Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Бакалаврська дипломна робота на тему:

«КОМП'ЮТЕРНА ГРА PAC-MAN З УДОСКОНАЛЕНИМ ІНТЕЛЕКТОМ NPC»

Виконав: студент 4 курсу, групи 3КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Костюк І.А .
(прізвище та ініціали)
Керівник: асистент кафедри КН
Малініч І. П.
(прізвище та ініціали)
«07» 06 2024 р.
Рецензент: к.т.н., доц. каф. САІТ
Крижановський Є. М.
(прізвище та ініціали)
«07» 06 2024 р..

Допущено до захисту
Зав. кафедри Яровий А. А.
«07» 06 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«07» 03 2024 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТА
Костюка Ілля Андрійовича

1. Тема роботи: Комп'терна гра «Pac-Man» з удосконаленим інтелектом NPC.

2. Керівник роботи: Малініч Ілля Павлович, асистент.
затверджені наказом вищого навчального закладу «11» 03 2024 року №80

3. Термін подання студентом роботи «27» 05 2024 р.

4. Вихідні дані до роботи:

Кількість класів: 6;
Кількість користувачів: 1;
Інтерфейс користувача має бути інтуїтивно зрозумілий.

5. Зміст текстової частини (перелік питань, які потрібно розробити): вступ; аналіз предметної області та відомих реалізацій системи гри «Pac-Men», об'єктно-орієнтоване проектування системи гри «Pac-Men», програмна реалізація та тестування системи гри «Pac-Men», висновки, список використаних джерел.

6. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): зображення прототипу інтерфейсу користувача гри "Pac-Man", розробка загальної структурної схеми функціонування системи гри «Pac-Man», основна структурна форма програми, діаграма активності, діаграма класів, діаграма станів, основна схема функціонування гри "Pac-Man", основна схема функціонування привидів у грі "Pac-Man", вікно середовища розробки Visual Studio Code, головне

меню, очікування дій, ігровий процес, механіка зляканих привидів, завершення ефекту енерджайзера.

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	Завдання прийняв
1-3	Малініч І. П., асистент каф. КН		

8. Дата видачі завдання «07» 03 2024 р

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примі-тка
		початок	закінчення	
1	Обгрунтування доцільності створення гри Pac-Man з вдосконаленими алгоритмами інтелекту NPC	06.05.2024	11.05.2024	
2	Аналіз існуючих рішень	12.05.2024	20.05.2024	
3	Проектування алгоритмів інтелекту NPC	21.05.2024	26.05.2024	
4	Реалізація модулю вдосконалення алгоритмів інтелекту NPC	27.05.2024	29.05.2024	
5	Тестування гри після вдосконалення алгоритмів інтелекту NPC	30.05.2024	03.06.2024	
6	Оформлення матеріалів до захисту БДР	04.06.2024	06.06.2024	

Студент _____ (підпис) **Костюк І. А.** (прізвище та ініціали)

Керівник роботи _____ (підпис) **Малініч І. П.** (прізвище та ініціали)

АНОТАЦІЯ

Робота складається із 100 сторінок формату А4, які включать 57 малюнків і 4 таблиць. У списку використаних джерел зазначено 10 найменувань.

В даній бакалаврській роботі було розроблено ком’ютерну гру «Pac Man» з удосконаленими алгоритмами інтелекту NPC. Програмна реалізація дозволяє користувачу грати проти інтелектуального ігрового бота, алгоритми інтелекту якого – були вдосконалені.

Під час виконання бакалаврської дипломної роботи було використано об’єктно-орієнтовану мову програмування Python. Програмна реалізація створена, відлагоджена та протестована засобами інтегрованого середовища розробки Microsoft Visual Code. Також було використано A* алгоритм.

Ключові слова: гра, Pac-Men, NPC, неігрові персонажі.

ABSTRACT

The work consists of 100 pages of A4 format, which will include 57 figures and 4 tables. There are 10 names in the list of used sources.

In this bachelor's work, the computer game "Pac Man" was developed with advanced NPC intelligence algorithms. The software implementation allows the user to play against an intelligent game bot whose intelligence algorithms have been improved.

The object-oriented programming language Python was used during the execution of the bachelor thesis. The software implementation was created, debugged and tested using Microsoft Visual Code integrated development environment. The A* algorithm was also used.

Keywords: game, Pac-Men, NPC, non-game characers.

ЗМІСТ

ВСТУП

Розробка комп'ютерних ігор — одна з найактивніше розвиваючих сфер IT-індустрії. Класичні ігри, такі як Pac-Man, залишаються популярними серед гравців різних вікових категорій завдяки своїй простоті та захоплюючому ігровому процесу. Проте, з розвитком технологій з'являється можливість вдосконалення цих ігор, зокрема шляхом покращення алгоритмів інтелекту NPC (неігрових персонажів). Це дозволяє зробити ігровий процес більш динамічним і складнішим викликом для гравців.

З недавніх пір словосполучення «комп'ютерна гра» міцно увійшло в життя. Кожен, хто має комп'ютер, напевно, зміг відчути їх привабливість. Ймовірно, гра закладена в саму природу людини з найдавніших часів. Комп'ютерна гра - комп'ютерна програма або частина комп'ютерної програми, що служить для організації ігрового процесу, зв'язку з партнерами по грі, або сама виступає в якості партнера.

Комп'ютерні ігри класифікуються за такими ознаками як жанр, кількість гравців і спосіб їх взаємодії, візуальне уявлення і платформа. Всі комп’ютерні ігри відносяться до розряду розважальних, які не вирішують виробничі, облікові та інші завдання, це визначає особливості їх функціонального набору, алгоритму роботи і інтерфейсу. Особливість комп'ютерних ігор полягає в наявності складних алгоритмів, які визначають хід гри в залежності від дій користувача (гравця), при цьому, найчастіше, такі дії передбачити неможливо. Тому алгоритм повинен передбачати обробку якомога більшого набору варіантів дій, що значно їх ускладнює. Крім того, з огляду на розважальний характер комп'ютерних ігор, їх інтерфейс повинен володіти інтерактивністю і складними елементами графіки .

Для виконання бакалаврської дипломної роботи було обрано гру з жанру «Ігри з лабіринтом». Цікавість ігор з лабіринтами для людини в тому, що гра перед нею ставить задачу, розв'язок якої можна знайти за допомогою

логічного аналізу, продумування на кілька кроків уперед, і успішне розв'язання задачі приносить людині задоволення.

Мета бакалаврської роботи полягає у розширенні алгоритмів інтелекту для NPC в грі Pac-Man, використовуючи об’єктно-орієнтований підхід і мову програмування Python в середовищі розробки Visual Studio Code. щоб зробити гру більш захоплюючою.

Об’єктом є сама гра Pac-Man, а також процес створення та впровадження покращених алгоритмів інтелекту NPC, який використовуються в ній.

Предметом дослідження є розробка та випробування алгоритмів та методів інтелектуального керування неігровими персонажами, які використовуються в грі Pac-Man.

Такі завдання плануються для подальшої роботи над бакалаврською дипломною роботою:

- Обгрунтування доцільності створення гри Pac-Man з вдосконаленими алгоритмами інтелекту NPC.
- Аналіз існуючих рішень.
- Проектування алгоритмів інтелекту NPC.
- Реалізація модулю вдосконалення алгоритмів інтелекту NPC.
- Тестування гри після вдосконалення алгоритмів інтелекту NPC.
- Оформлення матеріалів до захисту БДР.

Покращення алогритмів інтелекту NPC Pac-Man має велике практичне значення для інших розробників ігор. Використовувані вдосконалені алгоритми інтелекту можна адаптувати для створення більш складних інтерактивних персонажів у різних ігрових жанрах, що дозволяє покращити ігровий процес, більш реалістично симулювати поведінку людини, а також як навчальний інструмент для дослідження алгоритмів інтелекту поведінки та сприяє технологічному та творчому прогресу в ігровій індустрії.

Також, результати роботи планується впровадити у Вінницькому національному технічному університеті.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВІДОМИХ РЕАЛІЗАЦІЙ СИСТЕМИ ГРИ “PAC-MAN”

1.1 Обґрунтування доцільності

Метою даної бакалаврської дипломної роботи є розширення функцій алгоритмів інтелекту NPC у грі «Pac-Man». Вона відноситься до логічних ігор, які допомагають розвинути реакцію та тренують логіку.

Іноді при тривалій роботі за комп'ютером виникає бажання трохи відпочити. Для цієї мети не підходять ігри типу стратегій, бо для них потрібно багато часу, а для 5-10 хвилинного відпочинку цілком підійде «Pac-Man». Pac-Man у цій грі — кругла жовта істота з ротом. Завдання гравця — керуючи Пакменом зібрати всі білі крапки на рівні, уникаючи зіткнень з привидами. Рівень закінчується, коли з'їдені всі крапки, після чого починається наступний, складніший. Також на рівні можуть з'являтися різні бонуси — фрукти, з'ївши які персонаж отримує додаткові бали.

До ігор такого типу пред’являються такі вимоги: використання простих засобів управління, зручний графічний інтерфейс, поступове ускладнення гри при досягненні певної кількості очок. Під час виконання програма повинна виконуватися коректно і не приводити до збоїв. Усі або майже всі великі ігрові об’єкти створюються випадково. Потім ця випадковість (рано чи пізно) стає світовим надбанням і змінює життя багатьох людей. «Pac- Man» - один з таких випадків.

Нескладна логічна головоломка, що була написана і згодом модифікована для себе і своїх колег, за короткий термін здобула світову популярність, спровокував великий скандал, низку судових розглядів і, у кінцевому рахунку, залишилася в історії як найпопулярніша комп'ютерна гра всіх часів.

Ідея гри Pac-Man зародилася в далекому 1980, коли більшість американських ігор були космічними шутерами, такими як Space Invaders або Defender. Pac –Man, же пропонував зовсім новий стиль гри, яка не

передбачає насильства. Тому, гра позиціонувалася як орієнтована на хлопчиків та дівчаток.[1].

Підвищення інтелекту NPC в іграх Pac-Man важливо з кількох важливих причин. По-перше, сам геймплей покращено завдяки більш складним NPC, що робить гру більш складною та цікавою для гравця. Вони змушені розробляти нові стратегії та шукати ефективні способи перемоги, що збагачує досвід і додає грі більше азарту та веселощів.

Підвищення інтелекту NPC також сприяє більш реалістичному представленню внутрішньої логіки гри. Складніші персонажі можуть приймати рішення, ближчі до того, як грають справжні гравці. Це не тільки підвищує рівень занурення, але й робить гру більш привабливою для тих, хто шукає реалістичний інтерактивний досвід.

Крім того, взаємодія з розумними NPC допомагає гравцеві вчитися. Ви можете спостерігати за такими персонажами, вчитися на їхніх діях і вдосконалювати власні навички гри. Це особливо важливо для тих, хто цікавиться стратегічними аспектами гри, оскільки дозволяє спробувати різні підходи та розробити найкращу стратегію успіху.

Таким чином, покращення алогритмів інтелекту NPC Pac-Man не тільки покращує реалістичність самої гри та симуляції, але також стимулює особистий ігровий розвиток гравців, роблячи це важливим фактором в ігровій індустрії.

Гра реалізована практично на всіх сучасних пристроях, включаючи КПК, мобільні телефони, ігрові відеоприставки, телевізори (як додаткова функція) і безліч кишенькових ігрових пристроїв. Є варіанти гри для всіх поширених ОС. Важко, якщо взагалі можливо, назвати таку обчислювальну платформу, де б не було цієї гри.

Гра була розроблена в основному працівником компанії Namco Тору Іватані протягом вісімнадцяти місяців. Гра була розроблена на основі мало відомого Coin Hunter. Згодом вийшла модифікація цієї гри, що називалася pakku-man (яп. パックマン). Ця назва походить від японського

описового вислову «паку-паку таберу» (яп. パクパク食べる, трансліт. «їсти, відкушувати, багаторазово широко розкриваючи рот та закриваючи його», їсти, плямкаючи). Часто згадується, що до Іватани прийшло натхнення від шматочка піци. В інтерв'ю 1986 року він зазначив, що це було наполовину правдою: Іватані округлив ієрогліф куті (яп. 口 «рот») як символ поїдання і надав йому подоби рота. Іватані спробував зробити гру цікавою для більш широких верств населення, крім звичайних підлітків, тому він додав в гру елемент лабіринту. В результаті гра отримала назву PUCK MAN.[1].

Перший аркадний автомат із грою було встановлено 22 травня 1980 року у кінотеатрі району Сібуя, Токіо. У нині неіснуючій восьмиповерховій будівлі з кількома кінотеатрами автомат був розташований у холі на одному з верхніх поверхів.

Однак у наступному році за гру взялася компанія Midway з США. Гра стала називатися Pac-Man через те, що назва PUCK MAN було занадто близько до лайки (англ. FUCK), і вандали могли легко його змінити, що погубило б гру на корені. Проте в Європі можна знайти як автомати Pac-Man, так і PUCK MAN.

Американська публіка добре зустріла гру, яка дала альтернативу Space Invaders, що виразилося в безпрецедентній популярності і доході, котрі перевершили успіх свого попередника. Навіть Іватани був здивований продажами в США. Гра незабаром стала загальносвітовим явищем у відеоіндустрії, були випущені численні продовження, стиль гри часто копіювався, але ніхто з клонів не міг перевершити оригінал.

Коли Midway випустила Pac-Man в США, компанія також змінила дизайн ігрових автоматів. Стиль Namco був дорожчий і менше підходив американському ринку. Автомат PUCK MAN був пофарбований у білий колір, на ньому були різнокольорові малюнки. Pac-Man був пофарбований у жовтий колір, малюнки були простими і легко впізнаваними. У грі метою було пройти всі 255 рівнів, набравши якомога більший рахунок. Максимальну кількість очок було зареєстровано 3 липня

1999 в Голлівуді: гравець Біллі Мітчел набрав 3333360 очок. Для цього йому знадобилося 6 годин. Він зібрав усі точки, всі енерджайзери, всі фрукти, з'їв всіх привидів на всіх 255 рівнях.[1].

Виробники відеоігор були вражені успіхом цієї відеогри. Її популярність перевищила рейтинг Asteroids, аркади з гігантськими на свій час продажами. Автомати Pac-Man розійшлися в кількості 350 000 екземплярів. Гра була настільки популярна, що на її основі народилося безліч підробок на початку 80- х. Унікальний ігровий дизайн змусив виробників ігор переглянути тему нескінченних інопланетних навал, присутню майже в кожній відеогрі того часу.

Введений в Pac-Man елемент гумору дозволив грі відповідати інтересам більш широких верств населення. Інтерес підлітків до Pac-Man перевищив інтерес до всіх тодішніх шутерів.

Багато популярних відеоігор 80-х в тій чи іншій мірі зобов'язані своїм існуванням Pac-Man: Q * Bert, Donkey Kong, Frogger. Сайт The Killer List of Videogames вважає Pac-Man грою № 1 всіх часів у списку 10 найпопулярніших відеоігор.[1] .

Отже, на перший погляд проста гра в свій час зробила революцію на ринку комп’ютерних ігор. Вона надала поштовху для розвитку ігрової індустрії та проявила інтерес людей до програмування.

Для покращення ігрового процесу в даній роботі буде змінено класичний алгоритм рухів привидів, але буде відтворено класичний інтерфейс.

1.2 Аналіз існуючих аналогів

З метою вдосконалення гри «Pac-Man» був проведений комплексний аналіз існуючих аналогів, який включав дослідження, тестування та оцінку різних версій гри. Це дослідження дозволило виявити найкращі практики та інноваційні рішення, які використовуються в сучасних відеоіграх, і визначити можливі напрямки для покращення ігрового процесу.

Аналіз розпочався з детального огляду історії розробки різних версій Pac-Man, щоб зрозуміти, як змінювалася ігрова механіка та додавалися нові елементи з часом. Було проводено порівняння таких ключових характеристик, як дизайн лабіринту, поведінка NPC, графіка та інші аспекти, які впливають на ігровий досвід.

Також, було проведено методологію дослідження:

- Пошук літератури та джерелознавство: Проведено детальний аналіз наукових праць, книг та інтернет-ресурсів, що описують розвиток гри «Pac-Man» та її аналогів. Це включало дослідження як технічних аспектів, так і культурного впливу гри на гравців різного віку.

- Тестування ігрового процесу: Встановлення та тестування різних версій "Pac-Man" дозволило отримати безпосереднє розуміння механік гри та ідентифікувати сильні та слабкі сторони кожної з них. Це також включало аналіз користувацького інтерфейсу та інтерактивності.

- Проведення опитувань та збору відгуків: Було проведено опитування серед гравців для збору їхніх вражень та рекомендацій щодо покращення гри. Відгуки гравців допомогли зрозуміти, які аспекти гри є найбільш привабливими, а які потребують вдосконалення.

- Порівняльний аналіз: Здійснено порівняння між оригінальною версією "Pac-Man" та її аналогами, щоб виявити основні відмінності та інновації. Це включало порівняння ігрових механік, складності рівнів, графічного оформлення та поведінки NPC.

Було проведено ґрунтовне тестування та дослідження таких ігрових проєктів, як "Ms. Pac-Man", "Pac-Man Championship Edition", "Pac-Man Championship Edition DX", "Pac-Man 256" та "Pac-Man World". Аналіз цих версій дозволив виявити унікальні ігрові механіки та інноваційні рішення, які зробили їх популярними серед гравців. Дослідження також включало порівняння різних підходів до реалізації NPC (неграних персонажів), що забезпечило глибоке розуміння ефективності алгоритмів інтелекту та можливостей їхнього вдосконалення:

Рисунок 1.1 – Гра «Ms. Pac-Man»

"Ms. Pac-Man" – це відеогра, випущена в 1982 році компанією Midway Manufacturing як спін-офф оригінальної гри Pac-Man. Гра була розроблена з метою вдосконалення ігрового процесу та додавання нових елементів, які б зробили гру ще більш захоплюючою та викликовою для гравців. У "Ms. Pac-Man" додано нові лабіринти з різними дизайнами та більш складними маршрутами для переслідування. Важливою особливістю цієї версії є поліпшена поведінка привидів, які тепер мають більш непередбачувані траєкторії руху, що значно підвищує рівень складності гри. Також, графічні покращення та яскравіші кольори зробили гру більш привабливою візуально, що сприяло її популярності серед гравців.

Крім того, "Ms. Pac-Man" стала першою грою в серії, де головним героєм стала жіноча персона, що позитивно вплинуло на залучення більш широкої аудиторії, включаючи жінок. Гра також представила нові види фруктів і бонусів, які з'являються на екрані, що додало грі додатковий рівень стратегії та тактики. Завдяки своїм інноваціям та вдосконаленням, "Ms. Pac-Man" отримала визнання критиків і стала однією з найуспішніших аркадних ігор свого часу, зберігаючи популярність серед гравців і до сьогодні.

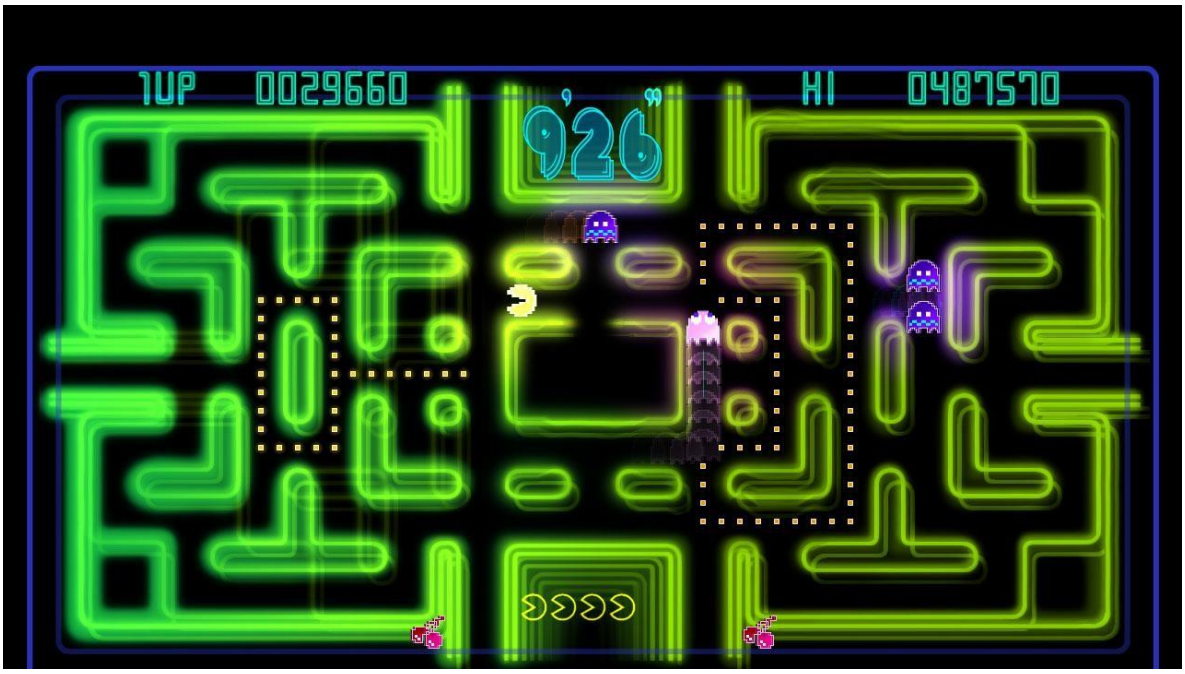

Рисунок 1.2 – Гра «Pac-Man Championship Edition»

"Pac-Man Championship Edition" – це модернізована версія класичної гри Pac-Man, випущена Namco Bandai у 2007 році для PlayStation Portable і Xbox 360. Ця версія зосереджена на швидкості та конкурентоспроможності, надаючи гравцям можливість змагатися за вищі бали в оновлених лабіринтах із динамічними елементами. Однією з головних переваг "Pac-Man Championship Edition" є її швидкий темп, який вимагає від гравців швидких реакцій і стратегічного мислення. Гра також має нові механіки, такі як зміна лабіринтів у реальному часі та збільшення швидкості руху Pac-Man і привидів, що робить ігровий процес більш захоплюючим і напруженим.

Однак, попри численні вдосконалення, "Pac-Man Championship Edition" має свої недоліки. Зокрема, висока швидкість гри може бути надто складною для новачків, що може відлякати деяких гравців. Також, орієнтація на швидкий темп і змагальний елемент може зменшити привабливість гри для тих, хто шукає більш спокійний і розслабляючий ігровий досвід. Незважаючи на ці недоліки, "Pac-Man Championship Edition" залишається важливим оновленням класичної гри, пропонуючи нові виклики і можливості для шанувальників оригінального Pac-Man.

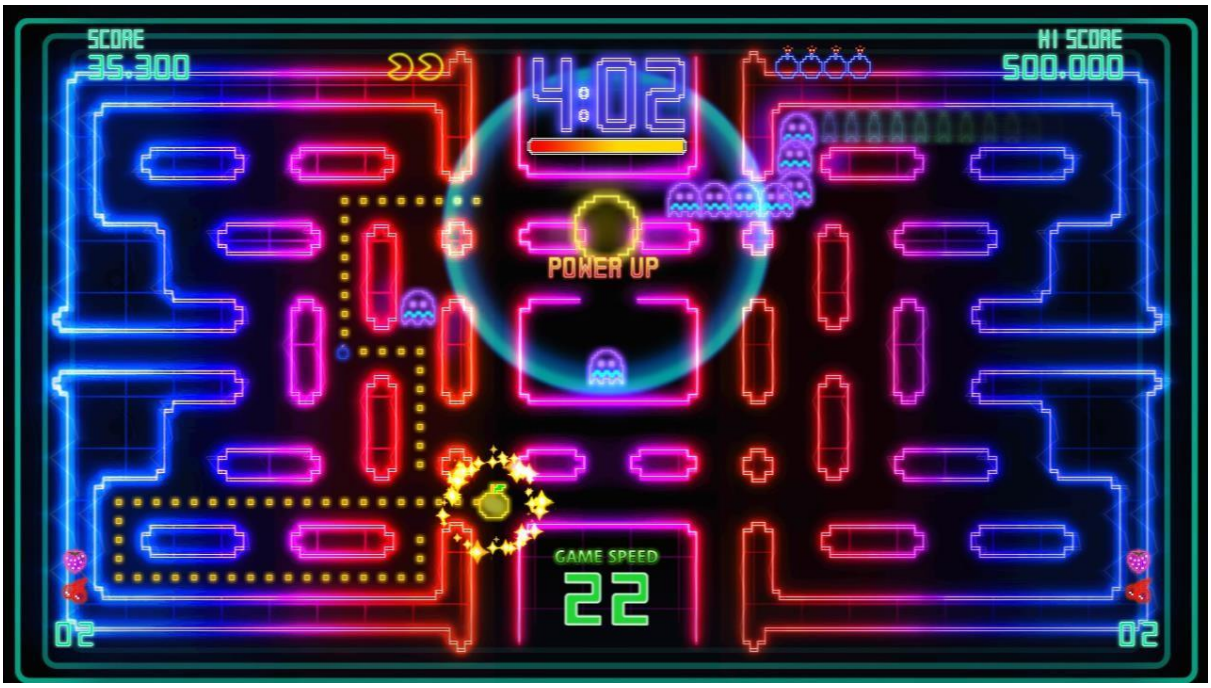

Рисунок 1.3 – Гра «Pac-Man Championship Edition DX»

Pac-Man Championship Edition DX — це продовження «Pac-Man Championship Edition», випущеного Bandai Namco у 2010 році. Ця гра пропонує більше динаміки та інтенсивності, нові функції та покращене керування. Однією з головних переваг Pac-Man Championship Edition DX є оновлений ігровий процес, включаючи різноманітні лабіринти, спеціальні бонуси та можливість налаштувати гру відповідно до особистих уподобань гравця. Гра також представляє нові типи привидів і режими гри, значно

розширюючи ігровий процес і додаючи нові стратегії для досягнення високих результатів.

Проте, як і в будь-якій грі, у "Pac-Man Championship Edition DX" є свої недоліки. Наприклад, висока інтенсивність гри може бути занадто складною для новачків, що може призвести до втрати інтересу у деяких гравців. Крім того, орієнтація на швидкі рефлекси і стратегічне мислення може бути занадто вимогливою для тих, хто шукає більш розслабляючий ігровий досвід. Незважаючи на ці недоліки, "Pac-Man Championship Edition DX" залишається визначною грою, яка пропонує гравцям унікальні виклики та захоплюючий ігровий процес, що робить її вартою уваги серед шанувальників Pac-Man.

Рисунок 1.4 – Гра «Pac-256»

Pac-Man 256 – це новітня версія легендарної гри, розроблена у 2015 році Hipster Whale у співпраці з Namco Bandai. Гра отримала свою назву від знаменитого глюка 256-го рівня в оригінальній грі Pac-Man. Однією з головних переваг "Pac-Man 256" є її інноваційний безкінечний геймплей. Гравець керує Pac-Man'ом, який безперервно рухається по лабіринту, уникаючи привидів і збираючи бонуси. Додатково, гра вводить нові елементи,

такі як павер-апи, які надають Pac-Man'у особливі здібності, що додає глибину і різноманітність ігровому процесу.

Однак, "Pac-Man 256" має й свої недоліки. Одним з них є те, що безкінечний геймплей може стати монотонним для деяких гравців після тривалого часу гри. Також, незважаючи на додаткові функції та павер-апи, гра може здатися менш стратегічно насиченою у порівнянні з іншими версіями Pac-Man. Втім, ці недоліки не заважають "Pac-Man 256" бути інноваційним і захоплюючим досвідом, який поєднує класичні елементи з сучасними геймплейними механіками, що робить її цікавою як для нових гравців, так і для шанувальників серії.

Рисунок 1.5 – Гра «Pac-Man World»

Pac-Man World – це значуща версія гри, яка стала першим значним відходом від класичного 2D-геймплею до тривимірного світу. Видана Namco Bandai у 1999 році для платформ PlayStation і Nintendo 64, ця гра дозволила гравцям досліджувати тривимірні рівні, зберігаючи при цьому основні елементи, що зробили оригінальну гру культовою. Однією з ключових переваг

"Pac-Man World" є її різноманітний геймплей, що включає не тільки збір крапок та уникання привидів, але й вирішення головоломок, битви з босами та платформерні елементи. Це значно розширило ігровий досвід та залучило нових гравців до серії.

Проте, "Pac-Man World" також стикається з деякими недоліками. Перехід до тривимірного геймплею не завжди був гладким, що призводило до проблем з камерою та управлінням, особливо на старих консолях. Деякі гравці також відзначали, що новий формат дещо віддалився від оригінального духу Pac-Man, надаючи перевагу більш аркадному і менш складному стилю гри. Незважаючи на ці недоліки, "Pac-Man World" залишається важливою частиною спадщини Pac-Man, демонструючи здатність серії адаптуватися та еволюціонувати з часом, зберігаючи при цьому свою унікальну ідентичність.

Також, аналізуючи існуючі реалізації гри Pac-Man та інших ігор з елементами алгоритмів інтелекту NPC, можна виявити такі основні недоліки:

- Обмежена адаптивність NPC, тобто багато класичних ігор використовують передбачувані патерни поведінки NPC, що знижує рівень виклику для гравців з часом, а також NPC часто діють за заздалегідь визначеними маршрутами, що робить їх поведінку легко прогнозованою.
- Неефективність алгоритмів великого масштабу, а саме - алгоритми, такі як BFS та DFS, може бути неефективною для великих ігрових карт або складних сценаріїв, що може призвести до затримок і підвисань гри.
- Немає кооперативної поведінки між NPC, а точніше, NPC діють незалежно один від одного у багатьох іграх, що перешкоджає реалізації складних стратегій і тактик..
- Недостатня реалістичність: NPC часто не враховують те, що відбувається навколо гравця та що він робить у реальному часі, що робить їхню поведінку менш реалістичною та нецікавою.

Отже, проведений аналіз аналогів не лише дозволив визначити поточний стан ігрової індустрії, але й вказав на конкретні напрями для покращення гри "Pac-Man". Виявлені недоліки та можливості вдосконалення стали основою для подальшої розробки та впровадження нових механік у гру, що сприятиме створенню більш захоплюючого та інтерактивного ігрового досвіду для гравців.

1.3 Постановка задачі на розробку програмного забезпечення

Зміст гри "Pac-Man" полягає у тому, що гравець, керуючи персонажем на ім'я Пакмен, повинен зібрати всі білі крапки (пелети) на рівні, уникаючи зіткнень з привидами. Рівень вважається завершеним, коли всі крапки зібрані, після чого гра переходить на наступний рівень, який стає складнішим за попередній. Привиди на початку кожного рівня розташовані в центрі екрану в невеликій кімнаті, звідки вони виходять по одному. Один із привидів починає рівень поза кімнатою. Кожен привид має унікальні характеристики та стратегії поведінки.

При реалізації програми було вирішено використовувати об'єктно-орієнтований підхід, що забезпечує низку переваг у порівнянні з процедурним підходом. Об'єктно-орієнтоване програмування (ООП) дозволяє більш ефективно управляти складністю системи, розбиваючи її на взаємопов'язані об'єкти, кожен з яких представляє конкретну сутність гри. Це сприяє кращій модульності та повторному використанню коду, спрощує розширення та підтримку програмного забезпечення.

Для забезпечення привабливого зовнішнього вигляду ігрового поля було вирішено використовувати яскраву графіку. Графічний інтерфейс має бути привабливим та інтуїтивно зрозумілим для користувача, що сприятиме залученню гравців та покращенню їхнього ігрового досвіду. Яскраві кольори та чіткі зображення дозволять гравцям легко орієнтуватися на полі та реагувати на динамічні події.

Інтерфейс програми, що розробляється, повинен бути зручний та зрозумілий користувачу. Важливо забезпечити інтуїтивно зрозумілу навігацію та управління грою, з мінімальною кількістю кроків для виконання основних дій. Програма має оперативно реагувати на натискання клавіш клавіатури, забезпечуючи плавне та своєчасне відображення змін на екрані. Це включає в себе швидкий відгук на рухи Пакмена, оновлення позицій привидів, а також відображення зібраних крапок і бонусів.

Загалом, розробка гри "Pac-Man" передбачає створення інтерактивного, привабливого та зручного ігрового середовища, яке поєднує в собі захоплюючий геймплей та якісний графічний інтерфейс. Застосування об'єктно-орієнтованого підходу сприятиме досягненню цих цілей, забезпечуючи високу якість та надійність програмного продукту.

Рисунок 1.6 – Зображення прототипу інтерфейсу користувача гри “Pac-Man”

Рисунок демонструє базовий прототип інтерфейсу користувача для гри "Pac-Man". Інтерфейс складається з трьох основних елементів:

1) Кількість очок:
   - Розташований у верхній частині екрану.

- Відображає поточну кількість очок, набраних гравцем під час гри.

2) Лабіринт:
   - Займає центральну частину екрану.
   - Основне ігрове поле, де розташовані Pac-Man, привиди, точки та інші ігрові елементи.
   - Гравець керує Pac-Man, який рухається по лабіринту, збираючи точки та уникаючи привидів.
3) Кількість життів:
   - Розташований у нижній частині екрану.
   - Відображає кількість життів, що залишились у гравця.
   - Зменшується щоразу, коли Pac-Man з'їдають привиди.

Цей інтерфейс забезпечує гравцю основну інформацію для гри: поточний рахунок, стан гри в лабіринті та кількість доступних життів.

1.4 Висновок

У цьому розділі дипломної роботи було детально розглянуто історію та основні характеристики гри «Pac-Man». Аналіз гри почався з дослідження її створення, розвитку та впливу на ігрову індустрію, що дало можливість краще зрозуміти контекст і значущість цього проекту. Вивчення історії гри показало, як «Pac-Man» став культовою іконою поп-культури та революціонізував жанр аркадних ігор. Гра, створена в 1980 році, здобула величезну популярність завдяки своїй простій, але захоплюючій механіці, яка привертала увагу широкого кола гравців.

Також було проведено порівняльний аналіз існуючих версій гри «Pac-Man», таких як «Ms. Pac-Man», «Pac-Man Championship Edition», «Pac-Man Championship Edition DX», «Pac-Man 256» та «Pac-Man World». Аналіз цих версій дозволив виявити унікальні ігрові механіки та інноваційні рішення, які зробили їх популярними серед гравців. Наприклад, «Ms. Pac-Man» ввела випадкові патерни руху привидів, що додало грі непередбачуваності, а «Pac-Man Championship Edition» пропонувала нові рівні зі зростаючою складністю та динамічними змінами в ігровому полі.

Було визначено, що вдосконалення алгоритмів інтелекту NPC (неграв персонажів) у «Pac-Man» може значно підвищити якість гри, зробити її більш складною та цікавою для сучасних гравців. Це не тільки забезпечить кращу взаємодію з гравцем, але й додасть новий рівень реалістичності та занурення у світ гри. Підвищення алгоритму інтелекту NPC дозволить створити більш викликові ситуації для гравців, стимулюючи їх використовувати стратегічне мислення та адаптуватися до нових умов гри. Проведений аналіз аналогів також дозволив визначити основні недоліки класичних ігор з елементами інтелекту NPC, такі як обмежена адаптивність, неефективність алгоритмів великого масштабу, відсутність кооперативної поведінки між NPC та недостатня реалістичність їхньої поведінки.

Таким чином, цей розділ роботи не лише пояснює основні теоретичні та історичні питання, але й закладає основу для подальшого аналізу та вдосконалення гри «Pac-Man». Дослідження, проведене в цьому розділі, дозволить ефективно розробити нові стратегії та механіки, що забезпечать значне підвищення якості ігрового процесу та збагачення досвіду гравців. У наступних розділах буде детально описано методологію розробки нових алгоритмів, реалізацію та тестування змін у грі, що дозволить досягти поставлених цілей. Ці кроки включатимуть розробку моделей для тестування різних сценаріїв поведінки NPC, а також інтеграцію та аналіз отриманих результатів для оптимізації ігрового процес.

Крім того, буде розглянуто можливість використання сучасних технологій машинного навчання та штучного інтелекту для покращення поведінки NPC. Це включатиме створення адаптивних алгоритмів, які можуть навчатися на основі дій гравця та автоматично налаштовувати складність гри відповідно до його навичок. Вивчення цих аспектів та їхнє впровадження в гру «Pac-Man» може не лише відновити інтерес до класичної гри, але й привернути нову аудиторію, зацікавлену в сучасних ігрових рішеннях. Дослідження також включатиме аналіз зворотного зв’язку від гравців та тестерів, що допоможе виявити слабкі місця та додаткові можливості для покращення гри. Це забезпечить всебічний підхід до вдосконалення «Pac-Man» і допоможе створити гру, яка відповідатиме очікуванням сучасних гравців та зберігатиме свою історичну та культурну значущість.

2 ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ СИСТЕМИ ГРИ «PAC MAN»

2.1 Універсальна мова моделювання UML в процесі проектування системи гри «Pac-Man»

UML (англ. Unified Modeling Language) - уніфікована мова об'єктно орієнтованого моделювання, використовується у парадигмі об'єктно орієнтованого програмування для розробки мобільних додатків. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення додатку гри «Pac-Man».[2].

Діаграми дають можливість представити додаток гри «Pac-Man» у такому вигляді, щоб її можна було легко перевести в програмний код, також графічний спосіб представлення інформації є найкращим для сприйняття людиною. Є такі основні uml-діаграми:

- діаграма класів (class diagram);
- діаграма стану (statechart diagram);
- діаграма взаємодії (interaction diagrams);
- діаграма послідовності (sequence diagram);
- діаграма взаємодії (collaboration diagram);
- діаграма компонентів (component diagram);
- діаграма активності (activity diagram);[2].

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об'єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки програмного забезпечення для гри «Pac-Man». Під час розробки гри «Pac-Man» щоб промоделювати роботу гри та краще зрозуміти ті зв'язки, що виникнуть під час розробки, різних модулів було створено такі діаграми: діаграма класів, діаграма станів, та діаграма активностей системи «Pac-Man»

2.2 Розробка загальної структурної схеми функціонування системи гри «Pac-Man»

Аналізуючи потоки вхідних і вихідних даних, розробляємо загальний інтерфейс програми, уточнюємо вимоги до функцій, що виконуються програмою визначаємо їх структуру. У даному продукті потрібно реалізувати основні модулі, які б забезпечували основні вимоги до системи.

На етапі ескізного проєктування були виділені такі об’єкти, що входять до складу програми: власне гра «Pac-Man»; ігрове вікно, яке включає ігрове поле з різними ігровими предметами і персонажами і яка також включає поле індикації на якому виведена вся потрібна користувачу інформація. Графічно це відображається в ієрархії об’єктів на рисунку 2.1

Рисунок 2.1 – Основна структурна схема програми

На наведеній схемі представлена основна структурна схема функціонування системи гри "Pac-Man". Схема включає кілька

ключових компонентів, які взаємодіють між собою для забезпечення повного ігрового процесу.

1) Гра "Pac-Man" - уся гра починається з верхнього рівня, який представлений блоком "Гра 'Pac-Man'". Цей блок є основним контейнером для всіх елементів гри
2) Ігрове вікно - наступним рівнем є "Ігрове вікно", яке відповідає за відображення всіх елементів гри на екрані користувача. Це включає як статичні, так і динамічні об'єкти.
3) Ігрове поле - центральним елементом схеми є "Ігрове поле". Саме на цьому полі розгортаються всі події гри. Ігрове поле містить кілька важливих елементів:

   - Гравець: керований гравцем персонаж, який повинен збирати точки та уникати привидів.

   - Привиди: ворожі NPC, які переслідують гравця. Привиди мають різні алгоритми поведінки, які роблять гру цікавою та викликаючою.

   - Фрукти: спеціальні об'єкти, які з'являються на полі та дають додаткові очки, коли гравець їх збирає.

   - Енерджайзер: спеціальний об'єкт, який тимчасово надає гравцю можливість з'їдати привидів. Після з'їдання енерджайзера привиди стають вразливими.
4) Поле індикації - нижче ігрового поля розташоване "Поле індикації", яке показує важливу інформацію для гравця:

   - Кількість життів: індикатор, що показує, скільки життів залишилося у гравця.

   - Поточні очки: відображає кількість очок, набраних гравцем під час поточної гри.

   - Найкращий результат: показує найвищий набраний результат серед всіх ігор, що дозволяє гравцям прагнути досягти нових висот.

Ця блок-схема показує, як різні компоненти гри Pac-Man взаємодіють, створюючи інтерактивну та захоплюючу гру. Кожен елемент має свою роль і разом вони утворюють цілісну систему, яка забезпечує користувачам цікаву гру.

2.3 Моделювання системи гри «Pac-Man» з використанням мови UML

Діаграма активності - по суті, є блок-схемою, яка наочно показує, як потік управління переходить від однієї діяльності до іншої. Активності на діаграмі "розкидані" по бігових доріжках, кожна з яких відповідає поведінці одного з об'єктів (наприклад, клієнта, менеджера, веб-сервера, сервера бази даних тощо). Завдяки цьому легко визначити, яким із об'єктів виконується кожна активність. Доріжка - частина області діаграми діяльності, де відображаються ті активності, які відповідає конкретний об'єкт. Призначені доріжки для розбиття діаграми відповідно до розподілу відповідальності за дії. Ім'я доріжки може означати роль або об'єкт, якому вона відповідає. В нашому випадку є дві доріжки користувач та саме програмне забезпечення.

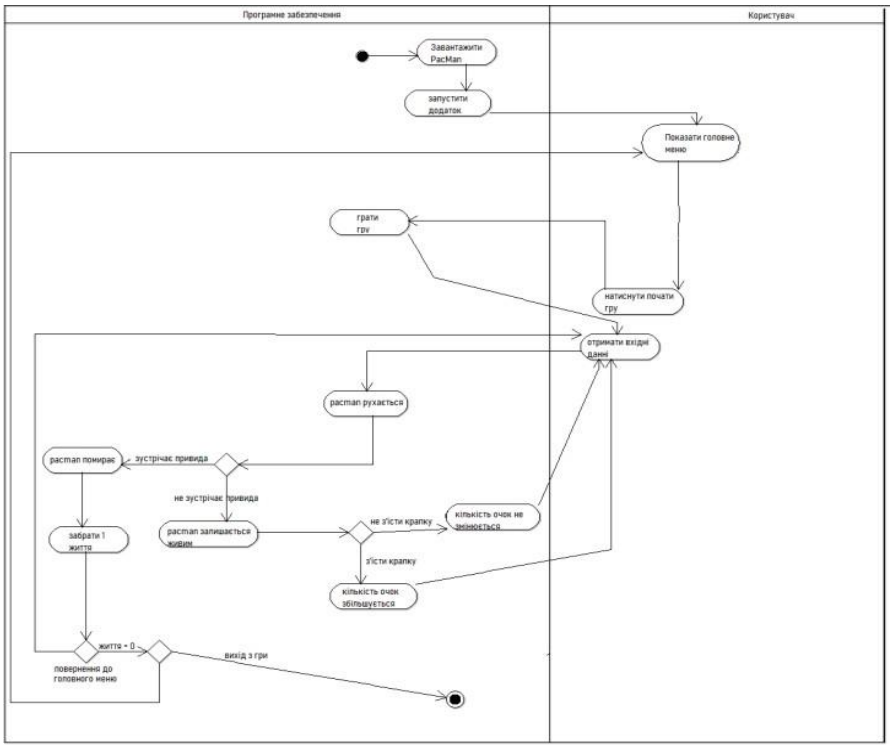

Рисунок 2.2 – Діаграма активності

Діаграма показує послідовність дій у грі "Pac-Man" від запуску програми до завершення гри, з урахуванням дій користувача. Вона поділена на дві частини: "Програмне забезпечення" і "Користувач".

Таблиця 2.1 – Табличне зображення діаграми активностей

Програмне забезпечення	Користувач
1	2
Завантажити Pac-Man: - Початок дій - Завантаження програми.	Показати головне меню: - Користувач бачить головне меню після запуску програми.
Запустити додаток: - Після завантаження програма запускається.	Запустити гру: - Користувач натискає кнопку для початку гри.
Показати головне меню: - Відображається головне меню, де користувач може вибрати початок гри.	Керувати Pac-Man: - Користувач керує Pac-Man за допомогою клавіш WASD або стрілок.
Грати гру: - Користувач запускає гру з головного меню. - Ініціалізація гри.	
Запустити рівень: - Гра запускає новий рівень.	
Створити гру: - Відображення гри на екрані.	
Рендерити рухи: - Програма рендерить рухи гравця та привидів.	

Продовження таблиці 2.1 – Табличне значення діаграми активностей

1	2
Розіслати розташування привидів: - Програма оновлює розташування привидів.	
Перевірка зіткнень: - Програма перевіряє, чи відбулося зіткнення між гравцем і привидами. - Якщо гравець зіткнувся з привидом, відбувається наступна дія.	
Забрати 1 життя: - Якщо відбулося зіткнення, у гравця забирається одне життя.	
Розіслати змінене місце: - Оновлюється розташування гравця після втрати життя. - Якщо у гравця залишилися життя, гра продовжується.	
Життя = 0: - Якщо у гравця більше немає життів, гра завершується.	
Кількість очок = мета: - Програма перевіряє, чи досягнуто мети за кількістю очок. - Якщо мета досягнута, гра завершується перемогою.	
Вихід з гри: - Якщо гравець втратив всі життя або досяг мети, гра завершується.	

Діаграма активностей відображає весь процес гри "Pac-Man", починаючи від завантаження та запуску програми до гри та її завершення. Вона ілюструє взаємодію між програмним забезпеченням та користувачем, а також основні умови, що впливають на перебіг гри.

Діаграма класів описує структуру об’єктів системи і їх індивідуальність по відношенню до інших класів. Вона охоплює важливі концепції для системи. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм. Відповідно до поставленого завдання система міститиме основні класи, які будуть наведені у таблиці 2.2.

Таблиця 2.2 – Основні класи в системі

Назва класу	Перший метод	Другий метод	Третій метод
1	2	3	4
Game	− render() – рендерить поле і об’єкти на ньому	− checkSurroundings() – перевіряє чи було убито пакмана	− gameOver() – завершення гри
Pacman	− update() – перевіряє куди рухається пакман	− draw() – змінює його вигляд в залежності від сторони руху	
GetCount	− drawPoints() – малює самі точки	− flipColor() – відповідає за мигання особливої точки	– getCount() – рахує кількість очок
Ghostt	− isAttacked() – привид атакований	− isDead() – привид мертвий	– move() – привид рухається

Продовження таблиці 2.2 – Основні класи в системі

1	2	3	4
HighScore	– drawReady() – на екрані вивід чи готовий гравець	– drawTilesAround() – відображати точки навкруги	– getHighScore()– рахує кількість очок
Music	– forcePlayMusic – визов певної мелодії в потрібний час	– playMusic – гра музики під час гри	

Отже, у даній системі використано 6 основних класів, кожен з яких виконує певну роль для роботи користувача з програмним додатком

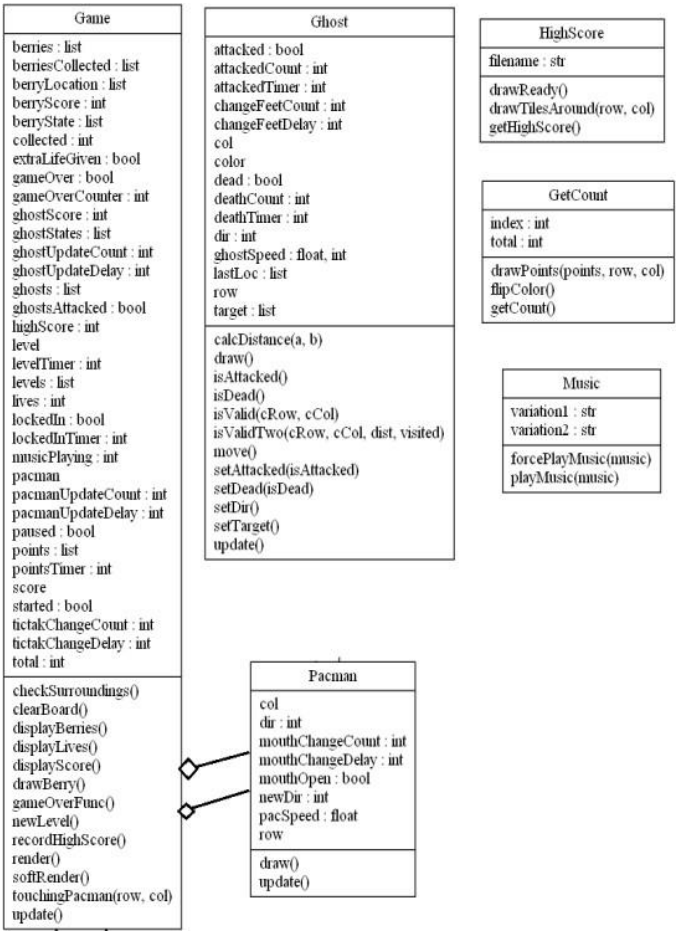

Рисунок 2.3 – Діаграма класів

Клас Game - є основним класом, який керує всією логікою гри. Він відповідає за рендеринг ігрового поля, управління ігровими об'єктами, обробку подій користувача та підтримку стану гри.

Методи класу Game:

- render(): Метод відповідає за оновлення графічного інтерфейсу гри. Він рендерить ігрове поле, Пакмана, привидів, точки та інші елементи на екрані. Також він відображає залишкові життя та зібрані фрукти.

- checkSurroundings(): Метод перевіряє, чи був Пакман убитий, порівнюючи його координати з координатами привидів. Якщо Пакман стикається з привидом, він втрачає життя.

- gameOver(): Цей метод викликається, коли Пакман втрачає всі свої життя. Він показує екран завершення гри, зупиняє всі дії гри та записує найвищий рахунок.

- displayLives(): Метод відображає кількість залишкових життів Пакмана на екрані у вигляді іконок.

- displayBerries(): Метод відображає зібрані фрукти на екрані, що Пакман зібрав під час гри. Кожен зібраний фрукт має своє зображення.

- touchingPacman(row, col): Метод перевіряє, чи привид торкається Пакмана, порівнюючи їх координати. Використовується для визначення зіткнень.

- newLevel(): Метод ініціалізує новий рівень гри, збільшуючи складність гри, перезапускаючи всі елементи гри та налаштовуючи нові цілі для привидів.

- recordHighScore(): Метод записує найвищий рахунок у файл для збереження результатів. Використовується для збереження досягнень гравця.

Клас Pacman - клас Pacman представляє головного героя гри. Він відповідає за управління Пакманом, його рух, анімацію та реакцію на зіткнення з іншими об'єктами.

Методи класу Pacman:

- update(): Метод відповідає за оновлення координат Пакмана відповідно до натиснутих користувачем клавіш. Він перевіряє можливість руху в заданому напрямку та змінює координати Пакмана.
- draw(): Метод відповідає за відображення Пакмана на екрані. Він змінює вигляд Пакмана в залежності від напрямку його руху та стану рота (відкритий чи закритий).

Клас GetCount - відповідає за підрахунок очок і управління точками на ігровому полі.

Методи класу GetCount:

- drawPoints(points, row, col): Метод малює точки на ігровому полі. Він розміщує точки у відповідних місцях лабіринту та оновлює їх відображення.
- flipColor(): Метод змінює колір енерджайзерів, змінюючи їх стан між активним і неактивним. Це додає візуальне різноманіття та сигналізує про зміни у грі.
- getCount(): Метод підраховує загальну кількість зібраних точок на полі. Використовується для визначення прогресу гравця та переходу на новий рівень.

Клас Ghost - представляє привидів у грі. Він відповідає за їх поведінку, рух, взаємодію з Пакманом та реакцію на зміни в грі.

Методи класу Ghost:

- isAttacked(): Метод перевіряє, чи привид був атакований, тобто чи Пакман з'їв енерджайзер, що робить привидів вразливими.
- isDead(): Метод перевіряє, чи привид мертвий, тобто чи був з'їдений Пакманом під час атаки.
- move(): Метод відповідає за рух привида відповідно до поточного режиму (переслідування, блукання, переляк). Він керує напрямком і швидкістю руху привида.

- setDir(): Метод визначає напрямок руху привида, базуючись на його цілі та положенні Пакмана. Він враховує можливість руху в заданому напрямку та перешкоди.
- calcDistance(a, b): Метод обчислює відстань між двома точками на полі, використовуючи евклідову метрику. Використовується для вибору оптимального маршруту.
- setTarget(): Метод встановлює ціль для привида, базуючись на його поточному стані та положенні Пакмана. Він визначає стратегію переслідування або втечі.
- setAttacked(isAttacked): Метод встановлює стан привида як атакований, що змінює його поведінку та швидкість руху.
- setDead(isDead): Метод встановлює стан привида як мертвий, що змінює його поведінку та цілі.

Клас HighScore - клас HighScore відповідає за зберігання та відображення найвищого рахунку гри.

Методи класу HighScore:

- drawReady(): Метод відображає повідомлення про готовність гравця на екрані перед початком гри. Це сигналізує про готовність до початку гри.
- drawTilesAround(row, col): Метод відображає точки навколо Пакмана на полі, щоб показати його оточення та можливі шляхи руху.
- getHighScore(): Метод повертає найвищий набраний рахунок, зчитуючи його з файлу збереження. Це дозволяє гравцеві бачити свої досягнення.

Клас Music – цей клас відповідає за управління музикою та звуковими ефектами в грі. Він забезпечує атмосферу гри та сигналізує про важливі події.

Методи класу Music:

- forcePlayMusic(music): Метод відтворює конкретну мелодію у визначений момент гри, незалежно від поточної музики.

Використовується для важливих подій, як початок або завершення рівня.

- playMusic(music): Метод відповідає за безперервне відтворення музики під час гри, змінюючи музику залежно від ситуації в грі (наприклад, під час атаки або переслідування).

Система гри «Pac-Man» матиме 2 стани: стан діяльності (активний) та стан паузи (пасивний). Взаємодія даних станів показана на рисунку 2.4. Якщо даний стан активний, то виконуються дії, а якщо стан був змінений на пасивний, то очікується зміна стану. При цьому вихід з програми виконується в стані паузи, але припинення роботи програми можливе і в активному стані.

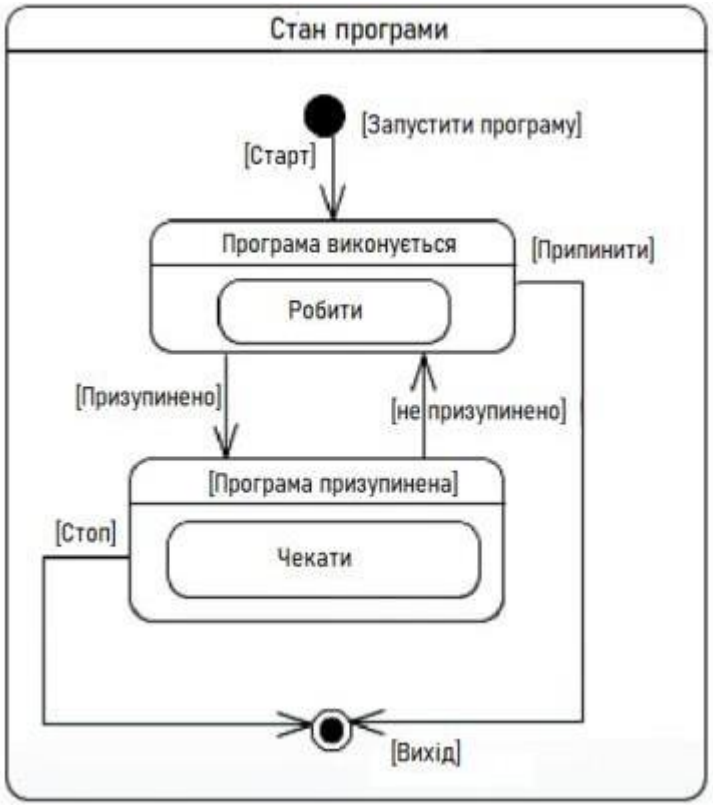

Рисунок – 2.4 – Діаграма станів

Діаграма станів зображує різні стани програми та переходи між ними в процесі виконання. Вона складається з наступних елементів:

1) Початковий стан (чорна точка):

- Стартовий стан, позначений як "Запустити програму".

2) Програма виконується:

- Основний стан роботи програми.

- Містить підстан "Робити", який визначає, що програма виконує свої основні функції.

3) Програма призупинена:

- Стан, коли програма тимчасово не виконує основні функції.

- Містить підстан "Чекати", що означає очікування подальших дій від користувача.

4) Переходи між станами:

- Від початкового стану до стану "Програма виконується" переходить стрілка з міткою [Старт].

- Від стану "Програма виконується" до стану "Програма призупинена" переходить стрілка з міткою [Призупинити].

- Від стану "Програма призупинена" назад до стану "Програма виконується" переходить стрілка з міткою [не призупинено].

- Від стану "Програма призупинена" до кінцевого стану переходить стрілка з міткою [Стоп].

- Від стану "Програма виконується" до кінцевого стану переходить стрілка з міткою [Припинити].

5) Кінцевий стан (біла точка в колі):

- Позначений як [Вихід], цей стан вказує на завершення роботи програми.

Ця діаграма демонструє простий цикл роботи програми з можливістю призупинення і відновлення роботи, а також завершення програми за потреби.

2.4 Розробка алгоритмів функціонування системи гри «Pac-Man»

Для зручного користування програмою необхідно розуміти алгоритм її роботи. Для того щоб розпочати гру необхідно натиснути клавішу “Space”. Розпочавши гру, алгоритм гри працює за наступними кроками:

– заповнення точками ігрового поля, генерування спеціальних точок(енерджайзерів на карті)

– Гравець курує пакманом для того щоб їсти точки, ухиляється від привидів, якщо підібрано енерджайзер можливо сам на них полює, і збирає додатково фрукти для збільшення кількості очок.

Наступний крок. – перевірка чи залишились ще точки, і чи залишилась у пакмана життя, якщо ні - то гра закінчується

Якщо точки і життя ще є – то ігровий персонаж продовжує спроби їх всі зібрати і не спійматися привиду. Схема алгоритму функціонування програмної реалізації гри «PacMan» наведена на рисунку 2.5

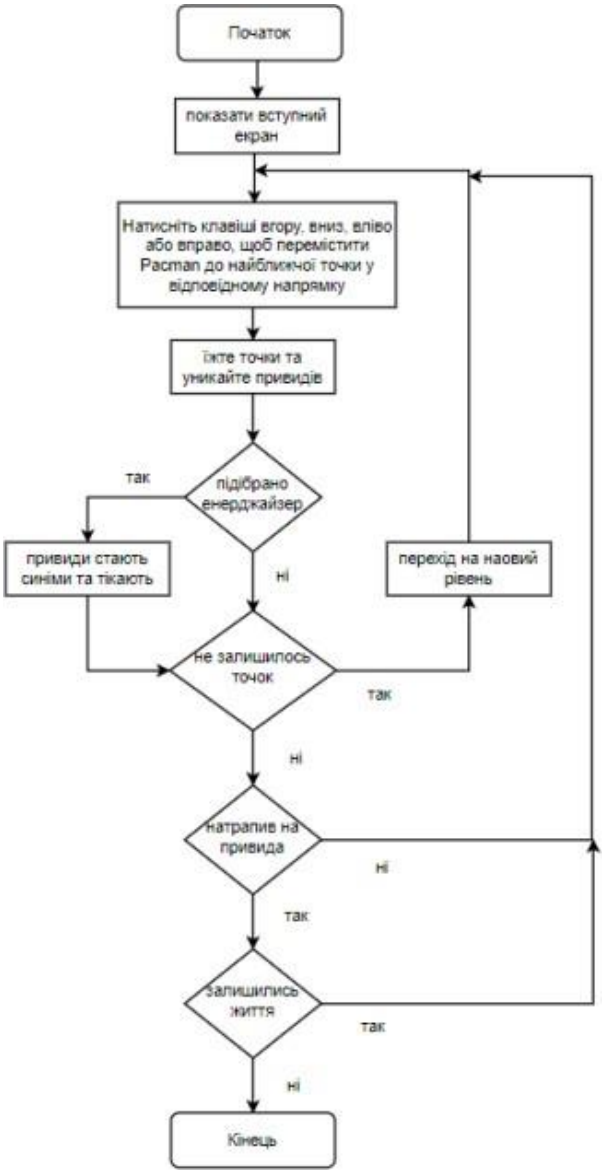

Рисунок 2.5 – Основна схема функціонування гри “Pac-Man”

Схема функціонування гри “Pac-Man” описує икл гри Pac-Man, включаючи початкові налаштування, дії гравця, обробку зіткнень та перевірку умов завершення гри або переходу на новий рівень.

1) Початок:
   - Ініціація гри.
2) Показати вступний екран:
   - Відображення вступного екрану гри.
3) Введення користувача:
   - Натисніть клавіші вгору, вниз, вліво або вправо, або WASD, щоб перемістити Pac-Man до найближчої точки у відповідному напрямку.
4) Ігровий процес:
   - Гравець збирає точки та уникає привидів.
5) Підбір енерджайзера:
   - Якщо енерджайзер піднято, то привиди стають синіми і тікають.
6) Перевірка залишку точок:
   - Якщо точок(бонусів) не залишилось, то перехід на новий рівень.
7) Перевірка на зіткнення з привидом:
   - Якщо Pac-Man натрапляє на привида, то виконується перевірка кількості життів.
8) Перевірка залишку життів:
   - Якщо життів не залишилось, то кінець гри.
   - Якщо життя залишилось, то гравець повертається до ігрового процесу.
9) Кінець:
   - Завершення гри.

Ця схема описує основний алгоритм гри “Pac-Man”.

2.5 Розробка алгоритмів функціонування основних модулів системи гри «Pac-Man»

Індивідуальні характеристики привидів визначають їхню унікальну поведінку та стратегії під час гри. Кожен з чотирьох привидів може перебувати в одному з трьох режимів: переслідування, блукання та переляку. У режимі переслідування привиди діють відповідно до своїх індивідуальних "характерів", кожен з яких має власну стратегію переслідування гравця. Наприклад, Блінкі (червоний привид) агресивно переслідує Пакмена, завжди прямує безпосередньо за ним, тоді як Пінкі (рожевий привид) намагається передбачити рухи гравця і зустріти його на поворотах. Інікі (блакитний привид) та Клайд (помаранчевий привид) діють менш передбачувано, зокрема, Інікі комбінує переслідування з випадковими рухами, а Клайд переміщується між переслідуванням і блуканням.

У режимі блукання привиди прямують до недосяжних точок у кутах екрану, при цьому кожен привид має свій власний кут для переміщення. Це дозволяє гравцеві зібратися з думками і вибудувати стратегію. Режим переляку активується після з'їдання енерджайзера Пакменом, і в цей час привиди змінюють свій колір на синій та рухаються в псевдовипадковому порядку, намагаючись уникнути зіткнення з Пакменом, який тепер може їх з'їсти.

Важливо зазначити, що привиди не можуть змінювати напрямок у середині шляху, за винятком випадків, коли вони знаходяться на розгалуженнях або в кутах, а також у моменті зміни режиму. Це створює додаткові можливості для стратегічного планування гравцем своїх дій, дозволяючи уникати переслідування в складних ситуаціях.

Швидкість привидів нижча за швидкість Пакмена, але Блінкі може його наздоганяти; Пакмен швидший на поворотах, але сповільнюється під час поїдання точок, що ускладнює уникання привидів.

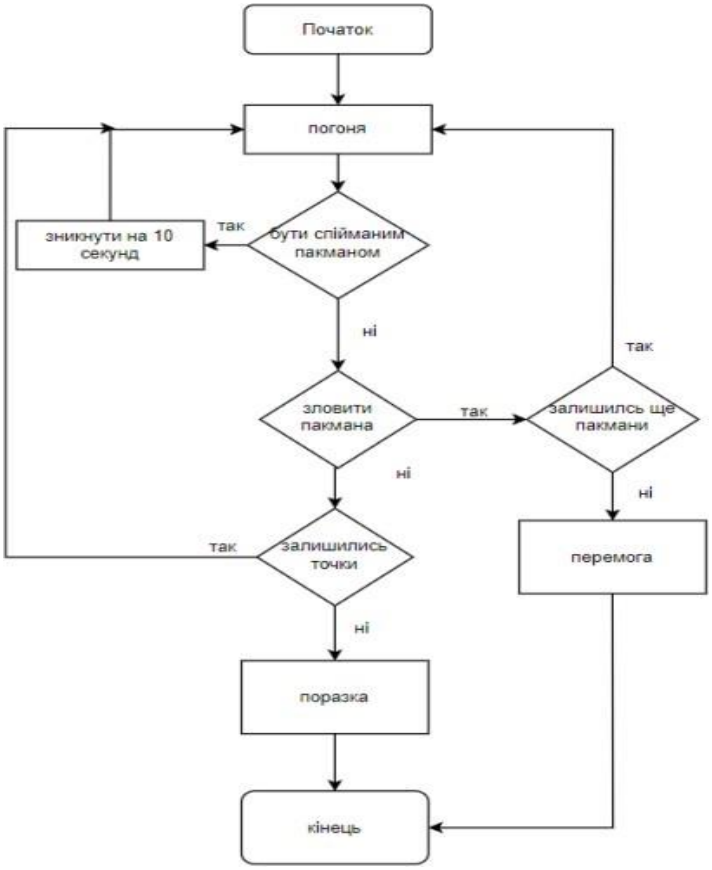

Рисунок 2.6 – Основна схема функціонування привидів у грі “Pac-Man”

Опис системи функціонування привидів у грі Pac-Man:

1) Початок:
   - Початок гри та ігрового процесу.
2) Погоня:
   - Привиди переслідують Пакмена.
3) Бути спійманим Пакменом:
   - Перевірка, чи був привид з'їдений Пакманом.
   - Якщо так, привид стає знищеним на 10 секунд, після чого повертається до погоні.
   - Якщо ні, привид продовжує переслідування Пакмана.
4) Зловити Пакмена:
   - Перевірка, чи вдалося привиду зловити Пакмана.
   - Якщо так, переходить до перевірки залишку Пакменів.
   - Якщо ні, привид продовжує полювання.

5) Залишились ще Пакмени:
   - Перевірка, чи залишилися ще Пакмани у грі.
   - Якщо так, продовжується переслідування.
   - Якщо ні, гра завершується перемогою привидів.
6) Залишились точки(поїнти):
   - Перевірка, чи залишилися ще точки на ігровому полі.
   - Якщо так, продовжується погоня.
   - Якщо ні, гра завершується поразкою Пакмана.
7) Поразка:
   - Гра завершується поразкою Пакмана, якщо всі точки зібрані або всі Пакмани спіймані.
8) Кінець:
   - Завершення гри.

Ця схема описує основний алгоритм поведінки привидів у грі "Pac-Man", включаючи переслідування Пакмана, обробку з'їдання привидів Пакманом, перевірку умов перемоги або поразки.

2.6 Висновок

У цьому розділі дипломної роботи детально розглянуто процес об’єктно-орієнтованого проектування системи гри «Pac-Man» з використанням універсальної мови моделювання UML.

Перш за все, було проведено аналіз основних концепцій та можливостей UML як інструменту для проектування програмних систем. UML надає широкий спектр діаграм, які дозволяють описати як статичні аспекти (структура системи), так і динамічні аспекти (поведінка системи) програмного забезпечення. Було створено діаграми класів, діаграми станів та діаграми активностей, які допомогли візуалізувати різні аспекти роботи системи та взаємодію між її компонентами.

Діаграма класів, яка була розроблена для системи гри «Pac-Man», показала структуру основних об’єктів системи та їх взаємозв'язки. Було виділено шість основних класів: Game, Pacman, GetCount, Ghost, HighScore та Music. Кожен з цих

класів виконує специфічні функції, необхідні для коректної роботи системи. Зокрема, клас Game відповідає за управління основною логікою гри, клас Pacman – за поведінку головного героя, а клас Ghost – за поведінку привидів. Діаграма класів дозволяє чітко визначити відповідальності кожного класу та зрозуміти, як вони взаємодіють між собою.

Діаграма станів була використана для моделювання поведінки системи у різних станах. Було виділено два основних стани: активний (діяльність) та пасивний (пауза). Ця діаграма допомагає зрозуміти, як система переходить між різними станами та які події викликають ці переходи. Наприклад, стан паузи активується під час зупинки гри, а активний стан – під час ігрового процесу.

Діаграма активностей була створена для моделювання основних процесів, які виконуються в системі. Вона наочно показує, як потік управління переходить від однієї діяльності до іншої, які об’єкти відповідають за виконання кожної з цих діяльностей та як вони взаємодіють між собою. Це дозволяє більш детально описати поведінку системи та зрозуміти, як різні компоненти працюють разом для досягнення спільної мети.

Також було розроблено основну структурну схему функціонування системи гри «Pac-Man», яка описує основні модулі програми та їх взаємодію. Ця схема допомагає зрозуміти загальну архітектуру системи та визначити, як різні компоненти взаємодіють між собою для забезпечення коректної роботи програми. Використання сучасних методів проектування та моделювання дозволило забезпечити високу якість програмного продукту та зробити його зручним і зрозумілим для користувачів та розробників.

Таким чином, розділ 2 роботи заклав міцну основу для подальшої реалізації та тестування системи гри «Pac-Man». Використання сучасних методів проектування та моделювання дозволило забезпечити високу якість програмного продукту та зробити його зручним і зрозумілим для користувачів та розробників.

Окрім того, закладена в розділі 2 методологія дозволяє не тільки створити функціональну ігрову систему, але й забезпечити її тестування на всіх етапах розробки. Це гарантує, що всі заплановані функції будуть реалізовані належним чином, а гра буде працювати без помилок і збоїв.

3 ПРОГРАМНА РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ ГРИ “PAC-MAN”

3.1 Обґрунтування вибору мови програмування

Для аналізу і порівняння було обрано 3 мови програмування: Python, Java та C#. Нижче наведено особливості кожної з мов програмування та порівняльну характеристику цих мов у таблиці 3.1.

Python – це мова програмування загального призначення. Вона оптимізована для створення якісного програмного забезпечення, високої продуктивності праці розробників, перенесення програм і інтеграції компонентів. Мова Python використовується сотнями тисяч розробників по всьому світу в таких напрямках як створення веб-сценаріїв, системного адміністрування, створення користувацьких інтерфейсів, налаштування програмних продуктів під користувача, чисельне програмування і тд. Python – одна з самих використовуючих мов програмування в світі. Серед переваг, Python відрізняється простотою, читабельністю і простим синтаксисом; вона легко інтегрується з зовнішніми компонентами, написаними на інших мовах програмування; має мультипарадигменну архітектуру і підтримує об’єктно орієнтоване, функціональне та модульне програмування; має велику колекцію уже запрограмованих інтерфейсів та утиліт.[3].

Синтаксис мови Java багато в чому походить від C та C++. Передусім, Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. На відміну від C++ наявний «Garbage collector», що допомагає запобігти витоку пам'яті шляхом видалення об'єктів, які більше не використовуються додатком.

Java представляє перерахування більш глибоко, ніж C#, тобто розглядає це як іменований екземпляр типу, що спрощує додавання користувацької поведінки до окремих перерахувань. Підтримка узагальнень у даній мові реалізуються з використанням стирань. Параметри загального

типу «стираються», а при компіляції в байт-код додаються приведення. C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML . Однією з переваг мови C# є наявність великої кількості бібліотек та шаблонів, що значно економить час програміста. Як і у Java наявний «Garbage collector» та підтримка узагальнень. В цій мові є делегати, які слугують методами, що можуть бути викликані без повідомлення цільового об'єкта. Для досягнення такої ж функціональності в Java необхідно використовувати інтерфейс з одним методом або іншим способом обходу, який може вимагати великої кількості додаткового коду.

Таблиця 3.1 – Порівняння мов об’єктно-орієнтованого програмування

Характеристика порівняння	Python	Java	C#
Швидкодія	+	-	+
Ручне управління пам’яттю	+	-	-
Перевантаження функцій	+	+	+
ООП	+	+	+
Шаблони	+	+	+

Для виконання бакалаврської дипломної роботи було обрано мову програмування Python, тому що вона є простою у використанні, містить основні підходи об’єктно-орієнтованого програмування. Також їй притаманна динамічна типізація і автоматичне управління пам’яттю, що дуже допомагає у процесі розробки програмного продукту.

3.2 Основні оператори мови програмування Python

Для реалізації системи «гра Pac-Man» було використано інкапсуляцію, наслідування та поліморфізм.

Інкапсуляція дозволяє поєднувати певні елементи в одне ціле з метою утворення нової сутності або абстракції нового рівня. Цей принцип дозволяє зробити деякі з компонентів доступними тільки всередині класу.

Мова програмування Python для реалізації інкапсуляції використовує модифікатори доступу, які дозволяють задати допустиму область видимості для членів класу, тобто контекст, в якому можна використовувати цю змінну або метод. Основі модифікатори доступу – public, private, protected.

– public надає доступ з будь-якої точки поза класом

– private дозволяє доступ тільки всередині класу

– protected реалізує доступ тільки всередині того ж пакета.

В об’єктно-орієнтованому програмуванні наслідування означає відношення «IS – A». Наслідування це одна із самих чудових концепцій об’єктно-орієнтованого програмування, так як воно має на увазі повторне використання.

Основна ідея наслідування в об’єктно-орієнтованому програмування полягає в тому, що клас може унаслідувати характеристики іншого класу. Клас, який наслідує інший клас, називається дочірнім класом або похідним, а клас, з якого унаслідуються характеристики називається батьківським, або основним.

Термін поліморфізм буквально означає наявність декількох форм. В контексті об’єктно-орієнтованого програмування, поліморфізм означає здатність об’єкта вести себе по-різному.

Поліморфізм в програмування реалізується через перезавантаження методу, або через його перевизначення.

Перевизначення методу відноситься до наявності методу з однаковою назвою в дочірньому і батьківських класах. Визначення методу відрізняється в батьківському і дочірньому класах, але назва залишається такою ж.[4].

3.3 Особливості середовища Visual Studio Code

Visual Studio Code – середовище розробки, розповсюджується компанією Microsoft. Поки що це не дуже популярний продукт, але з часом він себе ще проявить. Середовище розробки призначена для створення додатків з використанням різних мов програмування, серед яких Python. Visual Studio Code забезпечує автодоповнення аналізу коду, навігацію по коду, відлагодження, інтеграцію систем управління версіями (зокрема, Git). Перевагою інтегрованою середи розробки Visual Studio Code є робота з проектами (у тому числі повний рефакторінг коду Python). Система має гнучке налаштування від кольорової схеми і шрифту до розміщення панелей. Visual Studio Code підтримує усі існуючі стандарти Python в залежності від встановленої версії, а також такі популярні фреймворки, як Djanco або Flask при умові попереднього встановлення. Має вбудований термінал, який можна налаштувати під будь-яку оболонку. Ще однією великою перевагою Visual Studio Code є те, що він розповсюджується безплатно. Таким чином, Visual Studio Code є однією з найкращих середовищ розробки для програмування на мові Python.[5].

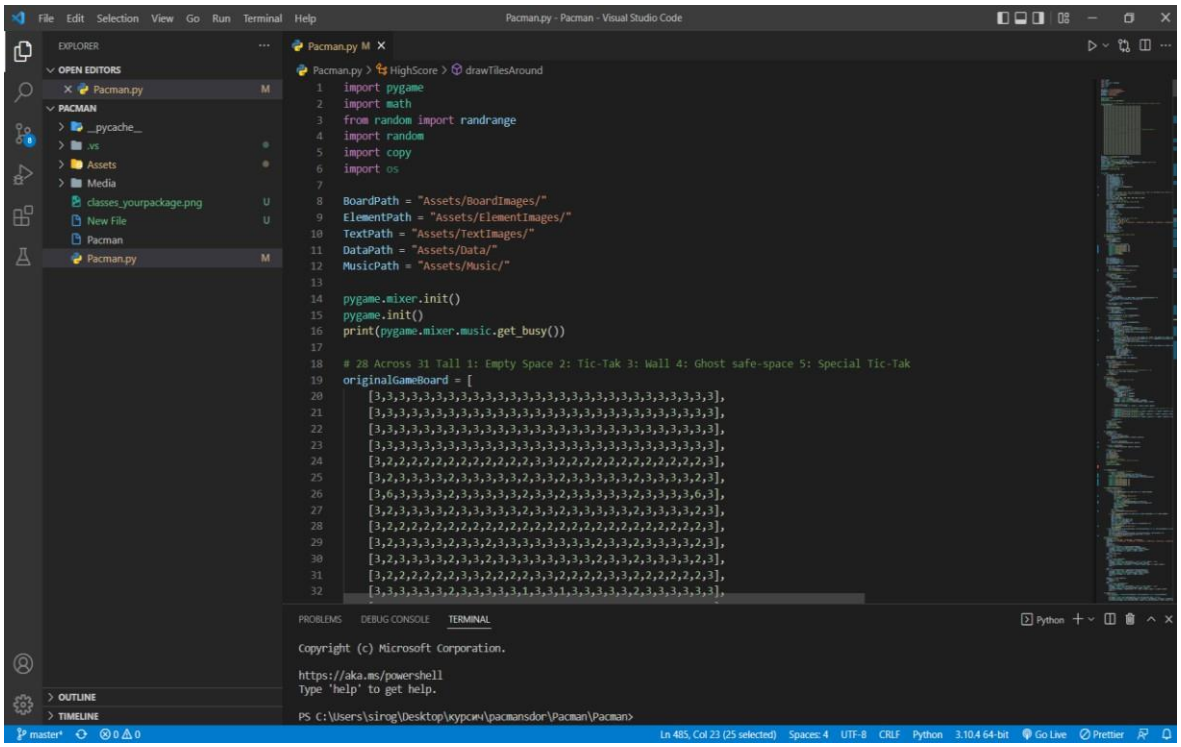

Рисунок 3.1 – Вікно середовища розробки Visual Studio Code

3.4 Програмна реалізація системи гри “Pac-Man”

Програма містить декілька модулів, які забезпечують функціонування системи.

math – цей модуль забезпечує доступ до математичних функцій, визначених стандартом C. Ці функції не можна використовувати з комплексними числами; Різниця між функціями, які підтримують комплексні числа, і функціями, які не підтримують, робиться, оскільки більшість користувачів не хочуть вивчати стільки математики, скільки потрібно для розуміння комплексних чисел. Отримання винятку замість комплексного результату дозволяє раніше виявити несподіване комплексне число, яке використовується як параметр, щоб програміст міг визначити, як і чому воно було згенеровано в першу чергу.[6].

Random - Цей модуль реалізує генератори псевдовипадкових чисел для різних розподілів. Для цілих чисел є рівномірний вибір із діапазону. Для послідовностей існує рівномірний вибір випадкового елемента, функція для створення випадкової перестановки списку на місці та функція

випадкової вибірки без заміни. На реальній лінії є функції для обчислення рівномірного, нормального (гауссового), логнормального, негативного експоненціального, гамма- і бета-розподілів. Для генерування розподілів кутів доступний розподіл фон Мізеса.[8].

Pygame — це набір модулів Python, призначених для написання відеоігор. Pygame додає функціональність на додаток до чудової бібліотеки SDL. Це дозволяє створювати повнофункціональні ігри та мультимедійні програми на мові Python. Pygame дуже портативний і працює майже на кожній платформі та операційній системі. Сам Pygame був завантажений мільйони разів. Використовує оптимізований код C і Assembly для основних функцій. Код C часто в 10-20 разів швидший, ніж код на Python, а код на складання може легко бути в 100 разів або більше, ніж код Python.[10].я

3.5 Тестування розробленої системи гри “Pac-Man”

Тестування є невід'ємною частиною розробки програмного забезпечення, оскільки дозволяє виявити і виправити помилки, підвищити стабільність і продуктивність програми. Під час тестування гри "Pac-Man" було проведено серію тестів для перевірки різних аспектів гри, включаючи функціональність, продуктивність, зручність користування та безпеку.

Тестування почалося з перевірки основної функціональності гри. Було кілька спроб проходження гри, що включали різні розвитки подій, поведінку гравця та взаємодію з NPC. На кожному етапі тестування проводилося ретельне спостереження за поведінкою гри, записувалися всі аномалії та несподівані результати. Особлива увага приділялася перевірці алгоритмів руху NPC, оскільки це було ключовим елементом удосконалення гри.

Запустивши гру для тестування – для гравця стає наявним меню – а отже - все працює коректно.

Рисунок 3.2 – Головне меню

Після натиснення "Space" гра починає працювати, ініціюючи завантаження всіх необхідних ігрових ресурсів і встановлення початкових параметрів гри, таких як позиції персонажів, стан ігрового поля та налаштування музики і звукових ефектів. Після завершення ініціалізації гра очікує від користувача рухів за допомогою стрілочок на клавіатурі, що дозволяють керувати персонажем Пакменом у різних напрямках: вгору, вниз, ліворуч і праворуч. Також гравець може використовувати клавіші WASD для керування, що надає альтернативний спосіб управління і може бути більш зручним для деяких користувачів. Ці дії гравця негайно відображаються на екрані, забезпечуючи миттєвий відгук та інтерактивний ігровий досвід.

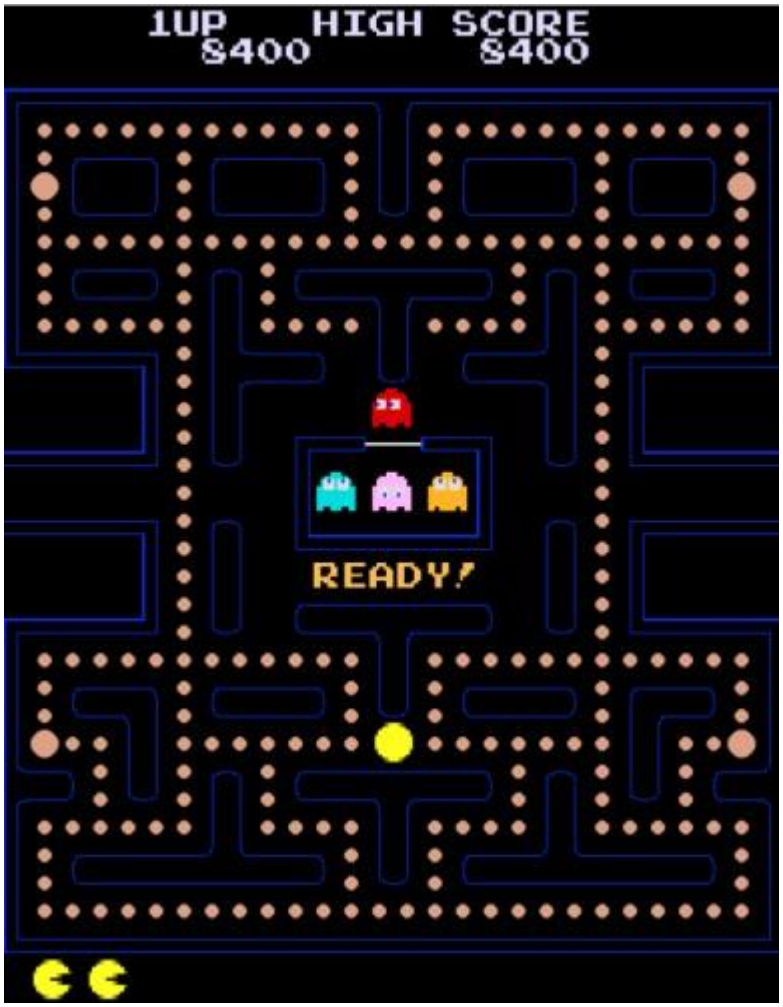

Рисунок 3.3 – Очікування дій

Після натиснення стрілочками в будь-яку сторону починається гра, і на нас починають полювати привиди та з’являються фрукти. Цей момент є стартовою точкою для всієї ігрової механіки, яка включає в себе різноманітні аспекти взаємодії гравця з ігровим світом.

Коли гравець натискає на стрілочку, він починає керувати персонажем, відомим як Пакмен. Основне завдання гравця полягає в тому, щоб зібрати всі точки на ігровому полі, уникаючи при цьому зустрічей із привидами. Привиди, які виступають в ролі антагоністів, мають власні траєкторії руху і намагаються впіймати Пакмена. Якщо привид доторкається до Пакмена, гравець втрачає життя і повинен починати рівень спочатку.

Управління за допомогою стрілочок є інтуїтивно зрозумілим, що робить гру доступною для широкої аудиторії. Ця простота управління, в поєднанні з захоплюючим геймплеєм, є однією з причин великої популярності гри «Pac-Man».

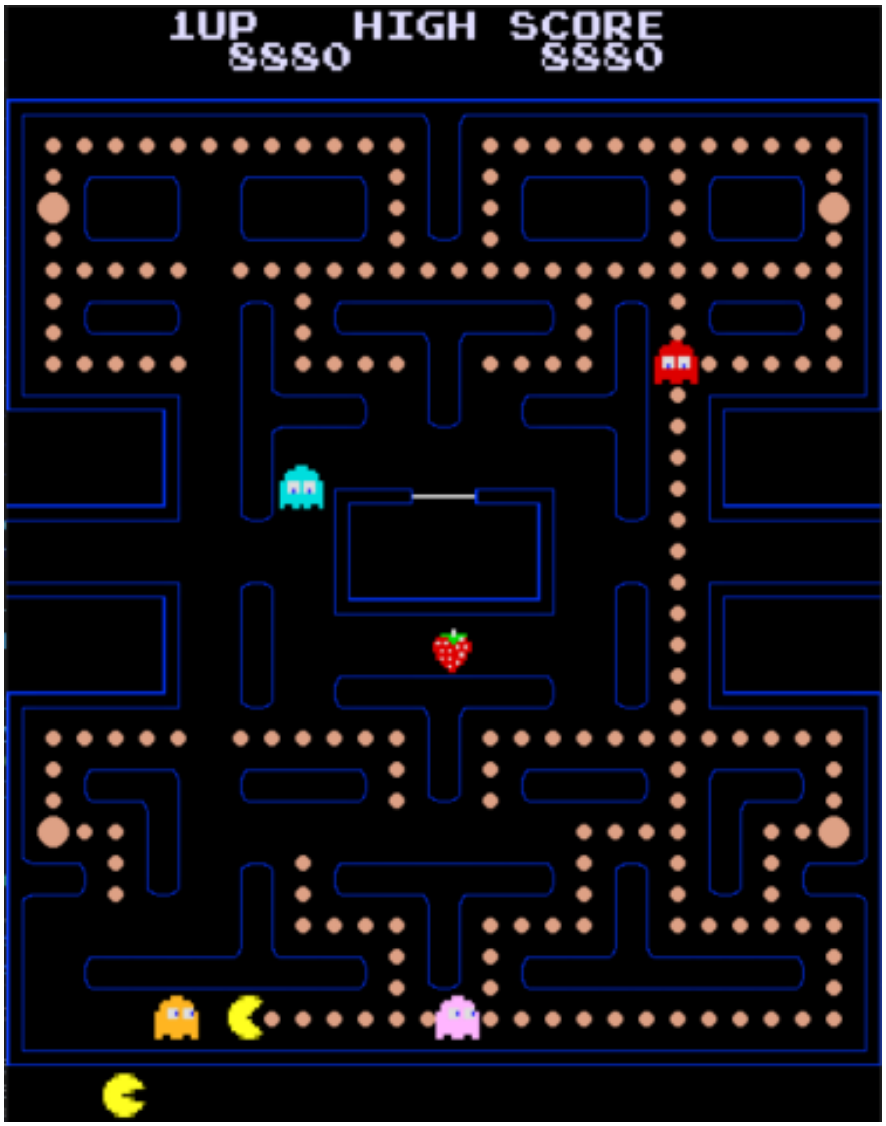

Рисунок 3.4 – Ігровий процес.

Якщо з’їсти енерджайзер, то привиди починають боятися і стають синього кольору. Це змінює динаміку гри, адже тепер Пакман може переслідувати привидів і з’їдати їх. Кожен з’їдений привид приносить гравцеві додаткові бали, які накопичуються на рахунку, що можна побачити на екрані гри. Цей ефект триває певний час, після чого привиди повертаються до свого звичайного стану і знову починають переслідувати Пакмана.

Енерджайзери розташовані в певних точках на ігровому полі, і гравцеві потрібно стратегічно планувати свій рух, щоб досягти їх у потрібний момент. Таким чином, використання енерджайзерів додає ще один рівень складності і глибини в гру, змушуючи гравця продумувати свої дії на кілька кроків вперед.

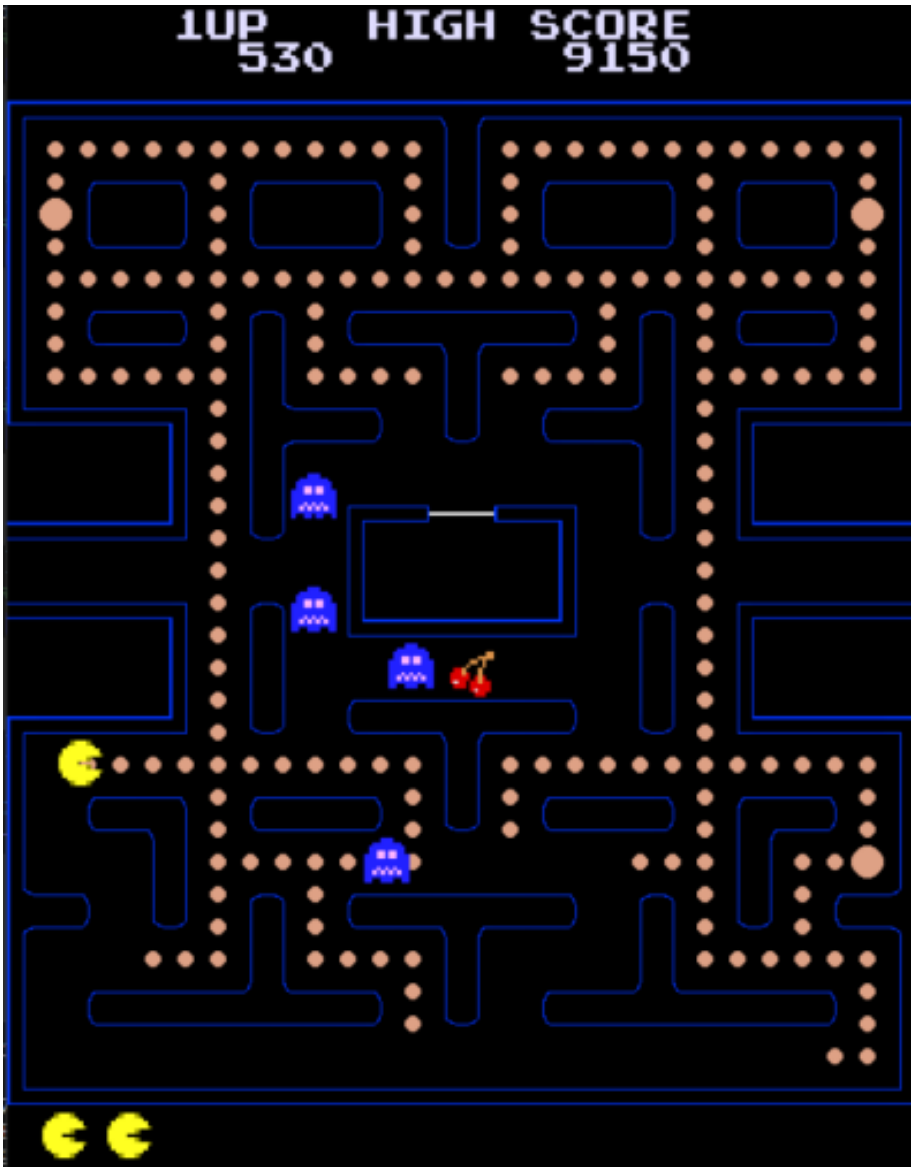

Рисунок 3.5 – Механіка зляканих привидів

При вбивстві одного з привидів під час дії енерджайзера, він повертається в свій "дім" у центрі лабіринту, де відновлює свої сили. Коли завершується ефект страху, привиди стають білими, а потім поступово повертаються до свого нормального кольору, що сигналізує про відновлення їх агресивної поведінки. Цей процес відновлення додає гри додаткової складності, оскільки гравець повинен постійно адаптувати свою стратегію залежно від стану привидів. Таким чином, грати в «Pac-Man» стає ще більш захоплюючим, адже потрібно враховувати як тимчасову перевагу, так і швидке повернення до небезпеки.

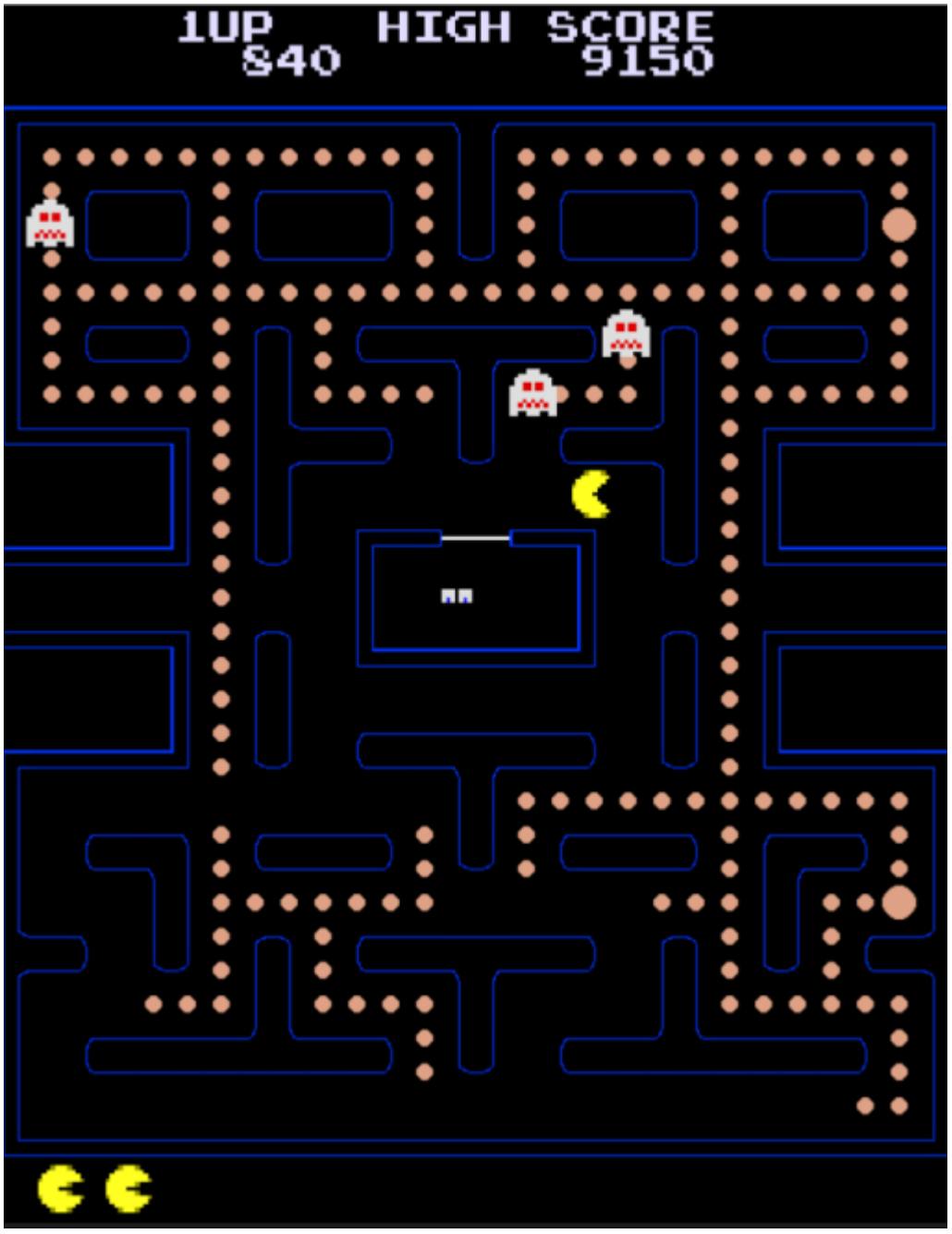

Рисунок 3.6 – завершення ефекту
енерджайзера

3.6 Тестування покращеного інтелекту NPC

При розробці та тестуванні гри Pac-Man з оновленим інтелектом NPC було реалізовано передові методи інтелекту, щоб створювати складніші, але водночас цікавіші ігрові ситуації для гравців.

Таблиця 3.2 – Використані алгоритми для покращення інтелекту NPC

1	2
Використання алгоритму A* для ефективного переслідування	Одним із ключових аспектів удосконалення є використання алгоритму A* для пошуку найкоротшого шляху до Pac-man. A* — найкращий алгоритм для таких завдань, оскільки він ефективно обробляє безліч можливих маршрутів і дозволяє NPC розумно реагувати на рухи гравців. Досягає високої швидкості обчислень. Це важливо для плавного та правильного переміщення NPC на карті гри.
Система станових автоматів для адаптивного поведінкового моделювання	Було впроваджено систему кінцевих автоматів для досягнення реалістичної поведінки NPC у різноманітних ситуаціях. Ця система дозволяє NPC перемикатися між різними станами (полювання, відступ, збір бонусів тощо) залежно від умов, встановлених на основі взаємодії гравців і внутрішніх параметрів NPC. Наприклад, NPC може перейти в режим активного переслідування, якщо неігровий персонаж має достатнє здоров’я та знаходиться поблизу гравця

Продовження таблиці 3.2 – Використані алгоритми для покращення інтелекту NPC

1	2
Оптимізоване керування рухом за допомогою алгоритму прогляду в ширину	Використовує алгоритм ширини перегляду, щоб дозволити NPC ефективно пересуватися по ігровій карті. Цей алгоритм дозволяє NPC оцінити всі доступні маршрути та вибрати найкращий маршрут до місця призначення або гравця. Широке поле зору дозволяє уникнути неефективного переміщення NPC і плавно і логічно пересуватися по карті.

Перш ніж покращувати інтелект NPC у грі Pac-Man, було проведено тестування з використанням секундоміра, щоб оцінити, скільки часу знадобиться NPC, щоб досягти гравця в різних ігрових ситуаціях. Це тестування включало кілька сценаріїв, таких як пряма погоня за гравцем, навігація через складні лабіринти та реакція на зміну напрямку руху гравця. Результати показали, що NPC часто витрачали час на неефективні рухи та невдалі стратегії полювання, що включало повторювані цикли та затримки під час прийняття рішень.

Наприклад, замість того, щоб швидко скорочувати відстань до гравця, привиди могли застрягати в кутах або вибирати шляхи, які були значно довшими. Ці недоліки виявилися особливо помітними в складних лабіринтах, де неправильні рішення призводили до значних затримок.

Аналіз результатів тестування показав необхідність вдосконалення алгоритмів навігації та прийняття рішень, щоб зробити поведінку NPC більш ефективною та непередбачуваною для гравця.

.

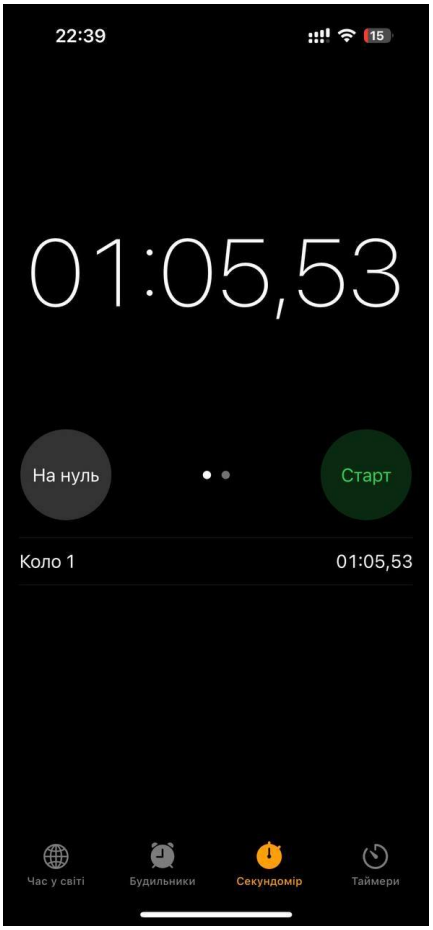

Рисунок 3.7 – Замір часу до вдосконалення алгоритмів інтелекту NPC

Після вдосконалення інтелекту NPC за допомогою алгоритмів A*, станових автоматів і оптимізованого керування рухом було досягнуто значного прогресу в тестуванні за допомогою секундоміра. NPC тепер приймають точніші та швидші рішення для полювання на гравців, використовуючи оптимальні шляхи та адаптивні стратегії. Час, за який NPC досягають гравця, значно скорочено, підвищуючи загальну інтенсивність і емоційність ігрового процесу.

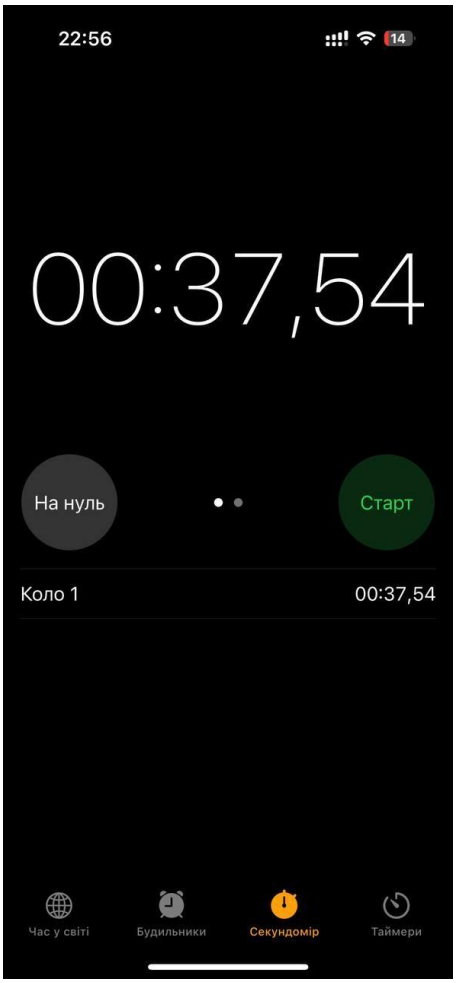

Рисунок 3.8 – Замір часу після вдосконалення алгоритмів інтелекту NPC

Ці технічні рішення та алгоритмічні підходи не тільки зробили гру Pac-Man більш цікавою та складною для гравців, але також вивели складність і стратегічне мислення на новий рівень. Використовуючи передові алгоритми інтелекту, тепер можна створювати NPC, які адаптуються до мінливих ігрових ситуацій і природно реагують на дії гравців з високою точністю. Такий підхід забезпечує гравцям глибший, емоційніший ігровий досвід і відкриває нові можливості для стратегічного планування та взаємодії з ігровим середовищем.

3.7 Висновок

У цьому розділі дипломної роботи було представлено програмну реалізацію системи гри «Pac-Man» за допомогою мови програмування

Python. Проведено всебічний аналіз мов програмування, середовищ розробки та інструментів, що допомогло зробити обґрунтований вибір найбільш придатних для цього проекту технологій. Python було обрано через його простоту, читабельність, багатий набір бібліотек і підтримку об'єктно-орієнтованого програмування.

Тестування системи показало, що всі компоненти працюють коректно. Запуск гри, реакція на натискання клавіш, поведінка персонажів, а також графічний інтерфейс – усе функціонує відповідно до очікувань. Зокрема, результати тестування показали, що вдосконалений інтелект NPC забезпечує значно складніші та цікавіші ігрові ситуації для гравців, що було однією з головних цілей роботи.

Додатково було проведено порівняння продуктивності до та після вдосконалення інтелекту NPC. Результати показали, що час, необхідний привидам для досягнення Pac-Man, значно скоротився, підвищуючи загальну інтенсивність ігрового процесу. Впровадження алгоритму A* та інших удосконалень дозволило створити більш розумних та передбачуваних ворогів, що зробило гру більш захоплюючою.

Таким чином, третій розділ роботи є важливим етапом у розробці гри «Pac-Man». У цьому розділі було застосовано теоретичні знання та сучасні програмні технології для створення ефективної та функціональної системи. Виконання тестування підтвердило коректність роботи програми та дозволило визначити напрямки для можливих майбутніх удосконалень.

Розробка та тестування програмного продукту виявилися успішними, що свідчить про високий рівень реалізації проекту. Отже, всі поставлені завдання було виконано, а основна мета роботи – вдосконалення інтелекту NPC – досягнута. Це забезпечує значне підвищення якості ігрового процесу та збагачення досвіду гравців, що робить гру «Pac-Man» сучасною та цікавою для широкого кола користувачів.

ВИСНОВКИ

В даній бакалаврській дипломній роботі було вдосконалено алгоритми інтелекту NPC. Було досягнуто мети, а саме – вдосконалено алгоритми поведінки привидів, використання алгоритму A*, задля створення більш розумних неігрових персонажів.

Було проведено тестування програмного продукту, з використанням секундоміра – і в ході багатьох спроб проходження рівня – час «життя» гравця був меншим, аніж до вдосконалення алгоритмів інтелекту NPC.

У першому розділі було досліджено основні теоретичні відомості у даній предметній області, були проаналізовано уже існуючі реалізації гри Pac-Man та інших ігор з алгоритмами інтелекту NPC, і було виявлено основні недоліки: Обмежена адаптивність NPC, неефективне використання алгоритмів великого масштабу, відсутність кооперативної поведінки між NPC та не вистачало реалістичної поведінки NPC

У другому розділі було проведено аналіз програмної системи та її проектування. Було наведені основні теоретичні відомості про мову моделювання UML, а також було використано основні її можливості при проектування даної програми. Було розроблено основну структурну схему функціонування програми.

У третьому розділі було проаналізовано мови програмування та середовище розробки. Було виконано безпосереднє програмування за допомогою ООП, описано програмну реалізацію системи. Також було виконано тестування системи на коректність роботи. У результаті тестів виявлено, що програма працює коректно.

Отже, під час виконання роботи було досягнуто поставленої задачі, а саме було вдосконалено алгоритми інтелекту NPC.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Pac-Man [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/abstract/document/8207594
2) Тарасов, О. Ф., and Л. В. Васильєва. "Сучасні методи проєктування програмних систем на основі ООП: навч. посіб. для студентів закладів вищої освіти спеціальності 122 «Комп'ютерні науки»." (2021). [Електронний ресурс]. / Тарасов, О. Ф. і Л. В. Васильєва
3) Подоба Віталій У яких випадках мова програмування Python є правильним вибором? [Електронний ресурс]. – Режим доступу: https://www.vitaliypodoba.com/2014/03/newbie-programmer-why-python-is-good-for-start/
4) Об'єкто-орієнтоване програмування в Python [Електронний ресурс]. – Режим доступу: https://python-scripts.com/object-oriented-programming-in python
5) Visual Studio Code – Code editing. Redefined [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/abstract/document/7476636
6) math module [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/math.html
7) What's the best way to generate a UML diagram from Python source code? [closed] [Електронний ресурс]. – Режим доступу: https://stackoverflow.com/questions/260165/whats-the-best-way-to-generate-a uml-diagram-from-python-source-code
8) random module [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/random.html?highlight=random#module random
9) Pygame [Електронний ресурс]. – Режим доступу: https://www.pygame.org/docs/ref/music.html
10) Максим Маринін [Електронний ресурс]. Режим доступу: https://computerrgameshistory.blogspot.com/2023/04/blog-post.html#more

ДОДАТКИ

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Комп'ютерна гра Pac-Man з удосконаленим інтелектом NPC

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 82,9% Схожість 17,1%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Костюк І.А.

Керівник роботи Малініч І.П.

ДОДАТОК Б

Акт впровадження

УЗГОДЖЕНО
Директор ЦДО ВНТУ
к.т.н., доц. каф. БМІОЕС
Тужанський С. Є.

«____» _____2024 року

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д.т.н., професор
Яровий А.А.

«____» _____2024 року

АКТ ВПРОВАДЖЕННЯ № 3 / 31.05.2024
результатів бакалаврської дипломної роботи

Замовник: Центр діджиталізації освітнього процесу Вінницького національного технічного університету
(найменування організації)

Цим актом підтверджується, що результати роботи – «Комп'ютерна гра Pac-Man з удосконаленим інтелектом NPC»
(найменування теми)

що виконав студент гр. 3КН–20б, Костюк І. А., на громадських засадах
(виконавець)

відповідно до планів робіт Центру діджиталізації освітнього процесу ВНТУ на 2023-2024 р. та впроваджено у Вінницькому національному технічному університеті
(строки виконання) (найменування організації, де здійснювалося впровадження)

1. Вид впроваджених результатів: програмні модулі
(експлуатація виробу, роботи, технології)

2. Характеристика масштабу впровадження: одиничне
(унікальне, одиничне, партія, масове, серійне)

3. Форма впровадження: модулі комп’ютерної гри Pac-Man

4. Новизна результатів роботи: удосконалений інтелект NPC
(піонерські, принципово нові, якісно нові, модифікації, модернізація старих розробок)

5. Впроваджені: у середовищі студентських веб-проєктів в якості шаблонного проєкту

6. Соціальний та науково-технічний ефект: студенти зможуть використати код гри для створення більш продвинутих алгоритмів неігрових персонажів 2D-ігор
(охорона навколишнього середовища, поліпшення й оздоровлення умов праці, удосконалення структури керування, науково-технічних напрямків, спеціальне призначення)

Від виконавця:
студент групи 3КН–20б

_____Костюк І. А.

Керівник: асистент кафедри КН

Від Центру діджиталізації
освітнього процесу ВНТУ, директор,
к.т.н., доц. каф. БМІОЕС:

_____Тужанський С.Є.

_____Малініч І.П.

ДОДАТОК В

Лістинг програми

```
import pygame
import math
from random import randrange
import random
import copy
import os

BoardPath = ".\\Assets\\BoardImages\\"
ElementPath = ".\\Assets\\ElementImages\\"
TextPath = ".\\Assets\\TextImages\\"
DataPath = ".\\Assets\\Data\\"
MusicPath = ".\\Assets\\Music\\"

pygame.mixer.init()
pygame.init()
print(pygame.mixer.music.get_busy())

# 28 Across 31 Tall 1: Empty Space 2: Tic-Tak 3: Wall 4: Ghost safe-space 5: Special Tic-Tak
originalGameBoard = [
    [3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3],
    [3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3],
    [3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3],
    [3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3],
    [3,2,2,2,2,2,2,2,2,2,2,2,2,3,3,2,2,2,2,2,2,2,2,2,2,2,3],
    [3,2,3,3,3,3,3,2,3,3,3,3,3,2,3,3,2,3,3,3,3,3,2,3,3,3,3,2,3],
    [3,6,3,3,3,3,3,2,3,3,3,3,3,2,3,3,2,3,3,3,3,3,2,3,3,3,3,6,3],
    [3,2,3,3,3,3,3,2,3,3,3,3,3,2,3,3,2,3,3,3,3,3,2,3,3,3,3,2,3],
    [3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3],
    [3,2,3,3,3,3,3,2,3,3,3,2,3,3,3,3,3,3,3,2,3,3,2,3,3,3,3,2,3],
    [3,2,3,3,3,3,3,2,3,3,3,2,3,3,3,3,3,3,3,2,3,3,2,3,3,3,3,2,3],
    [3,2,2,2,2,2,2,2,3,3,3,2,2,2,3,3,3,2,2,2,3,3,3,2,2,2,2,2,3],
    [3,3,3,3,3,3,3,2,3,3,3,3,3,1,3,3,1,3,3,3,3,3,3,2,3,3,3,3,3],
    [3,3,3,3,3,3,3,2,3,3,3,3,3,1,3,3,1,3,3,3,3,3,3,2,3,3,3,3,3],
    [3,3,3,3,3,3,3,2,3,3,1,1,1,1,1,1,1,1,1,1,3,3,2,3,3,3,3,3,3],
    [3,3,3,3,3,3,3,2,3,3,1,3,3,3,3,3,3,3,3,3,1,3,3,2,3,3,3,3,3],
    [3,3,3,3,3,3,3,2,3,3,1,3,4,4,4,4,4,4,4,3,1,3,3,2,3,3,3,3,3],
    [1,1,1,1,1,1,2,1,1,1,3,4,4,4,4,4,4,4,3,1,1,1,2,1,1,1,1,1,1], # Middle Lane Row: 14
    [3,3,3,3,3,3,3,2,3,3,1,3,4,4,4,4,4,4,4,3,1,3,3,2,3,3,3,3,3,3],
    [3,3,3,3,3,3,3,2,3,3,1,3,3,3,3,3,3,3,3,3,1,3,3,2,3,3,3,3,3,3],
    [3,3,3,3,3,3,3,2,3,3,1,1,1,1,1,1,1,1,1,1,3,3,2,3,3,3,3,3,3,3],
    [3,3,3,3,3,3,3,2,3,3,1,3,3,3,3,3,3,3,3,3,1,3,3,2,3,3,3,3,3,3],
    [3,3,3,3,3,3,3,2,3,3,1,3,3,3,3,3,3,3,3,3,1,3,3,2,3,3,3,3,3,3],
    [3,2,2,2,2,2,2,2,2,2,2,2,2,3,3,2,2,2,2,2,2,2,2,2,2,2,3],
    [3,2,3,3,3,3,3,2,3,3,3,3,3,2,3,3,2,3,3,3,3,3,2,3,3,3,3,2,3],
    [3,2,3,3,3,3,3,2,3,3,3,3,3,2,3,3,2,3,3,3,3,3,2,3,3,3,3,2,3],
    [3,6,2,2,3,3,2,2,2,2,2,2,2,1,1,2,2,2,2,2,2,2,3,3,2,2,6,3],
    [3,3,3,2,3,3,2,3,3,2,3,3,3,3,3,3,3,3,3,2,3,3,2,3,3,2,3,3,3],
    [3,3,3,2,3,3,2,3,3,2,3,3,3,3,3,3,3,3,3,2,3,3,2,3,3,2,3,3,3],
    [3,2,2,2,2,2,2,2,3,3,2,2,2,2,3,3,2,2,2,2,3,3,2,2,2,2,2,2,3],
```

```
    [3,2,3,3,3,3,3,3,3,3,3,3,2,3,3,2,3,3,3,3,3,3,3,3,3,3,2,3],
    [3,2,3,3,3,3,3,3,3,3,3,3,2,3,3,2,3,3,3,3,3,3,3,3,3,3,2,3],
    [3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3],
    [3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3],
    [3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3],
    [3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3],
]
gameBoard = copy.deepcopy(originalGameBoard)
spriteRatio = 3/2
square = 20 # Size of each unit square
spriteOffset = square * (1 - spriteRatio) * (1/2)
(width, height) = (len(gameBoard[0]) * square, len(gameBoard) * square) # Game screen
screen = pygame.display.set_mode((width, height))
pygame.display.flip()
musicPlaying = 0 # 0: Chomp, 1: Important, 2: Siren
pelletColor = (222, 161, 133)


class Game:
    def __init__(self, level, score):
        self.paused = True
        self.ghostUpdateDelay = 1
        self.ghostUpdateCount = 0
        self.pacmanUpdateDelay = 1
        self.pacmanUpdateCount = 0
        self.tictakChangeDelay = 10
        self.tictakChangeCount = 0
        self.ghostsAttacked = False
        self.highScore = HighScore.getHighScore()
        self.score = score
        self.level = level
        self.lives = 3
        self.pacman = Pacman(26.0, 13.5) # Center of Second Last Row
        self.ghosts = [
            Ghost(14.0, 13.5, "red", 0, self.pacman),
            Ghost(17.0, 11.5, "blue", 1, self.pacman),
            Ghost(17.0, 13.5, "pink", 2, self.pacman),
            Ghost(17.0, 15.5, "orange", 3, self.pacman)
        ]
        self.total = GetCount.getCount()
        self.ghostScore = 200
        self.levels = [[350, 250], [150, 450], [150, 450], [0, 600]]
        random.shuffle(self.levels)
        self.ghostStates = [[1, 0], [0, 0], [1, 0], [0, 0]]
        index = 0
        for state in self.ghostStates:
            state[0] = randrange(2)
            state[1] = randrange(self.levels[index][state[0]] + 1)
            index += 1
        self.collected = 0
        self.started = False
        self.gameOver = False
        self.gameOverCounter = 0
        self.points = []
        self.pointsTimer = 10
        self.berryState = [200, 400, False]
        self.berryLocation = [20.0, 13.5]
```

```
        self.berries = ["tile080.png", "tile081.png", "tile082.png", "tile083.png", "tile084.png",
"tile085.png", "tile086.png", "tile087.png"]
        self.berriesCollected = []
        self.levelTimer = 0
        self.berryScore = 100
        self.lockedInTimer = 100
        self.lockedIn = True
        self.extraLifeGiven = False
        self.musicPlaying = 0

    def update(self):
        print(self.ghostStates)
        if self.gameOver:
            self.gameOverFunc()
            return
        if self.paused or not self.started:
            HighScore.drawTilesAround(21, 10)
            HighScore.drawTilesAround(21, 11)
            HighScore.drawTilesAround(21, 12)
            HighScore.drawTilesAround(21, 13)
            HighScore.drawTilesAround(21, 14)
            HighScore.drawReady()
            pygame.display.update()
            return

        self.levelTimer += 1
        self.ghostUpdateCount += 1
        self.pacmanUpdateCount += 1
        self.tictakChangeCount += 1
        self.ghostsAttacked = False

        if self.score >= 10000 and not self.extraLifeGiven:
            self.lives += 1
            self.extraLifeGiven = True
            Music.forcePlayMusic("pacman_extrapac.wav")

        self.clearBoard()
        for ghost in self.ghosts:
            if ghost.attacked:
                self.ghostsAttacked = True

        index = 0
        for state in self.ghostStates:
            state[1] += 1
            if state[1] >= self.levels[index][state[0]]:
                state[1] = 0
                state[0] += 1
                state[0] %= 2
            index += 1

        index = 0
        for ghost in self.ghosts:
            if not ghost.attacked and not ghost.dead and self.ghostStates[index][0] == 0:
                ghost.target = [self.pacman.row, self.pacman.col]
            index += 1

        if self.levelTimer == self.lockedInTimer:
```

```
            self.lockedIn = False

        if self.ghostUpdateCount == self.ghostUpdateDelay:
            for ghost in self.ghosts:
                ghost.update()
            self.ghostUpdateCount = 0

        if self.tictakChangeCount == self.tictakChangeDelay:
            GetCount.flipColor()
            self.tictakChangeCount = 0

        if self.pacmanUpdateCount == self.pacmanUpdateDelay:
            self.pacmanUpdateCount = 0
            self.pacman.update()
            self.pacman.col %= len(gameBoard[0])
            if self.pacman.row % 1.0 == 0 and self.pacman.col % 1.0 == 0:
                if gameBoard[int(self.pacman.row)][int(self.pacman.col)] == 2:
                    Music.playMusic("munch_1.wav")
                    gameBoard[int(self.pacman.row)][int(self.pacman.col)] = 1
                    self.score += 10
                    self.collected += 1
                    pygame.draw.rect(screen, (0, 0, 0), (self.pacman.col * square, self.pacman.row *
square, square, square))
                elif gameBoard[int(self.pacman.row)][int(self.pacman.col)] == 5 or
gameBoard[int(self.pacman.row)][int(self.pacman.col)] == 6:
                    Music.forcePlayMusic("power_pellet.wav")
                    gameBoard[int(self.pacman.row)][int(self.pacman.col)] = 1
                    self.collected += 1
                    pygame.draw.rect(screen, (0, 0, 0), (self.pacman.col * square, self.pacman.row *
square, square, square))
                    self.score += 50
                    self.ghostScore = 200
                    for ghost in self.ghosts:
                        ghost.attackedCount = 0
                        ghost.setAttacked(True)
                        ghost.setTarget()
                        self.ghostsAttacked = True
        self.checkSurroundings()
        self.highScore = max(self.score, self.highScore)

        global running
        if self.collected == self.total:
            print("New Level")
            Music.forcePlayMusic("intermission.wav")
            self.level += 1
            self.newLevel()

        if self.level - 1 == 8:
            print("You win", self.level, len(self.levels))
            running = False
        self.softRender()

    def render(self):
        screen.fill((0, 0, 0)) # Flushes the screen
        currentTile = 0
        self.displayLives()
        self.displayScore()
```

```
        for i in range(3, len(gameBoard) - 2):
            for j in range(len(gameBoard[0])):
                if gameBoard[i][j] == 3:
                    imageName = str(currentTile)
                    if len(imageName) == 1:
                        imageName = "00" + imageName
                    elif len(imageName) == 2:
                         imageName = "0" + imageName
                    imageName = "tile" + imageName + ".png"
                    tileImage = pygame.image.load(BoardPath + imageName)
                    tileImage = pygame.transform.scale(tileImage, (square, square))
                    screen.blit(tileImage, (j * square, i * square, square, square))

                elif gameBoard[i][j] == 2:
                    pygame.draw.circle(screen, pelletColor,(j * square + square//2, i * square +
square//2), square//4)
                elif gameBoard[i][j] == 5:
                    pygame.draw.circle(screen, (0, 0, 0),(j * square + square//2, i * square + square//2),
square//2)
                elif gameBoard[i][j] == 6:
                    pygame.draw.circle(screen, pelletColor,(j * square + square//2, i * square +
square//2), square//2)

                currentTile += 1
        for ghost in self.ghosts:
            ghost.draw()
        self.pacman.draw()
        pygame.display.update()

    def softRender(self):
        pointsToDraw = []
        for point in self.points:
            if point[3] < self.pointsTimer:
                pointsToDraw.append([point[2], point[0], point[1]])
                point[3] += 1
            else:
                self.points.remove(point)
                HighScore.drawTilesAround(point[0], point[1])

        for point in pointsToDraw:
            GetCount.drawPoints(point[0], point[1], point[2])

        for ghost in self.ghosts:
            ghost.draw()
        self.pacman.draw()
        self.displayScore()
        self.displayBerries()
        self.displayLives()
        self.drawBerry()
        pygame.display.update()

    def clearBoard(self):
            for ghost in self.ghosts:
                HighScore.drawTilesAround(ghost.row, ghost.col)
            HighScore.drawTilesAround(self.pacman.row, self.pacman.col)
            HighScore.drawTilesAround(self.berryLocation[0], self.berryLocation[1])
            HighScore.drawTilesAround(20, 10)
```

```
            HighScore.drawTilesAround(20, 11)
            HighScore.drawTilesAround(20, 12)
            HighScore.drawTilesAround(20, 13)
            HighScore.drawTilesAround(20, 14)

    def checkSurroundings(self):
        for ghost in self.ghosts:
            if self.touchingPacman(ghost.row, ghost.col) and not ghost.attacked:
                if self.lives == 1:
                    print("You lose")
                    Music.forcePlayMusic("death_1.wav")
                    self.gameOver = True
                    for ghost in self.ghosts:
                        HighScore.drawTilesAround(ghost.row, ghost.col)
                    HighScore.drawTilesAround(self.pacman.row, self.pacman.col)
                    self.pacman.draw()
                    pygame.display.update()
                    pause(10000000)
                    return
                self.started = False
                Music.forcePlayMusic("pacman_death.wav")
                reset()
            elif self.touchingPacman(ghost.row, ghost.col) and ghost.isAttacked() and not
ghost.isDead():
                ghost.setDead(True)
                ghost.setTarget()
                ghost.ghostSpeed = 1
                ghost.row = math.floor(ghost.row)
                ghost.col = math.floor(ghost.col)
                self.score += self.ghostScore
                self.points.append([ghost.row, ghost.col, self.ghostScore, 0])
                self.ghostScore *= 2
                Music.forcePlayMusic("eat_ghost.wav")
                pause(10000000)
        if self.touchingPacman(self.berryLocation[0], self.berryLocation[1]) and not self.berryState[2]
and self.levelTimer in range(self.berryState[0], self.berryState[1]):
            self.berryState[2] = True
            self.score += self.berryScore
            self.points.append([self.berryLocation[0], self.berryLocation[1], self.berryScore, 0])
            self.berriesCollected.append(self.berries[(self.level - 1) % 8])
            Music.forcePlayMusic("eat_fruit.wav")

    def displayScore(self):
        textOneUp = ["tile033.png", "tile021.png", "tile016.png"]
        textHighScore = ["tile007.png", "tile008.png", "tile006.png", "tile007.png", "tile015.png",
"tile019.png", "tile002.png", "tile014.png", "tile018.png", "tile004.png"]
        index = 0
        scoreStart = 5
        highScoreStart = 11
        for i in range(scoreStart, scoreStart+len(textOneUp)):
            tileImage = pygame.image.load(TextPath + textOneUp[index])
            tileImage = pygame.transform.scale(tileImage, (square, square))
            screen.blit(tileImage, (i * square, 4, square, square))
            index += 1
        score = str(self.score)
        if score == "0":
            score = "00"
```

```
        index = 0
        for i in range(0, len(score)):
            digit = int(score[i])
            tileImage = pygame.image.load(TextPath + "tile0" + str(32 + digit) + ".png")
            tileImage = pygame.transform.scale(tileImage, (square, square))
            screen.blit(tileImage, ((scoreStart + 2 + index) * square, square + 4, square, square))
            index += 1

        index = 0
        for i in range(highScoreStart, highScoreStart+len(textHighScore)):
            tileImage = pygame.image.load(TextPath + textHighScore[index])
            tileImage = pygame.transform.scale(tileImage, (square, square))
            screen.blit(tileImage, (i * square, 4, square, square))
            index += 1

        highScore = str(self.highScore)
        if highScore == "0":
            highScore = "00"
        index = 0
        for i in range(0, len(highScore)):
            digit = int(highScore[i])
            tileImage = pygame.image.load(TextPath + "tile0" + str(32 + digit) + ".png")
            tileImage = pygame.transform.scale(tileImage, (square, square))
            screen.blit(tileImage, ((highScoreStart + 6 + index) * square, square + 4, square, square))
            index += 1

    def drawBerry(self):
        if self.levelTimer in range(self.berryState[0], self.berryState[1]) and not self.berryState[2]:
            berryImage = pygame.image.load(ElementPath + self.berries[(self.level - 1) % 8])
            berryImage = pygame.transform.scale(berryImage, (int(square * spriteRatio), int(square *
spriteRatio)))
            screen.blit(berryImage, (self.berryLocation[1] * square, self.berryLocation[0] * square,
square, square))

    def gameOverFunc(self):
        global running
        if self.gameOverCounter == 12:
            running = False
            self.recordHighScore()
            return

        HighScore.drawTilesAround(self.pacman.row, self.pacman.col)

        pacmanImage = pygame.image.load(ElementPath + "tile" + str(116 +
self.gameOverCounter) + ".png")
        pacmanImage = pygame.transform.scale(pacmanImage, (int(square * spriteRatio),
int(square * spriteRatio)))
        screen.blit(pacmanImage, (self.pacman.col * square + spriteOffset, self.pacman.row *
square + spriteOffset, square, square))
        pygame.display.update()
        pause(5000000)
        self.gameOverCounter += 1

    def displayLives(self):
        livesLoc = [[34, 3], [34, 1]]
        for i in range(self.lives - 1):
            lifeImage = pygame.image.load(ElementPath + "tile054.png")
```

```
            lifeImage = pygame.transform.scale(lifeImage, (int(square * spriteRatio), int(square *
spriteRatio)))
            screen.blit(lifeImage, (livesLoc[i][1] * square, livesLoc[i][0] * square - spriteOffset, square,
square))

    def displayBerries(self):
        firstBerrie = [34, 26]
        for i in range(len(self.berriesCollected)):
            berrieImage = pygame.image.load(ElementPath + self.berriesCollected[i])
            berrieImage = pygame.transform.scale(berrieImage, (int(square * spriteRatio), int(square *
spriteRatio)))
            screen.blit(berrieImage, ((firstBerrie[1] - (2*i)) * square, firstBerrie[0] * square + 5, square,
square))

    def touchingPacman(self, row, col):
        if row - 0.5 <= self.pacman.row and row >= self.pacman.row and col == self.pacman.col:
            return True
        elif row + 0.5 >= self.pacman.row and row <= self.pacman.row and col == self.pacman.col:
            return True
        elif row == self.pacman.row and col - 0.5 <= self.pacman.col and col >= self.pacman.col:
            return True
        elif row == self.pacman.row and col + 0.5 >= self.pacman.col and col <= self.pacman.col:
            return True
        elif row == self.pacman.row and col == self.pacman.col:
            return True
        return False

    def newLevel(self):
        reset()
        self.lives += 1
        self.collected = 0
        self.started = False
        self.berryState = [200, 400, False]
        self.levelTimer = 0
        self.lockedIn = True
        for level in self.levels:
            level[0] = min((level[0] + level[1]) - 100, level[0] + 50)
            level[1] = max(100, level[1] - 50)
        random.shuffle(self.levels)
        index = 0
        for state in self.ghostStates:
            state[0] = randrange(2)
            state[1] = randrange(self.levels[index][state[0]] + 1)
            index += 1
        global gameBoard
        gameBoard = copy.deepcopy(originalGameBoard)
        self.render()

    def recordHighScore(self):
        file = open(DataPath + "HighScore.txt", "w").close()
        file = open(DataPath + "HighScore.txt", "w+")
        file.write(str(self.highScore))
        file.close()

class HighScore:
    filename = "HighScore.txt"
    @staticmethod
```

```
    def getHighScore():
        file = open(DataPath + HighScore.filename, "r")
        highScore = int(file.read())
        file.close()
        return highScore
    @staticmethod
    def drawTilesAround(row, col):
        row = math.floor(row)
        col = math.floor(col)
        for i in range(row-2, row+3):
            for j in range(col-2, col+3):
                if i >= 3 and i < len(gameBoard) - 2 and j >= 0 and j < len(gameBoard[0]):
                    imageName = str(((i - 3) * len(gameBoard[0])) + j)
                    if len(imageName) == 1:
                        imageName = "00" + imageName
                    elif len(imageName) == 2:
                         imageName = "0" + imageName
                    imageName = "tile" + imageName + ".png"
                    tileImage = pygame.image.load(BoardPath + imageName)
                    tileImage = pygame.transform.scale(tileImage, (square, square))
                    screen.blit(tileImage, (j * square, i * square, square, square))

                    if gameBoard[i][j] == 2:
                        pygame.draw.circle(screen, pelletColor,(j * square + square//2, i * square +
square//2), square//4)
                    elif gameBoard[i][j] == 5:
                        pygame.draw.circle(screen, (0, 0, 0),(j * square + square//2, i * square +
square//2), square//2)
                    elif gameBoard[i][j] == 6:
                        pygame.draw.circle(screen, pelletColor,(j * square + square//2, i * square +
square//2), square//2)
    @staticmethod
    def drawReady():
        ready = ["tile274.png", "tile260.png", "tile256.png", "tile259.png", "tile281.png", "tile283.png"]
        for i in range(len(ready)):
            letter = pygame.image.load(TextPath + ready[i])
            letter = pygame.transform.scale(letter, (int(square), int(square)))
            screen.blit(letter, ((11 + i) * square, 20 * square, square, square))

class GetCount:
    index = 0
    total = 0
    @staticmethod
    def getCount():
        total = 0
        for i in range(3, len(gameBoard) - 2):
            for j in range(len(gameBoard[0])):
                if gameBoard[i][j] == 2 or gameBoard[i][j] == 5 or gameBoard[i][j] == 6:
                    total += 1
        return total
    @staticmethod
    def flipColor():
        global gameBoard
        for i in range(3, len(gameBoard) - 2):
            for j in range(len(gameBoard[0])):
                if gameBoard[i][j] == 5:
                    gameBoard[i][j] = 6
```

```
                pygame.draw.circle(screen, pelletColor,(j * square + square//2, i * square +
square//2), square//2)
            elif gameBoard[i][j] == 6:
                gameBoard[i][j] = 5
                pygame.draw.circle(screen, (0, 0, 0),(j * square + square//2, i * square + square//2),
square//2)
    @staticmethod
    def drawPoints(points, row, col):
        pointStr = str(points)
        index = 0
        for i in range(len(pointStr)):
            digit = int(pointStr[i])
            tileImage = pygame.image.load(TextPath + "tile" + str(224 + digit) + ".png")
            tileImage = pygame.transform.scale(tileImage, (square//2, square//2))
            screen.blit(tileImage, ((col) * square + (square//2 * index), row * square - 20, square//2,
square//2))
            index += 1

class Music:
    variation1 = "munch_1.wav"
    variation2 = "siren_1.wav"
    @staticmethod
    def playMusic(music):
        global musicPlaying
        if not pygame.mixer.music.get_busy():
            pygame.mixer.music.unload()
            pygame.mixer.music.load(MusicPath + music)
            pygame.mixer.music.queue(MusicPath + music)
            pygame.mixer.music.play()
            if music == Music.variation1:
                musicPlaying = 0
            elif music == Music.variation2:
                musicPlaying = 2
            else:
                musicPlaying = 1
    @staticmethod
    def forcePlayMusic(music):
        pygame.mixer.music.unload()
        pygame.mixer.music.load(MusicPath + music)
        pygame.mixer.music.play()
        global musicPlaying
        musicPlaying = 1

class Pacman:
    def __init__(self, row, col):
        self.row = row
        self.col = col
        self.mouthOpen = False
        self.pacSpeed = 1/4
        self.mouthChangeDelay = 5
        self.mouthChangeCount = 0
        self.dir = 0 # 0: North, 1: East, 2: South, 3: West
        self.newDir = 0

    def update(self):
        if self.newDir == 0:
            if canMove(math.floor(self.row - self.pacSpeed), self.col) and self.col % 1.0 == 0:
```

```
                self.row -= self.pacSpeed
                self.dir = self.newDir
                return
        elif self.newDir == 1:
            if canMove(self.row, math.ceil(self.col + self.pacSpeed)) and self.row % 1.0 == 0:
                self.col += self.pacSpeed
                self.dir = self.newDir
                return
        elif self.newDir == 2:
            if canMove(math.ceil(self.row + self.pacSpeed), self.col) and self.col % 1.0 == 0:
                self.row += self.pacSpeed
                self.dir = self.newDir
                return
        elif self.newDir == 3:
            if canMove(self.row, math.floor(self.col - self.pacSpeed)) and self.row % 1.0 == 0:
                self.col -= self.pacSpeed
                self.dir = self.newDir
                return

        if self.dir == 0:
            if canMove(math.floor(self.row - self.pacSpeed), self.col) and self.col % 1.0 == 0:
                self.row -= self.pacSpeed
        elif self.dir == 1:
            if canMove(self.row, math.ceil(self.col + self.pacSpeed)) and self.row % 1.0 == 0:
                self.col += self.pacSpeed
        elif self.dir == 2:
            if canMove(math.ceil(self.row + self.pacSpeed), self.col) and self.col % 1.0 == 0:
                self.row += self.pacSpeed
        elif self.dir == 3:
            if canMove(self.row, math.floor(self.col - self.pacSpeed)) and self.row % 1.0 == 0:
                self.col -= self.pacSpeed

    def draw(self):
        if not game.started:
            pacmanImage = pygame.image.load(ElementPath + "tile112.png")
            pacmanImage = pygame.transform.scale(pacmanImage, (int(square * spriteRatio),
int(square * spriteRatio)))
            screen.blit(pacmanImage, (self.col * square + spriteOffset, self.row * square + spriteOffset,
square, square))
            return

        if self.mouthChangeCount == self.mouthChangeDelay:
            self.mouthChangeCount = 0
            self.mouthOpen = not self.mouthOpen
        self.mouthChangeCount += 1
        if self.dir == 0:
            if self.mouthOpen:
                pacmanImage = pygame.image.load(ElementPath + "tile049.png")
            else:
                pacmanImage = pygame.image.load(ElementPath + "tile051.png")
        elif self.dir == 1:
            if self.mouthOpen:
                pacmanImage = pygame.image.load(ElementPath + "tile052.png")
            else:
                pacmanImage = pygame.image.load(ElementPath + "tile054.png")
        elif self.dir == 2:
            if self.mouthOpen:
```

```
            pacmanImage = pygame.image.load(ElementPath + "tile053.png")
        else:
            pacmanImage = pygame.image.load(ElementPath + "tile055.png")
    elif self.dir == 3:
        if self.mouthOpen:
            pacmanImage = pygame.image.load(ElementPath + "tile048.png")
        else:
            pacmanImage = pygame.image.load(ElementPath + "tile050.png")

    pacmanImage = pygame.transform.scale(pacmanImage, (int(square * spriteRatio),
int(square * spriteRatio)))
    screen.blit(pacmanImage, (self.col * square + spriteOffset, self.row * square + spriteOffset,
square, square))

class Ghost:
    def __init__(self, row, col, color, changeFeetCount, pacman):
        self.row = row
        self.col = col
        self.attacked = False
        self.color = color
        self.dir = randrange(4)
        self.dead = False
        self.changeFeetCount = changeFeetCount
        self.changeFeetDelay = 5
        self.target = [-1, -1]
        self.ghostSpeed = 1/4
        self.lastLoc = [-1, -1]
        self.attackedTimer = 240
        self.attackedCount = 0
        self.deathTimer = 120
        self.deathCount = 0
        self.pacman = pacman

    def update(self):
        if self.target == [-1, -1] or (self.row == self.target[0] and self.col == self.target[1]) or
gameBoard[int(self.row)][int(self.col)] == 4 or self.dead:
            self.setTarget()
        self.setDir()
        self.move()

        if self.attacked:
            self.attackedCount += 1

        if self.attacked and not self.dead:
            self.ghostSpeed = 1/8

        if self.attackedCount == self.attackedTimer and self.attacked:
            if not self.dead:
                self.ghostSpeed = 1/4
                self.row = math.floor(self.row)
                self.col = math.floor(self.col)

            self.attackedCount = 0
            self.attacked = False
            self.setTarget()

        if self.dead and gameBoard[self.row][self.col] == 4:
```

```
            self.deathCount += 1
            self.attacked = False
            if self.deathCount == self.deathTimer:
                self.deathCount = 0
                self.dead = False
                self.ghostSpeed = 1/4

    def draw(self):
        ghostImage = pygame.image.load(ElementPath + "tile152.png")
        currentDir = ((self.dir + 3) % 4) * 2
        if self.changeFeetCount == self.changeFeetDelay:
            self.changeFeetCount = 0
            currentDir += 1
        self.changeFeetCount += 1
        if self.dead:
            tileNum = 152 + currentDir
            ghostImage = pygame.image.load(ElementPath + "tile" + str(tileNum) + ".png")
        elif self.attacked:
            if self.attackedTimer - self.attackedCount < self.attackedTimer//3:
                if (self.attackedTimer - self.attackedCount) % 31 < 26:
                    ghostImage = pygame.image.load(ElementPath + "tile0" + str(70 + (currentDir -
(((self.dir + 3) % 4) * 2))) + ".png")
                else:
                    ghostImage = pygame.image.load(ElementPath + "tile0" + str(72 + (currentDir -
(((self.dir + 3) % 4) * 2))) + ".png")
            else:
                ghostImage = pygame.image.load(ElementPath + "tile0" + str(72 + (currentDir -
(((self.dir + 3) % 4) * 2))) + ".png")
        else:
            if self.color == "blue":
                tileNum = 136 + currentDir
                ghostImage = pygame.image.load(ElementPath + "tile" + str(tileNum) + ".png")
            elif self.color == "pink":
                tileNum = 128 + currentDir
                ghostImage = pygame.image.load(ElementPath + "tile" + str(tileNum) + ".png")
            elif self.color == "orange":
                tileNum = 144 + currentDir
                ghostImage = pygame.image.load(ElementPath + "tile" + str(tileNum) + ".png")
            elif self.color == "red":
                tileNum = 96 + currentDir
                if tileNum < 100:
                    ghostImage = pygame.image.load(ElementPath + "tile0" + str(tileNum) + ".png")
                else:
                    ghostImage = pygame.image.load(ElementPath + "tile" + str(tileNum) + ".png")

        ghostImage = pygame.transform.scale(ghostImage, (int(square * spriteRatio), int(square *
spriteRatio)))
        screen.blit(ghostImage, (self.col * square + spriteOffset, self.row * square + spriteOffset,
square, square))

    def isValidTwo(self, cRow, cCol, dist, visited):
        if cRow < 3 or cRow >= len(gameBoard) - 5 or cCol < 0 or cCol >= len(gameBoard[0]) or
gameBoard[cRow][cCol] == 3:
            return False
        elif visited[cRow][cCol] <= dist:
            return False
        return True
```

```
def isValid(self, cRow, cCol):
    if cCol < 0 or cCol > len(gameBoard[0]) - 1:
        return True
    for ghost in game.ghosts:
        if ghost.color == self.color:
            continue
        if ghost.row == cRow and ghost.col == cCol and not self.dead:
            return False
    if not ghostGate.count([cRow, cCol]) == 0:
        if self.dead and self.row < cRow:
            return True
        elif self.row > cRow and not self.dead and not self.attacked and not game.lockedIn:
            return True
        else:
            return False
    if gameBoard[cRow][cCol] == 3:
        return False
    return True

def setDir(self):
    dirs = [[0, -self.ghostSpeed, 0],
            [1, 0, self.ghostSpeed],
            [2, self.ghostSpeed, 0],
            [3, 0, -self.ghostSpeed]
    ]
    random.shuffle(dirs)
    best = 10000
    bestDir = -1
    for newDir in dirs:
        if self.calcDistance(self.target, [self.row + newDir[1], self.col + newDir[2]]) < best:
            if not (self.lastLoc[0] == self.row + newDir[1] and self.lastLoc[1] == self.col + newDir[2]):
                if newDir[0] == 0 and self.col % 1.0 == 0:
                    if self.isValid(math.floor(self.row + newDir[1]), int(self.col + newDir[2])):
                        bestDir = newDir[0]
                        best = self.calcDistance(self.target, [self.row + newDir[1], self.col + newDir[2]])
                elif newDir[0] == 1 and self.row % 1.0 == 0:
                    if self.isValid(int(self.row + newDir[1]), math.ceil(self.col + newDir[2])):
                        bestDir = newDir[0]
                        best = self.calcDistance(self.target, [self.row + newDir[1], self.col + newDir[2]])
                elif newDir[0] == 2 and self.col % 1.0 == 0:
                    if self.isValid(math.ceil(self.row + newDir[1]), int(self.col + newDir[2])):
                        bestDir = newDir[0]
                        best = self.calcDistance(self.target, [self.row + newDir[1], self.col + newDir[2]])
                elif newDir[0] == 3 and self.row % 1.0 == 0:
                    if self.isValid(int(self.row + newDir[1]), math.floor(self.col + newDir[2])):
                        bestDir = newDir[0]
                        best = self.calcDistance(self.target, [self.row + newDir[1], self.col + newDir[2]])
    self.dir = bestDir

def calcDistance(self, a, b):
    dR = a[0] - b[0]
    dC = a[1] - b[1]
    return math.sqrt((dR * dR) + (dC * dC))

def setTarget(self):
    if gameBoard[int(self.row)][int(self.col)] == 4 and not self.dead:
```

```
            self.target = [ghostGate[0][0] - 1, ghostGate[0][1]+1]
            return
        elif gameBoard[int(self.row)][int(self.col)] == 4 and self.dead:
            self.target = [self.row, self.col]
        elif self.dead:
            self.target = [14, 13]
            return

        quads = [0, 0, 0, 0]
        for ghost in game.ghosts:
            if ghost.target[0] <= 15 and ghost.target[1] >= 13:
                quads[0] += 1
            elif ghost.target[0] <= 15 and ghost.target[1] < 13:
                quads[1] += 1
            elif ghost.target[0] > 15 and ghost.target[1] < 13:
                quads[2] += 1
            elif ghost.target[0]> 15 and ghost.target[1] >= 13:
                quads[3] += 1

        while True:
            self.target = [randrange(31), randrange(28)]
            quad = 0
            if self.target[0] <= 15 and self.target[1] >= 13:
                quad = 0
            elif self.target[0] <= 15 and self.target[1] < 13:
                quad = 1
            elif self.target[0] > 15 and self.target[1] < 13:
                quad = 2
            elif self.target[0] > 15 and self.target[1] >= 13:
                quad = 3
            if not gameBoard[self.target[0]][self.target[1]] == 3 and not
gameBoard[self.target[0]][self.target[1]] == 4:
                break
            elif quads[quad] == 0:
                break

    def move(self):
        self.lastLoc = [self.row, self.col]
        if self.dir == 0:
            self.row -= self.ghostSpeed
        elif self.dir == 1:
            self.col += self.ghostSpeed
        elif self.dir == 2:
            self.row += self.ghostSpeed
        elif self.dir == 3:
            self.col -= self.ghostSpeed

        self.col = self.col % len(gameBoard[0])
        if self.col < 0:
            self.col = len(gameBoard[0]) - 0.5

    def setAttacked(self, isAttacked):
        self.attacked = isAttacked

    def isAttacked(self):
        return self.attacked
```

```
    def setDead(self, isDead):
        self.dead = isDead

    def isDead(self):
        return self.dead

game = Game(1, 0)
ghostsafeArea = [15, 13]
ghostGate = [[15, 13], [15, 14]]

def canMove(row, col):
    if col == -1 or col == len(gameBoard[0]):
        return True
    if gameBoard[int(row)][int(col)] != 3:
        return True
    return False

def reset():
    global game
    game.ghosts = [
        Ghost(14.0, 13.5, "red", 0, game.pacman),
        Ghost(17.0, 11.5, "blue", 1, game.pacman),
        Ghost(17.0, 13.5, "pink", 2, game.pacman),
        Ghost(17.0, 15.5, "orange", 3, game.pacman)
    ]
    for ghost in game.ghosts:
        ghost.setTarget()
    game.pacman = Pacman(26.0, 13.5)
    game.lives -= 1
    game.paused = True
    game.render()

def displayLaunchScreen():
    pacmanTitle = ["tile016.png", "tile000.png", "tile448.png", "tile012.png", "tile000.png",
"tile013.png"]
    for i in range(len(pacmanTitle)):
        letter = pygame.image.load(TextPath + pacmanTitle[i])
        letter = pygame.transform.scale(letter, (int(square * 4), int(square * 4)))
        screen.blit(letter, ((2 + 4 * i) * square, 2 * square, square, square))

    characterTitle = [
        "tile002.png", "tile007.png", "tile000.png", "tile018.png", "tile000.png", "tile002.png",
"tile020.png", "tile004.png", "tile018.png",
        "tile015.png", "tile042.png", "tile015.png",
        "tile013.png", "tile008.png", "tile002.png", "tile010.png", "tile013.png", "tile000.png",
"tile012.png", "tile004.png"
    ]
    for i in range(len(characterTitle)):
        letter = pygame.image.load(TextPath + characterTitle[i])
        letter = pygame.transform.scale(letter, (int(square), int(square)))
        screen.blit(letter, ((4 + i) * square, 10 * square, square, square))

    characters = [
        [
            "tile449.png", "tile015.png", "tile107.png", "tile015.png", "tile083.png", "tile071.png",
"tile064.png", "tile067.png", "tile078.png", "tile087.png",
            "tile015.png", "tile015.png", "tile015.png", "tile015.png",
```

```
            "tile108.png", "tile065.png", "tile075.png", "tile072.png", "tile077.png", "tile074.png",
"tile089.png", "tile108.png"
        ],
        [
            "tile450.png", "tile015.png", "tile363.png", "tile015.png", "tile339.png", "tile336.png",
"tile324.png", "tile324.png", "tile323.png", "tile345.png",
            "tile015.png", "tile015.png", "tile015.png", "tile015.png",
            "tile364.png", "tile336.png", "tile328.png", "tile333.png", "tile330.png", "tile345.png",
"tile364.png"
        ],
        [
            "tile452.png", "tile015.png", "tile363.png", "tile015.png", "tile193.png", "tile192.png",
"tile211.png", "tile199.png", "tile197.png", "tile213.png", "tile203.png",
            "tile015.png", "tile015.png", "tile015.png",
            "tile236.png", "tile200.png", "tile205.png", "tile202.png", "tile217.png", "tile236.png"
        ],
        [
            "tile451.png", "tile015.png", "tile363.png", "tile015.png", "tile272.png", "tile270.png",
"tile266.png", "tile260.png", "tile281.png",
            "tile015.png", "tile015.png", "tile015.png", "tile015.png", "tile015.png",
            "tile300.png", "tile258.png", "tile267.png", "tile281.png", "tile259.png", "tile260.png",
"tile300.png"
        ]
    ]
    for i in range(len(characters)):
        for j in range(len(characters[i])):
            if j == 0:
                letter = pygame.image.load(TextPath + characters[i][j])
                letter = pygame.transform.scale(letter, (int(square * spriteRatio), int(square *
spriteRatio)))
                screen.blit(letter, ((2 + j) * square - square//2, (12 + 2 * i) * square - square//3, square,
square))
            else:
                letter = pygame.image.load(TextPath + characters[i][j])
                letter = pygame.transform.scale(letter, (int(square), int(square)))
                screen.blit(letter, ((2 + j) * square, (12 + 2 * i) * square, square, square))

    event = ["tile449.png", "tile015.png", "tile452.png", "tile015.png",  "tile015.png", "tile448.png",
"tile453.png", "tile015.png", "tile015.png", "tile015.png",  "tile453.png"]
    for i in range(len(event)):
        character = pygame.image.load(TextPath + event[i])
        character = pygame.transform.scale(character, (int(square * 2), int(square * 2)))
        screen.blit(character, ((4 + i * 2) * square, 24 * square, square, square))

    wall = ["tile454.png", "tile454.png", "tile454.png", "tile454.png", "tile454.png", "tile454.png",
"tile454.png", "tile454.png", "tile454.png", "tile454.png", "tile454.png", "tile454.png", "tile454.png",
"tile454.png", "tile454.png"]
    for i in range(len(wall)):
        platform = pygame.image.load(TextPath + wall[i])
        platform = pygame.transform.scale(platform, (int(square * 2), int(square * 2)))
        screen.blit(platform, ((i * 2) * square, 26 * square, square, square))

    instructions = ["tile016.png", "tile018.png", "tile004.png", "tile019.png", "tile019.png",
"tile015.png", "tile019.png", "tile016.png", "tile000.png", "tile002.png", "tile004.png", "tile015.png",
"tile020.png", "tile014.png", "tile015.png", "tile016.png", "tile011.png", "tile000.png", "tile025.png"]
    for i in range(len(instructions)):
        letter = pygame.image.load(TextPath + instructions[i])
```

```
        letter = pygame.transform.scale(letter, (int(square), int(square)))
        screen.blit(letter, ((4.5 + i) * square, 35 * square - 10, square, square))

    pygame.display.update()

running = True
onLaunchScreen = True
displayLaunchScreen()

def pause(time):
    cur = 0
    while not cur == time:
        cur += 1

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            game.recordHighScore()
        elif event.type == pygame.KEYDOWN:
            game.paused = False
            game.started = True
            if event.key == pygame.K_w:
                if not onLaunchScreen:
                    game.pacman.newDir = 0
            elif event.key == pygame.K_d:
                if not onLaunchScreen:
                    game.pacman.newDir = 1
            elif event.key == pygame.K_s:
                if not onLaunchScreen:
                    game.pacman.newDir = 2
            elif event.key == pygame.K_a:
                if not onLaunchScreen:
                    game.pacman.newDir = 3
            elif event.key == pygame.K_SPACE:
                if onLaunchScreen:
                    onLaunchScreen = False
                    game.paused = True
                    game.started = False
                    game.render()
                    pygame.mixer.music.load(MusicPath + "pacman_beginning.wav")
                    pygame.mixer.music.play()
                    musicPlaying = 1
            elif event.key == pygame.K_q:
                running = False
                game.recordHighScore()

    if not onLaunchScreen:
        game.update()
```

ДОДАТОК Г

ГРАФІЧНА ЧАСТИНА

КОМП'ЮТЕРНА ГРА PAC-MAN З УДОСКОНАЛЕНИМ ІНТЕЛЕКТОМ NPC

Виконав студент 4 курсу, групи 3КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напрямку підготовки, спеціальності)
Костюк І. А.
(прізвище та ініціали)
Керівник: асистент кафедри КН
Малініч І.П .
(прізвище та ініціали)
«07» 06 2024 р.

Вінниця ВНТУ – 2024 рік

Рисунок Г.1 – Гра «Ms. Pac-Man»

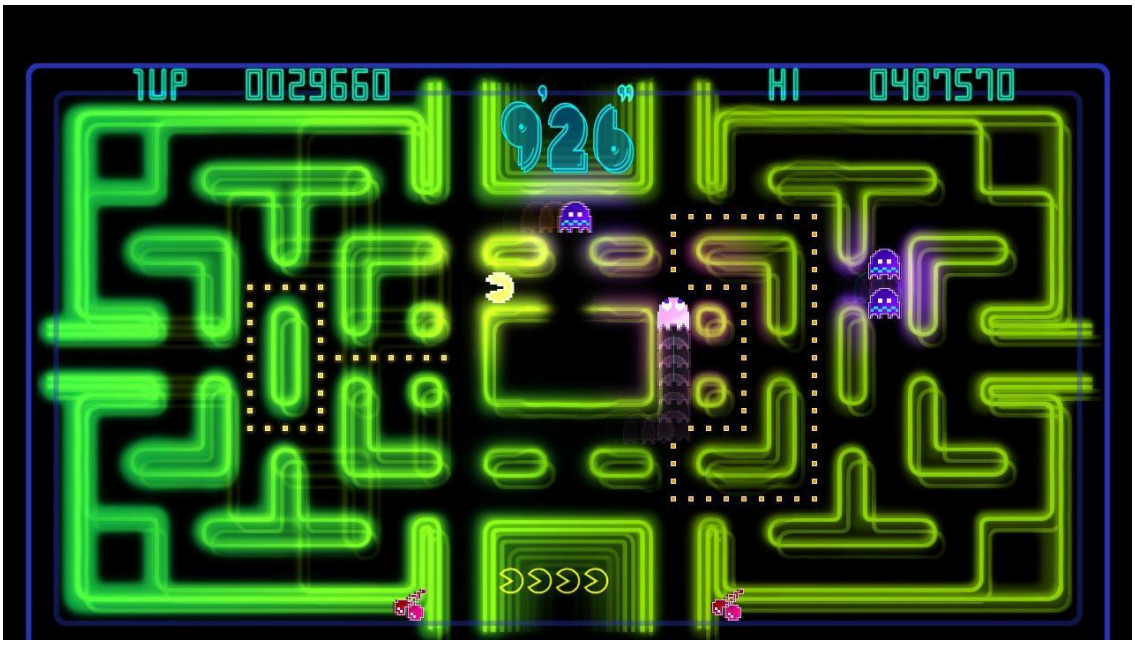

Рисунок Г.2 – Гра «Pac-Man Championship Edition»

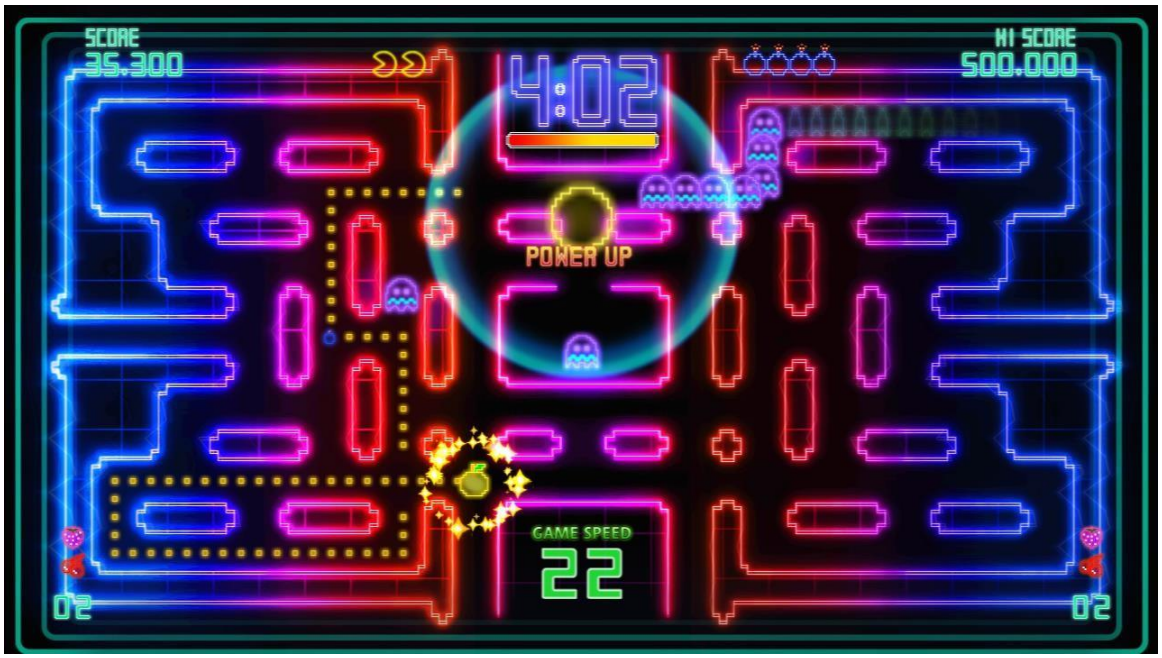

Рисунок Г.3 – Гра «Pac-Man Championship Edition DX»

Рисунок 1.4 – Гра «Pac-256»

Рисунок 1.5 – Гра «Pac-Man Championship World»

Рисунок Г.6 – Зображення прототипу інтерфейсу користувача гри “Pac-Man

Рисунок Г.7 – Основна структурна схема програми

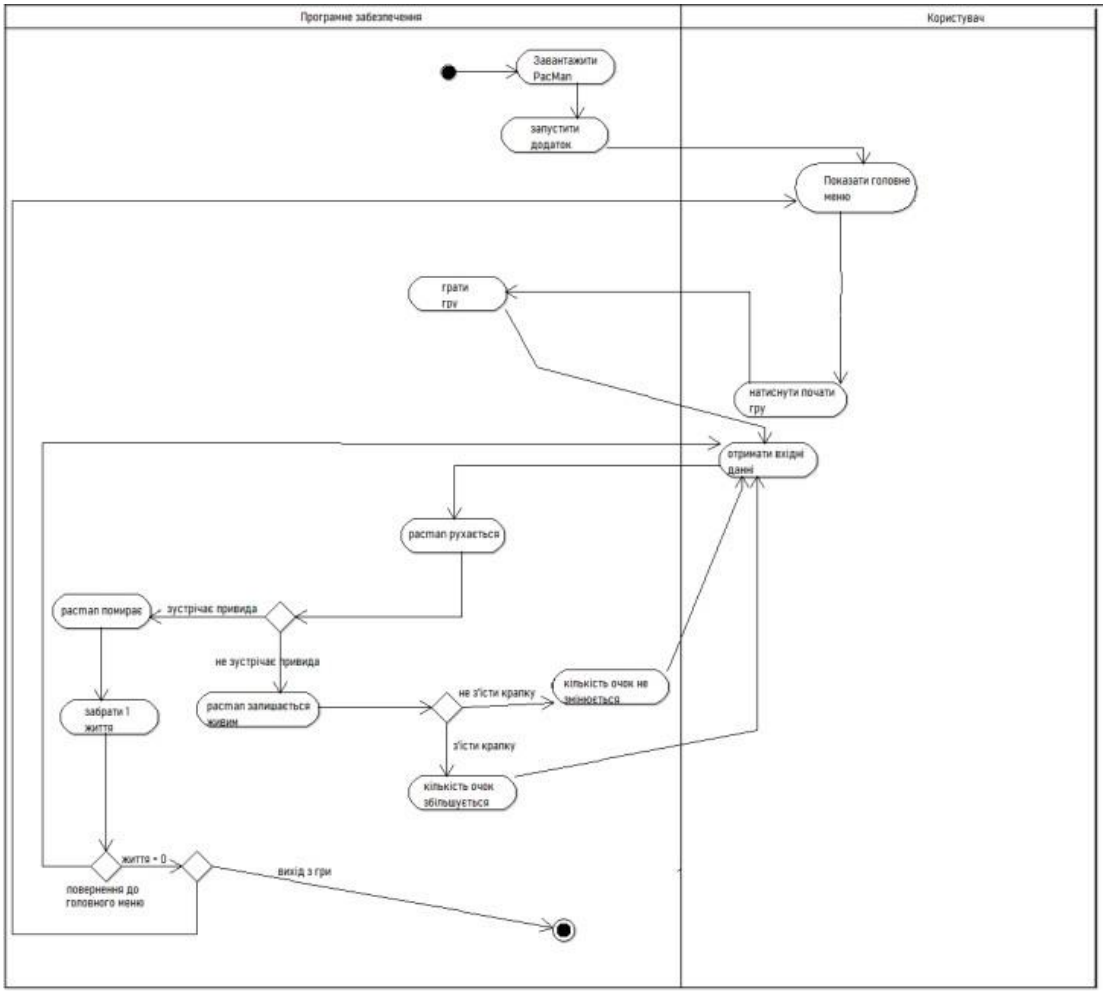

Рисунок Г.8 – Діаграма активності

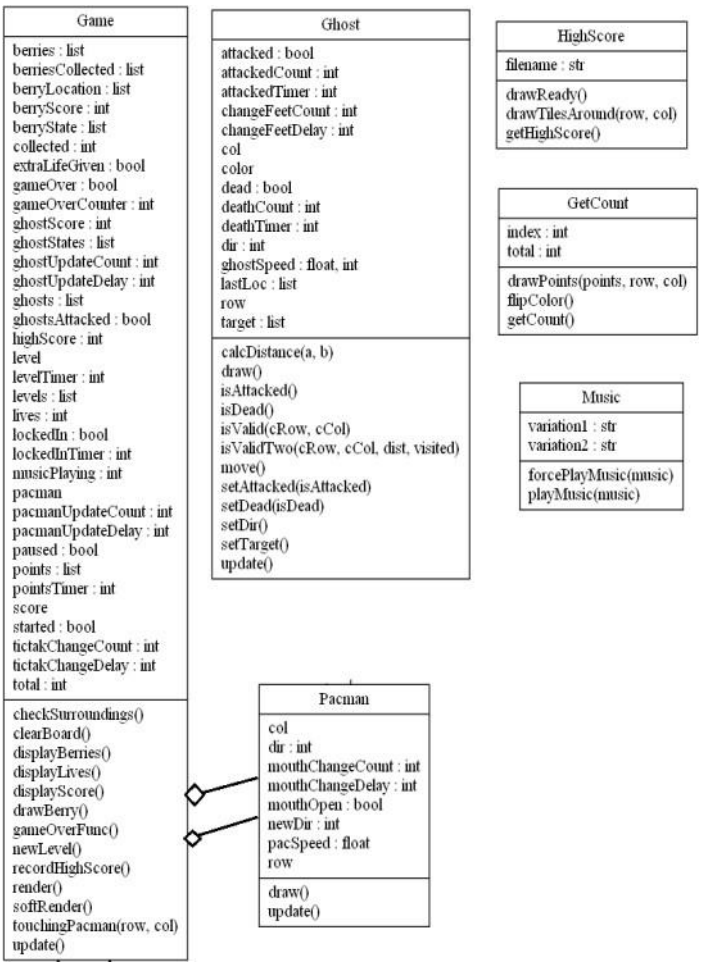

Рисунок Г.9 – Діаграма класів

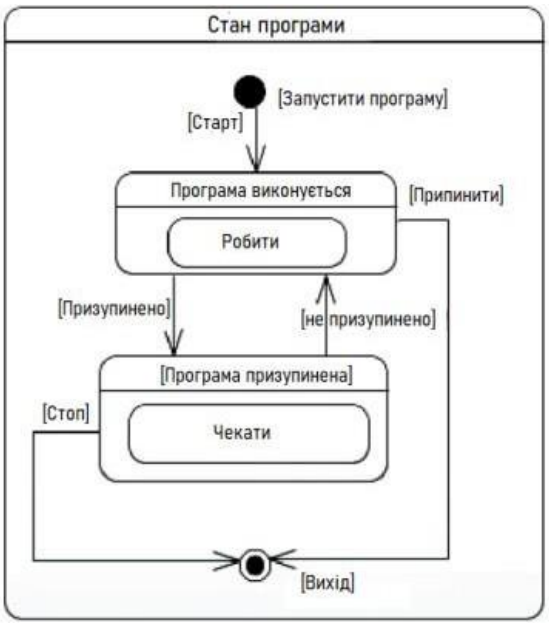

Рисунок – Г.10 Діаграма станів

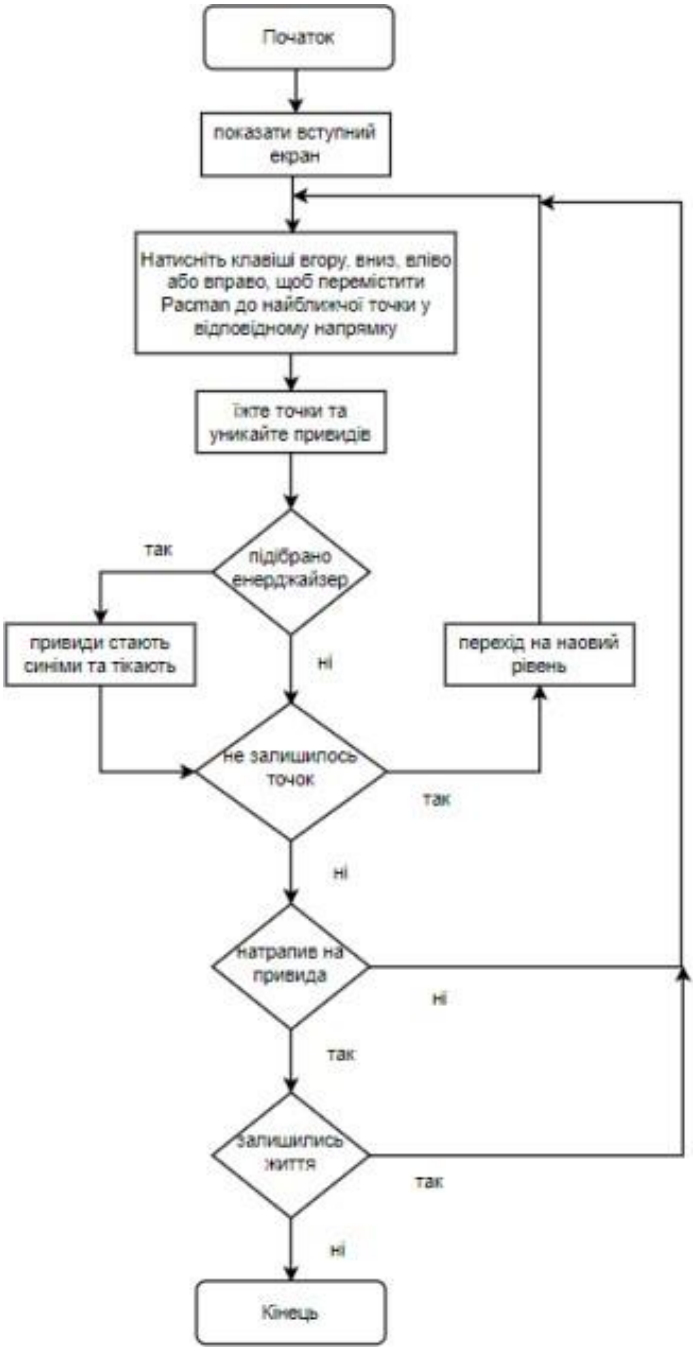

Рисунок Г.11– Основна схема функціонування гри “Pac-Man”

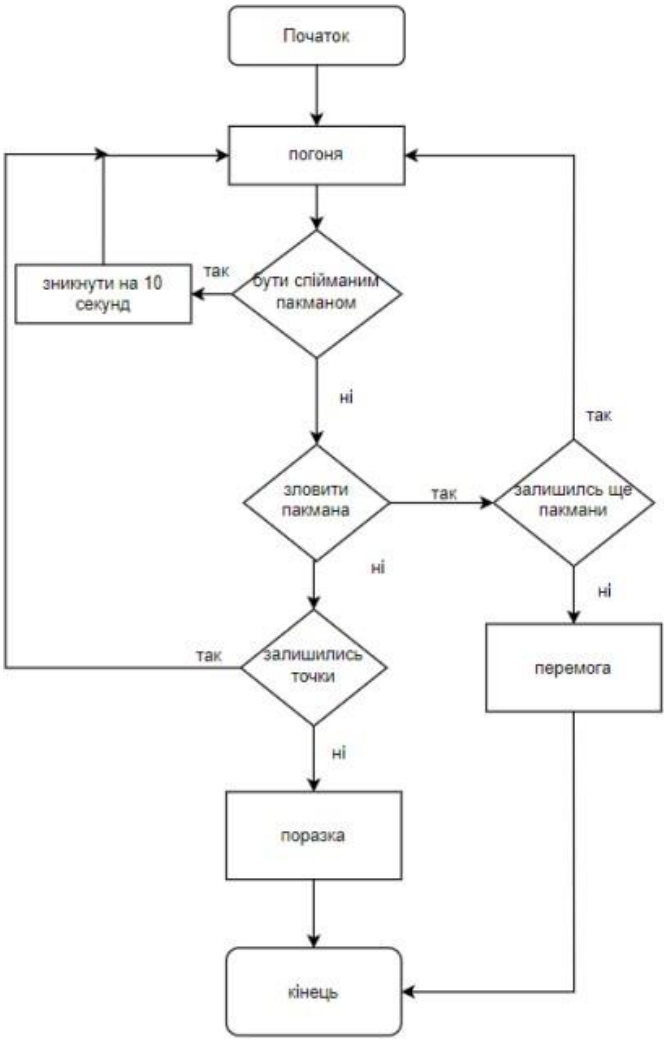

Рисунок Г.12 – Основна схема функціонування привидів у грі “Pac-Man”

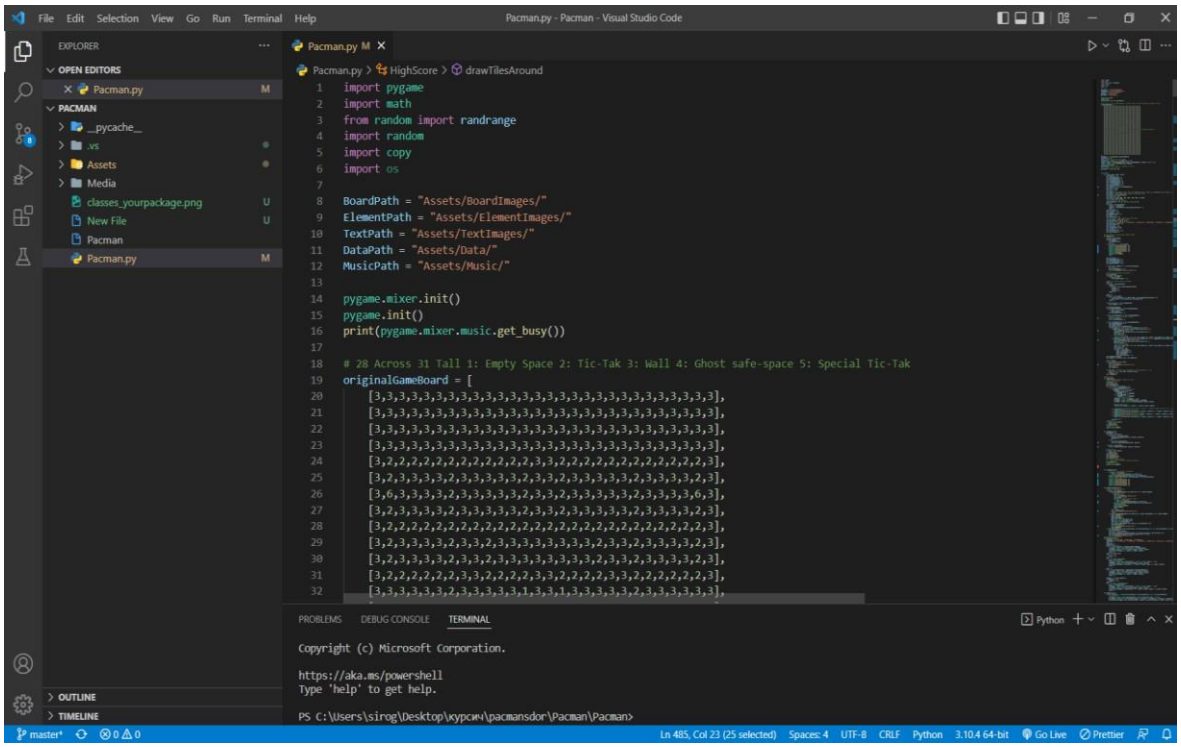

Рисунок Г.13 – Вікно середовища розробки Visual Studio Code

Рисунок Г.14 – Головне меню

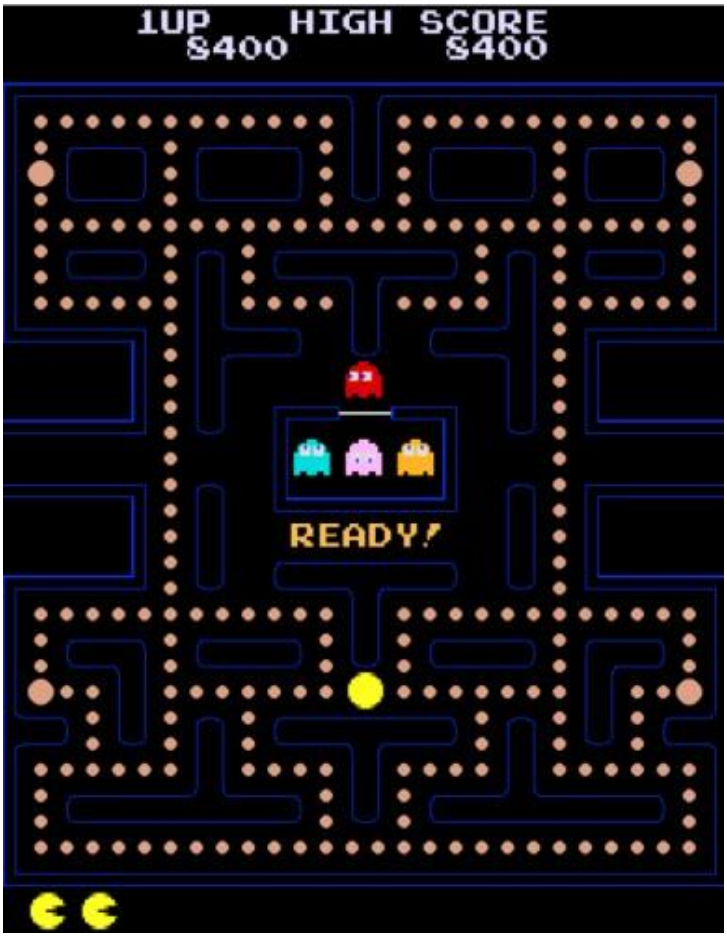

Рисунок Г.15 – Очікування дій

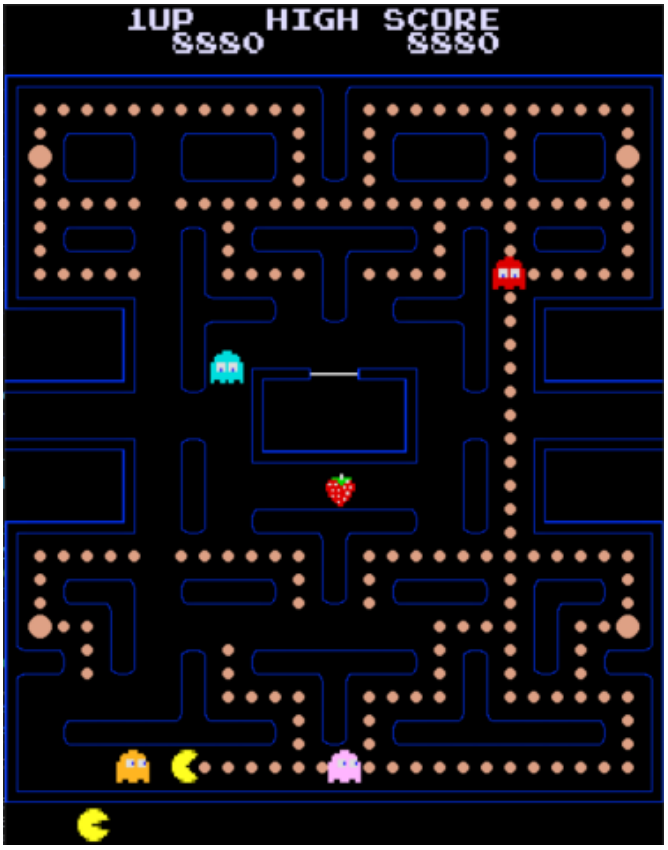

Рисунок Г.16 – Ігровий процес

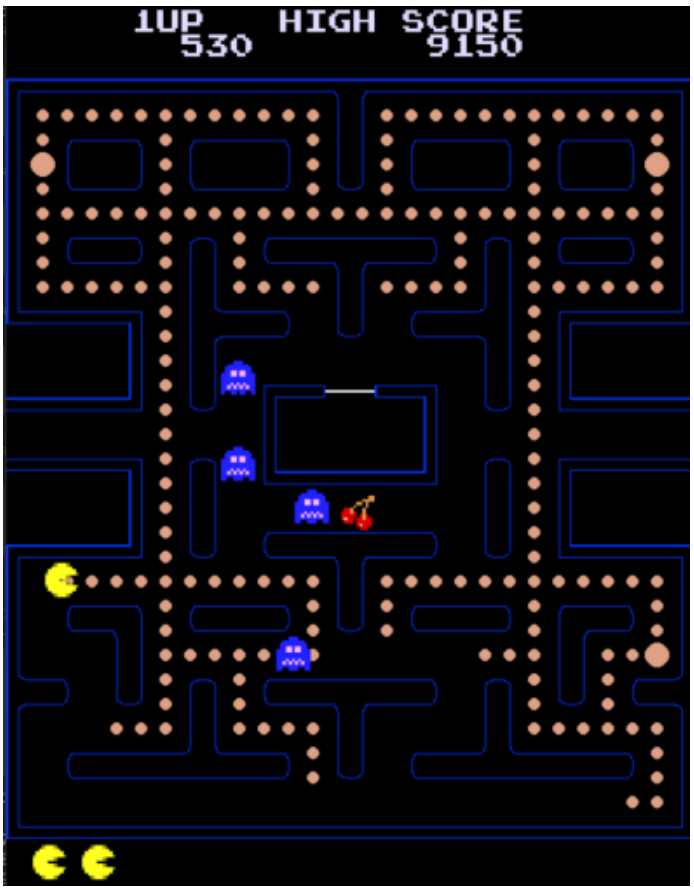

Рисунок Г.17 – Механіка зляканих привидів

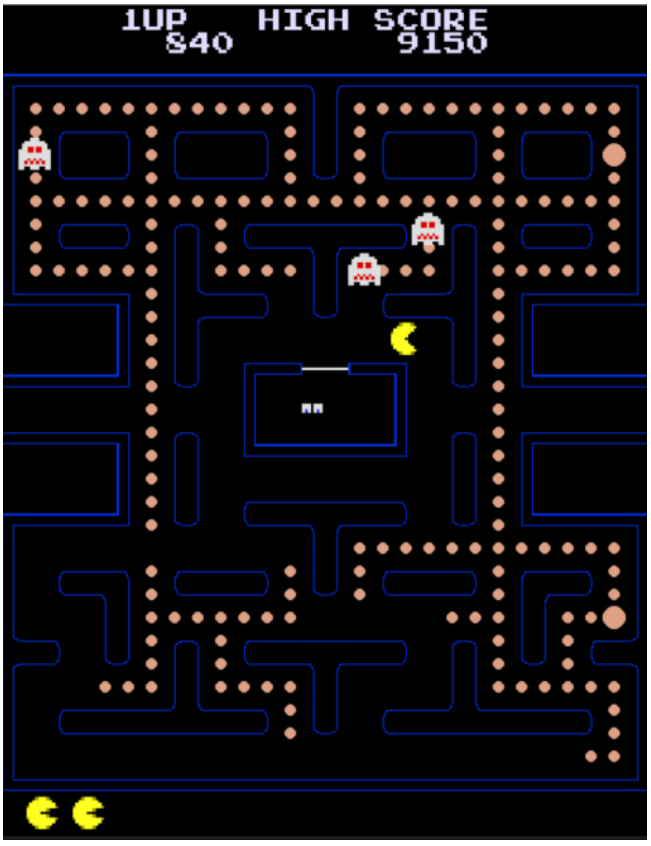

Рисунок Г.18 – завершення ефекту енерджайзера

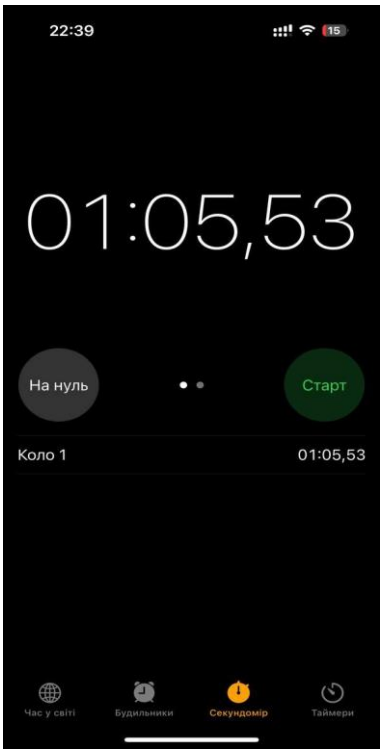

Рисунок Г.19 – Замір часу до вдосконалення алгоритмів інтелекту NPC

Рисунок Г.20 – Замір часу після вдосконалення алгоритмів інтелекту NPC

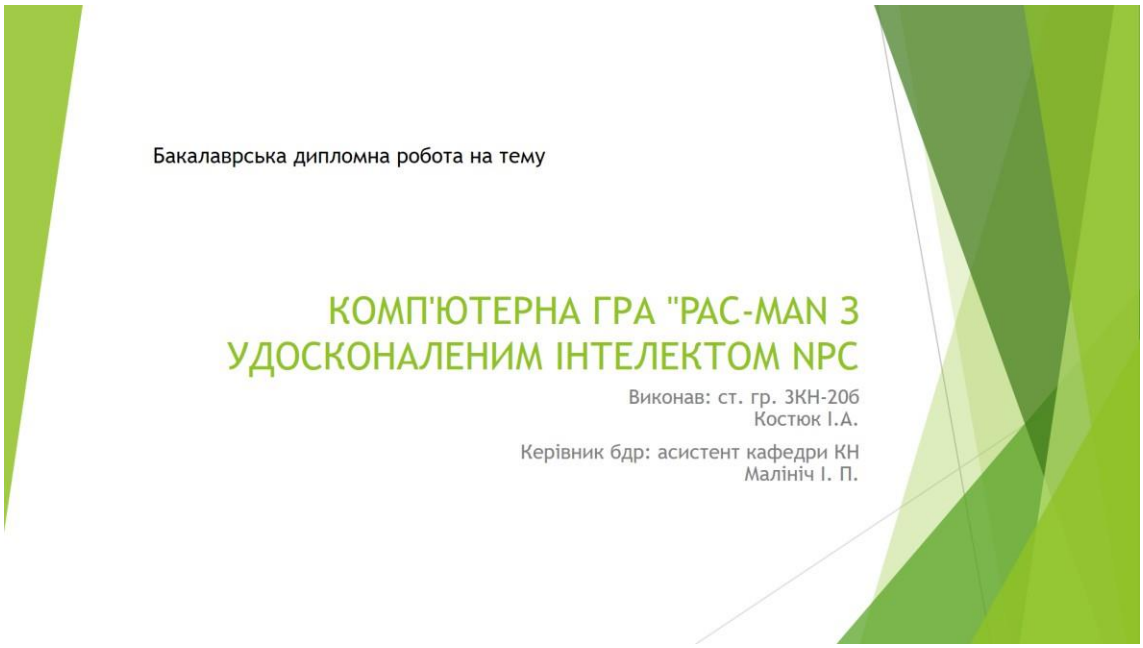

Рисунок Г.21 – Титульний слайд презентації

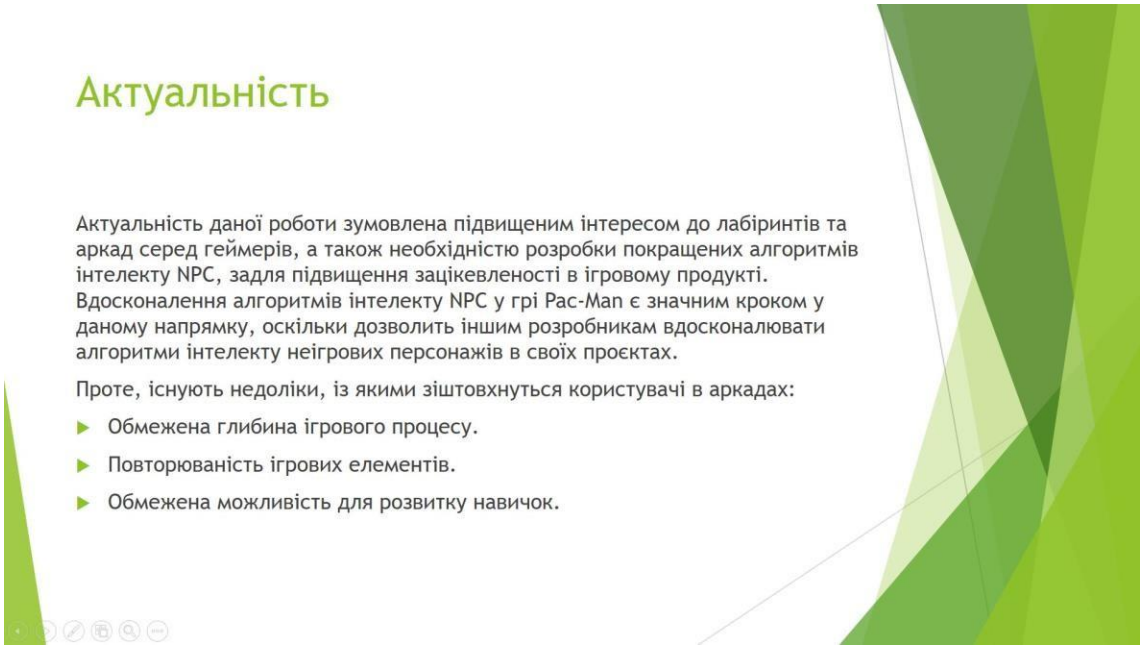

Рисунок Г.22 – Слайд презентації про актуальність

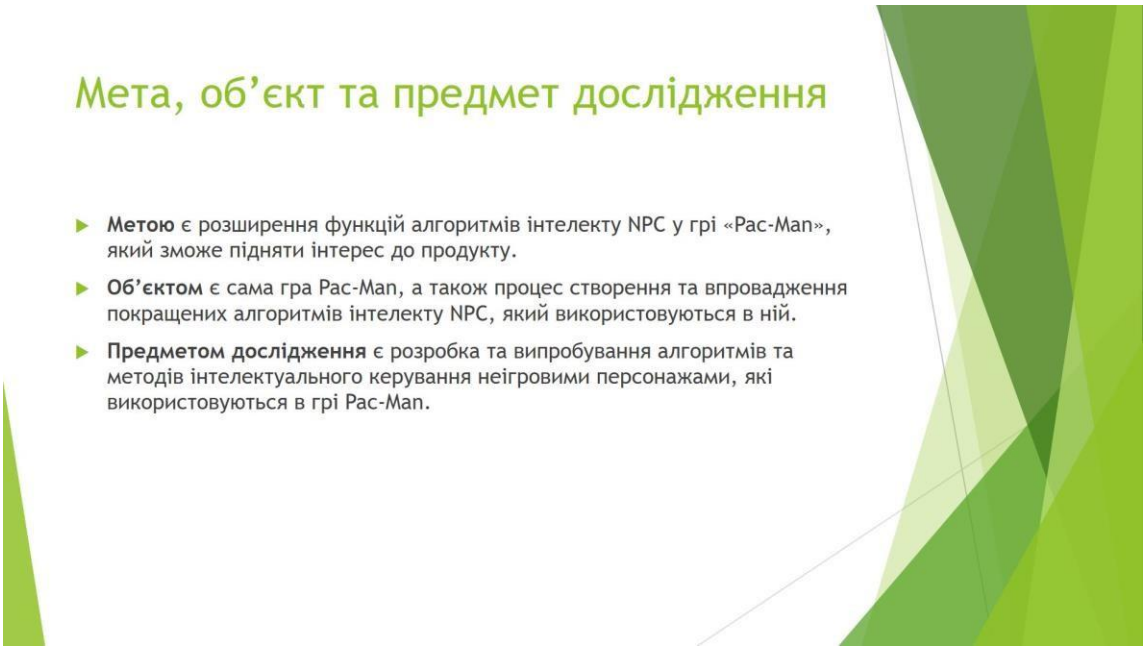

Рисунок Г.23 – Слайд презентації про мету, об’єкт та предмет дослідження

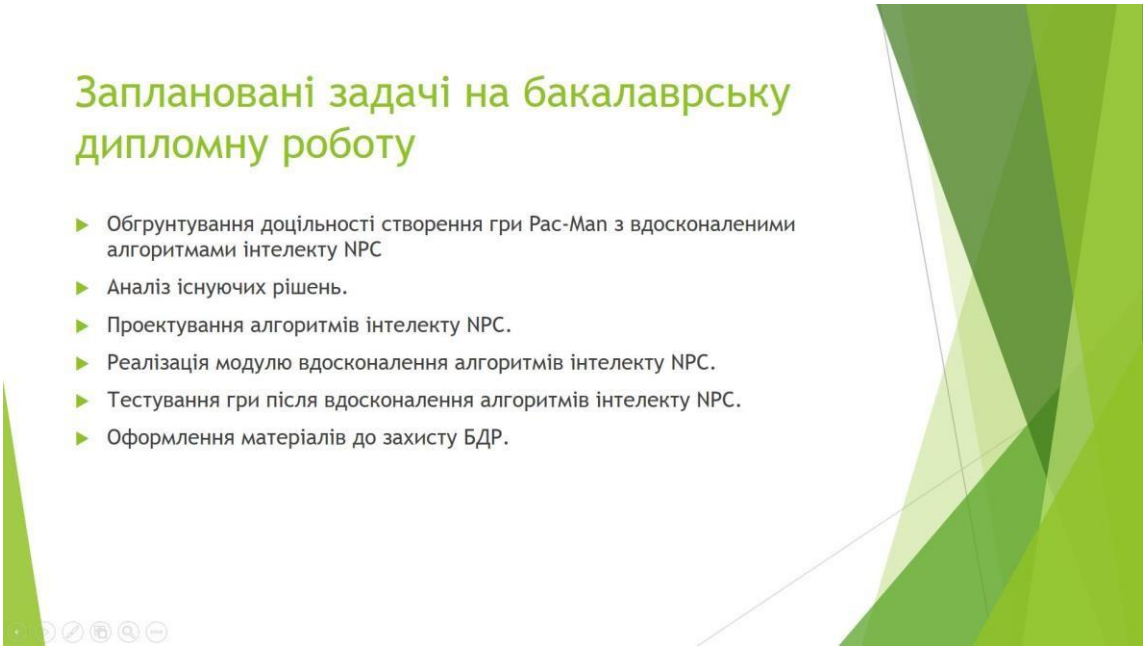

Рисунок Г.24 – Слайд презентації про заплановані задачі на бакалаврську дипломну роботу

Рисунок Г.25 – Слайд презентації про існуючі аналоги гри «Pac-Man»

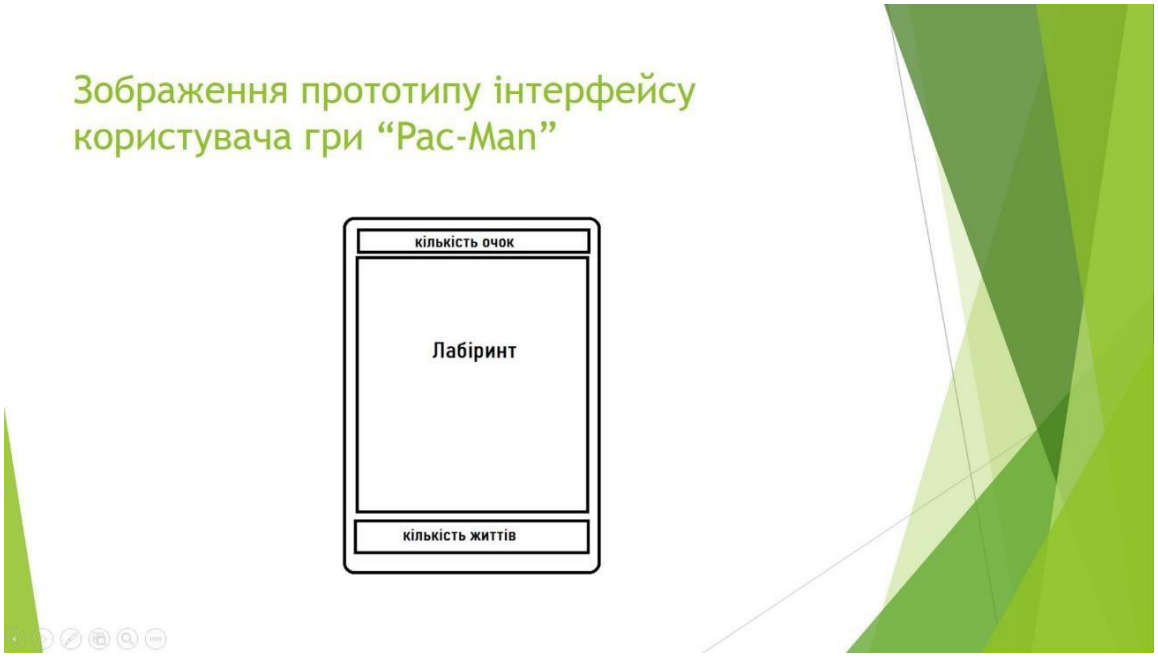

Рисунок Г.26 – Слайд презентації про зображення прототипу інтерфейсу користувача гри “Pac-Man”

Рисунок Г.27 – Слайд презентації про основну структурну схему програми

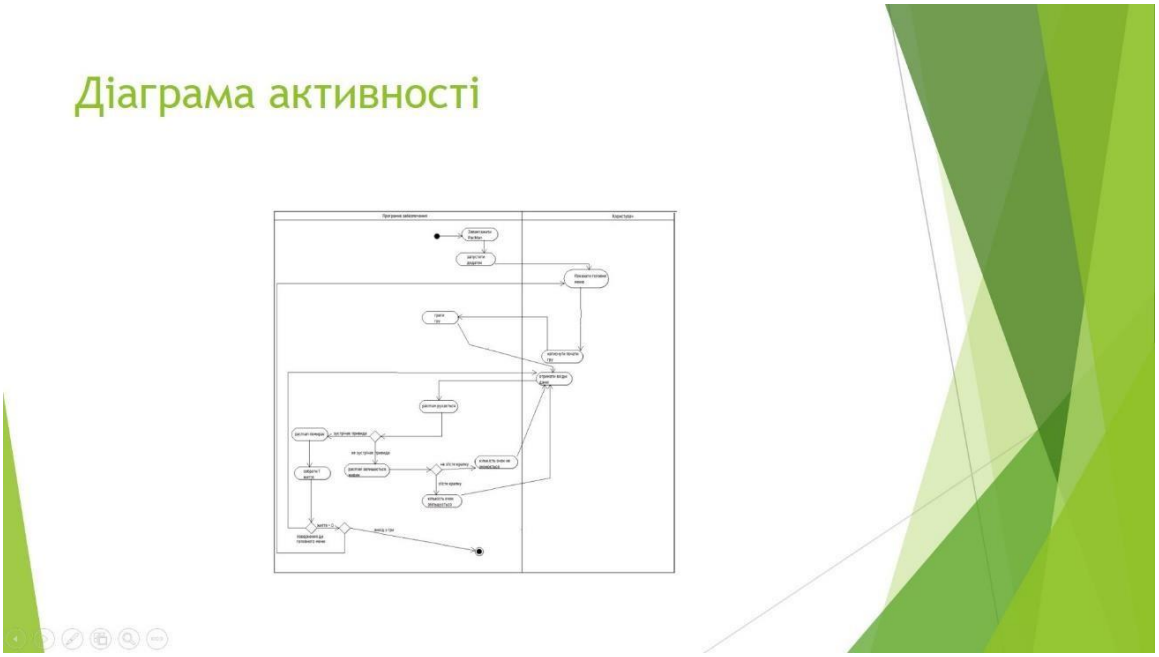

Рисунок Г.28 – Слайд презентації про діаграму активності

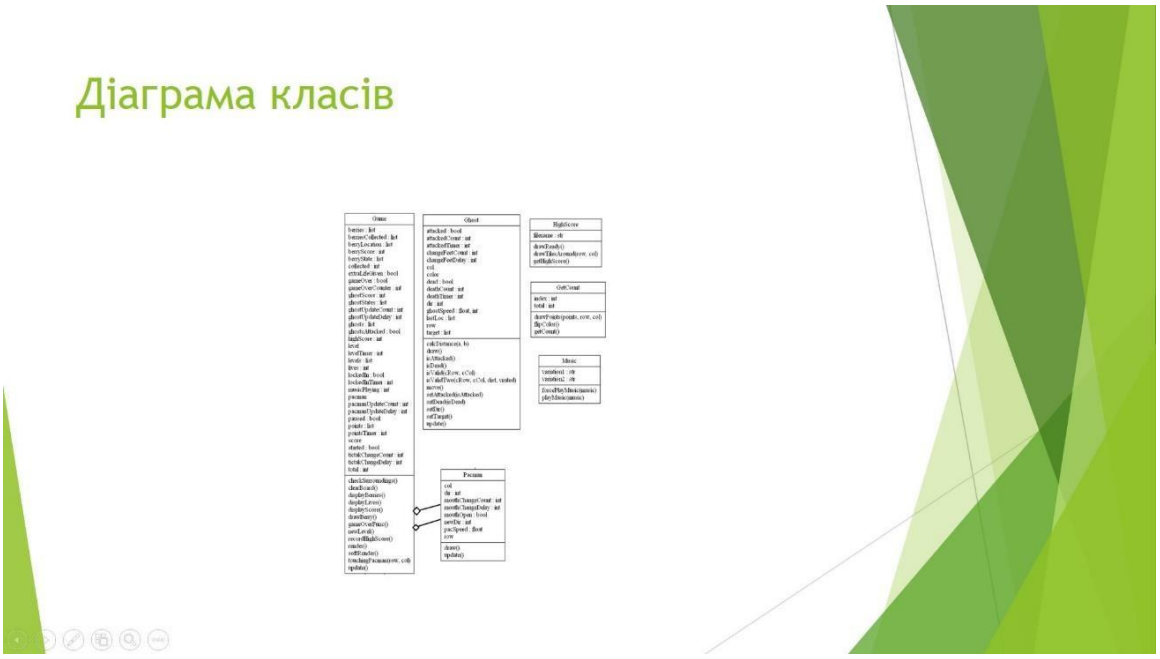

Рисунок Г.29 – Слайд презентації про діаграму класів

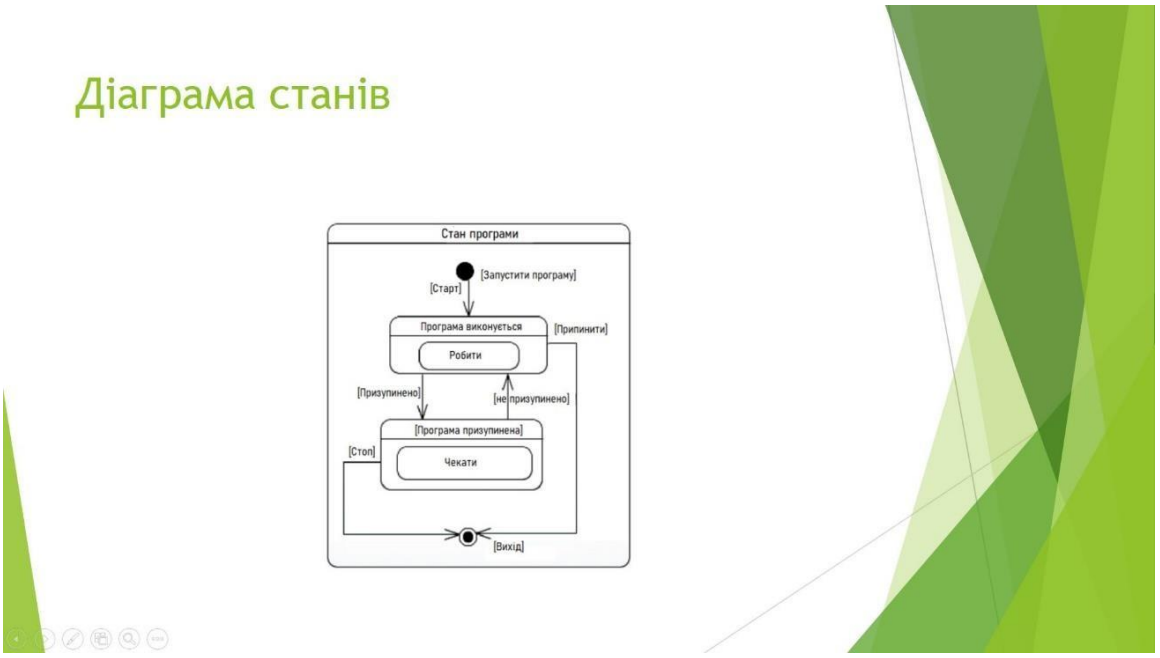

Рисунок Г.30 – Слайд презентації про діаграму станів

Рисунок Г.31 – Слайд презентації про основну схему функціонування гри “Pac-Man”

Рисунок Г.32 – Слайд презентації про основну схему функціонування привидів у грі “Pac-Man”

Рисунок Г.33 – Слайд презентації про використані технології

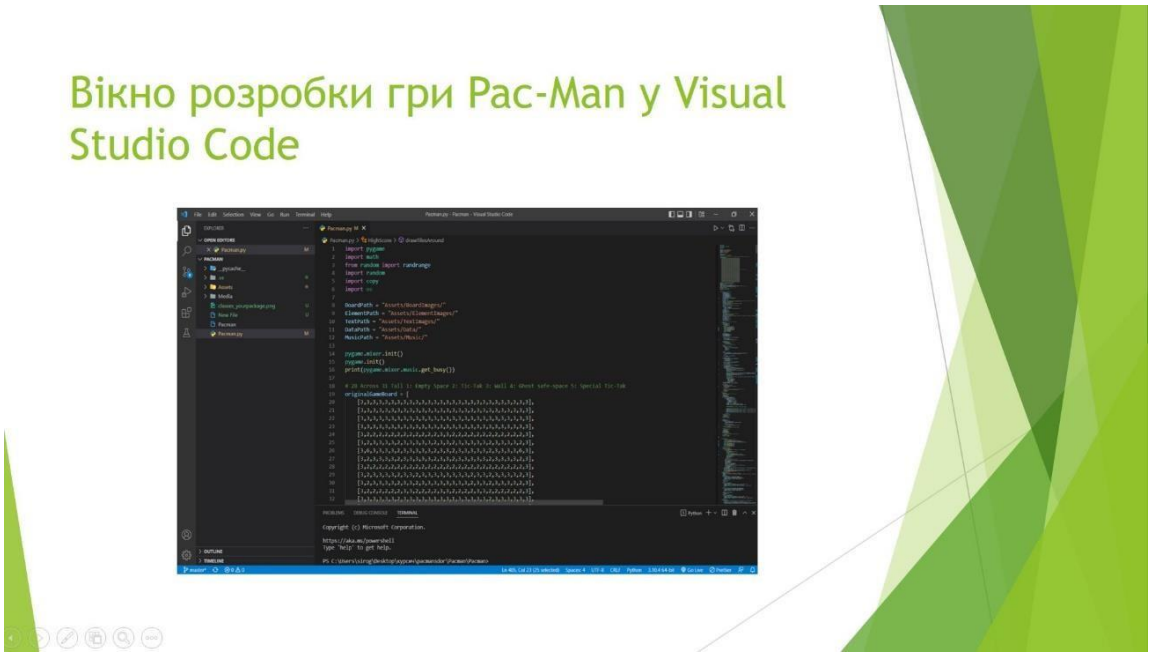

Рисунок Г.34 – Слайд презентації про вікно розробки гри Pac-Man у Visual Studio Code

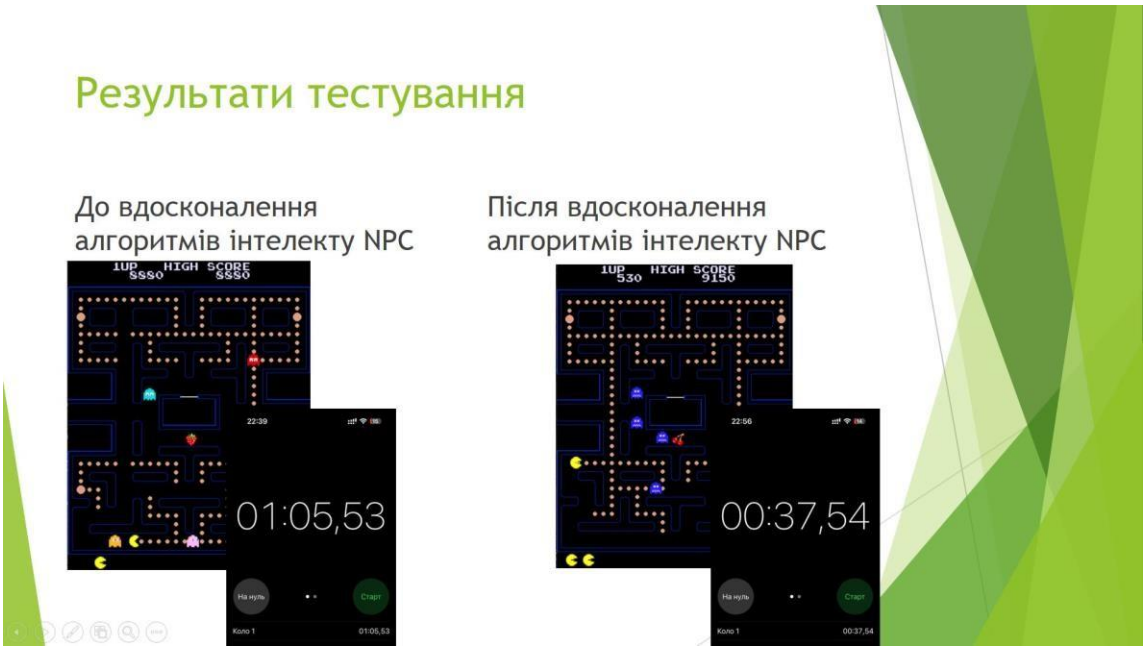

Рисунок Г.35 – Слайд презентації про результати тестування

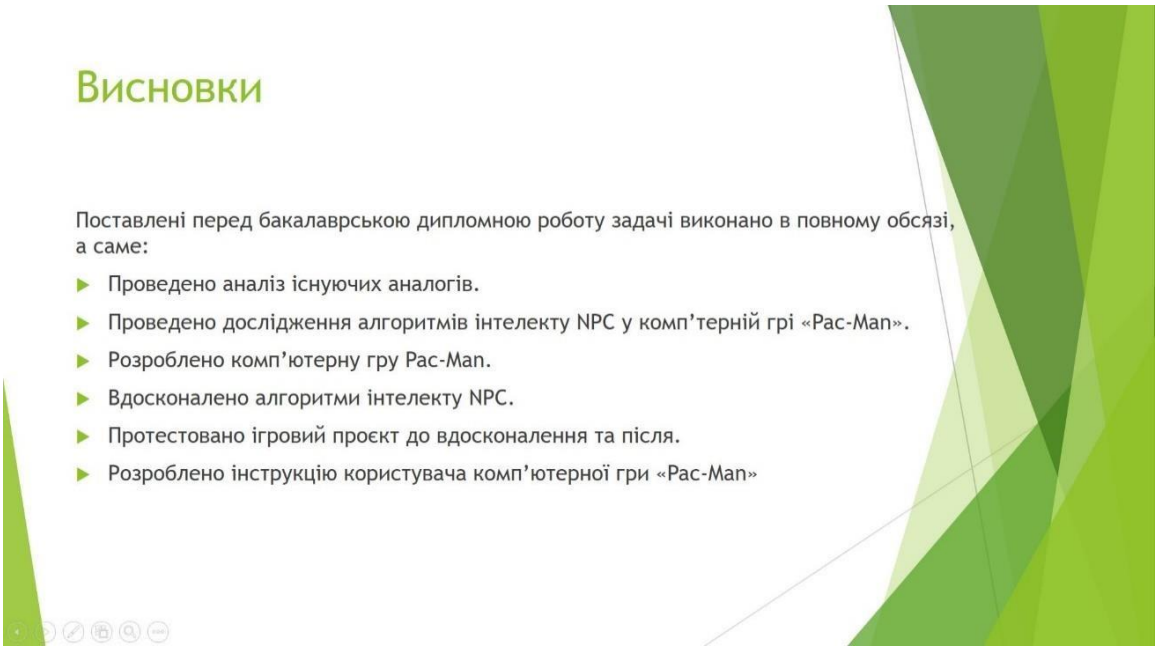

Рисунок Г.36 – Слайд презентації про висновки

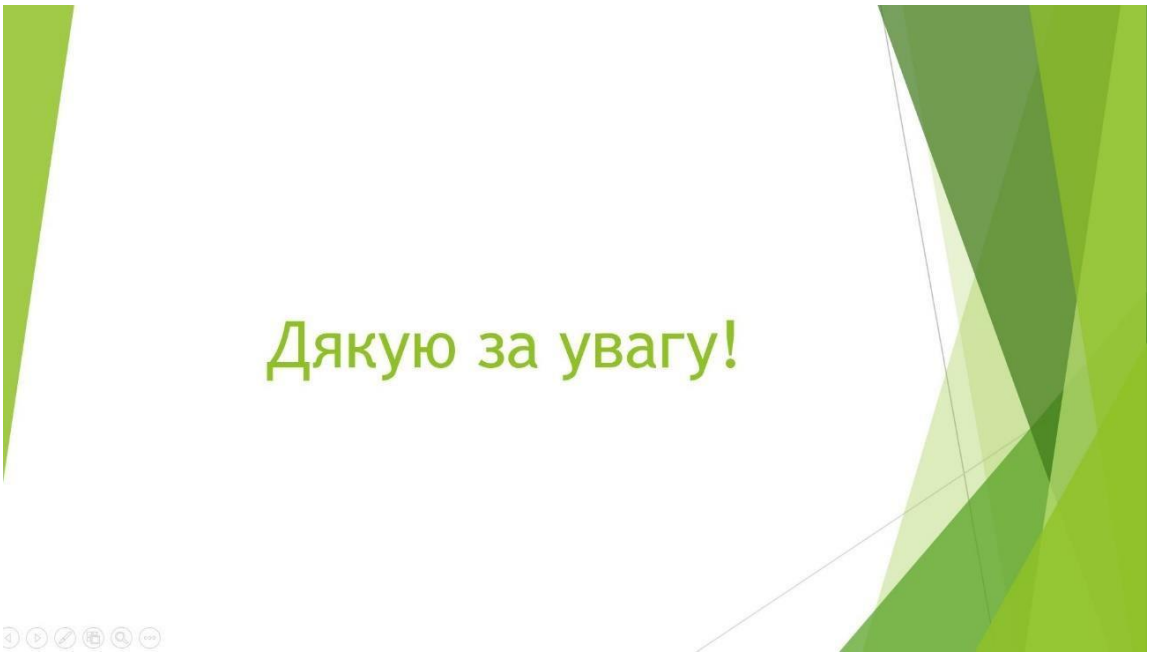

Рисунок Г.37 – Останній слайд

ДОДАТОК Д

Інструкція користувача

Для керування Pac-Man використовуйте клавіші зі стрілками на клавіатурі: вгору, вниз, вліво та вправо, або клавіші WASD для аналогічних дій. Щоб розпочати гру, натисніть пробіл. Якщо гравцю потрібно зробити паузу в грі, натисніть клавішу "P". Щоб продовжити гру після паузи, необхідно натиснути "P" ще раз. Необхідно використовувати ці клавіші для навігації Pac-Man по лабіринту, збираючи крапки та уникаючи привидів.