НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

до бакалаврської роботи
на ступінь вищої освіти бакалавр
на тему: **«Розробка гри "LastBattle" з використанням ігрового рушія Unity мовою C#»**

Виконав: студент 4 курсу, групи ПД-44
Спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Марковський П.П.
(прізвище та ініціали)
Керівник Дібрівний О.А.
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)
Нормоконтроль_____
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність - 121 «Інженерія програмного забезпечення»

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

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Марковський Павло Павлович
(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка гри "LastBattle" з використанням ігрового рушія Unity мовою C#»

   Керівник роботи: доцент кафедри ІПЗ, доктор філософії Дібрівний О. А.
   (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. Положення побудови відеогри
   3.2. Методи побудови відеоігор
   3.3. Існуючи відеоігри жанру перегони
   3.4. Науково-технічна література

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
    4.1. Аналіз предметної галузі
    4.2. Засоби програмної реалізації
    4.3. Проектування відеогри
    4.4. Розробка відеогри
    4.5. Тестування відеогри
    4.6. Висновки
5. Перелік демонстраційного матеріалу
    5.1. Титульний слайд
    5.2. Мета, об’єкт та предмет дослідження
    5.3. Порівняння аналогів
    5.4. Технічне завдання
    5.5. Програмні та технічні засоби реалізації
    5.6. Use case діаграма
    5.7. Діаграми класів
    5.8. Алгоритм програми
    5.9. Тестування
    5.10. Апробація результатів дослідження
    5.11. Висновки
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	25.02.23-27.02.23	Виконано
2	Аналіз та дослідження існуючих аналогів	28.02.23-29.02.23	Виконано
3	Дослідження програмних засобів	30.02.23-31.02.23	Виконано
4	Моделювання об’єкту проєктування	01.03.23-05.03.23	Виконано
5	Розробка гри “LastBattle”	06.03.23-25.04.23	Виконано
6	Вступ, висновки, реферат	26.04.23-02.05.23	Виконано
7	Розробка обов’язкових демонстраційних матеріалів	03.05.23-14.05.23	Виконано
8	Попередній захист роботи	22.05.23	Виконано
9	Здача роботи	1.06.2023	Виконано

Студент _____ Марковський П.П.
(підпис) (прізвище та ініціали)

Керівник роботи_____ Дібрівний О.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 61 с., 3 табл., 16 рис., 18 джерел.

UNITY, C#, РУШІЙ, ООП, SOLID, ПАТТЕРНИ

Об`єкт дослідження – процес розробки ігрового додатка з використанням Unity.

Предмет дослідження – технології розробки відеогри у жанрі shoot 'em up за допомогою рушія Unity.

Мета роботи – розробити ігровий додаток "LastBattle" з можливістю його використання на платформі Android.

Методи дослідження – методи побудови відеоігор, методи структурного аналізу і проектування, методи розробки програмного забезпечення, методи тестування відеоігор, методи верифікації програмного забезпечення.

Визначено основні недоліки та переваги обраної вибірки конкурентних ігор у жанрі shoot 'em up.

Здійснено поектування гри у жанрі shoot 'em up з урахуванням аналізу ігор-аналогів.

На основі результатів виконаних досліджень розроблено гру у жанрі shoot 'em up.

Упровадження розробленої схеми (методики) дозволяє використовувати дослідження для розробки ігор із дотриманням сучасних потреб ринку.

ЗМІСТ

Стор.

ВСТУП…… 11

РОЗДІЛ 1. ВИВЧЕННЯ ПРЕДМЕТНОЇ ГАЛУЗІ……13

1.1. Відеоігри……..13

1.1.1. Огляд історії комп'ютерних ігор……13

1.1.2. Значення комп'ютерних ігор в економіці та суспільстві…….14

1.1.3. Де можуть використовуватись комп'ютерні ігри…….14

1.2. Жанри ігор……...15

1.3. Як створюються ігри ……..17

1.4. Рушії для розробки ігор……..18

1.5. Платформи для відеоігор……20

1.6. Ігри-конкуренти……...25

1.6.1. Ігри-аналоги……..25

1.6.2. Порівняння ігор……27

РОЗДІЛ 2. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОБОТИ……..29

2.1 Рушій Unity……29

2.2 Інтегроване середовище Visual Studio……29

2.3 Inkscape……..30

РОЗДІЛ 3. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ……31

3.1 Концепція гри……31

3.2 Use case діаграма……...32

3.3 Діаграма діяльності (Activity Diagram)……...34

3.4 Діаграма класів проєкту……...36

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ……39

4.1 Планування розробки ПЗ……39

4.2 Підготовка проєкту……...40

4.3 Розробка проєкту……..44

4.3.1 Головні класи гри……44

4.3.2 Алгоритм створення та налагодження сцени……49
4.3.3 Алгоритм зброї……51
4.3.4 Розробки інтерфейсу гри……54
4.3.5 Завершення розробки……..57
ВИСНОВОК……..58
ПЕРЕЛІК ПОСИЛАНЬ……..60
ДОДАТОКА……..……62

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

UI – user interface

AI - artificial intelligence

ПК – персональний комп’ютер

Рушій – програма, що бере на себе виконання фізики та графічних аспектів гри

ПЗ - програмне забезпечення

ВСТУП

Актуальність роботи полягає в тому, що ігрова індустрія постійно розвивається та зростає популярність ігор на різних платформах. Розробка власної гри є актуальною та перспективною задачею для програмістів, які бажають займатися розробкою ігор. Основною причиною вибору даної сфери стали особиста любов та інтерес. Процес створення віртуальних світів - захоплюючий та неймовірно цікавий. Самі ігри є корисними не лише для відпочинку, але і для медицини та навчання. Комп'ютерні ігри можуть бути використані для тренування медичних фахівців, а також для підвищення рівня здоров'я пацієнтів. Наприклад, ігрові технології можуть бути використані для реабілітації людей після інсульту або травми головного мозку, допомагаючи покращити їх когнітивні та моторні навички.

Можуть бути використані для створення симуляцій, що можуть допомогти у вивченні складних процесів та ситуацій. Наприклад, симуляції можуть бути використані для тренування пілотів, лікарів, пожежників та інших фахівців, що працюють у високоризикових професіях. Тому актуальність цієї сфери діяльності висока, що є ще однією причиною вибору саме цього напрямку.

Розробка гри "LastBattle" на базі ігрового рушія Unity мовою C# є актуальним завданням, оскільки дозволяє вивчити процес розробки гри на одній з найпопулярніших платформ для розробки ігор, а також розвивати навички програмування та проектування ігор.

Об'єкт дослідження - процес розробки геймплею, що базується на симуляції космічних битв.

Предмет дослідження - технології розробки відеогри за допомогою рушія Unity.

Мета і завдання роботи - збільшення зацікавленості гравців жанром shoot 'em up шляхом створення гри "LastBattle", розробленої з використанням сучасних технічних засобів на основі аналізу переваг і недоліків ігор-аналогів.

Методика дослідження - методи побудови відеоігор, методи структурного аналізу і проектування, методи розробки програмного забезпечення, методи тестування відеоігор, методи верифікації програмного забезпечення.

Наукова новизна роботи – використання старого жанру Shoot 'em up та покращення його завдяки сучасним методам розробки ігор, а також його підлаштування під сучасні потреби індустрії.

Практична значущість результатів дослідження – отримання досвіду з використанням новітніх технологій розробки та реалізація проекту, що може стати корисним для навчання нових спеціалістів

РОЗДІЛ 1. ВИВЧЕННЯ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Відеоігри

1.1.1. Огляд історії комп’ютерних ігор

Комп’ютерні ігри - це розважальні програми, створені для електронних пристроїв, таких як персональні комп’ютери, ігрові консолі, смартфони, планшети тощо.

Історія комп’ютерних ігор почалася в 1958 році, коли американський науковець Вільям Гіггінботом зі студентами Массачусетського технологічного інституту створив першу гру "Теніс для двох". Гра була створена для використання на PDP-1, одному з перших електронних комп'ютерів.

У наступні роки існували лише прості ігри, що складалися з простих малюнків та використовувались на навчальних курсах та дослідженнях.

У 1971 році створено першу аркадну гру Spacewar!, яка була запущена на кількох комп'ютерах PDP-1. Гра була винайдена студентами Массачусетського технологічного інституту та є першою грою, яка була більш-менш доступна для громадськості.

У 1980-х роках почали з'являтись перші ігрові консолі, такі як Atari 2600, яка дозволяла гравцям грати в домашніх умовах. У цей період були створені такі легендарні ігри, як Pac-Man, Donkey Kong, Super Mario Bros. Також почали з'являтись перші рольові ігри, такі як Ultima та Wizardry, які вимагали від гравців більшої уваги до деталей та розвитку свого персонажа.

З появою персональних комп'ютерів та Інтернету в 1990-х роках ігри стали більш популярними та доступними для широкого кола користувачів. Компанії, такі як Blizzard Entertainment, Electronic Arts, та Microsoft Game Studios, стали лідерами у галузі розробки та видавництва ігор. За останні десятиліття комп'ютерні ігри стали однією з найбільш прибуткових галузей розваг у світі.

1.1.2. Значення комп'ютерних ігор в економіці та суспільстві

Комп'ютерні ігри мають значний вплив на економіку та суспільство. Вони стали однією з найбільш прибуткових галузей розваг у світі та щорічно генерують мільярди доларів. У 2020 році, згідно з дослідженням Newzoo, глобальні доходи від комп'ютерних ігор склали більше 159 мільярдів доларів, що є надзвичайно великою сумою.

Крім економічної вигоди, комп'ютерні ігри також мають великий вплив на суспільство та культуру. Вони допомагають розвивати креативність та інтелектуальні здібності, розвивають соціальні навички та допомагають в боротьбі з депресією та стресом. Крім того, вони стали важливим інструментом в освіті та дослідженнях.

Комп'ютерні ігри також мають значний вплив на культуру та мистецтво. Багато ігор мають унікальні сюжети та графічний дизайн, що використовуються в кіно, музиці, телебаченні та інших медіа. Крім того, комп'ютерні ігри стали важливим засобом для популяризації та збереження культурної спадщини та історії.

1.1.3. Де можуть використовуватись комп'ютерні ігри

Крім розваг та культурного впливу, комп'ютерні ігри можуть використовуватись в різних галузях, включаючи освіту, медицину, військову справу та інші.

Освіта - комп'ютерні ігри можуть використовуватись як ефективний засіб навчання та розвитку різних навичок. Вони можуть бути використані для навчання дітей математики, історії, географії та інших предметів, а також для розвитку логічного мислення, уваги та пам'яті.

Медицина - комп'ютерні ігри можуть бути використані в медицині як інструмент для діагностики та лікування різних захворювань, а також для реабілітації після травми або операції. Наприклад, віртуальна реальність може бути використана для діагностики та лікування фобій та тривоги, а також для реабілітації після інсульту або травми головного мозку.

Військова справа - комп'ютерні ігри можуть бути використані в військовій справі як інструмент для тренування та розвитку тактичних навичок та стратегій. Вони можуть бути використані для навчання військовослужбовців, для тренування реакції на різні ситуації та для розвитку координації та співпраці в команді. Зараз найчастіше ігри використовують для навчання пілотів та танкістів для відтворення симуляції.

Усі ці аспекти показують, що комп'ютерні ігри можуть мати значну практичну користь, і їх використання не обмежується лише розвагами. Таким чином, вивчення комп'ютерних ігор може бути корисним для розвитку не лише геймерів, але й для суспільства в цілому.

Зважаючи на важливість комп'ютерних ігор в сучасному світі, дослідження в галузі їх розробки, ігрової механіки та взаємодії з користувачами є дуже актуальними та цікавими. Таким чином, у цій дипломній роботі метою є вивчення та аналіз комп'ютерних ігор з точки зору процесу їх розробки.

1.2. Жанри ігор

Згідно зі статистикою, на сьогоднішній день існує більше 20 різних жанрів комп'ютерних ігор, кожен з яких має свої особливості та нюанси. Розуміння різних жанрів відеоігор є дуже важливим для вивчення та аналізу комп'ютерних ігор з точки зору їх механіки та розвитку.

Один з найпопулярніших жанрів - це шутер (FPS - first-person shooter), в якому гравець відтворює персонажа з першої особи та веде бій з ворогами за допомогою зброї. Інші популярні жанри містять:

- RPG (role-playing game), у яких гравець бере на себе роль персонажа та взаємодіє з віртуальним світом;
- action-adventure, що комбінує дії та пригоди з розв'язуванням головоломок;

- симулятори, у яких гравець може контролювати різні аспекти віртуального світу, такі як політ, гонки, ферма тощо;
- стратегії, що охоплюють управління ресурсами, будівництво, битви тощо;
- спортивні ігри, які симулюють різні види спорту та змагання.

Кожен з найпопулярніших жанрів комп'ютерних ігор має свої особливості та нюанси. Розглянемо детальніше кожен з них:

- Шутер (FPS - first-person shooter): це один з найпопулярніших жанрів комп'ютерних ігор, у якому гравець контролює персонажа з першої особи та веде бій з ворогами за допомогою зброї. Цей жанр включає такі піджанри, як тактичні шутери, науково-фантастичні шутери, військові шутери тощо. Головною метою гри є знищення ворогів та виконання завдань.
- RPG (role-playing game): у цьому жанрі гравець бере на себе роль персонажа та взаємодіє з віртуальним світом, виконуючи завдання, розвиваючи персонажа та взаємодіючи з іншими гравцями. Цей жанр має різні піджанри, такі як екшн-рпг, які поєднують елементи RPG та екшну, та традиційні RPG, які більше фокусуються на історії та розвитку персонажа.
- Action-adventure: цей жанр комбінує дії та пригоди з розв'язуванням головоломок. Цей жанр включає такі піджанри, як екшн-пригоди, графічні пригоди, пригоди з відкритим світом тощо. Головною метою гри є виконання завдань та розв'язування головоломок, а також розвиток персонажа.
- Симулятори: цей жанр дозволяє гравцю контролювати різні аспекти віртуального світу, такі як політ, гонки, ферма тощо. Цей жанр має різні піджанри, такі як симулятори війни, симулятори космічного польоту, симулятори підводного світу та інші. Особливістю симуляторів є те, що гравець контролює окремі аспекти світу та взаємодіє з ними за допомогою різних інструментів.

- Стратегії: у цьому жанрі гравець приймає рішення щодо управління віртуальним світом, включаючи збір ресурсів, військові дії та розвиток технологій. Цей жанр включає такі піджанри, як реальні часу стратегії, покрокові стратегії та глобальні стратегії.
- Спорт: у цьому жанрі гравець може брати участь у різних віртуальних спортивних подіях, таких як футбол, баскетбол, гольф тощо. Цей жанр може бути реалістичним або аркадним, залежно від того, наскільки деталізованими є рухи персонажів та реалістичність самої гри.

Ці жанри не є вичерпним списком, але вони є основними та найпопулярнішими серед комп'ютерних ігор.

Крім того, існують такі жанри, як музичні ігри, віртуальна реальність та ігри, що базуються на історії та культурі. Розуміння різних жанрів відеоігор може допомогти дослідникам зрозуміти особливості ігрової механіки та взаємодії з користувачами в кожному жанрі, що може призвести до покращення розробки та взаємодії з користувачами в майбутньому.

1.3. Як створюються ігри

Створення відеоігор - це довгий та складний процес, що включає в собі багато етапів. Ось загальна схема створення відеоігор:

- Концептуалізація: цей етап включає в себе формування ідеї гри та її концептуалізацію. В результаті цього етапу визначається жанр гри, її механіки та сюжет.
- Проектування: на цьому етапі розробляється детальний план розробки гри, складаються розділи про геймплей, характеристики персонажів, локації та інші елементи.
- Розробка: на цьому етапі розроблюються окремі елементи гри, такі як графіка, звук, інтерфейс користувача, механіки гри та інші. Розробка

зазвичай відбувається за допомогою спеціалізованих програмних засобів, таких як редактори графіки, редактори звуку, мови програмування тощо.

- Тестування: цей етап включає в себе тестування гри на різних етапах її розробки. Тестування допомагає виявляти помилки та недоліки гри, що потребують виправлення.
- Випуск: цей етап включає в себе підготовку гри до випуску. До випуску входять такі етапи, як збірка гри, тестування на останній стадії, а також випуск гри на ринку.
- Підтримка: після випуску гри на ринку розробники продовжують працювати над грою, виправляючи помилки та додаючи новий контент.

У процесі створення відеоігор використовуються різні технології та інструменти, такі як рушії гри, мови програмування, 3D-моделювання, аудіо та відео-редактори тощо. Також у створенні гри беруть участь різні фахівці, такі як програмісти, дизайнери, художники, композитори, звукорежисери та інші.

Розробники відеоігор можуть використовувати готові рушії гри (наприклад, Unity, Unreal Engine, CryEngine тощо), які надають розробникам готовий набір інструментів для створення гри, включаючи графіку, фізику, звук, AI та інше. Використання готового рушія гри дозволяє зосередитись на творчості та геймплеї, а не на розробці базового движка.

Крім того, у створенні відеоігор використовуються різні мови програмування, такі як C++, C#, Java, Python, Lua тощо. Кожна мова програмування має свої переваги та недоліки та вибір мови залежить від конкретного завдання.

Для створення графіки використовуються різні програми для 3D-моделювання, такі як Maya, 3ds Max, Blender тощо. Ці програми дозволяють створювати 3D-об'єкти та анімацію, які потім імпортуються у рушій гри.

Для створення аудіо ефектів та музики використовуються різні програми, такі як FL Studio, Ableton Live, Logic Pro тощо.

В кінці усі ці ресурси збираються до єдиного проєкту, що і є готовою відеогрою.

1.4. Рушії для розробки ігор

У минулому ігри розроблялися з використанням мов програмування, таких як C++, та ручного написання коду, що призводило до значних витрат часу та зусиль. Зараз розробники використовують різні рушії, які дозволяють спростити процес розробки та зменшити час, необхідний для створення ігри.

Один з найбільш популярних рушіїв для розробки ігор є Unity. Unity є повнофункціональним інтегрованим середовищем розробки (IDE), яке включає в себе все, що потрібно для створення ігор, включаючи 2D та 3D графіку, фізику, аудіо, AI та інші функції. Використання Unity дозволяє розробникам швидко створювати ігри для різних платформ, таких як комп'ютери, консолі та мобільні пристрої.

Інший популярний рушій для розробки ігор - Unreal Engine, який створений компанією Epic Games. Unreal Engine, також є повнофункціональним IDE, який надає розробникам можливість створювати ігри з високоякісною графікою та фізикою. Unreal Engine використовують для розробки багатьох відомих ігор, таких як Fortnite, Gears of War та Batman: Arkham Asylum.

Інші рушії для розробки ігор включають CryEngine, який використовується для створення ігор з вражаючою графікою, та GameMaker Studio, який дозволяє створювати прості ігри без необхідності програмування.

Кожен рушій має свої переваги та недоліки. Наприклад, Unity є дуже популярним рушієм, який має велику спільноту розробників та багато ресурсів для навчання. Він також має простий інтерфейс та дозволяє розробникам створювати ігри для різних платформ. Однак Unity має свої обмеження щодо графічної якості та швидкості роботи, що може бути проблемою для деяких проєктів.

Unreal Engine, з іншого боку, має вражаючу графіку та фізику, що дозволяє розробникам створювати реалістичні ігри. Він також має більш потужну систему штучного інтелекту та фізики, ніж Unity. Однак Unreal Engine має більш високі вимоги до апаратного забезпечення та складніший інтерфейс.

CryEngine також має дуже добру графіку та фізику, але його використання не таке популярне, як Unity або Unreal Engine, тому може бути складно знайти ресурси для навчання та підтримки.

GameMaker Studio, з іншого боку, дозволяє розробникам створювати прості ігри без необхідності програмування, що робить його доступним для новачків. Однак його можливості щодо графіки та фізики обмежені, тому він не підходить для більш складних проєктів.

У загальному, вибір рушія залежить від потреб проєкту та вмінь розробника. Кожен з рушіїв має свої переваги та недоліки, і розробники повинні зважити на них, перш ніж вибрати рушій для свого проєкту.

1.5. Платформи для відеоігор

Звісно, залежно від контексту, під платформою для відеоігор можна мати на увазі різні речі, такі як ігрові консолі, персональні комп'ютери, мобільні пристрої тощо. Тому у цьому розділі я розгляну різні типи платформ, що використовуються для гри в відеоігри, та найвідоміші компанії, які їх виготовляють.

- Ігрові консолі

Ігрові консолі є спеціалізованими пристроями, що призначені тільки для гри в відеоігри. Найвідоміші серед них такі платформи, як Sony PlayStation, Microsoft Xbox та Nintendo Switch.

Sony PlayStation: PlayStation - це ігрова консоль, що випускається компанією Sony. PlayStation 5 є найновішою версією консолі та має потужний процесор, що забезпечує високу якість графіки та відтворення відео. Однією з особливостей PlayStation є її ексклюзивні ігри, які доступні тільки на цій платформі. До таких ігор належать The Last of Us Part II, Ghost of Tsushima та Spider-Man: Miles Morales.

Microsoft Xbox: Xbox - це ігрова консоль, що випускається компанією Microsoft. Xbox Series X та Xbox Series S є найновішими версіями консолі та мають подібну потужність до PlayStation 5. Xbox також має свої ексклюзивні ігри, до яких належать Halo Infinite, Forza Horizon 5 та Fable.

Nintendo Switch: Nintendo Switch - це гібридна ігрова консоль, що випускається компанією Nintendo. Вона має можливість відтворення як на телевізорі, так і на портативній консолі. Однією з особливостей Nintendo Switch є її ексклюзивні ігри, що включають в себе такі хіти, як The Legend of Zelda: Breath of the Wild, Super Mario Odyssey та Animal Crossing: New Horizons.

- Персональні комп'ютери

Персональні комп’ютери (ПК) є однією з найбільш універсальних платформ для відеоігор. Основними компаніями, що виробляють комп'ютери та компоненти для них, є такі бренди як Dell, HP, ASUS, Acer, Lenovo та інші.

Однією з переваг використання ПК для гри є можливість налаштувати компоненти для виконання специфічних завдань. Крім того, на ПК є доступ до великої кількості ігор та платформ, включаючи відкриту платформу Steam, яка містить найбільший каталог ігор на ПК.

Незважаючи на це, одним з недоліків використання ПК для гри є те, що не всі ігри можуть бути запущені на всіх конфігураціях комп’ютерів. Також велика кількість налаштувань та можливостей комп'ютерів може зробити процес гри складнішим для користувачів з меншим досвідом в цій сфері.

Щодо видів ПК для гри, можна виділити настільні та ноутбуки. Настільні ПК зазвичай є потужнішими та дозволяють налаштовувати окремі компоненти, такі як процесор, графічна карта, пам'ять та інші. Ноутбуки, з іншого боку, зазвичай менш потужні, але мають переносність та зручність використання.

Окрім того, є також відносно новий варіант платформи для відеоігор - міні-комп'ютери, такі як Raspberry Pi, Nvidia Shield та інші. Ці пристрої зазвичай мають менші розміри та меншу потужність, але можуть бути досить гнучкими та доступними для деяких користувачів.

- Мобільні пристрої

Наступною платформою, про яку варто згадати, є мобільні пристрої. За останні кілька років мобільні ігри зарекомендували себе як одна з найпопулярніших категорій відеоігор. Мобільні ігри можна грати на смартфонах і

планшетах з Android та iOS операційними системами. Найбільш популярними платформами для мобільних ігор є Google Play і App Store.

Google Play є магазином додатків для Android, розробленим компанією Google. Він містить велику кількість ігор, які можна скачати на пристрої з Android. App Store є магазином додатків для iOS, який був розроблений компанією Apple. Він також містить велику кількість ігор, які можна скачати на iPhone та iPad.

- VR

VR - це абревіатура від Virtual Reality, технологія, яка дозволяє користувачеві відчувати віртуальне середовище так, ніби він перебуває у ньому фізично. Вона стала доступною для масового використання в останні роки завдяки розвитку інтернету, графічних процесорів та спеціального обладнання.

Основними компаніями, що виготовляють VR-платформи, є Oculus (заснований у 2012 році, пізніше був придбаний Facebook), HTC (що співпрацює з Valve) та Sony зі своєю Playstation VR. Також на ринку присутні більш доступні варіанти, наприклад, Samsung Gear VR, Google Daydream, а також Oculus Go та Oculus Quest.

Переваги VR-технології полягають у можливості занурення відвідувача в ігровий світ, здійсненні реалістичного візуального та звукового досвіду, та поглибленні в інтерактивний процес. Використання VR дозволяє створювати ігри та додатки, що раніше не були можливі.

Щодо недоліків VR-ігор, на перший план виходить їх вартість. Потрібні досить дорогі технології, які можуть бути недосяжними для багатьох користувачів. Також може виникати дискомфорт під час використання, зокрема погіршення зору, пітливість обличчя, поганий контроль рухів тощо. Також деякі гравці можуть відчувати дезорієнтацію або нудоту під час гри, особливо якщо відчуття простору не відповідає руху користувача.

Також слід відзначити, що на даний момент ринок VR-ігор ще не настільки розвинений, як класичний ринок ігор для персональних комп'ютерів або консолей. Тому вибір ігор може бути обмеженим, а якість їх виконання може бути нерівною.

Отже, можна зробити висновок, що VR-ігри - це експериментальна галузь геймінгу, яка ще не настільки поширена, як класичні ігри. Однак з появою нових технологій і підвищенням інтересу гравців, розвиток цієї галузі може зростати та привернути ще більше уваги у майбутньому.

Яку саме платформу обрати, сказати неможливо, оскільки це суб'єктивний вибір, який залежить від бажань та можливостей. Але у кожної з платформ є свої переваги та недоліки, з якими варто ознайомитись перед вибором. Також ні ці недоліки варто звертати увагу не лише користувачам, але і розробникам ігор. Від того, яку гру та для якої платформи розробити, може залежати її успішність.

Нижче наведено порівняльну таблицю платформ для відеоігор з їхніми перевагами та недоліками:

Таблиця 1.1 - порівняння платформ для відеоігор

Платформа	Переваги	Недоліки
ПК	Гнучкість, можливість модифікації, висока якість графіки та звуку.	Вимагає високої потужності, деякі ігри можуть бути несумісні з деякими комплектуючими, можуть виникати проблеми зі сумісністю між різними версіями операційних систем.

Продовження таблиці 1.1 - порівняння платформ для відеоігор

PlayStation	Ексклюзивні ігри, простота використання, готовість до використання безпосередньо після придбання.	Недоступність деяких ігор, висока вартість деяких ігор, відсутність можливості модифікації.
Xbox	Можливість грати в ігри з платформи Windows, готовність до використання безпосередньо після придбання, підтримка громадської гри (multiplayer).	Недоступність деяких ігор, висока вартість деяких ігор, відсутність можливості модифікації.
Nintendo Switch	Мобільність, можливість грати відеоігри будь-де та в будь-який час, ексклюзивні ігри.	Менший вибір ігор порівняно з іншими платформами, менша потужність графічного процесора.
Mobile	Доступність, можливість грати в будь-який час та в будь-якому місці, простота використання.	Обмежена потужність, незручне керування у деяких іграх.
VR	Іммерсивний досвід, можливість занурення в ігровий світ, підвищення реалістичності гри.	Висока вартість, обмежений вибір ігор, можливість відчуття головного болю після сеансу гри.

1.6. Ігри-конкуренти

1.6.1. Ігри-аналоги

Жанр Shoot 'em up, також відомий як "shmup", є одним з найпопулярніших жанрів комп'ютерних ігор, який включає в себе ігри, де гравець контролює літаючий корабель або інший літальний апарат та знищує безліч ворожих сил, які атакують його з усіх боків. У цьому розділі ми розглянемо деякі з найбільш відомих та популярних ігор у жанрі shoot 'em up.

- "Space Invaders" - це класична аркадна гра, створена в 1978 році компанією Taito. Гравець керує космічним кораблем та знищує хвилі ворожих сил, які опускаються зверху до нижньої частини екрану. Ця гра є однією з перших ігор у жанрі shoot 'em up та стала важливим культурним явищем в історії відеоігор.
- "Gradius" - це ігра, створена в 1985 році компанією Konami. Гравець керує космічним кораблем, який можна покращувати, збираючи різні бонуси. У грі є безліч різних ворожих сил, що атакують з усіх боків, тому гравець повинен бути дуже уважним та швидким.
- "R-Type" - це ігра, створена в 1987 році компанією Irem. Гравець керує космічним кораблем, який має вбудований модуль зброї, який можна змінювати залежно від ситуації. У грі є безліч ворожих сил, які атакують з різних напрямків, і гравець повинен бути готовий до різних стратегічних рішень.
- "Galaga" - це ігра, створена в 1981 році компанією Namco. Гравець керує космічним кораблем та знищує ворожі сили, які атакують з різних напрямків, а також здобуває бонуси за знищення певної кількості ворожих сил у певному порядку. Гра має класичний 8-бітний стиль та є однією з найпопулярніших ігор у жанрі shoot 'em up.
- "Ikaruga" - це ігра, створена в 2001 році компанією Treasure. Гравець керує космічним кораблем, який може змінювати свою поларність з білої на чорну

та навпаки. Ворожі сили також мають поларність, і корабель гравця може збирати енергію від ворожих сил зі супротивною поларністю. Гра має неперевершені графічні ефекти та глибокий ігровий процес.

- "Cave Story" - це ігра, створена в 2004 році одиночним розробником Дайсуке Амая. Гравець керує маленьким роботом на ім'я Quote, який має різні зброї та навички. Гра має ретро-стиль графіки та звуку, а також неперевершений ігровий процес та глибоку історію.
- "Mushihimesama" - це ігра, створена в 2004 році компанією Cave. Гравець керує космічним кораблем та знищує безліч ворожих сил. Гра має дуже високу складність та неймовірні графічні ефекти, що роблять її однією з найскладніших та найкрасивіших ігор у жанрі shoot 'em up.

Жанр shoot 'em up має безліч інших варіантів, і ці ігри є лише деякими з найвідоміших та популярних. Ці ігри не тільки цікаві та захопливі, але й мають високу складність та вимагають від гравців швидкості реакції та стратегічного мислення. Вони також можуть бути використані для розвитку та поліпшення різних навичок, таких як координація рухів, швидкість реакції та зосередженість.

1.6.2. Порівняння ігор

Таблиця 1.2 – порівняльна таблиця ігор-аналогів

Назва гри	Переваги	Недоліки
Galaga	Простий геймплей, відносно легкий для новачків.	Обмежений вибір зброї та незначна кількість різноманітності.
R-Type	Графіка високої якості та складний геймплей.	Дуже складно для новачків, може відлякувати від гри.

Gradius	Цікава система дозволів на покращення зброї та відзначення балів.	Надто погана графіка.
DoDonPachi	Швидкий темп гри та багато варіантів вибору зброї.	Дуже складний геймплей та несумісність з деякими платформами.
Touhou Project	Яскрава фентезійна графіка та багато грального персонажів.	Доступність тільки на ПК.
Ikaruga	Унікальна система кольорової поляризації зброї та ворогів.	Доступність тільки на певних платформах та складний геймплей.
Radiant Silvergun	Система комбінації зброї та відповідних ворогів.	Доступність тільки на певних платформах та складний геймплей.
Mushihimesama	Багато ворогів та проходів у різних режимах.	Доступність тільки на певних платформах та складний геймплей.

Проаналізувавши переваги та недоліки ігор-аналогів було зроблено висновок, що у всіх подібних іграх є певні схожі недоліки, які я усунув у своєму проєкті.

Основні мінуси:

- Погана, застаріла графіка
- Непопулярність 3D графіки
- Незрозумілий інтерфейс, що ускладнює розуміння гри
- Малий вибір платформ для гри
- Відсутність підтримки мобільних пристроїв.

Для своєї гри я обрав 3D простір та найновіші моделі та ефекти, що покращить графічну якість гри у порівнянні з аналогами. Також завдяки вибору двигуна Unity мені буде легко розробити гру для ПК та мобільних пристроїв. Також я зроблю інтуїтивно зрозумілий, простий інтерфейс, що дозволить грати у гру будь-кому та легко у ній розібратись.

РОЗДІЛ 2. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОБОТИ

2.1. Рушій Unity

Unity є одним з найпопулярніших ігрових двигунів, що використовуються в індустрії розробки ігор. Я обрав Unity для моєї дипломної роботи з кількох причин.

По-перше, Unity є дуже доступним інструментом для створення ігор. Він має велику спільноту розробників, яка допомагає початківцям зрозуміти основи розробки ігор. Крім того, Unity має велику кількість ресурсів та плагінів, що дозволяє прискорити розробку.

По-друге, Unity дозволяє розробникам створювати ігри для різних платформ, таких як ПК, мобільні пристрої та консолі. Це означає, що розробники можуть створювати ігри для багатьох цільових аудиторій та отримувати більше прибутку.

По-третє, Unity має великий набір інструментів та ресурсів для створення графіки та анімації. Це дозволяє розробникам створювати різноманітну графіку та анімацію без необхідності використання інших програм.

Переваги Unity перед іншими ігровими двигунами полягають у його доступності, багатому наборі ресурсів та плагінів, а також у здатності створювати ігри для різних платформ. Крім того, Unity дозволяє розробникам створювати ігри за допомогою скриптів на мові C#, що дозволяє збільшити швидкість розробки та зробити її більш зручною. Також Unity підтримує велику кількість ігрових форматів та інтегрується з різними сторонніми програмами.

2.2. Інтегроване середовище Visual Studio

Visual Studio є інтегрованим середовищем розробки (IDE) від Microsoft, яке широко використовується для розробки програмного забезпечення, включаючи ігри. Я використовував Visual Studio для розробки гри в рамках моєї дипломної роботи.

Однією з переваг Visual Studio є його підтримка мови програмування C#, яка є однією з основних мов програмування, використовуваних для розробки ігор з використанням Unity. Visual Studio забезпечує зручний інтерфейс для написання коду, а також функціональні можливості, такі як автодоповнення, перевірка помилок та налагодження коду.

Visual Studio також має вбудовану систему керування версіями, яка дозволяє зберігати копії коду на різні моменти часу та відновлювати попередні версії у разі необхідності. Це забезпечує більшу безпеку при роботі зі складним кодом та дозволяє швидше відновлювати роботу у разі виникнення проблем.

Visual Studio має інтегровану систему налагодження коду, що дозволяє розробникам легко знаходити та виправляти помилки в програмному коді. Це дозволяє ефективніше відлагоджувати програму та зменшувати час, витрачений на пошук та виправлення помилок.

Узагалі, Visual Studio є потужним інструментом для розробки програмного забезпечення, включаючи ігри. Він забезпечує широкі можливості для розробки коду та відлагодження програми, що дозволяє розробникам ефективно працювати над своїми проєктами.

2.3. Inkscape

Inkscape - це вільне та відкрите ПЗ для векторної графіки. Я використовував Inkscape для створення графіки для гри в рамках моєї дипломної роботи.

Однією з основних переваг Inkscape є те, що він є безкоштовним та відкритим програмним забезпеченням, що дозволяє користувачам використовувати його без будь-яких обмежень. Також Inkscape є переносним, що означає, що його можна встановлювати на різних операційних системах, таких як Windows, Mac і Linux.

Inkscape має інтуїтивний інтерфейс та широкий функціонал для створення векторної графіки, такий як редактування шляхів, робота з формами та колірними

градієнтами, створення тексту, робота зі шрифтами, робота з кривими Безьє. Також, Inkscape підтримує різні формати файлів, такі як SVG, EPS, PDF, AI, DXF та інші.

Ще однією з переваг Inkscape є те, що він має відкритий код, що дозволяє розробникам створювати додаткові розширення та плагіни для програми. Це дозволяє користувачам розширювати функціонал програми та пристосовувати її під свої потреби.

Узагалі Inkscape є потужним та зручним інструментом для створення векторної графіки, який може бути використаний для розробки графіки для ігор. Він має широкий функціонал та підтримує різні формати файлів, що дозволяє користувачам створювати професійну графіку для своїх проєктів.

РОЗДІЛ 3. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Концепція гри

Концепція — система поглядів, те або інше розуміння явищ і процесів; єдиний, визначальний задум.

Перед початком будь-якого проєкту обговорюють його концепцію. Перед початком роботи над проектом, дизайнери займаються розробкою самої ідеї. Вибирається ідея, жанр та простір - 2D, чи 3D. Обговорюється основна сюжетна лінія та механіки. Після цього вибирається рушій та мова програмування і починається робота над проєктом. У моєму випадку концепцію гри я представив у вигляді таблиці та визначив основні характеристики.

Таблиця 3.1 – Концептуальна таблиця гри LastBattle

Назва гри	LastBattle
Жанр	shoot 'em up
Простір	3D
Графіка	Растрова
Двигун	Unity
Мова програмування	C#
Платформи	Android, Windows
Ігровий процес	Гравець з’являється у космосі з початковим кораблем. Його задача якомога довше витримувати хвилі ворогів, знищуючи їх. Також гравцю необхідно покращувати зброю та корабель між хвилями для того, щоб виживати довше.
Сюжет	Гравець відіграє роль ветерана війни, який був зраджений владою та

	визнаний ворогом держави. Герой був оточений та намагається вижити й вирватися з оточення. Але ворогів виявляється надто багато, тож рано чи пізно він гине.

3.2 Use case діаграма

Use case діаграма - це діаграма, яка використовується для моделювання функціональних вимог до системи, тобто того, що система повинна робити. Вона допомагає описати, як користувачі взаємодіють із системою, які дії вони можуть виконувати та як система повинна реагувати на ці дії.

Use case діаграма складається з акторів, які взаємодіють із системою, використовуючи use case'и - сценарії взаємодії з системою. Кожен use case описує один конкретний сценарій взаємодії між акторами та системою, який може бути виконаний системою для задоволення певної потреби користувача. Наприклад, use case «Реєстрація користувача» може включати такі дії, як введення даних користувача, перевірку на унікальність даних, збереження даних в базі даних та відправлення підтвердження реєстрації на електронну пошту користувача.

Use case діаграми можуть бути використані для розробки вимог до системи, проектування інтерфейсу користувача, тестування системи та управління проектами. Вони допомагають комунікації між різними зацікавленими сторонами в проекті та забезпечують зрозумілість вимог до системи.

У моїй діаграмі представлений лише один актор, тому що гра не є мультиплеєрною і у неї може грати лише один гравець на одному пристрої.

Загалом у моїй системі є лише два основні варіанти використання гри. Перший - це взаємодія з меню гри, а другий - взаємодія з інтерфейсом самої гри.

Варіанти використання моєї гри:

Варіанти використання гри в головному меню:

- Налаштування
  - Гучність музики
  - Гучність звуків
- Вихід
- Початок гри

Варіанти використання гри у головному рівні:

- Збільшити/зменшити мапу гри
- Увімкнути магазин
- Повернутись у меню
- Перезапустити рівень

Використовуючи ці дані, можна сформувати use case діаграму, завдяки якій можна зрозуміти, як гравець повинен використовувати гру.

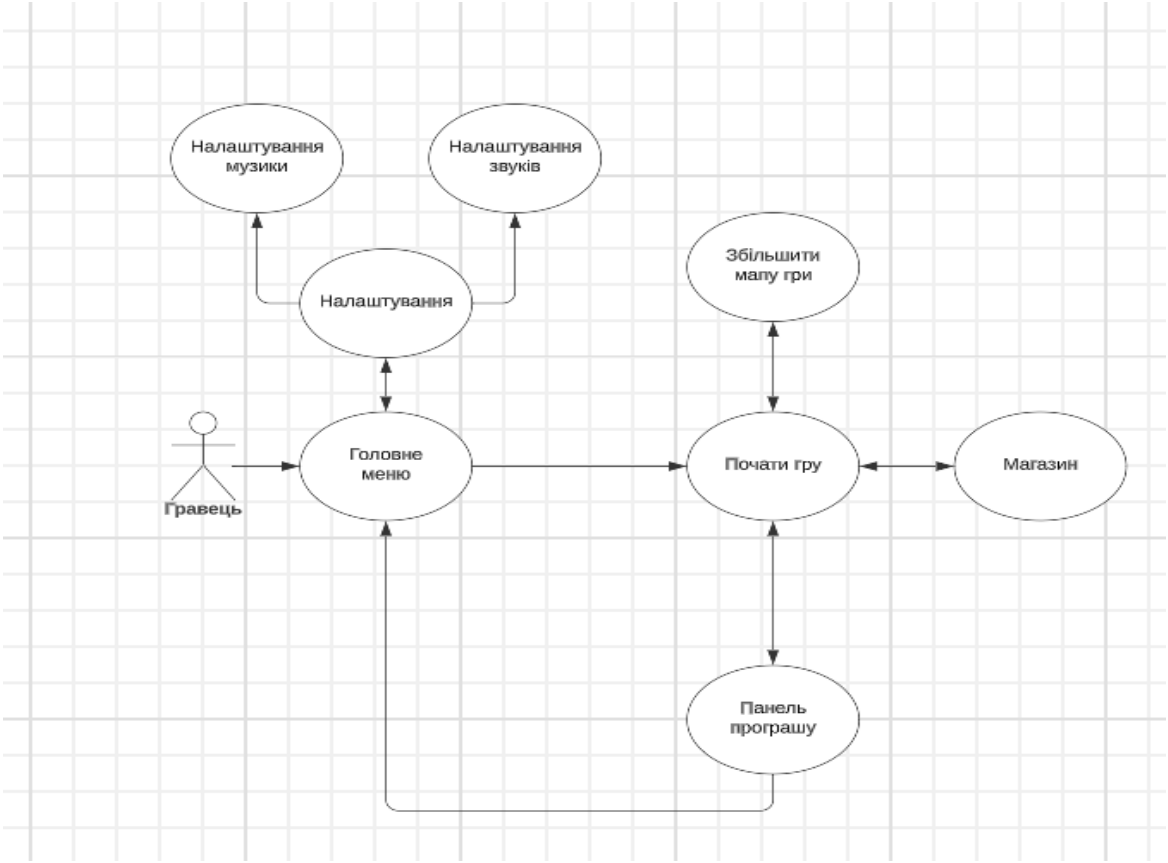


Рисунок 3.1 – Use case діаграма

3.3 Діаграма діяльності (Activity Diagram)

Activity Diagram - це графічна діаграма, яка використовується для моделювання процесів, процедур або дій, що відбуваються в системі або в бізнес-процесі. Це може бути опис взаємодії між користувачем та системою, або процес взаємодії між різними елементами системи.

Activity Diagram містить вершини, які представляють дії або стани, та зв'язки між ними, які показують послідовність виконання дій. Вершини можуть бути подіями, що спричиняють початок дії, діями, що виконуються, коли подія відбувається, або станами, у яких система може знаходитися. Зв'язки можуть бути різних типів, таких як стрілки, що вказують напрямок виконання, або умови, які повинні бути виконані, щоб перейти до наступної дії.

Activity Diagram можуть бути використані для аналізу бізнес-процесів, проектування та розробки програмного забезпечення, моделювання взаємодії між користувачем та системою, а також для відлагодження та відліку програмного забезпечення. Вони можуть допомогти визначити слабкі місця в бізнес-процесах або програмному забезпеченні та покращити ефективність діяльності компанії.

Для своєї діаграми я використовував декілька видів фігур:

- округлені прямокутники позначають дії, які виконуються;
- ромби позначають розгалуження вибору;
- довгі товсті ризики позначають початок та кінець паралельних дій;
- чорне коло позначає старт процесу;
- чорний кружок у колі позначає кінець процесу.

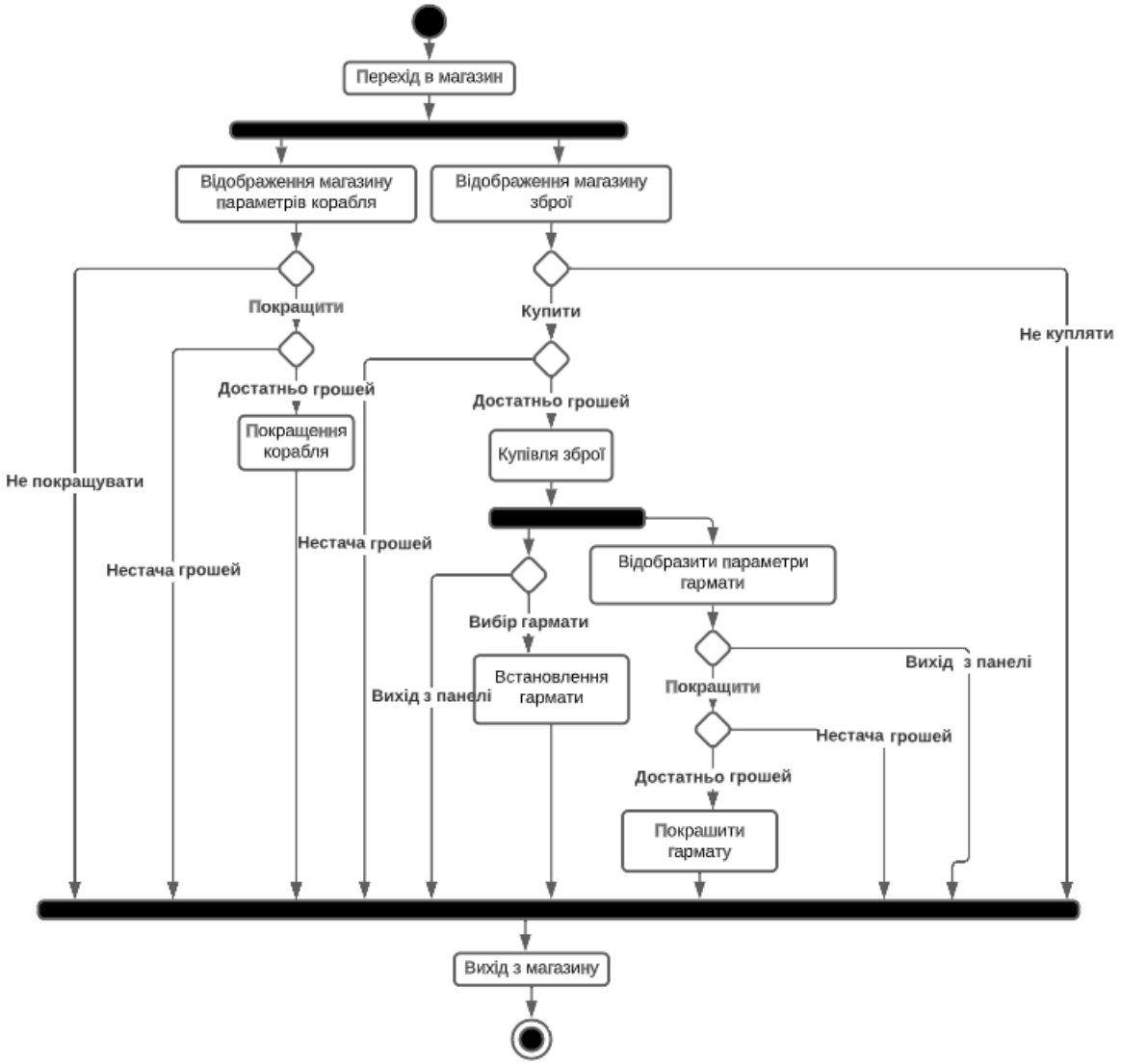


Рисунок 3.2 – Діаграма діяльності

У своїй діаграмі я відобразив, як саме гравець може користуватись магазином та які він має можливості.

Актор (в нашому випадку гравець), переходячи у магазин, має дві функції. Перша – це покращення характеристик гравця, таких як кількість життів та щитів за умови достатньої кількості грошей. Друга – це можливість вибрати та купити нову зброю, після чого відкривається доступ до нових функцій: встановлення зброї на корабель та відкриття панелі покращення характеристик зброї, у якій можна покращити перезарядку та кількість пошкоджень, що наносить зброя.

Після завершення купівлі зброї та покращення її та корабля, гравець може закрити магазин та продовжити гру.

3.4 Діаграма класів проєкту

Діаграма класів (Class Diagram) - це одна з головних діаграм у мові UML (Unified Modeling Language), яка використовується для візуального представлення класів, їх атрибутів, методів та взаємозв'язків між класами в системі або програмному продукті.

У діаграмі класів кожен клас зображується у вигляді прямокутника з трьома секціями: ім'я класу, атрибути та методи. Атрибути - це властивості класу, а методи - дії, які можна виконати з об'єктом цього класу. Взаємозв'язки між класами можуть бути зображені за допомогою стрілок, які вказують на тип відносин, таких як агрегація, композиція, наслідування та інші.

Діаграма класів може бути використана для моделювання будь-якої системи, що містить класи, які мають властивості та методи. Це можуть бути програмні продукти, бази даних, об'єктно-орієнтовані системи та інші. Діаграма класів допомагає встановити структуру системи та розуміння взаємозв'язків між класами, що допомагає в розробці та підтримці програмного забезпечення, а також підвищує його якість та зручність використання.

У моєму проєкті за час його розробки було відтворено 78 класів, 8 інтерфейсів та 7 переліків. Основними класами у мене виступають ті класи, у яких в назві присутні приставки Data та Controller, які відповідно до назви відповідають за збереження даних та за контроль. Основні класи відображенні на рисунку 3.3, а інтерфейси та енумератори зображенні на рисунку 3.4.

рисунку 3.3, а інтерфейси та енумератори зображенні на рисунку 3.4.

Рисунок 3.3 – Діаграма класів

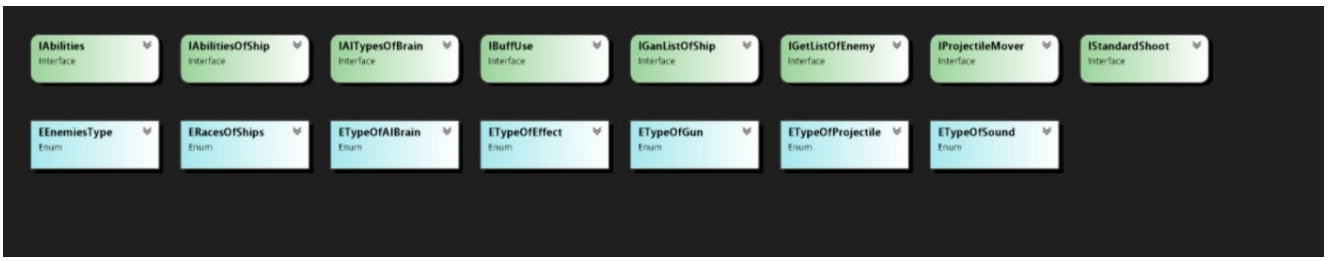


Рисунок 3.4 – Діаграма інтерфейсів та переліки

Також у моєму проекті присутні класи, що наслідуються один від одного. Вони зображенні на рисунку 3.5.

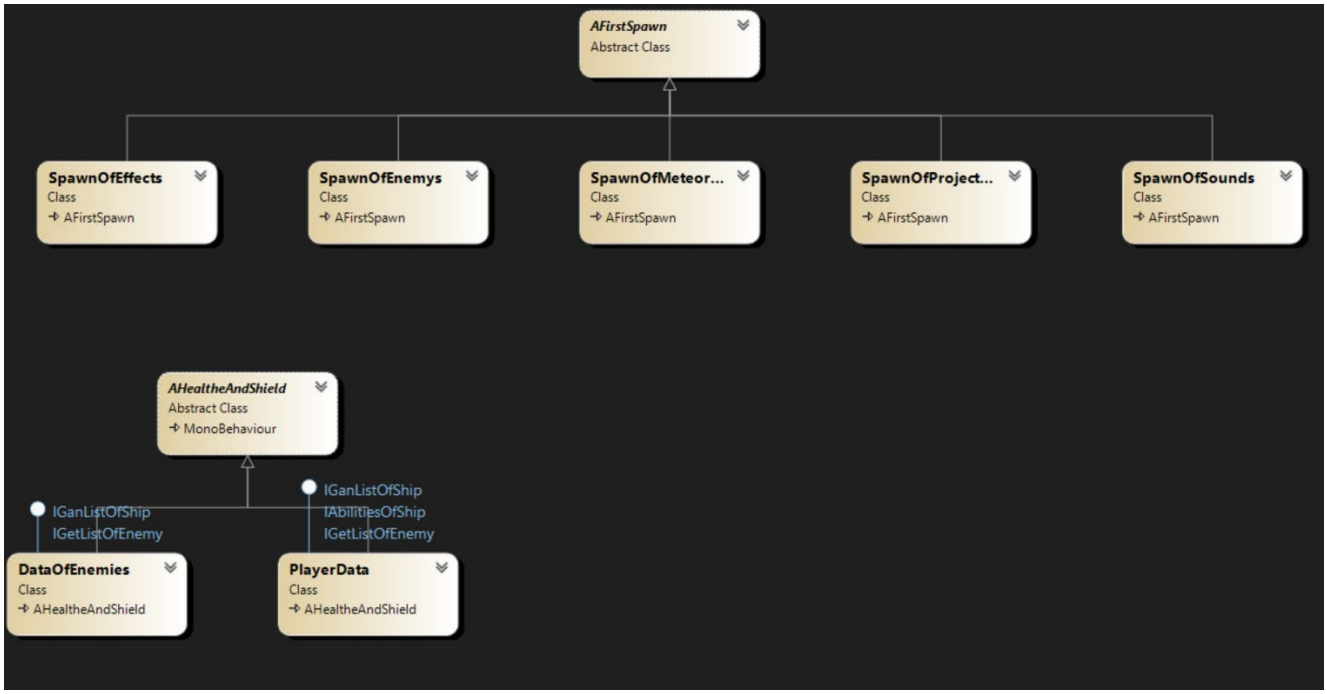


Рисунок 3.4 – Діаграма класів нащадків

На цій діаграмі можна побачити два класи, які є абстрактними, від них наслідуються інші класи. Це було зроблено для реалізації поліморфізму та інкапсуляції.

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Планування розробки ПЗ

Перед початком роботи над будь-яким проєктом необхідно спланувати його розробку. Необхідно спланувати хід роботи та послідовність розробки компонентів гри та її механік.

Першою задачею для мене стало розробити ідею гри та продумати механіки. Для своєї гри я вирішив вибрати жанр shoot 'em up, тому що ігри цього жанру тривалий час залишаються популярними завдяки їхній простоті та інтуїтивності.

З метою отримання переваг над іграми-аналогами, я вирішив вибрати 3D простір, оскільки він досить рідко використовується в іграх цього жанру та візуально приємніший у іграх жанру шутер. Вибравши жанр та простір, я перейшов до розробки самого проєкту.

Для початку я вирішив підготувати сам проєкт та знайти ассети. Після того як я знайшов усі необхідні звуки, ефекти та моделі, я почав розробку основи гри.

План розробки проєкту:

- розробка точки входу;
- створення класу даних рівня;
- розробка скриптів для створення рівню;
- реалізація poolObject для об’єктів гри;
- розробка основних механік корабля (рух, життя, щит);
- створення механіки стрільби;
- створення ШІ;
- розробка магазину;
- реалізація підвищення рівня складності;
- додавання ефектів та звуків;
- додавання Інтерфейсу;
- розробка меню гри;

- рефакторинг коду та налагодження архітектури коду;
- тестування;
- кінцеві виправлення.

Це основні етапи розробки мого проєкту, що були виконані. На рисунку 4.1 зображений GitHub репозиторій мого проєкту, яким я користувався, щоб зберігати прогрес розробки та мати можливість повернутись до минулих версій гри.

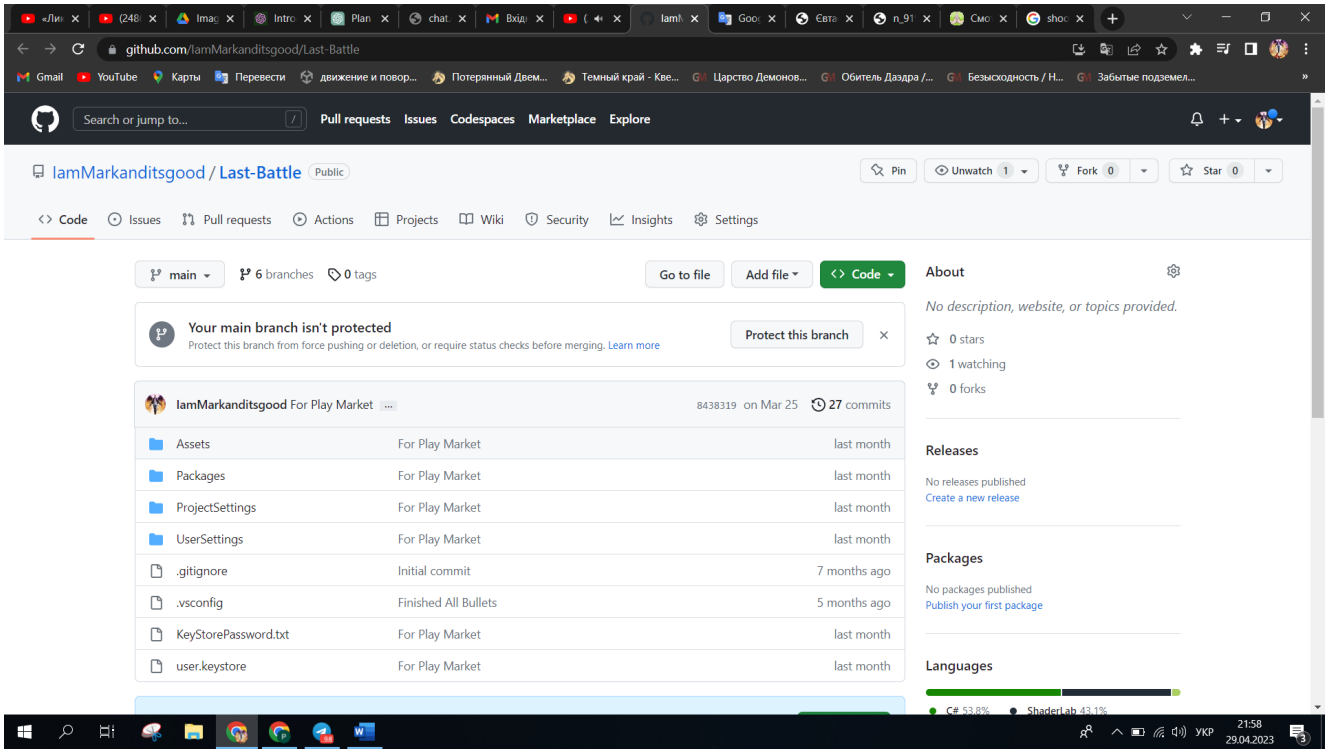
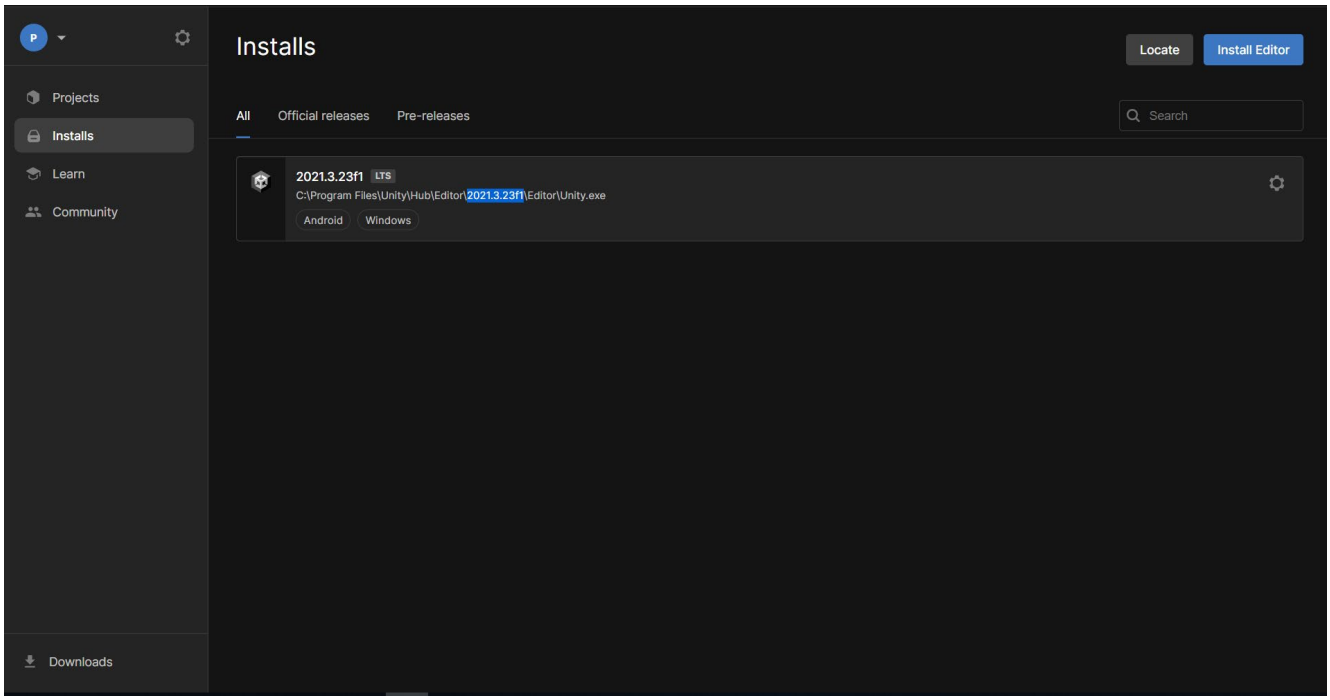


Рисунок 4.1 – Репозиторій з проектом

4.2 Підготовка проєкту

Для початку роботи необхідно вибрати та встановити найновішу версію рушія, в якому ми будемо працювати. Для роботи я встановив версію 2021.3.23f1, що зображена на рисунку 4.2. Після цього необхідно налаштувати інтерфейс для комфортної роботи. Я розташував компоненти інтерфейсу таким чином, щоб було найзручніше взаємодіяти між ними. Мій варіант розташування компонентів зображений на рисунку 4.3.



зображений на рисунку 4.3.

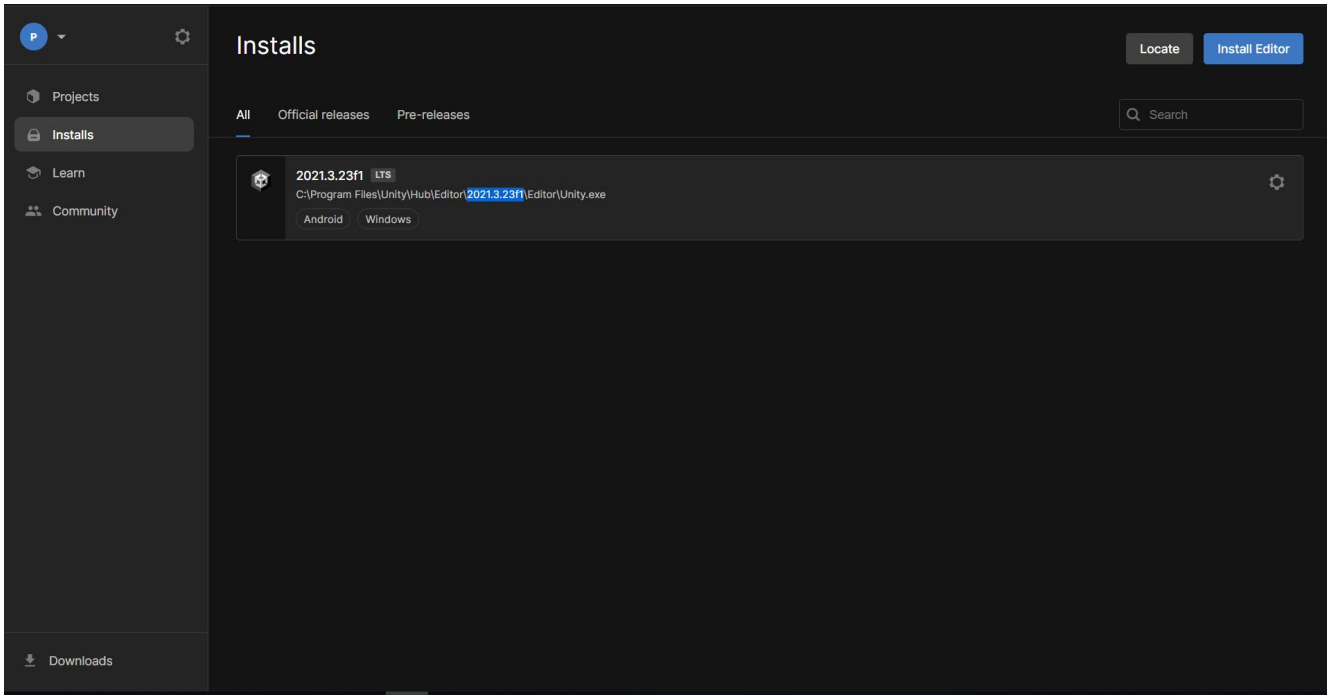


Рисунок 4.2 – Версія рушія Unity

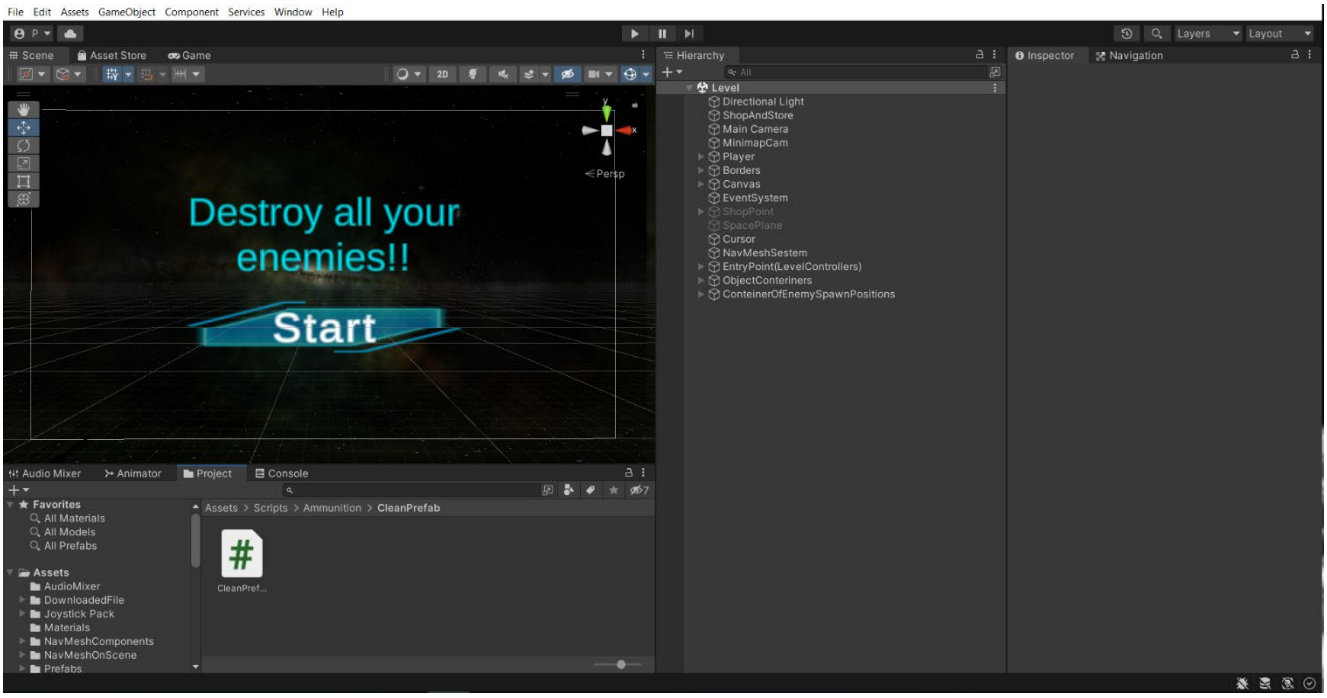


Рисунок 4.3 – Приклад налаштування інтерфейсу Unity

Налаштувавши рушій, я перейшов до пошуку необхідних компонентів гри (звуки, ефекти, моделі, матеріали, шейдери, спрайти). Для пошуку звуків я використовував сайт Zvukipro, зображений на рисунку 4.4.

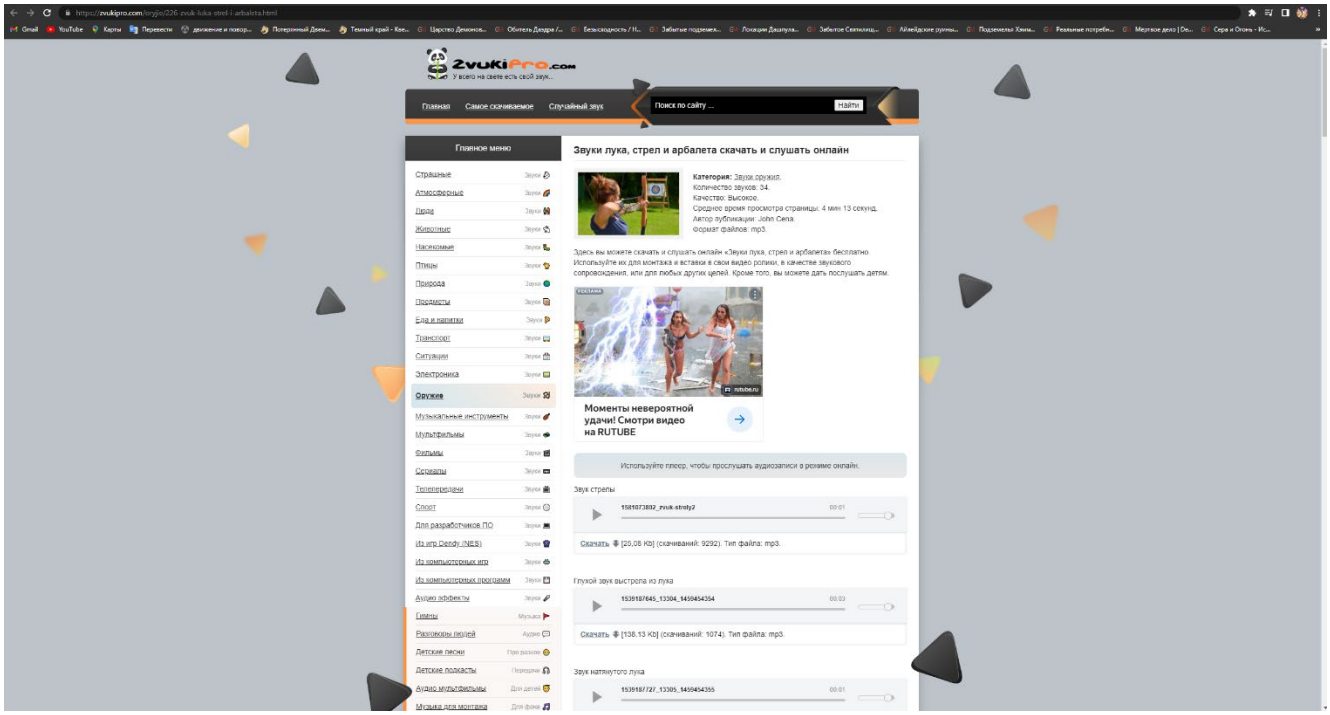


Рисунок 4.4 – Сайт Zvukipro

Для будь-якої гри одним із найважливіших компонентів є звукове супроводження. На мою думку, без якісного звуку будь-яка гра не здатна досягнути популярності. Саме тому я витратив значну кількість часу для пошуку якісного звуку.

Для проєкту мені необхідні були звуки смерті корабля героя та смерті ворогів, для останніх я вибрав два різні варіанти: для імперських та інопланетних кораблів. У сумі я додав три різні види звуків смерті. Також я додав звуки для двигуна гравця та звук натискання на кнопки інтерфейсу. Найбільше звуків я додав для вибухів різних видів снарядів та пострілів гармат. Ці звуки я помістив у різні папки проекту: DeathsOfShips, Engines, Explosions, Shots та UI.

Далі я перейшов до пошуку ефектів, 3D моделей та UI елементів. Їх я знайшов у вбудованому магазині Unity Asset store.

Unity Asset Store - це онлайн-магазин, який пропонує розробникам ігор, які використовують Unity, різноманітні ресурси, такі як готові моделі, матеріали, текстури, звукові ефекти, скрипти та інші інструменти для покращення процесу розробки ігор.

В Asset Store можна знайти безкоштовні та платні ресурси, зокрема, готові шаблони гри, які можна використовувати як основу для своєї гри, або різноманітні додаткові компоненти для розширення функціональності гри. Asset Store є важливою частиною екосистеми Unity, яка допомагає розробникам створювати більш якісні ігри за менший час.

Спочатку я знайшов UI елементи, які, на мою думку, найкраще підходять для сетінгу моєї гри. Набір UI елементів, які я обрав, зображений на рисунку 4.5.

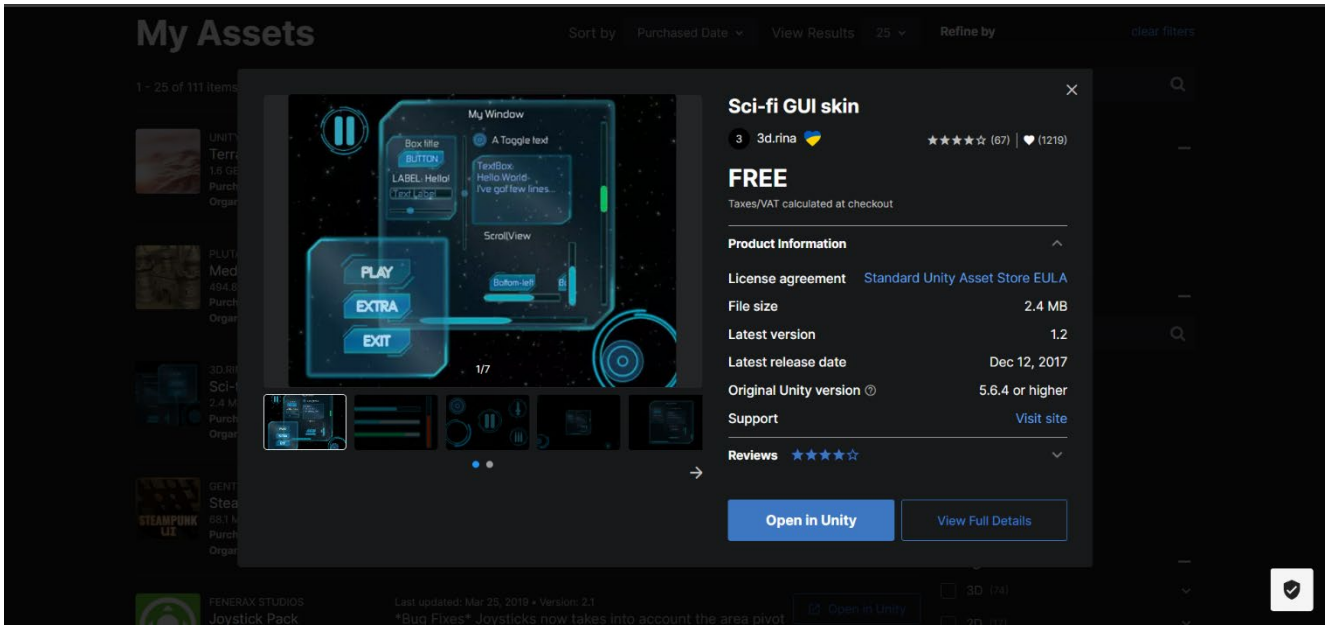


Рисунок 4.4 – Набір UI елементів в Assets Store

Після цього я додав у проєкт набір 3D моделей кораблів, зброї та компонентів сцени, ефекти пострілів, вибухів та польоту снарядів. Усі компоненти, окрім ефекту двигуна, я скачав безкоштовно. Ефекти двигунів мені довелось купити, тому що безкоштовні версії ефектів виявились недостатньо якісні.

На пошук усіх компонентів гри я витратив 1/3 часу розробки проєкту. Стільки часу я витратив тому, що займався підбором моделей та матеріалів, що найкраще підходили б за сетингом один до одного, моделі я також підбирав, зважаючи на кількість полігонів, з метою зменшити навантаження на пристрій користувача. Також через необхідність вибору безкоштовних ассетів кількість можливих варіантів вибору зменшувалася в десятки разів, що також ускладнювало процес пошуку.

У результаті папка з усіма асетами зайняла 1,25 ГБ місця пам’яті мого комп’ютера, тоді як папка Assets, яка містить основні компоненти мого проєкту, займає 1,30 ГБ пам’яті.

4.3 Розробка проєкту

4.3.1 Головні класи гри

Розробка мого проєкту складалася з трьох послідовних процесів: написання алгоритму, тестування та виправлення помилок. Розпочав я проєкт з розробки «точки входу» та основних класів, що містять усю основну інформацію про рівень.

Загалом у мене є 3 класи з даними: **LevelData** – основний скрипт даних, що містить інформацію про розміри сцени, про кількість метеоритів на сцені, про ворогів та про поточну хвилю ворогів. **ObjectsComposition** - цей скрипт містить списки з об’єктами, що були створені з допомогою PoolObject шаблону та поміщені в особисті пули. **PrefabsStorey** - у цьому класі зберігаються префаби об’єктів, які використовуються іншими скриптами. Наприклад, для створення метеоритів на сцені використовується префаб з цього класу.

Керує усім процесом на сцені один клас, що і є «точкою входу». Клас **LevelController** – це єдиний клас для роботи зі сценою, що наслідує MonoBehaviour, розміщений на самій сцені та містить методи Start, Update та інші. У ньому я підключив класи:

- WaveController- відповідає за хвилі ворогів;
- FollowForPlayer – відповідає за контроль камери;
- ShopAreaController – відповідає за магазин (не працює з UI);
- CreatingController – відповідає за створення нових об’єктів на сцені.

Маніпулюючи з методами цих класів, клас **LevelController** керує рівнем, користуючись вбудованими методами Unity, такими як Update, Start та інші. Наприклад, у методі Start ми звертаємось до класів CreatingController, щоб створити сцену, та WaveController, щоб розпочати першу хвилю ворогів. Приклад методів нижче:

```
private void Start()
{
    if (_levelData == null)
```

```
    {
        _levelData = LevelData.instance;
    }

    SetTheScene();
}
private void SetTheScene()
{
    Time.timeScale = 0;

    _creatingController.CreateScene();
    _waveController.SpawnFirstWave(_levelData);
}
```

Цей метод викликається на сцені лише один раз, через що він був обраний для створення сцени та для початку хвилі ворогів. Також у класі були використані методи Update та FixedUpdate, які викликаються кожного кадру та кожну 0.02 секунду відповідно. Саме тому ці методи були використані для контролю камери у методі FixedUpdate та для контролю хвиль ворогів у Update. Також у методі Update присутній виклик методу, що відповідає за перевірку можливості використовувати магазин у поточний момент часу. Ці два методи взаємодіють між собою через змінну IsPeacefulTime, що знаходиться у класі **LevelData** та змінюється залежно від стану рівня (іде бій, мирний час). Також для того, щоб хвилі ворогів починались не одна за одною, а з певним інтервалом часу, під час якого можна скористатись магазином, була використана корутина TimeBetweenWaves.

Корутина (Coroutine) - це легковажна нитка виконання, яка дозволяє призупиняти виконання функції і повернутися до неї пізніше, зберігаючи її стан.

У Unity корутини часто використовуються для виконання довготривалих або ітеративних задач, таких як затримки (delay), анімації, загрузка ресурсів тощо, без блокування іншої логіки гри.

Корутина починається з ключового слова «IEnumerator» і використовується за допомогою функції «StartCoroutine», яка запускає її в окремій нитці виконання. У тілі корутини можуть бути використані ключові слова «yield return», що дозволяють призупинити виконання і повернути значення до основного потоку виконання.

Приклад корутини у моєму коді:

```
public IEnumerator TimeBetweenWaves(float timeOfPeace)
    {
        yield return new WaitForSeconds(timeOfPeace);

        NewWave newWave = new NewWave();

        newWave.StartNewWave(_waveController, _levelData);

        StopCoroutine(_coroutine);
    }
```

Наступним етапом розробки гри стала розробка класів корабля гравця. Для оптимізації коду я помістив на корабель 2 класи. Перший - це клас, що відповідає за збереження даних корабля, а другий - за контроль над кораблем. Назви цим класам я надав відповідно до їх функцій: **PlayerData** та **PlayerController**.

Клас **PlayerData** містить основні параметри гравця, такі як життя, щит, швидкість поточна та максимальна. Також він містить список зброї, якою гравець володіє та список з ворогами, що присутні на сцені. Список з ворогами використовується переважно для наведення самокерованих ракет, а список зі зброєю використовується для маніпуляцій над зброєю у магазині та на сцені.

Так само клас **PlayerController** маніпулює з даними зі скрипту, названого вище, та методами з підключених класів, таких як:

- EffectController – відповідає за керування ефектів;
- Sight – відповідає за контроль прицілу;

- SeterOfHPAndShield – відповідає за встановлення життів та щита;
- InputKeyboardController – відповідає за роботу з клавіатурою (якщо використовується ПК);
- InputJoysticksController – відповідає за роботу з джойстиком (якщо використовується смартфон);
- InputMouseController – відповідає за роботу з мишкою (якщо використовується ПК);
- HealtheAndShieldController – відповідає за контроль життів та щита.

Клас **PlayerController,** як і **LevelController,** використовує Start, Update та FixedUpdate. Окрім них, він містить методи Unity: LateUpdate та OnTriggerEnter.

LateUpdate є однією з функцій життєвого циклу скрипта в Unity. Ця функція викликається після виконання всіх функцій Update() у всіх об'єктах на сцені.

Таким чином, LateUpdate() дає можливість оновлювати позицію, орієнтацію та інші параметри об'єктів на сцені, які були змінені під час виконання функцій Update() перед остаточним оновленням кадру.

Наприклад, якщо ви хочете перемістити камеру за гравцем, який рухається, ви можете змінювати позицію камери під час виконання функції Update() гравця. Однак, якщо ви використовуєте LateUpdate(), то зміни, які ви здійсните в цій функції, будуть застосовуватися до кадру після того, як будуть виконані всі функції Update().

Крім цього, LateUpdate() також може бути корисним, якщо вам потрібно взаємодіяти з іншими об'єктами на сцені, які вже оновлені, під час виконання функцій Update() у цих об'єктів.

OnTriggerEnter - це метод, який викликається, коли об'єкт з тригер-колайдером зіштовхується з іншим об'єктом, який також має тригер-колайдер.

Тригер-колайдер - це компонент у Unity, який не має фізичної поведінки, але може бути використаний для виявлення зіткнень з іншими об'єктами на сцені.

Коли два тригер-колайдери зіштовхуються, метод OnTriggerEnter() викликається на скрипті, який прикріплений до об'єкта з тригер-колайдером.

Клас **PlayerController** використовує метод LateUpdate для переміщення прицілу гравця, а метод OnTriggerEnter - для перевірки взаємодії з об’єктами сцени. Наприклад, у моєму випадку відбувається перевірка, чи був об’єкт, з яким ми зіштовхнулись, об’єктом для збільшення характеристик (життя, щит).Метод Start використовується для отримання даних об’єкта та для вимикання можливості переміщення по осі Y. У методі Update я виконую перевірку життів та щита гравця та їх перезаписання у випадку зміни під час гри. Також у цьому методі відбувається запуск методу PlayerDeath, який відповідає за процес смерті та програшу гравця, якщо його життя опустилось нижче 0.

Так само у методі FixedUpdate відбувається перевірка, чи є пристрій, на якому запущена гра, смартфоном або ПК. І залежно від пристрою виконується метод класу InputKeyboardController та InputMouseController або InputJoysticksController, у яких виконується перевірка даних пристрою введення та використання цих даних для переміщення, повороту корабля або для пострілу зброї.

Наступним етапом розробки я вибрав написання скриптів для ворога. Архітектура коду ворожих кораблів досить схожа на архітектуру корабля гравця. На корабель так само поміщено скрипти для даних та керуючий скрипт, що називаються: **EnemyController** та **DataOfEnemies**.

DataOfEnemies, як і клас даних гравця, містить основну інформацію про корабель, але також у ньому міститься ScriptableObject, що містить статичні дані, що призначаються до запуску гри у інспекторі.

ScriptableObject є класом у Unity, який дозволяє створювати об'єкти, що зберігають дані в пам'яті, але не потребують інстанціювання у сцені.

Клас ScriptableObject можна використовувати для зберігання інформації, яку необхідно зберігати між сценами, або для створення налаштувань, що можуть бути змінені без необхідності зміни самого коду.

Наприклад, якщо у вас є гра, яка має кілька рівнів, ви можете створити об'єкт ScriptableObject, який зберігатиме інформацію про поточний рівень гри. При переході до іншого рівня, дані будуть зберігатися у пам'яті, і ви можете використовувати їх у наступному рівні без необхідності повторного завантаження з диску.

Так само другий головний скрипт ворожого корабля, що називається **EnemyController,** досить сильно повторює алгоритм **PlayerController**. Єдина різниця лише у тому, що замість даних з клавіатури, джойстика або миші клас використовує для переміщення **AIMoverController**, що використовує 3 скрипти, які відповідають за поведінку AI, та взаємодіє з NavMesh системою. Для пострілу зі зброї клас перевіряє не натискання на клавішу миші, а відстань до гравця.

4.3.2 Алгоритм створення та налагодження сцени

Після запуску сцени і перед початком самого процесу гри необхідно підготувати усі компоненти сцени. У моєму випадку мені необхідно створити компоненти та помістити їх у окремі списки об’єктів, щоб потім використовувати на сцені, не створюючи їх з початку. Цей тип алгоритму називається Object Pool.

Пул об'єктів (англ. Object Pool) - це підхід у програмуванні, який дозволяє створювати певну кількість об'єктів заздалегідь та повторно використовувати їх замість того, щоб створювати нові об'єкти при кожному запиті.

В Unity пул об'єктів можна використовувати для ефективного управління об'єктами, які можуть бути створені та знищені багато разів під час гри. Наприклад, якщо ви створюєте багато куль, ви можете використовувати пул об'єктів, щоб не створювати нові об'єкти кожен раз, коли гравець вистрілює.

Використовуючи цей шаблон, я створюю ворогів, звуки, ефекти та снаряди для гармат. Також при створенні сцени вона заповнюється метеоритами різних розмірів. За їх створення відповідає клас **SpawnOfMeteorites**.

Сам код зображений на рисунку 4.5.

```
using UnityEngine;

1 reference
public class SpawnOfMeteorites : AFirstSpawn
{
    private const int yPosition = 0;

    private LevelData _levelData;

    2 references
    public override void FirstSpawn()
    {
        _levelData = LevelData.instance;

        for (int i = 0; i < _levelData.NumberOfMeteorites; i++)
        {
            Randomizer randomizer = new Randomizer();
            float xPosition = randomizer.RandomXForSpawn(_levelData.XLevelSize);
            float zPosition = randomizer.RandomZForSpawn(_levelData.ZLevelSize);

            GameObject obj = randomizer.RandomObject(PrefabsStorey.instance.Meteorites);
            GameObject meteorite = Object.Instantiate(obj, new Vector3(xPosition, yPosition, zPosition) , Quaternion.identity);
            meteorite.transform.SetParent(LevelData.instance.MeteoriteConteiner);

        }
    }
}
```

Рисунок 4.5 – приклад коду SpawnOfMeteorites

У цьому скрипті відбувається створення заданої кількості метеоритів з довільними розмірами та довільним оберненням у просторі. Клас **SpawnOfMeteorites** наслідує абстрактний клас **AFirstSpawn**. Це зроблено для реалізації гнучкого коду. **AFirstSpawn** також наслідують інші класи, що відповідають за початкове створення об’єктів на сцені, або класи, що будуть додані у майбутньому. Завдяки тому що всі класи, які відповідають за створення перших об’єктів на сцені, наслідують **AFirstSpawn,** ми можемо у майбутньому, за необхідності, додати новий клас для створення об’єктів у список типу **AFirstSpawn**, що розміщений у головному класі **CreatingController,** та отримати результат. Це дозволяє нам модифікувати гру, не змінюючи вже створені коди.

Приклад коду, що відповідає за послідовний запуск методів створення об’єктів, зображений на рисунку 4.6.

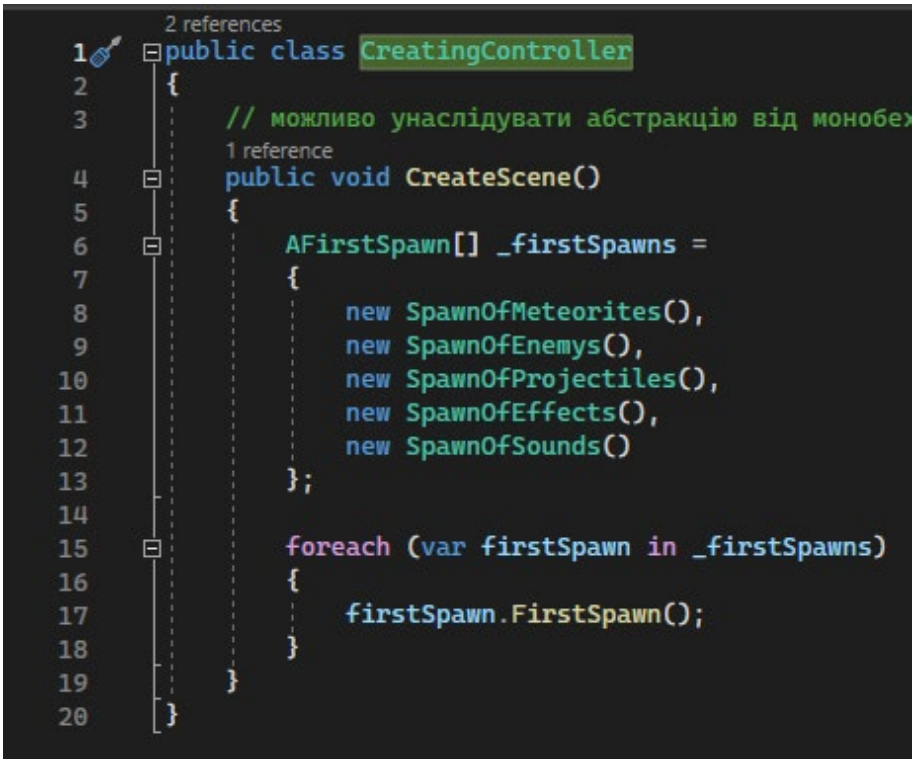


Рисунок 4.6 – приклад коду CreatingController

4.3.3 Алгоритм зброї

Після розробки основних алгоритмів ворога, сцени та корабля гравця я розпочав розробку алгоритмів зброї. Під час розробки я намагався реалізувати максимально зручний та гнучкий для модернізацій код, задіявши при цьому мінімальну кількість ресурсів.

Для початку я створив інтерфейс **IStandardShoot,** який реалізують усі класи, що відповідають за зброю. Завдяки цьому, якщо ми хочемо додати новий тип гармат, ми можемо просто прикріпити зброю з новим класом, що реалізовує цей інтерфейс, у список гармат корабля і, пройшовшись по цьому списку, викликати методи інтерфейсу Shoot(), який реалізовують класи зброї. Приклад такого виклику виглядає так:

```
for (int i = 0; i < gunList.Count; i++)
    {
    if (gunList[i].activeInHierarchy)
```

```
        {
            IStandardShoot standardShoot =
gunList[i].GetComponent<IStandardShoot>();
            standardShoot.Shoot();
        }
    }
```

Цей приклад коду може використовуватись не лише гравцем, але і ШІ саме завдяки тому, що у прикладі ми викликаємо метод інтерфейсу, який реалізовують класи, що відповідають за зброю. Таким чином ми можемо створити будь-який вид зброї: ніж, вогнемет, лазерну зброя чи іншу, і незалежно від того, як саме працює їхній код, ми можемо використовувати їх, не змінюючи вже готові коди проекту, що дозволяє позбутись випадкових створень помилок у проєкті.

Приклад реалізації інтерфейсу зброєю, що стріляє снарядами різного типу, зображений на рисунку 4.7.

```
3 references
public void Shoot()
{
    if (_dataOfGun.IsCharged)
    {
        GameObject bullet = GetObject();
        if (bullet != null)
        {
            CreateProjectile createProjectile = new CreateProjectile();

            UseShotEffect();
            UseShotSound();
            SetTransform(bullet);
            SetScriptableObjAndLayer(bullet);
            createProjectile.CustomizeProjectile(_dataOfGun, bullet);

            _dataOfGun.IsCharged = false;

            _coroutine = StartCoroutine(WaitShoot(_dataOfGun.ReCharge));
        }
    }
}
```

Рисунок 4.7 – приклад реалізації методу інтерфейса IStandardShoot

Це приклад основного типу зброї, що використовується у моєму проєкті. Працює він так: після виклику цього метода (з класу, що відповідає за перевірку

натискання на кнопку миші) відбувається перевірка, чи є зброя перезаряджена, отримавши зміну IsCharged з класу, що відповідає за дані зброї: **DataOfGun.** У випадку, якщо ця зміна повертає true, тоді відбувається отримання снаряду, що підходить саме для цієї зброї, звертаючись до методу GetObject(), приклад якого нижче:

```
private GameObject GetObject()
   {
      ObjectsComposition ammunitionStore = ObjectsComposition.instance;
      GameObject bullet = ammunitionStore.PoolStandardBullets;
      return bullet;
   }
```

Після отримання снаряду у коді викликаються методи UseShotEffect() та UseShotSound(), які реалізують звуки та ефекти пострілу.

Після виконання ефектів відбувається перенесення отриманого снаряду у позицію дула зброї у методі SetTransform() і задається ScriptableObject та шар снаряда. Після чого викликається метод, що, використовуючи данні з ScriptableObject, налаштовує снаряд.

Мій алгоритм використовує приклад з реального життя. При розробці я відтворив метод пострілу справжніх танків, які використовують снаряди, що програмуються. Танк виконує постріл снарядом, а потім програмує його, передаючи координати цілі, необхідну швидкість тощо. Моя зброя у грі працює за таким самим алгоритмом. Код переносить оболонку снаряда у місце пострілу (дуло гармати), після чого вмикає її, щоб куля почала рух, і програмує її через методи SetScriptableObjAndLayer(), що, як було сказано вище, передає дані для боєприпасу, та CustomizeProjectile(), що виконує налагодження боєприпасу.

Після пострілу у поле IsCharged з класу **DataOfGun** записується false, що означає, що зброя розряджена, а потім виконується корутина, що відповідає за час перезаряджання зброї.

4.3.4 Розробки інтерфейсу гри

Перед останнім етапом розробки я розробив інтерфейс гри. Це був досить стандартний процес. Відповідно до принципу єдиної відповідальності я маю класи, що відповідають за дані, та класи-керівники, що маніпулюють цими даними, щоб виконувати ті чи інші функції. Усі класи, що відповідають за UI в рівні, зображені на рисунках 4.8 та 4.9.

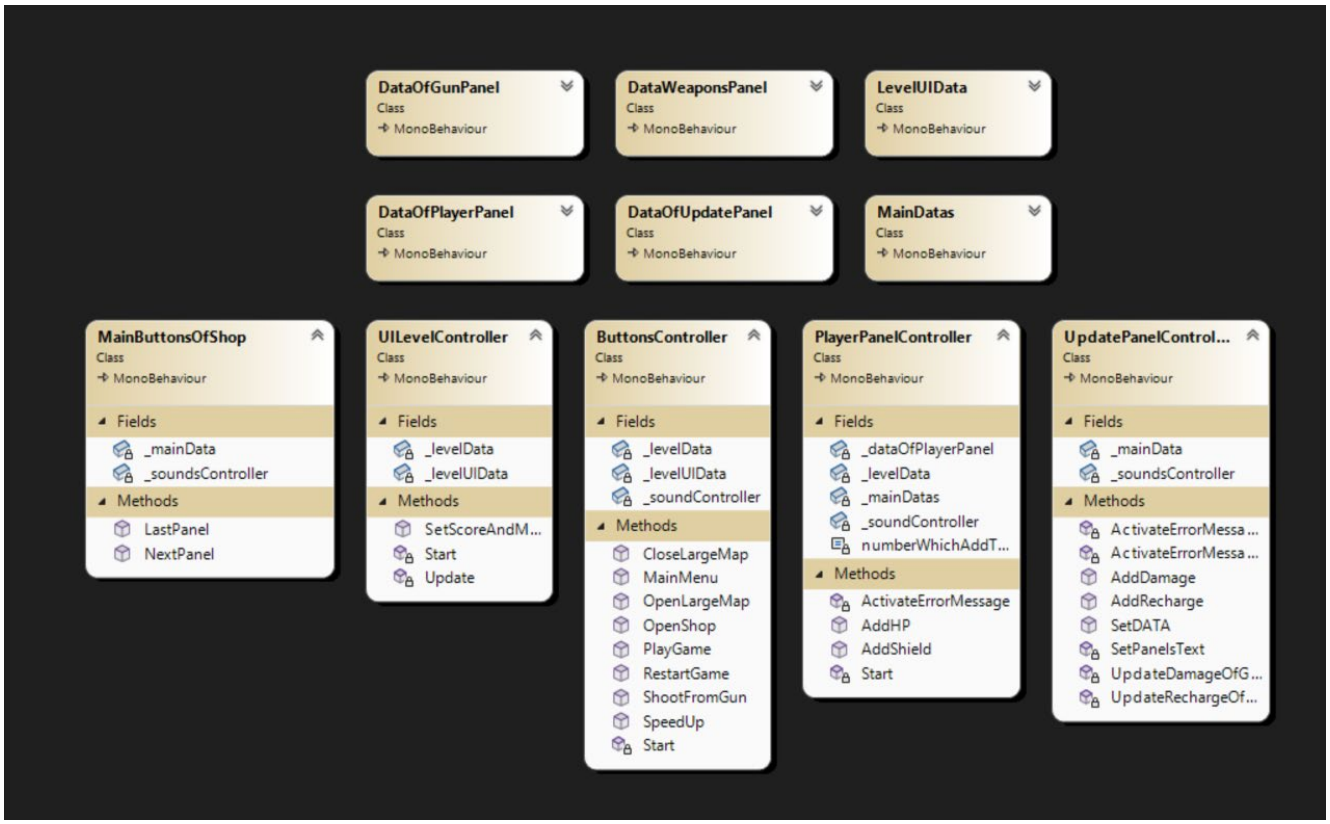


Рисунок 4.8 – діаграма основних класів UI

Рисунок 4.9 – діаграма другорядних класів UI

На рисунку 4.8 у верхній частині можна побачити класи, що відповідають за збереження даних, а нижче зображені класи, що відповідають за керування UI елементами на рівні. Всього у мене в проєкті є 6 класів з даними та 5 керуючих класів.

- MainButtonsOfShop – відповідає за основні кнопки в магазині.
- UILevelController – відповідає за відображення рахунку та грошей, також відображає панель програшу, якщо гравець гине.
- ButtonsController – відповідає за кнопки інтерфейсу гравця під час бою (мапа, кнопки пострілу, пришвидшення тощо).
- PlayerPanelController – керує панеллю покращення характеристик корабля гравця в магазині.
- UpdatePanelController – керує панеллю покращення зброї в магазині.

На рисунку 4.9 зображені другорядні класи, що виконують функції перемикання сторінок в магазині, закриття магазину, встановлення тих чи інших параметрів, купівлі, встановлення зброї, відкриття панелі покращення зброї чи

відображення повідомлень про нестачу грошей або максимальний рівень покращення.

Також є класи, що відповідають за UI на сцені меню гри. Вони та керівник меню зображені на рисунку 4.10.

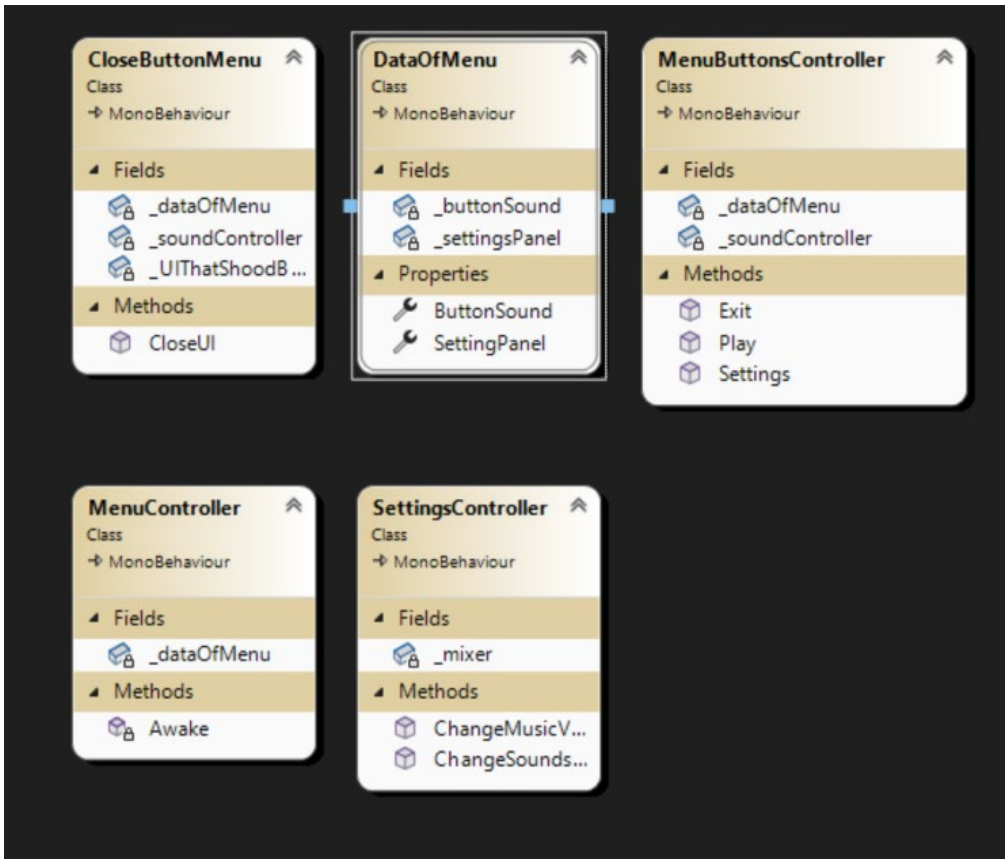


Рисунок 4.10 – діаграма класів меню

На цій діаграмі ми бачимо 4 класи, що відповідають за UI та 1, що відповідає за всі інші функції меню. Методом керування меню є **MenuController**, що у методі Awake вмикає елемент AudioSourse зі звуком для натискання кнопок.

Інші чотири класи відповідають тільки за інтерфейс.

- CloseButtonMenu – універсальний клас для кнопки закриття.
- DataOfMenu – клас, що зберігає усі дані меню.
- MenuButtonsController – відповідає за кнопки меню (початок гри, вихід, налаштування).
- SettingsController - відповідає за контроль панелі налаштувань.

4.3.5 Завершення розробки

Паралельно з розробкою UI елементів проєкту я займався виправленням помилок та налагодженням архітектури. Під час розробки проєкту мені доводилось час від часу змінювати архітектуру та деякі алгоритми програми.

Основні труднощі, з якими я зіткнувся, - оптимізація коду та його реалізація в максимально гнучкій та зручній для покращень формі. Під час роботи над проєктом я вивчав нові шаблони та алгоритми, які дозволили мені покращити код. Загалом структура мого коду виглядає так.

У мене є класи з приставкою у назві Data. Ці класи відповідають за збереження інформації. За контроль відповідають класи з приставкою Controller.

Завдяки класу **LevelController** відбувається створення рівня (локації та усіх компонентів рівня) та контроль за станом рівня (іде бій, мирний час, програш гравця чи відкритий магазин). Паралельно з цим працюють класи **PlayerController** та **EnamyController** що відповідають за отримання даних (з клавіатури, миші, джойстика, алгоритм ШІ) та їх використання у грі (переміщення, повертання у просторі, постріл). Також працює клас **GunController,** що розміщений на всіх гарматах на рівні. Він очікує на команду пострілу від того чи іншого корабля на сцені й у випадку команди пострілу виконує постріл та програмування боєприпаса.

Паралельно з цим працюють класи, що відповідають за інтерфейс гравця чи магазин.

ВИСНОВКИ

Комп'ютерні ігри мають значний вплив на економіку та суспільство, культуру та мистецтво. Вони можуть використовуватися в різних галузях, включаючи освіту, медицину, військову справу та інші. Зважаючи на важливість комп'ютерних ігор в сучасному світі, дослідження в галузі їх розробки, ігрової механіки та взаємодії з користувачами є дуже актуальними та цікавими.

У цій дипломній роботі була розроблена гра "LastBattle" з використанням ігрового рушія Unity. Гра створена в 3D просторі та у жанрі shoot 'em up, що дозволяє гравцеві контролювати космічний корабель, знищувати ворожі кораблі з використанням різних видів зброї та уникати ворожих атак.

Такі ігри не тільки цікаві та захопливі, а й мають високу складність та вимагають від гравців швидкості реакції та стратегічного мислення. Вони також можуть бути використані для розвитку та поліпшення різних навичок, таких як координація рухів, швидкість реакції та зосередженість.

Наукова новизна роботи полягає у використанні старого жанру shoot 'em up та його покращенні з використанням сучасних методів розробки ігор, що дозволяє підлаштувати гру під сучасні потреби індустрії.

У процесі розробки гри "LastBattle" були використані сучасні інструменти розробки ігор та програмування, такі як ігровий рушій Unity, мова програмування C# та сучасні шаблони програмування. Переваги Unity перед іншими ігровими двигунами полягають у його доступності, багатому наборі ресурсів та плагінів, а також у здатності створювати ігри для різних платформ.

Було проведено дослідження та аналіз сучасних тенденцій у галузі розробки ігор та аналогів гри, що дозволило зробити гру більш цікавою та захопливою для гравців.

Під час розробки гри були розроблені 2 сцени: меню та головна сцена. Було розроблено 4 типи ворогів з різними варіантами AI та два види рас ворожих кораблів. Також розроблено корабель гравця з великою кількістю різної зброї, що може бути модернізована у магазині. Магазин дозволяє купувати нову зброю,

міняти її, покращувати, а також покращувати й сам корабель. Для гри використовувався 3D простір з реалістичною графікою у фантастичному жанрі. У результаті було створено повністю робочу гру з урахуванням усіх потреб сучасної індустрії.

Оцінка розробленої гри показала, що "LastBattle" є цікавою та захопливою, має приємну графіку та звукові ефекти, а також добре працює на різних платформах.

Отже, розробка гри "LastBattle" з використанням ігрового рушія Unity є важливим кроком у розвитку галузі розробки ігор та їх використанні в індустрії.

ПЕРЕЛІК ПОСИЛАНЬ

1. Bernal-Merino M. Á. On the Translation of Video Games / Bernal-Merino M. Á. – Barcelona.: JoSTrans, The Journal of Specialised Translation, 2006. – 216 с.
2. Developing 2D Games with Unity: Independent Game Programming with C#/ Jared Halpern. - 2018 p. - 44c.
3. Game design/ Alpina Publisher. - 2019 p.-102 c.
4. Head First C#: A Learner's Guide to Real-World/Jill Elison Garth and Andrew Stellman - 208p. -50 c.
5. Thayer A. Localization of digital games: The process of blending for the global games market. Technical Communication / Thayer A. – Oxford: Lancer Books, 2004. – 19 с.
6. Unity in Action: Multiplatform Game Development in C# with Unity 5/ Joe Hocking//2018 p. - 80 c.
7. Збірник звуків та аудіо ефектів [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://zvukipro.com/oryjie/226-zvuk-luka-strel-i-arbaleta.html
8. Building a NavMesh [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html
9. C# ООР [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.w3schools.com/cs/cs_oop.php
10. List of shoot 'em up (Shmup) games [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.pcgamingwiki.com/wiki/List_of_shoot_%27em_up_(Shmup)_games
11. Object Pooling in Unity 2021 is Dope AF [Електронний ресурс]: [Відео]. – Електронні дані. – Режим доступу: https://www.youtube.com/watch?v=7EZ2F-TzHYw

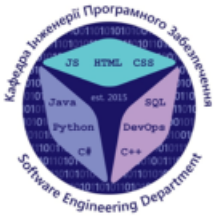
12. ParticleSystem [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.unity3d.com/ScriptReference/ParticleSystem.html
13. Shoot 'em up [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://en.wikipedia.org/wiki/Shoot_%27em_up
14. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://evergreens.com.ua/ua/articles/uml-diagrams.html
15. Unified Modeling Language [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://uk.wikipedia.org/wiki/Unified_Modeling_Language
16. Unity Documentation [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.unity3d.com/Manual/index.html
17. What Makes a Good Video Game? 4 Key Elements [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.pluralsight.com/blog/film-games/what-makes-a-great-game-the-key-elements-of-successful-games
18. 3 Simple Steps To Improve Your Game’s Graphics [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://gameanalytics.com/blog/3-steps-to-improve-your-games-graphics/

ДОДАТОКА

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



Розробка гри "LastBattle" з використанням ігрового рушія Unity мовою C#

Виконав студент 4 курсу
групи ПД-41
Марковський Павло Павлович
Керівник роботи
доцент кафедри ІПЗ, доктор філософії
Дібрівний О. А.

Київ – 2023

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: збільшення зацікавленості гравців жанром shoot 'em up шляхом створення гри "LastBattle", розробленої з використанням сучасних технічних засобів на основі аналізу переваг і недоліків ігор-аналогів.

Об'єкт дослідження: процес розробки геймплею, що базується на симуляції космічних битв.

Предмет дослідження: технології розробки відеогри за допомогою рушія Unity.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати існуючі ігри в жанрі shoot 'em up та визначити їх переваги й недоліки.
2. Ознайомитися з існуючими технічними засобами для розробки ігор та вибрати ті, що найбільше підходять для виконання проєкту.
3. Визначити основні параметри гри (жанр, простір, вид графіки, ігрові особливості).
4. Спроєктувати гру, застосовуючи вибрані параметри гри.
5. Розробити гру, використовуючи вибрані засоби розробки.

АНАЛІЗ АНАЛОГІВ

	Galaga	**R-Type**	**Gradius**	**LastBattle**
Переваги	• Класична аркадна гра зі швидким темпом гри • Простий, але веселий геймплей • Легко опановується початківцями	• Інноваційна система зброї та унікальні кораблі гравця • Графічно вражаючий дизайн та ефекти • Великий рівень виклику та складності	• Глибока стратегічна гра, що вимагає тактичного мислення • Різноманітність озброєння та посилень • Захоплюючі бос-битви	• **Великий вибір зброї.** • **3D простір та растрова графіка** • **Розроблена для найпопулярніших платформ** • **Велика різноманітність ворогів та босів** • **Присутнє поступове збільшення динаміки та складності, що дає можливість грати і новачкам, і професіоналам**
Недоліки	• Обмежений варіант ігрових режимів • Може стати повторюваним через обмежений контент • Відсутність сюжетної лінії • Застаріла піксельна графіка	• Деякі рівні можуть бути надто важкими або несправедливими • Може бути недостатньо доступним для новачків через незрозумілий інтерфейс та велику кількість UI елементів • Вимагає багато повторних спроб та вивчення рівнів для проходження гри • Застаріла піксельна графіка • Можливо грати лише на платформі Arcade	• Відсутність збереження прогресу та обмежена кількість життів • Не надає великої кількості інновацій та унікальності в зброї • Застаріла піксельна графіка • Відсутня версія для телефонів	• **Наявний лише один рівень** • **Відсутність можливості змінювати корабель гравця** • **Мала кількість посилень** • **Відсутність незвичайних здібностей для гравця**

ВИМОГИ ДО ІГРОВОГО КОНТЕНТУ

1. Реалізація гри у 3D просторі.
2. Наявність налаштувань звуку та музики.
3. Декілька варіантів расових особливостей для ворожих кораблів.
4. Різні види ворожих кораблів.
5. Наявність різних видів зброї для ворогів.
6. Поступове збільшення складності гри.
7. Можливість покращення корабля гравця та зброї гравця.
8. Наявність різноманітної зброї для корабля гравця.

КОНЦЕПТ ГРИ

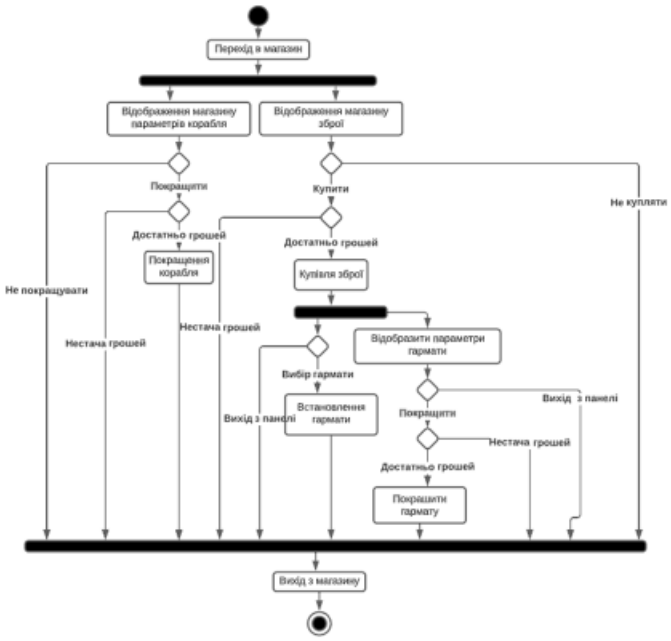
1. Гравець опиняється в космосі та керує космічним кораблем.
2. З часом з’являються хвилі ворогів, кількість та якість яких росте з кожною хвилею.
3. Головна мета гравця – прожити якомога довше та назбирати якомога більше балів за знищення ворогів.
4. Гравець після кожної хвилі ворогів може скористатися магазином, щоб покращити корабель або купити та покращити зброю.
5. Гра закінчується, коли корабель гравця знищують.

ПРОГРАМНІ ТА ТЕХНІЧНІ ЗАСОБИ РЕАЛІЗАЦІЇ

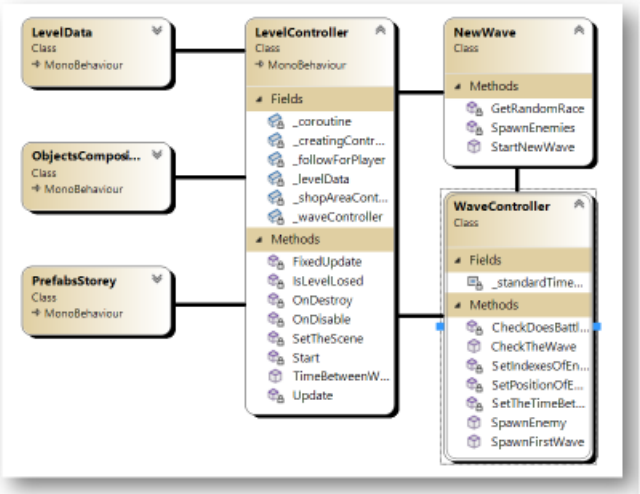
Visual Studio

Inkscape

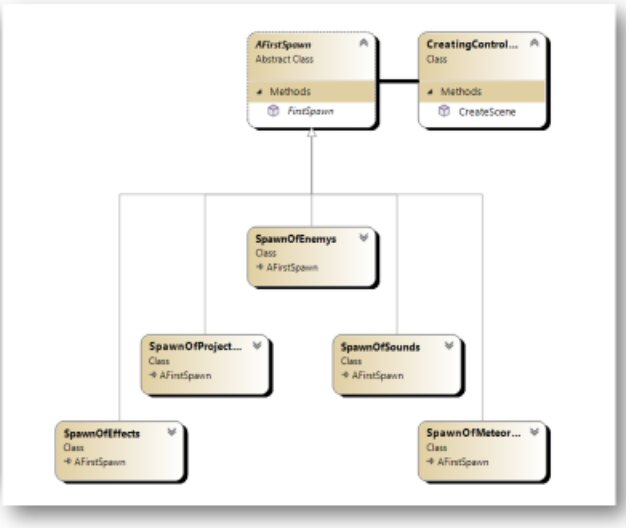
ДІАГРАМА ДІЯЛЬНОСТІ



ДІАГРАМА КЛАСІВ

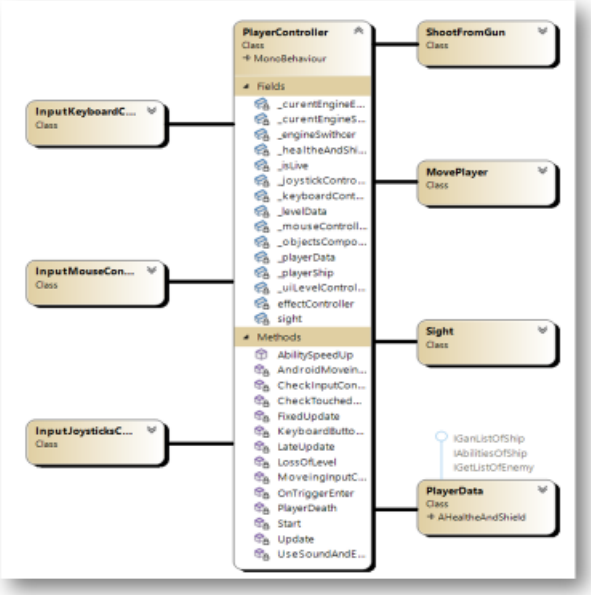


Класи головного рівня

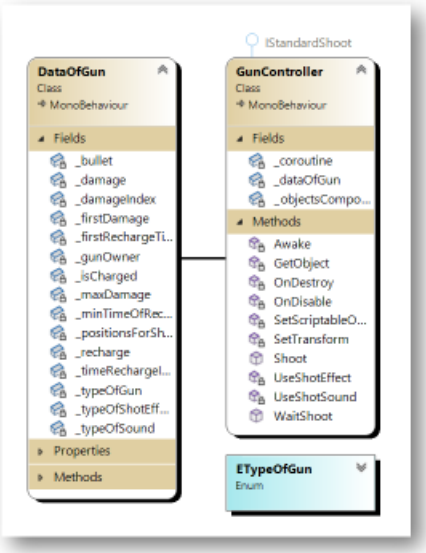


Класи створення рівня

ДІАГРАМА КЛАСІВ

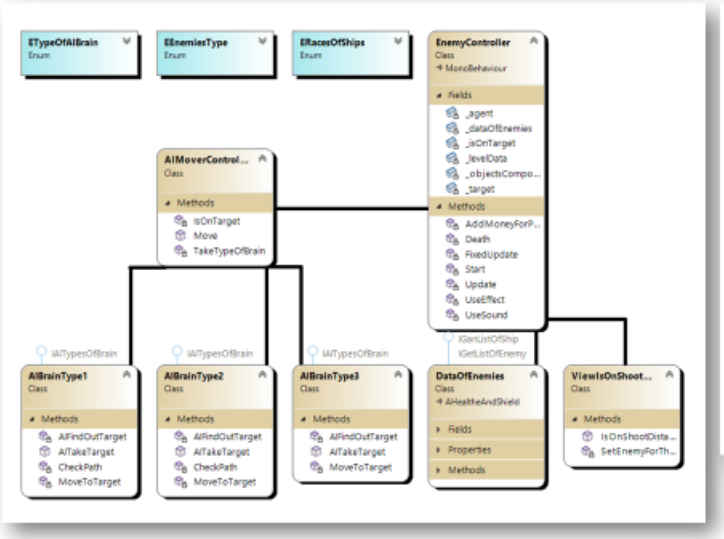


Класи гравця

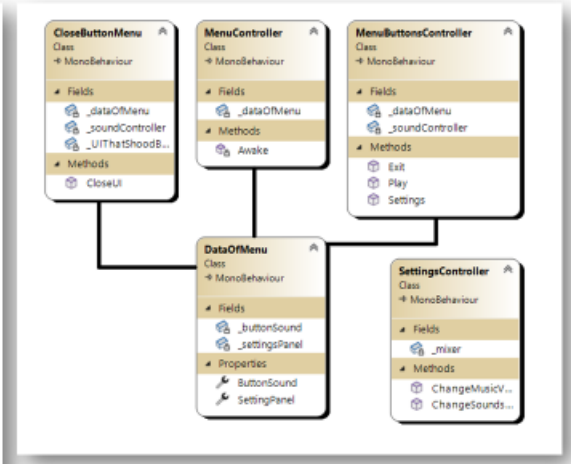


Класи зброї гри

ДІАГРАМА КЛАСІВ

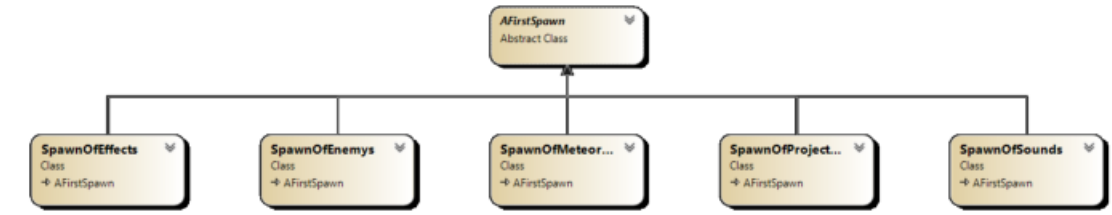


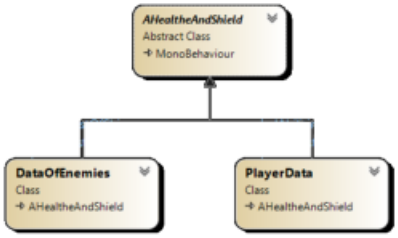
Класи ворогів



Класи меню

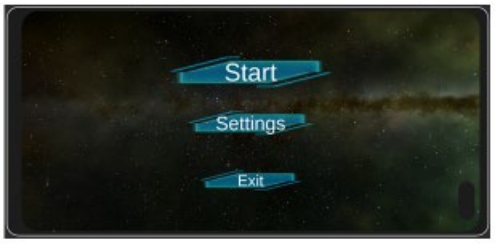
ДІАГРАМА КЛАСІВ НАЩАДКІВ





ДІАГРАМА ІНТЕРФЕЙСІВ

ІНТЕРФЕЙС КОРИСТУВАЧА



Інтерфейс меню

Панель налаштувань

Інтерфейс на головній сцені

Інтерфейс магазину

ІГРОВИЙ ПРОЦЕС

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Марковський П.П. Розробка гри "lastbattle" з використанням ігрового рушія unity мовою c# / О.А. Дібрівний, П.П. Марковський // Застосування програмного забезпечення в інфокомунікаційних технологіях: Матеріали всеукраїнської науково-технічної конференції. Збірник тез. 20.04.2023, ДУТ, м. Київ – К.: ДУТ, 2023 – С. 127-128.

2. Марковський П.П. Використання відеоігор в освіті / О.А. Дібрівний, П.П. Марковський // Сучасні інтелектуальні інформаційні технології в науці та освіті: Матеріали третьої всеукраїнської науково-практичної конференції. Збірник тез. Подано до друку.

ВИСНОВКИ

1. Проаналізовано існуючі ігри в жанрі shoot ‘em up і визначено їх переваги та недоліки. Основними перевагами були визначені велика кількість зброї, різноманітність ворогів та динамічність. Недоліки були визначанні наступні: застаріла графіка, часто поганий баланс, мала кількість доступних для гри платформ.
2. Проаналізовано та вибрано необхідні для розробки технічні засоби, такі як рушій Unity, ілюстратор Inkscape та середовище розробки Visual Studio.
3. Розроблено вимоги до гри, що базуються на перевагах та недоліках проаналізованих ігор-аналогів. Головними вимогами були вибрані: 3D простір, векторна графіка, баланс, що дозволить грати новачкам та професіоналам, велика різноманітність зброї та ворогів.
4. Спроєктовано та розроблено гру у жанрі shoot ‘em up з урахуванням аналізу ігор-аналогів, а саме: спроєктовані та розроблені структура проекту, алгоритми та рівні гри.
5. Виконано внутрішнє тестування гри з допомогою телефона та персонального комп’ютера та проведено навантажувальне тестування шляхом створення тестових ситуацій, що вимагають великого обчислювального навантаження.

ДЯКУЮ ЗА УВАГУ!