ПРОЦЕДУРНАЯ ГЕНЕРАЦИЯ КОНТЕНТА В ИГРАХ

Бубнова Валерия Алексеевна

студент, Российский технический университет МИРЭА, РФ, г. Москва

С развитием информационных технологий развивается и ниша компьютерных игр. Разработчики создают невероятно реалистичные игры с огромным количеством проработанного контента. Игровой контент - это все, что содержится в рамках игры, исключая программную логику. Зачастую, игровой контент создается вручную. Данный процесс является достаточно трудоемким и занимает большое количество времени. Решением выделенных проблем может послужить процедурная генерация.

Процедурная генерация контента (Procedural content generation, PCG) - автоматическое или полуавтоматическое создание игрового контента на основе алгоритмов. Метод PCG используется разработчиками, когда имеется необходимость создания большого количества контента с уникальными характеристиками. Процедурная генерация позволяет генерировать практически любой вид контента – ландшафты и игровые уровни, текстуры, модели, анимации, игровые правила, персонажей и их поведение, а также звук и музыку.

Опишем некоторые алгоритмы для создания игрового контента. Следующие алгоритмы основаны на плиточном методе генерации. Суть метода - создание контента из небольших изображений либо моделей правильной формы, называемых тайлами.

Процедурная генерация текстур

Одним из самых распространенных алгоритмов процедурной генерации текстур является алгоритм коллапса волновой функции (Wavefunction Collapse). На вход алгоритма подается изображение, составленное из небольшого количества плиток. Алгоритм анализирует входное изображение и определяет соседей каждой плитки на окрестностях фон Неймана. На основе этого анализа алгоритм создает набор правил, по которым будет генерироваться выходное изображение (рисунок 1).

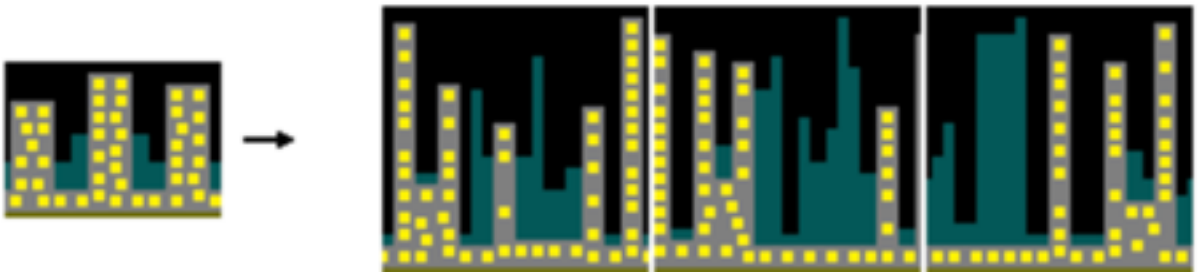

Рисунок 1. Генерация изображений алгоритмом коллапса волновой функции

На вход алгоритма также подается размер сетки, каждая ячейка которой размером с одну плитку. Алгоритм ставит случайно выбранную плитку на ячейку сетки, которая также выбирается случайно. Опираясь на ранее сформированные правила, алгоритм создает плитке

соседа в одном из возможных направлениях. Таким образом, для каждой новой плитки создаются соседи, пока вся сетка не будет заполнена.

Также существует алгоритм коллапса волновой функции на основе ограничений. Он отличается тем, что на вход поступает не картинка, а набор плиток, поэтому правила расположения плиток создаются иным способом. Сначала алгоритм анализирует все плитки и создает их допустимые комбинации при условии, что цвета соседних тайлов должны совпадать вдоль грани касания. Далее алгоритм подбирает к каждой плитке всех вероятных соседей в окрестностях мура на основе ранее выявленных комбинаций.

Недостаток алгоритмов состоит в том, что они могут зайти в тупик, так как предыдущие размещённые тайлы могут снижать количество текущих вариантов до нуля, это особенно вероятно, когда для генерации используется большое количество разных плиток.

Процедурная генерация ландшафта

Для процедурной генерации ландшафта нередко используют шум Перлина - алгоритм градиентного шума (рисунок 2).

Рисунок 2. Шум Перлина

Работает он следующим образом: светлые участки шума Перлина считываются как возвышенности, а темные - как низменности. На их основе алгоритм создает приблизительный рельеф будущей карты. Процесс повторяется несколько раз, пока не получится достаточно достоверный ландшафт с горами, долинами и ущельями.

Недостаток данного алгоритма состоит в том, что генерируемый ландшафт получается довольно «квадратным». С другой стороны, во многих проектах это успешно используется как элемент визуального стиля игры.

Процедурная генерация подземелий

Данный метод генерации основан на клеточном автомате. В разработке игр клеточные автоматы позволяют создавать естественные пещеры (рисунок 9).

Клеточный автомат представляет собой сетку ячеек, каждая из которых имеет состояние и правило для определения того, в какое состояние переходит ячейка на основе ее состояния и ее окружения. Все ячейки имеют одинаковый набор правил, но могут находиться в разных состояниях, в простейшем случае ячейки могут быть «включены» или «выключены». Все переходы между состояниями происходят одновременно; каждая ячейка определяет состояния своих соседей и, следуя определенным правилам, решает, в каком состоянии будет в следующей итерации, только потом все клетки одновременно меняют состояние. Стоит заметить, что в данном методе клетка проверяет соседей в 8 направлениях, иначе говоря, в окрестности Мура.

Итак, перейдем к алгоритму. Алгоритм случайным образом задает состояние каждой ячейки сетки, затем проходит по каждой ячейке и применяет к ячейке правила, например:

- Если ячейка включена и у нее есть хотя бы 4 включенных соседа, она остается включенной;
- Если ячейка выключена, но имеет хотя бы 4 включенных соседа, ячейка становится включенной;
- Ячейка, не подходящие под вышеописанные условия становится либо остается выключенной.

Данный алгоритм исполняется несколько раз, каждую итерацию клетка определяет свое новое состояние. Множество итераций позволяют «сгладить» карту и уменьшить количество несвязанных пещер. На рисунке изображено состояние карты после трех итераций.

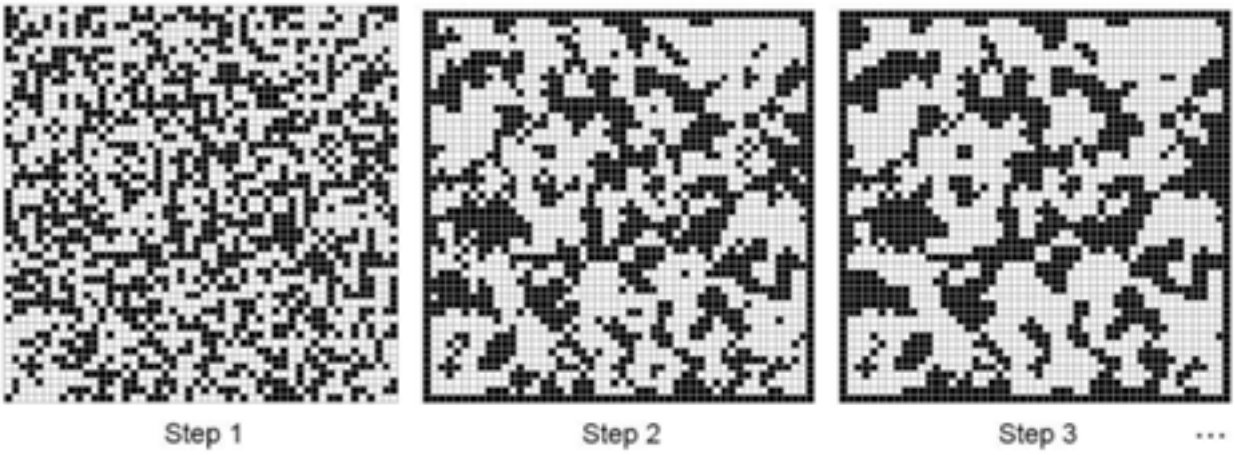

Рисунок 3. Итерации алгоритма на основе клеточного автомата

Стоит заметить, что разработчик сам задает значения, при которых ячейка будет менять состояния, и количество итераций, также он может задать, с какой вероятностью ячейка перейдет в состояние «включена» при задании состояний случайным образом.

Недостаток данного метода заключается в том, что разработчику после генерирования карты необходимо самостоятельно обеспечить некоторые соединения, так как велика вероятность создания разделённых областей.

Автоматизация создания контента с помощью процедурной генерации - поистине гениальное решение. ПГК позволяет создавать игры с огромным количеством контента за минимальные сроки, уменьшать затраты на разработку, так как создание игрового контента частично или полностью перекладывается на программистов, повышается разнообразие контента, а за счет этого повышается и реиграбельность. Но также у процедурной генерации есть и отрицательные стороны. Алгоритмы процедурной генерации трудно полноценно тестировать. Иногда могут проскакивать случаи, шанс появления которых крайне мал, и предугадать появление которых во время разработки алгоритма практически невозможно, поэтому бывает трудно гарантировать, что каждая сгенерированная карта будет играбельной. Генерация контента в играх, упор которых ставится на сюжет, персонажей, предопределенные события и т.д., может стать задачей настолько трудоемкой, что более разумным становится использование ручного подхода.

Несмотря на перечисленные недостатки, ручное создание контента, в основном, уступает процедурной генерации. В любом случае, перед разработкой игры стоит взвесить все «За» и «Против», так как ручное и процедурное создание контента имеют значительные преимущества друг перед другом в разных жанрах игр.

Список литературы:

1. Noor Shaker, Antonios Liapis, Julian Togelius, Ricardo Lopes, and Rafael Bidarra. Constructive generation methods for dungeons and levels // Procedural Content Generation in Games Switzerland, 2016. C. 31-55 (дата обращения 16.03.2022).

2. Procedural generation: [Электронный ресурс]: Режим доступа – свободный: URL: https://www.wikiwand.com/en/Procedural_generation (дата обращения: 16.03.2022).

3. The Constrained Tile Placement Algorithm behind Generate Worlds: [Электронный ресурс]: Режим доступа – свободный: URL: https://ijdykeman.github.io/procedural_generation/2019/11/08/generate-worlds-algorithm.html (дата обращения: 17.03.2022).

4. The Wavefunction Collapse Algorithm explained very clearly: [Электронный ресурс]: Режим доступа – свободный: URL: https://robertheaton.com/2018/12/17/wavefunction-collapse-algorithm/ (дата обращения: 17.03.2022).