Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____Юрій КРАВЧЕНКО
«_____»_____2022 року

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

галузі знань 17 «Електроніка та телекомунікації»
за спеціальністю 172 «Телекомунікації та радіотехніка»
освітньо-професійна програма «Мережеві та інтернет технологій»

на тему:

РОЗРОБКА КОМП'ЮТЕРНОЇ ГРИ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав: студент групи МІТ -41
Абдуллаєв Аліяр Бахруз Огли _____
(ім'я та ПРІЗВИЩЕ) (підпис)

Керівник: завідувач кафедри мережевих та інтернет технологій
(посада)

Ю.В. Кравченко _____
(науковий ступень, вчене звання, ім'я та ПРІЗВИЩЕ) (підпис)

Київ - 2022

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Юрій КРАВЧЕНКО

«_____»_____2021 року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ

Здобувачу вищої освіти Абдуллаєву Аліяру Бахруз Огли
(прізвище, ім'я, по батькові)

1. Тема роботи:
Розробка комп'ютерної гри з використанням технологій штучного інтелекту
затверджена на засіданні кафедри МІТ «24» грудня 2021 р. протокол №8
2. Термін здачі закінченої роботи «30» травня 2022 р.
3. Вихідні дані до проекту (роботи)
Сучасні технології створення комп’ютерних ігор
4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити, обсяг – 35-40 стор.)
Вступ
1. Дослідження технології створення комп’ютерних ігор. Постановка задачі
1.1 Огляд та аналіз комп'ютерної гри з використанням технологій штучного інтелекту
Постановка задачі
2. Розробка комп'ютерної гри з використанням технологій штучного інтелекту.
3. Дослідження розробленої комп'ютерної гри
3.1. Алгоритм дослідження гри
3.2. Оцінка ефективності гри
3.3. Рекомендації до підвищення ефективності гри
Висновки
5. Перелік графічного матеріалу 8-10 слайдів
Дата видачі завдання

Керівник роботи _____ Юрій КРАВЧЕНКО
(підпис) (посада, прізвище, ім'я, по батькові)

Завдання прийняв до виконання _____ Аліяр АБДУЛЛАЄВ
(підпис) (ім'я, ПРІЗВИЩЕ)

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ РОБОТИ

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Підготовчий	20.01.2022	
2	Розділ 1	15.02.2022	
3	Розділ 2	15.03.2022	
4	Розділ 3	15.04.2022	
5	Доповідь та слайди	25.05.2022	
6	Пояснювальна записка	30.05.2022	

Здобувач вищої освіти _____ Аліяр АБДУЛЛАЄВ
(підпис)

Керівник _____ Юрій КРАВЧЕНКО
(підпис)

РЕФЕРАТ

Пояснювальна записка: 45 с., 13 рис., 3 табл., 14 джерел.

Мета роботи: на основі дослідження різних схем роботи штучного інтелекту в іграх, адаптувати один з методів для реалізації власної програми, розробити гру.

Об’єкт дослідження: організація роботи штучного інтелекту у сфері комп'ютерних ігор.

Предмет дослідження: характеристичні особливості методів побудови ефективної моделі ігрового штучного інтелекту.

Методи дослідження: вивчення теоретичних матеріалів, побудова логічних схем та моделей, перевірка теоретичних даних в ігровому рушії.

У ході даної дипломної роботи було проведено аналітичний огляд типів комп'ютерних ігор, де застосований штучний інтелект, та способів організації роботи штучного інтелекту; побудована власна модель штучного інтелекту; розроблено гру з використанням технологій штучного інтелекту.

Після створення гри була проведена оцінка її якості та визначено рекомендації щодо її підвищення.

З практичної точки зору, створена гра може бути використана будь-яким користувачем, а також слугувати прототипом для створення ігор з використанням технологій штучного інтелекту для інших розробників.

Наукова новизна дослідження полягає у евристичній концепції створення ігор зі штучним інтелектом та в конкретних рекомендаціях щодо покращення якості подібних ігор.

Галузь використання: комп'ютерні ігри.

Ключові слова:

ШТУЧНИЙ ІНТЕЛЕКТ, ГРА, ШАХИ, ЕВРИСТИЧНЕ ПРОГРАМУВАННЯ, MINIMAX, АЛЬФА-БЕТА ОБРІЗКА, JAVASCRIPT

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI – Artificial intelligence

JS – JavaScript

ШІ – Штучний інтелект

ВСТУП

Світовий ринок медіаспоживання швидко змінюється, телебачення всіляко витісняється захоплюючими інтернет-технологіями і комп'ютерними іграми.

Зростання технологічних можливостей і інтересу до ігрової індустрії робить область інтерактивних розваг областю інформаційних технологій у світі, що швидко розвивається.

На даний момент комп'ютерні ігри – не просто фрагменти коду, а вдало акумульовані методи виразності та передачі інформації, сукупність візуальної складової, що представлена досить розвиненими технологіями комп'ютерної графіки та візуалізації анімаційних ефектів, музичного супроводу та застосування засобів штучного інтелекту.

Ігровий штучний інтелект став одним із основних засобів виразності в комп'ютерних іграх і найчастіше визначає рівень якості кінцевого продукту розробників.

Ігровий штучний інтелект дещо відрізняється від звичного академічного розуміння штучного інтелекту. Його основною метою і досі залишається створити видимість інтелектуальності внутрішньоігрових персонажів, природності їхньої поведінки, реакцій та адекватності їх намірів. Ігровий штучний інтелект досить часто використовує деякі галузі знань штучного інтелекту як науки в цілому, включаючи різноманітні алгоритми пошуку оптимального маршруту, алгоритми управління та прийняття рішень, проте ігрова індустрія також може похвалитися і досить складними взаємопов'язаними інтелектуальними системами, які і виділяють ігровий штучний інтелект в самостійну галузь знань.

Ігровий штучний інтелект часто описують за допомогою програмних методик та іншого технічного інструментарію, що використовуються розробниками при написанні комп'ютерної гри з метою створення так званих інтелектуальних ігрових систем, призначених насамперед для того, щоб надати гравцеві цікавий ігровий процес, відчуття та враження від гри.

Будь-яка зміна стану ігрової системи вимагатиме певної множини рішень від певної множини ігрових персонажів, тому розробка простих і ефективних моделей управління прийняття рішень дозволяє зробити ігровий процес більш органічним і природним.

Поняття «комп'ютерні шахи» є ровесником науки кібернетики та її розділу «штучний інтелект». Шахи є ідеальною моделлю для дослідження складних завдань, особливо тих, де потрібен перебір варіантів. Розробку шахової програми відносять до проблеми розробки штучного інтелекту з таких причин:

– завдання має відношення до проблеми штучного інтелекту, оскільки шахи вважаються інтелектуальною грою;

– існує об'єктивний критерій зробленої роботи – сила шахової програми;

– велика диференційованість цього критерію – можливість поступового вдосконалення програми.

1. ДОСЛІДЖЕННЯ ТЕХНОЛОГІЇ СТВОРЕННЯ КОМП'ЮТЕРНИХ ІГОР. ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд та аналіз комп'ютерної гри з використанням технологій штучного інтелекту

Використання штучного інтелекту в комп'ютерних іграх на сьогоднішній день є необхідною практикою, якщо ми говоримо про комерційні великобюджетні проекти, оскільки подібні технології дозволяють не лише скоротити витрати на виробництво, а й розширити ігрові можливості. За рахунок автоматизованих процесів можна заощадити на роботі художників і розробників, які можуть витрачати багато часу на створення однотипних рівнів і персонажів, тоді як технології процедурного генерування справляються з цим завданням швидше.

Крім установки на економію, впровадження штучного інтелекту в ігри зумовлене бажанням внести до них різноманітність та створити правдоподібну імітацію «спонтанної» поведінки персонажів. Наприклад, у 2017 році компанія Nvidia спільно з Remedy Entertainment створили автоматизовану технологію глибокого навчання в реальному часі, яка анімує міміку 3D-моделей відповідно до запропонованих аудіофайлів. Особливістю подібної технології є низький час очікування та досить високий рівень правдоподібності результату (порівняно з іншими системами анімації осіб). Це може значно прискорити та вдосконалити процес створення неігрових персонажів.

Генеративно-змагальні мережі набувають все більшої популярності, оскільки можуть запропонувати певний рівень спонтанності та непередбачуваності. З їхньою допомогою створюються ігрові рівні, предмети, анімація і рухи персонажів та й самі гри. Дослідники М. Газдіал (M. Guzdial) та М. Рідл (M. Riedl) [1] зі Школи інтерактивної обчислювальної техніки (Технологічний інститут Джорджії) розробили метод машинного навчання, який генерує гру на основі «переглянутих» (оброблених) відео проходження ігор Super Mario Bros. (1985 р.), Kirby's Adventure (1993 р.) та Mega Man (1987 р.), зроблених для NES

(рис.1.1). Робота дробилася на два основні напрямки: 1) генерація етапів гри, структури, рівнів; 2) створення правил, механік, динаміки гри; іншому – наративу, аудіовізуальній частині приділялося менше уваги, принаймі, на початковій стадії розробки.

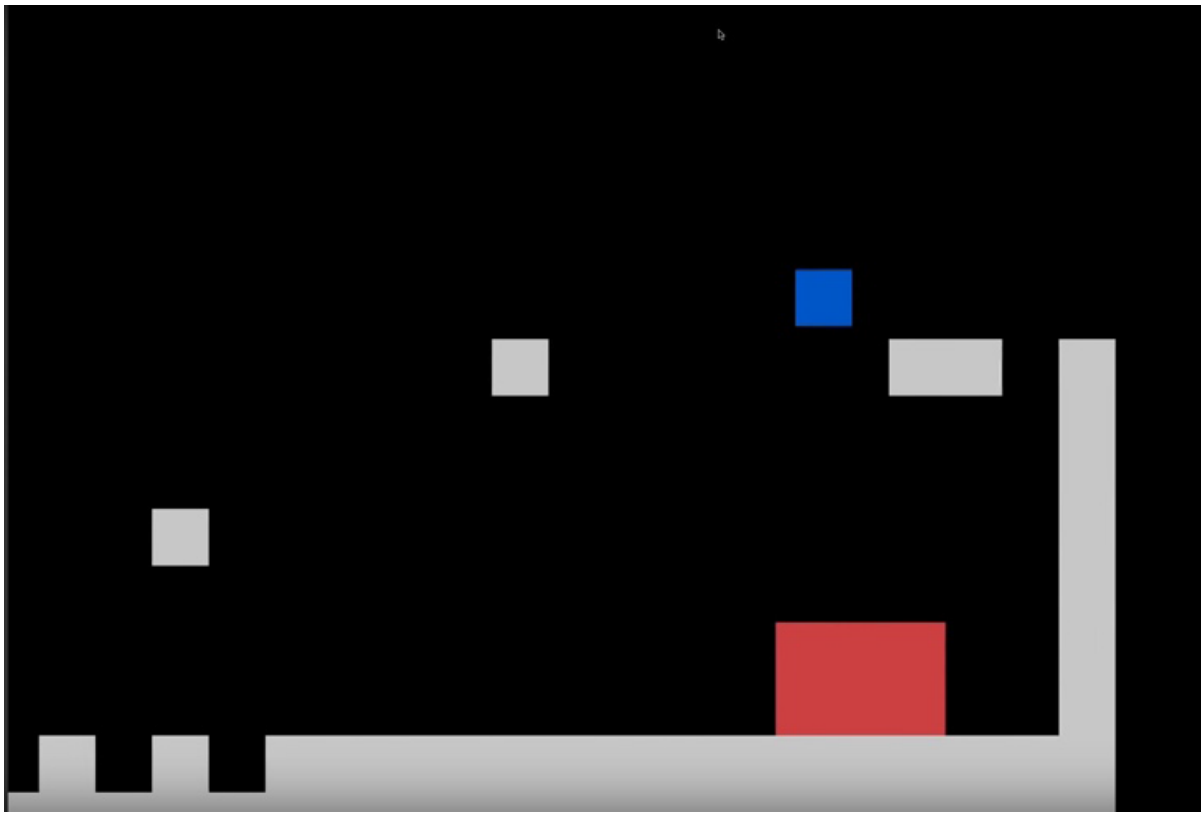

Рисунок 1.1 – Скріншот гри М. Газдіала та М. Рідла.

Важко в найближчому майбутньому очікувати створення незвичайних ігор, що задовольняють вибагливий смак сучасного геймера, що працюють на основі виключно генеративних алгоритмів, у той час як впровадження окремих елементів, зроблених за їх допомогою, вже має успіх. 2D-гра Starbound (2016 р.) претендує на роль успішного проекту з процедурного генерування контенту (рис.1.2). У ній рівні, блоки всередині рівня, флора та фауна генеруються щоразу заново. Звичайно, у грі є встановлені елементи, але їх конфігурації є унікальними. За рахунок більш простої графіки та геймплею порівняно з No Man's Sky (2016 р.), Starbound вдалося впоратися з поставленим завданням.

Рисунок 1.2 – Скріншот рівня з гри Starbound.

Оскільки штучний інтелект – це цілий сонм різних технологій і галузь наукового знання, говорити про якесь єдине завдання, яке стоїть перед ним, не доводиться.

Кожна система має своє вузькоспрямоване призначення (програма, здатна розпізнавати образи, розпізнавати мову тощо). Незважаючи на буквальне значення терміна, введеного в науковий обіг у 1956 році, штучний інтелект далеко не завжди передбачає симуляцію людських розумових здібностей. Родоначальник напряму, американський інформатик Джон Маккарті стверджував, що «інтелект» [intelligence] – це обчислювальна частина здатності досягати цілей у світі [2]. Різного роду інтелектуальні здібності зустрічаються у людей, багатьох тварин і навіть деяких машин. Очевидно, що під штучним інтелектом не мається на увазі утопічна симуляція людського мислення, це лише набір технологій, спрямованих на вирішення множинних завдань за допомогою обчислень, алгоритмів і навчання. Звичайно, дані технології прагнуть деяких творчих або розумових здібностей, властивих людині, але в цілому являють собою моделі обчислень, які аналізують, зіставляють, класифікують дані, шукають шляхи вирішення завдань та ін.

Штучні нейромережі навчаються, їх алгоритми калібруються під конкретні потреби і вони демонструють здібності, що перевершують людські (наприклад, прогнози в медицині). Незважаючи на вражаючі результати тих же штучних нейромереж, в них здійснюються формальні процедури: хоча результат їх роботи і порівняний в деяких випадках з результатом розумової діяльності людини, ця робота заснована на виконанні формальних процедур, і ми не можемо зробити висновок, що має місце мислення. Тотожність результатів не свідчить про тотожність процесів (формальних операцій та мислення). Вихідні дані нейромереж – це не наслідок мислення, але результат адаптації техніки під нашу дійсність шляхом мільйонів спроб і помилок (відкриття методу зворотного поширення помилки значно вплинуло на розвиток штучних нейромереж). Відсутність мислення не зменшує можливостей штучного інтелекту, просто важливо пам'ятати про нього при зверненні до цієї галузі. Штучний інтелект передбачає будь-яку «розумну» поведінку техніки, тобто коли вона реагує на користувача, підлаштовується під нього, якби вона могла відчувати, бажати, приймати рішення. Так, гра Nevermind (2015 р.) змінює рівень складності залежно від реакцій гравця: якщо він дав волю страху, гра стане складніше, якщо йому вдалося заспокоїтись, гра стане легшою. Смарт-об'єкти сучасного світу не мислимі без штучного інтелекту [3].

Дослідження систем штучного інтелекту та розробка комп'ютерних ігор довгі роки йшли паралельно та перетиналися лише в «ігровій» частині. З 1950-х років настільні ігри були полем застосування можливостей штучного інтелекту: прагнення створити алгоритми, які допомогли б машині перемогти людину в шашки, шахи, го, стало двигуном для розвитку технології. З появою аркадних гральних автоматів, дослідники все частіше зверталися до відеоігор, проте потужності старих ігор не дозволяли застосовувати розробки в галузі штучного інтелекту, та й самі розробки були незначними. У 1970-ті були натяки на використання подібних технологій, зокрема спроби змоделювати неігрових персонажів, які динамічно реагують на гравця, проте правила їхньої поведінки були досить простими, якщо не сказати: примітивними. Тільки з 2000-х років у

цифрову епоху почалася більш менш продуктивна взаємодія дослідників штучного інтелекту та розробників комп'ютерних ігор.

Подібна кооперація не позбавлена низки труднощів. Як зазначають автори книги з штучного інтелекту в іграх Ю. Тогеліус та Г. Янакакіс, можна зафіксувати очевидний розрив між дослідженнями та їх застосуваннями на практиці [4]. По-перше, дослідники та розробники вирішують різні завдання. Перші, якщо звертаються до проблеми навчання неігрових персонажів у реальному часі, не приділяють належної уваги наслідкам такого навчання (найчастіше їм хочеться досягти непередбачуваних, незвичайних результатів). Для розробників те, як поводиться персонаж по відношенню до гравця або гри, важливо, адже неадекватна поведінка, непередбачувана реакція тощо можуть порушити хід гри, зруйнувати ігровий процес. Розробники повинні більш-менш точно знати, яким буде результат навчання програм. По-друге, автори вважають, що технології штучного інтелекту важко адаптуються до середовища комп'ютерних ігор, оскільки основні механіки та ігрові патерни склалися в 1980-і та 1990-і, коли практично не було можливості застосовувати штучний інтелект в ігровій індустрії, а значить, ігри без впровадження подібних технологій добре справляються зі своїми завданнями. По-третє, якщо дослідники більше концентруються на завданнях, пов'язаних з поведінкою неігрових персонажів, їхньою правдоподібністю, то розробники ігрового штучного інтелекту цим не обмежуються. Найчастіше велику цінність для них представляють технології процедурного генерування контенту, які можуть полегшити виробництво ігор, але це мало цікавить вчених. Якщо No Man's Sky могла стати проривом для ігрової індустрії, то для дослідників AI ця гра набагато прозаїчніша. Незважаючи на труднощі, зараз все частіше можна спостерігати колаборацію між дослідниками та розробниками штучного інтелекту та дослідниками та розробниками комп'ютерних ігор [5].

Донедавна ресурсу настільних ігор вистачало для того, щоб відкривати дослідникам штучного інтелекту нові горизонти. Поступово машині вдалося обіграти людину в шашки, шахи, а в 2017 році – в го, одну із найскладніших та

багатоваріантних настільних ігор. Програма AlphaGo Master від Google DeepMind виграла матч із трьох ігор у найсильнішого, за версією рейтингу Remi Coulom, у світі гравця в го Ке Цзе (Ke Jie). Тепер настала черга комп'ютерних ігор ставити нові бар'єри для штучного інтелекту. Мова може йти як про однокористувацькі ігри, так і багато-користувацькі. Перші кроки в цьому напрямі вже зроблено: у 2016 році DeepMind спільно з Blizzard оголосили про створення платформи для розробки ШІ-систем для гри в StarCraft II (2010 р.) – одного з найрозвиненіших на сьогоднішній день майданчиків на кіберспортивній арені, а у 2018 році їхня програма AlphaStar змогла здобути перемогу в тестових матчах з рахунком «5–0» у гравця з ніком MaNa, одного з найкращих геймерів, які грають за расу протосів. Однак до «повної» перемоги ще далеко, тому що на даному етапі програма грає лише за одну расу та з деякими ігровими обмеженнями [6].

Описані вище завдання стоять швидше перед дослідниками штучного інтелекту в цілому, ігровий AI має свою специфіку і орієнтований більше не на симуляцію ігрового досвіду (коли програма замінює живого гравця), а на підтримку різних технологій, які можуть зробити гру цікавішою для гравця. Погляд розробника ігрового штучного інтелекту спрямований углиб гри, шукає вирішення внутрішньоігрових завдань. Основні напрямки:

– поведінка неігрових персонажів;
– оточення;
– метаігровий рівень.

Моделювання неігрових персонажів має на увазі не тільки кодування здатності «підтримати бесіду» (і зробити це правдоподібно, тут використовується нелінійність гілок діалогу, анімація обличчя тощо), а й побудова інтерактивної драми (персонаж реагує, у тому числі емоційно, на дії гравця), опис алгоритмів пошуку шляху неігрового персонажа (як йому поводитися в оточенні, як більш реалістично дістатися гравця або іншої мети), алгоритмів поведінки ворогів та союзників в іграх та ін.

Другий напрямок стосується генерування рівнів, предметів, способу впорядкування предметів на рівні, а також оточення в іграх загалом, яке стає все

більш «чуйним» до дій користувача. Для посилення реалістичності воно також має бути «розумним» і реагувати на гравця, змінюватися залежно від його активності. Створення подібної «чуйності» у технічному сенсі близьке до створення неігрового персонажа, який асоціюється частіше з живою сутністю (наприклад, гриб Гумба в серії ігор Mario; рослини у Plants vs. Zombies (2009 р.); анімовані предмети, що атакують гравця у Cuphead (2017 р)) [4].

Існує дуже тонка грань між програмною частиною гри та технологією штучного інтелекту, адже дерево рішень лежить в основі багатьох ігрових елементів. Це запорука інтерактивності гри. Сприйняття, інтелект, інстинкт, розум, перцепція – складні і неоднозначні у визначенні, тому перенесення в середовище алгоритмів і обчислень лише посилює невизначеність.

Проблема не вирішується і в рамках третього напряму розробки ігрового штучного інтелекту: метаігрового рівня, який має на увазі роботу з камерою, наприклад адаптивність камери в Heavy Rain (2010 р.), або інтерактивний наратив, як у Dear Esther (2012 р.). Даний напрямок відповідає за підстроювання гри до гравця, оперування незаскриптованими моментами, тим, як гра може підлаштовуватися під конкретного гравця, змінювати свою структуру або елементи, «пам'ятати» про його вибори тощо. Ця класифікація не може вважатися вичерпною, вона лише демонструє різноманіття сфер застосування ігрового штучного інтелекту.

Окремо варто виділити область, яка також активно розвивається: використання технологій big data та інших методів для збору та обробки різноманітної інформації про гравців та гру. Деякі результати аналізу доступні у відкритому доступі, наприклад статистика проекту OpenDota, деякі використовуються творцями ігор на етапі виробництва.

Виділяють такі завдання для штучного інтелекту в іграх, пов'язаних з поведінкою гравця [2]:

– розробка «розумного» та людиноподібного неігрового персонажа для кращої інтеракції з гравцями;

– передбачення поведінки гравців, щоб удосконалити процес тестування гри та геймдизайну в цілому;

– класифікація поведінки гравців, щоб задіяти персоналізацію гри;

– відстеження патернів, що найчастіше зустрічаються чи послідовностей дій, визначення поведінки гравця у грі.

1.2 Системи ігрового штучного інтелекту

Термін «штучний інтелект» використовується для позначення великого напряму наукових та прикладних досліджень. Така назва, що закріпилася за цим напрямком, у більшості людей швидше асоціюється з розумними роботами або комп'ютерами, що мислять, численні образи яких були створені в науково-фантастичних творах.

Самі дослідники виділяють дві основні цілі своєї роботи – це [7]:

– автоматизація людської діяльності, особливо тих її видів, які зазвичай вважалися інтелектуальними;

– створення комп'ютерних моделей, що імітують процеси вирішення людиною тих чи інших інтелектуальних завдань з метою пояснення сутності цих процесів.

Область штучного інтелекту є вкрай неоднорідною. У ній існують різні напрями досліджень, які виділяються або за завданням (предметна область), що вимагає інтелектуального аналізу, або за інструментарієм, або за розроблюваною моделлю мислення.

До напрямків, що виділяються на основі розв'язуваного завдання, відносяться:

– машинний переклад;

– автоматичне реферування та інформаційний пошук;

– системи мовного спілкування;

– ігровий інтелект, доказ теорем та автоматизація наукових досліджень;

– комп'ютерний зір;
– вилучення даних;
– створення текстів та музики та ін.

Перелічені напрями характеризуються тим, що значна частина проведених досліджень присвячена не процесам мислення, а предмету інтелектуального аналізу.

Напрямки в ШІ, що виділяються за інструментарієм, включають [7]:

– штучні нейронні мережі;
– еволюційні обчислення;
– розпізнавання образів;
– експертні системи;
– евристичне програмування;
– мультиагентний підхід тощо.

Відмінність даних напрямів у тому, що у них розвивається апарат розв'язання великого класу завдань. Наприклад, способи розпізнавання образів можна застосовувати при категоризації текстів у інформаційному пошуку, у системах мовного спілкування, комп'ютерного зору та інших напрямах першого типу. У той самий час, на вирішення одного завдання можуть застосовуватися різні методи. Наприклад, нейронні мережі здатні вирішувати ті завдання, які вирішуються методами розпізнавання образів, а еволюційні обчислення можуть заміняти методи евристичного програмування і навпаки.

Дослідження в області ШІ почалися з парадигми «мислення як пошук» та з розробки методів вирішення формально поставлених завдань. Подальша зміна парадигм була пов'язана зі збільшенням універсальності машинних систем, завдяки зменшенню обсягу інформації, що готується їм людиною. Якщо на першому етапі розвитку ШІ опис кожного завдання формувалося людиною, то на другому етапі людина вже задавала опис деякої (досить вузької) предметної області, що включає цілий комплекс завдань. На третьому етапі машинна система

отримує можливість, принаймі частково, будувати опис предметної області самостійно в рамках заданого людиною представлення [8].

На рис. 1.3 представлена схема, що умовно зображує структуру базового рівня області штучного інтелекту.

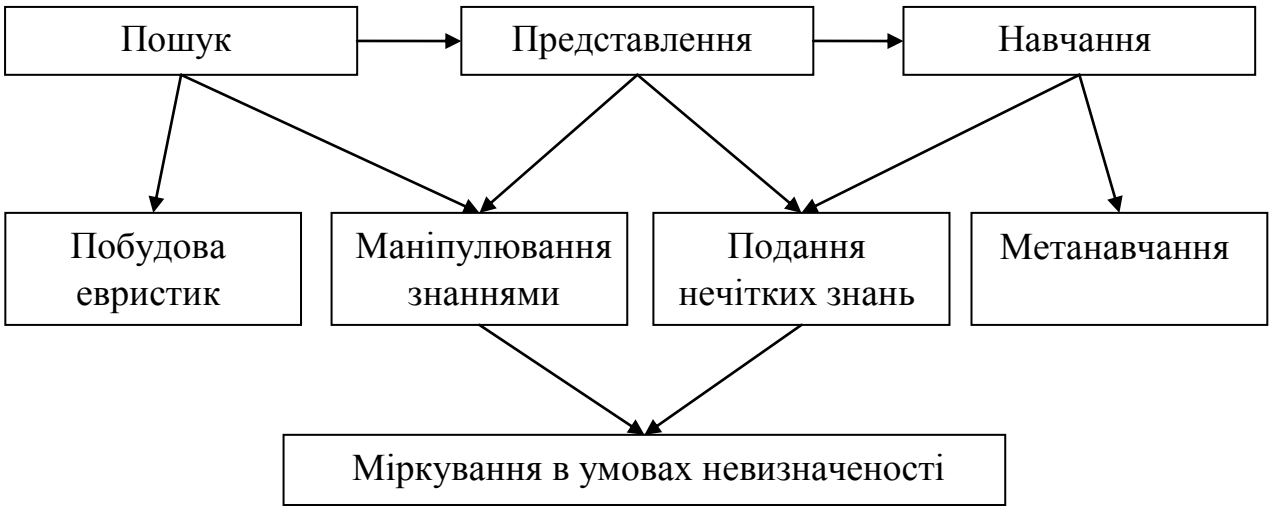

Рисунок 1.3 – Структура базового рівня галузі штучного інтелекту [8]

Подальший розвиток галузі ШІ пов'язаний з подальшою універсалізацією машинних систем та отриманням більш широкого доступу до інформації. Останнє може бути пов'язане з напрямком, у якому досліджуються втілені системи, тобто системи, вміщені у конкретне інформаційне, фізичне, соціальне оточення. Так, згідно з позицією Р. Брукса, інтелект не може виникнути в невтілених системах, таких як традиційні системи доказу теорем або експертні системи. Окрім самої цієї тези вчений також пропонує і конкретну багатошарову архітектуру, що складається з простих утворень, що взаємодіють, і названу ним категоріальною архітектурою, для керування роботами. Цей напрямок не є принципово новим, оскільки багато чого запозичує з робототехніки. Однак у втілених системах мається на увазі, що сенсорна інформація, що надходить, повинна служити основою навчання, в результаті якого повинна формуватися система знань з метою їх використання для подальшого вирішення поставлених завдань. Проблема навчання «з нуля» на основі сенсорної інформації ставить безліч

додаткових проблем, вирішення яких ще потрібно шукати в майбутньому. Ймовірно, сучасний стан в галузі ШІ можна охарактеризувати як етап синтезу, на якому відбувається поєднання методів, отриманих раніше у межах ізольованих напрямків досліджень.

1.3 Теоретичні основи евристичного програмування

Першим домінуючим напрямом в області ШІ був напрямок, який зазвичай позначається терміном «евристичне програмування», в якому була прийнята метафора, що мислення – це пошук у просторі рішень. Даний напрямок тісно пов'язаний з лабіринтною гіпотезою мислення в психології та дослідженнями формалізації процесу доказу теорем у математиці.

Слово «евристичний» означає «що сприяє відкриттю». Таким чином, проводиться паралель між евристичним програмуванням та науковим пошуком, що наголошує на нетривіальності проблеми пошуку та її значущості для розуміння природи інтелекту.

Роботи у напрямі евристичного програмування призвели до формування власної термінології та методів дослідження.

Центральним поняттям в евристичному програмуванні є поняття дерева варіантів (простору станів), яке зазвичай набуває форми дерева гри чи дерева цілей. Корінь дерева є вихідним станом, з якого виходять гілки, що відповідають тому, як цей стан може бути змінено. Листям дерева, які мають нащадків, відповідають стани, для яких виконується критерій закінчення. У дереві цілей кореню, як правило, відповідає ціль, а дочірнім вузлам – підцілі, досягнення яких необхідне або достатнє для досягнення мети [9].

На рис. 1.4 і 1.5 наведено приклади фрагментів дерева гри та дерева цілей відповідно. Розв'язання деякої задачі зводиться до знаходження листа, що задовольняє умовам задачі і побудові шляху від кореня дерева до цього листа. Усі можливі шляхи від кореня до листя становлять простір рішень.

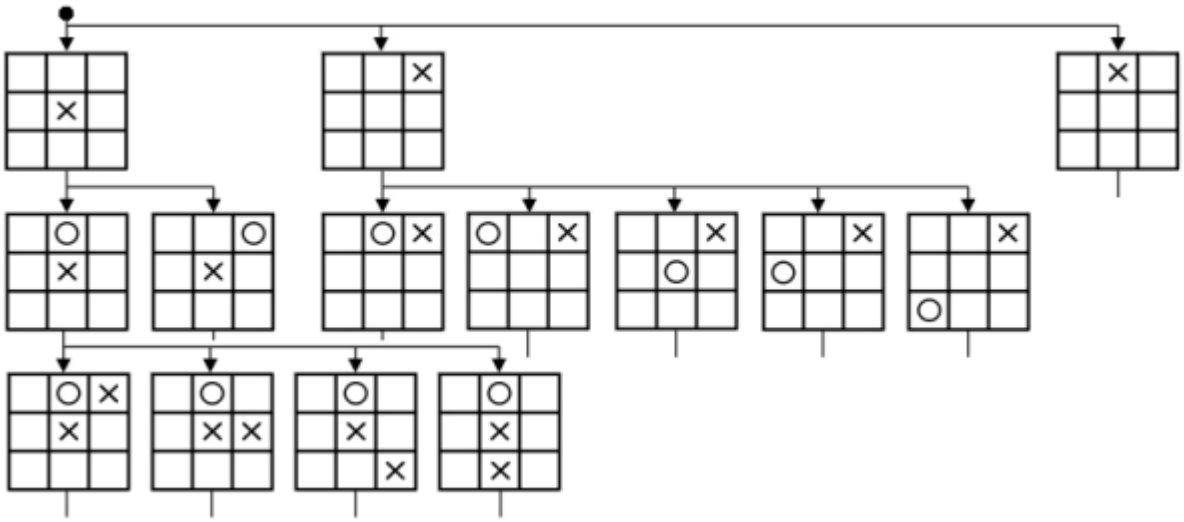

Рисунок 1.4 – Приклад фрагменту дерева гри («Хрестики-нолики»)

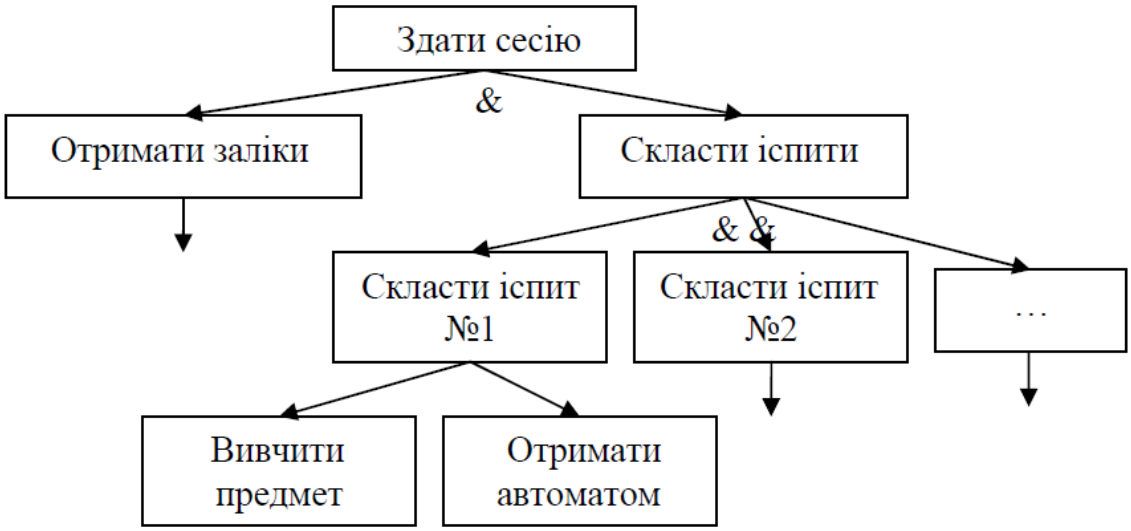

Рисунок 1.5 – Приклад фрагменту дерева цілей

Альтернативним уявленням є граф станів. Якщо в дереві станів одному й тому ж стану можуть відповідати різні вузли залежно від способу досягнення цього стану, то в графі станів кожному стану відповідає лише один вузол, проте до нього може входити кілька дуг, причому у такому графі можуть бути цикли. Подання у вигляді графа є компактнішим, але його аналіз значно складніший.

Дерева гри та цілей можуть бути експліцитними та імпліцитними. Експліцитне дерево задається у явному вигляді шляхом вказівки всіх його вузлів

та дуг, у той час як імпліцитне дерево задається шляхом вказівки вихідної позиції та правил формування дерева (правил гри або допустимих операцій із зміни поточного стану).

Вирішення завдання пошуку на експліцитному дереві тривіальне: достатньо лише з листя дерева вибрати те, що підходить. Проте експліцитні дерева у реальних завданнях зустрічаються рідко [10].

Процедура, яка перетворює імпліцитне дерево в експліцитне, називається процедурою, що породжує. У класичному евристичному програмуванні вважається, що така процедура, що породжує, існує (і детермінована) і вона відома. Іншими словами, конкретна дія в конкретній ситуації завжди призводить до однієї й тієї ж відомої зміни цієї ситуації.

Оскільки дерево в імпліцитній формі є як би невидимим для програми і програма може побудувати («побачити») тільки частину експліцитного дерева, при фіксованих витратах обчислювальних ресурсів, цей процес називають пошуком. Тут нескладно простежити аналогію між пошуком рішення на імпліцитному дереві та пошуком шляху у реальному лабіринті, де огляд також обмежений.

Оскільки все дерево породжене бути не може, стає важливим, у якому порядку відбувається породження вузлів (чи у якому порядку здійснюється обхід вузлів імпліцитного дерева). Залежно від цього порядку процедури, що породжують, можна розділити на кілька типів. Дві стратегії пошуку, що найбільш сильно відрізняються – це пошук в глибину і пошук в ширину.

При пошуку в глибину на кожному кроці відвідується один із ще не розглянутих нащадків поточного вузла (зазвичай найлівіший). Якщо не відвіданих нащадків немає, то провадиться повернення до батьківського вузла, а поточний вузол позначається як вивчений. При пошуку в ширину спочатку відвідуються всі вузли, що знаходяться на одній глибині з поточним вузлом, а потім здійснюється перехід до їх нащадків, що знаходяться на наступному рівні глибини. На рис. 1.6 представлені схеми обходу в глибину та ширину [10].

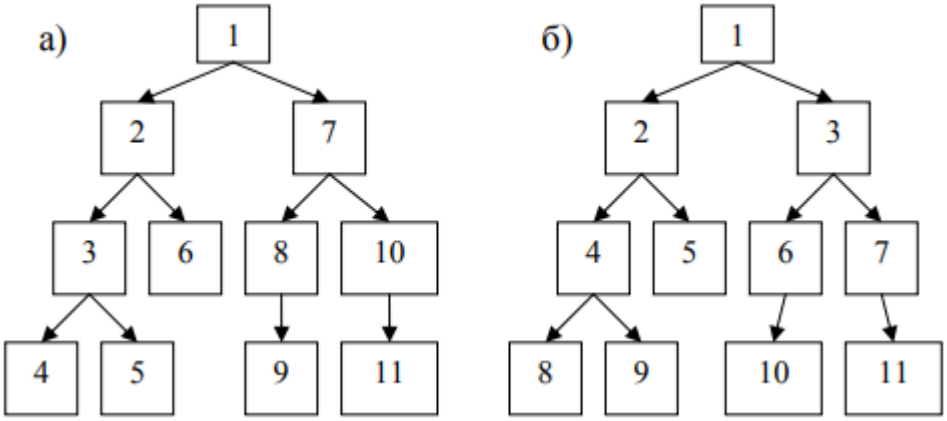

Рисунок 1.6 – Порядок обходу вузлів дерева під час пошуку (а) у глибину; (б) завширшки

Оскільки порядок обходу дерева варіантів пошуку в глибину і ширину різниться, за наявності обмеження на кількість відвіданих вузлів з використанням цих двох стратегій пошуку різні вузли залишаться не відвіданими.

Ще один найчастіше використовуваний метод перебору – це породження комбінаторних об'єктів у лексикографічному порядку, тобто, введення лінійного порядку на безлічі варіантів та перерахування всіх варіантів у цьому порядку. Цей спосіб зручний тим, що якщо дерево варіантів вдається звести до одного з типів комбінаторних об'єктів, це може суттєво спростити процедуру пошуку. Розглянемо кілька типових завдань.

1) Перерахування всіх бінарних рядків довжини N.

Приклад перерахування в лексикографічному порядку N=3:

000, 001, 010, 011, 100, 101, 110, 111.

2) Перерахування всіх перестановок символів у рядку довжини N.

Приклад перерахування в лексикографічному порядку N=3.

123, 132, 213, 231, 312, 321.

3) Перерахування всіх K-елементних підмножин множини з N елементів.

Приклад N=5, K=3.

00111, 01011, 01101, 01110, 10011, 10101, 10110, 11001, 11010, 11100.

Самі алгоритми породження комбінаторних об'єктів складаються з процедури генерації першого об'єкта та процедури переходу від n-го до (n+1) об'єкта [9].

Однак при породженні об'єктів у лексикографічному порядку проблематичним виявляється введення евристик, що відсікають неперспективні гілки, оскільки ці об'єкти, як правило, відповідають листям на дереві варіантів, і процедура породження переходить від листа до листа, минаючи некінцеві вузли на дереві (рис.1.7).

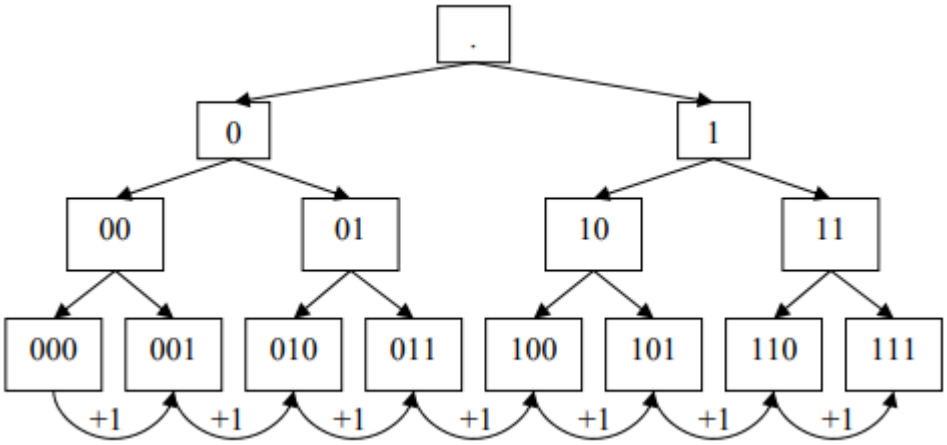

Рисунок 1.7 – Відмінність процедури обходу дерева та породження кінцевих об'єктів у лексикографічному порядку

Оскільки найчастіше все листя досягнуто бути не може, виявляється необхідним визначати вузли, з яких немає необхідності породжувати дерево варіантів. Центральним поняттям евристичного програмування є евристика – спосіб або прийом скорочення перебору, відсікання неперспективних гілок на дереві варіантів.

1.4 Постановка задачі

Розробити гру «Шахи» з використанням технології штучного інтелекту.

2. РОЗРОБКА КОМП'ЮТЕРНОЇ ГРИ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Вибір мови прорамування

JavaScript – це динамічна мова комп'ютерного програмування. Вона легка і найчастіше використовується як частина веб-сторінок, чиї реалізації дозволяють клієнтському сценарію взаємодіяти з користувачем і створювати динамічні сторінки. Це інтерпретована мова програмування з об'єктно-орієнтованими можливостями [11].

Вперше JavaScript з’явилась в Netscape 2.0 у 1995 році під назвою LiveScript. Основне ядро мови загального призначення було вбудовано в Netscape, Internet Explorer та інші веб-браузери.

Клієнтський JavaScript є найпоширенішою формою мови. Сценарій повинен бути включений в HTML-документ або посилання на нього, щоб код інтерпретував браузер.

Це означає, що веб-сторінка не обов’язково має бути статичним HTML, але може включати програми, які взаємодіють з користувачем, керують браузером і динамічно створюють вміст HTML.

Механізм на стороні клієнта JavaScript надає багато переваг перед традиційними серверними сценаріями CGI. Наприклад, можливо використовувати JavaScript, щоб перевірити, чи ввів користувач дійсну адресу електронної пошти в поле форми.

Код JavaScript виконується, коли користувач надсилає форму, і лише якщо всі записи дійсні, вони будуть передані на веб-сервер.

JavaScript можна використовувати для захоплення ініційованих користувачем подій, таких як натискання кнопок, навігація за посиланнями та інші дії, які користувач ініціює явно чи неявно.

Переваги використання JavaScript [11]:

– Менше взаємодії з сервером: можна перевірити введення користувача перед відправкою сторінки на сервер. Це економить трафік сервера, що означає менше навантаження на сервер.

– Негайний відгук відвідувачам: їм не потрібно чекати перезавантаження сторінки, щоб побачити, чи не забули вони щось ввести.

– Підвищена інтерактивність: є можливість створювати інтерфейси, які реагують, коли користувач наводить на них курсор миші або активує їх за допомогою клавіатури.

– Розширені інтерфейси: можнна використовувати JavaScript, щоб включити такі елементи, як компоненти перетягування та повзунки, щоб надати відвідувачам сайту багатий інтерфейс.

Ми не можемо розглядати JavaScript як повноцінну мову програмування. У ній відсутні такі важливі характеристики [11]:

– JavaScript на стороні клієнта не дозволяє читати або записувати файли. Це було збережено з міркувань безпеки.

– JavaScript не можна використовувати для мережевих програм, оскільки така підтримка недоступна.

– JavaScript не має багатопоточних чи багатопроцесорних можливостей.

Знову ж таки, JavaScript – це легка інтерпретована мова програмування, яка дозволяє вбудувати інтерактивність у статичні сторінки HTML.

JavaScript можна реалізувати за допомогою операторів JavaScript, які розміщуються в тегах HTML <script>... </script> на веб-сторінці.

Можна розмістити теги <script>, що містять JavaScript, будь-де на веб-сторінці, але зазвичай рекомендується зберігати їх у тегах <head>.

Тег <script> сповіщає програму браузера почати інтерпретувати весь текст між цими тегами як сценарій. Простий синтаксис вашого JavaScript буде виглядати наступним чином:

```
<script ...>
  JavaScript code
</script>
```

Тег script має два важливі атрибути [11]:

– Мова: цей атрибут визначає, яка мова сценаріїв використовується. Як правило, його значення буде javascript. Хоча в останніх версіях HTML (і XHTML, його наступника) використання цього атрибута було припинено.

– Тип: цей атрибут тепер рекомендований для вказівки мови сценаріїв, що використовується, і його значення має бути встановлено на "text/javascript".

Отже, синтаксис JavaScript буде виглядати наступним чином:

```
<script language="javascript" type="text/javascript">
  JavaScript code
</script>
```

Усі сучасні браузери мають вбудовану підтримку JavaScript: Internet Explorer, Firefox, Chrome та Opera.

Існує можливість включати код JavaScript у будь-яке місце в HTML-документі. Однак найбільш переважними способами включення JavaScript у файл HTML є такі:

– Сценарій у розділі <head>...</head>.

– Сценарій у розділі <body>...</body>.

– Сценарій у розділах <body>...</body> і <head>...</head>.

– Написаний сценарій у зовнішньому файлі, а потім включений в розділ <head>...</head>.

Однією з найбільш фундаментальних характеристик мови програмування є набір типів даних, які вона підтримує. Це типи значень, які можна представити та маніпулювати на мові програмування [12].

JavaScript дозволяє працювати з трьома примітивними типами даних:

– Числа, наприклад, 123, 120,50 тощо.

– Рядки тексту, наприклад, "Цей текстовий рядок" тощо.

– Логічне значення, наприклад, true чи false.

JavaScript також визначає два тривіальних типи даних, null і undefined, кожен з яких визначає лише одне значення. На додаток до цих примітивних типів даних, JavaScript підтримує складений тип даних, відомий як об’єкт.

Як і багато інших мов програмування, JavaScript має змінні. Змінні можна розглядати як іменовані контейнери. Можна помістити дані в ці контейнери, а потім звернутися до даних, просто назвавши контейнер.

Область дії змінної – це область програми, в якій вона визначена. Змінні JavaScript мають лише дві області дії:

– Глобальні змінні: глобальна змінна має глобальну область, що означає, що її можна визначити в будь-якому місці коду JavaScript.

– Локальні змінні: локальна змінна буде видима лише в межах функції, де вона визначена. Параметри функції завжди локальні для цієї функції.

Функція — це група багаторазового коду, який можна викликати в будь-якому місці програми. Це усуває необхідність писати той самий код декілька раз. Це допомагає програмістам писати модульні коди. Функції дозволяють програмісту розділити велику програму на ряд невеликих і керованих функцій.

Як і будь-яка інша просунута мова програмування, JavaScript також підтримує всі функції, необхідні для написання модульного коду за допомогою функцій.

2.2 Алгоритм Minimax та альфа-бета обрізка

Minimax – це правило прийняття рішень, яке використовується в штучному інтелекті, теорії рішень, теорії ігор, статистиці та філософії для мінімізації можливих втрат у гіршому випадку (максимальні втрати) [13]. Коли йдеться про виграші, це називають «maximin» – максимізувати мінімальний приріст. Спочатку сформульована для теорії гри з нульовою сумою для n гравців, яка охоплює як випадки, коли гравці роблять альтернативні ходи, так і ті, коли вони роблять одночасні ходи, вона також була поширена на більш складні ігри та на загальне прийняття рішень за наявності невизначеності.

Максимальне значення — це найвище значення, яке гравець може отримати, не знаючи дій інших гравців; тобто, це найменше значення, яке інші гравці можуть змусити отримати гравця, коли знають про дії гравця. Його формальне визначення наступне [13]:

$$\underline{v_i} = \max_{a_i} \min_{a_{-i}} v_i(a_i, a_{-i}) \quad , \tag{2.1}$$

де:

i – індекс гравця, який цікавить.

$-i$ – позначає всіх інших гравців, крім гравця і.

a_i – це дія гравця і.

a_{-i} позначає дії всіх інших гравців.

v_i – функція значення гравця і.

Розрахунок максимального значення гравця виконується в найгіршому випадку: для кожної можливої дії гравця перевіряються всі можливі дії інших гравців і визначається найгірша можлива комбінацію дій – та, яка дає гравцеві i найменше значення. Потім визначається, яку дію гравця можна виконати, щоб переконатися, що це найменше значення є найвищим із можливих.

Мінімальне значення гравця – це найменше значення, яке інші гравці можуть змусити отримати, не знаючи дій гравця; тобто, це найбільше значення, яке гравець може отримати, коли знає дії інших гравців. Його формальне визначення наступне [13]:

$$\overline{v_i} = \min_{a_{-i}} \max_{a_i} v_i(a_i, a_{-i}) \quad , \tag{2.2}$$

Визначення дуже схоже на визначення максимального значення — лише порядок операторів максимуму та мінімуму є зворотним.

Алгоритм Minimax – це рекурсивний алгоритм вибору наступного ходу в грі з n гравцями, як правило, для двох гравців. Цінність пов’язана з кожною позицією або станом гри. Це значення обчислюється за допомогою функції оцінки позиції і вказує, наскільки добре гравцеві було б досягти цієї позиції. Потім гравець робить хід, який максимізує мінімальне значення позиції в результаті можливих

наступних ходів суперника. Якщо настає черга A рухатися, A дає значення кожному з своїх законних ходів.

Можливий метод розподілу полягає в тому, щоб призначити певний виграш для A як +1, а для B як −1. Це призводить до комбінаторної теорії ігор, розробленої Джоном Хортоном Конвеєм. Альтернативою є використання правила, згідно з яким, якщо результатом ходу є негайний виграш для A, йому призначається додатна нескінченність, а якщо це негайний виграш для B, то негативна нескінченність. Значення A будь-якого іншого ходу є максимальним значенням, отриманим у результаті кожної з можливих відповідей B. З цієї причини A називають максимізуючим гравцем, а B називають мінімізуючим гравцем, звідси і назва алгоритму Minimax.

Альфа-бета обрізка — це алгоритм пошуку, який прагне зменшити кількість вузлів, які оцінюються алгоритмом Minimax у своєму дереві пошуку. Це алгоритм змагального пошуку, який зазвичай використовується для машинної гри в іграх для двох гравців (хрестики-нолики, шахи тощо). Він припиняє оцінку ходу, коли знайдено хоча б одну можливість, яка доводить, що хід гірший, ніж раніше розглянутий хід. Такі кроки не потрібно оцінювати далі. При застосуванні до стандартного дерева Minimax він повертає той самий рух, що й Minimax, але обрізає гілки, які не можуть вплинути на остаточне рішення [14].

Дерево ігор може представляти багато ігор з нульовою сумою для двох гравців, таких як шахи, шашки та реверси. Кожен вузол дерева представляє можливу ситуацію в грі. Кожному кінцевому вузлу (результату) гілки присвоюється числова оцінка, яка визначає значення результату для гравця з наступного ходу.

Алгоритм підтримує два значення, альфа і бета, які відповідно представляють мінімальний рахунок, який гарантується гравцеві, що максимізує, і максимальний рахунок, який гарантується гравцеві, що мінімізує. Спочатку альфа – це негативна нескінченність, а бета – позитивна нескінченність, тобто обидва гравці починають з найгіршого можливого результату. Щоразу, коли максимальний рахунок, який гарантується гравцеві, що мінімізує (тобто, гравцеві

«бета»), стає меншим за мінімальний рахунок, який гарантується гравцеві, що набирає максимум (тобто, гравцеві «альфа») (тобто бета<альфа), гравцеві не потрібно розглядати подальших нащадків цього вузла, оскільки вони ніколи не будуть досягнуті під час реальної гри.

3. ДОСЛІДЖЕННЯ РОЗРОБЛЕНОЇ КОМП'ЮТЕРНОЇ ГРИ

3.1. Алгоритм дослідження гри

Алгоритм розробки розробленої комп’ютерної гри:

– генерація ходів та візуалізації дошки;

– оцінка позиції;

– пошук дерева за допомогою Minimax;

– альфа-бета обрізка.

1) Генерація ходів та візуалізація дошки.

Для реалізації генерації ходів була використана бібліотека chess.js, а chessboard.js – для візуалізації дошки.

Chessboard.js є окремою шаховою дошкою JavaScript. Він розроблений як «просто дошка» і надає потужний API, щоб його можна було використовувати різними способами. Ось неповний список речей, які можливо робити з chessboard.js:

– Використовувати chessboard.js, щоб відображати ігрові позиції разом зі своїми експертними коментарями.

– Використовувати chessboard.js, щоб мати тактичний веб-сайт, де користувачі повинні вгадати найкращий хід.

– Створити шаховий сервер і дозволити користувачам грати в свої ігри за допомогою дошки chessboard.js.

Chessboard.js досить гнучкий, щоб з відносною легкістю впоратися з будь-якою з цих ситуацій.

Бібліотека chess.js в основному реалізує всі правила шахів. chess.js — це шахова бібліотека Javascript, яка використовується для генерації/перевірки шахових ходів, розміщення/переміщення фігур і виявлення шахів/матів/пату – в основному все, крім ШІ.

chess.js був ретельно протестований у node.js та більшості сучасних браузерів.

На основі цього були розраховані всі ходи для даного стану дошки.

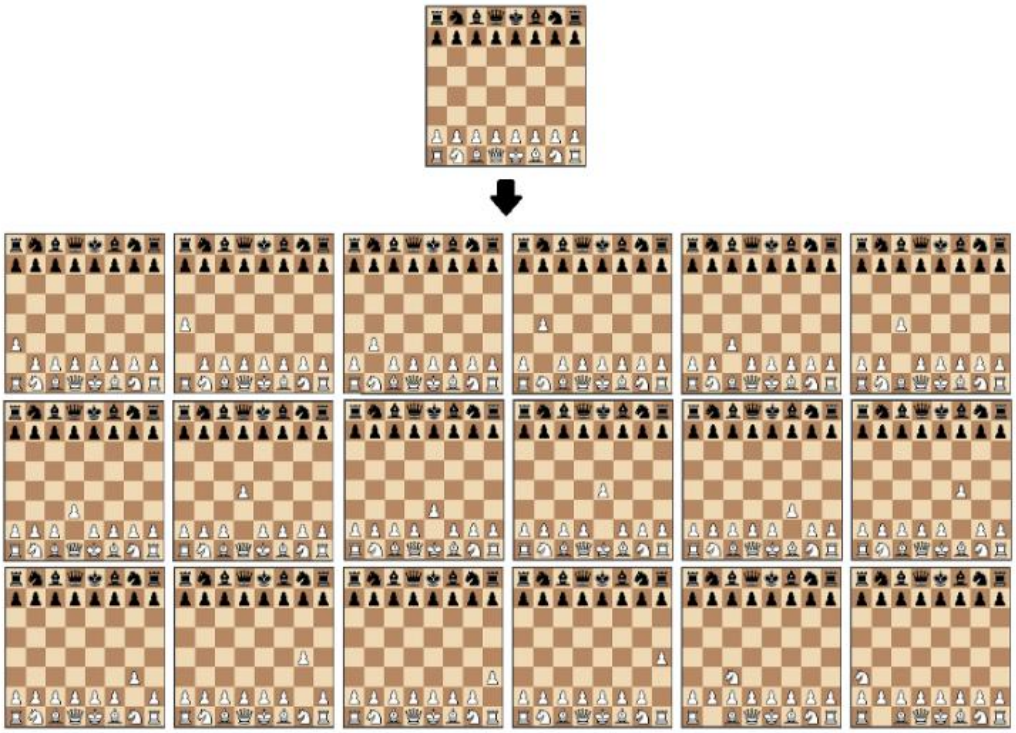

Рисунок 2.1 – Візуалізація функції генерації руху. Початкова позиція використовується як вхідні дані, а вихідними є всі можливі ходи з цієї позиції

Використання даних бібліотек допомагає зосередитися лише на важливому завданні цього дослідження: створенні алгоритму штучного інтелекту, який знаходить найкращий хід.

Спочатку створюємо функцію, яка просто повертає випадковий хід з усіх можливих ходів:

```
1   var calculateBestMove =function(game) {
2       //generate all the moves for a given position
3       var newGameMoves = game.ugly_moves();
4       return newGameMoves[Math.floor(Math.random() * newGameMoves.length)];
5   };
```

Хоча цей алгоритм не являється сильним гравцем у шахи, але він є відправною точкою для розробки, оскільки користувач може грати проти нього.

2) Оцінка позиції.

Для визначення того, яка сторона сильніша в конкретній позиції потрібно підрахувати відносну міцність фігур на шаховій дошці (рис. 2.2).

Рисунок 2.2 – Відносна міцність фігур на дошці

За допомогою функції оцінки позиції створено алгоритм, який вибирає хід, що дає найвищу оцінку:

```
var calculateBestMove = function (game) {

    var newGameMoves = game.ugly_moves();
    var bestMove = null;
    //use any negative large number
    var bestValue = -9999;

    for (var i = 0; i < newGameMoves.length; i++) {
        var newGameMove = newGameMoves[i];
        game.ugly_move(newGameMove);

        //take the negative as AI plays as black
        var boardValue = -evaluateBoard(game.board())
        game.undo();
        if (boardValue > bestValue) {
            bestValue = boardValue;
            bestMove = newGameMove
        }
    }

    return bestMove;

};
```

Єдине відчутне покращення полягає в тому, що алгоритм тепер захоплюватиме фрагмент, якщо зможе.

Чорні грають за допомогою простої функції оцінки.

3) Пошук дерева за допомогою Minimax.

Далі створено дерево пошуку, з якого алгоритм зможе вибрати найкращий хід. Це робиться за допомогою алгоритму Minimax.

У цьому алгоритмі рекурсивне дерево всіх можливих ходів досліджується на задану глибину, а позиція оцінюється в кінцевих «листках» дерева:

```
var minimax = function (depth, game, isMaximisingPlayer) {
    if (depth === 0) {
        return -evaluateBoard(game.board());
    }
    var newGameMoves = game.ugly_moves();
    if (isMaximisingPlayer) {
        var bestMove = -9999;
        for (var i = 0; i < newGameMoves.length; i++) {
            game.ugly_move(newGameMoves[i]);
            bestMove = Math.max(bestMove, minimax(depth - 1, game, !isMaximisingPlayer));
            game.undo();
        }
        return bestMove;
    } else {
        var bestMove = 9999;
        for (var i = 0; i < newGameMoves.length; i++) {
            game.ugly_move(newGameMoves[i]);
            bestMove = Math.min(bestMove, minimax(depth - 1, game, !isMaximisingPlayer));
            game.undo();
        }
        return bestMove;
    }
};
```

Після цього повертаємо або найменше, або найбільше значення до батьківського вузла, залежно від того, білий він чи чорний для переміщення. Тобто намагаємося або мінімізувати, або максимізувати результат на кожному рівні (рис.2.3).

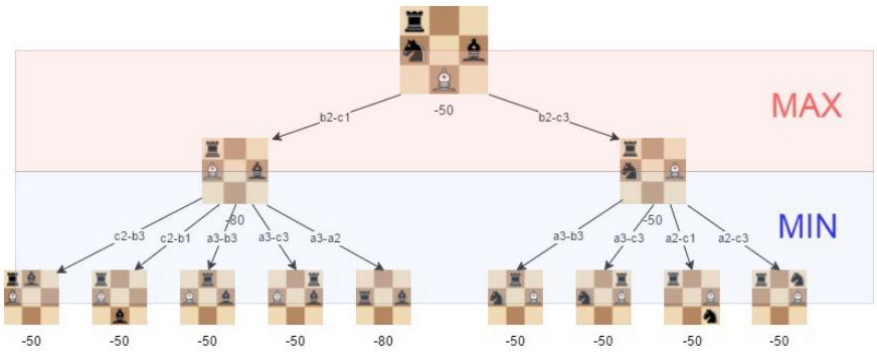

Рисунок 2.3 – Візуалізація алгоритму Minimax в штучному положенні (найкращий хід для білих – b2-c3, тому що є гарантія, що зможе дістатися до позиції, де оцінка -50)

З Minimax, розроблений алгоритм починає розуміти деякі основні тактики гри в шахи.

Ефективність алгоритму Minimax значною мірою залежить від глибини пошуку, яку можемо досягти. Це те, що покращиться на наступному кроці.

4) Альфа-бета обрізка.

Альфа-бета-обрізка – це метод оптимізації алгоритму Minimax, який дозволяє не враховувати деякі гілки в дереві пошуку. Це допомагає оцінити дерево пошуку Minimax набагато глибше, використовуючи ті самі ресурси.

Альфа-бета обрізка заснована на ситуації, коли можливо припинити оцінку частини дерева пошуку, якщо знайдений рух, який призводить до гіршої ситуації, ніж раніше виявлений рух (рис.2.4).

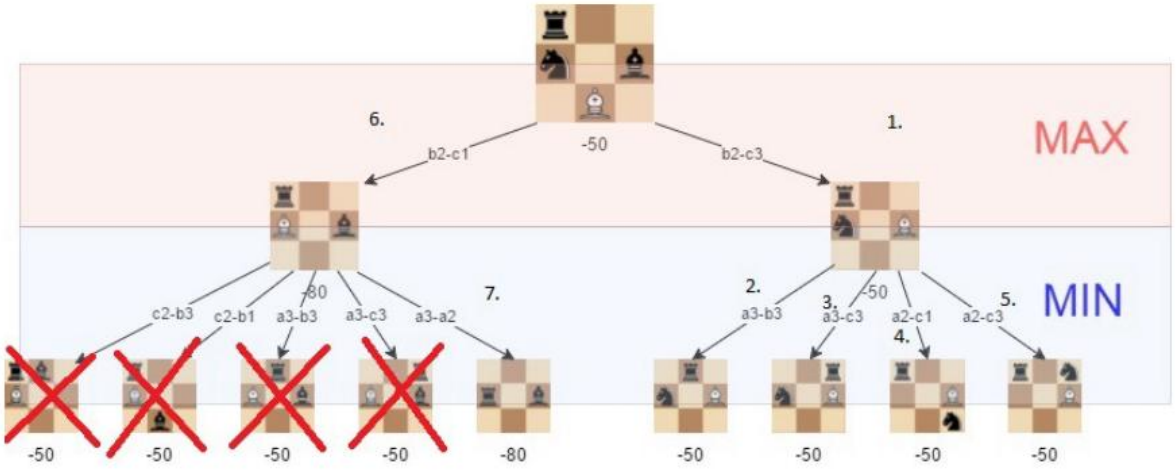

Рисунок 2.4 – Позиції, які не потрібно досліджувати

Альфа-бета обрізка не впливає на результат алгоритму Minimax – вона лише робить його швидшим.

Альфа-бета алгоритм також є ефективнішим, якщо спочатку відвідаємо ті шляхи, які ведуть до хороших ходів.

5) Покращена функція оцінки.

Функція початкової оцінки досить наївна, оскільки враховується лише той матеріал, який знайдено на дошці. Щоб покращити це, додано до оцінки фактор, який враховує положення фігур. Наприклад, кінь у центрі дошки краще (оскільки він має більше варіантів і, таким чином, більш активний), ніж лицар, що знаходиться на краю дошки.

Візуалізовані квадратні таблиці представлені на рис.2.5. Є можливість зменшити або збільшити оцінку, в залежності від місця розташування.

```
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -2.0, -3.0, -3.0, -4.0, -4.0, -3.0, -3.0, -2.0],
[ -1.0, -2.0, -2.0, -2.0, -2.0, -2.0, -2.0, -1.0],
[  2.0,  2.0,  0.0,  0.0,  0.0,  0.0,  2.0,  2.0 ],
[  2.0,  3.0,  1.0,  0.0,  0.0,  1.0,  3.0,  2.0 ]
```

```
[ -2.0, -1.0, -1.0, -0.5, -0.5, -1.0, -1.0, -2.0],
[ -1.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -1.0],
[ -1.0,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -1.0],
[ -0.5,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -0.5],
[  0.0,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -0.5],
[ -1.0,  0.5,  0.5,  0.5,  0.5,  0.5,  0.0, -1.0],
[ -1.0,  0.0,  0.5,  0.0,  0.0,  0.0,  0.0, -1.0],
[ -2.0, -1.0, -1.0, -0.5, -0.5, -1.0, -1.0, -2.0]
```

```
[  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0],
[  0.5,  1.0,  1.0,  1.0,  1.0,  1.0,  1.0,  0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[  0.0,   0.0, 0.0,  0.5,  0.5,  0.0,  0.0,  0.0]
```

```
[ -2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0],
[ -1.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -1.0],
[ -1.0,  0.0,  0.5,  1.0,  1.0,  0.5,  0.0, -1.0],
[ -1.0,  0.5,  0.5,  1.0,  1.0,  0.5,  0.5, -1.0],
[ -1.0,  0.0,  1.0,  1.0,  1.0,  1.0,  0.0, -1.0],
[ -1.0,  1.0,  1.0,  1.0,  1.0,  1.0,  1.0, -1.0],
[ -1.0,  0.5,  0.0,  0.0,  0.0,  0.0,  0.5, -1.0],
[ -2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0]
```

```
[-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0],
[-4.0, -2.0,  0.0,  0.0,  0.0,  0.0, -2.0, -4.0],
[-3.0,  0.0,  1.0,  1.5,  1.5,  1.0,  0.0, -3.0],
[-3.0,  0.5,  1.5,  2.0,  2.0,  1.5,  0.5, -3.0],
[-3.0,  0.0,  1.5,  2.0,  2.0,  1.5,  0.0, -3.0],
[-3.0,  0.5,  1.0,  1.5,  1.5,  1.0,  0.5, -3.0],
[-4.0, -2.0,  0.0,  0.5,  0.5,  0.0, -2.0, -4.0],
[-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0]
```

```
[0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0],
[5.0,  5.0,  5.0,  5.0,  5.0,  5.0,  5.0,  5.0],
[1.0,  1.0,  2.0,  3.0,  3.0,  2.0,  1.0,  1.0],
[0.5,  0.5,  1.0,  2.5,  2.5,  1.0,  0.5,  0.5],
[0.0,  0.0,  0.0,  2.0,  2.0,  0.0,  0.0,  0.0],
[0.5, -0.5, -1.0,  0.0,  0.0, -1.0, -0.5,  0.5],
[0.5,  1.0, 1.0,  -2.0, -2.0,  1.0,  1.0,  0.5],
[0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0]
```

Рисунок 2.5 – Візуалізовані квадратні таблиці

З наступним удосконаленням отримано алгоритм на основі технологій штучного інтелекту, який гідно грає в шахи (рис.2.6).

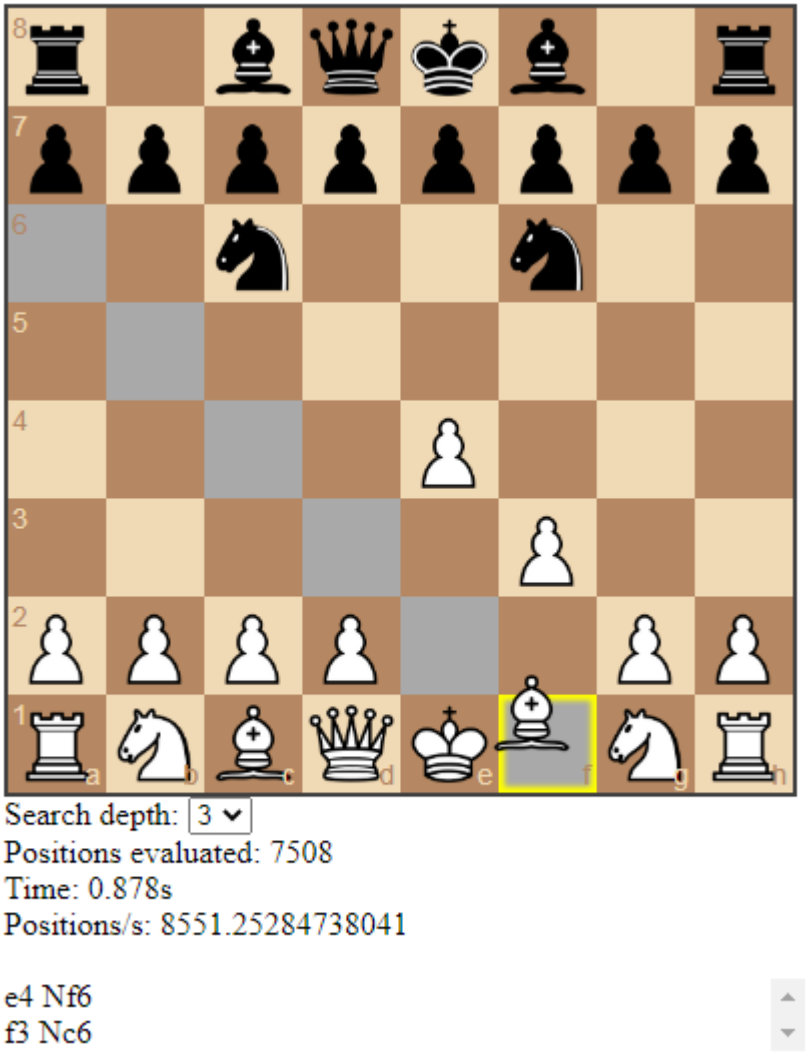

Рисунок 2.6 – Інтерфейс гри в шахи з використанням технологій штучного інтелекту

Фігури розташовуються на шахівниці, яка поділена на 64 клітини чорного і білого кольору порівну.

По горизонталі дошка позначена латинськими літерами від a до h, по вертикалі цифрами від 1 до 8. Ці цифри і літери служать для запису шахової партії. Перед початком партії «чорні» та «білі» мають однакову кількість фігур.

3.2. Оцінка ефективності гри

Спочатку поглянемо на вже створений теоретичний базис. За весь час було створено лише два основні методи оцінки:

1) Порівняльний метод – порівняння об'єктів між собою. Порівняти пару схожих ігор – дуже просто, будь-хто може це зробити, на ігрових форумах постійно це відбувається.

Але таке порівняння буде незрозумілим для тих, хто не бачив хоча б одну з порівнюваних ігор. Це порівняння лише для посвячених гравців. До того ж кожне таке порівняння буде унікальним, і це не дозволить створити систему.

Метод добре підходить для розважальних цілей. Можна влаштовувати суперечки, голосування, порівнюючи схожі проекти. Можна навіть скласти якийсь рейтинг, але для цього доведеться порівняти кожну гру з усіма іншими іграми по черзі. На це потрібно витратити багато часу, а в результаті вийде лише набір оцінок «краще, гірше, приблизно однакові». Такий результат не підходить для даного дослідження.

2) Абсолютний спосіб – порівняння з еталоном.

Абсолютний метод найбільше підходить для системного підходу. Він активно використовується в науці. Цим способом вимірюється: відстань – еталонним метром, вага – еталонним кілограмом.

Абсолютним методом користуються і багато ігрових сайтів, ставлячи іграм оцінку від 0 до 10 (рідше – від 0 до 5, від 0% до 100%). І тут шкала показує рівень відповідності стандарту у відсотках. Наприклад, оцінка «8/10» означає, що гра на 80% відповідає стандарту. Багато людей використовують шкалу оцінок як зручний шаблон, не замислюючись про його внутрішню логіку. 10 балів – це ідеальна гра, до якої немає жодних претензій – приблизно такий сенс вкладають у шкалу оцінок, але не замислюються над цим, а залишають за замовчуванням.

Потрібно максимально точно описати поняття «ідеальна гра», щоб унеможливити різні інтерпретації, і залишити лише одне конкретне значення.

Для створення критеріїв не можна виключати вже створені системи. Тому спочатку потрібно порівнювати гру щодо її відношення до категорії «комп'ютерні ігри», а значить і за основними критеріями оцінки комп'ютерних ігор.

Розрізняють три основні компоненти гри:

1. Графіка – це основна, але не єдина частина оформлення гри. "Оформлення" звучить не так звично, як "графіка", але це більш ємне слово. До оформлення гри входить: графіка, звук, інтерфейс, загальний стиль. Все це разом дає перші враження від гри, на які ведуться основні маси гравців. Зовнішній вигляд - гарна обгортка ігрового процесу, і перша складова, яку ми оцінюватимемо окремо від цілого.

2. Геймплей - іноземне слово, яке міцно вкоренилося в середовищі гравців, що буквально позначає "грабування гри", а за змістом означає "те, наскільки цікаво грати в гру". Людині, далекій від ігрової індустрії, складно пояснити значення такого терміна. По-русски простіше назвати цей компонент «Правила гри» або «Механіка гри». До цього поняття входить те, які дії можна робити всередині гри, які ситуації відбуваються, як гра реагує на ті чи інші вчинки гравця. «Механіку» цінують, в основному, досвідченіші гравці, які проводять за іграми більшу частину вільного часу. Геймплей – основа будь-якої гри, її робоча частина, це наш другий та центральний компонент.

3. Сюжет - оповідальна частина гри, що розповідає історію світу, персонажів, що розкриває їх мотиви та вчинки. Сюжети успішних ігор стають частиною загальноігрової культури, вони найбільш цікаві тим гравцям, які вважають за краще більше спілкуватися та ділитися думками на форумах, ніж грати. Вчені-нарратологи вважають, що сюжет - це основний компонент комп'ютерних ігор, а самі ігри - нова інтерактивна форма оповіді (наступна ланка в ланцюжку "мова - книга - театр - кіно - комп'ютерні ігри"). Частково вони мають рацію, але не всі ігри вписуються в цю систему. Сюжет — це лише один спосіб прикрасити гру і прикувати до неї увагу гравців, а не якась незамінна частина, без якої не існуватиме гра в цілому. Третій компонент — смислова начинка гри, яка може бути відсутня.

Поєднання цих трьох складових та представлення самої гри дозволяють нам дати первинну оцінку грі.

Крім цих складових можна виділити і менш важливі, але також компоненти, що виділяються, такі як звук, інтерфейс, складність, персонажі та ін. і узагальнити (таблиця 3.1).

А формула обчислення первинної ігрової оцінки матиме вигляд, поданий у таблиці 3.2.

Таблиця 3.1 – Вид основних ігрових компонентів

Оформлення	Графіка	Музика	Інтерфейс
Механіка	Геймплей	Режими	Складність
Зміст	Сюжет	Персонажі	Новизна

Таблиця 3.2 – Формула отримання підсумкової ігрової оцінки

Score = (X+Y+Z)/3	**X = (X1+X2+X3)/3**	X1	X2	X3
	Y = (Y1+Y2+Y3)/3	Y1	Y2	Y3
	Z = (Z1+Z2+Z3)/3	Z1	Z2	Z3

Оцінка складається з 3-х основних критеріїв і з декількох жанрових складових.

Показники такої оцінки набагато жорсткіше регламентовані порівняно з цільною цифровою оцінкою. З одного боку, це добре тому, що зменшується випадковий фактор, розкид оцінок за якоюсь конкретною грою буде мінімальним. З іншого боку, обмежуються можливості ігрових критиків.

У цій оцінці немає кінцевих цифр, право проставляти їх залишилося у законних власників. Вибудувана паралельна система буде більш об'єктивною у своїх оцінках.

Водночас, у цій оцінці відразу ж наочно показано, якому жанру і наскільки точно відповідає дана гра, яка сторона гри найбільш розвинена: оформлення,

геймплей чи сюжет. Оцінка очищена від випадковостей та упередженостей ззовні, але всередині вона не має однозначності. Кожен гравець сам може вирішити, який із компонентів оцінки буде для нього основним, а які – другорядними. Виходить, оцінка кожної гри одна, але інтерпретувати її можна по-різному, залежно від своїх смаків.

При цьому функція оцінювання частково перекладається з ігрових критиків на всіх гравців. Самостійно робити усвідомлений вибір у зручній системі - набагато простіше, ніж шукати для себе відповідного критика зі схожими смаками, щоб орієнтуватися в існуючій системі.

Це природний результат. Ігри – витвори мистецтва, самовираження кожного автора, і їх не можна однозначно оцінити жодною цифрою, ні системою кількох показників. Все, що можливо – це оцінити технічну складову гри.

Визначимо оцінку якості розробленої гри за 10-бальною шкалою (Таблиця 3.3).

Таблиця 3.3 – Підсумкова ігрова оцінка розробленої гри

4,11	**5,00**	7	0	8
	7,33	8	7	7
	0	0	0	0

Зниження оцінки відбулось завдяки відсутності музики та сюжету гри «Шахи». Тому основна увага має бути спрямована саме на три компоненти. Кожен гравець сам визначатиме, який компонент для нього важливіший, і орієнтуватиметься по ньому.

3.3. Рекомендації до підвищення ефективності гри

Для підвищення ефективності гри запропоновано наступні рекомендації:

1) Для людей з обмеженими можливостями:

– додати звукове супроводження гри;

– додати голосове керування.

2) Додати музику.

3) Додати персонаж, який представляє собою штучний інтелект (з діалоговим вікном).

Звукове супроводження гри

Для того, щоб мінімізувати обмеження можливостей користувачів, сформованих порушенням функцій опорно-рухового апарату, пропонується додати голосове керування шаховими фігурами.

Для користувачів з проблемами зору також варто впровадити і звукове супроводження гри, а саме кожен хід має бути озвучений: назва фігури та її переміщення.

Музика та персонаж, який представляє собою штучний інтелект (з діалоговим вікном) для користувачів гратиме додаткову розважувальну роль. При цьому чат з персонажем можливо розробити використовуючи технології штучного інтелекту.

Рекомендовані функції можуть значно розширити коло користувачів розробленої гри «Шахи».

ВИСНОВКИ

У ході даної дипломної роботи:

– було проведено аналітичний огляд типів комп'ютерних ігор, де застосований штучний інтелект, та способів організації роботи штучного інтелекту;

– побудована власна модель штучного інтелекту;

– розроблено гру з використанням технологій штучного інтелекту.

Для розробки гри «Шахи» була використана мова програмування JavaScript, алгоритм Minimax та альфа-бета обрізка.

Майбутнє штучного інтелекту в іграх не можна назвати однозначним. Воно залежить не тільки від розвитку технологій, винаходу нових методів або вдосконаленні вже існуючих (машинне навчання, афективний комп'ютер, алгоритмічна творчість (computational or artificial creativity) тощо), багато в чому воно залежить від доцільності техніки ігровим завданням (коли технічне не превалює над смисловим, коли виключається те, що не відноситься безпосередньо до конкретної задачі, коли реалізація спецефектів не стає самоціллю, а виконує функцію, що підтримує, доповнює, посилює геймплей).

ПЕРЕЛІК ПОСИЛАНЬ

1. Guzdial M., Riedl M. Automated Game Design via Conceptual Expansion // AIIDE 2018, University of Alberta, Edmonton, AB, Canada, November 13–17, 2018.

2. McCarthy J. Basic Questions // What is artificial intelligence? URL: http://www.formal.stanford.edu/jmc/whatisai/node1.html.

3. Artificial Intelligence in Games / AI Frontiers // Medium. 20 September. 2018. URL: https://medium.com/aifrontiers/an-overview-ofartificial-intelligence-for-video-games-f491229c0e7d.

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

5. Artificially intelligent game bots pass the Turing test on Turing's centenary // ScienceDaily. September 26. 2012. URL: https://www.sciencedaily.com/releases/2012/09/120926133235.htm.

6. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II // DeepMind. URL: https://deepmind.com/blog/alphastar-mastering-realtime-strategy-game-starcraft-ii/.

7. Моделі та методи штучного інтелекту у комп’ютерних іграх. / Л.О. Нікітіна, С. О. Нікітін. - Х.: «Друкарня Мадрид», 2018. - 102 с.

8. Месюра В.І., Ваховська Л.М.. Основи проектування системи штучного інтелекту. Навчальний посібник. – Вінниця: УНІВЕРСУМ – Вінниця, 2000.

9. Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1):81-106.

10. Murthy, S. Automatic construction of decision trees from data: A Multidisciplinary survey.1997.

11. JavaScript. URL: https://www.javascript.com/.

12. ECMA-262 ECMAScript (JavaScript) Specification. URL: https://www.ecma-international.org/publications-and-standards/standards/ecma-262/.

13. Wikipedia. Minimax. URL: https://en.wikipedia.org/wiki/Minimax.

14. Wikipedia. Alpha–beta pruning. URL: https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning.