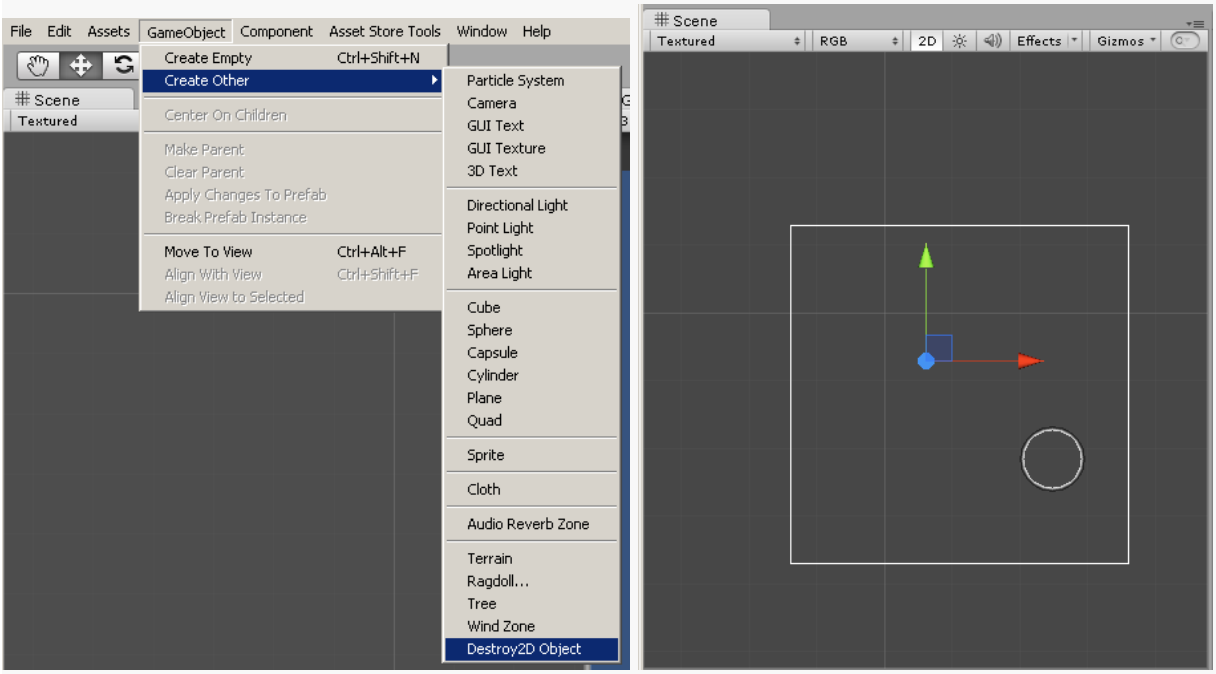
Welcome to Destroy2D!

In three easy steps I will introduce you to the key concepts behind Destroy2D and how to incorporate its functionalities into your scripts. All textures used in this tutorial are available . Go ahead and start a new scene in Unity and import the Destroy2D package.

1. Create your terrain

Go to GameObject menu -> Create Other -> “Destroy2D Object”. You should see a white square in your scene. That’s the area containing your terrain. If you move your mouse over this area, the brush will appear and you can start drawing. Yes, it’s that easy.



Great! You now have a terrain. If you check the inspector, you should see some options. Let me explain what they do:

Material: This is the material used by the terrain.

Size: The size (in Unity units) of the terrain.

Resolution: How much detail this terrain has. The larger the number, the more details and triangles this terrain can have. If the resolution is too low, the terrain will render faster but will be too “blocky”. If the resolution is too high, you can create more organic terrains at the cost of triangle count and draw calls.

Nodes per mesh: Internally, Destroy2D separates the terrain in smaller meshes to optimize rendering. This number is usually lower than half of the resolution.

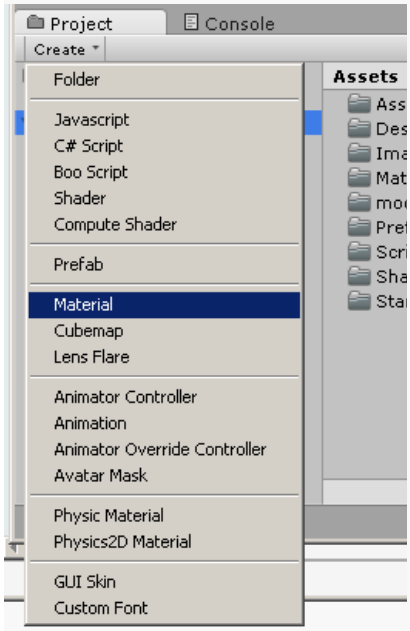
If you do any changes to one of those options (except for changing the material), you have to click on the “Commit” button to apply the changes.

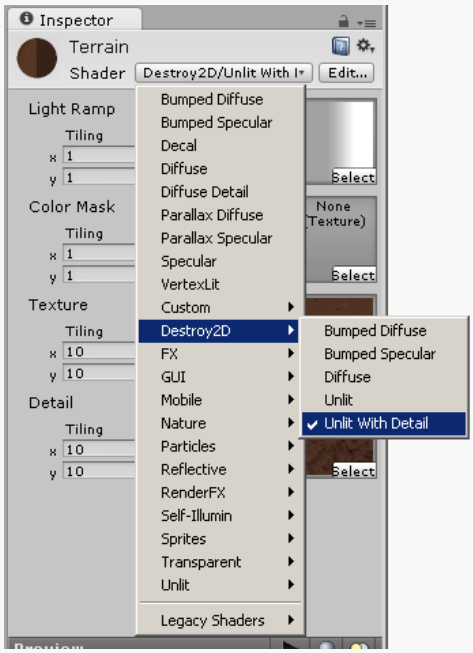
2. Paint your terrain

You will probably notice your terrain is a bright magenta thing. This is because you don’t have a material assigned to it. Don’t worry, this step is also very easy.

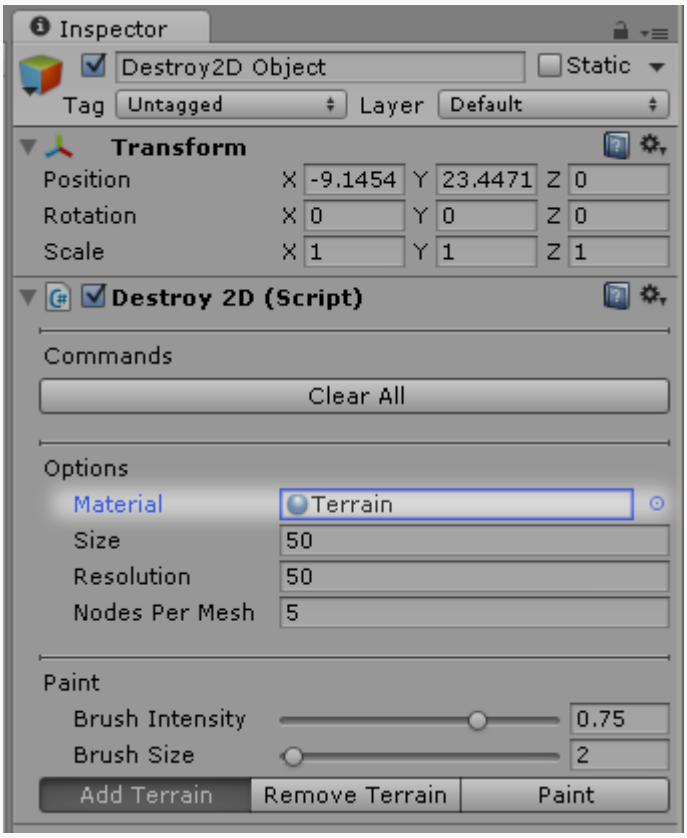
You can either duplicate one of the sample materials (in the Project tab, in Destroy2D/Materials) or create a new material from scratch and use one of the Destroy2D shaders.

For this tutorial, we will do the latter. Create a new material, name it "Terrain" and choose the “Destroy2D Unlit Detail” shader.





Once you create this material, select your “Destroy2D Object” and assign this material to the “material” field.



As you can see, the material accepts four textures.

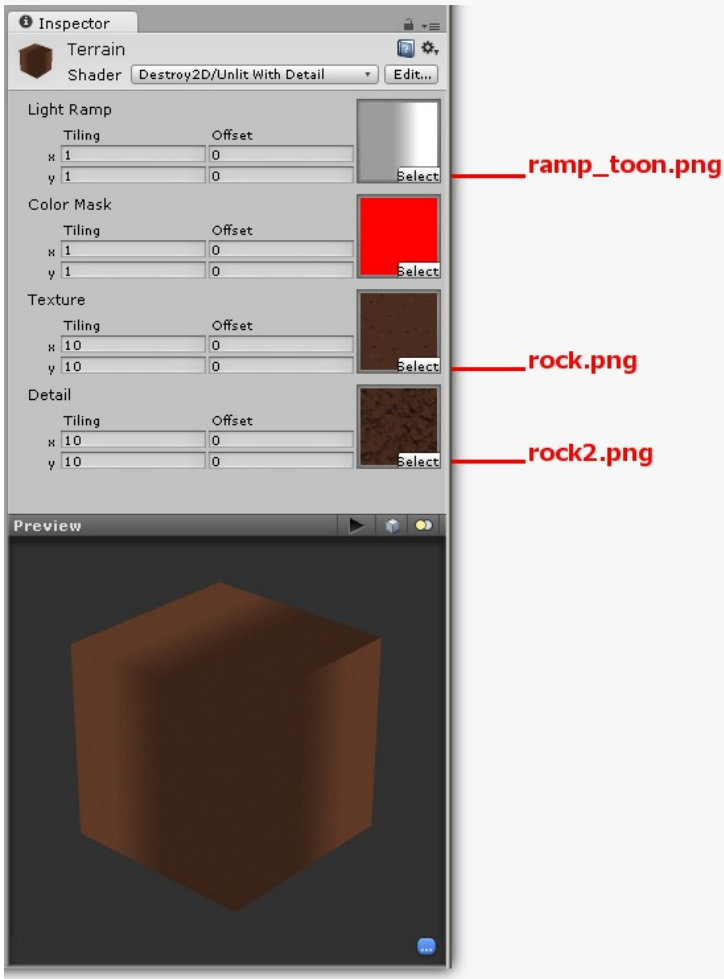
The first one is the Light Ramp. This is a gradient Destroy2D uses to define the contour of the terrain. The colors to the left will be at the center of the terrain, the colors to the right will be at the edges. Select the ramp texture named “ramp_toon.png”. You will notice it consists of a simple texture with a lighter area to the right. This means the edges of the terrain will have a lighter tonality, giving some depth to it. If you don’t need this effect, you can leave it blank. For this tutorial, let’s use the sample ramp I included in this tutorial.

The second texture is the Color Mask. You can ignore this texture, as this is assigned by the Destroy2D Object at runtime. This is used by the shaders to determine which areas use the two different textures.

The third texture is the main texture. When you assign a texture to this slot, the entire terrain will use it. Increase the tilings of this texture so your terrain don’t look like a giant blurred mess.

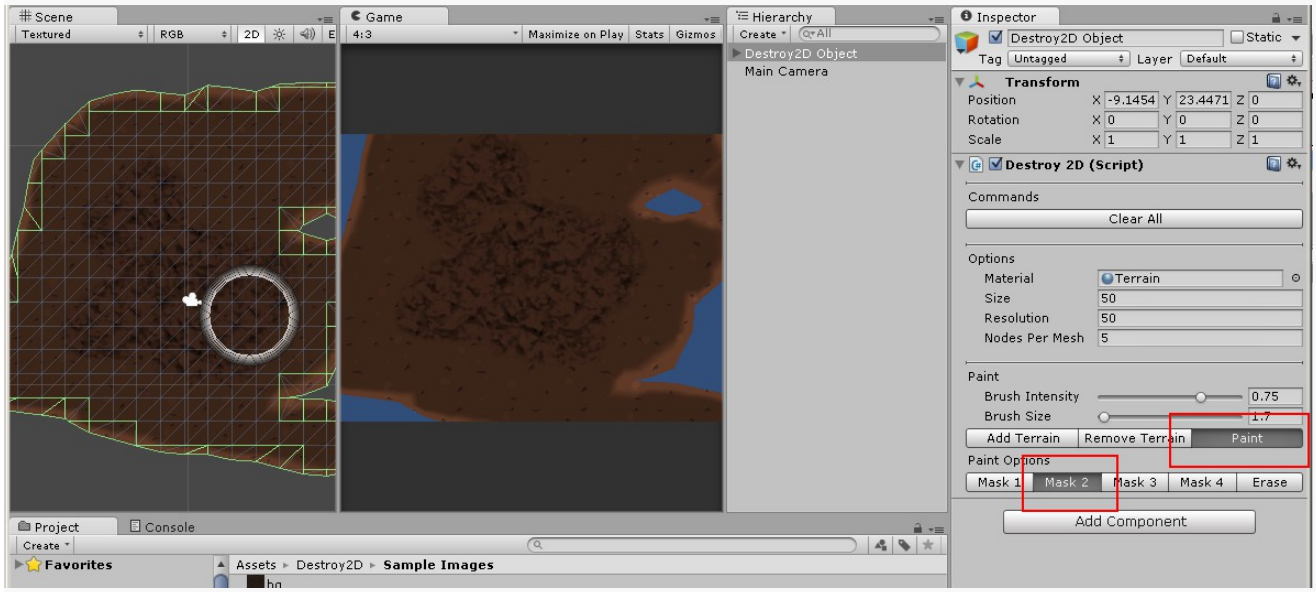
The fourth texture is the detail texture. This is a second texture you can paint on your terrain. Assign a different texture to it and you should see nothing happen. This is expected, as the terrain mask is not yet painted.

For this tutorial, let`s use the following textures (all of them included in the Destroy2D package):



For this next step, make sure you've added some terrain shape first. To paint the texture details, select your “Destroy2D Object” and check the inspector. In the “Paint” section, click on the “Paint” button to see the paint options. Select “Mask 2” and draw something on your terrain.

You should see the detail texture appear in the areas you painted.



What you just did was to paint the second mask of the Color Mask. The “Destroy2D Unlit Detail” shader only deals with two masks (one for the main texture and the other for the detail texture). The other masks are not used by this shader but can be painted nonetheless. This is implemented in case you wish to create a new shader that uses all four masks.

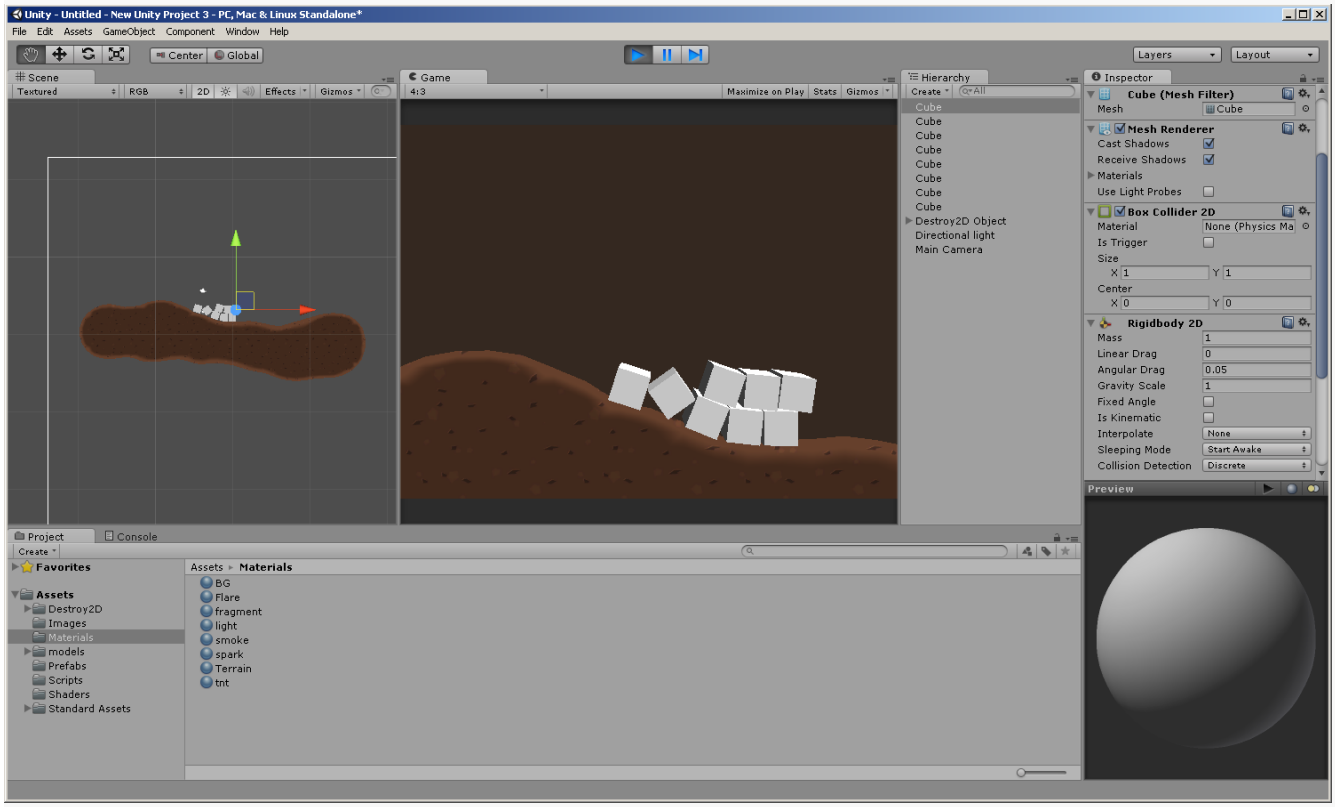
Anyway, now we have a nice textured terrain! What can we do with it, you ask? Let’s just drop some things on it and see what happens.

3. Use your terrain

In the Hierarchy tab, click on Create and select Cube. A new cube should have been added to your scene. It’s probably black because we don’t have any lights in the scene. Add a directional light so we can see what’s going on.

Scale and position this cube somewhere outside your terrain. Make sure the position in the Z axis is 0. Replace the default collider by a Box Collider 2D and add a Rigidbody 2D component. Duplicate this cube a few times so you can see them interactig with the terrain and with each other.

Now run.



It falls! As you can see, Destroy2D is fully integrated with Unity's physics engine. You can add other types of standard 2D colliders primitives (circles, polygons, edges) and it works just as well.

What else can we do with the terrain? Well, we can remove or add parts of it at runtime. Create a new c# script named "Destroy.cs". Add this code to the script:

```
using UnityEngine;
using System.Collections;

public class Destroy : MonoBehaviour {

	void OnCollisionEnter2D(Collision2D coll) {
		if (coll.gameObject.name == "Destroy2D Mesh") {
			Destroy2D terrain =
GameObject.Find("Destroy2D Object").GetComponent<Destroy2D>();
			terrain.destroyAt(transform.position, 3,
0.5f);
			Destroy(this);
		}
	}
}
```

This script is also included in the package. You can find it in Destroy2D/Scripts.

Assign this code to the cubes you created and run. You should see the cubes destroying the terrain. This is because we are removing the terrain when the cubes collide with the terrain.

After the terrain around the cube is destroyed, we remove the script from the cube so it destroys the terrain only once.

The "destroyAt(..)" function accepts three values: the position where the terrain will be removed, the radius of the "hole" and the intensity.

Go ahead and play with these values. Make sure to check what happens if you use a negative intensity. ;)

So, that's it for this tutorial. I hope you enjoy creating 2D terrains with this tool. Don't forget to watch our video tutorial as well! If you have any questions or need help with any of the features, don't hesitate to contact us at angelo@axis3d.com.br.