Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Проєктування та розробка 2D-гри на платформі Unity

Виконав: студент IV курсу, групи СП-42
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Леськів Є. Р.
(підпис) (прізвище та ініціали)

Керівник Бойко І. В.
(підпис) (прізвище та ініціали)

Нормоконтроль Стоянов Ю. М.
(підпис) (прізвище та ініціали)

Завідувач кафедри Петрик М. Р.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

Тернопіль
2024

АНОТАЦІЯ

Кваліфікаційна робота бакалавра за спеціальністю 121 – Інженерія програмного забезпечення. Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-42, 2024 рік. Пояснювальна записка до кваліфікаційної роботи на здобуття освітнього ступеню «бакалавр» містить: сторінок 50, рисунків 30, додатків 2.

Тема: Проєктування та розробка 2D-гри на платформі Unity.

У кваліфікаційній роботі розглянуто процес розробки казуальної гри з використанням двигуна Unity. Використання простих геймплеєвих механік та мінімалістичного дизайну позитивно впливає на успішність проекту, забезпечуючи високий користувацький досвід. Чітко визначені завдання та використання діаграм варіантів використання і класів спростили процес розробки і відповідали очікуванням користувачів. Графічний дизайн, покращений завдяки двовимірній комп'ютерній графіці, значно підвищив візуальну привабливість продукту.

Ключові слова: 2D-гра, казуалка, Unity, UML, аркада, діаграма, проєктування, розробка, тестування, тест-кейс, баг-репорт.

ANNOTATION

Bachelor's qualifying work on specialty 121 - Software engineering. Ivan Pulyuy Ternopil National Technical University, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, Group SP-42, 2024. The explanatory note to the qualification work for obtaining the bachelor's degree contains: 50 pages, 30 figures, 2 appendices.

Topic: Design and development of a 2D game on the Unity platform.

The qualifying work examines the process of developing a casual game using the Unity engine. The use of simple gameplay mechanics and minimalistic design has a positive effect on the success of the project, providing a high user experience. Clearly defined tasks and the use of use case and class diagrams simplified the development process and met user expectations. The graphic design, enhanced by two-dimensional computer graphics, greatly increased the visual appeal of the product.

Keywords: 2D game, casual game, Unity, UML, arcade, diagram, design, development, testing, test case, bug report.

ЗМІСТ

АНОТАЦІЯ 4
ANNOTATION 5
ВСТУП 7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 8
1.1 Аналіз подібних проєктних рішень 8
1.2 Аналіз вимог 11
1.3 Опис методології 14
1.4 Висновки. Розділ 1 16
РОЗДІЛ 2. ПРОЄКТУВАННЯ ГРИ 17
2.1 Постановка завдання 17
2.2 Варіанти використання 18
2.3 Діаграма класів 22
2.4 Висновки. Розділ 2 26
РОЗДІЛ 3. ОПИС ТА ТЕСТУВАННЯ РОЗРОБЛЕНОЇ ГРИ 28
3.1 Опис графіки, дизайну. 28
3.2 Опис механізму гри 32
3.3 Тестування гри 41
3.4 Висновки. Розділ 3 45
РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ 47
4.1 Працездатність людини – оператора. 47
4.2 Заходи, що покращують умови праці оператора. 49
ВИСНОВКИ 53
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 54
ДОДАТОК А – Програмний код 57
ДОДАТОК Б – Диск з роботою 66

ВСТУП

У сучасному світі розробка відеоігор досягла значних висот і продовжує стрімко розвиватися, відкриваючи нові горизонти для цієї індустрії щороку. Однак не всі мають можливість грати в усі ігри, що існують. Це може бути пов'язано з відсутністю необхідного пристрою або з тим, що деякі ігри вимагають високих технічних параметрів.

Незважаючи на це, 2D-ігри залишаються доступними для широкої аудиторії, оскільки вони, як правило, не потребують потужного обладнання. Ці ігри можуть бути досить простими, але водночас надзвичайно захопливими. Основні дії в таких іграх можуть включати тапання екрану, рух персонажа або взаємодію з об'єктами за допомогою дотиків чи миші.

Варто зазначити, що 2D-ігри можуть зацікавити людей усіх вікових категорій. Наразі існує безліч таких ігор, які можуть бути не лише розважальними, а й мати освітню, виховну або оздоровчу мету. Вони можуть допомагати розвивати логічне мислення, дрібну моторику, уважність та інші корисні навички.

Застосувавши платформу Unity для розробки надає велику кількість інструменті для створення гри чи то додатку. Можна створювати 2D і 3D проєкти, використовуючи анімації, фізику, звуки, та багато іншого. На Unity можна створити гру для інших платформ, як Windows, macOS, IOS, Android, PS5, та багатьох інших [1].

Основна мета нашого проєкту – створення 2D-гри, яка слугуватиме засобом для відпочинку і відволікання від буденних справ, таких як робота чи навчання. Щоб гравець міг відволіктися в будь-який зручний для нього час, гра буде розроблена для мобільних пристроїв, адже смартфон є майже у кожної людини. Мобільна платформа забезпечить доступність гри для більшої кількості користувачів, дозволяючи грати у будь-якому місці та у будь-який час.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1Аналіз подібних проєктних рішень

Гра Knife Hit – це популярна мобільна гра від студії Ketchapp, вона відома своїми простими але захоплюючими аркадними іграми. Гра доступна для платформ iOS і Android. Основна мета гри полягає в тому, щоб метати ножі в обертові колоди, уникаючи попадання в уже вбиті ножі та інші перешкоди [2].

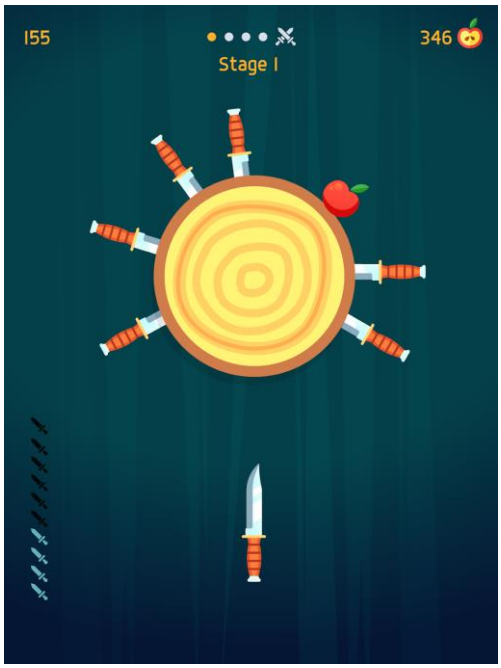

Рисунок 1.1 – Скріншот гри Knife Hit

Опис гри:

- Геймплей: Гравець метає ножі у дерев'яну колоду, яка обертається. Кожен рівень має певну кількість ножів, які необхідно метнути в колоду. Якщо гравець потрапляє ножом у вже вбитий ніж, рівень потрібно починати заново.
- Боси: Через певну кількість рівнів гравець стикається з "босом" – великою колодою або іншим об'єктом, який має специфічні властивості або складніший рух.

- Складність: З кожним рівнем складність гри зростає. Колоди починають обертатися швидше або змінюють напрямок руху, додаються рухомі перешкоди.
- Нагороди та бонуси: У грі можна збирати яблука, які можна обмінювати на нові ножі. Також є спеціальні ножі, які можна отримати за виконання певних завдань або досягнення в грі.
- Графіка та звук: Гра має мінімалістичний дизайн з яскравими кольорами та приємними звуковими ефектами, які доповнюють атмосферу гри.

Отже Knife Hit – це захоплююча аркадна гра, яка ідеально підходить для коротких ігрових сесій. Вона вимагає від гравця точності, уваги та терпіння, що робить її як захоплюючою, так і викликом. Гра пропонує простий, але затягаючий геймплей, який може сподобатися широкому колу гравців.

Гра Stack – це популярна мобільна гра від розробника Ketchapp, відомого своїми простими, але захоплюючими аркадними іграми. Stack доступна для платформ iOS та Android, і вона представляє собою чудовий приклад мінімалістичного дизайну, який поєднує простоту геймплею з візуальною естетикою [3].

Опис гри:

- Геймплей: Основна мета гри – будувати вежу, складаючи блоки один на один. Гравець починає з одного великого блоку, на який потрібно розміщувати наступні блоки. Кожен новий блок рухається горизонтально вздовж екрану з певною швидкістю. Гравець повинен вчасно натиснути на екран, щоб зупинити блок і розмістити його на попередньому. Якщо новий блок не співпадає повністю з попереднім, частина блоку, що не співпала, відрізається, і наступний блок стає меншим.

Рисунок 1.2 – Скріншот гри Stack

- Складність: Гра з кожним новим рівнем гри блоки стають меншими, що ускладнює точне розміщення кожного наступного блоку. Чим вища вежа, тим більше точності потрібно для її подальшого будівництва. Швидкість руху блоків може також збільшуватись, додаючи додатковий виклик для гравця. Гра продовжується доти, поки гравець не зробить помилку, тобто повністю не промахнеться повз попередній блок.
- Нагороди та бонуси: Гра пропонує різні нагороди, такі як візуальні теми для блоків, які можна розблокувати, досягаючи певних висот або виконуючи певні завдання. Ці теми змінюють вигляд блоків і фон, додаючи різноманітності та інтересу до гри. Гравці можуть змагатися з друзями або іншими гравцями за найвищий результат, що стимулює повторні ігрові сесії.
- Графіка та звук: Stack має стильний мінімалістичний дизайн з чіткими лініями та яскравими кольорами. Візуальні ефекти створюють відчуття гармонії та задоволення від гри. Звуковий супровід включає приємні звукові ефекти, які підсилюють відчуття успіху при вдалих діях гравця.

Графіка є однією з ключових особливостей, які роблять гру такою привабливою та легко впізнаваною.

Отже Stack – це захоплююча аркадна гра, яка ідеально підходить для коротких ігрових сесій. Вона вимагає від гравця точності, уваги та терпіння, що робить її як захоплюючою, так і викликом. Простий геймплей, стильний дизайн та мінімалістична естетика роблять Stack популярною грою серед широкого кола гравців, які цінують як візуальну привабливість, так і задоволення від гри.

1.2 Аналіз вимог

Гра призначена для розважальних цілей. Вона розроблена для того, щоб гравці могли відволіктися від рутини під час перерви або вільного часу.

Функціональні вимоги:

1. Вхід у гру:
   - Встановлення та запуск: Для входу в гру потрібно спершу встановити її на пристрій, а потім просто запустити.
   - Інтерфейс користувача: При запуску гри користувач побачить основне меню з різними опціями, такими як "Плей", "Налаштування" та "Вихід".
2. Запуск процесу гри:
   - Меню гри: У головному меню натискання кнопки "Плей" запускає процес гри. Після завершення гри натискання кнопки "Рестарт" дозволяє плавно перезапустити гру.
   - Інтуїтивність: Меню повинно бути інтуїтивно зрозумілим, щоб користувач легко знаходив потрібні опції.
3. Процес гри:
   - Рух кубика: Гравець керує кубиком, який рухається до точок для їх збору. Потрібно уникати ворогів, які з'являються на шляху.

- Механіка збору: Кожна зібрана точка додає певну кількість балів до рахунку гравця.

4. Процес завершення гри:
   - Колізія з ворогом: Якщо кубик стикається з ворогом, гра завершується, і гравець бачить повідомлення про поразку.
   - Рестарт: Гравець має можливість перезапустити гру, не виходячи до головного меню.
5. Відкриття налаштувань:
   - Перехід до налаштувань: Через меню натискання кнопки "Налаштування" відкриває вікно з можливістю змінювати параметри гри.
6. Перегляд гри:
   - Графіка: Гра повинна мати просту, але привабливу графіку з мінімальною кількістю елементів, щоб не перевантажувати користувача.

Нефункціональні вимоги:

1. Безпека даних:
   - Конфіденційність: Гра не збирає та не використовує ніяких особистих даних користувача.
   - Захист: Усі дані гри зберігаються локально, без підключення до інтернету.
2. Продуктивність:
   - Швидкодія: Гра повинна працювати без затримок, забезпечуючи плавний ігровий процес.
   - Реакція: Всі команди та дії гравця повинні виконуватись миттєво.
3. Технічне обслуговування:
   - Простота: Гра повинна бути легкою в обслуговуванні, з можливістю швидкого внесення змін та оновлень.
   - Документація: Повна технічна документація повинна бути доступна для розробників.

4. Масштабованість:
   - Розширення: Гра повинна бути готова до збільшення кількості користувачів і впровадження нових функцій без втрати продуктивності.
5. Сумісність:
   - Пристрої: Гра повинна коректно відображатися та функціонувати на різних пристроях, включаючи мобільні телефони, планшети.
6. Відповідність:
   - Стандарти: Гра повинна відповідати загальноприйнятим стандартам розробки програмного забезпечення та ігор.
   - Користувацький досвід: Відповідність найкращим практикам забезпечення позитивного користувацького досвіду.

Додаткові вимоги:

1. Адаптивність:
   - Різні рівні: Гра повинна пропонувати різні рівні складності, щоб задовольнити потреби як початківців, так і досвідчених гравців.
2. Зворотний зв'язок:
   - Оцінки та відгуки: Гравці повинні мати можливість залишати відгуки та оцінки про гру, щоб розробники могли враховувати їхні побажання та покращувати гру.
3. Соціальні функції:
   - Поділитися: Гравці можуть ділитися своїми досягненнями у соціальних мережах.
   - Рейтингові таблиці: Впровадження рейтингових таблиць, де гравці можуть порівнювати свої результати з іншими.
4. Підтримка кількох мов:
   - Інтернаціоналізація: Гра повинна підтримувати кілька мов для розширення аудиторії.
5. Резервне копіювання:

- Збереження прогресу: Можливість збереження ігрового прогресу в хмарі для відновлення при зміні пристрою.

1.3Опис методології

Scrum – це гнучка методологія управління проектами, яка допомагає командам структурувати та керувати своєю роботою, використовуючи набір цінностей, принципів і практик. Подібно до регбійної команди, що тренується для великої гри, Scrum заохочує команди вчитися на досвіді, самоорганізовуватися під час вирішення завдань та постійно вдосконалюватися, аналізуючи свої успіхи та невдачі [4].

Найчастіше Scrum використовується у сфері розробки програмного забезпечення, однак його принципи можуть застосовуватися до будь-якої командної роботи. Це одна з причин, чому Scrum настільки популярний. Scrum описує набір зустрічей, інструментів і ролей, які допомагають командам структурувати та ефективно керувати своєю роботою.

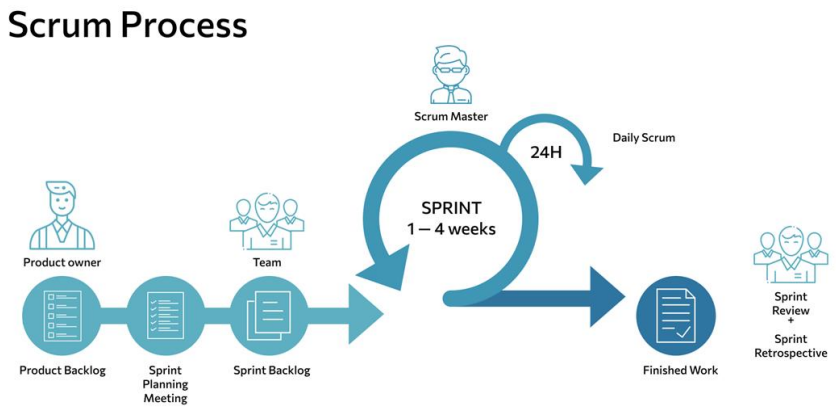

Рисунок 1.3 – Зображення методології Scrum

Основні елементи Scrum

Команда Scrum

До складу команди входять три ключові ролі: власник продукту, скрам-майстер і команда розробників. Власник продукту відповідає за формування та пріоритизацію списку завдань (беклогу продукту), скрам-майстер забезпечує дотримання процесів Scrum і допомагає команді покращувати свою роботу, а команда розробників безпосередньо виконує завдання.

Цінності Scrum

Scrum ґрунтується на п'яти основних цінностях: відданість, сміливість, зосередженість, відкритість та повага. Вони допомагають командам ефективніше співпрацювати та досягати своїх цілей.

Артефакти Scrum

1. Беклог продукту – це динамічний список завдань, який постійно підтримується та пріоритизується власником продукту.
2. Беклог спринту – набір завдань з беклогу продукту, які команда вибрала для виконання в поточному спринті.
3. Інкремент – завершений продукт або функціонал, створений під час спринту, який відповідає критеріям "готовності".

Церемонії Scrum

1. Планування спринту – зустріч, на якій команда визначає, які завдання з беклогу продукту будуть виконані під час спринту.
2. Щоденний Scrum (стандап) – коротка щоденна зустріч, де команда обговорює свій прогрес і перешкоди.
3. Огляд спринту – зустріч в кінці спринту, де команда демонструє результати своєї роботи та отримує зворотний зв'язок.
4. Ретроспектива спринту – аналіз того, що спрацювало добре і що можна покращити в наступних спринтах.

Відмінності між Scrum та іншими гнучкими методологіями

Scrum часто плутають з Agile, оскільки обидві методології зосереджені на постійному вдосконаленні. Однак Agile – це філософія, яка наголошує на безперервних інкрементальних покращеннях, тоді як Scrum – це конкретний фреймворк, що допомагає організувати роботу команди. Іншою популярною

методологією є Kanban, яка фокусується на візуалізації роботи і обмеженні кількості незавершених завдань.

Переваги Scrum

Scrum допомагає розподілити складні завдання на менші, керовані частини, що робить його ідеальним для складних проектів. Завдяки чітким ролям і запланованим подіям, Scrum забезпечує прозорість і колективну відповідальність протягом усього циклу розробки. Часті релізи мотивують команду та роблять користувачів щасливими, оскільки вони можуть бачити прогрес за короткий проміжок часу.

Отже, scrum – це потужний інструмент для управління проектами, який допомагає командам ефективно організовувати свою роботу, швидко адаптуватися до змін і постійно вдосконалювати свої процеси. Хоча впровадження Scrum може вимагати часу та зусиль, його довгострокові переваги роблять його вартим для багатьох організацій.

1.4 Висновки. Розділ 1

Аналіз предметної області показав, що популярні аркадні ігри, такі як Knife Hit і Stack від Ketchapp, приваблюють гравців своїм простим, але захоплюючим геймплеєм і мінімалістичним дизайном. Knife Hit вимагає точності та уваги, метаючи ножі в обертові колоди, а Stack викликає гравців складати блоки у вежу, що вимагає швидкої реакції.

Функціональні та нефункціональні вимоги до подібних ігор забезпечують плавний ігровий процес, безпеку даних і високу продуктивність. Застосування методології Scrum дозволяє ефективно організувати роботу команди, швидко адаптуватися до змін і постійно вдосконалювати процеси.

Таким чином, аналіз подібних ігор і вибір відповідної методології створюють міцну основу для розробки успішного проекту

РОЗДІЛ 2. ПРОЄКТУВАННЯ ГРИ

2.1 Постановка завдання

Мета проєкту: Створення інтерактивної гри, яка дозволяє користувачам відпочити та розважитися у вільний час, збираючи точки і уникаючи ворогів.

Короткий опис гри: "TapCube" — це аркадна гра, в якій гравець керує кубиком, збираючи точки та уникаючи зіткнень з ворогами. Гра має бути простою у використанні та цікавою для різних вікових груп.

Завдання проекту:

1. Розробка інтерфейсу користувача:
   - Створення головного меню з кнопками для запуску гри, налаштувань та виходу.
   - Розробка екранів налаштувань для регулювання звуку та інших параметрів гри.
   - Створення екрану завершення гри з можливістю рестарту та повернення до головного меню.
2. Реалізація ігрового процесу:
   - Програмування механіки руху кубика, збір точок і уникнення ворогів.
   - Реалізація логіки завершення гри при зіткненні з ворогом.
   - Забезпечення плавного перезапуску гри після завершення.
3. Налаштування та опції:
   - Додавання можливостей для налаштування рівня звуку та інших параметрів гри.
   - Забезпечення підтримки декількох мов інтерфейсу.
   - Розробка налаштувань графіки для різних типів пристроїв.
4. Оптимізація продуктивності:
   - Оптимізація коду гри для плавної роботи на різних платформах (мобільні пристрої, планшети, комп'ютери).

- Забезпечення швидкого завантаження гри і відсутності лагів під час ігрового процесу.

5. Безпека даних:
   - Гра не повинна збирати або зберігати особисті дані користувачів.
   - Всі дані повинні зберігатися локально на пристрої користувача.
6. Тестування:
   - Проведення тестування для виявлення та виправлення помилок.
   - Збір зворотного тестування і внесення необхідних змін для покращення гри.

Очікувані результати:

- Випуск функціональної версії гри, яка працює на різних платформах.
- Отримання позитивних відгуків від користувачів.
- Відсутність помилок і проблем з продуктивністю.

Критерії успіху:

- Гра працює без помилок і збоїв на всіх підтримуваних платформах.
- Позитивні відгуки та високий рейтинг у магазинах додатків.
- Проект завершений в межах запланованого часу і бюджету.

2.2 Варіанти використання

Для подальшого розроблення було спроєктовано повну діаграму варіантів використання для подальшого розроблення веб-сайту.

Діаграма варіантів використання (Use Case Diagram) в UML зображує взаємозв'язки між акторами та варіантами використання в системі. Вона відображає різні сценарії використання системи різними типами користувачів і зазвичай доповнюється іншими типами діаграм [5].

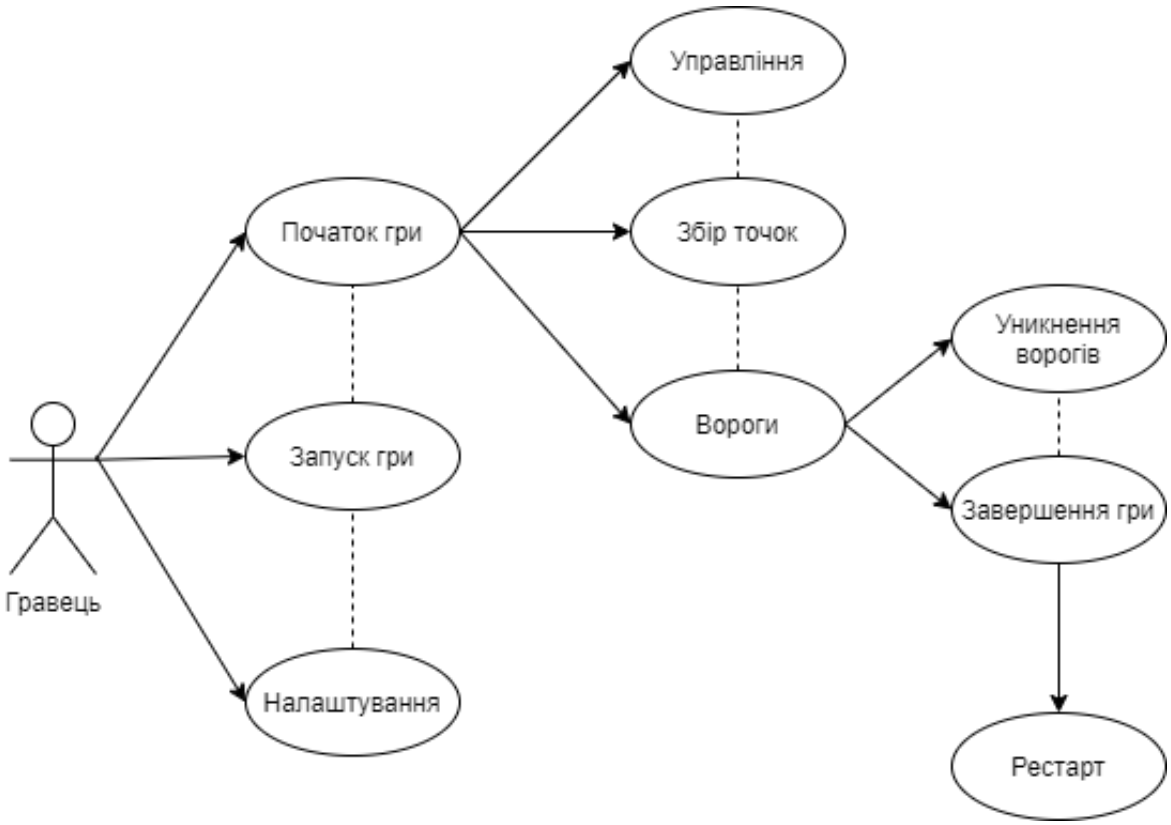

Рисунок 2.1 – Повна діаграма варіантів використання

Основні варіанти використання

1. Запуск гри

   Актори: Гравець

   Короткий опис: Гравець відкриває встановлену гру на своєму пристрої.

   Основний потік події:

   - Встановлення гри;
   - Відкриття гри.

   Передумови: Гра має бути встановлена.

   Результат: Відображається головне меню гри.

2. Початок гри

   Актори: Гравець

   Короткий опис: Гравець натискає кнопку "Плей" у головному меню, щоб розпочати гру.

   Основний потік події:

   - Відкрита гра;

- Вікно головного меню;
- Натиснення кнопки.

Передумови: Гра запущена, і гравець знаходиться у головному меню.

Результат: Починається ігровий процес, де гравець керує кубиком.

3. Управління кубиком

Актори: Гравець

Короткий опис: Гравець використовує сенсорний екран або клавіатуру для керування рухом кубика.

Основний потік події:

- Запустити гру;
- Керування кубиком.

Передумови: Гра розпочата.

Результат: Кубик рухається відповідно до команд гравця.

4. Збір точок

Актори: Гравець

Короткий опис: Гравець керує кубиком, щоб збирати точки, розташовані на ігровому полі.

Основний потік події:

- Запустити гру;
- Керувати кубиком;
- Рухати кубик до точок.

Передумови: Кубик знаходиться на ігровому полі.

Результат: Зібрані точки додаються до рахунку гравця.

5. Уникнення ворогів

Актори: Гравець

Короткий опис: Гравець керує кубиком, щоб уникати зіткнень з ворогами.

Основний потік події:

- Керувати кубиком;
- Збирати точки;

- Під час гри не зіткнутися з ворогами.

Передумови: Вороги присутні на ігровому полі.

Результат: Кубик успішно уникає ворогів.

6. Завершення гри

Актори: Гравець

Короткий опис: Гра завершується, коли кубик стикається з ворогом.

Основний потік події

- Грати в гру;
- Зіткнення з ворогом.

Передумови: Кубик стикається з ворогом.

Результат: Відображається екран завершення гри з опціями рестарту або повернення до головного меню.

7. Рестарт гри

Актори: Гравець

Короткий опис: Після завершення гри гравець може натиснути кнопку "Рестарт" для початку нової гри.

Основний потік події:

- Грати в гру;
- Зіткнення з ворогом;
- Вікно з завершення гра.

Передумови: Гра завершена, і відображається екран завершення гри.

Результат: Починається нова гра.

8. Налаштування

Актори: Гравець

Короткий опис: Гравець натискає кнопку "Налаштування" у головному меню для переходу до екрану налаштувань.

Основний потік події:

- Відкрити гру;
- Вікно головного меню;
- Натиснення кнопки.

Передумови: Гравець знаходиться у головному меню.

Результат: Відображається екран налаштувань, де гравець може змінити різні параметри гри.

2.3 Діаграма класів

Діаграма класів в Уніфікованій мові моделювання (UML) — це статична структурна діаграма, що описує структуру системи, показуючи класи системи, їх атрибути, операції (або методи) та взаємозв'язки між об'єктами [6].

Клас діаграми в UML складається з набору класів та набору взаємозв'язків між ними. Кожен клас представляє групу об'єктів з схожими ролями в системі. Він має структурні характеристики (атрибути), що описують стан об'єкта, та поведінкові характеристики (операції), що визначають можливості взаємодії об'єктів.

Графічне представлення класу включає назву класу, його атрибути та операції. Взаємозв'язки між класами можуть бути у формі спадкування (у випадку "є-це" відношення), асоціації, агрегації чи композиції. Діаграма класів використовується для моделювання статичної структури класифікаторів у системі та надає основну нотацію для інших структурних діаграм, які підтримує UML.

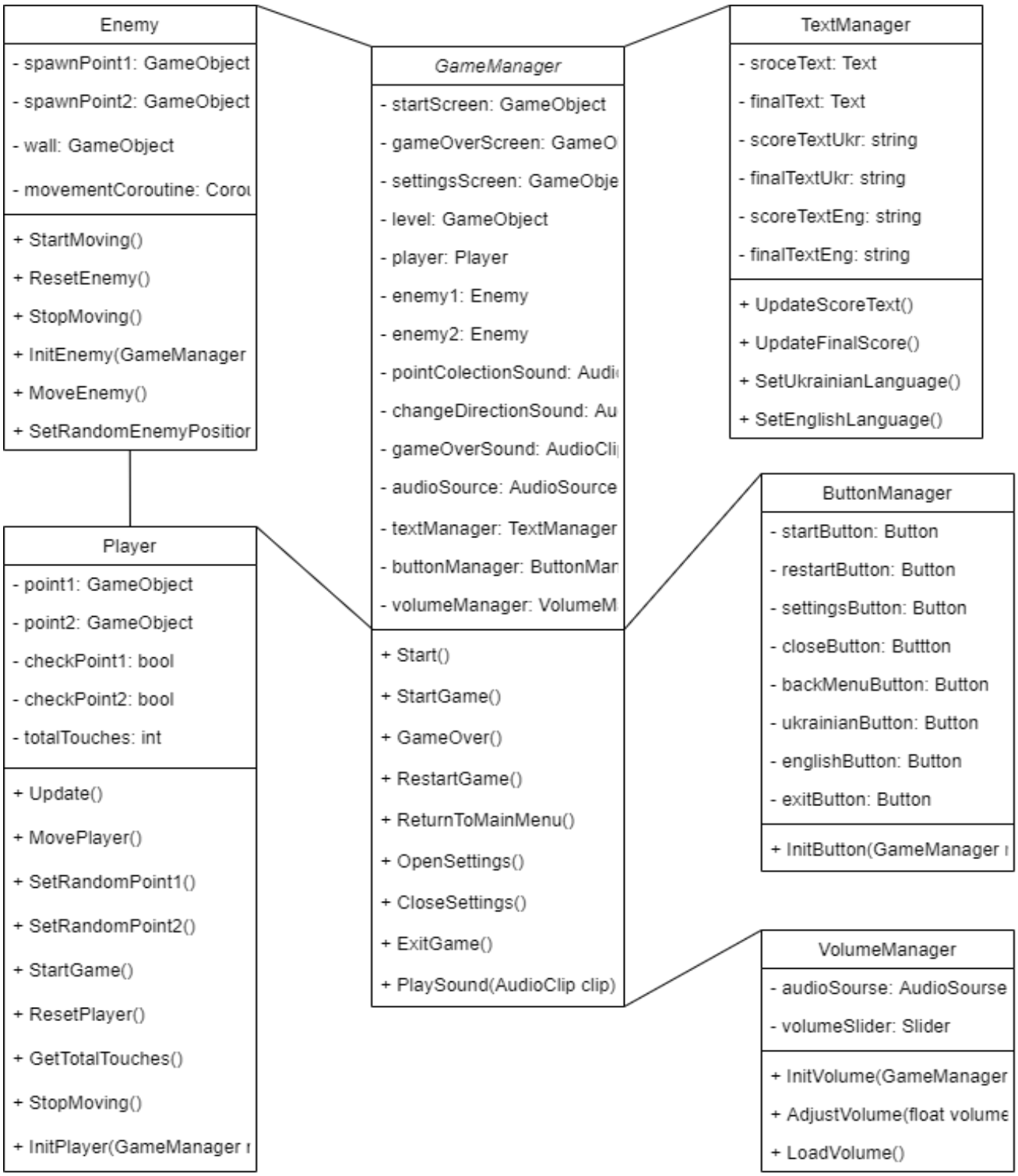

Рисунок 2.2 – Повна діаграма класів

1. GameManager

Керує загальним станом гри та її перебігом. Він обробляє переходи між різними екранами гри (екран старту, екран завершення гри, екран налаштувань) та керує ініціалізацією та взаємодією гравця, ворогів та інших елементів гри.

- startScreen, gameOverScreen, settingsScreen, level: Об'єкти гри, які представляють різні екрани та рівень.
- player, enemy1, enemy2: Екземпляри класів Player та Enemy.
- pointCollectionSound, changeDirectionSound, gameOverSound: Аудіо-кліпи для різних подій гри.
- audioSource: Компонент AudioSource для відтворення звуків.
- textManager, buttonManager, volumeManager: Менеджери для керування текстом, кнопками та гучністю відповідно.
- Start(): Ініціалізує аудіо-джерело, менеджери кнопок і гучності, завантажує збережені налаштування гучності та встановлює англійську мову за замовчуванням.
- StartGame(): Починає гру, активуючи рівень, ініціалізуючи та запускаючи гравця і ворогів.
- GameOver(): Обробляє логіку завершення гри, деактивуючи рівень, зупиняючи гравця і ворогів, та оновлюючи остаточний рахунок.
- RestartGame(): Перезапускає гру, скидаючи стан гравця і ворогів та починаючи гру знову.
- ReturnToMainMenu(): Повертається до головного меню, скидаючи стан гравця і ворогів та активуючи екран старту.
- OpenSettings(), CloseSettings(): Відкриває та закриває екран налаштувань.
- ExitGame(): Вихід з гри.
- PlaySound(AudioClip clip): Відтворює вказаний аудіо-кліп.

2. Player

Керує рухом гравця та його взаємодіями у грі.

- point1, point2: Об'єкти гри, які представляють цільові точки для переміщення гравця.
- checkPoint1, checkPoint2: Булеві значення, які вказують, до якої точки рухається гравець.
- totalTouches: Ціле число, що відстежує кількість разів, коли гравець досяг цільової точки.

- Update(): Викликається кожен кадр для переміщення гравця.
- MovePlayer(): Обробляє логіку переміщення гравця між точками та оновлює загальну кількість дотиків.
- SetRandomPoint1(), SetRandomPoint2(): Встановлює випадкові позиції для цільових точок.
- StartGame(): Ініціалізує стан гравця для нової гри.
- ResetPlayer(): Скидає стан гравця.
- GetTotalTouches(): Повертає загальну кількість дотиків.
- StopMoving(): Зупиняє рух гравця.
- InitPlayer(GameManager manager): Ініціалізує гравця з екземпляром менеджера гри.

3. Enemy

Керує рухом ворогів та їх взаємодіями у грі.

- spawnPoint1, spawnPoint2, wall: Об'єкти гри, які представляють точки появи та стіну для руху ворога.
- movementCoroutine: Корутин для обробки руху ворога.
- InitEnemy(GameManager manager): Ініціалізує ворога з екземпляром менеджера гри.
- StartMoving(): Запускає корутин для руху ворога.
- ResetEnemy(): Скидає позицію ворога та зупиняє корутин руху, якщо він активний.
- StopMoving(): Зупиняє корутин руху ворога.
- MoveEnemy(): Корутин, що обробляє логіку руху ворога.
- SetRandomEnemyPosition(): Встановлює випадкову позицію для ворога в межах точок появи.

4. TextManager

Керує оновленням тексту гри, включаючи рахунок та налаштування мови.

- scoreText, finalText: UI елементи Text для відображення рахунків.
- scoreTextEnglish, finalTextEnglish, scoreTextUkrainian, finalTextUkrainian: Рядки для тексту на різних мовах.

- UpdateScoreText(): Оновлює текст рахунку відповідно до поточної мови.
- UpdateFinalScore(int score): Оновлює остаточний текст рахунку відповідно до поточної мови.
- SetUkrainianLanguage(): Встановлює українську мову та оновлює текст.
- SetEnglishLanguage(): Встановлює англійську мову та оновлює текст.

5. ButtonManager

Керує ініціалізацією та діями кнопок UI у грі.

- startButton, restartButton, settingsButton, closeButton, backMenuButton, ukrainianButton, englishButton, exitButton: UI елементи Button.
- InitButtons(GameManager manager): Ініціалізує кнопки та призначає їх відповідні дії.

6. VolumeManager

Керує налаштуваннями гучності гри та взаємодією з повзунком гучності UI.

- audioSource: Компонент AudioSource для керування гучністю.
- volumeSlider: UI елемент Slider для регулювання гучності.
- Start(): Додає слухача до повзунка гучності для регулювання гучності.
- InitVolume(GameManager manager): Ініціалізує аудіо-джерело з екземпляром менеджера гри.
- AdjustVolume(float volume): Регулює гучність аудіо-джерела та зберігає налаштування.
- LoadVolume(): Завантажує збережене налаштування гучності та оновлює повзунок.

2.4 Висновки. Розділ 2

У цьому розділі було детально розглянуто основні етапи проєктування гри "TapCube". Початкова частина розділу зосереджується на постановці завдання, що включає визначення мети проєкту, основних завдань та критеріїв успіху.

Детальний опис гри дозволяє зрозуміти, яким чином гравець взаємодіє з ігровим середовищем, що включає збір точок та уникнення ворогів.

Далі розглянуто діаграму варіантів використання, яка допомагає візуалізувати різні сценарії взаємодії гравця з грою, включаючи запуск гри, керування кубиком, збір точок, уникнення ворогів та завершення гри. Кожен варіант використання детально описано, що дозволяє чітко уявити послідовність дій гравця та необхідні передумови для кожного сценарію.

Розділ також містить діаграму класів, яка описує структуру гри на рівні класів, їх атрибути, методи та взаємозв'язки. Кожен клас гри, включаючи GameManager, Player, Enemy, TextManager, ButtonManager та VolumeManager, має чітко визначені ролі та функції, що забезпечують правильну взаємодію елементів гри.

Завдяки детальному опису елементів гри та їх взаємодії, створено міцну основу для подальшого розроблення та реалізації проєкту. Розроблені діаграми та описи допоможуть у забезпеченні високої якості гри, оптимізації її продуктивності та досягненні позитивного користувацького досвіду.

РОЗДІЛ 3. ОПИС ТА ТЕСТУВАННЯ РОЗРОБЛЕНОЇ ГРИ

3.1 Опис графіки, дизайну.

Дизайн – це процес створення рішень для людей, щоб вони краще взаємодіяли з оточенням. Він орієнтований на користувача і включає в себе всі аспекти від продуктів до інтерфейсів, спрямований на вирішення конкретних потреб і проблем [7].

Графіка – це візуальне представлення даних, що включає малюнки, фотографії, графіки, діаграми, символи та інші форми, які можуть бути використані для декоративних або інформаційних цілей. На комп'ютерах графіка поділяється на растрову (бітмапну) та векторну форми, що дозволяє створювати та редагувати зображення для друку, відображення та зберігання на екранах.

2D комп'ютерна графіка відіграє важливу роль у створенні комп'ютерних ігор, особливо в їх ранній історії та в сучасних жанрах, які не потребують тривимірного простору. Цей тип графіки використовується для створення зображень і анімацій на основі двовимірних моделей, таких як геометричні фігури та текстури [8].

Ранні відеоігри, розроблені для 8-бітних консолей і домашніх комп'ютерів, широко використовували спеціалізовані 2D графічні чіпи. Приклади таких чіпів включають ANTIC від Atari та VIC-II від Commodore. Ці апаратні засоби дозволяли виконувати важливі графічні операції, такі як обробка спрайтів та швидке переміщення блоків даних (бліттинг).

У сучасних 2D іграх графічні елементи створюються за допомогою растрових карт, або піксельних матриць, які містять зображення персонажів, фонів та інших елементів. Використання шарів дозволяє розробникам створювати складні сцени, де кожен шар може редагуватися незалежно, що полегшує процес створення та анімації.

Багато графічних інтерфейсів, таких як Mac OS та Microsoft Windows, також використовують двовимірну графіку, що сприяло її популярності в ігровій

індустрії. Сучасні 2D графічні редактори, такі як Adobe Illustrator, CorelDraw, GIMP та Inkscape, надають потужні інструменти для створення та редагування графіки, яка використовується в іграх.

Загалом, 2D комп'ютерна графіка залишається важливим елементом у розробці ігор, забезпечуючи ефективні та гнучкі засоби для створення візуально привабливих ігрових проектів.

Вигляд головного персонажа, ворогів, точок для збирання, кнопок, тексту та фону, а ще стінок біля, який появляються точки зображено на рисунках 3.1-3.6.

Рисунок 3.1 – Головний персонаж

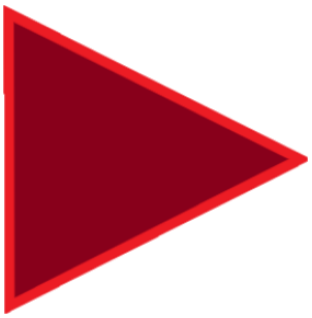

Рисунок 3.2 – Вороги

Рисунок 3.3 – Точки для збирання

Рисунок 3.4 – Вигляд решти кнопок

Рисунок 3.5 – Фон та текст

Рисунок 3.6 – Стінки

Основні складові гри включають:

1. Головне меню: Гравець вітається початковим екраном на якому видно назву гри та кнопки з три кнопки “Play”, “Settings”, “Exit”.

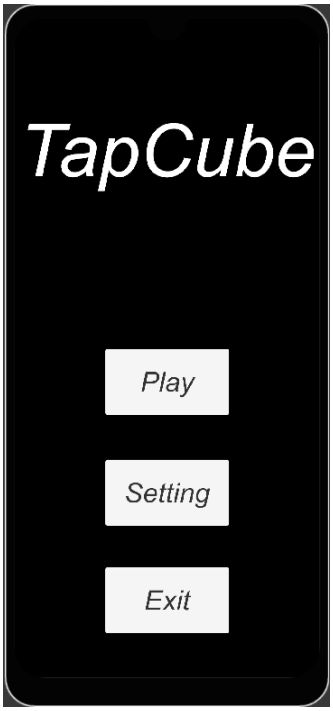

Рисунок 3.7 – Головне меню

2. Панель гри: На цьому екрані гравець керує кубом, використовуючи торкання екрану для зміни напрямку руху також він автоматично рухається між двома точками, які потрібно зібрати, що призводить до збільшення рахунку.

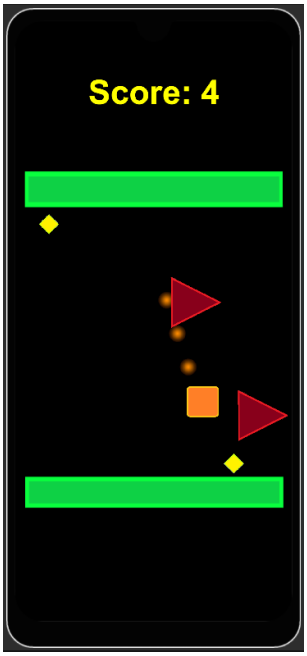

Рисунок 3.8 – Процес гри

3. Панель закінчення гри: Після завершення гри на екрані відображається кінцевий рахунок гравця та кнопки "Restart" і "Main Menu" для початку нової гри або повернення на головний екран.

Рисунок 3.9 – Панель закінчення гри

4. Налаштування: Гравець може налаштувати гучність звуку за допомогою слайдера та вибрати мову гри між англійською та українською.

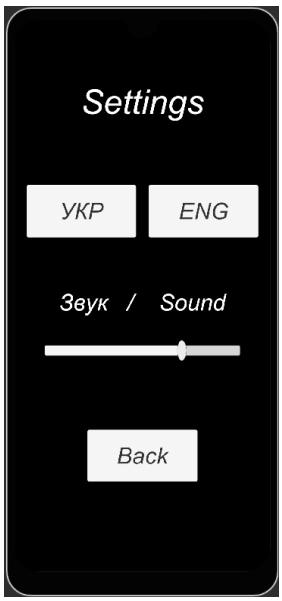

Рисунок 3.10 – Налаштування

3.2 Опис механізму гри

Гра "TapCube" - це казуальна аркадна ігра, створена на платформі Unity. Головна мета гравця - керувати рухом куба і збирати якомога більше очок, уникуючи зіткнень з рухаючимися ворожими об'єктами. Ось детальний опис цієї казуальної гри:

Гра "TapCube" створена для максимального комфорту і веселощів. Гравець керує кубом, який автоматично рухається між двома точками на екрані. Гравець повинен торкатися екрану, щоб змінювати напрямок руху куба та збирати розташовані точки для набуття максимальної кількості очок. Проте гравцеві необхідно уникати зіткнень з двома ворожими об'єктами, які рухаються вгору та вниз по екрану, створюючи перешкоди.

Unity - це інтегрована середовище розробки ігор, яка дозволяє створювати якісні проекти у 2D та 3D форматах для різних платформ, включаючи Windows, macOS, iOS, Android та консолі. Ця платформа надає широкі можливості для роботи з графікою, анімаціями, фізикою, звуком і музикою, що дозволяє розробникам створювати реалістичні і захоплюючі ігри [1].

Особливо вражаюча можливість Unity - це його підтримка мобільних платформ. Розробники можуть оптимізувати ігри для iOS та Android, використовуючи спеціальні інструменти та техніки, що дозволяють досягти високої продуктивності й ефективності на цих пристроях. Це робить Unity ідеальним вибором для тих, хто планує створювати ігри для широкої аудиторії мобільних користувачів [1].

Крім того, Unity має велику та активну спільноту розробників, що стимулює обмін знаннями, підтримку і вдосконалення навичок. Існує безліч онлайн-ресурсів, таких як документація, форуми та навчальні матеріали, які сприяють швидкому і ефективному освоєнню цієї платформи.

Казуальні ігри - це підгрупа відеоігор, спрямованих на простоту в освоєнні, швидкість гри і доступність для гравців будь-якого віку та рівня навичок. Ці ігри часто створюються незалежними розробниками або меншими компаніями, які спеціалізуються на створенні простих і розважальних ігор.

На відміну від складніших ігор, таких як шутери від першої особи чи рольові ігри, казуальні ігри мають просте і легко зрозуміле інтерфейс, що дозволяє швидко освоювати їх. Основна увага у таких іграх зосереджена на механіках гри, які не вимагають значних зусиль для навчання чи часових вкладень, наприклад, головоломки чи карточні ігри [9].

Основні елементи коду:

1. Гравець і збір точок: Контролюється гравцем і переміщується між двома точками збору та збирає ці точки для збільшення свого рахунку. Лістинг зображено на рисунку 3.11.

```
void MovePlayer()
{
    if (checkPoint1)
    {
        transform.position = Vector2.MoveTowards(transform.position, point1.transform.position, 2f * Time.deltaTime);
        if (Vector2.Distance(transform.position, point1.transform.position) < 0.1f)
        {
            PlaySound(pointCollectionSound);
            totalTouches++;
            UpdateScoreText();
            checkPoint1 = false;
            checkPoint2 = true;
            SetRandomPoint1();
        }
    }
    else if (checkPoint2)
    {
        transform.position = Vector2.MoveTowards(transform.position, point2.transform.position, 2f * Time.deltaTime);
        if (Vector2.Distance(transform.position, point2.transform.position) < 0.1f)
        {
            PlaySound(pointCollectionSound);
            totalTouches++;
            UpdateScoreText();
            checkPoint1 = true;
            checkPoint2 = false;
            SetRandomPoint2();
        }
    }
}
```

Рисунок 3.11 – Метод руху гравця і збір точок

Цей метод відповідає за рух гравця між двома точками (point1 і point2). Гравець переміщується до ближньої точки зі швидкістю 2 одиниці на секунду. Якщо гравець досягає одну з точок, відтворюється звук збору точки, підраховується загальна кількість зібраних очок, оновлюється текст рахунку і гравець переходить до наступної точки збору. Метод також випадковим чином розміщує нові точки збору в межах заданих координат.

2. Ворожі об'єкти: Рухаються вгору і вниз, якщо гравець в них врізається, гра завершується. Лістинг зображено на рисунках 3.12 – 3.13.

```
IEnumerator MoveEnemy(GameObject enemy)
{
    while (true)
    {
        while (enemy.transform.position.x <= wall.transform.position.x)
        {
            enemy.transform.Translate(Vector2.right * 2f * Time.deltaTime);
            yield return null;
        }
        SetRandomEnemyPosition(enemy);
    }
}
```

Рисунок 3.12 – Метод руху ворогів

Цей метод виконує рух ворожих об'єктів (enemy1 і enemy2). Ворожі об'єкти рухаються праворуч зі швидкістю 2 одиниці на секунду, поки не досягнуть межі

wall.transform.position.x. Після досягнення межі, ворожі об'єкти повертаються на випадкову позицію в межах заданих координат (spawnPoint1 до spawnPoint3). Цей процес повторюється постійно, створюючи відчуття постійного руху для ворожих об'єктів.

```
void OnTriggerEnter2D(Collider2D other)
{
    if (other.gameObject == enemy1 || other.gameObject == enemy2)
    {
        PlaySound(gameOverSound);
        StartCoroutine(ExecuteGameOverAfterDelay(0.25f));
    }
}
```

Рисунок 3.13 – Метод зіткнення

Метод OnTriggerEnter2D в грі "TapCube" викликається автоматично, коли об'єкт гравця зіткнеться з коллайдером іншого об'єкту, який має компонент Collider2D та Rigidbody2D. У цьому випадку метод викликається при зіткненні гравця з одним з ворожих об'єктів (enemy1 або enemy2). При зіткненні гравець програє і гра завершується.

Для коректного зіткнення було реалізовано в Unity (Rigidbody 2D), який забезпечує об'єктам у грі взаємодіяти з іншими об'єктами та середовищем, дотримуючись фізичних законів, таких як закони Ньютона та (Polygon Collider 2D), що дозволяє визначити точний обсяг об'єкта для виявлення зіткнень з іншими об'єктами у сцені гри. Зображено на рисунку 3.14 – 3.15.

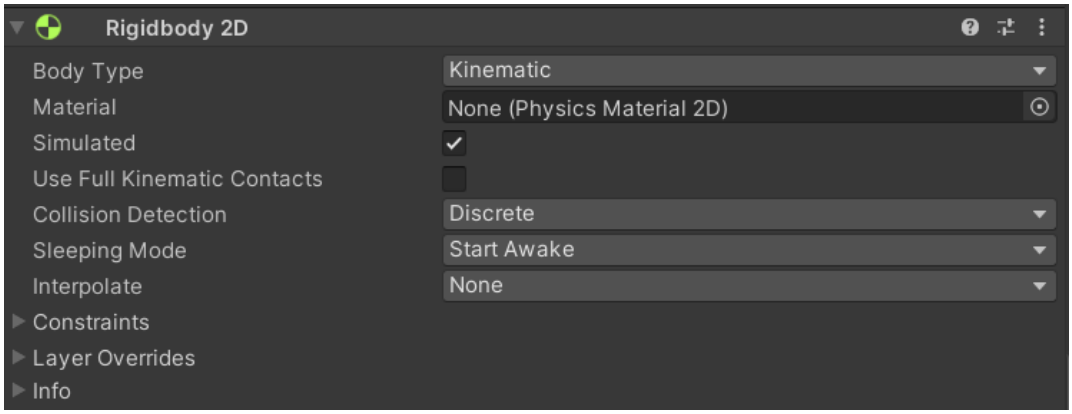

Рисунок 3.14 – Rigidbody 2D

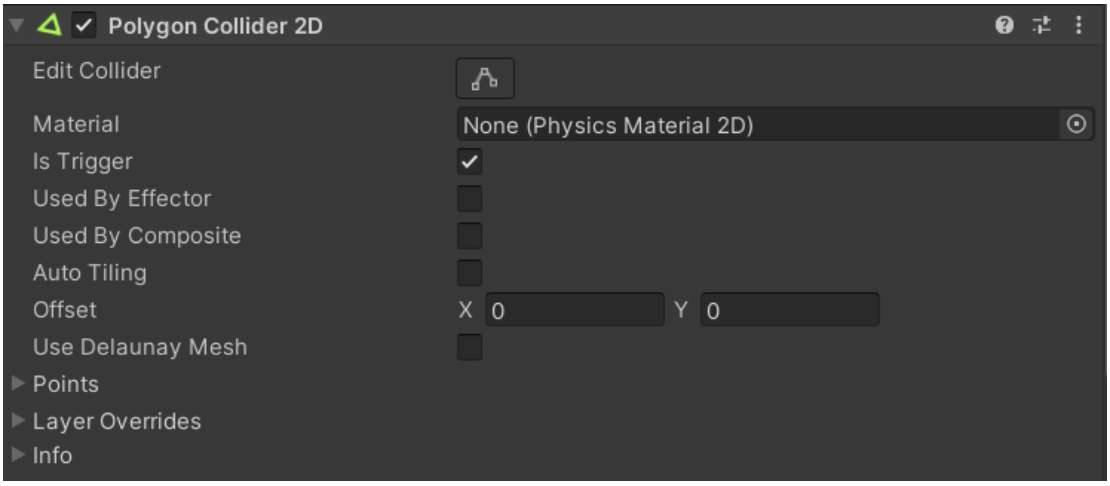

Рисунок 3.15 – Polygon Collider 2D

3. Старт гри: Запускає всі елементи. Лістинг зображено на рисунку 3.16.

```
public void StartGame()
{
    startScreen.SetActive(false);
    level.SetActive(true);
    gameOver = false;
    totalTouches = 0;
    UpdateScoreText();
    SetRandomPoint1();
    SetRandomPoint2();
    if (enemy1MovementCoroutine != null)
        StopCoroutine(enemy1MovementCoroutine);
    if (enemy2MovementCoroutine != null)
        StopCoroutine(enemy2MovementCoroutine);
    enemy1MovementCoroutine = StartCoroutine(MoveEnemy(enemy1));
    enemy2MovementCoroutine = StartCoroutine(MoveEnemy(enemy2));
}
```

Рисунок 3.16 – Метод старту гри

Цей метод ініціалізує гру, коли гравець натискає кнопку "Play". Він приховує стартовий екран (startScreen) і показує основний рівень гри (level). Збираємося знову встановити початкові значення, включаючи скидання лічильника очок (totalTouches), оновлення тексту рахунку і випадкове розміщення точок збору. Також метод зупиняє всі активні корутини для руху ворожих об'єктів (якщо такі є) і створює нові корутини для їх руху.

4. Завершення гри: Після зіткнення гравця з ворогом запускається панель завершення гри. Лістинг зображено на рисунку 3.17.

```
void GameOver()
{
    gameOver = true;
    StopCoroutine(enemy1MovementCoroutine);
    StopCoroutine(enemy2MovementCoroutine);
    gameOverScreen.SetActive(true);
    level.SetActive(false);
    if (PlayerPrefs.GetString("Language") == "Ukrainian")
    {
        finalText.text = finalTextUkrainian + totalTouches.ToString();
    }
    else
    {
        finalText.text = finalTextEnglish + totalTouches.ToString();
    }
    restartButton.gameObject.SetActive(true);
}
```

Рисунок 3.17 – Метод завершення гри

Цей метод викликається при завершенні гри. Він встановлює (gameOver) в (true), зупиняє корутини, що відповідають за рух ворожих об'єктів, відображає екран закінчення гри (gameOverScreen), приховує основний рівень гри (level). Також він оновлює текст завершального рахунку залежно від обраної мови гри (англійська або українська) і активує кнопку "Restart", щоб гравець міг розпочати гру знову та кнопку "Menu", щоб гравець міг вийти в головне меню. Кнопки реалізована через (OnClick) в Unity зображено на рисунку 3.18.

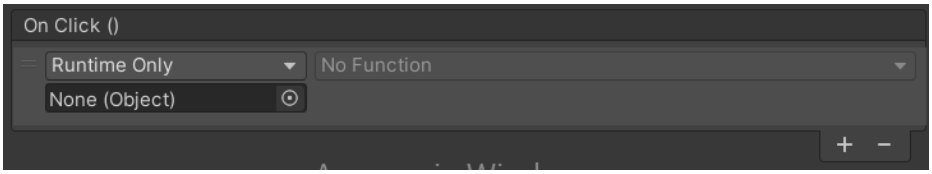

Рисунок 3.18 – OnClick

5. Рестарт гри: Викликається для бистрого початку нової гри. Лістинг Зображено на рисунку 3.19.

```
public void RestartGame()
{
    restartButton.gameObject.SetActive(false);
    gameOverScreen.SetActive(false);
    checkPoint1 = true;
    checkPoint2 = false;
    totalTouches = 0;
    UpdateScoreText();
    SetRandomEnemyPosition(enemy1);
    SetRandomEnemyPosition(enemy2);
    StartGame();
}
```

Рисунок 3.19 – Метод рестарту

Метод RestartGame забезпечує плавний перехід від екрану закінчення гри до нового сеансу гри, дозволяючи гравцеві швидко розпочати новий раунд після програшу або закінчення попередньої гри.

6. Генерація точок: Випадкове розміщення точок на екрані, до яких куб гравця повинен наближатися для їх збору. Лістинг зображено на рисунку 3.20.

```
void SetRandomPoint1()
{
    float randomX = Random.Range(-1.8f, 1.8f);
    float yPosition = point1.transform.position.y;
    point1.transform.position = new Vector2(randomX, yPosition);
}

void SetRandomPoint2()
{
    float randomX = Random.Range(-1.8f, 1.8f);
    float yPosition = point2.transform.position.y;
    point2.transform.position = new Vector2(randomX, yPosition);
}
```

Рисунок 3.20 – Метод випадкового розміщення

Методи SetRandomPoint1 та SetRandomPoint2 забезпечують генерування випадкових координат X в межах від -1.8 до 1.8. Це забезпечує випадкове розташування точки по горизонталі на екрані. Зберігаються поточні значення координати Y для точки point1 та point2. Це дозволяє зберегти вертикальне положення точки на екрані без змін, (new Vector2(randomX, yPosition)) створює

новий вектор позиції для точок point1, point2 з випадковою координатою X і збереженою координатою Y.

7. Налаштування звуку. Лістинг зображено на русинку 3.21.

```
void AdjustVolume(float volume)
{
    audioSource.volume = volume;
    PlayerPrefs.SetFloat("Volume", volume);
}
```

Рисунок 3.21 – Метод звуку

Цей метод викликається при зміні значення слайдера гучності. Він встановлює гучність відтворення звуку (audioSource.volume) на вибране значення і зберігає це значення в налаштуваннях гравця (PlayerPrefs). Також він оновлює значення слайдера гучності в інтерфейсі користувача для відображення актуального рівня гучності.

Додавання звуків та реалізація зміни гучності звуків в Unity зображена на рисунках 3.22 – 3.24.

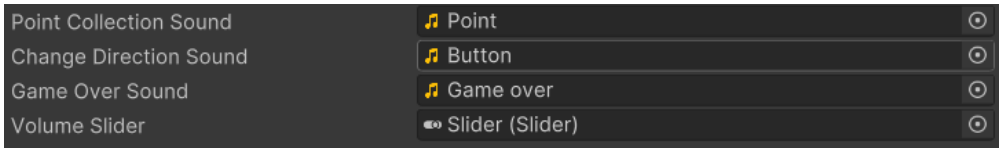

Рисунок 3.22 – Добавили звуки

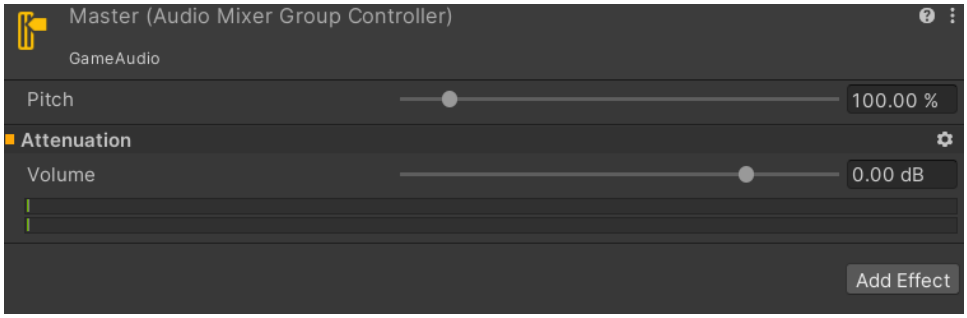

Рисунок 3.23 – Створили Mixer та добавили Volume

Mixer використовується для змішування звуків з різних джерел в єдиний аудіосигнал. Регулятор гучності (volume) в Unity дозволяє контролювати гучність кожного сигналу окремо, щоб збалансувати звуковий мікс.

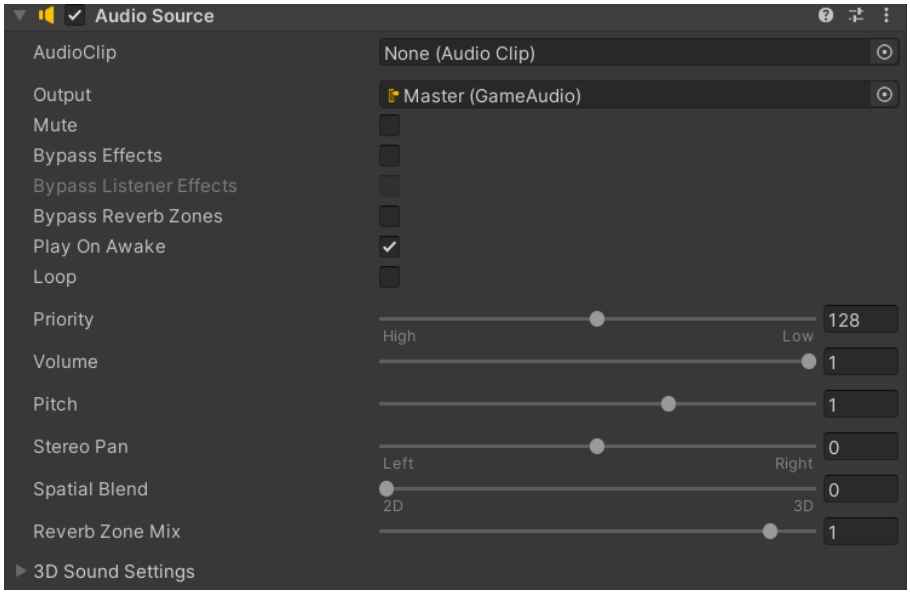

Рисунок 3.24 – Audio Source

Audio Source використовується для представлення звукових джерел у вашій грі чи додатку. Кожен Audio Source може містити аудіокліп — конкретний звуковий файл, який відтворюється під час гри. Це може бути музика, звуки ефектів чи голоси персонажів. Добавили Mixer в Output для коректного регулювання гучності в налаштуваннях.

8. Налаштування мови: Вибір мови українська або англійська. Лістинг зображено на рисунку 3.25.

```
public void SetUkrainianLanguage()
{
    PlayerPrefs.SetString("Language", "Ukrainian");
    UpdateScoreText();
}

public void SetEnglishLanguage()
{
    PlayerPrefs.SetString("Language", "English");
    UpdateScoreText();
}
```

Рисунок 3.25 – Методи мов

Ці методи встановлюють мову гри на українську та англійську. Вони зберігають вибрану мову у налаштуваннях гравця і оновлюють весь текстовий контент гри згідно обраної мови, включаючи текст рахунку і фінальний текст.

3.3 Тестування гри

1. Функціональне тестування являє собою процес перевірки програмного забезпечення, спрямований на підтвердження правильності його роботи згідно з визначеними функціональними вимогами та бізнес-завданнями. Це тестування забезпечує відповідність системи запланованим функціям і можливостям, а також її взаємодію з іншими компонентами [10].

Основна мета функціонального тестування полягає у перевірці коректного виконання функцій, обробки введених і виведених даних, а також відповідей системи на різні ситуації та умови. Цей вид тестування акцентує увагу на перевірці правильності роботи системи, не заглиблюючись у її внутрішню структуру.

Тест-кейс – це послідовність дій для перевірки, чи виконує система задані вимоги та функціонує правильно. Його мета – переконатися, що всі функції працюють як очікується, і що система відповідає стандартам та вимогам клієнтів. Тест-кейси складаються командою забезпечення якості та служать інструкціями для тестування. Збірка тест-кейсів називається тестовим набором [11].

Тест кейс 1: Початок гри

Опис:

Перевірити, чи гра починається правильно при натисканні кнопки "Почати".

Кроки для відтворення:

- Запустити гру.
- Натиснути кнопку "Почати".

Очікуваний результат:

- Стартовий екран ховається.

- Рівень гри активується.
- Гравець починає рух до точки (point1).
- Починається рух ворогів.

Тест кейс 2: Збирання очок

Опис:

Переконатися, що збирання очок працює правильно і рахунок оновлюється.

Кроки для відтворення:

- Почати гру.
- Дочекатися, поки гравець досягне точки (point1).
- Дочекатися, поки гравець досягне точки (point2).

Очікуваний результат:

- Гравець досягає точки (point1), грається звук, рахунок збільшується на 1, і гравець починає рух до точки point2.
- Гравець досягає точки (point2), грається звук, рахунок збільшується на 1, і гравець починає рух до точки point1.

Тест кейс 3: Зміна мови

Опис:

Перевірити, чи змінюється мова правильно при натисканні кнопки мови.

Кроки для відтворення:

- Запустити гру.
- Натиснути кнопку "Налаштування".
- Натиснути кнопку "Українська".
- Спостерігати за текстом.
- Натиснути кнопку "Англійська".
- Спостерігати за текстом.

Очікуваний результат:

- Коли обрана українська мова, текст рахунку змінюється.
- Коли обрана англійська мова, текст рахунку змінюється.

Тест кейс 4: Регулювання гучності

Опис:

Переконатися, що регулювання гучності працює правильно за допомогою слайдера гучності.

Кроки для відтворення:

- Запустити гру.
- Натиснути кнопку "Налаштування".
- Відрегулювати слайдер гучності.
- Спостерігати за рівнем гучності аудіо.

Очікуваний результат:

- Рівень гучності аудіо змінюється відповідно до значення слайдера.
- Налаштування гучності зберігається і залишається після перезапуску гри.

Тест кейс 5: Обробка закінчення гри

Опис:

Перевірити, чи відображається екран завершення гри, коли гравець стикається з ворогом.

Кроки для відтворення:

- Почати гру.
- Привести гравця до зіткнення з ворогом (enemy1 або enemy2).

Очікуваний результат:

- Відтворюється звук закінчення гри.
- Відображається екран завершення гри.
- Показується остаточний рахунок.
- Гра зупиняється, і рух ворогів зупиняється.

Ці тест кейси охоплюють основні функції гри та забезпечують роботу ключових можливостей відповідно до очікувань.

Відстеження багів, також відоме як відстеження дефектів або проблем, є важливою частиною процесу розробки програмного забезпечення. Цей процес включає виявлення, документування та вирішення проблем або помилок, що виникають під час розробки програм. Коли програма виконується неочікувано або не працює належним чином, це може свідчити про наявність багу. Такі помилки

можуть бути дрібними і впливати на функціональність лише частково, або мати критичний характер, що призводить до повного відмови програми або втрати даних [12].

Баг-репорт 1

Опис: Гра перезапускається при натисканні кнопки "Рестарт" під час гри.

Серйозність: S.2 – критична

Пріоритет: P.1 – високий пріоритет

Передумови: Гра вже розпочата.

Кроки для відтворення:

- Почати гру, натиснувши кнопку "Рестарт".
- Під час гри знову натиснути кнопку "Рестарт".

Очікуваний результат: Натискання кнопки "Старт" під час гри має бути або недоступним, або має перезапустити гру з правильною ініціалізацією.

Фактичний результат: Гра перезапускається без належної ініціалізації, що може спричинити некоректний стан гри.

Баг-репорт 2

Опис: Зміни гучності не застосовуються негайно після налаштування повзунка.

Серйозність: S.3 – помірна

Пріоритет: P.2 – середній пріоритет

Передумови: Гра розпочата, і повзунок гучності видно.

Кроки для відтворення:

- Відкрити меню налаштувань, натиснувши кнопку "Налаштування".
- Налаштувати повзунок гучності.

Очікуваний результат: Зміни гучності мають застосовуватися негайно під час налаштування повзунка.

Фактичний результат: Зміни гучності не застосовуються, поки меню налаштувань не буде закрито і знову відкрито.

Баг-репорт 3

Опис: Налаштування мови не зберігається після перезапуску гри.

Серйозність: S.3 – помірна

Пріоритет: P.2 – середній пріоритет

Передумови: Мова встановлена на українську або англійську.

Кроки для відтворення:

- Відкрити меню налаштувань і змінити мову на українську.
- Закрити гру та знову її відкрити.

Очікуваний результат: Гра має запам'ятовувати останнє налаштування мови і застосовувати його при запуску.

Фактичний результат: Гра за замовчуванням використовує англійську мову, незважаючи на попереднє налаштування мови.

Баг-репорт 4

Резюме: Гра не закінчується, якщо обидва вороги торкаються гравця одночасно.

Серйозність: S.2 – критична

Пріоритет: P.1 – високий пріоритет

Передумови: Обидва вороги знаходяться поруч з гравцем.

Кроки для відтворення:

- Розташувати обох ворогів близько до гравця.
- Дозволити обом ворогам одночасно торкнутися гравця.

Очікуваний результат: Гра має закінчитися негайно, якщо будь-який ворог торкається гравця.

Фактичний результат: Гра не закінчується, якщо обидва вороги одночасно торкаються гравця, дозволяючи грі продовжуватись у некоректному стані.

3.4 Висновки. Розділ 3

У даному розділі детально описано графічний дизайн та механізм створеної гри "TapCube". Графічний дизайн використовує двовимірну комп'ютерну графіку,

включаючи зображення головного персонажа, ворогів, точок для збирання, кнопок, тексту та фону, які створюються за допомогою растрових карт. Ці елементи значно підвищують візуальну привабливість гри і допомагають створювати складні сцени.

Механізм гри описує спосіб керування гравцем, який управляє кубом, збираючи точки для збільшення рахунку і уникаючи зіткнень з рухомими ворожими об'єктами. Ці механіки роблять гру доступною і простою для освоєння, що є характерною рисою казуальних ігор. Для реалізації функціоналу гри використовуються основні елементи коду, включаючи рух об'єктів, обробку зіткнень, управління звуком та інтерфейсом.

Також розглядає процес функціонального тестування гри, включаючи детальний опис тест-кейсів для перевірки основних функцій і можливостей програми. Аналіз баг-репортів, зокрема помилок ініціалізації гри, обробки звуку та зміни мови, підкреслює важливість відстеження і виправлення помилок для забезпечення оптимальної функціональності і стабільності програмного забезпечення.

Таким чином, описані аспекти графічного дизайну, механіки гри та процес тестування виявляють ключові складові успішного створення і підтримки ігрового продукту.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Працездатність людини – оператора.

Працездатність людини-оператора є ключовим аспектом у забезпеченні ефективності та безпеки в різних сферах діяльності. Вона визначається як здатність людини виконувати свої професійні обов'язки на належному рівні якості та продуктивності протягом визначеного часу. Працездатність включає фізичні, психічні та емоційні аспекти, які впливають на здатність оператора виконувати свої завдання безпечно та ефективно.

Фізичні фактори, які впливають на працездатність, включають стан здоров'я, рівень втоми, режим роботи та відпочинку. Зручність робочого місця, якість повітря, освітлення та температура також мають значний вплив. Якщо робоче місце не відповідає ергономічним стандартам, це може призвести до фізичного дискомфорту, швидкої втомлюваності та зниження продуктивності. Оптимальні умови освітлення та вентиляції важливі для підтримання належного рівня працездатності протягом усього робочого дня.

Психологічні фактори включають рівень стресу, мотивацію, особисті якості, досвід та навички. Високий рівень стресу може суттєво знижувати працездатність, викликаючи роздратування, зниження концентрації та ефективності. Мотивація є важливим чинником, що стимулює працівника до виконання своїх обов'язків з максимальною віддачею. Особисті якості та професійний досвід також відіграють важливу роль у здатності оператора справлятися з різноманітними завданнями. Розумова перевантаженість та монотонність праці можуть знижувати працездатність, призводячи до швидкої втоми та зниження концентрації [13].

Організаційні фактори стосуються організації робочого процесу, графіку роботи, тривалості змін та наявності перерв. Важливою є підтримка з боку керівництва та колективу. Грамотно організований робочий процес дозволяє уникнути надмірного навантаження на працівників, забезпечуючи рівномірний розподіл обов'язків та ефективну координацію роботи. Оптимальний графік

роботи, що включає достатню кількість перерв для відпочинку та відновлення, сприяє підтриманню високого рівня працездатності протягом усього робочого дня.

Методи підвищення працездатності включають оптимізацію робочих умов, вдосконалення ергономіки робочого місця та забезпечення комфортних умов праці. Раціоналізація режиму роботи передбачає запровадження оптимальних графіків, що враховують біологічні ритми людини, та перерв для відпочинку та відновлення. Психологічна підтримка включає проведення тренінгів для підвищення стресостійкості, розвиток командного духу та створення позитивного психологічного клімату в колективі. Медико-санітарні заходи включають регулярні медичні огляди, фізичну активність та правильне харчування. Регулярні медичні огляди дозволяють вчасно виявляти та лікувати захворювання, що можуть впливати на працездатність. Фізична активність сприяє підтриманню фізичної форми та здоров'я, а правильне харчування забезпечує організм необхідними речовинами для ефективної роботи.

Забезпечення високого рівня працездатності людини-оператора є необхідним для підтримання продуктивності та безпеки праці. Інтеграція комплексного підходу, що враховує фізичні, психологічні та організаційні аспекти, дозволяє створити умови для ефективної діяльності оператора.Це включає не лише забезпечення комфортних фізичних умов праці, але й створення сприятливого психологічного клімату, що підтримує мотивацію та знижує рівень стресу. Впровадження раціонального режиму роботи та перерв, а також медико-санітарні заходи сприяють підтриманню високого рівня працездатності протягом усього робочого дня [14].

4.2 Заходи, що покращують умови праці оператора.

Покращення умов праці оператора є важливим завданням для підвищення ефективності та безпеки виробничого процесу. Це питання висвітлюється з комплексним підходом, що включає ергономічні, організаційні, психологічні та медико-санітарні заходи.

Ергономічні заходи передбачають оптимізацію робочого місця. Важливо забезпечити достатнє місце для розміщення обладнання та інструментів, а також правильно організувати робочі зони. Використання ергономічних стільців та столів допомагає підтримувати правильну поставу та знижує навантаження на опорно-рухову систему. Встановлення якісного освітлення запобігає втомі очей і зменшує ризик виникнення помилок через погану видимість. Контроль мікроклімату, зокрема забезпечення належного повітрообміну та підтримання комфортної температури на робочому місці, а також підтримання оптимального рівня вологості, є критично важливим для здоров'я оператора.

Організаційні заходи включають раціоналізацію режиму роботи. Встановлення оптимальних графіків, які враховують біологічні ритми людини та забезпечують достатній час для відпочинку, є важливим. Регулярні короткі перерви сприяють відновленню сил та зниженню втоми, а організація активного відпочинку під час робочих змін підвищує загальну продуктивність. Чергування завдань допомагає запобігти монотонності та психічній перевантаженості, а впровадження командних методів роботи дозволяє розподілити навантаження та забезпечити взаємодопомогу серед співробітників.

Психологічна підтримка є ще одним важливим аспектом покращення умов праці. Проведення тренінгів для розвитку стресостійкості, підвищення кваліфікації та професійних навичок допомагає операторам справлятися з викликами на робочому місці. Надання можливості отримання психологічної допомоги сприяє подоланню стресу та інших психологічних проблем. Впровадження системи матеріального та нематеріального заохочення за досягнення у роботі підвищує

мотивацію операторів. Створення умов, за яких оператори можуть брати участь у прийнятті важливих рішень, підвищує їх відповідальність за результати праці.

Комплексний підхід до покращення умов праці оператора, що включає ергономічні, організаційні, психологічні та медико-санітарні заходи, дозволяє значно підвищити працездатність та безпеку праці. Це сприяє створенню ефективного та безпечного робочого середовища, що забезпечує високий рівень продуктивності та задоволення від роботи. Заходи, що покращують умови праці оператора [14].

Для забезпечення здоров'я працівників, які працюють з відеодисплейними терміналами (ВДТ), електронно-обчислювальними машинами (ЕОМ) та персональними електронно-обчислювальними машинами (ПЕОМ), а також для запобігання професійним захворюванням та підтримки працездатності, необхідно враховувати наступні заходи:

1. Внутрішньо змінні регламентовані перерви для відпочинку:
   - При організації праці з використанням ВДТ, ЕОМ і ПЕОМ слід передбачити регламентовані перерви для відпочинку.
   - Внутрішньо змінні режими праці і відпочинку мають передбачати додаткові нетривалі перерви до появи ознак втомлення і зниження працездатності.
   - Для розробників програм з використанням ЕОМ слід призначати перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за ВДТ.
   - Для операторів ЕОМ слід призначати перерви для відпочинку тривалістю 15 хвилин через кожні дві години.
   - Для операторів комп'ютерного набору слід призначати перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за ВДТ.
   - Якщо виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 години.

2. Режими праці і відпочинку при 12-годинній робочій зміні:

- Перерви в перші 8 годин роботи встановлюються аналогічно перервам при 8-годинній робочій зміні.
- Протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, регламентовані перерви тривалістю 15 хвилин мають відбуватися через кожну годину.

3. Зміна змісту роботи для зменшення монотонності:
   - Чергування операцій усвідомленого тексту і числових даних.
   - Чергування вводу даних та редагування текстів.
4. Комплекс фізичних вправ для активного відпочинку:
   - Під час деяких перерв доцільно виконувати комплекс вправ, спрямованих на зняття нервового напруження, м'язове розслаблення, відновлення функцій фізіологічних систем, зняття втоми очей, поліпшення мозкового кровообігу і працездатності.
5. Психологічне розвантаження:
   - За умови високого рівня напруженості робіт з ВДТ показане психологічне розвантаження у спеціально обладнаних приміщеннях (кімнатах психологічного розвантаження) під час регламентованих перерв або в кінці робочого дня.
6. Індивідуальний підхід при хронічних скаргах:
   - При хронічних скаргах на зорове втомлення, допускається індивідуальний підхід до обмеження часу робіт з ВДТ, зміни характеру праці, чергування з іншими видами діяльності, не пов'язаними з ВДТ.
7. Врахування характеру трудової діяльності:
   - Розробка внутрішньо змінних режимів праці і відпочинку здійснюється з урахуванням характеру трудової діяльності, напруженості і важкості праці для кожної професії.
   - Встановлення внутрішньо змінних режимів праці і відпочинку відповідно до професійних груп (розробники програм, оператори електронно-обчислювальних машин, оператори комп'ютерного набору).

8. Законодавчі норми та трудові норми:

- Тривалість обідньої перерви визначається чинним законодавством про працю і Правилами внутрішнього трудового розпорядку підприємства.
- Перерви для відпочинку і особистих потреб (згідно з трудовими нормами).

Запровадження вищезазначених заходів сприятиме збереженню здоров'я працівників, покращенню їх працездатності та запобіганню професійним захворюванням, пов'язаним з роботою з ВДТ, ЕОМ та ПЕОМ [16].

ВИСНОВКИ

У результаті детального аналізу всіх аспектів розробки проєкту "TapCube" можна зробити наступні висновки:

Процес розробки казуальної гри вимагає використання простих і захоплюючих геймплеєвих механік, а також мінімалістичного дизайну для забезпечення найкращого користувацького досвіду. Використання методології Scrum дозволяє ефективно організовувати процес розробки, а також швидко адаптуватися до змін і вдосконалювати продукт відповідно до потреб користувачів.

Проєктування гри "TapCube" включало в себе чітко визначені завдання, які були успішно втілені через детальні діаграми варіантів використання та діаграми класів. Це забезпечило зручність у розробці і відповідність реалізації очікуванням користувачів.

Графічний дизайн гри використовував двовимірну комп'ютерну графіку для створення елементів, що підвищили її візуальну привабливість і сприяли унікальному користувацькому досвіду. Механіка гри, також як і функціональне тестування, підтвердили важливість якості і стабільності програмного забезпечення для досягнення успішного релізу.

У підсумку, успішна розробка "TapCube" підкреслює важливість комплексного підходу до розробки ігор у казуальному сегменті, що включає ретельне проектування, використання ефективних методологій і високу якість реалізації. Тільки такий підхід може забезпечити створення продукту, який задовольняє вимоги і очікування користувачів, забезпечуючи позитивний користувацький досвід.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Платформа Unity [Електронний ресурс] – Режим доступу до ресурсу: https://unity.com/.

2. Гра Knife Hit [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.ketchapp.knifehit&hl=en&pli=1.

3. Гра Stack [Електронний ресурс] – Режим доступу до ресурсу: https://play.gohttps://play.google.com/store/apps/details?id=com.ketchapp.stackogle.com/store/apps/details?id=com.ketchapp.knifehit&hl=en&pli=1.

4. Методологія Scrum [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/agile/scrum.

5. Діаграма варіантів використання [Електронний ресурс] – Режим доступу до ресурсу: https://www.lucidchart.com/pages/uml-use-case-diagram.

6. Діагарама класів [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/.

7. Дизайн [Електронний ресурс] – Режим доступу до ресурсу: https://www.strate.education/gallery/news/design-definition.

8. 2D computer graphics [Електронний ресурс] – Режим доступу до ресурсу: https://graphics.fandom.com/wiki/2D_computer_graphics.

9. Казуальні ігри [Електронний ресурс] – Режим доступу до ресурсу: https://aaagameartstudio.com/blog/casual-games-in-2d-style-as-a-part-of-mobile-art/.

10. Функціональне тестування [Електронний ресурс] – Режим доступу до ресурсу: https://www.browserstack.com/guide/functional-testing.

11. Тест-кейс [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/searchsoftwarequality/definition/test-case.

12. Відслідкування багів [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/topics/bug-tracking.

13. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ЦИВІЛЬНИЙ ЗАХИСТ / О. Г.Левченко, О. В. Землянська, Н. А. Праховнік, В. В. Зацарний. – Київ, 2019. – 267 с. – (КПІ ім. Ігоря Сікорського).

14. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ / Т. Є.Стиценко, Г. В. Пронюк, Н. М. Сердюк, І. І. Хондак. – Харків, 2018. – 336 с. – (ХНУР).

15. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОХОРОНА ПРАЦІ / Я. О. Сєріков, Л. Ф. Коженевські, М. В. Хворост. – Харків, 2021. – 255 с. – (ХНУМГ ім. О. М. Бектова).

16. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, ДСанПІН 3.3.2.007-98, №7 від 10.12.98 [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98/print.

ДОДАТКИ

ДОДАТОК А – Програмний код

```
using System.Collections;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class GameManager : MonoBehaviour
{
public GameObject startScreen;
public GameObject gameOverScreen;
public GameObject settingsScreen;
public GameObject level;
public Player player;
public Enemy enemy1;
public Enemy enemy2;

public AudioClip pointCollectionSound;
public AudioClip changeDirectionSound;
public AudioClip gameOverSound;
private AudioSource audioSource;

public TextManager textManager;
public ButtonManager buttonManager;
public VolumeManager volumeManager;

void Start()
```

```
{
    audioSource = GetComponent<AudioSource>();

    buttonManager.InitButtons(this);
    volumeManager.InitVolume(this);
    volumeManager.LoadVolume();
    textManager.SetEnglishLanguage();
}

public void StartGame()
{
    startScreen.SetActive(false);
    level.SetActive(true);
    gameOverScreen.SetActive(false);

    player.InitPlayer(this);
    enemy1.InitEnemy(this);
    enemy2.InitEnemy(this);

    player.StartGame();
    enemy1.StartMoving();
    enemy2.StartMoving();
}

public void GameOver()
{
    gameOverScreen.SetActive(true);
```

```
    level.SetActive(false);

    player.StopMoving();
    enemy1.StopMoving();
    enemy2.StopMoving();

    textManager.UpdateFinalScore(player.GetTotalTouches());
}

public void RestartGame()
{
    gameOverScreen.SetActive(false);
    player.ResetPlayer();
    enemy1.ResetEnemy();
    enemy2.ResetEnemy();
    StartGame();
}

public void ReturnToMainMenu()
{
    startScreen.SetActive(true);
    gameOverScreen.SetActive(false);
    level.SetActive(false);

    player.ResetPlayer();
    enemy1.ResetEnemy();
    enemy2.ResetEnemy();
```

```
}

public void OpenSettings()
{
    settingsScreen.SetActive(true);
}

public void CloseSettings()
{
    settingsScreen.SetActive(false);
}

public void ExitGame()
{
    Application.Quit();
}

public void PlaySound(AudioClip clip)
{
    audioSource.PlayOneShot(clip);
}
}

public class Player : MonoBehaviour
{
public GameObject point1;
public GameObject point2;
```

```
private bool checkPoint1 = true;

private bool checkPoint2 = false;

private int totalTouches = 0;


void Update()

{

    MovePlayer();

}


void MovePlayer()

{

    if (checkPoint1)

    {

        transform.position = Vector2.MoveTowards(transform.position,
point1.transform.position, 2f * Time.deltaTime);

        if (Vector2.Distance(transform.position, point1.transform.position)
< 0.1f)

        {

            totalTouches++;


GameManager.Instance.PlaySound(GameManager.Instance.pointCollectionSound);

            checkPoint1 = false;

            checkPoint2 = true;

            SetRandomPoint1();

        }

    }

    else if (checkPoint2)

    {
```

```
        transform.position = Vector2.MoveTowards(transform.position,
point2.transform.position, 2f * Time.deltaTime);
        if (Vector2.Distance(transform.position, point2.transform.position)
< 0.1f)
        {
            totalTouches++;

GameManager.Instance.PlaySound(GameManager.Instance.pointCollectionSound);
            checkPoint1 = true;
            checkPoint2 = false;
            SetRandomPoint2();
        }
    }
}

void SetRandomPoint1()
{
    float randomX = Random.Range(-1.8f, 1.8f);
    float yPosition = point1.transform.position.y;
    point1.transform.position = new Vector2(randomX, yPosition);
}

void SetRandomPoint2()
{
    float randomX = Random.Range(-1.8f, 1.8f);
    float yPosition = point2.transform.position.y;
    point2.transform.position = new Vector2(randomX, yPosition);
}
```

```
public void StartGame()
{
    totalTouches = 0;
    checkPoint1 = true;
    checkPoint2 = false;
    SetRandomPoint1();
    SetRandomPoint2();
}

public void ResetPlayer()
{
    totalTouches = 0;
    checkPoint1 = true;
    checkPoint2 = false;
    SetRandomPoint1();
    SetRandomPoint2();
}
public int GetTotalTouches()
{
    return totalTouches;
}
public void StopMoving()
{
    enabled = false;
}
public void InitPlayer(GameManager manager)
```

```
{
    GameManager.Instance = manager;
}
}
public class Enemy : MonoBehaviour
{
public GameObject spawnPoint1;
public GameObject spawnPoint2;
public GameObject wall;
private Coroutine movementCoroutine;
public void InitEnemy(GameManager manager)
{
    GameManager.Instance = manager;
}
public void StartMoving()
{
    movementCoroutine = StartCoroutine(MoveEnemy());
}
public void ResetEnemy()
{
    if (movementCoroutine != null)
        GameManager.Instance.StopCoroutine(movementCoroutine);
    SetRandomEnemyPosition();
}
public void StopMoving()
{
    if (movementCoroutine != null)
```

```
        GameManager.Instance.StopCoroutine(movementCoroutine);
}
IEnumerator MoveEnemy()
{
    while (true)
    {
        while (transform.position.x <= wall.transform.position.x)
        {
            transform.Translate(Vector2.right * 2f * Time.deltaTime);
            yield return null;
        }
        SetRandomEnemyPosition();
    }
}
void SetRandomEnemyPosition()
{
    float randomX = Random.Range(spawnPoint1.transform.position.x,
spawnPoint2.transform.position.x);
    float randomY = Random.Range(spawnPoint1.transform.position.y,
spawnPoint2.transform.position.y);
    transform.position = new Vector2(randomX, randomY);
}
}
```

ДОДАТОК Б – Диск з роботою