Єгор Місько,
перший (бакалаврський) рівень вищої освіти,
освітньо-професійна програма: «Середня освіта (Інформатика)»,
Житомирський державний університет імені Івана Франка
науковий керівник: ***Олександр Мосіюк****,*
кандидат педагогічних наук,
доцент кафедри комп'ютерних наук та інформаційних технологій,
Житомирський державний університет імені Івана Франка

ВИКОРИСТАННЯ АЛГОРИТМУ ПОШУКУ ШЛЯХУ A* ПРИ РОЗРОБЦІ ІГОР

Анотація. *У статті описується використання алгоритму пошуку шляху A* при розробці комп'ютерних стратегічних ігор. Проаналізовано основні кроки виконання алгоритму та порівняно їх ефективність із алгоритмом Дейкстри.*

Ключові слова: *алгоритм, алгоритму пошуку шляху A*, комп'ютерна стратегічна гра.*

Постановка проблеми. Зростання популярності відеоігор у всьому світі стало поштовхом до зростання зацікавленості дослідників у вирішенні багатьох питань штучного інтелекту, пов'язаних з відеоіграми. До них варто віднести такі задачі: на прийняття рішень, пошуку шляху, моделювання стратегії поведінки персонажів тощо.

Задачі пошуку шляху зазвичай стосується знаходження найкоротшої траєкторії між двома кінцевими точками. Прикладами таких проблем є: планування транзиту, маршрутизація телефонного трафіку, навігація лабіринтом. Наприклад у рольових та стратегічних іграх персонажі в режимі реального часу, мають прокладати шлях до вказаної користувачем точки та уникнути перешкод.

Аналіз актуальних досліджень. Пошук шляхів у комп'ютерних іграх досліджувався протягом багатьох років. Це одна із найпопулярніших питань алгоритмізації. Різні алгоритми пошуку, такі як алгоритм Дейкстри, алгоритм пошуку по ширині та алгоритм пошуку по глибині у графах, були створені для вирішення проблеми найкоротшого шляху до появи алгоритму **A*** як доказово оптимального рішення для пошуку шляху.

З моменту створення, алгоритм успішно привернув увагу тисяч дослідників, щоб докласти зусиль до його удосконалення. Зокрема Сяо Ц. та Хао Ш. детально описують алгоритм **A*** у сучасних відеоіграх та способи його вдосконалення [2]; Зеяд А., Мохд С., Хошєнг К. розглядають основні способи пошуку шляху у відеоіграх [3]; в праці Ранджита М., Казака Н., Лінсі Дж. описано алгоритми штучного інтелекту при розробці відеоігор з можливостями до адаптації [1].

У цьому контексті важливо фахівцям слідкувати за новітніми підходами до оптимізації алгоритму, тому **метою статті** є аналіз використання алгоритму пошуку шляху **A*** при розробці відеоігор та способи його вдосконалення.

Виклад основного матеріалу. Алгоритм **A*** – це загальний алгоритм пошуку, який можна використовувати для розв'язку багатьох складних задач, і пошук шляху просто є однією з них. Для побудови траєкторії алгоритм **A*** неодноразово «досліджує» найперспективніше незвідане місце, яке він зберігає у своїй пам'яті. Коли локація повністю досліджена, алгоритм завершується, якщо це кінцева точка; в іншому випадку він позначає всіх сусідів цього місця для подальших досліджень.

Загалом алгоритм складається містить наступну послідовність.

1. Початковий вузол додається до відкритого списку.
2. Далі виконується послідовність таких кроків:
    a. Пошук вузла із найнижчим значенням f у відкритому списку і звернення до нього.
    b. Перенесення його у закритий список.
    c. Для кожного доступного вузла з поточного вузла, перевіряються такі умови:
        I. Якщо він є у закритому списку, він ігнорується.
        II. Якщо його немає у відкритому списку, то він додається до нього; поточний вузол робиться батьківським для цього вузла та записуються значення f, g та h цього вузла.
        III. Якщо він вже є у відкритому списку, то перевіряється чи це кращий шлях. У випадку позитивного результату батьківський вузол замінюють на поточний вузол і перераховують значення f та g.
    d. Зупинка відбувається тоді, коли
        I. Цільовий вузол додається до закритого списку.
        II. Не вдалося знайти цільовий вузол, а відкритий список порожній.
3. Трасування назад від цільового вузла до початкового формує шуканий шлях.

У стандартній термінології, що використовується, коли говорять про алгоритм **A***, g (n) представляє точну «вартість» від початкової точки до будь-якої точки n, h(n) представляє орієнтовну «вартість» від точки n до пункту призначення, а f (n) = g (n) + h (n).

Секрет успіху **A*** полягає в тому, що він розширює алгоритм Дейкстри, вводячи евристичний підхід. Алгоритм Дейкстри гарантовано знайде

найкоротший шлях у зв’язаному зваженому графі, якщо жодне з ребер не має від’ємного значення, але він недостатньо ефективний, оскільки всі можливі стани повинні бути спочатку дослідженні. У той же час алгоритм **A*** значно покращує обчислювальну ефективність, вводячи евристичний підхід. Використання евристичного підходу означає, що замість вичерпного розширення розглядаються лише ті стани, які виглядають як кращі варіанти. Евристична функція, яка використовується в алгоритмі **A***, полягає в оцінці «вартості» від будь-яких вузлів на графу до бажаного пункту призначення. Якщо кошторисна «вартість» точно дорівнює реальній «вартості», тоді вибираються лише вузли, що знаходяться на найкращому шляху, і більше нічого не розширюється. Таким чином, якісна евристична функція, яка може точно оцінити вартість, може зробити алгоритм набагато швидшим.

На рис. 1 продемонстровано процес зростання пошуків із використанням різних евристичних витрат при спробі подолати велику перешкоду. Коли евристика дорівнює нулю (показано на рис. 1а), алгоритм **A*** переходить до алгоритму Дейкстри. Всі сусідні вузли розширені. Коли евристика використовує евклідову відстань (показано на рис. 1б), розглядаються лише ті вузли, які виглядають як кращі варіанти. Коли евристика трохи завищена (показано на рис. 1с), то це призводить до вивчення набагато меншої кількості вузлів, ніж евристичні підходи, що не переоцінюють. Проте однозначної підходу до визначення ступеня переоцінки на даний час не має, що є певною проблемою для цього алгоритму.

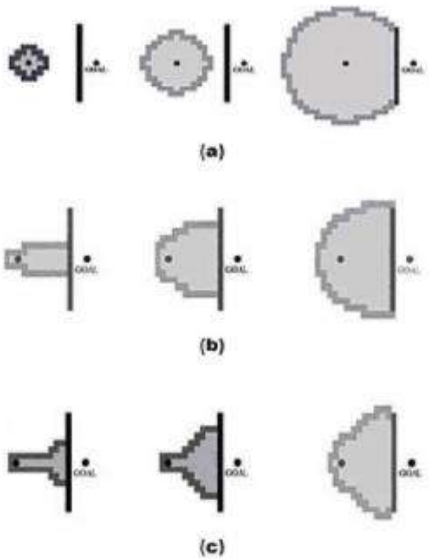

*Рис 1. Графічна реалізація процесу виконання алгоритму A**

Незважаючи на те, що **A*** є найефективнішим алгоритмом пошуку шляху, його потрібно використовувати розумно; в деяких випадках його використання може призвести до марного використання ресурсів. Алгоритм **A*** вимагає величезного обсягу пам'яті для відстеження прогресу кожного пошуку, особливо при пошуку у великих і складних середовищах. Зменшення необхідної пам’яті для пошуку шляхів є складною проблемою в ігровому штучному інтелекту. У цій галузі було проведено багато робіт.

Найпопулярніший спосіб уникнути втрати пам'яті – це попередньо виділити мінімальний обсяг пам'яті [2]. Загальна ідея полягає в тому, щоб виділити частинку пам'яті (Node Bank) до того, як **A*** розпочне виконання. Якщо вся пам'ять вичерпується, то створюється новий буфер для прогресу пошуку. Допускаються зміни розміру цього буфера, щоб менше пам'яті було витрачено даремно. Обсяг мінімальної пам'яті головним чином залежить від складності середовища. Таким чином, налаштування потрібно виконувати перед застосуванням цієї стратегії до конкретного додатка.

Після того, як вузол було ініціалізовано у Node Bank, його потрібно десь розмістити для швидкого пошуку. Хеш-таблиця може бути найкращим вибором, оскільки вона дозволяє постійно зберігати і шукати дані. Така хеш-таблиця дозволяє миттєво з’ясувати, чи перебуває певний вузол у списках **ЗАКРИТО** або **ВІДКРИТО**.

Черга пріоритетів є найкращим спосібом підтримувати список **ВІДКРИТО**. Це може бути реалізовано за допомогою двійкової купи. Введення нової структури даних для зберігання даних може допомогти значно пришвидшити **A***.

Висновки. Основа алгоритму пошуку шляхів – це лише невелика частина алгоритмів, які використовуються при створенні комп’ютерних відеоігор. Алгоритм **A*** є одним із найпопулярніших алгоритмів для пошуку оптимального шляху. Для його оптимізації було прикладено багато зусиль. Шляхи підвищення ефективності пошуку **A*** включають оптимізацію базового простору пошуку, зменшення використання пам'яті, поліпшення евристичних функцій та впровадження нових структур даних.

Список використаних джерел і літератури.

1. Ranjitha M., Kazaka N., Lincy J. Artificial Intelligence Algorithms and Techniques in the computation of Player-Adaptive Games. URL: https://iopscience.iop.org/article/10.1088/1742-6596/1427/1/012006/pdf

2. Xiao C., Hao Sh. A*-based Pathfinding in Modern Computer Games. *International Journal of Computer Science and Network Security,* 2011. Vol.11, №1, 2011. P. 125-130. URL: https://www. Computer_Games researchgate.net/publication/267809499_A-based_Pathfinding_in_Modern_

3. Zeyad A. A., Mohd Sh. S., Hoshang K. A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games. International Journal of Computer Games Technology, 2015. URL: https://www.hindawi.com/journals/ijcgt/2015/736138/