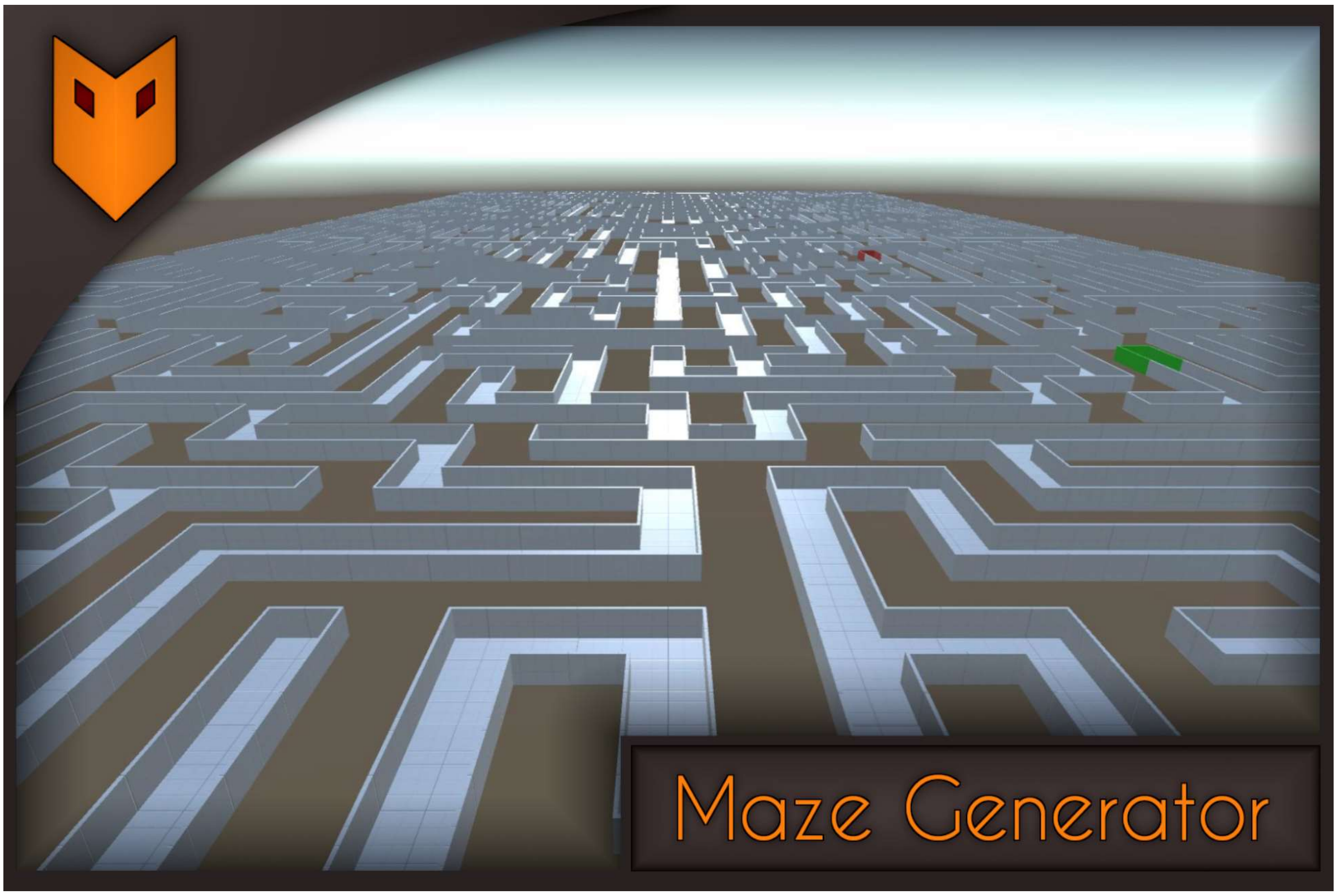
# Finix's Maze Generator

Hello and thanks for checking out my asset for maze generation in the Unity Game Engine. Here you can find all the documentation to get started as well as extending the package to fit your needs.

Inspired by Mazes for Programmers by Jamis Buck this tool comes with a variety of features such as swapable generation algorithms, multiple instatiation methods and modifier components to further customize your desired maze.



# Installation

Finix's Maze Generator is fully self contained and does not depend on any external assets. If you do not need the sample scenes, standard prefabs or the triplanar shader, feel free to exclude the art and resources folder when importing the package.

Keep in mind that for a walls and pillars instatiation the walls will be stretched to fit the maze size, as such the UVs will also be stretched, it is reccomended to use the included standard triplanar shader for the walls or other similar implementation.

## How to import the package

- Open the Unity Editor on version 2020.3.20 or above.
- On the tool bar go to Window ▶ PackageManager.
- On the top left of the Package Manager window select the "Packages: In Project" dropdown menu and switch to "My Assets".
- Select the Finix's Maze Generator Package.
- On the bottom right click on the Download button.
- When the package finishes downloading on the bottom right of the Package Manager window click Import.

# Getting Started

Now that you've installed this package, on the Assets folder you can go to Finix's Maze Generator ▶ Scenes ▶ Sample Scene and mess around with the maze already pre-built in the scene.

## Table of Contents

## Using the package

### Adding a Maze into a scene

- To add a maze in another scene go to the tool bar on top of the Editor and selecting Tools ▶ Maze Generator ▶ Quad Maze.
- Alternitavely you can go to the Hierarchy window and create a new Empty Object and on the inspector window of that object add a new Quad Maze Component, by using this method you need to manually assign what prefabs you want for the different maze parts and Generation methods. When all of that is set-up click the Generate Maze button.

### Customizing your Maze

Upon inspecting the Quad Maze component on the Inspector window you might notice a bunch of parameters to tweak and change. To see more information about each one check the documentation refering to the QuadMaze class.

You're main concern now should be setting up your desired prefabs. To do so first we need to talk about the two generation types:

- Walls Type: Generates only the walls and pillars of the maze, to use it simply set the Generation Type parameter to Walls Type and add a prefab to the Wall Prefab Slot and

the Pillar Prefab in the bottom.
- Prefab Type: This one is a more involved process of generating a maze and requires some steps in order to work properly, however once your done, you can have more customization by adding a list of tiles that will be randomly selected per Cell and Modifiers that can help further with the variety and gameplay balance of the maze. To learn more about this Genaration Type check this next page on how to Setup Your Own Prefabs.

By doing this you can now admire your very own procedurally generated maze and change it's layout with a few clicks of a button. You can leave it static and manually adjust it to your liking or toggle the Generate On Start checkbox to generate a new map layout each time the scene loads.

## Adding Modifiers

Modifiers are Components that can change both the Layout and Tile generation of the Maze. To use one of the already built modifiers (EntranceExitModifier, UniqueTileModifier):

- Select the Maze game object and, on the inspector window, click the Add Component button.
    - Alternitavely you can search for the script in the Project window and drag it into the Inspector window.
- Search for your desired modifier and click on it.
- Assign the correct parameters, which will very from modifier to modifer. Check the scripting reference for more info if needed.
- Click create maze.

# Setting Up Your Own Prefabs

So, you want to add your own prefabs to the maze? Here's a step-by-step guide:

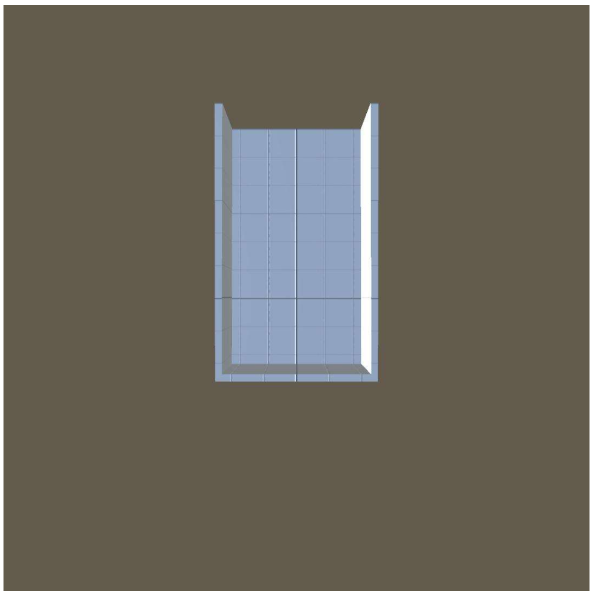
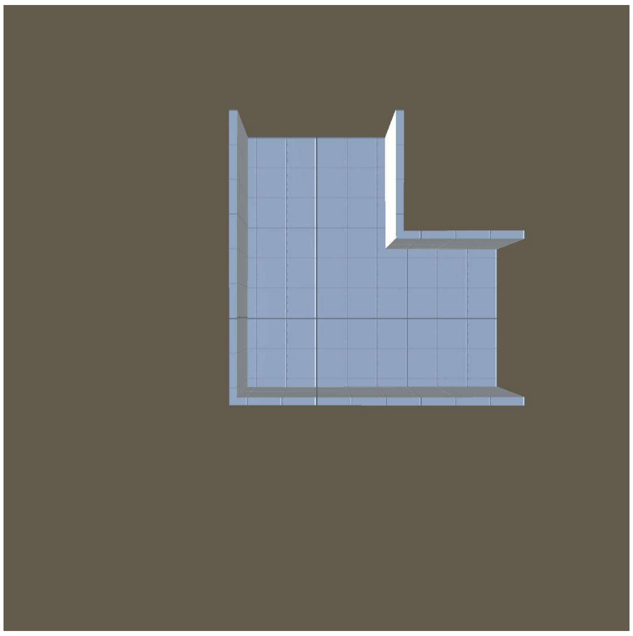
## Table of Contents

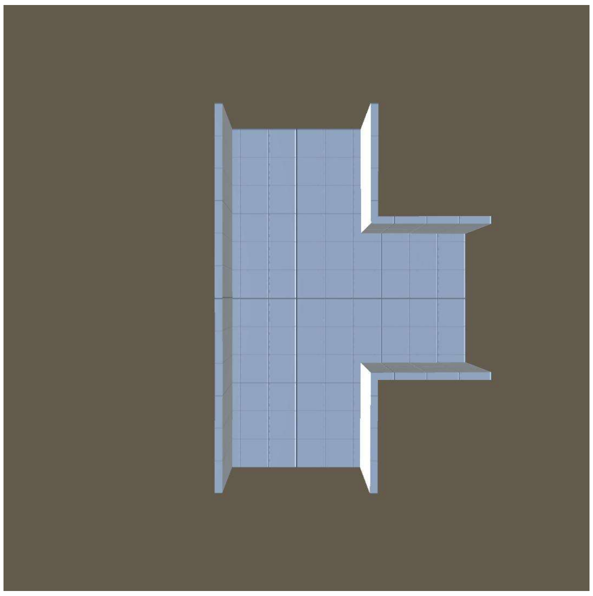
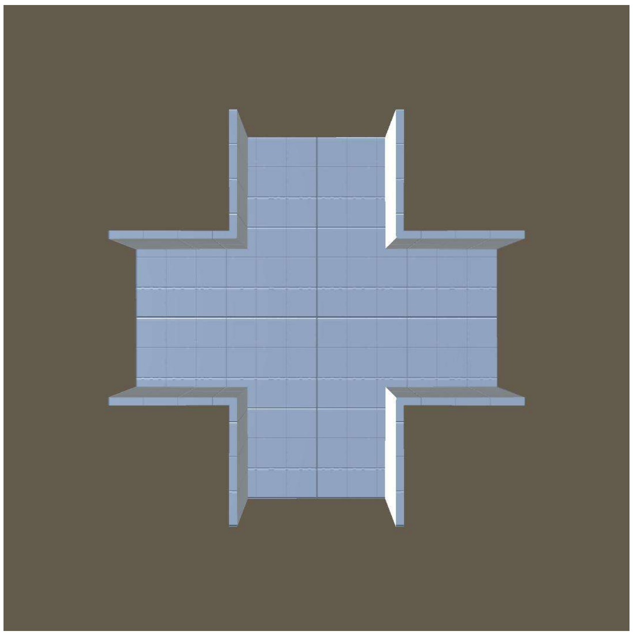
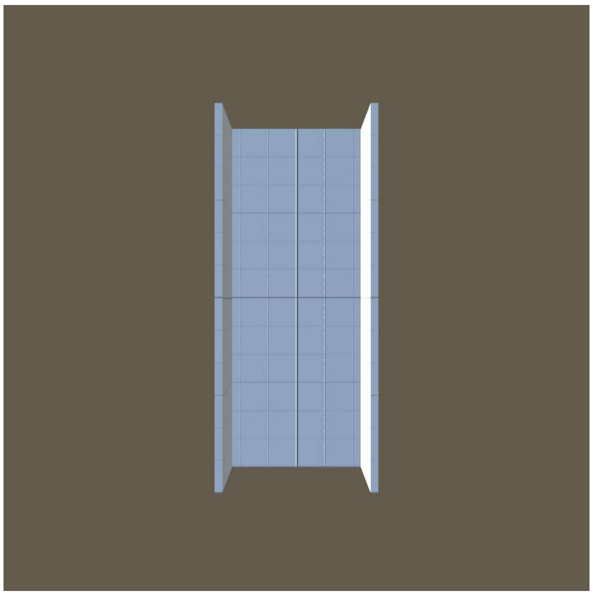
## Step 1: Get modular maze parts

In order to generate a prefab maze, first you need to have to have at least one pre-built prefab for each scenario.

So either pick up your chosen 3D modelling software or buy a modular maze pack from the asset store. In this example I'll be using the standard pieces that come with this asset.

By scenario, we're talking about how many paths a Cell can have to its neighbouring cells and each unique pattern it creates. In the case of a Quad Maze there are 5 unique scenarios possible:

| | |
|---|---|
| Dead End  | Corner  |

| | |
|---|---|
| T-Section  | Crossroads  |
| Corridor  | |

## Step 2: Converting the Meshes to prefab

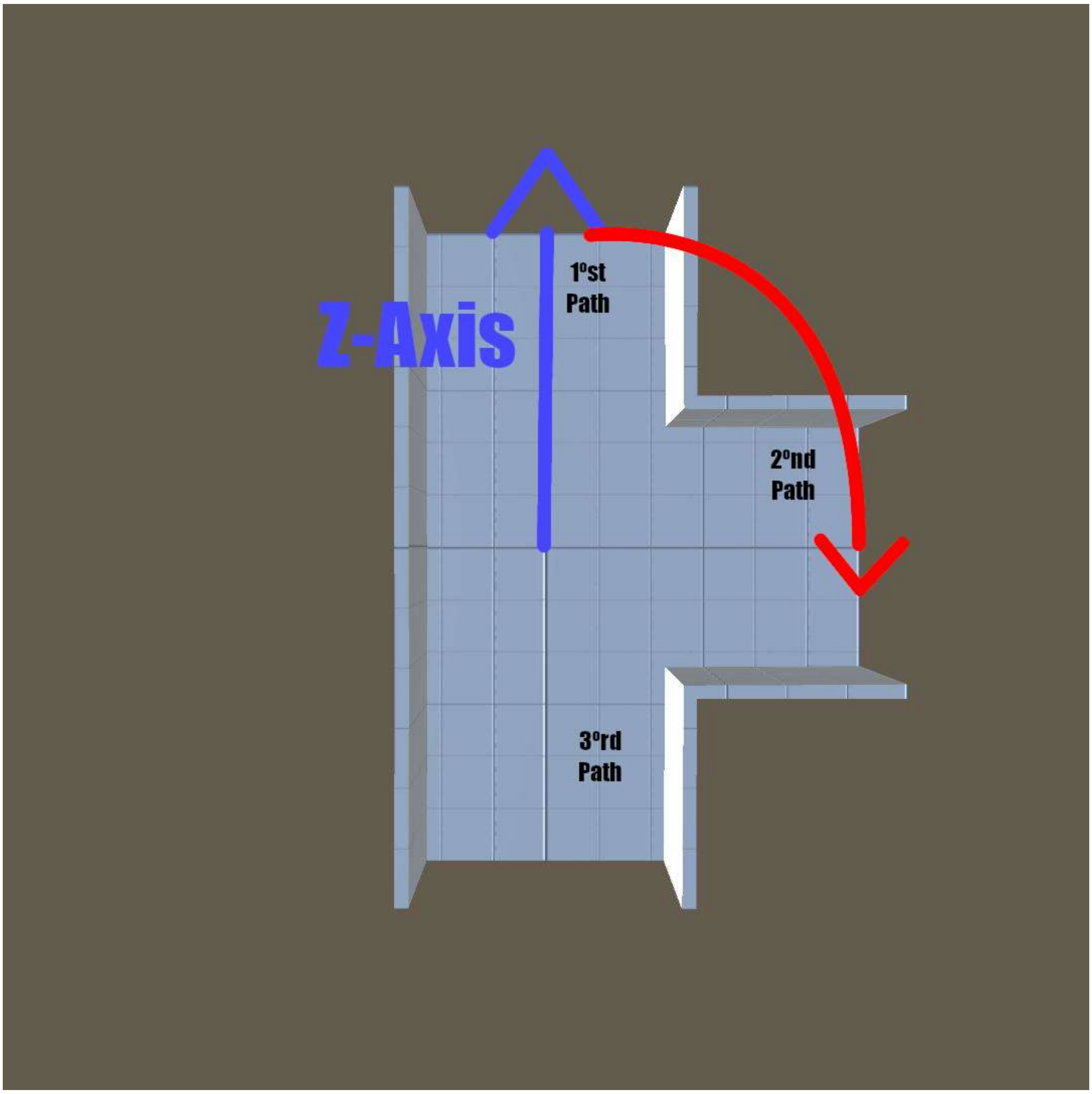
Now that you have the basic mesh pieces it's time to convert them into prefabs (if you're using an asset from the asset store, chances are the pieces are already in a prefab somewhere):

- First import your mesh file (.fbx, .obj, .blend, etc...) into the project by dragging the file into the editor.
- Next drag the piece you're converting to a prefab into the scene, it will create a game object with the piece name.
- Right click the object and select **unpack completely**.

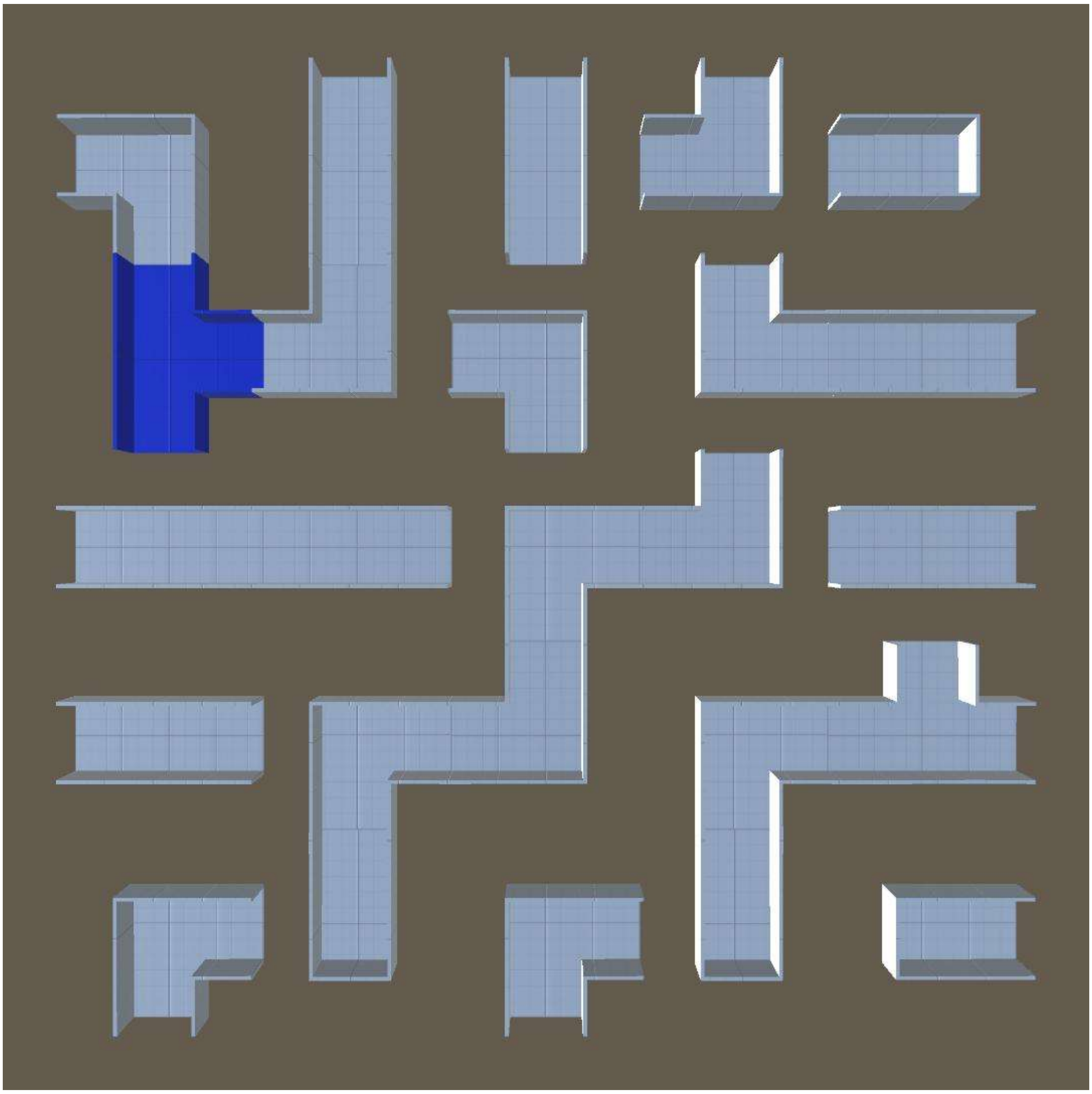
- Since these are game objects, you can add colliders, scripts for game logic, visual props as a child of this object, enemy or item spawns and even make some duplicates with small differences to make your maze more diverse and less boring.
- Finally drag the object to the Project window.

## Important Note: Rotate the prefabs correctly

It's important that the prefabs are rotated in a certain way, the value of the base Y rotation is not important but the paths must be like the images with the first path starting in the **Z+ axis** and each next path going in a **clock wise** direction like the images above are laid out:



Not doing so will result in some glitches when generating the maze like so:



# Package Contents

## Folder Structure

```
└── FinixMazeGenerator/
    ├── Art/
    │   ├── Materials/
    │   │   ├── DefaultMaze.mat
    │   │   ├── Entrance.mat
    │   │   ├── Exit.mat
    │   │   └── Unique.mat
    │   ├── Textures/
    │   │   ├── Grid_01_BaseMap.png
    │   │   ├── Grid_01_Emissive.png
    │   │   ├── Grid_01_Normal.png
    │   │   └── Grid_02_BaseMap.png
    │   ├── quad_maze_parts.fbx
    │   └── StandardTriplanar.shader
    ├── Icons/
    │   └── Maze_Icon.png
    ├── Resources/
    │   └── Prefabs/
    │       ├── Corner.prefab
    │       ├── Corridor.prefab
    │       ├── Crossroads.prefab
    │       ├── Deadend.prefab
    │       ├── Entrance.prefab
    │       ├── Exit.prefab
    │       ├── Intersection.prefab
    │       ├── Pillar.prefab
    │       └── Wall.prefab
    ├── Scenes/
    │   └── SampleScene.unity
    ├── Scripts/
    │   ├── Core/
    │   │   ├── Cells/
    │   │   │   ├── Cell.cs
    │   │   │   └── QuadCell.cs
    │   │   ├── Grids/
    │   │   │   ├── Grid.cs
    │   │   │   └── QuadGrid.cs
    │   │   ├── Algorithms.cs
    │   │   └── Distances.cs
    │   ├── Editor/
    │   │   ├── MazeEditor.cs
    │   │   └── StandardTriplanarInspector.cs
    │   ├── EntranceExitModifier.cs
    │   ├── IMazePrefabModifier.cs
    │   ├── QuadMaze.cs
    │   └── UniqueTileModifier.cs
    ├── FinixMazeGenerator.asmdef
    ├── ThirdPartyNotcice.md
├── README.md
└── Documentation.pdf
```

# FinixMakesGames.MazeGenerator

This section refers to the first layer between the Unity Engine and the one you'll probably be working more close with, be it to automate certain Maze settings or creating your own Modifiers.

## Table of Contents

# Maze

class in FinixMakesGames.MazeGenerator / Inherits from:MonoBehaviour

## Description

An abstract class meant to store common data between the different types of mazes.

## Enums

| Enum | Description |
|---|---|
| GenerationType | An enum used in the Inspector window to select the generation method used when creating a maze |
| AlgorithmType | An enum used in the Inspector window to select the algorithm used when creating the layout of the maze |

# QuadMaze

class in FinixMakesGames.MazeGenerator / Inherits from:Maze

## Description

A Unity component capable of creating rectangular mazes.

## Properties

| Property | Description |
|---|---|
| seed | A string value used to determine the layout of a maze, leaving it empty will generate a random seed |
| cellSize | The number of units that a single cell will ocupy in a scene. |
| rows | The number of Cells created along the local z-axis. |
| collumns | The number of Cells created along the local x-axis. |
| generationType | An Enum used to determine whether the maze will be created with prefab tiles, or with simple walls and pillars |
| algorithm | The algorithm used to create the Maze layout |
| --- | --- |
| corridorObjects | A list of prefabs used in when generating the prefab with tiles, will select a random tile from this list every time it encounters a Cell with a corridor layout. |
| cornerObjects | A list of prefabs used in when generating the prefab with tiles, will select a random tile from this list every time it encounters a Cell with a corner layout. |
| intersectionObjects | A list of prefabs used in when generating the prefab with tiles, will select a random tile from this list every time it encounters a Cell with a intersection/t-section layout. |
| crossroadObjects | A list of prefabs used in when generating the prefab with tiles, will select a random tile from this list every time it encounters a Cell with a crossroad/4-way layout. |
| deadEndObjects | A list of prefabs used in when generating the prefab with tiles, will select a random tile from this list every time it encounters a Cell with deadend. |

| Property | Description |
|---|---|
| --- | --- These can be set to empty if the maze is being generated with the walls and pillars |
| wallPrefab | A prefab used as the walls when generating the maze with the walls and pillars method. Note: this prefab will be stretched to fit the cellSize. |
| pilarPrefab | A prefab used as the pillar when generating the maze with the walls and pillars method. |
| --- | ---These can be set to empty if the maze is being generated with the tile prefabs |
| generateOnStart | If set to true, it will generate a new maze every time the scene loads on the Start() frame. |
| --- | --- |
| grid | A QuadGrid class with the maze layout information |
| uniqueTiles | A dictionary of Cells and Prefabs used for Modifiers and the prefab tiles method. When detecting a cell contained in the dictionary it will spawn the corresponding prefab instead of a random one. |

## Public Methods

| Method | Description |
|---|---|
| CreateMaze | Generates the maze |
| ResetMaze | Destroys every child object of this maze object |

# Modifiers

This section refers to the modifiers feature of the asset. Modifiers are extra components the user can add into the maze game object and will automatically be picked up by the Maze component. These modifiers can either change the Maze at a layout level (changing the Grid information) or by selecting special cell and generating a special prefab instead of the selected ones.

Currently this tool only has a feature for prefabs modifiers and comes with 2 pre-built modifiers to test the feature. To learn more about making your own modifiers check the IMazePrefabModifier interface.

## Table of Contents

# IMazePrefabModifier

interface in FinixMakesGames.MazeGenerator

## Description

An interface used to create Maze Modifiers.

## Public Methods

| Method | Description |
|---|---|
| ApplyModifier | Adds one or multiple <Cell,GameObject> pairs to the uniqueTiles property in the given QuadMaze. |

## Creating your own Prefab Modifiers

- Create a new C# Script in the editor.
- Keep the Monobehaviour inheritance, so that you can later add it to the Maze game object.
- Make the scipt inherit from the IMazePrefabModifier interface.
- Implement the ApplyModifier function and other extra logic you might need, keep in mind that this function needs:
    - To read from the Maze Grid the Cells you intend to modify.
    - The prefabs you wish to add in those Cells instead of the default ones.
    - Acess the uniqueTiles Dictionary in the QuadMaze reference and insert a pair of that Cell and Prefab.
- Finally test the modifier by adding it has a component to the same game object that has the QuadMaze component and hitting the Create Maze button.

# EntranceExitModifier

class in FinixMakesGames.MazeGenerator / Inherits from:MonoBehaviour, IMazePrefabModifier

## Description

A modifier designed to add a pair of prefab tiles. This modifier will always pick the pair most further apart possible.

## Properties

| Property | Description |
|---|---|
| entrancePrefab | The tile used for the entrance cell. |
| entranceLinks | How many links/paths the entrance prefab is supposed to have. |
| exitPrefab | The tile used for the exit cell. |
| exitLinks | How many links/paths the exit prefab is supposed to have. |

## Inhereted Members

## Public Methods

| Method | Description |
|---|---|
| ApplyModifier | Adds one or multiple <Cell,GameObject> pairs to the uniqueTiles property in the given QuadMaze. |

# UniqueTileModifier

class in FinixMakesGames.MazeGenerator / Inherits from:MonoBehaviour, IMazePrefabModifier

## Description

A modifier designed to select candidate Cells and chooses a random one to apply a choosen prefab.

## Properties

| Property | Description |
|---|---|
| tilePrefab | The tile used for the unique cell. |
| tileLinks | How many links/paths the entrance prefab is supposed to have. |

## Inhereted Members

## Public Methods

| Method | Description |
|---|---|
| ApplyModifier | Adds one or multiple <Cell,GameObject> pairs to the uniqueTiles property in the given QuadMaze. |

# FinixMakesGames.MazeGenerator.Core

This section provides information about every class located in the namespace FinixMakesGames.MazeGenerator.Core . The Core refers to every class that does not need to interact with the Unity Engine and simply handles the most basic Maze logic and Data. Said data wil later be read by the FinixMakesGames.MazeGenerator classes that will handle instatiation of the Maze into a Unity Scene

## Table of Contents

# Cell

class in FinixMakesGames.MazeGenerator.Core

## Description

An Abstract Class that represents the basic unit of a maze. Similar to a node in a graph, the Cell is meant to represent a point in a maze Grid and it stores how that point connects to other neighbouring points/Cells.

## Properties

| Property | Description |
|---|---|
| links | Array of all Cells linked to this Cell. |

## Public Methods

| Method | Description |
|---|---|
| Link | Links this Cell to another Cell, making a path between them. |
| UnLink | Removes a link from a choosen Cell to this one. |
| IsLinked | Check if a Cell is linked to this one. |
| Neighbors | Returns a list of all neighboring cells whether they're linked or not. |
| CellDistances | Returns a Distances class with all paths and their relative distance to this Cell. |

# QuadCell

class in FinixMakesGames.MazeGenerator.Core / Inherits from:Cell

## Description

Class that represents the basic unit of a QuadGrid and by extension a QuadMaze.

## Constructor

| Type | Result |
|---|---|
| QuadCell | Creates a new cell with the given row and collumn components, sets the neighbouring cells to null. |

## Properties

| Property | Description |
|---|---|
| row | The position of the Cell within the QuadGrid on the horizontal axis. |
| collumn | The position of the Cell within the QuadGrid on the vertical axis. |
| north, east, west, south | References to each of the four neighbouring Cells relative to this one. |

## Inherited Members

## Properties

| Property | Description |
|---|---|
| links | Array of all Cells linked to this Cell |

## Public Methods

| Method | Description |
|---|---|
| Link | Links this Cell to another Cell, making a path between them. |
| UnLink | Removes a link from a choosen Cell to this one. |
| IsLinked | Check if a Cell is linked to this one. |
| Neighbors | Returns a list of all neighboring cells whether they're linked or not. |
| CellDistances | Returns a Distances class with all paths and their relative distance to this Cell. |

# Grid

class in FinixMakesGames.MazeGenerator.Core

## Description

An Abstract Class use for various types of Maze Grids that represents a Maze in raw data form, without any graphics.

## Public Methods

| Method | Description |
|---|---|
| RandomCell | Returns a random Cell from the Grid. |
| Size | Returns the number of total Cells present in this Grid. |
| DeadEnds | Returns a list of Cells that have only one link to another cell. |

# QuadGrid

class in FinixMakesGames.MazeGenerator.Core / Inherits from:Grid

## Description

A class that represents a rectangular Maze in it's raw data form without any graphics.

## Constructor

| Type | Result |
|---|---|
| QuadGrid | Creates a new grid with the given row and collumn components, creates a new collection of cells and assings it's neighbors. |

## Properties

| Property | Description |
|---|---|
| cells | A double Array of QuadCell that stores all Cells present in this grid . |
| distances | A class Distances variable that can store a collection of paths. Null by default. |

## Inhereted Members

## Public Methods

| Method | Description |
|---|---|
| RandomCell | Returns a random Cell from the Grid. |
| Size | Returns the number of total Cells present in this Grid. |
| DeadEnds | Returns a list of Cells that have only one link to another cell. |
| ToString | Represents the maze in form of a string. Usefull for debugging. |

# Algorithms

class in FinixMakesGames.MazeGenerator.Core / Implemented in FinixMakesGames.MazeGenerator.Core

## Description

A static class that holds a collections of methods that picks up a new Grid class and generates it's maze layout.

## Static Methods

| Method | Description |
|---|---|
| BinaryTree | Applies the Binary Tree algorithm to a Grid (Only available to QuadGrid). |
| Sidewinder | Applies the Sidewinder algorithm to a Grid (Only available to QuadGrid). |
| AldousBroder | Applies the Aldous-Broder algorithm to any type of Grid. |
| Wilson | Applies Wilson's algorithm to any type of Grid. |
| HuntAndKill | Applies the Hunt-And-Kill algorithm to a Grid (Only available to QuadGrid). |
| RecursiveBacktracker | Applies the Recursive Backtracker algorithm (Also known has a Randomized Depth-First Search) to any type of Grid. |

# Distances

class in FinixMakesGames.MazeGenerator.Core

## Description

Class that represents a data collection of Cells in a Maze that forms a path or a set of paths relative to a root Cell and it's distance values.

## Constructor

| Type | Result |
|---|---|
| Distances | Creates a new Distances class with a the provided root Cell component with the distance value set to 0, every other Cell distance value will be relative to this root Cell. |

## Public Methods

| Method | Description |
|---|---|
| this[Cell cell] | An override of the [] accessor. When given a Cell it will return it's integer distance to the root cell. If there is no path between these two Cells the value will be null. |
| Cells | Returns a IEnumerable Collection of all cells present in the collection. |
| PathTo | Returns a new Distances Class with ONLY the path between the root Cell and the provided goal Cell. |
| Max | Returns the Cell furthest away from the root Cell. |

# FinixMakesGame.MazeGenerator.EditorScripts

This Section refers to additional Unity.Editor scripts used to enhance the user experience in the inspector window.

## Table of Contents

# MazeEditor

class in FinixMakesGames.MazeGenerator.EditorScripts / Inherits from:Editor

## Description

Sets up a custom Inspector window for the QuadMaze component.