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 3D ГРИ "The Language of Guns" ЖАНРУ ШУТЕР-СТРАТЕГІЯ З ВИКОРИСТАННЯМ ІГРОВОГО ДВИГУНА UNITY»

Виконав: студент 4 курсу, групи ПД–44
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Левчук М.П.
(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 підготовки – 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

ЛЕВЧУКА МИКОЛИ ПЕТРОВИЧА
(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка 3D гри "The Language of Guns" жанру шутер- стратегія з використанням ігрового двигуна Unity»

   Керівник роботи: Дібрівний Олесь Андрійович, 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 Методики створення відеоігрового програмного продукту;
   3.2 Ресурси науково-технічної літератури, які стосуються розробки програмного забезпечення для відеоігор;
   3.3 Офіційна документація Unity
   3.4 Офіційна документація Microsoft Visual Studio
   3.5 Офіційна документація мови програмування C#

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

    4.1 Аналіз актуальності та проблематики розроблюваної гри
    4.2 Аналіз та вибір інструментів для розробки даної гри
    4.3 Опис проектування гри
    4.4 Висновки

5. Перелік демонстраційного матеріалу (назва основних слайдів)

    5.1 Титульний слайд
    5.2 Індивідуальне завдання
    5.3 Аналіз та загальна характеристика ігор-шутерів
    5.4 Мета, об'єкт та предмет дослідження
    5.5 Аналіз існуючих рішень
    5.6 Технічні завдання
    5.7 Висновки
    5.8 Кінцевий слайд

6. Дата видачі завдання «25» лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	08.04.2023	Виконано
2	Дослідження аналогів та актуальності додатку	13.04.2023	Виконано
3	Аналіз та вибір інструментів для розробкидодатку	14.04.2023	Виконано
4	Проектування та реалізація	02.05.2023	Виконано
5	Вступ, висновки, реферат	16.05.2023	Виконано
6	Розробка обов‘язкових демонстраційнихматеріалів	17.05.2023	Виконано
7	Попередній захист роботи		
8	Здача роботи	01.06.2023	

Студент _____ Левчук М.П.
(підпис) (прізвище та ініціали)

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

РЕФЕРАТ

Текстова частина бакалаврської роботи с.58, 7 рис., 18 джерел.

Ключові слова: сюжет, 3D гра, шутер, Unity, механіка, ігровий рушій.

Об’єкт дослідження – ігровий процес додатку в жанрі «шутер-стратегія».

Предмет дослідження – програмне забезпечення для реалізації ігрового додатку в жанрі «шутер-стратегія»..

Мета роботи – розширення функціональних можливостей гри жанру «шутер-стратегія» шляхом додавання механік сучасної війни.

Наукова новизна – Однією з ключових інновацій є впровадження тактичних елементів в геймплей шутера, що дозволяє гравцеві взяти під контроль ігрове поле та планувати свої дії.

У дипломній роботі було розглянуто ринок комп’ютерних ігор жанру шутер, виявлено переваги та недоліки інструментів їх розробки.

Для створення комп’ютероної гри був використаний рушій для створення ігор Unity, а код був написаний мовою програмування C# у інтегрованому середовищі розробки Visual Studio 2022

ЗМІСТ

ВСТУП

Відеоігри - це дуже популярна форма забав в сучасному світі. В останній час вони стали суттєвою частиною культурного та розважального життя суспільства. Даними, які дають дослідження, доведено, що відеоігри мають потужний вплив на психологічні, соціальні та культурні сторони життя людей.

У цьому контексті гра жанру шутер-стратегія є особливо цікавою та актуальною темою для дослідження. Шутери та стратегії є двома з найбільш популярних жанрів відеоігор. Поєднання цих жанрів може створити унікальний геймплей, який приверне увагу гравців та дозволить розширити можливості розвитку відеоігор.

Метою даної дипломної роботи є дослідження наукової новизни та можливостей створення відеоігри жанру шутер-стратегія. В роботі будуть розглянуті основні принципи створення гри, її геймплею та механіки. Також будуть розглянуті технологічні аспекти створення гри та її розробка з використанням сучасних програмних засобів. В результаті роботи ми плануємо розробити прототип гри жанру шутер-стратегія, що має потенціал стати успішним продуктом на ринку відеоігор.

Отже, дана дипломна робота є актуальною та важливою, оскільки вона розкриває можливості розвитку нового жанру відеоігор та досліджує наукову новизну гри жанру шутер-стратегія.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття відеогри

Відеогра або комп'ютерна гра є електронним розважальним застосунком, який залучає користувача до інтерактивної співдії за допомогою різних керуючих обладнань, таких як джойстик, клавіатура, контролер, миша, та інші. Вона дає візуальне відображення діяльності на відеопристроях, таких як монітор, екран, дисплей або пристрої віртуальної реальності. Багато відеоігор також мають аудіовідтворення, яке поширюється через динаміки або навушники, а також можуть мати додаткові відчуття, наприклад тактильну відгуку у вигляді вібрації. Варто зауважити, що існують відеоігри, які не обмежуються конкретним пристроєм виведення відео та не відносяться до категорії комп'ютерних відеоігор.

Використовуючи спеціальні алгоритми, користувач може імітувати взаємовплив з уявними постатями, сутностями або іншими гравцями через програму. Це дає змогу створити віртуальне середовище, в якому людина може взаємодіяти та відтворювати різні сценарії. Такий підхід дозволяє відтворити різні ситуації та експериментувати з різними варіантами поведінки.

Багато навчальних процесів використовують гейміфікацію, щоб спростити засвоєння матеріалу. В деяких школах комп'ютерні ігри використовуються як засіб навчання, особливо коли йдеться про шкільну програму. Наприклад, "Сходинки до інформатики" - гра для школярів, де діти навчаються навичкам праці з комп'ютером, закріплюють вивчений матеріал з інших предметів, а старші учні навіть ознайомлюються з основами програмування. Цей підхід допомагає зробити навчання цікавим та залучити учнів до активної пізнавальної діяльності

В останні роки кіберспорт став визнаним офіційним видом спорту в деяких країнах. Це означає, що гравці, які займаються відеоіграми на професійному рівні,

отримують визнання і підтримку, схожу з традиційними спортсменами. Кіберспорт стає все більш популярним і отримує значну кількість шанувальників та глядачів, а також привертає увагу спонсорів та медіа. Це розкриває нові перспективи для гравців у цій сфері та розвитку кіберспортивної індустрії в цілому.

У результаті можна підсумувати, що відеоігри стають все більш важливою складовою нашого повсякденного життя і поступово проникають у різні його сфери, не обмежуючись лише розвагами. Ігри виявляють свій вплив на освіту, культуру, здоров'я та інші аспекти нашого життя. Вони можуть бути застосовані як інструмент для навчання, спілкування, спортивних змагань та навіть лікування. Це каже про все більшу визнаність їхньої значущості і потенціалу у нашому суспільстві

Відеоігри можуть бути доступні на різних платформах, які класифікують їх. Такими платформами можуть бути аркадні автомати, консолі для ігор, такі як PlayStation та Xbox, або персональні комп'ютери. З недавніх часів, відеоігри також можуть бути доступні на мобільних пристроях, таких як смартфони та планшети, а також на системах доповненої та віртуальної реальності (AR та VR відповідно), а також у форматі ігор у хмарі. Крім того, відеоігри можуть бути класифіковані за жанром в залежності від доступних типів взаємодій для гравців.

У 1950-1960-х роках з'явилися перші відеоігри, які представляли собою прості налаштування електронних пристроїв для відображення на великих комп'ютерах. Але першою грою, що здобула популярність серед широкої аудиторії, стала аркадна гра "Computer Space", випущена в 1971 році. У наступному році з'явилася одна з найуспішніших ігор того періоду - "Pong", а також перша домашня ігрова консоль "Odyssey" від Magnavox. Золотий вік відеоігор настав в кінці 70-х і з початку 80-х років, проте через недбале видавництво та перенасиченість ринку однотипними іграми, галузь зазнала кризи. Однак, після цього ринок зріс і компанії, родом з Японії, такі як Sony, Nintendo, Sega та інші, стали його лідерами. Розробники вдосконалили підходи розробки та поширення відеоігор, щоб

уникнути подібних криз у майбутньому. На сьогоднішній день успішне створення відеоігри вимагає залучення розробників, видавництв, магазинів, комп'ютерних та консольних інженерів, а також фахівців інших областей.

У першому десятилітті 21-го століття основний фокус в ігровій індустрії був на великих проектах від відомих розробників та видавництв, так званих AAA-іграх. Це призвело до обмеження ризиків і експериментів, оскільки вони не завжди були прибутковими. Однак з поширенням інтернет-технологій в другому десятилітті ситуація змінилася. Створення та видання ігор стало значно доступнішим, а на передній план вийшли інді-студії та їх проекти. Вони стали популярним напрямком у галузі відеоігор протягом 2010-х років. Це призвело до значного зростання значення та вартості ігрової індустрії. Особливо в Азії велику популярність отримали ігри на мобільних пристроях, таких як смартфони, планшети та портативні консолі, наприклад, PlayStation Portable (PSP). Також спостерігається зміна демографії споживачів, адже з'являються ігри, спрямовані на різні аудиторії та більш інклюзивні. Одним з нових напрямків є сервісні пропозиції ігор, які називають "Games as a Service" (GaaS), де гра може бути доступна через підписку на певний період часу, замість традиційної покупки за одну суму. Завдяки цьому зростанню, доходи глобальної ігрової індустрії в 2020 році склали 159 мільярдів доларів США, що втричі перевищує прибуток індустрії музики за попередній рік і в декілька разів більше, ніж прибуток кіноіндустрії в той же період.

Найраніші відеоігри використовували різні способи виведення інформації на екрани. Один з найстаріших прикладів - обладнання для забав з електронно-променевою трубкою, розроблений Томасом Т. Голдсматом-молодшим та Естлом Рей Манном. Цей пристрій, що використовував технологію радіолокації, мав аналоговий контроль, що імітував постріл ракети по паперовій мішені на екрані. Інші ранні прототипи включали гру "Чернетки" Крістофера Стрейчі, комп'ютер Nimrod, який був представлений на Британському святкуванні у 1951 році, OXO - хрестики-нолики на комп'ютері EDSAC Олександра С. Дугласа у 1952 році,

інтерактивну гру для двох осіб - Tennis for Two Вільяма Хігінботама у 1958 році, а також Spacewar!, створену учнями Массачусетського технологічного інституту (MIT) Мартіном Грецом, Стівом Расселом та Вейном Вітаненом, яку часто називають першою грою жанру "рів". Кожна з цих ігор мала свої способи відображення: NIMROD використовував панель підсвічування, OXO - графічний дисплей, Tennis for Two - осцилограф, а Spacewar! - дисплей на основі векторів DEC PDP-1.

Прототипи цих ігор відкрили нові горизонти для майбутніх відеоігор. У 1966 році Ральф Х. Баєр створив настільний теніс на телевізійному екрані в компанії Sanders Associates, що призвело до створення "Brown Box". Цей винахід був ліцензований компанією Magnavox для створення першої домашньої консолі Magnavox Odyssey, яка з'явилася на ринку у 1972 році. Ідея Magnavox Odyssey надихнула Нолана Бушнелла і Теда Дабні на створення меншої та доступнішої аркадної платформи. Вони створили аркадну відеогру Computer Space, яка вийшла у 1971 році. Бушнелл і Дабні заснували компанію Atari Inc. і випустили аркадну гру пінг-понг у 1972 році, відтворюючи ідею тенісу з Odyssey. Magnavox та Sanders подали позов до суду проти Atari за порушення патентів Баєра, але справа була вирішена мирним шляхом з компенсацією за використання патентів. Це дало можливість Atari у 1975 році створити Pong - ще одну успішну домашню консоль, що започаткувала індустрію відеоігор. Баєр і Бушнелл стали відомими як "батьки відеоігор".

Термінологія

Термін "відеоігри" був введений для виділення ігор, які використовують відеодисплей для графічного відображення, від ігор, що використовують інші пристрої, такі як принтери. Використання цього терміну також відрізняло відеоігри від ігор, де стан відображався лампами, але не формувався зображення, наприклад, гра Merlin.

Термін "комп'ютерна гра" може охоплювати всі види відеоігор, оскільки вони

користуються комп'ютерними процесорами. Проте, існують специфічні застосування цього терміна, які вказують на те, що гри запускаються на персональних комп'ютерах, а не на консолях. У минулому були інші терміни, що використовувалися для опису ігор на консолях, наприклад "телевізійна гра" або "телегра". В Японії такі ігри відомі як "terebi geemu", що є перекладом з англійської. Крім того, термін "електронна гра" може використовуватися для опису відеоігор, але також може відноситися до ранніх портативних електронних ігор, що не мали відеовиходів. У сучасному світі термін "телевізійна гра" широко використовується для опису відеоігор на консолях, які підключаються до телевізорів.

Термін "відеогра" з'явився приблизно у 1973 році і вперше згадується в статті журналу BusinessWeek від 10 листопада 1973 року. Нолан Бушнелл стверджує, що термін виник у журналі Computer Space вже у 1971 році. Дослідження журналів Vending Times і Cashbox довели, що термін активно використовувався з березня 1973 року, зокрема виробниками аркадних ігор. Історик кмоп’ютерних ігор Кіт Сміт вказує на широку популярність терміну завдяки підтримці фахівців у цій галузі. За теорією, Ед Адлум, колишній керівник розважального відділу журналу Cashbox, засновник журналу RePlay Magazine, використав термін "відеоігри" у випуску журналу у вересні 1982 року. Бушнелл, Пет Карнс і розробники, такі як Генрі Лейзер і брати Мак'юен, не використовували термін "телевізійні ігри" і прийняли термін "відеоігри" з опису музичних автоматів з журналу Billboard. Адлум пояснив, що до 1970-х років розважальні центри переважно мали аркадні ігри без відеовиходу, такі як пінбол та електромеханічні ігри. З появою відеоігор виникла плутанина щодо термінології для опису нової продукції.

Платформа

Відеоігри працюють на різних платформах, що складаються з електронних компонентів та програмного забезпечення, яке спеціально розроблено для конкретної системи. Цей процес також часто описують терміном "система". Зазвичай ігри створюються для певної платформи або обмеженої кількості

платформ, а потім можуть бути адаптовані для роботи на інших платформах - проведені порти. Іноді ексклюзивні версії гри створюються з метою набування спроможєної переваги на ринку. Процес портування може відбутися набагато пізніше, коли конкурентна перевага вже не має вагомого значення, але продажі ще потрібно збільшити. Для залучення більшої аудиторії існує також можливість випуску ремастерів - коли більшість коду старої гри залишається без змін, але графіка та звукове оформлення покращуються, деякі помилки виправляються, і гра стає доступною для новіших платформ. Ремейки включають створення повністю нового коду, графіки та звукового оформлення на основі сюжету старої гри.

Комп'ютерна гра

Більшість комп'ютерних ігор призначені для використання на персональних комп'ютерах (ПК), які підключаються до пристроїв виведення відео, таких як монітори або телевізори. Використання ПК не обмежується лише грамі, оскільки його різноманітні компоненти можуть призводити до відмінностей та помилок між запущеними на ньому іграми, особливо порівняно з консолями. Будучи відкритою платформою, ПК дає розробникам можливість експериментувати з компонентами, поєднувати їх у різних комбінаціях, змінювати програмне забезпечення, використовувати доступніші версії та модифікувати ігри. Виникла також можливість хмарного геймінгу, яка дозволяє гравцям звертатися до серверів, що виконують всі обчислення, а передача відео відбувається через Інтернет. Проте, для отримання високої якості такого досвіду потрібне швидке та стабільне підключення, і можуть виникати затримки.

Домашня консоль

Домашні консолі можна розглядати як більш стандартизовану версію ПК, але зазвичай замкнуту платформу, яку неможливо змінювати. Вони призначені для підключення до телевізора або монітора і спеціально розроблені для відеоігор. Розробники мають можливість оптимізувати ігри для певного набору електронних і програмних компонентів, що дозволяє забезпечити єдність ігрового

досвіду та уникнути непотрібних помилок. Зазвичай на консолях запускаються ігри, розроблені спеціально для цієї конкретної консолі або продуктів від тієї ж компанії, але не для конкурентів. Це призводить до розбіжностей на ринку консолей та обмежує можливості експериментів, які доступні на ПК. Наразі найпопулярнішими консолями є PlayStation та Xbox.

Портативна консоль

Портативна ігрова консоль - це компактний пристрій, що містить вбудовані контролери, дисплей, аудіосистему та батарею, що дозволяє грати в ігри в будь-якому місці. Зазвичай портативні консолі менш потужні, ніж стаціонарні консолі або ПК. У 1990-х та 2000-х роках багато портативних консолей використовували картриджі, що дозволяло грати у більш ніж одну гру на цій платформі. Однак у 2010-х роках портативна консоль стала менш популярною через появу смартфонів та ігор на них, які стали більш зручним варіантом для гравців. [1]

Аркадна відеогра

Аркадна відеогра - це гра, що призначена для гри на спеціальному, максимально спеціалізованому пристрої, зазвичай, призначеному для однієї гри та розміщеному у спеціальному просторі, так званій ігровій шафі. Ігрова шафа містить дисплей, динаміки і контролери, а також, іноді, інші елементи, необхідні для гри. Часто аркадні ігри зібрані у спеціальних центрах, де для їх запуску використовуються монети. Зазвичай аркадні ігри мають яскраве оформлення, що приваблює погляд, та відповідає тематиці гри. Хоча більшість аркадних ігор розміщені у вертикальних шафах, деякі розміщуються на столах з прозорим верхом, які дозволяють розмістити екран у горизонтальному положенні. У настільних аркадних іграх, як правило, грають у сидячому положенні. На сьогоднішній день аркадні ігри не так популярні, як вони були у 90-х та 2000-х роках, проте, іноді, їх можна знайти у кінотеатрах та розважальних центрах, де є спеціально відведені зали для їх гри.

Браузерна гра

Гра у браузері має переваги спільного середовища для гри, що означає використання спільного інтерпретатора коду, такого як Blink у Google Chrome, для створення стандартизованого ігрового досвіду. Ці відеоігри можуть бути доступні на сайтах, наприклад, Miniclip, або використовують технології розробки, такі як Java або Flash.

Мобільна гра

Після стандартизації платформ iOS та Android, мобільні ігри зайняли значну частину ринку, використовуючи переваги, що надає їм доступність компонентів, таких як акселерометр, GPS або камера для доповненої реальності, які не завжди доступні на інших платформах.

Хмарні ігри

Хмарні ігри можна грати на будь-якому пристрої, який має з'єднання з Інтернетом та дисплей, включаючи не потужні комп'ютери, консолі, ноутбуки та мобільні телефони, за допомогою програмного забезпечення постачальника сервісу. У цьому випадку, всі обчислення відбуваються на сервері постачальника послуг, тоді як клієнт отримує лише відображення гри, що дуже зручно для користувачів не потужних пристроїв. Однак, є затримки між введенням гравцем даних та відображенням інформації на екрані, але це вирішується за допомогою методів передбачення. Для гри в хмарні ігри потрібен швидкий Інтернет та стабільний доступ до мережі. До прикладів таких ігор належать Xbox Cloud Gaming, Playstation Now, які використовують спеціальні сервери з лезом.

Віртуальна реальність

Ігри у віртуальній реальності (VR) є новим видом ігор, який потребує використання спеціальних пристроїв, таких як окуляри та маніпулятори, щоб гравець міг максимально зануритись у світ гри та прямо взаємодіяти з ним. Для підключення до цих пристроїв зазвичай потрібен окремий блок для обробки, який може бути ПК, консоль тощо. На сьогоднішній день провідними компаніями-виробниками пристроїв віртуальної реальності є Oculus (належить Meta, колишній

Facebook), Valve та Sony. Окуляри реагують на рухи голови гравця та дозволяють бінокулярно бачити 3D-об'єкти, тоді як маніпулятори забезпечують пряму взаємодію з ігровим світом.

Емуляція

Емулятор - це програмне або апаратне забезпечення, яке дозволяє імітувати середовище, для запуску якого було оптимізовано гру. Зазвичай він реалізується як віртуальна машина на сучасній системі, яка імітує апаратне забезпечення оригіналу та дозволяє грати в старі ігри. Хоча в судовому праві США емулятори є законними, саме програмне забезпечення, що буде використовуватися на них, може підпадати під закон про авторське право. Крім того, існує емульоване програмне забезпечення від офіційних виробників, таке як Virtual Console або Switch Online від Nintendo.

Зворотна сумісність

Зворотна сумісність означає можливість грати в ігри, розроблені для старіших платформ, на новіших. Це досягається за допомогою вбудованого програмного забезпечення, яке дозволяє підтримувати ігри, розроблені для старіших платформ. Наприклад, PlayStation 2 може запускати ігри для PlayStation, просто зчитуючи дані з носія, а Nintendo Wii може запускати ігри для Nintendo GameCube. Зворотна сумісність схожа на емуляцію, але відрізняється тим, що вбудоване програмне забезпечення дозволяє грати в ігри без необхідності використання окремих емуляторів.

Ігрові медіа

На початку історії відеоігор, щоб пограти, необхідно було мати спеціальні пристрої з ігрою вбудованою в апаратні компоненти. З того часу технології розвинулися, і тепер більшість ігор розповсюджуються через фізичні носії (наприклад, CD-диски, DVD-диски, карти флеш-пам'яті) або через інтернет. Сучасні консолі дозволяють мати цілу колекцію ігор та грати в них на вибір без необхідності вставляти носій у консоль. Більше того, зараз фізичні носії служать

для встановлення гри на внутрішню пам'ять пристрою, оскільки зчитування з них занадто повільне для постійного використання.

Зазвичай, ігри можна розширювати за допомогою нового вмісту, пакетів, розширень – програмного забезпечення, котре розроблюються окремо від самої гри. Це може бути безкоштовним або використовуватися для заробітку після виходу гри на ринок. Деякі ігри надають можливість створювати власний цифровий контент та ділитися ним із іншими гравцями. Інші ігри, зазвичай на ПК, можуть отримати додатковий функціонал завдяки створеним гравцями модифікаціям, які змінюють або доповнюють гру. Ці модифікації можуть бути неофіційними та розробленими за допомогою реверс-інжинірингу гри, але деякі компанії надають можливість офіційно модифікувати свої ігри.

Пристрій введення

Для фіксації дій користувача та переведення їх у поведінку в ігровому світі використовуються різні пристрої, такі як геймпад, джойстик, клавіатура, мишка та інші контролери. Контролери можуть мати різні елементи керування, такі як кнопки для визначення напрямку зору, тригери на боках, аналогові джойстики та перехрестя управління. Стандартні контролери зазвичай входять у комплект з консолями, але існують також й інші контролери, наприклад, руль та педалі, танцювальні майданчики, ігрові пістолети та інші спеціалізовані пристрої для певних жанрів ігор. Деякі новіші контролери мають додаткові технології, такі як дисплей, сенсорний екран та датчики виявлення руху, що надають більше можливостей для взаємодії із ігровим світом. Крім того, цифрові камери та детектори руху можуть зчитувати рухи тіла гравця і переносити їх до ігрового світу, що створює або доповнену, або віртуальну реальність і сприяє кращому зануренню у гру. Контролери доступні для купівлі окремо від виробника консолі або від сторонніх виробників, а також вони вбудовуються в портативні консолі та аркадні шафи.

Відображення та вихід

Ціль кожної відеогри, як і підказує її назва, має сенс в тому, щоб відтворити графіку на зовнішньому дисплеї, такому як телевізори на основі електронно-променевих трубок, новіші екрани з рідкокристалічним дисплеєм (LCD), оснащені екрани, проектори або комп'ютерні монітори зі схожими технологіями відображення. Однак, кількість деталей, яку може відображати графіка, а також частота оновлення, частота кадрів та роздільна здатність екрану обмежуються як самою грою, так і платформою, на якій вона запускається. Ці обмеження залежать від технологій відображення, кількості операцій на секунду та кількості полігонів у графіці, а також від конкретної платформи.

Крім того, відображення гри може змінюватися в залежності від обраного дисплею та типу графіки, яку використовує гра (2D або 3D), а також від технологій, які використовуються в самій грі.

Зазвичай, звук, який супроводжує програмне забезпечення, доповнює графіку та відтворюється на внутрішніх динаміках платформи або на зовнішніх пристроях. Звук може бути використаний для передачі дій користувача та інформації про події в ігровому світі, а також як фонова музика або музичне супроводження до певних подій.

Деякі ігрові платформи та їх контролери пропонують додаткові технології зворотного зв'язку з гравцем, які гра може використовувати. Ці функції зазвичай включають тактильний відгук вбудованих у геймпади, коли контролер вібрує в руках гравця, що імітує події у грі, які безпосередньо впливають на гравця.

Класифікація

Жанр

Властивість відеоігор полягає у їх поділі на жанри, аналогічно до інших форм медіа. Однак, на відміну від кіно, музики або серіалів, де зображення, сюжетні лінії та музичні супроводи формують основу класифікації, жанри відеоігор визначаються на основі взаємодії з процесом гри, оскільки саме цей елемент є головним у грі. Сюжет не є основним фактором, тому що платформер залишається

платформером, незалежно від того, чи є це фентезі чи фантастика. Виключенням є лише жанр ігор-жахів, де використовуються такі елементи, як надприродні явища, фантастика або психологічний тиск, щоб викликати страх у гравця.

Назви ігрових жанрів зазвичай є коротким і ємним описом того, що гравець може очікувати від гри. Наприклад, жанри екшн, платформер, шутер та рольова гра (RPG) містять в собі загальний опис ігрового процесу. Однак, деякі жанри носять назву на честь відомих творів, які вплинули на їх створення. Наприклад, гра "Rogue" визначила жанр "roguelike", серія ігор "Grand Theft Auto" заснована на жанрі "gta-клонів", а фільм "Battle Royale" послужив основою для жанру "Battle Royale" ігор.

Термінологія в ігровій індустрії постійно змінюється та оновлюється. Імена жанрів можуть змінюватись з часом, коли гравці, ЗМІ або розробники створюють нові терміни. Наприклад, перші шутери називались "Doom-like" або "клонами Doom", тому що вони нагадували гру "Doom", яка була видана в 1993 році та вважається впливовою у своєму жанрі.

Жанри можна ієрархізувати, поділяючи їх на вищі та нижчі рівні. "Шутер" та "екшн" є жанрами вищого рівня, описуючи загальний стиль гри. Для кожного жанру існує кілька піджанрів, що описують конкретну реалізацію. Наприклад, піджанри шутерів можуть бути від першої або третьої особи.

Деякі ігри складно класифікувати до якогось одного жанру. Тому існують міжжанрові типи, такі як пригодницький екшн, які поєднують у собі елементи кількох різних жанрів.

Режим

Режим відеоігор - це один з ключових аспектів, що визначає кількість гравців, які можуть брати участь у грі одночасно. Відеоігри можна розділити на дві основні категорії: для одного гравця та для багатьох гравців. Остання категорія може мати декілька підкатегорій, в залежності від того, як саме гра організовує гравців. Наприклад, деякі ігри дозволяють грати на одному комп'ютері, тоді як

інші потребують LAN-мережі, або використовують інтернет для гри з іншими гравцями з усього світу.

Більшість відеоігор для кількох гравців містять у собі елемент змагання, коли гравці конкурують між собою за перемогу. Також існують командні або кооперативні ігри, в яких гравці співпрацюють між собою, щоб досягти спільної мети. Деякі ігри мають асиметричний ігровий процес, де гравці мають різний набір здібностей та мету гри.

Онлайн-ігри потребують використання серверного апаратного забезпечення, щоб організувати гру для декількох гравців одночасно. Ці ігри можуть підтримувати різну кількість гравців - від декількох до сотень тисяч. Ігри, які підтримують гру для сотень гравців одночасно, класифікуються як MMO (massively multiplayer online games). У таких іграх гравці можуть взаємодіяти між собою та відтворювати віртуальні світи за допомогою своїх персонажів.

Існують такі види ігор, які мають умовну нульову кількість гравців та обмежену взаємодію з ігровим світом. Ці ігри називаються симуляторами, де можна встановити початкові умови симуляції та залишити процес самостійно протікати, пасивно спостерігаючи за результатами. Прикладом такої гри є Життя, яку вигадав Джон Конвей у 1970-му році..

Намір

Багато відеоігор розробляються з метою забезпечення розваги та розважального дозвілля. Ці ігри часто називають "основними іграми". Однак, окрім цієї підмножини існує також категорія ігор, що мають інші цілі, крім розваг. До таких ігор можна віднести:

Казуальні ігри

Більшість ігор мають на меті забаву та розвагу, але є й такі, що призначені для щоденного використання та мають простий ігровий процес. Ці ігри зазвичай доступні і мають прості правила, спрямовані на широку аудиторію. Їх часто можна відновити після виходу з гри, наприклад, під час перерви в роботі або

очікуванні в черзі. Ці ігри називаються казуальними, і часто вони є веб-іграми або іграми для мобільних пристроїв, що не потребують глибокого мислення. Наприклад, три-в-ряд, знаходження прихованих об'єктів або клікер - ігри, в яких потрібно багато разів клікати по об'єктах. Казуальні ігри поширені в соціальних мережах, де гравець може ділитися своїми досягненнями з друзями, а також отримувати додаткові можливості для гри. Наприклад, Tetris та Candy Crush Soda Saga - це дуже популярні казуальні ігри.

Файтинг ігри

Жанр файтингових ігор характеризується прямим поєдинком між персонажами на обмеженій арені. Це відрізняє файтинги від інших ігор, таких як "побий їх усіх". У більшості файтингів гравець не може покинути арену і битва складається з кількох раундів з обмеженим часом. У інтерфейсі гри також відображаються шкали, що відображають життєві показники персонажів.

Стратегії

Стратегічні ігри вимагають від гравця розробки та реалізації ефективної стратегії з метою досягнення успіху. У таких іграх гравець має контролювати різні аспекти гри, такі як господарство, військова тактика, дипломатія та інші. Вони можуть бути покроковими, де гравці здійснюють ходи по черзі, або в реальному часі, де дії відбуваються безперервно. У стратегічних іграх гравцям необхідно приймати важливі рішення, аналізувати ситуацію та реагувати на зміни в грі, щоб здобути перемогу.

Пригодницькі ігри

Пригодницькі ігри - це ігри, які розповідають захоплюючу історію, в якій головний герой просувається через сюжет і взаємодіє з ігровим світом. У таких іграх використовуються різні предмети, взаємодія з квестовими персонажами і розв'язування різних завдань та головоломок. Гравець має можливість приймати рішення, що впливають на розвиток сюжету та подальший хід гри.

Рольові ігри

Рольові комп'ютерні ігри, або RPG, є популярним жанром ігор, який поєднує елементи традиційних рольових ігор з ігровим процесом. Багато сучасних ігор, особливо AAA-проектів, включають елементи RPG для збільшення тривалості та цікавості гри. У рольових іграх гравець приймає управління одним або декількома персонажами, кожен з яких має унікальні характеристики, здібності та навички. Протягом гри ці характеристики можуть розвиватися, параметри персонажів покращуватися, а нові здібності вивчатися.

Розвиваючі ігри

Ігри можуть мати не тільки розважальну функцію, але й стати корисним інструментом для освіти дітей та студентів. Вони можуть бути інтерактивними та викликати інтерес до навчання. Розвиваючі відеоігри можна розділити на дві групи - перші зосереджуються на розвагах та запам'ятовуванні матеріалу, не вимагаючи критичного мислення. Інші мають на меті мотивувати гравців до розв'язання проблем та закріплення матеріалу, при цьому не забуваючи про розважальну складову гри. Прикладами таких ігор можуть бути The Oregon Trail та серія Carmen Sandiego. Крім того, деякі елементи відеоігор можуть бути корисними для навчання, хоча їх перші розробники і не думали про це. Наприклад, Minecraft має відкритий світ, що дозволяє досліджувати та вивчати різні предмети. А гра SpaceChem має складні елементи головоломки, які допомагають вчитися алгоритмічного мислення.

Серйозні ігри

Серйозні ігри - це ті, які зосереджуються на вирішенні конкретної проблеми або виконанні певних завдань, а не просто на розвагах. Навчальні ігри є одним з типів серйозних ігор, але категорія включає в себе й інші види ігор, такі як фітнес-ігри, де гравцеві потрібно виконувати фізичні вправи для збереження форми, на приклад, як у грі Wii Fit. До серйозних ігор також можна віднести симулятори польотів, як Microsoft Flight Simulator, рекламні ігри, які сприяють популяризації продукту, на приклад, Pepsiman, або новинні ігри, що намагаються передати певне

повідомлення, на приклад, NarcoGuerra.

Художня гра

Відеоігри можуть не тільки втілювати собою шедеври мистецтва, але й виступати засобом передачі історії або задуму. Такі ігри називаються артхаусними та створені з метою викликати емоційну реакцію у гравця. Вони можуть мати нетипові для відеоігрів елементи, такі як відсутність кінцевої цілі, а замість цього бути спрямовані на дослідження ігрового світу та його сценарію гравцем. Зазвичай такі ігри є інді-іграми, створеними однією людиною або маленькою командою розробників і базуються на особистому досвіді. До прикладів художніх ігор належать "That Dragon, Cancer", Flower та Passage.[2]

Останні дані про доходи від геймінгової індустрії свідчать про її неймовірне зростання, що порівнюється з успіхом кінотеатрального бізнесу. Ігри стають все більш популярними формами розваги, і не виключено, що незабаром вони зможуть перевершити кінопрокат за обсягом прибутку. Таким чином, ігрова індустрія прогресує й розширює свій вплив на розважальний сектор, притягуючи інвестиції та заробляючи вражаючі суми грошей.

За останній період спостерігається значний зріст на ринку комп'ютерних ігор, з доходами, які збільшилися з $29,4 млрд до $32,3 млрд. У той же час, ігри для консолей відзначилися трохи меншими доходами в розмірі $33,3 млрд у 2017 році, але все ж перевищили показники 2016 року майже на 4%. Японія та Китай стали лідерами у зростанні доходів у 2017 році, оскільки ці країни мають найбільшу кількість гравців.

1.2 Поняття шутеру

Шутер - це жанр відеоігор, в основі якого лежить геймплей, в якому гравець контролює персонажа, зазвичай з озброєнням, з метою виконання завдань та виживання у ворожій території. У шутерах можна виділити кілька піджанрів:

перші особи, треті особи, тактичні та аркадні.

Шутери першої особи, або FPS (First-Person Shooter), це ті ігри, де гравець бачить дію через очі свого персонажа. У цьому жанрі дуже важливе точне прицілювання та реакція гравця на швидкі зміни ситуації. Найвідомішими прикладами таких ігор є Doom, Call of Duty, Battlefield, Half-Life.

Шутери третьої особи, або TPS (Third-Person Shooter), це жанр, в якому гравець спостерігає за персонажем ззовні, а не через його очі. У цьому жанрі гра часто більш зосереджена на сюжеті та настрої, а не на дії. Найвідомішими представниками цього жанру є серії Gears of War, Uncharted та Max Payne.

Тактичні шутери, або Tactical Shooter, це жанр, де гравець має враховувати різні фактори, такі як покриття та позицію ворога, а також ефективність та доступність зброї. У таких іграх важлива комунікація та координація між гравцями, що робить їх більш реалістичними та складними у геймплеї. До прикладів таких ігор належать Rainbow Six, Ghost Recon, SWAT.

Аркадні шутери, або Arcade Shooter, це жанр, в якому геймплей більше орієнтований на розвагу та швидкість дії, а не на реалістичну симуляцію. У таких іграх зазвичай мало фізіки та розуміння взаємодії з оточенням, але дозволяє гравцю використовувати різноманітну зброю та знищувати численних ворогів. Такі ігри можуть мати більш насичену графіку та музику, що підсилює враження від проходження. До таких ігор можна віднести такі класичні твори, як "Space Invaders" та "Galaga". Іншими популярними жанрами шутерів є тактичні шутери, в яких гравець повинен працювати в команді та використовувати стратегічні навички, та мультиплеєрні шутери, в яких гравці змагаються один з одним в різних режимах гри, таких як "Capture the Flag" та "Team Deathmatch". Кожен жанр шутерів має свої унікальні особливості та специфіку, але всі вони дозволяють гравцеві насолоджуватися процесом стрільби та використовувати різноманітні тактики для перемоги.

До основних елементів геймплею шутерів можна віднести такі фактори, як

стріляння, бігання та приховування за укриттями. Ігри жанру шутер часто пропонують різноманітні режими гри, такі як мультиплеєр, сюжетний режим та кооперативний режим.

У сюжетному режимі гравець зазвичай грає роль головного героя, який повинен виконувати певні місії та завдання, вбиваючи ворогів на шляху. Гра може мати різні налаштування, такі як місце дії, часовий період або рівень складності. У цьому режимі гравець може знайти різноманітні зброї, аптечки, броню та інші предмети, які допоможуть йому у виконанні завдань.

У мультиплеєрному режимі гравці можуть грати один проти одного або у команді. Гра може мати різні режими мультиплеєру, такі як «Захоплення прапора», «Вільна для всіх» або «Командний матч». У цьому режимі гравці можуть змагатися між собою та використовувати різні стратегії, зброю та інші предмети, які допоможуть їм перемогти в сутичці.

У кооперативному режимі гравці можуть грати разом, щоб виконувати різні місії та завдання. Гра може мати різні режими кооперації, такі як «Виживання», «Схованки» або «Пошук і знищення». У цьому режимі гравці можуть працювати разом та використовувати різні стратегії, щоб виконати завдання та перемогти ворогів.

Жанр шутер зазвичай вимагає від гравця добре розвинутих навичок стрільби та грі у команді, особливо в мультиплеєрному режимі. Головною метою гравця в шутерах є знищення ворожих персонажів, тварин чи машин, використовуючи для цього різноманітну зброю. Залежно від конкретної гри, можуть бути доступні різні види зброї, такі як пістолети, гранати, автоматичні гвинтівки, снайперські гвинтівки та багато іншого. Деякі шутери можуть мати елементи RPG, такі як можливість розвивати навички та здібності персонажа гравця, або різноманітні завдання та місії, які потрібно виконати, щоб продовжувати гру. Також можуть бути доступні різні режими гри, включаючи кампанію, мультиплеєр та кооператив. Крім того, деякі шутери можуть мати елементи екшену та пригод,

такі як можливість взаємодії з оточенням та різноманітними об'єктами, що збільшують іммерсивність гри. В цілому, шутери є дуже популярним жанром відеоігор і пропонують гравцям незабутні емоції та адреналін.

Однак, з появою онлайн-ігор та мультиплеєрних режимів шутерів стали включати і можливості для командної гри. Гравці можуть об’єднуватися в команди та виконувати різноманітні завдання, які часто вимагають тактичного мислення та співпраці.

Ще однією характеристикою шутерів є наявність різних видів зброї, яка може бути здобута в процесі гри або куплена за внутрішню валюту, отриману за успішне виконання завдань. Зброя може відрізнятися за вагою, точністю стрільби, швидкістю перезарядки та іншими параметрами, що дозволяє гравцеві вибирати найбільш оптимальну для себе модель.

Також, шутери можуть включати в себе елементи RPG (рольових ігор), коли гравець може прокачувати свої навички та вдосконалювати свої здібності. Це може включати в себе збільшення міцності персонажа, збільшення точності стрільби та інші характеристики, які дозволяють гравцю бути більш ефективним на полі бою.

Одним з найбільш популярних жанрів шутерів є FPS (від англ. First Person Shooter), де той, хто грає, спостерігає за дією з першої особи. У цьому жанрі зазвичай відсутній елемент підйому на рівні, тому що гравцеві пропонується іммерсивний досвід, де він сам є персонажем гри. Іншими популярними жанрами є TPS (від англ. Third Person Shooter), де гравець спостерігає за персонажем зі сторони, та Rail Shooter, де користувач не може керувати рухом персонажа, а лише відстрілює з встановленого маршруту руху. В шутерах зазвичай використовуються різноманітні види зброї - від простих пістолетів і гранат до складних кібернетичних пристроїв і ракетних комплексів. Окрім того, головними ворогами в цьому жанрі є зазвичай військові загони, монстри, зомбі та інші ворожі створіння. Шутери також можуть бути поділені на піджанри, такі як survival horror, де гравець має боротися з надприродними силами,

та tactical shooter, де акцент робиться на тактичному плануванні і координації команди. Шутери є одним з найбільш емоційно напружених жанрів відеоігор, які вимагають від гравця швидкість реакції, точність та стратегічне мислення.[3]

1.3 Розробка відеоігор

Розробка відеоігор - це складний і довготривалий процес, що вимагає від команди розробників не лише технічної грамотності, але й творчого мислення. Спочатку розробники обговорюють ідеї, складають концепцію та розробляють ігровий сценарій. Після цього починається робота над графікою та музикою.

Історія розробки відеоігор починається ще в 1950-х роках, коли з'явилися перші електромеханічні ігри, наприклад, "Tennis for Two". Пізніше з'явилися перші комп'ютерні ігри, зокрема "Spacewar!", яку розробили в 1962 році. У 1970-х роках комп'ютерні ігри стали доступнішими, а в 1980-х з'явилися перші консолі. З тих пір розробка відеоігор почала розвиватися експоненційно, особливо в останні десятиліття.

Сьогодні процес розробки відеоігор включає в себе кілька етапів. На початку розробники визначають жанр ігри, ідею та концепцію. Далі починається розробка ігрового сценарію, який визначає сюжет, персонажів, ігровий світ та інші ключові елементи. Після цього починається робота над графікою та звуковим оформленням, розробка інтерфейсу та фізичної моделі. Останнім етапом є тестування ігри та її випуск на ринок.

Розробка відеоігор може бути досить складною, особливо коли розробники працюють над амбітними проектами, такими як відкриті світи або мультиплеєрні ігри. Однак це не заважає компаніям продовжувати роботу над інноваційними іграми, які дарують гравцям нові досвіди та емоції.

Розробка відеоігор розпочалася в далекому 1958 році, коли на екрані з'явилася перша відеогра Pong. З того часу індустрія відеоігор розвивалася

стрімко, з'являлися нові жанри, технології та інноваційні підходи до розробки ігор. Сьогодні розробка відеоігор - це складний технічний процес, що вимагає від команди розробників величезних зусиль, знань та досвіду.

У процесі розробки відеоігор, команди розробників зазвичай починають з формування концепції гри, вибору жанру, механік та естетики. Після цього розпочинається розробка геймплею, ігрового двигуна, візуальних ефектів та звукового оформлення. Розробники також займаються оптимізацією гри для різних платформ та пристроїв, тестуванням та налагодженням роботи ігри.

У сучасній індустрії відеоігор, розробники використовують різні методики та технології для поліпшення процесу розробки ігор. Наприклад, Agile-методологія дозволяє знизити час та затрати на розробку, а Scrum дозволяє командам розробників більш ефективно співпрацювати між собою.

Інноваційні технології також допомагають розробникам покращувати якість ігор та забезпечувати кращий іммерсивну практику для користувачів. Наприклад, використання віртуальної реальності та доповненої реальності дозволяє створювати більш реалістичні ігрові світи та дозволяє гравцям взаємодіяти з ними на більш особистому рівні. Технології ШІ також дають змогу розробникам формувати більш інтелектуальних та складних ворогів, що робить ігри більш цікавими та викликаючими для гравців.

Процес розробки відеоігор зазвичай складається з кількох етапів, починаючи від концепції та прототипування гри до програмування та тестування. Команда розробників зазвичай складається з дизайнерів, програмістів, художників, звукоінженерів та інших фахівців, які працюють разом, щоб створити ігровий продукт.

У світі відеоігор з'явилися багато легендарних ігор, які залишаються популярними і донині. Наприклад, серія ігор "Mario" від компанії Nintendo, що була випущена в 1985 році, стала символом геймінгу та є однією з найпопулярніших серій відеоігор у світі. Іншим прикладом є ігри серії "Final

Fantasy" від Square Enix, які з'явилися в 1987 році та стали популярними завдяки своїй унікальній графіці та глибокому сюжету.

У залежності від типу гри та її складності, процес розробки відеоігор може займати від декількох місяців до декількох років. Ігри можуть бути розроблені від ідеї або концепту до повноцінного продукту. Спочатку розробники визначають жанр ігри, її механіку та історію. Потім вони приступають до створення прототипу гри, який дозволяє протестувати ідеї та функції. Далі розробники створюють графічний дизайн, звукові ефекти, програмне забезпечення та інші компоненти гри.

Розробка відеоігор також включає в себе важливий етап тестування. Це може бути як внутрішнє тестування, коли розробники самі перевіряють гру на наявність помилок та багів, так і залучення зовнішніх тестерів, що дозволяє зібрати відгуки та відгуки гравців щодо гри.

Одним з ключових факторів успіху відеоігор є геймплей, який повинен бути цікавим та забезпечувати гравцеві відчуття задоволення. Також важливою є графіка та звукові ефекти, які допомагають створити іммерсивний світ та поглинути гравця в гру.

Загалом, розробка відеоігор - це складний процес, який вимагає багато творчого та технічного напруження. Але успіх гри та її популярність серед гравців можуть бути дуже задоволенними для розробників та вигідними для компаній.

Інді розробка

Інді розробка відеоігор є досить популярною в галузі геймдеву, особливо в останні роки. Це означає, що гра розробляється невеликою командою або навіть одним розробником, а не великою компанією.

Інді розробка має свої коріння ще з 1970-х років, коли були створені перші варіації відеоігор на персональних комп'ютерах. Однак, на сьогоднішній день, інді-геймінг можна вважати досить молодим рухом, який зародився в 2000-х роках.

Основні принципи інді розробки полягають у тому, що розробники мають повну свободу у створенні своїх ігор, від ідеї до реалізації, включаючи ігровий дизайн, графіку та звуковий дизайн. Більшість інді-розробників мають велику любов до гри та бажання створювати щось унікальне та незабутнє для гравців.

Одним з піонерів інді-геймінгу є компанія Id Software, яка випустила такі ігри, як "Doom" та "Quake" у 1990-х роках. Вони були одними з перших, хто використовував 3D-графіку та мультиплеєрні режими в своїх іграх.

З появою платформи Steam в 2003 році інді-геймінг почав активно розвиватись, оскільки він надавав незалежним розробникам можливість дистрибуції своїх ігор через цю платформу. Це дало змогу розробникам створювати ігри на свій смак та продавати їх напряму гравцям.

Зараз інді-розробники продовжують дивувати світ своїми ідеями та незалежними проектами. Інді-геймінг став досить популярним і вже не є чимось новим в індустрії відеоігор. Проте, його початки відносяться до 70-х років, коли розробники самостійно створювали та розповсюджували свої ігри без допомоги великих компаній.

Зараз інді-розробники можуть використовувати безкоштовні ресурси, які не були доступні в минулому, такі як інтернет, відкриті програмні засоби та розробницькі платформи. Це дозволяє їм розвивати свої ідеї та створювати незалежні проекти.

Інді-розробники можуть створювати ігри в різних жанрах, від платформерів та різноманітних казуальних ігор до складних рольових ігор з відкритим світом. Однією з привілегій інді-розробників являється те, що вони можуть створювати нішеві ігри, які не знайдуть собі місце в більш комерційних проектах.

Одним з успішних прикладів незалежної інді-гри є "Minecraft", яка була створена Маркусом Перссоном в 2009 році. Гра стала однією з найбільш продаваних ігор у світі, з продажів понад 200 мільйонів копій, та викликала багато наслідувань та клонів.

Іншим успішним проектом є "Undertale" створена Тобі Фоксом в 2015 році. Гра стала популярною завдяки своїй унікальній механіці та глибокому сюжету, а також захопила серця гравців по всьому світу.

Однак, розробка інді-ігор може бути складною і залежати від фінансових обмежень. Багато інді- розробників відчувають нестачу коштів на фінішні роботи над проектом, маркетинг та рекламу, що може спричинити провал проекту. Одним з відомих прикладів такої ситуації є інді-гра "Fez" від розробника Поля Фішера. Розробка гри зайняла понад 5 років та зіткнулася з проблемами фінансування, але після випуску отримала високі оцінки критиків та популярність серед гравців.

Щоб підтримати інді-розробників, на ринку існують різні платформи, такі як Steam та GOG, які дозволяють продавати та рекламувати інді-ігри. Крім того, існують інструменти, такі як Kickstarter та IndieGoGo, які дозволяють збирати кошти від фанатів та підтримувати проекти інді-розробників.

Інді-розробка відеоігор продовжує розвиватися, і вона вважається однією з найбільш творчих та інноваційних галузей в галузі геймдеву. Інді-ігри можуть мати невеликий бюджет, але завжди привертають увагу гравців своїми унікальними концепціями, стилем та геймплеєм. Багато інді-розробників продовжують створювати ігри, які перевертають звичні ігрові жанри та забезпечують нові ігрові досвіди для гравців.[4]

2. ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Огляд ігрового рушія

Unity є мультиплатформним ігровим рушієм, розробленим компанією Unity Technologies. Уперше він був продемонстрований у 2005 році на AWDC, організованому компанією Apple, як рушій ексклюзивно для Mac OS X. Але згодом функціонал Unity став доступним на 25 платформах, включаючи Windows, Linux, iOS та Android, а також для веб-середовища. Unity можна застосовувати для формування різноманітних ігор, включаючи 2D та 3D ігри, ігри VR та AR, а також для моделювання об'єктів для будівництва, машинобудування, проектування архітектури, і багато іншого, включаючи симуляцію фізики.

Unity - це інструмент для створення відеоігор, розроблений компанією Unity Technologies з метою забезпечити доступність для розробників-початківців. Перші покази рушія відбулися на конференції AWDC, яку проводила компанія Apple в 2005 році. Оригінально рушій задумувався як ексклюзивний для платформи Mac OS X, але згодом став доступним для 25 платформ, включаючи Windows, Linux, iOS та Android, а також для веб-середовища. Unity підтримує розробку як 2D, так і 3D ігор, а також ігор у віртуальній та доповненій реальності. Крім цього, рушій може бути використаний для моделювання об'єктів, симуляції фізики, будівництва та проектування архітектури. У 2006 році Unity отримав нагороду "Найкраще використання графіки Mac OS X" від компанії Apple..

У 2007 році була випущена наступна версія рушія Unity - 2.0, яка мала значно більше функцій - близько 50. Оптимізація 3D обробника дозволила створювати деталізованіші 3D-середовища, а також підвищити якість завантажування спрямованого світла та тіней у реальному часі. Також ця версія рушія мала нову

організацію асинхронного коду, що збільшило кількість операцій на секунду, котрі можуть виконуватися одночасно.

Unity 3.0 був важливим оновленням, яке внесло численні покращення в обробку 3D-середовища. Зокрема, були вдосконалені методи відображення карт тіней та відкладеного завантаження, додані нові функції для легшого створення та редагування дерев, а також покращена робота з UV-картами, що дозволило ефективніше організувати 3D-об'єкти на 2D-зображенні..

На початку 2012 року було зареєстровано більше мільйона щоденних користувачів Unity. Опитування, проведене журналом Game Developer в тому ж році, показало, що Unity є найпопулярнішим двигуном для створення ігор на мобільні пристрої. У вересні 2012 року було випущено нову версію двигуна - Unity 4, яка підтримувала DirectX 11 та Flash, а також мала передрелізну версію для Linux.

У 2013 році Facebook додав підтримку рушія Unity на свій сайт. Це дозволило відслідковувати рекламні кампанії та аналізувати дії гравців у іграх. У 2016 році Facebook випустив нову платформу для створення ігор на ПК з використанням рушія Unity. Це значно спростило та прискорило процес публікації ігор на Facebook.

У 2015 році журнал The Verge висловив думку, що Unity є інноваційним рушієм для розробки ігор, оскільки спрощує процес створення ігор. Випущена п'ята версія двигуна дозволила публікувати ігри для веб-браузерів завдяки підтримці WebGL. Крім того, були вдосконалення в освітленні та обробці аудіо, а також підтримка платформи Nintendo Switch, API Vulkan, 4K-відео та створення 360-огляду для пристроїв віртуальної реальності. Незважаючи на критику, що виникла, генеральний директор компанії заявив, що низька якість ігор не є проблемою рушія, а лише побічним ефектом його популярності..

Починаючи з 2016 року, Unity почала випускати нові версії зі значними

змінами в порівнянні з попередніми версіями. Наступною версією після 5.6 стала Unity 2017, в якій з'явилися інструменти графіки в реальному часі та звіти про ефективність. Крім того, розробники Unity Technologies визначили кінематографічний напрямок для рушія, і додали таймлайн, що полегшувало створення анімацій, а також розширення Cinemachine для простішої роботи з камерами. Такі зміни були важливим кроком в розвитку рушія, який зберіг свою популярність серед розробників ігор.

У 2018 році Unity запровадили налаштування відображення графіки, яке залежало від платформи. Це дозволяло створювати більш точні і реалістичні графічні ефекти для різних пристроїв. Крім того, в цій версії Unity були представлені нові інструменти машинного навчання, які дозволяли розробникам створювати інтелектуальних агентів для ігор. Ці інструменти дозволяли швидко навчити ігрових персонажів виконувати складні завдання, забезпечували автоматизоване вирішення різних проблем та поліпшували продуктивність гри.

В 2019 році Unity отримав підтримку мови Wolfram, що дозволило розробникам обчислювати складні математичні формули відразу в середовищі рушія без необхідності використання зовнішніх програм. Це значно спрощує процес створення ігор з математичними складовими, такими як фізична модель, ефекти руйнування та інші.

В 2020 році компанія Unity представила новий продукт, який називається MARS. Це набір інструментів, який дозволяє створювати ігри у просторі доповненої реальності. MARS є потужним інструментом, який допомагає розробникам створювати вражаючі ігри у доповненій реальності, використовуючи сучасні технології. MARS надає користувачам можливість швидко і легко створювати та тестувати ігри у просторі доповненої реальності з використанням різних пристроїв, таких як смартфони і планшети.

На даний момент актуальною є стабільна версія Unity 2021.3 LTS, яка була випущена 19 травня 2022 року. Позначення LTS вказує на те, що ця версія

підтримуватиметься протягом двох років.

Unity є потужним інструментом для розробки ігор і анімації, який підтримує створення як 2D, так і 3D контенту. Для написання логіки, можна використовувати мову програмування C#, яка є єдиною мовою підтримуваною Unity. Раніше, також підтримувались мови Boo та UnityScript, але це припинили з версії Unity 5 та Unity 2017.1 відповідно, повністю перейшовши на C#. Крім того, існують плагіни, які дозволяють створювати логіку за допомогою візуального скриптингу, без написання коду. [3]

Unity - це інструмент для розробки ігор, який надає розробникам можливості створення як 2D, так і 3D ігор. Для 2D графіки Unity дозволяє використовувати спрайти, а для 3D існують різноманітні технології зменшення розміру текстур, карти текстур, а також можливість налаштування роздільної здатності залежно від платформи. У Unity також можна додати ефекти вибухів, відбиття від поверхонь, ефект паралаксу, SSAO, динамічні тіні, постпроцесинг - як для 2D, так і для 3D ігор.

За даними статистики за 2018 рік, більше половини ігор для мобільних пристроїв та понад половина ігор у віртуальній та доповненій реальності були створені на платформі Unity. Деякі платформи навіть досягли показника в 90%. Окрім цього, Unity підтримує машинне навчання та бібліотеку Google TensorFlow, а також використовується для розробки автопілотів.

Існує перелік застарілих технологій, які більше не підтримуються Unity. Один із таких прикладів – розширення до веб-браузера, котре використовувалося для запуску Web-ігор. На його місце прийшов WebGL, який компілює код у JavaScript у дві стадії: спочатку з C# на C++, а потім вже на JavaScript.

Unity та Nintendo мають співпрацю і постачають набір для розробки програмного забезпечення (SDK) для розробки відеоігор на Wii U, який є частиною кожної ліцензії розробника для Wii U. Це дозволяє розробникам створювати ігри для цієї платформи за допомогою Unity та розширити своє

аудиторію.[11]

2.2 Огляд мови програмування

C# є мовою програмування, яка підтримує багато парадигм і забезпечує строгу типізацію, що означає, що тип змінної не може змінюватися під час виконання програми. Ця мова була розроблена Андерсом Хейлсбергом у 2000 році для проекту .NET на Windows і пізніше була перенесена на інші платформи завдяки проекту Mono, котрий має відкритий код та орієнтований на підтримку CLI. C# підтримує такі парадигми програмування, як функціональна, імперативна та об'єктно-орієнтована..

На сьогоднішній день, найсвіжіша версія мови програмування C# - це C# 10, яка була випущена для .NET 6.0.

На початку розробки .NET фреймворку, бібліотеки були написані на SMC (Simple Managed C) – системі компіляторів для керованого коду. У 1999 році, Андерс Хейлсберг та його команда створили COOL "C-like Object Oriented Language", який пізніше був перейменований в C# на конференції Professional Developer Conference в 2000 році. Сам Хейлсберг, який раніше займався TurboPascal та Visual J++, вказав на проблеми у наявних тоді мовах програмування, таких як C++, Java, та Smalltalk, що призвело до розробки C# та CLR. Пізніше, мова програмування C# продовжувала розвиватись, а остання версія на даний момент – C# 10, була випущена для .NET 6.0.

Джеймс Гослінг, відомий як творець мови програмування Java, на початку вважав C# "імітацією Java", проте пізніше він додав, що C# має меншу надійність, продуктивність та безпеку порівняно з Java. Крім того, Клаус Крефт та Анжеліка висловили думку, що Java та C# настільки схожі, що вони майже ідентичні, але це не змінило того, як програмісти пишуть код, і обидві мови позичають у одна одній особливості. Але Андерс Хейлсберг, який розробив мову C#, вважає, що вона

більш схожа за дизайном на мову програмування C++.

Після випуску C# 2.0 у листопаді 2005 року, мови програмування C# та Java розійшлися у різних напрямках, тому вони значно відрізняються одна від одної. Однією з найбільших відмінностей є використання дженеріків, де C# використовує реіфікацію, що дозволяє використовувати дженерік клас як будь-який інший клас, тоді як у Java реалізовано параметризацію типів. Крім того, C# додав до свого арсеналу функціональні особливості, які дозволяють використовувати замикання та методи-розширення, а також анонімні типи, які дозволяють не створювати визначення класу. LINQ також був доданий до мови, що дозволяє зменшувати кількість повторюваного коду та використовувати функціональне програмування.

C# був спроектований з урахуванням універсальності та сумісності зі збіркою спільної мовної інфраструктури (CLI), на якій він базується. Більшість вбудованих типів мови відповідають типам, що існують у CLI. Однак опис мови не обмежує можливості компілятора створювати машинний код з коду C#, як це можна зробити в Fortran або C++.

Мова програмування C# дозволяє користувачам оголошувати змінні строго типізованими ключовим словом var. Також, для створення масивів, можна використовувати ключове слово new, яке дозволяє компілятору автоматично визначити тип змінних. Це робить процес написання коду більш простим та швидким, особливо при роботі з більш складними структурами даних.

Мова програмування C# має вбудовану підтримку строгого булевого типу даних, що дозволяє використовувати тільки вирази, які повертають оператор true. Відмінності в порівнянні з мовою C++ полягають у тому, що у C++ булевий тип може легко конвертуватися між типом цілих чисел. У свою чергу, C# не дозволяє такої можливості, тому програмісти змушені використовувати лише суто булевий тип даних.

C# забезпечує більшу безпеку при конвертації типів даних порівняно з C++.

В C# єдиним неявним перетворенням, яке дозволяється, є безпечне перетворення, наприклад, конвертація чисел з int до long, оскільки long має більший діапазон чисел. Це правило діє під час компіляції, виконання (JIT) та під час виконання коду. Крім того, C# не дозволяє перетворень між булевими та числовими типами даних і вимагає явного визначення будь-якої небезпечної взаємодії з кодом, наприклад, за допомогою safe і unsafe блоків.

Ще однією цікавою можливістю C# є підтримка явного визначення коваріантних та контраваріантних перетворень типів та їх підтипів у дженеріках. Це дає можливість більш гнучко працювати з типами та забезпечує безпеку при використанні дженеріків. У відміну від C++, де це може бути просто семантичним нюансом типів, що повертають віртуальні методи, в C# це підтримується напряму і дозволяє ефективніше використовувати дженеріки у проекті.

В C# немає поняття глобальних змінних та функцій, оскільки кожен метод та член типу повинен бути визначений всередині класу. Замість цього, статичні члени публічних класів можуть забезпечити схожий функціонал. У C# також не можна перекривати змінні блоку вище з локальними змінними. Це відрізняється від мов C та C++.

В C#, можна використовувати строгі вказівники на функції, використовуючи ключове слово delegate, яке дає можливість створювати відносини типу publisher-subscriber. Це означає, що виклик методу може спричинити виклик інших методів з інших класів. Крім того, ключове слово event дозволяє реалізувати шаблон спостерігача або підписки на подію, коли методи викликаються відповідно до відбування певної події (яка визначена як член event).

C# має вбудований механізм автоматичного управління пам'яттю, що дозволяє прибирати зайві об'єкти безпосередньо в процесі виконання програми. Це досягається завдяки прибиральнику сміття, який виконується одночасно з основною програмою і відповідає за звільнення пам'яті від об'єктів, які більше не потрібні. Це зменшує навантаження на розробника, оскільки він не потребується

звільняти пам'ять вручну.

У C# заборонено багатократне наслідування класів, що означає, що клас може мати лише один базовий клас. Проте, клас може реалізувати кілька інтерфейсів, які дозволяють забезпечити поліморфізм та розширення функціональності. Це спеціальне рішення для забезпечення простоти та зменшення складності архітектури програм, що розробляються для середовища .NET Framework. На відміну від C++, де можливе багатократне наслідування класів, але при цьому можуть виникати проблеми з конфліктами імен та залежностями між класами. Крім того, в C# підтримується перевантаження операторів, що дозволяє зручно працювати з об'єктами, тоді як в Java ця можливість відсутня.

LINQ - це інструмент, котрий дозволяє розробникам написати запити до даних безпосередньо в коді C#. Оскільки LINQ входить до складу .NET Framework, він може працювати з будь-якими джерелами даних, які підтримуються цією платформою, такі як XML, ADO.NET та Entity Framework. Використання LINQ дозволяє розробникам більш зрозуміло записувати запити до даних та зберігати їх у вигляді змінних, що полегшує подальшу обробку цих даних. Крім того, інтегроване середовище розробки може надавати підказки та автодоповнення для LINQ-запитів, що полегшує написання коду. Завдяки можливості фільтрування даних і маніпулювання ними за допомогою LINQ, розробники можуть легко обробляти та аналізувати великі обсяги даних.

З усіх доступних мов програмування розробники вибрали C# для даного проекту, оскільки мова має багато корисних функцій та можливостей, які необхідні для розробки проекту. Крім того, C# підтримує багато операційних систем та має простий синтаксис, що робить її легкою для вивчення і розуміння.

Загалом, C# є ідеальним вибором для даної розробки з-за його функціоналу, який дозволяє легко реалізувати основні принципи ООП, а також підтримки для платформи Unity. [14]

2.3 Огляд середовища розробки

Visual Studio - це інтегроване середовище розробки (IDE) від компанії Microsoft, яке використовується для створення програм для різних платформ та веб-додатків. Воно надає можливість розробляти програмне забезпечення для Windows, використовуючи різні технології Microsoft, такі як WinForms, WPF, Windows Store, Microsoft Silverlight та інші. Visual Studio має вбудований редактор коду, який забезпечує розумне сприйняття коду та його переробку. Також він містить конструктори графічних та веб-інтерфейсів, аналізатор коду та інші інструменти для розробки програмного забезпечення. Завдяки плагінам, можна додати новий функціонал, такий як підтримка інших мов програмування та систем керування версіями. Visual Studio підтримує 36 мов програмування, що дозволяє розробникам працювати з будь-якою мовою програмування. Версія Community доступна безкоштовно для всіх користувачів.

Один із принципів спільноти Visual Studio полягає у тому, що це повнофункціональний інструмент для розробників, який доступний для студентів, відкритого коду та окремих користувачів. Нова версія Visual Studio - 2022 - уже підтримується. На початку свого становлення Visual Studio не мала жодних інструментів для розробки, але за допомогою VSPackage-файлів та програм-інсталятора можна легко встановити все необхідне. Оскільки сама IDE є платформою, вона забезпечує взаємодію з різними сервісами, такими як SVsSolution для роботи з рішеннями та проектами, SVsUIShell для роботи з вікнами та інтерфейсом користувача та SVsShell для реєстрації пакетів VSPackage-функціоналу. Всі інші редактори, дизайнери та інструменти надаються як окремі VSPackage-пакети. Visual Studio використовує Component Object Model для доступу до VSPackage. SDK для Visual Studio теж має MPF, яка дозволяє кодувати пакети довільною мовою, сумісною з CLI. Ці служби створюють основу

для розширень, які додають новий функціонал до Visual Studio IDE. Підтримка додаткових мов програмування здійснюється за допомогою VSPackage-функціоналу, відомого як Language Service. Цей сервіс розкриває різні інтерфейси для створення VSPackage-розширень, які додають різний новий функціонал. Сервіс може змінювати кольори відображення підсвічування синтаксису, допомагати з автоматичним завершенням кодування, перевіркою правильності написання дужок та відображенням підказок про параметри та методи. Для створення нових модулів сервісу мов для керованого коду можна використовувати MPF-обєкти.

MPF-об'єкти (англ. "Modular Programming Framework") є одним із способів розширення функціональності програми за допомогою модулів. Ці об'єкти дозволяють створювати модулі з визначеними інтерфейсами, що можуть бути використані у програмі за допомогою спеціального механізму, що забезпечує їх інтеграцію.

MPF-об'єкти можуть бути написані на різних мовах програмування, але для керованого коду більшість рекомендують використовувати мови, які підтримують механізм взаємодії зі збирачем мусору. Наприклад, для створення MPF-об'єктів можна використовувати мови програмування такі як Python, Java або C#.

При створенні нових модулів сервісу мов для керованого коду, використання MPF-об'єктів дозволяє ефективно організувати код і забезпечити його модульність. Крім того, цей підхід дозволяє зменшити залежності між різними частинами програми і робить їх більш переносимими між різними платформами і операційними системами.

Загалом, використання MPF-об'єктів є корисним інструментом для розширення функціональності сервісу мов для керованого коду, який дозволяє створювати ефективні та модульні рішення.

При виборі середовища розробки було враховано кілька критеріїв, зокрема

наявність вбудованої підтримки мови C# та зручний і простий інтерфейс. Ідеально відповідаючим всім вимогам була саме ця IDE, тому її і було обрано для розробки проекту. [13]

3. ПРОЕКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГОПРОДУКТУ

3.1 Розробка діаграми класів продукту

У сфері розробки програмного забезпечення існує статична діаграма, відома як UML діаграма класів. Ця діаграма використовується для відображення та опису структури системи, включаючи класи, їх властивості, методи і взаємозв'язки між об'єктами системи.

Діаграми класів є важливими елементами об'єктно-орієнтованого моделювання, що використовуються для побудови структури проекту та перетворення моделей на програмний код. Вони також дозволяють моделювати дані. Класи в цих діаграмах виступають як основні компоненти системи і відображають взаємодії між ними.

Діаграми класів відображаються у вигляді вікон, які містять три розділи:

- Верхній розділ містить назву класу, яка відображається жирним шрифтом та розміщується по центру вікна. Перша літера кожного слова в назві класу велика.
- Середній розділ містить атрибути класу, які вирівнюються зліва відносно вікна. Назви атрибутів записуються з маленької літери
- Нижній розділ містить методи класу. Вони також вирівнюються зліва. Назви методів записуються з маленьких літер.

Під час створення системи важливо визначити взаємозв'язки між багатьма класами шляхом утворення схеми класів. Ця схема допомагає зрозуміти структуру системи та взаємини між класами. Крім того, класи можуть бути розподілені на підкласи для подальшої деталізації та організації системи.

Залежність - це взаємозв'язок між елементами схеми, де один елемент залежить від іншого. Якщо змінюється перший елемент, це призводить до змін у

другому елементі. Залежності можна показати на схемі за допомогою пунктирних ліній з незаповненими стрілками, які ведуть від замовника до постачальника.

Крім того, можна розширити ці діаграми класів за допомогою діаграм станів UML. Діаграми станів використовуються для моделювання поведінки об'єктів та відображення їх різних станів і переходів між ними. Це дозволяє краще розуміти, як об'єкт поводиться в різних умовах та як змінюється його стан залежно від подій

У програмному моделюванні асоціація визначається як зв'язок між об'єктами. Якщо асоціація відбувається між двома об'єктами, то її називають бінарною асоціацією. Ця асоціація може бути відображена у вигляді лінії, що з'єднує ці об'єкти. Важливо відзначити, що асоціація може з'єднувати будь-яку кількість класів. Якщо асоціація включає три класи, то вона називається потрійною. Крім того, асоціація може мати ім'я, а кожен з кінців може мати свою роль, власність або інші атрибути, які допомагають уточнити її характеристики.

У сфері програмного моделювання можна виділити чотири основних типи асоціацій: двонаправлені, однонаправлені, зворотні та агрегатні. Двонаправлені асоціації означають взаємне зв'язування між об'єктами, де кожен з них може виконувати роль і займати активну позицію. Однонаправлені асоціації передбачають, що зв'язок між об'єктами відбувається лише в одному напрямку, тобто один об'єкт вказує на інший, але не навпаки. Зворотні асоціації показують відносини, які відбуваються у протилежному напрямку до основної асоціації. Агрегатні асоціації вказують на тісний зв'язок між об'єктами, де один об'єкт є частиною або складовою іншого. Найбільш поширеними типами асоціацій є двонаправлені та однонаправлені, оскільки вони зустрічаються найчастіше у реалізації взаємозв'язків між об'єктами.

Агрегація в програмному моделюванні є підтипом відношення "має/has". Вона є більш конкретною формою асоціації і виражає частину або компоненту цілого. Агрегатні зв'язки також можуть мати атрибути та ролі, але вони обмежені

двома об'єктами одночасно. Також варто зазначити, що іноді на схемі не роблять розрізнення між агрегацією та асоціацією.

У мові UML асоціації відображаються за допомогою ліній, які з'єднують взаємопов'язані класи між собою. Крім простої лінії, кожна асоціація може мати додаткові позначення, які деталізують відношення між класами та їхніми взаємодіями. Це дозволяє визначити додаткові атрибути, ролі, або будь-яку іншу інформацію, необхідну для зрозуміння контексту асоціації. [16]

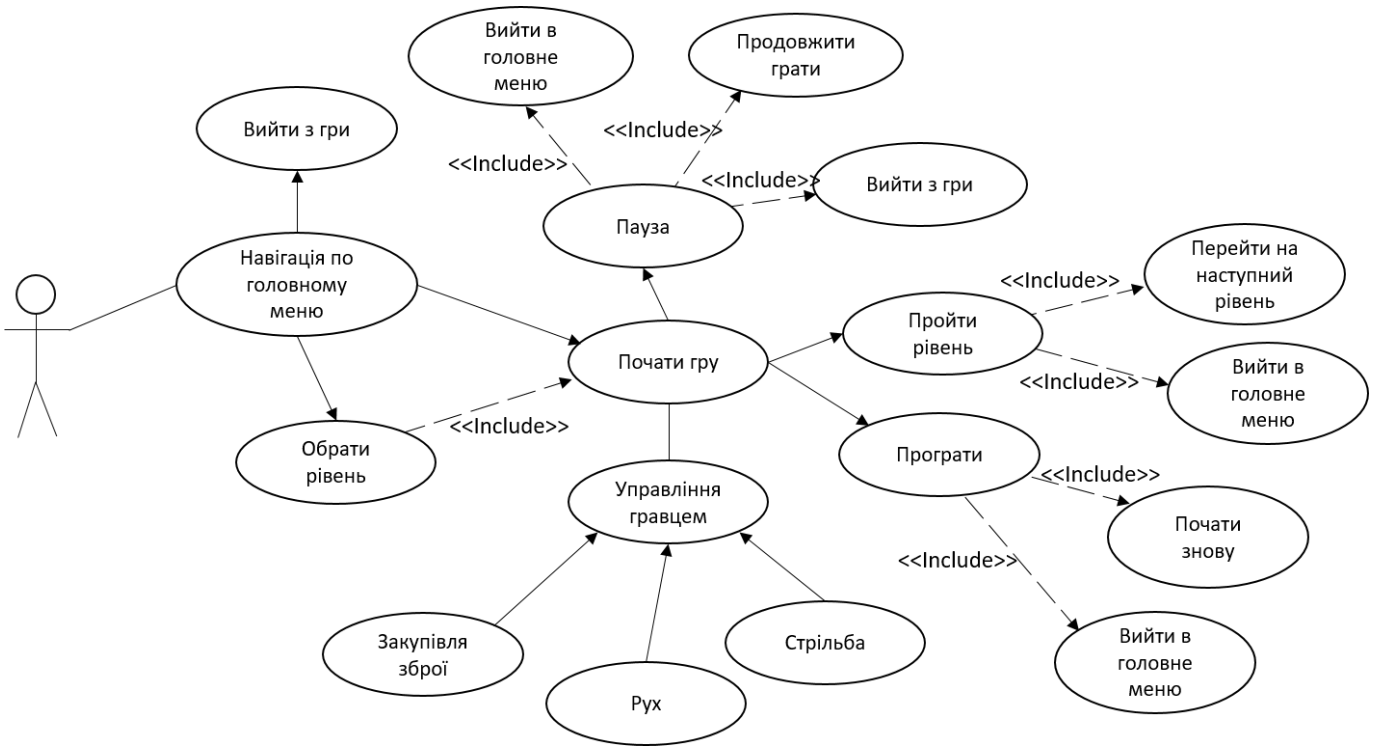


Рисунок 3.1 — Діаграма використання

3.2 Розробка основних механік

Відповідальність за процес проходження рівнів лежить на скрипті LevelControl.

Його код написано нижче.

using System.Collections.Generic;

```
using UnityEngine;
using TMPro;
using System;

namespace Assets.Scripts
{
   public class LevelControl : MonoBehaviour
   {
      [SerializeField] private TMP_Text _destroyedTargetsAndCountOfTargets;
      [SerializeField] private List<GameObject> _targetsNeetToKill;
      [SerializeField] private GameObject _menusPlate;
      [SerializeField] private GameObject _player;

      private int _destroyedTargets = 0;
      private int _allTargets;

      public event Action LevelDone;
      public event Action LevelFailed;

      private void Start()
      {
         _player.GetComponent<Health>().PlayerDead += ActivateLoosePanel;
         for (int i = 0; i < _targetsNeetToKill.Count; i++)
         {
            _targetsNeetToKill[i].GetComponent<IEnemieble>().TargetDestroyed +=
RefreshUI;
         }
         _allTargets = _targetsNeetToKill.Count;
         _destroyedTargetsAndCountOfTargets.text = _destroyedTargets + " / " +
_allTargets;
      }

      private void RefreshUI()
      {
         _destroyedTargets++;
         if(_destroyedTargets == _allTargets)
         {
            Invoke(nameof(ActivateWinPanel), 2f);
            LevelDone?.Invoke();
         }
         _destroyedTargetsAndCountOfTargets.text = _destroyedTargets + " / " +
```

```
_allTargets;
        }

        private void ActivateWinPanel()
        {
            _menusPlate.SetActive(true);
        }

        private void ActivateLoosePanel()
        {
            LevelFailed?.Invoke();
            _menusPlate.SetActive(true);
        }
    }
}
```

Однією з основних механік можна вважати механіку закупки перед місією. Вона відповідає за вивід всіх можливих видів зброї, які можна придбати під свою стратегію.

Його код написано нижче.

```
using UnityEngine;
using TMPro;

public class WeaponsShopPlate : MonoBehaviour
{
    [SerializeField] private TMP_Text _moneyText;
    [SerializeField] private int _currentMoney;
    [SerializeField] private GameObject _menu;
    [SerializeField] private GameObject _shopMenu;
    [SerializeField] private GameObject _gameScene;
    [SerializeField] private GameObject _warning;
    [SerializeField] private Animator _animator;
    private const string MONEY_TEXT = "Your money: ";

    public static int WeaponsCanBuy = 3;

    public int CurrentMoney
    {
```

```
        get => _currentMoney;
        set
        {
            _currentMoney = value;
        }
    }

    private void Start()
    {
        _moneyText.text = MONEY_TEXT + _currentMoney;
        WeaponsCanBuy = 3;
    }

    public void SetMoney(int money)
    {
        _moneyText.text = MONEY_TEXT + money;
    }

    public void PlayGame()
    {
        if (WeaponsCanBuy < 3)
        {
            _shopMenu.SetActive(false);
            _menu.SetActive(false);
            _gameScene.SetActive(true);
        }
        else if (WeaponsCanBuy == 3)
        {
            _warning.GetComponent<Animator>().Play("WarningAppear");
        }
    }
}
```

3.3 Розробка графічного інтерфейсу.

Стартує гра з головного меню, де можна почати рівень або вийти з гри (рис. 3.2).

Рисунок 3.2 — Головне меню

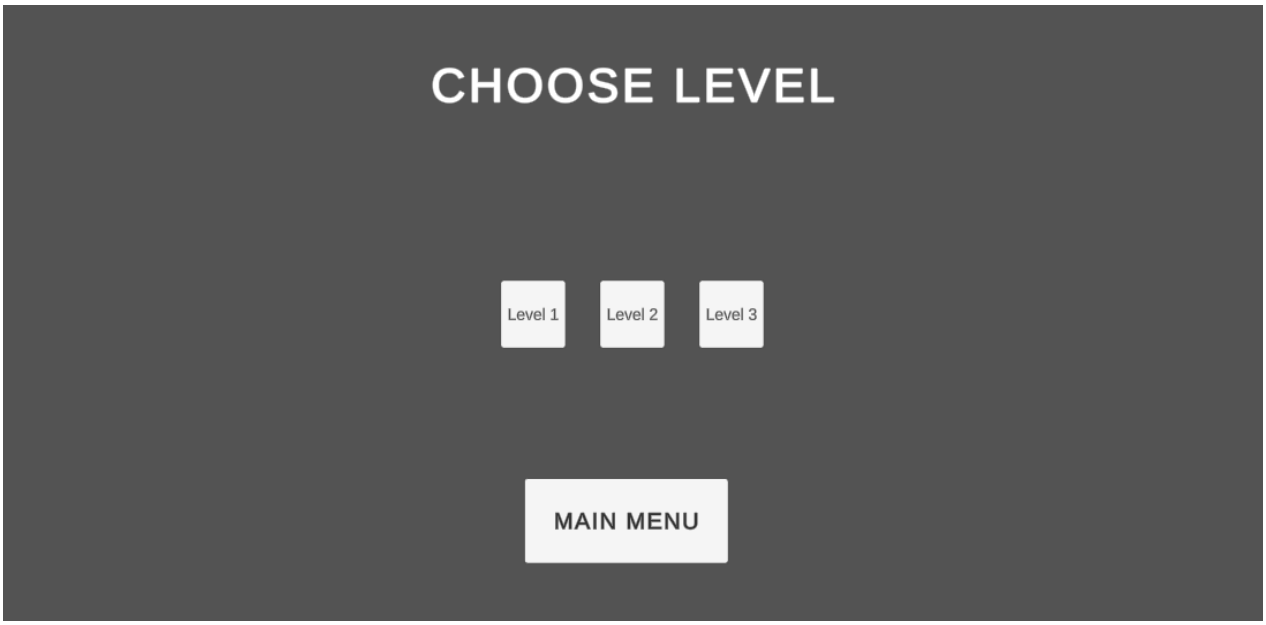


Рисунок 3.3 — Панель для вибору рівня

Коли гравець починає проходити рівень, спершу спливає вікно з передісторією місії та самим завданням. В цьому ж вікні гравець може ознайомитись з даними розвідки та зайти в магазин. Відштовхуючись від даних розвідки та від того скільки в нього грошей, гравець повинен купити хоча б один вид зброї та почати рівень.

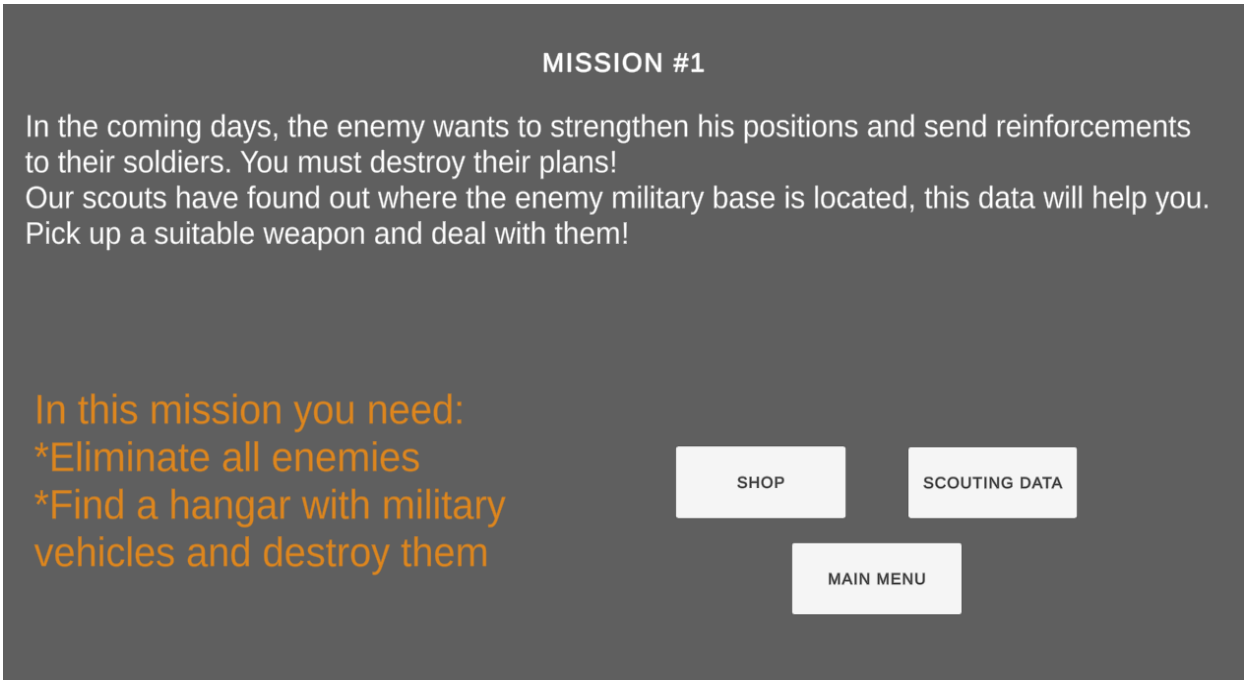


Рисунок 3.4 — Початкова панель рівня

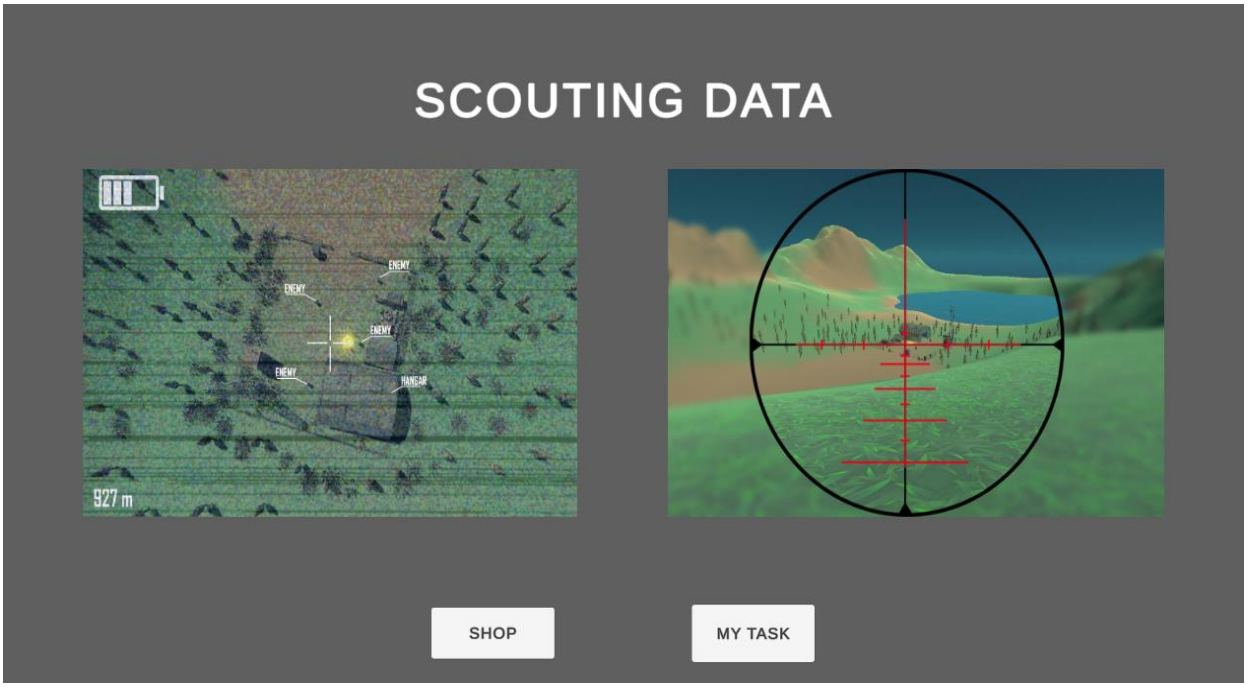


Рисунок 3.5 — Панель даних розвідки

Рисунок 3.6 — Панель магазина

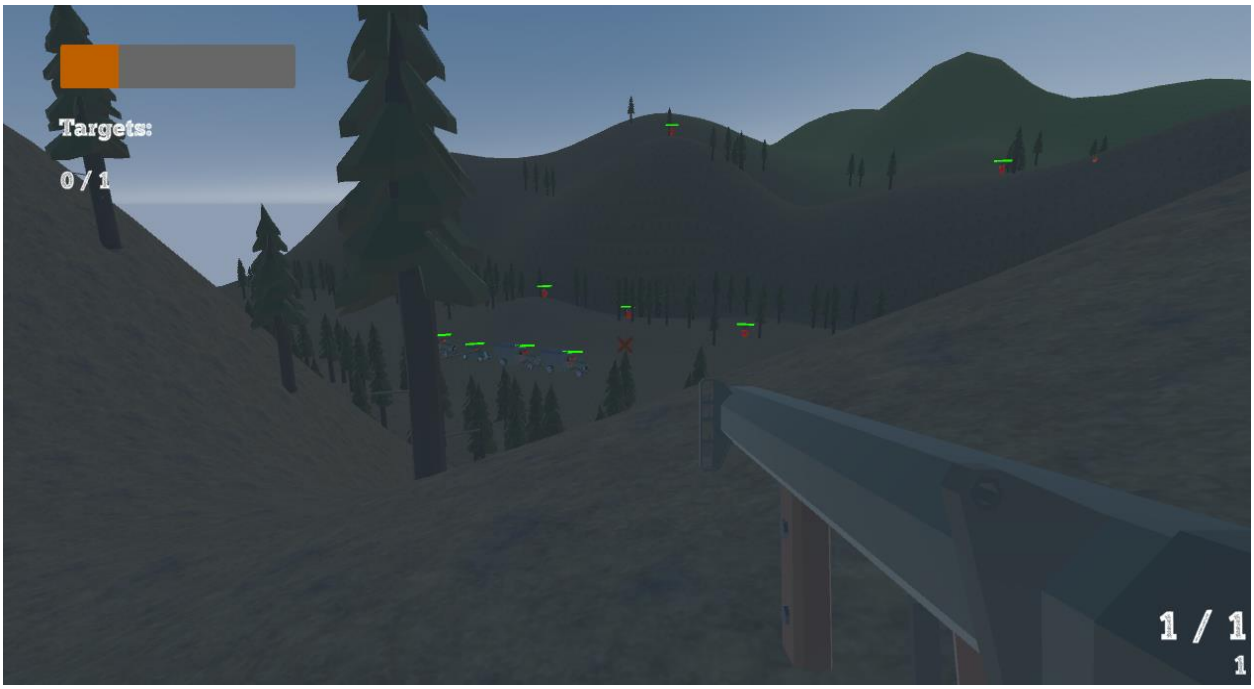


Рисунок 3.7 — Ігровий процес

3.4 Тестування

Тестування програмного забезпечення є процесом перевірки артефактів та

поведінки програми з метою виявлення помилок і підтвердження її відповідності вимогам. Цей процес також сприяє об'єктивному оцінюванню програмного забезпечення і виявленню потенційних ризиків з точки зору бізнесу. Методи тестування можуть включати, але не обмежуються такими аспектами:

- Аналіз вимог до програмного продукту з різних перспектив, таких як галузева, бізнес, ефективність використання, безпека, масштабованість.
- Перегляд архітектурних рішень та загального дизайну продукту з метою виявлення потенційних проблем та покращень.
- Співпраця з розробниками з метою поліпшення технік кодування, використання шаблонів проектування та написання ефективних тестів, що базуються на різних методах, таких як граничні умови.
- Виконання програм для перевірки правильності та відповідності поведінки програмного забезпечення заданим вимогам.
- Перевірка інфраструктури розгортання та виконання автоматизованих скриптів.

Тестування програмного забезпечення допомагає користувачам та спонсорам отримати об'єктивну оцінку якості програмного забезпечення та виявити потенційні ризики збоїв. У процесі тестування беруть участь методи моніторингу та спостереження, що дозволяють виявляти проблеми та покращувати процеси виробничої діяльності.

Проблеми та відмови в роботі програмного забезпечення

Помилки в програмному забезпеченні виникають в результаті наступного процесу: розробник створює код, який призводить до виникнення помилок або непередбачуваної поведінки у виконавчому коді. Якщо така помилка присутня, система може видавати неправильні результати, що може призвести до відмови.

Не всі баги обов'язково призводять до збоїв. Наприклад, помилки в неактивному коді ніколи не спричинять збоїв. Помилка, яка не виявила збоїв, може

спричинити збій при зміні середовища. Прикладами можуть бути інші програми, які виконуються на новому комп'ютері, зміна початкових даних або взаємодія з іншим програмним забезпеченням. Одна помилка може мати різноманітні симптоми збою.

Не всі неполадки програмного забезпечення є наслідком помилок у коді. Погано визначені або нерозгорнуті вимоги до ПЗ можуть призвести до серйозних проблем і великих складнощів. Це особливо стосується нефункціональних вимог, таких як безпека, тестованість, масштабованість, обслуговування та продуктивність. Нерозпізнані або неправильно сформульовані вимоги можуть стати причиною помилок, які будуть виявлені пізніше в процесі розробки та експлуатації програми. Тому важливо приділяти достатню увагу уточненню та деталізації вимог, щоб уникнути подібних проблем.

Динамічне, пасивне та статичне тестування

У тестуванні програмного забезпечення існує широкий спектр підходів. Статичне тестування, таке як огляди, списки to-do або перевірки, вимагає ручної перевірки коду без його виконання, в той час, як динамічне тестування, таке як написання тестового коду з визначеними випадками, виконується автоматично.

Мануальне тестування, яке часто включає коректуру, є неявним видом тестування. У цьому випадку розробники вручну перевіряють вихідний код на наявність помилок перед компіляцією, а компілятори роблять звіряють коректність синтаксису та взаємодію коду у статичному контексті. З другої сторони, автоматичне тестування відбувається під час виконання самої програми. Тестовий скрипт можна створити ще до повного завершення програми, щоб перевірити відокремлені логічні блоки коду та застосувати їх до окремих функцій або класів. Переважно для такого виконуються методи, такі як субституція замість елементів коду з необхідною логікою або здійснення з середовища налаштування.

Мануальне тестування виконується людиною, яка загально робить перевірку

коду шляхом запуску програми вручну та тестування її практичності. У такому випадку, той хто тестує - спостерігає за зовнішнім діянням програми та перевіряє, чи задовільняє вона результатам. З другої сторони, автоматичне тестування реалізовується за допомогою написаного коду, який автоматично вводить в дію програму та перевіряє її внутрішніє становище. Це дозволяє провести більш вичерпне тестування, вводити перевірку логічних аспектів коду та внутрішньої взаємодії між компонентами програми.

Пасивне тестування має значення перевірки діяння системи без ініціативної співдії з кодовою частиною або програмним забезпеченням. У такому виді тестування, ті хто тестують, не вводять тестові показники, а розглядають журнали системних приміток і накресленнь. Вони знаходять моделі та особливе діяння, яке допомагає ухвалювати рішення. Це охоплює перевірку на час реалізування в автономному режимі та розгляд журналу, щоб установити проблеми та підсумувати.

Дослідницький підхід

Дослідницький метод тестування є підходом до тестування програмного забезпечення, при якому одночасно проводиться дослідження продукту, проектування тесту та його негайне виконання. Термін "дослідницьке тестування" був введений Джемом Канером в 1984 році і описує цей тип тестування як "стиль тестування, що надає відповідальність окремому тестувальнику за постійне покращення якості своєї роботи, розглядаючи навчання пов'язане з тестуванням, проектування тестів, їх виконання та інтерпретацію результатів як взаємодіючі дії, що відбуваються паралельно протягом усього проекту".

«Коробковий» підхід

Методики випробування ПЗ можуть бути відокремлені на дві категорії: випробовування білого ящика та випробовування чорного ящика. Ці підходи формулюють перспективу, з якої людина, яка тестує, закладає тестові випадки. Зокрема, можна вживати гібридний захід, знаменитий як тестування сірого ящика,

в якому тести опрацьовуються на основі конкретних часток дизайну. Нині тестування сірого ящика стало більш поширеним, що призвело до розмивання "жорсткої" роздільної риски між випробуванням білого та чорного ящиків.[16]

ВИСНОВКИ

В результаті виконання бакалаврської роботи було виконано:

- Проведено аналіз предметної області
- Отримано чітке визначення розробки відеоігор
- Розглянуто концепцію шутера в контексті відеоігор
- Вивчено процес створення відеоігор
- Вибрано відповідний рушій для розробки
- Вибрано мову програмування для проекту
- Обрано відповідне середовище розробки
- Розроблено внутрішню структуру гри
- Розроблено графічний інтерфейс
- Реалізовано основні геймплейні механіки

У результаті виконання всіх поставлених завдань було успішно протестовано гру. При зваженні на всі вимоги та деталі, була розроблена повноцінна відеогра у жанрі шутер-стратегія.

Засновуючись на результаті проведеного тестування, можна стверджувати, що програма цілком функціональна і готова для реального використання.

Як перспективну ідею для подальшого розвитку можна розглядати розширення гри шляхом додавання багато різних рівнів з унікальним дизайном, нових ворогів і нову різноманітну зброю.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Відеогра [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https:/uk.wikipedia.org/wiki/Відеогра
2. Жанри відеоігор [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://uk.wikipedia.org/wiki/Жанри_відеоігор
3. The History of First-Person Shooters [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.pcmag.com/news/the-complete-history-of-first-person-shooters
4. What Is Game Development? [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.freecodecamp.org/news/what-is-game-development/
5. What is game development? Everything you need to know [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.pulsecollege.com/what-is-game-development-everything-you-need-to-know/
6. Schell J. The Art of Game Design / Schell J. - Florida : CRC Press, 2008. - 520p.
7. Левчук М.П. IV Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 07.04.2023р. ДУТ, М.Київ
8. Левчук М.П. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 20.04.2023р. ДУТ, М.Київ
9. Video Game Genres [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.hp.com/us-en/shop/tech-takes/video-game-genres
10. Electronic shooter game [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.britannica.com/topic/electronic-shooter-game
11. Unity [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу:

https://unity.com

12. The Best Gaming Engines You Should Consider for 2023 [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.incredibuild.com/blog/top-gaming-engines-you-should-consider
13. Visual Studio [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://visualstudio.microsoft.com
14. C# Language Documentation [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/
15. What is Microsoft Visual Studio? [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://softwarekeep.com/help-center/what-is-microsoft-visual-studio-where-can-i-download-it
16. Game Testing: Types & How to Test Mobile/Desktop Apps [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.guru99.com/game-testing-mobile-desktop-apps.html
17. Game Testing 101: Basic Tips and Strategies [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://starloopstudios.com/game-testing-101-tips-and-strategies/
18. UML Use Case Diagram Tutorial [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.lucidchart.com/pages/uml-use-case-diagram

ДОДАТОК А ПРЕЗЕНТАЦІЯ ДО ЗВІТУ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка 3D гри "The language of guns" жанру шутер-стратегія
з використанням ігрового двигуна Unity

Виконавець: студент 4 курсу
групи ПД-44
Левчук Микола Петрович
Керівник роботи:
доцент кафедри, доктор філософії Дібрівний Олесь Андрійович

Київ – 2023

1

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – розширення функціональних можливостей гри жанру «шутер-стратегія» шляхом додавання механік сучасної війни.
- **Об’єкт дослідження** – ігровий процес додатку в жанрі «шутер-стратегія».
- **Предмет дослідження** – програмне забезпечення для реалізації ігрового додатку в жанрі «шутер-стратегія».

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз сучасних програмних рішень, які використовуються для розробки ігрових додатків за допомогою платформи Unity.
2. Розглянути актуальність жанру «шутер-стратегія».
3. Розглянути програмні засоби для розробки комп'ютерної гри жанру «шутер-стратегія».
4. Ознайомитись з іграми аналогами в жанрі «шутер-стратегія».
5. На основі аналізу аналогічних ігор в цьому жанрі розробити вимоги до комп'ютерної гри.
6. Розробити гру в жанрі «шутер-стратегія».
7. Протестувати гру.

3

АНАЛІЗ АНАЛОГІВ

Назва застосунку	UpGun	Lovely Planet	Polygon	**Розроблений продукт «The Language of Guns»**
Платформа	Windows	Windows	Windows	**Windows**
Мультиплеєр	+	-	+	**-**
Штучний інтелект ворогів	-	+	-	**+**
Реалістичний звуковий дизайн	-	-	+	**+**
Різноманітність зброї	-	-	+	**+**

4

КОНЦЕПТ ГРИ

- Гравець в ролі спецпризначенця, опиняється в особливих та важливих місцях ворогів і має ламати їхні плани
- Кожний рівень гри починається з опису місії та завдань які гравець повинен виконати
- Кожний рівень містить в собі дані розвідки, які надають додаткову інформацію, яка допоможе гравцеві обрати найбільш підходящу зброю, та продумати тактику
- Наступним етапом в грі є закупівля зброї
- Далі починається рівень, де гравцеві потрібно виконати поставлені перед ним завдання, щоб перейти на наступний рівень

5

ВИМОГИ ДО ІГРОВОГО ДОДАТКУ

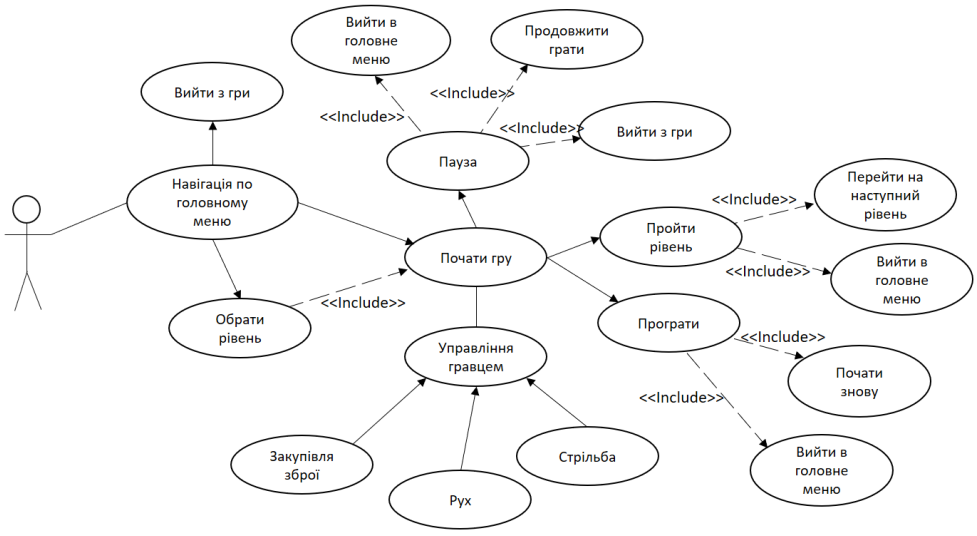
- Ігровий додаток має забезпечувати ігрову механіку, щоб гравці мали можливість взаємодіяти з грою та розвивати свої стратегічні навички.
- Створити набір управління для гравця: рух гравця та стрільба.
- Розробити дані розвідки у вигляді картинок, які допомагатимуть обрати зброю гравцю та додати їх до панелі з розвідкою.
- Створити магазин зброї для гравця.
- Розробити різні види зброї, які гравець зможе купувати.
- Створити систему здоров’я гравця та ворогів.

6

ІНСТРУМЕНТИ ТА ТЕХНОЛОГІЇ

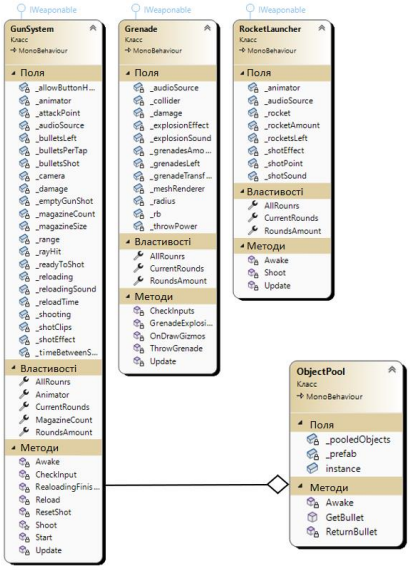
7

ДІАГРАМА ВИКОРИСТАННЯ



8

ДІАГРАМА КЛАСІВ ЗБРОЇ

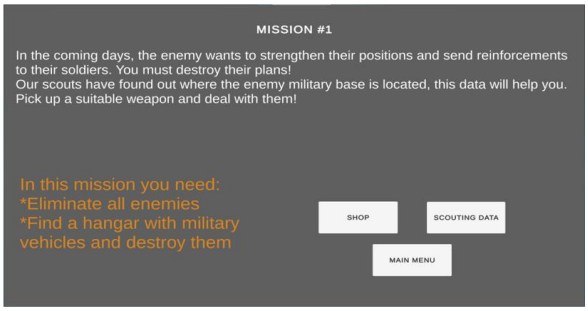


ЕКРАННІ ФОРМИ

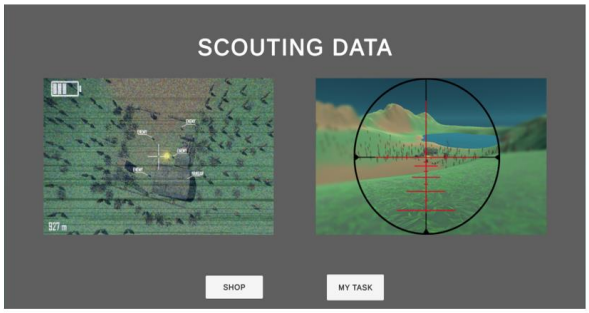
Головне меню

Меню вибору рівня

ЕКРАННІ ФОРМИ



Панель опису місії та її завдання



Панель даних розвідки

ЕКРАННІ ФОРМИ

Панель магазину

Процес проходження рівня

ЕКРАННІ ФОРМИ

13

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- Левчук М.П. Система частинок. Всеукраїнська науково-технічна конференція Науково-технічна конференція «Сучасний стан та перспективи розвитку IoT» 07.04.23 ДУТ, м.Київ – К.: ДУТ, 2023. – С. 107 – 108.

- Левчук М.П. Анімація в Unity. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ» 20.04.23 ДУТ, м.Київ – К.: ДУТ, 2023. – С. 120 – 121.

14

ВИСНОВКИ

1. Проведено аналіз сучасних програмних рішень, які використовуються для розробки ігрових додатків за допомогою платформи Unity.
2. Розглянуто актуальність жанру «шутер-стратегія».
3. Розглянуто програмні засоби для розробки комп’ютерної гри жанру «шутер-стратегія».
4. Розроблено вимоги на основі аналізу аналогічних ігор в цьому жанрі до комп’ютерної гри.
5. За вимогами розроблено гру в жанрі «шутер-стратегія».
6. Протестовано гру.

15

ДЯКУЮ ЗА УВАГУ!

16