Міністерство освіти і науки України
Тернопільський національний педагогічний університет
імені Володимира Гнатюка

Фізико-математичний факультет
Кафедра інформатики та методики її навчання

Кваліфікаційна робота

ПРОЄКТУВАННЯ ТА РОЗРОБКА 2D АРКАДНОЇ ГРИ «APPLE MADNESS» ЗАСОБАМИ GODOT ENGINE

Спеціальність 122 Комп'ютерні науки

Освітня програма «Інженерія ігрових проектів»

Здобувача вищої освіти освітньо-кваліфікаційного рівня «бакалавр»
Пузича Артура Сергійовича

НАУКОВИЙ КЕРІВНИК :
кандидат педагогічних наук,
доцент кафедри інформатики та
методики її навчання
Лещук Світлана Олексіївна

РЕЦЕНЗЕНТ:
декан факультету комп'ютерно-
інформаційних систем і програмної інженерії,
кандидат технічних наук, доцент
Баран Ігор Олегович

Тернопіль – 2026

ЗМІСТ

ВСТУП

Комп’ютерні ігри є однією з найпоширеніших форм цифрових розваг у сучасному світі. З розвитком інформаційних технологій ігрова індустрія перетворилася на окрему галузь цифрового мистецтва та програмної інженерії, яка поєднує графічний дизайн, анімацію, програмування, звуковий супровід і проєктування інтерактивних систем. Особливу популярність серед користувачів здобули 2D аркадні ігри, що вирізняються простими правилами, динамічним ігровим процесом та доступністю для широкої аудиторії.

Сучасні ігрові рушії значно спростили процес створення комп’ютерних ігор, надаючи розробникам готові інструменти для роботи з графікою, фізикою, анімацією та логікою взаємодії об’єктів. Одним із таких рушіїв є Godot Engine — безкоштовне середовище розробки з відкритим програмним кодом, яке активно використовується незалежними розробниками для створення 2D та 3D ігор. Завдяки простоті використання, гнучкості та широкому набору функцій Godot Engine став популярним інструментом для реалізації навчальних і творчих проєктів у сфері GameDev.

Актуальність теми. Розробка 2D аркадних ігор є актуальним напрямом сучасної ігрової індустрії, оскільки подібні проєкти не потребують складного технічного забезпечення, мають зрозумілий ігровий процес та користуються популярністю серед гравців різного віку. Крім програмної реалізації, важливу роль у створенні таких ігор відіграє візуальна складова: дизайн персонажів, створення анімацій, розробка інтерфейсу та художнє оформлення сцени. Саме тому розробка власної 2D гри дає змогу поєднати навички програмування та цифрового дизайну в одному програмному продукті.

У межах даного дослідження розробляється 2D аркадна гра «Apple Madness», у якій гравець керує персонажем-фермером та повинен ловити корисні об’єкти, уникаючи небезпечних. Ігровий процес базується на механіці випадкового падіння предметів, системі підрахунку очок та життів, а також використанні анімованих 2D-об’єктів.

Предметом дослідження є методи проєктування та реалізації 2D аркадної гри, створення анімованих ігрових об’єктів і програмна реалізація основних ігрових механік засобами Godot Engine.

Об’єктом дослідження є процес розробки комп’ютерної 2D гри жанру Arcade у середовищі Godot Engine.

Метою роботи є проєктування та розробка 2D аркадної гри «Apple Madness» із використанням засобів Godot Engine, а також створення власних графічних і анімаційних матеріалів для реалізації повноцінного ігрового продукту.

Для досягнення поставленої мети було визначено такі **завдання**:

- проаналізувати сучасний стан індустрії комп’ютерних ігор та особливості жанру 2D Arcade;
- дослідити функціональні можливості Godot Engine;
- розробити концепцію гри «Apple Madness» та структуру її ігрового процесу;
- створити графічні матеріали та анімації персонажа й ігрових об’єктів;
- реалізувати систему керування персонажем та механіку випадкового падіння об’єктів;
- реалізувати систему очок, життів та взаємодії з предметами;
- розробити користувацький інтерфейс гри;
- провести тестування та оптимізацію програмного продукту.

Наукова новизна отриманих результатів полягає у поєднанні художнього оформлення та програмної реалізації 2D аркадної гри із застосуванням власноруч створених графічних і анімаційних елементів у середовищі Godot Engine.

Практичне значення дослідження полягає у створенні повноцінного ігрового застосунку, який може бути використаний як приклад реалізації 2D аркадних механік під час вивчення дисциплін, пов’язаних із розробкою комп’ютерних ігор, комп’ютерною графікою та цифровим дизайном. Отримані

результати можуть бути використані для подальшого розвитку проєкту або як основа для створення інших ігор подібного жанру.

Обсяг і структура роботи. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків. У роботі наведено ілюстрації, схеми, приклади програмного коду та графічні матеріали, що демонструють процес проєктування та розробки гри «Apple Madness».

РОЗДІЛ 1. АНАЛІЗ СФЕРИ РОЗРОБКИ 2D АРКАДНИХ ІГОР

1.1. Сучасний стан розвитку індустрії комп’ютерних ігор

Індустрія комп'ютерних ігор є однією з найбільш динамічних і прибуткових галузей сучасної цифрової економіки. За останні десятиліття відеоігри пройшли шлях від простих аркадних розваг до складних інтерактивних систем, що поєднують програмування, комп'ютерну графіку, анімацію, звуковий дизайн, елементи штучного інтелекту та мережеві технології [1]. Сьогодні ігрова індустрія не лише забезпечує розважальні потреби користувачів, а й активно впливає на розвиток суміжних сфер, зокрема освіти, медицини, реклами та професійної підготовки фахівців за допомогою технологій гейміфікації.

Стрімкий розвиток індустрії комп'ютерних ігор став можливим завдяки поширенню персональних комп'ютерів, ігрових консолей та мобільних пристроїв, а також постійному вдосконаленню апаратного забезпечення. Зростання продуктивності процесорів і графічних прискорювачів дало змогу створювати складні ігрові світи з високим рівнем деталізації та реалістичності. Водночас розвиток спеціалізованих програмних засобів значно спростив процес створення ігор і зробив його доступнішим не лише для великих студій, а й для незалежних розробників.

За даними аналітичних досліджень, світовий ринок відеоігор демонструє стабільне зростання протягом останніх років. Відповідно до звіту компанії Newzoo, кількість гравців у світі перевищує три мільярди осіб, а обсяги доходів ігрової індустрії становлять сотні мільярдів доларів щорічно [30]. Це свідчить про те, що комп'ютерні ігри стали невід'ємною складовою сучасної культури та цифрового середовища.

Сучасна GameDev-індустрія характеризується значною різноманітністю платформ і жанрів. Ігри створюються для персональних комп'ютерів, мобільних пристроїв, ігрових консолей та вебсередовища. Водночас спостерігається активний розвиток технологій доповненої та віртуальної реальності, які

відкривають нові можливості для взаємодії користувача з цифровим контентом. Незважаючи на це, традиційні двовимірні ігри продовжують користуватися значною популярністю завдяки своїй доступності, зрозумілості та порівняно невисоким вимогам до апаратних ресурсів.

Важливим явищем сучасної ігрової індустрії є розвиток незалежного сегмента розробки ігор. Незалежні розробники, або інді-студії, створюють проєкти з обмеженими фінансовими та кадровими ресурсами, проте нерідко саме такі ігри стають джерелом інноваційних рішень у сфері геймдизайну. Значною мірою цьому сприяє доступність безкоштовних або умовно безкоштовних рушіїв для розробки ігор, серед яких особливе місце займають Godot Engine, Unity та Unreal Engine.

Успішними прикладами незалежних двовимірних ігор є Hollow Knight, Celeste, Undertale та Stardew Valley. Ці проєкти демонструють, що комерційний успіх гри не завжди залежить від фотореалістичної графіки чи великих бюджетів. Значно важливішими факторами є продумана ігрова механіка, цілісний художній стиль та якісний користувацький досвід.

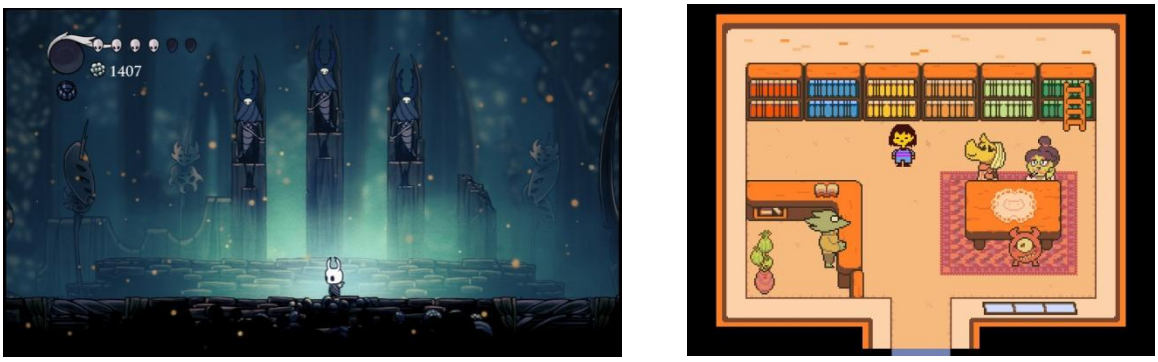

Рис. 1.1. Незалежні двовимірні проєкти

Окремої уваги заслуговує розвиток інструментів для створення ігор. Сучасні рушії пропонують інтегровані середовища розробки, що поєднують засоби програмування, редактори сцен, системи анімації, фізичні рушії та інструменти експорту проєктів на різні платформи. Це значно скорочує час розробки та дає змогу зосередитися на реалізації творчих ідей. Одним із таких інструментів є Godot Engine — відкритий рушій, який набуває все більшої

популярності серед незалежних розробників завдяки своїй гнучкості, безкоштовності та широким можливостям для створення двовимірних ігор [29].

Рис. 1 2. Середовище розробки Godot Engine

Водночас із розвитком технологій зростають і вимоги користувачів до якості ігрових продуктів. Сучасні гравці очікують від ігор не лише цікавого ігрового процесу, а й інтуїтивно зрозумілого інтерфейсу, стабільної роботи, візуальної привабливості та належного рівня оптимізації. У зв’язку з цим процес створення комп'ютерних ігор набуває міждисциплінарного характеру, поєднуючи знання з програмування, комп'ютерної графіки, психології сприйняття та проєктування користувацького досвіду.

Слід також відзначити зростання освітнього значення ігрової розробки. Вивчення технологій створення ігор дедалі частіше включається до навчальних програм закладів вищої освіти, оскільки дозволяє студентам застосовувати теоретичні знання з програмування та комп'ютерної графіки під час реалізації комплексних практичних проєктів. Розробка власної гри сприяє формуванню навичок командної роботи, аналізу вимог, проєктування програмних систем та тестування програмного забезпечення.

Таким чином, сучасна індустрія комп'ютерних ігор є потужною галуззю цифрової економіки, що постійно розвивається та інтегрує новітні інформаційні технології. Особливе місце в її структурі займає сегмент двовимірних ігор, який завдяки доступності інструментів розробки та популярності серед користувачів залишається актуальним напрямом для досліджень і практичної реалізації студентських проєктів. Саме тому проєктування та розробка 2D аркадної гри засобами сучасних рушіїв, зокрема Godot Engine, становить значний науково-практичний інтерес у межах підготовки фахівців у галузі комп'ютерних наук.

1.2. Особливості жанру 2D аркадних ігор

Аркадні ігри є одним із найстаріших і водночас найбільш упізнаваних жанрів у світовій індустрії відеоігор. Перші представники цього напряму з'явилися ще в 1970-х роках у вигляді спеціалізованих ігрових автоматів, встановлених у громадських місцях. Саме від англійського слова arcade, що означає залу з ігровими автоматами, походить назва жанру. Незважаючи на значні зміни в технологіях розробки, аркадні ігри й сьогодні залишаються популярними серед широкого кола користувачів завдяки простоті освоєння, динамічному ігровому процесу та коротким ігровим сесіям [2].

Рис. 1.3. Аркадні автомати

У сучасному розумінні аркадна гра являє собою програмний продукт, основою якого є швидка взаємодія користувача з ігровим середовищем, орієнтація на миттєву реакцію гравця та поступове ускладнення завдань. На відміну від рольових ігор або стратегій, де значна увага приділяється розвитку сюжету чи управлінню складними системами, аркадні ігри концентруються на відпрацюванні певної базової механіки, яка становить основу всього ігрового процесу.

Однією з ключових характеристик аркадних ігор є простота правил. Гравець повинен швидко зрозуміти принципи взаємодії з грою та розпочати проходження практично без додаткового навчання. Це особливо важливо в умовах високої конкуренції на сучасному ринку цифрових розваг, де користувачі віддають перевагу продуктам із низьким порогом входження.

Характерною особливістю аркадних ігор також є циклічність ігрового процесу. Основна механіка багаторазово повторюється протягом гри, однак зі зростанням складності або зміною окремих параметрів. Наприклад, у класичній грі Pac-Man гравець переміщується лабіринтом, збираючи об'єкти та уникаючи ворогів, тоді як складність поступово зростає за рахунок швидкості пересування супротивників.

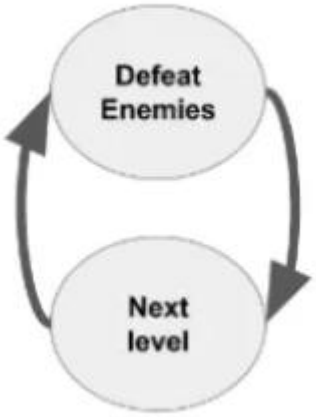

Рис. 1.4.Цикл ігор жанру аркад

Подібний принцип використовується і в сучасних аркадних проєктах. У грі «Fruit Ninja» основною механікою є розрізання фруктів, що з’являються на екрані. Попри простоту задуму, гра утримує увагу користувача завдяки зміні темпу, збільшенню кількості об’єктів та впровадженню додаткових режимів.

Ще однією важливою рисою аркадних ігор є орієнтація на досягнення високого результату. Більшість представників жанру використовують систему підрахунку очок, рекордів або рейтингів, що стимулює гравців до багаторазового повторення ігрових сесій з метою покращення власних показників. Такий підхід позитивно впливає на повторну залученість користувачів та збільшує тривалість взаємодії з ігровим продуктом.

З погляду візуальної організації значна частина аркадних ігор реалізується саме у двовимірному форматі. Це пояснюється низкою переваг 2D-графіки, серед яких можна виділити відносну простоту створення графічних ресурсів, менші вимоги до технічних характеристик пристроїв та можливість

формування виразного художнього стилю. Двовимірне представлення також сприяє кращому сприйняттю ігрового простору, оскільки всі важливі елементи залишаються в межах поля зору користувача.

Важливим компонентом аркадних ігор є поступове збільшення складності. Такий підхід забезпечує баланс між доступністю та викликом для гравця. На початкових етапах користувач знайомиться з основними механіками, тоді як у подальшому гра вимагає більшої концентрації, швидкості реакції та точності дій. Саме підтримання оптимального рівня складності вважається одним із ключових завдань геймдизайну.

У дослідженнях, присвячених теорії ігрового дизайну, часто використовується поняття стану потоку, запропоноване американським психологом Мігаєм Чіксентмігаї. Відповідно до цієї концепції, максимальне залучення користувача виникає тоді, коли складність завдання відповідає рівню його навичок. Надто проста гра швидко викликає нудьгу, тоді як надмірно складна — призводить до розчарування та втрати інтересу [27].

Протягом розвитку індустрії комп'ютерних ігор сформувалося кілька різновидів аркадних ігор. До найпоширеніших належать платформери, нескінченні ранери, ігри з механікою ухилення від перешкод, аркадні головоломки, космічні шутери та ігри, засновані на ловлі або збиранні об'єктів. Незважаючи на різноманітність форм реалізації, усі вони ґрунтуються на спільних принципах: швидкому темпі взаємодії, зрозумілих правилах та безпосередньому впливі дій користувача на результат ігрової сесії.

Особливий інтерес у межах даного дослідження становлять аркадні ігри, побудовані на механіці ловлі об'єктів. Суть таких ігор полягає в необхідності своєчасно реагувати на появу предметів, що рухаються у визначеному напрямку, та взаємодіяти з ними відповідно до встановлених правил. Як правило, користувач повинен збирати корисні об'єкти, отримуючи за це бали, і водночас уникати небажаних предметів, зіткнення з якими призводить до штрафів або завершення гри.

Популярність подібних механік пояснюється їх універсальністю та доступністю для широкої аудиторії. Для початку гри користувачеві достатньо освоїти мінімальний набір дій, наприклад переміщення персонажа по горизонталі. У подальшому складність може збільшуватися шляхом підвищення швидкості руху об'єктів, зміни частоти їх появи або введення нових типів предметів із різними функціональними властивостями.

З позиції проєктування програмного забезпечення аркадні ігри є зручним об'єктом для практичної реалізації студентських проєктів. Вони дозволяють поєднати основні етапи розробки комп'ютерних ігор, зокрема створення графічних ресурсів, проєктування інтерфейсу користувача, програмування ігрової логіки, організацію взаємодії між об'єктами та тестування готового продукту. Водночас відносна простота жанру дає змогу зосередитися на якості реалізації окремих механік і загальній цілісності ігрового досвіду.

Рис. 1.5. Основні етапи розробки комп'ютерних ігор

Важливу роль в аркадних іграх відіграє користувацький інтерфейс. Оскільки ігровий процес характеризується високою динамікою, інформація про кількість набраних очок, залишок життів, рівень складності чи поточний стан гри повинна бути зрозумілою та легко зчитуватися. Надмірне перевантаження інтерфейсу може негативно вплинути на концентрацію користувача та знизити комфорт взаємодії з грою.

Не менш важливим є використання системи візуального та звукового зворотного зв’язку. Кожна дія гравця повинна супроводжуватися відповідною реакцією з боку ігрового середовища. Отримання очок, втрата життя, успішне виконання завдання або перехід до нового етапу мають підкріплюватися

візуальними ефектами, анімаціями чи звуковими сигналами. Такий підхід підвищує рівень залучення користувача та сприяє формуванню позитивного ігрового досвіду [3].

Слід зазначити, що попри стрімкий розвиток тривимірних технологій, двовимірні аркадні ігри продовжують займати важливе місце на ринку цифрових розваг. Це підтверджується успіхом численних незалежних проєктів, орієнтованих на використання стилізованої 2D-графіки та класичних принципів побудови ігрового процесу. Значною мірою цьому сприяє поширення мобільних платформ, для яких короткі ігрові сесії та інтуїтивно зрозуміле керування є особливо актуальними.

Для реалізації аркадних ігор сучасні розробники активно використовують спеціалізовані рушії, що надають готові інструменти для створення двовимірних сцен, анімації, фізичної взаємодії та програмування поведінки об'єктів. Одним із таких засобів є Godot Engine, який завдяки відкритому вихідному коду, відсутності ліцензійних відрахувань та широким можливостям підтримки 2D-графіки набуває все більшої популярності серед незалежних розробників.

Таким чином, жанр 2D аркадних ігор характеризується простотою базових правил, динамічністю ігрового процесу, орієнтацією на швидку реакцію користувача та високою повторюваністю ігрових сесій. Завдяки доступності для широкого кола гравців і відносній простоті реалізації аркадні ігри залишаються актуальним напрямом сучасної індустрії комп'ютерних ігор. Їхні особливості створюють сприятливі умови для проєктування та розробки власних програмних продуктів навчального призначення, зокрема 2D аркадної гри «Apple Madness», що базується на механіці ловлі об’єктів і передбачає поступове ускладнення ігрового процесу.

1.3. Аналіз програмного забезпечення для розробки ігор

Розробка сучасних відеоігор є складним багаторівневим процесом, який передбачає використання спеціалізованого програмного забезпечення для реалізації графіки, логіки, фізики, анімації та користувацького інтерфейсу. Вибір інструментів безпосередньо впливає на якість фінального продукту, швидкість розробки та можливості масштабування проєкту. Саме тому аналіз середовища розробки є важливим етапом перед початком створення будь-якої гри.

Сучасні ігрові рушії умовно поділяються на універсальні та спеціалізовані. Універсальні рушії, такі як Unity та Unreal Engine, використовуються для створення ігор різних жанрів і платформ. Вони підтримують як 2D, так і 3D розробку, мають розвинуті системи фізики, освітлення, анімації та мережевої взаємодії. Завдяки цьому вони широко застосовуються як у великих комерційних проєктах, так і в інді-розробці.

Окремо варто виділити спеціалізовані рушії, орієнтовані переважно на 2D-розробку. До таких належить Godot Engine, який є відкритим програмним забезпеченням і активно використовується для створення аркадних, платформерних та мобільних ігор. Його ключовою перевагою є легкість, гнучкість та низькі системні вимоги, що робить його особливо привабливим для навчальних і інді-проєктів [4].

Важливо також розглядати додаткові інструменти, які використовуються в процесі розробки ігор. До них належать графічні редактори для створення 2D-арту, такі як Adobe Photoshop та Aseprite, які дозволяють розробляти спрайти, текстури та елементи інтерфейсу. Для роботи з анімацією часто застосовуються як вбудовані інструменти рушіїв, так і зовнішні програми, що забезпечують покадрову або процедурну анімацію.

Окрему категорію становлять середовища для програмування. У більшості сучасних рушіїв використовується вбудована підтримка мов програмування, наприклад C# у Unity або GDScript у Godot Engine. Це дозволяє реалізовувати ігрову логіку, керування об’єктами, взаємодію користувача з

грою та системи штучного інтелекту. Вибір мови програмування часто визначається архітектурою рушія та вимогами проєкту.

Порівнюючи сучасні ігрові рушії, важливо враховувати не лише їхні технічні можливості, але й зручність використання, швидкість розробки та доступність для навчальних і інді-проєктів. Наприклад, Unity має потужну екосистему, великий набір готових рішень та активну спільноту розробників. Це робить його універсальним інструментом для створення ігор будь-якого масштабу. Водночас Unity потребує значних ресурсів і має складнішу архітектуру, що може ускладнювати навчальний процес для початківців.

Таблиця 1.1

Порівняльна характеристика ігрових рушіїв

Критерій порівняння	**Godot Engine**	**Unity**	**Unreal Engine**
Ліцензія	Безкоштовна, відкритий вихідний код (MIT)	Умовно безкоштовна, комерційна ліцензія	Безкоштовний до певного рівня прибутку, роялті
Орієнтація	2D та 3D	2D та 3D	Переважно 3D
Підтримка 2D-ігор	Висока, окремий 2D-рушій	Висока	Обмежена
Складність освоєння	Низька	Середня	Висока
Мова програмування	GDScript, C#, C++	C#, Visual Scripting	C++, Blueprint
Вимоги до апаратного забезпечення	Низькі	Середні	Високі
Наявність відкритого коду	Так	Ні	Частково
Швидкість створення невеликих проєктів	Висока	Висока	Середня
Інструменти для роботи зі спрайтами	Вбудовані та зручні	Добре реалізовані	Менш орієнтовані на 2D
Підтримка фізики для 2D	Вбудована	Вбудована	Вбудована
Розмір спільноти	Середній, активно зростає	Дуже великий	Великий
Кількість готових ресурсів та плагінів	Середня	Дуже велика	Велика
Доцільність використання в навчальних проєктах	Висока	Висока	Середня
Доцільність для інді-розробки	Висока	Висока	Середня
Доцільність для проєкту «Apple Madness»	Оптимальна	Можлива	Недоцільна

Unreal Engine, у свою чергу, орієнтований на високоякісну графіку та складні 3D-проєкти. Його перевагою є візуальна система програмування та

високий рівень графічного рендерингу. Однак для 2D аркадних ігор його можливості часто є надмірними, що призводить до перевантаження проєкту зайвими функціями.

На фоні цих рушіїв Godot Engine виглядає більш збалансованим рішенням для розробки 2D-ігор. Його головною перевагою є простота структури та орієнтація саме на 2D-розробку, що дозволяє швидко створювати ігрові сцени, працювати зі спрайтами та реалізовувати базову фізику без зайвих складнощів. Крім того, Godot має відкритий вихідний код, що робить його доступним для безкоштовного використання та модифікації.

Ще однією важливою перевагою є система сцен і вузлів, через яку будується ігрова логіка як ієрархічна структура. Це значно спрощує організацію проєкту, особливо у 2D-іграх, де велика кількість об’єктів взаємодіє між собою. Такий підхід є більш інтуїтивним відносно компонент інших систем рушіїв.

Також варто враховувати інструменти для роботи з графікою та анімацією. У контексті 2D-розробки часто використовуються легкі редактори спрайтів, наприклад Aseprite, якими створюють покадрову анімацію та працюють з піксель-арт стилем [5]. У поєднанні з Godot Engine це забезпечує ефективний робочий процес від створення графіки до її інтеграції в гру.

Окремо слід зазначити, що вибір програмного забезпечення залежить також від цілей проєкту. Для великих комерційних ігор доцільно використовувати більш складні рушії з розширеними можливостями, тоді як для навчальних, інді та аркадних проєктів оптимальним є легке та гнучке середовище. У випадку гри «Apple Madness» саме Godot Engine забезпечує оптимальний баланс між функціональністю та простотою реалізації.

Таким чином, аналіз програмного забезпечення показує, що вибір інструментів розробки повинен базуватися на типі гри, її складності та вимогах до продуктивності. Для 2D аркадних проєктів найбільш раціональним рішенням є використання спеціалізованих або напівуніверсальних рушіїв, які забезпечують швидку розробку та зручну інтеграцію всіх елементів гри.

1.4. Огляд Godot Engine як інструменту розробки

Godot Engine є сучасним відкритим ігровим рушієм, який активно використовується для створення 2D та 3D відеоігор різного рівня складності. Його головною особливістю є орієнтація на гнучкість, простоту використання та доступність, що робить його особливо популярним серед інді-розробників, студентів та невеликих студій. Завдяки відкритому вихідному коду рушій постійно розвивається спільнотою, що забезпечує його актуальність та швидке впровадження нових технологій.

Однією з ключових переваг Godot Engine є його унікальна архітектура, яка базується на системі сцен і вузлів. Кожен ігровий об’єкт у рушії представлений як вузол, а сцена є ієрархічною структурою таких вузлів. Це дозволяє будувати логіку гри у вигляді модульної системи, де кожен елемент можна легко редагувати, повторно використовувати та комбінувати з іншими. Такий підхід значно спрощує розробку 2D-ігор, оскільки забезпечує високу організованість проєкту.

Особливу увагу варто звернути на те, що Godot має вбудовану підтримку 2D-графіки, яка не є просто надбудовою над 3D-системою, а реалізована як окрема оптимізована підсистема. Це дозволяє досягати високої продуктивності навіть на слабших пристроях. Для 2D аркадних ігор, таких як «Apple Madness», це є критично важливою перевагою, оскільки жанр передбачає велику кількість одночасних об’єктів та швидку зміну сцен.

Також рушій підтримує декілька мов програмування, серед яких найбільш поширеною є GDScript — спеціально розроблена мова, синтаксис якої подібний до Python. Вона є простою для освоєння та дозволяє швидко реалізовувати ігрову логіку, взаємодію об’єктів, систему подій та інші механіки [6]. Крім того, Godot підтримує C# та C++, що розширює можливості для більш складних проєктів.

Ще однією важливою особливістю є інтегрований набір інструментів для роботи з анімацією, фізикою та користувацьким інтерфейсом. Рушій має власну систему анімації, яка дозволяє створювати як прості покадрові анімації, так і

складні процедурні переходи між станами об’єктів. Це робить його зручним для створення динамічних аркадних ігор, де важлива плавність ігрового процесу.

Важливою перевагою Godot Engine є його висока продуктивність при роботі з 2D-об’єктами. На відміну від універсальних рушіїв, які часто орієнтовані на 3D-графіку і лише адаптовані під 2D, Godot має нативну 2D-систему рендерингу. Це дозволяє ефективно обробляти велику кількість спрайтів, анімацій та фізичних взаємодій без значного навантаження на систему. Для аркадної гри «Apple Madness», де передбачається постійна поява та рух об’єктів, це є критично важливим фактором.

Ще одним значним аспектом є система сигналів, яка використовується для організації взаємодії між об’єктами. Замість жорсткого зв’язування елементів між собою, Godot дозволяє використовувати подієву модель, де один об’єкт може «надсилати сигнал», а інший — його обробляти. Це значно спрощує архітектуру гри та робить код більш гнучким і зрозумілим. Такий підхід особливо корисний у 2D аркадах, де велика кількість ігрових елементів постійно взаємодіє між собою.

У порівнянні з іншими рушіями, такими як Unity або Unreal Engine, Godot виграє у швидкості розробки простих та середніх за складністю проєктів. Unity пропонує значно ширший функціонал, але потребує більше часу на налаштування проєкту та має складнішу структуру компонентів. Unreal Engine, у свою чергу, є надто ресурсомістким для 2D-аркад і орієнтований переважно на високобюджетні 3D-проєкти.

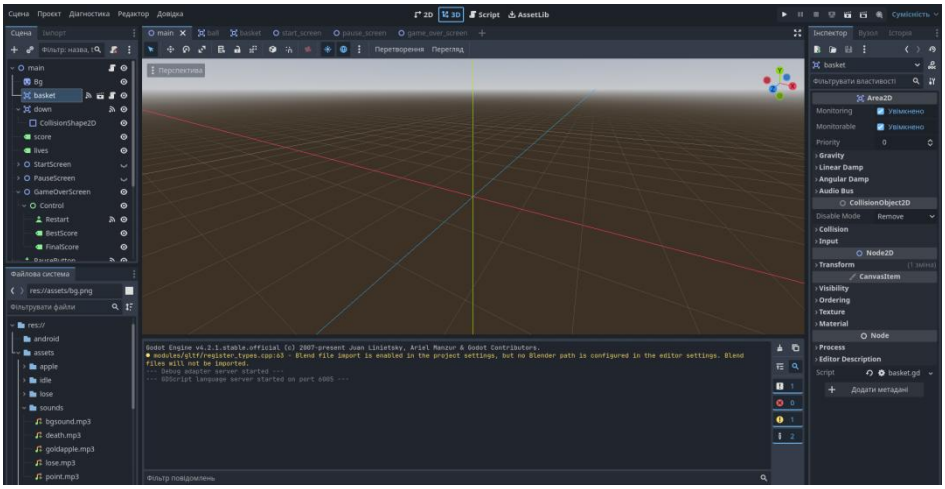

Рис. 1.6. Інтерфейс середовища

Окремо варто зазначити зручність роботи з користувацьким інтерфейсом у Godot. Рушій має вбудовану систему UI-вузлів, яка дозволяє швидко створювати меню, HUD-елементи та інтерактивні інтерфейси без необхідності використання зовнішніх бібліотек. Це дозволяє легко реалізувати такі елементи гри, як лічильник очок, життя гравця або екрани завершення рівня.

Для проєкту «Apple Madness» вибір Godot Engine є обґрунтованим з точки зору балансу між простотою, продуктивністю та функціональністю. Рушій дозволяє швидко реалізувати базові ігрові механіки, інтегрувати графічні та анімаційні елементи, а також забезпечує стабільну роботу гри на різних пристроях. Це особливо важливо для аркадного жанру, де ключову роль відіграє швидкість реакції та плавність ігрового процесу.

Таким чином, Godot Engine є оптимальним вибором для розробки 2D аркадної гри, оскільки поєднує в собі легкість освоєння, потужні інструменти для 2D-розробки та достатню гнучкість для реалізації складніших механік у майбутньому.

1.5. Аналіз графічних та анімаційних інструментів

Сучасна розробка відеоігор неможлива без використання спеціалізованих графічних та анімаційних інструментів, які забезпечують створення візуального контенту, його оптимізацію та подальшу інтеграцію в ігровий рушій. У процесі виробництва 2D-ігор використовуються як універсальні графічні редактори для створення ілюстрацій і спрайтів, так і вузькоспеціалізовані програми для анімації, які дозволяють формувати рух персонажів, об’єктів та елементів інтерфейсу.

До найбільш поширених інструментів для створення 2D-графіки належать такі програми, як Adobe Photoshop, Krita та Aseprite. Adobe Photoshop широко використовується в індустрії завдяки гнучкості шарової структури, підтримці різних форматів та можливості детальної роботи з текстурами. Krita, у свою чергу, є безкоштовним рішенням з відкритим кодом, яке орієнтоване на цифровий живопис і концепт-арт. Aseprite спеціалізується саме на піксельній графіці, що робить його особливо популярним у розробці інді-ігор та ретро-стилізованих проєктів.

Окрему категорію становлять інструменти для анімації, які дозволяють оживляти створені графічні елементи. У професійній розробці 2D-ігор активно використовуються Spine, DragonBones та Adobe Animate. Spine орієнтований на скелетну анімацію, що дозволяє створювати плавні рухи персонажів із мінімальною кількістю кадрів. DragonBones є безкоштовною альтернативою з подібним функціоналом і підтримкою інтеграції в різні ігрові рушії.

Особливе місце у даній роботі займає саме Adobe Animate, оскільки він був використаний як основний інструмент для створення анімаційних елементів гри «Apple Madness». Дана програма поєднує можливості класичної покадрової анімації та автоматизованої інтерполяції рухів, що дозволяє створювати як прості циклічні анімації (наприклад, рух персонажа або обертання об’єктів), так і більш складні інтерактивні сцени.

Важливою перевагою Adobe Animate є підтримка експорту в різні формати, зокрема HTML5 Canvas, що робить його зручним для інтеграції в

сучасні ігрові рушії, включаючи Godot Engine [8]. Це дозволяє використовувати створені анімації без суттєвої втрати якості та з мінімальними витратами на додаткову адаптацію.

У порівнянні з іншими інструментами, такими як Spine або DragonBones, Adobe Animate має більш універсальний характер. Якщо Spine спеціалізується переважно на кістковій анімації персонажів, то Animate дозволяє працювати як з персонажами, так і з інтерфейсом, графічними ефектами та елементами середовища. Це робить його особливо зручним для невеликих і середніх проєктів, де важлива швидкість розробки та гнучкість.

У практичній частині проєкту «Apple Madness» Adobe Animate використовувався для створення ключових візуальних елементів: анімації головного персонажа, руху об’єктів, ефектів взаємодії та частково елементів інтерфейсу. Такий підхід дозволив забезпечити єдиний візуальний стиль гри та спростити процес інтеграції графіки в рушій Godot.

Таким чином, аналіз графічних та анімаційних інструментів показує, що вибір програмного забезпечення безпосередньо впливає на якість, швидкість та ефективність розробки ігрового продукту. Використання Adobe Animate у межах даної роботи є обґрунтованим рішенням, яке поєднує функціональність, зручність та достатній рівень професійних можливостей для реалізації 2D аркадної гри.

РОЗДІЛ 2. ПРОЄКТУВАННЯ КОМП'ЮТЕРНОЇ ГРИ «APPLE MADNESS»

2.1. Аналіз аркадних ігор із механікою ловлі об’єктів

Аркадні ігри є одним із найстаріших та водночас найстабільніше популярних жанрів у відеоігровій індустрії. Їхня ключова особливість полягає у простих правилах, високій динаміці та орієнтації на швидкість реакції гравця. Такі ігри зазвичай не потребують тривалого навчання, але забезпечують поступове зростання складності, що формує «циклічний» ігровий процес і мотивує до повторних спроб проходження [10].

Однією з найбільш поширених підмеханік у жанрі аркад є механіка ловлі або збору об’єктів. Її суть полягає в тому, що гравець керує персонажем або платформою, завданням якої є перехоплення певних об’єктів, що падають, рухаються або з’являються в ігровому просторі. У разі успішного збору гравець отримує бали, ресурси або інші ігрові переваги, тоді як пропуск об’єкта може призводити до втрати очок або програшу.

Класичним прикладом реалізації подібної механіки є гра «Fruit Ninja», де гравець повинен швидко реагувати на появу об’єктів і взаємодіяти з ними через жести. Хоча механіка тут базується на розрізанні, а не ловлі, принцип швидкої реакції та обробки великої кількості об’єктів залишається аналогічним.

Рис 2.1. Гра «Fruit Ninja»

Більш прямим прикладом ігор із механікою збору об’єктів є «Super Mario Bros.» та інші платформери серії. У цих іграх гравець збирає монети, бонуси та предмети, які розміщені на рівнях або з’являються у процесі проходження. Ця механіка формує додаткову мотивацію дослідження рівня та підсилює ігрову винагороду.

Рис. 2.2. Гра «Super Mario Bros.»

Схожий підхід використовується і в мобільних аркадах, таких як «Doodle Jump», де гравець повинен не лише уникати перешкод, але й збирати бонусні об’єкти для отримання переваг. У таких іграх механіка збору часто поєднується з елементами виживання та поступового ускладнення геймплею.

Рис. 2.3. Гра «Doodle Jump»

З точки зору геймдизайну, механіка ловлі об’єктів базується на трьох ключових компонентах: генерації об’єктів, системі взаємодії та системі винагороди [11]. Генерація об’єктів визначає частоту та випадковість появи

ігрових елементів, що впливає на складність гри. Система взаємодії описує спосіб, яким гравець може ці об’єкти «зловити» — через рух, натискання або фізичний контакт. Система винагороди визначає наслідки успішної взаємодії, формуючи ігрову мотивацію.

Окремо слід зазначити, що дана механіка широко використовується в мобільних іграх через свою простоту та інтуїтивність. Вона не потребує складного управління, що робить її доступною для широкої аудиторії. Крім того, такі ігри легко масштабуються шляхом збільшення швидкості падіння об’єктів, додавання нових типів предметів або введення негативних елементів, які необхідно уникати.

У контексті даної дипломної роботи механіка ловлі об’єктів є основою ігрового процесу гри «Apple Madness», де гравець взаємодіє з об’єктами, що з’являються у межах ігрового простору, намагаючись набрати максимальну кількість очок та уникнути негативних сценаріїв. Такий підхід дозволяє поєднати класичні принципи аркадного жанру з сучасними вимогами до візуальної подачі та інтерактивності.

2.2. Формування концепції гри «Apple Madness»

Формування концепції комп'ютерної гри є одним із найважливіших етапів перед безпосередньою розробкою, оскільки саме на цьому рівні визначається загальна логіка майбутнього продукту, його структура, художній напрям та принципи взаємодії з користувачем. Концепція гри виступає базовою моделлю, яка об'єднує всі наступні етапи розробки — від проєктування механік до створення графічного та програмного контенту.

Гра «Apple Madness» позиціонується як 2D аркадна гра з елементами реакційного геймплею, де основна увага приділяється швидкості прийняття рішень та координації дій гравця. Основна ігрова ідея полягає у взаємодії з об'єктами, що з'являються у верхній частині екрану та рухаються донизу, що створює умови постійної динаміки та необхідності швидкої реакції. Такий підхід є типовим для аркадного жанру, який орієнтований на прості правила, але складне освоєння майстерності.

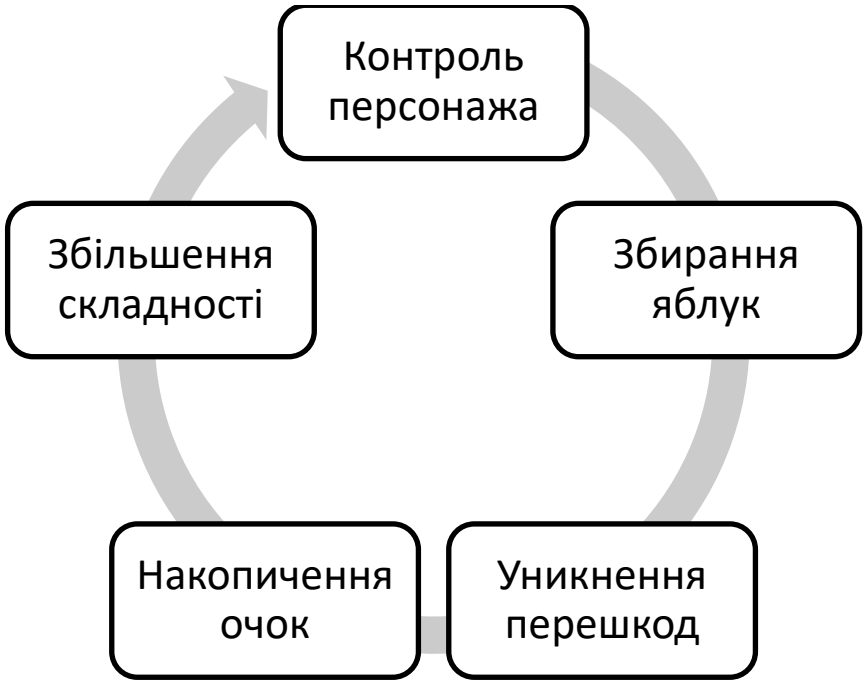

Рис. 2.4. Ігровий цикл «Apple Madness»

У структурі концепції ключову роль відіграє визначення базового ігрового циклу. Ігровий цикл у «Apple Madness» складається з повторюваних дій: поява об'єктів → їх рух у межах ігрового поля → взаємодія гравця з

об’єктами → нарахування або зняття очок → збільшення складності. Подібні цикли характерні для багатьох успішних аркадних проєктів, оскільки вони забезпечують стабільну залученість гравця та формують відчуття постійного прогресу навіть у коротких сесіях [12].

Важливою складовою концепції є поділ ігрових об’єктів на позитивні та негативні. Позитивні об’єкти (яблука) виконують функцію основного джерела очок і мотивації, тоді як негативні об’єкти створюють ризик втрати результату або завершення гри. Такий підхід дозволяє сформувати систему прийняття рішень у реальному часі, де гравець постійно оцінює ризик і потенційну винагороду.

Як приклади подібних механік можна навести такі ігри як «Fruit Ninja», де гравець реагує на появу об’єктів у реальному часі, та «Doodle Jump», де важливим є постійний рух і уникнення небезпек. Хоча ці ігри відрізняються за способом взаємодії, їх об’єднує спільний принцип — швидка реакція на змінне ігрове середовище.

Додатково концепція гри передбачає поступове ускладнення, яке реалізується через збільшення швидкості падіння об’єктів, зменшення інтервалів їх появи та введення нових типів перешкод. Такий механізм дозволяє підтримувати баланс між доступністю гри для новачків і викликом для більш досвідчених гравців.

Окрему увагу в концепції приділено візуальній простоті та читабельності. Оскільки гра передбачає високу швидкість подій на екрані, важливо, щоб кожен об’єкт був легко впізнаваним і не перевантажував сприйняття користувача. Це особливо важливо для аркадних ігор, де візуальна ясність напряму впливає на ігрову ефективність.

Важливим аспектом концепції є також визначення загального художнього стилю гри. «Apple Madness» орієнтується на стилізовану 2D графіку з яскравими кольорами та чіткими формами, що забезпечує швидке зчитування ігрових об’єктів навіть у динамічних умовах. Центральним візуальним символом виступає яблуко, яке виконує не лише ігрову функцію, але й формує

впізнавану ідентичність проєкту. Подібний підхід широко використовується в казуальних іграх, де один домінантний образ формує всю візуальну систему, наприклад у «Cut the Rope».

З технічного боку концепція гри передбачає реалізацію на базі ігрового рушія Godot Engine, який підтримує розробку 2D-проєктів, роботу зі сценами та гнучку систему скриптів. Вибір цього рушія обумовлений його легкістю, відкритим кодом та зручністю інтеграції з 2D-графікою, що є критично важливим для аркадних ігор із великою кількістю об’єктів на екрані.

Окремим елементом концепції є визначення цільової аудиторії. Гра орієнтована на широку групу користувачів, включаючи як новачків у відеоіграх, так і більш досвідчених гравців, які шукають прості, але динамічні ігрові сесії. Завдяки мінімальному порогу входження «Apple Madness» може використовуватися як гра для короткочасного дозвілля, що відповідає сучасним тенденціям мобільного та казуального геймінгу.

Крім того, концепція передбачає наявність системи мотивації гравця через накопичення очок та можливість покращення власного результату. Це формує так званий «повторюваний ігровий цикл», коли користувач повертається до гри з метою перевершити попередній результат. Подібна модель є однією з основних у сучасних аркадних і мобільних іграх, оскільки вона забезпечує високу реіграбельність без складних сюжетних структур.

Таким чином, сформована концепція гри «Apple Madness» поєднує простоту аркадного жанру, динамічний геймплей, чітку візуальну структуру та технічну реалізацію на базі сучасного ігрового рушія. Це створює цілісний фундамент для подальших етапів проєктування та розробки програмного продукту.

2.3. Проєктування структури комп’ютерної гри

Проєктування структури комп’ютерної гри є одним із ключових етапів розробки ігрового продукту, оскільки саме на цьому рівні формується логіка взаємодії між усіма компонентами системи. Від того, наскільки грамотно буде спроєктована структура гри, залежить зручність користувацького досвіду, стабільність геймплейних механік та загальна цілісність проєкту.

У межах гри «Apple Madness» структурне проєктування охоплює основні складові інтерфейсу та ігрового процесу, включаючи головне меню, користувацький інтерфейс під час гри, ігрову сцену та систему ігрових об’єктів. Кожен із цих елементів виконує окрему функцію, але водночас є частиною єдиної взаємопов’язаної системи.

Проєктування головного меню

Головне меню є одним із ключових елементів структури комп’ютерної гри, оскільки саме воно формує перше враження користувача та забезпечує базову навігацію між основними функціями продукту. У контексті розробки 2D аркадної гри «Apple Madness» головне меню розглядається як інтерактивний візуальний компонент, який поєднує естетичну привабливість, інтуїтивну зрозумілість та функціональну мінімалістичність.

Основною метою проєктування головного меню є створення простої та логічної структури взаємодії, яка дозволяє гравцеві швидко розпочати гру, ознайомитися з налаштуваннями або вийти з програми. В аркадних іграх, таких як *Cuphead* або *Super Meat Boy*, головне меню зазвичай не перевантажене елементами та виконує лише базові функції, що дозволяє зосередити увагу на ігровому процесі.

Структурно головне меню гри «Apple Madness» передбачає наявність таких основних елементів:

- кнопка «Почати гру»;
- кнопка «Налаштування»;
- кнопка «Про гру»;
- кнопка «Вихід».

Кожен із цих елементів виконує чітко визначену функцію та реалізується у вигляді інтерактивного UI-компонента. При цьому особлива увага приділяється візуальній ієрархії: найбільш важлива дія («Почати гру») має бути візуально домінантною, наприклад за рахунок розміру, контрасту або анімаційного підсвічування.

Окремим аспектом є стилістичне узгодження меню з загальною художньою концепцією гри. Оскільки «Apple Madness» належить до аркадного жанру, доцільним є використання яскравої кольорової палітри, спрощених форм та легких анімаційних ефектів. Подібний підхід використовується в іграх Brawl Stars та Angry Birds, де інтерфейс є частиною загального візуального стилю, а не відокремленим елементом.

Важливим принципом проєктування є забезпечення зворотного візуального зв’язку. При наведенні курсора або виборі кнопки користувач повинен отримувати миттєву реакцію системи у вигляді зміни кольору, масштабу або анімації. Це підвищує інтуїтивність взаємодії та покращує загальний користувацький досвід.

Також слід враховувати принцип доступності інтерфейсу. Елементи головного меню повинні бути достатньо великими для легкого сприйняття, мати чіткий контраст із фоном та бути розташованими відповідно до логіки читання (зверху вниз або по центру екрана). Це особливо важливо для аркадних ігор, орієнтованих на широку аудиторію.

Проєктування користувацького інтерфейсу

Користувацький інтерфейс є центральним елементом взаємодії між гравцем і ігровою системою, оскільки саме через нього передається вся ключова інформація про стан гри, прогрес та доступні дії. У процесі розробки 2D аркадної гри «Apple Madness» користувацький інтерфейс проєктується як мінімалістична, але функціонально насичена система візуальної комунікації, що не перевантажує гравця, але забезпечує повний контроль над ігровою ситуацією.

Основним принципом проєктування інтерфейсу є принцип інформаційної достатності, згідно з яким користувач повинен отримувати лише ту інформацію, яка є критично важливою для прийняття ігрових рішень. В аркадних іграх, таких як Geometry Dash або Fruit Ninja, інтерфейс часто зведений до мінімуму, що дозволяє зосередити увагу на геймплеї, а не на елементах керування [14].

У межах гри «Apple Madness» користувацький інтерфейс включає такі основні компоненти:

- лічильник очок;
- індикатор життів або помилок;
- кнопки паузи та виходу до меню;
- візуальні ефекти зворотного зв’язку.

Кожен із цих елементів виконує окрему функцію в системі ігрової комунікації. Наприклад, лічильник очок є основним індикатором успішності гравця, тоді як індикатор життів відображає рівень ризику та наближення завершення гри. Подібні системи широко використовуються в аркадних іграх, таких як Pac-Man або Space Invaders, де прості числові або графічні індикатори формують основу ігрового досвіду.

Важливим аспектом є візуальна інтеграція інтерфейсу в загальний художній стиль гри. Елементи UI повинні не виглядати відокремлено, а органічно вписуватися у візуальну систему «Apple Madness». Це досягається через використання єдиної кольорової палітри, стилізованих іконок та узгоджених типографічних рішень. Наприклад, якщо гра використовує мультяшну стилістику, то й елементи інтерфейсу повинні мати округлі форми та м’які кольорові переходи.

Окрему роль відіграє принцип візуальної ієрархії. Найважливіші елементи (наприклад, кількість очок або життів) повинні бути більш помітними за рахунок розміру, контрасту або розташування у верхній частині екрана. Менш важлива інформація, наприклад кнопка паузи, може бути візуально менш домінантною, але легко доступною.

Також у проєктуванні інтерфейсу важливим є забезпечення миттєвого зворотного зв’язку. Будь-яка дія користувача повинна супроводжуватися візуальною або звуковою реакцією. Наприклад, при отриманні очок може з’являтися коротка анімація або ефект підсвічування, що підсилює відчуття взаємодії з грою. Подібні механіки активно використовуються в іграх *Plants vs.* Zombies та Candy Crush Saga, де кожна дія супроводжується візуальним підтвердженням.

Проєктування ігрової сцени

Ігрова сцена є основним середовищем, у якому відбувається взаємодія всіх ігрових механік, персонажів та об’єктів. У контексті розробки 2D аркадної гри «Apple Madness» вона виконує роль центрального простору, що визначає ритм геймплею, складність взаємодій та загальну візуальну динаміку гри. Саме тому проєктування ігрової сцени є одним із ключових етапів створення ігрового продукту.

Основною метою проєктування ігрової сцени є створення зрозумілого, функціонального та візуально цілісного простору, який підтримує основну механіку гри — взаємодію з об’єктами, що з’являються в межах екрану. У подібних аркадних проєктах, таких як *Doodle Jump* або *Fruit Ninja*, ігрова сцена зазвичай має спрощену структуру, де головна увага приділяється не складному оточенню, а швидкості реакції та точності дій гравця.

У грі «Apple Madness» ігрова сцена проєктується як двовимірний простір з фіксованою або обмеженою камерою, в межах якого відбувається поява та рух ігрових об’єктів. Основними елементами сцени є:

- фон (background);
- ігрова зона взаємодії;
- межі екрану або умовні зони появи об’єктів;
- персонаж гравця;
- динамічні об’єкти (наприклад, яблука або перешкоди).

Фон ігрової сцени виконує не лише декоративну, але й композиційну функцію. Він задає загальний настрій гри та формує візуальну глибину. У

«Apple Madness» доцільним є використання простого, стилізованого фону, який не перевантажує сприйняття та дозволяє гравцеві концентруватися на основних об’єктах. Подібний підхід використовується в іграх Angry Birds та Jetpack Joyride, де фон є статичним або слабо анімованим.

Важливим аспектом є композиційне розташування елементів сцени. Ігрова зона повинна бути організована таким чином, щоб забезпечувати рівномірний розподіл об’єктів і уникати візуального перевантаження окремих ділянок екрану. Це дозволяє підтримувати баланс складності та забезпечує передбачуваність ігрового процесу.

Окрему роль відіграє система появи та руху об’єктів у межах сцени. У аркадних іграх часто використовується принцип випадкової генерації об’єктів у визначених межах. Це створює ефект непередбачуваності та підтримує інтерес гравця. Наприклад, у грі Fruit Ninja об’єкти з’являються знизу екрана та рухаються по різних траєкторіях, що змушує гравця швидко реагувати.

Також важливим елементом є візуальна читабельність сцени. Усі інтерактивні об’єкти повинні чітко відрізнятися від фону за кольором, формою та контрастом. Це дозволяє уникнути помилок сприйняття та підвищує точність взаємодії. У сучасних 2D іграх часто використовується принцип контрастного виділення об’єктів, що взаємодіють із гравцем.

Крім того, ігрова сцена повинна підтримувати логіку прогресії складності. У процесі гри можуть змінюватися швидкість появи об’єктів, їх кількість або траєкторії руху. Така адаптивність дозволяє підтримувати баланс між доступністю та викликом для гравця.

Проєктування системи ігрових об’єктів

Система ігрових об’єктів є фундаментальною складовою будь-якої комп’ютерної гри, оскільки саме вона визначає набір взаємодій, логіку геймплею та загальну динаміку ігрового процесу. У межах 2D аркадної гри «Apple Madness» ця система побудована на принципах простоти, читабельності та функціональної однозначності, що є характерним для аркадного жанру.

Основною метою проєктування системи ігрових об’єктів є створення набору елементів, які безпосередньо беруть участь у геймплейній взаємодії та забезпечують реалізацію основної механіки гри — реагування гравця на появу об’єктів у межах ігрової сцени. Подібний підхід широко використовується в класичних аркадних іграх, таких як Tetris або Fruit Ninja, де кожен об’єкт має чітко визначену функцію та наслідки взаємодії.

У грі «Apple Madness» ігрові об’єкти поділяються на кілька основних категорій:

- позитивні об’єкти (наприклад, яблука, які потрібно збирати);
- негативні об’єкти (перешкоди або «шкідливі» елементи);
- службові об’єкти (елементи, що впливають на механіку гри, наприклад бонуси або множники очок);
- декоративні об’єкти (елементи фону, що не впливають на геймплей).

Кожна з цих категорій виконує окрему функціональну роль у структурі гри. Позитивні об’єкти є основним джерелом прогресу гравця, оскільки саме їх взаємодія забезпечує нарахування очок. Негативні об’єкти, навпаки, створюють ризик втрати ресурсів або завершення гри, що формує елемент виклику. Подібна двоїста система широко використовується в іграх Plants vs. Zombies та Cut the Rope, де баланс між корисними та шкідливими об’єктами визначає складність геймплею.

Важливим аспектом проєктування є візуальна диференціація об’єктів. Кожен тип ігрового елемента повинен мати чітко відмінні характеристики форми, кольору та анімації, що дозволяє гравцеві швидко ідентифікувати його функцію без додаткових пояснень. Наприклад, у позитивних об’єктів може використовуватися яскрава тепла кольорова палітра, тоді як негативні об’єкти можуть бути оформлені в темніших або контрастних кольорах.

Також важливим є визначення поведінкової моделі кожного об’єкта. Це включає траєкторію руху, швидкість, реакцію на взаємодію з гравцем та умови зникнення зі сцени. У аркадних іграх часто застосовується принцип

передбачуваної фізики з елементами випадковості, що дозволяє зберігати баланс між складністю та контрольованістю ігрового процесу.

Окрему роль відіграє система колізій, яка визначає взаємодію між ігровими об’єктами та персонажем. У 2D іграх зазвичай використовується спрощена система зіткнень, заснована на геометричних фігурах (кола, прямокутники), що дозволяє оптимізувати продуктивність і забезпечити стабільність геймплею [15]. Подібні рішення застосовуються у багатьох інді-проєктах, таких як Celeste, де точність взаємодії має критичне значення.

Крім того, система ігрових об’єктів тісно пов’язана з системою винагород і прогресії. Взаємодія з об’єктами повинна безпосередньо впливати на стан гри — нарахування очок, збільшення складності або активацію додаткових механік. Це створює замкнений цикл «дія — реакція — результат», який є основою аркадного геймплею.

Отже, система ігрових об’єктів у «Apple Madness» є структурованою взаємопов’язаною моделлю, що забезпечує реалізацію основної ігрової механіки. Її правильне проєктування дозволяє досягти балансу між простотою сприйняття, динамікою взаємодії та загальною ігровою складністю, що є критично важливим для успішного функціонування аркадної гри.

2.4. Проєктування персонажа та візуального стилю гри

Формування художнього стилю та кольорової палітри є одним із ключових етапів візуального проєктування відеогри, оскільки саме ці елементи визначають загальне емоційне сприйняття ігрового світу, його впізнаваність та ступінь занурення користувача. Художній стиль у відеоіграх можна розглядати як систему візуальних принципів, що об’єднують форму, колір, текстуру, ступінь деталізації та спосіб зображення простору в єдину композиційну модель.

У сучасному геймдизайні художній стиль виконує не лише естетичну функцію, але й комунікативну. Він передає інформацію про жанр гри, її темп, наративний тон та цільову аудиторію. Наприклад, мінімалістичний та контрастний стиль гри «Limbo» одразу формує відчуття тривожності та невідомості, тоді як яскравий і насичений стиль «Fortnite» створює асоціації з динамікою, доступністю та аркадністю [16].

Рис. 2.5. Візуальний стиль гри «Limbo»

Важливим теоретичним підходом до формування художнього стилю є принцип візуальної цілісності (visual consistency). Він передбачає, що всі елементи гри — персонажі, середовища, інтерфейс та об’єкти — повинні бути виконані в єдиній стилістичній системі. Порушення цієї цілісності призводить

до когнітивного дисонансу у гравця, коли окремі елементи світу сприймаються як «чужорідні» або випадкові.

У процесі розробки гри «Apple Madness» художній стиль визначається через поєднання стилізації та спрощеної форми з елементами комічної гіперболізації. Такий підхід дозволяє одночасно забезпечити легку читабельність об’єктів і створити впізнаваний візуальний образ гри. Аналогічні рішення можна спостерігати в іграх Angry Birds або Cut the Rope, де прості форми та яскрава стилізація сприяють швидкому сприйняттю ігрових ситуацій.

Рис 2.6. Приклади ігор які слугували натхненням стилю «Apple Madness»

Окрему роль у формуванні стилю відіграє рівень деталізації (level of detail). У 2D-іграх деталізація часто свідомо обмежується для того, щоб уникнути перевантаження візуального поля користувача. Це особливо важливо в мобільних або казуальних іграх, де швидкість сприйняття інформації має критичне значення. Наприклад, у грі Monument Valley використовується спрощена геометрія та чисті кольорові площини, що дозволяє гравцю концентруватися на просторових задачах, а не на зайвих деталях.

Наступним важливим аспектом є психологія кольору [19]. Колір у відеоіграх виступає інструментом невербальної комунікації, який впливає на емоційний стан гравця та його поведінкові реакції. Теоретично кольори поділяються на теплі, холодні та нейтральні, кожна з яких виконує окрему функцію в композиції. Теплі кольори (червоний, оранжевий, жовтий) зазвичай асоціюються з активністю, небезпекою або енергією, тоді як холодні (синій, блакитний, фіолетовий) створюють відчуття спокою, дистанції або технологічності.

У грі Journey кольорова палітра безпосередньо змінюється залежно від локації, що формує емоційну динаміку подорожі гравця. Подібний підхід

використовується і в Ori and the Blind Forest, де колір є ключовим інструментом підсилення драматичних моментів та підкреслення змін у світі гри.

У контексті «Apple Madness» кольорова палітра формується на основі контрастного поєднання природних відтінків (зелені, червоні, жовті тони) з додатковими акцентними кольорами, які підкреслюють ігрові інтерактивні елементи. Такий підхід дозволяє одночасно зберегти «органічність» тематики (фрукти як базовий мотив) та забезпечити чітке розмежування ігрових об’єктів.

Рис. 2.7. Палітра гри

Важливим теоретичним принципом також є домінанта кольору (color dominance), яка передбачає наявність основного кольору або обмеженої групи кольорів, що формують загальний настрій гри. Надмірна кількість кольорів без системи призводить до візуального шуму, що знижує читабельність ігрового простору. Саме тому в професійних проєктах зазвичай використовується обмежена палітра з чіткою ієрархією відтінків.

Таким чином, формування художнього стилю та кольорової палітри є комплексним процесом, який поєднує художні, психологічні та функціональні аспекти. Він визначає не лише зовнішній вигляд гри, але й спосіб взаємодії гравця з візуальною інформацією, що є критично важливим для створення цілісного ігрового досвіду.

Продовжуючи розгляд формування художнього стилю, важливо звернути увагу на роль стилістичної уніфікації між різними компонентами гри. У сучасних відеоіграх художній стиль не обмежується лише персонажами або фонами - він охоплює також інтерфейс, анімацію, ефекти частинок і навіть спосіб подачі текстової інформації. Всі ці елементи повинні підпорядковуватися єдиній візуальній логіці, що формує цілісний досвід сприйняття.

Рис. 2.8. Фонове зображення гри

Одним із важливих аспектів є взаємодія стилю та функціональності. У добре спроєктованих іграх візуальний стиль не лише прикрашає гру, але й допомагає гравцю орієнтуватися в ігровому просторі. Наприклад, у грі Dead Cells чітке розділення кольорів між небезпечними зонами, інтерактивними об’єктами та фоном дозволяє гравцеві швидко приймати рішення навіть у динамічних ситуаціях. Подібний підхід використовується і в Hades, де контраст між персонажами, ворогами та середовищем підсилює читабельність бою.

У межах проєкту «Apple Madness» цей принцип реалізується через чітке відокремлення ігрових елементів за допомогою кольору та стилістичних акцентів. Інтерактивні об’єкти (наприклад, яблука, бонуси або перешкоди) повинні мати візуально відмінні характеристики, які дозволяють гравцю миттєво їх ідентифікувати. Це особливо важливо для казуального жанру, де швидкість сприйняття інформації є критичною.

Окремо слід розглянути питання контрастності та візуальної ієрархії. Контраст у відеоіграх використовується для виділення ключових елементів і спрямування уваги гравця. Він може бути колірним, світлотним або

формальним. Наприклад, у грі «Cuphead» високий рівень контрасту між персонажами та фоном забезпечує їхню миттєву впізнаваність навіть у складних сценах з великою кількістю візуальних ефектів.

У теоретичному плані візуальна ієрархія кольору базується на принципі домінанти, акценту та нейтрального фону. Домінанта формує загальний настрій сцени, акценти використовуються для ключових інтерактивних об’єктів, а нейтральні тони слугують для фону та допоміжних елементів. Така структура дозволяє уникати перевантаження візуального поля та підвищує ефективність сприйняття інформації [17].

Ще одним важливим аспектом є емоційна синхронізація кольору та геймплею. Колір може змінюватися відповідно до ігрових подій, підсилюючи емоційний ефект. Наприклад, у грі «Celeste» зміна кольорових схем рівнів супроводжує психологічний розвиток персонажа та підкреслює складність випробувань. У «Apple Madness» подібний підхід може бути реалізований через поступову зміну палітри залежно від складності рівня або швидкості ігрового процесу.

Також варто враховувати адаптацію кольорової системи до різних пристроїв і умов перегляду. У сучасній індустрії важливо, щоб гра залишалася читабельною як на великих моніторах, так і на мобільних екранах. Це вимагає тестування контрастності, насиченості та загальної кольорової гармонії в різних умовах освітлення та масштабування.

Окрему роль відіграє стилістична оптимізація для технічних обмежень. У 2D-іграх часто застосовується принцип економії ресурсів, коли складність візуальних елементів компенсується грамотним використанням кольору та форми. Це дозволяє досягти високої естетичної якості навіть при обмежених технічних можливостях, що характерно для інді-розробки.

Таким чином, формування художнього стилю та кольорової палітри є не лише творчим, але й аналітичним процесом, який поєднує естетику, психологію сприйняття та функціональну логіку геймдизайну. У випадку «Apple Madness» цей процес спрямований на створення простої, але виразної візуальної системи, яка забезпечує швидке сприйняття ігрових об’єктів і підтримує загальну динаміку геймплею.

РОЗДІЛ 3. РОЗРОБКА КОМП'ЮТЕРНОЇ ГРИ «APPLE MADNESS»

3.1. Розробка графічної складової гри

Графічна складова комп'ютерної гри «Apple Madness» є одним із ключових компонентів, що визначає її візуальну привабливість, читабельність і загальне сприйняття користувачем. У 2D аркадних іграх саме графіка виконує не лише декоративну функцію, але й безпосередньо впливає на ігровий процес, оскільки допомагає гравцеві швидко розпізнавати об'єкти, оцінювати ситуацію на екрані та приймати рішення.

У процесі розробки графічної складової гри було враховано основні принципи візуального дизайну, такі як контрастність, простота форм, узгодженість стилю та функціональна зрозумілість кожного елемента. Особлива увага приділялась тому, щоб усі графічні об'єкти були стилістично єдиними та відповідали загальній концепції гри, орієнтованій на динамічний аркадний геймплей.

Окремим важливим аспектом є оптимізація графіки під реальний ігровий процес. У 2D-іграх, особливо аркадного типу, надмірна деталізація може ускладнювати сприйняття ігрової сцени, тому візуальні елементи були спрощені, але при цьому збережено їхню виразність. Подібний підхід широко використовується в сучасних інді-проєктах, таких як Cuphead або Stardew Valley, де стилізація відіграє ключову роль у формуванні ідентичності гри.

Для реалізації графічної частини використовувалися інструменти Adobe Animate, що дозволили створювати спрайти, анімації та UI-елементи в єдиному робочому середовищі. Це забезпечило узгодженість стилю та спростило процес інтеграції графіки в ігровий рушій Godot Engine.

Створення спрайтів персонажа

Створення спрайтів персонажа є базовим етапом формування графічної складової 2D-ігри «Apple Madness», оскільки саме спрайти визначають зовнішній вигляд головного героя в різних станах його взаємодії з ігровим середовищем. Спрайт у контексті 2D-графіки являє собою окреме зображення

або кадр анімації, який використовується для відображення персонажа в конкретний момент часу.

У процесі розробки було визначено набір основних станів персонажа, необхідних для коректного відображення його поведінки в грі. До них належать: стан спокою, рух ліворуч і праворуч, реакція на взаємодію з ігровими об’єктами, а також додаткові анімаційні стани, пов’язані з ігровими подіями. Такий підхід дозволяє забезпечити логічну послідовність поведінки персонажа та підвищує рівень його візуальної зрозумілості для користувача.

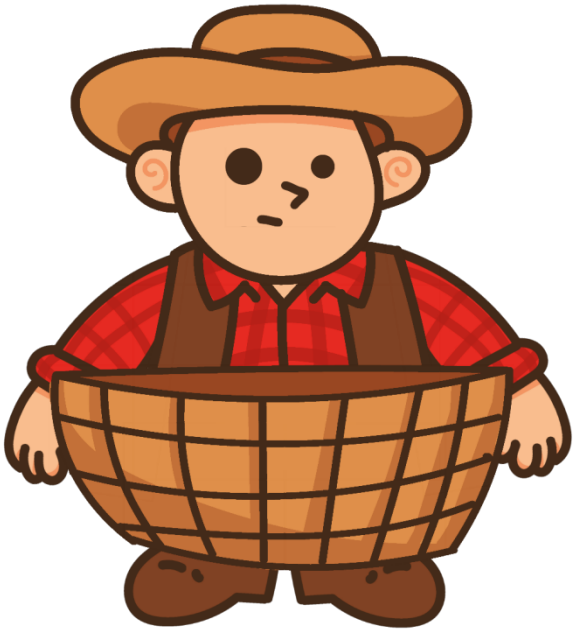

Рис. 3.1. Головний персонаж

При створенні спрайтів особлива увага приділялась уніфікації стилю, пропорційності елементів та читабельності силуету. Важливо, щоб персонаж залишався легко впізнаваним навіть у зменшеному масштабі, оскільки в ігровому процесі об’єкти часто відображаються на різних рівнях масштабування. З цією метою використовувались спрощені форми, чіткі контури та обмежена кількість дрібних деталей.

Технічно спрайти створювались у середовищі Adobe Animate, що дозволило працювати з покадровою структурою анімації та одразу перевіряти

поведінку персонажа в динаміці. Кожен стан персонажа реалізовувався як окремий набір кадрів, об’єднаних у відповідні анімаційні цикли. Це забезпечило зручність подальшої інтеграції у рушій Godot Engine, де спрайти використовуються як окремі текстурні ресурси.

Рис. 3.2. Кадри деяких анімацій головного героя

Окремим аспектом є оптимізація кількості кадрів в анімаційних циклах. Надмірна кількість зображень може негативно впливати на продуктивність, тому було знайдено баланс між плавністю анімації та ресурсною ефективністю. Такий підхід є стандартною практикою у розробці 2D-ігор, де важливо зберігати стабільну продуктивність на різних пристроях.

Створення ігрових об’єктів

Ігрові об’єкти у проєкті «Apple Madness» є ключовими інтерактивними елементами, що формують основну механіку гри та безпосередньо впливають на ігровий процес. Вони виступають як носії ігрової логіки, з якими взаємодіє головний персонаж, і забезпечують реалізацію базового циклу гри, пов’язаного зі збором або уникненням об’єктів.

Основним типом ігрових об’єктів у даному проєкті є яблука, які виконують роль цільових елементів для взаємодії. Саме їх поява, рух та взаємодія з персонажем формують основну динаміку гри. Яблука виступають як позитивні об’єкти, за які гравець отримує ігрові бали або інші винагороди, що стимулює активність і підтримує інтерес до ігрового процесу.

Під час розробки візуального вигляду яблук основна увага приділялася їхній простоті та миттєвій впізнаваності. Оскільки ігровий процес є динамічним, об’єкти повинні легко ідентифікуватися навіть у русі та при невеликому розмірі на екрані. З цією метою використовувались максимально спрощені форми з чітким силуетом, контрастним кольором та мінімальною кількістю деталей, що відповідає принципам читабельності 2D-ігрової графіки.

Окрему увагу було приділено кольоровому рішенню яблук. Колір використовується як засіб швидкої візуальної комунікації, тому було обрано насичені та контрастні відтінки, які дозволяють об’єкту виділятися на фоні ігрового середовища. Такий підхід забезпечує швидке розпізнавання об’єктів навіть у складних сценах з великою кількістю елементів.

Рис. 3.3. Ігрові об’єкти

З технічної точки зору яблука реалізовані як окремі ігрові сутності у рушії Godot Engine, що дозволяє задавати їм індивідуальні параметри поведінки. До таких параметрів належать швидкість появи, траєкторія руху, колізійні властивості та реакція на зіткнення з персонажем. При взаємодії з головним героєм активується відповідна логіка, що забезпечує нарахування очок та видалення об’єкта зі сцени.

Додатково було враховано аспект випадковості появи яблук, що дозволяє уникнути передбачуваності ігрового процесу. Генерація об’єктів відбувається на різних позиціях і з різними інтервалами часу, що створює ефект неперервної динаміки та підтримує ігрову напругу.

Розробка елементів інтерфейсу

Елементи користувацького інтерфейсу в грі «Apple Madness» виконують функцію візуального посередника між ігровою системою та гравцем,

забезпечуючи передачу інформації про поточний стан гри, результати взаємодії та доступні дії. Інтерфейс у 2D-аркадних іграх є критично важливим компонентом, оскільки він повинен бути максимально зрозумілим, ненав’язливим і водночас інформативним.

Під час проєктування інтерфейсу основний акцент був зроблений на принципах мінімалізму та функціональної читабельності. Це означає, що кожен елемент інтерфейсу має чітке призначення і не перевантажує ігровий екран зайвою інформацією. Такий підхід дозволяє зберегти фокус гравця на основному ігровому процесі, не відволікаючи його від взаємодії з ігровими об’єктами.

До основних елементів інтерфейсу належать відображення кількості набраних балів, індикатор ігрового прогресу, а також елементи управління, що дозволяють взаємодіяти з грою (наприклад, кнопки старту, паузи та перезапуску). Кожен із цих елементів має чітко визначене місце на екрані, що забезпечує швидке сприйняття інформації без необхідності додаткового аналізу з боку гравця.

Особлива увага приділялася візуальній узгодженості інтерфейсу з загальним художнім стилем гри. Усі UI-елементи виконані в єдиній стилістиці, що відповідає кольоровій гамі та формальним особливостям ігрового середовища. Це дозволяє інтегрувати інтерфейс у загальну візуальну систему гри, уникаючи відчуття «відокремленого шару», який не пов’язаний з ігровим світом.

Рис. 3.4. Користувацький інтерфейс гри

З технічної точки зору елементи інтерфейсу реалізовані як окремі UI-компоненти у рушії Godot Engine. Це дозволяє гнучко керувати їхньою поведінкою, змінювати значення в реальному часі та адаптувати інтерфейс до різних станів гри. Наприклад, при зміні ігрового стану (гра активна, пауза,

завершення) інтерфейс динамічно оновлюється, відображаючи відповідну інформацію.

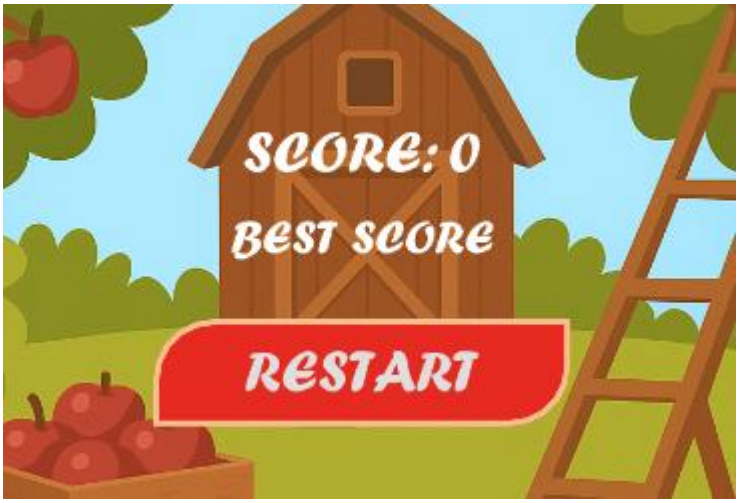

Рис. 3.5. Вікно перезапуску гри

Також важливим аспектом є забезпечення зручності сприйняття інтерфейсу на різних пристроях і роздільних здатностях екрана. Для цього використовувались відносні позиції елементів та масштабування, що дозволяє зберігати пропорційність і читабельність незалежно від розміру вікна гри.

Створення анімацій персонажа та об’єктів

Анімація у грі «Apple Madness» є одним із ключових елементів, що забезпечує відчуття динаміки, «живості» ігрового світу та підсилює загальне візуальне сприйняття. У 2D-іграх анімація виконує не лише декоративну роль, але й є важливим інструментом візуальної комунікації, який дозволяє гравцеві швидко розуміти стан об’єктів, їхню поведінку та реакцію на взаємодію.

У процесі розробки анімацій було створено цикли руху для головного персонажа, а також анімаційні ефекти для ігрових об’єктів, зокрема яблук. Для персонажа реалізовано базові стани руху, які включають плавну зміну положення елементів тіла між ключовими кадрами. Такий підхід дозволяє досягти природності руху навіть при відносно невеликій кількості кадрів, що є важливим для оптимізації продуктивності гри.

Особлива увага приділялася принципу плавності анімації. Перехід між кадрами був побудований таким чином, щоб уникнути різких змін форми або

положення персонажа, що могло б негативно вплинути на сприйняття гри. Це досягається шляхом правильного розподілу ключових кадрів і використання рівномірного інтервалу між ними.

Анімація яблук як ігрових об’єктів була реалізована у більш простій формі. Основною метою тут є привернення уваги гравця та створення відчуття руху в ігровому середовищі. Для цього використовувалися легкі ефекти коливання, обертання або вертикального руху, які не перевантажують сцену, але роблять об’єкти більш помітними та інтерактивними.

З технічної точки зору анімації були створені у середовищі Adobe Animate, після чого експортовані у форматі, придатному для використання у рушії Godot Engine. У самому рушії анімації інтегрувалися як окремі ресурси, що дозволило гнучко керувати їхнім запуском залежно від ігрових подій. Наприклад, при зіткненні персонажа з яблуком активується відповідний анімаційний ефект збору об’єкта.

Важливим аспектом є також узгодження анімацій з ігровою логікою. Кожна анімація повинна не лише виглядати естетично, але й відповідати функціональному стану об’єкта. Таким чином забезпечується синхронізація візуальної та програмної частин гри, що є критично важливим для цілісного ігрового досвіду.

3.2. Реалізація програмної складової гри

Програмна складова гри «Apple Madness» є основою функціонування всього ігрового процесу, оскільки саме вона забезпечує взаємодію між ігровими об’єктами, керування персонажем, обробку подій та реалізацію ігрових правил. У межах даного етапу відбувається перетворення попередньо розробленої графічної та концептуальної частини у повноцінний інтерактивний програмний продукт.

Реалізація програмної логіки здійснювалась у середовищі Godot Engine, яке надає зручні інструменти для створення 2D-ігор, включаючи систему сцен, вузлів та вбудовану мову програмування GDScript. Така архітектура дозволяє будувати модульну структуру гри, де кожен елемент (персонаж, об’єкти, інтерфейс) функціонує як окремий компонент із власною логікою.

Особливу увагу було приділено організації взаємодії між різними системами гри. Для цього використовувались сигнали та події, які дозволяють різним частинам гри обмінюватися інформацією без жорсткої прив’язки між собою. Це забезпечує гнучкість коду та спрощує подальше масштабування проєкту.

Програмна частина також включає реалізацію базових ігрових механік, таких як рух персонажа, генерація об’єктів, система підрахунку очок, обробка зіткнень та керування станами гри (активна гра, пауза, завершення). Усі ці елементи інтегровані в єдину логічну систему, що забезпечує стабільну роботу гри.

Програмування керування персонажем

На етапі розробки гри «Apple Madness» програмування керування персонажем було реалізовано як частина загальної системи ігрової логіки в рушії Godot Engine. Основним керованим об’єктом виступає кошик, який безпосередньо взаємодіє з ігровими елементами та є центральним інструментом гравця для збору об’єктів.

Реалізація керування базується на структурі сцен і вузлів, що дозволяє розділити логіку гри на окремі компоненти. Головний скрипт сцени містить змінні стану гри, такі як кількість очок, кількість життів та статус активності гри. Саме ці змінні визначають, коли персонаж може взаємодіяти з ігровим

середовищем, а коли керування має бути заблоковане (наприклад, у меню або на екрані паузи).

У процесі розробки було реалізовано систему станів гри, яка безпосередньо впливає на поведінку персонажа. При запуску гри активується змінна game_active, що дозволяє кошику реагувати на події та взаємодіяти з об’єктами. У неактивному стані керування фактично блокується через механізм паузи рушія (get_tree().paused), що забезпечує коректну зупинку всієї ігрової логіки.

```
func _process(delta):
    if state == "lose":
        return

    var moving = false

    if Input.is_action_pressed("right"):
        position.x += speed * delta
        moving = true
        $AnimatedSprite2D.flip_h = false
    elif Input.is_action_pressed("left"):
        position.x -= speed * delta
        moving = true
        $AnimatedSprite2D.flip_h = true

    position.x = clamp(position.x, 0, 800)
```

Рис. 3.6. Функція руху персонажа

Окрему роль відіграє обробка зіткнень між кошиком та ігровими об’єктами. При входженні об’єкта в зону взаємодії викликається відповідна функція, яка визначає тип об’єкта та застосовує відповідні зміни до стану гри. Таким чином, керування персонажем у даному проєкті реалізовано не через складну фізику руху, а через систему подій і логічних умов, що є типовим підходом для 2D-аркадних ігор.

Також у процесі розробки було інтегровано анімаційне управління персонажем. Залежно від стану гри кошик перемикається між різними анімаціями, наприклад, базовим станом очікування та анімацією програшу. Це дозволяє візуально підкреслити зміну ігрових подій і зробити поведінку персонажа більш зрозумілою для користувача.

Важливим елементом реалізації є прив’язка керування до загальної ігрової логіки. Персонаж активний лише у випадку, коли гра знаходиться у

стані запуску, що запобігає некоректній взаємодії під час паузи, меню або завершення гри. Такий підхід забезпечує стабільність роботи програми та узгодженість усіх ігрових систем.

Реалізація механіки генерації ігрових об’єктів

Однією з ключових складових розробленої гри є система генерації ігрових об’єктів, яка забезпечує динамічну появу елементів у межах ігрової сцени та формує основний ігровий процес. У межах проєкту «Apple Madness» ця механіка відповідає за створення об’єктів типу «яблуко» та їхніх варіацій, які відрізняються за поведінкою та впливом на стан гри.

Генерація об’єктів реалізована за допомогою функції створення нових екземплярів сцени (інстансів), що дозволяє динамічно додавати об’єкти під час виконання гри. Кожен об’єкт створюється на основі заздалегідь підготовленої сцени, після чого йому задаються випадкові початкові параметри, зокрема позиція появи, швидкість падіння, тип та напрямок руху. Це дозволяє забезпечити непередбачуваність ігрового процесу та підвищити його варіативність.

```
func create_ball():
    var ball = load("res://scenes/ball.tscn").instantiate()
    ball.position.x = randi_range(40, window_size.x -60)
    ball.position.y = randi_range(-40, -100)
    ball.speed = randi_range(100, 250) + score * 5

    var r = randi_range(1, 110)

    if r <= 50:
        ball.ball_type = "red"
    elif r <= 90:
        ball.ball_type = "green"
    elif r <= 100:
        ball.ball_type = "blue"
    else:
        ball.ball_type = "death"


    ball.rotation_degrees = randi_range(-60, 60)

    if randi() % 2 == 0:
        ball.scale.x = -ball.scale.x

    get_tree().current_scene.add_child(ball)
```

Рис 3.7. Фрагмент коду генерації ігрового об’єкта класу яблуко

Особливу роль у цій системі відіграє випадковість, яка використовується для визначення характеристик кожного об’єкта. Зокрема, за допомогою

генератора випадкових чисел задається тип об’єкта: звичайне яблуко, бонусне яблуко або «шкідливий» об’єкт. Такий підхід дозволяє балансувати складність гри та формувати різні сценарії взаємодії гравця з ігровими елементами.

```
1   extends Area2D
2
3   var speed = 0
4   var ball_type = "red"
5
6   func _ready():
7       add_to_group("BALLS")
8       if ball_type == "red":
9           $Sprite2D.texture = load("res://assets/apple/яблуко червоне.png")
10      elif ball_type == "green":
11          $Sprite2D.texture = load("res://assets/apple/яблуко зелене.png")
12      elif ball_type == "blue":
13          $Sprite2D.texture = load("res://assets/apple/heart.png")
14      elif ball_type == "death":
15          $Sprite2D.texture = load("res://assets/apple/death.png")
16
17
18  func _process(delta):
19      position.y += speed * delta
```

Рис. 3.8. Фрагмент коду генерації ігрового об’єкта класу яблуко

Крім типу об’єкта, випадковим чином визначається його горизонтальна позиція у межах ширини ігрового вікна. Це забезпечує рівномірне розподілення об’єктів по сцені та запобігає передбачуваним патернам появи. Додатково варіюється швидкість падіння об’єктів, яка може залежати від поточного ігрового рахунку, що поступово збільшує складність гри в процесі проходження.

Після створення об’єкт автоматично додається до активної ігрової сцени як дочірній елемент головного вузла. Це дозволяє йому взаємодіяти з іншими елементами гри, зокрема з ігровим персонажем та межами сцени. Для контролю кількості активних об’єктів використовується механізм групування та видалення, який забезпечує очищення сцени від об’єктів, що втратили актуальність.

Окремо слід зазначити, що система генерації інтегрована з основним ігровим циклом. Після знищення або взаємодії з об’єктом одразу ініціюється створення нового, що підтримує безперервність ігрового процесу. Таким чином формується замкнений цикл: поява об’єкта → взаємодія → видалення → нова генерація.

Завдяки такій структурі реалізації, система генерації об’єктів у «Apple Madness» є гнучкою, масштабованою та легко модифікованою, що дозволяє у майбутньому розширювати кількість типів об’єктів або змінювати правила їх появи без суттєвих змін у загальній архітектурі гри.

Розробка системи підрахунку очок та життів

Система підрахунку очок та життів є однією з ключових складових ігрової логіки, оскільки вона безпосередньо визначає прогрес гравця, умови завершення гри та загальну динаміку ігрового процесу. У розробленій грі «Apple Madness» ця система реалізована як частина центрального скрипту керування грою та інтегрована з усіма основними ігровими подіями.

Основними ігровими параметрами виступають змінні, що зберігають поточний рахунок гравця та кількість життів. Рахунок (score) збільшується у випадку успішної взаємодії гравця з певними об’єктами, зокрема коли персонаж ловить «правильні» яблука. Життя (lives), у свою чергу, зменшуються при помилкових діях або контакті з негативними об’єктами. Таким чином формується базова система винагороди та покарання, яка стимулює уважність та швидкість реакції гравця.

```
func _on_basket_area_entered(area):
    if area.ball_type == "red":
        score += 1
        $Music/point.play()
    elif area.ball_type == "green":
        lose_life()
        $Music/death.play()
    elif area.ball_type == "blue":
        lives += 1
        $Music/goldapple.play()
    elif area.ball_type == "death":
        lives = 0
        $basket.play_lose_animation()
        game_over()
    area.queue_free()
    create_ball()
    update_ui()
```

Рис. 3.9. Функція перевірки колвізій

Оновлення значень очок і життів відбувається у реальному часі під час ігрового процесу. Після кожної взаємодії з об’єктом викликається функція

оновлення інтерфейсу, яка синхронізує внутрішні змінні з візуальним відображенням на екрані. Це забезпечує миттєвий зворотний зв’язок для користувача, що є важливим елементом ігрового дизайну.

Окремо реалізовано механізм перевірки критичних умов завершення гри. У випадку, якщо кількість життів досягає нуля, активується сценарій завершення гри, який блокує подальшу взаємодію з ігровими об’єктами, зупиняє ігровий процес та викликає екран результатів. Таким чином система життів виконує роль обмежувального ресурсу, що визначає тривалість ігрової сесії.

Також у системі передбачено збереження найкращого результату (best score), який оновлюється лише у випадку перевищення попереднього значення. Це дозволяє реалізувати елемент довготривалої мотивації гравця та стимулює повторні проходження гри з метою покращення результату.

Важливою частиною реалізації є синхронізація логіки підрахунку з візуальним інтерфейсом. Значення рахунку та життів відображаються на екрані через текстові елементи інтерфейсу, які оновлюються після кожної зміни відповідних змінних. Такий підхід забезпечує узгодженість між внутрішнім станом гри та її візуальним представленням.

Розробка інтерфейсу користувача

Основними елементами інтерфейсу є відображення поточного рахунку (Score) та кількості життів (Lives). Ці дані оновлюються в реальному часі та синхронізуються з внутрішніми змінними ігрової логіки. Оновлення текстових полів здійснюється через окрему функцію оновлення інтерфейсу, яка змінює значення відповідних UI-елементів після кожної ігрової події. Таким чином забезпечується постійний зворотний зв’язок для користувача.

Окрему роль відіграють екранні стани гри, які реалізовані у вигляді окремих інтерфейсних сцен: стартовий екран, екран паузи та екран завершення гри. Стартовий екран відображається на початку гри та блокує ігрову взаємодію до моменту запуску. Після натискання кнопки старту активується основний ігровий процес, а стартовий екран приховується.

Екран паузи використовується для тимчасового призупинення гри. При його активації ігровий процес зупиняється через механізм призупинення дерева сцени, що блокує всі фізичні та логічні процеси. Це дозволяє гравцю тимчасово припинити гру без втрати прогресу.

```
# кнопки
func _on_StartButton_pressed():
    print(123)
    start_game()

func _on_PauseButton_pressed():
    pause_game()

func _on_ResumeButton_pressed():
    resume_game()

func _on_RestartButton_pressed():
    restart_game()
```

Рис. 3.10. Функції керування інтерфейсом

Екран завершення гри активується у випадку втрати всіх життів. На цьому екрані відображається фінальний результат, а також найкращий досягнутий результат за всі спроби. Дані передаються з основної ігрової логіки до UI-елементів перед відображенням екрану результатів.

Керування інтерфейсом реалізовано через набір кнопок, які відповідають за запуск гри, паузу, відновлення та перезапуск. Кожна кнопка викликає відповідну функцію в основному скрипті гри. Наприклад, кнопка старту ініціює запуск ігрового процесу, а кнопка паузи активує відповідний режим призупинення.

У структурі коду інтерфейс тісно пов’язаний із логікою гри. Це видно через прямі звернення до UI-елементів, таких як текстові поля рахунку та життів, а також через керування видимістю різних екранних станів. Наприклад, при запуску гри відбувається приховування стартового екрану та відображення кнопки паузи, тоді як при завершенні гри активується екран результатів і приховується ігрова кнопка.

Створення меню та екрана завершення гри

Меню та екран завершення гри є важливими компонентами

користувацького інтерфейсу, оскільки вони забезпечують організацію ігрового процесу до початку гри та після її завершення. У розробленій грі «Apple Madness» ці елементи реалізовані як окремі інтерфейсні екрани, що змінюють свій стан залежно від етапу гри.

Стартове меню виконує функцію первинної взаємодії з користувачем. Воно відображається одразу після запуску гри та блокує основний ігровий процес до моменту натискання кнопки старту. У межах реалізації меню використовується механізм керування видимістю UI-елементів, що дозволяє показувати або приховувати відповідні сцени інтерфейсу. Під час активного стартового меню ігрова сцена перебуває у призупиненому стані, що запобігає будь-яким взаємодіям з ігровими об’єктами до початку гри.

Після натискання кнопки старту відбувається ініціалізація основних параметрів гри, зокрема рахунку, життів та стану активності гри. Одночасно стартове меню приховується, а на екран виводяться елементи ігрового інтерфейсу. Таким чином забезпечується плавний перехід від екрана меню до активного ігрового процесу.

Екран завершення гри активується у випадку, коли гравець втрачає всі доступні життя. Цей екран виконує роль підсумкового інтерфейсу, де відображаються результати поточної ігрової сесії, а також найкращий досягнутий результат за всі попередні спроби. Відображення цих даних реалізовано через текстові елементи, значення яких оновлюються перед появою екрана завершення гри.

Важливою частиною логіки є збереження та порівняння результатів. Якщо поточний рахунок перевищує попередній найкращий результат, він автоматично оновлюється. Це додає елементу повторної іграбельності та мотивує користувача до покращення власних результатів у наступних сесіях.

Окрему роль відіграє система кнопок на екрані завершення гри. Вона дозволяє користувачу перезапустити гру або повернутися до початкового стану. При повторному запуску відбувається повне скидання ігрових параметрів та повторна ініціалізація ігрового процесу, що забезпечує коректний перезапуск без залишкових даних попередньої сесії.

3.3. Тестування та оптимізація програмного продукту

Етап тестування є завершальною частиною розробки програмного продукту та спрямований на перевірку коректності роботи всіх ігрових механік, стабільності застосунку та відповідності реалізованого функціоналу поставленим вимогам. У межах розробки гри «Apple Madness» тестування виконувалося як у процесі розробки (внутрішнє тестування), так і після створення основної версії гри за участі сторонніх користувачів.

Основною метою тестування було виявлення можливих помилок у роботі ігрових механік, перевірка стабільності генерації об’єктів, коректності підрахунку очок і життів, а також оцінка зручності користувацького інтерфейсу. Окрему увагу було приділено перевірці переходів між ігровими станами: старт, пауза, активна гра та завершення гри.

У процесі тестування також аналізувалась продуктивність гри, зокрема стабільність роботи при великій кількості одночасно активних ігрових об’єктів. Це дозволило оцінити ефективність реалізованої логіки видалення та повторного створення об’єктів.

Перевірка працездатності основних механік здійснювалася шляхом багаторазового проходження ігрового процесу та тестування всіх ключових сценаріїв взаємодії. Зокрема, перевірялися механіки руху та взаємодії ігрового персонажа з об’єктами, система підрахунку очок, зміна кількості життів, а також умови завершення гри.

Особлива увага приділялася перевірці коректності роботи функції генерації об’єктів, яка відповідає за динамічну появу елементів у сцені. Було підтверджено, що після взаємодії з об’єктом або його зникнення система стабільно створює нові елементи, що забезпечує безперервність ігрового процесу.

Також тестувалася система обробки зіткнень, яка визначає тип об’єкта та відповідну реакцію гри: збільшення рахунку, зменшення або додавання життів, а також завершення гри у випадку критичних подій. Усі сценарії показали коректну роботу без критичних помилок.

Додатково проводилося користувацьке тестування за участі одногрупників, які взаємодіяли з грою без попереднього ознайомлення з її внутрішньою логікою. Це дозволило оцінити інтуїтивність інтерфейсу та зрозумілість ігрових правил. За результатами тестування було отримано зворотний зв’язок щодо балансу складності та швидкості появи об’єктів, що частково було враховано при фінальній налаштуванні параметрів гри.

Подальший розвиток гри «Apple Madness» може здійснюватися в кількох напрямках, спрямованих на розширення функціональності та покращення ігрового досвіду користувача. Одним із основних напрямків є ускладнення ігрової механіки шляхом додавання нових типів об’єктів із унікальними властивостями, що впливатимуть на геймплей більш різноманітно.

Також перспективним є впровадження системи рівнів складності, яка буде поступово збільшувати швидкість появи об’єктів та змінювати їхню поведінку залежно від прогресу гравця. Це дозволить зробити ігровий процес більш динамічним і довготривалим.

Ще одним напрямком розвитку є вдосконалення візуальної складової гри, зокрема додавання більш складних анімацій, покращення ефектів взаємодії та розширення графічного стилю. Окремо можна розглянути інтеграцію звукових та візуальних ефектів для кожного типу об’єктів, що підвищить рівень занурення.

Крім того, можливим є впровадження системи рекордів із локальним або онлайн-збереженням результатів, що дозволить порівнювати досягнення різних гравців. Також можна додати систему досягнень, яка стимулюватиме користувача виконувати додаткові ігрові завдання.

Таким чином, розроблений проєкт має потенціал для подальшого розвитку як повноцінна аркадна гра з розширеним функціоналом, покращеною графікою та більш глибокою ігровою механікою.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було досягнуто поставленої мети — спроєктовано та розроблено двовимірну аркадну гру «Apple Madness» із використанням ігрового рушія Godot Engine, а також досліджено особливості створення подібних програмних продуктів у контексті сучасної індустрії розробки комп’ютерних ігор.

У ході дослідження проаналізовано сучасний стан GameDev-індустрії та визначено тенденції розвитку незалежної розробки ігор. Встановлено, що стрімке поширення доступних інструментів для створення інтерактивних застосунків сприяє зростанню популярності невеликих ігрових проєктів, зокрема двовимірних аркадних ігор. Саме такі проєкти є ефективним середовищем для набуття практичних навичок у сфері програмування, цифрового дизайну та проєктування користувацької взаємодії.

Досліджено особливості жанру 2D Arcade та визначено його основні характеристики: прості правила, швидкий темп ігрового процесу, інтуїтивно зрозуміле керування та орієнтацію на миттєвий зворотний зв’язок із користувачем. Аналіз популярних аркадних механік дозволив сформувати концепцію власного програмного продукту, засновану на взаємодії гравця з різними типами об’єктів і системі винагород та покарань.

У роботі проведено аналіз сучасних ігрових рушіїв і засобів розробки. За результатами порівняння функціональних можливостей, складності освоєння, ліцензійних умов та придатності до створення двовимірних ігор обґрунтовано вибір Godot Engine як основного середовища розробки. Визначено, що цей рушій є ефективним інструментом для створення навчальних та інді-проєктів завдяки відкритому вихідному коду, зручній системі сцен і підтримці спеціалізованих засобів для роботи з двовимірною графікою.

У процесі проєктування гри «Apple Madness» було сформовано її загальну концепцію, визначено цільову аудиторію, спроєктовано структуру програмного продукту та логіку взаємодії його компонентів. Розроблено

архітектуру основних сцен гри, спроєктовано головне меню, користувацький інтерфейс, систему ігрових об’єктів та принципи організації ігрового процесу.

Особливу увагу приділено створенню візуальної складової гри. Розроблено дизайн головного персонажа, спрайти ігрових об’єктів та елементи інтерфейсу, що забезпечують стилістичну цілісність програмного продукту. Використання Adobe Animate дозволило реалізувати необхідні графічні елементи та анімації, що позитивно вплинуло на виразність і динамічність гри.

Практичним результатом роботи стало створення повноцінного програмного продукту — двовимірної аркадної гри «Apple Madness». У середовищі Godot Engine реалізовано основні ігрові механіки: керування персонажем, генерацію об'єктів із різними властивостями, систему нарахування очок і життів, механізми завершення гри, а також елементи користувацького інтерфейсу. Розроблено головне меню, систему паузи та екран завершення гри, що забезпечують цілісність користувацького досвіду.

У межах дослідження проведено практичне тестування створеного програмного продукту за участі потенційних користувачів. Результати тестування підтвердили працездатність реалізованих механік, зрозумілість інтерфейсу та загальну стабільність роботи застосунку. Виявлені в процесі апробації зауваження були враховані під час фінального налаштування параметрів гри.

Отримані результати мають практичну цінність і можуть бути використані під час вивчення дисциплін, пов’язаних із програмуванням, комп’ютерною графікою та розробкою ігрових проєктів. Розроблена гра може слугувати демонстраційним прикладом реалізації аркадних механік засобами Godot Engine, а також основою для подальшого вдосконалення програмного продукту.

Перспективами розвитку проєкту є розширення функціональних можливостей гри шляхом додавання нових рівнів складності, системи досягнень, локального або мережевого збереження рекордів, збільшення кількості типів ігрових об’єктів, удосконалення звукового супроводу та

візуальних ефектів. Подальший розвиток гри може сприяти перетворенню навчального проєкту на завершений комерційно орієнтований продукт.

Таким чином, усі поставлені завдання кваліфікаційної роботи виконано в повному обсязі, а поставлену мету досягнуто. Результати дослідження підтверджують доцільність використання Godot Engine для розробки двовимірних аркадних ігор та демонструють можливість створення якісного програмного продукту із застосуванням сучасних засобів ігрової інженерії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бібічков І. Як розробляють ігри?. URL: https://lemon.school/blog/yak-rozroblyayut-igry.
2. Варич, М. О. Аналіз основних принципів та правил розробки геймдизайну. *Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації-2025/Матеріали V Всеукраїнської науково-технічної конференції молодих вчених, аспірантів і студентів, Одеса, 25-26 вересня 2025 р.-Одеса, Видавництво ОНТУ, 2025 р.–505 с. Збірник включає матеріали доповідей учасників конференції, які об'єднані за*, 2025, 448.
3. Гордієнко, А. В. Комп’ютерні ігри та їхні позитивні психологічні ефекти. *Наукові записки НаУКМА. Педагогічні, психологічні науки та соціальна робота*, 2017, 199: 58-62.
4. Ігрові жанри URL: https://training.qatestlab.com/blog/technicalarticles/games-genres/
5. Історія геймдеву XX століття: від Spacewar! до Sonic the Hedgehog. *Skvot.io*. URL: https://skvot.io/uk/blog/istoriya-geymdevu-hh-stolittyavid-spacewar-do-sonic-the-hedgehog.
6. Ковальчук, А. Платформи для розробки комп’ютерних ігор. *Актуальні питання сучасної інформатики: матеріали доп. VI Всеукр. наук.-практ. конф. з міжнар. участю, 18-19 листоп. 2021 р.* 2022. р. 78-82.
7. Корбут К. С., Брянцев О. А. Концепт-дизайн комп'ютерних ігор. Conference: Дизайн, візуальне мистецтво та творчість. Сучасні тенденції та технології. Матеріали І міжнародної науково-практичної конференції. Запорізький національний університет. 2022. 90 с. URL: https://www.researchgate.net/publication/366921467_KONCEPTDIZAJN_KOMP'UTERNIH_IGOR
8. Кудрявцева О. Деякі особливості розробки ігор. *Рекомендовано Вченою радою Житомирського державного університету імені Івана Франка, протокол № 3 від 4.02. 2022 р. Рецензенти*, 2022, 261.

9. Куценко А. Огляд сучасних програм комп’ютерної анімації. Наукова діяльність як шлях формування професійних компетентностей майбутнього фахівця (НПК-2018) : матеріали Міжнар. науково-практ. конф., м. Суми. Суми, 2018. С. 21–22.

10. Лугова Т. А., Блажко О. А. Проектування комп’ютерних ігор для навчання : навчальний підручник. Одеса : ФОП «Побута», 2019. 122 с.

11. Майстренко, Аліна Андріївна; карабін, Оксана Йосифівна. Особливості розробки 3d комп’ютерної гри у середовищі GODOT. *РЕДАКЦІЙНИЙ КОМІТЕТ*, 2025, 83.

12. Мельник О. Комп’ютерна анімація та 3D-моделювання: навчальний посібник. Умань : УДПУ ім. Павла Тичини, 2018. 141 с. URL: https://dspace.udpu.edu.ua/bitstream/123456789/16481/3/Anim.pdf.

13. Мови програмування, на яких написані популярні комп'ютерні ігри. *IT Step*. URL: https://kam.itstep.org/blog_3/programminglanguages-in-which-popular-computer-games-are-written.

14. Момот Р. Комп’ютерна анімація як засіб підтримки навчального процесу. Наукова діяльність як шлях формування професійних компетентностей майбутнього фахівця (НПК-2018) : матеріали Міжнар. науково-практ. конф., м. Суми, 6 груд. 2018 р. 2018. С. 65–66.

15. Мудрий, Я. П. Комп’ютерні ігри та їх класифікація. *Актуальні питання сучасної інформатики*, 2016, 2: 133-138.

16. Поливач Є. Ю. Зміст авторських прав на комп'ютерні програми та їх види. *Прикарпатський юридичний вісник*. 2021. № 1. С. 47–50. URL: https://doi.org/10.32837/pyuv.v0i1.730 (дата звернення: 15.06.2026).

17. Радайтіс, Б. А.; Яценко, О. І. Навчальні комп’ютерні ігри: правила розробки та етапи створення. In: *VI International scientific and practical conference «The aspects of contemporary scientific research that encompass both theoretical and practical components»*. 2024. p. 113-117.

18. Рендюк, С. П.; Колодін, В. М. *Розробка комп'ютерої гри у жанрі «Аркада» на платформі Unity*. 2019. PhD Thesis. Полтавський національний технічний університет імені Юрія Кондратюка.
19. Розробка персонажа комп'ютерної гри. Студія розробки ігор KLONA URL: https://klona.ua/uk/portfolio/rozrobka-personazha-dlya-kompyuternoyi-g
20. Хорошевська, І. О.; Лапіна, А. А. Особливості розробки інтерфейсів для мобільних ігор. 2023.
21. Чайка, Г. В. Позитивні впливи комп'ютерних ігор. *Актуальні проблеми психології: Збірник наукових праць Інституту психології імені ГС Костюка НАПН України. Том VI: Психологія обдарованості*, 2019, 16: 269-278.
22. Шеветовська А. ЯК СТВОРИТИ ПЕРСОНАЖ З НУЛЯ URL: https://skvot.io/uk/blog/kak-sozdat-personazha-s-nulya
23. Що Таке UX-UI Дизайн Ігрових Інтерфейсів Та Хто Ним Займається? URL: https://varto.school/blog/shho-take-ux-ui-dyzajn-igrovyh-interfejsiv-tahto-nym-zajmayetsya/
24. Animate посібник користувача. Adobe. URL: https://helpx.adobe.com/ua/animate/user-guide.html.
25. Computer animation: algorithms and techniques. Choice reviews online. 2008. Vol. 46, no. 02. P. 46–0934–46–0934. URL: https://doi.org/10.5860/choice.46-0934.
26. Computer animation: algorithms and techniques. Choice reviews online. 2008. Vol. 46, no. 02. P. 46–0934–46–0934. URL: https://doi.org/10.5860/choice.46-0934.
27. Csikszentmihalyi M. Flow: The Psychology of Optimal Experience. New York : Harper \& Row, 1990.
28. Fullerton T. Game Design Workshop: A Playcentric Approach to Creating Innovative Games. CRC Press, 2024.
29. Godot Engine. Godot Engine Documentation. URL: https://docs.godotengine.org.

30. Newzoo. Global Games Market Report. URL: https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report.
31. Philipe J. The Importance of Concept Art for Video Games. State Of The Games Industry: веб-сайт. URL: https://philipejohnson.wordpress.com/the-importance-ofconcept-art-for-video-games/
32. Rogers S. Level Up! The Guide to Great Video Game Design. Wiley, 2014.
33. Schell J. The Art of Game Design: A Book of Lenses. CRC Press, 2019.
34. Sharpe D. T. The psychology of color and design. Nelson-Hall, 1974
35. Sharpe D. T. The psychology of color and design. Nelson-Hall, 1974.
36. Sierra Rativa A., Postma M., Van Zaanen M. The Influence of Game Character Appearance on Empathy and Immersion: Virtual Non-Robotic Versus Robotic Animals. Simulation & Gaming. 2020. Vol. 51, no. 5. P. 685–711. URL: https://doi.org/10.1177/1046878120926694.
37. Tillman B. Creative character design. 2-ге вид. CRC Press, 2019. 224с
38. Video Game History URL: https://www.history.com/topics/inventions/historyof-video-games