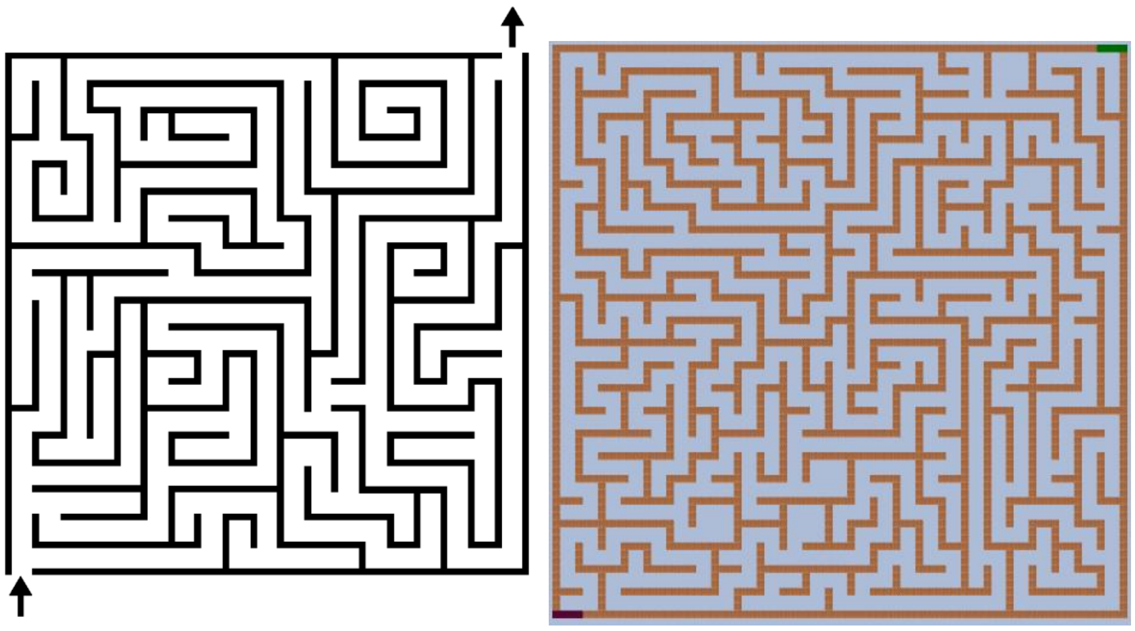
How to use MazeGenerator2D

Description

MazeGenerator is using for creating custom 2D maze. You can use custom wall to create your custom maze. Maze is using backtrack algorithm and cell system: each cell is rectangle and maze size is measured in cells. Also you can customize your maze by using room creation, you can set room count, room width and heght. MazeGenerator can use seed for create same mazes(example: you want save/load generated level and you just can save/load only seed value) and mazeGenerator can create everytime random maze.

To create maze you can use prefabs(maze will automatically created) or use maze skeleton(get wall positions). At this moment walls using rectangle prefabs(walls) to create maze.

Maze have one entrance and one exit, as it shown on the picture 1.



Picture 1. Left picture - entrance and exit, right picture – maze example.

You can get access to maze, when scene is active, example: you want change level in your game. Maze script will help you to solve this task.

- At first step you need to get gameObgect “MazeGenerator”
- At second step you need to get maze script
- And at third step you can use public method’s of MazeScript for your tasks.

Main functions of script:

1. Delete maze
2. Create maze new width and height
3. Create maze with seed(to create same mazes)
4. Set new prefab for wall, start and end walls
5. Get all impasses (cell center positions in array of Vector2)
6. Get all crossroads (cell center positions in array of Vector2)
7. Get all cell centers (cell center positions in array of Vector2)
8. Get way from start to end (cell center positions in array of Vector2)
9. Get gameObject of the start and end wall
10. Verify the position on the wall
11. Add custom rooms with width and height
12. Get all cell centers of rooms (cell center positions in List<Vector2>)
13. SetSeed rules

In demo version you can see how to use maze generator and maze script to create your game!

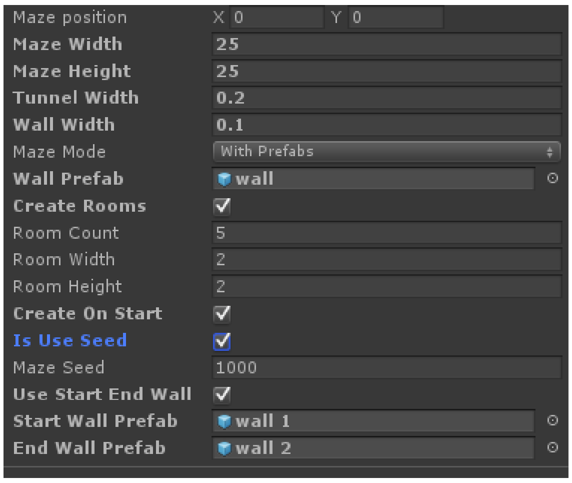
User's manual

If you want add a maze to the scene, you need to use "Maze" prefab(pic.2), you must transfer it to scene from folder "Prefabs".

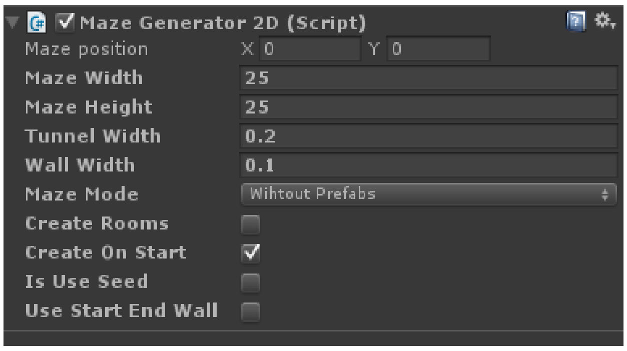
Picture 2. MazeGenerator prefab.

On next step you can chose: use prefabs or get skeleton of maze.

If you chose prefabs mode, you must set rectangle wall prefab, also you can set start and end wall prefabs(pic.3 left picture). You must set TunnelWidth and wallWidth, wallWidth must be less or equal than TunnelWidth. This values need to calculate walls count per cell(formula: nWalls = (tunnelWidth/wallWidth)+2). At right picture (pic.3) Script UI without prefabs, room and seed.



Maze position	X 0 Y 0
Maze Width	25
Maze Height	25
Tunnel Width	0.2
Wall Width	0.1
Maze Mode	With Prefabs
Wall Prefab	wall
Create Rooms	✓
Room Count	5
Room Width	2
Room Height	2
Create On Start	✓
Is Use Seed	✓
Maze Seed	1000
Use Start End Wall	✓
Start Wall Prefab	wall 1
End Wall Prefab	wall 2

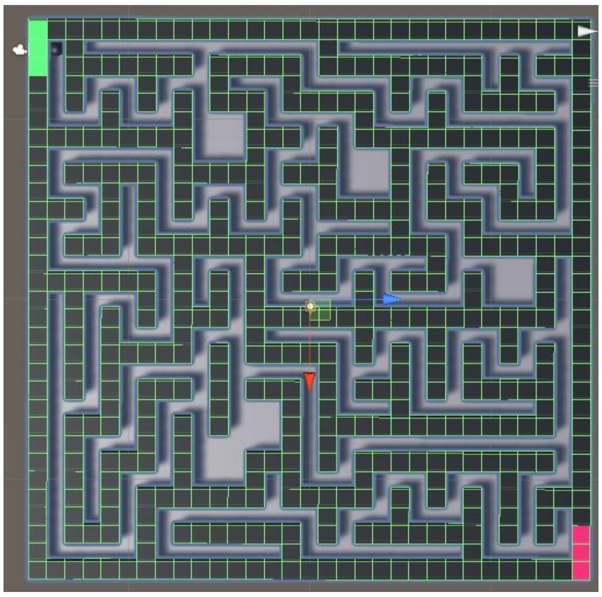


Maze Generator 2D (Script)

Maze position	X 0 Y 0
Maze Width	25
Maze Height	25
Tunnel Width	0.2
Wall Width	0.1
Maze Mode	Wihtout Prefabs
Create Rooms	
Create On Start	✓
Is Use Seed	
Use Start End Wall	

Picture 3. Maze options.

Also you can set maze width and height in cells, maze start position(position maze center, pic.4). If flag "Create maze on start" is set – maze will be created after initialization of the MazeGenerator object. If flag "Create ceiling" is set – ceiling will be created.



Picture 4. Start position.

Scripting

List of public variables:

Prefabs:

```
public GameObject WallPrefab, StartWallPrefab, EndWallPrefab;
```

Maze size in cells:

```
public int MazeWidth = 10, MazeHeight = 10;
```

Position of maze center

```
public Vector2 StartPosition = new Vector2(0, 0);
```

Flags to create maze on start and flag to create ceiling:

```
public bool CreateOnStart = true;
```

Tunnel width and wall width

```
public float TunnelWidth = 1;
public float WallWidth = 1;
```

Seed variables for random creation(if UseSeed is false, all mazes will be random)

```
public bool IsUseSeed = false;
public int MazeSeed = 1000;
```

Room options(if CreateRooms = false, room will be not created)

```
public bool CreateRooms = false;
    public int RoomCount = 5;
    public int RoomWidth = 2;
    public int RoomHeight = 2;
```

MazeGenerator mode – with prefabs or without prefabs

```
public MazeGeneratorMode MazeMode = MazeGeneratorMode.WithPrefabs;
```

List of public functions:

Function for set wall prefab

```
public void SetWallType(GameObject newWall)
```

Function for set start wall prefab

```
public void SetStartWallType(GameObject newGround)
```

Function for set end wall prefab

```
public void SetStopWallType(GameObject newGround)
```

Function returns size of the maze(in cells)

```
public Vector2 GetMazeSize()
```

Function for deleting all maze objects

```
public void DeleteMaze()
```

Function for delete maze and create new maze with new width and height(in cells)

```
public void ClearAndGenerateNewMaze(int newMazeWidth, int newMazeHeight)
```

Function for creating new maze with new width and height(in cells)

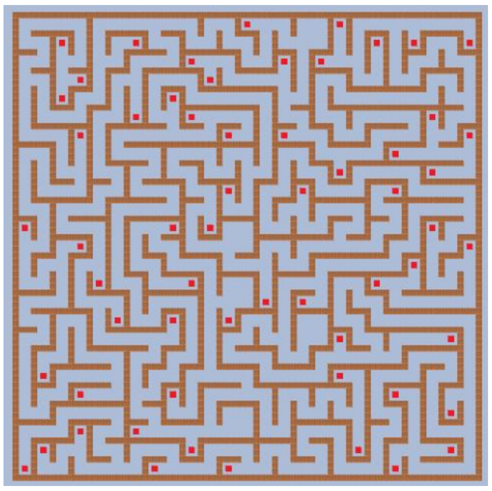
```
public void GenerateNewMaze(int newMazeWidth, int newMazeHeight)
```

Function for creating new maze with new width(in cells), height(in cells), position(in cells) and seed for random.

```
public void GeneratemazeWithSeed(int newMazeWidth, int newMazeHeight, Vector3
position, int seed)
```

Function return list of all impases in maze (cell center positions, pic.5)

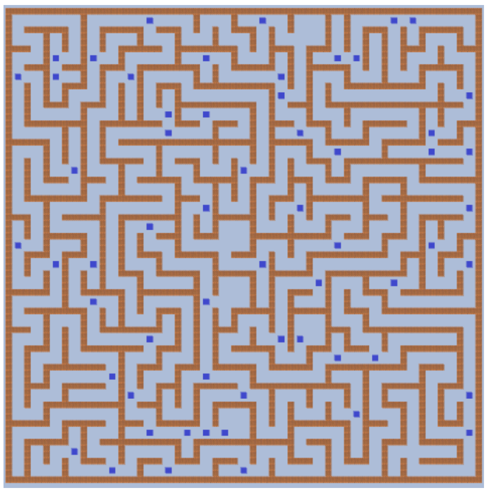
```
public List<Vector2> GetImpasses()
```



Picture 5. Shows all impasses.

Function return list of all crossRoads in maze (cell center positions, pic.6)

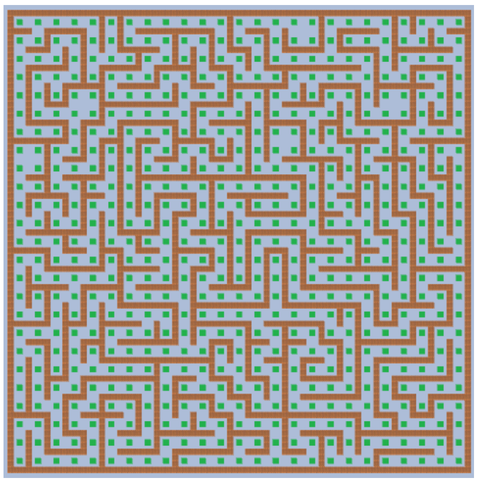
```
public List<Vector2> GetCrossRoads()
```



Picture 6. Shows all crossRoads.

Function return list of all cell centers in maze (cell center positions, pic.7)

```
public Vector2[,] GetGridCenters()
```



Picture 7. Example of way from start to exit.

Function will return start position in maze (cell center position)

```
public Vector2 GetCenterPositionOfStartCell()
```

Function will return end position in maze (cell center positions)

```
public Vector2 GetCenterPositionOfEndCell()
```

Function will return EndWall GameObject(to delete or change properties)

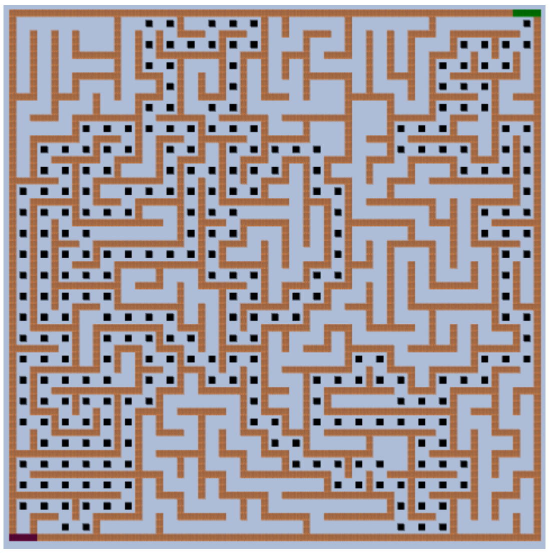
```
public GameObject GetEndWallObject()
```

Function will return StartWall GameObject(to delete or change properties)

```
public GameObject GetStartWallObject()
```

Function will return way(array of Vector2 cell center position, pic.8) from start position to end(exit from maze) position

```
public Vector2[] FindWayToEnd()
```



Picture 8. Example of way to exit.

Set seed for Random function

```
public void SetSeedForRandom(int s)
```

Set using seed random

```
public void SetUseSeed(bool useSeed)
```

Check position, return true if wall is at position

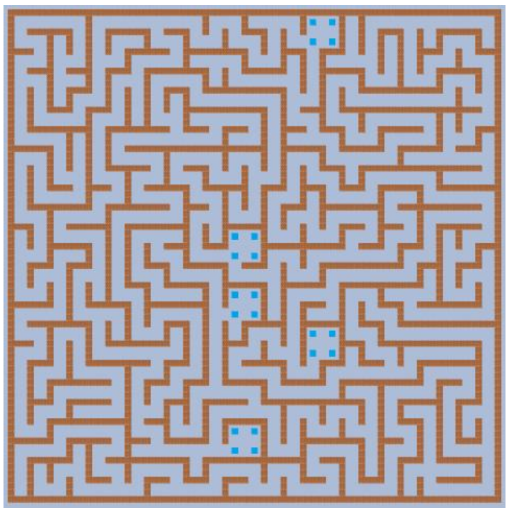
```
public bool IsThereWall(Vector2 position)
```

Get room position function(returns list of all cells from all rooms, pic.9)

```
public List<Vector3[]> GetRooms()
```

Set room creation flag(if false, room will be not created)

```
public void SetRoomCreation(bool create)
```



Picture 9. Room positions

Get walls position function(returns list of Vector3 position of walls to create them)

```
      public List<Vector3> GetWallArray()
```

Help

Next code will help you to get the gameObgect “MazeGenerator”:

```
private GenerateMaze maze;
void Start () {
        mazeScript = GameObject.Find("Maze2D").GetComponent<MazeGenerator2D>();
    }
```

Next code shows how you can get start position:

```
Vector3 startPos = maze.GetCenterPositionOfStartCell();
```

Next code shows how you can create new maze:

```
public void OnGenMaze()
    {
        mazeScript.ClearAndGenerateNewMaze(20, 20);

    }
```

If you have some questions, you can write me on e-mail: **radiomaster71@gmail.com**