МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: «Розробка ігрового додатку на двигуні Unity»

Виконав студент 4 курсу групи К-25
Спеціальність 122 комп'ютерні науки
Тарановський Дмитро Андрійович

Керівник асистент
Штефан Наталія Зінов'ївна

Консультант к.т.н., доцент
Великодний Станіслав Сергійович

Рецензент к.ф.-м.н., доцент
Ткач Тетяна Борисівна

Одеса 2020

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ECS – Entity Component System

HUD – Heads-Up Display

UI – User interface

UML Unified Modeling Language

Аркада – це жанр відеоігор

Геймплей – ігровий процес

Пули – (англ. Pool) особливі списки об'єктів

Спрайт – двомірне зображення, що застосовується в комп'ютерній графіці

Action – дія

Android – операційна система

Apple iOS – операційна система

Billboard – спрайт, постійно повернений обличчям до камери

Components – компоненти

C++ – мова програмування

C# – мова програмування

Entity – об’єкт гри, істота

Fighting – жанр відеоігор

Image texture – тестурне зображення

Impostor – спрайт, який замінює тривимірну модель на великій відстані

Java – мова програмування

Platformer – жанр відеоігор

Linux – операційна система

Racing – жанр відеоігор

Shooter – жанр відеоігор

Static Structure diagram – діаграма класів

Swift – мова програмування

Texture atlas – текстурний атлас

Tower Defense – «Баштовий захист» жанр відеоігор

Unreal Engine – двигун для відеоігор

Unity – двигун для відеоігор

Use-Case – варіант використання у мові моделювання UML

.NET – програмна платформа

ВСТУП

Перші комп'ютерні ігри з'явилися лише в 50-60х роках минулого століття, але за цей час ігрова індустрія зробила величезний стрибок у розвитку. Дуже швидко з якихось примітивних програм гри перетворилися на витвір мистецтва.

Ігри аркади набули неабиякої популярності і актуальність серед любителів пограти в комп'ютерні ігри. Саме цей жанр отримав найбільшу кількість шанувальників шанувальників, які захоплюються комп'ютерними іграми. Аркада передбачає швидкість дій гравця, а також вміння оперативно і правильно приймати рішення в ігрових ситуаціях.

Процес гри arcade не змінюється протягом всіх її рівнів і етапів. Відмінна графіка і режими ігор даного жанру дозволяють отримати максимальне задоволення їх гравцям.

Arcade відрізняються наявністю заохочувальної бонусної системи. За кожне значуще досягнення гравець отримує окуляри, які підсилюють його зацікавленість і азарт. Система заохочення є основним стимулом в аркадних іграх [1][1)].

Як дипломний проект, буде розроблена аркадна гра «Who came for Dave». Жанр аркадних ігор є дуже популярним і поширеним, а аудиторія гравців в такі ігри (їх ще називають казуальними) величезна. Причиною популярності є простота в керуванні і ігровому процесі – аркади, не вимагають від гравця серйозних навичок і великої кількості часу для освоєння.

Будь-яка людина, навіть не знайома з іграми, за декілька хвилин може освоїти керування і спокійно отримувати задоволення від ігрового процесу. Велика частина аудиторії аркад – діти, тому багато ігор використовують просту, але приємну графіку, часто стилізовану під мультфільми.

[1)] [1] Все игры жанра «аркада / arcade», вышедшие в 2020 году. URL: https://www.kinonews.ru/games-arcade/. (дата звернення 02.03.2020).

Також аркадні ігри не вимагають від гравця великої кількості часу, або постійного залучення – гравець може включити гру на кілька хвилин, щоб скрасити очікування, або коли потрібно трохи розслабитися, а через пару хвилин гру можна спокійно вимкнути і повернутися до своїх справ.

Загальні характеристики кваліфікаційної роботи:

- повний обсяг сторінок пояснювальної записки – 49;
- кількість рисунків – 22;
- кількість таблиць – 6;
- кількість посилань – 10.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Більш трьох десятиріч ринок відеоігор активно розвивається. Сучасні технології відкривають фантастичні можливості, про які раніше можна було тільки мріяти. Зараз ринок відеоігор захоплює віртуальна реальність, але всі пам'ятають з чого починалася індустрія. І аркади залишаться, як класика, до якої завжди приємно повернутися.

Аркада – це жанр відеоігор, ігровий процес (геймплей), яких заснований на швидкої реакції гравця, мінімальному ступені свободи керованих персонажів і простому управлінні.

До аркадні ігор належать всі проекти жанрів «файтинг» (fighting), частина ігор жанру «гонки» (racing), частина ігор жанру «шутер» (shooter), ігровий процес в цих іграх простий і короткий – одна ігрова сесія займає всього кілька хвилин, однак гравець не сумуватиме ні на секунду. Завдання можуть бути різні, проте аркадні ігри об'єднує простота в керуванні і цілі – зібрати певну або найбільшу кіль очків, знищити всіх ворогів на рівні, або дістатися до виходу [2][1)].

Platformer (з англ. – «платформер») – жанр відеоігор, ігровий процес в якому складається зі стрибків персонажа по різноманітних платформах (звідси і назва) та через перешкоди, збирання предметів, звичайно необхідних для завершення рівня.

Action (з англ. – «дія») – жанр комп'ютерних ігор, в якому робиться наголос на використання фізичних можливостей гравця, в тому числі координації очей і рук і швидкості реакції. Поєднується з безліччю жанрів, наприклад – action-platformer.

[1)] [2] Аркада – что это такое? (путаница с этим жанром видеоигр). URL: https://tangar.info/slovar/arkada. (дата звернення 02.03.2020).

Tower Defense (з англ. – «Баштовий захист») – піджанр комп'ютерних стратегічних ігор. Завдання гравця в іграх подібного жанру – розправитися з наступаючими ворогами, званими в деяких іграх «Кріп» (від англ. Creep, повзуча тварюка), до того, як вони перетнуть карту, за допомогою будівництва веж, що атакують їх, коли ті проходять поблизу. Противники і вежі зазвичай розрізняються за характеристиками і ціною.

Коли вороги переможені, гравець заробляє гроші, які використовуються для покупки або модернізації веж. Вороги часто наступають не постійно, а невеликими групами, які називають «хвилями». Хвилі можуть наступати після закінчення певного часу, або після клацання гравця.

1.2 Характеристика об’єкту розробки

Гра має назву «Who came for Dave», що в перекладі означає «Хто прийшов за Дейвом». Це 2D-гра жанр якої можна визначити, як аркада, action-platformer з элементами tower defense, тобто гра, де необхідно боротися з великою кількістю різноманітних ворогів і захищати певну точку або об'єкт. Змішуючи різні жанри, можна отримати оригінальну гру, яка зможе зацікавити гравця.

Цілями проекту є створення невеликої гри, отримання досвіду і практичних навичок розробки ігор, таких як створення різних ігрових систем, інтерфейсу, зміни мови ігрового тексту, збереження і завантаження даних через зовнішній файл, створення простого штучного інтелекту.

1.3 Аналітичний огляд аналогів

На першому етапі розробки слід виконати аналітичний огляд аналогів програмного продукту для чіткої постановки завдання до диплому. Так, гра «Pac-Man» (рис. 1) – одна із знакових відеоігор всіх часів.

Рисунок 1 – Скріншот гри «Pac-Man»

Мета гри дуже проста – гравець знаходиться в лабіринті, наповненому «їжею» (зображеної у вигляді точок), і йому потрібно з'їсти їх все, щоб пройти на наступний рівень. Завдання ускладнюють чотири примари, які переслідують Пекмена. Якщо Пекмен зустрінеться з одним з привидів, він втрачає життя і повертається на початок, так само як з'їдені їм точки.

Крім простого тікання від привидів, єдиний захист Пекмена – це чотири гранули-енерджайзера, розташовані в кутах лабіринту. Якщо з'їсти одну – привиди починають боятися і відступати в перебігу невеликого часу, а на ранніх рівнях Пекмен може з'їсти кого-небудь з них, щоб отримати бонусні очки.

З'їдене привид не видаляти його повністю, а повертає на початкове положення, щоб заново почати переслідування. Крім поїдання точок і привидів, існує ще одна можливість отримати бонусні очки – фруктики, які з'являються на кожному рівні ближче до середини лабіринту.

Кожен рівень Pac-Man використовує один і той же лабіринт з 240 точками (їжею) і 4 енерджайзер. Хоч лабіринт весь час однаковий, рівні стають все складніше через зміну швидкості самого Пекмена і прискорення привидів.

Після 21 рівня ніяких змін в грі більше не робиться, тому всі інші рівні ідентичні за складністю.

Наступним аналогом розглянемо гру «Galaga» (рис. 2):

Рисунок 2 – Скріншот гри «Galaga»

Класична аркада, перенесена на ПК прямо з ігрових автоматів. Дія «Galaga» відбувається на одному екрані, де гравець повинен знищувати прибульців і набирати очки, переміщаючи свій корабель виключно в горизонтальній площині [3][1)].

«Galaga» – класичний ретро-шутер, який раз по раз ставав хітом ігрових автоматів, і в епоху піксельних корабликів, і пізніше. Проста графіка, легкий музичний фон, нескладні рівні роблять цю гру одну з улюблених як у дорослих, та к і серед дітей.

Гра «Sonic Generations» (рис. 3) одна з найпоширеніших аркадних ігор. Ця гра про знаменитого синього надшвидкісного їжака Соника, в якій гравцям

[1)] [3] Лучшие аркады на ПК. URL: https://cubiq.ru/luchshie-arkady-na-pk/. (дата звернення 02.03.2020).

належить носитися по рівнях, боротися з ворогами, а так само збирати золоті кільця – словом, все як завжди.

Згідно з сюжетом гри, вічний ворог синього їжака, доктор Еггман все-таки завоював світ, і тепер Сонику належить перемогти його. Різні знакові персонажі, такі як лисеня Тейлз, єхидна Наклз, їжачиха Емі і команда Хаотікс також з'являються в грі.

Рисунок 3 – Скріншот гри «Sonic Generations»

Захоплива гра «Rayman Legends» (рис. 4) – головний персонаж Реймі біжить справа наліво до фінішу, збираючи бонуси і роздаючи ворогам на горіхи. Origins виділялася з купки колег по жанру, по-перше, власним пізнаваним стилем, по-друге, неповторною сумасшедшинкой і, по-третє, ретельно продуманим дизайном.

На одному рівні доводиться зменшуватися до крихітних розмірів, на іншому – буквально прогризати собі дорогу крізь тістечка, а на третьому гра ра-

птом зменшить темп і навчить Реймі грати в хованки. При цьому рівні не прогинаються під тягарем ідей – етапи пробігаються майже інтуїтивно, а вірні рішення приходять на льоту.

Процес збудований з ювелірною точністю. Коли великі рівні, орієнтовані на дослідження і збір бонусів, починають набридати, гравцеві пропонують динамічний етап з гонитвою або битву з босом, а в кінці кожного світу відкривається музичний рівень, де всі дії треба здійснювати точно в ритм грає на тлі композиції [4][1)].

Рисунок 4 – Скріншот гри «Rayman Legends»

Баланс складності чітко дотриманий: початок бадьорий і безтурботне, а ближче до кінця доводиться дуже постаратися, щоб хоча б пройти рівень, – не кажучи вже про те, щоб зібрати всі бонуси. Темп відбувається багаторазово зросте.

Всі вищерозглянуті аналоги мають багато спільного: нескладну графіку, простий сюжет гри, легкий музикальний супровід і наявність ускладнення рівнів. Саме ці характеристики будуть виступати у ролі основи для ідеї дипломного проекту.

[1)] [4] Rayman Legends. Лучшие моменты. URL: https://www.igromania.ru/article/23707/Rayman_Legends.html. (дата звернення 10.03.2020).

1.4 Обґрунтування вибору середовища розробки

Розробка ігор для смартфонів, ПК і консолей значно відрізняється. Хоча б тому, що у них різні технічні характеристики, пристрої введення/виводу і способи поширення продукту. Відразу зробити одну гру на кілька платформ не вийде.

Якщо розробка гри планується під персональний комп'ютер, то в такому разі слід враховувати наступні моменти:

- введення з клавіатури і миші – те, до чого користувачі звикли з дитинства;
- висновок картинки на екран монітора. моделей моніторів багато, вони відрізняються частотою зміни кадру, розмірами, дозволом – це потрібно враховувати під час створення інтерфейсу гри;
- великий обсяг оперативної і відеопам'яті; можна дозволити собі деталізовані текстури, плавні анімації, високополігональні об'єкти світу і великі карти;
- величезна різноманітність відеокарт, процесорів і інших комплектуючих, що робить тестування гри трудомістким процесом;
- можливість поширення старим добрим способом (диски) або через онлайн-магазини (найпопулярніший на даний момент – це steam).

1.4.1 Вибір двигуна для створення гри

Unreal Engine – дуже просунутий движок, спільнота якого останнім часом швидко зростає, чому сприяє компанія-розробник Epic Games. За Unreal Engine проводяться мітапи, стрім, а в цьому році пройшла перша конференція, присвячена розробці на Unreal (рис. 5).

Платформи: движок в першу чергу для тих, хто хоче робити проекти з крутий графікою на ПК і консолях. Для мобільних пристроїв теж підходить,

але поки популярних мобільних ігор на Unreal Engine небагато: Fortnite і PUBG.

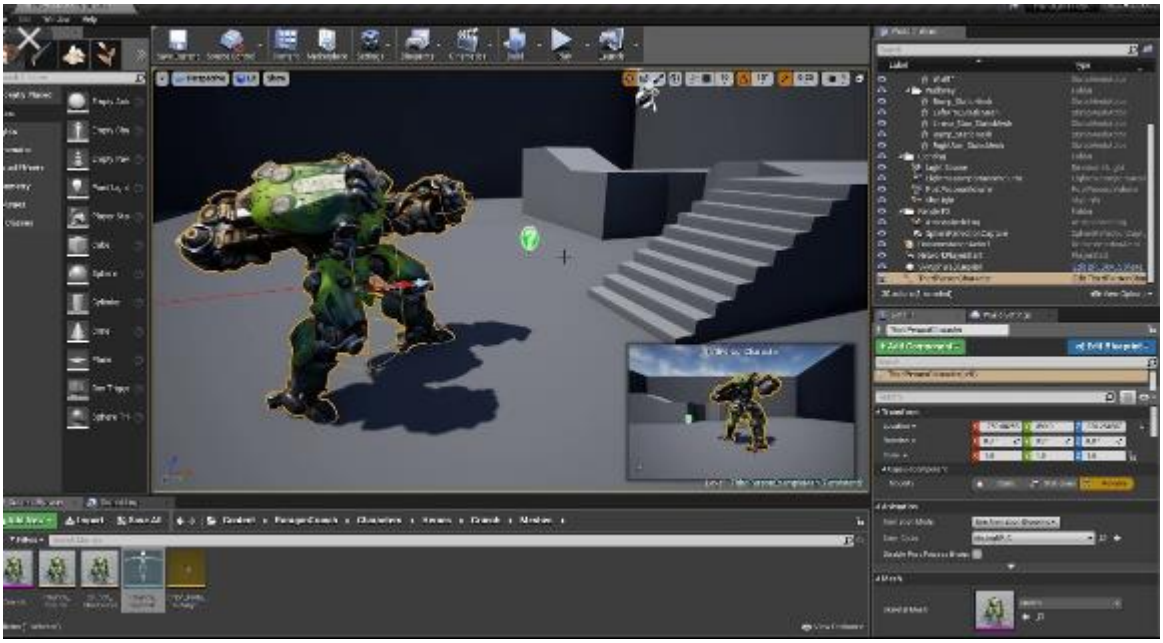

Рисунок 5 – Скріншот робочого вікна Unreal

Мова розробки: C++. Когось це може відлякати, але є рішення – блюпрінти. З їх допомогою теоретично можна розробити гру, не написавши жодного рядка коду. На практиці - це дуже корисно для швидкої розробки прототипів. Також є магазин Accet Unreal Engine Marketplace, де можна скачати готові моделі, звуки і повноцінні проекти.

Злі язики кажуть, що Unreal Engine перевершує Unity по графіку. Насправді це просто різні движки. Хоча частки і пост-ефекти в Unreal Engine за замовчуванням все ж красивіше [5][1)].

Візуалізація в редакторі чудова. Просто очі розбігаються від достатку опцій рендеринга (пов'язаних, наприклад, з освітленням або зі складністю

[1)] [5] Обзор основных движков. URL: https://vc.ru/pixonic/46004-development-engine. (дата звернення 11.03.2020).

шейдеров). Тут ви знайдете масу ультрасучасних шейдеров, які також поставляються разом з движком. В принципі, Unreal пропонує найкращий механізм рендеринга на ринку. Можна створювати дивовижно красиві сцени.

Один з найпопулярніших движків на сьогодні. Unity – багатоплатформовий інструмент для розробки додатків та ігор (рис. 6), що працює на операційних системах. Створені за допомогою Unity додатки працюють під системами Windows, OS X, Android, Apple iOS, Linux, а також на гральних консолях Wii, PlayStation 3 і XBox 360.

Є можливість створювати інтернет-додатки за допомогою спеціального модуля для браузера Unity, а також за допомогою експериментальної реалізації в межах модуля .

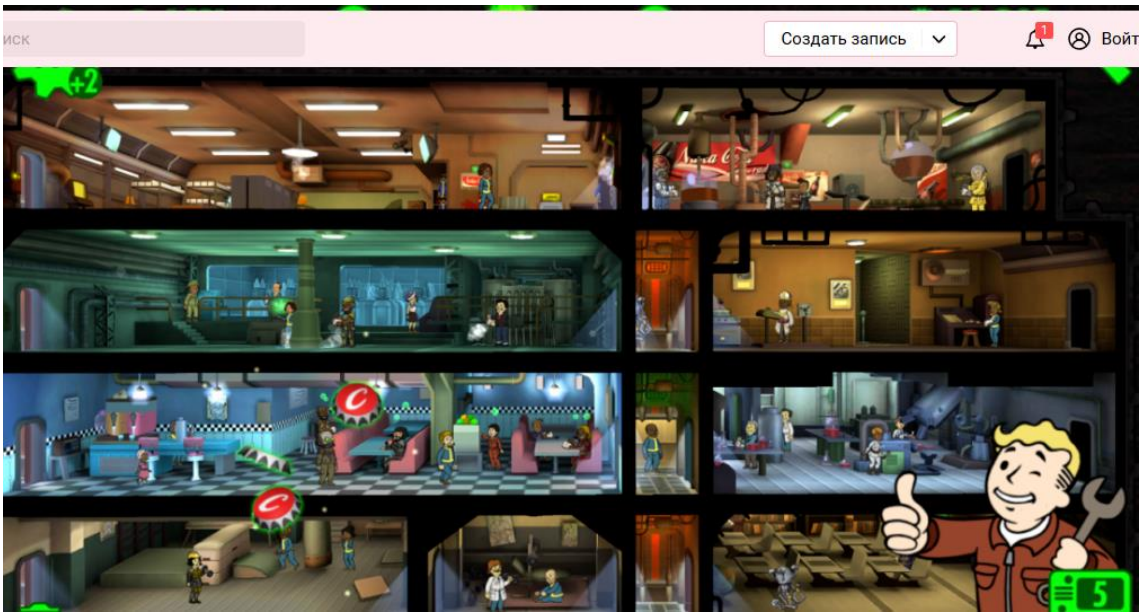

Рисунок 6 – Скріншот робочого вікна Unity

Застосування, створені за допомогою Unity, підтримують DirectX та OpenGL [6][1)].

[1)] [6] Обзор игрового движка Unity3d. URL: https://gamedevmania.ru/engines/unity3d. (дата звернення 10.03.2020).

Технічні характеристики:

– сценарії на C# та JavaScript;
– ігровий двигун, повністю пов’язаний із середовищем розробки, що дозволяє випробовувати гру прямо в редакторі;
– робота з ресурсами можлива через звичайний Drag&Drop;
– система успадкування об’єктів;
– підтримка імпортування великої кількості форматів файлів;
– вбудований редактор ландшафтів;
– вбудована підтримка мережі;
– існує рішення для спільної розробки – Asset Server. Також можна використовувати зручний для користувача спосіб контролю версій. Наприклад, SVN або Source Gear.

Редактор Unity має простий Drag&Drop інтерфейс, який легко налаштовувати. Він складається з різних вікон, завдяки чому можна проводити налагодження гри прямо в редакторі [7][1)].

Проект в Unity поділяється на сцени (рівні) – окремі файли, що містять ігрові світи зі своїм набором об’єктів, сценаріїв, і налаштувань. Об’єкти у сценах, містять набори компонентів, з якими взаємодіють скрипти.

Також у них є назва, може бути тег (мітка) і шар, на якому він повинен відображатися. Так, у будь-якого предмета на сцені обов’язково присутній компонент Transform – він зберігає в собі координати місця розташування, повороту і розмірів по всіх трьох осях.

У версії двигуна 2019 з'явився новий вид ігрових об'єктів – істота (entity). За умовчанням вони не мають ніяких компонентів і відображаються в спеціальному вікні entity debugger.

Істоти є частю нової системи ECS (Entity Component System). Ціль системи поліпшити продуктивність і зручність розробки.

[1)] [7] Движок Unity – особенности, преимущества и недостатки. URL: https://cubiq.ru/dvizhok-unity/. (дата звернення 10.03.2020).

Якщо раніше всі змінні і функції для певного об'єкта зберігалися в одному скрипті MonoBehaviour, то тепер вони поділяються на два окремих класа – компонент, який зберігає змінні конкретної істоти, та системи, яка управляє потрібними істотами.

Таким чином для руху використовується один скрипт, який застосовується до всіх об'єктів, які потрібно рухати. Раніше кожен об'єкт рухався самостійно, що сильно навантажувало систему.

Також Unity підтримує фізику за допомогою фізичних двигунів Unity Physics та Box2D. У редакторі є система успадкування об’єктів, дочірні об’єкти будуть повторювати всі зміни позиції, повороту і масштабу батьківського об’єкта. Скрипти в редакторі прикріплюються до об’єктів у вигляді окремих компонентів [7][1)].

При імпорті текстури в двигуні можна згенерувати карту відображень, проте безпосередньо на модель текстуру прикріпити не можна – буде створено матеріал, з яким буде призначений шейдер і тільки потім матеріал прикріпиться до моделі.

Редактор Unity підтримує написання і редагування шейдерів. Крім того він містить компонент для створення анімації, яку також можна створити попередньо в 3D-редакторі та імпортувати разом з моделлю, а потім розбити на файли.

Скриптова система ігрового рушія зроблена на Mono – вільний відкритий проект з реалізації .NET Framework. Програмісти можуть використовувати UnityScript (власна скриптова мова, подібна до JavaScript та ECMAScript) та C#.

Гра «Who came for Dave» розроблена на ігровому двигуні Unity, версії 2019. Було обрано саме цей двигун, бо він безкоштовний, багатофункціональний і простий у навчанні.

[1)] [7] Движок Unity – особенности, преимущества и недостатки. URL: https://cubiq.ru/dvizhok-unity/. (дата звернення 08.03.2020).

1.4.2 Обґрунтування вибору мови програмування

Для створення сучасних комп'ютерних ігор застосовуються різні мови програмування. Їх вибір залежить від обраного двигуна і платформи, на якій буде працювати гра. Так, для розробки ігор на платформі iOS та MacOS підійде мова Swift, для розробки браузерних ігор – Java Script, для ігор на ПК, як правило, використовують C++ і C#. В якості мови програмування в Unity 2019 присутні такі мови: C# та JavaScript.

У грі «Who came for Dave» скрипти будуть написані на мові C#, бо ця мова зараз дуже актуальна та позволяє раціонально створювати різноманітні інтернет-додатки. Також C# інтегрувала в собі переваги мови Java і C++, що й обумовлює популярність даної мови серед розробників. При цьому в об’єднаній мові виключені деякі суперечливі директиви, макроси, скасовані глобальні змінні.

C# – об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft) [8][1)].

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML.

Перейнявши багато що від своїх попередників – мов C++, C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

[1)] [8] Язык программирования C#: краткий обзор. URL: https://techrocks.ru/2019/02/16/c-sharp-programming-language-overview/. (дата звернення 10.03.2020).

C# є дуже близьким до мови програмування Java. Мова Java була створена компанією Sun Microsystems, коли глобальний розвиток інтернету поставив задачу розподілених обчислень. Взявши за основу популярну мову C++, Java виключила з неї потенційно небезпечні речі (типу вказівників без контролю виходу за межі).

Для розосереджених обчислень була створена концепція віртуальної машини та машинно-незалежного байт-коду, свого роду посередника між вихідним текстом програм і апаратними інструкціями комп'ютера чи іншого інтелектуального пристрою.

Java набула чималої популярності і була ліцензована також компанією Microsoft. Але з плином часу Sun почала винуватити Microsoft, що та при створенні свого клону Java робить її сумісною виключно з платформою Windows, чим суперечить самій концепції машинно-незалежного середовища виконання і порушує ліцензійну угоду. Microsoft відмовилася піти назустріч вимогам Sun, і тому з'ясування стосунків набуло статусу судового процесу. Суд визнав позицію Sun справедливою, і зобов'язав Microsoft відмовитися від позаліцензійного використання Java [9][1)].

У цій ситуації в Microsoft вирішили, користуючись своєю вагою на ринку, створити свій власний аналог Java, мови, в якій корпорація стане повновладним господарем. Ця новостворена мова отримала назву C#. Вона успадкувала від Java концепції віртуальної машини (середовище .NET), байт-коду і більшої безпеки вихідного коду програм, плюс врахувала досвід використання програм на Java.

Нововведенням C# стала можливість легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проектів.

[1)] [9] Введение в Java. URL: https://metanit.com/java/tutorial/1.1.php. (дата звернення 10.03.2020).

Якщо програми на різних мовах виконуються на платформі .NET, .NET бере на себе клопіт щодо сумісності програм (тобто типів даних, за кінцевим рахунком).

Станом на сьогодні C# визначено флагманською мовою корпорації Microsoft, бо вона найповніше використовує нові можливості .NET. Решта мов програмування, хоч і підтримуються, але визнані такими, що мають спадкові прогалини щодо використання .NET.

C# розроблялась як мова програмування прикладного рівня для CLR і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR [8][1)].

[1)] [8] Язык программирования C#: краткий обзор. URL: https://techrocks.ru/2019/02/16/c-sharp-programming-language-overview/. (дата звернення 10.03.2020).

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

2.1 Призначення та вимоги до системи

Призначення системи: дана розробка є дипломним проектом. Програмний комплекс призначений для створення комп'ютерної гри на двигуні Unity.

Розробка виконується з метою:

- розробити гру у жанрі аркада;
- отримати досвід розробки;
- полегшити подальшу розробку проектів з використанням вже готового програмного забезпечення та ігрових систем.

Вимоги до функціональних характеристик.

Програмний продукт повинен володіти наступними функціональними характеристиками:

- сюжет;
- переміщення головного героя;
- ігрові меню;
- переміщення між різними ігровими меню;
- налаштування гри з можливістю змінити мову ігрового тексту, змінити гучність звуку і музики, перемикати режим вертикальної синхронізації;
- система завантаження і збереження ігрових даних у зовнішній файл.

У процесі роботи програми вхідною інформацією для програми повинні бути:

- натискання клавіш на клавіатурі;
- натискання кнопок миші.

Надійне (стійке) функціонування програмного комплексу має бути забезпечене шляхом:

- контролю коректності та повноти вхідних даних – усі дані, що вводяться користувачем, перевіряються на формальну коректність;

– відновлення після відмови – в разі виникнення програмного збою система повинна відновлювати роботу з останнього зафіксованого стабільного стану.

Контроль вхідної і вихідної інформації – програма повинна контролювати вибір користувачем пункту меню «Вихід».

Вимоги до технічного забезпечення – програмний комплекс розрахований на функціонування при такому наборі технічних засобів – мінімальна апаратна конфігурація комп’ютеру:

– процесор з тактовою частотою 1.2 ГГц;
– оперативна пам’ять об’ємом не менш, ніж 512 Мб;
– відеокарта та монітор з роздільною здатністю не менше 1024x768;
– клавіатура;
– вільний дисковий простір не менш, ніж 100 Мб;
– додатково: маніпулятор типу «миша».

Вимоги до середовищ програмування.

Розробка програми повинна вестися мовою програмування C# у середовищі розробки Visual Studio, та на двигуні Unity. Вибір інших середовищ розробки недоцільний.

Вимоги до програмних засобів, використовуваних програмою.

Для роботи програми необхідна операційна система WINDOWS 7 і більш пізня.

2.2 Проектування графічних елементів

У комп'ютерній графіці реального часу, текстурний атлас (англ. Texture atlas) – це зображення, що містить набір (або «атлас») підзображень, кожне з яких є текстурою для деякого 2D або 3D об'єкта. Підекстури відображаються на об'єкт, використовуючи UV-перетворення, при цьому координати в атласі задають, яку частину зображення потрібно використовувати.

У додатках нерідко використовується безліч маленьких текстур, причому перемикання з однієї текстури на іншу є відносно повільним процесом.

Тому в подібних ситуаціях буває доцільно застосування одного веикого зображення замість безлічі маленьких. Наприклад, в іграх з tile-графікою можна отримати хороший виграш в швидкості виведення картинки.

Дипломний проект проектується як 2D-гра. Для її створення використовувались такі елементи, як:

1. «Текстура зображення» (Image texture) – являє собою набір метрик, розрахованих при обробці зображення, призначених для кількісного визначення текстури. Текстура зображення дає нам інформацію про просторове розташування кольору або інтенсивності в зображенні або вибраної області зображення. Її розмір повинен співпадати зі ступенем 2-ки, для продуктивності;
2. Спрайт – двомірне зображення, що застосовується в комп'ютерній графіці. Частіше за все – растрове зображення, що вільно переміщується по екрану. Спостереження за спрайтом під невідповідним кутом приводить до розкриття ілюзії. Тобто легше за все сприймати спрайт, як проекцію, що переміщуються в просторі так, що різниця не помітна;
3. Billboard – спрайт, постійно повернений обличчям до камери (по аналогії з рекламними щитами на автодорогах, які повернені під найбільш вигідним кутом);
4. Impostor – спрайт, який замінює тривимірну модель на великій відстані;

Слово спрайт походить від англ. Sprite – фея, ельф. Чи означає це поняття об'єкт комп'ютерної графіки. Але в більш вузькому сенсі спрайт є кілька картинок, які поєднані в одну.

Атласи можуть містити як під-текстури однакових розмірів, так і під-текстури відрізняються розмірів (зазвичай, є ступенем двійки).

Для складання атласів використовуються як програми-генератори, так і ручне складання. При використанні MIP-текстурування необхідно передбачити, що під-текстури повинні бути розставлені таким чином, щоб не виникало ситуації, коли одна з них «залазить» на іншу [9][1)].

У спрайтовій анімації часто використовуються спрайтові атласи – зображення, на яких знаходиться кожен кадр для анімації. Це дозволяє ефективніше працювати і змінювати анімацію, а також підвищує продуктивність – комп'ютеру не доводиться шукати кожен кадр, так як вони згруповані разом. На рисунку 7 зображений атлас анімації головного героя. Його тіло поділене на 2 частини, які анімуються незалежно одна від одної. Це спрощує створення анімацій.

Рисунок 7 – Спрайтовий атлас головного герою

[1)] [9] Что такое графический спрайт (Sprite)?. URL: https://www.econ-dude.pw/2017/08/chto-takoe-graficheskij-sprajt-sprite.html#gsc.tab=0. (дата звернення 20.03.2020).

2.3 Моделювання UML-засобами

2.3.1 Діаграма варіантів використання

Основна мета створення будь-якої програмної системи це створення програмного продукту, який допомагає користувачу виконувати свої повсякденні завдання. Для створення таких програм насамперед визначаються вимоги, яким повинна задовольняти система. Проте якщо дати користувачам написати ці вимоги на папері, то часто можна одержати список функцій, по якому важко судити чи буде майбутня система виконувати своє призначення і чи зможе вона полегшити користувачу виконання його роботи взагалі.

Для того, щоб точніше зрозуміти як повинна працювати система, все частіше використовується опис функціональності системи через варіанти використання (Use-Case або прецеденти). Варіанти використання це – опис послідовності дій, які може здійснювати система у відповідь на зовнішні дії користувачів або інших програмних систем. Варіанти використання відображають функціональність системи.

Діаграми варіантів використання описують функціональне призначення системи або те, що система повинна робити. Мета розробки діаграм наступна:

- визначити загальні межі і предметну область;
- сформулювати загальні вимоги до функціональної поведінки проектованої системи;
- розробити початкову концептуальну модель системи її подальшої деталізації у формі логічних і фізичних моделей;
- підготувати початкову документацію для взаємодії розробників системи з замовниками і користувачами [10][1)].

Діаграма варіантів використання – це граф спеціального вигляду, який є графічною нотацією для представлення конкретних варіантів використання,

---

[1)] [10] Розробка uml діаграми варіантів використання. URL: https://studfile.net/preview/5200239/page:6/. (дата звернення 20.03.2020).

акторів, можливо деяких інтерфейсів, і відносин між цими елементами. При цьому окремі компоненти діаграми можуть бути поміщені в прямокутник, який позначає проектовану систему в цілому. Слід зазначити, що відносинами даного графа можуть бути тільки деякі фіксовані типи взаємозв'язків між акторами і варіантами використання, які в сукупності описують сервіси або функціональні вимоги до модельованої системи.

Для об’єкту розробки було створено наступну діаграму (рис. 8):

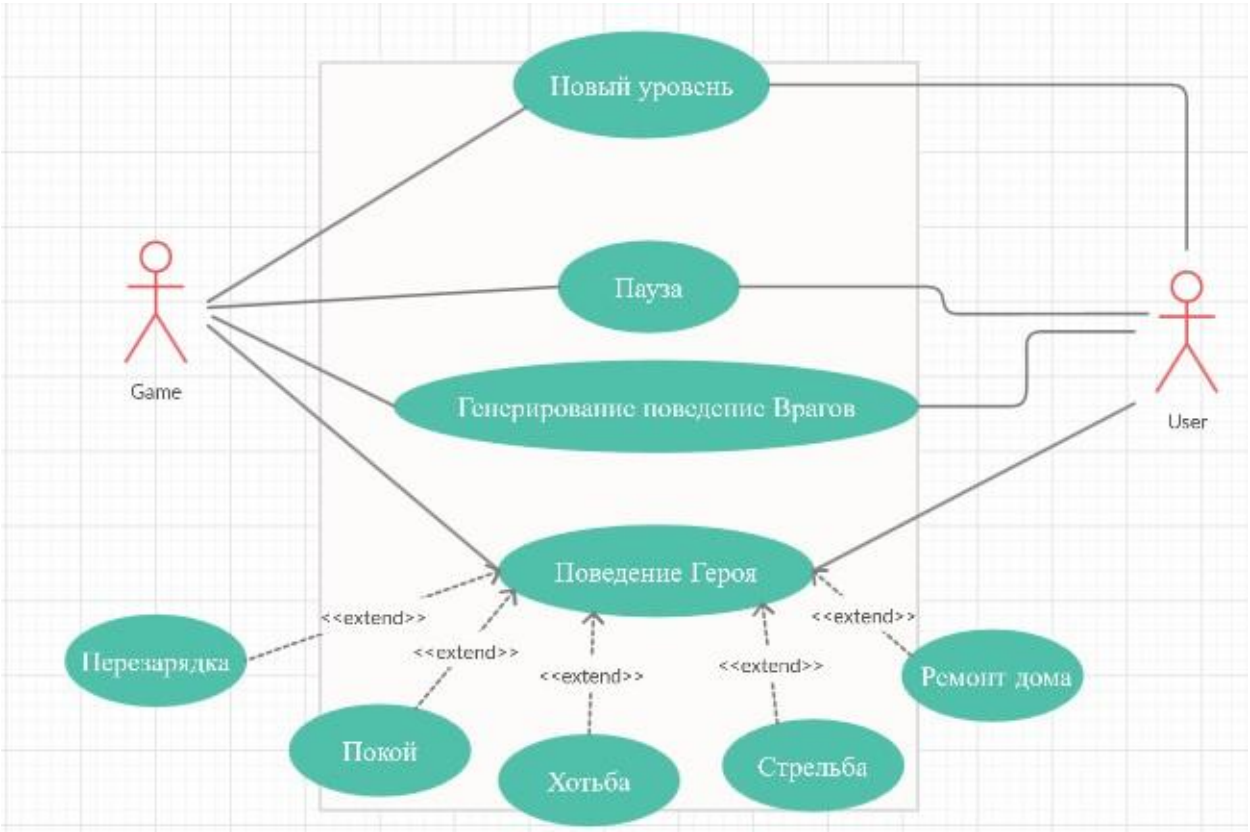

Рисунок 8 – Діаграма варіантів використання

2.3.2 Діаграма класів

Діаграма класів (англ. Static Structure diagram) – діаграма, що демонструє класи системи, їх атрибути, методи і взаємозв'язку між ними.

Являється однією з основних UML-діаграм.

Це набір статичних, декларативних елементів моделі. Діаграми класів можуть застосовуватися й при прямому проектуванні, тобто в процесі розробки нової системи, та при зворотному проектуванні – описі існуючих та використовуваних систем. Інформація з діаграми класів безпосередньо відображається в вихідному коді програми – в більшості існуючих інструментів UML-моделювання можлива кодогенерація для певної мови програмування (зазвичай Java або C++). Таким чином, діаграма класів – кінцевий результат проектування та відправна точка процесу розробки [11][1)].

Клас – це основний будівельний блок ПС. Це поняття є і в ГО мовах програмування, тобто між класами UML і програмними класами є відповідність, що є основою для автоматичної генерації програмних кодів або для виконання реінжинірингу.

Кожен клас має назву, атрибути і операції. Клас на діаграмі показується у вигляді прямокутника, розділеного на 3 області. У верхній міститься назва класу, в середній – опис атрибутів (властивостей), в нижній – назви операцій – послуг, що надаються об'єктами цього класу.

Діаграма класів займає центральне місце в проектуванні об'єктно-орієнтованої системи. Нотація класів використовується на різних етапах проектування та будується з різним ступенем деталізації. Мова UML застосовується не тільки для проектування, але і з метою документування, а також створення ескізів проекту.

У будь-якому об'єктно-орієнтованому процесі проектування діаграма класів є результатом, тому що вона є моделлю, найбільш близькою до реалізації, тобто коду. Існують інструменти, що здатні перетворити діаграму класів в код – такий процес називається кодогенерацією та підтримується безліччю IDE та засобів проектування.

UML-діаграма основних класів проекту представлена на рисунку 9.

[1)] [10] Отношения классов — от UML к коду. URL: https://habr.com/ru/post/150041/. (дата звернення 23.03.2020).

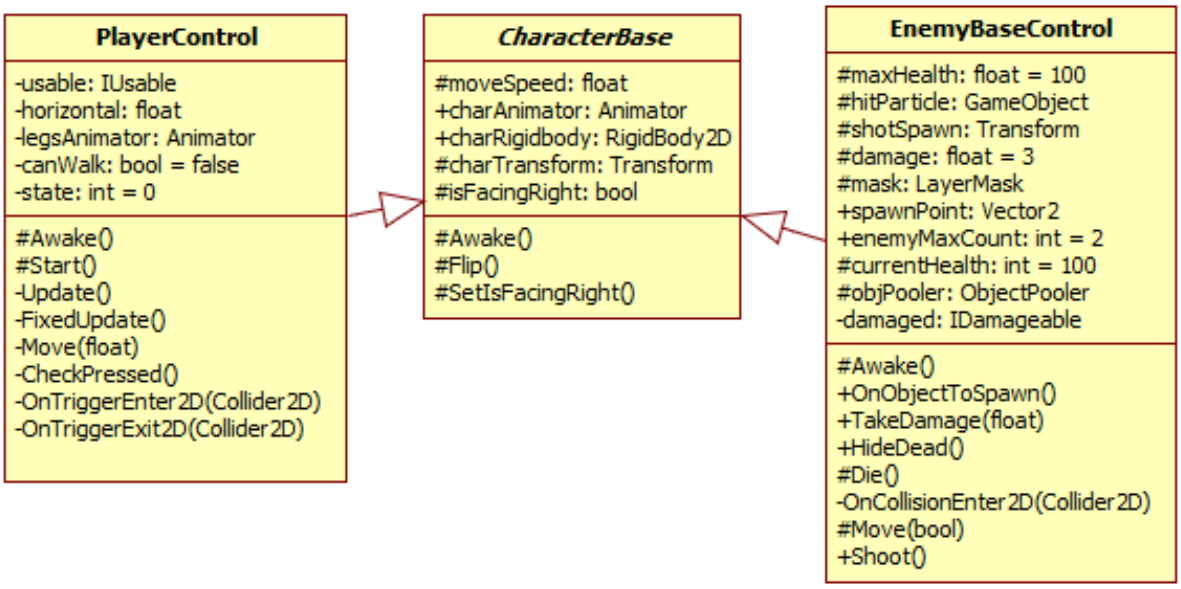

Рисунок 9 – UML-діаграма класів проекту

3 РЕАЛІЗАЦІЯ ТЕХНІЧНОГО ПРОЕКТУ

3.1 Розробка скриптів

Скрипт – це програма або програмний файл сценарій, які автоматизують деяку задачу, яку користувач робив би вручну, використовуючи інтерфейс програми. Скрипти пишуться на скриптових мовах, які відрізняються за своїм синтаксисом, сферами застосування та можливостями. До скриптових мов належать: AngelScript, Perl, Python, PHP, JavaScript, JScript та інші.

Сфера застосування скриптів величезна. Наприклад:

- можливість звертатися до баз даних;
- seo-скрипти, що допомагають просувати сайти, використовуючи спеціальні програми автоматизації браузера;
- лічильники відвідування, які допомагають спостерігати статистику відвідувань;
- можливість створювати записи в гостьових книгах;
- можливість залишати коментарі до вподобаним статей;
- на скриптах засновані всі cms і форуми;
- динамічне відображення веб-сайту;
- організація зміни частини сайту без перевантаження всієї сторінки тощо.

Поведінка ігрових об’єктів в Unity контролюється за допомогою компонентів (Components), які приєднуються до них. Незважаючи на те, що вбудовані компоненти Unity можуть бути дуже різнобічними, їх недостатньо, щоб реалізувати усі власні особливості геймплея.

Unity дозволяє створювати свої компоненти, використовуючи скрипти. Вони дозволяють активувати ігрові події, змінювати параметри компонентів, і відповідати на дії користувача яким завгодно способом.

Unity підтримує дві мови програмування:

- C#, стандартна в галузі мова, подібна до Java або C++;

– UnityScript, мова, розроблена спеціально для використання в Unity за зразком JavaScript.

Нижче наведено скрипт “CharacterBase”, написаний на мові C#, який є базовим скріптом для головного герою та ворогів:

```
public abstract class CharacterBase : MonoBehaviour
{
     protected float moveSpeed;
     public Animator charAnimator;
     public Rigidbody2D charRigidBody;
     protected bool isFacingRight;

     protected virtual void Awake ()
     {
          charAnimator = GetComponent<Animator> ();
          charRigidBody = GetComponent<Rigidbody2D> ();
               SetIsFacingRight();
     }

     protected virtual void Flip()
     {
               isFacingRight = !isFacingRight;
               charTransform.Rotate(0,180,0);
     }
}
```

В цьому скрипті зберігаються основні змінні для персонажів – швидкість пересування, посилання на компоненти для взаємодії з анімацією, фізикою і позицією персонажу. А також змінна, яка визначає, чи дивиться персонаж вправо. Крім цього, скрипт може розгортати персонажа вліво або вправо.

Наступна частина скрипту відповідає за пересування героя:

```
     void Update ()
     {
          horizontal = CanWalk ? Input.GetAxisRaw ("Horizontal")
: 0f;
               if (horizontal != 0)
                    if (State < 2)  State = 1;
                else if (State < 2) State = 0;
          Move(horizontal);
     }

     void Move(float move)
```

```
     {
               float absMove = Mathf.Abs(move);
               charAnimator.SetBool("isWalk", move!=0);
                if (LegsAnimator!=null)
                    LegsAnimator.SetBool("isWalk", move != 0);
charRigidBody.velocity = new Vector2 (move * moveSpeed *
Time.fixedDeltaTime, 0);
     }
```

Наступна частина скрипту відповідає за взаємодію з ігровими об'єктами за допомогою інтерфейсу IUsable:

```
     void CheckPressed()
     {
          if (Input.GetButtonDown ("Use"))
               if (usable != null) usable.Use ();
     }

     void OnTriggerEnter2D (Collider2D other)
     {
          IUsable temp = other.GetComponent<IUsable>();
          if(temp!=null) usable =  temp;
     }
     void OnTriggerExit2D (Collider2D other)
     {
          if(temp!=null) usable = null;
     }
```

При натисканні необхідних клавіш, персонаж рухається або взаємодіє з об'єктами, також перемикається його стан. Всього станів персонажа п'ять: спокій, ходьба, стрільба, перезарядка і ремонт будинку. При зміні стану змінюється анімація головного героя.

Також налаштовані переходи між станами – коли герой знаходиться в спокійному стані, він може робити будь-яка дію. Але, наприклад, при ремонті будинку він не може стріляти.

Скрипт під назвою “LocalizationManager” виконує важливу функцію – змінює мову ігрового тексту. Наступна частина скрипту завантажує необхідний файл з текстом, відповідно до обраної мови:

```
     public void LoadLocalizedText( int ind )
     {
          LocalizationText = new Dictionary<string, string> ();
          string filePath = Application.streamingAssetsPath+text-
FileName[ind];
          if (File.Exists (filePath))
          {
               string dataJson = File.ReadAllText (filePath);
               LocalizationData loadedData =
                    JsonUtility.FromJson<LocalizationData>      (
dataJson );
               for(int i=0; i<loadedData.items.Length; i++)
               {
                    LocalizationText.Add                  (loaded-
Data.items[i].key,
loadedData.items[i].value);
               }
               foreach(LocalizedText el in locTexts)
                    el.ChangeLanguage();
          }
     }
```

Гравець може змінювати мову прямо під час гри, через меню паузи. Також в гру можна додавати нові файли для різних мов.

Результат роботи скрипта «LocalizationManager» зображено на рисунку 10.

Рисунок 10 – Результат роботи скрипта «LocalizationManager»

Скрипт «CameraFollow» відповідає за те, щоб ігрова камера невідступно слідувала за головним героем:

```
public class CameraFollow : MonoBehaviour
{
     [SerializeField] private Transform target;
     [SerializeField] private Vector2 minDot, maxDot;

     void Update ()
     {
               Vector2 CameraFollowPosition = target.position;
               float  newX  =  Mathf.Clamp(CameraFollowPosition.x,
minDot.x,                                maxDot.x);
                float  newY  =  Mathf.Clamp(CameraFollowPosition.y,
minDot.y,                                maxDot.y);
               transform.position = new Vector3(newX, newY, -10f);
     }
}
```

Цей скрипт задає камері ціль, за якою вона буде слідувати. Ціллю може бути гравець або будь-який інший ігровий об'єкт. Також задаються максимальні і мінімальні координати, в межах яких може рухатися камера.

Кожен кадр камері передаються координати цілі, і якщо ці координати знаходяться між мінімальним і максимальним межами, камера слідує за ціллю, інакше, залишається в заданих межах, поки ігровий об'єкт не повернеться в них.

Створення та видалення ігрових об'єктів – досить ресурсовитратна операція для будь-якого проекту. Такий етап слід одразу продумати і спроектувати. Якщо в грі постійно створюються і видаляються ігрові об'єкти (наприклад кулі), це може позначитися на продуктивності.

Щоб уникнути цього, використовуються пули (англ. Pool) – особливі списки об'єктів. Коли в грі потрібен якийсь об'єкт, він з'являється в потрібних координатах і виконує задану функцію, а потім деактивується і може використовуватися заново.

Приклад пула ігрових об'єктів наведено на рисунку 11:

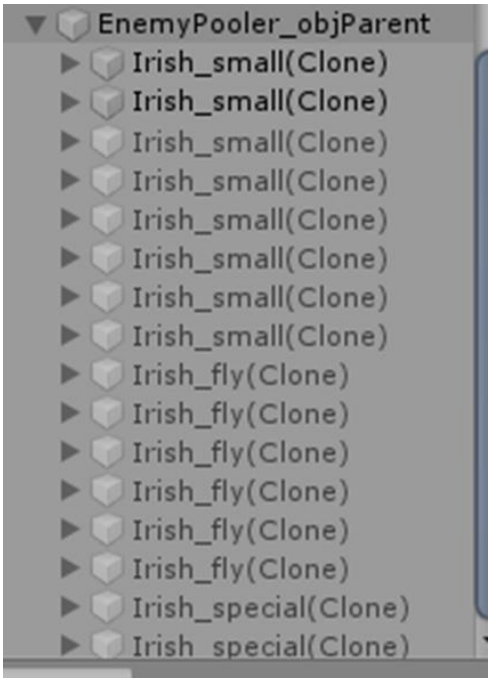

Рисунок 11 – Пул ворогів. Чорним позначені активні об'єкти, сірим – неактивні

3.2 Створення інтерфейсу гри

Сучасні ігри та програми часто потребують підтримки широкого спектра різних дозволів екрану і, особливо в даній ситуації, потребують інтерфейси цих ігор.

Система створення інтерфейсів в Unity забезпечена рядом різних інструментів для реалізації цих можливостей, які також можна комбінувати між собою масою різних способів, що є дуже зручним для розробника-початківця.

Інтерфейс користувача (UI – англ. User interface) – інтерфейс, що забезпечує передачу інформації між користувачем-людиною і програмно-апаратними компонентами комп’ютерної системи.

Під сукупністю засобів і методів інтерфейсу користувача маються на увазі:

– Засоби виведення інформації з пристрою до користувача – весь доступний діапазон впливів на організм людини (зорових, слухових, тактильних, нюхових тощо): екрани (дисплеї, проектори) і лампочки, динаміки, зумери і сирени, вібромотор тощо.

– Засоби введення інформації / команд користувачем в пристрій – безліч всіляких пристроїв для контролю стану людини: кнопки, перемикачі, потенціометри, датчики положення і руху, сервоприводи, жести особою і руками, навіть знімання мозкової активності користувача.

За наявністю тих чи інших засобів введення, інтерфейси поділяються на типи: жестовий, голосовий, брейн та інші. Можливі змішані варіанти. Ці засоби повинні бути необхідними і достатніми, бути зручними і практичними, розташованими і скомпонованими розумно і зрозуміло, відповідати фізіології людини, не повинні призводити до негативних наслідків для організму користувача (все це входить в поняття ергономіки).

– Методи – набір правил, закладених розробником пристрою, згідно з якими сукупність дій користувача повинна привести до необхідної реакції пристрою і виконання необхідного завдання – так званий логічний інтерфейс. Правила ці повинні бути досить ясні для розуміння, природні і легкі для запам’ятовування (все це входить в поняття юзабіліті).

Збільшення в пристрої (при рівній функціональності) засобів вводу-виводу дає спрощення побудови методів керування і спрощення правил користування, але зате призводить до складності сприйняття інформації користувачем – інтерфейс стає перевантаженим. І навпаки – зменшення засобів відображення і контролю призводить до ускладнення правил керування – кожен елемент несе на собі занадто багато функцій. Тому проектувальники інтерфейсів намагаються прийняти компромісне рішення між цими двома крайностями у кожному окремому випадку.

Ігровий процес в «Who came for Dave» супроводжує HUD (англ. Heads-Up Display, інтерфейс, який показується поверх ігрового екрану).

Цей інтерфейс показує основну інформацію про стан гри для користувача. Вгорі екрану знаходиться іконка використовуваної зброї, індикатор патронів та індикатор міцності будинку, як показано на рисунку 12.

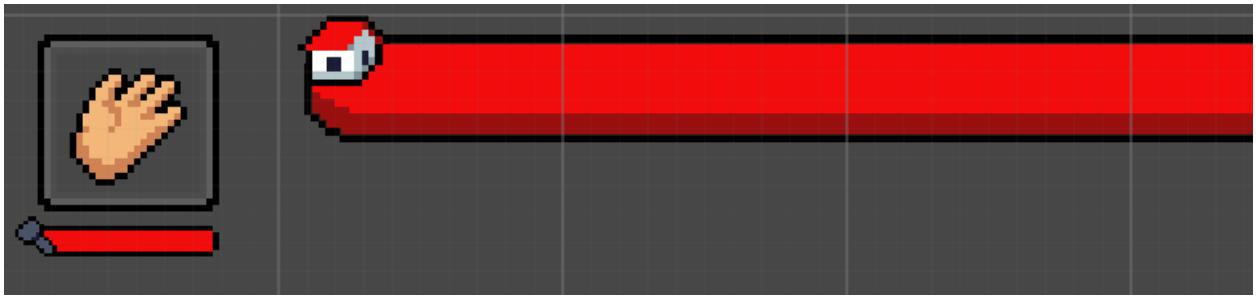

Рисунок 12 – Ігровий HUD

Інтерфейс виконаний в мінімалістичному стилі, щоб гравець міг завжди швидко оцінити ситуацію і прийняти рішення. Також кожен індикатор позначений відповідною іконкою, що полегшує розпізнавання елементів. Під час діалогів на екрані з'являються чорні смуги, що імітують кадр з кіно. Поряд з кожною реплікою знаходиться іконка говорить, що допомагає зрозуміти, хто в даний момент говорить. Інтерфейс під час діалогів зображено на рисунку 13.

Рисунок 13 – Інтерфейс під час діалогів

Гравець в будь-який момент може припинити гру за допомогою меню паузи. Також в цьому меню він може перезапустити рівень заново, вийти з гри, або в головне меню. Головне меню та меню паузи показано на рисунку 14 та рисунку 15.

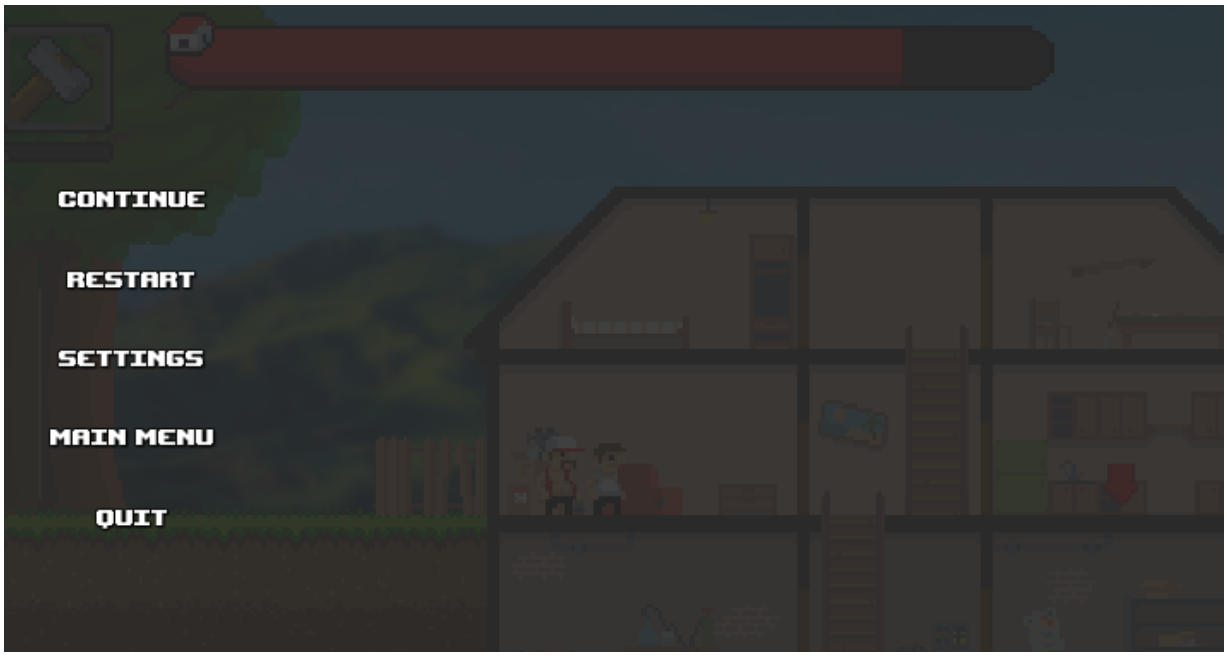

Рисунок 14 – Меню паузи

Рисунок 15 – Головне меню

Також з головного меню, або з меню паузи можна перейти в меню налаштувань. Тут гравець може змінити мову ігрового тексту, гучність звуку і музики, включити або відключити вертикальну синхронізацію (функція, що прибирає розриви при відображенні кадру). Меню налаштувань зображено на рисунку 16.

Рисунок 16 – Меню налаштувань

При натисканні на кнопки в будь-якому з меню, вони активують задані в скрипті функції. Наприклад такі функції виконують кнопки в головному меню:

```
public class MainMenu:MonoBehaviour
{
    [SerializeField]
    private GameObject settings;
    public void ContinueGame()
    {
         SceneChanger.Instance.GoToSceneWithIndex(1);
    }
    public void NewGame()
    {
              GameData.Instance.SetDefaultProgress();
              SceneChanger.Instance.GoToSceneWithIndex(2);
    }
    public void QuitGame()
    {              Application.Quit ();     }}
```

Функція SceneChanger.Instance.GoToSceneWithIndex() активує перехід на іншу сцену. Список доступних для переходу сцен описується в Build Settings (рис. 17).

Рисунок 17 – Меню Build Settings

У грі дуже багато складних об'єктів (ті, які складаються с інших об'єктів). Наприклад, на рисунку 18 показано усі ігрові об'єкти, з яких складається дім головного герою.

Рисунок 18 – О б'єкти, з яких складається дім головного герою

3.3 Анімація

Комп'ютерна анімація – мистецтво створення рухомих зображень, за допомогою комп'ютерів. Є підрозділом комп'ютерної графіки та анімації. На відміну від більш загального поняття «графіка CGI», що відноситься як до нерухомих, так і до рухомих зображень, комп'ютерна анімація має на увазі тільки рухомі. На сьогодні отримала широке застосування як в області розваг, так і у виробничій, науковій та діловій сферах. Будучи похідною від комп'ютерної графіки, анімація успадковує ті ж способи створення зображень:

– векторна графіка;

– растрова графіка;

– тривимірна графіка (3D).

Розстановка ключових кадрів проводиться аніматором. Проміжні ж кадри генерує спеціальна програма. Цей спосіб найбільш близький до традиційної мальованої анімації, тільки роль фазовщика бере на себе комп'ютер, а не людина (рис. 19).

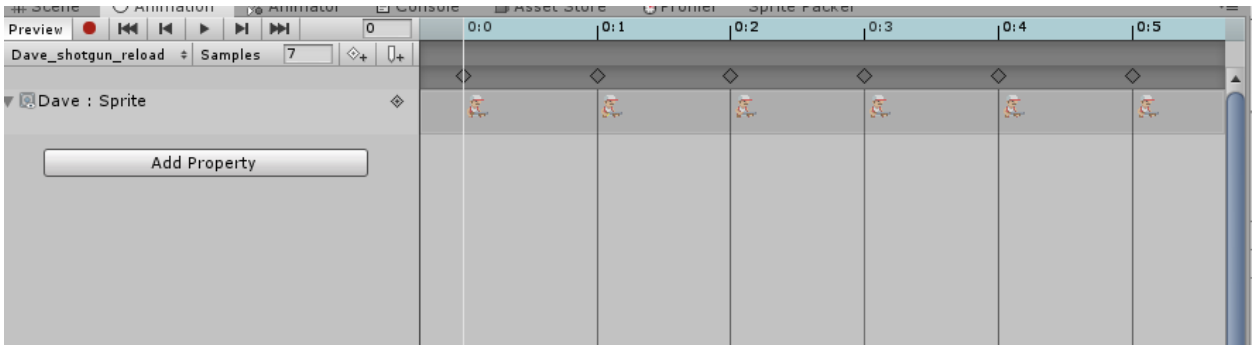

Рисунок 19 – Вікно анімації

Система анімації в Unity дозволяє створювати чудово анімованих персонажів. Вона підтримує блендінг, мікшування, додавання анімацій, синхронізацію циклу ходьби, анімаційні шари, контроль всіх аспектів програвання (час, швидкість, ваги блендінга), скіннінг мешів з 1, 2 або 4 кістками на

вершину, а також засновані на фізиці rag-dolls (ганчіркові ляльки) і процедурну анімацію.

3.4 Опис роботи

При вході в гру, гравець бачить головне меню. Якщо він ще не починав грати, кнопка «Продовжити» буде недоступна. Гравець може змінити налаштування, або почати нову гру. Після цього завантажується туторіал, який навчає основним ігровим механікам – переміщення по поверху, переміщення між поверхами і лагодження будинку. Туторіал показано на рисунку 20.

Рисунок 20 – Туторіал

Після того, як гравець полагодить будинок, починається основна історія. Гравцеві належить захищати будинок головного героя від наступаючих хвиль ворогів (рис. 21). Робить він це за допомогою різної вогнепальної зброї. Кожна зброя має деяку кількість патронів і періодично герой перезаряджає її.

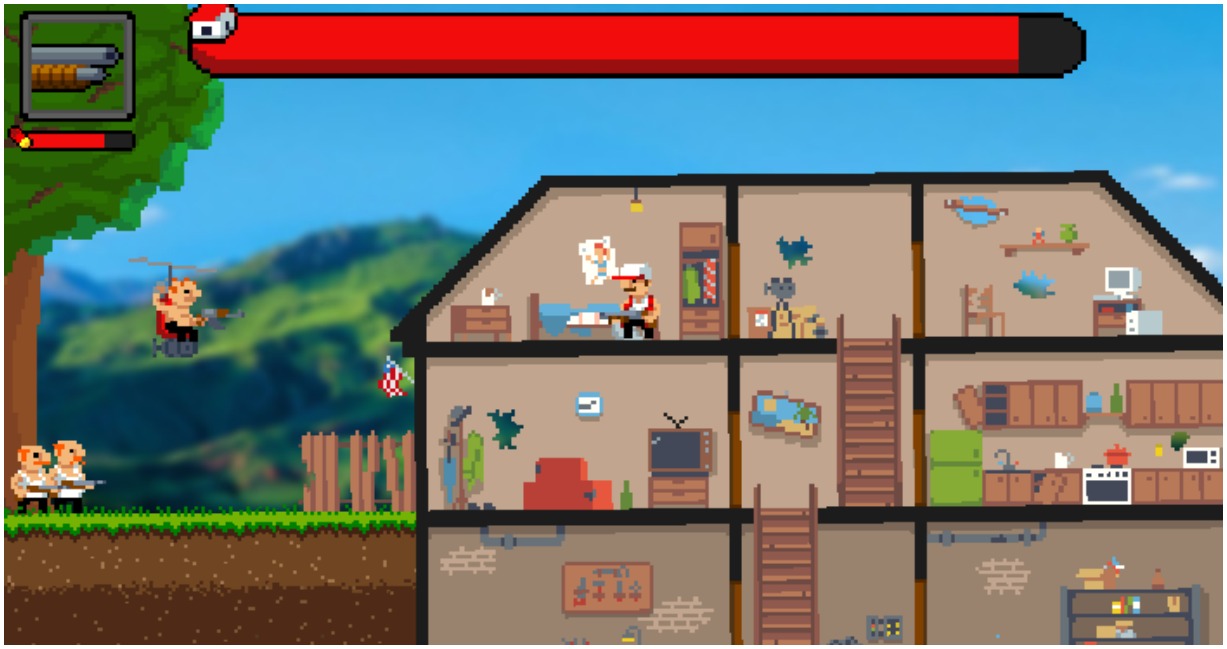

Рисунок 21 – Вороги атакують дім

Щоб пройти хвилю, гравцеві необхідно вбити певну кількість ворогів. Після цього активується таймер до наступної хвилі. Тепер гравець не може стріляти, але може ремонтувати будинок. Після закінчення таймера починається нова хвиля. Таймер до наступної хвилі показано на рисунку 22.

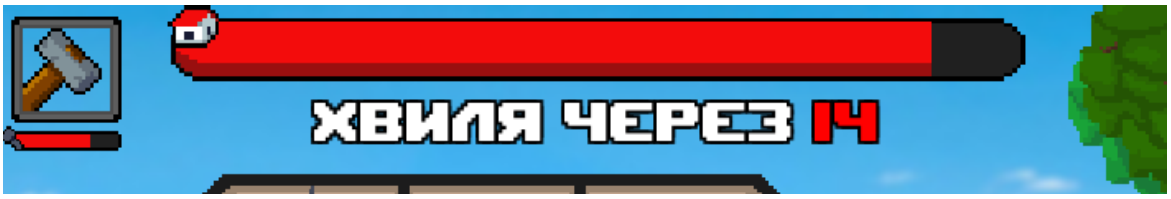

Рисунок 22 – Таймер до наступної хвилі ворогів

Після проходження усіх хвиль, гравця чекає кінцівка і фінальні титри. Гра зберігає дані про проходження в файл, тому гравець може вийти і продовжити проходження в будь-який момент. Якщо файл збереження буде видалений або пошкоджений, гра створить новий і проходження доведеться починати заново.

3.4 Тестування програми

Мета тестування: виявлення функціональних помилок, невідповідностей технічному завданню і очікуванням Замовника шляхом реалізації стандартних, а також нетривіальних тестових сценаріїв.

Класифікація функцій, які будуть протестовані у першому тесті:

– переміщення вправо та вправо;

– стрільба;

– перезаряджання зброї.

Календарний план робіт 1-го тесту представлено у таблиці 1.

Таблиця 1 – Календарний план робіт 1-го тесту

Номер тесту	Дата початку	Дата кінця	Часи			
			12:00	12:10	12:25	13:00
1	15.05.2019	15.05.2019				
2						
3						

Функції, які будуть протестовані у другому тесті:

– зменшення до нуля випадків, коли неправильно рухаються;

– оптимізація процесу створення ворогів.

Таблиця 2 – Календарний план робіт 2-го тесту

Номер тесту	Дата початку	Дата кінця	Часи			
			14:10	14:45	15:00	16:00
1	15.05.2019	15.05.2019				
2						

Функції, які будуть протестовані у третьому тесті:

– тестування старту та виходу з гри;

– рішення проблем з переходом між сценами;

– тестування перезапуску рівня.

Календарний план робіт 3-го тесту представлено у таблиці 3.

Таблиця 3 – Календарний план робіт 3-го тесту.

Номер тесту	Дата початку	Дата кінця	Часи			
			10:30	10:35	10:40	11:00
1	16.05.2019	16.05.2019				
2						
3						

Функції, які будуть протестовані у четвертому тесті:

– знаходження непроходимих частин гри;

– рішення спростити чи підвищити складність у грі.

Таблиця 5 – Успішність виконання тестів.

Відмітка на виконання	Види тестів	Примітка
✓	Переміщення персонажу по ігровому полю.	При натисканні відповідних клавіш, персонаж рухався по ігровому полю.
✓	Тестування багів в поведінці ворогів.	Змінено та оптімізовано скрипти ворогів.
✓	Тестування усіх активних кнопок та переходу між сценами.	Тестування реакції виконання функцій при натискання на кнопки.
✓	Тестування усього ігрового процессу.	Проходження гри та покращення її проходження.

ВИСНОВКИ

Під час роботи над дипломним проектом було проаналізовано велику кількість джерел інформації щодо розробки програмних продуктів на платформі Unity, а в особливості – розробки інтерфейсу та написання комп’ютерних ігор. Можна зробити висновок, що Unity є одним з лідерів у індустрії серед існуючих програмних рішень.

Ця платформа має низький поріг входження, тобто не передбачає наявності масштабної бази знань у розробника. Вона має зручний, інтуітивно зрозумілий інтерфейс, що сприяє швидкому вивченню різноманітності елементів програми. Також, постійна підтримка розробників та своєчасне оновлення функціоналу дає змогу створити у найкоротший термін стабільний, якісний продукт, який відповідає сучасним запитам.

На сьогоднішній день ринок комп'ютерних ігор задає досить високу планку для розроблюваних продуктів. Сучасна гра повинна мати сильні сторони в області графіки, сюжетної складової, особливостей геймплея.
Підводячи підсумок, можна зауважити, що комп’ютерні ігри вже тривалий час є частиною нашого життя. Кількість їх жанрів і різновидів часом важко перелічити.

Як програмний додаток до дипломного проекту була розроблена комп’ютерна двовимірна гра у жанрі «tower defense», сутність виконаної роботи полягала в створенні гри і придбанні досвіду і навичок розробки. Розроблений проект має наступні характеристики: легкий та зрозумілий ігровий процес, цікава історія, приємна графіка і анімація у мультяшному стилі, зручне керування і можливість змінювати налаштування прямо під час гри.

Гра має звід правил, за якими проходить ігровий процес і може представляти інтерес будь-якому любителю ігор, або просто тому, хто хоче приємно провести дозвілля.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Все игры жанра «аркада / arcade», вышедшие в 2020 году. URL: https://www.kinonews.ru/games-arcade/. (дата звернення 02.03.2020).
2. Аркада – что это такое? (путаница с этим жанром видеоигр). URL: https://tangar.info/slovar/arkada. (дата звернення 02.03.2020).
3. Лучшие аркады на ПК. URL: https://cubiq.ru/luchshie-arkady-na-pk/. (дата звернення 02.03.2020).
4. Rayman Legends. Лучшие моменты. URL: https://www.igromania.ru/article/23707/Rayman_Legends.html. (дата звернення 10.03.2020).
5. Обзор основных движков. URL: https://vc.ru/pixonic/46004-development-engine. (дата звернення 11.03.2020).
6. Обзор игрового движка Unity3d. URL: https://gamedevmania.ru/engines/unity3d. (дата звернення 10.03.2020).
7. Движок Unity – особенности, преимущества и недостатки. URL: https://cubiq.ru/dvizhok-unity/. (дата звернення 10.03.2020).
8. Язык программирования C#: краткий обзор. URL: https://techrocks.ru/2019/02/16/c-sharp-programming-language-overview/. (дата звернення 10.03.2020).
9. Введение в Java. URL: https://metanit.com/java/tutorial/1.1.php. (дата звернення 10.03.2020).
10. Розробка uml діаграми варіантів використання. URL: https://stud-file.net/preview/5200239/page:6/. (дата звернення 20.03.2020).