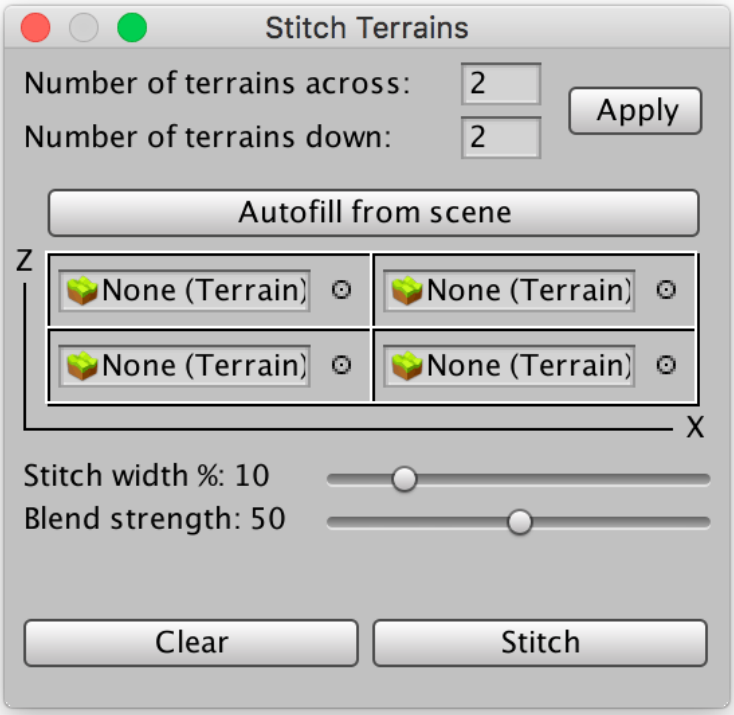
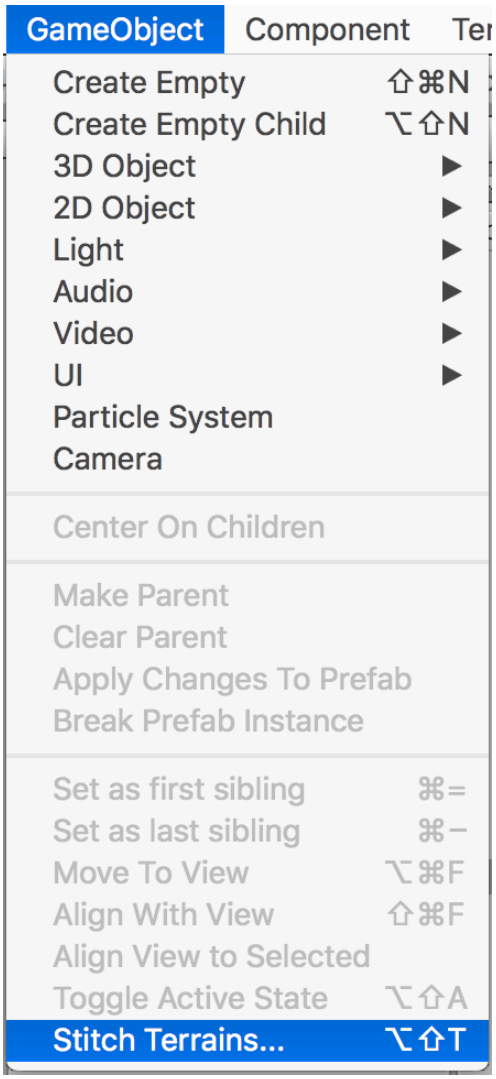
Welcome to Stitchscape! You can stitch terrains in the editor, or at runtime. See “Runtime Stitching” on page 7 for details about how to use Stitchscape in your own scripts. To use it in the editor, select the menu item in the GameObject menu called "Stitch Terrains...", or use the Alt-Shift-T keyboard shortcut.

This will bring up a “Stitch Terrains” window where you can set up the terrains that you want stitched:





Initially there aren’t any terrains assigned. You can assign terrains either manually or by using the “Autofill from scene” button. When setting up terrains manually, first change the number of terrains across and down if necessary, then click the “Apply” button. Note that “across” means arranged on the world X axis, and “down” means arranged on the world Z axis, as viewed from above when choosing “top” as the scene camera view. You can resize the Stitchscape window if needed if you have many terrains. Next, assign your terrains in the grid below. You can do this by either dragging terrain objects from the hierarchy view to the appropriate grid boxes in the Stitchscape window, or you can use the selector icons in the grid boxes to select terrains from a list of Terrain objects in the scene.

If you use the “Autofill from scene” button, Stitchscape will get a list of all terrains in the scene and attempt to order them into a grid. The terrains must be already set up as a grid in the scene, and positioned next to each other. Ideally they should match up exactly, so they look seamless after stitching, but autofill will still work if they are reasonably close (plus or minus one unit). It's not necessary to set the across and down fields when using autofill since those will be calculated automatically.

Note that for Stitchscape to work, all terrains must have the same heightmap resolution. Also, all terrains should be located at the same height (i.e., have the same value for the Y axis). They should be positioned on the X and Z axes so that there is no overlap or gaps between terrains. Stitchscape can only alter the terrain heightmaps, and doesn't make any other adjustments to the terrain objects.

Also note that if you use a terrain grid size of 1x1, then the single terrain will be made seamlessly repeatable with itself, so it can be tiled indefinitely. In this case, after stitching, you will need to duplicate and position the terrains yourself.

Below the terrain grid is the “Stitch width %” slider. This is the percentage of the terrain width from the edges of each terrain that will be affected by the stitching. The larger the number, the wider the band that will be affected. The minimum stitch width is 1%, and the maximum is 50%. For best results, you should use larger values for terrains that don't match very well, and small values for terrains that nearly match to begin with.

After that is the “Blend strength” slider. This ranges from 0 to 100, and affects how strongly Stitchscape will blend the terrain to the midpoint. When stitching a given terrain, Stitchscape uses detail from the opposite terrain to create a more natural look, but this can result in a somewhat “mirrored” look if the blend strength is set to 0. If it’s set to 100, then the result is similar to smoothing. The default of 50 typically works well, but this can be adjusted to suit different terrains.

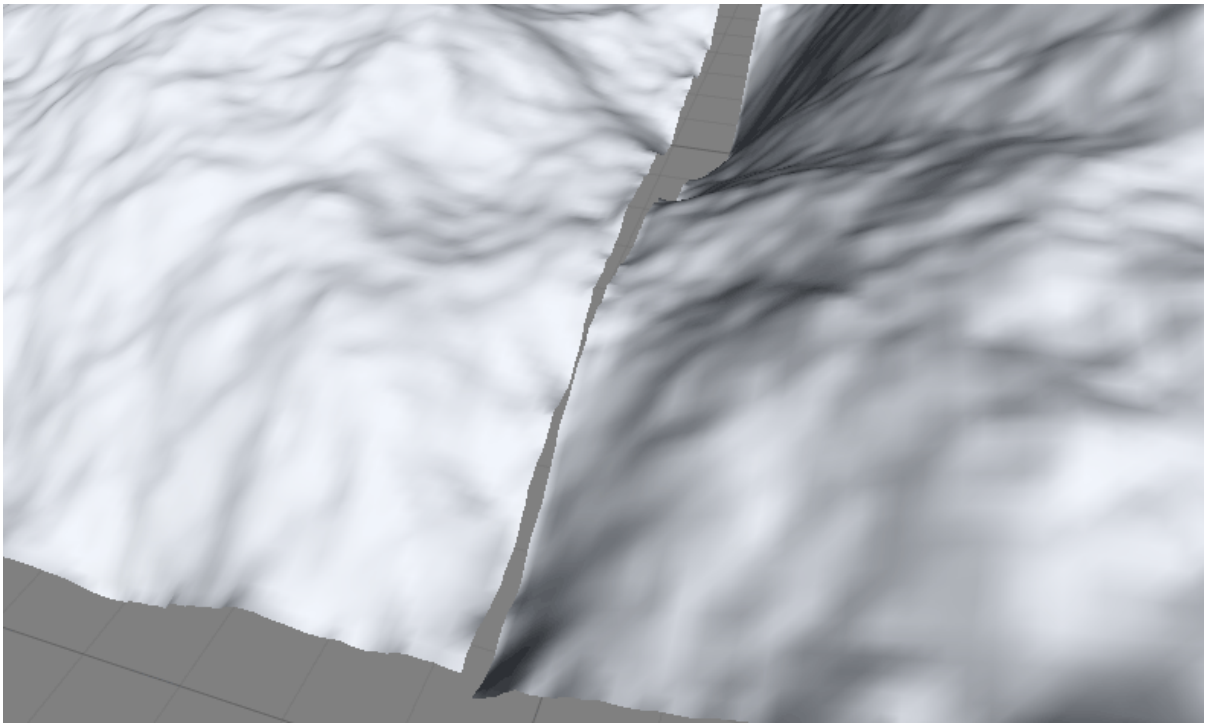
The “Clear” button will remove all terrains from the stitching grid (but doesn't affect the terrains themselves) in case you want to start over.

When you’re all set, click the “Stitch” button. This will alter the terrains so they match up seamlessly. You can undo the stitching using the Unity undo function, which is useful if you want to experiment with different settings.

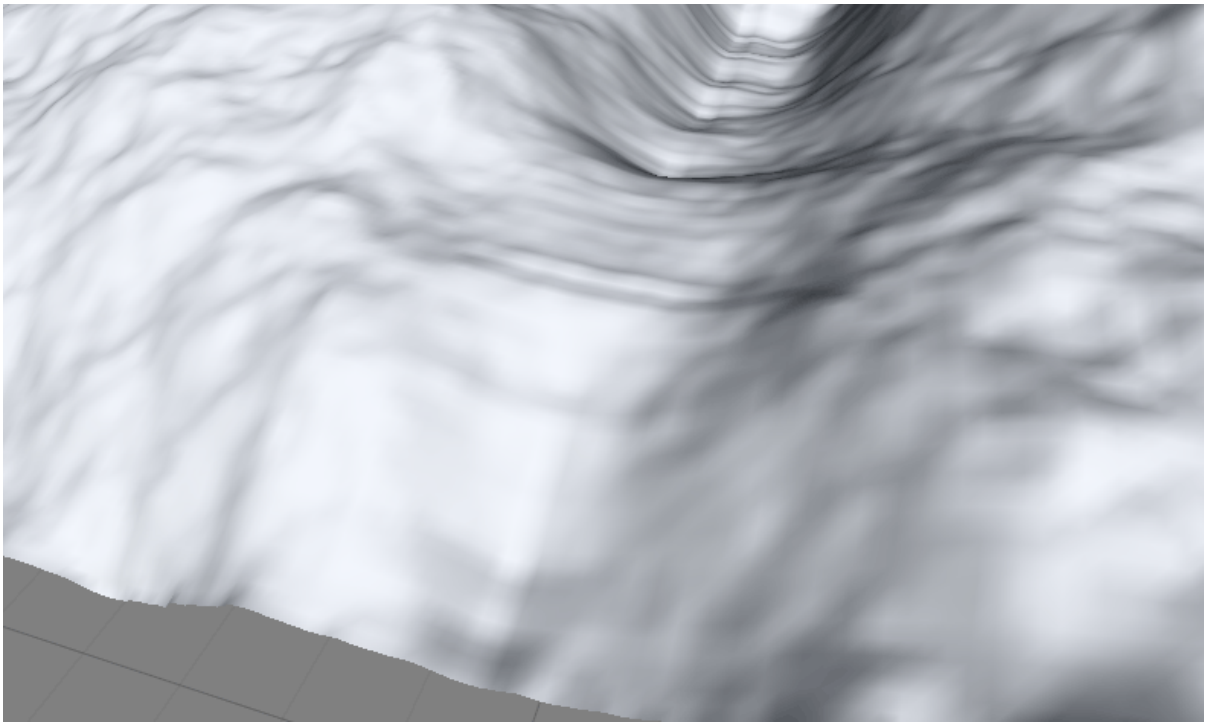
Note that proper use of the Terrain.SetNeighbors function in your scripts is still required in order for terrain LOD to match up correctly at a distance. Stitchscape can’t do this automatically.

On the next pages you can see the results of different Stitchscape settings.

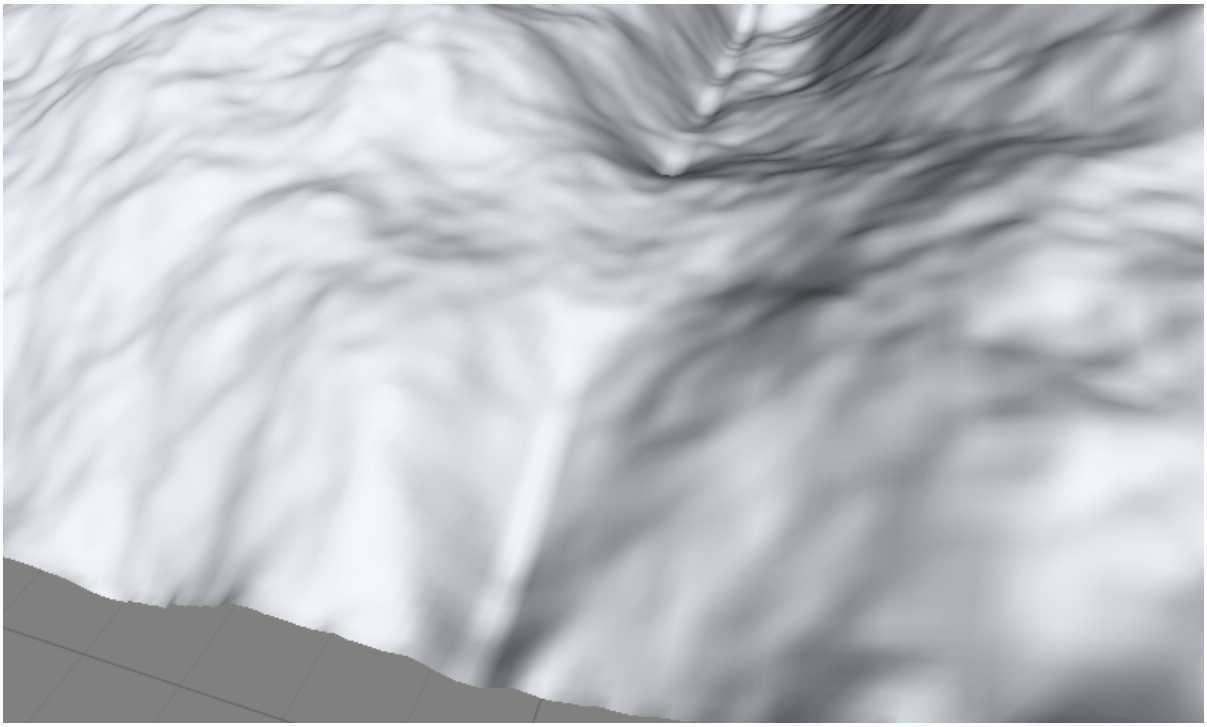
Here we have a pair of terrains that we'll stitch:



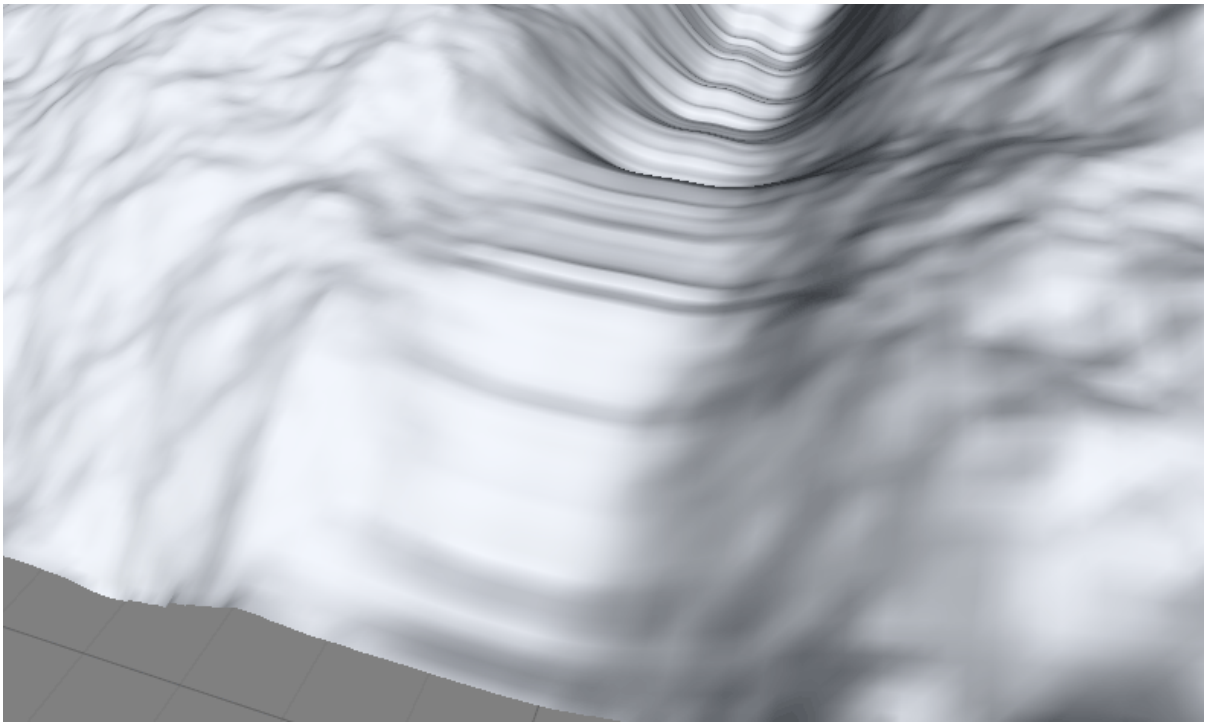
The default settings of 10% width and 50 strength results in this:



Setting the strength to 0 gives us this instead, where more detail is kept but some mirroring is visible:

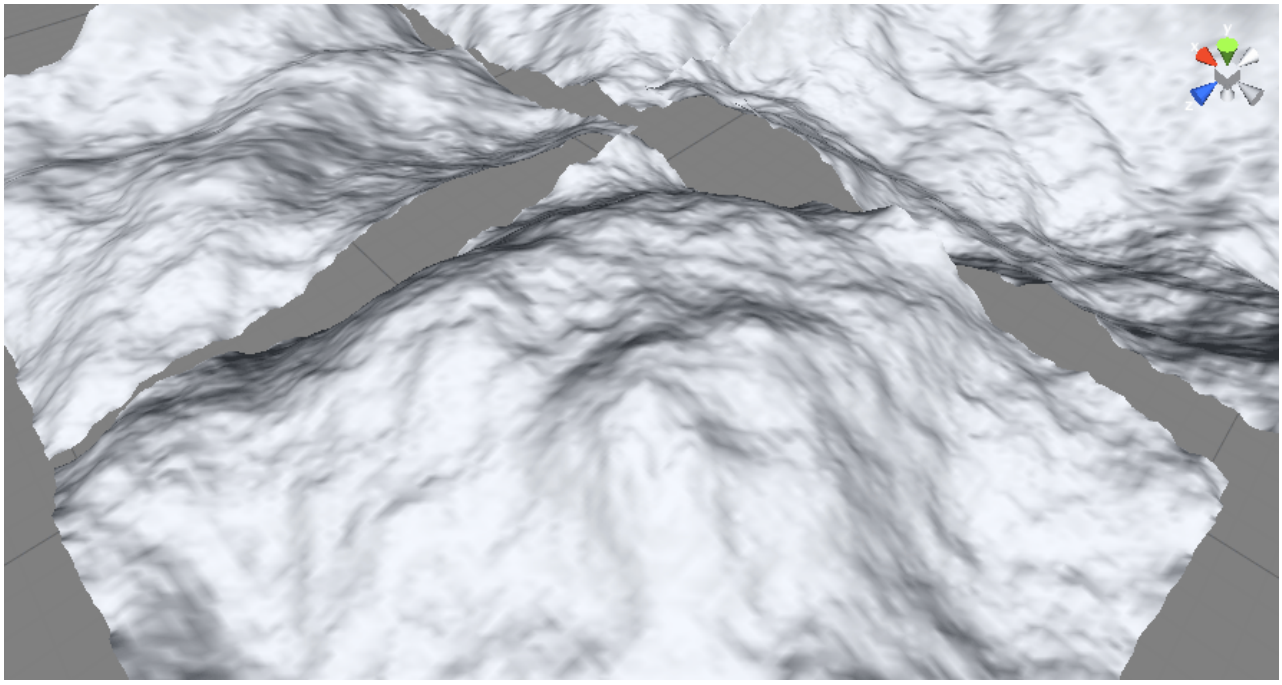


Setting the strength to 100 smooths out the terrain blending completely:

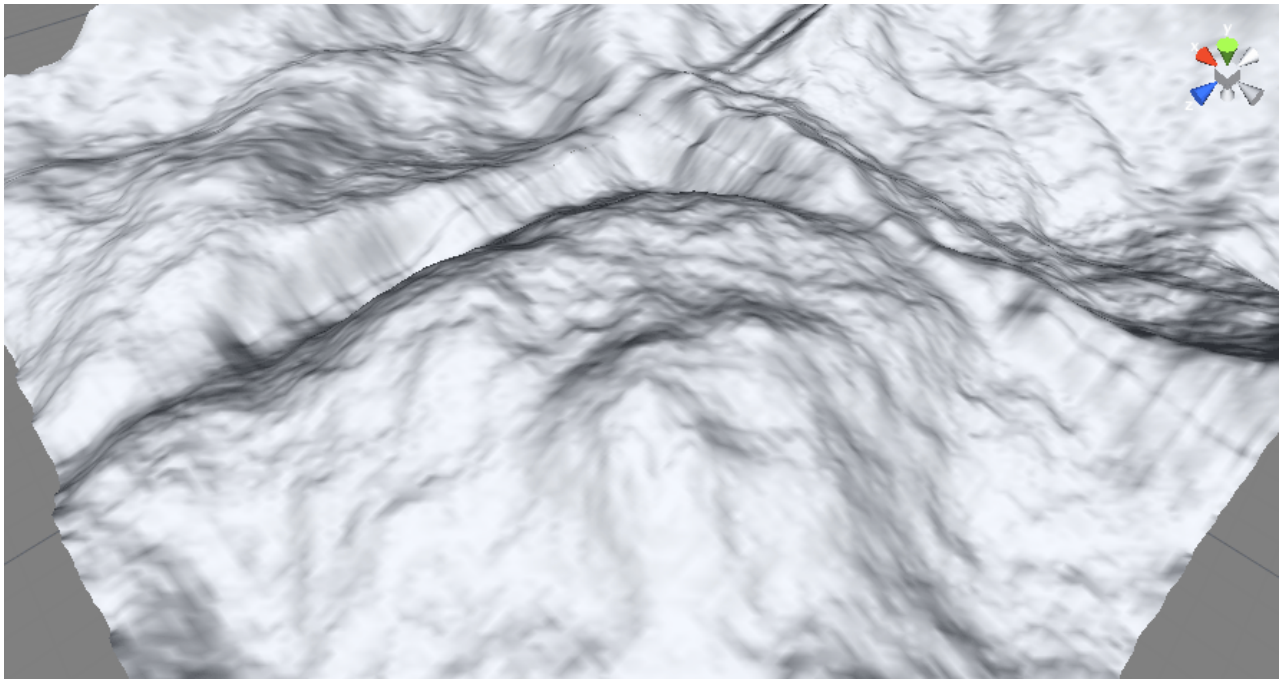


A high strength tends to look better with small stitch widths.

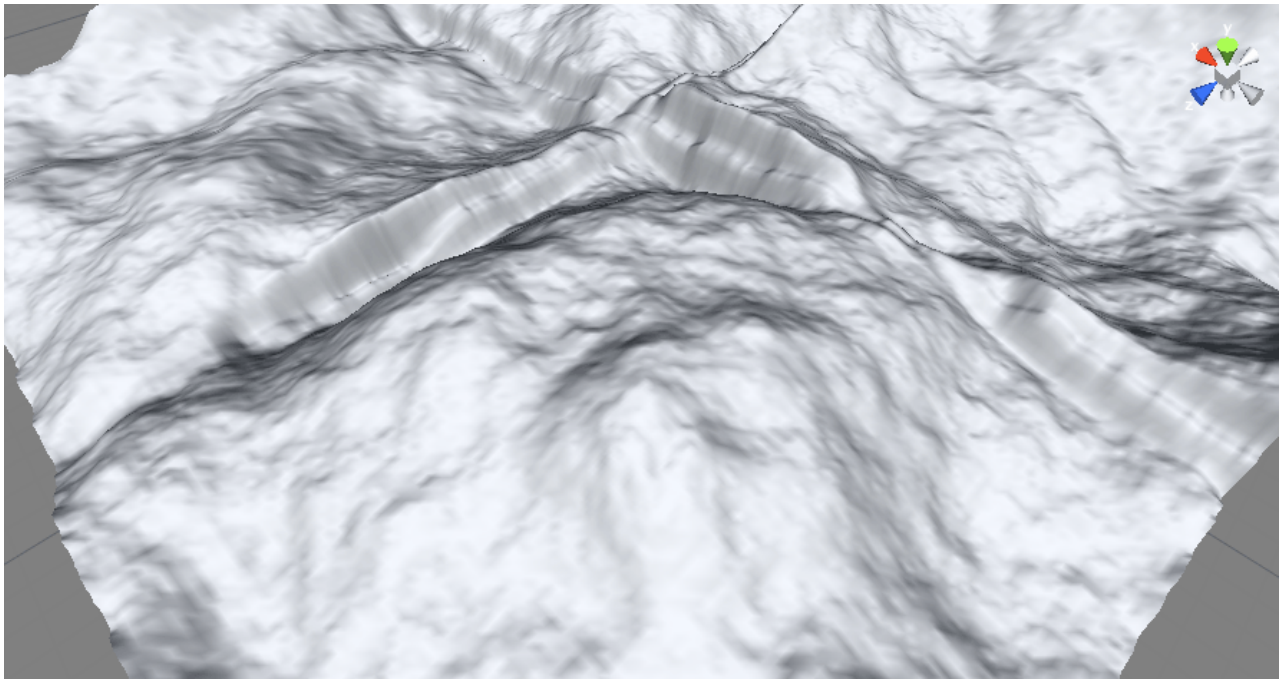
To see the effect of stitch width, let's zoom and and see four unstitched terrains:



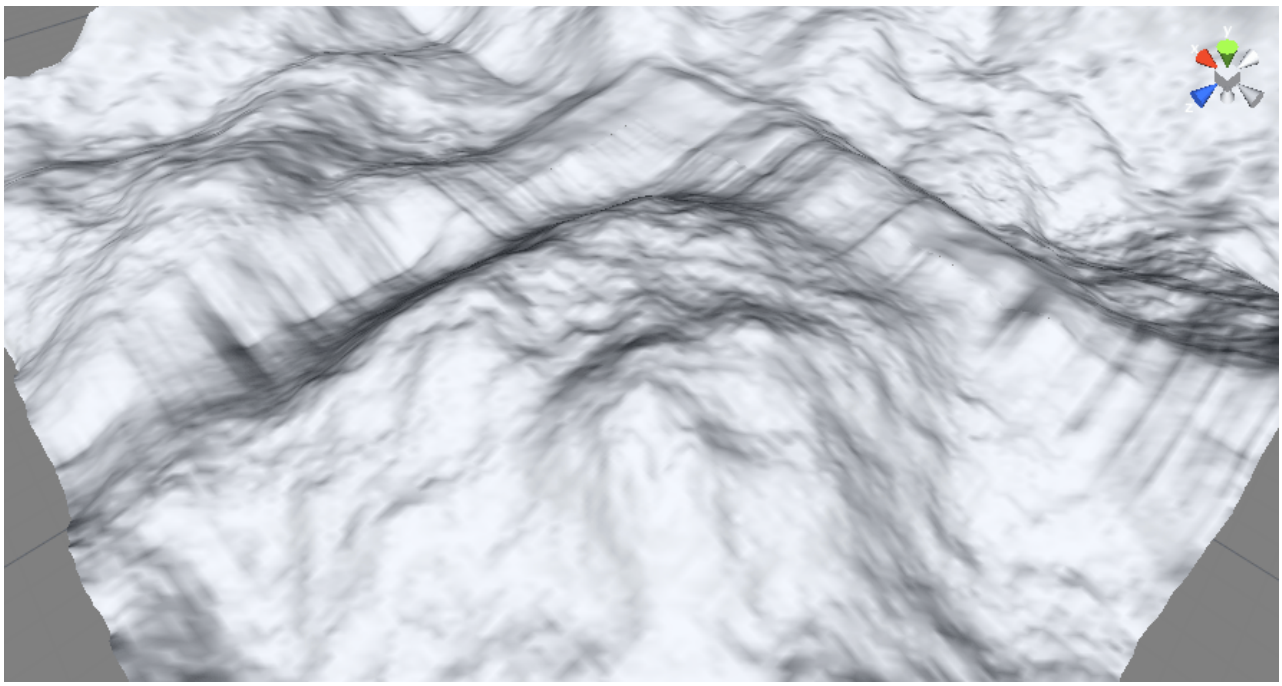
These terrains don't match up closely, but a 10% stitch width works reasonably well:



Reducing the width to 5% starts to show the limitations of stitching disparate terrains; it would work better with terrains that match up more closely:



A wider width of 25% works better for these terrains:



Since different settings work better or worse depending on the terrains involved, feel free to experiment and find the best match for your particular terrains.

Runtime stitching

You can stitch terrains in your scripts by using the TerrainStitch function, which is in the Stitch class. This function has several parameters:

terrain1: TerrainData for one of two terrains used for stitching

terrain2: TerrainData for the second terrain used for stitching

StitchDirection: an enum that indicates whether the terrains should be stitched across or down

stitchWidthPercent: how much of the terrain is affected; the same as “Stitch width %” in the editor

blendStrength: how strong the stitching is; the same as “Blend strength” in the editor

singleTerrain: a boolean indicating you want a single terrain that will be made repeatable with itself

Note that stitchWidthPercent and blendStrength are floats ranging from 0.0 (0%) to 1.0 (100%). StitchDirection is either StitchDirection.Across or StitchDirection.Down.

If you use true for singleTerrain, then terrain1 and terrain2 should be the same TerrainData.

The heightmap resolution for terrain1 and terrain2 need to be the same, or else TerrainStitch will print an error and no stitching will occur.

When stitching multiple terrains, you’ll need to call TerrainStitch multiple times. For example, let’s look at four terrains laid out 2x2. We’ll have Terrain1 next to Terrain2 on one row, and Terrain3 next to Terrain4 on the second row below that. So you’d call TerrainStitch with StitchDirection.Across with Terrain1 and Terrain2, then again with Terrain3 and Terrain4. Then you’d call it with StitchDirection.Down with Terrain1 and Terrain3, followed by Terrain2 and Terrain4.

You’ll need to import the Stitchscape namespace (“import Stitchscape” in Unityscript or “using Stitchscape” in C#). If you’re using Unityscript, you’ll first need to make a folder in the project root called “Plugins” (if one doesn’t exist already) and move the Stitchscape script there, or else you’ll get script errors. On the next page is a code example showing TerrainStitch in action.

Warning

If you use TerrainStitch at runtime on assets in the project, the assets are altered permanently and no undo is possible. For best results, use TerrainStitch only on fully instantiated objects (instantiating a terrain does not automatically instantiate the associated TerrainData, so you need to do both), or on objects created entirely with code. Always use version control or at least have backups of the project when developing.

Here's an example of two side-by-side terrains being stitched, using 10% stitch width and 50% blend strength. To use it, have a scene with two terrains in it next to each other, call the script StitchExample and attach it to an object in the scene, then drag the two terrains onto the appropriate slots on the script. When run, duplicates of the terrains are created, placed 3000 units away from the originals, and stitched.

C#:

```
using UnityEngine;
using Stitchscape;

public class StitchExample : MonoBehaviour {
     public Terrain terrain1;
     public Terrain terrain2;

     void Start () {
          var newTerrain1 = Instantiate (terrain1) as Terrain;
          var terrainData1 = Instantiate (newTerrain1.terrainData) as TerrainData;
          newTerrain1.terrainData = terrainData1;
          var pos = terrain1.transform.position;
          pos.z += 3000;
          newTerrain1.transform.position = pos;
          var newTerrain2 = Instantiate (terrain2) as Terrain;
          var terrainData2 = Instantiate (newTerrain2.terrainData) as TerrainData;
          newTerrain2.terrainData = terrainData2;
          pos.x = terrain2.transform.position.x;
          newTerrain2.transform.position = pos;
          Stitch.TerrainStitch (terrainData1, terrainData2, StitchDirection.Across,
0.1f, 0.5f, false);
     }
}
```

Unityscript:

```
#pragma strict
import Stitchscape;

var terrain1 : Terrain;
var terrain2 : Terrain;

function Start () {
     var newTerrain1 = Instantiate (terrain1);
     var terrainData1 = Instantiate (newTerrain1.terrainData);
     newTerrain1.terrainData = terrainData1;
     var pos = terrain1.transform.position;
     pos.z += 3000;
     newTerrain1.transform.position = pos;
     var newTerrain2 = Instantiate (terrain2);
     var terrainData2 = Instantiate (newTerrain2.terrainData);
     newTerrain2.terrainData = terrainData2;
     pos.x = terrain2.transform.position.x;
     newTerrain2.transform.position = pos;
     Stitch.TerrainStitch (terrainData1, terrainData2, StitchDirection.Across, 0.1, 0.5,
false);
}
```