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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка методики генерації сценарію поведінки супротивника в іграх жанру покрокової стратегії»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
(код, найменування спеціальності)
освітньо-професійної програми «Інженерія програмного забезпечення»
(назва)

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

_____ Єгор КУЗЬМЕНКО
(підпис)

Виконав: здобувач вищої освіти група ПДМ-64
Єгор КУЗЬМЕНКО

Керівник: Олесь ДІБРІВНИЙ
доктор філософії (PhD)

Рецензент: _____
науковий ступінь, вчене звання Ім’я, ПРІЗВИЩЕ

Київ 2024

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
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кузьменку Єгору Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Розробка методики генерації сценарію поведінки супротивника в іграх жанру покрокової стратегії

керівник кваліфікаційної роботи: Олесь ДІБРІВНИЙ доктор філософії (PhD),

затверджені наказом Державного університету інформаційно-телекомунікаційних технологій «19» жовтня 2023 року №145.

2. Строк подання кваліфікаційної роботи «29» грудня 2023 р.

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
1. Аналіз наявних математичних методів та технологій для розробки методики генерації сценарію поведінки супротивника в іграх жанру покрокової стратегії.
2. Дослідження інформаційних технологій за темою.
3.Розробка сценарію поведінки супротивника в іграх жанру покрокової стратегії.
4.Опис проектування системи.
5. Опис використаних технології.

5. Перелік графічного матеріалу: *презентація*

1. Титульний слайд.
2. Мета, об’єкт та предмет дослідження.
3. Існуючі методи вирішення задачі генерації сценарію поведінки супротивника в іграх жанру покрокової стратегії.
4. Попарне порівняння перемог існуючих методів відносно один одного
5. Сценарій поведінки супротивника.
6. Математична модель розрахунку найбільшої цінності дії.
7. Критерії якості штучного інтелекту супротивника в грі жанру покрокової стратегії.
8. Алгоритм роботи реалізованого супротивника.
9. Результати дослідження.
10. Результати тестування розробленого методу в порівнянні з RHEA.
11. Апробація результатів дослідження.
12. Висновки.

6. Дата видачі завдання «19» жовтня 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	19.10-05.11.23	
2	Узгодження плану роботи та списку використаних джерел з науковим керівником	06.11-07.11.23	
3	Аналіз існуючих математичних рішень	08.11-12.11.23	
4	Дослідження інформаційних технологій в напрямку створення штучного інтелекту супротивника	13.11-26.11.2023	
5	Проєктування сценарію поведінки супротивника в покрокових стратегіях	27.11-03.12.2023	
6	Розробка модифікованої математичної моделі та проведення дослідження	04.12-10.11.2023	
7	Оформлення роботи: вступ, висновки, реферат	11.12-20.12.2023	
8	Розробка демонстраційних матеріалів	21.12-29.12.2023	

Здобувач вищої освіти _____ Єгор КУЗЬМЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Олесь ДІБРІВНИЙ
(підпис)

РЕФЕРАТ

Текстова частина магістерської 74 с., 40 рис., 10 фор., 7 табл., 25 джерел.

Мета дослідження: підвищення якості роботи супротивника в грі жанру покрокової стратегії за рахунок створення власного методу.

Об'єкт дослідження: процес генерації поведінки супротивника в грі жанру покрокової стратегії.

Короткий зміст роботи: алгоритми штучного інтелекту супротивника в грі.

Результати проведеного наукового дослідження дають зрозуміти, що використані математичні методи, алгоритмічні моделі, та програмні засоби надають можливість вдосконалити генерацію сценарію поведінки супротивника в іграх жанру покрокової стратегії.

Практичні значення отриманих результатів полягає в тому, що на основі проведених теоретичних досліджень було розроблено новий метод генерації сценарію поведінки супротивника в іграх жанру покрокової стратегії. Результати цього дослідження надалі дають змогу пришвидшити роботу алгоритмів штучного інтелекту генерації рівнів.

КЛЮЧОВІ СЛОВА: ШТУЧНИЙ ІНТЕЛЕКТ, ПОКРОКОВА СТРАТЕГІЯ, СЦЕНАРІЙ СУПРОТИВНИКА.

ABSTRACT

Text part of the master’s qualification work: 74 pages, 40 figures, 10 formulas, 7 tables, 25 references.

The purpose of the work: to improve the quality of the opponent's work in the game of the turn-based strategy genre due to the creation of his own method.

Object of research: the process of generating the opponent's behavior in a turn-based strategy game.

Subject of research: artificial intelligence algorithms of the opponent in the game. The purpose of the study: to improve the quality of the opponent's work in the game of the turn-based strategy genre due to the creation of his own method.

Summary of the work: the results of the conducted scientific research make it clear that the used mathematical methods, algorithmic models and software tools make it possible to improve the generation of the scenario of the opponent's behavior in turn-based strategy games.

The practical significance of the obtained results is that a new method of generating the scenario of the opponent's behavior in turn-based strategy games was developed on the basis of theoretical studies conducted. The results of this study further allow us to come to the work of algorithms for generating levels of artificial intelligence.

KEY WORDS: ARTIFICIAL INTELLIGENCE, TURN STRATEGY GAME, OPPONENT SCENARIO.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня магістра

Направляється здобувач КУЗЬМЕНКО Є.О. до захисту кваліфікаційної роботи

за спеціальністю 121 Інженерія програмного забезпечення

освітньо-професійної програми «Інженерія програмного забезпечення»

на тему: «Розробка методики генерації сценарію поведінки супротивника в іграх жанру покрокової стратегії».

Кваліфікаційна робота і рецензія додаються.

Директор ННІ ІТ _____ Андрій БОНДАРЧУК
(підпис)

Висновок керівника кваліфікаційної роботи

Магістерська робота, що виконав студент Кузьменко Є.О., відповідає завданню та завершена вчасно. В ході виконання роботи студентом була розроблена методика генерації сценарію поведінки супротивника в іграх жанру покрокової стратегії. Магістром проаналізовано всі наявні методи розробки сценарію поведінки супротивника та покращено роботу розробленої методики. По даним результатам виконаної роботи можна зробити висновок, що студент вдало використовує знання, отримані в Державному університеті інформаційно-телекомунікаційних технологій, правильно і чітко формулює завдання, може розв'язувати їх та доводить рішення до кінцевого результату. При вирішенні поставлених завдань приймає грамотні обґрунтовані рішення. При оформленні текстових та графічних матеріалів застосовувались сучасні інформаційні технології і офісні пакети.

Все це дозволяє оцінити виконану кваліфікаційну роботу здобувача Кузьменка Є.О. на оцінку «добре» та присвоїти йому кваліфікацію магістр з інженерії програмного забезпечення.

Керівник кваліфікаційної роботи _____ Олесь ДІБРІВНИЙ
(підпис)

«____» _____ 202__ року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач Кузьменко Є.О. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедри ІПЗ _____ Ірина ЗАМРІЙ
(підпис)

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну магістерську роботу

здобувача вищої освіти Кузьменка Єгора Олексійовича

на тему «Розробка методики генерації сценарію поведінки супротивника в іграх жанру покрокової стратегії »

Актуальність.
В сучасному світі ігрова індустрія є одним з найбільш стрімко зростаючих галузей. На сьогодення велика увага спрямовується на розвиток ігор та їх інтелектуальні аспекти, що робить дане дослідження актуальним та важливим для розвитку штучного інтелекту для покрокових стратегій.
Позитивні сторони.
1. Детально розглянуті та аргументовані основні аспекти створення методики генерації сценарію поведінки штучного інтелекту в жанрі покрокових стратегій.
2. Авторський погляд на проблему та впровадження нових підходів у вирішенні задачі створення штучного інтелекту супротивника в іграх покрокової стратегії.
3. Розроблена методика була протестована на практиці через проведення тестових ігор та порівняння результатів показало її успішність.

Недоліки.
1. В роботі можна відзначити потребу у більш глибокому аналізі деяких аспектів розробленої методики.
2. Варто звернути увагу на можливість розширення та поглиблення деяких теоретичних підходів, що може збільшити повноту розгляду теми.

Відзначені зауваження не впливають на загальну позитивну оцінку кваліфікаційної магістерської роботи.

Висновок: *кваліфікаційна робота на здобуття ступеня магістра заслуговує оцінку "добре", а здобувач* ***Кузьменко Єгор Олексійович*** *заслуговує присвоєння кваліфікації магістр з інженерії програмного забезпечення.*

Рецензент: _____ _____
науковий ступінь, вчене звання *підпис* *Ім'я, ПРІЗВИЩЕ*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

RHEA - Rolling Horizon Evolution Algorithm
MCTS - Monte Carlo Tree Search
OSLA - One step look ahead
RB - Rule Based
ШІ - Штучний Інтелект
FM - Fenite machine

ВСТУП

Покрокові стратегічні ігри завжди викликали особливий інтерес в гравців та дослідників ігрової індустрії через їх складність та потенційно безмежні можливості відтворення стратегій і тактик. У цьому контексті великий акцент приділяється реалізації штучного інтелекту, що виступає як опонент гравця та розвиває власні стратегії дій. Однак існуючі методи реалізації штучного інтелекту мають свої обмеження та недоліки.

Завдання полягає в таких завданнях як:

1. Аналіз існуючих методів та підходів створення штучного інтелекту супротивника в іграх жанру покровові стратегії.
2. Створення власного алгоритму роботи штучного інтелекту на основі розробленого сценарію.
3. Впровадження алгоритму в програмну розробку ігрової моделі.
4. Проведення тестування розробленого методу.
5. Порівняння отриманих результатів з вже існуючими методами.

Об’єкт дослідження: процес генерації поведінки супротивника в грі жанру покрокової стратегії.

Предмет дослідження: алгоритми штучного інтелекту супротивника в грі.

Мета дослідження: підвищення якості роботи супротивника в грі жанру покрокової стратегії за рахунок створення власного методу.

Практичне значення полягає в створенні програмного забезпечення штучного інтелекту супротивника , що діє за заданим сценарієм та може знаходити оптимальну стратегію дій аналізуючи надану систему цінностей.

Актуальність даної магістерської роботи полягає у зростаючому інтересі до розробки ігор та застосуванні штучного інтелекту в геймінгу. Сучасна геймінг-індустрія зростає швидко, збільшуючи вимоги до якості ігрового процесу та його інтелектуальної складності. Розробка нових методик створення штучного інтелекту для покрокових стратегічних ігор стає основним завданням у сфері дослідження і розвитку. Постійне покращення технологій, а також зростаючі

можливості обчислювальних систем вимагають більш ефективних та складних алгоритмів штучного інтелекту для досягнення високої грейдерної здатності, оптимізації стратегій та підвищення реалістичності поведінки супротивників у грі. Таким чином, дана робота є актуальною, оскільки спрямована на розв'язання актуальних завдань розробки штучного інтелекту для ігор, які відображають останні тенденції у галузі геймінгу та впроваджують новаторські підходи до створення ігрових інтерфейсів та ігрових стратегій.

Наукова новизна даної магістерської роботи базується на розробці та застосуванні власного алгоритму штучного інтелекту для створення супротивника в покрокових стратегічних іграх. Відмінністю цього дослідження є вдосконалення існуючих методів та підходів штучного інтелекту у гральній індустрії, а також розробка нового алгоритму, який забезпечить підвищену якість поведінки супротивника в грі.

Дослідження включає у себе створення алгоритму штучного інтелекту, який базується на власному сценарії дій та може розрізняти та аналізувати складні контекстні ситуації в грі. Такий підхід спрямований на досягнення більшої ігрової глибини та інтелектуальної складності супротивника, що робить дане дослідження новаторським у розробці ігор та штучного інтелекту для них.

1 СТВОРЕННЯ ТА МОДЕЛЮВАННЯ СЦЕНАРІЮ ПОВЕДІНКИ СУПРОТИВНИКА В ІГРАХ ЖАНРУ ПОКРОКОВОЇ СТРАТЕГІЇ

1.1 Формулювання поняття жанру покрокової стратегії та особливостей супротивників в грі

Покрокова стратегія - піджанр тактичних ігор, які базуються на ігровій механіці за принципом покрокового режиму. У таких іграх гравці почергово виконують свої ходи. Цей тип гри зазвичай має повільніший темп та акцентується на стратегічному плануванні, аніж на швидкій дії. Багато тактичних покрокових ігор базуються на історичних подіях, часто включаючи військові одиниці, такі як піхота, кіннота та артилерія. Жанр покрокової стратегії почав набирати популярність на початку 1990-х років з виходом декількох популярних ігор, таких як X-Com: UFO Defense[1] та Jagged Alliance[2]. Ці ігри отримали продовження та імітації, і жанр залишається популярним й досі. Останнім часом жанр відзначається зростанням популярності, частково завдяки успіху перезавантаження XCOM та інших схожих ігор.

Рис.1.1. Інтерфейси X-Com: UFO Defense та Jagged Alliance

У покрокових стратегіях гравці та їх бойові одиниці взаємодіють. Бойовий час поділяється на фрагменти (ходи), під час яких окремі одиниці можуть

рухатися в певному порядку.

Термін «покроковий бойовий режим» відноситься до взаємодії між гравцями та їх одиницями в настільних або відео іграх. Бойовий час поділяється на фрагменти (ходи), під час яких окремі одиниці діють у певному порядку. Всім на полі бою дається час для дій, поки гравець розглядає свій наступний хід.

Жанр покрокової стратегічної ігри є досить важливим і часто використовується для аналізу питань теорії ігор. Ігри жанру покрокової стратегії відносяться до ряду стохастичних ігор.

Стохастична гра - це гра, в якій гравці приймають рішення в умовах невизначеності або випадковості. Гра G включає в себе множину станів S, множину гравців [m], множину можливих дій для кожного гравця (A1, A2, ..., Am), функції винагороди rj і перехідне ядро P.

Стохастична гра G називається покроковою стохастичною грою, якщо в кожному стані s ∈ S є один гравець i ∈ [m] (називається контролером стану s і позначається i = cr(s)), дія якого at s повністю визначає винагороду і перехід до наступного стану. Формально для всіх j ∈ [m] існує деяка функція r′ j : S ×(A1⊔· · ·⊔Am) → [−1, 1] і деяке перехідне ядро P ′: S ×(A1⊔· · · ⊔Am) →Δ(S), так що rj (s, a) = r′j(s, acr(s)) для всіх s ∈ S, j ∈ [m], a = (a1, . . . , am) ∈ A, і P(·|s, a) = P ′ (·|s, acr(s)) для всіх s ∈ S, a ∈ A.

Функції винагороди: для кожного гравця j і кожного стану s, існують функції rj, які визначають винагороду, яку гравець j отримує при виборі конкретної дії, а в стані s. Функції rj визначаються функціями r′j, які залежать від стану s і дії гравця cr(s).

Перехідне ядро: Це P ′, яке визначає ймовірності переходу від одного стану до іншого в залежності від дій гравців і контролера стану s.[3]

Головна мета гри полягає в досягненні стратегічної переваги шляхом розміщення власних одиниць, використання тактичних здібностей та ведення боротьби з противником. Гравці мають можливість керувати своїми військами, будувати структури або бази, розвивати свої ресурси та використовувати унікальні властивості своїх персонажів або одиниць для досягнення перемоги.

Супротивник в покроковій стратегічній грі виступає як програмно реалізований гравець, який застосовує вивчені стратегії та приймає рішення для досягнення максимального рівня складності та цікавого ігрового рішення для гравця.

Від якості створюваного супротивника залежить ефектність та цікавість покрокової стратегічної гри. Він повинен бути збалансованим, логічним, не передбачуваним заздалегідь і не виконувати одні й ті самі дії змушуючи гравця йти до іншої гри. Тому важливо на етапах створення покрокової стратегії приділити цьому більше часу задля покращення якості гри.

1.2 Аналіз наукових та практичних підходів до створення поведінки супротивника в іграх жанру покрокової стратегії

Через свою потенційно величезну економічну цінність виробництво ігор на основі штучного інтелекту останніми роками привернуло велику увагу науковців і промисловості. Штучний інтелект просунувся від використання простих правил до різноманітних ігор, і він довів, що є грізним суперником, перемагаючи в деяких випадках людей-чемпіонів. [4]

Стохастичні ігри, які є покроковими або нескінченними горизонтами, є більш складними випадками, і дослідження таких ігор все ще є відкритим викликом.

Одним із застосувань штучного інтелекту в грі є метод кінцевого автомата.

FSM (Finite State Machine) — це метод реалізації штучного інтелекту, який використовується для прийняття рішення щодо NPC (не гральний персонаж). Застосування FSM, яке часто зустрічається, полягає в тому, щоб сформувати NPC з інтелектом, щоб він міг реагувати на персонажа гравця, щоб NPC, здавалося, міг думати. Ігри мають різні жанри і все більше змінюються відповідно до розвитку апаратних і програмних технологій.

Скінченний автомат (FSM) — це методологія проектування системи керування, яка описує поведінку або принципи роботи системи за допомогою кінцевого автомата (стану), події (інциденту) та дії. У кінцевій машині система

займає один стан. Система переключиться або перейде в інший стан, якщо отримає вхідні дані для певних подій. Система продовжуватиме виконувати ті самі дії в стані, доки система не отримає певні події або від зовнішніх пристроїв, або від самих компонентів системи. [5]

Традиційний FSM вимагає, щоб розробники завершили всі очікувані стани та переходи архітектури на етапі розробки, а код важко піддається змінам після загально завершених етапів створення програмного продукту. Таким чином, FSM схильна стикатися з проблемами комбінаторного вибуху (стани та переходи FSM важко контролювати, коли складність навколишнього середовища зростає). У різних середовищах або різних іграх розробникам доводиться перекодувати станції та переходи раніше, а не використовувати їх повторно.

Принцип дії FSM добре описує суть базових правил стохастичних ігор на основі яких будується жанр покрокової стратегії. Тому основа методу для штучного інтелекту супротивника має так само спиратися на розуміння суті роботи скінченного автомату.

Також часто для створення супротивників в іграх часто використовують такі методи як: A* та MCTS.

Алгоритм A* – це алгоритм оптимізації, який мінімізує вартість шляху. Вартість f(n) обчислюється за рівнянням[6]:

Однак алгоритм A* не ідеально підходить для відеоігор. Через обмежений час пошуку агентам важко знайти найкращий шлях, окрім неоптимального. Коли агенти опиняються в глухому куті, вони можуть крутитися і не можуть знайти вихід. Це можна вирішити додавши агентам здатність до навчання. Іншими словами, якщо агенти зможуть згадати де вони заходять у глухий кут, невдач можна уникнути. Під час тренувальної фази агент запам'ятовує розташування точок відгалуження та ефективні дії.

Алгоритм пошуку дерев Монте-Карло (MCTS) є одним із найефективніших алгоритмів і агентом гри зі штучним інтелектом. Це різновид пошукової техніки, яка не спирається на професійні польові знання. Тому на нього не потрібно вивчати багато знань про предметну область, що може заощадити час на розробку. У

системі NPC, заснованій на традиційному FSM, робот дає фіксований метод реагування на атаку гравців, після чого гравці швидко вивчають його поведінку і легко знаходять стратегію для перемоги над штучним інтелектом. У порівнянні з FSM, MCTS може більш гнучко реагувати на дії гравців. Алгоритм MCTS був створений для вирішення проблеми повторюваності FSM. Принцип роботи MCTS полягає в тому, щоб визначити всі операції, доступні для поточної ситуації. Потім для кожної можливої дії буде проаналізовано, як гравець може відповісти. Після цього етапу він розглядає всі можливі дії для гравця і те, які відповіді він може зробити тощо. В основному існує 4 етапи у виконанні MCTS, а саме вибір, розширення, моделювання та зворотне поширення. Спочатку на дереві пошуку є тільки один вузол і він потрібен, щоб прийняти рішення з цієї ситуації. Кожен вузол пошуку містить три типи основної інформації: представлену ситуацію, кількість відвідувань і сукупний бал.

Також згадується комбінація алгоритму A* та алгоритму Q-навчання. Агент може змінювати ціль за допомогою алгоритму Q-навчання, щоб підвищити ефективність. Дерево пошуку Монте-Карло, також є свого роду вдосконаленим алгоритмом, який може замінити традиційний FSM і вибирати вузол з найвищою частотою доступу, а не всі вузли.

Найбільш розповсюдженими та перспективними для розробки стохастичних ігор є методи розгалуженого дерева(наприклад вже розглянутий MCTS) та еволюційні. Завдяки тому, що мають ряд переваг та особливих якостей.

Інший варіант створення штучного інтелекту в грі покрокової стратегії це використання генетичних алгоритмів.

Генетичний алгоритм — це метод розв’язання задач як обмеженої, так і необмеженої оптимізації, який ґрунтується на природному відборі — процесі, який рухає біологічну еволюцію. Генетичний алгоритм неодноразово змінює популяцію окремих рішень.

Генетичні алгоритми зазвичай представлені лінійно, найчастіше такі алгоритми є двійковими. Оригінальний опис еволюційного алгоритму Голланда використовував масиви бітів. Масиви інших типів і структур можна

використовувати по суті таким же чином.

У генетичних алгоритмах та еволюційних обчисленнях кросинговер, також званий рекомбінацією, є генетичним оператором, який використовується для об’єднання генетичної інформації двох батьків для створення нових нащадків.

Мутація — це генетичний оператор, який використовується для підтримки генетичного різноманіття хромосом популяції генетичного або, загалом, еволюційного алгоритму.[7]

Ігрові еволюційні алгоритми, зокрема еволюційні алгоритми Rolling Horizon, нещодавно зуміли перевершити сучасні технології за показником виграшу в багатьох відеоіграх. Проте найкращі результати в грі значною мірою залежать від конкретної конфігурації модифікацій і гібридів, введених у кількох документах, кожна з яких додає додаткові параметри до основного алгоритму. Крім того, найкращі раніше опубліковані параметри були знайдені лише з кількох комбінацій, вибраних

Принцип роботи методу RHEA полягає в тому, що він використовує еволюційні алгоритми (EA) для розробки внутрішньо-ігрової послідовності дій на кожному ігровому тику з обмеженим часом обчислення на виконання. У цьому застосуванні EA для гри генотип описується як вектор цілих чисел довжини L (індивідуальна довжина), де кожне ціле число a знаходиться в діапазоні [0, N), де N є максимальною кількістю дій у задану гру. Це перекладається як фенотип, як послідовність дій у грі, починаючи зі стану S0, або, іншими словами, поведінка гравця. Якщо дія, вибрана на будь-якому етапі гри, є не за правилами або неможлива з розумних підстав (наприклад, входити в стіну), ігровий движок автоматично розглядає її як «нічого не робити», тому EA не бачить жодних осіб або генів у осіб, як по за правилами або нездійсненні. Щоб оцінити індивіда в цьому контексті, RHEA використовує пряму модель (FM) гри, внутрішню модель світу, щоб імітувати дії, одну за одною.[8]

Все ж таки, зосереджуючись на поставленій задачі створення методу генерації поведінки супротивника в іграх жанру покрокової стратегії оглянемо наукові методи розробки більш детально.

Для створення штучного інтелекту супротивника в грі жару покрокової стратегії розробники використовують різні методи вони можуть базуватися на таких речах як:

- A* алгоритм;
- Принципи скінченного автомату;
- Q-навчання.

Алгоритм A* – це алгоритм оптимізації, який мінімізує вартість шляху. Вартість f(n) обчислюється за рівнянням:

$$F(n) = g(n) + h(n) \quad (1.1)$$

f(n) — оціночна функція вартості переміщення з початкового стану в цільовий через стан n;

g(n) — фактична вартість переходу з початкового стану в стан n;

h(n) — оцінка вартості переміщення зі стану n до цільового стану;

h(n)обчислюється евристичною функцією для оцінки відстані до цільового вузла.

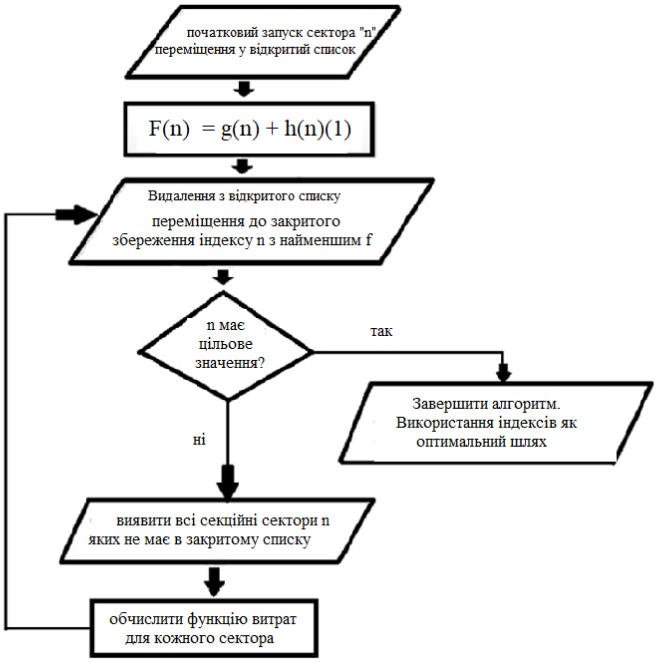


Рис. 1.2. A* алгоритм

Однак алгоритм A* не ідеально підходить для відеоігор. Через обмежений

час пошуку агентам важко знайти найкращий шлях, окрім неоптимального. Коли агенти опиняються в глухому куті, вони можуть крутитися і не можуть знайти вихід. Це можна вирішити додавши агентам здатність до навчання. Іншими словами, якщо агенти зможуть згадати де вони заходять у глухий кут, невдач можна уникнути.

Скінченний автомат (FSM) — це методологія проектування системи керування, яка описує поведінку або принципи роботи системи за допомогою кінцевого автомата (стану), події (інциденту) та дії. У кінцевій машині система займає один стан. Система переключиться або перейде в інший стан, якщо отримає вхідні дані для певних подій. Система продовжуватиме виконувати ті самі дії в стані, доки система не отримає певні події або від зовнішніх пристроїв, або від самих компонентів системи.

Рис. 1.3. Приклад алгоритму роботи скінченного автомату

Принцип дії FSM добре описує суть базових правил стохастичних ігор на основі яких будується жанр покрокової стратегії.

Q-навчання (Q-learning) - це алгоритм навчання з підкріпленням без моделі для вивчення значення дії в певному стані. Воно не потребує моделі середовища (отже, "без модельне") і може працювати з проблемами стохастичних переходів та винагород, не потребуючи адаптації.

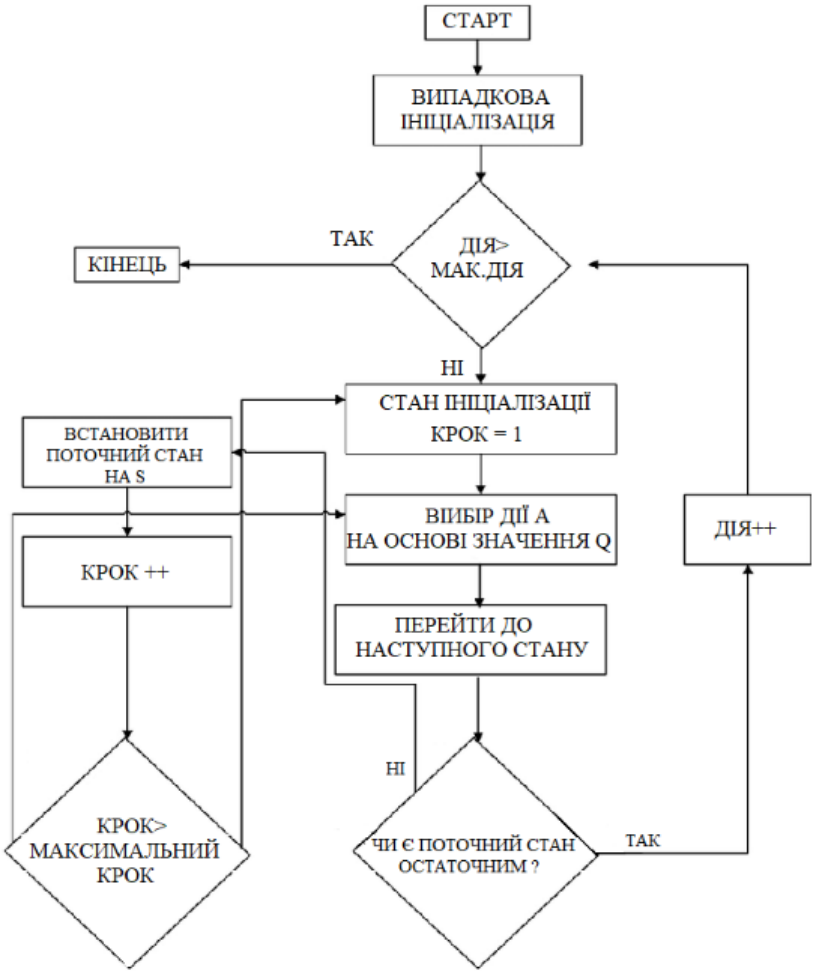


Рис. 1.4. Алгоритм Q-навчання

Для будь-якого скінченного процесу прийняття рішень Маркова, Q-навчання знаходить оптимальну стратегію в тому розумінні, що максимізує очікуване значення загальної винагороди протягом всіх послідовних кроків, починаючи з поточного стану. Q-навчання може ідентифікувати оптимальну стратегію вибору дій для будь-якого заданого скінченного процесу прийняття рішень Маркова за умови нескінченного часу дослідження і частково випадкової стратегії. "Q" вказує на функцію, яку обчислює алгоритм - очікувані винагороди за дію, взяту в певному стані.

Відповідно на основі цих трьох напрямків існують більш модифіковані і покращені методи, які використовують для створення суперника в покрокових стратегіях. До найпопулярніших методів створення штучного інтелекту можна віднести такий список:

- Rule Based (RB);
- One-Step Look Ahead (OSLA);
- Monte Carlo (MC);
- Monte Carlo Tree Search (MCTS);
- Rolling Horizon Evolutionary Algorithms (RHEA);
- Рандомні дії (RND).

Rule Based базується на виконанні дій згідно до правил гри. Є досить поширеною версією штучного інтелекту в покрокових стратегіях. Так як має перевагу в простоті свого створення. Він використовує набір жорстко заданих правил, за рахунок чого і стає досить простим у реалізації. Цей метод не потребує складних математичних моделей чи аналізу великих обсягів даних. Рішення штучного інтелекту в методі RB базуються на наборі визначених правил, які описують, які дії слід вибрати в певних ситуаціях у грі. Наприклад, якщо певний ресурс наближається до критичного рівня, може бути встановлене правило для його автоматичного збільшення.

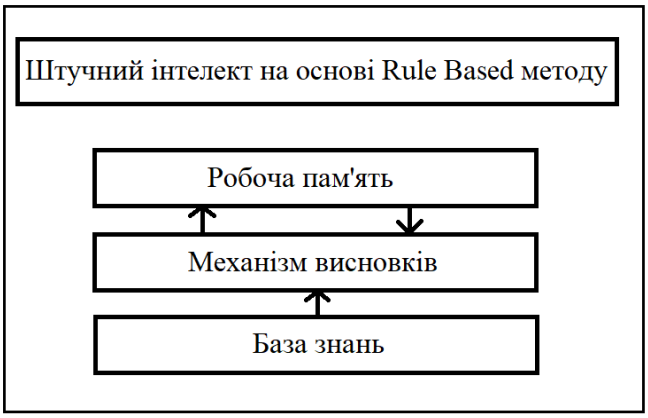


Рис. 1.5. Схема роботи методу Rule Based

Логічним обмеженням гри можна вважати адаптивність даного методу. Оскільки RB використовує фіксований набір правил, його адаптивність до змін у

грі або виявлення нових стратегій обмежена. Він може бути ефективним лише у визначених сценаріях або для обмеженого набору завдань.

Метод One-Step Look Ahead - простий метод, який використовує форвардну модель(FM) для просування поточного стану гри один раз для кожної доступної дії в даному ході. Потім кожна дія отримує бал на основі оцінки стану, до якого вона призводить. Цей стан оцінюється за допомогою функції оцінки стану, рівномірно розриваючи зв’язки випадковим чином. Потім у грі виконується дія з найвищим значенням. Якщо кілька ходів отримують високий бал, вибирається один випадково.

Вираз "одно-кроковий погляд" посилається на те, що агент дивиться лише на один крок (або хід) в майбутнє, а не глибше в дерево гри.

Метод Монте-Карло представляє великий клас обчислювальних алгоритмів, які використовують випадкові вибірки як засіб оптимізації, чисельного інтегрування та генерації вибірок за ймовірністю. Пошук за цим методом виконує розгортання (послідовність випадкових дій) від поточного стану гри (S0) до заданої глибини в межах бюджету. Останній стан, досягнутий розгортанням, оцінюється за допомогою тієї самої функції оцінки стану. MC повертає дію, яка, починаючи з S0, досягла найвищого середнього значення за всі ітерації.

Метод пошуку Монте-Карло для дерева (Monte Carlo Tree Search, MCTS) — метод пошуку який поєднує точність пошуку в дереві з узагальненістю випадкового вибіркового вивчення. Покращена версія Monte Carlo. Він отримав значний інтерес через свій вражаючий успіх у складній задачі комп'ютерної гри в Го, а також виявився корисним в ряді інших областей. Алгоритм пошуку дерев Монте-Карло (MCTS) є одним із найефективніших алгоритмів і агентом гри зі штучним інтелектом. Це різновид пошукової техніки, яка не спирається на професійні польові знання. Тому розробникам не потрібно вивчати багато знань про предметну область, що може заощадити час на розробку. У системі NPC, заснованій на традиційному FSM, робот дає фіксований метод реагування на атаку гравців, після чого гравці швидко вивчають його поведінку і легко знаходять стратегію для перемоги над штучним інтелектом. У порівнянні з FSM, MCTS може

більш гнучко реагувати на дії гравців. Алгоритм MCTS був створений для вирішення проблеми повторюваності FSM. Принцип роботи MCTS полягає в тому, щоб визначити всі операції, доступні для поточної ситуації. Потім для кожної можливої дії буде проаналізовано, як гравець може відповісти. Після цього етапу він розглядає всі можливі дії для гравця і те, які відповіді він може зробити тощо. В основному існує 4 етапи у виконанні MCTS, а саме вибір, розширення, моделювання(або ж симуляція) та зворотне поширення. Спочатку на дереві пошуку є тільки один вузол і він потрібен, щоб прийняти рішення з цієї ситуації. Кожен вузол пошуку містить три типи основної інформації: представлену ситуацію, кількість відвідувань і сукупний бал.

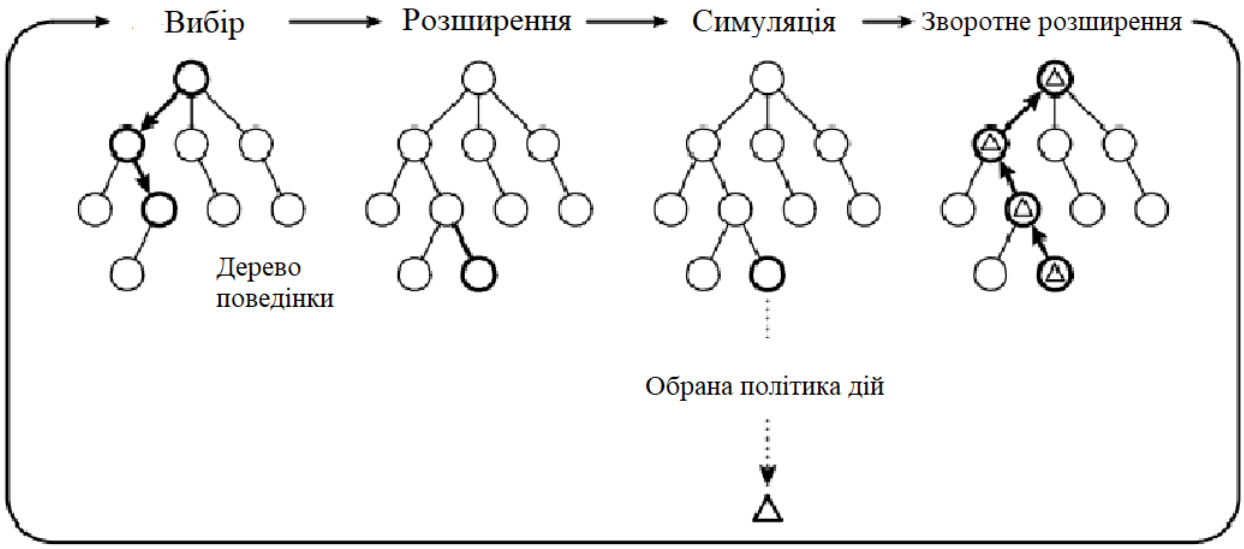


Рис. 1.6. Алгоритм дії MCTS

Головним супротивником MCTS в створенні штучного інтелекту ля покроковий стратегій, або стратегій реального часу - є еволюційні алгоритми Rolling Horizon, нещодавно зуміли перевершити сучасні технології за показником виграшу в багатьох відеоіграх. Метод RHEA використовує еволюційні алгоритми (EA) для розробки внутрішньо-ігрової послідовності дій на кожному ігровому тику з обмеженим часом обчислення на виконання. У цьому застосуванні EA для гри генотип описується як вектор цілих чисел довжини L (індивідуальна довжина), де

кожне ціле число a знаходиться в діапазоні [0, N), де N є максимальною кількістю дій у задану гру. Це перекладається як фенотип, як послідовність дій у грі, починаючи зі стану S0, або, іншими словами, поведінка гравця. Якщо дія, вибрана на будь-якому етапі гри, є не за правилами або неможлива з розумних підстав (наприклад, входити в стіну), ігровий движок автоматично розглядає її як «нічого не робити», тому EA не бачить жодних осіб або генів у осіб, як по за правилами або нездійсненні. Щоб оцінити індивіда в цьому контексті, RHEA використовує пряму модель (FM) гри, внутрішню модель світу, щоб імітувати дії, одну за одною. RHEA природно обробляє безперервні простори дії. Політика постійно вдосконалюється за допомогою розгортання все точнішої політики планування.

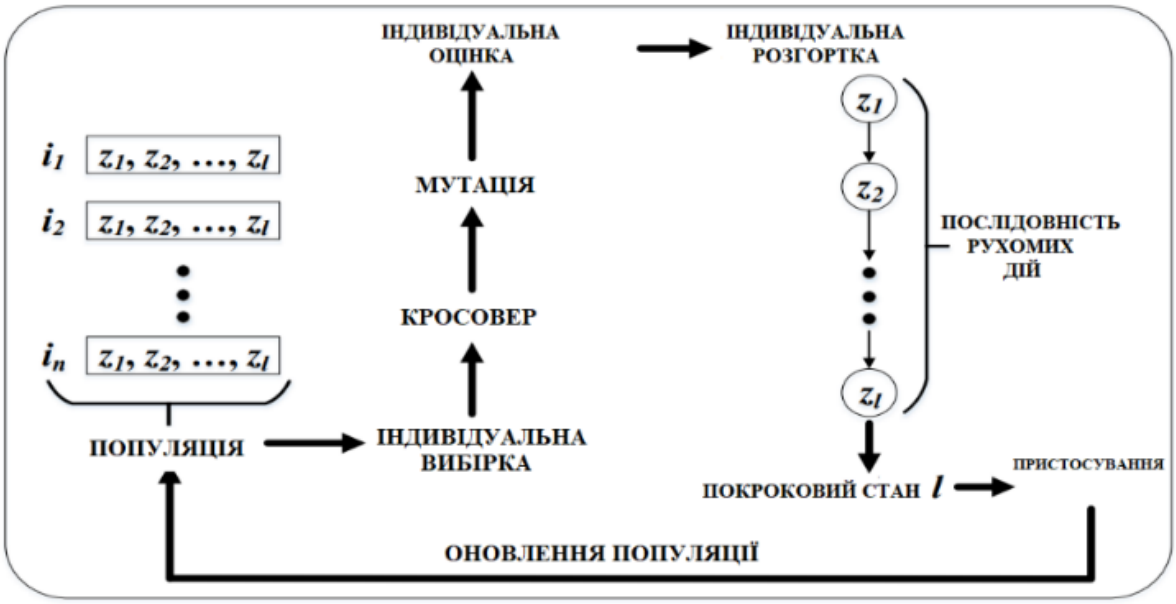


Рис. 1.7. Алгоритм роботи RHEA

Стиль гри RHEA приблизно можна описати таким чином: він приймає розумні короткострокові рішення (наприклад, атакує ворожі підрозділи, які намагаються захопити їхні міста), але йому бракує тактичної та стратегічної глибини, необхідної для координації кількох підрозділів у хід, або ефективно витрачати ресурси в різних ситуаціях.

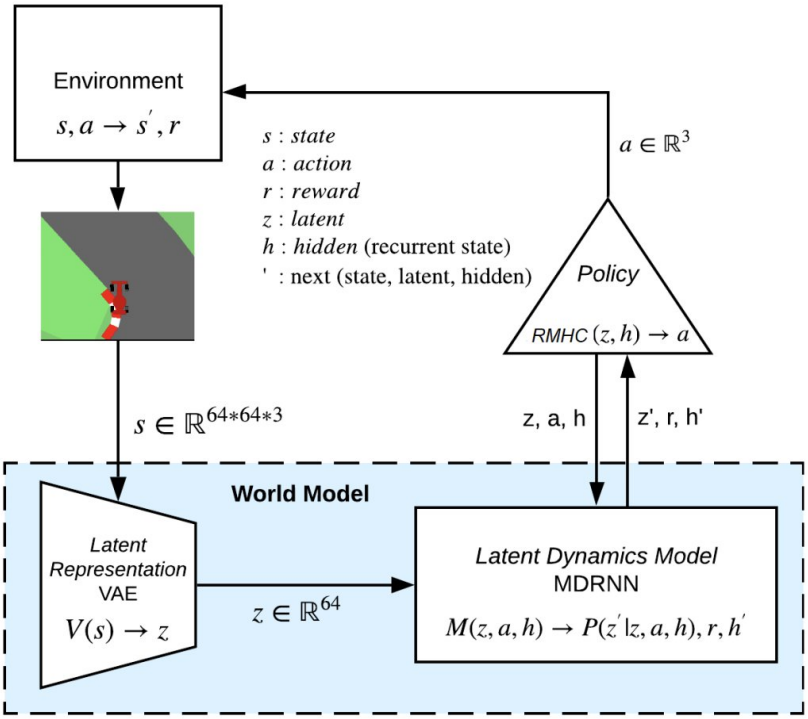


Рис. 1.8. Приклад алгоритму на основі RHEA

RHEA є досить розповсюдженим вирішенням для створення штучного інтелекту в іграх різних жанрів. Також розробники пропонують поєднувати його з іншими алгоритмами. Тим самим покращивши його слабкі сторони.

Метод випадкових дій передбачає, що ігровий супротивник буде вибирати дії зі списку доступних без будь-якої стратегічної логіки або оцінки якості виконання. Це означає, що він обиратиме дії довільно, без врахування поточного стану гри, рішень гравців чи мети перемоги.

Цей метод є найслабшим серед представлених, оскільки його стратегія полягає в випадковому виборі дій, і він не використовує ніяких обґрунтованих стратегій для досягнення успіху у грі. Без логічного підходу до вибору дій чи оцінки оптимальності виконання, випадковий метод є найменш ефективним і може привести до менш вдалих результатів у грі порівняно з іншими більш стратегічними методами.

Тому далі розгляд цього методу поряд з іншими може бути не доцільним, оскільки він не має суттєвих переваг у контексті досягнення успішних результатів у грі.

1.3 Порівняльна характеристика методів генерації поведінки супротивника в жанрі покрокової стратегії

За дослідженнями Дієго Перес-Лібана, Ю-Джен Хсу, Ставрос Еммануілідіс, Боббі Деван Акрам Халек та Ралука Д. Гайна було проведено статистичний аналіз кількості перемог відповідних методів створення штучного інтелекту супротивника. Для дослідження науковці провели понад 200 ігор. Результати яких записали у відповідну табличку. Цей статистичний аналіз, проведений Дієго Перес-Лібаном та співавторами, є цінним внеском у розуміння ефективності різних методів створення штучного інтелекту для ігор

Як ми можемо бачити метод RHEA статистично виявився більш сильним штучним супротивником. На наступному місці виявився метод заснований на правилах гри, а після Monte Carlo Three Search. Найслабшим показав себе штучний інтелект на основі випадкових дій, що говорить про те, що відсутність стратегічного підходу до ігор цього жанру є головним недоліком для створюваного методу.

Другим з кінця виявився метод One-Step Look Ahead, що було очікувано зважаючи на те, що він є більш примітивною версією методу Монте-Карло.

Таблиця 1.1

Статистика перемог інших методів

ГРАВЕЦЬ	**RHEA**	**RB**	**MCTS**	**MC**	**OSLA**	**RND**
RHEA	*	58.60%	63.00%	77.80%	74.80%	100.00%
RB	41.40%	*	56.20%	62.40%	70.20%	98.80%
MCTS	37.00%	43.80%	*	62.00%	60.80%	98.60%
MC	22.20%	37.80%	38.00%	*	54.80%	90.00%

Продовження таблиці 1.1

Статистика перемог інших методів

ГРАВЕЦЬ	RHEA	RB	MCTS	MC	OSLA	RND
OSLA	25.20%	29.80%	39.20%	45.20%	*	99.40%
RND	0.00%%	1.20%%	1.40%	1.00%	0.60%	*

Засновуючись на проведеному аналізі можемо зробити висновок, що є доцільним порівнювати результати розробленого методу саме з Rolling Horizon Evolutionary алгоритмами.

За іншими статистичними порівняннями продуктивності роботи таких методів як MCTS та RHEA було виявлено такі результати.[17]

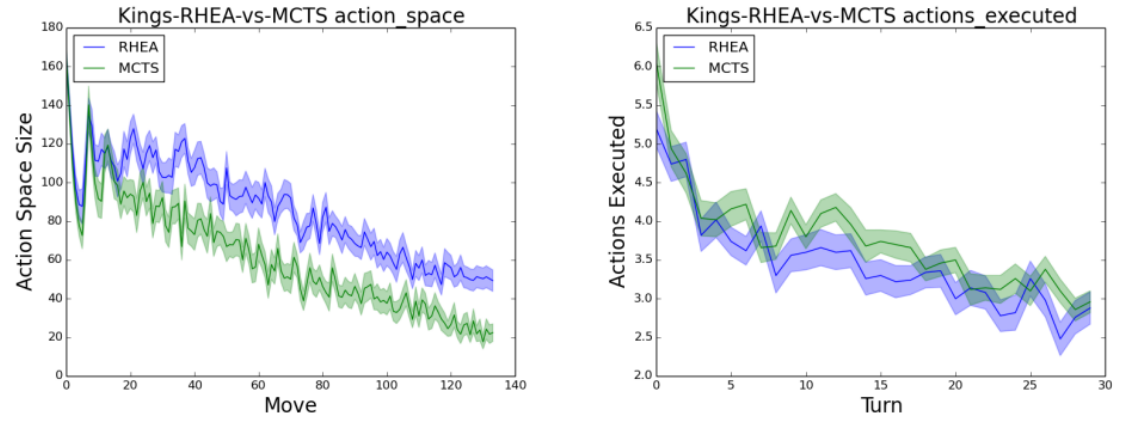


Рис. 1.9. Продуктивність MCTS та RHEA

Результати дослідження, хоча й чисто ілюстративні, але показують відомі проблеми, які виникають у традиційних агентів на основі пошуку, коли вони мають справу з високими факторами розгалуження в цих іграх.

Отже за проведеним дослідженням можна зробити висновок, що якість супротивників штучного інтелекту часто залишає бажати кращого. У цьому відношенні особливий інтерес становлять покрокові стратегії (TBS). Ці ігри зосереджені на прийнятті рішень високого рівня, а не на поведінкових діях низького рівня.

2 АНАЛІЗ МОДЕЛЕЙ ТА МЕТОДІВ ГЕНЕРАЦІЇ СЦЕНАРІЮ ПОВЕДІНКИ СУПРОТИВНИКА В ІГРАХ ЖАНРУ ПОКРОКОВОЇ СТРАТЕГІЇ

2.1 Аналіз та постановка проблеми існуючих методів для вирішення поставленої задачі

Повністю оглянувши існуючі методи для вирішення поставленої задачі ми можемо зробити висновки щодо їх доцільності проаналізувавши більш ретельно їх недоліки та переваги.

До недоліків методу Rule Based відноситься його обмеженість дій та передбачуваність. Діючи за чітким сценарієм такий метод стає досить легким і нудним для потенційного гравця, що швидко навчиться порядку дій, що написані в сценарії такого супротивника зі штучним інтелектом.

Перевагою ж методу можна вважати його надійність. Прописавши один чіткий сценарій розробнику набагато легше попередити помилки та баги. Відповідно до цього також можна віднести легкість створювання такого супротивника, адже на його розробку достатньо прописати чіткий алгоритм дій вздовж сценарію гри.

Аналізуючи наступний метод, One-Step Look Ahead, зрозуміло, що до переваг можна віднести прогнозованість дій на крок вперед з оцінкою найліпшого варіанту. Мінусом є те, що на відміну від Monte Carlo Tree Search прогноз з розрахунком відбувається лише на 1 крок. Не заглиблюючись в наступні шляхи вирішення.

Monte Carlo Tree Search – один з найкращих методів для створення штучного інтелекту супротивника в іграх жанру покрокової стратегії. Найбільшою перевагою є головна особливість методу: вміння розгортати розрахунок дій для пошуку найкращого по значенням результату. Тим самим алгоритм отримує статистично одні з найкращих результатів при тестах.

Через відсутність механізму запам’ятовування варіантів, які показали гірші показники, при великому обсязі варіантів дій та необхідності дальнішнього

розрахунку швидкість методу сильно зменшується на великих обсягах інформації.

Останній метод - Rolling Horizon Evolutionary Algorithms. Проаналізувавши принцип дії методу можна зробити висновок, що головною його перевагою є гарне прогнозування найкращих варіантів при малих горизонтах дій. До недоліків методу можна віднести підвищення неефективності методу на більших відстанях. Також через принцип дії алгоритму він може не помітити як в список виконуваних дій може потрапити дія яка є неможливою.

Підводячи підсумки проаналізованої інформації можна зробити наступну таблицю:

Таблиця 2.1

Недоліки та переваги існуючих методів

Метод	**Недоліки**	**Переваги**
Rule Based	• Обмежений в виконанні • Передбачуваність	• Надійність • Легкість створення в проєкті
One-Step Look Ahead	• Через обмеженість глибини кроків програє Monte Carlo Tree Search методу	• Оцінка якості наступного кроку
Monte Carlo Tree Search	• Зниження продуктивності при збільшені кількості варіантів дій • Відсутність запам’ятовування неправильних шляхів	• вміння розгортати розрахунок дій для пошуку найкращого по значенням результату
Rolling Horizon Evolutionary Algorithms	• Зниження ефективності при великих значеннях • Відсутність якісної перевірки на не правильні данні	• Прогнозування найкращих варіантів при малих горизонтах дій
Випадкові дії	• Відсутність стратегії	• Надійність • Легкість в створенні

Отже аналізуючи отриману інформацію можна зробити висновок, що критеріями якості створюваного методу можна вважати такі фактори: надійність у виконанні, оцінка якості наступних кроків та можливість отримання оптимального

шляху та стабільна продуктивність незалежно від кількості кроків. Проблеми існуючих методів включають обмеженість в виконанні, передбачуваність, зниження продуктивності при збільшенні кількості варіантів дій, відсутність стратегій та інші. Важливо уникнути цих проблем у новому продукті, зосередившись на розширенні можливостей виконання, запобіганні передбачуваності, покращенні продуктивності та врахуванні стратегій для більшої успішності.

Отже, для розробки нового продукту варто використовувати сильні сторони існуючих методів, такі як їхня надійність та легкість у використанні. Одночасно, слід активно працювати над усуненням недоліків, щоб новий продукт був більш гнучким, ефективним та стратегічним.

2.2 Виявлення можливих шляхів подальшого розвитку генерації сценарію поведінки супротивника в іграх жанру покрокової стратегії

По-перше, як вже зазначалося, одною з головних проблем штучного інтелекту вважають великі обчислювальні вимоги, що для глибокого навчання з підсиленням становлять значний виклик. Другим є необхідність створення системи для оцінки інформації, навченої в процесі навчання з підсиленням, щоб забезпечити, що модель може точно виводити оптимальні рішення, навіть в разі неповної інформації чи тої, що дає хибні твердження.

Таким чином ми підходимо до питання проблематики методів створення штучного інтелекту для суперників в іграх жанру покрокової стратегії. В роботі «Тактичний штучний інтелект з використанням нейронної мережі для стратегії в реальному часі»[9] було проаналізовано такі проблеми ігрового штучного інтелекту як:

- в простоті реагування та передбачуваності їх дій;
- використання попередньо визначеного алгоритма, але не дивлячись на це, показують хороші результати у боротьбі з гравцями-людьми, через високу швидкість дій та доступ до внутрішньої інформації гри.

Все це викликає негатив у користувачів ігрових продуктів, який можна

уникнути використовуючи більш просунуті технології та алгоритми для створення штучного інтелекту.

У 2019 році Justesen, N., Uth, L. M., Jakobsen, C., Moore, P. D., Togelius, J., & Risi, S. у своїй роботі «Blood bowl: A new board game challenge and competition for AI»[10] автори розробляють популярну настільну гру Blood Bowl, як новий виклик для штучного інтелекту. Blood Bowl є повністю доступною для спостереження, стохастичною, покроковою, сучасною настільною грою з сіткою ігрової дошки.

В своїй роботі автори протестували випадкового агента в 350 000 іграх проти самого себе, і йому жодного разу не вдалося набрати жодного очка. Це чудово і показує, що для результатів в створеній програмі потрібне належне планування. Ігри, в яких випадкові агенти ніколи (як майже ніколи) не набирають очок і не виграють, є надзвичайно складними для багатьох алгоритмів, таких як пошук дерева Монте-Карло та Q-навчання, оскільки вони покладаються на випадкове дослідження. Таким чином, рекомендації до досліджень в цьому напрямку автори підкреслюють саме такі як:

- Запровадження багато-агентного підходу до спільного планування, щоб ефективно обрізати величезні гілки ігрового дерева. Крім того, запровадивши підхід незалежного виявлення, можна розділити незалежні одиниці, які не впливають одна на одну, на різні групи з метою зменшення кількості одиниць у кожній групі.
- Використовувати в розробці метод пошуку дерева Монте-Карло для імітації глибоких вузлів. Алгоритми одно-раундового пошуку, запропоновані в цій статті, є повними алгоритмами і можуть бути використані для перевірки продуктивності нового алгоритму[11].

Аналізуючи ж вже розглянуті проблеми в існуючих методах можна виявити такі недоліки як:

- Відсутність стратегії;
- Відсутність якісної перевірки на не правильні данні;
- Відсутність запам’ятовування неправильних шляхів;
- Обмеженість в виконанні;

- Передбачуваність.

Відповідно, розробляєма методика повинна включати в собі усунення цих недоліків або ж зменшення їх впливу на результати виконання розробляємого штучного інтелекту.

Одною з ключових складнощів, з чим стикається штучний інтелект у покрокових стратегіях, є розробка ефективної стратегії прийняття рішень, що враховує не лише поточний стан гри, а й завдану мету досягнення. Це вимагає від алгоритмів вибору дій відчуття глибини аналізу та здатності адаптуватися до різноманітних сценаріїв гри.

Один із перспективних шляхів у вирішенні цієї проблеми полягає в використанні розумних алгоритмів прийняття рішень. Ці алгоритми мають здатність ураховувати комплексні фактори, пов'язані з поточним станом гри, такі як позиції гравців, наявні ресурси, структура поля тощо.

Крім того, важливою є розробка стратегій, що ґрунтуються на аналізі попередніх кращих ходів. Цей підхід передбачає вивчення та акумуляцію даних щодо оптимальних дій у минулих сценаріях гри для вибору наступних кращих кроків у майбутньому.

Такий підхід до розробки стратегій для штучного інтелекту в покрокових стратегіях сприяє покращенню його здатності досягати більш ефективних та обґрунтованих результатів у грі.

У покрокових стратегіях штучний інтелект, особливо супротивник, може стикатися з проблемою передбачуваності дій, що знижує його ефективність та унеможливлює досягнення бажаних результатів. Для вирішення цієї проблеми, можна використовувати підходи, спрямовані на збільшення рівня випадковості в прийнятті рішень та ускладнення передбачуваності стратегії супротивника.

Один із шляхів полягає у збільшенні степенів випадковості в прийнятті рішень штучним інтелектом. Це може включати в себе використання алгоритмів, які враховують стохастичні елементи та параметри, що змінюються випадковим чином. Це дозволяє уникнути формування певних патернів чи стабільних стратегій у супротивника, що ускладнює передбачуваність його дій для противника.

Крім цього, для підвищення складності передбачення дій супротивника можна використовувати алгоритми з параметрами, що змінюються динамічно відносно поточних умов гри. Це означає, що стратегія супротивника адаптується до змінюваних умов, ускладнюючи передбачуваність його наступних кроків.

Такий підхід дозволяє підвищити складність стратегії супротивника та знизити його передбачуваність, що може сприяти покращенню якості гри та результативності штучного інтелекту в покрокових стратегіях.

2.3 Формування математичної моделі створення супротивника в іграх жанру покрокової стратегії

Формулюючи задачу для створення математичної моделі супротивника в покроковій стратегії, ми можемо розглянути цей процес як задачу оптимізації агентом своїх дій у грі з урахуванням певних обмежень, цілей та правил.

Основні елементи математичної моделі для створення супротивника в покроковій стратегії:

- Стан гри;
- Список можливих дій;
- Функція цілі;
- Оптимізація дій.

Визначення стану гри на певному кроці, як сукупність параметрів, що описують поточний контекст гри, такі як розташування фігур, структура карти, розподіл агентів та інші важливі аспекти мають високий вплив на результативність обраної стратегії штучного інтелекту супротивника. А отже на початку ініціалізації дії супротивник повинен аналізувати ці фактори та повертати данні, на основі яких далі буде розрахована політика дій.

Перед тим як визначити дії, які варто виконати в конкретному стані гри, супротивник повинен аналізувати ці параметри і надавати відповідні дані, на основі яких буде розроблятися стратегія дій. Цей аналіз дозволяє визначити можливі варіанти дій, які є доцільними в поточному стані гри. У математичній моделі ці варіанти можуть бути представлені у вигляді змінних, які відображають доступні

дії для супротивника.

Отже відповідно до минулого пункту супротивник має набір допустимих дій, які може здійснити в поточному стані гри. Для математичної моделі вони будуть сформульовані у вигляді перемінних. В конкретній програмній реалізації обрані можливі дії супротивника матимуть такий вигляд та систему оцінювання.

Таблиця 2.2

Список можливих дій в практичній реалізації методу

Можливі дії	**Вартість (бали дії)**	**Бали цінності**
Дії атаки		
Вистріл	1	100
Граната	2	200
Атака мечем	3	300
Дії пересування		
Пересування	1	50
Знищення перепони	1	50

Відповідно, для покращення роботи агента необхідно сформувати формули, за якими буде обрано оптимальну дію. В даному етапі створення методу, можна вважати максимальну дію за балами цінності найкращим вибором зі списку.

Для досягнення максимального результату виконання поставленим пріоритетом, в розробляємому методі, повинен фігурувати такий математичний підхід аналізу цінності дії:

$$x=argmax_{x\in A(m)}\ (X_m+\sqrt{\frac{l}{k}}), \tag{2.1}$$

де:

"x" - обрана дія;

X - значення вибору дії "x" із списку доступних

m - індекс обраної дії зі списку;

l - кількість разів, коли список m був відвіданий;

k - кількість разів, коли список m був відвіданий, і було обрано дію "x";

"A(m)" - множина дій.

Ця формула розрахунку цінності дії представляє алгоритм, який використовується для прийняття рішень при аналізі вибору дії. Основна ідея полягає у виборі дії "x", яка максимізує певну функцію. Ось кілька ключових елементів формули:

Argmax - це операція, яка повертає значення аргументу "x", що максимізує вираз на який вона застосована.

Xm - це значення, пов'язане з обраним вибором дії "x" для індексу "m".

$\sqrt{(l/}$"k") - це складова формули, яка враховує важливість обраної дії "x" в залежності від кількості разів, коли список "m" був відвіданий та коли саме було обрано дію "x".

l - це кількість разів, коли список "m" був відвіданий.

k - це кількість разів, коли список "m" був відвіданий, і було обрано дію "x".

A(m) - це множина дій, які можна виконати відносно обраного списку "m".

В цілому, ця формула визначає процес вибору дії і вибирає ту, яка максимізує відповідний вираз або функцію. Вона може застосовуватися у багатьох областях, де потрібне прийняття оптимальних рішень на основі вхідних даних та їх аналізу.

Сценарій поведінки супротивника в іграх розроблено для створення системи прийняття рішень, яка спрямована на взаємодію супротивника з гравцем. Цей сценарій дозволяє супротивнику реагувати на дії гравця та приймати відповідні рішення залежно від обставин у грі.

При цьому розробляємий сценарій дії супротивника матиме такий вигляд:

Таблиця 2.3

Сценарій поведінки супротивника

Етапи сценарію	**Дії**
Обробка стану	Поки хід гравця – стан очікування. Коли наступає хід суперника – дія.
Перевірка	Перевірка оточення для виявлення ворогів та перешкод.
Рішення	Якщо ворог у видимому полі зору і можна вдатися до атаки, то відбувається атака зі списку доступних атак.
Етапи сценарію	**Дії**
Рішення	Якщо є перешкоди на шляху до ворога, обійти або ж знищити перешкоду. Якщо ворог у видимому полі зору, але недосяжний, пересування в його напрямку, використовуючи стратегію навігації.
Навігація	Використання алгоритмів навігації для руху в напрямку ворога чи обходу перешкод.
Контроль	Контроль вищезазначених правил .

Описані етапи в сценарії обрано на основі класичних стратегій в покрокових стратегічних іграх та механік, що забезпечують оптимальне реагування супротивника на дії гравця. Етапи спроектовані таким чином, щоб супротивник зміг ефективно реагувати на зміни в грі та приймати відповідні рішення, враховуючи обмеження та мету досягнення.

Метою кожного ходу супротивника є накопичення найбільшої кількості балів дії для цього сформулюємо формулу максимізації стратегічного планування. Яка буде залежати напряму від вищеописаної формули найбільшої цінності дії. Також

для пошуку максимуму накопичення балів буде фігурувати обмеження. Розмір використаних балів не повинен перевищувати доступні. Отже формула набуває такого вигляду:

$$P = \sum x_i, \tag{2.2}$$

$$\sum c_i \leq i, \tag{2.3}$$

де:

x_i – обрана дія;

c_i – вартість виконаної дії;

i – кількість балів дії.

Дана формула представляє собою модель оцінки оптимальності вибору дії супротивника в ігровій ситуації. У цій моделі вартість обраної дії обчислюється як сума вартостей всіх виконаних дій.

Пошук x_i, що максимізує цю вартість, дозволяє супротивнику вибрати оптимальну дію – таку, яка дасть найбільший результат в контексті даної гри чи ситуації.

Отже, формула допомагає супротивнику аналізувати та вибирати дії, що максимізують його результативність в ігровому середовищі, де вартість кожної дії залежить від набутих балів та відповідних коефіцієнтів.

Також важливим аспектом є те, що оскільки атака має вищу пріоритетність, бо головна ціль – перемогти гравця, то обхід перешкод та пересування до цілі повинно відбуватися з використанням мінімальної витрати балів дії. Для того, щоб максимізувати їх витрату на атаку.

Пошук мінімального шляху для супротивника в грі покрокової стратегії можна виразити за допомогою алгоритму пошуку найкоротшого шляху, наприклад, алгоритму A*. Даний алгоритм шукає найкоротший шлях в графі між двома вузлами. У випадку поля гри у вигляді квадратів переміщення, де кожне поле може бути розглянуте як вузол графа, а рухи між цими полями як ребра графа, використання алгоритму A* може бути ефективним рішенням.

Формула A* шукає найкоротший шлях шляхом врахування вартості кожного кроку і та поступового розгортання пошуку в напрямку, що найбільше відповідає до мети або цілі. Загальний вигляд такої формули:

$$f(n) = g(n) + h(n), \tag{2.4}$$

де:

g(n): вартість найкоротшого шляху від стартового вузла до поточного вузла n.

h(n): евристична функція, яка оцінює вартість найкоротшого шляху від поточного вузла до цільового вузла.

f(n): сумарна вартість шляху через поточний вузол n.

Формула A* шукає найоптимальніший шлях шляхом оцінки f(n) для кожного вузла і вибору вузла з найменшою сумарною вартістю f(n) для подальшого розгортання. Цей процес триває досягнення цільового вузла або досягнення максимальної глибини пошуку.

Якщо перед супротивником існує перешкода, він повинен проаналізувати чи є можливість її зруйнувати та розрахувати, що є вигіднішим: зруйнувати перешкоду чи обійти(зрозуміло, що якщо перешкоду неможна зруйнувати - її потрібно обійти) . За умови, що вигіднішою вважається та дія, яка використає менше балів дії ми маємо таку умову.

$$\text{Мінімальна дія} = \begin{cases} \text{Знищення перешкоди, якщо } P_{destroy} * C_{destroy} < P_{bypass} * C_{bypass} \\ \text{Обхід перешкоди , якщо } P_{destroy} * C_{destroy} > P_{bypass} * C_{bypass} \\ \text{Будь яка з обраних дій, якщо } P_{destroy} * C_{destroy} = P_{bypass} * C_{bypass} \end{cases}, \tag{2.5}$$

де:

$P_{destroy}$ - ймовірність успішного знищення перешкоди;

P_{bypass} - ймовірність успішного обходу перешкоди;

C_{bypass} -вартість успішного знищення перешкоди;

$C_{destroy}$ - вартість дії знищення перешкоди.

Відповідно до цього ймовірність успішного знищення перешкоди може набувати значення 0 або 1 відносно того чи можна знищити перешкоду.

Розрахунок ймовірності успішного обходу перешкоди P_{bypass} можна виразити як відношення кількості успішних обходів до загальної кількості спроб обійти перешкоду:

$$P_{bypass} = \frac{\text{Кількість успішних спроб обійти перешкоду}}{\text{Загальна кількість спроб обійти перешкоду}}, \quad (2.6)$$

Ця формула представляє відношення кількості разів, коли супротивник успішно обійшов перешкоду, до загальної кількості спроб обходу перешкоди протягом певного часу або ітерацій в процесі гри. Це є одним із ключових аспектів розглядуваної математичної моделі - урахування невизначеності ймовірності успішного обходу на початку гри. Це важливий момент, оскільки відсутність попередніх спроб обходу перешкоди у випадку початку гри призводить до визначення ймовірності обходу як нульової. Визначення такого стартового показника уведене для раціонального підходу до розрахунків успішності обходу перешкоди впродовж гри.

Якщо гра тільки почалася і відсутні дані щодо попередніх спроб обходу перешкоди, можна вважати, що ймовірність успішного обходу P_{bypass} на початку гри рівна нулю. Тобто, поки не було спроб обійти перешкоду, ймовірність успіху цієї дії є невідомою і може бути розрахованою рівною нулю.

Відповідно до розробленої математичної моделі сформуємо критерій оцінювання розробляємого методу.

Для оцінки ефективності роботи методу необхідно проаналізувати кількість перемог відносно інших лідуючих методів. Як було зазначено вище, таким вважається метод RHEA. Також окрім перемог проведемо оцінку того, наскільки ефективним є розрахунок дій обраного методу. Для цього буде необхідно порівняти середню кількість набраних балів за гру з кількістю раундів які було

проведено в грі, так як покрокова стратегія є стохастичною грою – це є важливим критерієм.

Набрані бали за гру будуть розраховуватись як сума P за всі проведені ігри які будуть позначатись індексом j.

Отже формула ефективності штучного інтелекту супротивника набуває такого вигляду:

$$Efect = \frac{\bar{S}}{\bar{n}}, \tag{2.7}$$

$$\bar{S} = \frac{\sum S_j}{j}, \; \bar{n} = \frac{\sum N_j}{j}, \tag{2.8}$$

$$S = \sum P_j, \tag{2.9}$$

де :

$\bar{S}$- Середня кількість набраних балів цінності за проведені ігри;

$\bar{n}$- середнє значення кількості ходів за проведені переможні ігри;

n- кількість ходів за гру;

P- стратегічне планування;

j – кількість проведених ігор.

Формула враховує ефективність штучного інтелекту супротивника через середню кількість балів, які він набрав, та середню кількість ходів, зроблених у переможних іграх.

Отже, за допомогою цієї формули можна порівняти різні методи штучного інтелекту та оцінити їх ефективність у грі покрокової стратегії. Вона дозволяє враховувати результативність розробленого методу в термінах набраних балів та кількості ходів, що є важливим для оцінки продуктивності алгоритмів у таких типах ігор.

3 РОЗРОБКА МОДЕЛІ ГЕНЕРАЦІЇ СЦЕНАРІЮ ПОВЕДІНКИ СУПРОТИВНИКА В ІГРАХ ЖАНРУ ПОКРОКОВОЇ СТРАТЕГІЇ

3.1 Опис використаних програмних засобів

Для створення штучного інтелекту супротивника в грі жанру покрокової стратегії є доцільним використовувати програмний рушій Unity. Середовище розробки комп'ютерних ігор, розроблене американською компанією Unity Technologies. Unity дозволяє створювати додатки, що працюють на більш ніж 25 різних платформах, що включають персональні комп'ютери, ігрові консолі, мобільні пристрої, інтернет-програми та інші. Випуск Unity відбувся у 2005 році і з того часу триває постійний розвиток. До переваг обраного рушія можна віднести: кросплатформенність, зрозумілий та зручний інтерфейс, двигун об’єднує інструменти для візуального компонування гри та програмування, велика кількість потужних допоміжних засобів, пов’язаних із проектуванням та дизайном, сучасний рівень тривимірної графіки та спецефектів та безкоштовність.

Також Unity підтримує вбудований браузер Asset Store, що допоможе облегшити процеси пошуку графічних рішень для створення загального середовища для тестування методу.

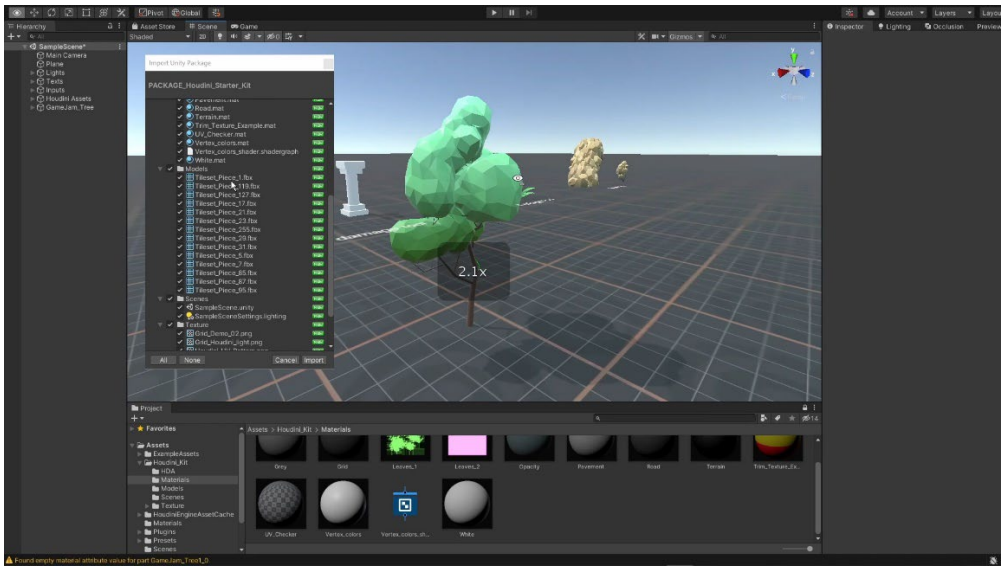


Рис. 3.1. Ігровий рушій Unity

Особливість Unity полягає у його розширених можливостях для розробки ігор у тривимірному просторі. Це середовище надає доступ до великого арсеналу інструментів для створення реалістичних графічних об'єктів, анімації, фізики та штучного інтелекту. І саме в останній складовій - штучному інтелекті, Unity пропонує зручні інструменти для розробки імплементації алгоритмів, а також для моделювання поведінки супротивників у покрокових стратегічних іграх. Ці можливості допоможуть вам з легкістю створити та реалізувати штучний інтелект, необхідний для вашого проекту.

Використання ресурсів з Unity Asset Store може значно полегшити створення штучного інтелекту супротивника в покроковій стратегії. Завантаження вже готових до використання персонажів, арт-об'єктів оточення та звукових ефектів дозволить швидше створювати живе та реалістичне ігрове середовище. Це стане перевагою, бо для поставленої задачі немає необхідності самостійно створювати необхідні модельки ігрових об’єктів, а отже економія часу на розробці дозволить більше сконцентруватися в створенні розробляємого методу штучного інтелекту.

Оскільки рушієм обрано Unity, обрана мова програмування відповідно C#.

C# — об’єктно-орієнтована, компонентно-орієнтована мова програмування. C# надає мовні конструкції для безпосередньої підтримки цих концепцій, що робить C# природною мовою для створення та використання програмних компонентів.

Розробка власного штучного інтелекту для ігор у покроковому жанрі вимагає потужних і ефективних інструментів програмування. Використання мови програмування C# у поєднанні з ігровим рушієм Unity надає багато переваг для створення інтелектуальних систем в ігровому середовищі.

C# є однією з найпопулярніших мов програмування для розробки на платформі .NET. Ця мова надзвичайно потужна та ефективна, дозволяючи розробникам писати чіткий, точний та легко збережуваний код. Її використання в ігровій розробці спільно з Unity дозволяє швидко створювати складні системи

ШІ, використовуючи багатофункціональні інструменти та платформи для розробки.

Unity надає широкий вибір інструментів для розробників на будь-якій платформі. Від розширеного редактора коду Visual Studio Code до потужного інтегрованого середовища розробки Visual Studio, вам доступні всі необхідні інструменти для створення штучного інтелекту в сприятливому середовищі. Можливість використання улюблених бібліотек та перенесення навичок і коду між різними платформами забезпечує швидкість та ефективність розробки штучного ітелекту в Unity.

Відповідно для створення програмного коду та розробки методу знадобиться середа розробки. Інтегроване середовище розробки Visual Studio можна використовувати для редагування, налагодження та складання коду, а також для публікації програми. На додаток до стандартного редактора та відладчика, що надаються більшістю інтегрованих середовищ розробки, Visual Studio включає компілятори, засоби завершення коду, графічні конструктори та багато інших функцій для покращення процесу розробки програмного забезпечення.

Одним із найпопулярніших розширень Visual Studio для роботи з розробкою ігор є «Game development with Unity». Це розширення дозволяє зручно працювати з кодом Unity в середовищі Visual Studio. Завдяки цьому розширенню, ви можете користуватися усіма перевагами Visual Studio, такими як підказки, відлагодження коду, підтримка системи керування версіями (VCS) та багато іншого, працюючи над проектами Unity. VS Tools for Unity забезпечує взаємодію між Visual Studio та Unity, що спрощує розробку ігор та дозволяє швидко та ефективно працювати з кодом.

Розширення спрощують процес розробки ігор в Unity, роблять його більш зручним та продуктивним, а також допомагають уникнути помилок та покращити створюваного коду в розробці проєкту.

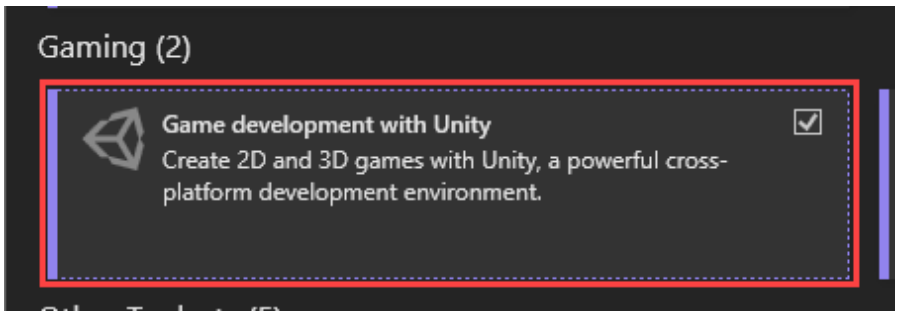


Рис. 3.2. розширення Game development with Unity

3.2 Опис структури проекту

Відповідно для створення алгоритму дій для супротивника в ігровому середовищі, було проведено аналіз сценарію гри та визначено ключові аспекти стратегії поведінки. Ця стратегія базується на оцінці вартості кожної можливої дії в грі та їх ефективності. Під час моделювання оптимальних дій було розроблено формулу, яка оцінює найбільш перевагу дії за умови максимізації вартості з врахуванням обмежень і критеріїв.

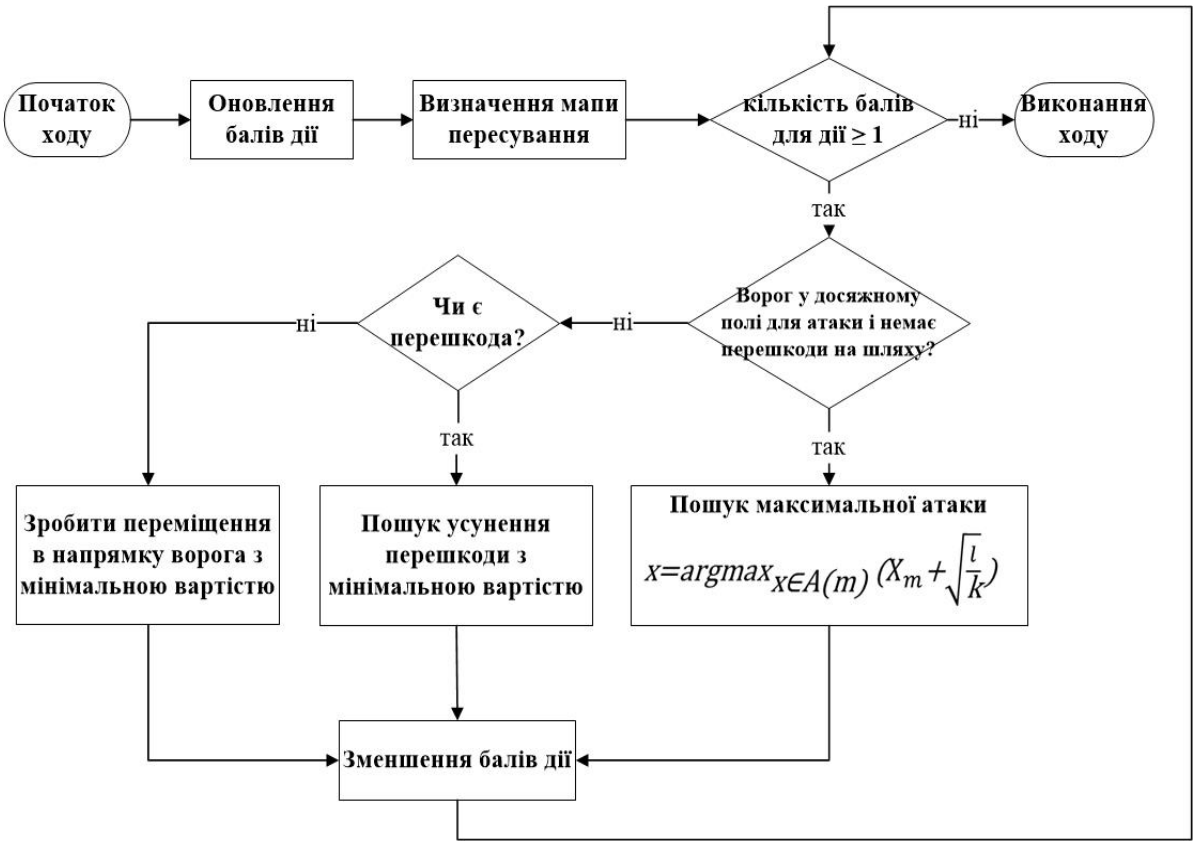


Рис. 2.3. Алгоритм дії розробленого методу для супротивника

Отриманий алгоритм дій дозволяє супротивнику в грі вибирати оптимальні ходи, спираючись на критерії ефективності та вартості дій, що дозволяє йому приймати більш обдумані та вигідні рішення в ігровому процесі.

Наступним кроком розробки програмного продукту для реалізації створюваного методу є формування діаграми станів штучного інтелекту супротивника. Діаграма станів ШІ є ключовим елементом структури програмного продукту у грі жанру покрокової стратегії. Вона ілюструє різні можливі стани, у які може переходити інтелект супротивника під час гри. Ця діаграма слугує як модель поведінки, яка допомагає програмі керувати діями супротивників, робить їхню стратегію більш динамічною та реагуючою на зміни в грі.

Стани ШІ включають в себе "Очікування ходу", "Виконання ходу" та "Зайнятий" або ж "Виконання дії" . Ці стани показують, як супротивник реагує на зміни в грі: чекаючи своєї черги для виконання дій, виконуючи свій хід та завершуючи дії, а також перехідний стан після виконання ходу. Основні стани, такі як "Очікування ходу", "Виконання ходу" та "Зайнятий" або "Виконання дії", ілюструють різні фази стратегії супротивника в грі. "Очікування ходу" показує, що супротивник готовий приймати рішення та чекає своєї черги для виконання дій. "Виконання ходу" відображає активність інтелекту супротивника під час власного ходу або дії. "Зайнятий" або "Виконання дії" показує, що супротивник завершує виконання ходу та готується до наступних дій.

Рис. 3.4. Діаграма станів штучного інтелекту

Ця діаграма важлива, оскільки вона створює модель поведінки супротивника, допомагаючи програмі керувати ШІ у грі. Вона робить стратегію інтелекту більш динамічною та реагуючою на зміни у грі, дозволяючи програмі пристосовувати стратегії супротивника в реальному часі. Це дозволяє створювати більш реалістичних та цікавих штучних супротивників для гравців у покрокових стратегіях.

3.3 Опис інтерфейсу

Розробка поведінки супротивника в грі жанру покрокової стратегії вимагає реалізації повноцінного ігрового поля де можна протестувати продуктивність та реалізацію самого методу. Також важливим елементом розробки штучного інтелекту супротивника є коректне відображення його поведінки та логіки, для правильного оцінювання запропонованого методу. Відповідно до цього було створено ігрове поле де було розміщено всі необхідні елементи для тестування розробленого штучного інтелекту супротивника.

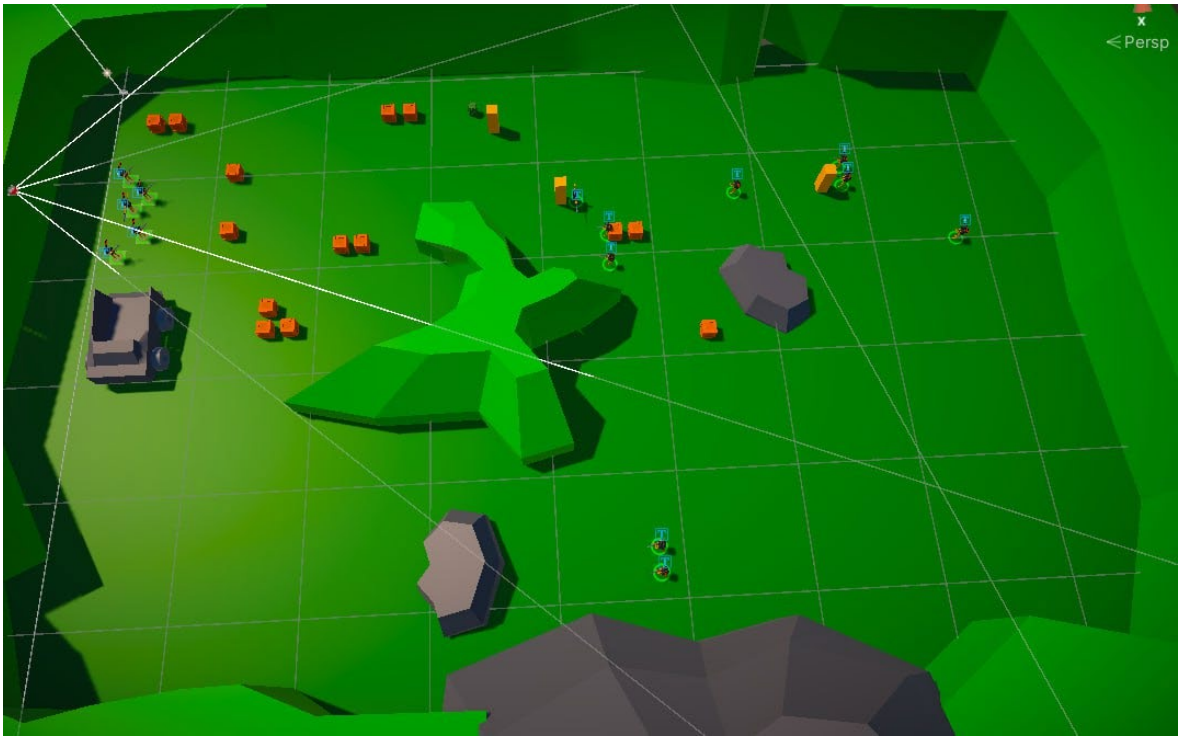


Рис. 3.5 Створена мапа рівня

Також було розроблено ігрову панель для гравця де було сформовано всі

можливі дії, як є доступними і реалізованими в тому числі для штучного супротивника.

До необхідних елементів ui/ux частини гри відносяться:

- Відображення номеру раунду;
- Ілюстрація доступних дій;
- Візуалізація ігрового поля;
- Кнопка завершення ходу;
- Відображення наявних балів дій;
- Відображення кількості здоров’я ігрової одиниці.

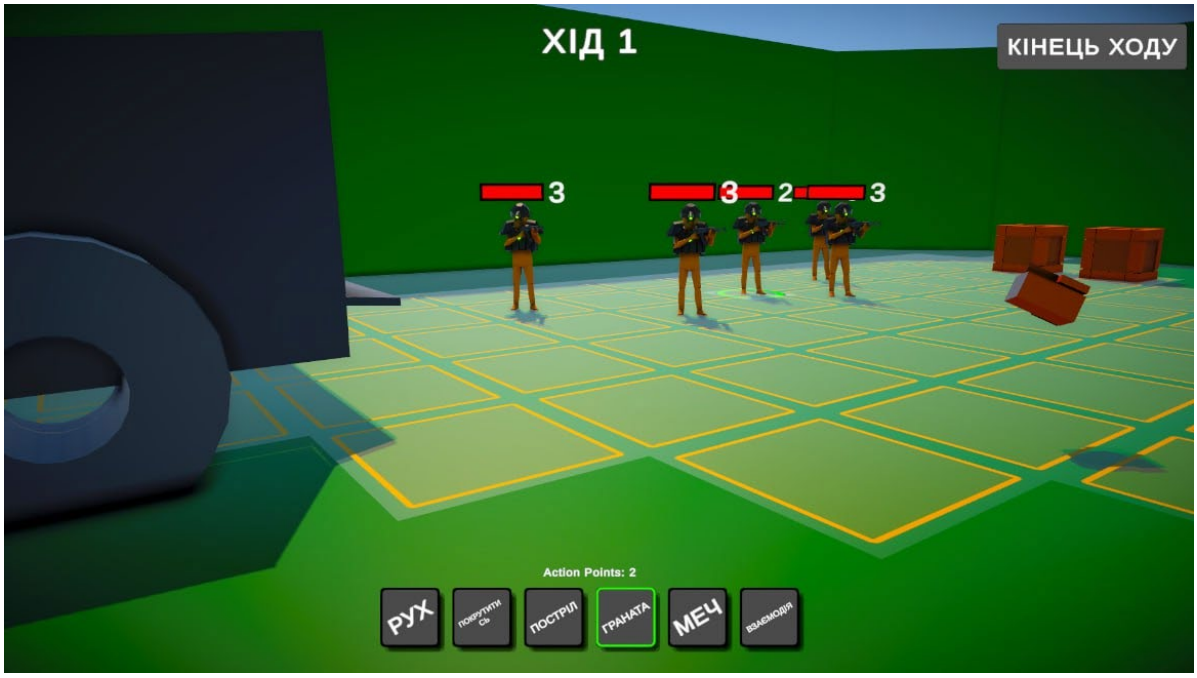


Рис. 3.6. Інтерфейс гри

Інтерфейс розроблений для гравця гри не є важливим елементом в оцінці ефективності розробляємого штучного інтелекту, але все ж таки є необхідним для розуміння логіки відтворення дій з боку гравця, бо як вже зазначалося в цій роботі – розуміння поведінки гравця є важливим аспектом створення штучного інтелекту супротивника в грі.

Також важливими елементами для правильного тестування та корегування розробленого методу було створення необхідних ігрових моделей стратегічних одиниць сили(надалі юнітів), перешкод, що мають можливість бути винищеними

та мати лише функцію перешкоди, для тестування здібності штучного інтелекту супротивника оцінювати шлях до заданого ворога, правильно розраховане ігрове поле для дій юнітів.

Одною з ключових частин розробленого продукту є саме система оцінки дій та її візуальне вирішення. Для повноцінного аналізу роботи штучного інтелекту супротивнуку необхідно було мати змогу присвоїти кожній дії значення і її вартість. Отже пересування, стрільба, ближня атака, знищення перешкоди і в цілому будь-яка дія гравця, та штучного інтелекту, матиме свою вартість. А кожен юніт в свою чергу матиме бали які зможе витрачати на відповідні дії.

Відображення відповідної кількості наявних балів буде відбуватися над головою кожного юніту поруч з плашкою здоров’я, символізуючи наявно кількість очок доступних для використання конкретною ігровою одиницею.

Система наявності балів є важливим елементом, завдяки їй надалі буде відбуватись оцінка продуктивності розробленого методу в порівнянні з виконанням поставленої задачі - перемогти гравця.

Рис. 3.7. Відображення балів дій

Ігрове поле у грі є фундаментальним елементом, що визначає можливості руху та взаємодії гравців або штучного інтелекту. Кожна клітина поля представляє собою потенційну позицію для пересування або взаємодії. Для аналізу штучним інтелектом його можливості дії ігрове поле розмежоване відповідними клітинами ходу, де вартість пересування на максимально можливу відстань оцінюється в один ігровий бал. А максимальна відстань переміщення дорівнює чотирьом клітинкам радіусу від ігрової одиниці.

Також до можливих клітинок пересування не входять ті елементи оточення, які є перешкодою і не мають змоги бути усуненими. Відповідно, гравець, або ж штучний інтелект супротивника, повинен оцінювати не лише вартість атаки, але й переміщення на зручні для цього позиції.

Розроблене ігрове поле і розподілення його на клітини пересування є важливим елементом для розрахунку дій та їх ефективності з боку розробляємого штучного інтелекту супротивника.

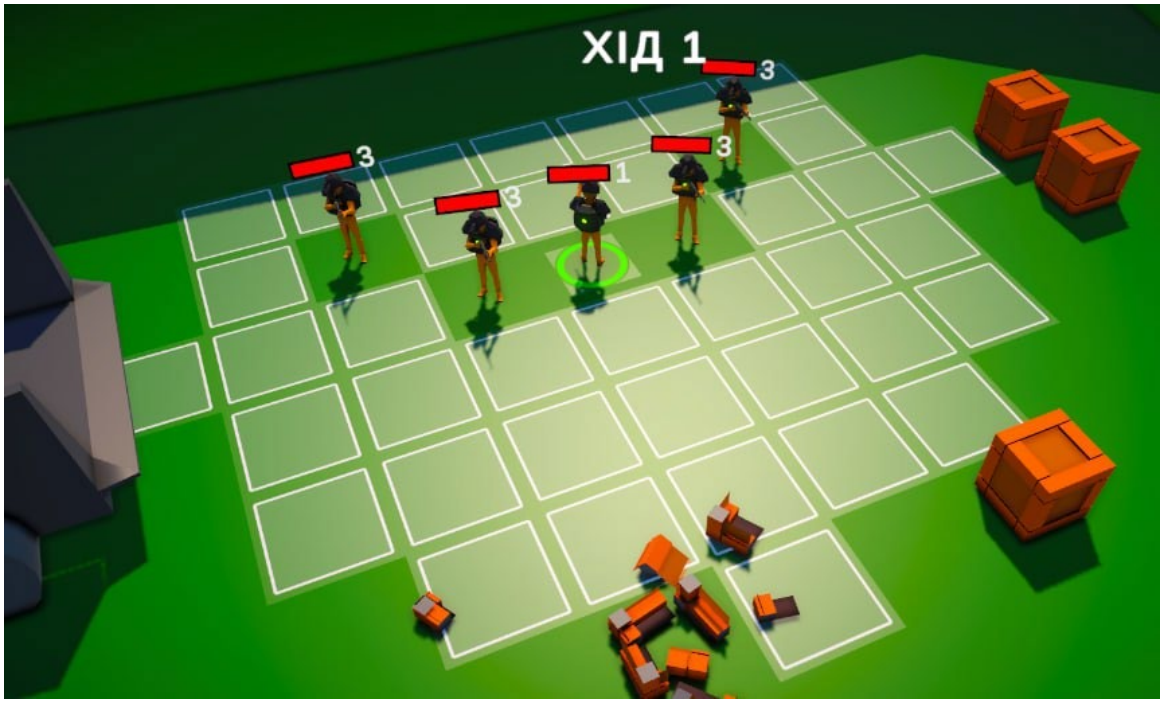


Рис. 3.8. Поле ходу

Створення ігрового поля та розподіл його на клітини для пересування є важливим етапом у розвитку штучного інтелекту супротивника. Цей процес потребує аналізу, розрахунків та стратегічного планування, щоб ефективно

визначати оптимальні рухи і дії для досягнення поставленої мети в грі.

Відповідно пересування на полі має свою вартість та стратегічну цінність. Виконуючи пересування юніт витрачає бали дії. Це також реалізовано не тільки для результативного вираховування роботи штучного інтелекту, але й для того, щоб гравець міг аналізувати доступні дії в грі.

Застосування витрат балів дії впливає на загальну стратегію гравця, змушуючи його уважно аналізувати кожен крок та приймати обдумані рішення. Це стимулює гравця до більш глибокого розуміння гри та планування стратегій, сприяючи розвитку стратегічного мислення.

Однією з ключових переваг цієї механіки є її здатність допомогти гравцеві не тільки ефективно управляти своїми діями, а й створювати більш обдумані та виважені стратегії в грі. Крім того, вона стимулює активну участь гравця у формуванні власних тактик, роблячи гру більш цікавою та захопливою.

На представленому зображенні(Рис 3.3.5.) видно, як юніт витрачає бали дії внаслідок вибору опції переміщення по клітинках ігрового поля.

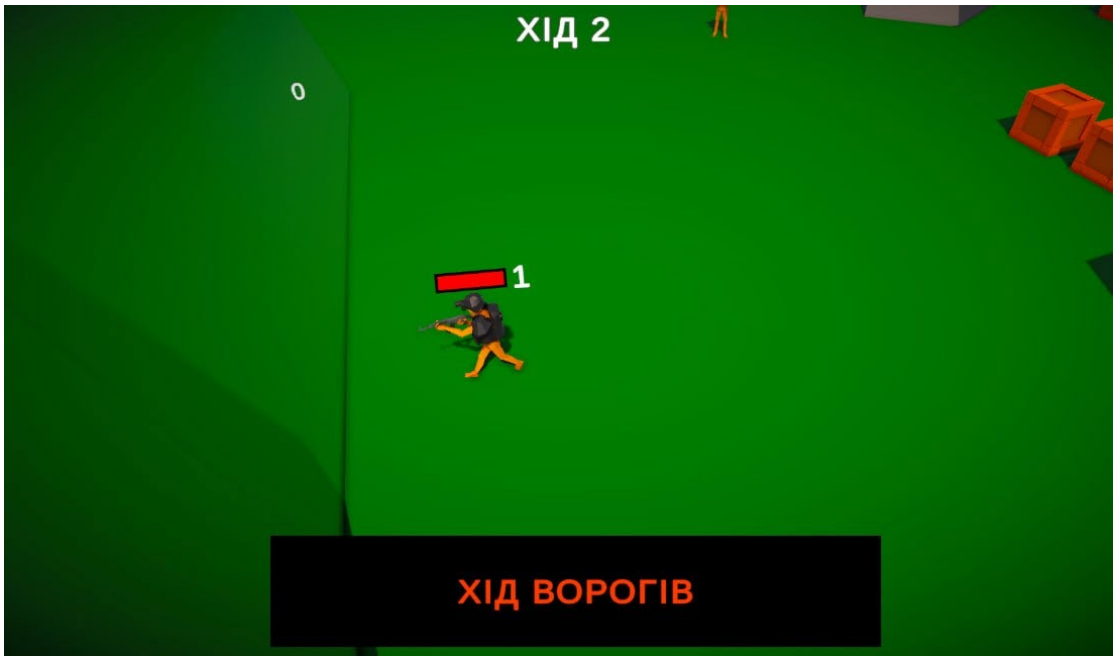


Рис. 3.9. Зміна кількості балів

Оскільки покрокова стратегія в своїй основі має принципи стохастичної гри її важливою частиною є зміна ходу з почерговим замороженням дій супротивника.

Це є ключовою відмінністю від стратегій реального часу.

Зміна ходу з подальшим замороженням дій супротивника є однією з ключових тактичних складових покрокової стратегії. Це означає, що гравець чи штучний інтелект у грі мають змогу переосмислювати свої дії на кожному кроці гри, надаючи можливість адаптуватися до змін у ігровому середовищі та реакції супротивника. Основні умови для завершення ходу включають вичерпання балів дії, відсутність можливості виконання дій або досягнення перемоги.

Відповідно до цього для створюваного додатку було розроблено та реалізовано зміну ходу. Умовами завершення ходу є:

- Відсутність балів дій;
- Відсутність можливості виконання дій;
- Перемога в грі.

Також для статистичного вираховування успішності створеного методу було додано розрахунок кількості ходів, що були реалізовані. Відповідно задача штучного інтелекту супротивника виконати не тільки найбільше перемог за своїх конкурентів, але й за найменшу кількість ходів.

З оновленням ходу також відновлюється наявна кількість балів дії для кожного живого ігрового юніту.

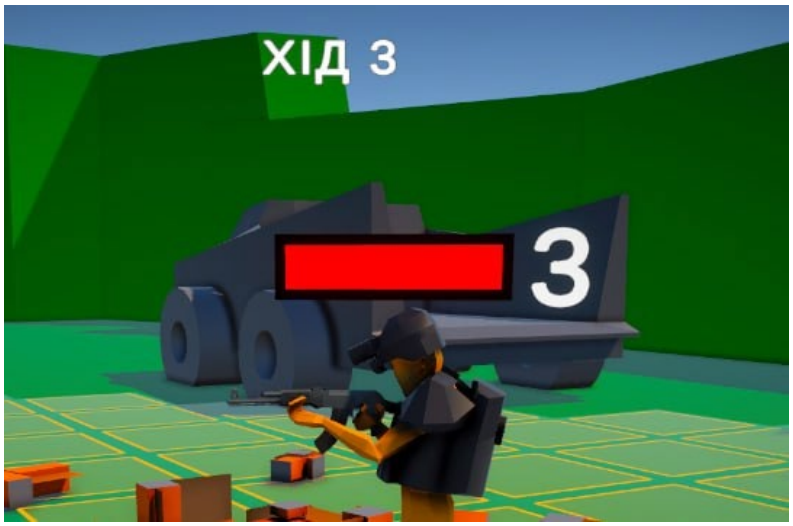


Рис. 3.10. Оновлення балів дії з оновленим ходом

Створення перешкод на шляху існуючих юнітів також ускладнює стратегію гри, змушуючи штучний інтелект аналізувати витрату наявних балів з урахуванням

не можливості одразу нанести шкоди своєму супротивнику, тобто гравцеві.

Також наявність перешкод можна використовувати для того, щоб блокувати дії супротивника чи використовувати її в якості захисту в наступному кроці. Штучний інтелект, для покращення ігрового процесу не повинен бути занадто передбачуваним, тому повинен вміти в своєму розрахунку вигідних кроків використовувати і так звані хитрощі, що в своїй суті є більш глибшими стратегічними діями.

Для реалізації цього елементу ігрового оточення було вирішено використати перешкоди двох типів, а саме:

- Перешкоди, які можна тільки обійти;
- Перешкоди, які можна зруйнувати використовуючи бали руху.

Отже, для штучного інтелекту важливо мати можливість раціонально оцінювати та аналізувати ці перешкоди, враховуючи їх вплив на загальну стратегію гри. Такі елементи створюють складність та виклик для інтелекту, стимулюючи його розвиток та вдосконалення стратегій для досягнення бажаного результату у грі.

Перешкоди, які не можна знищити створюють простір для більш складних стратегій та рішень, оскільки вони не можуть бути просто прибрати шляхом атаки чи руйнування, як це може бути з іншими видами перешкод.

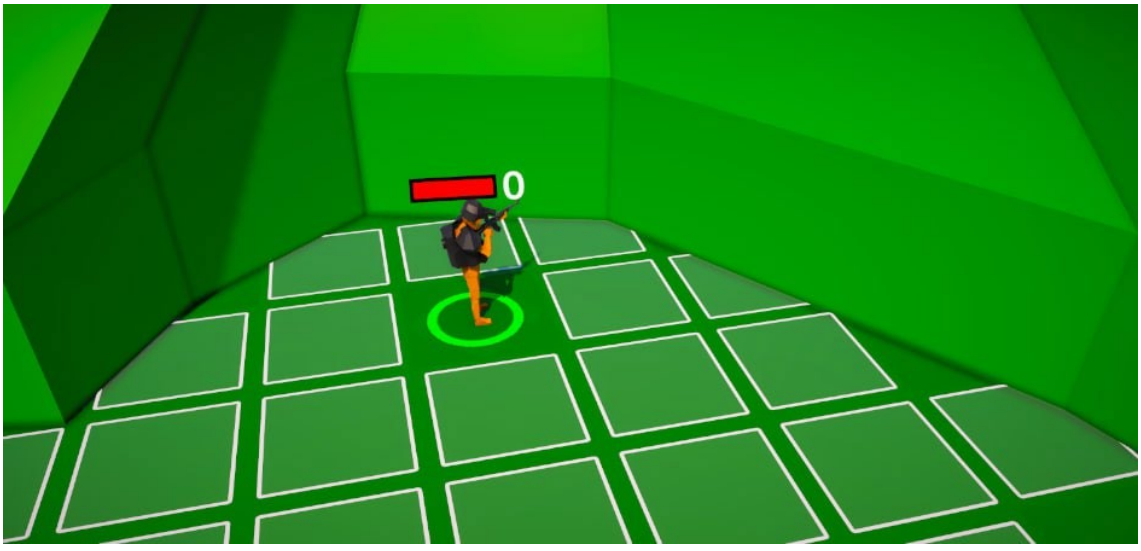


Рис. 3.11. Візуалізація перешкод

Відповідно до реалізованих не руйнівних перешкод необхідно створити

перешкоди, які гравець, або ж штучний інтелект, матиме змогу зруйнувати задля розширення тактичного вибору дій, або використання таких блоків як елементу захисту від нападника.

Перешкоди, які можна зруйнувати в покроковій стратегічній грі, представляють значну стратегічну цінність, оскільки вони не лише впливають на хід бою, але й надають можливості глибшого планування та управління ресурсами.

Ці перешкоди створюють динамічні ситуації, де гравці мають можливість впливати на гру, вибираючи, які перешкоди зруйнувати в першу чергу. Це вимагає від гравців аналізувати поточну ситуацію, прогнозувати можливі наслідки та оцінювати відносну важливість кожної перешкоди для досягнення своєї стратегічної мети.

Стратегічна цінність полягає в тому, що зруйнування перешкод може вплинути на кінцевий результат гри. Гравці можуть використовувати цю можливість для створення переваги в битві, змушуючи супротивника витрачати ресурси на реконструкцію чи зміну стратегії. Також, руйнування перешкод може відкривати нові шляхи атаки або змінювати ландшафт бою, що робить гру більш динамічною та цікавою.

Окрім того, ця можливість відкриває широкий спектр стратегічних варіантів. Гравець може обирати, які перешкоди зруйнувати, залежно від своїх стратегічних цілей, ресурсів та поточної ситуації на полі бою. Це вимагає від нього уважного розгляду кожної можливості та вміння приймати обґрунтовані рішення.

Витрати ігрових балів на зруйнування перешкод є ключовою складовою стратегії гри, що впливає на вирішення гравця чи штучного інтелекту супротивника. Ця дія вимагає обережного розгляду наявних ресурсів і відповідну стратегічну увагу. Витрати ресурсів на зруйнування перешкод можуть обмежити можливості агента в майбутньому, ускладнюючи процес тестування і поліпшення методу штучного інтелекту.

Ця тактика вимагає не лише вирішення, чи варто зруйнувати перешкоду, але й урахування можливих варіантів використання цих ресурсів для інших стратегічних дій, таких як атака або оборона. Гравець чи штучний інтелект мають

приймати рішення на основі поточної ситуації в грі, оцінюючи, як ця дія вплине на загальну стратегію.

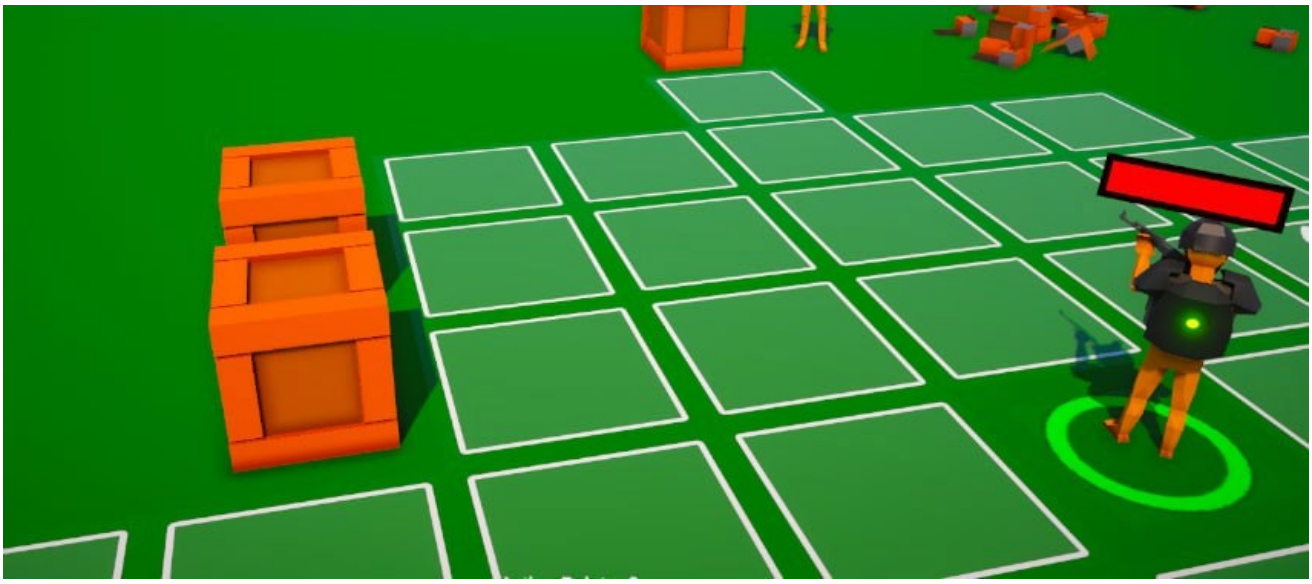


Рис. 3.12 Перешкоди, що можна зруйнувати

Також цілком зрозуміло, що для дії в зруйнування перешкоди також витрачатимуться ігрові бали, що буде перешкоджати та ускладнювати стратегію даної гри для тестування створеного методу штучного інтелекту супротивника.

Витрати балів на зруйнування перешкод виступають як стратегічне вирішення гравця або штучного інтелекту супротивника. Вони повинні враховувати, чи є така дія наразі вигідною в контексті поточної ситуації в грі. Це вимагає від гравця або інтелектуального агента вирішувати, чи варто витрачати ці ресурси на зруйнування перешкоди чи краще зберегти їх для інших дій, таких як атака чи оборона.

Важливо пам'ятати, що витрати ігрових балів на зруйнування перешкоди можуть вплинути на загальну стратегію гри, обмежуючи можливості агента. Це робить процес тестування та вдосконалення методу штучного інтелекту складнішим, оскільки необхідно збалансувати витрати ресурсів на зруйнування перешкод та інші стратегічні дії для досягнення успіху в грі.

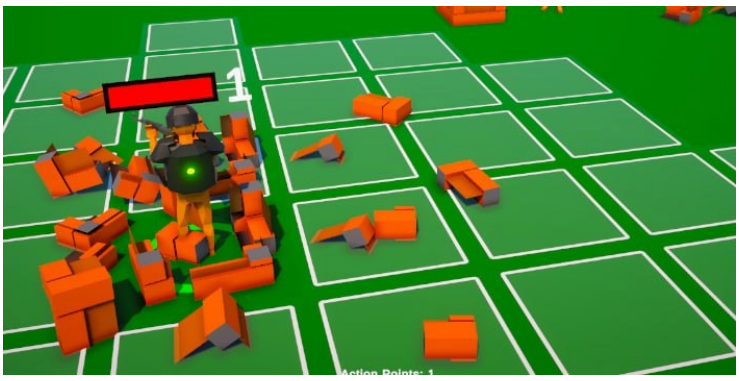


Рис. 3.13. Руйнування перешкоди

Окремою важливою частиною розробки стохастичної гри є обмеження дій супротивника, під час дії ходу іншого гравця. Під час виконання ходу штучним інтелектом панель гравця блокується і діє надпис «ХІД ВОРОГІВ».

Такий підхід до блокування дій під час ходу противника дозволяє створювати напруженість та стратегічні обмеження, сприяючи активному аналізу ситуації в грі. Це не лише додає тактичну глибину, а й змушує гравців бути більш обережними та уважними під час вибору своїх ходів.

Також під час того як юніти гравця виконують дії, штучний інтелект заходиться в не активному стані та аналізує ситуацію тільки після завершення ходу гравця.

Окремим візуальні елементом є реалізація блокування панелі гравця під час того як його ігрові одиниці виконують поставлене завдання. Після виконаних дій гравець знову отримує змогу на коригування та продовження, або завершення, свого ходу.

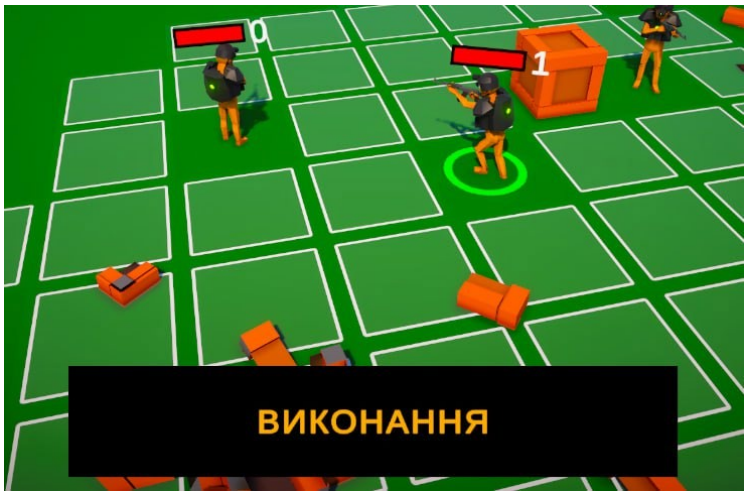


Рис. 3.14. Блокування панелі гравця під час роботи ШІ

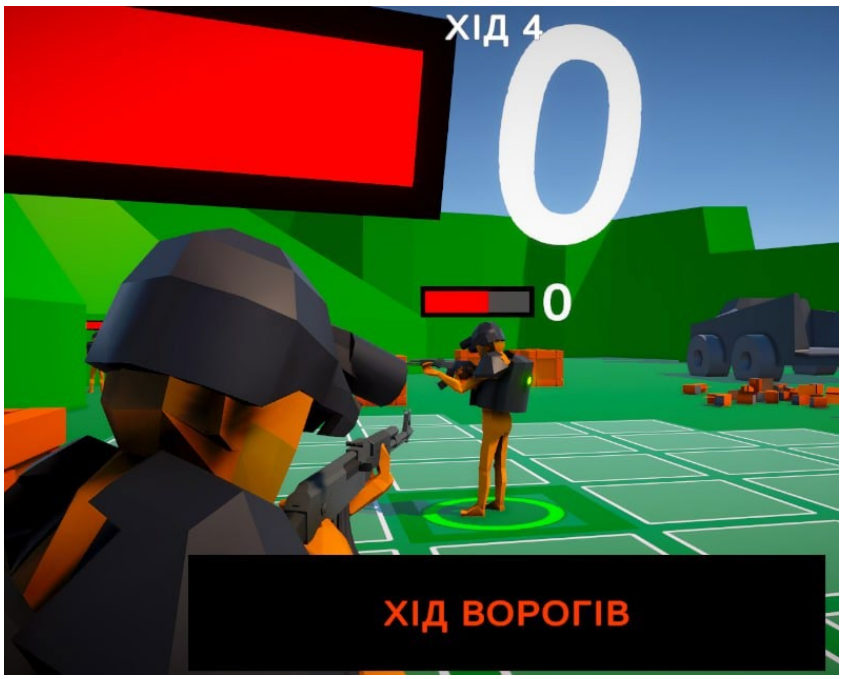


Рис. 3.15. Віднімання балів дії та планки здоров’я ігрового юніта

Для аналізу розуміння ситуації на ігровому полі також реалізовано візуалізацію стану здоров’я кожного юніта. Відповідно, на початку гри його розмір є максимальним, а отримуючи ураження від супротивника показник здоров’я поступово зменшується. Чим менше залишається здоров'я, тим вразливішим стає юніт перед подальшими атаками і ризикує вибуттям з гри. Дізнаватись і візуально спостерігати за змінами у рівні здоров'я кожного юніта допомагає гравцеві приймати рішення про те, коли застосовувати засоби лікування або визначати найбільш вразливих у їх тимчасових стратегіях.

Загальний стан здоров'я команди або групи юнітів є важливим фактором для стратегічного управління, оскільки гравець повинен зберігати баланс між атакою і захистом, враховуючи здоров'я кожного юніта для досягнення успішних результатів у грі.

Якщо показник здоров’я юніту досягне нуля, він вважатиметься мертвим та не повинен приймати подальшої участі в ході гри та бути ігровою одиницею. А отже після того, як такий юніт був переможений опонентом необхідна реалізація його виходу з гри. Таким чином буде також регулюватись баланс гри, роблячи її зрозумілою для гравця.

Рис. 3.16. Ініціалізація смерті юніта

3.1 Опис розроблених класів

У даному розділі наводиться детальний опис класів, реалізованих для розробки штучного інтелекту в грі жанру покрокової стратегії. Програма реалізує розроблений метод створення штучного інтелекту, який має на меті оптимізувати стратегічне прийняття рішень та поведінку супротивника в грі.

```
public class EnemyAI : MonoBehaviour
{
    // Стани, в яких може перебувати EnemyAI
    8 references
    private enum State
    {
        WaitingForEnemyTurn,
        TakingTurn,
        Busy,
    }
```

Рис. 3.17. клас EnemyAI. Стани штучного інтелекту

Як можна побачити на представленому фрагменті коду розроблений штучний інтелект супротивника має такі три стани як:

- Очікування ходу;
- Виконання;
- Зайнятий.

```
private State state;  // Поточний стан EnemyAI
private float timer;  // Таймер, який використовується для визначення тривалості дії в стані TakingTurn

Unity Message | 0 references
private void Awake()
{
    state = State.WaitingForEnemyTurn;  // Початковий стан - очікування ходу ворога
}

Unity Message | 0 references
private void Start()
{
    //  Відстеження ходу в грі
    TurnSystem.Instance.OnTurnChanged += TurnSystem_OnTurnChanged;
}
```

Рис. 3.18. Початок класу EnemyAI

У цьому фрагменті коду використовується змінна **state**, що представляє собою поточний стан об'єкта, який має ім'я EnemyAI. Також є змінна **timer**, що використовується для вимірювання часу тривалості дії в конкретному стані, особливо в стані Taking Turn.

Перед початком гри у функції Awake(), змінній state присвоюється початковий стан State.WaitingForEnemyTurn, що вказує на очікування ходу ворога.

У функції Start(), яка викликається при запуску об'єкта, встановлюється слухач подій. Конкретно, ця функція встановлює обробник подій TurnSystem_OnTurnChanged для події OnTurnChanged в об'єкті TurnSystem.Instance. Цей обробник подій відповідає за відстеження зміни ходу в грі і викликається при кожній його зміні.

Фрагмент коду Update() визначає логіку поведінки штучного інтелекту ворога в залежності від його поточного стану: очікування ходу ворога, виконання ходу чи перехід у стан "зайнятий".

Таким чином, визначається початковий стан об'єкта та встановлюється обробник подій для відстеження ходу в грі, що дозволяє відповідно реагувати на зміни станів і ходу в процесі гри.

Для вибору найкращої можливої дії розробляємого штучного інтелекту реалізовано Метод `GetBestEnemyAIAction`. Його основна функція - визначити оптимальну ворожу дію, яка має бути виконана в поточній ситуації гри. Цей метод

враховує доступні можливості для ворожого об'єкта, оцінює кожну можливу дію з певної позиції та вибирає найбільш вигідну засновуючись на розробленій формулі вище.

Основна мета методу полягає у визначенні найвигіднішої дії для ворожого об'єкта на основі розглянутих критеріїв, таких як поточний стан гри, доступні можливості ворожого об'єкта та вартість кожної дії.

Отже, метод `GetBestEnemyAIAction` забезпечує стратегічне прийняття рішення для ворожого об'єкта в грі, вибираючи на основі обміркованої стратегії та вартості можливі дії.

```
2 references
public EnemyAIAction GetBestEnemyAIAction()
{
    List<EnemyAIAction> enemyAIActionList = new List<EnemyAIAction>();

    List<GridPosition> validActionGridPositionList = GetValidActionGridPositionList();

    foreach (GridPosition gridPosition in validActionGridPositionList)
    {
        EnemyAIAction enemyAIAction = GetEnemyAIAction(gridPosition);
        enemyAIActionList.Add(enemyAIAction);
    }

    if (enemyAIActionList.Count > 0)
    {
        enemyAIActionList.Sort((EnemyAIAction a, EnemyAIAction b) => b.actionValue - a.actionValue);
        return enemyAIActionList[0];
    } else
    {
        // No possible
        return null;
    }

}
```

Рис. 3.19. Визначення найкращої дії штучним інтелектом

У наступному фрагменті коду реалізована функція `GetEnemyAIAction`, що відповідає за вибір дії для штучного інтелекту в грі покрокової стратегії. Для прийняття оптимального рішення щодо дії супротивника на конкретній позиції у грі, відбувається оцінка потенційних цілей, що містяться на заданій позиції.

Код виконує такі кроки:

1. Отримання кількості потенційних цілей (ворожих одиниць) на вказаній позиції у грі.

2. Створення нового об'єкта типу `EnemyAIAction` та ініціалізація його параметрів:

- `gridPosition`: позиція в грі, для якої розглядається ворожа дія.

- `actionValue`: значення дії, що обчислюється як добуток кількості потенційних цілей на заданій позиції на певну константу (у даному випадку, на 10). Ця константа відображає цінність дії в залежності від кількості цілей на позиції.

Описаний код визначає можливі дії для супротивника на конкретній позиції, враховуючи кількість цілей, що можуть бути атаковані з цієї точки координати. Чим більше потенційних ціле, тим більша вартість дії, що вказує на більш вигідний вибір для супротивника.

```
2 references
public override EnemyAIAction GetEnemyAIAction(GridPosition gridPosition)
{
    // Отримання кількості потенційних цілей (ворожих одиниць) на заданій позиції
    int targetCountAtGridPosition = unit.GetAction<ShootAction>().GetTargetCountAtPosition(gridPosit

    // Створення нового об'єкта EnemyAIAction та його ініціалізація
    return new EnemyAIAction
    {
        gridPosition = gridPosition,
        actionValue = targetCountAtGridPosition * 10,  // Цінність дії на основі кількості цілей
    };
}
```

Рис. 3.20. Пересування

Описаний фрагмент виконує оцінку потенційних дій для супротивника на конкретній позиції у грі покрокової стратегії.

```
2 references
public override EnemyAIAction GetEnemyAIAction(GridPosition gridPosition)
{
    // Отримання ворожої одиниці на вказаній позиції
    Unit targetUnit = LevelGrid.Instance.GetUnitAtGridPosition(gridPosition);


    return new EnemyAIAction
    {
        gridPosition = gridPosition,
                                    // Значення дії на основі формули, яка включає кількість здоров'я ворожої одиниці
        actionValue = 100 + Mathf.RoundToInt((1 - targetUnit.GetHealthNormalized()) * 100f),
    };
}
```

Рис. 3.21. Визначення пріоритетної цілі для стрільби

У поданому вище фрагменті коду реалізована функція `GetEnemyAIAction`, що виконує розрахунок дії для штучного інтелекту в грі покрокової стратегії для конкретної позиції у грі. Основна мета цієї функції полягає у визначенні дії супротивника в залежності від стану ворожої одиниці, розташованої на вказаних координатах.

Функція починається з отримання ворожої одиниці за вказаною точкою в грі. Далі, створюється новий об'єкт типу `EnemyAIAction`, де параметр `gridPosition` відображає позицію, на якій розглядається дія супротивника. Однак, основна увага приділяється параметру `actionValue`, який представляє собою значення дії. Значення `actionValue` обчислюється з урахуванням здоров'я ворожої одиниці за певною логікою: в даному випадку, значення `actionValue` визначається як сума 100 і округленого дробу, який є оберненим значенням здоров'я цільової ворожої одиниці, помноженого на 100. Такий підхід враховує той факт, що чим менше здоров'я у ворожої одиниці, тим вищий пріоритет має дія для супротивника. Визначаючи значення `actionValue` за такою логікою, програма намагається оптимізувати стратегічні вибори супротивника в грі на основі здоров'я цільової одиниці.

```
public override EnemyAIAction GetEnemyAIAction(GridPosition gridPosition)
{

    return new EnemyAIAction
    {
        gridPosition = gridPosition,
        actionValue = 200,  // Цінність дії встановимо 200
    };
}
```

Рис. 3.22. Атака мечем

Значення actionValue для цієї дії фіксується в розмірі 200 балів дії. Отримана цінність вказує на важливість чи силу даної дії в контексті покрокової стратегії гри. Враховуючи, що дія меча вбиває з одного удару, цей параметр вказує на високий пріоритет цієї дії для штучного інтелекту супротивника. Велика цінність вказує на

важливість вибору дії "меч" під час розгляду стратегічних рішень супротивника.

Наступний фрагмент визначає метод з ім'ям GetEnemyAIAction, який перевизначає метод із базового класу. Усередині цього методу створюється новий об'єкт типу EnemyAIAction і встановлюються його властивості.

```
public override EnemyAIAction GetEnemyAIAction(GridPosition gridPosition)
{
    return new EnemyAIAction
    {
        gridPosition = gridPosition,
        actionValue = 0,
    };
}
```

Рис. 3.23. Дія пересування

Об'єкт цього типу має два публічні властивості:

- gridPosition;
- actionValue.

Властивість gridPosition отримує значення, передане методу у вигляді аргументу gridPosition. Це значення призначається властивості об'єкта EnemyAIAction.

Отже, метод створює і повертає новий об'єкт EnemyAIAction з вказаною позицією на сітці та значенням дії, що дорівнює 0.

У наступному фрагменті коду визначено метод Update(), який викликається на кожному кадрі гри. Умовний оператор перевіряє, чи зараз гравець має хід, і виходить з методу, якщо це так. Далі використовується конструкція switch для обробки різних станів об'єкта EnemyAI.

У стані WaitingForEnemyTurn може викликатися додаткова логіка, яка наразі не визначена. У стані TakingTurn таймер зменшується на основі Time.deltaTime, і якщо він досягає або перевищує нуль, виконується перевірка на вдале виконання ходу. Якщо хід виконано успішно, переходиться до стану Busy, в іншому випадку викликається метод NextTurn() для переходу до наступного ходу.

У стані Busy немає конкретних дій, що визначаються в даному фрагменті, але

цей стан може служити для інших обчислень чи операцій, які можуть виконуватися, коли об'єкт EnemyAI знаходиться у стані зайнятості.

Цей метод Update() дозволяє об'єкту EnemyAI динамічно реагувати на зміни в грі та його власний стан, виконуючи відповідні дії на кожному кадрі гри. Взаємодія з TurnSystem та використання таймера визначають поведінку об'єкта, забезпечуючи відповідність правилам гри та інтелектуальну поведінку ворогів.

```
Unity Message | 0 references
private void Update()
{
    // Якщо зараз гравець має хід, вихід з методу
    if (TurnSystem.Instance.IsPlayerTurn())
    {
        return;
    }

    // Логіка в залежності від поточного стану EnemyAI
    switch (state)
    {
        case State.WaitingForEnemyTurn:

            break;
        case State.TakingTurn:
            // Зменшення таймера для визначення тривалості ходу
            timer -= Time.deltaTime;
            if (timer <= 0f)
            {
                // Якщо EnemyAI вдало виконав свій хід, перехід до стану Busy, в іншому випадку перехід до наступного ходу
                if (TryTakeEnemyAIAction(SetStateTakingTurn))
                {
                    state = State.Busy;
                }
                else
                {
                    TurnSystem.Instance.NextTurn();
                }
            }
            break;
        case State.Busy:

            break;
    }
}
```

Рис. 3.24. метод Update()

Ці методи і обробник події доповнюють логіку класу EnemyAI, дозволяючи йому ефективно реагувати на події в грі та взаємодіяти з іншими компонентами системи. Вони використовуються для управління станами об'єкта та ініціації дій під час визначених подій.

Наступним розглянемо обробник потій для класу EnemyAI. Він вбудований у внутрішню логіку об'єкта EnemyAI, роблячи його здатним ефективно пристосовуватися до змін у грі та виконувати визначені дії відповідно до логіки гри

та інтелектуальних стратегій ворогів.

```
private void SetStateTakingTurn()
{
    timer = 0.5f;  // Встановлення таймера для визначення тривалості дії в стані TakingTurn
    state = State.TakingTurn;  // Перехід до стану TakingTurn
}
```

Рис. 3.26. Метод SetStateTakingTurn

Отже, метод SetStateTakingTurn встановлює таймер на 0.5 секунди та змінює стан об'єкта EnemyAI на TakingTurn. Його можна викликати для ініціації ходу ворога та встановлення таймера для визначення тривалості дії в стані TakingTurn.

```
private void TurnSystem_OnTurnChanged(object sender, EventArgs e)
{
    // Під час зміни ходу перехід до стану TakingTurn, якщо зараз не хід гравця
    if (!TurnSystem.Instance.IsPlayerTurn())
    {
        state = State.TakingTurn;
        timer = 2f;  // Встановлення таймера для визначення тривалості дії в стані TakingTurn
    }
}
```

Рис. 3.27. Метод TurnSystem_OnTurnChanged

Метод TurnSystem_OnTurnChanged викликається при зміні ходу в грі. Якщо зараз не хід гравця, то стан об'єкта EnemyAI встановлюється в TakingTurn, і таймер встановлюється на 2 секунди для визначення тривалості дії в стані TakingTurn. Обробник події реагує на зміну ходу у грі. Якщо зараз не хід гравця, він переводить об'єкт у стан TakingTurn і встановлює таймер для керування тривалістю цього стану. Це забезпечує штучному інтелекту супротивника можливість реагувати на зміни ходу в грі та вживати відповідні дії.

Наступний метод TryTakeEnemyAIAction призначений для виконання дій EnemyAI для всіх ворожих одиниць. Він перебирає всі ворожі одиниці в грі та

спробує викликати метод TryTakeEnemyAIAction для кожної з них, передаючи також аргумент onEnemyAIActionComplete. Якщо хоча б одна спроба успішна, метод повертає true, в іншому випадку - false.

```
private bool TryTakeEnemyAIAction(Action onEnemyAIActionComplete)
{
    // Виконання дій EnemyAI для всіх ворожих одиниць
    foreach (Unit enemyUnit in UnitManager.Instance.GetEnemyUnitList())
    {
        if (TryTakeEnemyAIAction(enemyUnit, onEnemyAIActionComplete))
        {
            return true;
        }
    }

    return false;
}
```

Рис. 3.28. Продовження класу EnemyAI

Даний фрагмент коду вбудований у внутрішню логіку об'єкта EnemyAI, надаючи йому здатність динамічно реагувати на зміни у грі та виконувати певні дії, що відповідають стратегіям інтелекту ворогів. Метод TryTakeEnemyAIAction() визначає логіку виконання дій ворога для конкретної ворожої одиниці (Unit). Під час виконання метод переглядає всі можливі дії ворожої одиниці та вибирає ту, яка максимізує вартість для об'єкта EnemyAI, забезпечуючи оптимальний вибір дії відповідно до стратегій інтелектуального поведінки ворогів у грі. Це дозволяє покращити рівень геймплею та забезпечити більш розумне та вдумливе поведінку ворожих одиниць у відповідь на дії гравця.

Цей фрагмент коду є ключовим для забезпечення адаптивності штучного інтелекту супротивника (EnemyAI) у грі. Метод TurnSystem_OnTurnChanged дозволяє супротивнику вчасно реагувати на зміну ходу, що дозволяє встановити відповідний стан об'єкта EnemyAI та правильно керувати тривалістю його дій. Це важливо, оскільки штучний інтелект повинен ефективно аналізувати ситуацію на

кожному кроці гри, вибирати найбільш оптимальні дії і приймати стратегічні рішення для досягнення своїх цілей. Наприклад, встановлення таймера на певний час дозволяє контролювати тривалість ходу інтелекту супротивника, щоб він реагував швидко та адекватно до ситуації в грі. Такий підхід допомагає ворогам у грі вчасно відреагувати на події та підвищує реалістичність їхньої поведінки.

```
private bool TryTakeEnemyAIAction(Unit enemyUnit, Action onEnemyAIActionComplete)
{
    EnemyAIAction bestEnemyAIAction = null;
    BaseAction bestBaseAction = null;

    // Перегляд всіх можливих дій ворожої одиниці
    foreach (BaseAction baseAction in enemyUnit.GetBaseActionArray())
    {
        // Перевірка, чи може ворожа одиниця витратити очки дії на дану дію
        if (!enemyUnit.CanSpendActionPointsToTakeAction(baseAction))
        {
            // Ворог не може дозволити собі цю дію
            continue;
        }

        // Логіка вибору найкращої дії для EnemyAI
        if (bestEnemyAIAction == null)
        {
            bestEnemyAIAction = baseAction.GetBestEnemyAIAction();
            bestBaseAction = baseAction;
        }
        else
        {
            EnemyAIAction testEnemyAIAction = baseAction.GetBestEnemyAIAction();
            if (testEnemyAIAction != null && testEnemyAIAction.actionValue > bestEnemyAIAction.actionValue)
            {
                bestEnemyAIAction = testEnemyAIAction;
                bestBaseAction = baseAction;
            }
        }
    }
```

Рис. 3.30. метод TryTakeEnemyAIAction()

Всі можливі дії ворожої одиниці переглядаються за допомогою циклу foreach, де для кожної дії перевіряється, чи штучний інтелект супротивника може витратити бали дії на це. Якщо супротивник не може, процес переходить до наступної дії.

Логіка вибору найкращої дії для EnemyAI використовується за допомогою порівняння вартостей різних дій. Якщо поточна дія має більшу вартість (позначену як actionValue), ніж попередньо визначена найкраща дія, то визначається нова найкраща дія та базова дія.

Цей підхід дозволяє EnemyAI розглядати всі можливі дії ворожої одиниці та вибирати ту, яка максимізує його потенційну вигоду в грі. Використання порівнянь actionValue визначає вагомість кожної дії в контексті ігрового середовища, що може бути важливим для створення інтелектуальної та стратегічної поведінки ворогів.

У фрагменті коду, що розглядається наступним відбувається завершення методу TryTakeEnemyAIAction(). Після визначення найкращої дії та базової дії для ворожої одиниці, код перевіряє, чи існує можливість виконати цю дію.

```
// Виконання найкращої дії, якщо така існує
if (bestEnemyAIAction != null && enemyUnit.TrySpendActionPointsToTakeAction(bestBaseAction))
{
    bestBaseAction.TakeAction(bestEnemyAIAction.gridPosition, onEnemyAIActionComplete);
    return true;
}
else
{
    return false;
}
```

Рис. 3.31. Виконання найкращої дії, якщо така існує

Умовний оператор перевіряє, чи найкраща дія була успішно визначена (bestEnemyAIAction != null) і чи ворожа одиниця може витратити свої бали дії на цю дію за допомогою enemyUnit.TrySpendActionPointsToTakeAction (bestBaseAction).

Якщо обидві умови виконуються, то викликається метод bestBaseAction.TakeAction(...), що представляє виконання обраної дії. Після цього викликається переданий колбек onEnemyAIActionComplete, і метод повертає true для підтвердження успішного виконання дії.

У випадку, якщо жодна дія не була визначена або виконання неможливе, метод повертає false. Це може вказувати на те, що ворог не має можливості виконати жодну дію в даному контексті або що відсутні додаткові можливості взаємодії з іншими об'єктами в грі.

Отже таким чином реалізована ігрова логіка для штучного інтелекту супротивника в контексті тактичної гри, за розробленим математичним методом. Цей код відповідає за інтелект супротивника, який базується на виборі та виконанні оптимальних дій під час їхнього ходу.

Клас EnemyAI має внутрішній стан, який визначає його поведінку в різних ситуаціях гри. Зокрема, це може бути очікування ходу гравця, самостійний хід ворога або стан зайнятості, коли ворог виконує свою дію.

Інтеракція з системою гри визначена через обробник події, який реагує на зміну ходу в грі та ініціює хід ворога. Крім того, визначено методи для вибору та виконання оптимальної дії ворога.

Метод TryTakeEnemyAIAction() визначає найкращу дію для ворога, розглядаючи можливі варіанти та визначаючи їхню вартість через поле actionValue. Враховуються обмеження витрати очок дії та взаємодія з іншими об'єктами гри.

Додаткові методи, такі як SetStateTakingTurn() та TurnSystem_OnTurnChanged(), допомагають у керуванні станами та реагуванні на зміни у грі.

Узагальнено, цей код втілює інтелект штучного супротивника, дозволяючи йому динамічно реагувати на зміни в грі, вибирати оптимальні дії та взаємодіяти з іншими елементами гри, забезпечуючи тактичну та стратегічну ігрову динаміку.

3.4 Результати тестування розробленої методик

Цілі тестування розробленого штучного інтелекту для супротивника в іграх жанру покрокової стратегії важливі для визначення його функціональності та ефективності в грі.

Перша ціль полягає у визначенні та оцінці ефективності самого інтелекту, а основний акцент буде зроблено на тому, як добре штучний інтелект адаптується до різних сценаріїв гри, приймає рішення та виконує стратегії в умовах, що змінюються. Ця мета включає в себе оцінку швидкості прийняття рішень, аналізу ситуації та здатності до оптимальних дій в грі, що потребує стратегічного мислення.

Друга ціль полягає у порівнянні результатів гри проти розробленого штучного інтелекту з іншими алгоритмами чи стратегіями. Це допоможе з'ясувати, наскільки ефективний розроблений інтелект в порівнянні з проаналізованими раніше методами такими як: RHEA та MCTS. Порівняльний аналіз дозволить визначити сильні та слабкі сторони розробленого методу, його переваги та недоліки, що допоможе в розумінні конкурентних переваг розробленого штучного інтелекту.

Після чого буде сформовано результативну таблицю для порівняння результатів та підведено підсумки створюваної методики створення сценарію поведінки штучного інтелекту супротивника в іграх жанру покрокової стратегії.

Отже , відповідно до поставлених цілей було створене програмне забезпечення на якому і тестувалася розроблена методика.

Порівняння з гравцями є важливим етапом, оскільки людський гравець може мати власні унікальні методи та підходи до гри, які відрізняються від стандартних стратегій або алгоритмів, врахованих у штучному інтелекті. Оцінка успішності штучного інтелекту порівняно з людиною допомагає визначити його реальну ефективність, а також ідентифікувати можливі області для подальшого вдосконалення.

Взаємодія з людьми дозволяє виявити та оцінити непередбачені сценарії чи стратегії, які можуть виникнути під час гри. Такий аналіз дає можливість ідентифікувати та вдосконалювати алгоритми для кращої адаптації до різноманітних стратегій гри, що сприяє підвищенню реалістичності та виразності поведінки штучного інтелекту у контексті покрокової стратегії.

Таке порівняння також дозволяє визначити, наскільки близько штучний інтелект досягає рівня гри, який може конкурувати з реальними гравцями та забезпечувати цікавий ігровий досвід. Було проведено 50 тестових ігор для розробленого штучного інтелекту супротивника проти людини, що показало на тестах такий результат:

Таблиця 3.1.

Результати тестування розробленого методу

Метод	**Коефіцієнт виграшу проти людини**	**Середня кількість ходів за переможну гру**	**Середня кількість набраних балів цінності**	**Ефективність**
Розроблений метод	56%	9	4700	522

Як можна бачити за табличкою оцінки результативності методу відповідно до сформованих раніше формул, то як супротивник для людини метод показав себе достатньо ефективно. Він показав значний коефіцієнт виграшу у розмірі 56%. Це свідчить про його здатність до успішних стратегічних рішень, які дозволяють здобувати перемоги у грі проти людини. До того ж при врахуванні усіх показників, метод демонструє високу ефективність зі значенням 522. Це підтверджує високий рівень досягнення стратегічних цілей та успішний результат у грі проти людини.

Для порівняння метод RHEA у відповідній статистиці з тією ж кількістю ігор показав ось такий результат:

Таблиця 3.2.

Результати тестування методу RHEA

Метод	**Коефіцієнт виграшу проти людини**	**Середня кількість ходів за переможну гру**	**Середня кількість набраних балів цінності**	**Ефективність**
Розроблений метод	55%	11	5687	517

Як можна бачити за таблицею загальна ефективність розробленого методу у грі проти гравця перевищує ефективність методу RHEA. Це свідчить про кращі стратегічні можливості розробленого методу для досягнення успіху в ігровому процесі. Відповідно до проведеного аналізу, можна зробити висновок, що метод RHEA працює менш ефективно ніж розроблений метод на тестовому порівнянні з грою проти гравця.

Також додатково необхідно протестувати ефективність розробленого методу зіставивши його проти штучного інтелекту супротивника на базі методу RHEA. Це додаткове тестування виявляється ключовим для отримання глибшого уявлення про те, які саме аспекти та стратегії виявляються переважними під час гри між штучними інтелектами, розробленими на основі різних методик. Порівняння їх ефективності у грі між собою може виявити сильні та слабкі сторони кожного підходу до створення штучного інтелекту для ігрових середовищ. Для цього було проведено додаткові 50 ігрових раундів де розроблений штучний інтелект на основі запропонованого методу змагався з вже існуючим на основі RHEA. І результати були записані в наступну таблицю:

Таблиця 3.3.

Порівняльна таблиця тестування розробленого методу проти RHEA

Метод	**Коефіцієнт виграшу**	**Середня кількість ходів за переможну гру**	**Середня кількість балів цінності за проведені ігри**	**Ефективність досягнення перемоги**
Розроблений метод	52%	9	4850	538
RHEA	48%	11	5800	527

Це вказує на більшу здатність розробленого методу до досягнення успішних результатів у грі в покроковій стратегії.

Отже підводячи підсумок проведеного тестування можна зробити такі висновки, що розроблений метод штучного інтелекту для гри в покрокову стратегію виявився більш ефективним у порівнянні з методом RHEA. Розроблений метод показав кращі показники коефіцієнта виграшу проти людини 55%, та 52% проти існуючого методу, що свідчить про його здатність до успішних стратегічних рішень.

Також розроблений метод продемонстрував меншу середню кількість ходів за переможну гру, а саме 9 проти 11 у RHEA, вказуючи на його здатність до ефективного досягнення успішних результатів швидше за інші методи.

У порівнянні з RHEA, розроблений метод показав більшу середню кількість набраних балів цінності за проведені ігри, в перерахунку на кількість ходів для перемоги. Тому що ефективність досягнення перемоги виявилась вищою, а саме 538 проти 527.

Отже, отримані результати підтверджують перевагу розробленого методу штучного інтелекту у вигляді кращої стратегічної ефективності та здатності до успішного досягнення перемоги у грі в покрокову стратегію порівняно з методом RHEA.

ВИСНОВКИ

Після аналізу та проведення досліджень в рамках магістерської роботи, можна зробити такі висновки:

У результаті досліджень була розроблена нова модель поведінки штучного інтелекту супротивника для покрокових стратегій. Ця модель забезпечує максимізацію кількості перемог штучного інтелекту, одночасно мінімізуючи кількість використаних балів дій.

В рамках дослідження була розроблена та впроваджена схема функціонування програмного забезпечення, яка реалізує розроблений метод для штучного інтелекту супротивника в іграх жанру покрокової стратегії.

Проведено серію тестових ігор для визначення ефективності розробленого методу, в результаті чого цей метод здобув перемогу у 26 з 50 ігор.

Розрахунки та порівняння розробленого методу із RHEA продемонстрували, що розроблений метод є більш ефективним на 11,61 одиницю ефективності.

Ці результати свідчать про високий потенціал та успішність розробленого методу штучного інтелекту супротивника для покрокових стратегічних ігор.

ПЕРЕЛІК ПОСИЛАНЬ

1. X-COM: UFO defense on steam. Welcome to Steam. URL: https://store.steampowered.com/app/7760/XCOM_UFO_Defense (дата звернення: 11.11.2023).
2. Jagged alliance 1: gold edition on steam. Welcome to Steam. URL: https://store.steampowered.com/app/283270/Jagged_Alliance_1_Gold_Edition (дата звернення: 15.11.2023).
3. Daskalakis, C., Golowich, N. & Zhang, K.. (2023). The Complexity of Markov Equilibrium in Stochastic Games. Proceedings of Thirty Sixth Conference on Learning Theory, in Proceedings of Machine Learning Research https://proceedings.mlr.press/v195/daskalakis23a.html.
4. Jiayu Zhou. Studies Advanced in Artificial intelligence based Game. ACE (2023) Vol. 8: 481-487. DOI: 10.54254/2755-2721/8/20230255.
5. M. F. Sanjaya, H. Pratiwi, and P. Adytia, “Application of the Finite State Machine Method in the Desktop-Based ‘Heroes Of Dawn’ RPG Turn-Based Game”, TEPIAN, vol. 2, no. 2, pp. 69–73, Jun. 2021.
6. Muqing Ge. Review on the application of deep learning algorithms in video game AI agent. ACE (2023) Vol. 4: 548-553. DOI: 10.54254/2755-2721/4/2023322.
7. Dockhorn, A.; Lucas, S. Choosing Representation, Mutation, and Crossover in Genetic Algorithms. IEEE Comput. Intell. Mag. 2022, 17, 52–53
8. R. D. Gaina, S. Devlin, S. M. Lucas and D. Perez, "Rolling horizon evolutionary algorithms for general video game playing", IEEE ToG, pp. 1-11, 2021.
9. Oleksandr Svyrydenko, Oleksandr Svyrydenko. “Tactical artificial intelligence using neural network for real-time strategy”, Проблеми інформатизації та управління, Vol. 2 No. 66 (2021), pp. 27-33
10. Justesen, N., Uth, L. M., Jakobsen, C., Moore, P. D., Togelius, J., & Risi, S. (2019). Blood bowl: A new board game challenge and competition for AI. In 2019 IEEE Conference on Games (CoG) (pp. 1-8). IEEE.
11. Y. Chen, L. Bai, Y. Tan, Y. Liu and H. Nan, "Research on turn-based war chess game based on reinforcement learning," 2023 IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA), Shenyang, China, 2023, pp. 561-565, doi: 10.1109/ICPECA56706.2023.10075872.
12. Unity documentation. Unity Documentation. URL: https://docs.unity.com/ (дата звернення: 10.11.2023).
13. Visual Studio documentation. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022 (дата звернення: 23.11.2023).
14. Yu-Jhen Hsu, Diego Perez-Liebana.(2020). MCTS pruning in Turn-Based Strategy Games. [електронний ресурс] URL: http://www.diego-perez.net/papers/MCTSPruningTribesAIIDE2020.pdf (дата звернення 23.10.2023)
15. Devavrat Shah.Varun Somani, Qiaomin Xie, Zhi Xu. (2020) On Reinforcement Learning for Turn-based Zero-sum Markov Games. FODS '20:

Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference. (pp. 139–148) https://doi.org/10.1145/3412815.3416888

16. Alexander Dockhorn, Jorge Hurtado-Grueso, Dominik Jeurissen, Diego Perez-Liebana (2020). STRATEGA - A General Strategy Games Framework. [електронний ресурс] URL: https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/69386/Perez-Liebana%20Stratega%20-%20A%20General%20Strategy%20Games%20Framework%202020%20Accepted.pdf?sequence=2 (дата звернення 23.10.2023)

17. Perez-Liebana, D., Hsu, Y.-J., Emmanouilidis, S., Khaleque, B., & Gaina, R. (2020). Tribes: A New Turn-Based Strategy Game for AI Research. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 16(1), 252-258. https://doi.org/10.1609/aiide.v16i1.7438

18. Jialian Li, Yichi Zhou, Tongzheng Ren, Jun Zhu Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR 124:201-210, 2020.

19. Z Tang, Y Zhu, D Zhao et al., "Enhanced Rolling Horizon Evolution Algorithm with Opponent Model Learning", IEEE Transactions on Games, 2020.

20. Sebastiano M. Cossu. Beginning Game AI with Unity: Programming Artificial Intelligence with C#. Apress; 1st ed. edition (December 6, 2020) 160p.

21. M. B. Maurice Bergsma and P. Spronck, “Adaptive Spatial Reasoning for Turn-Based Strategy Games”, AIIDE, vol. 4, no. 1, pp. 161-166, Sep. 2008.

22. Dockhorn, A.; Lucas, S. Choosing Representation, Mutation, and Crossover in Genetic Algorithms. IEEE Comput. Intell. Mag. 2022, 17, 52–53

23. Lu, Y.; Li, W. Techniques and Paradigms in Modern Game AI Systems. Algorithms 2022, 15, 282. https://doi.org/10.3390/a15080282

24. Verma, R., Chaudhary, A., Gupta, D., Kumar, A. (2023). Artificial Intelligence in Gaming. In: Jain, R., Travieso, C.M., Kumar, S. (eds) Cybersecurity and Evolutionary Data Engineering. ICCEDE 2022. Lecture Notes in Electrical Engineering, vol 1073. Springer, Singapore. https://doi.org/10.1007/978-981-99-5080-5_18

25. Krishnendu Chatterjee, Tobias Meggendorfer, Raimundo Saona, and Jakub Svoboda. Faster Algorithm for Turn-based Stochastic Games with Bounded Treewidth Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). 2023, 4590-4605

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ
(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-ТЕЛЕКОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

МАГІСТЕРСЬКА РОБОТА

«РОЗРОБКА МЕТОДИКИ ГЕНЕРАЦІЇ СЦЕНАРІЮ ПОВЕДІНКИ СУПРОТИВНИКА В ІГРАХ ЖАНРУ ПОКРОКОВОЇ СТРАТЕГІЇ»

Виконав: студент групи ПДМ-64, Кузьменко Єгор Олексійович

Керівник: доктор філософії, доцент кафедри ІПЗ Дібрівний Олесь Андрійович

Київ - 2023

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення якості роботи супротивника в грі жанру покрокової стратегії за рахунок створення власного методу.

Об'єкт дослідження: процес генерації поведінки супротивника в грі жанру покрокової стратегії.

Предмет дослідження: алгоритми штучного інтелекту супротивника в грі.

ІСНУЮЧІ МЕТОДИ ВИРІШЕННЯ ЗАДАЧІ ГЕНЕРАЦІЇ СЦЕНАРІЮ ПОВЕДІНКИ СУПРОТИВНИКА В ІГРАХ ЖАНРУ ПОКРОКОВОЇ СТРАТЕГІЇ

Метод	Недоліки	Переваги
Випадкові дії	Відсутність стратегії	Надійність Легкість в створенні
One-Step Look Ahead	Через обмеженість глибини кроків програє Monte Carlo Tree Search методу	Оцінка якості наступного кроку
Monte Carlo Tree Search	Зниження продуктивності при збільшені кількості варіантів дій Відсутність запам'ятовування неправильних шляхів	вміння розгортати розрахунок дій для пошуку найкращого по значенням результату
Rolling Horizon Evolutionary Algorithms	Зниження ефективності при великих значеннях Відсутність якісної перевірки на не правильні данні	Прогнозування найкращих варіантів при малих горизонтах дій
Rule Based	Обмежений в виконанні Передбачуваність	Надійність Легкість створення в проєкті

3

ПОПАРНЕ ПОРІВНЯННЯ ПЕРЕМОГ ІСНУЮЧИХ МЕТОДІВ ВІДНОСНО ОДИН ОДНОГО

ГРАВЕЦЬ	RHEA	RB	MCTS	OSLA	RND
RHEA	*	58.60%	63.00%	74.80%	100.00%
RB	41.40%	*	56.20%	70.20%	98.80%
MCTS	37.00%	43.80%	*	60.80%	98.60%
OSLA	25.20%	29.80%	39.20%	*	99.40%
RND	0.00%%	1.20%%	1.40%	0.60%	*

4

СЦЕНАРІЙ ПОВЕДІНКИ СУПРОТИВНИКА

Етапи сценарію	Дії
Обробка стану	Поки хід гравця – стан очікування. Коли наступає хід суперника – дія.
Перевірка	Перевірка оточення для виявлення ворогів та перешкод.
Рішення	Якщо ворог у видимому полі зору і можна вдатися до атаки, то відбувається атака зі списку доступних атак. Якщо ворог у видимому полі зору, але недосяжний, пересування в його напрямку, використовуючи стратегію навігації. Якщо є перешкоди на шляху до ворога, обійти або ж знищити перешкоду.
Навігація	Використання алгоритмів навігації для руху в напрямку ворога чи обходу перешкод.
Контроль	Контроль вищезазначених правил та не порушував стратегію гри.

5

МАТЕМАТИЧНА МОДЕЛЬ РОЗРАХУНКУ НАЙБІЛЬШОЇ ЦІННОСТІ ДІЇ

$$x = argmax_{x \in A(m)} \left(X_m + \sqrt{\frac{l}{k}}\right),$$

де:
x - обрана дія;
X - значення вибору дії x із списку доступних
m - індекс обраної дії зі списку;
l - кількість разів, коли список m був відвіданий;
k - кількість разів, коли список m був відвіданий, і було обрано дію x;
$A(m)$ - множина дій.

Таблиця можливих дій

Можливі дії	Вартість (бали дії)	Бали цінності
Дії атаки		
Постріл	1	100
Граната	2	200
Атака мечем	3	300
Дії пересування		
Пересування	1	50
Знищення перепони	1	50

6

КРИТЕРІЇ ЯКОСТІ ШТУЧНОГО ІНТЕЛЕКТУ СУПРОТИВНИКА В ГРІ ЖАНРУ ПОКРОКОВОЇ СТРАТЕГІЇ

Критерій	Характеристика оцінювання
Коефіцієнт виграшу	Кількість перемог відносно іншого методу
Стратегічне планування	$P = \sum x_i,$ $\sum c_i * x_i \leq i,$ де: x_i – обрана дія; c_i – вартість виконаної дії; i – кількість балів дії.
Ефективність досягнення перемоги	$Efect = \frac{\bar{S}}{\bar{n}}$

$$Efect = \frac{\bar{S}}{\bar{n}},$$

$$\bar{S} = \frac{\sum S_j}{j}, \bar{n} = \frac{\sum N_j}{j},$$

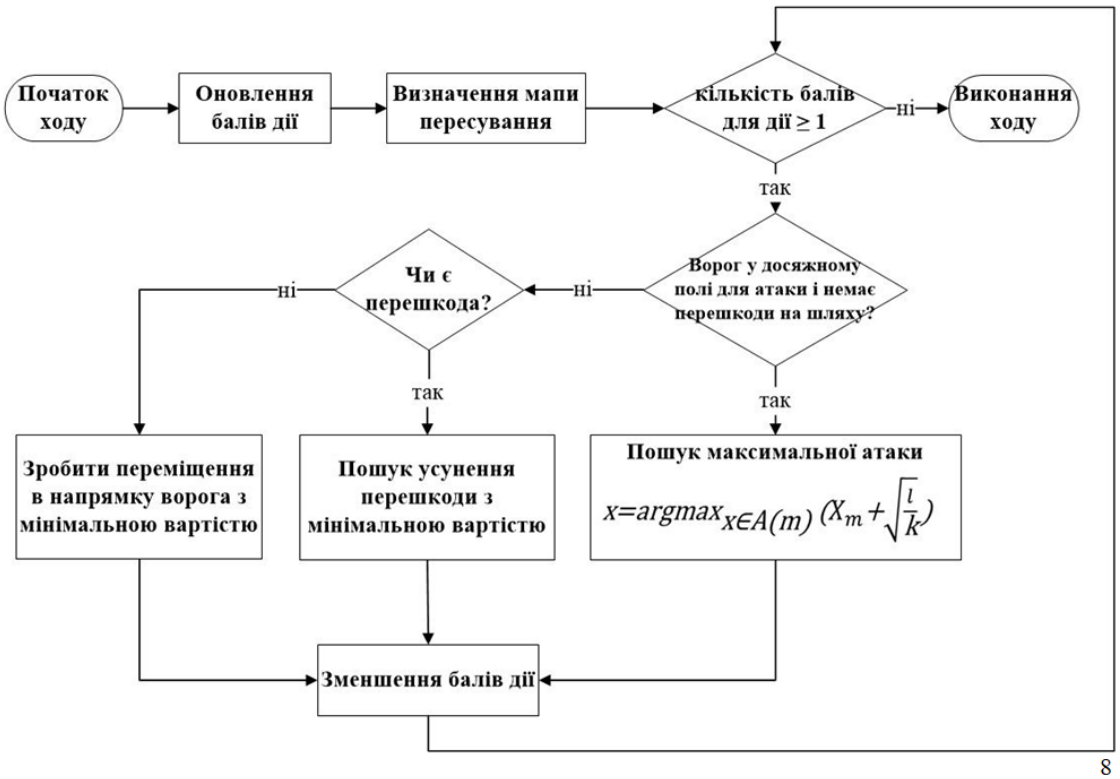
$$S = \sum P_j,$$

де :

$\bar{S}$- Середня кількість набраних балів цінності за проведені ігри;
$\bar{n}$- середнє значення кількості ходів за проведені переможні ігри;
n- кількість ходів за гру;
P- стратегічне планування;
j – кількість проведених ігор.

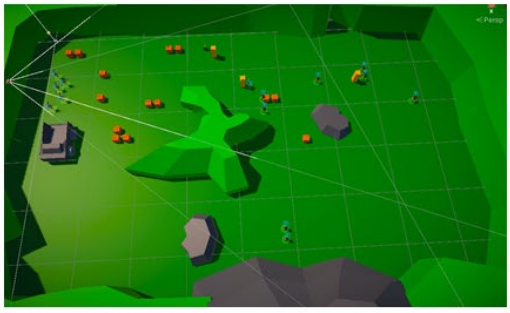
7

АЛГОРИТМ РОБОТИ РЕАЛІЗОВАНОГО

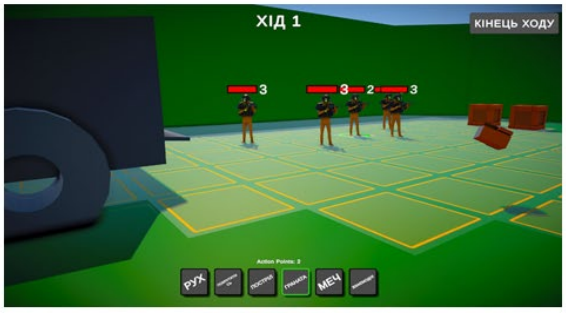


8

РЕЗУЛЬТАТИ ВИКОНАННЯ

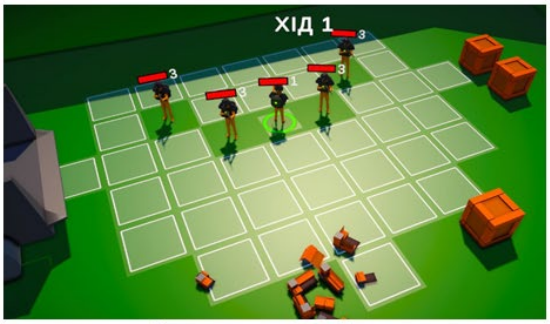


Загальна карта ігрового поля



Ігровий інтерфейс

Ігрові персонажі



Система переміщення

9

РЕЗУЛЬТАТИ ТЕСТУВАННЯ РОЗРОБЛЕНОГО МЕТОДУ В ПОРІВНЯННІ З RHEA

Метод	Коефіцієнт виграшу	Середня кількість ходів за переможну гру	Середня кількість балів цінності за проведені ігри	Ефективність досягнення перемоги
Розроблений метод	52%	9	4850	538
RHEA	48%	11	5800	527

ВИСНОВКИ

1. Розроблена власна модель поведінки штучного інтелекту супротивника для покрокових стратегій, що максимізує перемоги штучного інтелекту з мінімізацією використаних балів дій.
2. Розроблено схему функціонування програмного забезпечення, що реалізує створений метод для штучного інтелекту супротивника ігор жанру покрокової стратегії.
3. Проведено 50 тестових ігор з яких перемогу отримав розроблений метод 26 разів.
4. Проведено розрахунки згідно з якими розроблений метод є ефективнішим за RHEA на 11 одиниць ефективності.

АПРОБАЦІЯ

Тези:

1. Кузьменко Є.О., Дібрівний О.А. Методики генерації сценарію поведінки супротивника в іграх жанру покрокової стратегії. IV Проблеми комп'ютерної інженерії : Науково-практ. конф., м. Київ, 1 груд. 2023 р. Київ, 2023. С. 11–15.
2. Кузьменко Є.О., Дібрівний О.А. Огляд програмних засобів для створення штучного інтелекту для ігри жанру покрокової стратегії. IV Проблеми комп'ютерної інженерії : Науково-практ. конф., м. Київ, 1 груд. 2023 р. Київ, 2023. С. 11–15.

Стаття:

Кузьменко Є. О., Дібрівний О. А. Методики генерації сценарію поведінки супротивника в іграх жанру покрокової стратегії. Зв'язок. 2023. Т. 6, № 166. (подано до друку)

ДЯКУЮ ЗА УВАГУ!