Avesta: 2D Tilemap Generator

Changelog

v2.1.0

- Add Destructible Tile support

v2.0.0

- Add **Layer support**! Now you can generate multiple layers using different algorithms
- Now main entry point is **LevelGenerator** script
- Add **Basic Random Algorithm**
- Add **Full Grid Algorithm**
- Add **Path Algorithm**
- Add **Random Walk Algorithm**

v1.2.0

- Code performance optimization. Now it is faster than before!
- Add important help sections under “**Window/Avesta**”

v1.1.1

- Add **MapObjectPlacer** to make it possible to place any object within the terrain on the Tilemap
- Code cleanup

v1.0

- First version of Avesta. It supports random terrain generation

Note: If you are upgrading from v1.+ please remove everything first and then install Avesta v2.0.0

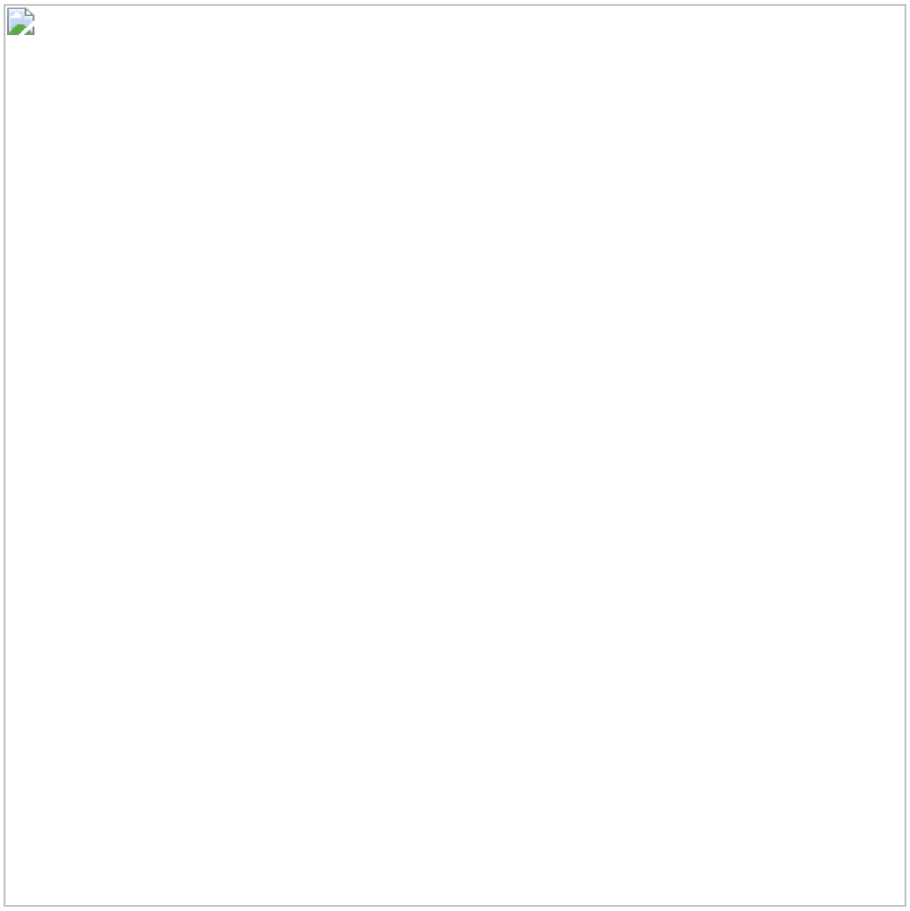
Introduction

First of all, we would like to personally thank you for trying out this asset! We hope it helps you on your game dev journey.

https://www.youtube.com/embed/kMmgDftX-Qg

What is Avesta?

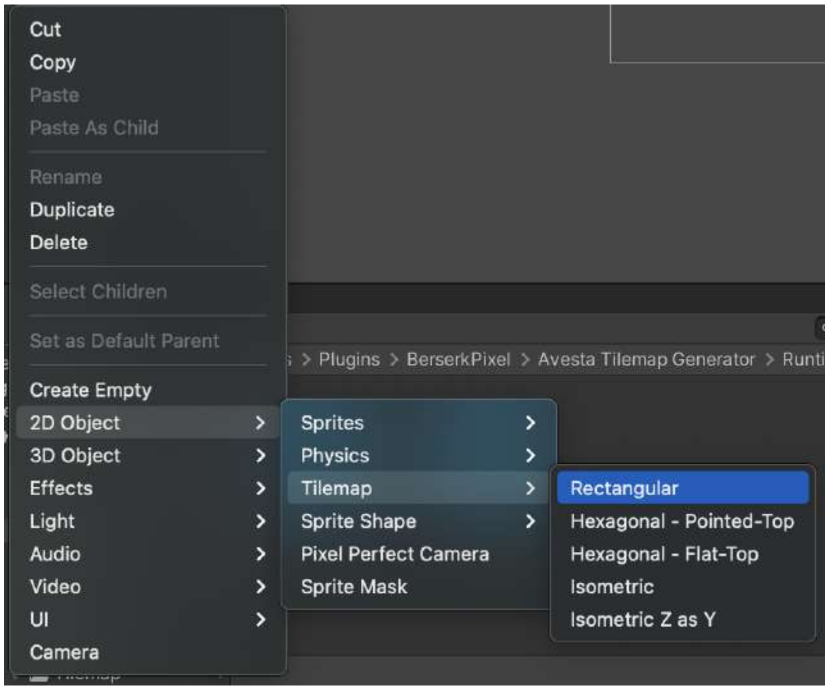
Avesta is an easy-to-use random Tilemap Generator Tool. It contains several algorithms for you to play around. This is not a replacement for a normal Tilemap, it's a creative tool to start designing your world!



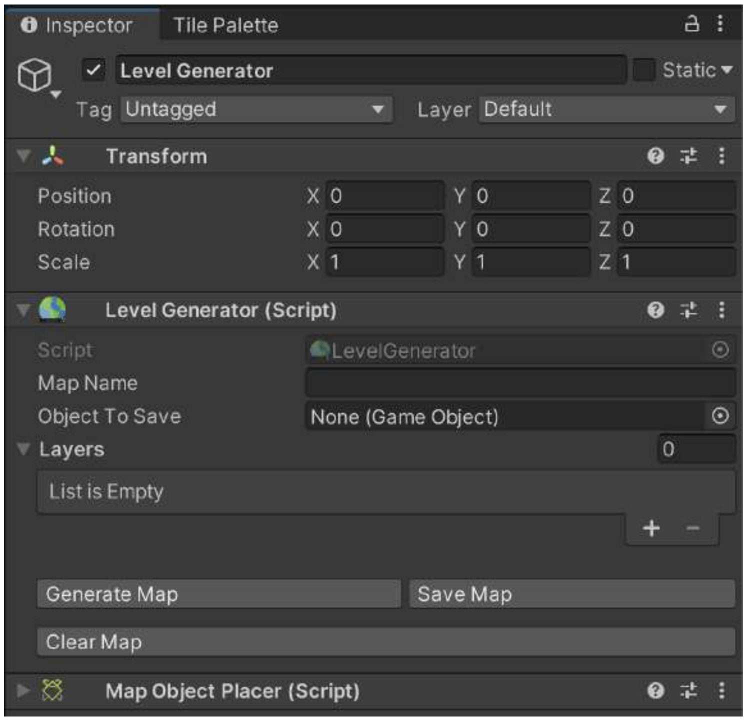
Start using Avesta

Setup

First of all, make sure you have a *Rectangular Tilemap* in your Scene by right clicking on your Hierarchy, **2D Object -> Tilemap -> Rectangular**. This will create a GameObject with a **Grid** script attached to it and with a *child GameObject* with a **Tilemap** component attached.



Once you have your Rectangular Tilemap in place, drag the **Level Generator** prefab from Runtime/Prefabs folder to your scene. Click the added prefab to see it on the Inspector.

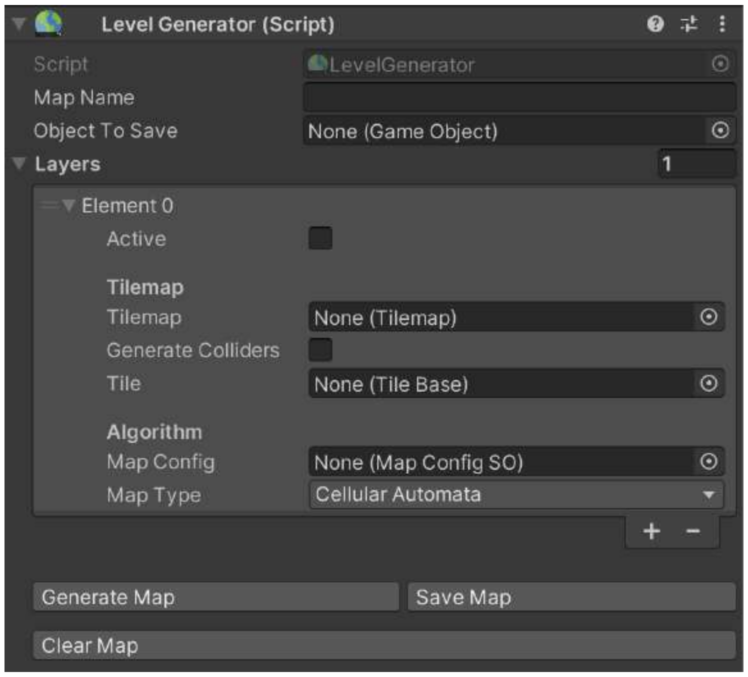


Level Generator
Tag Untagged
Layer Default

Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Script LevelGenerator
Map Name
Object To Save None (Game Object)
Layers 0

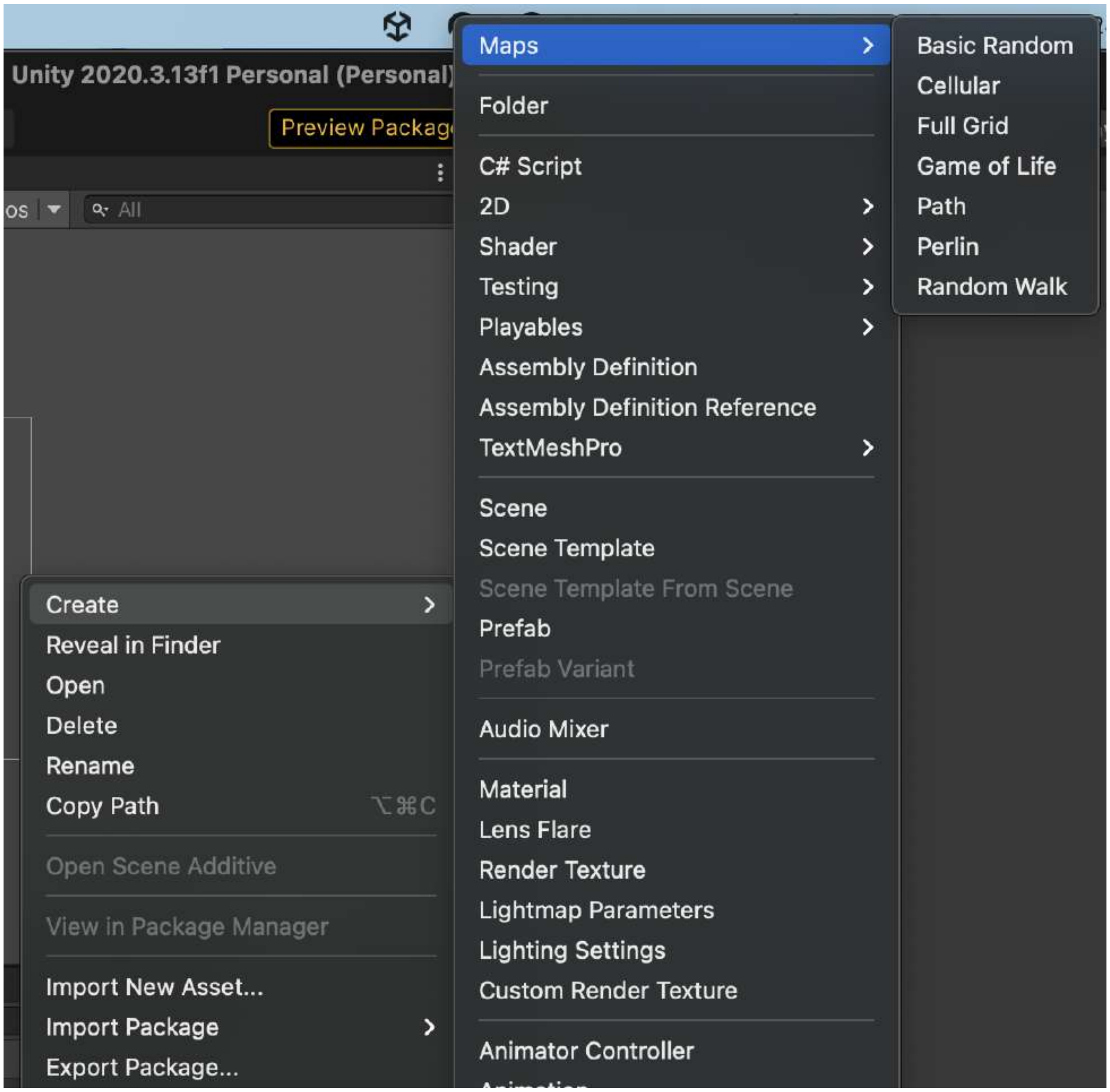
You will notice it doesn't do much... yet. As from **Avesta v2.0.0** you have support for multiple layers, each with different configurations. Now, change that "*0*" to the right of the "*Layers*" attribute to a "**1**".



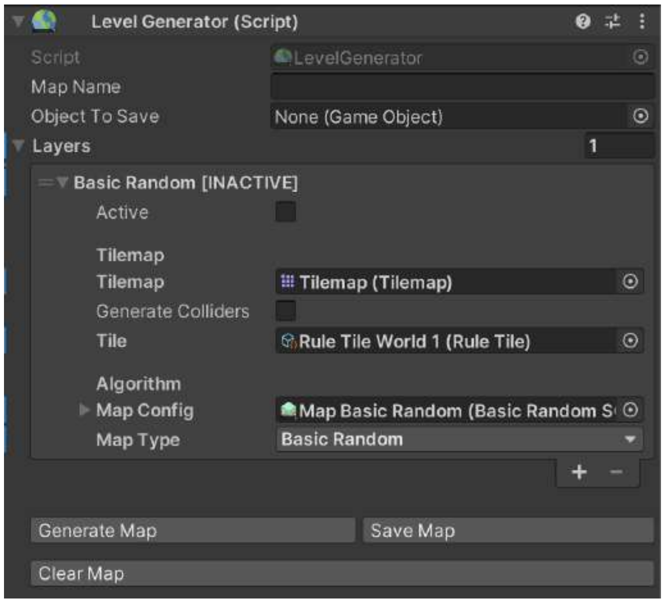
This is the first step of your journey. Drag the previously created Tilemap to the *Tilemap field*, choose a tile (or a Rule Tile for example) and put it into the **Tile** field.

Next you need a **MapConfiguration** file. In any folder you want, right click then:

Create → Maps → Choose any of this algorithms.



After that you should see a new file created. You can edit this file or even better, just drag it into the **LevelGenerator's** layer **Map Config** field under the *Algorithm* header

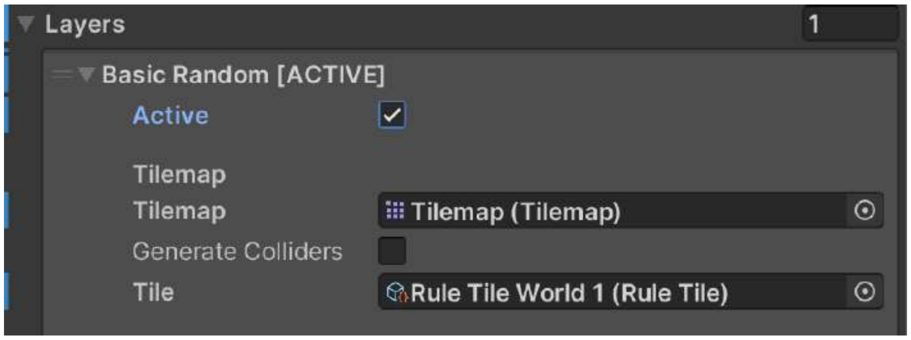

Level Generator (Script)

Script	LevelGenerator
Map Name	
Object To Save	None (Game Object)
Layers	1

Basic Random [INACTIVE]

Active	☐
Tilemap	
Tilemap	Tilemap (Tilemap)
Generate Colliders	☐
Tile	Rule Tile World 1 (Rule Tile)
Algorithm	
Map Config	Map Basic Random (Basic Random S
Map Type	Basic Random

As you can see, the name for this Layer has changed matching the algorithm's name as well! Also you can see on the picture above that this layer is **Inactive** by default. Let's change that but clicking the "Active" property under the title:


Layers 1

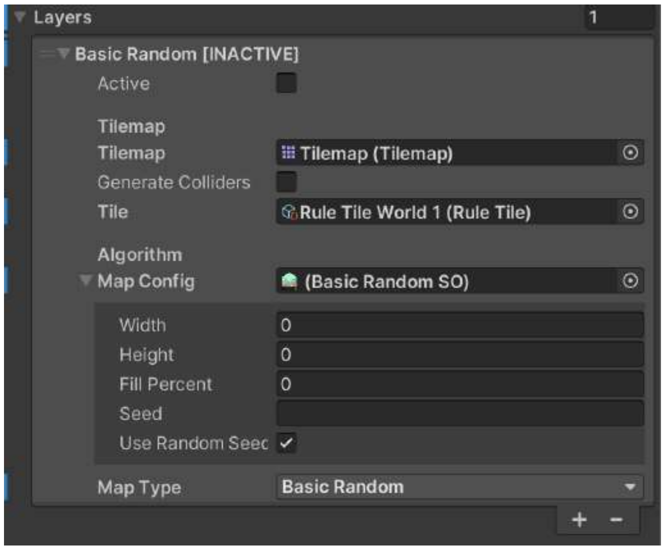
Basic Random [ACTIVE]

Active	✓
Tilemap	
Tilemap	Tilemap (Tilemap)
Generate Colliders	☐
Tile	Rule Tile World 1 (Rule Tile)

This again will change the title to display the status of this layer. This is useful whenever one is playing around with different algorithms and layers to create the best possible map.

Also, the "**Generate Colliders**" field helps us creating standard collision detection for the given Tilemap. This is super useful if you have a Walls map layer for example.

From here you can edit all the configurations for that specific algorithm.

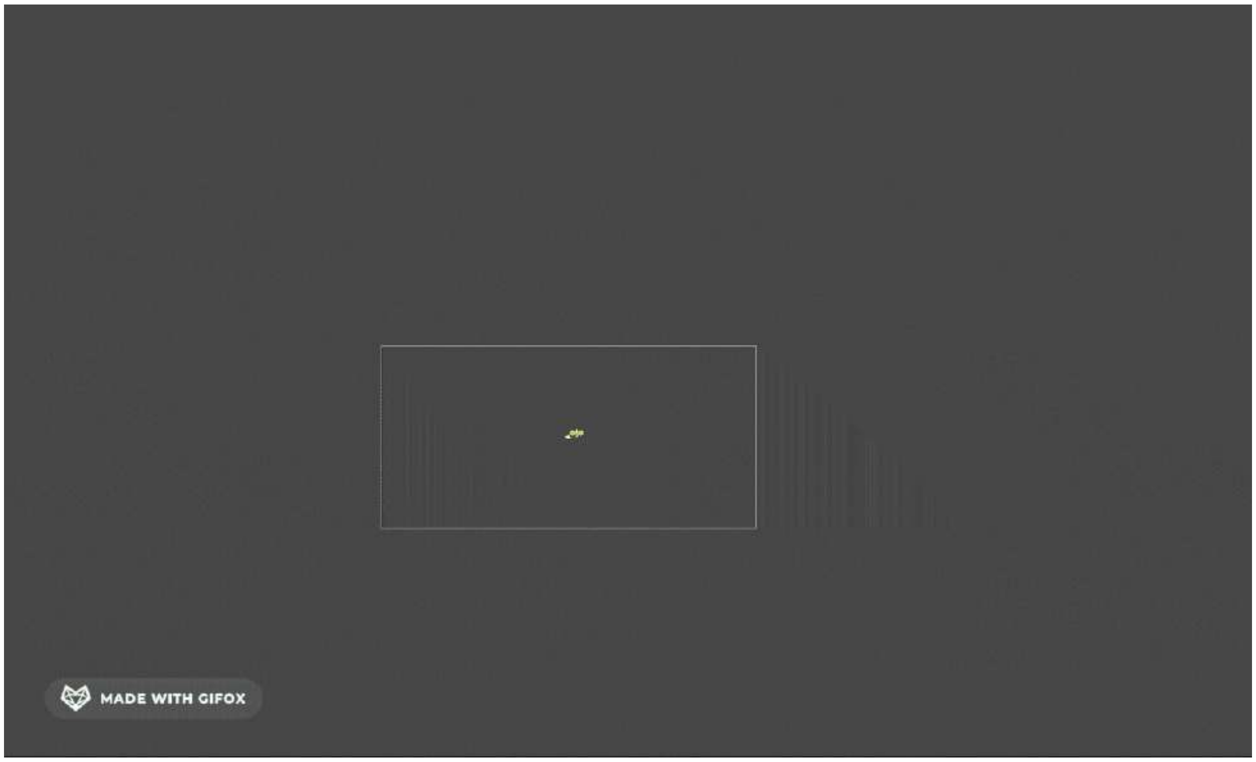


Change some parameters such as the **width** and **height** and then press the "**Generate Map"** button below!

The "**Clear Map**" button just resets all Tilemaps on the active layers, clearing all the tiles on them. This will not reset any configuration made on that layer, just the visuals and the collisions if the "*Generate Colliders*" is enabled.

The "**Save Map**" button creates a new *Prefab* of any GameObject you drag on the **Object to Save** field. This will preserve all the configurations. This is useful when you want to create chunks of maps so afterwards you can just drag them to your Scene easily.

Prefabs are saved on a new folder "**Assets/Prefabs/Maps/Generated/**" with a distinctive name or alternatively you can name the prefab using the "**Map Name**" field on the Inspector. After saving you can just Drag and Drop that prefab on any scene and use it as a normal Tilemap.

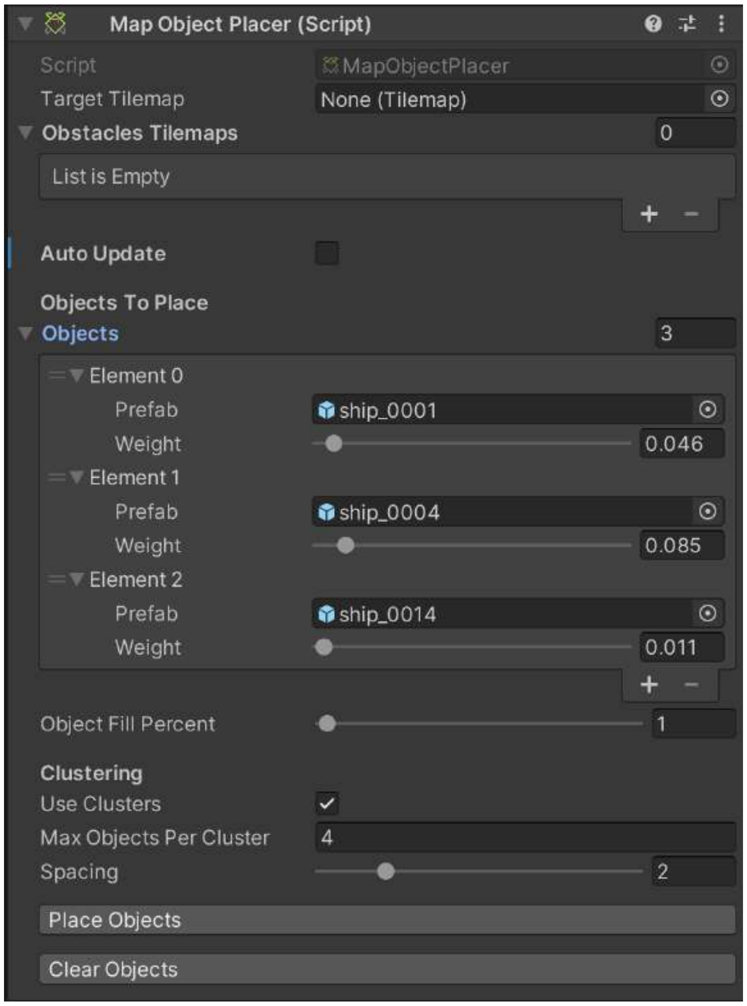


Finally you can drag and drop your character inside the map and play around. We provide 2 sample scripts for a simple player movement and a camera follow so you can test around your newly created map!

Placing Objects

We imagined that you would like to place objects on your Tilemap and this has never been as simple as with **Avesta**!

Attach the **MapObjectPlacer** script to your *Level Generator* game object (or any other you want).



Map Object Placer (Script)

Script	MapObjectPlacer
Target Tilemap	None (Tilemap)
Obstacles Tilemaps	0
List is Empty	
Auto Update	☐
Objects To Place	
Objects	3
Element 0	
Prefab	ship_0001
Weight	0.046
Element 1	
Prefab	ship_0004
Weight	0.085
Element 2	
Prefab	ship_0014
Weight	0.011
Object Fill Percent	1
Clustering	
Use Clusters	✓
Max Objects Per Cluster	4
Spacing	2

Drag the target Tilemap where you want the objects to be placed. If you have any other Tilemap that is used as *water* or *walls* and you want to avoid those ones when placing objects, just drag them into the "**Obstacles Tilemaps**" field.

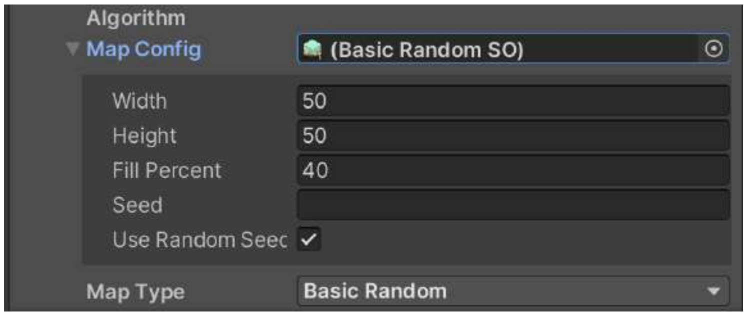
Change the Objects size to match how many different type of prefabs you would like to place. Drag your objects on the fields and play around with the weight. The ***bigger the weight, the more chances it has to appear.*** Adjust your **Object Fill Percent** for the total amount of objects to place and finally press the **Place Objects** button.

The "**Clear Objects**" button just removes all the objects you placed on that given **Target Tilemap** field.

Supported Algorithms Configurations

All configurations have **Width** and **Height** fields that needs to match to whatever **dimensions** you would like this layer to have.

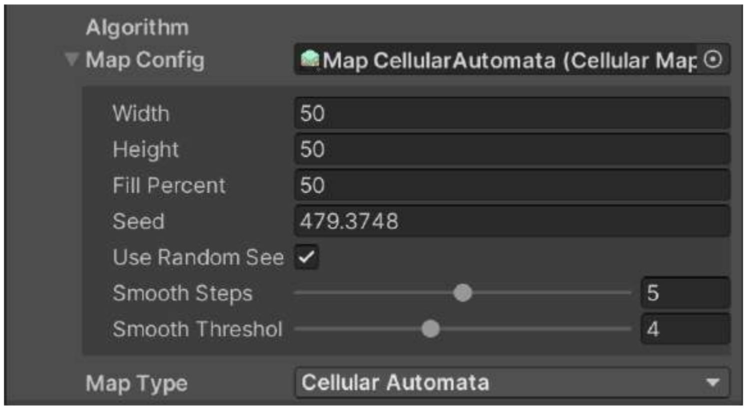
Basic Random



Algorithm	
Map Config	(Basic Random SO)
Width	50
Height	50
Fill Percent	40
Seed	
Use Random Seec	✓
Map Type	Basic Random

- **Fill Percent [*float*]**: how much do you want to “paint” the tilemap. The higher the number the more tiles will it have.
- **Seed [*string*]**: just a random seed for the generator. It will be overridden if the “Use Random Seed” is enabled.
- **Use Random Seed [*bool*]**: toggle for using or not a random seed.

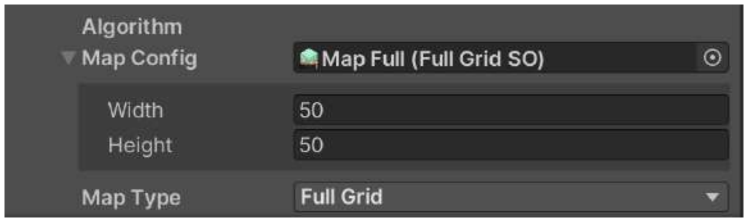
Cellular Automata



Algorithm	
Map Config	Map CellularAutomata (Cellular Mar
Width	50
Height	50
Fill Percent	50
Seed	479.3748
Use Random See	✓
Smooth Steps	5
Smooth Threshol	4
Map Type	Cellular Automata

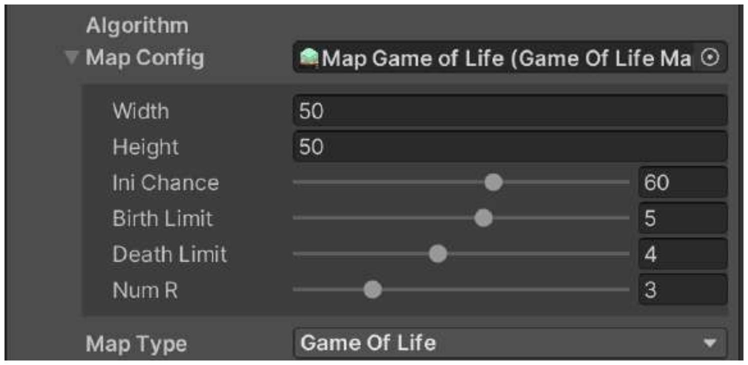
- **Fill Percent [*float*]**: how much do you want to “paint” the tilemap. The higher the number the more tiles will it have.
- **Seed [*string*]**: just a random seed for the generator. It will be overridden if the “Use Random Seed” is enabled.
- **Use Random Seed [*bool*]**: toggle for using or not a random seed.
- **Smooth Steps [*int range(0-10)*]**: How much to smooth the generated map.
- **Smooth Threshold [*int range(0-10)*]**: The threshold the smooth step will have.

Full Grid


Algorithm
Map Config: Map Full (Full Grid SO)
Width: 50
Height: 50
Map Type: Full Grid

Basic configuration. It will fill the entire tilemap with the desired dimensions.

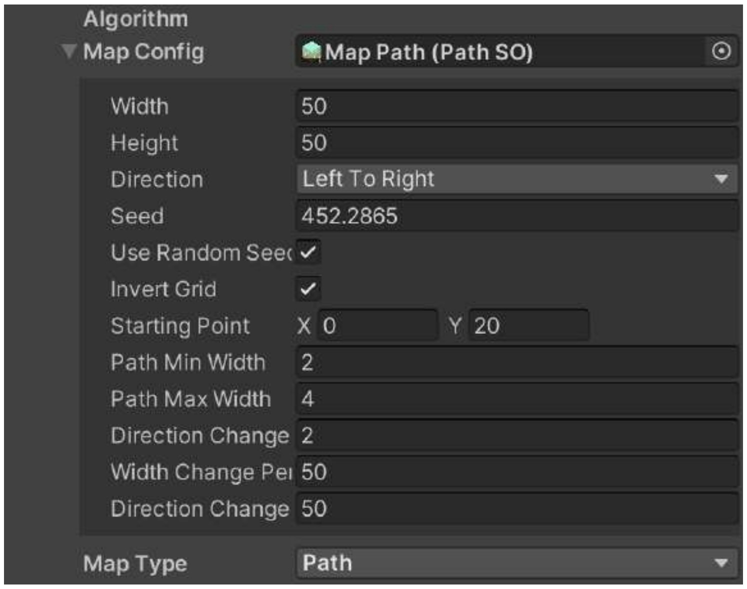
Game Of Life


Algorithm
Map Config: Map Game of Life (Game Of Life Ma
Width: 50
Height: 50
Ini Chance: 60
Birth Limit: 5
Death Limit: 4
Num R: 3
Map Type: Game Of Life

- **Ini Chance [*int range(0-100)*]**: The initial chance for the algorithm.
- **Birth Limit [*int range(1-8)*]**: Limits the number of iterations.
- **Death Limit [*int range(1-8)*]**: Limits the amount of deaths.
- **Num R[*int range(1-10)*]**: Total number of runs/iterations.

If you want to know more about this algorithm here is a good resource

Path



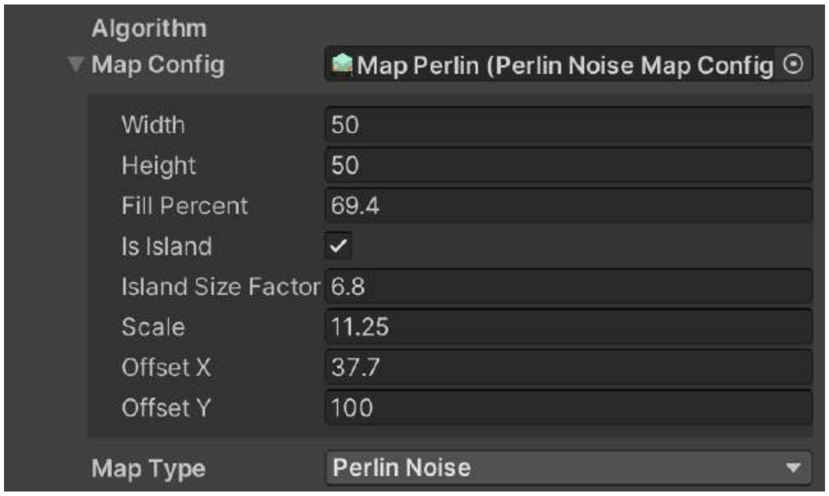
Algorithm
Map Config: Map Path (Path SO)

Width	50
Height	50
Direction	Left To Right
Seed	452.2865
Use Random Seed	✓
Invert Grid	✓
Starting Point	X 0 Y 20
Path Min Width	2
Path Max Width	4
Direction Change	2
Width Change Per	50
Direction Change	50

Map Type: Path

- **Direction [*TopToBottom, BottomToTop, LeftToRight, RightToLeft*]**: The direction where the path should move.
- **Seed [*string*]**: just a random seed for the generator. It will be overridden if the “Use Random Seed” is enabled.
- **Use Random Seed [*bool*]**: toggle for using or not a random seed.
- **Invert Grid [*bool*]**: useful if you are trying to achieve either a cave or an island/walking path look.
- **Starting Point [*Vector2Int*]**: where should the path start.
- **Path Min Width [*int*]**: the minimum units the path can have.
- **Path Max Width [*int*]**: the maximum units the path can have.
- **Direction Change Distance [*int*]**: the amount in units the direction can change.
- **Width Change Percentage [*int*]**: a percentage for changing the width.
- **Direction Change Percentage [*int*]**: a percentage for changing the direction.

Perlin Noise



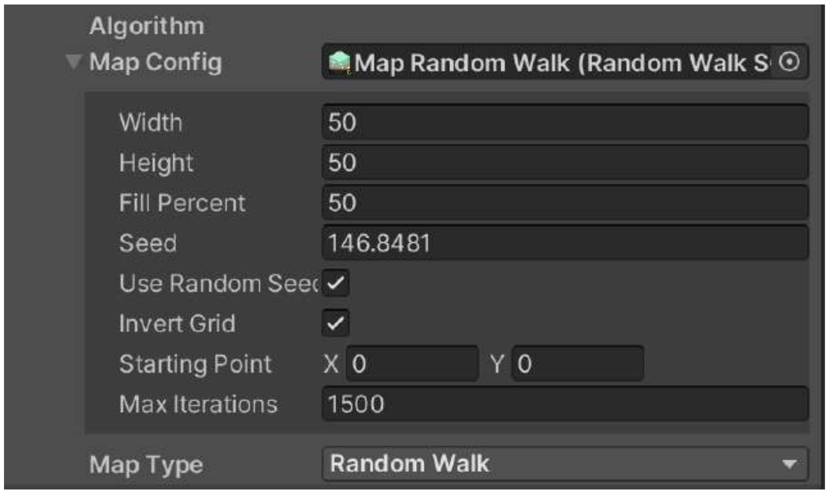
Algorithm
Map Config: Map Perlin (Perlin Noise Map Config

Width	50
Height	50
Fill Percent	69.4
Is Island	✓
Island Size Factor	6.8
Scale	11.25
Offset X	37.7
Offset Y	100

Map Type: Perlin Noise

- **Fill Percent [*float*]**: how much do you want to “paint” the tilemap. The higher the number the more tiles will it have.
- **Is Island [*bool*]**: if enabled it will create an island-type-of-map (no walls)
- **Island Size Factor [*float*]**: how big the island should be.
- **Scale [*float*]**: the whole scale of the map. It’s kind of the “zoom” factor the map will have.
- **Offset X [*float*]**: the offset on the X axis.
- **Offset Y [*float*]**: the offset on the Y axis.

Random Walk



Algorithm
Map Config: Map Random Walk (Random Walk S

Width	50
Height	50
Fill Percent	50
Seed	146.8481
Use Random Seec	✓
Invert Grid	✓
Starting Point	X 0 Y 0
Max Iterations	1500

Map Type: Random Walk

- **Fill Percent [*float*]**: how much do you want to “paint” the tilemap. The higher the number the more tiles will it have.

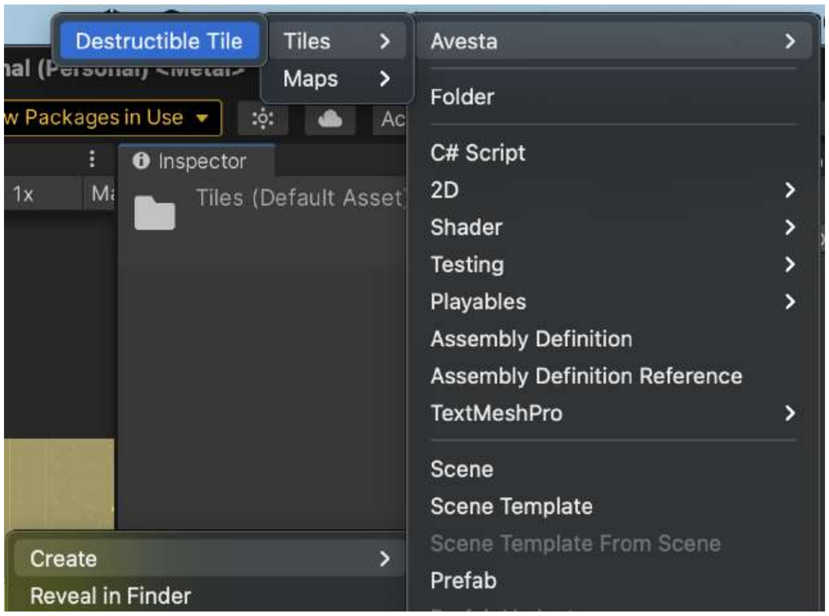
- **Seed [*string*]**: just a random seed for the generator. It will be overridden if the "Use Random Seed" is enabled.
- **Use Random Seed [*bool*]**: toggle for using or not a random seed.
- **Invert Grid [*bool*]**: useful if you are trying to achieve either a cave or an island/walking path look.
- **Starting Point [*Vector2Int*]**: where should the map start.
- **Max Iterations [i*nt*]**: how many iterations should be done for the final map. 1000-1500 iterations have been prove to be a nice result on our tests but of course it's up to you.

Destructible Tilemaps

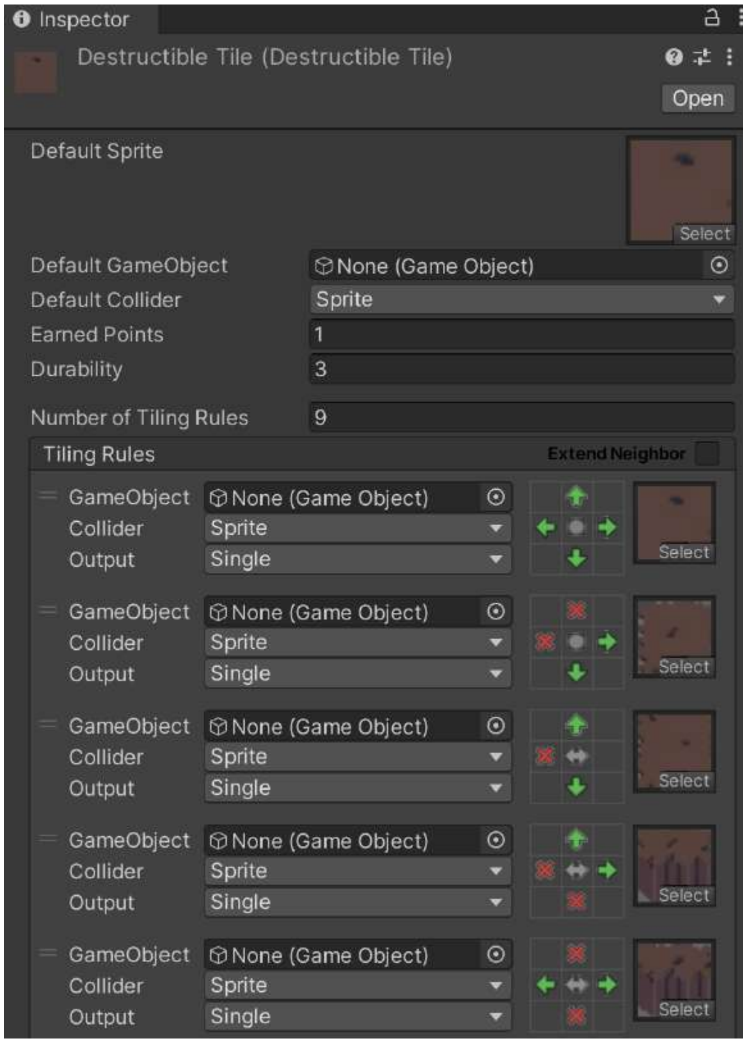
Welcome to the world of destruction! Creating 2D destructible maps has never been easier.

Destructible Tile

The first step is to create a Destructible Tile. For this right click on your Project window go to **Create** → **Avesta** → **Tiles** → **Destructible Tile**



This will create a new asset



Inspector

Destructible Tile (Destructible Tile)

Open

Default Sprite — Select

Default GameObject: None (Game Object)

Default Collider: Sprite

Earned Points: 1

Durability: 3

Number of Tiling Rules: 9

Tiling Rules — Extend Neighbor

GameObject: None (Game Object)
Collider: Sprite
Output: Single

GameObject: None (Game Object)
Collider: Sprite
Output: Single

GameObject: None (Game Object)
Collider: Sprite
Output: Single

GameObject: None (Game Object)
Collider: Sprite
Output: Single

GameObject: None (Game Object)
Collider: Sprite
Output: Single

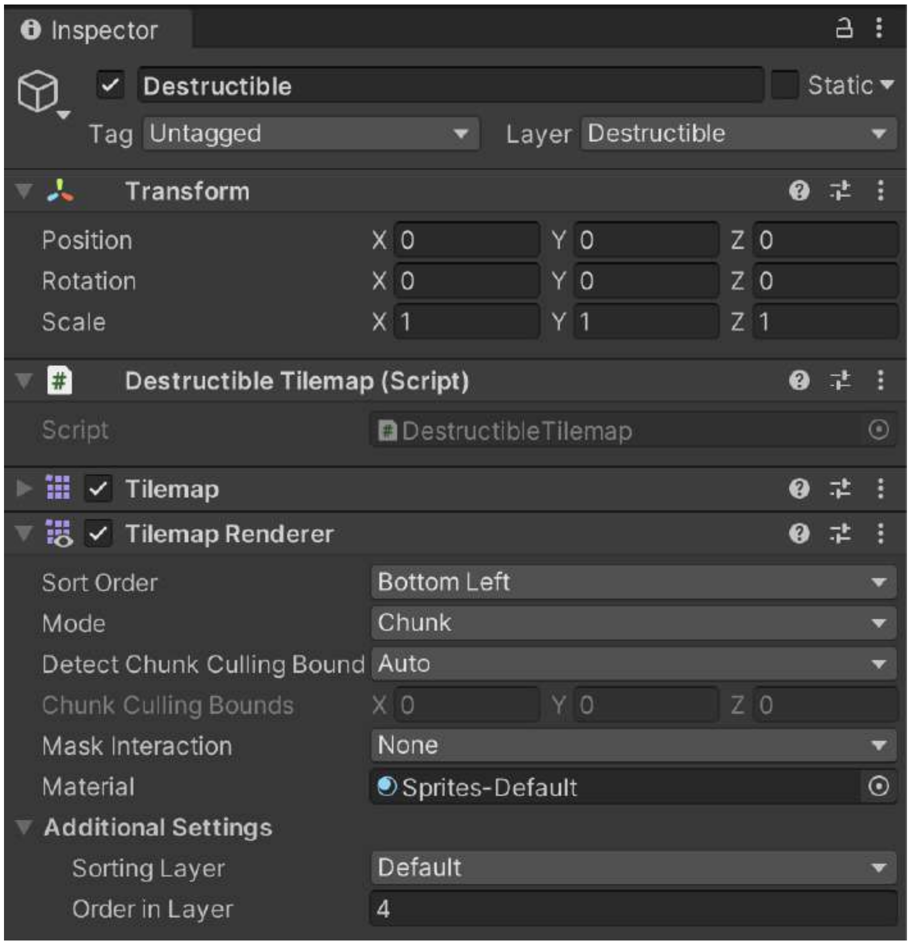
Destructible Tile inherits from the Rule Tile. It has the same functionality but it adds some fields

- **Earned Points**: just an integer to be returned when a tile is destroyed. This is useful if you want to earn “gold”, “coins”, “experience points” or anything of that sort to unlock new features on your game.
- **Durability**: in simple words, it’s the “health” of each tile. The amount of hits it can handle.

The rest is just as any other Rule Tile you had done before.

Destructible Tilemap

For a normal Tilemap to have Destructible capabilities, just add the Destructible Tilemap script to it. Also you can create/change the Layer of this object so it makes it easier afterwards to interact with it.


Inspector

✓ Destructible — Static
Tag Untagged — Layer Destructible

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

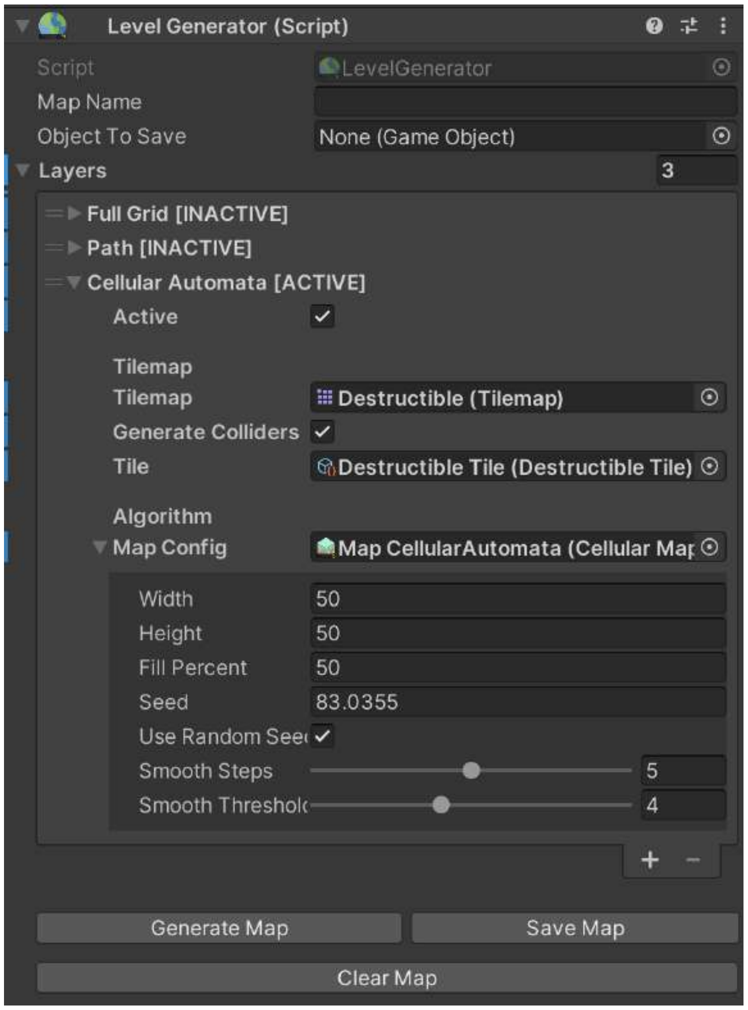
Destructible Tilemap (Script)
Script DestructibleTilemap

✓ Tilemap

✓ Tilemap Renderer
Sort Order Bottom Left
Mode Chunk
Detect Chunk Culling Bound Auto
Chunk Culling Bounds X 0 Y 0 Z 0
Mask Interaction None
Material Sprites-Default
Additional Settings
Sorting Layer Default
Order in Layer 4

Note: at this point you can also add a TilemapShadow to this Tilemap. Be aware about the Order In Layer. Adjust it accordingly so it looks good.

Once the script is added you can use it normally in your Level Generator script as any other Tilemap. Drag it and also drag the previously created Destructible Tile.



Level Generator (Script)

Script	LevelGenerator
Map Name	
Object To Save	None (Game Object)
Layers	3

Full Grid [INACTIVE]
Path [INACTIVE]
Cellular Automata [ACTIVE]

Active	✓

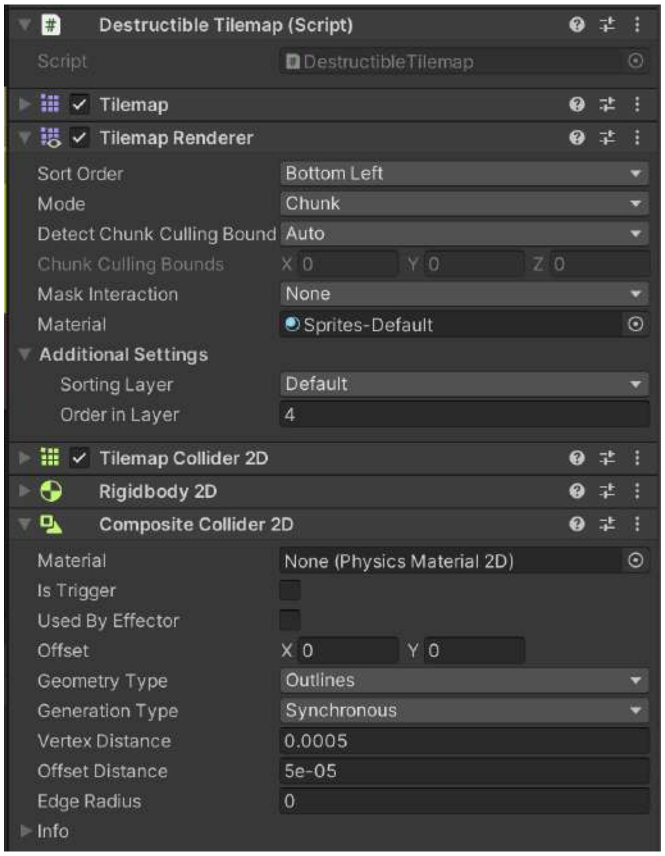
Tilemap

Tilemap	Destructible (Tilemap)
Generate Colliders	✓
Tile	Destructible Tile (Destructible Tile)

Algorithm

Map Config	Map CellularAutomata (Cellular Map
Width	50
Height	50
Fill Percent	50
Seed	83.0355
Use Random Seed	✓
Smooth Steps	5
Smooth Threshold	4

Notice that we are generating the colliders. This way we can detect the collisions and destroy the tiles accordingly.


Destructible Tilemap (Script)

Script	DestructibleTilemap

Tilemap

Tilemap Renderer

Sort Order	Bottom Left
Mode	Chunk
Detect Chunk Culling Bound	Auto
Chunk Culling Bounds	X 0 Y 0 Z 0
Mask Interaction	None
Material	Sprites-Default
Additional Settings	
Sorting Layer	Default
Order in Layer	4

Tilemap Collider 2D

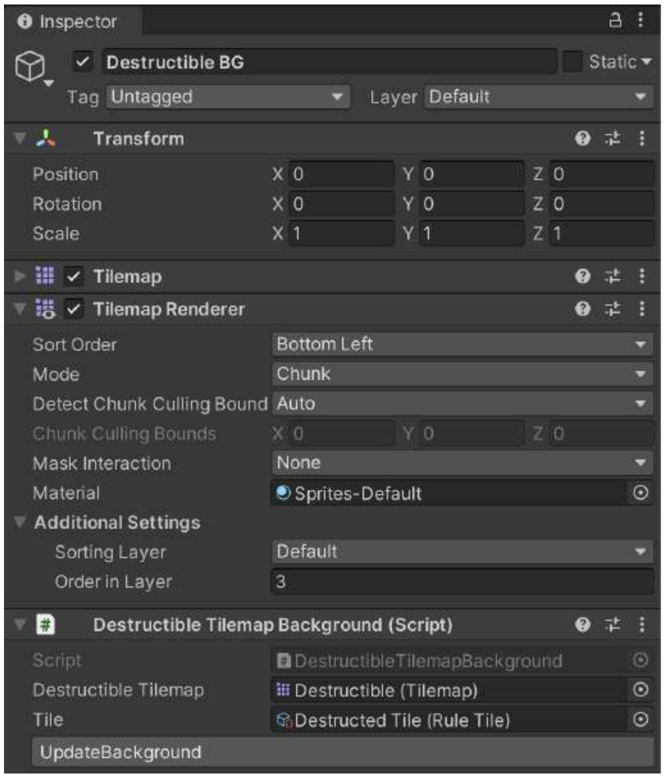
Rigidbody 2D

Composite Collider 2D

Material	None (Physics Material 2D)
Is Trigger	
Used By Effector	
Offset	X 0 Y 0
Geometry Type	Outlines
Generation Type	Synchronous
Vertex Distance	0.0005
Offset Distance	5e-05
Edge Radius	0
Info	

In the demo scene we are using a **Geometry Type** of **Outlines** since it's more performant but Polygons works as well.

You can also add a "destroyed tile" status when a tile is destroyed (a simple visual like a ground tile). To do this, **create a new Tilemap** object on your hierarchy. **Add** the **Destructible Tilemap Background script** to it and drag the destructed tile and tilemap to follow. Afterwards just click on the "Update Background" button. This will copy the reference Tilemap and add the visual tile.



```
Inspector
Destructible BG                       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

Tilemap
Tilemap Renderer
Sort Order                   Bottom Left
Mode                         Chunk
Detect Chunk Culling Bound   Auto
Chunk Culling Bounds         X 0   Y 0   Z 0
Mask Interaction             None
Material                     Sprites-Default
Additional Settings
  Sorting Layer              Default
  Order in Layer             3

Destructible Tilemap Background (Script)
Script                       DestructibleTilemapBackground
Destructible Tilemap         Destructible (Tilemap)
Tile                         Destructed Tile (Rule Tile)
UpdateBackground
```

Note: this tilemap should be rendered below the main Destructible Tilemap. Just change the Order In Layer to do this.

If you have any doubts, you can see this in work in the Destructible Scene.

Destroying Tiles

> *Too much blah blah until now, let me destroy things!*

To destroy tiles we need a new script. This one could be attached to the player or anything you want. On this script you just need to have a point in the world and send it to the destructible tilemap.

In this example we will see how we can destroy a tile when the mouse is clicked.

```
public class ClickAndDestroy : MonoBehaviour
{
    // the Layer of the Destructible Tilemap
    [SerializeField] private LayerMask targetMask;
    // how big we need to check
    [SerializeField] private float checkRadius = .4f;
    // Depending on the type of game you could have different tools/weapons with different amount of damages
    [SerializeField] private int tileDamage = 3;

    [Header("FX")]
    // show a visual on the grid
    [SerializeField] private Transform tileVisuals;
    [SerializeField] private ParticleSystem destroyParticles;
    private Camera _mainCamera;
    private Vector2 _mousePosition;

    private void Awake()
    {
        _mainCamera = Camera.main;
    }

    private void Update()
    {
        // get the moouse position
        _mousePosition = _mainCamera.ScreenToWorldPoint(Input.mousePosition);
        // check if collides with something using the LayerMask
        Collider2D overlapCircle = Physics2D.OverlapCircle(_mousePosition, checkRadius, targetMask);

        DestructibleTilemap destructibleTilemap = null;

        // display a visual if it hits the target mask
        if (overlapCircle != null && overlapCircle.TryGetComponent(out destructibleTilemap))
        {
            if (!tileVisuals.gameObject.activeSelf)
            {
                tileVisuals.gameObject.SetActive(true);
            }

            tileVisuals.position = destructibleTilemap.GetTileCenter(_mousePosition);
        }
        else
        {
            tileVisuals.gameObject.SetActive(false);
        }

        if (!Input.GetMouseButton(0) || overlapCircle == null || destructibleTilemap == null) return;

        // if we clicked on a proper destructible tile we perform the visuals and destroy the tile

        destroyParticles.transform.position = _mousePosition;
        destroyParticles.Play();
        // here we send the hit position where we clicked and the amount of damage to that tile.
        // depending on the type of game you could have different tools/weapons with different
        // amount of damages.
        destructibleTilemap.PerformContact(_mousePosition, tileDamage);
    }

    private void OnDrawGizmos()
    {
        Gizmos.color = Color.red;
        Gizmos.DrawWireSphere(_mousePosition, checkRadius);
    }
}
```

Extras

On the Samples folder you will find 3 different Scenes: **Demo Scene, Occlusion Scene** and **Destructible Scene**.

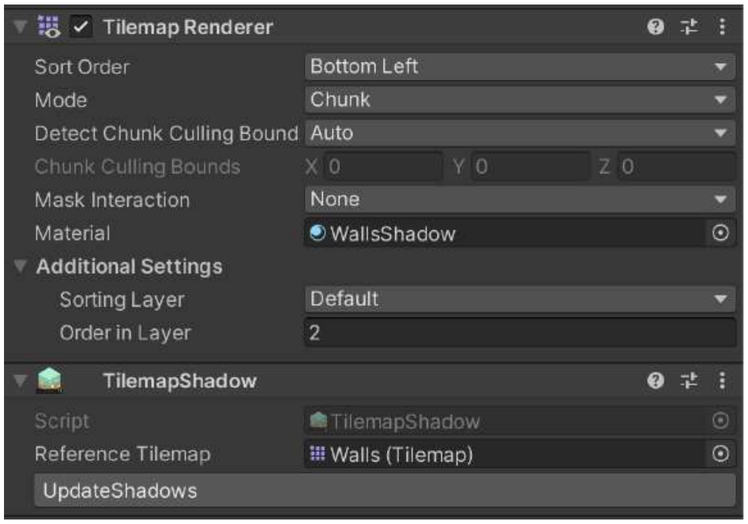
Demo Scene

This is a basic scene where you can play around with the LevelGenerator settings. Some things to notice are the Order in Layer for the different Tilemaps under the Additional Settings on the Tilemap Renderer.

For example, in a normal settings you would have

Layer	Order In Layer
Ground	0
Paths/Decorations	1
Wall Shadows	2
Walls	3

In the Demo, we provide a **TilemapShadow** script with a simple "shadow material" (just a normal gray color with alpha)



Tilemap Renderer

Sort Order	Bottom Left
Mode	Chunk
Detect Chunk Culling Bound	Auto
Chunk Culling Bounds	X 0 Y 0 Z 0
Mask Interaction	None
Material	WallsShadow
Additional Settings	
Sorting Layer	Default
Order in Layer	2

TilemapShadow

Script	TilemapShadow
Reference Tilemap	Walls (Tilemap)

UpdateShadows

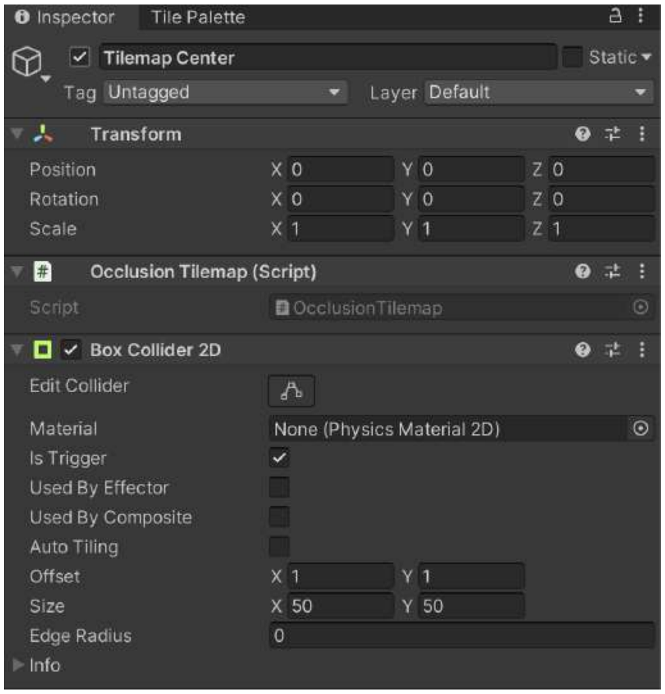
This script copies a tilemap and also offsets it downwards (0.25f down) of the reference tilemap.

Occlusion Demo Scene

This scene has the same functionality as the previous one but it adds a simple way of making it more performant but hiding and showing the tilemaps to the player while it moves.

The player has a **OcclusionCullingDetector** script on one of the child game objects. This one helps detecting a tilemap and activate it.

Another thing to note is that each Chunk container has an Occlusion Tilemap and a Trigger Collider components attached to them. This helps the **OcclusionCullingDetector** script detect them.


Inspector | Tile Palette

✓ Tilemap Center — Static
Tag: Untagged — Layer: Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

Occlusion Tilemap (Script)
Script: OcclusionTilemap

✓ Box Collider 2D

Edit Collider	
Material	None (Physics Material 2D)
Is Trigger	✓
Used By Effector	
Used By Composite	
Auto Tiling	
Offset	X 1, Y 1
Size	X 50, Y 50
Edge Radius	0

Info

Wrapping up!

Now you are set to start your own game! Play around with the settings in the inspector and come up with great worlds!

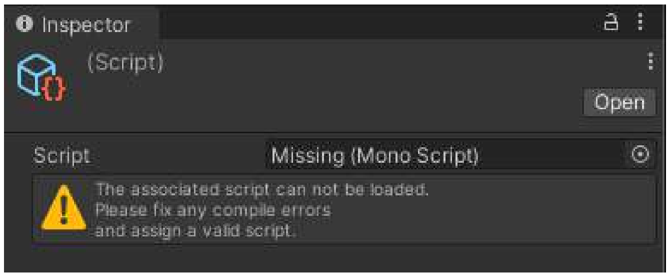
The possibilities of expansion are endless! Just use the Tilemap as usual afterwards to place different objects and tiles to tweak your world as it fits better.

Thanks again for trying Avesta! If you are curious about what else we have, check more of our content in our Shop on the Asset Store!

For any further question not covered here feel free to contact us at support@berserkpixel.studio

F.A.Q

Help! I can’t see the Rule tiles properly. It shows *Missing (Mono Script)*



Script Missing (Mono Script)

The associated script can not be loaded.
Please fix any compile errors
and assign a valid script.

This is because in order to use Rule Tiles, one has to install the 2D-Extras preview package. For safety reason we are not allowed to include a preview package into our assets. But it’s a super quick fix!

Solution 1

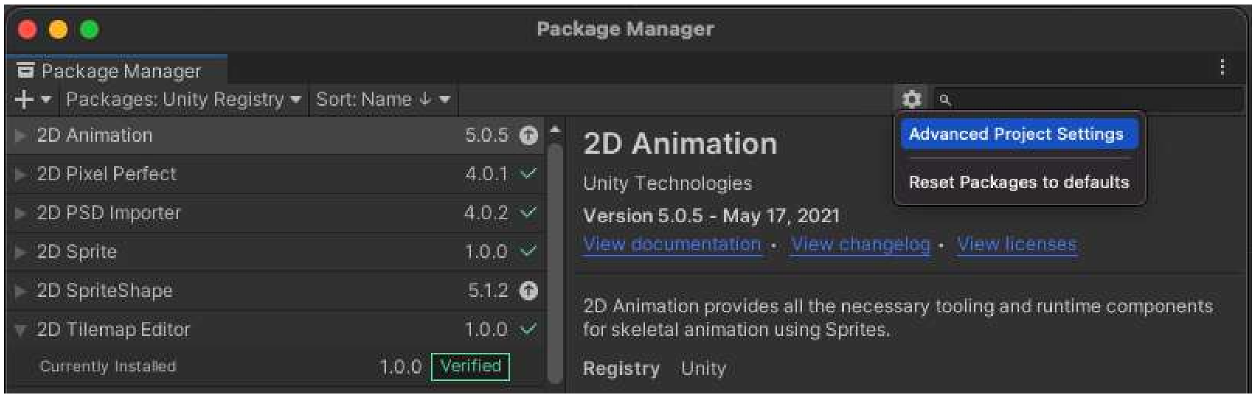
The following line needs to be added to your **Packages/manifest.json** file in your Unity Project under the dependencies section:

```
"com.unity.2d.tilemap.extras": "https://github.com/Unity-Technologies/2d-extras.git#master"
```

For more information please check out the official Github Repo: https://github.com/Unity-Technologies/2d-extras

Solution 2

Another way of adding it is going to you Package Manager window (Window → Package Manager), click on the gear icon and select Advanced Project Settings.



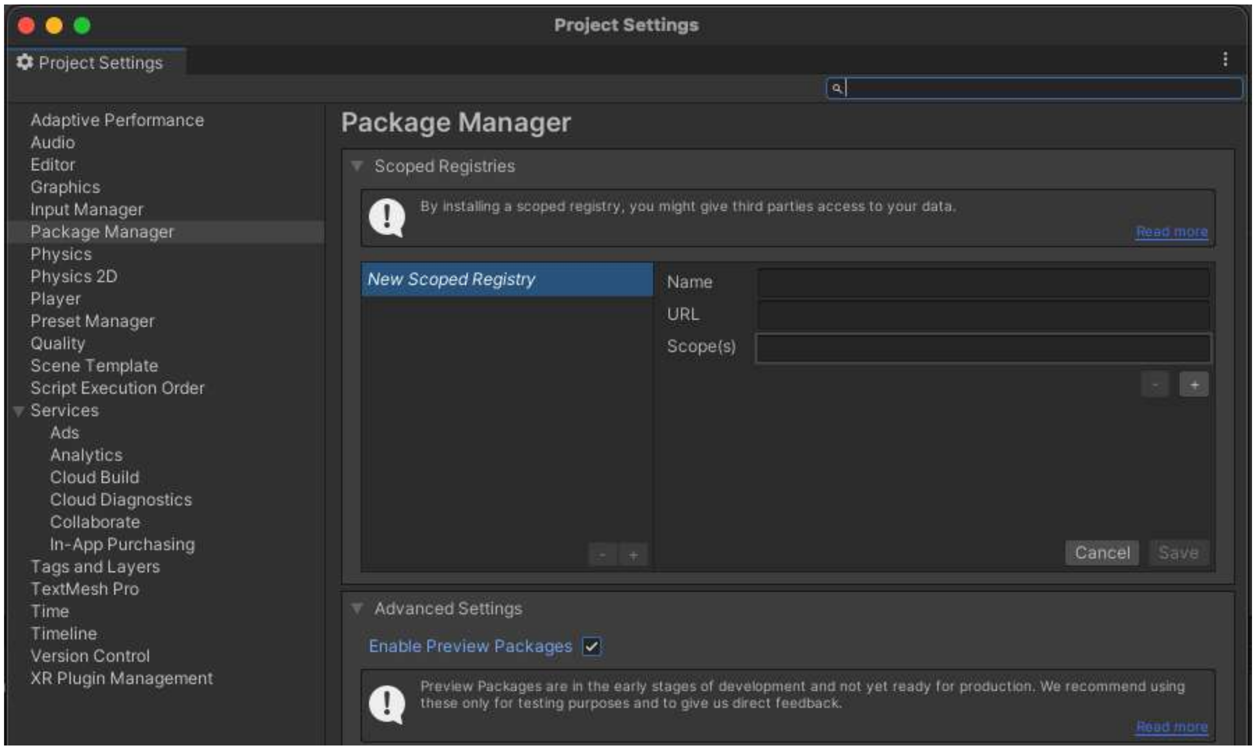
2D Animation	5.0.5
2D Pixel Perfect	4.0.1 ✓
2D PSD Importer	4.0.2 ✓
2D Sprite	1.0.0 ✓
2D SpriteShape	5.1.2
2D Tilemap Editor	1.0.0 ✓
Currently Installed	1.0.0 Verified

2D Animation
Unity Technologies
Version 5.0.5 - May 17, 2021
View documentation • View changelog • View licenses

2D Animation provides all the necessary tooling and runtime components for skeletal animation using Sprites.

Registry Unity

Afterwards just enable preview packages



Package Manager

Scoped Registries

By installing a scoped registry, you might give third parties access to your data. Read more

New Scoped Registry

Name

URL

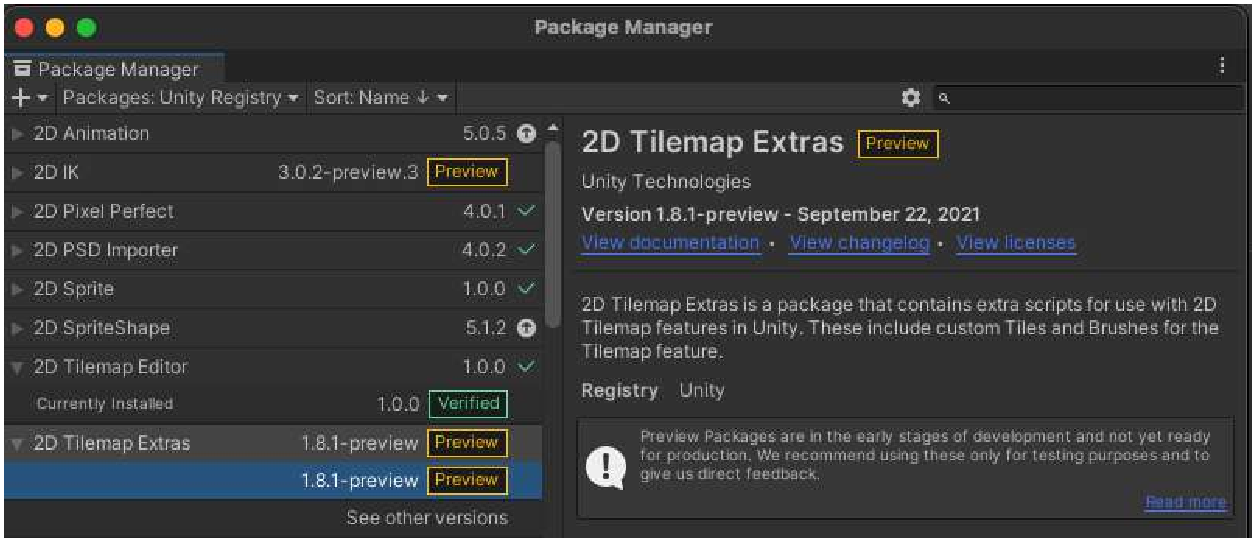
Scope(s)

Advanced Settings

Enable Preview Packages ✓

Preview Packages are in the early stages of development and not yet ready for production. We recommend using these only for testing purposes and to give us direct feedback. Read more

And then the 2D Tilemap Extras should appear in your Package Manager



2D Animation	5.0.5	
2D IK	3.0.2-preview.3	Preview
2D Pixel Perfect	4.0.1	✓
2D PSD Importer	4.0.2	✓
2D Sprite	1.0.0	✓
2D SpriteShape	5.1.2	
2D Tilemap Editor	1.0.0	✓
Currently Installed	1.0.0	Verified
2D Tilemap Extras	1.8.1-preview	Preview
	1.8.1-preview	Preview

2D Tilemap Extras Preview

Unity Technologies

Version 1.8.1-preview - September 22, 2021

View documentation • View changelog • View licenses

2D Tilemap Extras is a package that contains extra scripts for use with 2D Tilemap features in Unity. These include custom Tiles and Brushes for the Tilemap feature.

Registry Unity

Preview Packages are in the early stages of development and not yet ready for production. We recommend using these only for testing purposes and to give us direct feedback. Read more