МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра Інформаційних технологій

Кваліфікаційна робота магістра

на тему: Дослідження та оптимізація впровадження штучного інтелекту в ігрових застосунках

Виконала студентка групи МІС-22зф
спеціальності 122 Комп'ютерні науки
Лаврик Анна Сергіївна

Керівник д.техн.наук, професор
Казакова Надія Феліксівна

Рецензент д.т.н., професор
Мещеряков Володимир Іванович

Одеса 2023

АНОТАЦІЯ

Дослідження та оптимізація впровадження штучного інтелекту в ігрових застосунках є актуальною проблемою в сучасному інформаційному світі. Ця дипломна робота присвячена детальному аналізу та оптимізації застосування штучного інтелекту в стратегічних іграх реального часу. У вступі надано загальний огляд теми та визначено основні цілі дослідження.

Об’єктом дослідження є визначення впливу впровадження штучного інтелекту в ігрові застосунки.

Ціль дослідження та оптимізації впровадження штучного інтелекту в ігрових застосунках полягає в розвиненні та вдосконаленні ігрової індустрії, зокрема, стратегічних ігор реального часу, за допомогою передових технологій штучного інтелекту.

Дослідження ґрунтується на використанні аналізу літературних джерел, експериментальних досліджень, моделювання і реалізації ігрових систем з штучним інтелектом.

У першому розділі "Аналіз існуючих стратегічних ігор реального часу" починається з класифікації стратегічних ігор реального часу, де визначаються основні види таких ігор та їх особливості. Далі розглядається можливість використання нейронних мереж для покращення ігрового процесу, а також застосування елементів штучного інтелекту в цьому контексті. Здійснюється аналіз і визначення впливу цих технологій на стратегічні ігри реального часу. Розділ завершується висновками, які відображають основні висновки із проведеного аналізу.

У другому розділі "Модель та метод покращення стратегічної гри реального часу з нейронною мережею з елементами штучного інтелекту" включає в себе побудову моделі стратегічної гри реального часу, що базується на нейронній мережі з елементами штучного інтелекту. Визначаються завдання, які ця мережа повинна вирішувати під час гри, а також методи для покращення ігрового процесу за допомогою штучного інтелекту.

Розглядаються основні аспекти удосконалення характеристик гри з використанням нейронної мережі. Розділ завершується висновками, які підсумовують результати роботи над моделлю та методами оптимізації.

Розділ три "Розробка інфраструктури гри з нейронною мережею з елементами штучного інтелекту" охоплює створення середовища для проведення моделювання, тестування та валідації гри. Розглядається створення серверного компоненту, необхідного для взаємодії з ігрою. В цьому розділі роботи дається висновок, який підсумовує розробку інфраструктури та підготовляє до наступного етапу − моделювання та тестування.

У розділі чотири "Моделювання та тестування стратегічної гри реального часу за допомогою тактичної системи штучного інтелекту і результати" присвячений конкретному моделюванню та тестуванню гри з використанням тактичної системи штучного інтелекту. Розглядаються результати моделювання та тестування, аналізуються ефективність і відповідність задачам. Розділ завершується висновками, які резюмують результати дослідження та надають рекомендації щодо подальших можливостей вдосконалення ігрового досвіду.

Ця дипломна робота є важливим внеском у розвиток ігрової індустрії та застосування штучного інтелекту в галузі стратегічних ігор реального часу. Вона дозволяє краще зрозуміти можливості цих технологій та їх вплив на ігровий процес, а також надає практичні рекомендації для їх впровадження.

Ключові слова. Штучний інтелект, стратегічні ігри реального часу, моделювання гри, ігрова індустрія.

Структура роботи. Робота складається з вступу, чотирьох розділів, висновків до розділів, висновку, переліку джерел посилань. Повний обсяг роботи становить 73 сторінки, містить 10 рисунків, 1 таблицю, 20 джерел.

ABSTRACT

Research and optimization of the implementation of artificial intelligence in game applications is an urgent problem in the modern information world. This thesis is devoted to a detailed analysis and optimization of the use of artificial intelligence in real-time strategy games. The introduction provides a general overview of the topic and defines the main objectives of the study.

The object of research is the process of introducing artificial intelligence into game applications.

The goal of research and optimization of the implementation of artificial intelligence in gaming applications is to develop and improve the gaming industry, in particular, real-time strategy games, with the help of advanced artificial intelligence technologies.

The research is based on the use of analysis of literary sources, experimental studies, modeling and implementation of game systems with artificial intelligence.

The first chapter "Analysis of existing real-time strategic games" begins with the classification of real-time strategic games, where the main types of such games and their features are determined. Next, we consider the possibility of using neural networks to improve the gameplay, as well as the application of elements of artificial intelligence in this context. Analysis and determination of the impact of these technologies on real-time strategic games is carried out. The chapter ends with conclusions that reflect the main conclusions from the conducted analysis.

In the second chapter, "A model and method of improving a real-time strategy game with a neural network with elements of artificial intelligence" includes the construction of a model of a real-time strategy game based on a neural network with elements of artificial intelligence. The tasks that this network must solve during the game are defined, as well as methods to improve the gameplay using artificial intelligence. The main aspects of improving the characteristics of the game using a neural network are considered. The chapter ends with conclusions that summarize the results of work on the model and optimization methods.

Chapter Three, "Developing a Neural Network Game Infrastructure with Artificial Intelligence Elements" covers creating an environment for simulation, testing, and game validation. The creation of a server component necessary for interaction with the game is under consideration. In this section of the work, a conclusion is given that summarizes the development of the infrastructure and prepares for the next stage - modeling and testing.

Chapter four, "Real-time strategic game simulation and testing using a tactical artificial intelligence system and results" is devoted to concrete simulation and testing of a game using a tactical artificial intelligence system. The results of modeling and testing are considered, the effectiveness and compliance with the tasks are analyzed. The chapter concludes with conclusions that summarize the research findings and provide recommendations for further opportunities to improve the gaming experience.

This thesis is an important contribution to the development of the gaming industry and the application of artificial intelligence in the field of real-time strategy games. It allows you to better understand the capabilities of these technologies and their impact on the gameplay, as well as provides practical recommendations for their implementation.

Keywords. Artificial intelligence, real-time strategy games, game simulation, gaming industry.

Structure of work. The work consists of an introduction, four chapters, conclusions to the chapters, a conclusion, a list of reference sources. The full volume of work is 73 pages, contains 10 figures, 1 table, 20 sources.

ЗМІСТ

ВСТУП 9
1 АНАЛІЗ ІСНУЮЧИХ СТРАТЕГІЧНИХ ІГОР РЕАЛЬНОГО ЧАСУ 11
1.1 Класифікація стратегічних ігор реального часу 11
1.2 Застосування нейронних мереж для активізації стратегічних ігор 18
1.3 Застосування елементів штучного інтелекту для активізації стратегічних ігор 25
Висновки до 1 розділу 31
2 МОДЕЛЬ ТА МЕТОД ПОКРАШЕННЯ СТРАТЕГІЧНОЇ ГРИ РЕАЛЬНОГО ЧАСУ З НЕЙРОННОЮ МЕРЕЖЕЮ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ 33
2.1 Побудова моделі стратегічної гри реального часу на основі нейронної мережі зі штучним інтелектом. 33
2.2 Задачі в межах гри, які повинні вирішувати нейронні мережі з елементами штучного інтелекта 39
2.3 Удосконалення характеристик гри 47
Висновок до 2 розділу 49
3 РОЗРОБКА ІНФРАСТРУКТУРИ ГРИ З НЕЙРОННОЮ МЕРЕЖЕЮ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ 50
3.1 Створення середовища для проведення моделювання, тестування і валідації гри 50
3.2 Створення серверного компоненту 60
Висновок до розділу 3 63
4 МОДЕЛЮВАННЯ ТА ТЕСТУВАННЯ СТРАТЕГІЧНОЇ ГРИ РЕАЛЬНОГО ЧАСУ ЗА ДОПОМОГОЮ ТАКТИЧНОЇ СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ І РЕЗУЛЬТАТИ 65
4.1 Моделювання та тестування тактичної системи штучного інтелекту 65
4.2 Результати моделювання та тестування 66
Висновки до розділу 4 69
ВИСНОВКИ 70
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 72

ВСТУП

Актуальність дослідження полягає у тому, що штучний інтелект в іграх постійно еволюціонує і має великий потенціал для розвитку. Нові технології та алгоритми дозволяють створювати все більш інтелектуальних ігрових персонажів та системи, які здатні приймати виважені рішення та надавати змагальний досвід.

Оптимізація впровадження штучного інтелекту в ігрових застосунках є необхідним завданням для досягнення оптимальної продуктивності та ефективності. Розробники ігор постійно прагнуть до досягнення більшої швидкості, точності та реалістичності впроваджених інтелектуальних систем, що робить цю тему надзвичайно важливою.

Дослідження впровадження штучного інтелекту в ігрових застосунках також може приносити значний внесок у галузь. Оптимізовані алгоритми та технології можуть допомогти розробникам створити ігри з більш інноваційним та глибшим геймплеєм, що забезпечить конкурентну перевагу на ринку.

Враховуючи всі ці чинники, можна визначити, що дослідження та оптимізація впровадження штучного інтелекту в ігрових застосунках є надзвичайно актуальною темою. Вона відкриває шлях до створення більш реалістичних та інтелектуально-складних ігор, які забезпечують цікавий та задовільний досвід гравцю.

Об’єктом дослідження є процес впровадження та визначення впливу штучного інтелекту в ігрових застосунках.

Предметом дослідження є аналіз та оптимізація впровадження штучного інтелекту в стратегічних іграх реального часу.

Метою дослідження є розробка моделей та методів для покращення інтелектуального компонента ігор реального часу за допомогою штучного інтелекту.

Ґрунтуючись на меті дослідження, доцільно виокремити такі **завдання**:

1. Розробити класифікацію стратегічних ігор реального часу.
2. Визначити задачі, які повинні вирішувати нейронні мережі з елементами штучного інтелекту.
3. Розробити методи для удосконалення характеристик гри на основі нейронної мережі з елементами штучного інтелекту.
4. Створити середовище для проведення моделювання, тестування і валідації гри.

Методи дослідження. Дослідження ґрунтується на використанні аналізу літературних джерел, експериментальних досліджень, моделювання і реалізації ігрових систем з штучним інтелектом.

Теоретичне значення даної роботи полягає в розширенні теоретичних знань в галузі застосування штучного інтелекту в ігровій індустрії. Ця робота внесе важливий внесок у розуміння та розвиток принципів, які стоять за використанням штучного інтелекту в ігрових застосунках. Дослідження розглядає можливості застосування нейронних мереж та елементів штучного інтелекту для покращення геймплею стратегічних ігор реального часу.

Практичне значення даного дослідження полягає у тому, що воно може допомогти вдосконалити геймплей ігрових застосунків, розробити більш інтелектуальних та реалістичних ворогів, покращити систему штучного інтелекту, що використовується у грі, або розробити інноваційні методи використання штучного інтелекту у гральній індустрії.

Апробація результатів була проведена з метою визначення ефективності розробленої системи та її потенційного застосування у різних сферах.

1 АНАЛІЗ ІСНУЮЧИХ СТРАТЕГІЧНИХ ІГОР РЕАЛЬНОГО ЧАСУ

1.1 Класифікація стратегічних ігор реального часу

Стратегічні ігри реального часу, або RTS-ігри, є важливими компонентами сучасної геймінг-індустрії і розуміння їх основних принципів заслуговує на увагу дослідників. Вони викликають інтерес не лише серед широкого кола гравців, але й серед наукової спільноти.

Основною особливістю RTS-ігр є поєднання стратегічного та тактичного мислення, тут гравці мають можливість: побудувати й керувати власною базою, збирати ресурси, розвивати технології та формувати армію для завоювання ворожих територій. Вони вимагають від гравців не лише швидкого прийняття рішень, але й здатності аналізувати складні ситуації та передбачати можливі наслідки своїх дій.

Стратегічні ігри реального часу також надають можливість гравцям випробувати свої логічні та аналітичні навички: гравці повинні розв'язувати складні завдання, приймати рішення щодо розподілу ресурсів та керування військами тощо. Все це висуває вимоги до логічного мислення та майстерності стратегічного планування.

Значення дослідження RTS-ігор полягає в тому, що це може розширити наше розуміння стратегічних процесів та мислення загалом. Вивчення основних принципів цих ігор може допомогти в розробці нових моделей стратегічного мислення та доповнити існуючі теорії [1]

Крім того, дослідження RTS-ігор може бути корисним для розробників таких ігор. Вивчення вимог гравців та їх поведінки може допомогти створити більш ефективні та привабливі ігрові середовища. Розуміння та аналіз стратегічних принципів таких ігор може сприяти досягненню більшої реалістичності та глибини геймплею.

Сучасний світ відеоігор пропонує широкий спектр жанрів, і серед них особливе місце займають стратегічні ігри в режимі реального часу (RTS). Цей жанр розділяється на три основні рівні складності: прості RTS, середні RTS та складні RTS. Кожен з цих рівнів має свої унікальні особливості і приваблює гравців з різними рівнями досвіду.

Реалтайм стратегії (Real-Time Strategy, RTS) є одним з популярних жанрів в ігровій індустрії, який привертає увагу гравців різних рівнів досвіду. Проте, важливим етапом у знайомстві з цим жанром є вибір простих RTS-ігор, спрямованих на новачків та гравців, які лише починають ознайомлюватися з ним. Ці ігри мають свої особливості та характеристики, спрямовані на ефективне введення гравця в основні аспекти та механіки цього жанру.

Однією з ключових характеристик простих RTS-ігор є обмежена кількість одиниць і ресурсів, доступних гравцю на початку гри. Це робить гру менш складною для новачків, оскільки вони можуть зосередитися на обмірковуванні дій, які потрібно виконувати в обмежених умовах. Однак такий обмежений ресурсний старт також створює можливість для вивчення основних аспектів гри, таких як збирання ресурсів, будівництво бази та формування армії [2]

Збирання ресурсів є однією з найважливіших механік у багатьох RTS-іграх. Гравцеві необхідно ефективно управляти видобутком ресурсів, щоб мати можливість будувати нові споруди та займатись найманням військових одиниць. У простих RTS-іграх ресурсні джерела можуть бути обмеженими, тому гравцеві слід обирати, на що витрачати свої ресурси, і стратегічно планувати розвиток.

Будівництво бази є ще однією важливою частиною RTS-ігор. Гравцям доводиться визначити оптимальність розташування своїх споруд для максимальної ефективності. Це включає в себе вибір місця для видобутку ресурсів, розташування оборонних структур і споруд для тренування військових одиниць. Прості RTS-ігри надають гравцям можливість вчитися цим аспектам гри поступово і без поспішання.

Формування армії є ще однією ключовою механікою. Гравцям слід вирішувати, які військові одиниці створювати, і як їх правильно використовувати в бою. В простих RTS-іграх, де кількість доступних одиниць може бути обмеженою, важливо знати, які одиниці найефективніші в конкретній ситуації.

Усі ці аспекти спільно допомагають новачкам освоювати основи RTS-ігор. Гравці в простих RTS-іграх можуть пройти гру, не глибоко аналізуючи стратегію, і вони часто служать відмінним вступом до складніших RTS-ігор, де ресурси та стратегія вимагають більш глибокого розуміння.

Середні RTS-ігри дійсно представляють інший рівень складності порівняно з простими іграми цього ж жанру. Вони надають гравцям більше ресурсів та бойових одиниць на початку гри, що робить їх складнішими та вимагає більш глибокого стратегічного аналізу та планування.

Основною відмінністю середніх RTS-ігор є більший обсяг ресурсів. Гравцям доводиться керувати більшими обсягами ресурсів і вирішувати, як їх ефективно використовувати. Це включає в себе не тільки збирання ресурсів, але і ефективні їх витрати на розвиток бази, покращення технологій та створення різних типів військових одиниць [3]

У середніх RTS-іграх також важливо збалансувати розвиток бази та ведення бойових дій. Гравцям доводиться вирішувати, яку кількість ресурсів витрачати на розвиток, а яку на створення військових одиниць. Неправильне рішення може призвести до того, що гравець буде недостатньо підготовлений до ворожого нападу або, навпаки, витратить багато ресурсів на армію і база залишиться відстороненою.

Один з ключових аспектів середніх RTS-ігор − це розробка більш складних стратегій. Гравцям доводиться враховувати можливі варіанти ворожих дій та реагувати на них. Вони повинні вибудовувати свою стратегію з урахуванням розвитку військових та економічних аспектів гри. Це вимагає глибокого аналізу, планування та пристосування під конкретну ситуацію на полі бою.

Складні RTS-ігри дійсно є найвищим рівнем складності у світі реалтайм стратегій. Вони вимагають від гравців високого рівня стратегічного мислення та майстерності, оскільки представляють глибший геймплей та велику кількість аспектів керування, які вимагають уваги і ретельного планування.

Однією з ключових характеристик складних RTS-ігор є велика кількість аспектів керування. Гравцям доводиться управляти не тільки армією, але і ресурсами, науковим дослідженням, дипломатією, економікою та багатьма іншими факторами. Це створює багатогранну гру, де гравці повинні збалансувати різні аспекти і розвивати стратегії, які відповідають конкретній ситуації.

Однак однією з найважливіших характеристик складних RTS-ігор є високий рівень аналізу та планування. Гравці повинні ретельно розглядати кожен аспект гри, включаючи стратегії супротивника, і розробляти плани дій, які враховують всі можливі варіанти подій. Вони також повинні вести ретельний облік ресурсів та військових одиниць, щоб забезпечити ефективну стратегію.

Складні RTS-ігри можуть вимагати багато годин практики, щоб досягти високого рівня майстерності. Гравці повинні навчатися на помилках і постійно вдосконалювати свої навички, щоб мати змогу конкурувати на вищому рівні. Однак саме ця складність і вимоги роблять складні RTS-ігри настільки захоплюючими і викликають в гравців бажання досягати чималих успіхів у цьому захопливому жанрі [4]

Історичні RTS-ігри дозволяють гравцям поглибитися в реалістичний світ минулого і відчути себе частиною великих подій та конфліктів. Автори таких ігор намагаються якомога точніше відтворити історичну атмосферу, події та персонажів.

Age of Empires − це серія ігор, що охоплює різні історичні епохи, починаючи зі стародавнього світу та закінчуючи ранньомодерним періодом. Гравці можуть керувати своєю цивілізацією, розбудовувати міста, займатися технологічним розвитком та приймати участь у військових конфліктах.

Серія Total War, у свою чергу, зосереджена на реалістичній військовій стратегії. Гравці можуть керувати великими арміями, організовувати битви та політичні інтриги, відтворюючи важливі історичні події і конфлікти. Деякі ігри з серії, наприклад Rome: Total War або Medieval II: Total War, розташовані в конкретних історичних періодах.

«Cossacks 3» – це гра в жанрі стратегії в реальному часі від українського розробника GSC Game World. Вона є перезапуском оригінальної гри «Cossacks», яка була випущена в 2001 році.

Гра включає велику кількість юнітів, велику базу різноманітних будівель, розширену систему економіки та розгорнуті битви. “Cossacks 3” є відмінним вибором для любителів стратегічних ігор, особливо тих, хто цінує історичну точність та реалізм.

Ці ігри не тільки надають можливість гравцям зануритися в історичну тематику, але також дозволяють розвивати тактичні та стратегічні навички, аналізувати ситуацію та приймати рішення, які впливають на подальший розвиток гри. Історичні RTS-ігри є популярним жанром серед істориків, геймерів та тих, хто цікавиться минулим.

Фантастичні RTS-ігри дозволяють гравцям зануритися в уявний світ, де присутні різноманітні фантастичні раси, технології та магія. Граючи у такі ігри, гравці отримують можливість керувати фракціями з унікальними особливостями і властивостями.

StarCraft − це одна із найвідоміших фантастичних RTS-ігор. Вона розповідає про війну між трьома різними расами - людьми, зергами і протосами. Кожна з цих рас має свою унікальну структуру, технології та військові одиниці, що вносить різноманітність в гру. Гравці можуть розбудовувати бази, займатися економікою, створювати армії та приймати участь у великих сутичках.

Warcraft - ще одна популярна фантастична RTS-ігра, зосереджена на баталіях між різними фракціями із фантастичних світів. Warcraft дозволяє гравцям керувати різними расами, такими як люди, орки, ельфи та інші.

Захоплюючі сюжети та епічні тривалість гри роблять Warcraft популярною серед шанувальників жанру.

У фантастичних RTS-іграх гравцям надається можливість експериментувати з різними стратегіями, розвивати технологічні потенціали та магічні здібності своїх армій, а також взяти участь в епічних битвах.

Сучасні RTS-ігри пропонують гравцям інтерактивний досвід в уявному сучасному світі з сучасною технікою, зброєю та геополітичними відносинами. Гравцям необхідно взяти на себе керівництво військовими силами, займатись розвитком технологічних досягнень та брати участь у збройних конфліктах.

Серія Command & Conquer − одна з найвідоміших представників такої сучасної RTS-ігрової тематики. Гравці можуть вибрати сторону конфлікту і керувати основними арміями та технологіями цих фракцій. Гра пропонує велику кількість одиниць, будівель, техніки та ресурсів для використання, що дозволяє гравцям розвивати свою стратегію в боротьбі за ресурси та владу.

У сучасних RTS-іграх гравцям надаються різноманітні можливості для розвитку своєї армії, з урахуванням тактичних особливостей та стратегічного планування. Важливою складовою таких ігор є геополітичні аспекти, включно з дипломатією, зібранням розвідданних, створенням союзів та проведенням військових операцій [5]

Ці сучасні RTS-ігри надають гравцям можливість побачити сучасні конфлікти та випробувати свої стратегічні навички в умовах сучасного світу. Як і в інших жанрах RTS, головний акцент робиться на прийнятті стратегічних рішень, координації військових дій та розвитку військово-економічних потенціалів. Сучасні RTS-ігри привертають увагу тих, хто цікавиться сучасною військовою стратегією та випробуванням свого стратегічного мислення.

Сеттинг гри в стратегічних іграх RTS грає важливу роль у створенні неповторного ігрового досвіду. Історичні RTS дозволяють відчути дух минулих епох, фантастичні RTS вносять у світ фантазії та магії, а сучасні RTS виводять гравців у реальний світ сучасних конфліктів та технологій.

Ігрові механіки базового будівництва та тактичні механіки є важливими елементами сучасних відеоігор, які вимагають від гравців розвитку різних навичок, включаючи стратегічне мислення, аналіз та прийняття рішень.

Базові будівельні ігрові механіки є одними з найпоширеніших ігрових елементів і використовуються в різних жанрах ігор. Гравці в цих іграх мають можливість будувати та розвивати свою власну базу, збирати ресурси та формувати армію. Такі механіки вимагають від гравця стратегічного мислення, планування та ефективного управління ресурсами. Це дозволяє гравцям розвивати вміння приймати рішення та керувати обмеженими ресурсами.

До прикладів ігор з базовими будівельними механіками можна віднести такі популярні та впливові твори, як "Civilization" від Sid Meier, де гравці вибудовують свою імперію, здатну витримати випробування часом, та "Starcraft" від Blizzard Entertainment, де вони керують фракціями і розвивають свою воєнну базу.

Тактичні ігрові механіки спрямовані на розвиток більш складних бойових тактик та стратегій. У таких іграх гравцям доводиться розглядати велику кількість деталей та приймати швидкі та обмірковані рішення в бойових ситуаціях. Це включає в себе визначення позицій, обрання оптимальних атак та оборонних стратегій.

Однією з відомих ігор, що використовують тактичні механіки, є "XCOM: Enemy Unknown" від Firaxis Games. У цій грі гравцям потрібно керувати командою бійців та розробляти тактику в боях з прибульцями та розвивати базу проєкту.

Стратегічні ігри реального часу є одним з найпопулярніших жанрів комп'ютерних ігор, які спроектовані для змагання на стратегічному рівні. Особливістю цього жанру є те, що геймплей відбувається в режимі реального часу, без перерв або пауз між ходами. Гравець повинен негайно приймати рішення, адаптуватися до швидкозмінних обставин та вчасно реагувати на дії противників.

Одна з ключових особливостей стратегічних ігор реального часу - це керування ресурсами. Гравець повинен ефективно управляти різними ресурсами, такими як дерево, камінь, золото або нафта. Це вимагає стратегічного планування, так як гравцеві необхідно розробляти довгострокові та короткострокові стратегії для максимізації використання ресурсів та забезпечення ефективного розвитку своєї бази або країни.

Ще однією важливою особливістю стратегічних ігор реального часу є наявність множинних одиниць. Гравці керують великою кількістю військових одиниць, кожна з яких має свої унікальні характеристики та вміння. Відправка правильних одиниць у відповідний момент часу може бути вирішальним фактором у битвах та перемогах.

Стратегічні ігри реального часу також вимагають від гравців широкого спектру навичок, таких як швидке реагування, планування, прийняття рішень під час стресових ситуацій та багато іншого. Ці ігри сприяють розвитку таких навичок, як аналітичне мислення, стратегічне мислення, креативність та командна співпраця.

Загалом, стратегічні ігри реального часу відносяться до складних та захоплюючих ігрових жанрів, які вимагають від гравців високого рівня ігрової компетенції та стратегічних навичок. Вони сприяють розвитку різних когнітивних вмінь та можуть бути знадобитися в реальних ситуаціях, таких як управління бізнесом або стратегічне планування в армії.

1.2 Застосування нейронних мереж для активізації стратегічних ігор

Застосування нейронних мереж у галузі активізації стратегічних ігор є однією з актуальних та цікавих тем в сучасному інформаційному та розважальному просторі. Стратегічні ігри завжди викликали інтерес у гравців, оскільки вони вимагають від них розробки складних стратегій, аналізу ситуацій та прийняття важливих рішень. Використання нейронних мереж у цій

галузі може значно покращити якість геймплею, зробити ігри більш високорівневими та захоплюючими для гравців.

Однією з головних переваг використання нейронних мереж у стратегічних іграх є змога підвищити рівень інтелектуальної складності гри. Нейронні мережі − це алгоритми штучного інтелекту, які можуть навчатися та адаптуватися до нових ситуацій. Вони можуть аналізувати велику кількість даних та приймати рішення на основі цієї інформації.

При використанні нейронних мереж у стратегічних іграх, опоненти можуть бути більш інтелектуально вимогливими. Нейронні мережі можуть аналізувати історію гри, прогнозувати можливі ходи та реагувати на стратегії гравців. Це робить гру більш складною та непередбачуваною, що відкриває нові горизонти для розвитку стратегічних навичок гравців.

Підвищена інтелектуальна складність гри завжди привертає увагу гравців, в пошуках викликів і можливостей для розвитку своїх навичок. Використання нейронних мереж у стратегічних іграх може зробити гру більш цікавою і захопливою.

Крім того, важливо відзначити, що використання нейронних мереж у стратегічних іграх може призвести до створення враження, що гравці грають проти реальних, розумних опонентів, навіть якщо це комп'ютерні програми. Це може підвищити інтерес до гри та збільшити активність гравців.

У сучасному світі, коли стратегічні рішення стають все більш складними і вимагають глибокого аналізу та прогнозування, застосування нейронних мереж в стратегічних іграх відіграє важливу роль. Це відкриває нові можливості для розвитку гравців та підвищення рівня їх стратегічної компетенції. У цьому науковому дослідженні розглянуто важливість застосування нейронних мереж у стратегічних іграх та їхній вплив на симуляцію реальних стратегічних ситуацій [6]

Передусім, важливо відзначити, що нейронні мережі є потужним інструментом для моделювання та аналізу складних систем. Вони здатні аналізувати великі обсяги даних і виділяти з них закономірності, які не завжди

сприймаються людським розумом. У контексті стратегічних ігор, нейронні мережі можуть бути навчені аналізувати історію гри, рухи супротивників та інші параметри для прогнозування майбутніх подій.

Однією з важливих переваг застосування нейронних мереж є їхній здатність до навчання. Це означає, що вони можуть постійно покращувати свою продуктивність, адаптуючись до змінних умов гри. Гравці можуть використовувати навчені моделі для аналізу варіантів та вибору оптимальної стратегії в реальному часі. Це особливо корисно у стратегічних іграх, де велику роль відіграє аналіз імовірностей та оптимізація рішень.

Крім того, нейронні мережі можуть бути використані для симуляції реальних стратегічних ситуацій. Вони можуть відтворювати складні сценарії та динаміку подій у віртуальному середовищі. Це дозволяє гравцям практикувати та вдосконалювати свої навички прийняття стратегічних рішень, не ризикуючи реальними ресурсами.

Також, важливо враховувати, що нейронні мережі можуть аналізувати великий обсяг інформації про різні аспекти гри, включаючи інформацію про ресторани та готелі в Україні. Це може бути корисно при прийнятті стратегічних рішень, пов'язаних з господарською діяльністю в цій галузі.

Покращення якості гри та розваг для гравця є важливою метою в галузі розробки відеоігор. Це вимагає розуміння когнітивних здібностей гравців і використання високоякісних підходів у розробці гральних досвідів. В даній роботі буде розглянуто те, як нейронні мережі та інші інноваційні технології можуть сприяти досягненню цієї цілі.

По-перше, нейронні мережі можуть бути використані для створення реалістичних і швидких штучних інтелектів (ШІ) у гральних іграх. Завдяки навчанню на великому обсязі даних, ШІ можуть реагувати на дії гравців більш природним і реалістичним способом. Це покращує імерсію та реалізм грального досвіду, що, в свою чергу, задовольняє потреби гравців.

По-друге, нейронні мережі можуть бути використані для персоналізації гральних досвідів. Вони можуть аналізувати стиль гри кожного гравця, його

вподобання і навіть реакції на певні ситуації в грі. На основі цієї інформації гра може адаптуватися до конкретного гравця, надаючи йому якомога більше задоволення від проходження гри.

По-третє, використання інших інноваційних технологій, таких як віртуальна реальність (VR) та розширена реальність (AR), може значно покращити ігровий досвід. Ці технології дозволяють створювати імерсивні ігрові середовища, де гравці можуть взаємодіяти з ігровим світом більш активно. Наприклад, VR може зробити геймерські рухи і дії більш реалістичними, а AR може додавати елементи гри в реальний світ.

Ще однією цікавою можливістю є використання аналізу даних ігрового процесу для оптимізації грального досвіду. Нейронні мережі можуть аналізувати гравців і відстежувати їхні реакції на різні елементи гри. На основі цього аналізу розробники можуть внести зміни в гру, щоб зробити її більш захоплюючою.

Однією з головних задач застосування нейронних мереж в стратегічних іграх є навчання ігрових агентів, які можуть брати участь у грі на рівні штучного інтелекту. Ця задача включає в себе навчання нейронних мереж приймати стратегічні рішення, оптимізувати свою гру та адаптуватися до різних ситуацій.

Важливо зазначити, що стратегічні ігри, такі як шахи, або великі мультиплеєрні ігри, вимагають від гравців розуміння складних стратегій, прогнозування рухів супротивників і прийняття оптимальних рішень. Використання нейронних мереж дозволяє створити ігрових агентів, які можуть вчитися цьому складному мистецтву та постійно покращувати свою стратегічну ефективність [7]

Основною складністю в цій задачі є навчання нейронних мереж здійснювати вибори між різними стратегіями в реальному часі. Для досягнення цієї мети, використовуються методи підсиленого навчання, де агенти навчаються на основі нагород та покарань, які вони отримують після

кожного ходу. Навчання нейронних мереж таким чином допомагає їм розвивати оптимальні стратегії відповідно до умов гри.

Застосування нейронних мереж у навчанні агентів також вимагає розробки відповідних ігрових середовищ, де агенти можуть навчатися і вдосконалювати свої навички. Це може бути реалістичною симуляцією гри або великою кількістю ігрових даних з реальних ігор.

Моделювання грального середовища є складним та важливим аспектом багатьох наукових досліджень. Цей підхід дозволяє відтворити різні стратегічні аспекти реальних ситуацій та вивчати їх в контрольованому середовищі.

Моделювання грального середовища може включати в себе різні елементи, такі як ймовірність рішень гравців, взаємодію факторів, а також інші аспекти, що враховуються в певному гральному середовищі. Застосування математичних моделей дозволяє оцінювати різні стратегії та їх ефективність, а також розробляти оптимальні рішення в умовах обмежень та ризику.

Крім того, для моделювання грального середовища можуть використовуватися комп'ютерні програми та інструменти, які дозволяють створити імітацію реального грального процесу. Такі програми можуть відтворювати різні аспекти гри, такі як розташування гравців, генерація випадкових подій, реалістична анімація та інше. Вони дозволяють проводити експерименти, аналізувати рішення гравців та досліджувати різні стратегії без реальних витрат.

Стратегічні ігри реального часу можуть бути використані в моделюванні грального середовища для аналізу широкого спектру ситуацій. Вони дозволяють досліджувати вплив різноманітних факторів, таких як конкуренція, обмеження та ризики на стратегічні рішення гравців. Такі ігри є ефективним інструментом для вивчення та аналізу стратегічного планування та прийняття рішень в умовах невизначеності та конкуренції.

Оптимізація геймплею використовується для налаштування різних аспектів гри з метою забезпечення оптимального геймплею для гравців. Одним із способів оптимізації може бути використання нейронних мереж.

Нейронні мережі є потужним інструментом для аналізу та передбачення складних систем, таких як гра. Вони можуть використовуватися для аналізу поведінки гравців, аналізу даних гри, а також для розробки імітаційних моделей гри [8]

Одним зі способів використання нейронних мереж для оптимізації геймплею є балансування гри. Нейронна мережа може аналізувати дії гравців, вимірювати їх ефективність та рівень складності гри, а потім змінювати параметри гри, такі як сила ворогів, кількість ресурсів або швидкість гри, з метою створення збалансованішого ігрового досвіду.

Також, нейронні мережі можуть використовуватися для адаптації рівня складності гри до навичок гравців. Вони можуть аналізувати поведінку гравців, оцінювати їх рівень вмінь та навичок, і змінювати параметри гри, щоб забезпечити оптимальний рівень насолоди та виклику від гри.

Крім того, нейронні мережі можуть використовуватися для аналізу взаємодії гравців. Вони можуть вивчати стратегії та взаємодію різних гравців, враховувати їх рішення та прогнозувати вплив цих рішень на хід гри. Це може допомогти вирішити проблеми дисбалансу в грі та покращити взаємодію між гравцями.

Нейронні мережі є потужним інструментом для оптимізації геймплею, оскільки вони можуть аналізувати великі обсяги даних та враховувати складні взаємозв'язки в грі. Використання нейронних мереж дозволяє створити більш приємний та захоплюючий геймплей для гравців, а також забезпечити більш точне налаштування різних аспектів гри.

Використання нейронних мереж для покращення графічного рівня гри може мати кілька важливих переваг. Один з основних способів використання нейронних мереж − це генерація реалістичних текстур і моделей об'єктів у грі.

Нейронні мережі можуть бути навчені розпізнавати і аналізувати різні властивості об'єктів, такі як текстури, форми та освітлення. З цими знаннями вони можуть створити нові графічні елементи, які відповідають заданим параметрам [9]

Наприклад, нейронні мережі можуть автоматично створювати реалістичні текстури, які відображають різні матеріали, такі як дерево, метал або тканина. Вони можуть враховувати особливості кожного матеріалу, такі як його поверхнева структура, кольоровий тон та відблиск, і відтворювати їх у деталях.

Також нейронні мережі можуть бути використані для створення реалістичної анімації, яка додасть життя графічним об'єктам у грі. Вони можуть навчатися розпізнавати рухи та поведінку різних об'єктів і відтворювати їх з високою точністю. Наприклад, нейронна мережа може навчитися створювати реалістичний рух водяних хвиль або ж вітру на деревах.

В результаті використання нейронних мереж для покращення графічного рівня гри можна отримати дуже реалістичний візуальний ефект. Це дозволяє гравцям ще більше зануритися в світ гри і насолоджуватися його красою і деталями. Таке підвищення графічного рівня гри може зробити її більш привабливою для гравців і сприяти її популярності.

Отже, застосування нейронних мереж у стратегічних іграх – це важлива та перспективна область, яка приносить численні переваги. Вона сприяє створенню більш різноманітних та цікавих ігрових досвідів для гравців, допомагаючи розвивати їхні інтелектуальні навички. Завдяки використанню нейронних мереж можливо створити більш складні та реалістичні ігрові сценарії, а також реагувати на дії гравців з більшою точністю.

У цій області існують численні можливості для подальших досліджень та інновацій. Наприклад, можливо розвивати більш потужні та ефективні нейронні мережі, які здатні адаптуватися до стилю гри кожного гравця. Також, можливо досліджувати використання нейронних мереж у вирішенні більш

складних стратегічних завдань, таких як вирішення проблем у галузі бізнесу чи оборони [10]

В цілому, застосування нейронних мереж у стратегічних іграх є захоплюючою областю, яка має потенціал для подальшого розвитку та вдосконалення геймінгу, а також розвитку інтелектуальних навичок гравців.

1.3 Застосування елементів штучного інтелекту для активізації стратегічних ігор

Штучний інтелект (ШІ) у стратегічних іграх займає важливе місце і використовується для різних цілей. Цей інструмент, який базується на комп'ютерних алгоритмах та обчисленнях, має широкий спектр застосувань, серед яких створення віртуальних супротивників (некерованих персонажів - NPC) та оптимізація геймплею.

Агенти з сфери штучного інтелекту використовують алгоритми для прийняття рішень у грі. Ця практика є важливою в контексті розвитку і вдосконалення ігрових систем та інтелектуальних агентів. Алгоритми, які застосовуються в таких системах, можуть бути різними за складністю і ефективністю, і вони використовуються для досягнення оптимальних стратегій у грі.

Один із найпростіших алгоритмів, які використовують агенти, - це випадковий вибір. У цьому випадку агент приймає рішення на основі випадкових дій. Цей підхід може бути використаний для дослідження ігрового середовища та визначення базових стратегій. Однак він часто є непродуктивним в умовах складних ігор, де потрібно розробляти більш обчислювально-складні стратегії [11]

Для розв'язання більш складних завдань ігрової стратегії агенти можуть використовувати алгоритми машинного навчання. Ці алгоритми базуються на навчанні з використанням великої кількості даних та статистичних методів.

Вони дозволяють агентам вивчати оптимальні стратегії на основі досвіду та вдосконалювати їх у процесі гри. Зазвичай це включає в себе використання нейронних мереж та глибокого навчання для аналізу ігрового стану та прийняття рішень.

Використання алгоритмів машинного навчання для розв'язання ігрових завдань є надзвичайно важливим у сучасному світі інтелектуальних агентів. Нейронні мережі та глибоке навчання дозволяють агентам аналізувати складні ігрові сценарії та приймати вирішальні рішення на основі великої кількості даних.

Важливим фактором в застосуванні машинного навчання є збір і обробка даних. Ігрові агенти можуть використовувати дані зі своїх попередніх ігор або дані з ігор інших гравців для навчання. Таке навчання дозволяє агентам розвивати свої стратегії на основі реальних даних та адаптуватися до змінних умов гри.

Одним з прикладів успішного застосування машинного навчання у грі є гра в шахи. Програма AlphaZero, розроблена компанією DeepMind, використовуючи глибоке навчання самого себе грати у шахи, досягла неймовірних результатів у грі проти сильних супротивників. Ця програма навчилася грати на рівні світових чемпіонів та внесла значний вклад у розвиток інтелектуальних агентів.

Окрім глибокого навчання, існують також і інші методи машинного навчання, що використовуються для покращення ігрових стратегій. Наприклад, метод підсиленого навчання (reinforcement learning) дозволяє агентам навчатися на основі винагород та покарань, які вони отримують за свої дії у грі. Цей підхід дозволяє агентам навчатися на льоту та розвивати оптимальні стратегії спираючись свій на досвід [12]

Крім того, важливо відзначити, що машинне навчання знайшло застосування не лише у відеоіграх, але і у багатьох інших галузях. Наприклад, вона використовується для прогнозування ринкових тенденцій, оптимізації виробництва, аналізу медичних даних та багатьох інших задач.

Прогнозування поведінки гравців у відеоіграх є актуальною та важливою проблемою в сучасній галузі розваг та розвитку індустрії геймінгу. Системи штучного інтелекту (ШІ) в цьому контексті стають незамінним інструментом, який дозволяє аналізувати дані гравців та передбачати їхню майбутню поведінку і реакції в грі. Цей підхід робить геймінговий досвід більш цікавим і захоплюючим, оскільки дозволяє створювати більш складні та реалістичні взаємодії з неігровими персонажами (NPC).

Однією з ключових переваг використання систем ШІ для прогнозування поведінки гравців є їхня здатність аналізувати великі обсяги даних. Гравці залишають велику кількість слідів під час гри, такі як рухи, прийняті рішення, взаємодія з іншими гравцями та NPC, інформація про час гри та багато іншого. Системи ШІ можуть обробляти ці дані і використовувати їх для створення прогностичних моделей.

Для досягнення цієї мети, системи ШІ використовують різноманітні методи та алгоритми, такі як машинне навчання, нейронні мережі, глибоке навчання та інші. Вони аналізують попередні дії гравця, його вибори, реакції на певні події у грі та інші параметри для створення прогностичних моделей. Ці моделі дозволяють передбачити, як гравець може вести себе в майбутніх ситуаціях та які дії він може виконати.

Прогнозування поведінки гравців має різноманітні застосування в індустрії геймінгу. Воно дозволяє створювати більш інтелектуальних NPC, які можуть адаптуватися до дій гравців та надавати більш складні виклики. Також це дозволяє персоналізувати геймплей для кожного гравця, створюючи унікальний досвід [13]

Українська геймінгова індустрія також використовує системи ШІ для покращення якості своїх ігор. Знання про цей підхід та вміння впроваджувати його можуть стати важливими для розвитку геймінгової галузі в Україні.

Задача покращення інтелектуального рівня NPC (Non-Player Characters) у відеоіграх має велике значення для подальшого розвитку індустрії та покращення ігрового досвіду гравців. Цей аспект відіграє важливу роль у

підвищенні реалізму гри і створенні більш викликаючого інтерактивного середовища.

NPC (Non-Player Character) − це персонажі в комп'ютерних іграх, які не керуються гравцем, а замість цього програмною логікою або алгоритмами, які визначають їхню поведінку. Вони можуть виконувати різні ролі в грі, виступаючи союзниками, ворогами або нейтральними персонажами, і взаємодіяти з гравцем або між собою.

На даний момент із зростанням потужності обчислювальної техніки і розвитком технологій штучного інтелекту, NPC стають все більш складними і реалістичними. Вони можуть мати програмні моделі розуміння світу гри, вміти приймати рішення, планувати дії та навіть навчатися взаємодіяти з гравцем більш ефективно.

ШІ NPC може включати в себе такі аспекти, як визначення цілей, розробка стратегій, аналіз оточуючого середовища, реакція на події та подавання відповідей на дії гравця. Деякі ігри навіть використовують машинне навчання для покращення реалізації ШІ NPC.

Алгоритмічна складність створення розумних NPC (некерованих персонажів) у відеоіграх є невід'ємною складовою сучасного геймдизайну та вимагає застосування глибоких алгоритмів для моделювання прийняття рішень та взаємодії з гравцем. Ця проблема є досить складною та вимагає значних обчислювальних ресурсів та спеціалізованих підходів [14]

Алгоритмічна складність полягає у створенні NPC, які можуть реалістично реагувати на дії гравця та приймати відповідні рішення. Основні виклики включають в себе аналіз ситуації, прийняття рішень, планування дій та взаємодію з іншими NPC та об'єктами в грі. Ці функції потребують великої кількості обчислень та використання різних алгоритмів штучного інтелекту.

Одним із підходів до реалізації розумних NPC є використання алгоритмів машинного навчання, таких як нейронні мережі. За допомогою навчання на великих обсягах даних NPC можуть "вивчати" оптимальні стратегії та приймати рішення, спираючись на досвід.

Також важливо враховувати ресурсні обмеження при розробці таких алгоритмів. Велика кількість NPC в грі може призвести до значного навантаження на обчислювальний ресурс гри, що може вплинути на продуктивність ігрового середовища. Тому оптимізація та паралельна обробка графічних обчислень грають важливу роль у розробці розумних NPC.

Використання методів машинного навчання та штучного інтелекту безумовно є важливим кроком у покращенні інтелектуального рівня NPC (некерованих персонажів) у відеоіграх. Але, як вже зазначено, цей підхід вимагає деяких обмежень та ресурсів.

Одним з ключових аспектів використання машинного навчання є необхідність великої кількості даних для навчання моделей. У випадку NPC це означає, що для досягнення високого рівня інтелектуальної поведінки NPC потрібно зібрати велику кількість геймплейних данних з великої кількості різних сценаріїв гри. Це може бути важко зробити, оскільки вимагає часу та ресурсів для запису і анотування даних.

Крім того, навчання моделей машинного навчання може бути витратним за обчислювальними ресурсами. Навіть з великою кількістю даних, нейронні мережі та інші моделі можуть потребувати значних обчислювальних потужностей для тренування та виконання в грі. Це може вплинути на продуктивність гри, особливо на більш старих апаратних засобах.

Однак, не дивлячись на ці виклики, застосування машинного навчання може значно покращити ігровий досвід гравців, забезпечивши більшу реалістичність та інтелектуальність NPC. Важливо бути готовим до вкладання зусиль у збір та обробку даних, а також вдосконалення алгоритмів для досягнення бажаних результатів.

Реалізм поведінки NPC (Non-Player Character) є однією з важливих складових імерсії у відеоіграх. Гравці сьогодні висувають високі вимоги до реалістичності віртуальних світів, і одним із ключових аспектів, які можуть підняти рівень імерсії, є поведінка і реакція NPC.

Реалістична поведінка NPC включає в себе декілька аспектів, які потрібно враховувати. Перш за все, це адаптація до навколишнього середовища. NPC повинні демонструвати здатність адаптуватися до змін у грі, рухатися відповідно до обставин, уникати перешкод, інтерагувати з об'єктами і предметами навколо них. Це створює враження живого світу, де всі об'єкти і персонажі мають свою роль і ціль.

Крім того, реалістична поведінка NPC включає в себе реакцію на дії гравця. NPC повинні мати різноманітні варіанти відповідей на дії гравця, включаючи діалоги, жести, емоційні вирази обличчя та інші форми комунікації. Це дозволяє гравцям відчувати, що їхні дії мають вагу і впливають на гру.

Важливим аспектом є також адекватна реакція NPC на агресивні дії гравця. Якщо гравець вчиняє насильство в грі, то NPC повинні реагувати відповідним чином, демонструючи страх, захист або спротив.

Перш за все, важливо зазначити, що розумні NPC можуть значно покращити геймплей відеоігор. Гравці отримують більше викликів та задоволення, коли вони стикаються з розумними противниками або союзниками, які можуть застосовувати стратегії та тактики. Це робить гру більш цікавою і динамічною. Завдяки розумним NPC, гравці повинні розвивати власну стратегію та адаптуватися до дій NPC, що робить гру більш варіативною [15]

Один з основних аспектів взаємодії з NPC − це реалізм. Ігри з розумними NPC стають більш імерсивними і реалістичними. Розумні NPC ведуть себе більш природним чином, враховуючи контекст та ситуацію. Наприклад, вони можуть реагувати на зміну погоди, виявляти інтерес до різних предметів чи рухатися залежно від часу доби. Це збільшує захоплення гравців і робить гру більш життєвою.

Розумні NPC також відкривають нові можливості для створення ігор нових жанрів. Один із прикладів − ігри, де взаємодія з NPC грає основну роль. Наприклад, ігри, де гравці мають взаємодіяти зі спеціалізованими NPC-

персонажами, які вчать їх різним навичкам або надають інформацію про грунтовний світ гри. Це може стати основою для нового жанру ігор, де розумні NPC стають ключовими персонажами, з якими гравці спілкуються та співпрацюють.

Покращення інтелектуального рівня NPC − це важлива задача для розвитку індустрії відеоігор. Це сприяє підвищенню реалізму, глибини та викликів гри. Однак це вимагає великих зусиль у розробці алгоритмів, навчанні машин та досягненні реалістичної поведінки NPC. Подальший розвиток цього напрямку може відкрити нові горизонти для геймдизайну та ігрової індустрії в цілому.

Висновки до 1 розділу

Підсумовуючи розділ 1, аналіз існуючих стратегічних ігор реального часу надає важливий інсайт у світ сучасних ігор та їхні можливості для подальшого дослідження та вдосконалення. У цьому розділі було розглянуто класифікацію стратегічних ігор реального часу, а також застосування нейронних мереж і елементів штучного інтелекту для активізації таких ігор.

Класифікація стратегічних ігор реального часу є важливим етапом у розумінні різноманітності цього жанру ігор. Аналіз показує, що ці ігри можуть бути розділені на декілька основних категорій в залежності від їхнього геймплею та цільової аудиторії. Наприклад, існують стратегічні ігри в реальному часі для поціновувачів військових симуляцій, де головним завданням є ведення війни та побудова армії. З іншого боку, існують ігри для тих, хто більше зацікавлений у розвитку і управлінні містами або цивілізаціями. Ця класифікація допомагає розуміти різноманітність стратегічних ігор та вибирати найбільш підходящий жанр для конкретних досліджень.

Застосування нейронних мереж для активізації стратегічних ігор реального часу є однією з передових технологій у цьому напрямку. Нейронні

мережі можуть бути використані для створення розумних ігрових агентів, які можуть приймати стратегічні рішення на основі аналізу гри в реальному часі та динамічно адаптуватися до змінних умов. Це робить ігровий процес цікавішим для гравців, оскільки вони змушені конкурувати з розумними опонентами.

Застосування елементів штучного інтелекту також має великий потенціал для покращення стратегічних ігор реального часу. Штучний інтелект може допомогти вирішувати складні завдання, такі як оптимізація ресурсів чи передбачення дій опонентів. Це сприяє підвищенню складності гри і створює нові можливості для розвитку стратегічних навичок у гравців.

2 МОДЕЛЬ ТА МЕТОД ПОКРАШЕННЯ СТРАТЕГІЧНОЇ ГРИ РЕАЛЬНОГО ЧАСУ З НЕЙРОННОЮ МЕРЕЖЕЮ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Побудова моделі стратегічної гри реального часу на основі нейронної мережі зі штучним інтелектом.

Проаналізувавши велику кількість досліджень та матеріалів щодо підходів до створення штучного інтелекту для стратегічних ігор у реальному часі, можна зробити висновок, що найбільш якісні та гнучкі рішення використовують комбінацію кількох підходів для створення різних частин штучного інтелекту.

Для так званого тактичного штучного інтелекту найбільш перспективною видається технологія нейронних мереж, яка дозволить йому правильно реагувати навіть у абсолютно невідомій ситуації.

Розглянемо особливості побудови класичної нейронної мережі, яка називається багаторівневим перцептроном [6].

Штучні нейронні мережі (ШНМ) − це програмна реалізація нейронних структур нашого мозку. Вони можуть змінювати тип сигналів, що передаються залежно від електричних або хімічних сигналів, що передаються в них. Нейронна мережа в мозку людини − це величезна взаємопов’язана система нейронів, у якій сигнал, переданий одним нейроном, може передаватися тисячам інших нейронів. Навчання відбувається шляхом багаторазової активації деяких нейронних зв'язків. Це підвищує ймовірність досягнення бажаного результату при відповідній вхідній інформації (сигналах). У цьому типі навчання використовується зворотний зв’язок – коли результат правильний, нейронні зв’язки, які до нього призводять, стають щільнішими.

Штучні нейронні мережі імітують поведінку мозку в простішій формі. Їх можна дресирувати як контрольовано, так і неконтрольовано. У

контрольованій ШМН мережа навчається шляхом введення відповідної вхідної інформації та прикладів вихідної інформації. Наприклад, спам-фільтр у вхідній пошті: вхідною інформацією може бути список слів, які зазвичай з’являються в спам-повідомленнях, а вихідною інформацією може бути класифікація відповідного повідомлення (спам чи не спам). Такий тип навчання збільшує вагу з’єднань ШНМ.

Біологічний нейрон моделюється в ШНМ за допомогою функції активації. Для завдань класифікації (наприклад, виявлення спаму) функція активації повинна мати властивість «перемикати». Іншими словами, якщо вхід більше певного значення, вихід повинен змінити стан, наприклад, з 0 на 1 або з -1 на 1. Це імітує «включення» біологічного нейрона. Сигмоїдна функція (рис.2.1) зазвичай використовується як функція активації:

$$f(z) = \frac{1}{1+\exp(-z)} \tag{2.1}$$

Графік цієї функції має такий вигляд (рис. 2.1).

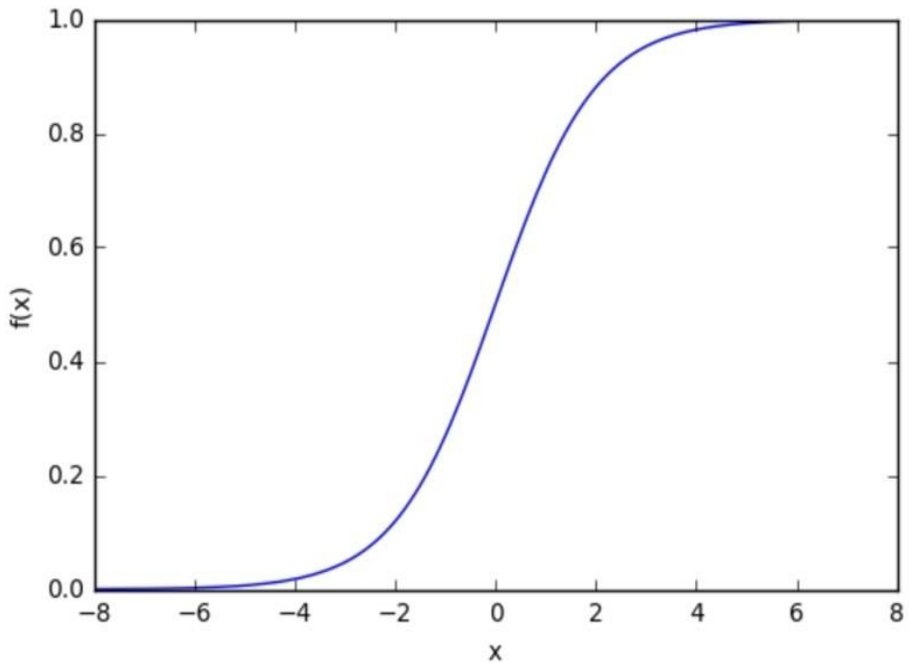

Рисунок 2.1 – Сигмоїдна функція (графіка)

На графіку видно, що функція є «активаційною» − вона зростає від 0 до 1 із кожним збільшенням значення x. сигмоподібна функція плавна і

безперервна. Це означає, що функція має похідну, що, у свою чергу, є дуже важливим фактором для навчання алгоритму.

Як згадувалося раніше, біологічні нейрони ієрархічно з’єднані в мережі, причому вихідні дані деяких нейронів є входом для інших нейронів. Можна представити такі мережі як зв'язані шари з вузлами. Кожен вузол отримує зважений вхід, активує функцію активації на основі суми вхідних даних і генерує вихід.

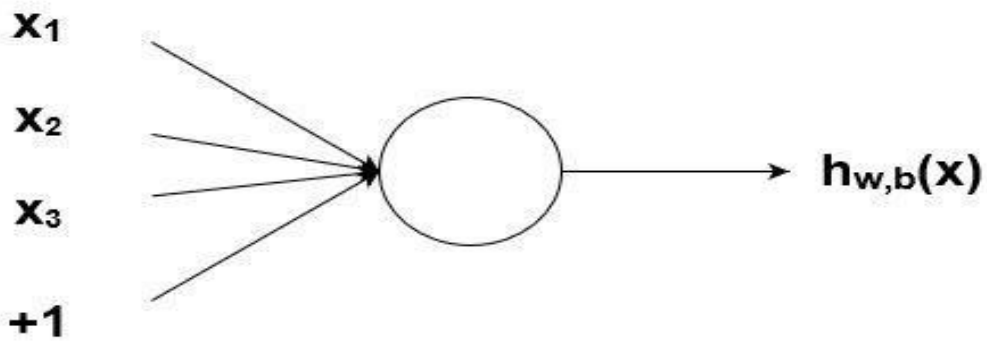

Рисунок 2.2 – Штучний нейрон із входами від попереднього вузла x1...x3 і зміщенням 1

Коло на зображенні позначає вузол. Вузол є «розташуванням» функції активації. Він бере зважені вхідні дані, підсумовує їх, а потім передає їх у функцію активації. Вихід функції активації представлено h (у деякій літературі вузол також називають перцептроном). Числа (недвійкові) використовуються як ваги, які потім множаться на вході та підсумовуються у вузлі.

Зважений вхід у вузол має вигляд:

$$x_1 w_1 + x_2 w_2 + x_3 w_3 + b \qquad (2.2)$$

де wi− числові значення ваги (ваги − це значення, які змінюються в процесі навчання), а b − вага елемента зі зміщенням на +1, де вага b робить вузол більш гнучким.

Як працює кожен вузол/нейрон/персептрон, було пояснено вище. Але в повній нейронній мережі є багато таких взаємопов’язаних вузлів. Структури

таких мереж можуть приймати незліченну кількість різних форм, але найпоширеніша складається з вхідного рівня, прихованого рівня та вихідного рівня. Приклад такої структури показано нижче: (рис. 2.3)

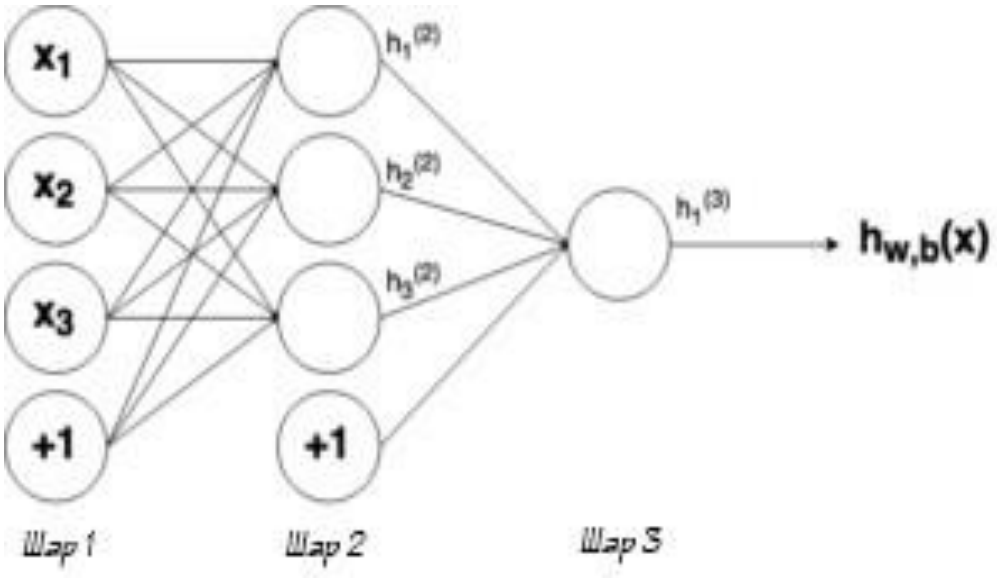

Рисунок 2.3 – Модель зв’язків у триступеневій штучній нейронній мережі

На зображенні вище можна побачити три рівні мережі. Шар 1 − це вхідний рівень, де мережа приймає зовнішні вхідні дані. Шар 2 називається прихованим шаром. Цей рівень не є ані частиною входу, ані виходу. І, нарешті, шар 3 є вихідним. Існує багато зв’язків між рівнем 1 (W1) і рівнем 2 (W2). Кожен вузол в Ø1 має з’єднання з усіма вузлами в Ø2 і з кожного вузла у h2 проходить через підключення до єдиного вихідного вузла в h3. Кожна з цих сполук повинна мати відповідну вагу.

Щоб продемонструвати процес прямого розповсюдження з уже відомим входом у нейронні мережі, слід почати з попереднього прикладу з трьома шарами. Така система представлена нижче у вигляді системи рівнянь: [16]

$$h1(2) = f(w11(1)x1 + w12(1)x2 + w13(1)x3 + b1(1)) \quad (2.3)$$

$$h2(2) = f(w21(1)x1 + w22(1)x2 + w23(1)x3 + b2(1)) \quad (2.4)$$

$$h3(2) = f(w31(1)x1 + w32(1)x2 + w33(1)x3 + b3(1)) \quad (2.5)$$

$$h1(3) = f(w11(2)h1(2) + w12(2)h2(2) + w13(2)h3(2) + b1(2)) \quad (2.3)$$

Останній рядок обчислює вихід одного вузла на останньому третьому рівні, який представляє кінцеву точку виходу в нейронній мережі. У ньому замість зважених вхідних змінних (x_1,x_2,x_3) використовуються зважені виходи вузлів з другого шару (h1(2),h2(2),h3(2)) і переміщення. Така система рівнянь також добре показує ієрархічну структуру нейронної мережі.

Наступний крок − вставити в систему значення ваг і переміщень і обчислити результат: [16]

$$h1(2) = f(0.2 \times 1.5 + 0.2 \times 2.0 + 0.2 \times 3.0 + 0.8) = 0.8909 \quad (2.7)$$

$$h2(2) = f(0.4 \times 1.5 + 0.4 \times 2.0 + 0.4 \times 3.0 + 0.8) = 0.9677 \quad (2.8)$$

$$h3(2) = f(0.6 \times 1.5 + 0.6 \times 2.0 + 0.6 \times 3.0 + 0.8) = 0.9909 \quad (2.9)$$

$$h1(3) = f(0.5 \times 0.8909 + 0.5 \times 0.9677 + 0.5 \times 0.9909 + 0.2) = 0.8354 \quad (2.10)$$

Обчислення вагових коефіцієнтів, які з’єднують шари в мережі, називаються вивченням системи. Контрольоване навчання − це зменшення похибок між введенням і бажаним результатом. Якщо нейронна мережа з вихідним рівнем і деяким входом x, де вихід має бути числом 2, в той час як мережа виводить 5, тоді усунення похибки виглядає так: abs(2−5)=3.

Важливість контрольованого навчання полягає в тому, що надається багато пар введення-виведення вже відомих даних, і значення вагових коефіцієнтів потрібно змінити на основі цих прикладів, щоб значення помилки стало мінімальним. Ці пари введення-виведення позначаються як (x(1),y(1)),…,(x(m),y(m)), де m – кількість екземплярів, які потрібно навчити. Кожне вхідне або вихідне значення може представляти вектор значень. Наприклад, x(1) не обов’язково має мати тільки одне значення, а може містити N-вимірний набір значень.

Під час навчання мережі за допомогою (x,y) мета полягає в тому, щоб покращити пошук правильного y при заданому x. Це робиться шляхом зміни значень ваг, щоб мінімізувати помилку. Для цього знадобиться градієнтний спуск (рисунок 2.4).

Цей графік показує похибку як функцію скалярного значення ваги w. Мінімальна можлива похибка позначена чорним хрестиком, але точно не відомо, яке значення w дає це мінімальне значення. Розрахунок починається з випадкового значення змінної w, яке дає похибку, позначену червоною крапкою під цифрою «1» на кривій. Щоб отримати мінімальну помилку, чорний хрестик, потрібно змінити w. Одним з найпоширеніших методів є градієнтний спуск.

По-перше, існує градієнт помилки «1» відносно w. Градієнт − це висота нахилу кривої у відповідній точці. Це показано на графіку чорними стрілками. Градієнт також дає деяку інформацію про напрямок - якщо він додатний, коли w збільшується, то похибка в цьому напрямку збільшується, якщо він від'ємний, вона зменшується (див. графік).

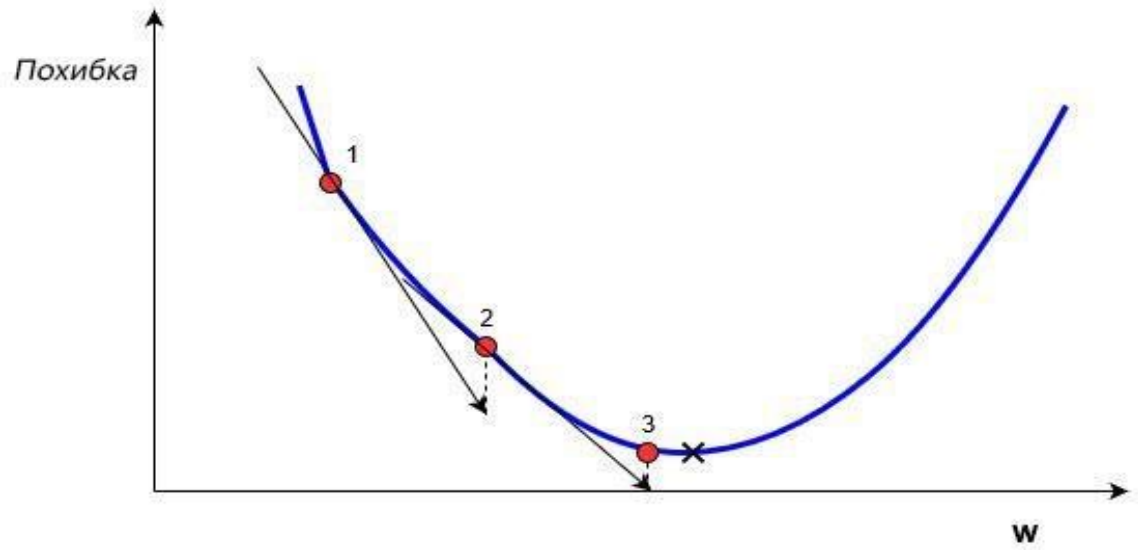

Рисунок 2.4 – Похибка в залежності від скалярного значення ваги w

Як стало зрозуміло, з кожним кроком слід намагатись зменшувати помилку. Величина градієнта показує, наскільки швидко змінюється крива похибки або функція у відповідній точці. Чим більше значення, тим швидше змінюється помилка у відповідній точці залежно від w.

Метод градієнтного спуску використовує градієнт для визначення наступної зміни w для досягнення мінімального значення кривої. Це

ітераційний метод, де значення w оновлюється кожного разу: $w_н = w_{ст} - \alpha * \nabla error$ (2.12), Де wn − нове значення w, $w_{ст}$ − поточне або «старе» значення w, ∇error − градієнт помилки на $w_{ст}$, а α − крок [16]

Крок α також показує, як швидко відповідь наближається до мінімальної помилки. З кожною ітерацією в такому алгоритмі градієнт повинен зменшуватися. На графіку вище можна побачити, що градієнт «зменшується» з кожним кроком. Коли відповідь досягає мінімального значення, необхідно вийти з ітераційного процесу. Вихід може бути здійснений за умовою «якщо помилка менше певного числа». Це число називається точністю.

Градієнтний спад для кожної ваги w(ij)(l) і bi(l) у нейронній мережі виглядає наступним чином: [16]

$$wij(l) = wij(l) - \alpha\ \partial\ w\partial ij(l)\ J(w, b) \quad (2.12)$$

$$bij(l) = bij(l) - \alpha\ \partial\ b\partial ij(l)\ J(w, b) \quad (2.13)$$

Наведений вище вираз фактично аналогічний представленню градієнтного спуску: $w_н = w_{ст} - \alpha * \nabla error$. Є лише деякі позначення, але цього достатньо, щоб зрозуміти, що зліва знаходяться нові значення, а справа − старі. Знову ж таки, ітераційний процес використовується для обчислення ваг на кожній ітерації, але цього разу на основі оціночної функції J(w,b).

2.2 Задачі в межах гри, які повинні вирішувати нейронні мережі з елементами штучного інтелекта

Штучний інтелект в гральній індустрії має значний потенціал для створення реалістичного та захоплюючого ігрового досвіду. Однією з ключових областей застосування штучного інтелекту в іграх є використання нейронних мереж для моделювання поведінки інтелектуальних супротивників та союзників.

За допомогою нейронних мереж можна створювати агентів, які здатні аналізувати дії гравців у грі та приймати відповідні рішення. Це дозволяє

створювати геймплей, який не лише відповідає задачам гри, але й адаптується до вчинків гравців.

Нейронні мережі в іграх можуть використовуватися для різноманітних завдань. Наприклад, вони можуть допомагати супротивникам адаптуватися до стратегій гравців, прогнозувати їхні дії та намагатися досягти максимальної геймплейної ефективності. Також нейронні мережі можуть бути використані для створення інтелектуальних союзників, які співпрацюють з гравцем, надаючи допомогу і стратегічну підтримку.

Один з основних викликів у застосуванні нейронних мереж в ігровому середовищі полягає в їхній навченості і адаптації до змінних умов гри. Це вимагає розробки складних алгоритмів навчання та постійного вдосконалення моделей [17]

Українські розробники також активно застосовують штучний інтелект у гральній індустрії, створюючи високоякісні ігри та використовуючи передові методи нейронних мереж для покращення геймплею.

Завдяки поєднанню штучного інтелекту та ігрової індустрії можливо створювати ігри, які не лише розважають, але й надають можливість гравцям взаємодіяти з реалістичними та інтелектуальними персонажами, підвищуючи рівень поглиблення та задоволення від гри.

Завдання оптимізації графіки в ігровій індустрії має важливе значення для створення ігор, які надають гравцям найвищий рівень реалізму та імерсії. Одним із ключових компонентів цього завдання є створення високоякісних текстур для об'єктів у грі. Текстури визначають зовнішній вигляд об'єктів, надаючи їм характер, деталізацію і реалізм.

Для досягнення цієї мети, вчені та розробники ігор в останні роки активно використовують нейронні мережі. Ці мережі можуть аналізувати велику кількість даних та генерувати текстури автоматично. Вони базуються на глибокому навчанні, яке надає їм здатність вивчати зразки та закономірності у текстурах, що робить їх дуже потужними інструментами для створення візуальних компонентів гри.

Однією з основних переваг використання нейронних мереж у генерації текстур є їх здатність враховувати різні аспекти, що впливають на зовнішній вигляд об'єктів. Вони можуть аналізувати текстурні особливості, враховувати освітлення та тіні, а також імітувати природні процеси, такі як знос та старіння. Це дозволяє створювати текстури, які виглядають природними та реалістичними, покращуючи візуальний досвід гравців.

Застосування нейронних мереж у генерації текстур вимагає обширних обчислювальних ресурсів та великих наборів даних для навчання. Однак результати цього підходу вражають своєю якістю та деталізацією, роблячи графіку більш реалістичною та захоплюючою.

Ще однією важливою складовою графіки в іграх є моделі персонажів. Тут нейронні мережі можуть бути використані для автоматичної генерації персонажів з високою деталізацією. Однією з головних переваг використання нейронних мереж є їх здатність аналізувати та розпізнавати структуру людського тіла. Вони можуть аналізувати анатомічні особливості, включаючи пропорції, м'язи та кістки, щоб створити персонажів з реалістичною анімацією та рухами. Це допомагає підвищити правдоподібність персонажів і забезпечити їхню природність [18]

Крім того, нейронні мережі можуть аналізувати обличчя та генерувати високоякісні текстури для обличчя персонажів. Це дозволяє створювати персонажів з виразними обличчями, які можуть виражати емоції та взаємодіяти з іншими персонажами та гравцями в грі.

Також, нейронні мережі можуть генерувати різні види одягу та аксесуарів для персонажів, роблячи їх стильними та унікальними. Це дозволяє розробникам створювати різноманітних персонажів, які можуть відповідати різним жанрам та налаштуванням гри.

Застосування нейронних мереж для автоматичної генерації персонажів з високою деталізацією покращує процес розробки ігор і забезпечує більш реалістичний та імерсивний геймплей. Ця технологія продовжує зростати та

розвиватися, відкриваючи нові можливості для індустрії відеоігор та надаючи гравцям більш захопливий ігровий досвід.

Анімація в іграх є надзвичайно важливою, оскільки вона допомагає створити живий та імерсивний ігровий світ. Нейронні мережі дійсно революціонізують цей аспект розробки ігор, дозволяючи автоматично генерувати рухи персонажів та об'єктів з вражаючою деталізацією та реалістичністю.

Однією з ключових переваг використання нейронних мереж у створенні анімації є їх здатність вивчати зразки реальних анімаційних даних. Вони аналізують рухи людей та об'єктів, враховуючи фізичні закони та взаємодію з оточуючим світом. Це дозволяє їм створювати рухи, які виглядають природними та плавними.

Крім того, нейронні мережі можуть синтезувати нові рухи, використовуючи вивчені зразки. Це дозволяє розробникам ігор створювати унікальні та натуральні анімації для персонажів та об'єктів. Наприклад, вони можуть генерувати анімації для персонажів у різних ситуаціях, від руху та бою до міміки обличчя та взаємодії з іншими персонажами.

Цей підхід до створення анімації робить її більш доступною та ефективною для розробників ігор. Він дозволяє створювати анімації високої якості без необхідності великої кількості ручної роботи. Це особливо важливо для ігор з великою кількістю персонажів та складними сценами [19]

Покращення інтерфейсу користувача є ключовим аспектом розробки ігор, оскільки вони мають надавати гравцям зручність і доступність, а також допомагати їм зрозуміти гру та приймати оптимальні рішення. Нейронні мережі можуть бути надзвичайно корисним інструментом для оптимізації інтерфейсу гри та надання рекомендацій гравцям.

Аналіз дій користувачів є однією з ключових функцій нейронних мереж у покращенні інтерфейсу гри. Вони можуть відстежувати, як гравці взаємодіють з грою, аналізувати їх рухи, рішення та взаємодію з об'єктами в грі. На основі цього аналізу нейронні мережі можуть розробляти інтерфейс,

який оптимізований для конкретного гравця. Наприклад, вони можуть пропонувати підказки, рекомендації або навіть змінювати складність гри, щоб зробити її більш захоплюючою або доступною.

Покращення інтерфейсу також може включати автоматичну адаптацію до потреб гравців. Нейронні мережі можуть аналізувати стиль гри гравця та надавати інтерфейс, який найкраще підходить для їхніх вподобань. Наприклад, якщо гравець виявляє певні тенденції в стратегії гри, інтерфейс може пропонувати рекомендації, які відповідають цим стратегіям.

Крім того, нейронні мережі можуть надавати допомогу у навчанні гравців. Вони можуть створювати персоналізовані навчальні матеріали, які допомагають гравцям краще розуміти гру та розвивати їх навички. Це особливо корисно для новачків, які потребують додаткової підтримки та навчання.

Генерація контенту за допомогою нейронних мереж є однією з найцікавіших та перспективних областей в розробці ігор. Цей підхід дозволяє розробникам створювати безмежно різноманітний та цікавий контент для гравців, підвищуючи тривалість та розвагу від гри.

Однією з основних можливостей нейронних мереж є генерація рівнів та завдань в грі. Вони можуть аналізувати існуючі рівні та створювати нові на основі цього аналізу. Це дозволяє створювати різноманітні завдання, які залишають гравців зацікавленими та мотивованими [20]

Додатково, нейронні мережі можуть генерувати діалоги для персонажів у грі. Вони можуть вивчати структуру розмов та створювати автентичні діалоги, які відображають характери персонажів та розвивають сюжет гри.

Музика та звуки також можуть бути згенеровані нейронними мережами. Вони можуть аналізувати настрій та контекст гри та створювати відповідну музичну супровідку та звуковий декор. Це додає до гри атмосферу та емоційну глибину.

Генерація контенту за допомогою нейронних мереж розширює можливості розробників та робить ігри більш цікавими та варіативними для

гравців. Вона дозволяє створювати ігри з безмежними можливостями і нескінченним полем для розваг, що забезпечує тривалість та популярність гри на ринку.

Планування та дизайн гри є фундаментальними етапами у використанні нейронних мереж для оптимізації графіки та інших аспектів гри.

Перший, інтегральний, етап використання нейронних мереж у галузі комп'ютерних наук та штучного інтелекту полягає у чіткому визначенні конкретних завдань, які передбачається вирішувати за допомогою цих мереж. Цей етап включає в себе ряд важливих аспектів, таких як: вибір відповідних типів мереж, розробка їхньої архітектури та формулювання конкретних завдань, які мають бути виконані за допомогою нейронних мереж.

Один із ключових аспектів цього етапу - це вибір типу нейронної мережі, яка найкраще підходить для вирішення конкретного завдання. Наприклад, для завдань, пов'язаних із обробкою зображень, використовуються зазвичай згорткові нейронні мережі (CNN), вони ефективно впораються з визначенням об'єктів на зображеннях або класифікацією зображень. У випадку текстового аналізу використовуються рекурентні нейронні мережі (RNN) або трансформери для обробки послідовних даних [21]

Далі, не менш важливим аспектом є розробка архітектури мережі. Вона включає в себе визначення кількості шарів, кількості нейронів у кожному шарі, вибір активаційних функцій та інших параметрів, які впливають на продуктивність та точність мережі. Коректний вибір архітектури може вирішально вплинути на успішність вирішення завдання.

Останнім, але не менш важливим етапом є формулювання конкретних завдань для мережі. Це може включати в себе визначення критеріїв успіху, метрик для оцінки результатів та методів тренування мережі з використанням наявних даних.

Після чіткого визначення поставлених завдань у розробці комп'ютерної гри, однією з ключових складових процесу стає інтеграція нейронних мереж у програмний код гри. Цей крок є вкрай важливим, оскільки від нього залежить

здатність гри до вивчення та адаптації до різних ситуацій, що виникають під час геймплею.

Інтеграція нейронних мереж передбачає використання спеціалізованих бібліотек та фреймворків для машинного навчання. Зазвичай ці бібліотеки надають розробникам інструменти для створення та тренування нейронних мереж з різними архітектурами. Важливо відзначити, що в Україні існують талановиті програмісти та дослідники в області машинного навчання, які активно вносять свій внесок у цю сферу.

Крім того, розробники повинні створити інтерфейс, який дозволить взаємодіяти з грою та нейронними мережами. Це включає в себе створення можливості передачі даних між грою та мережами, щоб забезпечити ефективну обробку інформації та прийняття відповідних рішень. У цьому контексті важливо мати на увазі високу швидкодію та надійність інтерфейсу, оскільки з іншого боку, графічний двигун гри також вимагає значних ресурсів обчислювальної потужності [9]

Необхідно також наголосити, що розробка гри з використанням нейронних мереж відкриває безмежні можливості для створення ігор зі складними та інтелектуальними противниками, а також для розвитку нових геймплейних механік, які можуть зацікавити гравців. Дана інтеграція дозволяє створити гри, які здатні навчатися від гравців та пристосовуватися до їхнього стилю гри, що робить геймплей більш захоплюючим та непередбачуваним.

Після успішної інтеграції мережі глибокого навчання, необхідно розглянути процес навчання цієї мережі для виконання конкретних завдань у межах відеоігор. Ця задача передбачає великий обсяг роботи, яка включає в себе як навчання на великій кількості даних, так і налаштування параметрів мережі для досягнення бажаних результатів.

Перш за все, слід зазначити, що мережі глибокого навчання вказують на нейронні мережі, які мають багато шарів (відомих як глибокі шари), які використовуються для аналізу та вирішення різних завдань. Ці мережі набули

великої популярності завдяки їх здатності вчитися і розуміти складні залежності в даних.

Для навчання мережі для виконання завдань у межах відеоігор, необхідно здійснити кілька важливих кроків. Перший крок - це підготовка великої кількості даних для навчання. У випадку використання мережі для генерації текстур, ці дані могли б включати в себе зразки різних текстур, які слід генерувати. Для досягнення високої якості вивчення, важливо мати різноманітні дані, що включають в себе різні текстурні структури та їх особливості.

Після підготовки даних, наступним кроком є конфігурування самої мережі. Це включає в себе визначення архітектури мережі, кількості шарів, кількості нейронів у кожному шарі, вибір функцій активації та оптимізаційних алгоритмів. На цьому етапі також важливо налаштувати параметри навчання, такі як швидкість навчання (learning rate) та регуляризація, для досягнення бажаних результатів [13]

Основна мета навчання мережі полягає в тому, щоб вона стала здатною розпізнавати структуру та особливості текстур у вхідних даних. Цей процес включає в себе тренування мережі на підготовлених даних та налаштування параметрів таким чином, щоб мережа вчилася ефективно та досягала високої точності при розпізнаванні текстур.

Після завершення навчання мережі глибокого навчання, важливим етапом є проведення тестування та оптимізації її роботи в грі. Це важливий крок, оскільки він допомагає впевнитися в ефективності та стабільності мережі під час реальної геймплею. Тестування включає в себе оцінку різних аспектів, таких як якість графіки, анімації та інших елементів, які були оптимізовані за допомогою мережі, а також перевірку загальної функціональності гри.

Перший крок у тестуванні − це оцінка якості графіки та анімації, які були згенеровані мережею. Це включає в себе перевірку деталей, реалістичності та збігу з очікуваною якістю. Оптимізація мережі може вимагати деяких

компромісів між якістю та швидкістю генерації, тому важливо визначити оптимальний баланс.

Крім того, під час тестування слід оцінити продуктивність мережі та її вплив на роботу гри. Це включає в себе вимірювання швидкості обчислень, використання пам'яті та інших ресурсів. Оптимізація мережі може включати в себе редагування архітектури для зменшення обчислювального навантаження та використання більш ефективних алгоритмів обробки даних.

Важливим аспектом оптимізації є покращення швидкості та ресурсозбереження мережі. Гра повинна працювати плавно та без перебоїв навіть на комп'ютерах з обмеженими характеристиками. Це може вимагати оптимізації параметрів мережі, використання апаратного прискорення та інших технік для забезпечення оптимальної продуктивності.

Отже, використання нейронних мереж в процесі розробки гри вимагає детального планування, реалізації, навчання та оптимізації. Проте цей підхід може значно підвищити якість графіки та інших аспектів гри, роблячи її більш привабливою для гравців та надаючи їм нові імерсивні ігрові досвіди.

2.3 Удосконалення характеристик гри

Розвиток штучного інтелекту у геймдеві справді відкриває безмежні можливості для поліпшення характеристик ігор та покращення взаємодії з гравцями. Сучасні технології дозволяють створювати більш інтелектуальних ігрових персонажів, забезпечувати реалістичнішу поведінку комп'ютерних супротивників і створювати більш складні сценарії гри.

Один з основних аспектів використання штучного інтелекту в геймдеві − це створення "розумних" ігрових персонажів. Сучасні AI алгоритми дозволяють цим персонажам адаптуватися до дій гравця, виконувати складні рішення в реальному часі та навіть навчатися на основі досвіду.

Крім того, штучний інтелект також використовується для оптимізації графіки та фізики у іграх. Він дозволяє реалізувати більш високий рівень

деталізації, створювати реалістичні фізичні ефекти та покращувати загальну якість графіки.

Також слід зазначити, що штучний інтелект у геймдеві використовується для розв'язання проблем зі збереженням ігрових світів та оптимізації продуктивності гри. Він дозволяє створювати більш масштабні ігрові світи, які можуть бути завантажені та відновлені більш ефективно.

Одним з напрямків застосування штучного інтелекту є розробка високоінтелектуальних ворогів та персонажів гри. Штучний інтелект дозволяє створювати персонажів, які мають здатність адаптуватися до змінюючихся умов гри, самостійно формувати стратегії, виконувати швидкі та складні рішення та реалістично реагувати на дії гравця. Це поліпшує ігровий досвід, робить гру більш викликаючою та захоплюючою.

Додатково, штучний інтелект може вдосконалювати різноманітні механіки гри, зокрема алгоритми навігації, штучну фізику та системи штучного життя. Нейронні мережі та генетичні алгоритми використовуються для вирішення проблем балансу гри та оптимізації параметрів, забезпечуючи якісну та розумну геймплейну динаміку.

Багато проектів також стосуються генерації контенту за допомогою штучного інтелекту. Нейронні мережі здатні навчатися на великій кількості даних та створювати нові образи, рівні та складові гри, що забезпечує більшу різноманітність та цікавість геймплею.

Додатково, штучний інтелект може бути використаний для автоматичного тестування гри, що забезпечує виявлення помилок, оптимізацію продуктивності та поліпшення якості гри.

Загалом, штучний інтелект стає безперечно важливим інструментом у геймдеві для вдосконалення характеристик гри та забезпечення незабутнього та захоплюючого досвіду для гравців.

Висновок до 2 розділу

У даному розділі була розглянута модель та метод покращення стратегічної гри реального часу за допомогою нейронної мережі з елементами штучного інтелекту. Використання штучного інтелекту в гральних сценаріях є актуальним та дозволяє підвищити рівень інтелектуальності та реалістичності персонажів гри.

Побудова моделі стратегічної гри реального часу на основі нейронної мережі дозволяє створити систему, яка може навчатися на основі даних, аналізувати геймплей та приймати рішення в реальному часі. Це дозволяє персонажам гри бути більш інтелектуальними, прогнозувати дії гравців та адаптуватися до змінюючихся умов гри.

Нейронні мережі з елементами штучного інтелекту мають багато завдань в межах гри, які вони можуть вирішувати. Вони можуть навчитись планувати та виконувати складні стратегії, аналізувати ситуацію та вирішувати тактичні завдання. Також вони можуть бути використані для розробки балансу гри і оптимізації параметрів, що забезпечить цікавий та рівноправний досвід гравців.

Одним із основних напрямків є удосконалення характеристик гри. Штучний інтелект може допомогти вдосконалити алгоритми навігації персонажів, реалістичність фізичної моделі, розвивати та прогресувати логіку вирішення завдань. Це дозволить гравцям отримати більш захоплюючий досвід у грі.

Отже, модель покращення стратегічної гри реального часу на основі нейронної мережі з елементами штучного інтелекту виявляється перспективним інструментом, який може покращити ігровий досвід гравців та розширити можливості в геймдеві. Дослідження в даному напрямку може призвести до створення більш реалістичних та інтелектуальних ігрових систем в майбутньому.

3 РОЗРОБКА ІНФРАСТРУКТУРИ ГРИ З НЕЙРОННОЮ МЕРЕЖЕЮ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

3.1 Створення середовища для проведення моделювання, тестування і валідації гри

Для розробки штучного інтелекту прийнято використання платформи .NET. Це пов'язано з тим, що нейронна мережа та інші елементи штучного інтелекту повинні бути інтегровані в користувацький додаток. Це дозволяє гравцям використовувати гру з ботом навіть без інтернету, а також прискорює продуктивність і стабільність, оскільки всі розрахунки і рішення виконуються миттєво в доступному користувачеві програмному коді.

Попередня обробка даних вимагає ряду дій, які виконуються поза нейронною мережею. У рамках тестового середовища створено окремий модуль штучного інтелекту. На додаток до структури нейронної мережі та завантажених ваг і зміщень, модуль реалізує функції попередньої обробки.

Наступні методи є основними для попередньої обробки даних: [15]

- GetAnalyzingTerritory()
- SplitTerritory(Територія ter)
- HandleZones(List<Zone> zns)
- GetUnitsPower (зона zn)
- IsOwnUnitsExists(зона zn)
- CountPower (зона zn)
- CountInfluence(List<Zone> zns)

Методи працюють один за одним: спочатку визначається область для аналізу, а потім ця область ділиться на частини, так звані зони.

Після поділу на зони кожна з них піддається обробці. Алгоритм визначає кількість військ і загальну «силу» підрозділу в зоні. Потім це значення використовується для визначення впливу.

Для зон, де немає бойових підрозділів, подальший аналіз не проводиться. Клас Territory відповідає за загальну територію в системі класів, кожна зона є об'єктом класу Zone.

Після проходження першого рівня обробки даних нейронна мережа застосовується до певного списку зон. У зв’язку з необхідністю використання мережі як в серверній частині, так і в користувальницькій програмі було вирішено створити власне рішення на основі мови C#.

Цей підхід довелося застосувати через низку проблемних моментів у двигуні Unity, які не дозволяють зручно додавати пакети з Nuget.

Nuget − це основний менеджер пакетів для проектів, написаних за допомогою .NET Framework і стандарту для розробки. Окрім завантаження пакетів зі спеціальних репозиторіїв, є можливість додавати бібліотеки класів безпосередньо до вихідного коду проекту Unity [15]

Це створює певні проблеми, оскільки підхід ручного додавання пакетів є надто негнучким, що призводить до помилок синхронізації та помилок компіляції.

Наприклад, використання бібліотеки SugnalR вимагає додавання більше 15 файлів різних бібліотек, що створює потенційні проблеми в майбутньому під час оновлення поточної версії бібліотеки (оскільки бібліотеки часто підвищують необхідний рівень залежностей під час переходу на нову версію).

Для створення нейронної мережі основними елементами є нейрони. У вибраній схемі класів за цей елемент мережі відповідає Neuron, він реалізує інтерфейс INeuron. Кожен з нейронів розробленої мережі належить до класу NeuronLayer, який відповідає за кожен окремий рівень мережі, незалежно від типу шару та його властивостей.

Кожен шар нейронів, у свою чергу, належить до класу NeuralNetwork, основного класу для нейронної мережі. Рисунок 3.1 показує структуру класу.

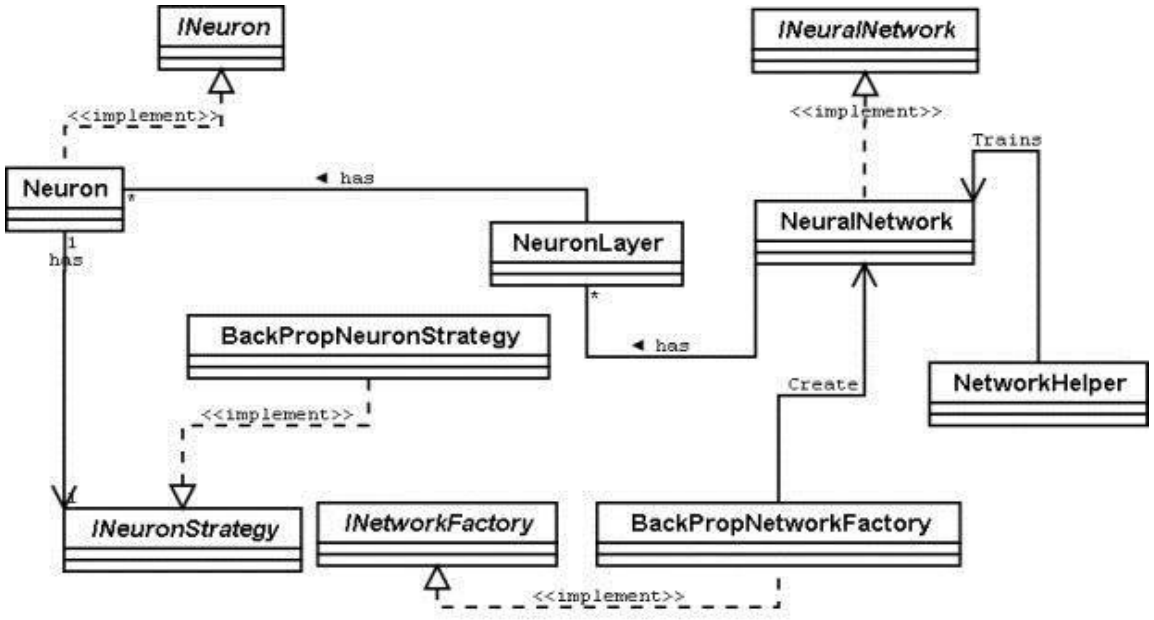

Рисунок 3.1 – Структура класу для розробки нейронної мережі

Оскільки структура нейронної мережі може змінюватися, а коефіцієнти клієнтського рівня залежать від сервера, було вирішено виділити окремий клас BackPropNetworkFactory. Ця назва класу пов'язана з тим, що в системі існує 2 мережевих підходи для різних елементів системи. Оскільки мережа, яка створюється, є нейронною, для її навчання використовується алгоритм зворотного поширення.

Цей клас відповідає за створення нейронної мережі з необхідними коефіцієнтами після їх завантаження на початковому етапі гри. У NetworkHelper, який відповідає за навчання нейронної мережі, є своя реалізація на стороні сервера і є по суті інтерфейсом для надання інформації для навчання нейронних мереж.

Як видно з функціональної схеми навчання мережі, навчання мережі може проходити двома способами: автоматичним моделюванням і адаптацією ментором.

Метод автоматичної симуляції складний для сервера, але це ефективний спосіб визначити негайну винагороду за те чи інше рішення в рамках вибору дій для групи.

Оскільки вихідний рівень нейронної мережі складається з 3 нейронів, які повертають значення в діапазоні від -1 до 1, два з яких задають напрямок, а один режим переходу, можна визначити, що для групи (тобто для набору бойових підрозділів у межах зони) визначено 27 можливих рішень.

Оскільки вся територія поділена на 25 квадратів, кількість необхідних симуляцій збільшується пропорційно до кількості зон, зайнятих бойовими підрозділами.

Така кількість необхідних моделювань створює навантаження на сервер і перешкоджає ефективному використанню можливостей моделювання для повного навчання моделі. Тому навчання проводиться в симуляціях зі спрощеними режимами зонування для прискорення процесу [15]

Для навчання системи використовується веб-інтерфейс, який дозволяє розміщувати об’єкти на території, передавати їх на аналіз, покроково переглядати результати алгоритму обробки та коригувати рішення алгоритму. Такий спосіб навчання нейронної мережі набагато швидший, оскільки правильний варіант надається в готовому вигляді і не вимагає додаткових ресурсів.

Навчання самої мережі − це, по суті, зміна коефіцієнта, що відбувається за типовими алгоритмами. На відміну від людського сприйняття, нейронні мережі для комп’ютера по суті представлені як набір значень у матричній формі, і всі дії над цими значеннями також відбуваються в рамках лінійної алгебри.

Однак платформа .NET не має вбудованого алгоритму роботи з матрицями. Для забезпечення ефективної та швидкої роботи з матрицями використовувалася бібліотека Math.NET Numerics.

Це простий і потужний інструмент, який ефективно працює з матрицями. Щоб підключити його до проекту, слід завантажити пакет Nuget або додати файл бібліотеки DLL і використати наступний код у файлі для підключення:

- з MathNet.Numerics.LinearAlgebra;

- з MathNet.Numerics.LinearAlgebra.Double

При розробці тестового середовища було виділено деякі моменти, які особливо важливі при розробці тестового середовища. Перша з них – це структура керування камерою (рис 3.2)

Для управління камерою був створений окремий скрипт CameraMovement.cs, який відповідав за всі рухи камери. Особливістю цього скрипта є адаптація рухів до сенсорних пристроїв. Це дозволяє ефективно використовувати цю модель камери навіть під час ігор на мобільних пристроях [15]

Для спрощення управління камерою на багатьох рівнях кожен з них був розділений на управління певним об'єктом.

Рисунок 3.2 – Налаштування камери в тестовому середовищі

Керування рухами камери здійснюється через базовий об’єкт (виділений на малюнку). Код, написаний для управління, досить великий і включає такі функції: [15]

- Рух

- Збільшити, зменшити

- Поворот

Кожен з варіантів реалізується окремим методом. Нижче наведено приклад реалізації методу ротації:

```
private void MobileCameraRotate()
    {
        if (!rotate) return;         if (Input.touchCount > 1)
        {
            Touch touch0 = Input.GetTouch(0);              Touch
touch1 = Input.GetTouch(1);              float deltaRotate;
            Touch leftTouch = new Touch();              Touch
rightTouch = new Touch();
            if (touch0.position.x < touch1.position.x)
            {
```

```
                leftTouch = touch0;
                rightTouch = touch1;
            }
            else if (touch0.position.x > touch1.position.x)
            {
                leftTouch = touch1;
                rightTouch = touch0;
            }

            if (leftTouch.deltaPosition.y >= 0 &&
rightTouch.deltaPosition.y <= 0)
            {

deltaRotate=((Mathf.Abs(leftTouch.deltaPosition.y)    +
Mathf.Abs(rightTouch.deltaPosition.y)) /
                                Screen.height) *
                               1080;
                rotor.transform.Rotate(0, -deltaRotate * smoot *
mobileRotSpeed, 0);
            }
            else if (leftTouch.deltaPosition.y <= 0 &&
rightTouch.deltaPosition.y >= 0)
            {
                deltaRotate  =
     ((Mathf.Abs(leftTouch.deltaPosition.y) +
Mathf.Abs(rightTouch.deltaPosition.y)) /
                                Screen.height) *
                               1080;
                rotor.transform.Rotate(0, deltaRotate * smoot *
mobileRotSpeed, 0);
        }
    }
```

Крім управління камерою, при розробці системи великий акцент був зроблений на створенні зручного інтерфейсу користувача.

Для його створення в Unity було використано елементи інтерфейсу користувача, вбудовані в ігровий движок. За компонуванням інтерфейс був розділений на 5 умовних частин, але з точки зору ігрового движка було побудовано два типи взаємодії - з інтерфейсом і з ігровим полем.

Багато елементів інтерфейсу користувача тісно перекриваються елементами на дошці, вибір ігрового об’єкта впливає на інтерфейс користувача, а використання інтерфейсу користувача може впливати на дошку. Така досить складна система реалізує зручний функціонал і забезпечує комфортний ігровий процес, оскільки при необхідності можна використовувати різні, більш зручні способи віддачі команд.

Всі команди, які можна задавати в грі, зведені в зрозумілу структуру. Це необхідно, тому що навички гравця і бота повинні бути зрівняні, тому що якщо гравець має більше інструментів управління, бот не зможе йому протистояти, оскільки він не може видавати певні команди. Кожен з ордерів має свій код, що дозволяє зручно синхронізувати дані з сервером і передавати їх між гравцями.

Модуль контролера використовується для отримання та надсилання даних, зв’язку з модулем веб-запитів та інших функцій керування в процесі гри.

Структурно контролери поділяються на дві частини – контролер, який відповідає за управління підрозділами, і контролер, який відповідає за управління будівлями. Однак для синхронізації взаємодії ці контролери обмінюються інформацією.

Основний функціонал у контролерах викликається у функції оновлення. Це викликано функціональністю компонентів, які успадковують клас MonoBehaviour. Життєвий цикл (повний) компонентів на рисунку 3.3.

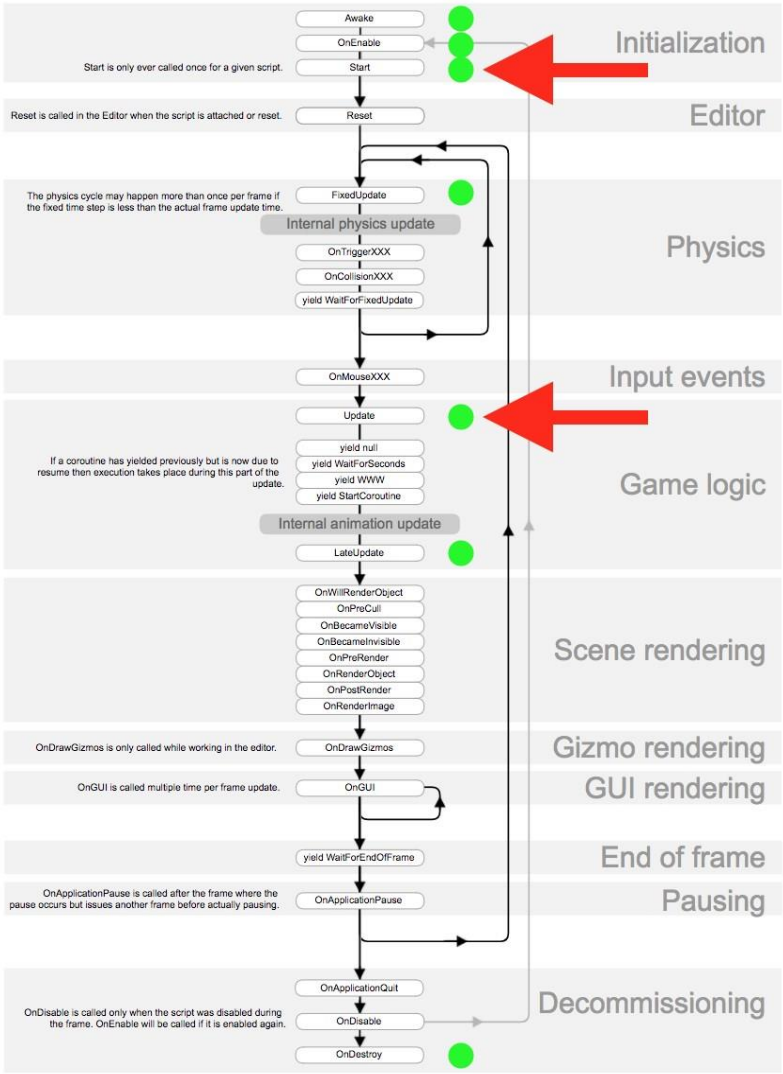

Рисунок 3.3 – Життєвий цикл в Unity

На рисунку методи, доступні для використання в сценаріях, виділені зеленим кольором. Коли до цих методів додається логіка, Unity виконує їх у фазі життєвого циклу, до якої належить кожен метод.

Червоні стрілки вказують на методи, створені за замовчуванням у всіх класах, що успадковують MonoBehaviour. Вигляд створеного сценарію можна побачити на рисунку 3.4.

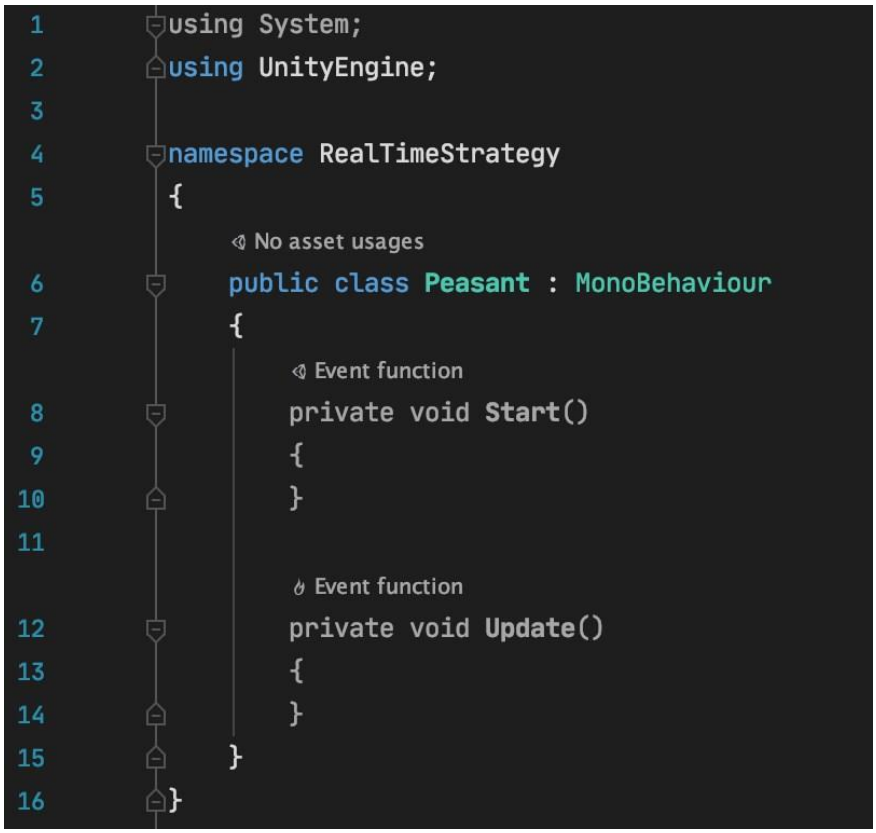

Рисунок 3.4 – Порожній клас (скрипт) в Unity

З контролерами дії, які виконує користувач, безпосередньо впливають на логіку гри. Відповідно до схеми, метод оновлення є першим методом, який активується після того, як вводиться кожен кадр.

Суть контролерів полягає в тому, що в кожному кадрі перед рендерингом він перевіряє, які дії виконав гравець, а що потрібно змінити (на рисунку 3.5 видно, що рендеринг відбувається після попередніх етапів). Тому метод Update є найбільш логічно завантаженим об'єктом класу, в якому використовуються практично всі методи.

```
private void Update()
{
    if (!EventSystem.current.IsPointerOverGameObject() && Input.GetMouseButtonUp(0) && Input.touchCount == 0)
    {
        var isDoubleClick = Time.time - _lastClickTime < CatchTime;
        Debug.Log( message: "Catch Double Click: " + isDoubleClick);
        var ray = camera.ScreenPointToRay(Input.mousePosition);
        if (!MultiSelect && Physics.Raycast(ray, out var hit,  maxDistance: Mathf.Infinity, unitLayerMask))
        {
            HandleUnitClick(hit, isDoubleClick);
        }
        else if (!MultiSelect && Physics.Raycast(ray, out var hitGround,  maxDistance: Mathf.Infinity, terrainLayerMask))
        {
            HandleMoveToPoint(hitGround);
        }
        else if (MultiSelect && Physics.Raycast(ray, out var selectGround :RaycastHit ,  maxDistance: Mathf.Infinity, terrainLayerMask))
        {
            HandleMultiSelection(selectGround);
        }
        _lastClickTime = Time.time;
    }
}
```

Рисунок 3.5 – Метод оновлення в UnitController

По-перше, метод перевіряє, що клацання не відбулося над елементом інтерфейсу користувача (таким чином надаючи контроль виключно елементам інтерфейсу користувача, незалежно від того, що знаходиться «внизу» на ігровому полі). Після проходження тесту метод ScreenPointToRay створює промінь від об’єкта камери в напрямку, куди вказує камера. Далі йде серія перевірок на перетин певних ігрових елементів - бойових одиниць або ігрового поля [15]

Коли вибрано бойову одиницю, для обробки клацання по ній викликається наступний метод (Рис 3.6). В інших випадках також викликаються спеціальні обробники.

Також у цій частині коду можна побачити використання протоколів у консолі Unity. Статичний клас Debug дозволяє викликати статичний метод Log, який дозволяє виводити потрібне повідомлення на консоль. Окрім Debug, варто згадати також використання класу Time. Цей клас відповідає за поточні таймери в екосистемі Unity.

```
private void HandleUnitClick(RaycastHit hit, bool isDoubleClick)
{
    var unit = hit.transform.gameObject.GetComponent<IUnit>();
    if (unit != null)
    {
        var cUnit = _allUnits.Find( match: x :ControlledUnit => unit.Id == x.UnitId);
        var alreadySelected :bool = _controlledUnits // List<ControlledUnit>
            .FirstOrDefault( predicate: x :ControlledUnit => x.UnitId == cUnit.UnitId) != null;
        if (!_controlledUnits.Any())
        {
            AddUnitToSelection(cUnit);
        }
        else if (alreadySelected)
        {
            UnselectAll();
            Debug.Log( message: ("cUnit.LastSelectedTime; " + cUnit.LastSelectedTime));
            if (isDoubleClick && Time.time - cUnit.LastSelectedTime <= CatchTime)
            {
                AddUnitsToSelection(_allUnits.Where(x :ControlledUnit =>
                    x.Unit.GetTypeName() == unit.GetTypeName() &&
                    Vector3.Distance( a: x.Object.transform.position, b: cUnit.Object.transform.position) < 20));
            }
            else
            {
                AddUnitToSelection(cUnit);
            }
        }
        else
        {
            UnselectAll();
            AddUnitToSelection(cUnit);
        }
    }
}
```

Рисунок 3.6 – Процес вибору бойової одиниці

Для синхронізації з сервером створено два окремих модуля, пріоритетна синхронізація під час гри здійснюється в класі SignalRConnector.

В цілому схема роботи в Unity з бібліотекою така ж, як і в веб-додатках, оскільки повна сумісність дозволяє повноцінно використовувати клієнт для взаємодії з сервером. На клієнті повідомлення від сервера обробляються за шаблоном Subscriber, коли методи всередині сценарію підписуються на події отримання даних від сервера.

3.2 Створення серверного компоненту

Під час розробки робочого прототипу проекту була створена схема бази даних для кожного мікросервісу, крім мікросервісу аналітики та штучного інтелекту.

Структура бази даних обширна, оскільки необхідно описати весь процес гри. Центральною таблицею в структурі є GameHistory, яка зберігає дані кожного успішного або невдалого запиту на створення нової гри. Майже всі таблиці в базі даних мікросервісу безпосередньо пов’язані з цією таблицею. Помилки під час гри записуються в окрему таблицю GameProcessIssues. Сторона, яка бере участь у грі, незалежно від її взаємодії, зберігається в таблиці ConflictSides. Коли гравець грає, поле PalyerID у таблиці заповнюється.

Ігрові дані про об’єкти та їх властивості зберігаються в таблицях WarriorLibrary і BuildingLibrary. Усі ігрові події та команди також класифіковані, усі дані формалізовані у вигляді таблиць ActionType та EventType [15]

Через специфіку архітектури мікросервісу деякі поля, такі як: B. PalyerID, який на перший погляд має бути зовнішнім ключем, не впливає на таблиці, оскільки дані зберігаються в іншій службі. Зв’язування даних відбувається в компоненті попередньої обробки даних (API Gateway), а синхронізація − через брокер повідомлень (Service Bus).

Для розробки бази даних використовувався підхід «перший код» [17]. Цей підхід полягає у використанні моделей, написаних на C#, для створення таблиць і бази даних загалом. Перевагою цього підходу є те, що для розробки використовується лише C#. Моделі для отримання даних повинні бути створені в кожному випадку, але при використанні іншого підходу ту саму роботу потрібно виконати двічі: створити моделі та створити таблиці в базі даних.

Однак цей підхід має певні недоліки: при такому підході швидкість роботи з базою даних дещо нижча, оскільки запити до сховища даних здійснюються через LINQ to Database. LINQ (Language-Integrated Query) − це частина мови програмування C#, яка забезпечує комплексний і практичний підхід до створення запитів для будь-якого збору даних. LINQ також

допомагає спростити маніпулювання та обробку даних. У певному сенсі це як розширена версія SQL, вбудована в C#.

Запити, написані в LINQ to Database для отримання даних із бази даних, перекладаються на SQL. Цей процес залежить від драйвера, який використовується у рамках, який надає доступ до даних. Драйвер не підтримує всі сценарії, що робить SQL більш функціональним у деяких випадках. Крім того, під час написання запитів LINQ драйвер часто не перекладає їх оптимальним чином, що знижує продуктивність порівняно зі сценаріями, написаними вручну.

Щоб використовувати LINQ, потрібно спочатку пов’язати дані моделі з таблицями в базі даних. Незважаючи на те, що таблиці створюються з моделей, потрібно додати їх до файлу налаштувань, щоб з’єднати їх. Для налаштування необхідно підключити базу даних до проекту. Додавання служб і модулів виконується у файлі запуску, який використовується для створення веб-хостингу у виконуваному файлі. Ця структура є стандартною для проектів, розроблених за допомогою .NET Core версії 2 і пізніших. У файлі запуску підключено все проміжне програмне забезпечення, веб-сокети та обробники запитів. Якщо веб-додатку потрібно використовувати сторонню базу даних або модуль, його потрібно додати до колекції служб, яка ініціалізується перед першим запуском хосту [15]

Щоб підключити базу даних до класу Startup, потрібно використовувати метод AddDbContext. Цей метод застосовується до IServiceCollection і додає сховище даних. AddDbContext − це загальний метод, у якому тип, переданий у метод, є класом, який успадковує базовий клас для контексту бази даних DbContext.

Цей файл ініціалізує всі налаштування підключення до бази даних. Кожна з таблиць підключена як DbSet. Приклад підключення таблиці «Будівлі»:public DbSet<Building> Buildings{ get; речення; }

Ця властивість класу є загальною властивістю, в яку переноситься клас моделі. Відповідно до параметрів Entity Framework за замовчуванням, усі

властивості, які є класами автоматичного зіставлення, додаються до SQL-запиту, який створює таблицю. Якщо потрібні спеціальні налаштування відображення полів, використовується анотація.

Деякі параметри анотації при створенні класів моделі [8]:

– Ключове поле: [Ключ];

– Поле із забороненими нульовими значеннями: [Обов'язкове];

– Поле, яке не додається до моделі: [NotMapped]; – поле, яке задає зовнішній ключ для об’єкта та використовується для прив’язки даних: [ForeignKey("BuildingId")];

- Поле синхронізації даних: [Timestamp]; − Уточнення функцій для створення поля в таблиці бази даних: [Column(TypeName = "decimal(20, 10)")]. У цьому випадку кількість символів при створенні стовпця в таблиці бази даних обмежена.

Висновок до розділу 3

У розділі 3 даної роботи "Розробка інфраструктури гри з нейронною мережею з елементами штучного інтелекту" детально розлянуто процес створення необхідної інфраструктури проекту. Головною метою цього розділу було створення середовища, яке б дозволило проводити моделювання, тестування та валідацію гри, оснащеної нейронною мережею та елементами штучного інтелекту.

Першим кроком було створення середовища для моделювання, що включало в себе всі необхідні компоненти для реалізації гри. Це включало в себе створення інтерфейсу гри, управління гравцями та всіма іншими аспектами, необхідними для дослідження.

Другим важливим кроком було створення серверного компоненту. Сервер відігравав ключову роль у взаємодії між гравцями та нейронною мережею з елементами штучного інтелекту. Він відповідав за обробку запитів гравців, розподіл ресурсів і інформації, необхідної для гри.

Завдяки розробці цих двох складових − середовища для гри та серверного компоненту, була можливість створити функціональну інфраструктуру для даного проекту. Ця інфраструктура дозволила ефективно моделювати та тестувати гру з нейронною мережею, що включає елементи штучного інтелекту. Вона стала важливим інструментом для дослідження та розвитку гри з інтелектуальною системою.

4 МОДЕЛЮВАННЯ ТА ТЕСТУВАННЯ СТРАТЕГІЧНОЇ ГРИ РЕАЛЬНОГО ЧАСУ ЗА ДОПОМОГОЮ ТАКТИЧНОЇ СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ І РЕЗУЛЬТАТИ

4.1 Моделювання та тестування тактичної системи штучного інтелекту

CameraMovement.cs є важливим скриптом, який відповідає за управління рухами камери у грі. Однією з особливостей цього скрипта є його адаптованість до сенсорних пристроїв, що робить його ефективним для використання в іграх на мобільних пристроях. Це важливо, оскільки ігри на мобільних пристроях стають все популярнішими.

Управління камерою на різних рівнях гри розділено на управління окремими об'єктами, що спрощує процес і дозволяє краще керувати камерою в різних сценах гри.

Код для управління рухами камери включає такі функції:

- Рух
- Збільшення та зменшення масштабу
- Поворот

Кожна з цих функцій реалізована окремим методом у скрипті.

Цей метод відстежує рухи користувача на сенсорному екрані та застосовує відповідні обертання до об'єкта "rotor", що відповідає за рух камери.

Основний функціонал контролерів викликається у функції "оновлення" (Update). Ця функція викликається на кожному кадрі гри і відповідає за перевірку дій гравця та зміну логіки гри відповідно до цих дій. Важливою особливістю цієї функції є те, що вона активується після того, як користувач вводить кожен кадр гри.

Контролери дійсно відіграють ключову роль у грі і відповідають за відгук гри на дії гравця. Метод Update, який зазвичай викликається кожен кадр, має важливе значення, оскільки в ньому відбувається перевірка та обробка майже всіх дій гравця.

У методі Update виконуються такі операції, як зчитування вхідних даних гравця (таких як натискання клавіш або рух миші), обробка цих даних і оновлення стану гри (таких як зміна положення гравця або стану об'єктів в грі). Результати цих операцій можуть відображатися на екрані або впливати на подальший хід гри.

Оскільки метод Update викликається кожен кадр, він забезпечує гладку та швидку реакцію гри на дії гравця. Це важливо для забезпечення зручності та задоволення гравця.

Проте, важливо пам'ятати, що метод Update може впливати на продуктивність гри, особливо якщо він містить складні обчислення або викликається багато разів за кадр. Тому слід бути уважним під час реалізації логіки гри в цьому методі і уникати зайвого навантаження.

4.2 Результати моделювання та тестування

Структура навчання системи виявилася проблематичною, оскільки, на відміну від типових завдань, для вирішення яких використовуються нейронні мережі, системі не можна дати чітко «правильний» варіант. Однак функціональна система, яка забезпечує можливість отримання даних про поточний стан військ, дозволяє моделювати ситуації з різними варіантами і надавати системі «правильний» варіант на основі розрахунку максимальної кількості балів. Це ускладнює навчання моделі для початкового завантаження, але дозволяє зробити правильні висновки. Також додана можливість коригувати дії шляхом вказівки правильного результату.

Основним показником вимірювання ефективності системи була оцінка ефективності застосування бойових частин. При зіткненні бойових підрозділів

підраховувався загальний залік угруповання військ, яке увійшло в зону зіткнення. Після зміни зони конфлікту переміщення фокуса в іншу точку на карті або знищення всіх бойових підрозділів однієї зі сторін підраховується загальний бал. Як правило, цей рахунок є різницею втрат гравців у поточній зоні. Якщо втрати менші, то дії в зоні були успішними. Ефективність застосування військ при використанні такої метрики може бути врахована навіть у разі повної втрати бойових частин на території, оскільки вони можуть завдати бойовим частинам противника більше шкоди, ніж завдати. Ефективність дій також можна визначити за значенням рейтингу - чим він вищий, тим вище різниця між втратами гравця і суперника, а значить, юніти використовувалися максимально ефективно [15]

Для тестування було використано кілька типових заздалегідь визначених алгоритмів, які зазвичай використовуються в стратегіях. Це типи ботів із традиційними назвами «атакуючі», «захисні» та «збалансовані».

Далі в таблиці 1 наведено порівняльний опис результатів боїв ботів під керуванням нейронної мережі та одного з ботів, які мають певний алгоритм.

Для визначення ефективності навчання було створено 3 варіанти моделі. Перший дізнавався за результатами тестувальників («правильний» варіант відповіді нейромережі дала одна людина). Другий варіант викладався виключно шляхом моделювання та підрахунку оцінок. Третє – це комбінація даних навчання, які використовуються.

Таблиця 1 – Результати тестів

	Атака	Захист	Баланс
1	67%	61%	64%
2	72%	65%	71%
3	75%	66%	73%

При аналізі результатів стає зрозуміло, що ефективність нейронної мережі досить висока в порівнянні зі стандартними ботами навіть при невеликих обсягах тестових даних. Проти агресивного агента ефективність вище, тому що отримати перевагу в бою набагато простіше − бот не чекає підкріплення і втрачає одиниці невеликими групами. Нижчі результати з захисним ботом пов’язані з тим, що він часто відступає за межі зони зіткнення, оскільки потрібна більша маса військ або відсутня стратегічна точка захисту. Вихід із зони не змінює значення бойових частин, тому перемоги немає.

Всього для оцінки було проведено 10 ігор. У кожній із партій кількість боїв була різною. Цікавою є також залежність кількості боїв у матчі від тренованості нейронної мережі. На початку нейронна мережа заповнюється випадковими коефіцієнтами, результат відповідний − швидка поразка через помилки управління. У міру тренування нейронної мережі кількість зіткнень − тактичних активацій штучного інтелекту − за матч зростала, оскільки до кінця бою проходило більше часу завдяки більшій ефективності. Кількість зіткнень за гру почала зменшуватися, так як висока ефективність використання зіткнень призводила до великих втрат бойових одиниць, на відновлення яких потрібен деякий час [15]

Отже , можна зробити висновок, що найкращим варіантом є поєднання навчання за результатами моделювання та навчання за техногенними прикладами. Проблема навчання лише через симуляцію полягає в тому, що людські зразки для наслідування вже давно мають дещо загальніше уявлення про зображення та розрахунок кроків.

Ефективність штучного інтелекту можна підвищити, навчивши мережу на більшій кількості даних, а алгоритм моделювання можна вдосконалити, щоб повністю автоматизувати процес.

Висновки до розділу 4

Під час моделювання та тестування тактичної системи штучного інтелекту було проведено обширні дослідження щодо її ефективності та можливостей. Основною метою було розроблення системи, яка могла б забезпечити стратегічну гру в реальному часі на високому рівні. Під час тестування були використані різні сценарії та умови гри для оцінки роботи системи.

Результати дослідження свідчать про те, що розроблена тактична система штучного інтелекту демонструє вражаючі можливості у грі в реальному часі. Вона здатна адаптуватися до різних гральних ситуацій, приймати стратегічні рішення та виконувати їх у найкращий спосіб. Ця система виявила високий рівень інтелектуальної самостійності та аналітичних здібностей, що дозволяє їй конкурувати з людьми в грі.

Крім того, під час тестування було виявлено, що система здатна навчатися в режимі реального часу. Вона вдосконалює свої стратегії на основі навчання з кожного ігрового досвіду, що робить її ще більш ефективною та конкурентоспроможною.

ВИСНОВКИ

Дана магістерська робота присвячена дослідженню та оптимізації впровадження штучного інтелекту в ігрових застосунках, представляє важливий внесок у сферу ігрової індустрії та розробки ігор. Результати дослідження, представлені в цій роботі, надають докладний аналіз різних аспектів використання нейронних мереж з елементами штучного інтелекту для оптимізації геймплею в стратегічних іграх реального часу.

В ході дослідження була розроблена класифікація стратегічних ігор реального часу, що дозволяє краще розуміти різноманітність цього жанру і визначати специфічні завдання, які можуть бути вирішені за допомогою нейронних мереж. Також було визначено ключові завдання, які передбачається вирішувати нейронним мережам для покращення ігрового процесу та підвищення інтелектуальної складності гри.

Одним з найважливіших досягнень цієї роботи є розробка методів для удосконалення характеристик гри на основі нейронної мережі з елементами штучного інтелекту. Ці методи дозволяють оптимізувати ігровий процес, роблячи його більш цікавим та імерсивним для гравців. Використання нейронних мереж для аналізу інформації та прийняття рішень в режимі реального часу відкриває нові перспективи для розробників ігор.

Крім того, було створено спеціальне середовище для проведення моделювання, тестування та валідації гри. Це середовище дозволяє проводити дослідження та експерименти з використанням розроблених методів та перевіряти їхню ефективність у практичних умовах.

Дана магістерська робота має велике значення для розвитку галузі ігрової індустрії, оскільки вона пропонує конкретні методи та інструменти для оптимізації геймплею в стратегічних іграх реального часу за допомогою

штучного інтелекту. Результати дослідження можуть бути використані розробниками ігор для створення більш цікавих та виразних ігрових досвідів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Aha D. W., Molineaux M., Ponsen M. Learning to win: case-based plan selection in a real-time strategy game. Case-Based reasoning research and development. Berlin, Heidelberg, 2005. С. 5–20. URL: https://doi.org/10.1007/11536406_4 (дата звернення: 15.11.2023).

2. Cadena P., Garrido L. Fuzzy case-based reasoning for managing strategic and tactical reasoning in starcraft. Advances in artificial intelligence. Berlin, Heidelberg, 2011. С. 113–124. URL: https://doi.org/10.1007/978-3-642-25324-9_10 (дата звернення: 18.11.2023).

3. Churchill D., Saffidine A., Buro M. Fast heuristic search for RTS game combat scenarios. Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment. 2021. Т. 8, № 1. С. 112–117. URL: https://doi.org/10.1609/aiide.v8i1.12527 (дата звернення: 17.11.2023).

4. Jasper Laagland. A HTN planner for A real-time strategy game. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=2e991c2c94cb53d2db144a5e869e781dce69c443 (дата звернення: 17.11.2023).

5. Inferring strategies from limited reconnaissance in real-time strategy games / J. Hostetler et al. Citeseerx, 15–17 August, 2012. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c8dfaa08c179ebf6b7e51ffd8af94babaf2d4da3 (дата звернення: 22.11.2023).

6. McCorduck P. Machines who think. Monoskop. URL: https://monoskop.org/images/1/1e/McCorduck_Pamela_Machines_Who_Think_2nd_ed.pdf (дата звернення: 19.11.2023).

7. Pillai A. Designing and implementing A neural network library for handwriting detection, image analysis etc.- the brainnet library - full code, simplified theory, full illustration, and examples. CodeProject - For those who code. URL: https://www.codeproject.com/Articles/14342/Designing-And-Implementing-A-Neural-Network-Librar (дата звернення: 23.11.2023).

8. Molineaux M., Klenk M., W. Aha D. Goal-Driven autonomy in a navy strategy simulation. Cdn.aaai.org. URL: https://cdn.aaai.org/ojs/7576/7576-13-11106-1-2-20201228.pdf (дата звернення: 19.11.2023).

9. New Rider Champandard AI Game. 2003 p 134-156 (дата звернення: 20.11.2023).

10. Piscataway, NJ: Institute for Electrical and Electronics Engi- neers, 2009 (дата звернення: 20.11.2023).

11. Morris R. Workshop summary kasparov vs. big blue: the significance for artificial intelligence. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ed52810365a27466c77a404321b3ecb66df7f3da (дата звернення: 19.11.2023).

12. Buro M. Call for AI Research in RTS Games. Proceedings of the AAAI Workshop on AI in Games. AAAI Press, 2004 (дата звернення: 23.11.2023).

13. Hierarchical reinforcement learning for real-time strategy games / R. Niel et al. SURFconext. URL: https://pure.rug.nl/ws/portalfiles/portal/61781086/ICAART_2018_66.pdf (дата звернення: 19.11.2023).

14. Muska and Lipman. AI Techniques for Game Programming. Buckland 2002 p. 94-108 (дата звернення: 23.11.2023).

15. Комп'ютерні системи штучного інтелекту. Методичні вказівки до виконання лабораторних робіт студентами денної та заочної форми навчання спеціальностей 123 «Комп’ютерна інженерія», 122 «Комп’ютерні науки та інформаційні технології» / Укл.: Є.В. Мелешко – Кропивницький: ЦНТУ, 2016. – С.8-13 (дата звернення: 20.11.2023).

16. Schwab. AI Game Engine Programming. Charles River Media. 2004. p 9-68 (дата звернення: 19.11.2023).

17. Shantia Amirhossein, Marco A. Wiering. Connectionist reinforcement learning for intelligent unit micro management in starcraft. ResearchGate. URL: https://www.researchgate.net/profile/Marco-Wiering/publication/224260306_Connectionist_reinforcement_learning_for_intelli

gent_unit_micro_management_in_StarCraft/links/09e4150b09820e80f7000000/Connectionist-reinforcement-learning-for-intelligent-unit-micro-management-in-StarCraft.pdf?origin=publication_detail&_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uRG93bmxvYWQiLCJwcmV2aW91c1BhZ2UiOiJwdWJsaWNhdGlvbiJ9fQ (дата звернення: 22.11.2023).

18. Gabriel Synnaeve, Pierre Bessiere. A bayesian model for RTS units control applied to starcraft. Researchgate. URL: https://www.researchgate.net/profile/Pierre-Bessiere-2/publication/261086581_A_Bayesian_Model_for_RTS_Units_Control_applied_to_StarCraft/links/00b7d53788c9bad85c000000/A-Bayesian-Model-for-RTS-Units-Control-applied-to-StarCraft.pdf?origin=publication_detail&_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uRG93bmxvYWQiLCJwcmV2aW91c1BhZ2UiOiJwdWJsaWNhdGlvbiJ9fQ (дата звернення: 20.11.2023).

19. Simple Hierarchical Ordered Planner. URL: http://www.cs.umd.edu/projects/shop/index.html (дата звернення: 22.11.2023).

20. Funge J. D. AI for animation and games: a cognitive modeling approach. A K Peters/CRC Press, 1999. 41-49 p (дата звернення: 22.11.2023).