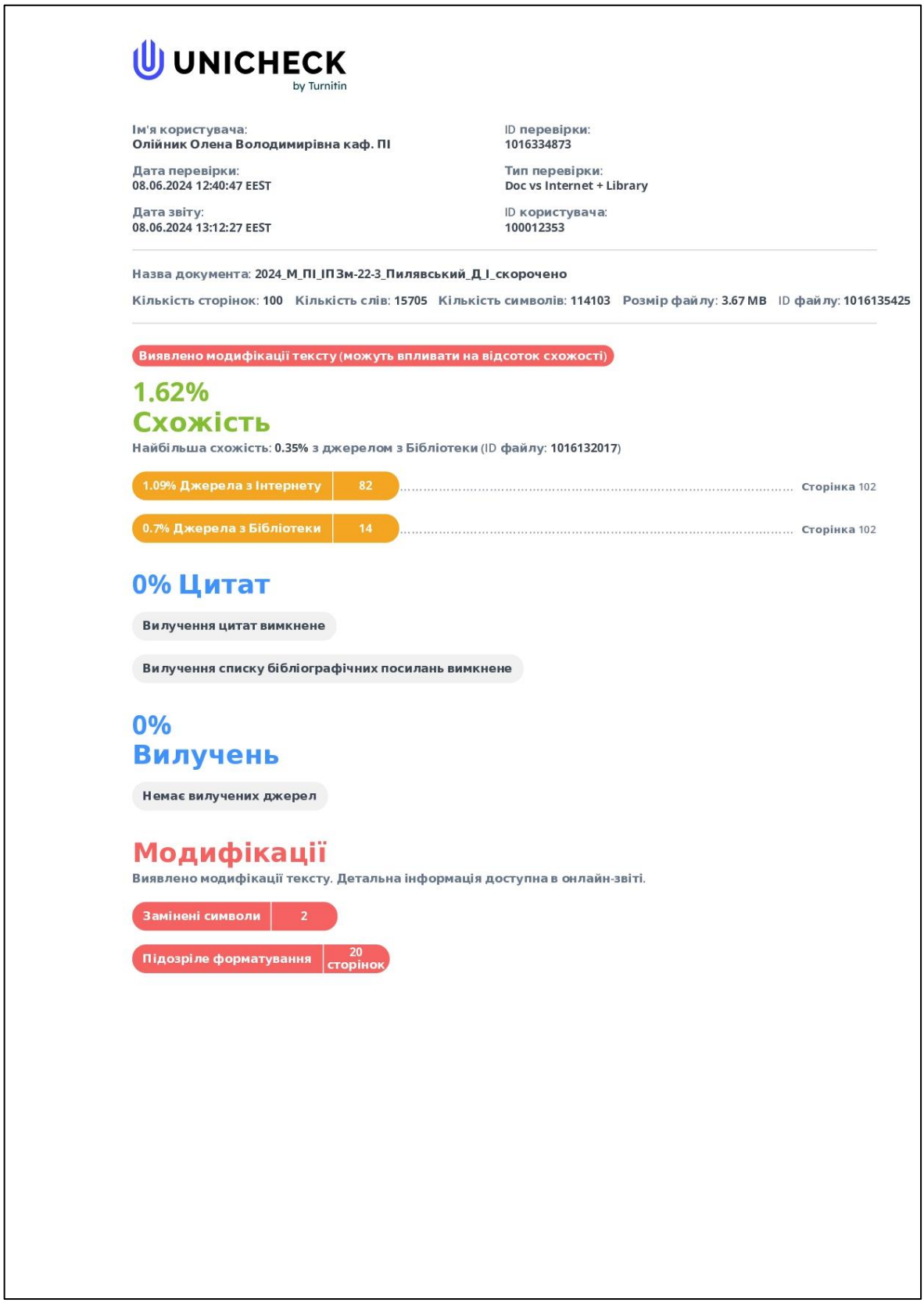
ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016334873

Дата перевірки:
08.06.2024 12:40:47 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
08.06.2024 13:12:27 EEST

ID користувача:
100012353

Назва документа: 2024_М_ПІ_ІПЗм-22-3_Пилявський_Д_І_скорочено

Кількість сторінок: 100 Кількість слів: 15705 Кількість символів: 114103 Розмір файлу: 3.67 MB ID файлу: 1016135425

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

1.62%
Схожість

Найбільша схожість: 0.35% з джерелом з Бібліотеки (ID файлу: 1016132017)

1.09% Джерела з Інтернету 82 ..... Сторінка 102

0.7% Джерела з Бібліотеки 14 ..... Сторінка 102

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

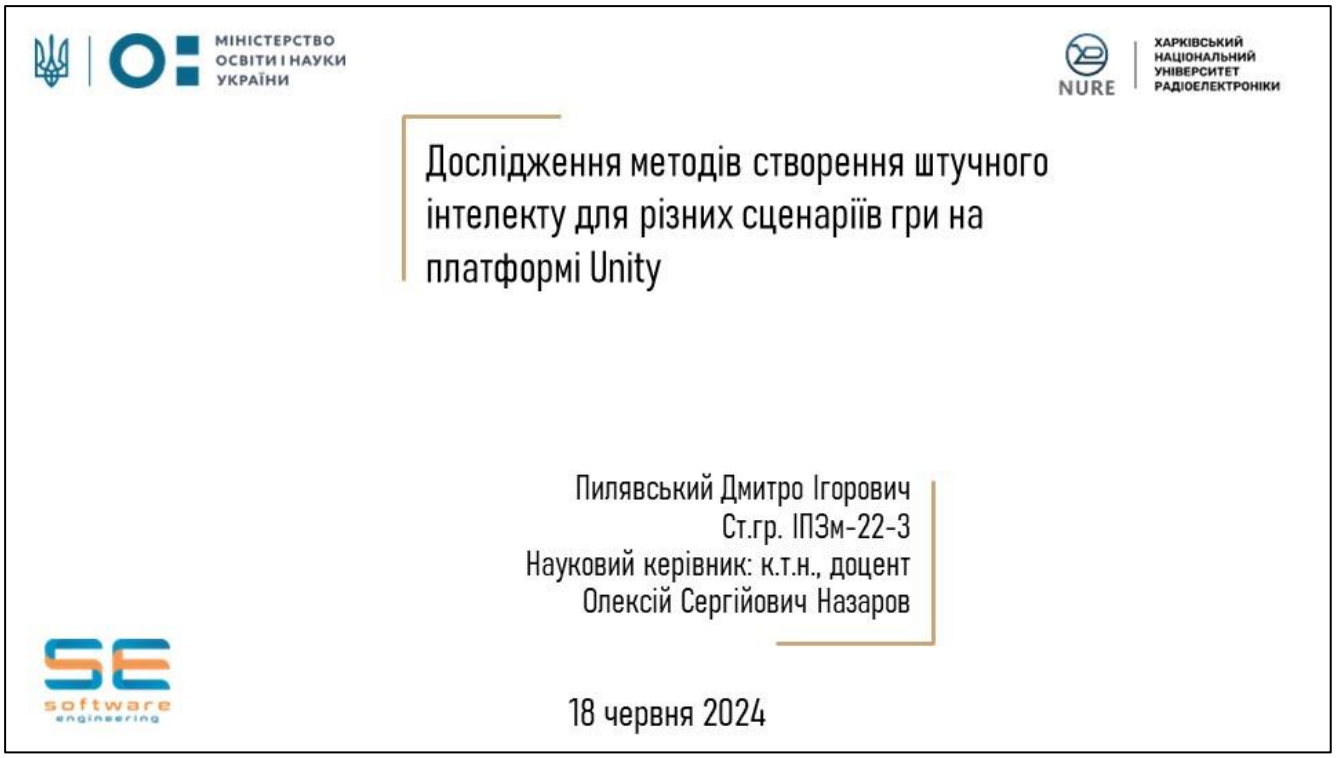
Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

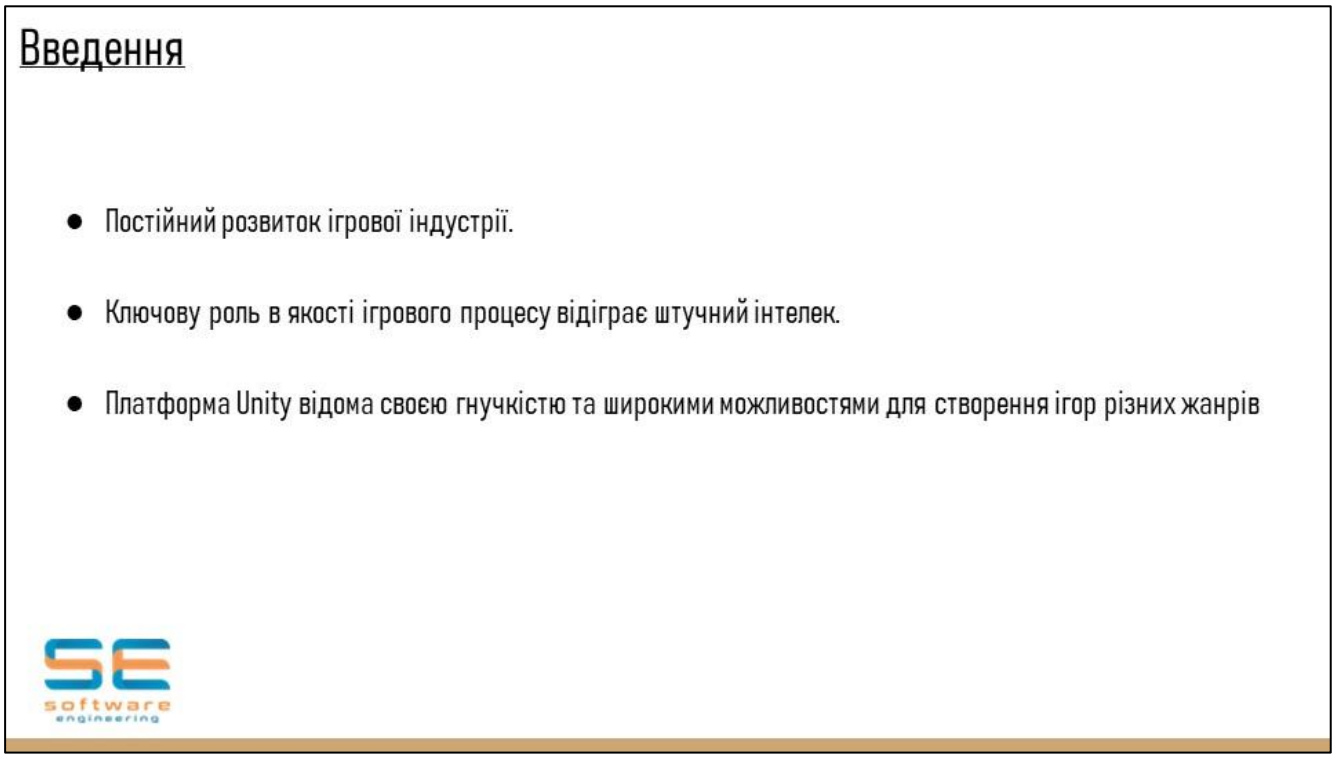
Замінені символи 2

Підозріле форматування 20 сторінок

ДОДАТОК Б

Слайди презентації





Дослідження

- Об'єктом дослідження є методи створення штучного інтелекту.
- Предметом дослідження є ефективність алгоритмів пошуку шляху та алгоритмів прийняття рішень, їх реалізація, простота та доречність.
- Метою дослідження є аналіз і порівняння алгоритмів створення ігрового ШІ для різних сценаріїв гри.
- Дослідження включає порівняння ефективності алгоритмів пошуку шляху та прийняття рішень у контексті ігрових ситуацій.

Проблематика

- Часова складність алгоритмів
- Використання пам'яті
- Зрозумілість та структура алгоритмів
- Баланс витрат і вигод

Задачі

- Аналіз поточного стану ігрового ШІ та визначення основних викликів.
- Вивчення можливостей та обмежень платформи Unity.
- Визначення різних типів ігрових сценаріїв та їхніх особливостей.
- Порівняння алгоритмів пошуку шляху та прийняття рішень за визначеними критеріями.
- Проведення експериментальних досліджень та аналіз отриманих результатів.
- Формулювання висновків і рекомендацій для розробників ігрового ШІ

Інструменти розробки

Unity

VISUAL STUDIO 2022

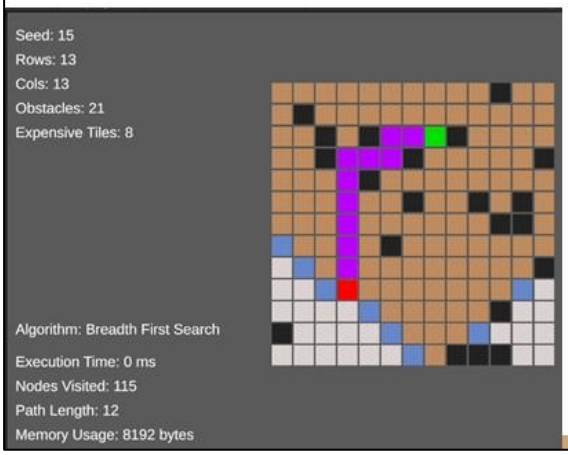
Методологія

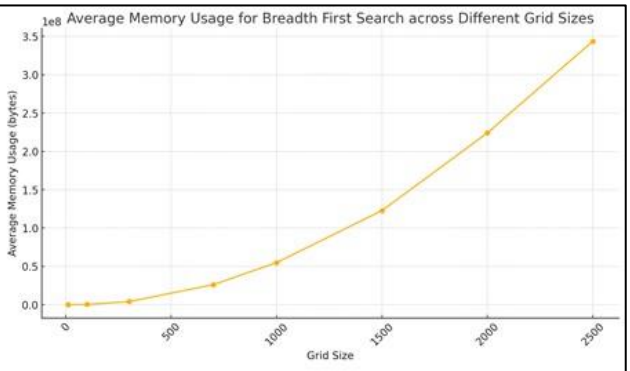
- Аналіз літератури
- Розробка експериментальної бази
- Проведення експериментів
- Збір та аналіз даних
- Порівняння результатів
- Формулювання висновків

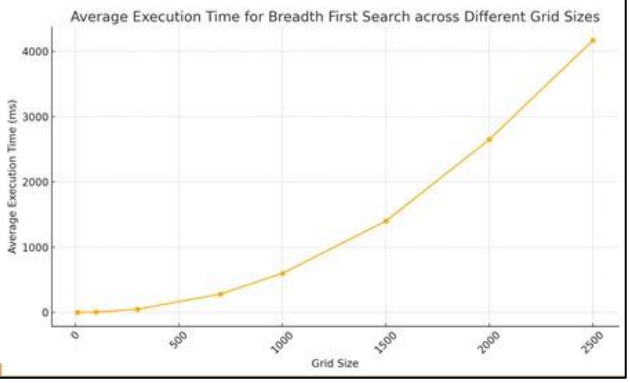
Алгоритми пошуку шляху

Breadth First Search

- шукає найкоротший шлях
- досліджує граф рівень за рівнем



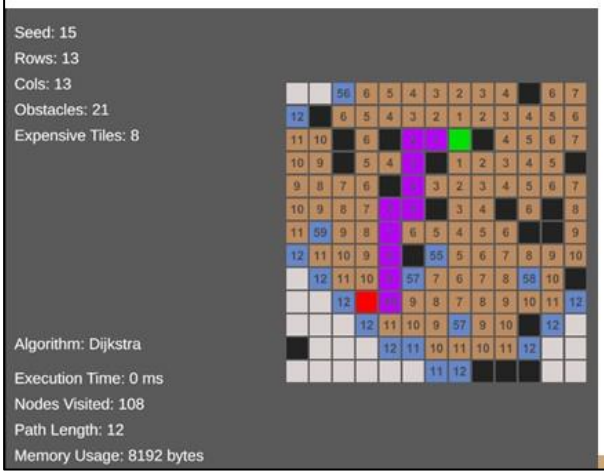


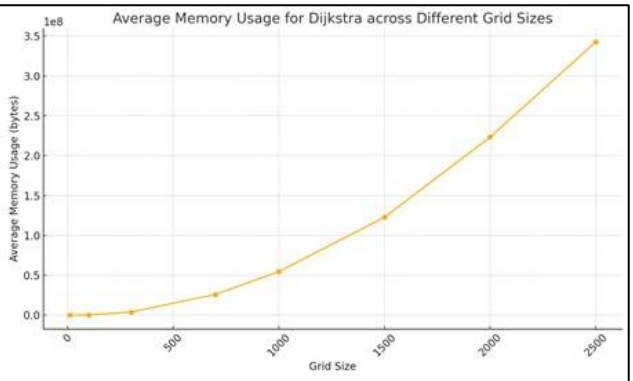


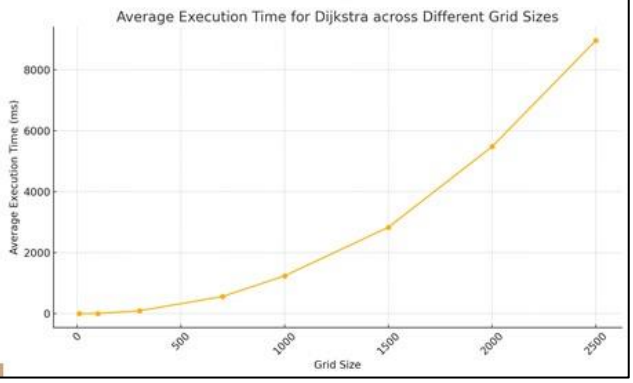
Алгоритми пошуку шляху

Dijkstra

- шукає найкоротший шлях
- використовує пріоритетну чергу для поступового розширення



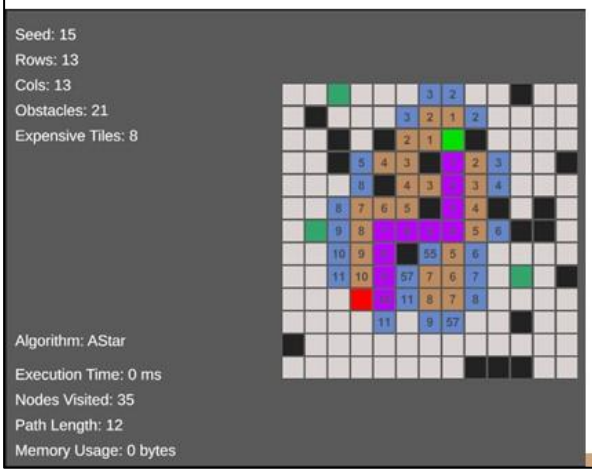


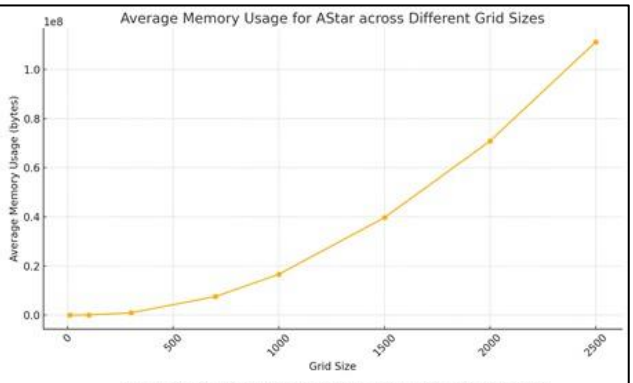


Алгоритми пошуку шляху

A-Star

- шукає найкоротший шлях
- комбінує пошук за найменшою вартістю та жадібний пошук, використовуючи евристичну функцію для оцінки вартості шляху

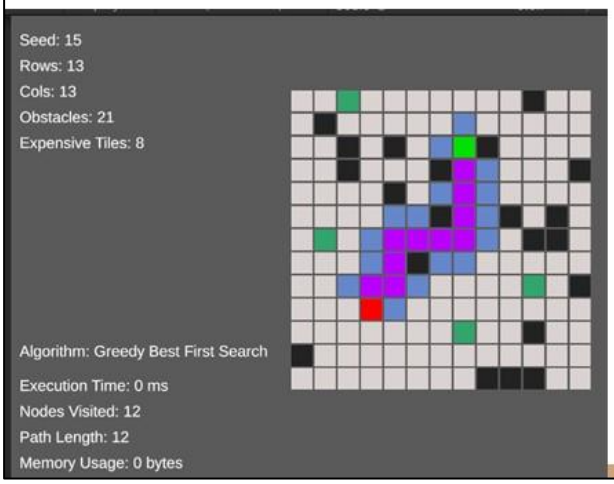


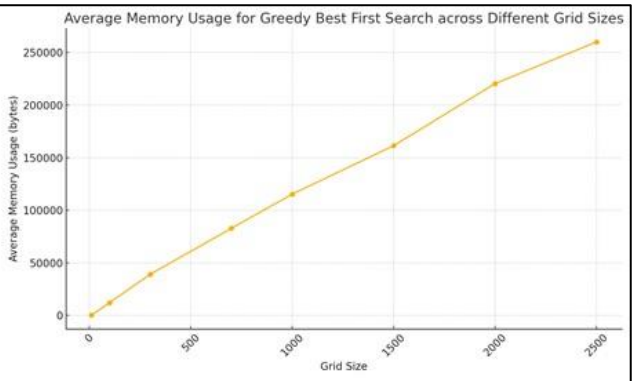


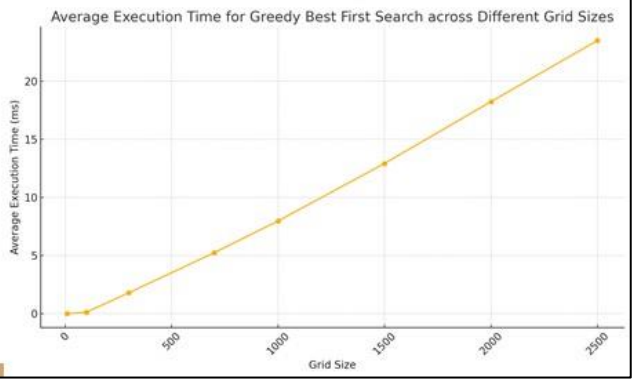
Алгоритми пошуку шляху

Greedy Best First Search

- швидкий, але не гарантує знаходження найкоротшого шляху
- обирає вершини на основі евристичної функції, яка оцінює вартість до цільової вершини, і розширює найперспективніші вузли,



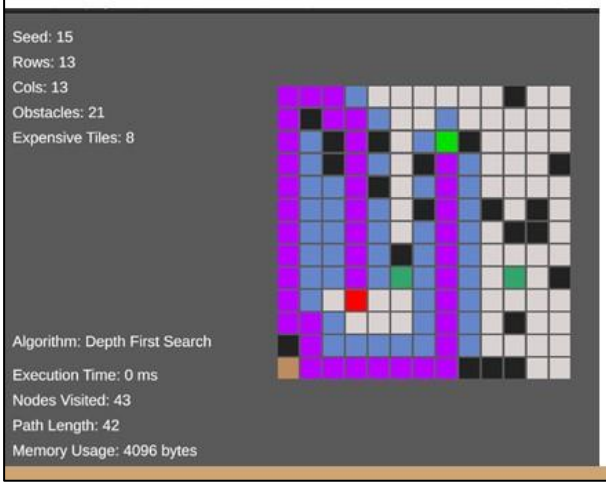


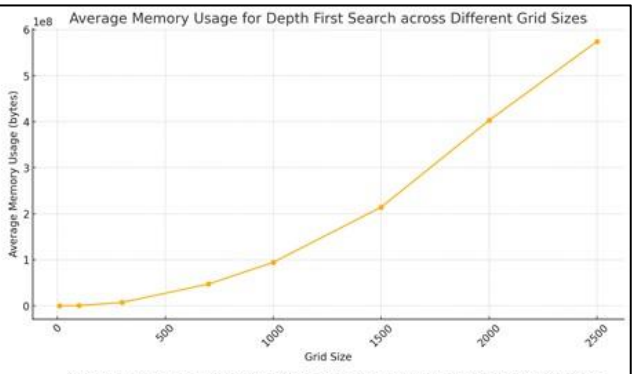


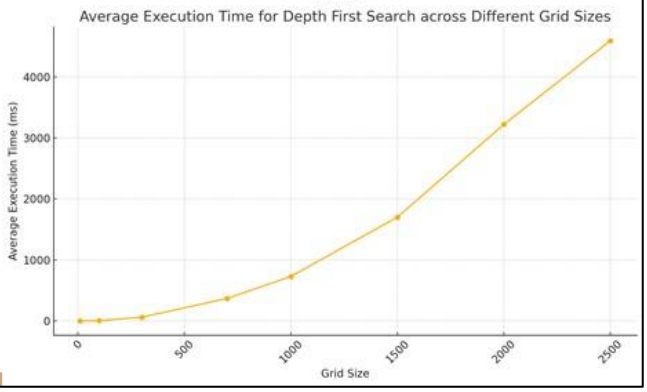
Алгоритми пошуку шляху

Depth First Search

- підходить для задач, де важливо досліджувати всі можливі шляхи, наприклад, у лабіринтах, але не ефективний для знаходження найкоротших шляхів
- працює за принципом глибини



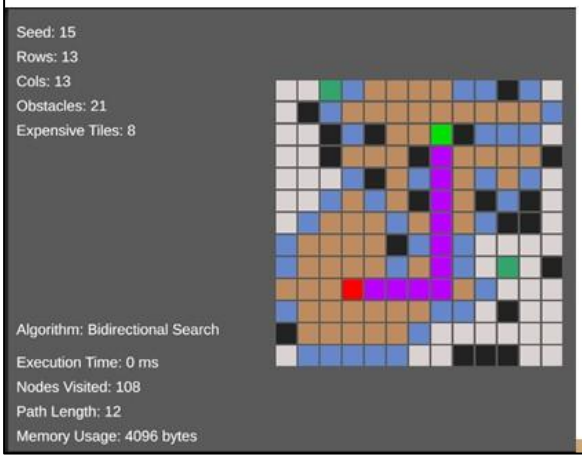


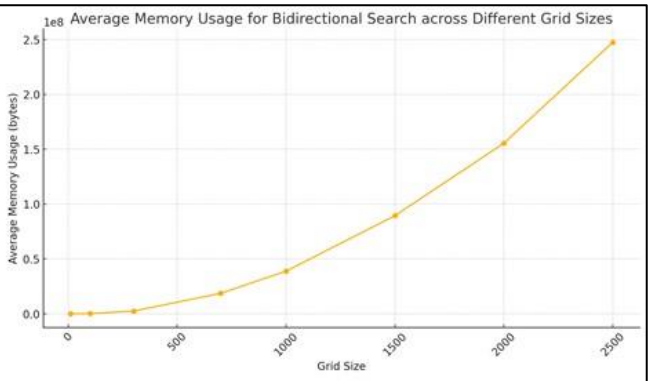


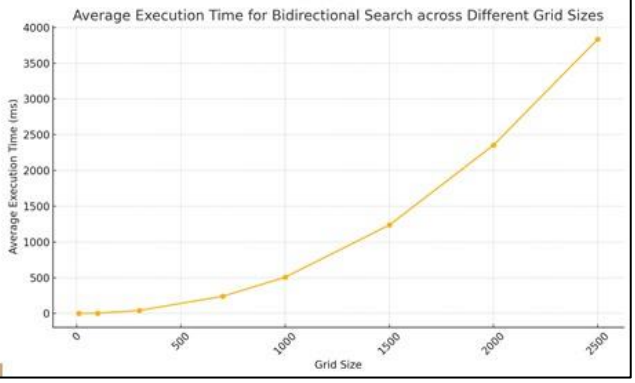
Алгоритми пошуку шляху

Bidirectional Search

- гарантує знаходження найкоротшого шляху
- працює одночасно з двох напрямків: від початкової вершини та від цільової вершини, доки обидва фронти пошуку не зустрінуться



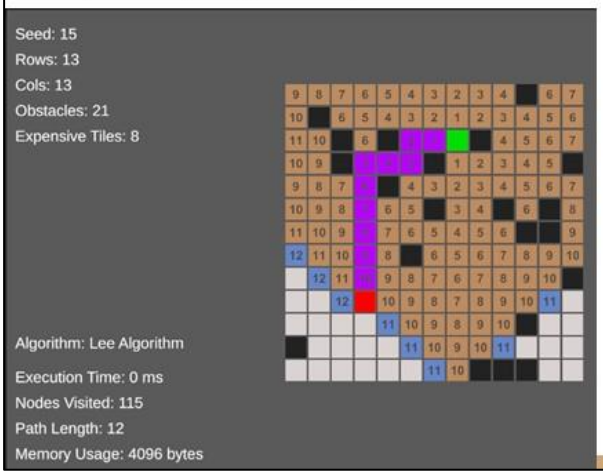


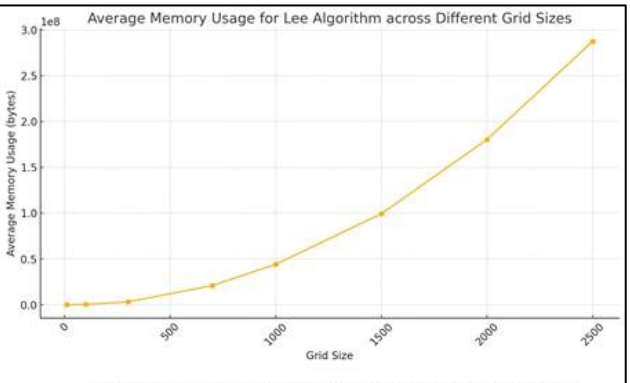


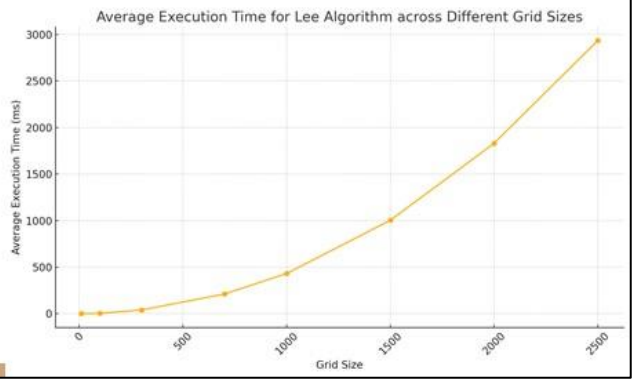
Алгоритми пошуку шляху

Lee Algorithm

- гарантує знаходження найкоротшого шляху
- розширюючи фронт пошуку рівномірно у всі сторони



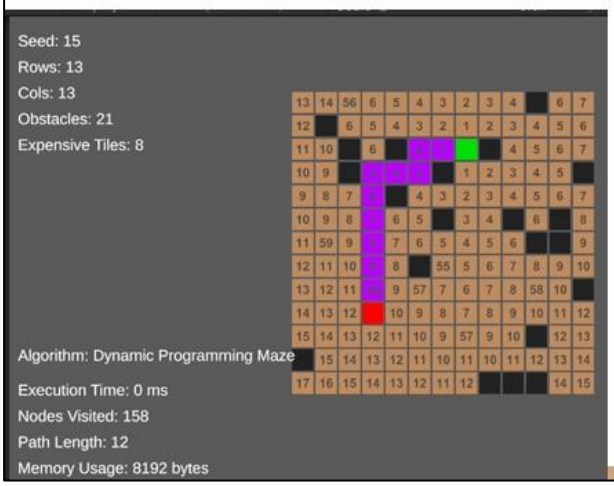


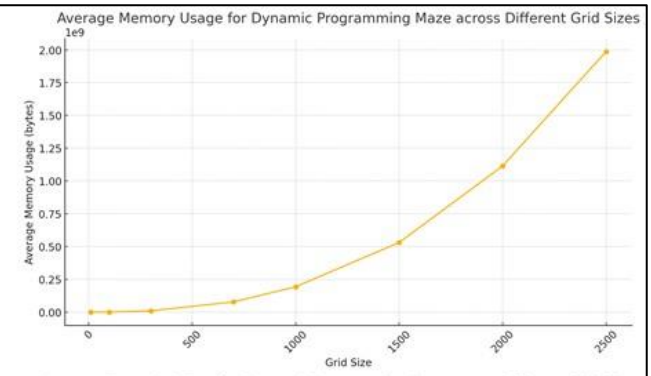


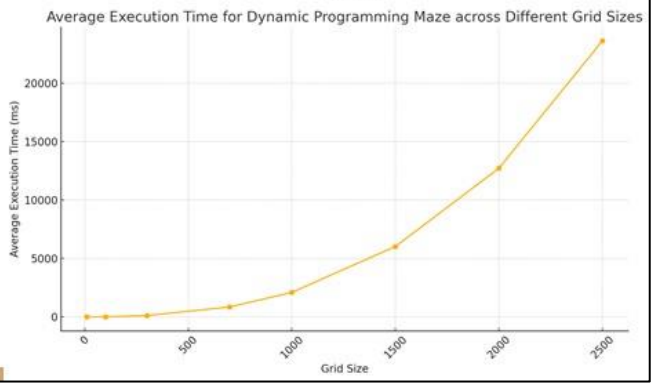
Алгоритми пошуку шляху

Dynamic Programming Maze

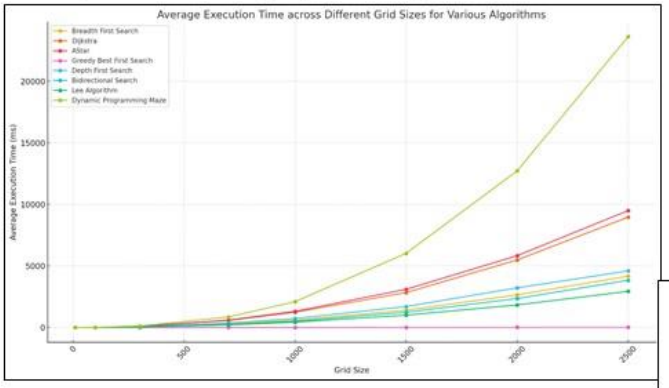
- забезпечує найкоротший шлях до цільової вершини
- використовує метод динамічного програмування для оновлення вартості шляхів

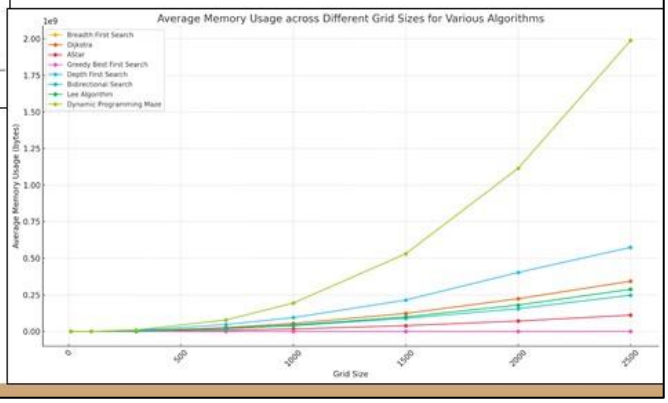






Алгоритми пошуку шляху

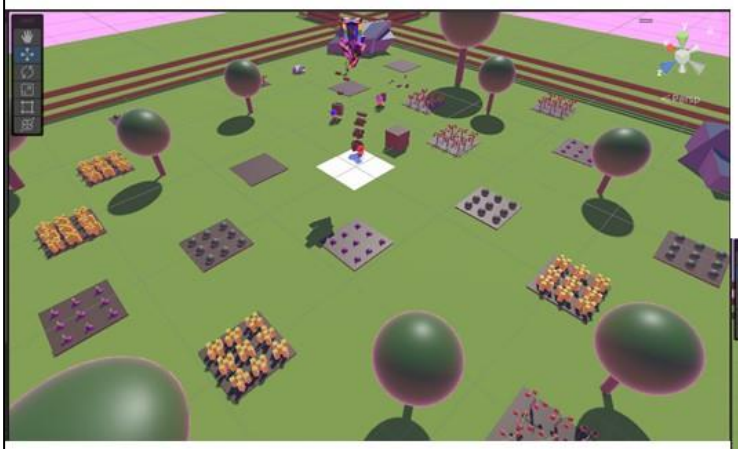




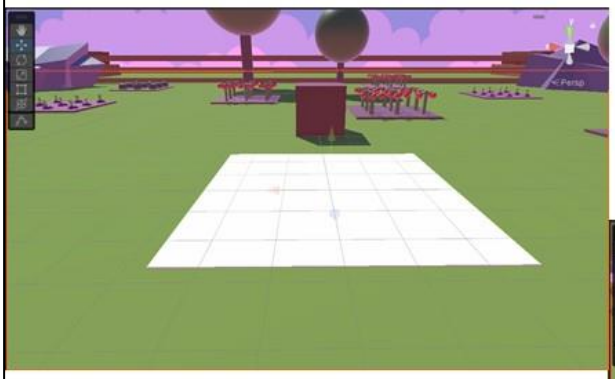
SE software engineering

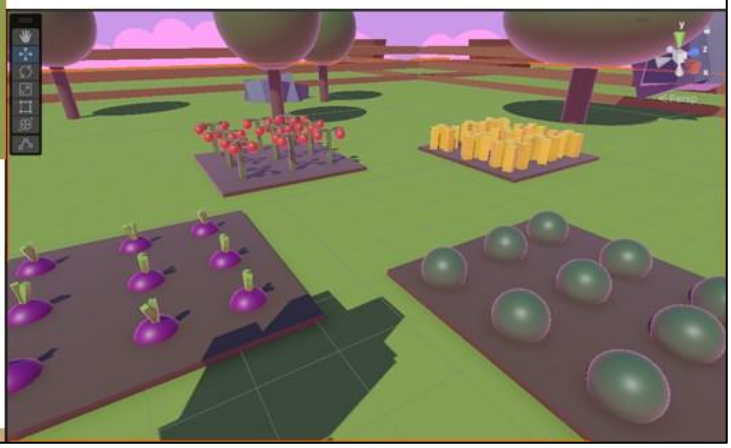
Алгоритми прийняття рішень

Середа тестування



Алгоритми прийняття рішень

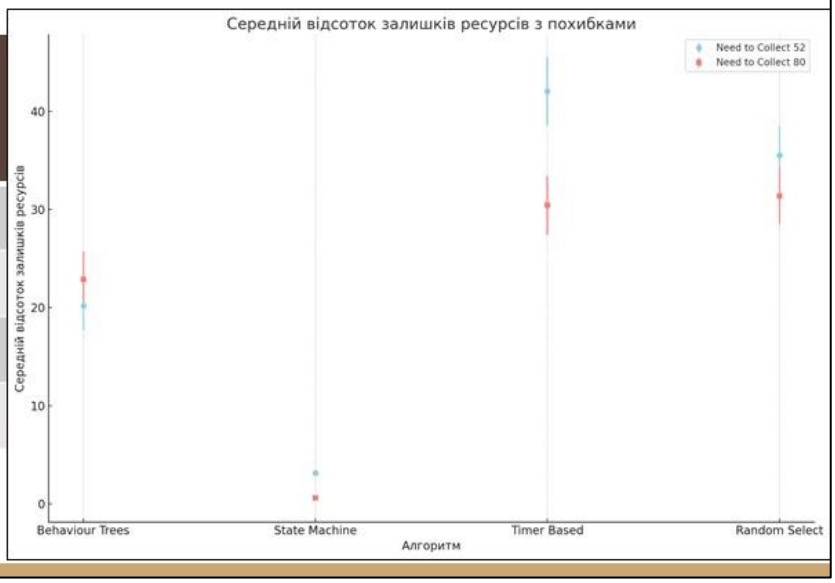




Алгоритми прийняття рішень

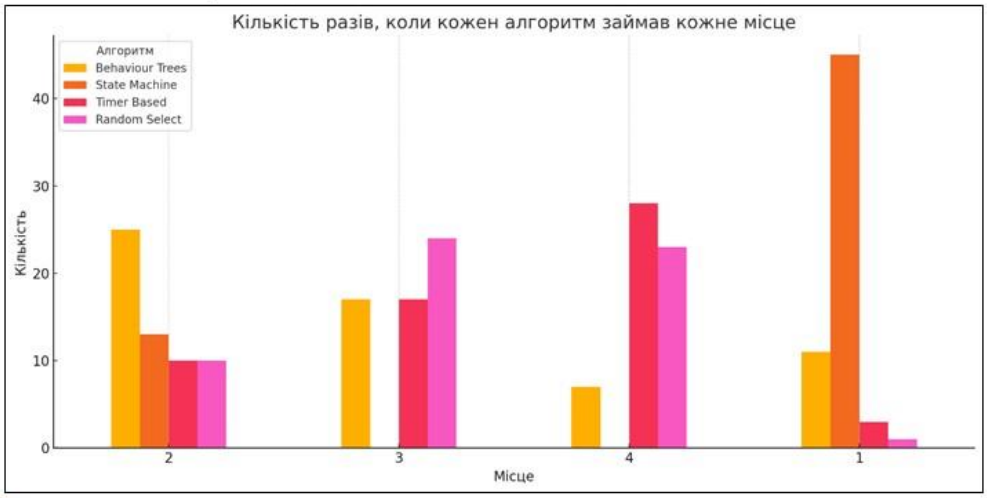
Залежність результатів від кількості ресурсів, що потрібно зібрати

Algorithm	Mean Remains to be Assembled (52), %	Mean Remains to be Assembled (80), %
Behaviour Trees	20.192	22.884
State Machine	3.159	0.625
Timer Based	42.032	30.432
Random Select	35.508	31.394

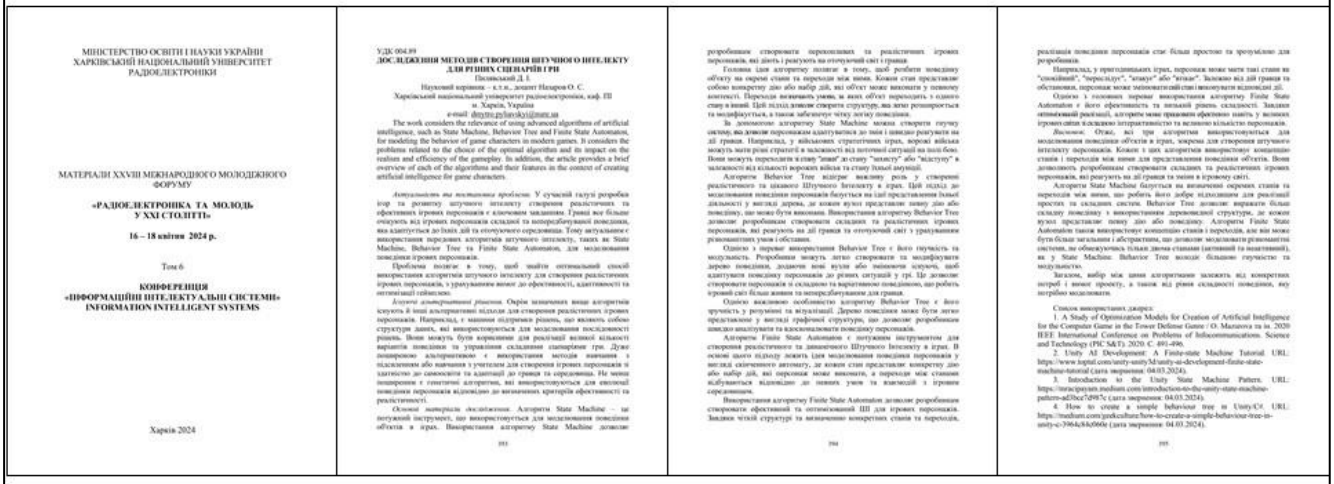


Алгоритми прийняття рішень

Результати – зайняті місця



Публікація результатів



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ XXVIII МЕЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ»

16 – 18 квітня 2024 р.

Том 6

КОНФЕРЕНЦІЯ «ІНФОРМАЦІЙНІ ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ» INFORMATION INTELLIGENT SYSTEMS

Харків 2024

УДК 004.89

ДОСЛІДЖЕННЯ МЕТОДІВ СТВОРЕННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ РУШІЙ СЦЕНАРІЇВ І ГРИ

Пилявський Д. І.
Науковий керівник – к.т.н., доцент Назаров О. С.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: dmytro.pylivskyi@nure.ua

The work considers the relevance of using advanced algorithms of artificial intelligence, such as State Machine, Behavior Tree and Finite State Automaton, for modeling the behavior of game characters in modern games. It considers the problems related to the choice of the optimal algorithm and its impact on the realism and efficiency of the gameplay. In addition, the article provides a brief overview of each of the algorithms and their features in the context of creating artificial intelligence for game characters.

Актуальність та постановка проблеми. У сучасній галузі розробки ігор та розвитку штучного інтелекту створення реалістичних та ефективних ігрових персонажів є ключовим завданням. Гравці все більше очікують від ігрових персонажів складної та непередбачуваної поведінки, яка адаптується до їхніх дій та оточуючого середовища. Тому актуальними є використання передових алгоритмів штучного інтелекту, таких як State Machine, Behavior Tree та Finite State Automaton, для моделювання поведінки ігрових персонажів.

Проблема полягає в тому, щоб знайти оптимальний спосіб використання алгоритмів штучного інтелекту для створення реалістичних ігрових персонажів, з урахуванням впливу до ефективності, адаптивності та оптимізації геймплею.

Аналіз останніх досліджень і публікацій. Окрім зазначених вище алгоритмів існують й інші альтернативні підходи для створення реалістичних ігрових персонажів. Наприклад, є машини підтримки рішень, що являють собою структури даних, які використовуються для моделювання послідовності рішень. Вони можуть бути корисними для реалізації великої кількості варіантів поведінки та управління складними сценаріями гри. Дуже поширеною альтернативою є використання методів навчання з підкріпленням або навчання з учителем для створення ігрових персонажів зі здатністю до самонавчання та адаптації до гравця та середовища. Не менш поширеними є генетичні алгоритми, які використовуються для еволюції поведінки персонажів відповідно до визначених критеріїв ефективності та реалістичності.

Основні матеріали дослідження. Алгоритм State Machine – це потужний інструмент, що використовується для моделювання поведінки об'єктів в ігрях. Використання алгоритму State Machine дозволяє розробникам створювати переконливих та реалістичних ігрових персонажів, які діють і реагують на оточуючий світ і гравця.

Головна ідея алгоритму полягає в тому, щоб розбити поведінку об'єкту на окремі стани та переходи між ними. Кожен стан представляє собою конкретну дію або набір дій, які об'єкт може виконати у певному контексті. Переходи визначають умови, за яких об'єкт переходить з одного стану в інший. Цей підхід дозволяє створити структуру, яка легко розширюється та модифікується, а також забезпечує чітку логіку поведінки.

За допомогою алгоритму State Machine можна створити гнучку систему, що дозволяє персонажам адаптуватися до змін і швидко реагувати на дії гравця. Наприклад, у військових стратегічних іграх, ворожі війська можуть мати різні стратегії в залежності від поточної ситуації на полі бою. Вони можуть переходити в стан "атака" де стану "захисту" або "відступу" в залежності від кількості ворожих військ та стану їхньої амуніції.

Алгоритм Behavior Tree відіграє важливу роль у створенні реалістичного та цікавого Штучного Інтелекту в ігрях. Цей підхід до моделювання поведінки персонажів базується на ідеї представлення їхньої діяльності у вигляді дерева, де кожен вузол представляє певну дію або поведінку, що може бути виконана. Використання алгоритму Behavior Tree дозволяє розробникам створювати складних та реалістичних ігрових персонажів, які реагують на дії гравця та оточуючий світ з урахуванням різноманітних умов і обставин.

Однією з перевag використання Behavior Tree є його гнучкість та модульність. Розробники можуть легко створювати та модифікувати дерево поведінки, додаючи нові вузли або змінюючи існуючі, щоб адаптувати поведінку персонажів до різних ситуацій у грі. Це дозволяє створювати персонажів із складною та варіативною поведінкою, що робить ігровий світ більш живим та непередбачуваним для гравця.

Однією виключно особливістю алгоритму Behavior Tree є його зручність у розумінні та візуалізації. Дерево поведінки може бути легко представлене у вигляді графічної структури, що дозволяє розробникам швидко аналізувати та вдосконалювати поведінку персонажів.

Алгоритм Finite State Automaton є потужним інструментом для створення реалістичного та динамічного Штучного Інтелекту в ігрях. В основі цього підходу лежить ідея моделювання поведінки персонажа у вигляді скінченного автомату, де кожен стан представляє конкретну дію або набір дій, які персонаж може виконати, а перехоли між станами відбуваються відповідно до певних умов та взаємодій з ігровим середовищем.

Використання алгоритму Finite State Automaton дозволяє розробникам створювати ефективний та оптимізований ШІ для ігрових персонажів. Завдяки чіткій структурі та визначенню конкретних станів та переходів, реалізація поведінки персонажів стає більш простою та зрозумілою для розробників.

Наприклад, у приглядливих іграх, персонаж може мати такі стани як "спокійний", "переслідує", "атакує" або "втікає". Залежно від дій гравця та обставин, персонаж може змінювати свій стан і виконувати відповідні дії.

Однією з головних переваг використання алгоритму Finite State Automaton є його ефективність та низький рівень складності. Завдяки оптимізації реалізації, алгоритм може працювати ефективно навіть у великих ігрових світах з великою інтерактивністю та великою кількістю персонажів.

Висновок. Отже, всі три алгоритми використовуються для моделювання поведінки об'єктів в ігрі, зокрема для створення штучного інтелекту персонажів. Кожен з цих алгоритмів використовує концепцію станів і переходів між ними для представлення поведінки об'єктів. Вони дозволяють розробникам створювати складних та реалістичних ігрових персонажів, які реагують на дії гравця та зміни в ігровому світі.

Алгоритм State Machine базується на визначенні окремих станів та переходів між ними, що робить його добре підходящим для реалізації простих та складних систем. Behavior Tree дозволяє вирішити більш складну поведінку з використанням деревовидної структури, де кожен вузол представляє певну дію або поведінку. Алгоритм Finite State Automaton також використовує концепцію станів і переходів, але він може бути більш загальним і абстрактним, що дозволяє моделювати різноманітні системи, не обмежуючись тільки двома станами (активний та неактивний), як у State Machine. Behavior Tree володіє більшою гнучкістю та модульністю.

Загалом, вибір між цими алгоритмами залежить від конкретних потреб і вимог проекту, а також від рівня складності поведінки, яку потрібно моделювати.

Список використаних джерел:

1. A Study of Optimization Models for Creation of Artificial Intelligence for the Computer Game in the Tower Defense Game / O. Mazurova та ін. 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T). 2020. С. 491–496.
2. Unity AI Development: A Finite-state Machine Tutorial. URL: https://www.toptal.com/unity-unity3d/unity-ai-development-finite-state-machine-tutorial (дата звернення: 04.03.2024).
3. Introduction to the Unity State Machine Pattern. URL: https://niraciparyan.medium.com/introduction-to-the-unity-state-machine-pattern-ad3bce7d9f7c (дата звернення: 04.03.2024).
4. How to create a simple behaviour tree in Unity/C#. URL: https://medium.com/geekculture/how-to-create-a-simple-behaviour-tree-in-unity-c-3964cb3b5b6e (дата звернення: 04.03.2024).

Підсумки

- Unity забезпечує гнучкі можливості для реалізації та тестування ігрового ШІ.
- A* найкращий для складних середовищ, Дейкстра гарантує найкоротший шлях.
- Greedy Best First Search швидкий, але не завжди знаходить оптимальний шлях.
- Depth First Search не ефективний для нашої ситуації, але може бути корисним для обходу лабіринтів.
- Поведінкові дерева та машини станів добре підходять для створення адаптивного ігрового штучного інтелекту.
- Оптимізація алгоритмів за часом і пам'яттю критична для продуктивності гри.

Дякую за увагу!

ДОДАТОК В

Апробація результатів роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ XXVIII МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ»

16 – 18 квітня 2024 р.

Том 6

КОНФЕРЕНЦІЯ
«ІНФОРМАЦІЙНІ ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ»
INFORMATION INTELLIGENT SYSTEMS

Харків 2024

УДК 004.89

ДОСЛІДЖЕННЯ МЕТОДІВ СТВОРЕННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ РІЗНИХ СЦЕНАРІЇВ ГРИ

Пилявський Д. І.
Науковий керівник – к.т.н., доцент Назаров О. С.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: dmytro.pyliavskyi@nure.ua

The work considers the relevance of using advanced algorithms of artificial intelligence, such as State Machine, Behavior Tree and Finite State Automaton, for modeling the behavior of game characters in modern games. It considers the problems related to the choice of the optimal algorithm and its impact on the realism and efficiency of the gameplay. In addition, the article provides a brief overview of each of the algorithms and their features in the context of creating artificial intelligence for game characters.

Актуальність та постановка проблеми. У сучасній галузі розробки ігор та розвитку штучного інтелекту створення реалістичних та ефективних ігрових персонажів є ключовим завданням. Гравці все більше очікують від ігрових персонажів складної та непередбачуваної поведінки, яка адаптується до їхніх дій та оточуючого середовища. Тому актуальним є використання передових алгоритмів штучного інтелекту, таких як State Machine, Behavior Tree та Finite State Automaton, для моделювання поведінки ігрових персонажів.

Проблема полягає в тому, щоб знайти оптимальний спосіб використання алгоритмів штучного інтелекту для створення реалістичних ігрових персонажів, з урахуванням вимог до ефективності, адаптивності та оптимізації геймплею.

Існуючі альтернативні рішення. Окрім зазначених вище алгоритмів існують й інші альтернативні підходи для створення реалістичних ігрових персонажів. Наприклад, є машини підтримки рішень, що являють собою структури даних, які використовуються для моделювання послідовності рішень. Вони можуть бути корисними для реалізації великої кількості варіантів поведінки та управління складними сценаріями гри. Дуже поширеною альтернативою є використання методів навчання з підсиленням або навчання з учителем для створення ігрових персонажів зі здатністю до самоосвіти та адаптації до гравця та середовища. Не менш поширеним є генетичні алгоритми, які використовуються для еволюції поведінки персонажів відповідно до визначених критеріїв ефективності та реалістичності.

Основні матеріали дослідження. Алгоритм State Machine – це потужний інструмент, що використовується для моделювання поведінки об'єктів в іграх. Використання алгоритму State Machine дозволяє

розробникам створювати переконливих та реалістичних ігрових персонажів, які діють і реагують на оточуючий світ і гравця.

Головна ідея алгоритму полягає в тому, щоб розбити поведінку об'єкту на окремі стани та переходи між ними. Кожен стан представляє собою конкретну дію або набір дій, які об'єкт може виконати у певному контексті. Переходи визначають умови, за яких об'єкт переходить з одного стану в інший. Цей підхід дозволяє створити структуру, яка легко розширюється та модифікується, а також забезпечує чітку логіку поведінки.

За допомогою алгоритму State Machine можна створити гнучку систему, яка дозволяє персонажам адаптуватися до змін і швидко реагувати на дії гравця. Наприклад, у військових стратегічних іграх, ворожі війська можуть мати різні стратегії в залежності від поточної ситуації на полі бою. Вони можуть переходити зі стану "атаки" до стану "захисту" або "відступу" в залежності від кількості ворожих військ та стану їхньої амуніції.

Алгоритм Behavior Tree відіграє важливу роль у створенні реалістичного та цікавого Штучного Інтелекту в іграх. Цей підхід до моделювання поведінки персонажів базується на ідеї представлення їхньої діяльності у вигляді дерева, де кожен вузол представляє певну дію або поведінку, що може бути виконана. Використання алгоритму Behavior Tree дозволяє розробникам створювати складних та реалістичних ігрових персонажів, які реагують на дії гравця та оточуючий світ з урахуванням різноманітних умов і обставин.

Однією з переваг використання Behavior Tree є його гнучкість та модульність. Розробники можуть легко створювати та модифікувати дерево поведінки, додаючи нові вузли або змінюючи існуючі, щоб адаптувати поведінку персонажів до різних ситуацій у грі. Це дозволяє створювати персонажів зі складною та варіативною поведінкою, що робить ігровий світ більш живим та непередбачуваним для гравця.

Однією важливою особливістю алгоритму Behavior Tree є його зручність у розумінні та візуалізації. Дерево поведінки може бути легко представлене у вигляді графічної структури, що дозволяє розробникам швидко аналізувати та вдосконалювати поведінку персонажів.

Алгоритм Finite State Automaton є потужним інструментом для створення реалістичного та динамічного Штучного Інтелекту в іграх. В основі цього підходу лежить ідея моделювання поведінки персонажів у вигляді скінченного автомату, де кожен стан представляє конкретну дію або набір дій, які персонаж може виконати, а переходи між станами відбуваються відповідно до певних умов та взаємодій з ігровим середовищем.

Використання алгоритму Finite State Automaton дозволяє розробникам створювати ефективний та оптимізований ШІ для ігрових персонажів. Завдяки чіткій структурі та визначенню конкретних станів та переходів,

реалізація поведінки персонажів стає більш простою та зрозумілою для розробників.

Наприклад, у пригодницьких іграх, персонаж може мати такі стани як "спокійний", "переслідує", "атакує" або "втікає". Залежно від дій гравця та обстановки, персонаж може змінювати свій стан і виконувати відповідні дії.

Однією з головних переваг використання алгоритму Finite State Automaton є його ефективність та низький рівень складності. Завдяки оптимізованій реалізації, алгоритм може працювати ефективно навіть у великих ігрових світах зі складною інтерактивністю та великою кількістю персонажів.

Висновок. Отже, всі три алгоритми використовуються для моделювання поведінки об'єктів в іграх, зокрема для створення штучного інтелекту персонажів. Кожен з цих алгоритмів використовує концепцію станів і переходів між ними для представлення поведінки об'єктів. Вони дозволяють розробникам створювати складних та реалістичних ігрових персонажів, які реагують на дії гравця та зміни в ігровому світі.

Алгоритм State Machine базується на визначенні окремих станів та переходів між ними, що робить його добре підходящим для реалізації простих та складних систем. Behavior Tree дозволяє виражати більш складну поведінку з використанням деревовидної структури, де кожен вузол представляє певну дію або поведінку. Алгоритм Finite State Automaton також використовує концепцію станів і переходів, але він може бути більш загальним і абстрактним, що дозволяє моделювати різноманітні системи, не обмежуючись тільки двома станами (активний та неактивний), як у State Machine. Behavior Tree володіє більшою гнучкістю та модульністю.

Загалом, вибір між цими алгоритмами залежить від конкретних потреб і вимог проекту, а також від рівня складності поведінки, яку потрібно моделювати.

Список використаних джерел:

1. A Study of Optimization Models for Creation of Artificial Intelligence for the Computer Game in the Tower Defense Genre / O. Mazurova та ін. 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T). 2020. С. 491-496.

2. Unity AI Development: A Finite-state Machine Tutorial. URL: https://www.toptal.com/unity-unity3d/unity-ai-development-finite-state-machine-tutorial (дата звернення: 04.03.2024).

3. Introduction to the Unity State Machine Pattern. URL: https://mracipayam.medium.com/introduction-to-the-unity-state-machine-pattern-ad3bce7d987c (дата звернення: 04.03.2024).

4. How to create a simple behaviour tree in Unity/C#. URL: https://medium.com/geekculture/how-to-create-a-simple-behaviour-tree-in-unity-c-3964c84c060e (дата звернення: 04.03.2024).

ДОДАТОК Г

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-22-3 (група)

Пилявський Дмитро Ігорович
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

зауважень немає

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

09.06.2024