КВАЛІФІКАЦІЙНА РОБОТА

Група ІПЗс-2019
Рудий А.Р.

2023

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Рудий Андрій Володимирович

УДК 340.11

Розробка комп'ютерної гри з використанням технології Roblox Studio.

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавра

Нормоконтроль	Студент
_____Стисло О.В. (підпис, дата, розшифрування підпису)	_____Рудий А.Р. (підпис, дата, розшифрування підпису)
Допускається до захисту Завідувач кафедри	Керівник роботи
_____к.т.н., доц. Пашкевич О.П. (підпис, дата, розшифрування підпису)	_____ к.ф-м.н., доц. Пашкевич О.П (підпис, дата, розшифрування підпису)

Івано-Франківськ – 2023

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ «_____»
_____ 202__ року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Рудий Петро Володимирович

(прізвище, ім’я, по батькові)

1. Тема кваліфікаційної роботи

Розробка комп'ютерної гри з використанням технології Roblox Studio.

керівник роботи:

Пашкевич Олег Петрович, кандидат фізико-математичних наук

затверджена наказом вищого навчального закладу від « 11» листопада 2022 року № 155/1НВ

2. Термін подання студентом роботи 14.06.2023

3. Вихідні дані роботи: Мова програмування Lua, Luau

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

1. Опис наявних жанрових аналогів
2. Розробка основного функціоналу
3. Реалізація функціоналу та механік

5. Дата видачі завдання 10.10.2022

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд існуючих аналогів	19.02.2023	Виконано
2.	Проектування та розробка GDD	18.03.2023	Виконано
3.	Розробка UI інтерфейсу	09.04.2023	Виконано
4.	Проектування та розробка основного функціоналу	28.04.2023	Виконано
5.	Оформлення пояснювальної записки	10.05.2023	Виконано
6.	Оформлення графічного матеріалу та підготовка до захисту роботи	25.05.2023	Виконано

Студент _____ Рудий А.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Бойчук А.М.
(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
14	демонстрація гейплею Foxhole	37	структура UI інтерфейсу

АНОТАЦІЯ

В результаті розробки гри в жанрі top-down action на платформі Roblox Studio було досягнуто успіху. Гра надає захоплюючий геймплей, який пропонує гравцям активний досвід та екшн-пригоди у віртуальному світі.

У першому розділі нашого дослідження був проведений детальний аналіз існуючих аналогів та конкурентів, де було виділено їх переваги та недоліки з погляду функціонального наповнення та дизайну. Це дозволило нам зробити вдалий вибір і покращити нашу гру відповідно до отриманих висновків.

У другому розділі ми зосередились на розробці механік гри та початкової архітектури коду. Ми провели проектування функціоналу та розробку прототипу гри, використовуючи інструменти, які дозволили нам візуалізувати та протестувати концепції геймплею. Крім того, ми створили дизайн у програмному середовищі з урахуванням сучасних концепцій дизайну, що сприяє привабливості та привертанню нових гравців.

Третій розділ присвячений програмній реалізації гри, включаючи написання коду та створення кінцевої архітектури. Для розробки гри на платформі Roblox Studio була використана мова програмування Lua (Luau).

КЛЮЧОВІ СЛОВА: GAMEDEV, ROBLOX STUDIO, LUA, LUAU.

SUMMARY

The development of a top-down action game on the Roblox Studio platform has resulted in a success. The game provides an exciting gameplay that offers players an active experience and action adventures in the virtual world.

In the first section of our research, we conducted a detailed analysis of existing analogues and competitors, where we highlighted their advantages and disadvantages in terms of functionality and design. This allowed us to make a good choice and improve our game in accordance with the findings.

In the second section, we focused on the development of game mechanics and initial code architecture. We designed the game's functionality and prototyped it using tools that allowed us to visualize and test gameplay concepts. In addition, we created a design in a software environment taking into account modern design concepts, which contributes to the attractiveness and attraction of new players.

The third section is devoted to the software implementation of the game, including writing code and creating the final architecture. The Lua programming language was used to develop the game on the Roblox Studio platform.

KEYWORDS: GAMEDEV, ROBLOX STUDIO, LUA, LUAU

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

Gamedev – ігрова розробка;

UI – інтерфейс користувача;

RS – Roblox Stduio

PC(ПК) – персональний комп’ютер

ВСТУП

Актуальність теми. Тема розвитку ігрової індустрії в Україні на сьогоднішній день є надзвичайно актуальною. Вона викликає неабиякий інтерес як серед широкої громадськості, так і серед фахівців галузі. Ігрова індустрія стала не просто розважальним сегментом, а справжнім глобальним бізнесом, який здатний не тільки забезпечити значні прибутки, але й сприяти розвитку економіки та культури країни.

Перш за все, слід зазначити, що Україна вже довгий час є центром розробки ігрових програмного забезпечення (ігрових девелоперів). Багато українських студій отримали світове визнання та нагороди за свої ігри, які отримали популярність як на внутрішньому ринку, так і за його межами. Це створює основу для подальшого розвитку галузі та залучення інвестицій.

Популярність ігрової індустрії в Україні величезна серед молоді, а це означає, що ігри є потужним інструментом для залучення молоді до творчого та інноваційного мислення. Розвиток цієї галузі може стимулювати розумовий розвиток, творчість та навички проблемного мислення у молодого покоління. Крім того, ігри можуть бути використані для популяризації освітніх тем, таких як наука, історія, культура тощо, що сприятиме зростанню загальної освіченості суспільства.

Україна також має потенціал стати провідною країною у сфері виробництва ігрового обладнання та віртуальної реальності. Сучасні технології та

наукові розробки в цій галузі розкривають широкі перспективи для розвитку індустрії. Ігрова індустрія може стати новим джерелом експортних доходів для України та збільшити її присутність на світовому ринку.

Зазначимо також, що розвиток ігрової індустрії в Україні сприятиме залученню іноземних інвестицій, створенню нових робочих місць та підтримці талановитої молоді. Інвестори все більше розуміють потенціал цієї галузі та готові вкладати кошти в українські ігрові студії та проекти. Це відкриває шлях для розвитку інноваційних технологій, обміну досвідом з провідними світовими гравцями та сприяє підвищенню конкурентоспроможності української галузі.

Отже, актуальність теми розвитку ігрової індустрії в Україні незаперечна. Ця галузь має великий потенціал у сфері економіки, культури та освіти. Розвиток ігрової індустрії сприятиме технологічному прогресу, привертанню інвестицій та створенню нових можливостей для молоді. Важливо підтримувати та сприяти цьому розвитку, щоб забезпечити яскраве майбутнє для ігрової індустрії в Україні.
Мета роботи. Розробка клієнтської та серверної частини гри з використанням технології RobloxStudio, реалізація функціоналу та впровадження механік.

Об’єкт роботи. Розробка мультиплеєрної гри в жанрі “Top-Down action”.

Предмет роботи. Розвиток gamedev індустрії.

Завдання роботи. Відповідно до вибраної теми в роботі покладені такі задачі:

-пошук і аналіз існуючих аналогів: Дослідження наявних комп'ютерних ігор жанру Top-down action та їх оцінка з точки зору геймплею, графіки, звуку, функціональності та популярності серед гравців.

-визначення оптимального набору інструментів для розробки гри на платформі Roblox Studio, вибір мови програмування (Lua або Luau), розгляд можливостей і додаткових бібліотек для покращення розробки та підтримки гри.

-розробка прототипу: Створення простого прототипу гри, що демонструє основні механіки, управління персонажем, взаємодію з об'єктами та ворогами.

- Проведення тестування продукту: Виконання ретельного тестування гри для виявлення та виправлення помилок

Методи роботи. Roblox Studio: Інтегроване середовище розробки для створення і налаштування гри, включаючи розміщення об'єктів, налаштування фізики, скриптів та інтерфейсу.

Lua та Luau: Мови програмування, які використовуються в Roblox Studio для написання скриптів та логіки гри. Lua є основною мовою програмування, а Luau є її покращеною версією з додатковими можливостями.

Результати роботи. Результатом виконання кваліфікаційної роботи є мультиплеєрна комп’ютерна гра в жанрі top-down action .

Структура роботи. Розділи – 3. Загальний обсяг основної частини – 57 сторінок. Список використаних джерел – 20.

РОЗДІЛ 1. АНАЛІЗ РИНКУ ТА ЖАНРОВИХ АНАЛОГІВ

1.1 Ринок ігрової індустрії в Україні

У 2021 році вартість ігрової індустрії оцінювалась у 178,8 мільярда доларів, що більше на 14,5% в порівнянні з 2020 роком.

Це помітна різниця порівняно з прогнозом 2016 року, в якому передбачалося загальна вартість в 90 мільярда доларів на той самий період – велика різниця в 77% між цими цифрами, що свідчить про можливість подальшого прискорення зростання. Попередні прогнози передбачали, що до 2025 року ігрова індустрія буде вартувати 268 мільярдів доларів.

Ігри залишаються популярним хобі, а світова пандемія лише підкреслила це. Починаючи з тих темних днів, глобальні доходи ігрової індустрії мали перевищити 200 мільярдів доларів у 2022 році. Темп, з яким ми спостерігаємо зростання ігрової індустрії щороку, може означати, що вищезазначена оцінка є досить скромною.

Азія лідирує за обсягом гравців. Згідно з нещодавнім дослідженням DFC Intelligence у серпні 2021 року, виявлено, що на Азію припадає 1,48 мільярда гравців з загальної кількості 3 мільярда гравців у всьому світі. Іншими словами, 45% гравців базуються в Азії, а гравців в Європі складають 22% (або 715 мільйонів) від загальної кількості.

Роль ігрової індустрії в Україні та її зростання в останні роки. Він надає огляд основних аспектів цього сектору, включаючи ринковий потенціал, економічний вплив та перспективи майбутнього.

Ринковий потенціал:

Ігрова індустрія в Україні має значний потенціал для росту. За останні роки спостерігається зростання популярності відеоігор серед населення.

Висока доступність технологій, широке поширення смартфонів і високошвидкісного Інтернету сприяють розширенню аудиторії гравців.

Економічний вплив:

Ігрова індустрія в Україні вносить значний внесок у валовий внутрішній продукт (ВВП) країни. Розвиток цього сектору стимулює економічне зростання і сприяє створенню нових робочих місць.

Видання відеоігор в Україні також привертають інвестиції та сприяють розвитку інших суміжних галузей, таких як розробка програмного забезпечення та інформаційні технології.

Талановитість і технічний потенціал:

Україна має потужний потенціал у сфері розробки відеоігор. Країна відома своїми програмними розробниками та технічними спеціалістами.

Багато українських компаній успішно працюють у сфері розробки відеоігор та отримують міжнародне визнання.

Міжнародні успіхи:

Українські відеоігри отримали визнання на міжнародній арені. Деякі українські розробники створили ігри, які отримали нагороди та високі оцінки від геймерської спільноти та видань. Українські ігри також здобули популярність серед гравців за кордоном, що допомагає просувати український ігровий бренд та розширювати міжнародну аудиторію.

Підтримка держави:

Українська держава проявляє зацікавленість у розвитку ігрової індустрії та надає підтримку у вигляді фінансових стимулів та сприяння створенню інфраструктури для розвитку галузі.

Наприклад, в Україні існують програми державної підтримки для ігрових розробників, які надають фінансову допомогу та консультативну підтримку.

Майбутні перспективи:

За прогнозами, роль ігрової індустрії в Україні буде продовжувати зростати. За рахунок підтримки держави, талановитих розробників та широкого ринку, відкриваються нові можливості для розвитку галузі.

Очікується, що українські відеоігри продовжать отримувати визнання та займати все більшу частку на світовому ринку.

Загалом, ігрова індустрія в Україні знаходиться на шляху до стабільного та стійкого зростання. За підтримки держави, розвитку технологій та талановитих

розробників, українська ігрова індустрія має всі можливості стати гравцем на світовій арені. Це не тільки сприятиме економічному розвитку країни, але й підвищить її репутацію у глобальній ігровій спільноті. Зростання популярності відеоігор та посилення інтересу до цього сегменту розваг дозволяють очікувати подальшого зростання ігрової індустрії в Україні у найближчому майбутньому.

Мультиплеєрні ігри стали одним з найбільш важливих сегментів ігрової індустрії. Українська ігрова індустрія також не залишається осторонь цього тренду, з ростом популярності мультиплеєрних ігор серед гравців. Україна входить до числа лідерів серед країн, які розробляють мультиплеєрні ігри. Найпопулярніші жанри включають MMORPG, MOBA та шутери. Значна кількість компаній працюють у цьому сегменті, такі як Plarium, GSC Game World, 4A Games та інші.

Українська ігрова індустрія є досить конкурентною в цьому сегменті, що підтверджується успішними проектами та нагородами. Наприклад, компанія Vostok Games отримала нагороду від критиків за свій проект Survarium, а Plarium стала переможцем у номінації "Краща гра в жанрі стратегії" на GTP Indie Cup 2019.

Мультиплеєрні ігри приваблюють все більше гравців у світі, що веде до зростання популярності індустрії в цілому. Українська ігрова індустрія, зокрема, демонструє високий рівень в цьому сегменті та має потенціал для подальшого розвитку.

Наголошується, що успіх мультиплеєрних ігор в Україні в значній мірі пов'язаний з активною геймінговою спільнотою. Гравці активно займаються обговоренням ігор, обміном досвідом та побудовою спільнот, що сприяє підвищенню рівня геймплею та розвитку мультиплеєрних ігрових кланів і команд.

Помітний вплив на розвиток мультиплеєрного сегмента в Україні має також підтримка держави та інвесторів. Державні програми та фінансова підтримка сприяють створенню нових ігрових студій, привертають талановитих розробників та сприяють запуску якісних мультиплеєрних проектів.

Важливим фактором є також розвиток інфраструктури для гравців. З'являються спеціалізовані ігрові центри, де гравці можуть зустрічатися, спілкуватися та змагатися один з одним. Такі місця стимулюють соціальну взаємодію між гравцями та сприяють розвитку геймінгової культури в Україні. Популярність мультиплеєрних ігор у Ланцюговому магазині Steam також свідчить

про зростання інтересу до цього сегмента в Україні. Багато українських гравців активно беруть участь у мультиплеєрних іграх та взаємодіють з гравцями з інших країн, що сприяє культурному обміну та популяризації української ігрової спільноти. Загалом, мультиплеєрний сегмент ігрової індустрії в Україні має значний потенціал для подальшого зростання. Широкий спектр жанрів, талановиті розробники, активна геймінгова спільнота та підтримка держави створюють сприятливі умови для розвитку мультиплеєрних ігор в Україні.

Додаткові інвестиції в українську мультиплеєрну індустрію можуть стимулювати її зростання та розширення. Іноземні інвестори можуть вкладати кошти в українські розробницькі студії, що дозволить їм розробляти нові проекти, покращувати графіку та геймплей, а також залучати більше гравців.

Підтримка та співпраця з іншими країнами також можуть принести позитивні результати. Українські розробники можуть співпрацювати з міжнародними видавництвами та студіями для створення спільних проектів або ліцензування своїх ігор для ринків за межами України. Це дозволить привернути більше уваги до української ігрової індустрії та збільшити її присутність на світовій сцені.

Необхідно також звернути увагу на розвиток професійної освіти в галузі геймдеву. Запуск спеціалізованих курсів та програм навчання, спрямованих на розробку мультиплеєрних ігор, може допомогти підготувати талановитих фахівців, які зможуть працювати в українських та міжнародних розробницьких студіях.

1.2 Роль RobloxStudio в ігровій індустрії

Платформа Roblox та її інтегроване середовище розробки Roblox Studio відіграють важливу роль у розвитку ігрової індустрії. Ось кілька ключових аспектів, які підкреслюють їх значення:

Творча свобода: Roblox та Roblox Studio надають користувачам безліч можливостей для самовираження та творчості. Гравці можуть створювати власні ігри та світи, використовуючи інтуїтивний інтерфейс Roblox Studio. Це дозволяє їм реалізувати свої уявлення та створювати унікальний контент.

Навчання та розвиток: Roblox та Roblox Studio є потужними навчальними інструментами для молодих розробників. Вони можуть вивчати програмування,

дизайн, скриптування та інші технічні навички, працюючи з платформою. Це сприяє розвитку комп'ютерної грамотності та підготовці нового покоління ігрових розробників.

Спільнота та співпраця: Roblox має велику активну спільноту гравців та розробників. Це створює унікальну можливість для співпраці, обміну досвідом та взаємної підтримки. Розробники можуть отримувати зворотний зв'язок від гравців, вдосконалювати свої проекти та співпрацювати над спільними ігровими викликами.

Монетизація та підприємництво: Roblox надає можливість розробникам отримувати прибуток зі своїх ігор через внутрішню валюту Robux. Це дозволяє створювати підприємницькі можливості для розробників та мотивує їх продовжувати розви

Глобальний вплив: Roblox є масштабною глобальною платформою, яка привертає мільйони гравців з усього світу. Це надає розробникам можливість залучати аудиторію з різних країн та культур, що розширює їх потенційний ринок. Крім того, гравці можуть взаємодіяти з іншими гравцями з різних країн, спілкуватися та спільно виконувати завдання, що сприяє культурному обміну та розумінню.

Підтримка молодих талантів: Roblox надає можливість молодим розробникам отримати визнання та успіх у світі ігрової індустрії. Багато відомих ігор, які починалися на Roblox, отримали подальший успіх та були випущені на інших платформах. Це надихає молодих талантів та сприяє їхньому професійному розвитку.

Інноваційність та експерименти: Roblox дозволяє розробникам творити ігри у різних жанрах та форматах, що стимулює інновації та експерименти. Це дає можливість випробовувати нові ідеї, геймплейні механіки та визначати нові тренди в ігровій індустрії.

Узагалі, Roblox та Roblox Studio є важливими інструментами для розвитку ігрової індустрії, надаючи творчу свободу, навчання, спільноту та підприємницькі можливості для розробників, а також забезпечуючи незабутні враження для гравців з усього світу.

1.3 Огляд та аналіз існуючих аналогів top-down action ігор

Нашою метою є проведення аналізу аналогів у світі комп'ютерних ігор з фокусом на їх механіках та геймплеї. Цей аналіз допоможе визначити якість та ефективність геймплею в різних аналогах, а також виявити чинники, що впливають на задоволення гравців та їх взаємодію з ігрою.

Механіки геймплею

Механіки геймплею визначають спосіб, яким гравець взаємодіє з ігровим світом та які дії він може здійснювати. Під час аналізу аналогів, ми дослідимо різноманітність механік, таких як стрілянина, розв'язання головоломок, експлорація світу, взаємодія з персонажами тощо. Буде оцінено, наскільки ці механіки є цікавими, різноманітними та викликають відчуття задоволення у гравців.

Баланс геймплею

Баланс геймплею відіграє важливу роль у створенні гри, яка є викликаючою та справедливою для гравців. Аналізуючи аналоги, ми оцінимо, наскільки добре розбалансовані різні аспекти гри, такі як сила та властивості персонажів, економічна система, складність завдань тощо. Гармонійний баланс дозволяє гравцям відчувати справедливу виклик та уникнути нудоти або нерівності.

Взаємодія зі світом та персонажами[20, 1]

Успішні аналоги забезпечують гравцям взаємодію з живим та реалістичним світом, а також цікавими персонажами. Аналізуючи аналоги, ми оцінимо, наскільки глибоко іммерсивний світ та наскільки реалістично взаємодіють персонажі з гравцем.

Перед розробкою гри було проведено аналіз таких аналогів:

1. Foxhole - це одна з таких ігор, яка відрізняється від інших своїм спрямуванням на війну та реалістичністю. Гравці можуть брати участь у боях, будувати фортифікації та взаємодіяти з іншими гравцями на різних картах.

У Foxhole, гравці можуть взяти на себе роль військового командира в режимі реального часу, щоб керувати бойовими операціями своєї сторони (рис 1.1). Гра пропонує різні режими гри, включаючи бійки, наступи та оборону. Крім того, гравці можуть будувати фортифікації та інфраструктуру, таку як дороги та

залізниці, щоб забезпечити своїй стороні перевагу на полі бою.

Що робить Foxhole унікальною грою, це її фокус на стратегії та реалістичності військових дій. Гравці повинні не тільки брати участь у боях, але і керувати ресурсами та виробництвом зброї та інших матеріалів, щоб забезпечити успіх своїй стороні. Гра також пропонує реалістичну фізику та взаємодію з оточуючим середовищем, що додає грі ще більше глибини та реалізму.

Незважаючи на те, що гра може бути викликом для новачків, вона пропонує глибоку та задовільну геймплейну механіку для досвідчених гравців. Foxhole пропонує унікальний досвід гри, який залежить від співпраці та координації з іншими гравцями, що додає грі ще більше захоплення.

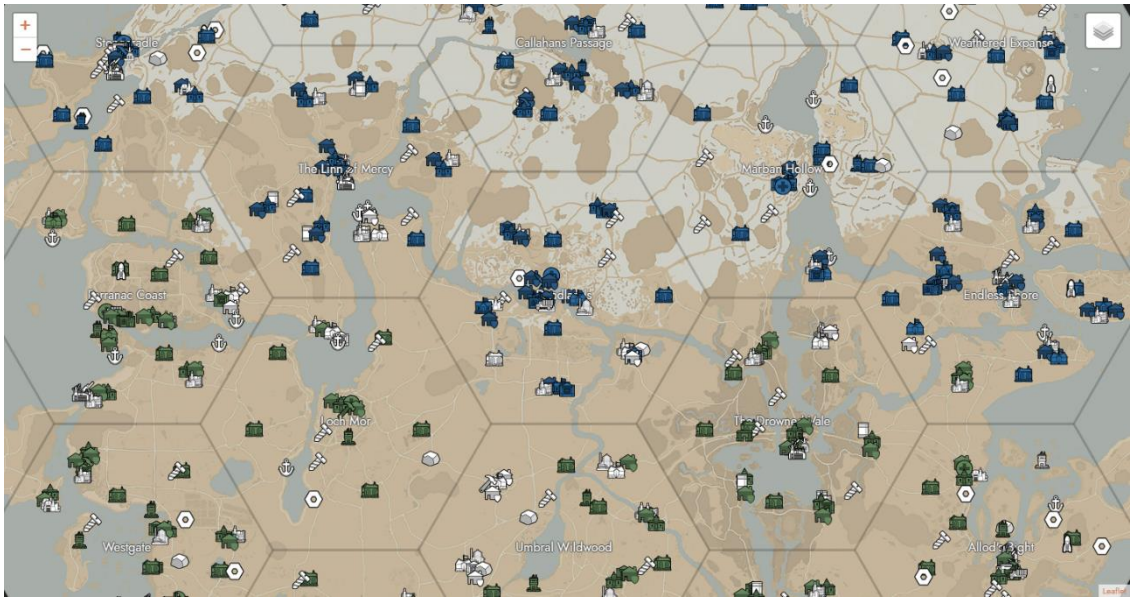

Рисунок 1.1 – демонстрація гейплею Foxhole

2. Hotline Miami - це 2D екшн-гра, розроблена Dennaton Games і видана Devolver Digital у 2012 році. Гра отримала широке визнання завдяки своєї незвичайній стилістиці, захоплюючому геймплею та унікальному звуковому оформленню.

Hotline Miami діє в альтернативній реальності 1980-х років в Маямі, США. Гравець приймає роль безіменного головного героя, який отримує таємні повідомлення по телефону і отримує завдання вбити певних цілей на різних рівнях. Гра пропонує насильницьку та криваву дію, використовуючи різноманітну зброю, таку як пістолети, ножі, кийки та інші предмети побутового використання.

Одним з найцікавіших аспектів Hotline Miami є система стрільби та бійки. Гра використовує підхід "один дотик - одне вбивство", що означає, що як головний герой, так і вороги можуть бути убиті одним ударом або пострілом. Це створює швидку та безкомпромісну гру, де реакція та точність гравця є вирішальними факторами. Щоб вижити, гравцю потрібно добре планувати свої дії, прогнозувати рухи ворогів та швидко реагувати на зміни ситуації.

Що стосується камери, гра відображається у вигляді 2D сверху, а гравець може бачити увесь рівень на екрані. Камера слідкує за героєм, коли він рухається по рівню, дозволяючи гравцю побачити поточну ситуацію та планувати свої наступні кроки.

Також важливим аспектом камери є її позиціонування щодо рівнів. Гра складається з декількох поверхів будівель, і камера може переміщатись вгору та вниз, дозволяючи гравцеві бачити всі рівність простору та взаємодіяти з ними. Це дає можливість розгорнутися в багатошаровому середовищі та використовувати різні стратегії для досягнення цілей.

Крім того, гра Hotline Miami має дуже стильний візуальний дизайн, який відтворює естетику 80-х років. Пастельні кольори, психоделічні ефекти та ретро-піксельна графіка створюють неповторну атмосферу. Камера відтворює цей стиль, пропонуючи динамічні переходи між різними ракурсами та використання різних кадрових рамок для підкреслення насильницької дії та загострення геймплею.

Узагалі, Hotline Miami - це гра, яка відома своєю непередбачуваністю, високою складністю та захоплюючою геймплейною механікою. Система стрільби та камера грають важливу роль у створенні напруженого та швидкого досвіду, що дає гравцеві можливість зануритися в жорстокий світ цієї альтернативної 1980-х реальності.

1.4 Постановка задачі

У відповідності до вищенаведеного для полегшення розуміння моделей розрахунків та оптимізації процесу розрахунку необхідно створити Web-застосунок, за допомогою якого можна було б обчислювати кількість вибраної посівної культури (пшениці, ячменю, вівса, кукурудзи, буряка) під ділянку поля

певної площі з можливістю додавання в розрахунок додаткових факторів (якість грунту, відстань між рядами, тощо), який оптимізує процес планування посівних робіт. Для цього необхідно вирішити такі задачі:

– провести аналіз переваг та недоліків уже існуючих сайтів-аналогів; - вибрати мову та технології програмування;

– розробити сучасний та зручний дизайн застосунку

– проаналізувати методики розрахунку посівних зернових культур;

– розробити максимально гнучкий калькулятор під різні види посівних культур;

– провести тестування продукту;

– розробити заходи з охорони праці.

Висновки до розділу 1

В розділі було проведено дослідження про посівні роботи та проаналізовано уже існуючі аналоги які проводять відповідні. Було проведено аналіз ринку, розгляд перспектив індустрії та її сильні сторони.

РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ САЙТУ ТА ЙОГО ФУНКЦІОНАЛУ

2.1 Написання Game Design Document

Game Design Document (GDD) - це документ, що описує концепцію і деталі гри, який використовується при розробці відеоігор. Це один з основних інструментів у процесі геймдизайну.

GDD містить інформацію про всі аспекти гри, включаючи геймплей, механіку, історію, персонажів, рівні, візуальний стиль, звуковий дизайн, інтерфейс та інше. Документ надає команді розробників ясне розуміння того, як гра повинна виглядати і працювати на різних етапах розробки.

Основні цілі GDD включають:

1. Опис концепції: GDD дозволяє описати загальну концепцію гри, включаючи її основні ідеї, жанр, цільову аудиторію та унікальні особливості. Він допомагає команді зрозуміти, що потрібно досягти з грою і які є основні принципи її створення.

2. Керівництво розробкою: GDD служить основним документом для команди розробників під час реалізації гри. Він містить детальні інструкції та специфікації щодо геймплею, систем, персонажів, механік і інших елементів гри. Це дозволяє забезпечити єдність в розумінні всіх членів команди та виконання задуманого.

3. Комунікація зі зацікавленими сторонами: GDD може бути використаний для комунікації з видавцями, інвесторами або іншими зацікавленими сторонами. Цей документ може допомогти пояснити концепцію гри, її потенціал та привернути увагу до проекту.

4. Планування ресурсів: GDD допомагає зробити оцінку обсягу роботи, потрібних ресурсів (людських, фінансових, технічних) та термінів розробки. Він може бути використаний для планування бюджету і визначення ресурсів, необхідних для успішної реалізації проекту.

Game Design Document є важливим інструментом для створення гри, він забезпечує структурованість і згуртованість у процесі розробки, сприяє ефективній комунікації між членами команди та допомагає забезпечити виконання бажаних

характеристик та якостей гри.

GDD:

Жанр гри: Мультиплеєрний шутер з видом зверху.

Опис гри: "TestProject" - це інтенсивна мультиплеєрна сесійна гра, в якій гравці змагаються у різних режимах проти інших гравців або спільно проти ворожих NPC. Гра має вид зверху, а поведінка камери схожа на "Hotline Miami", де швидкість і динаміка гри викликають почуття адреналіну та насиченості.

Ігрові режими: командний бій, кооперативна гра

Режим командного бою: Гравці об'єднуються в команди (наприклад, червона команда проти синьої команди).

Кожен гравець вибирає персонажа з різними унікальними навичками та зброєю.

Гра проходить на спеціальних аренах, що складаються з різних тематичних рівнів (місто, пустеля, фабрика тощо).

Мета гравців - знищити всіх ворогів і вижити якомога довше.

Гравці можуть використовувати обставини середовища, щоб здобути перевагу над ворогом.

Режим кооперативної гри:

Гравці об'єднуються в команду для спільної боротьби проти хвиль ворожих NPC.

Ворожі NPC атакують команду гравців хвилями з різною складністю та чисельністю.

Кожен гравець має можливість вибрати роль з різними спеціальними навичками (лікар, снайпер, штурмовик тощо).

Гравці повинні співпрацювати, використовувати тактику та обмінюватися ресурсами для виживання.

Бойова система:

Гравці можуть використовувати різні види зброї (пістолет, автомати, снайперські гвинтівки тощо), що дає їм можливість вести ефективну боротьбу проти ворогів.

Система будування:

У грі "TestProject" існує система будування, яка дозволяє гравцям створювати різні об'єкти для забезпечення переваги в бою.

Гравці можуть будувати різні типи загорож для захисту, такі як барикади, стіни або перешкоди, щоб перешкодити ворогам.

Крім того, гравці можуть будувати функціональні об'єкти, такі як важке озброєння, наприклад, ракетні системи залпового вогню (РСЗО), які можна використовувати для нанесення значних пошкоджень ворогам або знищення їх оборонних структур.

Для будівництва гравці збирають ресурси під час гри, наприклад, шляхом знищення ворожих NPC або захоплення спеціальних зон на карті.

Управління:

Управління грою використовує клавіатуру та мишу для точного керування персонажем.

Гравці можуть переміщатися по рівню, використовуючи клавіші стрибка, руху та обертання.

Використання різних збройних систем та спеціальних навичок доступне за допомогою клавіш швидкого доступу або комбінацій клавіш.

Прогресія та розвиток:

Гравці отримують досвідченість та рівні під час гри, що відкриває доступ до нової зброї, навичок та можливостей будування.

Вони також можуть отримувати винагороди та досягнення за досягнення конкретних цілей або виконання завдань.

Мультиплеєрний режим:

Гра пропонує мультиплеєрний режим, де гравці можуть співпрацювати або змагатися один з одним.

У режимі командного бою гравці об'єднуються в команди і змагаються проти інших команд.

У режимі кооперативної гри гравці об'єднуються, щоб спільно впоратися з хвилями ворожих NPC.

Через мультиплеєрний режим гравці можуть взаємодіяти між собою, спілкуватися та розробляти стратегії для досягнення перемоги.

Програвання та виграш:

Гравці отримують очки або нагороди за успішне завершення завдань або перемоги над ворогами.

За накопичені очки гравці можуть отримати доступ до нових персонажів, зброї або покращень для будівництва.

У разі смерті гравця він може повернутися до гри після певного періоду часу або в певній точці рівня, щоб продовжити боротьбу.

Графіка та атмосфера:

використовується візуальний стиль з видом зверху, схожий на "Hotline Miami", з яскравими кольорами та стилізованими деталями.

Звуковий супровід включає енергійну музику, звуки вибухів та вогню, що підкреслюють напруженість та екшен гри.

Платформи:

"TestProject" розробляється для геймплею на персональних комп'ютерах

2.2 Стек технологій

Luau - модифікована версія мови програмування Lua, створена компанією Roblox Corporation для використання в Roblox Studio. Вона призначена для поліпшення продуктивності, безпеки та функціональності Lua, зберігаючи при цьому зворотну сумісність.

Luau є мовою статичного типу, що означає, що типи змінних та виразів повинні бути оголошені перед використанням. Це допомагає запобігти помилкам і робить код більш зрозумілим та підтримуваним. Luau також має кілька нових функцій, які не доступні в Lua, таких як узагальнення, виведення типів та порівняння зразків.

Luau наразі знаходиться на стадії бета-тестування, але вже використовується великою кількістю розробників Roblox. Це потужна і універсальна мова, яка ідеально підходить для розробки ігор.

Ось деякі основні особливості Luau:

1. Статичне типізування
2. Узагальнення
3. Виведення типів

4. Порівняння зразків
5. Поліпшена продуктивність
6. Посилення безпеки
7. Нові функції

Luau є значним вдосконаленням порівняно з Lua і має всі шанси стати стандартною мовою для розробки ігор в Roblox.

Ось деякі переваги використання Luau:

Поліпшена продуктивність: Luau призначена для більш ефективної роботи порівняно з Lua, що може призвести до значних покращень продуктивності в іграх.

Посилення безпеки: Luau включає кілька нових функцій безпеки, які можуть допомогти захистити ігри від зловмисних атак[8,16].

Нові функції: Luau включає кілька нових функцій, які можуть зробити розробку ігор простішою і приємнішою

2.3 Модульна архітектура та її переваги

Модульна архітектура коду є важливим принципом розробки в геймдеві (гейм-розробці), який сприяє організації проекту та полегшує розширення, підтримку і тестування гри[4]. Вона базується на розділенні функціональності на незалежні модулі або компоненти, які можуть взаємодіяти між собою за допомогою чітко визначених інтерфейсів.

Основні переваги модульної архітектури в геймдеві:

– Розділення відповідальностей: Кожен модуль відповідає за конкретну частину функціональності гри, таку як графіка, фізика, штучний інтелект, управління ресурсами тощо[3]. Це дозволяє розробникам спеціалізуватись на конкретних областях і полегшує підтримку та розширення гри.

– Зручна розробка паралельно: Завдяки модульній архітектурі розробники можуть працювати над різними модулями незалежно один від одного. Це сприяє паралельній розробці і прискорює процес створення гри.

– Легке тестування: Модульна архітектура сприяє легкому тестуванню окремих модулів[3]. Це дозволяє швидше виявляти і виправляти помилки, а також забезпечує більшу стабільність гри в цілому.

– Повторне використання коду: Компоненти модульної архітектури можуть бути повторно використані в інших проектах або навіть в межах одного проекту. Це збільшує продуктивність розробки і сприяє швидкому створенню нових функцій і можливостей гри.

– Складність керування: Завдяки модульності керувати розробкою і підтримкою великих проектів стає більш простим. Кожен модуль може мати свою внутрішню логіку, тестування і підтримку, що полегшує командну роботу та організацію проекту в цілому.

2.4 Проектування компонентів гри

1. Управління

У відеоіграх існує кілька основних видів камер, які використовуються для створення різних ефектів та досвіду гравця. Особливо в мультиплеєрних іграх, шутерах та іграх з видом від третьої особи, вибір камери відіграє важливу роль у створенні належного геймплею та іммерсивного досвіду.

Перший основний тип камери - це перша особа або "вид від першої особи". У цьому типі камери гравець бачить світ гри очима головного героя. Цей погляд надає найбільшу іммерсію, оскільки гравець відчуває себе "всередині" гри. Цей тип камери часто використовується у шутерах від першої особи (FPS), таких як "Call of Duty" або "Battlefield".

Другий тип камери - це вид від третьої особи або "вид ззовні". У цьому типі камери гравець бачить свого персонажа ззовні, зі змінними кутами огляду. Цей погляд надає кращий огляд навколишнього середовища та дозволяє гравцю бачити свого персонажа. Багато мультиплеєрних ігор, таких як "Fortnite" або "PlayerUnknown's Battlegrounds" (PUBG), використовують цей тип камери.

Третій тип камери - це "камера спостерігача" або "камера глядача". Цей тип камери використовується в мультиплеєрних іграх для спостереження за грою без прив'язки до певного персонажа. Камера може переміщатися навколо різних областей гри, слідкувати за діями гравців або стежити за епічними подіями. Камера спостерігача зазвичай використовується у відеотрансляціях або стрімінгових

турнірах, де коментатори або глядачі можуть бачити гру з різних кутів і отримувати більш повну картину подій.

Існує також гібридний тип камери, який поєднує першу і третю особу. Наприклад, у деяких шутерах від третьої особи, як "Gears of War" чи "Tom Clancy's The Division", гравець бачить свого персонажа зі спини, але може перемикатися до першої особи, коли це необхідно для більшої точності при прицілюванні.

Крім основних типів камер, в іграх можуть бути й інші варіації або спеціальні ефекти. Наприклад, камера "булейтайм" може сповільнювати час і перехоплювати динаміку подій, щоб гравець міг легше виконувати складні рухи або стріляти ворогів. Камера з "фіксованим ракурсом" може бути затверджена у певній позиції, наприклад, надворі або зібрана на определённой позиції надворі, для створення артістичного ефекту або для показу великих областей гри [15].

Вибір камери відіграє важливу роль у геймплеї і впливає на взаємодію гравця з грою. Розробники ігор ретельно планують та налаштовують камери, щоб забезпечити оптимальний досвід гравця та забезпечити баланс між іммерсією, комфортом та функціональністю гри.

Ось кілька основних способів реалізації камери від третього лиця в іграх:

Фіксована камера: У цьому підході камера розташована на фіксованій позиції відносно персонажа гравця. Камера може бути розташована ззаду або зверху і дивитися на персонажа під кутом. Цей підхід надає стабільний ракурс та дозволяє гравцю бачити свого персонажа та його оточення. Приклади цього підходу можна побачити в іграх, таких як "Tomb Raider" або "Assassin's Creed".

Автоматично регульована камера: В цьому випадку камера розташована ззаду або збоку від персонажа, але її положення автоматично адаптується до дій гравця. Наприклад, коли гравець рухається вперед, камера відступає назад, щоб забезпечити кращий огляд. Цей підхід дозволяє гравцю зосередитися на грі, а не на керуванні камерою. "The Legend of Zelda: Breath of the Wild" є прикладом гри з автоматично регульованою камерою.

Ручне керування камерою: В деяких іграх гравець може самостійно керувати камерою для досягнення потрібного ракурсу. Гравець може обертати, нахиляти або збільшувати/зменшувати зум камери за допомогою контролера або

миші/клавіатури. Цей підхід надає гравцеві більшу свободу вибору огляду та дозволяє налаштувати камеру під власні потреби. Наприклад, у грі "

Камера зверху, також відома як камера з пташиного польоту або камера зверху вниз, є одним з типів камери від третього лиця. У цьому типі камери перспектива виглядає зверху, з вищої точки над персонажем або сценою гри[14].

Камера зверху надає гравцеві широкий огляд навколишнього середовища та позиції персонажа. Деякі переваги цього типу камери включають:

Стратегічне планування: Камера зверху надає гравцеві можливість краще бачити весь ігровий світ і розуміти тактичні аспекти гри. Це особливо корисно у стратегічних іграх, де гравець повинен приймати вирішальні рішення, спираючись на широкий огляд ситуації.

Огляд областей: Завдяки камері зверху гравець може легко бачити великі області гри, такі як мапи, рівні або поля бою. Це дозволяє зручно навігувати по середовищу та зорієнтуватися у просторі.

Відстеження подій: Камера зверху може також слугувати для відстеження дій гравця і надавати йому сповіщення про ворогів, предмети або інші важливі події, які відбуваються у грі.

2. Мультирежимність та її переваги

Мультиплеєрні, сесійні ігри зазвичай мають систему різних режимів гри, що дозволяє гравцям взаємодіяти один з одним у різних способах і в різних сценаріях. Це особливо поширено у мультиплеєрних шутерах, які спрямовані на багатокористувацькі битви і змагання між гравцями.

Один із найпоширеніших режимів гри у мультиплеєрних шутерах - це "Deathmatch" або "Вільна битва". У цьому режимі гравці змагаються один з одним, намагаючись набрати якомога більше вбивств. Зазвичай перемагає той, хто досягне певного кількості вбивств або набере найбільшу кількість очок за обмежений час. Цей режим дозволяє гравцям випробувати свої навички у прямому протистоянні один з одним і підвищує динаміку гри.

Окрім цього, існують режими, де гравці можуть взаємодіяти як команди, так і протистояти один одному. Наприклад, "Team Deathmatch" або "Командна вільна битва" дозволяє гравцям битися у складі команд і перемагати за рахунок набраної

кількості вбивств команди в цілому. Цей режим сприяє співпраці, координації та стратегічному плануванню між членами команди.

Можна зазначити наступні переваги мультирежимного підходу до мультиплеєрних ігор:

– Різноманітність геймплею: Система різних режимів гри дозволяє гравцям насолоджуватися різними варіантами геймплею і різними типами викликів. Кожен режим має свої особливості, правила та цілі, що додає різноманітності та інтриги до гри.

– Соціальний аспект: Мультиплеєрні, сесійні ігри з різними режимами гри створюють можливості для взаємодії та спілкування між гравцями. Вони можуть об'єднувати гравців у команди або протистояти один одному, сприяючи формуванню спільнот і соціальних зв'язків між ними.

– Більше можливостей для розвитку навичок: Різні режими гри дозволяють гравцям вдосконалювати різні навички та стратегії, можна розвивати командну гру та координацію з командою.

– Підтримка довготривалих ігрових сесій: Система різних режимів гри додає глибину та тривалість до мультиплеєрного досвіду. Гравці можуть переключатися між різними режимами, що дозволяє зберегти інтерес до гри протягом тривалого часу і надихати їх на подальшу гру.

– Конкуренція та виклик: Онлайн мультирежими нерідко пропонують змагальні аспекти, такі як ліги, рейтинги та турніри. Гравці можуть випробовувати свої навички, змагатися з іншими гравцями та підніматися по лідерським таблицям. Це стимулює бажання постійно вдосконалюватися та досягати нових висот[9].

Постійні оновлення та додатковий контент: Розробники мультирежимних ігор зазвичай надають постійну підтримку гри шляхом випуску оновлень, нових режимів, карт або персонажів. Це дозволяє гравцям отримувати нові враження, тримати ігру свіжою та продовжувати грати на протязі тривалого часу.

3. Аналіз та проектування складової графічного інтерфейсу

UI є ключовим елементом взаємодії користувача з програмними продуктами, зокрема, відеоіграми. Він включає в себе набір графічних та текстових елементів, що дозволяють користувачеві взаємодіяти з грою і контролювати її події. Ігровий

інтерфейс вимагає особливої уваги, оскільки впливає на сприйняття гри, її геймплей і загальний досвід користувача.

Одним з основних переваг ігрового інтерфейсу є зручність і простота використання. Графічні елементи та контрольні панелі мають бути зрозумілими та інтуїтивно зрозумілими, щоб гравець міг швидко орієнтуватися в грі. Чіткість та простота використання є ключовими факторами, оскільки вони допомагають гравцеві швидко прийняти рішення та реагувати на події в грі.

Інший важливий аспект ігрового інтерфейсу - налаштування. Гра повинна надавати можливість гравцеві налаштовувати інтерфейс залежно від його потреб і вподобань. Можливість зміни розташування елементів, налаштування розміру шрифтів, зміна кольорів та інші параметридозволяють гравцям налаштувати інтерфейс таким чином, щоб він відповідав їхнім індивідуальним потребам. Наприклад, деякі гравці можуть віддавати перевагу компактному інтерфейсу з мінімальною кількістю елементів, щоб мати більше місця на екрані для спостереження за грою, тоді як інші можуть бажати більш деталізованого інтерфейсу з багатьма інформаційними панелями.

Ще однією важливою перевагою ігрового інтерфейсу є його адаптабельність до різних жанрових відеоігор. Кожен жанр має свої особливості та вимоги до інтерфейсу. Наприклад, в екшн-або стрілялках, де швидка реакція та точність грають важливу роль, гравцеві може бути потрібен хотбар, який дозволяє швидко вибирати зброю або заклинання. У рольових іграх (RPG) інтерфейс може містити інвентар, характеристики персонажа та інші елементи для керування і прокачування персонажа. Стратегічні ігри можуть мати складний інтерфейс з картою, меню будівництва, інформаційними вікнами та іншими елементами, що дозволяють гравцю керувати своїми військами та ресурсами.

Незважаючи на переваги, ігровий інтерфейс також має свої недоліки. Один з найбільш поширених недоліків полягає в його перенасиченості та завантаженості. Якщо на екрані з'являється занадто багато елементів інтерфейсу, він може стати візуально неприємним та заважати гравцю бачити гру. Крім того, недостатні оптимізації інтерфейсу може призвести до зайвих кроків або дій для досягнення певних функцій. Надмірність інтерфейсу може призвести до перевантаження гравця і порушити іммерсивний досвід гри.

Інший недолік ігрового інтерфейсу полягає в його складнощах для новачків. Деякі ігри мають складні та заплутані інтерфейси, що може стати перешкодою для нових гравців, які можуть почуватися втраченими або не розуміти, як користуватися різними функціями. Це може відлякувати нових користувачів від гри та погіршити їх перший враження[20, 11, 2].

Окрім того, ігровий інтерфейс може бути проблематичним для гравців з фізичними обмеженнями. Недостатня роздільна здатність, неправильна розмітка елементів або недостатня можливість налаштування можуть створювати перешкоди для гравців з обмеженими можливостями, ускладнюючи їм гру та знижуючи загальний комфорт користування.

Правильно розроблений ігровий інтерфейс повинен забезпечувати збалансовану кількість функцій та інформації, не перенавантажуючи гравця [12,3]. Крім того, надаючи можливість налаштування інтерфейсу, розробники можуть забезпечити індивідуалізацію ігрового досвіду для різних типів гравців і їхніх вподобань. Однак, розробка ігрового інтерфейсу вимагає уваги до деталей, щоб уникнути перенасиченості та складнощів для користувачів.

Наприклад, ігровий інтерфейс має включати основні елементи, що є необхідними для багатьох жанрових відеоігор. Один з найпоширеніших елементів - хотбар. Хотбар - це рядок або панель, на якій розташовані скорочення до основних інструментів, предметів або дій, які гравець може використовувати під час гри. Він забезпечує швидкий доступ до необхідних елементів та дозволяє гравцю швидко реагувати на змінні ситуації [17,18].

Інший важливий елемент - інвентар. Інвентар представляє собою список або сітку, в якому зберігаються предмети та ресурси, які гравець зібрав під час гри. Це дозволяє гравцю керувати своїми ресурсами, обмінюватися або використовувати їх у відповідності до потреб гри [6]. Наявність інвентаря особливо важлива в RPG-іграх та іграх про виживання, де збирання та управління ресурсами є важливою складовою геймплею.

Зв'яжмо все разом і підкреслимо ключові аспекти ігрового інтерфейсу у мультиплеєрних іграх. В цьому контексті, ігровий інтерфейс відіграє критичну роль у забезпеченні комунікації, співпраці та змагання між гравцями у режимі реального часу.

Одним з найважливіших аспектів є комунікаційна панель або чат. Це місце, де гравці можуть обмінюватися повідомленнями, стратегічними планами або просто спілкуватися. Чат може включати різні канали, такі як приватні повідомлення, командні чати або глобальний чат для всіх гравців. Добре розроблений чат дозволяє гравцям ефективно спілкуватися з командою, координувати дії та обмінюватися важливою інформацією під час гри.

Другим аспектом є відображення інформації про інших гравців. Це можуть бути ніки, статуси, відображення життєвої енергії або показники прогресу. Ця інформація допомагає гравцям визначати союзників та противників, відстежувати їхні дії та приймати стратегічні рішення під час гри. Наприклад, відображення життєвої енергії може дати гравцям контекст про потенційно слабких або сильних гравців, допомагаючи вирішувати, на кого зосередитися в бою.

Третім важливим аспектом є система оцінки або рейтингу гравців. Це може бути показник, який відображає рівень навичок або досвіду гравця[7]. Така система допомагає вирівнювати гравців за рівнем вмінь та створює більш рівноправні та збалансовані ігрові сесії. Рейтингова система може спонукати гравців до зростання своїх навичок, стимулювати здорове змагання та підвищувати ігрову мотивацію[7]. Крім того, вона може допомогти створити рівномірні команди або матчмейкінгові системи, що забезпечують адекватні співставлення гравців за їхнім рівнем навичок.

Ще одним важливим аспектом ігрового інтерфейсу в мультиплеєрних іграх є система команд та співпраці[2]. Це може включати механізми для створення команд або груп, можливість об'єднатися з іншими гравцями для досягнення спільних цілей та співпрацювати в ігровому світі. Інтерфейс повинен надати зручні та інтуїтивно зрозумілі інструменти для організації командної гри, спілкування та координації дій між гравцями.

Необхідним елементом ігрового інтерфейсу в мультиплеєрних іграх є також система оповіщень та повідомлень[9]. Це можуть бути сповіщення про важливі події в грі, запити на виконання дій, повідомлення про досягнення чи виклики від інших гравців. Інтерфейс повинен забезпечити чітку та ефективну комунікацію з гравцями, дозволяючи їм своєчасно реагувати на події та спілкуватися з іншими гравцями.

Остаточно, ігровий інтерфейс у мультиплеєрних іграх має бути легким у використанні, інтуїтивним та ефективним. Він повинен допомагати гравцям максимально концентруватися на грі, забезпечуючи швидкий доступ до важливої інформації та інтуїтивне управління. Це особливо важливо у швидкотемпових мультиплеєрних іграх, де кожна секунда може мати значення.

Для різних жанрових відеоігор, ігровий інтерфейс може мати свої особливості. Наприклад, у шутерах з першої особи можуть бути важливими елементами графічний приціл, відображення боєприпасів та екран смерті. У стратегічних іграх ігровий інтерфейс може включати карту, списки одиниць та діалогові вікна для управління базою або ресурсами.

Також варто згадати про адаптивність ігрового інтерфейсу у мультиплеєрних іграх. Оскільки гравці можуть грати на різних пристроях з різними розмірами екранів, важливо, щоб ігровий інтерфейс був адаптований до різних роздільність екранів та пропонував оптимальну візуалізацію для кожного пристрою.

Зважаючи на всі ці фактори, розробники мультиплеєрних ігор повинні ретельно планувати та працювати над ігровим інтерфейсом, щоб забезпечити гармонійну ігрову досвід для гравців. Добре збалансований та функціональний ігровий інтерфейс може зробити гру більш захоплюючою, сприяти комунікації та співпраці між гравцями, а також підвищити задоволення від взаємодії з віртуальним світом.

2.5 Огляд сторонніх бібліотек та модулів

1. FastCast

FastCast є модулем, написаним на мові програмування Luau, призначеним для емуляції фізики кульок та виявлення попадань у чистому Lua. Він не використовує фізичного реплікування.

Швидкість: FastCast розроблено з метою досягнення максимальної швидкості. Він використовує ряд технік для покращення продуктивності, таких як кешування та оптимізація променевого зондування.

Точність: FastCast розроблено з метою досягнення максимальної точності. Він використовує ряд технік для покращення точності, таких як використання бібліотеки променевого зондування високої точності.

Простота використання: FastCast розроблено з метою надання легкості в використанні. Його API простий та зрозумілий, і є кілька прикладів, що допоможуть вам почати.

Основні функції:

Фізика кульок: FastCast може бути використаний для моделювання фізики кульок, включаючи гравітацію, опір повітря та зіткнення.

Виявлення попадань: FastCast може бути використаний для виявлення попадання кульок в інші об'єкти. Це може бути використано для створення таких речей, як зброя із псевдографіком та зброя на основі снарядів.

Кешування: FastCast використовує кешування для покращення продуктивності. Це означає, що траєкторію кулі потрібно обчислити лише один раз, навіть якщо вона випускається декілька разів.

Оптимізації: променевого зондування: FastCast використовує кілька оптимізацій променевого зондування для покращення продуктивності. Це включає використання бібліотеки променевого зондування високої точності та кешування результатів променевих зондувань.

Загалом, FastCast є потужним і універсальним модулем для емуляції фізики кульок та виявлення попадань у Roblox. Він є швидким, точним та легким у використанні.

Основні дизайн-рішення, що були прийняті при створенні FastCast:

Продуктивність: FastCast був розроблений з метою досягнення максимальної швидкості. Це було досягнуто за допомогою ряду технік, таких як кешування та оптимізації променевого зондування.

Точність: FastCast був розроблений з метою досягнення максимальної точності. Це було досягнуто за допомогою ряду технік, таких як використання бібліотеки променевого зондування високої точності.

Простота використання: FastCast був розроблений з метою надання легкості в використанні. Його API простий та зрозумілий, і є кілька прикладів, що допоможуть вам почати.

2. Fusion

Fusion є сучасною бібліотекою реактивного користувацького інтерфейсу, створеною спеціально для Roblox і Luau. Вона надає декларативний синтаксис для побудови користувацьких інтерфейсів, що полегшує створення складних та відгукливих інтерфейсів. Fusion також має систему керування станом, яка дозволяє відстежувати дані у вашому додатку, а також систему компонентів, що спрощує повторне використання коду.

Ось деякі переваги використання Fusion:

Простота використання: Fusion призначений для легкості навчання та використання, навіть для початківців. Декларативний синтаксис дозволяє писати код, який є зрозумілим і лаконічним.

Якість: Fusion надає потужний набір функцій для побудови користувацьких інтерфейсів, включаючи підтримку анімацій, переходів та макетів.

Гнучкість: Fusion є дуже гнучкою бібліотекою, яку можна використовувати для побудови різноманітних користувацьких інтерфейсів. Вона не обмежена жодним конкретним стилем або дизайном.

Продуктивність: Fusion розроблено з урахуванням продуктивності, навіть для великих та складних користувацьких інтерфейсів.

Особливості Fusion:

Декларативний синтаксис: Fusion використовує декларативний синтаксис для побудови користувацьких інтерфейсів. Це означає, що ви можете описати бажану поведінку вашого інтерфейсу, не заморочуючись деталями його реалізації.

Керування станом: Fusion надає систему керування станом, яка дозволяє відстежувати дані у вашому додатку. Ці дані можуть використовуватися для керування поведінкою вашого інтерфейсу.

Система компонентів: Fusion надає систему компонентів, що спрощує повторне використання коду. Компоненти є невеликими, самодостатніми частинами коду, які можна використовувати для побудови складних користувацьких інтерфейсів.

Основні дизайн-рішення, прийняті при створенні Fusion:

Простота використання: Fusion був розроблений для простоти навчання та використання, навіть для початківців. Декларативний синтаксис та система

керування станом призначені для спрощення написання зрозумілого та лаконічного коду, який надалі буде зручно пере використовувати.

Продуктивність: Fusion був розроблений з метою ефективності, навіть для великих та складних користувацьких інтерфейсів. Fusion використовує ряд технік для покращення продуктивності, таких як кешування та лінива завантаження.

Гнучкість: Fusion був розроблений з метою гнучкості для побудови різноманітних користувацьких інтерфейсів. Він не обмежений жодним конкретним стилем або дизайном[19].

Висновки до розділу 2

Результатом розділу був огляд інструментів, аналіз компонентів та написаний Game Design Document

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ФУНКЦІОНАЛУ ТА МЕХАНІК ГРИ

3.1 Розробка серверної частини

Серверна частина архітектури складається з головного серверного файлу. Основну частину Main файлу складає модуль який відповідає за запуск Event-ної системи. Event або “подійна” система відповідає за реагування системи на зовнішні події, через вбудовані екземпляри RobloxAPI – RemoteEvent та RemoteFunction.

Модуль відповідаючий за формування цієї системи – Builder. Це підфайл Main скрипта, який імпортується в та викликається в ньому. В ньому є єдиний метод – Build, який відповідає за піднімання всієї системи. Механізм реалізований через прохід по директорії модулів, які предназначені для обробки подій. Будь якій файл верхнього рівня повинен мати метод Listen, в якому підключається обробник подій.

Всі EventHandler-и створені для тонкої прослойки, яка практично не виконує логіки, окрім підключення RemoteEvent та RemoteFunction до внутрішній методів, які для обробки даних використовують підподулі.

Build метод:

```
for _, eventHandlerModule in pairs(dir:GetChildren()) do
    if eventHandlerModule:IsA("ModuleScript") then
        local handler = require(eventHandlerModule)::EventHandler
        if not handler.Listen or typeof(handler.Listen) ~= 'function' then
            warn("Incorrectly created event handler: " .. tostring(eventHandlerModule))
        else
            handler.Listen()
        end
    end
end
```

3.2 Розробка UI

Архітектура UI частини гри влаштована таким чином:

Центральним файлом для запуску являється файл "UI". UI файл - це екземпляр "LocalScript", і він являється точкою входу в збір всіх компонентів

графічного інтерфейсу. Проте файл UI виступає в ролі деякої вхідної точки, і практично не виконує ніякої логіки, підтягуючи тільки модуль "Builder", викликає його метод Build, який в якості єдиного аргументу приймає директорію із компонентами

Builder - Центральний модуль збору графічного інтерфейсу гри. Основний метод даного модулю це метод Build(). Даний метод проходиться по всій детекторі переданій в його аргумент, та робить валідацію модулів які входять в елементи-нащадки цієї директорії. Після того як файл пройшов валідацію, і ми впевнились що даний файл (перевіряємий файл на даній ітерації) являється компонентом, ми захоплюємо(імпортуємо) цей файл (використовючи функцію require), та перевіряємо чи являється імпортуємий об’єкт типу "function", адже архітектура розрахована на те що кожен компонент буде являти собою функціональний елемент. Після того як ми получили підтвердження що елемент являється функціональним ми запускаємо його на виконання.

Кожен компонент UI це модуль, який повертає деяку функцію, з нульовою кількістю вхідних параметрів. Ця функція і є компонентом програми, а її виклик - запуск компонента.

Система складається із одного центрального файлу який збирає та запускає весь графічний інтерфейс. Здебільшого UI збудований так щоб вішати функціонал на всі вже створенні та впорядковані GUI меню.

Вся система знаходиться по шляху: StarterPlayer -> StarterPlayerScripts.

Там лежить папка з іменем UI. Саме тут лежить вся кодова база по контролю графічними елементами.

Файлом який запускає в дію систему є "UI". Вся частина UI складається із будівельних блоків, якими в даному випадку представленні "Компонентами". Компонент - це функціональна та цільна частина графічного інтерфейсу. В нашому випадку компонент скоріше виступає "обгорткою" над графічним елементом, ніж представляє його. Така структура зв’язана з особливостями бібліотеки Fusion.

Структура директорії(рис 1.2)

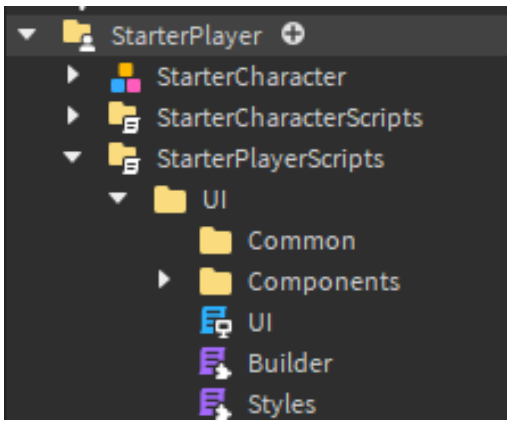

Рисунок 1.2 – структура UI інтерфейсу

3.3 Реалізація системи будування

Головними типами в цій системі є BaseBuilding, ActiveBuilding та InteractiveBuilding. BaseBuilding представляє базову будівлю і має методи, такі як OnBuild та OnDestroy, що виконуються під час побудови та знищення будівлі відповідно. Він також надає методи для отримання моделі будівлі, власника, назви та типу будівлі.

ActiveBuilding є розширенням BaseBuilding і додає метод OnUpdate, який виконується під час оновлення активної будівлі. Цей тип будівлі може мати додаткову функціональність, яка активується під час оновлення.

InteractiveBuilding є розширенням ActiveBuilding і додає поле Actions, яке представляє набір можливих дій, які можна виконати з інтерактивною будівлею. Також є метод CharacterSpawned, який викликається під час появи персонажа біля будівлі.

Інші типи, такі як ViewModelHandler, RotationModule та ViewModel, надають додаткову функціональність для роботи з моделями будівель. Наприклад, ViewModelHandler дозволяє управляти видимістю моделі, RotationModule забезпечує функціонал повороту моделі, а ViewModel надає методи для отримання інформації про модель та її управління.

Також в системі присутні типи BuildingController і BuildingViewModelConfiguration, які відповідають за управління будівлями і конфігурацію моделей будівель відповідно.

Ці типи дозволяють розширити функціональність будівель та керувати ними

у грі на Roblox Studio. Вони надають абстрактні способи опису та роботи з різними типами будівель і їх взаємодії з гравцями.

Система потребує постійної реплікації, тож її архітектура була розділена на 3 основні модулі: серверний, клієнтський, та загальний (призначений для використання як на клієнті так і на сервері).

1. Клієнт представлений головним файлом – BuildModule.

На клієнтській частині скрипт BuildModule в загальному призначений для прив'язки подій, та передачі їх на обробники. Цей скрипт використовує 2 модульних файли – BuildBinder та BuildManager, і його задача полягає в тому щоб після того коли із зовні буде відправлена подія про те що гравець хоче побудувати якийсь конкретний об'єкт, підключити в дію цих два модульних скрипти.

BuldBinder відповідає за те щоб прикріпити за конкретною подією вводу користувача якусь дію, яка повинна реагувати якраз на цей самий ввід. За саму ж реалізацію поведінки системи, після деякого користувацького вводу відповідає BuildManager. В BuldBinder можна виділити такі методи:

- BindBuild
- BindCancel
- BindRotate
- UnbindAll

Важливою складовою системи будування на клієнті являється об'єкт ViewModel. Його основне призначення в тому щоб дати користувачеві візуальне представлення того як буде виглядати об'єкт який він вибрав для будівництва, та надати контроль для коректного розміщення самої будівлі. Тобто цей об'єкт – це візуальний індикатор для гравця. Він може бути налаштований для різних пристроїв, адже приймає різні обробники користувацького вводу. Проте основним обробником виставленим по стандарту являється MouseHandler який призначений для обробки вводу з миші, для гравців які очевидно використовують PC.

ViewModel – це концепт моделі пре перегляду, і на його функціонал полягає в візуальний індикатор доступності розміщення об'єкту в даній області. В нашому випадку реалізована через зміну кольору самої моделі об'єкту, що являється досить стандартним прийомом в інших проектах. Для інтуїтивності було вибрано 2 кольори ідентифікації: червоний та зелений. Причина в тому що ці 2 кольори несуть

інтуїтивну візуальну інформації яку потрібно донести до користувача, тож не треба буде додатково навантажувати нашого користувача роз’ясненнями геймплейних особливостей.

2. Розглянемо основні об’єкти задіяні в роботі сервера:

BuildingManager:

handlersDir : Шлях до папки Buildings , де зберігаються обробники будівель.

Types : Змінна, яка містить модуль Types з розширеними типами даних для системи будівель.

buildDir: Змінна, що містить шлях до папки Build , де знаходяться модулі із функціями для будівництва.

buildEvent: Змінна, яка містить подію Build з розширеними типами даних.

CanBuild: Змінна, яка містить модуль CanBuild з функціями для перевірки можливості будівництва.

BuildingInfoList: Змінна, яка містить список інформації про будівлі.

HandlerManagerFactory: Змінна, яка містить фабрику HandlerManagerFactory для створення обробників будівель.

ModelContainer : Контейнер, в якому будуть розміщені створені моделі будівель, для котролю групою будівель.

BuildingControllerRegister : Об'єкт, що реєструє контролери будівель

BuildingFactories: Об'єкт, що містить фабрики будівель.

module.Build(owe: Player, buidlingId: string, cf: CFrame): Цей метод відповідає за будівництво будівлі. Він отримує параметри: ower (гравець, який будує), buidlingId (ідентифікатор будівлі) і cf (позиція та орієнтація будівлі у просторі).

У цьому методі спочатку перевіряється, чи існує інформація про будівлю з вказаним buildingId . Якщо немає, генерується помилка.

Далі створюється копія моделі будівлі (buildingInfo.OriginModel:Clone()) і розміщується у ModelContainer . Метод PivotTo використовується для переміщення моделі у вказану позицію і орієнтацію.

Після цього виконується перевірка можливості будівництва за допомогою функції CanBuild . Якщо функція повертає false , модель будівлі знищується, а генерується попередження.

На останок створюється екземпляр buildingFactory з використанням

module.BuildingFactories.Get(buidlingId) , і цей об'єкт реєструється для гравця з використанням module.BuildingControllerRegister.RegisterFor(ower, buildingFactory.New(ower, model)) .

Цей модуль на сервері відповідає за обробку будівництва будівель, перевірку можливості будівництва та реєстрацію контролерів для будівель гравців.

controllersDir : Шлях до папки Controllers , де зберігаються контролери будівель.

RS: Змінна, що містить посилання на ReplicatedStorage, сервіс Roblox, який використовується для обміну даними між сервером і клієнтом.

BUIDLING_TYPES : Змінна, яка містить модуль BUILDING_TYPES з типами будівель.

module.GetControllerChainIndex(building: Types.BaseBuilding): Цей метод повертає індекс контролера в ланцюжку пріоритетів для вказаної будівлі. Він отримує параметр building , який є об'єктом будівлі типу Types.BaseBuilding. Метод шукає тип будівлі в PriorityChain і повертає відповідний індекс. Якщо тип не знайдено, генерується помилка.

module.RegisterFor(owner: Player, building: Types.BaseBuilding): Цей метод реєструє будівлю для власника (гравця). Він отримує параметри owner (гравець, якому належить будівля) і building (об'єкт будівлі типу Types.BaseBuilding). Метод отримує індекс контролера за допомогою module.GetControllerChainIndex(building) і реєструє будівлю для всіх контролерів з PriorityChain з індексом меншим або рівним отриманому індексу.

module.SetupAll(): Цей метод викликає метод Setup() для всіх контролерів будівель. Він використовується для налаштування всіх контролерів перед використанням.

Цей модуль на сервері відповідає за реєстрацію контролерів будівель, визначення пріоритетів контролерів та налаштування цих контролерів. Контролери використовуються для управління різними аспектами будівель, залежно від їх типу.

– BuildEventHandler:

buildDir: Змінна, що містить шлях до папки Build, де знаходяться модулі для будівництва.

buildEvent: Змінна, яка містить подію Build з розширеними типами даних.

BuildingManager: Змінна, яка містить об’єкт BuildingManager для керування

системи будівництва.

module.OnBuild(plr: Player, buildingID: string, cf: CFrame): Цей метод є обробником події будівництва (Build). Він отримує параметри plr (гравець, який будує), buildingID (ідентифікатор будівлі) і cf (позиція та орієнтація будівлі у просторі).

У цьому методі перевіряється, чи передані коректні параметри plr , buildingID і cf . Якщо будь-який з параметрів відсутній, генерується помилка "Not valid params".

Потім викликається метод BuildingManager.Build(plr, buildingID, cf), який відповідає за фактичне будівництво будівлі з використанням BuildingManager .

На останок, метод повертає true , що показує успішне виконання функції.

module.Listen(): Цей метод встановлює прослуховування події buildEvent на сервері. Він призначає функцію module.OnBuild як обробник події buildEvent.OnServerInvoke , що означає, що коли клієнт викличе подію buildEvent на сервері, функція module.OnBuild буде викликана для обробки цієї події.

Цей модуль на сервері служить для обробки події будівництва, передачі параметрів в BuildingManager для фактичного будівництва будівлі та встановлення прослуховування події для виклику будівництва з клієнта.

– ActiveController:

Цей код використовується для створення системи будівництва в грі на платформі Roblox з використанням Roblox Studio. Головна мета цієї системи - активувати і оновлювати будівлі типу "ActiveBuilding" на кожному кадрі гри.

Основний принцип роботи коду полягає в налаштуванні головного циклу гри (головного циклу оновлення) і виклику методу "OnUpdate()" для кожної будівлі, що знаходиться в списку будівель.

Коли викликається функція "module.Setup()", вона налаштовує систему будівництва. Ця функція додає завдання "main" до головного циклу гри за допомогою функції "BaseGameLoop.AddTask()". Завдання "main" представляє собою цикл, який проходить через кожну будівлю, що знаходиться в списку будівель, і викликає для неї метод "OnUpdate()".

Метод "OnUpdate()" виконує оновлення будівлі, включаючи всі необхідні обчислення, перевірки та зміни стану будівлі. Цей метод виконується на кожному кадрі гри, щоб забезпечити актуальність і коректну роботу будівлі.

Після додавання завдання main до головного циклу гри, функція "module.Setup()" повертає результат асинхронного запуску головного циклу гри за допомогою функції "BaseGameLoop.AsyncStart()". Це дозволяє грі розпочати оновлення будівлей і виконувати їх на кожному кадрі.

Загалом, цей код створює систему будівництва, де будівлі типу "ActiveBuilding" оновлюються на кожному кадрі гри, щоб забезпечити їхню коректну роботу та відображення. Він додає цей функціонал до головного циклу гри, щоб гарантувати, що будівлі оновлюються відповідно до вимог гри та взаємодіють з іншими компонентами гри належним чином.

Модуль "BaseGameLoop" виконує управління головним циклом гри. Він надає можливість додавати завдання до циклу, асинхронно запускати цикл та зупиняти його роботу.

Функція "AsyncStart()" виконує асинхронний запуск головного циклу гри. Вона встановлює прапорець "Enabled" в значення "true", що дозволяє циклу продовжувати виконання. Після цього викликається корутин, яка виконує цикл гри.

У циклі проходиться через список завдань "TaskList" і викликається кожне завдання, передаючи йому об'єкт даних "DataObject". Після цього вимірюється час виконання циклу і, якщо час виконання менше, ніж встановлене значення "TickRate", виконується очікування за допомогою функції "task.wait()" до виконання наступного кадру. Цей процес повторюється, доки прапорець "Enabled" залишається увімкненим.

Функція "AddTask()" використовується для додавання завдання до головного циклу гри. Вона приймає назву завдання (типу string) і функцію, яка приймає параметри і не повертає значення. Ця функція додає завдання до списку "TaskList" з ключем, що відповідає його назві.

Функція "Stop()" використовується для зупинки головного циклу гри. Вона встановлює прапорець "Enabled" в значення "false", що призводить до припинення виконання циклу.

Модуль "BaseGameLoop" повертається як результат виконання коду. За його допомогою можна керувати головним циклом гри, додавати та видаляти завдання, а також асинхронно запускати і зупиняти роботу циклу.

3.4 Створення модуля економіки

Основною частиною цього модуля є файл AccountService. Він надає весь основний інтерфейс для роботи з цією системою. Цей модуль, під назвою AccountService , відповідає за керування гравцівськими рахунками в грі. Давайте розглянемо кожен метод та поле цього модуля детальніше:

Поля:

-Accounts: Поле, яке містить список об'єктів рахунків. Кожен об'єкт рахунку містить поля Owner (гравець, якому належить рахунок) і Money (кількість грошей на рахунку).

-DEFAULT_MODENY_COUNT: Поле, яке містить значення за замовчуванням для початкової кількості грошей на рахунку (Money) при реєстрації нового рахунку.

Методи:

module.Find(player: Player): Types.Account?: Цей метод приймає гравця (Player) як параметр і повертає об'єкт рахунку (Types.Account), пов'язаний з цим гравцем. Він перевіряє кожен об'єкт рахунку у списку Accounts і порівнює поле Owner з заданим гравцем. Якщо знайдено відповідний об'єкт рахунку, повертається цей об'єкт, інакше повертається nil .

module.RegisterAccount(player: Player): Цей метод реєструє новий рахунок для заданого гравця. Він перевіряє, чи існує вже рахунок для цього гравця, використовуючи метод module.Find(player). Якщо рахунок вже зареєстрований, генерується помилка. Інакше створюється новий об'єкт рахунку з власником (гравцем) і значенням Money , встановленим на DEFAULT_MODENY_COUNT , і додається до списку Accounts .

module.UnloadAccount(player: Player): Цей метод видаляє рахунок заданого гравця. Він перевіряє, чи існує рахунок для цього гравця, використовуючи метод module.Find(player) . Якщо рахунок не зареєстрований, генерується помилка. Поки цей метод просто виводить попередження "TODO: UnloadAccount", як підказку, що функціонал видалення рахунку ще не реалізований.

module.Send(sender: Player, target: Player, count: number): Цей метод відправляє гроші від одного гравця (sender) до іншого гравця (target). Він отримує об'єкти

рахунків для обох гравців, використовуючи метод module.Find(player). Потім перевіряє, чи обидва гравці зареєстровані, викликаючи module.Find(sender) та module.Find(target). Якщо хоча б один з гравців не зареєстрований, генерується помилка на стороні серверу.

Якщо гравець-відправник має достатню кількість грошей на рахунку, метод зменшує кількість грошей на рахунку відправника за допомогою методу module.Sub(sender, count), а потім збільшує кількість грошей на рахунку отримувача за допомогою методу module.Add(target, count).

module.Sub(player: Player, count: number): boolean : Цей метод зменшує кількість грошей на рахунку заданого гравця (player) на вказану кількість (count). Він перевіряє, чи існує рахунок для гравця, використовуючи метод module.Find(player) . Якщо рахунок існує, метод перевіряє, чи є достатня кількість грошей на рахунку для відняття. Якщо так, кількість грошей зменшується і метод повертає true . Якщо немає достатньо грошей, метод повертає false . Якщо рахунок не зареєстрований, генерується помилка.

module.Add(player: Player, count: number): Цей метод збільшує кількість грошей на рахунку заданого гравця (player) на вказану кількість (count). Він перевіряє, чи існує рахунок для гравця, використовуючи метод module.Find(player) . Якщо рахунок існує, кількість грошей збільшується на вказану кількість. Якщо рахунок не зареєстрований, генерується помилка.

module.AutoRegister(): Цей метод встановлює прослуховування подій PlayerAdded та PlayerRemoving , щоб автоматично реєструвати та видаляти рахунки гравців. При додаванні нового гравця відбувається виклик методу module.RegisterAccount , а при видаленні гравця - методу module.UnloadAccount . Після налаштування прослуховування метод повертає об'єкт модуля module .

В цілому, модуль AccountService надає функціонал для реєстрації, видалення та управління рахунками гравців у грі, а також передачі грошей між рахунками.

Для полегшення роботи з цим модулем використовується об’єкт: AccountAPI,

Цей модуль є простою обгорткою для взаємодії з модулем AccountService у зручний спосіб. Він надає чотири методи, які спрощують роботу з рахунками.

Його методи спрощують роботу з рахунками гравців, приховуючи деталі реалізації. Вони приймають необхідні параметри, викликають відповідні методи з

модуля AccountService і повертають результат. Такий підхід дозволяє зосередитись на основному функціоналі гри, не вдаючись у деталі управління рахунками гравців.

Загалом, цей модуль допомагає забезпечити зручну і просту взаємодію з функціями модуля AccountService, спрощуючи операції з рахунками гравців та передачу грошей між ними.

3.5 Реалізація системи завантаження режимів гри

Код в системі завантаження рівнів починається з клієнтського модуля GameModeClientModule, який є центральним об’єктом на клієнтській частині.

Модуль містить функцію onNewGameMode, яка викликається при початку нового режиму гри. Ця функція отримує ідентифікатор режиму гри та перевіряє наявність директорії з будівлями цього режиму гри. Якщо директорія не знайдена, генерується помилка. Далі створюється клієнтська будівля гри з використанням GameModeBuildCreator.CreateForSide і отриманої з директорії будівель.

Функція Run підключає обробник події newGameModeEvent.OnClientEvent, який викликає функцію onNewGameMode при отриманні нового режиму гри.

Код серверного модуля в системі завантаження рівнів починається з модуля GameModeLoader. Модуль містить функцію Load, яка викликається для завантаження режиму гри. Вона використовує обробник OnNewGame, який отримує параметри map (карта) та id (ідентифікатор режиму гри).

У функції Load створюється змінна serverBuild, викликаючи GameModeBuildCrator.CreateForSide з директорії GameModeBuildCreator, і отриманням серверної будівлі гри типу Types.ServerGameModeBuild.

Також створюється змінна connection для зберігання підключення до події changeGameEvent.Event. У цьому обробнику події реалізовано розрядку режиму гри. Відбувається виклик методу SideLoader.Unload серверної будівлі для розрядки карти. Далі сповіщається всіх клієнтів за допомогою події finishGameEvent про завершення гри на цій карті. Змінна serverBuild очищується за допомогою table.clear, і викликається знову функція Load для завантаження наступного режиму гри. Після цього підключення до події changeGameEvent розривається.

3.6 Реалізація системи Replicator

Система "Реплікатор" є компонентою, розробленою для зменшення затримки між дією користувача і відображенням цієї дії на екрані. Основна мета системи полягає в тому, щоб гравець міг бачити візуальні ефекти своїх дій практично в режимі реального часу, незалежно від часу, необхідного для синхронізації з сервером та іншими клієнтами.

Система "Реплікатор" складається з двох основних компонентів: ReplicatorHost та ReplicatorClient.

ReplicatorHost виконується на серверній стороні гри і відповідає за імітацію та передачу візуального ефекту. Коли гравець виконує дію, яка повинна бути візуально відображена, ReplicatorHost створює спеціальну подію, що містить дані про цей ефект. Подія потім відправляється до всіх підключених клієнтів у грі.

ReplicatorClient виконується на клієнтській стороні гри і приймає подію, що була відправлена від ReplicatorHost. Клієнт слухає цю подію і виконує необхідні дії для візуального відображення ефекту на локальному екрані. Наприклад, якщо гравець стріляє, ReplicatorClient може відтворити відповідний вогняний ефект або анімацію на екрані клієнта, щоб користувач мав відчуття миттєвої реакції на свою дію, та не бачить ніякої затримки.

Особливістю системи "Реплікатор" є те, що ReplicatorHost не приймає повідомлення від себе самого. Це досягається за допомогою маркера відправника, який додається до об'єкту, що відправляється. Таким чином, клієнти можуть ідентифікувати, що цей сигнал вже був відправлений хостом і не повторно відображати його.

Застосування системи "Реплікатор" включає різноманітні візуальні ефекти, такі як вогнепальні вибухи, частинки, анімації тощо. Вона дозволяє гравцям бачити свої дії відразу після їх виконання, створюючи враження безпосередньої відгуку на їх дії щоб покращити час відповіді.

В цілому, система "Реплікатор" допомагає покращити візуальний досвід гравців і зменшити відчутну затримку між дією користувача і відображенням цієї дії, що робить гру більш плавною та захоплюючою.

Клієнтська частина представлена наступними модулями

-ActionManager:

Функція module.Create(handlerDir) приймає один аргумент handlerDir, який є директорією з модулями обробників дій. Вона повертає об'єкт, який містить методи для використання обробників дій.

Основні методи об'єкта:

UseActions(actions: {Types.Action}, ignoreOwner: boolean?): Цей метод приймає масив дій actions та параметр ignoreOwner (опціональний). Він перебирає всі дії і викликає відповідний обробник дії (ActionHandler) залежно від ключа дії (action.Key). Якщо ignoreOwner встановлено на true, то дії, власником яких є гравець, ігноруються і не викликають обробник дії.

Загальний принцип роботи цього модуля полягає в тому, що він дозволяє використовувати обробники дій, які визначені в системі "Реплікатор". Це дозволяє клієнтам виконувати дії, отримані від сервера через реплікацію, і виконувати їх локально на клієнті.

Наприклад, якщо сервер відправляє дію "стріляння" до клієнта, клієнтський модуль "Реплікатор" може використати відповідний обробник дії для візуального ефекту стріляння на клієнті.

-ReplicatorClient:

модуль ReplicatorClient містить одну функцію OnGetPackage, яка викликається при отриманні пакету дій від сервера. Модуль експортує цю функцію для зовнішнього використання.

Основний компонент модуля:

module - таблиця, яка буде експортована з модуля.

Функція OnGetPackage(actions) приймає аргумент actions, який представляє масив дій отриманих від сервера. Вона викликає метод UseActions з об'єкта ActionManager з параметром true. Це дозволяє виконати дії, ігноруючи власника дій (тобто дії, виконані не гравцем-клієнтом).

Після виклику OnGetPackage виконується обробка отриманих дій шляхом виклику відповідних обробників дій з використанням ActionManager.

Функція OnGetPackage може використовуватися для обробки пакету дій, отриманих від сервера, і виклику відповідних обробників дій для виконання цих дій на клієнті.

-ReplicateHost:

виконує роль хоста в системі "Реплікатор". Він експортує декілька функцій та об'єктів для зовнішнього використання.

ActionManager - об'єкт, який представляє менеджер дій (ActionManager). Цей об'єкт імпортується з модуля ActionManager та створюється за допомогою функції Create, якій передається шлях до папки "Handlers" для імпорту обробників.

OnGetPackage(actions) - функція, яка викликається при отриманні пакету дій. Вона передає отримані дії до ActionManager для виконання з виключенням власника.

Після імпорту та створення ActionManager, функція OnGetPackage може бути викликана для обробки пакету дій. Вона передає отримані дії до ActionManager з параметром true, що означає, що власник дій буде ігнорований.

- ReplicatorEventHandler:

Основні методи об’єкту:

OnReplicate(plr, actions: {Types.Action}) - ця функція викликається при отриманні запиту на реплікацію дій від конкретного гравця plr. Вона приймає список дій actions, які клієнт бажає реплікувати, і викликає подію Replicate на клієнтській стороні, передаючи отримані дії усім клієнтам за допомогою FireAllClients.

Listen() - ця функція викликається для підключення обробника OnReplicate до події Replicate на сервері. Вона встановлює прослуховування події Replicate, яка відбувається при отриманні запиту на реплікацію дій від клієнта, та передає керування функції OnReplicate.

Після визначення функцій, модуль повертає таблицю module, яка містить внутрішні функції та об'єкти для зовнішнього використання.

Окремо слід зазначити, що модуль ReplicatorEventHandler взаємодіє з іншими модулями, такими як Replicator, Types, які забезпечують необхідні об'єкти та функціональність для роботи з реплікацією. Вони імпортуються і використовуються всередині модуля ReplicatorEventHandler. Цей модуль забезпечує підключення подій, та заставляє реагувати систему на зовнішні події. Саме він являється вхідною точкою всієї системи зі сторони сервера, та запускає всі інші підмодулі, та об’єкти зв’язані з цією системою.

Висновки до розділу 3

В розділі було описано процес реалізації ігрових механік, реалізація архітектури та впровадження функціоналу. Було розібрано такі речі як:

- Розробка серверної частина, створення точки входу програми через реалізацію сінг-скрипт архітектури.
- Розробка системи зберігання прогресу.
- Розробка клієнтської частини гри
- Реалізація систем оптимізації та покращення ігрового досвіду

ВИСНОВКИ

В результаті створення гри в жанрі top-down action на платформі Roblox Studio було досягнуто успіху. Гра надає захоплюючий геймплей, який пропонує гравцям активний досвід та екшн-пригоди у віртуальному світі. Розробка гри включала в себе наступні кроки:

- Аналіз існуючих ігор у жанрі top-down action для визначення їх переваг та недоліків.
- Вибір технологій для реалізації гри, таких як Roblox Studio, який надає потужні інструменти для створення ігрового середовища.
- Розробка захоплюючого та привабливого дизайну гри для забезпечення приємного візуального досвіду гравцям.
- Використання передових методик розробки геймплею, щоб забезпечити баланс між викликами та задоволенням.
- Проведення тестування гри для виявлення та усунення помилок та недоліків, забезпечення гладкого і безпроблемного геймплею.
- Застосування заходів з охорони праці під час розробки гри, щоб забезпечити безпеку розробників та гравців.

У результаті виконання цих кроків була створена захоплююча гра в жанрі top-down action на платформі Roblox Studio. Гра пропонує гравцям незабутні враження та можливість насолодитись захоплюючим геймплеєм.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Beginning Lua Programming, by Kurt Jung and Aaron Brown: вебсайт. URL: https://www.amazon.com/Beginning-Lua-Programming-Aaron-Brown/dp/1484204429 (Дата звертання “08.01.2023”).
2. Developing Games with Lua, by Paul Schuytema: вебсайт. URL: https://www.packtpub.com/product/developing-games-with-lua/97817832888 (Дата звертання “011.03.2023”).
3. Game Development with Lua, by Paul Schuytema: вебсайт. URL: https://www.packtpub.com/product/game-development-with-lua/97817821613(Дата звертання “18.03.2023”).
4. Learning Lua Programming, by Raul Fernandez Hernandez: вебсайт. URL: https://www.packtpub.com/product/learning-lua-programming/978178355306 (Дата звертання “25.03.2023”).
5. Love2D Game Development, by Michael Ivanov: вебсайт. URL:https://www.packtpub.com/product/love2d-game-development/978183860 (Дата звертання “30.03.2023”).
6. Lua for the World of Warcraft Addon Programming, by James Whitehead II вебсайт. URL: https://www.amazon.com/World-Warcraft-Programming-Second-Whitehead/dp/0982817851 (Дата звертання “5.04.2023”).
7. Lua Game Development, by Devyn McArthur: вебсайт. URL:https://www.amazon.com/Lua-Game-Development-Devyn-McArthur/dp/1783288414 (Дата звертання “24.02.2023”).
8. Lua Game Development Cookbook, by Mario Kasuba: вебсайт. URL: https://www.packtpub.com/product/lua-game-development-cookbook/9781783289533 (Дата звернення: 04.03.2023).

9. Lua Programming, by Kurt Jung and Aaron Brown: вебсайт. URL: https://www.amazon.com/Lua-Programming-Kurt-Jung/dp/161729184X (Дата звернення: 19.04.2023).
10. Lua Programming Gems, edited by Lennart Andersson: вебсайт. URL: https://www.lua (Дата звернення: 25.04.2023).
11. Lua Quick Start Guide, by Gabor Szauer: вебсайт. URL: https://www.packtpub.com/product/lua-quick-start-guide/9781789343221 (Дата звернення: 04.05.2023).
12. Mastering Lua, by Paul Schuytema: вебсайт. URL: https://www.packtpub.com/product/mastering-lua/9781785882759 (Дата звернення: 10.05.2023).
13. Programming Game AI by Example, by Mat Buckland: вебсайт. URL: https://www.amazon.com/Programming-Game-Example-Mat-Buckland/dp/1556220782 (Дата звернення: 13.05.2023).
14. Programming in Lua, by Roberto Ierusalimschy: вебсайт. URL: https://www.lua.org/manual/5.4/ (Дата звернення: 15.05.2023).
15. Procedural Generation in Game Design, by Tanya X. Short and Tarn Adams: вебсайт. URL: https://www.amazon.com/Procedural-Generation-Game-Design-Tanya/dp/1498799191 (Дата звернення: 18.05.2023).
16. Roblox Developer Journal: Lua Programming and Game Development, by Alexander Hillestad: вебсайт. URL: https://www.amazon.com/Roblox-Developer-Journal-Programming-Development/dp/1548370364 (Дата звернення: 20.05.2023).
17. Roblox Lua: Scripting for Beginners, by Douglas Snipp: вебсайт. URL: https://www.amazon.com/Roblox-Lua-Scripting-Beginners-Snipp/dp/1729378038 (Дата звернення: 20.05.2023).
18. Roblox Lua: The Complete Guide, by Brandon LaRouche: вебсайт. URL: https://www.amazon.com/Roblox-Lua-Complete-Brandon-LaRouche/dp/1789616709 (Дата звернення: 15.06.2023).
19. The Roblox Lua Dictionary: Building Tools for Beginners, by Douglas Snipp: вебсайт. URL: https://www.amazon.com/Roblox-Lua-Dictionary-Building-Beginners/dp/1729378151 (Дата звернення: 15.06.2023).

20. Unity in Action: Multiplatform Game Development in C#, by Joe Hocking вебсайт. URL: https://www.manning.com/books/unity-in-action-second-edition (Дата звернення: 30.05.2023)