



Випускна кваліфікаційна робота бакалавра

**Моделювання між об'єктної взаємодії обмеженої екосистеми з не
обмеженим ресурсом (на прикладі Vampire Survivors)**

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проекту

№	П.І.Б.	Група
1	Гичак Данило Миколайович	КН-П-212
2	Власіхін Іван Сергійович	КН-П-212

Автор звіту

Гичак Данило Миколайович

Керівник

Доцент кафедри ІТ та ФП, к.т.н., Масрі М.М

АНОТАЦІЯ

Дипломна робота присвячена створенню гри у жанрі action-survivor з елементами RPG, натхненої механіками проєкту *Vampire Survivors*, але з унікальною архітектурою та реалізацією, орієнтованою на гнучкість, масштабованість та легкість розширення. Основною метою роботи стало проєктування програмної структури, яка дозволяє швидко та безперешкодно додавати нові ігрові механіки, персонажів, вміння, ворогів та інші елементи контенту без необхідності змінювати існуючий код.

Всі основні елементи гри реалізовано вручну: логіка персонажа, бойова система, прокачування, механіка ворогів, хвилі появи, звуковий супровід та інтерактивний інтерфейс. Особливу увагу приділено архітектурним рішенням — було використано об'єктно-орієнтоване програмування та шаблони проєктування, такі як Singleton, Strategy, Component та частково Observer, що забезпечило структурованість і модульність коду. Це дало змогу забезпечити просту інтеграцію нового контенту навіть розробникам, які вперше ознайомлюються з проєктом.

Графічна складова виконана на основі безкоштовних відкритих ресурсів та нейромережевих генераторів. У зв'язку з відсутністю дизайнера у команді, було прийнято рішення використовувати готові спрайти та елементи інтерфейсу з відкритих джерел.

Проєкт реалізовано з використанням рушія Unity та мови програмування C#. Вся логіка, включно з анімаціями, рухом камери та освітленням, написана вручну без використання стандартних шаблонів або візуальних тулів Unity. Звукові ефекти керуються через окремий аудіоконтролер, що дозволяє редагувати звуки без зміни коду.

Командна робота організовувалася через онлайн-платформу Discord, а керування завданнями — за допомогою канбан-дошки. Такий підхід забезпечив чіткий розподіл ролей, прозорість у

виконанні завдань та ефективну внутрішню комунікацію.

Результатом роботи став функціональний, гнучкий і зручний для розширення ігровий продукт, який може слугувати як основа для подальшої розробки повноцінного комерційного проекту або освітнього інструмента з програмування ігрових механік.

ЗМІСТ

АНОТАЦІЯ	2
ЗМІСТ	4
ВСТУП	5
ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ	6
Загальний опис	6
Призначення	7
Вимоги до функціоналу	7
Головне меню	7
Системи взаємодії	8
Архітектура	8
Основні компоненти	8
Технології	10
Етапи розробки	10
Вимоги до якості	11
РОЗДІЛ 1	12
РОЗДІЛ 2	15
РОЗДІЛ 3	18
ВИСНОВКИ	23
ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ НАВЕДЕНОЇ ЧАСТИНИ ПРОЄКТУ	25

ВСТУП

У ХХІ столітті відеоігри зайняли чільне місце серед провідних галузей цифрових технологій, перетворившись не лише на джерело розваг, але й на

важливий напрямок для вивчення, творчості та професійної реалізації. Сучасні

ігри вражають своєю складністю, глибиною і технологічним рівнем реалізації

— від графіки до сценаріїв, від штучного інтелекту до мережевої взаємодії між гравцями. У зв'язку з цим питання побудови надійної, гнучкої та масштабованої архітектури програмного забезпечення набуває особливої актуальності.

Розробка навіть невеликого ігрового проєкту вимагає від команди не тільки знання технічних засобів, а й уміння проєктувати систему таким чином,

щоб її можна було безболісно розширювати: додавати нових ворогів, нові здібності персонажів, інтерфейсні елементи, механіки та візуальні ефекти.

Одним із ключових викликів у таких проєктах є уникнення жорсткої залежності між модулями та дотримання принципів модульності та повторного використання коду.

Дипломна робота зосереджена на розробці гри у жанрі action-survivor, натхненої Vampire Survivors, з акцентом на архітектурні рішення та чистоту

коду. У процесі створення застосовано патерни об'єктно-орієнтованого програмування, ручну реалізацію усіх ключових систем (рух, атака, система

ворогів, прокачування тощо) та активне використання можливостей рушія Unity. Основним завданням було зробити систему відкритою для подальшого

розширення та модифікацій навіть без глибокого занурення в архітектуру.

У роботі описано ключові етапи створення проєкту: проєктування системи керування гравцем і ворогами, реалізацію бойових механік, організацію інтерфейсу користувача, підключення аудіо та оптимізацію процесу командної роботи. Також розглянуто труднощі, з якими зіткнулась команда, та шляхи їх подолання.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

Проектування модульної ігрової архітектури з динамічним розширенням функціональності (на прикладі гри Vampire Survival)

Загальний опис

Завданням програмного забезпечення (ПЗ) є створення інтерактивної однокористувацької аркадної гри у жанрі виживання з елементами RPG-прокачки, побудованої за принципом поступового ускладнення ігрового процесу. Гравець керує персонажем, який автоматично атакує хвилі ворогів, накопичує досвід, вибирає покращення, намагаючись вижити якомога довше.

Гра створюється на рушії Unity з використанням C# і використовує хмарну платформу PlayFab для збереження прогресу гравця, авторизації, ведення таблиці рекордів та зберігання валюти (монет), заробленої під час забігів.

Основною функцією системи є моделювання динамічної людино-машинної взаємодії в умовах високої інтенсивності, де ігрове середовище реагує на дії гравця через спаун ворогів, підвищення складності, надання винагород та розвиток персонажа. Особливу увагу приділено плавності керування, інтуїтивності інтерфейсу та збереженню мотивації гравця через прогресію та досягнення.

Гра включає кілька взаємопов'язаних підсистем, які формують єдиний цілісний геймплей:

Система бою – автоматична атака ворогів на основі певного інтервалу часу, який можна покращувати. Вороги з'являються хвилями, мають різну швидкість, здоров'я і типи поведінки. Перемога над ворогами приносить досвід та ресурси. Система прогресії персонажа – гравець підвищує рівень, обираючи покращення (наприклад, збільшення дальності атаки, додаткову шкоду, нові типи зброї, пасивні ефекти тощо). Це стимулює формування індивідуальної тактики кожного забігу.

Система монетизації через монети – за кожний завершений забіг (незалежно від виграшу чи програшу) гравець отримує певну кількість монет. Монети слугують внутрішньоігровою валютою та зберігаються у профілі PlayFab.

Система рекордів та змагання – у профілі PlayFab зберігаються дані про найкращі результати гравця. Також реалізована таблиця лідерів, що стимулює конкуренцію між гравцями.

Система вибору контенту – гра пропонує 2 ігрові локації (із власним візуальним оформленням) та 2 персонажі на вибір. Кожен персонаж має унікальні базові характеристики (наприклад, більшу швидкість або більше здоров'я).

Інтерфейс користувача – реалізовано головне меню, інтерфейс бою, меню вибору покращень, екран результатів. Усі компоненти підтримують інтерактивність, зворотній зв'язок та адаптивність.

Система авторизації через PlayFab – користувач має можливість створити акаунт або увійти через вже існуючий. Дані зберігаються у хмарі та дозволяють гравцеві зберігати прогрес, навіть якщо він змінює пристрій.

Таким чином, ПЗ виконує роль інтерактивної системи, яка керує як ігровим процесом у режимі реального часу, так і довготривалим прогресом гравця. Воно формує мотиваційний цикл, де гравець щоразу намагається досягти кращого результату, отримати більше монет, відкрити нові можливості та поліпшити свої попередні досягнення, зберігаючи при цьому індивідуальний стиль гри.

Гнучкість системи дозволяє розширювати її у майбутньому: додавати нові персонажі, карти, типи ворогів та внутрішньоігрові цілі.

Призначення

Гра призначена для платформи ПК (Windows). Вона орієнтована на фанатів аркад, rogue-lite, auto-attack-ігор, vampires-like, а також на досвід користувача з елементами RPG-прокачки та колекціонування досягнень.

Вимоги до функціоналу

Головне меню

- Запуск гри
- Вибір персонажа (2 варіанти)
- Вибір локації (2 доступні мапи)
- Авторизація через PlayFab (реєстрація/вхід)
- Перегляд таблиці рекордів
- Налаштування (звук)

Системи взаємодії

- **Головний ігровий процес**
Гравець керує персонажем у режимі top-down. Персонаж автоматично атакує ворогів. Гравець ухиляється, збирає досвід, монети, та виживає максимально довго.
- **Система хвиль**
Вороги з'являються групами з поступовим збільшенням складності. Наприкінці — поява босів.
- **Система досвіду та прокачки**
За кожного вбитого ворога гравець отримує досвід. При досягненні нового рівня — вибір одного з трьох покращень (швидкість, атака, дальність, броня, нова зброя тощо).
- **Система монет**
Після завершення забігу гравець отримує монети. Їх кількість залежить від тривалості гри, кількості вбитих ворогів та досягнутого рівня. Монети зберігаються в базі даних PlayFab та використовуються для подальших покупок (опційно).
- **Система рекордів**
Максимальний час виживання та кількість вбитих ворогів зберігаються у профілі PlayFab. Реалізується таблиця лідерів (топ-10).

Архітектура

Основні компоненти

- **Ядро**
 - Головний ігровий цикл
 - Менеджер сцен та станів
 - Завантаження/збереження через PlayFab
- **Гравець**

- Вибір персонажа
- Автоматична атака
- Прокачка та збір ресурсів
- Життя, шкода, апгрейди
- Середовище
 - Генерація карт (2 варіанти: "ліс" і "пустка")
 - Оточення: декоративні об'єкти та перешкоди
- Вороги
 - Різні типи: звичайні, елітні, боси
 - ШІ-поведінка: прямий рух до гравця
 - Система зростаючої складності
- PlayFab
 - Авторизація
 - Збереження монет, рекордів, прогресу
 - Таблиця лідерів
- Інтерфейс користувача (UI)
 - Головне меню
 - Панель гравця (здоров'я, досвід, таймер, монети)
 - Меню покращень
 - Екран завершення гри (результати, отримані монети)

Технології

- Рушій: Unity
- Мова програмування: C#
- База даних: PlayFab SDK
- Графіка: піксель-арт, спрайти
- Звук: безкоштовні звукові ефекти (атаки, смерть, апгрейд)
- Публікація: ПК

Етапи розробки

Етап	Тривалість	Опис
1. Архітектура та каркас	1–2 тижні	Менеджери сцен, UI, PlayFab
2. Ігрова логіка	2–3 тижні	Управління, вороги, атаки
3. Система прокачки	1–2 тижні	Апгрейди, досвід
4. Монети та PlayFab	1–2 тижні	Авторизація, збереження
5. Кінцева збірка	1 тиждень	Баланс, оптимізація, тестування

Вимоги до якості

- FPS не нижче 30 на середніх ПК
- Відсутність критичних помилок
- Правильна інтеграція з PlayFab
- Повноцінна ігрова петля "виживання–нагорода–повтор"

РОЗДІЛ 1

ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Після детального аналізу вимог технічного завдання команда розробників прийняла рішення використовувати ігровий рушій Unity, оскільки він вирізняється високою продуктивністю, зручністю у використанні, широкими можливостями для розробки та підтримкою мови програмування C#.

Основа програмної архітектури складається з наступних компонентів:

- **Game Manager:** Основні механізми гри, включаючи таймер гри, керування станами гри, завантаження ігрових сцен, виклик різних вікон тощо.
- **Player:** Всі аспекти, що стосуються гравця, включаючи управління аватаром, розвиток, візуальна частина тощо.
- **World Manager:** Система, що керує та містить декілька підсистем таких як: ігрова камера, створенням ворогів на карті, та розташуванням анімаційних об'єктів в світі.
- **Enemy:** Логіка та поведінка всіх неігрових персонажів, ворогів.
- **Audio:** Всі елементи звукового супроводу, включаючи музику, звукові ефекти інтерфейсу та гравця тощо.
- **UI:** Елементи інтерфейсу користувача, включаючи головне меню, пауза , вибір мапи тощо.
- **Design:** Частина Design охоплює створення та налаштування інтерфейсних елементів для головного меню, розвитку головного героя, а також внутрішньоігрових меню — меню паузи, меню гравця та панелі показників.

Оскільки команда не мала у своєму складі графічного дизайнера, для реалізації візуальної складової проєкту були використані графічні ресурси з відкритим доступом. Це стосується як інтерфейсних елементів, так і зображень (спрайтів) для аватарів гравців, неігрових персонажів, об'єктів ігрового середовища, а також базових елементів брендингу (логотип, шрифт тощо).

Особлива увага приділялася зручності та інтуїтивності інтерфейсу,

щоб забезпечити комфортну взаємодію користувача з грою на всіх етапах геймплею.

Програмну частину проєкту було повністю реалізовано з використанням мови програмування C# та Unity Scripting API. Основна функціональність розроблялася власноруч, без застосування сторонніх шаблонів або готових бібліотек, таких як Asset Store-пакети для управління камерою, освітленням або геймплейними елементами. Усі ключові компоненти, включаючи логіку руху камери, обробку освітлення, управління персонажами та внутрішні ігрові системи, були написані вручну, що дозволило досягти високого рівня контролю над архітектурою проєкту.

Разом з тим, у роботі використовувались базові компоненти Unity, зокрема стандартні об'єкти сцени, системи колізій, UI-система та інструменти роботи з анімацією. Для організації та збереження даних активно застосовувалися Scriptable Objects та prefabs, які значно спростили процес додавання нового контенту.

Під час проєктування програмної архітектури активно застосовувалися об'єктно-орієнтовані принципи та патерни проєктування, зокрема Singleton, Strategy, Component та частково Observer. Такі рішення дозволили створити гнучку, масштабовану й легко підтримувану систему.

Ключові ігрові компоненти, як-от керування гравцем або менеджер гри, реалізовані через патерн Singleton, що забезпечує глобальний доступ та гарантує єдиний екземпляр класу. Система скілів побудована за принципом патерну Strategy: кожен скілл є окремим класом зі своєю поведінкою, що дозволяє легко додавати нові типи атак (наприклад, ближній бій, ударна атака чи зона дії) без зміни існуючого коду.

Архітектура також використовує компонентний підхід, властивий Unity, що дозволяє чітко розділити відповідальність між логікою об'єкта та його візуальним або фізичним представленням. Частково реалізовано патерн Observer для взаємодії між системами, наприклад, оновлення інтерфейсу користувача при зміні здоров'я або досвіду гравця.

Завдяки такому підходу будь-який розробник, навіть без попереднього ознайомлення з усім кодом, зможе швидко додати нову

пасивну здатність, ворога або інший ігровий елемент. Проєкт спроектовано з акцентом на розширюваність, що робить його зручним для подальшої розробки й масштабування.

Командна робота була організована з використанням методології Kanban, що забезпечувало прозорість та ефективність управління завданнями. Для візуалізації робочого процесу застосовувалась Kanban-дошка, яка дозволяла відстежувати статус виконання задач у реальному часі та координувати діяльність команди.

Для оперативної комунікації та обміну інформацією команда використовувала платформу Discord, що сприяло швидкому вирішенню питань і підтримці злагодженості в роботі.

Вихідний код розробленого проєкту розміщено у публічному репозиторії на платформі GitHub, що забезпечує зручне відстеження змін, колаборацію з іншими розробниками та контроль версій. Готова користувацька версія гри доступна на веб-сервісі itch.io.

РОЗДІЛ 2

Система авторизації користувача

Система авторизації у грі відіграє ключову роль у збереженні індивідуального прогресу кожного гравця, а також забезпечує персоналізацію ігрового досвіду. Вона дає змогу:

- Створити унікальний профіль гравця, до якого прив'язується персональна статистика;
- Зберігати та відновлювати важливі дані (кількість монет, обраного персонажа, найкращі результати тощо) у хмарному сховищі;
- Синхронізувати прогрес між різними пристроями (наприклад, ПК і WebGL-версією гри);
- Вести загальну таблицю лідерів з використанням PlayFab статистики;
- Забезпечити цілісність даних і захист від втрати при видаленні локальних збережень;
- Розширити можливості майбутньої соціальної взаємодії між гравцями (досягнення, PvP, подарунки тощо).

Система побудована на основі **PlayFab Client API**, що дозволяє використовувати різні способи автентифікації: за унікальним ID пристрою, за email та паролем, а також за допомогою гостьового входу.

Структура реалізації

Клас **PlayFabManager**

Цей клас реалізує базову логіку:

- **Реєстрація нового користувача:** через RegisterPlayFabUserRequest, з валідацією мінімальної довжини пароля.
- **Авторизація:** через LoginWithPlayFabRequest, збереження токена сесії (SessionTicket) та унікального ідентифікатора гравця (CustomId) у PlayerPrefs.
- **Прив'язка CustomId до облікового запису:** дозволяє надалі входити автоматично без логіна/пароля.
- **Виведення повідомлень про статус або помилки:** через TextMeshProUGUI.

Клас AutoLogin

Забезпечує **автоматичний вхід**:

- При запуску сцени перевіряє, чи збережено PlayFabPlayerId і PlayFabSessionTicket.
- Якщо так — виконується вхід через LoginWithCustomID.
- Успішний вхід: приховує панель логіна, показує панель користувача з нікнеймом (через GetAccountInfoRequest).
- Якщо дані недійсні або їх немає — показує форму входу.
- Реалізовано функцію **Logout()**, яка видаляє локальні збережені дані й повертає гру до стану неавторизованого користувача.

Взаємодія з інтерфейсом

- Гравець взаємодіє з полями вводу логіна/пароля (TMP_InputField), кнопками входу/реєстрації, повідомленнями про статус (TextMeshProUGUI).

- Стан UI перемикається відповідно до статусу авторизації:
 - loginPanel — для неавторизованого користувача;
 - userPanel — для авторизованого, з відображенням нікнейму.

Взаємодія з ігровими системами

Система авторизації є основою для наступних підсистем:

- **Система монет** — оновлення та збереження валюти можливе тільки при активній авторизації;
- **Таблиця лідерів** — пов'язана з PlayFab ID гравця;
- **Вибір персонажа** — ідентифікується для кожного користувача індивідуально;
- **PlayerController** — після входу до гри застосовується налаштований прогрес (через зв'язок із GameManager / DataManager).

Переваги та розширюваність

- **Масштабованість** — легко інтегрується підтримка email/пароля, Facebook, Google, Steam тощо.
- **Безпечність** — збереження даних у PlayFab, не втрачаються при очищенні пам'яті або переустановці гри.
- **Гнучкість** — окрема реалізація Logout дозволяє змінювати користувача без перезапуску.

РОЗДІЛ 3

Система таблиці лідерів

Таблиця лідерів реалізована як рейтинг гравців за сумарною кількістю **нанесеної шкоди** (TotalDamage) за всі ігрові сесії. Вона стимулює змагальність, повторну гру та розвиток персонального рекорду. Гравець може побачити свою позицію серед інших користувачів у глобальному рейтингу, який зберігається у хмарному сервісі PlayFab.

Мета:

- Мотивація через конкуренцію;
- Глобальне збереження результатів;
- Прив'язка до акаунта гравця;
- Підвищення залученості через соціальні елементи.

Реалізація

- **Надсилення результату:**
 - Метод SendLeaderboard(int score) у PlayFabManager відправляє значення score у PlayFab через UpdatePlayerStatisticsRequest.
 - Значення записується в статистику під назвою "TotalDamage".
- **Отримання топу:**
 - Метод GetLeaderboard() викликає GetLeaderboardRequest на "TotalDamage" і отримує топ-8 гравців.
 - Результати виводяться через UI: використовується rowPrefab з трьома полями (позиція, ім'я, значення).

- Ім'я береться з DisplayName, встановленого при реєстрації через UpdateUserTitleDisplayNameRequest.
- **Очищення старих записів:**
 - Метод ClearLeaderboard() очищує UI перед оновленням.

Поведінка

1. Після забігу викликається SendLeaderboard(score) із загальною кількістю завданої шкоди.
2. PlayFab оновлює статистику "TotalDamage" для поточного користувача.
3. При відкритті таблиці — GetLeaderboard() запитує топ-8 записів і відображає їх у таблиці.
4. Для кожного запису показується місце, ім'я гравця та його результат.

РОЗДІЛ 4

Система монет та збереження прогресу

Система монет у проєкті виконує роль внутрішньоігрової валюти, яка надається гравцеві за досягнення у грі (наприклад, тривалість забігу, кількість вбитих ворогів). Вона є ключовим елементом збереження прогресу гравця та може бути використана для розблокування нового контенту, що підвищує глибину геймплею та мотивацію до повторного проходження.

Особливість системи полягає у **подвійному збереженні**:

- Локально — у PlayerPrefs для швидкого доступу та роботи офлайн;
- У хмарному профілі PlayFab — для збереження між пристроями, обліку та захисту від втрати.

Система забезпечує **двосторонню синхронізацію**:

- Після авторизації гра порівнює локальні монети з тими, що зберігаються у PlayFab;
- У разі розбіжності — оновлюється PlayFab або локальні дані, залежно від більшого значення.

Реалізація

Компоненти:

- **CoinManager.cs**
 - Singleton, зберігає поточну кількість монет (coins) і заробіток за останній забіг (gainedCoins);
 - метод SaveCoins() додає зароблені монети до балансу, зберігає їх у PlayerPrefs;

- якщо користувач авторизований (isLoggedIn), викликає PlayFabManager.AddVirtualCurrency().
- **PlayFabManager.cs**
 - метод AddVirtualCurrency(int coins) додає монети до облікового запису користувача в PlayFab (валюта "CN").
- **AutoLogin.cs**
 - при успішному автологіні викликає GetUserInventory:
 - якщо монети в PlayFab більші — замінює локальні;
 - якщо локальні більші — надсилає різницю в PlayFab;
 - оновлює змінну CoinManager.instance.isLoggedIn = true.

Поведінка

1. При старті гри клас AutoLogin намагається авторизувати гравця.
2. Якщо вдалося:
 - Отримує DisplayName і відображає його в інтерфейсі.
 - Завантажує кількість монет з PlayFab і порівнює з локальним значенням.
 - Синхронізує в той чи інший бік, зберігаючи максимальне значення.
3. Після завершення кожного забігу:
 - Викликається CoinManager.SaveCoins():

- монети додаються до балансу;
- локально зберігаються у PlayerPrefs;
- надсилаються до PlayFab, якщо вхід активний.

ВИСНОВКИ

У процесі виконання дипломної роботи було здійснено повноцінну розробку функціонального прототипу гри у жанрі *action-survivor* з елементами RPG та автоматизованої прокачки персонажа. Основою слугувала аналітична модель популярної гри *Vampire Survivors*, але розроблений проєкт реалізовано з урахуванням власної архітектури, сучасних підходів до структурування ігрових систем та хмарного зберігання даних.

Основні досягнення дипломної роботи включають:

1. Реалізація ігрового середовища та базової архітектури гри

Розроблено 2D-ігрове середовище з підтримкою руху гравця, автоспрямованої атаки ворогів, накопичення досвіду та вибору покращень. Архітектура побудована на принципах об'єктно-орієнтованого програмування з активним використанням архітектурних патернів Singleton, Strategy, Observer та Component. Це забезпечує зручне масштабування, розширення і повторне використання коду.

2. Інтеграція авторизації та збереження прогресу через PlayFab

Впроваджено систему авторизації користувача з можливістю реєстрації, входу за логіном і паролем, автоматичним збереженням сесії, а також підтримкою автологіну. Авторизація є обов'язковою умовою для доступу до персоналізованого збереження даних, зокрема монет, рекордів та вибраного персонажа. Застосування хмарного сервісу PlayFab забезпечує платформну незалежність і безпечне збереження ігрового прогресу.

3. Реалізація основи внутрішньоігрової економіки

Розроблено систему монет, яка слугує внутрішньоігровою валютою. Вона передбачає автоматичне нарахування гравцеві монет за результатами забігу, подальше збереження цієї інформації в PlayFab і можливість використання накопичених монет для розблокування нового контенту.

5. Побудова таблиці лідерів на основі загальної шкоди

Реалізовано механізм глобального лідерборду, який формує рейтинг гравців на основі загальної кількості завданої шкоди за всі забіги. Це створює умови для змагальності, повернення гравців у гру та соціального залучення. Дані зберігаються в PlayFab у вигляді глобальної статистики.

6. Універсальність і готовність до розширення

Уся логіка гри розроблена вручну без використання генераторів коду, з чітким поділом відповідальностей та високим ступенем підтримованості. Проєкт структурований так, що його можна легко розвивати: додавати нових ворогів, карти, режими гри, онлайн-функціонал або кооператив.

Таким чином, створений ігровий прототип не лише продемонстрував можливість побудови сучасної модульної гри з нуля, але й показав ефективність поєднання клієнтської логіки Unity з хмарними сервісами PlayFab. Отримані результати можуть слугувати основою для подальших ітерацій розробки, включаючи впровадження багатокористувацького режиму, системи досягнень, PvP або косметичного магазину.

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ НАВЕДЕНОЇ ЧАСТИНИ ПРОЄКТУ

№ п.п.	Ресурс	Призначення
1.	Unity; Розробник: Unity Technologies; Документація: посилання	Використано в якості ігрового рушія та середовища розробки
2.	Мова програмування C# Розробник: Microsoft Документація: посилання	Була обрана мовою програмування для реалізації ігрової логіки
3.	Visual Studio; Розробник: Microsoft Документація: посилання	Використано в якості головного інструмента редактору коду
4.	PlayFab; Розробник: Microsoft Документація: посилання	Використано для баз даних
5.	itch.io Розробник: Leaf Corcoran Документація: посилання	Використано для публікації збірки проєкту