

ТЕХНИЧЕСКИЕ НАУКИ



УДК 004.8:004.9

Разработка и оптимизация агентов искусственного интеллекта в мобильных пошаговых стратегиях на базе Unity 3D: алгоритмы радиального зонирования

В.С. Токарев, Д.О. Гуковский

Донской государственный технический университет, г. Ростов-на-Дону, Российская Федерация

Аннотация

В статье рассматривается задача оптимизации интеллектуальных агентов в мобильных играх, обусловленная ограниченными вычислительными ресурсами и автономным питанием устройств. Показано, что регулярный перерасчет маршрутов при использовании классических алгоритмов поиска пути, включая A* и алгоритм Дейкстры, может создавать избыточную нагрузку на процессор и снижать производительность приложения. Предложен гибридный подход, сочетающий традиционные методы поиска пути с авторской системой радиального зонирования. Метод основан на формировании вокруг игрока невидимых концентрических зон, при пересечении которых противником выполняется расчет оптимального маршрута. Решение реализовано в Unity 3D с применением паттерна «пул объектов». Экспериментальные результаты показали четырехкратное сокращение среднего времени расчета пути, что подтверждает практическую эффективность предложенного алгоритма для мобильных игровых приложений.

Ключевые слова: Unity 3D, оптимизация мобильных игр, искусственный интеллект, поиск пути, пошаговая стратегия, радиальное зонирование, эвристические алгоритмы

Для цитирования. Токарев В.С., Гуковский Д.О. Разработка и оптимизация агентов искусственного интеллекта в мобильных пошаговых стратегиях на базе Unity 3D: алгоритмы радиального зонирования. *Молодой исследователь Дона*. 2026;11(5):10–15.

Development and Optimization of AI Agents in Mobile Turn-Based Strategy Games Based on Unity 3D: Radial Zoning Algorithms

Viktor S. Tokarev, Denis O. Gukovsky

Don State Technical University, Rostov-on-Don, Russian Federation

Abstract

The article studies the problem of optimizing the intelligent agents in mobile games caused by the limited computing resources and self-power capacity of devices. It has been ascertained that regular path recalculation in common pathfinding algorithms, such as A* and Dijkstra's algorithms, can impose an excessive load on CPU and reduce performance of an application. A hybrid approach has been proposed, which combines common pathfinding algorithms with the author's proprietary technique of radial zoning. The technique is based on the formation of invisible concentric zones around a player that, once intruded by an enemy, ensure optimal path computation. The solution was implemented in Unity 3D game engine using the "object pool" pattern. Experiment resulted in a fourfold reduction of the mean pathfinding computation time, which confirms the practical efficiency of the proposed algorithm for mobile gaming applications.

Keywords: Unity 3D, optimization of the mobile games, artificial intelligence, pathfinding, turn-based strategy, radial zoning, heuristic algorithms

For Citation. Tokarev VS, Gukovsky DO. Development and Optimization of AI Agents in Mobile Turn-Based Strategy Games Based on Unity 3D: Radial Zoning Algorithms. *Young Researcher of Don*. 2026;11(5):10–15.

Введение. Рынок мобильных игр в настоящее время является крупнейшим сегментом глобальной игровой индустрии. Количество пользователей мобильных игр превышает 2,5 миллиарда человек по всему миру [1]. Жанры стратегий и головоломок входят в топ-5 наиболее популярных категорий среди приложений, предназначенных для скачивания игр и утилит. С учетом перечисленных факторов игроки ожидают нестандартных и содержательно интересных решений, однако на мобильных платформах масштаб проекта практически всегда ограничивается производительностью устройства и уровнем энергопотребления телефона.

Классические алгоритмы поиска пути, широко применяемые в коммерческой разработке игр, при периодическом пересчете маршрутов приводят к избыточному энергопотреблению [2]. В пошаговых играх, таких как «Шахматы», искусственный интеллект должен принимать решение на каждом ходу, и использование неоптимизированных алгоритмов с высокой вероятностью приводит к снижению частоты кадров в секунду и чрезмерным затратам энергии [3]. Особенно выраженной данная проблема является в играх, где игровое поле представляет собой открытую местность со строгим алгоритмом передвижения, поскольку в таких проектах каждый ход требует повторного расчета узлов маршрута.

Анализ успешных игровых проектов показывает, что их привлекательность во многом определяется художественной концепцией, которая способна компенсировать ограниченное количество ключевых механик. В данной статье авторский алгоритм поиска пути рассматривается на примере самостоятельно разработанной пошаговой стратегии «Behind Cells».

Цель данной работы — демонстрация работоспособности авторского алгоритма поиска пути и оптимизация методов взаимодействия с интеллектуальными агентами на примере игры собственной разработки «Behind Cells». В исследовании рассматривается система, использующая гибридный подход к работе с интеллектуальными агентами, который сочетает классический поиск пути и авторский метод «система радиального зонирования».

Задачи исследования

В задачи исследования предметной области входит: формулирование базовых игровых механик игры «Behind Cells», проектирование гибридной системы работы с интеллектуальными агентами, позволяющей агенту строить локальный маршрут до искомого объекта, а не до границ всей карты, а также реализация рабочего прототипа в игровом движке Unity 3D с применением методов оптимизации. В процессе оптимизации необходимо реализовать паттерн «пул объектов», изолировать вычисления от метода, срабатывающего каждый компьютерный тик, — «Update», а также обеспечить хранение данных в одномерных массивах «value types». Кроме того, требуется провести сравнительный анализ производительности разработанного алгоритма с классическим A* во внутреннем тестировщике производительности Unity Profiler.

Материалы и методы

Концепция игры «Behind Cells»

Для понимания сущности игры и роли искусственного интеллекта в ее структуре необходимо кратко рассмотреть концепцию и сюжет проекта. Сюжет игры основан на похищении главного героя коварным волшебником, являющимся заядлым любителем головоломок. Проиграв волшебнику, герой попадает в магическое подземелье, построенное по принципу шахматной доски. Игроку предстоит выбраться из подземелья, минуя стража, который на каждом уровне будет пытаться дойти до игрока и одолеть его. Именно страж выступает в роли интеллектуального агента: на каждый ход игрока он совершает собственное действие, рассчитывая оптимальный путь до местоположения героя и пытаясь настичь его.

Игровое поле представляет собой прямоугольную сетку, в которой в самом простом случае игрок начинает движение из левого верхнего угла, а стражник, охраняющий выход, располагается в правом нижнем углу.

Игровой цикл разделен на фазу обдумывания, в течение которой игрок выбирает клетку для перемещения, и фазу действия, в рамках которой непосредственно осуществляется передвижение игрока и противника.

Ключевыми сущностями на игровом поле являются страж, сложность поведения которого измеряется глубиной просчета допустимых ходов, рядовые элементы, среди которых ловушки и иные игровые противники, а также люк, представляющий собой переход на следующий уровень.

Визуальный стиль игры выполнен в жанре «пиксель-арт», интерфейс имеет минималистичное оформление, а вид на игрока реализован от третьего лица таким образом, чтобы создавать впечатление ортогональности. Игра средствами 2D-графики формирует ощущение объема — данный прием в разработке игр называется 2,5D-графикой.

Архитектура искусственного интеллекта

В процессе реализации искусственного интеллекта стража была разработана указанная выше гибридная модель, сочетающая классический поиск пути и авторский эвристический подход «система радиального зонирования». В пространстве вокруг целевого объекта формируются невидимые концентрические кольца, при попадании в которые ИИ-агент начинает строить маршрут до игрока более эффективно.

Алгоритм работы интеллектуального агента можно разделить на несколько этапов:

- сканирование территории: агент определяет, в каком кольце он находится относительно игрока. Полученная информация о местоположении противника относительно игрока выступает первичным фильтром для отсеивания заведомо ошибочных векторов движения;
- анализ препятствий: на поле присутствуют преграды, не позволяющие проходить через них, поэтому после сканирования необходимо также запомнить, в каких клетках находятся непроходимые препятствия;
- расчет маршрута: вместо построения полного пути от своей клетки до клетки игрока агент формирует локальный маршрут до ближайшего радиуса, выпущенного относительно позиции игрока, что позволяет снизить экспоненциальный рост вычислений, присущий стандартным алгоритмам, перечисленным выше.

На рис. 1 приведена схема, демонстрирующая принцип работы алгоритма с визуальной точки зрения: R1 — зона захвата, R2 — зона сближения, R3 — зона обнаружения. Безусловно, функцию поиска игрока противником выполняют все кольца, однако на схеме для каждого из них указано отдельное название, позволяющее понять, где противник обнаруживает игрока, где сближается с ним, а где атакует. Интеллектуальный агент в зависимости от уровня сложности выбирает вектор движения, минимизирующий попадание в зоны с ловушками и зоны, которые расположены дальше желаемого радиуса и не являются оптимальными.

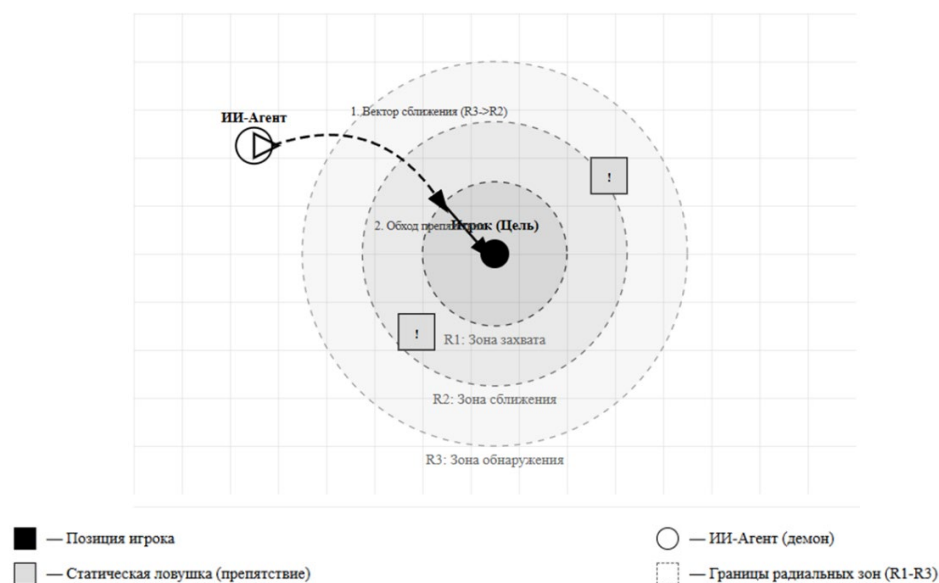


Рис. 1. Схема реализации гибридной системы управления интеллектуальными агентами: показан процесс иерархического выбора вектора движения с учетом радиальных зон

Техническая реализация и оптимизация

Игра разрабатывается на игровом движке Unity, что обеспечивает ее кроссплатформенность. Для повышения производительности игрового приложения были реализованы следующие меры:

- использование паттерна «пул объектов» для расчета узлов графа и построения невидимых колец вокруг игрока, что исключает частое создание и удаление игровых объектов `GameObject`, к числу которых относятся и указанные кольца [4, 5];
- изоляция логики искусственного интеллекта в отдельный менеджер, функционирующий только в фазу обдумывания;
- хранение данных о ловушках и радиусах, в пределах которых пересекаются луч, выпущенный из местоположения игрока, и противник, в одномерных массивах. Данный подход позволяет минимизировать количество итераций работы сборщика мусора `Garbage Collector`, каждый вызов которого отрицательно влияет на производительность;
- исключение тяжелых методов поиска игровых объектов на сцене, в том числе метода `FindObjectsOfType`.

Сложность игры против интеллектуального агента масштабируется за счет глубины просчета ходов. Это обеспечивается количеством колец, построенных от местоположения игрока, а также их плотностью, то есть расстоянием между кольцами относительно друг друга.

Разработанная архитектура позволила существенно сократить энергозатраты на фазе обдумывания. В тех случаях, когда обычно применялся бы тяжеловесный алгоритм поиска пути, в предложенной системе используются математические операции, формирующие кольца вокруг игрока с постепенным увеличением их радиуса, а также методы фиксации коллизий интеллектуального агента с одним из таких колец. При обнаружении соответствующего пересечения инициируется процесс расчета расстояния между игроком и противником.

Результаты исследования. Для сравнения с классическим алгоритмом A* [6] в среде Unity Profiler было проведено профилирование вычислительной нагрузки, направленное на оценку эффективности разработанной гибридной системы радиального зонирования. Тестирование выполнялось на мобильном устройстве среднего ценового сегмента при сборке проекта с компилятором IL2CPP. Тестовая конфигурация включала сетку 6×12 клеток и 5 статических ловушек.

При программной реализации ИИ-агента, представленного демоном, был применен ряд архитектурных ограничений [7], соответствующих современным практикам оптимизации высоконагруженных мобильных проектов на Unity [8]:

- паттерн Object Pool использовался для расчетов пути, что позволило избежать ресурсоемких операций реинициализации и уничтожения узлов графа в памяти во время игры;
- вычислительная логика поиска пути была полностью изолирована от покадрового метода Update [9] и делегирована собственному менеджеру управления состояниями State Manager, активируемому исключительно в фазу обдумывания;
- из алгоритма полностью исключены ресурсоемкие методы поиска, например FindObjectsOfType [10], а также операции боксинга и анбоксинга в C#. Это позволило установить ручной контроль над работой сборщика мусора и исключить микрофризы, возникающие при динамическом выделении памяти во время хода противника.

Таблица 1

Сравнительный анализ времени расчёта пути ИИ-агентом с помощью Unity Profiler

Параметр оценки	Классический алгоритм A*	Гибридная система ИИ
Среднее время расчёта пути в миллисекундах	1.85 мс	0.42 мс
Пиковая нагрузка при перестроении в миллисекундах	3.10 мс	0.85 мс
Количество обрабатываемых узлов в среднем	48 узла	12 узлов в пределах 2-х колец, исходящих от игрока
Выделение памяти «GC Alloc» (сколько новой памяти выделяется под вычисления, иницилируя работу «Garbage collector»)	~2.4 КБ	0 КБ (при переиспользовании игровых объектов, среди которых концентрические кольца, при реализации паттерна «Пул объектов»)

В таблице 1 представлены сравнительные метрики времени выполнения скриптов, выраженные в миллисекундах на кадр расчета, для двух алгоритмов на сетке 6×12 клеток при наличии 5 статических препятствий, представленных ловушками.

Полученные данные отчетливо демонстрируют, что внедрение системы радиального зонирования позволило сократить среднее время расчета маршрута в 4 раза: с 1,85 мс до 0,42 мс. Снижение пиковой нагрузки объясняется тем, что радиальный алгоритм выполняет поиск исключительно в пределах локальных концентрических зон, так называемых колец, формируемых вокруг игрока, тогда как недостижимые сегменты графа исключаются из обработки. Таким образом, за счет контролируемого управления памятью удалось минимизировать количество вызовов сборщика мусора Garbage Collector и повысить общую производительность системы.

На рис. 2 приведен скриншот контекстного интерфейса, карты, расположения игрока, стража и ловушек на клеточном поле в разработанной игре «Behind Cells».



Рис. 2. Игровой процесс «Behind Cells»: игрок (слева сверху) и страж (справа внизу) на клеточном поле

Обсуждение. Двухфазная система, предложенная в разработанной игре «Behind Cells», в сочетании с радиальным поиском решает значимую техническую задачу. Поскольку расчеты ИИ-агента выполняются только в специально выделенную фазу обдумывания, а не в каждом кадре Update, процессорное время мобильного устройства высвобождается для рендеринга сложной 2.5D-графики, включающей пиксель-арт и объемные объекты.

По сравнению с классическим рейкастингом [11] или использованием ресурсоемкого Unity NavMesh [12], которые являются избыточными для клеточного поля, данный подход демонстрирует высокую эффективность. Радиальная эвристика позволяет алгоритму поиска пути в большинстве штатных ситуаций работать со сложностью $O(N)$, где N — количество клеток внутри активного кольца, а не площадь всей карты.

Гибкость алгоритма обеспечивает возможность его адаптации к внедрению новых игровых объектов и механик, например: добавлению новых типов препятствий, изменению поведения стража в зависимости от уровня сложности. Все перечисленные изменения реализуются без переработки базовой архитектуры системы. Полученные результаты подтверждают эффективность и практическую применимость предложенного подхода к созданию производительных мобильных игр с интеллектуальными агентами.

Заключение. В ходе исследования была создана система поиска пути, основанная на гибридном подходе к работе с интеллектуальным агентом.

К ключевым результатам исследования относятся:

- спроектирована двухфазная система игрового цикла, в которой точно заданы временные интервалы работы интеллектуального агента;
- разработана новая концепция взаимодействия с интеллектуальным агентом;
- сформирована техническая база, позволяющая масштабировать игру в направлении многопользовательского режима при наличии нескольких игроков на поле;
- экспериментально подтверждено четырехкратное снижение времени расчета пути по сравнению с классическими эвристическими методами.

Созданное решение предлагает авторский способ разработки игр, оптимизированных для мобильных устройств: оно позволяет сохранить производительность и обеспечить необходимый баланс между сложностью игрового процесса и низким энергопотреблением.

Список литературы

1. Newzoo. *Global Mobile Market Report 2023*. URL: <https://resources.newzoo.com/global-mobile-market-report> (дата обращения: 10.03.2025).
2. Hart PE, Nilsson NJ, Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*. 1968;4(2):100–107. <https://doi.org/10.1109/TSSC.1968.300136>
3. Millington I, Funge J. *Artificial Intelligence for Games*. 3rd Ed. CRC Press; 2019. 1010 p.
4. Tulleken H. *Deep Dive: The Design and Implementation of Object Pools*. *Game Developer*. 2024. URL: <https://www.gamedeveloper.com/programming/deep-dive-the-design-and-implementation-of-object-pools> (дата обращения: 20.03.2026).
5. Ellison G. Boost Unity Performance with Object Pooling. Wayline, 2025. URL: <https://www.wayline.io/blog/unity-object-pooling-performance> (дата обращения: 20.03.2026)
6. Dill K. Spatial Reasoning for Strategic Decision Making. In book: *Game AI Pro 360: Guide to Tactics and Strategy*. CRC Press; 2019. P. 211–220. <https://doi.org/10.1201/9780429054969>
7. Rabin S (Ed). *Game AI Pro 360: Guide to Movement and Pathfinding*. CRC Press; 2019. 320 p. <https://doi.org/10.1201/9780429055096>
8. Rabin S (Ed). *Game AI Pro 2: Collected Wisdom of Game AI Professionals*. CRC Press; 2015. 578 p. <https://doi.org/10.1201/b18373>
9. Cossu S. *Beginning Game AI with Unity: Programming Artificial Intelligence*. Apress Berkeley; 2021. 143 p. <https://doi.org/10.1007/978-1-4842-6355-6>
10. Palacios J. *Unity 2018 Artificial Intelligence Cookbook*. Packt Publishing; 2018. 334 p.
11. Stanescu M, Bethel CL. A Comparison of Grid-Based Pathfinding Techniques for Video Games. In: *Proceedings of the 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*. Volume 2; 2015. P. 453–458.
12. Juliani A, Berges VP, Teng E, Cohen A, Harper J, Elion C, et al. Unity as a Platform for Intelligent Agents. *arXiv*. 2020;arXiv:1809.02627v2

Об авторах:

Виктор Сергеевич Токарев, студент Института сквозных технологий Донского государственного технического университета (344003, Российская Федерация, г. Ростов-на-Дону, пл. Гагарина, 1), victor.tokareff@gmail.com

Денис Олегович Гуковский, студент Института сквозных технологий Донского государственного технического университета (344003, Российская Федерация, г. Ростов-на-Дону, пл. Гагарина, 1), denisguk25032004@gmail.com

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the Authors:

Viktor S. Tokarev, Student at the Institute of End-to-End Technologies, Don State Technical University (1, Gagarin Sq., Rostov-on-Don, 344003, Russian Federation), victor.tokareff@gmail.com

Denis O. Gukovsky, Student at the Institute of End-to-End Technologies, Don State Technical University (1, Gagarin Sq., Rostov-on-Don, 344003, Russian Federation), denisguk25032004@gmail.com

Conflict of Interest Statement: the authors declare no conflict of interest.

All authors have read and approved the final manuscript.