УДК 004.896

Д. О. Серебренніков

О. В. Сілагін

АНАЛІЗ ПІДХОДІВ ДО СТВОРЕННЯ ІГРОВОГО ШТУЧНОГО ІНТЕЛЕКТУ

Вінницький національний технічний університет

Анотація

Проаналізовано три поширені підходи до ігрового штучного інтелекту: Utility AI, скінченні автомати (FSM) та дерево поведінки (Behaviour Tree). Оцінено їх за критеріями швидкодії, гнучкості, масштабованості та взаємодії з гравцем

Ключові слова: *штучний інтелект, гра, ігровий штучний інтелект, Utility Ai, інструмент*

Abstract

Three common approaches to gaming AI are analyzed: Utility AI, Finite State Machines (FSM), and Behavior Tree. They are evaluated based on the criteria of speed, flexibility, scalability, and player interaction.

Keywords: *artificial intelligence, game, gaming artificial intelligence, Utility Ai, tool*

Вступ

Найкращим початком дослідження штучного ігрового інтелекту буде використання експертної думки в області вивчення, що буде продовжена особистим внеском дослідно-наукової роботи. Базовим джерелом слугує найпопулярніша серія книг з ігрової розробки відомим британським автором Ian Millington під назвою “AI for Games”. [1, 2, 3]

“…game AI is a matter of perception, rather than arriving at the best…” цитата третього видання як найкраще описує різницю між інтелектом бізнес додатків та ігор. [3]

Що ж мається на увазі під цим коротким висловом?

Ігровий інтелект не завжди повинен прагнути бути академічно точним, навіть якщо це зробити легше, швидше і продуктивніше. Це може бути цікавим з точки зору симуляції, але ігри - це спрощення реальності із зручним інтерфейсом який дає можливість відпочити від досконалих моделей всесвіту.

Але ця характеристика не означає простоту моделювання подібної технології, адже, якщо взяти за приклад фізику, яка використовується майже у всіх іграх, то змоделювати її набагато складніше такою, що може працювати з достатньою швидкодією та компенсувати не цікавість реальних фізичних законів - 2D, різношарова колізія, саб-кроки, softbody та багато інших технік використовуються аби досягти кращих результатів, тоді як можна було б створити кришталеві сітки атомів із молекулярними законами аби вирішити будь-яку задачу у будь-якій грі - але це неможливо втілити навіть на 1% з актуальними пристроями.

Веселий інтелект поліцейських машин - це та сама механіка яка зробила першу гру серії «GTA» такою популярною, хоча спочатку була лише помилкою у коді.

Ігровий штучний інтелект:

- Чесний. Гравець уявляє, що створений ігровий світ є реальним, а тому він очікує хочаб базові правила за якими він сам керується, наприклад якщо переслідуючий супротивник буде бачити через стіни - такий код набагато легше написати, адже не потрібно обраховувати ніякі алгоритми, наприклад A*, для подолання перешкод, але це може дуже сильно зіпсувати сприйняття гри у цілому.

- Несподіваний. Інтелект не повинен наскучити гравцю. Нехай він показує різні рішення навіть при однакових обставинах, інколи виконуючи абсурдні речі, наприклад басейни у грі «Sims», або гіганти у грі «Elder Scrolls», або перевдягання у грі «Hitman».

- Швидкодійний. Хоч характеристика обробки і не є суто притаманним явищем ігровому розуму, але нажаль дуже велика кількість ідей не можлива у втіленні із-за низької продуктивності на девайсах користувачів, якщо гравець не буде мати бажаних 60 FPS, тоді йому навіть не буде цікаво знати як працює інтелект у дійсності.

- Гнучкий у редагуванні. Отримуємий досвід гравцем - це не єдина ціль механіки. Логіка штучного інтелекту повинна бути піддатлива у встановленні дизайну бажаної поведінки, вона може бути більш криптичною або лінгвістичною або візуальною. Більшість алгоритмів вирішують свою гнучкість ще на стадії математичної моделі і це одна з важливих характеристик, яка допомагає підбирати правильний інструмент під задачу.

Отож, проаналізувавши проблему, було прийнято рішення провести дослідження області ігрового штучного інтелекту для того, щоб обрати найкращий алгоритм для реалізації подальшого інструменту.

Результат дослідження

Utility Ai (Utility system) - підхід, що викликає дії залежно від значення корисності.

Вектор корисності може визначатися від стану агенту або параметрів середовища, а самі дії можуть бути реалізовані у вигляді процедур або даних. Алгоритм є найпростішим, оскільки базується на чітких математичних розрахунках, а Utility може бути залежністю від декількох інших Utility. Рисунок 1

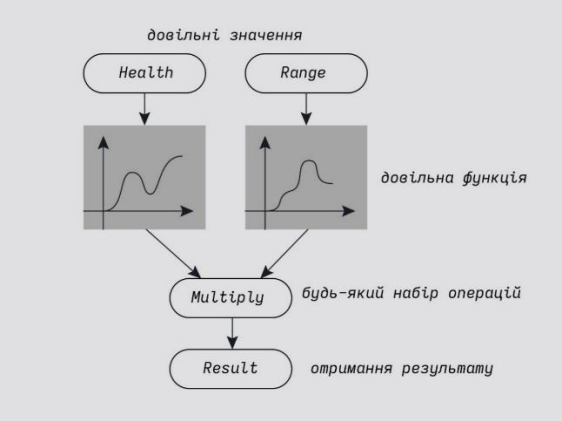

Рисунок 1 – Схема роботи Utility Ai на прикладі

Перевагою є легка і швидка реалізація, а також висока оптимізація за рахунок відсутності додаткових інструкцій, спрямованих на покращення читаємості.

Недоліком є складність розуміння та масштабування системи, особливо зі збільшенням кількості параметрів і дій.

Особливо ефективно підходить для вирішення однозалежних задач, наприклад коли гучність звуку залежить від часу перебування у матчі.

Практично застосовується у всіх іграх, у простіших покриває більший об’єм логіки, а у складніших замінюється або удосконалюється іншими алгоритмами. Наприклад гра Sims 3 використовує саме цей математичний алгоритм для покриття логіки ігрових персонажів.

Finite State Machine (FSM) - система, що використовує граф із одним активним вузлом, а переходи між вузлами викликають відповідні дії.

Може бути реалізований по-різному: часто використовують шаблон State, де кожен вузол представлений окремим об’єктом з власною логікою та можливістю ініціювати переходи. Водночас структура може бути централізованою — коли активація вузлів керується зовнішньою системою, без розділення на окремі об’єкти для кожного стану. Рисунок 2.

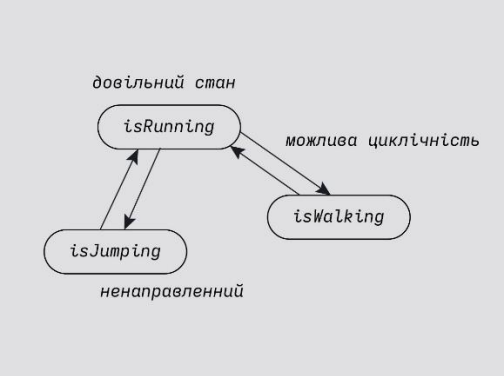

Рисунок 2 – Схема роботи Finite State Machine на прикладі

Перевагою є простота реалізації та чітке відокремлення логіки у вузли, що допомагає уникати складних дерев рішень.

Недоліками є обмежена масштабованість і гнучкість, оскільки основна логіка зазвичай прив’язана саме до моментів переходів між вузлами.

Особливо ефективно підходить для керування анімацією, де кількість вузлів невелика, а необхідна поведінка природно описується саме переходами.

Практично застосовується у більшості ігор, адже є базовим рішенням для відокремлення станів, що не повинні накладатися один на одного, тому у приклад можна привести будь-яку гру і вона скоріше за все буде мати декілька FSM реалізацій - верхнє меню гри «Dota 2» з одним активним станом.

Behaviour tree – дерево, верхні вузли, якого вирішують логіку над дочірніми вузлами.

Вузли зазвичай поділяють на:

- композитні - керують порядком виконання дочірніх
- декоратори - змінюють поведінку дочірнього вузла
- листові - виконують дії чи перевірки

Зображено на рисунку 3.

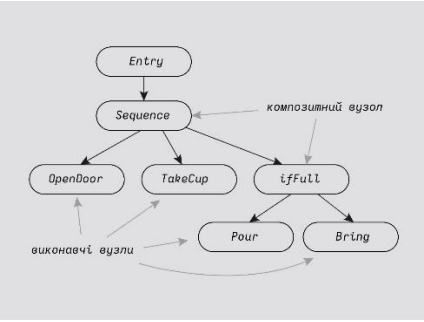

Рисунок 3 – Схема роботи Behaviour Tree на прикладі

Перевагою є зрозумілість, перевикористання, масштабованість, легкість у використанні.

Недоліком є додаткове зменшення швидкодії із-за функціонального стилю використання AoS (Aray of Structures).

Особливо ефективно підходить для побудови логіки, яку можна представити у вигляді простих, взаємопов'язаних блоків без складних зовнішніх залежностей.

Практично застосовується у грі "Halo".

Висновки

У роботі було досліджено підходи Utility AI, Finite State Machine та Behaviour Tree з погляду реалістичності поведінки, продуктивності, гнучкості редагування та масштабованості; встановлено, що Utility AI доцільний для швидких, метрик-керованих рішень і простих залежностей, FSM — для систем із взаємовиключними станами (анімації, UI), а Behaviour Tree — для ієрархічної, добре візуалізованої логіки з високою потребою у повторному використанні; на підставі аналізу рекомендовано комбінований підхід: застосовувати Behaviour Tree як «каркас» поведінки, в його листках використовувати утилітарні функції для пріоритизації дій, а локальні взаємовиключні стани оформлювати як вбудовані FSM; такий гібрид забезпечує керовану складність, передбачувану продуктивність і потрібний рівень «несподіваності» для гравця з урахуванням обмежень цільових платформ.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ian Millington. Ai for Games 1st Edition, 2006 року
2. Ian Millington. Ai for Games 2st Edition, 2009 року
3. Ian Millington. Ai for Games 3st Edition, 2019 року

Серебренніков Дмитро Олександрович *— студента групи 1КН-24м, кафедри комп'ютерних наук, Вінницький національний технічний університет, м. Вінниця, e-mail:* seredim6@gmail.com

Сілагін Олексій Віталійович *– канд. техн. наук, доцент кафедри комп'ютерних наук, Вінницький національний технічний університет, м. Вінниця, e-mail: avsilagin@vntu.edu.ua*

Dmytro Serebrennikov *— student of Computer Science Department group 1KH-24m, Vinnytsia National Technical University, Vinnytsia. e-mail: seredim6@gmail.com*

Oleksii Silagin *– Candidate of Technical Sciences, Associate Professor of the Department of Computer Science, Vinnytsia National Technical University, Vinnytsia, e-mail: avsilagin@vntu.edu.ua*