DOI 10.37882/2223-2966.2024.8.34

МЕТОДЫ ПРОЦЕДУРНОЙ ГЕНЕРАЦИИ ВНУТРЕННЕГО ТРЁХМЕРНОГО ОКРУЖЕНИЯ ДЛЯ ИММЕРСИВНЫХ СРЕД

THE METHODS OF PROCEDURAL GENERATION OF INTERNAL 3D ENVIRONMENT FOR IMMERSIVE SURROUNDING

A. Tsyganov

Summary. Due to the increasing complexity of 3D level design, this paper examines methods for procedural generation of internal environments for immersive surrounding, with a focus on creating believable environments in computer games. The result of the research is a developed algorithm that generates a realistic interior for three-dimensional rooms.

Keywords: three-dimensional environment, environment generation, algorithms, immersive surrounding, computer games.

Цыганов Александр Дмитриевич
Иркутский государственный университет
alexashalafe99@gmail.com

Аннотация. В связи с растущей сложностью дизайна трёхмерных уровней в данной работе рассматриваются методы процедурной генерации внутреннего окружения для иммерсивных сред, с фокусом на создание правдоподобной среды в компьютерных играх. Результатом исследования является разработанный алгоритм, генерирующий реалистичный интерьер для трёхмерных комнат.

Ключевые слова: трёхмерное окружение, генерация окружения, алгоритмы, иммерсивные среды, компьютерные игры.

Введение

Сложность задачи дизайна трёхмерных уровней возрастает из-за постоянно повышающихся визуальных стандартов к интерактивному контенту [1]. Это увеличивает нагрузку на дизайнеров уровней, которые вынуждены тратить больше времени на создание сцен окружения [2]. Помимо этого, процедурно сгенерированный контент часто выглядит слишком детерминированным, что приводит к потере достоверности, которая является особенно важным элементом в иммерсивных средах, например, компьютерных играх [3]. Поэтому возможность процедурно воссоздать правдоподобное трёхмерное окружение становится важным способом сэкономить ресурсы проекта [4].

Материалы и методы

Исследование в рамках работы состоит из двух частей.

1. Изучение существующих методов и алгоритмов в области процедурной генерации [5].
2. Разработка собственного алгоритма генерации внутреннего трёхмерного окружения.

Литературный обзор

Обзор действующих методов для генерации трёхмерного окружения в данной работе включает алгоритм древовидной генерации, «Шум Перлина» и алгоритм клеточного автомата [6].

Алгоритм древовидной генерации (Tree-Based Generation) использует древовидную структуру данных для создания трёхмерного окружения [7]. Корень дерева представляет собой начальный объект, например куб или сферу. Каждый узел дерева соответствует поддереву, которое порождает дочерние узлы, рекурсивно генерируя более детализированные объекты [8].

Процесс генерации начинается с корня дерева и продолжается рекурсивно для каждого узла, до достижения установленного уровня детализации. На каждом уровне дерева применяется набор правил и процедур для модификации и дополнения объектов, что приводит к созданию более сложных и детализированных структур.

Преимущества использования данного метода.

- Генерация разнообразных структур, от простых до сложных, с высоким уровнем детализации.
- Иерархическая структура дерева позволяет управлять уровнем детализации сгенерированного окружения.
- Рекурсивный подход может эффективно генерировать сложные структуры с минимальными затратами вычислительных ресурсов.
- Проблемы и сложности, с которыми можно столкнуться при реализации:
- Алгоритм древовидной генерации может быть сложным в реализации и требует глубокого понимания структуры данных и рекурсивного программирования.
- Несмотря на гибкость, алгоритм может быть недостаточно гибким для создания реалистичных окружений с большим количеством деталей и разнообразием.

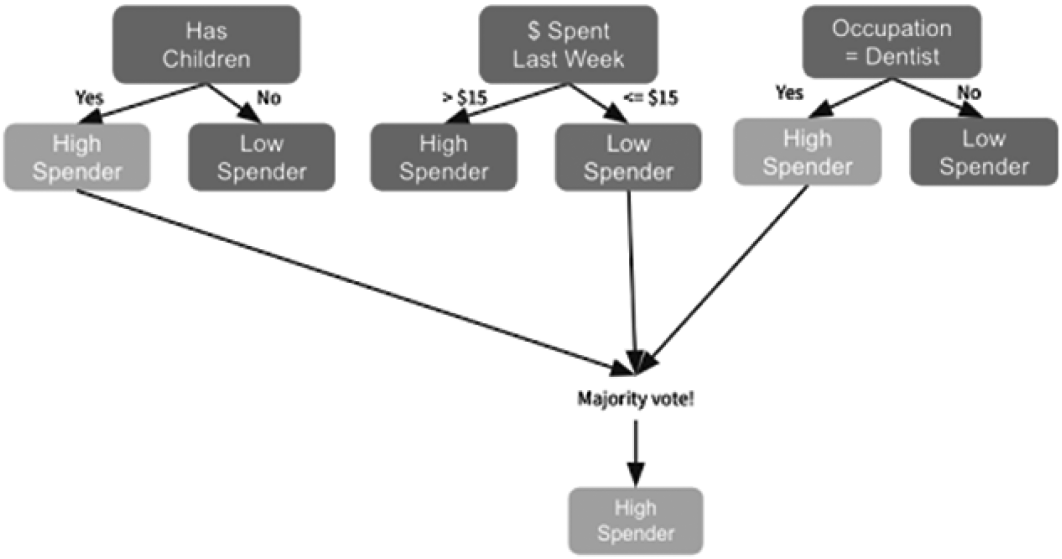

Рис. 1. Пример использования «дерева»

- Если правила генерации недостаточно сложны, то результаты могут быть повторяющимися и недостаточно уникальными.

Шум Перлина (Perlin Noise) — алгоритм, генерирующий случайные, но гладкие и непрерывные данные в многомерном пространстве. Он широко используется в компьютерной графике и игровой индустрии для создания реалистичных текстур, ландшафтов, облаков и других элементов окружения [9].

Алгоритм работает следующим образом:

1. создаётся регулярная сетка в многомерном пространстве;
2. в каждой точке сетки генерируется случайный вектор с единичной длиной;
3. для каждой точки в пространстве вычисляется значение шума с использованием интерполяции между векторными значениями в соседних точках сетки. Для интерполяции используется специальная функция сглаживания, которая обеспечивает плавный переход между значениями.

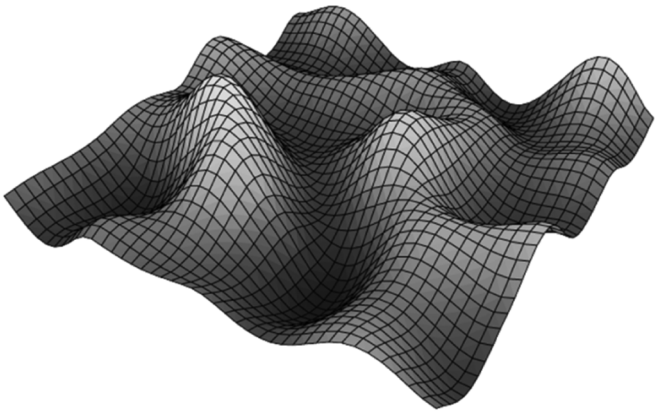

Рис. 2. Результат работы алгоритма

Основные преимущества данного метода.

- Шум Перлина является непрерывным, что обеспечивает гладкие переходы между значениями и делает его подходящим для генерации реалистичных структур.
- Алгоритм может быть периодическим, это позволяет создавать бесконечные повторяющиеся структуры, например текстуры.
- В алгоритме есть возможность настраивать параметры генерации, такие как частота, амплитуда и размер сетки, что позволяет создавать разнообразные структуры.

Проблемы и сложности, с которыми можно столкнуться при реализации.

- Шум Перлина может быть повторяющимся при больших масштабах.
- Алгоритм может быть недостаточно детализированным для создания сложных структур с большим количеством деталей.

Алгоритм клеточного автомата (Cellular Automata) — это вычислительная модель, основанная на дискретном пространстве и времени. Он использует сетку клеток, каждая из которых имеет свое состояние, и правила, которые определяют изменение состояния клеток в зависимости от состояния соседних [10].

Процесс генерации окружения с использованием клеточного автомата происходит по следующим шагам.

1. Создаётся сетка клеток в трёхмерном пространстве, каждой из которых назначается начальное состояние.
2. На каждом шаге времени для каждой клетки применяются правила, определяющие новое состояние клетки на основе состояния соседних.

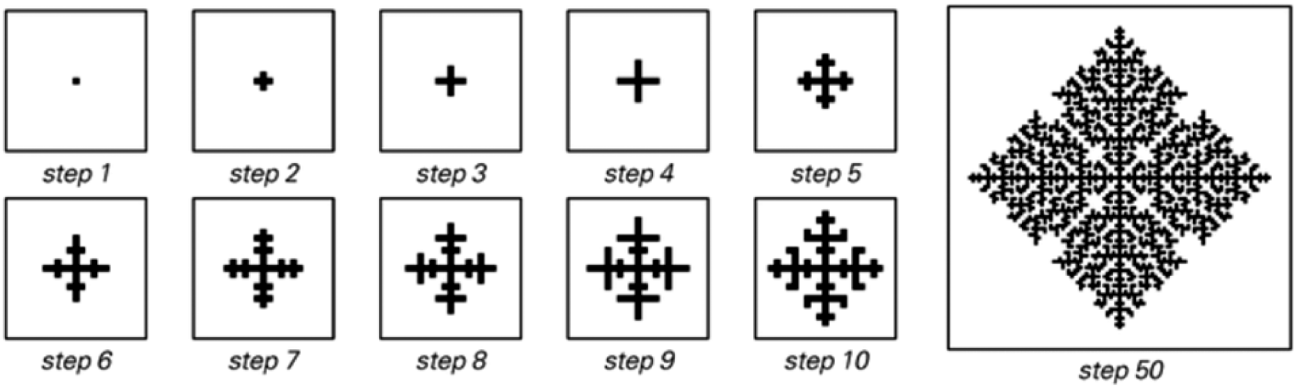

Рис. 3. Последовательность генерации

3. Шаги 2 и 3 повторяются до достижения установленного уровня стабилизации или определённого числа итераций.

Преимущества использования данного метода.

- Алгоритм работает с дискретными данными (состояниями клеток), что делает его простым и эффективным.
- Изменения состояния клеток определяются локальными правилами, которые учитывают только состояние соседних клеток. Несмотря на простоту правил, алгоритм способен выдавать сложные и непредсказуемые структуры.
- Процесс генерации является итеративным, что даёт возможность создавать сложные структуры постепенным изменением состояния клеток.

Проблемы и сложности, с которыми можно столкнуться при реализации.

- Алгоритм клеточного автомата не всегда может точно контролировать форму и размер сгенерированных структур.
- Результаты генерации могут быть повторяющимися, если правила и начальное состояние недостаточно сложны.
- Для создания желаемых структур требуется тщательная настройка правил и начального состояния клеток.

Далее в работе идёт описание собственного алгоритма для генерации внутреннего трёхмерного окружения, разработанного с учётом анализа существующих решений.

Задача, поставленная перед алгоритмом: разместить в трёхмерном пространстве с заданными размерами (комната) готовые 3D-объекты (мебель) таким образом, чтобы итоговая сцена выглядела смоделированной вручную, то есть недетерминированной. Для реализации был выбран язык программирования C++. Тестирование алгоритма проводилась в игровом движке Unreal Engine 5.

Процесс работы алгоритма включает в себя следующие этапы.

1. Предварительная подготовка: проводится пользователем вручную.
2. Построение расчётной сетки плоскости (по входным данным).
3. Расстановка объектов по сцене.

На первом этапе пользователю необходимо самостоятельно настроить для всех 3D-объектов такие данные как тип, приоритет и теги.

Тип объекта — это модель, характеризирующая место расположения объекта в комнате. Возможен выбор одного из трёх типов.

- Near_the_wall — объект должен был расположен возле стены или на стене. Пример — шкаф, телевизор, подоконник и прочее.
- Large_size — крупногабаритный предмет, может располагаться в любой точке комнаты. Это, как правило, самые важные предметы для конкретной комнаты. Пример: диван для гостиной, кровать для спальни и прочие.
- Decoration — мелкие объекты окружения, которые будут расположены в последнюю очередь.

Далее задаётся приоритет размещения — числовое значение, характеризующее, насколько важно разместить данный объект в комнате. Пользователь определяет это числовое значение для каждого предмета интерьера на своё усмотрение. В этой части стоит также упомянуть, что при работе алгоритма возможны ситуации, когда не все объекты удаётся расположить в комнате с заданными входными параметрами. Это означает, что комната слишком маленькая и в ней невозможно уместить все требуемые объекты. В таких случаях алгоритм размещает объекты в зависимости от их приоритета.

Теги нужны для определения связей между объектами, за счёт них будет создаваться реалистичное окружение. Например, как мы знаем, зубную щётку принято

ставить на раковину для умывания или рядом с ней. В таком случае 3D-сцена будет выглядеть неорганично, если алгоритм решит, что щётку можно поставить посреди комнаты. Теги помогают определить связи, благодаря которым алгоритм поймёт, что объекты должны располагаться рядом.

Второй этап работы — построение расчётной сетки. Входными данными выступают числовые значения — размеры комнаты в метрах, их подаёт пользователь. Согласно этим числам, алгоритм создаст матрицу, которая нужна, чтобы помечать, какие ячейки уже заняты, а на какие можно поставить объект. По этим отметкам объектам будут присвоены соответствующие координаты для размещения на сцене.

Третий этап — непосредственная расстановка объектов согласно параметрам объекта (этап 1) и матрице доступности (этап 2). Для каждого предмета сначала просчитываются и записываются в массив все варианты расположения, а затем их них выбирается случайный. В соответствующие ячейки матрицы доступности ставятся отметки о том, что данные ячейки больше недоступны для размещения других объектов. Алгоритм продолжает работу по такому принципу до тех пор, пока не разместит на сцене все объекты.

Результаты

Данный алгоритм предлагает систематический подход к генерации внутреннего трёхмерного окружения, используя ручную настройку параметров объектов и матрицу доступности для определения их расположения.

К достоинствам данного алгоритма можно отнести.

- Гибкость: алгоритм позволяет пользователю настроить свойства каждого объекта, определяя его тип, приоритет и теги, что даёт большую степень контроля над результатом.
- Учёт ограничений: использование матрицы доступности позволяет учитывать размеры комнаты и препятствовать перекрытию объектов, что делает генерируемое окружение более реалистичным.
- Управление связями: система тегов помогает создавать отношения между объектами, что повышает правдоподобность генерируемого окружения.

К недостаткам алгоритма можно отнести.

- Ручная настройка.
- Ограниченная случайность: случайный выбор расположения объектов из предопределенного набора может привести к однообразным результатам.

План по дальнейшей доработке алгоритма включает в себя добавление функции автоматической генерации параметров объектов, основанной на предварительно обученной модели.

Заключение

Итогом работы является анализ существующих методов генерации трёхмерного окружения, на основе которого был разработан собственный алгоритм для генерации внутреннего трёхмерного окружения для комнат, его планируется использовать для будущей компьютерной игры в жанре «Побег из комнаты». В этой игре описанный алгоритм будет использоваться для реализации механики изменения внутриигрового окружения. Алгоритм не только воссоздаёт правдоподобный интерьер, но и проверяет его на корректность, а также адаптируется под действия пользователя непосредственно в компьютерной игре. Данный алгоритм можно использовать в любых средах, где требуется генерация внутреннего трёхмерного окружения, например, для приложений по продаже недвижимости, где пользователь самостоятельно сможет создать дизайн интерьера желаемой квартиры.

ЛИТЕРАТУРА

1. Журавлев Д. (2023). Процедурная генерация трехмерных окружений для иммерсивных среда. В: Материалы конференции «Информационные технологии и компьютерная графика» (с. 125–132).
2. Иванов А. (2022). Генерация виртуальных сред для иммерсивного обучения: новые подходы и возможности. В: Материалы конференции «Образование и технологии» (с. 258–265).
3. Кузнецов А.В. (2021). Методы процедурной генерации трехмерных окружений для игр. В: Материалы конференции «Игровые технологии и инновации» (с. 187–194).
4. Суханов А.А. (2020). Применение методов процедурной генерации для создания иммерсивных виртуальных сред. В: Материалы конференции «Виртуальная реальность и ее применение» (с. 312–319).
5. Колосов В.И. (2019). Использование искусственного интеллекта для процедурной генерации контента в иммерсивных средах. В: Материалы конференции «Искусственный интеллект и его применение» (с. 145–152).
6. Шаке́р Н.С., Мадле́нер С., Ферна́ндес А. (2015). Обзор процедурной генерации контента в играх. Визуальный компьютер, 31(6), 651–671.
7. Шаке́р Н.С. (2014). Процедурная генерация уровней для компьютерных игр. (Кандидатская диссертация). Университет Портсмута.
8. Анхе́лес М.Х., Ферна́ндес А.М. (2015). Генерация деревьев для процедурной генерации трехмерных сред. В: Процедурная генерация контента в играх (с. 193–210). CRC Press.
9. Перлин К. (1985). Генерация реалистичного рельефа с использованием шума Перлина. В: Заметки к курсу SIGGRAPH '85 (с. 15–22).
10. Анхе́лес М.Х., Ферна́ндес А.М. (2006). Процедурная генерация уровней с использованием клеточных автоматов. Компьютеры и графика, 30(6), 1004–1012.

Журнал «Современная наука: актуальные проблемы теории и практики»