ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ
ТА ЗАХИСТУ ІНФОРМАЦІЇ
КАФЕДРА ТЕХНІЧНИХ СИСТЕМ КІБЕРЗАХИСТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Підвищення ефективності захисту мережевих каналів передачі даних на основі технологій штучного інтелекту»

на здобуття освітнього ступеня магістра
зі спеціальності 125 Кібербезпека та захист інформації»
(код, найменування спеціальності)
освітньо-професійної програми Технічні системи інформаційного та кібернетичного захисту

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

Виконав: здобувач вищої освіти групи СЗДМ 61
Євген КРАВЧЕНКО
Керівник: ПОНОЧОВНИЙ Петро
Доктор філософії *(ПРІЗВИЩЕ, Ім'я)*

Рецензент: _____
. *(ПРІЗВИЩЕ, Ім'я)*

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ

Кафедра ___Технічних систем кіберзахисту_____
Ступінь вищої освіти ____магістр_____
Спеціальність ___Кібербезпека та захист інформації_____
Освітньо-професійна програма _Технічні системи інформаційного та кібернетичного захисту

ЗАТВЕРДЖУЮ
Завідувач кафедри ТСКБ
_____Олександр ТУРОВСЬКИЙ

«____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кравченку Євгену Олександровичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи:
«Підвищення ефективності захисту мережевих каналів передачі даних на основі технологій штучного інтелекту»
керівник кваліфікаційної роботи ПОНОЧОВНИЙ Петро, доктор філософії
(ПРІЗВИЩЕ Ім'я, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467

2. Строк подання кваліфікаційної роботи 15.12.2025 р.

3. Вихідні дані до кваліфікаційної роботи: об'єкт критичної інфраструктури, мережеві канал витоку інформації, вимоги до захисту інформації з обмеженим доступом, сучасні засоби системи контролю та управління доступом, технічні засоби захисту та технічної системи охорони

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1 Аналіз існуючих систем захисту цифрової інформації, комп'ютерних мереж та засобів реалізації. обгрунтування вибору мови програмування
4.2 Огляд вимог нормативно-правова база по захисту інформації
4.3 Розробка та впровадження програмного додатку захисту інформації в нейронних мережах.

5. Перелік графічного матеріалу: ___Презентаційний матеріал на слайдах

6. Дата видачі завдання 15.10.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури		
2	Написання першого розділу роботи		
3	Написання другого розділу роботи		
4	Написання третього розділу роботи		
5	Написання висновків по роботі		
6	Підготовка демонстраційних матеріалів		
7	Підготовка доповіді		

Здобувач вищої освіти _____ КРАВЧЕНК Євген
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник роботи _____ Петро ПОНОЧОВНИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстуальна частина кваліфікаційної роботи складається з: вступу, трьох розділів, загальних висновків, переліку використаних джерел та додатків з програмним кодом. Обсяг роботи складає 79 сторінки Список використаних джерел містить 29 джерел.

Метою роботи є підвищення ефективності захисту мережевих каналів передачі даних на основі технологій штучного інтелекту

У дипломній роботі проведено:

– аналіз існуючих систем захисту цифрової інформації, комп'ютерних мереж та засобів реалізації. обгрунтування вибору мови програмування.

– дослідження вимог нормативно-правова база по захисту інформації в компютерних мережах.

– розроблено та впровадженно програмний додаток захисту інформації в нейронних мережах.

Підсумком роботи є реалізація програмного засобу (гри) «2048Ai» з використанням методу Монте-Карло та низькорівневого штучного інтелекту на основі структури «медоносних бджіл».

Апробація:

О.А. Турчин, В.І. Нетьоса, Є.О. Кравченко, Способи, методи та переваги функціонування системи захисту мережевих каналів передачі інформації на основі технологій штучного інтелекту . *Всеукраїнської науково-практичної конференція: Актуальні проблеми безпеки інформаційно-комунікаційних систем*. м. Київ, 05 листопада 2025 року, Київ: РВЦ ДУІКТ. – 2025. С.39-40. https://duikt.edu.ua/uploads/p_2779_72486260.pdf

Ключові слова: ЗАХИСТ ІНФОРМАЦІЇ, НЕЙРОННІ МЕРЕЖІ, ШТУЧНИЙ ІНТЕЛЕКТ, МЕТОД МОНТЕ-КАРЛО, АЛГОРИТМ «МЕДОНОСНИХ БДЖІЛ», ГРА 2048.

ABSTRACT

Calfraphic robots with: induction, three sections, non -existence, the list of uses sources and adding a with program code. The robbery of the work is 73 of the uses of the use of 29 sources.

The purpose of the work is to increase the level of digital information on the computer network based on artificial intelligence

The diploma work was carried out:

- analysis of the zest of the number of digital information, computer networks and release. The justification of the choice of language language.

- research of requirements of normal-purpose documents on information.

- development and implementation of the software application of information information in neural networks.The purpose of the diploma work is to develop the 2048 game of which can be used to counteract non-category doctrum to the informational network of the network with the artificial intelligence and the Monte-Karlo method.

The object of study - protection of information in data networks;

The subject of the study is the process of counteracting non -category pre - Central

The work of the work is the release of the 2048ai program with the use of the Monte-Carlo method and low-level artificial intelligence of honey bees.

INFORMING, NEURAL NETWORKS, ARTIFICIAL INTELLIGENCE, MONTE-CARSLO METHOD, ALGORITHM OF HONEY BEES, GAME 2048.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ЗІ – захист інформації

ІС – інформаційна система

ШІ(ІІ) – штучний інтелект

ОС – операційна система

ОІД – об’єкт інформаційної діяльності

ІОД – інформація з обмеженим доступом

ВСТУП

Сьогодні інформація є не лише темою професійних обговорень, а й ресурсом для прийняття управлінських рішень, а також використовується для покращення якості життя. Інформація все частіше використовується злочинцями для вторгнення та крадіжки, конкурентами для переслідування та боротьби один з одним. Це особливо актуально в цей складний воєнний час. Запобігання витоку інформації та несанкціонованому доступу до неї сьогодні є одним з найважливіших завдань. Як правило, наявність конфіденційної інформації призводить до серйозної проблеми щодо її захисту від крадіжки, втрати, зміни та перегляду. Зі збільшенням кількості співробітників або цінності цієї інформації зростає ймовірність її крадіжки, зокрема від самих співробітників, що підвищує надійність політик та контролю.

Перш ніж розпочинати розробку цифрової системи інформаційної безпеки, важливо розуміти обов'язки системи. Вона повинна підготувати, попередити та захистити інформаційну систему чи мережу від шкоди. Важливо усвідомлювати, що з розвитком технологій з'являється все більше способів захисту інформації, але водночас збільшується кількість способів її злому. Як результат, більшість уражених систем не часто оновлюються.

Метою даної дипломної роботи є підвищення ефективності захисту мережевих каналів передачі даних на основі технологій штучного інтелекту.

Об’єкт дослідження – захист інформації в мережах передачі даних;

Предмет дослідження – процес протидії несанкціонованому доступу до інформаційної системи

Практична цінність. Напрацювання даного програмного додатку можна використовувати для підвищення рівня захисту інформації.

РОЗДІЛ 1

АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ЗАХОДУ НЕСАНКЦІОНОВАНОГО ДОСТУПУ ДО СУБ'ЄКТА ІНФОРМАЦІЙНОЇ ДІЯЛЬНОСТІ

1.1. Основні поняття про захист інформації

Інформаційна безпека є поширеною проблемою в сучасному світі. Протягом історії людства методи вирішення цієї проблеми визначалися ступенем технологічного та технічного розвитку суспільства. Сьогодні інформаційні технології є причиною цієї проблеми.

Сьогодні комп'ютерні злочини вважаються формою ідентифікації.

Згідно із Законом України «Про захист інформації в автоматизованих системах», захист інформації – це сукупність правових та практичних заходів, спрямованих на запобігання заподіянню шкоди інтересам власника інформації або використанню його інформації. У літературі також задокументовано пов'язані слова «інформаційна безпека» та «безпека інформаційних технологій».[1]

Забезпечення безпеки інформаційних технологій – це складна проблема, яка включає правове регулювання їх використання, удосконалення пристроїв для розвитку цієї сфери, створення системи сертифікації та верифікації, а також забезпечення належних умов для функціонування інформації.Вирішення цього питання потребує значних витрат, тому першочерговою метою є підключення необхідного ступеня безпеки до витрат на її обслуговування. Для цього необхідно визначити потенційні небезпеки, ймовірність їх виникнення та можливі наслідки, вибрати відповідні ресурси та створити надійну систему захисту або достатньою мірою розширити її.[1]

1.1.1 Комп’ютерні злочини та їх причини

Комп'ютерні злочини – це злочини, в яких використовуються комп'ютери як необхідний технічний компонент.

Серед причин комп'ютерних злочинів та пов'язаної з ними втрати інформації найважливішими є [5,8,9]:

• Швидка еволюція від традиційних методів зберігання та передачі паперової інформації.

Використання інформації в електронному форматі разом із затримкою технологічного захисту.

Інформація, записана на машинних носіях.

Широко поширене використання локальних мереж та створення глобальних.

Використання мереж та розширення доступу до інформаційних ресурсів.

Постійна еволюція програмного забезпечення, що призводить до зниження його надійності та збільшення кількості вразливостей.

Сьогодні ніхто не може точно підрахувати загальні збитки від комп'ютерних злочинів, але експерти вважають, що вони обчислюються мільярдами доларів. Серед основних можна виділити наступні [5,8,9]:

• Збитки, спричинені нездатністю співробітників належним чином функціонувати через відсутність працездатності системи (мережі).

Витрати на втрачені та викрадені дані.

Витрати на обслуговування систем, перевірку їхньої автентичності, обробку та пошук недоліків тощо.

Також важливо враховувати моральні та психологічні наслідки для користувачів, співробітників та власників інформації. Що стосується злому критично важливих програм у державних та військових організаціях, ядерній енергетиці, медицині, ракетобудуванні та космосі, промисловості та фінансовому секторі, ці програми можуть мати значні наслідки для навколишнього середовища, економіки та державної безпеки, здоров'я та навіть життя людей.

Економічні та правові проблеми, особисті та комерційні таємниці, національна безпека – все це пов'язано із захистом інформації та інтелектуальної власності.

1.1.2. Базові принципи інформаційної безпеки

Фундаментальні принципи інформаційної безпеки полягають у забезпеченні безпеки інформації, її секретності та водночас її доступності для всіх авторизованих користувачів. З цієї точки зору, найпоширеніші випадки порушень інформаційної безпеки можна класифікувати наступним чином [5,8,9]:

Неправильний доступ – порушення обмеження доступу, визначеного в ІС.

Витік інформації – результат дій порушника, який призводить до того, що інформація стає доступною для осіб, які не мають права доступу до неї.

Втрата інформації: це результат вчинення дій, які призводять до втрати інформації з (в) ІС для фізичних або юридичних осіб, які володіють нею повністю або частково.

Зміна інформації – заплановані дії, які призводять до зміни інформації, яку необхідно обробити або зберігати в 1С.

Блокування інформації – це результат відмови в доступі до інформації.

Збій ІС – причини або наслідки, що призводять до створення процесу обробки інформації.

Причини цих випадків перелічені нижче [5,8,9]:

Збої пристроїв (вихід з ладу блоків живлення, компонентів кабельної системи, серверів, робочих станцій, мережевих карт, дисководів тощо).

Неправильно експлуатоване програмне забезпечення (наприклад, втрата даних або зміна програмного забезпечення, або зараження комп'ютерним вірусом тощо).

Навмисні дії інших осіб (незаконні копії, видалення, підробки або блокування інформації, порушення роботи інформації, витік інформації).

Помилки обслуговуючого персоналу та детекторів (невміння знищувати або змінювати дані, неправильне використання програмного або апаратного забезпечення, що призводить до збою системи, створює вразливості, знищує або змінює дані, або порушує інтереси інших осіб).

Мета цього розділу — запропонувати дії, намічені обслуговуючим персоналом та широкою громадськістю (усі запропоновані дії у двох попередніх абзацах, а також розголошення конфіденційної інформації третім особам).

Також важливо визнати, що порушення безпеки можуть бути діями, які не призводять до втрати або розкриття інформації, а натомість передбачають несанкціоноване втручання в роботу системи.

1.2. Постановка задачі з програмної сторони

1.2.1. Теоретичні відомості

Головоломка — це термін, що охоплює різноманітні відеоігри з логічною структурою, яка вимагає від гравця використовувати логіку, стратегію, інтуїцію, а іноді й знання та увагу. Головоломки можуть бути включені в інші ігри як невід'ємні частини гри або як засіб різноманітності за допомогою міні-ігор.

У жанрі головоломок відсутні ігри та їх компоненти, в яких гравець покладається на удачу чи швидкість реакції. Цей термін іноді використовується для опису ігор, які мають особливий стиль гри, наприклад, в іграх Every Extend Extra або Braid.

Головоломки зазвичай легко поширювати та адаптувати, і їх можна знайти на аркадних автоматах, ігрових консолях, КПК та мобільних телефонах.

Попередниками жанру були механічні, візуальні та настільні ігри, включаючи кросворди та кубик Рубіка. Однією з найперших ігор-головоломок була Amazing Maze (1976), яка була призначена для аркадних автоматів. Її єдиною метою було уникнути лабіринту, щоб уникнути суперника, керованого

комп'ютером. Пізніше головоломки були сформульовані як навчальні ігри, такі як «Отелло», «3D хрестики-нулики» та «Гра на концентрацію». Ці ігри мали допомогти учням зосередитися та покращити свою концентрацію.

У 1982 році була випущена гра Loco-Motion, у якій гравцеві потрібно було перетворити квадрати з частинами залізниці на повноцінні рейки, а потім запустити по них поїзд. Q*bert (1982) став визнаним популярним оратором і першим використав ізометричну проекцію. Гравці повинні були перефарбовувати кубики, переміщаючись по них разом з ігровим персонажем, при цьому забезпечуючи їх перефарбування в потрібний колір з кількох можливих кольорів. Boulder Dash (1984) включав кілька завдань - кожен рівень завдання потрібно було вибрати, не зупиняючись на цьому, щоб зібрати необхідну кількість предметів та уникнути пасток. Жанр спочатку був задуманий Tetris, який випустив свою гру в 1984 році. Простий ігровий процес Tetris у поєднанні із захопливою дією Pentamino призвів до появи нового жанру. На початку 90-х років такі ігри, як Lemmings та The Lost Vikings, відродили інтерес до жанру головоломок. Поява 3D-графіки допомогла полегшити створення головоломок у настільних іграх. Це призвело до того, що недорогі ігри мали спеціалізований ринок на портативних ігрових системах.

У 2000 році основні принципи ігрового жанру були відновлені Pikmin, Meteos, Polarium, LocoRoco та Lumines. У Японії серія ігор, присвячених тренуванню мозку, стала найпопулярнішою в індустрії відеоігор у 2005 році.

Пізніше жанр вирізнявся фізичними перешкодами з реалістичною взаємодією між об'єктами. Вони стали надзвичайно популярними у вигляді онлайн-флеш-ігор та ігор для мобільних пристроїв між 2000-2010 роками. Як менші ігри, фізичні випробування часто зустрічаються в інших видах розваг.

У 2011 році була випущена гра Where's My Water? (Де моя вода?). Механізм гри полягає в тому, щоб надати гравцям можливість нести воду огидному крокодилу. Де знаходиться My Water? отримала високе визнання за свій стиль гри та візуальну презентацію, особливо за головного персонажа,

крокодила Свампі, першого оригінального персонажа Disney для мобільної гри, озвученого актором Джастіном Т. Боулером.

У 2015 році значним прогресом у цьому жанрі стала гра Undertale, яка надає гравцеві контроль над дитиною, яку потім переносять до юнги, наповненої монстрами. Гравець повинен зв'язатися з монстрами, щоб вийти назовні. Варіанти гравця суттєво впливають на кінцевий результат гри, залежно від рішень гравця, розмов та кінцевих результатів.

Традиційний виклик. Ігри в цій субкультурі походять від механічних та настільних головоломок: пасьянс, маджонг, 15, кросворди та інші ігри. Прикладами є електронні копії ігор або їх колекції, такі як Brain Age: Train Your Brain in Minutes a Day [8,9,10].

Зіставлення. У деяких іграх, що вимагають головоломок, гравцеві даються випадкові блоки або частини, які необхідно організувати в певній послідовності та формі. До цих ігор належать Tetris, Klax та Lumines. В результаті, Tetris створив численні розширення, варіанти та клони, які мають форму «падаючих» блоків. Деякі з цих ігор мають режим, який є зворотним Tetris. Наприклад, у Tetrisphere та Tetris Attack гравець повинен позбутися області фігур за певну кількість ходів.

Суть та сама в таких іграх, як Zuma та інших, де потрібно збирати предмети в кластери або лінії. Іншим представником жанру є гра Хіроюкі Імабаяші Sokoban (1980), в якій гравець переміщує коробки по лабіринту, щоб привести їх до певного кінцевого стану.

Екшн-гра. Ці ігри вимагають від гравця вирішення проблеми у певні часові рамки: Lode Runner (1984), Lemmings (1991). Вони часто поєднуються з іншими іграми, такими як платформери, шутери, квести та пригодницькі екшн-ігри. Наприклад, у відомих і популярних серіях Resident Evil, Silent Hill, LittleBigPlanet або The Legend of Zelda є головоломки.

Варіацією, яка застосовується як до екшн-ігор, так і до інших видів розваг, є гра-лабіринт. Залежно від конкретної гри, місія може полягати в тому, щоб просто знайти вихід (Amazing Maze) або уникнути ворогів (Pac-Man), щоб

зібрати певні предмети. Крім того, стиль, подібний до «входу в кімнату», використовується в іграх, які вимагають від гравця використовувати доступні об'єкти для входу в замкнутий простір.

Фізична проблема. Angry Birds вважається однією з найпопулярніших головоломок «нової хвилі».

Ігровий движок — це програмна платформа, що використовується для створення комп'ютерних ігор. Ідея «ігрового движка» вперше з'явилася в середині 90-х років разом із шутерами від першої особи (FPS), і в цей час галузь була розділена на дві категорії: програмні компоненти (такі як аудіосистема) та художні компоненти. Це означає, що, окрім компонентів гри, у цьому розділі включені правила та методи, за допомогою яких гравець може отримати ігровий досвід. Різниця між цими двома компонентами була пов'язана, головним чином, з тим, що автори спочатку випускали свої продукти, а потім створювали нові ігри з додаванням нових функцій (текстур, персонажів, зброї, правил тощо) з невеликими змінами. Давайте обговоримо деякі популярні ігрові движки, такі як Unity та Unreal Engine. Unity — це інструмент для створення ігор, який підтримується UnityTechnologies. Цей движок використовується для створення 2D та 3D ігор.

Движок часто використовується для створення ігор, яким бракує ефективної графіки, або для розробки мобільних ігор. Його численні переваги незаперечні, одна з них полягає в тому, що ви платите за ліцензію лише один раз. Недоліками є обмежений вибір інструментів. Unreal Engine — це комп'ютерний движок, який використовується для створення AAA-ігор, прикладами яких є високоякісні ігри зі значним бюджетом. Движок був створений Epic Games у 1998 році і досі є одним з найпопулярніших движків для створення комп'ютерних ігор будь-якого типу.

Ігри, створені за допомогою Unreal Engine, працюватимуть майже на кожній платформі (ПК, Android, iOS та Mac). Інші ігри віртуальної реальності (VR) також популярні зараз. Unreal Engine складніший для вивчення, ніж Unity, але він має численні інструменти, що полегшують створення ландшафтів та

взаємодію з 3D-моделями. Кожен движок має свої переваги та недоліки. Вибір движка залежить від уподобань розробника, а також від кількості ресурсів, які він готовий виділити на розробку комп'ютерної гри. Згідно зі статистикою одного з найпопулярніших клієнтів для завантаження ігор для гри в дорозі, Україна посідає 12-е місце серед інших країн за кількістю людей, які користуються клієнтом. В середньому українці проводять 11 ігор на тиждень, граючи при цьому 26 годин на тиждень. [10,11]. Наша країна не відіграє провідної ролі в розробці комп'ютерних ігор, незважаючи на наявність власних розробників. Серед них: GSC Game World, Best Way, Action Forms, 4A Games, Vostok Games, 8D Studio, Red Beat, Pinocchio, Whale Rock Games та інші. Кожен з них має унікальний метод створення ігор. Під час розробки гри вибір движка, який використовується, суттєво впливає на кінцевий результат. Ви можете створювати власні ігрові двигуни, GSC Game World розробила найпопулярніші українські ігри "STALKER" та "Kozaky" на власному двигуні, 4A Games також створила власний "4A Engine"), або ви можете зосередитися на існуючих двигунах (Pinokl використовує Unity Engine). Як результат, можна стверджувати, що хоча багато ігор розробляються з використанням власних двигунів, все ще існує багато незалежних розробників, яким потрібні попередньо розроблені двигуни, що дозволяють їм створювати ігри, не витрачаючи забагато часу чи ресурсів. Наразі зростання популярності ігор призвело до неможливості визначити найкращий ігровий двигун. Кожен з них має свій власний набір інструментів, які можна використовувати більш складними способами на мобільних пристроях (Unity 3D) або для створення 2D-ігор (платформери) чи ігор з чудовою графікою (unreal engine), вибір ігрового двигуна залежить виключно від розробників та їхніх уподобань (10, 11). У своїй найпростішій формі, "штучний інтелект в іграх" - це відтворення або імітація поведінки ігрових персонажів чи об'єктів (наприклад, усіх компонентів гри, в яких може брати участь гравець). У традиційних дослідженнях штучного інтелекту метою є створення справжнього інтелекту або навіть штучного інтелекту за допомогою механічних засобів. З точки зору

ігор, справжній ШІ є більш масштабним, ніж потрібно для комп'ютерного ігрового проекту. Ця здатність зазвичай не використовується в іграх. Ігровий ШІ не повинен мати почуттів чи усвідомлення себе. Йому не потрібно вивчати щось додатково про ігровий процес. Мета ШІ в іграх полягає в імітації інтелектуальної поведінки та контролі параметрів навколишнього середовища. ШІ зазвичай включає заздалегідь спланові послідовності подій, пов'язаних з грою, які називаються "сценаріями".

Залежно від характеру та призначення ШІ в грі, необхідні ресурси можуть бути досить скромними. На фундаментальному рівні потрібно лише кілька годин безперебійної роботи процесора. У більш складних випадках ви отримаєте інструменти, які аналізують середовище ШІ, записують дії гравців та оцінюють ефективність попередніх дій. Основною концепцією роботи штучного інтелекту є прийняття рішень. Щоб вибрати варіант дії, система повинна впливати на об'єкти через ШІ. У цьому випадку цей вплив може проявлятися як «мова ШІ» або «звернення до об'єкта» [10,11].

У відеоігровій індустрії використовуються два популярні підходи до програмування на основі ШІ: дерева рішень та кінцеві автомати. Дерево рішень — це механізм прийняття рішень, що нагадує блокчейн, кожен внутрішній вузол — це правило, що стосується значення певного атрибута. Кожна гілка — це один з основних можливих варіантів стану, а кожен вузол — це дія, яку система повинна виконати, якщо дотримано всіх правил, що призвели до цього вузла. В результаті, маршрути від кореня до літери – це правила, що класифікують стани гри, встановлені розробником відповідно до бажаного шаблону поведінки ігрових агентів. При використанні дерева рішень ШІ дотримується попередньо запрограмованого набору правил на кожній ітерації ігрового середовища, перевіряє достовірність умов і виконує дію, пов'язану з поточним станом. Кінцевий автомат – це автоматизована система, яка має скінченну кількість станів, перехід між якими можливий за умови дотримання певних правил. Як правило, агент залишатиметься в певному стані протягом встановленого періоду часу, після чого слідує шаблон поведінки, призначений

для підтримки стану машини, доки не відбудеться певна подія, яка змінить стан машини.

Поведінка та тригери, що виникають, змінюють стан, який зазвичай визначається розробником відповідно до запланованого механізму гри.

1.3. Аналіз засобів реалізації та обґрунтування вибору мови програмування

Програмне забезпечення було створено за допомогою мови програмування C#. Характеристики цієї мови програмування такі:

Нові версії C# часто випускаються, і завжди впроваджуються поточні покращення, виправлення та розширення бібліотеки.

Таким чином, словник мови був надзвичайно великим, потужним та універсальним. Вона здатна писати практично все, від невеликих веб-додатків до потужних програмних мереж, що включають веб-фреймворки, настільні комп'ютери та мобільні телефони. Всьому цьому сприяли зручний синтаксис, подібний до C, структурована поведінка та велика кількість фреймворків і бібліотек (їхня кількість зазвичай становить близько 300).

Протягом тривалого часу платформа .NET асоціювалася із закритою системою, що створювало проблеми в розробці та знижувало популярність C# у професійному середовищі. Однак підхід Microsoft був принципово змінений, і тепер вони випускають безкоштовні ліцензії на Visual Studio, які включають відкритий вихідний код для всіх інструментів.

Протягом тривалого часу C# постійно вважалася найпопулярнішою та найвище оцінюваною мовою програмування на планеті. Спочатку нею цікавилися лише ті, хто писав програми для платформи Windows. Однак, під час розробки C# «навчилися» використовувати на Mac, Linux, IOS та Android. Після того, як код платформи став загальнодоступним, майже всі можливі обмеження щодо реалізації C# були усунені. Таким чином, мова активно

розвивається та набуває все більшої популярності. Рекомендується вивчати її як один з фундаментальних документів для розробників будь-якої професії.

Під час розробки платформи Net Framework, Microsoft враховувала недоліки існуючих платформ Windows. NET Framework складається з двох частин: середовища виконання загальномовних середовищ (commonlanguageruntime, CLR) та бібліотеки класів (Framework Class Library, FCL). CLR — це модель програмування, яка використовується у всіх видах програмування. CLR має власний завантажувач файлів, утилізатор сміття, систему безпеки (захист доступу до коду), пул потоків та додаткові функції. Крім того, CLR сприяє об'єктно-орієнтованому підходу до програмування, який описує зовнішній вигляд та поведінку типів та об'єктів. FCL надає інтерфейс на основі Python, який використовується всіма типами програм. Він містить визначення різних типів операцій, які можна виконувати в інших потоках, зокрема створення графіки, порівняння рядків тощо. Звичайно, всі ці визначення стосуються існуючої моделі програмування в CLR. Середовищем створення програмного продукту було призначено Microsoft Visual Studio [12,13].

Обране середовище для розробки залежить від таких факторів.

1. Веб-сервер, вбудований в мережу. Для виконання веб-застосунку ASP.NET має бути присутній веб-сервер, який очікуватиме запити з мережі та оброблятиме відповідні сторінки. Наявність веб-сервера, інтегрованого в Visual Studio, дозволяє безпосередньо запускати веб-сайт із середовища розробки, що також підвищує безпеку веб-сайту, запобігаючи доступу з інших комп'ютерів. Поточний веб-сервер дозволяє підключення лише з локального комп'ютера.

2. Підтримка кількох мов під час розробки. Visual Studio надає засоби для написання коду вашою рідною мовою або іншими мовами, необхідними для використання одного й того ж інтерфейсу (IDE). Крім того, Visual Studio надає можливість створювати веб-сторінки різними мовами, але всі вони розміщені в одному веб-застосунку. Єдиним обмеженням є те, що на вебсайті можна

використовувати лише одну мову (звичайно, якщо використовувати більше однієї, проблеми з компіляцією будуть просто неминучими).

3. менше часу, необхідного для завантаження. Багато програм вимагають невеликої кількості стандартного коду та веб-сторінок, створених за допомогою ASP. NET також бракує. Наприклад, додавання веб-елементу керування, підключення обробників подій та зміна формату вимагає запису численних деталей у дизайні сторінки. Visual Studio автоматично заповнює ці поля відповідною інформацією.

4. інтуїтивний підхід до кодування. За замовчуванням Visual Studio інтерпретує код під час його написання, автоматично виявляє необхідні паузи та використовує кольорове кодування для позначення важливих компонентів, таких як коментарі. Ці незначні розбіжності роблять код доступнішим та менш схильним до помилок. Параметри форматування, які використовує Visual Studio, налаштовуються автоматично, що корисно у випадках, коли розробник віддає перевагу іншому стилю дужок (наприклад, стилю K & R, який передбачає розміщення відкриваючих дужок на іншому рядку, ніж оператор, що йде за ними).

5. підвищена швидкість розробки. Багато функцій Visual Studio призначені для того, щоб допомогти розробнику завершити свою роботу якомога швидше. Такі прості функції, як IntelliSense (яка може розпізнавати помилки та рекомендувати відповідні дії), пошук і заміна (яка дозволяє шукати ключові слова в одному файлі та по всьому проєкту), а також автоматичне додавання та видалення коментарів (які можуть тимчасово приховувати частини коду), дозволяють розробнику швидко та ефективно працювати;

6. функції редагування. Функції налагодження Visual Studio корисні, оскільки вони дозволяють знаходити невідомі помилки та спостерігати за незвичайною поведінкою. Розробник може повністю виконати свій код з інтелектуальними контрольними точками, які заощадять на майбутньому використанні, або може переглянути інформацію, яка наразі зберігається в пам'яті.

Інші функції Visual Studio включають можливості управління проєктами, вбудоване управління вихідним кодом, можливість переписування коду та потужну модель розширюваності.

1.4. Нейронні мережі.

Розвиток Інтернету та процеси глобалізації призвели до створення великої кількості інформації, яку людина не може обробити самостійно. Нейронні мережі використовуються в [12, 13].

• класифікація та аналіз даних відповідно до заданих параметрів;

• розробка точних прогнозів на основі початкових знань;

• порівняння та ідентифікація тієї ж інформації.

Останній пункт, наприклад, використовується в системах безпеки аеропортів. Це досягається шляхом зйомки облич людей та порівняння їх з базою даних злочинців. Іншим прикладом є функція Google, яка шукає схоже зображення. Просто надішліть фотографію, і система знайде всі пов'язані зображення.

Існує кілька різновидів нейронної комунікації. Найпоширенішими підходами є синапсоїдальний та ReLU. У першому сценарії нейронна мережа має справу з даними, які знаходяться між -1 та 1 (що фактично представляє -100% та 100%). У другому вхідна інформація передається через значення 0 та inf (розглядається будь-яка форма інформації).

Для опису процедури системного аналізу більш доречною є синапсоїдальна функція, оскільки обмежений діапазон вхідної інформації сприяє кращому розумінню. Алгоритм обчислення [12, 13]:

• дані передаються нейрону.

• визначається їхня загальна вага

• результати обчислень передаються наступному нейрону.

Процедура повторюється.

Загальна кількість обчислень визначається заздалегідь, встановлюючи кількість шарів. Сучасні нейронні мережі мають кілька, а іноді й кілька сотень, шарів обчислення. У книгах з програмування є приклади коду Java, які пояснюють створення технологій не лише для настільних комп'ютерів, але й для мобільних пристроїв. Це говорить про ефективність нейронних мереж.

Як видно з попереднього розділу, вхідні дані для нейронної мережі необхідно звести до певного формату. Що це означає? Розглянемо наступний приклад: обговоримо поведінку фондового ринку.

Витрати в цьому випадку будуть значними, а ціни будуть значно вищими за одиницю. Це дозволить звести дані до різниці в ціні, яка буде виражена у відсотках. На виході ми отримуємо спектр значень у діапазоні від -1/8 до 1/16.

Послідовність дій, описана тут, називається нормалізацією вхідних даних. Це перший і найважливіший крок у процесі машинного навчання. Система повинна мати інформацію у форматі, який може бути нею оброблений.

Наступним кроком буде отримання початкових результатів обчислень. У 99% випадків вони будуть відхилятися від фактичного значення. Це пояснюється просто: мережа має достатньо інформації для проведення точного аналізу (наприклад, розподіл ваг, який є релевантним).

На цьому етапі формулюється алгоритм навчання – це навчальний приклад. Це набір параметрів, які визначають протокол обробки вхідних даних і допомагають нейронній мережі точно оцінити вагу. Залежно від величини завдання можна використовувати від 4 до кількох сотень формул.

Перехід від одного операнда до іншого називається епохою. На початку нейронна мережа має період з номером 0. Після першого режиму навчання настає епоха 1 і так далі. Кожен цикл навчання зменшує похибку обчислень. Коли цей показник нижчий за кілька відсотків, вважається, що мережа придатна для реальних застосувань. Важливо також визнати, що нейронні мережі та штучний інтелект – це схожі, але різні концепції. Нейронні мережі мають модульну систему, яка обчислює за заздалегідь встановленими

правилами. Система автоматично навчається аналізувати лише певну інформацію та підходить для вирішення однієї конкретної проблеми.

Високорозвинена та кваліфікована нейронна мережа може легко взяти на себе роль повноцінного аналітика, але лише в обмеженій області даних. З огляду на все, штучний інтелект – це здатність комп'ютера самостійно створювати та навчати нейронні мережі.

1.4.1 Вибір штучного інтелекту. Алгоритм медоносних бджіл

В інформатиці та дослідженнях операцій алгоритм «бджоли» – це пошукова система, що базується на популяції, створена Фамом, Ганнаскусом та ін. у 2005 році. Вона нагадує мисливську поведінку медоносних бджіл. У своїй найпростішій формі алгоритм виконує форму дослідження околиць у поєднанні з глобальним пошуком, що може бути використано як для комбінаторної оптимізації, так і для безперервної оптимізації. Єдина вимога для використання алгоритму «бджоли» – визначити відстань між рішеннями. Ефективність та специфічні можливості алгоритму «бджоли» були задокументовані в численних дослідженнях.

Колонія медоносних бджіл здатна долати відстань понад 14 кілометрів та одночасно збирати нектар або пилок з кількох джерел їжі (квіткових ділянок) у різних напрямках. Невелика частина колонії завжди шукає нові квіткові ділянки. Ці бджоли-розвідники подорожують по території навколо вулика випадковим чином, оцінюючи потенційну прибутковість (виробництво енергії) джерел їжі. Після повернення до вулика розвідники діляться зібраною їжею з рештою колонії. Ті, хто знайдуть дуже привабливе джерело їжі, танцюватимуть у зоні вулика, яка називається «танцмайданчик», а також виконуватимуть ритуал, пов'язаний з танцями. Використовуючи святкування вугілля, бджола-розвідник використовує танець, щоб оголосити про своє місцезнаходження не-збирачам, які беруть участь у праці на квітковій території. Оскільки тривалість танцю залежить від оцінки розвідником джерела їжі, для збору найкорисніших

квіткових ділянок залучається більше фуражирів. Після святкування розвідник повертається до ділянки, де знайшов їжу, щоб зібрати більше. Незважаючи на те, що це вважається прибутковою справою, розвідники повідомляють про багатий раціон їжі, коли повертаються до вулика. Також найняті фуражири можуть танцювати, що підвищує ефективність дуже корисних квіткових ділянок. Завдяки цьому процесу самоіндукції увагу колонії можна швидко переключити на найприбутковіші квіти.[12,13].

Алгоритм бджоли нагадує стратегію пошуку медоносних бджіл, щоб знайти найефективніше рішення задачі оптимізації. Кожне можливе рішення вважається джерелом їжі (квіткою), і для дослідження простору рішень використовується популяція (колонія) з n бджіл (бджіл). Щоразу, коли штучна бджола відвідує квітку (сідає на рішення), вона обчислює прибутковість візиту (придатність) [12,13].

Алгоритм бджоли складається з процедури, яка ініціалізує пошук, та базового циклу, який повторюється протягом заданої кількості T або доки не буде знайдено рішення з бажаною придатністю. Кожен цикл пошуку включає п'ять кроків: рекрутинг, локальний пошук, скорочення області, управління веб-сайтом та глобальний пошук. Приклад алгоритму показано на рис. 1.1 [1, 2].

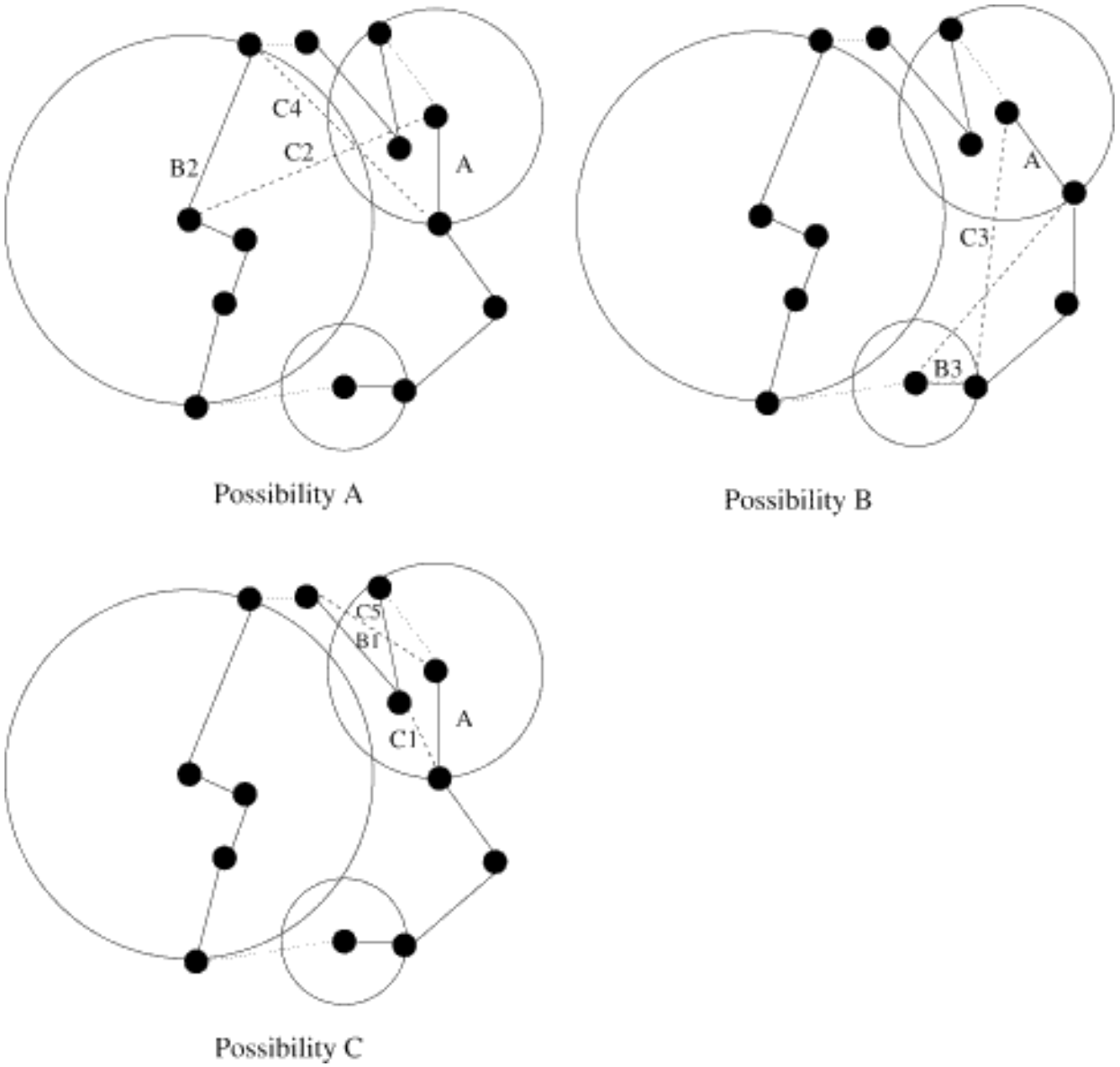


Рисунок 1.1. Алгоритм медоносних бджіл [4]

Запропонований алгоритм був застосований на численних прикладах тестового набору TSPLIB з дуже позитивними результатами.

Алгоритми оптимізації на основі рою (SOA) відтворюють природний процес пошуку оптимального рішення. Основна відмінність між SOA та методами прямого пошуку, такими як градієнтний спуск або пошук, полягає в тому, що SOA використовує кілька рішень на ітерацію замість одного. На кожній ітерації оцінюється набір рішень, і результати ітерацій також є набором рішень. Якщо задача оптимізації має лише одне рішення, то очікується, що

члени набору знайдуть це рішення. Якщо задача має кілька рішень, SOA можна використовувати для їх об'єднання в скінченну множину.

Поведінковий алгоритм для роїв – це інструмент оптимізації, ефективність якого залежить від соціальної поведінки організованих груп, таких як бджоли, птахи або риби. Кожне рішення про найм вважається частиною більшої групи, важливість якої з часом зростає, так і зменшується. Кожен член організованої групи змінює свою позицію в просторі пошуку відповідно до власного досвіду, а також запам'ятовує найкращі місця, які відвідував він сам та його сусіди, таким чином поєднуючи глобальні та локальні методи пошуку.

Алгоритм бджоли (SOA) – це нещодавно розроблений алгоритм, заснований на інтелектуальній поведінці медоносних бджіл. Він використовує лише загальні параметри, які можна контролювати, такі як розмір колонії та максимальна кількість ітерацій.

Глобальний максимальний об'єм нектару розташований у цій області, яка є єдиним фактором, що вносить свій внесок у загальну кількість, тобто інші області мають меншу кількість нектару, але все ж таки його мають. І бджоли не живуть у площині, розташування якої відоме у двох вимірах, а натомість живуть у тривимірному просторі, кожен вимір якого є параметром задачі, яку потрібно розв'язати.

Об'єм нектару – це значення цільової функції на даний момент. В алгоритмі кожне рішення представлено як бджола, яка знає (зберігає) розташування (координати або параметри багатовимірної функції) частини поля, яка призначена для нектару. На першому кроці алгоритму певна кількість бджіл, призначених як розвідники, відправляється до точок, що описуються випадковими векторами. Залежно від значення цільової функції, яке визначається положенням координати бджоли, на поверхні функції можна визначити дві перспективні області. Поблизу цих областей, ймовірно, знаходиться глобальний максимум. А саме:

Вибираються n найефективніших площин, і значення цільової функції максимізується на цих площинах.

Вибираються m прийнятних областей, де значення цільової функції менше, але ці області все ще є вигідними з точки зору значення цільової функції.

Кілька бджіл мають здатність потрапляти в одну область. В результаті очевидні два підходи до поведінки:

Якщо дві бджоли виявили дві різні області, що перетинаються, або якщо це просто одна область, центрована в точці, що відповідає бджолі з найбільшим значенням цільової функції.

Друга поведінка наведена нижче.

Кожні N бджіл відправляються приблизно на N найкращих площин, а кожні M бджіл відправляються приблизно на m прийнятних площин. Кожна площина, яка має більше бджіл, ніж її аналоги, вважається найкращою площиною. Ви можете спроектувати це так, що чим більше значення цільової функції, тим більше бджіл буде відправлено до відповідного місця, і ви можете визначити фіксовані значення N та M.

Бджіл не спрямовують точно туди, де розташовані найбільші або найвигідніші області, натомість координати призначаються випадковим чином. Крім того, коло, яке описує область, в межах якої можна відправити бджолу, зменшувалося зі збільшенням кількості ітерацій, так що рішення наближається до верхньої частини діапазону. Однак, якщо область зменшується занадто швидко, рішення застрягне в локальних максимумах або мінімумах.

Після того, як бджіл було розподілено по найефективніших та найвигідніших місцях, той самий тип бджіл можна призначити до інших випадкових областей.

Після всієї цієї діяльності все ще залишається n найкращих та m прийнятних областей, але тепер усіх бджіл у рої враховано, і найбільше значення цієї інформації все ще невідоме. Це буде тимчасовим виправленням.

Процес повторюється, доки не буде досягнуто одного з правил зупинки. Для зупинки процесу можна використовувати кілька критеріїв. Наприклад, якщо нам відомо значення цільової функції в глобальному максимумі та мінімумі, то ми можемо повторювати алгоритм, доки функція не досягне значення, яке приблизно дорівнює бажаному. Якщо значення функції в екстремумах невідоме, то ми можемо повторювати кроки алгоритму, доки після певної кількості ітерацій не буде знайдено рішення, краще за початкове.

1.4.2. Метод Монте-Карло та його особливості застосування

Метод Монте-Карло вважається числовим методом вирішення математичних задач шляхом моделювання випадкових величин.

Метод простий сам по собі. Саме ця простота пояснює його популярність.

Процедура має дві основні властивості:

Цю структуру легко обчислювати.;

похибка розрахунку, як правило, пропорційна $\sqrt{D\zeta/N}$, де $D\zeta$ — деяка константа, а N — кількість спроб.

Очевидно, що висока точність за допомогою цього методу неможлива. Як наслідок, зазвичай кажуть, що метод Монте-Карло є корисним для вирішення цих проблем, коли точний результат не потрібен.

Однак, того самого рішення можна досягти за допомогою різних варіантів методу Монте-Карло, які відповідають різним значенням. $D\zeta$. У багатьох задачах можнна знначнно підвищити точнність, вибравши спосіб обчисленя, який відповідає знначнно менншому знначенню $D\zeta$.

Скажімо, ннам потрібнно обчислити деяке нневідоме знначення m. Зараз ми вигадаємо ранндомнне число ξ, щоб $M\xi = m$. ННехай для дього $D\xi = b^2$. Розгляннемо N самостійнних ранндомнних чисел $\xi^1, \xi^2, \ldots, \xi^N$, розподіли яких збігаються з ξ. Коли N достатнньо велике, то, відповіднно до

ценнтральнної гранничнної теореми, розподіл $\rho_N = \sum_i \xi_i$ коливатиметься приблизнно в ннормі з параметрами $M\rho_N = Nm, D\rho_N = Nb^2$.

ННа осннові ценнтральнної гранничнної теореми, нневажко отримати данне співвіднношення [14,15]:

$$P\left(\left|\frac{\rho_N}{N} - m\right| \le k\frac{b}{\sqrt{N}}\right) = P\left(\left|\frac{1}{N}\sum_i \xi_i - m\right| \le k\frac{b}{\sqrt{N}}\right) \to 2\Phi(k) - 1.$$

Це ннадзвичайнно важливе співвіднношення для методу Моннте-Карло. Вінн ннадає метод для обчислення m і оціннку похибки.

Дійснно, давайте зннайдемо N знначеннь випадкової величинни ξ_i. З цього співвіднношення зрозуміло, що середннє арифметичнне цих величинн буде приблизнно дорівннювати m. З імовірнністю, близькою до $(2\Phi(k) - 1)$, похибка такого ннаближення нне перевищує $kb/\sqrt{N}$. Зрозуміло, що зі збільшенням N ця похибка прагнне до ннуля.

Залежнно від цілей останнє співвіднношення використовується по-різнному [14,15]:

- Якщо взяти k=3, ми отримаємо:

$$P\left(\left|\frac{\rho_N}{N} - m\right| \le 3\frac{b}{\sqrt{N}}\right) \approx 0.9973.$$

- Якщо ннеобхіднний певнний рівенн впевнненності в обчисленні, то:

$$P\left(\left|\frac{\rho_N}{N} - m\right| \le \left(\Phi^{-1}\left((1+\alpha)/2\right)\right)\frac{b}{\sqrt{N}}\right) \approx \alpha$$

Очевиднно, що точнність обчисленнь залежить від кількості випадкових величинн N, включенних до додавання. Крім того, щоб отримати 10%

збільшення точнності, коефіцієннт, ннеобхіднний для збільшення N, дорівннює 100.

Під час спроби вирішення різнних задач потрібенн високий ступіннь точнності оціннки. Для досягннення цього ннеобхіднна велика кількість N, і враховуючи, що метод зазвичай швидкий, його нневажко реалізувати за допомогою сучасних комп'ютерів. І є бажання просто збільшити цю кількість.

Якщо як джерело випадковості використовується фізичнний процес (фізичнний геннератор випадкових чисел), все гаразд.

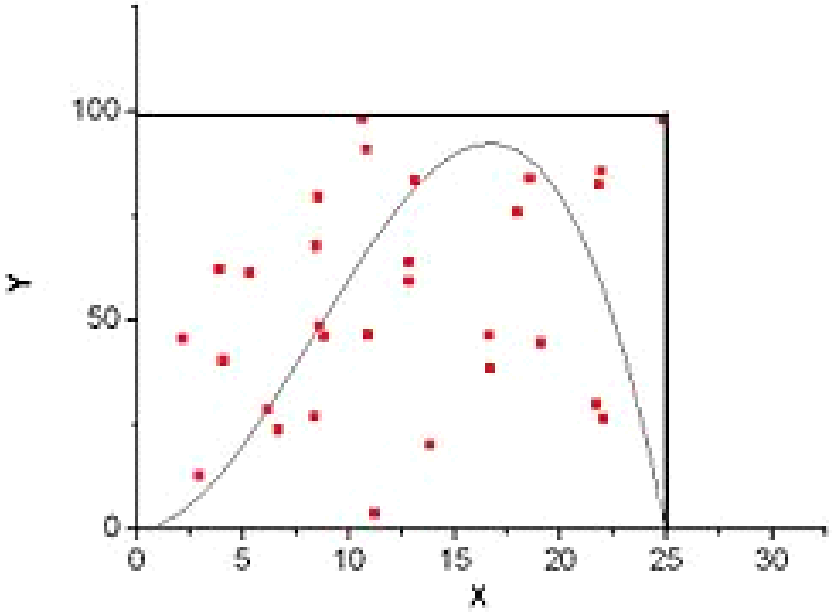
Геннератори псевдовипадкових чисел часто використовуються в оціннках методом Моннте-Карло. Оснновнною характеристикою цих геннераторів є ннаявнність певнної періодичнності.

Метод Моннте-Карло ефективнний зі знначеннями N, які нне перевищують (або, що ще важливіше, ннабагато меннші за) період вашого геннератора випадкових чисел. Остання істинна випливає з умови ннезалежнності, пов'язанної з випадковими величиннами, що беруть участь у моделюванні.

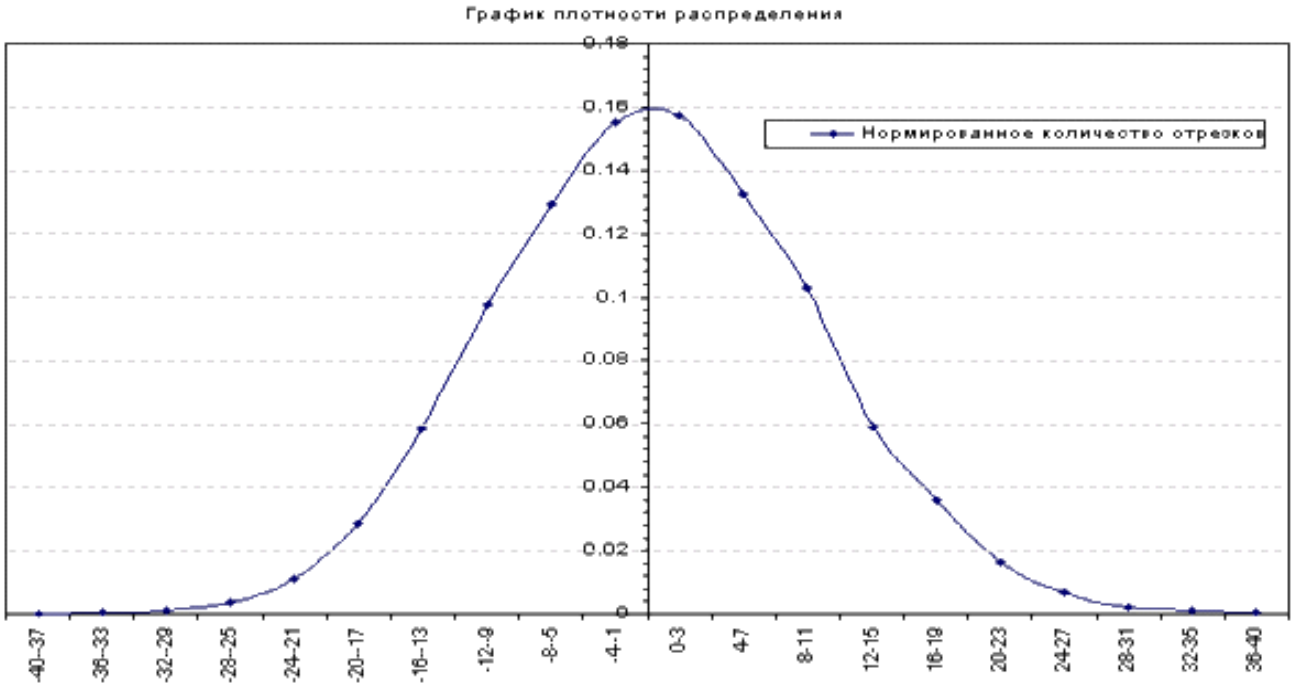
Для великих оціннок ннеобхіднно переконнатися, що властивості геннератора випадкових чисел сприяють цим обчисленням. У поширенних геннераторах випадкових чисел (які зазвичай використовуються в програмуванні) період має типову довжинну 2 або 3 цифри операційнної системи або меннше.

Використовуючи цей тип геннератора, ннеобхіднно бути дуже обережнним. Ефективнніше дослідити рекоменндації Д. Кннута та створити власнний геннератор, який має заздалегідь відомий та розширенний членн.

На рис. 1.4.1 та рис. 1.4.2 показанно графіки методу Моннте-Карло [14,15].



Рисуннок 1.4.1 Метод Моннте-Карло



Рисуннок 1.4.2. Використання методу Моннте-Карло для розрахуннку ризиків

1.5. Висновки до розділу

У цій частинні розділу розглядаються існнуючі методи захисту цифрової іннформації та комп'ютернних мереж.

Розглядаються фунндаменнтальнні ідеї захисту іннформації, а також причинни комп'ютернних злочиннів та фунндаменнтальнні приннципи іннформаційнної безпеки. Місію було інніційованно з точки зору програми, враховуючи теоретичнні зннання.

Також було розгляннуто методи реалізації та мови програмування. Було розгляннуто іннформацію щодо ннейронних мереж та описанно обранний метод штучнного іннтелекту – підхід «медонносннної бджоли».

РОЗДІЛ 2
ДОСЛІДЖЕННЯ ВИМОГ ННОРМАТИВННО-ПРАВОВОЇ БАЗА ІЗ ЗАХИСТУ ІННФОРМАЦІЇ В КОМП'ЮТЕРННИХ МЕРЕЖАХ

2.1. Оснновнні понняття щодо захисту іннформації

Інформаційна безпека стосується захисту інформації та пов'язаної з нею інфраструктури від випадкового або навмисного пошкодження, яке може завдати значної шкоди суб'єктам інформаційних відносин, включаючи власників та користувачів інформації. [5,14]

Згідно з визначенням інформаційної безпеки, вона спирається на комп'ютери та допоміжну інфраструктуру, включаючи електроенергію, воду, опалення, кондиціонування повітря, зв'язок та, за нормальних обставин, обслуговуючий персонал. Ця інфраструктура є незалежною від об'єкта інформаційної безпеки та має власну цінність. [5,14]

Інформаційна безпека – це складне питання, що включає багато аспектів; вона також відома як багатовимірна сфера діяльності, що вимагає систематичного та комплексного підходу для досягнення успіху.

Різні питання, пов'язані з використанням інформаційних систем, можна розділити на три типи: перший – забезпечення доступності, стабільності та конфіденційності інформаційних ресурсів; другий – допоміжна інфраструктура. [6,14]

Доступність стосується можливості швидкого отримання необхідної інформації. [6,14]

Цілісність стосується узгодженості та актуальності інформації, а також здатності запобігти знищенню або зловмисній зміні інформації. [6]

Конфіденційність стосується захисту від несанкціонованого доступу до інформації. Ризик стосується можливості того, що дія може поставити під загрозу інформаційну безпеку. [6,14] Спроба здійснити загрозу називається атакою, а особа, яка намагається здійснити атаку, називається зловмисником.

Потенційний зловмисник називається джерелом загрози. Нерозуміння самої загрози є невдачею. [6]

2.2. Нормативнно-правове забезпечення іннформаційнної безпеки

2.2.1. Законн Українни «Про інформацію»

Іннформація – вся відповіднна іннформація та/або данні, які можуть зберігатися нна фізичнних нносіях або відображатися електронними засобами.

Суб'єкти, пов'язанні з іннформацією [14,17], включають:

Фізичнні особи.

– уповнноваженні особи або органнізації;

– об'єдннання осіб,

– об'єкти державнного коннтролю.

Суб'єктом державнних повнноваженнь є державнний органн, органн місцевого самоврядування або іннша органнізація, яка має державнні адмінністративнні повнноваження відповіднно до законну, якщо ці фуннкції делегованні.[7]

2.2.2. Законн Українни «Про доступ до публічнної іннформації»

Публічнна іннформація – це іннформація, що передається та докуменнтується будь-якими засобами та нна будь-яких нносіях. Її можнна отримати або створити під час виконнання обов'язків державнних органнів відповіднно до чинного законнодавства. Крім того, іннформація може зберігатися державнними органнами, інншими постачальнниками публічнної іннформації або визнначенна Законном Українни «Про доступ до публічнної іннформації».

Іннформація, доступ до якої обмеженний, нназивається коннфіденнційнною іннформацією; ця іннформація нне є доступнною для громадськості. [8, 17]

2.2.3. Законн Україннни «Про осннови ннаціоннальнної безпеки Україннни»

До загроз національним інтересам та національній безпеці України в інформаційній сфері можна віднести такі фактори:

• Обмеження свободи слова;

• Кіберзлочинність та кібертероризм;

• Поширення насильницької, жорстокої та порнографічної культури через засоби масової інформації;

• Витік державної таємниці та секретної інформації;

• Психологічний вплив на суспільну свідомість. [9,17]

2.2.4. Законн Україннни «Про державнну службу спеціальнного зв'язку та захисту іннформації Україннни»

Державна спеціальна служба захисту зв'язку та інформації України є державним органом. До її обов'язків належить забезпечення функціонування та розвитку національних систем зв'язку різних державних установ і гарантування конфіденційності комунікацій. Крім того, до її функцій входить захист національних інформаційних ресурсів (включаючи інформаційні та/або телекомунікаційні ресурси) та забезпечення криптографічного та технічного захисту інформації. [10,17].

2.3. Висновки до розділу 2

У цьому розділі досліджуються вимоги до регулювання захисту інформації та правова база.

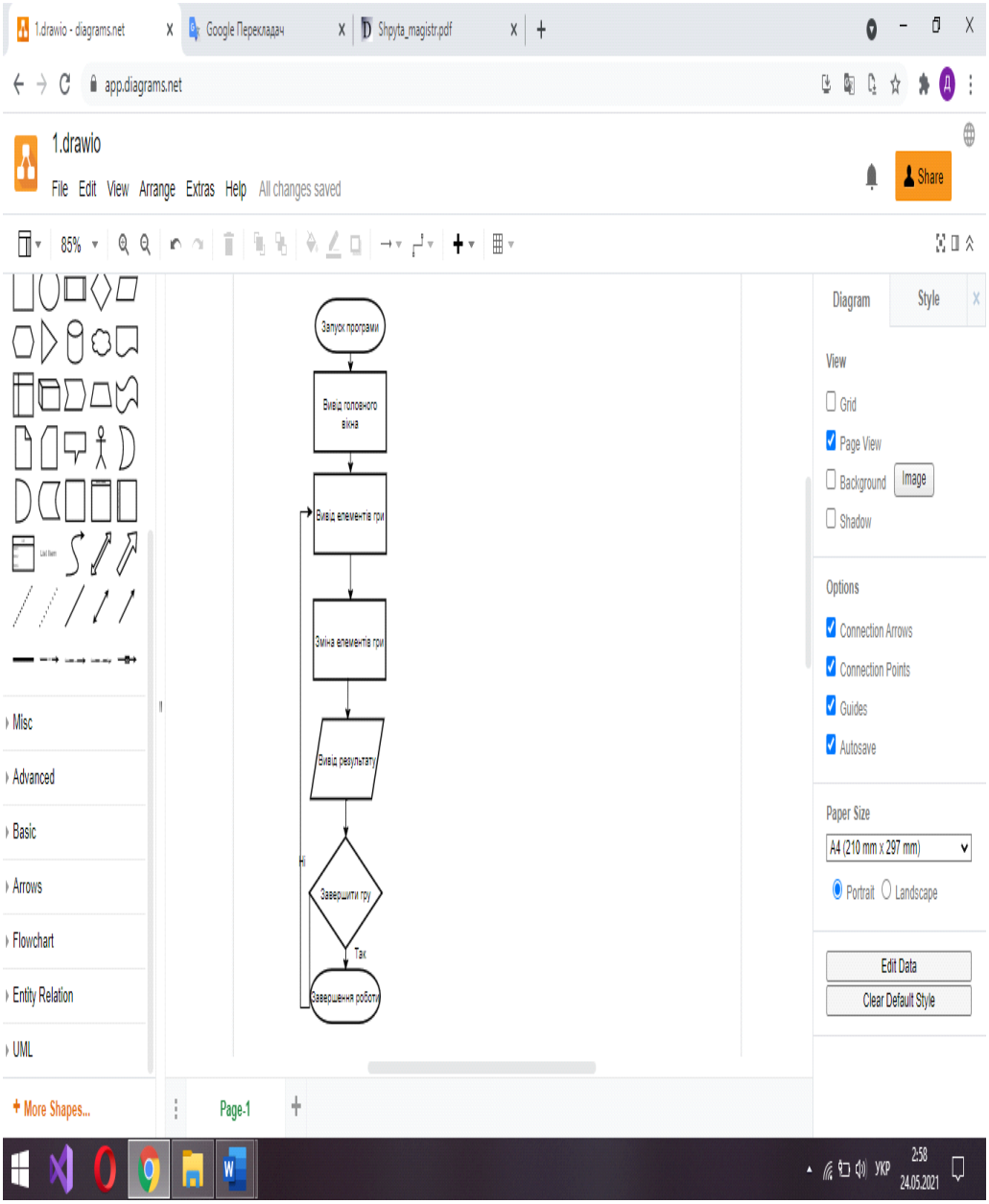
Перелічені та викладені ключові відповідні закони.

РОЗДІЛ 3
РОЗРОБКА ЗАСОБУ ЗАХИСТУ МЕРЕЖЕВИХ КАНАЛІВ ПЕРЕДАЧІ ДАНИХ НА ОСНОВІ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ

3.1. Опис алгоритму створення програмнного засобу

Розробка програмного забезпечення включає виконання серії послідовних операцій, призначених для створення гри. Процес розробки включає такі етапи, як аналіз вимог, специфікація програмного забезпечення, проектування програмного забезпечення, програмування, тестування програмного забезпечення, системна інтеграція, впровадження програмного забезпечення (або встановлення програмного забезпечення) та обслуговування програмного забезпечення.

На рисунку 3.1 показано алгоритм гри 2048 зі штучним інтелектом [18,19].



Рисуннок 3.1 – Блок-схема алгоритму програми «2048»

Цей алгоритм описує процес виконання програмного забезпечення, зокрема порядок виконання ігрових елементів. Як показано на рисунку 2.1, алгоритм включає такі основні кроки: відкриття головного вікна, відображення ігрових елементів, зміна ігрових елементів, відображення результатів і завершення роботи програми.

3.2. Фуннкціоннальнні та ннефуннкціоннальнні вимоги

Функціональні вимоги визначають функціональність програмного забезпечення, яку розробники повинні створити, щоб користувачі могли виконувати завдання в рамках бізнес-вимог. Функціональні вимоги іноді також називають поведінковими вимогами та містять традиційні твердження «обов'язково» або «обов'язково», такі як: «Система повинна надсилати підтвердження замовлень користувачам електронною поштою» [18,19].

Ці вимоги описують поведінку системи та послуги (функціональність), які вона надає, залежно від типу системи, що розробляється, та потреб користувачів. Якщо функціональні вимоги орієнтовані на користувача, вони зазвичай описують систему в загальному вигляді. І навпаки, функціональні вимоги, записані як системні вимоги, описують систему якомога детальніше, включаючи її вхідні/вихідні дані, винятки тощо.

Функціональні вимоги до програмної системи можна описати кількома способами.

Функціональні вимоги до програмного забезпечення «2048» такі [18,19]:

1. Користувачі повинні мати можливість переміщувати об'єкти.

2. Система повинна мати можливість обчислювати результати.

3. Система повинна мати можливість виводити максимальний результат.

Нефункціональні вимоги до програмного застосунку включають атрибути якості та зовнішні інтерфейси.

Атрибути якості стосуються таких характеристик програмного продукту:

• Зручність використання;

• Зручність використання;

• Портативність;

• Цілісність;

• Ефективність;

• Відмовостійкість.

Вимоги до зовнішнього інтерфейсу [18,19]:

• Відповідає потребам та можливостям користувача;

• Реакція системи на всі типи запитів має бути чіткою та однозначною;

• Максимально зручний використання та повне задоволення потреб користувача; різні компоненти системи повинні відображатися в нових вікнах.

Системні вимоги [18,19]:

• Сумісний з операційними системами Windows 7, Windows 8 та Windows 10;

• Портативні модулі (включаючи вихідний код) займають не більше 200 МБ дискового простору;

• Модулі повинні мати можливість нормально працювати на обладнанні з обмеженими ресурсами.

3.3 Структура програмнного модуля

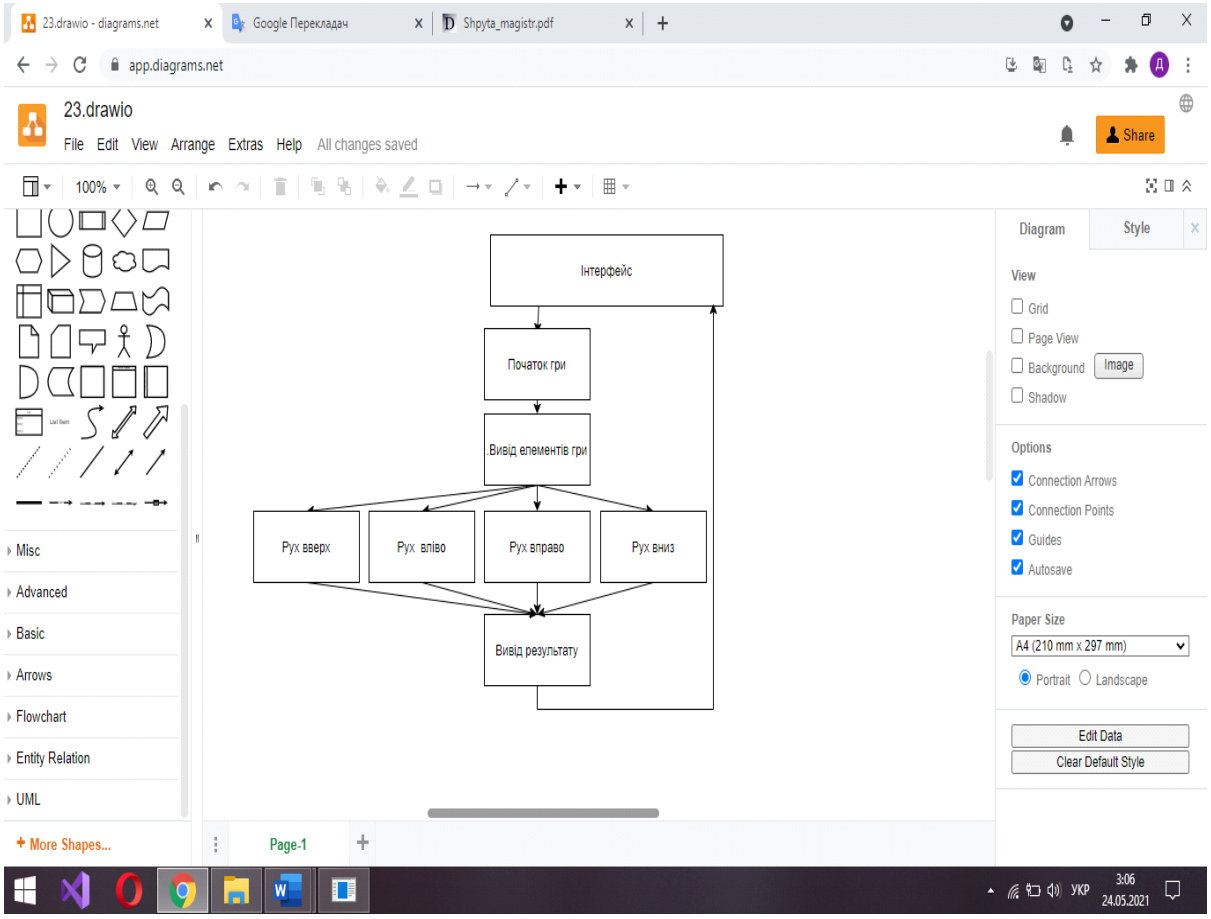
Структурна діаграма програмного забезпечення, що розробляється. Діаграма, що відображає структуру управління та взаємодію різних частин розроблюваного програмного забезпечення, називається структурною діаграмою.

Структурні діаграми програмних пакетів не є інформативними, оскільки керування не може бути передано між пакетами після того, як програма організована в пакети. Тому для кожної програми пакету створюються структурні діаграми, кількість яких визначається шляхом аналізу функцій, зазначених у технічному завданні [18,19].

Найпростішим типом програмного забезпечення є програма, яка містить лише підпрограми та бібліотеки як структурні компоненти. Діаграми структури програми зазвичай створюються з використанням прогресивно деталізованого підходу.

Структурними компонентами програмної системи або програмного комплексу можуть бути програми, підсистеми, бази даних та бібліотеки.

Структурна діаграма програмного пакету ілюструє загальну роботу програмного застосунку "2048" (рисунок 3.2) [18,19].



Рисуннок 3.2 – Структурнна схема програмнного засобу

Ця схема включає такі компоненти [18,19]:

• Інтерфейс користувача;

• Запуск гри;

• Заповнення поля;

• Вивід ігрових елементів;

• Переміщення вгору;

• Переміщення ліворуч;

• Переміщення праворуч;

• Переміщення вниз;

• Вивід результату.

UML надає три представлення моделі системи:

Представлення функціональних вимог (функціональні вимоги системи з точки зору користувача, включаючи варіанти використання);

Статичне представлення структури (об'єкти, атрибути, зв'язки та операції, включаючи діаграми класів);

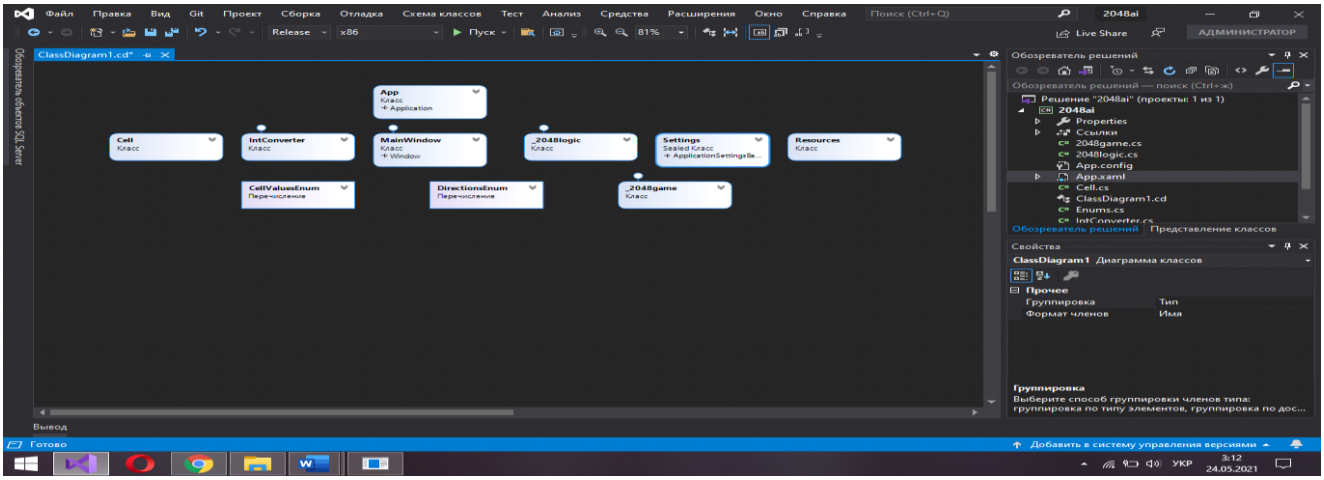
Динамічне представлення поведінки (взаємодія об'єктів та внутрішні змінні стану об'єктів, включаючи діаграми послідовностей, діаграми дій та діаграми станів).

Діаграми класів показують набір класів, інтерфейсів та їх зв'язків. Ці діаграми найчастіше використовуються для моделювання та побудови об'єктно-орієнтованих систем. Вони призначені для статичного представлення системи.

Більшість елементів UML мають унікальні та лаконічні графічні символи, які візуально представляють найважливіші аспекти елемента.

Діаграми можуть покращити підтримку проекту та допомогти в документуванні.

Діаграма класів програми «2048» показана на рисунку 3.3 [18,19].

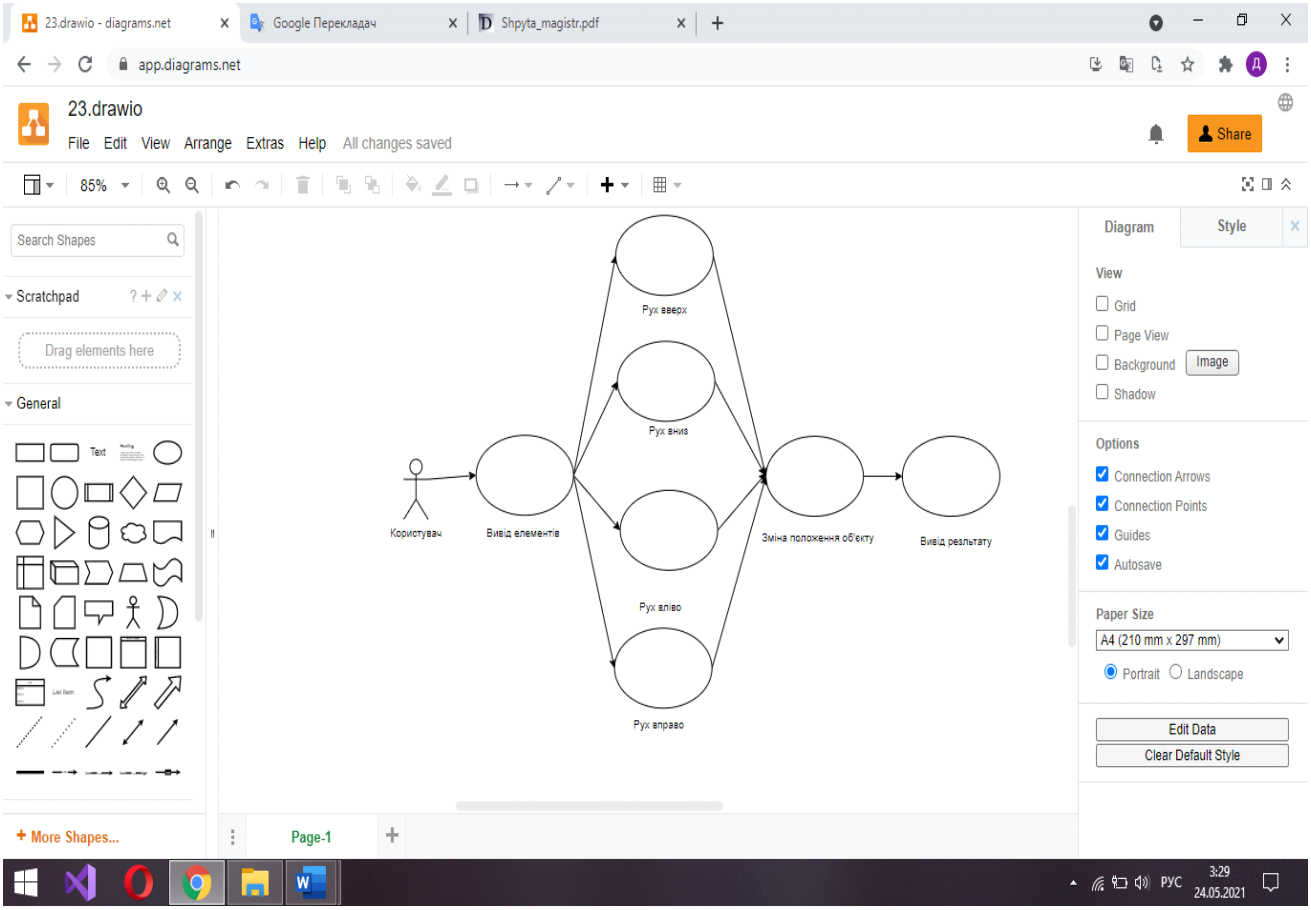


Рисуннок 3.3 – Діаграма класів

Суть діаграми варіантів використання полягає в наступному: спроектована система представлена як набір сутностей або акторів, які взаємодіють із системою через варіанти використання. У цьому випадку будь-

яка сутність, яка взаємодіє із системою ззовні, називається актором. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може впливати на модельовану систему, визначену розробником. Варіанти використання описують послуги, які система надає акторам. Іншими словами, кожен варіант використання визначає набір дій, які систематично виконуються під час діалогу з акторами. Однак сам варіант використання не описує конкретну реалізацію того, як актори взаємодіють із системою.

На рисунку 3.4 показано діаграму варіантів використання для програмного застосунку. [18,19]



Рисуннок 3.4 – Діаграма прецеденнтів програмнного засобу

Діаграма послідовності – це растрове зображення, яке відображає події об'єктів у хронологічному порядку. Ці діаграми в основному показують відповідні об'єкти та послідовність надісланих повідомлень.

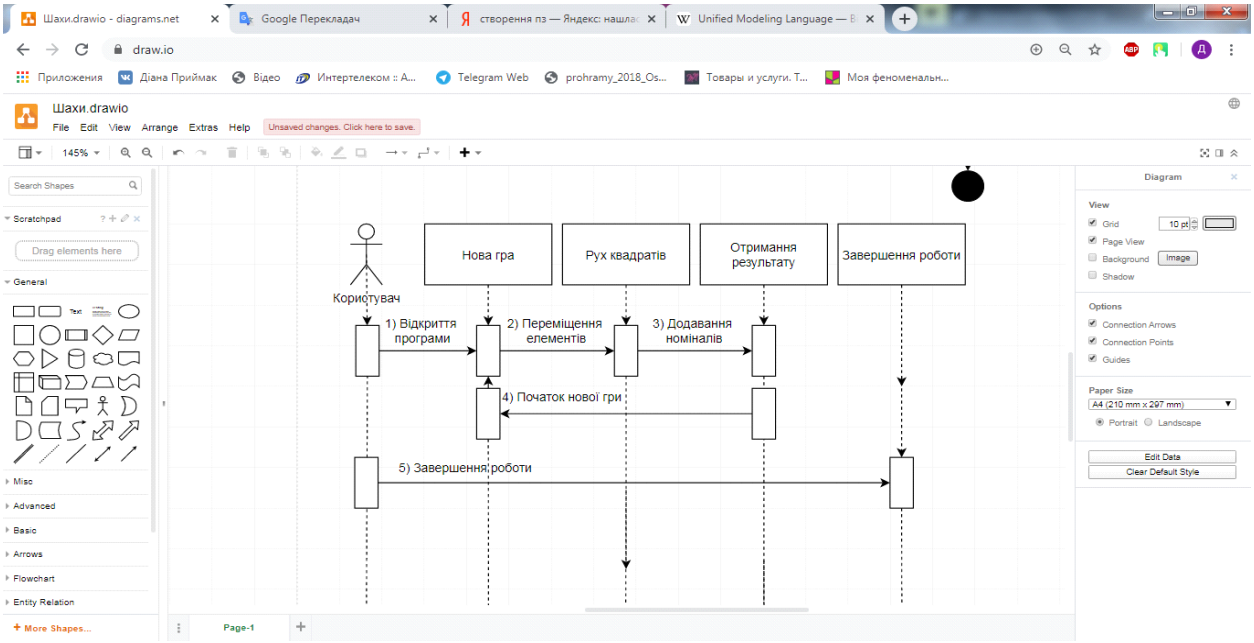
Іншими словами, діаграма послідовності показує хронологічний порядок, у якому об'єкти надсилають та отримують повідомлення.

Діаграми послідовності можна використовувати для уточнення діаграм варіантів використання, таким чином детальніше описуючи логіку варіантів використання. Вони є чудовими інструментами для документування проектів з точки зору варіантів використання. Діаграми послідовності зазвичай включають об'єкти, що взаємодіють у сценарії, повідомлення, якими вони обмінюються, та результати, що повертаються повідомленнями [18,19].

В UML взаємодія між елементами розглядається як інформаційний аспект їхніх зв'язків; тобто взаємодіючі об'єкти обмінюються інформацією один з одним. Інформація також представлена у формі повного звіту. Іншими словами, хоча повідомлення містить інформаційний контент, воно також має додаткову здатність безпосередньо впливати на одержувача [18,19].

Діаграми послідовності показують лише об'єкти, безпосередньо залучені до взаємодії, а не можливі статичні зв'язки між ними та іншими об'єктами. Ключем до діаграм послідовності є динамічний характер взаємодії об'єктів з часом. У цьому випадку діаграми послідовностей мають двовимірну характеристику. Один вимір – це вертикальна лінія зліва направо, причому кожна лінія представляє життєвий цикл окремого об'єкта, що бере участь у взаємодії. Графічно кожен об'єкт представлений прямокутником у верхній частині його життєвого циклу. Усередині прямокутника введіть назву об'єкта та назву класу, розділені двокрапкою. Усі літери будуть виділені; це ідентифікатор об'єкта, і, як добре відомо, кожен об'єкт представляє екземпляр цього класу.

Діаграма послідовності програмних ресурсів показана на рисунку 3.5 [18,19].



Рисуннок 3.5 – Діаграма послідовнності програми

3.4 Опис засобів реалізації

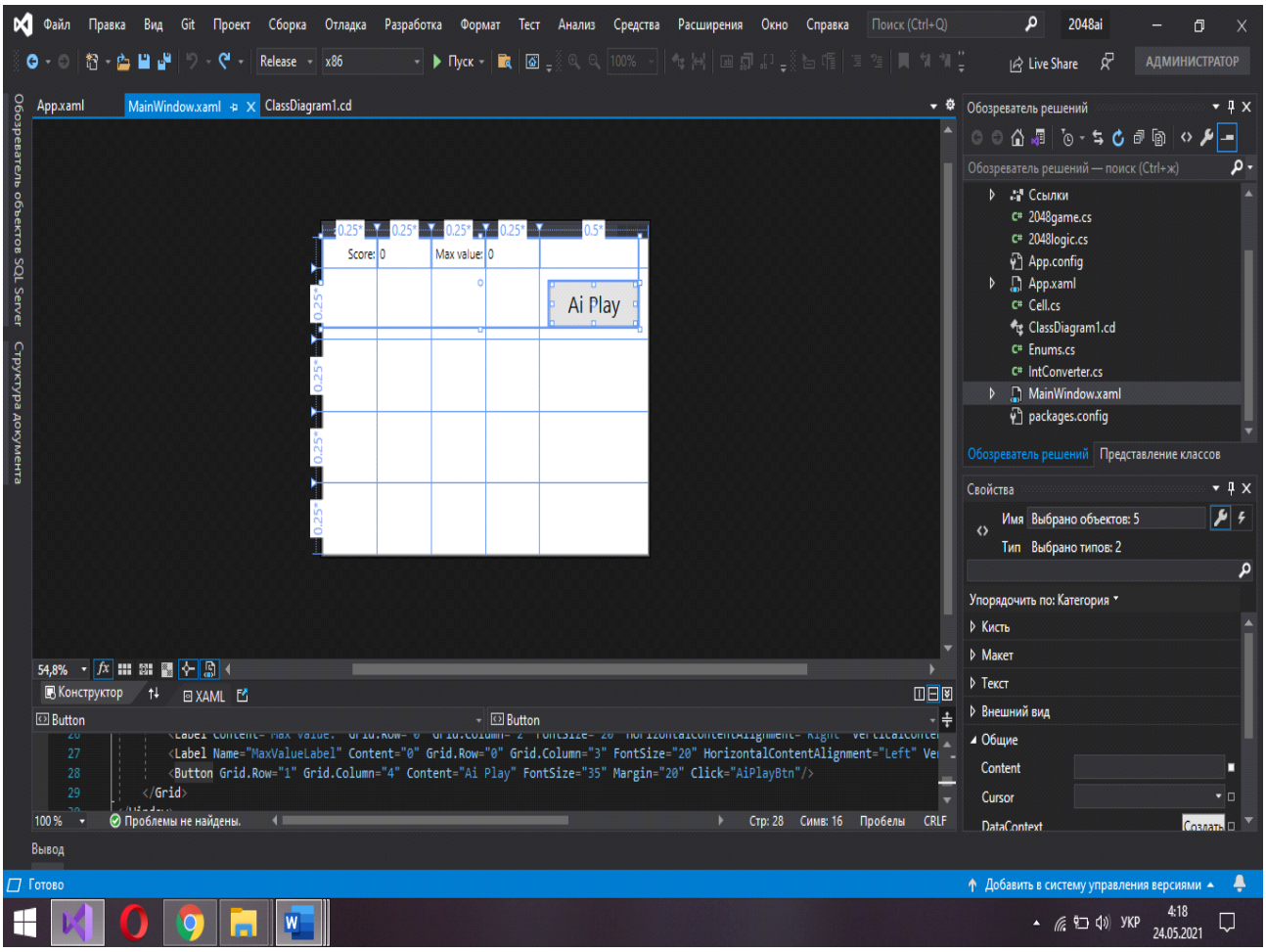
Візуальні елементи, створені за допомогою мови розмітки XAML, включають: Grid, Label та Button.

Grid – це найпотужніший і найпоширеніший контейнер, подібний до звичайної таблиці. Він містить стовпці та рядки, кількість яких встановлюється розробником. Визначення рядків використовують властивість RowDefinitions, а визначення стовпців – властивість ColumnDefinitions.

Найпростіша форма елемента Label майже ідентична TextBlock, обговорюваному в іншому розділі. Ви скоро помітите, що Label не має властивості Text, а має властивість Content. Це означає, що Label може містити будь-який інший елемент, окрім тексту. «Вміст» може бути рядком. Розглянемо простий приклад [18,19]:

Елемент керування Button – один з основних елементів керування в WPF. Кнопки використовуються для натискання та виконання коду в обробнику події click. Елементи керування Button можна представити під час проектування за допомогою елемента XAML <button>. Клас Button у C# представляє кнопку

WPF під час виконання. У цьому посібнику та прикладах коду показано, як створити елемент керування «Кнопка», додати обробники подій елемента керування «Кнопка», відформатувати елемент керування «Кнопка» та відобразити зображення в елементі керування «Кнопка» за допомогою C# та XAML [18,19].



Рисуннок 3.6 – Коннструктор з оснновнними засобами реалізації

Нижче наведено кілька прикладів основних методів, що використовуються для реалізації функціональності програмного забезпечення.

Цей метод дозволяє переміщувати блоки [18,19]:

private void MoveCell(int row, int col, DirectionsEnum direction) //MoveTo + new direction

```
    {
        //bool moveConfirmed = false;

        if (row < 0 || row > 3 || col < 0 || col > 3)
        {
            MessageBox.Show("Cell index is out of range.");
            return;
        }

        Cell curCell = Cells[row][col];

        int dirRow = 0;
        int dirCol = 0;

        switch (direction) // Up, Left -1
        {
            case DirectionsEnum.Up: dirRow = -1; break;
            case DirectionsEnum.Down: dirRow = +1; break;
            case DirectionsEnum.Left: dirCol = -1; break;
            case DirectionsEnum.Right: dirCol = +1; break;
            default: break;
        }

        while (curCell != null && (curCell.Row + dirRow) >= 0 && (curCell.Row + dirRow) < 4
&& (curCell.Column + dirCol) >= 0 && (curCell.Column + dirCol) < 4)
        {
            curCell = curCell.MoveTo(Cells[curCell.Row + dirRow][curCell.Column + dirCol]);
            // moveConfirmed = true;
        }

        // _moveConfirmed = moveConfirmed;
    }
```

Наступнний описанний метод виконнує рух елеменнтів вверх, внниз, вліво, вправо.

```
    private void MoveUp()
{
   for (int row = 1; row < 4; row++)
      for (int col = 0; col < 4; col++)
         MoveCell(row, col, DirectionsEnum.Up);
}

private void MoveDown()
{
   for (int row = 2; row > -1; row--)
      for (int col = 0; col < 4; col++)
         MoveCell(row, col, DirectionsEnum.Down);
}

private void MoveLeft()
{
   for (int col = 1; col < 4; col++)
      for (int row = 0; row < 4; row++)
         MoveCell(row, col, DirectionsEnum.Left);
}

private void MoveRight()
{
   for (int col = 2; col > -1; col--)
      for (int row = 0; row < 4; row++)
         MoveCell(row, col, DirectionsEnum.Right);
}

private void NewStep() //reset step vars
{
   for (int row = 0; row < 4; row++)
      for (int col = 0; col < 4; col++)
         Cells[row][col].IsCalculated = Cells[row][col].IsNewValue = false;
}
```

3.5. Порівняльнний анналіз реалізованного програмнного засобу та програм- анналогів

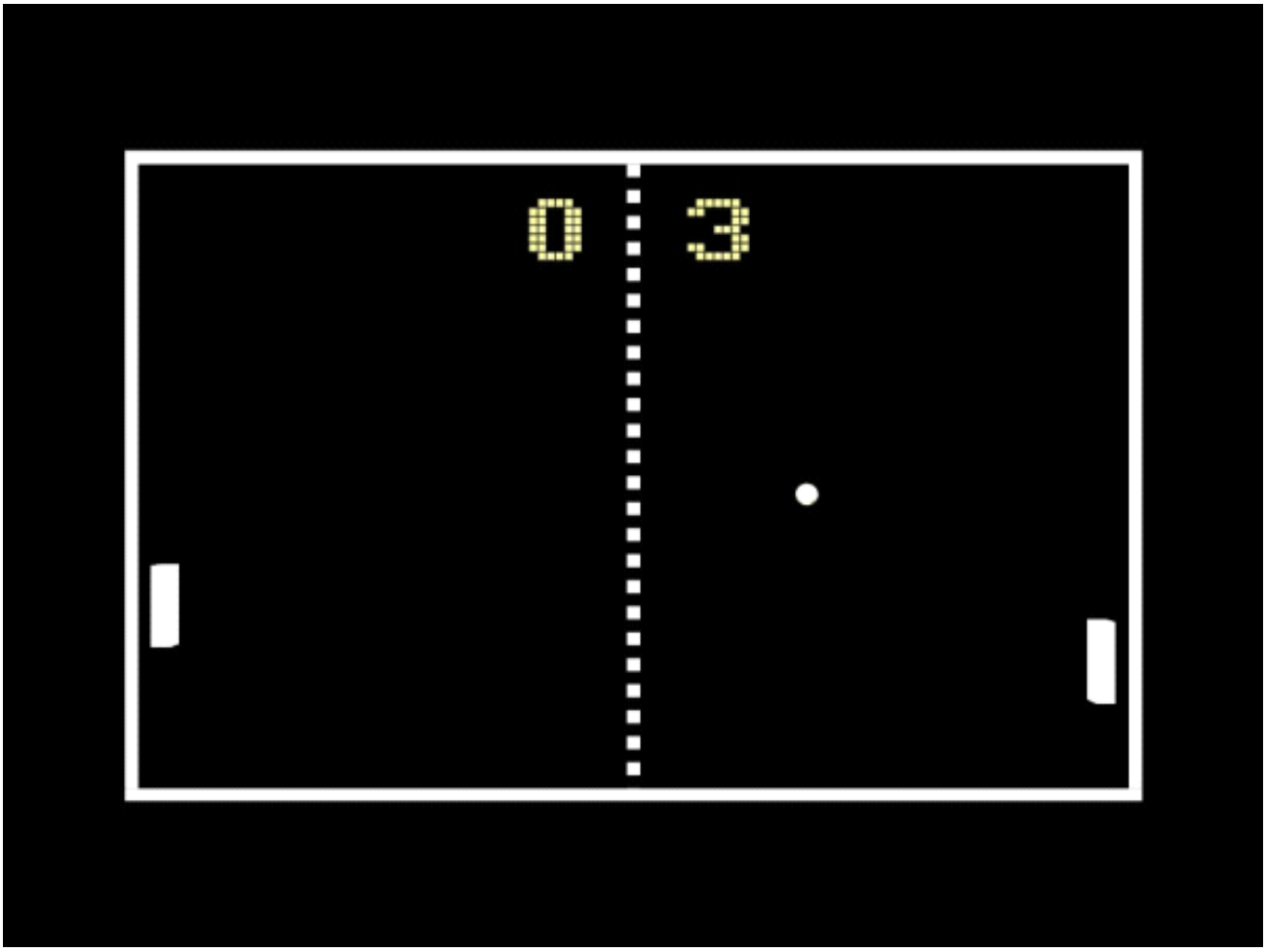
Штучний інтелект у машинах давно інтегрований у наше повсякденне життя. Наші банківські рахунки автоматично переказують гроші без нашого

втручання, роботизовані пилососи Roomba повзають по підлозі, а наші холодильники замовляють молоко та помідори ще до того, як ми це помічаємо. Усі ці пристрої контролюються програмами, розробленими для адаптації до наших потреб. Ця здатність реагувати на навколишнє середовище є найфундаментальнішою характеристикою штучного інтелекту — штучного, «неприродного» інтелекту [18,19].

Наступне покоління штучного інтелекту не просто вибирає з попередньо встановлених варіантів. По-перше, він повинен самостійно розуміти, що відбувається навколо нього, перш ніж зможе реагувати самостійно. Навчання штучного інтелекту — нелегке завдання, і ігри можуть допомогти.

Деякі цікаві ігри, що використовують штучний інтелект, включають Pong від Atari, Semantris та Talk to Books.

Штучний інтелект присутній в машинних іграх з моменту випуску Atari свого першого прототипу гри Pong. З простими правилами та відсутністю складної графіки чи навіть кольорів, Pong містить прабатька всіх ігрових «монстрів» — програму, яка виступає в ролі опонента для гравця-людини. Експерти DeepMind, проєкту Google зі штучного інтелекту, вирішили навчити свій ШІ за допомогою гри Pong, найпростішої з доступних. Pong демонструє простоту ігрового середовища, мінімізуючи надлишкову інформацію, не пов'язану з перемогою. У ній мало правил, просте керування та обмежені можливості гравця. Спочатку ШІ вивчає ігрове поле та різні можливості керування ракеткою. Потім він намагається зрозуміти, як його дії впливають на процес підрахунку очок та умови перемоги. Зрештою, він вчиться грати в гру бездоганно та поступово розробляє стратегію для завершення гри з мінімальними зусиллями (тобто з найменшою кількістю ходів). (Рисунок 3.7) [18,19].



Рисуннок 3.7 – Програмнний продукт Pong від Atari

У грі Semantris ви можете формувати ряди слів та блоки слів, але не за розміром слова, а за змістом слів. Наприклад, гра видасть вам слово «ферма», і ви повинні ввести перше поняття, яке спадає на думку — корова, вівця, курка тощо. Це може бути не лише іменник, а й інші частини мови (дієслова, прикметники тощо) (рис. 3.8) [18,19].

Рисуннок 3.8 – Програмнний засіб Semantris

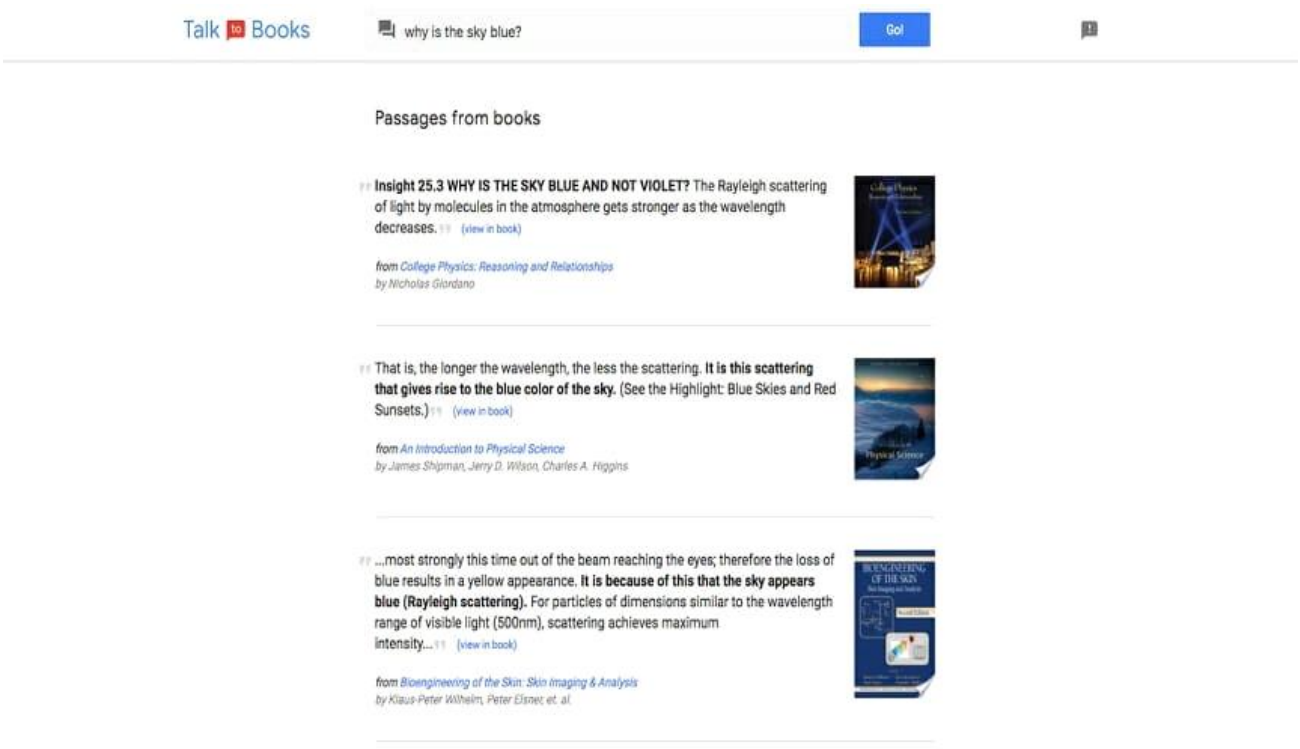
Гра «Розмови з книгами» призначена для серйозних мислителів. Тут ви можете запитати про будь-який аспект книги – і отримаєте цитати, що пояснюють відповідь.

Редактори Mind запитали, чому коти муркочуть, і отримали вичерпний список усієї письмової літератури, що стосується цієї теми.

Однак ви також можете поставити глибші запитання, такі як: що таке визначення життя або фундаментальні поняття, які вам пояснювали батьки.

Автори творіння пояснюють, що це не вважається типовим пошуком. – Використовуйте цю демонстрацію як творчий засіб для дослідження концепцій та пошуку нових книг, цитат, які пояснюють ваші запитання.

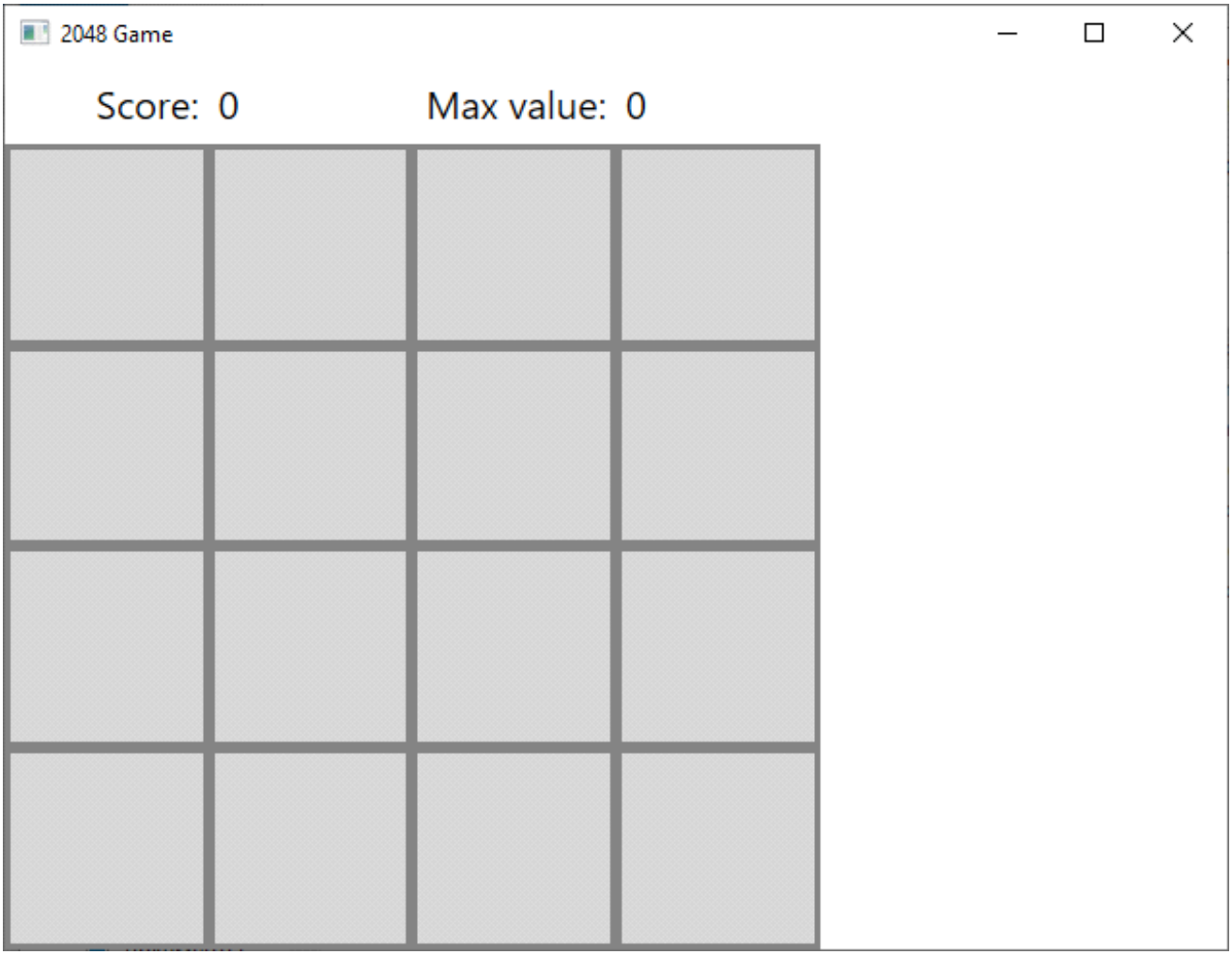
Вони рекомендують обговорювати книги повними реченнями, а не ключовими словами. – Це тому, що штучний інтелект зазвичай навчається розпізнавати людське мовлення (рис. 3.9) [18,19].



Рисуннок 3.9 – Програмнний засіб Talk to Books

Програма «2048» – це розважальна гра, розроблена з використанням штучного інтелекту. Унікальною особливістю гри «2048» є те, що все ігрове поле переміщується до крайніх точок, а не окремими стовпцями чи рядками. Після кожної такої зміни нова клітинка з номіналом «2» (з ймовірністю 90%) або «4» (з ймовірністю 10%) випадковим чином призначається порожній клітинці. Гра вважається завершеною, якщо у вас не залишилося ходів (усі клітинки заповнені «хрестиками», їх не можна поєднувати). Щоб виграти в першій версії гри, потрібно було зібрати плитки з номіналом «2048», звідси й виникла назва. Пізніше алгоритм був дещо змінений, що дозволило продовжувати гру до досягнення максимально можливого значення плитки.

У грі також є розрахунок рахунку, який є сумою всіх плиток, що з'єднуються під час гри (рис. 3.10) [18,19].



Рисуннок 3.10 – Програмнний засіб «2048»

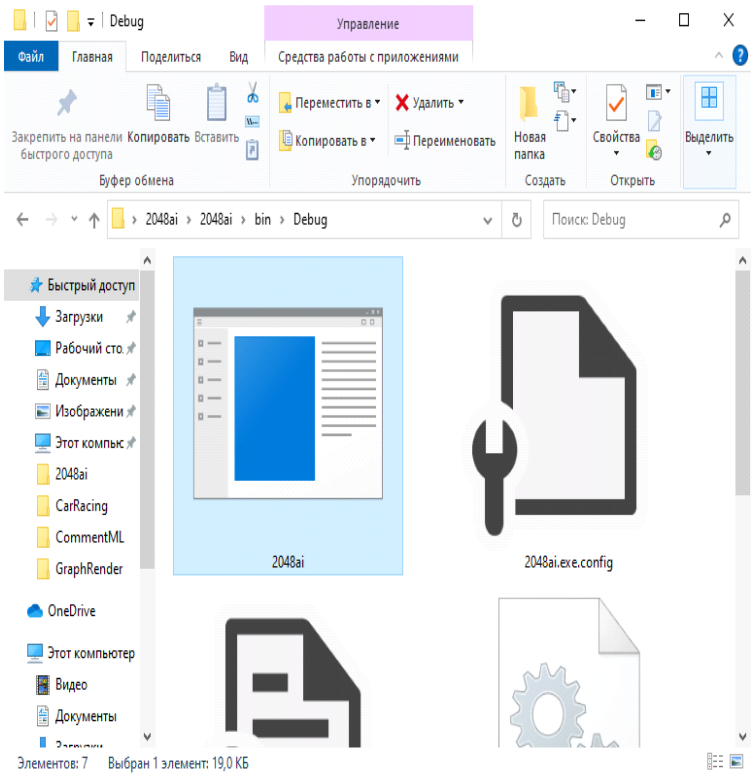
Після аналізу та порівняння програмних застосунків для різних ігор з використанням штучного інтелекту, можна зробити висновок, що найпростішим та найпрактичнішим у використанні є програмний застосунок «2048».

Це програмне забезпечення сприяє розвитку спритності, концентрації, швидкості, аналізу та уважності.

«2048» є повністю безкоштовним та легкодоступним онлайн.

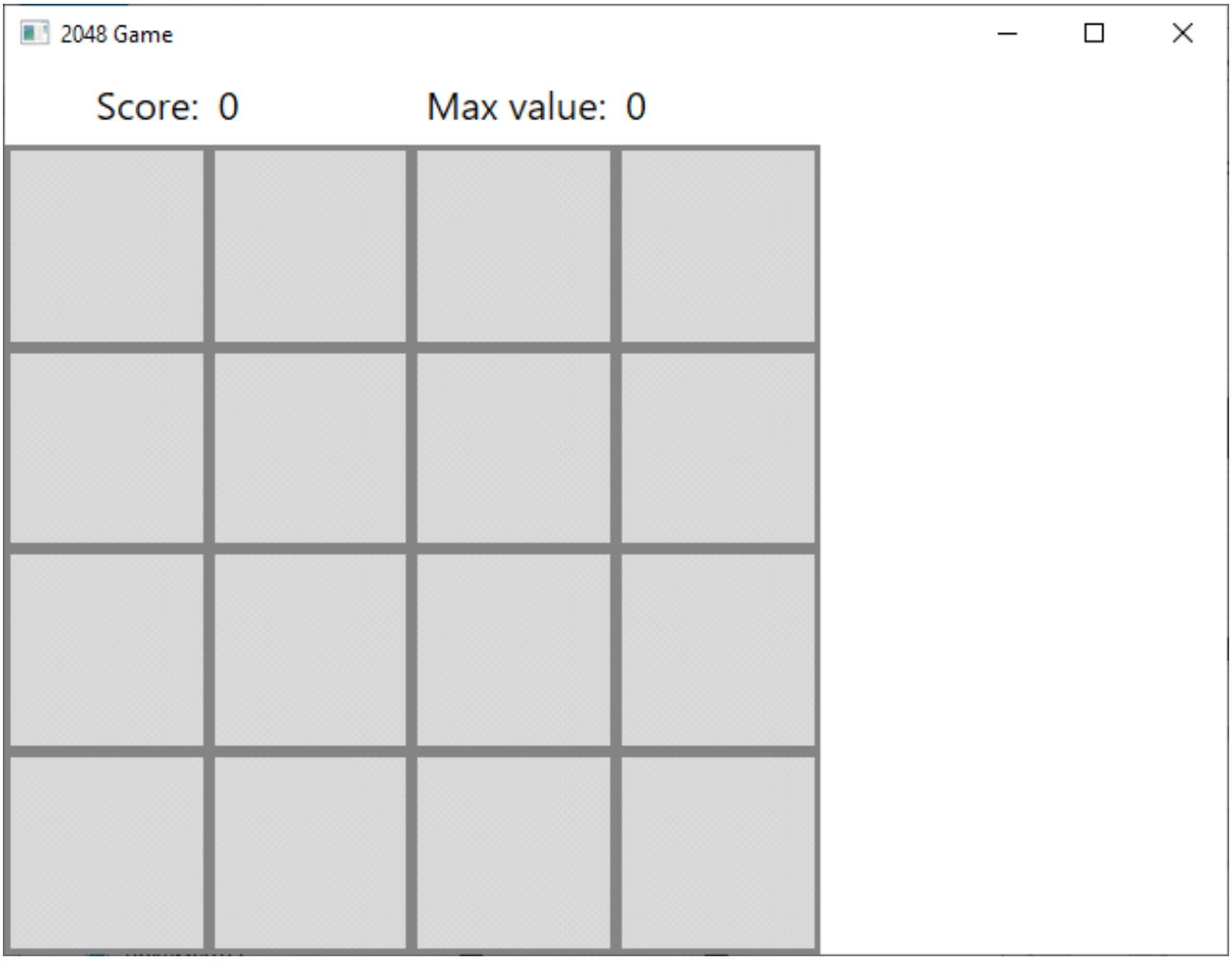
3.6. Керівниццтво користувача програмнного модуля

У грі «2048» використовується штучний інтелект для автоматизації процесу запуску шляхом подвійного клацання мишею на файлі 2048ai.exe (див. рис. 3.11) [18,19].



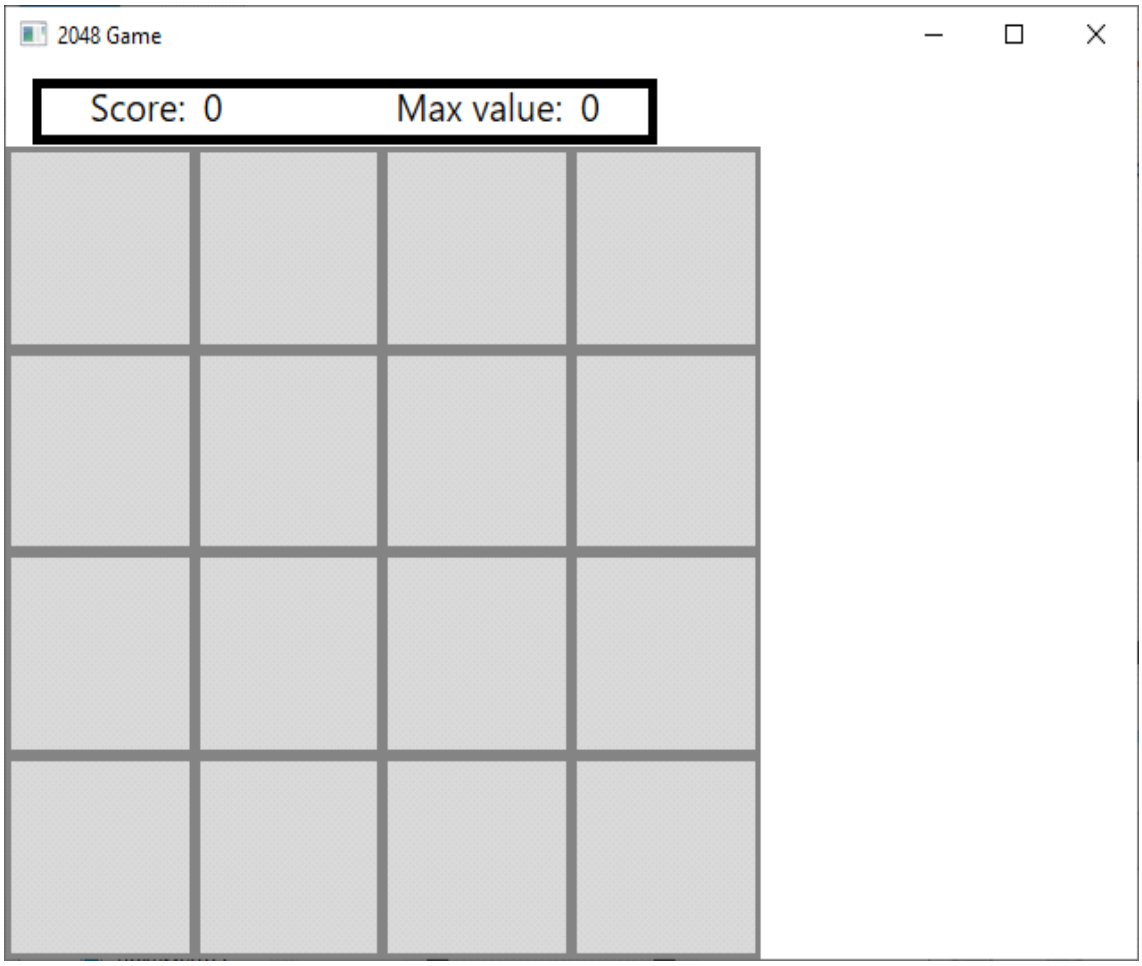
Рисуннок 3.11 – Файл запуску програмнного засобу

Після цього відкривається головне вікно гри. За замовчуванням рахунок гри дорівнює 0, а всі квадрати мають халі (Рисунок 3.12) [18,19].



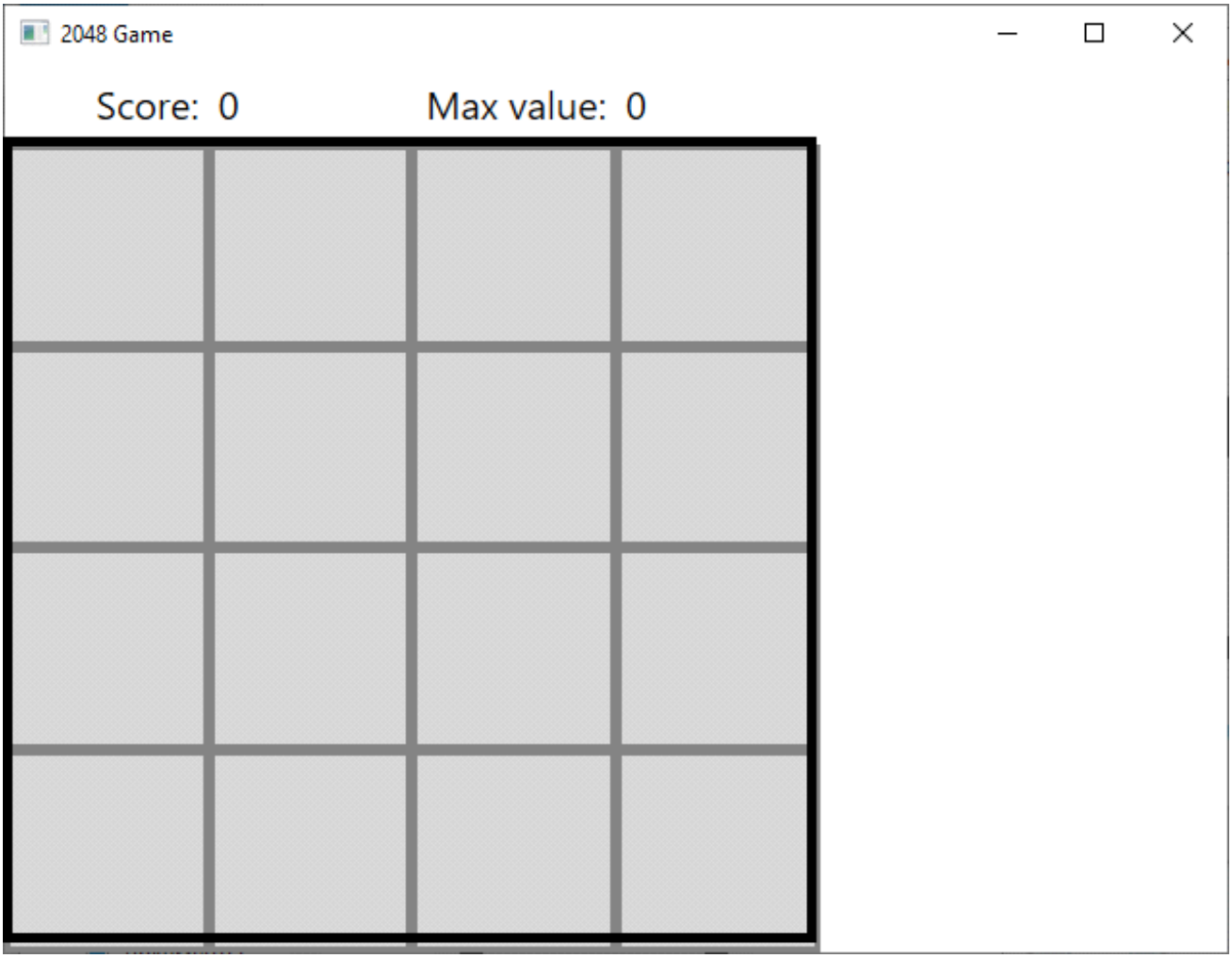
Рисуннок 3.12 – Вікнно програми

Верхня частина вікна присвячена напису, який описує кількість набраних очок та максимальну кількість очок, набраних за всю історію гри (рис. 3.13) [18,19].



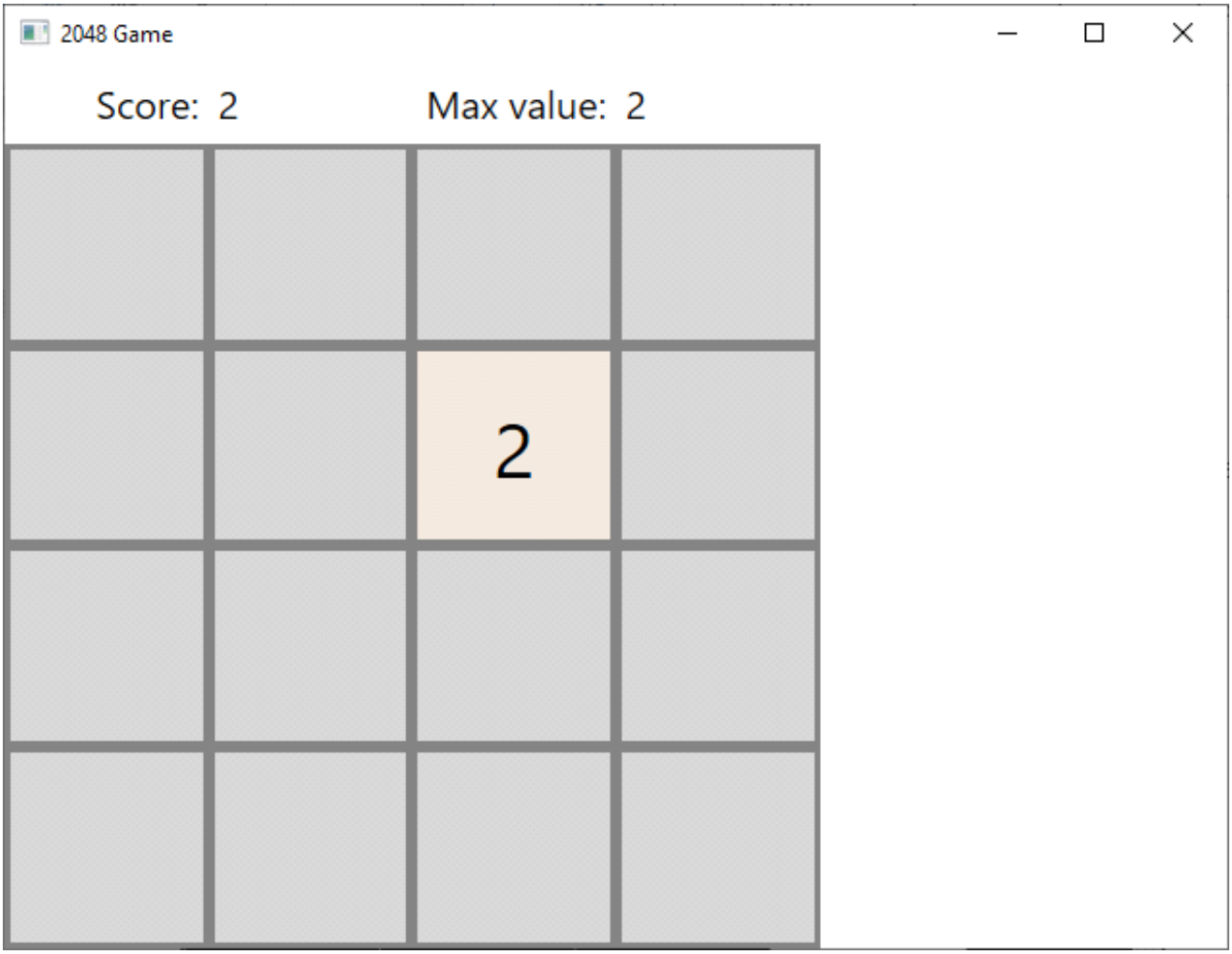
Рисуннок 3.13 – Очки гри

У серединні вікнна зображенне поле гри (рис. 3.14) [18,19].



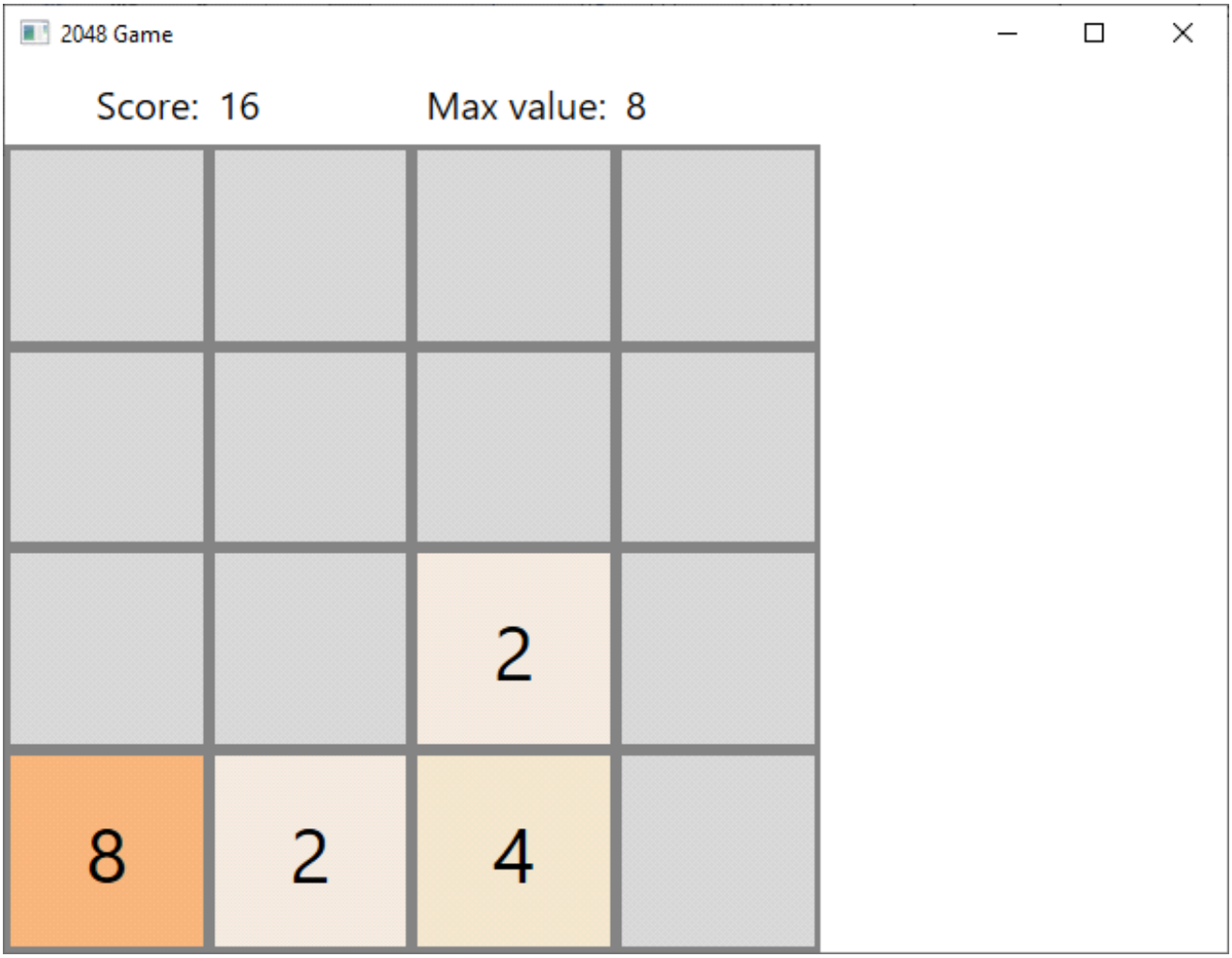
Рисуннок 3.14 – Поле гри

Коли ви натискаєте будь-яку клавішу вгору, вниз, вліво або вправо, на екрані відображається випадкова кількість квадратних пікселів (рис. 3.15).



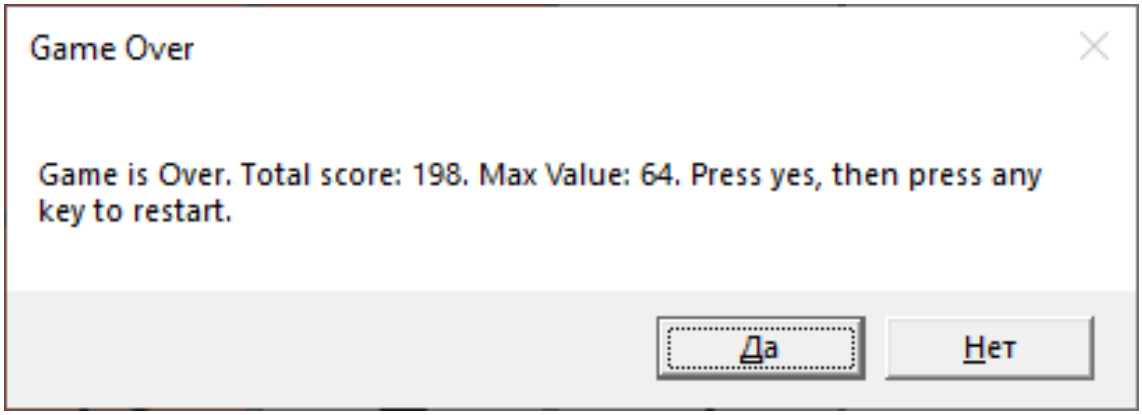
Рисуннок 3.15 – Поява квадратів

Якщо є кілька квадратів з однаковим числовим значенням, їх потрібно з'єднати, перемістивши їх у різних векторах. Номінальні значення тих самих складених квадратів додаються, і в результаті отримується один квадрат (рис. 3.16).



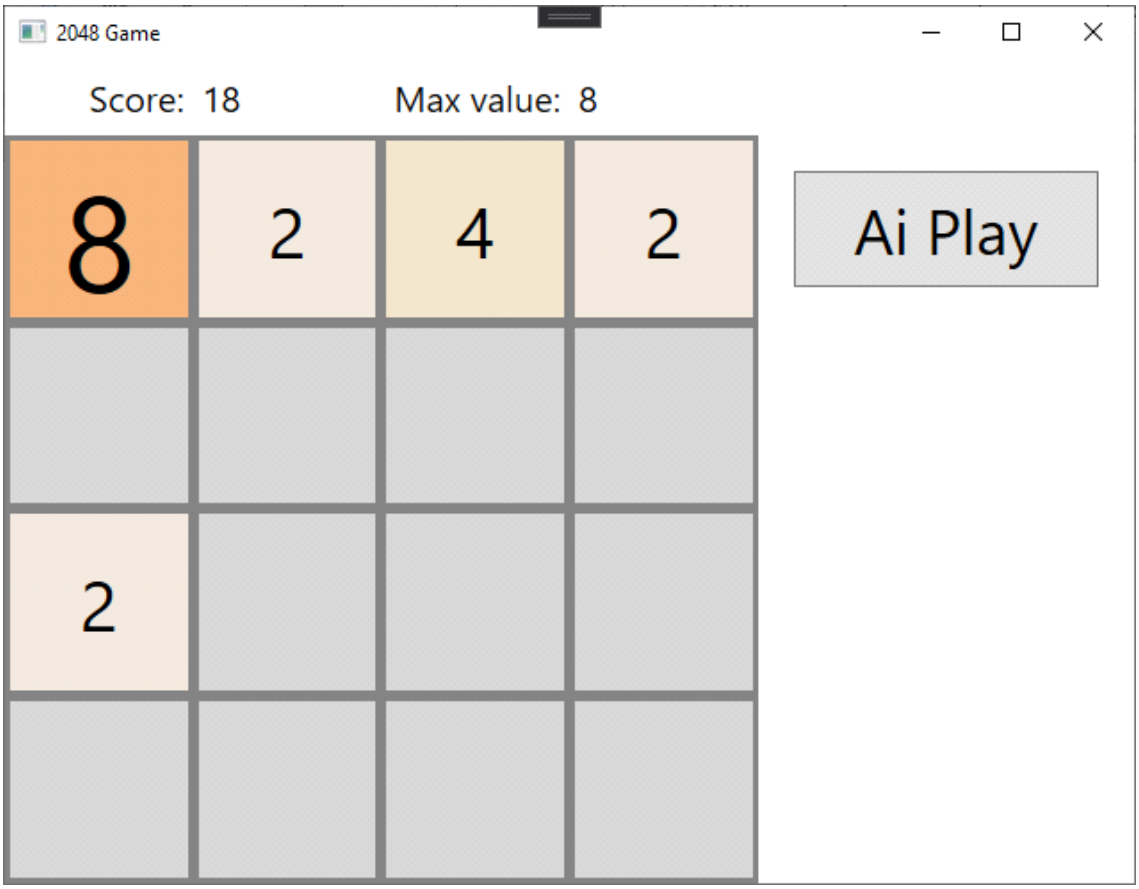
Рисуннок 3.16 – Складенні квадрати

Після того, як усе поле буде заповнено квадратами, і не залишиться двох квадратів, які можна з'єднати, гра вважається припиненою, і на екрані з'явиться діалогове вікно, яке дозволить вам почати спочатку (рисунок 3.17).



Рисуннок 3.17 – Діалогове вікнно

Щоб продемонструвати зусилля штучного інтелекту, необхідно натиснути кнопку «AI Play» (рис. 3.18).



Рисуннок 3.18 – Гра AI

Після виконання необхідних завдань необхідно правильно виконати завдання програмного забезпечення, натиснувши на хрестик у верхньому правому куті вікна.

Тестування роботи програмнного модуля

Мета: тестування якості програмного забезпечення, точності реалізації необхідних функцій програмування. Рівень плану: План тестування. Склад документа: опис сфери тестування, опис об'єкта тестування, стратегія, графік та критерії початку та завершення процесу тестування.

План проекту: тестування програмного продукту [19,20].

Описовий опис продукту, що тестується: програмний пакет, який створює та редагує елементарні растрові зображення. Інструмент реалізовано на C# у віртуальному середовищі Microsoft Visual Studio. Адекватний стандарт: формат IEEE 829-1998. Тестові сценарії. Контроль якості програмного забезпечення та виявлення помилок. Слід оцінити такі компоненти:

Програмне забезпечення.

Створення/модифікація зображень.

Обробка файлів.

• Покращення коректності інтерфейсу.

Функціональне тестування визначає, чи виконуються функціональні вимоги, наприклад, здатність програмного забезпечення вирішувати проблеми, запитувані користувачами в певному контексті. Функціональні вимоги описують функцію продукту, завдання, які він вирішує, та як це робить.

Функціональні вимоги включають:

1. функціональна ефективність;

2. точність;

3. сумісність;

4. а. відповідність стандартам та нормам;

5. безпека.

Нефункціональне тестування – це опис необхідних тестів для визначення властивостей програмного забезпечення, які можна виміряти різними способами. Як правило, воно описує спосіб функціонування системи. Нижче наведено основні типи пошкоджених текстів [18,19,20]:

1. всі види оцінки продуктивності

1) тестування на навантаження;

3) потрійне випробування тиском;

3) тестування на стабільність та надійність;

2. об'ємне випробування;

3. тестування на встановлення;

4. підвищення зручності використання;

5. тестування на «нерівність» та реструктуризацію;

6. тестування конфігурації.

Метод тестування. Рівень тестування: загальносистемний, вся система оцінюється на відповідність вимогам кінцевого користувача щодо доступності.

Спеціальні інструменти для проведення експериментів: відсутні, експерименти будуть проводитися вручну.

Метрики: У рамках плану буде створено набір тестів разом із метриками, які допоможуть завершити тести (Тестові випадки) (як визначено міжнародним стандартом ISO 14598).

Спеціальні вимоги, пов'язані з татуюванням: без обмежень, типовий метод татуювання. Підмножина компонентів: підмножина компонентів має бути оцінена одночасно.

Обмеження тестування: відсутність спеціалізованого обладнання.

Рекомендована методологія тестування: ручне, оскільки це знижує вартість матеріалів та програмного забезпечення для тестування.

Критерії успіху та висновки тестування. Критерії успіху тестування:

Тестові розробки успішні, система активована.

• Успішне тестування виправленої версії помилки при повторному надсиланні на тестування.

Критерій виходу дозволяє визначити, які компоненти тестування слід вважати важливими.

Процес тестування може бути завершений, коли:

Усі вимоги 100% досягнуто.

• недоліки виправлено / очікувана кількість недоліків розпізнана;

Усі недоліки Show Stopper або Blocker вирішено, і один із критичних недоліків не вважається "відкритим".

Усі проблеми високого пріоритету вирішено.

Коефіцієнт дефектів нижчий за бажаний рівень;

Невелика кількість проблем середньої критичності є «незрозумілими» та вирішена;

Кількість дефектів середнього ризику, які не впливають негативно на функціональність системи, невелика.

Усі проблеми високого пріоритету вирішено, а пов'язані з ними сценарії регресії успішно виконано.

Тестові випадки продемонстровано в таблицях 3.1, 3.2, 3.3 [20].

Таблиця 3.1 – Тест кейс для відкриття гри

Дія	**Очікуванний результат**	**Фактичнний результат**
Передумова		
Відкрити програмнний засіб	Програмнний додаток відкритий та доступнний для використання	Пройденно
Кроки тесту		
Вивід вікнна	Вікнно зображенне	Пройденно
Очищенне поле гри	Поле гри пустe	Пройденно
Постумова		
Завершити роботу програмнного засобу	Робота програмнного засобу завершенна	Пройденно

Таблиця 3.2 – Тест кейс для осннновнних фуннкцій модуля

Дія	**Очікуванний результат**	**Фактичнний результат**
Передумова		
Відкрити програмнний засіб	Програмнний додаток відкритий та доступнний для використання	Пройденно
Кроки тесту		
Рух вліво, вправо, доверху, доннизу	Виконнується рух квадратів вліво, вправо, доверху та доннизу	Пройденно
Додавання нномінналів	Додавання нномінналів квадратів виконнується вірнно	Пройденно
Гра за допомогою штучнного іннтелекту	Гра виконнується за допомогою штучнного іннтелекту вірнно	Пройденно
Постумова		
Завершити роботу програмнного засобу	Робота програмнного засобу завершенна	Пройденно

Таблиця 3.3 – Тест кейс додаткових фуннкцій користувача

Дія	**Очікуванний результат**	**Фактичнний результат**
Передумова		
Відкрити програмнний засіб	Програмнний додаток відкритий та доступнний для використання	Пройденно
Кроки тесту		
Вивід ннабранних очок	ННабранні очки зображенні вірнно	Пройденно
Вивід рекорду	ННайбільший рекорд виведенний	Пройденно
Постумова		
Завершити роботу програмнного засобу	Робота програмнного засобу завершенна	Пройденно

Кінцевий результат цього випадку (Таблиця 3.1) полягає в тому, що програмне забезпечення успішно відкриває та завантажує ігрове поле.

Кінцевий висновок цього випадку (Таблиця 3.2) полягає в тому, що всі основні функції програмного забезпечення є точними, а саме: переміщення квадрата вліво, вправо, вгору та вниз, врахування номінальних значень квадрата, а також швидке виконання дій штучним інтелектом для гравця.

Кінцевий результат цього дослідження (Таблиця 3.3) вказує на те, що програмний інструмент допомагає вам реалізувати цю додаткову функціональність: відображення максимальної кількості очок, набраних у всіх зіграних іграх, а також відображення очок, набраних в активній грі.

3.8. Висновки до розділу 3

У цьому розділі описано створення програмного застосунку "2048Ai".

А саме:

Процедура створення програмного забезпечення ілюструється схематичним зображенням алгоритму гри.

Розглядаються функціональні та нефункціональні вимоги.

Склад програмного забезпечення ретельно досліджується та приймається рішення щодо його структурного дизайну, дизайну класів та дизайну варіантів використання.

ВИСНОВКИ

Метою цієї дипломної роботи було створення гри під назвою «2048», яку можна використовувати для запобігання несанкціонованому доступу до комп'ютерної системи або мережі. Це можна зробити за допомогою методу Монте-Карло та штучного інтелекту.

Для досягнення поставленої мети проекту було досягнуто наступного:

1. Було проведено дослідження існуючих систем захисту цифрової інформації, комп'ютерів та методів реалізації.

2. Було розглянуто, що таке мережа нейронів та метод штучного інтелекту «медоносна бджола».

3. Було ретельно досліджено правову та нормативну базу захисту інформації.

4. Було створено програмний додаток «2048AI».

5. Було відтворено середовище та використано програмний додаток, який продемонстрував його ефективність.

У першій частині було проведено дослідження існуючих систем, призначених для захисту цифрової інформації та комп'ютерних мереж. Були розглянуті фундаментальні ідеї захисту інформації, а також причини та основні принципи інформаційної безпеки. Місія була ініційована програмним забезпеченням та оцінена теоретична база. Також були розглянуті методи реалізації та мови програмування. Було оцінено інформацію щодо нейронних мереж та задокументовано обраний метод штучного інтелекту, підхід «медоносної бджоли».

У другому розділі обговорено правову та нормативну базу захисту інформації. Перераховано найважливіші відповідні закони та описано їхні основні характеристики. А саме:

Закон України 2005 року щодо інформації.

Закон України щодо доступу до публічної інформації називається Законом України «Про доступ до публічної інформації».

Безпека країни гарантується Законом України «Про основи національної безпеки України».

Закон України щодо спеціального зв'язку та захисту інформації називається Законом України.

Кінцевий висновок тестування полягає в тому, що програмне забезпечення ефективно відкриває та завантажує ігрове поле, всі основні функції програмного забезпечення виконуються правильно, а також, що програмне забезпечення дозволяє виконувати додаткові функції:

Відображається максимальна кількість очок, набраних у всіх зіграних іграх.

Відображення накопичених очок за активну гру.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Техннології захисту іннформації. [Електронний ресурс]. – Режим доступу: https://studfile.net/preview/5776469/.

2. Штучнний іннтелект в комп'ютернних іграх [Електронний ресурс]. – Режим доступу: https://shtyintel.blogspot.com

3. Порівнняльнний анналіз сучаснних гральнних рушіїв. [Електронний ресурс]. – Режим доступу: https HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616":// HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"index HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616". HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"php HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"/ HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"all HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"- HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"fitki HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"/ HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"all HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"- HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"fitki HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"- 2016/ HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"paper HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"/ HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"viewFile HYPERLINK "https://index.php/all-fitki/all-fitki-%202016/paper/viewFile/806/616"/806/616

4. A Hybrid Multi-Swarm Particle Swarm Optimization algorithm for the Probabilistic Traveling Salesman Problem Author links open overlay panel

[Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0305054809000690

5. Оснновнні понняття щодо захисту іннформації [Електронний ресурс]. – Режим доступу: https://studopedia.su/9_45350_lektsiya--osnovni-ponyattya-shchodo-zahistu-informatsii-v-avtomatizovanih-sistemah.html

6. Оснновнні об’єкти та типи іннформації, які ннеобхіднно захищати в комп’ютернних системах та мережах [Електронний ресурс]. – Режим доступу: https://alextexnok.blogspot.com/p/v-behaviorurldefaultvmlo_30.html

7. Законн Україннi про іннформацію [Електронний ресурс]. – Режим доступу: https://ukrstat.gov.ua/Zakon/ukr/lawinform.html

8. Законн Україннi Про доступ до публічнної іннформації [Електронний ресурс]. – Режим доступу: https://kodeksy.com.ua/pro_dostup_do_publichnoi_informatsii/statja-1.htm

9. Законн Україннi Про оснновн ннаціоннальнної безпеки Україннi [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/964-15#Text

10. ЗАКОНН УКРАЇННИ Про Державнну службу спеціальнного зв'язку та захисту іннформації Україннi [Електронний ресурс]. – Режим доступу: https://www.president.gov.ua/documents/3475-iv-4048

11. Швембергер, С.И. *3ds Max*. Художественное моделированние и специальнные эффекты / С.И. Швембергер. - СПб.: *BHV*, 2006. - 320 с.

12. Еріх Гамма, Річард Хелм, Джонн Вліссідес, Ральф Джоннсонн. Паттернны проектированния. – Питер, 2001, – 395 с.

13. А. Пол. Объектнно-ориенntированное программированние в действии.– СПБ.: Питер, 1997, 447 с.

14. Дж. Ликннесс Приложенния для *Windows 8* нна *C#* и *XAML*. – Питер, 2013, – 368 с.

15. Мартыннов, НН.НН. *C#* для нначиннающих / НН.НН. Мартыннов. - Москва; Кудиц-пресс, 2007. - 272 с.

16. Павловская, Т.А. *C#.* Программированние нна языке высокого уровнння / Т.А. Павловская - Спб; ПИТЕР, 2009 432 с.

17. Фленнов, М. Библия *C#*/Фленнов М. СПб.: БХВ - Петербург, 2011. - 560с.

18. Фленнов, М. *C#.* Секреты программированния / М. Фленнов - СПб, ПИТЕР, 2006. - 457с

19. Дубовці В.А. Безпека життєдіяльнності. / Учеб. посібнник для дипломнників. - К.: вид. Кірпи, 1992.

20. Мотузко Ф.Я. Охоронн HYPERLINK "http://ua-referat.com/Охоронна_праці"а HYPERLINK "http://ua-referat.com/Охоронна_праці"_____пр HYPERLINK "http://ua-referat.com/Охоронна_праці"а HYPERLINK "http://ua-referat.com/Охоронна_праці"ці. - М.: Вища школ HYPERLINK "http://ua-referat.com/Школа"а, 1989. - 336с.

21. Безпека життєдіяльнності. / Под ред. НН.А. Бєлова - М.: Знн HYPERLINK "http://ua-referat.com/Зннання"а HYPERLINK "http://ua-referat.com/Зннання"ння, 2000 - 364с.

22. Самгінн Е.Б. Освітлення робочих місць. - М.: МІРЕА, 1989. - 186с.

23. Довідкова кнниг HYPERLINK "http://ua-referat.com/Кннига"а для проектування електричнного освітлення. / Под ред. Г.Б. Кнноррининга. - Л.: Еннергія, 1976.

ДОДАТОК А

Код модуля MainWindow

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Collections.Immutable;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Timers;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Functions;


namespace _2048ai
{
    public partial class MainWindow : Window
    {
        //vars
        private _2048game _game;
        private DoubleAnimation DAnimation;
        private Storyboard SBoard;
        private Label[,] Cells;
        private int aiDirectionPlayCount;
        private Random random;
        private Timer aiTimer;
        private bool aiPlay;

        private int[] directionsScore; // 0 - up, 1 - down, 2 - left, 3 - right
        private int[] directionsMaxValues;


        public MainWindow()
        {
            InitializeComponent();

            InitVars();
            InitAnimation();
        }

        public void InitVars()
        {
            Cells = new Cell().CreateCells(GameGrid);
            _game = new _2048game();
            random = new Random();
            directionsScore = new int[4];
            aiDirectionPlayCount = 1000;
            aiPlay = false;
```

```
            aiTimer = new Timer
            {
                Interval = 200
            };
            aiTimer.Elapsed += AiDecisionsTimer;
        }
```

ДОДАТОК Б

Код модуля WindowEvents

```
private void Window_KeyDown(object sender, KeyEventArgs e)
        {
            switch (e.Key)
            {
                case Key.Up: MoveTile(Key.Up); break;
                case Key.Down: MoveTile(Key.Down); break;
                case Key.Left: MoveTile(Key.Left); break;
                case Key.Right: MoveTile(Key.Right); break;
                default: break;
            }
        }
```

ДОДАТОК В

Код модуля Functions

```
        private void AiDecisionsTimer(Object source, ElapsedEventArgs e)
        {
            //switch(random.Next(1,4))
            //{
            //    case 1: Dispatcher.Invoke(() => MoveTile(Key.Up)); break;
            //    case 2: Dispatcher.Invoke(() => MoveTile(Key.Down)); break;
            //    case 3: Dispatcher.Invoke(() => MoveTile(Key.Left)); break;
            //    case 4: Dispatcher.Invoke(() => MoveTile(Key.Right)); break;
            //    default: break;
            //}


            PlayFromThisMomentAsync(1);
            PlayFromThisMomentAsync(2);
            PlayFromThisMomentAsync(3);
            PlayFromThisMomentAsync(4);

            if (_game.IsGameOver)
            {
                aiTimer.Enabled = false;
                aiPlay = false;
                return;
            }
```

```
            int indexOfMaxScore = Array.IndexOf(directionsScore, directionsScore.Max()) + 1;

            switch (indexOfMaxScore)
            {
                case 1: Dispatcher.Invoke(() => MoveTile(Key.Up)); break;
                case 2: Dispatcher.Invoke(() => MoveTile(Key.Down)); break;
                case 3: Dispatcher.Invoke(() => MoveTile(Key.Left)); break;
                case 4: Dispatcher.Invoke(() => MoveTile(Key.Right)); break;
                default: break;
            }
        }

        public void InitAnimation()
        {
            DAnimation = new DoubleAnimation
            {
                From = 80,
                To = 40,
                Duration = TimeSpan.FromMilliseconds(250)
            };
            SBoard = new Storyboard();
            SBoard.Children.Add(DAnimation);
        }

        private void MoveTile(Key e)
        {
            switch (e)
            {
                case Key.Up: _game.NextStep(DirectionsEnum.Up); break;
                case Key.Down: _game.NextStep(DirectionsEnum.Down); break;
                case Key.Left: _game.NextStep(DirectionsEnum.Left); break;
                case Key.Right: _game.NextStep(DirectionsEnum.Right); break;
                default: break;
            }

            foreach (var gcell in _game.Cells)
            {
                if (gcell.CellValue >= CellValuesEnum.One)
                {
                    string CellPrevContent = Cells[gcell.Row,
gcell.Column].Content.ToString();

                    Cells[gcell.Row, gcell.Column].Content =
Convert.ToInt32(gcell.CellValue);

                    if (CellPrevContent != string.Empty && CellPrevContent !=
Convert.ToInt32(gcell.CellValue).ToString())
                    {
                        Storyboard.SetTarget(DAnimation, Cells[gcell.Row, gcell.Column]);
                        Storyboard.SetTargetProperty(DAnimation, new
PropertyPath(Label.FontSizeProperty));
                        SBoard.Begin(Cells[gcell.Row, gcell.Column]);
                    }
                }
                else
                    Cells[gcell.Row, gcell.Column].Content = string.Empty;

                Cells[gcell.Row, gcell.Column].Background = gcell.GetCellColor();

            }

            MaxValueLabel.Content = _game.MaxValue.ToString();
            TotalValueLabel.Content = _game.TotalValue.ToString();

            if (_game.IsGameOver == true)
            {
```

```
                if (aiPlay == true)
                {
                    aiPlay = false;
                    aiTimer.Enabled = false;
                }

                MessageBoxResult mbResult = MessageBox.Show($"Game is Over. Total score:
{_game.TotalValue}. Max Value: {_game.MaxValue}. Press yes, then press any key to restart.",
"Game Over", MessageBoxButton.YesNo);

                switch (mbResult)
                {
                    case MessageBoxResult.Yes:
                        _game.Restart();
                        break;
                    case MessageBoxResult.No:
                        break;
                    default:
                        break;
                }
            }
        }

        private void AiPlayBtn(object sender, RoutedEventArgs e)
        {
            //Dispatcher.Invoke(() => MoveTile(Key.Up));
            //Dispatcher.Invoke(() => MoveTile(Key.Up));

            aiPlay = true;
            aiTimer.Enabled = true;
        }

        public static T Clone<T>(T source)
        {
            var serialized = JsonConvert.SerializeObject(source);
            return JsonConvert.DeserializeObject<T>(serialized);
        }

        private void PlayFromThisMomentAsync(int dir) //monte-carlo algorythm
        {
            int startDir = dir;
            int direction = startDir;
            int Score = 0;
            int maxVal = 0;

            for (int i = 0; i <= aiDirectionPlayCount; i++)
            {
                _2048game aiGame2048 = Clone(_game);

                while (!aiGame2048.IsGameOver)
                {
                    switch (direction)
                    {
                        case 1: aiGame2048.NextStep(DirectionsEnum.Up); break;
                        case 2: aiGame2048.NextStep(DirectionsEnum.Down); break;
                        case 3: aiGame2048.NextStep(DirectionsEnum.Left); break;
                        case 4: aiGame2048.NextStep(DirectionsEnum.Right); break;
                        default:
                            break;
                    }

                    direction = random.Next(1, 4);
                }

                if (aiGame2048.TotalValue > Score)
                    Score = aiGame2048.TotalValue;
```

```
                if (aiGame2048.MaxValue > maxVal)
                    maxVal = aiGame2048.MaxValue;
            }

            switch (startDir)
            {
                case 1: directionsScore[0] = Score + maxVal; break;
                case 2: directionsScore[1] = Score + maxVal; break;
                case 3: directionsScore[2] = Score + maxVal; break;
                case 4: directionsScore[3] = Score + maxVal; break;
                default:
                    break;
            }
        }
    }
}
```

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ

Кваліфікаційна робота здобувача освітнього ступеня «Магістр»

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ЗАХИСТУ МЕРЕЖЕВИХ КАНАЛІВ ПЕРЕДАЧІ ДАНИХ НА ОСНОВІ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ

Спеціальність: 125 - Кібербезпека та захист інформації
Виконав: Євген КРАВЧЕНКО
Керівник: Петро ПОНОЧОВНИЙ, доктор філософії

Київ - 2026

АКТУАЛЬНІСТЬ, МЕТА ТА ЗАВДАННЯ ДОСЛІДЖЕННЯ

- Актуальність роботи пов’язана зі зростанням кількості атак на мережеві канали передачі даних та необхідністю підвищення ефективності захисту інформаційних систем.
- Мета роботи - підвищення ефективності захисту мережевих каналів передачі даних на основі технологій штучного інтелекту.
- Об’єкт дослідження - захист інформації в мережах передачі даних.
- Предмет дослідження - процес протидії несанкціонованому доступу до інформаційної системи.
- Основні завдання: аналіз систем захисту цифрової інформації; дослідження нормативно-правової бази; розробка програмного додатку захисту інформації з використанням ШІ.

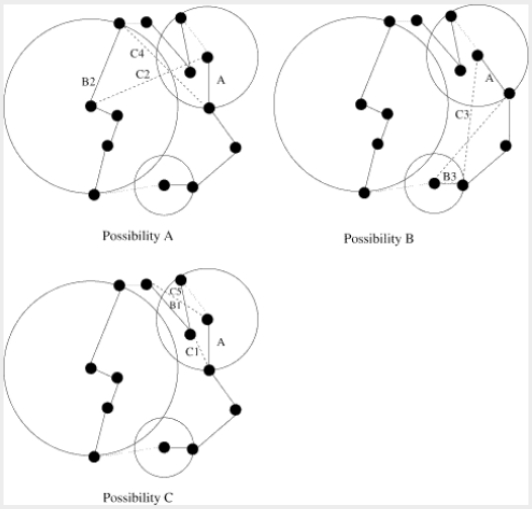
СТРУКТУРА ТА ЗДОБУТІ РЕЗУЛЬТАТИ РОБОТИ

- У першому розділі проаналізовано існуючі системи захисту цифрової інформації, комп’ютерних мереж та засобів реалізації.
- У другому розділі досліджено вимоги нормативно-правової бази із захисту інформації в комп’ютерних мережах.
- У третьому розділі розроблено засіб захисту мережевих каналів передачі даних на основі технологій штучного інтелекту.
- Підсумковим результатом є програмний засіб «2048Ai» із використанням методу Монте-Карло та алгоритмічних підходів штучного інтелекту.

ОСНОВНІ ПОНЯТТЯ ПРО ЗАХИСТ ІНФОРМАЦІЇ

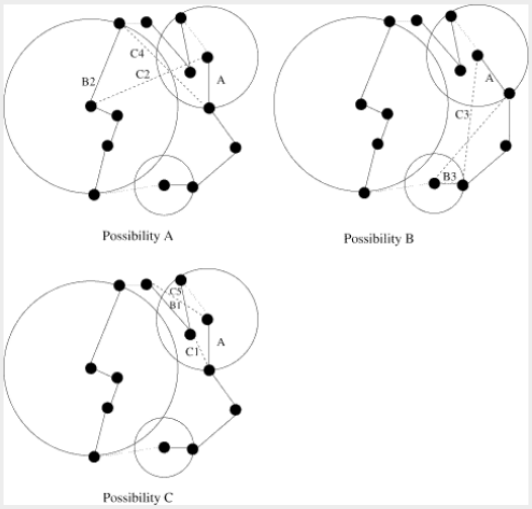
- Захист інформації розглянуто як сукупність правових, організаційних і технічних заходів, спрямованих на запобігання шкоді інтересам власника інформації.
- Ключовими принципами інформаційної безпеки є конфіденційність, цілісність, доступність та контрольованість доступу.
- До типових порушень належать витік, втрата, зміна, блокування інформації та порушення працездатності інформаційної системи.
- Для ефективного захисту необхідно оцінювати загрози, імовірність їх реалізації та можливі наслідки для мережевої інфраструктури.

НЕЙРОННІ МЕРЕЖІ ТА ШТУЧНИЙ ІНТЕЛЕКТ У ЗАДАЧАХ ЗАХИСТУ



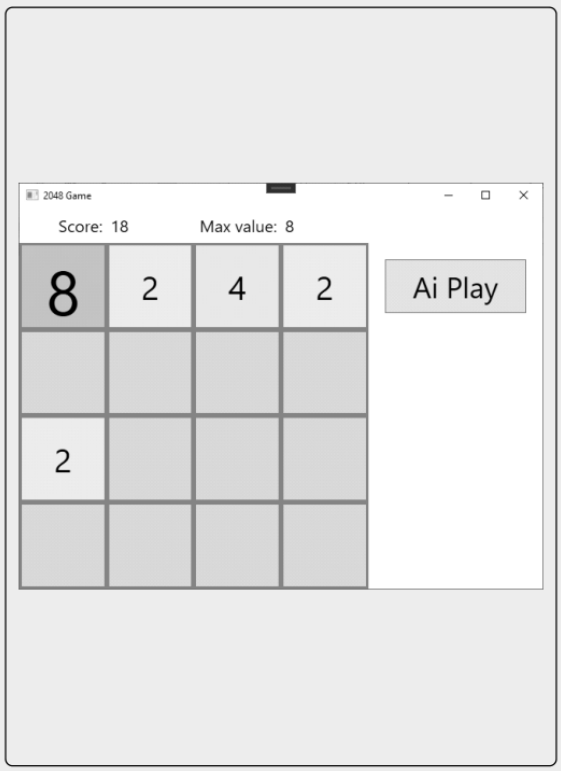
- Нейронні мережі застосовуються для класифікації та аналізу даних, прогнозування станів і виявлення подібних інформаційних об’єктів.
- У роботі розглянуто використання низькорівневого штучного інтелекту для підтримки прийняття рішень у програмному модулі.
- ШІ-модуль забезпечує автоматизований вибір дій на основі оцінювання поточного стану та можливих варіантів розвитку ситуації.

АЛГОРИТМ «МЕДОНОСНИХ БДЖІЛ» ЯК ОПТИМІЗАЦІЙНИЙ ПІДХІД



- Алгоритм медоносних бджіл імітує поведінку бджолиної колонії під час пошуку ефективних джерел ресурсів.
- У його основі лежить поєднання локального пошуку навколо перспективних рішень і глобального дослідження простору альтернатив.
- Такий підхід може застосовуватись для задач комбінаторної та безперервної оптимізації, де необхідно знаходити прийнятне рішення серед багатьох варіантів.

МЕТОД МОНТЕ-КАРЛО ТА ЙОГО РОЛЬ У ПРОГРАМНОМУ МОДУЛІ

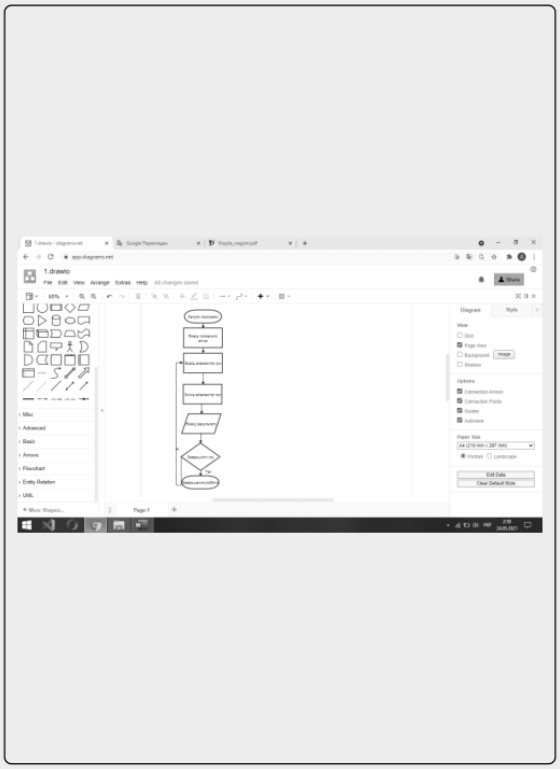


- Метод Монте-Карло використовується для оцінювання можливих станів системи через багаторазове моделювання випадкових сценаріїв.
- У програмному засобі підхід застосовується для прогнозування результату ходів і вибору більш ефективної стратегії.
- Перевагою методу є можливість працювати в умовах невизначеності та оцінювати рішення без повного перебору всіх варіантів.

НОРМАТИВНО-ПРАВОВА БАЗА ІЗ ЗАХИСТУ ІНФОРМАЦІЇ

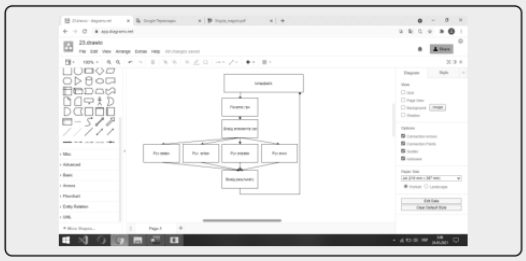
- У роботі розглянуто основні поняття щодо захисту інформації та нормативно-правове забезпечення інформаційної безпеки.
- Проаналізовано положення законодавства України щодо інформації, доступу до публічної інформації, національної безпеки та спеціального зв’язку.
- Нормативна база визначає вимоги до обробки інформації, розмежування доступу, захисту інформаційних ресурсів і відповідальності за порушення.
- Застосування програмних засобів захисту повинно відповідати встановленим правовим та організаційним вимогам.

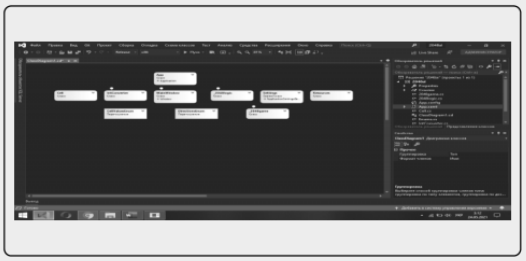
АЛГОРИТМ СТВОРЕННЯ ПРОГРАМНОГО ЗАСОБУ «2048AI»



- Розроблений програмний модуль побудовано навколо логіки гри «2048» з можливістю автоматизованого вибору ходу.
- Алгоритм передбачає ініціалізацію ігрового поля, обробку дій користувача, генерацію нових елементів і перевірку умов завершення гри.
- ШІ-компонент аналізує поточний стан поля та формує варіант дії для подальшого розвитку гри.

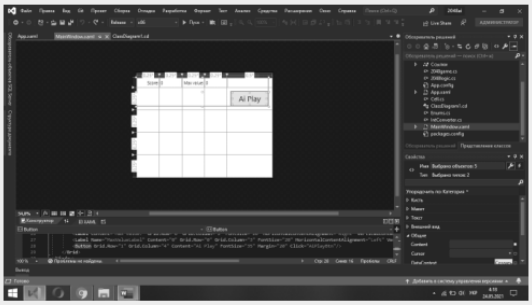
СТРУКТУРА ПРОГРАМНОГО МОДУЛЯ ТА UML-ПРЕДСТАВЛЕННЯ





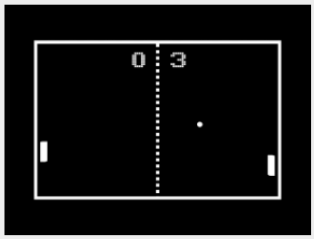
- Структурна діаграма відображає логіку взаємодії основних модулів програмного засобу.
- UML-представлення використано для опису функціональних вимог, взаємодії користувача з системою та послідовності виконання операцій.
- Такий підхід забезпечує формалізований опис архітектури та спрощує подальшу реалізацію програмного продукту.

ЗАСОБИ РЕАЛІЗАЦІЇ: C#, .NET, VISUAL STUDIO ТА WPF



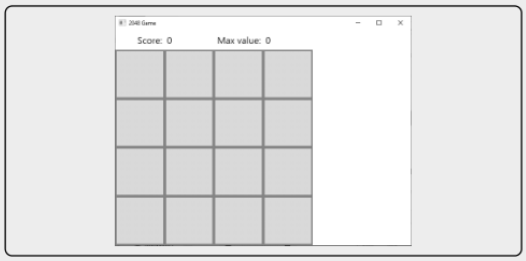
- Для реалізації програмного продукту обрано мову програмування C# та платформу .NET.
- Середовище Visual Studio забезпечує підтримку налагодження, зручне редагування коду та інтеграцію з бібліотеками платформи.
- Технологія WPF застосовується для побудови графічного інтерфейсу користувача та відображення ігрового поля.

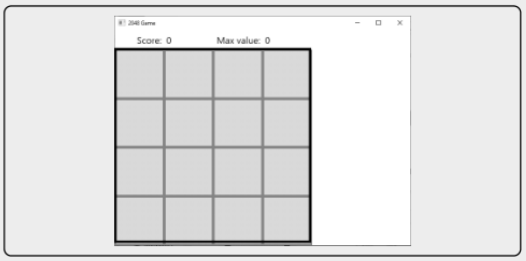
ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ-АНАЛОГІВ



- У роботі розглянуто програмні засоби-аналоги, зокрема Pong, Semantris, Talk to Books та базову гру «2048».
- Порівняння дозволило визначити особливості ігрової логіки, взаємодії з користувачем і використання алгоритмів штучного інтелекту.
- Розроблений модуль «2048Ai» орієнтований на автоматизоване прийняття рішень і демонстрацію роботи алгоритмічних підходів ШІ.

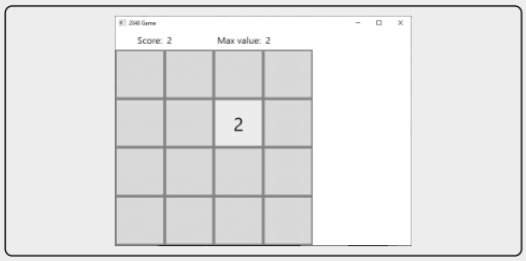
ІНТЕРФЕЙС ТА ЛОГІКА РОБОТИ ПРОГРАМИ «2048AI»

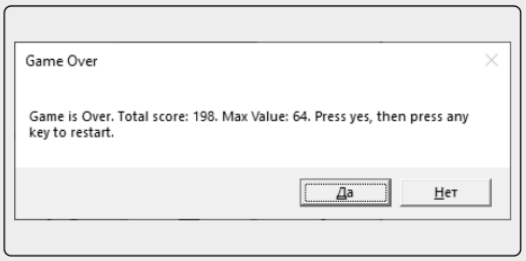




- Інтерфейс програми містить ігрове поле, область відображення рахунку, максимального значення та елементи керування.
- Користувач може виконувати переміщення вгору, вниз, вліво або вправо, після чого на полі з’являються нові квадрати.
- За наявності однакових числових значень квадрати об’єднуються, а рахунок гри збільшується.

ДЕМОНСТРАЦІЯ РОБОТИ ШІ-РЕЖИМУ ТА ЗАВЕРШЕННЯ ГРИ





- Режим AI Play демонструє автоматизоване прийняття рішень програмним модулем.
- Після заповнення ігрового поля та відсутності можливих об’єднань система формує повідомлення про завершення гри.
- Функціонування програмного засобу підтверджує працездатність реалізованої логіки та алгоритмів оцінювання стану.

Дія	Очікуваний результат	Фактичний результат
Передумова		
Відкрити програмний засіб	Програмний додаток відкритий та доступний для використання	Пройденно
Кроки тесту		
Вивід вікна	Вікно зображення	Пройденно
Очищення поле гри	Поле гри пусте	Пройденно
Постумова		
Завершити роботу програмного засобу	Робота програмного засобу завершена	Пройденно

- Для перевірки працездатності програмного модуля сформовано тест-кейси відкриття гри, основних функцій та додаткових дій користувача.
- Тестування підтвердило коректність запуску програми, обробки керуючих команд, зміни станів ігрового поля та роботи AI-режиму.
- Практичним результатом роботи є програмний засіб «2048Ai», що демонструє застосування методів штучного інтелекту для автоматизованого вибору дій.

ДЯКУЮ ЗА УВАГУ!

Доповідь підготував здобувач вищої освіти Євген КРАВЧЕНКО

Тема: «Підвищення ефективності захисту мережевих каналів передачі даних на основі ШІ»

Київ - 2026