

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Відтворення на Python ігор "Super Mario"»

Виконав студент групи ЗППЗд-20-1

ГОБАН Василь Федорович

Керівник роботи:

ЧЕПЮК Ларіна Олексіївна

Консультант:

ЛОКТИКОВА Тамара Миколаївна

Рецензент:

МОРОЗОВ Андрій Васильович

ЖИТОМИР – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ГОБАН Василь Федорович**

Керівник роботи: **Чепюк Ларіна Олексіївна**

Тема роботи: **Відтворення на Python ігор "Super Mario",**

затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р., №74/с**

Вихідні дані для роботи: **технології розробки гри у жанрі 2D платформер на прикладі гри "Super Mario" з використанням мови Python.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Локтікова Т.М.	04.03.2024р.	04.03.2024р.
2	Локтікова Т.М.	01.04.2024р.	01.04.2024р.
3	Локтікова Т.М.	13.05.2024р.	13.05.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з гри "Super Mario" та пояснювальної записки, яка містить 65 сторінок, 19 рисунки, 15 інформаційних джерел.

Метою кваліфікаційної роботи є розробка 2D гри у жанрі платформер, схожої на "Super Mario", з використанням найефективнішого програмного забезпечення, зокрема мови програмування Python та бібліотеки Pygame.

Прототип гри, що розробляється, є цінним активом, який може бути використаний для різних цілей. Він може стати основою для подальшого розвитку та вдосконалення гри.

Також, прототип може бути використаний для демонстрації потенціалу та можливостей даної розробки в загальному користуванні.

Для розробки комп'ютерної гри, схожої на "Super Mario", знайдено та використано найефективніші програмні засоби, зокрема Python та бібліотеку Pygame.

Також було використано різноманітні двовимірні об'єкти з доступних джерел, готові матеріали та текстури, щоб відтворити елементи та атмосферу класичної гри "Super Mario".

КЛЮЧОВІ СЛОВА: ІГРОВИЙ ДВИГУН, КОМП'ЮТЕРНА ГРА, 2D ПЛАТФОРМЕР, PYGAME, PYTHON, SUPER MARIO.

ІПЗ.КР.Б-121-24-ІІЗ

					ІПЗ.КР.Б-121-24-ПЗ							
Змн.	Арк.	№ докум.	Підпис	Дата								
Розроб.		Гобан В.Ф.			Відтворення на Python ігор "Super Mario". Пояснювальна записка			Літ.	Арк.	Аркушів		
Керівник		Чепюк Л.О.								3	65	
Рецензент		Морозов А.В						Житомирська політехніка група ЗІПЗд-20-1				
Н. контр.		Тичина О.П.										
Зав.каф.		Вакалюк Т.А.										

ABSTRACT

The graduation thesis of the bachelor consists of the game "Super Mario" and an explanatory note that contains 65 pages, 19 drawings, 15 information sources.

The goal of the qualification work is to develop a 2D game in the platform genre, similar to "Super Mario", using the most efficient software, in particular the Python programming language and the Pygame library.

Research methods include logical method, analysis, comparative method, processing of literary sources and design.

A game prototype in development is a valuable asset that can be used for a variety of purposes. It can become the basis for further development and improvement of the game. Also, the prototype can be used to demonstrate the potential and capabilities of this development in general use.

To develop a computer game similar to "Super Mario", the most effective software tools, in particular Python and the Pygame library, were found and used.

A variety of off-the-shelf 2D objects, off-the-shelf materials and textures were also used to recreate the elements and atmosphere of the classic Super Mario game.

KEYWORDS: GAME ENGINE, COMPUTER GAME, 2D PLATFORMER, PYGAME, PYTHON, SUPER MARIO.

					ИПЗ.КР.Б-121-24-ИЗ	Арк.
						4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ВІДТВОРЕННЯ ГРИ “SUPER MARIO” НА PYTHON	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір та обґрунтування інструментальних засобів.....	16
Висновки до розділу 1	28
РОЗДІЛ 2. ПРОЄКТУВАННЯ ГРИ.....	31
2.1. Визначення варіантів використання та об’єктно-орієнтованої структури...	31
2.2. Проєктування та реліз алгоритмів роботи системи.....	33
2.3. Реалізація гри.....	35
Висновки до розділу 2	42
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ГРОЮ.....	43
3.1. Графічне оформлення гри.....	43
3.2. Структура інтейфейсу	47
Висновки до розділу 3	49
ВИСНОВКИ.....	50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	53
ДОДАТКИ	55

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

3D - тривимірне

2D - двовимірне

Геймплей - термін, яким називають особливості взаємодії людини з відеогрою

ПК - персональний комп'ютер

Мультиплеєр - багатокористувацька гра

					ІІЗ.КР.Б-121-24-ІІЗ	Арк.
						6

ВСТУП

Актуальність теми пов'язана з тим, що комп'ютерні ігри є однією з найпопулярніших форм розваг у сучасному світі, а також важливим напрямком у галузі інформаційних технологій. Розробка ігор вимагає поєднання знань у сфері програмування, дизайну, графіки та фізики. Однією з найвідоміших ігор в історії є "Super Mario", яка стала культовою завдяки своїй інноваційній механіці та захоплюючому ігровому процесу.

Мета даної кваліфікаційної роботи полягає у відтворенні гри "Super Mario" з використанням мови програмування Python та бібліотеки Pygame. Даний проект має на меті дослідити технології розробки 2D платформерів і створити функціональний прототип гри, який буде включати основні елементи оригінальної гри.

Використання Python та Pygame для розробки 2D ігор дозволяє швидко і ефективно створювати прототипи, забезпечуючи при цьому високу продуктивність і гнучкість розробки. Дана кваліфікаційна робота спрямована на дослідження цих технологій та їх застосування у розробці гри в жанрі платформер, схожої на "Super Mario".

Об'єктом дослідження є гра у жанрі 2D платформер, зокрема гра "Super Mario".

Предметом дослідження є технології розробки гри у жанрі 2D платформер на прикладі гри "Super Mario" з використанням мови програмування Python та бібліотеки Pygame.

Для досягнення поставленої мети необхідне виконання таких **завдань**:

1. Провести аналіз існуючих технологій розробки 2D платформерів.
2. Вивчити особливості мови програмування Python та бібліотеки Pygame.
3. Розробити прототип гри у жанрі 2D платформер на прикладі "Super Mario".
4. Реалізувати основні елементи гри: персонаж, вороги, платформи, фізика руху, колізії.

5. Провести тестування та оптимізацію розробленої гри.

6. Оцінити можливості подальшого розвитку та вдосконалення створеного прототипу.

					ПЗ.КР.Б-121-24-ПЗ	Арк.
						8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ВІДТВОРЕННЯ ГРИ “SUPER MARIO” НА PYTHON

1.1. Постановка задачі

Традиційно комп'ютерна гра є програмним забезпеченням, призначеним для забезпечення ігрового процесу. Під час гри використовується спеціальна програма, що створює імітацію взаємодії між ігровим персонажем та користувачем (або групою користувачів) у віртуальному просторі за певним алгоритмом. Для здійснення гри використовуються різноманітні пристрої введення, такі як комп'ютерна миша, клавіатура, камера, джойстик тощо.

Комп'ютерна гра, так само як і традиційна гра, є симуляцією реальності, де гравець свідомо взаємодіє зі світом, який є віртуальним. Проте, в комп'ютерних іграх існують обмеження в просторі, часі та можливостях. Це відрізняє їх від традиційних ігор. Візуальні ефекти комп'ютерних ігор створюються розробниками і є результатом їхньої творчої діяльності, а не власної уяви гравця. У комп'ютерних іграх також використовуються правила, які впроваджені у їхній алгоритм.

У сучасному світі комп'ютерні ігри стали невід'ємною частиною повсякденного життя багатьох людей. Саме тому вони часто створюються на основі зовнішніх джерел, наприклад фільмів, книг, серіалів або навіть міфів та історичних подій. Однак з часом почали з'являтися випадки, коли вже відомі ігрові серії випускають додаткові матеріали, які розширюють уявний світ гри.

Спеціально розроблені ігри можуть бути використані як навчальний матеріал або для наукових досліджень, але такі ігри випускаються досить рідко. Проте, зростає попит на розробку якісних і захоплюючих ігрових продуктів, які відповідають сучасним вимогам користувачів.

Ігрова індустрія має настільки вагомий вплив на сучасне суспільство, що деякі ігри вже давно стали об'єктом аматорських і професійних змагань, що

призвело до створення цілком нового виду спорту - кіберспорту. Вплив комп'ютерних ігор на сучасне суспільство має значний масштаб, що спричиняє розповсюдження індустрії ігор у сфері прикладного програмного забезпечення.

Наприклад, деякі європейські навчальні заклади використовують відомі ігри у процесі навчання, а для тренування солдатів створюються спеціальні симулятори. Багато країн визнали кіберспорт як офіційний вид спорту, а в 2011 році уряд США надав комп'ютерним іграм статус окремого виду мистецтва, нарівні з театром та кіно.

Комп'ютерні ігри стали неодмінною частиною сучасного життя суспільства, і за останні декілька років їх використання значно розширилось. Тепер ігри використовуються людством не тільки задля розваг та відпочинку, як було на початку, але й для навчання та наукових досліджень.

З огляду на значущість ігрової індустрії та її вплив на сучасне суспільство, актуальним є дослідження та розробка гри в жанрі 2D платформер, яка ґрунтується на класичній грі "Super Mario". Метою даної роботи є створення функціональної версії гри "Super Mario" з використанням мови програмування Python та бібліотеки Pygame.

Основні завдання, які необхідно вирішити для досягнення цієї мети, включають:

1. Аналіз ігрових механік "Super Mario": вивчення основних механік гри, таких як рух персонажа, стрибки, взаємодія з об'єктами, збирання монет, битви з ворогами та проходження рівнів. Необхідно детально дослідити ці аспекти, щоб забезпечити точне відтворення ігрового процесу.

2. Вибір технологій для розробки гри: обґрунтування вибору мови програмування Python та бібліотеки Pygame для створення гри. Оцінка можливостей та обмежень цих технологій, а також аналіз їх придатності для реалізації проекту.

3. Проектування архітектури гри: розробка структури гри, визначення основних компонентів та їх взаємодії. Це включає створення ігрового циклу,

обробку вводу користувача, реалізацію фізики та колізій, а також управління станом гри.

4. Реалізація графіки та анімацій: створення або використання готових двовимірних спрайтів для відтворення персонажів, ворогів, елементів рівня та інших об'єктів. Забезпечення плавності анімацій та відповідності графічних елементів стилю "Super Mario".

5. Розробка звукового супроводу: вибір та інтеграція звукових ефектів і музики, що відповідають атмосфері гри. Це включає звуки стрибків, збору монет, битв з ворогами та інші звукові ефекти.

6. Тестування та відлагодження гри: проведення тестування на різних етапах розробки для виявлення та виправлення помилок. Оптимізація гри для забезпечення стабільної роботи та високої продуктивності.

7. Документування процесу розробки: опис всіх етапів роботи, використаних технологій, архітектурних рішень та отриманих результатів. Це необхідно для забезпечення можливості подальшого розвитку та вдосконалення гри.

Результатом виконання поставлених завдань буде створення функціонального прототипу гри в жанрі 2D платформер, який відтворює основні елементи "Super Mario". Це дозволить отримати практичні навички розробки ігор на Python та Pygame, а також створить базу для подальших проектів у цій галузі.

1.2. Аналіз аналогів програмного продукту

Аналіз та дослідження аналогічних додатків у жанрі платформер є ключовим етапом при створенні власної гри. Це дозволяє виявити, які ігри вже існують на ринку, оцінити їхні особливості та успіхи, а також знайти можливі недоліки, які можна обійти у власному проекті.

Платформери є одним з найпопулярніших жанрів відеоігор, зокрема завдяки таким класичним іграм як Super Mario Bros., Sonic the Hedgehog та

Crash Bandicoot. Цей жанр залишається актуальним, і на ПК можна знайти безліч варіантів платформерів на будь-який смак. Під час виконання кваліфікаційної роботи було розглянуто різні проекти та ігри, серед яких виділяються кілька найуспішніших.

Першою для аналізу було обрано гру Dead Cells (рис. 1.1). Це платформер-"рогалик" з елементами "метроїдванії", розроблений французькою студією Motion Twin. Гра доступна для операційних систем Windows, MacOS, Linux, а також для консолей Nintendo Switch, PlayStation 4 і Xbox One. У Dead Cells персонаж має лише одне життя — після смерті гравець починає гру заново, зберігаючи знання про локації, що дозволяє швидше знаходити необхідні проходи та секрети.

Гравець досліджує похмурий фентезійний світ без безпечних зон та точок збереження. Головний герой пробирається через підземелля, бореться з ворогами та збирає предмети. Він може використовувати різну зброю, яку знаходить або отримує під час гри: мечі, ножі, гранати, капкани та турелі.



Рисунок 1.1 - Знімок екрана з гри Dead Cells

Властивості зброї та предметів у грі є випадковими і змінюються в залежності від рівня складності.

Кожен рівень і підземелля генеруються випадковим чином. Гра автоматично створює складні лабіринти, використовуючи раніше підготовлені елементи, та випадковим чином розміщує предмети та ворогів. Гравець має багато можливостей помирати, але в цьому процесі він набуває досвіду і вчиться на своїх помилках.

Dead Cells отримала високі оцінки критиків, завдяки своїй приємній візуальній графіці, складному геймплею та високій загальній якості гри.

Наступною грою для аналізу була обрана Hollow Knight. Це гра в жанрі метроїдванія, розроблена інді-студією Team Cherry. Гра була випущена 24 лютого 2017 року для операційної системи Windows (рис. 1.2).



Рисунок 1.2 - Знімок екрана з гри Hollow Knight

Основна механіка гри Hollow Knight полягає у дослідженні великого інтегрованого світу, де гравець стикається з платформами, секретами та ворогами, з якими потрібно боротися. Гравець досліджує світ, використовуючи спеціальні поліпшення для переміщення та бойові навички, такі як відбивання ворожих атак. У стилі багатьох метроїдваній, деякі області стають доступними лише після перемоги над босами та отримання відповідних покращень. Тому гравцям доводиться повертатися до вже

відвіданих зон для пошуку секретів або продовження сюжету. Деякі зони змінюються в процесі гри, що додає елемент несподіванки. Кожна зона пов'язана з іншими через кілька переходів, що дозволяє проходити гру різними шляхами. Карти кожної зони стають доступними тільки після знаходження картографа у цій зоні та придбання карти за місцеву валюту, або купівлі її в магазині карт за вищою ціною, після чого гравець заповнює карту, відвідуючи різні локації.

Бойова система Hollow Knight заснована на використанні головним героєм цвяха — місцевого аналога меча. Атакувати можна в чотири напрямки: вгору, вниз, праворуч, ліворуч. Цвях можна покращити кілька разів у коваля, а також вивчити спеціальні прийоми володіння цвяхом у майстрів бою. Удари по ворогам накопичують спеціальний ресурс душу (Soul) у спеціальну посудину, який можна використовувати для лікування або спеціальних прийомів.

Якщо гравець гине, він втрачає всю накопичену валюту і третину основної посудини душі, і на місці смерті з'являється ворожа тінь (Shade). Якщо вбити тінь, то гравець повертає всю накопичену валюту та відновлює головну посудину душі. Hollow Knight отримала визнання за свою вражаючу графіку, зручне та чуйне керування, а також складний і захоплюючий геймплей.

Наступною грою для аналізу була обрана Mark of the Ninja (рис. 1.3). Це стелс-екшен, розроблений канадською студією Klei Entertainment для Xbox 360 і PC, випущений у сервісах Xbox Live Arcade і Steam.

У Mark of the Ninja гравець управляє ніндзя, який досліджує різні локації та виконує завдання, що вимагають обережності та вправності. Головний герой може ховатися у вентиляційних отворах, каналізаційних люках та за предметами інтер'єру. Ніндзя використовує різноманітні засоби для виживання та перемоги, такі як меч для безшумних вбивств, бамбукові дротики для відволікання та знищення джерел світла, а також інше спорядження. Гра пропонує різні підходи до проходження рівнів, включаючи

уникнення супротивників, ховання, використання відволікань та вирішення головоломок. У процесі гри гравець може купувати та відкривати додаткове спорядження для атакуючих та відволікаючих дій.

У Mark of the Ninja кожна дія, така як знищення або обхід супротивника, ховання тіла або вимикання пасток, приносить очки, які можна використовувати для відкриття нових прийомів, покращення спорядження та виконання бонусних завдань. Гравець також може збирати артефакти та вирішувати головоломки, що приносять додаткові бонуси.

Mark of the Ninja вирізняється стильним графічним оформленням, яке допомагає створити відповідну атмосферу для всієї історії гри. Вона виділяється серед інших двовимірних ігор свого жанру та надає гравцям велику свободу вибору стратегій та тактик, дозволяючи проходити рівні різними способами та розвивати власний стиль гри.



Рисунок 1.3 - Знімок екрана з гри Mark of the Ninja

Жанр платформерів відомий своїм елементом колекціонування предметів, які є необхідними для проходження рівнів. Гравець управляє персонажем, який переміщується між платформами різних розмірів, висот та масштабів, що залежить від типу і складності гри. Більшість платформерів

мають фантастичні елементи та мальовану графіку. Головними героями таких ігор можуть бути як звичайні люди у незвичайних обставинах, так і міфічні істоти (наприклад, гноми, дракони) або антропоморфні тварини.

1.3. Вибір та обґрунтування інструментальних засобів

Розробка програмного забезпечення зазвичай вимагає вибору мови програмування, яка найкраще відповідає вимогам проекту. Однією з найпопулярніших мов програмування є Python, створена Гвідо ван Россумом. Python є об'єктно-орієнтованою мовою програмування з великим набором функцій та можливостей, що дозволяє розробникам створювати різноманітне програмне забезпечення.

Python належить до сімейства мов високого рівня і відзначається власними особливостями. Її об'єктно-орієнтований підхід дозволяє моделювати реальні об'єкти та їх взаємодії. Прості та зрозумілі синтаксичні конструкції спрощують процес розробки та роблять код легким для розуміння. Автоматичне управління пам'яттю значно полегшує роботу з ресурсами та зменшує ймовірність помилок.

Однією з найпривабливіших особливостей Python є її простий та зрозумілий синтаксис. Це робить Python ідеальною мовою для новачків у програмуванні, а також допомагає зменшити кількість помилок у коді. Крім того, Python підтримує автоматичне управління пам'яттю, що сприяє ефективному використанню ресурсів та знижує ризик виникнення витоків пам'яті.

Python підтримує кросплатформену розробку завдяки своїй сумісності з різними операційними системами, такими як Windows, macOS та Linux. Це означає, що написаний код на Python може бути виконаний на різних платформах без значних змін у вихідному коді. Це дозволяє створювати кросплатформені додатки, які можуть працювати на різних операційних системах з мінімальними зусиллями. Python також може використовуватися

для розробки мобільних додатків завдяки фреймворкам, таким як Kivu та BeeWare.

Ігрова розробка є ще однією галуззю, де Python може виявити себе дуже ефективно. Ігровий двигун Godot, який стає все більш популярним у галузі, підтримує Python (через модифіковану версію під назвою GDScript). Це дозволяє розробникам створювати складні та захоплюючі ігри, використовуючи потужні функції та бібліотеки, доступні у Python.

Окрім стандартних функцій та бібліотек, існує також велика спільнота розробників Python, яка активно працює над розширенням мови та створенням власних бібліотек і фреймворків. Це сприяє активному обміну знаннями та досвідом, а також надає доступ до великої кількості готових рішень та кодових зразків, що можуть значно полегшити розробку програмного забезпечення.

Python вирізняється високим рівнем абстракції, зрозумілим синтаксисом та широкими можливостями, що робить його потужним інструментом для розробки програмного забезпечення у різних галузях, включаючи ігрову індустрію. Використання Python дозволяє розробникам ефективно створювати високоякісні ігри та програми з меншими зусиллями і часом, що сприяє прискоренню процесу розробки та досягненню бажаних результатів.

Ігрові движки є потужними інструментами для створення ігор, які дозволяють розробникам ефективно використовувати свої знання та ресурси для створення як 2D, так і 3D ігор. Вони надають набір інструментів, які спрощують такі завдання, як програмування геймплею, робота з графікою, фізикою, звуком, штучним інтелектом та іншими аспектами гри. Ці движки забезпечують основну архітектуру, на яку розробники можуть спиратися для створення своїх ігор.

Ігрові движки надають широкий спектр функцій та інструментів, таких як введення, рендеринг, скрипти, виявлення зіткнень, штучний інтелект та багато іншого. Вони дозволяють розробникам використовувати готові компоненти, замість розробки їх з нуля, що дає можливість зосередитися на унікальних аспектах гри, таких як дизайн персонажів, текстури, фізика,

геймплей та інші елементи. Використання ігрових движків допомагає прискорити процес розробки ігор, зменшити витрати на розробку та забезпечити високу якість готового продукту.

При виборі ігрового движка розробники повинні враховувати свої потреби та цілі проекту, а також оцінювати доступні рішення та їх можливості. Кожен ігровий движок має свої особливості, переваги та обмеження, тому важливо знайти той, який найкраще відповідає вимогам проекту.

Існує велика кількість ігрових движків, які відрізняються за багатьма параметрами, включаючи наявність безкоштовного чи платного доступу. Вони включають прості бібліотеки для популярних мов програмування, а також повноцінні редактори з розширеним функціоналом. Для аналізу були обрані одні з найпопулярніших ігрових движків: Unity, Unreal Engine та CryEngine.

Unity

Unity – це ігровий движок та інтегроване середовище розробки, що використовується для створення ігор, віртуальної реальності (VR), доповненої реальності (AR) та інтерактивних додатків. Він надає широкий спектр можливостей та інструментів для розробників, що дозволяє їм створювати професійні ігри з різноманітними геймплейними механіками та візуальними ефектами.

Основна мета Unity полягає в наданні розробникам потужного та надійного набору інструментів, що спрощує використання движку для розробників будь-якого рівня кваліфікації, включаючи початківців. Компанія Unity Technologies також розширила своє охоплення в інших галузях, зосередившись на розробці 3D у реальному часі, що зробило Unity одним із найпотужніших ігрових движків.

Unity дозволяє розробляти ігри для різних платформ, включаючи ПК, мобільні пристрої, ігрові консолі та віртуальну реальність. Це робить Unity універсальним інструментом для досягнення широкої аудиторії та максимізації потенційної бази користувачів.

Однією з ключових особливостей Unity є його гнучкість та

розширюваність. Він має велику кількість доступних плагінів та розширень, які дозволяють розробникам розширити функціональність двигуна та використовувати спеціалізовані інструменти для своїх проектів. Інтерфейс Unity є інтуїтивно зрозумілим та добре організованим, що полегшує роботу розробників та зменшує час, необхідний для освоєння інструментів та процесу розробки. Він також має потужні функції для роботи з графікою, анімацією, фізикою, штучним інтелектом та звуком, що дозволяє створювати вражаючі візуальні та аудіо елементи для ігор (рис. 1.4).

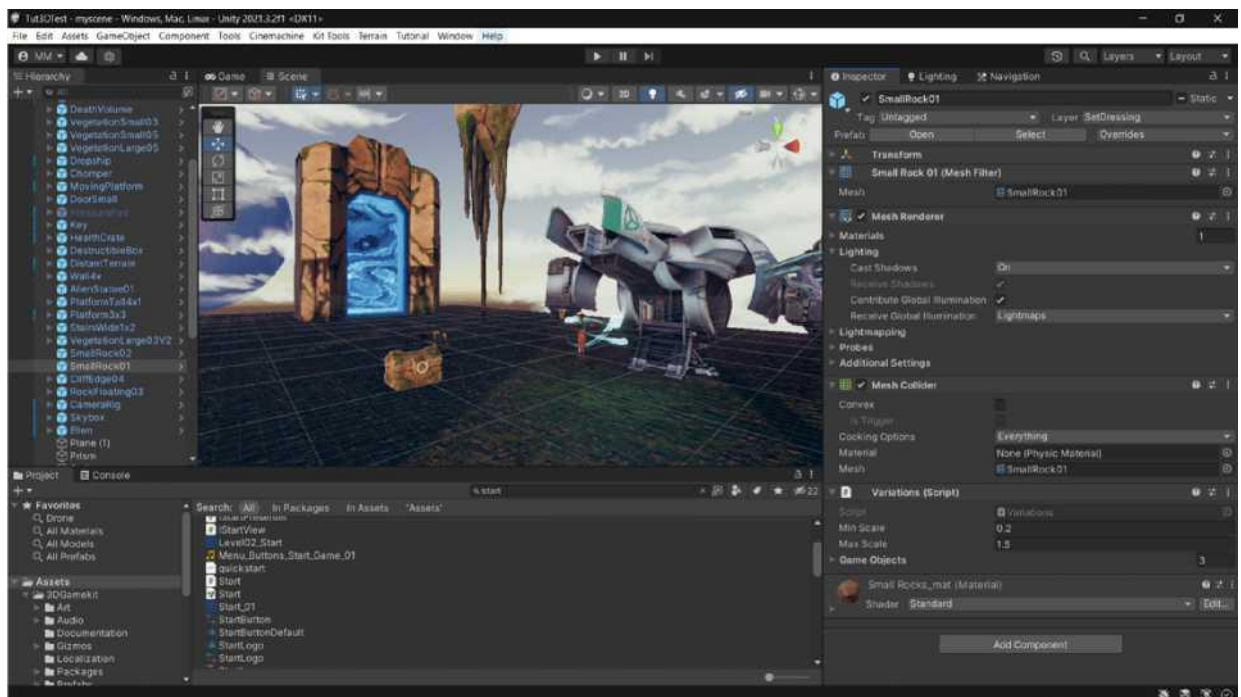


Рисунок 1.4 - Інтерфейс ігрового движка Unity

Unity має кілька важливих переваг:

1. Кросплатформеність. Unity дозволяє розробляти ігри для різних платформ, включаючи ПК, мобільні пристрої та консолі.
2. Розширюваність. Движок підтримує велику кількість плагінів та розширень, що дозволяє додавати спеціалізовані інструменти та розширювати функціонал.
3. Інтуїтивний інтерфейс. Unity пропонує добре організований та зручний інтерфейс, що значно спрощує процес розробки.
4. Потужність. Unity забезпечує потужні функції для роботи з графікою,

анімацією, фізикою, штучним інтелектом та звуком.

Проте, Unity також має свої недоліки:

1. Висока вартість. Деякі функції та додаткові плагіни Unity можуть потребувати значних фінансових вкладень.

2. Обмеження продуктивності. В окремих випадках Unity може бути менш продуктивним у порівнянні з іншими ігровими движками при обробці великих обсягів даних або складних графічних ефектів.

3. Залежність від розробників. Оновлення та розвиток Unity можуть вимагати від розробників постійного вивчення нових версій та функцій, що додає додаткові складнощі у підтримці проєктів.

Unreal Engine

Unreal Engine, створений у 1996 році, був спочатку розроблений для ігор від першої особи. З часом його застосування значно розширилося, охопивши різні жанри та платформи.

Unreal Engine надає розробникам великий набір інструментів для створення високоякісних ігор з вражаючою графікою. Він підтримує всі основні платформи, включаючи ПК, консолі та мобільні пристрої, що дозволяє розробникам охоплювати широку аудиторію. Однією з головних особливостей Unreal Engine є його потужний редактор рівнів, який надає безліч можливостей для налаштування та створення складних ігрових світів.

Серед інших важливих переваг Unreal Engine слід зазначити його потужний графічний рушій, що забезпечує фотореалістичну графіку та підтримку сучасних технологій, таких як трасування променів. Крім того, Unreal Engine має вбудовані інструменти для роботи з фізикою, анімацією та звуком, що дозволяє створювати реалістичні та захоплюючі ігри.

Недоліки Unreal Engine включають високу складність освоєння для новачків через велике різноманіття функцій та налаштувань. Крім того, використання рушія може вимагати значних ресурсів від комп'ютера розробника, що робить його менш доступним для невеликих команд або індивідуальних розробників (рис. 1.5).

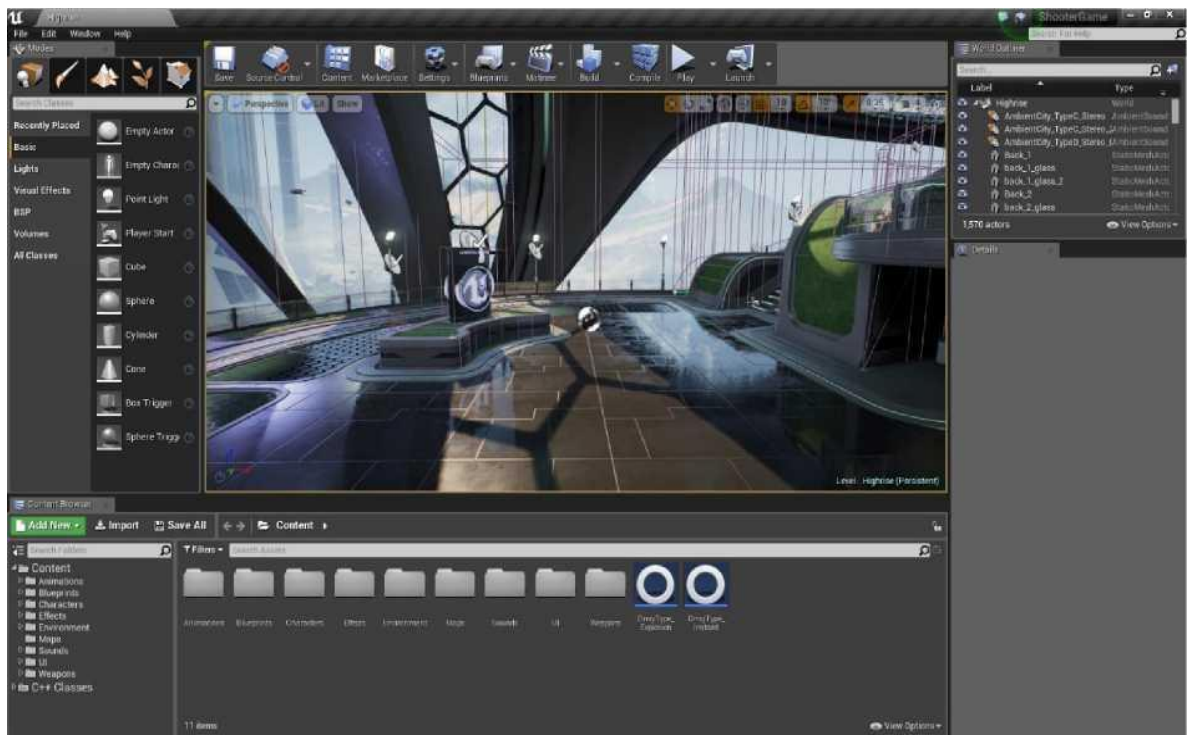


Рисунок 1.5 - Інтерфейс середовища розробки Unreal Engine

Це стало можливим завдяки простоті розробки та відсутності необхідності вивчати складну мову програмування, таку як C++. Спочатку Unreal Engine був доступний лише на платній основі. Проте у 2015 році розробники зробили крок назустріч спільноті, випустивши безкоштовну пробну версію. Цей крок значно підвищив популярність рушія, оскільки розробникам відкрилася можливість створювати складні ігри, що раніше було неможливо з іншими безкоштовними движками.

Розроблений на мові програмування C++, рушій Unreal Engine дозволяє створювати ігри для різних операційних систем і платформ, таких як Microsoft Windows, Linux, Mac OS, консолей Xbox, Xbox 360, PlayStation 2, PlayStation Portable, PlayStation 3, Dreamcast і Nintendo GameCube. У грудні 2009 року Марк Рейн продемонстрував роботу рушія Unreal Engine 3 на iPod Touch і iPhone 3GS, а в березні 2010 року була показана його робота на комунікаторі Palm Pre, що працює на мобільній платформі webOS.

Особливості створення ігор на Unreal Engine:

- модульна система компонентів. Unreal Engine використовує модульну архітектуру, яка дозволяє переносити ігри та інші розробки з однієї платформи на іншу;

- розробка на Python. Хоча Unreal Engine спочатку розроблений на C++, існують можливості інтеграції з Python, що дозволяє створювати ігри для платформ, таких як Windows, Mac OS і Linux, використовуючи Python;

- підтримка мобільних платформ. Останнім часом розробники Unreal Engine активно працюють над створенням кейсів для платформи Android, що дозволяє створювати ігри для мобільних пристроїв.

Переваги Unreal Engine:

- простота використання. Рушій легко освоїти навіть розробнику без досвіду програмування. Навігація та налаштування подібні до інших програмних засобів;

- широкий спектр функцій. Unreal Engine пропонує великий набір функцій, що дозволяє розробникам створювати свої продукти без обмежень;

- універсальність та доступність. Unreal Engine використовується як досвідченими розробниками, так і новачками в галузі створення ігор;

- активна спільнота користувачів. Спільнота постійно створює навчальні матеріали, ділиться порадами та допомагає вирішувати проблеми, пов'язані з використанням рушія;

- вбудована система візуального скриптингу. Це дозволяє легко створювати ігрову логіку без написання коду;

- умовно безкоштовний рушій. Розробники не сплачують за використання Unreal Engine, якщо їх гра не приносить значного доходу. У разі, якщо гра заробляє понад \$1,000,000, відсоток від доходу складає 5%, що є вигідними умовами;

- мультиплатформеність. Unreal Engine підтримує розробку ігор для різних платформ, включаючи ПК, консолі, мобільні пристрої та віртуальну реальність, що дозволяє охопити широку аудиторію.

Недоліки Unreal Engine:

- високі вимоги до апаратного забезпечення. Для ефективної роботи Unreal Engine потрібне потужне обладнання. Для розробки та запуску ігор необхідно мати комп'ютер з достатньою кількістю оперативної пам'яті та потужним графічним процесором;

- великий обсяг проекту. Проекти, розроблені на Unreal Engine, можуть мати значний обсяг через використання різноманітних компонентів та ресурсів, таких як моделі, текстури, звукові ефекти тощо. Це може збільшити обсяг виконуваного файлу гри та вимагати більшої потужності обробки ЦП та обсягу пам'яті для ефективного відтворення гри.

CryENGINE

CryENGINE є ігровим рушієм, розробленим німецькою компанією Crytek, і який вперше був використаний у відеогрі Far CryENGINE (рис. 1.6).

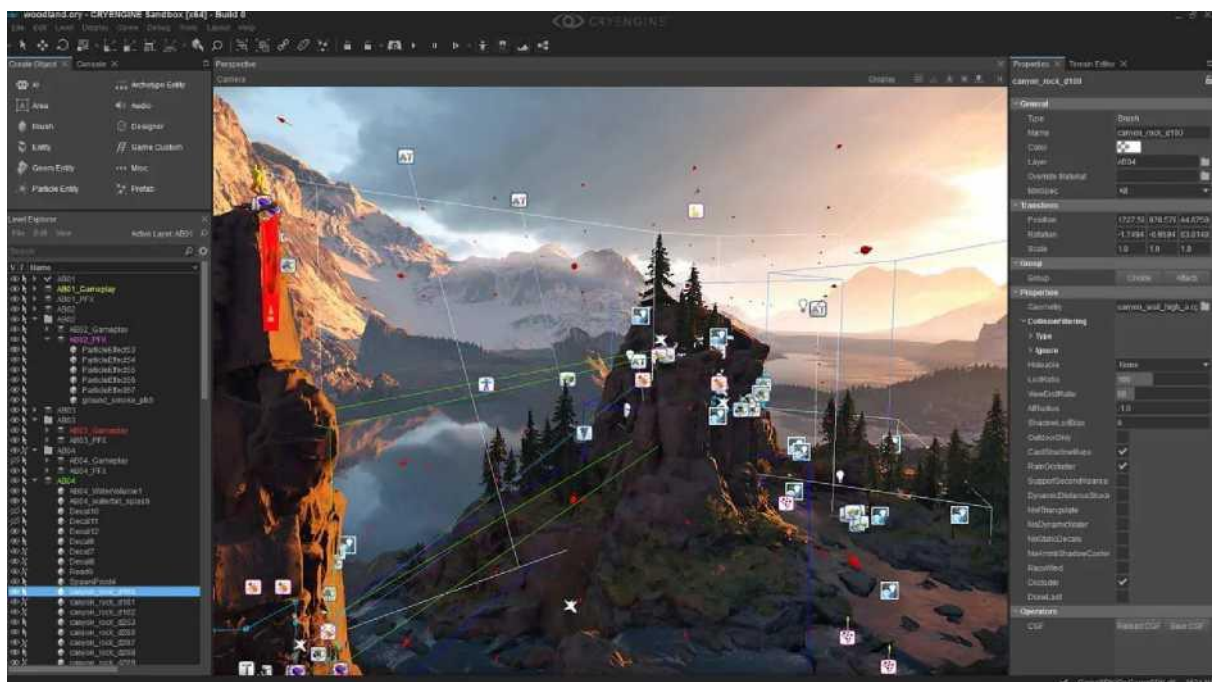


Рисунок 1.6 - Інтерфейс ігрового движка CryENGINE

Цей рушій відомий своєю потужною графікою та широкими можливостями, які надають розробникам ігор велику свободу творчості. Він має передові редактори для рівнів, анімації, штучного інтелекту та фізики, що дозволяє реалізувати складні механіки та інтерактивні елементи в іграх.

CryENGINE має великий функціонал і можливості, що дозволяють розробникам використовувати передову 3D-графіку та масштабовані

обчислення. Цей рушій включає значний набір вбудованих інструментів, що дозволяють створювати ігри без необхідності використання стороннього програмного забезпечення. Незважаючи на те, що жодна гра не змогла повністю використати всі можливості CryENGINE, він залишається ідеальним інструментом для створення ігор нового покоління. Вбудована система анімації включає параметричну скелетну анімацію, процедурне деформування руху та систему індивідуальних персонажів.

Основні особливості CryENGINE:

- можливість створення онлайн-ігор;
- підтримка платформ для ПК, Xbox і PlayStation;
- використання передових графічних технологій;
- наявність зручного редактора рівнів;
- можливості оптимізації гри під GPU-рендеринг;
- імпорт моделей і текстур з графічних редакторів Maya і 3ds Max;
- програмування місій у вільному форматі;
- зручний режим тестування ігрового процесу і меню ігор;
- створення великих закритих і відкритих локацій;
- використання передового штучного інтелекту;
- наявність всіх необхідних інструментів для повноцінної розробки без використання сторонніх програм;
- інтерфейс рушія реалізований англійською мовою.

Переваги CryENGINE:

- графічна якість. CryENGINE відомий своєю вражаючою графікою та реалістичними візуальними ефектами. Він забезпечує високу деталізацію, реалістичне освітлення і фотореалістичні текстури, що дозволяє створювати неймовірно реалістичні ігрові світи;

- фізична модель. CryENGINE має потужну фізичну модель, яка дозволяє реалістично відтворювати рухи об'єктів, взаємодію об'єктів з навколишнім середовищем та передавати різні фізичні ефекти;

- інструменти розробки. CryENGINE надає розширений набір інструментів для розробки ігрових проектів. Він має зручний візуальний редактор, що дозволяє швидше створювати та налаштовувати ігрові об'єкти, механіки та інтерактивні елементи.

Недоліки CryENGINE:

- високі вимоги до апаратного забезпечення. CryENGINE потребує потужного обладнання для ефективної роботи. Це може обмежити доступність розробки для користувачів з менш потужними системами;

- складність в освоєнні. CryENGINE має досить складну структуру та інтерфейс, що може вимагати значного часу та зусиль для освоєння. Розробникам необхідно мати досвід у створенні ігор або витратити час на вивчення документації та посібників;

- ліцензійні витрати. Використання CryENGINE пов'язане з ліцензійними витратами, особливо для комерційних проектів. Це може стати обмеженням для незалежних розробників або студій з обмеженими фінансовими ресурсами.

Програмні засоби реалізації

Для реалізації алгоритмів штучного інтелекту популярним вибором розробників є мова Python, для якої на сьогоднішній день написано багато бібліотек для аналізу даних та створення агентів штучного інтелекту. Оскільки велика частина нашого проекту припадає на створення середовища для тестування алгоритмів штучного інтелекту, який буде сумісним із різними агентами, Python є зручним варіантом [6]. Оскільки розроблюване середовище є грою і ми обрали як мову програмування Python, логічним є обрати у якості інструмента для розробки комп'ютерних ігор відомий серед Python-розробників набір модулів Pygame [7].

Pygame – це бібліотека Python, призначена для написання відеоігор. Pygame додає функціональність над бібліотекою SDL. SDL це кросплатформна бібліотека для розробки програмного забезпечення, призначена для забезпечення рівня абстракції апаратного забезпечення для

комп'ютерних мультимедійних апаратних компонентів. Розробники програмного забезпечення можуть використовувати її для написання високопродуктивних комп'ютерних ігор та інших мультимедійних програм, які можуть працювати на багатьох операційних системах, таких як Android, iOS, Linux, MacOS і Windows. Це дозволяє створювати повнофункціональні ігри та мультимедійні програми мовою Python. До його переваг можна додати наступні.

1) Безкоштовність. Pygame був випущений під ліцензією LGPL, безкоштовною загальною громадською ліцензією на програмне забезпечення обмеженого використання, що дозволяє створювати ігри з відкритим вихідним кодом, безкоштовне програмне забезпечення та комерційні продукти.

2) Для основних функцій використовується оптимізований код мовами C і Assembly. Код на мові C часто в 10-20 разів швидший за код мови Python, а код на мові Асемблер може бути в 100 і більше разів швидшим за код на Python.

3) Портативність. Pygame поставляється з бінарними інсталяторами і деінсталяторами для Windows або MacOSX. Для встановлення Pygame не потрібні інструменти установки.

4) Є сумісним з багатьма операційними системами. Pygame підтримує такі операційні системи як Linux (поставляється з більшістю основних дистрибутивів linux), Windows (95, 98, ME, 2000, XP, Vista, 64-бітний Windows), Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX та QNX. Код містить підтримку AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS та OS/2, хоча вони не зазначені, як такі, що підтримуються офіційно. Його можна використовувати на кишенькових пристроях та ігрових консолях.

5) Легке керування основним циклом. Рішення, які саме функції викликати, приймає сам розробник, а не бібліотека. Це надає більше контролю за використанням інших бібліотек.

6) Для використання всіх функцій не потрібний графічний інтерфейс користувача. Можна використовувати Pygame з командного рядка, якщо потрібно використати його для обробки зображень, отримання введення джойстика або відтворення звуків.

7) Невеликий обсяг коду. Головний код Pygame залишається простим, а додаткові речі, такі як бібліотеки GUI та ефекти, розробляються окремо поза Pygame.

8) Модульність. Багато кореневих модулів pygame можна ініціалізувати та використовувати окремо. Для розробки коду ми використали інтегроване середовище розробки Visual Studio Code, який також називають VS Code:

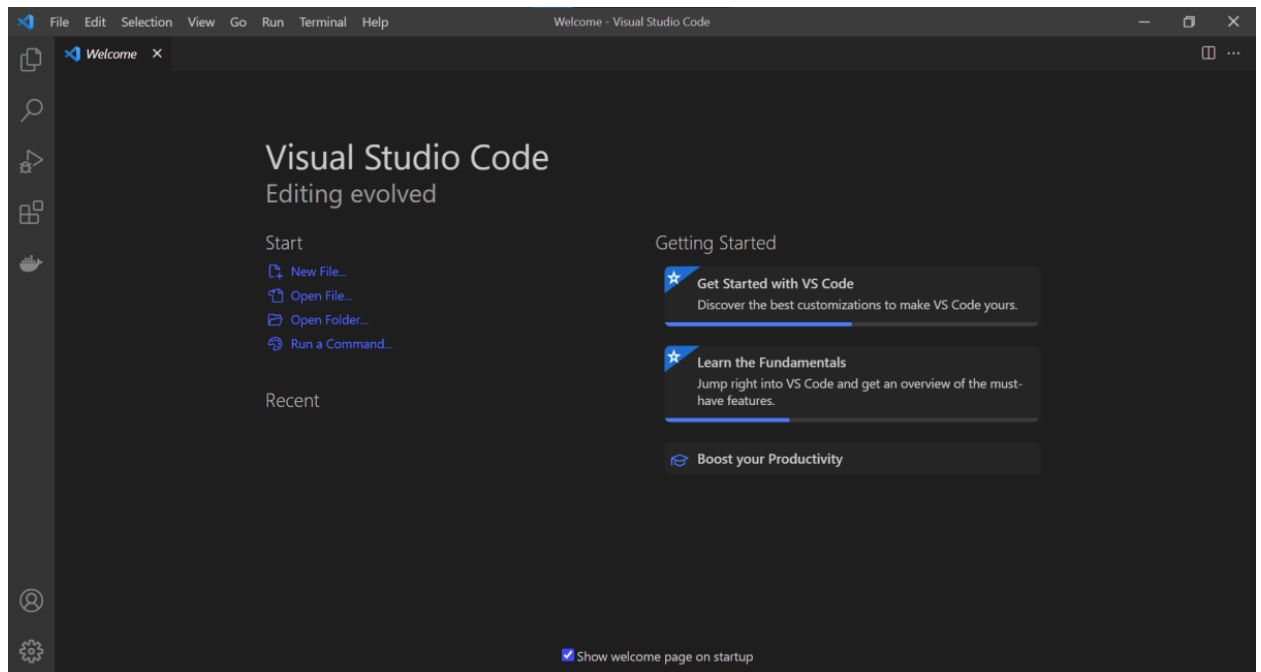


Рисунок 1.7 - Стартове вікно Visual Studio Code

Це редактор вихідного коду, створений компанією Microsoft для Windows, Linux і macOS [8]. Функції включають підтримку налагодження, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та вбудований Git. Користувачі можуть змінювати тему оформлення, поєднання клавіш, уподобання та встановлювати розширення, що додають додаткову функціональність. В опитуванні розробників Stack Overflow 2021 Visual Studio Code був названий найпопулярнішим інструментом розробки програмного забезпечення, 70% з 82 000 респондентів

повідомили, що використовують саме його [9]. Для покращення розробки на мові Python нам потрібно додати у Visual Studio Code пакет для розробки на Python:

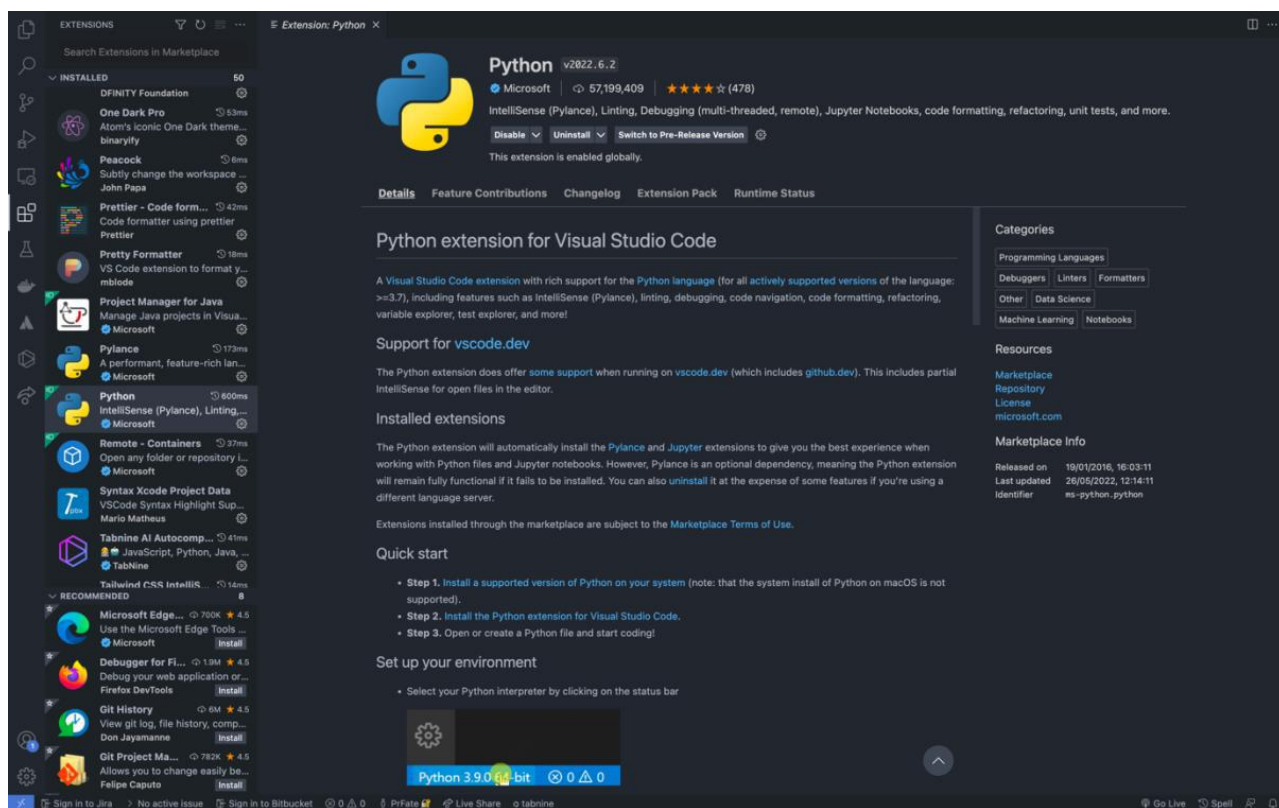


Рисунок 1.8 -Пакет для розробки на Python для Visual Studio Code

Додатково для розробки ігри нам потрібен інструмент для створення візуальних моделей предметів, а також спосіб створювати блок-схеми для візуалізації алгоритмів. Усі ці можливості надає редактор векторної графіки Figma.

Висновки до розділу 1

У першому розділі проведено детальний аналіз предметної області, пов'язаної з комп'ютерними іграми. Дослідження поняття комп'ютерних ігор показало їх значення та важливість у сучасному світі. Класифікація комп'ютерних ігор дозволила розподілити їх на різні категорії залежно від їх основних характеристик та жанрів.

Розробка комп'ютерних ігор включає багатоетапний процес, кожен етап

якого має свої специфічні завдання та вимоги. Від ідеї та концепції до релізу та підтримки – всі етапи важливі для створення якісного та захоплюючого ігрового продукту. Використання Python та бібліотеки Pygame дозволяє ефективно реалізувати основні етапи розробки 2D платформеру, що сприяє створенню якісної гри, схожої на "Super Mario".

Огляд та аналіз додатків-аналогів були проведені з метою визначити наявність схожих програмних рішень на ринку, а також для виявлення їх переваг та недоліків. Це дозволило отримати уявлення про існуючі тенденції та інновації у галузі комп'ютерних ігор, а також зробити висновки про потенційні можливості для подальшого дослідження та розробки. Огляд показав, що в першу чергу треба звернути увагу на зручність інтерфейсу та дизайну, тому що платформери не є іграми зі складним сюжетом, характеризуються простотою геймплею і основний акцент в цьому жанрі ігор робиться на візуальну складову, тобто зрозумілий інтерфейс та гарна картинка.

Також розглянуто засоби програмної реалізації ігрового застосунку. Було обрано мову програмування Python, яка є популярною та широко використовується для розробки різноманітних програмних проєктів, включаючи відеоігри. Вона має потужні можливості та зручну синтаксичну структуру, що дозволяє ефективно реалізовувати функціонал ігрових проєктів.

Був проведений аналіз існуючих ігрових фреймворків для Python, зокрема Pygame. Виявлено, що кожен з них має свої особливості, переваги та недоліки. Цей аналіз дозволив обґрунтувати вибір Pygame як ігрового фреймворку для реалізації проєкту.

Досліджено та проаналізовано середовище розробки Python, яке є одним з найпопулярніших інструментів для розробки програмного забезпечення на. Його потужність, зручність і багатий набір інструментів дозволяють розробникам ефективно працювати над проєктами та забезпечувати високу якість коду.

Засновуючись на проведеному аналізі та дослідженні, вибір мови програмування Python, ігрового фреймворку Pygame є обґрунтованим і оптимальним для реалізації проекту гри у жанрі 2D платформер, такого як гра "Маріо". Ці засоби надають широкі можливості для розробки ігрового функціоналу, забезпечують зручну роботу розробників та дозволяють досягти бажаного результату в процесі реалізації проекту.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ГРИ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури

Основною метою даного проекту є створення гри в жанрі 2D платформер, яка базується на класичній грі "Super Mario". Гра має включати основні елементи, притаманні оригіналу: пересування персонажа по платформах, збір монет, подолання перешкод і битви з ворогами. Основною концепцією гри є захоплюючий ігровий процес з елементами пригод та екшну, що забезпечить гравцям задоволення та викликатиме бажання проходити рівні знову і знову.

Сюжет гри розгортається у вигаданому королівстві, де головний герой (назвемо його Маріо) має врятувати принцесу, яку викрав злий дракон. Для цього Маріо повинен пройти через декілька різноманітних рівнів, що наповнені різноманітними перешкодами, ворогами та пастками. Кожен рівень представляє собою унікальний світ з власним дизайном, музичним супроводом та викликами.

Маріо – сміливий та відважний персонаж, який володіє здатністю високо стрибати, бігати та боротися з ворогами. Він може збирати монети та інші корисні предмети, які допомагають йому в подорожі. Маріо має декілька життів, і кожна помилка може коштувати йому одного життя. Основні здібності Маріо включають стрибки на ворогів, щоб перемогти їх, і стрибки по платформах для подолання перешкод.

У грі присутні різноманітні вороги та перешкоди, які ускладнюють шлях Маріо до мети. Серед ворогів можна виділити:

1. Гумба – маленькі вороги, яких можна перемогти, стрибнувши на них зверху.

2. Купа – вороги, які рухаються по платформах та можуть бути переможені шляхом стрибка на них.

3. Бос рівня – сильний ворог, з яким Маріо зустрічається в кінці кожного рівня. Для перемоги над босом потрібно використовувати особливі стратегії.

Перешкоди включають:

1. Прірви та обриви – падіння в які призводить до втрати життя.

2. Шипи – при контакті з ними Маріо втрачає життя.

3. Рухомі платформи – вимагають точності та швидкості під час пересування.

Гра складається з декількох рівнів, кожен з яких представляє собою унікальне середовище:

1. Лісовий рівень – початковий рівень, який знайомить гравця з основними механіками гри. Він наповнений деревами, кущами та простими ворогами.

2. Бонусний рівень – рівень з піщаними дюнами, кактусами та ворогами, що адаптовані до пустельного середовища.

3. Фінальний рівень – рівень з слизькими платформами, крижаними перешкодами та морозними ворогами.

Основні механіки гри включають:

1. Пересування – Маріо може бігати ліворуч і праворуч, стрибати по платформах та підбирати предмети.

2. Стрибки – можливість стрибати на ворогів та платформи.

3. Збір монет – на кожному рівні розташовані монети, які гравець може збирати для збільшення кількості очок.

4. Здоров'я – у Маріо є кілька життів, які він може втратити при зіткненні з ворогами або падінні у прірву.

2.2. Проектування та реліз алгоритмів роботи системи

Для створення високоякісної та успішної комп'ютерної гри кожному розробнику необхідно пройти всі етапи від ідеї до введення додатку в експлуатацію. Основні етапи створення відеоігор можна поділити на п'ять складових (рис. 2.1).



Рисунок 2.1 - Основні етапи створення комп'ютерних ігор

1. Аналіз та планування:

- вивчення ринку комп'ютерних ігор та аналіз популярних трендів;
- визначення цілей ігри, цільової аудиторії та основних концепцій геймплею;
- розробка детального плану створення гри, включаючи часові рамки та ресурси.

2. Проектування:

- створення концептуального дизайну гри, включаючи сюжет, персонажі, світи, рівні та геймплейні механіки;
- розробка сценарію та сценаріїв рівнів;

- створення макетів ігрових об'єктів, інтерфейсу та візуального стилю гри.

3. Розробка:

- програмування геймплейних механік та логіки гри;
- розробка графіки, звукового супроводу та інших мультимедійних елементів;
- імплементація сценаріїв рівнів та досконалювання ігрової механіки.

4. Тестування та відладка:

- проведення внутрішнього тестування гри для виявлення помилок, багів та недоліків;
- виправлення помилок, оптимізація продуктивності та вдосконалення геймплею на основі зворотного зв'язку тестувальників.

5. Реліз та підтримка:

- підготовка гри до релізу, включаючи пакування, документацію та маркетингову стратегію;
- розповсюдження гри через платформи постачання ігор;
- підтримка гри після релізу, випуск оновлень, виправлення помилок та реагування на зворотний зв'язок користувачів.

Етап аналізу та планування. Спочатку вивчається ринок комп'ютерних ігор і проводиться аналіз популярних трендів. Визначаються цілі гри, цільова аудиторія та основна концепція геймплею. Розробляється детальний план створення гри, включаючи часові рамки та необхідні ресурси.

Етап проектування. Розробляється ідея гри, визначається її жанр, встановлюються основні ігрові механіки. Створюється концептуальний дизайн, детальний дизайн гри, графічні елементи, музика та звукові ефекти. Також проектуються складні рівні та локації.

Етап розробки. На цьому етапі створюється програмне забезпечення для реалізації задуманої гри і визначається її ігрова механіка. Ігрова механіка включає набір правил, що забезпечують інтерактивну взаємодію між гравцем

і грою. Основою механіки є об'єкти з власними властивостями, які реагують на дії гравця.

Графічне оформлення. Сьогодні графіка в іграх стала надзвичайно реалістичною з деталізованим промальовуванням і освітленням. Створення образів об'єктів починається з концептуальних ескізів, на основі яких розробляються 2D або 3D-моделі. Для рухомих об'єктів створюється анімація.

Звукове оформлення. Хоча звукове оформлення не є обов'язковим, воно грає важливу роль у створенні атмосфери гри та зануренні гравця. Фонова музика передає емоційний настрій, а звукові ефекти, такі як звуки взаємодії з предметами або кроки персонажів, створюють реалістичну акустичну обстановку.

Етап тестування та відлагодження. На цьому етапі проводиться тестування гри, виявлення та виправлення помилок. Розробляються варіанти гри для різних пристроїв та платформ.

Етап релізу та підтримки. Гра готова до випуску на ринок. Встановлюються ціни, проводяться рекламні кампанії. Після випуску забезпечується постійна підтримка гри, включаючи виправлення помилок, випуск оновлень, додаткового контенту та взаємодію з користувачами.

2.3. Реалізація гри

Визначення цільової аудиторії має велике значення при розробці гри, оскільки воно допомагає уточнити інтереси, потреби і вимоги гравців, на яких спрямована гра. Це допомагає розробнику створити гру, яка відповідає потребам цільової аудиторії, та підвищити її привабливість для цих гравців.

Визначення цільової аудиторії також допомагає в ефективному маркетингу і просуванні гри, орієнтуючи рекламні кампанії та стратегії на цільову аудиторію.

Враховуючи інтереси та вимоги гравців, розробники можуть створити більш успішну та задоволену аудиторію гри.

Цільовою аудиторією гри є геймери, які цікавляться жанром платформерів і мають інтерес до 2D ігор.

Це можуть бути як любителі ігор, так і досвідчені гравці, які насолоджуються викликами, які надає цей жанр. Оскільки платформери мають простий, але веселий геймплей, вони можуть привернути увагу широкого спектру гравців, включаючи як дорослих, так і підлітків.

Процес написання коду для гри "Super Mario" на Python з використанням бібліотеки Pygame включає декілька етапів, починаючи від налаштування середовища розробки до реалізації ігрових механік та тестування. Нижче наведено покроковий процес створення цієї гри.

1. Налаштування середовища розробки

Перед початком написання коду необхідно налаштувати середовище розробки:

1. **Visual Studio Code:** завантажується та встановлюється остання версія VS Code з офіційного сайту (<https://code.visualstudio.com/>).

2. **Установка Pygame:** відкривається командний рядок і виконується команда:

```
pip install pygame
```

2. Створення основного файлу гри

Створюється новий файл, наприклад, main.py, і імпортуються необхідні модулі:

```
import pygame
import sys
from pygame.locals import *
```

Ініціалізується Pygame та налаштовуються основні параметри вікна гри:

```
pygame.init()
```

```
# Параметри екрану
screen_width = 800
```

```

screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("Super Mario")

```

Кольори

```
WHITE = (255, 255, 255)
```

Частота кадрів

```
clock = pygame.time.Clock()
```

```
fps = 60
```

3. Створення класів для ігрових об'єктів

Створюються класи для головного героя, ворогів, платформ та інших об'єктів гри.

Клас для гравців(Player)

```
class Player(pygame.sprite.Sprite):
```

```
    def __init__(self, x, y):
```

```
        super().__init__()
```

```
        self.image = pygame.image.load("mario.png").convert_alpha()
```

```
        self.rect = self.image.get_rect()
```

```
        self.rect.topleft = (x, y)
```

```
        self.velocity_y = 0
```

```
        self.jumping = False
```

```
    def update(self):
```

```
        # Отримання вводу
```

```
        keys = pygame.key.get_pressed()
```

```
        if keys[pygame.K_LEFT]:
```

```
            self.rect.x -= 5
```

```
        if keys[pygame.K_RIGHT]:
```

```

        self.rect.x += 5

    if keys[pygame.K_SPACE] and not self.jumping:
        self.velocity_y = -15
        self.jumping = True

    # Гравітація
    self.velocity_y += 1
    if self.velocity_y > 10:
        self.velocity_y = 10
    self.rect.y += self.velocity_y

    # Перевірка зіткнення з землею
    if self.rect.bottom > screen_height:
        self.rect.bottom = screen_height
        self.jumping = False

```

Клас для платформи (Platform)

```

class Platform(pygame.sprite.Sprite):
    def __init__(self, x, y, width, height):
        super().__init__()
        self.image = pygame.Surface((width, height))
        self.image.fill((0, 255, 0))
        self.rect = self.image.get_rect()
        self.rect.topleft = (x, y)

```

4. Створення ігрового циклу

Додається основний ігровий цикл в main.py:

```

player = Player(100, screen_height - 100)
platforms = pygame.sprite.Group()
all_sprites = pygame.sprite.Group()

```

```

all_sprites.add(player)

# Створення платформи
platform = Platform(0, screen_height - 40, screen_width, 40)
platforms.add(platform)
all_sprites.add(platform)

running = True
while running:
    clock.tick(fps)

    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    # Оновлення
    all_sprites.update()

    # Перевірка зіткнення
    if pygame.sprite.spritecollide(player, platforms, False):
        player.rect.y = platform.rect.top - player.rect.height
        player.jumping = False

    # Відображення
    screen.fill(WHITE)
    all_sprites.draw(screen)
    pygame.display.flip()

```

5. Додавання інших елементів гри

Додаються інші елементи гри, такі як вороги, монети, рівні тощо, використовуючи аналогічний підхід. Створюються класи для кожного типу об'єкта та додаються до ігрового циклу.

Клас для ворога (Enemy)

```
class Enemy(pygame.sprite.Sprite):  
    def __init__(self, x, y):  
        super().__init__()  
        self.image = pygame.image.load("enemy.png").convert_alpha()  
        self.rect = self.image.get_rect()  
        self.rect.topleft = (x, y)  
        self.direction = 1  
  
    def update(self):  
        self.rect.x += 2 * self.direction  
        if self.rect.right >= screen_width or self.rect.left <= 0:  
            self.direction *= -1
```

Додавання ворогів до ігрового циклу:

```
enemy = Enemy(200, screen_height - 100)  
all_sprites.add(enemy)
```

Звукові ефекти для гри відграють велике значення. Вони допомагають покращити геймплей, передають інформацію гравцеві, впливають на настрій та забезпечують задоволення від гри. Вони додають реалізму та взаємодію з віртуальним світом, створюючи звукове оточення, яке залучає гравця. Звукові ефекти також впливають на настрій гравця. Вони можуть створювати напруженість, радість, сум, страх або будь-який інший емоційний стан, що відповідає задуму гри. Правильно підібраний звук може підсилити емоції та враження, які гравець отримує від гри.

У роботі для відтворення звуків використовуються файли формату wav з папки проєкту та AudioClip. Нижче наведений код відповідає за управління

відтворенням звукових ефектів.

```
def game_over():
    pygame.mixer.stop()
    pygame.mixer.Sound('data\\game_over.wav').play()
    image = pygame.transform.scale(load_image('game_over.png'), (1024,
512))

    image_menu = load_image('menu_btn1.png')
    continues = True
    while continues:
        for event in pygame.event.get():
            if event.type == pygame.MOUSEBUTTONDOWN:
                if 368 < event.pos[0] < 656 and 390 < event.pos[1] < 456:
                    pygame.mixer.stop()
                    continues = False
            if event.type == pygame.MOUSEMOTION:
                if 368 < event.pos[0] < 656 and 390 < event.pos[1] < 456:
                    image_menu = load_image('menu_btn2.png')
                else:
                    image_menu = load_image('menu_btn1.png')
            if event.type == pygame.QUIT:
                terminate()
        screen.blit(image, (0, 0))
        screen.blit(image_menu, (368, 390))
        pygame.display.flip()
    main_menu()
```

Всі дії у грі мають певні звуки (рис. 2.2). Наприклад, у головного героя озвучуються стрибок, зазнавання шкоди, атака, смерть. Також для атмосфери гри було додано фонову музику. Вони знаходяться в папці, де знаходяться решта файлів гри.

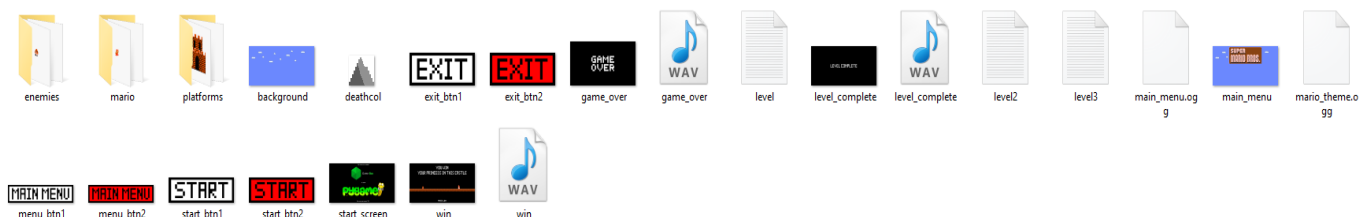


Рисунок 2.2 - Звукові файли гри

Висновки до розділу 2

У даному розділі було розглянуто реалізацію заданого темою проекту.

Була визначена концепція та сценарій гри, що надали основу для подальшої розробки. Була встановлена цільова аудиторія гри, яка дозволила спрямувати увагу на потреби та вподобання гравців.

Етап проектування гри охоплював розробку фізичних властивостей об'єктів, руху об'єктів, анімації та звукових ефектів. Ці аспекти були детально вивчені та реалізовані, щоб забезпечити плавний та реалістичний геймплей.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ГРОЮ

3.1. Графічне оформлення гри

Графічне оформлення є одним з найважливіших аспектів розробки комп'ютерних ігор, оскільки саме візуальна складова гри значною мірою визначає її привабливість для гравців. Розглянемо основні етапи створення графічного оформлення для гри в жанрі 2D платформер на прикладі "Super Mario" з використанням мови програмування Python та бібліотеки Pygame.

Спрайти головного персонажу

Для візуального представлення головного героя був використаний спрайт мario, який складається з різних спрайтів для створення анімації декількох дій, таких як стояння, біг, стрибки, отримання пошкоджень та смерть (рис. 3.1).

Додатково завантажено спрайт ворога з анімацією (рис. 3.2).

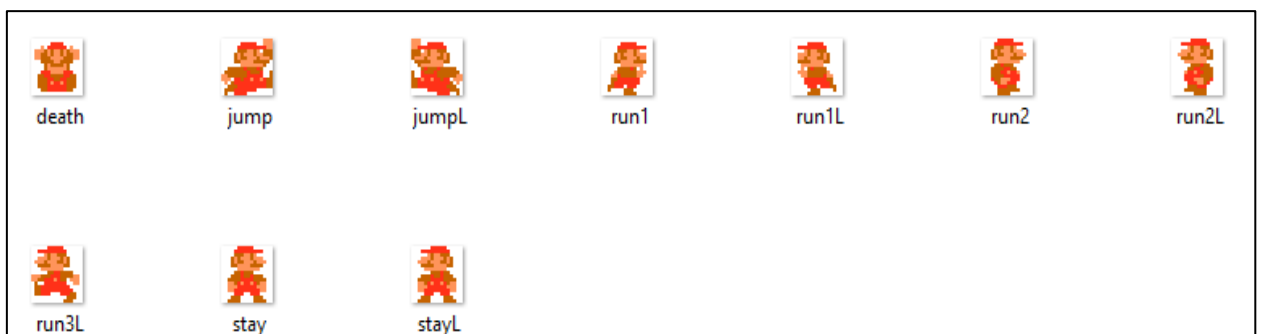


Рисунок 3.1 - Спрайти головного персонажу

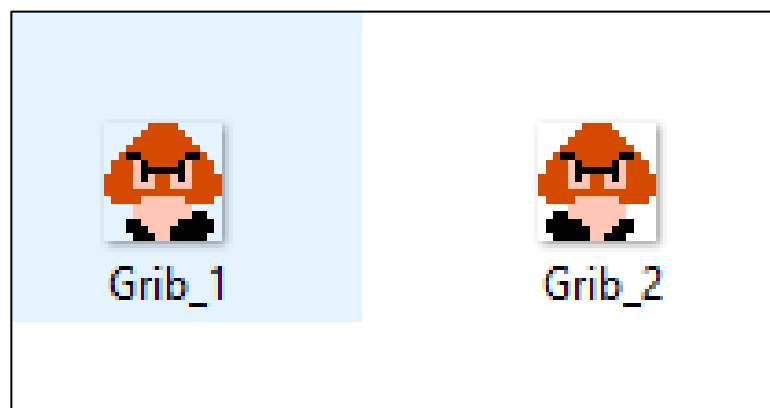


Рисунок 3.2 - Спрайт ворогу

Наступним кроком у реалізації гри є вибір картинок для фонового

зображення. Крім того, також були використані візуальні елементи для складання графічного інтерфейсу гри.

Для використання в якості фонового зображення були вибрані картинки, доступні в відкритому доступі <https://www.deviantart.com/>(рис. 3.3).



Рисунок 3.3 - Спрайти фонового зображення

Для відображення візуальних елементів гри було використано спрайти платформ у вигляді землі з травою, по яким буде стрибати та бігати ігровий персонаж (рис. 3.4).

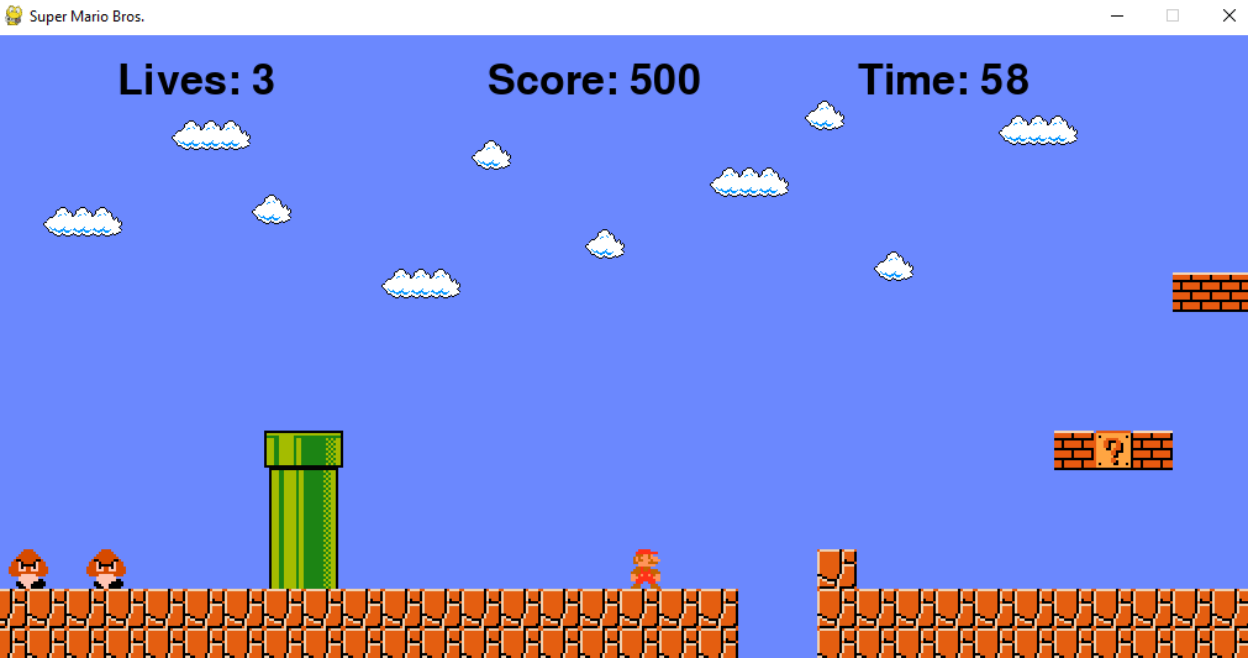


Рисунок 3.4 - Спрайти платформ

Гра складається з трьох рівнів, що вказує на наявність поступового

прогресу і зростання складності для гравця. Головний герой в цій грі має здолати різні перешкоди, що включає уникання ворожих шипів (рис. 3.5), ухилення від стріл (рис. 3.6) тощо. Однак, особливу увагу заслуговує факт того, що головний герой повинен здолати лицаря, який має здатність стріляти вогняним шаром для нанесення ураження (рис. 3.7).

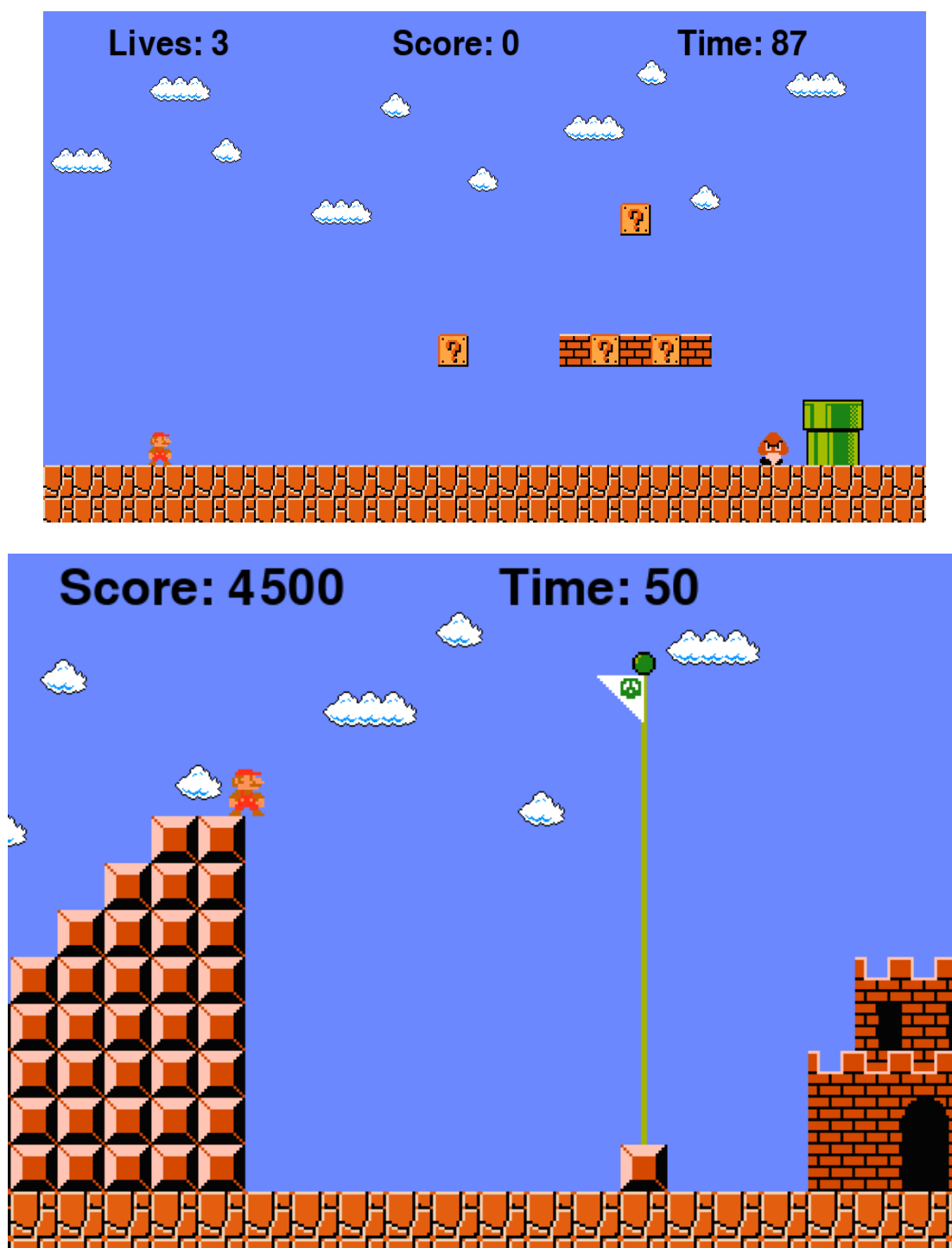


Рисунок 3.5 - Перший рівень гри

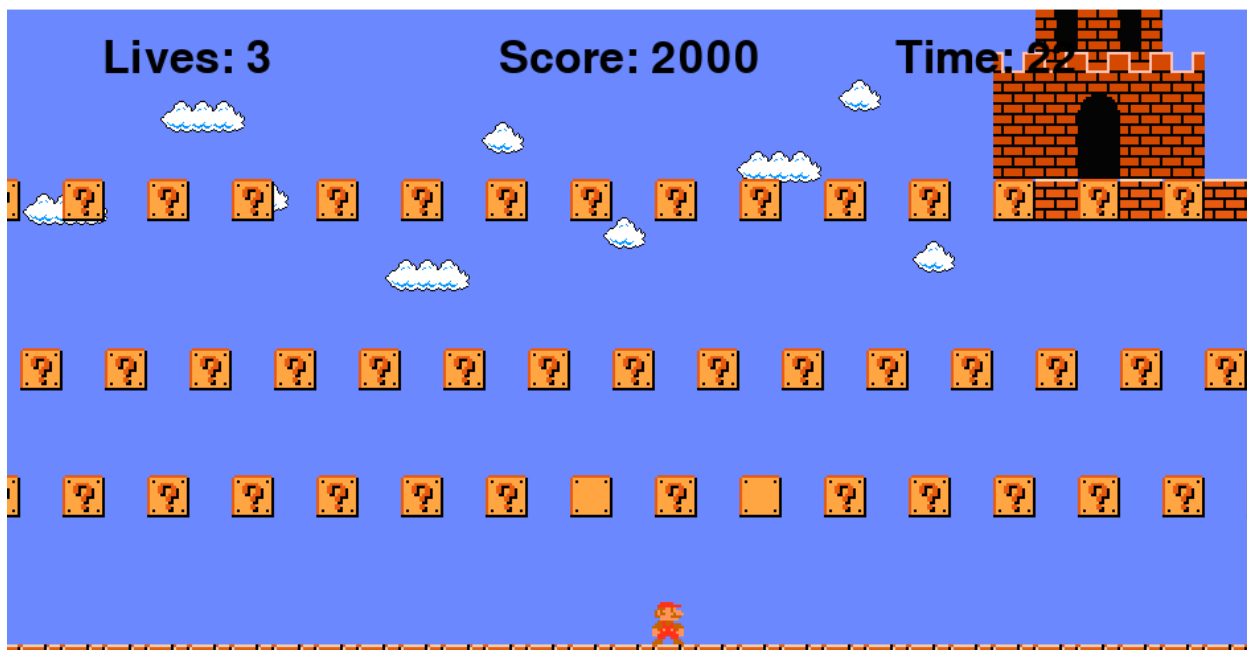


Рисунок 3.6 - Другий рівень гри

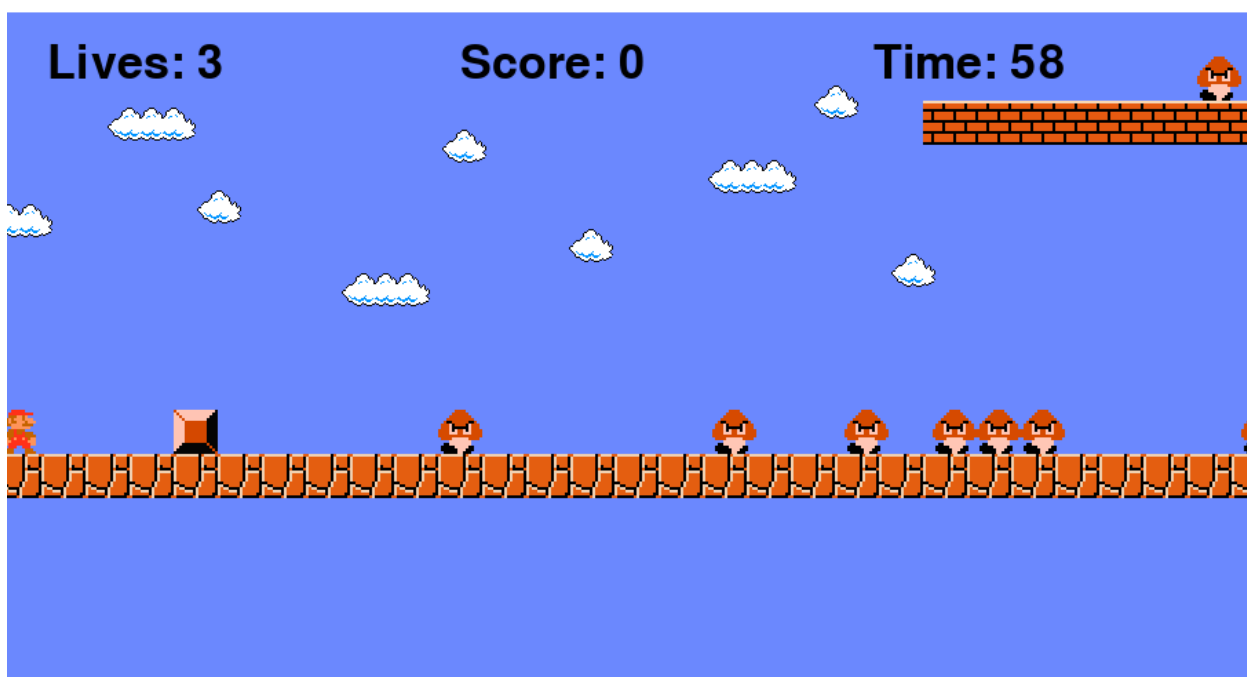


Рисунок 3.7 - Третій рівень гри

Після завершення налаштування ігрових сцен, анімацій, звуку та інших складових, створюється збірка гри. У налаштуваннях збірки можна вибрати цільову платформу, таку як Windows, MacOS, Linux або інші. Після вибору платформи, можна здійснити збірку гри, яка включатиме всі необхідні ресурси, скрипти та файли, необхідні для її виконання.

Згенерованою збіркою гри можна поділитися з іншими користувачами,

використовуючи різні методи поширення, наприклад, завантаження на платформи постачальників контенту або розповсюдження через Інтернет. Це дозволяє гравцям насолоджуватися грою на своїх пристроях і ділитися нею з іншими.

3.2. Структура інтейфейсу

Розробка початку, паузи та кінця гри.

Стартове меню гри є ключовим елементом, який впливає на перше враження гравця та його подальшу взаємодію з грою. Важливо створити яскравий, зрозумілий та зручний інтерфейс, який зацікавить і залучить гравця.

Для досягнення цього потрібно враховувати кілька факторів.

Перш за все, дизайн стартового меню повинен бути привабливим і відповідати стилістиці гри.

По-друге, стартове меню повинно бути зрозумілим та легким у використанні. Треба використовувати інтуїтивно зрозумілі символи та піктограми, щоб показати основні функції і можливості гри. Розміщувати кнопки необхідно таким чином, щоб вони були легкодоступні та зручні для натискання.

По-третє, гравець повинен мати можливість легко навігувати по стартовому меню. Необхідно забезпечити наявність кнопок "Грати", "Налаштування", "Вийти" та інших важливих функцій.

Головне меню гри складається з таких елементів як основні кнопки, що мають такі функції (рис. 3.8):

- Start – почати гру;
- Exit – вийти з гри.



Рисунок 3.8 - Початкове меню гри

Коли головний персонаж вмирає, гра завершується і з'являється меню кінця гри (рис. 3.9). У ньому гравець має можливість розпочати гру заново, повернутися до початкового меню та вийти з гри.



Рисунок 3.9 - Меню кінця гри

Висновки до розділу 3

Графічне оформлення гри було продумано ретельно: створено спрайти головного героя, інтерфейсу та ворогів, що забезпечило візуально привабливе середовище для гравців.

Окрему увагу було приділено розробці меню початку, паузи та кінця гри. Було створено відповідні екрани та функціонал, що дозволяють гравцям зручно взаємодіяти з грою та налаштовувати її параметри.

Підсумовуючи результати аналізу, можна зробити висновок, що реалізація проекту пройшла успішно та відповідає запланованим цілям і вимогам. Розроблена гра має потенціал стати цікавим та захоплюючим досвідом для цільової аудиторії. Подальший розвиток та покращення гри можуть включати розширення механік, додавання нових рівнів та завдань, покращення графічного оформлення та оптимізацію продукту для різних платформ.

ВИСНОВКИ

Постійний прогрес у розвитку комп'ютерних технологій розширює можливості геймдеву та приводить до виникнення нових тенденцій у галузі комп'ютерних ігор.

Крім того, ігрова індустрія стає все більш доступною для різних категорій користувачів. Широка аудиторія, включаючи дітей, підлітків, дорослих та навіть літніх людей, проявляє зацікавленість до ігор і використовує їх як засіб розваги, відпочинку та соціалізації.

Оскільки ігри стають все популярнішими та приносять значний прибуток, ця галузь привертає все більше талановитих розробників і інвесторів. Створення якісних ігор вимагає команди творчих професіоналів, які здатні створити неперевершений геймплей, захоплюючі сюжети та незабутні емоції для гравців.

Враховуючи все вище викладене, можна зробити висновок, що галузь комп'ютерних ігор у жанрі 2D платформер на базі ігрового движка Unity має великий потенціал та перспективи подальшого розвитку. Розробка таких ігор сприяє збагаченню ринку ігор та задоволенню потреб різноманітної аудиторії гравців. Зростання інтересу до ігор, покращення технологій та доступність розвитку ігрових проєктів роблять цю галузь привабливою для молодих спеціалістів та підприємницьких зусиль. Ігри залишаються не тільки джерелом розваги, але й потужним засобом візуалізації інформації, соціалізації та навчання, що робить їх значущим фактором у сучасному суспільстві.

У першому розділі проведено детальний аналіз предметної області, пов'язаної з комп'ютерними іграми. Дослідження поняття комп'ютерних ігор показало їх значення та важливість у сучасному світі. Класифікація комп'ютерних ігор дозволила розподілити їх на різні категорії залежно від їх основних характеристик та жанрів.

Розробка комп'ютерних ігор включає багатоетапний процес, кожен етап якого має свої специфічні завдання та вимоги. Від ідеї та концепції до релізу та підтримки – всі етапи важливі для створення якісного та захоплюючого ігрового продукту. Використання Python та бібліотеки Pygame дозволяє ефективно

реалізувати основні етапи розробки 2D платформеру, що сприяє створенню якісної гри, схожої на "Super Mario".

Огляд та аналіз додатків-аналогів були проведені з метою визначити наявність схожих програмних рішень на ринку, а також для виявлення їх переваг та недоліків. Це дозволило отримати уявлення про існуючі тенденції та інновації у галузі комп'ютерних ігор, а також зробити висновки про потенційні можливості для подальшого дослідження та розробки. Огляд показав, що в першу чергу треба звернути увагу на зручність інтерфейсу та дизайну, тому що платформери не є іграми зі складним сюжетом, характеризуються простотою геймплею і основний акцент в цьому жанрі ігор робиться на візуальну складову, тобто зрозумілий інтерфейс та гарна картинка.

Також розглянуто засоби програмної реалізації ігрового застосунку. Було обрано мову програмування Python, яка є популярною та широко використовується для розробки різноманітних програмних проєктів, включаючи відеоігри. Вона має потужні можливості та зручну синтаксичну структуру, що дозволяє ефективно реалізовувати функціонал ігрових проєктів.

Був проведений аналіз існуючих ігрових фреймворків для Python, зокрема Pygame. Виявлено, що кожен з них має свої особливості, переваги та недоліки. Цей аналіз дозволив обґрунтувати вибір Pygame як ігрового фреймворку для реалізації проєкту.

Досліджено та проаналізовано середовище розробки Python, яке є одним з найпопулярніших інструментів для розробки програмного забезпечення на. Його потужність, зручність і багатий набір інструментів дозволяють розробникам ефективно працювати над проєктами та забезпечувати високу якість коду.

Засновуючись на проведеному аналізі та дослідженні, вибір мови програмування Python, ігрового фреймворку Pygame є обґрунтованим і оптимальним для реалізації проєкту гри у жанрі 2D платформер, такого як гра "Маріо". Ці засоби надають широкі можливості для розробки ігрового функціоналу, забезпечують зручну роботу розробників та дозволяють досягти бажаного результату в процесі реалізації проєкту.

У другому розділі було розглянуто реалізацію заданого темою проекту.

Була визначена концепція та сценарій гри, що надали основу для подальшої розробки. Була встановлена цільова аудиторія гри, яка дозволила спрямувати увагу на потреби та вподобання гравців.

Етап проектування гри охоплював розробку фізичних властивостей об'єктів, руху об'єктів, анімації та звукових ефектів. Ці аспекти були детально вивчені та реалізовані, щоб забезпечити плавний та реалістичний геймплей.

У третьому розділі було ретельно продумано графічне оформлення гри: створено спрайти головного героя, інтерфейсу та ворогів, що забезпечило візуально привабливе середовище для гравців.

Окрему увагу було приділено розробці меню початку, паузи та кінця гри. Було створено відповідні екрани та функціонал, що дозволяють гравцям зручно взаємодіяти з грою та налаштовувати її параметри.

Підсумовуючи результати аналізу, можна зробити висновок, що реалізація проекту пройшла успішно та відповідає запланованим цілям і вимогам. Розроблена гра має потенціал стати цікавим та захоплюючим досвідом для цільової аудиторії. Подальший розвиток та покращення гри можуть включати розширення механік, додавання нових рівнів та завдань, покращення графічного оформлення та оптимізацію продукту для різних платформ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Креативна економіка та креативні індустрії: навч. посібн. [Електронне видання]. – Житомир: Державний університет «Житомирська політехніка», 2020. – 45 с.
2. UAPLAY – жанри комп’ютерних ігор [Електронний ресурс] - Режим доступу до ресурсу: <https://uaplay.com.ua/usi-zhanry-pc-ihor/>.
3. UAPLAY – платформер [Електронний ресурс] - Режим доступу до ресурсу: <https://uaplay.com.ua/tag/platformer/>.
4. Steam – гра Dead Cells [Електронний ресурс] - Режим доступу до ресурсу: https://store.steampowered.com/app/588650/Dead_Cells/.
5. Steam – гра Hollow Knight [Електронний ресурс] - Режим доступу до ресурсу: https://store.steampowered.com/app/367520/Hollow_Knight/.
6. Hollowknight.fandom – гра Hollow Knight [Електронний ресурс] – Режим доступу до ресурсу: [https://hollowknight.fandom.com/wiki/Hollow_Knight_\(game\)](https://hollowknight.fandom.com/wiki/Hollow_Knight_(game)).
7. Klei – гра Mark of the Ninja [Електронний ресурс] - Режим доступу до ресурсу: <https://www.klei.com/games/mark-ninja>.
8. Paris Buttfield-Addison. Unity Game Development Essentials. – O'Reilly Media. – 2019. – 216 p.
9. GamedevAcademy – ігровий движок Unity [Електронний ресурс] - Режим доступу до ресурсу: <https://gamedevacademy.org/what-is-unity>.
10. Gregory Jason. Game Engine Architecture. – New York, CRC Press. – 2019. – No. 3. – pp. 12 – 22.
11. GolosKarp.at.info – ігровий движок Unreal Engine [Електронний ресурс] - Режим доступу до ресурсу: <https://goloskarp.at.info/associated/63c5623f4a18e/>.
12. Report.if – ігровий движок Unreal Engine [Електронний ресурс] - Режим доступу до ресурсу: <https://report.if.ua/rozvagy/unreal-engine-korystuvachyji-oglyad-platformy-dlya-stvorenniya-mobilnoyi-gry/>.
13. Igrodrom — ігровий движок CryEngine [Електронний ресурс] - Режим доступу до ресурсу: <https://igrodrom.net/ua/files/cryengine-84/>.

14. Informatics—середовище розробки Visual Studio [Електронний ресурс]
- Режим доступу до ресурсу:

https://informatics.in.ua/programming_csharp/part_01.php.

15. Docs.unity3d – Box Collider 2D. [Електронний ресурс] - Режим доступу
до ресурсу: <https://docs.unity3d.com/Manual/class-BoxCollider2D.html>.

ДОДАТКИ

					ПЗ.КР.Б-121-24-ПЗ	Арк.
						55

Вихідний код гри "Super Mario"

Файл:game.py

```
import pygame
import sys
import os
import Platformes
from Player import Character
import Enemies
pygame.init()
pygame.mixer.init()

def main():
    Level(load_level('level.txt'), 90)
    level_complete()
    Level(load_level('level2.txt'), 30)
    level_complete()
    Level(load_level('level3.txt'), 60)
    win()

def load_level(filename):
    filename = "data/" + filename
    try:
        with open(filename, 'r') as mapFile:
            level_map = [line.strip() for line in mapFile]
    except FileNotFoundError as message:
        print('такого файлу не існує ', filename)
```

```
raise SystemExit()
```

```
max_width = max(map(len, level_map))
```

```
return list(map(lambda x: x.ljust(max_width, '.'), level_map))
```

```
def load_image(name, colorkey=None):
```

```
    fullname = os.path.join('data', name)
```

```
    try:
```

```
        image = pygame.image.load(fullname)
```

```
    except pygame.error as message:
```

```
        print('Cannot load image:', name)
```

```
        raise SystemExit(message)
```

```
    image = image.convert_alpha()
```

```
    if colorkey is not None:
```

```
        if colorkey is -1:
```

```
            colorkey = image.get_at((0, 0))
```

```
            image.set_colorkey(colorkey)
```

```
    return image
```

```
def initialize():
```

```
    global screen, WIDTH, HEIGHT
```

```
    WIDTH, HEIGHT = 1024, 512
```

```
    size = WIDTH, HEIGHT
```

```
    screen = pygame.display.set_mode(size)
```

```
    pygame.display.set_caption('Super Mario Bros.')
```

```
main_menu()
```

```
def main_menu():
```

```
    pygame.mixer.Sound('data\\main_menu.ogg').play(-1)
```

```

image_start = load_image('start_btn1.png')
image_exit = load_image('exit_btn1.png')
background = load_image('main_menu.png')
continues = True
while continues:
    screen.blit(background, (0, 0))
    screen.blit(image_start, (412, 250))
    screen.blit(image_exit, (434, 350))
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            terminate()
        if event.type == pygame.MOUSEMOTION:
            if 412 < event.pos[0] < 612 and 250 < event.pos[1] < 316:
                image_start = load_image('start_btn2.png')
            else:
                image_start = load_image('start_btn1.png')
            if 434 < event.pos[0] < 590 and 350 < event.pos[1] < 416:
                image_exit = load_image('exit_btn2.png')
            else:
                image_exit = load_image('exit_btn1.png')
        if event.type == pygame.MOUSEBUTTONDOWN:
            if 412 < event.pos[0] < 612 and 250 < event.pos[1] < 316:
                pygame.mixer.stop()
                continues = False
            if 434 < event.pos[0] < 590 and 350 < event.pos[1] < 416:
                terminate()
    screen.blit(background, (0, 0))
    screen.blit(image_start, (412, 250))
    screen.blit(image_exit, (434, 350))
    pygame.display.flip()

```

```

main()

def game_over():
    pygame.mixer.stop()
    pygame.mixer.Sound('data\\game_over.wav').play()
    image = pygame.transform.scale(load_image('game_over.png'), (1024, 512))
    image_menu = load_image('menu_btn1.png')
    continues = True
    while continues:
        for event in pygame.event.get():
            if event.type == pygame.MOUSEBUTTONDOWN:
                if 368 < event.pos[0] < 656 and 390 < event.pos[1] < 456:
                    pygame.mixer.stop()
                    continues = False
            if event.type == pygame.MOUSEMOTION:
                if 368 < event.pos[0] < 656 and 390 < event.pos[1] < 456:
                    image_menu = load_image('menu_btn2.png')
                else:
                    image_menu = load_image('menu_btn1.png')
            if event.type == pygame.QUIT:
                terminate()
        screen.blit(image, (0, 0))
        screen.blit(image_menu, (368, 390))
        pygame.display.flip()
    main_menu()

def level_complete():
    pygame.mixer.stop()
    image = load_image('level_complete.png')
    screen.blit(image, (0, 0))

```

```

pygame.display.flip()
music = pygame.mixer.Sound('data\\level_complete.wav')
music.play()
pygame.time.wait(int(music.get_length() * 1000))
pygame.mixer.stop()

```

```

def win():
    pygame.mixer.stop()
    image = load_image('win.png')
    screen.blit(image, (0, 0))
    pygame.display.flip()
    music = pygame.mixer.Sound('data\\win.wav')
    music.play()
    continues = True
    while continues:
        for event in pygame.event.get():
            if event.type == pygame.MOUSEBUTTONDOWN:
                continues = False
    main_menu()

```

```

def terminate():
    pygame.mixer.quit()
    pygame.quit()
    sys.exit()

```

```

def camera_configure(camera, target_rect):
    l = -target_rect.x + WIDTH/2
    t = -target_rect.y + HEIGHT/2
    w, h = camera.width, camera.height

```

```

l = min(0, l)
l = max(-(w - WIDTH), l)
t = max(-(h - HEIGHT), t)
t = min(0, t)

return pygame.Rect(l, t, w, h)

```

```

class Level:

```

```

    def __init__(self, level, time_in_sec):

```

```

        self.all_sprites = pygame.sprite.Group()
        self.platforms = pygame.sprite.Group()
        self.flag = pygame.sprite.Group()
        self.hero = pygame.sprite.Group()
        self.enemies = pygame.sprite.Group()
        self.participles = pygame.sprite.Group()
        self.finish = pygame.sprite.Group()
        self.secret_blocks = pygame.sprite.Group()

```

```

        self.player, level_length_width, level_length_height =
self.generate_level(level, time_in_sec)

```

```

        total_level_width = level_length_width * 32
        total_level_height = level_length_height * 32
        self.camera = Camera(camera_configure, total_level_width,
total_level_height)

```

```

        self.clock = pygame.time.Clock()

```

```

self.background = load_image('background.png')
self.music = pygame.mixer.Sound('data\\mario_theme.ogg')

self.play()

def generate_level(self, level, time_in_sec):
    x = 0
    y = 0
    player = None
    for row in level:
        for col in row:
            if col == '-':
                Platformes.Platform('platform', self.all_sprites, self.platforms, x, y)
            if col == 'M':
                player = Character(self.all_sprites, self.hero, x, y, time_in_sec)
            if col == 'B':
                Platformes.Platform('block', self.all_sprites, self.platforms, x, y)
            if col == 'L':
                Platformes.Platform('ladder', self.all_sprites, self.platforms, x, y)
            if col == '?':
                Platformes.SecretBlock('secret_block', self.all_sprites,
self.platforms, self.secret_blocks, x, y)
            if col == 't':
                Platformes.Platform('pipe_L', self.all_sprites, self.platforms, x, y)
            if col == 'y':
                Platformes.Platform('pipe_R', self.all_sprites, self.platforms, x, y)
            if col == 'T':
                Platformes.Platform('pipe_TL', self.all_sprites, self.platforms, x, y)
            if col == 'Y':
                Platformes.Platform('pipe_TR', self.all_sprites, self.platforms, x, y)

```

```

        if col == 'I':
            Platformes.NotObstacle('flag_I', self.all_sprites, x, y)
        if col == 'O':
            Platformes.NotObstacle('flag_O', self.all_sprites, x, y)
        if col == 'F':
            Platformes.NotObstacle('flag_I', self.all_sprites, x, y)
            Platformes.NotObstacle('flag_F', self.all_sprites, x - 16, y)
        if col == 'G':
            Enemies.Grib(self.all_sprites, self.enemies, x, y)
        if col == '*':
            Platformes.Castle('castle', self.all_sprites, self.finish, x, y)
        x += 32
        y += 32
        x = 0
    return player, len(level[0]), len(level)

```

```

def play(self):
    self.music.play(-1)
    left = False
    right = False
    up = False
    continues = True
    gameover = False
    while continues:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                terminate()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_LEFT:
                    left = True

```

```

        elif event.key == pygame.K_RIGHT:
            right = True
        elif event.key == pygame.K_UP:
            up = True
    if event.type == pygame.KEYUP:
        if event.key == pygame.K_LEFT:
            left = False
        elif event.key == pygame.K_RIGHT:
            right = False
        elif event.key == pygame.K_UP:
            up = False
    screen.blit(self.background, (0, 0))
    self.player.update(left, right, up, self.platforms, self.enemies,
                       self.participles, self.secret_blocks, screen)
    self.enemies.update(self.platforms, self.enemies)
    self.participles.update(self.camera.state)
    self.camera.update(self.player)
    for sprite in self.all_sprites:
        if self.player.rect.x - 1024 < sprite.rect.x < self.player.rect.x + 1024: #
оптимізація гри))
            screen.blit(sprite.image, self.camera.apply(sprite))
    self.player.draw_UI(screen)
    self.clock.tick(60)
    pygame.display.flip()
    if pygame.sprite.spritecollideany(self.player, self.finish):
        continues = False
    if self.player.lives == 0:
        continues = False
        gameover = True
    if self.player.timer <= 0:

```

```

        continues = False
        gameover = True
    if gameover:
        game_over()

```

```

class Camera(object):
    def __init__(self, camera_func, width, height):
        self.camera_func = camera_func
        self.state = pygame.Rect(0, 0, width, height)

    def apply(self, target):
        return target.rect.move(self.state.topleft)

    def update(self, target):
        self.state = self.camera_func(self.state, target.rect)

initialize()
start_screen()

```