ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ГРИ "GROPIUS" ПІД ПЛАТФОРМУ ANDROID У ЖАНРІ "ВИЖИВАННЯ 2.5D" З ВИКОРИСТАННЯМ ІГРОВОГО РУШІЯ UNITY МОВОЮ C#»**

Виконав: студент 4 курсу, групи ПД – 42

спеціальності:

121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Вдовін Д.Е

(прізвище та ініціали)

Керівник Дібрівний О.А.

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність - 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач Кафедри
Інженерії програмного забезпечення
Негоденко О.В.
“ ” 2023 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

ВДОВІНА ДМИТРА ЕДУАРДОВИЧА

(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка гри “Gropius” під платформу Android у жанрі “Виживання 2.5D” з використанням ігрового рушія Unity мовою C#»

Керівник роботи Дібрівний О.А. доктор філософії
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «24» лютого 2023 року № 26.

2. Строк Подання студентом роботи « 1 » червня 2023 року

3. Вхідні дані до роботи:

3.1 Науково-технічна література, пов’язана з розробкою Unity мобільних ігор.

3.2 Документація Unity.

3.3. Офіційна документація Rider.

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).

4.1 Аналіз розроблюваної гри.

4.2 Дослідження технологій та методів, для створення гри.

4.3 Розробка мобільної гри.

4.4 Висновки.

5.Перелік демонстраційного матеріалу

5.1 Титульний слайд.

5.2 Мета, об’єкт, предмет.

5.3 Задачі дипломної роботи.

5.4 Вимоги до програмного забезпечення

5.5 Аналоги.

5.6 Концепт гри.

5.7 Програмні та технічні засоби реалізації.

5.8 Діаграма діяльності.

5.9 Діаграма класів.

5.10 Екранні форми.

5.11 Відео гри.

5.12 Апробація результатів дослідження.

5.13 Висновки.

5.14 Кінцевий слайд.

6. Дата видачі Завдання « 25 » лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми бакалаврської роботи	25.02.23-27.02.23	Виконано
2	Підбір джерел інформації	28.02.2023	Виконано
3	Вимоги до продукту	10.03.2023	Виконано
4	Концепція та реалізація	20.03.2023	Виконано
5	Вступ, висновки, реферат	13.04.2023	Виконано
6	Розробка презентації	06.05.2023	Виконано
7	Попередній захист	15.05.2023	
8	Здача роботи	01.06.2023	

Студент ____ Вдовін Д.Е.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дібрівний О.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 61 с., - табл., 17 рис., 15 джерел.

Об’єкт дослідження – ігровий процес у грі в жанрі виживання.

Предмет дослідження – мобільна відеогра в жанрі “Виживання” 2.5D для Android за допомогою рушія Unity.

Мета роботи – покращення ігрового процесу в жанрі “Виживання 2.5D” з використанням ігрового рушія Unity мовою C#.

Наукова новизна – дослідження виживання в мобільній грі. Для досягнення цієї мети, було створено гру, в якій гравець може по різному досягти своєї мети.

В основі дипломного проекту проведено аналіз ринку мобільних телефонів. Було проаналізовано та визначено зміни та обмеження інших програмних засобів. Проаналізовано найпоширеніші аспекти розробки жанру відеоігор виживання.

Мобільну гру було розроблено за допомогою ігрового двигуна Unity. Концепція гри була написана в Rider мовою C#. В якості мобільної платформи була обрана операційна система Android. Для створення ігрових моделей використовувалися Blender і різні інші продукти.

Мобільна гра була розроблена для того, щоб відволіктися від реальності, зануритись в свій відеоігор, та відчути на собі емоції створеного світу для гравця.

ЗМІСТ

ВСТУП

Актуальність теми. Витрати на мобільні ігри перевищать 100 мільярдів доларів, що на 5% менше, ніж у попередні роки. Проте ринок мобільних ігор продовжує розвиватися і залучати все більше і більше гравців. Особливою популярністю користуються мобільні ігри, які доступні широкій аудиторії. Також з кожним роком ігри стають більш популярними серед різного населення, багато людей які грають в ігри хочуть прийти додому, та відпочити пограти в щось цікаве, та захоплююче.

Однак з кожним роком користувачі стають все прискіпливими до мобільних ігор. Тому розробка мобільних ігор стає все складнішою і більш затребуваною.

Дуже давно з'явився цікавий жанр "виживання", в якому гравець має по різному досягати своїх цілей, та намагатись прожити якомога довше. Цей жанр приносить дуже багато грошей, та позитивних відгуків користувачів, саме тому він є актуальним на даний момент.

Об'єктом дослідження ігровий процес у грі в жанрі виживання.

Предметом дослідження мобільна відеогра в жанрі "Виживання" 2.5D для Android за допомогою рушія Unity.

Метою роботи покращення ігрового процесу в жанрі "Виживання 2.5D" з використанням ігрового рушія мовою C#.

Методика дослідження. В процесі дипломної роботи для мобільної гри були використані різні інструменти, такі як Unity3d, Rider та мова програмування C#. Крім того, були використані патерни проектування, зокрема SOLID, Singleton

DRY та інші, які дозволили покращити ефективність та якість роботи програми. Також в процесі дипломної роботи використовувались такі програми як Mixamo, MakeHuman. Використання таких інструментів та патернів є важливим етапом у процесі розробки програмного забезпечення, зокрема ігрових додатків, тому їх використання в дипломній роботі є доцільним та допомогло досягти поставлених цілей.

Наукова новизна є дослідження виживання в мобільній грі, зокрема вивчення впливу стратегій гри на результативність гравців. Для досягнення цієї мети, було створено гру, в якій гравець може по різному досягти своєї мети.

Завдання досліждення. Для реалізації поставленої мети потрібно вирішити наступні завдання:

1. Провести аналіз середовища програмування відеоігор.
2. Провести аналіз різних підходів та методів розробки мобільних ігор.
3. Провести аналіз програмних засобів.
4. Розробити архітекутуру гри
5. Розробити мобільну гру.
6. Провести тестування гри.

Практична значущість результатів - розроблений програмний продукт для плафтформи Android дозволила зрозуміти різні підходи, які можуть бути використані в майбутніх проєктах.

РОЗДІЛ 1 АНАЛІЗ ДОСЛІДЖУВАНОЇ ГАЛУЗІ

1.1 Опис загальних рис та особливостей ігрової індустрії

Відеоігри – це розважальне програмне забезпечення, яке зазвичай містить графіку, звук і взаємодію з користувачем. Гравці керують персонажами чи об’єктами у віртуальному світі, діють від їх імені та взаємодіють з іншими гравцями онлайн або офлайн.

Відеоігри є дуже популярною формою сучасних розваг у всьому світі. Ці ігри дозволяють гравцям взаємодіяти з відеоіграми за допомогою пристроїв введення, таких як контролери, клавіатури або джойстики. Це не лише спосіб грати та розважатися, але й інструмент для навчання, змагань та розвитку дрібної моторики та координації рук і очей.

Історія відеоігор починається з 1950-1960-х років, коли були створені перші моделі таких ігор. У 1970-х роках відеоігри були повноцінною індустрією з багатьма видавцями та виробниками. Однак у наступні десятиліття індустрія відеоігор зазнала спаду через випуск численних ігор низької якості. Це змушує розробників і видавців дуже ретельно контролювати процес розробки нових ігор.

У 2000-х роках більшість відеоігор розробляли великі компанії, але з появою Інтернету незалежні розробники почали створювати власні ігри. Це дозволяє людям у всьому світі створювати власні відеоігри та приєднуватися до індустрії.

Наприкінці 2000-х років з’явився новий спосіб використання відеоігор: кіберспорт або eCompetitions. Це стало новим видом змагань і розваг, які приваблюють все більше і більше людей. Сьогодні відеоігри є дуже популярним видом розваги.

1.1.2 Історія розвитку відеоігор

Перші відеоігри з’явилися в 1950-х роках, коли комп’ютерні майстери почали створювати складні відеоігри, щоб продемонструвати потужність комп’ютерів. Завдяки раннім комп’ютерам, таким як ENIAC і UNIVAC, технології значно просунулися вперед, і комп’ютери почали використовувати не лише для обчислень, але й для інших цікавих речей, наприклад створення відеоігор. Ранні відеоігри були розроблені в діапазоні від простих ігор, таких як хрестики-нулики та шахи, до більш складних ігор, таких як парний теніс, створених у 1958 році. Ці ігри використовуються як демонстрації, щоб показати, як можна використовувати комп’ютери для створення веселих ігор.

У 1961-62 роках програмісти Массачусетського технологічного інституту Стівен Рассел і Мартін Гретцер створили першу відеогру для розповсюдження: «Spacewar!». Гра заснована на концепції космічного бою, який дозволяє двом гравцям керувати своїм космічним кораблем і змагатися один з одним. Незважаючи на простоту графіки гри, вона користується популярністю серед користувачів комп'ютерів.

У 1970-х роках почали розробляти мови програмування для створення відеоігор. Програмісти все більше працювали над розробкою власних ігор, і в 1972 році з'явилася перша аркадна гра «Pong», яка стала популярною і поширеною. Також народилася перша домашня консоль: Magnavox Odyssey, випущена в 1972 році.

У 1980-х роках була представлена перша популярна домашня консоль Atari 2600, на якій можна було грати в десятки ігор, зокрема Space Invaders і Pac-Man. Консоль була настільки популярною та вплинула на цілу культуру, що термін «Associations» був придуманий для позначення популярності відеоігор.

1990-ті роки відкрили нову еру відеоігор із випуском першої PlayStation Sony, яка стала однією з найпопулярніших побутових консолей. Відтоді інші компанії, такі як Microsoft і Nintendo, випустили власні консолі, допомагаючи галузі в цілому.

Сьогодні відеоігри є однією з найприбутковіших галузей розваг і мають глибокий вплив на культуру, як і інші індустрії, такі як кіно та музика. Розробники відеоігор постійно створюють нові захоплюючі ігри зі складнішою графікою та більш різноманітними ігровими можливостями, щоб грати в різних жанрах і сценаріях. У результаті відеоігри продовжують привертати увагу мільйонів геймерів у всьому світі.

1.1.3 Сучасний ринок відеоігор

Відеоігри — це графічні програми, призначені для розваги та розвитку інтелекту дітей. Цей ринок постійно змінюється, а люди, в свою чергу, шукають щось нове та цікаве. Ось чому інновації відіграють таку важливу роль в індустрії відеоігор. За останні 50 років було розроблено дев’ять поколінь ігрових консолей, щоб зробити їх ефективнішими та зручнішими.

На початку індустрії кілька компаній виготовляли ігрові приставки, зокрема Atari, Mattel і Fairchild. Нові конкуренти, такі як Sega та Nintendo, з’явилися у 1980-х роках і досі популярні. Nintendo відома розробкою невеликих і компактних портативних ігрових консолей.

У 1990-х роках багато компаній з виробництва відеоігор закрилися або перейшли на інші платформи, щоб залучити нових гравців. Зараз основними виробниками відеоігор є PlayStation від Sony, Xbox від Microsoft, DS і Wii від Nintendo. Ці компанії постійно розробляють нові консолі та нові ігри, щоб задовольнити потреби своїх клієнтів.

1.1.4 Класифікація ігор за жанрами

Розуміння жанрів відеоігор є захоплюючою темою для геймерів, дизайнерів і всіх, хто цікавиться світом відеоігор Він дає вам знати про різні розділи гри, їхні особливості та цілі. Для гравців це може бути корисним, оскільки вони можуть відкривати нові ігри в жанрах, які їм уже подобаються, або інших жанрах, які можуть стати новими фаворитами.

Для дизайнерів жанри відеоігор є важливим елементом у створенні нових ігор, що дозволяє їм досліджувати та вдосконалювати різні ігрові ідеї та концепції. Тому розуміння жанрів відеоігор є корисним і важливим для всіх, хто цікавиться цим захоплюючим ринком, що швидко розвивається.

Є такі жанри відеоігор:

1. Екшн-ігри
2. Пригодницькі ігри
3. Ігри-пригоди
4. Рольові ігри
5. Симуляційні ігри
6. Ритмічні ігри
7. Стратегічні ігри
8. Військові ігри
9. Ігри-головоломки
10. Спортивні ігри
11. Бездіяльні ігри

Екшн-ігри дуже популярний серед геймерів у всьому світі та включає різноманітні фізичні випробування, які гравці повинні подолати, зокрема стрибки, бійки та стрільбу. Ці ігри можуть бути іграми від першої особи, де гравець дивиться на гру очима свого суперника, або іграми від третьої особи, де гравець спостерігає за своїм персонажем збоку або повністю. Приклад такої гри (рис. 1.1).

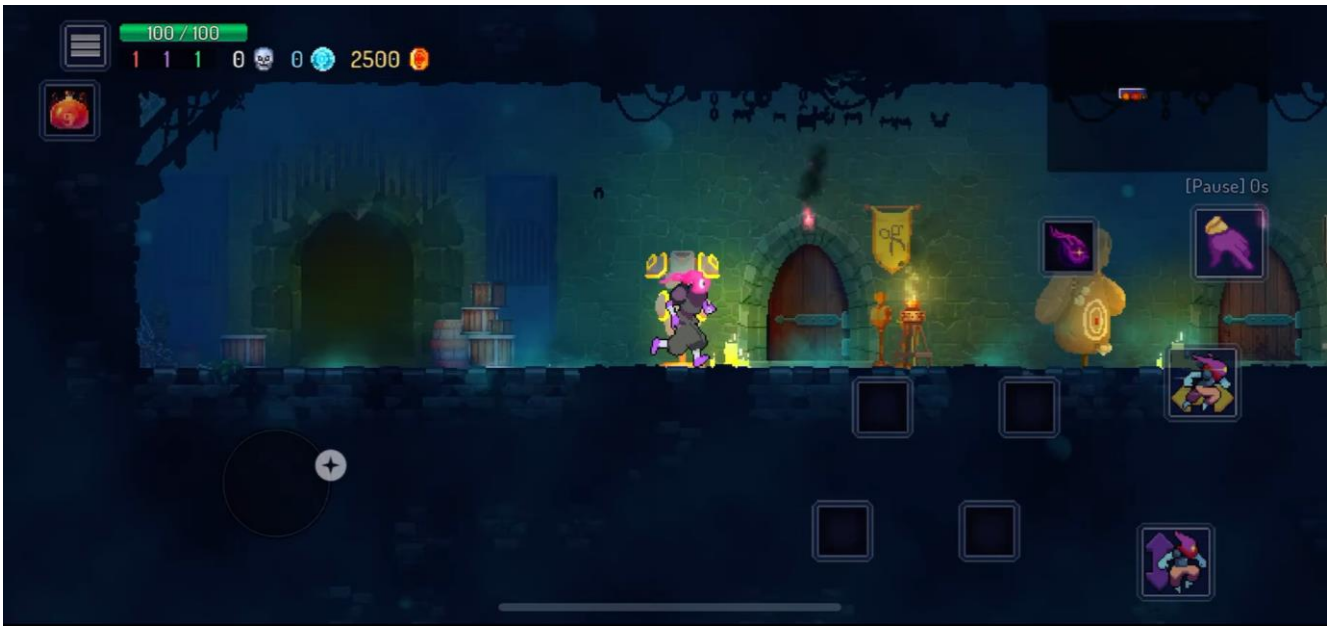


Рисунок 1.1 — Демонстрація геймплею в Dead Cells

Пригодницька гра — це відеогра, яка поєднує в собі елементи бойовиків і пригодницьких ігор. Ці ігри можуть містити елементи дії, такі як стрілянина чи бійки, але вони також мають складні сюжетні лінії та персонажів, з якими можна взаємодіяти. . Приклад такої гри на (рис. 1.2).

Рисунок 1.2 — Демонстрація геймплею в Grand Theft Auto: San Andreas

Ігри-пригоди - це ігри, які занурюють гравців у захоплюючі світи, повні таємниць, загадок і небезпек. Ці ігри, як правило, є іграми для одного гравця, і історія та обстановка відіграють важливу роль у створенні незабутніх вражень для гравця. Ігри можуть включати такі речі, як головоломки, розшифровка повідомлень, пошук і використання інструментів, відкриття замкнених дверей і дослідження нових місць. Загалом пригодницькі ігри дозволяють гравцеві відчути себе персонажем у віртуальному світі та досліджувати його самостійно. Приклад такої гри на (рис. 1.3).

Рисунок 1.3 — Демонстрація геймплею в Machinarium

Рольові ігри - гравці беруть на себе роль унікального героя і керують ним, покращуючи його характеристики та здібності. Основна мета гри - уникати складних місій і перемогти монстрів, які можуть дати гравцеві очки досвіду. Ці бали використовуються для покращення персонажа та отримання нових навичок, які допомагають гравцеві долати складніші випробування. Такі ігри дозволяють гравцям стати частиною фантастичного світу і зануритися в нього на короткий проміжок часу, створюючи унікальний ігровий досвід. Приклад такої гри на (рис. 1.4).

Рисунок 1.4 — Демонстрація геймплею в Titan Quest: Legendary Edition

Симуляційна відеогра — це відеогра, яка намагається якомога точніше імітувати реальні події. Гравці можуть грати в різні симуляції з різними цілями: вивчати нові навички, розважатися, аналізувати різні сценарії та навіть передбачати можливі результати в певному сценарії. Симулятори можуть бути самими різними: від симуляторів польотів і гонок до симуляторів життя і бізнесу, де гравцям доводиться виконувати різноманітні завдання і приймати рішення, що впливають на розвиток сюжету. Приклад такої гри на (рис. 1.5).

Рисунок 1.5 — Демонстрація геймплею Fallout Shelter

Ритм-гра — це відеогра, у якій гравець має підтримувати ритм і грати під музику. Жанр часто містить елементи бойовиків, такі як швидкість і рефлекси, але

основний акцент робиться на імітації ритму музики. Для гри гравці можуть використовувати спеціальні контролери, такі як контролери для танцполу або гітари. У ці ігри можна грати як поодинці, так і в командах, де гравці змагаються, щоб посидіти й набрати якомога більше очок. Приклад такої гри на (рис. 1.6).

Рисунок 1.6 — Демонстрація геймплею Beatstar

Стратегічна гра – це тип відеоігор, у якому розумова активність і планування важливіші за реальні дії гравця. У таких іграх гравцеві доводиться продумувати кожен рівень і вирішувати головоломки, щоб перемогти в одиночній або

багатокористувацькій грі. Стратегічні ігри можуть передбачати управління економікою, будівництво військових баз, управління арміями та планування нападів на ворогів. Гра вимагає від гравця проявити своє стратегічне мислення та навички швидкого прийняття рішень для перемоги. Приклад такої гри на (рис. 1.7).

Рисунок 1.7 — Демонстрація геймплею Rome: Total War

Військова гра — це відеогра, у якій гравці беруть участь у карткових битвах, а також у тактичних чи стратегічних боях. Геймплей є або покроковим, або в реальному часі, а військові ігри різною мірою зосереджені на військовій стратегії чи тактиці. Це можуть бути бойові схеми, тактичне планування та командування військами, використання різних типів зброї та обладнання тощо. Ці ігри можуть бути одиночними або багатокористувацькими, де гравці змагаються один з одним або працюють у групах для досягнення спільної мети. Приклад такої гри на (рис. 1.8).

Рисунок 1.8 — Демонстрація геймплею World War Heroes: WW2 FPS

Спортивна відеогра — це відеогра, яка імітує різні види спорту. Гравці можуть взяти на себе роль гравця, брати участь у віртуальних вправах, розвивати свої навички та навички, долати різні виклики та отримувати нагороди. Ці ліги можуть представляти широкий спектр видів спорту, включаючи командні види спорту, такі як футбол, баскетбол і хокей, а також індивідуальні види спорту, такі як легка атлетика, бойові мистецтва та екстремальні види спорту. Спортивні відеоігри приносять задоволення і допомагають підвищити фізичну активність і розвинути рухову координацію. Приклад такої гри на (рис. 1.9).

Рисунок 1.9 — Демонстрація геймплею Rocket League Sideswipe

Гра-головоломка — це відеогра, яка вимагає від гравця виконання розумових завдань для швидкого та ефективного вирішення проблем. У таких іграх потрібно знаходити рішення складних завдань, знаходити логічні зв’язки та дотримуватися певних правил. Різні типи головоломок: кросворди, головоломки для пошуку слів, головоломки з числами, головоломки про зв’язки та логічні головоломки, кожна з яких має свої особливості та вимоги до гравця. Ігри-головоломки допомагають розвивати мислення, концентрацію та інші когнітивні навички. Приклад такої гри на (рис. 1.10).

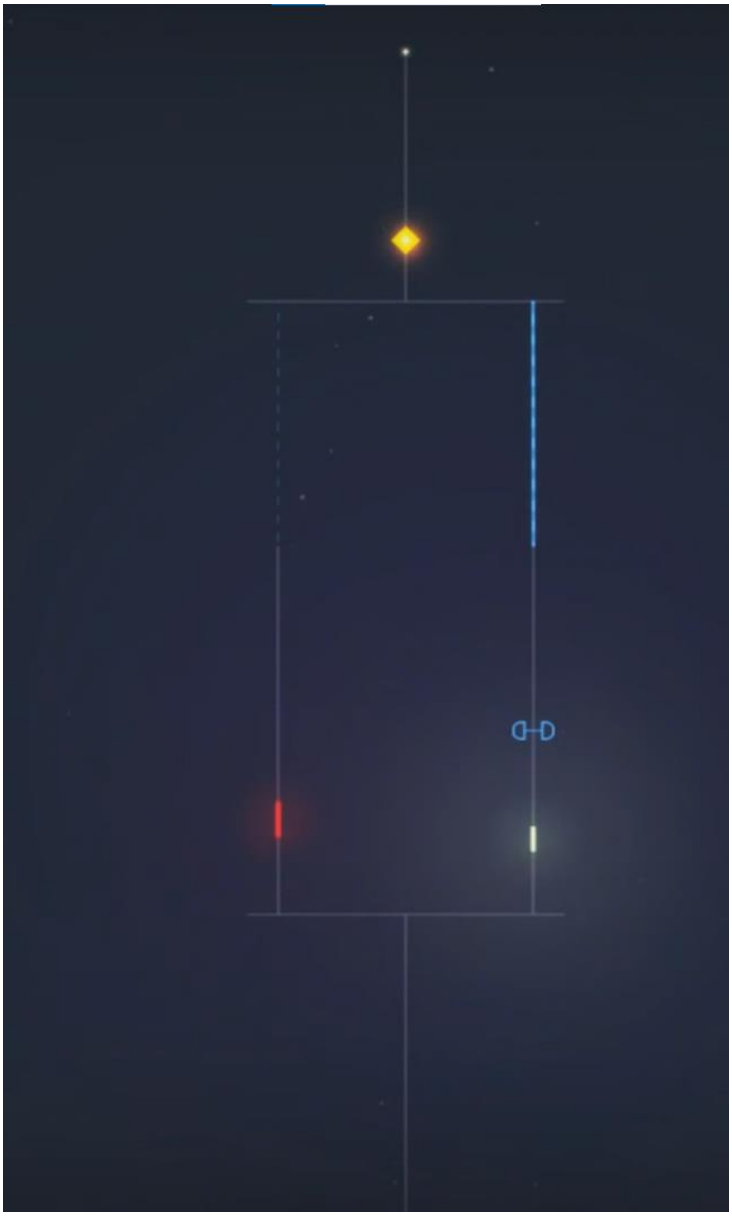


Рисунок 1.10 — Демонстрація геймплею Linelight

1.1.5 Розробка мобільних відеоігор

Для створення мобільної гри є такі етапи:

1. Складіть свій план.
2. Виберіть програмне забезпечення.
3. Вивчіть свою мову програмування.
4. Розпочніть свій проект.
5. Реалізуйте свою графіку.
6. Підніміть свою гру.
7. Випустіть свою гру.

Перш ніж почати створювати власні ігри, вам потрібно точно зрозуміти, який жанр ви хочете створювати та які ігри вам подобаються. Також важливо визначити, 2D чи 3D гра, який художній стиль і які персонажі грають. Ви повинні записати всі свої ідеї та описати, як саме ви бачите свою гру. Зошит, де ви можете записувати все, що ви хочете зробити в грі, допоможе вам упорядкувати свої думки та успішно створювати ігри.

Щоб створювати мобільні ігри, потрібно використовувати правильне програмне забезпечення. Найпопулярнішим з них є Unity, який має високі графічні можливості і може бути безкоштовним до виходу гри. Іншим популярним варіантом є Unreal Engine UDK, у якому є готовий базовий проект, який ви можете випробувати та вивчити інструмент. Якщо ви не впевнені, чи вибрати вам Unity чи UDK, порівняння їхніх функцій допоможе вам вирішити. Існує також програмне забезпечення GameSalad, яке не потребує знання програмування та використовує «систему поведінкової логіки». Хоча це трохи важко освоїти, і ви не зможете створювати складні ігри відразу, це дуже потужний і ви можете легко перенести свої ігри на iOS, Android і Windows.

Якщо ви хочете дізнатися більше про розробку ігор за допомогою GameSalad, ви можете скористатися доступними онлайн-курсами.

Якщо ви хочете бути успішним розробником мобільних ігор, вивчення мови програмування є важливим кроком у вашій освіті. Однією з найпопулярніших мов

розробки ігор є C#, яка широко використовується розробниками Unity. Це об’єктно-орієнтована мова, яка є відносно простою та легкою для розуміння.

Крім того, Python також є дуже популярною мовою для розробки мобільних ігор. Однією з головних переваг Python є великий вибір бібліотек, що значно спрощує процес розробки. Незалежно від того, яку мову програмування ви виберете, важливо зосередитися на тому, щоб навчитися розробляти складні та захоплюючі мобільні ігри.

Розпочати власний проект з розробки мобільних ігор може здатися складним, але якщо ви виконаєте кілька простих кроків, ви швидко побачите результати. Основна ідея полягає в тому, щоб почати з розробки базової ігрової механіки та правил, щоб визначити основні елементи гри. Наступні кроки — створити перший рівень гри, розробити графіку та запустити тести, щоб переконатися, що гра працює належним чином. Останнім кроком є розробка інтерфейсу користувача, який містить екрани, меню та параметри гри, які допомагають користувачеві взаємодіяти з грою. Важливо зауважити, що під час розробки гри ви повинні регулярно тестувати, щоб виявляти та виправляти помилки та покращувати функціональність гри.

Перш ніж почати працювати над графікою мобільної гри, вам потрібно вирішити, чи є у грі 2D чи 3D графіка. Ви можете використовувати різні програми для графічного дизайну, наприклад Photoshop для 2D-графіки або Maya і 3ds Max для 3D-графіки.

Якщо у вас немає необхідних графічних навичок, ви можете найняти дизайнера, який допоможе створити візуальну частину гри. Є багато сайтів, де можна знайти таких експертів.

Розробляючи графіку, важливо пам’ятати, що вона повинна відповідати темі гри та відображати її історію. Перед початком розробки вам слід ретельно продумати дизайн і підтримувати графіку у відповідності до потреб гри.

Після того, як ви розробили свою гру, ви повинні зосередитися на її вдосконаленні, щоб надати користувачам привабливий і незабутній досвід. Це

вимагало кількох важливих кроків, таких як видалення пікселів, створення простої та інтуїтивно зрозумілої системи керування та додавання цікавого сюжету та атмосфери разом із тривожною музикою. Ви також повинні провести багато тестів, щоб переконатися, що все працює правильно.

Після того, як патч буде завершено, вам потрібно розблокувати гру. Не зациклюйтеся на фазі випуску, оскільки це може сповільнити вашу гру. Спробувати завжди варто, але не чекайте відповідного моменту. Якщо ваша гра має потенціал, вона знайде свою аудиторію, і ви можете покращити гру, випускаючи оновлення.

Якщо ви плануєте монетизувати свою гру, не забудьте про маркетинговий бюджет. Ви також можете подати свою гру на перевірку на кількох платформах. Коли ваша гра буде готова до випуску, ви можете відправити її на такі платформи, як Apple або Google Play. Вони містять важливу інформацію про подання претензії.

1.1.6 Статистика мобільних відеоігор

Через попередній успіх мобільних ігор, які були популярні під час періоду ізоляції, очікується, що ринок скоротиться в 2022 році. Однак експерти прогнозують, що ринок знову зросте в найближчі роки, досягнувши тут 211 мільярдів доларів у 2025 році.

За даними Newzoo, ринок мобільних ігор досягне 184 мільярдів доларів у 2022 році, що на 4,3% менше, ніж у 2021 році. Очікується, що ринок Північної Америки також зросте на -5,1% до 48,4 мільярдів доларів до 2022 року. Європа - 3,5% до 32,9 мільярдів доларів . та Азії. Тихоокеанський регіон -5,6% до $87,9 млрд.

Незважаючи на ці недоліки, мобільні ігри залишаються найприбутковішим сектором індустрії та продовжують розвиватися. Тому важливо знати статистику ринку мобільних ігор, щоб досягти успіху в цій галузі.

До 2022 року дохід від мобільних ігор досягне 92 мільярдів доларів, що становитиме 50% від загального доходу від азартних ігор. А продажі впали на -

6,4% порівняно з роком. З більш ніж 90 мільярдами завантажень мобільних ігор. Очікується, що до 2027 року ринок мобільних ігор досягне 419 мільярдів доларів. До 2022 року середній споживач мобільних ігор становитиме 164 долари.

Користувачі в Індонезії, Бразилії, Саудівській Аравії, Сінгапурі та Південній Кореї витрачають більше 5 годин на день на десять додатків. . До 2022 року Китай, Індія та США стануть найбільшими у світі виробниками фільмів та мобільних ігор. Буде більше мобільних кінотеатрів, а також Китай, США. А Японія стане найбільшим ринком мобільного зв'язку за споживчими витратами. До 2022 року буде 1419 програм та ігор. 700 000 ігор будуть доступні в App Store, близько 500 000 в Google Play до 2023 року і 200 000 ігор в App Store.

Цікава історія про заробіток в мобільних іграх! За останні роки кількість користувачів, які грають у мобільні ігри, різко зросла. У цьому сенсі тому, що в мобільні ігри можна грати будь-де і будь-коли.

Що стосується монетизації мобільних ігор, покупки в програмі замінили на перегляд реклами, та покупки в магазині. Більшість гравців, які відповіли на це питання, віддають перевагу безкоштовним іграм з рекламою, аніж іграм які платні.

Цікаво, що люди, залежні від мобільних ігор, на 21% частіше впливають на рішення споживача про покупку, ніж люди, які не залежні. Більшість мобільних ігор доступні через покупки в програмі, і більше 30% жінок платять за мобільні ігри.

Ці цифри показують, що мобільні ігри є прибутковим бізнесом і його популярність зростає з кожним роком. Оскільки мобільні ігри можуть залучити більше користувачів, ніж традиційні комп’ютерні ігри, вони стають більш популярними, і видавці бачать у них можливість просувати свої продукти та послуги.

1.2 Жанр виживання

1.2.1 Загальна інформація про жанр

Ігри на виживання — це один із піджанрів відеоігор, де гравець потрапляє у ворожий і небезпечний світ з обмеженими ресурсами та обладнанням. Основна мета гри - крафтити більше предметів, збирати ресурси знаходячи воду, та їжу, щоб вижити. Шукати та знаходити воду, та інші матеріали, зібрані та знайдені для захисту від ворогів. Ігри — це відкриті світи, де гравці можуть досліджувати різні території та взаємодіяти з іншими гравцями чи побічними персонажам. Ігри на виживання можуть бути дуже напруженими, тому що гравці постійно повинні переживати голод, спрагу, хвороби та небезпеку, що створює захоплюючий і динамічний геймплей. Цей жанр здобув велику популярність в останні роки, особливо серед любителів викликів та пригод.

Ігри про виживання в цілому це піджанр відеоігор, які часто плутають із популярним піджанром ігор про виживання-екшену. Але цей термін використовувався для опису різноманітних відеоігор з 1990-х років. Основна відмінність полягає в тому, що в іграх на виживання гравці опиняються у ворожому відкритому світі з невеликим обладнанням і повинні збирати ресурси та обладнання вручну, щоб вижити якомога довше. Головна мета - зібрати якомога більше корисних предметів і залишитися в грі.

1.2.2 Появлення та як розвивався жанр виживання

Цей жанр був випущений у 1992 році з випуском Unreal World, який зосереджувався на елементах ігрового процесу, а не на вмісті. Пізніше, коли з’явилися ігри про виживання з більшим вмістом і налаштуваннями, жанр став дуже популярним у Minecraft, дозволяючи гравцям будувати та збирати ресурси для створення світу. З часом жанр ігор про виживання розвивався та об’єднувався з іншими жанрами, створюючи різні піджанри та жанри, які досі подобаються геймерам у всьому світі.

1.3 Аналіз існуючих відеігор

Minecraft — це відеогра, яка не має сюжету чи глибокого сенсу, але завдяки своїй творчій свободі та складній системі вона може подарувати гравцям незабутні враження. Основна увага в грі полягає в дослідженні світу та створенні своєї імперії, але є й інші режими, такі як Творчий, Пригодницький і Виживання. Майнкрафт — це шедевр, який розроблявся роками, і його рекламують як одну з найкращих відеоігор для створення та гри.

Механіка Minecraft дуже цікава і багатогранна. Гравці можуть створювати і руйнувати все, що завгодно на світі, поєднуючи різні матеріали з природи. Гравці можуть пересуватися по світу, але будьте обережні, тому що вода або лава можуть вплинути на здоров'я персонажа. Такі дії, як напад, біг, використання інструментів і їжі, крадіжка, захоплення, удари та взаємодія з іншими гравцями є важливою частиною гри. Ключове слово гри - свобода, яка дає кожному гравцеві можливість змінювати світ і щось створювати.

Ark: Survival — це захоплююча гра, де гравці виживають на зеленому тропічному острові, населеному стародавніми динозаврами. У грі є елементи виживання в пісочниці, рольової гри та шутера від третьої особи. Щоб вижити в цій небезпечній пустелі, гравці повинні збирати ресурси, будувати будинки, збирати інструменти, насіння, інструменти, їсти та пити.

У грі є можливість приборкати динозаврів і використовувати їх як транспортні засоби та ресурси. Гравці також можуть полювати на диких динозаврів, щоб отримати їхні ресурси та м’ясо. Гра містить велику різноманітність динозаврів та інших тварин, кожен з яких має свої особливості та унікальні характеристики.

Гравці можуть заробляти очки досвіду та проходити в режимі RPG, щоб отримати нові здібності та отримати доступ до потужної зброї та броні. У грі також є можливість грати як гравець і приєднатися до племені, щоб поділитися ресурсами та працювати разом, щоб вижити в небезпечному племені.

Ark: Survival Evolved — це величезна, детальна та захоплююча гра, яка дозволяє гравцям досліджувати та виживати у світі динозаврів. Гра поєднує в собі різноманітні ігрові елементи, які дають гравцям повну свободу та незабутні враження.

Don't Starve Together — це гра на виживання, дія якої відбувається в темному фентезійному світі, де гравці розміщені на випадково згенерованих картах із постійною небезпекою. Щоб вижити, гравці повинні разом збирати ресурси з різних біомів, збирати їжу, готувати їжу, виготовляти зброю для боротьби з монстрами, будувати та ремонтувати будівлі та розпалювати багаття вночі.

У грі є 3 режими гри, і ви можете грати з друзями або незнайомцями. Основна ідея полягає в тому, що кожен гравець повинен вижити, тому важливо, щоб вони ділилися ресурсами та підтримували один одного.

Тому що смерть кожного гравця може призвести до того, що вони стануть привидами, які можуть зруйнувати здоров’я інших гравців. Нові карти роблять кожну гру унікальною, а музика та звукові ефекти надають грі унікальну атмосферу.

1.4 Висновки до розділу 1

У рамках першого розділу даної дипломної роботи ми проаналізували, чому мобільні ігри користуються великим попитом, зробили аналіз різних жанрів відеоігор, провели детальний аналіз ринку мобільних ігор.

Аналіз існуючих відеоігор допомагає зрозуміти, що може запропонувати ринок, які типи ігор є найпопулярнішими та що можна покращити в існуючих іграх. Цей тип аналізу може бути корисним розробникам, які хочуть створювати успішні ігри, а також геймерам, які хочуть вибрати найкращу гру для себе.

Жанр виживання останнім часом дуже популярний в ігровій індустрії. Це пов’язано з великою кількістю геймерів, які хочуть відчути хвилювання життя під постійною загрозою. Жанр стрімко розвивається, з’являються нові ідеї та можливості, які дають гравцям краще уявлення про світ, у якому кожен крок може бути останнім.

Опис загальних характеристик і характеристик індустрії відеоігор включає, серед іншого, конкуренцію, ризик, технології та розвиток. Ігрова індустрія дуже конкурентна, оскільки кількість розробників та ігор зростає з кожним днем. Ризик пов'язаний з багатьма невдалими проектами, що призводять до фінансових і репутаційних втрат.

Технології допомагають розробникам створювати більш складні та реалістичні ігри. Розвиток індустрії сприяє збільшенню кількості гравців, які бажають використовувати внутрішньоігрові гроші, що збільшує прибутки компанії та стимулює розвиток індустрії.

Важливим аспектом успіху будь-якої гри є задоволення потреб гравців, інновації та розвиток геймплею. З цією метою, розробники мобільних ігор повинні зосередитись на створенні нових ігрових механік та удосконаленні існуючих.

Ще одним важливим аспектом є монетизація гри. Розробники повинні знайти баланс між задоволенням гравців та заробітком на грі. Для цього можна використовувати різні стратегії, такі як продаж інтерактивних елементів гри,

підписки на додатковий контент, рекламу та інші. Однак важливо пам'ятати, що надмірна монетизація може викликати негативну реакцію гравців та зменшення популярності гри.

У цілому, розробка та випуск мобільних ігор є складним та ризикованим процесом, який потребує великих зусиль та інвестицій. Проте, успішність гри може призвести до значного прибутку та популярності. Вірна стратегія розвитку, зосередження на потребах гравців та забезпечення якісної геймплею є ключем до успіху у цій галузі.

Одним з ключових аспектів розвитку відеоігор є постійне зростання вимог гравців до якості графіки та звуку, а також до реалістичності та глибини геймплею. Це ставить перед розробниками складні завдання з розробки нових технологій та інноваційних рішень, щоб задовольнити потреби своїх користувачів та зберегти лідерство на ринку відеоігор.

На основі вишеописаної інформації було вирішено вибрати жанр виживання, який на ринку зараз потребує попит у користувачів, адже цей жанр включає в себе різні аспекти ігрових емоцій. Та дозвовляє поринути в цікавий світ відеоігор який створив розробник. Адже в таких іграх можна провести багато часу, та ненадовго забути реальність.

РОЗДІЛ 2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ

2.1 Загальна інформація про ігрові двигуни

Ігровий двигун - це програмне забезпечення для розробки відеоігор. Це також можна назвати архітектурною, системною або інженерною грою. До основних функцій ігрових лаунчерів часто відносять створення 2D або 3D зображень, рух фізичних об'єктів, рух персонажів, розвиток штучного інтелекту, відтворення звуку і можливість передачі зображень. Ігровий движок є важливою частиною розробки гри, оскільки він дозволяє створювати візуальне середовище гри та відтворювати різні об’єкти у грі.

Платформи призначены для розробки відеоігор різних жанрів з різними мовами програмування. Вони мають інтегроване середовище розробки із функціями та налаштуваннями, які дозволяють ефективно розробляти ігри без простоїв.

Однією з головних особливостей програми запуску ігор є метод візуалізації, який створює ігрову графіку у 2D та 3D формі. Для ефективної роботи графічний сервіс повинен адаптуватися до різних типів інсталяцій і мати можливість поєднувати косметичні аспекти гри в єдине зображення.

Програмне забезпечення для створення ігор має багато функцій, зокрема штучний інтелект, фізичний движок, аудіо движок і мережу.

Штучний інтелект дає автоматичні рекомендації на основі поведінки гравців у грі та впливає на продуктивність і стратегію гри.

Фізичний механізм відповідає за проектування та розробку моделювання реальних дій, рухів і поведінки в грі. Він пропонує реалістичність відеоігор із діями та реакціями в реальному часі.

Звуковий механізм контролює звукові ефекти, створювані під час взаємодії в грі. Це аудіофайли, які інтегруються в логіку гри за допомогою програмного забезпечення.

Мережа дозволяє гравцям грати разом через Інтернет і підтримує мережевий механізм для багатокористувацьких або соціальних ігор.

Ці компоненти працюють разом, щоб створити повнофункціональну гру для кожного користувача. Іноді розробникам не потрібні одна або декілька функцій, пов’язаних із грою, яку вони розробляють. Однак компаніям і розробникам рекомендується використовувати всі патчі, щоб отримати найкращий ігровий процес.

2.2 Аналіз ігрових двигунів

2.2.1 Unreal Engine

Unreal Engine — це програмне забезпечення для розробки відеоігор, розроблене Epic Games у 1988 році. Спочатку двигун був розроблений як шутер від першої особи, але сьогодні він використовується для створення різноманітних ігор, таких як бойовики, рольові ігри, стелс та інші. Unreal Engine використовує мову програмування C++, що дозволяє розробникам створювати складні ігри з динамічними та візуальними ефектами. Хоча деякі розробники пропонують інші інструменти, такі як Unity, Unreal Engine дуже популярний серед багатьох розробників ігор, оскільки пропонує найпотужніші та гнучкі ігрові інструменти, що полегшує роботу як для початківців, так і для досвідчених розробників.

Unreal Engine — потужний ігровий движок, який дозволяє користувачам створювати власні ігри. Цей движок має різні інструменти та механізми, які допомагають у створенні ігор. Unreal Engine відомий своїми налаштуваннями та низькою вартістю порівняно з іншими ігровими движками. Через це його обирають багато початківців і професіоналів, які прагнуть створювати власні ігри. Двигун Unreal складається з різних частин, включаючи графічний движок, фізичний движок, звуковий движок і так далі. Користувачі можуть використовувати різноманітні редактори для розробки ігор. Unreal Engine —

потужний інструмент для створення різних типів ігор, від ігор із низькою графікою до ігор з високою графікою.

Ця потужна програма для розробки відеоігор та інших інтерактивних програм. Пропонує високоякісну графіку, зручний інтерфейс і широкий спектр інструментів і опцій розробника.

Головною перевагою Unreal Engine є дивовижна якість графіки, яка може зробити відеоігри максимально реалістичними та веселими. Крім того, Unreal Engine простий у використанні та улюблений розробниками всіх рівнів.

Інтерфейс Unreal Engine постійно оновлюється найновішими інструментами та опціями, що дозволяє розробникам створювати більш складні та цікаві ігри. Крім того, Unreal Engine дозволяє створювати відеоігри та інші програми без необхідності писати складний код і сценарії, використовуючи простий код і вузли.

Unreal Engine використовує мову програмування C++, одну з найпопулярніших серед розробників. Це робить Unreal Engine першим вибором для тих, хто знає C++ і хоче створювати високоякісні відеоігри та інші програми.

Крім того, Unreal Engine має велику спільноту розробників, яка активно співпрацює з іншими розробниками та ділиться з ними знаннями та досвідом. Це дозволяє розробникам розвивати й покращувати свій контроль над Unreal Engine і створювати більш складні та цікаві відеоігри та інші програми.

Особливості:

1. Якість графіки: Unreal Engine має потужні інструменти для створення красивих візуальних зображень у режимі реального часу для створення реалістичних ігрових світів і подій.
2. Кросплатформенність: Unreal Engine підтримує Windows, macOS, Linux, iOS, Android, PlayStation, Xbox та інші платформи. Це означає, що розробники можуть використовувати ті самі інструменти для створення ігор і програм для різних пристроїв.

3. Модульність: Unreal Engine має модульну структуру, яка дозволяє додавати та використовувати різні об’єкти та інструменти, що робить процес розробки простішим та ефективнішим.
4. Універсальність: Unreal Engine підтримує корпус, інтелект, звук, світло та інші системи. Це дозволяє розробникам швидко додавати функціональність до своїх проектів.

2.2.2 Cry Engine

CryEngine — потужний інструмент для створення високоякісних 3D ігор, успішно використовується розробниками по всьому світу. Двигун сумісний із PlayStation 4, Xbox One, Windows, Linux, Oculus Rift, OSVR, PSVR тощо. І може використовуватися для розробки різноманітних ігор і моделей для таких платформ, як HTC Vive. CryEngine відомий своєю системою «що бачиш, те й граєш», яка прискорює процес розробки. Крім того, CryEngine надає розробникам вихідний код, який дозволяє розробку та оптимізацію.

Як і Unreal, CryEngine є безкоштовним двигуном із системою роялті, яка вимагає від розробників платити за використання двигуна, якщо їхній проект перевищує 5000 доларів річного доходу. Розробники також можуть придбати ексклюзивні комерційні ліцензії від Crytech, які надають додаткові функції та підтримку. Багато популярних ігор, таких як Hunt: Showdown, Crisis і The Climb, засновані на CryEngine, а деякі бібліотеки розробників використовують модифіковані версії двигуна для своїх проектів, що робить його унікальним і цікавим для гравців.

Щоб використовувати CryEngine, вам потрібні навички програмування та розробки ігор, але для початку ви можете використовувати вбудовані компоненти та готові приклади. Крім того, існує спільнота розробників, яка надає підтримку та вказівки щодо використання CryEngine.

Однією з головних переваг CryEngine є його потужність і гнучкість, що дозволяє створювати ігри з високоякісною графікою та фізикою, включаючи використання таких функцій, як штучний інтелект, анімація та фізична взаємодія. Загалом, CryEngine — чудовий вибір для розробників ігор та тих, хто хоче більше контролювати свою продуктивність і розробку.

Особливості:

1. Графіка: CryEngine відомий своєю потужною графікою та реалістичними результатами. Він використовує передові методи, такі як трасування променів, для створення дивовижних ефектів, таких як відблиски, тіні та світло.
2. Фізика: CryEngine має потужний фізичний механізм, який створює реалістичне середовище та ефекти. Він підтримує динамічне руйнування, рух об’єктів, моделювання віртуальної реальності тощо.
3. AI: CryEngine має набір інструментів штучного інтелекту, які допомагають розробникам створювати складні та інноваційні поведінкові системи, які оживляють ігри.
4. Багатокористувацька гра: CryEngine — це потужний інструмент для розробки багатокористувацьких ігор, який включає підтримку багатокористувацьких ігор, оптимізацію мережі тощо.
5. Оптимізація: CryEngine має відкритий інтерфейс програмування (API), який дозволяє використовувати спеціальні інструменти та розширення двигуна, дозволяючи розробникам створювати власні рішення та потреби.

2.2.3 Unity

Unity — це кросплатформенний ігровий движок, розроблений Unity Technologies для створення відеоігор і симуляцій на різних пристроях, включаючи комп’ютери, ігрові консолі та мобільні пристрої. Його вперше було анонсовано на Всесвітній конференції розробників Apple у 2005 році як ігровий движок для Mac OS X, але з тих пір він розширився до 27 різних платформ.

Unity — це спеціально створений двигун, який підтримує 2D- і 3D-графіку з функцією перетягування та написана на мові програмування C#. Ігровий двигун особливо популярний для створення мобільних ігор і зосереджений на розробці мобільних платформ.

Хоча 2D-конвеєр Unity3D є менш зрілим, ніж 3D-конвеєр, Unity є підходящою платформою для створення 2D-ігор, особливо якщо ви плануєте випускати свої ігри на кількох мобільних пристроях.

Агрегація також є хорошим варіантом для розробки VR, хоча наразі ринок VR дуже малий. Мобільні пристрої та PSVR є найбільшими ринками VR, і Партнерство готове перенести гру на різні платформи, такі як PS4 і ПК, або інші мобільні магазини.

Unity підтримує такі графічні API, як Direct3D на Windows і Xbox One, OpenGL на Linux, macOS і Windows, OpenGL ES на Android і iOS, WebGL на WebGL і рідні API на консолях.

Особливості:

1. Швидкість розробки: Unity має багато готових компонентів і бібліотек, які дозволяють розробникам створювати те, що вони хочуть, і простий інтерфейс, який дозволяє створювати ігри.
2. Спільнота: підрозділ має велику спільноту розробників, які діляться своїм досвідом і знаннями. Це допоможе вам швидко вирішити проблеми, що виникли під час розробки, і знайти більше корисних функцій.
3. Додаткові інструменти: Unity надає безкоштовні шаблони, форми, слова та інші елементи, які ви можете використовувати у своєму проекті.
4. 3D-ігри: Unity має потужний механізм 3D-розробки, який дозволяє створювати складні світи та ефекти.
5. Багато інструментів: Unity має багато інструментів розробки, таких як: Visual Studio, Rider, Unity Hub тощо, які допомагають розробникам працювати в комфортному середовищі та ефективно працювати над проектом.

Зробивши повний аналіз двигунів, я зробив свій вибір двигун Unity. Адже саме Unity є ідеальним ігровим двигуном для мобільних ігор, на ньому написано багато мобільних ігор. Саме цей двигун є дуже оптимізованим, та найкращим для створенням мобільної гри.

2.3 Підбір оптимального середовища розробки

2.3.1 Visual Studio Code

Visual Studio Code — це безкоштовний редактор коду, який підтримує кілька мов програмування та середовищ виконання. Його функціональність можна розширити додатковими бібліотеками. Редактор швидкий і чуйний, що дозволяє користувачам швидко розпочинати свої проекти.

Однією з найкращих переваг Visual Studio Code є те, що він має розширені функції, такі як просте автозавершення, графічні рішення, стрічка, редагування кількома курсорами та інтелект підказок.

Крім того, внутрішній контроль джерел із підтримкою Git спрощує керування проектами та співпрацю. Редактор регулярно оновлюється новими функціями та оновленнями з використанням останніх технологій і протоколи мовного сервера.

Однак різні мови програмування та плагіни можуть мати різні рівні підтримки, і в деяких випадках може знадобитися надати додатковий вміст. Загалом Visual Studio Code є потужним та інтуїтивно зрозумілим інструментом для розробників будь-якого рівня кваліфікації.

2.3.2 Visual Studio

Visual Studio, програма Microsoft, є одним із найпопулярніших інструментів розробки програмного забезпечення. Він містить усі інструменти, необхідні розробникам для створення, тестування та розгортання програм. Visual Studio підтримує такі мови програмування, як C#, C++ і Python, що дозволяє розробникам

вибирати мову, яка найкраще підходить для їхніх проектів. Крім того, продукт містить шаблони та звіти, які дозволяють розробникам ефективніше керувати своїм часом.

Visual Studio — це інтегроване середовище розробки, яке надає розробникам різноманітні інструменти для створення, редагування та програмування. До них належать кодеки, які підтримують завершення та очищення коду, вбудовані засоби редагування вихідного коду та інструменти рівня терміналу з контролем версій.

За допомогою Code розробники можуть легко змінювати та копіювати свій код. Розробники готові знаходити та виправляти помилки, а нарешті візуальні аніматори визначають прогалини в проекті.

Visual Studio підтримує автоматичне кодування, тестування та розгортання, а також співпрацю з клієнтами та керування проектами. Це робить Visual Studio потужним програмним інструментом.

2.3.3 Rider

JetBrains Rider — це інтегроване середовище розробки, яке дозволяє розробникам створювати різні типи програм, наприклад консолі, мобільні програми, веб-програми та ігри. Rider підтримує багато мов програмування, включаючи C#, VB.NET, F#, JavaScript і TypeScript. Він також підтримує різні фреймворки, такі як .NET, Mono, .NET Core, Xamarin, ASP.NET, ASP.NET Core та WPF.

Головна мета Rider — спростити розробку додатків для розробників .NET. Це робиться за допомогою таких функцій, як завершення коду, генерація коду, рефакторинг, навігація, тестування коду тощо. Крім того, Rider постачається з усіма необхідними інструментами, включаючи налагоджувач, модульні тести, клієнт NuGet, інструменти бази даних, вікно попереднього перегляду WPF XAML та інтеграцію з редактором Docker і Unity.

З точки зору швидкості та ефективності пам’яті, Rider є однією з найкращих і найпотужніших IDE для розробників .NET. Загалом, Rider — це потужний

інструмент розробки програм, який допомагає розробникам кодувати та створювати чудові програми.

Rider використовує IntelliJ як програму з редактором для зміни вихідного коду. IntelliJ відстежує, що робить користувач, і викликає мовні служби для завершення речень та інших дій. Деякі мови, такі як JavaScript і TypeScript, повністю реалізовані в Rider, але для інших мов, таких як C#, VB.NET і F#, Rider використовує внутрішню обробку ReSharper. Інтерфейс IntelliJ відображає список, повернутий ReSharper. Для деяких мов інтерфейс IntelliJ надає додаткові інструменти, такі як елементи керування, але для мов .NET це тонкий рівень інтерфейсу, який забезпечує редагування та іншу інфраструктуру.

2.3.4 Порівняння середовищ розробки

Microsoft Visual Studio та JetBrains Rider — це засоби розробки, які надають можливості для розробки програмного забезпечення. Visual Studio — продукт Microsoft, який є основою для розробки додатків у середовищі .NET. Це єдине інтегроване середовище розробки, яке включає інструменти для створення прототипів, проектування, моделювання та тестування програм.

З іншого боку, Rider від JetBrains — це кросплатформна альтернатива Visual Studio, яка має багато функцій для створення програм. Rider побудовано на мові програмування Java і містить більшість функцій Visual Studio IDE.

Щоб порівняти Visual Studio з Rider, потрібно зосередитися на ключових функціях, таких як керування кодом, зміна розміру, IntelliSense, глобальний пошук і інструменти пошуку. Обидва засоби розробки мають багато функцій, які дозволяють розробникам створювати програмне забезпечення швидко та ефективно.

Однак між ними є відмінності, особливо коли йдеться про плагіни та функції. Visual Studio існує вже більше двох десятиліть, тому вона має широкий вибір плагінів і функцій. Це дає розробникам доступ до широкого спектру інструментів,

що робить зручнішим у використанні. Rider, з іншого боку, має менше плагінів і функцій, але все ще є дуже потужним інструментом розробки програм.

Перш ніж обирати між Visual Studio та Rider, розробники повинні розглянути вимоги свого поточного проекту та перевірити функції та плагіни, які їм потрібні в обох засобах розробки.

Rider базується на розробці, підтримує інтеграцію з Git і дозволяє використовувати ORM сторонніх розробників. Це програма зі зрозумілим та інтуїтивно зрозумілим інтерфейсом, який можна налаштувати відповідно до ваших потреб і ORM. Rider є більш ефективним, ніж Visual Studio, оскільки він працює та компілює код швидше. Дозволяє швидко створювати проекти за допомогою додаткових функцій створення. Rider також забезпечує кращий аналіз коду, що особливо важливо для великих програмних проектів.

Так як для створення гри було використано ігровий двигун Unity, програмний код для неї розробляється за допомогою Rider, адже це найшвидше середовище розробки, на найкраще для створення мобільних ігор на Unity.

2.4 Висновки до розділу 2

Тому при виборі найкращого середовища для розробки слід враховувати специфіку проекту та відповідність обладнання та технології їх потребам. Наприклад, для легких проектів краще використовувати наприклад Visual Studio Code. Для створення складних програм може знадобитися інтегроване середовище розробки, таке як Rider, адже він швидший.

При аналізі відеоігор слід враховувати наступні моменти:

Різні середовища розробки мають різні характеристики, включаючи стабільність процесу та розмір.

Вибір програми запуску гри повинен залежати від деталей проекту та потреб користувачів.

Такі популярні ігрові середовища розробки, як Unity та Unreal Engine, можуть підійти для багатьох проектів завдяки своїм великим спільнотам користувачів, документації та підтримці.

Також, потрібно враховувати ціну середовища розробки та обладнання, яке необхідне для роботи з ним. Деякі середовища можуть бути досить дорогими та не є прийнятними для менш великих проектів або невеликих команд з обмеженим бюджетом. Тому, у випадку, якщо вартість середовища та необхідного для нього обладнання є високою, то може бути доцільно розглянути альтернативні варіанти, що більш підходять за фінансовими можливостями проекту.

Для менших проектів можна використовувати безкоштовні або дешеві інструменти, тоді як для великих проектів може бути необхідне інвестування у комерційні програмні рішення або налаштування власних інфраструктур. Проте, в будь-якому випадку, важливо забезпечити якісну та стабільну роботу середовища розробки для максимально ефективної роботи розробників та досягнення успіху проекту.

Крім того, важливо звернути увагу на підтримку та оновлення обраного середовища розробки. Якщо платформа не має постійних оновлень та підтримки, це може призвести до зменшення продуктивності та безпеки проекту. Тому, варто вибирати ті середовища, які мають активну спільноту розробників та постійно оновлюються. Наявність оновлень та підтримки також може забезпечити більші можливості для розширення та вдосконалення функціоналу проекту.

Так як для створення гри було використано ігровий двигун Unity, програмний код для неї розробляється за допомогою Rider, адже це найшвидше середовище розробки, на найкраще для створення мобільних ігор на Unity

Крім врахування технічних аспектів, важливо також звернути увагу на досвід та уміння розробників. Саме тому, на основі другого розділу було вирішено розробляти гру на Unity, та використовувати середовище розробки Rider.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка ігрових механік, та їх структура

Логіка гри реалізована за допомогою скриптів в Unity. Всі скріпти, які були написані, реалізують ігрові механіки, та визначають поведінку ігрових об’єктів.

Відеогра містить 42 скрипти. Основні скріпти та їх логіку можна поділити на деякі частини: інтерфейс, логіка монстрів, логіка рівня, інвентар, бойова система, збереження, логіка персонажа, крафт, опис персонажу.

3.2 Інтерфейс

Клас MainMenu. Метод PlayGame викликається при натисканні кнопки в головному меню гри та переходить на наступну сцену у списку. Отримуємо активну поточну сцену, і до її індексу додається 1 для переходу на наступну сцену. Метод ExitGame викликається при натисканні кнопки виходу з гри для завершення програми. Приклад із розробленої гри (рис. 3.1).

Рисунок 3.1 — Демонстрація головного меню гри

Клас Menu це скрипт для обробки подій в меню гри, який зупиняє час, дозволяє вийти з гри, та дає доступ до головних налаштувань.

Клас VolumeController відповідає за налаштування гучності звуку в грі. Він прив'язується до трьох слайдерів - для основної гучності, музики та звуків голосів. Скрипт зберігає значення гучності, щоб при наступному запуску гри гучність залишалась на тому ж рівні. Крім того, скрипт відповідає за зміну гучності всіх звукових джерел, які містяться в списку звуків.

Клас Indicators цей скріпт відповідає за показники здоров'я, їжі та води у грі. Він відстежує, скільки їжі та води є у гравця, а також як швидко вони споживаються з часом. Якщо рівень води та їжі стає надто низьким, то здоров'я починає падати.

Клас Questions цей скрипт контролює завдання у грі, яке вимагає вбити задану кількість монстрів. Завдання містить інформацію про кількість потрібних убивств та кількість вже здійснених убивств. Коли гравець вбиває монстра, лічильник убивств збільшується, а текст завдання оновлюється. Якщо гравець вбиває потрібну кількість монстрів, то завдання вважається виконаним, і виводиться сповіщення про те, що потрібно вбити фінального боса.

3.3 Логіка монстрів

Клас ScreamerManAI цей скріпт відповідає за поведінку монстра в грі. Він містить логіку здоров'я, пошкодження, звуків. Як тільки головний герой нанесе удар монстру, то запускається анімація “Hurt”, та програється звук болю, а його полоска здоров'я зменшиться. Якщо кількість монстрів вбитих монстрів буде більше 20, то з’являється сповіщення, що потрібно вбити Боса. Після того, як полоска здоров'я стає рівна нулю, то програється анімація смерті, та звук смерті, об’єкт пропадає зі сцени через 7 секунд.

Клас Enemy цей скрипт є частиною гри і реалізує поведінку ворожого монстра. Він відповідає за те, щоб ворог програвав анімації, відтворював звуки,

рухався по карті, діставав удари та атакував головного героя. Як тільки монстер з’являється на карті, то монстер за допомогою масиву ділянок, вибирає випадкову ділянку, де він патрулює, як тільки персонаж зайде в цю ділянку, то монстер прораховує чи являється цей об’єкт “Player” , та якщо головний герой тікає від монстра на достатню дистанцію, то він повертається назад на свою ділянку. Також при атакі, бігі, рухі, у монстера є анімації. Приклад із розробленої гри (рис. 3.2).

Рисунок 3.2 — Демонстрація ворога

Клас Damage цей скрипт додає звуковий ефект, коли монстер близько біля об’єкта “Player”, то зменшує здоров’я у головного героя, за допомогою зіткнення колайдера.

Клас BossController встановлюється початкове значення здоров'я, заповнюється зображення смуги здоров'я та зберігається посилання на головну камеру. Як тільки головний персонаж знаходить Боса та бачить його, то з’являється полоска здоров'я посередині зверху екрану, як тільки персонаж не бачить Боса, то полоска здоров'я пропадає з екрану. Як тільки головний герой нанесе удар Босу, то запускається анімація “Hurt”, та програється звук болю, , а його полоска здоров'я зменшиться. Якщо здоров'я стає меншим або дорівнює

нулю, то запускається анімація смерті, деактивується колайдер, встановлюється швидкість NavMeshAgent на нуль, заповнює смугу здоров'я нулем, лічильник вбивств збільшується на одиницю та перевіряється, чи досягнуто потрібну кількість вбивств. Якщо так, виконується перехід на нову сцену.

Клас BossBehaviour визначає відстань між босом-зомбі та ціллю (гравцем). Якщо відстань більше чим потрібно і позиція боса-зомбі не дорівнює позиції його ділянки, то бос-зомбі рухається до своєї ділянки. Якщо відстань більше, а позиція дорівнює позиції ділянки, то бос-зомбі дивиться на гравця і виконує анімацію атаки, через деякий час. Якщо відстань менше діапазону, але більше діапазону атаки, то бос-зомбі рухається до гравця, щоб вийти на дистанцію атаки. Якщо відстань атаки менше, то бос-зомбі виконує анімацію атаки і зупиняється на деякий час, після чого він продовжує рух до гравця. Якщо ж нічого з цього не проходить, бос-зомбі просто рухається до гравця. Також в цьому скрипті як тільки анімація атаки проходить ближче до кінця, то його швидкість ставиться на нуль, для того щоб він не біг під час анімації за гравцем, і як тільки анімація програлась, то його швидкість знову стає більше нуля. Цей скрипт допомагає реалізувати поведінку боса-зомбі в грі, де він може рухатися, атакувати та виконувати різні анімації в залежності від відстані до гравця.

Також, монстри перевіряють за допомогою NavMeshAgent, чи можуть вони пройти деякі об’єкти на карті, як тільки зробити всі об’єкти статичними, та запекти карту, то всі монстри з NavMeshAgent, зможуть розуміти, де їм потрібно обходити.

3.4 Логіка рівня

Клас RespawnWater об'єкти води створюються випадковим чином в межах розміру об'єкту, до якого цей скрипт прикріплений, в даному випадку, це криниця. Раз в 0,5 секунд, створюється об'єкт води. Як тільки об'єкт створюється то він в колекції, починається перевірка, чи є вже об'єкт в колекції, якщо немає то

створюється новий, якщо є то ні. Якщо об'єкт пропадає, то коллекція очищається, викидає всі значення null, та повертає нову колекцію, в якій об'єкт не null.

Клас Respawn Zombie створює новий об'єкт зомбі через деякий час. Цей клас приймає префаб зомбі, та позицію в якій буде створений об'єкт, та його початковий оберт. Позиція створення об'єкта обчислюється як випадкова позиція всередині прямокутної зони з координатами. Якщо кількість об'єктів в колекції менша за 2, тоді новий об'єкт додається до колекції. Інакше, виконується метод ClearList(), який очищує список від об'єктів з недійсними посиланнями.

Клас GatherResources перевіряє час останньої атаки за допомогою змінної lastAttackTime та встановленого часу перезарядки атаки. якщо час з останньої атаки більше за встановлений час перезарядки атаки, то запускається анімація атаки знаряддям праці. На знаряддях праці є точка, за допомогою якої знаходяться та перетинаються колайдери. Проходимося по кожному колайдеру, та перевіряємо, чи є у нього компонент “TreesController” або “StoneController”. Якщо такий компонент є, то додається предмет в інвентар.

Клас StoneController це скрипт, який керує об'єктом каменем в грі. В класі, є публічне поле, в якому є кількість ударів, необхідних для розрушення каменю. Також коли по каменю йде удар, то програється анімація, та звук удару по каменю. Коли по каменю буде зроблено деяка кількість ударів, він буде знищений.

Клас TreesController це скрипт, який працює . В класі, є публічне поле, в якому є кількість ударів, необхідних для того, щоб зрубати дерево. Також коли по дереву йде удар, то програється анімація, я та звук удару по каменю. Якщо дерево, ще не було зрубано, то програється анімація тряски дерева, якщо кількість ударів досягла максимальної по дереву, то програється анімація падіння, і дерево знищується зі сцени через 3 секунди.

Клас ItemPickUp скрипт, який дозволяє гравцеві збирати предмети з зазначеного місця в грі. Клас має змінну item, яка містить інформацію про зібраний предмет, і змінну parent, яка містить інформацію про батьківський об'єкт, який містить цей предмет. Також клас містить кнопку, яка стає активною коли гравець

знаходиться біля предмету, також коли гравець натискує кнопку, предмет переходить в інвентар до гравця. Якщо предмет був підібраний, то його батьківський об'єкт видаляється. Метод OnTriggerStay відслідковує, чи перебуває гравець в межах зони зібрання, тоді як метод OnTriggerExit відслідковує, чи покинув гравець зону зібрання. Приклад із розробленої гри (рис. 3.3).

Рисунок 3.3 — Демонстрація підбору предмета

3.5 Інвентар

Клас Inventory реалізує систему інвентаря, яка дозволяє гравцеві збирати, зберігати, викидати і використовувати предмети в грі. Отже, при додаванні нового предмета в інвентар, спочатку перевіряється, чи інвентар не заповнений повністю. Якщо інвентар заповнений, то повідомлення про це відображається за допомогою об'єкту GameMassage1. Якщо інвентар не заповнений, то перевіряється, чи є в інвентарі стаки не заповнених предметів з таким же іменем та ID, як у додаваному предметі. Якщо є, то до цього стаку додається новий екземпляр предмету. Якщо такого стаку не існує, то шукається перший порожній слот в інвентарі, і до нього додається новий предмет або створюється новий стак. Якщо всі слоти заповнені,

то новий предмет не додається. При видаленні предмета з інвентарю використовується інший метод, який шукає індекс предмета в списку items і замінює його на порожній предмет. Також скидається стак предмету в InventorySlot, що відповідає цьому предмету. Якщо стак відповідного предмету стає порожнім, то він повністю видаляється з інвентарю. Метод RemoveEat() видаляє задану кількість предметів зі стаку, якщо такий стак існує. Якщо після видалення предметів стак стає порожнім, він повністю видаляється з інвентарю. Методи AddItem() та RemoveItem() використовуються для додавання та видалення кількості предметів, які не обмежені одним стаком. Для додавання кількості предметів використовується цикл, який викликає метод Add() для кожного екземпляру предмета. Для видалення кількості предметів перевіряється, чи в інвентарі є достатня кількість предметів.

Клас InventorySlot, в ньому є методи які використовуються для збільшення кількості елементів у слоті. Збільшується кількість елементів в слоті на 1, якщо він не є повним, і відображає нову кількість у відповідному текстовому полі. Збільшується кількість елементів в слоті на кількість елементів, що передані в якості предмета. Також у класі є метод, який викликається, коли слот має бути очищений. Він встановлює значення всіх змінних до значень за замовчуванням, щоб показати, що слот порожній. Є також метод, який викликається, коли елемент було видалено зі стеку в слоті. Він вимикає кнопки видалення та використання, оскільки в слоті не залишилося жодного елемента, і відключає відображення кількості елементів у відповідному текстовому полі. Також метод викликається при натисканні на кнопку видалення елементу зі слоту. Він видаляє елемент зі стеку і генерує новий елемент випадкової позиції поблизу гравця в грі. Також відтворюється звук видалення елементу. Клас містить в собі ще метод, який відповідно виконує дії згідно типу елемента. Додається переданий об'єкт типу предмет до інвентарної клітинки. Він встановлює значення змінних об'єкту класу, які відповідають за зображення предмету, можливість видалення та використання. Якщо предмет має назву "Empty", то викликається метод для очищення клітинки. Також в класі є метод, який додає один раз одиницю предмету до інвентарної

клітинки. Якщо клітинка порожня, то метод нічого не робить. Також в цьому класі зроблено очистку клітинок, очищує інвентарну клітинку в тому випадку, якщо вона не містить жодного предмету. Є різні варіанти використання предметів із інтвентарю залежно від предметів екземплярів класу, який викликає цей метод. Якщо предмет їстівний, то метод спробує змінити кількість їжі у гравця та видалити один екземпляр цього предмета з інвентаря. За такою ж схемою гравець може випити предмет, надіти його як зброю, чи полікуватися. Після виконання дій залежно від варіанту використання, буде перевірка чи є аудіо джерело та звук, якщо є, то відтворить цей звук один раз.

Клас Item у цьому класі є кілька властивостей, таких як назва предмета, іконка, список інших предметів, ідентифікатор та деякі властивості, що описують те, як предмет можна використовувати (чи можна його з’їсти, пити, який вплив має на здоров'я, голод та спрагу тощо).

Клас InventoryUI у ньому відбувається ініціалізація деяких змінних та об'єктів. Зокрема, він знаходить всі інвентарні слоти на сцені та додає до них різні обробники. Крім того, він викликає метод UpdateUI(), щоб відобразити вміст інвентаря на початку гри. SyncronizeItem() - цей метод відповідає за синхронізацію змісту інвентаря між всіма слотами та інвентарем. Також є метод, який викликається коли гравець намагається використовувати зброю. Якщо жодна зброя ще не активна, то метод вмикає зброю та встановлює її пошкодження. Якщо зброя вже активна, то метод вимикає її. Це потрібно, наприклад, якщо гравець зберігає свій прогрес і відновлює гру з місця, де він зупинився - цей метод допоможе зберегти стан інвентарю. AllWeaponActive(): цей метод призначений для вимкнення всіх зброї, щоб гравець міг знову вибрати зброю, яку він хоче використовувати. Якщо взятий предмет зброї, то вмикається кнопка атаки, або добування ресурсів. Також, якщо взята зброя, або предмет для добування ресурсів, то всі кнопки, та об’єкти які на даний момент взяті вимикаються разом з їх кнопками. Приклад із розробленої гри (рис. 3.4).

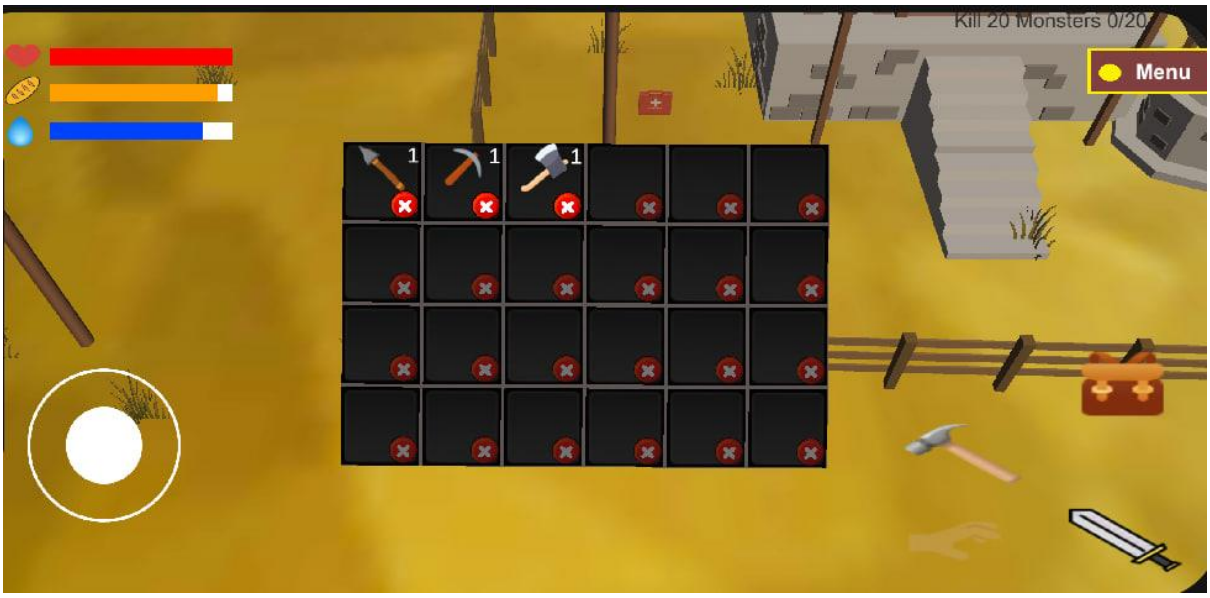


Рисунок 3.4 — Демонстрація UI інвентаря

3.6 Бойова система

Клас AttackCharacter відповідає за реалізацію атаки головного персонажа. Він перевіряє, чи пройшов достатній час з моменту попередньої атаки, і якщо так, то запускає анімацію атаки, відтворює звук атаки та збільшує час останньої атаки. Далі він використовує метод сфери для знаходження всіх ворогів, які перебувають у зоні атаки головного персонажа, та запускає цикл, в якому кожному ворогу, який має певний компонент, що відповідає за його поведінку, наноситься певний рівень пошкодження. Метод сфери використовується для візуалізації радіуса атаки, який показується на сцені як куля, це зроблено для того, щоб перевіряти чи зіткнулись колайдери з колайдером ворогу. Приклад із розробленої гри (рис. 3.5).

Рисунок 3.5 — Демонстрація битви з ворогом

3.7 Збереження прогресу

Клас SaveInventory зберігає список інвентарних, який конвертує список слотів в список об'єктів. Далі метод перетворює список у JSON формат і предмети в слотах зберігаються. Далі перевіряється, чи існує збереження для інвентарю за ключами. Якщо так, завантажується рядок зі збереженням, розпаковується і зберігається результат. Далі по списку слотів знаходимо предмет за його ідентифікатором та присвоюємо його до відповідного слоту, разом зі збереженням кількості предметів у слоті. Оновлюємо інтверфейс користувача, приймаємо ідентифікатор предмета та повертаємо відповідний об'єкт.

Клас SaveIndicators у цьому класі створюється змінна indicators, яка є містить інформацію про показники гравця. Зберігається значення показників у локальне сховище за допомогою ключів. Використовуються ці ж ключі для отримання значень з локального сховища та присвоюється їх змінним.

Клас SaveTransform відповідає за зберігання та завантаження позиції персонажу в грі. Для зберігання використовується PlayerPrefs - система зберігання даних у вигляді ключ-значення на локальному комп'ютері. В коді також є змінна

id, яка використовується як індекс для зберігання позиції та обертання об'єкта. Клас також містить зберігання та завантаження позиції персонажу.

3.8 Крафтова система

Клас CraftSlotUI приймає рецепт як аргумент та встановлює властивості UI-елементів, що відповідають цьому рецепту. Він створює іконки для кожного інгредієнта у рецепті та встановлює зображення для кнопки крафта.

Клас CraftUi Даний скрипт містить код для відображення інтерфейсу створення предметів у грі. У ньому використовуються різні змінні та об'єкти, такі як список рецептів, графічні об'єкти слотів для створення предметів, а також звукові ефекти. С самого початку відбувається ініціалізація графічних об'єктів слотів для створення предметів. Для кожного рецепту створюється окремий об'єкт слоту, який потім додається до списку слотів. Потім відбувається оновлення графічного інтерфейсу, зокрема встановлюється активність кнопок створення предметів в залежності від того, чи відповідає поточний інвентар рецепту для створення предмету. Відображення інтерфейсу створення предметів при кліку на відповідну кнопку. Також відтворюється звуковий ефект, відбувається створення предмету при кліку на відповідну кнопку. Після цього відбувається оновлення графічного інтерфейсу. Приклад із розробленої гри (рис. 3.6).

Рисунок 3.6 — Демонстрація UI крафтової систем

3.9 Логіка головного персонажу

Клас CustomCharacterController який встановлює параметри анімації, відтворює звук кроків, якщо умова виконується, тобто швидкість персонажа більша за нуль і звук не відтворюється. "Update()" - метод, який викликається кожен кадр. У ньому зчитується вхідний сигнал джойстика, встановлюються параметри анімації в залежності від цього сигналу. Якщо джойстик натиснутий, то виконується ходьба, якщо натиснути джойстик більше, то персонаж бігає. Приклад із розробленої гри (рис. 3.7).

Рисунок 3.7 — Демонстрація бігу персонажу

3.10 Тестування відеогри

Тестування було проведено на різних мобільних девайсах. Заохочено було грати в мобільну відеогру більше 5 людей, під час розробки механік. Проаналізовано всі відгуки реальних людей, та на основі цих фактів, та описів недоліків, було вирішено багато різних проблем гри. А саме багато користувачів скаржилися на незручний інтвентар. Проблема заключалась в тому, що предмети в інвентарі було незручно використовувати, адже кнопка використання на телефонах була занадто маленькою, та неклікабельною. Цю проблему було

вирішено, було зроблено так, щоб при натисканні на весь предмет в інвентарі він міг використовуватись.

Було проведене тестування всієї гри та на основі цих тестувань, визначено, що відеогра відповідала всім потрібним для легкої гри умовам. А саме було протестовано монстрів, монстра-боса, протестовано систему добування ресурсів, систему крафту, інвентарю, системи збереження прогресу. Також на основі відгуків, користувачі скаржилися на дуже швидкий голод, та зневоднення. На основі цієї інформації, було спростовано системи голоду, та зневоднення.

Під час тестування, була знайдена проблема з тряскою екрану. Проблема заключалася в тому, що на комп'ютері гра не підвисала, а на телефоні були якісь підвісання під час бігу персонажа. На основі цієї інформації, та перевірки коду, була знайдена проблема, та вона була вирішена.

На основі всіх відгуків, було виделіно плюси гри:

1. Керування зрозуміле для користувача.
2. Інвентар розроблений зручно.
3. Крафтова система працює відмінно.
4. Добування ресурсів.

1. Тест-кейс: Перевірка головного меню

   Кроки:

   1. Запустити гру.
   2. Переконатись, що відображається головне меню.
   3. Перевірити наявність опцій (звук, музика).
   4. Переконатись, що є кнопка виходу.
   5. Натиснути кнопку Грати.

      Очікуваний результат: Гра має розпочатися.
2. Тест-кейс: Перевірка налаштувань звуку та музики

   Кроки:

1. Запустити гру.
2. У головному меню виберіть опції.
3. На сторінці параметрів перевірити повзунки для регулювання звуку та музики.
4. Змінити значення повзунків на різні значення.
5. Переконатись, що звук і музика змінюються.
6. Вийти із налаштувань.

   Очікуваний результат: Звук та музика повинні бути налаштовані відповідно до вибраних значень.

3. Тест-кейс: Перевірка переміщення персонажа

   Кроки:

   1. Запустити гру.
   2. Почати гру.
   3. Переконатись, що персонаж відображається на екрані.
   4. Використати джойстик керування для переміщення персонажа в різні напрямки.
   5. Перевірити, чи персонаж переміщається відповідним чином.

      Очікуваний результат: Персонаж повинен вільно переміщатися ігровим світом.

4. Тест-кейс: Перевірка битви з монстрами

   Кроки:

   1. Запустити гру.
   2. Почати гру
   3. Переконатись, що персонаж знаходиться поруч із монстром.
   4. Використовувати зброю персонажа для атаки монстра.
   5. Перевірити, що здоров'я монстра зменшується при кожній успішній атаці.
   6. Перевірити, що персонаж отримує шкоду від монстра при контакті.
   7. Продовжувати бій до поразки монстра чи смерті персонажа.

Очікуваний результат: Здоров'я монстра має зменшуватись при атаці персонажа, а здоров'я персонажа має зменшуватись при контакті з монстром.

5. Тест-кейс: Перевірка інвентарю та системи крафту

   Кроки:

   1. Запустити гру.
   2. Почати гру.
   3. Переконатися, що персонаж має інвентар.
   4. Перевірити, що інвентар порожній.
   5. Підібрати предмети, які можна підняти чи скрафтити.
   6. Переконатись, що предмети з'являються в інвентарі.
   7. Використовувати систему крафта для створення нових предметів із доступних ресурсів в інвентарі.
   8. Перевірити, що нові предмети з'являються на інвентарі після крафта.

   Очікуваний результат: Інструмент повинен відображати підібрані предмети, а система крафту повинна створювати нові предмети з доступних ресурсів.

3.11 Висновки до 3 розділу

В цьому розділі було розглянуто розробку гри, та її функіонал. Оцінюючи всі зазначені теми, можна сказати, що вони є важливими складовими будь-якої гри. Отже, розробка всіх цих складових допомагає створити гру, яка буде цікавою та залучати гравців у свій унікальний світ. Розробляючи цю зру, було зроблено багато роботи з тестуванням, та розроблений цікавий світ для користувача.

ВИСНОВКИ

У результаті виконання даної дипломної роботи було розроблено мобільну відеогру на Android в жанрі “Виживання”. Під час виконання даної роботи було досліжено створення мобільної відеогри.

Дана дипломна робота проаналізувала популярність мобільних ігор, жанри відеоігор та ринок мобільних ігор. Аналіз існуючих відеоігор є корисним для розробників, які хочуть створити успішну гру, та геймерів, які хочуть знайти найкращу гру для себе. Жанр виживання дуже популярний у геймерській індустрії, оскільки дає гравцям відчуття хвилювання. Ключовими факторами успіху будь-якої гри є задоволення потреб гравців, інновації та розвиток геймплею.

При виборі середовища для розробки проекту, варто враховувати його специфіку та потреби в обладнанні та технологіях. Для аналізу відеоігор важливо враховувати характеристики середовищ та вибрати програму запуску гри, яка відповідає деталям проекту та потребам користувачів. Найпопулярніші ігрові середовища, такі як Unity та Unreal Engine, можуть бути корисними завдяки своїм спільнотам та підтримці, але варто враховувати вартість та підтримку. Основуючись на цих критеріях, рекомендовано розробляти гру на Unity та використовувати середовище розробки Rider.

Була створена гра на основі проаналізованих жанрів, основи створення мобільного геймплею, середовищ розробки відеоігор. Проаналізовано недоліки гри, проведено тестування, під час якого виділили плюси гри:

1. Керування за допомогою джойстика.
2. Інвентарна система.
3. Крафтова система.
4. Добування ресурсів.

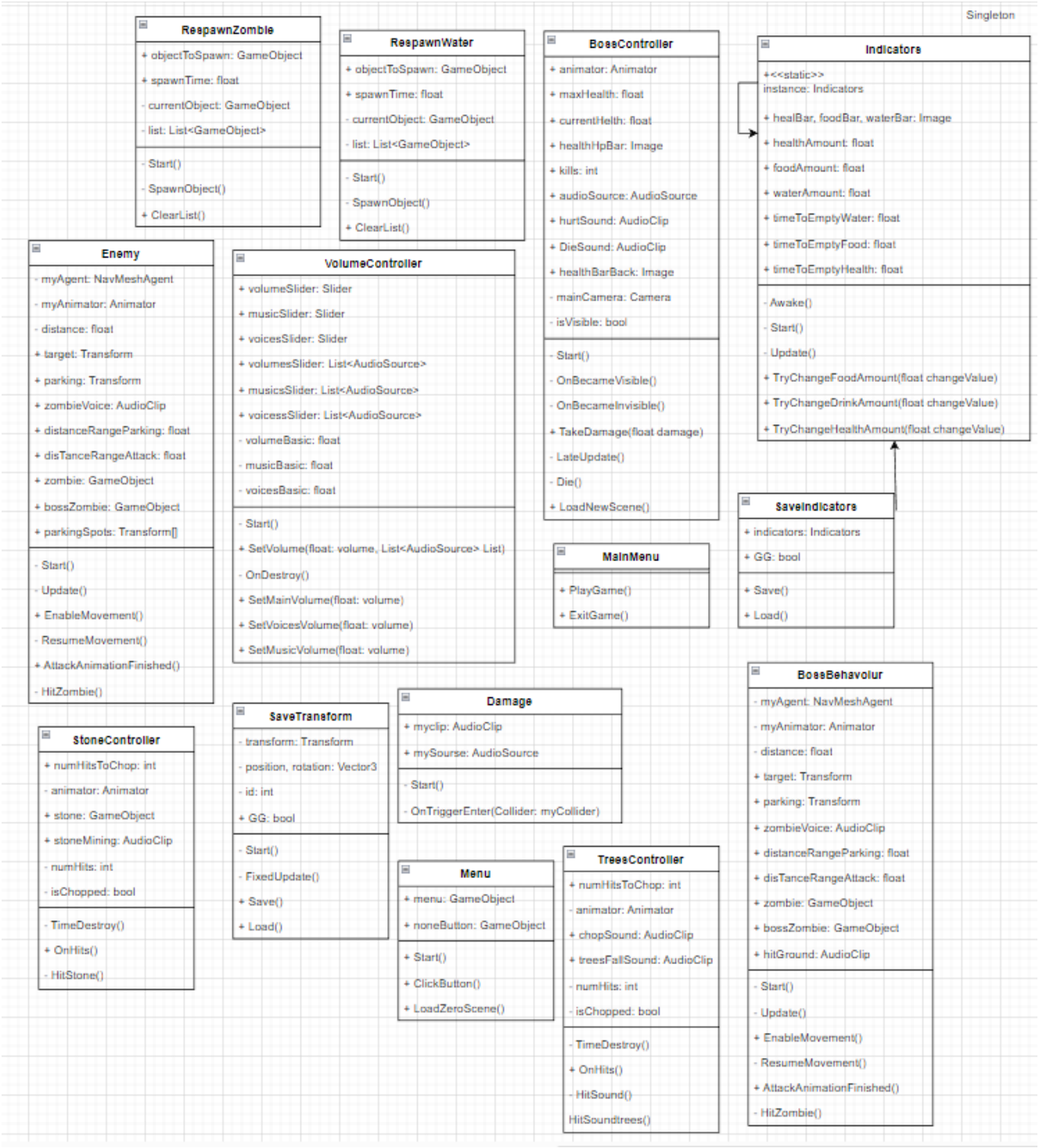
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Відео ігри [Електронний ресурс] – Режим доступу до ресурсу: https://www.twinkl.com/teaching-wiki/video-games
2. Повний посібник із жанрів і типів відеоігор [Електронний ресурс] – Режим доступу до ресурсу: https://codakid.com/video-game-genres/#Survival_Games
3. Як створити мобільну гру за 7 кроків [Електронний ресурс] – Режим доступу до ресурсу: https://www.pluralsight.com/blog/film-games/begin-create-first-mobile-game
4. 187 статистичних даних про мобільні ігри за 2023 рік, які вразять вас [Електронний ресурс] – Режим доступу до ресурсу: https://www.blog.udonis.co/mobile-marketing/mobile-games/mobile-gaming-statistics
5. Що таке гра на виживання? [Електронний ресурс] – Режим доступу до ресурсу: https://honeysanime.com/what-is-survival-game-gaming-definition-meaning/
6. Огляд Ark: Survival Evolved [Електронний ресурс] – Режим доступу до ресурсу: https://mmos.com/review/ark-survival-evolved
7. Don't Starve Together Огляд [Електронний ресурс] – Режим доступу до ресурсу: https://mmos.com/review/dont-starve-together
8. Що таке ігровий движок? [Електронний ресурс] – Режим доступу до ресурсу: https://fullscale.io/blog/what-is-game-engine/
9. Що таке Unreal Engine? [Електронний ресурс] – Режим доступу до ресурсу: https://www.educba.com/what-is-unreal-engine/

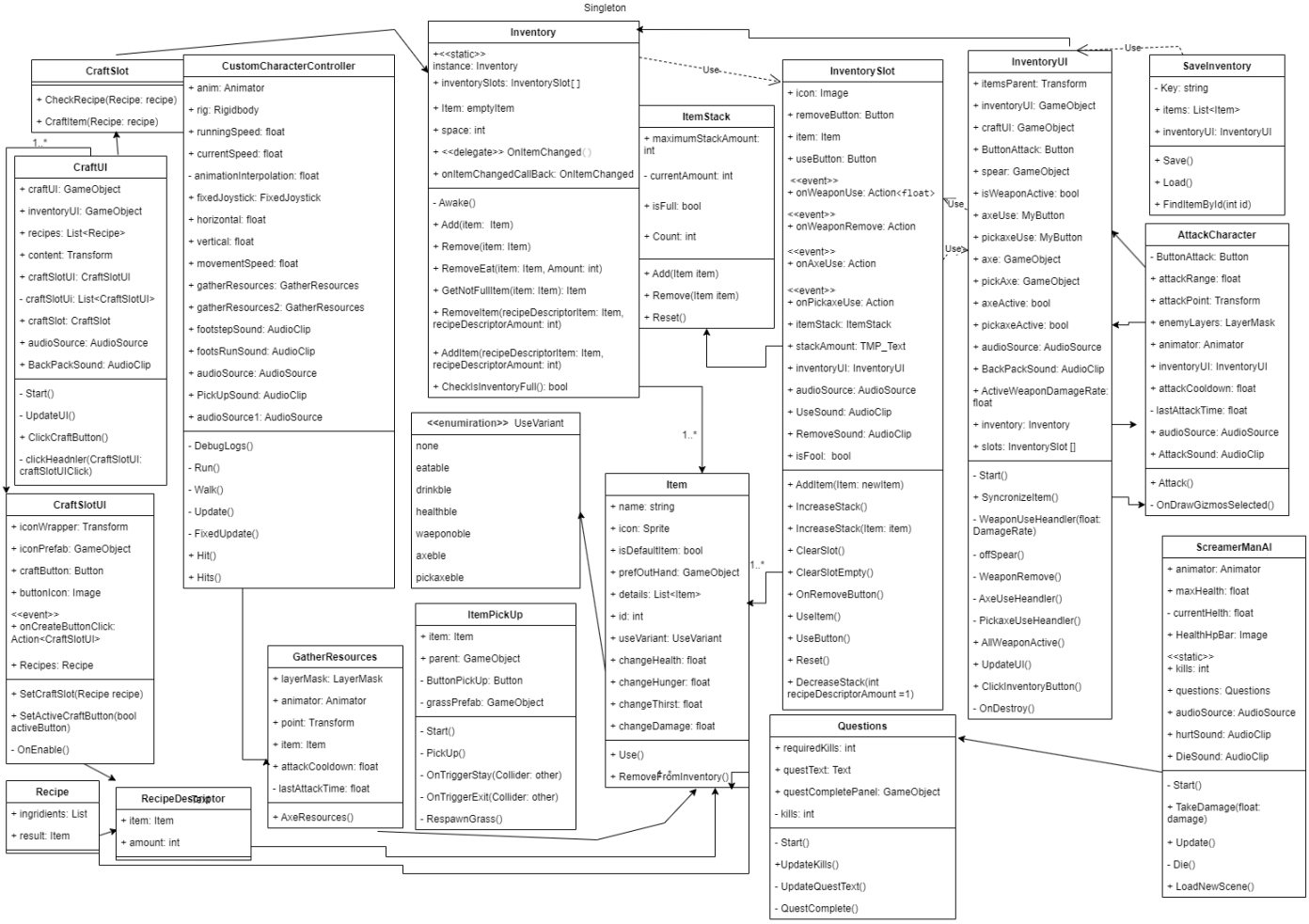
10. Посібник з ігрового движка Unity: як почати роботу з найпопулярнішим ігровим движком [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/unity-game-engine-guide-how-to-get-started-with-the-most-popular-game-engine-out-there/%20
11. Чи є Unity безкоштовним? [Електронний ресурс] – Режим доступу до ресурсу: https://gamedevbeginner.com/is-unity-free/
12. Який найкращий ігровий движок: чи підходить вам CryEngine? [Електронний ресурс] – Режим доступу до ресурсу: https://www.gamesindustry.biz/what-is-the-best-game-engine-is-cryengine-the-right-game-engine-for-you
13. Що таке код Visual Studio? Розширюваний редактор коду Microsoft [Електронний ресурс] – Режим доступу до ресурсу: https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html
14. Що таке Visual Studio? [Електронний ресурс] – Режим доступу до ресурсу: https://www.incredibuild.com/integrations/visual-studio
15. Створення .NET IDE за допомогою JetBrains Rider [Електронний ресурс] – Режим доступу до ресурсу: https://www.codemag.com/article/1811091/Building-a-.NET-IDE-with-JetBrains-Rider

ДОДАТОК А

Діаграма класів 1 частина

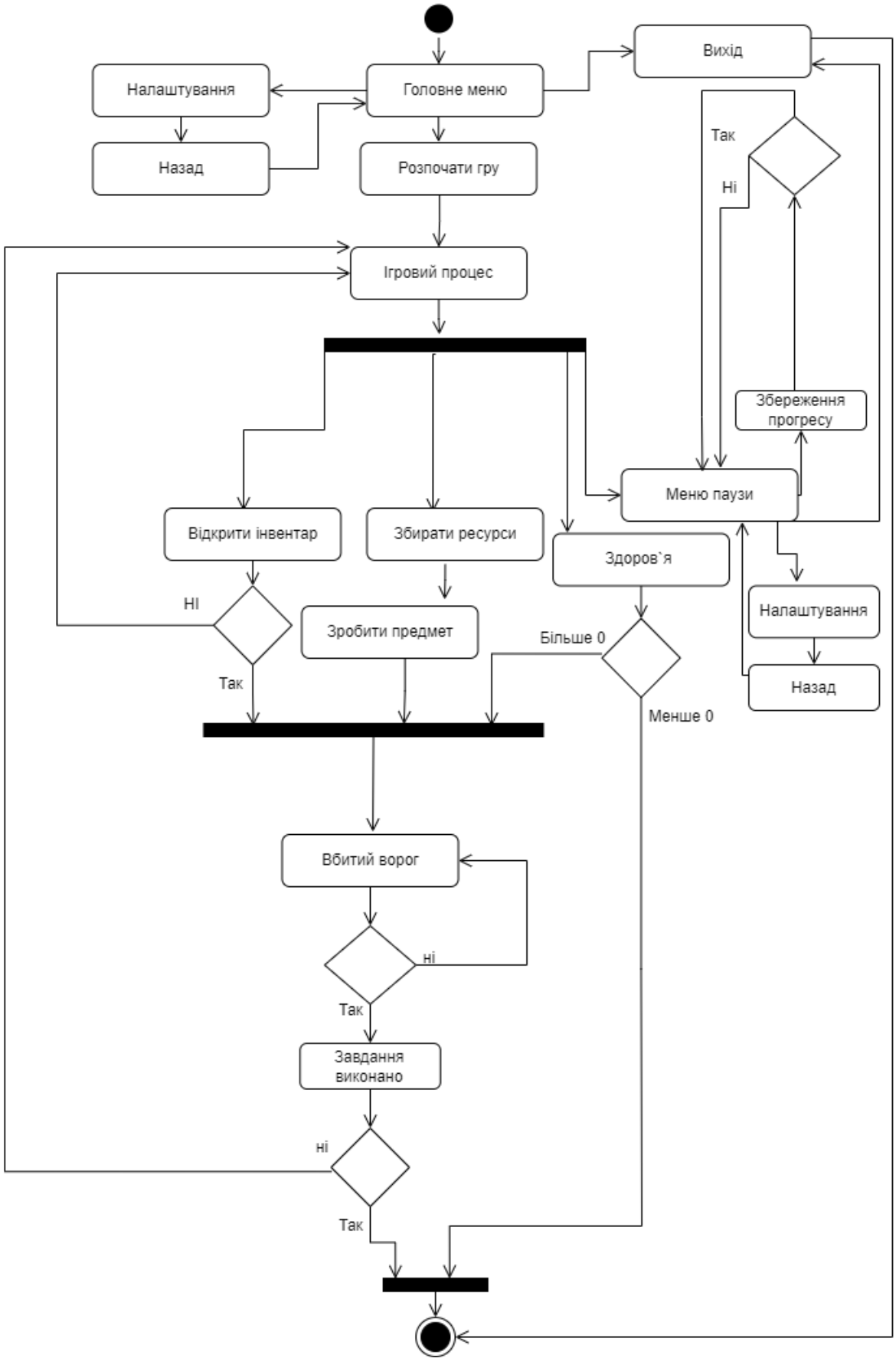


Діаграма класів 2 частина



ДОДАТОК Б

Діаграма діяльності



ДОДАТОК В

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка гри “Gropius” під платформу Android у жанрі “Виживання 2.5D” з використанням ігрового рушія Unity мовою C#

Виконав студент 4 курсу
Групи ПД-42
Вдовін Дмитро Едуардович
Керівник роботи
Доктор філософії
Дібрівний Олесь Андрійович

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** - покращення ігрового процесу в жанрі “Виживання 2.5D” з використанням ігрового рушія Unity мовою C#.
- **Об’єкт дослідження** - ігровий процес у грі в жанрі виживання.
- **Предмет дослідження** - мобільна відеогра в жанрі “Виживання” 2.5D для Android за допомогою рушія Unity.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз середовища програмування відеоігор.
2. Провести аналіз різних підходів та методів розробки мобільних ігор.
3. Провести аналіз програмних засобів.
4. Розробити архітектуру гри.
5. Розробити мобільну гру.
6. Провести тестування гри.

АНАЛІЗ АНАЛОГІВ

Показник	**Survival Island: Evolve**	**Last Pirate Island Survival**	**Survival Island: EVO 2 PRO**	**Gropius**
Платформи	Android,	Android	Android	Android
Наявність реклами у грі	-	-	+	-
Система голоду та зневоднення	+	+	-	+
Система ремесла	+	-	+	+
Система будівлі	+	-	+	-
Різноманітність музики	-	+	-	+
Різноманітність локацій	-	+	-	+

ВИМОГИ ДО ІГРОВОГО КОНТЕНТУ

1. Реалізувати систему їжи та води, для поповнення його голоду та зневоднення.
2. Реалізувати бойову систему.
3. Реалізувати штучний інтелект монстрів.
4. Реалізувати крафтову систему.
5. Реалізувати систему голоду та зневоднення, та система здоров’я.
6. Реалізувати добування ресурсів за допомогою кирки або топору.
7. Реалізувати систему респавну води, та їжі.
8. Реалізувати анімації персонажів.
9. Реалізувати ходьбу персонажу за допомогою джойстика.

5

КОНЦЕПТ ГРИ

1. Основна ігрова механіка полягає в використанні їжи та води, для поповнення голоду, зневоднення, та системи здоров’я, вода та їжа зменшується з часом, здоров’я зменшується якщо персонаж голодує, та зневоднений. Персонаж може поновлювати воду, їжу та здоров`я.
2. Вороги патрулюють в різних місцях, стають націлені на гравця, якщо ворог бачить головного героя.
3. Головний герой може використовувати крафтову систему, та систему інвентарю за допомогою інтерфейсу.
4. Головний герой може бігти або ходити по ігровому світу за допомогою джойстику.
4. В ігровому світі можливо добувати ресурси, за допомогою кирки або топору.
5. При збиранні їжі та води, вона знову з`являється через деякий час.
6. Гра розрахована на аудиторію від 12 років.

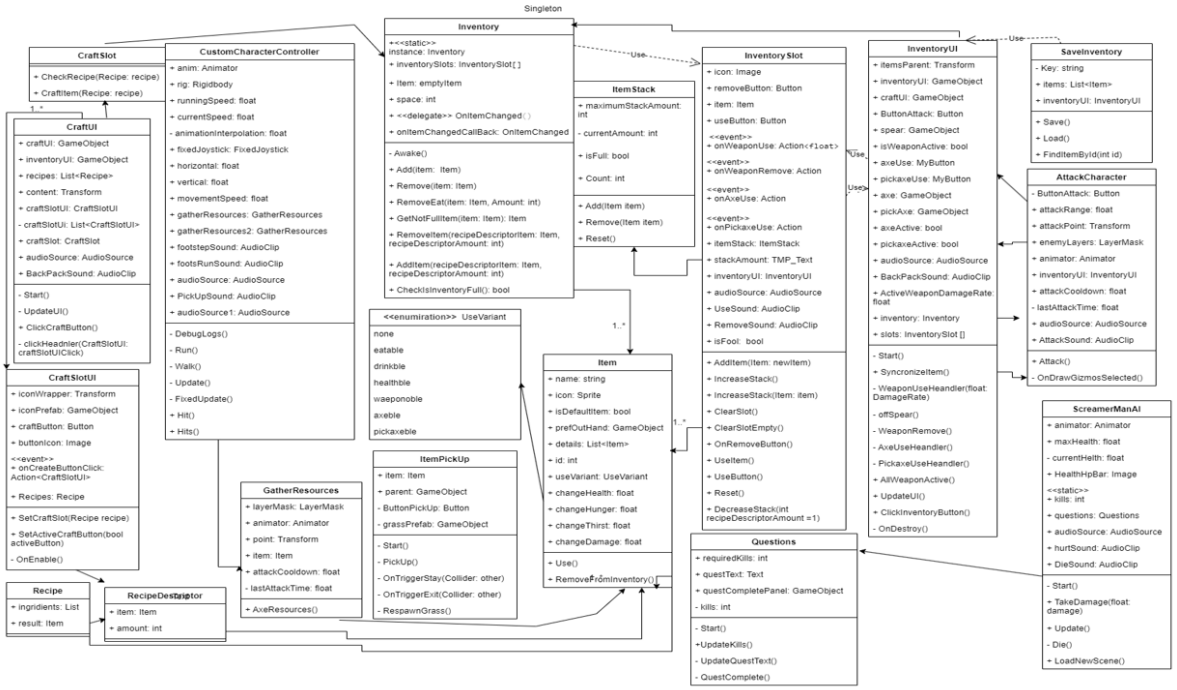
6

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Rider

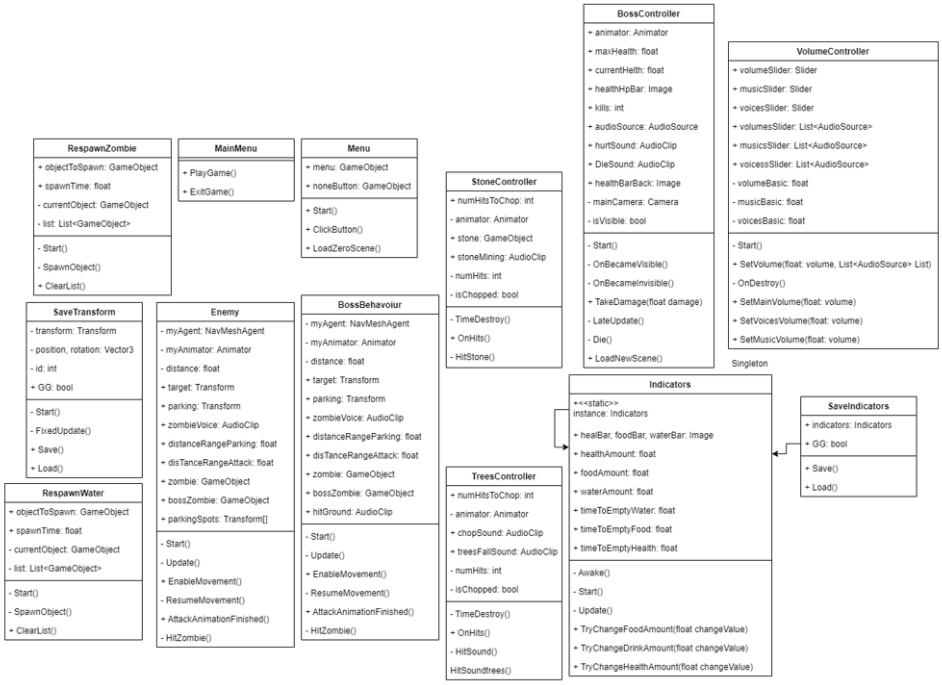
7

МЕТОДИ ТА КЛАСИ ПРОГРАМИ



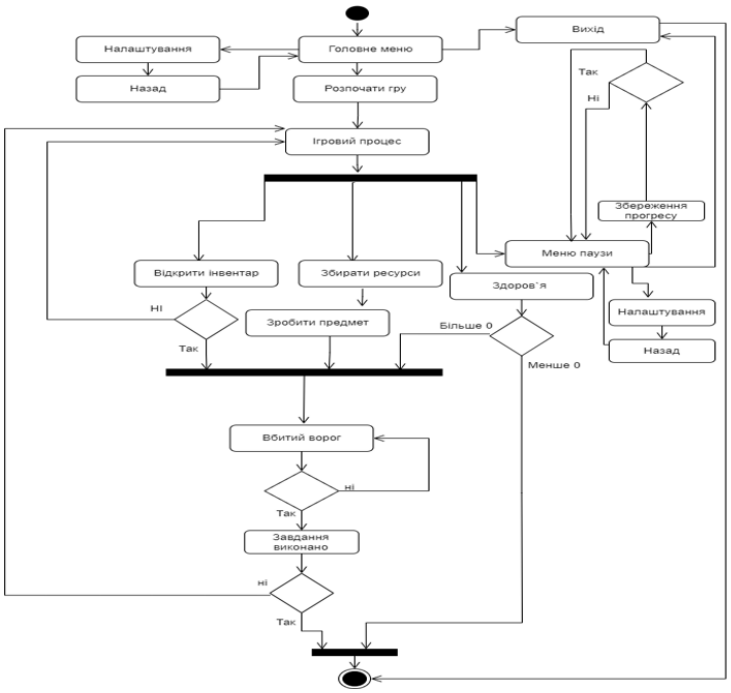
8

МЕТОДИ ТА КЛАСИ ПРОГРАМИ



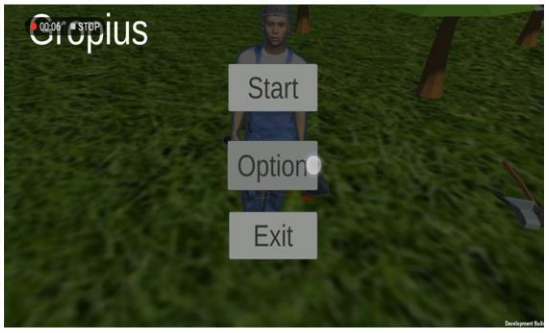
9

ДІАГРАМА ДІЯЛЬНОСТІ



10

ЕКРАННІ ФОРМИ



Головне меню гри

Геймплей гри

11

ВІДЕО ПРЕДСТАВЛЕННЯ ГРИ

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Вдовін Д.Е. Дослідження відеоігор в жанрі “виживання”/ Матеріали всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інфокомунікаційних технологіях». 20.04., ДУТ, м.Київ 2023 с. 125-126.

2. Вдовін Д.Е. Сучасні інформаційні технології в україні і світі / IV Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку iot» 05.04., ДУТ, м.Київ 2023. с. 34-36.

ВИСНОВКИ

1. Досліджено історію мобільних відеоігор, проаналізовано аналіз середовища програмування відеоігор.
2. Проведено аналіз програмних засобів. Обрано двигун Unity, мову програмування C#.
3. Досліджено та проаналізовано основні алгоритми розробки мобільних відеоігор.
4. Створено архітектуру гри.
5. Розроблено ігровий додаток.
6. Протестовано гру за допомогою тест кейсів, при тестуванні знайдені помилки та зроблено виправлення їх.

ДЯКУЮ ЗА УВАГУ!