

Генератор лабиринтов на языке Python

Кокухина Софья Александровна

Приамурский государственный университет им.Шолом-Алейхема

Студент

Аннотация

Цель исследования – разработка программы на языке Python для генерации случайных лабиринтов и их визуализации в консоли. Проект демонстрирует работу с двумерными массивами, циклами, рекурсией и алгоритмами поиска путей. Реализованный генератор позволяет создавать уникальные лабиринты разного размера, а также находить выход из них с помощью алгоритма поиска в глубину (DFS).

Ключевые слова: Python, лабиринт, генерация, алгоритм поиска, рекурсия, двумерный массив.

Maze generator in Python

Kokukhina Sofya Alexandrovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The purpose of the article is to develop a Python program for generating random mazes and visualizing them in the console. The project demonstrates working with two-dimensional arrays, loops, recursion, and pathfinding algorithms. The implemented generator allows you to create unique mazes of different sizes, as well as find a way out of them using the depth-first search algorithm (DFS).

Keywords Python, maze, generation, search algorithm, recursion, two-dimensional array.

1. Введение

1.1 Актуальность

Задачи генерации лабиринтов часто используются в программировании и математике для демонстрации алгоритмов поиска пути, работы с графами и структур данных. Подобные алгоритмы находят практическое применение в компьютерных играх, при создании процедурно генерируемых миров, в робототехнике для моделирования навигации и в образовательных проектах для наглядного изучения рекурсии и алгоритмов.

Использование Python в данной задаче является оправданным благодаря его простоте и широкому спектру встроенных инструментов для работы со случайными числами, массивами и рекурсивными функциями.

1.2 Обзор исследований

Статья А.М. Титовой и Е.В. Трофименко рассказывает о генерации лабиринтов в игровом движке Unity3D [1]. В описании патента Е.С. Банделя говорится о программе, предназначенной для генерации случайных лабиринтов и решения их с помощью алгоритма поиска пути. Область применения программы: игровая индустрия, разработка ПО [2]. Программа М.В. Ляхнова и Е.Н. Коршуновой позволяет составить и найти самый короткий путь, чтобы выбраться из лабиринта. Программа для ЭВМ обеспечивает сохранение целостности, конфиденциальности и защищённости данных. Программа для ЭВМ способствует предотвращению использования тупиковых веток, тем самым экономя время на прохождение лабиринта. Программа для ЭВМ осуществляет надежное хранение полученной информации [3]. В статье В.С. Володченко рассматриваются различные алгоритмы создания лабиринтов в сфере компьютерных игр [4]. А.В. Коркин рассказал в своей публикации о генерации идеальных лабиринтов с помощью алгоритма эллера [5].

1.3 Цель исследования

Целью исследования является разработка программы на Python, которая автоматически генерирует лабиринт заданного размера и отображает его в консоли.

2 Материалы и методы

В качестве инструмента для разработки программы выбран язык программирования Python версии 3.12.

Алгоритм генерации лабиринта основан на рекурсивном поиске в глубину (DFS). Его суть заключается в последовательном «прорезании» проходов в стенах двумерного массива. На каждом шаге выбирается случайное направление движения, после чего стена между клетками разрушается, и рекурсивный вызов продолжается из новой точки. Такой подход позволяет построить уникальный лабиринт, в котором гарантированно существует путь между любой парой точек.

Подготовка проекта выполнялась в несколько этапов. Сначала в PyCharm создавался новый проект, в котором указывался установленный ранее интерпретатор Python 3.12. Затем в корне проекта создавался файл `main.py`, в который добавлялись все необходимые функции и основной код программы. Структура проекта оставалась минимальной, так как весь функционал генерации и вывода лабиринта реализовывался в одном файле (Рис. 1).

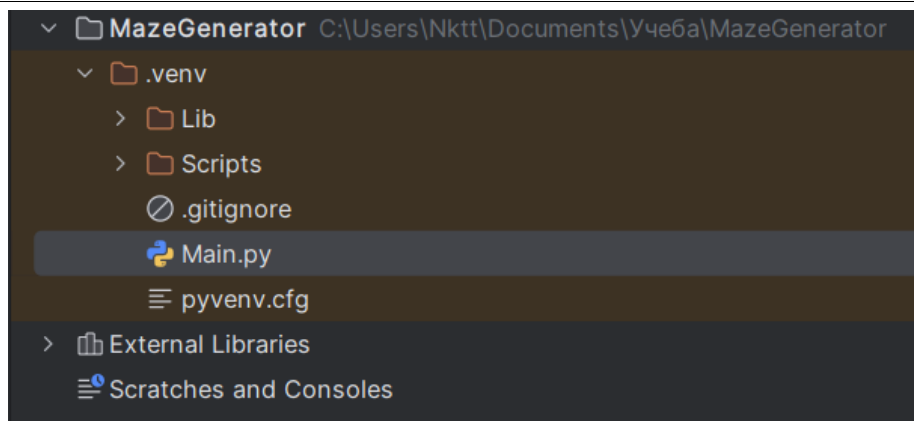


Рисунок 1 – Структура проекта

3 Результаты и обсуждение

Для начала нужно перейти к объявлению и инициализации основных переменных для реализации программы. Создаётся двумерный массив `maze`, заполненный символами `#`, которые обозначают стены. Далее определяются размеры лабиринта `width` и `height`, а также стартовая точка, с которой начинается генерация (Рис. 2).

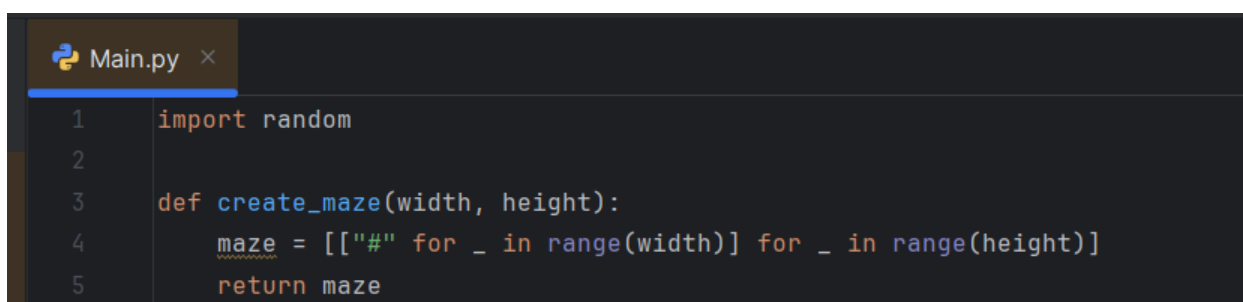


Рисунок 2 – Объявление переменных программы

Следующим шагом реализуется функция `carve_passages(x, y, maze, width, height)`, выполняющая «прорезание» стен. Для каждой клетки случайным образом выбирается порядок обхода направлений (вверх, вниз, влево, вправо). Если в выбранном направлении находится непосещённая клетка, программа разрушает стену между ними и рекурсивно продолжает процесс из новой позиции (Рис. 3).

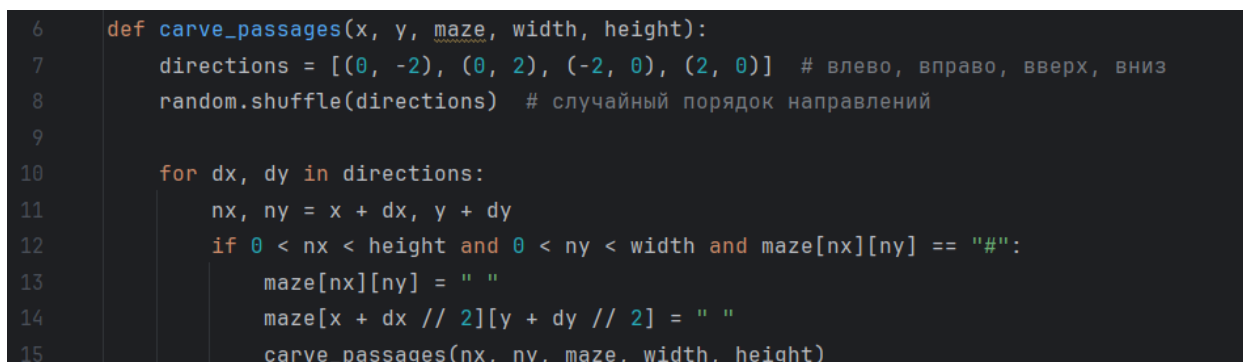


Рисунок 3 – Функция шифрования сообщений

Основная функция `generate_maze(width, height)` (Рис.4) объединяет этапы генерации: создаёт заготовку сетки, открывает стартовую точку и вызывает рекурсивный алгоритм. Дополнительно реализуется функция `print_maze(maze)`, которая выводит результат в консоль в виде текстовой сетки (Рис. 5).

```
16 def generate_maze(width, height):
17     maze = create_maze(width, height)
18     maze[1][1] = " " # стартовая точка
19     carve_passages(x: 1, y: 1, maze, width, height)
20     return maze
21
```

Рисунок 4 – Функция генерации лабиринта

```
21
22 def print_maze(maze):
23     for row in maze:
24         print("".join(row))
25
26 ▶ if __name__ == "__main__":
27     maze = generate_maze(width: 21, height: 21)
28     💡 print_maze(maze)
29
```

Рисунок 5 – Функция запуска программы

После завершения написания кода программа запускается в PyCharm с помощью комбинации клавиш `Shift + F10`. В консоли появляется сгенерированный лабиринт. При каждом запуске программа создаёт новый вариант структуры благодаря случайному выбору направлений. Пример генерации лабиринта размером 21×21 представлен на рисунке (Рис. 6).

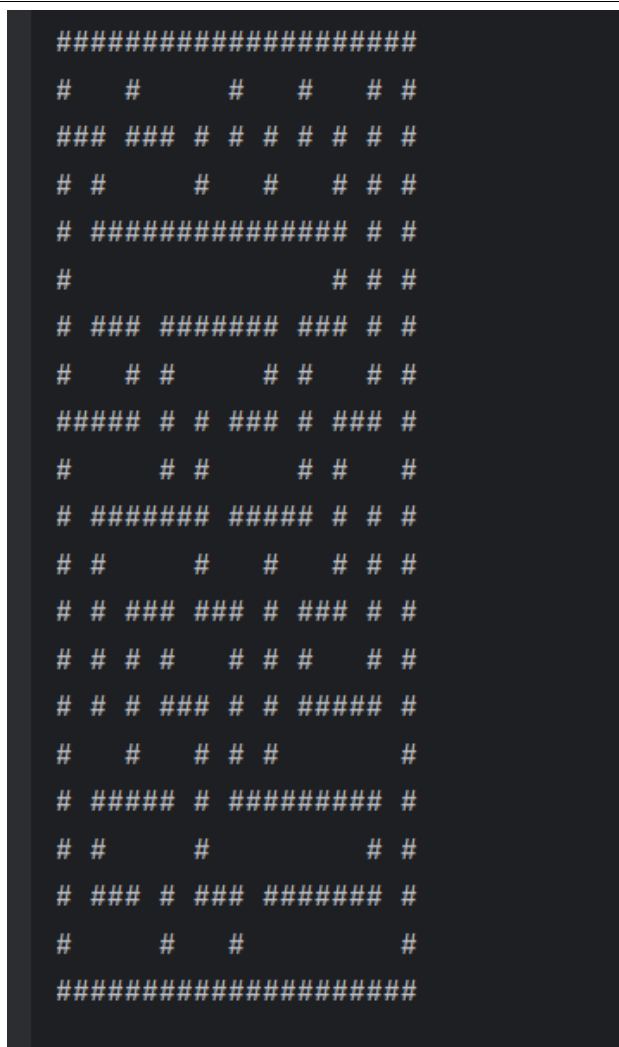


Рисунок 6 – Пример работы программы

4 Выводы

В ходе работы был реализован простой, но наглядный пример генерации лабиринтов на языке Python. Программа демонстрирует принципы работы с двумерными массивами, рекурсией, случайной генерацией и базовыми алгоритмами поиска. Реализация в PyCharm обеспечивает удобство разработки и тестирования.

Разработанная программа успешно создаёт лабиринты различного размера, при каждом запуске выдавая уникальный результат. Возможность изменять параметры генерации делает приложение универсальным и позволяет расширить его функциональность – например, добавив поиск выхода, графическую визуализацию или сохранение результата в файл.

Библиографический список

1. Титова А. М., Трофименко Е. В. Генерация лабиринта на игровом движке Unity3d // Сборник трудов Межвузовской научной конференции молодых ученых и студентов. М.: Научная книга. 2024. С. 908-914.
2. Бандель Е.С. Программа генерации и решения лабиринта. Свидетельство о

- государственной регистрации программы для ЭВМ RU № 2024669502.
3. Ляхнов М. В., Коршунова Е. Н. Программное решение для генерации лабиринта. Свидетельство о государственной регистрации программы для ЭВМ RU №2022663739.
 4. Володченко В. С., Ланцова Д. С., Миронова Т. А. Способы генерации лабиринтов в индустрии компьютерных игр // Вопросы науки и образования. 2019. № 31(81). С. 8-11.
 5. Коркин А. В. Генерация идеальных лабиринтов с помощью алгоритма Эллера // Материалы Всероссийской научно-практической конференции. Кемерово: Кузбасский государственный технический университет им. Т.Ф. Горбачева, 2014. С. 261-262.