2

Ігрова індустрія постійно розвивається завдяки технологічним досягненням і зростанню інтересу до віртуальних розваг. Ключову роль в якості ігрового процесу відіграє штучний інтелект, який забезпечує реалістичність і цікавість гри. Платформа Unity відома своєю гнучкістю та широкими можливостями для створення ігор різних жанрів, що робить дослідження методів створення ігрового ШІ на цій платформі особливо актуальним.

3

### Об’єкт дослідження

Об’єктом дослідження є методи створення ігрового ШІ для різних сценаріїв гри на платформі Unity, зокрема алгоритми пошуку шляху та прийняття рішень для агентів у грі.

### Предмет дослідження

Предметом дослідження є ефективність алгоритмів пошуку шляху та алгоритмів прийняття рішень, їх реалізація, простота та доречність у контексті платформи Unity.

### Чітке визначення напряму дослідження

Метою дослідження є аналіз і порівняння алгоритмів створення ігрового ШІ для різних сценаріїв гри на платформі Unity. Дослідження включає порівняння ефективності алгоритмів пошуку шляху та прийняття рішень у контексті ігрових ситуацій.

4

Покращення зрозумілості та структури алгоритмів є важливим для забезпечення легкої модифікації та розширення. Це включає створення добре документованих, модульних і зрозумілих кодів, що полегшує їхнє вдосконалення та підтримку.

Уникання непотрібних ітерацій та оптимізація використання циклів допомагає зменшити витрати часу на виконання алгоритмів. Це може включати використання ефективних алгоритмічних конструкцій та уникнення зайвих операцій.

Використання оптимальних структур даних є важливим для забезпечення ефективного доступу та модифікації даних. Вибір відповідних структур даних допомагає підвищити продуктивність алгоритмів та зменшити витрати на їх виконання.

Розробка алгоритмів, які можуть ефективно працювати як з невеликими, так і з великими обсягами вхідних даних, є важливим аспектом для забезпечення їхньої універсальності та масштабованості.

Оцінка витрат часу та ресурсів на оптимізацію алгоритмів порівняно з отриманими вигодами є важливим для прийняття обґрунтованих рішень щодо впровадження оптимізації. Це включає аналіз компромісів між продуктивністю, використанням ресурсів та складністю реалізації.

Зменшення часової складності алгоритмів є важливим завданням для досягнення швидшого виконання та підвищення продуктивності гри. Це включає аналіз і оптимізацію алгоритмів для скорочення часу виконання, що є особливо важливим у реальних ігрових умовах, де затримки можуть негативно вплинути на геймплей.

Мінімізація просторової складності алгоритмів допомагає зменшити вимоги до пам'яті, що є критичним фактором, особливо для мобільних пристроїв та інших обмежених середовищ. Оптимізація використання пам'яті включає вибір ефективних структур даних і методів управління пам'яттю.

Ці аспекти є надзвичайно важливими при розробці програмного забезпечення, особливо в ігровій індустрії, де продуктивність та плавність геймплею є ключовими факторами успіху. Оптимізація алгоритмів ШІ дозволяє створювати більш реалістичні, ефективні та захоплюючі ігрові враження для користувачів.

5

6

7

Методологія дослідження включає проведення експериментів з використанням алгоритмів пошуку шляху та прийняття рішень для агентів у різних ігрових сценаріях на платформі Unity. Основні етапи методології:

1. \*\*Аналіз літератури\*\* - вивчення наукових статей та публікацій за темою дослідження.

2. \*\*Розробка експериментальної бази\*\* - створення ігрового середовища та реалізація обраних алгоритмів у Unity.

3. \*\*Проведення експериментів\*\* - тестування алгоритмів на різних ігрових сценаріях.

4. \*\*Збір та аналіз даних\*\* - оцінка ефективності та продуктивності алгоритмів за показниками, такими як час виконання, використання пам'яті, довжина шляху та кількість відвіданих вузлів.

5. \*\*Порівняння результатів\*\* - порівняльний аналіз алгоритмів для визначення їхніх переваг та недоліків у різних умовах.

6. \*\*Формулювання висновків\*\* - розробка рекомендацій для оптимізації ігрового ШІ на платформі Unity.

8

### Breadth First Search (BFS)

Breadth First Search працює рівня за рівнем, починаючи з початкового вузла і досліджуючи всі його сусіди перед переходом на наступний рівень. BFS шукає найкоротший шлях у неважених графах, оскільки досліджує всі можливі шляхи рівномірно. Однак, він може споживати значну кількість пам'яті при великих графах.

9

### Алгоритм Дейкстри (Dijkstra)

Алгоритм Дейкстри знаходить найкоротший шлях від початкової точки до всіх інших точок графа. Використовує пріоритетну чергу для поступового розширення найкоротшого шляху, гарантуючи знаходження найкоротшого маршруту в будь-яких графах, включаючи важені. Однак, може бути менш ефективним у великих середовищах з відомим місцезнаходженням цілі.

10

### A\* Алгоритм

Алгоритм A\* є одним з найефективніших для пошуку найкоротших шляхів у графах. Він використовує евристичну функцію для оцінки вартості шляху від початкової точки до цільової. A\* комбінує пошук за найменшою вартістю та жадібний пошук, що дозволяє швидше знаходити оптимальний шлях. Алгоритм гарантує знаходження найкоротшого шляху, але може вимагати значного обсягу пам'яті для великих графів .

11

### Greedy Best First Search

Алгоритм Greedy Best First Search обирає вершини на основі евристичної функції, яка оцінює вартість до цільової вершини. Це робить алгоритм швидким, але він не гарантує знаходження найкоротшого шляху, оскільки не враховує поточну вартість шляху, лише евристику

12

### Depth First Search (DFS)

Алгоритм Depth First Search працює за принципом глибини: він проходить від початкової вершини до найглибшої можливості, перш ніж відступати назад. Використовує стек для управління фронтом пошуку. Підходить для задач, де важливо досліджувати всі можливі шляхи, наприклад, у лабіринтах, але не ефективний для знаходження найкоротших шляхів у великих графах через глибокий пошук у всіх напрямках.

13

### Bidirectional Search

Bidirectional Search працює одночасно з двох напрямків: від початкової вершини та від цільової вершини, доки обидва фронти пошуку не зустрінуться. Це дозволяє значно зменшити кількість оброблених вершин, що робить алгоритм дуже ефективним для великих графів. Алгоритм гарантує знаходження найкоротшого шляху, але потребує додаткового зберігання даних для обох пошуків .

14

### Lee Algorithm

Алгоритм Lee працює за принципом пошуку в ширину (BFS), розширюючи фронт пошуку рівномірно у всі сторони від початкової точки до цільової точки. Гарантує знаходження найкоротшого шляху в неважених графах, але може бути повільним для великих графів​​.

15

### Dynamic Programming Maze

Алгоритм Dynamic Programming Maze використовує метод динамічного програмування для знаходження найкоротшого шляху у двовимірних сітках. Він поступово оновлює вартість шляхів від стартової точки до кожної вершини, забезпечуючи найкоротший шлях до цільової вершини. Подібний до алгоритму Дейкстри, але використовує інший підхід до оновлення вартостей та обробки вершин. Алгоритм гарантує знаходження найкоротшого шляху, але може бути повільним для великих сіток через значні обчислювальні ресурси .