**Міністерство освіти і науки України**
**Національний технічний університет**
**«Дніпровська політехніка»**

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

**Кафедра** Програмного забезпечення комп'ютерних систем
(повна назва)

# ПОЯСНЮВАЛЬНА ЗАПИСКА
## кваліфікаційної роботи ступеня

*бакалавра*
(назва освітньо-кваліфікаційного рівня)

**студента** *Волошина Андрія Дмитровича*
(ПІБ)

**академічної групи** *122-20ск-1*
(шифр)

**спеціальності** *122 Комп'ютерні науки*
(код і назва спеціальності)

**освітньої програми** *Комп'ютерні науки*
(назва освітньої програми)

**на тему:** *Розробка комп'ютерної гри Herro: Part I з використанням Unity C#*

| **Керівники** | **Прізвище, ініціали** | **Оцінка за шкалою** | | **Підпис** |
|---|---|---|---|---|
| | | **рейтинговою** | **інституційною** | |
| кваліфікаційної роботи | *доц. Кабак Л.В.* | | | |
| **розділів:** | | | | |
| спеціальний | *доц. Кабак Л.В.* | | | |
| економічний | *проф. Вагонова О.Г.* | | | |
| | | | | |
| | | | | |
| **Рецензент** | *доц. Каштан В.Ю.* | | | |
| **Нормоконтролер** | *доц. Гуліна І.Г.* | | | |

**Дніпро**
**2023**

**Міністерство освіти і науки України**
**НТУ «Дніпровська політехніка»**

**ЗАТВЕРДЖЕНО:**
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

# ЗАВДАННЯ
## на кваліфікаційну роботу

*бакалавра*
(назва освітньо-кваліфікаційного рівня)

**студента** *122-20ск-1* *Волошина А.Д.*
(група) (прізвище та ініціали)

**тема кваліфікаційної роботи** *Розробка комп'ютерної гри Herro: Part I з використанням Unity C#*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

| Розділ | Зміст виконання | Термін виконання |
|---|---|---|
| ***Спеціальний*** | ***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми*** | ***13.05.2023 р.*** |
| ***Економічний*** | ***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки*** | ***27.05.2023 р.*** |

Завдання видав _____ _____
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Волошин А.Д.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

# РЕФЕРАТ

Пояснювальна записка: 69 с., 28 рис., 0 табл., 3 дод., 19 джерел.

Об'єкт розробки: комп'ютерна гри Herro: Part I з використанням Unity C#.

Мета кваліфікаційної роботи: створення комп’ютерної гри у жанрі 2D-платформер, а все задля того, щоб розташувати гру на сучасних цифрових магазинах, наприклад Steamstore (від компанії Valve Corp.) або Epicgamestore (від компанії Epic Games) та отримувати прибуток від продаж. Користувачі в свою чергу одержують гру від якої отримують гарні емоції при прохожденні. Все це створено у програмі Unity3D використовуючи двигун Unity від компанії Unity Technologies, та мову програмування C#.

Вступ розглядає аналіз та сучасний стан проблеми, конкретизує мету кваліфікаційної роботи та галузь її застосування, наводиться обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, встановлено актуальність завдання та мету розробки, сформульовано постановку завдання та вказано вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі було проведено аналіз існуючих рішень, вибрані платформи для розробки, виконано проектування і розробку програми. Описано роботу програми, включаючи алгоритм та структуру її функціонування. Також розглянуто виклик та завантаження програми, визначено вхідні і вихідні дані та детально охарактеризовано набір параметрів технічних засобів.

У розділі, присвяченому економіці, була встановлена трудомісткість розробленої інформаційної системи. Був проведений розрахунок вартості роботи зі створення програми та визначений час, необхідний для його виконання.

Практичне значення полягає у створенні гри, що надає можливість відволіктися від усіх проблем, які оточують користувача, і насолоджуватися грою отримуючи емоції від проходження рівнів або повного проходження гри в кінці.

Ігри залишаються дуже актуальними у сучасному світі, оскільки вони пропонують не тільки розвагу, але й можливість відпочити, втілити фантазії та зануритися в інші світи. Вони також сприяють соціальному взаємодії, спільній грі з друзями або онлайн спільнотами, що робить їх популярними та цікавими для широкої аудиторії.

Список ключових слів: ДВИГУН, ПЛАТФОРМЕР, ГЕЙМІНГ, ГЕЙМЕР.

# ABSTRACT

Explanatory note: 69 pp., 28 fig., 3 extra., 19 sources.

Object of development: computer game Herro: Part I using Unity C#.

The purpose of the qualification work: to create a computer game in the genre of 2D platformer, and all in order to place the game on modern digital stores, such as Steamstore (from Valve Corp.) or Epicgamestore (from Epic Games) and make a profit from sales. Users, in turn, receive a game that gives them good emotions when they play it. All this was created in the Unity3D program using the Unity engine from Unity Technologies and the C# programming language.

The introduction considers the analysis and current state of the problem, specifies the purpose of the qualification work and its scope, provides a justification for the relevance of the topic and clarifies the task statement.

The first section analyzes the subject area, establishes the relevance of the task and the purpose of development, formulates the task statement and specifies the requirements for software implementation, technologies and software tools.

The second section analyzes existing solutions, selects platforms for development, and designs and develops the program. The program operation is described, including the algorithm and structure of its functioning. It also describes how to call and load the program, defines the input and output data, and describes in detail the set of parameters of the technical means.

In the section on economics, the labor intensity of the developed information system was established. The cost of creating the program was calculated and the time required for its implementation was determined.

The practical significance is to create a game that provides an opportunity to distract from all the problems that surround the user and enjoy the game, getting emotions from passing levels or completing the game at the end.

Games remain very relevant in the modern world as they offer not only entertainment but also the opportunity to relax, fulfill fantasies and immerse yourself in other worlds. They also facilitate social interaction, joint play with friends or online

communities, which makes them popular and interesting for a wide audience.

Keywords: ENGINE, PLATFORMER, GAMING, GAMER.

**ЗМІСТ**

# ВСТУП

У сучасному світі практично кожна особа має можливість користуватися глобальною мережею. З цього випливає, що практично будь-яка сфера діяльності пов'язана з Інтернетом. Робота, навчання, розваги - все це все більше й більше здійснюється в онлайн-режимі. Тому створення та підтримка сучасних ігор стали надзвичайно актуальними, завдяки значному поширенню Інтернету.

Ігри відіграють значну роль у сучасному світі та мають вплив на різні аспекти нашого життя. Вони стали важливою складовою культури та розваг, а також мають соціальний, економічний та освітній вплив. Ось деякі ключові ролі ігор у сучасному світі:

1. Розвага та відпочинок. Ігри є популярною формою розваги, яка надає людям можливість відпочити, відволіктися від повсякденних справ і насолодитися взаємодією з віртуальним світом. Вони надають можливість відчути емоції, випробувати нові витівки та створювати незабутні враження.

2. Соціальна взаємодія. Багато ігор пропонують мультиплеєрний режим, що дозволяє гравцям спілкуватися та співпрацювати в онлайн-середовищі. Це стимулює соціальну взаємодію, співпрацю та конкуренцію між гравцями з різних країн та культур.

3. Розвиток навичок. Ігри можуть сприяти розвитку різних навичок та когнітивних здібностей. Вони вимагають стратегічного мислення, прийняття рішень у складних ситуаціях, реакції на зміни та співпраці з іншими гравцями. Деякі ігри також сприяють розвитку творчості, логічного мислення та проблемного мислення.

4. Освіта та навчання. Ігри можуть бути використані як педагогічний інструмент для навчання та набуття знань у різних галузях. Вони можуть навчати історії, науки, мови, математики та багатьох інших предметів через інтерактивний та залучний спосіб. Ігри дозволяють студентам розвивати критичне мислення, проблемне розв'язування та співпрацю, а також надають можливість для практичного застосування отриманих знань.

5. Економічний вплив. Галузь геймінгу має значний економічний вплив у сучасному світі. Ігрова індустрія стала однією з найбільш дохідних та швидкорозвиваючихся галузей, створюючи робочі місця та внесок до ВВП багатьох країн. Вона також стимулює інновації в галузі технологій, комп'ютерної графіки та інших суміжних галузях.

6. Спільноти та культура. Ігри об'єднують людей у спільноті зі спільними інтересами. Гравці обмінюються досвідом, стратегіями та створюють власні культурні норми та традиції. Вони створюються спільноти, які взаємодіють через форуми, соціальні мережі та зустрічі, сприяючи зв'язкам та обміну ідеями.

7. Терапевтичний вплив. Деякі ігри використовуються в якості терапевтичного інструменту для покращення фізичного та психологічного стану людей. Наприклад, використовуються відеоігри для реабілітації після травми або для зменшення стресу та тривоги.

Загалом, ігри відіграють важливу роль у сучасному світі, надаючи розвагу, спілкування, навчання та сприяючи економічному розвитку. Вони стали не просто формою розваги, але й важливим культурним феноменом, що впливає на наші життя та суспільство у цілому.

2D ігри є популярними серед гравців у всьому світі. Вони пропонують захоплюючий геймплей, візуальну привабливість та можливість втягнутися в захоплюючий світ гри. Метою моєї роботи є створення власної 2D гри, використовуючи відповідні інструменти та технології.

Розробка ігор включає кілька етапів, кожен з яких є важливим для створення якісної та задовільної гри. Ось основні етапи розробки ігор:

1. Концептуалізація. Цей етап включає визначення ідеї та концепції гри. Розробники визначають основні механіки, тему, атмосферу та цільову аудиторію гри. Також створюються концепт-арт, опис гри та дизайн персонажів.

2. Прототипування. У цьому етапі створюються прототипи гри, які дозволяють перевірити та валідувати ігрові механіки, управління та інтерфейс. Прототипи можуть бути простими зразками гри, зображеннями, або навіть простими інтерактивними демонстраціями.

3. Геймдизайн. На цьому етапі розробляються деталі гри, такі як рівні, завдання, системи прогресу, баланс гри, штучний інтелект, аудіо та візуальні ефекти. Важливо створити цікавий та викликаючий ігровий досвід для гравців.

4. Розробка. Після завершення геймдизайну розпочинається активна розробка гри. Розробники працюють над програмуванням геймплею, рівнями, графікою, аудіо та іншими аспектами гри. Використовуються різні інструменти та технології, такі як движки розробки ігор, для забезпечення ефективності та продуктивності розробки.

5. Тестування. Після завершення розробки проводяться тестування гри для виявлення та виправлення помилок, багів та недоліків. Тестування також допомагає оцінити геймплей, баланс гри та загальний досвід гравців. Тестування може бути як внутрішнім (розробники та тестери власної команди), так і зовнішнім (залучення зовнішніх тестерів або бета-тестування з гравцями).

6. Випуск та підтримка. Після успішного завершення тестування гра готова до випуску. Вона може бути опублікована на різних платформах, таких як комп'ютери, консолі, мобільні пристрої або онлайн-платформи. Після випуску розробники також можуть надалі підтримувати гру, випускаючи оновлення, доповнення та виправлення.

Кожен етап розробки ігор вимагає ретельного планування, комунікації та співпраці між розробниками. Це дозволяє створити якісну та успішну гру, яка задовольняє потреби гравців.

Метою кваліфікаційної роботи є створення комп’ютерної 2D гри, яка буде цікава великій кількості гравців, які є фанатами цієї категорії ігор. За основу взятий язик програмування C#. А сам проект створювався на двигуну Unity3D.

Беручи це все до уваги було сформовано тему кваліфікаційної роботи: «Розробка комп'ютерної гри Herro: Part I з використанням Unity C#.».

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1. Загальні відомості з предметної галузі

Людині завжди потрібен відпочинок, чи від роботи, чи від навчання, чи після фізичних навантажень. Тому тема комп’ютерних ігор зараз має великий пріоритет.

Що таке комп’ютерні ігри взагалі? Комп'ютерні ігри - це програмне забезпечення, яке дозволяє користувачам взаємодіяти з віртуальним світом або сценаріями, створеними на комп'ютері. Вони можуть мати різні форми, включаючи відеоігри, онлайн-ігри, мобільні ігри, комп'ютерні ігри в доповненій реальності (AR) та віртуальній реальності (VR).

Комп'ютерні ігри зазвичай включають в себе взаємодію з гравцями або штучним інтелектом через управління персонажем або об'єктами у віртуальному середовищі гри. Гравці можуть керувати персонажем, виконувати завдання, розв'язувати головоломки, виконувати віртуальні спортивні заходи або взаємодіяти з іншими гравцями у мультиплеєрному режимі.

Комп'ютерні ігри можуть бути розроблені для розваги, освіти, тренування навичок або співробітництва. Вони можуть мати різні жанри, такі як екшн, пригодницькі, стратегічні, рольові, головоломки, гонки та багато інших.

За допомогою комп'ютерних ігор гравці можуть погрузитися у віртуальні світи, створені розробниками, і насолоджуватися уявною реальністю, яка може бути відмінною від їхнього повсякденного життя. Комп'ютерні ігри стали популярним видом розваги, що має великий вплив на культуру і суспільство.

Існує безліч різновидностей комп'ютерних ігор. Ось кілька загальновідомих жанрів:

1. Екшн. Це жанр, де гравець контролює головного героя, який зазвичай займається бойовими діями і бореться з ворогами або рішенням складних завдань. Приклади: "Call of Duty", "Assassin's Creed", "Grand Theft Auto".

2. Пригодницькі. Це жанр, де гравець проводить час, розв'язуючи головоломки, вирішуючи логічні завдання та розслідуючи історію. Це може бути розповідальна гра з фокусом на характери та сюжет. Приклади: "The Legend of Zelda", "Uncharted", "Tomb Raider".

3. Рольові (RPG). Це жанр, де гравець приймає роль фіктивного персонажа і керує його характеристиками та діями. Гравець може вирішувати завдання, вдосконалювати навички персонажа, взаємодіяти з іншими персонажами та розвивати уявний світ. Приклади: "The Elder Scrolls V: Skyrim", "Fallout", "Final Fantasy".

4. Стратегічні. Це жанр, де гравець приймає рішення про керування ресурсами, розміщенням військ та плануванням стратегії для досягнення мети. Це може бути стратегія в реальному часі (Real-Time Strategy, RTS) або пошагова стратегія (Turn-Based Strategy). Приклади: "Civilization", "StarCraft", "XCOM".

5. Головоломки. Це жанр, в якому гравець розв'язує логічні головоломки, задачі або складні завдання. Це може включати лабіринти, розбивання кодів, складання головоломок та інші ментальні виклики. Приклади: "Portal", "Tetris", "Braid".

6. Симулятори. Це жанр, в якому гравець може відтворювати реальне життя або віртуальний світ, симулюючи певні аспекти діяльності, такі як керування авіалайнером, фермою, автомобілем або будівництвом міста. Приклади: "The Sims", "Euro Truck Simulator", "Farming Simulator".

Це лише кілька прикладів різновидностей комп'ютерних ігор, і вони можуть між собою поєднуватися або мати піджанри. Жанри комп'ютерних ігор постійно розвиваються, і нові ігри часто поєднують елементи з різних жанрів для створення унікального досвіду гри. Наразі така модель планування процесів дуже популярна в бізнесі та підприємствах, зокрема в IT-компаніях. Це стосується як малого бізнесу, так і середнього, і великого.

Перша визнана комп'ютерна гра називалася "Spacewar!" і була розроблена групою студентів Массачусетського технологічного інституту (MIT) під керівництвом Стіва Рассела і Мартина Грейсмейл у 1962 році (рис. 1.1.). Гра

"Spacewar!" була космічним симулятором, в якому два гравці могли керувати космічними кораблями, воювати один з одним та збивати зоряні кораблі.

Рис. 1.1. Фото першої комп’ютерної гри "Spacewar!".

Варто відзначити, що існує багато ранніх комп'ютерних ігор, які були створені в різних часах та незалежно один від одного. Наприклад, в 1958 році, фізик Вільям Гіггінсботом створив гру "Tennis for Two" для комп'ютера Donner Model 30. Вона вважається однією з перших комп'ютерних ігор, які були запущені на електронних комп'ютерах (рис. 1.2.).

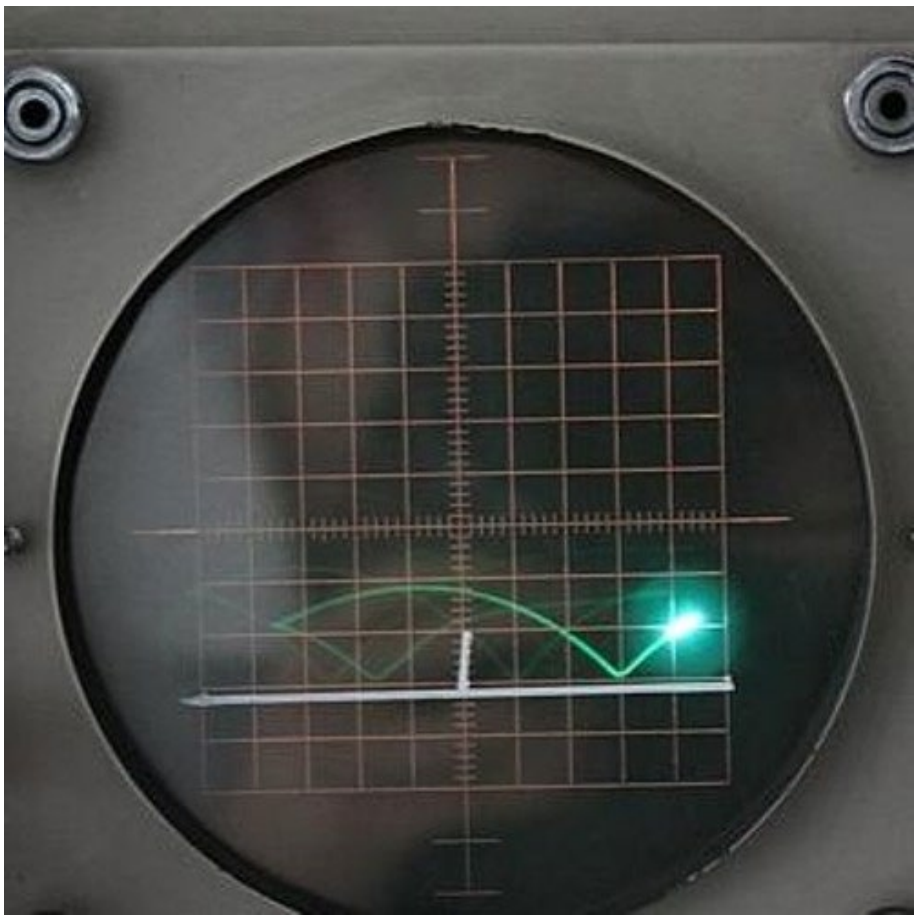

Рис. 1.2. Фото комп’ютерної гри "Tennis for Two".

Таким чином, існує декілька ранніх комп'ютерних ігор, і визнання першої залежить від критеріїв, які застосовуються. Проте "Spacewar!" зазвичай вважається однією з найвпливовіших і перших комп'ютерних ігор, що знайшла широке визнання та поширення.

Однією з ключових причин розвитку комп'ютерних ігор є постійний технологічний прогрес. Протягом останніх десятиліть зростання обчислювальної потужності комп'ютерів, розширення графічних можливостей, поліпшення звукового відтворення та введення нових технологій, таких як віртуальна реальність, привели до збільшення можливостей і реалістичності комп'ютерних ігор.

Інший важливий фактор - це зростання популярності інтернету та розвиток онлайн-ігор. Можливість грати з іншими гравцями з усього світу в режимі

реального часу відкрила нові можливості для соціальної взаємодії та конкуренції в ігровому середовищі.

Також варто відзначити, що комп'ютерні ігри стали значним економічним сегментом. Великі компанії, такі як Electronic Arts, Activision Blizzard та Ubisoft, інвестують значні кошти в розробку і маркетинг комп'ютерних ігор. Ігрова індустрія привертає велику кількість гравців та генерує значні доходи, що стимулює подальший розвиток та інновації у галузі.

Не менш важливою причиною росту комп'ютерних ігор є інтерес користувачів до віртуальних світів та можливості втілення в них різних ролей і фантазій. Ігри надають можливість експериментувати, випробувати нові ролі, зануритися у фантастичні або реалістичні світи та відчувати себе частиною захоплюючих історій.

Отже, постійний технологічний прогрес, зростання популярності онлайн-ігор, економічна вигідність та інтерес користувачів до віртуальних світів є ключовими факторами, що сприяють розвитку комп'ютерних ігор.

В наш час комп’ютерні ігри набирають велику популярність серед ПК користувачів, особливо підліткового віку. Звісно у них є свої плюси та мінуси.

Головні плюси комп’ютерних ігор:

1. Знімають стрес та втому.

2. Розвиток когнітивних навичок.

3. Розвиток соціальних взаємин.

4. Творчий потенціал.

5. Можливість знайти заробіток.

Хоча комп'ютерні ігри мають багато переваг, вони також можуть мати деякі негативні аспекти.

Мінуси комп’ютерних ігор:

1. Витрачення часу.

2. Соціальна ізоляція.

3. Проблеми із здоров’ям.

4. Лудоманство.

Одні з головних частин програмного забезпечення та комп’ютерних ігор для звичайного користувача це:

1. Зручний та приємний інтерфейс.

2. Якнайбільша візуалізація компонентів системи.

3. Легкість в експлуатації.

4. Наявність навчання або підказок.

Інтерфейс гри - це спосіб взаємодії гравця з комп'ютерною грою. Він включає в себе всі елементи та контроли, які гравець використовує для керування грою, отримання інформації та сприйняття грального процесу. Основними складовими інтерфейсу гри є:

1. Графічний інтерфейс користувача (GUI). Графічний інтерфейс включає в себе всі візуальні елементи гри, такі як головне меню, ігрове поле, панель інструментів, іконки, текстові повідомлення тощо. GUI надає гравцю візуальну зв'язок з грою.

2. Керування грою. Це включає в себе різні пристрої та контролери, такі як клавіатура, миша, геймпад, джойстик або сенсорний екран. Гравець використовує ці засоби для керування персонажем або об'єктами в грі.

3. Аудіоінтерфейс. Звуковий інтерфейс включає в себе всі аудіоефекти, звукові елементи та музику в грі. Він допомагає створювати атмосферу і передавати аудіальні враження гравцю.

4. Меню та налаштування. Інтерфейс гри також містить меню, де гравець може вибирати режими гри, налаштовувати параметри, зберігати прогрес, виконувати налаштування графіки, звуку та керування.

5. Інформаційні елементи. Інтерфейс гри включає в себе різні інформаційні елементи, такі як панель здоров'я, міні-карта, показники очків, рівень гравця, завдання та інше. Ці елементи надають гравцеві необхідну інформацію про стан гри та його прогрес.

Інтерфейс гри повинен бути зрозумілим, інтуїтивним та зручним для гравця, дозволяючи йому насолоджуватися грою та легко взаємодіяти з віртуальним світом (рис. 1.3.).


Рис. 1.3. Скріншот інтерфейсу гри «The SIMS 2».

Розглянемо деякі популярні двигуни для розробки комп’ютерних ігор, їх переваги та недоліки.

Unity3D - це потужний рушій та інтегроване середовище розробки (IDE), що використовується для створення ігор, візуалізації, віртуальної реальності (VR), доповненої реальності (AR) та інших інтерактивних додатків. Він був розроблений компанією Unity Technologies і став одним із найпопулярніших інструментів для створення геймплею.

Unity3D надає розробникам широкий набір інструментів та функцій, які дозволяють їм створювати ігри та додатки для різних платформ, включаючи комп'ютери, мобільні пристрої, консолі, веб-браузери, а також віртуальну та доповнену реальність. Він підтримує різні мови програмування, такі як C# та JavaScript, та надає зручний інтерфейс для створення та керування ігровими

об'єктами, сценами, анімацією, фізикою, звуком та іншими елементами геймплею.

Середовище розробки Unity3D має візуальний редактор, який дозволяє розробникам створювати сцени та ігрові об'єкти шляхом перетягування та налаштування компонентів. Воно також надає доступ до багатьох ресурсів та бібліотек, таких як готові моделі, текстури, звуки та спецефекти, що значно спрощує процес створення ігор.

Unity3D має багатий набір функцій, включаючи підтримку фізичної симуляції, штучного інтелекту, анімації персонажів, освітлення, аудіо ефектів, створення користувацького інтерфейсу та багато іншого. Він також надає можливість інтеграції з різними сторонніми інструментами та платформами, що робить його універсальним вибором для розробників ігор та додатків.

Unity3D дозволяє розробляти ігри як для мобільних пристроїв і настільних комп'ютерів, так і для консолей та веб-браузерів. Він підтримує багато платформ, включаючи iOS, Android, Windows, macOS, PlayStation, Xbox та інші.

Unity3D має активну спільноту розробників, яка надає навчальні матеріали, посібники, приклади коду та плагіни для розширення функціональності рушія. Це робить Unity3D доступним як для початківців розробників, так і для професіоналів, які прагнуть створити якісні ігрові та інтерактивні додатки.

Інтерфейс двигуна Unity3D (рис. 1.4.) виглядає так:

Рис. 1.4. Інтерфейс додатку Unity3D.

Основними перевагами Unity3D є:

1. Крос-платформеність.

2. Великий набір інструментів.

3. Велике співтовариство.

4. Швидкість розробки.

Unity 3D - це потужний рушій інтерактивного візуалізації та розробки ігор, розроблений компанією Unity. Він є одним з найпопулярніших інструментів для створення високоякісних ігор, віртуальної реальності (VR), доповненої реальності (AR) та інших інтерактивних візуальних додатків.

Unity3D надає розробникам широкий набір інструментів і функцій, що дозволяють створювати реалістичні та ефектні графічні сцени, фізичні симуляції, штучний інтелект, анімацію персонажів, освітлення, звукові ефекти та багато іншого. Він має потужний візуальний редактор, який дозволяє розробникам створювати і маніпулювати об'єктами та сценами у зручний спосіб.

Однією з головних переваг Unity3D є його фотореалістичний рендеринг. Рушій використовує передові технології, такі як рейтрейсинг в реальному часі та глобальне освітлення, що дозволяють створювати надзвичайно реалістичні графічні сцени з високою якістю освітлення, відблисків та тіней.

Unity3D також має підтримку кросплатформенної розробки, що дозволяє створювати ігри для різних платформ, включаючи комп'ютери, консолі, мобільні пристрої та веб-браузери. Рушій підтримує такі платформи, як Windows, macOS, Linux, PlayStation, Xbox, Nintendo Switch та інші.

Крім того, Unity3D має активну спільноту розробників, яка надає різноманітні навчальні матеріали, документацію, уроки та плагіни, що сприяють розширенню можливостей рушія. Спільнота також забезпечує підтримку, обмін знаннями та можливість взаємодії з іншими розробниками.

Узагалі, Unity3D є потужним інструментом для створення високоякісних ігор та інтерактивних візуальних додатків, і він широко використовується як незалежними розробниками, так і великими студіями розробки ігор.

Unity3D має такий інтерфейс (рис. 1.5.):

Рис. 1.5. Інтерфейс додатку Unity3D.

## 1.2. Призначення розробки та область застосування.

Темою кваліфікаційної роботи виступає: «Розробка комп'ютерної гри Herro: Part I з використанням Unity C#». В якості функціональної основи взята мова програмування C#. Тобто головною метою роботи є створення комп’ютерної гри, яка виконана у жанрі 2D.

Головними критеріями розроблювальної комп’ютерної гри є:

1. Зручний інтерфейс.
2. Легкість у куруванні.
3. Зрозумілий геймплей.

Гра призначена для:

1. Відпочинку.
2. Змагання серед інших гравців.
3. Отримання гарних емоцій.

Система позиціонується як 2D гра, у яку зможе зіграти майже кожен геймер, незважаючи на вік, стать та розумові здатності.

## 1.3. Підстава для розробки

Відповідно до ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

1. ОПП 122 «Комп’ютерні науки»;
2. графік навчального процесу та навчальний план;
3. наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;
4. завдання на кваліфікаційну роботу на тему «Розробка комп'ютерної гри Herro: Part I з використанням Unity C#».

## 1.4. Постановка завдання

Метою кваліфікаційної роботи є розробка комп'ютерної 2D гри. За основу взята мова програмування C# та двигун Unity3D. Також весь інтерфейс та візуальні частини гри були розроблені у Adobe Photoshop.

Гра повинен реалізувати такі дії як:

1. Збереження даних локально.
2. Створення головного та другорядних персонажів.
3. Можливість ставити гру на паузу.
4. Можливість почати гру спочатку.

Для виконання проекту необхідно:

1. Проаналізувати існуючі ігри.
2. Спроектувати логіку гри.
3. Розробити дизайн та зручний інтерфейс додатку.

## 1.5. Вимоги до програми або програмного виробу.

### 1.5.1. Вимоги до функціональних характеристик.

Розроблене програмне забезпечення, для того, щоб досягнути поставлених цілей, повинно підтримувати виконання таких дій:

1. Реагування на дії користувача в локальному режимі.
2. Авто-збереження будь яких змін користувачем.
3. Надання максимально швидкої навігації по коду гри.

Для підтримки вище перераховані функцій у додатку має бути реалізовано:

1. Програмна та апаратна сумісності.
2. Стандартна конфігурація яка дає змогу ввести застосунок в експлуатацію.

### 1.5.2. Вимоги до інформаційної безпеки.

Для коректної роботи програми потрібно реалізувати:

1. Можливість редагування даних.
2. Можливість тривалої роботи протягом 24 годин (1 доба).
3. Контроль та обробка вхідних даних.
4. Збереження цілісності даних у випадку збою системи.
5. Локальне збереження даних для більшої захищеності.

Також система візуального планування задач повинна мати наступні характеристики:

1. Захист від несанкціонованого доступу.
2. Відновлення після збою протягом 10 хвилин.
3. Захист даних користувача при повторному підключені.

### 1.5.3. Вимоги до складу та параметрів технічних засобів.

Для забезпечення надійного функціонування програмного забезпечення необхідно, щоб обчислювальна машина, на якій буде експлуатуватися комп’ютерна гра, мала такі характеристики:

1. Маніпулятор “миша”.
2. Клавіатура.
3. Доступ до онлайн мережі.
4. 10 Гб вільного місця на жорсткому диску.
5. Процесор Intel Core i5-6400 з тактовою частотою 2.7 ГГц.
6. Не менше ніж 4 Гб оперативної пам’яті.
7. Рідкокристалічний монітор з діагоналлю 21".

Вище наведені характеристики являють собою рекомендовані. Це означає, що при наявності характеристик не нижче зазначених, розроблений додаток буде функціонувати відповідно до вимог щодо надійності, безпеки та швидкості обробки даних.

### 1.5.4. Вимоги інформаційної та програмної сумісності.

Для коректного функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде експлуатуватися веб-додаток, відповідало наступним вимогам:

1. Операційна система Windows 7/10/11.

Комп’ютерна гра має бути реалізована на мові програмування C# з використанням двигуна Unity3D. Для створення графіки було використано Adobe Photoshop.

**Висновки**:

Комп'ютерні ігри є надзвичайно популярними і важливими аспектами сучасної культури та розваг. Вони надають людям можливість втекти в інші світи, взаємодіяти з віртуальними персонажами та отримувати новий досвід. Отже, комп'ютерні ігри мають суттєвий вплив на розвиток технологій, креативності та соціального взаємодії.

При розробці даної гри необхідно використовувати нові технології та підходи. Це може сприяти покращенню враженню геймерів у ході прохождення гри.

# РОЗДІЛ 2
# ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 2.1. Функціональне призначення системи

У результаті зробленої кваліфікаційної роботи було отримано комп’ютерну гру для платформи Windows. Для цього не потребується інтернет-з'єднання, тому що всі потрібні файли повинні знаходитись на фізичному носії користувача.

## 2.2. Опис застосованих математичних методів

У процесі проектування та розробки цієї інформаційної системи використовувалися тільки прості арифметичні операції, без використання математичних методів.

## 2.3. Опис використаних технологій та мов програмування

1. Unity3D.

Unity3D - це ігровий движок, створений компанією Unity Technologies, який працює на різних платформах (рис. 2.1.).


Рис. 2.1. Логотип програми Unity3D.

Історія створення цього движка є захопливою та навчальною. Цікавою її робить той факт, що двоє хлопців вирішили створити власну гру, але виявили, що наявні інструменти не відповідають їх потребам. Вони вирішили самостійно розробити власний движок для створення гри. Проте, коли вони закінчили роботу над движком, виявилося, що їм більше подобається працювати над самим движком, ніж над іграми. Це спровокувало народження одного з найвідоміших та потужних ігрових двигунів. Ця історія також має важливу навчальну складову, яка нагадує, що ніколи не можна передбачити, який поворот приймуть наші починання.

Unity є прекрасним варіантом для розробки проектів середньої складності як для персональних комп'ютерів, так і для мобільних пристроїв. Велика кількість готових ресурсів, включаючи скрипти, значно полегшує процес. Крім того, наявність широкої спільноти також є великим перевагою, оскільки вона готова допомогти вирішити будь-які проблеми, які виникнуть у вас.

Якщо ви працюєте над невеликим проектом, наприклад, клікером або чимось подібним, варто розглянути, чи не буде Unity занадто великим та

масштабним для такого завдання. Рекомендую взяти до уваги більш прості движки, наприклад, LibGDX.

Одним з переваг Unity є його список підтримуваних платформ, на яких можна запускати програми. Unity працює практично на всіх платформах - на комп'ютерах (з усіма операційними системами), на Android, iOS, SmartTV, у веб-браузерах і навіть на екзотичних системах, таких як Tizen OS. Проте, є кілька підводних каменів. Якщо ви працюєте зі щось специфічним, наприклад, з низькорівневим доступом до апаратного забезпечення на Android, вам може знадобитися писати частину коду на Java і потім інтегрувати це з Unity. Те саме стосується й iOS. Крім того, збірка додатків для iOS можлива тільки на MacOS X. Це означає, що без наявності MacBook або аналогічного обладнання випуск гри для iOS може бути проблематичним. Це обмеження не залежить від Unity, а від самої компанії Apple. Такі вимоги існують, і варто пам'ятати про це, особливо якщо ви плануєте працювати з iOS.

Що стосується процесу розробки ігор, основні опції - це Windows або Mac OS X. Хоча існують експериментальні збірки редактора для Linux, вони ще мають деякі проблеми та нестабільні. Весь процес розробки гри відбувається в редакторі Unity, і ви можете редагувати код скриптів у MonoDevelop (який використовується за замовчуванням) або скористатися стороннім редактором. Багато людей використовують Visual Studio, а деякі впроваджують Sublime Text для цих цілей.

Одним з сильних аспектів Unity є його Ассети. Все в грі, включаючи код та зображення, представлено як Ассети. Ці Ассети можна експортувати та імпортувати. Це означає, що сторонні розробники можуть створювати готові компоненти для ігор. Вам лише потрібно замінити зображення та внести корективи у скрипти, і гра готова до релізу. Знову ж таки, все не так просто, адже диявол криється в деталях. Різні Ассети можуть бути несумісні між собою або не відповідати стилістично. Однак, ці деталі вже вважаються.

Unity Asset Store - це спеціальний онлайн-магазин, де можна придбати готові Ассети від сторонніх розробників. Будь-хто може створити свій власний

Ассет і розмістити його для продажу в цьому магазині. Деякі люди зуміли побудувати повноцінний бізнес, завдяки великому ринку користувачів Unity. Особливо важливою є можливість доступу до магазину прямо з редактора Unity. Це означає, що додавання нових Ассетів максимально спрощується. Ви можете перейти до магазину, клікнути на потрібний Ассет, і він негайно завантажується та додається до вашого поточного проекту. Це швидко та зручно.

Ще однією захоплюючою особливістю є активна спільнота Unity. Вона величезна. Якщо у вас є питання, ймовірно, воно вже багато разів поставлялося, і вирішувалося не один раз. Ви можете шукати відповіді на спеціалізованих форумах або на Stack Overflow. Читайте блоги людей, що розробляють ігри на цьому движку. Інформації просто багато. Якщо ж у вас все ще є питання, на яке ви не знайшли відповіді, ви можете задати його на офіційному форумі Unity, і ймовірність отримати відповідь в той же день дуже висока. Це величезний перевага движка порівняно з іншими. Ви ніколи не будете самі, завжди знайдете підтримку.

Звичайно, також є деякі недоліки. Першим з них є повільна продуктивність Unity. Порівняно з іншими двигунами, такими як LibGDX або Cocos2D-X, Unity працює повільніше. Розумію, що ці двигуни мають різні цілі та орієнтацію, і Unity спрямований на більші проекти з багатою функціональністю. Однак, факт залишається фактом. Для невеликої 2D гри, такої як платформер, Unity може працювати повільніше, ніж альтернативи. Це може не помітитись на ПК, але на мобільних пристроях це може бути відчутним.

Іншим аспектом є великий об'єм програми. Великий розмір означає, що при створенні порожнього проекту зі стандартними налаштуваннями для Android, розмір інсталяційного файлу становить близько 20 мегабайт. Для ПК ця цифра збільшується до 100 мегабайт. Для великих проектів розміром у гігабайти це не проблема, але для невеликих Android-ігор, де графіка і звуки займають лише п'ять мегабайт, додаткові 20 мегабайт можуть бути неприємним обтяженням.

Іншим недоліком, який пов'язаний з Unity, є його спрямованість на об'єкти та прикріплені до них скрипти, що може спонукати розробника до створення поганої архітектури. Додавання нових функцій здається досить простим завданням: достатньо написати скрипт і прикріпити його до об'єкта. Однак, зі зростанням проекту, зв'язки між скриптами і об'єктами ускладнюються, і додавання нових можливостей стає все складніше. Гра стає повільнішою і може виникати більше помилок. Це нагадує ситуацію з Delphi і прив'язкою обробників подій до кнопок. Знаючі люди зрозуміють. Тим, хто не знаходиться в цій темі, скажемо, що це небажано для великих додатків. Зрозуміло, що на Unity можна писати інакше, контролювати кількість скриптів і зв'язок між ними. Більш того, великі проекти так і реалізують. Однак, недосвідчені розробники часто надмірно використовують прикріплення скриптів до об'єктів, а движок не обмежує їх у цьому.

Якщо ви бажаєте створити власну гру, але не хочете докладати великих зусиль для вивчення програмування, то Unity є відмінним варіантом для вас. Ви зможете легко розміщувати об'єкти та взаємодіяти з ними, навіть не знаючи жодної мови програмування. У разі виникнення проблем велика спільнота готова вам допомогти. Але якщо ви маєте серйозні плани щодо розробки ігор, Unity також є чудовим вибором. Це потужний двигун, здатний підтримувати проекти класу AAA.

2. Microsoft Visual Studio.

Microsoft Visual Studio є набором продуктів, розроблених компанією Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та інші корисні інструменти.

В комплект входят наступні основні компоненти:

- Visual Basic.NET - для розробки додатків на VisualBasic;
- Visual C ++ - на традиційній мові C ++;
- Visual C # - на мові C # (Microsoft);
- Visual F # - на F # (Microsoft Developer Division).

Функціональна структура середовища включає в себе:

- редактор вихідного коду, який включає безліч додаткових функцій, як автодоповнення IntelliSense, рефракторінг коду і т. д .;

- відладчик коду;

- редактор форм, призначений для спрощеного конструювання графічних інтерфейсів;

- веб-редактор;

- дизайнер класів;

- дизайнер-схеми без даних.

Нище приведен інтерфейс програми Microsoft Visual Studio (рис. 2.2.):

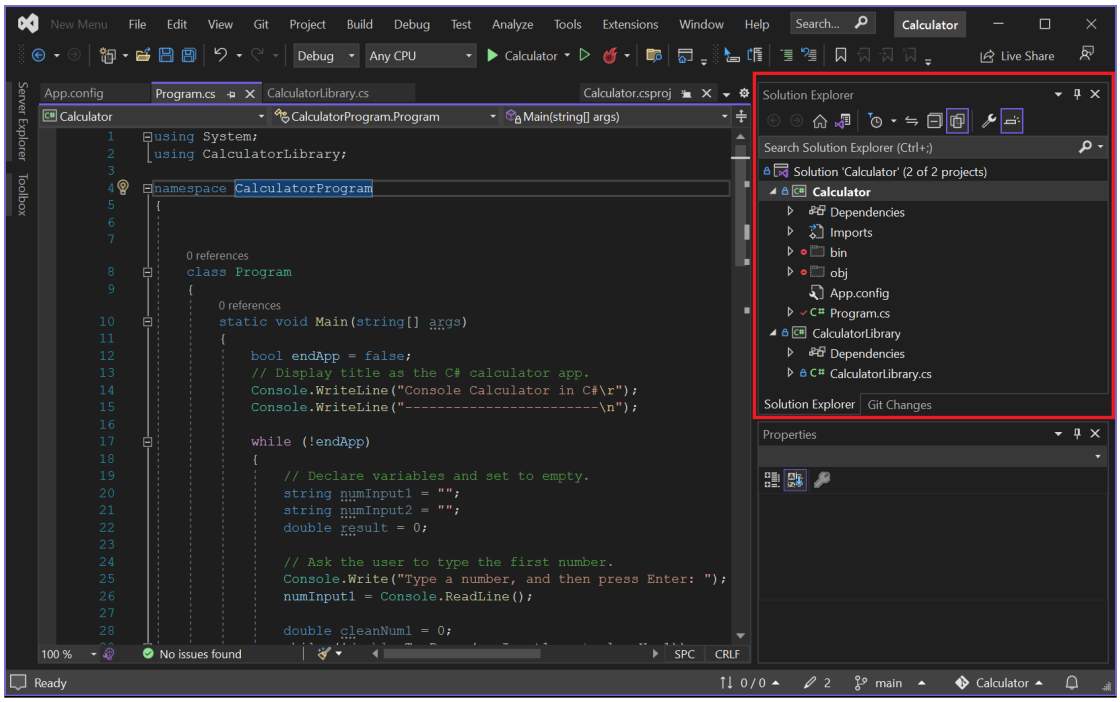


Рис. 2.2. Інтерфейс програми Microsoft Visual Studio.

Visual Studio також дозволяє розширювати свої можливості шляхом створення та підключення сторонніх додатків, відомих як плагіни. Ці плагіни дозволяють розширити функціональність Visual Studio на різних рівнях, включаючи підтримку систем контролю версій, таких як Subversion і Visual SourceSafe, а також додавання нових інструментів і наборів для редагування,

візуального проектування коду для предметно-орієнтованих мов програмування та інші аспекти процесу розробки програмного забезпечення.

Різновиди комерційних версій Visual Studio, починаючи з найбільш доступних до найбільш преміальних, включають Visual Studio Professional, Visual Studio Premium і Visual Studio Ultimate.

Інтегроване середовище розробки Visual Studio надає широкий спектр високорівневих функціональних можливостей, що виходять за рамки базового керування кодом. Нижче перераховані основні переваги, які пропонує середовище розробки Visual Studio:

- Visual Studio має вбудований веб-сервер, який є необхідним для обслуговування веб-додатків ASP.NET. Цей веб-сервер очікує веб-запити та обробляє відповідні сторінки. Наявність інтегрованого веб-сервера в Visual Studio дозволяє запускати веб-сайти прямо з середовища розробки, а також підвищує безпеку, оскільки забороняє доступ до тестового веб-сайту з зовнішніх комп'ютерів. Тестовий сервер дозволяє з'єднання тільки з локального комп'ютера, що усуває можливість отримання доступу до нього зовні.

- У Visual Studio є підтримка широкого спектру мов програмування під час розробки. Ви можете писати код на своїй рідній мові або в будь-якій іншій бажаній мові, використовуючи той самий інтерфейс розробки (IDE). Крім того, Visual Studio також дозволяє створювати веб-сторінки на різних мовах та включати їх у один веб-додаток. Єдиним обмеженням є те, що кожна веб-сторінка може використовувати лише одну мову.

- Редукція коду. Багато додатків потребують великої кількості типового коду, і це стосується також веб-сторінок ASP.NET. Наприклад, додавання веб-елемента керування, приєднання обробників подій та виконання форматування часто вимагає внесення деталей у розмітку сторінки. У Visual Studio ці деталі можна автоматично згенерувати, що дозволяє зменшити обсяг необхідного коду.

- Зручне стилізування коду. По замовчуванню Visual Studio форматує код під час введення, автоматично додаючи потрібні відступи та використовуючи кольорове виділення для коментарів і інших елементів. Ці невеликі деталі

роблять код більш зрозумілим для читання і зменшують ймовірність помилок. Крім того, Visual Studio дозволяє налаштовувати параметри автоматичного форматування коду, що особливо зручно, коли розробник вподобає інший стиль розташування фігурних дужок, наприклад, стиль K&R, де відкриваюча дужка розміщується на тому ж рядку, що й оголошення, перед яким вона стоїть.

- Збільшена продуктивність розробки. Багато з функцій Visual Studio спрямовані на прискорення процесу розробки. Зручні функції, такі як IntelliSense, яка автоматично виявляє помилки і пропонує правильні варіанти, функції пошуку і заміни, які дозволяють знаходити ключові слова в окремих файлах або в усьому проекті, а також функції автоматичного додавання і видалення коментарів, які можуть тимчасово приховувати блоки коду, допомагають розробнику працювати швидко і ефективно. Ці зручності дозволяють зосередитися на самому процесі розробки та зменшують час, який потрібен для виконання рутинних завдань.

- Можливості для налагодження. Інструменти налагодження, доступні в Visual Studio, є незамінним засобом для виявлення складних помилок та діагностування непередбачуваної поведінки. Розробник може виконувати свій код по одному рядку, встановлювати інтелектуальні точки зупинки, і, за бажанням, зберігати їх для подальшого використання, а також переглядати поточну інформацію з пам'яті в будь-який момент. Ці можливості дозволяють ефективно відстежувати проблеми в коді, спрощують процес пошуку помилок і дозволяють зосередитися на вирішенні проблем без зайвого зусилля.

Visual Studio також пропонує безліч інших функцій, як-то: можливість керування проектами, вбудовані засоби управління вихідним кодом, можливість рефакторінгу коду та потужна модель розширення. Крім того, якщо використовувати Visual Studio 2008 Team System, розробник отримує розширені можливості для модульного тестування, спільної роботи та керування версіями коду. Ці додаткові можливості дозволяють забезпечити більш гнучке та продуктивне управління проектами, поліпшують процес розробки і спільної роботи, а також забезпечують ефективну роботу з версіями коду.

Одним з недоліків є обмеженість вбудованого відладчика Visual Studio щодо відстеження режиму ядра в коді. Для відлагодження в режимі ядра в операційній системі Windows зазвичай використовуються зовнішні інструменти, такі як WinDbg, KD або SoftICE. Visual Studio не надає повної підтримки для такого режиму відлагодження.

3. C#

C#, також відома як C-sharp або сі-Шарп, є мовою програмування, яка поєднує об'єктно-орієнтовані та аспектно-орієнтовані концепції. Вона була розроблена в період з 1998 по 2001 роки групою інженерів під керівництвом Андерса Хейлсберга в компанії Microsoft як основна мова розробки додатків для платформи Microsoft .NET. Компілятор C# включений до стандартного набору інструментів самого .NET, що дозволяє створювати та компілювати програми на C# навіть без використання інших інструментальних середовищ, таких як Visual Studio (рис 2.3.).

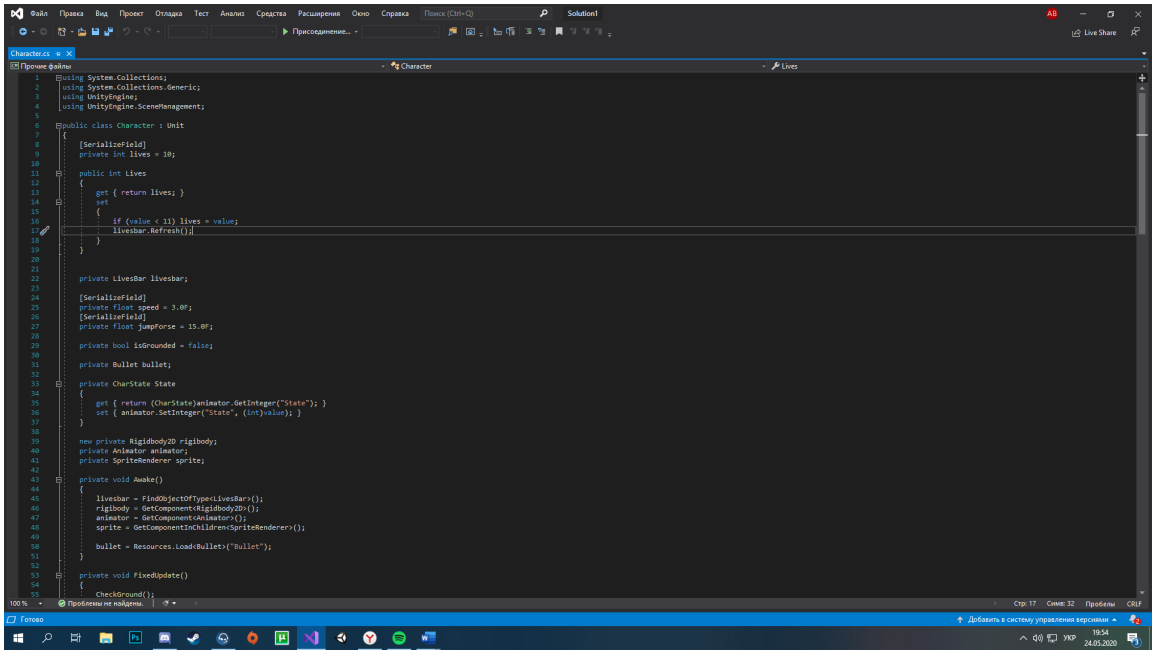


Рис. 2.3. Основний вигляд коду C#.

C# належить до сім'ї мов з C-подібним синтаксисом, і його синтаксис найбільш схожий на C++ і Java. Ця мова має строгу статичну типізацію і підтримує такі концепції, як поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки та коментарі у форматі XML. C# взяв багато речей від своїх попередників, таких як C++, Java, Delphi, Модула і Smalltalk. Враховуючи досвід їх використання, C# виключив деякі моделі, які виявилися проблематичними при розробці програмних систем, наприклад, він не підтримує множинне успадкування класів.

C# був розроблений як мова програмування на рівні додатків для CLR (Common Language Runtime) і, відповідно, він сильно залежить від можливостей, які надає сама CLR. Це особливо стосується системи типів C#, яка відображає FCL (Framework Class Library). Можливість включення або виключення певних особливостей мови визначається тим, чи можуть ці особливості бути перетворені в відповідні конструкції CLR. З розвитком CLR від версії 1.1 до 2.0, C# значно розширив свої можливості, і подібний процес очікується і в майбутньому. CLR надає C#, а також усім іншим мовам, орієнтованим на .NET, багато можливостей, від яких "класичні" мови програмування не мають. Наприклад, збірка сміття не реалізована безпосередньо в C#, але CLR виконує збірку сміття для програм, написаних на C# так само, як для програм, написаних на VB.NET, J# та інших мовах.

Символ # в назві мови можна розглядати з двох точок зору. З одного боку, він може бути сприйнятий як дві пари плюсів ++, що вказує на новий етап розвитку мови в порівнянні з C++ (аналогічно до кроку від C до C++). З іншого боку, символ # є музичним символом і, разом з буквою C, утворює назву англійського ноти "C-sharp" (до-дієз). Цей музичний аспект також вплинув на вибір назви мови. Незважаючи на те, що символ # (октоторп) фактично використовується для позначення номерів на більшості клавіатур і відрізняється від символу дієз # (Unicode U+266F), компанія Microsoft, яка є автором мови, кілька разів зверталася до своїх клієнтів з проханням прийняти це уявлення.

Перша версія мови C# була стандартизована як ECMA (стандарт ECMA-334 C# Language Specification, 3-й видання, червень 2005 року) і ISO (ISO/IEC 23270:2003, Information technology - C# Language Specification). Засновуючись на цих специфікаціях, наразі відомо, принаймні, про два незалежні реалізації C#, які перебувають на різних стадіях розробки.

В кінці 2005 року компанія Microsoft оголосила про намір зробити публічною специфікацію другої версії C#, але до цього часу ця специфікація ще не отримала статус міжнародного стандарту. Таким чином, Microsoft залишається, практично, єдиним розробником, що просуває розвиток C# на ринку.

Перший етап проекту C# був розпочатий у грудні 1998 року і мав кодову назву COOL (C-style Object Oriented Language). Версія 1.0 була анонсована разом з платформою .NET у червні 2000 року, і тоді ж була випущена перша бета-версія. Остаточна версія C# 1.0 була випущена в лютому 2002 року.

Перша версія C# мала схожі можливості з Java 1.4, але також включала кілька розширень. Наприклад, C# вводив поняття властивостей, індексаторів, подій, делегатів, циклу foreach, структур, що передаються за значенням, автоматичного перетворення вбудованих типів в об'єкти (boxing) при необхідності, атрибутів і вбудованих засобів взаємодії з некерованим кодом (DLL, COM) та інших функцій.

Крім цього, C# включив деякі можливості, які були присутні в C++, але відсутні в Java. Такі можливості включали беззнакові типи, перевизначення операцій з певними обмеженнями, передачу параметрів в методи за посиланням, методи зі змінним числом параметрів та оператор goto. Крім того, в C# існувала обмежена можливість роботи з покажчиками, яка була доступна лише в певних частинах коду, позначених ключовим словом "unsafe" та при використанні відповідної опції компілятора.

У жовтні 2003 року Microsoft вперше представила другу версію специфікації проекту C# 2.0. У 2004 році були випущені бета-версії цього

проекту під кодовою назвою "Whidbey". Остаточна версія C# 2.0 була випущена 7 листопада 2005 року разом з Visual Studio 2005 і .NET 2.0.

В червні 2004 року Андерс Хейлсберг розкрив плани щодо майбутніх розширень мови C# 3.0 на сайті Microsoft. У вересні 2005 року були опубліковані проект специфікації C# 3.0 та бета-версія C# 3.0, яка може бути встановлена як додаток до існуючих Visual Studio 2005 і .NET 2.0.

C# 3.0 включає в себе значні додатки до мови, серед яких ключові слова "select", "from" і "where", що дозволяють виконувати запити на кшталт SQL, XML та інших колекцій (цей запитовий механізм, що інтегрований у мову, відомий як Language Integrated Query або LINQ).

Ініціалізація об'єкта разом з його властивостями:

- Customer c = new Customer (); c.Name = "James".
- Customer c = new Customer {Name = "James"}.
- listOfFoo.Where (delegate (Foo x) {return x.size> 10;}).
- listOfFoo.Where (x => x.size> 10).
- Автоматичне виведення тип локальної змінної: var x = "hello".
- Безіменні типи: var x = new {Name = "James"}.

## 2.4. Опис структури системи та алгоритмів її функціонування

Щоб запустити гру потрібно відкрити файл під назвою Game_2D.exe (рис. 2.4.).

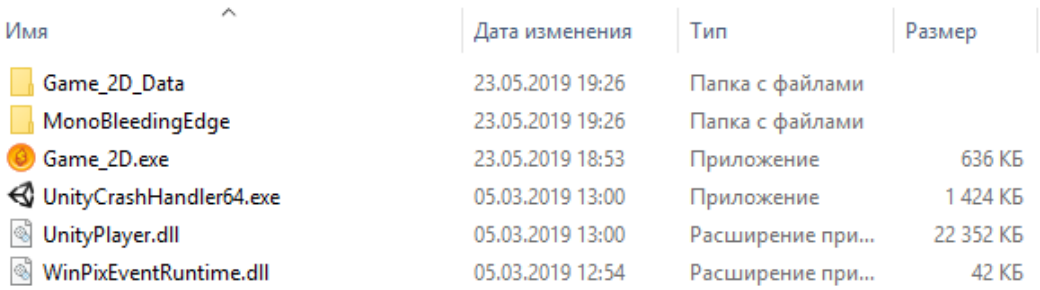


Рис. 2.4. Знаходження файлу гри у папці.

Після відкриття файлу відкриється вікно налаштувань Game_2D Configuration (рис 2.5.).

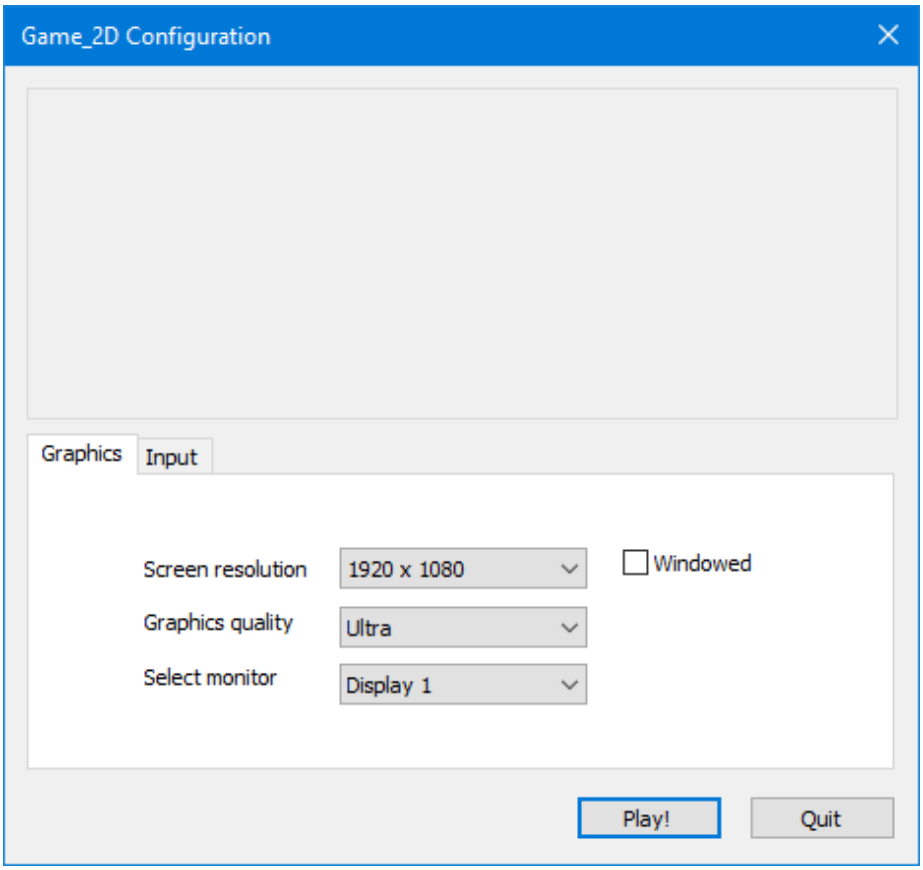


Рис. 2.5. Вікно програми налаштувань.

Зараз на рисунку відкрита вкладка Graphics яка дає можливість налаштувати роздільну здатність гри, налаштування графіки, вибрати кількість моніторів та чи буде гра у вікні або у повноекранному режимі.

У вкладці Input є можливість налаштувати клавіші керування (рис 2.6.).

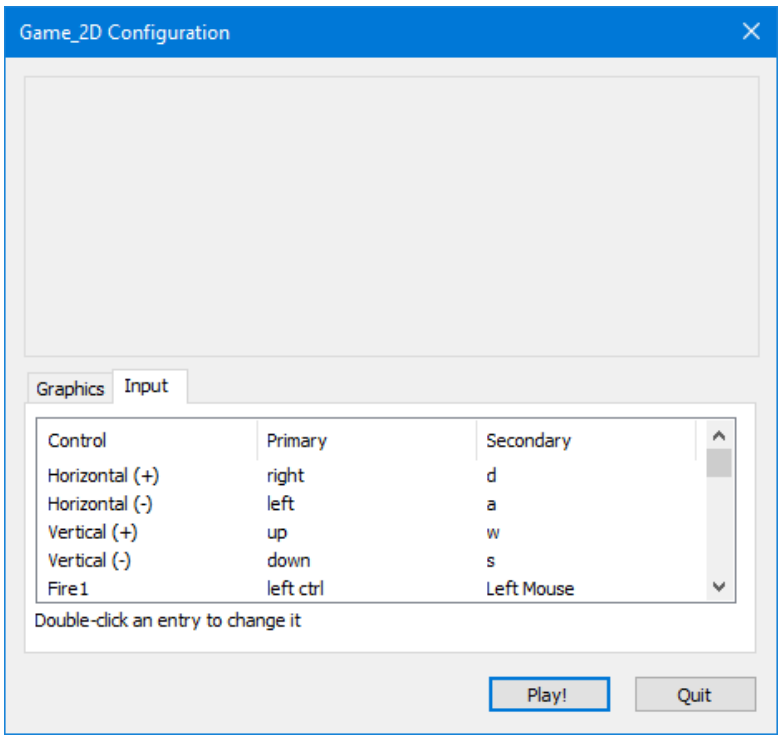


Рис. 2.6. Налаштування клавіш керування.

Щоб відкрилась гра потрібно натиснути на кнопку «Play!» (рис. 2.7.).

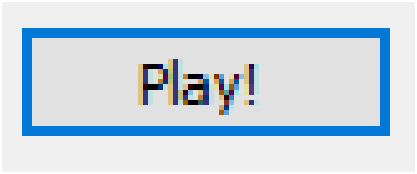


Рис. 2.7. Кнопка «Play!».

Натиснувши на кнопку «Play!» відкриється вікно з грою, але спочатку з'явиться заставка (рис 2.8.).

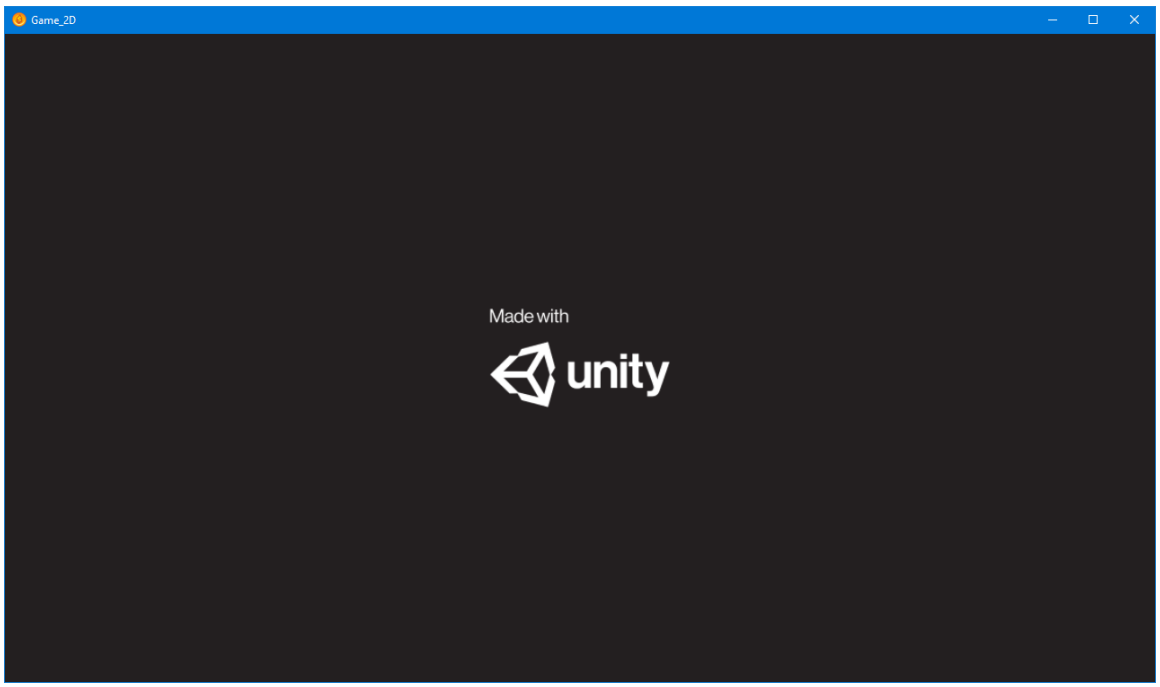


Рис. 2.8. Заставка гри.

Після цього з'явиться початкове меню гри (рис. 2.9.).

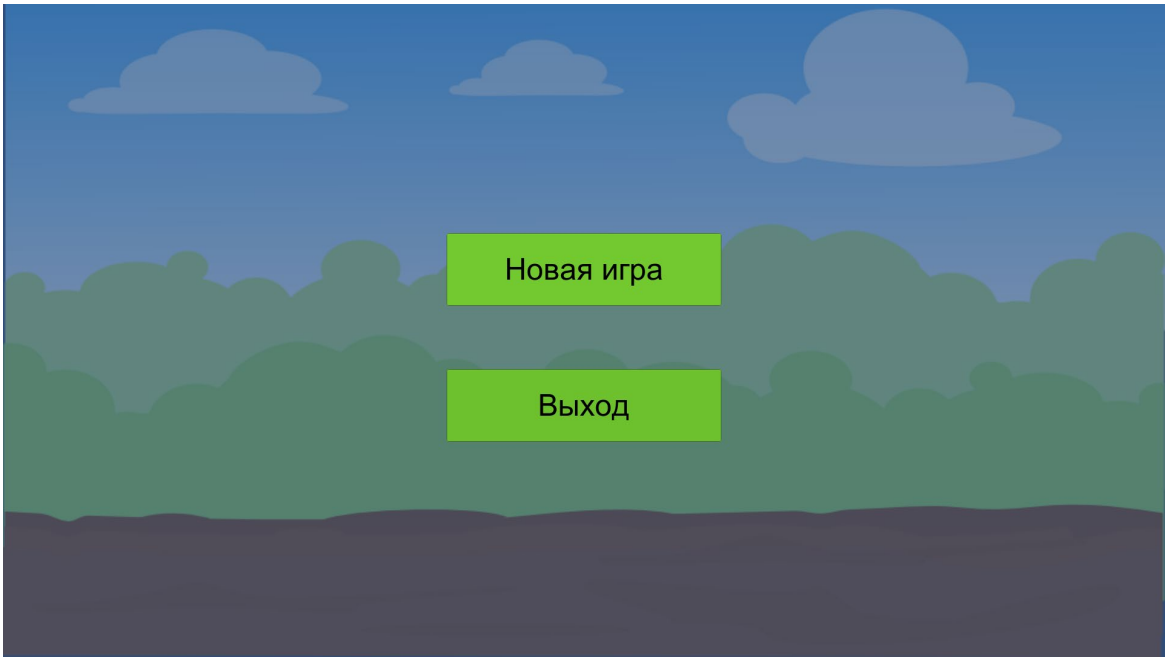


Рис. 2.9. Початкове меню гри.

Далі на екрані з'явиться сама гра (рис. 2.10.).

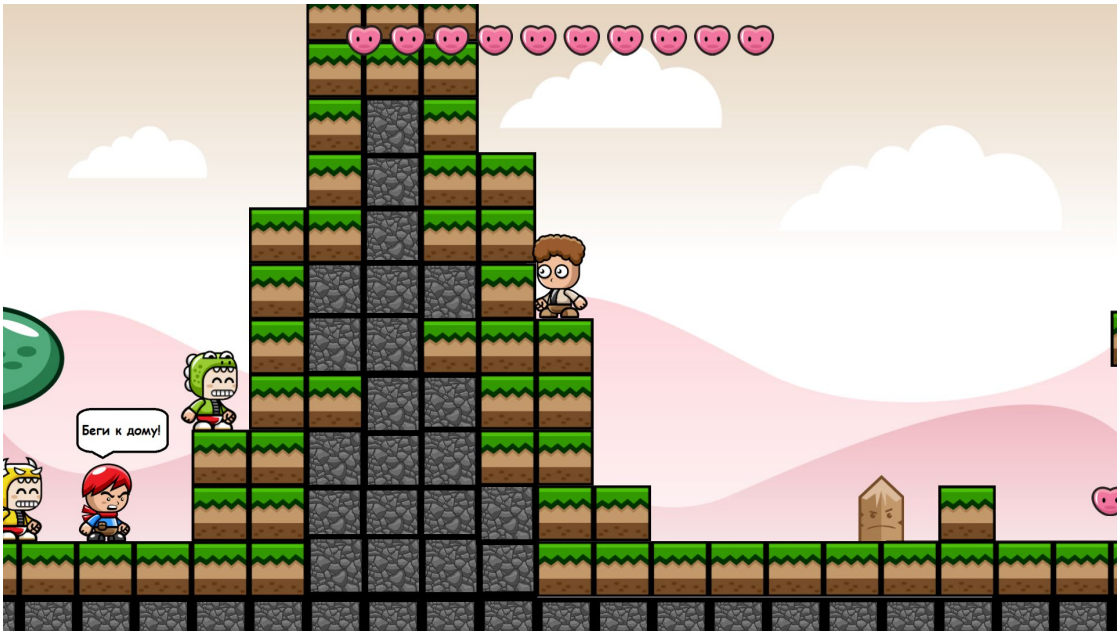


Рис. 2.10. Вікно гри.

Спочатку ми бачимо нашого персонажа. Він вміє стріляти, якщо можна так сказати. В нього куля, якою він стріляє, синього кольору (рис. 2.11.).

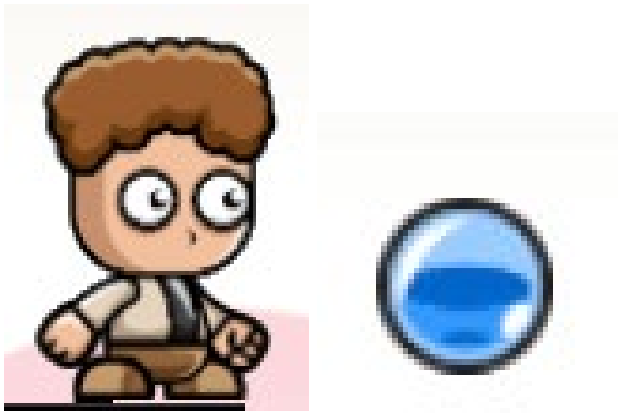

Рис. 2.11. Персонаж та його куля.

Також у грі існують вороги у вигляді монстрів, їх три та є один об’єкт який не вважається монстром, від уявляє собою перешкоду на яку краще не попадати (рис. 2.12.).

Рисунок 2.12. О’бєкт «Перешкода».

Перший ворог - це монстр, який просто стоїть. Щоб його нейтралізувати потрібно вистрілити в нього (рис. 2.13.).

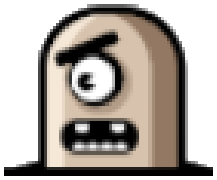

Рис. 2.13. «Монстр1».

Другий ворог - це монстр який вміє стріляти. У нього кулі, якими він стріляє зеленого кольору. Він стріляє з періодичністю півтори секунди. Щоб його нейтралізувати необхідно стрибнути на нього зверху (рис. 2.14.).

Рис. 2.14. «Монстр2» та його куля.

Третій ворог - це монстр який рухається зліва направо та навпаки якщо зустріне перешкоду у вигляді стінки. Щоб його нейтралізувати необхідно також стрибнути на нього зверху (рисунок 2.15.).

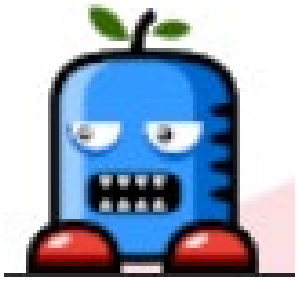

Рис. 2.15. «Монстр3».

Також у даній грі існує кількість життів, а саме десять. Але при кожному попаданні снаряду в нашого  персонажа кількість знижується на 1 одиницю. Звісно на карті присутній об’єкт завдяки якому кількість, при збиранні його, підвищіться (рис. 2.16.).

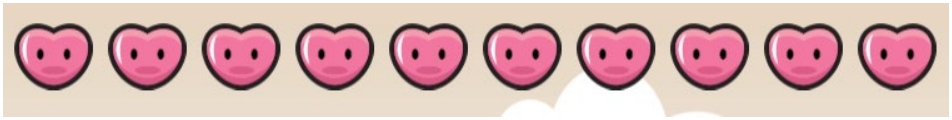

Рис. 2.16. Панель життів та об’єкт для додавання кількості життів.

На рисунку нижче показана структурна схема гри (рис 2.17.).

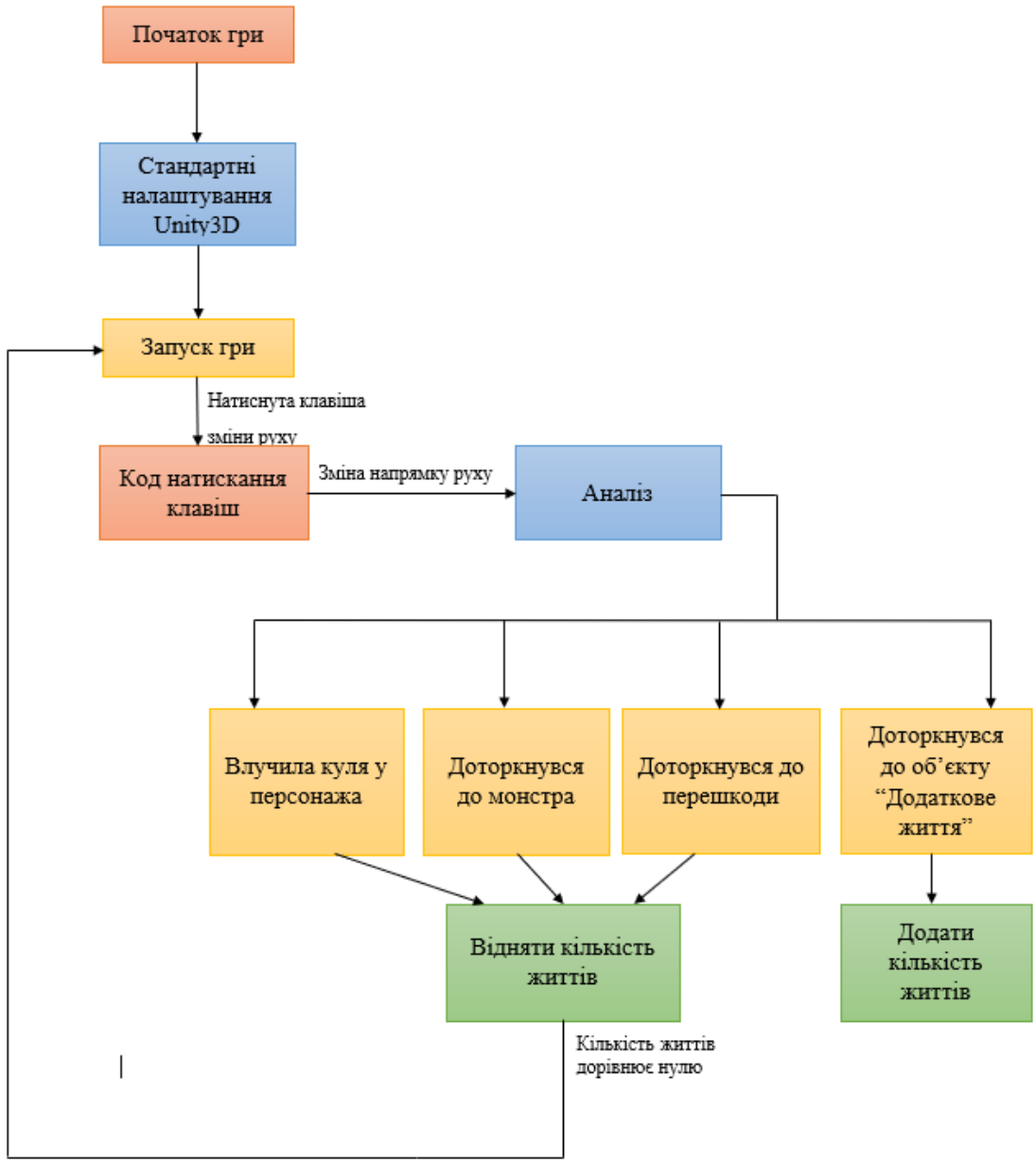


Рис. 2.17. Структурна схема гри.

## 2.5. Обґрунтування та організація вхідних та вихідних даних програми

Проаналізувавши гру я зрозумів, що якщо вона створена для персонального комп'ютера та їхніх користувачів, то мені потрібно підібрати зрозумілий інтерфейс, який буде надавати зручність при проходженні гри

користувачем. Для цього доцільно буде обрати мишку та клавіатуру, як основні пристрої для керуванням персонажем у грі.

Вхідні данні: управління у грі відбувається при використанні клавіатури та комп’ютерної миші.

Вихідні дані генеруються при роботі гри та при обробці даних які отримує від геймера в наслідок проходженні гри або просто його взаємодій.

Вихідні дані в застосунку це реакція інтерфейсу на натискання кнопок та клавіш миші. В ігрових сценах це переміщення гравця всередині сцени (рівня), вистріли, якими він наділений, або знищуючи ворогів.

## 2.6. Опис розробленої системи

Після компіляції програмного продукту ми отримуємо набір файлів, включаючи виконавчий файл з розширенням .exe, а також різноманітні інші файли та папки. Обраний формат файлу придатний лише для запуску на операційних системах Windows. Видалення будь-яких файлів з папки проекту може призвести до ненадійної роботи програми.

### 2.6.1. Використані технічні засоби

При розробці гри був використан персональний комп’ютер із такими характеристиками:

- Процесор i5-8400.
- Відеокарта Nvidia GTX 1660 TI.
- Оперативна пам'ять HyperX 16 ГБ DDR4-2666.

Тестування проводилося на таких пристроях:

1. Персональний комп’ютер із наступними характеристиками:
    - Процесор i5-10400.
    - Відеопроцесор Intel HD Graphics 630.
    - Оперативна пам'ять HyperX 8 ГБ DDR4-2400.

2. Персональний комп’ютер із наступними характеристиками:

- Процесор i3-4130.
- Відеокарта Nvidia GTX 750 TI.
- Оперативна пам'ять HyperX 8 ГБ DDR3-1600.

### 2.6.2. Використані програмні засоби

Гру було створено у Unity3D. Код був написаний в середовищі написанні коду Microsoft Visual Studio від компанії Microsoft. Коригування та створення текстур була використана програма Adobe Photoshop від компанії Adobe. Для внутрішнього тестування використовувався внутрішній емулятор з Unity.

### 2.6.3. Виклик та завантаження програми

Задля того щоб запустити створений продукт на персональному комп’ютері гру не потрібно встановлювати. Користувачеві достатньо завантажити архів з скомпільованими файлами та запустити файл з розширенням .exe від імені адміністратора.

### 2.6.4. Опис інтерфейсу користувача

Створена гра має стандартний та зрозумілий, для будь-якого геймера, інтерфейс. Весь функціонал гри легко освоюється без необхідності спеціалізованих знань у даній сфері. Крім того, для більшої ясності щодо роботи гри, геймер може ознайомитись з короткою інструкцією щодо користування.

Для того, щоб почати рухатись праворуч, потрібно натиснути на кнопку на клавіатурі “D(в)” (рис. 2.17.).


Рис. 2.17. Клавіша для керування персонажом праворуч.

Для того, щоб почати рухатись ліворуч, потрібно натиснути на кнопку на клавіатурі “А(ф)”  (рис. 2.18.).


Рис. 2.18. Клавіша для керування персонажом ліворуч.

Щоб стрибнути потрібно натиснути клавішу “SPACE” (рис. 2.19.).


Рис. 2.19. Клавіша для того, щоб персонаж стрибнув.

Для того щоб почати стріляти потрібно натиснути на “ЛКМ” (ліва кнопка миші) (рис. 2.20.).

Рис. 2.20. Клавіша для того, щоб персонаж почав стріляти.

Основні сцени створенної гри:

- меню налаштувань (рис. 2.6.);
- меню завантаження програми (рис. 2.8.);
- вікно паузи гри (рис. 2.21.);
- головне меню (рис. 2.9.);
- перший рівень (рис. 2.10.);
- сцена «Дом» яка знаходиться між першим та другим рівнями (2.22.);
- другий рівень (рис. 2.23.).

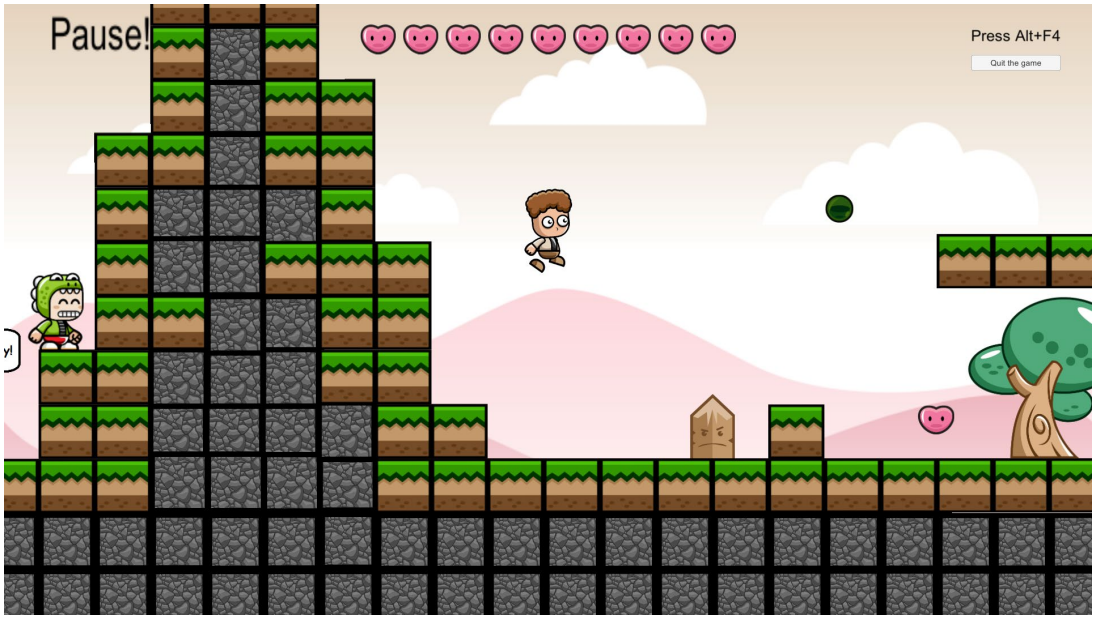


Рис. 2.21. Вікно паузи гри.

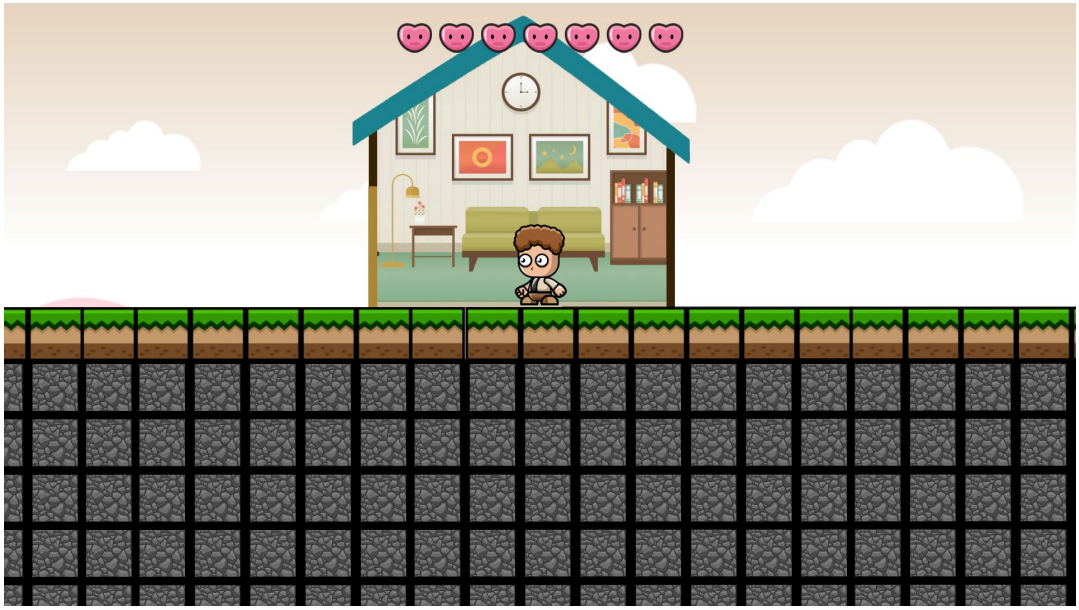

Рис. 2.22. Сцена «Дом».

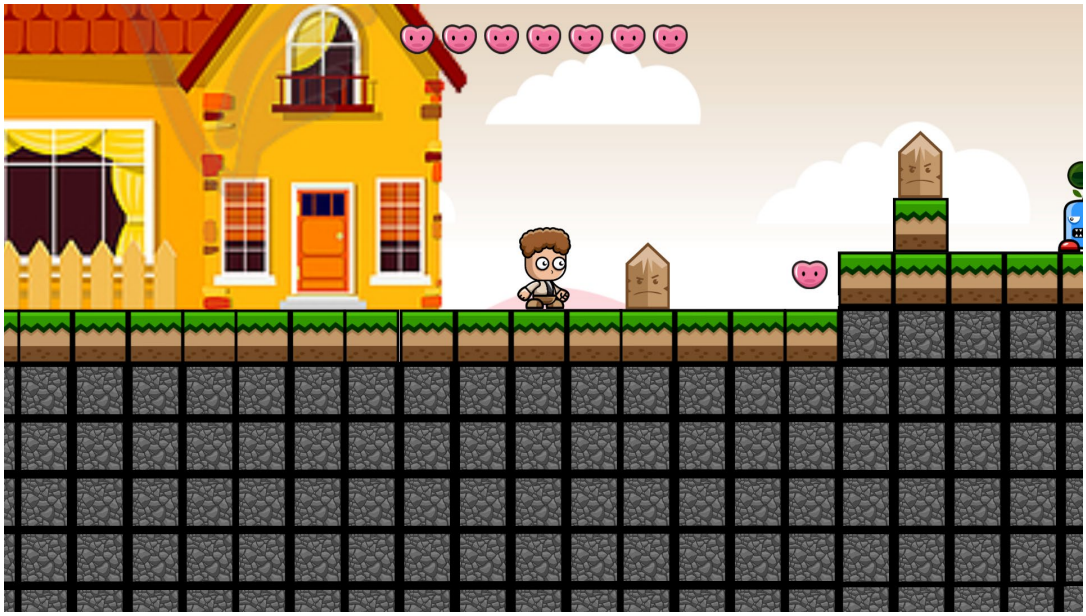

Рис. 2.23. Другий рівень.

# РОЗДІЛ 3
# ЕКОНОМІЧНИЙ РОЗДІЛ

## 3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

- передбачуване число операторів програми – 374;

- коефіцієнт складності програми – 1,3;

- коефіцієнт корекції програми в ході її розробки – 0,3;

- годинна заробітна плата програміста – 124 грн/год;

Згідно інформації наданої "Українською спільнотою програмістів (DOU)", новачок у сфері веб-розробки отримує приблизно 600 доларів США на місяць. Враховуючи офіційний курс валют Національного банку України на кінець червня 2023 року, який становить 36,57 гривень за долар США, можна зробити припущення, що середня місячна заробітна плата в гривнях складає 21 942 гривень. З урахуванням восьмигодинного робочого дня (всього 176 робочих годин на місяць), можна вважати, що середня заробітна плата за годину становить 124 гривні.

- коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,25;

- коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1;

- вартість машино-години ЕОМ – 12 грн/год.

Оскільки комп’ютер витрачає 0,6 кВт електроенергії за годину, а вартість електроенергії складає 2,64 грн за кВт/год, то витрати на електроенергію за годину складають 0,6 * 2,64 = 1.58 грн. Якщо додати інші витрати, пов'язані з роботою комп’ютера, то загальна вартість за годину становить приблизно 15 грн.

Крім витрат на комп'ютер, приміщення, комунікації та інші витрати, ціна була підвищена до рівня, який відповідає зразкам, а саме 12 гривень за годину.

У зв'язку з творчим характером роботи програміста, нормування праці в процесі створення ПЗ стає значно складнішим. Тому трудомісткість розробки ПЗ може бути визначена за допомогою системи моделей, які мають різну точність оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{\text{отл}} + t_{\text{д}}, \quad (3.1)$$

де $t_o$ – витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

$t_u$ - витрати праці на дослідження алгоритму рішення задачі;

$t_a$ - витрати праці на розробку блок-схеми алгоритму;

$t_n$ - витрати праці на програмування по готовій блок-схемі;

$t_{\text{отл}}$ - витрати праці на налагодження програми на ЕОМ;

$t_{\text{д}}$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q * C * (1 + p) = q * C * (1 + p), \quad (3.2)$$

де q – передбачуване число операторів;

С – коефіцієнт складності програми;

р – коефіцієнт кореляції програми в ході її розробки.

За формулою (3.2) умовне число операторів складає:

$$Q = 374 * 1{,}3 * (1 + 0{,}3) = 632{,}06$$

Витрати праці на вивчення опису задачі $t_u$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q*B}{(75..85)*k}, \text{ людино-годин,} \quad (3.3)$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на вивчення опису задачі за формулою (3.3) складають:

$$t_u = \frac{632,06*1,25}{80*1,1} \approx 8,98, \text{ людино-годин,}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)*k}, \text{ людино-годин,} \quad (3.4)$$

Виходячи з формули (3.4), витрати праці на розробку алгоритму рішення задачі складають:

$$t_a = \frac{632,06}{23*1,1} \approx 24,98, \text{ людино-годин,}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)*k}, \text{ людино-годин,} \quad (3.5)$$

За формулою (3.5) витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{632,06}{23*1,1} \approx 24,98, \text{ людино-годин,}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5)*k}, \text{ людино-годин,} \tag{3.6}$$

Використовуючи формулу (3.6) витрати праці на налагодження програми на ЕОМ за умови автономного налагодження одного завдання дорівнюють:

$$t_{\text{отл}} = \frac{632{,}06}{4*1{,}1} \approx 143{,}65, \text{ людино-годин,}$$

- за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = 1{,}5 * t_{\text{отл}}, \text{ людино-годин,} \tag{3.7}$$

Витрати праці на налагодження програми на ЕОМ за умови комплексного налагодження завдання за формулою (3.7) складають:

$$t_{\text{отл}}^{k} = 1{,}5 * 414{,}87 \approx 215{,}48, \text{ людино-годин,}$$

Витрати праці на підготовку документації:

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \text{ людино-годин,} \tag{3.8}$$

де $t_{\text{др}}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{\text{др}} = \frac{Q}{(15..20)*k}, \text{ людино-годин,} \tag{3.9}$$

$t_{\text{до}}$ - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 * t_{др}, \text{ людино-годин,} \quad (3.10)$$

За формулами (3.8) – (3.10) витрати праці на підготовку документації складають:

$$t_{др} = \frac{632{,}06}{18*1{,}1} \approx 31{,}92, \text{ людино-годин,}$$

$$t_{до} = 0{,}75 * 31{,}92 \approx 23{,}95, \text{ людино-годин,}$$

$$t_{д} = 31{,}92 + 23{,}95 = 55{,}87, \text{ людино-годин,}$$

Розрахувавши всі показники, за формулою (3.1) отримаємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 8{,}98 + 24{,}98 + 24{,}98 + 143{,}65 + 55{,}87 = 308{,}46,$$

людино-годин.

### 3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми *Зз/п* і витрат машинного часу, необхідного на налагодження програми на ЕОМ

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t * С_{ПР}, \text{ грн,} \quad (3.12)$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година.

Враховуючи те, що середня годинна зарплата Junior Full-Stack розробника становить 124 грн/год, за формулою (3.12) заробітна плата виконавців:

$$З_{ЗП} = 308{,}46 * 124 \approx 38\,249{,}04, \text{ грн},$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} * С_{Мч}, \text{ грн}, \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.;

$С_{Мч}$ – вартість машино-години ЕОМ, грн/год.

За формулою (3.13) вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = 143{,}65 * 12 \approx 1\,723{,}8, \text{ грн},$$

Виходячи з цього, за формулою (3.11) витрати на створення програмного забезпечення:

$$К_{ПО} = 38\,249{,}04 + 1\,723{,}8 = 39\,972{,}84, \text{ грн},$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k * F_p}, \text{ міс}, \quad (3.14)$$

де $B_k$ - число виконавців;

$F_p$ - місячний фонд робочого часу (при 40 годинному робочому тижні $F_p$ =176 годин).

За формулою (3.14) очікуваний період створення програмного забезпечення:

$$T = \frac{308{,}46}{1*176} \approx 1{,}75 \text{ міс.}$$

**Висновок**: для розробки програмного продукту знадобиться витратити 308,46 людино-годин. З урахуванням вхідних даних можна припустити, що розробка продукту займе приблизно 1,75 місяців за стандартного робочого тижня та робочого місяця. Вартість цього програмного продукту складатиме 39972,84 гривень, що включає заробітну плату фахівців, витрати на придбання та обслуговування ЕОМ.

# ВИСНОВКИ

Метою кваліфікаційної роботи було створення комп’ютерної гри, що зможе викликати в користувачів (гравців) бажання й надалі заходити та грати, проходячи рівень за рівнем до самого кінця. Робота є актуальною для всіх користувачів, які обожнюють ігри у жанрі 2D-платформер.

Ігри на даний момент часу є досить корисними, наприклад вони допомагають користувачам відволіктися від проблем які їх оточують, навчають витримці, концентрації уваги, тренують уважність тощо. 2D ігри є досить актуальними і мають широкий попит.

Програма не тільки призначена для користувачів-фанатів саме цього жанру ігор, а також для користувачів, у яких це буде перша комп’ютерна гра, та тих користувачів, які вперше отримають досвід з цим жанром.

Отже, головна мета створення даної роботи – догодити користувачам, які є фанатами ігор жанру 2D-платформер та зацікавити їхню увагу саме на цей проект.

Гра зроблена за допомогою мови програмування C#.

Під час виконання даної кваліфікаційної роботи були виконані наступні задачі:

1. Ознайомлення із мовою програмування C#.
2. Ознайомлення із двигуном Unity3D.
3. Дослідження та аналіз сучасних ігор у жанрі 2D-платформер.
4. Структурування матеріалу з даної теми.
5. Розроблення алгоритму роботи програми.
6. Створення комп’ютерної гри із зручним інтерфейсом.

Створена гра має наступні властивості:

1. Надає можливість користувачеві налаштувати гру перед її запуском.
2. Перед початком гри потрапити у головне меню.
3. Безпосередньо проходження рівнів.
4. Додано аудіо-супроводження.

4. Є можливість ставити гру на паузу.

5. У меню паузи обирати, продовжувати гру або вийти.

Створена гра надає користувачу можливість отримати позитивний досвід від процесу.

Визначено трудомісткість розробленої комп’ютерної гри (308,46 люд-год), проведений підрахунок вартості роботи по створенню гри (39972,84 грн) та розраховано час на її створення (1,75 міс).

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Можливості Adobe Photoshop: обзор версій та перевах програми. / URL: https://www.softmagazin.ru/articles/vozmozhnosti_adobe_photoshop_obzor_versiy_i_dostoinstv/;

2. Айсманн К. Маски та композиція в Photoshop. – К. : Петля, 2007. – 545 с.

3. Опис середовища розробки Microsoft Visual Studio / URL: https://studbooks.net/2258619/informatika/opisanie_sredy_razrabotki_microsoft_visual_studio;

4. Що таке Visual Studio? / URL: https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022;

5. Microsoft Visual Studio Professional OLP - Засоби розробки / URL: https://genesisua.com/VisualStudio-Professional.html;

6. МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ (Опис Unity) / URL: https://er.nau.edu.ua/bitstream/NAU/45191/1/ФККПІ_2020_122_Приймак_РС.pdf;

7. Скриптування Unity Documentations / URL: Unity - Manual: Scripting (unity3d.com);

8. Обзор ігрового двигуна Unity3D. URL: https://gamedevmania.ru/engines/unity3d;

9. Скрипти для Unity / URL: Unity - Manual: Work with sprites (unity3d.com);

10. Мова програмування C# / URL: http://bourabai.kz/alg/c-sharp.htm;

11. Стиллмен Э. Head First. Досліджуємо C#. 3-є вид. – К. : Загорний, 2017. – 797 с.

12. Хейлсберг А. Мова програмування C#. Класика Computers Science. 4-е вид. – К. : Загорний, 2012. – 773 с.

13. C# Microsoft Documentation / URL: C# docs - get started, tutorials, reference. | Microsoft Learn;

14. Методичні вказівки до виконання економічного розділу в дтпломних проектах студентів спеціальності «Комп'ютерні системи» / Уклад. О.Г.

Вагонова, О.Б. Нікітіна, Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 11 с.

15. Бойко В.В. Економіка підприємств України. Осноіний курс: Підручник для ВНЗ. – Д.: Пороги, 1997. – 312 с.

16. Бусигін А.В. Підприємство. Осноіний курс: Підручник для ВНЗ. – М.: ИНФРА-М, 1997. – 608 с.

17. Скворцов Н.Н., Бізнес-план підприємства. – К.: Вища школа, 1995. – 187с.

18. Мете А.Ф., Штец К.А., Бельгольський Б.П. ті ін. Огранізація та планування підприємств. – М.: Метталургія, 1986. – 560 с.

19. Савчук В.П., Прилипко С.И., Величко Е.Г. Аналіз та розробка інвестиційних пректів. – Навчальний посібник. – Київ: Абсолют-В. Ельга. 1999. – 304 с.

**ДОДАТОК А**

# ЛІСТИНГ ПРОГРАМИ

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class Character : Unit

{

    [SerializeField]
    private int lives = 10;
    public int Lives

    {

        get { return lives; }
        set

        {

            if (value < 11) lives = value;
            livesbar.Refresh();

        }

    }

    private LivesBar livesbar;

    [SerializeField]
    private float speed = 3.0F;

    [SerializeField]
    private float jumpForse = 15.0F;
    private bool isGrounded = false;
    private Bullet bullet;
    private CharState State

    {

        get { return (CharState)animator.GetInteger("State"); }
```

```
    set { animator.SetInteger("State", (int)value); }

}

new private Rigidbody2D rigibody;
private Animator animator;
private SpriteRenderer sprite;

private void Awake()

{

    livesbar = FindObjectOfType<LivesBar>();
    rigibody = GetComponent<Rigidbody2D>();
    animator = GetComponent<Animator>();
    sprite = GetComponentInChildren<SpriteRenderer>();
    bullet = Resources.Load<Bullet>("Bullet");

}

private void FixedUpdate()

{

    CheckGround();

}

private void Update()

{

    if(isGrounded)State = CharState.Idle;
    if (Input.GetButtonDown("Fire1")) Shoot();
    if (Input.GetButton("Horizontal")) Run();
    if (isGrounded && Input.GetButtonDown("Jump")) Jump();

}

private void Run()

{

    Vector3 direction = transform.right * Input.GetAxis("Horizontal");
```

```
        transform.position = Vector3.MoveTowards(transform.position, transform.position + direction, speed * Time.deltaTime);
        sprite.flipX = direction.x < 0.0F;
        if(isGrounded)State = CharState.Run;

    }

    private void Jump()

    {

        rigibody.AddForce(transform.up * jumpForse, ForceMode2D.Impulse);

    }

    private void Shoot()

    {

        Vector3 position = transform.position; position.y += 0.8F;
        Bullet newBullet = Instantiate(bullet, position, bullet.transform.rotation) as Bullet;
        newBullet.Parent = gameObject;
        newBullet.Direction = newBullet.transform.right * (sprite.flipX ? -1.0F : 1.0F);

    }

    public override void ReceiveDamage()

    {

        Lives--;
        rigibody.velocity = Vector3.zero;
        rigibody.AddForce(transform.up * 8.0F, ForceMode2D.Impulse);
        if (Lives <= 0)

            { SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex); }

        Debug.Log(lives);

    }

    private void CheckGround()

    {
```

```
        Collider2D[] colliders = Physics2D.OverlapCircleAll(transform.position, 0.3F);
        isGrounded = colliders.Length > 1;
        if (!isGrounded) State = CharState.Jump;

    }

    private void OnTriggerEnter2D(Collider2D collider)

    {

        Bullet bullet = collider.gameObject.GetComponent<Bullet>();
        if (bullet && bullet.Parent != gameObject)

        {

            ReceiveDamage();

        }

    }

}

public enum CharState
{

    Idle,
    Run,
    Jump

}

  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;

  public class monsters : MonoBehaviour

  {

      [SerializeField] private float speed = 2f;
      [SerializeField] private int lives = 1;
      [SerializeField] private float jumpforce = 5f;
      private bool isGrounded = false;
```

```
private Rigidbody2D rb;
private Animator anim;
private SpriteRenderer sprite;


private States State

{

   get { return (States)anim.GetInteger("state"); }
   set { anim.SetInteger("state", (int)value); }

}


private void Awake()

{

   rb = GetComponent < Rigidbody2D > ();
   anim = GetComponent<Animator>();
   sprite = GetComponentInChildren<SpriteRenderer>();

}

private void FixedUpdate()

{

   CheckGround();

}

private void Update()

{

   if (isGrounded) State = States.idle;
   if (Input.GetButton("Horizontal"))
      Run();
   if (isGrounded && Input.GetButtonDown("Jump"))
      Jump();

}
```

```
        private void Run()

        {

            if (isGrounded) State = States.run;
            Vector3 dir = transform.right * Input.GetAxis("Horizontal");
            transform.position = Vector3.MoveTowards(transform.position, transform.position +
dir, speed * Time.deltaTime);
            sprite.flipX = dir.x < 0.0f;

        }

        private void Jump()

        {

            rb.AddForce(transform.up * jumpforce, ForceMode2D.Impulse);

        }

        private void CheckGround()

        {

            Collider2D[] collider = Physics2D.OverlapCircleAll(transform.position, 0.3f);
            isGrounded = collider.Length > 1;

            if (!isGrounded) State = States.jump;

        }

    }

    public enum States

    {

        idle,
        run,
        jump

    }

    public class CameraController : MonoBehaviour
```

```
{

    [SerializeField] private Transform bad;
    private Vector3 pos;

    private void Awake()

    {

        if (!bad)
            bad = FindObjectOfType<monster>().transform;

    }

    private void Update()

    {

        pos = bad.position;

        transform.position = Vector3.Lerp(transform.position, pos, Time.deltaTime);

    }

}
```

**ДОДАТОК Б**

**ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ**

**ДОДАТОК В**

# ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

| **Ім’я файла** | **Опис** |
|---|---|
| Пояснювальні документи | |
| Кваліфікаційна_робота_Волошин_А_Д.doc | Пояснювальна записка до кваліфікаційної роботи. Документ Word. |
| Кваліфікаційна_робота_Волошин_А_Д.pdf | Пояснювальна записка до кваліфікаційної роботи в форматі PDF. |
| Програма | |
| Game_2D.zip | Архів. Містить откомпільовану програму. |
| Презентація | |
| Презентація.ppt | Презентація кваліфікаційної роботи. |