[Overview]

[What’s the Ultimate Grids Engine]

The Ultimate Grids Engine it’s a framework for grid-based games, which allows you to create and populate grids quickly and effortless.

It’s a very fast, mobile-friendly, robust and extendable engine built with **quality** at its core to start creating your own 2D,3D, Rectangle or Hexagon grid-based games right now with Unity.

It’s been designed to act as a foundation for **all sorts of grid-based games**, from dungeon crawlers like Binding of Isaac, to adventure tile based games like old Zelda games, through tactics games like Into The Breach, and really **any game where a grid is required for the player to move or interact with the world**.

It’s a fast, **production ready**, versatile, lightweight, and easy to extend solution that will help you give a serious boost to your Project and save yourself a lot of time.

[Where Do I Start]

First of all, **you don’t have to read all that documentation,** but it is recommended to do so. The engine is built with Unity good practices in mind. So if this is not your first Unity project, you’ll probably be ok on your own. And you can always go back here if something’s not clear.

Besides reading the documentation it’s also recommended to look at the code’s comments directly, usually they cover pretty much any question you might have.

And if all that doesn’t help, you can always use the **support or discord link** on the Asset’s page.

[How to Install]

Introduction

Whatever version of Unity you're using, remember to always import the asset in an **empty project**, so that the engine's project settings get properly imported.

This asset relies on a few **Unity packages** to function. On import this will very likely cause **errors**, that's normal, and easily fixed. It's all explained here.

Importing the asset

To import the asset, follow these steps:

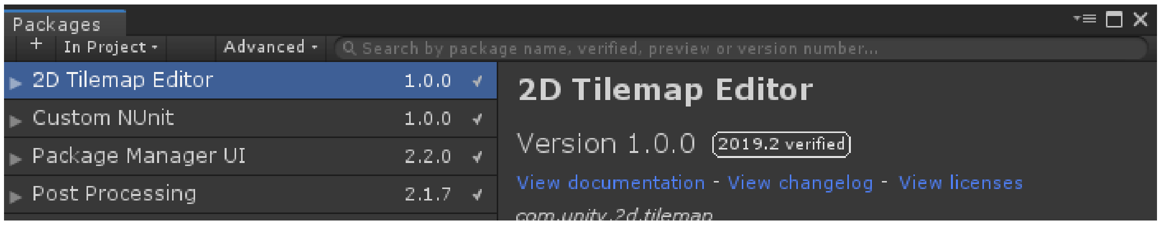
1. Create a **new project** from Unity Hub, pick Unity 2019.2 as your Unity version, and 3D as the Template
2. Go to the Asset Store window and **import** the project (it has to be in an empty project, not an existing one)
3. On import you'll get errors referencing GridBrush related scripts and PostProcessing, that's normal, don't panic, **keep reading** :)

Installing the dependencies

There are only 2 packages needed for the asset to work: 2D Tilemap Editor, and the PostProcessing stack.

Go to Window > Package Manager, and install the following packages:

1. 2D Tilemap Editor
2. PostProcessing


Packages
+
In Project
Advanced
Search by package name, verified, preview or version number...

2D Tilemap Editor	1.0.0 ✓
Custom NUnit	1.0.0 ✓
Package Manager UI	2.2.0 ✓
Post Processing	2.1.7 ✓

2D Tilemap Editor
Version 1.0.0 2019.2 verified
View documentation - View changelog - View licenses
com.unity.2d.tilemap

[Contents of The Asset]

Introduction

When you import the asset into your project, you'll get a UltimateGridsEngine folder, containing two subfolders. Here's a rundown of the contents of these folders, and of the general folder structure.

Common

This folder contains all the scripts and visual assets required for the engine to work. The main structure is made of the following folders:

- **Animations**: all the animations.
- **Editor**: editor scripts such as the brushes used to paint tiles and grid objects on the scene.
- **Fonts**: the font(s) used.
- **Materials**: all materials used such.
- **Models**: contains all models.
- **Prefabs**: prefabs which aren't scene/demo specific go here.
- **Runtime**: the "core" of the engine and pretty much every script which is part of the editor goes here.
- **Shaders**: contains any shader included in the engine such as the Texture/Grid Overlay Shader.
- **Textures**: any texture used on the demo scenes goes here.

Demos

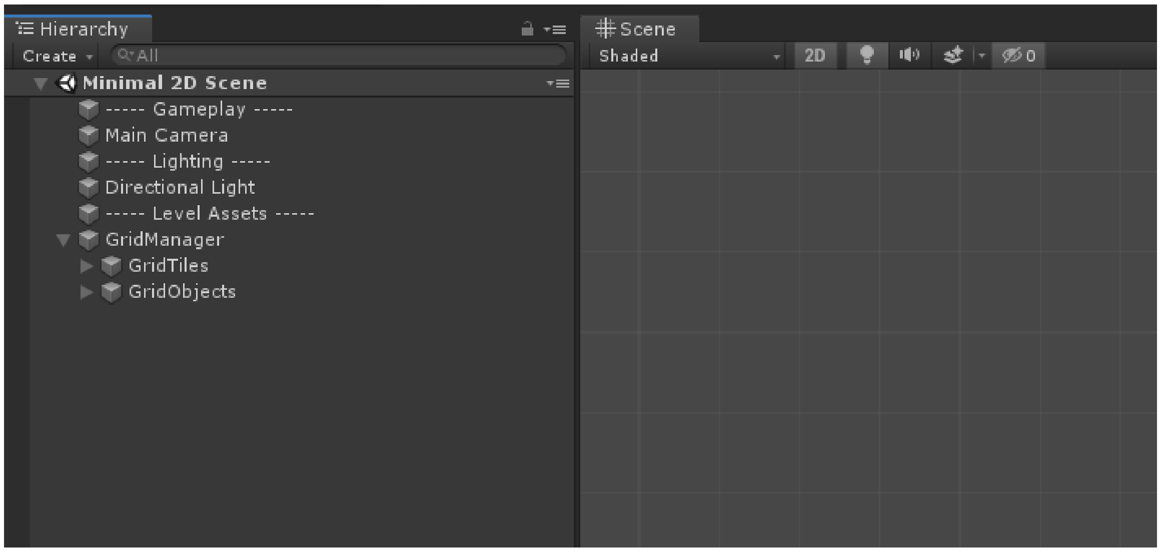
Demos contains the demo scenes included in the engine. These are used to showcase features or the engine as a whole or specific features.

- **3DHex-Demo:** demo scene which showcases how to create a 3D Hex grid.
- **Main-Demo**: the main demo scene featured in the asset store page.
- **PathVisualizer**: small 2D Rectangle Grid which features a pathfinding visualizer and some examples on how to expand on the GridTile.cs base class.
- **Rect-Minimal**: example scene which has the minimum required elements to create a 3D rectangle grid.
- **Hex-Minimal**: example scene which has the minimum required elements to create a Hex grid.

[Minimal Scene Requirements]

In the Ultimate Grids Engine like in most Unity projects, a level is made of a Scene. This can be huge or small, it's up to you. But whatever you do, there are a few elements required for the engine to work properly on a scene. These are:

- **GridManager**: a gameobject containing these components: GridManager and a Grid component which you should also reference on the GridManager component, the GridManager component is responsible for handling most grid methods also the Grid's gameobject will be the parent of every GridTile or GridObject placed using the Tile Palette.
- **Camera**: a gameobject acting as the camera rig and containing the BoxCameraFollower and has the actual camera as a child of it with two scripts attached to it: the CameraShaker and the PixelArtCamera which makes the camera render the scene properly in a pixelated way (This GameObject is replaceable and you may use your own custom solution for camera shaking and camera moving/following).



```
Minimal 2D Scene
    ----- Gameplay -----
    Main Camera
    ----- Lighting -----
    Directional Light
    ----- Level Assets -----
    GridManager
        GridTiles
        GridObjects
```

[How to Create and Populate a Grid]

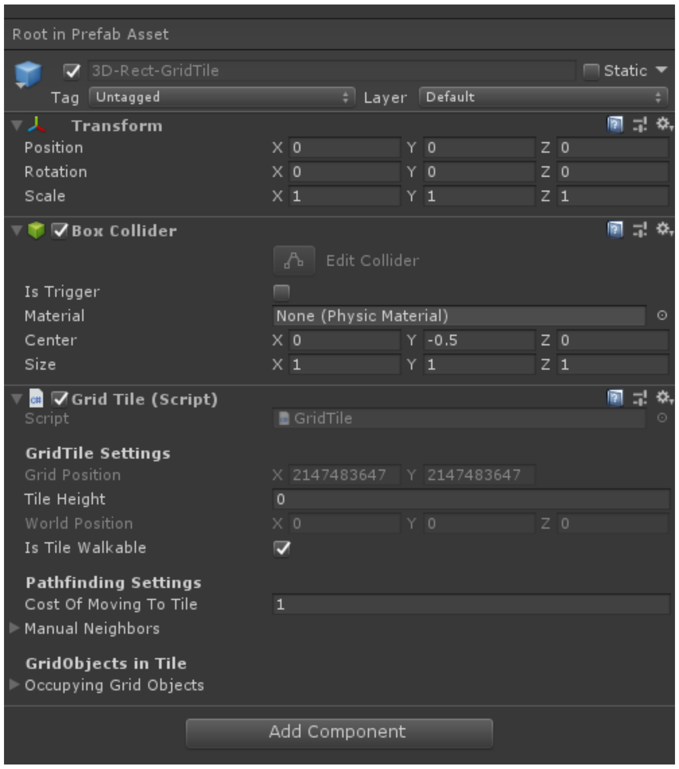
Introduction

This engine features tile-based grids, which means you'll need to create a tile to populate your grid, then setup the grid and add the tiles to it using the tile palette.

Creating a Tile

There are many different ways you can create tiles in the Ultimate Grids Engine. Here a step by step on the one we recommend:

1. Start with an **empty gameobject.** Ideally you'll want to separate the components part from the visual part. The best possible hierarchy has the components on the root gameobject and the visual parts (model, sprite, etc) nested on a child gameobject usually named "ModelHolder".
2. On your top level object, add a **BoxCollider2D**. Adjust its size to match your sprite/model dimensions.
3. Add a **GridTile** component. Check the various settings to make sure they're ok with you.
4. Drag the gameobject into the **Project Window** to turn it into a **prefab**.

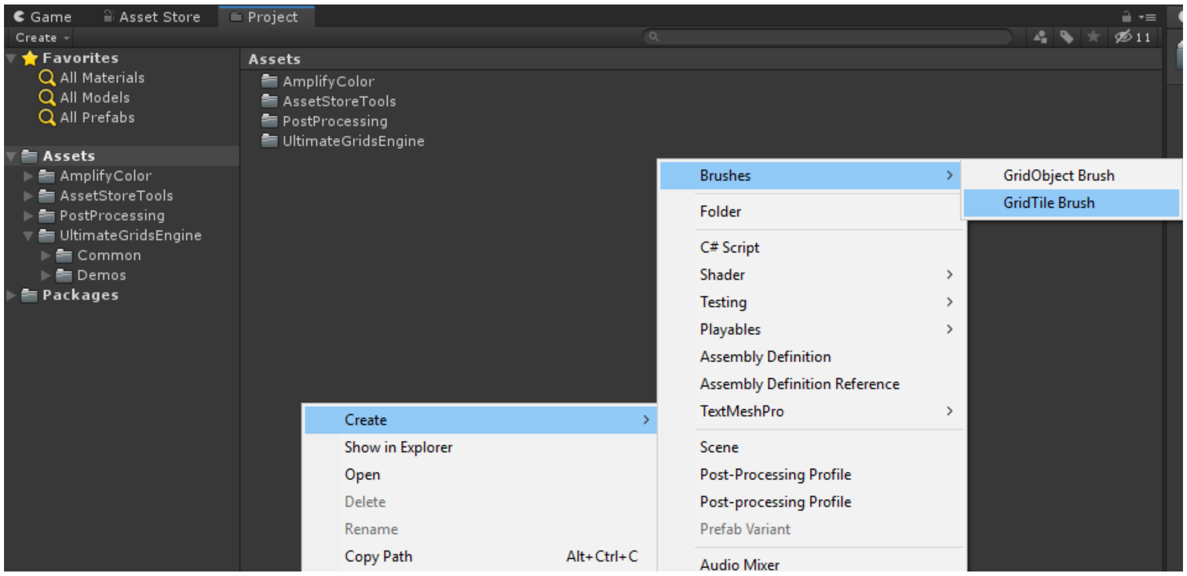


```
Root in Prefab Asset
[✓] 3D-Rect-GridTile                         [ ] Static
Tag  Untagged              Layer  Default
Transform
  Position        X 0      Y 0      Z 0
  Rotation        X 0      Y 0      Z 0
  Scale           X 1      Y 1      Z 1
[✓] Box Collider
                  Edit Collider
  Is Trigger      [ ]
  Material        None (Physic Material)
  Center          X 0      Y -0.5   Z 0
  Size            X 1      Y 1      Z 1
[✓] Grid Tile (Script)
  Script          GridTile
  GridTile Settings
  Grid Position   X 2147483647   Y 2147483647
  Tile Height     0
  World Position  X 0      Y 0      Z 0
  Is Tile Walkable [✓]
  Pathfinding Settings
  Cost Of Moving To Tile  1
  Manual Neighbors
  GridObjects in Tile
  Occupying Grid Objects
Add Component
```

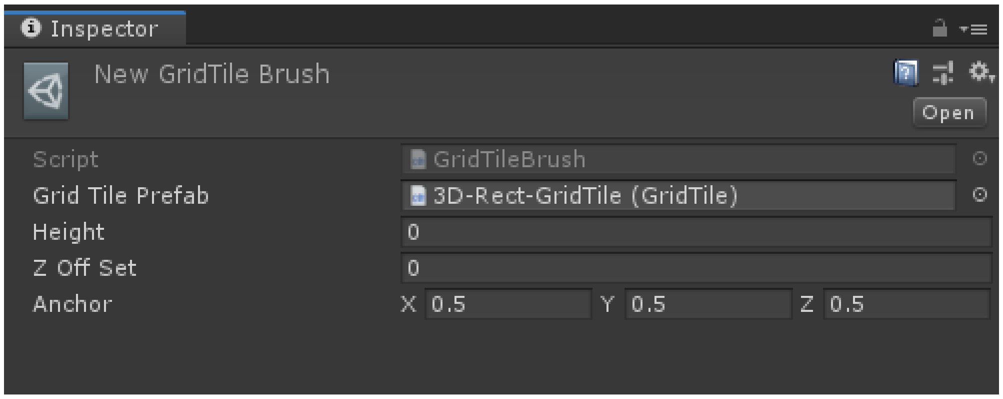
Creating a Tile Brush

Now that we have a tile prefab, we need to create a tile brush for it we can add the tile to our grid later.

1. Right click inside your assets folder in the project window and click Create > Brushes > GridTile Brush.



2. Rename the newly created GridTile brush file to whatever you want, just make sure you'll be able to remember this easily.
3. Select the GridTile brush and on the inspector add your tile prefab to the Grid Tile Prefab Field.

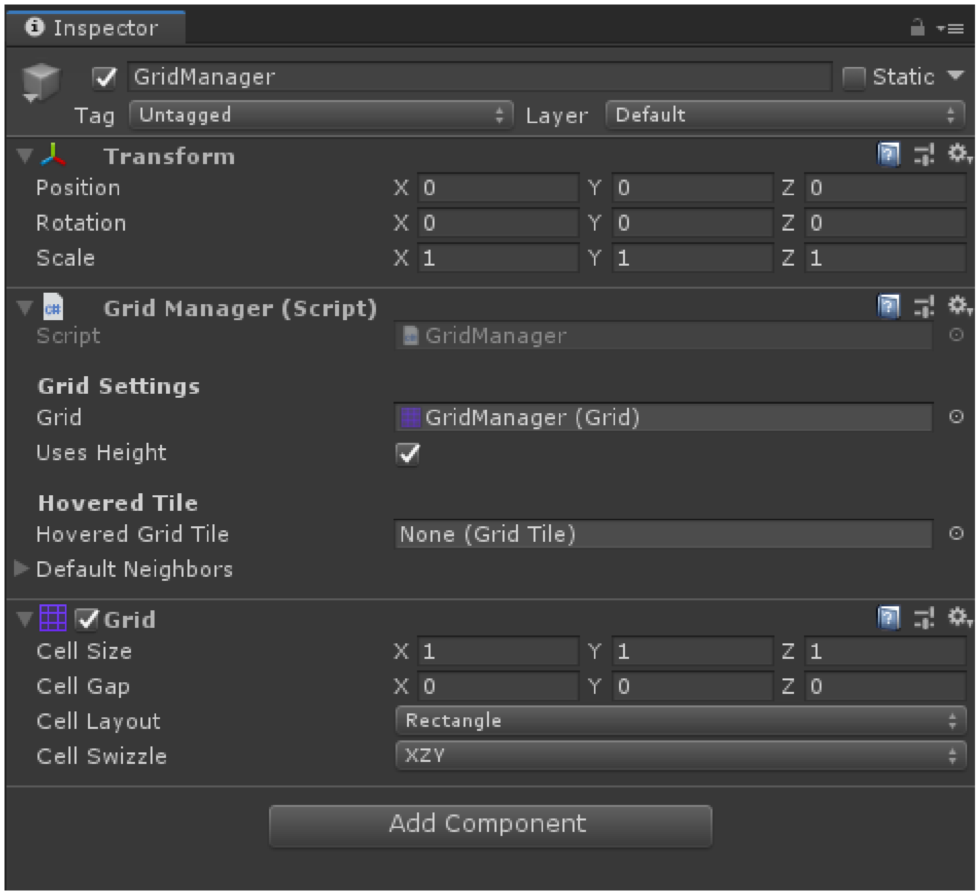


Script GridTileBrush
Grid Tile Prefab 3D-Rect-GridTile (GridTile)
Height 0
Z Off Set 0
Anchor X 0.5 Y 0.5 Z 0.5

Creating a Grid

Now that we have created a new GridTile brush for the tile, we are going to create a new grid in our scene.

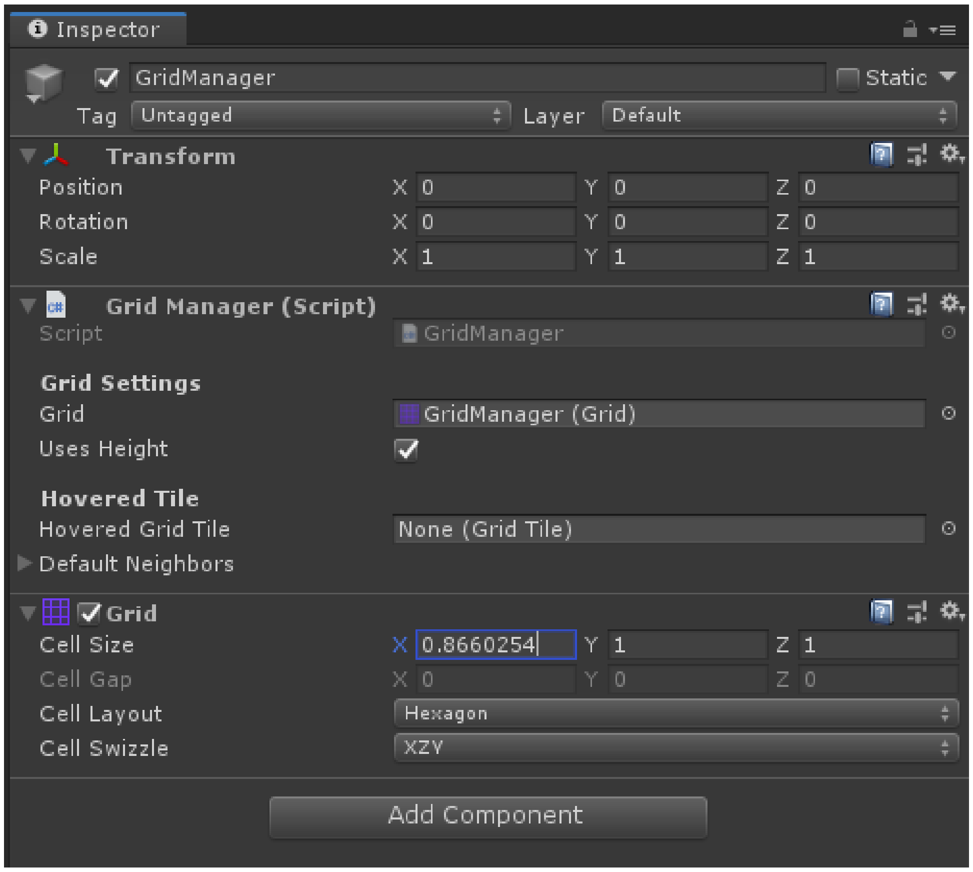
1. Start by creating a new **empty gameobject** on your scene. Rename it to "GridManager" and add a GridManager component to it. A Grid component should now be attached to it automatically.



```
Inspector
GridManager                                   Static
Tag  Untagged                     Layer  Default
Transform
Position        X 0        Y 0        Z 0
Rotation        X 0        Y 0        Z 0
Scale           X 1        Y 1        Z 1
Grid Manager (Script)
Script          GridManager
Grid Settings
Grid            GridManager (Grid)
Uses Height     ✓
Hovered Tile
Hovered Grid Tile   None (Grid Tile)
Default Neighbors
Grid
Cell Size       X 1        Y 1        Z 1
Cell Gap        X 0        Y 0        Z 0
Cell Layout     Rectangle
Cell Swizzle    XZY
Add Component
```

2. Go ahead and modify the Grid component settings to whichever you'll want for your grid, most of the times using the default settings will be enough except for the cell layout and cell swizzle, on the cell layout you can choose the layout you want for your grid **Rectangle or Hexagon** and on the Cell Swizzle you can choose the desired alignment **XZY** it's default for **3D** grids and **XYZ** its default for **2D** grids.

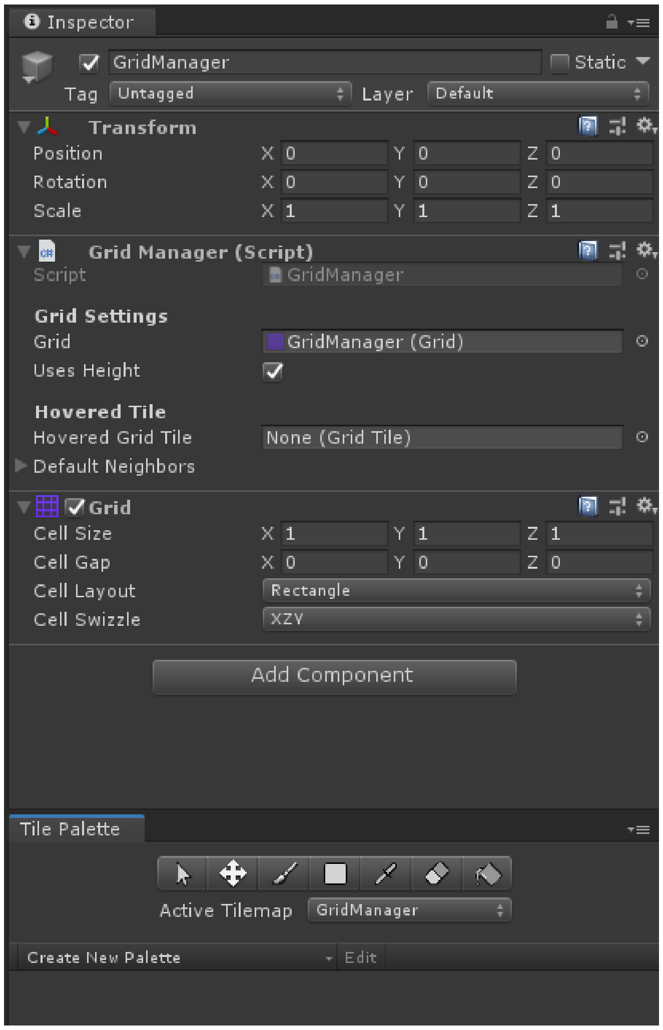
3. If you are making a **Hexagonal Grid** make sure to set the **X** value of the Cell Size field to **0.8660254.** If you want some more info on why Click Here.



Populating the Grid with Tiles

Now that we have created a new Grid in our scene and have a tile brush, we are going to paint tiles into the grid.

1. Open the Tile Palette (Window > 2D > Tile Palette).
2. Select the GridManager gameobject on the scene's Hierarchy and make sure that this is the gameobject which shows as the Active Tilemap on the Tile Palette.


Inspector

GridManager Static

Tag Untagged Layer Default

Transform

Position X 0 Y 0 Z 0

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Grid Manager (Script)

Script GridManager

Grid Settings

Grid GridManager (Grid)

Uses Height ✓

Hovered Tile

Hovered Grid Tile None (Grid Tile)

Default Neighbors

Grid

Cell Size X 1 Y 1 Z 1

Cell Gap X 0 Y 0 Z 0

Cell Layout Rectangle

Cell Swizzle XZY

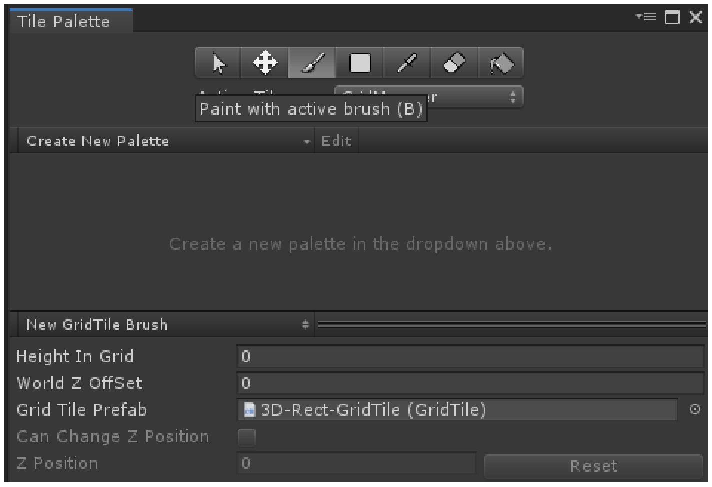
Add Component

Tile Palette

Active Tilemap GridManager

Create New Palette Edit

3. On the tile palette Click on the brush selector “Default Brush” and you should get a list of the available brush on the project, find the one you just created and select it. Press B or click the paint Icon on the Tile Palette.


Tile Palette

Paint with active brush (B)

Create New Palette Edit

Create a new palette in the dropdown above.

New GridTile Brush

Height In Grid 0

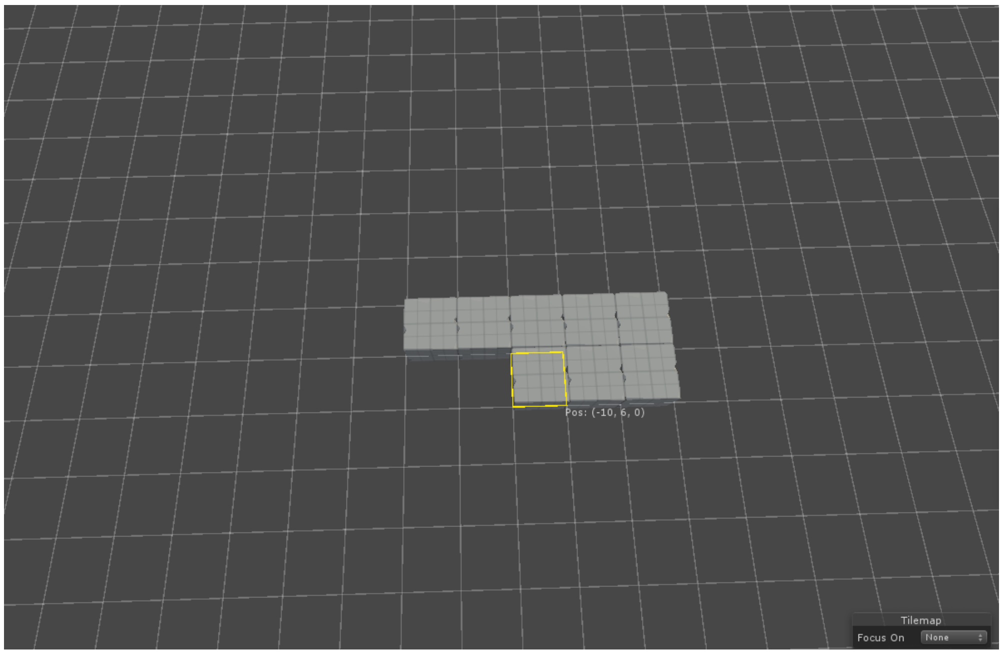
World Z OffSet 0

Grid Tile Prefab 3D-Rect-GridTile (GridTile)

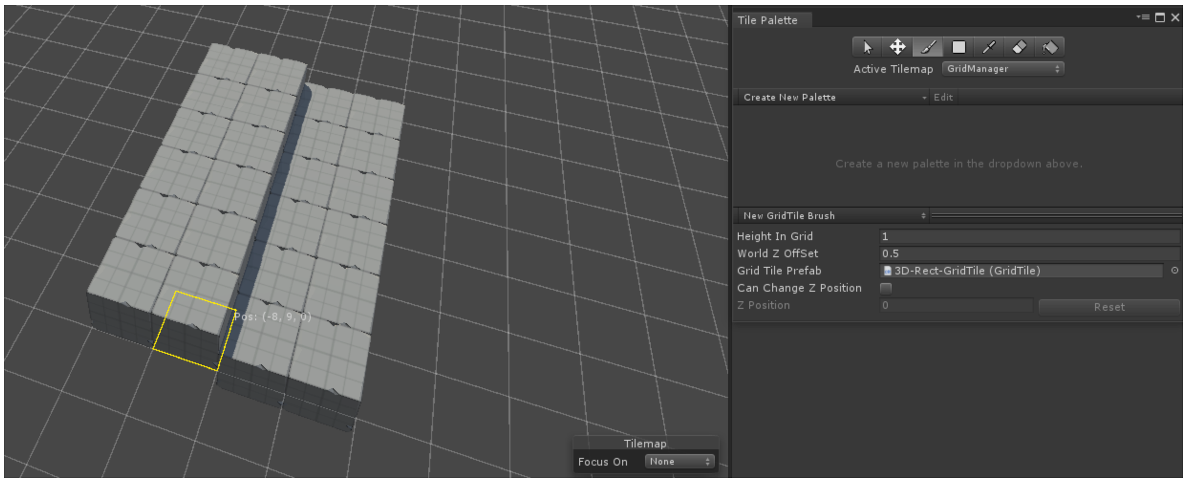
Can Change Z Position

Z Position 0 Reset

4. Now on the scene view you can move your mouse around and you should be able to paint and erase the tile into/from the grid. You should also be able to see which is the Grid Position which your mouse is currently hovering.



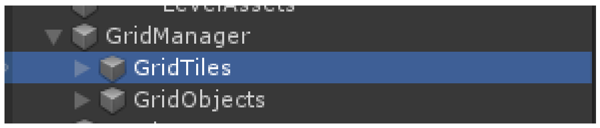
5. You can modify the tile's Grid Height and World Z Offset on the Tile Palette. The height in Grid field is the height the tile will have on the grid (this is used for neighbor finding when the height is enabled on the grid manager) and the World Z Offset just modifies the Height it will have on the scene (doesn't affect the Height in Grid).



Height In Grid 1
World Z OffSet 0.5
Grid Tile Prefab 3D-Rect-GridTile (GridTile)
Can Change Z Position ☐
Z Position 0
Reset

In this example we set the upper floor tiles height in the grid to 1 and z offset to 0.5

6. Whenever you paint a tile into the grid it will be nested as a child object of the GridTiles object which has the GridManager as a parent.



7. Now you know how to create and populate grids, you can also check out the demo scenes to learn more about the settings and how the engine works.

If you have any doubt or a suggestion please contact us through the discord or support links found on the Asset's Page

Disclaimer: This is the V1.0 of the documentation, help us maker it better by asking any question you have through discord and it will probably be added here.