АЛГОРИТМИ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ЛАБИРИНТІВ

Сухоцкая А.В.
Научный руководитель – к.т.н., проф. Чайников С.И.
Харьковский национальный университет радиоэлектроники
(61166, Харьков, пр. Науки,14, каф. Системотехники, тел. (057) 702-13-06)

In this paper described: the algorithms for generating random labyrinths, that can be used to diversify the design of the levels, to adapt the design to subject of your game, to add other objects for the diversification and complexity of the game layout. This is research about different algorithms, analyzing their effectiveness, comparing their opportunities for needs of different customers or tasks. It could be useful in game industry, as well as in real life, for example for architecture and designing of buildings.

Лабіринти в цілому можуть бути класифіковані за сімома різними параметрами. Це: мірність, топологія, мозаїка, маршрутизація, текстура і фокус.

Текстура: клас текстури описує стиль проходів в лабіринті. До цього класу відносяться такі види: Зміщений, Симетричний, Біговий, Однорідний. Фокус: клас показує, що генерація лабіринту може бути двох загальних типів: "Додавання стін" і "Прорізання проходів". Вони відрізняються алгоритмами генерації.

– Додавання стін – алгоритм фокусується на стінах, розпочинає з порожньої області (чи зовнішньої межі) і додає стіни. У реальному житті такі лабіринти – це лабіринти з живоплотів, вітрильників або дерев'яних стін.

– Прорізання проходів – алгоритм фокусується на проходах, запускаються з цілісного блоку і прорізає проходи. У реальному житті такі лабіринти – це тунелі, або канали труб.

Існує безліч алгоритмів створення лабіринтів, з різними особливостями. Ці алгоритми реалізовані за принципом "Прорізання проходів", проте, якщо нічого не забороняє, можна так само реалізувати їх за принципом "Додавання стін".

Алгоритм "Рекурсивний відкат" (Recursive backtracker) [1]: Цей алгоритм швидкий, простий для розуміння, і що легко реалізовується. Потрібно досить пам'яті, щоб зберігати увесь лабіринт, і потрібно місце в стеку пропорційно розміру лабіринту, тому для занадто великих лабіринтів цей алгоритм неефективний. Але для невеликих лабіринтів, він працює відмінно.

Алгоритм Прима (Prim's): Створимо «заготівлю» – лабіринт, в якому усі локації повністю оточені стінами (зрозуміло, далеко від стартової точки в такому лабіринті не підеш). Зіставимо кожній локації змінну-атрибут (відповідно, у нас вийде двовимірний масив атрибутів), яка може приймати

значення Inside (усередині), Outside (зовні) і Border (на межі). Спочатку атрибут кожної локації має бути рівний Outside. Виберемо випадкову локацію в лабіринті і присвоїмо її атрибуту значення Inside. Присвоїмо також атрибутам сусідніх з нею локацій значення Border.

Алгоритм Краскала [2] (Kruskal's): передусім, створимо заготівлю, аналогічну тій, що використали в алгоритмі Прима – лабіринт з усіма можливими стінами. В процесі побудови лабіринту тільки руйнуємо існуючі стіни, тому якщо між якимись двома локаціями існує шлях, він вже нікуди не зникне.

Алгоритм Еллера (Eller's): дозволяє створювати лабіринти, мають тільки один шлях між двома точками. Сам по собі алгоритм дуже швидкий і використовує пам'ять ефективніше, ніж інші популярні алгоритми (такі як Prim і Kruskal), вимагаючи пам'яті пропорційно числу рядків. Це дозволяє створювати лабіринти великого розміру при обмежених розмірах пам'яті. Загальна ідея алгоритму полягає в порядкової генерації, де між кожними двома клітинами рядки при певних умовах (щоб не було циклів і недоступних клітин) випадковим чином виникає стінка.

Алгоритм «Зростаюче дерево» (Growing tree) [1]: це загальний алгоритм для створення лабіринтів з різними текстурами. Потрібно пам'ять для зберігання розміру лабіринту. Цей алгоритм може бути реалізований подібно до алгоритму «Рекурсивний відкат», або реалізований по-іншому подібно до алгоритму Прима.

Алгоритм «Рекурсивний розподіл» (Recursive division) [1]: цей алгоритм подібний до алгоритму «Рекурсивний відкат», так як вони обидва використовують стек, але цей алгоритм фокусується на стінах, а не проходах, тобто використовує підхід «Додавання стін». Цей алгоритм особливо хороший, тому що має фрактальну природу: можна теоретично продовжувати процес нескінченно поступово покращуючи рівні деталізації.

Алгоритм двійкового дерева (Binary tree) [1]: суть алгоритму полягає в тому, щоб прокласти шлях у випадковому напрямку з кожної клітини поля: або наверх, або вправо (залежить від обраного зміщення). Обробляється тільки 1 клітина за одиницю часу, отже, є можливість генерувати лабіринти нескінченного розміру, зберігаючи лише кінцевий результат (лабіринт) без необхідності зберігати будь-яку побічну інформацію.

Література

1. Lee, C.Y., An Algorithm for Path Connections and Its Applications, IRE Transactions on Electronic Computers, vol. EC-10, number 2, 1961 – P. 364-365.

2. Германн Керн. Лабиринт: основные принципы, гипотезы, интерпретации // Керн Г. Лабиринты мира. – СПб.: Азбука-классика, 2007, с. 7-33.