

Реализация и исследование поведения ИИ-агента патрулирования в игровом движке Unreal Engine 5

Ёлгин Роман Николаевич

*Приамурский государственный университет имени Шолом-Алейхема
Студент*

Андриенко Иван Сергеевич

*Приамурский государственный университет имени Шолом-Алейхема
Студент*

Аннотация

В работе рассматривается реализация базового поведения искусственного интеллекта типа «патруль» в игровом движке Unreal Engine 5. На основе шаблона проекта с видом от третьего лица разработан ИИ-агент, совершающий циклическое перемещение по заданным контрольным точкам сцены. Логика поведения реализована средствами Blueprint на уровне контроллера ИИ (BP_AIController) и персонажа (BP_AICharacter) с использованием встроенной навигационной системы (NavMesh) и узла AI MoveTo.

Ключевые слова: Unreal Engine 5; искусственный интеллект; игровой ИИ; патрулирование; навигационная сетка; NavMesh; Blueprint; AI MoveTo.

Implementation and Analysis of Patrol AI Agent Behaviour in Unreal Engine 5

Yolgin Roman Nikolaevich

*Sholom-Aleichem Priamursky State University
Student*

Andrienko Ivan Sergeevich

*Sholom-Aleichem Priamursky State University
Student*

Abstract

This article presents the implementation of a basic patrol-type artificial intelligence behaviour in the Unreal Engine 5 game engine. Based on the third-person project template, an AI agent is developed that performs cyclic movement along predefined control points of the scene. The behaviour logic is implemented using Blueprint at the AI controller (BP_AIController) and character (BP_AICharacter) levels, relying on the built-in navigation system (NavMesh) and the AI MoveTo node.

Keywords: Unreal Engine 5; artificial intelligence; game AI; patrolling; navigation mesh; NavMesh; Blueprint; AI MoveTo.

1 Введение

1.1 Актуальность

Интеллектуальные агенты, управляющие персонажами-противниками и неигровыми персонажами, являются ключевым элементом современных компьютерных игр. Даже простые сценарии поведения — такие как патрулирование по маршруту, реагирование на игрока и возврат к исходному маршруту — существенно влияют на воспринимаемое качество игры.

1.2 Обзор исследований

Г. Н. Яннакакис и Ю. Тогелиус предоставляют современный обзор методов игрового ИИ (поиск пути, навигация, поведение агентов, обучение), формулируя общие принципы построения ИИ-систем без привязки к конкретному движку [1]. Рабин С. в своей работе включает практические статьи разработчиков игр о навигации, патрулировании, поведении NPC и оптимизации ИИ-систем в реальных проектах [2]. Х. Чжан и Х. Чжан рассматривают реализацию интеллектуального поиска пути на основе NavMesh в трёхмерных картах Unreal Engine 4, описывая построение навигационной сетки и интеграцию алгоритмов перемещения ИИ-персонажей [3]. И. Гупта, Р. Дамми и Р. Каур анализируют процесс проектирования и разработки шутера от третьего лица в формате «maze escape» на Unreal Engine, уделяя внимание структуре уровней, игровым механикам и использованию встроенных средств движка для организации поведения персонажей [4]. Т. Р. Раджапов рассматривает процесс разработки игрового приложения в жанре roguelike на движке Unreal Engine 5, уделяя особое внимание проектированию и созданию игрового окружения [5].

1.3 Цель исследования

Цель исследования – разработать и описать простую систему патрулирования ИИ-агента в Unreal Engine 5 на основе Blueprint, а также проанализировать её работу на тестовой сцене с учётом настроек навигации и коллизии патрульных точек.

2 Материалы и методы

В качестве программной платформы выбран игровой движок Unreal Engine 5. В качестве исходной конфигурации использован шаблон проекта Third Person с видом от третьего лица и стандартным играбельным персонажем.

Основными объектами разработки являются:

- Blueprint-класс BP_PatrolPoint (Actor), используемый в качестве маркера контрольных точек маршрута;
- Blueprint-класс персонажа BP_AICharacter (Character), выступающий в роли ИИ-агента;
- Blueprint-класс BP_AIController (AIController), реализующий логику патрулирования.

3 Результаты и обсуждения

3.1. Подготовка сцены и навигации

Для эксперимента был создан проект на Unreal Engine 5 на основе шаблона Third Person.

В качестве тестовой сцены использовалась стандартная карта ThirdPersonMap с добавленными несколькими препятствиями (статические кубы и цилиндр), формирующими пространство для перемещения агента.

На сцену добавили объём навигационной сетки, позволяющий ИИ-персонажу строить маршруты. На уровне разместили актор NavMeshBoundsVolume, охватывающий область, где должен двигаться агент. После построения навигации в режиме редактора (клавиша Р) теперь отображается зелёная область NavMesh (рис. 1).

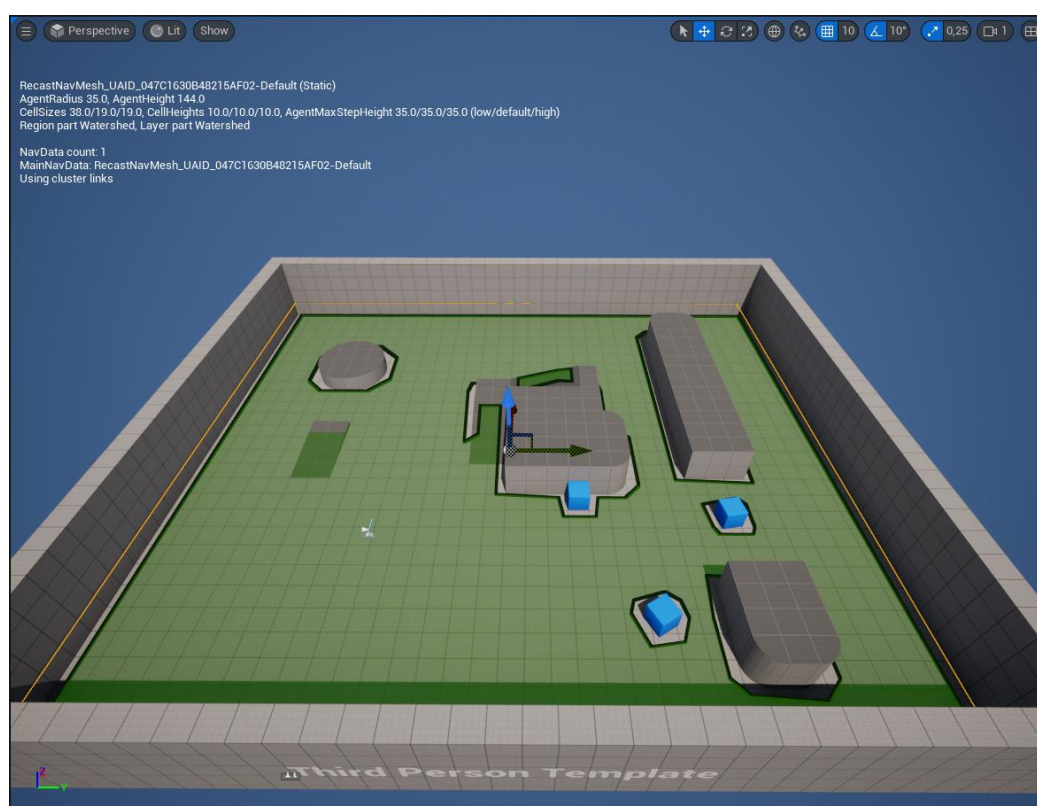


Рисунок 1. Скриншот уровня с включённым NavMesh

3.2. Реализация патрульных точек (BP_PatrolPoint)

Для задания маршрута патрулирования был создан вспомогательный класс патрульной точки:

на основе класса Actor создан Blueprint BP_PatrolPoint;

в качестве визуализации добавлен компонент StaticMesh (простой белый куб), чтобы расположение точек было наглядно видно в редакторе;

для исключения влияния патрульных точек на навигацию коллизия меша была отключена (Collision Presets = NoCollision), а сам актор используется только как маркер позиции.

Полученный Blueprint BP_PatrolPoint был несколько раз размещён на уровне таким образом, чтобы образовать замкнутый маршрут (рис. 2).



Рисунок 2. Размещение нескольких BP_PatrolPoint на уровне

3.3. Настройка персонажа-агента (BP_AICharacter)

В качестве ИИ-персонажа использовался Blueprint-класс BP_AICharacter, полученный дублированием стандартного персонажа проекта (шаблон Third Person). В нём были произведены следующие настройки:

- в Class Defaults для свойства AI Controller Class назначен класс BP_AIController (см. раздел 4);
- свойство Auto Possess AI установлено в значение Placed in World or Spawnd, что позволяет созданному на уровне экземпляру персонажа автоматически захватываться контроллером ИИ;
- для игрового экземпляра BP_AICharacter на уровне параметр Auto Possess Player отключён (Disabled), чтобы этот персонаж не управлялся игроком.

Таким образом, при запуске уровня движок автоматически создаёт и привязывает к BP_AICharacter экземпляр BP_AIController, который отвечает за логику патрулирования (рис. 3).

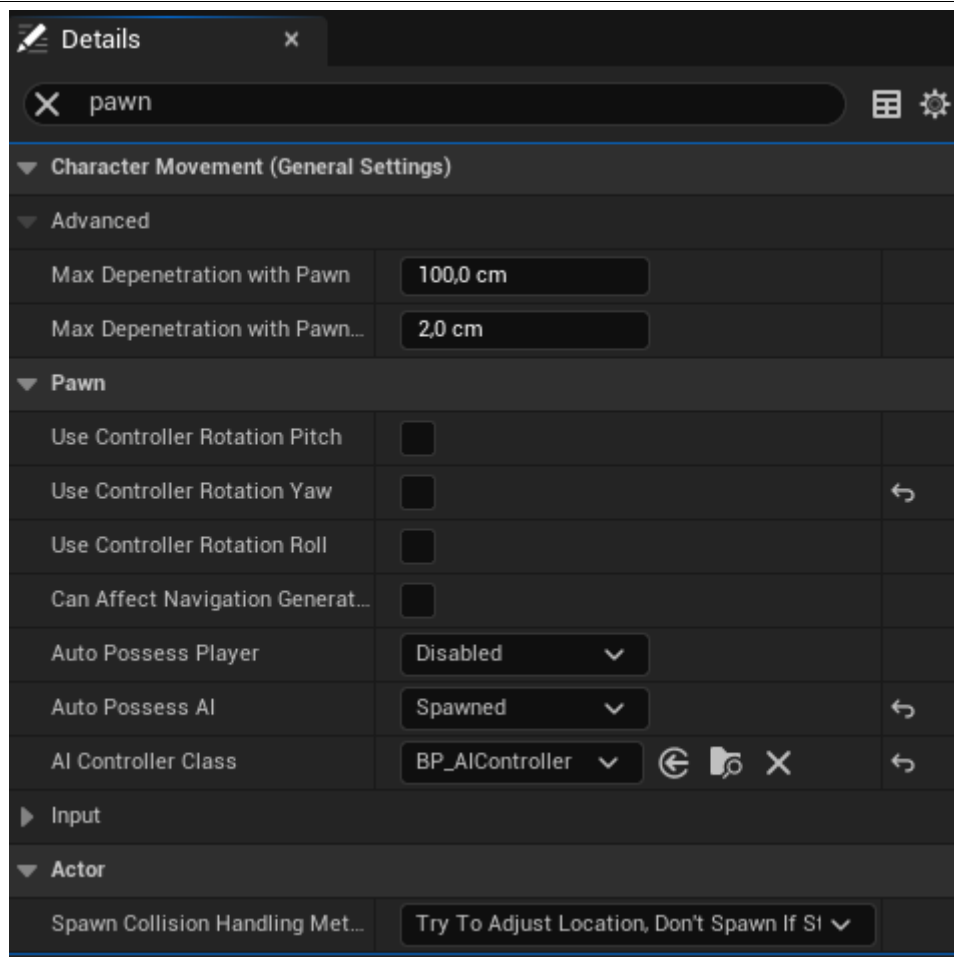


Рисунок 3. Настройки BP_AICharacter: AI Controller Class и Auto Possess AI

3.4. Структура контроллера ИИ (BP_AIController)

Вся логика патрулирования реализована внутри Blueprint-класса контроллера ИИ BP_AIController.

- В нём определены следующие переменные:
- PatrolPoints — массив (Array) типа Actor, содержащий ссылки на все патрульные точки (BP_PatrolPoint);
- CurrentPatrolIndex — целочисленная переменная (Integer), хранящая индекс текущей точки маршрута.

3.4.1. Сбор патрульных точек при запуске (Event BeginPlay)

При запуске игры (событие Event BeginPlay) контроллер выполняет инициализацию массива патрульных точек:

Сначала через Get All Actors Of Class ищутся все объекты типа BP_PatrolPoint на уровне.

Результирующий массив (Out Actors) построчно копируется в собственный массив контроллера PatrolPoints с помощью цикла ForEachLoop и узла Add.

После завершения цикла в переменную CurrentPatrolIndex записывается значение 0, то есть начальной считается первая точка маршрута.

В завершение вызывается пользовательское событие MoveToNextPatrolPoint, которое запускает движение агента к первой точке.

Структура этого фрагмента логики показана на рис. 4.

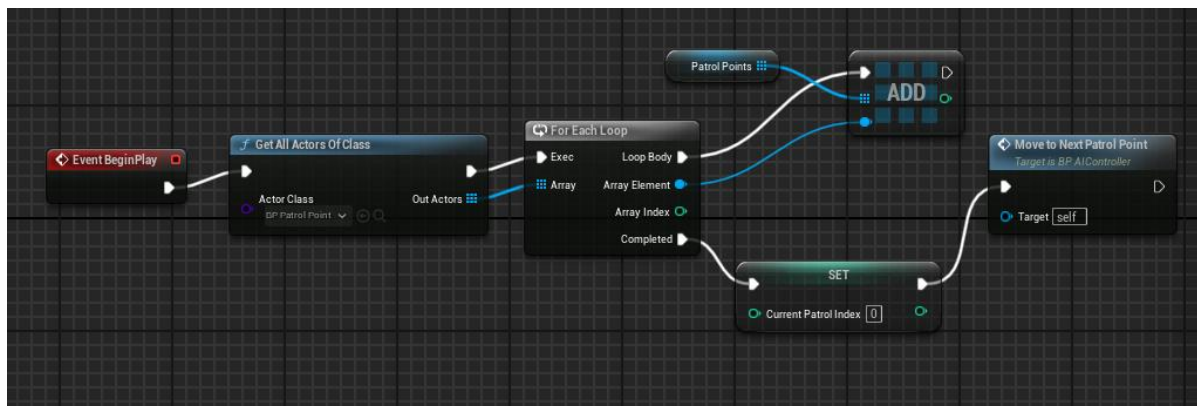


Рисунок 4. Blueprint-фрагмент Event BeginPlay в BP_AIController: поиск BP_PatrolPoint, заполнение массива PatrolPoints, инициализация индекса и вызов MoveToNextPatrolPoint

3.5. Логика движения между точками (событие MoveToNextPatrolPoint)

Пользовательское событие MoveToNextPatrolPoint реализует переход агента к текущей точке патруля и подготовку перехода к следующей.

Проверка наличия точек

В начале события выполняется проверка:

- вычисляется длина массива PatrolPoints (узел Length);
- если длина равна нулю, дальнейшая логика не выполняется (маршрут не задан);
- если длина массива больше нуля, продолжается выбор текущей точки.

Текущая патрульная точка определяется как:

- элемент массива PatrolPoints с индексом CurrentPatrolIndex (узел Get для массива);
- затем от выбранного актора вызывается GetActorLocation, получая мировые координаты точки.

Запуск перемещения с помощью AI MoveTo

Для перемещения персонажа к выбранной позиции используется стандартный узел AI MoveTo:

- в параметр Pawn подаётся результат функции Get Controlled Pawn контроллера (управляемый персонаж BP_AICharacter);
- в параметр Destination подаётся координата патрульной точки (вектор из GetActorLocation);
- радиус достижения цели (Acceptance Radius) задаётся, например, 50.0, чтобы агент не стремился попасть точно в геометрический центр точки.

При вызове AI MoveTo навигационная система Unreal Engine строит путь по NavMesh и начинает перемещение персонажа (рис. 5).

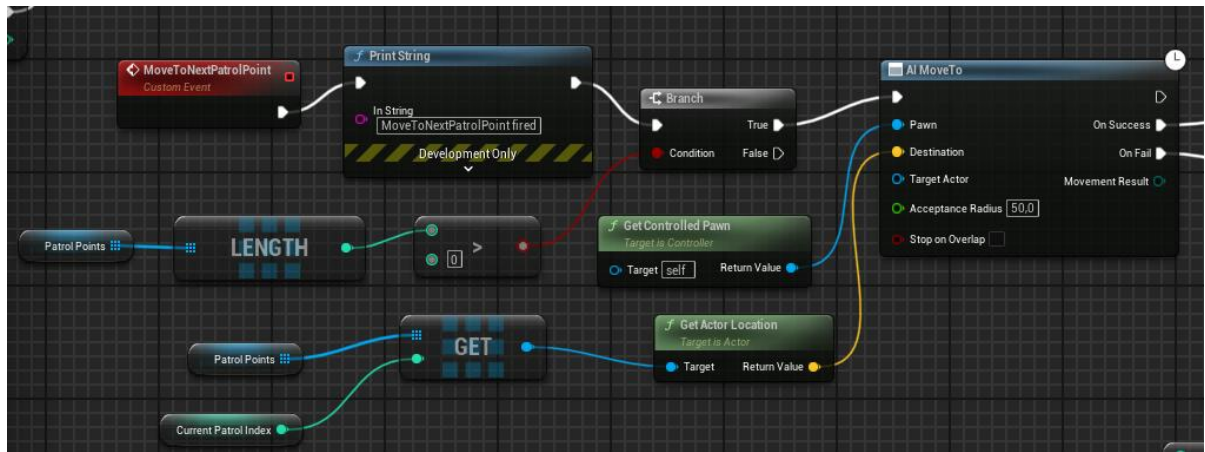


Рисунок 5. Blueprint-фрагмент MoveToNextPatrolPoint: выбор точки по индексу и вызов AI MoveTo

После успешного достижения цели AI MoveTo генерирует событие On Success.

1. В обработчике этого события выполняются следующие действия: Вычисляется следующий индекс:

$$\text{CurrentPatrolIndex}_{\text{new}} = (\text{CurrentPatrolIndex}_{\text{old}} + 1) \bmod N,$$

Где N — число патрульных точек (длина массива PatrolPoints). В реализации это выполняется через узлы сложения (+1) и целочисленного остатка от деления (Modulo) с делителем Length(PatrolPoints).

2. Результат записывается обратно в переменную CurrentPatrolIndex (Set CurrentPatrolIndex).

3. Сразу после этого рекурсивно вызывается событие MoveToNextPatrolPoint, что приводит к запуску нового AI MoveTo уже для следующей точки маршрута.

Благодаря использованию операции по модулю маршрутом получается замкнутая циклическая последовательность точек — после последней точки агент возвращается к первой (рис. 6).

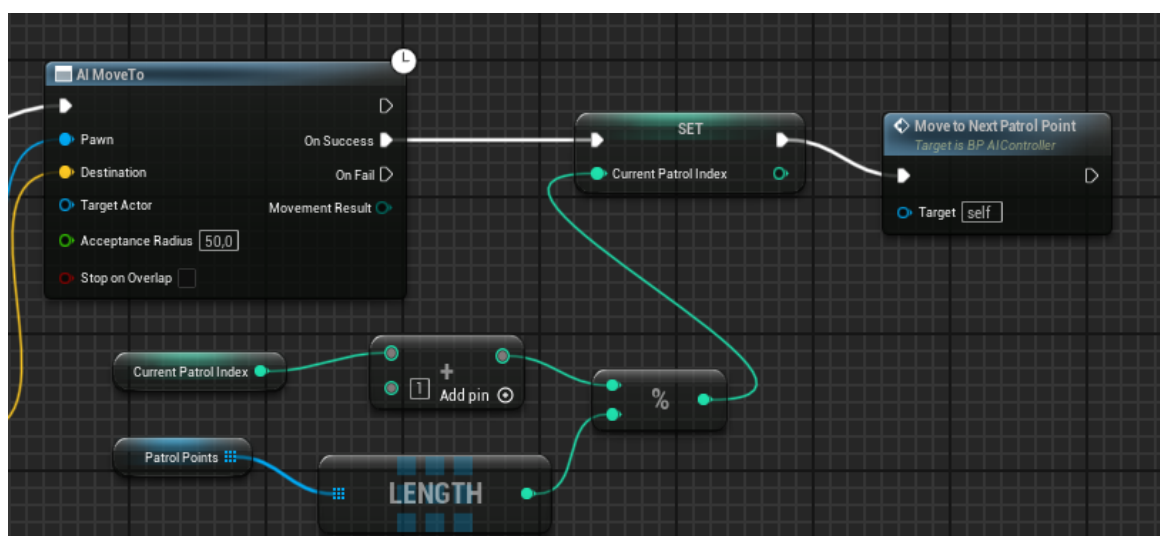


Рисунок 6. Blueprint-фрагмент обработки On Success: инкремент индекса по модулю длины массива и повторный вызов MoveToNextPatrolPoint)

6. Отладка и особенности реализации

В процессе отладки использовались вспомогательные экранные сообщения (Print String):

- после заполнения массива PatrolPoints выводилось его количество, что позволяло проверить корректность поиска всех BP_PatrolPoint на уровне;
- при входе в событие MoveToNextPatrolPoint выводился текущий индекс точки, что помогало убедиться в корректности обновления CurrentPatrolIndex.

В частности, была выявлена важная особенность:

- при использовании крупных кубов с включённой коллизией в качестве патрульных точек часть маршрута оказывалась недостижимой для ИИ, так как коллизия объектов вырезала участки NavMesh и препятствовала построению пути;
- после отключения коллизии у BP_PatrolPoint (Collision Presets = NoCollision) и расположения точек строго внутри зелёной области навигационной сетки (NavMesh) агент стал корректно проходить весь маршрут без остановок.

Этот момент можно рассматривать как частный практический результат: коллизия вспомогательных акторов, используемых только как маркеры маршрута, должна быть отключена, чтобы не ухудшать навигацию ИИ.

3.7. Результат работы патрульной системы

В итоге реализована простая, но функциональная система патрулирования:

- ИИ-персонаж BP_AICharacter, управляемый BP_AIController, автоматически находит все акторы BP_PatrolPoint на уровне;
- строится массив точек маршрута, индекс текущей точки хранится в переменной CurrentPatrolIndex;
- при запуске игры агент последовательно перемещается по всем точкам маршрута с помощью узла AI MoveTo, замыкая путь в цикл;
- изменение маршрута сводится к добавлению, удалению или перемещению акторов BP_PatrolPoint на уровне, без изменения логики Blueprint-контроллера.

На рис. 7 показан ИИ-персонаж в одном из состояний патрулирования, при прохождении очередной контрольной точки.

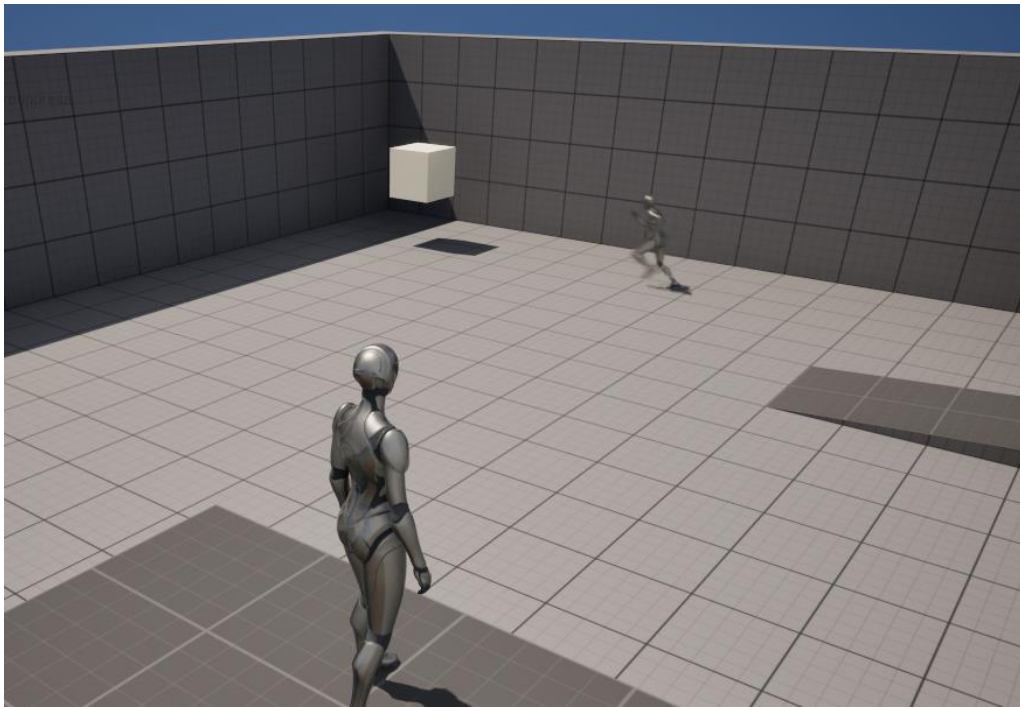


Рисунок 7. Скриншот игрового режима: ИИ-агент, движущийся к одной из патрульных точек маршрута

Библиографический список

1. Yannakakis G. N., Togelius J. Artificial intelligence and games. New York : Springer, 2018. Т. 2. С. 2475-1502.
2. Rabin S. (ed.). Game AI pro: collected wisdom of game AI professionals. CRC Press, 2013.
3. Zhang X., Zhang X. Based on Navmesh to implement AI intelligent pathfinding in three-dimensional maps in UE4 //Proceedings of the 2022 5th International Conference on Algorithms, Computing and Artificial Intelligence. 2022. С. 1-5.
4. Gupta I., Dhammi R., Kaur P. Overview of Design and Development of a Third-Person Shooter Maze Escape Game Using Unreal Engine //International Journal of Scientific Research and Technology. 2025. Т.2. №5.
5. Раджапов Т. Р. Ігровий застосунок в жанрі roguelike на рушії Unreal Engine 5. Розробка ігрового середовища. 2025.