УДК 004.89

ДОСЛІДЖЕННЯ МЕТОДІВ СТВОРЕННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ РІЗНИХ СЦЕНАРІЇВ ГРИ

Пилявський Д.І.
Науковий керівник – кандидат технічних наук, 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.

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.

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.

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. 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).

2. 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).

3. 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).