Міністерство освіти і науки України

Луцький національний технічний університет

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ГРИ PACMAN З ГРАФІЧНИМ ІНТЕРФЕЙСОМ, НА БАЗІ ШТУЧНОГО ІНТЕЛЕКТУ

DEVELOPMENT OF THE PACMAN GAME WITH A GRAPHICAL INTERFACE, BASED ON ARTIFICIAL INTELLIGENCE

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗ-42
Баландюк Андрій Олександрович
_____
(підпис)

Керівник:
д.т.н., професор
Андрущак Ігор Євгенович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«1» 06 2024 р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«2» 02 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Баландюку Андрію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *Розробка гри Растап з графічним інтерфейсом, на базі штучного інтелекту*

Керівник роботи: *д.т.н., професор, Андрущак Ігор Євгенович*
затверджені наказом закладу вищої освіти від «*30» грудня 2023 р.* № *477/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення, опис функціонального наповнення об'єкта проєктування, практична реалізація об'єкта проєктування.

5. Перелік графічного матеріалу: *17 рисунків, 3 таблиці.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Андрущак І. Є.*		
Специфікації вимог до розробленої системи	*Андрущак І. Є.*		
Розробка об'єкта проєктування	*Андрущак І. Є.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	4,4 %		
Академічна доброчесність	*Ящук А. А.*		

7. Дата видачі завдання *«2» лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*07.02.2024 р.*	
2	*Аналіз проблеми розробки та впровадження об'єкту проєктування*	*27.02.2024 р.*	
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*12.03.2024 р.*	
4	*Розробка функціонально-структурної схеми роботи об'єкта проєктування*	*17.04.2024 р.*	
5	*Практична реалізація об'єкта проєктування*	*18.05.2024 р.*	
6	*Тестування та налагодження об'єкта проєктування*	*01.06.2024 р.*	
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*05.06.2024 р.*	

Здобувач вищої освіти _____ (підпис) *Баландюк А. О.* (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) *Андрущак І. Є.* (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу

Здобувач вищої освіти: *Баландюк Андрій Олександрович*

Тема: *«Розробка гри Растап з графічним інтерфейсом, на базі штучного інтелекту»*

Коротка характеристика кваліфікаційної роботи: *Кваліфікаційна робота бакалавра складається з трьох розділів, в яких було проведено аналіз сучасного стану проблеми, чітко поставлено завдання, описана структура гри та методи її реалізації, здійснено розробку гри, проведено тестування, а також описано способи використання розробленого методу у робототехніці.*

Самостійні розробки і пропозицій автора: *Автором був самостійно розроблений інтелектуальний агент на базі штучного інтелекту.*

Практичне значення роботи *Полягає в ефективному використанні методів штучного інтелекту при розробці відеоігор.*

Недоліки: *Суттєвих недоліків в роботі не виявлено.*

Загальний висновок: *Кваліфікаційна робота бакалавра виконана на високому рівні. Матеріал викладено чітко і зрозуміло. Вважаю, що кваліфакаційна робота відповідає усім вимогам і заслуговує оцінки «відмінно», а її авторові, студенту Баландюку А.О., може бути присвоєний кваліфікаційний рівень "бакалавр" зі спеціальності «Інженерія програмного забезпечення».*

Рецензент: Кошелюк В.Т., к.т.н., доцент кафедри комп'ютерних наук

Рецензент кваліфікаційної роботи _____
(підпис)

«07» 06 2024 р.

Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

ВІДГУК
КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач вищої освіти *Баландюк Андрій Олександрович*
(прізвище, ім'я, по батькові)
група *ІПЗ-42*

Тема кваліфікаційної роботи *«Розробка гри Растап з графічним інтерфейсом, на базі штучного інтелекту»*

Керівник *Андрущак Ігор Євгенович*

Актуальність теми *Робота актуальна та своєчасна. Штучний інтелект це досить нова, та ще мало досліджена тема, ігри будуть завжди популярні для людей різного віку. Розробка інтелектуального агенту з використанням штучного інтелекту є актуальною наразі.*

Об'єкт дослідження *Інтелектуальний агент для гри на базі штучного інтелекту*

Характеристика теоретичного рівня, наявності самостійних розробок і практичної значимості роботи *Автор аналізує різні види застосування штучного інтелекту в індустрії відеоігор. Визначає їх переваги та недоліки, обґрунтовує свої рішення. Чітко описує структуру гри та засоби для її реалізації. Зміст роботи повністю відповідає обраній темі. До позитивних ри кваліфікаційної роботи бакалавра можна віднести системність та послідовність викладення матеріалу, а також якісна практична реалізація.*

Зауваження та недоліки: *Суттєвих недоліків в роботі не виявлено.*

Загальний висновок: *Кваліфікаційна робота бакалавра виконана на високому рівні. Всі вимоги було дотримано, а робота описана логічно та послідовно. Пояснювальна записка відповідає усім стандартам оформлення.*

Керівник _____ **(Андрущак І. Є.)**
(підпис) (прізвище, ім'я, по батькові)

«04» 06 2024 р.

Міністерство освіти і науки України

Луцький національний технічний університет

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ГРИ PACMAN З ГРАФІЧНИМ ІНТЕРФЕЙСОМ, НА БАЗІ ШТУЧНОГО ІНТЕЛЕКТУ

DEVELOPMENT OF THE PACMAN GAME WITH A GRAPHICAL INTERFACE, BASED ON ARTIFICIAL INTELLIGENCE

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗ-42
Баландюк Андрій Олександрович
_____
(підпис)

Керівник:
д.т.н., професор
Андрущак Ігор Євгенович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» _____ 2024 р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна

_____
(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Баландюку Андрію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *Розробка гри Pacman з графічним інтерфейсом, на базі штучного інтелекту*

Керівник роботи: *д.т.н., професор, Андрущак Ігор Євгенович*
затверджені наказом закладу вищої освіти від «*30*» *грудня 2023 р.* № *477/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу*.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення, опис функціонального наповнення об'єкта проєктування, практична реалізація об'єкта проєктування.

5. Перелік графічного матеріалу: *17 рисунків; 3 таблиці*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Андрущак І. Є*		
Специфікації вимог до розробленої системи	*Андрущак І. Є*		
Розробка об'єкта проєктування	*Андрущак І. Є*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	*___%*		
Академічна доброчесність	*Яцук А. А.*		

7. Дата видачі завдання *«2» лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*07.02.2024 р.*	
2	*Аналіз проблеми розробки та впровадження об'єкту проєктування*	*27.02.2024 р.*	
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*12.03.2024 р.*	
4	*Розробка функціонально-структурної схеми роботи об'єкта проєктування*	*17.04.2024 р.*	
5	*Практична реалізація об'єкта проєктування*	*18.05.2024 р.*	
6	*Тестування та налагодження об'єкта проєктування*	*01.06.2024 р.*	
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*05.06.2024 р.*	

Здобувач вищої освіти _____ *Баландюк А. О.*
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ *Андрущак І. Є.*
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Баландюк А. О. Розробка гри Pacman з графічним інтерфейсом, на базі штучного інтелекту. Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі проведено аналіз сучасного стану проблеми використання штучного інтелекту при розробці відеоігор і сформульовано завдання. В другому розділі визначено вимоги до розроблюваної системи, а також засоби, методи і алгоритми розробки. У третьому розділі розроблено гру Pacman, розроблено метод штучного інтелекту, проведено тестування, дотримано вимог безпеки та наведено приклад використання розробленого методу у робототехніці. У висновках узагальнено інформацію, відображену у попередніх частинах. Результати розробки демонструють можливості використання методів штучного інтелекту не тільки у відеоіграх, а й в інших сферах діяльності.

Ключові слова: штучний інтелект, гра, метод алгоритм, аналіз, тестування, графіка, python.

ABSTRACT

Balandiuk A. O. Development of the Pacman Game with a Graphical Interface, Based on Artificial Intelligence. Manuscript.

Bachelor's qualifying thesis of the educational program "Software Engineering". Lutsk National Technical University. Lutsk, 2024.

The bachelor's qualifying thesis consists of an introduction, three sections, conclusions, a list of used sources and annexes.

In the first chapter, an analysis of the current state of the problem of using artifician intellegence in video game development and the task was formulated. The second section defines the requirements for the developed system, as well as means, methods and development algorithms. In the third chapter, the Pacman game is developed, an artificial intelligence method is designd, testing is conducted, safety requirements are met, and an example of the developed method's application in robotics is provided. The conclusions summarize the information presented in the previous parts. The development results demonstrate the possibilities of using artifician intelligence methods not only in video games but also in other areas of activity.

Keywords: artificial intelligence, game, method, algorithm, analysis, testing, graphics, Python.

ЗМІСТ

ВСТУП

Актуальність теми. Штучний інтелект – елемент інформатики, він спеціалізується на виконаннях роботи, котра потребує людського інтелекту. Основні задачі для штучного інтелекту включають: здатність до навчання, аналіз ситуації та прийняття рішень для вирішення проблем.

Штучний інтелект відіграє велику роль у розвитку індустрії відеоігор, з його допомогою розробники створюють складних суперників, які можуть адаптуватися до дій гравця та різними методами протидіяти. Супротивники зі штучним інтелектом забезпечують захоплюючий ігровий досвід, який забезпечує гравців незабутніми враженнями від гри.

Актуальність цієї роботи зумовлена малим рівнем дослідженності штучного інтелекту, що дає можливості для його аналізу та застосування.

Метою кваліфікаційної роботи бакалавра є розробка гри з використанням штучного інтелекту, що реагує на дії гравця та додаткові способи його застосування.

Для досягнення мети потрібно виконати наступні завдання:

- провести аналіз поточного стану проблеми;
- здійснити вибір засобів та методів розробки;
- спроєктувати структуру гри;
- розробити графічний дизайн гри;
- реалізувати поведінку під управлінням штучного інтелекту;
- проаналізувати ефективності різних моделей поведінки;
- забезпечити цілісність та безпеку;
- провести аналіз використання розробленого методу у робототехніці.

Об’єкт дослідження: розробка інтелектуального ігрового агента для гри, на базі штучного інтелекту.

Предмет дослідження: інтелектуальний агент.

Практична цінність розробки: використання методів штучного інтелекту у відеоіграх.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНОГО СТАНУ ПРОБЛЕМИ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В ІНДУСТРІЇ ВІДЕОІГОР І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Розробка ігор є однією з найбільш швидкозростаючих галузей програмування та інформаційних технологій загалом. З моменту появи перших аркадних ігор у 70-х і 80-х роках минулого століття, ігрова індустрія зазнала значних змін. Сучасні ігри є високотехнологічними продуктами, що вимагають значних інвестицій, часу та кваліфікації. Станом на сьогодні, мільйони людей по всьому світу залучені до створення ігор, вони охоплюють різні платформи, від мобільних пристроїв до потужних ігрових консолей та персональних комп’ютерів.

Що стосується розвитку гри «Pacman», то вона була вперше випущена компанією «Namco» у 1980 році. Гра швидко здобула популярність завдяки своїй інноваційній ігровій механіці та яскравій графіці. Головний герой гри, повинен з’їдати всі точки на рівні, уникаючи зустрічі з привидами. Успіх гри зумовив появу численних клонів та її варіацій, а також вплинув на розвиток індустрії в цілому (рис. 1.1).

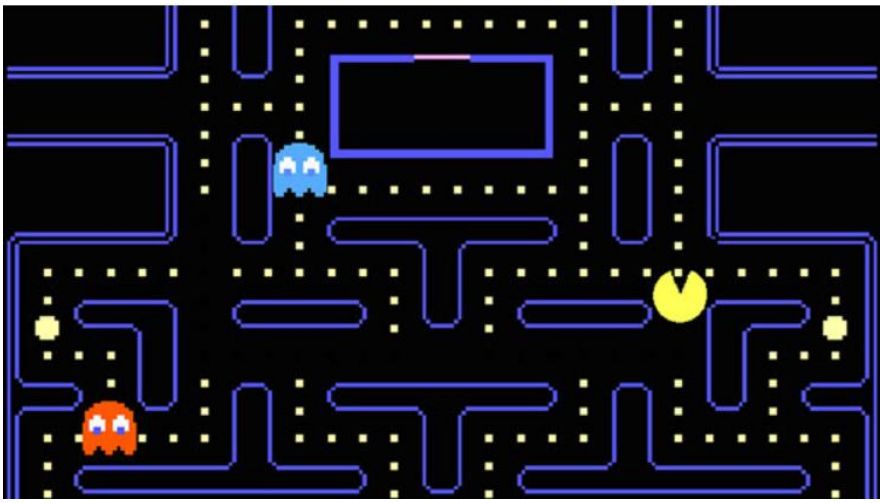

Рисунок 1.1 – Pacman 1980 [1]

На даний момент ігри з елементами штучного інтелекту стають популярними, оскільки вони дозволяють створювати більш складні та реалістичні сценарії. Штучний інтелект використовується для моделювання поведінки персонажів, прийняття рішень у реальному часі, адаптації до дій гравця та інших аспектів ігрового процесу. У випадку з «Pacman», методи штучного інтелекту можуть бути використані для управління поведінкою головного героя та привидів, що переслідують його.

Штучний інтелект в іграх має свої особливості, до переваг можна віднести:

– ігри можуть підлаштовувати рівні складності під навички гравця, це забезпечує баланс і підвищує задоволення від гри;

– персонажі в іграх можуть демонструвати поведінку, що буде схожою на людську, вона буде реагувати на обставини та приймати складні рішення;

– штучний інтелект може генерувати великі та різномантні карти, це збільшить масштаб гри та її різноманітність;

– також штучний інтелект може аналізувати поведінку гравців та їх вподобання, це допомагає створювати ігри для більшої кількості гравців.

Проте, штучний інтелект має і свої недоліки, до них можна віднести:

– вимоги до великої кількості вкладень, потрібні спеціалісти та багато фінансових ресурсів;

– ігри можуть стати менш унікальними, якщо розробники будуть використовувати однакові методи штучного інтелекту;

– штучний інтелект вимагає великих обчислювальних ресурсів, тому потрібно мати потужні пристрої для розробки;

– розробники можуть зловживати штучним інтелектом і на основі зібраних ним данних, сприяти надмірним витратам гравців та призвести до залежності.

Також важливим аспектом є використання методів машинного навчання, що дозволяють персонажам гри адаптувати свою поведінку на основі досвіду, який вони отримали під час проходження великої кількості тестів. Це може

значно покращити ігровий процес, зробити його більш непередбачуваним та цікавим для гравця.

Проаналізуємо приклади штучного інтелекту, що є у вільному доступі. Розглянемо штучний інтелект «Stockfish», який використовується у грі в шахи.

«Stockfish» – це найпотужний шаховий рушій у світі. Він розроблемний для аналізу шахових партій, надання рекомендацій для навчання як нових, так і досвідчених гравців.

Його основні можливості:

– використання сучасних алгоритмів та методів оцінки позицій, це дозволяє йому проводити мільйони ходів в секунду та робити точні прогнози розвитку партій;

– він використовує методи, що дозволяють йому визначати лише ходи, що матимуть позитивний результат;

– «Stockfish» можна налаштувати під різні рівні складності, котрі допоможуть у навчанні початківців і професіоналів.

Є велика проблема, яка не дозволяє використовувати «Stockfish» для інших задач – це його алгоритми, вони налаштовані лише для шахмат і не можуть застосовуватись у інших іграх [2].

Також проаналізуємо складну систему штучного інтелекту з назвою «AI Director».

«AI Director» – це метод, який використовується для створення динамічного і адаптивного ігрового середовища у відеоіграх.

Її основні функції включають:

– постійне відслідкування дій та стану гравця, такі як позиція на карті, рівень здоров’я та боєприпаси;

– можливость змінювання складаності гри з огляду на навички гравця;

– генерує випадкові події, які додають у ігровий процес багато різноманіття;

– для підтримки атмосфери гри, змінює аудіо та візуальні ефекти.

Метод «AI Director» потребує великої кількості вичислювальних

можливостей, тому він є хорошим варіантом тільки для великих розробок [3].

Окрім технічних аспектів, важливо також враховувати користувацький досвід та забезпечити зручність використання гри. Інтуїтивно зрозумілий інтерфейс, плавна анімація та зручне керування є ключовими факторами, що впливають на задоволеність гравців.

Загалом, створення інтелектуального агента для гри «Pacman» є перспективним напрямком, оскільки він представляє собою приклад використання сучасних технологій, таких як штучний інтелект, у одній із найшвидше зростаючих сфер – ігровій розробці. Використання нових технологій, методів штучного інтелекту та графічного дизайну забезпечує можливість створення гри, яка відповідає сучасним вимогам та очікуванням гравців.

Сучасний стан розробки ігор, завдяки доступності інструментів та підтримки спільноти, є досить сприятливим. Використання штучного інтелекту відкриває нові можливості для створення інноваційних та захоплюючих ігрових проєктів. Проте, для досягнення успіху, необхідно ретельно планувати процес розробки та постійно вдосконалювати свої знання та навички.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Після проведення аналізу, для кваліфікаційної роботи бакалавра були поставлені наступні завдання:

- провести аналіз поточного стану проблеми;
- здійснити вибір засобів та методів розробки;
- спроєктувати структуру гри;
- розробити графічний дизайн гри;
- реалізувати поведінку під управлінням штучного інтелекту;
- проаналізувати ефективності різних моделей поведінки;
- забезпечити цілісність та безпеку;
- провести аналіз використання розробленого методу у робототехніці.

Висновки до розділу 1

Було здійснено аналіз різних методів штучного інтелекту, які застосовуються у відеоіграх. Основна мета їх застосування полягає у наданні нових можливостей, що забезпечить гравців захоплюючим ігровим досвідом. Завдяки методам штучного інтелекту, ігри можуть підлаштувати рівень складності під навички гравця, створюючи більш реалістичні ігрові ситуації, що сприятиме кращій оцінці гри.

Було проаналізовано дві сучасні моделі штучного інтелекту, які використовуються в індустрії відеоігор. Основною їх проблемою є обов’язковий зв’язок з системою, для якої їх розробили, та велика кількість ресурсів для налаштування та застосування.

Поставлено завдання на кваліфікаційну роботу бакалавра.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

Ідея проєкту полягає в створенні класичної гри «Pacman» з графічним інтерфейсом та реалізацію агента штучного інтелекту для керування поведінкою привидів.

Вибір гри «Pacman» був обумовлений її популярністю та значенням у світі відеоігор. Створення сучасної версії дозволить продемонструвати навички в програмуванні. А методи штучного інтелекту, що використовуватимуться у ній, можна поєднати з функціоналом інших приладів.

Метою цієї роботи є створення кращої версії старої гри. Різне поводження привидів та головного персонажа робитиме кожну гру унікальною. Штучний інтелект, для керування привидами, буде реалізовано за допомогою алгоритмів пошуку та прийняття рішень.

Основні цілі системи включають:

– реалізація штучного інтелекту, що забезпечить нові виклики та непередбачуваність;

– надання гравцям можливості налаштовувати гру за допомогою різних макетів карт;

– впровадження графічного інтерфейсу, який буде привабливим у використанні.

Наведемо умови до програмного забезпечення.

Ігровий процес:

– ініціалізація гри з різними макетами ігрових полів;

– рух головного героя та супротивників по ігровому полю;

– збір головним персонажем їжі та таблеток;

– реалізація поведінки персонажів з використанням методів штучного інтелекту;

– визначення перемоги та поразки в грі.

Графічний інтерфейс:

– відображення ігрового поля, головного героя та супротивників;

– відображення поточного рахунку.

Інтерактивність:

– можливість керування головним персонажем за допомогою клавіатури.

Ще для проєкту потрібно описати моделі, які будемо реалізовувати у грі.

Діаграма класів є однією з основних моделей, вона візуалізує структуру системи, показуючи класи, їх атрибути, методи та зв'язки між ними. Діаграма класів допомагає розробникам зрозуміти взаємодію компонентів у системі – це збільшує ефективність та точність при розробці будь-якого програмного забезпечення [4].

Побудуємо діаграму класів, вона допоможе більш точно описати логіку у грі (рис. 2.1).

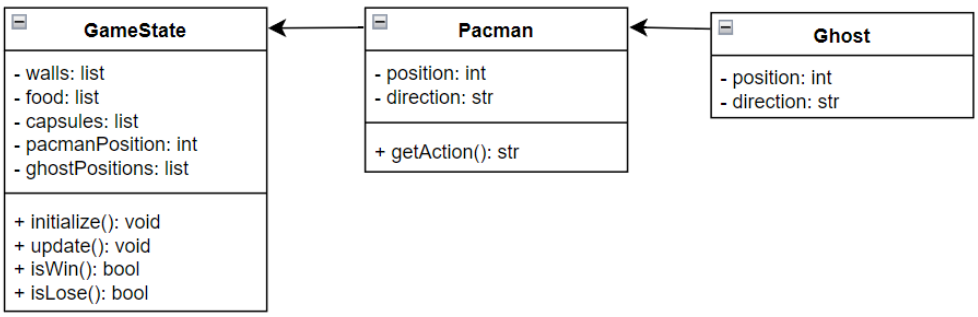

Рисунок 2.1 – Діаграма класів

Діаграма класів показує структуру та взаємозв'язок між основними класами в грі.

Клас «Pacman» представляє головного героя гри – пакмена. Він містить інформацію про позицію та напрямок руху пакмена.

Атрибути:

– position: int – поточна позиція пакмена на ігровому полі;

– direction: str – поточний напрямок руху пакмена.

Методи:

– getAction(): str – дії пакмена на основі вводу користувача або методу штучного інтелекту.

Клас «Ghost» представляє привидів, що переслідують головного персонажа. Він містить інформацію про позицію та напрямок руху кожного привида.

Атрибути:

– position: int – поточна позиція привидів на ігровому полі;

– direction: str – поточний напрямок руху привидів.

Методи для даного класу відсутні.

Клас «GameState» відповідає за збереження поточного стану гри. Він містить інформацію про розташування стін, їжі, таблеток, позиції головного героя та привидів.

Атрибути:

– wall: list – список з розташуванням стін на ігровому полі;

– food: list – список з розташуванням їжі на ігровому полі;

– capsules: list – список з розташуванням таблеток на ігровому полі;

– pacmanPosition: int – позиція пакмена на ігровому полі;

– ghostPosition: list – список з позиціями привидів на ігровому полі.

Методи:

– initialize(): void – ініціалізація елементів;

– update(): void – оновлення стану гри на основі проведених дій пакмена та привидів;

– isWin(): bool – перевірка чи виграв пакмен;

– isLose(): bool – перевірка чи програв пакмен.

Діаграма станів чітко ілюструє послідовність дій, які виконує пакмен під час гри. Кожен стан відповідає певному етапу гри, а зміни станів зумовлені взаємодією головного героя з іншими елементами ігрового середовища, такими як їжа, таблетки та привиди (рис. 2.2).

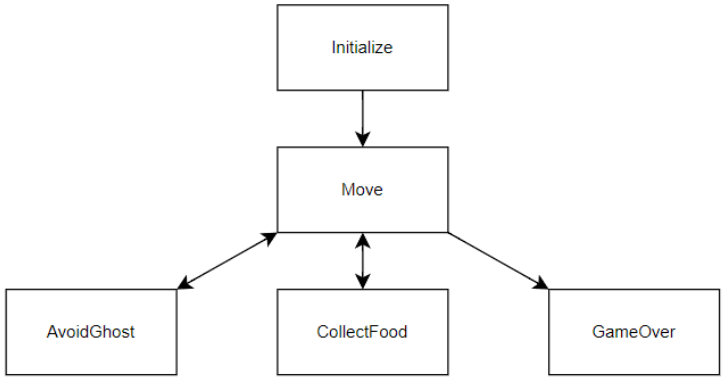

Рисунок 2.2 – Діаграма станів

«Initialize» – це початковий стан гри, всі елементи гри розташовуються на ігровому полі. Після завершення ініціалізації пакмен переходить до стану «Move».

«Move» – у цьому стані пакмен рухається по ігровому полю згідно з командами гравця або згідно з методом штучного інтелекту. Під час руху може перейти до станів «CollectFood», «AvoidGhost» або «GameOver», залежно від взаємодії з іншими елементами гри.

«CollectFood» – у цьому стані пакмен збирає їжу або таблетки, які знаходяться на його шляху, після збору повертається у стан «Move».

«AvoidGhost» – у цьому стані пакмен уникає привидів, що переслідують його. Якщо він потрапляє в зону до привида, він намагається змінити напрямок руху, щоб уникнути зіткнення. Якщо пакмен успішно уникає привидів, він повертається до стану «Move».

«GameOver» – це кінцевий стан гри, який настає, коли привид ловить пакмена або коли всі їжа на рівні зібрана. У цьому стані відображається результат гри. Вона завершується, далі немає переходів до інших станів.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для реалізації ігрового процесу потрібно обрати сучасні засоби, методи та алгоритми, які дозволяють забезпечити високоякісний ігровий процес, зручний

графічний інтерфейс та інтерактивність. Вибір відповідних інструментів залежить від багатьох факторів, таких, як продуктивність, простота використання та активна підтримка спільноти.

Таким чином, правильний вибір є ключовим фактором розробки якісних та інтерактивних ігор.

Для вибору засобів реалізації потрібно проаналізувати усі популярні варіанти:

1) Java – потужна та універсальна мова програмування. Вона використовується для розробки веб-додатків, корпоративних систем і мобільних додатків для Android. Java забезпечує високу продуктивність, автоматичне управління пам’яттю та велику стандартну бібліотеку, що робить її популярною серед розробників у всьому світі [5].

Переваги Java:

– Java-програми можна запускати на будь-якому пристрої;
– має вбудовані механізми безпеки, які забезпечують захист;
– автоматично звільняє невикористану пам’ять;
– широкий набір готових бібліотек для різних задач;
– легке створення багатозадачних програм.

Недоліки Java:

– програми можуть працювати повільніше, ніж на інших мовах;
– велике використання пам’яті пристрою;
– складна для початківців;
– реалізація графічного інтерфейсу користувача слабка та не зручна;
– обмежений доступ до апаратного забезпечення.

2) C++ – мова програмування, що поєднує високорівневе та низькорівневе програмування. Вона ефективно керує ресурсами та пам’яттю, що робить її ідеальною для розробки високопродуктивних додатків, таких як ігри та системне програмування [6].

Переваги C++:

– програми працюють швидко та ефективно;

– повне управління над пам’яттю пристрою;

– доступ до апаратних ресурсів та оптимізацій;

– підтримка об’єктно-орієнтованого програмування.

Недоліки C++:

– вивчення та використання потребує багато часу;

– великий ризик помилок, що пов’язані з управлінням пам’яттю;

– потрібно вручну керувати пам’яттю, щоб не допустити її витоків;

– часто виникають проблеми з перенесенням коду між різними компіляторами;

– через складність мови, збільшується час розробки.

3) Python – мова програмування, відома своїм синтаксисом. Вона широко використовується для веб-розробки, науки про дані, машинного навчання та автоматизації завдань завдяки великій стандартній бібліотеці та активній спільноті [7].

Переваги Python:

– має простий синтаксис, що робить його легким для використання;

– є багато вбудованих бібліотек;

– велика спільнота розробників, які допомагають один одному;

– можна запускати на різних системах без змін у коді;

– підтримка об’єктно орієнтованого програмування;

– простота коду сприяє швидкій розробці та доповненню програм.

Недоліки Python:

– трохи повільніша ніж інші мови програмування;

– використовує набагато більше пам’яті для великих програм;

– погана ефективність при розробці мобільних додатків;

– проблеми з багатозадачними програмами.

Також для аналізу проаналізуємо рейтинг популярності мов програмування (рис. 2.3).

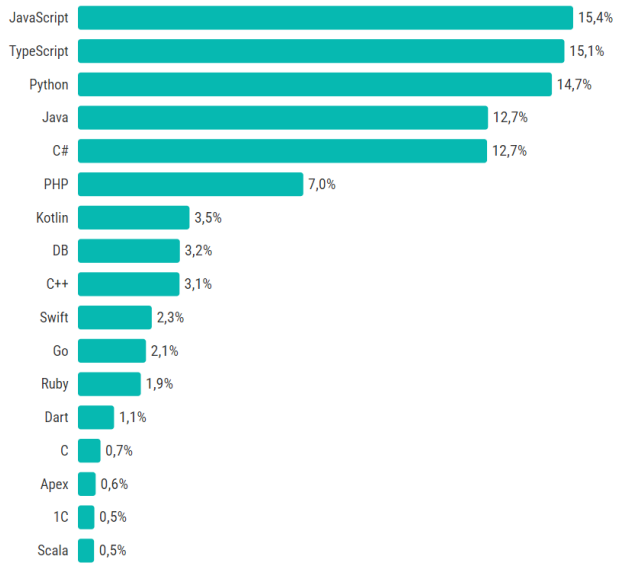

Рисунок 2.3 – Рейтинг мов програмування 2024 [8]

Розглянувши вище вказані мови програмування, їх переваги та недоліки, можна стверджувати, що кожна мова програмування особлива, в залежності від задач, які ми будемо виконувати. Для задовільненная поставленних умов розробки найкраще підійде мова програмування Python. Вона має є безліч потрібних для розробки бібліотек, підтримує об’єктно орієнтоване програмування та дуже легка до вивчення.

Висновки до розділу 2

Під час написання другого розділу було визначено та реалізовано низку вимог до гри Pacman, що включають у себе як функціональні, так і нефункціональні аспекти. Аналіз та визначення вимог дозволили чітко сформулювати цілі та завдання, які стоять перед системою.

Діаграми станів та класів допомогли візуалізувати структуру та логіку гри, що сприяло більш ефективному процесу розробки.

Було ретельно проаналізовано мови програмування, виділено їх переваги та недоліки і, на основі цього, обрано мову програмування для розробки.

РОЗДІЛ 3
РОЗРОБКА ГРИ PACMAN

3.1 Практична реалізація об’єкта проєктування

Першим кроком є створення основного модуля гри, він буде відповідати за цикл, обробку інших подій, приклад коду наведено у лістингу 3.1.

Лістинг 3.1 – Основний модуль гри

```
class GameState:
    def __init__(self, layout):
        # Ініціалізація об'єкту з початковими параметрами
        self.layout = layout  # Зберігає макет гри
        self.pacmanPos = layout.getPacmanStart()  # Встановлює початкову позицію пакмена
        self.ghostPos = layout.getGhostStarts()  # Встановлює початкові позиції привидів
        self.food = layout.getFood()  # Отримує розташування їжі на ігровому полі
        self.score = 0  # Встановлює початковий рахунок гри
    def isWin(self):
        # Перевіряє, чи гравець виграв (вся їжа зібрана)
        return self.food.count() == 0
    def isLose(self):
        # Перевіряє, чи гравець програв (пакмен зіткнувся з привидом)
        return self.pacmanPos in self.ghostPos
```

Кінець лістингу 3.1

Почнемо з розробки керування пакменом. Потрібно розробити спосіб, що дозволить власноруч керувати головним героєм, для цього створимо агента «KeyboardAgent», в лістингу 3.2 наведена його реалізація.

Лістинг 3.2 – KeyboarAgent

```
class KeyboardAgent(Agent):
  # Виконання дії, яку вибрав гравець
  def __init__(self, index=0):
    self.lastMove = Directions.STOP  # Зберігає останній хід гравця
    self.index = index  # Індекс агента
    self.keys = []  # Зберігає натиснуті клавіші

  def getAction(self, state):
```

```
    if keys != []:
      self.keys = keys  # Оновлення списку натиснених клавіш
      legal = state.getLegalMove(self.index)  # Отримання дозволених дій
для поточного стану
      move = self.getMove(legal)  # Отримання ходу на основі натиснених
клавіш
    if move == Directions.STOP:  # Якщо не обрано жодного ходу
    if self.lastMove in legalMove:  # Якщо останній хід є допустимим
        move = self.lastMove  # Виконати останній хід
    self.lastMove = move  # Оновлення останнього ходу
    return move  # Повернення обраного ходу

def getMove(self, legal):
  # Прив’язка дій до кнопок клавіатури

  move = Directions.STOP
  if ('Left' in self.keys) and Directions.WEST in legal:  move =
Directions.WEST  # Рух вліво
  if ('Right' in self.keys) and Directions.EAST in legal: move =
Directions.EAST  # Рух вправо
  if ('Up' in self.keys) and Directions.NORTH in legal:   move =
Directions.NORTH  # Рух вверх
  if ('Down' in self.keys) and Directions.SOUTH in legal: move =
Directions.SOUTH  # Рух вниз
  return move  # Повернення обраного ходу
```

Кінець лістингу 3.2

Далі потрібно створити модель поведінки привидів на ігровому полі, почнемо з найпростішого, реалізуємо привидів, що будуть просто робити випадкові, але дозволені рухи, реалізація подана у лістингу 3.3.

Лістинг 3.3 – Випадковий привид

```
class RandomGhost(GhostAgent):

    def getAction(self, state):
        legalMove = state.getLegalMove(self.index)  # Отримання
дозволених дій для поточного стану

        return random.choice(legalMove)  # Випадковий вибір з дозволених
дій
```

Кінець лістингу 3.3

Тепер створимо спрямованого привида, в лістингу 3.4 розробимо його модель поведінки, що буде приймати рішення, залежно від ситуації на полі.

Лістинг 3.4 – Спрямований привид

```
class DirectionalGhost(Agent):
    # Клас для спрямованого привида
    def __init__(self, index, prob_attack=0.8, prob_scaredFlee=0.8):
        self.index = index  # Індекс привида
        self.prob_attack = prob_attack  # Імовірність атаки на Pacman
        self.prob_scaredFlee = prob_scaredFlee  # Імовірність втечі, коли
привид наляканий
def getAction(self, state):
    # Вибір дії залежно від того, чи з'їв Pacman таблетку та стану
привида
    legal = state.getLegalMove(self.index)  # Отримання дозволених дій
pacmanPos = state.getPacmanPos()  # Отримання позиції пакмена
    ghostPos = state.getGhostPos(self.index)  # Отримання поточної
позиції привида
    Timer = state.getGhostTimer(self.index)  # Отримання часу наляканого
стану привида
    if Timer > 0:
        # Якщо привид наляканий, обирає дію для втечі
        return self.getScaredFleeMove(legal, pacmanPos, ghostPos)
    else:
        # Якщо привид не наляканий, обирає дію для атаки
        return self.getAttackMove(legal, pacmanPos, ghostPos)
```

Кінець лістингу 3.4

Далі у лістингу 3.5 опишемо цю модель поведінки, а саме, пропишимо переслідування та втечу, вони будуть працювати на основі пошуку найкращого маршруту, залежно від ситуації.

Лістинг 3.5 – Моделі поведінки

```
def getAttackAction(self, legal, pacmanPosition, ghostPosition):
    # Переслідування пакмена
    distances = [self.manhattanDistance(pacmanPos,
self.getSuccessorPosition(ghostPos, move)) for move in legalMoves]
    bestMove = legalMoves[distances.index(min(distances))]  # Вибір дії,
що мінімізує відстань до пакмена
    return bestMove
def getScaredFleeAction(self, legalMoves, pacmanPos, ghostPos):
    # Втеча від пакмена
    distances = [self.manhattanDistance(pacmanPos,
self.getNextPosition(ghostPos, action)) for move in legalMoves]
    bestMove = legalMoves[distances.index(max(distances))]  # Вибір дії,
що максимізує відстань від пакмена
    return bestMove
```

```
def manhattanDistance(self, xy1, xy2):
    # Обчислення відстані
    return abs(xy1[0] - xy2[0]) + abs(xy1[1] - xy2[1])
```

Кінець лістингу 3.5

Тепер візуалізуємо дану модель поведінки (рис. 3.1).

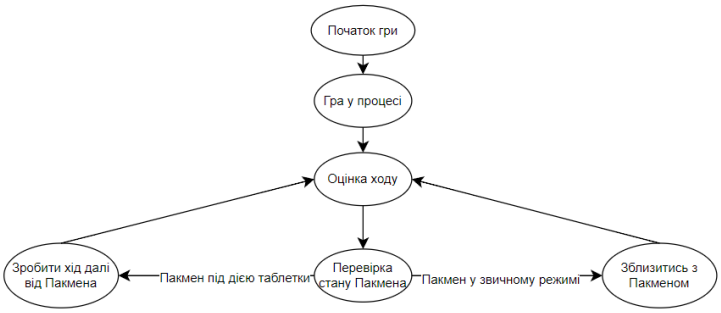

Рисунок 3.1 – Модель поведінки

Після цього розробимо метод для прийняття рішень ігровими персонажам, розглянемо «Minimax» [9] та «Alpha-Beta» [10].

«Minimax» – це метод, що знаходить хороші ходи та аналізує моменти того, що суперник буде їм протидіяти. Основа «Minimax» – це максимізація можливого виграшу та мінімізація усіх можливих втрат які можуть бути нанесені суперником.

Розглянемо роботу цього методу. Головний герой – це пакмен, тому для нього викликається функція «Max()», вона аналізує усі доступні ходи для головного героя. Після цього викликається функція «Min()». Функція «Min()» оцінює позицію та повертає можливі ходи для кожного привида. Так як ситуація оцінюється для пакмена, нам повертається рух з найкращою позицією.

Функція «Min()» працює в зворотньому порядку відносно функції «Max()».

Ці функції викликають одна одну, поки не досягнуть потрібної глибини пошуку. Вони перевіряють усі можливі ситуації поведінки, після цього відправляють оцінку вгору, а програма на їх основі аналізує та робить найсильніші ходи за кожну зі сторін.

Основним мінусом методу «Minimax» є те, що він обробляє усі можливі дії. Якщо він зустрічає карту великого розміру та декілька привидів, то сильно збільшує час аналізу, тому даний метод є неефективним.

Метод «Alpha-Beta» майже не відрізняється від «Minimax», його можна назвати доповненням. «Alpha-Beta» вирішує проблеми свого аналога тим, що відсікає погані ситуації, які не ведуть до поліпшення результату.

Метод передає дві оцінки для огляду. Перша оцінка – альфа, вона показує найкращий результат, що уже знайшли. Усі ходи, у яких оцінка ситуації гірша за альфу, є хибними. Друга оцінка – бета, вона відповідає за найгірший для нас результат, він буде вибраний як хід суперника.

Таким чином, «Alpha-Beta» може сильно зменшити обсяг інформації для обробки. Однак метод може потрапити на гірші ходи з початку, в такому випадку час може залишитись таким самим, як у методі «Minimax». Проте при знаходженні хороших ходів спочатку, він зможе проаналізувати ситуацію глибше, на більшу кількість ходів.

В додатку А наведено програмну реалізацію «Alpha-Beta». Робота даного методу (рис. 3.2).

Рисунок 3.2 – Метод «Alpha-Beta»

Графіка є важливою частиною більшості комп'ютерних ігор, вона показує нам те, як гравці взаємодіюсь з ігровим світом. Будь-який вид візуалізації робить гру привабливою та інтуїтивно зрозумілою.

Графіка дозволяє краще орієнтуватись в ігровому середовищі, а саме бачити персонажів, їх цілі, супротивників та різноманітні бонуси. Хороша графіка може покращити сприйняття гри. Вона має свою основну задачу, а саме передавання великої кількості інформації для гравця.

Також графіка полегшує роботу при тестуванні гри, оскільки розробник може краще відслідковувати непередбачувані ситуації.

Наступним кроком є створення графічного вигляду гри, нам потрібно написати логіку створення візуальної частини.

Основою поля гри є стіни, вони чітко показують ігрову область, спосіб їх відображення вказаний нижче у лістингу 3.6.

Лістинг 3.6 – Відображення стін

```
def drawWalls(self, wallMatrix):
    for xNum, x in enumerate(wallMatrix):
        for yNum, cell in enumerate(x):
            if cell:  # Якщо є стіна
                pos = (xNum, yNum)  # Позиція стіни на матриці
                screen = self.to_screen(pos)  # Перетворення позиції на
координати екрану
```

Кінець лістингу 3.6

Головною діючою особою гри є пакмен, в лістингу 3.7 наведений код для його відображення.

Лістинг 3.7 – Відображення пакмена

```
def drawPacman(self, pacman, idx):
    # Отримання позиції пакмена
    pacmanPos = self.getPosition(pacman)
    # Перетворення позиції на координати екрану
    screenCoords = self.to_screen(pacmanPos)
    # Отримання кінцевих точок для пакмена
    adcEndpoints = self.getEndpoints(self.getDirection(pacman))
    # Малювання пакмена з заданими параметрами
```

```
    return [circle(screen_point, PACMAN_SCALE * self.gridSize,
fillColor=fillColor, endpoints=endpoints)]
```

Кінець лістингу 3.7

Не менш важливими є суперники пакмена, а саме привиди, для їх відображення скористаємося кодом, що наведений у лістингу 3.8.

Лістинг 3.8 – Відображення привидів

```
def drawGhost(self, ghost, agentIndex):
    # Отримання позиції привида
    ghostPos = self.getPosition(ghost)
    # Перетворення позиції на координати екрану
    screenCoords = self.to_screen(ghostPos)
    # Масштабування та зміщення координат для малювання привида
    vertexCoords = [(x * self.gridSize * GHOST_SIZE + screen_Coords [0],
y * self.gridSize * GHOST_SIZE + screen_pos[1]) for x, y in GHOST_SHAPE]

    # Отримання кольору привида
    ghostColor = self.getGhostColor(ghost, agentIndex)
    # Малювання тіла привида з заданими параметрами
    ghost Body = polygon(vertexCoords, ghostColor, filled=True)
```

Кінець лістингу 3.8

Візуальний вигляд елементів ігрового поля вказаний на рисунку 3.3.

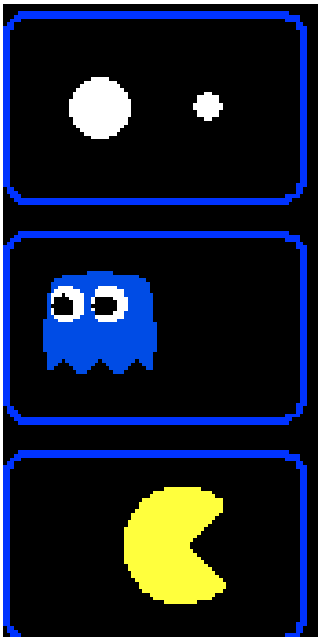

Рисунок 3.3 – Елементи ігрового поля

Карти для гри мають важливу роль у різноманітності ігрового процесу. Зміна розташування стін, їжі, таблеток та привидів змушують гравця змінювати свою стратегію для досягення успіху. Він повинен підлаштуватись під новий сценарій.

Створимо модуль, який буде приймати макети та графічно їх відображати в ігровій області, в додатку Б наведено код програми.

Після цього створюємо макети, дивимось отриманий результат. Код для першої карти наведений у лістингу 3.9.

Лістинг 3.9 – Перша карта

```
%%%%%
%G. .%
%. ..%
% . .%
%. ..%
%   .%
%  ..%
%   .%
%P  .%
%%%%%
```

Кінець лістингу 3.9

Результат роботи коду показаний на рисунку 3.4.

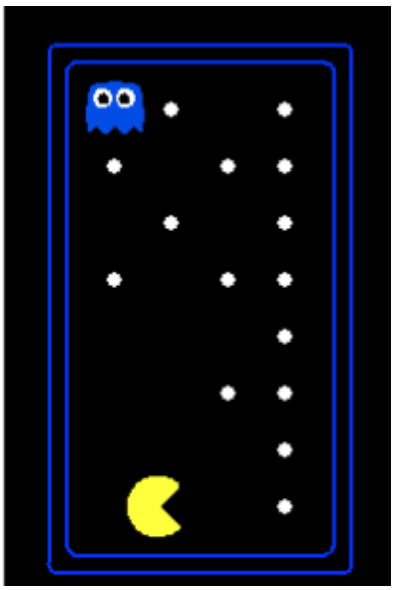

Рисунок 3.4 – Перша карта

Далі створимо більшу карту, де додамо 2 привида та 2 таблетки, код наведений у лістингу 3.10.

Лістинг 3.10 – Друга карта

```
%%%%%%%%%%%%%%%%%%%%
%o...%........%....%
%.%%.%.%%%%%.%.%%.%
%.%..............%.%
%.%.%%.%%  %%.%%.%.%
%......%G  G%......%
%.%.%%.%%%%%.%%.%.%
%.%..............%.%
%.%%.%.%%%%%.%.%%.%
%....%...P....%...o%
%%%%%%%%%%%%%%%%%%%%
```

Кінець лістингу 3.10

Результат роботи коду показаний на рисунку 3.5.

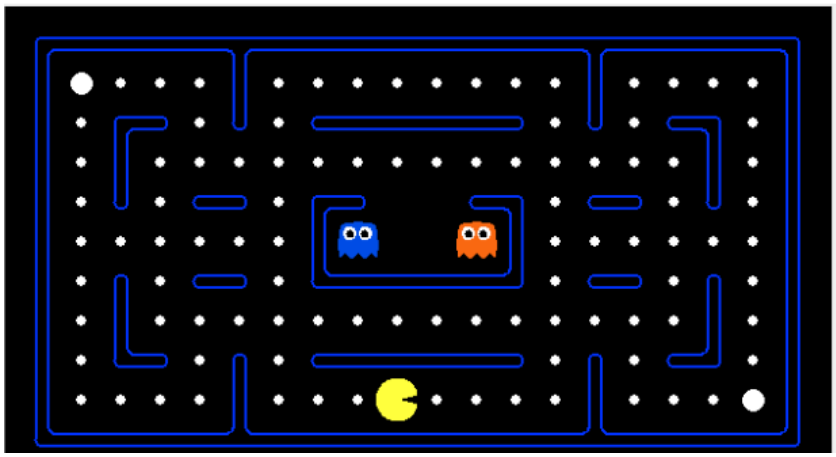

Рисунок 3.5 – Друга карта

Система очок є дуже важливою частиною гри, вона спонукає гравця працювати на результат, це додає додаткової мотивації. Також це сприяє реіграбельності, гравець буде старатись покращувати свої попередні рекорди. За допомогою системи очок легше налаштувати штучний інтелект, він буде обирати результати, які призведуть до найкращого результату.

Система очок показує прогрес гравця в реальному часі. Коли пакмен з’їдає їжу, гравець одразу бачить приріст очок.

Також буде доцільно зробити систему штрафів, вона буде знижувати результат за витрачений час, це буде спонукати гравця мислити швидше, обирати найкращий маршрут та уникати можливих повторів у виборі шляхів. Дані рішення додають елементу змагання, можна обмінюватися своїми результатами з друзями, це додасть стимулу для покращення своїх результатів.

Також система очок спонукає гравця до стратегічних планувань кожної дії. Наприклад, гравець може ризикнути і піти в куток, щоб зібрати усю їжу у ньому зараз та не отримати більше штрафів.

Для почату створимо нагороди для пакмена, 10 очок за кожну одиницю їжі, код наведений у лістингу 3.11.

Лістинг 3.11 – Нагорода за їжу

```
def consume(position, state):
    posX, posY = position
    if gameState.data.food[posX][posY]:  # Перевірка, чи є їжа в цій позиції
        gameState.data.scoreChange += 10  # Збільшення рахунку
        gameState.data.food = gameState.data.food.copy()  # Створення копії масиву їжі
        gameState.data.food[posX][posY] = False  # Видалення їжі з поточної позиції
        gameState.data._foodEaten = position  # Збереження позиції з'їденої їжі
```

Кінець лістингу 3.11

Далі врахуємо очки за зіткнення з привидом, код у лістингу 3.12.

Лістинг 3.12 – Очки за привидів

```
def collide(gameState, ghostState, agentIdx):
    # Перевірка, чи зляканий привид
    if ghostState.scaredTimer > 0:
      gameState.data.scoreChange += 200  # Додати 200 очок
      GhostRules.placeGhost(gameState, ghostState)  # Повернути привида на початкову позицію
      ghostState.scaredTimer = 0  # Скидання таймера зляканого стану
      gameState.data._eaten[agentIdx] = True  # Відмітка привида як з'їденого
    else:
      # Якщо пакмен програє, зменшити рахунок
```

```
        if not gameState.data._win:
          gameState.data.scoreChange -= 500  # Зменшити рахунок на 500 очок
          gameState.data._lose = True  # Встановити стан програшу

collide = staticmethod(collide)
```

Кінець лістингу 3.12

Ще потрібно додати штраф за час проходження карти, приклад коду наведений у лістингу 3.13.

Лістинг 3.13 – Штраф за рух

```
if agentIndex == 0: # перевірка чи це пакмен
  state.data.score += -PENALTY # штраф за час
else:
  GhostRules.decrementTimer()
```

Кінець лістингу 3.13

Останнім додамо виведення результатів після закінчення гри, код наведений у лістингу 3.14.

Лістинг 3.14 – Виведення результатів

```
print(('Середній рахунок:', sum(scores) / float(len(scores))))
print(('Рахунок:       ', ', '.join([str(score) for score in
scores])))
print(('Відсоток перемог:      %d/%d (%.2f)' % (wins.count(True),
len(wins),
winRate)))
print(('Рекорд:       ', ', '.join([ ['Програв', 'Виграв'][int(w)] for w
in
wins])))
```

Кінець лістингу 3.14

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Перейдемо до тестування. Запустимо гру та перевіримо керуванання пакменом за допомогою клавіатури (рис. 3.6).

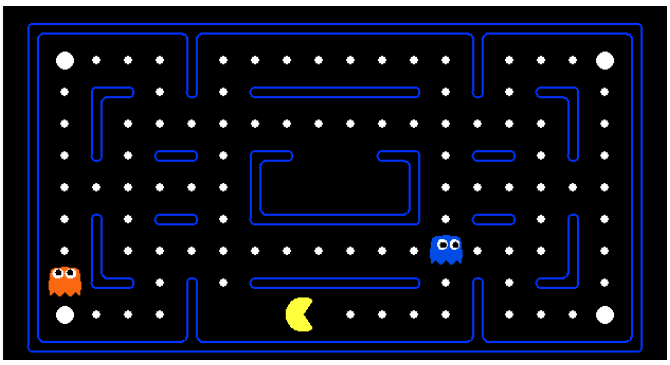

Рисунок 3.6 – Керування за допомогою клавіатури

Після цього потрібно перевірити чи правильно працює рахунок, за їжу +10 очок, за привида під дією таблетки +200 очок, при звичаних ходах -1 очко (рис. 3.7).

Рисунок 3.7 – Правильність роботи очків

А також при програші гри, що відбувається при зіткненні з привидом нараховуються -500 очок (рис. 3.8).

Рисунок 3.8 – Штраф за програш

Перевіримо виведення результату при виграші (рис. 3.9).

```
Пакмен переміг! Рахунок: 1101
('Середній рахунок:', 1101.0)
('Рахунок:       ', '1101')
Відсоток перемог:      1/1 (1.00)
('Рекорд:        ', 'Виграв')
Витрачений час: 20.78 секунд
```

Рисунок 3.9 – Виграш

А також при програші (рис. 3.10).

```
Пакмен помер! Рахунок: -501
('Середній рахунок:', -501.0)
('Рахунок:       ', '-501')
Відсоток перемог:      0/1 (0.00)
('Рекорд:        ', 'Програв')
Витрачений час: 1.43 секунд
```

Рисунок 3.10 – Програш

Переверка роботи гри успішно завершена, всі її елементи працюють стабільно і швидко.

3.3 Тестування роботи штучного інтелекту

Тестування штучного інтелекту є дуже важливим етапом у розробці гри. Основна мета тестування полягає у перевірці його коректного функціонування та ефективності у різних ситуаціях.

Головним аспектом тестування штучного інтелекту для гри є оцінка швидкості прийняття рішень та максимізація кінцевого рахунку.

Перший тест проведемо на маленькій карті з двома привидами, для тесту зіграємо 25 ігор, вони будуть проведені спочатку з випадковими привидами, потім зі спрямованими, карта тесту показана на рисунку 3.11.

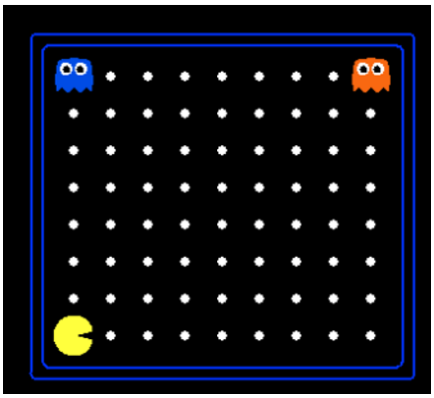

Рисунок 3.11 – Карта для першого тесту

Проведемо тести і запишемо основні параметри, що чітко опишуть швідкість та точність роботи агента на базі штучного інтелекту. Результати проведеного тесту наведені у таблиці 3.1.

Таблиця 3.1 – Тест першої карти

Тип привидів / Атрибут	Випадкові	Спрямовані
Кількість перемог	25	15
Максимальна кількість очок	1086	1104
Середня кількість очок	1049.64	608.96
Затрачений на тест час	43.57	26.83

На основі цього тесту можна бачити, що з випадковими привидами, пакмен зміг перемогти усі ігри, а зі спрямованими лише 15.

За рахунок випадкових привидів пакмен не міг зібрати усі таблетки безпечно, тому максимальна кількість очок поступається спрямованим.

За рахунок великої кількості програшів, час розрахунку у випадкових привидів більший. Середній результат очок при випадкових привидах більший на 72 %.

Наступний тест проведемо на карті середнього розміру (рис. 3.12).

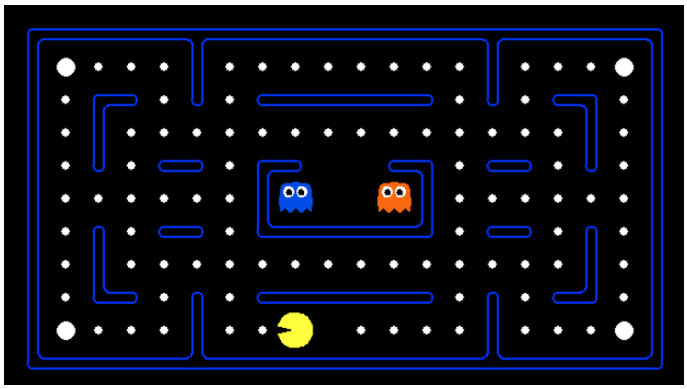

Рисунок 3.12 – Карта для другого тесту

Результат тестування наведений у таблиці 3.2.

Таблиця 3.2 – Тест другої карти

Тип привидів / Атрибут	Випадкові	Спрямовані
Кількість перемог	20	13
Максимальна кількість очок	2353	2455
Середня кількість очок	1631.4	1546.68
Затрачений на тест час	26.80	11.30

На середній карті результат перемог погіршився для обох варіантів. Максимальна кількість очок так само залишилась більшою у спрямованих привидів.

Середня кількість очок залишилась меншою, але різниця стала набагато меншою, всього 5 %.

Затрачений на тест час теж залишився суттєво меншим у спрямованих привидів.

Останній тест проведемо на великій карті – це покаже роботу сетоду у складній ситуації (рис. 3.13).

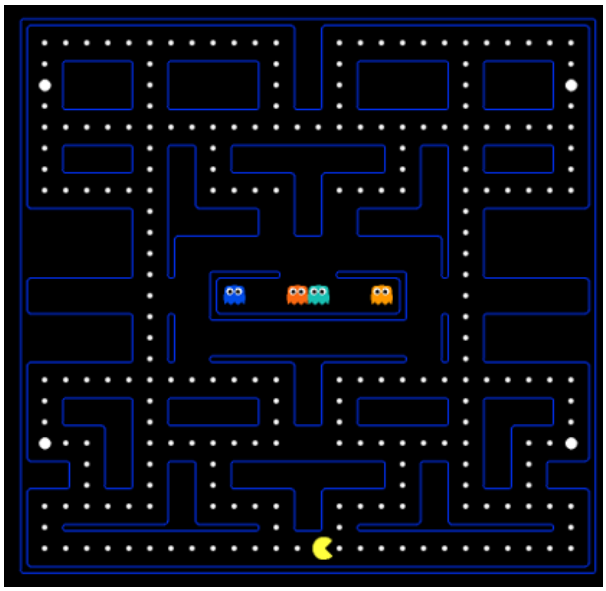

Рисунок 3.13 – Карта для третього тесту

Результат тестування наведений у таблиці 3.3.

Таблиця 3.3 – Тест третьої карти

Тип привидів / Атрибут	Випадкові	Спрямовані
Кількість перемог	10	1
Максимальна кількість очок	3140	3304
Середня кількість очок	1778	1375
Затрачений на тест час	147.57	30.90

На карті з 4 привидами результати майже не змінилися. Слід зазначити, що обіграти спрямованих привидів пакмен зміг лише один раз, проте часу затраченого на тест випадкових привидів витрачено в 5 раз більше.

Підведемо підсумки тестування, випадкові привиди дозволяють досягти більшої кількості перемог, проте їх аналіз потребує більше часу. Водночас, спрямовані привиди створять більш складні умови, що ускладнює можливість перемоги, але у разі успіху отримуємо більшу кількість очок.

3.4 Захист інформаційно-комп’ютерної системи

Захист є дуже важливою частиною розробки будь-якого програмного забезпечення, особливо для комп’ютерних ігор. Впровадження такої системи потребує особливої уваги, при поганому рівні захисту можна зазнати не тільки репутаційних, але й фінансових збитків. Якщо зловмисник отримає доступ до гри, то він зможе повністю її знищити. Основними чинниками захисту інформації є забезпечення стабільної роботи та цілісності системи.

Для забезпеченя правильної роботи гри, було проведено обробку винятків та різних ситуацій – це допомогло уникнути критичних збоїв програми та забезпечити стабільну роботу.

Хоч гра є локальним проєктом, було дотримано методів програмування та безпеки коду, щоб зменшити можливість появи можливих вразливостей. Це забезпечує не тільки безпеку, але й довготривалу надійність гри. Ще було проведено безліч стрес-тестів з різними сценаріями, для виявлення невеликих

помилок, це допоможе грі підтримувати стабільну роботу при великій кількості параметрів.

Цілісність системи є аспектом безпеки, якому потрібно приділити багато уваги. При передачі даних через будь-які сервіси краще хешувати його. Зміна даних буде приводити до зміни хешу і це буде просто помітити.

Також дуже важливо робити резервне копіювання даних, воно дозволить відновити попередню версію проєкту, незалежно від типу і складності помилки. Також це допоможе при шкідливих програмах або хакерах які пошкодили або змінили дані вашої розробки.

3.5 Використання методу «Alpha-Beta» у робототехніці

Розроблений метод «Alpha-Beta» можна використовувати не тільки у ігровому середовищі, але й у сферах іншої діяльності.

Хорошим прикладом, буде робототехніка. Наприклад, робот-полосос, що намагається прибрати усі куточки кімнати. Стіни кімнати стають зоною діяльності, яка утримує робота, а меблі стають аналогом перешкод на шляху. Метод, розроблений для гри, може керувати рухом робота, забезпечуючи ефективне прибирання кімнати та уникнення зіткнень з об’єктами.

Основні принципи роботи включають:

– виявлення стін та меблів та їх уникнення;

– знаходження короткого і ефективного маршруту для прибирання всього приміщення;

– динамічне реагування на зміни в оточенні, наприклад, якщо хтось перемістить меблі.

Розглянемо кілька конкретних прикладів використання методу:

– робот-пилосос – як вже згадувалося, робот може прибирати приміщення, уникаючи зіткнень зі стінами та меблями;

– робот для складу – робот може використовувати цей метод для навігації між стелажами для переміщення товарів;

– робот для порятунку людей – у ситуаціях, коли робот використовується для пошуку людей після катастроф, цей метод допоможе ефективно обстежувати місцевість.

Таким чином, розроблений метод може бути використаний у робототехніці, забезпечуючи виконання завдань у різних умовах.

Висновки до розділу 3

У третьому розділі було розроблено основний функціонал гри, а саме: створення карти та основних персонажів, змога керування пакменом з клавіатури, розроблення методу поведінки персонажів та двох моделей для керування привидами – випадкові та направлені. Також проведено тести, що показали переваги та недоліки розроблених методів. Під час розробки було дотримано усіх вимог безпеки. Проведено аналіз використання розробленого методу у робототехніці.

ВИСНОВКИ

У ході розробки гри «Pacman» було здійснено комплексний підхід до реалізації та впровадження різних аспектів розробки програмного забезпечення.

Проведений аналіз поточного стану проблем використання штучного інтелекту в індустрії відеоігор та визначення актуальності, підтвердили необхідність створення інтелектуального агента для класичної гри з використанням вдосконалених технологій та методів.

Здійснено вибір засобів та методів розробки, а саме, поставлено усі цілі та вимоги до гри, діаграми станів та класів сприяли більш ефективному та швидкому процесу розробки, а аналіз мов програмування допоміг обрати оптимальний варіант для розробки проєкту.

Було спроєктовано структуру гри, яка включає основні компоненти та взаємодії між ними. Розробка графічного інтерфейсу забезпечила інтуїтивно зрозумілий і привабливий візуальний вигляд гри.

Особливу увагу було приділено реалізації поведінки персонажів під управлінням штучного інтелекту, що забезпечило створення нових викликів та непередбачуваності під час гри.

Аналіз ефективності різних методів поведінки, дозволив вибрати найбільш оптимальні рішення для створення цікавого ігрового процесу.

Забезпечення безпеки даних було досягнуто шляхом впровадження засобів та методів програмування, що гарантують стабільну та надійну роботу системи.

Проведено аналіз методу для використання у сфері робототехніки, наведено принципи його роботи. Це дозволить використовувати алгоритм не тільки у відеоіграх.

Таким чином, проєкт був успішно розроблений, він досягнув усіх поставлених цілей. Реалізована гра забезпечує гравця незабутніми враженнями від гри, що відповідає їх сучасним очікуванням, а інтелектуальний агент може бути використаний і у інших розробках.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Історія розвитку ігор: від Pacman до WOW. URL: http://surl.li/umlak (дата звернення: 05.03.2024).

2. Stockfish – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Stockfish (дата звернення: 07.03.2024).

3. Ігровий штучний інтелект – Вікіпедія. URL: http://surl.li/ulbua (дата звернення: 12.03.2024).

4. Підручник з діаграми класів UML: абстрактний клас із прикладами. URL: http://surl.li/umlas (дата зверенення: 25.03.2024).

5. Розробка на Java – з чого почати знайомство з мовою | DOU. URL: https://dou.ua/lenta/articles/how-to-learn-java (дата звернення: 15.03.2024).

6. Низькорівневе програмування: що це і де застосовується. URL: https://foxminded.ua/nyzkorivneve-prohramuvannia (дата звернення: 18.03.2024).

7. Що таке мова програмування Python та для чого вона використовується. URL: http://surl.li/umlag (дата звернення: 19.03.2024).

8. Рейтинг мов програмування-2024: з якими мовами працюють та що планують вчити айтівці різних напрямків. URL: http://surl.li/umkzz (дата зверенення: 24.03.2024).

9. Minimax Brilliant Math & Science Wiki. URL: https://brilliant.org/wiki/minimax (дата звернення: 16.04.2024).

10. Alpha Beta pruning – Scaler Topics. URL: http://surl.li/umkzj (дата звернення: 17.04.2024).

ДОДАТКИ

Додаток А

Лістинг методу «Alpha-Beta»

```
class AlphaBetaAgent(Agent):
    #Базовий клас для агентів
    def __init__(self, index=0, depth=3):
        self.index = index
        self.depth = depth
       # Виконання альфа-бета пошуку для вибору найкращої дії
    def getAction(self, state):
        action, _ = self.alpha_beta(state, self.depth, self.index,
float('-inf'), float('inf'))
        return action

       # Основна функція альфа-бета пошуку
    def alpha_beta(self, state, depth, agent_index, alpha, beta):
        if depth == 0 or state.isWin() or state.isLose():
            return None, self.evaluationFunction(state)
        # Для пакмена
        if agent_index == 0:
            return self.maximize(state, depth, alpha, beta)
        # Для привидів
        else:
            return self.minimize(state, depth, agent_index, alpha, beta)

       # Функція максимізації для пакмена
    def maximize(self, state, depth, alpha, beta):
        depth = int(depth)
        actions = state.getLegalActions(0)
        if not actions:
            return None, self.evaluationFunction(state)
        max_value = float('-inf')
        max_action = None
        for action in actions:
            next_state = state.generateSuccessor(0, action)
            _, value = self.alpha_beta(next_state, depth, 1, alpha, beta)
            if value is not None and value > max_value:
                max_value = value
                max_action = action
            if max_value > beta:
                return max_action, max_value
            alpha = max(alpha, max_value)
        return max_action, max_value

       # Функція мінімізації для привидів
    def minimize(self, state, depth, agent_index, alpha, beta):
        depth = int(depth)
        actions = state.getLegalActions(agent_index)
        if not actions:
```

```
            return None, self.evaluationFunction(state)
        next_agent = (agent_index + 1) % state.getNumAgents()
        if next_agent == 0:
            depth -= 1
        min_value = float('inf')
        min_action = None
        for action in actions:
            next_state = state.generateSuccessor(agent_index, action)
            _, value = self.alpha_beta(next_state, depth, next_agent,
alpha, beta)
            if value is not None and value < min_value:
                min_value = value
                min_action = action
            if min_value < alpha:
                return min_action, min_value
            beta = min(beta, min_value)
        return min_action, min_value

        # Функція оцінки стану гри для прийняття рішень
    def evaluationFunction(self, currentGameState):
        newPos = currentGameState.getPacmanPosition()
        newFood = currentGameState.getFood()
        newGhostStates = currentGameState.getGhostStates()
        newScaredTimes = [ghostState.scaredTimer for ghostState in
newGhostStates]
        foodDistances = [util.manhattanDistance(newPos, food) for food in
newFood.asList()]
        closestFoodDistance = min(foodDistances) if foodDistances else 0
        ghostDistances = [util.manhattanDistance(newPos,
ghost.getPosition()) for ghost in newGhostStates]
        capsulePositions = currentGameState.getCapsules()
        capsuleDistances = [util.manhattanDistance(newPos, capsule) for
capsule in capsulePositions]
        closestCapsuleDistance = min(capsuleDistances) if
capsuleDistances else 0
        remainingFood = currentGameState.getNumFood()
        # Оцінка для різних факторів
        foodWeight = -1
        ghostWeight = 2
        scaredGhostWeight = -2
        capsuleWeight = -3
        remainingFoodWeight = -10
        # Обчислення рахунку гри
        score = currentGameState.getScore() \
                + foodWeight * closestFoodDistance \
                + ghostWeight * min(ghostDistances) \
                + scaredGhostWeight * min(newScaredTimes) \
                + capsuleWeight * closestCapsuleDistance \
                + remainingFoodWeight * remainingFood

        return score
```

Додаток Б

Лістинг модуля для прийняття моделей карт

```
class Layout:
    def __init__(self, layoutText):
      #Ініціалізує макет ігрового поля на основі текстового представлення
      self.width = len(layoutText[0])
      self.height = len(layoutText)
      self.walls = [[False for _ in range(self.height)] for _ in
range(self.width)]
      self.food = [[False for _ in range(self.height)] for _ in
range(self.width)]
      self.capsules = []
      self.agentPositions = []
      self.processLayoutText(layoutText)

    def processLayoutText(self, layoutText):
      #Обробляє текстовий макет і створює відповідні масиви для стін, їжі
та агентів
      for y in range(self.height):
        for x in range(self.width):
          layoutChar = layoutText[self.height - y - 1][x]
          self.processLayoutChar(x, y, layoutChar)

    def processLayoutChar(self, x, y, layoutChar):
      #Обробляє окремий символ макету і створює відповідні елементи
      if layoutChar == '%':
        self.walls[x][y] = True
      elif layoutChar == '.':
        self.food[x][y] = True
      elif layoutChar == 'o':
        self.capsules.append((x, y))
      elif layoutChar == 'P':
        self.agentPositions.append((x, y, 0))
      elif layoutChar in '1234':
        self.agentPositions.append((x, y, int(layoutChar)))

    def getPacmanStart(self):
      #Повертає початкову позицію пакмена
      for x, y, agentType in self.agentPositions:
        if agentType == 0:
          return (x, y)


    def getLayout(name, back=2):
      #Завантажує макет з файлу
      if name.endswith('.lay'):
        layout = readLayoutText(name)
      else:
        layout = readLayoutText(name + '.lay')
```

```
        return Layout(layout)


def readLayoutText(filename):
  #Зчитує текстовий макет з файлу
  with open(filename) as f:
    return [line.strip() for line in f]
```