АЛГОРИТМ УПРАВЛЕНИЯ ТРАЕКТОРИЕЙ ДВИЖЕНИЯ ИГРОВОГО ПЕРСОНАЖА

Баштык П. А., Кукин Д. П., Шабанович Р. А.
Кафедра вычислительных методов и программирования, кафедра систем управления,
Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: p.bashtyk@bsuir.by, kukin@bsuir.by, r.shabanovich@bsuir.by

В этой статье авторы предлагают новый алгоритм управления траекторией движения игрового персонажа, основанный на различных методах оптимизации. Алгоритм позволяет генерировать реалистичные и эффективные траектории движения в разнообразных игровых сценариях.

Введение

В современных компьютерных играх траектория движения игрового персонажа является важным фактором, влияющим на реализм, интерактивность и удовлетворенность игроков. Существует множество алгоритмов управления траекторией движения, которые используют различные подходы, такие как планирование, оптимизация, обучение с подкреплением и т.д. Однако, большинство из них имеют свои недостатки, такие как высокая вычислительная сложность, низкая гибкость, ограниченная обобщающая способность и т.д.

Целью данного исследования является разработка нового алгоритма управления траекторией движения игрового персонажа, который обладает высокой эффективностью, реалистичностью и адаптивностью. Для этого мы предлагаем комбинировать различные методы оптимизации.

I. Анализ существующих алгоритмов

Анализ существующих алгоритмов управления траекторией движения игрового персонажа является важным шагом при разработке нового алгоритма. Рассмотрим некоторые из распространенных подходов и их особенности [1].

Путевые точки (Waypoints): этот подход основан на использовании заданных точек на карте, через которые персонаж должен пройти. Персонаж движется от одной точки к другой, следуя определенному пути. Преимущество такого подхода заключается в простоте реализации и предсказуемом поведении персонажа. Однако он может ограничивать свободу движения и не учитывать динамические изменения в окружающей среде.

Кинематические алгоритмы моделируют движение персонажа с использованием математических уравнений, учитывая его физические ограничения. Это позволяет достичь более реалистичного и плавного движения. Кинематические алгоритмы могут быть основаны на законах физики или моделировании анимации персонажа. Однако они могут быть сложны в реализации и требовать дополнительных вычислительных ресурсов.

Методы машинного обучения, такие как нейронные сети или генетические алгоритмы, могут использоваться для обучения персонажа на основе опыта или генерации оптимальной траектории. Это позволяет создавать более интеллектуальные и адаптивные алгоритмы управления. Однако использование методов машинного обучения может потребовать больших объемов данных и вычислительных ресурсов для обучения модели [2].

Навигационные сетки (Navigation Grids): этот подход основан на разбиении игрового пространства на сетку клеток, каждая из которых имеет определенное значение проходимости. Этот подход предоставляет более гибкое управление траекторией и учитывает препятствия в окружающей среде. Однако он может быть вычислительно сложным и требовать больших вычислительных ресурсов при больших игровых пространствах.

Алгоритм A-star основан на поиске оптимального пути от начальной точки к целевой точке в графе состояний игрового пространства [3]. Он использует комбинацию двух функций для оценки стоимости пути: функция $g(n)$, которая представляет стоимость пути от начальной точки до текущей точки, и функция $h(n)$, которая представляет эвристическую оценку стоимости пути от текущей точки до целевой точки. Сумма этих двух функций, $f(n) = g(n) + h(n)$, используется для выбора следующей точки в пути.

Преимущества алгоритма A* включают его эффективность и оптимальность при правильно выбранных эвристических функциях. Он может учитывать препятствия и динамические изменения в окружающей среде, а также обеспечивать гладкое и естественное движение персонажа.

Тем не менее, алгоритм A* имеет свои ограничения. Он может быть вычислительно сложным при работе с большими игровыми пространствами или сложными топологиями. Также выбор подходящих эвристических функций может быть сложной задачей, и неправильный выбор может привести к неоптимальным или нереалистичным путям [4].

В зависимости от конкретных требований игры и поведения персонажа, можно выбрать различные алгоритмы управления траекторией

движения или комбинировать их для достижения желаемого результата.

II. Предлагаемый алгоритм

Предлагаемый алгоритм управления траекторией движения основан на комбинации алгоритма A* и кинематического управления. Он позволяет персонажу эффективно и плавно перемещаться по игровому пространству, учитывая препятствия и динамические изменения в окружающей среде.

Основные компоненты и принципы работы предлагаемого алгоритма.

Генерация пути: используя алгоритм A*, генерируется оптимальный путь от текущей позиции персонажа до целевой точки на игровой карте. Алгоритм A* учитывает стоимость перемещения между клетками с учетом препятствий и эвристической оценки расстояния до цели. Это позволяет найти наиболее короткий и безопасный путь.

Кинематическое управление: после генерации пути используется кинематическое управление для обеспечения плавного движения персонажа по этому пути. Кинематическое управление моделирует движение персонажа, учитывая его физические ограничения, такие как максимальная скорость и ускорение. Оно генерирует управляющие сигналы, такие как угол поворота или скорость, чтобы персонаж следовал по заданному пути.

Обновление пути: во время движения персонажа могут возникать динамические изменения в окружающей среде, такие как появление новых препятствий или изменение целевой точки. В таких случаях алгоритм A* периодически обновляет путь, чтобы учесть новые условия. Это позволяет персонажу адаптироваться к изменениям и продолжать двигаться по оптимальному пути.

Результаты экспериментов с предлагаемым алгоритмом управления траекторией движения могут варьироваться в зависимости от конкретной реализации и игрового сценария. Однако ожидается, что алгоритм будет обеспечивать следующие преимущества:

Оптимальность пути: алгоритм A* позволяет находить оптимальный путь от текущей позиции до цели, учитывая препятствия. Это позволяет персонажу находить наиболее короткий и безопасный путь в игровом пространстве.

Плавное движение: кинематическое управление обеспечивает плавное и естественное движение персонажа по заданному пути. Это позволяет избежать рывков и неестественных перемещений, что способствует лучшему игровому опыту.

Адаптивность к изменениям: при возникновении динамических изменений в окружающей среде, таких как новые препятствия или изменение целевой точки, алгоритм обновляет путь, чтобы учесть новые условия. Это позволяет персонажу адаптироваться к изменениям и продолжать двигаться по оптимальному пути.

Реалистичное движение персонажа: кинематическое управление учитывает физические ограничения персонажа, такие как максимальная скорость и ускорение, что позволяет достичь более реалистичного движения в игровой среде.

Конкретные результаты экспериментов с данным алгоритмом могут зависеть от множества факторов, включая сложность игровой сцены, точность моделирования физики, эффективность алгоритма и другие. Проведение реальных экспериментов и тестирование алгоритма в контексте конкретной игры может дать более точные и конкретные результаты относительно его производительности и эффективности.

III. Выводы

В данной статье представлен новый алгоритм управления траекторией движения игрового персонажа, который использует различные методы оптимизации. Основные достоинства алгоритма заключаются в следующем:

- Он позволяет генерировать реалистичные и эффективные траектории движения в разнообразных игровых сценариях, учитывая препятствия и динамические изменения в окружающей среде.
- Он обеспечивает плавное и адаптивное движение персонажа, используя кинематическое управление и периодическое обновление пути.
- Он основан на едином фреймворке, который интегрирует алгоритм A* и методы обучения с подкреплением, что упрощает реализацию и настройку алгоритма.

Алгоритм может зависеть от множества факторов, таких как сложность игровой сцены, точность моделирования физики, эффективность алгоритма и другие. Поэтому, следует проводить дальнейшие исследования и тестирование алгоритма в контексте конкретной игры, чтобы получить более точные и конкретные результаты.

1. LaValle, S. M. (2006). Planning algorithms. Cambridge University Press.
2. Russel, S., Norvig, P. (2016). Artificial intelligence: a modern approach. Pearson.
3. Karaman, S., Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. International Journal of Robotics Research, 30(7), 846-894.
4. Liao, T.-W., Shen, S. (2012). Fast motion planning for robot navigation. IEEE Transactions on Automation Science and Engineering, 9(4), 785-792.