МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«УПРАВЛІННЯ ЕЛЕМЕНТАМИ КОМП'ЮТЕРНОЇ ГРИ З ВИКОРИСТАННЯМ МОЖЛИВОСТЕЙ НЕЙРОННОЇ МЕРЕЖІ»**

Виконав: студент 2 курсу, групи 8.1223
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
Р. О. Черновол
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Кривохата А. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук, д.т.н., професор

_____ Шило Г.М.
(підпис)

" 05 " березня 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ СТУДЕНТОВІ

Черноволу Руслану Олександровичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Управління елементами комп'ютерної гри з використанням можливостей нейронної мережі

керівник роботи Матвіїшина Надія Вікторівна, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » березня 2024 року № 385-с

2. Строк подання студентом роботи 25.11.2024

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Загальні поняття про нейронні мережі
2. Вплив нейронних мереж на комп'ютерну індустрію
3. Розробка гри використанням ШІ

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 04.03.2024

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	24.03.2024	
2.	Збір вихідних даних.	14.05.2024	
3.	Обробка методичних та теоретичних	31.06.2024	
	джерел.		
4.	Розробка першого та другого розділу.	26.08.2024	
5.	Розробка третього розділу.	13.09.2024	
6.	Оформлення та нормоконтроль	12.11.2024	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	12.12.2024	

Студент _____ Р.О.Черновол
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.В. Матвіїшина
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Управління елементами комп’ютерної гри з використанням можливостей нейронної мережі»: 38 с., 17 рис., 10 джерел, 1 додатків.

АЛГОРИТМ, ГРАВЕЦЬ, КОМП’ЮТЕР, НЕЙРОННА МЕРЕЖА, ШТУЧНИЙ ІНТЕЛЕКТ

Об’єкт дослідження – процес розробки комп’ютерних ігор

Предмет дослідження – розробка комп’ютерної гри з використанням можливостей нейронної мережі для управляння елементами гри.

Метод дослідження – аналіз, порівняння, експеримент.

Мета роботи: дослідити можливості застосування нейронних мереж у комп’ютерних іграх, проаналізувати існуючі методи та порівняти їх ефективність, розробити нейронну мережу із застосуванням інструментарію Python, оцінити її вплив на якість гри.

SUMMARY

Master’s Qualification Thesis «Managing elements of the computer game using neural network capabilities»: 38 pages, 17 figures, 10 references, 1 annex.

ALGORITHM, PLAYER, COMPUTER, NEURAL NETWORK, ARTIFICIAL INTELLIGENCE

Object of the study is the process of developing computer games

The subject of the study is the development of a computer game using the capabilities of a neural network to control the elements of the game.

Research method – analysis, comparison, experiment.

The purpose of the work is to investigate the possibilities of using neural networks in computer games, analyze existing methods and compare their effectiveness, develop a neural network using Python tools, and evaluate its impact on the quality of the game.

ЗМІСТ

Завдання на кваліфікаційну роботу ..... **Помилка! Закладку не визначено.**
Реферат 4
Summary 5
Вступ 7
1 Загальні поняття про нейронні мережі 8
1.1 Історія розвитку нейронної мережі 8
1.2 Складові ШІ 9
1.3 Робота нейронної мережі 11
1.4 Висновки до 1 розділу 13
2 Вплив нейронних мереж на комп'ютерну індустрію 14
2.1 Класифікація комп'ютерних ігор 14
2.2 ШІ при розробці ігор 15
2.3 Методи тестування комп'ютерних ігор за допомогою ШІ 17
2.4 Висновок до розділу 2 20
3 Розробка гри з управлінням ШІ 21
3.1 Особливості використання бібліотек Python для роботи зі штучним інтелектом 21
3.2 Реалізація гри 25
3.3 Демонстрація гри 28
Висновки 32
Додаток А Код гри 34

ВСТУП

В кваліфікаційній роботі досліджується потенціал нейронних мереж у створенні більш інтелектуальних і реалістичних персонажів для комп’ютерних ігор. За допомогою штучного інтелекту, персонажі зможуть діяти більш природно, адаптуючись до дій гравця та створюючи динамічні і непередбачувані ігрові ситуації.

Головна мета роботи полягає в розробці гри з використанням штучного інтелекту, яка б ефективно керувала поведінкою персонажів у віртуальному світі. Для досягнення цієї мети планується детально вивчити процес створення, навчання та тестування таких мереж.

У процесі дослідження буде проведений глибокий аналіз сучасних рішень у сфері застосування нейронних мереж в іграх. Це допоможе визначити найефективніші підходи.

Результати цього дослідження сприяють розширенню знань про застосування штучного інтелекту в розробці ігор та відкриють нові перспективи для створення більш інтерактивних та захоплюючих ігрових світів. Розроблені інструменти та моделі можуть бути використані в розробці нових ігор та покращенні існуючих.

Акценти кваліфікаційної роботи спрямовані на підвищення рівня інтерактивності ігрового процесу, а результати дослідження можуть знайти широке застосування в ігровій індустрії та сприяти створенню більш якісних та захоплюючих ігор.

1 ЗАГАЛЬНІ ПОНЯТТЯ ПРО НЕЙРОННІ МЕРЕЖІ

1.1 Історія розвитку нейронної мережі

Нейронні мережі, як одна з ключових технологій штучного інтелекту, мають багаторічну історію, що бере свій початок у середини XX століття. Вони були натхненні біологічними процесами, які відбуваються в мозку, і стали важливим інструментом для розв’язання складних задач у різних сферах. Дослідження у цій галузі пройшли кілька важливих етапів, кожен з яких зробив свій внесок у сучасний стан технології.

Історія нейронних мереж починається у 1943 році, коли нейрофізіолог Уоррен Маккалок і логік Волтер Піттс запропонували математичну модель нейрона. Вони створили просту схему роботи нейрона, яка здатна обробляти логічні функції. Їхня робота стала основою для майбутніх досліджень.

У 1958 році Френк Розенблатт розробив перцептрон – одну з перших моделей штучного нейрона, здатну навчатися. Перцептрон використовувався для класифікації даних і був реалізований у вигляді апаратної системи. Однак його обмеження, зокрема неможливість вирішувати нелінійні задачі, сповільнили розвиток цієї технології.

У 1969 році Марвін Мінський і Сеймур Пейперт у своїй книзі "Perceptrons" продемонстрували недоліки перцептронів, зокрема їхню нездатність обробляти задачі, що потребують нелінійної обробки. Ця критика призвела до зниження інтересу до нейронних мереж на кілька десятиліть. Вчені переключилися на інші напрями штучного інтелекту, такі як експертні системи.

З появою потужних комп’ютерів і великих обсягів даних у 2000-х роках нейронні мережі отримали новий імпульс. Ключові етапи:

– 2006 рік: Джеффрі Хінтон і його команда розробили методику попереднього навчання шарів у глибоких нейронних мережах, що допомогло вирішити проблему "затухаючих градієнтів";

– 2009 рік: Глибокі нейронні мережі стали використовуватися для розпізнавання образів і мови, демонструючи вражаючі результати.

Сьогодні нейронні мережі широко застосовуються в різних сферах, таких як:

– розпізнавання мови та обличчя;
– автономне водіння;
– генеративні моделі, такі як GPT (у тому числі ChatGPT);
– аналіз медичних даних.

Ключові технології, що сприяли сучасному прориву:

– convolutional Neural Networks (CNN) для аналізу зображень;
– recurrent Neural Networks (RNN) та їхні вдосконалення (LSTM, GRU) для роботи з послідовностями;
– transformer-архітектури, які змінили підхід до роботи з текстами й мовою.

1.2 Складові ШІ

Системи штучного інтелекту функціонують завдяки поєднанню великих обсягів даних із складними алгоритмами ітеративної обробки. Це дозволяє їм виявляти закономірності та особливості в аналізованих даних, поступово вдосконалюючи свої можливості. У процесі обробки інформації система постійно тестує свої результати, оцінює їх ефективність і використовує отримані знання для підвищення точності та набуття нових навичок[1].

Основні складові, без яких штучний інтелект не може функціонувати:

– машинне навчання (Machine Learning, ML). Забезпечує можливість ШІ навчатися, використовуючи алгоритми, які аналізують дані, виявляють закономірності та генерують висновки. Це дозволяє системі вдосконалювати свою продуктивність із кожною новою порцією даних;

– глибоке навчання (Deep Learning). Підкатегорія машинного навчання, яка використовує штучні нейронні мережі, що нагадують роботу людського мозку. Цей підхід дозволяє розпізнавати складні патерни, фільтрувати шум у даних і працювати з великою кількістю неоднорідної інформації;
– нейронні мережі. Базова технологія, яка лежить в основі глибокого навчання. Штучні нейрони, що імітують клітини мозку, використовують математичні моделі та алгоритми для обробки даних. Це дає змогу створювати складні системи, які здатні адаптуватися до нових задач, аналізуючи інформацію більш подібно до людського мислення.

Ці компоненти працюють у тісному взаємозв’язку, забезпечуючи штучному інтелекту здатність аналізувати дані, навчатися на їхній основі та розв’язувати широкий спектр задач [1].

Нейронні мережі складаються з трьох основних шарів: вхідного, прихованого та вихідного, кожен з яких містить велику кількість вузлів — від тисяч до мільйонів. Дані надходять у вхідний шар, де кожен вхідний параметр отримує відповідну вагу. У процесі обробки інформації вузли, які з’єднують різні шари, множать ваги зв’язків, створюючи комбінації сигналів.

Для навчання на власному досвіді нейронна мережа порівнює отримані результати з очікуваними. Виявлені розбіжності використовуються для коригування ваг, зв’язків і порогових значень, що дозволяє мережі поступово вдосконалювати свою продуктивність. Цей процес забезпечує адаптацію моделі до завдань, які вона виконує.

Штучний інтелект можна класифікувати на кілька основних типів:

– реактивні системи. Ці алгоритми не мають пам’яті та базуються виключно на аналізі поточних даних. Вони здатні приймати рішення лише в межах однієї конкретної задачі. Наприклад, шаховий ШІ аналізує позиції на дошці та обирає оптимальний хід для перемоги;

- системи з обмеженою пам’яттю. Такі ШІ використовують невеликий обсяг пам’яті для зберігання попереднього досвіду, що дозволяє приймати більш обґрунтовані рішення. Наприклад, рекомендаційні системи можуть пропонувати місця для відвідування, аналізуючи інформацію про звички користувачів і їх місцезнаходження;
- теорія розуму. Цей тип ШІ здатний розуміти людські емоції, думки та наміри, а також враховувати соціальну взаємодію. Він ще перебуває на етапі розробки, але має потенціал для значного покращення комунікації між людьми та машинами [1];
- самосвідомі системи. Це гіпотетичні майбутні системи ШІ, які будуть володіти свідомістю, емоціями та здатністю до самоаналізу. Вони зможуть самостійно приймати рішення, розуміючи свій вплив на навколишнє середовище.;

1.3 Робота нейронної мережі

Нейронні мережі є основою багатьох сучасних систем штучного інтелекту, таких як розпізнавання обличчя, обробка природної мови, автономне водіння тощо. Їхня робота базується на обчислювальній моделі, що імітує функціонування людського мозку. У цій доповіді розглянуто основні принципи роботи нейронної мережі, її структуру та процес навчання.

Нейронна мережа складається з трьох основних шарів (рис. 1.1):

- вхідний шар: Отримує дані ззовні, наприклад, зображення, текст або числові дані. Кожен нейрон у цьому шарі представляє одну ознаку або компонент вхідних даних;
- приховані шари: Складаються з одного або більше рівнів нейронів, які обробляють інформацію. Кожен нейрон у прихованих шарах отримує вхідні значення, обчислює їх за допомогою ваг та передає результат далі. Від кількості таких шарів залежить складність мережі;

– вихідний шар: Повертає результат, наприклад, клас об’єкта на зображенні або ймовірність певного сценарію [2].

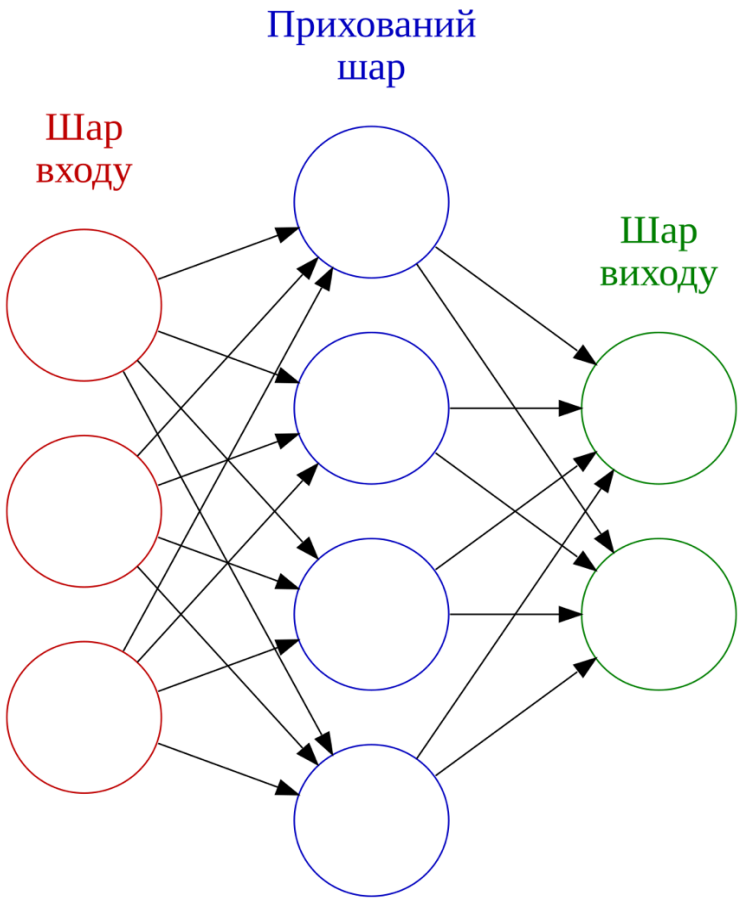


Рисунок 1.1 – Основні шари нейронної мережі [2]

Вважають основними компоненти нейронної мережі це (рис. 1.2):

– нейрони: Базові одиниці, що обчислюють вихідне значення за допомогою активаційної функції. Кожен нейрон отримує значення від попередніх шарів та перетворює їх [4];
– ваги: Параметри, які масштабують вхідні дані. Навчання мережі — це процес оптимізації цих ваг [4];
– функції активації: Математичні функції, які визначають, як нейрон реагує на вхідні дані. Популярні приклади: ReLU, Sigmoid, Tanh;
– функція втрат: Вимірює відхилення прогнозів мережі від фактичних значень. Наприклад, середньоквадратична помилка [4].

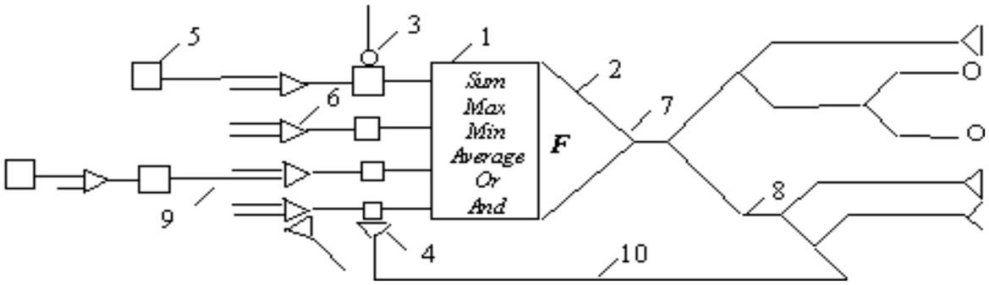


Рисунок 1.2 – Розширена модель нейронного елемента [4]

1.4 Висновки до 1 розділу

Історія розвитку нейронних мереж демонструє еволюцію від простих моделей до складних систем, що змінюють наше повсякденне життя. Від початкових математичних моделей до сучасних глибоких нейронних мереж пройдено довгий шлях, який дозволив перетворити цю технологію на потужний інструмент для вирішення складних задач у багатьох галузях. Подальші дослідження і розробки у цій сфері обіцяють ще більше інновацій у майбутньому.

Робота нейронної мережі базується на здатності імітувати складну поведінку за допомогою простих обчислювальних елементів. Завдяки гнучкій структурі та можливості навчатися на великих обсягах даних, нейронні мережі стають ключовим інструментом у вирішенні складних задач. Подальший розвиток цієї технології відкриває нові перспективи у сфері штучного інтелекту.

2 ВПЛИВ НЕЙРОННИХ МЕРЕЖ НА КОМП'ЮТЕРНУ ІНДУСТРІЮ

2.1 Класифікація комп'ютерних ігор

Комп'ютерні ігри стали важливою частиною сучасної культури та індустрії розваг. Вони охоплюють широкий спектр жанрів, платформ і механік, задовольняючи інтереси різних категорій гравців. Класифікація комп'ютерних ігор допомагає зрозуміти їхні особливості, а також зорієнтуватися в розмаїтті пропозицій на ринку[5].

Жанр гри визначає її основні ігрові механіки, тематику та стиль геймплею (рис. 2.1). Основні жанри включають:

- екшн (Action). Ігри цього жанру орієнтовані на швидкий темп, рефлекси та взаємодію гравця з ігровим світом через динамічні дії;
- пригодницькі ігри (Adventure). Фокусуються на сюжеті, дослідженні ігрового світу та вирішенні головоломок [5];
- рольові ігри (RPG – Role-Playing Games). Гравці керують персонажами, які розвиваються, отримуючи досвід і покращуючи характеристики;
- стратегії (Strategy). Ігри, що вимагають планування, тактики та управління ресурсами;
- симулятори (Simulation). Ігри, які моделюють реальні процеси або ситуації;
- спортивні ігри. Засновані на реальних спортивних змаганнях;
- головоломки (Puzzle). Гравець розв'язує задачі, часто абстрактні або логічні [5].



Рисунок 2.1 – Таблиця класифікації комп'ютерних ігор

2.2 ШІ при розробці ігор

Штучний інтелект (ШІ) відіграє ключову роль у розробці комп'ютерних ігор, забезпечуючи реалістичність ігрового процесу, адаптивність до гравців та автоматизацію багатьох аспектів створення ігор. Його використання охоплює як внутрішні механізми гри, так і зовнішні інструменти розробки [6].

ШІ дозволяє створювати реалістичних NPC, які реагують на дії гравця, планують стратегії та адаптуються до ігрової ситуації. Він автоматично створювати карти, рівні, персонажів чи сценарії, що дозволяє урізноманітнити гру. Аналізує стиль гри користувача і динамічно підлаштовує складність.

Сучасні алгоритми ШІ використовуються для створення природних діалогів з NPC. Моделі, подібні до GPT, дозволяють персонажам давати осмислені відповіді, розширюючи ігрову взаємодію.

Використання генеративних ШІ у розробці ігор виглядає цілком природним етапом розвитку індустрії, адже ігрові студії часто працюють із великими обсягами різноманітного медіаконтенту. Такі технології можуть значно спростити процес створення ігор, що вже демонструють кілька цікавих релізів [7].

Наприклад, гумористичний шутер High on Life, випущений у грудні 2022 року. За творчий напрямок у цій грі відповідав Джастін Ройланд — автор

відомого мультсеріалу «Рік та Морті»(Rick and Morty). High on Life успадкувала характерний стиль Ройланда, що поєднує провокаційний і абсурдний гумор із унікальною привабливістю [7].

Креативний директор гри поділився, що команда Squanch Games активно використовувала генеративні ШІ у процесі розробки. Хоча більшість художнього контенту створювали вручну, штучний інтелект допомагав із фінальними деталями, створенням прототипів і дрібними елементами, такими як плакати на стінах (рис. 2.2). За словами Ройланда, це сприяло створенню більш «дивної» і унікальної атмосфери гри [7].



Рисунок 2.2 – Приклад використання ШІ в комп’ютерній грі

Розвиток штучного інтелекту відкриває нові горизонти для ігрової індустрії, роблячи ігри більш реалістичними, інтерактивними та персоналізованими. Зокрема, вдосконалення моделей ШІ дозволить створювати NPC (неігрових персонажів), поведінка яких буде настільки природною, що їх складно буде відрізнити від реальних гравців.

Ігри стануть ще більш адаптивними, підлаштовуючись під інтереси, стиль гри та здібності кожного гравця. Це забезпечить унікальний ігровий досвід для кожного користувача.

Завдяки інтеграції генеративних моделей ШІ з’явиться можливість створювати унікальні ігрові світи в реальному часі. Такі світи будуть не лише технічно складними, а й естетично привабливими.

Крім того, ШІ-автори зможуть генерувати інтерактивні сюжетні лінії, які змінюються залежно від рішень гравця, надаючи історії динамічності та глибини. Це дозволить створювати справжні шедеври інтерактивного мистецтва.

2.3 Методи тестування комп’ютерних ігор за допомогою ШІ

Штучний інтелект (ШІ) активно використовується для автоматизації процесу тестування комп’ютерних ігор. Завдяки сучасним алгоритмам, він дозволяє швидко виявляти помилки, оцінювати ігровий процес та аналізувати взаємодію гравців із грою (рис. 2.3). Це значно скорочує час тестування і підвищує якість кінцевого продукту.

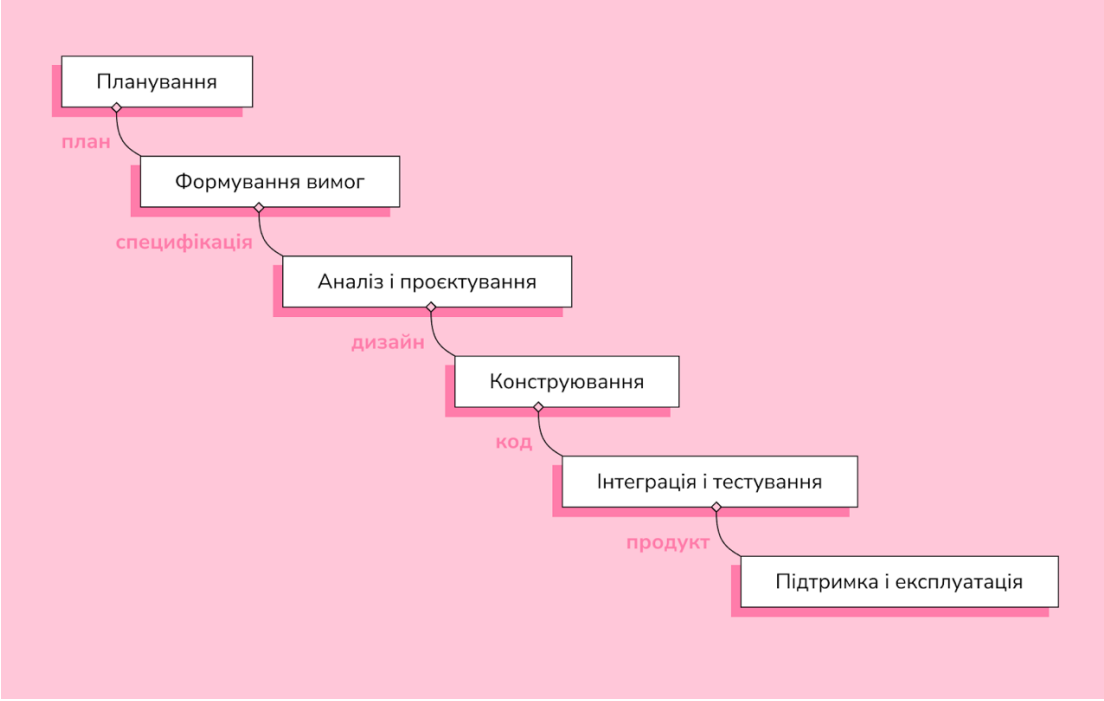


Рисунок 2.3 – Короткий план реалізації тестування гри

Штучний інтелект (ШІ) відіграє ключову роль у тестуванні комп'ютерних ігор, автоматизуючи рутинні процеси та забезпечуючи виявлення багів, оптимізацію балансу та покращення користувацького досвіду [8].

Автоматизоване функціональне тестування. ШІ моделює дії гравця, перевіряючи, чи правильно працюють ігрові функції, як-от рух персонажа, стрільба чи взаємодія з об'єктами (рис. 2.4). Також він ідентифікує баги, наприклад, зламані текстури чи невидимі стіни. Наприклад, у грі Cyberpunk 2077 ШІ використовували для перевірки маршрутів NPC, щоб уникнути некоректної поведінки у відкритому світі [8].

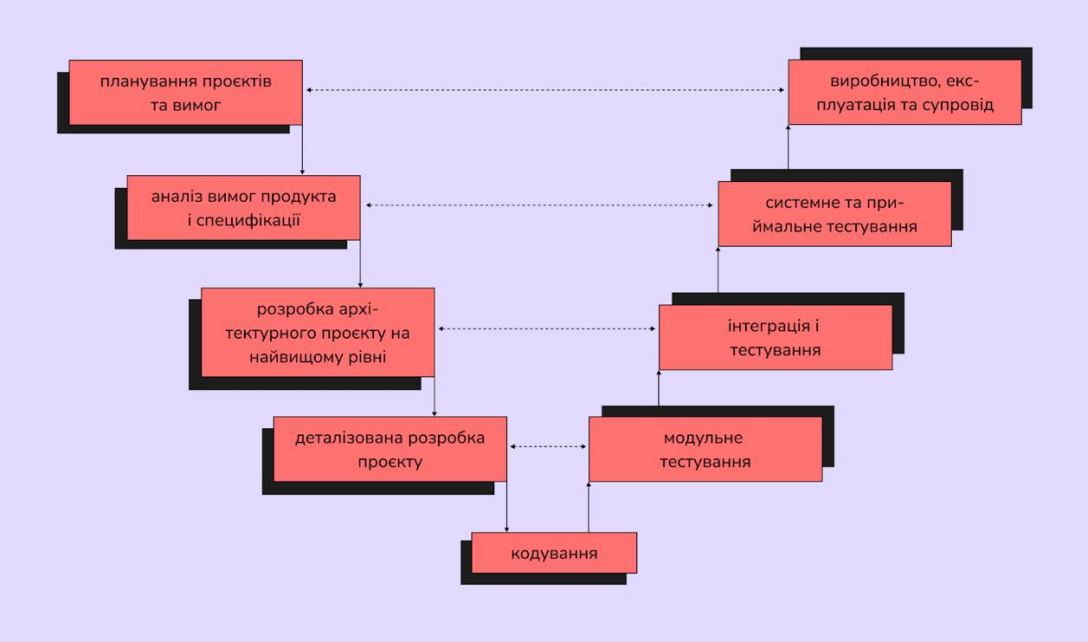


Рисунок 2.4 – Поетапний розгляд тестування

Тестування ігрового балансу. ШІ аналізує складність гри та забезпечує баланс між легкістю і викликами. Алгоритми оцінюють рівень складності, перевіряють збалансованість навичок персонажів і ефективність зброї. У багатокористувацьких іграх, таких як League of Legends, ШІ використовується для налаштування балансу героїв на основі аналізу даних [8].

Перевірка стабільності та продуктивності. ШІ виконує стрес-тестування, симулюючи багатокористувацькі сесії та взаємодію великої кількості NPC, перевіряючи стабільність серверів і продуктивність гри на різних платформах.

У масштабних іграх, як-от World of Warcraft, ШІ перевіряє серверну інфраструктуру під час пікових навантажень.

Тестування процедурної генерації контенту. У іграх із процедурною генерацією, таких як No Man’s Sky, ШІ перевіряє автоматично створені світи, оцінюючи їхню логічність та відповідність ігровим правилам [8].

Тестування UX/UI ШІ аналізує зручність інтерфейсу гри, перевіряючи навігацію в меню та взаємодію гравців із підказками й інструментами. Наприклад, у мобільних іграх ШІ допомагає визначити найбільш зручне розташування кнопок управління.

Адаптивне тестування на основі поведінки гравців. ШІ моделює різні стилі гри – агресивний, обережний чи збалансований, аналізуючи, як гравці взаємодіють із об’єктами. Це дозволяє виявляти нестандартні ситуації та баги.
При тестуванні сценаріїв гри ШІ перевіряє, чи правильно працюють сюжетні тригери та послідовність подій у грі. Наприклад, в інтерактивних іграх, таких як The Witcher 3, ШІ забезпечує відповідність дій NPC залежно від рішень гравця [8].

Завдяки ШІ процес тестування стає більш швидким, точним і ефективним, дозволяючи розробникам зосередитися на творчих аспектах ігрової розробки.

Переваги використання ШІ в тестуванні ігор:

- швидкість – ШІ може виконувати рутинні завдання значно швидше за людину;
- точність – Алгоритми ефективно знаходять складні та малопомітні баги;
- економія ресурсів – Зменшується потреба в ручному тестуванні, що скорочує витрати [8];
- масштабованість – ШІ може тестувати гру на різних платформах і в різних умовах [8];

2.4 Висновок до розділу 2

Класифікація комп’ютерних ігор дозволяє зрозуміти особливості ігор та їхню цільову аудиторію. Завдяки цьому користувачі можуть обирати відповідні ігри залежно від своїх уподобань, а розробники — орієнтуватися на певні категорії гравців. У сучасному світі, де ігри стали інструментом розваги, освіти й комунікації, розуміння класифікації допомагає глибше оцінити їхній внесок у культуру та суспільство.

Штучний інтелект значно змінює підхід до створення та розвитку комп’ютерних ігор, роблячи їх більш реалістичними, інтерактивними й адаптивними. Розробники отримують інструменти для підвищення ефективності своєї роботи, а гравці – унікальний досвід, що динамічно підлаштовується під їхні потреби. З розвитком технологій ШІ стає невід’ємною складовою ігрової індустрії майбутнього.

Штучний інтелект значно покращує процес тестування комп’ютерних ігор, забезпечуючи високу якість кінцевого продукту. Завдяки ШІ тестування стає швидшим, ефективнішим і більш детальним, що дозволяє розробникам зосередитися на творчих аспектах гри. У майбутньому ШІ продовжить вдосконалювати цей процес, адаптуючись до нових викликів індустрії.

3 РОЗРОБКА ГРИ З УПРАВЛІННЯМ ШІ

3.1 Особливості використання бібліотек Python для роботи зі штучним інтелектом

У створенні гри використовується тільки бібліотека random для кращого розуміння її можливостей бо мережа створюється з нуля.

Бібліотека random генерує випадкові числа для Ші і з цих даних він обирає найкращі варіанти для себе.

Бібліотека random в нейронній мережі може:

- генерація випадкових чисел: “random.random()”: Повертає випадкове число від 0.0 до 1.0; “random.randint(a, b)”: Повертає ціле випадкове число в діапазоні від a до b; “random.uniform(a, b)”: Повертає випадкове число з плаваючою точкою в діапазоні від a до b;
- вибір випадкових елементів: “random.choice(seq)”: Повертає випадковий елемент із послідовності; “random.sample(population, k)”: Повертає список з унікальних випадкових елементів з послідовності;
- перемішування елементів: “random.shuffle(x)”: Перемішує елементи списку x на місці.

Щоб додати Ші можливість самому робиті рішення також в майбутньому можна додати такі дві бібліотеки як TensorFlow и PyTorch.

TensorFlow і PyTorch – це дві з найпопулярніших бібліотек з відкритим вихідним кодом для глибокого навчання в Python. Вони надають широкий набір інструментів і функцій, які дають змогу створювати і навчати складні нейронні мережі.

TensorFlow, розроблений Google, спочатку був створений для досліджень у галузі машинного навчання, але швидко став стандартом в індустрії. Він пропонує високу продуктивність, особливо під час роботи з великими даними і великомасштабними моделями.

Функції TensorFlow:

- визначення архітектури: Створювати складні архітектури нейронних мереж, включно зі згортковими, рекурентними та іншими типами мереж;
- навчання моделей: Ефективно навчати моделі на великих об’ємах даних з використанням різних оптимізаторів і функцій втрат;
- деплоймент: Розгортати навчені моделі на різних платформах, включно з мобільними пристроями та веб-серверами;
- дослідження: Проводити передові дослідження в галузі глибокого навчання, завдяки гнучкості та масштабованості платформи.

PyTorch, розроблений Facebook, пропонує більш гнучкий та інтуїтивний підхід до глибокого навчання. Він часто використовується в академічних колах і стартапах завдяки своїй простоті використання і динамічному графу.

Функції PyTorch:

- швидке прототипування: Створювати та експериментувати з різними архітектурами нейронних мереж швидко та легко;
- кваліфікаційні дослідження: Проводити дослідження в галузі глибокого навчання, завдяки гнучкості та розширюваності платформи;
- продуктивність: Досягати високої продуктивності при використанні GPU і розподілених обчислень;
- інтеграція з іншими інструментами: Легко інтегрується з іншими бібліотеками Python, такими як NumPy і SciPy.

При створенні гри була використана бібліотека random бо вона не потребує навчання та легка для простого створення гри з використання Ші. Але в майбутньому можна додати складний фреймворк.

3.1 Написання коду

Цей код гри створений для демонстрації концепції інтерактивної гри, у якій ШІ (штучний інтелект) впливає на динаміку гри через адаптацію складності. Також він керує другою платформою, вибираючи кут, щоб відбити м’яч. Ось основні аспекти мети написання коду та ролі ШІ:

- розважальна складова. Гра створена для розваги, де гравець захищає ворота, використовуючи свої навички швидкості реакції та стратегічного руху. Завдяки механіці відбивання м’яча та єдиним воротам гра стає напруженою й цікавою;
- навчальна ціль. Користувачі можуть навчитися створювати ігри з використанням бібліотеки pygame, яка є потужним інструментом для розробки 2D-ігор. Гра демонструється реалізація фізики (рух м’яча), колізій (відбивання), а також базовий інтерфейс із текстом і графічною розміткою;
- використання динамічної складності. Вибір рівня складності додає варіативності до гри, дозволяючи гравцям налаштовувати ігровий досвід під свої навички.

В цій грі ШІ впливає на гру через попередньо визначені рівні складності, які контролюють швидкість м’яча. Легкий рівень визначає рух м’яча повільніше, надаючи гравцеві більше часу на реакцію. Складний рівень, м’яч рухається швидше, створюючи більше викликів.

Також штучний інтелект представлений ботом, який автоматично грає за правого гравця. Він впливає на гру, виконуючи такі функції, як постійно аналізує поточну позицію м’яча і рухається у вертикальному напрямку (вгору або вниз), щоб вирівняти свою позицію з м’ячем.

ШІ можна розширити, щоб автоматично змінювати швидкість м’яча в залежності від поточного рахунку або тривалості гри. Наприклад:

- якщо гравець успішно відбив 10 ударів поспіль, швидкість м’яча зростає;

– якщо гравець пропустив кілька м’ячів, ШІ може знизити швидкість, щоб вирівняти шанси.

У майбутньому штучний інтелект може також керувати поведінкою м’яча (змінювати траєкторію, "обманювати" гравця тощо), що зробить гру складнішою та цікавішою.

Початкова розробка коду має вигляд:

```
import pygame
import random

# Ініціалізація Pygame
pygame.init()

# Параметри вікна
WIDTH, HEIGHT = 1050, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Захист воріт")

# Кольори
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
GREEN = (34, 139, 34)
RED = (255, 0, 0)
BLUE = (0, 0, 255)
# Розмір ігрових об’єктів
PLAYER_WIDTH, PLAYER_HEIGHT = 20, 100
BALL_RADIUS = 15
GOAL_WIDTH = 200  # Ширина воріт
# Початкові позиції
player_pos = [50, HEIGHT // 2 – PLAYER_HEIGHT // 2]
ball_pos = [WIDTH // 2, HEIGHT // 2]
```

```
ball_speed = [0, 0]  # Швидкість м’яча буде визначатися рівнем складності

# Рівні складності
difficulty_levels = {
"Легко": [4, 4],
"Складно": [6, 6] # Новий рівень складності
```

Розгорнутий код представлений в Додатку А.

3.2 Реалізація гри

Для написання коду гри програми, потрібно познайомитись з такими бібліотеками, як pygame (бібліотека для розробки 2D-ігор, що включає інструменти для графіки, звуку, та обробки подій) та random (використовується для генерації випадкових чисел, якщо в майбутньому знадобиться додати непередбачувану поведінку) (рис. 3.1).

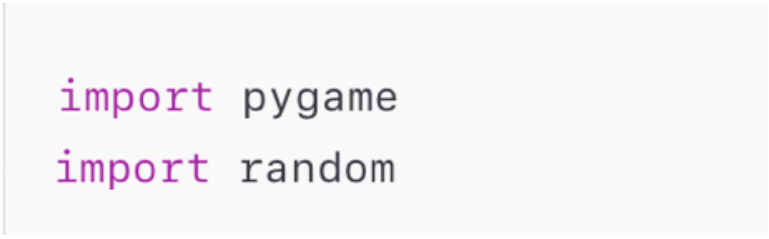


Рисунок 3.1 – Імпорт бібліотек pygame, random в код програми

Далі прописуємо налаштування ігрового вікна та кожного елемента на ньому. Створюється ігрове вікно розміром 1290x600 пікселів. Визначаємо колір у форматі RGB, використовуються для малювання об’єктів (фону, м’яча, воріт, гравця). Визначаємо ширину та висоту гравця, радіус м’яча та ширину воріт (рис. 3.2, 3.3, 3.4, 3.5).

```
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Захист воріт")
```

Рисунок 3.2 – Параметри ігрового поля

```
WHITE = (255, 255, 255)
GREEN = (34, 139, 34)
BLUE = (0, 0, 255)
RED = (255, 0, 0)
```

Рисунок 3.3 – Кольори об’єктів на полі

```
PLAYER_WIDTH, PLAYER_HEIGHT = 20, 100
BALL_RADIUS = 15
GOAL_WIDTH = 200
```

Рисунок 3.4 – Ширина, висота поля

```
player_pos = [50, HEIGHT // 2 - PLAYER_HEIGHT // 2]
ball_pos = [WIDTH // 2, HEIGHT // 2]
```

Рисунок 3.5 – Параметри гравця

Змінні, які застосовуються в коді мають наступний вигляд (рис. 3.6):

- running – контролює основний цикл гри;’’
- clock –обмежує кількість кадрів в секунду;
- score – підраховує кількість відбитих м’ячів;
- game_over – стає True, якщо м’яч влучив у ворота;

```
running = True
clock = pygame.time.Clock()
score = 0
game_over = False
```

Рисунок 3.6 – Вид змінних в програмі

Під час гри в правому верхньому куті екрана відображається поточний рахунок.

У разі завершення гри в центрі екрана з’являється повідомлення про результат «Гра закінчена».

Після завершення гри показується фінальне повідомлення, а потім програма автоматично завершується.

Цей інтерфейс базується на ChatGPT, моделі штучного інтелекту, розробленій OpenAI. Конкретно, використовується архітектура GPT-4 (Generative Pre-trained Transformer), яка є однією з найпередовіших моделей обробки природної мови на сьогодні.

GPT-4 створений для роботи з текстом: генерації відповідей, розв’язання задач, написання коду, аналізу даних, пояснень, творчого написання текстів і багато іншого. Його мета – бути помічником, який може розуміти природну мову і генерувати відповідний контент.

GPT-4 – це модель на основі трансформерів, яка навчається на величезних масивах текстових даних. Він розуміє контекст і генерує відповідь на основі того, що "запам’ятала" під час навчання. Здатний аналізувати і враховувати контекст бесіди, щоб давати релевантні і зрозумілі відповіді. Також може створювати унікальні тексти, включаючи історії, вірші, або навіть програмний код.

Але в цього ШІ є деякі обмеження, як:

– він не має власних думок або розуміння реального світу, його відповіді базуються виключно на даних, на яких він був навчений;

- не може створювати неправдиву або неточну інформацію, якщо вхідні дані недостатньо чіткі;
- результати можуть змінюватися залежно від того, як поставлене запитання.

GPT-4 взаємодіє з кодом програми, а саме:

- розбирається у вашому Python-коді;
- розуміє ваші вимоги;
- пропонує вдосконалення або виправлення коду;
- допомагає реалізувати складні алгоритми або додати нові функціональні можливості.

Гра симулює захист воріт, де гравець керує воротарем, щоб відбивати м’яч, який намагається потрапити у ворота. ШІ бот розрахований на те, щоб майже завжди відбивати м’яч, якщо він знаходиться в межах досяжності. Завдяки простій логіці, бот швидко реагує на зміни положення м’яча, що робить його складним для обігравання. Гра завершується, коли м’яч перетинає лінію воріт. Відбиті м’ячі збільшують рахунок гравця, а траєкторія м’яча змінюється залежно від місця зіткнення.

У грі є вибір рівня складності, який впливає на швидкість м’яча. ШІ у цій грі виконує роль модератора складності, керуючи швидкістю м’яча та забезпечуючи динамічний ігровий процес. Також нейрона мережа виступає ботом правого гравця, який відбиває м’яч. Це демонструє, як навіть прості механізми ШІ можуть зробити гру більш інтерактивною та цікавою.

3.3 Демонстрація гри

Після запуску програми з’являється ігрове вікно розміром 1290x600 пікселів (рис. 3.7). Поле має зелений фон, центральну лінію, круг у центрі та ворота з лівого боку. Гравець бачить екран із текстом вибрати рівні складності.

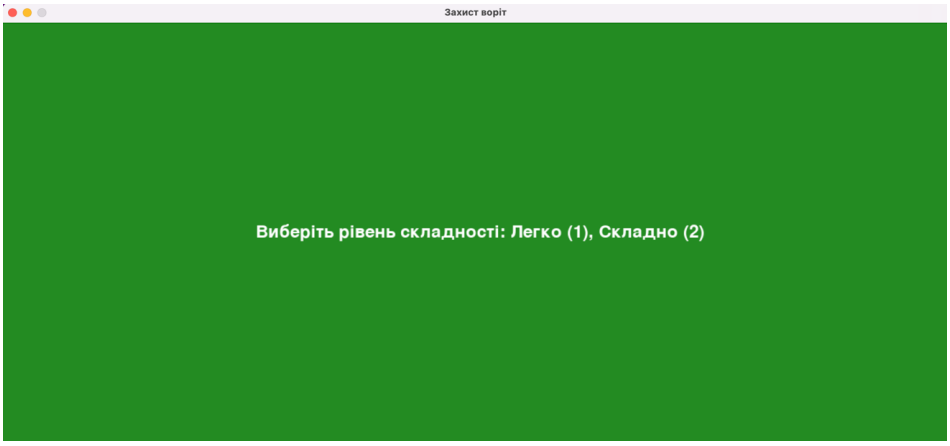


Рисунок 3.7 – Вид початкового екрану після запуску програми

Натискаючи клавіші 1 або 2 гравець визначає рівень складності:

- легко: М’яч рухається повільно;
- складно: М’яч має середню швидкість;

Гра починається з того, гравець керує воротарем (синій прямокутник) за допомогою стрілок вгору/вниз. М’яч (червоне коло) починає рухатися з центру поля в напрямку воріт. В той же момент ШІ підбирає кут відбиття м’яча та також намагається його відбити (рис. 3.8).



Рисунок 3.8 – Початок гри

Гравець та бот продовжує відбивати м’яч на тих пір поки не набере 10 балів. Це може забрати дуже багато часу, так як ШІ кожного разу після удару м’яча від синьої платформи, змінює траєкторію та швидкість підльоту до

гравця, що дає змогу забити гол та завершити гру. Але після трьох ударів від платформи м’яч набуває силу удару та після забиття голу, швидкість повертається в попередню.

Після того, як рахунок вже становить 7 балів, ШІ починає збільшувати швидкість м’яча для того, що гравець програв. Це відбувається на кожному рівні складності (рис. 3.9).



Рисунок 3.9 – Гравець набирає 7 балів та швидкість збільшується

Якщо ж все таки гравець володіє швидкою реакцією та захистив свої ворота до кінця, то його чекає перемога (3.10).



Рисунок 3.10 – Гравець виграє

Бувають випадки, коли гравець такими навиками не досконало володіє та результатом гри стає програш гравця. Тоді ігрове вікно видає таку картинку (3.11).



Рисунок 3.11 – ШІ виграє

Гра "Захист воріт" – це проста, але захоплююча гра, яка тренує реакцію гравця. Вона добре підходить для розваги та розвитку швидкості прийняття рішень. Додавання нових функцій може зробити гру ще цікавішою та тривалішою для гравців.

ВИСНОВКИ

У кваліфікаційній роботі було досліджено можливості використання нейронних мереж для ефективного управління елементами комп’ютерних ігор. Проаналізував сучасний стан досліджень у цій галузі, розроблено власну гру з використанням штучного інтелекту.

Результати дослідження підтвердили, що нейронні мережі мають великий потенціал для створення більш інтелектуальних і реалістичних ігрових персонажів. Завдяки здатності навчатися на великих обсягах даних та адаптуватися до змінних умов, нейронні мережі можуть забезпечити динамічну та непередбачувану поведінку персонажів, що значно збагачує ігровий процес.

Розроблена гра з управлінням нейронної мережі показала високу ефективність у керуванні поведінкою персонажу у тестовому середовищі. Персонажі, керовані нейронною мережею, демонстрували більш природну та адаптивну поведінку, ніж персонажі, керовані традиційними скриптами. Це свідчить про те, що використання нейронних мереж є перспективним напрямком розвитку комп’ютерних ігор.

Однак, незважаючи на досягнуті результати, існують ще багато напрямків для подальших досліджень. Зокрема, необхідно розробити більш складні моделі нейронних мереж, здатні керувати поведінкою великої кількості персонажів в умовах реального часу. Також важливо дослідити можливості використання різних типів нейронних мереж для вирішення різних ігрових завдань.

Результати дослідження підтвердили, що нейронні мережі мають великий потенціал для створення інтелектуальних і реалістичних ігрових персонажів. Завдяки здатності навчатися на великих обсягах даних та адаптуватися до змінних умов, нейронні мережі можуть забезпечити динамічну та непередбачувану поведінку персонажів, що значно збагачує ігровий процес.

ПЕРЕЛІК ПОСИЛАНЬ

1. Капсульна нейронна мережа. URL : https://uk.wikipedia.org/wiki/Капсульна_нейронна_мережа (дата звернення: 26.11.2024).

2. Штучна нейронна мережа. URL: https://uk.wikipedia.org/wiki/Штучна_нейронна_мережа(дата звернення: 27.11.2024).BI

3. Що таке штучний інтелект: історія, види та складові. URL : https://gigacloud.ua/blog/navchannja/scho-take-shtuchnij-intelekt-istorija-vidi-ta-skladovi(дата звернення: 27.11.2024).

4. Детальний опис компонентів та роботи нейронних мереж. URL : https://studfile.net/preview/7409892/ (дата звернення: 27.11.2024).

5. Жанри відеоігор. URL : https://uk.wikipedia.org/wiki/Жанри_відеоігор (дата звернення: 28.11.2024).

6. Концепція ШІ в комп’ютерних іграх. URL : https://uk.sharpcoderblog.com/blog/concept-of-ai-in-game-development# (дата звернення: 30.11.2024).

7. Як ШІ використовуються у розробці відеоігор, і до чого це може призвести. URL : https://mezha.media/articles/yak-shi-vykorystovuiut-u-rozrobtsi-videoihor/ (дата звернення: 30.11.2024).

8. Штучний інтелект у тестуванні ігор. URL : https://training.qatestlab.com/blog/technical-articles/artificial-intelligence-in-game-testing/ (дата звернення: 30.11.2024).

9. Stauffer M. Up & Running: A Framework for Building Modern PHP Apps. 3rd ed. California: O’Reilly Media, 2023. 559 p.

10. Aggarwal C. Neural Networks and Deep Learning: A Textbook 2nd ed. Berlin: Springer, 2023. 553 p.

ДОДАТОК А

Код гри

```
import pygame
import random

# Ініціалізація Pygame
pygame.init()

# Параметри вікна
WIDTH, HEIGHT = 1290, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Захист воріт")

# Кольори
WHITE = (255, 255, 255)
GREEN = (34, 139, 34)
RED = (255, 0, 0)
BLUE = (0, 0, 255)

# Розмір ігрових об’єктів
PLAYER_WIDTH, PLAYER_HEIGHT = 20, 100
BALL_RADIUS = 15
GOAL_WIDTH = 200  # Ширина воріт

# Початкові позиції
player_pos = [50, HEIGHT // 2 – PLAYER_HEIGHT // 2]
bot_pos = [WIDTH – 50 – PLAYER_WIDTH, HEIGHT // 2 – PLAYER_HEIGHT // 2]
ball_pos = [WIDTH // 2, HEIGHT // 2]

# Рівні складності
difficulty_levels = {
    "Легко": [4, 4],
    "Складно": [6, 6]
}

# Шрифти
font = pygame.font.Font(None, 36)
```

```
# Вибір рівня складності
def choose_difficulty():
    screen.fill(GREEN)
    text = font.render("Виберіть рівень складності: Легко (1), Складно (2)", True, WHITE)
    screen.blit(text, (WIDTH // 2 – 300, HEIGHT // 2 – 20))
    pygame.display.flip()

    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_1:
                    return "Легко"
                elif event.key == pygame.K_2:
                    return "Складно"

# Встановлення швидкості м’яча залежно від обраного рівня складності
difficulty = choose_difficulty()
initial_speed = difficulty_levels[difficulty][:]  # Збереження початкової швидкості
ball_speed = initial_speed[:]

# Функція для малювання розмітки поля
def draw_field():
    # Зелений фон
    screen.fill(GREEN)

    # Центральна лінія
    pygame.draw.line(screen, WHITE, (WIDTH // 2, 0), (WIDTH // 2, HEIGHT), 3)

    # Центральне коло
    pygame.draw.circle(screen, WHITE, (WIDTH // 2, HEIGHT // 2), 100, 3)
    pygame.draw.circle(screen, WHITE, (WIDTH // 2, HEIGHT // 2), 5)

    # Зона воріт
    pygame.draw.rect(screen, WHITE, (0, (HEIGHT – GOAL_WIDTH) // 2, 10, GOAL_WIDTH))
    pygame.draw.rect(screen, WHITE, (WIDTH – 10, (HEIGHT – GOAL_WIDTH) // 2, 10, GOAL_WIDTH))

# Функція для початкового розташування м’яча
```

```
def reset_ball(starting_direction="left"):
    ball_pos[0] = WIDTH // 2
    ball_pos[1] = HEIGHT // 2

    # Напрямок і початкова швидкість
    direction_x = -1 if starting_direction == "left" else 1
    ball_speed[0] = direction_x * initial_speed[0]
    ball_speed[1] = random.choice([-1, 1]) * initial_speed[1]

# Перед основним циклом гри
reset_ball(starting_direction="left")  # Початковий рух м’яча до гравця

# Головний цикл гри
running = True
clock = pygame.time.Clock()

player_score = 0
bot_score = 0
game_over = False
bot_hit_counter = 0  # Лічильник ударів бота

while running:
    draw_field()  # Малюємо розмітку поля

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Керування гравцем
    keys = pygame.key.get_pressed()
    if keys[pygame.K_UP] and player_pos[1] > 0:
        player_pos[1] -= 10
    if keys[pygame.K_DOWN] and player_pos[1] < HEIGHT - PLAYER_HEIGHT:
        player_pos[1] += 10

    # Логіка бота
    if bot_pos[1] + PLAYER_HEIGHT // 2 < ball_pos[1]:
        bot_pos[1] += 6
    if bot_pos[1] + PLAYER_HEIGHT // 2 > ball_pos[1]:
        bot_pos[1] -= 6

    # Рух м’яча
    ball_pos[0] += ball_speed[0]
    ball_pos[1] += ball_speed[1]
```

```
    # Відбивання м’яча від меж
    if ball_pos[1] – BALL_RADIUS <= 0 or ball_pos[1] + BALL_RADIUS >= HEIGHT:
        ball_speed[1] = -ball_speed[1]

    # Відбивання м’яча від гравця
    if (
        player_pos[0] < ball_pos[0] – BALL_RADIUS < player_pos[0] + PLAYER_WIDTH
        and player_pos[1] < ball_pos[1] < player_pos[1] + PLAYER_HEIGHT
    ):
        ball_speed[0] = -ball_speed[0]

    # Відбивання м’яча від бота
    if (
        bot_pos[0] < ball_pos[0] + BALL_RADIUS < bot_pos[0] + PLAYER_WIDTH
        and bot_pos[1] < ball_pos[1] < bot_pos[1] + PLAYER_HEIGHT
    ):
        bot_hit_counter += 1  # Збільшення лічильника ударів
        if bot_hit_counter % 3 == 0:  # Якщо це третій удар
            random_speed_multiplier = random.uniform(1.5, 2.5)  # Сила удару
            ball_speed[0] = -abs(ball_speed[0]) * random_speed_multiplier
            ball_speed[1] += random.uniform(-3, 3)
        else:  # Звичайний удар
            ball_speed[0] = -abs(ball_speed[0])

    # Перевірка на гол
    if ball_pos[0] – BALL_RADIUS <= 0:  # Гравець пропустив
        bot_score += 1
        reset_ball(starting_direction="right")
    elif ball_pos[0] + BALL_RADIUS >= WIDTH:  # Бот пропустив
        player_score += 1
        reset_ball(starting_direction="left")

    # Малювання гравців і м’яча
    pygame.draw.rect(screen, BLUE, (*player_pos, PLAYER_WIDTH, PLAYER_HEIGHT))
    pygame.draw.rect(screen, WHITE, (*bot_pos, PLAYER_WIDTH, PLAYER_HEIGHT))
    pygame.draw.circle(screen, RED, ball_pos, BALL_RADIUS)

    # Відображення рахунку
```

```
    score_text = font.render(f"Гравець: {player_score} | Бот: {bot_score}", True, WHITE)
    screen.blit(score_text, (WIDTH // 2 – 150, 10))

    # Завершення гри
    if player_score >= 10 or bot_score >= 10:
        game_over = True
        winner = "Гравець" if player_score > bot_score else "Бот"
        screen.fill(GREEN)
        end_text = font.render(f"Гра закінчена! Переможець: {winner}", True, WHITE)
        screen.blit(end_text, (WIDTH // 2 – 200, HEIGHT // 2 – 20))
        pygame.display.flip()
        pygame.time.wait(3000)
        running = False

    pygame.display.flip()
    clock.tick(60)

pygame.quit()
```