ОСОБЕННОСТИ ПРИМЕНЕНИЯ АЛГОРИТМОВ ГРАФОВОГО ПОИСКА ПУТЕЙ В РАЗРАБОТКЕ ТРЕХМЕРНЫХ ИГРОВЫХ МИРОВ

Рязанцев Д. Д., Рязанцев Н. Д.
Кафедра вычислительных методов и программирования,
Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: {d.riazantsev, n.riazantsev}@bsuir.by

В этой статье рассматриваются особенности применения алгоритмов графового поиска путей при разработке трехмерных игровых миров. Исследуется различные структуры данных, включая высотные карты, иерархические графы и воксельные сетки, которые помогают оптимизировать поиск пути и управление сложностью в игровых сценах. Рассмотрены преимущества и недостатки каждого метода, а также примеры их использования и интеграции для создания динамичных и увлекательных трехмерных игровых миров.

Введение

В сфере создания видеоигр сложно переоценить значимость реалистичных, увлекательных и полностью погружающих игровых вселенных. Одним из важнейших элементов разработки таких миров являются алгоритмы поиска пути на графах, которые позволяют персонажам и объектам находить оптимальные пути в сложных трехмерных средах. Существует множество методов реализации алгоритмов графового поиска в 3D-пространствах, каждая из которых обладает своими достоинствами и недостатками.

I. Основная проблема внедрения алгоритмов графового поиска в 3D-пространстве

Ключевая сложность внедрения алгоритмов графового поиска в 3D-пространстве заключается в экспоненциальном росте вычислительных затрат и объема данных. В отличие от 2D-пространства, где путь определяется двумя координатами, в 3D-пространстве добавляется третье измерение, что существенно усложняет расчеты и объем хранимой информации.

Специфические трудности:

- Размер пространства поиска: В 3D-пространстве количество допустимых маршрутов значительно увеличивается, что требует большего количества вычислительных ресурсов и памяти;
- Топология и навигация: В 3D-пространстве объекты и препятствия могут располагаться более сложными и непредсказуемыми способами. Это требует более сложных алгоритмов для определения доступности маршрутов;
- Эвристики и оценки: Использование эвристик для определения расстояний становится более сложным, так как нужно учитывать расстояния в трех измерениях.
- Визуализация и отладка: Визуализация и отладка алгоритмов маршрутизации в 3D-пространстве осложнены, так как необходимо принимать во внимание дополнительные координаты и взаимодействия между ними.

Основными способами решения этих проблем являются использование: воксельных сеток, высотных карт, иерархических графов. Эти подходы обеспечивают оптимальное сочетание производительности и точности, что делает их наиболее актуальными в создании современных трехмерных игровых пространств.

II. Воксельные структуры

Воксельные структуры представляют собой трёхмерные аналоги пикселей, где каждый воксель (объемный пиксель) представляет собой крошечный куб в пространстве, графически пример отображения вокселя. Они используются для деления 3D-пространства на отдельные единицы, что упрощает его визуализацию и обработку.

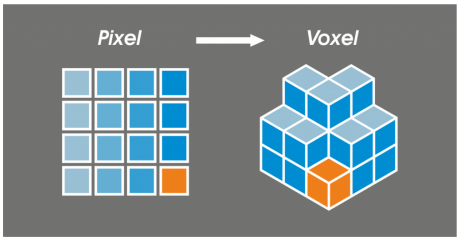

Рис. 1 – Графическое отображение вокселя в сравнении с пикселем

Моделирование игрового 3D-пространства через воксельную структуру — это превосходный метод для реализации графовых алгоритмов поиска пути, особенно в играх с изменяющимся окружением или подверженными разрушению объектами. Изобразить трёхмерное пространство в виде воксельной структуры можно следующим образом:

1. Деление пространства на воксели: Игровое пространство разбивается на воксели, так что каждый воксель имеет координаты в 3D-пространстве (x, y, z);

2. Инициализация массива: создается 3D-массив вокселей, где каждая ячейка содержит информацию о состоянии вокселя (доступный, недоступный, изменяемый и т. д.);
3. Создание связей между вокселями: Каждый воксель связывается с соседними вокселями (всего может быть до 26 соседей в 3D-пространстве). Эти связи будут представлять собой графовые рёбра;
4. Использование графовых алгоритмов: В большинстве случаев применяются весовые алгоритмы поиска пути, так как учитывается, что каждый воксель может иметь свои собственные веса переходов, зависимые от сложности прохождения или типов поверхностей;
5. Динамические обновления: Воксельная структура прекрасно подходит для игр с изменяющимся окружением. При изменении состояния вокселя (например, разрушение стены или появление препятствия), обновляется информация о состоянии вокселя и его связях.

Воксельные сетки предъявляют высокие требования к памяти и вычислительным ресурсам, но обладают способностью точно моделировать сложные структуры и идеально подходят для работы с динамическими и разрушимыми объектами. Эти сетки требуют применения алгоритмов сжатия и оптимизации памяти, а также могут использовать техники LOD для вокселей и хорошо согласуются с GPU для ускорения вычислений. Использование воксельных сеток целесообразно в играх с разрушающимися или изменяющимися средами, где необходима высокая детализация и точное изображение мира, а также в играх с многоуровневыми и сложными 3D-структурами.

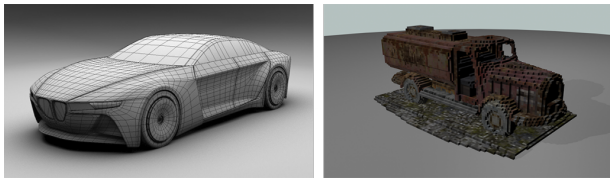

Рис. 2 – Различия в детализации объектов при использовании вокселей и пикселей

Так же применение воксельной структуры в игровом мире накладывает определенные сложности при создании трехмерных моделей. Для достижения высокой детализации объектов требуется большее количество вокселей по сравнению с пикселями,пример сравнения приведен на рисунке 2. Это связано с тем, что воксели являются объемными элементами, и каждый из них должен учитывать детали в трех измерениях, что увеличивает объем данных и сложность обработки. Кроме того, это требует большей вычислительной мощности и памяти для рендеринга и хранения воксельных моделей, а также оптимизации для обеспечения плавности игрового процесса. Поэтому разработчики должны тщательно балансировать между необходимой детализацией и доступными ресурсами.

Использование воксельных структур предоставляет гибкий и эффективный способ управления движением персонажей и объектов в трехмерном пространстве, особенно в условиях сложных и динамично изменяющихся игровых миров. Тем не менее, данное решение связано с рядом ограничений. Прежде всего, для достижения высокой степени детализации требуется значительное количество вокселей, что вызывает увеличение объема данных и потребность в больших вычислительных ресурсах. Во-вторых, рендеринг и обработка воксельных моделей могут быть более сложными и медленными по сравнению с традиционными методами. В-третьих, высокие требования к памяти и производительности могут ограничить возможности для сложных и обширных сцен, заставляя разработчиков оптимизировать и находить компромиссы между детализацией и производительностью. Кроме того, создание и обновление воксельных структур требует значительных усилий, особенно в случае динамических изменений или разрушения объектов. Эти ограничения требуют тщательного планирования и балансировки при проектировании игровых миров.

III. Высотные карты

Высотные карты просты в реализации и управлении. Они легко генерируются и обновляются, требуя при этом меньше памяти и вычислительных мощностей. Эти карты отлично подходят для создания статичных и плавных ландшафтов, особенно в обширных открытых пространствах, таких как равнины, горные цепи и холмистые местности. Высотные карты эффективно применяются в играх на больших открытых территориях, когда требуется высокая производительность и экономия памяти, а вертикальные структуры не являются основными элементами игрового мира.

Высотные карты – это двумерные представления трёхмерного рельефа поверхности, где каждое значение пикселя соответствует высоте. Они активно используются для моделирования ландшафтов и картографических задач, а также в играх для упрощения работы с трёхмерными мирами.

В игровом 3D-пространстве высотный граф (или граф высот) позволяет учитывать вертикальные аспекты мира, что особенно полезно в играх с многоуровневыми локациями, такими как горные местности, здания с этажами или подземелья, пример высотной карты показан на рисунке 3. Для создания высотного графа из трёхмерного игрового пространства необходимо:

1. Деление на уровни высоты: Игровое пространство делится на "уровни высоты". Каждый уровень станет узлом графа. На-

пример, в горных местностях это могут быть разные высоты горы, в здании - этажи;
2. Связи между уровнями: Создаются рёбра (связи) между уровнями высоты. Это могут быть лестницы, лифты, тропы или любые другие средства перемещения между уровнями;
3. Верхние и нижние уровни: Верхние узлы будут ассоциированы с нижними узлами, создавая вертикальные пути. Например, на горе это может быть подъем и спуск по тропе, а в здании - использование лестницы между этажами;
4. Учет расстояния и времени: При создании графа принимается во внимание расстояние и время, необходимое для перемещения между уровнями высоты. Это позволит алгоритмам поиска пути учитывать не только горизонтальные, но и вертикальные перемещения.

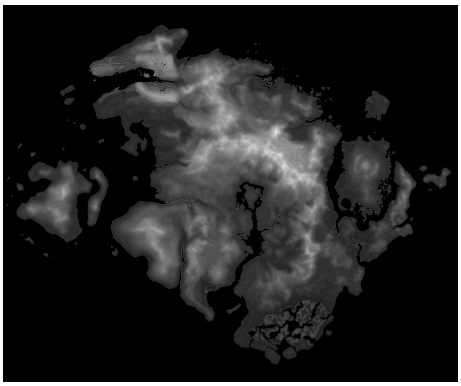

Рис. 3 – Пример высотной карты

Карты высот легко реализуются и контролируются. Их просто создавать и обновлять, занимая при этом меньше памяти и вычислительных ресурсов. Эти карты прекрасно подходят для формирования статичных и плавных ландшафтов, особенно в обширных открытых пространствах, таких как равнины, горные хребты и холмистые местности. Высотные карты эффективно используются в играх с большими открытыми территориями, когда важно высокое качество выполнения и экономия памяти, а вертикальные конструкции не являются главными элементами игрового мира.

Применяя высотный граф, можно использовать графовые алгоритмы поиска пути, такие как A*, для нахождения оптимальных маршрутов с учетом высоты. Этот подход существенно упрощает нахождение пути в сложных и многоуровневых 3D-пространствах.

Тем не менее, у данного метода есть и свои недостатки. Во-первых, высотные графы могут быть ограничены в своих возможностях представления подземелий и многоуровневых строений, таких как здания с несколькими этажами. Это может привести к необходимости использования дополнительных структур данных для полной модели игрового мира. Во-вторых, сложные вертикальные структуры, такие как мосты или навесы, сложно реализовать с помощью высотных карт, что может ограничить архитектурную гибкость и реализм игры. Наконец, обновление высотных карт в реальном времени может потребовать значительных вычислительных ресурсов, что может сказаться на производительности, особенно в играх с динамически меняющимся окружением.

IV. Иерархический граф

Иерархические графы – это структура данных, состоящая из узлов и рёбер, где узлы могут иметь вложенные подузлы, создавая иерархическую организацию. В игровом 3D-пространстве эта структура часто используется для управления сложными сценами и взаимодействиями объектов.

В игровом 3D-пространстве иерархические графы можно эффективно применять для поиска пути, создавая более управляемую и логичную структуру.Простейший игровой мир можно представить в виде иерархического графа следующим образом:

1. Вершина верхнего уровня: Главный узел представляет весь игровой мир;
2. Подуровни: Игровой мир разбивается на значимые зоны, такие как города, леса и горы. Каждая зона становится дочерним узлом главного узла;
3. Подзоны: Каждый город может делиться на кварталы, лес – на участки, а горы – на тропы. Эти подзоны станут дочерними узлами зон;
4. Конечные узлы: внутри каждого участка возможно создание узлов для конкретных объектов и мест. Например, здания в городе, деревья в лесу или пещеры в горах.

Когда игровой мир представлен иерархической структурой графа, разработчики могут более эффективно использовать различные алгоритмы поиска пути, такие как A*, Dijkstra или поиск в глубину. Локальные пути внутри одной подзоны обрабатываются быстро и эффективно, так как алгоритмы могут сосредоточиться на меньших областях. Для перемещения между зонами и подзонами используются обобщенные пути, что снижает вычислительную нагрузку и упрощает управление сложностью поиска пути. Это делает процесс более контролируемым и эффективным даже в больших и сложных игровых мирах. Иерархическая структура графа значительно упрощает навигацию, позволяя эффективно находить оптимальные маршруты и обеспечивать плавный игровой процесс.

Иерархические графы сложны в реализации и поддержке, требуя детальной проработки структуры сцены, но они более эффективны в управлении многоуровневыми сценами. Эти графы повышают производительность благодаря декомпо-

зиции задач и упрощают обновление и контроль объектов сцены. Иерархические графы также могут использоваться в сочетании с LOD (уровень детализации). Их целесообразно применять в играх с множеством взаимодействующих объектов, в сценах со сложными анимациями и зависимостями, а также в ситуациях, когда требуется управляемость и модульность сцены.

V. Комбинация различных методов

Помимо прочего, данные методы можно комбинировать.Комбинирование воксельных сеток, иерархических графов и высотных карт может значительно улучшить производительность и гибкость графовых алгоритмов поиска пути, а также позволяет максимально использовать уникальные преимущества каждого отдельного метода. Реализовать комбинации можно следующими способами:

1. Воксельные сетки и высотные карты:В игровых приложениях с изменяемыми ландшафтами и разрушаемыми объектами можно использовать воксельные сетки для моделирования подземных областей и разрушаемых стен, в то время как высотные карты используются для создания плавных поверхностей на поверхности;
2. Воксельные сетки и иерархические графы:В игровых приложениях с многоуровневыми локациями, такими как многоэтажные здания или большие подземелья, можно использовать воксельные сетки для детального представления каждого уровня, а иерархические графы для управления переходами между уровнями. Это позволяет быстро обрабатывать локальные пути внутри уровня и эффективно рассчитывать глобальные пути между уровнями;
3. Иерархические графы и высотные карты: В игровых приложениях жанров стратегия или RPG, где карта имеет сложную вертикальную структуру, высотные карты могут использоваться для определения рельефа и высотных изменений, а иерархические графы для управления переходами между различными зонами. Это облегчает поиск пути на глобальном уровне, учитывая все высотные изменения и переходы;
4. Комбинирование всех трех методов:В сложном игровом приложении с обширными мирами, можно использовать высотные карты для поверхностных ландшафтов, воксельные сетки для подземных уровней и разрушаемых объектов, а иерархические графы для управления переходами между различными зонами и уровнями. Этот подход позволяет объединить преимущества каждого метода, улучшая производительность и обеспечивая более реалистичное и интерактивное игровое пространство.

VI. Заключение

Специфика использования алгоритмов поиска путей в графах при создании трёхмерных игровых вселенных акцентирует внимание на значимости выбора подходящей структуры данных и методов оптимизации для формирования правдоподобного и увлекательного игрового опыта. Высотные карты, иерархические графы и воксельные сети предоставляют разработчикам множество инструментов для решения задач, связанных с управлением сложными и изменяющимися сценами. Каждый подход обладает своими уникальными плюсами и минусами, и их эффективное применение зависит от конкретных требований игры. Оптимизация и грамотное использование этих методов обеспечивают повышенную производительность, экономию ресурсов и создание более привлекательных и интерактивных игровых миров. В конечном счёте, использование графовых алгоритмов поиска путей позволяет разработчикам создавать глубоко проработанные и захватывающие трёхмерные игровые окружения.

1. Bagus, A. A. Graph Theory by The A* Algorithm for Pathfinding in Electronic Games / A. A. Bagus, A. A. Fasya //University of Jember
2. Barnouti, N. H. Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm /N. H. Barnouti, S. S. Mahmood //Al-Mansour University College
3. Rafiq, A. Pathfinding Algorithms in Game Development /A. Rafiq, T. A. Abdul Kadir, S. N. Ihsan //College of Computing and Applied Sciences Universiti Malaysia Pahang