Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему РОЗРОБКА АРКАДНОЇ ГРИ ДЛЯ ОС ANDROID
DEVELOPMENT OF AN ARCADE GAME FOR ANDROID OS

Виконав: студент 4 курсу, групи КНТ-137
Спеціальності 121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Клименко Д.А
(прізвище та ініціали)
Керівник Зайко Т.А.
(прізвище та ініціали)
Рецензент Гофман Є.О.
(прізвище та ініціали)

2021

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка
(повне найменування закладу вищої освіти)

Інститут, факультет ІІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

_____С.О. Субботін

"____" _____20___року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Клименка Данила Андрійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка аркадної гри для ОС Android
Development of an Arcade Game for Android OS
керівник проєкту (роботи) Зайко Тетяна Анатоліївна, к.т.н., доцент ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від "30" березня 2021 року № 103
2. Строк подання студентом проєкту (роботи) 26 травня 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз актуальності дослідження. Постановка завдань. 2. Порівняння рушіїв для розробки гри. 3. Вибір інструментарію розробки гри "Shoot Him". 4. Розробка програмної частини гри "Shoot Him". 5. Тестування гри

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Зайко Т.А., доцент		
Нормоконтроль	Липовець М.В., асистент		

7. Дата видачі завдання “ 19 ” квітня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз актуальності дослідження	2 тиждень	Розділ 1
3	Проаналізувати і порівняти існуючі рушії для розробки гри	3 тиждень	Розділ 2
4	Обрати інструментарій для розробки гри	4 тиждень	Розділ 3
5	Розробка програмної частини гри	5 тиждень	Розділ 4
6	Тестування гри і виправлення помилок	6 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	7 тиждень	Додатки
8	Нормоконтроль та рецензування	8 тиждень	
9	Захист роботи	9 тиждень	

Студент _____ Клименко Д.А.

(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Зайко Т.А.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 62 с., 2 табл., 34рис., 2 дод., 15 джерел.

ГРА АРКАДА НА ОС ANDROID, 2D TOP-DOWN ШУТЕР, VISUAL STUDIO, C#, UNITY.

Об’єкт дослідження – програмні засоби для розробки гри.

Предмет дослідження – методи розробки ігор для мобільних пристроїв.

Мета роботи – розробка гри для мобільних пристроїв.

Матеріали, методи та технічні засоби: об’єктно-орієнтована мова програмування C#, середовище розробки Microsoft Visual Studio, персональний компьютер з процесором Intel Core i5 під управлінням операційної системи Microsoft, 64-розрядна система.

Результати. Створено аркадний 2D шутер “Shoot him” для ОС Android мовою програмування C#, реалізовано збереження даних.

Висновки. Розроблено гру для проведення вільного часу.

Галузь використання – мобільні ігри.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 62 pages, 2 tables, 34 figures, 2 appendixes, 15 sources.

GAME ARCADE ON OS ANDROID, 2D TOP-DOWN SHOOTER, VISUAL STUDIO, C #, UNITY.

The object of research is software for game development.

The subject of research is methods of game development for mobile devices.

The purpose of the work is to development a game for mobile devices.

Matrices, methods and technical means: functional programming, C# programming, personal computer with Intel Core i5 processor with control of Microsoft operating systems, 64-bit system.

Results. Created a game “Shoot him” for Android - C # programming language, data storage is implemented.

Conclusions. the game is developed for spending free time.

The field of use - mobile games.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AS – Asset Store;
HP – Health Points,
TDS – Top-Down Shooter.

ВСТУП

Історія відеоігор починається у 1947 році, коли було написано першу ігрову комп'ютерну програму, й охоплює шість десятиліть. Власне відеоігри, які використовують відео для взаємодії із гравцем, зародилися в 1960-і. Частиною поп-культури відеоігри стали в кінці 1970-х з поширенням аркадних ігрових автоматів і домашніх ігрових приставок та ПК [1].

Індустрія відеоігор або індустрія інтерактивних розваг - економічний сектор, пов'язаний з розробкою, просуванням та продажем відеоігор. У неї входить велика кількість спеціальностей, за якими працюють тисячі людей по всьому світу [2].

З кожним роком ринок розваг помітно розширюється, а технології створення ігор стрімко розвиваються. За статистикою на 2020 рік у відеоігри грає близько 40% населення Землі. Комп'ютерні розваги сприяють більш насиченному життю людини, а також можуть покращувати розумові здібності. Також швидкими темпами розвивається кіберспорт, в якому вже є професійні спортсмени. Вони тренуються, беруть участь у змаганнях з чималими призовими (наприклад, призовий фонд головного турніру з Dota 2 у 2020 році перевищив 40 мільйонів доларів) [3].

Люди цінують свіжі ідеї та з задоволенням грають у нові ігри, тому розробка ігор завжди буде актуальна, а з часом навіть популярніше.

По всьому світу ігри використовують для підвищення навичок і здібностей людини. Як приклад, спеціальні ігри для пілотів літаків, інтерактивні та демонстраційні ігри для лікарів.

Наведені вище факти розкривають актуальність і користь сфери ігор у житті людини.

1 АНАЛІЗ АКТУАЛЬНОСТІ ДОСЛІДЖЕННЯ ТА ПОСТАНОВКА ЗАВДАНЬ

На сьогодні у світі налічується близько 210 тисяч ігор, але ігрова індустрія розвивається, технології не стоять на місці, більшість старих ігор втрачають актуальність, до того ж навіть зараз у більшості людей виникає дефіцит хороших ігор. Виходячи з цього, з упевненістю можно сказати, що для любої людини буде місце у сфері геймдеву. Фактично будь-яка конкуренція не має впливу на ваші успіхи, тому що у іграх люди цінують ідею та нові цікаві механіки.

У цій сфері кожен може проявити себе. Якщо ви композитор – здивуйте людей якісною та гарною музикою у грі. Маєте здібності до написання сценаріїв – люди с захопленням поринуть у цікавий сюжет. Тут немає обмежень, тому кожен зможе створити щось своє.

Дивлячись на показники доходу популярних ігр, можна зробити висновок, що ігрова індустрія досить прибуткове заняття і це чималий стимул почати займатися геймдевом.

За інформацією з інтернету за 2019 рік ігри принесли своїм творцям близько 110 млрд. доларів.

1.1 Статистика доходу різних платформ

На гістограмі нижче можемо бачити перевагу доходу з мобільної платформи (рис. 1.1) [4].

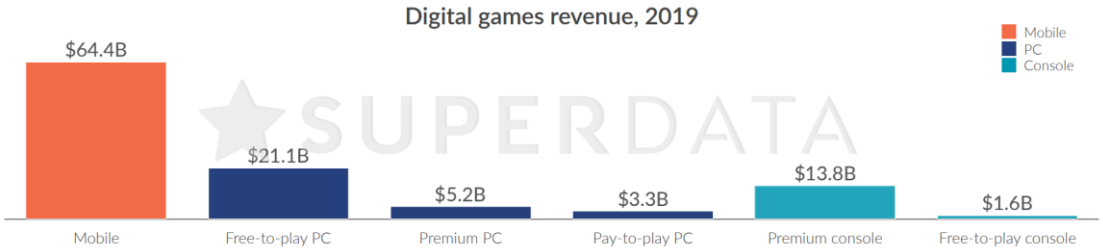


Рисунок 1.1 - Дохід різних платформ [4]

Розберімось чому так. Слід зазначити, що дохід прямо залежить від популярності платформи. В наш час майже у кожного є смартфон і основна перевага смартфонів перед іншими платформами полягає в тому, що смартфон завжди під рукою. Другим фактором популярності мобільної платформи є наявність великої кількості безкоштовних ігор, а ціна платних зазвичай не складає великої суми. Більшість сучасних ігор для смартфонів мають хорошу графіку і цікавий сюжет. Основний процент доходу є досягненням F2P ігор, а поширеними способами доходу є:

– реклама;
– преміум аккаунти ;
– battle pass;
– додатковий контент;
– обмежені за часом пропозиції;
– впровадження опитувань;
– інвестори.

Також треба звернути увагу на те, що на даній платформі існує ряд різних жанрів таких як:

– аркада;
– головоломка;
– гонка;
– екшн;
– спортивні;
– казуальні;
– карточні;
– три в ряд
– настільні;
– пригода;
– рольові;
– симулятори;
– стратегії.

Отже, мобільна платформа має багато переваг, підходить для користувачів різного віку і з різними вподобаннями. Кожен знайде чимало ігор за своїми інтересами і все це у зручному смартфоні, який завжди з собою.

1.2 Користь ігор у повсякденному житті

На сьогодні вже практично не має людини, яка не зіткнулася б з відеоіграми. Технології рухаються вперед, люди створюють різні програмні забезпечення для спрощення життя. Варто признати, що в наш час існує дуже багато корисних ігор. Дітям підійдуть пізнавальні та інтерактивні ігри, ігри-тренажери, а також деякі жанри, такі як:

– пригодницькі, в яких дитина зможе керувати подіями, зануритись в цікавий сюжет, шукати розгадки;

– головоломки, завдяки яким у дітей буде розвиватися мислення, а також навчить дитину писати і читати;

– симулятори, що дозволять дитині відчути себе пілотом літака, машиністом поїзда, або водієм автобусу.

Такі ігри сприяють розвитку пам’яті, розвивають увагу і моторику пальців, змушують мислити творчо.

1.2.1 Digital Combat Simulator

Пілоти американських штурмовиків A-10 Thunderbolt II почали наземні тренування за допомогою комп'ютерної гри Digital Combat Simulator (DCS). 355-я навчальна ескадрилья, розташована на базі ВПС США Девіс-Монтан використовує російський симулятор разом зі звичайною ігровою периферією для відпрацювання різних ситуацій в польоті.

Лабораторія симуляторів A-10, що відноситься до 355-ї ескадрильї, використовує спеціальні тренувальні зони, оснащені окулярами віртуальної реальності Oculus Quest VR, маніпуляторами Thrustmaster HOTAS Warthog, що

повторюють рукоятки управління , і педалями Thrustmaster TPR. Все обладнання було відкалібровано для роботи з DCS на тренувальному "шасі" від виробника спеціалізованих симуляторів Volar Sim.

До створення тренажерів A-10 наземна підготовка пілотів штурмовиків в Девіс-Монтані складалася лише з "навчання в класах, комп'ютерних уроків, і уроків на традиційних симуляторах. На віртуальних полях битв DCS пілоти відпрацьовують всі аспекти експлуатації літака, включаючи зльоти, посадки, дозаправку в повітрі, і використання озброєння. Також технологія VR використовується для демонстрації і розбору конкретних ситуацій, які було б неможливо розглянути на двомірних зображеннях.

Віртуальна реальність використовується в двох напрямках. З одного боку, вона задіяна в симуляції, а з іншого - VR-гарнітури використовуються для перегляду 360-градусного відео, записаного в ході реальних польотів. Проєкт зі створення "ігрових симуляторів" стартував в 2018 році, а сьогодні в Девіс-Монтані пілотам доступно 22 тренувальні зони [1].

1.2.2 Ігри для військових

Використання ігор давно взято на озброєння військовими. Ще в 1996 році на базі культової гри Doom II Корпус морської піхоти США розробив власну модифікацію, яка використовувалася для тренувань. А чеська компанія Bohemia Interactive, яка розробила гри Operation Flashpoint і Arma, поставляє версії своїх продуктів для сухопутних сил США.

З огляду на рівень графіки і фізики сучасних ігор, військові можуть отримати якісну і доступну симуляцію багатьох ситуацій. З одного боку, ігри можуть відтворити бойові ситуації, які не можна показати на навчаннях, а з іншого - запуск гри на порядок дешевше сухопутних навчань.

Використання DCS від компанії Eagle Dynamics обумовлено високим реалізмом льотної моделі, і відкритою структурою. Гра являє собою графічний движок і інтерфейс. Це дозволяє змоделювати будь-який сценарій в рамках гри.

Моделі штурмовиків Су-25 і A-10 Thunderbolt II спочатку отримали просунуті льотні моделі з умовним поділом літака на окремі частини, з можливістю докласти до кожної з них окремий момент сили. Це дозволяє реалістично симулювати різні ефекти, включаючи звалювання і штопор [1].

1.3 Розгляд та аналіз жанрів ігор

1.3.1 Загальні відомості жанрів

Для класифікації відеоігор використовується жанр відеогри. Сценарій, зміст, сюжет або сеттінг не визначає жанр. До того ж не існує єдиної класифікації жанрів, що виключає можливість точно визначати жанр будь-якої гри. Окрім того відоігри класифікуються за кількістю гравців, за платформами, та за іншими критеріями.

1.3.2 Класифікация жанрів

Розподіл відеоігор на жанри виконується за видом активності, яка найчастіше здійснюється в іграх даного жанру.

Найпопулярніші жанри:

а) симулятор - жанр характерною особливістю якого є точне відтворення фізичних законів реального світу властивостей реальних предметів, процесів та подій [5]. Нижче наведено більш детально про існуючі піджанри:

1) симулятор менеджменту та будування - це такий тип симуляційних відеоігор в якому гравець займається розбудовою чи розширенням спільнот, установ, імперій тощо з обмеженою кількістю доступних йому ресурсів, має реагувати на внутрішні проблеми (такі як злочини чи забруднення, як в SimCity), незалежні від дій гравця події (наприклад, стихійні лиха чи монстри, як в SimCity, або рівень необхідності тераформування планет) чи конкурувати з іншими гравцями. В однокористувацьких відеоіграх даного піджанру

проходження здебільшого є необмеженим у часі, а кінець може настати тоді, коли гравець сам того захоче, в той час як у багатокористувацьких відеоіграх проходження, як правило, триває допоки один із гравців не почне домінувати над іншими. Відеоігри жанру симулятора менеджменту та будування також можна розділити на два підтипи:

– симулятор бізнесу та симулятор містобудування. В першому випадку гравцеві необхідно буде займатися розвитком якогось конкретного підприємства, або цілої мережі підприємств, в той час як гравцям відеоігор жанру симулятора містобудування необхідно буде розширювати та розбудовувати вже цілу економічну систему, тобто міста чи поселення [6];

– симулятор життя - тип симуляційних відеоігор в якому гравець має можливість відігравати роль створіння чи групи створінь, взаємодіючи з іншими особинами, та подібні, інколи схожі на справжнє життя, дії. Найпершим симулятором життя вважають саме клітинний автомат «Життя», вигаданий Джоном Конвеєм 1970 року, а найпершою комерційною відеогрою - Little Computer People, випущеної 1985 року компанією Activision для платформи Commodore 64, яка згодом надихнула Вільяма Райта на створення серії The Sims, однієї з найвизначніших серій жанру симулятора життя, першу частину якої було випущено 2000 року;

– симулятор побачень, мета якого для гравця зосереджується на тому, щоб домогтися успіху в романтичних стосунках, гра в бога (симулятор бога), де гравець може взяти на себе роль бога, аби допомогти своїм шанувальникам у подальшому розвитку, цифрові домашні тварини, де гравець тренує, підтримує життя й спостерігає за симуляцією певної тварини, біологічні симулятори, в яких гравець може керувати істотами протягом кількох поколінь, прагнучи досягти

поставлених цілей, та соціальні симулятори, які відтворюють соціальні взаємодії між створіннями [6];

2) спортивний симулятор - тип симуляційних відеоігор, в якому відбувається принаймні часткова імітація процесу заняття спортом: вигаданим чи реальним. На відміну від більшості ігор, світ яких, гравцям швидше за все на початку є маловідомим, світ спортивних симуляторів безпосередньо пов'язаний із реальними спортивними подіями. Хоч, піджанр і є, більшою мірою, повною адаптацією реального спортивного заходу, ігровий процес таких ігор значно спрощується, особливо стосовно можливих помилок, допущених гравцями. Це пов'язано із тим, що гравець використовує для гри електронний пристрій, а не безпосередньо бере участь в спортивних заходах, що може значно впливати на розуміння різноманітних обставин. Першою грою жанру стала американська відеогра Tennis for Two 1958 року, яка симулювала гру в теніс чи пінг-понг на осцилографі. Згодом, був випущений культовий симулятор настільного тенісу, Pong. Серед інших, визначних відеоігор піджанру, можна вважати серію футбольних симуляторів FIFA, серію хокейних симуляторів NHL, Madden NFL (John Madden Football), Pro Evolution Soccer та інші [6];

б) стратегія - жанр, сенс якого полягає в плануванні дій, виробленні підходу і застосування тактики для досягнення перемоги. Гравець керує цілими підрозділами, підприємством або навіть всесвітом [5]. В залежності від умов і реалізації ігрового часу ігри цього жанру поділяються на два різновиди:

1) покрокові, де гравець та його супротивник або конкурент здійснюють дії покроково, по черзі. Кожен має обмежену кількість дій, що змушує гравців вибирати найкращі з можливих дій застосовуючи різні тактики [7];

2) в реальному часі, в яких ігровий процес є безперервним і гравці виконують дії одночасно. Такий піджанр більш динамічний ніж покроковий і потребує від гравця застосовувати тактику. Ігровий час

найчастіше не відповідає реальному і дії, які зайняли б години чи дні, у грі тривають хвилини [7];

в) рольова гра - жанр в якому гравець відігрує роль конкретного персонажа. Кожна роль має свої особливості, переваги і недоліки. Мета полягає у виконанні різних завдань для розвитку та просуванню по сюжету. Завдяння найчастіше корегуються відповідно до обраної ролі [5]. Поширеним видом рольової гри є MMORPG(Massively multiplayer online role-playing game), багатокористувацькі рольові онлайн-ігри, в яких гравці взаємодіють одне з одним у віртуальному світі. Відповідно до підвищення рівня персонажа можуть змінюватись глобальні параметри гри (підвищення рівня ворогів, їх навички та стиль поведінки). Основними способами отримання досвіду для персонажа є вбивство ворогів і виконання певних завдань [8];

г) екшн - жанр в якому необхідно використовувати рефлекси та реакцію. Це один із базових жанрів. Як правило більшість екшн-ігор пов’язані із агресивними діями щодо ворогів. Мета персонажа - битися, стріляти, переслідувати ціль. Якщо нявні елементи пригодницьких ігор, то такі ігри класифікуються як екшн-аркада і вирізняються простотою освоєння. Даний жанр поділяється на велику кількість піджанрів серед яких:

1) шутери, які вимагають від гравця боротися з суперником шляхом стрілянини;

2) файтинги, що імітують ближній бій;

3) платформери, в яких персонаж рухається, стрибаючи по платформах, і має долати перешкоди на своєму шляху;

4) лабіринти, сенс яких знайти вихід, частіше з обмеженням часу;

5) beat 'em up (побий їх усіх) - подібні на файтинги, з тією різницею, що персонажі вільно переміщуються ігровим світом (а не спеціальною ареною) і борються проти багатьох противників одночасно;

6) shoot 'em up (перестріляти їх усіх) - жанр в якому гравець бореться з великою кількістю ворогів за допомогою стрілянини [5].

1.4 Постановка завдання

Гра “Shoot Him” буде виконана в жанрі аркадного екшн-шутера для ОС Android, з 2D-графікою і видом зверху вниз. Сенс гри полягає в боротьбі з ворогом шляхом стрілянини. Основним функціоналом гри є здобуття монет з ворога, покупка снарядів з різними характеристиками.

1.5 Висновки за розділом 1

Сфера розробки ігор актуальна на сьогодні і має високі показники доходу. Ігри використовуються не тільки для розваг, а й у навчальних цілях, приносячи користь людям. Кожен жанр розвиває здібності людини, що прямо свідчить про важливість ігор у сучасному світі.

2 ПОРІВНЯННЯ РУШІЇВ ДЛЯ РОЗРОБКИ ГРИ

2.1 Критерії вибору рушія та їх порівняння

При виборі рушія будуть враховуватися такі критерії:

– складність використання та поріг входження;
– наявність навчальної літератури або відеороликів;
– вартість рушія та наявність обмежень на дохід з гри;
– наявність AS.

Розглянемо декілька поширених рушіїв:

а) Unity - найпопулярніший ігровий рушій на сьогоднішній день. З'явився в 2005 році. Велике поширення отримав з виходом версії Unity3D. На ньому зроблено велику кількість мобільних ігор та ігор в Steam, у тому числі і інді. У Unity найбільша кількість розробників, у більшості компаній процеси заточені саме під Unity (і під мову C# що використовується там). Крім того готових ассетів у Unity помітно більше ніж у будь-якого іншого рушія. Unity простий у використанні і не потребує досвіду, що дає можливість будь-кому вільно користуватися функціоналом даного рушія. В Unity використовується компонентно-орієнтований підхід, в рамках якого розробник створює об'єкти і до них додає різні компоненти. Завдяки зручному Drag & Drop інтерфейсу і функціональності графічного редактора рушій дозволяє малювати карти і розставляти об'єкти в реальному часі і відразу ж тестувати отриманий результат. Широкий спектр програмних засобів та багатоплатформність є ключовим фактором, тому Unity підходить як для інді компаній так і для великих компаній. Unity ідеально підходить для початківців, яким мало можливостей більш простих інструментів, і які, в той же час, не хочуть витрачатися на більш дорогі і просунуті рушії. Unity дозволяє швидко створити об'єкти, розставити і зв'язати їх, задіяти власний контент і вміст магазину Asset. AS Unity має велику базу різних ресурсів, як платних, так і безкоштовних. Так як рушій має величезну аудиторію користувачів, знайти рішення будь-якої проблеми не складе

труднощів - спільнота допоможе початківцям, офіційні, навчальні і призначені для користувача блоги дадуть всі необхідні знання. Ліцензія та вартість:

1) початкова версія - безкоштовна якщо обіг не перевищує 100 000 доларів;

2) Unity Plus - 35 доларів на місяць;

3) Unity Pro - 125 доларів на місяць [9];

б) Unreal Engine - ігровий рушій, розроблюваний і підтримуваний компанією Epic Games. Пристосований у першу чергу для шутерів від першої особи, рушій використовувався й при створенні ігор інших жанрів.Використовують в основному для 3D ігор на комп'ютери і приставки. Але з ростом технологій часто стали зустрічатися гри на цьому движку і на мобільних. Для 2D-ігор є спеціальна система 2D Paper. Написаний мовою C++, рушій дозволяє створювати ігри для більшості операційних систем і платформ. Для спрощення портування рушій використовує модульну систему залежних компонентів: підтримує різні системи рендерингу (Direct3D, OpenGL, Pixomatic), відтворення звуку (EAX, OpenAL, DirectSound3D), засоби голосового відтворення тексту, розпізнавання мовлення, модулі для роботи з мережею й підтримка різних пристроїв вводу. AS у Unreal Engine 4 не має великої кількості готових ресурсів, але всі вони якісні і добре виконані. Для гри у мережі підтримуються технології Windows Live, Xbox Live, і GameSpy, що дає можливість підключити до 64 гравців одночасно. Попри те, що офіційно засоби розробки не містять у собі підтримки великої кількості клієнтів на одному сервері, рушій використовувався для створення MMORPG-ігор. Один з найвідоміших представників жанру, Lineage II, використовує рушій Unreal Engine [10]. Ліцензія та вартість: безкоштовний для використання. Якщо гра приносить більше 3000 доларів за квартал, виробники беруть п’ять відсотків прибутку [9];

в) Godot - відкритий багатоплатформовий 2D та 3D гральний рушій, що розробляється співавторством Godot Engine Community. До публічного релізу у вигляді відкритого програмного засобу рушій використовувався всередині

деяких компаній Латинської Америки [3]. Оточення розробника працює на Windows, Linux, OS X, BSD і Haiku та може експортувати ігрові проєкти на ПК, консолі, мобільні та веб платформи. Мультиплатформний, має великі можливості в 2D і обмежені в 3D. Є гарна документація, завдяки відкритому коду дуже швидко виправляються помилки. Досить велика і активна спільнота. Працює на всіх платформах. Добре організована структура файлів всередині рушія. Підтримка декількох мов: GDScript, C # (поки не повна), C ++. AS Godot може конкурувати за рахунок великої бази безкоштовних ресурсів. Але також має недоліки і має проблеми з продуктивністю якщо на екрані знаходиться велика кількість об'єктів. Ліцензія та вартість: безкоштовний для використання. Якщо гра приносить більше 3000 доларів за квартал, виробники беруть п’ять відсотків прибутку [11];

г) GameMaker: Studio - один з найпопулярніших ігрових рушіїв, що дозволяє розробляти додатки під безліч платформ. Безкоштовна версія (Standard) обмежена компіляцією під Windows . У порівнянні з нею, Professional версія має безліч переваг, включаючи управління ресурсами, компіляцію для macOS, Ubuntu і запуск на Android . Також, в професійній версії можна купувати окремі модулі, що розширюють функціональність програми. Версія Master Collection містить всі поточні модулі і майбутні доповнення версії. AS GameMaker Studio має багато різних ресурсів, як платних, так і безкоштовних, але більшість з них не дуже якісні. Це в основному інді-2D ігри для Windows, смартфони або приставки. Створюються в основному за допомогою візуального програмування Drag & Drop. Є можливість програмувати власною зовсім простою мовою gml, що нагадує javascript. Ком'юніті досить велике і активне. Низький поріг входження, не обов'язкові знання з програмування. Висока швидкість розробки. Недоліки рушія: Тільки 2D гри. Є обмежена підтримка 3D, але не використовується в основному. Застарівша документація. Ліцензія та вартість: Під кожну платформу своя ліцензія. Мінімальна версія для Windows коштує 39 доларів, є безкоштовний 30-денний період [9];

д) Construct 2 - мультиплатформенний конструктор двовимірних ігор, що розробляється компанією Scirra.. На ньому роблять гри і на комп'ютери і на телефони і для інтернету. Ігри створюються без програмування за допомогою вибору вже встановлених варіантів, закладених як методи Box 2D Physics. Розробниками не сприймається всерйоз. Проте ігри створюються швидко і потім монетизуються. Є досить активне ком'юніті. Низький поріг входження, знання програмування не потрібно. Висока швидкість розробки та активне ком'юніті. AS Construct 2 має переважно платні ресурси. Ліцензія та вартість:

1) Personal - для окремих інді-розробників, з можливістю обмеженого комерційного використання (заробивши на грі більше 5000$, користувач зобов'язаний зробити апґрейд ліцензії до Business). Вартість ліцензії складає 129.99 доларів;

2) Business - для професійних ігрових студій з необмеженою можливістю комерційного використання. Коштує 429.99 доларів;

3) Free - безплатна демонстраційна версія рушія з суттєвими обмеженнями (неможливість створити більше 100 подій в проєкті, заборона використання families, відсутність експорту окрім як на HTML5 тощо). Комерційне використання заборонене [9].

Проаналізувавши розглянуті рушії, було виділено основні переваги та недоліки усіх функціональних особливостей даних рушіїв(табл. 2.1).

Таблиця 2.1 – Порівняльна характеристика найпоширеніших рушіїв

Функціональні можливості	Назва рушія				
	Unity	Unreal Engine	Godot	GameMaker Studio 2	Construct 2
1	2	3	4	5	6
Низький поріг входження	+	-	-	+	+
Наявність навчальної літератури	+	+	+	-	+
Безкоштовний	+	+	+	-	+

Продовження таблиці 2.1

1	2	3	4	5	6
Не має суттєвих обмежень	+	-	+	+	-
Наявність AS	+	+	+	+	+

2.2 Висновок за розділом 2

У сучасному світі існує багато інструментів для розробки гри будь-якого жанру. Рушії гри дозволяють швидко і просто створювати ігри під різні платформи. Відповідно до поставленої задачі за обраними критеріями можна обрати найкращий програмний засіб для розробки гри.

3 ВИБІР ІНСТРУМЕНТАРІЮ РОЗРОБКИ ГРИ «SHOOT HIM»

3.1 Вибір рушія гри

Для розробки функціоналу гри було обрано Unity. Гра “Shoot Him” буде виконана у 2D, з чим даний рушій добре впорається. Unity надає можливість створювати ігри під різні платформи, тому з реалізацією гри для ОС Android проблем не виникне. У AS Unity є готовий джойстик, який в подальшому буде використовуватися для пересування та повороту головного героя. Інтерфейс рушія Drag & Drop зручний і простий у розумінні, що зекономить час розробки гри. Unity підходить для початківців і має низький поріг входження. Рушій безкоштовний і не має суттєвих обмежень. У разі виникнення проблем під час розробки, у Unity є багато навчальної літератури та відеоуроків, а також форуми [12].

3.2 Вибір мови для розробки гри

Мовою програмування обрано C#. NET, і всі бібліотеки Unity побудовані з використанням коду на C#. C# - єдина мова програмування для використання всередині рушія. C# - об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (належить Microsoft). Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників - мов C++, Object Pascal, Модула і Smalltalk - C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, мова C#, на відміну від C++, не передбачає множинне успадкування класів.

C# - одна з найкращих мов програмування, в якіїй поєднуються потужність і гнучкість універсальних мов програмування [13].

3.3 Вибір середовища розробки.

Так як Microsoft Visual Studio - безкоштовне розширення від Unity а також є титульним компілятором C#, було обрано дане середовище розробки. Visual Studio - засіб для створення, редагування та зневадження сучасних веб-застосунків і програм для хмарних систем. Visual Studio розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X. Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціонується як легковагове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки [14].

3.4 Вибір середовища розробки графічних об’єктів

Для розробки графічних об’єктів було обрано Aseprite. Aseprite - це програма для створення анімованих спрайтів і піксельної графіки. Спрайт - це маленькі картинки, які можна використовувати в відеогрі. Можна малювати символи за допомогою рухів, заставок, текстур, візерунків, фонів, логотипів, колірних палітр, ізометричних рівнів. Aseprite має зручний інтерфейс, палітру кольорів, вбудовану функцію скріншоту, можливість вставити власні шрифти а також у Aseprite легко зберегти зображення [15].

3.5 Висновок за розділом 3

Інструментами розробки ігор відповідно до поставленої задачі було обрано найкращий рушій гри Unity з мовою програмування C# і середовищем розробки Visual Studio, а також графічний редактор Aseprite.

4 РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ГРИ «SHOOT HIM»

4.1 Загальна структура програми

Гра складається з сьоми сцен:

– початковий екран;
– налаштування;
– головне меню;
– магазин;
– покупка снаряду;
– інвентар;
– бій.

Ієрархія сцен відображена на рисунку 4.1.

Рисунок 4.1 — Ієрархія сцен

Використовуючи бібліотеку SceneManagement, у методі OnClick(), який вказується як подія натискання об’єкту з компонентом Button, здійснюємо перехід на іншу сцену. Для сцен з налаштуванням і покупкою снарядів використовується функця LoadScene.Additive(“Ім’я сцени”), що дозволяє завантажувати сцену поверх запущеної. При запуску цих сцен не має змоги повернутись на попередню сцену. Для інших - LoadScene(“Ім’я сцени”) зі звичайним завантаженням сцени.

Класи програми відображені на рисунку 4.2.

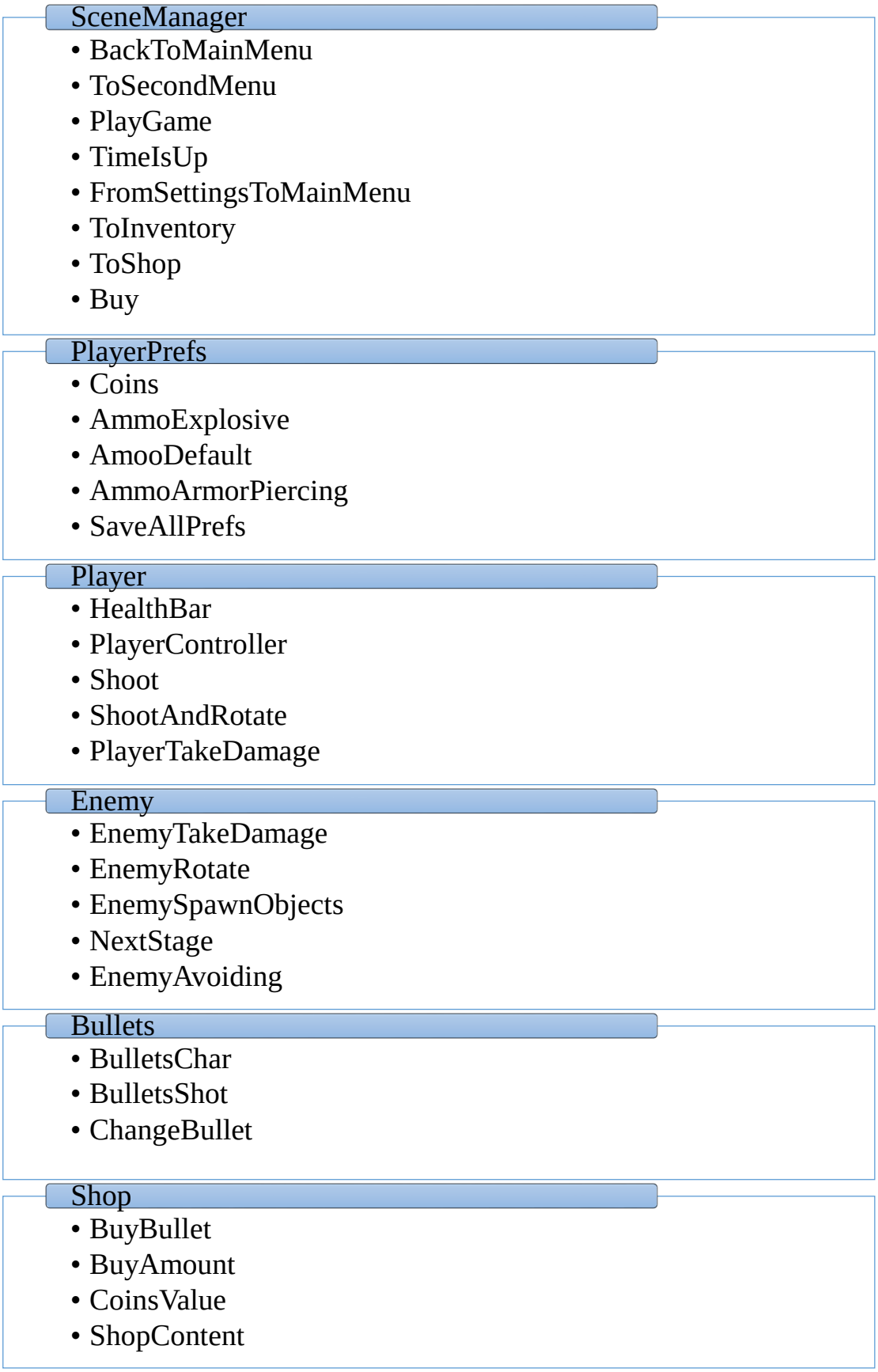


Рисунок 4.2 — Класи програми

Взаємодія об'єктів на сцені бою виконується з використанням коллайдерів. Об'єктам присвоюється компонент Rigidbody2D і коллайдери. Коллайдери на стінах виключають змогу вийти за межі арени. Але деякі об'єкти мають коллайдери з ввімкненою функцією IsTrigger. Снаряди мають коллайдер тріггер і у классі пострілу, при зіткненні з коллайдером об’єкта, виконується перевірка слою об’єкта. Снаряд зникає при зіткненні з об’єктом зі слоєм Solid, а сам об’єкт отримує урон рівний урону влученого снаряду. Також такі коллайдери використовуються для зони близької до стін. У классі DontStayCloseToWall при зіткненні з коллайдером виконується перевірка тегу. Якщо об’єкт з тегом “Enemy” зіткнеться з цим коллайдером, то перший змінить напрям свого руху в протилежний.

Збереження данних виконується з використанням набору методів PlayerPrefs. Для даних таких як монети і кількість снарядів створюється клас і статична змінна. В методах классу здіснюється запис числа в реестр і отримання числа з реестру. Таким чином при виході з гри і наступному вході всі дані зберігаються, а статичні змінні спрощують передачу інформації між сценами.

4.1.1 Початковий екран

Вигляд сцени початкового екрану відображено на рисунку 4.3.

Рисунок 4.3 – Початковий екран

На цій сцені є декілька елементів інтерфейсу:

– налаштування. З’єднавши дві шестерні, здійснюється перехід на сцену зі зміною керування;

– скидання прогрессу. Натиснувши, весь прогресс (монетки та патрони) скинеться;

– вхід до головного меню;

– particle system. Фонова система частинок;

– перемикач музики. Вмикає та вимикає музику в грі;

– кнопка “додати 100 монет” для спрощення тестування гри.

4.1.2 Налаштування

Вигляд сцени налаштування відображено на рисунку 4.4.

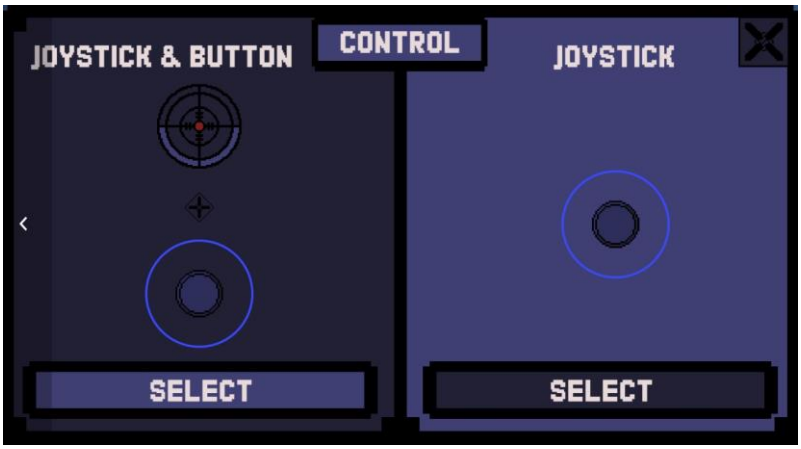


Рисунок 4.4 — Налаштування

На сцені з налаштуваннями три елемента інтерфейсу:

– обрати перший тип керування. Кнопка пострілу і джойстик повороту персонажа;

– обрати другий тип керування. Джойстик водночас відповідає за поворот персонажа та постріл(є можливість наведення поки ручка джойстика не відтянута до краю);

– вийти на початковий екран.

4.1.3 Головне меню

Вигляд сцени головного меню відображено на рисунку 4.5.

Рисунок 4.5 – Головне меню

На цій сцені чотири елемента інтерфейсу:

– повернутися до початкового екрану;
– кнопка магазину - відкриває сцену з магазином;
– інвентар - відкриває сцену з інвентарем;
– перехід до бою.

4.1.4 Магазин

Вигляд сцени магазину відображено на рисунку 4.6.

Рисунок 4.6 – Магазин

Сцена з магазином має п’ять елементів інтерфейсу:

– перехід до головного меню;
– поточна кількість монет;
– кнопка звичайного снаряду;
– кнопка бронебійного снаряду;
– кнопка вибухового снаряду.

4.1.5 Покупка снаряду

Вигляд сцени покупки снаряду відображено на рисунку 4.7.

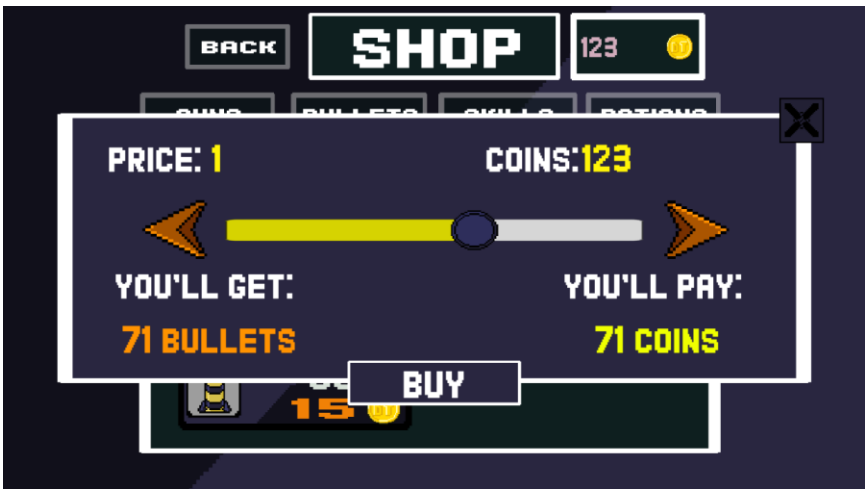


Рисунок 4.7 – Покупка снаряду

На цій сцені 8 елементів інтерфейсу:

– кнопка декременту кількості снарядів що купуються;
– кнопка інкременту кількості снарядів що купуються;
– слайдер вибору кількості снарядів що купуються;
– кнопка підтвердження покупки;
– поточна кількість монет;
– ціна снаряду;
– обрана кількість снарядів;
– кінцева ціна.

4.1.6 Інвентар

Вигляд сцени інвентаря відображено на рисунку 4.8.

Рисунок 4.8 – Інвентар

На цій сцені п’ять елементів інтерфейсу:

– кнопка повернення до головного меню;
– кількість звичайних снарядів;
– кількість бронебійних снарядів;
– кількість вибухових снарядів;
– поточна кількість монет.

4.1.7 Бій

Вигляд сцени бою відображено на рисунку 4.9.

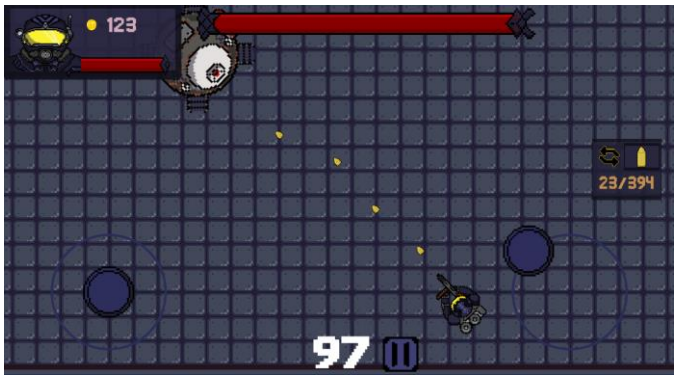


Рисунок 4.9 – Бій

На цій сцені 10 елементів інтерфейсу:

– поточна кількість монет;
– кількість патронів;
– зміна типу снарядів;
– обраний снаряд;
– показник здоров’я головного героя;
– показник здоров’я ворога;
– вихід до головного меню;
– джойстик пересування;
– джойстик повороту і пострілу;
– таймер забігу.

4.2 Графіка і дизайн гри

Для гри у графічному редакторі Aseprite було створено близько 100 спрайтів не враховуючи анімацій. Гра виконана у піксельному стилі. Дизайн гри з ухилом на простоту.

Приклади спрайтів виконаних у Aseprite наведено на рис. 4.10, рис. 4.11 і рис. 4.12.

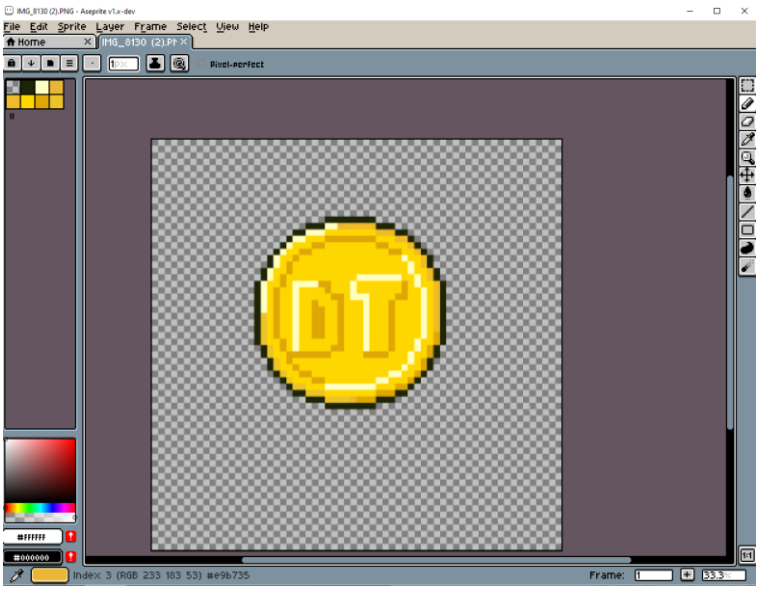


Рисунок 4.10 — Спрайт монети

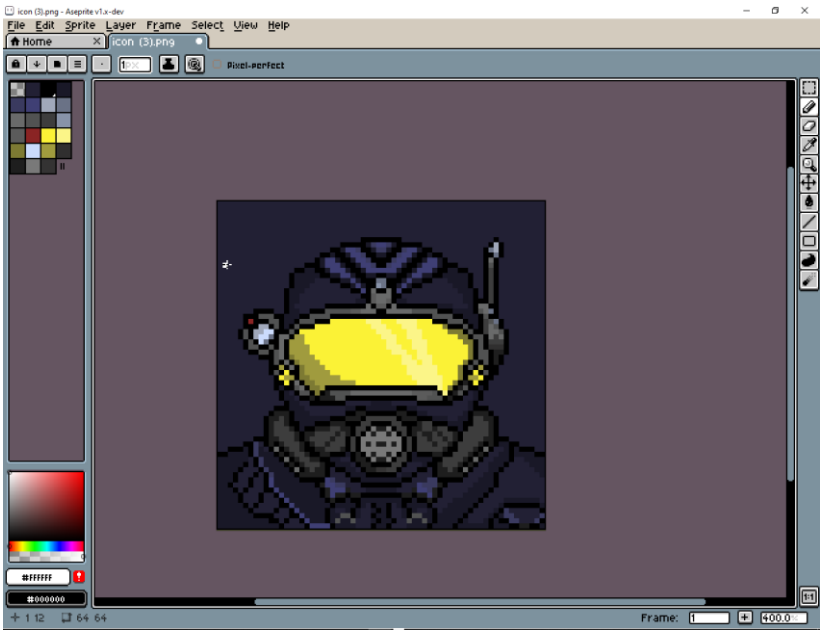


Рисунок 4.11 — Іконка персонажа

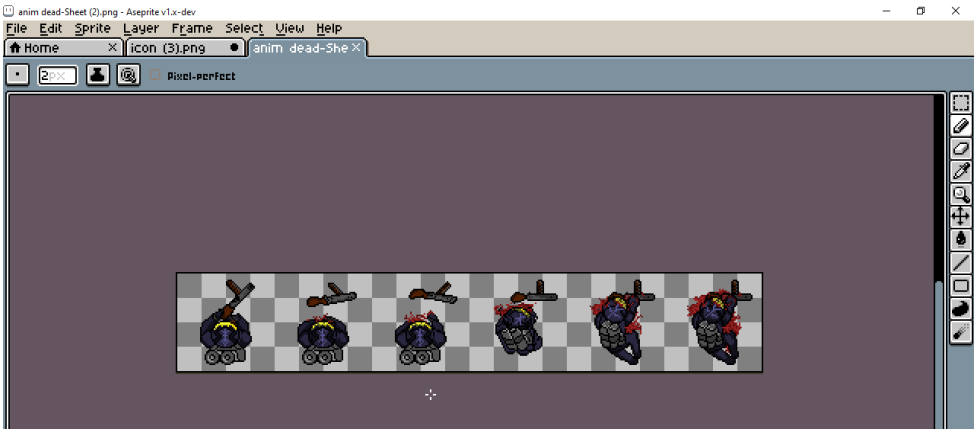


Рисунок 4.12 — Анімація смерті персонажа

4.3 Функціонал гри і геймплей

4.3.1 Перший вхід в гру і основні умови

При першому вході в гру у гравця є тільки 100 звичайних патронів. Переходимо до бою, так як інші сцени поки що не мають сенсу.

Зараз єдиною задачою є отримання монет з ворога. За кожний снаряд, що влучить в ціль отримуємо три монети. Так як HP ворога завжди оновлюється, звичайними патронами знищити його неможливо у зв’язку з мінімальною шкодою HP ворога.

4.3.2 Другий етап гри

Отримавши деяку кількість монет з’являється можливість купити бронебійні та вибухові снаряди, які завдають більшої шкоди, мають менший розкид але і коштують дорожче. На цьому етапі вирисовується деяка економіка і баланс гри. Ціль гри - знищити ворога, але без запасу бронебійних і вибухових снарядів це майже неможливо, при цьому варто мати й звичайні снаряди для фарму (з англ. farming – «ферма», що в контексті означає «здобуття монет»).

4.3.3 Ворог

Алгоритм ворога змушує його постійно змінювати напрям свого руху, використовуючи метод Random, що помітно ускладнює гравцю шанс влучити в ціль. Також ворог має здібність раз в п’ять секунд створювати різні об’єкти такі як:

– звичайна бочка;
– вибухова бочка;
– контейнер.

Дані об’єкти руйнуються після достатньої кількості влучень снарядами.

Ворог завдає шкоди персонажу при зіткненні.

Ворог має 100 HP на першій стадії, а втративши половину HP активує другу стадію, додатково отримуючи 30 HP і прискорюючись.

4.3.4 Персонаж

Персонаж має нижчу швидкість пересування ніж у ворога. Має всього 5 HP, тому слід бути обережним. Перезарядка триває дві секунди, час між вистрілами складає 0,13с, у магазині 30 патронів. Персонаж повністю анімований.

4.3.5 Характеристика снарядів

Так як гра виконана у жанрі аркада ігровою умовністю припускається зміна величини розкиду за типом снаряду, а не зброї. Характеристики снарядів наведені у таблиці 4.1.

Таблиця 4.1 — Характеристика снарядів

Характеристики снаряду	Тип снаряду		
	звичайний	бронебійний	вибуховий
Шкода	1 одиниця	3 одиниці	5 одиниць
Величина розкиду	0°—16° (рисунок 4.11)	0°—12° (рисунок 4.12)	0°—6° (рисунок 4.13)
Швидкість польоту	15	17	12
Ціна	1 монета	5 монет	15 монет

Розкид звичайного снаряду зображено на рис. 4.13, бронебійного – на рис. 4.14, а вибухового – на рис.4.15.

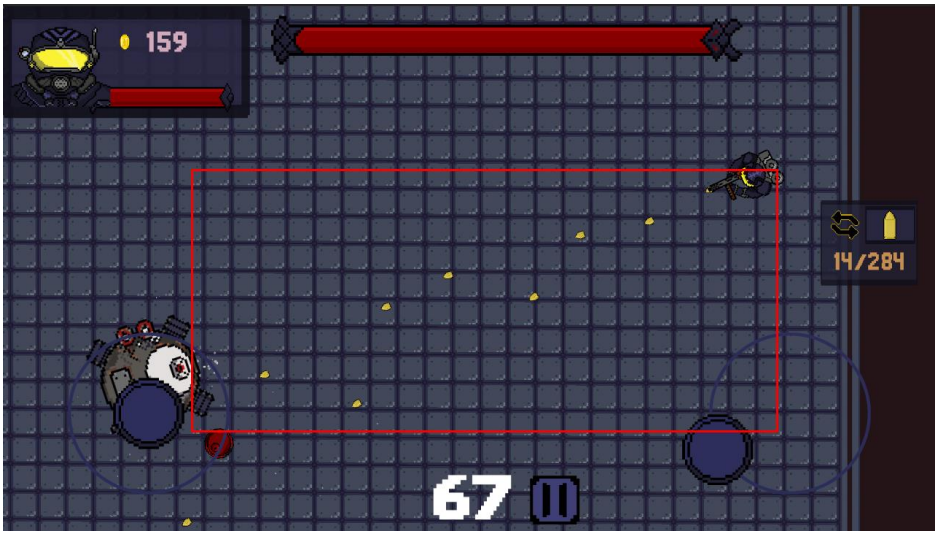


Рисунок 4.13 — Розкид звичайного снаряду

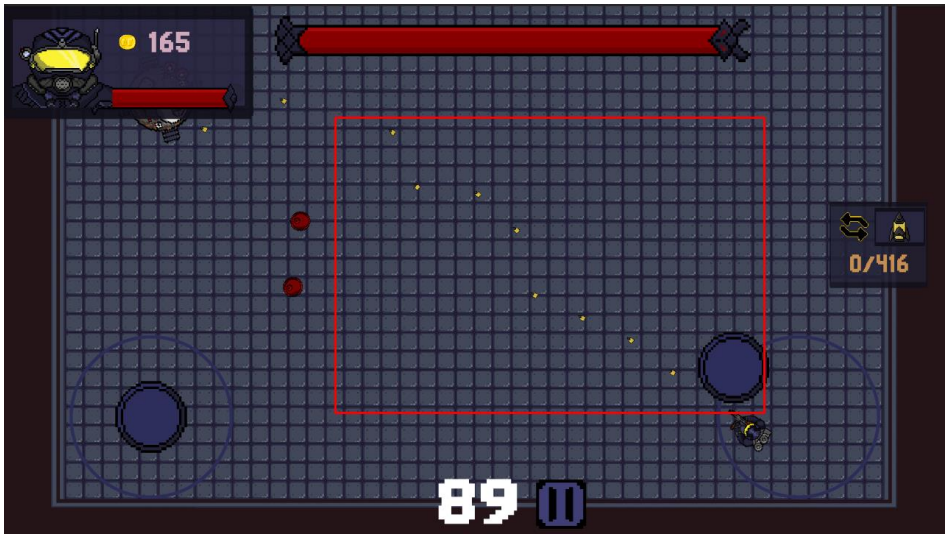


Рисунок 4.14 — Розкид бронебійного снаряду

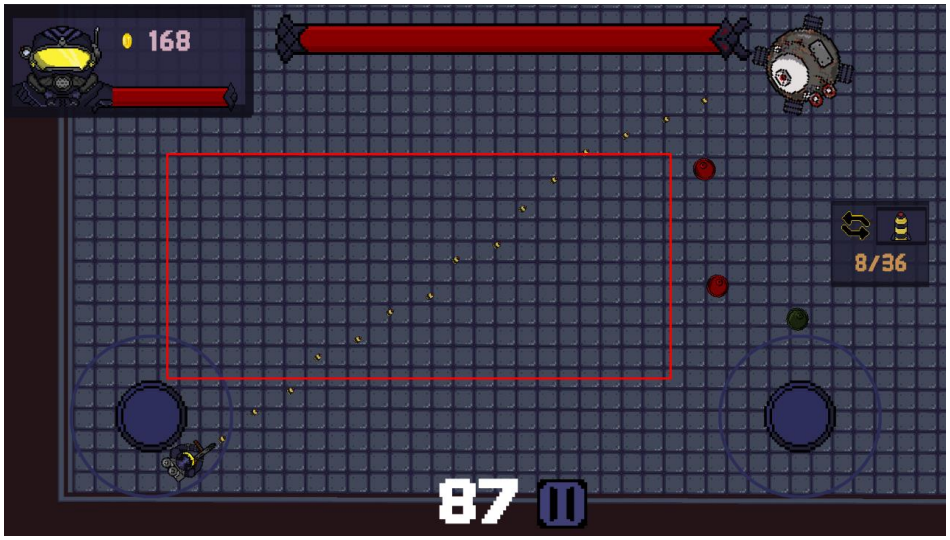


Рисунок 4.15 — Розкид вибухового снаряду

Зміна обраного типу снаряду здійснюється натисканням на спеціальну кнопку (рисунок 4.16).

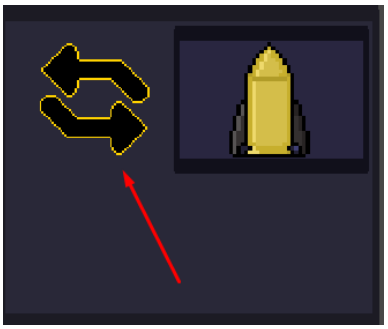


Рисунок 4.16 — Зміна снаряду

5 ТЕСТУВАННЯ ГРИ

Метою роботи було створення гри для мобільних пристроїв. За підсумками тестувань було виявлено як переваги гри, так і недоліки. Для тестування гри було задіяно 4 людини з різними мобільними пристроями під управлінням операційної системи Android.

Список смартфонів на яких тестувалася гра:

– Xiaomi Redmi Note 8 Pro 6/128GB;
– Huawei P Smart Plus 2018 4/64Gb;
– Samsung Galaxy S10 8/128GB;
– Xiaomi Redmi Note 9 Pro 6/128GB.

В ході тестування було виявлено, що на моделі Samsung Galaxy S10 8/128GB з'їзджає інтерфейс, так як гра розроблювалась на стандартному 1920x1080. Проблема була вирішена за допомогою правильного розташування якорів для кожного елементу. Невірне розташування якорів зображено на рис. 5.1 і рис. 5.2, а вірне - на рис. 5.3 і рис. 5.4.

Рисунок 5.1 — Якоря виставлено невірно

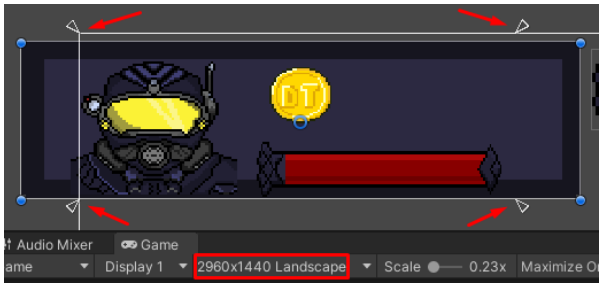


Рисунок 5.2 — Якоря виставлено невірно

Рисунок 5.3 — Якоря виставлено вірно

Рисунок 5.4 — Якоря виставлено вірно

Основним недоліком на всіх пристроях залищається затримка у секунду перед завантаженням сцени. Слід дослідити способи вирішення даної проблеми, знайти можливі методи для підвантаження сцен заздалегіть, або зробити завантажувальний екран між сценами.

Гра добре справляється зі збереженням інформації між сценами, та при повному виході з гри. Гра зберігає такі дані:

– кількість патронів;

– кількість монет;

– тип керування;

– перший запуск (якщо гравець запускає гру вперше, то деяким змінним присвоюються стартові значення, а в іншому випадку гра не скидуватиме ресурси до стартових значень при кожному запуску);

– музика (увімкнено/вимкнено).

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було розроблено гру «Shoot Him». При розробці мобільної гри було реалізовано:

– збереження даних;

– покупка снарядів у магазині;

– систему зміни снарядів з різними характеристиками.

Гра протестована і є унікальною, тому що реалізація основного функціоналу виконана без використання готових рішеньв. В ході виконання роботи було успішно виконано наступні завдання:

– створено гру виконану в жанрі аркадного екшн-шутера для ОС Android мовою програмування C#;

– створено унікальний дизайн гри, що відповідає обраному жанру.

Розроблена гра повністю відповідає вимогам технічного завдання.

Гру можна вважати цілком готовою до випуску.

На думку автора гра має великий потенціал і широкий спектр можливостей для розвитку ідеї.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Digital Combat Simulator [Электрон. ресурс]. – Режим доступа: https://www.aviaport.ru/digest.

2. Індустрія відеоігр [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Індустрія_відеоігр.

3. Грати в ігри це нормально [Електрон. ресурс]. – Режим доступу: https://hromadske.ua/posts.

4. Статистика дохода платформ [Электрон. ресурс]. – Режим доступа: https://dtf.ru/games/Статистика_дохода_разных_платформ.

5. Жанри відеоігор [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Жанри_відеоігор.

6. Симуляційна відеогра [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Симуляційна_відеогра.

7. Стратегічна відеогра [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Стратегічна_відеогра.

8. Рольова відеогра [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Рольова_відеогра.

9. Сравнение игровых движков [Электрон. ресурс]. – Режим доступа: https://sc2tv.ru/blogs/falcon/2019/07/23/sravnenie—igrovykh—dvizhkov.

10. Unreal Engine 4 (рушій гри) [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Unreal_Engine.

11. Godot (рушій гри) [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Godot.

12. Unity (рушій гри) [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Unity.

13. C Sharp [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp.

14. Visual Studio [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Visual_Studio#.

15. Aseprite [Электрон. ресурс]. – Режим доступа: https://8d9.ru/program/aseprite.

ДОДАТОК А

Текст програми

А.1 Компонент пересування персонажа PlayerController.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Collections.Specialized;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class PlayerController : MonoBehaviour
{
   public float speed;
   private float velocityY;
   private float velocityX;
   private Rigidbody2D rb;
   public Joystick joystick;
   public GameObject middle;
   public Animator playerAnimator;
   void Start()
   {
      rb = GetComponent<Rigidbody2D>();
   }


   private void FixedUpdate()
   {
      if(velocityX == 0 && velocityY == 0)
            rb.velocity = new Vector2(0,0);

      velocityX = joystick.Horizontal;
      velocityY = joystick.Vertical;

      if(joystick.Vertical > 0.1f)
      {
            rb.velocity = new Vector2(rb.velocity.x, velocityY * speed);
      }
      else if(joystick.Vertical < -0.1f)
      {
            rb.velocity = new Vector2(rb.velocity.x, velocityY * speed);
      }

      if(joystick.Horizontal > 0.1f)
      {
            rb.velocity = new Vector2(velocityX * speed, rb.velocity.y );
```

```
        }
        else if(joystick.Horizontal < -0.1f)
        {
              rb.velocity = new Vector2(velocityX * speed, rb.velocity.y);
        }

        middle.transform.position = transform.position;


        if (joystick.Horizontal >= 0.1f || joystick.Vertical >= 0.1f || joystick.Horizontal
<= -0.1f || joystick.Vertical <= -0.1f)
            playerAnimator.SetBool("IsRunningAnim", true);
        else
            playerAnimator.SetBool("IsRunningAnim", false);
    }

}
```

А.2 Компонент повороту персонажа і пострілу ShootAndRotate.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Collections.Specialized;
using System.Security.Cryptography.X509Certificates;
using UnityEngine;
using UnityEngine.UI;
using Random=UnityEngine.Random;

public class ShootAndRotate : MonoBehaviour
{
    public GameObject handle;
    public float offset;
    public GameObject target;
    public GameObject[] bullets;
    public Transform shotPoint;
    public Joystick aim;
    private bool isDown;

    [HideInInspector]
    public int totalAmmo;
    [HideInInspector]
    public int magazine = 30;
    [HideInInspector]
```

```
    public int currentAmmo;
    [HideInInspector]
    public float startReload;
    [HideInInspector]
    public float reload = 1.2f;
    private float timeBtwShots;
    public float startTimeBtwShots;

    public GameObject shootSound;
    public GameObject reloadSound;

    public bool isReloading = false;
    public Animator playerAnimator;
    private Rigidbody2D rb;

    public int idBullet = BulletsChar.id;

    public Text ammoDisplay;
    string stringAmmoTotal;
    string stringAmmoCurrent;

    public float recoil;
    public GameObject recoilObj;

    private void Start()
    {
        target.transform.position = new Vector2(target.transform.position.x
,target.transform.position.y + 1);

        BulletsChar.instance.DefaultBulletChar();

        totalAmmo = AmmoDefault.ammoDefault;

        if (totalAmmo < magazine)
        {
            currentAmmo = totalAmmo;
            totalAmmo = 0;
        }
        else
        {
            currentAmmo = magazine;
            totalAmmo -= magazine;
        }
```

```
        startReload = reload;
    }

    public void OnPointerDown()
        {
          isDown = true;
        }

    public void OnPointerUp()
        {
          isDown = false;
        }

    void FixedUpdate()
    {
        if (aim.Horizontal != 0 && aim.Vertical != 0)
            target.transform.position = handle.transform.position -
aim.transform.position;
        else target.transform.position = new Vector2(target.transform.position.x
,target.transform.position.y);

        Vector3 difference = target.transform.position - transform.position;
        float rotZ = Mathf.Atan2(difference.y, difference.x) * Mathf.Rad2Deg;
        transform.rotation = Quaternion.Euler(0f, 0f, rotZ + offset);

        recoil = Random.Range(-BulletsChar.recoilBullet, BulletsChar.recoilBullet);
        recoilObj.transform.rotation = Quaternion.Euler(0f,0f,recoil + rotZ + offset);

        idBullet = BulletsChar.id;

        if (ChangeBullet.isChanged == true)
          switch (idBullet) {

              case 1:
                totalAmmo = AmmoDefault.ammoDefault;
                if (totalAmmo < magazine)
                {
                    currentAmmo = totalAmmo;
                    totalAmmo = 0;
                }
                else
                {
                    currentAmmo = magazine;
                    totalAmmo -= magazine;
```

```
                }
                ChangeBullet.isChanged = false;
                break;
            case 2:
                totalAmmo = AmmoArmorPiercing.ammoArmorPiercing;
                if (totalAmmo < magazine)
                {
                    currentAmmo = totalAmmo;
                    totalAmmo = 0;
                }
                else
                {
                    currentAmmo = magazine;
                    totalAmmo -= magazine;
                }
                ChangeBullet.isChanged = false;
                break;
            case 3:
                totalAmmo = AmmoExplosive.ammoExplosive;
                if (totalAmmo < magazine)
                {
                    currentAmmo = totalAmmo;
                    totalAmmo = 0;
                }
                else
                {
                    currentAmmo = magazine;
                    totalAmmo -= magazine;
                }
                ChangeBullet.isChanged = false;
                break;
        }

        if (timeBtwShots <= 0 && currentAmmo != 0)
        {
            if(isDown == true)
            {
                if (aim.Horizontal >= 0.7f || aim.Vertical >= 0.7f || aim.Horizontal <= -0.7f
|| aim.Vertical <= -0.7f)
                {
                    timeBtwShots = startTimeBtwShots;
                    currentAmmo -= 1;
                    switch (idBullet)
                    {
```

```
                case 1:
                    Instantiate(shootSound, shotPoint.position, transform.rotation);
                    Instantiate(bullets[idBullet-1], shotPoint.position,
recoilObj.transform.rotation);
                    AmmoDefault.ammoDefault = totalAmmo + currentAmmo;
                    AmmoDefault.instance.SetValueOfAmmoDefault();
                    break;
                case 2:
                    Instantiate(shootSound, shotPoint.position, transform.rotation);
                    Instantiate(bullets[idBullet-1], shotPoint.position,
recoilObj.transform.rotation);
                    AmmoArmorPiercing.ammoArmorPiercing = totalAmmo +
currentAmmo;
                    AmmoArmorPiercing.instance.SetValueOfAmmoArmorPiercing();
                    break;
                case 3:
                    Instantiate(shootSound, shotPoint.position, transform.rotation);
                    Instantiate(bullets[idBullet-1], shotPoint.position,
recoilObj.transform.rotation);
                    AmmoExplosive.ammoExplosive = totalAmmo + currentAmmo;
                    AmmoExplosive.instance.SetValueOfAmmoExplosive();
                    break;
            }
        }
    }   }
    else
    {
        timeBtwShots -= Time.deltaTime;
    }

    if (currentAmmo == 0)
    {
        if(totalAmmo != 0)
        {
            if (isReloading == false)
            {
                Reload();
            }
            startReload -= Time.deltaTime;
            if (startReload <= 0)
            {
                playerAnimator.SetBool("isReloadingAnim", false);
                CheckAmmo();
                startReload = reload;
```

```
            }
        }
    }
    else
        isReloading = false;

    stringAmmoTotal = Mathf.Round(totalAmmo).ToString();
    stringAmmoCurrent = Mathf.Round(currentAmmo).ToString();
    ammoDisplay.text = stringAmmoCurrent + "/" + stringAmmoTotal;

}

private void Reload()
{
    if (currentAmmo == 0)
    {
        if (isReloading == false)
        {
            playerAnimator.SetBool("isReloadingAnim", true);
            Instantiate(reloadSound, shotPoint.position, transform.rotation);
            isReloading = true;
        }
        startReload -= Time.deltaTime;

    }

}

private void CheckAmmo()
{
    if(totalAmmo < 30)
    {
        currentAmmo = totalAmmo;
        totalAmmo = 0;
    }
    else
    {
        currentAmmo = magazine;
        totalAmmo -= magazine;
    }
}

private void ReloadAmmo()
{
```

```
        if (startReload <= 0)
        {
            playerAnimator.SetBool("isReloadingAnim", false);
            CheckAmmo();
            startReload = reload;
        }
    }
}
```

А.3 Компонент ворога Enemy.cs

```
using System.Collections;
using System.Collections.Generic;
using System.Security.Cryptography;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class Enemy : MonoBehaviour
{
    public float health;
    public float maxHp = 70;

    public float speed = 6;

    public float timeBtwChangePos;
    public float startTimeBtwChangePos;

    public int randomX;
    public int randomY;

    private Rigidbody2D rb;

    EnemyTouchingWalls etw;

    private float timeToChange = 1f;
    public Image dublicateBar;

    private float fillDublicate;
    public float currentHealth;
    public float diferenceBar;

    void Start()
    {
```

```
    health = maxHp;
    rb = GetComponent<Rigidbody2D>();
    randomX = Random.Range(-1, 2);
    randomY = Random.Range(-1, 2);

    Physics2D.queriesStartInColliders = false;

    etw = GetComponent<EnemyTouchingWalls>();


    fillDublicate = health * 100 / maxHp / 100;
    dublicateBar.fillAmount = fillDublicate;
}

public void FixedUpdate()
{
    if(health <= 0)
    {
        Destroy(gameObject);
        Coins.coins += 500;
        PlayerPrefs.SetInt("Coins", Coins.coins);
        SceneManager.LoadScene("SecondMenu");
    }

    startTimeBtwChangePos = Random.Range(0.1f,0.5f);

    if (timeBtwChangePos <= 0)
    {
        if (etw.touchB == false)
            randomX = Random.Range(-1, 2);
        if (etw.touchT == false)
            randomY = Random.Range(-1, 2);

        timeBtwChangePos = startTimeBtwChangePos;
    }
    else
    {
        timeBtwChangePos -= Time.deltaTime;
    }

        if (randomX == 0 && randomY == 0)
            rb.velocity = new Vector2(rb.velocity.x, rb.velocity.y);
        else if (randomX == 0 && randomY != 0)
            rb.velocity = new Vector2(rb.velocity.x, speed * randomY);
```

```
            else if (randomX != 0 && randomY == 0)
                rb.velocity = new Vector2(speed * randomX, rb.velocity.y);
            else
                rb.velocity = new Vector2(speed * randomX, speed * randomY);


        currentHealth = health * 100 / maxHp / 100;

        diferenceBar = dublicateBar.fillAmount - currentHealth;
        if (dublicateBar.fillAmount == currentHealth)
            diferenceBar = 0;

        if(timeToChange <= 0)
        {
            if(diferenceBar <= 0)
            {
                fillDublicate = health * 100 / maxHp / 100;
                dublicateBar.fillAmount = fillDublicate;
            }
            else if (diferenceBar > 0)
            {
                fillDublicate -= Time.deltaTime * 0.12f * diferenceBar * 10;
                dublicateBar.fillAmount = fillDublicate;
            }
        }
        else
            timeToChange -= Time.deltaTime;
    }

    public void TakeDamage(int damage)
    {
        health -= BulletsChar.damage;
        timeToChange = 1.5f;
    }
}
```

А.4 Компонент Coins.cs

```
public class Coins : MonoBehaviour
{
   public static Coins instance = null;

   public static int coins;

   public void Start()
   {
      if(instance == null)
         instance = this;
      GetValueOfCoins();
   }

   public void SetValueOfCoins()
   {
      PlayerPrefs.SetInt("Coins", coins);
   }

   public void GetValueOfCoins()
   {
     coins = PlayerPrefs.GetInt("Coins");
   }
```

ДОДАТОК Б

Слайди презентації

1

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ

Дипломна кваліфікаційна робота за темою «Розробка аркадної гри для ОС Android»

Виконав: Клименко Данило Андрійович, ст. гр. КНТ-137
Керівник: Зайко Тетяна Анатоліївна, к.т.н. доцент НУЗП

Рисунок Б.1 — Слайд №1

2

Мета роботи, об'єкт та предмет дослідження

Об’єкт дослідження – програмні засоби для розробки гри

Предмет дослідження – методи розробки ігор для мобільних пристроїв.

Мета роботи – розробка гри для мобільних пристроїв .

Рисунок Б.2 — Слайд №2

3

Порівняння рушіїв гри

Функціональні можливості	Назва рушія				
	Unity	Unreal Engine	Godot	GameMaker Studio 2	Construct 2
Низький поріг входження	+	-	-	+	+
Наявність навчальної літератури	+	+	+	-	+
Безкоштовний	+	+	+	-	+
Не має суттєвих обмежень	+	-	+	+	-
Наявність AS	+	+	+	+	+

Рисунок Б.3 — Слайд №3

4

Інструментарій розробки гри

Характеристики	Назва середовища розробки					
	Visual Studio	Visual Studio Code	Code Blocks	Project Rider	Eclip-se	MonoDevelop
Безкоштовність	+	+	+	-	+	+
Висока функціональність	+	-	-	+	+	-
Стабільність	+	-	-	-	-	-
Зручно для Unity	+	-	-	-	-	+

Рисунок Б.4 — Слайд №4

5

Розробка програмної частини гри «Shoot Him»

Загальна структура програми:

Використання бібліотеки SceneManagement для переходу між сценами.

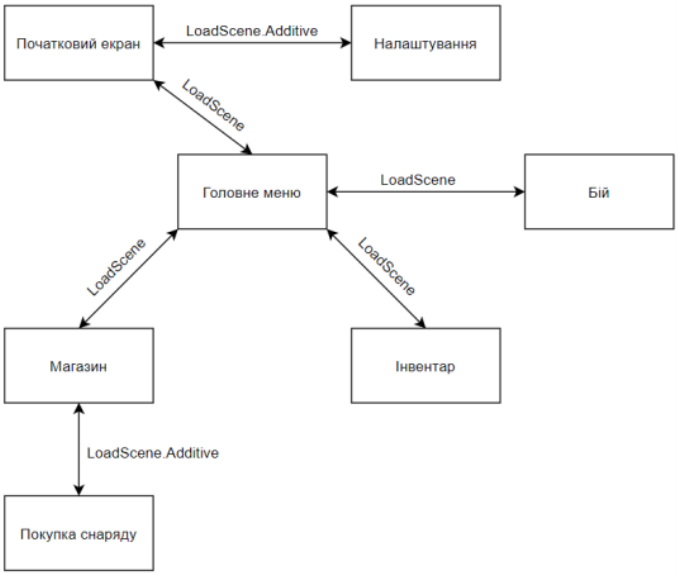


Рисунок 3 — Сцени гри

Рисунок Б.5 — Слайд №5

6

Розробка програмної частини гри «Shoot Him»

Загальна структура програми:

Використання бібліотеки SceneManagement для переходу між сценами.

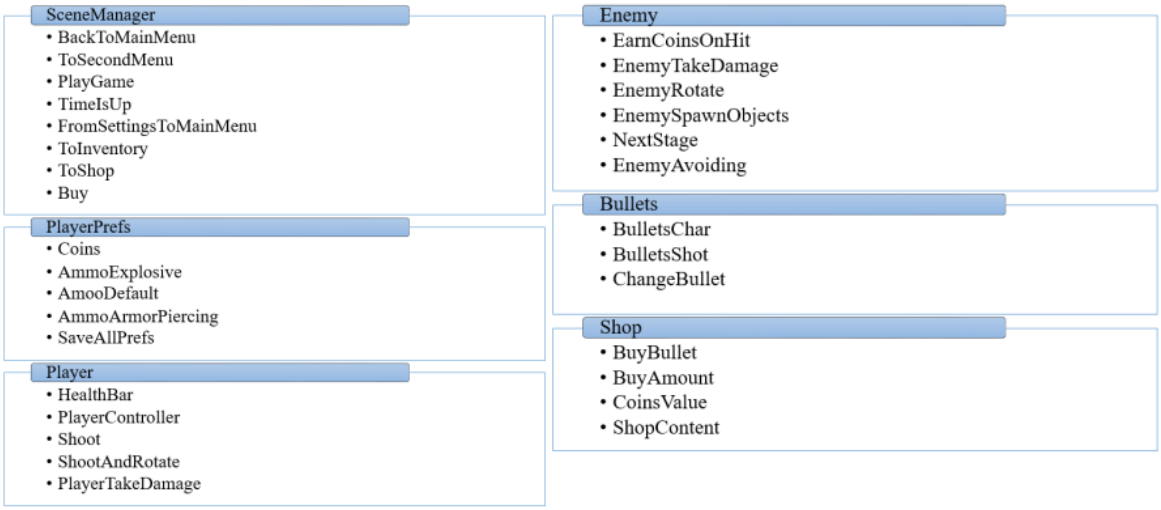


Рисунок 4 — Класи програми

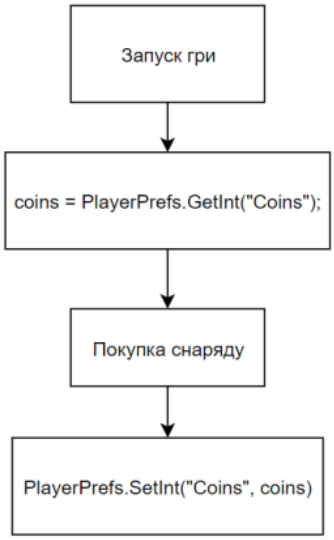
Рисунок Б.6 — Слайд №6

Розробка програмної частини гри «Shoot Him»

7

Збереження даних:

Використання набору методів PlayerPrefs для зберігання даних.



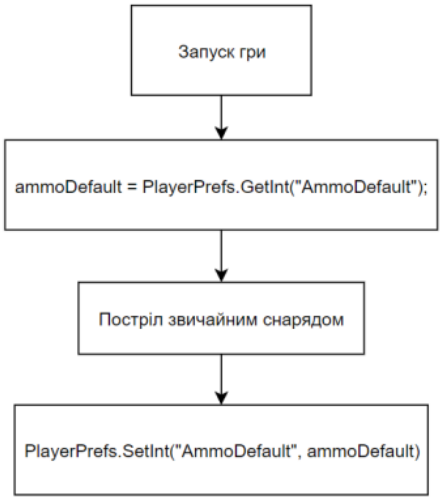


Рисунок 5 — Збереження кількості монет

Рисунок 6 — Збереження кількості снарядів

Рисунок Б.7 — Слайд №7

Розробка програмної частини гри «Shoot Him»

8

Статус і анімація персонажу:

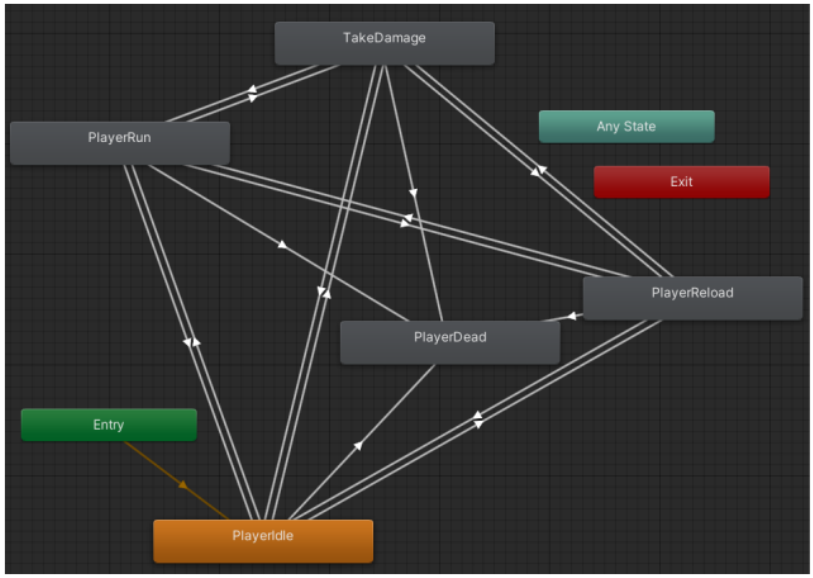


Рисунок 7 — Зв'язок статусів персонажа

Рисунок Б.8 — Слайд №8

Функціонал гри та геймплей

9

Характеристики снарядів:

Характеристики снаряду	Тип снаряду		
	Звичайний	Бронебійний	Вибуховий
Шкода	1 одиниця	3 одиниці	5 одиниць
Величина розкиду	0°—16°	0°—12°	0°—6°
Швидкість польоту	15	17	12
Ціна	1 монета	5 монет	15 монет

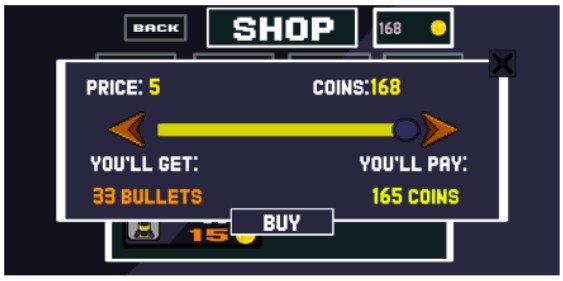


Рисунок 8 — Покупка бронебійних снарядів

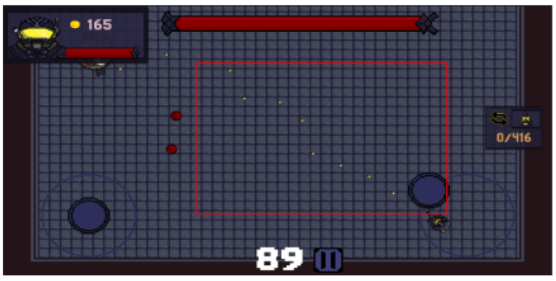


Рисунок 9 — Розкид бронебійних снарядів

Рисунок Б.9 — Слайд №9

Функціонал гри та геймплей

10

Ворог

- Простий штучний інтелект, який змушує його рухатись у випадковому напрямі
- Створює різні об’єкти перед собою
- Має 100 одиниць здоров’я, отримує 30 одиниць додатково на другій стадії
- Швидкість ворога збільшується на другій стадії
- При зіткненні з персонажем завдає шкоди.

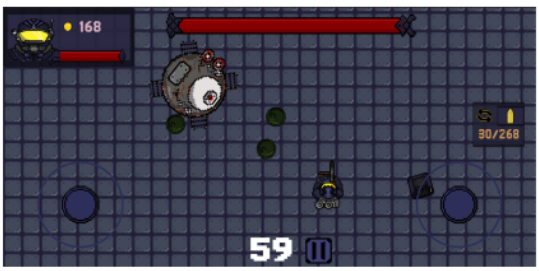


Рисунок 10 — Перша стадія

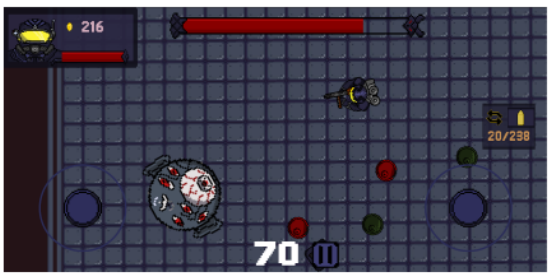


Рисунок 11 — Друга стадія

Рисунок Б.10 — Слайд №10

Функціонал гри та геймплей

11

Персонаж

- Два джойстика для пересування і повороту персонажа і пострілу.
- Нижча швидкість ніж у ворога
- Має 5 одиниць здоров'я
- Перезарядка зброї - 2 секунди
- Скорострільність - кожні 0,13с
- У магазині 30 патронів

Рисунок 12 — Анімація перезарядки

Рисунок Б.11 — Слайд №11

12

Висновок

Під час виконання кваліфікаційної роботи бакалавра було розроблено гру «Shoot Him». При розробці мобільної гри було реалізовано:

— збереження даних;

— систему зміни снарядів з різними характеристиками.

Гра протестована і є унікальною. В ході виконання роботи було успішно виконано наступні завдання:

— створено гру виконану в жанрі аркадного екшн-шутера для ОС Android мовою програмування C#;

— створено унікальний дизайн гри, що відповідає обраному жанру.

Розроблена гра повністю відповідає вимогам технічного завдання.

Гру можна вважати цілком готовою до випуску.

Рисунок Б.12 — Слайд №12

13

Дякую за увагу!

Рисунок Б.13 — Слайд №13