УДК 004.514

АЛГОРИТМИ ПОВЕДІНКИ NPC ПРОТИВНИКІВ В ІГРАХ ЖАНРУ METROIDVANIA

Крамська О.М.
e-mail: olena.kramska@nure.ua
Науковий керівник – ст. викл. Новіков Ю.С.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна

In this work, an intelligent NPC behavior algorithm for Metroidvania games is proposed based on Behavior Trees and Raycasting. The system integrates dynamic environment analysis to enable autonomous navigation and adaptive interaction with complex geometry, significantly enhancing the nature of AI. The research covers a comprehensive cycle of NPC states, ensuring efficient state management, scalability, and dynamic response to player actions. A practical implementation of the proposed modular architecture is demonstrated within the Unreal Engine 5 using Blueprints.

Жанр Metroidvania виник у 1980-х роках під впливом двох проєктів як Metroid та Castlevania, та протягом десятиліть він еволюціонував від простих лабіринтів з діями, зумовлених різними обмеженнями того часу, до складних екосистем з багатошаровою структурою рівнів та комбінаціями різних механік. Такий розвиток та розширення зумовили потребу в більш різноманітній поведінці ворожих NPC. Замість статичного патрулювання, сучасний гравець очікує від противників вільної орієнтації у просторі та динамічних реакцій на його дії. Незважаючи на новітні технології, створення таких адаптивних алгоритмів поведінки ускладнюється конфліктом між автономністю ШІ та структурою ігрового простору. Метою роботи є запропонування розробки гнучких алгоритмів, що дозволять NPC адекватно взаємодіяти з оточенням та відповідно реагувати на дії гравця.

Для реалізації різноманітної поведінки та реакції ворогів існують різні архітектурні моделі, кожна з яких має свої особливості та обмеження. Проаналізувавши найбільш розповсюдженні системи управління ШІ у сучасних іграх [1], було виділено три основні методи: скінченний автомат або машина станів (FSM), дерева поведінки (Behavior Trees) та планування дій (GOAP). Переваги та недоліки кожного з цих варіантів наведено у таблиці 1.

Після аналізу та порівняння попередніх моделей, для власного алгоритму поведінки ворожих NPC було обрано найбільш оптимальний варіант, а саме модель Behavior Trees. Для вирішення задачі для динамічного аналізу геометрії рівня можна інтегрувати систему променів Raycasting, яка дозволить противникам ідентифікувати перешкоди та межі

платформ на своєму шляху в реальному часі та залежно від цього коригувати логіку переміщення [3].

Таблиця 1 – Порівняння моделей ШІ поведінки

Модель	Переваги	Недоліки
FSM	- Висока продуктивність - Просте створення логіки для слабких ворогів	- Важке масштабування - Ускладнення структури при розширенні реакцій
Behavior Trees	- Модульність та гнучкість - Чітка структура ієрархії прийняття рішень	- Потреба в додаткових системах для аналізу геометрії світу - Навантаження на продуктивність при великій кількості активних вузлів
GOAP	- Високий рівень автономності - Здатність самостійно будувати складні плани для досягнення цілей	- Високе навантаження на обчислювальні ресурси - Важке балансування та прогнозування варіантів поведінки.

Для забезпечення різноманітної поведінки противників та взаємодії з персонажем гравця було визначено наступні стани, які будуть керуватися деревом поведінки: патрулювання (Patrol), тривога (Alert), переслідування (Chase), атака (Attack), загибель (Death) та відновлення (Respawn). Опис кожного зі станів із застосуванням Raycasting наведені у таблиці 2.

Таблиця 2 – Стани та логіка поведінки NPC

Стан	Опис поведінки	Застосування Raycasting
Patrol	Автономний рух в межах заданого маршруту	Променеве зондування поверхні перед NPC для визначення країв та вчасного розвороту
Alert	Тимчасова зупинка для аналізу оточення при фіксації подразників в зоні тригера	Перевірка прямої видимості та виявлення перешкод між NPC та гравцем
Chase	Активне наближення до гравця з метою скорочення дистанції для атаки	Постійне сканування поверхні для уникнення зіткнень та падіння
Attack	Перехід до відповідних бойових дій, або їх комбінацій (за наявності)	Розрахунок вектора атаки та перевірка досяжності цілі
Death	Припинення роботи логіки при нульовому показнику здоров’я	Деактивація сенсорної системи для оптимізації ресурсів
Respawn	Повернення у вихідну позицію у відновленому стані при виході гравця за межі визначеної тригерної зони	Скидання параметрів та повторна ініціалізація системи

Після визначення станів було побудовано базову блок-схему алгоритму (рис. 1), на якій показано основну логіку поведінки та послідовність переходів між станами [3].

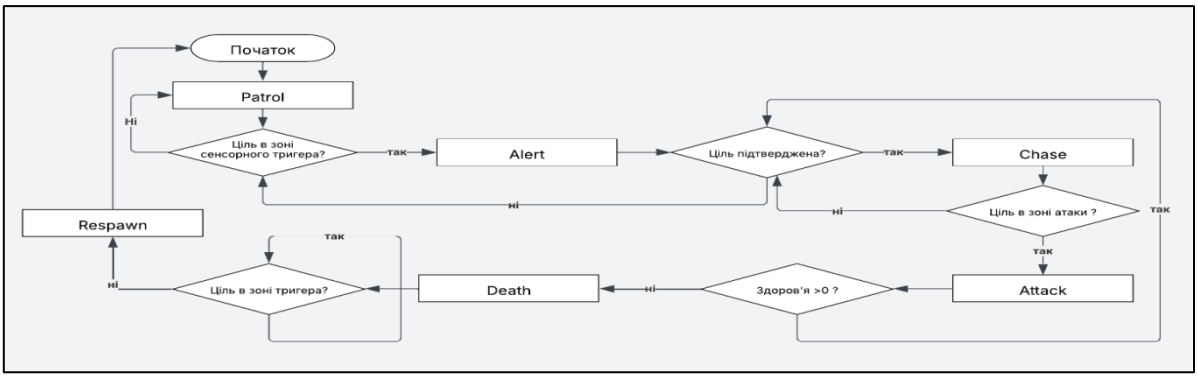

Рисунок 1 – Блок-схема роботи станів

На рисунку 2 зображено приклад практичної реалізації на рушії Unreal Engine 5 та візуальній мові програмування Blueprint виклику стану відновлення через тригерну зону. Логіка базується на події виходу персонажа гравця за межі зони, що запускає ітераційний цикл для пошуку всіх акторів визначеного класу в межах даної зони та їх повернення у вихідний стан через виклик відповідної функції.

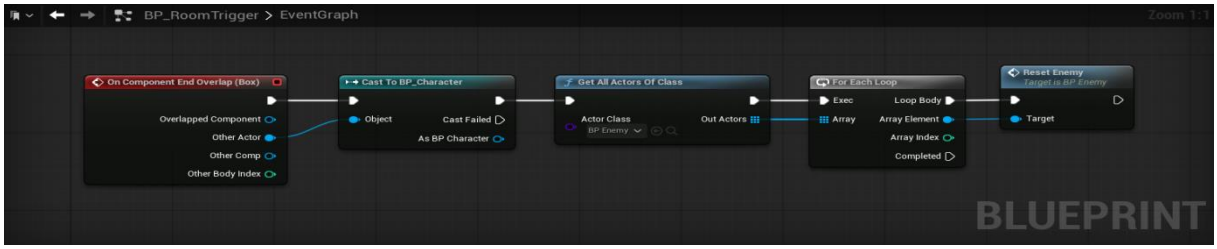

Рисунок 2 – Логіка роботи тригера для відновлення станів NPC

В результаті подібний алгоритм дозволяє зробити гнучку та масштабовану систему управління ШІ для NPC ігор жанру Metroidvania, використовуючи Behavior Trees у поєднанні з Raycasting.

Список використаних джерел:

1. Millington I. AI for Games. Taylor & Francis Group, 2019. 1010 p.

2. Marcos R., Sewell B. Blueprints Visual Scripting for Unreal Engine 5: Unleash the True Power of Blueprints to Create Impressive Games and Applications in UE5, 3rd Edition. Packt Publishing, Limited, 2022. 568 p.

3. Новіков Ю., Бідна Д. NPC’s schedule // Сучасні комп’ютерні та інформаційні системи і технології : матеріали II Всеукраїнської наук.-практ. інтернет-конф. (1–12 грудня 2021 р.). Мелітополь, 2021. С. 80–83.