ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

Старший преподаватель				Суетина Т. А.
должность, уч. степень, звание		подпись, дата		инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

Процедурная генерация лабиринта

Вариант 5

по курсу: Мультимедийный практикум

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	4128				Воробьев В. А.
			подпись, дата		инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1 Введение

Цель работы: получить навыки написания сценариев на C# и реализации алгоритмов процедурной генерации лабиринтов.

Задание:

1. Задать свой размер лабиринта.
2. Выполнить классификацию полученного алгоритма по измерению, гиперразмерности, топологии, тесселяции, маршрутизации, текстуре и фокусу.

2 Выполнение работы

Был написан сценарий на языке C#, который позволяет задать размер лабиринта и генерировать его с использованием процедурного алгоритма. В процессе работы был использован класс MazeController, который отвечает за отображение лабиринта, и класс MazeGenerator, который содержит логику генерации лабиринта.

В приложении представлен код сценария, написанного в ходе выполнения лабораторной работы. Также исходный код доступен на GitHub (URI - `https://github.com/vladcto/suai-labs/tree/main/6_semester/Мультимедия/3`).

Результат представлен на рисунках ниже.

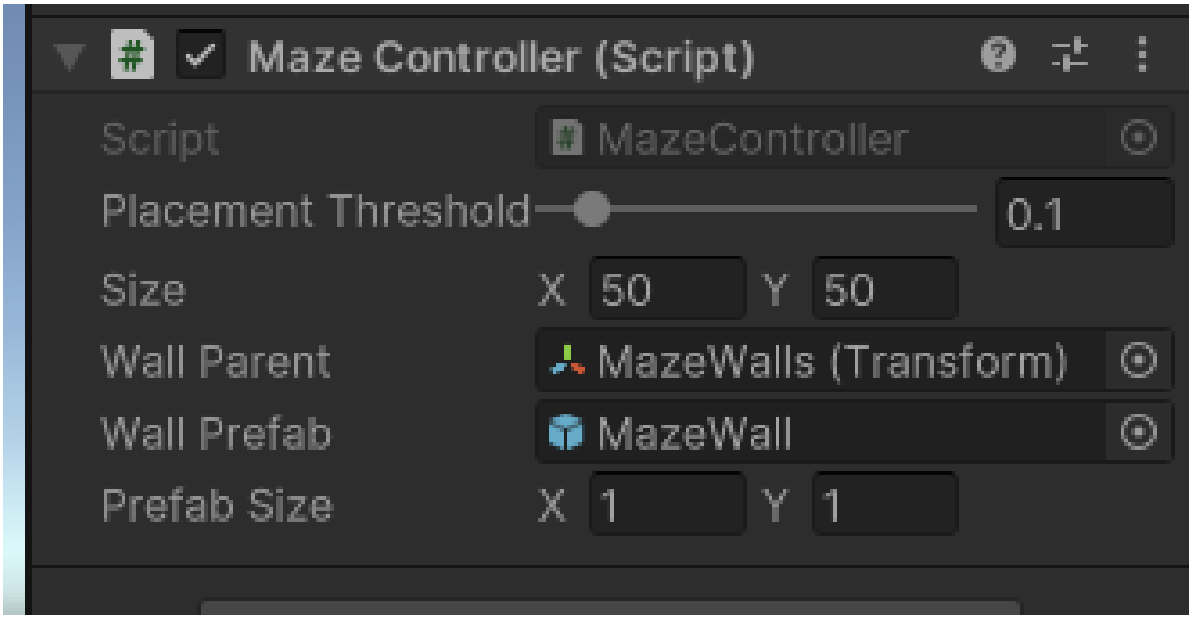

Рисунок 2.1 - MazeController в инспекторе

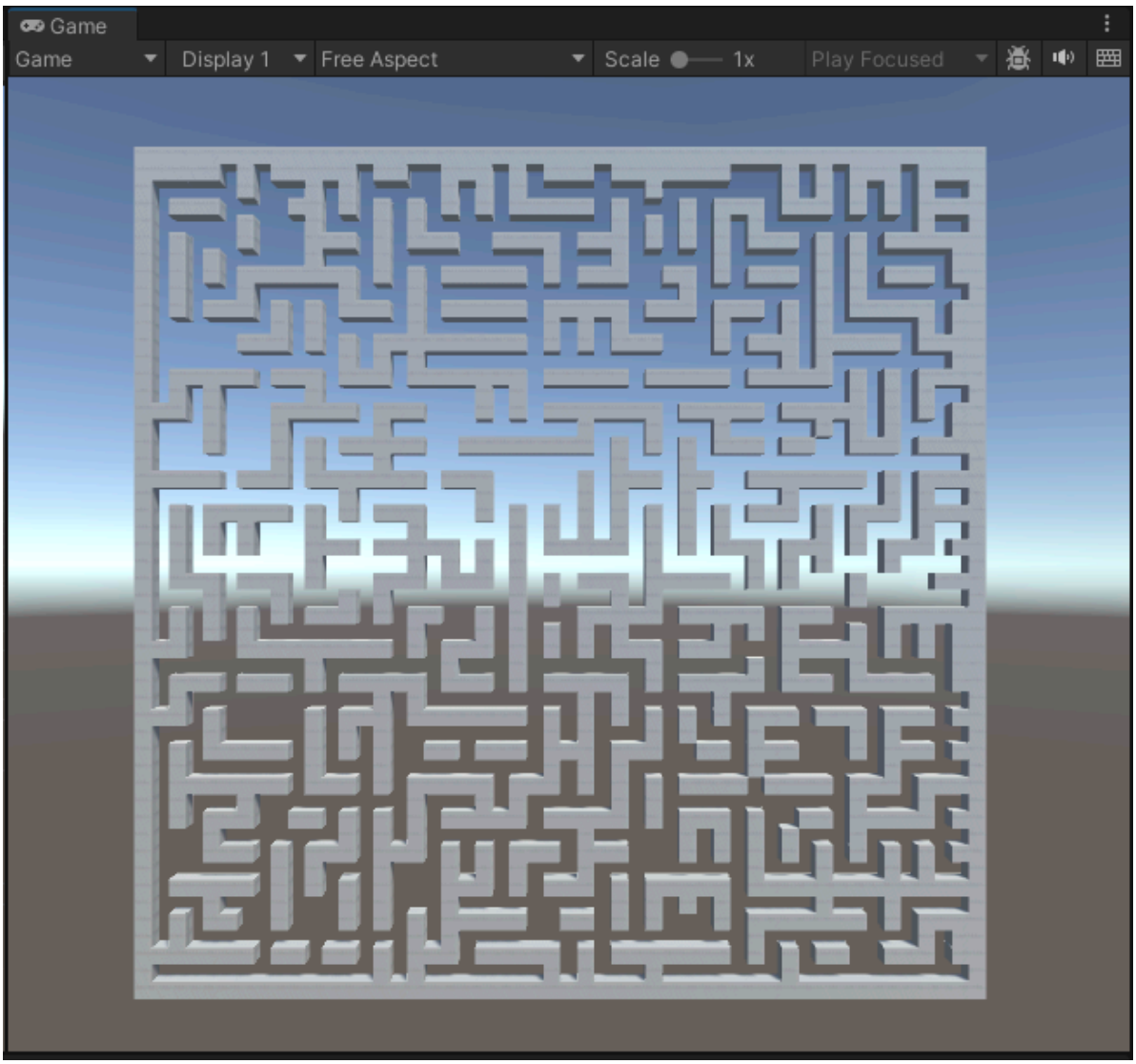

Рисунок 2.2 - Лабиринт 50x50

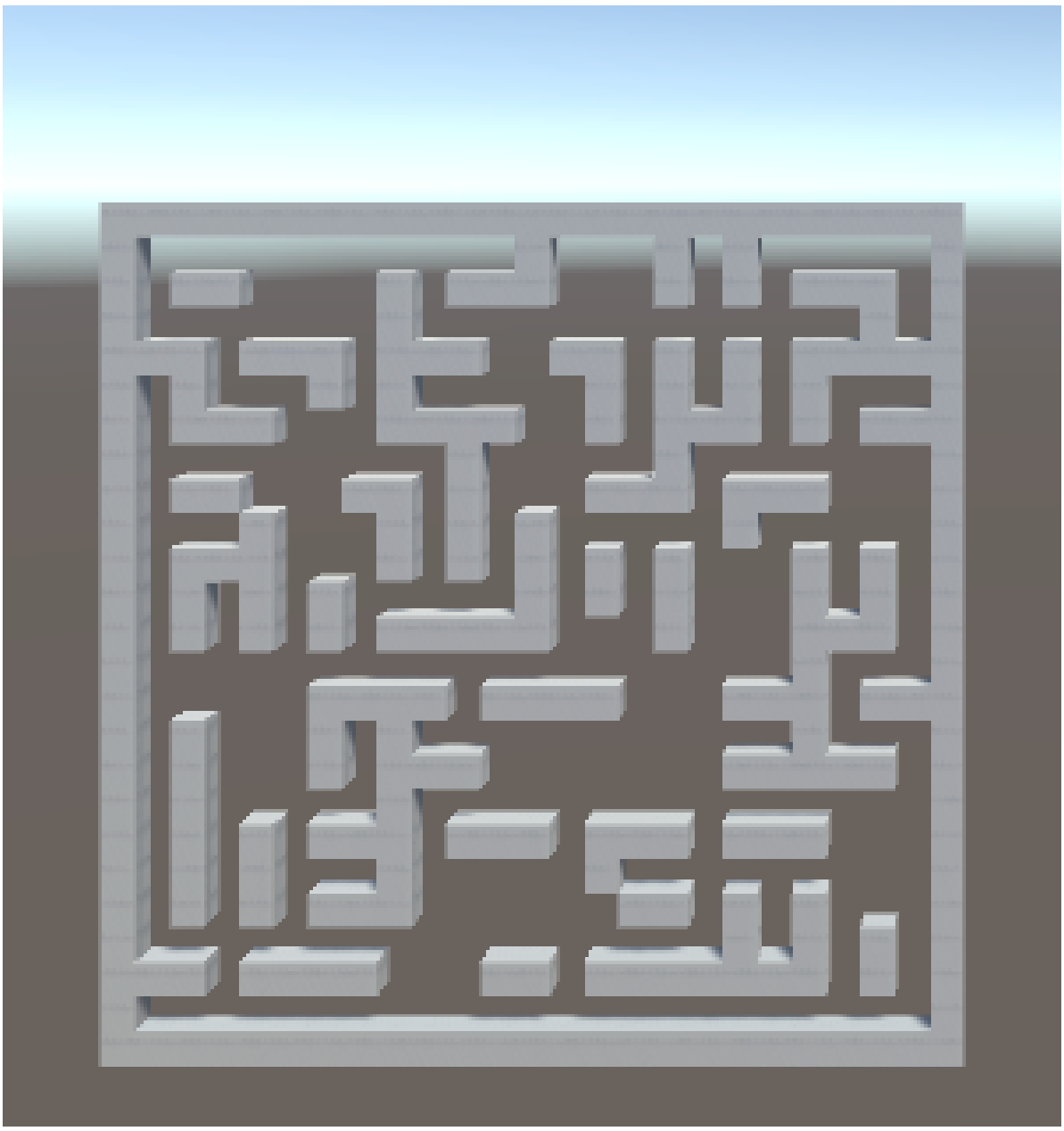

Рисунок 2.3 - Лабиринт 25x25

2.1 Классификация лабиринта

Разработанный лабиринт обладает следующей классификацией:

1) Размерность – двумерный;
2) Гиперразмерность –не-гиперлабиринты;
3) Топология – обычный;
4) Тесселяция – ортогональный;
5) Маршрутизация – разреженный;
6) Текстура – однородный;
7) Приоритет – добавление стен.

3 Вывод

В ходе выполнения лабораторной работы был разработан лабиринт, соответствующий следующей классификации: двумерный, не-гипермаз, обычный, ортогональный и переплетенный, разреженный, однородный с приоритетом добавления стен. Также были получены навыки написания сценариев на C# и реализации алгоритмов процедурной генерации лабиринтов.

ПРИЛОЖЕНИЕ

```
1  namespace DefaultNamespace
2  {
3      using UnityEngine;
4
5      public static class MazeGenerator
6      {
7          public static bool[,] Generate(Vector2Int size, float
               placementThreshold)
8          {
9              var (x, y) = (size.x, size.y);
10             var maze = new bool[x, y];
11             for (var i = 0; i < x; i++)
12             {
13                 for (var j = 0; j < y; j++)
14                 {
15                     TryCreateWall(maze, (i, j),
                           placementThreshold);
16                 }
17             }
18
19             return maze;
20         }
21
22         private static void TryCreateWall(
23             bool[,] maze,
24             (int x, int y) point,
25             float placementThreshold
26         )
27         {
28             var (maxX, maxY) = (maze.GetUpperBound(0), maze.
                   GetUpperBound(1));
29
30             if (point.x == 0 || point.y == 0 || point.x ==
                   maxX || point.y == maxY)
31             {
32                 maze[point.x, point.y] = true;
33
34                 return;
35             }
36
```

```
            if (point.x % 2 == 0 && point.y % 2 == 0 &&
               Random.value > placementThreshold)
            {
                maze[point.x, point.y] = true;
                var a = Random.value < .5 ? 0 : (Random.value
                    < .5 ? -1 : 1);
                var b = a != 0 ? 0 : (Random.value < .5 ? -1
                   : 1);
                maze[point.x + a, point.y + b] = true;
            }
        }
    }
}
```

```
namespace DefaultNamespace
{
    using System.Linq;
    using UnityEngine;

    public class MazeController : MonoBehaviour
    {
        [SerializeField, Range(0, 1)]
        private float _placementThreshold;
        [SerializeField]
        private Vector2Int _size;
        [SerializeField]
        private Transform _wallParent;
        [SerializeField]
        private GameObject _wallPrefab;
        [SerializeField]
        private Vector2 _prefabSize;

        private void Start()
        {
            _Generate();
        }

        private void Update()
        {
            if (Input.GetKeyDown(KeyCode.Space))
            {
                _Generate();
```

```
29              }
30          }
31
32          private void _Generate()
33          {
34              _wallParent.Cast<Transform>().ToList().ForEach(
                    child => Destroy(child.gameObject));
35              var maze = MazeGenerator.Generate(_size,
                    _placementThreshold);
36              for (var x = 0; x < _size.x; x++)
37              {
38                  for (var y = 0; y < _size.y; y++)
39                  {
40                      if (!maze[x, y]) continue;
41
42                      var placeVector = new Vector2(x *
                            _prefabSize.x, y * _prefabSize.y);
43                      Instantiate(_wallPrefab, placeVector,
                            Quaternion.Euler(0, 0, 0), _wallParent
                            );
44                  }
45              }
46          }
47      }
48  }
```