

# Super Tilemap Editor Manual

**Creative Spore**

**Support:** [creativespore@gmail.com](mailto:creativespore@gmail.com)

**Web:** <http://www.creativespore.com>

**Unity Forums:** user CreativeSpore

## Index

<a href="#">1 Introduction.....</a>	<a href="#">3</a>
<a href="#">2 Getting Started.....</a>	<a href="#">3</a>
<a href="#">2.1 Creating a Tileset.....</a>	<a href="#">3</a>
<a href="#">2.2 Creating a Tilemap.....</a>	<a href="#">6</a>
<a href="#">2.3 Creating tile colliders.....</a>	<a href="#">8</a>
<a href="#">3 Super Tilemap Editor.....</a>	<a href="#">9</a>
<a href="#">3.1 Tile Palette.....</a>	<a href="#">9</a>
<a href="#">3.1.1 Tile Views.....</a>	<a href="#">10</a>
<a href="#">3.2 Tile Properties.....</a>	<a href="#">11</a>
<a href="#">3.2.1 Tile Properties - Colliders.....</a>	<a href="#">11</a>
<a href="#">3.2.2 Tile Properties - Parameters.....</a>	<a href="#">12</a>
<a href="#">3.2.3 Tile Properties - Prefab.....</a>	<a href="#">13</a>
<a href="#">3.3 Tilemap.....</a>	<a href="#">13</a>
<a href="#">3.3.1 Tilemap - Paint.....</a>	<a href="#">14</a>
<a href="#">3.3.2 Tilemap - Renderer.....</a>	<a href="#">14</a>
<a href="#">3.3.3 Tilemap - Map.....</a>	<a href="#">15</a>
<a href="#">3.3.4 Tilemap - Collider.....</a>	<a href="#">16</a>
<a href="#">3.4 Tilemap Groups.....</a>	<a href="#">16</a>
<a href="#">3.5 Brush Groups.....</a>	<a href="#">17</a>
<a href="#">3.6 Brushes.....</a>	<a href="#">19</a>
<a href="#">3.6.1 AnimBrush.....</a>	<a href="#">20</a>
<a href="#">3.6.2 RandomBrush.....</a>	<a href="#">21</a>
<a href="#">3.6.3 RoadBrush.....</a>	<a href="#">22</a>
<a href="#">3.6.4 CarpetBrush.....</a>	<a href="#">24</a>
<a href="#">3.6.5 A2x2Brush.....</a>	<a href="#">25</a>
<a href="#">3.7 Custom Brushes.....</a>	<a href="#">25</a>
<a href="#">3.7.1 IBrush Interface.....</a>	<a href="#">25</a>
<a href="#">3.7.2 TileData.....</a>	<a href="#">26</a>

## 1 Introduction

This is the user manual for the Unity asset Super Tilemap Editor.

Super Tilemap Editor is a fast and easy to use tile editor to help you to create any game based on tiles.

This manual will teach you how to use and configure the tilesets and tilemaps to use them properly in your projects and created tiled games really easy and fun.

## 2 Getting Started

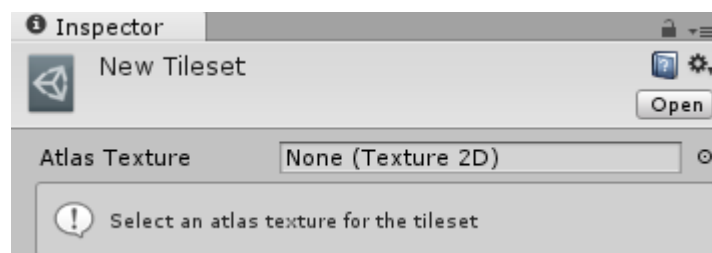
If you are really anxious to use the tilemap editor and have no time for reading all the manual, this will give you a fast introduction to start creating tilemaps for your games.

### 2.1 Creating a Tileset

Before creating a tilemap in the scene, you need to create a tileset to be used by the tilemap.

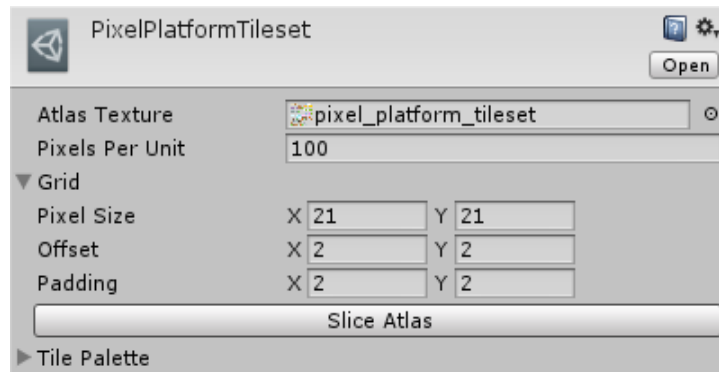
The tileset is basically composed by a texture atlas with the tiles, a list of the tiles sliced from the atlas and brushes (they will be explained later).

To create a tileset, go to "**Assets/Create/SuperTilemapEditor/Tileset**".



The tileset will be created without an atlas texture selected.

Let's select one. I am going to select the one included with the asset based on a Kenney's free to use texture.



Now, we can see new options: Grid and Tile Palette.

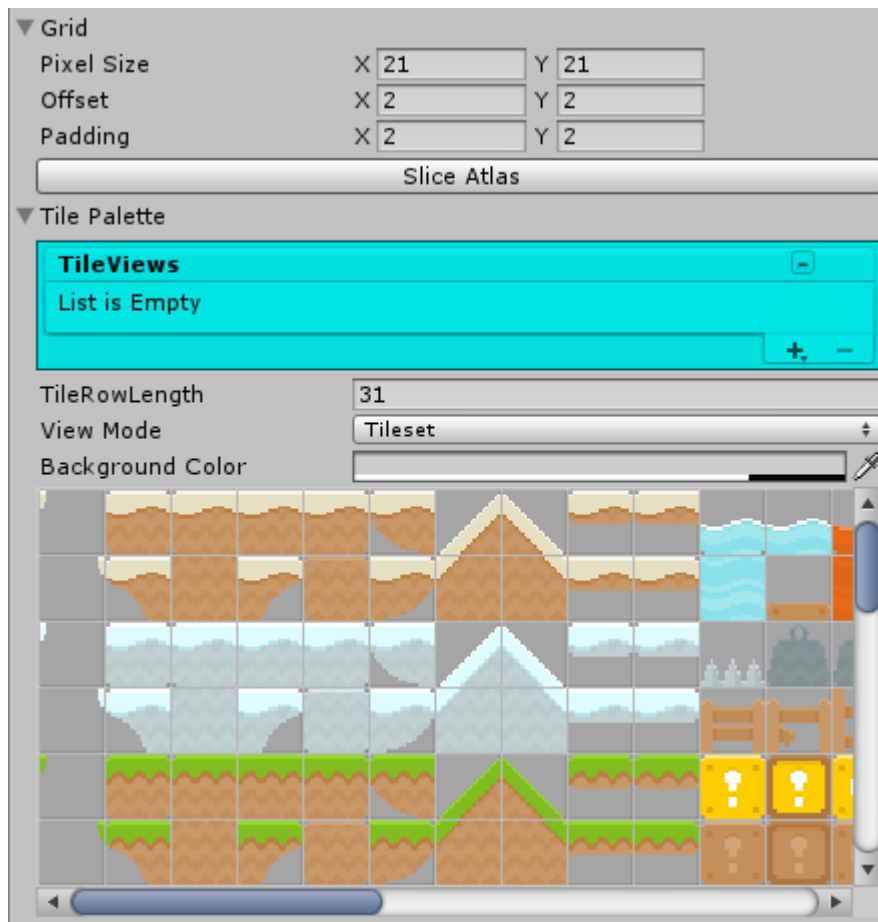
But we still need to slice the atlas to create the actual tiles used by this tileset.

**Pixels per Unit** will take this value from the Atlas Texture, but you can change it here to a different value. When a tilemap is created, this value will be used to set the cell size of the tile, but you can change the cell size as well after the creation of the tilemap.

Slicing the atlas works the same as Unity Sprite Editor:

- **Pixel Size:** is the size of the tile in pixels
- **Offset:** is an offset in pixels to start slicing the texture
- **Padding:** is the separation between tiles in pixels

Once we set the parameters, press "Slice Atlas" and you will see the tile palette. If you miss a parameter, fix it and slice again.

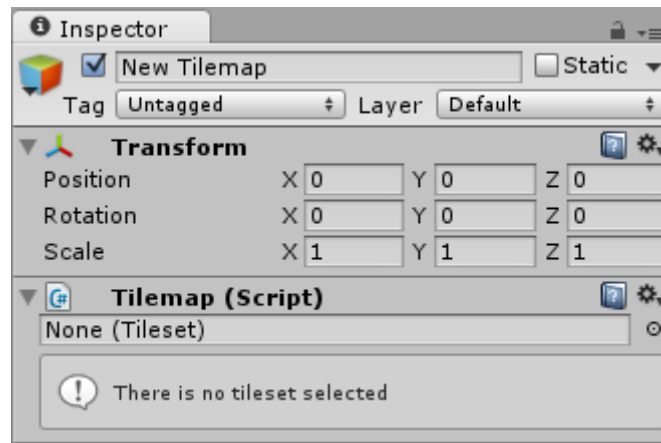


And that's it. You have finished creating your first tileset.  
Let's now create a tilemap using your new tileset.

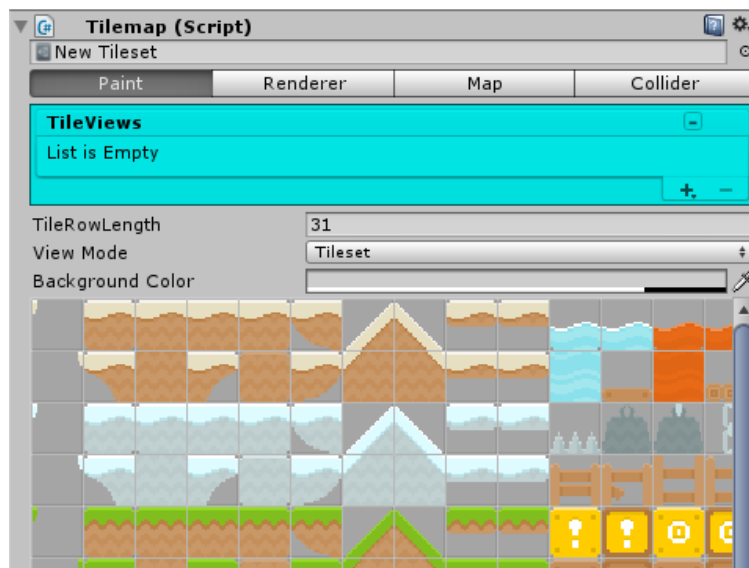
## 2.2 Creating a Tilemap

Tilemaps are gameobjects created in a scene.

To create a tilemap go to "**GameObject/SuperTilemapEditor/Tilemap**".

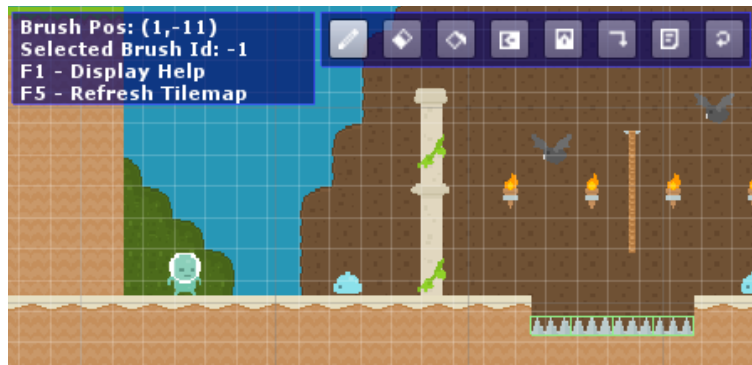


The tilemap need a Tileset to work. Let's select the tileset created in previous section "Creating a Tileset".



Once a tileset has been selected, a list of options appear.

To paint in the scene tilemap, you have to select the **Paint option**, then press left click over any tile in the tile palette and start painting in the scene by left clicking.



You can select more than one tile from the tile palette by dragging the mouse holding left button.

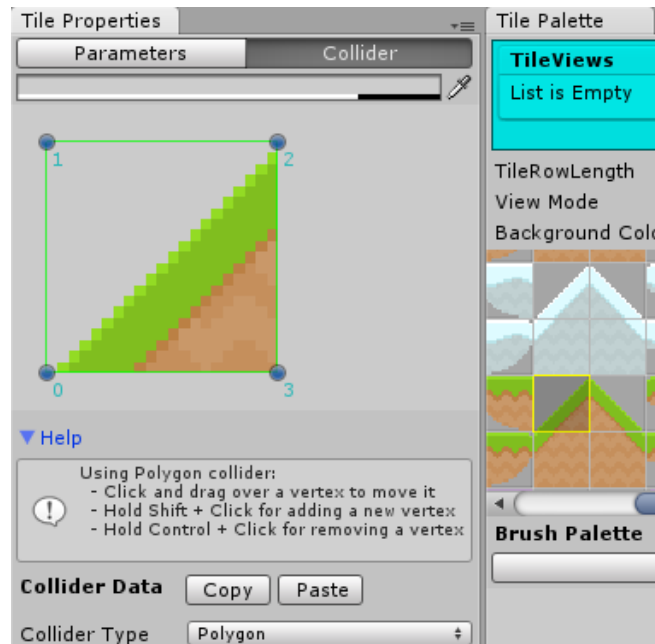
Hold center or right mouse button and drag to move the tile palette scroll.

While painting in the scene, you can:

- Paint holding left mouse button
- Erase holding shift + left mouse button
- Fill an area by double clicking with a single tile selected
- Copy tiles by dragging and holding right mouse button
- Cut tiles by dragging and holding right mouse button and Shift key
- Rotate and flip the brush tiles by using special keys:
  - Rotate 90° by using comma ',' and period '.'
  - Vertical Flip by pressing X
  - Horizontal Flip by pressing Y
  - Hold shift to only rotate or flip tile positions
- Use Ctrl-Z and Ctrl-Y to undo/redo changes

## 2.3 Creating tile colliders

To modify the collider properties of a tiles you have to open the tile properties window. Go to "**Window/SuperTilemapEditor/Tile Properties Window**".



Select Collider section in the tile properties window.

You will see the tile selected in the tile palette.

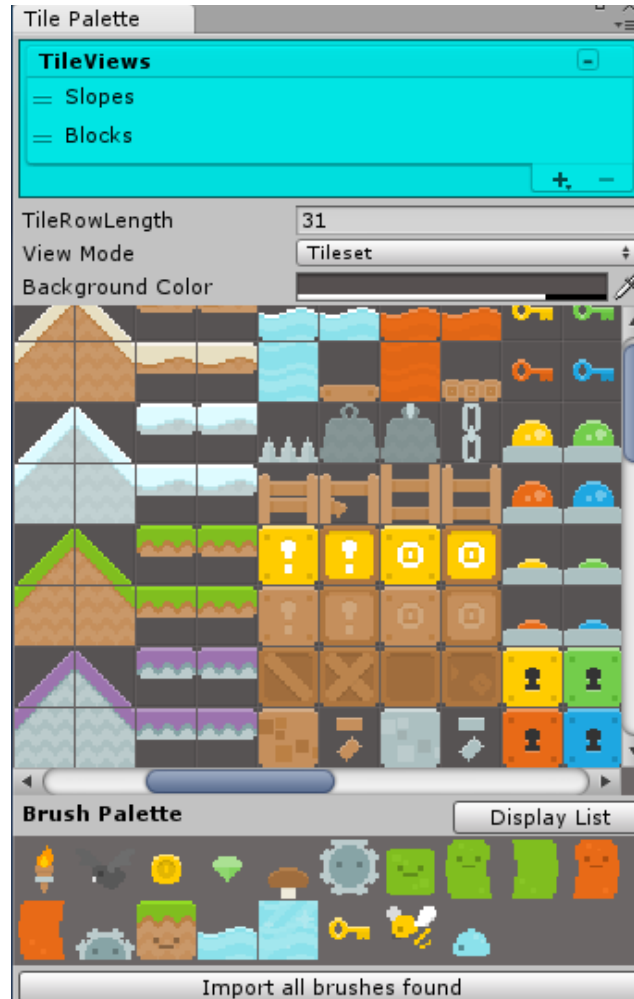
It could be None, Full or Polygon.

**Trick:** right click over a tilemap tile to select the tile faster from your level.



## 3 Super Tilemap Editor

### 3.1 Tile Palette



The Tile Palette can be found in different places like:

- The tileset inspector view
- The Tilemap inspector view with Paint tab selected
- The Tile Palette Window **"Window/SuperTilemapEditor/Tile Palette Window"**

This control is connected with a source tileset. In case of the Tile Palette Window, the source tileset will be set to the last selected tileset or tilemap ( in this case, the tileset used by the tilemap ).

Here are displayed the tile views ( a custom selection of tiles ), all the tileset tiles availables, and all the brushes found using the source tileset.

There are some settings you can change for a better display of the data:

- **TileRowLength:** this is set by default to the tileset maximum row length and specify the number of tiles per row. The tiles will be wrapped to fit this number.
- **View Mode:** select TileView or Tileset, to display the selected tileview or display all tiles in the tileset.
- **Background color:** change the color of the background.

From the tile scroll view, you can select one tile or a rectangle selection of tiles to be painted on a tilemap or for creating a new tileview.

Press and hold left mouse button to make a rectangle selection or just click over a tile to select it.

Press and hold right or center mouse button to move the tile scroll view.

You can also select a brush in the bottom brushes palette. Brushes will be explained in the brushes section.

Double click over a brush will ping the brush asset in the project view and show its properties in the inspector view.

Right click over a tile to display its properties.

### 3.1.1 Tile Views

Tile Views are selection of tiles saved as a tile view to display only that specific selection.

To create a tile view, make a rectangle selection of tiles, then press the '+' button and select "**Add Tile Selection to TileView**" to add a selection from the tile palette or "**Add Brush Selection**" to add the brush selection.

To remove a tile view, select the tile view and then press the '-' button.

Selecting a tile view will automatically display the tile view in the tile palette.

You can rearrange tile views as you wish by pressing and dragging the tile view in the list.

To see all tiles again, change the **View Mode** to Tileset.

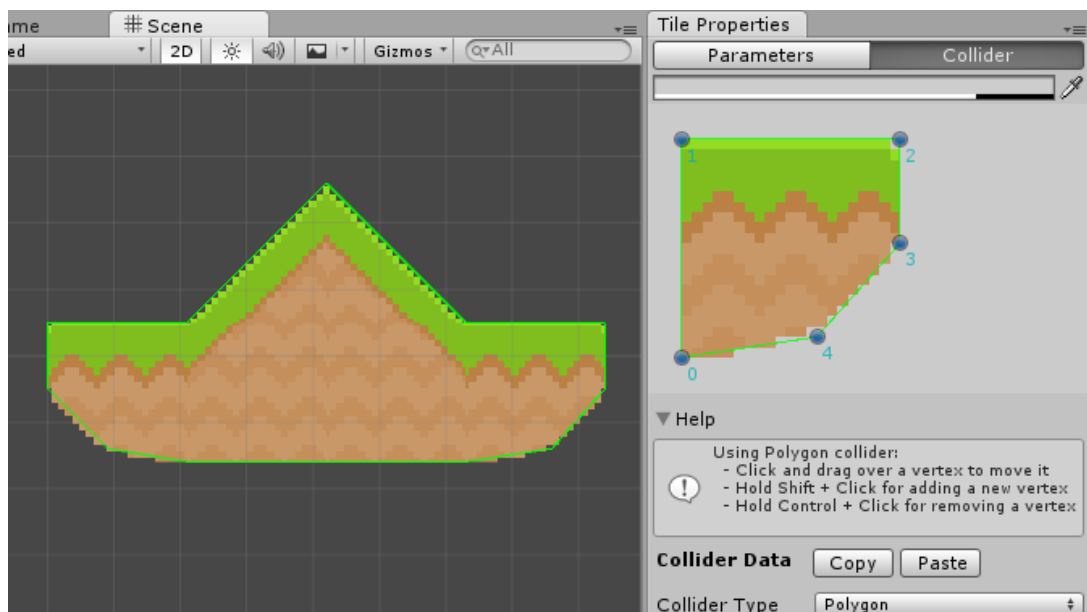
## 3.2 Tile Properties

The tiles properties are displayed by the Tile Properties Window "**Window/SuperTilemapEditor/Tile Properties Window**".

Here will be displayed the properties of the current selected tile. Select a tile using any tile palette. In case of multi-selection only one of the tiles will be displayed.

These properties are divided in Colliders and Parameters.

### 3.2.1 Tile Properties - Colliders



In this section you can set the collider type of the tile:

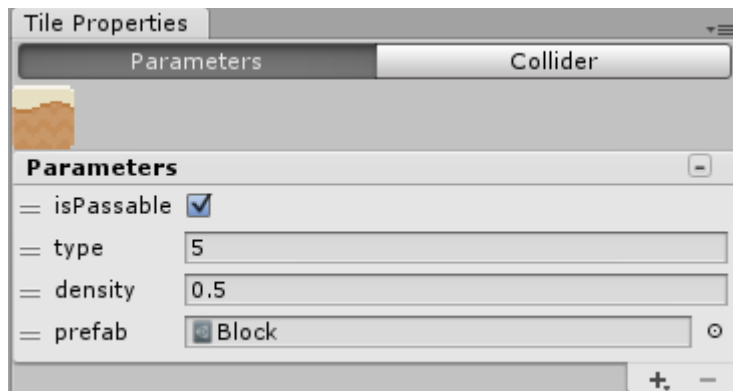
- **None:** the tile has no collider
- **Full:** the full tile is a collider
- **Polygon:** custom polygon collider

Only the polygon collider allows to create, remove and modify the collider vertices:

- Click and drag over a vertex to move it
- Hold Shift + Click for adding a new vertex (mouse should be over an edge)
- Hold Control + Click for removing a vertex ( with a minimum of 3 vertices )

You can copy the tile collider data of the current displayed tile and paste it selecting another tile or selection of tiles (**Multi-selection edition** will be displayed in that case and Paste will modify all of them with the previous copied data)

### 3.2.2 Tile Properties - Parameters



Each tile in the tileset, and brush assets can have a list of parameters of type int, float, bool and object.

These parameters can later be accessed by code for custom logic, like creating an instance or set special properties.

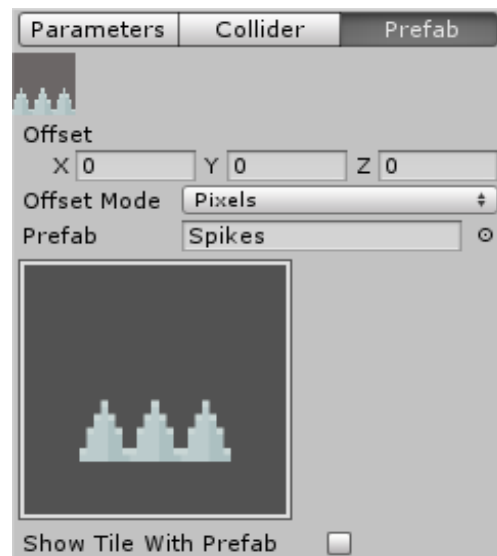
To access the parameters of a tile, select the tile or brush from any tile palette (Tile Palette Window, Tilemap\Paint or Tilesset).

To add a new parameter, press the '+' button, and the '-' button to remove a parameter.

You can rename any parameter by clicking over its name and drag parameters to rearrange them.

```
// If parameter "paramName" exists, sets the value to true
Tiles[0].paramContainer.FindParam("paramName").SetValue(true);
// Creates a new integer parameter named "paramName". To create a float parameter use 25f
Tiles[0].paramContainer.AddParam("paramName", 25);
// "paramName" should exist and be an integer. Takes the value of this integer parameter
Tiles[0].paramContainer.FindParam("paramName").GetAsInt();
```

### 3.2.3 Tile Properties - Prefab



A tile can create a prefab instead of being painted if a prefab is set in the **Prefab** data.

You can set an **offset** for this prefab, centered in the center of the tile.

The offset could be in units or pixels changing the **Offset Mode**.

If the offset is in pixels, the offset will keep the aspect ratio even if the tilemap cell size is changed later.

**Show Tile With Prefab** set if the tile should be shown or hidden when it has a prefab attached.

## 3.3 Tilemap

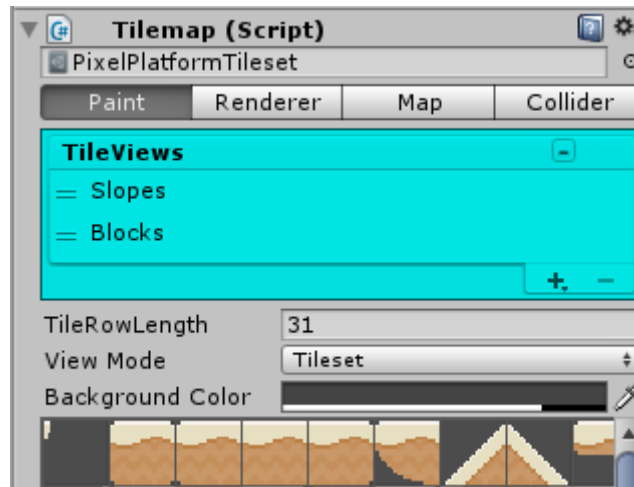
Tilemaps are the game objects containing the map of tiles. Each tilemap have an unique tilesset for drawing tiles.

Imagine the tilemap like an infinite plane where you can draw using the tile brush.

The transform properties can changed, to rotate, scale and move the tilemap plane over the 3D space.

The tilemap inspector shows different options for different setting of the tilemap.

### 3.3.1 Tilemap - Paint

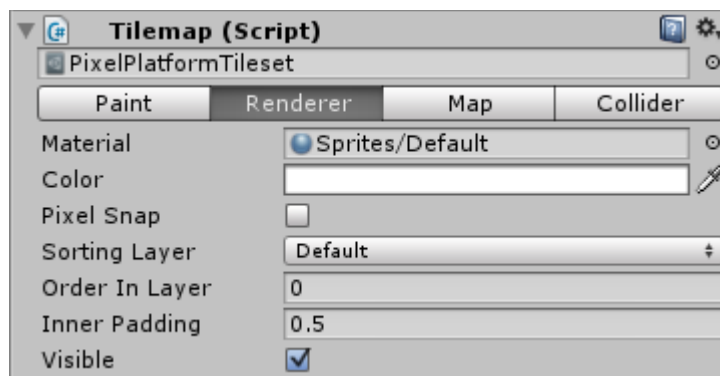


This tab should be selected in order to paint tiles on the selected tilemap.

The inspector will show the tile palette where you can access the different tile views, select a tile or a rect of tiles or select a brush.

To know more about painting tiles, check the section [#2.2.Creating a Tilemap](#).

### 3.3.2 Tilemap - Renderer

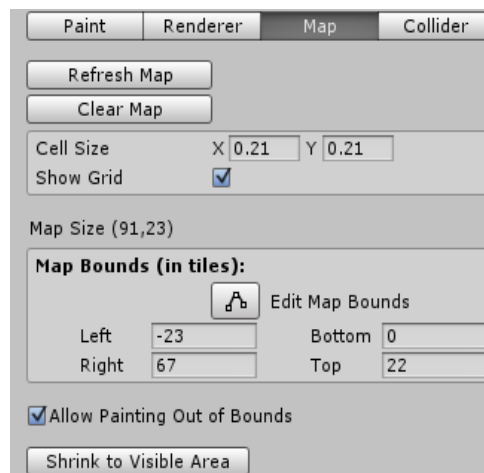


In this section you can change the visual aspects of the tilemap:

- **Material:** the material used to render the tile mesh
- **Color:** the tint color for the tilemap
- **Pixel Snap:** pixel perfect option, only if Sprite material is used
- **Sorting Layer:** the sorting layer used by the tilemap
- **Order In Layer:** the order inside the selected layer

- **Inner Padding:** the size, in pixels, the tile UV will be stretched. Use this to fix pixel precision artifacts when tiles have no padding border in the atlas.
- **Visible:** enable/disable the render of the tilemap

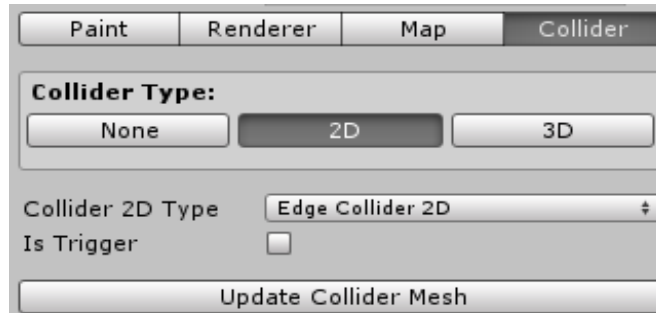
### 3.3.3 Tilemap - Map



This section display properties relatives to the tilemap:

- **Map Size:** this is the size of the map, automatically increased to enclose all drawn tiles.
- **Map Bounds:** this is the limit area of the map. It is automatically updated while painting outside the map bounds but you can manually set the bounds here or press the **Edit Map Bounds** button and drag the handles directly in the scene view.
- **Allow Painting Out of Bounds:** is unchecked it would be only possible to paint tiles inside the map bounds.
- **Shrink to Visible Area:** reduce the map bounds until fit all visible tiles.
- **Cell Size:** the size of the cell where the tile is drawn.
- **Show Grid:** show/hide the tile grid.
- **Refresh Map:** will refresh the whole tilemap, tiles and colliders. Use this button when changing tile properties like colliders to update the map with the changes.
- **Clear Map:** will remove all tiles from the tilemap.

### 3.3.4 Tilemap - Collider



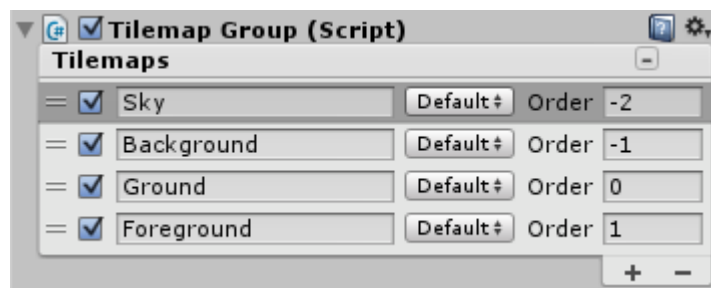
This section shows the collider settings:

- **Collider Type:** None, 2D and 3D
- **Collider Depth:** the depth of the collider (only 3D mode)
- **Collider 2D Type:** the type of collider used: Edge or Polygon (only 2D mode)
- **Is Trigger:** set isTrigger value for all colliders.
- **Update Collider Mesh:** updated the collider mesh. Use this button to update the mesh collider after changing collider settings like Collider Depth.

### 3.4 Tilemap Groups

A tilemap group is a script used to manage a group of tilemaps like if they were layers.

To create a tilemap group go to "**GameObject->SuperTilemapEditor->TilemapGroup**" or add the TilemapGroup script to an object with tilemap children.



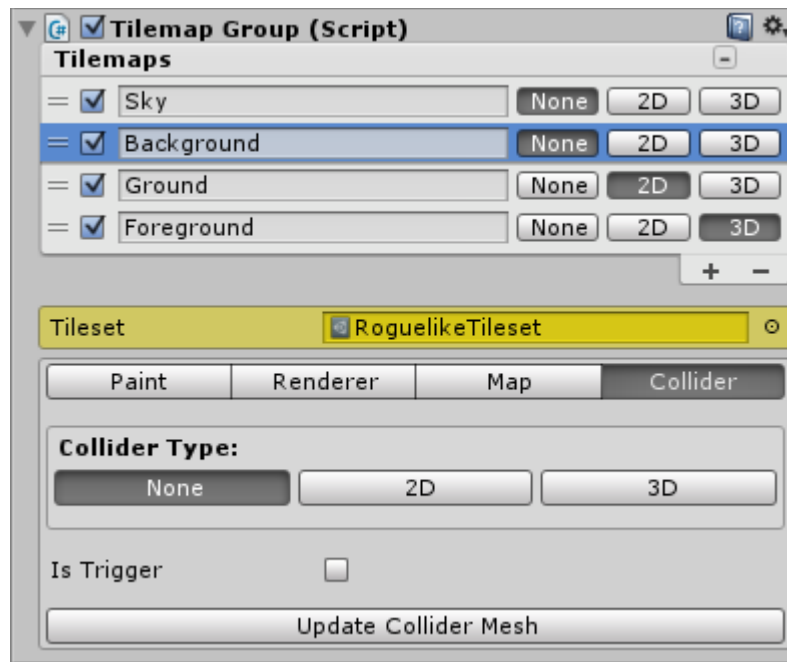
In a tilemap group, you can create, remove or select a tilemap directly in the reorderable list.



When a tilemap is selected the tilemap inspector is displayed below.

You can directly access some properties of the tilemap directly from the tilemap list, like sorting layer and sorting order.

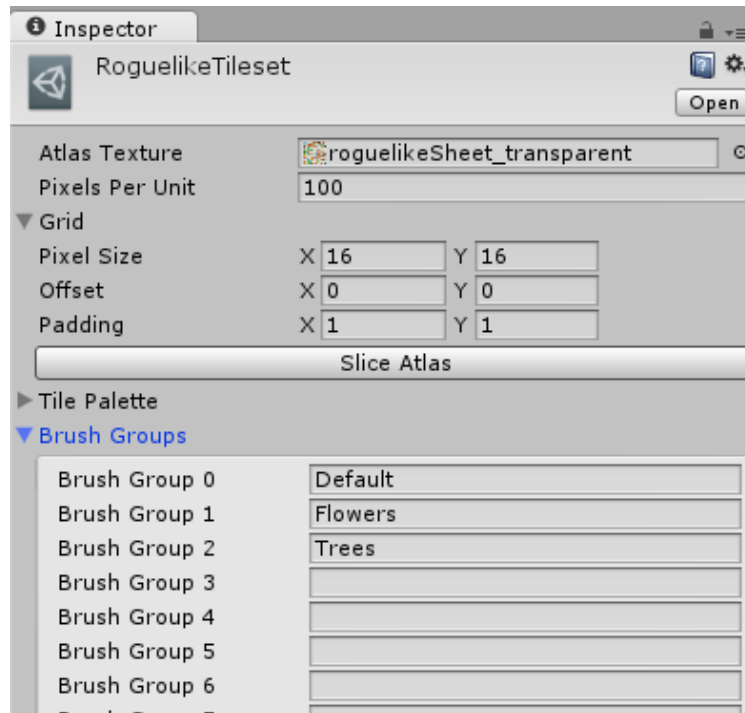
If you select the mode **Collider** you can change also the type of collider for the tilemap.



### 3.5 Brush Groups

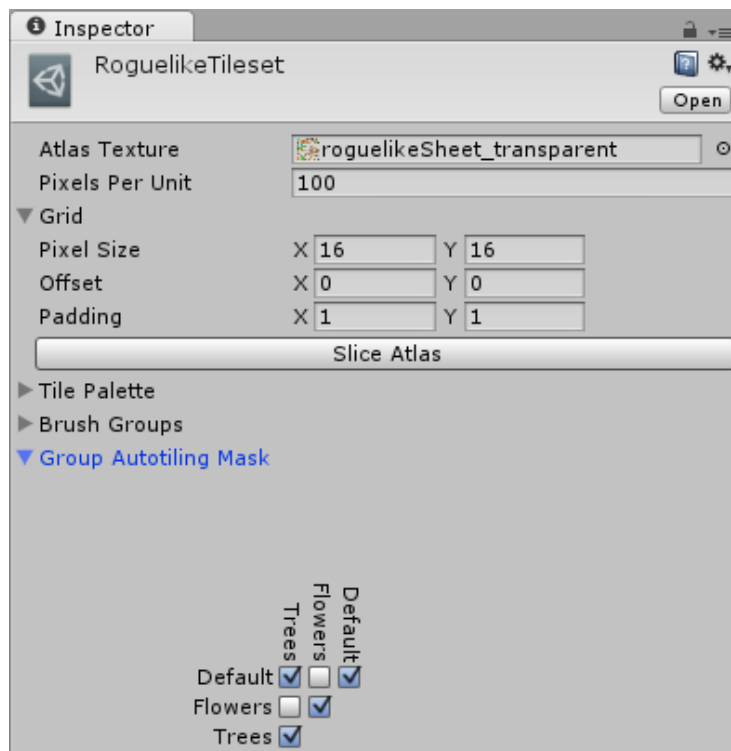
A brush group is like a Unity layer used for a game object. You can specify what group a brush belongs to and set the autotiling behaviour for each group.

To create a new group, select the tileset and open the **Brush Groups** foldout.



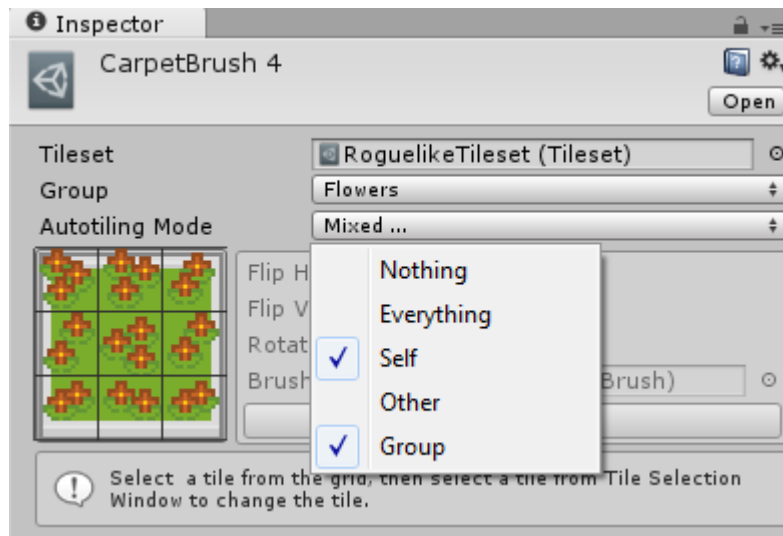
You can add a group by setting a name. There are up to 32 groups to be used.

To set the autotiling mask (what groups autotile with what groups), open the **Group Autotiling Mask** foldout.



You can check when two groups should autotile together by pressing the check box.

All brushes have a **Group** property and an **Autotile Mode**.



Only a group can be assigned to a brush, but you can mix the Autotiling Modes:

- **Self**: the brush autotiles only with tiles of the same brush
- **Other**: the brush autotiles with tiles of different brush or any tile not empty
- **Group**: use the **Group Autotiling Mask** to check if should autotile other tile is the brush of that tiles has a group that autotiles with the group of this brush

### 3.6 Brushes

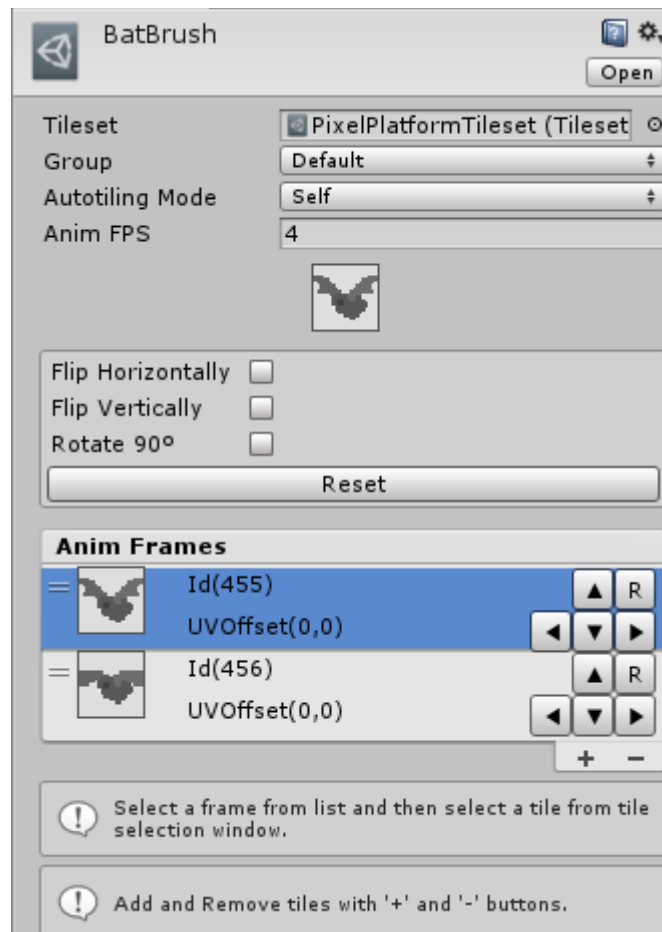
Brushes are scriptable objects derived from **TilesetBrush** and implementing the interface **IBrush**.

All have a common attribute **Tileset**, so they are associated to an specific tileset.

### 3.6.1 AnimBrush

To create an animated brush go to:

**"Assets/Create/SuperTilemapEditor/Brush/AnimBrush"**



First set the tileset used by the brush.

Anim FPS will be the frames per second of the animation.

To add a frame to the animation, press the '+' button, and press '-' button to remove the selected frame.

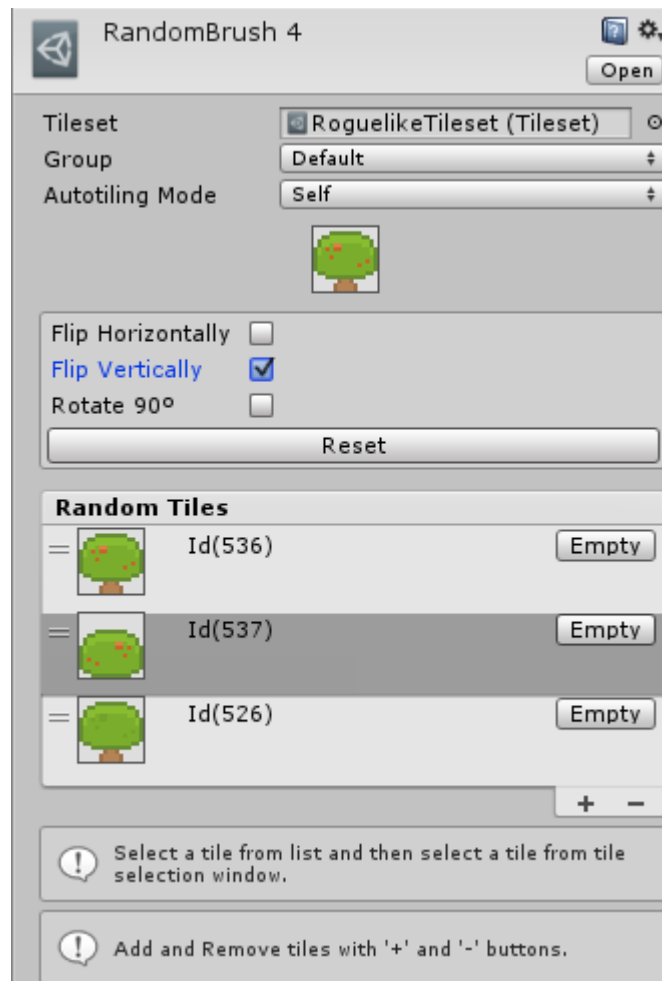
To change the frame tile, select the frame from the list and then select a tile from the tile palette.

You can also add an offset to the frame by pressing the arrow keys or reset the offset by pressing 'R'.

### 3.6.2 RandomBrush

To create an animated brush go to:

**"Assets/Create/SuperTilemapEditor/Brush/RandomBrush"**



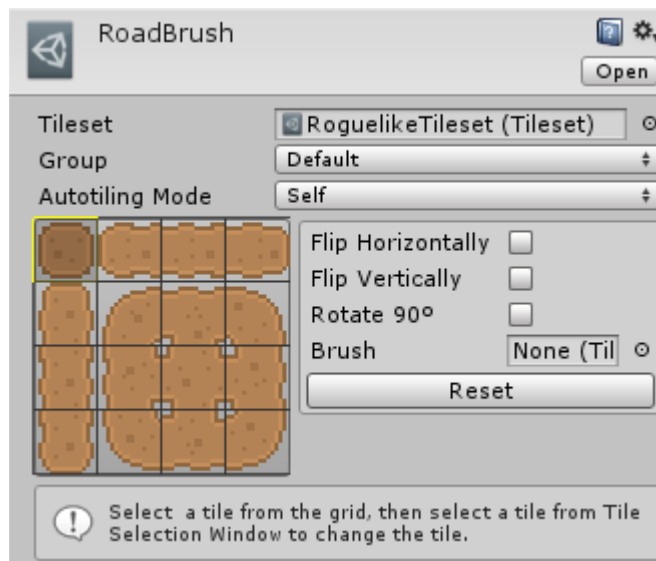
First set the tileset used by the brush.

Add or remove tiles from the random list by pressing '+' and '-' buttons.

The first tile will be the preview used in the brush palette.

To change the tile, select it and then select another tile from the tile palette.

### 3.6.3 RoadBrush

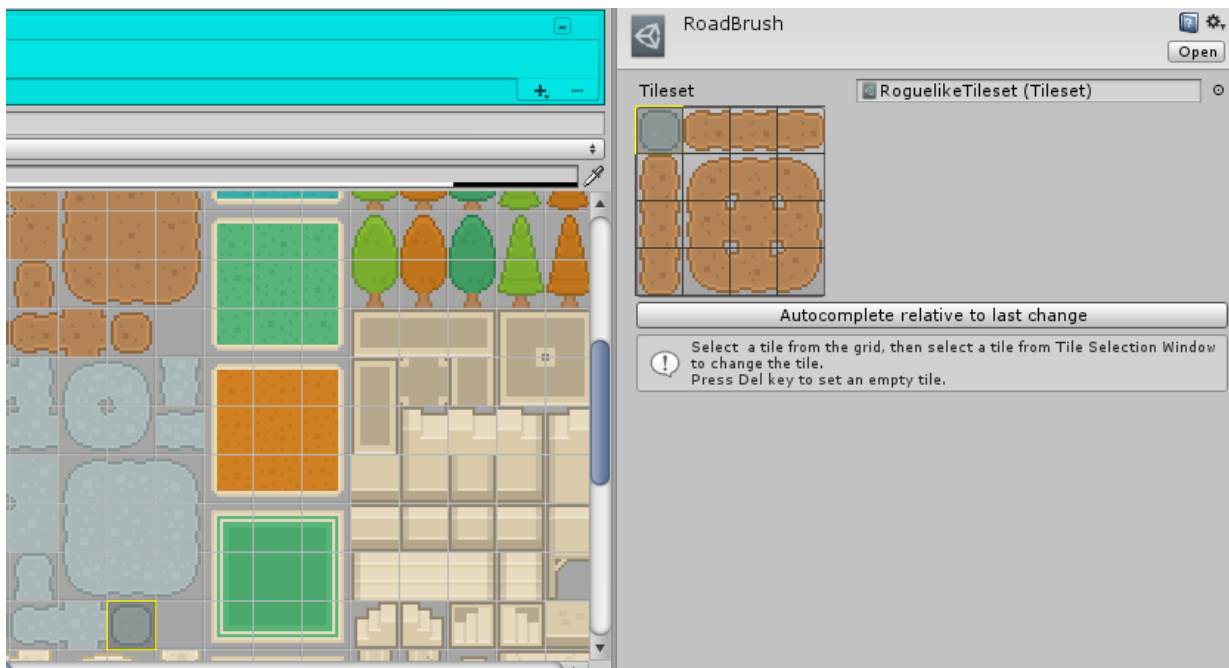


First set the tileset used by the brush.

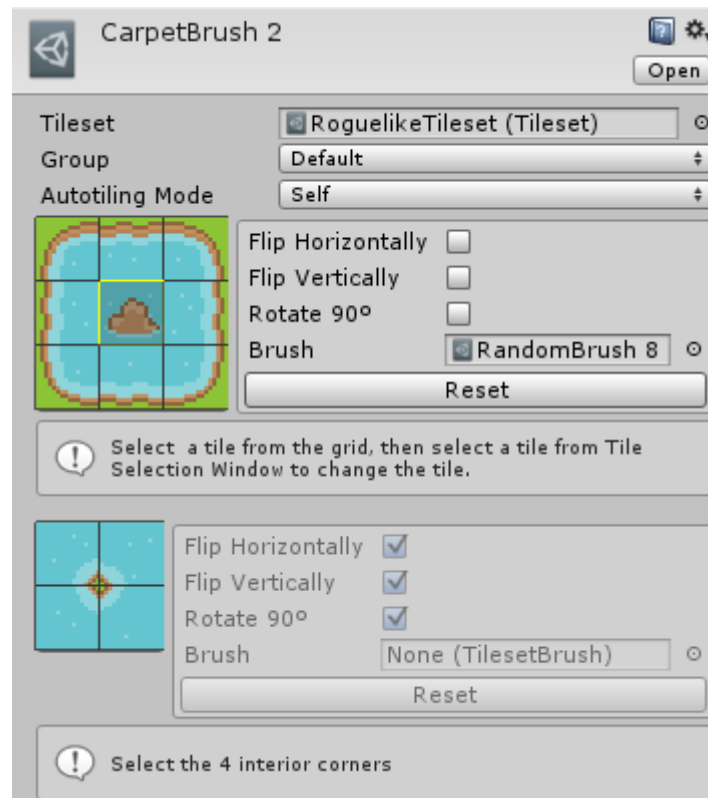
This brush is using 16 tiles for each road combination.

Follow the pattern and select one by one the tile from the brush and then select a tile from the tile palette to change it.

If road tiles are placed in same order in the tile atlas, you can create a new road brush faster by changing only one of the tiles of a complete road brush and the pressing the "Autocomplete" button.



### 3.6.4 CarpetBrush



First set the tileset used by the brush.

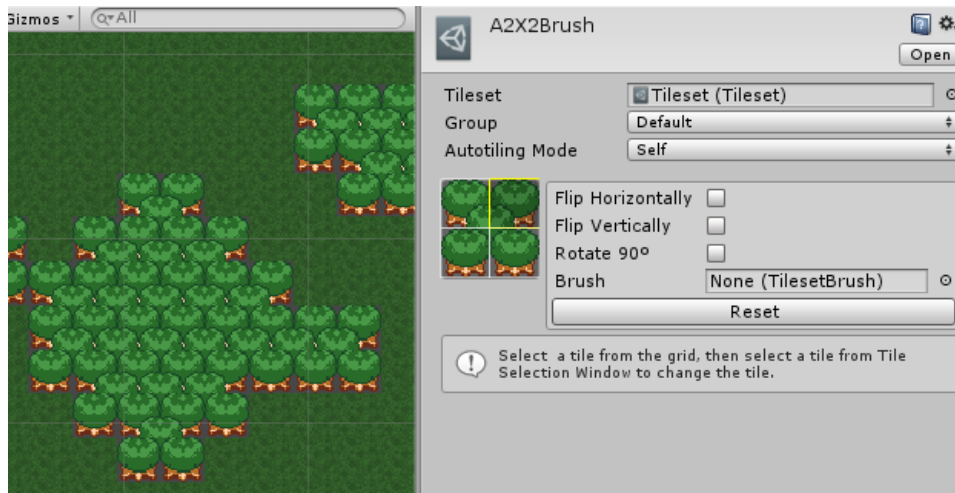
This brush is made to fill areas with interior corners.

Follow the pattern and select one by one the tile from the brush and then select a tile from the tile palette to change it.

If road tiles are placed in same order in the tile atlas, you can create a new road brush faster by changing only one of the tiles of a complete road brush and then pressing the "Autocomplete" button.

You have to autocomplete exterior and interior tiles separately.

### 3.6.5 A2x2Brush



First set the tileset used by the brush.

This brush is made of 4 tiles, one for each corner.

Follow the pattern and select one by one the tile from the brush and then select a tile from the tile palette to change it.

## 3.7 Custom Brushes

To create a new brush, you have to create a new class derived from **TilesetBrush** and implement the **IBrush** interface.

### 3.7.1 IBrush Interface

- **uint PreviewTileData()**: the tile data used to show the tile preview.
- **bool IsAnimated()**: return if the tile should be updated for animation.
- **Rect GetAnimUV()**: return the tile UV for the current frame.
- **uint GetAnimTileData()**: return the tile data of the current frame.
- **uint Refresh(Tilemap tilemap, int gridX, int gridY, uint tileData)**: this is called by the tilemap chunks when a tile needs to be refreshed. Return the updated tile data.
- **uint[] GetSubtiles(Tilemap tilemap, int gridX, int gridY, uint tileData)**: if a tile is divided in 4 subtiles, this is returning an array of 4 tile data, one per each subtile, from bottom to top, from left to right. Otherwise, it should return null.



### 3.7.2 TileData

The tile data contains the information used to draw a tilemap tile.

A tile data is stored in a single unsigned int:

- 16-bits[0-15] are used to store the **tileId**
- 12-bits[16-27] are used to store the **brushId** used to draw the tile
- 4bits[28,29,30,31] are used: 28 tile **updated**; 29 for **90 degrees rotation**; 30 for **vertical flip** and 31 for **horizontal flip**.

If you need to know the rotation degrees of a tile, you have to check the value [0, 1] of the flags Rot90 (90 degrees rotation), FlipV (flip vertical) and flipH (flip horizontal):

Rotation	Flip Vertical	Flip Horizontal	Rot90
0°	0	0	0
90°	0	0	1
180°	1	1	0
270°	1	1	1

To manage uint values of tiledata you need to know how to works with bitwise operators. You can find in Tilesset class all the constants you need for setting and getting data from tile data, but to make it easier, there is a class TileData you can use to deal with tile data.

Let's see some examples of using this class:

- **Read tile data from tilemap position (12, 45):**  

```
uint rawTileData = tilemap.GetTileData(12, 45);
TileData data = new TileData(rawTileData);
bool flipVertical = data.flipVertical;
bool flipHorizontal = data.flipHorizontal ;
bool rot90 = data.rot90 ;
bool brushId = data.brushId ;
bool tileId = data.tileId ;
```

- **Write tile data at tilemap position (50, 13)**

```
TileData data = new TileData();
```

```
//Fill all data you need in data ( remember a data without brushId or  
tileId is considered as an empty tile )
```

```
tilemap.SetTileData(12, 45, data.BuildData());
```