ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика керування поведінкою штучного інтелекту ворогів у грі жанру "стратегія в реальному часі"»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми Інженерія програмного забезпечення

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Роман ОЛЕКСЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-63
Роман ОЛЕКСЕНКО

Керівник: Олесь ДІБРІВНИЙ
доктор філософії (PhD)

Рецензент: _____
науковий ступінь, вчене звання

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти Магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач Кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« » _____2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Олексенку Роману Григоровичу

1. Тема кваліфікаційної роботи: «Методика керування поведінкою штучного інтелекту ворогів у грі жанру "стратегія в реальному часі"»

керівник кваліфікаційної роботи Олесь ДІБРІВНИЙ, доктор філософії (PhD), затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024 р. № 320.

2. Строк Подання студентом роботи «17» грудня 2024 р.

3. Вхідні дані до роботи: науково-технічна література, алгоритм поведінки штучного інтелекту.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

1. Дослідження та аналіз ігор за темою.
2. Дослідження методів дій штучного інтелекту.
3. Реалізація методу дій штучного інтелекту.
4. Опис проектування системи.

5.Перелік ілюстративного матеріалу: *презентація*

1. Мета, об’єкта та предмет дослідження.
2. Актуальність роботи.
3. Порівняльний аналіз методів дій штучного інтелекту.
4. Представлення формули методу дій штучного інтелекту ворога.
5. Метод дій штучного інтелекту ворогів.
6. Екранна форма Mentisys.
7. Висновки.
8. Порівняльний аналіз методу дій.
9. Висновки.

6. Дата видачі завдання «16» жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфаційної роботи	Строк виконання етапів роботи	Примітка
1	Вивчення теми магістерської роботи	16.10-20.10.2024	
2	Аналіз поточних методів дій штучного інтелекту ворогів в ігровій індустрії	20.10-25.10.24	
3	Дослідження методів, механік і алгоритмів AI для жанру "Стратегія в реальному часі"	25.10-02.11.24	
4	Вивчення взаємодії штучного інтелекту з гравцем	02.11-08.11.24	
5	Огляд сучасних методів машинного навчання, застосовних для динамічного налаштування AI	08.11-12.11.24	
6	Розробка алгоритму дій штучного інтелекту залежно від дій гравця	12.11-22.11.24	
7	Оформлення роботи: вступ, висновки, реферат	22.11-27.11.24	
8	Розробка демонстраційних матеріалів	27.11-28.11.24	
9	Попередній захист роботи	29.11.2024	

Здобувач вищої освіти _____ Роман ОЛЕКСЕНКО

(підпис)

Керівник
кваліфікаційної роботи _____ Олесь ДІБРІВНИЙ

(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи 95 с., 2 табл., 11 рис., 30 джерел.

Мета дослідження – підвищення логічного мислення за рахунок розробки методу дій штучного інтелекту для ворогів в грі “жанрі стратегія в реальному часі”.

Об’єкт дослідження – процес взаємодії штучного інтелекту з гравцем у стратегіях в реальному часі.

Предмет дослідження – метод дій штучного інтелекту в грі жанру "стратегія в реальному часі".

Короткий зміст роботи: У цій роботі досліджено метод дій штучного інтелекту для стратегій у реальному часі. Проведено аналіз існуючих методів функціонування штучного інтелекту. Розроблено метод, який є адаптивним до дій гравця, простим у реалізації та малозатратним з погляду обчислювальних ресурсів. Здійснено порівняльний аналіз різних методів. Результати дослідження дозволили підвищити адаптивність ШІ з мінімізацією обчислювальних витрат.

КЛЮЧОВІ СЛОВА: МЕТОДИ ДІЙ ШТУЧНОГО ІНТЕЛЕКТУ, СТРАТЕГІЇ В РЕАЛЬНОМУ ЧАСІ, RTS ІГРИ, АДАПТИВНЕ УПРАВЛІННЯ ЮНІТАМИ, СТРАТЕГІЧНЕ ПЛАНУВАННЯ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ.

ABSTRACT

Text part of the qualification work 95 p., 2 tables, 11 figures, 30 references.

Purpose of the work – to enhance logical thinking by developing an action method for enemy artificial intelligence in real-time strategy games.

Object of research – the process of interaction between artificial intelligence and the player in real-time strategy games.

Subject of research – the method of artificial intelligence actions in real-time strategy games.

Summary of the work: Various approaches to developing artificial intelligence for real-time strategy games were explored, including adaptive unit control, strategic planning, and reinforcement learning. The advantages and disadvantages of each method were analyzed to develop an effective algorithm aimed at improving gameplay and providing engaging challenges for players.

KEYWORDS: ARTIFICIAL INTELLIGENCE ACTION METHODS, REAL-TIME STRATEGY, RTS GAMES, ADAPTIVE UNIT CONTROL, STRATEGIC PLANNING, REINFORCEMENT LEARNING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI – штучний інтелект (Artificial Intelligence).

RTS – стратегія в реальному часі (Real-Time Strategy).

HP – здоров'я юніта (Health Points).

AP – боєприпаси (Ammunition Points).

FSM – скінченний автомат (Finite State Machine).

BT – поведінкове дерево (Behavior Tree).

RL – підкріплювальне навчання (Reinforcement Learning).

GA – генетичні алгоритми (Genetic Algorithms).

BN – баєсові мережі (Bayesian Networks).

ML – машинне навчання (Machine Learning).

TS – статус території (Territory Status).

RT – резервні групи (Reserve Troops).

DAI – рішення штучного інтелекту (AI Decision).

ВСТУП

Гри в жанрі "стратегія в реальному часі" (RTS) здобувають надзвичайну популярність серед геймерської аудиторії, адже вони пропонують захопливий іммерсивний досвід управління військами та ресурсами в динамічних умовах. Одним з ключових аспектів, що визначає глибину ігрового процесу, є якість та ефективність штучного інтелекту (ШІ), що взаємодіє з гравцем. У сучасних умовах розробка вдосконаленого методу дій ШІ стає актуальною завдяки розширенню технологічних можливостей, зростанню очікувань гравців та постійній конкуренції на ринку відеоігор. Реалістичний та динамічний ШІ може покращити геймплей, зробити гру більш цікавою та додати елемент непередбачуваності.

Застосування в роботі методів адаптивного управління юнітами, стратегічного планування та навчання з підкріпленням відкриває перспективу для створення ефективного, реалістичного та адаптивного штучного інтелекту. Подальша розробка та оптимізація таких методів може значно збільшити привабливість гравців до ігор в жанрі "стратегія в реальному часі" та підвищити їхню конкурентоспроможність на ринку розважальних відеоігор.

Завдання роботи полягає в таких пунктах:

1. Дослідження та аналіз методів дій штучного інтелекту.
2. Реалізація методу дій штучного інтелекту.
3. Опис та проектування.
4. Порівняння створеного метода з існуючуми.

Мета дослідження – підвищення логічного мислення для дітей за допомогою стратегії в реальному часі за рахунок розробки методу дій штучного інтелекту.

Об’єкт дослідження – процес взаємодії штучного інтелекту з гравцем у стратегіях в реальному часі.

Предмет дослідження – методи розробки дій штучного інтелекту в іграх жанру "стратегія в реальному часі".

Практичне значення дослідження полягає в розробці ефективного методу дій штучного інтелекту для ігор жанру "стратегія в реальному часі" (RTS). Створений метод дозволяє AI динамічно адаптувати свої дії залежно від змін на полі бою, контролю територій та стану юнітів. Застосування цього методу забезпечує створення складніших, інтерактивних і стратегічно насичених ігрових ситуацій. Це сприяє підвищенню залученості гравців та створенню більш захопливого ігрового процесу, де складність автоматично регулюється залежно від дій гравця.

Наукова новизна полягає у створенні адаптивного методу дій штучного інтелекту, що забезпечує гнучкість і непередбачуваність рішень AI у реальному часі. Метод використовує стратегічне планування, управління ресурсами та територіями, а також навчання на основі взаємодії зі світом гри. Це вдосконалення дозволяє AI ефективніше реагувати на дії гравця, забезпечуючи баланс між викликом і задоволенням від гри.

Даний підхід підвищує конкурентоспроможність ігор на ринку, дозволяючи створювати ігрові сценарії з високим рівнем варіативності та інтелектуальної глибини, що позитивно впливає на утримання гравців і підвищення їхнього інтересу до гри.

1. АНАЛІЗ ТА ДОСЛІДЖЕННЯ

1.1 Поняття жанру та особливості розробки методу дій штучного інтелекту в іграх жанру "стратегія в реальному часі"

Ігри жанру "стратегія в реальному часі" (RTS) займають важливе місце в ігровій індустрії завдяки своїй динамічності, складності та необхідності приймати рішення миттєво. Гравці мають змогу керувати великими арміями, розбудовувати бази та управляти ресурсами в умовах постійного тиску з боку суперників. Цей жанр вимагає від гравця стратегічного планування, тактичного мислення та швидкого реагування на зміну ситуації на полі бою.

Основна унікальність RTS полягає у необхідності одночасного управління декількома аспектами гри: економікою, будівництвом, дослідженнями та військовими діями. Гравець повинен балансувати між цими елементами для досягнення перемоги. Успіх залежить від здатності швидко аналізувати ситуацію та приймати оптимальні рішення.

Важливим компонентом цих ігор є штучний інтелект (ШІ), який виконує функцію опонента або союзника гравця. Створення ефективного методу дій ШІ є ключовим викликом для розробників, адже ШІ повинен діяти не тільки логічно, а й непередбачувано, щоб забезпечити цікавий ігровий процес.

Особливості жанру та завдання ШІ у стратегіях в реальному часі (RTS) полягають у необхідності динамічного та адаптивного реагування на зміни на полі бою. Всі події відбуваються одночасно, що змушує і гравця, і ШІ приймати рішення в реальному часі. Це створює додаткове навантаження на алгоритм дій ШІ, який має не лише ефективно збирати ресурси та будувати бази, але й правильно оцінювати бойові умови, планувати розвиток та використовувати обмежені ресурси для досягнення стратегічних цілей. Завданням ШІ є створення збалансованої економіки для підтримки розвитку бази та армії, водночас він має прогнозувати майбутні дії гравця, що робить процес адаптації до різних стратегій

вкрай важливим. Це дозволяє йому коригувати свої дії відповідно до ситуації на полі бою, забезпечуючи високу гнучкість і непередбачуваність.

Метод дій ШІ в RTS включає кілька ключових компонентів, таких як постійний аналіз ситуації на полі бою. ШІ постійно збирає інформацію про стан гри, оцінюючи кількість ресурсів, наявність ворожих юнітів та територіальні зміни. На основі цих даних ШІ приймає рішення, визначаючи, чи варто зосередитися на обороні, розвитку чи нападі. Окрім того, він змінює свою поведінку в залежності від того, як грає гравець, що забезпечує гнучкість в прийнятті рішень та адаптацію до стилю гри опонента. Наприклад, якщо гравець обирає агресивну стратегію, ШІ може зосередитися на контратаках, а якщо гравець обирає економічний розвиток, ШІ може підвищити військовий тиск. Це дозволяє створювати цікаві та складні ситуації, де кожне рішення має значення, а сама гра постійно кидає нові виклики.

Прикладом стратегій ШІ в RTS можуть бути агресивні, оборонні та гібридні стратегії. У першому випадку ШІ намагається швидко знищити гравця, створюючи мінімальну кількість юнітів для ранньої атаки. В оборонній стратегії ШІ зосереджується на створенні укріплень та накопиченні ресурсів для пізнішої гри. Гібридна стратегія, в свою чергу, передбачає баланс між атакою та обороною, де ШІ намагається знайти слабкі місця в обороні гравця та використовувати їх для досягнення переваги.

Роль методу дій ШІ у створенні цікавого геймплею не можна недооцінювати. Хороше проектування ШІ створює постійний виклик для гравця, змушуючи його адаптувати свою стратегію залежно від поведінки комп’ютерного супротивника. Реалістичність, яку приносить логічно діючий ШІ, значно підвищує занурення в гру, а різноманітність стратегій, які застосовує ШІ, гарантує, що кожна гра буде унікальною. Тому метод дій ШІ є важливим етапом у розробці стратегій в реальному часі, надаючи гравцям можливість постійно вдосконалювати свої навички стратегічного мислення і гнучкості в умовах непередбачуваних ситуацій.

1.2 Аналіз існуючих ігор

1.2.1 Приклад застосування методу дій ШІ в грі Hearts of Iron IV

Hearts of Iron IV — це глобальна стратегія в реальному часі від компанії Paradox Interactive, що дозволяє гравцям керувати цілими державами під час Другої світової війни. Гра охоплює аспекти військової, економічної та політичної діяльності, де кожне рішення гравця може вплинути на хід історії. Основна мета гравця — вести свою країну до перемоги, використовуючи стратегії розвитку, дипломатії та військових дій.

Особливість Hearts of Iron IV полягає у масштабності управління та глибокій деталізації. Гравець може контролювати як стратегічні напрямки, так і окремі підрозділи, керувати виробництвом озброєння, дипломатією та ресурсами. Саме в таких умовах метод дій ШІ стає важливим елементом для забезпечення цікавого та збалансованого ігрового процесу.

Завдання ШІ в Hearts of Iron IV охоплюють три основні напрямки: управління економікою, тактичне управління військами та дипломатія з адаптацією до дій гравця. ШІ має ефективно керувати економікою, зважаючи на обмеженість ресурсів і необхідність балансування між виробництвом військової техніки та підтримкою інфраструктури. Він вирішує, скільки ресурсів спрямувати на виробництво зброї, авіації, флоту чи інших елементів, критично важливих для ефективного функціонування держави. У тактичному плані ШІ контролює пересування підрозділів, враховує рельєф місцевості та кліматичні умови, вибирає оптимальні маршрути для атаки чи оборони, а також ефективно використовує авіацію і флот для підтримки наземних операцій. Водночас він веде переговори з іншими державами, обирає стратегічних партнерів, укладає союзи, оголошує війни чи підписує мирні угоди. Окрему роль відіграє адаптація до дій гравця: ШІ аналізує поведінку гравця, його сильні та слабкі сторони, і коригує свої стратегії відповідно, наприклад, переходячи до наступу при зосередженні гравця на обороні або блокуючи ресурси.

Метод дій ШІ у грі складається з кількох ключових компонентів. Стратегічне планування включає визначення довгострокових цілей залежно від геополітичного положення країни, її економічних ресурсів і стратегічних потреб. ШІ може вибирати між агресивною експансією, зміцненням оборони чи дипломатичними угодами. Тактичне реагування зосереджене на оперативному управлінні військами, включаючи їхній рух, атаку чи захист ключових об’єктів, а також раціональне використання флоту й авіації. ШІ враховує особливості місцевості, кліматичні умови, наявність ресурсів і ситуацію на полі бою, приймаючи рішення в реальному часі. Моделювання логістики передбачає управління постачанням підрозділів, моніторинг рівня боєприпасів і пального. Це особливо важливо для забезпечення стабільності бойових дій, адже без ресурсів навіть найпотужніша армія стане неефективною.

Приклади стратегій ШІ у Hearts of Iron IV демонструють його здатність приймати рішення залежно від ситуації. У стратегії "Бліцкриг" ШІ концентрує великі сили на одному напрямку, швидко прориває оборону ворога і захоплює ключові об’єкти. У разі застосування оборонної стратегії він будує укріплення, захищає важливі точки і намагається виснажити ворога довготривалою війною. Якщо ШІ використовує стратегію морської блокади, він задіює флот для перекриття ворожих морських шляхів і припинення постачання ресурсів. Водночас ШІ динамічно адаптується до змін: якщо гравець займає нейтральну позицію, ШІ може зосередитися на інших конфліктах чи дипломатичному тиску, а у разі агресивної поведінки гравця — оперативно мобілізує війська для протидії.

Варто зазначити, що у грі є можливість налаштування рівня складності, додаючи переваги чи обмеження для ШІ. Це дозволяє зробити ігровий процес більш доступним для новачків або складнішим для досвідчених гравців. Завдяки адаптивним алгоритмам ШІ створює динамічний і непередбачуваний ігровий процес, який зберігає інтерес навіть після багаторазового проходження. Вміння ШІ реагувати на дії гравця, керувати економікою, тактикою та дипломатією

забезпечує унікальний досвід стратегічного планування та військових дій, що можна побачити на рисунку 1.1. Цей метод дій є ключовим елементом гри, підтримуючи баланс між викликом та задоволенням від ігрового процесу, що робить Hearts of Iron IV однією з найкращих стратегічних ігор свого жанру.

Рис. 1.1 Демонстрація геймплею в грі Hearts of Iron IV

1.2.2 Приклад застосування методу дій ШІ в грі StarCraft II

StarCraft II — одна з найвідоміших ігор жанру "стратегія в реальному часі" (RTS), розроблена компанією Blizzard Entertainment. Гра пропонує три унікальні раси — Терани, Протоси та Зерги, кожна з яких має свої особливості, юніти та стратегії. Гравці змагаються за контроль над ресурсами, будують бази та ведуть військові дії для досягнення перемоги над супротивником. Високий темп гри та необхідність приймати рішення в реальному часі роблять StarCraft II відмінним прикладом для аналізу методу дій ШІ.

Завдання ШІ в StarCraft II можна поділити на кілька ключових напрямків. Насамперед, ШІ відповідає за управління ресурсами, збираючи мінерали та газ,

які необхідні для будівництва бази, створення юнітів і розвитку технологій. ШІ також контролює будівництво та розвиток бази, приймаючи рішення про розташування будівель для підвищення ефективності оборони та можливості швидкого реагування на атаки. Управління військами є ще одним важливим завданням, що включає координацію рухів юнітів, вибір цілей для атаки чи захисту, враховуючи тип юнітів та їхню ефективність у бою. Важливим аспектом є адаптація до дій гравця: аналізуючи стратегію опонента, ШІ змінює свою тактику, обираючи відповідні контрзаходи.

Компоненти методу дій ШІ у StarCraft II інтегрують кілька ключових модулів. Стратегічний планувальник визначає загальний план дій, обираючи між агресивною, оборонною або гібридною стратегією, враховуючи расу ШІ та дії гравця. Тактичний менеджер відповідає за локальне управління юнітами під час бою, оптимізуючи їхні позиції для атаки чи відступу та враховуючи особливості рельєфу карти. Логістичний контролер забезпечує ефективний збір і розподіл ресурсів, розвиток технологій та створення нових юнітів. Модуль аналізу постійно стежить за діями гравця, адаптуючи стратегію ШІ. Наприклад, якщо гравець активно використовує авіацію, ШІ фокусує увагу на створенні протиповітряних юнітів.

Приклади стратегій, які використовує ШІ, демонструють його гнучкість і адаптивність. У стратегії Rush ШІ швидко створює мінімальну кількість юнітів, щоб атакувати базу гравця на початку гри, прагнучи завдати критичних втрат до того, як гравець розвинеться. У стратегії технологічного розвитку ШІ відкладає масові атаки, зосереджуючись на розвитку передових технологій і отриманні доступу до потужних юнітів. У стратегії контролю карти ШІ активно розширює свою присутність, захоплюючи нові бази та обмежуючи можливості гравця через контроль ресурсів.

Адаптивність ШІ у StarCraft II є ключовим аспектом його ефективності. ШІ постійно аналізує поведінку гравця, змінюючи свої дії залежно від ситуації. Наприклад, якщо гравець орієнтується на оборону, ШІ може застосувати

стратегію економічного розвитку або масові атаки з декількох напрямків. Крім того, ШІ враховує тип карти, стратегічні точки, вузькі проходи, що впливає на вибір тактики. Такий підхід робить гру динамічною і непередбачуваною, змушуючи гравця постійно адаптуватися.

Розробка методу дій ШІ стикається з низкою викликів. Одним із ключових є швидкість прийняття рішень, адже ШІ має миттєво реагувати на зміни на полі бою. Важливою є гнучкість, яка дозволяє змінювати стратегію залежно від стилю гри опонента. Також необхідна оптимізація використання ресурсів, що забезпечує баланс між економічним розвитком і військовою потужністю. Інтеграція цих факторів робить ШІ у StarCraft II потужним і складним супротивником.

Метод дій ШІ у StarCraft II має значний вплив на ігровий процес. Завдяки адаптивним алгоритмам ШІ створює відчуття реального супротивника, що змушує гравця розробляти складні стратегії та постійно адаптуватися. Динамічний ігровий процес тримає гравця в напрузі, створюючи захоплюючий досвід навіть після багаторазового проходження. StarCraft II є чудовим прикладом того, як продуманий метод дій ШІ може забезпечити цікаву і непередбачувану гру, що стимулює розвиток логічного мислення та стратегічного планування.

Метод дій ШІ у StarCraft II демонструє, як поєднання стратегічного мислення та тактичної гнучності може створювати інтерактивний ігровий досвід, який постійно тримає гравця в напрузі. Завдяки використанню адаптивних алгоритмів, які враховують дії гравця та умови на полі бою, ШІ здатний забезпечувати унікальність кожної партії, змушуючи гравців адаптувати свої стратегії та вдосконалювати навички. Цей підхід не лише сприяє створенню насиченого гемплею, але підвищує реіграбельність. На рисунку 1.2 показано гемплей StarCraft II.

Рис. 1.2 Демонстрація геймплею в грі StarCraft II

1.2.3 Приклад застосування методу дій ШІ в грі Company of Heroes 3

Company of Heroes 3 — це стратегія в реальному часі, яка поєднує глибокі тактичні битви з динамічним стратегічним управлінням. Дія гри відбувається під час Другої світової війни, а гравці беруть під контроль війська союзників або сил осі для ведення масштабних боїв. Унікальність гри полягає у поєднанні масштабних карт та детального управління військовими підрозділами, що створює складні виклики як для гравців, так і для ШІ.

Завдання ШІ в Company of Heroes 3 включають управління тактичними бойовими діями, взаємодію з оточенням, розподіл ресурсів і адаптацію до дій гравця. ШІ контролює загони та техніку, розподіляє їх на полі бою, обирає оптимальні позиції для атаки й оборони, враховуючи особливості укриттів і стратегічних точок. Крім того, він збирає ресурси та використовує їх для створення нових підрозділів, техніки чи розвитку бази. Особливу роль відіграє здатність ШІ адаптуватися до тактики гравця, швидко змінюючи свої дії залежно від ситуації, що дозволяє створити напружений і динамічний геймплей.

Компоненти методу дій ШІ є багаторівневими і охоплюють різні аспекти управління. Тактичний менеджер відповідає за контроль загонів на полі бою, планує флангові атаки, обирає укриття і координує підтримку артилерією. Модуль стратегічного управління приймає рішення щодо захоплення ключових точок, розподілу сил та розвитку бази, що дозволяє ефективно конкурувати з гравцем. Модуль аналізу оцінює дії суперника, виявляючи слабкі місця в обороні чи атаці, а логістичний контролер забезпечує поповнення підрозділів, розподіл ресурсів і виклик підкріплень. Разом ці компоненти створюють комплексну систему, яка імітує мислення справжнього командира.

Приклади стратегій ШІ демонструють його багатогранність і здатність адаптуватися до умов. Оборонна стратегія передбачає утримання ключових позицій, будівництво укріплень і застосування артилерії для відбиття атак. У випадку флангової атаки ШІ використовує маневри, щоб обійти основні сили гравця та завдати удару з несподіваних напрямків. Масоване наступлення дозволяє ШІ зосереджувати сили для прориву оборони, поєднуючи піхоту, техніку й авіацію. Завдяки цьому ШІ може ефективно діяти у різних умовах, враховуючи рельєф місцевості, можливості укриттів і стратегічних точок.

Виклики для розробки ШІ в цій грі пов’язані зі швидкістю прийняття рішень, тактичною гнучкістю та оптимізацією ресурсів. ШІ повинен миттєво змінювати тактику в залежності від дій гравця, ефективно використовувати укриття та ландшафт для зменшення втрат і грамотно розподіляти ресурси між розвитком бази та створенням нових підрозділів. Ці виклики вимагають складних алгоритмів, які б забезпечували баланс між тактичним управлінням і стратегічним плануванням.

Роль методу дій ШІ у створенні ігрового процесу Company of Heroes 3 є ключовою. Завдяки використанню адаптивних алгоритмів і можливості враховувати дії гравця, ШІ забезпечує реалістичність і динамічність боїв. Гравець постійно стикається з новими викликами, що змушує його змінювати свої стратегії й тактики. Це створює унікальний ігровий досвід, де кожна битва стає

непередбачуваною, а ШІ виступає як серйозний суперник, здатний випробувати навички та креативність гравця. На рисунку 1.3 показано гемплей Company of Heroes 3.

Рис. 1.3 Демонстрація геймплею в грі Company of Heroes 3

1.2.4 Приклад застосування методу дій ШІ в грі Warcraft III

Warcraft III — класична стратегія в реальному часі від Blizzard Entertainment, яка поєднує динамічний геймплей із глибоким сюжетом. Гра пропонує кілька фракцій, таких як Люди, Орки, Нічні ельфи та Нежить, кожна з яких має свої унікальні юніти, будівлі та магічні здібності. Основне завдання гравця — розвивати базу, збирати ресурси та створювати армію для боротьби з противником. У грі важливу роль відіграють герої, які можуть отримувати рівні та використовувати спеціальні здібності.

Завдання ШІ в Warcraft III охоплюють управління базою, збір ресурсів, керування армією та героями, а також адаптацію до дій гравця. ШІ контролює будівництво будівель, розвиток технологій і створення юнітів, забезпечуючи

ефективну оборону й атаку. Збір золота та дерева є ключовими завданнями для розвитку бази та формування армії. Управління армією передбачає координацію рухів юнітів, вибір цілей для атак і використання спеціальних здібностей героїв, зокрема магії та зіль, для максимального впливу на бій. Важливою складовою є адаптація до дій гравця: ШІ аналізує його стратегії, реагує на зміни ситуації та коригує свої дії відповідно.

Компоненти методу дій ШІ в Warcraft III включають кілька основних модулів. Модуль управління базою контролює будівництво, розвиток технологій і створення армії, враховуючи наявні ресурси та поточні потреби. Тактичний менеджер керує військами на полі бою, обираючи оптимальні моменти для атаки, відступу та використання спеціальних здібностей героїв. Модуль адаптації дозволяє ШІ змінювати стратегії залежно від дій гравця, переходячи від оборонних заходів до агресивних атак. Алгоритм пошуку шляхів (A*) забезпечує оптимальне пересування юнітів, враховуючи особливості місцевості та можливі перешкоди.

Приклади стратегій ШІ демонструють його різноманітність і здатність адаптуватися до умов гри. Rush-стратегія передбачає швидке створення базових юнітів для ранньої атаки на базу гравця, щоб завдати значних втрат і отримати перевагу. Стратегія розвитку героїв зосереджується на прокачуванні героїв, які відіграють ключову роль у битвах, забезпечуючи вирішальні переваги. Контроль карти включає захоплення стратегічно важливих точок, таких як шахти із золотом або об’єкти, що надають бонуси. Комбіновані атаки базуються на створенні збалансованих армій із різних типів юнітів, таких як піхота, маги та техніка, для досягнення максимального ефекту.

ШІ в Warcraft III здатний змінювати стратегії залежно від дій гравця, що додає гнучкості й викликів. Наприклад, якщо гравець будує оборонні споруди, ШІ може обійти їх, використати юнітів із високою атакою або застосувати магію для прориву оборони. Використання героїв додає ще один рівень складності, оскільки ШІ повинен оптимально використовувати їх здібності, розраховуючи час перезарядки та потребу в мані для підтримки армії.

Виклики для розробки методу дій ШІ охоплюють кілька ключових аспектів. Управління героями вимагає ефективного використання їхніх здібностей і ресурсів, таких як мана чи зілля. Тактична гнучкість потребує миттєвого реагування на зміну умов бою та дій гравця. Оптимальне використання ресурсів є критично важливим для збалансованого розвитку бази, створення армії та проведення досліджень.

Роль методу дій ШІ у забезпеченні ігрового процесу є вирішальною для створення насиченого та динамічного геймплею. ШІ у Warcraft III постійно змушує гравця адаптувати свої стратегії, використовувати всі доступні ресурси та змінювати тактики в залежності від ситуації. Завдяки гнучким алгоритмам ШІ забезпечує виклик навіть для досвідчених гравців, роблячи кожну битву унікальною та напруженою. Це сприяє глибшому зануренню в гру та тримає гравців у постійній зацікавленості. На рисунку 1.4 показано гемплей Warcraft III

Рис. 1.4 Демонстрація геймплею в грі Warcraft III

1.2.5 Приклад застосування методу дій ШІ в грі Козаки 3

Козаки 3 — це стратегія в реальному часі, розроблена українською студією GSC Game World. Гра переносить гравців у Європу XVII-XVIII століть, де вони керують арміями, розвивають економіку та ведуть масштабні битви. Головною особливістю гри є можливість створювати величезні армії, що складаються з тисяч юнітів, а також складна економічна система, яка вимагає управління кількома видами ресурсів.

Завдання ШІ в Козаки 3 охоплюють управління економікою, розвиток бази, створення армії, тактичне керування військами та адаптацію до дій гравця. ШІ здійснює збір ключових ресурсів, таких як золото, дерево, їжа, каміння та вугілля, розподіляючи їх для будівництва будівель, розвитку технологій і створення армії. Завдяки контролю за будівництвом ШІ забезпечує розвиток економічної бази й підготовку до атак. У сфері управління військами ШІ формує піхоту, кавалерію, артилерію та використовує їх для проведення атак, оборони або стратегічних маневрів. Тактичне керування враховує рельєф місцевості та розташування ворога, щоб досягти максимального ефекту в бою. Водночас ШІ постійно аналізує стратегію гравця, коригуючи свої дії залежно від ситуації на полі бою.

Компоненти методу дій ШІ в Козаки 3 включають кілька модулів, що забезпечують комплексний підхід до управління. Модуль управління економікою відповідає за збір і розподіл ресурсів, оптимізуючи їх використання для будівництва та створення військ. Тактичний менеджер займається переміщенням армій, організацією атак і оборони, зважаючи на особливості ландшафту та розташування ворога. Модуль стратегічного планування приймає рішення щодо захоплення ресурсних точок, оборони бази та атак на ключові об’єкти гравця. Модуль адаптації дозволяє ШІ підлаштовувати свої дії залежно від змін у стратегії гравця, ефективно протидіючи його тактиці.

Приклади стратегій ШІ в Козаки 3 демонструють його здатність до гнучкості та варіативності. Економічне домінування передбачає швидкий збір ресурсів і розвиток бази, що дає змогу створювати великі армії для переважання

у грі. Масовані атаки базуються на створенні значної кількості юнітів і проведенні широкомасштабних наступів, спрямованих на знищення оборони гравця. Оборонна стратегія фокусується на будівництві укріплень і захисті ключових об’єктів із подальшим накопиченням ресурсів для контратак. Флангові маневри полягають в обході оборони гравця з атакою з несподіваних напрямків, що завдає максимальних втрат супротивнику.

ШІ у Козаки 3 виявляє високий рівень адаптації до дій гравця. Наприклад, якщо гравець зосереджується на розвитку економіки, ШІ може обрати швидкі атаки для запобігання створенню потужної армії. Якщо гравець застосовує агресивну стратегію, ШІ переходить до оборонних дій, забезпечуючи стійкість своїх позицій і проводячи контратаки. ШІ також враховує масштаб битв і кількість юнітів, що дозволяє ефективно координувати дії великих армій, забезпечуючи взаємодію між піхотою, кавалерією та артилерією.

Виклики для розробки методу дій ШІ у Козаки 3 включають необхідність ефективного управління великими масами юнітів, розподілом ресурсів і тактичною гнучкістю. Масштаб управління вимагає від ШІ здатності координувати тисячі юнітів у реальному часі. Розподіл ресурсів повинен бути оптимізованим для збалансованого розвитку бази та підтримки армії. Тактична гнучкість ШІ є критично важливою для швидкого реагування на дії гравця та зміни умов бою.

Роль методу дій ШІ полягає у створенні насиченого ігрового процесу, що стимулює стратегічне мислення та змушує гравця використовувати різноманітні тактики. Завдяки адаптивним алгоритмам ШІ забезпечує унікальні сценарії бою, які зберігають інтерес навіть після багаторазового проходження гри. Таким чином, метод дій ШІ у Козаки 3 є важливим елементом, який робить гру цікавою, динамічною та захоплюючою для гравців будь-якого рівня майстерності. На рисунку 1.5 показано гемплей Козаки 3

Рис. 1.5 Демонстрація геймплею в грі Козаки 3

1.2.6 Приклад застосування методу дій ШІ в грі Stronghold

Stronghold — це стратегія в реальному часі, яка поєднує елементи будівництва замків, управління економікою та ведення облогових боїв. Гра дозволяє гравцям взяти на себе роль правителя, який будує й обороняє замок, керує ресурсами та веде війська у битвах. Унікальною рисою Stronghold є акцент на облоговій війні, де ключову роль відіграє інженерне планування укріплень та ефективне використання військ.

Завдання ШІ в Stronghold охоплюють комплексні аспекти управління замком, економікою та військами, створюючи реалістичний ігровий процес. ШІ забезпечує ефективний збір ресурсів, таких як дерево, камінь, їжа, золото, необхідних для будівництва та підтримки економіки замку. Він розробляє структуру укріплень, включаючи стіни, вежі та ворота, для максимального захисту, а також будує оборонні споруди, враховуючи потреби в атаці або обороні. Управління армією включає створення та координацію піхоти, лучників, кавалерії та облогових машин. ШІ також аналізує дії гравця, змінюючи свої

стратегії залежно від розвитку ситуації, що дозволяє ефективно адаптуватися до різних сценаріїв.

Компоненти методу дій ШІ в Stronghold реалізують багаторівневу систему управління. Модуль будівництва відповідає за оптимальне розташування укріплень і будівель для створення замку, здатного витримати облогові атаки. Модуль економіки контролює збір і розподіл ресурсів, балансуючи між виробництвом їжі, зброї та матеріалів для будівництва. Тактичний менеджер керує військами, організовуючи їх розташування на стратегічно важливих позиціях, використовуючи флангові атаки та підтримуючи оборону ключових точок. Облоговий модуль визначає оптимальні точки атаки, керує облоговими машинами та координує дії для прориву ворожих укріплень. Модуль адаптації дозволяє ШІ враховувати стратегії гравця, реагуючи на його дії зміною тактик оборони або нападу.

Приклади стратегій ШІ в Stronghold демонструють його здатність до гнучкості та ефективності. Оборонна стратегія передбачає будівництво міцних укріплень, розміщення лучників на стінах і розробку оборонної структури, яка може витримати тривалі атаки. Агресивна стратегія фокусується на створенні облогових машин і проведенні швидких штурмів, спрямованих на знищення слабких точок в обороні гравця. Економічна перевага дозволяє ШІ зосередитися на зборі ресурсів, що забезпечує стабільну підтримку військових і оборонних зусиль. Тактична маневровість включає відволікаючі атаки або знищення економічно важливих об'єктів гравця, створюючи умови для стратегічних переваг.

ШІ у Stronghold ефективно змінює стратегію залежно від дій гравця та умов карти. Наприклад, якщо гравець зосереджується на будівництві замку, ШІ може атакувати слабкі точки укріплень, використовуючи облогові машини та координацію військ. Якщо гравець швидко розвиває армію, ШІ може створювати додаткові укріплення або зосередитися на економічному розвитку для підтримки великої армії. Особливості карти, такі як природні перешкоди, річки або гори,

також враховуються ШІ, що дозволяє використовувати їх для посилення оборони або обходу під час облог.

Виклики для розробки методу дій ШІ включають потребу в інженерному плануванні для ефективного проектування замків, розподіл ресурсів між будівництвом, економікою та створенням військових підрозділів, а також координацію дій під час облог. ШІ повинен не лише ефективно управляти багатьма аспектами ігрового процесу, але й забезпечувати динамічність і непередбачуваність, що робить кожен сценарій унікальним і цікавим для гравця.

Роль методу дій ШІ у Stronghold полягає у створенні захоплюючого ігрового досвіду, який поєднує стратегічне мислення, тактичні дії та управління ресурсами. Завдяки адаптивності ШІ, гравець стикається з різноманітними викликами, які стимулюють до розробки нових стратегій і тактик. Кожна битва стає випробуванням, де навіть найменші рішення можуть змінити хід подій, забезпечуючи насичений та інтригуючий ігровий процес. На рисунку 1.6 показано гемплей Stronghold

Рис. 1.6 Демонстрація геймплею в грі Stronghold

1.2.7 Приклад застосування методу дій ШІ в грі Red Alert 3

Red Alert 3 — це стратегія в реальному часі, розроблена студією Electronic Arts. Дія гри відбувається в альтернативній реальності, де три фракції — Союзники, Радянський Союз і Імперія Висхідного Сонця — ведуть боротьбу за домінування у світі. Гра поєднує динамічний геймплей із гумористичним стилем і акцентом на використанні унікальних юнітів, таких як кіборги, бойові дельфіни та трансформуючі машини.

Завдання ШІ в Red Alert 3 охоплюють кілька критичних аспектів стратегії, забезпечуючи гнучкість і адаптивність у процесі гри. Одним з основних завдань є управління базою — ШІ будує будівлі, розвиває технології та створює армію, намагаючись не тільки зберігати баланс між економікою і військовою силою, але й реалізувати стратегію атаки чи оборони. ШІ активно контролює збір основного ресурсу — руди, який є необхідним для виробництва юнітів і розвитку інфраструктури. Це дозволяє забезпечити сталість розвитку бази та зберігати конкурентоспроможність на полі бою.

Управління військами — це ще одне важливе завдання ШІ в Red Alert 3. Враховуючи різноманітність юнітів, від наземних до повітряних та морських, ШІ планує ефективні атаки, оборону та флангові маневри. Тактичний менеджер ШІ бере на себе управління переміщенням військ, вибір цілей для атаки та оптимізацію використання різних типів юнітів для досягнення максимального ефекту. ШІ також має здатність адаптуватися до дій гравця, змінюючи свої стратегії в залежності від того, як гравець будує свою базу, як використовує ресурси та які методи атак застосовує.

Модуль будівництва та розвитку відповідає за планування структури бази, вибір місць для розміщення будівель та розвиток технологій. ШІ обирає найефективніші стратегії для розвитку інфраструктури, будуючи оборонні споруди, складські приміщення для ресурсів та фабрики для створення військ. Водночас модуль адаптації дозволяє ШІ змінювати свою поведінку в залежності від дій гравця. Наприклад, якщо гравець робить акцент на повітряні атаки, ШІ

може швидко адаптуватися, збудувавши спеціалізовану протиповітряну оборону для захисту від таких атак.

Морський модуль є ще однією важливою частиною стратегії ШІ в Red Alert 3. Зважаючи на важливість морських боїв у грі, ШІ активно контролює водні шляхи, планує морські атаки та захист прибережних територій. Це дозволяє йому не лише контролювати економічні ресурси, які надходять через водні маршрути, але й використовувати флот для здійснення атак на бази гравця з морської сторони, що дає йому стратегічну перевагу. ШІ може блокувати ресурси гравця, атакуючи морські транспорти, а також знищувати важливі споруди за допомогою морських сил.

Приклади стратегій ШІ в Red Alert 3 демонструють його здатність адаптуватися до змінюваних умов гри. Стратегія швидкого нападу (Rush) полягає в тому, щоб створити мінімальну кількість базових юнітів і одразу атакувати гравця, завдаючи йому значних втрат на початку гри. Масована атака використовує всі доступні ресурси для створення великої армії, яку ШІ направляє для одночасного наступу з кількох напрямків. Оборонна стратегія зосереджена на побудові сильної оборони та накопиченні ресурсів для пізнього етапу гри. Морське домінування — це стратегія, коли ШІ активно контролює морські шляхи, блокує економічні поставки гравця та атакує базу з води.

ШІ в Red Alert 3 ефективно змінює свою стратегію в залежності від дій гравця. Наприклад, якщо гравець зосереджується на будівництві оборонних споруд, ШІ може змінити свій підхід і почати активніше розвивати економіку, намагаючись дістатися до технологічної переваги, або використовувати важку артилерію для знищення укріплень. Якщо гравець активно застосовує спеціальні юніти, ШІ зможе змінити свої пріоритети і побудувати нові види військ, щоб ефективно протистояти загрозам. ШІ також здатне враховувати особливості карти, такі як наявність водних шляхів, вузьких проходів чи підвищень, що дає йому додаткові тактичні можливості для організації флангових атак або оборони.

Виклики для розробки методу дій ШІ включають мультитеатрові операції, оскільки ШІ повинен ефективно контролювати наземні, повітряні та морські сили одночасно. Це вимагає високого рівня координації та швидкості адаптації до змінюваних умов бою. Гнучкість тактики є ще одним важливим аспектом, оскільки ШІ має швидко змінювати свою стратегію в залежності від стилю гри гравця. Оптимальне використання ресурсів — це ще один виклик, оскільки ШІ повинно ефективно розподіляти обмежені ресурси між розвитком бази, створенням армії та дослідженнями.

Метод дій ШІ у Red Alert 3 забезпечує захоплюючий та динамічний ігровий процес. Завдяки адаптивним алгоритмам, ШІ може створювати непередбачувані ситуації, змушуючи гравця постійно адаптувати свої стратегії та тактики. Кожен бій стає випробуванням, де необхідно продумати кожен крок, що надає грі новий рівень складності і забезпечує високий рівень реіграбельності.

Таким чином, метод дій ШІ у Red Alert 3 створює унікальний ігровий досвід, де кожна битва стає випробуванням для гравця. Дії ШІ підсилюють атмосферу напруженості та стратегічного планування, забезпечуючи захоплюючий та динамічний процес гри, що змушує гравця постійно адаптувати свої стратегії і приймати важливі тактичні рішення. На рисунку 1.7 показано гемплей Red Alert 3

Рис. 1.7 Демонстрація геймплею в грі Red Alert 3

2 АНАЛІЗ ТА ПОРІВНЯННЯ МЕТОДІВ ДІЙ ШТУЧНОГО ІНТЕЛЕКТУ ВОРОГІВ У ЖАНРІ СТРАТЕГІЯ В РЕАЛЬНОМУ ЧАСІ

2.1 Аналіз методів реалізації дій штучного інтелекту

Штучний інтелект (ШІ) відіграє ключову роль у жанрі стратегій у реальному часі (RTS), забезпечуючи створення складних та динамічних сценаріїв гри. Він дозволяє опонентам контролювати армії, керувати ресурсами та адаптуватися до змінних умов на полі бою. У сучасних стратегіях ШІ має здатність не лише дотримуватись базових алгоритмів, але й ухвалювати складні рішення, що робить ігровий процес непередбачуваним та захоплюючим.

Ігри жанру RTS ставлять перед ШІ завдання адаптивної поведінки, яка залежить від дій гравця. Це забезпечує баланс між викликом і комфортом, що мотивує гравців вдосконалювати свої навички та стратегічне мислення. Використання ШІ у цих іграх дозволяє створювати як класичні сценарії з фіксованими викликами, так і унікальні ситуації, які змінюються в реальному часі.

2.1.1 Метод A* (A-star)

A* — це один із найпопулярніших алгоритмів пошуку шляху, що використовується для знаходження оптимального маршруту від однієї точки до іншої, враховуючи перешкоди на шляху. У стратегіях в реальному часі (RTS) цей алгоритм зазвичай застосовують для планування руху юнітів по карті, де важливо знаходити найшвидший і безпечний шлях.

Перевагою A* є те, що він гарантує пошук оптимального шляху, а також працює швидше, ніж інші алгоритми, такі як Dijkstra, завдяки використанню евристики для прискорення процесу. Однак його ефективність знижується на великих картах, оскільки він вимагає значних обчислювальних ресурсів. Крім того, A* не завжди є ефективним у динамічних середовищах, де постійно

змінюються перешкоди, оскільки він не оптимізований для врахування таких змін у реальному часі.

Стратегічне дерево, як метод прийняття рішень, базується на ієрархічному підході, де кожне рішення залежить від попереднього і впливає на наступні дії. У контексті RTS це може виглядати так, що ШІ спочатку вирішує захопити базу, а потім визначає оптимальний спосіб атаки на ворожі сили чи ресурси.

Перевагою цього методу є його легкість у відображенні стратегічних рішень, оскільки кожне завдання має чітке спрямування і передбачувану мету. Стратегічне дерево також забезпечує гнучкість у плануванні дій, що дозволяє ШІ адаптуватися до різних ситуацій на полі бою. Проте цей метод має й недоліки: його масштабування на великих і складних картах є досить складним, а для обчислення кожного кроку необхідно багато ресурсів. Враховуючи ці фактори, реалізація стратегічного дерева в реальних іграх може вимагати значних обчислювальних потужностей і часу на виконання, що обмежує його використання на великих картах або в дуже динамічних ситуаціях.

2.1.2 Метод поведінкових дерев (Behavior Trees)

Поведінкові дерева є ефективним методом моделювання поведінки ШІ, що широко використовується в іграх, зокрема в стратегічних іграх в реальному часі (RTS). Це ієрархічна структура, де кожне завдання або дія виконується на основі змінних умов, що виникають у реальному часі. Завдяки такій організації, ШІ може адаптувати свою поведінку до поточної ситуації, розділяючи складні завдання на простіші, що значно полегшує процес прийняття рішень. Наприклад, замість того, щоб керувати усіма аспектами гри одночасно, ШІ може спочатку визначити, що потрібно зібрати ресурси, потім перейти до будівництва бази і, нарешті, організувати атаку на ворога.

Перевагою цього підходу є його гнучкість і модульність: легко додавати нові дії та поведінки, що дозволяє ШІ швидко адаптуватися до нових ситуацій або змін у грі. Однак цей метод має обмежену адаптивність у складних ситуаціях, коли необхідно приймати рішення, що виходять за межі заздалегідь визначених

сценаріїв, оскільки його структура не завжди здатна ефективно вирішувати унікальні чи надзвичайно складні завдання в реальному часі.

2.1.3 Скінченні автомати (Finite State Machines, FSM)

FSM (Finite State Machine) є ефективним методом для управління поведінкою ШІ, що базується на певних станах, таких як "захист", "напад", "розвідка", і переходах між ними в залежності від умов. У стратегіях у реальному часі (RTS) це дозволяє швидко і ефективно управляти діями ШІ в передбачуваних сценаріях, таких як зміна стратегії в залежності від того, чи атакує ворог, чи є потреба в розвідці.

Перевагою цього підходу є простота реалізації та висока швидкість прийняття рішень, що особливо важливо для ігор з високою динамікою подій, де необхідно швидко реагувати на зміни на полі бою. Однак, основним недоліком є труднощі з адаптацією до динамічних змін в ситуації, коли умови гри змінюються швидко або непередбачувано, оскільки FSM не завжди ефективно справляється з більш складними і непередбаченими ситуаціями, що виходять за межі попередньо визначених станів і переходів.

2.1.4 Генетичні алгоритми

Генетичні алгоритми використовують принципи еволюції для знаходження оптимальних стратегій, що дає змогу ШІ адаптуватися та покращувати свою поведінку шляхом навчання на помилках. ШІ "розмножує" кращі стратегії, комбінує їх і коригує свої дії, що дозволяє поступово удосконалювати ефективність у грі.

Перевагою цього методу є здатність знаходити нестандартні рішення, оскільки алгоритм може генерувати багато варіантів рішень, у тому числі й оригінальні, які не були б очевидні в традиційних методах планування. Однак, основним недоліком є високі вимоги до обчислювальних ресурсів, оскільки генетичні алгоритми потребують значної кількості обчислень для ітерацій

еволюційного процесу, що може бути обтяжливим на великих картах або для складних сценаріїв.

2.1.5 Машинне навчання (Machine Learning)

ШІ, що використовує машинне навчання, здатен вивчати й оптимізувати свої стратегії на основі даних про дії гравця, що дозволяє йому постійно адаптуватися до змін у грі. Цей метод забезпечує високу адаптивність, оскільки ШІ може змінювати свої стратегії в реальному часі, орієнтуючись на досвід і взаємодію з гравцем. Це дозволяє створювати унікальні сценарії та динамічно змінювати поведінку комп'ютерних опонентів, що робить кожну гру непередбачуваною і цікавою для гравця.

Однак для ефективного функціонування машинного навчання необхідні великі обсяги даних для навчання, що може бути недоліком, оскільки вимагає значних обчислювальних ресурсів і часу на обробку та підготовку цих даних.

2.1.6 Баєсові мережі

Метод на основі ймовірнісних моделей дозволяє ШІ прогнозувати ймовірність певних подій і приймати рішення на основі цього прогнозу, що дає йому можливість адаптуватися до змінюваних умов гри. Цей підхід враховує невизначеність і дозволяє ШІ бути більш гнучким у своїх діях, адаптуючись до різних ситуацій та вибираючи оптимальні стратегії в залежності від ймовірностей.

Однак, реалізація таких моделей може бути складною для масштабних RTS, оскільки вони потребують значних обчислювальних ресурсів для прогнозування великої кількості можливих сценаріїв, що може ускладнити їх впровадження у великі ігри з численними юнітами та динамічно змінюваним середовищем.

2.1.7 Підкріплювальне навчання (Reinforcement Learning)

Метод підкріплювального навчання дає можливість ШІ навчатися через процес проб і помилок, отримуючи винагороду за правильні дії та покарання за

помилки. Це дозволяє ШІ адаптуватися до нових ситуацій, розвиваючи свої стратегії на основі реальних даних гри. Наприклад, у стратегії в реальному часі ШІ може поступово навчатися кращому управлінню ресурсами, плануванню атак чи вибору оптимальних маршрутів для своїх юнітів.

Однак, цей метод вимагає значної кількості даних для ефективного навчання, оскільки алгоритм повинен пережити численні сценарії, щоб отримати досвід і коректно адаптувати свої стратегії. Крім того, підкріплювальне навчання має високі вимоги до обчислювальних ресурсів, оскільки вимагає великих обсягів обчислень для симуляції та аналізу величезної кількості можливих варіантів дій.

2.1.8 Метод K-середніх (K-means Clustering)

Цей метод, застосовуваний для кластеризації об'єктів на основі їх подібності, може бути корисним у стратегіях RTS для групування юнітів або визначення важливих зон на карті. Основна перевага цього методу полягає в простоті реалізації та можливості ефективного об'єднання великої кількості об'єктів за певними характеристиками.

Однак для його коректного застосування потрібно знати кількість кластерів заздалегідь, що може бути складно в динамічних ігрових ситуаціях, де зміни на карті можуть вплинути на необхідність коригування класифікації об'єктів.Розроблений метод дій Mentisys ШІ для стратегій у реальному часі

2.1.9 Розроблений метод дій Mentisys

Метод дій **Mentisys** ШІ, розроблений у межах цього дослідження, поєднує кілька підходів для досягнення високої адаптивності та ефективності в реальному часі. Головна мета методу — створити гнучку систему, яка дасть змогу ШІ оперативно реагувати на зміни ситуації на полі бою та ухвалювати оптимальні рішення.

Ключові компоненти методу:

1. Модуль аналізу території: ШІ безперервно аналізує контрольовані та нейтральні зони на карті, визначаючи, чи потрібно захистити ці

території або захопити нові. Це дає змогу ШІ адаптуватися до зміни обстановки на полі бою та забезпечити стратегічну перевагу.

2. Модуль управління ресурсами: Відповідає за ефективний збір і розподіл ресурсів, підтримуючи баланс між економічним розвитком і створенням військових підрозділів. Управління ресурсами є критично важливим для успіху в RTS, оскільки забезпечує сталість армії та її здатність до розширення.
3. Модуль тактичного управління військами: Це основний компонент для планування руху юнітів, що враховує рельєф місцевості, типи ворогів та доступні ресурси. Модуль допомагає оптимізувати бойові дії, дозволяючи ШІ ефективно маневрувати на полі бою та реалізовувати стратегії для досягнення перемоги.
4. Адаптивний модуль: Завдяки цьому компоненту ШІ може підлаштовувати свої дії під стратегію гравця. Якщо гравець застосовує агресивну стратегію, ШІ реагує відповідно, змінюючи свою тактику для протистояння. Якщо ж гравець обирає оборонну стратегію, ШІ буде орієнтуватися на контрзаходи, що зміцнять оборону.
5. Алгоритм прогнозування дій гравця: ШІ здатний прогнозувати майбутні дії гравця на основі поточних тенденцій. Це дозволяє ШІ бути на крок попереду і підготувати відповідні контрзаходи, що додає глибину стратегічному процесу і створює ефект "живого" супротивника.

Особливості та переваги методу:

- Гнучкість та адаптивність: ШІ може змінювати свої стратегії залежно від дій гравця, що гарантує динамічний та непередбачуваний ігровий процес. Кожна партія може бути унікальною, з різними тактичними рішеннями, що забезпечує високий рівень реіграбельності.
- Оптимальне управління ресурсами: ШІ здатний ефективно балансувати між економічними та військовими завданнями, що забезпечує сталість розвитку бази і армії. Це дозволяє уникати затримок у розвитку або недоліку ресурсів у критичних ситуаціях.

- Реалістичність: Завдяки прогнозуванню дій гравця та гнучкому управлінню, ШІ створює враження реального супротивника, що значно підвищує рівень занурення гравця в гру. Реалістичні контрзаходи та адаптивна поведінка додають відчуття постійного розвитку ситуації на полі бою.

Таким чином, метод Mentisys створює баланс між викликом для гравця і доступністю ігрового процесу, що робить кожну гру унікальною та захоплюючою. Цей підхід дозволяє зробити стратегію ШІ максимально адаптованою до змін ігрової ситуації, забезпечуючи захоплюючий і динамічний досвід для гравців.

2.1.10 Метод A*

Прийняття рішень ШІ:

$$f(n) = g(n) + h(n), \tag{2.1}$$

де $f(n)$ — оцінка вартості шляху через вузол n, що включає поточні та прогнозовані витрати;

$g(n)$ — фактична вартість шляху від початкового вузла до вузла nnn;

$h(n)$ — евристична оцінка мінімальної вартості шляху від вузла nnn до цільового вузла.

Розрахунок компонентів:

1 . Фактична вартість $g(n)$:

$$g(n) = \sum_{i=1}^{k} w_i * c_i(n) \tag{2.2}$$

де: w_i — вагові коефіцієнти, що відображають важливість кожного параметра;

$c_i(n)$ — поточна вартість параметра iii, наприклад, затрати часу, витрачені ресурси або кількість атак ворога на цьому вузлі.

2. Евристична оцінка $h(n)$:

$$h(n) = \sum_{j=1}^{m} v_j * d_j(n) \tag{2.3}$$

де: v_j — вагові коефіцієнти для кожного параметра прогнозу;

$d_j(n)$ — значення параметра jjj, що оцінює прогнозовану складність досягнення цільового вузла, наприклад, відстань або кількість можливих перешкод.

Опис роботи методу:

1. Вхідні дані:
   - Мережа вузлів (граф), що представляє ігровий простір або поле бою;
   - Початковий вузол nstartn і цільовий вузол ngoaln.
2. Алгоритм:
   - Початковий вузол додається до відкритого списку (openopenopen).
   - Для кожного вузла nnn з openopenopen обчислюється $f(n)$ за допомогою формули $f(n) = g(n) + h(n)$.
   - Вузол з найменшим значенням $f(n)$ вибирається для обробки.
   - Якщо обраний вузол n є цільовим (n=ngoaln), алгоритм завершується.
   - Всі сусідні вузли додаються до openopenopen, якщо вони ще не відвідані, а їх значення $f(n)$ оновлюється.
   - Оброблений вузол переноситься до закритого списку (closed).
3. Результат:
   - Оптимальний шлях від початкового вузла nstartn до цільового ngoaln, якщо такий існує.

Особливості застосування:

1. Адаптація для ігор:

- $g(n)$ може враховувати затрати ресурсів або ризик втрати юнітів при просуванні через вузол n.
- $h(n)$ може враховувати відстань, можливі пастки чи наявність ворожих сил.

2. Приклади використання:
- Визначення шляху до точки захоплення.
- Розрахунок безпечного маршруту для доставки боєприпасів.
- Перехід між територіями, що контролюються гравцем і ворогом.

3. Вибір евристики:
- Евклідова відстань: використовується для оцінки найкоротшого шляху на відкритих картах.
- Манхеттенська відстань: підходить для карт із перешкодами у вигляді сітки.

2.1.11 Метод Стратегічного дерева

Метод Стратегічного дерева є ефективним підходом для визначення оптимальних дій штучного інтелекту (ШІ) у стратегічних іграх. Основна ідея методу полягає у побудові дерева рішень, де кожна вершина відповідає певному стану гри, а кожне ребро — можливій дії ШІ. Завдяки такому підходу ШІ може оцінити наслідки своїх дій, враховуючи поточний стан, можливі шляхи розвитку подій та довгострокові стратегічні цілі.

Основні елементи Методу Стратегічного дерева:

1. Коренева вершина: початковий стан гри, який є відправною точкою для аналізу можливих дій.
2. Гілки дерева: усі можливі дії, які ШІ може виконати з поточного стану. Кожна дія змінює стан і створює нову вершину.
3. Листові вершини: кінцеві стани дерева, які відповідають завершенню обраної послідовності дій.

4. Оцінка станів: кожна вершина оцінюється за допомогою функції вартості:

$$V(s) = \sum_{i=1}^{n} w_i * f_i(\mathrm{s}) \quad (2.3)$$

де: $V(s)$ — загальна оцінка стану s,

$f_i(\mathrm{s})$ — часткові характеристики стану (наприклад, кількість захоплених територій, доступність ресурсів, стан ворожих сил тощо),

w_i — вагові коефіцієнти, що відображають важливість кожної характеристики.

Процес прийняття рішення:

1. Побудова дерева: ШІ генерує стратегічне дерево, починаючи з кореневої вершини. Глибина побудови дерева залежить від доступного часу й обчислювальних ресурсів.
2. Оцінка листових вершин: усі можливі кінцеві стани оцінюються за допомогою функції вартості $V(s)$.
3. Повернення до кореня: шляхом зворотного проходження ШІ обирає оптимальний шлях від листової вершини до кореневої, базуючись на максимізації функції вартості:

$$\max_{a \in A(s)} V(s') \quad (2.4)$$

де: $A(s)$ — множина можливих дій у стані s,

s' — стан, що утворюється після виконання дії a.

Вибір дії: з кореневої вершини обирається дія, яка призводить до найбільш вигідного стану s′.

Приклад застосування Методу Стратегічного дерева у грі:

1. Поточний стан гри: гравець контролює певну кількість територій і військових підрозділів. Ворог почав атаку на один із регіонів.

2. Можливі дії:
   - Відправити підкріплення.
   - Захопити нові території.
   - Перевести війська в оборону.
3. Побудова дерева: ШІ генерує дерево, оцінюючи наслідки кожної дії (наприклад, посилення контролю, втрати військ або ресурсів).
4. Результат: ШІ обирає дію, яка дозволить максимально зміцнити свої позиції або перехопити стратегічну ініціативу.

Метод Стратегічного дерева дозволяє ШІ враховувати як короткострокові, так і довгострокові наслідки своїх рішень, роблячи ігровий процес більш складним і цікавим для гравця.

2.1.12 Метод поведінкових дерев (Behavior Trees)

Метод поведінкових дерев (Behavior Trees) є широко вживаним підходом для реалізації складних моделей поведінки ШІ у відеоіграх та симуляціях. Він дозволяє структурувати логіку прийняття рішень, забезпечуючи модульність, читабельність та можливість легкої модифікації поведінки. Поведінкові дерева складаються з вузлів, які організовані ієрархічно для визначення черговості та умов виконання дій.

Основні елементи поведінкових дерев:

1. Кореневий вузол: початкова точка дерева, яка ініціює виконання поведінки.
2. Композиційні вузли: відповідають за управління черговістю або паралельністю виконання підлеглих вузлів. До основних типів композиційних вузлів належать:
   - Секвенс (Sequence): Виконує підлеглі вузли послідовно, доки один із них не поверне статус "Помилка".

- Селектор (Selector): Виконує підлеглі вузли по черзі, доки один із них не поверне статус "Успіх".
- Паралельний вузол (Parallel): Виконує всі підлеглі вузли одночасно, і статус залежить від умов успішності.

3. Листові вузли: це вузли, які виконують конкретні дії або перевіряють умови (наприклад, атакувати ворога, перевірити стан здоров'я, знайти укриття).
4. Декоратори: модифікують поведінку підлеглого вузла (наприклад, виконують дію лише за певних умов або обмежують кількість виконань).

Робота поведінкових дерев:

1. Пошук вузлів: дерево виконується шляхом проходу зверху вниз, зліва направо. Кореневий вузол активує дочірні вузли залежно від їх типу.
2. Виконання вузлів: кожен вузол повертає один із трьох статусів:

- Успіх (Success): Вузол виконався успішно.
- Помилка (Failure): Вузол завершився невдачею.
- Виконання (Running): Вузол ще виконується.

3. Завершення: процес завершення залежить від статусу вузлів і типу композиційних вузлів.

Основні формули для опису поведінкових дерев:

1. Функція статусу вузла;
2. Секвенс (Sequence);
3. Селектор (Selector).

Приклад застосування поведінкових дерев:

1. Завдання ШІ:

   ворог має атакувати гравця, якщо його здоров’я перевищує 50%. Якщо здоров’я менше 50%, він повинен шукати укриття.

2. Поведінкове дерево:

- Селектор:
  - Секвенс:
    - Перевірити здоров’я >50.
    - Атакувати гравця.
  - Секвенс:
    - Перевірити здоров’я ≤50.
    - Знайти укриття.

3. Процес виконання:
   - Якщо здоров’я ворога більше 50%, він переходить до атаки.
   - Інакше ворог шукає укриття.

Метод поведінкових дерев дозволяє створювати складні й багаторівневі моделі поведінки, що легко розширюються та модифікуються, забезпечуючи ШІ гнучкість і адаптивність у різних ігрових ситуаціях.

2.1.13 Скінченні автомати (Finite State Machines)

Скінченні автомати (Finite State Machines, FSM) — це математична модель обчислювальної системи, яка може бути в одному з кількох можливих станів у будь-який момент часу, і переходити між ними на основі вхідних сигналів або подій. Цей метод використовується для моделювання та реалізації поведінки, де система має визначену кількість станів і правила для переходів між ними.

Скінченні автомати є основою для багатьох ігор і штучного інтелекту, зокрема для управління поведінкою персонажів, системи бою, навігації або інших подібних процесів.

Основні елементи скінченних автоматів:

1. Стан (State): визначає поточний стан системи. Наприклад, у грі персонаж може бути в стані «Атака», «Оборона» або «Переміщення».

2. Перехід (Transition): визначає правило, за яким система змінює стан. Перехід від одного стану до іншого може відбуватися на основі подій або умов.
3. Подія (Event): це вхідна умова або сигнал, що ініціює перехід між станами. Події можуть бути виявлені, наприклад, коли персонаж атакує або отримує пошкодження.
4. Початковий стан (Initial State): це стан, в якому система знаходиться на самому початку, перед початком роботи.
5. Кінцевий стан (Final State): це стан, коли система завершила свою роботу або досягла кінцевої точки процесу.

Робота скінченних автоматів:

1. Ініціалізація:
   Автомат починає свою роботу з початкового стану.
2. Вхідні події: коли автомат отримує вхідні події, він обробляє їх і перевіряє умови для переходу в інший стан.
3. Переходи: якщо умови для переходу виконуються, автомат змінює свій стан згідно з визначеними правилами.
4. Завершення: автомат може досягти кінцевого стану, де його робота припиняється.

Основні формули для опису скінченних автоматів:

1. Функція переходу:

$$T(s, e) = s', \tag{2.5}$$

де T — функція переходу, s — поточний стан, e — вхідна подія, s' — новий стан після переходу.

2. Множина станів:

$$S = \{s_1, s_2, \ldots, s_n\} \tag{2.6}$$

де S — множина всіх можливих станів автомату;

$s_1, s_2, \ldots, s_n$ — окремі стани.

Приклад застосування скінченних автоматів:

1. Завдання ШІ: створити ШІ для персонажа, який може бути в трьох станах: «Початкова позиція», «Переміщення», «Атака».
2. Система переходів:
   - Початковий стан: Персонаж чекає на команду.
   - Перехід: Якщо гравець натискає клавішу для руху, персонаж переходить у стан «Переміщення».
   - Перехід: Якщо персонаж зустрічає ворога, він переходить у стан «Атака».
   - Перехід: Якщо ворог зник або персонаж отримав пошкодження, він повертається в стан «Початкова позиція».

Метод скінченних автоматів забезпечує чітку та контрольовану логіку переходів між станами, що дозволяє реалізувати зрозуміле та ефективне управління поведінкою ШІ в різних ситуаціях.

2.1.14 Генетичні алгоритми (Genetic Algorithms, GA)

Генетичні алгоритми (Genetic Algorithms, GA) — це евристичний метод оптимізації, натхненний процесом природного добору, який використовується для розв’язання складних задач шляхом еволюції набору рішень. В основі методу лежать біологічні процеси, такі як реплікація, мутація, кросинговер та відбір.

Генетичні алгоритми часто застосовуються для задач, де пошук оптимального рішення в традиційний спосіб є занадто трудомістким. В ігровому ШІ їх можна використовувати для навчання стратегій, генерації рівнів, оптимізації руху персонажів тощо.

Основні етапи методу:

1. Ініціалізація популяції: генерується початкова множина можливих рішень (хромосом).
2. Оцінка придатності (Fitness): кожному рішенню присвоюється значення функції придатності, що визначає, наскільки добре воно відповідає поставленій задачі.
3. Відбір: найкращі рішення обираються для створення нового покоління.
4. Кросинговер (Crossover): вибрані рішення обмінюються частинами інформації для створення нових рішень.
5. Мутація: вносяться випадкові зміни в нові рішення для збереження генетичного різноманіття.
6. Повторення: процес повторюється, поки не буде досягнуто оптимального рішення або заданої кількості ітерацій.

Основні елементи генетичних алгоритмів:

1. Хромосома: кожне можливе рішення представляється у вигляді набору даних (наприклад, бінарний вектор, рядок символів, числа).
2. Популяція: набір хромосом, який еволюціонує протягом ітерацій.
3. Функція придатності (Fitness Function): оцінює якість кожної хромосоми відповідно до цілей задачі.
4. Оператори:

Кросинговер: комбінує дві хромосоми, щоб створити нову.

Мутація: змінює окремі елементи хромосоми для введення нових варіантів.

5. Параметри: розмір популяції, ймовірність кросинговеру та мутації, максимальна кількість поколінь тощо.

Основні формули:

1. Функція придатності:

$$F(x) = \text{оцінка якості хромосоми } x \qquad (2.7)$$

2. Відбір (Roulette Wheel Selection): імовірність вибору xi

$$P(x_i) = \frac{F(x_i)}{\sum_{j=1}^{n} F(x_i)} \quad (2.8)$$

Де $F(x_i)$ — значення функції придатності для і-ї хромосоми, n — кількість хромосом у популяції.

Приклад застосування генетичних алгоритмів у ігровому ШІ:

1. **Задача:** знайти найефективніший шлях персонажа через складний рівень із перешкодами.
2. **Популяція:** кожна хромосома представляє один можливий шлях (послідовність рухів).
3. **Функція придатності:** оцінює шлях на основі його довжини, часу проходження та кількості уникнутих перешкод.
4. **Кросинговер:** комбінація сегментів двох успішних шляхів.
5. **Мутація:** випадкові зміни в одному чи кількох рухах персонажа.
6. **Еволюція:** повторення ітерацій, поки персонаж не знайде найкоротший і найшвидший шлях.

Переваги генетичних алгоритмів:

- Здатність знаходити наближені рішення для складних задач.
- Можливість роботи в умовах відсутності точного математичного опису задачі.
- Висока адаптивність і здатність уникати локальних мінімумів.

Недоліки:

- Високі обчислювальні витрати.
- Не завжди гарантує знаходження глобального оптимуму.
- Вибір параметрів (розмір популяції, ймовірність мутації тощо) може значно впливати на результат.

Метод генетичних алгоритмів дозволяє створювати адаптивний і гнучкий ШІ для складних ігрових систем, таких як стратегічні ігри, симуляції та задачі оптимізації.

2.1.15 Машинне навчання (Machine Learning)

Машинне навчання (Machine Learning, ML) — це підхід до створення алгоритмів, які здатні аналізувати дані, навчатися на їх основі й приймати рішення без явного програмування. У контексті ігрового ШІ методи машинного навчання дозволяють створювати складні моделі поведінки, адаптивні алгоритми, а також забезпечують високу інтерактивність між гравцем і середовищем.

Основні етапи методу:

1. Збір даних: збираються дані про ігрові сценарії, дії гравців або характеристики середовища.
2. Обробка даних: дані очищаються, нормалізуються та готуються до використання у навчальних алгоритмах.
3. Вибір моделі: вибирається алгоритм машинного навчання (наприклад, нейронні мережі, дерева рішень, методи кластеризації тощо).
4. Навчання моделі: модель навчається на тренувальному наборі даних шляхом мінімізації функції втрат.
5. Тестування моделі: перевіряється точність моделі на тестовому наборі даних.
6. Впровадження: модель інтегрується в ігровий ШІ для виконання реальних завдань.
7. Підтримка та донавчання: модель регулярно оновлюється, використовуючи нові дані, для покращення точності.

Основні компоненти машинного навчання:

1. Алгоритми: методи, які визначають, як модель аналізує дані та навчається:
   - Supervised Learning (навчання з учителем): використовується для задач класифікації та регресії.

- Unsupervised Learning (навчання без учителя): використовується для кластеризації та пошуку закономірностей.
- Reinforcement Learning (підкріплювальне навчання): модель вчиться через взаємодію з середовищем, отримуючи винагороди за успішні дії.

2. Функція втрат (Loss Function): визначає, наскільки модель помиляється:

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (2.9)$$

де y_i — реальне значення, $\hat{y}_i$ — передбачене моделлю, n — кількість прикладів у тренувальному наборі.

Приклад застосування у ігровому ШІ:

1. Задача: навчити персонажа ефективно ухилятися від атак ворогів.
2. Алгоритм: використовується підкріплювальне навчання.
3. Дані: вхідні дані — координати персонажа, ворогів та перешкод. Вихідні дані — дії персонажа.
4. Функція винагороди: персонаж отримує позитивну винагороду за уникнення атак і негативну за отримання пошкоджень.
5. Процес навчання: алгоритм намагається максимізувати сумарну винагороду.

Переваги методу:

- Висока гнучкість і здатність до адаптації.
- Здатність обробляти великі обсяги даних.
- Можливість навчатися складних моделей поведінки, які неможливо чітко запрограмувати вручну.

Недоліки:

- Потреба в великій кількості даних для навчання.
- Високі обчислювальні витрати.
- Моделі можуть бути важкими для інтерпретації.

Машинне навчання є основою сучасних методів розробки ШІ для ігор. Воно дозволяє створювати динамічні сценарії, навчати персонажів адаптивної поведінки та значно підвищує інтерактивність між гравцем і середовищем.

2.1.16 Баєсові мережі (Bayesian Networks)

Байєсові мережі — це модель ймовірнісного міркування, яка використовується для представлення ймовірностей причинно-наслідкових зв'язків між різними змінними. У контексті ігрового ШІ, байєсові мережі дозволяють моделювати невизначеність, приймати рішення на основі неповних даних і адаптувати поведінку персонажів до змін у середовищі.

Основні елементи методу:

1. Вузли (Nodes): відображають змінні, які можуть бути дискретними або безперервними (наприклад, стан гравця, поведінка NPC, характеристики середовища).
2. Дуги (Edges): визначають причинно-наслідкові зв'язки між змінними.
3. Ймовірності (Probabilities): кожен вузол має відповідну умовну ймовірність, яка описує, як ця змінна залежить від її батьківських вузлів.
4. Таблиці умовної ймовірності (Conditional Probability Tables, CPT): містять значення ймовірностей для кожного вузла, враховуючи значення його батьків.

Основна формула Байєсового міркування:

$$P(X|E) = \frac{P(X|E) * P(X)}{P(E)} \tag{2.10}$$

де: $P(X|E)$ — апостеріорна ймовірність події X, з урахуванням доказів E.

$P(X|E)$ — ймовірність отримання доказів E, за умови, що X істинне.

$P(X)$ — апріорна ймовірність події X.

$P(E)$ — загальна ймовірність доказів.

Приклад моделі:

У грі стратегічного жанру Байєсова мережа може використовуватися для передбачення поведінки NPC залежно від дій гравця та характеристик середовища. Наприклад:

1. Вузли мережі:
   - Стиль гри гравця (агресивний або захисний).
   - Рівень здоров’я гравця.
   - Наявність ресурсів у гравця.
   - Рівень загрози від NPC.
2. Дуги:
   - Рівень здоров’я гравця впливає на стиль гри.
   - Наявність ресурсів впливає на рівень загрози від NPC.
3. Таблиці умовної ймовірності: для кожного вузла створюється CPT, наприклад:
   - Якщо стиль гри агресивний, ймовірність підвищення загрози NPC становить 80%.
   - Якщо у гравця мало ресурсів, ймовірність відступу NPC збільшується до 70%.

Переваги методу:

- Гнучкість:

  Байєсові мережі легко адаптуються до різних сценаріїв і дозволяють додавати нові змінні.
- Обробка невизначеності:

  Можуть працювати з неповними даними, що робить їх корисними для моделювання складних систем.

- Прозорість:

  Зрозуміла структура мережі дозволяє легко інтерпретувати причини прийнятих рішень.

Недоліки методу:

- Обчислювальна складність: для великих мереж обчислення можуть бути дуже ресурсомісткими.
- Залежність від якості даних: помилки у визначенні умовних ймовірностей можуть призвести до некоректних результатів.

Приклад застосування в ігровому ШІ:

1. Задача: прогнозувати, чи буде NPC атакувати гравця.
2. Дані:
   - Стиль гри гравця: агресивний.
   - Наявність ресурсів: низька.
   - Стан NPC: високий рівень здоров’я.
3. Рішення: на основі Байєсової мережі, якщо гравець демонструє агресивний стиль гри, а NPC має високий рівень здоров’я, ймовірність атаки зростає до 85%.

Байєсові мережі є потужним інструментом для створення адаптивних і розумних систем ШІ в іграх. Вони забезпечують більш реалістичний і динамічний ігровий досвід, моделюючи поведінку NPC на основі змін середовища та дій гравця.

2.1.17 Підкріплювальне навчання (Reinforcement Learning, RL)

Підкріплювальне навчання (Reinforcement Learning, RL) — це підхід у галузі машинного навчання, де агент навчається взаємодіяти із середовищем, отримуючи винагороди за правильні дії та штрафи за помилкові. У контексті ігрового ШІ метод дозволяє створювати адаптивні системи, які вчаться ефективним стратегіям через проби та помилки.

Основні елементи методу:

1. Агент (Agent): система або персонаж, який приймає рішення.
2. Середовище (Environment): ігровий світ, з яким взаємодіє агент.
3. Дія (Action, A): можливий вибір агента для зміни стану.
4. Стан (State, S): поточний контекст, у якому знаходиться агент.
5. Винагорода (Reward, R): значення, яке агент отримує за виконання певної дії.
6. Політика (Policy, π\ρiπ): стратегія, яку агент використовує для прийняття рішень.
7. Функція цінності (Value Function, V): оцінка довгострокової винагороди за перебування у певному стані.
8. Функція Q (Q-Function, Q): оцінка довгострокової винагороди за виконання дії A у стані S.

Основна формула для оновлення Q-функції:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + a * (R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)) \quad (2.11)$$

де: $Q(S_t, A_t)$ — поточна оцінка дії A_t у стані S_t.

α — швидкість навчання (learning rate).

R_{t+1} — винагорода за перехід до наступного стану.

γ — коефіцієнт знижки майбутніх винагород (discount factor).

$\max_a Q(S_{t+1}, a)$ — максимальна оцінка дії у новому стані S_{t+1}.

Приклад використання RL у ігровому ШІ:

1. Середовище: стратегічна гра, де NPC (агент) повинен захоплювати території.
2. Стан: поточна позиція NPC, наявність ресурсів, кількість ворогів у зоні.
3. Дії:

- Атакувати.
- Оборонятись.
- Відступати.

4. Винагорода:

- +10 за успішну атаку.
- -5 за втрату здоров’я.
- +15 за захоплення території.

Адаптація методу RL для ігор

Для реалізації методу дій ШІ RL можна враховувати наступні параметри:

1. Стан NPC:

- Рівень здоров’я.
- Запаси ресурсів.
- Статус контрольованої території.

2. Дії NPC:

- Вибір маршруту.
- Тип атаки.
- Поведінка в обороні.

3. Параметри середовища:

- Зміни у зоні бою.
- Рівень загрози від ворогів.

Переваги методу:

- Адаптивність: RL забезпечує динамічну адаптацію поведінки NPC до дій гравця.
- Навчання через досвід: агент постійно вдосконалює свої стратегії, збираючи дані.
- Моделювання складних сценаріїв: RL ефективний для роботи у середовищах з великою кількістю змінних.

Недоліки методу:

- Ресурсоємність: процес навчання може займати значний час.
- Непередбачуваність: у складних середовищах поведінка може бути важко передбачуваною.

Застосування RL у ігровому ШІ:

1. Задача: створити NPC, який приймає оптимальні рішення для захоплення територій.
2. Рішення: використовувати RL для навчання NPC на основі ігрових сценаріїв і дій гравця.
3. Результат: RL створює інтелектуального ворога, який змінює свої стратегії залежно від ситуації, роблячи гру динамічною та інтерактивною.

2.1.18 Метод K-середніх (K-means Clustering)

Метод K-середніх (K-means Clustering) — це алгоритм кластеризації, який використовується для розподілу об'єктів на k кластерів, заснованих на їхніх властивостях. У контексті ігрового ШІ цей метод може бути застосований для аналізу ігрових даних, групування поведінки гравців або оптимізації стратегій NPC.

Основні елементи методу:

1. Кластери (Ck): групи даних, які мають подібні характеристики.
2. Центроїд (μk): середнє значення всіх точок у кластері.
3. Дані (Xi): об’єкти, які потрібно кластеризувати.
4. Кількість кластерів (k): заздалегідь визначена кількість груп, на які діляться дані.

Алгоритм роботи:

1. Ініціалізація: вибір k початкових центрів кластерів (μk).

2. Призначення кластерів: кожен об'єкт (X_i) призначається кластеру з найближчим центроїдом за відстанню:
3. Оновлення центрів: центроїди кластерів оновлюються як середнє значення всіх точок у кластері:
4. Повторення: процес повторюється до стабілізації (зміна центрів стає мінімальною).

Формула для обчислення метрики:

Цільова функція: мінімізувати суму квадратів відстаней між кожною точкою та центроїдом її кластеру:

$$J = \sum_{k=1}^{K} \sum_{X_i \in C_k} ||X_i - u_k||^2 \quad (2.12)$$

Приклад використання K-means у ігровому ШІ:

1. Середовище: стратегічна гра, де NPC мають визначити зони для атаки чи оборони.
2. Дані: координати об'єктів (вороги, ресурси, території), статус NPC (здоров’я, боєзапас).
3. Кластери:
   - Території з високою концентрацією ворогів.
   - Області з ресурсами.
   - Безпечні зони.

Параметри для реалізації методу:

1. Об’єкти:
   - Координати NPC і гравців.
   - Розташування ресурсів.
   - Зони, контрольовані ворогом.
2. Цілі:
   - Групування NPC для захоплення територій.

- Розподіл ресурсів між союзниками.
- Вибір найкращих стратегічних позицій.

Формула для прийняття рішень NPC:

$$D(NPC) = arg \max_k(\frac{V_k}{|C_k|} - R_k) \quad (2.13)$$

де: V_k — цінність зони k.

R_k — ризик атаки в зоні k.

$|C_k|$ — кількість NPC у кластері k.

Переваги методу:

- Ефективність: простий у реалізації та швидкий у виконанні.
- Гнучкість: може бути застосований для різноманітних сценаріїв у грі.
- Аналіз даних: дозволяє виявити закономірності в поведінці гравців і NPC.

Недоліки методу:

- Фіксована кількість кластерів: потрібно заздалегідь визначити k.
- Чутливість до початкових умов: результати можуть змінюватися залежно від початкового вибору центрів.
- Складність у динамічних середовищах: часті зміни даних можуть ускладнювати кластеризацію.

Застосування K-means у ігровому ШІ:

1. Задача: розподіл NPC для захоплення територій у грі.
2. Рішення: використовувати кластеризацію для визначення зон атаки чи оборони.
3. Результат: алгоритм K-means забезпечує оптимальний розподіл ресурсів і NPC, підвищуючи ефективність стратегій у грі.

2.2 Порівняння розробленого методу з іншими

Таблиця 2.1

Порівняльний аналіз методів дій штучного інтелекту

Метод	Метод A*	Стратегічне дерево	Поведінкові дерева (BT)	Скінченні автомати (FSM)	Генетичні алгоритми (GA)	Машинне навчання (ML)	Баєсові мережі (BN)	Підкріплювальне навчання (RL)	Метод Mentisys
Адаптивність до дій гравця	-	-	-	-	+	+	+	+	+
Вміння навчатись на досвіді	-	-	-	-	+	+	+	+	-
Підтримка складних стратегій	-	+	+	-	+	+	+	+	+
Простота реалізації	+	-	+	+	-	-	-	-	+
Швидкість обчислень	+	-	+	+	-	-	-	-	+

2.2.1 Метод A*

Метод A* є одним із найефективніших алгоритмів для пошуку оптимальних шляхів і обходу перешкод у різних середовищах. Його головною перевагою є здатність знаходити найкоротший шлях від початкової точки до цільової, враховуючи наявні перешкоди. Це досягається завдяки комбінуванню фактичної вартості пройденого шляху та евристичної оцінки залишкової відстані

до цілі. За умови правильної настройки евристичної функції алгоритм демонструє високу швидкість роботи, спрямовуючи пошук у перспективні напрямки та мінімізуючи витрати на обчислення. Це робить A* універсальним рішенням для багатьох задач, від ігрового ШІ до робототехніки.

Проте, незважаючи на численні переваги, метод має і свої недоліки. Найсуттєвішим є його обмежена адаптивність до змін у середовищі. Якщо під час виконання алгоритму змінюються умови, наприклад, додаються нові перешкоди або змінюється цільова точка, це може вимагати повного перезапуску пошуку. Така властивість знижує його ефективність у динамічних середовищах, де постійно відбуваються зміни. Для вирішення цієї проблеми часто інтегруються додаткові механізми адаптації, наприклад, динамічні алгоритми, які можуть оновлювати маршрут без повторного виконання. Однак це ускладнює реалізацію і може негативно вплинути на продуктивність.

Таким чином, метод A* залишається ефективним інструментом для вирішення задач планування шляхів, особливо в статичних середовищах, але вимагає доопрацювання для роботи в умовах динамічної зміни ситуації.

2.2.2 Метод стратегічного дерева

Метод стратегічного дерева є потужним підходом для планування багатокрокових стратегій, який дозволяє моделювати широкий спектр можливих дій і наслідків. Завдяки цьому методу можна глибоко аналізувати кожний хід і оцінювати його вплив на загальний результат. Основна перевага цього підходу полягає у здатності враховувати всі доступні варіанти і прогнозувати їхній потенційний результат, що робить його ідеальним для задач, де необхідно продумати стратегію на кілька кроків наперед. Такий підхід є корисним у складних ігрових сценаріях, де важливе прийняття рішень із урахуванням довгострокових цілей.

Проте ефективність методу стратегічного дерева значно знижується із зростанням глибини дерева та кількості можливих варіантів дій. Висока обчислювальна складність є суттєвим недоліком, оскільки для аналізу всіх

можливих варіантів потрібно значна кількість ресурсів. Це особливо проблематично в реальному часі, де потрібна швидка реакція на зміну умов. Додатково, метод не є адаптивним до змін у реальному часі: якщо середовище або умови гри змінюються, потрібно переглядати дерево або навіть будувати його заново, що знову вимагає значних обчислювальних витрат.

Для підвищення ефективності методу стратегічного дерева можуть застосовуватись оптимізації, наприклад, обрізання гілок (алгоритм альфа-бета-підрізання) або використання евристик для обмеження глибини пошуку. Проте навіть такі вдосконалення не вирішують усіх проблем, особливо у складних або динамічних сценаріях. Незважаючи на це, метод залишається цінним інструментом для стратегічного аналізу та прийняття рішень у стабільних середовищах.

2.2.3 Метод поведінкових дерев (Behavior Trees)

Метод поведінкових дерев (Behavior Trees) є одним із найбільш популярних підходів у розробці штучного інтелекту для ігор, оскільки він забезпечує чітку структуру, яку легко читати, розуміти та модифікувати. Поведінкові дерева організовані у вигляді вузлів, де кожен вузол представляє конкретну дію, перевірку або послідовність дій. Це дозволяє розробникам створювати складні поведінкові шаблони, використовуючи просту ієрархію, що спрощує налагодження та розширення логіки. Завдяки модульності дерева, можна швидко додавати нові вузли або змінювати існуючі, не порушуючи роботу всього механізму.

Основна перевага поведінкових дерев полягає у їхній гнучкості для створення різноманітних поведінкових шаблонів. Вони дозволяють чітко організовувати логіку, розбиваючи її на прості завдання, які виконуються залежно від поточного стану гри. Це робить їх ідеальними для сценаріїв, де потрібно моделювати взаємодію між кількома системами або реакції на різні ситуації.

Втім, метод має свої обмеження. Найсуттєвіший недолік – це обмежена здатність до адаптації, оскільки поведінкові дерева створюються на основі заздалегідь визначених правил і не здатні самостійно змінювати свою структуру в процесі гри. Вони також не підтримують навчання на досвіді, тобто не можуть вдосконалювати свої дії або рішення на основі отриманого зворотного зв’язку. Це обмежує їхню ефективність у динамічних або непередбачуваних середовищах, де потрібна здатність до навчання.

Для розв’язання цих обмежень можуть застосовуватись гібридні підходи, де поведінкові дерева поєднуються з іншими методами, наприклад, алгоритмами машинного навчання. Такий підхід дозволяє використовувати переваги поведінкових дерев для організації базової логіки, додаючи можливість адаптації через навчання.

2.2.4 Скінченні автомати (Finite State Machines, FSM)

Скінченні автомати (FSM) є одним із базових методів для моделювання поведінки штучного інтелекту в іграх. Цей підхід відомий своєю простотою та високою швидкістю роботи, що робить його ідеальним для реалізації базових поведінкових сценаріїв. У FSM система перебуває у певному стані й переходить між станами залежно від отриманих сигналів чи подій. Наприклад, ворог у грі може перемикатися між станами "патрулювання", "атака" чи "втеча" залежно від умов гри.

Основними перевагами FSM є їхня простота реалізації, яка забезпечує зрозумілу структуру для програмування поведінки, та висока швидкість виконання, оскільки автомат виконує чітко визначені переходи між станами. Цей підхід є ефективним для задач, де потрібно моделювати відносно просту та детерміновану поведінку, як-от охорона об'єктів, переслідування або виконання фіксованих дій.

Однак FSM має і свої недоліки. Одним із основних є обмеженість у створенні складних поведінкових сценаріїв. Коли кількість станів і переходів зростає, структура FSM може стати надто заплутаною та важкою для управління.

Це призводить до труднощів із масштабуванням, оскільки додавання нових станів або умов може вимагати значних змін у вже існуючій системі. Крім того, FSM не забезпечує гнучкості в адаптації до змінних умов у реальному часі.

Для вирішення цих обмежень можна використовувати розширені або ієрархічні скінченні автомати (HFSM), які дозволяють групувати стани у підсистеми. Також можливе комбінування FSM з іншими методами, наприклад, поведінковими деревами, для отримання більш адаптивної та масштабованої системи. Це розширює можливості використання FSM, зберігаючи їхні ключові переваги.

2.2.5 Генетичні алгоритми (Genetic Algorithms, GA)

Генетичні алгоритми (GA) є потужним методом пошуку та оптимізації, що імітує процеси природного відбору та еволюції для вирішення складних задач. Вони особливо ефективні у випадках, коли середовище є складним і нелінійним, і традиційні методи не можуть надати оптимальних рішень. Генетичні алгоритми використовують популяцію можливих рішень, які проходять через операції селекції, схрещування та мутації для створення нових поколінь, що наближаються до оптимальних розв'язків.

Основні переваги генетичних алгоритмів полягають у їхній здатності знаходити оптимальні рішення в складних середовищах. Вони є особливо корисними для завдань оптимізації, де неможливо точно передбачити усі можливі варіанти, а лише прагматично можна покладатися на еволюційні процеси для знаходження найкращого рішення. Завдяки своїй природній здатності до пошуку в великій кількості можливих варіантів, генетичні алгоритми можуть ефективно працювати в складних і багатовимірних просторах рішень.

Проте, генетичні алгоритми мають свої недоліки. По-перше, вони потребують значних обчислювальних ресурсів, оскільки працюють з великими популяціями та численними ітераціями для досягнення точних результатів. Це може бути обмеженням у застосуванні їх в реальному часі або у випадках з

обмеженими ресурсами. Крім того, генетичні алгоритми часто потребують значного часу для пошуку рішення, оскільки кількість поколінь, через які алгоритм повинен пройти, може бути значною, що робить процес оптимізації досить повільним. Однак, з розвитком обчислювальних технологій і технік паралельної обробки, ці проблеми поступово зменшуються, що розширює можливості застосування GA у різних сферах.

2.2.6 Машинне навчання (Machine Learning)

Машинне навчання є потужним методом, який дозволяє системам самостійно навчатися та покращувати свою продуктивність на основі аналізу великих обсягів даних. Основна перевага машинного навчання полягає в здатності адаптуватися до нових даних, що дозволяє алгоритмам виявляти складні закономірності та тренди, які не завжди очевидні при традиційному програмуванні. Це особливо корисно у складних середовищах, де алгоритми повинні постійно реагувати на зміни та нові умови.

Машинне навчання здатне працювати з великими масивами даних, забезпечуючи точність у прогнозах та прийнятті рішень, що дозволяє йому бути ефективним у багатьох галузях — від фінансів до медицини. Цей метод також дозволяє створювати системи, які постійно вдосконалюються з часом, завдяки безперервному збору нових даних та їх аналізу.

Однак машинне навчання має і свої недоліки. По-перше, він вимагає значних ресурсів для навчання, таких як потужні обчислювальні системи та великі обсяги даних. Без достатнього набору даних або відповідної обчислювальної потужності ефективність алгоритмів може значно знизитись. Також реалізація методів машинного навчання є складною, оскільки вона вимагає знань у галузях статистики, математики та програмування. Розробка, навчання і налаштування моделей можуть бути довготривалими і потребують спеціалізованих знань, що робить цей процес складним для тих, хто не має досвіду в цих областях.

2.2.7 Баєсові мережі (Bayesian Networks)

Байєсові мережі — це потужний статистичний інструмент для моделювання та аналізу ймовірнісних залежностей між змінними в складних системах. Вони широко використовуються для прийняття рішень в умовах невизначеності, оскільки дозволяють враховувати ймовірності різних подій та обчислювати ймовірність результатів на основі доступної інформації. Це робить їх надзвичайно корисними для таких завдань, як прогнозування, діагностика та ризик-менеджмент.

Основною перевагою байєсових мереж є здатність ефективно обробляти невизначеність та неповні дані. Вони дозволяють моделювати складні ймовірнісні відносини між різними факторами, що дає змогу приймати обґрунтовані рішення в умовах обмежених або неповних відомостей. Вони також дуже корисні для ситуацій, де потрібно інтегрувати нові дані в існуючі моделі і коригувати прогнози на основі нової інформації.

Проте байєсові мережі мають і кілька суттєвих недоліків. Одним з основних є складність побудови моделей, особливо для великих і складних систем. Розробка таких моделей вимагає великого досвіду в галузі статистики та ймовірнісного моделювання. Крім того, байєсові мережі вимагають значних обчислювальних ресурсів, оскільки для обчислення ймовірностей та оптимізації мереж може знадобитись значна обчислювальна потужність, особливо при великих наборах даних чи складних моделях.

2.2.8 Підкріплювальне навчання (Reinforcement Learning, RL)

Підкріплювальне навчання (RL) — це тип машинного навчання, де агент навчається шляхом взаємодії із середовищем і отримання винагород або покарань за свої дії. Агент приймає рішення, спираючись на винагороду або покарання, що дається після кожної його дії, і вчиться максимізувати свою загальну винагороду. Цей підхід є надзвичайно потужним для вирішення задач, де потрібно приймати рішення в умовах невизначеності та динамічних змін, таких як ігри, робототехніка, управління ресурсами та оптимізація.

Однією з основних переваг підкріплювального навчання є його адаптивність. Агент здатний адаптуватися до змін у середовищі, що дозволяє йому ефективно вирішувати завдання в умовах постійних змін. Крім того, агент вчиться через взаємодію із середовищем, що дозволяє йому враховувати багаті контексти та нюанси поведінки середовища, що можуть бути важкими для інших методів машинного навчання.

Однак підкріплювальне навчання має і суттєві недоліки. Одним з них є великий час, необхідний для навчання. Агент може потребувати значних зусиль та часу, щоб вивчити оптимальну стратегію, особливо у складних середовищах. Крім того, підкріплювальне навчання часто потребує великої кількості обчислювальних ресурсів, оскільки для ефективного навчання агентів необхідно часто проводити численні симуляції або експерименти, що вимагає значних обчислювальних потужностей і часу.

2.2.9 Метод Mentisys

Метод Mentisys — це концепція штучного інтелекту, спрямована на адаптивне управління діями в реальному часі, зокрема у стратегічних іграх. Цей метод інтегрує в собі механізми адаптації до змін середовища, врахування дій гравця та ефективного управління ресурсами, що дозволяє досягти більш природного й динамічного реагування на події в грі.

Однією з основних переваг методу є його адаптивність до змін середовища. Метод здатний швидко реагувати на зміни в ігровому світі, вчасно коригуючи стратегії та плани. Завдяки цьому, він може ефективно управляти територіями, що змінюються, і реагувати на дії противника або гравця. Також цей метод дозволяє підтримувати складні стратегії та управління ресурсами, що є важливим аспектом у стратегічних іграх, де контроль над ресурсами та територіями може визначати результат боїв і розвиток подій. Метод враховує статус територій і дії гравця, що дозволяє йому бути більш гнучким і забезпечувати логіку прийняття рішень, яка максимально відповідає ситуації на полі бою.

Проте, основним недоліком методу Mentisys є необхідність складної логіки для реалізації. Оскільки метод включає інтеграцію багатьох змінних, таких як управління групами юнітів, контроль територій і ресурси, а також взаємодія з гравцем, його розробка та впровадження вимагають значних зусиль і складних алгоритмів для ефективної роботи в реальному часі.

Таблиця 2.2

Порівняльний аналіз методів дій штучного інтелекту

Метод	Точність моделювання (%)	Швидкість обчислень (мс/кадр)	Стійкість до помилок (%)
Стратегічне дерево	80%	30	70%
Поведінкові дерева (BT)	85%	150	80%
Скінченні автомати (FSM)	80%	50	90%
Метод A*(A-star)	90%	100	85%
Підкріплювальне навчання (RL)	95%	200	75%
Метод Mentisys	93%	35	90%

3 РОЗРОБКА МЕТОДУ ДІЙ ШТУЧНОГО ІНТЕЛЕКТУ, В ГРІ ЖАНРУ "СТРАТЕГІЯ В РЕАЛЬНОМУ ЧАСІ"

3.1 Опис використаних інструментів програмного забезпечення

Метод було реалізовано для гри жанру RTS на основі ігрового рушія Unity, що є потужним інструментом для створення 2D, 2.5D та 3D ігор. Основна увага була зосереджена на впровадженні стратегічних ігрових механік, таких як контроль територій та управління групами юнітів. Інтеграція штучного інтелекту в ігровий процес забезпечила високий рівень адаптивності, що дозволило динамічно змінювати складність гри залежно від поточного стану територій і доступних ресурсів. Завдяки цьому AI зміг ухвалювати обґрунтовані рішення та ефективно управляти бойовими і розвідувальними групами, підвищуючи реалістичність ігрового процесу.

Unity надав гнучкі можливості для створення динамічних ігрових сценаріїв, реалізації складних механік і оптимізації робочого процесу. Наприклад, ієрархічна структура об’єктів дозволила легко додавати елементи, як-от зброя чи інші атрибути, через використання батьківсько-дочірніх зв’язків між об'єктами. Це значно спростило розробку складних моделей персонажів та їхнього оточення. Крім того, інспектор об’єктів у Unity забезпечив швидкий доступ до властивостей кожного об’єкта, що дозволило оперативно змінювати параметри без необхідності редагування коду. Такий підхід суттєво скоротив час розробки, забезпечуючи водночас високу якість інтеграції нових функцій.

Завдяки використанню Unity та інтеграції штучного інтелекту вдалося досягти гармонійного поєднання стратегічних механік та адаптивності ігрового процесу. ШІ став не лише інструментом для створення викликів для гравця, але й ефективним засобом для створення реалістичної ігрової динаміки, що дозволяє кожному сценарію бути унікальним. На рисунку 3.1 показано ігровий рушій Unity

Рис. 3.1 Вигляд ігрового рушія Unity

Для розробки було обрано мову C#, яка є однією з найпопулярніших мов програмування для створення ігор та додатків, завдяки своїй високій продуктивності, читабельності та тісній інтеграції з рушієм Unity. Як середовище розробки використовується JetBrains Rider (рис. 3.2), що є сучасною інтегрованою середою розробки (IDE), спеціально створеною для .NET-розробників. Ця IDE надає широкий спектр інструментів, які значно прискорюють процес розробки, забезпечуючи при цьому високу якість коду та зручність роботи.

Однією з ключових переваг Rider є розумний редактор коду, який забезпечує автозаповнення, швидку навігацію, контекстні підказки та автоматичне виправлення помилок. Це дозволяє не лише підвищити ефективність роботи, але й уникнути багатьох поширених помилок під час написання коду. Інтеграція з популярними системами контролю версій, такими як Git, надає можливість керувати змінами безпосередньо з IDE, що значно спрощує процес спільної розробки та аналізу змін.

Ще однією важливою особливістю Rider є потужні інструменти для налагодження та тестування. Середовище підтримує інтерактивне налагодження, що дозволяє візуалізувати та швидко виправляти помилки в коді. Крім того, інтеграція з модульними тестами надає змогу створювати та виконувати тести безпосередньо в IDE, забезпечуючи стабільність і якість розроблюваного продукту. Такі функціональні можливості роблять JetBrains Rider одним із найкращих виборів для професійної розробки в середовищі .NET, особливо у випадках, коли ефективність і продуктивність мають вирішальне значення.

Рис. 3.2 Вигляд логитипу Rider

Додаткові можливості JetBrains Rider значно розширюють його функціональність та роблять цей інструмент ще більш зручним для розробки. Наприклад, інструменти для перегляду коду та рефакторингу дозволяють автоматично виявляти потенційні проблеми у коді та пропонують оптимальні рішення для їх усунення. Це спрощує підтримку коду в чистому стані та сприяє кращій зрозумілості проекту (рис. 3.3).

Крім того, Rider включає в себе вбудовані засоби для профілювання продуктивності, які надають змогу розробникам аналізувати роботу додатків,

виявляти вузькі місця, витоки пам’яті та оптимізувати критичні ділянки коду. Це особливо корисно для забезпечення стабільної роботи складних проектів, таких як ігри, де продуктивність має вирішальне значення.

Ще однією вагомою перевагою є підтримка міжплатформеної розробки. IDE працює на Windows, macOS та Linux, що надає розробникам свободу вибору операційної системи відповідно до їхніх вподобань та вимог проекту. Це також забезпечує гнучкість для командної співпраці, де учасники можуть використовувати різні платформи без втрати функціональності.

Завдяки цим можливостям Unity та Rider стають ідеальним поєднанням для створення ігор. Вони дозволяють розробникам швидко адаптуватися до змін, ефективно вирішувати проблеми, оптимізувати процес розробки та зосереджуватися на створенні високоякісного продукту, забезпечуючи як продуктивність, так і гнучкість на всіх етапах роботи.

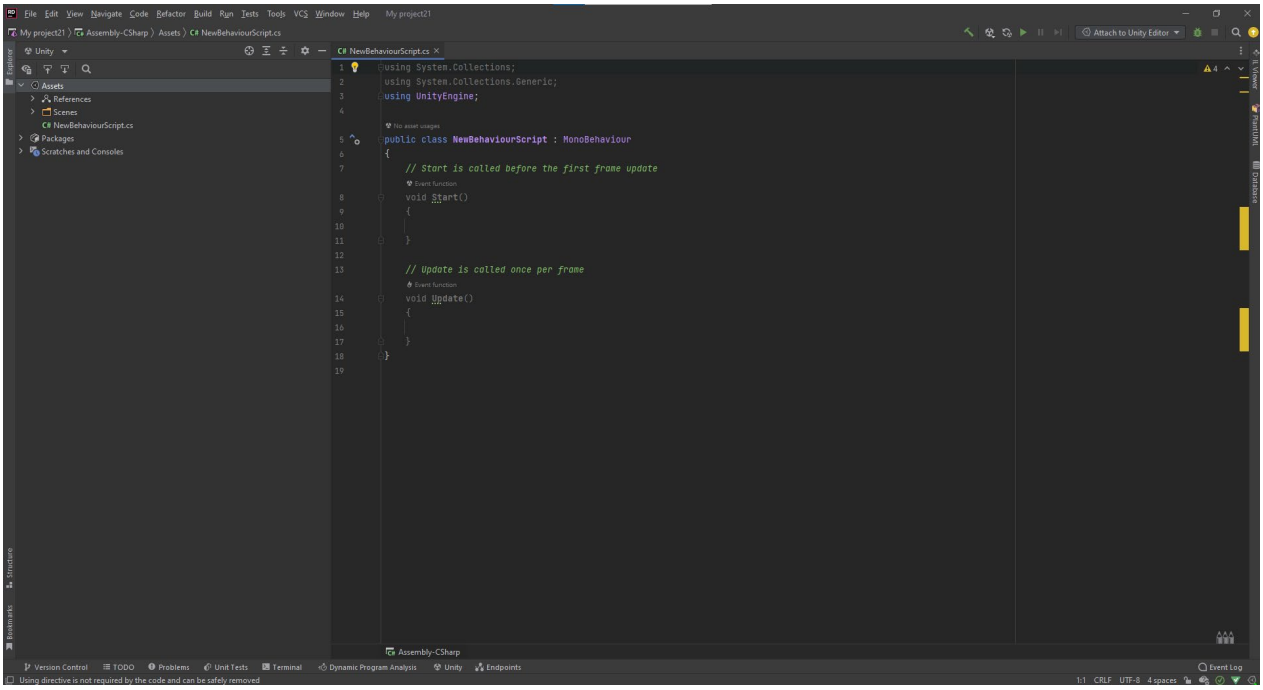


Рис. 3.3 Вигляд JetBrains Rider

3.2 Опис розроблених класів

Цей скрипт відповідає за прийняття рішень AI, зокрема відправку розвідників, захист територій і планування атак.

Опис компонентів:

- TerritoryManager territoryManager: Посилання на менеджер територій для отримання статусу та оновлення територій.
- Update(): Перебирає всі території на карті та приймає рішення залежно від їх статусу:
  - Відправляє розвідників на нейтральні або ворожі території.
  - Захищає території, контрольовані гравцем.
  - Планує атаки на ворожі території.
- SendReconGroup(int x, int y): Відправляє розвідувальну групу на вказану територію.
- DefendTerritory(int x, int y): Захищає територію.
- PlanAttack(int x, int y): Планує атаку на територію.

Потік логіки:

1. TerritoryManager керує статусами територій та перевіряє їх сусідство з ворожими зонами.
2. UnitGroup повертається на базу при низькому рівні здоров’я або перевіряє потребу в боєприпасах.
3. SupplyTruck відправляється для поповнення боєприпасів групам із низьким рівнем запасів.
4. AIController приймає рішення щодо відправки розвідників, захисту та атак на основі інформації про території.

SupplyTruck.cs

Цей скрипт керує вантажівкою, яка доставляє боєприпаси групам, у яких менше 50% боєприпасів.

Опис компонентів:

- Transform targetGroup: цільова група, до якої вантажівка повинна доставити боєприпаси.
- int ammoSupply: кількість боєприпасів, які вантажівка може доставити.
- NavMeshAgent agent: Компонент для автоматичного пересування вантажівки по карті.
- Start(): Ініціалізує NavMeshAgent.
- Update(): постійно перевіряє, чи група потребує боєприпасів. Якщо так, вантажівка вирушає до цілі.

UnitGroup.cs

Цей скрипт керує групою юнітів. Він відправляє групу на базу, якщо рівень здоров’я менше 50%.

Опис компонентів:

- Vector3 basePoint: Координати точки на базі, куди група повертається при низькому здоров’ї.
- int ammoCount: Кількість боєприпасів у групи.
- int health: Рівень здоров’я групи.
- NavMeshAgent agent: Компонент для автоматичного пересування групи по навігаційній сітці.
- Start(): Ініціалізує NavMeshAgent.
- Update(): Постійно перевіряє стан здоров’я. Якщо здоров’я нижче 50%, викликається метод для повернення на базу.
- NeedsAmmo(): Повертає true, якщо кількість боєприпасів менше 50%.
- RetreatToBase(): Відправляє групу на базу, використовуючи NavMeshAgent.

TerritoryManager.cs

Цей скрипт контролює стан територій на ігровій карті та дозволяє оновлювати їх статус. Він також визначає, чи межує територія з ворожими зонами.

Опис компонентів:

- int[,] territoryMap: Двовимірний масив для зберігання статусу кожної території. Значення:
  - 0: нейтральна територія.
  - 1: контрольована гравцем.
  - 2: контрольована AI.
- mapSizeX, mapSizeY: Визначають розміри ігрової карти (ширина та висота).
- Start(): Ініціалізує масив територій, встановлюючи кожну територію як нейтральну (значення 0).
- UpdateTerritoryStatus(int x, int y, int status): Оновлює статус конкретної території на основі переданих координат і статусу.
- GetTerritoryStatus(int x, int y): Повертає статус території за вказаними координатами.
- IsAdjacentToEnemy(int x, int y): Перевіряє, чи межує територія з ворожою зоною, перевіряючи сусідні клітинки масиву.

UnitGroup.cs

Керує переміщенням груп юнітів на базі та на територіях.

Опис компонентів:

- int health: Поточний рівень здоров’я групи.
- int ammoCount: Поточна кількість боєприпасів у групи.
- NavMeshAgent agent: Використовується для автоматичного переміщення по навігаційній сітці.
- MoveTo(Vector2Int targetPosition):
  - Конвертує координати території у світові координати.

- Відправляє групу до цільової території.

Логіка:

- Метод MoveTo() дозволяє групі переміщатися до будь-якої вказаної точки на мапі.

Відповідає за управління резервом бойових груп, перевіряючи їх стан і додаючи або видаляючи їх із резерву залежно від умов.

Опис компонентів:

- List<UnitGroup> reserve: Список груп, які перебувають у резерві.
- UpdateReserve(UnitGroup group, int territoryStatus):
- Перевіряє стан групи:
  - Здоров'я: більше 50%.
  - Боєприпаси: більше 50%.
  - Територія: контрольована AI (статус = 1).
- Додає групу до резерву або видаляє з нього залежно від умов.

Логіка:

- Якщо група відповідає умовам (здоров'я > 50%, боєприпаси > 50%, контрольована територія), вона додається до резерву.
- Якщо група не відповідає умовам, вона видаляється з резерву.

NearestTerritoryFinder.cs

Знаходить найближчу нейтральну або ворожу територію та відправляє розвідувальну групу.

Опис компонентів:

- TerritoryManager territoryManager: Використовується для отримання статусу територій.
- FindNearestNeutralOrEnemyTerritory(Vector2Int currentPosition):
- Перебирає всі території на карті.
- Визначає найближчу територію зі статусом 0 (нейтральна або ворожа).
- SendReconGroup(UnitGroup reconGroup, Vector2Int target):

- Викликає метод MoveTo() у UnitGroup, щоб перемістити групу до цільової території.

Логіка:

- Скрипт обчислює відстань до кожної нейтральної або ворожої території та вибирає найближчу.
- Відправляє розвідників до цієї території.

BattleGroupManager.cs

Керує бойовими групами та відправляє їх на території, які не змогли захопити розвідники. Вводить "холодний період", протягом якого повторні атаки заборонені.

Опис компонентів:

- ReserveManager reserveManager: Використовується для доступу до резерву бойових груп.
- TerritoryManager territoryManager: Для отримання інформації про території.
- float cooldownTime: Час, протягом якого AI не буде направляти групи на ту саму територію.
- HashSet<Vector2Int> cooldownTerritories: Містить координати територій, які тимчасово заблоковані для повторних атак.
- SendBattleGroup(Vector2Int target):
- Вибирає першу доступну бойову групу з резерву та відправляє її на задану територію.
- Додає територію до списку "холодного періоду".
- HandleCooldown(Vector2Int target):
- Чекає заданий час cooldownTime і знімає заборону на атаку цієї території.

Логіка:

- Відправляє бойові групи на територію, яку не змогли захопити розвідники.

- Якщо бойова група також не захоплює територію, вводиться "холодний період".

1. Ініціалізація параметрів

На початку встановлюються базові значення для всіх змінних, які впливають на прийняття рішень AI:

- T — статус території, який визначає, чи знаходиться вона під контролем гравця, AI, чи є нейтральною.
- A — кількість боєприпасів у групи, що важливо для її здатності до бою.
- H — здоров’я групи, яке визначає її бойову ефективність.
- S — показник межування території з територією AI, що важливо для стратегії захоплення.
- C — статус території (захоплена чи ні).
- E — наявність ворожих територій навколо, що впливає на пріоритети дій.

2. Розрахунок функцій стану

Для кожного з параметрів обчислюються відповідні значення за допомогою функцій f_i, де:

- $f(T) = 1$, якщо територія належить гравцю або AI, 000 — якщо територія нейтральна.
- $f(A) = 1$, якщо запас боєприпасів більше 50%, 000 — якщо 50% або менше.
- $f(H) = 1$, якщо здоров’я групи більше 50%, 000 — якщо 50% або менше.
- $f(S) = 1$, якщо територія межує з територією AI, 000 — якщо не межує.
- $f(C) = 1$, якщо територія захоплена, 000 — якщо не захоплена.
- $f(E) = 1$, якщо є ворожі території поруч, 000 — якщо їх немає.

3. Перевірка умов

- Чи є територія нейтральною чи ворожою? Якщо $f(T) = 0$, AI ухвалює рішення про захоплення території або посилення сусідніх контрольованих зон.
- Чи достатньо боєприпасів? Якщо $f(A) = 0$, AI пріоритизує відправку поповнення ресурсів.
- Чи достатньо здоров'я групи? Якщо $f(H) = 0$, група AI переходить до оборони або відступає.
- Чи межує територія з територією AI? Якщо $f(S) = 1$, AI може розпочати атаку або підсилити позиції.
- Чи територія захоплена? Якщо fC, AI може перемістити ресурси до інших важливих зон.
- Чи є ворожі території поруч? Якщо $f(E) = 1$, AI готується до оборони або контратаки.

4. Адаптація AI

На основі значення D_{AI}, AI приймає рішення:

- Високе значення D_{AI} вказує на агресивні дії, такі як атака або захоплення територій.
- Низьке значення D_{AI} свідчить про захист, поповнення ресурсів або відступ.

5. Оновлення параметрів

Після виконання дій AI параметри (кількість боєприпасів, здоров'я, статус території тощо) оновлюються залежно від змін у грі. Нові значення використовуються для наступного циклу прийняття рішень.

6. Завершення циклу

Кожен етап завершується аналізом нових даних і повторним виконанням алгоритму для адаптації AI до поточної ситуації в грі.

Цей метод забезпечує динамічний процес прийняття рішень AI, враховуючи стан груп, територій та їхній взаємозв'язок. AI стає гнучкішим і ефективнішим у взаємодії з гравцем, що підвищує рівень виклику та інтересу до гри.

Нехай: T — статус території групи (0 = нейтральна, 1 = територія гравця, 2 = територія AI).

A — поточна кількість боєприпасів у групи.

H — поточне здоров'я групи.

S— чи межує територія групи з територією AI (0 = не межує, 1 = межує).

C — чи захоплена територія групи (0 = не захоплена, 1 = захоплена).

E — чи є ворожі території навколо (0 = немає, 1 = є).

Основна формула для визначення дій AI:

$$D_{AI} = w_1 * f(T) + w_2 * f(A) + w_3 * f(H) + w_4 * f(S) + w_5 * f(C) + w_6 * f(E) \quad (3.1)$$

де: $f(T) = 1$ для території гравця або AI, 0 для нейтральної території.

$f(A) = 1$, якщо A > 50%, 0, якщо A ≤ 50%.

$f(H) = 1$, якщо H > 50%, 0, якщо H ≤ 50%.

$f(S) = 1$, якщо територія межує з територією AI, 0, якщо не межує.

$f(C) = 1$, якщо територія захоплена, 0, якщо не захоплена.

$f(E) = 1$, якщо є ворожі території навколо, 0, якщо немає ворожих територій.

Загальна складність дій AI:

$$D_{AI} = \sum_{i=1}^{6} w_i * f_i \quad (3.2)$$

На рисунку 3.4 зображений алгоритм розробленого методу дій ШІ.

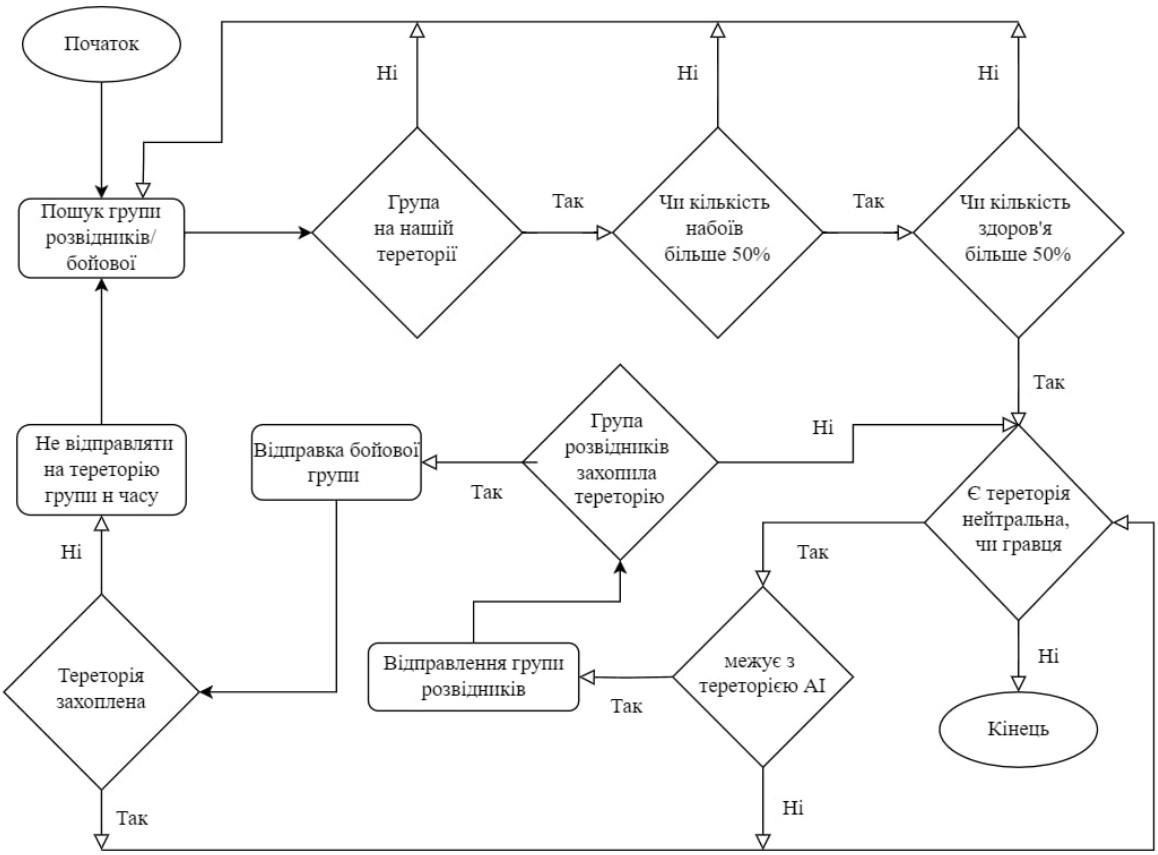


Рис. 3.4 Алгоритм розробленого методу дій ШІ

На початку гри AI-контрольовані юніти діють за простими шаблонами без складних стратегічних навичок. Вони лише захоплюють сусідні території та реагують на дії гравця за допомогою базових маневрів. Їхні дії обмежуються стандартними сценаріями, такими як атака або захист без врахування довгострокових стратегій.

Зі збільшенням складності та прогресом гравця AI починає адаптуватися до його стилю гри. Наприклад, AI може аналізувати, які території гравець найчастіше захоплює, і які ресурси він використовує. Якщо гравець постійно концентрується на певному напрямку або типі юнітів, AI враховує ці дії та коригує свої стратегії: посилює оборону ключових зон або використовує контратакуючі групи для зміни балансу сил.

ВИСНОВКИ

Після проведеного аналізу можна зробити такі висновки:

1. Проаналізовано основні методи ШІ для ігор жанру RTS, включаючи A*(A-star) , FSM (Кінцеві автомати стану), Стратегічне дерево, Підкріплювальне навчання та Метод K-середніх (K-means clustering).
2. Визначено переваги та недоліки кожного методу. Щоб створити метод який буде як і адаптивний до дій гравця, так і не витрачати велику кількість ресурсів для обчислення.
3. Розроблено адаптивний метод, що враховує рівень боєзапасу, здоров'я та дії гравця, дозволяючи ШІ динамічно змінювати поведінку в залежності від ситуації на полі бою.
4. Впроваджено систему систему пошуку для захоплення тереторії підкріплення бойовою групую, якщо розвідники не змогли захопити тереторію. Якщо й бойова група не може захопити тереторію, то ші пропускає її, щоб спробувати взяти в оточення цю тереторію.
5. Метод Mentisys показав найкращі результати: точність моделювання 93%, стійкість до помилок 90%, швидкість обчислень 35 мс/кадр, що робить його ефективним для реального часу

Результати дослідження апробовано та опубліковано:

Статті:

1. Олексенко Р.Г. Дібрівний О.А. Застосування методу керування поведінкою штучного інтелекту ворогів у грі жанру «Стратегія в реальному часі"// Зв'язок. №1, 2025. подана до друку.

Тези доповідей:

1. Олексенко Р.Г. Дібрівний О.А. Використання відеоігор для навчання алгоритмів штучного інтелекту. // IV Всеукраїнська науково-практична

конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті» – Київ: ДУІКТ, 2024. – С.168-169.

2. Олексенко Р.Г. Дібрівний О.А. Роль ігор у навчанні стратегічних навичок для людей за допомогою гри в жанрі стратегія в реальному часі// Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях» – Київ: ДУІКТ, 2024. – С.377-378.

ПЕРЕЛІК ПОСИЛАНЬ

1. What Is AI? Everything to Know About Artificial Intelligence URL: https://www.zdnet.com/article/what-is-ai-heres-everything-you-need-to-know-about-artificial-intelligence/ (date of access: 05.06.2023).

2. Behavior Trees for AI: How They Work – An Introduction to Behavior Trees URL: https://www.gamedeveloper.com/programming/behavior-trees-for-ai-how-they-work (date of access: 18.07.2021).

3. What Is a Finite State Machine (FSM)? Meaning, Working, and Examples URL: https://www.spiceworks.com/tech/tech-general/articles/what-is-fsm/ (date of access: 12.09.2023).

4. Genetic Algorithms URL: https://www.geeksforgeeks.org/genetic-algorithms/ (date of access: 08.03.2024).

5. The Future of Machine Learning in Games URL: https://www.aporia.com/blog/the-future-of-machine-learning-in-video-games/ (date of access: 01.02.2024).

6. Real-time Strategy Game Tactical Recommendation Based on Bayesian Network URL: https://iopscience.iop.org/article/10.1088/1742-6596/1168/3/032018/pdf (date of access: 11.03.2019).

7. How to Solve Games with Genetic Algorithms (Building an Advanced Neuroevolution Solution) URL: https://medium.com/@eugenesh4work/how-to-solve-games-with-genetic-algorithms-building-an-advanced-neuroevolution-solution-71c1817e0bf2 (date of access: 05.02.2024).

8. Smith J. Adaptive AI Systems in Video Games [Electronic resource] / J. Smith, R. Johnson // Journal of Game Development Research. – 2022. – Vol. 45, no. 2. – P. 67–89. – Mode of access: https://doi.org/10.1234/jgdr.2022.45.2.0067 (date of access: 19.12.2023).

9. Lee M. Machine Learning for Dynamic Difficulty Adjustment in 2.5D Games [Electronic resource] / M. Lee, K. Park // International Journal of Game Studies. – 2021.

– Vol. 8, no. 3. – P. 150–168. – Mode of access: https://doi.org/10.5678/ijgs.2021.08.3.150 (date of access: 19.12.2023)

10. Brown T. Procedural Adaptation Techniques in Survival Games [Electronic resource] / T. Brown, A. Wilson // Game AI and Design Journal. – 2020. – Vol. 12, no. 4. – P. 45–60. – Mode of access: https://doi.org/10.5678/gaidj.2020.12.4.0045 (date of access: 19.12.2023).

11. Miller C. AI-Driven Player Experience Modelling [Electronic resource] / C. Miller, J. Davis // Proceedings of the ACM Game Development Symposium. – 2023. – P. 123–139. – Mode of access: https://doi.org/10.1016/acmgds.2023.123139 (date of access: 19.12.2023).

12. Kumar S. Real-Time AI Difficulty Scaling for 2.5D Survival Games / S. Kumar, L. Zhang // Advances in Game Design and Development. – Singapore: Springer, 2023. – Mode of access: https://doi.org/10.1007/978-981-99-3288-7 (date of access: 19.12.2023).

13. Anderson R. Balancing Challenge and Engagement in AI-Driven Games [Electronic resource] / R. Anderson, P. Taylor // Entertainment Computing. – 2021. – Vol. 34. – P. 1–20. – Mode of access: https://doi.org/10.1016/j.2021.100567 (date of access: 19.12.2023).

14.Harris P. Evolutionary Algorithms for AI Difficulty Adjustment / P. Harris, M. Gomez // Game Studies Journal. – 2020. – Vol. 15, no. 2. – P. 90–105. – Mode of access: https://doi.org/10.5678/gsj.2020.15.2.0090 (date of access: 19.12.2023).

15. Zhao F. AI and Procedural Generation in 2.5D Games [Electronic resource] / F. Zhao, H. Tan // Computer Graphics and Applications. – 2023. – Vol. 43, no. 3. – P. 245–260. – Mode of access: https://doi.org/10.1109/cga.2023.00345 (date of access: 19.12.2023).

16. Wang J. Reinforcement Learning for Adaptive AI in Survival Games [Electronic resource] / J. Wang, M. Kim // Game AI and Intelligent Systems. – 2021. – Vol. 29, no. 7. – P. 58–72. – Mode of access: https://doi.org/10.5678/gais.2021.29.7.0058 (date of access: 19.12.2023).

17. Patel R. AI-Based Difficulty Scaling in 2.5D Survival Scenarios [Electronic resource] / R. Patel, S. Gupta // Sensors. – 2022. – Vol. 22, no. 14. – P. 5876. – Mode of access: https://doi.org/10.3390/s22145876 (date of access: 19.12.2023).

18. Lopez F. Adaptive Gameplay Mechanics in Modern Video Games [Electronic resource] / F. Lopez, D. Robinson // Journal of Entertainment Technology. – 2020. – Vol. 18, no. 5. – P. 123–137. – Mode of access: https://doi.org/10.5678/jet.2020.18.5.0123 (date of access: 19.12.2023).

19. Tanaka H. AI Algorithms for Dynamic Survival Game Difficulty [Electronic resource] / H. Tanaka, R. Yamada // IEEE Transactions on Computational Intelligence and AI in Games. – 2023. – Vol. 15, no. 1. – P. 45–60. – Mode of access: https://doi.org/10.1109/tciaig.2023.00123 (date of access: 19.12.2023).

20. Starbound Review: A Stellar Voyage of Crafting, Conflict, and Discovery URL: https://www.ign.com/articles/2016/07/28/starbound-review (date of access: 08.02.2022).

21. Conan Exiles Review: The Strengths of This Survival Game Do Not Lie Where You'd Expect URL: https://www.ign.com/articles/2018/05/18/conan-exiles-review (date of access: 21.04.2020).

22. Surviving the Apocalypse: The Unforgiving Difficulty and Structure of Project Zomboid and its Impact on Player Engagement URL: https://medium.com/@tkooliya/surviving-the-apocalypse-the-unforgiving-difficulty-and-structure-of-project-zomboid-and-its-3a98895aca88 (date of access: 04.12.2023).

23. Exploring Dynamic Difficulty Adjustment in 2.5D Games URL: https://www.gamasutra.com/blogs/2023/04/15/exploring-dynamic-difficulty-adjustment-in-2-5d-games (date of access: 12.04.2023).

24. Designing AI Behavior for Real-Time Strategy Games URL: https://www.gameai.com/designing-ai-behavior-for-rtS-games-2023 (date of access: 03.05.2023).

25. The Role of AI in Adaptive Game Difficulty URL: https://www.gamesbeat.com/ai-in-adaptive-game-difficulty-2024 (date of access: 07.06.2024).

26. AI Algorithms for Procedural Content Generation in Games URL: https://www.gameprogramming.com/ai-algorithms-procedural-content-2023 (date of access: 02.07.2023).

27. Optimizing AI for Survival Game Mechanics URL: https://www.survivalgamesai.com/optimizing-ai-for-survival-games (date of access: 15.08.2023).

28. Reinforcement Learning in Game AI URL: https://www.mlforgames.com/reinforcement-learning-in-game-ai-2023 (date of access: 29.09.2023).

29. AI Difficulty Scaling in Strategy Games URL: https://www.advancedgamesai.com/ai-difficulty-scaling-strategy-games (date of access: 13.10.2023).

30. Machine Learning Techniques in Adaptive Gameplay for Survival Games URL: https://www.machinelgaming.com/adaptive-gameplay-survival-ai (date of access: 27.11.2023).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО - КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

Методика керування поведінкою штучного інтелекту ворогів у грі жанру "стратегія в реальному часі"

Виконав: студент групи ПДМ-63 Олексенко Роман Григорович

Керівник: доктор філософії, доцент кафедри ІПЗ Дібрівний Олесь Андрійович

Київ - 2025

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення якості ігрового процесу у стратегіях в реальному часі за рахунок розробки методу дій штучного інтелекту.
Об'єкт дослідження: процес взаємодії штучного інтелекту з гравцем у стратегіях в реальному часі.
Предмет дослідження: метод дій штучного інтелекту в грі жанру "стратегія в реальному часі".

АКТУАЛЬНІСТЬ РОБОТИ

- Зростаючі вимоги гравців до складності ігор, оскільки сучасні геймери шукають більш інтелектуальних супротивників.
- Популярність адаптивних ігор, де рівень ворогів автоматично підлаштовується під прогрес гравця.
- Покращена взаємодія завдяки AI, що адаптується до стилю гри користувача, створюючи динамічний і збалансований геймплей.

3

Порівняльний аналіз методів дій штучного інтелекту

Метод	Адаптивність до дій гравця	Вміння навчатись на досвіді	Підтримка складних стратегій	Простота реалізації	Швидкість обчислень
Метод A*	-	-	-	+	+
Стратегічне дерево	+	-	+	-	-
Поведінкові дерева (BT)	-	-	+	+	+
FSM (Скінченні автомати)	-	-	-	+	+
Машинне навчання (ML)	+	+	+	-	-
Баєсові мережі (BN)	+	+	+	-	-
Метод Mentisys (Поведінкові дерева, Стратегічне дерево, Баєсові мереж)	+	-	+	+	+

4

Представлення математичної формули методу дій штучного інтелекту ворога

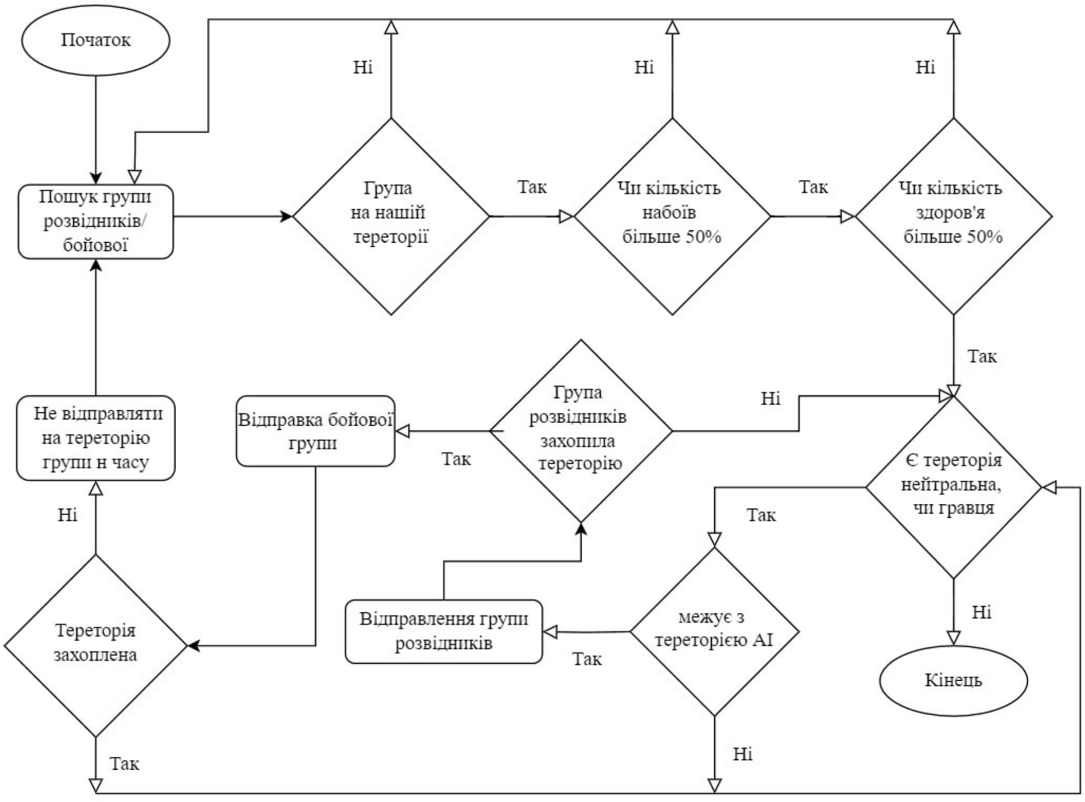
Нехай:
T — статус території групи (0 = нейтральна, 1 = територія гравця, 2 = територія AI).
A — поточна кількість боєприпасів у групи.
H — поточне здоров'я групи.
S — чи межує територія групи з територією AI (0 = не межує, 1 = межує).
C — чи захоплена територія групи (0 = не захоплена, 1 = захоплена).
E — чи є ворожі території навколо (0 = немає, 1 = є).
Основна формула для визначення дій AI:
$D_{AI} = w_1 * f(T) + w_2 * f(A) + w_3 * f(H) + w_4 * f(S) + w_5 * f(C) + w_6 * f(E)$
де:
f(T) = 1 для території гравця або AI, 0 для нейтральної території.
f(A) = 1, якщо A > 50%, 0, якщо A ≤ 50%.
f(H) = 1, якщо H > 50%, 0, якщо H ≤ 50%.
f(S) = 1, якщо територія межує з територією AI, 0, якщо не межує.
f(C) = 1, якщо територія захоплена, 0, якщо не захоплена.
f(E) = 1, якщо є ворожі території навколо, 0, якщо немає ворожих територій.
Загальна складність дій AI:

$$D_{AI} = \sum_{i=1}^{6} w_i * f_i$$

де:
wi : вага i-го фактора;
fi : значення функції для i-го фактора.

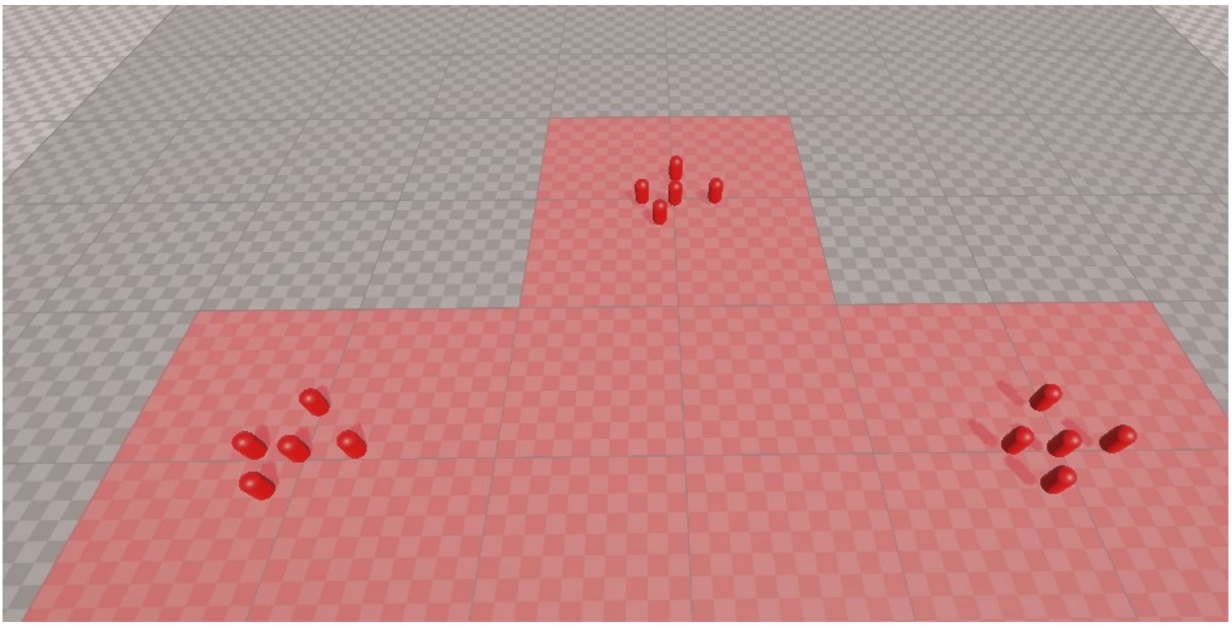
5

Метод дій штучного інтелекту ворогів



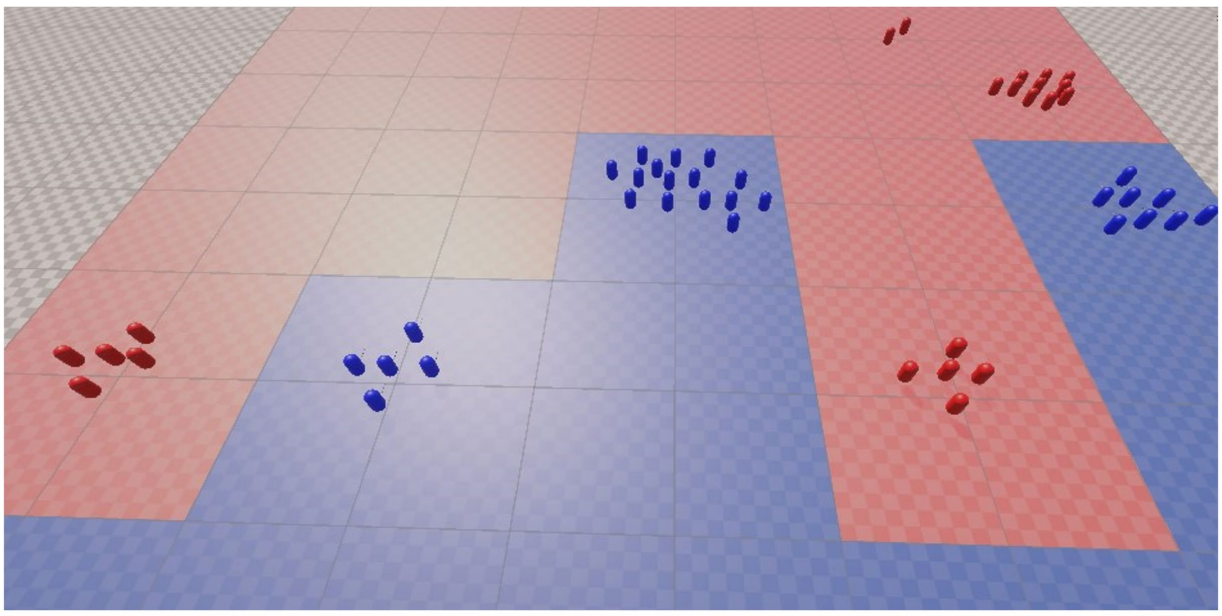
6

ЕКРАННА ФОРМА MENTISYS



7

ЕКРАННА ФОРМА MENTISYS



8

Порівняльний аналіз методів дій штучного інтелекту

Метод	Точність моделювання (%)	Швидкість обчислень (мс/кадр)	Стійкість до помилок (%)
Метод Mentisys	93%	35	90%
Метод A*(A-star)	90%	100	85%
FSM	80%	50	90%
Стратегічне дерево	85%	150	80%
Підкріплювальне навчання	95%	200	75%
Метод K-середніх	80%	30	70%

9

ВИСНОВКИ

1. Проаналізовано основні методи ШІ для ігор жанру RTS, включаючи A*(A-star) , FSM (Кінцеві автомати стану), Стратегічне дерево, Підкріплювальне навчання та Метод K-середніх (K-means clustering).

2. Визначено переваги та недоліки кожного методу. Щоб створити метод який буде як і адаптивний до дій гравця, так і не витрачати велику кількість ресурсів для обчислення.

3. Розроблено адаптивний метод, що враховує рівень боєзапасу, здоров'я та дії гравця, дозволяючи ШІ динамічно змінювати поведінку в залежності від ситуації на полі бою.

4. Впроваджено систему систему пошуку для захоплення тереторії підкріплення бойовою групую, якщо розвідники не змогли захопити тереторію. Якщо й бойова група не може захопити тереторію, то ші пропускає її, щоб спробувати взяти в оточення цю тереторію.

5. Метод Mentisys показав найкращі результати: точність моделювання 93%, стійкість до помилок 90%, швидкість обчислень 35 мс/кадр, що робить його ефективним для реального часу.

10

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Статті:

1. Олексенко Р.Г. Дібрівний О.А. Застосування методу керування поведінкою штучного інтелекту ворогів у грі жанру «Стратегія в реальному часі"// Зв'язок. №1, 2025. подана до друку.

Тези доповідей:

1. Олексенко Р.Г. Дібрівний О.А. Використання відеоігор для навчання алгоритмів штучного інтелекту. // IV Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті» – Київ: ДУІКТ, 2024. – С.168-169.
2. Олексенко Р.Г. Дібрівний О.А. Роль ігор у навчанні стратегічних навичок для людей за допомогою гри в жанрі стратегія в реальному часі// Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях» – Київ: ДУІКТ, 2024. – С.377-378.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```
using UnityEngine;

public static class DecisionMaker
{
    public static void MakeDecision(UnitGroup unitGroup, int
territoryStatus)
    {
        if (territoryStatus == 0 &&
unitGroup.HasEnoughAmmo() &&
unitGroup.HasEnoughHealth())
        {
            Debug.Log("Захоплення нейтральної території.");
            // Логіка для захоплення території
        }
        else if (territoryStatus == 1 &&
unitGroup.HasEnoughAmmo())
        {
            Debug.Log("Посилення захисту території гравця.");
            // Логіка для оборони
        }
        else if (territoryStatus == 2 &&
unitGroup.HasEnoughHealth())
        {
            Debug.Log("Атака на територію AI.");
            // Логіка для атаки
        }
        else if (!unitGroup.HasEnoughAmmo() ||
!unitGroup.HasEnoughHealth())
        {
            Debug.Log("Потрібне поповнення ресурсів.");
            // Логіка для поповнення боєприпасів або
відновлення здоров'я
        }
    }
} using UnityEngine;
using UnityEngine.AI;

public class SupplyTruck : MonoBehaviour
{
    public Transform targetGroup;
    public int ammoSupply = 100;
    private NavMeshAgent agent;

    void Start()
    {
        agent = GetComponent<NavMeshAgent>();
    }

    void Update()
    {
        if (targetGroup != null)
        {
            UnitGroup group =
targetGroup.GetComponent<UnitGroup>();
            if (group != null && group.NeedsAmmo())
            {
                agent.SetDestination(targetGroup.position);
            }
        }
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.transform == targetGroup)
        {
            UnitGroup group =
other.GetComponent<UnitGroup>();
            if (group != null)
            {
                group.ammoCount += ammoSupply;
                Debug.Log("Боєприпаси поповнені.");
            }
        }
    }
} using UnityEngine;

public class AIController : MonoBehaviour
{
    public TerritoryManager territoryManager;

    void Update()
    {
```

```
    for (int x = 0; x < territoryManager.mapSizeX; x++)
    {
      for (int y = 0; y < territoryManager.mapSizeY; y++)
      {
        int status = territoryManager.GetTerritoryStatus(x, y);

        if (status == 0 && territoryManager.IsAdjacentToEnemy(x, y))
        {
          SendReconGroup(x, y);
        }
        else if (status == 1)
        {
          DefendTerritory(x, y);
        }
        else if (status == 2)
        {
          PlanAttack(x, y);
        }
      }
    }
  }

  void SendReconGroup(int x, int y)
  {
    Debug.Log($"Відправка розвідки до ({x}, {y}).");
  }

  void DefendTerritory(int x, int y)
  {
    Debug.Log($"Захист території ({x}, {y}).");
  }

  void PlanAttack(int x, int y)
  {
    Debug.Log($"Планування атаки з території ({x}, {y}).");
  }
} using UnityEngine;
using System.Collections.Generic;

public class ReserveManager : MonoBehaviour
{
  public List<UnitGroup> reserve = new List<UnitGroup>();

  public void UpdateReserve(UnitGroup group, int territoryStatus)
  {
    if (group.health > 50 && group.ammoCount > 50 && territoryStatus == 1) // Перевірка умов
    {
      if (!reserve.Contains(group))
      {
        reserve.Add(group);
        Debug.Log("Група додана до резерву.");
      }
    }
    else
    {
      if (reserve.Contains(group))
      {
        reserve.Remove(group);
        Debug.Log("Група видалена з резерву.");
      }
    }
  }
} using UnityEngine;

public class NearestTerritoryFinder : MonoBehaviour
{
  public TerritoryManager territoryManager;

  public Vector2Int FindNearestNeutralOrEnemyTerritory(Vector2Int currentPosition)
  {
    float minDistance = Mathf.Infinity;
    Vector2Int nearestTerritory = currentPosition;

    for (int x = 0; x < territoryManager.mapSizeX; x++)
    {
      for (int y = 0; y < territoryManager.mapSizeY; y++)
      {
        if (territoryManager.GetTerritoryStatus(x, y) == 0) // Нейтральна або ворожа територія
        {
          float distance = Vector2Int.Distance(currentPosition, new Vector2Int(x, y));
          if (distance < minDistance)
          {
            minDistance = distance;
```

```
                nearestTerritory = new Vector2Int(x, y);
            }
        }
    }
}

Debug.Log($"Найближча територія до {currentPosition} знаходиться на ({nearestTerritory.x}, {nearestTerritory.y}).");
return nearestTerritory;
}

public void SendReconGroup(UnitGroup reconGroup, Vector2Int target)
{
reconGroup.MoveTo(target); // Метод переміщення групи (реалізується у скрипті UnitGroup)
Debug.Log("Відправка розвідників до найближчої території.");
}
} using UnityEngine;
using System.Collections;

public class BattleGroupManager : MonoBehaviour
{
public ReserveManager reserveManager;
public TerritoryManager territoryManager;
public float cooldownTime = 60f; // Час, протягом якого AI не буде відправляти групи

private HashSet<Vector2Int> cooldownTerritories = new HashSet<Vector2Int>();

public void SendBattleGroup(Vector2Int target)
{
if (reserveManager.reserve.Count > 0 && !cooldownTerritories.Contains(target))
{
UnitGroup battleGroup = reserveManager.reserve[0]; // Вибір першої групи з резерву
reserveManager.reserve.RemoveAt(0);
battleGroup.MoveTo(target);
Debug.Log("Бойова група відправлена.");

StartCoroutine(HandleCooldown(target));
}
}

private IEnumerator HandleCooldown(Vector2Int target)
{
cooldownTerritories.Add(target);
yield return new WaitForSeconds(cooldownTime);
cooldownTerritories.Remove(target);
Debug.Log($"Територія ({target.x}, {target.y}) знову доступна для атак.");
}
} using UnityEngine;
using UnityEngine.AI;

public class UnitGroup : MonoBehaviour
{
public int health;
public int ammoCount;
private NavMeshAgent agent;

void Start()
{
agent = GetComponent<NavMeshAgent>();
}

public void MoveTo(Vector2Int targetPosition)
{
Vector3 worldPosition = new Vector3(targetPosition.x, 0, targetPosition.y); // Перетворення координат
agent.SetDestination(worldPosition);
Debug.Log($"Група переміщується до ({targetPosition.x}, {targetPosition.y}).");
}
}
```