Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційних радіотехнологій та технічного захисту інформації
(повна назва)
Кафедра медіаінженерії та інформаційних радіоелектронних систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
(позначення документа)
Створення мобільного ігрового додатку
(тема)

Виконав:
студент 2 курсу, групи СТМм-22-1
Анна ЧАСОВСЬКА
(прізвище, ініціали)

Спеціальність 171 Електроніка
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи, технології і комп'ютерні засоби мультимедіа
(повна назва освітньої програми)

Керівник: ст. викладач Костянтин КОЛІСНИК
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри Володимир КАРТАШОВ
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій та технічного захисту інформації
Кафедра Медіаінженерії та інформаційних радіоелектронних систем

Рівень вищої освіти другий (магістерський)
Спеціальність 171 Електроніка
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма " Системи, технології і комп'ютерні засоби мультимедіа"

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові Часовській Анні Олександрівні
(прізвище, ім'я, по батькові)
1. Тема роботи Створення мобільного ігрового додатку
затверджена наказом по університету від *" 20 " листопада 2023 р. № 1371СТ*
2. Термін здачі студентом закінченої роботи 08.01.2024
3. Вихідні дані до проекту (роботи)
1.Проаналізувати методики створення мобільних ігрових додатків
2.Розробити план по проектуванню та дизайну мобільного ігрового додатку
3.Розробити мобільний ігровий додаток та провести його тестування
4. Перелік питань, що потрібно опрацювати в роботі
ВСТУП
1. ОГЛЯД МОБІЛЬНИХ ІГРОВИХ ДОДАТКІВ
2. МЕТОДОЛОГІЯ СТВОРЕННЯ МОБІЛЬНИХ ІГРОВИХ ДОДАТКІВ
3. РОЗРОБКА МОБІЛЬНОГО ІГРОВОГО ДОДАТКУ
4. ТЕСТУВАННЯ ТА НАЛАШТУВАННЯ
5. ОЦІНКА ДОСВІДУ КОРИСТУВАЧА
ВИСНОВКИ
ПЕРЕЛІК ПОСИЛАНЬ
ДОДАТКИ
5. Перелік графічного матеріалу із зазначенням обов'язкових креслеників, схем, плакатів, комп'ютерних ілюстрацій: 1. Назва і тема роботи; 2. Постановка задачі; 3. Структура мобільних ігрових додатків; 4. Вибір технології та розробки; 5. План проектування та дизайну мобільного ігрового додатку 6. Розробка мобільного ігрового додатку; 7. Тестування та відладка; 8. Оцінка досвіду користувача; 9. Висновки.
6. Дата видачі завдання: 20.11.2023

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд мобільних ігрових додатків	20.11.23–30.11.23	
2	Методологія створення мобільних ігрових додатків	01.12.23-10.12.23	
3	Розробка мобільного ігрового додатку	11.12.23-15.12.23	
4	Тестування та відладка	16.12.23-20.12.23	
5	Оцінка досвіду користувача	21.12.23-25.12.23	
6	Графічна частина проекту	26.12.23-31.12.23	
7	Перевірка керівником проекту	01.01.24-03.01.24	
8	Перевірка нормоконтролем	04.01.24-06.01.24	
9	Перевірка зав. кафедрою, рецензування	07.01.24–08.01.24	

Студент _____ Анна ЧАСОВСЬКА
(підпис)

Керівник роботи _____ Костянтин КОЛІСНИК
(підпис) прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи має: 99 с., 60 рис., 7 табл., 2 додатка, 26 джерел.

МОБІЛЬНІ ІГРОВІ ДОДАТКИ, МЕХАНІКИ, РОЗРОБКА, АНАЛІЗ, ДІАГРАМИ, АНІМАЦАЯ, ЗВУК, UNITY, ТЕСТУВАННЯ, ВІДЛАДКА, ОЦІНКА.

Об'єкт дослідження – мобільні ігрові додатки.

Предмет дослідження – технології розробки мобільних ігрових додатків, що застосовуються у сучасному виробництві мобільних ігор.

Мета кваліфікаційної роботи – проаналізувати структуру мобільних ігрових додатків, провести проектування та дизайн мобільного ігрового додатку, розробити механіки та анімацію мобільного ігрового додатку та провести тестування та відладку.

Методи дослідження – теоретичний аналіз, дослідження програмних засобів, аналіз структури мобільних ігрових додатків, створення діаграм на основі аналізу.

Мета цієї роботи полягає у розробці гри для ОС Android на платформі Unity. Для вирішення поставленої задачі був проаналізований поточний стан ринку мобільних ігрових додатків, визначені основні принципи та концепції дизайну ігор, спроектована та розроблена архітектура ігрового додатоку, виконано тестування розробленого мобільного ігрового додатку.

ABSTRACT

The explanatory note of the qualification work has: 99 pages, 60 figures, 7 tables, 2 appendixs, 26 sources.

MOBILE GAMES, MECHANICS, DEVELOPMENT, ANALYSIS, DIAGRAMS, ANIMATION, SOUND, UNITY, TESTING, DEBUGGING, EVALUATION.

The object of research is mobile game applications.

The subject of research is mobile game application development technologies used in the modern production of mobile games.

The purpose of the qualification work is to analyze the structure of mobile game applications, design and design a mobile game application, develop the mechanics and animations of a mobile game application, and conduct testing and debugging.

Research methods – theoretical analysis, software research, analysis of the structure of mobile game applications, creation of diagrams based on the analysis.

The purpose of this work is to develop a game for the Android OS on the Unity platform. To solve the task, the current state of the mobile game application market was analyzed, the main principles and concepts of game design were determined, the game application architecture was designed and developed, and the developed mobile game application was tested.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8
ВСТУП 9
1 ОГЛЯД МОБІЛЬНИХ ІГРОВИХ ДОДАТКІВ 11
1.1 Історія розвитку мобільних ігрових додатків 11
1.2 Аналіз поточного стану ринку мобільних ігор 15
1.3 Технології та платформи для розробки мобільних ігрових додатків 18
1.3.1 Unity 18
1.3.2 Unreal Engine 20
1.3.3 Solar2D (Corona SDK) 21
1.3.4 AppGameKit 22
1.3.5 MonoGame 23
1.4 Основні принципи та концепції дизайну мобільних ігор 23
2 МЕТОДОЛОГІЯ РОЗРОБКИ МОБІЛЬНОГО ІГРОВОГО ДОДАТКУ 27
2.1 Вибір технології і розробки мобільного ігрового додатку 27
2.2 Визначення цільової аудиторії 30
2.3 Проектування та дизайн мобільного ігрового додатку 34
2.4 Розробка ігрового контенту 39
3 РОЗРОБКА МОБІЛЬНОГО ІГРОВОГО ДОДАТКУ 43
3.1 Створення ігрової механіки 43
3.1.1 Механіки гравця 43
3.1.2 Механіки платформ та пасток 47
3.1.3 Механіки ворогів 51
3.2 Графіка та анімація 55
3.2.1 Дизайн персонажей і рівнів 56
3.2.2 Анімація персонажей 61
3.2.3 Анімація платформ і пасток 65
3.2.4 Анімація ворогів 68
3.3 Звукове оформлення 70
3.4 Розробка інтерфейсу користувача 75

4 ТЕСТУВАННЯ ТА ВІДЛАДКА 78
4.1 Тестування функціональності мобільного ігрового додатку 78
4.2 Виправлення помилок і недоробок 80
5 ОЦІНКА ДОСВІДУ КОРИСТУВАЧА 82
5.1 Збір і аналіз відгуків користувачів 82
5.2 Оцінка задоволення гравців 83
ВИСНОВКИ 85
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 86
ДОДАТКИ 88

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – Операційна Система, базовий комплекс програм, що виконує керування апаратною складовою комп'ютера або віртуальної машини.

MOBA – Multiplayer Online Battle Arena – ігрова багатокористувацька онлайн арена для бою.

MMORPG – Massively Multiplayer Online Role-Playing Game – масова багатокористувацька онлайн рольова гра

AI – artificial intelligence – штучний інтелект.

RTS – Real-Time Strategy – стратегія в реальному часі

RPG – role-playing game – жанр комп'ютерних ігор.

SSAO – Screen Space Ambient Occlusion – техніка рендерингу в комп'ютерних графіках.

VR – virtual reality – віртуальна реальність.

AR – augmented reality – доповнена реальність.

GDD – Game Design Document – документ, що містить інформацію про проект в галузі геймдизайну.

UE – Unreal Engine – двигун для розробки відеоігор і симуляцій.

ВСТУП

Зараз у світі практично не залишилося людей, які не користувалися б мобільним пристроєм. У зв'язку з цим ринок мобільних ігрових додатків не стоїть на місці та активно розвивається. Щомісяця в магазинах App Store і Google Play з'являються нові ігри. Популярність мобільних ігрових додатків обумовлена рядом факторів:

– доступність: мобільні ігри доступні майже на всіх сучасних смартфонах, що дозволяє багатьом людям насолоджуватися ними, не купуючи додаткового обладнання;

– зручність: мобільні пристрої завжди під рукою, і гравці можуть грати в будь-який час і в будь-якому місці. Тому мобільні ігрові додатки – це чудовий спосіб згаяти час в очікуванні або під час невеликих перерв;

– безкоштовні ігри з мікротранзакціями: багато мобільних ігор можна завантажити безкоштовно, що знижує бар’єр доступу. Розробники ігрових додатків заробляють гроші за допомогою мікротранзакцій, які дозволяють гравцям купувати внутрішньоігрові предмети та оновлення;

– різноманітність жанрів: на мобільних платформах є ігри майже всіх жанрів, які приваблюють різну аудиторію. Головоломки, стратегії, аркади, гонки тощо;

– соціальні функції: багато мобільних ігрових додатків дозволяють спілкуватися з друзями та іншими гравцями.

У наш час створення мобільних ігрових додатків продовжує бути дуже актуальною темою. Оскільки мобільні пристрої, такі як смартфони та планшети, дуже поширені, ці платформи забезпечують доступ до широкої та різноманітної глобальної аудиторії. Таке охоплення дозволяє розробникам орієнтуватися на широке коло гравців та збільшують шанси вдалої монетизації.

Мобільні ігрові додатки пропонують різноманітні варіанти монетизації, зокрема покупки в програмі, рекламу та продаж ігор преміум-класу. Це може зробити розробку мобільних ігор прибутковим бізнесом для розробників.

Мобільні ігрові додатки також нерідко мають застосування у сфері освіти та навчання для отримання нових навичок і підвищення мотивації та залучення студентів. Крім того, розробка мобільних ігор дає розробникам можливість розвивати навички програмування, дизайну та управління проектами, які можуть бути корисними для кар'єрного зростання.

Мобільні платформи дозволяють розробникам експериментувати та розробляти нові ідеї, не витрачаючи значних грошей, що може призвести до нових інновацій в індустрії ігор. Тому створення мобільних ігрових додатків залишається актуальним і перспективним напрямком в ігровій індустрії, що пропонує багато можливостей для комерційного успіху і творчого розвитку.

Мета цієї роботи полягає у розробці гри для ОС Android на платформі Unity. Для вирішення поставленої задачі був проаналізований поточний стан ринку мобільних ігрових додатків, визначені основні принципи та концепції дизайну ігор, спроектована та розроблена архітектура ігрового додатоку, виконано тестування розробленого мобільного ігрового додатку.

1 ОГЛЯД МОБІЛЬНИХ ІГРОВИХ ДОДАТКІВ

Для визначення особливостей використання методик створення мобільних ігрових додатків з метою проведення аналізу щодо їх використання розглянемо історію їх виникнення [1].

1.1 Історія розвитку мобільних ігрових додатків

Першою грою в історії розробки мобільних ігрових додатків стала «Тетріс» 1994 року, яка була винайдена співробітником Обчислювального центру Академії наук СРСР Олексієм Пажитновим. Його захопила ідея схожої за змістом американської головоломки Pentomino Puzzle, але, навідміну від Тетрісу, в ній потрібно було зібрати одну велику фігуру з дрібніших об’єктів, що складаються з 5 блоків.

З технічних причин створити однакові фігури з п'яти компонентів не вдалося, тому ідея складалась з чотирьох блоків і семи варіантах фігур, які можуть падати та обертатись. Монітори комп’ютерів Обчислювального центру підтримували тільки текст, а не зображення, тому блоки фігур довелось скласти з дужок — [ ].

Популярність «Тетріса» росла в світі геймдеву та не могла залишитися без уваги гігантів індустрії. За можливість адаптації і випуску консольних та PC-версій гри стали боротися Andromeda Software, Mirrorsoft, SEGA, Nintendo та інші. У 2014 році показники завантажень мобільного ігрового додатку сягнули рекордних 425 мільйонів [2].

«Змійка» також з’явилася задовго до мобільних телефонів з іграми — в 1977 році. Суть полягала в тому, щоб керувати «змійками», змушуючи їх «з’їдати» цілі, що хаотично з’являються у різних точках ігрового поля. Крім того, у багатокористувацькій версії повинно було не дати супернику отримати більше очок.

Мобільна «Змійка» з'явилася на Nokia в 1997 році. Розробником став Танел Орманто, який неодноразово зізнавався, що ніколи раніше не створював

мобільних ігор. «Змійка» як мобільний ігровий додаток була багатокористувацькою з самого початку, у неї можна було грати разом через інфрачервоний порт.

Рух «змійки» відбувався в 4 напрямках, і Танел Орманто забезпечив невелику затримку перед тим, як зіткнутися зі стіною, роблячи гру повністю реалістичною на складних рівнях. Він пояснив, що якщо ніхто не проходить до фіналу, інтерес до гри поступово спадає.

Пізніше світ побачила «Змійка II», в якій зіткнення зі стіною вже не було фатальним. Друга версія розв'язала руки читерам — гравці швидко зрозуміли: якщо встигати поставити гру на паузу в той час, коли «змійка» їсть свою «їжу», її довжина не зміниться, і відповідно, грати буде простіше.

Популярність «Тетріс» і «Змійки» пояснити досить легко. Обидві гри зачепили геймерів своєю простою і зрозумілою механікою, доступністю (телефони продавалися вже зі встановленими іграми) і можливістю змагатися з друзями. Саме ці ігри навчили людей довго дивитися в екран телефонів і дозволили побачити в своїх мобільних не тільки апарати для зв'язку.

Згодом кожна нова модель телефону ставала все компактнішою, спритнішою і «розумнішою». З'явилася технологія бездротової передачі даних (WAP), і сторонні розробники, що не співпрацювали з виробниками телефонів безпосередньо, відчули свободу. Вони отримали реальну можливість пропонувати свої ігри користувачам, замість того, щоб чекати, поки виробники мобільних телефонів помітять їх творіння і погодяться на співпрацю.

У 2001-му з'явилася єдина стандартизована технологія зі специфікаціями для розробників — Java 2 Micro Edition. Цей формат створив єдині вимоги для всіх розробників та дозволяв створювати ігри для малопотужних мобільних телефонів і тим самим збільшувати аудиторію геймерів [3].

Індустрія розвивалася, вимоги гравців зростали, і щоб залучити до «мобільного» світу шанувальників PC, найпопулярніші десктопні ігри стали адаптуватися під проходження телефоном. Десь сильно спрощували сюжет, десь

страждав геймплей. Якісні мобільні версії вийшли для комп'ютерних Prince of Persia та Splinter Cell.

З виходом на ринок iPhone у багатьох студій розробки з'явилася можливість зробити щось нове. Сенсорний дисплей та наявність акселерометра допомогли розробникам реалізувати нові рішення в управлінні грою, зробити геймплей приємнішим. А поява у 2008 році App Store взагалі стала революцією у сфері геймдева. Розробники отримали готовий майданчик для просування своїх ігор та збору фідбеку від користувачів. Тепер заявити про себе могли не лише іменитi корпорації, а й дрібні інді-студії.

Велика популярність приходила до геймдев-студій, які розробляли ігри під їх проходження. Однією з таких ігор був культовий Doodle Jump.

Маленьку компанію Lima Sky заснували два брати родом із Хорватії, Ігор та Марко Пусенякі, у 2009 році. Їм подобалась філософія створення невеликої лінійки якісних продуктів, які можна розвивати роками, тому як тільки дізналися про заплановане відкриття App Store, вирішили створити свою програму та випустити її на яблучній платформі.

Беручися за створення Doodle Jump, брати не ставили за мету заробити на грі мільйони. Вони просто хотіли провести годину за улюбленою справою та подивитися, що вийде цього разу. І доки Марк опрацьовував двигун майбутньої гри, Ігор створював ескізи. Графіка все ніяк не виходила такою, якою її хотів бачити сам творець, тому до останнього жоден із братів не знали, як виглядатиме Doodle Jump.

Творці до останнього сумнівалися, чи зможе така, здавалося б, проста гра конкурувати з дорогими проектами великих корпорацій. У той час такі хіти, як Fruit Ninja та Chocolate Frenzy, захопили ринок завдяки вражаючій 3D-графіці та чудовій механіці.

Після того як Doodle Jump потрапив до списку рекомендованих мобільних ігрових додатків на App Store, стався перший стрибок популярності, після якого були спади і нові прориви. Щоб збільшити аудиторію користувачів, в 2010 році

розробники випустили Android і BlackBerry-адаптації своєї гри. Це рішення допомогло завоювати нових прихильників.

Брати продовжували активну роботу над грою: додавали нові локації, наприклад космос та пекло, випускали різдвяні оновлення, а у 2015 році випустили Doodle Jump DC. Історія Doodle Jump – це яскравий приклад того, як можна розвивати найпростішу гру.

Потрапити в Android Market зі своїм ігровим додатком завжди було простіше, ніж в App Store. Однак більший дохід можна отримати саме на «яблучній» платформі. Тільки за останній квартал 2017 року власники iOS-девайсів залишили в магазині App Store 11,5 млрд. доларів, тоді як в Google Play витратили вдвічі менше. Тому, щоб зачепити якомога більше гравців і збільшити свій дохід, розробники працюють над створенням ігрових додатків як для Android, так і для iOS.

Популярна у всьому світі Angry Birds була придумана трьома студентами, які перемогли у конкурсі на створення кращої гри від HP та Nokia на фестивалі розробників Assembly. Вони представили гру King of the Cabbage World, яка посіла перше місце. Їхній проект відразу ж викупила геймдев-компанія Sumea і перейменувала в Mole War, що стала першою багатокористувацької грою у режимі реального часу [4].

Після такої успішної колаборації розробники створили власну компанію Relude. Студія займалася розробкою мобільних ігрових додатків, серед яких — адаптації таких відомих PC-ігор, як Need for Speed і SWAT.

Після менш вдалого релізу карткової гри Pyramid Solitaire Saga у квітні 2012 року з'явилася культова Candy Crush Saga. У цій грі була логіка «3 в ряд», мультяшність, насичений колір та несподівана озвучка. Надихнувшись успіхом гри і бажаючи розвивати льодяниковий всесвіт, студія випустила спін-оффи Candy Crush — Candy Crush Soda Saga, Candy Crush Jelly Saga і Candy Crush Friends Saga. Але досі найбільш популярною і прибутковою залишається саме оригінальна версія.

1.2 Аналіз поточного стану ринку мобільних ігор

Ринок мобільних ігор неоднорідний через різний рівень життя в різних країнах, відсутність перекладів з англійської для країн, де розмовляють лише рідною мовою, культурні особливості та традиції, цензурні обмеження [5].

На 2022 рік компанія Google виділяє 5 регіонів мобільних ігор: Азія, Північна Америка, Європа, Південна Америка, Близький схід та Африка. Найбільш розвиненими можна вважати ринки Азії, Північної Америки та Європи. При цьому Азія є найбільшим за кількістю гравців.

Азія є центром розвитку мобільних проектів і ринком, на який орієнтуються компанії. За оцінками компанії Niko близько 31 мільярда доларів ринку відеоігор у Китаї приходяться саме на мобільні додатки.

Кожен гравець із Китаю став приносити в середньому на 10 доларів більше кожного місяця. Така динаміка робить цей ринок дуже привабливим для інвесторів.

Найбільшою особливістю ринку мобільних ігрових додатків у США є те, що він є центром, який виробляє різноманітні інтерактивні розваги. Проблема мобільних ігор у США полягає в тому, що конкуренція за увагу американських користувачів є дуже високою, оскільки всі можливі аудиторії вже грають.

Японія є світовим лідером у створенні контенту. Крім того, для пересічного японця мобільні ігри є оптимальним рішенням, враховуючи щільний робочий графік і високу інтенсивність життя. Очікується, що японський ринок мобільних ігрових додатків зростатиме в середньому на 10% на рік.

Корейський ринок дуже схожий на японський, але має менші перспективи через меншу кількість населення та відповідно розробників. Але оскільки Південна Корея є технологічним центром для виробників смартфонів, таких як Samsung, вона має перспективи забезпечити своїх громадян якомога більшою кількістю високоякісних телефонів і мобільного Інтернету.

Європа, включно з провідними країнами (Велика Британія, Франція та Німеччина), має щорічний дохід на суму близько 14,69 мільярдів доларів. Для

європейців поняття «купити гру» стало ближчим, оскільки додаткові грошові вливання не потрібні. Європейські геймери також консервативні, коли йдеться про жанри, включаючи спортивні симулятори, шутери від першої особи, стратегічні ігри та інші жанри, що створені для ПК і консолей. Вийти на цей ринок простіше, як і локалізувати його англійською або іншими мовами світу, але при великій кількості гравців заробляти гроші можуть тільки великі компанії.

Ринок ігор у Бразилії та Аргентині становить 2 млрд. доларів. Також прогнозується серйозне зростання споживання мобільних ігрових додатків через зростання числа смартфонів у населення та розвитку технологій мобільного інтернету.

Лідируючою операційною системою виявився Android, оскільки кількість завантажень набагато вища, ніж у аналогічних програм від Apple [6].

Найпопулярніші жанри мобільних ігор розподілені між:

- MOBA;
- MMORPG;
- королівська битва;
- карткові стратегічні ігри;
- головоломки, включаючи казуальні та гіперказуальні ігри.

Цей список базується на кількості людей, які завантажили гру, і на тому, скільки грошей було витрачено на ігри в кожному жанрі [7].

MOBA — це суміш стратегії в реальному часі (RTS) і дії. На багатокористувацьких бойових аренах гравець керує одним персонажем – «героєм» або «чемпіоном» із доступного списку, набираючи унікальний набір навичок і здібностей. Це може бути маг, воїн, лицар або представник іншої «професії». Мобільна версія жанру відрізняється від комп’ютерної (DOTA 2 або League of Legends), оскільки не відображає такої ж глибини механіки та стратегії. Мобільні пристрої обмежені в обчислювальній потужності, розмірі дисплея та вхідних параметрах.

Жанр MOBA на Android розвивається семимильними кроками. Компактна версія League of Legends і хардкорна Vainglory вже з'явилися на смартфонах і планшетах, а також проекти, стилізовані під всесвіти Marvel і DC.

MMORPG означає Massively Multiplayer Online Role Playing Game і є одним із найстаріших ігрових жанрів. Загалом, це рольова онлайн-гра, в якій гравець посилює силу героя та отримує нові здібності в боях проти неігрових персонажів або інших реальних гравців. MMORPG з часом розвивалися. Визначення змінилося, жанр ожив і більше не обмежується Warcraft.

MMORPG для мобільних пристроїв була менш популярною, ніж для ПК, тому що не кожен смартфон міг підтримувати ці ігри. Але з розвитком смартфонів цей жанр повертає свою популярність. Серед найпопулярніших мобільних MMORPG Dungeon Hunter 5, Arcane Legends і Adventure Quest.

Суть жанру Battle Royale полягає в тому, що користувач бореться проти 99% гравців на величезній карті з такими ресурсами, як зброя, озброєння та амуніція. Ресурси розкидані в різних місцях, і гравець повинен збирати їх для боротьби з ворогами. Головна мета мобільної гри Battle Royal — залишитися в живих. Ідея частково взята з антиутопічного японського фільму «Королівська битва», випущеного в 2000 році.

Карткові стратегічні ігри – категорія, відома як ігри з колекційними картками (TCG). Гравець повинен битися своєю колодою з колодою супротивника. Карти розділені за рівнями потужності. Карти вищого рівня можна заробляти під час гри або витрачаючи реальні гроші, щоб розблокувати нову колоду.

Найпопулярнішим представником цього жанру є Hearthstone, у нього понад 100 мільйонів гравців на ПК, iOS та Android. Інші популярні мобільні карткові ігри: Elder Scrolls: Legends, HEX: Shards of Fate, Eternal, Shadowverse, Magic: The Gathering і Pokémon Trading Card Game. Ігри цього жанру абсолютно безкоштовні.

Казуальні ігри – категорія, що домінує на ринку ігор для мобільних пристроїв. Вона має 58% рівня проникнення мобільних пристроїв, і є жанром ігор,

у який найчастіше грають користувачі Android. Казуальні ігри включають все: від таких ігор, як Candy Crush, до слотів і безкоштовного бінго.

Більше половини користувачів смартфонів мають на своїх пристроях хоча б одну гру-головоломку. Тетріс, Angry Birds, варіації «три в ряд» – навіть ті користувачі, які не вважають себе геймерами, встановлюють ці мобільні ігри на своїх смартфонах. Люди різного віку та соціальних груп люблять головоломки. Мета головоломок зрозуміла, але в той же час вона нелегко досяжна, для перемоги потрібно використовувати логіку і кмітливість.

Ігри-головоломки вважаються піджанром казуальних ігор, які охоплюють широку аудиторію, займають міцні позиції та приносять найбільший дохід на ринку мобільних ігор. Станом на липень 2022 року найпопулярнішими головоломками в усьому світі є Candy Crush Saga, Fill the Fridge, Fill Up the Fridge: Organizing Game, Shoot Bubble – Pop Bubbles і Fishdom.

Гіперказуальні ігри не вимагають жодних втручань від користувача, а спрощена ігрова механіка є лише плюсом. Вони зосереджені на мінімалістичних, але чітких зображеннях, мультяшному художньому стилі та елементарних системах прогресу. Приклади включають Dodgeball Duel, Terrarium і Hole.io.

1.3 Технології та платформи для розробки мобільних ігрових додатків

Вибір платформи для розробки мобільних ігрових додатків може мати величезний вплив на успіх. Слід враховувати такі фактори, як особливості платформи, зручність використання, вартість і підтримка спільноти [8].

1.3.1 Unity

Інтегроване середовище розробки, яке надає інструменти для створення мультиплатформних ігор. Він включає безліч функцій, встановлених моделей, текстур і документації.

Понад 50% всіх мобільних ігор розроблено саме на Unity. Це потужне, але в той самий час просте в роботі ПЗ, що дозволяє створювати і випускати 2D і 3D-

ігри. Розробка мобільних ігор на Unity відкриває перед розробниками безліч можливостей платформи для підтримки та монетизації створених ігор.

В Unity існує магазин готових асестів та плагінів. Це дозволяє розробляти проєкти швидше і з меншими витратами. У програмі є повноцінний графічний редактор, де можна розміщувати карти, локації та персонажів. До прийнятного вигляду їх доводять у Photoshop. Створюючи 3D-гру Unity, ви можете імпортувати 3D-моделі від більшості сторонніх видавців, що спрощує ваш робочий процес.

Переваги двигуна Unity:

– простий для розуміння редактор і набір інструментів: навіть новачки в розробці мобільних ігрових додатків можуть вивчити основи за кілька днів;

– сучасні графічні рівні можуть конкурувати з більш дорогими рушіями. Звісно, Unity відстає від UnrealEngine за функціональністю, але задовільний з точки зору стандартного набору ефектів постобробки, Deferred Lighting і прискореного SSAO (Screen Space Ambient Occlusion) з обробкою карти світла;

– ігровий рушій Unity умовно безкоштовний. Платити потрібно лише під час оновлення пакета передплати. Ліцензійні знижки діють кілька разів на рік, зазвичай -20%;

– велика спільнота розробників, багато випущених ігор;

– внутрішній магазин матеріалів, де ви можете придбати фрагменти коду, матеріали та попередньо визначені звуки. Можливість створювати реалістичні відео;

– розробка Unity дозволяє легко імпортувати на пристрої Windows, Linux, OS X, Android, iOS, PlayStation, Xbox, Nintendo, VR і AR.

Недоліки середовища розробки Unity:

– розробка гри на Unity вимагає навичок програмування;

– багато вбудованих компонентів роблять продукт громіздким. Це може бути проблематично, оскільки користувачі не люблять завантажувати великі ігри;

– розробники не мають доступу до вихідного коду гри. Навіть якщо ви купуєте платну ліцензію, вихідний код не надається;

– відсутня інтеграція із зовнішніми сервісами та бібліотеками (наприклад, Facebook), тому розробникам доводиться налаштовувати їх вручну;

– неможливо додати сторонню фізику або SpeedTree до двигуна.

1.3.2 Unreal Engine

Мобільні системи стали одним із найбільш пріоритетних напрямків GameDev на Unreal Engine, оскільки підтримка iOS та Android також гарантована. Можно досить швидко імпортувати гру з однієї платформи на іншу за допомогою модульної системи залежних компонентів.

Основна мета Unreal Engine — спростити розробку якісного проекту, в тому числі з ігровим і стабільним мультиплеєром. Численні ресурси з можливістю керувати не лише механікою гри, але й графікою є однією з головних особливостей UE.

Нюанси, що відрізняють цей двигун від інших:

– повний набір інструментів OOB. Все, що вам потрібно, це встановити середовище розробки та запустити його – Unreal Engine вже містить усі необхідні функції;

– робота з C++. Незважаючи на те, що освоїти її складніше, ніж C# або Python, ця мова програмування працює набагато швидше. Це покращує якість і продуктивність кінцевої версії проекту;

– створення візуального сценарію. Система Visual Scripting Blueprints дозволяє розробляти ігри, навіть не знаючи C++. Хоча все одно доведеться редагувати та налаштовувати код для кращих результатів, це значно збільшить швидкість створення базових об’єктів.

Розвитком Unreal Engine став двигун серії 5, який створив новий рівень фотореалізму, та отримав кілька особливих та помітних нововведень:

– Наніт. Ця назва представлена віртуалізованою геометрією з мікрополігонами, що усуває неприємну концепцію полігонального бюджету. Ця геометрія дозволяє показувати на екрані геометрію кінематографічної властивості, що включає мільйони та мільярди багатокутників;

– Люмен. Ще одна особливість для докладних зображень є динамічним глобальним освітленням. Завдяки цій системі Lumen зображення реагує на зміну сцени та умов освітлення, тобто кількість світла у грі змінюватиметься в реальному часі;

– симулювання звуку реверберації. Простіше кажучи, UE5 має реалістичний ефект відлуння, що з'являється в обмеженому просторі. Звук гасне поступово, система розраховує його залежно від параметрів намальованої області;

– поліпшений фізичний двигун. Закони природи працюють правильніше, моделі персонажів рухаються реалістичніше, а фізичні частинки ведуть так, як вони повинні у реальному світі;

– нове моделювання рідини. Система дозволяє створювати ефектні сцени з рідинами та підвищувати реалістичність зображення. Порівняно з попереднім поколінням UE різні рідини мають різні параметри, а атмосфера проектів набагато вражаюча.

Таким чином, можна отримати результат з відмінною графікою, реалістичною фізикою і стабільною продуктивністю.

1.3.3 Solar2D (Corona SDK)

Solar2D, офіційно відомий як Corona SDK, дозволяє розробникам ігор створювати 2D-мобільні програми. Це ігровий двигун на базі Lua з упором на простоту ітерацій та використання. Комплект розробника безкоштовний для використання та має відкритий вихідний код. Він також підтримує розробку ігор на кількох платформах.

Solar2D має активну спільноту користувачів та ринок, де доступні 2D-функції та плагіни. Він також включає Solar2D Playground, інтерактивний веб-сайт для миттєвого створення та запуску проектів в Інтернеті.

Ця платформа відома своєю швидкістю, що дозволяє розробникам швидко та ефективно створювати ігри. Corona SDK підтримує як розробку 2D, так і 3D-ігор і пропонує різні плагіни для додаткової функціональності. Вона також підтримує кілька платформ і надає простий процес написання коду, що робить її хорошим вибором для незалежних студій.

1.3.4 AppGameKit

AppGameKit – ще один двигун для розробки ігор, що не потребує спеціальних навичок програмування. Це мобільний кросплатформовий двигун, що дозволяє кодувати гру та розгортати її на кількох платформах. Як розробник розроблять гру, вона стає доступною всім пристроям.

Можна розробляти 2D ігри з деякими базовими елементами 3D, так як AppGameKit чудово впорається з цим завданням. Також можно знайти достатньо варіантів для розробки та розгортання своєї гри. Найкращий спосіб розробити мобільну гру – це використовувати AppGameKit Studio – універсальний робочий простір, в якому все необхідне для втілення ідеї від початкової концепції до готової до гри.

Підходить для незалежних студій розробки. Переваги AppGameKit:

- безліч готових рішень для зручної розробки;
- швидкий компілятор;
- крос платформи;
- вбудовані інструменти монетизації.

Має потужний набір програм для створення мобільних ігор для різних платформ.

1.3.5 MonoGame

Набір інструментів для створення ігор з архітектурою класів, аналогічною XNA 4.0. Він використовує C# і підтримує будь-яку NET-мову, тому, якщо ви вже знаєте ці мови, створення мобільної гри за допомогою MonoGame не видасться вам складним. Багато навчальних посібників було створено, щоб допомогти вам створити свій перший проект.

Найбільші переваги, які він дає, – це велика спільнота користувачів та її технологія з відкритим вихідним кодом. Він також має велику кількість підтримуваних платформ.

Платформа використовує шейдери, написані різними мовами. Тому MonoGame розробила власну мову для написання шейдерів-MGFX. Цей набір інструментів також підтримує файли .fx (формат редагування ефектів Microsoft).

1.4 Основні принципи та концепції дизайну мобільних ігор

Незважаючи на велику кількість жанрів, процес проектування мобільних ігрових додатків є однаковим і складається з наступних етапів: ідеї, розробки концепції, розробки доказу концепції, створення документа про ігровий дизайн (GDD), прототипу гри, архітектури дизайну, підтримки гри.

Виявлення ідеї є першим кроком у створенні проекту, на ньому будується концепція гри, необхідно визначити основні ідеї та цілі. Визначення цільової аудиторії є одним із найважливіших питань для успіху гри [9].

Для цього можна використати додаткові запитання, які пропонують розв’язати задачу:

– визначити цільову та вікову групу гравців;

– сформулювати основну ідею гри, яка приверне увагу гравця і спонукає його вибрати цю гру;

– формулювання ідей, які розробники хочуть донести до клієнтів через ігри;

– визначити, чим дана гра відрізняється від інших (дизайн, концепція) і що дивує гравців під час гри.

Концепція гри – це короткий опис основних моментів створення та оформлення проекту. Вона повинна містити загальну інформацію про гру, причину її створення та ідею, а також містить розбивку всіх можливих дій, доступних гравцеві. Розробка концепції включає ескізи, ігрову механіку, налаштування, технології, взаємодії тощо.

Скетчінг – це розробка правильного образу, який найкраще відображає ідею та настрій гри та особистість персонажа. Створюється кілька варіантів ескізу, з яких для кожного елемента проекту вибирається найбільш підходящий варіант. Важливим моментом, якого дизайнер повинен дотримуватись при цьому, є створення всіх елементів в одному стилі: всі значки, внутрішні блоки, персонажі та інше повинні бути схожі за кольоровою гамою, стилем та зовнішнім виглядом.

Ігрова механіка – це набір правил, які описують дії гравців для досягнення цілей гри. Наприклад, традиційна шахова механіка включає опис дошки, початкові позиції та список можливих ходів для кожної фігури. Звичайно, ігрова механіка також повинна включати умови перемоги [10].

Налаштування стосуються двох моментів: історії та естетики. Історія описує світ гри, події, що відбулися в минулому, і події, що відбулися під час гри. Естетика – це те, як гра виглядає і звучить. Разом вони надзвичайно важливі для досвіду користувачів. Деякі абстрактні ігри можуть не мати сюжету або використовувати добре відому передісторію, як-от фільм.

Технічні вимоги залежать від ігрового пристрою, а які інструменти використовуватиме дизайнер, залежатиме від того, що технічно можливо. Наприклад, мобільні ігрові додатки, призначені для потужних планшетів або смартфонів, можуть терпіти трохи більш вимогливий дизайн. Вибір технології полягає в тонкому балансі між простим у написанні кодом, який зручно підтримувати, і достатньою продуктивністю на цільових пристроях.

Взаємодія включає механізми налаштування взаємодії користувача з грою, використання розробником налаштувань пристрою та методів введення, а також використання простору на екрані. Це дуже важлива частина мобільних пристроїв, яка може мати різний зовнішній вигляд.

Розробка доказу концепції полягає в перевірці можливості реалізації критичних функцій гри, обраних методів керування і захопливого геймплею. Також важливо переконатися, що художники команди здатні створити дизайн і стиль, що буде приваблювати цільову аудиторію.

Документ ігрового дизайну – це дуже детально описане визначення самого проекту. GDD є «живим» документом і може змінюватися під час розробки та відгуків. Він повинен змінюватися в міру зміни вимог. GDD зазвичай створюються та редагуються разом з розробниками та дизайнерами та використовуються для організації командної діяльності. На відміну від концептуальних документів високого рівня, GDD містить основні деталі впровадження. Окрім опису гри, GDD має також описувати гравців. Ігри, орієнтовані на занадто широку аудиторію, часто не мають характерних особливостей і можуть здаватися занадто дрібними та недостатньо привабливими для всіх [11].

Створення прототипів передбачає те, що більшість механіки вже перевірено. Це дає змогу створити відтворюваний прототип для цільової платформи. Він має охоплювати більшість основних механізмів і виглядати як основна частина гри. Іншими словами, це спрощена версія програми для тестування. Весь процес створення та розробки гри займає багато часу, тому навіть невеликі помилки можуть означати багатогодинну роботу. Прототипування необхідно для виявлення та вирішення проблем на ранній стадії.

Багато ігрових функцій і сценаріїв знаходяться на ранніх стадіях розробки. У світі дуже мало ігор, які мають тип і розклад, описані в GDD. З’являються нові ідеї, змінюються самі технології, проект має постійно мінливий характер розвитку і вимагає дуже гнучкого архітектурного рішення, заснованого на модульному підході. Створення такого типу архітектурного дизайну може бути важким завданням, але це найважливіший крок у процесі розробки гри.

Для багатьох мобільних і веб-проектів запуск – це лише початок довгого та важкого шляху. Для постійного зростання бази користувачів і високого рівня утримання на передовій лінії онлайн-магазину оновлення ігор є дуже важливим і

постійним процесом. Аналіз популярних на даний момент ігрових додатків показує, що оновлення повинні виходити кожні 2-5 тижнів. Крім того, кожне оновлення має додавати до гри більше вмісту.

Вибір стилю оформлення гри залежить від таких факторів, як сприйняття аудиторії, «ігрова історія», жанр і технічні вимоги. Кожен жанр має свої стандарти оформлення та аудиторію. Найголовніше питання при роботі над цією статтею – «Яку аудиторію я хочу залучити?» Наступне важливе питання – це сумісність між графікою та сюжетом гри. Іншими словами, якщо сюжет гри фантастичний, його буде важко реалізувати з мінімальною графікою.

Щоб розробити привабливу гру, недостатньо просто створити розважальний контент. Важливо створити веселу атмосферу для всіх. Як було сказано вище, найпопулярнішою грою є різновид гри «три в ряд». Він представляє інформацію в цікавій формі, що робить його чудовим варіантом для гравців різного віку [12].

Наприклад, зміст гри «Candy Crush», яка є яскравим представником ігор «три в ряд», полягає у розгадуванні головоломок за допомогою смачних цукерок. Гра була завантажена понад півмільярда разів, вік її аудиторії – від 8 до 80 років. Тому розробники приділяють особливу увагу дизайну кожного рівня, кожного персонажа і постійно додають різноманітні оновлення. Усі персонажі Candy Crush веселі, барвисті та мають трохи химерності. Легко малювати різні види істот з естетикою в стилі анімаційної студії Hanna-Barbera. У грі є 380 рівнів, кожен з яких має привабливий макет і забезпечує баланс між складним і доступним. Розробники ігор вважають, що будь-яку ідею можна використати для створення веселої, необразливої гри. Кожна нова гра має давати гравцям новий досвід. Популярність і успіх гри підтверджує той факт, що її продовження – «Candy Crush Jelly Saga», «Candy Crush Soda Saga» і «Candy Crush Friends Saga» – виходили протягом 10 років.

2 МЕТОДОЛОГІЯ РОЗРОБКИ МОБІЛЬНОГО ІГРОВОГО ДОДАТКУ

2.1 Вибір технології і розробки мобільного ігрового додатку

Методи розробки мобільних ігрових додатків включають різноманітні інструменти, мови програмування та методології для створення ігор, які запускаються на мобільних пристроях, таких як смартфони та планшети.

Мобільні ігрові додатки мають різні розміри екрана, обмежені ресурси та взаємодію з сенсорним екраном, тому розробка для цих платформ має унікальні характеристики [13].

На основі проведенного аналізу для розробки мобільного ігрового додатку була вибрана платформа Android, яка має кілька переваг:

– великий ринок користувачів: Android є найпопулярнішою мобільною платформою у світі, тобто є багато потенційних гравців. Розробка мобільних ігорових додатків для Android дозволяє охопити широку цільову аудиторію;

– різноманітність пристроїв: Android працює на різних пристроях від різних виробників, що дає можливість адаптувати гру до різних розмірів екрана та властивостей пристроїв;

– відкритий код і спільнота: Android базується на відкритому коді, що заохочує спільноту розробників, що дає доступ до різноманітних інструментів і ресурсів, які можуть спростити процес розробки та налагодження;

– гнучкість і свобода для розробників: розробники Android мають велику свободу у виборі інструментів і мов програмування для створення мобільних ігрових додатків. Android підтримує Java, Kotlin, C та інші мови, що дозволяє розробникам вибрати технологію, яка їм найбільше підходить;

– простіший процес публікації: Публікація мобільного ігрового додатку в Google Play є відносно простим і легким процесом порівняно з іншими магазинами програм. Це дозволяє розробникам швидше виводити свої ігри на ринок і охоплювати ширшу аудиторію;

– гнучка монетизація: Google Play пропонує різноманітні методи монетизації, зокрема рекламу, платні завантаження та покупки через програму. Розробники можуть вибрати метод, який найкраще підходить для їх проекту.

Для аналізу ефективності вибору технології для розробки мобільних ігрових додатків на ОС Android варто врахувати наступні критерії:

– час розробки: проаналізувати скільки часу потрібно, щоб розв'язати проблему за допомогою кожної з технологій, якщо взяти фахівців одного рівня в кожній з технологій і дати однакове технічне завдання;

– наявність фахівців: проаналізувати наскільки швидко можна знайти розробників, які здатні створити продукт на високому якісному рівні та в подальшому супроводжувати проект;

– зручність розробки та налагодження: проаналізувати наскільки розвинені інструменти розробки і налагодження в рамках даної технології;

– документація і технічна підтримка: перевірити існування регулярної технічної підтримки для даної технології, та наскільки часто виходять оновлення, як швидко виправляються критичні помилки;

– швидкість роботи: проаналізувати наскільки швидким буде інтерфейс програми, чи будуть помітні затримки в переходах між екранами і станами додатка.

Враховуючи ці технічні аспекти та провівши аналіз на основі порівняння відеоігрових рушіїв, для створення мобільного ігрового додатку було використане інтегроване середовище розробки ігор Unity, що включає в себе графічний редактор, двигун для рендерингу графіки, фізичний двигун, систему анімації, аудіо-систему та інші компоненти, необхідні для розробки ігор. Unity підтримує різноманітні платформи, включаючи iOS та Android, що робить його ідеальним інструментом для створення мобільних ігрових додатків [14].

Рушій має систему компонентів, що визначають та контролюють поведінку ігрового об’єкта, до якого він приєднаний. Компонент має багато властивостей, які можна налаштувати в інспекторі, наприклад, камери, освітлення, звук, фізика

зіткнень і об'єктів, анімація, графічний інтерфейс і багато іншого. Створення мобільного ігрового додатку проходить через створення та налаштування власних компонентів. Кожен компонент можна створити за допомогою наявного сценарію, описати реальну логіку гри та поведінку пов’язаних з ними об’єктів.

C# – одна з основних мов програмування, яка використовується для створення ігор в Unity. Ключовими аспектами використання C# для програмування мобільних ігрових додатків у Unity є:

– Unity використовує C# як основну мову програмування, що включає логіку гри, обробку подій, анімацію тощо. Це означає, що весь ігровий код можна написати мовою C#;

– інтеграція з платформами: Unity підтримує розробку для обох платформ iOS і Android. Можна використовувати той самий код C# для створення мобільних ігрових додатків для iOS та Android. Також Unity пропонує кросплатформну розробку;

– бібліотеки та ресурси: Unity має різноманітні активи, бібліотеки та інші компоненти, доступні через Asset Store. Багато з цих ресурсів можна використовувати для прискорення розробки мобільних ігрових додатків;

– оптимізація для мобільних пристроїв: важливо оптимізувати гру відповідно до продуктивності мобільного пристрою, такого як смартфон або планшет. Також розглянуто сенсорне введення, щоб забезпечити комфортний ігровий процес;

– опції мультимедіа: Unity дозволяє легко інтегрувати звукові ефекти та графіку у мобільні ігрові додатки. C# створює звуки, анімацію тощо. Використовується для програмування логіки гри;

– навчання та ресурси: Unity має велику та добре задокументовану базу знань, яка допомагає розробникам зрозуміти її функції. Крім того існує багато онлайн-ресурсів, курсів і форумів для вивчення C# та Unity для розробки мобільних ігор.

В результаті вибір технології і розробки повинен відповідати конкретним потребам та вимогам вашого проекту, а також враховувати тенденції ринку та можливості майбутнього розвитку.

2.2 Визначення цільової аудиторії

Визначення цільової аудиторії для мобільного ігрового додатку – це процес ідентифікації та розуміння потенційної групи користувачів гри та визначення їхніх ключових характеристик. Це важливий крок у розробці і маркетингу, оскільки дозволяє команді розробників адаптувати продукт до потреб і очікувань цільової аудиторії.

До основних факторів, що визначають цільову аудиторію мобільних ігорових додатків, відносяться:

– вікова група: діти, підлітки, дорослі або поєднання різних вікових груп. Інтереси та хобі: Цінності та інтереси цільової аудиторії залежать від жанру гри, теми, графіки, ігрового процесу тощо;

– доступність: проаналізувати фінансову спроможність цільової групи витрачати гроші на мобільні ігри та покупки в магазинах;

– платформи та пристрої: мобільні платформи (iOS, Android) і пристрої (смартфони, планшети) для запуска мобільного ігрового додатку;

– соціальні характеристики: соціальні аспекти цільової аудиторії, включаючи соціальні медіа, участь у спільноті та бажання взаємодіяти з іншими гравцями.

Цільова аудиторія мобільних ігрових додатків є досить різноманітною, і її склад включає різні вікові групи, інтереси та рівні експертизи в геймінгу. Розуміння цих різних груп дозволяє розробникам створювати ігри, які відповідають потребам і очікуванням різних цільових сегментів. Оскільки ігри є динамічною сферою, смаки та вподобання гравців можуть змінюватися з часом [15].

Для визначення сучасних тенденцій необхідно проаналізувати популярність жанрів мобільних ігрових додатків серед гравців різного віку (табл 2.1).

Табл. 2.1 – Аналіз популярності жанрів мобільних ігрових додатків

Жанр	Вибірка в цілому	15-18 років	18-29 роки	30-39 роки	40-59 роки
Puzzle	54%	21%	35%	68%	74%
Word Games	43%	15%	27%	52%	63%
Match-3	40%	9%	22%	56%	56%
Simulation	32%	45%	41%	27%	21%
Strategy	31%	18%	28%	33%	32%
First Person Shooter	31%	39%	49%	27%	14%
Hidden Object	30%	6%	20%	35%	47%
RPG	26%	27%	31%	29%	18%
Sandbox	25%	64%	33%	16%	5%
Shooter	23%	27%	33%	21%	14%
Adventure	19%	15%	20%	21%	19%
Time-management	19%	9%	18%	25%	18%
Tower defense	19%	21%	33%	13%	11%
Hyper Casual	16%	24%	25%	10%	11%
Sports	14%	6%	16%	22%	7%
Battle royal	14%	18%	27%	13%	0%
Educational	13%	0%	12%	17%	18%
MMORPG	13%	9%	14%	21%	5%
Action	13%	21%	14%	14%	5%
Fighting	12%	15%	12%	14%	9%
Vehicle shooter	11%	9%	18%	14%	4%
MOBA	11%	12%	20%	11%	2%

На основі табличних даних можна створити кільцеву діаграму для наочної демонстрації популярності жанрів мобільних ігрових додатків (рис 2.1).

Рис. 2.1 – Діаграма популярності жанрів вцілому

Треба зазначити, що популярність жанрів може змінюватися в залежності від віку гравців, тому слід враховувати вподобання кожної категорії (рис 2.2).

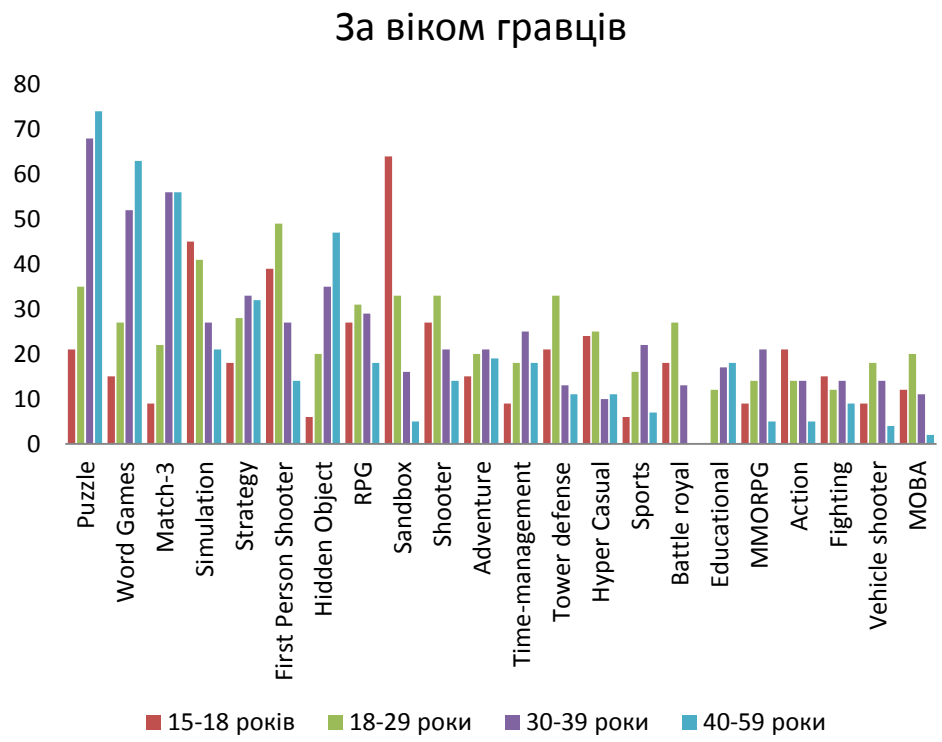

Рис. 2.2 – Діаграма популярності жанрів за віком гравців

Проведений аналіз показує, що більшість гравців віддає перевагу казуальним іграм, тобто відеоіграм, які призначені для різноманітних гравців, що мають невеликий ігровий досвід і можуть грати без особливих труднощів. Ключові особливості казуальних ігрових додатків включають простий геймплей, низькі бар'єри для входу, короткий час гри за сеанс, а також привабливий дизайн і графіку. Ці ігри можуть охоплювати різноманітні жанри, зокрема головоломки, аркади, симулятори та пригоди. Приклади казуальних ігор включають Candy Crush Saga, Angry Birds, Flappy Bird, Fruit Ninja, Crossy Road і Subway Surfers. Ці ігри в основному призначені для відпочинку та розваг і доступні на різних платформах, таких як мобільні телефони, планшети, комп'ютери та ігрові консолі.

Після збору інформації про цільову аудиторію можна сформувати портрет потенційного клієнта, який містить в собі характеристики та риси типових потенційних клієнтів групи. Спочатку створено кілька портретів, залежно від

вибраних жанрів створюваного мобільного ігрового додатку, таких як: платформер, казуальна гра та екшн-гра (табл. 2.2).

Табл. 2.2 – Портрет цільової аудиторії платформерів, казуальних ігор та екшн-ігор

Аудиторія	Жанр мобільних ігор		
	Платформер	Казуальні	Екшн-ігри
Вік	12-30 років	12-55 років	15-35 років
Інтереси	— Люди, які цінують швидкі та динамічні ігри з акцентом на стрибки, рефлекси та різноманітність рухів. — Гравці, які віддають перевагу простим інтерфейсам та ігровому процесу.	— Гравці, які шукають легкі, невимагаючі ігри для відпочинку та розваги. — Ті, хто не має багато часу на геймінг та шукає ігри, які можна швидко зіграти на коротких перервах.	— Гравці, які шукають емоційний та захоплюючий геймплей. — Ті, хто цінує великий арсенал зброї, ефекти та можливості керування персонажем.
Досвід геймінгу	Від початківців до середнього рівня досвіду. Простий геймплей може привертати новачків, а різноманітні виклики можуть задовольняти досвідчених гравців.	— Від початківців до середнього рівня досвіду. — Ті, хто може вперше випробовувати геймінг як форму розваги.	— Середній та високий рівень досвіду. — Гравці, які шукають складні завдання та конкурентний геймплей.
Стиль гри	— Яскраві та кольорові ігри з казковими або анімаційними графічними стилями. — Можливість використовувати прості жести або тач-екрани для управління головним героєм.	— Кольорові та привабливі ігри з простими правилами та інтуїтивним інтерфейсом. — Графічно прості ігри з приємною музикою та звуками.	— Ігри з високоякісною графікою та реалістичним дизайном. — Різноманітність сценаріїв та місій.
Зацікавлення у викликах	— Гравці, які цінують завдання на стрибання, уникаючи перешкод та розв'язування головоломок. — Ті, хто шукає аркадний ігровий процес та можливість піднімати рівень власної майстерності.	— Ігри, які не вимагають серйозного зосередження та навичок. — Можливість отримувати миттєве задоволення від гри.	— Гравці, які шукають швидкість, стрілянини та багатозадачність. — Ті, хто зацікавлений у багаторазових місіях та підвищенні навичок персонажа.

Створивши портрет цільової аудиторії окремих груп, відповідних до основних жанрів майбутнього мобільного ігрового додатку, можно скласти єдиний портрет (табл. 2.3).

Табл. 2.3 – Портрет цільової аудиторії мобільного ігрового додатку

Вік	12-30 років
Інтереси	— Гравці, які цінують простий та легкий управління, але при цьому шукають елементи динамічної гри. — Ті, хто любить казуальні ігри, але зацікавлений у викликах платформерів.
Досвід геймінгу	— Від початківців до середнього рівня досвіду. — Гравці, які шукають легкі завдання та при цьому можливість швидко вчитися грати.
Стиль гри	— Яскраві та кольорові ігри з простими правилами та привабливим дизайном. — Ігри зі спрощеним управлінням, щоб забезпечити ефективність на сенсорних екранах.
Зацікавлення у викликах	— Гравці, які цінують стрибки, швидкість та реакцію в реальному часі. — Люди, які шукають забавні та курйозні елементи в грі.

Цей портрет відображає головні характеристики цільової аудиторії мобільного ігрового додатку, що використовує цей опис для створення майбутнього продукту, який буде відповідати вподобанням та очікуванням цього сегменту гравців.

2.3 Проектування та дизайн мобільного ігрового додатку

Проектування та дизайн мобільного ігрового додатку — це комплексний процес, який включає в себе кілька етапів, одним з яких є аналіз ідеї через дослідження ринку. Попередньо визначивши цільову аудиторію можна почати вивчення конкурентів [16].

Аналіз існуючих рішень для створення мобільного ігрового додатку може включати вивчення їхніх переваг та недоліків, що допоможе визначити аспекти,

яких необхідно уникати чи навпаки притримуватись для більшого успіху власного продукту. Оскільки портрет цільової аудиторії вказує на мобільні ігрові додатки у жанрах платформер, казуальна гра та екшн-гра, будуть розглянуті ігри з відповіздними тегами (табл. 2.4).

Табл. 2.4 – Аналіз існуючих рішень у Cat Bird

Мобільний ігровий додаток	Cat Bird
Розробник	Raiyumi
Опис	"Вперед до пригод, покажи свої пазурі!"
Переваги	— Понад 40 складних та захоплюючих рівнів — Облік досягнень та загальний рейтинг гравців — Деталізована графіка та яскравий візуальний стиль — Художня та геймплейна різноманітність
Недоліки	— Наявність великої кількості реклами — Заїдають кнопки для стрибка та зльоту, що ускладнює проходження гри

"Cat Bird" — весела та захоплююча мобільна гра, розроблена невеликою інді-студією Raiyumi. Головний герой гри – кіт на ім'я Кіт Берд. Незважаючи на котячий вигляд, він має крила, як у птахів (рис 2.3).

Головна особливість Cat Bird в тому, що головний герой може літати через складні місця, уникаючи перешкод і ворогів, чіпляючись за різні поверхні, наприклад, залізні кулі або сходи. Гравці повинні використовувати фізику гри та елементи керування персонажами, щоб розгадувати головоломки.

Цей мобільний ігровий додаток має привабливий стиль піксельного арту, що нагадує 8-бітні ігри. Cat Bird пропонує різноманітні рівні, які кидають виклик гравцям і роблять гру довшою та цікавішою.

Ця гра ідеально підходить для тих, хто любить ігри на платформі з веселим геймплеєм, милим виглядом і навичками логічного мислення.

Рис. 2.3 – Скріншот мобільного ігрового додатку Cat Bird

"Dadish" – весела та мила мобільна гра від розробника Томаса К. Янга. (табл. 2.5). Гра вийшла в 2020 році і швидко завоювала популярність завдяки унікальному підходу до геймплея і стилю гумору (рис 2.4).

Табл. 2.5 – Аналіз існуючих рішень у Dadish

Мобільний ігровий додаток	Dadish
Розробник	Thomas K Young
Опис	"Досліджуйте захоплюючий світ, воювати з противниками на тему фаст-фуду і об'єднайте Дадіша з його зниклими дітьми в цій захоплюючій пригоді."
Переваги	— Поступово зростаюча складність. — Можливість перепроходити рівні — Проста та цікава стилістика
Недоліки	— Потребує забагато часу на проходженні більш складних рівнів — Деякі рівні майже неможливо пройти через невдалий левел-дизайн

"Dadish" – весела та мила мобільна гра від розробника Томаса К. Янга. Гра вийшла в 2020 році і швидко завоювала популярність завдяки унікальному підходу до геймплея і стилю гумору (рис 2.4).

Гравці керують Редькою, яка вирушає в неймовірну подорож, щоб врятувати дітей, які заблукали в рослинному світі. У грі можна досліджувати різні рівні, взаємодіяти з різними персонажами та виконувати різні завдання.

Основний геймплей Dadish — це поєднання платформерів і головоломок. Гравцям належить проходити різні рівні, уникати перешкод, вирішувати головоломки та збирати різні предмети.

Гра має яскраве барвисте оформлення, що зображує світ рослин, а також веселий саундтрек і цікаві діалоги. «Дадіш» привертає увагу гравців своєю простотою і крутою атмосферою.

Рис. 2.4 – Скріншот мобільного ігрового додатку Dadish

Гра також вражає своєю кількістю різноманітних персонажів, з якими гравець може взаємодіяти. Кожен персонаж має свою власну унікальну особливість або характер, що робить гру ще цікавішою.

"Yeah Bunny!" – це захоплююча мобільна аркадна гра, розроблена Adrian Zarzicki Studio (табл 2.6).

Табл. 2.6 – Аналіз існуючих рішень у Yeah Bunny!

Мобільний ігровий додаток	Yeah Bunny!
Розробник	Adrian Zarzycki
Опис	"Ця чудова аркада-платформер приймає вас на пригоду в ретро пікселів чарівний кролик світу."
Переваги	—Легке сенсорне керування при натисканні в будь-якій точці екрану —Досягнення різних супер дивовижних світів з унікальними персонажами та ворогами — Цікава система накопичення
Недоліки	— Задній фон зливається з перешкодами та ворогами — Подвійний стрибок працює криво та змінює направління персонажа коли це не потрібно

Випущена в 2017 році гра швидко набула популярності завдяки простому, але веселому геймплею та милому піксельному дизайну. Головний герой гри – кролик на ім'я Bunny. Гравці повинні керувати кроликом, який пересувається різними рівнями, збираючи моркву та інші предмети. Мета гри – допомогти кролику знайти кохання, яке забрав злий ведмідь (рис 2.5).

Гра передбачає біг і стрибки через різні перешкоди для проходження різних рівнів. Гравці можуть збирати моркву, щоб заробляти очки та використовувати їх для покращення різних характеристик кролика, зокрема швидкості та стрибків.

Піксельна графіка гри яскрава та мила, а музика робить гру ще приємнішою. Ця гра також вражає різними рівнями та великою кількістю секретів і прихованих об’єктів, які потрібно відкрити.

Рис. 2.5 – Скріншот мобільного ігрового додатку Yeah Bunny!

"Yeah Bunny!" ідеально підходить для любителів казуальних ігор зі свіжим геймплеєм, у який можна грати будь-коли та будь-де.

Таким чином проаналізувавши всі розглянуті недоліки можна зазначити основні проблеми, яких необхідно уникати та/або звернути увагу при створенні власного мобільного ігрового додатку, а саме: реалізація управління, коректно працюючий та інтуїтивно зрозумілий левел-дизайн та візуальний стиль, що не буде втомлювати та дезорієнтовувати гравця.

Розглянуті проекти також допомагають у постановці завдання, основною вимогою якого є створення готової гри у жанрі платформер для плаформи Android.

2.4 Розробка ігрового контенту

Створення мобільного ігрового додатку в жанрі платформер вимагає деталізованого постановлення завдання, щоб забезпечити розробникам чітке розуміння бажаного результату.

Режим гри складається з таких аспектів:

– управління: гравці керують своїми персонажами за допомогою вбудованих кнопок. Також необхідно додати можливість стрибати, бігати та використовувати спеціальні здібності;

– рівень: у грі є різні рівні з різними локаціями та завданнями. Складність прогресивно зростає з кожним рівнем;

– перешкоди і вороги: розмістити різні перешкоди на рівнях, такі як платформи для стрибків, шипи, ворожих персонажів тощо. Додати ворогів, які можуть рухатися та атакувати персонажа;

– графіка та анімація: у грі повинна бути приваблива і яскрава графіка, яка відповідає жанру платформера. Додати анімацію для рухів персонажів, стрибків, атак та інших дій;

– звуковий супровід: додати фонову музику та звукові ефекти відповідно до атмосфери гри;

– меню та інтерфейс: створити зручне та інтуїтивно зрозуміле головне меню та інтерфейс гри.

Для розробників інші ключові елементи, спрямовані на внутрішнє сприйняття та співпрацю в команді розробників містяться у "One Sheet" – короткому документі, який слугує для швидкого та лаконічного представлення ключових характеристик, особливостей та привабливостей продукту чи проекту. Цей термін часто використовується в індустрії розваг, зокрема в галузі розробки і випуску мобільних ігрових додатків, але його можна застосовувати і в інших галузях [17].

One Sheet призначений для швидкого враження та зацікавлення сторін, таких як інвестори, видавці, інші члени команди, або навіть потенційні гравці. Зазвичай він містить ключові елементи, такі як заголовок, короткий опис, головні особливості, механіка гри, цільова аудиторія, графіка, інформація про випуск та контактна інформація (рис 2.6).

Складання One Sheet для розробників:

– ім'я та назва компанії;

– назва гри або проекту;

– концепція – однорядковий опис мобільного ігрового додатку;

– платформа або жанр;

– X statement – однорядковий емоційний опис того, що гравці будуть робити в грі;

– опис – короткий виклад ігрового процесу в один рядок;

– ключові характеристики – 3 унікальні елементи, які відрізняють даний продукт від конкурентів;

– аудиторія;

– бенчмаркінг – чим дана гра буде відрізнятися від усіх інших?

– зображення, яке демонструє мобільний ігровий додаток.

Рис. 2.6 – One Sheet мобільного ігрового додатку

Перехід від розробки One Sheet до фактичного процесу розробки мобільних ігрових додатків вимагає значних кроків і планування:

– детальний план: розробка One Sheet є першим кроком у плануванні. Далі потрібно поставити конкретні цілі та завдання, визначити кроки розвитку та встановити терміни;

– створити проект: вибір інструменту розробки мобільних додатків – Unity. Платформа – Android;

– створити прототип: продемонструвати ключові механізми та ідеї. Створення прототипів дозволяє швидко тестувати та вдосконалювати концепції гри;

– дизайн гри: створити детальні ігрові моделі, які включають графіку, анімацію, аудіо тощо. Визначити стиль гри та створити концепт-арт;

– розробка коду: написати код, який реалізує функції гри. Розробка може включати роботу над різними аспектами гри, такими як керування гравцем, взаємодія об’єктів тощо;

– тестування: тестувати гру на різних пристроях і платформах, знаходити та виправляти помилки та проблеми. Тестування можна проводити внутрішньо або за допомогою тестувальників чи гравців;

– оптимізація та полірування: оптимізувати гру для оптимальної продуктивності та додати останні штрихи;

– запуск: завершити розробку та підготувати гру до випуску.

Випустивши мобільний ігровий додаток на вибраній платформі та забезпечивши підтримку, включаючи вирішення проблем та видачу оновлень, треба слідкувати за відгуками гравців, адаптувати гру до їхніх потреб та продовжувати розвивати та вдосконалювати проект [19].

Загальною метою є створення ігрового контенту, який не лише задовольняє потреби гравців, але й залишається актуальним та захоплюючим протягом тривалого періоду.

3 РОЗРОБКА МОБІЛЬНОГО ІГРОВОГО ДОДАТКУ

3.1 Створення ігрової механіки

Ігрова механіка стосується правил, процесів і систем взаємодії, які точно визначають, як гравці взаємодіють із ігровим світом і як різні компоненти гри взаємодіють один з одним. Механіка визначає, як гравці контролюють гру та взаємодіють з нею, як рухаються персонажі, як взаємодіють об’єкти в ігровому середовищі, як вирішуються завдання та конфлікти.

3.1.1 Механіки гравця

Механіка гравця в грі – це система взаємодій і правил, які точно визначають, як гравці взаємодіють з грою. Це стосується того, як персонаж або об’єкт контролюється, як він рухається в ігровому світі, як він взаємодіє з іншими персонажами та об’єктами, а також вплив гравця на саму гру [20].

Механіка платформерів – це основні елементи та правила гри, що визначають рух персонажа та його взаємодію з оточуючим світом в жанрі платформерів. Вони мають акцентовану на логіці та рефлексах геймплейну механіку, де гравець керує персонажем, який має переміщуватися, стрибати та взаємодіяти з різними платформами для досягнення мети.

Механіка руху гравця в платформерах грає критичну роль у визначенні геймплейного досвіду, тому була створена у першу чергу. Ключові елементи механіки руху гравця включають:

– біг і ходьбу: гравці можуть самостійно рухати свого персонажа вперед або назад. Це може бути важливо під час адаптації до різних ситуацій, таких як уникнення ворожих атак або зіткнення з платформами (рис 3.1);

– стрибки: стрибки відіграють важливу роль, дозволяючи пересуватися через прогалини, уникати небезпек і досягати вищих платформ. Також додана можливість подвійного стрибка;

– біг по стінах: можна триматися за стіни, що розширює здатність гравця взаємодіяти з навколишнім світом.

```
public void Move(float move, bool jump)
{
    anim.SetFloat("speed", Mathf.Abs(move));

    bool isTouchSide = Physics2D.OverlapCircle(wallJumpCheck.position, groundRadius, WhatIsGround);

    if (!isGrounded && move != 0 && isTouchSide)
    {
        ExtraJump = 1;
        anim.SetBool("wallJump", true);
    }
    else
        anim.SetBool("wallJump", false);

    rb.velocity = new Vector2(move * speed, rb.velocity.y);
    if (move > 0 && !FacingRight)
        Flip();
    else if (move < 0 && FacingRight)
        Flip();

    if(jump && (isGrounded || ExtraJump > 0))
    {
        if(audio)
            audio.Play("Player Jump");
        if (!isGrounded)
        {
            anim.SetBool("doubleJump", true);
            ExtraJump--;
        }
        isGrounded = false;
        rb.velocity = Vector2.up * jumpSpeed;
    }
    if (isGrounded && anim.GetBool("doubleJump"))
        anim.SetBool("doubleJump", false);
}
```

Рис. 3.1 – Код механіки руху гравця, який включає біг, ходьбу, стрибки та біг по стінах

Для бігу персонажа була встановлена горизонтальна швидкість та перевірка в якому напрямку рухається персонаж, і, якщо напрямок не відповідає напрямку, до якого обличчям дивиться персонаж, то викликається метод Flip, який обертає обличчя персонажа.

Для стрибків була втановлена вертикальна швидкість, гравітація, яка впливає на швидкість падіння персонажа після виконання дії та перевірка чи відбувається стрибок (jump), чи персонаж торкається землі (isGrounded), або чи є

додаткові стрибки (ExtraJump). Якщо умова виконується, виконується стрибок, і гравець може здійснити подвійний стрибок.

Для бігу по стінам був встановлен параметр, що визначає, чи торкається персонаж бічної поверхні, яка може бути використана для стрибка вздовж стіни та перевірка чи персонаж не торкається землі, рухається вліво або вправо та торкається бічної поверхні. Якщо умова виконується, персонаж може здійснити стрибок вздовж стіни.

Механіка збору предметів грає важливу роль у платформерах, додаючи глибину геймплею та стимулюючи гравця досліджувати рівні. За сценарієм гри для головного персонажа збирання фруктів являється ключовою роллю (рис 3.2).

```
public class Fruit: MonoBehaviour
{
    Animator anim;
    Сообщение Unity | Ссылок: 0
    private void Start()
    {
        anim = GetComponent<Animator>();
    }
    Сообщение Unity | Ссылок: 0
    private void OnTriggerEnter2D(Collider2D col)
    {
        if(col.gameObject.CompareTag("Player"))
        {
            FindObjectOfType<AudioManager>().Play("Fruit Collected");
            anim.SetTrigger("dead");
        }
    }
    Ссылок: 0
    public void Destroy()
    {
        GameManager.FruitCounter();
        Destroy(gameObject);
    }
}
```

Рис. 3.2 – Код механіки взаємодії гравця з фруктами

Для цього створено код, що запускає метод, який викликається, коли інший об'єкт збігається з колайдером фрукта, та перевіряє чи це зіткнення стосується гравця (об'єкта з тегом "Player"). Далі метод викликається під час обробки фруктом гравцем для збільшення лічильника фруктів та видаляє геймоб'єкт фрукта.

Механіка лічильника (або лічби) в платформерах може використовуватися для відстеження та відображення різних параметрів або досягнень гравця. Так як одним із запланованх жанрів мобільного ігрового додатку є екшн-гра, то було створено лічильник, який відраховує час на виконання завдання зі збору таких об’єктів як фрукти, що додає мотивації гравцю (рис 3.3).

```
void Update()
{
    if (ui_TimerText == null || fruit == null) return;
    timer -= Time.deltaTime;

    if(timer >= 0)
    {
        if(fruitCounter <= 0)
        {
            level++;
            GameData data = SaveSystem.LoadGameData();
            if((data != null && data.level < level) || data == null)
                SaveSystem.SaveGameData(this);
            LoadNextLevel();
            ui_TimerText = null;
            return;
        }
        int t = (int)timer;
        ui_TimerText.text = t.ToString();
    }
    else
    {
        RestartScene();
    }

    if(player.transform.position.y <= -10)
    {
        RestartScene();
    }
}
```

Рис. 3.3 – Код механіки лічильника

Лічильник таймера (timer) постійно зменшується на кількість секунд, яка пройшла з попереднього кадру та перевіряє чи таймер ще не вичерпався. Якщо таймер ще не завершився, виконується наступний блок коду, в іншому випадку викликається метод RestartScene, який починає рівень заново.

Також код перевіряє чи кількість зібраних фруктів (fruitCounter), якщо гравець зібрав всі фрукти на поточному рівні відбувається збільшення рівня (level++), ініціалізація або оновлення гри (збереження гри, завантаження наступного рівня через метод LoadNextLevel).

Параметр TimerText відповідає за посилання на текстове поле таймера та завершує метод. Якщо таймер ще не вичерпався, обчислюється залишковий час в секундах і встановлюється це значення в текстове поле для таймера.

3.1.2 Механіки платформ та пасток

Механіка платформ грає ще одну ключову роль у створенні цікавого та викликального геймплею. Платформи — це поверхні, на яких персонаж може рухатися, стрибати або взаємодіяти. Основними елементами механіки платформ представлені:

– статичні платформи: Звичайні непорушні платформи, на яких персонаж може стояти чи рухатися;

– пружини та трампліни: Механізми, які дозволяють персонажеві стрибати вище або на далекі відстані (рис 3.4);

– падіння платформ: Деякі платформи можуть падати через деякий час при контакті з персонажем, створюючи потребу у вчасних стрибках чи швидкій реакції гравця (рис 3.5).

```
public class Arrow : MonoBehaviour
{
    public float JumpForce;

    Сообщение Unity | Ссылок: 0
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if(collision.gameObject.CompareTag("Player") && collision.gameObject.GetComponent<PlayerInput>().enabled)
        {
            GetComponent<Animator>().SetTrigger("hit");
            collision.gameObject.GetComponent<PlayerController>().JumpPlayer(JumpForce);
        }
    }
    Ссылок: 0
    public void Destroy2AnimationFinish()
    {
        Destroy(gameObject);
    }
}
```

Рис. 3.4 – Код механіки об’єкта «стрілка», що підкидає гравця

Механіка працює через змінну, що визначає силу стрибка, яку отримає гравець при зіткненні з цією стрілою. Метод, який викликається, коли інший

об'єкт зіткнувся з колайдером цієї стрілки перевіряє чи це зіткнення стосується гравця (об'єкта з тегом "Player") та передає силу стрибка (JumpForce).

```
private void Update()
{
    if (isTouch)
    {
        timer += Time.deltaTime;
        if (timer > DestroyTime)
        {
            effect.SetActive(false);
            GetComponent<BoxCollider2D>().isTrigger = true;
            transform.position = Vector2.MoveTowards(transform.position, new Vector3(0, -100, 0), 10 * Time.deltaTime);
        }
        if (transform.position.y < -10)
            Destroy(gameObject);
    }
    else
    {
        transform.position = Vector2.MoveTowards(transform.position, StartPos + newPos, speed * Time.deltaTime);
        if (Vector2.Distance(transform.position, StartPos + newPos) < 0.05f)
        {
            newPos *= -1;
        }
    }
```

Рис. 3.5 – Код механіки падіння платформи

Спочатку код перевіряє чи вже об'єкт торкається чогось (isTouch). Залежно від цього виконується різний блок коду. Якщо об'єкт торкається персонажу, то timer збільшує лічильник таймера на час, що пройшов з останнього кадру, що дає можливість гравцеві встигнути зреагувати через затримку часу, після чого вимикається активність платформи, що дозволяє іншим об'єктам проходити через неї. Позиція об'єкта зміщується вниз з певною швидкістю і якщо вона по координаті у менше -10, то платформа знищується (рис 3.6).

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.CompareTag("Player") && collision.gameObject.GetComponent<PlayerInput>().enabled)
    {
        anim.SetBool("isJump", true);
        collision.gameObject.GetComponent<PlayerController>().JumpPlayer(JumpForce);
    }
}
```

Рис. 3.6 – Код механіки трампліну

Скрипт перевіряє чи об'єкт, з яким відбулося зіткнення, має тег "Player" і чи у гравця увімкнений компонент PlayerInput. Це призначено для перевірки, чи зіткнення відбулося з гравцем і чи може він реагувати на це зіткнення. Після чого

викликає метод JumpPlayer, передаючи силу стрибка (JumpForce), призначений для того, щоб ініціювати стрибок гравця при зіткненні.

Механіка пасток у платформерах використовується для створення викликів та небезпеки для гравця. Пастки можуть приймати різні форми і вимагати від гравця швидкі реакції та вивчення різних способів їх уникнення:

– шипи: Розташовані на стінах, стелях або підлозі, шипи можуть наносити гравцеві шкоду при контакті, створюючи зони, які гравець повинен уникати;

– пастки, що активуються: Активуються при контакті гравця з необхідним тригером (рис 3.7.);

– рухливі перешкоди: Пастки, які можуть рухатися або пересуватися, такі як гострий блок або ковзаючий шар (рис 3.8).

```
void FixedUpdate()
{
    if(isTouch)
    {
        if (fireField.enabled && fireField.IsTouching(player))
        {
            fireField.enabled = false;
            player.gameObject.GetComponent<PlayerController>().PlayerDead();
        }
        else
        {
            timer += Time.fixedDeltaTime;
            if (timer > delay)
            {
                if (anim.GetInteger("index") == 1)
                {
                    fireField.enabled = true;
                    anim.SetInteger("index", 2);
                    delay = 1f;
                }
                else
                {

                    fireField.enabled = false;
                    delay = 0.4f;
                    anim.SetInteger("index", 0);
                    isTouch = false;
                }
                timer = 0f;
            }
        }
    }
}
```

Рис. 3.7 – Код механіки вогняної пастки

Спочатку перевіряється чи область зони вогню (представленої об'єктом fireField) активна і торкається гравця (player). Якщо це так, тобто персонаж активував пастку та знаходиться в зоні ураження, то виконується наступне: вимикається область зони вогню та викликається метод PlayerDead об'єкта гравця, що відповідає за обробку смерті гравця в грі.

Якщо умова в першому кроці не виконується, то timer збільшує лічильник таймера на фіксовану кількість часу, що пройшло між кадрами та перевіряє чи таймер перевищив встановлену затримку (delay). Якщо так, то увімкнює область зони вогню, якщо ні встановлює нову затримку, перезапускаючи пастку.

```
void Start()
{
    ai = GetComponent<AIMovement>();
}
Сообщение Unity | Ссылок: 0
void FixedUpdate()
{
    ai.Move();
}
Сообщение Unity | Ссылок: 0
private void OnTriggerEnter2D(Collider2D collision)
{
    if(collision.gameObject.CompareTag("Player") && collision.gameObject.GetComponent<PlayerInput>().enabled)
    {
        collision.gameObject.GetComponent<PlayerController>().PlayerDead();
    }
}
```

Рис. 3.8 – Код механіки пили

Метод викликається з фіксованою частотою і викликає Move об'єкта ai, що представляє собою іншого персонажа чи ігровий об'єкт, і відповідає за його рух чи поведінку в грі. Тобто цей код забезпечує фіксований рух об'єкта пили в грі.

Коли інший об'єкт зіткнувся з колайдером, до якого прикріплений цей скрипт, спрацьовує перевірка, чи об'єкт зіткнення має тег "Player" і викликається метод PlayerDead, відповідаючи за обробку смерті гравця (рис 3.9).

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.CompareTag("Player") && collision.gameObject.GetComponent<PlayerInput>().enabled)
    {
        collision.gameObject.GetComponent<PlayerController>().PlayerDead();
    }
}
```

Рис. 3.9 – Код механіки шипів

Як і у минулому випадку код перевіряє коли об'єкт, до якого прикріплений цей скрипт, зіткнувся з гравцем і умови перевірки виконуються, то викликається метод PlayerDead, який може викликати реакцію на смерть гравця у грі.

3.1.3 Механіки ворогів

Механіка ворогів у платформерах включає різноманітні елементи, що визначають поведінку та взаємодію ворогів з гравцем. Такі механіки можуть створювати виклики та додавати стратегічний елемент до геймплею:

– рухливі ворогів: вороги можуть рухатися різними способами — пересуванням по заданому маршруту, стрибанням або літаючи (рис 3.10);

– агресивні вороги: вороги, які активно переслідують гравця, коли він знаходиться в певному радіусі;

– вороги з особливими властивостями: вороги, що мають спеціальні навички чи атаки, такі як телепортація, вибухи або імунітет до певних видів атак (рис 3.11).

```
void Update()
{
    ai.Move(true);
}

Сообщение Unity | Ссылок: 0
private void OnCollisionEnter2D(Collision2D collision)
{
    if (ai.Ai_Collision(collision))
    {
        ai.speed = 0;
    }
}
Сообщение Unity | Ссылок: 0
private void OnTriggerEnter2D(Collider2D collision)
{
    if (ai.Ai_Trigger(collision))
    {
        GetComponent<Animator>().SetBool("hit", true);
    }
}
Ссылок: 0
public void Die()
{
    GetComponent<Animator>().enabled = false;
}
```

Рис. 3.10 – Код механіки птаха

Цей метод викликається кожен кадр гри і вказує на рух птаха вперед або взаємодію з чимось, коли інший об'єкт потрапляє у тригер (зону активності) ворога, до якого прикріплений скрипт. Метод public void Die визивається, коли птах повинен вмерти. Тобто механіка птаха подібна до механіки пили з простою траєкторією пересування.

```
void Update()
{
    timer += Time.deltaTime;
    if (timer > waitTime)
    {
        RaycastHit2D[] hit = Physics2D.RaycastAll(transform.position, dir);
        foreach (var h in hit)
        {
            if (h.collider.gameObject.layer == 8) break; //ground

            if (h.collider.tag == "Player" && !run)
            {
                if (dir == Vector2.right && FacingRight)
                    Flip();
                else if (dir == Vector2.left && !FacingRight)
                    Flip();
                force = dir;
                run = true;
            }
        }
        if (run)
            rb.AddForce(force * speed * Time.deltaTime);
        dir *= -1;
    }
    anim.SetBool("run", run);
}
```

Рис. 3.11 – Код механіки переслідування гравця носорогом

Код реалізує логіку руху носорога в грі в режимі переслідування гравця. Параметр timer збільшує таймер на кількість пройденого часу від останнього кадру, якщо умова виконується, то RaycastHit2D здійснює лучевий каст, тобто процес перетворення одного типу даних, у вказаному напрямку dir від поточної позиції об'єкта, що відповідає за реагування носорога на гравця та початок його переслідування.

Якщо зіткнення відбулось із землею, то цикл припиняється (break), і код продовжує виконуватися, тобто поведінка ворога не змінюється.

Код в основному реалізує алгоритм преслідування гравця об'єктом, змінюючи напрямок руху, взаємодіючи з лучем, щоб визначити наявність гравця,

і використовуючи фізичний двигун для надання об'єкту руху в напрямку гравця, надаючи йому задану силу (рис 3.12).

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.CompareTag("Player") && !isDie)
    {
        collision.gameObject.GetComponent<PlayerController>().PlayerDead();
        enabled = false;
        anim.SetBool("run", false);
        isDie = true;
        return;
    }
    if (run)
    {
        timer = 0f;
        anim.SetBool("hitwall", true);
        rb.velocity =  (force.x > 0 ? new Vector2(-1.5f, 2) : new Vector2(1.5f, 2)) * geriTepmeHizi;
    }
    run = false;
}
```

Рис. 3.12 – Код механіки зіткнення об’єктів з носорогом

Спочатку код перевіряє чи носоріг зіткнувся з персонажем і якщо умови виконуються, то викликає метод PlayerDead, що відповідає за обробку смерті гравця.

Після задає нову швидкість (rb.velocity) об'єкта, орієнтовану у зворотному напрямку від напрямку руху (force). Тобто, носоріг реагує на зіткнення, зупиняється та може викликати удар ("hitwall").

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if(collision.CompareTag("PlayerFoot") && !isDie && !isUnDead)
    {
        FindObjectOfType<AudioManager>().Play("Enemy Death");
        collision.transform.parent.gameObject.GetComponent<PlayerController>().JumpPlayer();
        Die();
        anim.SetBool("isDie", true);
        enabled = false;
        isDie = true;

    }
}
Ссылок: 1
public void Die()
{
    GetComponent<BoxCollider2D>().enabled = false;
    GetComponent<CircleCollider2D>().enabled = false;
    rb.velocity = Vector2.up * Random.Range(2, 5);
}
```

Рис. 3.13 – Код механіки смерті носорога

Цей код відповідає за реакцію носорога на подію "зіткнення" (в цьому випадку, коли інший об'єкт входить у тригер) і реалізує логіку смерті об'єкта, якщо гравець, викликає метод JumpPlayer, тобто гравець здійснив стрибок, як реакцію на смерть ворога.

Для обробки логіки смерті носорога, вимикаються коллайдери (BoxCollider2D та CircleCollider2D), щоб носоріг перестав взаємодіяти з іншими об'єктами.

Отже, код реалізує сценарій смерті носорога, коли він зіткнувся з об'єктом гравця, який має тег "PlayerFoot" і визначає ряд подій, що відбуваються при цій ситуації (рис 3.14).

```
void Update()
{
    timer += Time.deltaTime;
    if(timer > delay)
    {
        GameObject p = Instantiate(slimeParticle, particleCheck.position, Quaternion.identity);
        Destroy(p, 5);
        timer = 0f;
    }

    ai.Move(true);

}
```

Рис. 3.14 – Код механіки слайма

Код виконує дві основні дії в кожному кадрі під час виконання програми. Перша збільшує таймер (timer) на час, який пройшов з останнього кадру та створює новий об'єкт частинок (slimeParticle) за допомогою функції Instantiate. Цей об'єкт частинок розташований у позиції, що визначена позначкою particleCheck.position, і його обертання встановлюється в єдиний обертовий об'єкт (Quaternion.identity). Тобто слайм, рухаючись, залишає за собой слід, з яким може взаємодіяти гравець.

Функцію Destroy існує для створеного об'єкта частинок (p), щоб він самознищився через 5 секунд. Друга дія викликає метод Move, що відповідає за рух самого слайма. Отже, загальною ідеєю цього коду є створення об'єкта

частинок з певною періодичністю і виклик руху слайма у кожному кадрі (рис 3.15).

```
private void Update()
{
    timer += Time.deltaTime;
    if(timer > delay)
    {
        GetComponent<Animator>().SetBool("particle", true);
        enabled = false;
    }
}

Сообщение Unity | Ссылок: 0
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.CompareTag("PlayerFoot"))
    {
        collision.transform.parent.gameObject.GetComponent<PlayerController>().PlayerDead();
        enabled = false;
    }
    else if (collision.gameObject.CompareTag("Player"))
    {
        collision.gameObject.GetComponent<PlayerController>().PlayerDead();
        enabled = false;
    }
}
```

Рис. 3.15 – Код механіки частинок слайма

Метод перевіряє, чи об'єкт, з яким сталось зіткнення, має тег "PlayerFoot" та викликає метод PlayerDead, що відповідає за обробку смерті гравця. Також якщо об'єкт має тег "Player" (але не "PlayerFoot"), виконується аналогічна логіка. Тобто розглядаються випадки, коли гравець міг стрибнути на частинки слайма та коли він в них просто зайшов.

Дія коду полягає в тому, що при зіткненні з об'єктом з тегом "PlayerFoot" або "Player", відбувається смерть гравця, після чого компонент, до якого прикріплений цей скрипт, вимикається.

3.2 Графіка та анімація

Графіка в платформних іграх відіграє важливу роль у створенні чіткого та привабливого візуального стилю, одночасно доносячи необхідну інформацію гравцеві [22]. До важливих аспектів ігрової графіки належать:

– колірна палітра: використання яскравих і насичених кольорів підвищує веселість і динамічність гри. Важливі об'єкти, персонажі та різні елементи ігрового процесу можна виділяти кольором;

– пропорції персонажів і анімація: важливо, щоб силует персонажа був чітко окреслений і зрозумілий, щоб гравець міг легко ним керувати. Плавна та виразна анімація під час стрибків, бігу та взаємодії з навколишнім середовищем робить гру веселою;

– правила та умови: фони та детальні зображення можуть додати глибини та атмосфери. Важливо, щоб фон не порушував чіткого розмежування між персонажами та платформами;

– дизайн платформи та рівня: створення різноманітних платформ, які виглядають як частина ігрового світу та логічно вписуються в гру. Графічний дизайн рівня повинен полегшити гравцям уявлення куди йти і де на них чекають випробування;

– ефекти середовища та анімація: створення яскравого і цікавого світу, використовуючи погодні ефекти (дощ, сніг), фонову анімацію та рухомі об’єкти. Важливо, щоб навколишні анімації не відволікали гравця від основної гри.

3.2.1 Дизайн персонажей і рівнів

Щоб створити консистентний світ гри необхідно використовувати тематичні елементи та стиль. За проведеним аналізом конкурентних мобільних ігрових додатків можна сказати, що піксельний стиль в платформерах користується популярністю.

Високоякісна піксельна графіка дозволяє створити деталізовані та стилізовані персонажі та рівні. А використання палітри та дизайну, що нагадує старі ігри, може створити враження ностальгії та додати унікальність.

Для економії часу і ресурсів замість створення ассетів персонажей та об’єктів було використано можливості вбудованого в Unity Asset Store. Це онлайн-магазин, який інтегрований безпосередньо в середовище розробки Unity, і

дозволяє розробникам швидко та легко знаходити, придбати, або безкоштовно завантажувати різноманітні ресурси для своїх проектів. Asset Store (рис 3.16) має широкий вибір готових компонентів, моделей, текстур, анімацій, звукових ефектів, плагінів та інших активів, які можна використовувати для поліпшення і розширення функціональності гри.

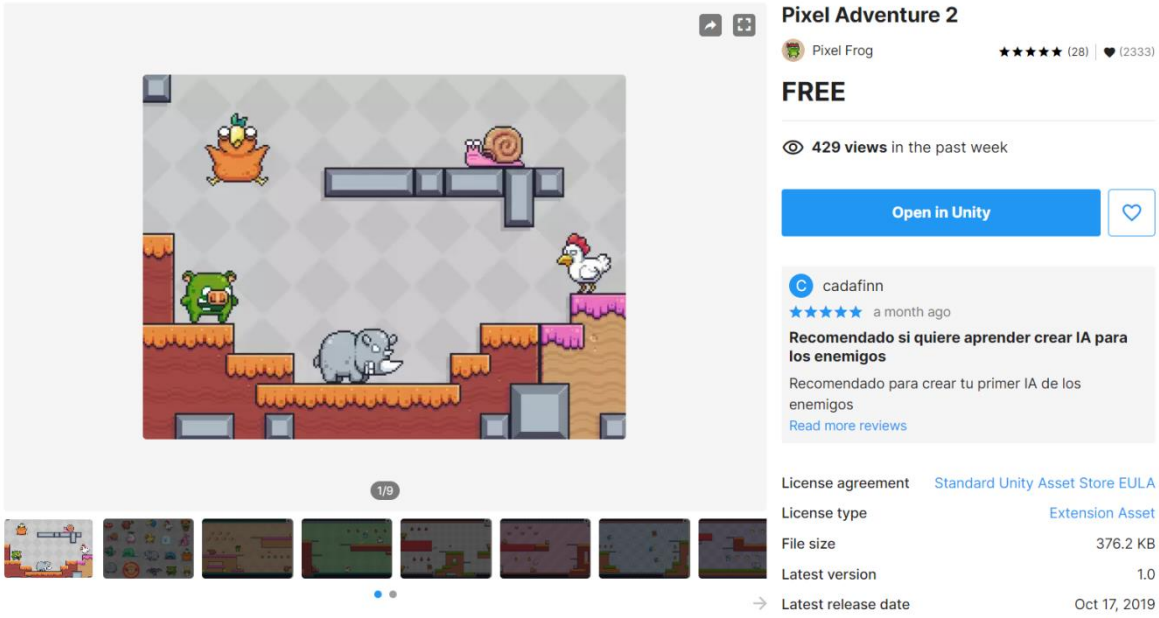

Рис. 3.16 –Unity Asset Store

З Asset Store було додано дві частини Pixel Adventure, які містять усі графічні зображення:

– 4 головних персонажів;

– 15 пасткок;

– 12 комбінацій наборів елементів;

– 20 різних ворогів;

– інші дрібні елементи (фрукти, коробки тощо).

Побудова рівнів у платформерах – це важлива та творча частина розробки гри. Необхідно забезпечити зростання важкості рівнів. Нові геймплейні елементи та виклики повинні вводитися поступово, та дати гравцеві можливість вивчати оточення та адаптуватися.

Перший рівень повинен бути простим та дозволяти гравцеві освоювати управління та механіку гри. Тому перший рівень було створено лише з

використанням звичайного оточення та розміщення фруктів (рис. 3.17), що дає гравцеві можливість зрозуміти принципи роботи основних механік.

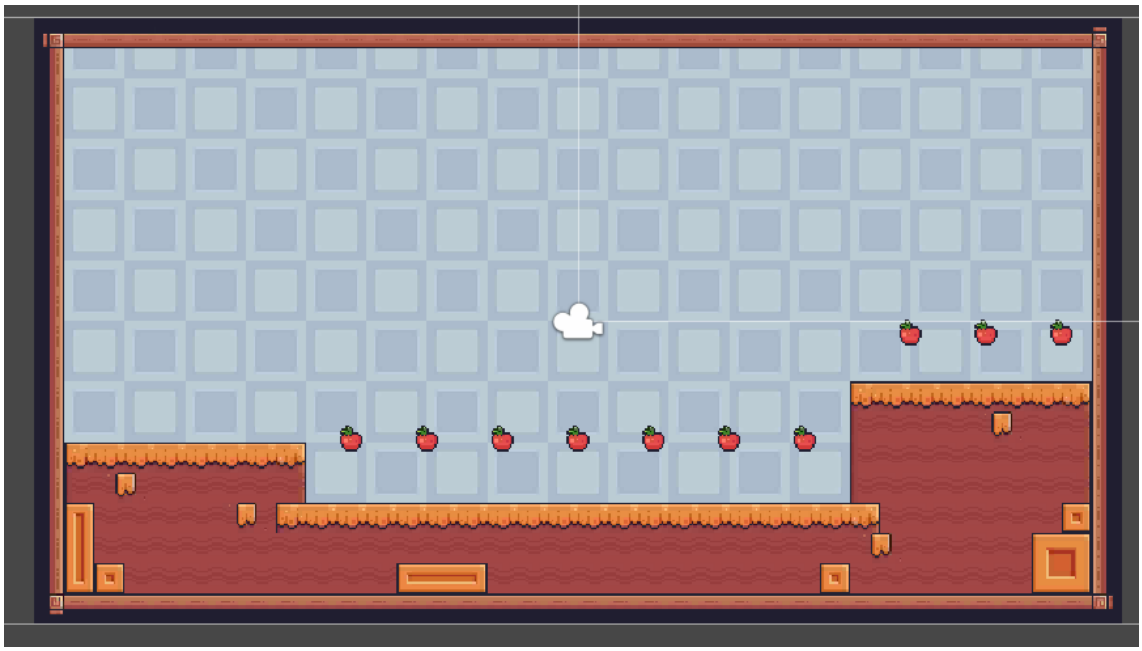

Рис. 3.17 – Сцена першого рівня

Другий рівень має на меті закріпити ефект навчання та наштовхнути гравця на думку про поступове збільшення складності (рис 3.18).

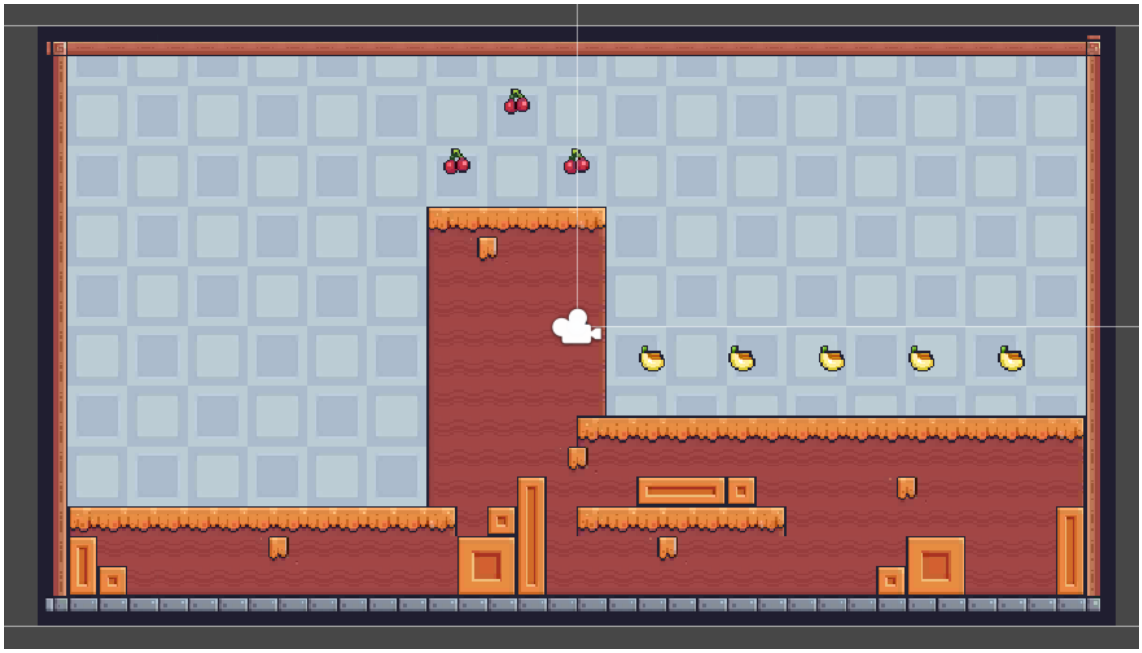

Рис. 3.18 – Сцена другого рівня

Для третього рівня вводиться механіка пасток та трамплінів на прикладі елементу «стрілка», яка підкидає гравця вгору, даючи змогу зробити потрійний стрибок (рис. 3.19).

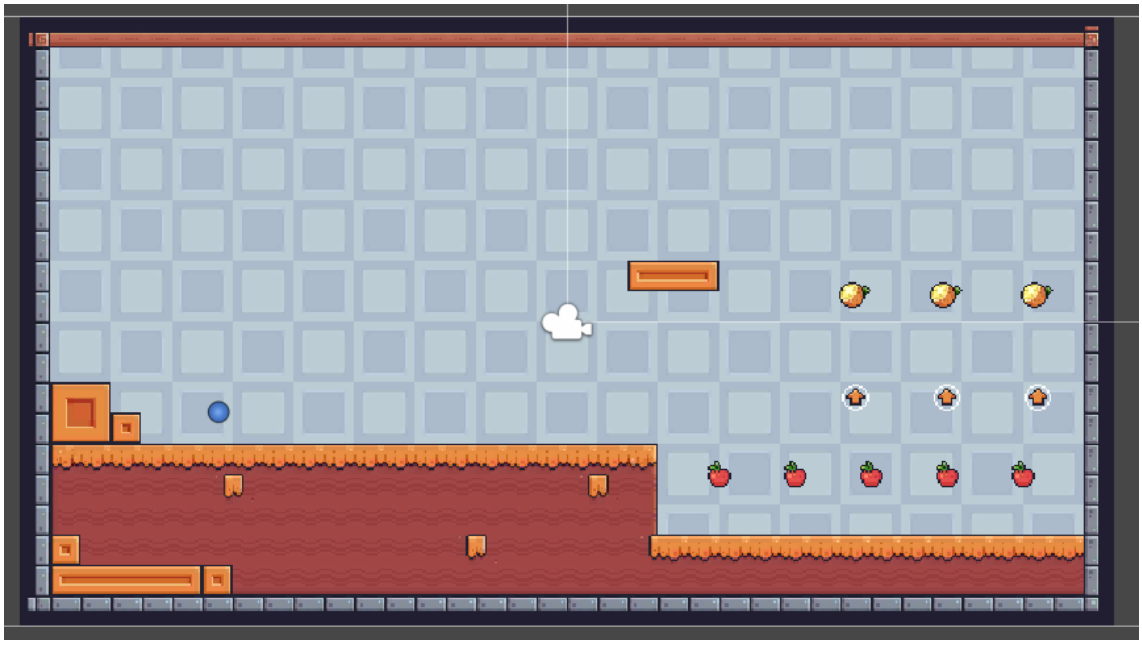

Рис. 3.19 – Сцена третього рівня

Далі об'єкти та платформи роблять чітке виділення головних шляхів та областей для досліження гравцем нових типів платформ та механік (рис 3.20).

Рис. 3.20 – Сцена четвертого рівня

У п’яту сцену задля різноманіття платформ та об'єктів додано декілька типів пасток: шипи та вогняна пастка, та введено поняття ворогів, з якими можна взаємодіяти: птах та слайм (рис. 3.21).

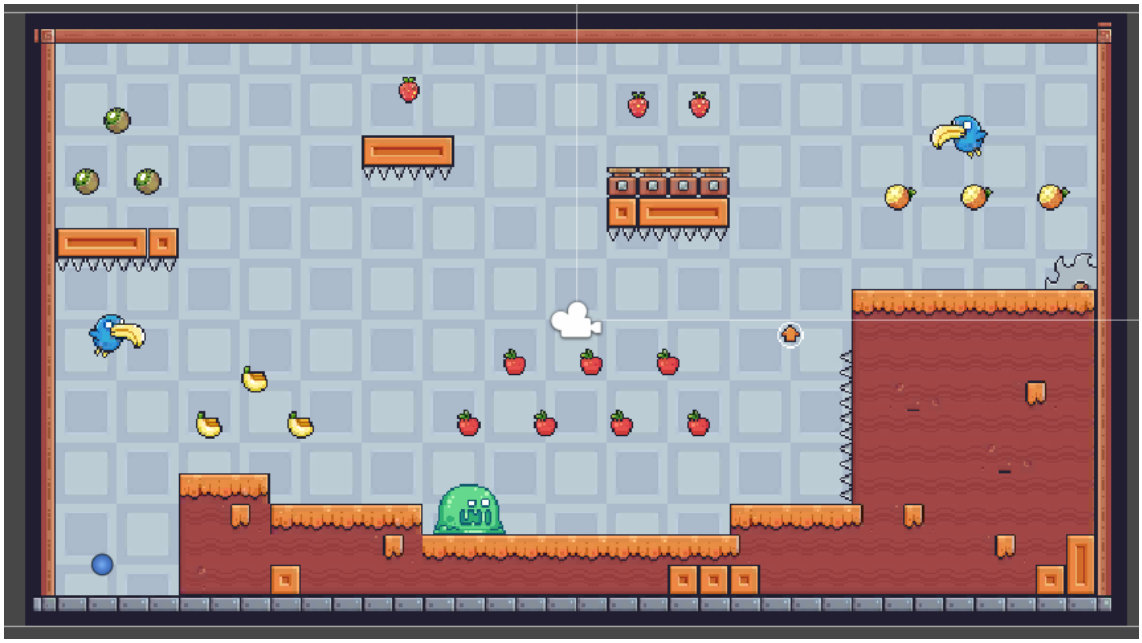

Рис. 3.21 – Сцена п’ятого рівня

До шостої сцени додано ворога типу носоріг, який являється найсильнішим серед створених противників та закріплює ефект зростання основого рівня важкості (рис 3.22).

Рис. 3.22 – Сцена шостого рівня

Таким чином нові концепції представлені гравцю в контрольованих умовах для їх засвоєння, додано різні види платформ, ворогів, перешкод, щоб уникнути монотонності. Також зони відпочинку після важких викликів, дозволяючи гравцеві оглянутися та підготуватися до наступних завдань.

3.2.2 Анімація персонажей

Створення анімації персонажа надає гравцю зручне та реалістичне відчуття контролю над героєм. Вона повинна чітко відображати стан, який визвав гравець своїми взаємодіями с навколишнім світом гри чи з самим персонажем.

Основним інструментом для роботи з анімаціями в Unity є Animator Controller, який дає можливість створювати анімації за допомогою інструментів, таких як Animation Window та Timeline, а потім використовувати Animator Controller для управління переходами між анімаціями під час виконання гри [23].

Основний принцип роботи Animator Controller ґрунтується на визначенні різних станів та переходів між ними:

– стани (States): кожен стан представляє собою конкретну анімацію чи групу анімацій, які відтворюються, коли об'єкт перебуває в цьому стані. Стани можуть включати рухи, обертання, різні дії та інші аспекти анімацій (рис. 3.23);

– параметри (Parameters): параметри визначають умови, за яких буде відбуватися перехід між станами. Параметри можуть бути числовими (наприклад, швидкість руху), булевими (наприклад, чи прискорений гравець), або вказувати на конкретні значення (наприклад, ім'я атаки);

– переходи (Transitions): переходи визначають, як і коли відбувається перехід між станами. Переходи відбуваються на підставі значень параметрів або внутрішніх умов;

– керування анімаціями під час виконання: за допомогою скриптів у проекті можна змінювати значення параметрів Animator Controller;

– параметризовані анімації: анімації можуть бути параметризовані, щоб забезпечити різноманітність рухів залежно від введених параметрів.

Для анімації станів у доданих ассетах є спеціальні спрайти, з якими спочатку працюють через Timeline (рис. 3.24).

Рис. 3.23 – Ассет для анімації подвійного стрибка

Рис. 3.24 – Доданий ассет до Timeline зі встановленою швидкістю відтворення

Після додавання відповідних до станів ассетів були задані параметри для переходу між цими станами. Переходи пов’язують стани між собою і дають можливість корректно відтворювати рухи гравця (рис. 3.25).

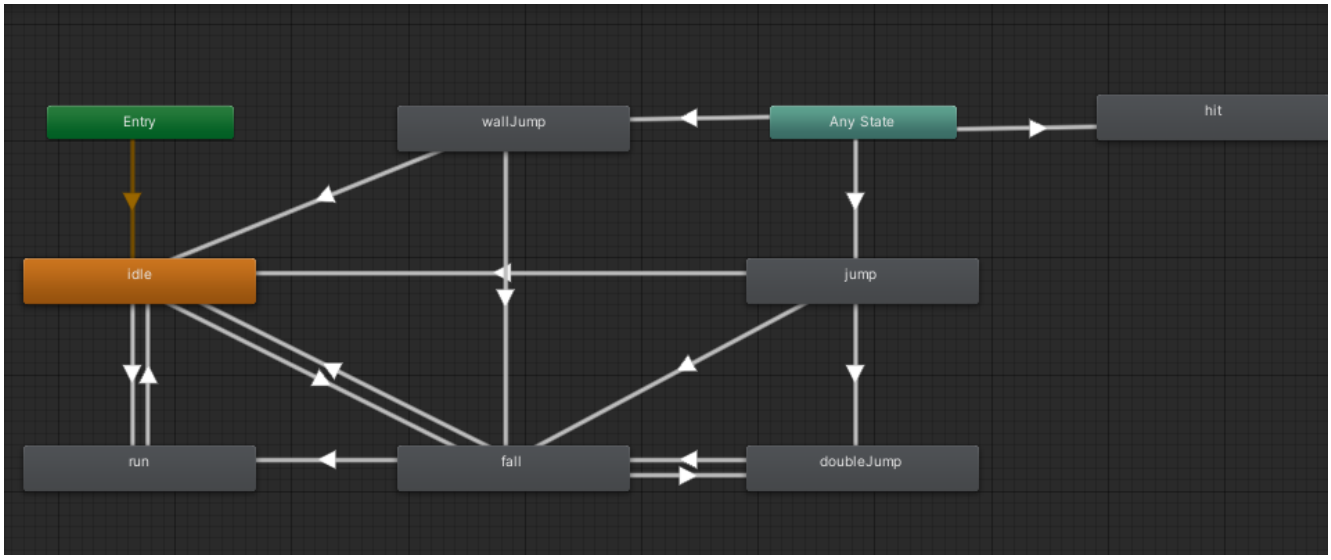

Рис. 3.25 – Зв’язки анімації механік гравця

Зв’язки представляють умови, за яких будуть відбуватись переходи. Кожен перехід має відповідний параметр, який додається до Animator (рис. 3.26).

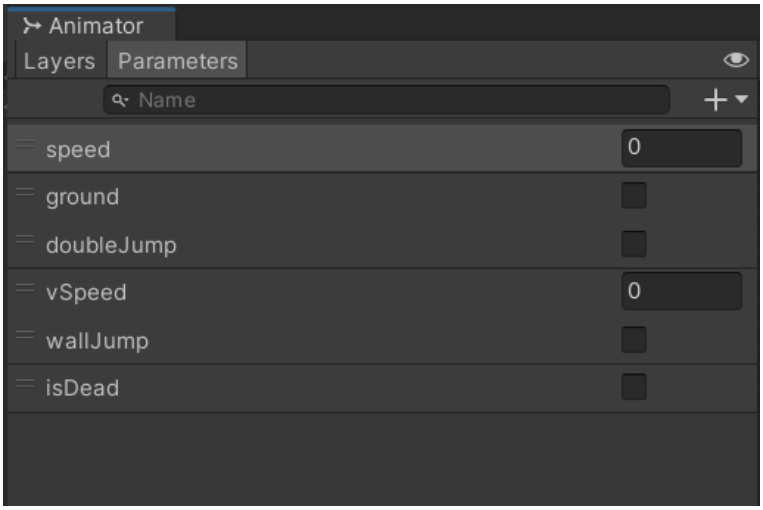

Рис. 3.26 – Параметри анімації механік гравця

З утворених зв’язків видно, що персонаж має анімацію навіть у стані спокою, з якого може перейти до анімації бігу, стрибка, падіння та бігу по стінам. Для цього відповідні параметри перевіряють виконання дій, за відсутності яких персонаж зберігає стан спокою і відтворює дану анімацію. Усі анімації підв’язані до персонажа (рис. 3.27).

Рис. 3.27 – Анімації механік гравця

З гравцем пов’язана камера, але, навідміну від багатьох платформерів, слідує за ним не постійно, а тільки при отриманні урона, коли виконується анімація смерті персонажа.

Вона представлена підлітанням модельки персонажа при зіткненні з ворожими об’єктами та вилітанням її за межі екрану. Для коректної роботи камери було створена окрема анімація (рис. 3.28).

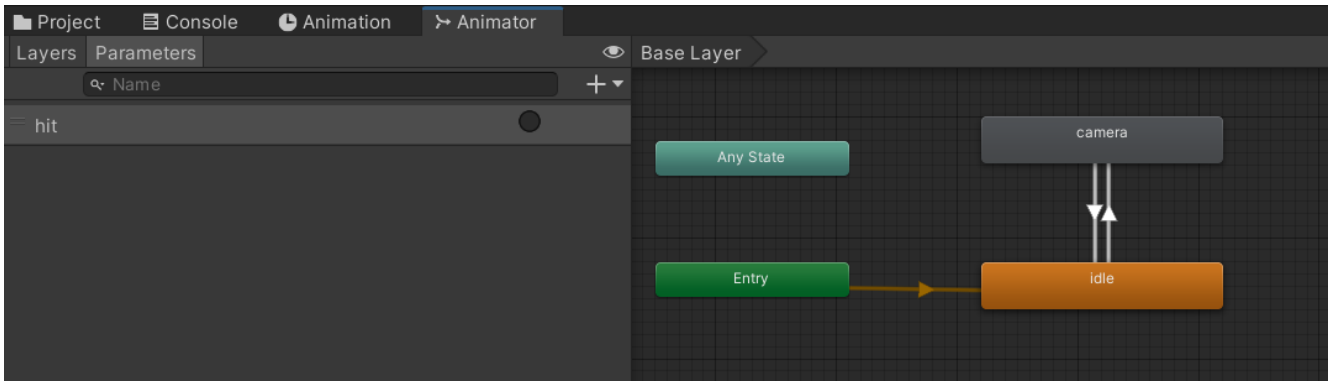

Рис. 3.28 – Анімація камери

Камера має лише один параметр виконання умов анімації у вигляді перевірки на отримання урону персонажем.

До механік гравця також відноситься збір фруктів, анімація яких складається зі стану спокою та взаємодії з персонажем (рис. 3.29).

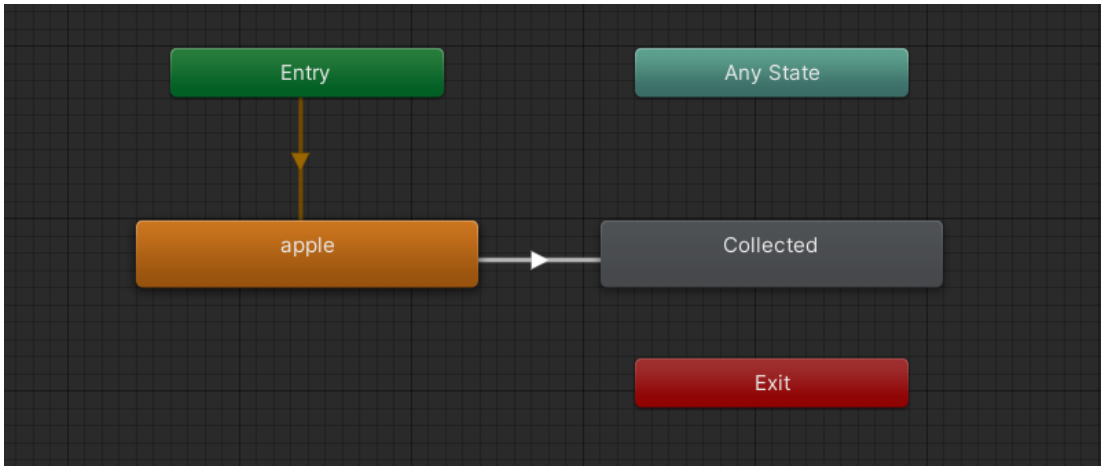

Рис. 3.29 – Анімація фруктів

Кожен фрукт має аналогічну логіку анімації, але повинен бути створений окремо, оскільки являється окремим об’єктом (рис. 3.30).

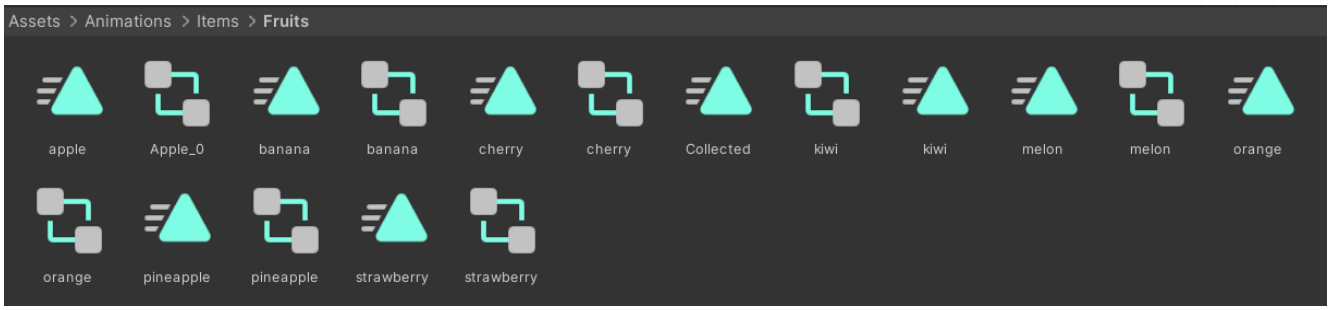

Рис. 3.30 – Анімації фруктів

Таким чином при контакті з гравцем буде зібран лише один фрукт, що зберігає цілісність анімації інших предметів.

3.2.3 Анімація платформ і пасток

Анімація платформ додає життєвості та динаміки грі. Для руху стрілки, що підкидає гравця, використовується параметр взаємодії з об’єктом, яка виконую умову переходу зі стану спокою у стан зникнення (рис 3.31).

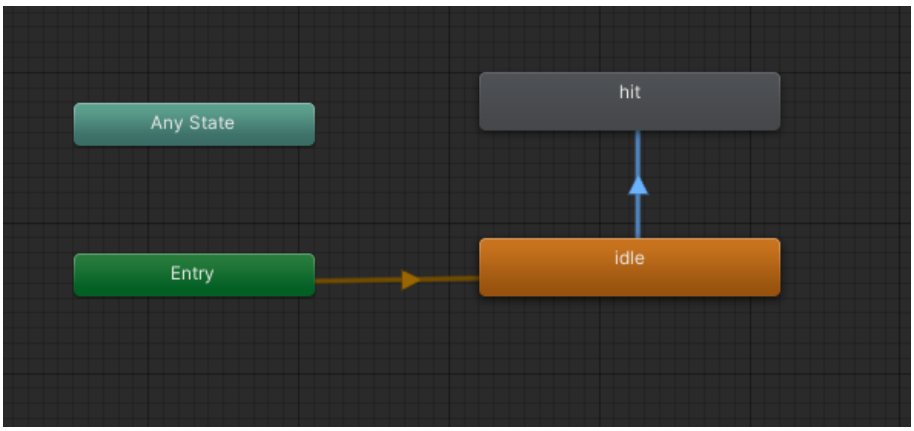

Рис. 3.31 – Анімація стрілки, яка підкидає гравця

Платформа що падає у стані спокою має анімацію зависання у повітрі та йде у стан падіння при виконанні умови взаємодії з персонажем (рис. 3.32).

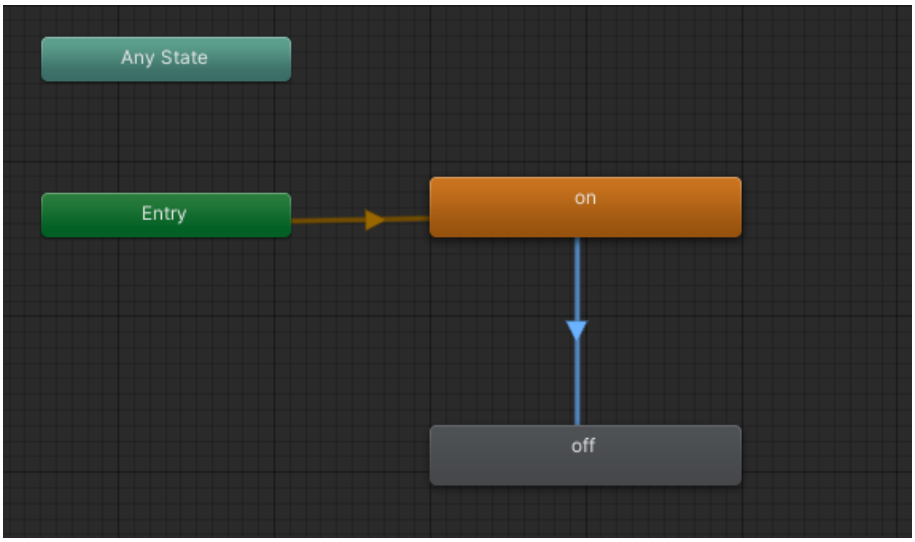

Рис. 3.32 – Анімація падіння платформи

Анімація трампліну має три стани: стан спокою, в якому знаходиться до взаємодії з гравцем, стан при взаємодії, коли гравець тримається в моменті натискання на платформу та має вистрибнути, та стан підкидання гравця, колий той покидає зону дії трампліну (рис. 3.33).

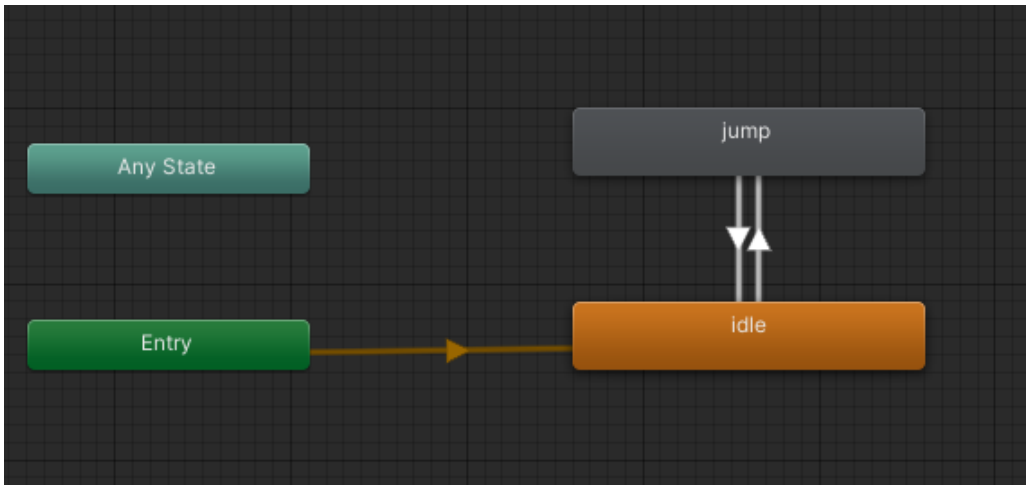

Рис. 3.33 – Анімація трампліну

Анімація пасток в платформері додає елементи виклику та стресу для гравця, роблячи гру більш цікавою. Вогняна пастка має статичний вигляд у стані спокою. При контакті з персонажем запускається анімація руху, який попереджає гравця про небезпеку та дає шанс втекти, після чого переходить у стан виверження полум’я та через декий час повертається у спокійний стан, якщо не була виконана умова смерті об’єкта (рис. 3.34).

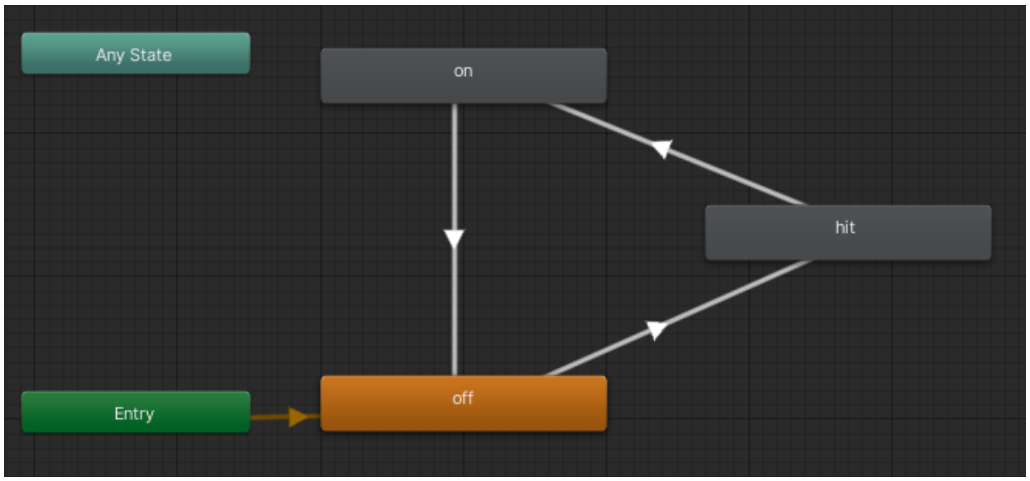

Рис. 3.34 – Анімація вогняної пастки

Пила знаходиться в постійному русі, тому має лише стан переміщення, який не змінюється ні при яких умовах (рис. 3.35).

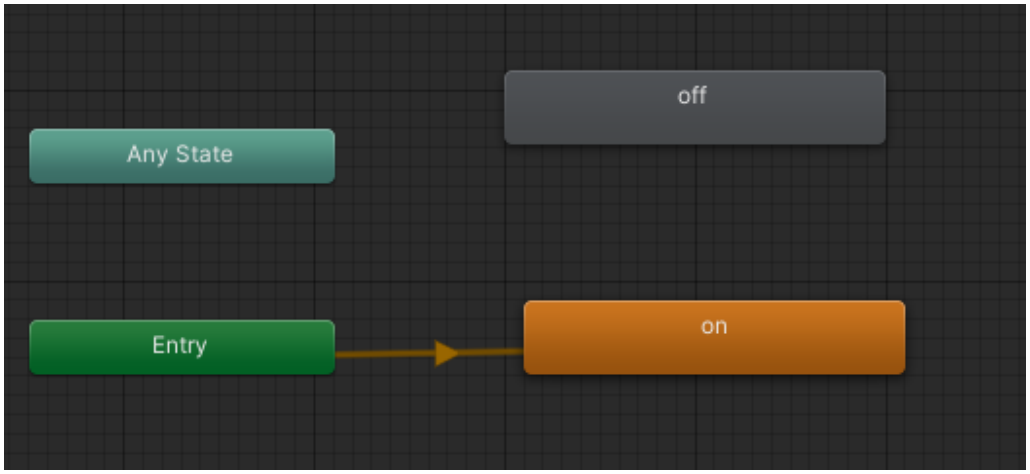

Рис. 3.35 – Анімація руху пили

Спрайт шипів виглядає як трикутник, через що виникає складність в анімації: потрібно зчитувати напрямок з якого гравець взаємодіє з пасткою для коректного відображення. Самі по собі шипи статичні і не змінюють свій стан, але мають умови взаємодії на гравця, які виклакають його смерть (рис. 3.36).

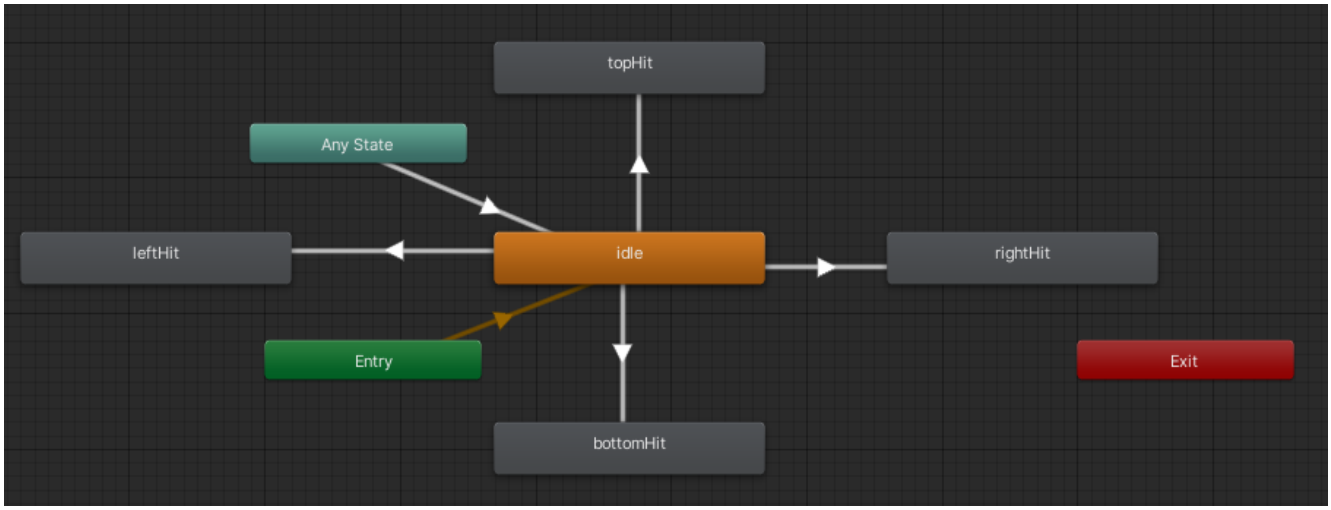

Рис. 3.36 – Анімація шипів

Умовно основні напрямки прийнято за стани об’єкта, що допомагає ввести необхідні параметри для взаємодії з гравцем. Це введено для того, щоб гравець не помирав від невидимих текстур на рівнях.

3.2.4 Анімація ворогів

Анімація ворогів у платформерах є важливим аспектом для створення живого та цікавого ігрового середовища. Використані підходи до анімації ворогів:

– базові анімації руху: створіно анімації для базового руху ворогів, такі як ходьба, біг або літання, в залежності від характеристик ворога.

– атака: додано анімації для різних видів атак. Це може включати удари, кидки чи інші прийоми залежно від типу ворога.

– анімація ушкодження та смерті: визначено анімації для ворогів при ушкодженні та смерті.

– перехідні анімації: забезпечено плавні переходи між різними анімаціями, щоб рух ворога виглядав природно та плавно.

– збільшення реакції: додано анімації реакції ворогів на події, такі як сповільнення під важким ударом чи виявлення гравця.

– патрулювання та спостереження: створено анімації для руху ворога під час патрулювання або спостереження за гравцем.

Анімація птаха схожа на анімацію пастки типу пила, що використовую підход базового руху та патралювання, тобто об’єкт літає в одній горизонталі. Але навідміну від пили має анімацію смерті, яка запускається при виконанні умови стрибку гравця на птаха (рис. 3.37).

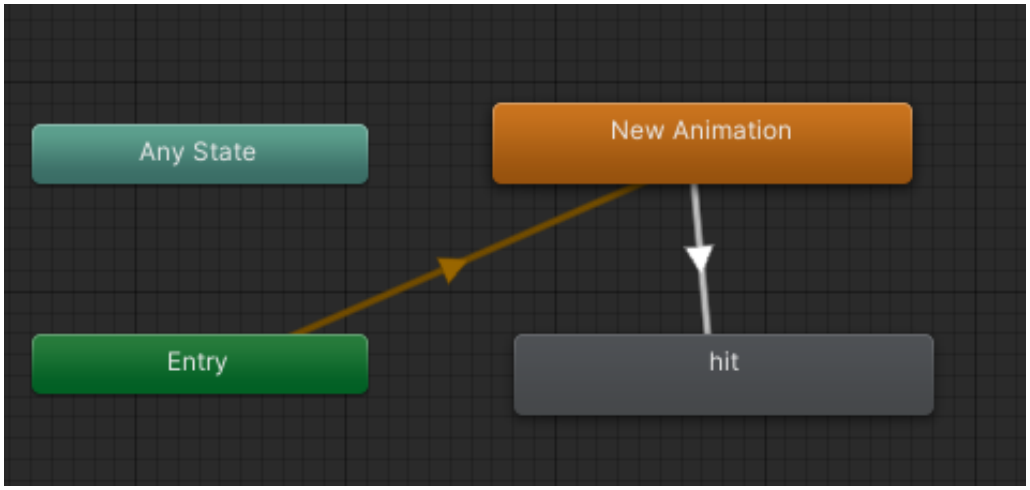

Рис. 3.37 – Анімація птаха

Слайм має дві складові: головний об’єкт, тобто його модель, та частинки слайма, які залишаються після нього. Сам слайм має аналогічний принцип та підходи в анімації як і птах (рис. 3.38).

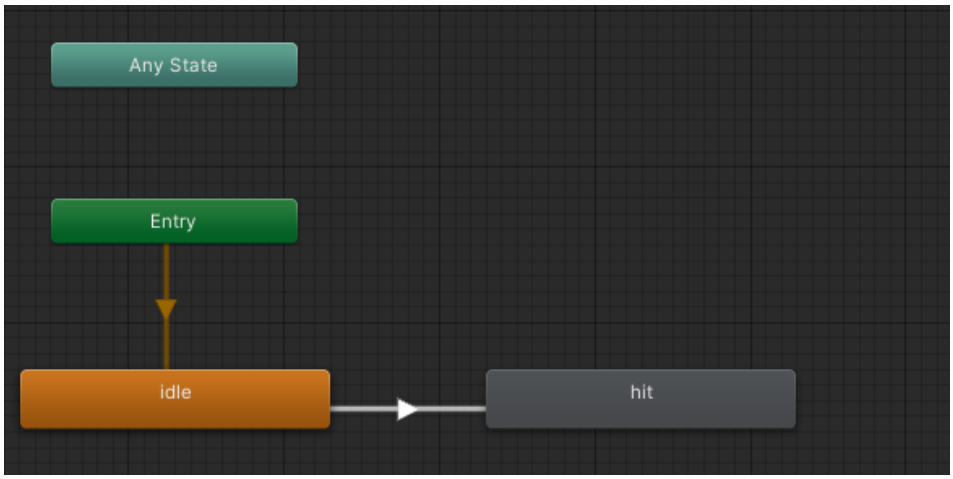

Рис. 3.38 – Анімація слайма

Частинки слайма мають стан спокою та стан зникання, як викликається умовою взаємодії з об’єктом або з плином часом, як вказано у коді (рис. 3.39).

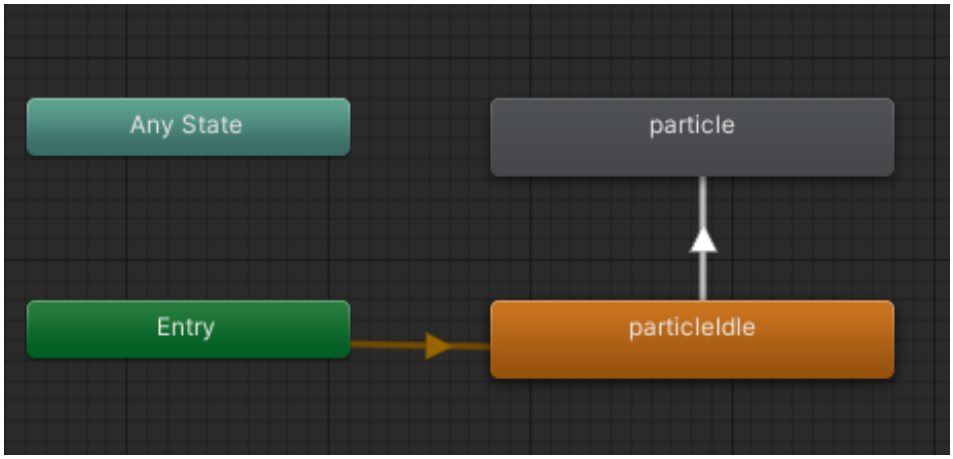

Рис. 3.39 – Анімація частинок слайма

Носоріг має всі вищесказані підходи анімації. До базових анімацій руху відноситься стан спокою, в якому об’єкт відтворює анімацію очікування. У стан атаки носоріг переходить, коли помічає гравця та починає рухатись в його напрямку з деяким прискоренням. При зіткненні цих об’єктів виконується умова смерті гравця, якщо персонаж на траєкторії руху ворога зникає (підстрибує), то

виконується умова зіткнення з іншою перешкодою, що запускає відповідну анімацію. Якщо носоріг все ще бачить гравця, виконується анімація повороту об’єкта в напрямку персонажа та попередній алгоритм поновлюється. Якщо гравець зник, об’єкт повертається в стан спокою.

Крім того при взаємодії гравця з носорогом через стрибок на ворога виконується умова смерті противника та запускається відповідна анімація (рис. 3.40).

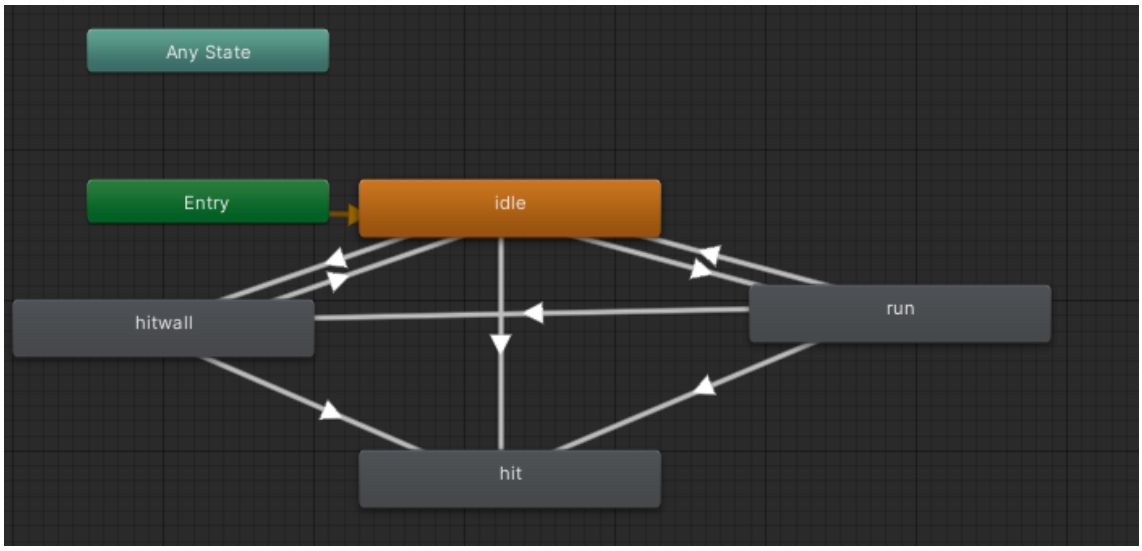

Рис. 3.40 – Анімація носорога

Важливо враховувати, що анімації повинні бути збалансованими та взаємодіяти з геймплеєм, роблячи ворогів не лише візуально привабливими, але й цікавими для гравця.

3.3 Звукове оформлення

Звуковий дизайн в платформних іграх відіграє важливу роль у створенні атмосфери та взаємодії гравців:

– фонова музика: вибрати або створити музику, яка відповідає настрою гри. Динамічні й енергійні пісні додають іграм азарту, а тихішу музику можна використовувати під час менш цікавих рівнів або роликів;

– звукове супроводження дії гравця: додати звуки для кроків, стрибків та інших дій гравця;

– навколишні звуки: додати у гру навколишні звуки, такі як звуки води, птахів, вітряків та інші звуки, що підходять для певного рівня чи місця;

– звуки ворогів і перешкод: кожен ворог може мати унікальні звуки атаки, пошкодження або руху. Це також стосується перешкод і пасток;

– звукові ефекти для взаємодії з об'єктами: додати звукові ефекти під час взаємодії з такими об’єктами, як кнопки гравця;

– звук анімації: забезпечити звуковий дизайн для різних анімацій, таких як атаки, стрибки, зіткнення та інші дії гравця.

Для керування звуком була створена звукова панель з можливістю налаштовувати гучність , підбирати трек і т.д (рис. 3.41).

```
void Awake()
{
    DontDestroyOnLoad(gameObject);
    if (go == null)
        go = gameObject;
    else
        Destroy(gameObject);
    foreach (Sound s in sounds)
    {
        s.source = gameObject.AddComponent<AudioSource>();
        s.source.clip = s.clip;
        s.source.volume = s.volume;
        s.source.loop = s.loop;
    }
}

Сообщение Unity | Ссылок: 0
private void FixedUpdate()
{
    if(SceneManager.GetActiveScene().buildIndex == 0 && !isMainScene)
    {
        Play("MainSceneMusic");
        Stop("SoundTrack");
        isMainScene = true;
    } else if(SceneManager.GetActiveScene().buildIndex != 0 && (isMainScene || isPause) )
    {
        Play("SoundTrack");
        Stop("MainSceneMusic");
        isMainScene = false;
        isPause = false;
    }
}
```

Рисунок 3.41 – Звукова панель

Код додає об'єкт до списку об'єктів, які не будуть знищені при зміні сцени. Це дозволяє тримати відтворювач звуків, активним між сценами гри.

Створений компонент AudioSource встановлює гучність звуку згідно з параметром s.volume у звуковому об'єкті та встановлює параметр loop для повторення звуку згідно з параметром s.loop у звуковому об'єкті.

Отже, загальна мета цього коду полягає в створенні та конфігурації компонентів AudioSource для кожного звукового об'єкта в масиві sounds, а також у забезпеченні того, що цей об'єкт не буде знищений при зміні сцени, і що у грі може бути лише один екземпляр цього об'єкта (рис. 3.42).

Далі код контролює відтворення звукових треків в залежності від індексу активної сцени і стану прапорців isMainScene та isPause. В залежності від умов відтворюється різний звуковий трек, і зупиняється інший.

Рис. 3.42 – Audio Manager

Audio Manager дозволяє додавати та міняти треки не вікористовуючи зміни у коді, та дає можливість змінити гучність та параметри повторного відтворення звуків.

Самі звуки поділяються на звукове супроводження анімації та фонову музику. Звукове супроводження створено за допомогою безкоштовших фонотек з можливістю знаходити та завантажувати необхідні звуки (рис. 3.43).

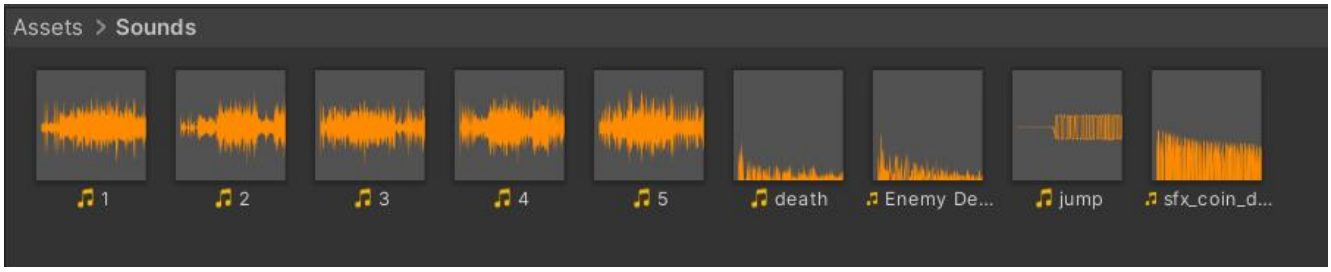

Рис. 3.43 – Бібліотека звуків

При створенні фонової музики важливо врахувати стиль та атмосферу – піксельна ретро-гра. Мелодія має бути динамічною і підходити під темп проходження рівнів гравцями.

Для створення мелодій для гри було використано генератор музики, який використовує штучний інтелект, базуються на великому обсязі даних звукових записів, щоб вивчити стилі, мелодії та інші характеристики музики (рис. 3.44).

Принцип роботи полягає у пошуці музики через вибрані теги, що відповідають за жанр, настрій, музикальний інструмент і т.д Після чого можна почати роботу з самою композицією.

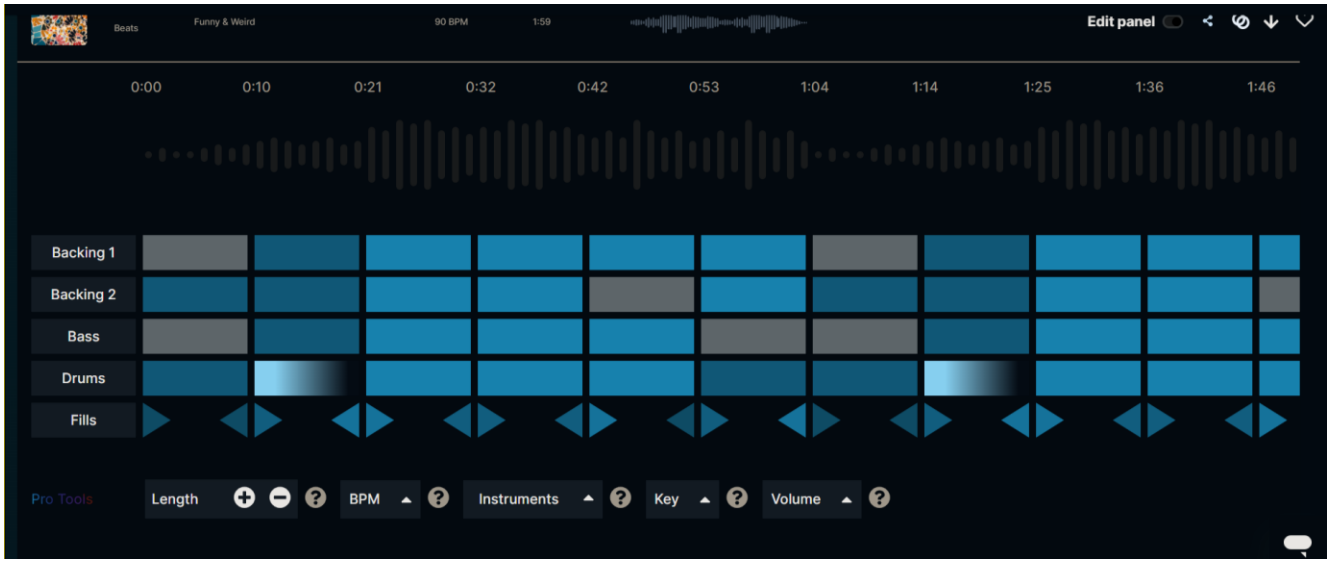

Рис. 3.44 – Робота з генератором музики

Для потрібного звучання можна самостійно налаштовувати компоненти (рис. 3.45), які утворюють музичний твір:

– мелодія: базовий музичний елемент, що складається із серії звуків, що повторює основну музичну думку пісні;

– гармонія: акорди та звуки, які супроводжують мелодію та надають пісні глибини та характеру.

– ритм: система пульсацій і часу, яка визначає темп і рух пісні. Ритм можна створити за допомогою ударних інструментів та інших елементів ритму.

– басова лінія: низькочастотний звук, який забезпечує стабільність і основу треку. Бас можна грати на бас-гітарі, контрабасі, синтезаторі чи іншому інструменті.

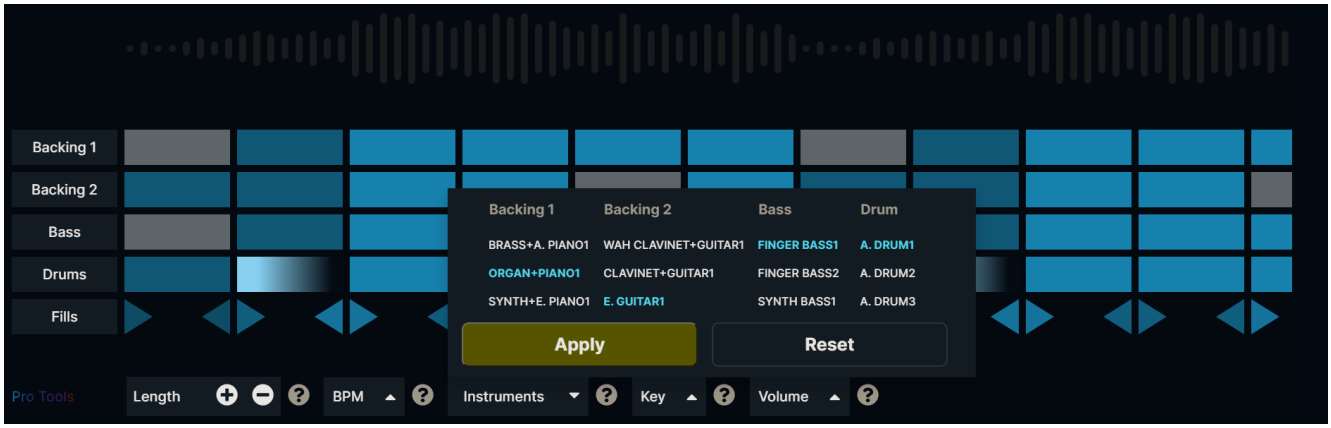

Рис. 3.45 – Робота з компонентами музичного твору

Створивши декілька треків, їх біло оброблено в Audacity – вільному та відкритому аудіоредактору, який дозволяє записувати та редагувати звукові файли на різних платформах. В програмі є велика кількість вбудованих ефектів для поліпшення та модифікації звуку.

Audacity є популярним інструментом серед аудіоінженерів, музикантів та користувачів, які потребують простого у використанні аудіоредактора з розширеними можливостями (рис. 3.46).

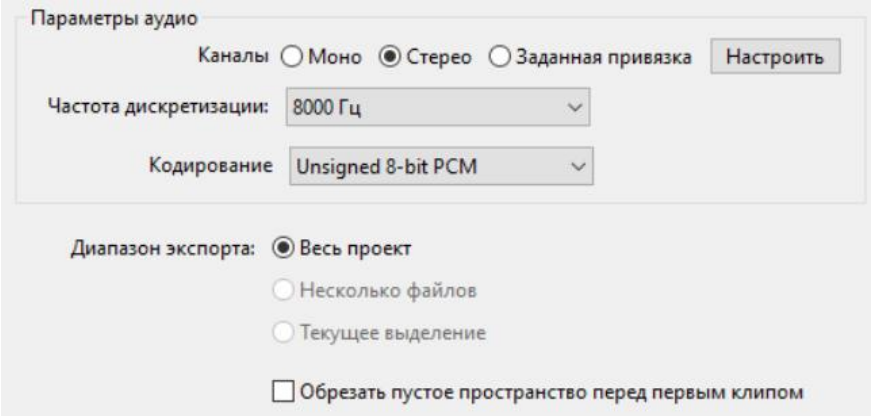

Рис. 3.46 – Експорт обробленого треку в Audacity

Експортувавши трек таким чином, було отримано звучання фонової музики як на старій приставці Sega, що відповідає атмосфері гри та поставленій цілі. Усі треки додані до Unity та грають у встановленому порядку.

3.4 Розробка інтерфейсу користувача

Головне меню встановлює перше враження та надає гравцеві можливість керувати основними параметрами гри (рис. 3.47).

Рис. 3.47 – Головне меню гри

Меню містить назву ігрового додатку, кнопку старту, вибір рівнів та налаштування. Старт: дозволяє гравцеві почати нову гру або вибрати рівень, а

також має опцію продовження гри. Мапа рівнів показує список доступних рівнів, та непройдені рівні, що мають їх зображення та розблокованість (рис. 3.48).

Рис. 3.48 – Вибір рівнів

Налаштування мають опцію зміни гучності музики та звуку і можливість перезапуски гри з нуля (рис.3.49).

Рис. 3.49 – Налаштування

У самій грі є можливість поставити гри на паузу та повернутись у головне меню. Для кнопок управління рухом персонажа забезпечено достатній розмір кнопок, щоб гравці могли легко натискати їх під час гри на мобільних пристроях або планшетах (рис. 3.50).

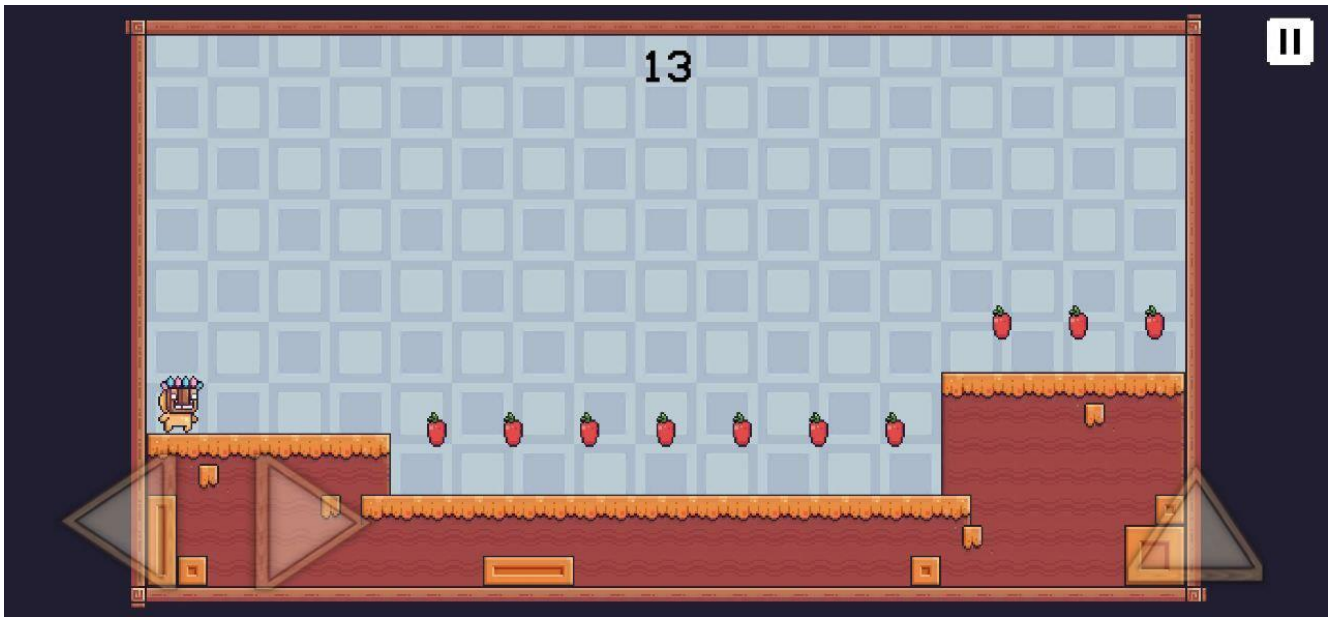

Рис. 3.50 – Скріншот гри

Завершення та перезапуск рівнів супроводжуються анімацією зустрічі гравця та виступає у ролі переходу між двома сценами (рис.3.51).

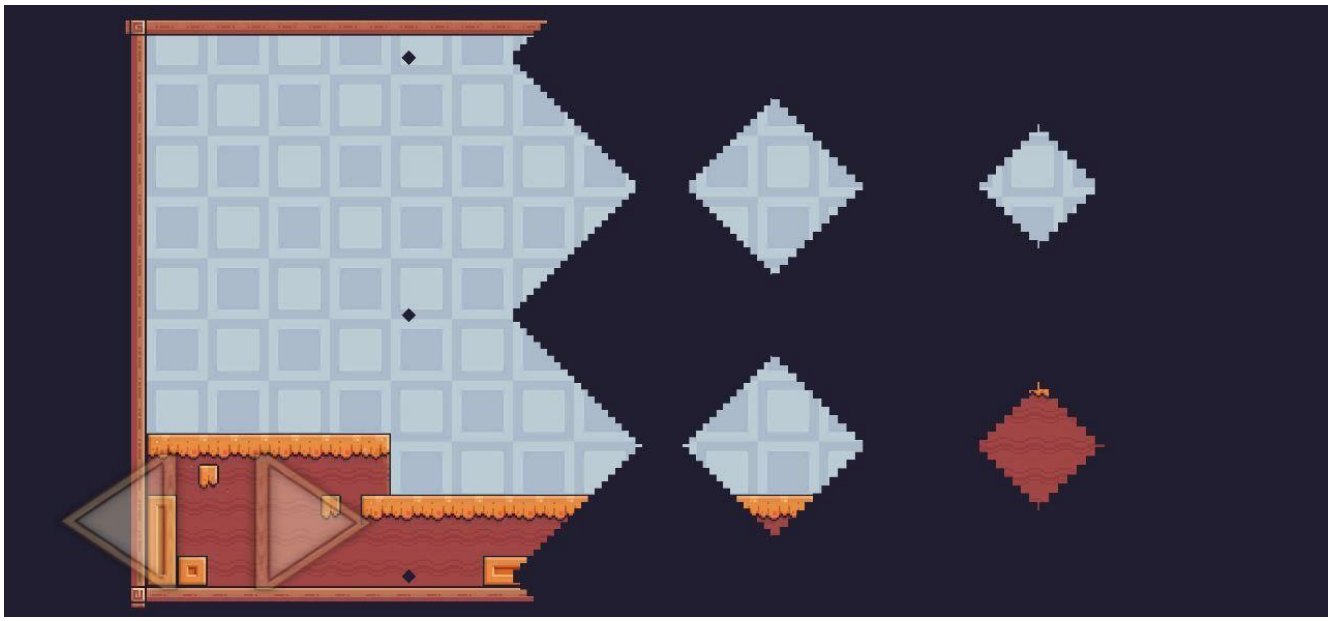

Рис. 3.51 – Анімація завершення/перезапуску рівня

У цьому випадку анімація використовується для виділення важливих подій або досягнень у вигляді проходження рівня.

4 ТЕСТУВАННЯ ТА ВІДЛАДКА

4.1 Тестування функціональності мобільного ігрового додатку

Тестування мобільних ігор — це унікальний процес, який враховує дизайн гри, інтерфейс користувача та деякі технічні аспекти, які відрізняються від традиційного тестування додатків [24].

Функціональне тестування є важливим кроком у розробці програмного забезпечення. Головна мета — переконатися, що програма відповідає заданим вимогам і працює коректно. Для функціонального тестування було проведено:

– перевірка введених даних: Перевірено точність і схвалення даних, введених користувачем. Виявлено помилки у вхідних даних і надано чіткі повідомлення про помилки користувачам;

– функціональний тест: Перевірено роботу окремих функцій програми. Виконано тестування на точність і придатність;

– перевірка сценарію: Тестові скрипти використовуються для перевірки взаємодії різних функцій і компонентів програми;

– тестування інтерфейсу: Перевірено, що інтерфейс добре представлений і інтуїтивно зрозумілий для користувачів. Скеровано взаємодією користувача через графічний інтерфейс.

Взято до уваги різноманіття сценаріїв використання та контекстів, у яких може використовуватися програма, для забезпечення її найвищої якості та функціональності.

Проведено аналіз тестування мобільних ігор – процес оцінки ефективності тестування, виявлення проблем і вдосконалення стратегій тестування для надання високоякісних мобільних ігорових додатків.

Оцінено ігрові аспекти, охоплені тестом, включаючи різні ігрові режими, рівні складності, операційні системи та пристрої. Проаналізовано різні сценарії використання гри та враховані під час тестування (табл. 4.1).

Табл. 4.1 – Тестування функціональності

Тестування	Очікуваний результат	Отриманий результат
Запуск та перезавантаження рівнів	При натисканні кнопок старту та вибору рівнів запускається гра, при перезапуску гра починається спочатку	Відповідає реальності
Управлінням персонажу	Персонаж коректно відгукується на всі дії викликані гравцем	Відповідає реальності
Отримання урону від ворогів та пасток	При контакті з ворогом чи пасткою гравець отримує урон та помирає, рівень перезапускається	Відповідає реальності
Нанесення урону ворогам	Стрибнуши на ворогів зверху, ті отримують урон та помирають	Відповідає реальності
Збирання фруктів	При контакті з фруктами вони збираються, зібравши всі фрукти на рівні починається новий	Відповідає реальності

Проведено тестування гравців для мобільного ігрового додатку, також відоме як «ігрове тестування» або «бета-тестування». Участь у тестуванні гравців дозволяє розробникам збирати відгуки, виявляти помилки та покращувати гру до загального випуску.

Створено механізм збору детальних відгуків від гравців про ігровий процес, графіку, звук та інші аспекти гри – зручний канал спілкування між розробниками та гравцями для обговорення питань та ідей.

Перевірено мобільну гру на реальних сценаріях використання на різних мобільних пристроях і операційних системах.

Гравці надсилали коментарі, пропозиції та скарги щодо гри, вели обговорення гри та обміну досвідом.

Бета-гравці є експертами з кінцевого використання, і їхні відгуки можуть значно покращити якість мобільного ігрового додатку до офіційного випуску.

Для тестування було відібрано 20 гравців, на основі яких було створено діаграму скарг (рис. 4.1).

Рис. 4.1 – Основні скарги гравців

На основі основних скарг гравців при тестуванні функціональності мобільного ігрового додатку можно розробити план з виправлення помилок та недоробок.

4.2 Виправлення помилок і недоробок

Виправлення помилок і недоробок в мобільних іграх — це важлива частина розробки гри, оскільки це може значно вплинути на користувачів і загальний успіх гри [25].

Найбільша кількість гравців вказало на різку зміну складності гри. Початково більш складні рівні створювались з метою демонстрації творчих ігрових можливостей та потенціалу розвитку складності, тому єдиним виправленням була зміна порядку відтворення рівнів 5 та 6.

Складним серед гравців вважався 5й рівень через ворога типу слайм, а точніше частинок слайма, тому було виправлено тривалість їх життя, тепер вони зникають швидше. Також для цього рівня було додано час проходження через таймер.

Через невелику кількість рівнів у мобільному ігровому додатку вважалось що є можливість в майбутньому додаванні нових, тому був відсутній фінальний

екран. Це було виправлено через автоматичне перекидання гравців після проходження останнього наявного рівня.

Початкова версія мобільного ігрового додатку мала не дуже чіткі хітбокси, які були виправлені: збільшено розмір хітбоксів ворогів, для можливості вбивства гравцем противників та зменшено хітбокси пасток типу пила та шипи для коректної взаємодії з гравцем та його вбивства.

Після тестування була виявлена проблема у недостатньо великим кнопкам управління, яку наразі виправлено через збільшення розмір відображення спрайтів.

На деяких рівнях гравцям не вистачало часу на проходження всіх пасток та збір фруктів, тому було проаналізовано середньостатистичний час на завершення цих рівнів та прораховано новий, на який було змінено початкові значення внутрішньоігрового таймера.

Таким чином було виправлено помилки та недоробки, які були виявлені при тестуванні функціональності мобільного ігрового додатку іншими гравцями.

Зосередження на цих аспектах може допомогти покращити якість мобільного ігрового додатку і забезпечити задоволення користувачів.

5 ОЦІНКА ДОСВІДУ КОРИСТУВАЧА

5.1 Збір і аналіз відгуків користувачів

Збір і аналіз відгуків користувачів мобільного ігрового додатку може надзвичайно допомогти розробникам зрозуміти сильні та слабкі сторони гри, а також виявити можливості для вдосконалення [26].

Для збору відгуків більшість користувачів віддає перевагу соціальним мережам, тому їх було визначено як основну платформу для пілкування з гравцями.

Збір та аналіз відгуків відбувався через користувачів, які приймали участь у тестуванні мобільного додатку, тому мають аргументовану думку (рис. 5.1).

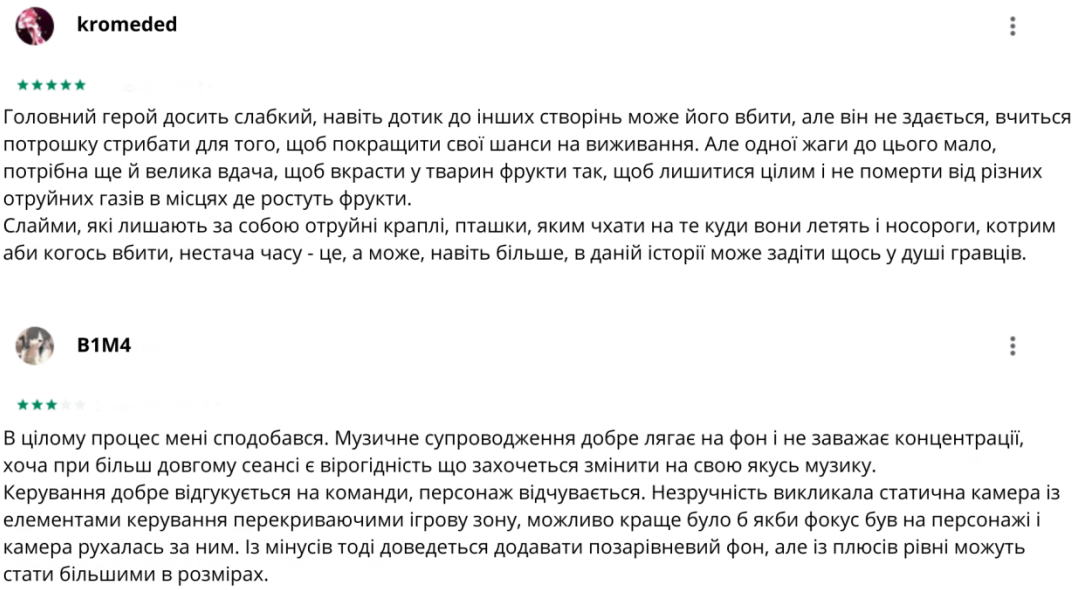

Рис. 5.1 – Збір відгуків користувачів

Для аналізу зібранних даних відгуки були розбиті на різні категорії, що допомогло визначити основні позитивні та негативні аспекти. Загальний підхід до збору та аналізу відгуків був систематичним і спрямованим на поліпшення якості продукту на основі відгуків користувачів.

Також самі користувачі мали можливість пропунувати нові елементи, механіки та покращення. Більшість гравців прийшли до таких рішень:

– обліковий запис: можливість мати обліковий запис, що зберігав би прогрес та яким можна було б поділитись з іншими гравцями;

– додаткові життя: для швидкого та легшого проходження додати додаткові життя, які дають більше однієї спроби проходження рівнів;

– внутрішньоігрова валюта: валюта, яка дасть можливість придбати покращення для персонажу та надасть більшого різноманіття в механіках гри;

– бонуси: персонаж може отримувати, накопичувати чи підбирати на рівнях різні довго чи короткострокові бонуси, які урізноманітнять геймплей.

Важливо не лише збирати відгуки, але й вживати заходи для поліпшення якості мобільного ігорового додатку, враховуючи отриману інформацію від користувачів.

5.2 Оцінка задоволення гравців

Оцінка задоволеності клієнтів (CSAT) — це показник, який використовується для вимірювання того, наскільки користувач задоволений продуктом, послугою або загальною взаємодією з певною компанією. Основна ідея полягає в тому, щоб дати користувачам можливість висловити своє задоволення на основі конкретних питань.

CSAT розраховується як відсоток задоволених клієнтів серед усіх відповідачів за формулою: $CSAT = \left(\frac{\text{Кількість задоволених відповдей}}{\text{Загальна кількість відповідей}}\right) * 100\%$.

Для розрахунку було проаналізовано отримані оцінки користувачів по шкалі від 1 до 10, де 1 – повністю незадовільно, а 10 – повністю задовільно.

Оскільки для розробка мобільного ігрового додатку входить ряд основних компонентів, які об'єднуються для створення цілісного ігрового досвіду, користувачі оцінювали ці параметри окремо. Таким чином в оцінку задоволення користувача входять дизайн персонажів та рівнів, їх анімація, звукове оформлення та музика, інтерфейс користувача, управління головним персонажем та ігрові механіки (рис. 5.2).

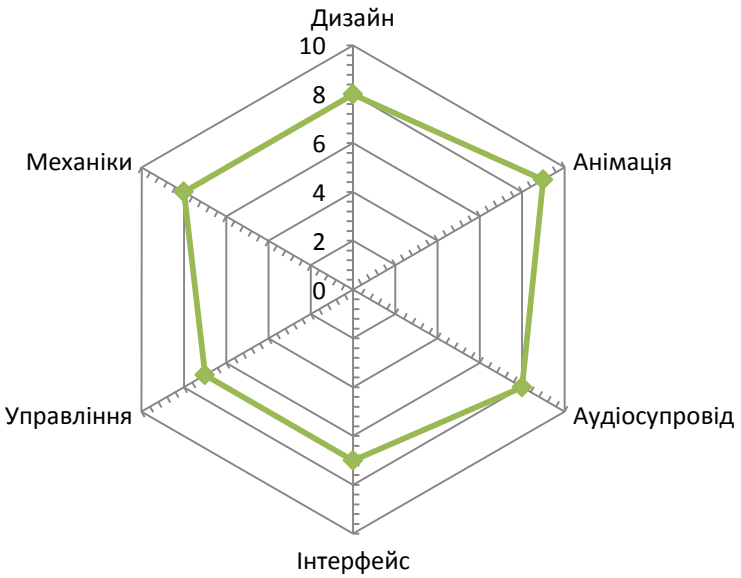

Рис. 5.2 – Оцінка задоволення користувача за основними компонентами гри

З діаграми видно, що гравцям, які прийняли участь у тестуванні мобільного ігрового додатку, найбільше сподобались дизайн персонажів та рівнів, анімація та звуковий супровід. Це означає, що візуальна складова гри виконана в єдиному стилі, добре поєднується між собою та не відволікає гравця під час проходження.

Загальне враження від гри оцінюється у 7.8 балів з 10 можливих, тобто продукт має багато переваг, які вразили користувача. Це включає високу якість виконання, ефективність, зручність у використанні або інші позитивні аспекти, які роблять його привабливим для користувачів.

Хоча продукт вражає своїми перевагами, є кілька областей, в яких можна вдосконалити його: управління та дизайн інтерфейсу користувача. Після проведеного аналізу з відгуків гравців щодо управління можна сказати, що найкращим рішенням буде забезпечити можливість налаштовувати розташування та розміри елементів управління самостійно. Для покращення роботи інтерфейсу можна додати короткі туторіали або підказки, які допоможуть гравцям зрозуміти функції інтерфейсу.

ВИСНОВКИ

Робота присвячена розробці мобільного ігрового додатку у жанрі платформеру на операційній системі Android. Було проаналізовано існуючі рішення конкуруючих мобільних ігрових додатків, з яких була сформована основна ідея та механіки для гри.

У даній роботі було розглянуто історію розвитку мобільних ігрових додатків, аналіз поточного стану ринку, технології та платформи для розробки мобільних ігор та основні принципи і концепції їх дизайну. На основі цих досліджень була визначена цільова аудиторія та розроблен контент гри у стилістиці ретро 2D. Було створено макет майбутнього продукту та розроблено план по його створенню.

Процесс розробки мобільного ігрового додатку складався зі створення механіки гравця, пасток та ворогів, роботи над графікою та анімацією, підбору звукового оформлення та розробки інтерфейсу користувача. Рівні створювались з урахуванням поступового зростання скадності, притримуючись стратегії, яку використовують розробники для того, щоб забезпечити гравцям відчуття постійного виклику і зацікавленості. Такий підхід дозволяє плавно залучати гравців, розвиваючи їх навички та знання гри.

Для виявлення та виправлення помилок було проведено тестування функціональності із залученням звичайних користувачів, за допомогою якого пропрацьовано такі елементи як розмір кнопок управління, хітбокси, час на проходження рівнів, дизайн інтерфейсу та зміна складності.

На підставі проведеного тестування було зібрано та проаналізовано відгуки користувачів та прорахована оцінка їх задоволення мобільним ігровим додатком, яка складає 7,8/10 балів, що свідчить про високий рівень виконання з невеликою кількістю недоліків та можливістю поліпшення, до якої входять пропозиції самих користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Juul, J. A Casual Revolution: Reinventing Video Games and Their Players. Cambridge, Massachusetts: MIT Press, 2009. – 272 p.
2. Koster R. Theory of Game Development and Entertainment. 2018. 288 p.
3. Gregory J. Game Engine. Programming and Internal Structure. Peter Press, 2021. 1136 p.
4. Bogost, I. Persuasive Games: The Expressive Power of Videogames. Cambridge, Massachusetts: MIT Press, 2007. – 432 p.
5. Egenfeldt-Nielsen, S., Smith, J. H., & Tosca, S. Understanding Video Games: The Essential Introduction. New York: Routledge, 2008. – 272 p.
6. Fullerton, T., Swain, C., & Hoffman, S. Game Design Workshop: A Playcentric Approach to Creating Innovative Games. Burlington, Massachusetts: Morgan Kaufmann, 2014. – 532 p.
7. Gee, J. P. What Video Games Have to Teach Us About Learning and Literacy. New York: Palgrave Macmillan, 2003. – 224 p.
8. Jenkins, H. Convergence Culture: Where Old and New Media Collide. New York: New York University Press, 2006. – 308 p.
9. Crawford, C. Chris Crawford on Game Design. Berkeley, California: New Riders Press, 2003. – 492 p.
10. Rouse III, R. Game Design Theory and Practice. Plano, Texas: Wordware Publishing Inc, 2005. – 723 p.
11. Salen, K., & Zimmerman, E. Rules of Play: Game Design Fundamentals. Cambridge, Massachusetts: MIT Press, 2004. – 688 p.
12. Miller, M., Paige, N., Clair, G. & Eckhardt, C. An Analysis of Peer Presence Social Group Dynamics to Enhance Player Engagement in Multiplayer Games, 2019. – 8 p.
13. Rollings, A. & Ernst Adams. Fundamentals of Game Design. New Jersey: Pearson Prentice Hall, 2007. – 66 p.

14. Salen, K., & Zimmerman, E. Rules of Play: Game Design Fundamentals. Cambridge, Massachusetts: MIT Press, 2004. – 688 p.
15. Schreiber, I., & Adams, E. Challenges for Game Designers: Non-Digital Exercises for Video Game Designers. Boston, Massachusetts: Course Technology PTR, 2008. – 352 p.
16. Werbach, K., & Hunter, D. For the Win: How Game Thinking Can Revolutionize Your Business. Philadelphia, Pennsylvania: Wharton Digital Press, 2012. – 288 p.
17. Zimmerman, E., & Salen, K. The Game Design Reader: A Rules of Play Anthology. Cambridge, Massachusetts: MIT Press, 2006. – 976 p.
18. Norman, D. The Design of Everyday Things. New York: Basic Books, 2002. – 369 с.
19. McCarthy, J. & Peter Wright. Technology as Experience. Cambridge, Massachusetts: The MIT Press, 2006. – 226 с.
20. Järvinen, A. Games without Frontiers: Theories and Methods for Game Studies and Design. Tampere: Tampere University Press, 2008. – 417 с.
21. Bruner, J. S., Jolly, A. and Sylva, K., Play. Its role in development and evolution. Penguin Books, New York, 1976. – 1161 с.
22. Avedon, E. M. The Structural Elements of Games. New York, 1971. – 419 с.
23. Bateman, C. & Richard Boon. XXI Century Game Design. Hingham, Massachusetts: Charles River Media, 2006. – 332 с.
24. Rollings, A. & Ernst Adams. Fundamentals of Game Design. New Jersey: Pearson Prentice Hall, 2007. – 66 с.
25. Salen, K. & Eric Zimmerman. Rules of Play. Game Design Fundamentals. Cambridge, Massachusetts: The MIT Press, 2007. – 37 с.
26. Fullerton, T. & Christopher Swain & Steven Hoffman . Game Design Workshop. Designing, Prototyping and Playtesting Games. San Francisco: CMP Books, 2004. – 480 с.