МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н. Каразіна

Факультет: **ННІ Каразінський банківський інститут**
Кафедра: **Інформаційних технологій та математичного моделювання**
Спеціальність: **122 Комп'ютерні науки**
Освітня програма: **Комп'ютерні науки та інформаційні технології в бізнесі**

Група: АК-41б **денна форма навчання**

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему:

«РОЗРОБКА СТРАТЕГІЧНОЇ ГРИ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ НА PYTHON»

ЗА НАКАЗОМ № 4601-5/335 ВІД 07 ЛЮТОГО 2025 РОКУ

здобувача вищої освіти **Паламарчук Аріни Михайлівни**

Робота допущена до захисту в ЕК
протокол кафедри ІТММ № 13 від 31.05.2025р.

Завідувач кафедри ІТММ
к.п.н.
_____ Н.І. Стяглик

Науковий керівник
Ph.D з «Комп'ютерних наук»
_____ Д.М.Ковальчук

м. Харків 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

Факультет навчально-науковий інститут "Каразінський банківський інститут"
Кафедра інформаційних технологій та математичного моделювання
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки та інформаційні технології в бізнесі

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ **Н. І. Стяглик**
Підпис ініціали, прізвище

"08" лютого 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Паламарчук Аріна Михайлівна
(прізвище, ім'я, по батькові студента)

1. Тема роботи: «Розробка стратегічної гри з елементами штучного інтелекту на Python»

керівник роботи: Ph.D з «Комп'ютерних наук» Ковальчук Д.М.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "08" лютого 2025 року № 4601-5/335

2. Строк подання студентом роботи 15 травня 2025 року

3. Перелік питань, які потрібно розробити:
У розділі 1: проаналізувати теоретичні засади, етапи еволюції, специфіку проєктування стратегічних ігор, а також визначити роль штучного інтелекту в їх реалізації

У розділі 2: дослідити сучасні алгоритмічні підходи до моделювання поведінки віртуальних агентів у стратегічних іграх, зокрема алгоритми A*, Minimax, FSM, тощо

У розділі 3: розглянути можливості мови програмування Python для розробки ігрових застосунків зі структурованою архітектурою, елементами геймплею, графіки та штучного інтелекту

У розділі 4: оцінити перспективи розвитку стратегічних ігор із ШІ, реалізованих засобами Python, з урахуванням актуальних трендів, ефективності технологій та напрямів їх подальшого вдосконалення.

4. План роботи

№ з/п	Назви етапів роботи
1	Вибір здобувачем теми кваліфікаційної бакалаврської роботи
2	Затвердження плану і завдання кваліфікаційної бакалаврської роботи
3	Здача кваліфікаційної бакалаврської роботи керівнику
4	Підпис кваліфікаційної бакалаврської роботи керівника
5	Підпис кваліфікаційної бакалаврської роботи у нормоконтролера
6	Допуск завідувачем кафедри до захисту кваліфікаційної бакалаврської роботи
7	Захист кваліфікаційної бакалаврської роботи

5. Дата видачі завдання 08 лютого 2025 року

Студент _____ А.М Паламарчук
підпис ініціали, прізвище

Керівник роботи _____ Д.М. Ковальчук
підпис ініціали, прізвище

РЕФЕРАТ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ
« РОЗРОБКА СТРАТЕГІЧНОЇ ГРИ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ НА PYTHON »
Паламарчук Аріна Михайлівна

Кваліфікаційна бакалаврська робота містить 58 сторінок, 2 таблиці, 18 рисунків, список літератури з 19 найменувань.

Об'єктом дослідження є процес створення стратегічних комп'ютерних ігор із використанням технологій штучного інтелекту.

Предмет дослідження є методи та засоби реалізації штучного інтелекту в стратегічних іграх з використанням мови програмування Python.

Метою кваліфікаційної бакалаврської роботи є розробка прототипу стратегічної комп'ютерної гри із вбудованими елементами штучного інтелекту на основі мови програмування Python, що забезпечує моделювання динамічної взаємодії між гравцем і віртуальним супротивником, а також аналіз ефективності відповідних алгоритмів і технологічних рішень.

Завданнями кваліфікаційної бакалаврської роботи є:

- проаналізувати теоретичні засади, етапи еволюції, специфіку проєктування стратегічних ігор, а також визначити роль штучного інтелекту в їх реалізації;
- дослідити сучасні алгоритмічні підходи до моделювання поведінки віртуальних агентів у стратегічних іграх, зокрема алгоритми A*, Minimax, FSM тощо;
- розглянути можливості мови програмування Python для розробки ігрових застосунків зі структурованою архітектурою, елементами геймплею, графіки та штучного інтелекту;
- оцінити перспективи розвитку стратегічних ігор із ШІ, реалізованих засобами Python, з урахуванням актуальних трендів, ефективності технологій та напрямів їх подальшого вдосконалення.

Актуальність дослідження. Індустрія комп'ютерних ігор демонструє стабільну тенденцію до зростання, при цьому важливу роль відіграє впровадження адаптивних систем, заснованих на штучному інтелекті. Використання Python як мови програмування з широким спектром інструментів для реалізації інтелектуальних агентів відкриває нові можливості для створення інноваційних ігрових рішень. У цьому контексті дослідження, присвячене створенню стратегічної гри з ШІ, є своєчасним і практично значущим.

За результатами дослідження: створено функціональний прототип стратегічної гри з базовими елементами інтелектуальної поведінки супротивника.

Практична новизна роботи полягає у розробленні підходу до створення стратегічної гри з використанням модульної архітектури та типових алгоритмів штучного інтелекту (на прикладі Python), що може бути

застосований як у навчальному процесі, так і для демонстраційних або експериментальних цілей.

Одержані результати можуть бути використані як навчально-методичний матеріал у курсах з програмування, розробки ігор та штучного інтелекту.

КЛЮЧОВІ СЛОВА: СТРАТЕГІЧНА ГРА, ПРОГРАМУВАННЯ, PYTHON, ШТУЧНИЙ ІНТЕЛЕКТ, АЛГОРИТМИ, ГЕЙМПЛЕЙ, ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ, АГЕНТНЕ МОДЕЛЮВАННЯ.

ABSTRACT
AT QUALIFICATION BACHELOR WORK «DEVELOPMENT OF A STRATEGIC GAME WITH ARTIFICIAL INTELLIGENCE ELEMENTS IN PYTHON»

Arina Palamarchuk

The qualification bachelor's thesis contains 58 pages, 2 tables, 18 figures, a list of references with 19 titles.

The object of the research is the process of creating strategic computer games using artificial intelligence technologies.

The subject of the research is methods and means of implementing artificial intelligence in strategic games using the Python programming language.

The purpose of the qualification bachelor's thesis is to develop a prototype of a strategic computer game with built-in elements of artificial intelligence based on the Python programming language, which provides modeling of dynamic interaction between the player and the virtual opponent, as well as analysis of the effectiveness of relevant algorithms and technological solutions.

The tasks of the qualification bachelor's thesis are:

- to analyze the theoretical foundations, stages of evolution, specifics of designing strategic games, and also to determine the role of artificial intelligence in their implementation;
- to investigate modern algorithmic approaches to modeling the behavior of virtual agents in strategy games, in particular the A*, Minimax, FSM algorithms, etc;
- to consider the capabilities of the Python programming language for developing game applications with structured architecture, gameplay elements, graphics and artificial intelligence;
- to assess the prospects for the development of strategic games with AI implemented using Python, taking into account current trends, the effectiveness of technologies and areas for their further improvement.

Relevance of the study. The computer game industry demonstrates a stable growth trend, with the implementation of adaptive systems based on artificial intelligence playing an important role. Using Python as a programming language with a wide range of tools for implementing intelligent agents opens up new opportunities for creating innovative game solutions. In this context, the study devoted to creating a strategic game with AI is timely and practically significant.

According to the results of the study: a functional prototype of a strategic game with basic elements of the opponent's intellectual behavior was created.

The practical novelty of the work lies in the development of an approach to creating a strategic game using modular architecture and typical artificial intelligence algorithms (using Python as an example), which can be applied both in the educational process and for demonstration or experimental purposes.

The results obtained can be used as educational and methodological material in courses on programming, game development and artificial intelligence.

KEYWORDS: STRATEGIC GAME, PROGRAMMING, PYTHON, ARTIFICIAL INTELLIGENCE, ALGORITHMS, GAMEPLAY, INTELLIGENT SYSTEMS, AGENT MODELING.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧОК, СИМВОЛІВ І ТЕРМІНІВ

AI (Artificial Intelligence) – штучний інтелект

CPU (Central Processing Unit) – центральний процесор

FPS (Frames Per Second) – кількість кадрів за секунду

GUI (Graphical User Interface) – графічний інтерфейс користувача

IDE (Integrated Development Environment) – інтегроване середовище розробки програмного забезпечення

OOP (Object-Oriented Programming) – об’єктно-орієнтоване програмування

RAM (Random Access Memory) – оперативна пам’ять комп’ютера

SDK (Software Development Kit) – набір програмних інструментів

ШІ – штучний інтелект

Python – мова програмування

Pygame – бібліотека Python

Алгоритм A* – евристичний пошуковий алгоритм

Minimax – алгоритм прийняття рішень у іграх з нульовою сумою

ВСТУП

У сучасному цифровому суспільстві штучний інтелект (ШІ) відіграє дедалі важливішу роль, охоплюючи широке коло галузей — від медицини й транспорту до фінансів, освіти та інформаційних технологій. Однією з динамічно зростаючих сфер застосування ШІ є ігрова індустрія, зокрема розроблення стратегічних комп’ютерних ігор. Інтеграція інтелектуальних алгоритмів у відеоігри відкриває нові горизонти для створення більш реалістичних, адаптивних і складних ігрових сценаріїв, що суттєво підвищує залучення та інтерес користувачів.

Стратегічні ігри є одним із найбільш захоплюючих жанрів відеоігор, які вирізняються високим рівнем складності, глибиною ігрового процесу, необхідністю довгострокового планування та прогнозування наслідків прийнятих рішень. Їх ключовою особливістю є вплив кожної дії гравця на подальший розвиток подій у грі. Це вимагає від користувача не лише оцінки поточної ситуації, але й побудови стратегії з урахуванням множини факторів, що впливають на ігрову динаміку. Відповідно, створення подібних ігор потребує впровадження новітніх підходів до моделювання поведінки опонентів, що, у свою чергу, зумовлює актуальність використання технологій ШІ.

Завдяки можливостям штучного інтелекту, стратегічні ігри здатні адаптуватися до дій гравця, реагуючи на зміну його стилю гри та створюючи віртуальних противників із більш реалістичною, гнучкою поведінкою. Такі підходи забезпечують унікальний досвід у кожній ігровій сесії, підвищуючи рівень залученості користувача та реіграбельність проєкту.

У межах даної роботи досліджується процес створення прототипу стратегічної гри з використанням мови програмування Python та інтеграцією алгоритмів штучного інтелекту. Основна увага приділяється побудові ігрової логіки, моделюванню поведінки віртуального супротивника та адаптації рівня складності гри залежно від дій користувача. Особливу роль у реалізації

проєкту відіграє бібліотека Pygame, яка надає зручні інструменти для побудови 2D-ігор і взаємодії з графічним інтерфейсом.

Метою кваліфікаційної бакалаврської роботи є розробка прототипу стратегічної комп'ютерної гри із вбудованими елементами штучного інтелекту на основі мови програмування Python, що забезпечує моделювання динамічної взаємодії між гравцем і віртуальним супротивником, а також аналіз ефективності відповідних алгоритмів і технологічних рішень.

Завданнями кваліфікаційної бакалаврської роботи є:

- проаналізувати теоретичні засади, етапи еволюції, специфіку проєктування стратегічних ігор, а також визначити роль штучного інтелекту в їх реалізації;

- дослідити сучасні алгоритмічні підходи до моделювання поведінки віртуальних агентів у стратегічних іграх, зокрема алгоритми A*, Minimax, FSM тощо.

- розглянути можливості мови програмування Python для розробки ігрових застосунків зі структурованою архітектурою, елементами геймплею, графіки та штучного інтелекту.

- оцінити перспективи розвитку стратегічних ігор із ШІ, реалізованих засобами Python, з урахуванням актуальних трендів, ефективності технологій та напрямів їх подальшого вдосконалення.

Актуальність обраної теми зумовлена зростаючим попитом на інтелектуальні ігрові системи та швидким розвитком інструментальних засобів для створення адаптивних ігрових середовищ. Отримані результати можуть бути використані як основа для подальших досліджень та практичних розробок у сфері геймдизайну, програмування й штучного інтелекту.

За результатами дослідження: створено функціональний прототип стратегічної гри з базовими елементами інтелектуальної поведінки супротивника.

Практична новизна роботи полягає у розробці підходу до створення стратегічної гри з використанням модульної архітектури та типових алгоритмів штучного інтелекту (на прикладі Python), що може бути застосований як у навчальному процесі, так і для демонстраційних або експериментальних цілей.

Одержані результати можуть бути використані як навчально-методичний матеріал у курсах з програмування, розробки ігор та штучного інтелекту.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ СТВОРЕННЯ СТРАТЕГІЧНИХ ІГОР

1.1. Класифікація та особливості стратегічних ігор

Стратегічні ігри — це жанр відеоігор, у яких ключовим елементом є планування дій гравця з урахуванням доступних ресурсів, часу, розміщення юнітів або побудов [1,3]. У цих іграх гравець часто стикається з необхідністю приймати складні рішення, що впливають на подальший розвиток подій. Основною метою є досягнення перемоги через ефективне управління ресурсами, створення стратегічних планів і прийняття рішень у межах обмежених умов.

Залежно від характеру геймплею стратегічні ігри поділяються на:

- покрокові стратегії (TBS — Turn-Based Strategy) — це коли гравці роблять дії по черзі, що дає можливість проаналізувати ситуацію яка виникла на полі бою або в рамках стратегічної карти. Завдяки цьому гравці можуть ретельно обдумувати свої ходи без обмеження часу, що робить гру більш комфортною та продуманою. Приклад: Civilization, XCOM [4];
- стратегії в реальному часі (RTS — Real-Time Strategy) — це події які відбуваються саме в цей момент без натискання паузи, вони вимагають швидкого прийняття рішень і негайної реакції. Важливість цих ігор складається не лише на стратегічному плануванні, але й на вмінні швидко адаптуватися на зміни у грі, які створюють напругу та хвилювання. Приклад: StarCraft [2];
- глобальні стратегії — це знання історії та політики, вміння створювати глобальні стратегії. Вони охоплюють багато сфер економіку , політику, військову сферу. Гравець бере на себе керування великою державою, планує внутрішню та зовнішню політику, керує війнами. Ця система вимагає стратегічне мислення в глобальному маштабі. Приклад: Hearts of Iron [1]. Особливості стратегічних ігор: високий рівень гравцевого

контролю над подіями; пріоритет логіки та планування над швидкістю реакції; складна взаємодія між ігровими елементами та системами [5];

- тактичні стратегії — в цих іграх йде зосередження на управлінні окремими гравцями або групами гравців у конкретних місцях, на відміну від глобальних стратегій, де об'єктом контролю є великі території. Ці ігри більш ставлять акцент на використання тактики, плану та розповсюдження сил у конкретних умовах. Приклад: Advance Wars, Fire Emblem [1]. У цьому типі стратегічних ігор немає незначних моментів, де гравець, дивлячись майже крізь землю, не лише бачить світ, а й успішно протистоїть йому – кожен крок і кожен рух є вирішальними. Тут гравець розробляє стратегію: керує, очолює армії, розподіляє ресурси, створює альянси або воює. Такий рівень контролю глибоко занурює у гру: перемоги та поразки залежать від власних зусиль та рішень гравца [4].

У стратегічній грі головне не швидкість, а інтелект. Тут не потрібно натискати всі кнопки одночасно, принаймні не так, як у бою віч-на-віч — у гравця є час подумати, налаштуватися, розрахувати, спланувати. І це надає грі такої унікальної атмосфери: переможець — не найшвидший, а найрозумніший [1,4,3].

У стратегії все глибоко взаємопов'язане — ніщо не існує ізольовано. Рішення, які приймаються в економіці, впливають на силу армії; політичні пакти впливають на нові підрозділи та постачання; технології визначають тактику, яку можливо застосовувати; успішний прорив у технологіях може вирішити кампанію. І щоб не просто грати, а й вигравати потрібно розуміти, як все взаємопов'язано, і вміти делікатно орієнтуватися в цьому мережевому світі [5].

Стратегічні ігри часто мають високу реіграбельність завдяки випадковим елементам, таким як генерація карт, різноманіття стратегій і варіативність результатів залежно від вибору гравця. Це дозволяє кожному новому проходженню гри бути унікальним [3].

Глибока адаптація до стилю гри у деяких стратегічних ігр дозволяє змінювати рівень складності, щоб відповідати здібностям гравця. Це створює більш персоналізований досвід і дозволяє новачкам і досвідченим гравцям знаходити відповідний виклик [2].

Мультитаскінг та багатозадачність - в RTS іграх, де дія відбувається в реальному часі, гравці повинні управляти кількома аспектами гри одночасно, наприклад, будувати бази, тренувати війська, захищатися від ворогів та планувати наступ. Крім того, стратегічні ігри можуть включати елементи інших жанрів, наприклад, рольових ігор (RPG), де є персонажі, що розвиваються, або економічних симуляторів, що додає глибини ігор [6].

1.2. Основні принципи побудови геймплею у стратегічних іграх

Побудова геймплею у стратегічній грі залежить від жанру, обраної тематики та механік. Основними принципами є:

- Баланс ресурсів — управління обмеженими засобами є основою геймплею. Ресурси можуть бути різними: від матеріальних (дерево, камінь, золото) до людських (військові юніти, робочі одиниці) або енергетичних (ману, енергію тощо). Гравець повинен оптимально використовувати наявні ресурси для досягнення переваги в грі, що сприяє розвитку тактичного та стратегічного мислення [5].
- Розвиток та прогресія — гравець повинен мати змогу покращувати свої можливості, такі як армія, будівлі, технології, а також адаптувати стратегії відповідно до змін у грі. Розвиток зазвичай відбувається в кілька етапів: від початкових обмежених ресурсів до глобальних і потужних стратегічних можливостей [4].
- Противники або суперники — штучний інтелект (ШІ) або інші гравці виступають як основне джерело виклику. ШІ може варіюватися від простих алгоритмів до складних адаптивних систем, здатних міняти стратегії залежно від дій гравця. Гравці, які змагаються один з одним, також

стимулюють розвиток стратегічних рішень, оскільки кожен з них намагається перевершити іншого [7].

- Сценарії та місії — це цілі, які варіюються від знищення супротивника до дипломатичної перемоги або навіть економічного домінування. Головні місії можуть бути як довгостроковими, так і короткостроковими, що дає можливість різноманітних підходів у геймплеї. Важливою складовою є здатність гри адаптуватися до дій гравця, змінюючи сценарії та вимоги місій [8].
- Географія та ігрове поле — розташування має значний вплив на тактику. Ігрове поле може бути поділено на різні території, кожна з яких має свої особливості, такі як ресурси, укріплення, ландшафтні перешкоди, що створюють перевагу або обмеження. Врахування географії допомагає гравцеві приймати правильні тактичні рішення, будувати оборону або проводити ефективні атаки [6].
- Тактичне мислення та адаптація до ситуації — стратегічні ігри вимагають від гравців здатності швидко адаптуватися до змінюваних умов. Залежно від ситуації на полі бою, наявних ресурсів і поточної стратегії, гравець має постійно змінювати тактику, що також залежить від зовнішніх факторів, таких як дії суперників або випадкові події в грі [7].
- Моральний вплив та дипломатія — у деяких стратегічних іграх важливу роль відіграють дипломатичні взаємини. Гравець може мати можливість укладати альянси, обмінюватися ресурсами або навіть маніпулювати іншими гравцями через шантаж або домовленості. Моральні дилеми також можуть додавати глибини грі, змінюючи стратегії у залежності від етичних виборів гравця [8].
- Динамічний розвиток подій — у багатьох стратегічних іграх ігровий світ змінюється залежно від дій гравця або інших учасників. Це може включати зміну погоди, економічних умов, розвитку технологій або політичної ситуації, що вимагає від гравця постійного моніторингу та адаптації своїх стратегій [7].

- Множинність варіантів розвитку подій — стратегічні ігри часто пропонують багато шляхів досягнення перемоги або виконання цілей. Це дозволяє гравцеві вибирати між різними стратегіями і шукати оптимальні варіанти розвитку подій, створюючи варіативність і реіграбельність гри [5].
- Механіка часу — у деяких стратегічних іграх час є важливим фактором для досягнення успіху. Наприклад, в реальному часі стратегічні ігри вимагають постійної уваги до кожної хвилини, тоді як у покрокових іграх час дає гравцю більше можливостей для планування та розмірковування [4].

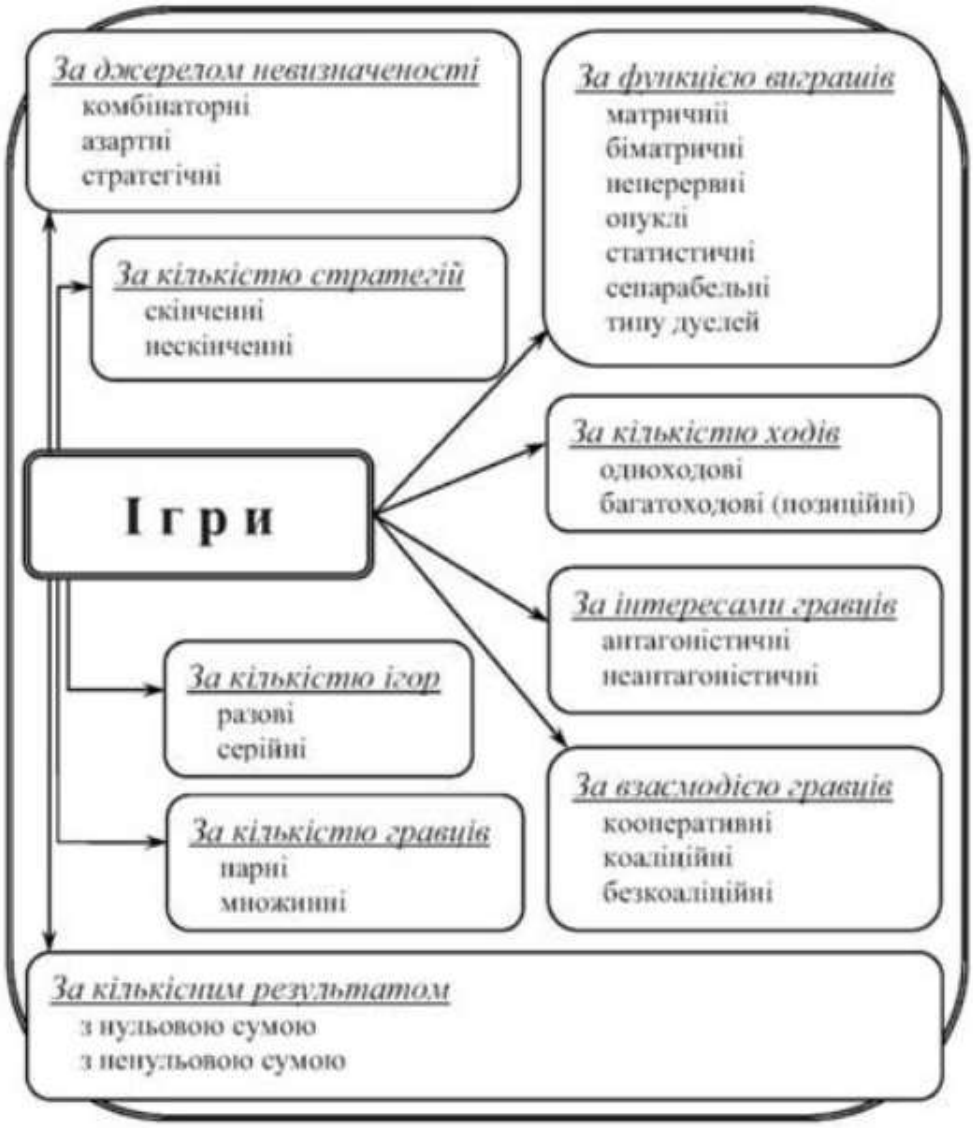

Рис. 1.1. Побудова геймплею у стратегічних іграх

1.3. Архітектура ігрових систем: механіка, логіка, взаємодія

Архітектура стратегічної гри визначається поєднанням кількох основних елементів, які взаємодіють між собою і забезпечують цілісність ігрового процесу. Кожен компонент архітектури грає важливу роль у створенні цікавого і динамічного досвіду для гравця. Основні складові архітектури стратегічних ігор:

- Ігрова механіка — це набір правил, які визначають, як працює ігровий процес і як гравець може взаємодіяти з ігровим світом. Вона включає механізми управління ресурсами, розвиток технологій, боротьбу між юнітами, механізм перемоги та інші аспекти, що забезпечують функціональність гри. Ігрова механіка встановлює основу для прийняття стратегічних рішень і визначає можливості для гравця [4].
- Ігрова логіка — це реалізація алгоритмів і систем, які визначають, як гри працюють на рівні даних. Наприклад, це може бути алгоритм, що обчислює кількість пошкоджень, які отримує юніт під час бою, механізм управління ресурсами, визначення умов перемоги або поразки, або взаємодія з іншими елементами світу гри. Ігрова логіка є важливою складовою, оскільки вона визначає, як система реагує на дії гравця і забезпечує інтерактивність та динамічність ігрового процесу [6].
- Інтерфейс взаємодії (UI) — це засоби, через які гравець може впливати на ігровий світ. В інтерфейсі часто використовуються елементи управління юнітами, відображення інформації про ресурси, здоров'я, статуси одиниць, карти, системи меню та інші візуальні елементи. Якість інтерфейсу безпосередньо впливає на зручність ігрового процесу та на здатність гравця швидко приймати правильні рішення [5].
- Ігровий рушій — це програмне середовище, яке забезпечує всі аспекти реалізації гри, від рендеринга до обробки взаємодій. Ігровий рушій визначає, як візуалізується ігровий світ, як обробляються фізичні та логічні

процеси, і як злагоджено взаємодіють усі компоненти гри. Прикладом ігрових рушіїв є Unity та Unreal Engine, але також можна використовувати мови програмування (наприклад, Python) та бібліотеки (наприклад, Pygame або Pyglet), що дозволяють створювати кастомізовані ігрові рушії для специфічних потреб [13].

Крім того, є кілька інших важливих аспектів, що впливають на загальну архітектуру ігрових систем:

- Штучний інтелект (ШІ) — важлива частина, особливо в стратегічних іграх. ШІ може бути реалізований для того, щоб забезпечити адаптивну поведінку супротивників, яка змінюється залежно від дій гравця. Наприклад, у стратегії з елементами бою ШІ може вибирати тактики для атаки або оборони, що змушує гравця постійно адаптувати свої стратегії до змінюваної ситуації на полі бою [7].
- Системи збереження і прогресії — ці компоненти дозволяють гравцеві зберігати свій прогрес у грі, підтримуючи безперервний ігровий досвід. Прогресія включає розвиток персонажів, набір нових технологій чи побудову нових об'єктів, що забезпечує відчуття досягнення та стимулює гравця продовжувати гру [6].
- Мережеві компоненти — у багатьох стратегічних іграх важливо забезпечити можливість взаємодії між гравцями через мережу. Це може включати обмін даними про стан гри, підключення до серверів для мультиплеєрного режиму або використання хмарних сервісів для збереження прогресу [8].
- Анімація та візуалізація — хоча стратегічні ігри часто мають текстові або 2D інтерфейси, правильна візуалізація світу та елементів гри може значно покращити загальне враження від гри. Візуальні ефекти можуть допомогти передати атмосферу, а також надати інформацію про статуси юнітів, території та інші важливі елементи гри [11].
- Звук — аудіо в стратегічних іграх може додавати атмосферу, посилювати емоційний вплив та покращувати сприйняття ігрових подій. Це

може бути звук битв, музика, голосові повідомлення чи ефекти, що допомагають створити відчуття присутності у грі [12].

Ці компоненти тісно пов’язані і формують основу будь-якої стратегічної гри. Вони забезпечують взаємозв’язок між гравцем, штучним інтелектом, сценарієм та візуалізацією, що дозволяє створити захоплюючий та динамічний досвід для користувача. Всі ці елементи повинні бути належним чином оптимізовані та інтегровані, щоб створити збалансовану та ефективну ігрову систему.

1.4. Психологічні аспекти взаємодії гравця зі стратегічною грою

Психологія гравця відіграє ключову роль у розробці стратегічних ігор. Гравці стратегій зазвичай схильні до аналітичного мислення, планування та отримання задоволення від подолання складних завдань:

- мотивація (ігри задовольняють потребу в контролі, досягненні цілей, розв’язанні логічних задач) [13];
- когнітивне навантаження (гравець активізує пам’ять, увагу, навички планування) [14];
- залучення (розгалужені сценарії, багатофакторна взаємодія сприяють тривалому утриманню уваги) [15].

Стратегічні ігри часто є джерелом стресу для гравців через необхідність приймати важливі рішення за обмежений час, від яких залежить успіх або поразка [16]. Стрес може бути як позитивним стимулом, так і негативним фактором, що знижує здатність до концентрації. Важливо враховувати цей аспект при створенні ігор, щоб підтримувати баланс між викликами та досягненням успіху [17].

З кожним рівнем складності гравець стикається з новими викликами. Стратегії можуть стати все більш складними, і гравець повинен адаптувати свої дії, застосовуючи нові стратегії. Це створює відчуття досягнення після успішного виконання складних завдань і підвищує рівень залученості [18].

Рис. 1.2. Психологічні взаємодії гравця з грою

Розуміння психологічних чинників дозволяє глибше адаптувати гру під потреби цільової аудиторії, підвищити залученість та рівень задоволення гравця. Ігрові дизайнери можуть використати ці знання для створення механік, які не тільки підтримують інтелектуальний виклик, але й знижують рівень фрустрації, допомагаючи гравцям підтримувати позитивний емоційний зв'язок з грою [2,3,5].

Додаткові аспекти психологічної взаємодії:

- емоційна прив'язаність до персонажів гри. Виграючи битви або досягаючи цілей, гравець почувається відповідальним за їхній успіх, що може підвищити емоційну залученість [4,6,8];
- ігрові стимули та винагороди. Механіка досягнень, здобуття нових можливостей або обладунків може стимулювати гравця продовжувати гру, прагнучи до нових цілей і значущих досягнень [4,5];
- групова динаміка та соціальна взаємодія. У багатокористувацьких стратегічних іграх важливою частиною є взаємодія з іншими гравцями. Це може викликати відчуття колективної відповідальності або конкуренції, що ще більше збільшує емоційну залученість [3,9].

РОЗДІЛ 2
ШТУЧНИЙ ІНТЕЛЕКТ ЯК СКЛАДОВА СУЧАСНОЇ ІГРОВОЇ ІНДУСТРІЇ

2.1. Поняття штучного інтелекту та його роль в іграх

Штучний інтелект (ШІ) у контексті відеоігор — це сукупність алгоритмів і систем, що імітують інтелектуальну поведінку віртуальних персонажів або ігрового середовища. Основна мета ШІ — створення ілюзії розумної поведінки, яка робить гру цікавішою, непередбачуванішою та реалістичнішою [6,9].

У стратегічних іграх ШІ особливо важливий, оскільки саме він забезпечує поведінку супротивників, здатних адаптуватися до дій гравця.

Роль ШІ в таких проєктах включає:

- прийняття рішень (атака, оборона, відступ);
- управління ресурсами та розвитком;
- аналіз ситуацій і вибір найоптимальнішої дії;
- побудову довгострокових стратегій [2,7,8].

Рис. 2.1. Штучний інтелект. Його роль в грі

2.2. Методи реалізації ШІ у стратегічних іграх

Штучний інтелект у стратегічних іграх використовує різноманітні методи і підходи для створення адаптивних та динамічних ігрових сценаріїв. Застосування ШІ дозволяє створювати віртуальних супротивників, які можуть реагувати на дії гравця, приймати стратегічні рішення, а також змінювати свою поведінку залежно від ситуації. Серед найбільш поширених методів реалізації ШІ у стратегічних іграх виділяються наступні:

- Скриптована поведінка. Це метод, при якому ШІ діє згідно заздалегідь визначеного набору правил, що описують реакції на певні події в грі. Цей підхід забезпечує простоту реалізації та високий рівень передбачуваності, оскільки дії супротивника чітко визначені. Однак обмеження цього методу полягає в тому, що поведінка супротивника залишається статичною і не адаптується до змін у грі. Зазвичай використовується в іграх з менш складною логікою або для створення сценаріїв, де важлива передбачуваність дій [4,6].
- Скінченні автомати станів (Finite State Machines, FSM). Це модель, в якій об'єкт (наприклад, супротивник) перебуває в одному з кількох можливих станів, і змінює стан в залежності від умов ігрового середовища. Кожен стан характеризується певними діями, а перехід між станами відбувається за певними правилами. Такий підхід дозволяє створити більш гнучку поведінку порівняно зі скриптованою поведінкою. Однак для складних ігрових ситуацій ця модель може стати надто громіздкою, оскільки потребує великої кількості станів і переходів [2,7].
- Агенти на основі правил (Rule-Based Agents). Це система, яка працює за допомогою набору логічних правил, що визначають, як агент повинен реагувати на певні ситуації. Вони дозволяють створювати досить гнучкі стратегії, але також мають обмеження в складних випадках, коли для правильного прийняття рішень необхідні більш складні моделі. Агенти на основі правил часто використовуються в іграх, де потрібно обробляти великі

обсяги простих умов, як наприклад, розподіл ресурсів чи управління юнітами [6,7].

- Дерев рішень. Це метод, що дозволяє моделювати процес прийняття рішень як дерево, де кожне розгалуження вказує на можливі дії залежно від обраних умов. Дерево рішень дозволяє чітко визначити всі можливі варіанти розвитку подій, даючи змогу супротивнику приймати логічні, поетапні рішення. Цей підхід ефективно працює в іграх, де необхідно враховувати численні варіанти розвитку подій, але може бути обмеженим у випадках, коли потрібно враховувати складні стратегії чи динамічні зміни в реальному часі [2,9].
- Машинне навчання. Використовується в більш складних системах, де ШІ може адаптуватися до змін середовища і стратегії гравця. Завдяки алгоритмам машинного навчання, такі системи можуть покращувати свою ефективність і здатність реагувати на нові стратегії. Вони можуть навчатися на основі даних про попередні ігри, що дозволяє їм постійно адаптуватися і змінювати свою поведінку. Основними перевагами є висока адаптивність та здатність до самонавчання, однак цей метод потребує великих обчислювальних ресурсів і складної реалізації [6,9,17].

Кожен з перерахованих методів має свої переваги та недоліки. Наприклад, скриптована поведінка є простим і ефективним методом для базових ігор, але обмежена в здатності до адаптації [7]. У той час як методи на основі машинного навчання можуть створити інтелектуальних супротивників, здатних до адаптації і самовдосконалення, вони вимагають значних обчислювальних ресурсів і складної реалізації [9,17].

Вибір методу залежить від складності гри, бажаного рівня інтелекту супротивника та технічних можливостей. Наприклад, для ігор з великою кількістю учасників і змінюваними умовами найбільш ефективними будуть методи на основі машинного навчання чи агентів на основі правил [6,9]. Для більш простих ігор, де дії супротивників можуть бути передбачуваними,

достатньо буде використання скриптованої поведінки або скінченних автоматів станів [7,8].

Загалом, успішна реалізація ШІ у стратегічних іграх дозволяє значно підвищити рівень інтелектуального виклику для гравця, роблячи гру більш динамічною і захоплюючою.

На рис. 2.2 зображено карта популярної гри Mobile Legends.

Рис. 2.2. Карта гри Mobile Legends

Mobile Legends — це гра більше, ніж просто бій. Вона про команду, думку, і правильний момент. Гра включає в себе багато стратегій та виборів. Мета цієї гри – зберегти свою базу у цілісності. Ця гра є командною, яка має 5 гравців і 5 противників. П’ятеро гравців, кожен із унікальною роллю: танк, маг, стрілець, вбивця, боєць об’єднуються заради однієї мети - знищити ворожу базу. Але шлях до перемоги — це вибір: тиснути чи відступити, кого прикрити, кого знищити вчасно, як захопити карту.

Здача мага наносити потужний урон на ворога та завдавати контроль, особливий урон він починає давати в середині або на при кінці гри, він повинен йти по середній лінії, вони тільки для мага. Його стратегія полягає в тому щоб захищати вежі своїх союзників, давати правильний урон по

противникам, прибирати міньйонів із своєї лінії, щоб вони не зруйнували вежу.

Міньйони – це маленькі помічники, вони допомагають зруйнувати ворожу вежу, та допомагають де коли навіть вбити противника. Але коли міньйони ворога, то вони допомагають ставати сильніше гравцю, якщо їх вбити.

Стратегія стрілка – це нанесення постійного високо фізичного урону на дистанції, особливо у пізній грі. Його завдання — вижити, розфармитися (ставати сильніше) й знищувати ворогів у ключові моменти бою. Стрілки йдуть на одній лінії з танком.

Танк — це щит команди, перший, хто заходить у бій, і останній, хто залишає поле бою. Його головна стратегія — захищати союзників, ініціювати сутички та контролювати ворогів. Він не про вбивства, а про виживання і створення простору для дій решти команди. Танк повинен допомагати усій команді.

Стратегія бійця - це баланс між силою та витривалістю. Він здатен і завдати шкоди, і витримати удари. Його стратегія полягає в тому, щоб бути універсальним воїном на передовій, який може як вести атаку, так і захищати союзників. Бійці стоять одні на своїй лінії, але коли потрібна допомога то він повинен піти у бій.

Стратегія вбивці-лісника — це герой, чия головна мета — знищувати вразливих ворогів (маги, стрілки) та контролювати джунглі. Його сила у раптових атаках, швидкості та мобільності. В кожної команди є свої джунглі в яких сидять дикі тварини, яких потрібно вбивати для фарму (збільшення сили героя) ці тварини через деякий час повертаються і їх можна вбити ще раз.

Стратегія гри тут у всьому — від складу команди до останнього удару по лорду. Перемагає не той, хто швидше тисне кнопки, а той, хто думає, відчуває гру. Саме це робить Mobile Legends по-справжньому захопливою [19].

2.3. Порівняльна характеристика алгоритмів ШІ

У стратегічних іграх часто використовуються такі алгоритми:

- A* — пошук найкоротшого шляху, ідеально підходить для навігації юнітів [7,9];
- Minimax — використовується в покрокових стратегічних іграх для побудови дерева можливих дій гравця і супротивника [6,7];
- Alpha-Beta Pruning — оптимізація алгоритму Minimax [6];
- FSM (Finite State Machine) — проста та ефективна реалізація для NPC, що перебувають у чітких станах [7,9];
- Behavior Trees — розширення FSM, більш гнучка система управління поведінкою [2,3];
- Monte Carlo Tree Search — імовірнісний метод прийняття рішень, використовується в складних іграх (наприклад, Go або Total War) [917].

Порівняльна характеристика алгоритмів ШІ наведена у табл. 2.1

Таблиця 2.1

Порівняльна таблиця алгоритмів ШІ.

ШІ	Опис	Переваги	Недоліки	Приклади використання
FSM (скінченний автомат)	Поведінка розділена на стани з чіткими переходами між ними	Простий у реалізації, легкий для налагодження	Важко масштабувати, не гнучкий	Вороги у класичних шутерах
Behavior Tree	Ієрархічна структура дій, що дозволяє гнучке управління поведінкою	Гнучкіший за FSM, модульний, легко розширюється	Може бути складним при великій гіллястості	AI в іграх типу *Assassin's Creed*, *Halo*
Pathfinding (A*)	Алгоритм пошуку найкоротшого шляху на мапі	Ефективний, широко застосовуваний, знаходить оптимальні шляхи	Вимагає ресурсів, не враховує поведінкову логіку	RTS, RPG (навігація персонажів)

Продовження табл. 2.1

ШІ	Опис	Переваги	Недоліки	Приклади використання
Minimax / Alpha-Beta	Прийняття рішень у покрокових іграх на основі передбачення ходів	Оптимальне рішення для двогравцевих ігор	Повільний при великій глибині дерева	Шахи, шашки, тактичні бої
Машинне навчання	AI навчається на основі прикладів або через взаємодію з середовищем	Адаптивність, здатність до самоудосконалення	Потрібне навчання, складність реалізації	Ігри зі складною або змінною поведінкою
Rule-Based System	Поведінка задається чітко прописаними правилами	Проста логіка, зрозумілий контроль	Жорсткий, непристосований до змін	NPC з базовою поведінкою в RPG
Нейронні мережі	Моделюють складну поведінку, можуть розпізнавати патерни	Потужні при складних задачах	Важко інтерпретувати, потрібне багато ресурсів	Ігри з адаптивним AI (експериментальні)

2.4. Переваги та виклики впровадження ШІ у відеоігри

Впровадження штучного інтелекту в стратегічні ігри має як позитивні сторони, так і технічні та ігрові виклики.

Переваги:

- динамічність ігрового процесу;
- варіативність дій супротивника;
- підвищення рівня занурення гравця;
- можливість створення адаптивних систем.

Виклики:

- висока складність реалізації в умовах обмеженого бюджету;
- проблеми оптимізації — продуктивність гри може знижуватися;
- ризик “надрозумного” супротивника, що робить гру нечесною;
- необхідність великої кількості тестування для уникнення помилок у логіці.

Рис. 2.3. Переваги та виклики ШІ в стратегіях

ШІ у стратегічних іграх є потужним інструментом, що дозволяє створювати захопливі та реалістичні сценарії. Правильне застосування ШІ значно підвищує якість гри, а також робить її цікавою навіть у відсутності багатокористувацького режиму [7,9].

РОЗДІЛ 3
ЗАСОБИ РОЗРОБКИ ІГОР НА PYTHON

3.1. Огляд бібліотек Python для розробки ігор (Pygame, Arcade, Panda3D)

Python є однією з найпопулярніших мов програмування, зокрема й у сфері розробки ігор. Його простота, кросплатформеність та велика кількість бібліотек роблять його зручним вибором для створення 2D- та 3D-ігор. Серед основних бібліотек варто виділити:

- Pygame — класична бібліотека для створення 2D-ігор. Підтримує обробку подій, графіку, звук, анімацію. Має простий інтерфейс, добре підходить для навчальних проєктів та прототипування [10];
- Arcade — новіша бібліотека для 2D-ігор, заснована на OpenGL. Забезпечує покращену графіку, простий API, добру продуктивність [13];
- Panda3D — потужний рушій для створення 3D-ігор. Має вбудовані інструменти рендерингу, фізики, анімації. Хоча складніший у використанні, дає більше можливостей [12].

Параметр	Pygame	Arcade	Panda3D
Тип	2D-движок	2D-движок	3D-движок (+ підтримка 2D)
Мова програмування	Python	Python	Python, C++
Простота використання	Дуже проста для початківців	Простий API, орієнтований на новачків	Складніший, потребує більше знань
Продуктивність	Середня (підійде для нескладних ігор)	Вища за Pygame завдяки OpenGL	Висока (професійний рендеринг)
Графіка	Базова 2D-графіка	Покращена 2D-графіка з анімацією, шейдерами	Потужна 3D-графіка, підтримка шейдерів
Підтримка анімації	Є	Покращена (спрайт-листи, анімаційні переходи)	Повноцінна (кісткова анімація, 3D-моделі)
Інтеграція зі звуком	Так (через модулі)	Обмежена	Є, підтримка мережевої синхронізації
Мережева підтримка	Windows, macOS, Linux	Windows, macOS, Linux	Менш масова, більше проф-спільнота
Документація та спільнота	Дуже велика	MIT	BSD
Ліцензія	Простих 2D-ігор, прототипів	Сучасних 2D-ігор, навчальних проектів	Складних 3D-ігор, VR/AR-проектів

Рис. 3.1. Бібліотека Python для розробки ігор

Кожна з бібліотек має свої сильні сторони. Вибір залежить від типу гри, досвіду розробника та вимог до продуктивності.

3.2. Інструменти для реалізації штучного інтелекту на Python

Python надає широкий спектр бібліотек для створення алгоритмів штучного інтелекту. Вони активно використовуються не лише в машинному навчанні, а й у розробці ігор. Найбільш корисними у цьому контексті є:

- NumPy — основна бібліотека для роботи з масивами, математичними операціями, яка потрібна для реалізації логіки ШІ [6];
- scikit-learn — містить реалізації базових алгоритмів класифікації, регресії, кластеризації. Підходить для простих моделей навчання [6];
- TensorFlow та PyTorch — глибоке навчання, складні нейронні мережі, що можуть використовуватись для навчання супротивників або адаптивної поведінки [6];
- deap — бібліотека для реалізації еволюційних алгоритмів, зручна для створення навчальних агентів [9];
- networkx — для побудови та аналізу графів, часто використовується у навігаційних алгоритмах типу A* [9].

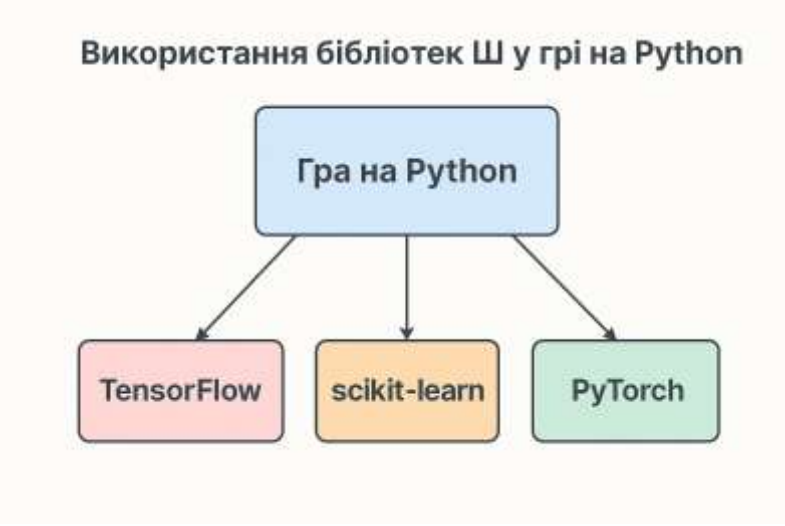

Рис. 3.2. Використання бібліотек ШІ у грі на Python

Завдяки цим інструментом можливо реалізовувати як прості автоматизовані моделі поведінки, так і повноцінних навчальних агентів, що адаптуються до гравця під час гри [6,9].

3.3. Особливості побудови графіки та анімації у стратегічних іграх

Графіка та анімація у стратегічних іграх виконують не лише естетичну функцію, але й суттєво впливають на ігрову механіку, інтерфейс користувача та сприйняття динаміки подій. Для Python доступні інструменти, які дозволяють реалізувати якісну 2D та навіть 3D візуалізацію [6,7].

Вимоги до графіки у стратегічних іграх:

1. Чіткість візуальної інформації — стратегічні ігри вимагають від гравця швидко орієнтуватися на мапі, бачити стани юнітів, споруд та ресурсів [6].
2. Масштабованість — можливість змінювати рівень деталізації (зум), підтримка великих карт. [7]
3. Анімація дій — рух юнітів, атаки, побудови повинні мати просту, але зрозумілу анімацію [7].
4. Інтерфейс користувача (UI) — ігрове меню, панель ресурсів, кнопки дій повинні бути логічно організовані [6].

Python дає змогу реалізовувати графічні елементи на різних рівнях складності:

- Pygame: підтримує завантаження спрайтів, відображення мап, анімацію кадрів. Використовується метод blit() для виводу зображень, а також таймери для зміни кадрів анімації [10];
- Arcade: надає готові методи для роботи зі спрайтами та анімацією, підтримує спрайт-листи (sprite sheets), колізії, шари рендерингу [11];
- Tiled Map Editor: часто використовується з Pygame або Arcade для створення тайлових мап, що зручно для стратегічного жанру [10].

Рис. 3.3. Приклад тайлової карти

Основні методи анімації у 2D-іграх включають:

- кадрову анімацію — зміна зображень спрайту у послідовності;
- плавні трансформації — рух об’єкта через зміну координат за певний час;
- подієва анімація — активація візуальних ефектів при дії (постріл, удар, будівництво).

Для прикладу, у Pygame цикл анімації виглядає так:

```
frame_index = 0
frames= [pygame.image.load(«unit1.png»), pygame.image.load(«unit2.png»)]

def update_animation():
    global frame_index
    screen.blit(frames [frame_index], (x, y))
    frame_index = (frame_index + 1) % len(frames)
```

Таблиця 3.1

Порівняльна таблиця атаки юніота

Кадр	**Зображення (іконка/спрайт)**	**Опис дії**
1	(юніт стоїть)	Статичний стан, юніт очікує
2	(піднята зброя)	Початок анімації атаки
3	(рух вперед із зброєю)	Кульмінація удару
4	(контакт з ворогом)	Завдання шкоди (можна показати ефект)
5	(повернення у вихідну позу)	Завершення анімації

Грамотна візуалізація є запорукою приємного ігрового досвіду. Вона допомагає гравцеві орієнтуватися, прогнозувати події та приймати стратегічні рішення.

3.4. Приклади реалізації ШІ-алгоритмів на Python у простих ігрових задачах

1. Алгоритм обробки бою і логіка ШІ у грі.

У цьому підрозділі розглядається базова реалізація бою між гравцем та ворогом у грі з використанням магічних заклинань. Гра передбачає використання класів для персонажів та заклинань, що дозволяє розділити різні аспекти гри:

- Character: клас, який визначає персонажів (гравець і ворог). Він містить інформацію про здоров'я, ману та метод перевірки живості персонажа;
- Spell: клас для магічних заклинань, що містить назву заклинання, шкоду та вартість мани. Також є метод для використання заклинання, який застосовує його до цілі, якщо вистачає мани;
- MagicBattleGame: головний клас, який управляє грою. Він включає методи для вибору заклинань гравцем, випадкового вибору заклинання ворогом, а також перевірки статусу кожного учасника битви.

Ось приклад коду для реалізації бою, де гравець і ворог по черзі використовують магічні заклинання:

```
import random

# Клас персонажа (гравець і ворог)
class Character:
    def __init__(self, name, health, mana):
        self.name = name
        self.health = health
        self.mana = mana

    def is_alive(self):
        return self.health > 0

# Клас для магічних заклинань
class Spell:
    def __init__(self, name, damage, mana_cost):
        self.name = name
        self.damage = damage
        self.mana_cost = mana_cost

    def cast(self, caster, target):
        if caster.mana >= self.mana_cost:
            caster.mana -= self.mana_cost
            target.health -= self.damage
            print(f"{caster.name} використовує заклинання {self.name} на
{target.name}, наносячи {self.damage} шкоди!")
        else:
            print(f"{caster.name} не вистачає мани для використання
заклинання {self.name}!")

# Основний клас гри
```

```
class MagicBattleGame:
    def __init__(self):
        self.player = Character("Гравець", 100, 50)
        self.enemy = Character("Ворог", 80, 40)
        self.spells =  [
            Spell("Вогняний м'яч", 25, 10),
            Spell("Льодовий шквал", 15, 8),
            Spell("Магічний щит", 0, 5),
            Spell("Електричний розряд", 20, 12)
        ]

    def player_turn(self):
        print("\nТвій хід!")
        print("Оберіть заклинання для використання:")
        for idx, spell in enumerate(self.spells):
            print(f"{idx + 1}. {spell.name} - Шкода: {spell.damage}, Мана:
{spell.mana_cost}")

        choice = int(input("Введіть номер заклинання: ")) - 1
        if 0 <= choice < len(self.spells):
            spell = self.spells [choice]
            spell.cast(self.player, self.enemy)
        else:
            print("Невірний вибір!")

    def enemy_turn(self):
        print("\nХід ворога!")
        spell = random.choice(self.spells)
        spell.cast(self.enemy, self.player)
```

```
        def display_status(self):
            print(f"\nСтатус гравця: Здоров'я: {self.player.health}, Мана: {self.player.mana}")
            print(f"Статус ворога: Здоров'я: {self.enemy.health}, Мана: {self.enemy.mana}")

        def play(self):
            print("Магічна Битва починається!")
            while self.player.is_alive() and self.enemy.is_alive():
                self.display_status()
                self.player_turn()
                if not self.enemy.is_alive():
                    print("\nВітаємо, ви перемогли ворога!")
                    break
                self.enemy_turn()
                if not self.player.is_alive():
                    print("\nВи програли, ворог переміг вас!")
                    break

    if __name__ == "__main__":
        game = MagicBattleGame()
        game.play()
```

2. Основний цикл гри та взаємодія з користувачем.

У грі гравець і ворог по черзі використовують заклинання. Гравець вибирає одне з доступних заклинань для атаки, а ворог атакує випадковим заклинанням. Ігровий процес триває до перемоги одного з учасників [10,11].

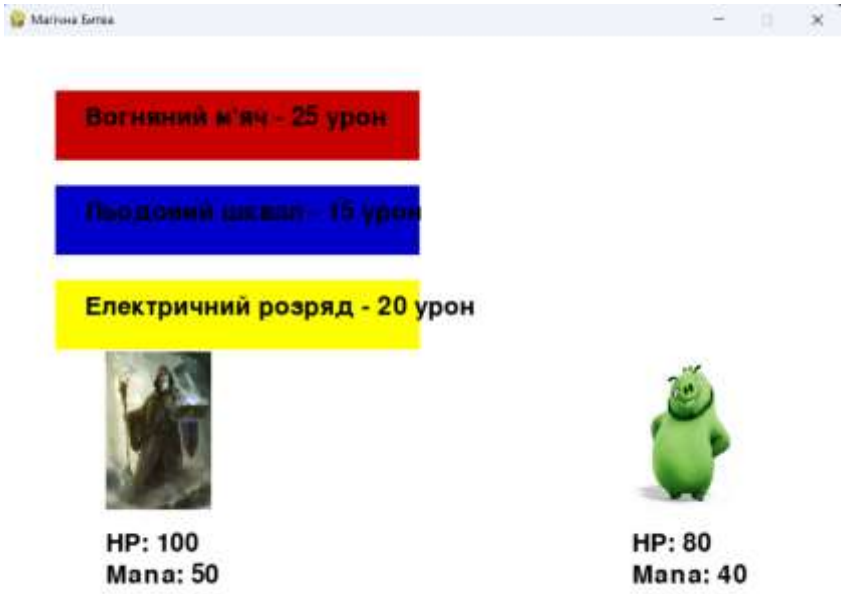

Рис. 3.4. Рисунок стратегічної гри

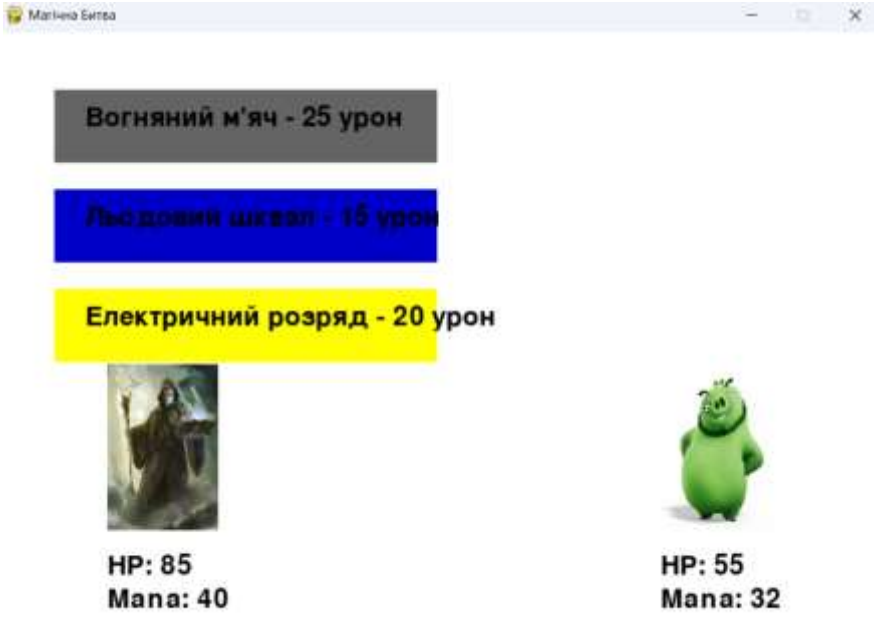

Рис. 3.5. Використання першої кнопки

На рис. 3.5 можна побачити, що після першого удару “Вогняний м’яч” у ворога здоров’я та мани стало менше, і так буде після кожного удару, поки хтось один не переможе.

Рис. 3.6. Використання другої кнопки

Біля кожної кнопки можна побачити кількість урону, який можна задати ворогу. Зараз вже задіяна друга кнопка “Льодовий шквал” (рiс. 3.6).

Рис. 3.7. Використання третьої кнопки

На рис. 3.7 використано третю кнопку “Електричний розряд” та можемо побачити що здоров’я та мани у ворога взагалі дуже мало, нам потрібно для перемоги використати ще одну з запропонованих кнопок.

Це базовий приклад гри з елементами штучного інтелекту, де використовуються прості магічні заклинання. Гру можна доповнювати новими механіками, такими як більш складний штучний інтелект, додаткові можливості для гравця, або навіть інтеграція елементів стратегічних ігор [7,10].

3. Реалізація системи поведінки ворога за допомогою штучного інтелекту.

У попередньому прикладі описано базовий бій між гравцем та ворогом, де ворог використовує випадкові заклинання. Покращити поведінку ворога, можливо додавши до нього алгоритм, який дозволяє йому адаптувати свої дії в залежності від стану гри, а саме — вибір заклинання на основі кількості здоров'я ворога та гравця.

```
class EnhancedAIEnemy(MagicBattleGame):
    def __init__(self):
        super().__init__()

    def enemy_turn(self):
        print("\nХід ворога!")
        # Стратегія: якщо здоров'я ворога низьке, він використовує заклинання для захисту або відновлення
        if self.enemy.health < 30:
            spell = self.get_defensive_spell()
        else:
            spell = self.get_offensive_spell()
        spell.cast(self.enemy, self.player)

    def get_defensive_spell(self):
```

```
        # Якщо здоров'я ворога низьке, він намагатиметься захиститись
        defensive_spells =  [spell for spell in self.spells if spell.name == "Магічний щит"]
        return random.choice(defensive_spells)

    def get_offensive_spell(self):
        # Якщо ворог у хорошому стані, він використовує атакуючі заклинання
        offensive_spells =  [spell for spell in self.spells if spell.name != "Магічний щит"]
        return random.choice(offensive_spells)
```

- EnhancedAIEnemy: наслідує від класу MagicBattleGame, щоб забезпечити основний функціонал гри;
- enemy_turn(): ворог приймає рішення про свою дію залежно від стану здоров'я. Якщо його здоров'я низьке, він використовує захисне заклинання (наприклад, Магічний щит). В іншому випадку він атакує;
- get_defensive_spell(): фільтрує заклинання, що мають оборонний характер, та вибирає випадкове з них;
- get_offensive_spell(): вибирає атакуюче заклинання, якщо ворог здатний атакувати.

4. Покращення штучного інтелекту для більш складної взаємодії.

В цьому підрозділі розглянемо ще одну стратегію для поведінки ворога, яка буде враховувати не лише здоров'я ворога, а й ману для визначення, яке заклинання він використовуватиме. Це дозволяє більш ефективно управляти ресурсами ворога, що робить гру більш цікавою та складною [6,7].

```
class ManaAwareAIEnemy (MagicBattleGame):
    def __init__(self):
        super().__init__()
```

```
    def enemy_turn(self):
        print("\nХід ворога!")
        if self.enemy.mana < 10:
            # Якщо в нього мало мани, він намагається використовувати заклинання з малою вартістю мани
            spell = self.get_mana_efficient_spell()
        else:
            # Якщо мана є, використовує атакуюче заклинання
            spell = self.get_offensive_spell()
        spell.cast(self.enemy, self.player)

    def get_mana_efficient_spell(self):
        # Повертає заклинання з найменшою вартістю мани
        efficient_spells = sorted(self.spells, key=lambda spell: spell.mana_cost)
        return efficient_spells [0]  # Вибираємо найефективніше за маною заклинання
```

- ManaAwareAIEnemy: цей клас враховує не тільки здоров'я, але й кількість мани, що дозволяє ворогу вибирати найбільш ефективне заклинання з точки зору витрати мани;
- enemy_turn(): якщо ворог має мало мани, він вибирає заклинання з найменшими витратами мани, щоб продовжити битву. Якщо мана є, він використовує потужніше атакуюче заклинання;
- get_mana_efficient_spell(): функція, що сортує всі заклинання за витратами мани та вибирає найбільш ефективне [6,7].

5. Поглиблене використання ШІ для динамічних змін у грі.

Ще один цікавий напрямок — це адаптація поведінки ворога в залежності від стратегії гравця. Наприклад, якщо гравець часто використовує одне й те саме заклинання, ворог може спробувати змінити свою стратегію для контратаки або використати інші методи для перемоги [7,8].

```
    class AdaptiveAIEnemy(MagicBattleGame):
```

```
def __init__(self):
    super().__init__()
    self.player_moves =  []

def player_turn(self):
    move = input("\nВаш хід! Виберіть заклинання: ")
    self.player_moves.append(move)
    super().player_turn()

def enemy_turn(self):
    print("\nХід ворога!")
    # Вивчення останнього ходу гравця та адаптація до нього
    if self.player_moves and self.player_moves [-1] == "1":
        print("Ворог помітив, що ви часто використовуєте Вогняний м'яч і контратакує!")
        spell = self.get_defensive_spell()
    else:
        spell = self.get_offensive_spell()
    spell.cast(self.enemy, self.player)
```

- AdaptiveAIEnemy: цей клас запам'ятовує останні дії гравця та адаптується до них. Якщо гравець часто використовує одне й те ж заклинання, ворог може спробувати використовувати більш ефективну контратаку [7,9];
- player_turn(): гравець вводить свій хід, і цей хід зберігається для подальшого аналізу ворогом [8];
- enemy_turn(): ворог реагує на останній хід гравця та намагається змінити стратегію в залежності від вибору гравця [6,7].

З такими алгоритмами штучного інтелекту можна зробити бій в грі більш непередбачуваним і цікавим. Кожен наступний крок ворога буде залежати від дій гравця, що додає стратегічний елемент до гри.

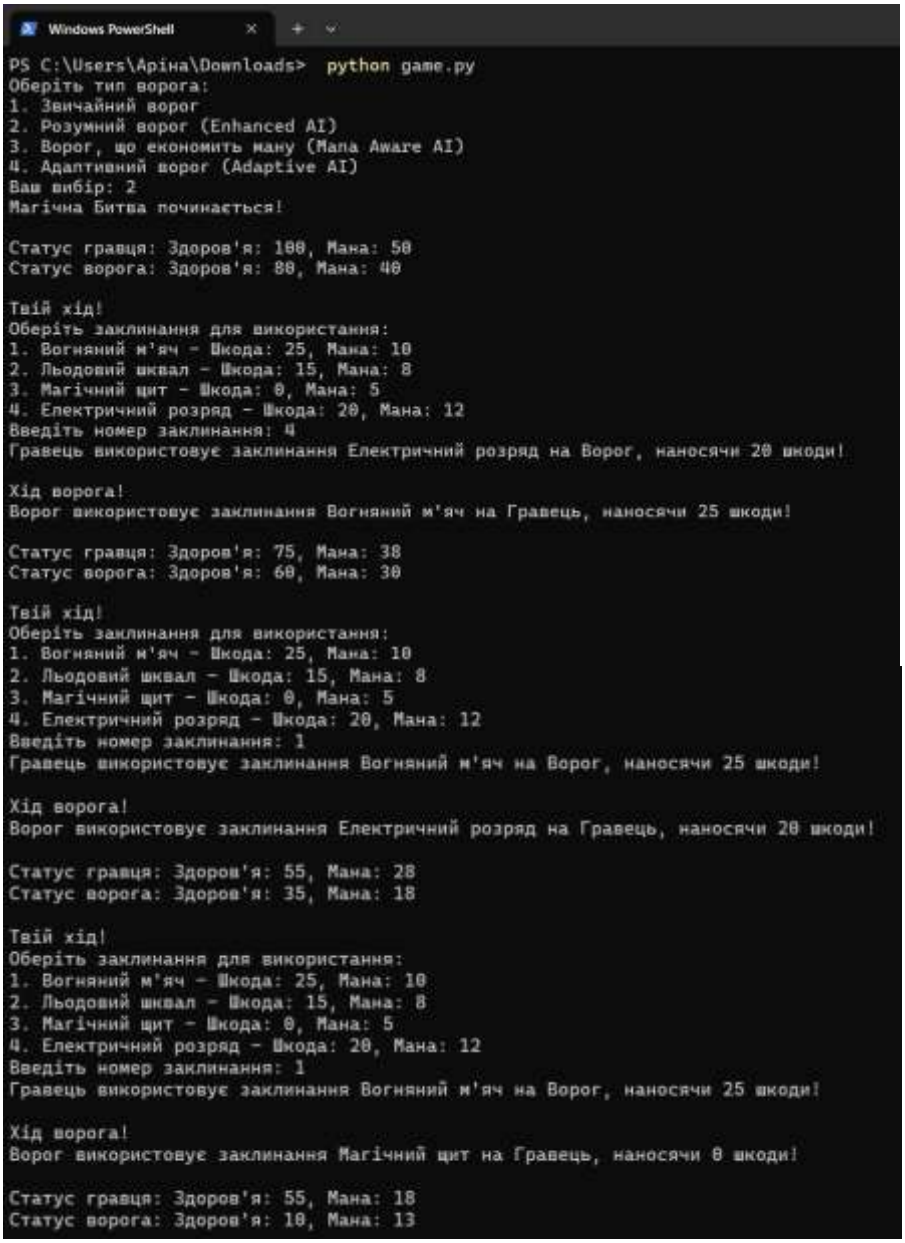

Рис. 3.8. Допрацьований код і відтворений у терміналі

РОЗДІЛ 4
АНАЛІЗ ПЕРСПЕКТИВ РОЗВИТКУ ІГОР ІЗ ШІ НА PYTHON

4.1. Тренди у розробці інтелектуальних ігор

Індустрія ігор, що базуються на штучному інтелекті, останніми роками демонструє стійку тенденцію до зростання і ускладнення. Одним із ключових напрямів є створення систем, що можуть адаптуватися до поведінки гравця, змінюючи стратегії залежно від прийнятих ним рішень [7,9].

Серед важливих трендів виділяються такі:

- індивідуалізація досвіду: ігрові механіки все частіше враховують особисті особливості користувача, створюючи унікальні сценарії та шляхи проходження [7,9];
- інтеграція алгоритмів навчання з підкріпленням: використання агентів, що вчяться у процесі взаємодії з ігровим середовищем, стає основою для побудови більш непередбачуваних і живих супротивників [7,9];
- автоматична генерація контенту: системи, здатні створювати карти, завдання чи персонажів без прямого втручання людини, дають можливість значно урізноманітнити геймплей [6,17];
- покращення моделювання поведінки: ігрові персонажі дедалі частіше демонструють складні моделі поведінки, що враховують емоційний стан, минулий досвід і цілісність сценарію [6,7];
- синтез класичних і сучасних підходів: у проектах поєднуються традиційні алгоритми (наприклад, дерева рішень) і методи глибокого навчання для створення більш гнучких і ефективних систем [6,9]. Важливим аспектом є те, що Python, завдяки своїй простоті, гнучкості та наявності потужних бібліотек для машинного навчання і обробки даних, активно використовується як основна мова для дослідження і розробки ШІ в іграх [6,7,9].

4.2. Ефективність Python у створенні стратегічних ігор з ШІ

Python залишається однією з провідних мов програмування для розробки стратегічних ігор із використанням штучного інтелекту. Його популярність пояснюється низкою об'єктивних переваг, які забезпечують високу ефективність під час створення інтелектуальних ігрових систем.

Однією з ключових переваг є велика кількість спеціалізованих бібліотек, що спрощують реалізацію алгоритмів машинного навчання, обробки даних та побудови нейронних мереж. Такі пакети, як TensorFlow, Keras, PyTorch, Scikit-learn, а також бібліотеки для ігор, як Pygame, дозволяють суттєво скоротити час розробки, концентруючись на логіці гри та поведінці штучного інтелекту [6,7,10].

Python відзначається читабельністю коду та простотою синтаксису, що полегшує підтримку проєктів та швидку інтеграцію нових функцій. Це особливо важливо для командних проєктів або великих ігрових систем, де декілька розробників працюють над єдиною кодовою базою.

Ще одним значним фактором є гнучкість інтеграції з іншими мовами та технологіями. Наприклад, продуктивні обчислювальні модулі можна реалізувати на C або C++, а керуючі компоненти залишити на Python, що дозволяє оптимізувати ресурси гри без втрати зручності розробки.

Python також активно використовується для прототипування: швидка побудова тестових моделей і механік дозволяє оцінити ефективність ідеї без значних витрат часу і ресурсів. Завдяки цьому процес розробки стратегічних ігор із ШІ стає гнучкішим і більш орієнтованим на якість кінцевого продукту.

Таким чином, поєднання потужних інструментів, простоти використання та високої адаптивності робить Python однією з найкращих платформ для створення стратегічних ігор із застосуванням технологій штучного інтелекту [6,7,10].

4.3. Потенційні напрями вдосконалення штучного інтелекту в іграх.

Штучний інтелект у стратегічних іграх продовжує активно розвиватися, відкриваючи нові можливості для підвищення реалізму, складності та індивідуальності ігрового процесу.

Одним із перспективних напрямів є розвиток адаптивного навчання — коли ігровий штучний інтелект аналізує дії гравця та змінює свою стратегію відповідно до його стилю гри. Завдяки цьому ШІ стає менш передбачуваним і забезпечує динамічний та захоплюючий ігровий процес [7,9,18].

Рис. 4.1. Приклад адаптивної поведінки в іграх

Це зображення демонструє віртуального персонажа, здатного до мови, погляду та контролю жестів, що є прикладом адаптивної поведінки в іграх.

Інший важливий напрям — використання глибокого навчання для створення складніших моделей прийняття рішень. Такі моделі дають змогу не лише реагувати на дії гравця, а й формувати власні стратегії, передбачаючи події на основі аналізу минулого досвіду.

Реалістична симуляція емоцій і поведінки персонажів є ще одним перспективним напрямом розвитку. Інтеграція моделей емоцій (рис. 4.2) дозволяє створювати унікальні профілі супротивників, які можуть проявляти страх, агресію або обережність залежно від ситуації в грі [7,9,6].

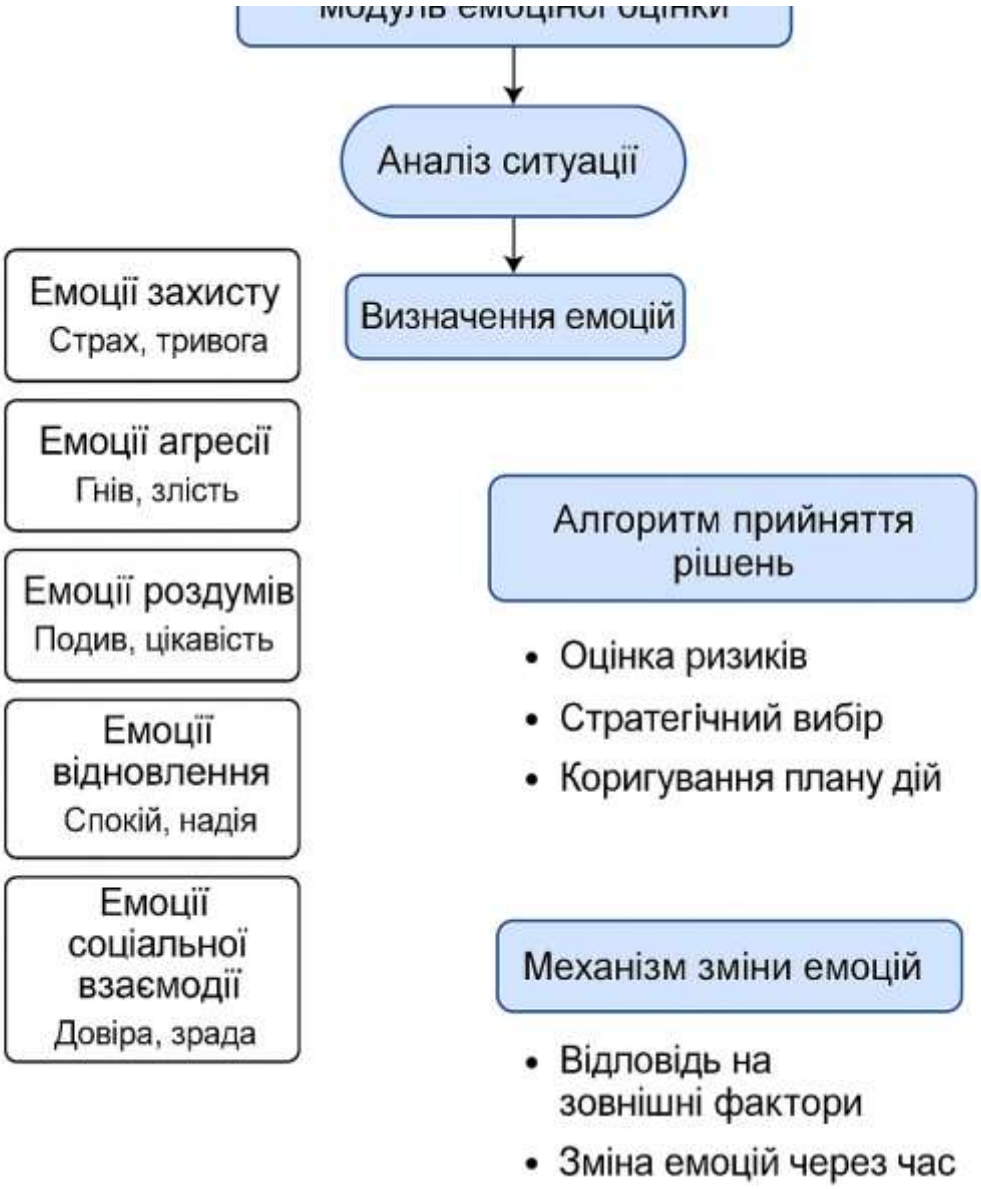

Рис. 4.2. Модуль емоційної оцінки

Важливу роль відіграє також впровадження самонавчальних агентів через підходи підкріпленого навчання (Reinforcement Learning). Це дає можливість створювати супротивників, які вдосконалюють свої навички під час ігрового процесу, без жорстко заданих сценаріїв.

Ще один напрямок удосконалення — оптимізація обчислювальних ресурсів. Важливо забезпечити баланс між складністю моделей штучного інтелекту та продуктивністю гри, що особливо актуально для мобільних ігор або платформ із обмеженими ресурсами [7,6,8].

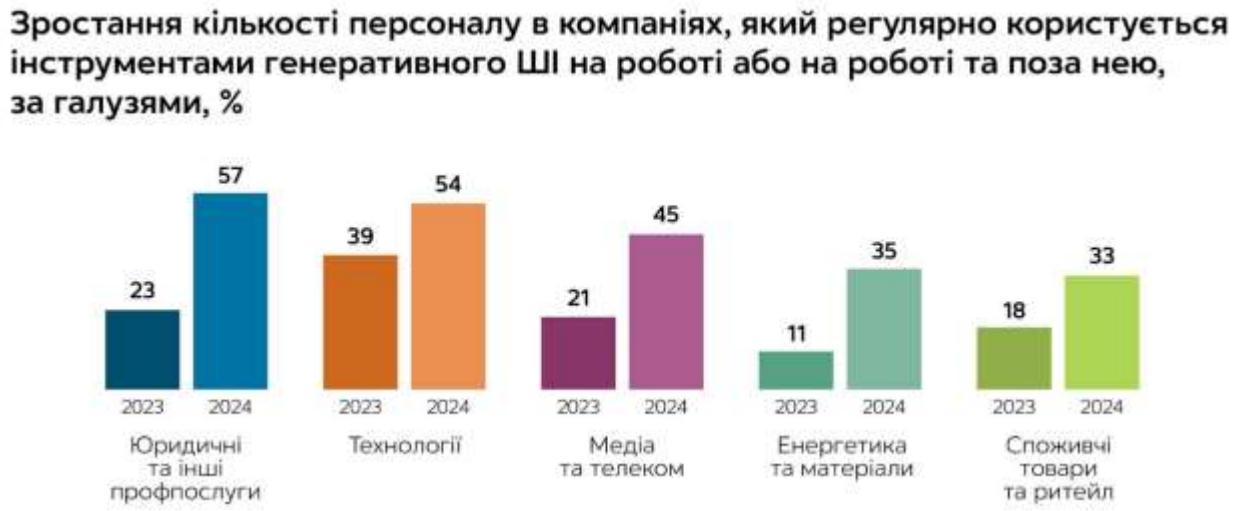

Рис. 4.3. Показники використання ШІ

Таким чином, майбутній розвиток ШІ у стратегічних іграх орієнтується на створення більш природних, гнучких і розумних супротивників, що робить геймплей глибшим і цікавішим для гравця [18].

4.4. Прогноз подальшого розвитку ШІ у стратегічних іграх

Штучний інтелект у стратегічних іграх займає ключову роль, визначаючи рівень складності гри та взаємодії з гравцем. Протягом останніх десятиліть ШІ значно удосконалився, а його можливості в області створення адаптивних та реалістичних супротивників стали більш очевидними. З кожним роком алгоритми ШІ стають більш гнучкими, здатними враховувати широкий спектр змінних та оптимізувати стратегії в реальному часі [9,8,6].

Одним з найперспективніших напрямків розвитку є інтеграція нейронних мереж для створення адаптивних супротивників, які здатні на базі досвіду та змін у грі коригувати свою стратегію. Це дозволить не лише змінювати поведінку в залежності від стилю гри гравця, але й створювати абсолютно нові стратегії в реальному часі, що значно ускладнює гру та робить її більш інтерактивною. Сучасні глибокі нейронні мережі дозволяють розробляти алгоритми, які вчяться з ігрового досвіду, тим самим ускладнюючи передбачуваність дій супротивника. Наприклад, у

майбутньому стратегічні ігри зможуть генерувати унікальні тактичні стратегії кожен раз, коли гравець вибирає певну лінію поведінки [7,6,9].

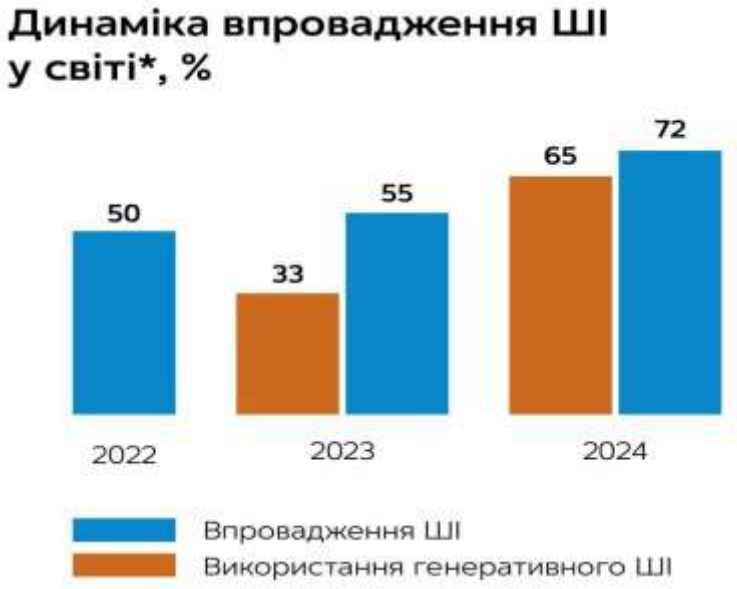

Рис. 4.4. ШІ у світі

Найбільшим досягненням може стати створення емоційного інтелекту в ігрових персонажах. Це дасть змогу створювати супротивників, які будуть не тільки підлаштовувати свою стратегію під гравця, але й мати емоційну реакцію на дії гравця. Наприклад, персонажі можуть показувати страх перед сильним супротивником, радість від перемоги або злість при програші, що може змінювати їхні дії та поведінку. Це підвищить рівень емоційного залучення та реалістичності гри [6,8,9].

Рис. 4.5. Емоції та поведінка персонажів у грі

Підкріплене навчання (Reinforcement Learning) є ключовим методом, який дозволяє створювати агентів, здатних вчитися на власному досвіді. Вони взаємодіють з ігровим світом, оцінюють результати своїх дій і коригують свою стратегію на основі отриманих винагород чи покарань. Цей метод активно застосовується у стратегічних іграх для покращення поведінки супротивників, роблячи їх більш гнучкими та адаптивними. В майбутньому можна очікувати, що методи підкріпленого навчання будуть розвиватися, і ШІ зможе ефективніше коригувати свої стратегії під конкретні дії гравця, навіть у складних ситуаціях [6,9,17].

Окрім нейронних мереж і підкріпленого навчання, розвиток гібридних моделей ШІ стане важливим етапом. Це методи, що комбінують традиційні алгоритми з новими підходами машинного навчання. Використання таких моделей дозволить створювати більш комплексні та реалістичні стратегії, що враховують не лише статичні змінні (наприклад, кількість ресурсів), але й динамічні (такі як психологічний стан персонажа, емоції та ситуаційні фактори). Ці стратегії забезпечать більш реалістичну взаємодію між персонажами, що сприятиме кращому зануренню гравця в гру та підвищенню її складності [6,7,9,17].

У майбутньому технології оптимізації ресурсів та розподілених обчислень допоможуть значно покращити ефективність ШІ в умовах обмежених ресурсів, таких як мобільні пристрої чи віртуальні реальності. Створення ігор з використанням потужних обчислювальних можливостей для більш складного ШІ буде особливо важливим для забезпечення продуктивності в реальному часі без втрати якості. Розвиток інтеграції з новими платформами, такими як мобільні ігри, віртуальна реальність (VR) та доповнена реальність (AR), дасть змогу використовувати ШІ для створення більш інтерактивних і захоплюючих ігрових середовищ [6,13,17].

Одним із найперспективніших напрямків розвитку стане вдосконалення соціального інтелекту ШІ, що дозволить створювати персонажів, здатних до глибших соціальних взаємодій, враховуючи

психологічні та емоційні аспекти. Це може включати адаптацію до командної гри, де ШІ агент може взаємодіяти з іншими персонажами, обираючи найефективніші стратегії в колективі. Розвиток таких алгоритмів відкриває нові можливості для створення реалістичних ігор, де рішення персонажів можуть мати соціальну і психологічну складову, що підвищує рівень залученості гравця [6,9,17].

Майбутнє ШІ у стратегічних іграх, без сумніву, буде тісно пов'язане з розвитком новітніх технологій, таких як 5G, обчислення на основі квантових комп'ютерів та обробка великих даних. Ці технології дозволять створювати ігри з більш реалістичною та інтелектуальною поведінкою персонажів, а також з покращеними можливостями для адаптації до потреб гравця. Завдяки цим досягненням, ігри з ШІ стануть ще більш динамічними, персоналізованими і глибокими. Це дозволить не лише створювати більш захоплюючі світи, але й надавати кожному гравцю унікальний досвід, що постійно змінюється відповідно до його дій і рішень [6,9,17].

4.5. Перспективи розвитку теми та практична значущість

Розглянута в роботі стратегічна гра з використанням елементів штучного інтелекту на Python є кроком у реалізації ідей в контексті інтеграції ШІ в ігрову індустрію. Це дає змогу виділити кілька напрямків для подальшого розвитку проекту:

1. Удосконалення моделей ШІ: одним із наступних кроків є використання більш складних моделей штучного інтелекту, таких як нейронні мережі, для розробки ворогів, здатних адаптувати свою стратегію залежно від дій гравця. Для цього може бути корисним застосування алгоритмів навчання з підкріпленням, що дозволяє створювати гнучкіші та більш динамічні стратегії ШІ [6,9,17].

2. Балансування ігрових механік: важливим етапом є коригування складності гри, налаштування економіки, керування ресурсами та інших

механік. Це дозволить зробити гру більш цікавою і стимулюючою для гравців, а також забезпечить можливість адаптації до різних типів гравців [10].

3. Використання сучасних технологій: застосування більш потужних графічних рушіїв і бібліотек, таких як Unity або Unreal Engine, дозволить значно підвищити візуальну складову гри, поліпшити продуктивність і забезпечити підтримку 3D-графіки, що зробить гру більш привабливою і доступною для широкого кола користувачів [11].

4. Мультиплеєрні можливості: розширення гри можливістю для гри кількох користувачів одночасно, інтеграція в онлайнові сервіси та додавання кооперативних режимів дозволить значно збільшити її популярність, оскільки це сприяє створенню більш конкурентних і соціальних взаємодій між гравцями [12].

Рекомендації для розробників:

- використання бібліотеки машинного навчання (TensorFlow або PyTorch), для створення більш інтелектуальних систем, що здатні до самонавчання, що дозволяє створювати ворогів з адаптивними стратегіями [6];
- створення інтуїтивно зрозумілого інтерфейсу, який спрощує взаємодію з грою та дозволяє гравцеві повністю зануритись у процес. Це включає підтримку інтуїтивних меню, візуалізацію елементів гри та забезпечення зрозумілих підказок і порад [5];
- проводити регулярне тестування гри з реальними користувачами для збору відгуків. Це допоможе визначити аспекти, які потребують доопрацювання або зміни, а також оцінити, наскільки вдало побудовані ігрові механіки та рівень складності [8].

Розробка ігор за допомогою Python та інтеграція елементів ШІ є важливим етапом у вдосконаленні навичок програмування і розвитку ігор як індустрії. З практичного погляду, результати цієї роботи мають таку значущість:

1. розробка такого проекту на Python є корисною для початківців у програмуванні та розробці ігор, адже вона дозволяє ознайомитися з основами програмування, концепцією ШІ та різними бібліотеками Python для створення ігор [10];

2. можливість для впровадження в реальні проекти: ідеї, які були реалізовані в цій роботі, можуть стати основою для комерційних проектів, навчальних платформ або тренажерів, що використовують інтерактивні елементи для навчання або розвитку [9].

Завдяки виконаній роботі був розроблений базовий прототип стратегічної гри з елементами штучного інтелекту, що реалізує основні принципи інтеграції ШІ в ігрові механіки. Гра демонструє базові функціональні можливості, зокрема боротьбу з адаптивним ворогом, управління ресурсами та використання магічних заклинань.

Поряд з цим, було реалізовано такі аспекти, як структура коду, яка дозволяє гнучко розширювати гру, додаючи нові елементи, персонажів або заклинання. Також важливим досягненням є реалізація ефективного контролю складності гри та її налаштувань.

Незважаючи на досягнення в розробці, є кілька обмежень:

- технічні обмеження: оскільки проект реалізований на Python, продуктивність гри та її графіка є обмеженими у порівнянні з іграми, створеними на інших більш потужних ігрових рушіях, таких як Unity або Unreal Engine [3];
- простота ШІ: алгоритми штучного інтелекту, що використовуються у грі, є базовими та не дозволяють створити більш складні стратегії ворогів. Можна покращити адаптивність ШІ, застосовуючи складніші методи [6].

Проте, подолання цих обмежень в майбутньому можливо за допомогою застосування більш потужних технологій, оптимізації алгоритмів і вдосконалення архітектури гри.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи була розроблена стратегічна гра на Python з елементами штучного інтелекту. Основним завданням роботи було створення основної механіки гри, що включає використання магічних заклинань, управління персонажами та інтеграцію штучного інтелекту для ворогів. Проект також передбачав створення інтерфейсу, що дозволяє гравцеві взаємодіяти з грою в режимі реального часу, а також забезпечує елементи стратегічного мислення, планування та управління ресурсами.

Розробка гри засвідчила ефективність використання мови програмування Python для створення простих, але гнучких ігор з елементами ШІ. Python, завдяки своїй простоті та великий кількості бібліотек, є чудовим вибором для створення прототипів та навчальних проектів, що мають на меті продемонструвати принципи інтеграції ШІ в ігри. Бібліотеки, такі як random для генерації випадкових подій і time для контролю часу, дозволяють розробникам зосередитися на розробці ігрової логіки та механік, а не на технічних аспектах програмування.

Ключовим результатом дослідження є створення системи для інтерактивної гри, яка може бути легко адаптована для складніших варіантів ігор. Наприклад, можна розширити гру за рахунок додавання більш складних тактик для штучного інтелекту ворога, що зробить гру більш захопливою та непередбачуваною для гравця. Це є важливим кроком у розвитку ігрової індустрії, де ШІ не лише надає реалістичність грі, але й може стати невід'ємною частиною інтерактивного процесу.

У рамках цієї роботи розглянуто основні аспекти створення стратегічних ігор з ШІ, включаючи психологічні чинники, що впливають на досвід гравців. Стратегічні ігри зазвичай включають розв’язання складних задач, що стимулює когнітивні функції, і тому вони мають значний вплив на мислення та аналітичні здібності гравців. У результаті проведеного

дослідження було виявлено, що важливими аспектами для успіху гри є здатність розвивати стратегії, керувати ресурсами і реагувати на зміни у поведінці ворогів, що вимагають від гравця адаптації та швидких рішень.

Також було розглянуто технічні та методологічні обмеження, що можуть виникнути при розробці подібних ігор, зокрема у частині складності створення реалістичних алгоритмів ШІ та забезпечення балансування ігрового процесу. Однак, сучасні технології машинного навчання та штучного інтелекту можуть допомогти значно поліпшити ці аспекти, що дозволить створювати більш динамічні та адаптивні ігри.

Прогнозуючи подальший розвиток цієї теми, можна відзначити, що з часом ігри з елементами ШІ стануть більш складними та інтерактивними. Розвиток технологій, таких як глибоке навчання і нейронні мережі, дозволить створювати ще більш інтелектуальні системи, які зможуть адаптуватися до стилю гри гравця, що підвищить рівень занурення та інтересу до гри.

З практичної точки зору, розробка стратегічних ігор з ШІ може стати корисним інструментом для навчання програмуванню, розумінню основ ШІ та створенню інтерактивних додатків. Це також відкриває можливості для створення комерційних проектів, які можуть бути використані в різних галузях, таких як освіта, розваги та тренування в управлінні ресурсами та стратегіями.

Таким чином, робота не лише продемонструвала потенціал Python як інструмента для створення ігор з елементами штучного інтелекту, а й показала важливість психологічних аспектів у проектуванні ігрових механік. У результаті, стратегічні ігри можуть стати не тільки цікавими, але й ефективними інструментами для розвитку аналітичних та когнітивних навичок у гравців. Надалі, для підвищення якості гри та її адаптації до змінюваних вимог користувачів, потрібно продовжити роботу над удосконаленням штучного інтелекту, графічних елементів і взаємодії з користувачем, що дозволить створити ще більш вражаючі та привабливі продукти на основі описаних принципів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Ролінг Р. Основи теорії ігор. — К.: Наукова думка, 2019.

2. Rabin, S. Game AI Pro: Collected Wisdom of Game AI Professionals. CRC Press, 2013.

3. Schell, J. The Art of Game Design: A Book of Lenses. CRC Press, 2008.

4. Adams, E., Rollings, A. Fundamentals of Game Design. New Riders, 2014.

5. Андрєєв А.В. Проектування стратегічних ігор. – Харків: ХНУРЕ, 2020.

6. Russell, S., Norvig, P. Artificial Intelligence: A Modern Approach. Prentice Hall, 2010.

7. Millington, I., Funge, J. Artificial Intelligence for Games. CRC Press, 2009.

8. Сидоров В.П. Штучний інтелект у комп’ютерних іграх. — К.: КНЕУ, 2021.

9. Yannakakis, G. N., Togelius, J. Artificial Intelligence and Games. Springer, 2018.

10. Sweigart, A. Making Games with Python & Pygame. Invent with Python, 2012.

11. Dawson, M. Python Programming for the Absolute Beginner. Cengage Learning, 2010.

12. McGugan, W. Beginning Game Development with Python and Pygame. Apress, 2007.

13. Panda3D Documentation [Електронний ресурс]. – Режим доступу: https://www.panda3d.org/manual/

14. Arcade Library [Електронний ресурс]. – Режим доступу: https://api.arcade.academy/en/latest/

15. Pygame Docs [Електронний ресурс]. – Режим доступу: https://www.pygame.org/docs/

16. Newzoo Global Games Market Report, 2023

17. Zhang, K., Deep Reinforcement Learning for Games, ACM Digital Library, 2022

18. Чеботарьов А.М. Інтелектуальні системи в геймдизайні. — Одеса: ОНУ, 2021.

19. Mobile Legends: Bang Bang [Електронний ресурс]. – Режим доступу: https://www.mobilelegends.com/