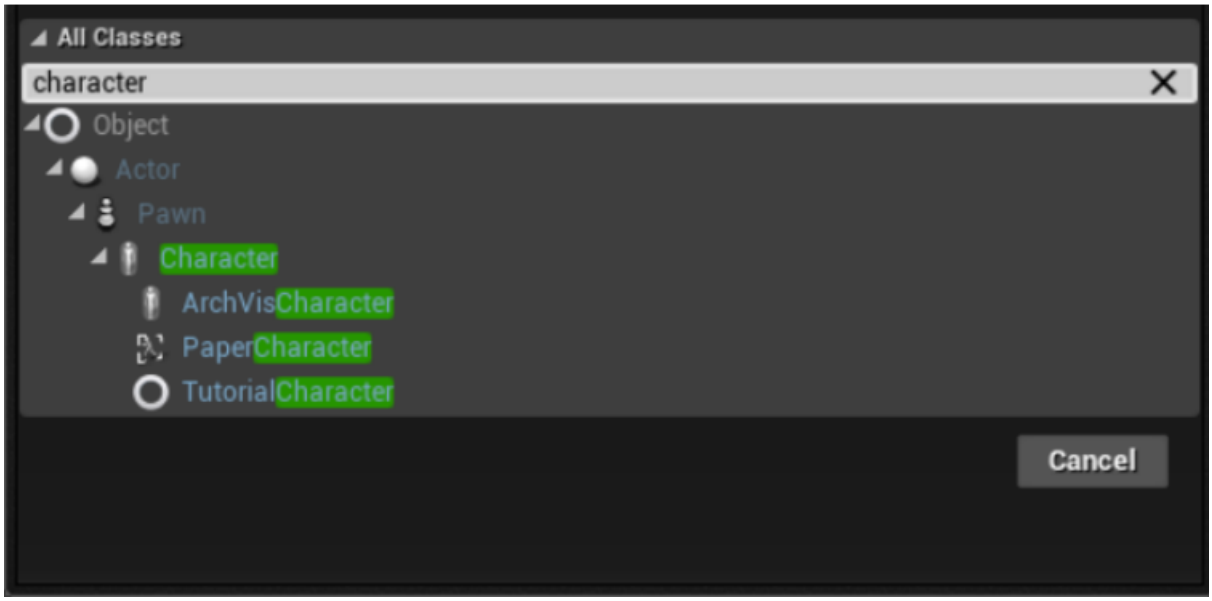


Рис. 2.2. Дерево успадкування об'єктів

2.2. Методи створення проектів

UE надає два методи для створення елементів геймплею - C++ і Blueprint [13][14]. C++ програмісти додають основні блоки геймплея, таким чином, щоб дизайнери рівнів могли створювати свої елементи гри для окремого рівня або всієї програми загалом. У такому випадку, програмісти працюють в IDE (наприклад - MS Visual Studio, Xcode), а дизайнер працює у Blueprint редакторі Unreal Engine 5.API геймплею та фреймворк класів повністю доступні з обох систем. Обидві системи можна використовувати окремо, але використовуючи їх разом виходить більш потужна і гнучка система. Це означає, що найкращою практикою буде злагоджена робота програмістів, які створюють основи ігрового процесу та дизайнерів рівнів, які використовують ці блоки для створення захопливої гри [15][16].

Найперше, що потрібно зробити, це скористатися майстром класів (class wizard), що надається Unreal Engine 5, для створення бази майбутнього C++ класу, який надалі буде розширено за допомогою Blueprint[17][18].

Після створення класу, майстер генерує файли і відкриє IDE, таким чином що можна відразу почати редагувати його.

Майстер класів генерує клас із методами BeginPlay() і Tick(), зі специфікатором перевантаження (override). Подія BeginPlay() відбувається коли Actor входить у гру, у стані, дозволеному для гри (playable state)[19]. Доброю практикою є ініціювання геймплей-коду класу в цьому методі. Метод Tick() викликається кожен кадр із параметром, який дорівнює часу, що минув з останнього свого виклику. У цьому методі повинна міститися постійно повторювана логіка. Якщо ж вона відсутня, то найкраще буде прибрати цей метод, що трохи збільшить продуктивність. Під час видалення коду цього методу потрібно переконатися, що також видалено рядок у конструкторі класу, який вказує, що Tick() має викликатися кожен кадр.

Під час створення класу Unreal ставить перед його назвою префікс. К наприклад, створюючи клас типу Actor, Unreal додасть префікс A (від слова actor). Щоб система рефлексії могла працювати, їй потрібно, щоб класи мали відповідні префікси.

SetupPlayerInputComponent() - після цієї функції проводиться оголошення змінних. Також, використане в цій функції ім'я буде відображено як ім'я компонента в редакторі.

UPROPERTY() - спеціальний макрос, який потрібно використовувати, щоб властивість було відображено в редакторі. Усе, що потрібно зробити, це написати макрос UPROPERTY(EditAnywhere) перед оголошенням змінної. Також можна додати до UPROPERTY() описувачі (specifiers), які керуватимуть поведінкою змінної в різних аспектах рушія. Описувачі VisibleAnywhere і BlueprintReadOnly використовуються для змінних-покажчиків.

VisibleAnywhere робить кожен компонент видимим у редакторі (зокрема в числі й у Blueprints) [22]. BlueprintReadOnly дає змогу отримувати посилання на компонент за допомогою нодів Blueprint. Але, в той же час, за допомогою нього неможливо задавати компонент. Що дуже важливо, компоненти мають бути

позначені як "тільки для читання" (read-only), тому що їхні змінні є покажчиками. Категорично не потрібно, щобкористувачі задавали їх, інакше є ризик, що вони можуть вказати на випадкове місце в пам'яті. Варто зауважити, що BlueprintReadOnly дозволяє задавати змінні всередині компонента. Для змінних, які не є покажчиками (int, float, boolean тощо) рекомендується використовувати EditAnywhere і BlueprintReadWrite.

Далі, коли змінні оголошені, їх потрібно ініціалізувати.

Для цього необхідно створити їх усередині конструктора. Один із варіантів функцій, що використовуються для створення компонентів - CreateDefaultSubobject<Type>("InternalName"). Ця функція створює компонент кожного типу, а потім призначає їхні адреси в пам'яті переданій змінній. Аргумент-рядок буде призначено внутрішнім ім'ям компонента (використовуваним движком, а не відображуваним ім'ям).

Потім потрібно налаштувати ієрархію (вибрати кореневий компонент, наступний, прикріплений до нього, і так далі), за допомогою RootComponent і SetupAttachment(<ім'я компонента>). RootComponent -компонент, який визначає перетворення (місце розташування, обертання, масштаб) цього актора у світі, всі інші компоненти мають бути якимось чином прив'язані до нього.

Потім потрібно вказати, який міш буде використано і поворот пружинного важеля. Рекомендується робити це в Blueprints, тому що небажано жорстко вказувати шляхи до ресурсів у C++, тому що під час переміщення ресурсу в іншу папку, логіка в Blueprints не порушиться, в той час, як у C++ коді доведеться поміняти кожне посилання на цей ресурс.

Для завдання повороту міша і пружинного важеля в Blueprints, необхідно створити Blueprint на основі BasePlayer [26].

2.3. Основні функції та методи

Події (Event) - це вузли, які визначають початок конкретної логічної послідовності та викликаються з коду ігрового процесу, щоб почати виконання окремої мережі в Event Graph. Вони дають змогу Blueprints виконувати низку дій у відповідь на певні події, що відбуваються в грі, такі як початок гри, скидання рівня, отримання шкоди тощо. Ці події можуть бути доступні через Контекстне меню в Blueprints для реалізації нових функцій, перевизначення або розширення функціональності за замовчуванням. Будь-яка кількість подій може бути використана в одному Event Graph; хоча може використовуватися тільки один із кожного типу [20].

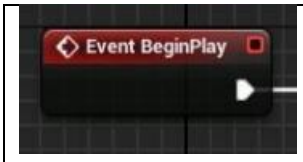
Редактор Blueprints уже містить безліч вузлів, що обробляють різноманітні події, але існує можливість створити і власні вузли обробки подій. Наприклад, натискання на певну клавішу.

Для цього потрібно в налаштуваннях проєкту створити новий Input Axis, задати для нього нього клавішу на клавіатурі, миші, або контролері, а потім створити відповідний вузол у редакторі EventGraph. Далі вибудувати логіку обробки події, і, після натискання на задану заздалегідь клавішу, будуть виконуватися заздалегідь описані дії.

Потрібно розуміти, що кожна окрема подія викликається в окремому об'єкті і якщо потрібно, щоб виконувалося кілька дій їх потрібно викликати паралельно. Для цих цілей у Blueprints вже існує вузол Event Dispatcher. Під час його вибору можна встановити категорію, додати опис, а також додати вхідні параметри. Усі прив'язані до нього вузли будуть викликані в той самий момент, коли викликано вузол Event Dispatcher. Додавання вхідних даних у ваш диспетчер подій дає змогу вам надсилати змінні кожній події, пов'язаній із вашим диспетчером подій. Це дає змогу передавати дані не тільки в межах вашого Blueprint Class, але також між вашим Blueprint Class і Level Blueprint. Створивши диспетчер подій, ви можете додавати вузли подій, пов'язувати вузли і пов'язувати вузли, пов'язані з ним. У плані рівня ви можете встановити спеціальний тип події для Диспетчера подій, і ця подія

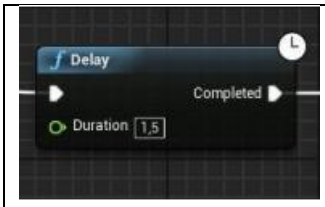
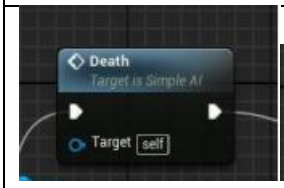
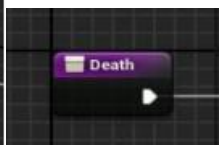
буде негайно пов'язана з ним. Ці події створюються так само так само, як і звичайні події (наприклад, шляхом Overlap або Clicked). Ці події автоматично пов'язуються з початком гри. У результаті викликаний блок Unbind All поверне всі події, але вони можуть бути повторно пов'язані з подією Bind.

Вузли подій у Blueprints

	Ця подія запускається для всіх Акторів, коли гра починається, будь-які актори,породжені після запуску гри, будуть негайно викликані.
	Це подія, яка викликається на кожному кадрі ігрового процесу. Дельта секунд Float – виводить кількість часу між кадрами.
	Зіставлення дій призначені для обробки натискань і відпускань клавіш.
	Подія викликається, коли щось починає компонент, наприклад, гравець, що йде в тригер. Для подій, коли об'єкти мають блокуюче зіткнення, наприклад, гравець вдарився об стіну.
	Подія викликається, коли гравець входить у зону тригера штучного інтелекту.

Також у редакторі Blueprints є функції. У візуальній логіці вони представляються у вигляді вузлів схеми. Функції мають одну точку входу і одну точку виходу. Усередині функції може міститися будь-який набір вузлів - елементів графічної логіки. Під час створення функції є можливість задати певний тип доступу. Найчастіше функція обробляє будь-які дані, тому у вкладці Details (деталі) можна задати, які саме параметри будуть вхідними і вихідними даними.

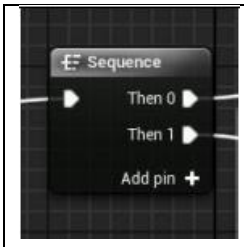
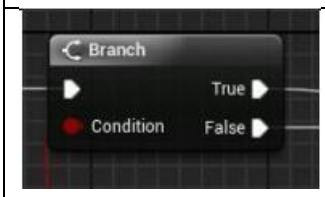
Функції в Blueprints

	Затримка. Виконайте приховану дію із затримкою (вказується в секундах). Повторний виклик під час зворотного відліку буде ігноруватися. На зображенні тривалість затримки (у секундах) 1.5.
	За допомогою цього вузла ми можемо знищити будь-якого актора, який буде вказано.
	Це функція приведення ВР (класу) до типу, який вказано як параметр. Використовуючи її, можна викликати кастомні функції actor-об'єктів, якщо невідомо заздалегідь, якого типу actor-об'єкт.
 	Також є можливість створювати власні функції. При подвійному натисканні лівою кнопкою миші на

	вузол функції, відкривається вікно з вузлом початку функції, у цьому вікні потрібно описати її роботу.

Також у редакторі Event Graph є функції, схожі з операторами мови C++. Наприклад, функція Sequence схожа з конструкцією оператора "switch": вона послідовно виконує прив'язані до виходів вузли.

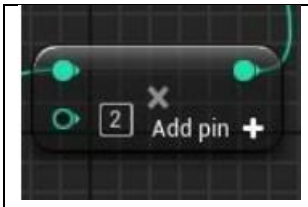
Вузли в Blueprints

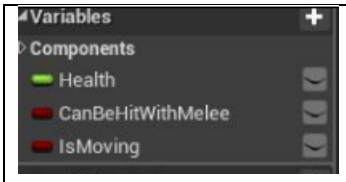
	Під час виклику функції вона послідовно викликає сигнали на виходах даної функції. Ці виклики завжди виконуються в порядку їх позначення 1 потім 2 потім 3 і так далі. За допомогою Add pin можна додавати вузли.
	Оператор розгалуження Якщо Умова істинна, виконання переходить до Істини, інакше - Брехня.
 	Встановлення будь-якої змінної рівною якомусь значенню. Вузол має певний колір і набір входів і виходів залежно від типу змінної, з якою він працює.

	Вузол Get. Отримати значення змінної.

Або функція Branch, вона схожа за будовою з умовою "if": всередині функції виконується перевірка на правдивість умови, і залежно від від результату перевірки, вибирається вихід функції, за яким продовжиться обробка події. Або, один вихід залишається незадіяним, у цьому випадку перевірка проводиться доти, доки не буде обрано вихід, у якому описано подальші дії. Редактор має і вузли, що виконують арифметичні та логічні операції. і логічні операції. Якщо їх виявляється недостатньо, то використовується окремий вузол - блок Math Expression, що дає змогу написати математичну формулу, після чого графік Blueprint буде згенеровано автоматично. Перевага цього функціоналу полягає не тільки в тому, що можна швидко створити математичний вираз, а й у продуктивності. Блоки Math Expression оброблятимуться швидше під час гри, ніж описані вручну вирази зі звичайних математичних блоків у Blueprint.[]

Операція в Blueprints

	Так виглядає вузол, що перемножує 2 значення. Можна задати значення безпосередньо, або отримати з будь-якої змінної
	Так виглядає вузол, що підсумовує 2 значення. Можна задати значення безпосередньо, або отримати з будь-якої змінної.

	Змінні. Кожен тип даних має свій колір. Залежно від того, до якого типу даних належить змінна, з якою працює вузол,змінюється колір вузла. якою працює вузол, змінюється колір вузла.

Ім'я змінної може бути будь-яким, але важливо пам'ятати такі моменти: імена змінних можуть містити числа, але ім'я не повинно починатися з цифри; ім'я змінної не повинно збігатися з ім'ям прихованої змінної Blueprint; також потрібно бути уважним при виборі типу змінних, наприклад, boolVar+1.5 (додавання булевої змінної з дробовою) є неприпустимим виразом.

РОЗДІЛ 3. Проектування та розробка гри

3.1. Реалізація гри

Для реалізації поставлених завдань і реалізації програмного модуля було обрано вбудовану в Unreal Engine 5 візуальну скриптову мову - Blueprints. мову - Blueprints.

Основа гри в Unreal Engine 5 надає розробникам потужний набір класів для створення гри. Проєкт може бути шутером, симулятором ферми, глибокою РПГ - це не має значення, фундамент дуже різнобічний, він виконує багато важкої роботи для вас і встановлює деякі стандарти.

Він досить інтегрований із рушієм, тому рекомендується дотримуватися цих класів замість того, щоб того, щоб заново створювати власну ігрову базу даних, як це часто буває з рушіями, такими як Unity3D.

Розуміння цієї основи дуже важливе для успішної та успішної роботи проєкту.[8]

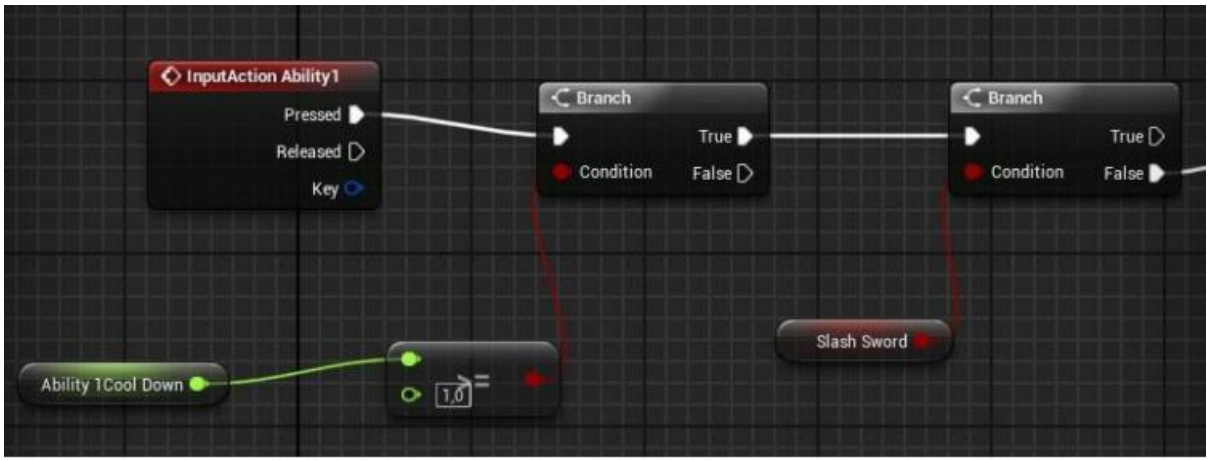


Рис. 3.1. Приклад Blueprints в Unreal Engine 5

Основні переваги Blueprints порівняно з C++ полягають у тому, що, по-перше, Blueprints, також, як і багато мов програмування, серед яких і C++, дає змогу реалізовувати програми за допомогою об'єктно-орієнтованого програмування.

Це дає змогу представити програму в найзручнішому вигляді, створюючи ієрархію класів.

По-друге, Blueprints дає змогу створювати нові класи без написання і компіляції коду, що, своєю чергою, скорочує час збирання проєкту, а також усуває затримки в роботі.

По-третє, скрипти на Blueprints дуже швидко компілюються, порівняно з кодом мовою C++. Це полегшує тестування і перевірку правильності роботи різних функцій.

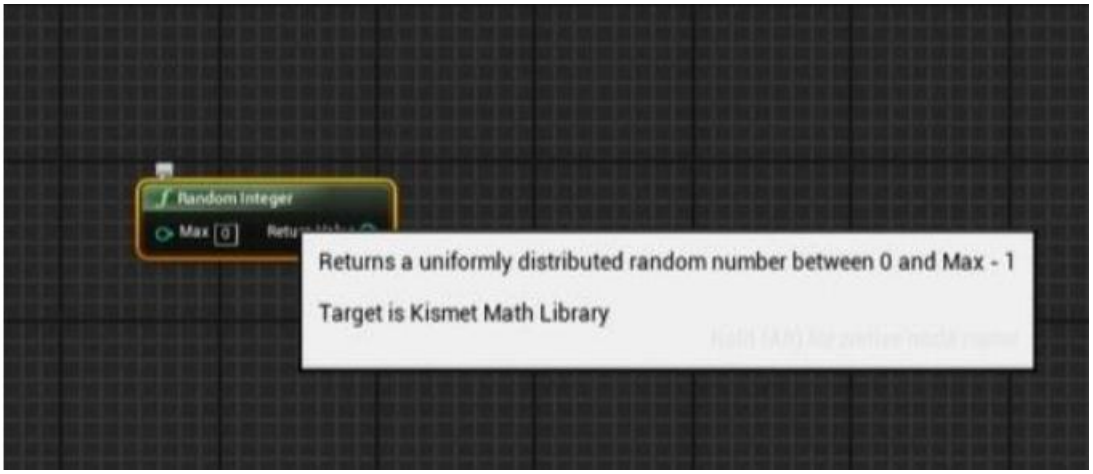


Рис. 3.2. Функція Random Integer знаходиться в бібліотеці Math

Також, Blueprints і C++ містять однаковий API (програмний інтерфейс додатка, інтерфейс прикладного програмування) (англ. application programming interface), тобто вони мають ідентичні процедури, класи та функції.

Перераховані вище властивості доводять, що код, написаний на C++, може бути представлений у вигляді Blueprints і сприяти найшвидшій та найефективнішій роботі алгоритму.

За допомогою Blueprints розробники можуть створювати такі речі,як:

- Ігрові режими - встановлювати правила гри, змінювати поведінку гри в загальному плані.
- Гравці - призначати гравців, надавати їм особливих рис і вигляду.
- Камери - створювати види для огляду і змінювати властивості камер у реальному часі.
- Керування - призначати клавіші для керування персонажем, автомобілем, або зовнішнім рівнем.

- Речі - зброя, предмети, що підбираються, та інше.

- Оточення - створення випадково генерованого оточення.

3.2. Опис логіки елементів гри

Для опрацювання переходу персонажа з одного стану в інший і відтворення відповідних анімацій реалізовано кінцевий автомат, з використанням вбудованих інструментів.

Кінцевий автомат - це набір станів і правил. Кінцеві автомати одночасно можуть перебувати тільки в одному стані. Для переходу в наступний стан мають виконуватися певні умови, що задаються правилами[8].

Стани можуть мати і двосторонній взаємозв'язок.

Нижче представлено кінцевий автомат, побудований під час роботи

Спочатку потрібно обрати тип проекту.

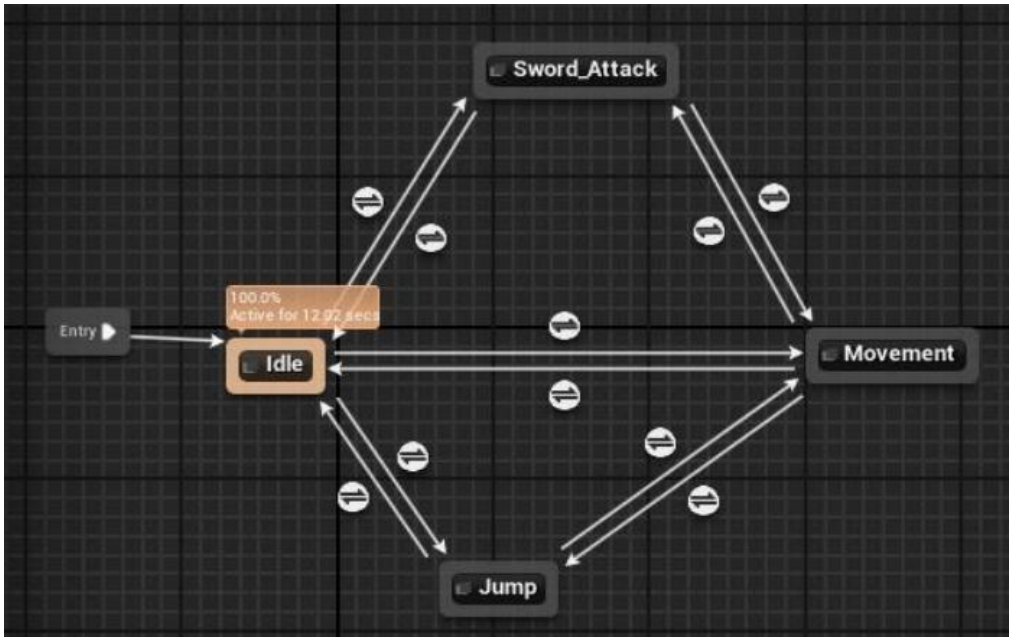


Рис. 3.3. Кінцевий автомат

У цьому скінченному автоматі описано переходи між такими станами:

- Бездіяльність (Idle).

- Рух (Movement).

- Стрибок (Jump).

- Атака зброєю (Sword Attack).

Вхід позначений подією Entry, що перекладається як "вхід".

З'єднаний із цим вузлом стан є станом за замовчуванням.

У програмі присутні такі частини:

- Удар мечем.
- Збір предметів.
- Збір елементів, що поповнюють здоров'я персонажа.
- Відновлення здоров'я.
- Шкода здоров'ю противника.
- Загибель противника.
- HUD - частина графічного користувацького інтерфейсу.
- Найпростіший штучний інтелект

Рис. 3.4. Предмет в грі

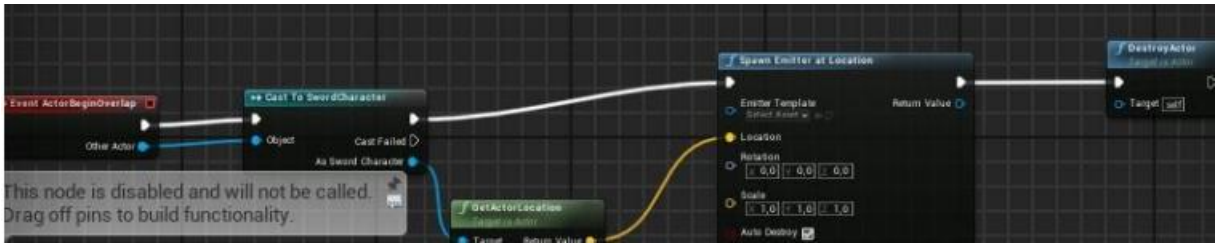


Рис. 3.5. Логіка збору предмету

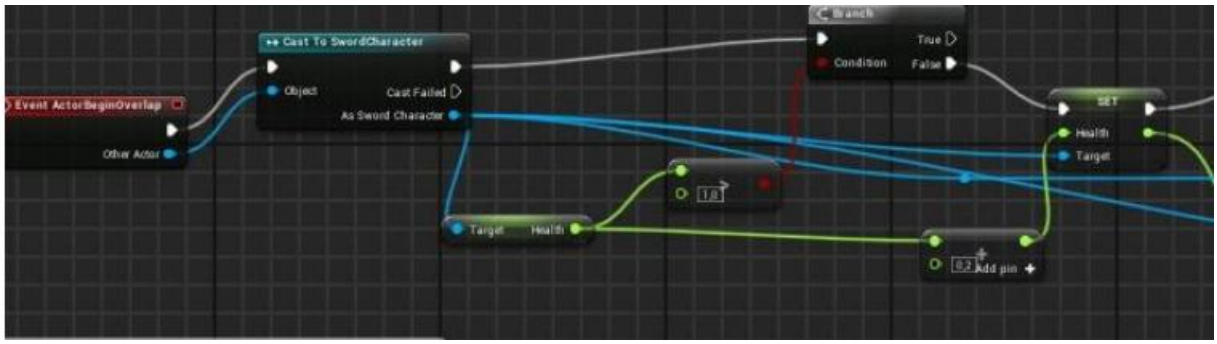


Рис. 3.6. Логіка збору предмету що поповнює життя

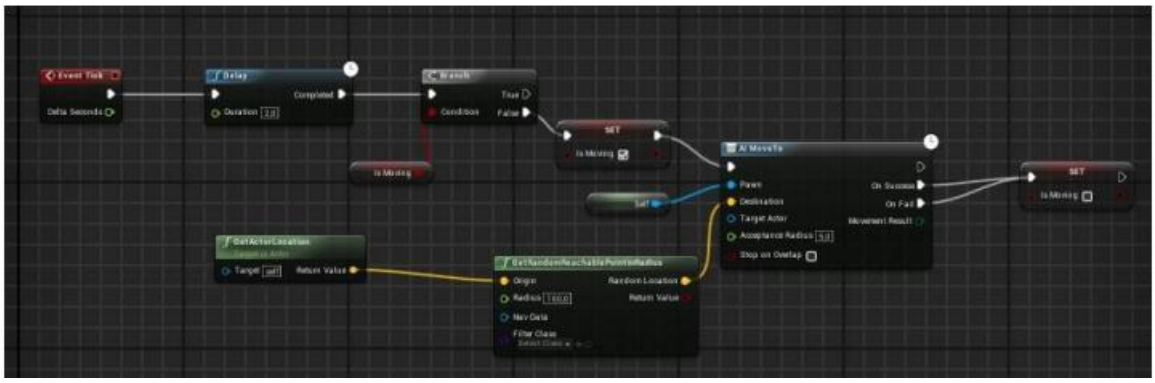


Рис. 3.7. Логіка переслідування противником персонажа

Ворог переслідує персонажа в усій області карти, позначеній для нього доступною для переміщення. Починає переслідування відразу після того, як побачить персонажа.

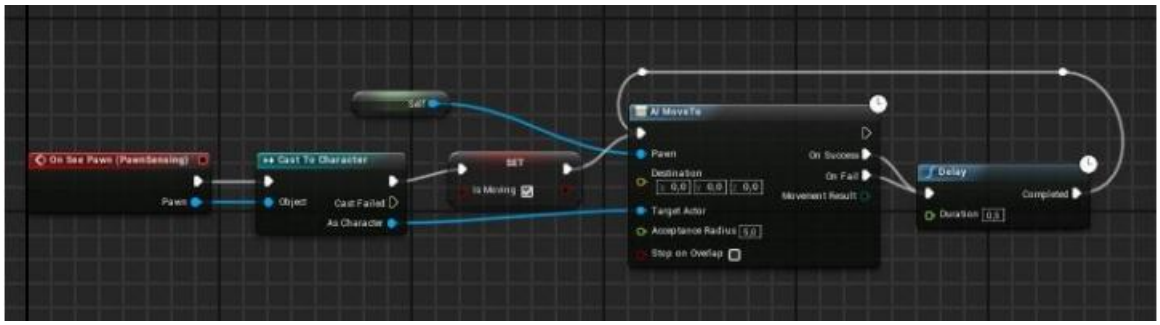


Рис. 3.8. Дія противника коли він бачить нашого персонажа

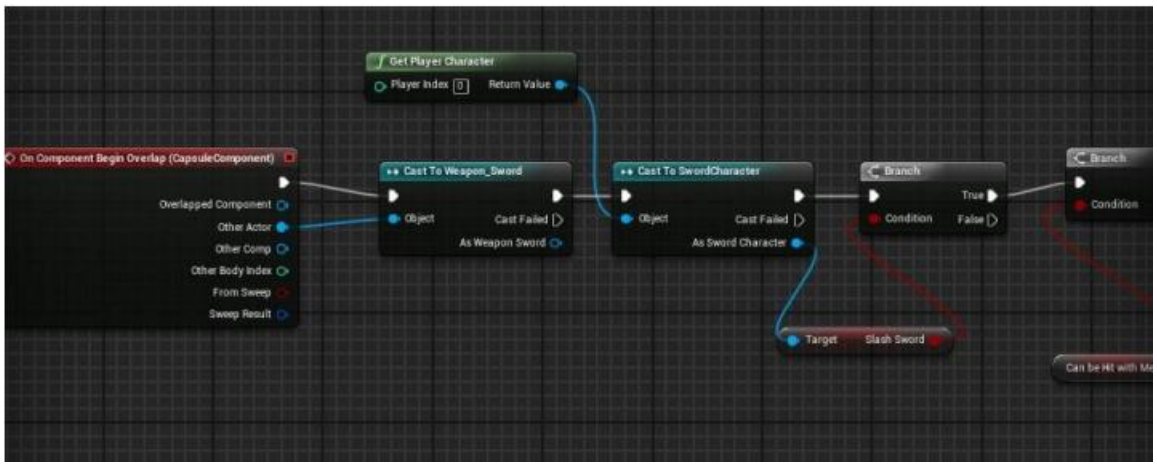


Рис. 3.9. логіка нанесення шкоди персонажу

Якщо противник може отримати шкоду, то від встановленого значення здоров'я (Health) віднімається значення 15 за кожен вистріл,якщо ми попадаєм в голову противнику то можем зняти 36 ХП. Далі йде порівняння поточного значення здоров'я з нулем. Якщо воно не нульове, то противник може бути знову поранений. Якщо воно

стало рівним нулю, то противник гине, актор противника знищується зі сцени.

РОЗДІЛ 4. Демонстрація гри

При запуску гри гравець з'являється у кімнаті де він вибирає в який портал попасти, кожен портал веде в якийсь світ(див. мал. 18).Після чого ми попадаєм в кімнату з противниками які відразу ж хочуть тебе вбити (див. мал. 19). Все, що вони можуть - це бігати за гравцем (див. мал.21) і наносити йому урон, якщо зможуть наздогнати. Гравець, в свою чергу, може бігати та вбивати ворогів стріляючи по ним з різної зброї та оберати її(див. мал. 20). Коли гравець бігає від ворогів, а вони в нього не вистрелили, то хітпоїнти гравця повільно відновлюються. Задача гравця – вижити, вбивши всіх ворогів, витративши не всю шкалу свого життя. Якщо гравець гине то він респавниться(див. мал. 22)

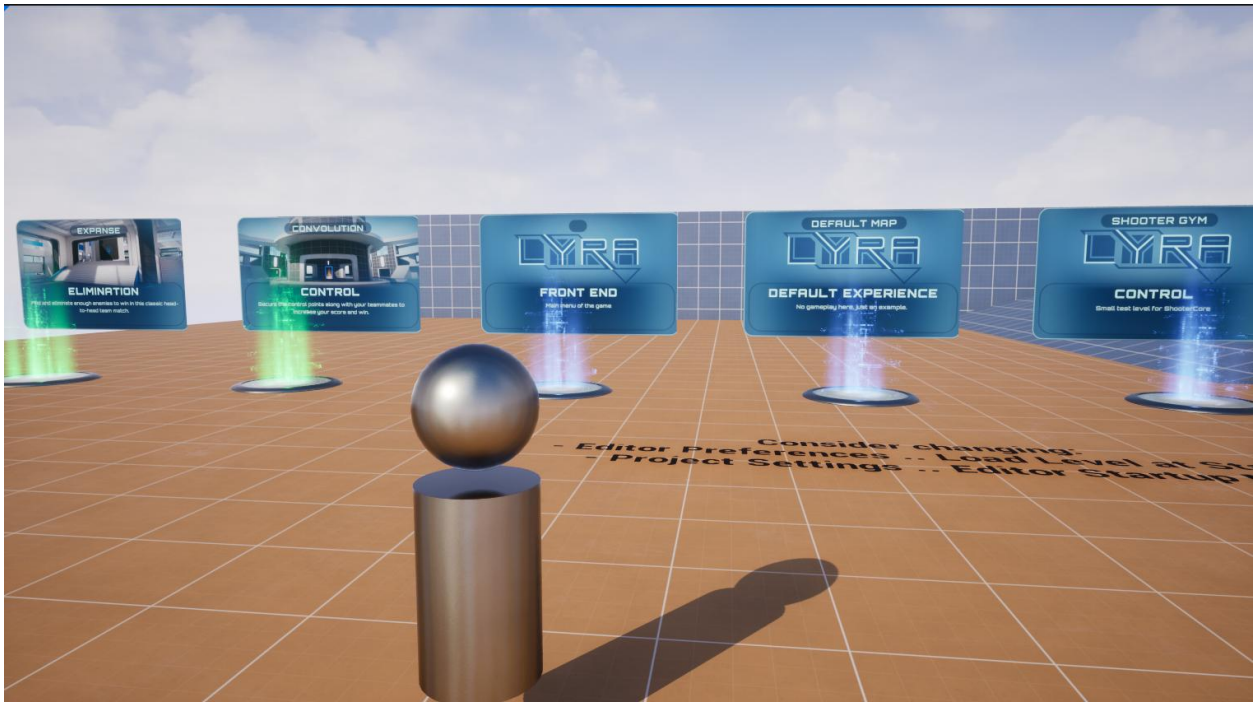


Рис. 4.1. Початкова точка спавна

Рис. 4.2. Кімната

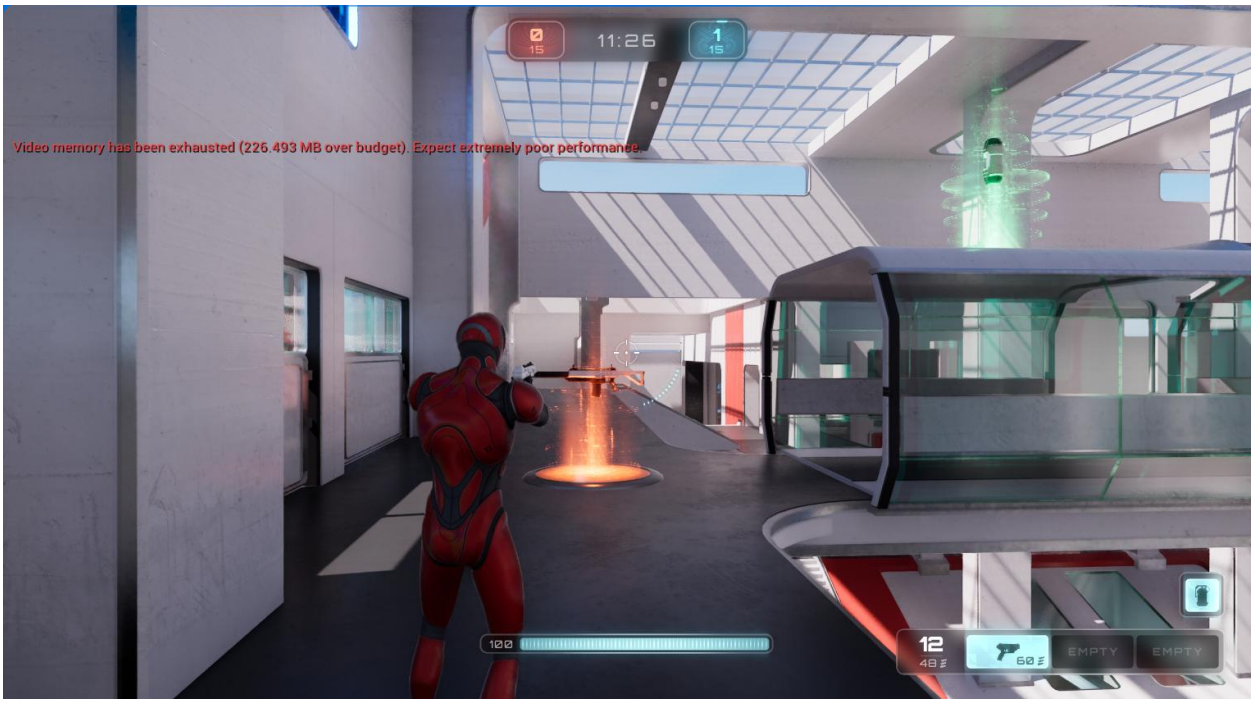


Рис. 4.3. Вибір зброї

Рис. 4.4. Бій з противником

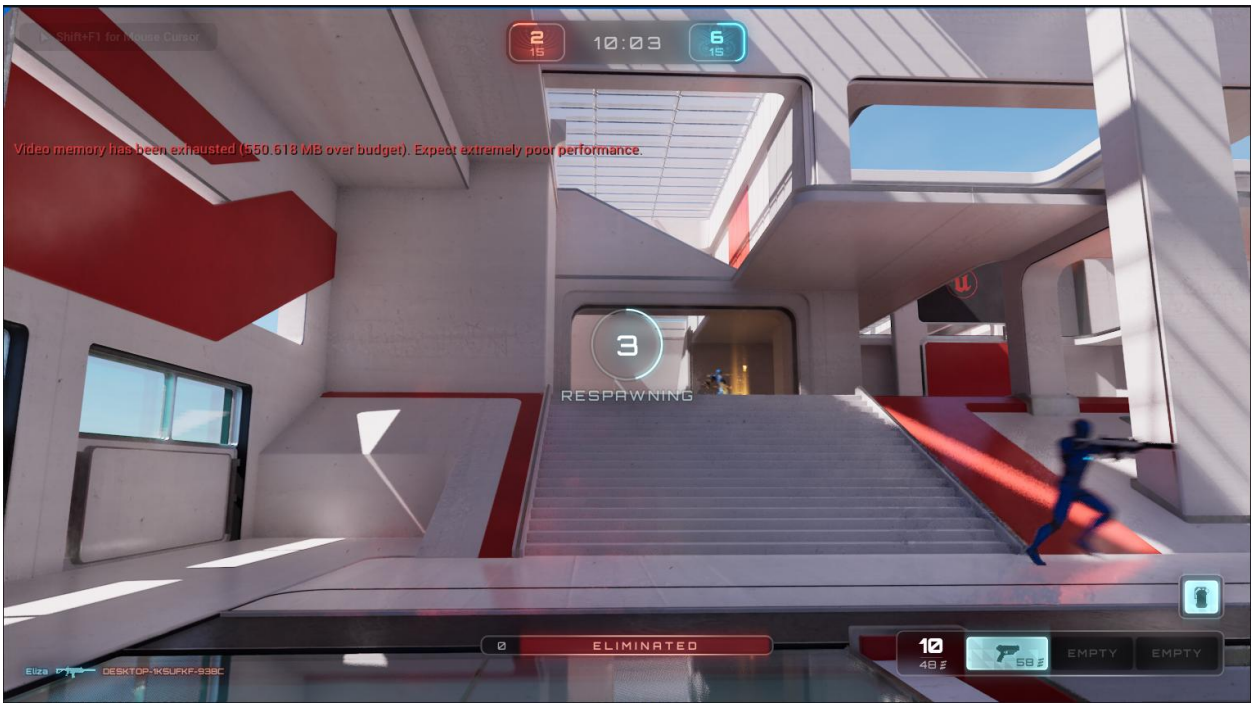


Рис. 4.5. Респавн після смерті

ВИСНОВКИ

У даній випускній кваліфікаційній роботі було проведено створення програмного модуля для Unreal Engine 5, симулятора бою.

У модулі реалізовано стрілянина, нанесення шкоди противнику, , відновлення здоров'я персонажа, елементи, що відновлюють здоров'я, найпростіший графічний користувальницький інтерфейс, найпростіший штучний інтелект (противник).

Створення модуля дасть змогу розробникам полегшити створення нових програмних продуктів, тому що відпаде необхідність їх створення з нуля.

Розробники використовують раніше реалізовані модулі, щоб отримати новий проект. Вносячи зміни і доповнюючи модуль, можна отримати новий продукт, таким чином, дизайнери рівнів можуть створювати свої елементи гри для окремого рівня або всієї програми загалом. Модуль, що реалізує механізм бою, є універсальним. Він підійде і для реалізації іншої холодної зброї ближнього бою. Для цього потрібно зробити невеликі зміни, відрегулювати значення змінних для нанесення потрібної шкоди.У підсумку досягнуто мети роботи, а саме реалізовано програмний модуль для Unreal Engine 5,що симулює стрілянину між персонажем та противником.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний сайт Unreal Ungine 5 https://www.unrealengine.com/en-US/unreal-engine-5
2. Інформація про розробки ігри на Unreal Ungine 5 https://avada-media.ua/ua/services/razrabotka-igr-na-unreal-engine-5/
3. https://www.epravda.com.ua/publications/2023/05/18/700238/
4. Кіберспорт. https://topnews.ck.ua/other/2021/02/02/106739.html
5. **https://ideadigital.agency/blog/osnovni-metodi-segmentatsiyi-auditoriyi/**
6. Apperley T. H. Genre and game studies: Toward a critical approach to video game genres / T. H. Apperley // Simulation & Gaming. – Vol. 37. – No. 1. – P. 6 – 23.
7. Buckland Mat. Programming Game AI by Example – Texas, Wordware Publishing. – s. 25 – 43.
8. Хокінг Д. М. Unity в дії. Мультиплатформенна розробка на практиці. / Д. М. Хокінг. – 336 с.
9. Крейтон, Р.Х. Основи розробки ігор у Unity / Р.Х. Крейтон. – Packt Publishing, – 83 с.
10. Clearwater D. What Defines Videogame Genre? Thinking about Genre Study after the Great Divide / D. Clearwater // The Journal of the Canadian Game Studies Association. – No. 5. – s. 29 – 49.
11. Unity Asset Store [Електронний ресурс] / режим доступу: https://assetstore.unity.com/
12. McShaffry, Mike, Graham David. Game coding complete: Fourth Edition. – Boston, Cource Technolog. – 184c