Міністерство освіти та науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)
Кафедра *Системотехніки*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

ГЮІК 501600.009 ПЗ
(позначення документа)

Розробка та дослідження методів штучного інтелекту у ігрових застосунках
(тема)

Виконав:
студент *2* курсу, групи *СПРм-20-1*
Муравльов Р.Ф.
(прізвище, ініціали)
спеціальності *122 – Комп'ютерні науки*
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Системне проектування*
(повна назва освітньої програми)
Керівник *доц. Хряпкін О.В.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри системотехніки _____ *Гребеннік І.В.*
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність *122 – Комп'ютерні науки*
(код і повна назва)

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____
2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Муравльову Роману Федоровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та дослідження методів штучного інтелекту у ігрових застосунках
затверджена наказом по університету від 08.11 2021 р. № 1516СТ
2. Термін подання студентом роботи до екзаменаційної комісії 16 грудня 2021 р.
3. Вихідні дані до роботи: Перелік використовуваних програмних засобів: ОС Windows 10, Unity 3D, ML-Agents tool kit.
4. Зміст пояснювальної записки (перелік питань, що потрібно розробити) 4.1 Вступ. 4.2 Аналіз предметної області. 4.3 Дослідження обраних методів навчання штучного інтелекту. 4.4 Розробка та порівняння методів штучного інтелекту у ігрових застосунках. 4.5 Висновок.
5.Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 5.1 гра Elite: Dangerous (зверху) та Пак-Мен (знизу) 5.2 Візуальне зображення набору правил, що керують привидами у грі Пак-Мен. 5.3 Розташування станів, де стрілки відображають можливі зміни стану 5.4 Схема алгоритму Навчання з підкріпленням 5.5 Автомат Багаторукий бандит 5.6 Нагорода для стратегії з різними є-жадібності 5.7 Інтерфейс Unity 5.8 Приклади готових симуляцій ML-Agents в Unity 5.9 Інтерфейс Microsoft Visual Studio 5.10 Гра

ЗігЗаг 5.11 Присвоювання нагороди 5.12 Віднімання нагороди 5.13 Фрагмент тренувальної конфігурації 5.14 Початок роботи ML-Agents 5.15 Початок симуляції гри для навчання ШІ 5.16 Графік накопичення нагород агента 5.17 Графік тривалості кожної спроби 5.18 Процес навчання агента 5.18 Графік процесу навчання за допомогою метода Q-learning протягом більш 3000 спроб

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання завдання на кваліфікаційне проектування*	*08.11.2021*	*Виконано*
2	*Аналіз вихідних даних та літератури за темою кваліфікаційної роботи*	*13.11.2021*	*Виконано*
3	*Постановка задачі та вибір методу її вирішення*	*16.11.2021*	*Виконано*
4	*Проведення експериментальних досліджень*	*18.11.2021*	*Виконано*
5	*Оформлення пояснювальної записки*	*21.11.201*	*Виконано*
6	*Підготовка презентації*	*25.11.2021*	*Виконано*
7	*Подання закінченої роботи науковому керівникові*	*10.12.2021*	*Виконано*
8	*Попередній захист*	*13.12.2021*	*Виконано*
9	*Подання роботи на рецензування*	*15.12.2021*	*Виконано*
10	*Подання роботи до екзаменаційної комісії*	*16.12.2021*	*Виконано*

Дата видачі завдання 08 листопада 2021 р.

Студент _____ Муравльов Р.Ф.
(підпис) (прізвище, ініціали)

Керівник роботи _____ доц. Хряпкін О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до звіту з кваліфікаційної практики містить 109 сторінок, 27 рисунки, 26 літературних джерел.

ШТУЧНИЙ ІНТЕЛЕКТ, ШІ, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ІГРОВИЙ ШТУЧНИЙ ІНТЕЛЕКТ, ГРА, ІГРОВИЙ ЗАСТОСУНОК, АГЕНТ, ЮНИТ, ГЕЙМПЛЕЙ, ОБ'ЄКТ.

Метою роботи є розробка та дослідження методів штучного інтелекту у ігрових застосунках.

Об’єктом дослідження є аналіз існуючих методів штучного інтелекту. Також визначення підходу до вирішення проблеми з навчанням штучного інтелекту у ігрових застосунках. Зроблені порівняння існуючих методів навчання штучного інтелекту, зроблен порівняльний аналіз алгоритмів, та виділен найбільш ефективний алгоритми навчання, який задовольняє поставлену задачу.

Результатом кваліфікаційної роботи є виявлений метод штучного інтелекту, який найбільше підходить для застосування у ігрових застосунках.

ABSTACT

The explanatory note to the report on attestation practice contains 109 pages, 27 figures, 26 literary sources.

ARTIFICIAL INTELLIGENCE, AI, MACHINE LEARNING, NEURAL NETWORKS, GAME ARTIFICIAL INTELLIGENCE, GAME, GAME APPLICATION, AGENT, UNIT, GAMEPLAY.

The aim of the work is to develop and research methods of artificial intelligence in game applications.

The object of research is the analysis of existing methods of artificial intelligence. Also, the definition of the approach to solving the problem of learning artificial intelligence in gaming applications. Comparisons of existing methods of teaching artificial intelligence, comparative analysis of algorithms, and the most effective learning algorithms that meet the task.

The result of the certification work is the identified method of artificial intelligence, which is most suitable for use in gaming applications.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Геймплей - це компонент гри, що відповідає за взаємодію гри та гравця.

Юніт – це ігрова одиниця (персонаж) у комп'ютерних стратегічних іграх.

Агент – це це будь-який об'єкт, який може сприймати своє оточення за допомогою датчиків та робити дії з використанням виконавчих механізмів

ВСТУП

Ігрова промисловість зараз переживає бурхливе зростання останні десятиліття. Продукти, які створюють великі ігрові компанії, показують, що створення ігор може виступати у вигляді деякого мистецтва. Комп'ютерні ігри дозволяють змагатися між гравцями або гравцями та комп'ютером. Комп'ютером виступає ігровий штучний інтелект (ШІ). Основна роль, яку він виконує це симуляція поведінки людини. Він може маскуватися під маскою людини, або виступати у вигляді простих ботів (неігрові персонажі), у яких перелік дій дуже обмежений і примітивний. Завдання програмування ботів з'являється під час створення більшості ігрових проектів. Симуляція поведінки людини зазвичай є нетривіальним завданням. Але для тих та інших видів штучним інтелектом витрачається дуже багато часу та ресурсів.

Тим не менш, основне обмеження для ШІ в іграх є швидка робота та обчислювальні ресурси комп'ютера . Тому, в ігровому ШІ основний принцип у тому, що поведінка імітується. Іншими словами, ШІ для ігор є більш штучнішим, ніж інтелектом у прямому розумінні цього слова. Система ШІ може бути вкрай проста і являти собою набір правил або може бути досить складною і виконувати роль командувача армії противника, з якою належить битися гравцю.

Традиційні дослідження в галузі ШІ намагаються створити справжній інтелект, хоча й штучними засобами. Деякі проекти намагаються створити штучний інтелект, який може навчатися та взаємодіяти в суспільстві, а також проявляти емоції.

Для цілей сучасних ігор справжній штучний інтелект перевищує вимоги розважального програмного забезпечення. Ігровий ШІ не повинен бути розумним або самосвідомим, йому не потрібно дізнаватися про що-небудь поза межами ігрового процесу. Справжня мета штучного інтелекту в іграх полягає в тому, щоб імітувати розумну поведінку, надаючи гравцеві реальний виклик — виклик, який гравець може потім подолати.

1 АНАЛІЗ ПРЕДМЕНОЇ ОБЛАСТІ

1.1 Підходи та концепції штучного інтелекту

Менш ніж через десять років після того, як взламали нацистську шифруючу машину Enigma, що допомогло перемогти у Другій світовій війні, математик Алан Тьюрінг вдруге змінив історію, поставивши просте запитання: «Чи можуть машини мислити?» Стаття Тьюринга «Обчислювальна техніка та інтелект» (1950) та її подальший тест Тьюринга встановили фундаментальну мету та бачення штучного інтелекту.

За своєю суттю ШІ є галуззю інформатики, яка прагне дати чітку відповідь на запитання Тьюринга. Це спроба відтворити або імітувати людський інтелект у машинах. Розширена мета штучного інтелекту викликала багато запитань і дискусій. Настільки, що жодне єдине визначення поля не є загальноприйнятим.

Основним обмеженням у визначенні ШІ як просто «машин, які є розумними» є те, що він насправді не пояснює, що таке штучний інтелект? Що робить машину розумною? ШІ – це міждисциплінарна наука з кількома підходами, але досягнення в машинному навчанні та глибокому навчанні створюють зміну парадигми практично в кожному секторі технологічної індустрії [1].

У своєму новаторському підручнику «Штучний інтелект: сучасний підхід» автори Стюарт Рассел і Пітер Норвіг підходять до цього питання, об’єднавши свою роботу навколо теми інтелектуальних агентів у машинах. З огляду на це, ШІ є «вивченням агентів, які отримують сприйняття з навколишнього середовища та виконують дії».

Норвіг і Рассел продовжують досліджувати чотири різні підходи, які історично визначили область ШІ:

1. Мислення по-людськи
2. Мислення раціонально
3. Діючи по-людськи
4. Діяти раціонально

Перші дві ідеї стосуються процесів мислення та міркування, а інші – поведінки. Норвіг і Рассел зосереджуються особливо на раціональних агентах, які діють для досягнення найкращого результату, відзначаючи, що «всі навички, необхідні для тесту Тьюринга, також дозволяють агенту діяти раціонально» [2].

Патрік Вінстон, професор Форда штучного інтелекту та інформатики в Массачусетському технологічному інституті, визначає ШІ як «алгоритми, що увімкнені через обмеження, які підтримуються моделями, орієнтованими на цикли, які пов'язують мислення, сприйняття та дії разом».

Хоча пересічній людині ці визначення можуть здатися абстрактними, вони допомагають зосередитися на цій галузі як області інформатики та забезпечують план наповнення машин і програм машинним навчанням та іншими підмножинами штучного інтелекту [3].

1.2 Чотири типу штучного інтелекту

1.2.1 Реактивні машини у понятті штучного інтелекту

Реактивна машина дотримується найосновніших принципів ШІ і, як випливає з її назви, здатна використовувати свій інтелект лише для того, щоб сприймати навколишній світ і реагувати на нього. Реактивна машина не може зберігати пам'ять і, як наслідок, не може покладатися на минулий досвід для прийняття рішень у реальному часі.

Пряме сприйняття світу означає, що реактивні машини призначені для виконання лише обмеженої кількості спеціалізованих завдань. Однак навмисне звуження світогляду реактивної машини не є заходом для скорочення витрат, а натомість означає, що цей тип ШІ буде більш надійним і надійним — він буде щоразу однаково реагувати на ті самі стимули.

Відомим прикладом реактивної машини є Deep Blue [4], який був розроблений IBM у 1990-х роках як суперкомп'ютер для гри в шахи і переміг міжнародного гросмейстера Гарі Каспарова в грі. Deep Blue був здатний лише ідентифікувати

фігури на шаховій дошці та знати, як кожна рухається, виходячи з правил шахів, визнаючи поточне положення кожної фігури та визначаючи, який хід буде найбільш логічним у цей момент. Комп'ютер не переслідував майбутні потенційні ходи свого опонента або намагався поставити свої фігури в кращу позицію. Кожен поворот розглядався як власна реальність, відокремлена від будь-якого іншого руху, зробленого заздалегідь.

Іншим прикладом реактивної машини для ігор є AlphaGo від Google [5]. AlphaGo також не здатна оцінювати майбутні ходи, але покладається на свою власну нейронну мережу для оцінки розвитку теперішньої гри, надаючи їй перевагу над Deep Blue в більш складній грі. AlphaGo також переміг конкурентів світового класу в грі, перемігши чемпіона Го Лі Седола в 2016 році.

Хоча реактивний машинний штучний інтелект обмежений за обсягом і нелегко змінюється, він може досягти рівня складності та пропонує надійність, коли створюється для виконання повторюваних завдань.

1.2.2 Штучний інтелект з обмеженою пам'яттю

Штучний інтелект з обмеженою пам'яттю має здатність зберігати попередні дані та прогнози під час збору інформації та зважування потенційних рішень — по суті, заглядаючи в минуле, щоб знайти підказки щодо того, що може статися далі. Штучний інтелект з обмеженою пам'яттю є складнішим і надає більші можливості, ніж реактивні машини [6].

ШІ з обмеженою пам'яттю створюється, коли команда постійно навчає модель, як аналізувати та використовувати нові дані, або створюється середовище ШІ, щоб моделі можна було автоматично навчати та оновлювати. Використовуючи штучний інтелект з обмеженою пам'яттю в машинному навчанні, необхідно виконати шість кроків: необхідно створити дані про навчання, створити модель машинного навчання, модель повинна мати можливість робити прогнози, модель повинна мати можливість отримувати зворотний зв'язок від людини або навколишнього

середовища, що зворотний зв'язок має зберігатися як дані, і ці кроки необхідно повторити як цикл.

Існують три основні моделі машинного навчання, які використовують штучний інтелект з обмеженою пам’яттю:

1 - Навчання з підкріпленням, яке вчиться робити кращі прогнози шляхом повторних спроб і помилок.

2 - Довгострокова пам’ять (LSTM), яка використовує минулі дані, щоб допомогти передбачити наступний елемент у послідовності. LTSM розглядають найновішу інформацію як найважливішу під час прогнозування та відкидають дані з більш глибокого минулого, але все ще використовують її для формування висновків.

3 - Еволюційні генеративні змагальні мережі (E-GAN), які розвиваються з часом, розвиваючись, досліджуючи дещо змінені шляхи на основі попереднього досвіду з кожним новим рішенням. Ця модель постійно шукає кращого шляху і використовує моделювання та статистику, або випадковість, щоб передбачити результати протягом усього циклу еволюційної мутації.

1.2.3 Теорія розуму штучного інтелекту

Теорія розуму — це лише теоретична. Ми ще не досягли технологічних і наукових можливостей, необхідних для досягнення наступного рівня штучного інтелекту.

Концепція заснована на психологічній передумові розуміння того, що інші живі істоти мають думки та емоції, які впливають на поведінку людини. З точки зору машин з штучним інтелектом, це означало б, що ШІ міг би зрозуміти, що відчувають люди, тварини та інші машини, і приймати рішення шляхом саморефлексії та рішучості, а потім використовуватиме цю інформацію для власного прийняття рішень. По суті, машини мали б бути здатними сприймати й обробляти концепцію «розуму», коливання емоцій під час прийняття рішень та безліч інших

психологічних концепцій у реальному часі, створюючи двосторонні відносини між людьми та штучним інтелектом.

1.2.4 Самосвідомість штучного інтелекту

Як тільки теорія розуму може бути створена в штучному інтелекті, колись у майбутньому, останнім кроком стане самосвідомість ШІ. Цей вид штучного інтелекту володіє свідомістю на рівні людини і розуміє своє існування у світі, а також присутність та емоційний стан інших. Він міг би зрозуміти, що може знадобитися іншим, не лише на основі того, що вони їм повідомляють, а й того, як вони це передають.

Самосвідомість у штучному інтелекті покладається як на те, що люди-дослідники розуміють передумову свідомості, а потім навчаться відтворювати її, щоб її можна було вбудувати в машини.

1.3 Категорії штучного інтелекту

Крім типів штучного інтелекту існують ще й дві великі категорії:

1.3.1 Вузький штучний інтелект

Вузький штучний інтелект оточує нас всюди і є найуспішнішим на сьогоднішній день реалізацією штучного інтелекту. Зосереджений на виконанні конкретних завдань, вузький ШІ за останнє десятиліття пережив численні прориви, які мали значні переваги для суспільства та сприяли кращій життєздатності людей.

Кілька прикладів вузького ШІ включають:

- Пошук Google
- Програмне забезпечення для розпізнавання зображень
- Siri, Alexa та інші персональні помічники
- Автомобілі з автопілотом

Велика частина вузького ШІ заснована на проривах у машинному та глибокому навчанні. Розуміння різниці між штучним інтелектом, машинним навчанням і глибоким навчанням може бути заплутаним. Френк Чен надає гарний огляд того, як їх розрізняти, зазначаючи:

«Штучний інтелект — це набір алгоритмів та інтелекту, які намагаються імітувати людський інтелект. Машинне навчання — одна з них, а глибоке навчання — одна з тих методик машинного навчання» [7].

Простіше кажучи, машинне навчання подає комп'ютерні дані та використовує статистичні методи, щоб допомогти йому «навчитися» поступово покращувати завдання, не будучи спеціально запрограмованим для цього завдання, усуваючи потребу в мільйонах рядків написаного коду. Машинне навчання складається як із контролюваного навчання (з використанням позначених наборів даних), так і з навчання без нагляду (з використанням немаркованих наборів даних).

Глибоке навчання — це тип машинного навчання, який виконує вхідні дані через біологічно натхненну архітектуру нейронної мережі. Нейронні мережі містять ряд прихованих шарів, через які обробляються дані, що дозволяє машині «глибше» заглиблюватися в своє навчання, створюючи зв'язки та зважуючи вхідні дані для найкращих результатів.

1.3.2 Загальний штучний інтелект

Створення машини з інтелектом на рівні людини, яку можна застосувати до будь-якого завдання, є Святим Граалем для багатьох дослідників ШІ, але пошуки Загального штучного інтелекта були пов'язані з труднощами.

Пошук «універсального алгоритму навчання та дії в будь-якому середовищі» не новий, але час не полегшив труднощі створення машини з повним набором когнітивних здібностей.

Загальний штучний інтелект вже давно є музою наукової фантастики-антиутопії, в якій надрозумні роботи переповнюють людство, але експерти погоджуються, що це не те, про що нам потрібно турбуватися найближчим часом [8].

1.4 Опис та призначення ігрових застосунків

Комп'ютерна гра – це комп'ютерна система, що сприяє такого роду дозвіллям або соціальним взаємодіям, якими характеризується поняття «гра». В даний час, у ряді випадків, замість терміну «комп'ютерна гра» може використовуватися «відеогра», «гра», тобто дані терміни можуть вживатися як синоніми і бути взаємозамінними. Межі поняття зумовлені культурно і технічно: вони можуть бути чіткими ин і, у міру того, як ігрова мотивація користувача переходить до освітньої, професійної, творчо-експресивної, терапевтичної, соціально-політичної тощо.

У наші дні комп'ютерні ігри є предметом великої галузі інформаційних технологій, до того ж ставши до початку 2021-х років найбільшим напрямом індустрії розваг. Комп'ютерно-ігрова індустрія задіює апаратні потужності, технології розробки та маркетингу, онлайнову інфраструктуру повідомлення комп'ютерів, технології моделювання та симуляції фізичних середовищ [9].

1.4.1 Жанри ігрових застосунків

Кількість комп'ютерних ігор просто величезна, і часто вони класифікуються за характеристиками чи завданнями, а не за типом геймплея. Тому категорії ігор, чи жанри, можуть поділятися на поджанри, а одна гра цілком може належати до кількох жанрів.

Звичайно, це може трохи заплутувати, але якщо розібратися в ігровій механіці, то легко можна зрозуміти, як класифікують свої ігри розробники та видавці. Наприклад, якщо ви любитель миттєвого задоволення від ігор, то вам найкраще підійдуть екшени. А якщо подобається вирішувати головоломки або керувати ресурсами, то швидше за все, вам більше сподобаються стратегії в реальному часі чи рольові ігри (RPG). Але незалежно від того, що саме вам цікаво, обов'язково знайдеться ігровий жанр, який припаде вам до душі.

Ось список усієї різноманітності типів комп'ютерних ігор, а також їх підкатегорії. Але вони є взаємовиключними категоріями і можуть перетинатися.

1 - ACTION

Сутність екшенів відображає їхню назву – у перекладі з англійської вона означає «дію», і тут гравець знаходиться в самому центрі дії та керує ним. Здебільшого воно пов'язане з фізичними випробуваннями, які потрібно подолати. Зазвичай в екшенах дуже легко розпочати гру, тому вони досі залишаються найпопулярнішими відеоіграми.

- Платформери

Платформери отримали свою назву завдяки тому, що в процесі гри її головний персонаж взаємодіє з платформами – зазвичай бігає, стрибає чи падає. Різноманітних платформерів дуже багато, найвідомішим із них, мабуть, можна назвати відому серію Super Mario Bros.

- Шутери

Шутери дозволяють гравцям використовувати у своїх діях зброю, зазвичай з метою знищити ворогів чи протиборчих гравців.

Шутери діляться за ракурсом огляду гравця. У шутерах від першої особи (FPS – Frst-person shooter) вид із очей головного персонажа. У шутерах від третьої особи дія показана так, що гравець може бачити головного персонажа, зазвичай трохи зверху та ззаду.

Шутери з видом зверху, як і слідує за назвою, демонструють вигляд повністю зверху. І якщо в шутерах від третьої особи зазвичай використовується шкала здоров'я, яка зростає або зменшується в залежності від рівня здоров'я або стану персонажа, то в шутерах з видом зверху часто застосовується принцип із деякою кількістю «життя», коли напис Game Over з'являється після вичерпання їхнього запасу.

- Файтинги

У файтингах все зосереджено на боях, які найчастіше проходять у формі єдиноборств. Зазвичай у файтингах великий набір грабельних персонажів, кожен з яких спеціалізується на своїх унікальних

здібностях або бойовому стилі. У більшості традиційних файтингів гравець повинен боротися, просуваючись все вище і перемагаючи все більш сильніших супротивників.

- Beat-em up

Ігри жанру Beat-em up, або броулери, також зосереджені на бойовці, але замість бою з одним суперником гравці стикаються з цілими хвилями ворогів.

- Стелс-Екшен

Щоб впоратися з випробуваннями в стелс-екшенах, знадобиться хитрість і точність. Хоча бій та інші дії також можуть допомогти досягти мети, але зазвичай у стелс-іграх вітається скритність.

- Виживання

Такий поджанр екшенів, як survival-action, останні кілька років завоював значну популярність. Ціль у цих іграх збирати ресурси для створення інструментів, зброї та укриттів з метою вижити максимально довго.

2 - ACTION-ADVENTURE

Найчастіше в action-adventure поєднується дві ігрові механіки – квести чи випробування протягом усієї гри, які потрібно проходити та долати, використовуючи зібрані предмети чи інструменти, а також елемент екшену, де застосовується різна зброя та спорядження.

Ігри жанру action-adventure більше зосереджені на дослідженні, вирішенні головоломок та пошуку видобутку, а проста бойовка є скоріше допоміжною по відношенню до геймплею загалом.

- Survival-Horror

В іграх жанру survival-horror для створення похмурого та суворого настрою часто використовується серйозна тематика. У багатьох подібних іграх зображується насильство та кров, тому вони призначені лише для дорослої аудиторії. Зазвичай, такі ігри можуть похвалитися

захоплюючим геймплеєм, посиленим ключовим елементом механіки: обмеженими ресурсами, наприклад, боєприпасами або зброєю.

- Візуальна новела

У більшості візуальних новел, шалено популярних насамперед у Японії, гравець має вибрати особливості чи параметри персонажа, які впливають на геймплей. Зазвичай у цих іграх безліч кінцівок, що визначаються діями гравця у певних точках сюжету. Популярними темами багатьох візуальних новел є симуляції побачень чи судових процесів.

3 - РОЛЬОВІ ІГРИ

Рольові ігри, або RPG – мабуть, другий за популярністю жанр. У ньому часто зустрічаються середньовічні чи фентезійні сеттинги, що переважно пов'язані з походженням жанру. Але рольові ігри фантастичної тематики також не обділені увагою хардкорних ролевиків і зробили в жанрі свій унікальний внесок.

Крім того, на жанр вплинули і культурні відмінності – зокрема багато геймерів ділять RPG на WRPG (західні) і JRPG (японські). Нарешті, однією із знакових фішок жанру є вибір гравця, що впливає на результат гри, тобто у багатьох RPG є кілька альтернативних кінцівок.

- Action-RPG

Action-RPG запозичують елементи жанрів action та action-adventure. Визначальною характеристикою action-RPG можна назвати бойову систему в реальному часі, де часто залежить від швидкості і реакції гравця, а не тільки від високих параметрів його персонажа типу спритності і харизми.

- MMORPG

MMORPG, або масові розраховані на багато користувачів мережеві рольові ігри, де сотні гравців активно взаємодіють один з одним в єдиному світі і зазвичай виконують одні і ті ж або схожі завдання.

- RPG з відкритим світом

Рольові ігри з відкритим світом, або «пісочниці», дозволяють гравцям вільно подорожувати своїм ігровим середовищем у пошуках пригод. Це одні з найбільш атмосферних та захоплюючих ігор, тому що величезна кількість ігрових персонажів та ситуацій, необхідних для створення побічних квестів та додаткових сюжетних ліній, дозволяє розробникам створювати справді живі віртуальні світи.

4 - СИМУЛЯТОРИ

Представники жанру симуляторів мають одну спільну рису – всі вони створені для симуляції реальних чи вигаданих систем, ситуацій чи подій.

- Симулятори будівництва та менеджменту

Найпопулярнішою грою цього жанру є SimCity. У ній гравець має будувати та розвивати місто, а також керувати ним, зокрема, планувати вулиці, розподіляти простори та стягувати податки з городян. В інших іграх охоплення може бути обмежене більшими або меншими рамками, наприклад, кафе, хмарочосом, парком розваг або цілою планетою.

- Симулятор життя

Такі симулятори можуть дозволяти гравцям управляти генетикою персонажа чи середовищем їхнього життя. У деяких випадках можна контролювати реакцію персонажа на певні ситуації.

- Симулятор техніки

Симулятори техніки намагаються відтворити відчуття польоту літаком або гелікоптером, водіння гоночної машини і навіть управління трактором на фермі. Бій у цих іграх теж зустрічається (найчастіше в авіаційних та танкових симуляторах). Також сюди відносяться симулятори поїздів та космічні симулятори.

5 - СТРАТЕГІЇ

Геймплей цього жанру сягає своїм корінням в традиційні настільні стратегічні ігри. Часто в цих іграх вам доступний цілий світ і всі його ресурси, а щоб впоратися з випробуваннями, гравцеві потрібно ретельно продумувати стратегію та тактику. Згодом значна частина цього жанру

перейшла від покрокових систем до реального часу, слідуючи бажанням гравців.

- 4-х стратегії

Завдяки масштабу закладених цілей у багатьох ігор цього жанру історичні сеттинги, а дія охоплює цілі епохи життя цивілізації (людської чи інопланетної).

- Стратугії у реальному часі

У стратегіях у реальному часі від гравцю потрібно збирати та розподіляти ресурси, будувати бази, а також розвивати економіку та військову силу.

6 - МОВА (моба)

Цей піджанр поєднує риси екшенів, RPG та стратегій у реальному часі. Тут зазвичай не потрібно нічого будувати та виробляти. Гравець управляє одним персонажем в одній із двох команд, кожна з яких намагається знищити базу супротивника. Часто їм допомагають керовані комп'ютером юніти, що атакують у заданому напрямку.

- Tower Defence

В обороні веж гравці повинні відображати хвилі ворогів, що керуються комп'ютером. Конкретні здібності веж і ворогів у різних іграх можуть відрізнятися, але зазвичай є кілька веж з різними здібностями, наприклад, уповільненням або отруєнням ворогів. Знищивши певну кількість ворогів, гравець отримує ресурси на купівлю нових веж або апгрейд існуючих, наприклад, збільшення втрат або дальності.

- Покрокова тактика

Ігри цього піджанру багато що запозичують у реальної військової тактики і зазвичай дозволяють керувати невеликим підрозділом з особливою увагою до озброєння, спорядження та характеристик кожного окремого бійця.

7 - СПОРТИВНІ ІГРИ

Спортивні ігри, як і слідує з назви, симулюють різні види спорту, наприклад, гольф, баскетбол, звичайний та американський футбол. Зустрічаються також олімпійські види спорту типу лижних гонок, і навіть барний спорт на кшталт більярду або дартс. Зазвичай, суперники управляються комп'ютером, хоча можна змагатися і з іншими гравцями.

- Гонки

Тут гравці беруть участь у гоночних заїздах проти суперників або на якийсь час. Змагатись можна як з комп'ютером, так і з людьми у режимі мультиплеер.

- Інші змагання

До цієї категорії можна віднести вигадані види спорту та самі по собі змагальні ігри, у тому числі кіберспортивні дисципліни.

Вибір відеоігор воістину безмежний, і існує не тільки безліч найрізноманітніших жанрів, а й величезна кількість успішних піджанрів у межах кожного з них. Технології постійно розвиваються, а інтерактивне середовище дуже швидко поповнюється чимось новим, з більш розвиненою системою штучного інтелекту. Це не повний список всіх жанрів та піджанрів, а лише основні, які зустрічаються най частіше.

1.5 Що таке штучний інтелект в іграх

Штучний інтелект в іграх дає гравцеві більшу адаптивність та чутливість при проходженні відеоігор, що забезпечуються завдяки тому, що неігрові персонажі поводяться творчо, ніби ними керує гравець-людина.

Штучний інтелект в іграх — від програмного забезпечення, яке керувало привидом у грі Пак-Мен, до алгоритмів створення всесвіту у Elite: Dangerous для дослідження космосу.

Рис. 1.1 - гра Elite: Dangerous (зверху) та Пак-Мен (знизу)

«Гра розуму» [10] в першу чергу була розроблена, щоб оцінити психологічний стан душі молодих новобранців. По суті, ця гра представляє своїм гравцям набір неможливих ситуацій, які перевіряють їх розумову стійкість перед обличчям неминучої поразки. Хоча гра здається нескінченно процедурною, вона також створює середовища та ситуації на льоту, дозволяючи гравцям виконувати будь-які дії у віртуальному світі так само, як і в реальності. Крім того, вона також реагує на психологічний стан своїх гравців, щоб адаптуватися та реагувати на поведінку людини під час гри.

Гра розуму, по суті, є відправною точкою для майбутнього відеоігор і штучного інтелекту. Сьогодні максимальні вдосконалення технологій, зокрема консолі, хмарний гемінг, надпотужні відеокарти, віртуальна реальність, гарнітури та алгоритми візуалізації, дозволили штучному інтелекту створювати вражаючі середовища, а віртуальні персонажі демонструють людську поведінку та інтелект. Сьогодні розробники та дизайнери ігор почали вирішувати фундаментальні теми ШІ в іграх з його останніми досягненнями в цій галузі. Вони почали переходити від

експериментальних лабораторій до ігрових продуктів і інструментів розробки, щоб досягти більшої реалістичності в штучних середовищах.

1.5.1 Штучний інтелект в ігровому досвіді

Донедавна той вид штучного інтелекту, що самонавчається, а саме, підмножина глибокого навчання революції машинного навчання, що призвело до прогресу в автомобілях з автопілотом, комп'ютерному баченні та обробці природної мови, не повністю ступив у комерційну гру. Проте, схоже, у майбутньому настане момент, коли розробники отримають доступ до цих інструментів для створення захоплюючих та інтелектуальних ігор. Можливо, це може призвести до створення інструментів розробки, які автоматизують основи складних ігор, які здатні змінюватися та реагувати на відгуки гравців, а також внутрішньоігрових персонажів, які будуть розвиватися в міру того, як з ними витрачатиметься більше часу.

З перших днів існування цього середовища розробники ігор програмували програмне забезпечення таким чином, що воно не тільки вдає, ніби це людина, але й допомагає створювати віртуальні світи без людського втручання з нуля. Однак сьогодні навіть ігровий дизайн, що розширює межі, не зовсім пов'язаний з сучасним ШІ. Це скоріше кружляє навколо створення набору складних систем, які призводять до непередбачуваного ігрового процесу. Наприклад, гіперреалістична західна гра Red Dead Redemption 2 від Rockstar [11] дозволяє гравцям взаємодіяти з неігровими персонажами безліччю способів, викликаючи різні реакції залежно від усього, включаючи дії, як-от одягання капелюха або навіть плями крові на ньому. .

Цей тип штучного інтелекту, який спрямований на створення відчуття реалізму замість руйнівних результатів, — це той тип ШІ, якого намагаються досягти більшість розробників — незалежно від інтелекту частин. Знову ж таки, метою, історично, однак, було не досягнення безпрецедентного рівня людського інтелекту, а створення досвіду, який залучає та стимулює гравців у спосіб, який наслідує реальність. Проте, за словами Майка Кука (наукового співробітника Королівської академії інженерії Лондонського університету королеви Марії) [12], це лише 50

відсотків штучного інтелекту. «Інші 50 відсотків ШІ – це психологія. Це те, як люди реагують на машини та технології та як вони їх сприймають. І насправді багато ігрового штучного інтелекту в кінцевому підсумку заглибилися в це». Крім того, більші студії, безумовно, відкриють рамки, коли справа доходить до створення середовища з відкритим світом, а створення систем ближче до досягнення складності реальності.

1.6 Методи та технології у штучному інтелекті

1.6.1 Прийняття рішень у системі штучного інтелекту

Основна концепція ШІ – прийняття рішень. Щоб виконати ці рішення, інтелектуальна система повинна мати можливість впливати на об'єкти, які використовують систему ШІ. Можно організувати це виконання у стратегії «натиснення ШІ» або «витягнення об'єктів».

Системи зі штучним інтелектом мають тенденцію ізолювати систему ШІ як окремий елемент архітектури гри. Така стратегія часто набуває форми окремого потоку або потоків, в яких ШІ витрачає свій час на підрахунок найкращих варіантів з урахуванням варіантів гри. Коли ШІ приймає рішення, це рішення передається залученим організаціям. Цей підхід найкраще працює в стратегічних іграх у реальному часі, де ШІ займається загальною картиною.

Системи витягування об'єктів найкраще працюють для ігор із простими об'єктами. У цих іграх об'єкти звертаються до системи ШІ, коли об'єкт «думає» або оновлюється. Цей підхід дуже добре працює в системах з великою кількістю об'єктів, яким не потрібно думати дуже часто, наприклад, шутери. Ця система також може скористатися багатопоточними методами, але вона вимагає додаткового планування [13].

1.6.2 Базове сприйняття штучного інтелекту

Щоб ШІ міг приймати важливі рішення, йому потрібен певний спосіб сприйняття навколишнього середовища. У простіших системах це сприйняття може бути простою перевіркою позиції об'єкта гравця. Оскільки системи стають все більш вимогливими, об'єкти повинні визначити ключові особливості ігрового світу, такі як життєздатні шляхи для проходження, місцевість, що забезпечує прикриття, та зони конфлікту.

Завдання для дизайнерів і розробників полягає в тому, щоб знайти спосіб визначити ключові функції, важливі для системи розвідки. Наприклад, обкладинка може бути заздалегідь визначена дизайнерами рівнів або може бути попередньо обчислена під час завантаження або компіляції карти. Деякі елементи необхідно оцінювати на льоту, наприклад, карти конфліктів і неминучі загрози.

1.6.3 Системи поведінки штучного інтелекту

Найпростіша форма, яку може прийняти інтелектуальна система, це система, заснована на правилах. Ця система розширює термін «штучний інтелект». Набір попередньо встановленої поведінки використовується для визначення поведінки ігрових об'єктів. За допомогою різноманітних дій загальним результатом може бути система поведінки, яка не є очевидною, хоча фактичного інтелекту задіяно дуже мало.

Класичним застосуванням цієї системи є гра Пак-Мен. Чотири привиди переслідували гравця. Кожен привид дотримувався простого набору правил. Один привид завжди мав повертати ліворуч, інший — праворуч, один повертався у випадковому напрямку, а останній повертався до гравця. Окремо привидів було б легко розгадати, і гравець міг би легко їх уникнути. Як група, їхній рух виглядає як складна, скоординована пошукова група, яка полює на гравця. Насправді, єдиний, хто навіть перевіряє позицію гравця, - це останній.

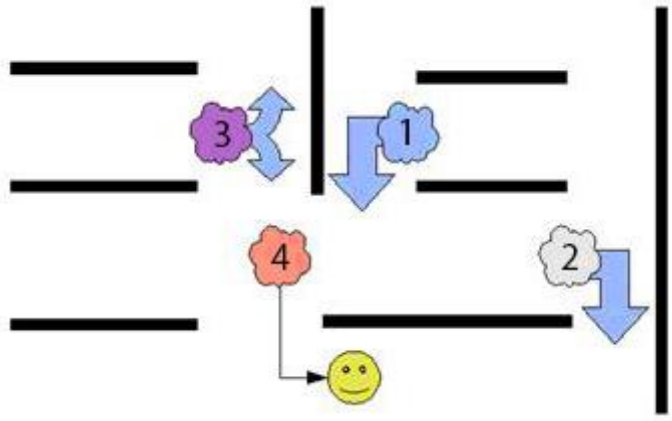

Рис. 1.2- Візуальне зображення набору правил, що керують привидами у грі Пак-Мен.

Як виходить з цього прикладу, правила не потрібно жорстко кодувати: вони можуть ґрунтуватися на сприйнятих станах або на параметрах об'єктів, які можна редагувати. Такі змінні, як агресія, сміливість, діапазон зору та швидкість мислення, можуть призвести до більш різноманітної поведінки об'єктів, навіть у рамках системи, заснованої на правилах. Системи, засновані на правилах, є найпростішою структурою для ШІ. Більш складні інтелектуальні системи спираються на низку умовних правил і керуються ними. У тактичних іграх правила визначають, яку тактику використовувати. У стратегічних іграх правила регулюють порядок складання та способи реагування на конфлікти. Системи, засновані на правилах, є основою ШІ.

1.6.4 Кінцевий стан у штучного інтелекту

Кінцевий стан (машина з кінцевим числом станів) - це спосіб концептуалізації та реалізації об'єкта, яка має різні стани протягом свого життя. "Стан" може представляти фізичні умови, в яких перебуває суб'єкт, або він може представляти емоційні стани, які суб'єкт може проявляти. У цьому прикладі емоційні стани - це не що інше, як емоційні стани справжнього ШІ, а заздалегідь визначені моделі поведінки, які вписуються в контекст гри.

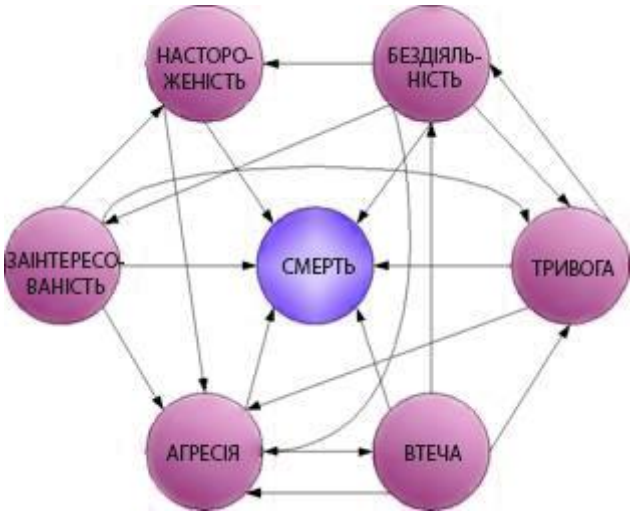

Рис. 1.3 - Розташування станів, де стрілки відображають можливі зміни стану

1.6.5 Прогнозування штучного інтелекту свого наступного ходу

Здатність ефективно передбачати наступний хід суперника має вирішальне значення в адаптивній системі. Можуть бути використані різні методи, такі як розпізнавання минулих моделей або випадкове припущення, щоб визначити наступну дію.

Одним з основних методів адаптації є відстеження минулих рішень та оцінка їх успіху. Система штучного інтелекту веде облік рішень, які гравець зробив у минулому. Минулі рішення мають бути певним чином оцінені. (наприклад, у бойових іграх показником успіху може бути отримана перевага або втрачене здоров'я чи перевага в часі). Додаткова інформація про ситуацію може бути зібрана, щоб надати рішенням певний контекст, наприклад відносне здоров'я, попередні дії та положення на рівні, адже люди грають по-різному, інколи спиною до стіни.

Це все можна оцінити, щоб визначити успіх попередніх дій і чи потрібна зміна тактики. Поки не буде створено список минулих дій, для керування діями суб’єкта можна використовувати загальні тактики або випадкові дії. Ця система може бути пов'язана з системами, заснованими на правилах, і різними станами.

У тактичній грі минула історія може визначити найкращу тактику для використання проти команди гравця, таку як оборона, наступ або деякі збалансовані

засоби гри. У стратегічній грі оптимальний склад підрозділів у армії можна виявити на основі кожного гравця. В іграх, де ШІ керує підтримуючими гравцями персонажами, адаптивний ШІ може краще доповнити природний стиль гравця, дізнавшись, як гравець діє [14].

1.6.6 Як штучнмй інтелект сприймає навколишній світ

Якщо ваші агенти прийматимуть довільні рішення, це добре для деяких ігор, але що, якщо вам потрібно більше? Якщо ваш агент збирається приймати гідні рішення, він повинен знати, що відбувається навколо нього. У робототехнічному застосуванні штучного інтелекту проводяться численні дослідження комп'ютерного зору, що дає роботам можливість сприймати навколишній світ у справжньому, стерео тривимірному баченні, як це роблять люди. Для наших цілей такий рівень витонченості є абсолютним зайвим.

Віртуальні світи, в яких відбувається більшість ігор, мають величезну перевагу перед реальними роботами ШІ. Все в нашому світі відома величина: десь у грі є список з усім, що в ньому існує. можна просто здійснити пошук у цьому списку за будь-якими критеріями, а потім одразу отримати інформацію, яку ваш агент може використовувати для прийняття більш значущих рішень.

1.6.6.1 Візуальне сприйняття

Підробка зору є основним рівнем надання агенту здатності сприйняття. Можна зробити це, шукаючи у своєму списку об'єктів будь-що в межах заданого діапазону. можливо отримати перше, що цікавить вашого агента, або отримати список речей у діапазоні, щоб агент міг прийняти оптимальне рішення щодо свого оточення.

Ця установка добре працює для простих ігор, але якщо є складніший стиль гри, наприклад, шпигунство або тактичний шутер від першої особи (FPS), агентам потрібно буде бути дещо виборчішим у тому, що вони «бачать». Якщо не хочеться,

щоб у агентів були очі на потилиці, можна вибрати список потенційних можливостей будь-чого за межами поля зору об'єкта. Це можливо зробити швидко, використовуючи трохи математики:

1. Обчисліть вектор між агентом і сутністю, про яку йдеться, віднімаючи позицію цілі від позиції агента.

2. Обчисліть кут між цим вектором і напрямком, у якому дивиться ваш агент.

3. Якщо абсолютне значення кута більше, ніж попередньо встановлений кут огляду агента, агент не бачить об’єкт.

У більш складних іграх може знадобитися врахувати, що гравець або інші об'єкти приховані якимось прикриттям. Для цього типу ігор може знадобитися виконати трасування променів, щоб побачити, чи не заблоковано щось потенційну ціль. Трасування променя — це математичний спосіб перевірити, чи перетинає промінь що-небудь, починаючи з однієї точки і рухаючись у заданому напрямку.

Попередній метод повідомляє, чи щось затьмарило центр цілі, але цього може бути недостатньо, щоб стримати вашого агента. Можливо, центр агента затьмарений, але його голова зручно висовується над кришкою. Використання кількох трас променів до конкретних точок на цілі може допомогти визначити не тільки, чи можна вразити ціль, але й те, де можна вразити ціль.

1.6.6.2 Звукове сприйняття

На перший погляд може здатися, що звук нічим не відрізняється від зору. Якщо є можливість бачити об'єкт, то, безперечно, можна його почути. Це правда, що якщо агент помітив об'єкт, агент може активно виявляти все, що робить об'єкт, поки він не перестане бути в полі зору. Однак додавання додаткового рівня слуху агентам може допомогти зробити зір більш ефективним. Відстеження шуму, який створюють сутності як рівня сприйняття, є ключовим для будь-якої гри у жанрі стелс, де потрібно залишатися непоміченим.

Так само, як і в звичному режимі зору, потрібно буде отримати список довколишніх об’єктів для перевірки. Можна зробити це знову за допомогою простої перевірки відстані, але те, що робить, щоб вилучити цей список, відрізняється.

Кожна дія, яку може виконати об'єкт, повинен мати певні рівні звуку, пов’язані з ним. Також ще можна попередньо налаштувати рівні звуку, щоб оптимізувати ігровий баланс, або базувати їх на фактичній енергії звукових ефектів, які відтворюються для дії, що гарно для реалістичності, але непотрібно. Якщо звук, який генерується, знаходиться в межах порогового значення, агент відчує цей об'єкт.

Якщо потрібно врахувати перешкоди, можна ще раз зібрати цей список, виконавши трасування променів з потрібним середовищем, щоб побачити, чи не буде щось заважати звуку. Оскільки дуже мало матеріалів є повністю звуконепроникними, потрібно бути більш креативними у тому, як вилучати об’єкти зі свого списку.

1.6.6.3 Інші органи чуття

Основну функціональність, необхідну для надання агентам зору та слуху, можна легко застосувати для імітації інших органів чуття. Ось список інших потенційних почуттів, які ще можна додати до гри, якщо цього вимагає дизайн:

Запах. Додати нюх у гру відносно легко: необхідно дати кожному об'єкту в грі окремий номер і силу запаху. Сила запаху визначає два фактори: радіус запаху і розмір запахового сліду, що залишився. Активні гравці часто відстежують свої останні кілька позицій з ряду причин. Однією з причин може бути допомога особам із запахом. Коли об'єкт гравця оновлює слід, сила запаху зменшується, оскільки слід стає «холодним». Коли агент із запахом оновлюється, йому потрібно перевіряти запахи так само, як він перевіряє звук: радіус і перевірку стін. Що стосується запаху, то фактор успіху базується на силі запаху та нюху агента, які перевіряються на основі сутності та сліду сутності.

Радар. Деякі ігри надають гравцям індивідуальний радар, що робить виявлення ще простіше. Все, що потрібно, це проста перевірка радіусу. Потім ШІ може перевірити результати для інтересу. Для командних ігор сам радар може стати цікавішим. Щоб створити командний ШІ, кожній команді потрібен радарний список об’єктів, які були знайдені радаром. Потім кожен член команди може виконати перевірку радіусу лише зі списком відомих об’єктів, щоб визначити, чи повинна команда реагувати. Команда може додати до списку, використовуючи радіолокаційне обладнання все, що «бачить» кожен член команди. Така поведінка допомагає об’єктам працювати як єдине ціле, оскільки кожен сповіщає інших про те, що бачить.

Дотик. Це відчуття є свого роду «дай», оскільки система зіткнень ігрового движка охоплює його автоматично. Просто потрібно переконатися, що розумні агенти реагують на пошкодження та події зіткнення.

Смак. Це, ймовірно, буде властивістю нюху, але вимагатиме від агентів активно «пробувати на смак» речі, які вони знаходять.

Вміти відчувати навколишній світ — це все добре, але що саме мають відчувати агенти? Для цього потрібно вказати та мати можливість ідентифікувати речі, які можна спостерігати за допомогою налаштувань агентів. Коли розпізнається, що саме у полі зору, агенти можуть реагувати на це відповідно до правил, які керують сутністю [15].

1.7 Сприйняття штучного інтелекту тимчасових об'єктів

Тимчасові об’єкти, які іноді називають частинками, наклейками або спецефектами, є візуальними ефектами в ігровому світі. Вони подібні до об'єктів тим, що одна загальна структура класів визначає будь-який потенційний тимчасовий об'єкт, але він відрізняється від звичайного об'єктів тим, що вони не думають, не реагують і не взаємодіють з об'єктами ігрового світу чи один з одним. Їхня єдина мета — виглядати красиво, на деякий час надавши світу додаткові деталі, а потім померти.

Тимчасові об'єкти використовуються для таких речей, як сліди від куль, дим, іскри, бризки крові і навіть сліди ніг.

Природа тимчасових об'єктів передбачає дуже малу обробку та відсутність виявлення зіткнень, за винятком дуже простих зіткнень. Проблема в тому, що деякі тимчасові об'єкти дають гравцям наочну підказку про те, що могло статися, наприклад, діри від куль і сліди опіків вказують на нещодавню битву, сліди на снігу можуть привести до потенційної цілі, то чому розумні агенти не можуть використовувати це теж?

Є два способи вирішити цю проблему: можна або покращити свою систему тимчасового об’єкта, щоб дозволити трасування променів, що порушить весь зміст системи тимчасового об’єкта, або є можливість скинути порожній об’єкт в загальне околиці тимчасового об’єкта. Ця порожня сутність не мала б здатності мислити та не мати пов’язаної з нею графіки, але агенти могли б її виявити, а тимчасова сутність мала б пов’язану інформацію, щоб надати «інтелект» агенту. Отже, коли на підлогу відтворюється лужу із кров’ю, туди також ще можно кинути невидимий об'єкт, щоб повідомити агентам, що щось відбувається.

У багатьох іграх, заснованих на стрілянині, було б добре, якби агенти були достатньо розумними, щоб ховатися за прикриттям, коли воно доступне, замість того, щоб просто стояти на відкритому повітрі, піддаючись обстрілу. Ця проблема є дещо більш спеціалізованою, ніж інші проблеми, які розглядалися раніше.

Ця дилема насправді складається з двох проблем: як визначити укриття з геометрії навколишнього світу, і як визначити укриття з об'єктів навколишнього світу. Щоб визначити, чи може об'єкт блокувати атаки, можна виконати просту перевірку, порівнявши розмір обмежувальної рамки потенційного укриття. Потім потрібно перевірити, чи може об'єкт вміститися за укриттям. Для цього необхідно створити промінь із відмінностей у позиціях стрільця та укрикриття. Цей промінь необхідно використовувати, щоб визначити, чи вільне місце за укриттям, а потім позначити це місце як наступну ціль агента [16].

1.8 Навігація штучного інтелекту у просторі

Є алгоритм, який умовно називається зіткнутися та повернути. Це однин із основних способів створення руху для об’єкта. Ось як це працює:

1. Рухайтеся в напрямку вашої мети.

2. Якщо ви вдаритеся об стіну, поверніть у напрямку, який наближає вас до цілі. Якщо, очевидно, немає кращого вибору, виберіть один навмання.

Цей підхід досить добре працює для простих ігор. Багато ігор використовували цей метод, щоб керувати тим, як монстри полюють на гравця. Зіткнення і поворот призводить до того, що об’єкт застрягає за увігнутими стінами або кутами, коли вони полюють на гравця, тому для ігор із зомбі або без цієї функції землі цей метод ідеальний.

Однак, якщо гра вимагає від своїх агентів трохи більшої витонченості, можна надати своїм агентам трохи пам’яті. Якщо агенти можуть відстежувати, де вони вже були, вони можуть почати приймати більш значущі рішення про те, як повернутись. Коли всі ходи будуть вичерпані, агенти можуть повернутися назад і зробити інший вибір. Таким чином, агент буде здійснювати систематичний пошук шляху до цілі. Ось як це працює:

- Рухайтеся до цілі.

- Зробіть вибір, коли вам подарують виделку.

- Коли будуть виявлені тупики, поверніться до останнього вибору і зробіть інший вибір.

- Якщо вивчені всі можливі шляхи, здайтеся.

Перевага цього методу полягає в тому, що він легкий у категорії обробки, що означає, що можна підтримувати велику кількість цих хлопців, не сповільнюючи гру. Цей метод також може отримати оптимальну користь від багатопотокової роботи. Недоліком є величезна втрата простору, оскільки кожен агент потенційно може відстежувати цілу карту можливих шляхів.

На щастя, є спосіб уникнути цих витрат, якщо агенти відстежують шляхи в спільній пам’яті. Тоді може виникнути проблема з конфліктами потоків, тому

потрібно зберігати шляхи до об’єктів в окремому модулі, до якого всі агенти зможуть надсилати запити під час переміщення та надсилати оновлення, коли вони знаходять нові шляхи. Модуль карти шляху може потім проаналізувати результати, щоб уникнути конфлікту.

1.8.1 Пошук шляху

Зіткнення та створені поворотами карти шляхів – це чудовий спосіб адаптуватися до змінних карт. У стратегічних іграх гравці не можуть чекати, поки їхні підрозділи знайдуть своє орієнтування. Крім того, ці карти шляхів можуть стати гігантськими, і пошук по них правильного шляху стане важким завданням.

Пошук шляху є, по суті, вирішеною проблемою в розробці ігор. Ігри вже достатньо старі, тому способи обробляли велику кількість ігрових об’єктів, які могли знайти свій шлях на великих, складних картах вже знайдені.

Основним алгоритмом для пошуку шляху є 'A*', який можна використовувати для виявлення оптимального шляху з будь-яких двох точок на графіку, у даному випадку карта. Простий онлайн-пошук виявить чистий алгоритм, який використовує такі описові терміни, як F, G і H.

По-перше, потрібно налаштувати два списки: список вузлів, які не були перевірені і список вузлів, які були перевірені. Кожен список складається з вузла позиції, передбачуваної відстані до цілі та посилання на його батьківський вузол. Це вузол, який помістив його в список і спочатку списки будуть порожні.

Далі треба додати свою початкову позицію до списку не перевірених і нічого в якості батьківського джерела. Потім треба ввести алгоритм:

- Необхідно вибрати зі списку найкрасивіший вузол.
- Якщо цей вузол є метою, все готово.
- Якщо цей вузол не є метою, треба додати його до списку Перевірені.
- Для кожного вузла, суміжного з цим вузлом:
  - Якщо вузол недоступний для проходження, потрібно проігнорувати його.

- Якщо вузол уже є в списку, з відміченим чи невизначеним, теж потрібно проігнорувати його.

- Все інше, додати до списку Не перевірених, встановивши цей вузол як батьківський і оцінивши відстань до цілі.

Коли об’єкт досягає цільової плитки, шлях можна побудувати шляхом відстеження батьків назад до вузла, який не має батьківського вузла. Це забезпечує оптимальний шлях, за яким може йти об’єкт. Оскільки цей процес має відбуватися лише тоді, коли агент або отримує наказ, або самостійно вирішує переміститися, багатопотоковість дійсно може отримати користь. Агент може надіслати запит до потоку шляху отримати назад шлях, коли він був виявлений, не впливаючи на продуктивність ШІ. Здебільшого система може швидко отримати результати; але у випадку великої кількості запитів шляху агент може або почекати, або просто почати рухатися в правильному напрямку, чи можливо, він може використовувати метод Зіткнення і повороту. У надзвичайно великих картах систему можна розбити на регіони, і всі можливі шляхи між регіонами (або точками шляху) можна попередньо обчислити. У цьому випадку шукач шляху може просто знайти найкращий шлях і негайно відобразити потрібні результати. Потік карти шляхів може просто стежити за змінами на карті, наприклад, коли гравець робить перешкоду, а потім за потреби ще раз запускає перевірку шляху. Оскільки алгоритм знаходиться у власному потоці, він може адаптуватися, не впливаючи на продуктивність решти гри.

У підсистемі пошуку шляху багатопоточність також може підвищити продуктивність системи. Це добре використовується в будь-якій стратегії в реальному часі або в системі з великою кількістю сутностей, кожна з яких шукає потенційно унікальний шлях. Декілька шляхів можна знайти одночасно в різних потоках. Звичайно, системі потрібно буде відстежувати, які шляхи виявляються. Один і той самий шлях не потрібно відкривати більше одного разу [17].

1.9 Проблемі зі штучним інтелектом в іграх

Існує багато варіації та сфер для застосування штучного інтелекту. Нажаль при розробці і експлуатації ШІ все ж доводиться зустрічати якісь помилки чи неочікувану реакцію штучного інтелекту. Зараз є можливість коротко описати деякі з основних проблем, які виникають при розробці штучного інтелекту для ігор. Цей список не є вичерпним, але має на меті дати відтінок проблем, з якими можна зіткнутися при взаємодії з ШІ.

1. Складні простори прийняття рішень: більшість сучасних ігор включають складні стратегічні (стратегічні ігри в реальному часі) або правдоподібні поведінки (інтерактивні драми). Обидва види поведінки мають властивість мати величезний простір для прийняття рішень, і, отже, традиційні методи штучного інтелекту на основі пошуку неможливо застосувати. Для роботи з такими складними іграми потрібні методи навчання або уявлення вищого рівня. Традиційно комп'ютерні ігри використовують створені вручну стратегії, закодовані розробниками ігор, але вони, як правило, повторюються, і гравці легко знаходять діри в цій системі та використовують їх для своєї вигоди.

2. Розробка знань: навіть якщо припустити, що стратегії або поведінка створені вручну, створення цих наборів поведінки в грі вимагає величезних інженерних зусиль. Розробники ігор повинні закодувати всі знання, які вони мають (або для досягнення стратегічної поведінки, або правдоподібної поведінки людини) певною мовою поведінки.

3. Підтримка авторства: поведінка, створена вручну, — це, зрештою, програмний код на складній мові програмування, схильний до людських помилок. Помилки поведінки можуть бути у вигляді програмних «багів» або недосягнення бажаного результату. Потрібні інструменти, щоб підтримувати авторів історій, які, як правило, не є експертами зі штучного інтелекту, у створенні поведінки на мові комп'ютерного програмування.

4. Непередбачувані ситуації: неможливо передбачити всі можливі ситуації та стратегії гравця, які можуть виникнути під час гри. Це ускладнює створення

правдоподібної поведінки, яка б належним чином реагувала на ці непередбачені обставини та дії гравця.

5. Спеціальна адаптація користувача: різні стратегії боротьби з різними гравцями (у випадку стратегічних ігор у реальному часі) або різні стилі розповіді (у випадку інтерактивних драм), різні типи розвитку історії, різні типи персонажів поведінки та взаємодії, або різні освітні проблеми. Оскільки розробники ігор починають включати можливості моделювання користувачів, стратегія та поведінка ШІ повинні, у свою чергу, адаптуватися на основі моделі користувача.

6. Можливість перегравання та мінливість: гравцеві може набриднути знову і знову бачити ті самі стратегії та поведінку. Хоча простої мінливості можна досягти за допомогою випадкового вибору поведінки або стратегії з великого сховища, це збільшує тягар авторства. Крім того, випадковий вибір викликає питання справжньої цікавості.

7. Риторичні цілі: можливо, навіть імовірно, що створені людиною поведінки або стратегії не досягають належним чином цілей гри, особливо в реалістичних, розширених іграх або програмах. Ці цілі можуть варіюватися від розваг до освіти, навчання тощо. Таким чином, гра має усвідомити, що цілі не досягаються на основі використання, і відповідно адаптуватися. Наприклад, певний користувач може нудьгувати або не засвоює задуманий урок [18].

Висновок, який можна зробити з попереднього списку, полягає в тому, що не тільки ігри можуть отримати користь від кращих методів ШІ, але ШІ також може отримати вигоду від проблем, які створюють ігри.

1.10 Постановка задачі

Метою цієї роботи є дослідження та порівняння алгоритмів штучного інтелекту в ігрових застосунках, та виявлення найефективніших способів навчання. Для цього буде взята гра для мобільних телефонів на системі Android у жанрі Аркади, вона працює по принципу гри Зиг-Заг. Суть цієї гри дуже проста, необхідно проходити різноманітні рівні, де з кожним новим рівнем збільшується складність.

Гравцеві необхідно дійти до кінця дороги, керуючи м'ячиком, а для керування використовувати натискання на екран. Кожне натискання змінює траєкторію руху м'ячика і він рухається або вліво, або вправо по дорожці. Ще один важливий момент гри, це збирання невеликих геометричних фігур, які розставлені в деяких місцях на дорозі по всьому маршруту м'ячика. Збирання цих фігур дуже важливе, тому що кожна фігура дає додаткові бали, які позитивно впливають на кінцевий результат загальної кількості балів.

Відповідно з метою необхідно розробити схему для порівняння та оцінювання систем штучного інтелекту на основі обраної гри, обрати з усіх методів алгоритми, придатні для застосування у ігрових застосунках, використати гру як модель для тестування, на яких обрані алгоритмі ШІ будуть натреновані, розробити метод, завдяки якому будуть тестуватися обрані моделі ШІ.

Отже, можна виділити три критерії, які є основними для розробки алгоритму:

- Проаналізувати всі доступні методи ШІ
- Обрати необхідний для дослідження метод
- Підготовити модель для тренування на основі гри
- Застосувати алгоритми для тестування
- Виявити найкращій алгоритм для навчання ШІ

Виходячи з цього можна ставити наступні завдання:

1. Аналіз.

Штучний інтелект доволі широко використовується не тільки в ігрових застосунках, а і у багатьох інших сферах, і доволі успішно, тому потрібно краще вивчити тематичну літературу.

2. Виявлення метода для навчання.

Відомо, що існує не один метод навчання штучного інтелекту, тому потрібно порівняти та обрати саме той, який найбільше підходить для використання при навчанні ШІ у ігрових застосунках.

3. Підготовка моделі.

Підготовити модель на основі ігрового застосунку для порівняння на ній обраних алгоритмів навчання штучного інтелекту. Для цього потрібно буде врахувати

всі можливі аспекти поведінки ШІ і дозволити в повній мірі використати весь потенціал моделі для тестування.

4. Застосування алгоритму.

Застосувати алгоритми методів штучного інтелекту у ігрових застосунках, дотримуючись рівних умов тестування.

5. Виявлення найкращого алгоритму.

Порівнюючи методи слід знаходити слабкі та сильні сторони обраних методів. Також потрібно звернути увагу на обсяг даних, який можна обробляти за якийсь час, складність реалізації та вимоги до апаратних ресурсів. Мова програмування C#, середовище розробки Unity 5 з використанням Unity Machine Learning Agents.

Оцінюватись буде можливість штучного інтелекта навчитись керувати м'ячиком, вдало проходити рівні, при цьому збираючи геометричні фігури, які розставлені на шляху руху м'ячика.

Критеріями оцінки успіху будуть слугувати: складність написання та реалізації алгоритму, процес навчання, який буде змінюватись під час кожної спроби алгоритму для навчання.

2 ДОСЛІДЖЕННЯ ОБРАНИХ МЕТОДІВ НАВЧАННЯ ШТУЧНОГО ІНТЕЛЕКТА

2.1 Опис обраного алгоритму навчання штучного інтелекту

Дослідивши доступну інформацію що до алгоритмів навчання штучного інтелекту можна виділити декілька методів, які виявились найбільш популярними технологіями машинного навчання в розробці ігор. Це навчання з підкріпленням, імітаційне навчання і зворотне навчання з підкріпленням. В даному випадку найбільше підійшло навчання з підкріпленням.

1. Навчання з підкріпленням (Reinforcement Learning) - один із способів взаємодії, в ході якого випробувана система (агент) навчається, взаємодіючи із середовищем. Якщо спростити, то робота будується за принципом "дають - бери, б'ють - біжи", отримуючи після будь-яких взаємодій з середовищем або шкоду, або нагороду.

Цей алгоритм намагається знайти мінімально можливу кількість спроб для дослідження різних елементів світу і відповідає за довгострокове планування («можливо, якщо зараз тут побили , то потім там винагородять»). Для налагодження такого виду алгоритмів потрібні середовища зі складними, але формальними і все ж таки обмеженими правилами.

Як приклад можна навести DeepMind [19], екс-підрозділи Google, розробники якого навчили ШІ грати в Quake 3 Arena [20] приблизно так само, як це робить людина. Для навчання використовувалася система із підкріпленням.

Як правило, установка навчання з підкріпленням складається з двох компонентів, агента та середовища. Коли середовище посилається на об’єкт, на який діє агент, то агент починає виконувати алгоритм Навчання з підкріпленням. Середовище починається з надсилання стану агенту, який потім на основі своїх знань виконує дію у відповідь на цей стан. Після цього середовище надсилає агенту пару наступних станів і винагороду. Агент оновить свої знання винагородою, яку

повертає середовище, щоб оцінити свою останню дію. Цикл продовжується, доки середовище не надішле кінцевий стан, який закінчує виконування алгоритму.

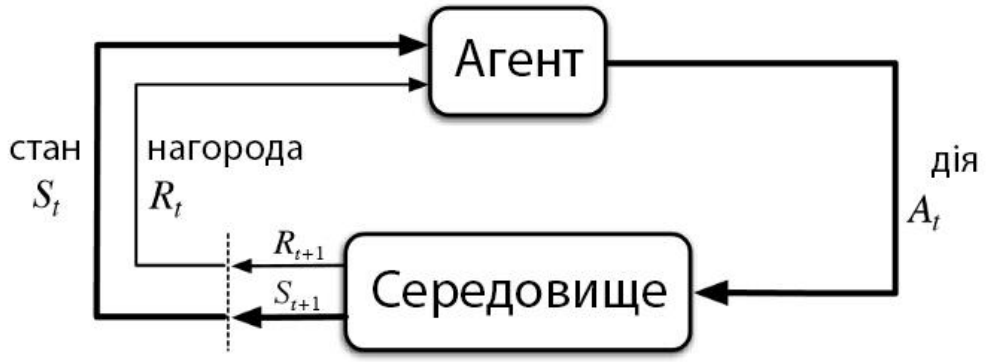

Рис. 2.1 - Схема алгоритму Навчання з підкріпленням

2.1.1 Метод навчання з підкріпленням

У навчанні з підкріпленням існує агент (agent) взаємодіє з навколишнім середовищем (environment), роблячи дії (actions). Навколишнє середовище дає нагороду за ці дії, а агент продовжує їх робити. Алгоритми з частковим навчанням намагаються знайти стратегію, яка приписує станам (states) навколишнього середовища дії, одну з яких може вибрати агент у цих станах. Середовище зазвичай формулюється як марківський процес прийняття рішень (МППР) [21] з кінцевою множиною станів, і в цьому сенсі алгоритми навчання з підкріпленням тісно пов'язані з динамічним програмуванням. Імовірності виграшів та переходу станів у МППР зазвичай є випадковими величинами, але стаціонарними в рамках завдання. При навчанні з підкріпленням, на відміну від навчання з учителем, не надаються вірні пари "вхідні дані-відповідь", а прийняття субоптимальних рішень (що дають локальний екстремум) не обмежується явно. Навчання із підкріпленням намагається знайти компроміс між дослідженням невивчених областей та застосуванням наявних знань (exploration vs exploitation). Баланс вивчення-застосування під час навчання з підкріпленням досліджується у задачі про багаторукому бандиті [22].

Формально найпростіша модель навчання з підкріпленням складається з:

- множини станів оточення (states) S;
- множини дій (actions) A;
- безлічі речовиннозначних скалярних "виграшів" (rewards).

У довільний час t агент характеризується станом st∈S і безліччю можливих дій A(st). Вибираючи дію a∈A(st), він перетворюється на стан st+1 і отримує виграш rt. На основі такої взаємодії з навколишнім середовищем агент, що навчається з підкріпленням, повинен виробити стратегію π:S→A, яка максимізує величину R=r0+r1+⋯+rn у разі МППР, що має термінальний стан, або величину:

$$R=\sum t\gamma trt,$$

для МППР без термінальних станів (де 0≤γ≤1 — множник, що дисконтує, для "майбутнього виграшу").

Таким чином, навчання з підкріпленням особливо добре підходить для вирішення завдань, пов'язаних із вибором між довгостроковою та короткостроковою вигодою.

2.1.2.1 Постановка завдання навчання із підкріпленням

S - безліч станів середовища

Гра агента із середовищем:

- ініціалізація стратегії π1(a|s) та стану середовища s1;
- для всіх t = 1 ... T:
    - агент вибирає дію at~πt(a|st);
    - середовище генерує нагороду rt+1~p(r|at,st) та новий стан st+1~p(s|at,st);
    - агент коригує стратегію πt+1(a|s).

Це марківський процес прийняття рішень (МППР), якщо P(st+1=s',rt+1=r|st,at,rt,st−1,at−1,rt−1,..,s1,a1) =P(st+1=s',rt+1=r|st,at), МППР називається фінітним, якщо |A|<∞, |S|<∞

Тепер, коли було визначено функцію виграшу, потрібно визначити алгоритм, який використовуватиме знаходження стратегії, що забезпечить найкращий результат. Наївний підхід до вирішення цього завдання передбачає такі кроки:

- випробувати усі можливі стратегії;
- вибрати стратегію з найбільшим очікуваним виграшем.

Перша проблема такого підходу полягає в тому, що кількість доступних стратегій може бути дуже великою або нескінченною. Друга проблема виникає, якщо виграші стохастичні — щоб точно оцінити виграш від кожної стратегії, потрібно багаторазово застосувати кожну з них. Цих проблем можна уникнути, якщо допустити деяку структуризацію і, можливо, дозволити результатам, отриманим від спроби однієї стратегії, проводити оцінку для інший. Двома основними підходами для реалізації цих ідей є оцінка функцій корисності та пряма оптимізація стратегій.

Підхід з використанням функції корисності використовує безліч оцінок очікуваного виграшу тільки для однієї стратегії (або поточної, або оптимальної). При цьому намагаються оцінити або очікуваний виграш, починаючи зі стану s, при подальшому дотриманні стратегії π,

$$V(s)=E[R|s,\pi],$$

або очікуваний виграш, при прийнятті рішення a в стані s та подальшому дотриманні π,

$$Q(s,a)=E[R|s,\pi,a],$$

Якщо для вибору оптимальної стратегії використовується функція корисності Q, оптимальні дії завжди можна вибрати як дії, що максимізують корисність. Якщо ж ми користуємося функцією V, необхідно мати модель оточення у вигляді ймовірностей $P(s'|s,a)$, що дозволяє побудувати функцію корисності виду

$$Q(s,a)=\sum_{s'}V(s')P(s'|s,a),$$

або застосувати так званий метод виконавець-критик, в якому модель ділиться на дві частини: критик, що оцінює корисність стану V, і виконавець, який вибирає потрібну дію в кожному стані. Маючи фіксовану стратегію π, оцінити E[R|·] при γ=1 можна просто усереднивши безпосередні виграші. Найбільш очевидний спосіб оцінки при γ∈(0,1) – усереднити сумарний виграш після кожного стану. Однак для цього потрібно, щоб МППР досяг термінального стану (завершився).

Тому побудова шуканої оцінки при γ∈(0,1) не очевидна. Проте, можна побачити, що R утворюють рекурсивне рівняння Беллмана:

$$E[R|st]=rt+\gamma E[R|st+1]$$

Підставляючи наявні оцінки V та застосовуючи метод градієнтного спуску з квадратичною функцією помилок, можна прийти до алгоритму навчання з тимчасовими впливами (temporal difference (TD) learning). У найпростішому випадку і стану, і дії дискретні і можна дотримуватись табличних оцінок для кожного стану.

2.1.2.2 Багаторукий бандит

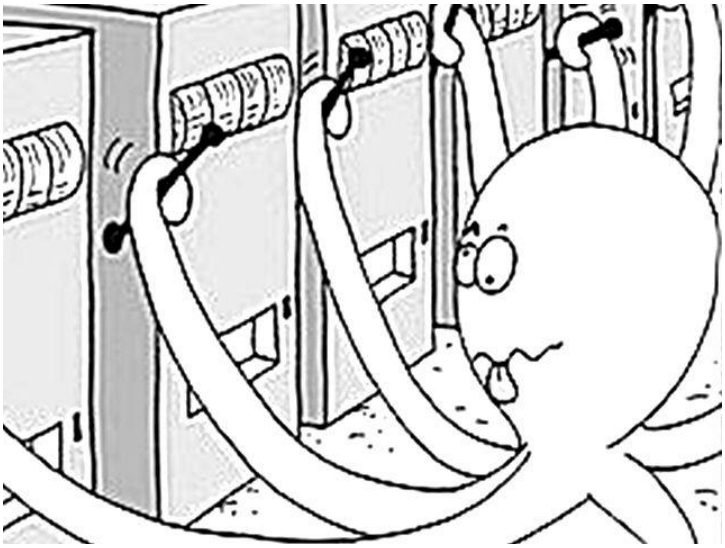

Рис. 2.2 - Автомат Багаторукий бандит

Формулювання.

A - безліч можливих дій (ручок автомата),

pa(r) — невідомий розподіл нагороди r∈R ∀a∈A,

πt(a) - стратегія агента в момент t ∀a∈A.

Гра агента із середовищем:

- ініціалізація стратегії π1(a);
- для всіх t = 1 ... T:
    - агент вибирає дію (ручку) at~πt(a);
    - середовище генерує нагороду rt-pat (r);
    - агент коригує стратегію πt+1(a).

Qt(a)=∑ti=1ri[ai=a]∑ti=1[ai=a]→max — середня нагорода у t іграх

, Q∗(a)=limt→∞Qt(a)→max — цінність дії a.

У нас є автомат - N-рукий бандит, на кожному кроці ми вибираємо за яку з N ручок автомата смикнути, тобто. безліч дій A = 1,2 ..., N.

Вибір дії at на кроці t тягне за собою нагороду R(at) при цьому R(a) ∀a∈A є випадковою величиною, розподіл якої невідомий. Стан середовища у нас від кроку до кроку не змінюється, а значить безліч станів S тривіально, ні на що не впливає, тому його можна проігнорувати. Для простоти слід думати, що кожній дії відповідає деякий розподіл, який змінюється з часом. Якби ці розподіли були відомі, то очевидна стратегія полягала б у тому, щоб підрахувати математичне очікування для кожного з розподілів, вибрати дію з максимальним математичним очікуванням і тепер робити це на кожному кроці.

Проблема в тому, що розподіли невідомі, однак можна оцінити математичне очікування деякої випадкової величини з невідомим розподілом. Для K експериментів ξk, оцінка математичного очікування це середнє арифметичне результатів експериментів:

$$E(\xi)=1K\sum Kk=1\xi k$$

Завдання є модельним для розуміння конфлікту між exploitation-exploration.

2.1.2.3 є-жадібна (є-greedy) стратегія

Нагорода для стратегії з різними є

Слід ввести параметр $\epsilon \in (0,1)$.

На кожному кроці t

- Отримаємо значення - випадкової величини рівномірно розподіленої на відрізку (0,1);
- Якщо $\alpha \in (0,\epsilon)$, то виберемо дію $at \in A$ випадково і рівноймовірно, інакше як у жадібній стратегії виберемо дію з максимальною оцінкою математичного очікування;
- Оновлюємо оцінки так само, як у жадібній стратегії.

Якщо $\epsilon=0$, це звичайна жадібна стратегія. Однак якщо $\epsilon>0$, то на відміну від жадібної стратегії на кожному кроці з ймовірністю є приходиться "дослідження" випадкових дій.

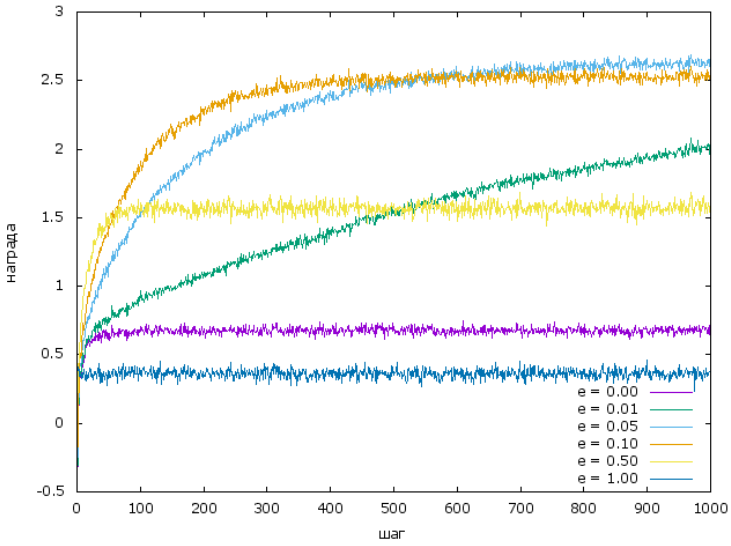

Рис. 2.3 - Нагорода для стратегії з різними є-жадібності

2.1.2.4 Стратегія Softmax

Основна ідея алгоритму softmax - зменшення втрат при дослідженні за рахунок більш рідкісного вибору дій, які мають невелику нагороду в минулому. Щоб цього домогтися кожної дії обчислюється ваговий коефіцієнт з урахуванням якого відбувається вибір дії. Чим більше Qt(a), тим більша ймовірність вибору a:

$$\pi t+1(a)=\exp(Qt(a)/\tau)\sum b\in A\exp(Qt(b)/\tau)$$

$\tau\in(0,\infty)$ — параметр, за допомогою якого можна настроювати поведінку алгоритму.

При $\tau\to\infty$ стратегія прагне до рівномірної, тобто softmax менше залежатиме від значення виграшу і вибирати дії більш рівномірно (exploration).

При $\tau\to 0$ стратегія прагне до жадібної, тобто алгоритм буде більше орієнтуватися на відомий середній виграш дій (exploitation).

Експонента використовується для того, щоб ця вага була ненульовою навіть у дій, нагорода від яких поки що нульова. Евристика: параметр τ має сенс зменшувати з часом.

2.1.2.5 Метод UCB

Попередні алгоритми після прийняття рішення використовують дані про середній виграш. Проблема в тому, що якщо дія дає нагороду з якоюсь ймовірністю, то дані від спостережень виходять не точні і ми можемо неправильно визначати найвигіднішу дію.

Алгоритм верхнього довірчого інтервалу (upper confidence bound або UCB) - сімейство алгоритмів, які намагаються вирішити цю проблему, використовуючи при виборі дані не лише про середній виграш, але й про те, наскільки можна довіряти значенням виграшу.

Також як softmax в UCB при виборі дії використовується ваговий коефіцієнт, який є верхньою межею довірчого інтервалу (upper confidence bound) значення виграшу:

$$\pi t+1(a)=Qt(a)+ba$$

$ba=2\ln\sum aPaPa\sqrt{}$ - бонусне значення, яке показує, наскільки недосліджена дія порівняно з іншими [23].

На відміну від попередніх алгоритмів, UCB не використовує у своїй роботі ні випадкові числа для вибору дії, ні параметри, якими можна впливати на його роботу. На початку роботи алгоритму кожна з дій вибирається по одному разу (для того, щоб можна було обчислити розмір бонусу для всіх дій). Після цього у кожний момент часу вибирається дія з максимальним значенням вагового коефіцієнта.

Незважаючи на це відсутність випадковості, результати роботи цього алгоритму виглядають досить не точно в порівнянні з іншими. Це відбувається через те, що цей алгоритм порівняно часто вибирає недосліджені дії.

2.1.2.6 Q-learning

На основі одержуваної від середовища винагороди агент формує функцію корисності Q, що згодом дає можливість вже не випадково вибирати стратегію поведінки, а враховувати досвід попередньої взаємодії з середовищем. Одна з переваг Q-навчання - те, що воно може порівняти очікувану корисність доступних дій, не формуючи моделі навколишнього середовища. Застосовується для ситуацій, які можна представити у МППР.

Таким чином, алгоритм це функція якості від стану та дії:

$$Q:S\times A\rightarrow R$$

Перед навчанням Q ініціалізується довільними значеннями. Після цього в кожний момент часу t агент вибирає дію at, отримує нагороду rt, переходить у новий

стан st+1, який може залежати від попереднього стану st та обраної дії, та оновлює функцію Q. Оновлення функції використовує виважене середнє між старим та новим значеннями :

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}},$$

де rt це нагорода, отримана при переході зі стану st у стан st+1, і це швидкість навчання (0<α≤1). Алгоритм закінчується, коли агент перетворюється на термінальний стан st+1 [24].

3 РОЗРОБКА ТА ПОРІВНЯННЯ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ У ІГРОВИХ ЗАСТОСУНКАХ

3.1 Інструменти для реалізації

На цей час існує багато разноманітних варіантів для того щоб реалізувати поставлену задачу. Головним критерієм повинна бути простота при застосуванні, доступність, швидка працездатність та зрозумілість інтерфейсу. Тому була обрана середа розробки ігрових застосунків Unity.

Рис. 3.1 – Інтерфейс Unity

Unity – це безкоштовне, міжплатформне середовище розробки комп'ютерних ігор, розроблене американською компанією Unity Technologies. Unity дозволяє створювати програми, 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. Основними перевагами Unity є наявність візуального середовища розробки, міжплатформної підтримки та модульної системи компонентів. До недоліків відносять появу складнощів при роботі з багатокомпонентними схемами та утруднення при підключенні зовнішніх бібліотек. На Unity написані тисячі ігор, програм, візуалізації математичних моделей, які охоплюють безліч платформ і

жанрів. У цьому Unity використовується як великими розробниками, і незалежними студіями [25]. Також Unity має свій спеціальний онлайн-магазин. У ньому можна знайти будь-які готові моделі, текстури чи ще щось. Щось із цього безкоштовне, а щось необхідно купувати. Дуже важливо, що цей магазин доступний прям у редакторі Unity. А це означає, що все обране у магазині буде закачуватись і додаватись у поточний проект, і не треба потім це все десь шукати, програма все зробить сама.

Також був використаний набір інструментів Unity Machine Learning Agents Toolkit (ML-Agents) — це новий плагін в ігровому движку Unity з відкритим вихідним кодом, який дозволяє іграм і симуляціям служити середовищем для навчання інтелектуальних агентів. Від гри у футбол до ходьби, стрибків зі стін та навчання ШІ собаки грі з палкою, Unity ML-Agents Toolkit надає широкий спектр умов для тренування агентів.

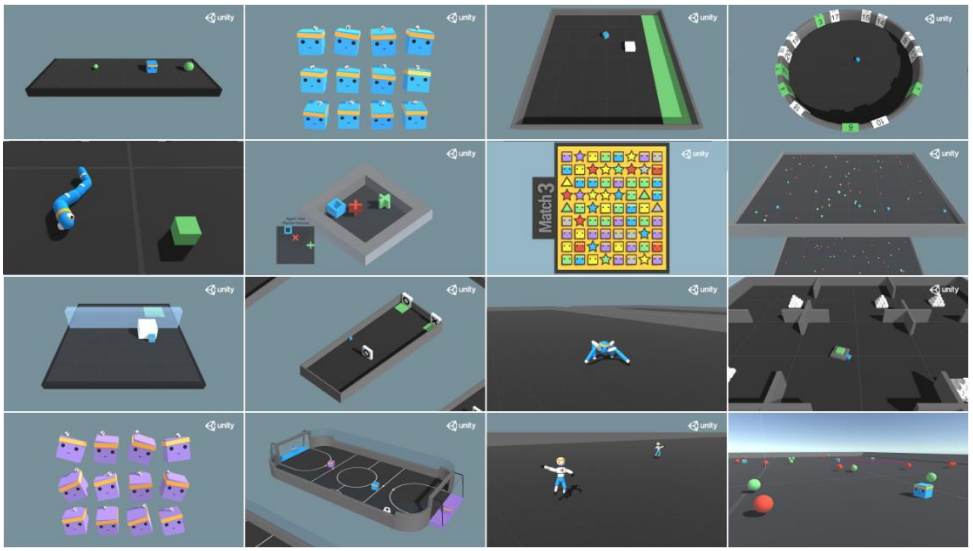

Рис. 3.2 – Приклади готових симуляцій ML-Agents в Unity

Особливості цього інструменту:

- Підтримка кількох конфігурацій середовища та сценаріїв навчання
- Гнучкий пакет SDK Unity, який можна інтегрувати у будь-яку гру або спеціальну сцену Unity

- Підтримка навчання сценаріїв конкуренції з одним, кількома агентами та кількома агентами за допомогою кількох алгоритмів Deep Reinforcement Learning
- Підтримка навчання з демонстрацій за допомогою двох алгоритмів імітаційного навчання
- Легко визначені сценарії навчання для складних завдань
- Навчаня надійних агентів за допомогою створення випадкового середовища
- Гнучкий контроль агента з прийняттям рішень на вимогу
- Навчання, використовуючи кілька одночасних екземплярів середовища Unity
- Використання Unity Inference Engine для забезпечення внутрішньої міжплатформної підтримки

Для написання та редагування коду буде використовуватись середовище розробки Microsoft Visual Studio 2019 та язик програмування C# (C Sharp). Microsoft Visual Studio — це серія продуктів фірми Майкрософт, які містять інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів.

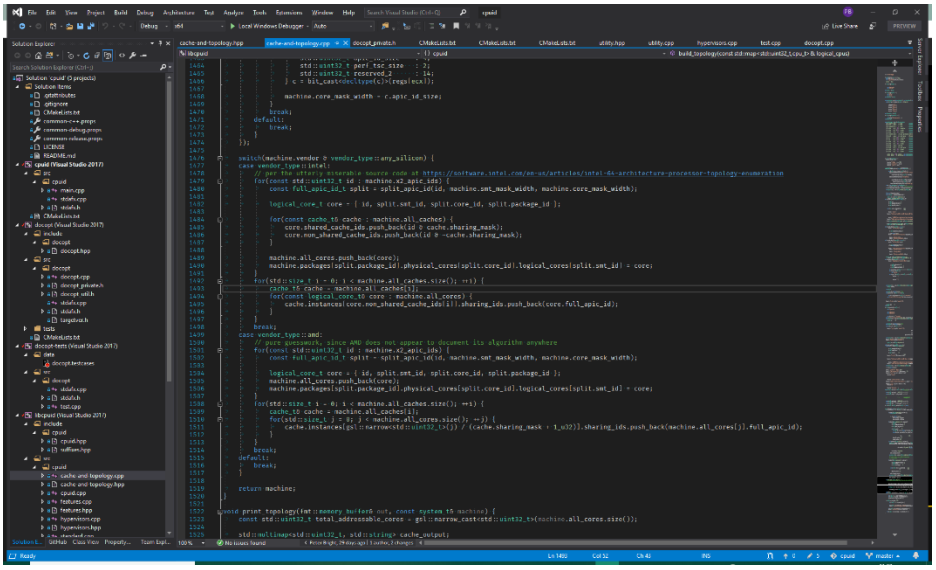

Рис. 3.3 – Інтерфейс Microsoft Visual Studio

3.2 Реалізація методів навчання штучного інтелекту

Для початку реалізації слід згадати як саме працює гра, на основі якої будуть навчатися всі обрані методи.

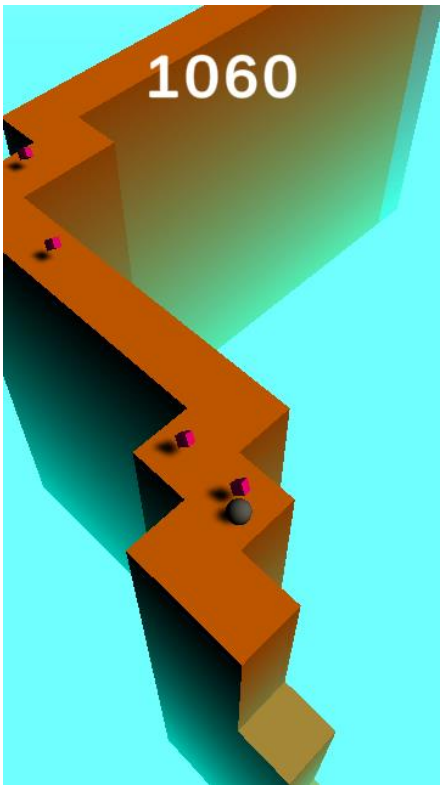

Рис. 3.4 – Гра ЗігЗаг

Як і писалося раніше мета гри доїхати м’ячиком до фінішу, по можливості, збирати всі кубики на шляху. Керування м’ячиком дуже просте, потрібно натискати на екран, при кожному натиску буде змінюватись напрямок руху м’ячика, вліво або вправо. З кожним проходженням рівнів нараховуються бали, а якщо збирати кубики, то балів буде більше. Якщо м’ячик падає з дороги вниз рівень починається спочатку.

Головним стимулом в навчанні ШІ є винагорода за правильні дії, тому наступним кроком необхідно визначити нагороду за правильні дії і те, що буде при не правильних діях, а саме, ці нагороди будуть відніматись.

```
private void TriggerCheck(Collider collidedObj)
{
    if (collidedObj.gameObject.CompareTag("reward"))
    {
        AddReward(1f);
        reward++;
        RewardCollector.Instance.ScoreAdd(reward);
    }
}
```

Рис. 3.5 – Присвоювання нагороди

Як можна побачити за кожну вдалу дію буде даватися один бал, всі ці бали будуть сумуватися, а при невдачі буде відніматися теж один бал.

```
private void FallCheck(Collider ObjCollided)
{
    if (ObjCollided.gameObject.TagCompare("moving"))
        move = true

    else if (ObjCollided.gameObject.TagCompare("fall")) || ObjCollided.gameObject.TagCompare("skip")
    {
        AddReward(-1f);
        EpisodeEnd();
    }
}
```

Рис. 3.6 – Віднімання нагороди

З усіх описаних алгоритмів в пункті 2.1.1 першим був обраний Q-learning. Далі, за допомогою інструкцій ML-Agents [26], для початку навчання алгоритму, була розроблена тренувальна конфігурація, на основі якої навчалися агенти. На рисунку 3.7 можна побачити фрагмент файла з тренувальною конфігурацією по алгоритму Q-learning. Для всіх інших алгоритмів були також створені такі конфігурації, але з урахуванням їх особливостей.

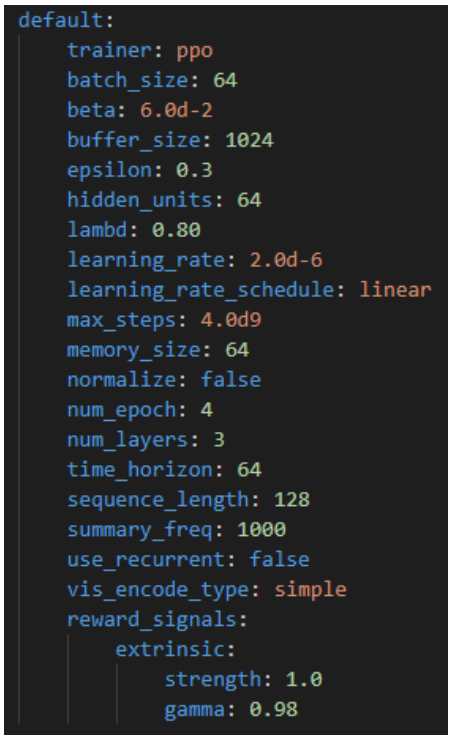

Рис. 3.7 – Фрагмент тренувальної конфігурації

Тепер необхідно запустити процес навчання. Для цього слід перейти до папки, де знаходиться ML-Agents та відкрити вікно терміналу, ввести команду, яка означає куди саме будуть записуватися все, що вивчить алгоритм.

```
mlagents-learn_zigzag_trainer_config.yaml--run-id="ZigZag_1"
```

Якщо система не знайде ніяких помилок, то все запуститься, в вікні терміналу з’явиться логотип Unity (рис. 15) і програма буде чекати запуску симуляції гри. Для чого, вже у програмі Unity, потрібно обрати вікно Game і зверху натиснути кнопку Плей, для запуску симуляції гри (рис. 16), одночасно з цим почнеться процес навчання ШІ.

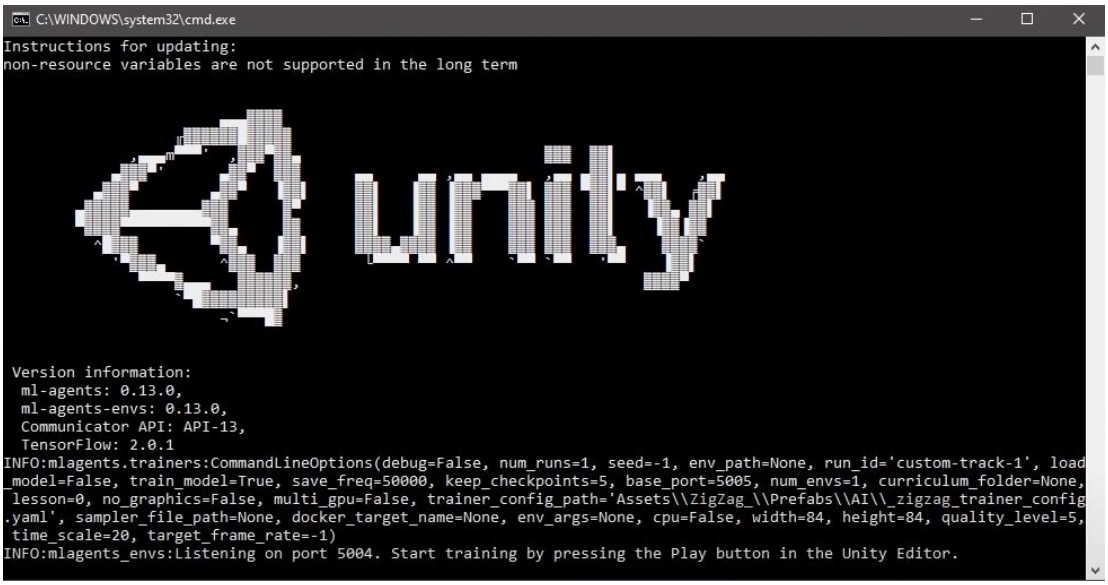

Рис. 3.8 – Початок роботи ML-Agents

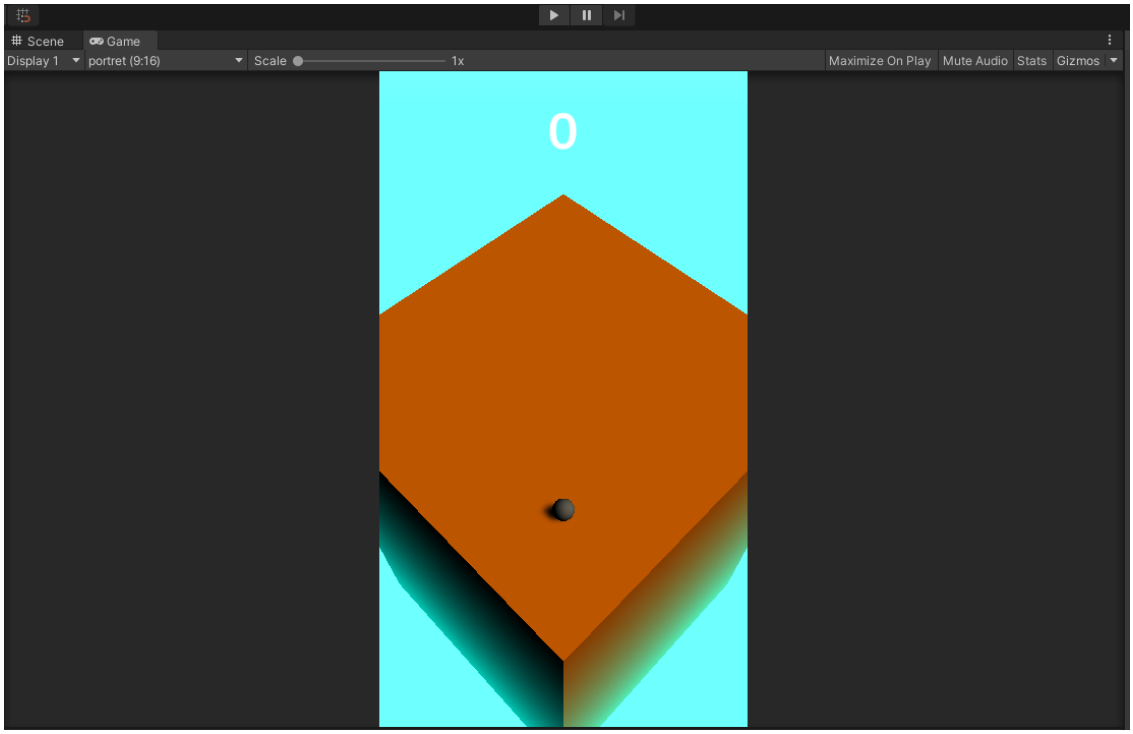

Рис. 3.9 – Початок симуляції гри для навчання штучного інтелекту

Для більшого контролю процесу навчання можна подивитись на графіки і зрозуміти як все проходить. Щоб це зробити необхідно відкрити нове вікно терміналу і застосувати наступну команду:

```
tensorboard--logdir=summaries
```

Після чого відкриється браузер з вкладкою, де можна побачити декілька графіків. На першому графіку (рис. 18) відображається сума нагород, яку отримує агент за кожну спробу.

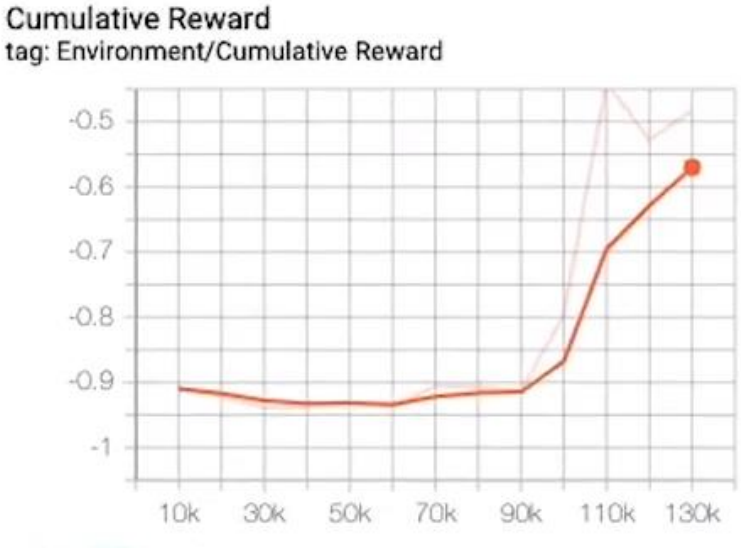

Рис. 3.10 – Графік накопичення нагород агента

А на наступному графіку можна побачити тривалість загального часу кожної спроби агента.

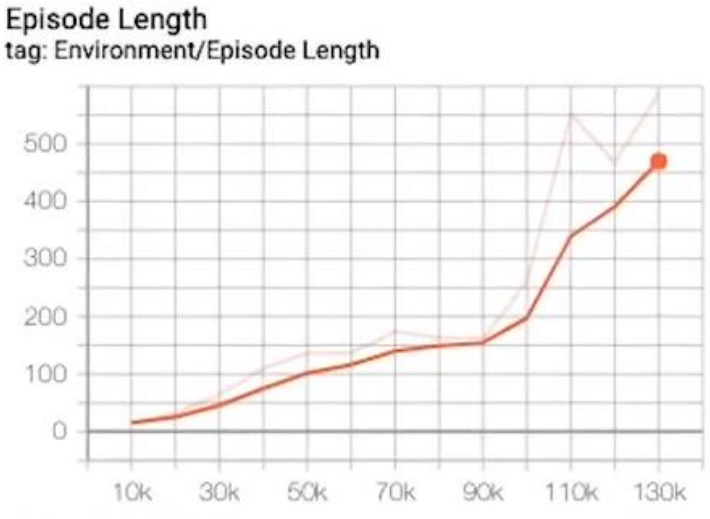

Рис. 3.11 – Графік тривалості кожної спроби

3.3 Порівняння кінцевих результатів навчання

На навчання кожного з 4-х алгоритмів було витрачено по 1 годині, цього виявилось цілком достатньо для формування загальної картини того, як ШІ навчився справлятися з поставленою задачею. За цей час кожен алгоритм зробив трохи більше 3000 спроб.

З перших хвилин навчання алгоритми показували себе дуже погано, спочатку ніяк не реагували і м’ячик просто падав вниз. Потім почали потроху намагатися щось робити, згодом вони інколи навіть доводили м’ячик до середини рівня, але це були поодинокі випадки. Десь через 20-30 хвилин навчання агенти починали показувати свої перші результати. Вони почали частіше проходити рівні, більш точно керувати м’ячиком, для того, щоб зібрати якомога більше кубиків по дорозі. Ближче к кінцю відведеного часу результати навчання переставали швидко рости, з’являлася більша стабільність і точність дій.

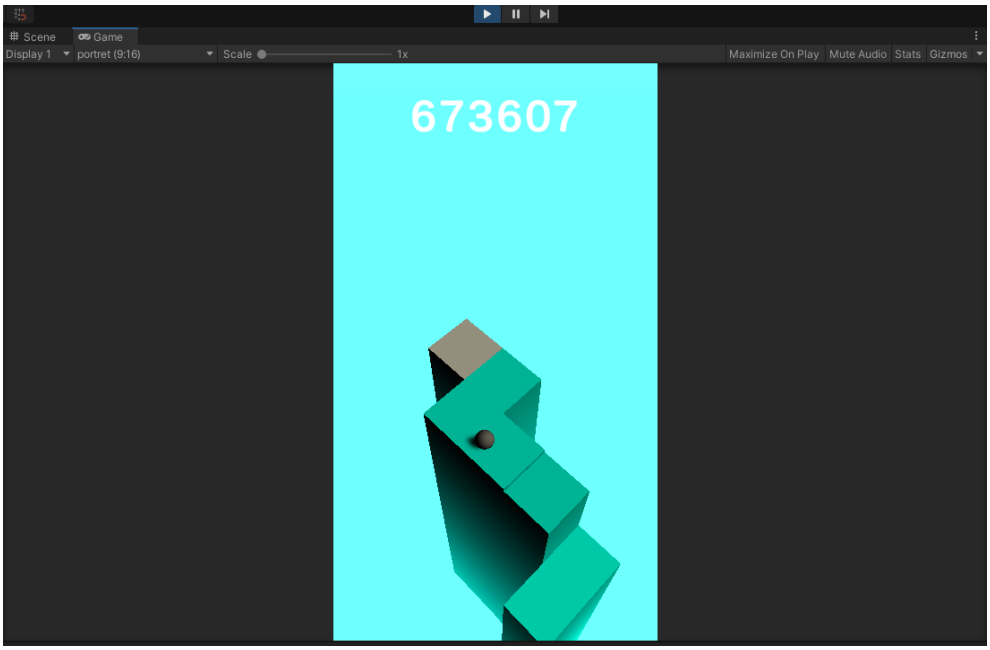

Рис. 3.12 – Процес навчання агента

Після завершення навчання ШІ за допомогою обраних методів були отримані наступні графіки результатів:

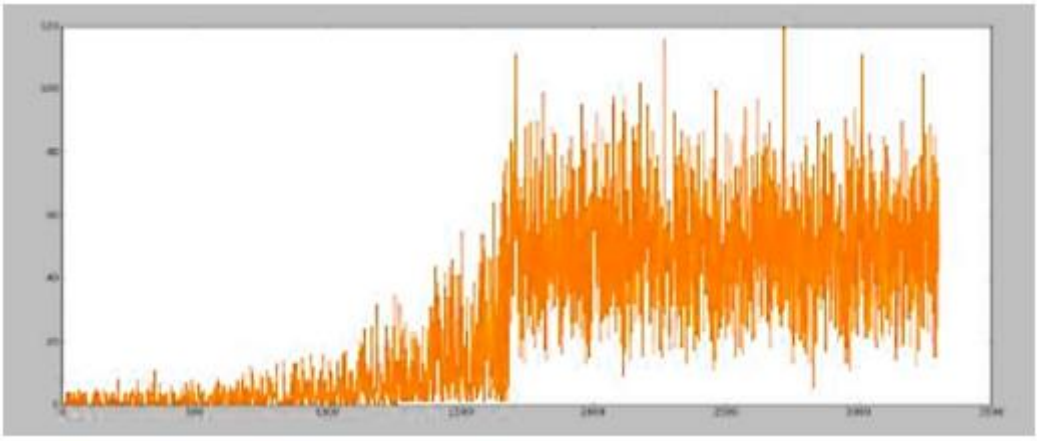

Рис. 3.13 – Графік процесу навчання за допомогою метода Q-learning протягом більш 3000 спроб

На рисунку 3.13 можна побачити як показав себе метод навчання Q-learning. Помітно як на середині процесу агент стрімко почав навчатися, але трохи пізніше це зростання сповільнилось. На другому рисунку (рис 3.14) показано середнє значення метода Q-learning за кожні 100 спроб.

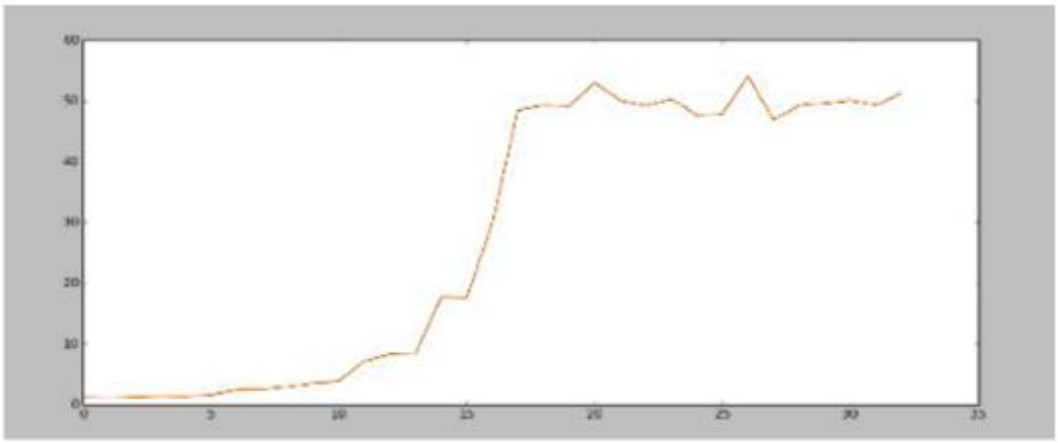

Рис. 3.14 – Графік середнього значення процесу метода Q-learning

Наступним методом навчання був Softmax. Процес проходив за таких же умов, що і перший алгоритм. На середині графіку помітно, як в один момент агент почав дуже швидко вчитись і майже відразу сповільнився і зростання продовжилось дуже повільно (рис 3.15).

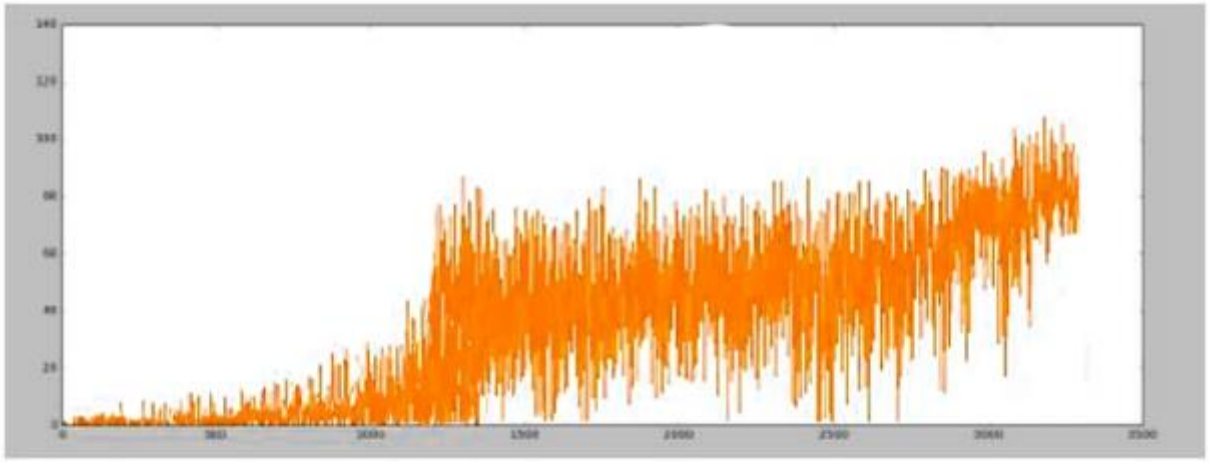

Рис. 3.15 – Графік процесу навчання за допомогою метода Softmax протягом більш 3000 спроб

Середнє значення алгоритму трохи плавніше, але все одно помітне швидке зростання (рис. 3.16).

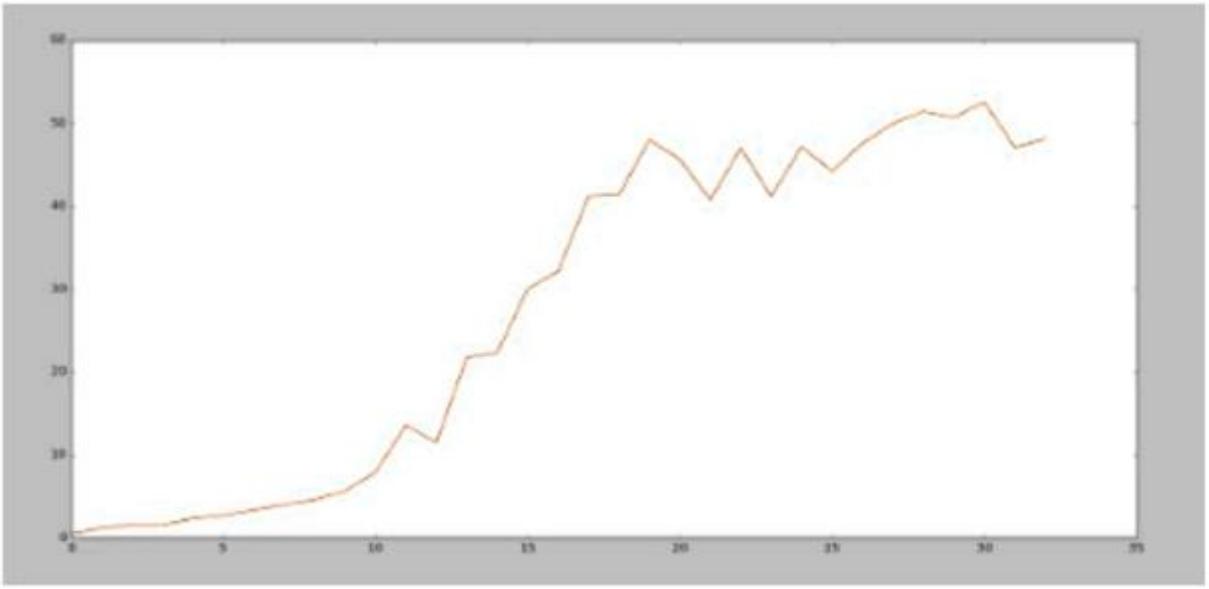

Рис. 3.16 – Графік середнього значення процесу метода Softmax

Третій метод навчання є-жадібна (є-greedy) стратегія. Він виявився самим повільним та плавним. На середині графіку знову помітно зростання навчання, але воно було найплавніше з усіх обраних методів (рис. 3.17).

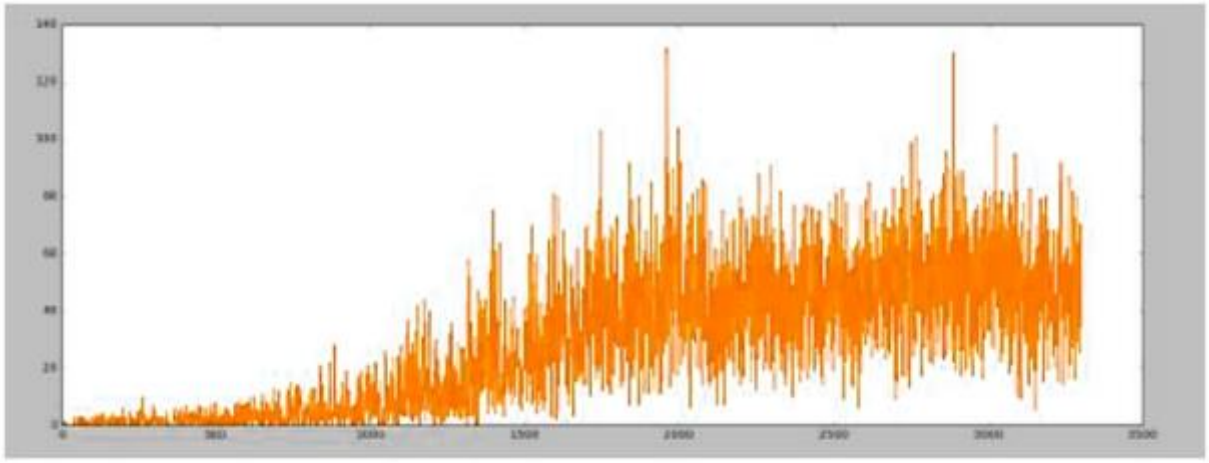

Рис. 3.17 – Графік процесу навчання за допомогою метода є-greedy протягом більш 3000 спроб

На наступному графіку цього методу якраз дуже помітне повільний процес навчання, але в той же час дуже стабільне зростання навичок.

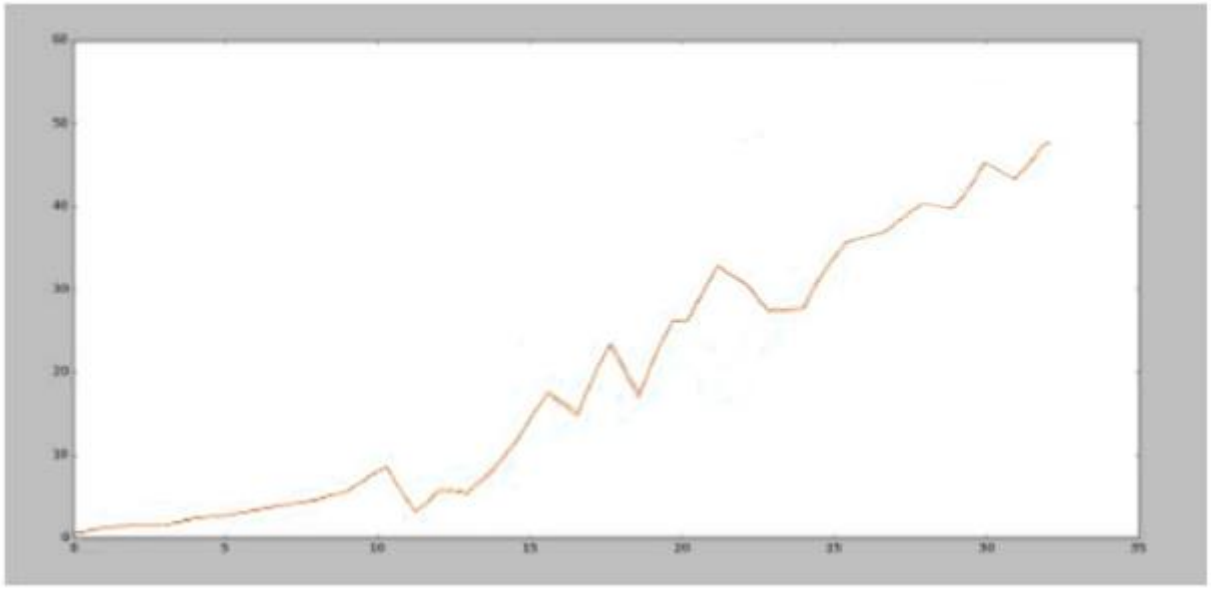

Рис. 3.18 – Графік середнього значення процесу метода є-greedy

Останній алгоритм навчання штучного інтелекту метод UCB. Якщо придивитись, то цей графік процесу навчання трохи нагадує перший метод своїм стрімким зростанням у середині, але це не так, далі прогрес навчання сильно сповільнюється і не росте (рис. 3.19).

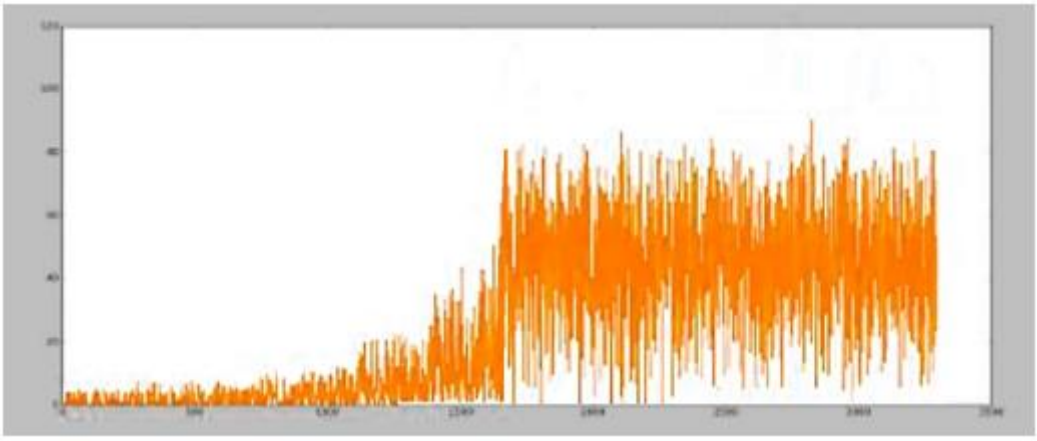

Рис. 3.19 – Графік процесу навчання за допомогою метода UCB протягом більш 3000 спроб

На другому графіку цього методу добре помітно стрімке, але й водночас плавну зміну графіку.

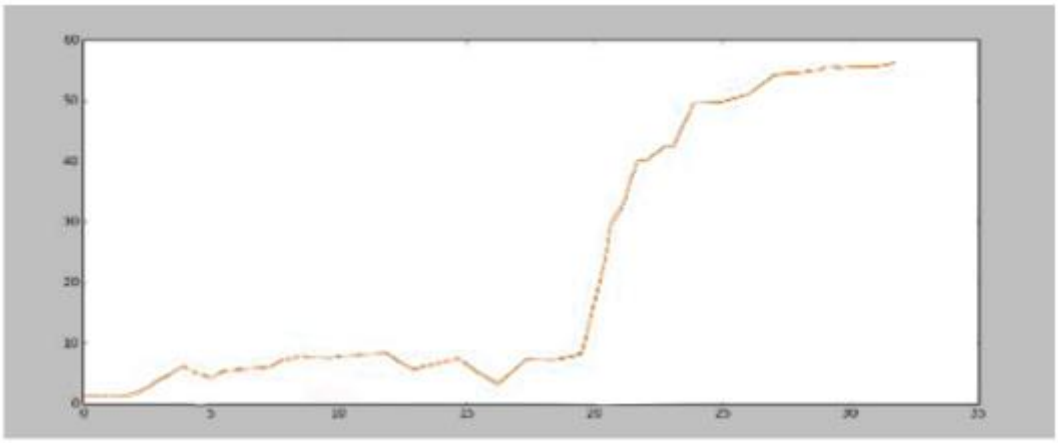

Рис. 3.20 – Графік середнього значення процесу метода UCB

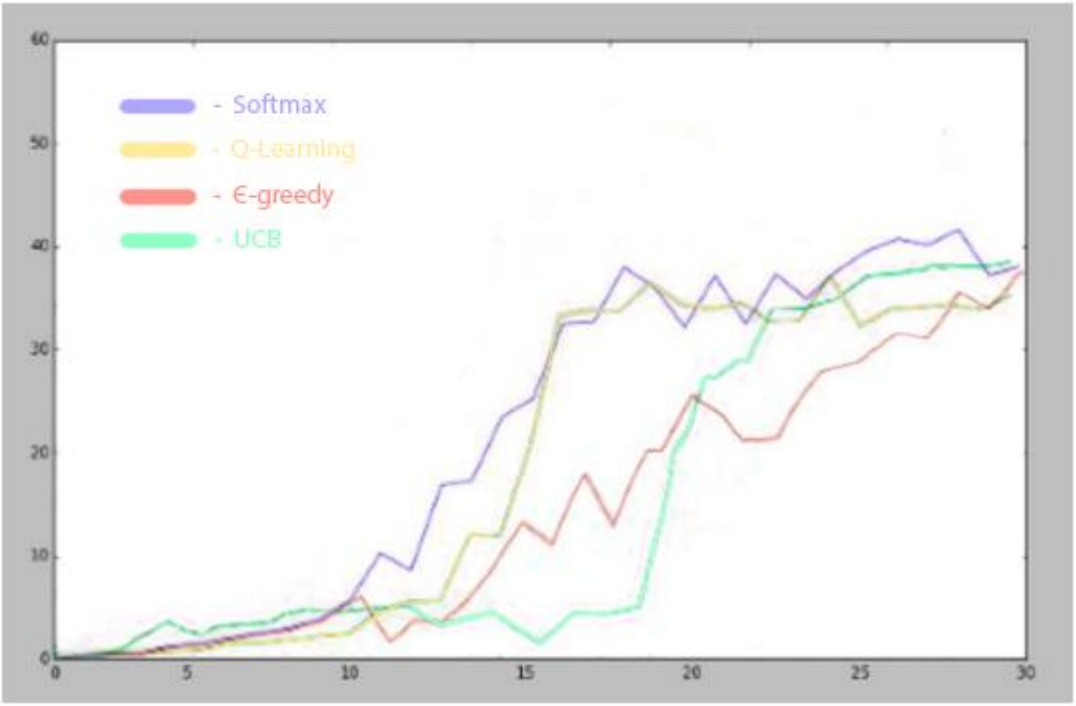

Рис. 3.21 – Графік порівняння алгоритмів

Порівнявши результати обраних алгоритмів, можна добре побачити, як саме відрізняються данні методи навчання ШІ. Перший тестований алгоритм Q-learning має досить не погане зростання на середині графіку, але потім навчання різко сповільнюється, що добре помітно. Далі Softmax, в нього самий ранній початок зростання, та досить плавний, потім зростання трохи падає, але цього достатньо для продовження навчання агента. Наступним методом навчання був є-greedy. Не зважаючи на рівне зростання графіку, він виявився самим повільним, що іноді буває дуже важливо знати, коли необхідно швидко навчати штучний інтелект. Останній алгоритм, UCB, виявився дуже повільним спочатку і аж до середини, але потім стрімко почав рости. Якщо для навчання агентів є не багато часу, то цей метод не підходить.

ВИСНОВКИ

Метою магістерської роботи стало дослідження та розробка методів штучного інтелекту у ігрових застосунках.

Під час цієї роботи було проаналізовано існуючи на даний час алгоритми та методи навчання штучного інтелекту та з них обрані ті, які найбільше підходять для використовуються у ігрових застосунках для симуляції розумної поведінки комп'ютера або схожої на поведінку людини-гравця.

Після проведення досліджень предметної області було з’ясовано, що метод Навчання з підкріпленням (Reinforcement Learning) - є один із найбільш підходящих способів взаємодії, в ході якого випробувана система (агент) навчається, взаємодіючи із навколишнім середовищем. А також були обрані алгоритми для навчання ШІ.

Обраний метод та алгоритми були детально розглянуті, була вивчена література та зроблені певні висновки. Також була обрана середа розробки і інструменти розробки, а також модель, на основі мобільної гри, для тестування алгоритмів штучного інтелекту.

На основі всіх виконаних підготовчих процесів було виконано дослідження алгоритмів навчання штучного інтелекту на основі ігрових застосунків. З усіх обраних і досліджених алгоритмів був обраних один, Softmax, яких показав себе найкраще.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. A. M. Turing (1950) Computing Machinery and Intelligence. Mind 49: 433-460.

2. Artificial intelligence : a modern approach/ Stuart Russell, Peter Norvig. Includes bibliographical references and index. ISBN 0-13-103805-2 1. Artificial intelligence I. Norvig, Peter. II. Title.Q335.R86 1995

3. Lecture 1: Introduction and Scope [Electronic resourse] - Mode of access: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/lecture-videos/lecture-1-introduction-and-scope/

4. Deep Blue [Electronic resourse] - Mode of access: https://www.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/

5. AlphaGo [Electronic resourse] - Mode of access: https://deepmind.com/research/case-studies/alphago-the-story-so-far

6. 4 Types of Artificial Intelligence [Electronic resourse] - Mode of access: https://www.bmc.com/blogs/artificial-intelligence-types/

7. AI, Deep Learning, and Machine Learning: A Primer [Electronic resourse] - Mode of access: https://a16z.com/2016/06/10/ai-deep-learning-machines/

8. Executive Office of the President National Science and Technology Council Committee on Technology [Electronic resourse] - Mode of access: https://obamawhitehouse.archives.gov/sites/default/files/whitehouse_files/microsites/ostp/NSTC/preparing_for_the_future_of_ai.pdf

9. Яблоков К. В. Исторические компьютерные игры как способ моделирования исторической информации // История и математика: Анализ и моделирование социально-исторических процессов / Ред. Малков С. Ю., Гринин Л. Е., Коротаев А. В. М.: КомКнига/УРСС, 2007. С. 263—303

10. Ender's Game (1/10) Movie CLIP - The Mind Game (2013) HD [Electronic resourse] - Mode of access: https://www.youtube.com/watch?v=oDRFKZVZwcA

11. Red Dead Redemption 2 [Electronic resourse] - Mode of access: https://www.rockstargames.com/ru/games/reddeadredemption2

12. Dr Mike Cook [Electronic resourse] - Mode of access: http://eecs.qmul.ac.uk/profiles/cookmichael.html

13. James Scott Nick Polson Artificial intelligence 2019p. c 273-280.

14. Woodcock, S. 2015. Game AI: The state of the industry, available online at http://www.gamasutra.com/features/20001101/.

15. G. N. Yannakakis. Preference Learning for Affective Modeling. In Proceedings of the Int. Conf. on Affective Computing and Intelligent Interaction, pages 126–131, Amsterdam, The Netherlands, September 2009. IEEE

16. A. J. Champandard. AI Game Development. New Riders Publishing, 2014.

17. G. N. Yannakakis and J. Togelius. Experience-Driven Procedural Content Generation. IEEE Transactions on Affective Computing, 2:147–161, 2011.

18. Z. Zeng, M. Pantic, G. Roisman, and T. Huang. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. IEEE Trans. Pattern Analysis and Machine Intelligence, 31(1):39–58, 2019.

19. DeepMind [Electronic resourse] - Mode of access: https://ru.wikipedia.org/wiki/DeepMind

20. Quake III Arena [Electronic resourse] - Mode of access: https://ru.wikipedia.org/wiki/Quake_III_Arena

21. Markov decision process [Electronic resourse] - Mode of access: https://en.wikipedia.org/wiki/Markov_decision_process

22. Multi-armed bandit [Electronic resourse] - Mode of access: https://en.wikipedia.org/wiki/Multi-armed_bandit

23. The Upper Confidence Bound Algorithm [Electronic resourse] - Mode of access: https://banditalgs.com/2016/09/18/the-upper-confidence-bound-algorithm/

24. Обучение с подкреплением [Електронний ресурс] – Режим доступу: http://neerc.ifmo.ru/wiki/index.php?title=mobileaction=toggle_view_desktop

25. Unity (игровой движок) [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Unity/

26. About ML-Agents package [Electronic resourse] - Mode of access: https://docs.unity3d.com/Packages/com.unity.ml-agents@2.0/manual/index.html