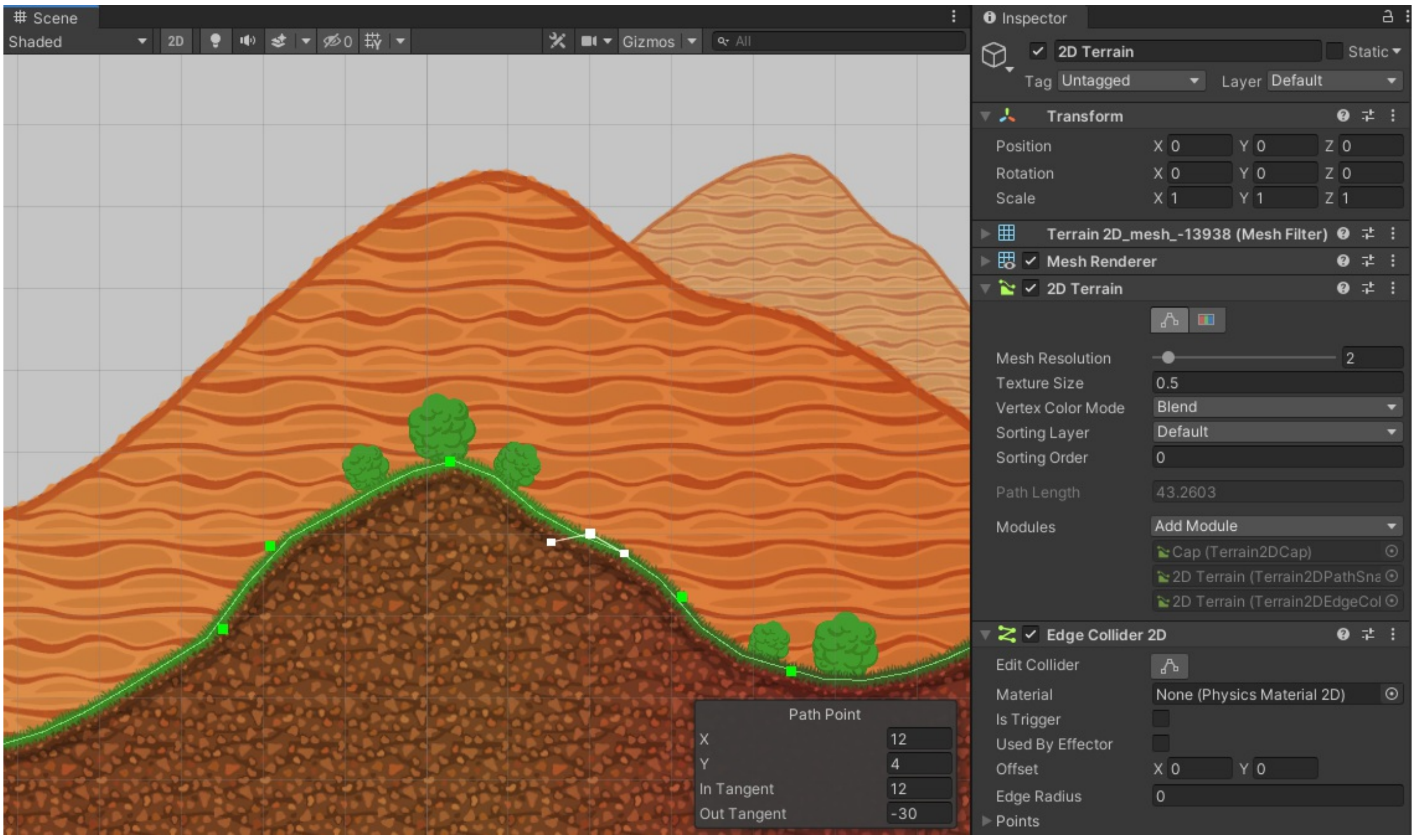
Table of Contents

This asset will generate dynamic 2D Terrain mesh for your game. Edit heightmap right in Editor SceneView trough curves, or generate it in Runtime trough API. Great solution for mobile 2D platformer or racing games.

- Compatible with all platforms
- Strongly optimized using Advanced Mesh API and Job System with Burst integration ready
- Optional Modules based on your needs
- Full Editor Undo / Redo support
- Well organized and commented source code without third-party dependencies



2D Terrain 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

Terrain 2D_mesh_-13938 (Mesh Filter)

Mesh Renderer

2D Terrain

Mesh Resolution	2
Texture Size	0.5
Vertex Color Mode	Blend
Sorting Layer	Default
Sorting Order	0
Path Length	43.2603
Modules	Add Module
	Cap (Terrain2DCap)
	2D Terrain (Terrain2DPathSna
	2D Terrain (Terrain2DEdgeCol

Edge Collider 2D

Edit Collider	
Material	None (Physics Material 2D)
Is Trigger	
Used By Effector	
Offset	X 0 Y 0
Edge Radius	0
Points	

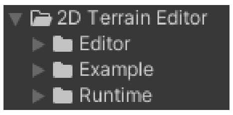
Path Point

X	12
Y	4
In Tangent	12
Out Tangent	-30

1. Installation

1.1 Download and import package.

The root **2D Terrain Editor** folder structure should look like this:

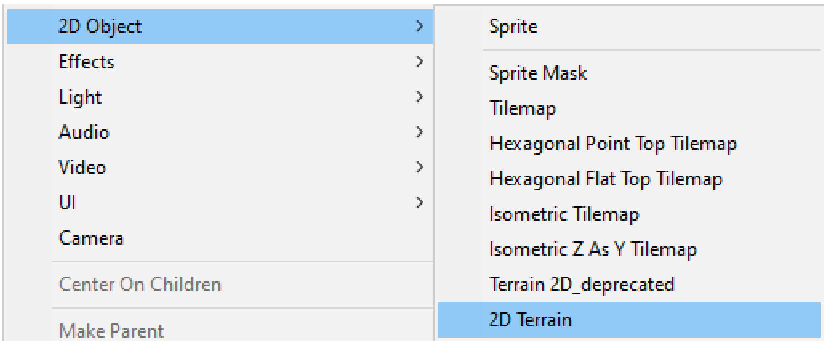


- **Editor** & **Runtime** folders contains core scripts and resources.
- **Example** folder contains demo scenes and sample materials that you can use in your project.

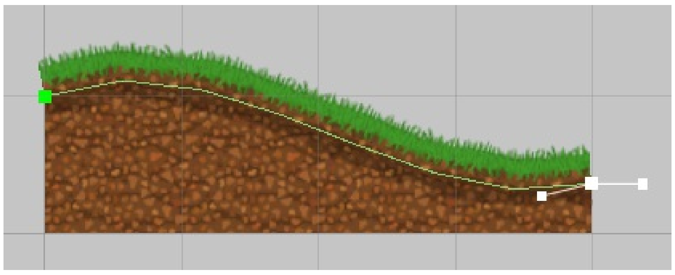
You are able to move **2D Terrain Editor** folder to any subfolder in your project.

2. Creating and editing 2D Terrain

2.1 Click **GameObject/2D Object/2D Terrain**. Default **2D Terrain** will be created on the scene.



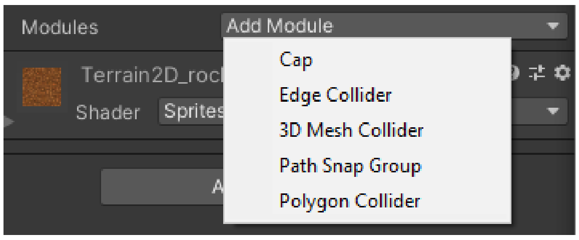
2.2 Select **2D Terrain** GameObject, navigate to **2D Terrain** component in the inspector and activate **Edit Path** tool. Using this tool you can create [Double Click], delete [Del], move, or change the parameters of Path Point Keys.



2.3 Use **Edit Vertex Colors** tool to assign color gradient to generated mesh vertices.

2.4 Click **Add Module** button to add features like 2D Collider or Cap.

There are several modules that can be attached to 2D Terrain as a components.



Every module has it's **Parent** Terrain 2D assigned.

Cap

Adds new GameObject with it's own **Mesh Renderer** component. Generates configurable mesh along the **Path Points** of **Parent**.

Edge Collider

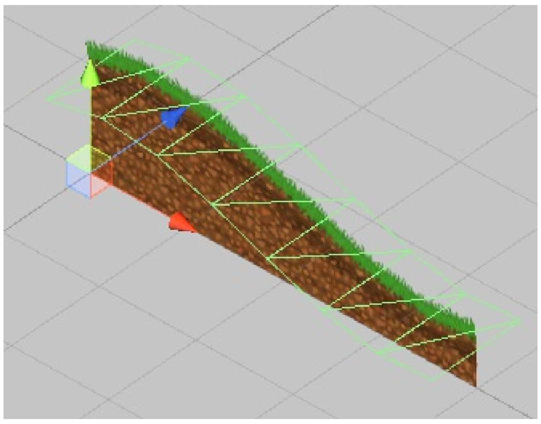
Generates Edge Collider 2D along the **Path Points** of **Parent**.

Polygon Collider

Wraps the whole **Parent** with Polygon Collider 2D.

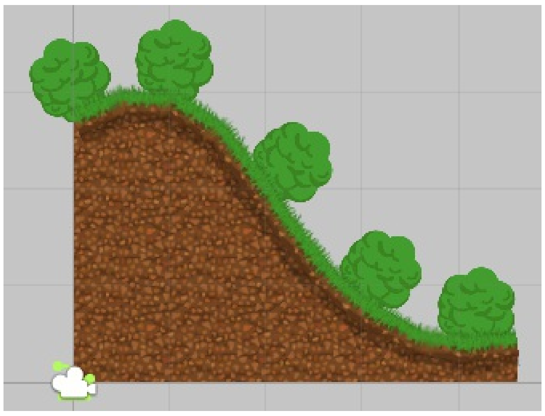
3D Mesh Collider

Generates 3D Mesh Collider along the **Path Points** of **Parent**.



Path Snap Group

Aligns all child transforms to **Parent** path by Y-axis.



Generating 2D Terrain

```
[SerializeField] private Material _myTerrainMaterial;

private Terrain2D _myTerrain;

//Create simple flat terrain from two points
void Awake()
{
    _myTerrain = new GameObject("New 2D Terrain").AddComponent<Terrain2D>();
    _myTerrain.MeshRenderer.material = _myTerrainMaterial;
    _myTerrain.MeshResolution = 5;

    AnimationCurve myPath = new AnimationCurve();
    myPath.AddKey(0, 10);
    myPath.AddKey(10, 10);
    _myTerrain.Path = myPath;

    //Build() should be called manually to apply changes
    _myTerrain.Build();
}
```

```
//Paint 2D Terrain with Red > Green > Blue gradient
void SetVertexColors()
{
    List<Terrain2D.VertexColorKeyPoint> newVertexColors = new List<Terrain2D.VertexColorKeyPoint>();

    newVertexColors.Add(new Terrain2D.VertexColorKeyPoint(0, Color.red));
    newVertexColors.Add(new Terrain2D.VertexColorKeyPoint(5, Color.green));
    newVertexColors.Add(new Terrain2D.VertexColorKeyPoint(10, Color.blue));

    _myTerrain.VertexColorKeys = newVertexColors;
    _myTerrain.Build();
}
```

Adding modules

```
[SerializeField] private Material _myCapMaterial;
[SerializeField] private float _myCapSize;

//Adding Cap module
void AddCapModule()
{
    Terrain2DCap newCap = _myTerrain.AddModule<Terrain2DCap>();

    newCap.MeshRenderer.material = _myCapMaterial;
    newCap.Size = _myCapSize;

    //Build() should be called manually to apply module changes
    newCap.Build();
}
```

```
[SerializeField] private GameObject[] _terrainProps;

//Snapping _terrainProps to terrain path
void SnapPropsToTerrain()
{
    Terrain2DPathSnapGroup pathSnapGroup = _myTerrain.AddModule<Terrain2DPathSnapGroup>();

    foreach (var p in _terrainProps)
        p.transform.parent = pathSnapGroup.transform;

    pathSnapGroup.Build();
}
```

Q: Is this asset compatible with URP / HDRP?

A: Yes.

Q: Is this asset optimized for mobile platforms?

A: Yes.

Q: Can I use custom materials / shaders?

A: Yes.

Q: How can I save generated meshes?

A: You can use Unity's FBX Exporter package.

Q: How can I create custom module?

A: Create C# class derived from Terrain2DModule.

```
[Terrain2DModuleInfo("My Custom Module")]
public class MyCustomModule : Terrain2DModule
{
    protected override void OnBuildPerformed(Terrain2D.BuildData buildData)
    {
        //Do your stuff here using buildData
    }
}
```

After that, you are able to add it in Terrain 2D component inspector (**Add Module** button) or via AddModule from script.