Генерация уровней однопользовательской 3D-игры на основе BSP-деревьев / А. Г. Кравец, А. В. Матохина, С. Е. Драгунов и др. // Прикладная математика и вопросы управления. – 2024. – № 1. – С. 41–54. – DOI 10.15593/2499-9873/2024.1.03

Библиографическое описание согласно ГОСТ Р 7.0.100–2018
Генерация уровней однопользовательской 3D-игры на основе BSP-деревьев / А. Г. Кравец, А. В. Матохина, С. Е. Драгунов, Н. А. Сальникова. – Текст : непосредственный // Прикладная математика и вопросы управления / Applied Mathematics and Control Sciences. – 2024. – № 1. – С. 41–54. – DOI 10.15593/2499-9873/2024.1.03

пермский политех

ПРИКЛАДНАЯ МАТЕМАТИКА И ВОПРОСЫ УПРАВЛЕНИЯ
№ 1, 2024
https://ered.pstu.ru/index.php/amcs

Научная статья
DOI: 10.15593/2499-9873/2024.1.03
УДК 004.8:004.946

Генерация уровней однопользовательской 3D-игры на основе BSP-деревьев

А.Г. Кравец[1,2], А.В. Матохина[1], С.Е. Драгунов[1], Н.А. Сальникова[3]

[1]Волгоградский государственный технический университет, Волгоград, Российская Федерация
[2]Государственный университет Дубна, Дубна, Российская Федерация
[3]Волгоградский институт управления – филиал Российской академии народного хозяйства и государственной службы при Президенте Российской Федерации, Волгоград, Российская Федерация

О СТАТЬЕ

Получена: 17 марта 2024
Одобрена: 25 апреля 2024
Принята к публикации: 27 апреля 2024

Финансирование
Исследование не имело спонсорской поддержки.
Конфликт интересов
Авторы заявляют об отсутствии конфликта интересов.
Вклад авторов
равноценен.

Ключевые слова:
искусственный интеллект, машинное обучение, видеоигры, однопользовательские видеоигры, алгоритмы генерации уровней видеоигры, процедурная генерация, BSP-tree, алгоритм туннелирования, клеточные автоматы 3D-модели, симуляция, виртуальная реальность.

АННОТАЦИЯ

Рассматриваются задачи генерации уровней для однопользовательских 3D игр и предлагается метод, направленный на ускорение и удешевление процесса разработки игровых уровней, раскрываются подходы и технологии, примененные для решения задач генерации уровней, реализации алгоритмов генерации комнат и путей между ними. После проведения анализа существующих методов генерации уровней выбран метод генерации BSP-tree, который может создавать уникальные уровни на основе входных переменных, позволяющий сократить сроки разработки игровых уровней.

Создание бесконечного уровня – сложная задача, однако с использованием некоторых полезных советов и техник становится гораздо проще. Первым шагом для организации бесконечного уровня является создание пустого объекта, который служит основой для уровня, затем можно добавлять различные элементы окружения. Для достижения эффекта бесконечного уровня предлагается использовать технику «прокрутки». Это означает, что когда игрок движется в одном направлении, то объекты в уровне перемещаются в противоположном направлении. Это создает иллюзию бесконечности и позволяет игроку продолжать исследовать новые области уровня.

© **Кравец Алла Григорьевна** – доктор технических наук, профессор, профессор кафедры систем автоматизированного проектирования и поискового конструирования, e-mail: allagkravets@yandex.ru, ORCID 0000-0003-1675-8652.
Матохина Анна Владимировна – кандидат технических наук, доцент, доцент кафедры систем автоматизированного проектирования и поискового конструирования, e-mail: matokhina.a.v@gmail.co, ORCID 0000-0002-6178-2988.
Драгунов Станислав Евгеньевич – магистрант кафедры систем автоматизированного проектирования и поискового конструирования, e-mail: dragunov.stanislav.e@gmail.com, ORCID 0000-0001-8879-4885.
Сальникова Наталия Анатольевна – кандидат технических наук, доцент, доцент кафедры информационных систем и математического моделирования, e-mail: ns3112@mail.ru, ORCID 0000-0001-9184-0387.

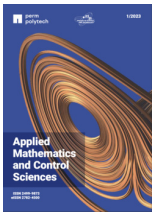
Perm Polytech Style: Kravets A.G., Matokhina A.V., Dragunov S.E., Salnikova N.A. Generation of levels of a single-player 3D game based on BSP trees. *Applied Mathematics and Control Sciences.* 2024, no. 1, pp. 41–54. DOI: 10.15593/2499-9873/2024.1.03

MDPI and ACS Style: Kravets, A.G.; Matokhina, A.V.; Dragunov, S.E.; Salnikova, N.A. Generation of levels of a single-player 3D game based on BSP trees. *Appl. Math. Control Sci.* **2024, 1**, 41–54. https://doi.org/10.15593/2499-9873/2024.1.03

Chicago/Turabian Style: Kravets, Alla G., Anna V. Matokhina, Stanislav E. Dragunov, and Natalia A. Salnikova. 2024. "Generation of levels of a single-player 3D game based on BSP trees". *Appl. Math. Control Sci.* no. 1: 41–54. https://doi.org/10.15593/2499-9873/2024.1.03

perm polytech

APPLIED MATHEMATICS AND CONTROL SCIENCES
№ 1, 2024
https://ered.pstu.ru/index.php/amcs



Article
DOI: 10.15593/2499-9873/2024.1.03
UDC 004.8:004.946

Generation of levels of a single-player 3D game based on BSP trees

A.G. Kravets[1,2], A.V. Matokhina[1], S.E. Dragunov[1], N.A. Salnikova[3]

[1]Volgograd State Technical University, Volgograd, Russian Federation
[2]Dubna State University, Dubna, Russian Federation
[3]Volgograd Institute of Management – branch of the Russian Presidential Academy of National Economy and Public Administration, Volgograd, Russian Federation

ARTICLE INFO

Received: 17 March 2024
Approved: 25 April 2024
Accepted for publication: 27 April 2024

Funding
This research received no external funding.
Conflicts of Interest
The authors declare no conflict of interest.
Author Contributions
equivalent.

Keywords:
artificial intelligence, machine learning, video games, single-player video games, video game level generation algorithms, procedural generation, BSP-tree, tunneling algorithm, cellular automata 3D models, simulation, virtual reality.

ABSTRACT

The paper examines the problems of level generation for single-player 3D games and proposes a method aimed at speeding up and reducing the cost of the process of developing game levels, revealing the approaches and technologies used to solve the problems of level generation, implementing algorithms for generating rooms and paths between them. After analyzing existing methods for generating levels, the BSP-tree generation method was selected, which can create unique levels based on input variables, allowing to reduce the development time of game levels.

Creating an endless level is a difficult task, but with some helpful tips and techniques it becomes much easier. The first step to organizing an infinite level is to create an empty object that serves as the base for the level, then you can add various environmental elements. To achieve the effect of an endless level, it is proposed to use the "scrolling" technique. This means that when the player moves in one direction, objects in the level move in the opposite direction. This creates the illusion of infinity and allows the player to continue exploring new areas of the level.

© **Alla G. Kravets** – Doctor of Technical Sciences, Professor, Professor of the Department of Computer-Aided Design and Search Design, e-mail: allagkravets@yandex.ru, ORCID 0000-0003-1675-8652.
Anna V. Matokhina – CSc of Technical Sciences, Associate Professor, Department of Computer-Aided Design and Search Design, e-mail: matokhina.a.v@gmail.co, ORCID 0000-0002-6178-2988.
Stanislav E. Dragunov – graduate student of the Department of Computer-Aided Design and Search Design, e-mail: dragunov.stanislav.e@gmail.com, ORCID 0000-0001-8879-4885.
Natalia A. Salnikova – CSc of Technical Sciences, Associate Professor, Department of Information Systems and Mathematical Modeling, e-mail: ns3112@mail.ru, ORCID 0000-0001-9184-0387.

Введение

Разработка компьютерных игр является активно развивающейся и захватывающей областью программирования и развлекательной индустрии. За последние годы мобильные игры сделали огромный скачок в своем качестве. Процедурная генерация уровней – ключевая особенность жанра rogue-like и rogue-lite компьютерных игр. Генерация 3D-уровней – более сложная и менее исследованная задача, по сравнению с 2D-генерацией.

В играх с процедурной генерацией контент (окружение, квесты, персонажи и так далее) не заготовлен заранее, а создается с помощью алгоритмов. Это делает игровой процесс менее предсказуемым и повышает реиграбельность [1; 2].

Процедурно генерируемые карты позволяют создавать игровой контент с помощью алгоритмов самостоятельно или с помощью игроков. Особенностью этого жанра является то, что схема игры заранее неизвестна, способов генерации карт большое множество, игрок должен обязательно пройти сгенерированный уровень и подняться на следующий. Рандомизированные карты влияют на все аспекты геймплея, начиная от стратегии и тактического поведения, до расположения предметов и врагов [3; 4].

Преимущества рандомизированных карт заключаются в большой вариативности механик и контента, обеспечивая высокую реиграбельность, продвижение игрока зависит от его собственных способностей, схему игры заранее узнать нельзя. На сегодняшний день разработано много способов генерации карт, но не один из них не является универсальным.

В связи с этим поставлена цель – провести анализ методов процедурной генерации уровней для интеграции в проект в играх жанра roguelike, выполнить мониторинг существующих решений, используемых в играх, и выбрать один из существующих методов генерации уровней для реализации однопользовательской 3D-игры.

Создание бесконечного уровня может показаться сложной задачей, однако с использованием ряда полезных советов и техник это становится гораздо проще. Первым шагом для создания бесконечного уровня является создание пустого объекта, который является основой для уровня. Затем можно добавлять различные элементы окружения, такие как блоки, растения или препятствия, чтобы создать интересные игровые ситуации [5; 6].

Для достижения эффекта бесконечного уровня целесообразно использовать технику «прокрутки». Это означает, что когда игрок движется в одном направлении, то объекты в уровне перемещаются в противоположном направлении. Это создает иллюзию бесконечности и позволяет игроку продолжать исследовать новые области уровня.

1. Анализ процесса генерации уровней и существующих методов

1.1. Анализ процедурной генерации уровней в 3D-играх

Процедурная генерация обеспечивает создание различных контентов не вручную, а с помощью алгоритмов без участия разработчиков, игроки могут менять параметры, настраивать формулу процедурной генерации. Генерация различных уровней осуществляется в соответствии с заранее написанными правилами, которые создают различные миры, наполненные множеством предметов, персонажей, оружия, музыки, избегая неуместных нестыковок [7; 8].

Модули генерации уровней для 3D-игры могут быть полезны для разработчиков игр, которым нужно генерировать большое количество уровней для своих игр. Это может быть особенно полезно для игр с открытым сюжетом, где игроки могут путешествовать по случайно созданным мирам.

Практическое применение модулей генерации уровней в 3D-игре может заключаться в сокращении времени, которое потребуется для создания большого количества уровней вручную. Вместо того, чтобы создавать каждый уровень вручную, разработчики могут использовать модули генерации уровней, чтобы создавать уровни автоматически. Это может сократить время разработки и увеличить количество уровней, доступных в игре.

Кроме того, модули генерации уровней могут сделать игровой процесс более интересным и разнообразным. Поскольку каждый уровень будет создаваться случайным образом, каждый раз, когда игрок проходит игру, он будет испытывать новые вызовы и новые ощущения. Это может сделать игру более захватывающей и продлить ее жизнь [9; 10]. Методы генерации уровней в играх существуют уже долгое время и продолжают эволюционировать. В прошлом разработчики игр использовали простые методы генерации уровней, такие как случайное расположение объектов, тайлсеты и прочее. Однако с развитием технологий для генерации случайных чисел и машинного обучения методы генерации уровней в играх стали более сложными и интересными [11; 12].

Например, в некоторых играх используются алгоритмы процедурной генерации, которые позволяют создавать уровни, основанные на определенных параметрах и правилах [13; 14]. Такие алгоритмы могут создавать уровни, которые соответствуют различным условиям, например, уровни, которые изменяются в зависимости от действий игрока, или уровни, которые генерируются на основе анализа прошлых уровней и действий игроков.

Кроме того, с развитием технологий и машинного обучения, некоторые разработчики игр начали использовать алгоритмы генерации, которые могут анализировать игроков и создавать уникальные уровни, соответствующие способностям и предпочтениям каждого игрока.

Таким образом, можно утверждать, что методы генерации уровней в играх продолжают эволюционировать и становиться все более сложными и интересными.

1.2. Обзор существующих методов генерации уровней

Методы генерации уровней в играх существуют долгое время, и они продолжают эволюционировать. В прошлом разработчики игр использовали простые методы генерации уровней, такие как случайное расположение объектов, тайлсеты и прочее. Однако с развитием технологий для генерации случайных чисел и машинного обучения методы генерации уровней в играх стали более сложными и интересными.

1.2.1. Метод BSP-tree

Проведем анализ существующих методов генерации уровней в 3D-играх. Рассмотрим метод BSP-tree для генерации уровней в играх. Он разработан в начале 90-х гг. и до сих пор является одним из наиболее популярных способов создания игровых уровней. BSP-tree позволяет генерировать уровни с различными разнообразными текстурами, формами и размерами. Этот метод гарантирует видимость каждой части уровня, что улучшает производительность игры и обеспечивает более высокую скорость загрузки уровня. BSP-tree может быть использован для оптимизации процесса выбора пути, что повышает качество геймплея игры [15; 16]. Однако обработка больших уровней может занять много времени и затратить дополнительные ресурсы компьютера. Принцип BSP-tree может порождать сложную взаимосвязь между соединенными участками уровня. Сложность уровней может быть высокой по сравнению с другими методами.

Метод генерации уровней Bsp-tree широко используется в различных играх и симуляторах. Например, метод Bsp-tree использован в классической игре Doom. Его применение

позволило создать уровни с различными разнообразными текстурами и формами, что улучшило игровой процесс и привлекло внимание множества игроков. Игра Quake, еще одна популярная игра, также использовала метод Bsp-tree для генерации уровней. В результате этого уровни стали более динамичными и интересными для игроков. Bsp-tree также можно использовать для создания ландшафтов и миссий в симуляторах полетов. Например, Microsoft Flight Simulator и X-Plane используют этот метод генерации уровней, чтобы создавать уникальные и интересные миссии.

В целом Bsp-tree может быть хорошим выбором для многих игр и оставляет много места для масштабирования и инноваций.

1.2.2. Алгоритм туннелирования

Метод генерации уровней с использованием алгоритма туннелирования – это один из способов создания случайных уровней в играх. Он работает, создавая лабиринт из туннелей, которые соединяют различные комнаты и зоны в уровне. Алгоритм туннелирования может создавать уровни с широким разнообразием форм и размеров, что может повышать интересность игры для игроков [17; 18]. Этот метод может быть легко расширен добавлением более софистицированных алгоритмов генерации комнат и зон. Алгоритм может быть настроен для создания заданных параметров, таких как количество комнат, сложность уровня, проходимость и т.д. Недостатком метода является возможное создание неэффективных маршрутов в уровне, которое может понизить качество геймплея. Некоторые игроки могут предпочитать более предсказуемые уровни, что может снизить интерес к игре. Генерируемый уровень может быть затратным в плане производительности, если количество объектов и детализации туннелей высокие.

Метод генерации уровней с использованием алгоритма туннелирования также используется в различных играх. В Minecraft алгоритм туннелирования используется для генерации большинства уровней в игре, что позволяет создавать большие и разнообразные миры, которые могут быть исследованы игроками. Игра Dwarf Fortress. Алгоритм туннелирования также используется в игре Dwarf Fortress для создания уровней лабиринтов и пещер, которые могут содержать множество разнообразных сокровищ и опасностей для игроков. Игра Spelunky использует метод туннелирования для генерации уровней, которые создаются каждый раз, когда игрок начинает новую игру. Это делает игру более интересной и разнообразной для игроков.

Алгоритм туннелирования следует выбирать для разработки случайных уровней для игр, он имеет преимущества и недостатки, которые разработчики должны учитывать при выборе метода генерации уровней.

1.2.3. Клеточные автоматы

Метод генерации уровней с использованием клеточных автоматов – это один из наиболее популярных способов создания случайных уровней в играх. Он работает, создавая искусственные клеточные структуры, которые могут быть использованы для генерации уровней. Клеточные автоматы могут создавать уровни с разнообразными формами и размерами, что увеличивает интерес для игроков. Этот метод может быть легко расширен использованием различных алгоритмов, которые могут управлять формой, цветом и расположением клеток, что позволяет значительно ускорить процесс создания уровней [19; 20]. Создание макетов может быть затратным по ресурсам, так как генерация уров-

ней на основе клеточных автоматов обычно требует большого количества вычислительных ресурсов. Этот метод может иметь ограниченный контроль над генерируемыми уровнями в текстурах, игроки могут определить закономерности генерации уровней, что может снизить интерес к игре.

Метод генерации уровней на основе клеточных автоматов широко используется в различных играх и приложениях. Игра No Man's Sky использует метод генерации уровней на основе клеточных автоматов для создания случайных галактик и планет в игре. Это позволяет создавать множество уникальных и разнообразных миров, которые могут быть исследованы игроками. В Diablo III метод клеточных автоматов используется для генерации случайных лабиринтов и подземелий, которые игроки могут исследовать. Это делает игру более интересной и разнообразной для игроков. Игра RimWorld также использует метод генерации уровней на основе клеточных автоматов для создания случайных миров для игроков. Это позволяет создавать множество уникальных и интересных ситуаций и задач, которые игрокам требуется решить. Метод генерации уровней с использованием клеточных автоматов можно применять для создания случайных уровней в играх.

1.2.4. Результаты сравнения существующих методов

Результаты сравнения рассмотренных методов генерации уровней 3D-игр приведены в таблице.

Сравнение методов генерации уровней

Название метода	Преимущества	Недостатки
BSP-деревья	Генерация уровней с различными текстурами и формами, высокая производительность игры, повышенное качество геймплея, масштабируемость алгоритма, настраиваемость	Увеличение уровней требует много ресурсов, порождает сложную взаимосвязь между соединенными участками уровня, высокая сложность уровней
Алгоритмы туннелирования	Разнообразие форм и размеров, легко расширяем, высокая настраиваемость	Неэффективные соединения, большая непредсказуемость, ресурсозатратность
Клеточные автоматы	Разнообразие форм и размеров, легко расширяем, высокая скорость работы	Ресурсозатратность, ограниченный контроль, закономерности в структуре уровня

После проведения сравнительного анализа рассмотренных методов генерации уровней выбран метод BSP-дерева для проектирования модуля генерации уровней однопользовательской 3D-игры, так как он отличается высокой производительностью, повышенным качеством геймплея, имеет масштабируемость алгоритма и гибкую настраиваемость, позволяет проводить генерацию уровней с различными текстурами и формами.

2. Метод построения BSP-деревьев

Метод BSP-деревьев – это процесс построения деревьев за счет разбиения n-мерного пространства с использованием гиперплоскость, на которой расположен тот или иной полигон. Каждый узел BSP-дерева представляет собой выпуклое подпространство и хранит информацию о плоскости, которая делит пространство, и ссылки на два новых узла [21; 22].

Двоичное разбиение множества объектов на плоскости строится с помощью рекурсивного разбиения плоскости прямыми, которые разбивают не только плоскость, но и объекты, расположенные на ней. Площадь уровня принимается за один лист BSP-дерева, а каждая вершина дерева обозначает поверхность. Каждый лист BSP-дерева случайным образом делится на две части, которые в следующем уровне становятся листьями, как показано на рис. 1. Действия алгоритма повторяются, пока размер листа не достигнет минимального порога.

Метод бинарного разбиения пространства позволяет определить видимость объектов, дерево всегда помогает создать само себя. Этот метод прост и удобен в реализации, с его помощью легко и быстро можно получить необходимое множество меньших фигур без пустых пространств. Каждый лист дерева соответствует грани разбиения и хранит все находящиеся на ней объекты, а каждый узел дерева соответствует разбивающей прямой, которая хранится в этом узле (рис. 2).

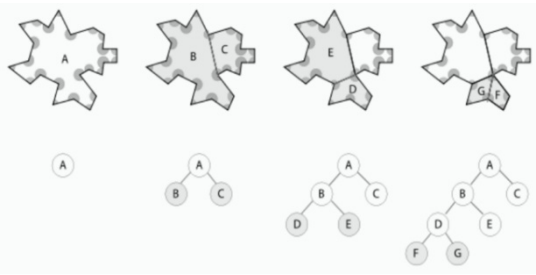


Рис. 1. Схема выполнения алгоритма BSP

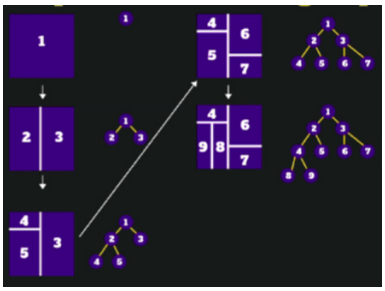


Рис. 2. Схема разделения пространства

2.1. Модификация алгоритма построения BSP-деревьев

Алгоритм Binary Space Partitioning (BSP) представляет собой иерархическое разбиение пространства, но он не предусматривает создания переходов между пространствами. Возникает необходимость дополнить алгоритм.

Для создания проходов между комнатами необходимо найти соседние комнаты, которые можно соединить. Для решения этой задачи листья дерева представляются в виде графа, по которому осуществляется обход его вершин (рис. 3).

Обход графа – это процесс посещения всех вершин графа или определенной части графа, который начинается с начальной вершины и заканчивается, когда все вершины были посещены или определенная цель достигнута.

Для проверки берутся две вершины (одного родителя) и проверяется их взаимное положение в пространстве. На основе определения их относительного положения добавляется проход между ними (рис. 4).

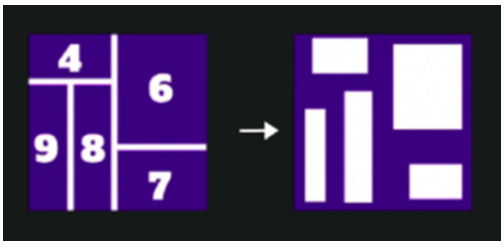


Рис. 3. Схема создания комнат

Рис. 4. Проверка соседей

Рис. 5. Блок-схема алгоритма

Основные этапы работы алгоритма следующие:

• вначале необходимо определить пространство для разбиения, для этого требуется:

– задать начальное прямоугольное пространство;

– задать минимальный размер комнаты;

• затем провести разделение пространства:

– проверить, можно ли разделить пространство на две части вертикальной или горизонтальной линией;

– если можно разделить, то добавить новые пространства к дереву;

– проверить, можно ли разделить новые пространства дальше;

– если можно разделить, то добавить новые пространства к дереву;

• на этом этапе производят создание комнат:

– для каждого пространства следует создать комнату, выбрав случайные угловые точки внутри пространства;

• далее производят создание коридоров;

– начиная с молодых ветвей, провести коридоры между листьями одного родителя;

– подняться на слой выше и повторить действия.

Блок-схема работы алгоритма приведена на рис. 5.

2.2. Описание работы класса BinarySpacePartitioner

Приведем описание работы класса BinarySpacePartitioner.

1. Создание корневого узла rootNode.

2. Конструктор BinarySpacePartitioner принимает ширину и длину подземелья dungeonWidth и dungeonLength соответственно. Он создает корневой узел rootNode с координатами (0, 0) и (dungeonWidth, dungeonLength).

3. Метод PrepareNodesCollection принимает максимальное количество итераций maxIterations, минимальную ширину и длину комнат roomWidthMin и roomLengthMin соответственно. Он создает очередь graph, добавляет корневой узел в очередь и список listToReturn и добавляет корневой узел в список.

4. Метод использует цикл while с условием, что количество итераций меньше максимального значения и очередь не пуста. На каждой итерации он извлекает первый узел из очереди graph, проверяет, может ли текущий узел быть разделен на две части методом SplitTheSpace, добавляет новые узлы в список listToReturn и очередь graph.

5. Метод SplitTheSpace принимает текущий узел currentNode, список listToReturn, минимальную ширину и длину комнат roomWidthMin и roomLengthMin и очередь graph.

Он вызывает метод GetLineDividingSpace, который возвращает линию разделения пространства, и разделяет текущий узел на два узла node1 и node2. Затем он добавляет новые узлы в список listToReturn и очередь graph методом AddNewNodeToCollections.

6. Метод AddNewNodeToCollections принимает список listToReturn, очередь graph и узел node. Он добавляет узел в список listToReturn и в очередь graph.

7. Метод GetLineDividingSpace принимает нижний левый угол bottomLeftAreaCorner, верхний правый угол topRightAreaCorner, минимальную ширину и длину комнат roomWidthMin и roomLengthMin. Он определяет ориентацию линии разделения пространства и возвращает объект Line, содержащий ориентацию и координаты линии.

8. Метод GetCoordinatesFororientation принимает ориентацию линии orientation, нижний левый угол bottomLeftAreaCorner, верхний правый угол topRightAreaCorner, минимальную ширину и длину комнат roomWidthMin и roomLengthMin. Он определяет случайные координаты для линии разделения пространства в соответствии с ориентацией и возвращает их как объект Vector2Int.

3. Проектирование интерфейса модуля генерации уровней

При проектировании интерфейса для модуля генерации уровней важно также учитывать возможности игрового движка и платформы, на которой работает игра. Интерфейс должен быть адаптирован под различные устройства и экраны и совместим с технологиями, используемыми в игровом движке Unity.

Также следует учитывать, что пользователи имеют различный уровень опыта и навыков, поэтому интерфейс должен быть разработан таким образом, чтобы он был понятен и легок в использовании как для новичков, так и для опытных пользователей.

Один из ключевых аспектов проектирования интерфейса для модуля генерации уровней – это удобство использования.

Интерфейс должен содержать настройки уровня и возможность создать новый уровень, очистить сцену, сохранить уровень, загрузить уровень. Пример готового интерфейса, интегрированного в Unity, приведен на рис. 6.

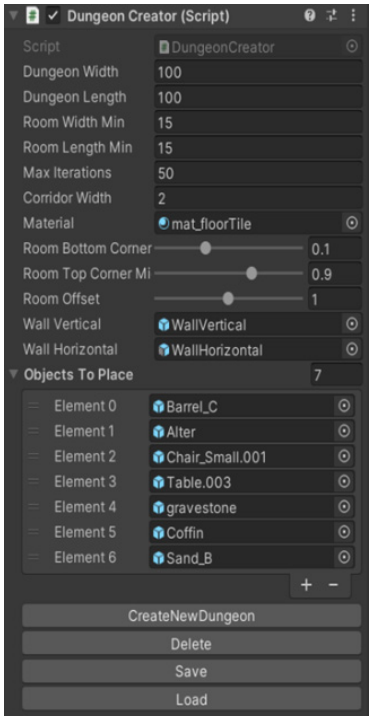


Рис. 6. Панель настроек

Ключевые элементы управления интерфейсом для модуля генерации уровней включают следующие этапы:

1. Панель настроек: это основной элемент управления, позволяющий настраивать параметры генерации уровней. Эта панель может включать элементы управления, такие как ползунки, переключатели, выпадающие списки и т.д., которые позволяют пользователю настраивать различные параметры, такие как размер уровня, количество комнат, сложность и т.д.

2. Кнопка генерации: это кнопка, которая запускает процесс генерации уровня с текущими настройками.

3. Кнопка очистки сцены: это кнопка, которая запускает процесс очистки сцены от объектов.

4. Сохранение и загрузка настроек: это элементы управления, позволяющие игроку сохранять и загружать настройки генерации уровней. Это полезно, если игрок хочет сохранить настройки для будущих игр или поделиться ими с другими игроками.

5. Просмотр уровня: это элемент управления, который позволяет пользователю просмотреть созданный уровень в 3D-виде. Это позволяет оценить настройки генерации и внести изменения при необходимости.

Качественно разработанный интерфейс может значительно улучшить пользовательский опыт и сделать процесс создания уровней более удобным и эффективным.

На рис. 7 показан общий вид игрового уровня, полученного в результате работы модуля генерации уровней.

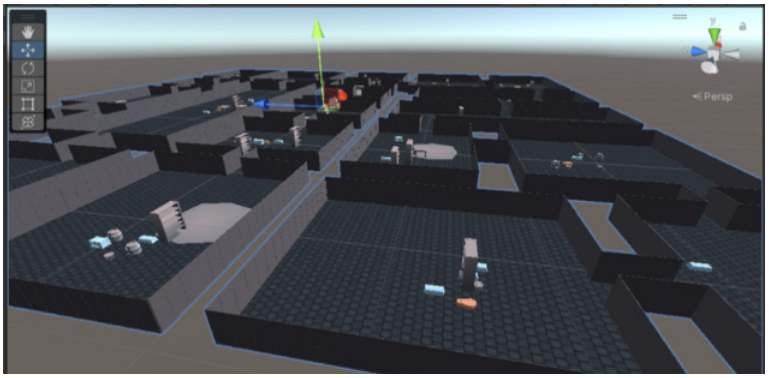


Рис. 7. Общий вид игрового уровня

4. Описание функционала модуля генерации уровней для однопользовательской 3D-игры

Модуль генерации уровней для однопользовательской 3D-игры предназначен для создания случайных уровней с различными типами комнат, коридоров и объектов на основе алгоритма BSP (Binary Space Partitioning).

Модуль включает следующие функции:

- создание уровня с заданными параметрами, такими как размер уровня, количество комнат и коридоров, сложность лабиринта, минимальный и максимальный размер комнаты, ширина коридора;
- размещение объектов на уровне, таких как стены, двери, окна, мебель и прочее;
- проверку правильности соединения комнат и коридоров, отсутствия пересечений стен и объектов, отсутствия замкнутых путей в лабиринте (возможность из одной точки уровня пройти в любую другую точку);
- возможность сохранения сгенерированного уровня в файл и загрузки его для последующего использования в игре;
- настройку параметров генерации уровня в зависимости от требований игры, таких как стиль игры.

Входными данными программы должны быть значения для настройки модуля, модели и текстуры из которых собирается уровень.

Выходными данными программы должен быть сгенерированный уровень. Уровень должен быть связанным: персонаж игрока должен из любого места достичь любого другого места. Уровень должен стилистически соответствовать сеттингу игры.

Тестирование отказоустойчивости заключалось в воспроизведении нагрузки на модуль генерации уровней. В процессе тестирования проверена работоспособность функционала модуля и его отказоустойчивость. Нагрузка совершалась увеличением размера карты и количества помещений. Также протестированы неправильные варианты использования.

При тестировании отказоустойчивости проверено, что система отрабатывает нестандартные ситуации без ошибок. Тестирование показало корректную работу модуля.

Заключение

Разработка модуля генерации уровней для однопользовательской 3D-игры может быть очень полезной для создания интересных и разнообразных уровней. BSP-tree позволяет эффективно разбивать уровень на различные секторы, которые могут быть заполнены различными объектами и элементами уровня. Это позволяет создавать уровни с различными характеристиками и сложностью, что улучшает игровой опыт и делает игру более интересной для игроков.

Разработка модуля генерации уровней также помогает оптимизировать процесс создания уровней, что может значительно снизить время, затрачиваемое на создание уровней. Алгоритм BSP-tree может быть легко интегрирован в игровой движок, что упрощает процесс разработки и позволяет быстро создавать новые уровни.

Однако при разработке модуля генерации уровней необходимо учитывать некоторые ограничения и проблемы. Например, BSP-tree может потребовать большого количества вычислительных ресурсов, особенно при работе с большими уровнями. Также при использовании BSP-tree может возникнуть трудность с генерацией проходов между комнатами, что может быть решено использованием дополнительных алгоритмов, которые позволяют осуществить обход по графу.

Разработка модуля генерации уровней представляет интересное и перспективное направление в области разработки игр. BSP-tree может быть использован для создания уровней с различными характеристиками и сложностью, что позволяет создавать более интересные и увлекательные игры. Однако при использовании BSP-tree необходимо учитывать потенциальные проблемы, такие как необходимость оптимизации и балансировки дерева, чтобы избежать слишком большого количества пересечений и улучшить производительность. Также, BSP-tree может иметь трудности с созданием динамически изменяемых уровней, где объекты могут перемещаться и изменять секторы.

Таким образом, BSP-tree представляет собой мощный инструмент для генерации уровней в 3D-играх, который может значительно улучшить качество игрового процесса. Однако его использование требует тщательной оптимизации и балансировки, чтобы достичь наилучшего результата.

Список литературы

1. Analysis of people's fascinations for computer games / R.G. Klimenko, A.G. Kravets, I.V. Strukova, N.I. Tatarova // Information Innovative Technologies. International Scientific-Practical Conference. – Moscow, 2022. – С. 48-57.

2. Кравец, А.Г. Предсказательное моделирование трендов технологического развитии / А.Г. Кравец, Н.А. Сальникова // Известия СПбГТИ(ТУ). – 2020. – № 55 (81). – С. 103–108. DOI: 10.36807/1998-9849-2020-55-81-103-108

3. Industrial cyber-physical systems: risks assessment and attacks modeling / A.G. Kravets, N. Salnikova, K. Dmitrenko, M. Lempert // Studies in Systems, Decision and Control. – 2020. – Т. 260. – С. 197–210. DOI: 10.1007/978-3-030-32648-7_16

4. Камаев, В.А. Об одном нейросетевом подходе к идентификации сложных систем / В.А. Камаев, М.В. Щербаков // Вестник компьютерных и информационных технологий. – 2004. – № 3 (3). – С. 20–24.

5. Educational computer games development: methodology, techniques, implementation / O.A. Shabalina, P.N. Vorobkalov, A.V. Kataev, A.G. Kravets // Proceedings of the 2016 Conference on Information Technologies in Science, Management, Social Sphere and Medicine (ITSMSSM 2016). – Томск, 2016. – С. 419–423. DOI: 10.2991/icaicte.2013.84

6. Bolshakov, A.A. Decision Support System for Selecting Designs of Autostereoscopic Displays / A.A. Bolshakov, A.V. Klyuchikov // Cyber-Physical Systems: Design and Application for In-dustry 4.0 (Studies in Systems, Decision and Control). – 2021. – Т. 342. – С. 73–88. DOI: 10.1007/978-3-030-66081-9_6

7. Процедурная генерация в играх [Электронный ресурс]. – URL: https://skillbox.ru/media/gamedev/protsedurnaya_generatsiya_v_igrakh/ (дата обращения: 20.02.2024).

8. Спиридонов, М.П. Процедурная генерация игровых уровней в UNITY / М.П. Спиридонов, С.В. Ковалев // Информатика и вычислительная техника: сб. науч. тр. – Чебоксары: Чувашский гос. ун-т им. И.Н. Ульянов, 2021. – С. 264–268.

9. Кузеванов, К.И. Разработка игрового приложения с процедурной генерацией уровней / К.И. Кузеванов // Материалы XXX научной конференции Амурского государственного университета. – Благовещенск: Амурский гос. ун-т, 2021. – С. 32–34.

10. Третьяков, А.А. Процедурная генерация массивной 3D-геометрии с использованием улучшенного алгоритма Marching Cubes / А.А. Третьяков // Южно-Сибирский научный вестник. – 2021. – № 4(38). – С. 8–15.

11. Качалов, Д.Л. Разработка веб-приложения для поиска в больших данных событий с заданными параметрами / Д.Л. Качалов, М.В. Щербаков // XIX Региональная конференция молодых исследователей Волгоградской области. – Волгоград, 2015. – С. 195–197.

12. Чан, В.Ф. Обзор архитектур систем поддержки принятия решений, использующих аналитику данных в режиме реального времени / В.Ф. Чан, М.В. Щербаков, Т.А. Нгуен // Известия Волгоградского государственного технического университета. – 2016. – № 3 (182). – С. 95–100.

13. Старых, Ф.А. Автоматизация процедурной генерации уровней в видеоиграх на основе МЭС / Ф.А. Старых, Т.А. Саргсян, А.Г. Ванцян. – М.: Инфра-М, 2022. – С. 281–286.

14. Фатеенков, Д.В. Процедурная генерация игровых локаций на Unity / Д.В. Фатеенков // Постулат. – 2022. – № 2 (76). – С. 10–11.

15. Введение в BSP деревья [Электронный ресурс]. – URL: https://gamedev.ru/code/articles/BSP/ (дата обращения: 20.02.2024).

16. Wang, T. An object-oriented BSP tree algorithm for Hidden Surface Removal / T. Wang // International Journal of Image and Graphics. – 2002. – Vol. 2, no. 3. – P. 395–411.

17. Как работает алгоритм «туннелирования» для генерации игровых уровней [Электронный ресурс]. – URL: https://dtf.ru/gamedev/113303/ (дата обращения: 20.02.2024).

18. Что такое VPN-туннель [Электронный ресурс]. – URL: https://selectel.ru/blog/vpn-tunnel/ (дата обращения: 20.02.2024).

19. Эволюционирующие клеточные автоматы [Электронный ресурс]. – URL: https://habr.com/ru/articles/455958/ (дата обращения: 20.02.2024).

20. Лобанов, А.И. Модели клеточных автоматов. Компьютерные исследования и моделирование / А.И. Лобанов. – 2010. – Т. 2, вып. 3. – С. 273–293.

21. Использование BSP-деревьев для создания игровых карт [Электронный ресурс]. – URL: https://habr.com/ru/articles/332832/ (дата обращения: 20.02.2024).

22. Game Engine своими руками #3: BSF-деревья [Электронный ресурс]. – URL: https://xakep.ru/2001/12/24/14185/ (дата обращения: 20.02.2024).

References

1. Klimenko R.G., Kravets A.G., Strukova I.V., Tatarova N.I. Analysis of people's fascinations for computer games. *Information Innovative Technologies. International Scientific-Practical Conference.* Moscow, 2022, pp. 48-57.

2. Kravets A.G., Sal'nikova N.A. Predskazatel'noe modelirovanie trendov tekhnologicheskogo razvitii. *Izvestiia SPbGTI(TU)*, 2020, no. 55 (81), pp. 103-108. DOI 10.36807/1998-9849-2020-55-81-103-108.

3. Kravets A.G., Salnikova N., Dmitrenko K., Lempert M. Industrial cyber-physical systems: risks assessment and attacks modeling. *Studies in Systems, Decision and Control*, 2020, vol. 260, pp. 197-210. DOI: 10.1007/978-3-030-32648-7_16.

4. Kamaev V.A., Shcherbakov M.V. Ob odnom neirosetevom podkhode k identifikatsii slozhnykh system. *Vestnik komp'iuternykh i informatsionnykh tekhnologii*, 2004, no. (3), pp. 20-24.

5. Shabalina O.A., Vorobkalov P.N., Kataev A.V., Kravets A.G. Educational computer games development: methodology, techniques, implementation. *Proceedings of the 2016 Conference on Information Technologies in Science, Management, Social Sphere and Medicine (ITSMSSM 2016). Tomsk*, 2016, pp. 419-423. DOI:10.2991/icaicte.2013.84.

6. Bolshakov A.A., Klyuchikov A.V. Decision Support System for Selecting Designs of Autostereoscopic Displays. *Cyber-Physical Systems: Design and Application for In-dustry 4.0 (Studies in Systems, Decision and Control)*, 2021, vol. 342. pp. 73-88. DOI:10.1007/978-3-030-66081-9_6.

7. Procedural generation in games. Available at: https://skillbox.ru/media/gamedev/protsedurnaya_generatsiya_v_igrakh/ (Accessed February 20, 2024).

8. Spiridonov M.P., Kovalev S.V. Protsedurnaia generatsiia igrovykh urovnei v UNITY. *Informatika i vychislitel'naia tekhnika : sbornik nauchnykh trudo.* Cheboksary, Chuvashskii gos. un-t im. I.N. Ul'ianova, 2021, pp. 264-268.

9. Kuzevanov K.I. Razrabotka igrovogo prilozheniia s protsedurnoi generatsiei urov-nei. *Materialy XXX nauchnoi konferentsii Amurskogo gosudarstvennogo universiteta.* Blagoveshchensk, Amurskii gos. un-t, 2021, pp. 32-34.

10. Tret'iakov A.A. Protsedurnaia generatsiia massivnoi 3D-geometrii s ispol'zovaniem uluchshennogo algoritma Marching Cubes. *Iuzhno-Sibirskii nauchnyi vestnik*, 2021, no. 4(38), pp. 8-15.

11. Kachalov D.L., Shcherbakov, M.V. Razrabotka veb-prilozheniia dlia poiska v bol'shikh dannykh sobytii s zadannymi parametrami. *XIX Regional'naia konferentsiia molodykh issledovatelei Volgogradskoi oblasti.* Volgograd, 2015, pp. 195-197.

12. Chan V.F. Obzor arkhitektur sistem podderzhki priniatiia reshenii, ispol'zuiushchikh analitiku dannykh v rezhime real'nogo vremeni. *Izvestiia Volgogradskogo gos. tekhn. un-ta*, 2016, no. 3 (182), pp. 95-100.

13. Starykh F.A., Sargsian T.A., Vantsian A.G. Avtomatizatsiia protsedurnoi generatsii urovnei v videoigrakh na osnove MES. Moskva, Infra-M, 2022, pp. 281-286.

14. Fateenkov D. V. Protsedurnaia generatsiia igrovykh lokatsii na Unity. *Postulat*, 2022, no. 2 (76), pp. 10-11.

15. Introduction to BSP trees. Available at: https://gamedev.ru/code/articles/BSP/ (Accessed February 20, 2024).

16. Wang T. An object-oriented BSP tree algorithm for Hidden Surface Removal. *International Journal of Image and Graphics*, 2002, vol. 2, no. 3, pp. 395-411.

17. How the tunneling algorithm works to generate game levels. Available at: https://dtf.ru/gamedev/113303/ (Accessed February 20, 2024).

18. What is a VPN tunnel. Available at: https://selectel.ru/blog/vpn-tunnel/ (Accessed February 20, 2024).

19. Evolving cellular automata. Available at: https://habr.com/ru/articles/455958/ (Accessed February 20, 2024).

20. Lobanov A. I. Modeli kletochnykh avtomatov. *Komp'iuternye issledovaniia i modelirovanie*, 2010, vol. 2, iss. 3, pp. 273–293.

21. Using BSP trees to create game maps. Available at: https://habr.com/ru/articles/332832/ (Accessed February 20, 2024).

22. DIY Game Engine #3: BSF trees.Available at: https://xakep.ru/2001/12/24/14185/ (Accessed February 20, 2024).