Петрозаводский государственный университет Институт математики и информационных технологий

Кафедра теории вероятностей и анализа данных

09.03.02 – Информационные системы и технологии

Отчет о практике по научно-исследовательской работе

Исследование и сравнение алгоритмов генерации уровней в игровых мирах

Выполнили:

студенты 3 курса группы 22305

О. А. Плугин подпись

И. О. Левицкий подпись

Руководитель:

Р. В. Сошкин, к. т. н., старший преподаватель

подпись

Итоговая оценка

оценка

Петрозаводск — 2024

**СОДЕРЖАНИЕ**

[Определения 3](#_Toc166706851)

# Определения

Алгоритм –

# Введение

## Актуальность исследования

Разнообразие геймплея: В современном мире игроки ожидают увлекательного разнообразного геймплея, и одним из эффективных способов достижения этого является использование процедурной генерации уровней. Различные алгоритмы генерации уровней позволяют создавать уникальные игровые миры при каждом запуске игры. Это способствует увеличению интереса и вовлеченности игроков. Такая вариативность особенно важна для жанров игр, например, rogue like или sandbox, где повторяемость контента может быстро наскучить игрокам.

Удержание аудитории: Удержание игроков является важным фактором успеха игры. Процедурная генерация уровней может значительно увеличить время, проводимое игроками в игре, за счёт разнообразности поставляемого контента.

Экономия ресурсов разработчика: Создание большого количества уникальных уровней вручную занимает значительных финансовых и временных затрат. Алгоритмы генерации уровней позволяют разработчикам сократить эти затраты, автоматически создавая уровни.

## Цель работы

Целью работы является исследование и сравнение различных алгоритмов генерации уровней для выявления эффективности, практичности, применимости и влияния их на игровой процесс.

## Задачи работы

1. Провести обзор и классификацию существующих алгоритмов генерации уровней
2. Разработать и реализовать прототипы нескольких ключевых алгоритмов генерации уровней для демонстрации их работы и сравнения результатов
3. Оценить преимущества и недостатки описанных в работе алгоритмов по нескольким параметрам
4. Провести анализ разработанных прототипов
5. Исследовать, какие из алгоритмов лучше всего подходят

# Теория

# Алгоритмы генерации лабиринтов

## Алгоритм двоичного дерева

### Описание

Алгоритм двоичного дерева невероятно прост.

Для каждой клетки лабиринта он случайным образом выбирает, убрать стену в одном из двух направлений. Любая из пар направлений может быть использована, при условии использования её на всём поле – Север/Запад, Юг/Восток, Юг/Запад, Север/Восток.

Особенность алгоритма заключается в том, что ему не нужно хранить состояние лабиринта в памяти, т.к. по условиям генерации гарантируется доступ в любую ячейку поля. Это позволяет генерировать лабиринты огромных размеров без использования большого объёма памяти.

### Сложность алгоритма и затраты памяти

Размер поля равен **m \* n**.

Сложность алгоритма: **O(m \* n)**

Затраты памяти:

* **m \* n** ячеек поля
* **2 \* m \* n** стен (по 1 на направление)

### Недостатки и преимущества

Преимущества

* Простота реализации
* Скорость работы алгоритма
* Малые затраты памяти
* Возможность генерировать лабиринты больших размеров

Недостатки

* Два прямых коридора по краям лабиринта (для гарантии доступа)
* Сильное диагональное смещение
* Однообразие структуры лабиринта
* Ограниченность в направлениях

### Примеры работы алгоритма

## Алгоритм Sidewinder

### Описание

Алгоритм Sidewinder схож с алгоритмом бинарного дерева. Он так же использует случайные числа для определения решения, но в отличие от более простого алгоритма бинарного дерева, содержит только один сквозной коридор и менее заметное смещение по диагонали.

Для каждой строки создаётся множество, называемое «пробежкой» которое может завершиться при ограничении пробежки в её направлении стеной.

По завершении пробежки случайным образом выбирается стена, разделяющая строки, из сохранённого множества, и удаляется. Это необходимо для сохранения гарантии доступа ко всем ячейкам.

### Сложность алгоритма и затраты памяти

Сложность и затраты остаются равными алгоритму бинарного дерева.

Размер поля равен **m \* n**

Сложность алгоритма: **O(m \* n)**

Затраты памяти:

* **m \* n** ячеек поля
* **2 \* m \* n** стен (по 1 на направление)

### Недостатки и преимущества

Преимущества

* Простота реализации
* Скорость работы алгоритма
* Малые затраты памяти
* Возможность генерировать лабиринты больших размеров

Недостатки

* Слабая запутанность лабиринта
* Ограниченность в направлениях

### Примеры работы алгоритма

## Алгоритм Олдоса-Бродера

### Описание

Начиная с случайной точки в поле, алгоритм блуждает по нему в поисках непосещенных вершин поля, каждый раз случайно выбирая направление. При нахождении непосещенной вершины он убирает стену между этой клеткой и клеткой, из которой пришёл.

# Источники

1. <https://weblog.jamisbuck.org/2011/2/1/maze-generation-binary-tree-algorithm>
2. <https://upcommons.upc.edu/bitstream/handle/2117/328169/memoria.pdf>