ПРОЦЕДУРНАЯ ГЕНЕРАЦИЯ УРОВНЕЙ С ИСПОЛЬЗОВАНИЕМ ДВОИЧНОГО РАЗБИЕНИЯ ПРОСТРАНСТВА

Дунин А. Т.
Кафедра информационных технологий автоматизированных систем, Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: dunin.vit@gmail.com

В докладе рассматривается возможность применения метода двоичного разбиения пространства для процедурной генерации уровней при разработке roguelike игр.

Введение

Существует множество способов генерации карт. Естественно, ни один способ не является универсальным решением, поэтому большинство разработчиков подстраивают выбранные ими способы под свои игры.

При заполнении области объектами (например, комнатами в подземелье) в случайном порядке вы рискуете тем, что всё будет слишком случайным. Результат может оказаться абсолютно бесполезным хаосом. Один из способов решения этой проблемы это использование метода двоичного разбиения пространства (Binary Space Partitioning, BSP).

BSP-деревья можно использовать для создания простейших и наиболее характерных для roguelike карт — прямоугольных комнат, соединённых друг с другом коридорами.

I. Определение BSP-дерева

Чтобы составить представление о том, что такое BSP-дерево, взгляните на рис. 1.

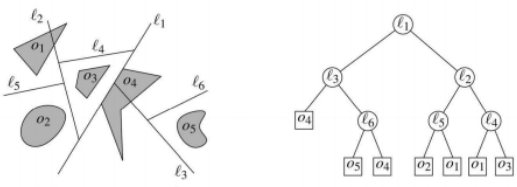

Рис. 1 – Двоичное разбиение пространства и соответствующее дерево

На нем показано двоичное разбиение пространства (BSP) для множества объектов на плоскости, а также соответствующее ему дерево. Как видите, двоичное разбиение пространства получается путем рекурсивного деления плоскости прямой линией: сначала мы делим всю плоскость прямой l_1, затем делим верхнюю полуплоскость прямой l_2, а нижнюю - прямой l_3 и так далее. Разделительные прямые не только разбивают плоскость, но могут и разрезать на части объекты. Деление продолжается, пока внутри каждой области не останется только один объект. Этот процесс естественно моделируется двоичным деревом. Каждый лист дерева соответствует грани окончательного разбиения; часть объекта, попавшая в эту грань, хранится в листе. Внутренние узлы соответствуют разделительным прямым, сама прямая хранится в узле. Если на сцене присутствуют одномерные объекты (отрезки прямых), то может получиться, что объект является частью разделительной прямой, в таком случае эти объекты хранятся во внутреннем узле в виде списка.

Двоичное разбиение пространства, или BSP-дерево для множества S объектов в d-мерном пространстве определяется тогда как двоичное дерево T, обладающее следующими свойствами.

- Если card(S) $\leq$ 1, то T - листовый узел; фрагмент объекта из S (если существует) явно хранится в этом узле. Если обозначить листовой узел v, то множество (возможно, пустое), хранящееся в этом узле, обозначается $S(v)$;
- Если card(S) $>$ 1, то в корне v дерева T хранится гиперплоскость h_v вместе с множеством $S(v)$ объектов, целиком содержащихся в h_v. Левым потомком v является корень BSP дерева T^- для множества $S^- := \{h_v^- \cap s : s \in S\}$, а правым потомком v - корень BSP-дерева T^+ для множества $S^+ := \{h_v^+ \cap s : s \in S\}$.

Размером BSP-дерева называется сумма размеров множеств $S(v)$ по всем узлам v BSP дерева. Иными словами, размер BSP-дерева равен общему числу сгенерированных фрагментов объектов. Если BSP не содержит бесполезных разделительных линии - отсекающих пустое подпространство - то число узлов дерева растет с увеличением размера BSP дерева не быстрее, чем линейно. Строго говоря, размер BSP-дерева ничего не говорит об объеме необходимой для его хранения памяти, поскольку ничего не известно о том, сколько памяти нужно для одного фрагмента. Тем не менее, это хорошая характеристика для сравнения качества различных BSP-деревьев для одного и того же множества объектов.

II. BSP-деревья и алгоритм художника

Допустим, мы построили BSP-дерево T для множества S объектов в трехмерном пространстве. Как с его помощью получить упорядочение по глубине, необходимое для отображения S посредством алгоритма художника? Обозначим P_{view} точку обзора и предположим, что P_{view} расположена выше разделительной плоскости, хранящейся в корне T. Тогда очевидно, что ни один объект, расположенный ниже разделительной плоскости, не может заслонять объекты, расположенные выше нее. Следовательно, мы можем без опаски отображать все объекты (точнее, фрагменты объектов) из поддерева T^- раньше объектов из T^+. Порядок фрагментов объектов в поддеревьях T^+ и T^- рекурсивно определяется таким же образом. Эффективность любого алгоритма, в котором используются BSP-деревья - существенно зависит от размера BSP-дерева. Поэтому разделительные плоскости нужно выбирать так, чтобы фрагментация объектов была минимальна. Прежде чем переходить к разработке стратегии, порождающей небольшие BSP-деревья, нужно определиться с допустимыми типами объектов. Поскольку скорость стоит на первом месте, типы объектов на сцене должны быть простыми: нужно отказаться от искривленных поверхностей и представлять всё полиэдральной моделью. Таким образом, требуется построить BSP-дерево небольшого размера для заданного множества треугольников в трехмерном пространстве [1].

III. Генерация изначальной геометрии уровня

Алгоритм достаточно прост. Изначально создаём прямоугольник, размером со всё игровое поле (рис 2).

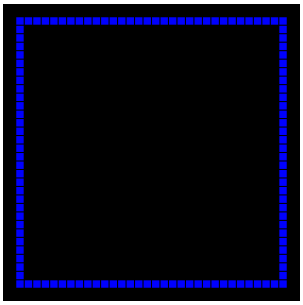

Рис. 2 – Двоичное разбиение пространства и соответствующее дерево

Затем делим его случайным образом на две части — либо горизонтально, либо вертикально. Где будет проходить линия разделения также выбираем случайным образом (рис. 3).

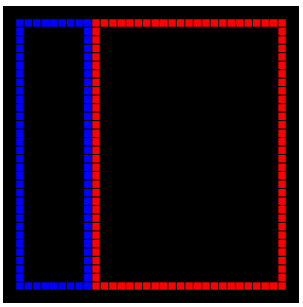

Рис. 3 – Двоичное разбиение пространства и соответствующее дерево

Рекурсивно проделываем тоже самое для новых прямоугольников до некого предела (рис. 4) [2-3].

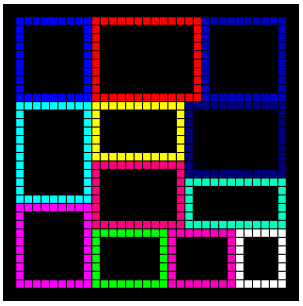

Рис. 4 – Двоичное разбиение пространства и соответствующее дерево

Заключение

BSP-деревья находят применение в различных областях, в частности для генерации уровней в играх. Использование процедурных алгоритмов генерации уровней позволяет экономить время на создание сложных закрытых уровней в играх. Также благодаря процедурной генерации можно привнести разнообразие в игры. Современные алгоритмы позволяют создавать уровни неотличимые от уровней созданных человеком.

Список литературы

1. Mark de Berg. The evaluation and management of bradycardia / Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars//– 2008. – Vol. 386.
2. Информационные системы и сети. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://gamedevelopment.tutsplus.com/tutorials/how-to-use-bsp-trees-to-generate-game-maps–gamedev-12268. Дата доступа : 18.10.2020.
3. Информационные системы и сети. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.gridsagegames.com/blog/2014/06/procedural-map-generation/. Дата доступа : 18.10.2020.