ПРОЦЕДУРНА ГЕНЕРАЦІЯ ДВОВИМІРНОГО ІГРОВОГО ПРОСТОРУ НА ОСНОВІ КОМБІНАЦІЇ АЛГОРИТМУ БІНАРНОГО РОЗБИТТЯ ПРОСТОРУ, АЛГОРИТМУ КРУСКАЛА ТА АЛГОРИТМУ «A*»

Недодаєв В. А.
Науковий керівник – стат. викл. каф. ПІ Новіков Ю. С.
Харківський національний університет радіоелектроніки
(61166, Харків, просп. Науки,14, каф. ПІ, тел. (057) 702-14-46)
e-mail: vladyslav.nedodaiev@nure.ua, тел. (050) 631-25-50

The idea to generate game maps is not new. It comes from role-playing boardgames and aims to provide a unique experience during each game session. There are many algorithms designed to generate various structures and data, find different solutions and calculate optimal paths, but most of them have a rather limited and specific purpose. At the same time, when properly applied and combined, these algorithms together allow you to create an interesting game map for the player, improving the gameplay and increasing the replayability. The given work is devoted to procedural generation of the game environment based on a combination of popular algorithms: binary space partitioning, Kruskal's, «A*».

Концепція процедурної генерації ігрового простору з'явилася в настільних іграх. Згодом вона перейшла в комп'ютерні ігри і сьогодні є дуже поширеною в рольових іграх та іграх з відкритим світом.

Існує чимало алгоритмів, кожний з яких може бути застосований для досягнення різних цілей. Створення цікавих для гравця ігрових мап, в основі яких лежить простір подібний до підземелля з кімнатами та коридорами, вимагає комбінації алгоритмів.

Поширеним рішенням є випадкове розташування приміщень. Перевагами такого підходу є простота реалізації та швидкість виконання. Однак, наявні значні недоліки:

1. Недостатній контроль над процесом генерації.
2. Хаотичність та непередбачуваність результатів.

Імплементуючи та поєднуючи алгоритми ми можемо частково або повністю нівелювати обидва недоліки: зробити процес генерації більш гнучким та налаштовуваним, зменшити хаотичність, контролюючи при цьому вплив фактору випадковості, який потрібен для створення унікального ігрового середовища.

Використання бінарного розбиття простору дозволяє досягти більшого контролю над процесом генерації, рівномірно розподіляючи приміщення по ігровій мапі. В результаті виконання ігрова мапа ділиться на сектори, які утворюють структуру під назвою «BSP-дерево». В реалізації цього алгоритму варто накладати обмеження на відношення розмірів фрагментів, які утворюються під час розділення.

Після генерації секторів необхідно згенерувати кімнату в кожному з них, до тих пір, поки не буде вичерпано ліміт по кількості кімнат, якщо він

є. Вони можуть бути згенеровані випадково або обрані з підготовлених заздалегідь, але не мають виходити за межі сектору.

Наступним кроком буде з'єднання кімнат коридорами. Для цього будемо використовувати алгоритми, які базуються на обробці неорієнтованих зважених зв'язних графів. Отримавши перелік всіх створених приміщень – з'єднуємо ребрами-коридорами ті з них, які знаходяться в сусідніх секторах. Сусідніми вважатимемо ті сектори, які мають спільну сторону. Таким чином, маємо граф, в якому кімнати – це вершини, а коридори – ребра. Кожному ребру присвоюємо вагу використовуючи мангеттенську метрику, яка розраховує відстань між з'єднаними приміщеннями.

Якщо відобразити отриманий граф – він буде одноманітним, оскільки кожна кімната з'єднана з усіма сусідніми. Виділивши мінімальне кістякове дерево за допомогою алгоритму Крускала ми визначимо набір коридорів, при якому всі вершини графа будуть з'єднані. На цьому етапі ми можемо проконтролювати видалення необхідної кількості ребер, які не входять до мінімального кістякового дерева, оскільки їхня наявність є опціональною.

Після цього, кожне ребро, яке залишилося, має бути перетворено на послідовність клітин, які утворюють коридор. Так само як кожне ребро графу з'єднує пару вершин, кожний коридор на ігровій мапі має з'єднувати пару кімнат. Коридори мають початок та закінчення – визначені на мапі точки, між якими має бути знайдено оптимальний шлях. Попередньо можна визначити проміжні точки, через які має проходити коридор. В такому випадку необхідно шукати шлях послідовно від однієї точки до іншої до тих пір, поки не буде з'єднано останню пару.

Ігрові мапи великих розмірів вимагають ретельного підбору алгоритмів. Оптимальними для застосування є алгоритм Дейкстри та алгоритм «A*». Саме «A*» дозволяє налаштувати правила пошуку шляху через задавання евристичних функцій, які допомагають будувати шлях спираючись на метрики, пов'язані з самою мапою або конкретними фрагментами цієї мапи.

Результатом поєднання алгоритму бінарного розбиття простору, алгоритму Крускала та алгоритму «A*» буде гнучкий та налаштовуваний процедурний генератор ігрового простору, який створює зручну для використання, обробки та модифікації структуру ігрової мапи.

Список використаних джерел:

1.Basic BSP Dungeon generation. Дата оновлення: 12.10.2020. URL: http://roguebasin.roguelikedevelopment.org/index.php?title=Basic_BSP_Dungeon_generation

2.Pierre Vigier. Vagabond – Dungeon and cave generation. Дата оновлення: 22.06.2019. URL: https://pvigier.github.io/2019/06/23/vagabond-dungeon-cave-generation.html.

3.Алгоритм Крускала. Дата оновлення: 15.06.2020. URL: https://uk.wikipedia.org/wiki/Алгоритм_Крускала.

4.A* search algorithm. Дата оновлення: 11.02.2021. URL: https://en.wikipedia.org/wiki/A*_search_algorithm