ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: “РОЗРОБКА 2D КОМП’ЮТЕРНОЇ ГРИ "SLIMY" У ЖАНРІ ПЛАТФОРМЕР З ВИКОРИСТАННЯМ ІГРОВОГО РУШІЯ UNITY”

Виконав: студент 4 курсу, групи ПД–44
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)
Карасовський В. В.
(прізвище та ініціали)

Керівник Гребенюк В.В.
(прізвище та ініціали)

Рецензент
_____
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри
Інженерії програмного
забезпечення
_____ Негоденко О.В.
“____”_____2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

Карасовському Віталію Володимировичу

(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка 2D комп’ютерної гри “Slimy” у жанрі Платформер з використанням ігрового рушія Unity»

Керівник роботи: Гребенюк Віктор Вікторович, доцент кафедри, доктор філософії

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Затверджені наказом вищого навчального закладу від «16» лютого 2023 року
3. Вхідні дані до роботи:

4. Строк подання студентом роботи «1» червня 2023 року
5. Вхідні дані до роботи

5.1 Методи розробки відеоігрового програмного продукту;
5.2 Науково-технічна література з питань, пов’язаних з програмним забезпеченням щодо розробки відеоігор;

5.3 Офіційна документація Unity
5.4 Офіційна документація Microsoft Visual Studio
5.5 Офіційна документація мови програмування C#

6. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).
6.1Аналіз актуальності та проблематики розроблюваної гри
6.2 Аналіз та вибір інструментів для розробки даної гри
6.3 Опис проектування гри
6.4 Висновки
7. Перелік демонстраційного матеріалу (назва основних слайдів)
7.1 Титульний слайд
7.2 Індивідуальне завдання
7.3 Аналіз та загальна характеристика ігор-платформерів
7.4 Мета, об'єкт та предмет дослідження
7.5 Аналіз існуючих рішень
7.6 Технічні завдання
7.7 Висновки
7.8 Кінцевий слайд

8. Дата видачі завдання « 25 » лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	09.04.2022	Виконано
2	Дослідження аналогів та актуальності додатку	15.04.2022	Виконано
3	Аналіз та вибір інструментів для розробкидодатку	17.04.2022	Виконано
4	Проектування та реалізація	03.05.2022	Виконано
5	Вступ, висновки, реферат	16.05.2022	Виконано
6	Розробка обов‘язкових демонстраційнихматеріалів	17.05.2022	Виконано
7	Попередній захист роботи		
8	Здача роботи	01.06.2022	

Студент Карасовський В. В.
(підпис) (прізвище та ініціали)

Керівник роботи Гребенюк В.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи с., 4 рис., 42 джерела.

Ключові слова: відеогра, платформер, Unity, сюжет, механіка, ігровий двигун.

Об’єкт дослідження – гра жанру платформер.

Предмет дослідження – розробити 2D ігровий додаток Slimy з використанням рушія Unity в жанрі платформер з використанням випадкової генерації рівнів.

Мета роботи – розробити 2D ігровий додаток Slimy з використанням рушія Unity в жанрі платформер з використанням випадкової генерації рівнів.

Наукова новизна – перетворення існуючих ігрових механік жанру платформер для створення цікавої та легкою в розумінні гри, у яку гравець буде хотіти витрачати час.

У дипломній роботі проаналізовано ринок комп’ютерних ігор жанру платформер, знайдено переваги та недоліки інструметів для розробки.

Комп’ютерна гра була створена на рушії Unity, код був написаний в інтегрованому середовищі розробки Visual Studio 2022 мовою програмування C#.

Ця гра може бути використана як спосіб ненадовго відволіктися та витратити час на нескладний процес.

ЗМІСТ

ВСТУП

Ринок комп’ютерної індустрії наразі відіграє важливу роль в світовому ринку.

2017 рік став самим більшим в історії по продажам комп’ютерних ігор. Ігровий ринок достигнув позначки 107 млрд долларів США і 2,2 млрд аудиторії гравців по всьому світі.

В наш час, однією з вагових частин ігрового ринку являють багатокористувацькі ігри, які перелічують велику кількість. Серед таких жанрів перспективним напрямків являються розробка ігор сімейства Platformer. Хоч більшість з них являються примітивними, але вони продовжують набувати популярність.

Потенціал ринку і наявність попиту мотивує програмістів-початківців проявити себе в області розробки ігор.

Дана робота актуальна в зв’язку з тим, що ігри жанру “Platformer” почала набувати популярні в період початку коронавірусу і продовжує набирати оберти.

Таким чином, була вибрана ціль дипломної роботи, а саме: створити ігровий проект на основі Unity.

Для досягнення даної цілі потрібно:

- провести аналіз предметної області;
- засвоїти інструментальні засоби;
- створити концептуальну модель;
- здійснити малювання дизайну;
- здійснити програмну реалізацію ігрового проекту.

1. ЗАГАЛЬНІ ВІДОМОСТІ

1.1 Класифікація багатокористувацьких ігор

Багатокористувацька гра – тип відеоігор або її складова в яку одночасно грають декілька (два й більше) гравців, що може бути можливим в різних точках світу. [1]

Багатокористувацькі ігри можна поділити на такі види:

- Через локальну мережу;
- Через інтернет.

Багатокористувацькі ігри мають велику групу своїх представників, які поділяються на окремі жанри і діляться на окремі піджанри, до яких належать:

- MMORPG;
- MMORTS;
- MMOFPS;
- MMORG;
- MOBA;
- Браузерні ігри.

1.2 Геймдев, як професія

Розробник ігор, або геймдев (gamedev) – це людина, яка займається розробкою ігор в професійному напряму.

Перше, і не головне завдання геймдева – розробка коду. Саме за допомогою нього створюється механіка гри і будь-які супутні сервіси, які потрібні для роботи.

Серед обов’язків геймдева також може входити створення інструментів для розробки, за допомогою яких буде створюватися сама гра. Це необхідно, тому що це може прискорити та спростити розробку.

До плюсів даної професії можна перерахувати :

- **Актуальність**. Це нова професія, яка має великі перспективи для нового покоління розробників. Наразі є багато ресурсів і відеоуроків, які допоможуть зробити перший крок;
- **Висока заробітна плата**. Наразі середня ринкова заробітна плата варіюється в таких діапазонах:

  1 Junior – від 1000$ до 1500$;

  2 Senior – від 1500$ до 3000$;

  3 Middle – від 4000$ до 4500$;

  4 Team Lead – від 5000$.
- **Високі перспективи**. Наразі для того, щоб стати геймдевом не потрібно мати особливих навичок в розробкі. Компанії активною ведуть набори майбутніх розробників з подальшим їх навчанням.

Хоч ця професія здається багатообіцяючою, але вона має свої мінуси:

- **Велике навантаження**. Дана професія требує багато часу і відповідальності від сторони самого розробника – лише від нього залежить, як буде виглядити кінцевий продукт;
- **Сфера розробки ігор росте дуже стрімко, ніж інші галузі**. Таким чином розробнику потрібно бути завжди в курсі останніх технологій і підтримувати технічні знання, щоб його продукт не втратив актуальності;
- **Розробка невідомого продукту**. Так як цього ніхто ще не розробляв, то неможливо розрахувати приблизний час на розробку не існуючої технології.

1.3 MMORPG

MMORPG (Massively multiplayer online role-playing game) наразі являється самим популярним жанром багатокористувацьких ігор. Кожен гравець має можливість створити і редагувати зовнішність свого персонажу як забажає, щоб взаємодіяти з ігровим світом і іншими гравцями. Сенс гри заключається в тому,

щоб прокачувати свого персонажу, що включає в себе фарм (отримання золота і очків досвіду, одягу, зброї.), проходження квестів і багато чого іншого. За фарм і виконання завдань гравця нагороджують золотом і очками досвіду, які можуть витратити для покращення навичок свого персонажу і купівлю різних предметів. Наразі найбільш популярним представником свого жанру являється гра World of Warcraft (в спільноті називають WOW).

Рисунок 1 – World of Warcraft

1.4 MMORTS

MMORTS (Massively multiplayer online real time strategy) – наразі рідкісний жанр ігор, але не менше привабливий. Кожен гравець, має мати стратегічні навички, виконує дії, від яких буде залежати, що буде з його локацією. Подібного роду ігри, як правило, лише мають одну мету: завоювати як можна більше поселень

других гравців або нейтральні території. Найбільш популярними наразі представляється гра Age of Emperies Online.

Рисунок 2 – StarCraft 2

1.5 MMOFPS

MMOFPS (Massively multiplayer online first-person shooter) – шутер. Гравці на старті виконують закуп зброї і обладнання. Головна мета – завоювати перевагу над ворожою командою шляхом відстрілювання ворогів або захопленням точок. Яркими представниками даного жанру належать популярні серії ігор такі як: Counter-Strike, Battlefield, Call of Duty.

Рисунок 3 – Destiny 2

1.6 MMORG

Даний жанр включає в себе створення і покращення власної гоночного автомобіля, щоб змагатися з соперниками, які являються реальними гравцями.

Рисунок 4 – Asphalt 9 : Legends

1.7 MOBA

MOBA (Multiplayer Online Battle Arena) – займає 2 місце серед багатокористувацьких ігор (після MMORPG). Гра тісно пов'язана з жанром MMORPG, тому що жанр MOBA зародилась як окрема карта для гри World Of Warcraft. В іграх подібного жанру, як правило, відсутні завдання, а прокачування персонажу відбувається в межах однієї гри, яка триває від 30 хвилин до 2 годин. Наразі найбільш популярними представниками даного жанру являються Dota 2 і League Of Legends.

Рисунок 5 – Dota 2

1.8 Браузерні ігри

Браузерна гра — це різновид ігор, основною характеристикою якого є використання інтерфейсу браузера. Такі ігри представлені різноманітними

жанрами і, як правило, не вимагають встановлення іншого програмного забезпечення, окрім самого браузера та за потреби відповідних плагінів для нього (Flash, Java чи Silverlight).

Переважна більшість браузерних ігор орієнтовані на бізнес-модель Free-to-play, тобто, дозволяють грати безкоштовно якийсь час, поступово спонукаючи гравця витрачати реальні гроші для придбання ігрової валюти чи предметів. [2]

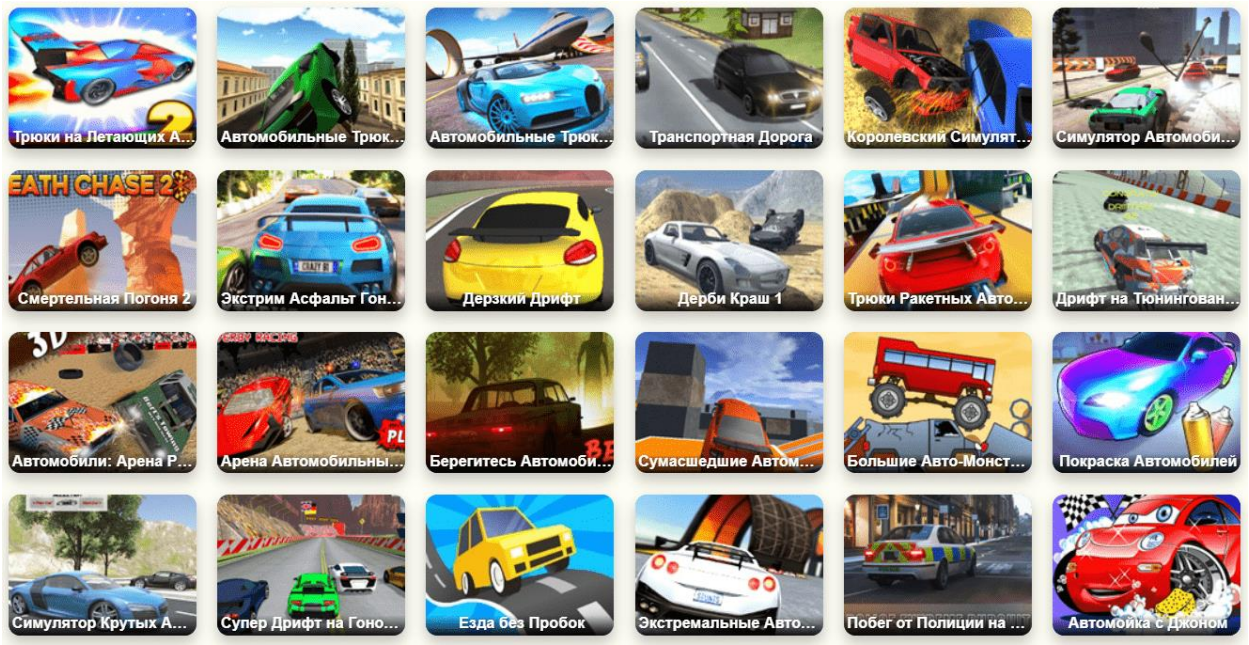


Рисунок 6 – Flash ігри

1.9 Platformer

Platformer – це популярний жанр комп’ютерних ігор, відеогра, головне завдання полягає в тому, щоб проходити через велику кількість платформ, обминати пастки або ворогів, збирати предметів (в більшості випадках потрібно для проходження рівня), як правило, стрибаючи.

Гадаю, що всі ознайомлені з серією ігор (або однією з них) Super Mario, де головним героєм являється звичайний сантехнік італійської національності, який хоче спасти свою принцесу від жахливого монстра.

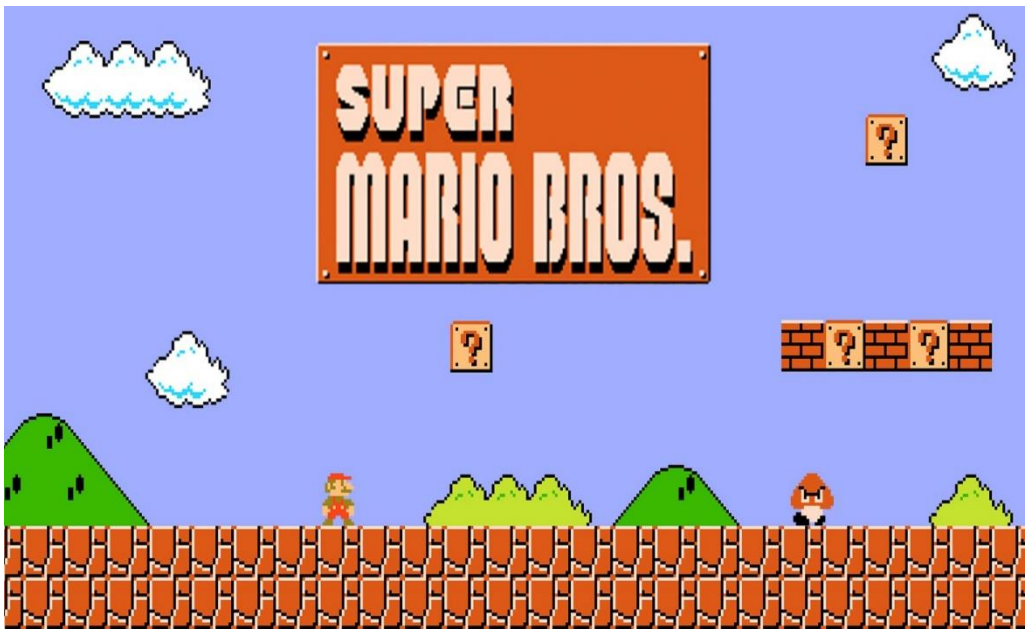


Рисунок 7 – Super Mario Bros.

Більшість Platformer ігор являються двовимірними (2D). На малюнках 1 і 2 представлені типові представники Platformer ігор - Dead Cells і Shovel Knight.

Рисунок 8 - Dead Cells

Рисунок 9 – Shovel Knight

1.10 Висновок по розділу

Багатокористувацькі ігри – це жанр, який стрімко розвивається. Його популярність пояснюється тим, що він має умовно необмежену свободу дій, яка представляється гравцям. Для розробників-початківців Platformer ігри – найбільш підходящий початок.

2 ІНСТРУМЕНТАЛЬНІ ЗАСОБИ

2.1 Unity

Unity — багатоплатформовий інструмент для розробки відеоігор і застосунків, і рушій, на якому вони працюють. Створені за допомогою Unity програми працюють на настільних комп'ютерних системах, мобільних пристроях та гральних консолях у дво- та тривимірній графіці, та на пристроях віртуальної чи доповненої реальності. Застосунки, створені за допомогою Unity, підтримують DirectX та OpenGL. [3]

На рисунку 10 зображений головний інтерфейс Unity2D.

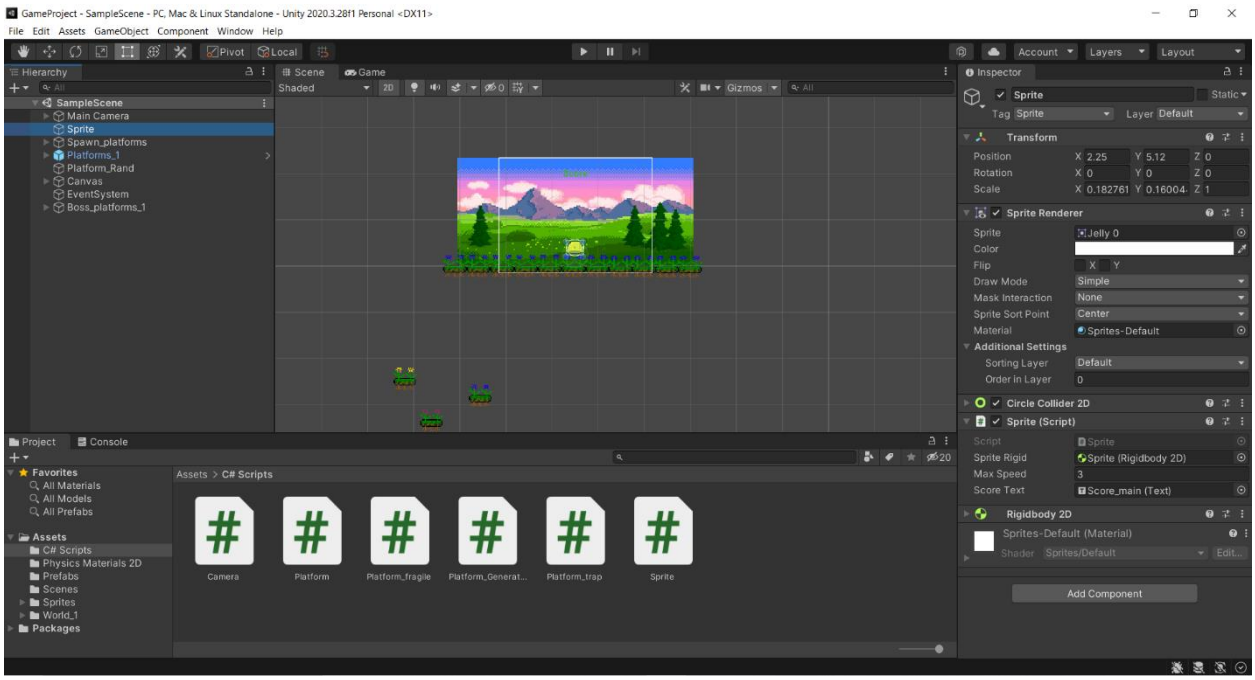


Рисунок 10 – Головний інтерфейс Unity

2.2.1 Project Window (Вікно Проекту)

Project Window відповідає за роботу з ресурсами, які знаходяться в проекті.

В нижній частині оглядачу містить список ієрархії, який відображає структуру папок в проекті. В вигляді іконок представлені окремі ресурси проекту, які указують їх тип(спрайт, скрипт і т.д.).

2.2.1 Hierarchy Window (Вікно Ієрархії)

Вікно містить всі об’єкти, які знаходять в сцені. До них можуть належати UI елементи (малюнки, кнопки і інше), екземпляри префабів (об’єкти користувача), 2D моделі і інше. Кожен об’єкт може містити в собі дочірні об’єкти або входити до складу об’єкту більш вищого рангу. Об’єкти можна вибрати в ієрархії і об’єднати його в інший об’єкт, утворюючи з ним батьківський зв’язок (parenting). Дочірний об’єкт наслідує всі значення свого батька.

2.2.2 Toolbar (Панель інструментів)

В Панелі Інструментів розташовуються елементи, які потрібні для трансформацію, кнопки запуски і зупинення гри, випадаюче меню, шари. Наразі Unity має всі потрібні функції для комфортної розробки.

2.2.3 Scene View (Огляд Сцени

В даному вікні встановлюється розміщення елементів гри. При потрібній необхідності можна змінювати розміри вікна.

2.2.4 Game View (Гра)

В даному вікні ми бачимо додаток очима користувача. За допомогою цього вікна тепер не потрібно білдити проект, щоб побачити фінальний результат.

2.2.5 Inspector Window (Інспектор)

Інспектор містить інформацію про конкретно вибраний об’єкт і його властивості. Тут можна міняти функціонал об’єктів в сцені.

2.2 MonoDevelop

MonoDevelop – відкрите інтегроване середовище розробки для платформ Linux, Mac OS X та Microsoft Windows, передусім націлене на розробку програм, які використовують і Mono, і Microsoft .NET framework. На даний момент підтримуються мови C#, Java, Boo, Visual Basic.NET, CIL, Python, Vala, C та C++. Також MonoDevelop підтримує такі технології, як Gtk#, ASP.NET MVC, Silverlight, MonoMac и MonoTouch.[4]

MonoDevelop має всі основні можливості, які необхідні для інтегрованого середовища розробки в нашому часі, а саме :

- підсвічення синтаксису, яку можна настроїти;
- автоматичне підставлення коду;
- згортання/розгортання виділених блоків коду;
- автоматичне формування відступів в коді;
- зручна навігація по коду (навігація по класам, методам, властивостями);
- візуальний редактор форм для проектів на Gtk#;
- можливість створення розкладок інтерфейсу і переключення між ними в любу хвилину;
- наявність великої кількості стандартних шаблонів проекту;
- після компіляції вихідного коду надається можливість автоматичного створення бінарних пакетів і архівів;
- можливість працювати з базами даних;
- створення застосунків з GUI, які підтримують кілька мов програмування;
- наявна інтеграція з Subversion;
- підтримка NUnit;
- документація створюється автоматично;

- розширення можливостей за рахунок доповнень і зовнішніх інструментів;
- наявність інтеграції з Microsoft Visual Studio і .NET Framework (в середовищі Microsoft Windows).

На рисунку 11 представлений інтерфейс роботи з додатком.

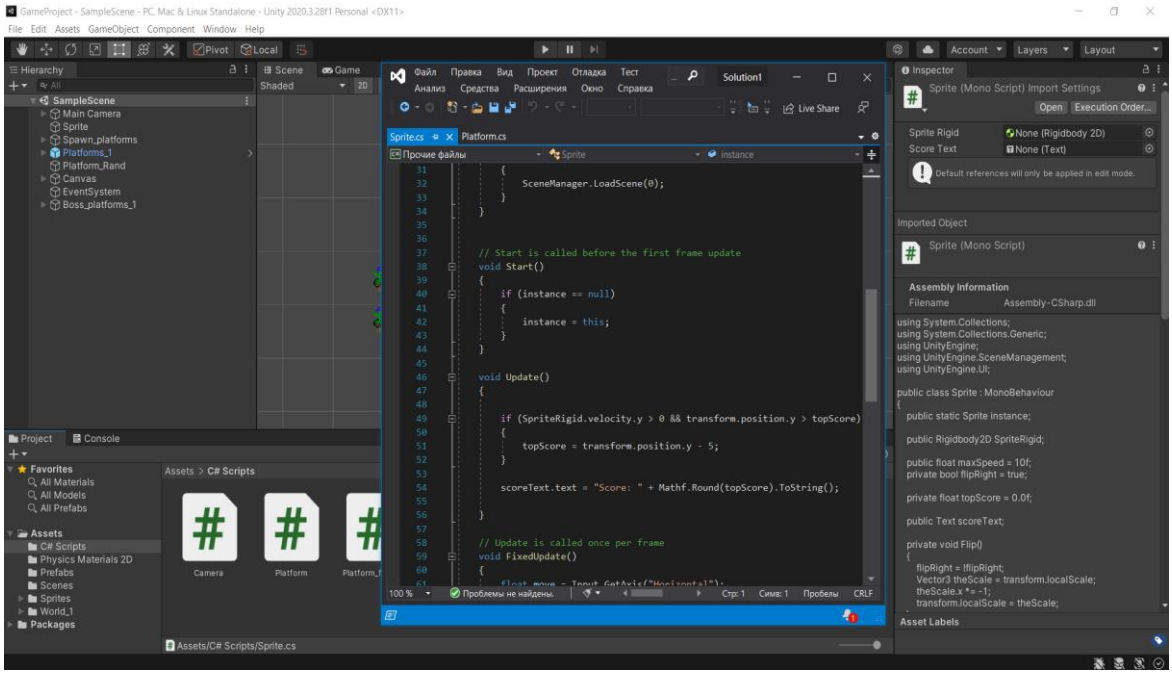


Рисунок 11 – MonoDevelop

2.3 Мова програмування C#

2.3.1 Основні відомості

C# (вимовляється Сі-шарп) — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (належить Microsoft).

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 від своїх попередників — мов C++, Object Pascal, Модула

і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, мова C#, на відміну від C++, не передбачає множинне успадкування класів.[5]

Дана мова програмування була вибрана в якості основного, так як володіє потрібними якостями для реалізації, має вбудовану підтримку подій і узагальнень, що полегшить її реалізацію.

2.3.2 Структура C# скрипту в Unity

Unity використовує компонентний вихід. Компонент – це клас, який успадковується від Example. Один компонент може відповідати тільки за одну поведінку.

Скрипт може бути створений в панелі проектору. Для цього потрібно натиснути на кнопку “Create” і вибрати потрібні налаштування, на якій будемо створювати скрипт. Приєднати скрипт к об’єкту можна шляхом перенесення або натисканням на кнопку “Add Component”.

Після того Unity створить і відкриє скрипт за замовчуванням в MonoDevelop, в якому можна буде побачити наступне:

```
using UnityEngine;
using System.Collections;
public class NewBehaviourScript : Example {
    // Use this for initialization
    void Start () {
    }
    // Update is called once per frame void Update () {
    }
}
```

Назва створеного скрипту повинна відповідати назві створеного автоматично створеного класу, який успадковується від вбудованого класу MonoBehaviour.

2.4 Abode Photoshop

Abode Photoshop – це графічний редактор, призначений для малювання ілюстрацій в векторі. Малюнок в векторній графікі на відміну від растрової графіки, можна легко і швидко масштабувати без збільшення пікселів. Малюнок не втрачає якість і має можливість виконувати печать в високій якості.

Векторну графіку застосовую в різноманітних напрямках: від web-розробки (створення дизайну сайту) до звичайного редагування зображення.

2.4.1 Навігація по проекту

Зменшення/збільшення виконується клавішами ctrl+/ctrl-. Переміщувати полотно можна за допомогою утримання Пробілу.

2.4.2 Налаштування тексту

Abode Photoshop має зручне налаштування тексту. Панель “Символ” налаштовує конкретно текст і його зображення. Абзац настроює групу тексту. В редакторі є в наявності художній текст.

2.4.3 Налаштування зображення

Всі зображення матимуть не великий розмір (навіть якщо це важкі конструкції) за допомогою вставлення зв’язків, а зображення, які будуть підготовлюватися до друку, будуть впроваджуватися у файл.

2.4.4 Розміщення сторінок

На полотні може розміщуватися більше однієї сторінки, розміщуватися в просторі в різній площині, при різних заданих розмірах.

2.4.5 Використання планшету

Використовуючи планшет можна створювати фігури любої складності і легко їх редагувати. Пензлик типу “Ляпка” реагує на натиски. Насиченість кольору залежить від сили натискання на планшет.

Інтерфейс програми і робота з нею представлені на рисунку 12.

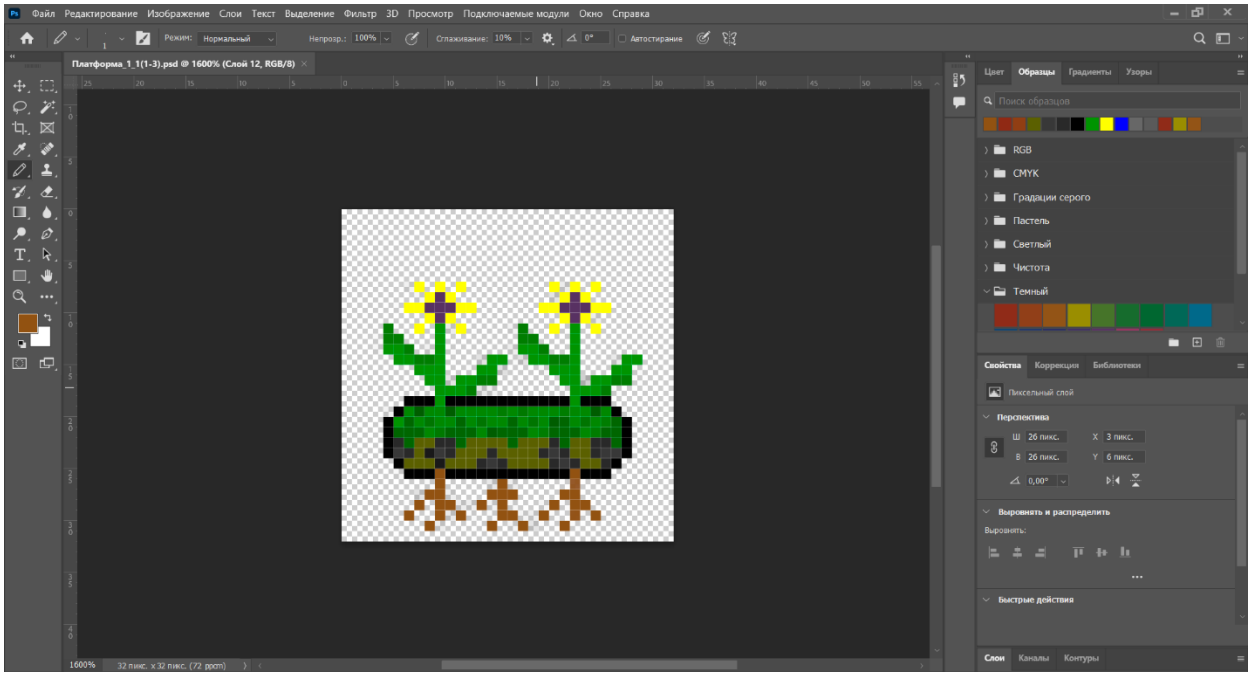


Рисунок 12 – Інтерфейс Abode Photoshop 2022

2.5 Висновок по розділу

Можна розробляти ігри на C#, але це непрактичний шлях. Щоб створити повномасштабну гру лише на C#, доведеться запрограмувати все самостійно. Буде неможливо мати доступ до активів, які запобіжать надмірності в розробці гри, і неможливо побачити оновлення в реальному часі, чим розробники ігор активно користуються. Правда в тому, що розробники ігор використовували C# для розробки ігор. І вони робили це за допомогою ігрового движка.

Багато популярних і вдосконалених ігрових движків використовують C# як мову програмування. Unity, Godot і UrhoSharp – лише деякі з них. Інженер-

програміст Гаррісон Феррон пояснює популярність C# серед розробників ігор і движків його доступністю, тобто цю мову відносно легше вивчити. Загалом розробники вважають, що їхній досвід розробки ігор із C# набагато легший, але на цьому переваги використання C# для розробки ігор не закінчуються.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Концептуальна модель

Концептуальна модель відіграє важливу роль в баченні фінального результату додатку. Її використовують в кожному проекті незалежно від складності і терміну.

Концептуальна модель – це найбільш важливий крок для створення ігрового проекту. На даному етапі геймдизайнер створює і описує свої ідеї в спеціальному документі для подальшої роботи і обдумання. Фінальним результатом повинен бути документ, який описує гру, який показує фінальний продукт, а також основні представлення всіх елементів гри. Далі документ будуть використовувати тестувальники, продюсери, дизайнери, програмісти і інвестори.

Так як даний проект був створений одним розробником без реальних навичок, то тільки має сенс описати концепцію лише в вигляді тезисів, а далі редагувати їх по ходу рішення практичних задач.

При описі концептуальної моделі ігрового проекту були сформовані наступні тезиси:

Жанр: Платформер.

Режим: Однокористуваць гра.

Графіка: Піксельна.

Простір: 2D.

Вигляд камери: Векторна.

Ціль: Отримати 1000 очків. Очки нараховуються в залежності від висоти від стартової платформи.

Світ: Складається з однієї сцени.

Гроші: Лічильник грошей має динамічну структуру:

- збільшується після підбирання монетки;

- операції над ресурсами мають прив’язку до лічильника.

Розгортання: програмний застосунок, Play Market.

Засіб: Unity, Microsoft Visual Studio.

Мова програмування: C#.

В результаті, в описаних тезисах були визначені всі деталі гри, описані ігрові об’єкти, а також позначені засоби проектування, які необхідні для реалізації проекту.

3.2 Малювання дизайну

На даному етапі слід створити зовнішній вигляд гри, елементи інтерфейсу і спрайти, а також вибрати вид камери. Графіка в 2D іграх можуть мати 3 види камери:

- вигляд зверху;
- вигляд збоку;
- ортографічна проекція.

В ортографічних проекціях напрямок променів проецирування перпендикулярно плоскості проецирування, де самі промені направлені вздовж осей X, Y, Z. Завдяки цьому виходить плоске зображення об’єкта в трьох напрямках. На рисунку 13 представлена відмінність ортографічної проекції від 3D.

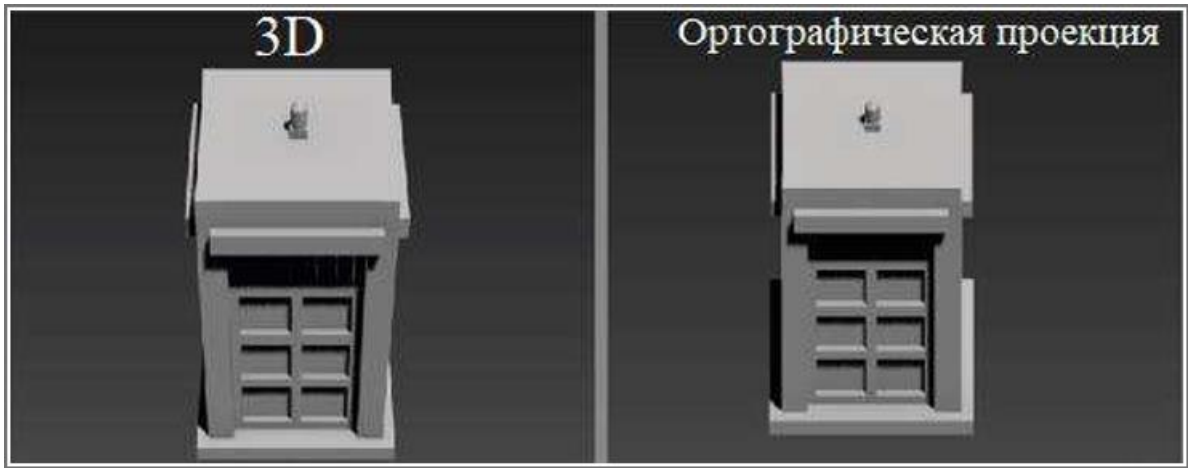


Рисунок 13 – 3D і ортографічна проекція

За допомогою програми Abode Photoshop 2022 були створені векторні зображення. Векторне зображення піддається масштабуванню без втрати якості, тому можна спочатку створити зображення, а потім підігнати їх під потрібне розширення. Розширення кожного зображення в пікселях може бути будь-яким, але в розробкі ігор прийнято брати розміри, керуючись формулою 2^n*(64x64, 256x256 і т.д.). Всі зображення будуть збережені в растровому форматі PNG, так як двигун Unity не сприймає векторні формати. В зв'язку з тим, рекомендується з самого початку визначити розміри зображень, щоб уникнути втрати якості при переносі зображення в середовище Unity.

Спрайти, або персонажі - це графічні елементи в комп'ютерній графіці. На рисунку 14 представлений спрайти головних героїв.

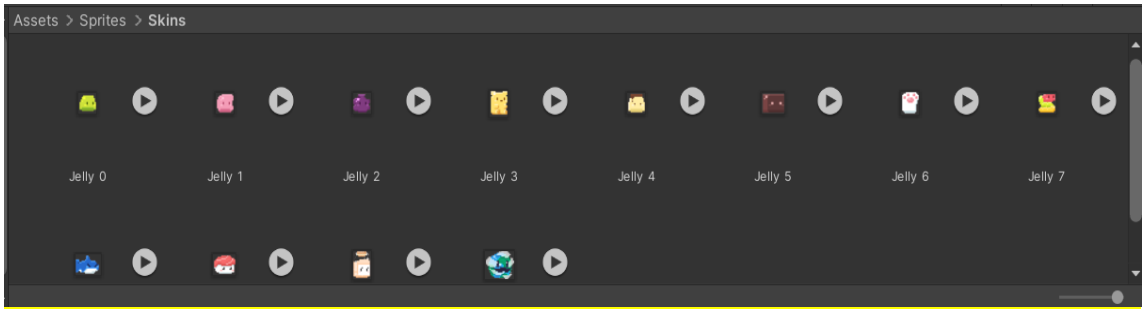


Рисунок 14 – Спрайти головних героїв

Наступним кроком будуть створення елементів інтерфейсу. Вони потрібні для того, щоб користувач зміг побачити оформлення локації – задній фон (background), платформи, пастки і інші елементи довкілля. На рисунку 15 зображені платформи 1 локації.

Рисунок 15 – Платформи 1 локації

3.3 Створення 2D макету в середовищі Unity

3.3.1 Основи Unity

В середовищі комп'ютерних розробок інтерфейс користувача в називають UI. UI – це довкілля, яке користувач бачe на своєму екрані, і з якою він може взаємодіяти. Раніше створення UI було не легкою задачею. В наш час була створена об'єктна модель, через яку можна створити інтерфейси за допомогою візуалізації.

Для того щоб створити любий UI елемент в середовищі Unity потрібно натиснути правою кнопкою миші в вікні ієрархії, вибрати UI, а потім вибрати нам потрібний графічний елемент. На рисунку 16 представлені всі доступні види UI в Unity.

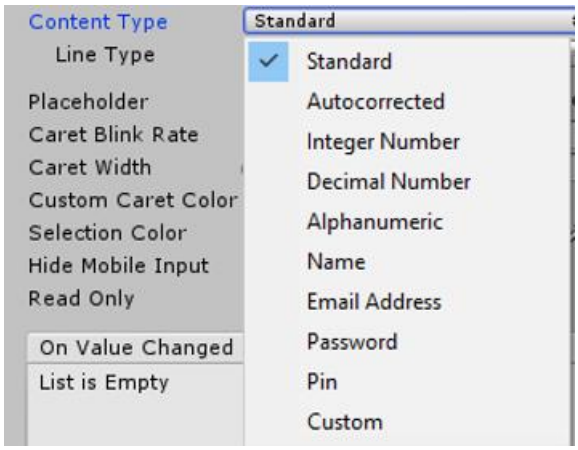


Рисунок 16 - Види UI в Unity

Любий графічний елемент створюється дочірнім до об’єкту Canvas. Об’єктна модель створення інтерфейсу в Unity має на увазі, що всі елементи мають бути тільки дочірніми до Canvas.

3.3.2 Canvas (Полотно)

Canvas – це елемент, який використовується для створення графічних елементів (малюнків, текстів, кнопок і інше). Canvas можна побачити на рисунку 17.

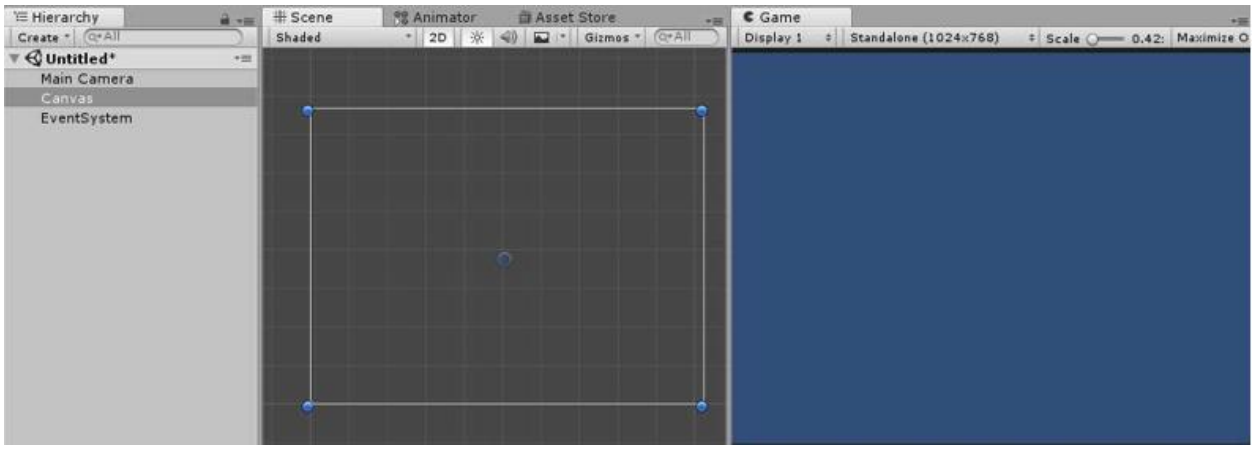


Рисунок 17 – Вигляд Canvas

Canvas змінюється автоматично в залежності від роздільності екрану. Властивості Canvas можна побачити на рисунку 18.

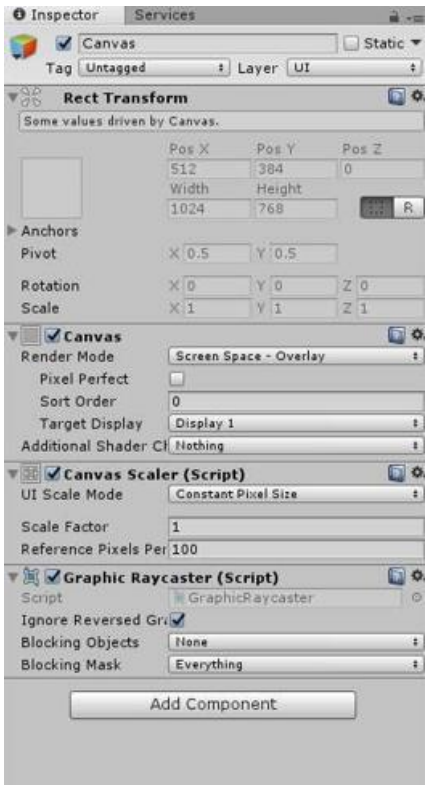


Рисунок 18 – Налаштування Canvas

Render Mode (рисунок 17) – це властивість, яка дає можливість налаштувати, як графічні елементи, яка знаходяться всередині Canvas, і повинна відображатися на екрані.

- Screen Space – Overlay – це всі графічні елементи, які відображаються поверх ігрової сцени. В цьому режимі під розмір екрану автоматично підлаштовуються розміри полотна.

- Screen Space – Camera – це всі графічні елементи, які відображаються за допомогою камери. Щоб це зробити, потрібно елемент Camera помістити в властивість Render Camera. В цьому випадку, можна впливати на зовнішній вигляд Canvas якщо змінювати налаштування камери.

- Word Space – всі UI елементи знаходяться в 3D просторі і вважаються звичайними 3D об’єктами.

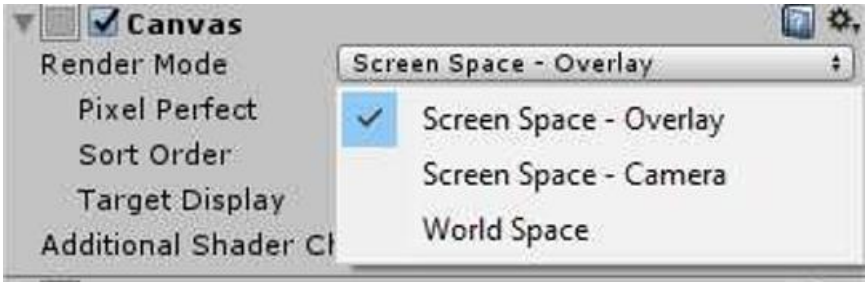


Рисунок 19 – Вигляд Render Mode

Прив’язати до однієї з дев’яти точок (до кутів, серединам сторін, або до центру екрана) можна любий створений на Canvas графічний елемент. Для цього в панелі Rect Transform (рисунок 20) можна використати функцію Anchor Presets (рисунок 21).

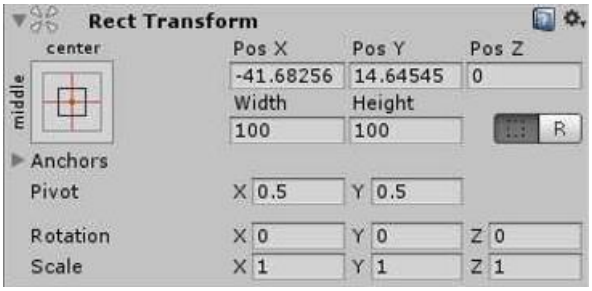


Рисунок 20 – Вигляд Rect Transform

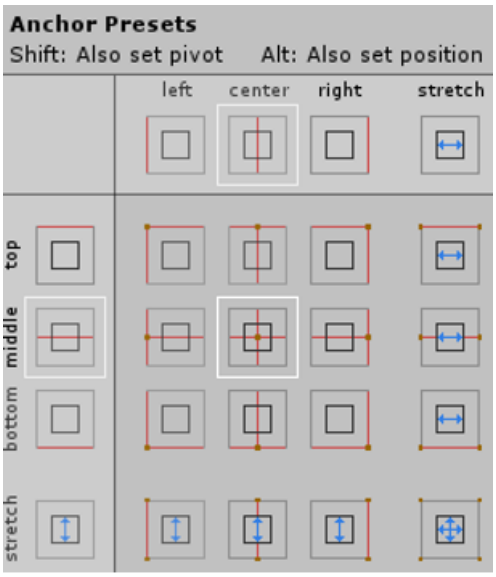


Рисунок 21 – Вигляд Anchor Presets

Координати місцезнаходження елемента будуть звітуватися від точки, яка була вибрана в якості “якорю”.

Також в любого елемента в панелі React Transform є властивість Pivot - це точка на самому елементі, від якої будуть звітуватися її координати (малюнок 20).

Text. Представляє собою елемент, відображаючий текст. Його можна задати в спеціальному полі або в скриптах.

Image. Його використовують для того, щоб розмістити будь-яке зображення на екрані. За умовчанням об’єкт зображення створюється завжди одного і того розміру, але використовуючи функцію Set Native Size в інспекторі Image можна змінити розміри зображення під потрібний розмір.

Button. Кнопка вважається важливим елементом в кожному проекті. На кнопку можна натискати, вставляти в нього текст або давати їй подію (Event). Компонент Button має властивість, яка позволяє налаштовувати ті функції, які будуть оброблятися по натисканню на кнопку. Щоб кнопкі зрозуміти яке їй дають завдання, потрібно описати це в окремому скрипті, при цьому функція потрібно ставити public, щоб вона мала до неї доступ.

Toggle. Це важкий елемент, який складається з кількох зображень і тексту. В дочірнім елементі Text можна вказати потрібний текст, а в елементі Image можна вказати зображення. **Toggle** має тільки одну подію, яка спрацьовує лише при зміні стану даного об’єкта.

Sider. Використовуючи даний елемент керування можна робити налаштування, наприклад, яскравість екрану або гучність звуків. Елемент також являється складовим: що складається з більш простих. Його дуже часто використовують в іграх для організації в інтерфейсі для створення лінії стану здоров’я або мани, завдяки вбудованій властивості, яка позволяє слайдеру замальовувати ту область, яка знаходиться лівіше повзунка.

Scrollbar. Приблизно схожий, як Slider, але використовують в складі більш складних компонентів, щоб могла бути взаємодія з Scrollbar.

Dropdown. Являє собою складовим і представляє собою список, що має можливість розкриватися. Елементи списку можна задати в налаштуваннях самого компоненту.

Input Field. Представляє собою текстове поле, в якому користувач може вводити які-небуть данні. Даний елемент має велику налаштувань, наприклад:

- заміна тексту з проханням ввести яку-небуть інформацію (дочірній елемент Placeholder);
- content type – це властивість, яка представляє собою список, що розкривається, який містить декілька елементів (малюнок 20). Вибрав один з них можна задати символи, які можна написати всередині Input Field (цілі числа, букви і інше);
- можливість коректування мигаючого курсору. Налаштувати можна частоту мигання, товщину, колір;
- елемент може бути однорядковим або багаторядковим (дається можливість при натисканні клавіші Enter перейти на інший рядок);

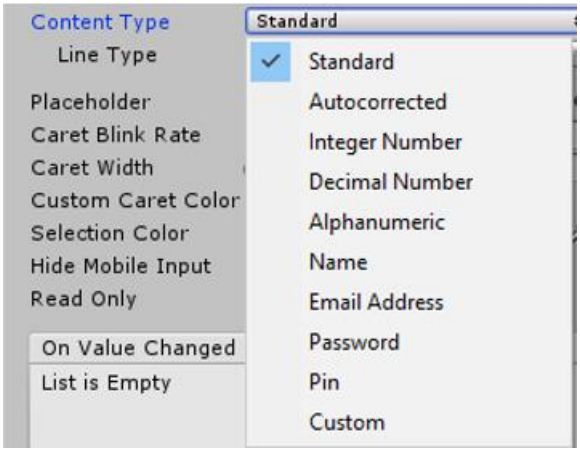


Рисунок 22 – Content type

На відміну від всіх минулих елементів інтерфейсу у даного елемента присутні зразу 2 події, які спрацьовують при роботі з ним. Перший – подія, яка спрацьовує при любій зміні. Другий – спрацьовує після того, як користувач закінчив печатати.

Panel. Представляє собою підкладку, на якій можна помістити любий інший UI.

Scroll View. Комбінований елемент, який ґрунтується на компоненті Scroll Rect. Для нормального функціонування потрібно декілька властивостей:

- площа покриття, всередині якої елемент буде прокручуватись цілий контент;
- контент;
- scrollbar, який буде прокручувати елементи.

3.4 Написання скриптів

В Unity скрипти переглядаються в якості об’эктів, відповідаючих за поведінку. Так як і інші компоненти в Unity, вони можуть бути прикрілені до об’єктів. Це можна побачити у вікні інспектора.

Скрипти позволяють реалізувати любі варіанти поведінки об’єкта. Скрипт – це інструмент, за допомогою якого можна вибирати поведінку для любого об’єкта в ігрі. Це можуть бути персонажі, об’єкти середовища, або, наприклад, розширення функціональності ігрового геймплею. Скрипти можуть бути використані навіть для створення графічних ефектів або реалізації штучного інтелекту.

3.5 Діаграми

UML (Unified Modeling Language) - уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення.

UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю.

UML був створений для визначення, візуалізації, проєктування й документування в основному програмних систем. UML не є мовою програмування,

але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

В мові UML існує 12 типів діаграм:

- 4 типи представляють статичну структуру застосунку;
- 5 типів представляють поведінкові аспекти системи;
- 3 типи представляють фізичні аспекти функціонування системи (діаграма реалізації).

Самими доступними діаграмами для застосунку являються:

- Діаграма прецедентів (Use-case diagram);
- Діаграма класів (Class diagram);
- Діаграма активності (Activity diagram);
- Діаграма розгортання (Deployment diagram);
- Діаграма послідовності (Sequence diagram);
- Діаграма об’єктів (Object diagram);
- Діаграма співробітництва (Collaboration diagram);
- Діаграма стану (Statechart diagram). [6]

3.5.1 Діаграма прецедентів

Діаграма прецедентів – це діаграма, яка відображає співвідношення між actor(учасником) і Use case (прецедентами) і являються основною частиною моделі прецедентів, яка позволяє описати систему на концептуальному рівні.

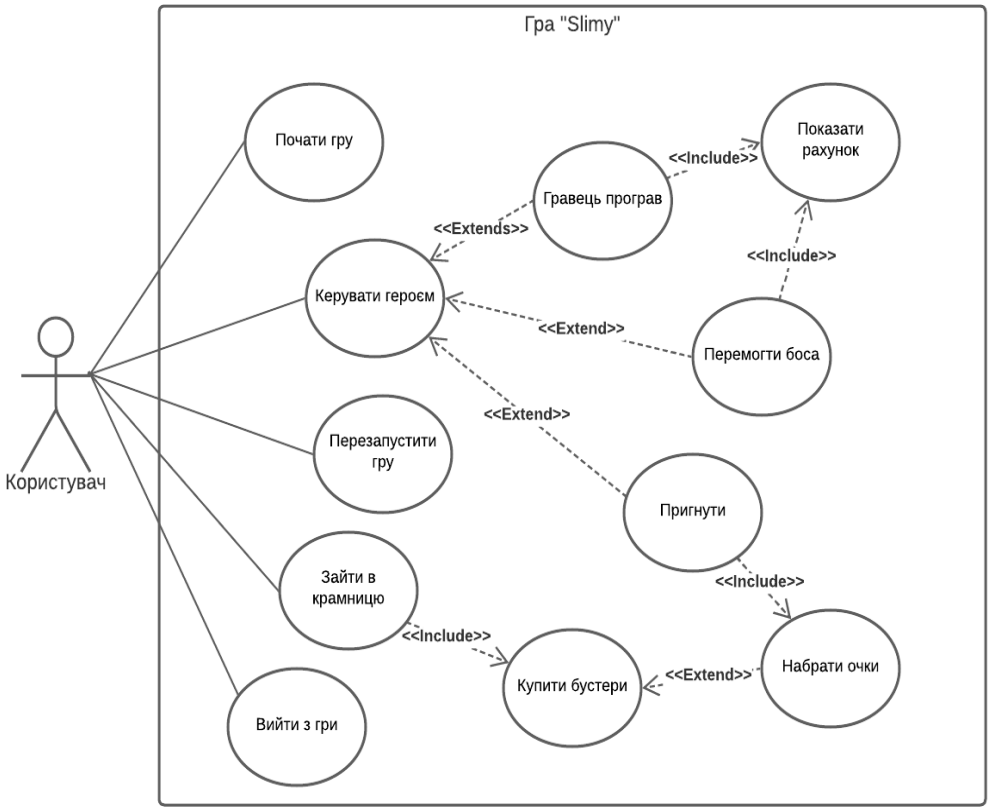


Рисунок 23 – Діаграма прецедентів

В даній діаграмі можна побачити, що користувач даної програми може виконувати такі функції:

1) Почати гру – користувач користується програмою;

2) Керувати героєм за допомогою клавіатури або сенсорного екрану;

3) Користувач може перезапустити рівень при програші;

4) Користувач може зайти в крамницю, де може придбати бустери за очки, які отримані в процесі проходження рівнів;

5) Гравець може в будь-який час вийти з програми.

3.5.2 Діаграма класів

Діаграма класів – це діаграма, яка являє собою набором декларативних, статичних елементів моделі. За допомогою неї можна зрозуміти повне і розгорнути представлення зв’язків в програмному коді, функціональності і

інформації про окремих класах. Застосунок генерується найчастіше за допомогою діаграми класів.

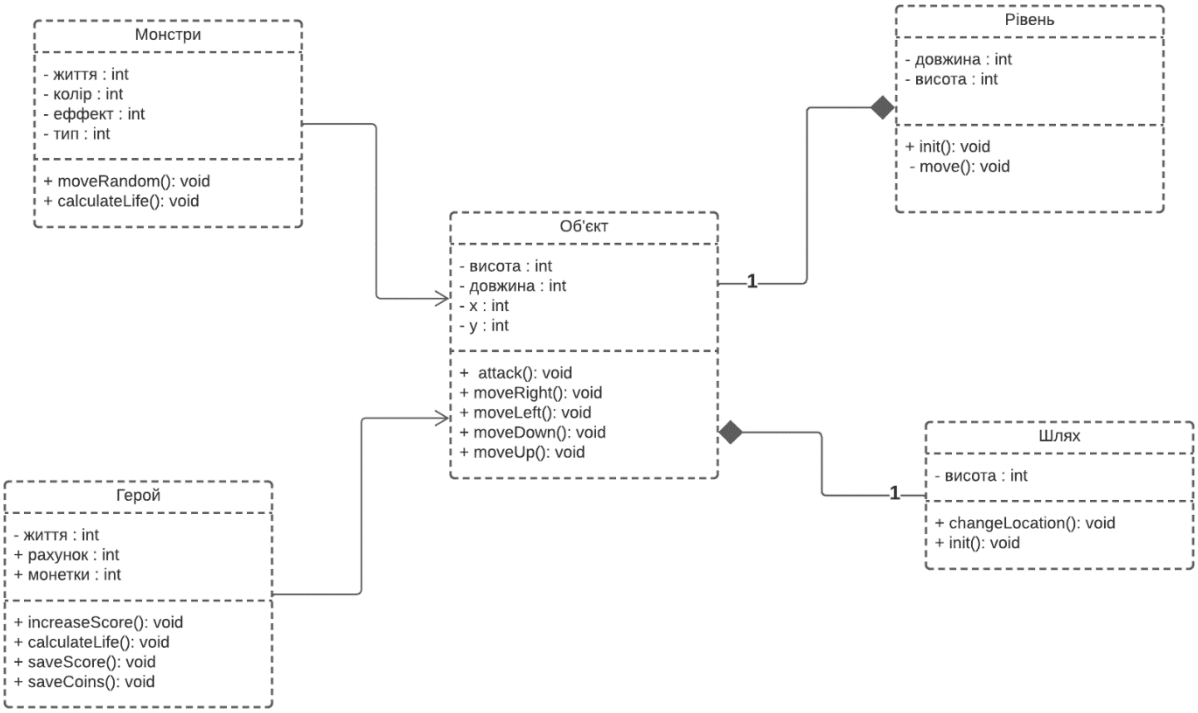


Рисунок 24 – Діаграма класів

Для класу “герой” є таблиця, яка має наступні атрибути: життя, колір, ефекти, тип. В даній діаграмі можна побачити зв’язки цієї сутності з іншими : з яким об’єктом або монстром відбувається взаємодія, на якому рівні знаходиться і який шлях йому залишився для зміни локації.

3.5.3 Діаграма активності

Діаграма активності - це візуальне представлення графу активності. Граф активності є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дії.[7]

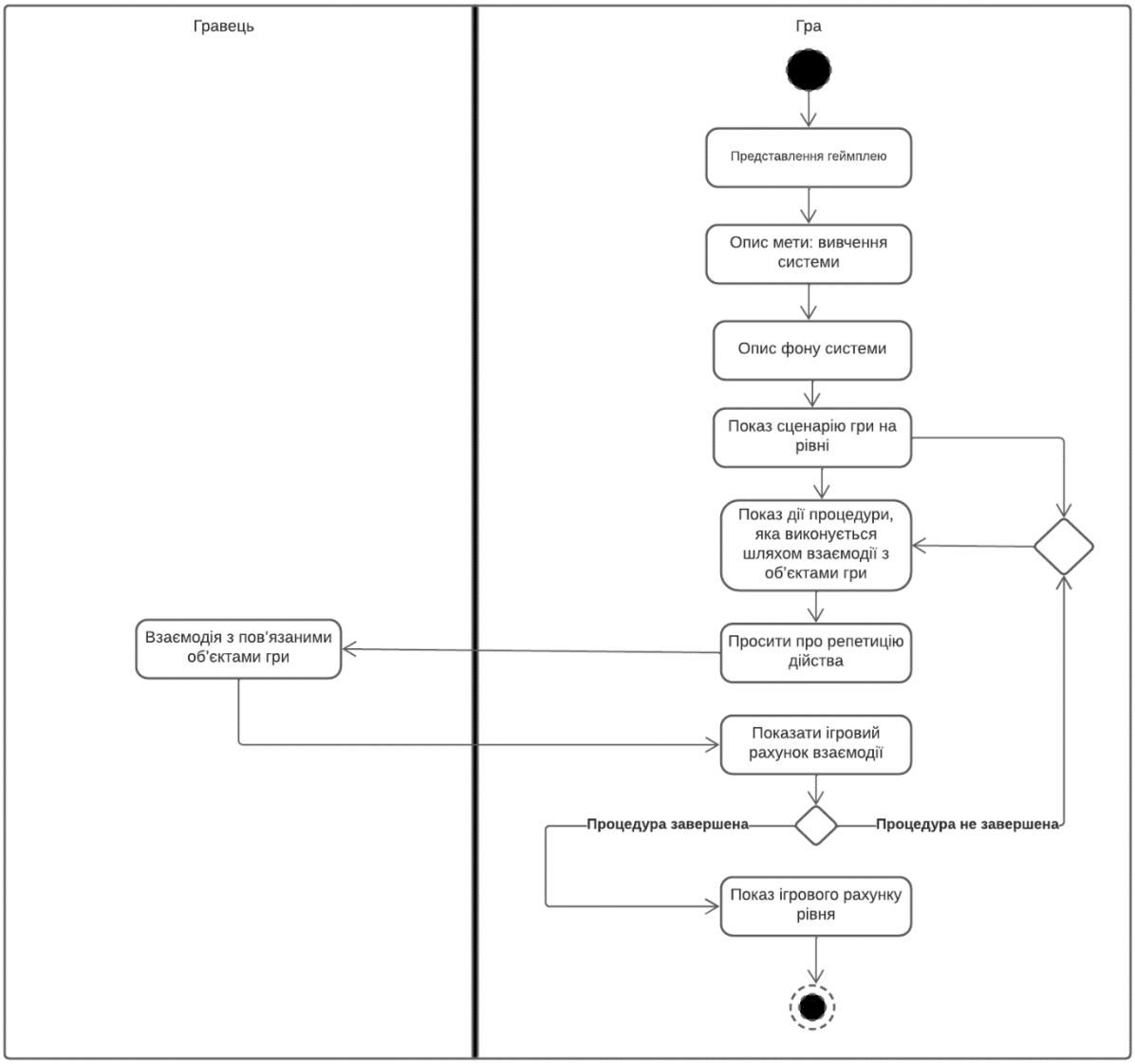


Рисунок 25 – Діаграма активності

Діаграма активності для застосунку максимально показує, які є інтеграції в даній системі. Актор (у нашому випадку - гравець), який зайшов в застосунок, бачить ознайомлення з грою – її основну сюжет і геймплей. Далі можна побачити, що у нас відбувається розгалуження: гравець програв чи пройшов рівень (Так/Ні). Якщо він програв, то може вернутися до минулого рівня. Якщо користувач пройшов рівень, то він може почати наступний.

3.5.4 Діаграма послідовності

Діаграма послідовності – різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність надісланих повідомлень. [8]

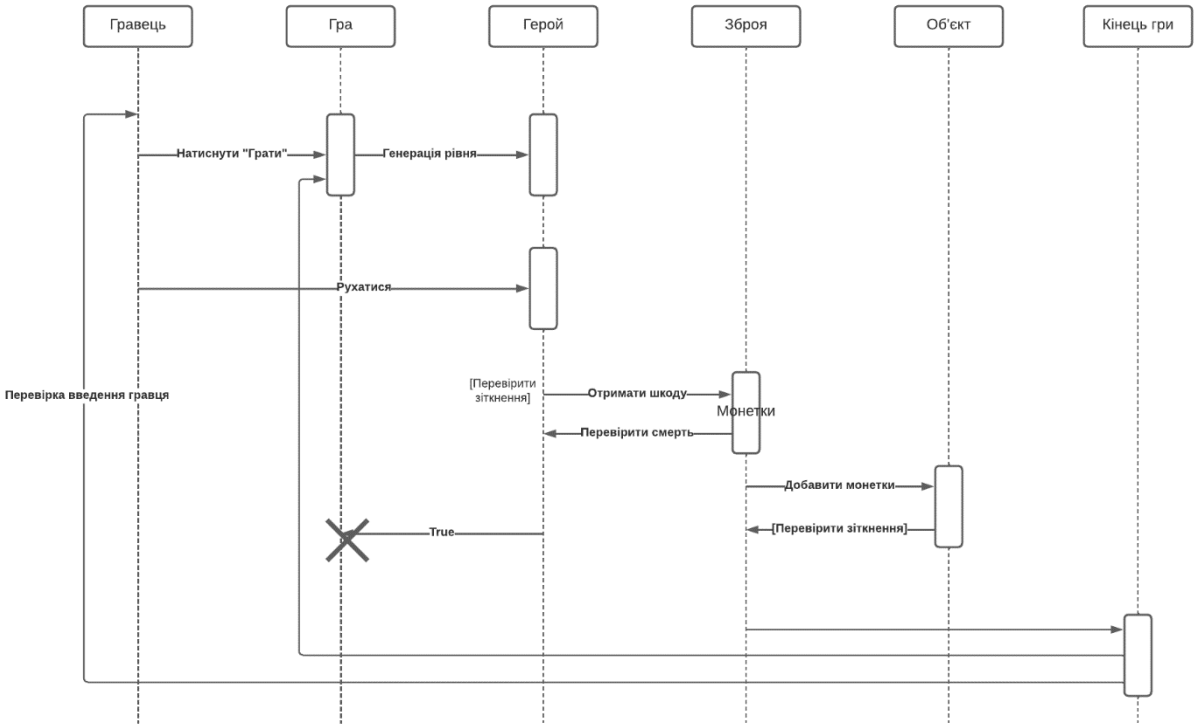


Рисунок 26 – Діаграма послідовності

3.5.5 Діаграма розгортання

Діаграма розгортання — це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. [9]

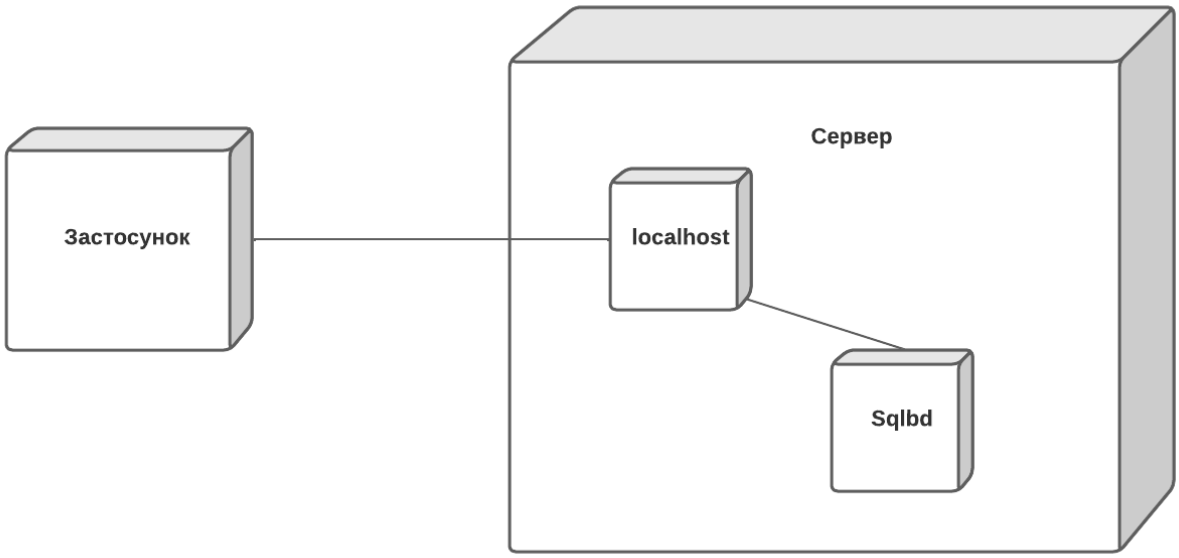


Рисунок 27 – Діаграма розгортання

Таблиця 1.1 Структура застосунку

Вузол	Опис.
Застосунок	Застосунок
Localhost	Сервер застосунку. Здійснює зв’язок з базами даних і самим застосунком.
Sqlbd	Системна база даних

3.5.6 Аналіз структури застосунку

При проведенні аналізу структури застосунку, була складена приблизна схема компонентів застосунку (рисунок 28):

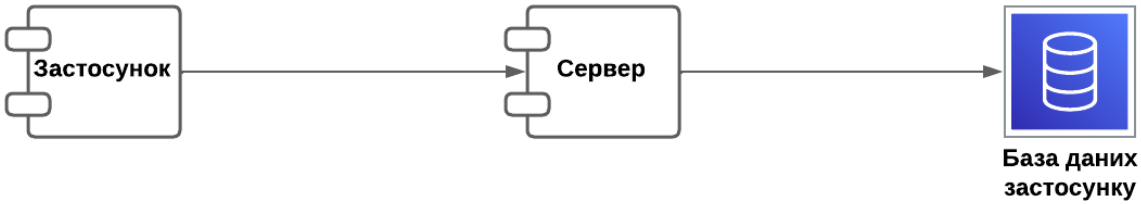


Рисунок 28 – Схема компонентів застосунку

База даних застосунку зберігає інформацію про користувачів і інші системні таблиці.

3.5.7 Діаграма об’єктів

Діаграма об'єктів — це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів.

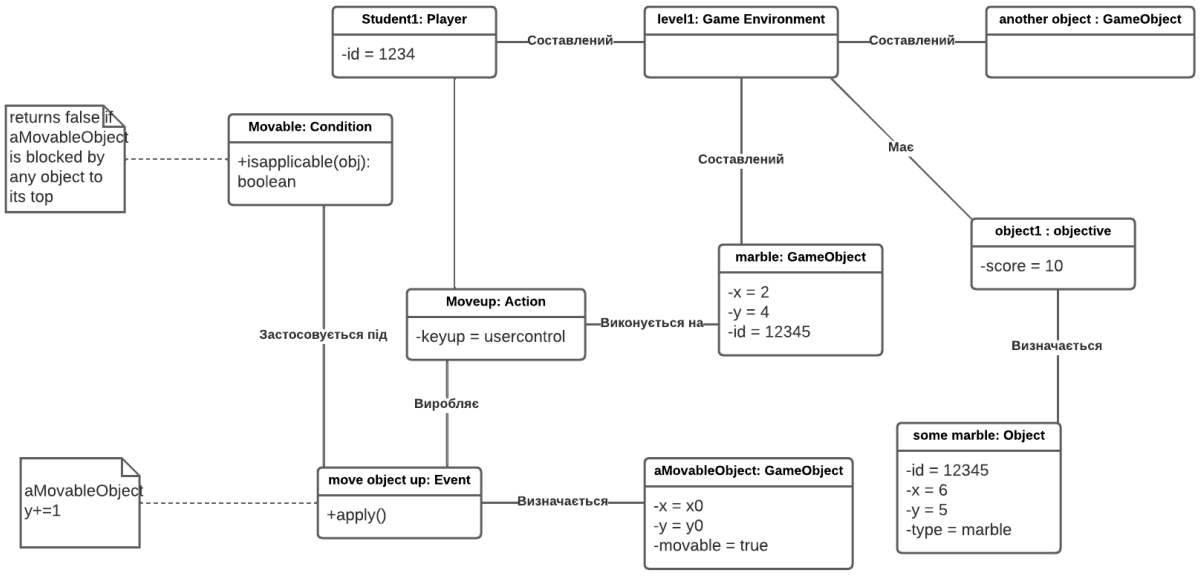


Рисунок 29 – Діаграма об’єктів

3.6 Кнопки

Напишемо кілька простих скриптів для реалізації кнопок. По натисканню кнопки Start повинна відкриватися сцена, а по натисканню кнопки + - магазин.

Для MainMenu добавляємо новий компонент MenuControl.cs, відкриваємо його і пишемо скрипт, як на рисунку 30:

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

public class MenuControls : MonoBehaviour {

    public void PlayPressed()
    {
        SceneManager.LoadScene("main");
    }

}

```

Рисунок 30 – Скрипт для переходу в іншу сцену

Прикріпляємо подію кнопки до цього методу. Натискаємо кнопку Start і добавляємо метод, натискаючи на +, як показано на рисунку 31:

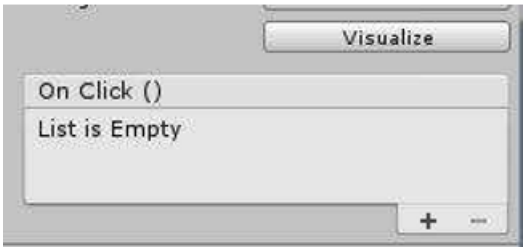


Рисунок 31 – Добавлення методу

В вікні, що появилося, потрібно перетягнути об’єкт, в якому міститься потрібний скрипт. В нашому випадку це Main Menu.

Після того потрібно вибрати скрипт MenuControls і найти метод PlayPressed() (рисунок 32).

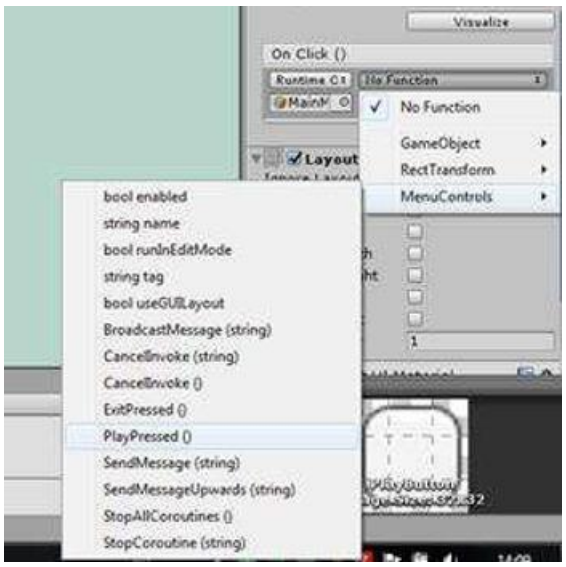


Рисунок 32 – Вибір методу скрипту

Тепер створемо кнопку для переходу всередину сцени main. Для цього створемо скрипт ButtonController і пропишемо наступне (рисунок 33):

Рисунок 33 – Вибрані кнопки

На саму кнопку переміщуємо об’єкт зі скриптом і виберемо функцію ShowBuy, як показано на рисунку 34.

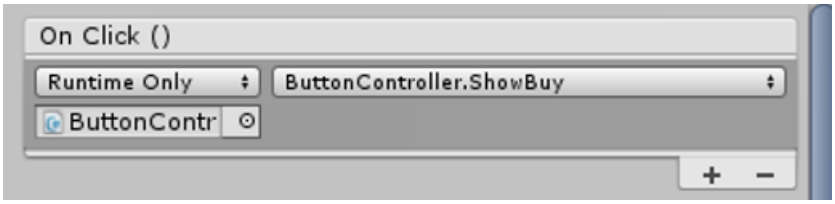


Рисунок 34 – Вибір функції скрипту

Тепер робимо аналогічні дії для того, щоб по натисканню на кнопку ми вертались в вихідну сцену.

3.7 Монетки

Збирання та колекціонування монет стали основними елементами двовимірних ігор, особливо у двовимірних платформерах .

Щоб підібрати монету в Unity, потрібно створити сценарій, який буде прикріплено до об’єкта монети та буде знищено, коли гравець контактуватиме з нею, оновлюючи значення лічильника.

На рисунку 35 зображений інтерфейс (лічильник) монет в грі.

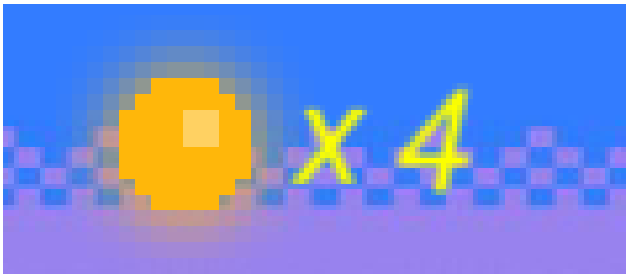


Рисунок 35 – UI лічильника монет

Також в грі є крамниця, де можна купити бустери (рисунок 36):

- Додаткове життя – 300 монет за 1шт.;
- Супер стрибок (10 секунд) – 100 монет за 1шт.;
- Імунітет до пасток (10 секунд) – 150 монет за 1шт..

Рисунок 36 - Магазин

3.8 Локації

Локації будуть змінюватися при досягненні певної кількості очків (рисунок 37). На малюнку можна побачити, що при досягненні гравцем певних очків викликається змінна target, яка відповідає за зміну спрайтів і заднього фону.

```
void Update()
{

    if (SpriteRigid.velocity.y > 0 && transform.position.y > topScore)
    {
        topScore = transform.position.y - 5;
    }

    scoreText.text = "Score: " + Mathf.Round(topScore).ToString();

    if (topScore > 99.0f)
    {
        target.transform.localPosition = new Vector3(-5.0f, 0f, 0f);
    }

    if (topScore > 199.0f)
    {
        target2.transform.localPosition = new Vector3(-5.0f, 0f, 0f);
    }

    if (topScore > 299.0f)
    {
        target3.transform.localPosition = new Vector3(-5.0f, 0f, 0f);
    }

    if (topScore > 399.0f)
    {
        target4.transform.localPosition = new Vector3(-5.0f, 0f, 0f);
    }

}
```

Рисунок 37 – Зміна локації

3.9 Системні вимоги

Мінімальні вимоги для застосунку представлені в таблиці 2.1:

Таблиця 2.1 Мінімальні системи вимоги

1. Операційна система	32/64-розрядна Windows 10 версії 14393.0 або новішої
2. CPU	Архітектура Intel P4/NetBurst або еквівалент AMD (AMD K7)
3. GPU	GeForce3 либо RadeonR100 (7xxx)
4. HDD	Хоча б 40 МБ для ігрових даних
5. Оперативна пам’ять	256 Mb

Рекомендовані вимоги для застосунку представлені в таблиці 2.2:

Таблиця 2.2 Рекомендовані системи вимоги

1. Операційна система	64-розрядна Windows 10 версії 14393.0 або новішої
2. CPU	Intel Pentium D або AMD K8-Based CPUs
3. GPU	GeForce 7300 GT або ATI Radeon HD 2400 XT
4. HDD	150 МБ для зберігань і звуків
5. Оперативна пам’ять	1 Gb

3.10 Тестування

Тестування – це важлива частина в розробці кожного програмного продукту. За допомогою тестування можна визначити які проблеми має продукт на своєму етапі розробки. Серед таких проблем вважаються:

- Логічні і функціональні;
- помилки часу виконання і обчислень;
- помилки вводу-виводу і маніпулювання даними;
- помилка інтерфейсу;

Для тестування мого проекту було прийнято рішення загрузити проект на в веб-сервіс Github і залишити в відкритому доступі.

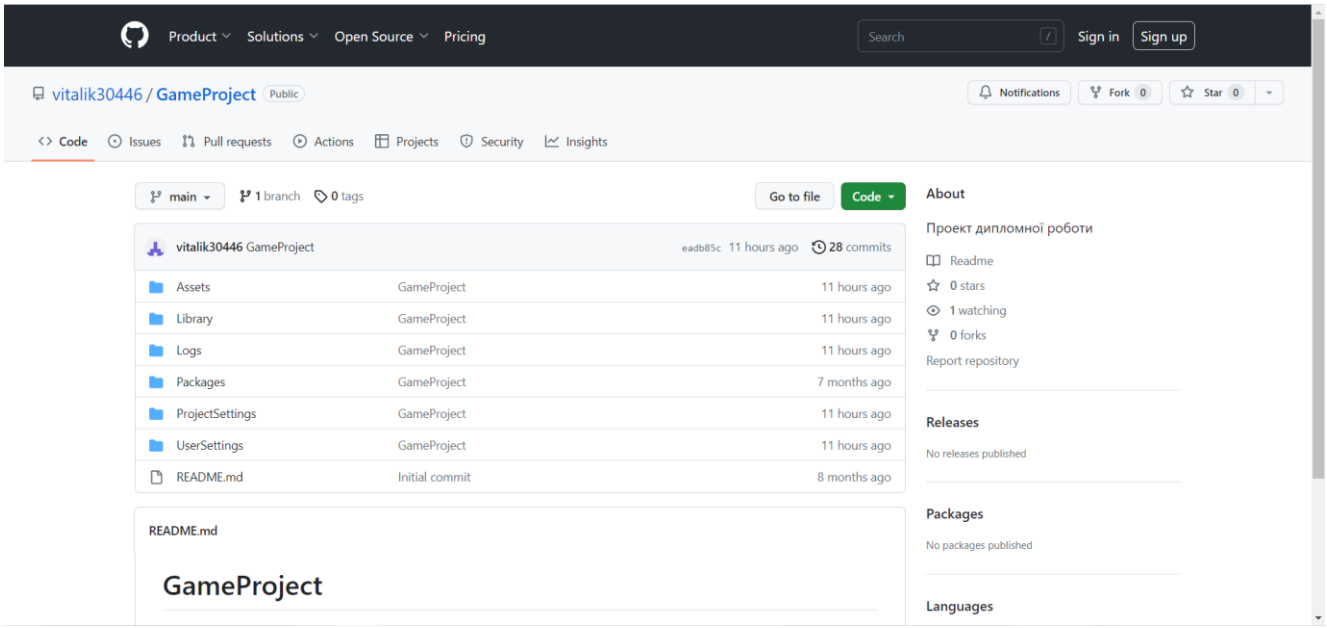


Рисунок 38 – вигляд проекту в GitHub

Наступним кроком було найти волонтерів, які зможуть протестувати даний додаток. Для цього я вирішив розповісти про свій проект в телеграм чата гуртожитку і своєї кафедри. В результаті набрались волонтери в кількості 43 людей. Кожній групі волонтерів я дав завдання знайти певні види помилок.

Для полегшення роботи тестування і власного підведення підсумків було вирішено створити гугл форму, де тестувальники могли вибрати вид помилки і детально описати її.

В кінці тестування були підведені підсумки, що були знайдені такі проблеми:

- Помилка інтерфейсу (14 голосів) – якщо в магазині натиснути дуже швидко на придбання бустера, то бустер міг придбатися навіть якщо немає коштів.
- помилка інтерфейсу (19 голосів) – під час зміни рівня спрайт або задній фон локації міг зникнути, хоч гра продовжується.
- проблема двигуна (8 голосів) – при використання бустерів в правильному порядку відбувалося вічне збільшення швидкості і висоти стрибка героя, хоч бустер міг лише бути активним 10 секунд.

Після оголошення результатів тестування було прийнято рішення негайно виправити помилки, які були знайдені тестувальниками.

3.11 Висновок по розділу

Для любої гри потрібний ігровий двигун. Ігровий двигун – це серце нашої гри, в ній ми втілюємо всі наші ідеї, перероблюємо логіку гри і взаємодію ігрових об’єктів.

Ринок ігрових двигунів наповнений великим різноманіттям. Є двигуни, які тільки розраховані для створення 2D ігор, а є які тільки для 3D, але на мою думку більш універсальним варіантом, ніж Unity і Unreal Engine поки не придумали. В них є всі необхідні інструменти для виконання робочого процесу, але в них є свої переваги і недоліки. Я пробував обидва продукти, але для цього проекту я вирішив вибрати Unity.

ВИСНОВКИ

В процесі вивчення предметної області, була виявлена і описана класифікація багатокористувацьких ігор. У рамках проекту було розглянуто 6 напрямків багатокористувацьких ігор: MMORPG, MMORTS, MMOFPS, MMORG, MOBA і Platformer, проте, існує ще безліч різних жанрів, але так як вони менш поширені або являються копіями, вони тут не переглядалися.

Також були визначені і освоєні інструментальні засоби, за допомогою яких буде відбуватися реалізація ігрового проекту. Основною і найбільш важною програмою для розробки ігор являється двигун, в нашому випадку їй являється Unity. Unity використовує мову C# і має закритий вихідний код.

При складанні концептуальної моделі було прийнято рішення створити гру в жанрі Platformer, Platformer ігри – одно/багатокористувацькі ігри в 2D/3D графіці. Виходячи з цього були підібрані і створені ігрові спрайти і елементи інтерфейсу в програмі Abode Photoshop 2022, з наступним використанням їх в середовище Unity.

При програмуванні в середовищі MonoDevelop на двигуні Unity, були створені скрипти для кнопок, локації, лічильника і різних операцій над ресурсами.

Таким чином, в процесі написання дипломної роботи, було реалізовано наступне:

- проведений аналіз предметної області;
- освоєні інструментальні засоби;
- створена концептуальна моделі;
- здійснено малювання дизайну;
- здійснена програмна реалізація ігрового проекту.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Multiplayer Video Game [Електронний ресурс]. // Wikipedia. - Режим доступу: https://en.wikipedia.org/wiki/Multiplayer_video_game (дата звернення: 08.10.2022)
2. Браузерна гра [Елетронний ресурс]. // Wikiwand. – Режим доступу: https://www.wikiwand.com/uk/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80%D0%BD%D0%B0_%D0%B3%D1%80%D0%B0 (дата звернення: 05.11.2022)
3. Unity (ігровий двигун) [Електронний ресурс]. // Wikipedia. – Режим доступу: https://ru.wikipedia.org/wiki/Unity (дата звернення 05.11.2022)
4. MonoDevelop [Електронний ресурс]. // MonoDevelop. – Режим доступу: https://www.monodevelop.com/documentation/
5. C Sharp // Wikipedia. – Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp (дата звернення: 08.10.2022)
6. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс]. // Evergreens . – Режим доступу: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення 10.10.2022)
7. Архітектура ігрових додатків [Електронний ресурс]. // Europa. – Режим доступу: https://ec.europa.eu/programmes/erasmus-plus/project-result-content/1af09656-090c-42c5-bb89-04d21810b880/07_Bachelor_Game%20Applications%20Architecture.pdf (дата звернення 10.10.2022)
8. Діаграми UML. Діаграми послідовностей [Електронний ресурс]. Naurok. – Режим доступу: https://naurok.com.ua/diagrami-poslidovnostey-uml-240152.html (дата звернення 10.10.2022)
9. What is Deployment Diagram? // Visual-Paradigm. – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/ (дата звернення 05.11.2022)

ДОДАТОК А

ПРЕЗЕНТАЦІЯ ДО ЗВІТУ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА 2D КОМП'ЮТЕРНОЇ ГРИ "SLIMY" У ЖАНРІ ПЛАТФОРМЕР З ВИКОРИСТАННЯМ ІГРОВОГО РУШІЯ UNITY

Виконав студент 4 курсу

Групи ПД-44
Карасовський Віталій Володимирович
Керівник роботи

доктор філософії ,доцент кафедри ІПЗ Гребенюк Віктор Вікторович

Київ – 2023

Індивідуальне завдання

- Провести огляд методик для оцінки якості додатку та визначити особливості створювання рушія в сучасних компаніях, визначити та описати метрики і показники ефективності оцінки;
- Проаналізувати програмні, технічні, та інші рішення, що використовуються на поточний момент для створення ігрового додатку інструментом Unity;
- Визначити недоліки існуючих засобів для покращення розробки ігрових додатків та на інші проблемні питання;
- Провести огляд IT-засобів, які можуть бути використані при розробці програмного забезпечення кваліфікаційної роботи бакалавра;
- Визначити об’єкт, предмет, мету та постановку завдань кваліфікаційної роботи бакалавра.

Інструменти та технології

- Платформа розробки – .NET
- Мова програмування – C#
- Фреймворки – Entity Framework Core, Unity Engine Framework
- Середовища розробки – Unity, Microsoft Visual Studio, Microsoft Visual Studio Code

Аналіз аналогів

Переваги

- Кроссплатформленість.

Недоліки

- Використання “честної” процедурної генерації або її відсутність;
- “Несправедливий” штучний інтелект;
- Відсутність гри+.

Мета, об’єкт та предмет роботи

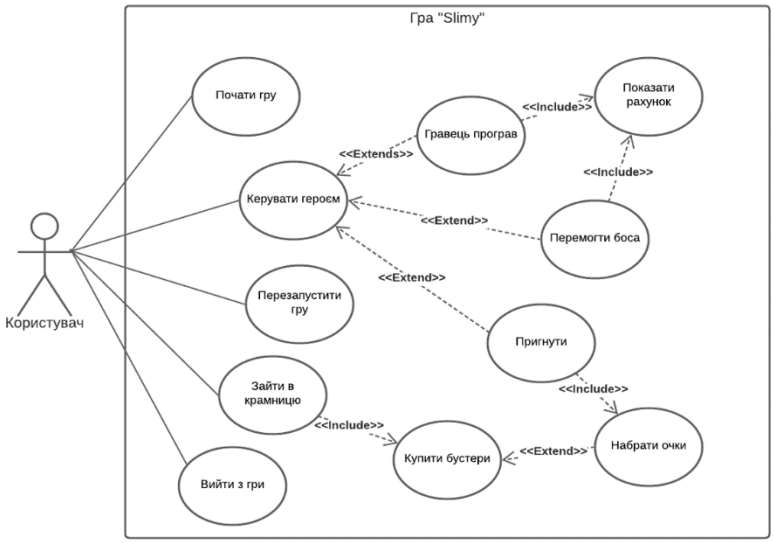
- ***Мета роботи*** – розробити 2D ігровий додаток Slimy з використанням рушія Unity в жанрі платформер з використанням випадкової генерації рівнів.
- ***Об'єкт дослідження*** – гра жанру платформер.
- ***Предмет дослідження*** – ігровий додаток з використанням випадкової генерації рівнів.

Постановка технічного завдання:

Майбутній програмний продукт повинен включати наступні ключові функції:

- можливість висліджувати висоту (ігровий рахунок гравця);
- можливість придбати бустери за реальні гроші, купуючи ігрові набори;
- можливість висліджувати появу ворога певним попереджувальним знаком і звуком;
- можливість збирати ігрову валюту (монетки) і висліджувати їх на екрані;
- можливість витрачати ігрову валюту в магазині;
- можливість використовувати бустери під час проходження гри;
- підвищення складності гри після наступного проходження.

Діаграма прецедентів



Приклад використання

Робота пройшла апробацію на:

- Науково-технічній конференції “Застосування програмного забезпечення в інфокомунікаційних технологіях” (сторінка 124);
- III Всеукраїнська Науково-практична конференція “Сучасні інтелектуальні інформаційні технології в науці та освіті” (подано на друк).

Висновки

- Проведено аналіз предметної галузі ігрових додатків
- Розглянуто програмні забезпечення, які можуть бути використані для аналізу та розробки власного ігрового додатку.
- Проведено аналіз засобів розробки програмного забезпечення
- Був розроблений 2D ігровий додаток, використовуючи рушій Unity.

Дякую за увагу