Contents

1 Introduction

This is the user manual for **e2d v1.0.0** . It is a 2D terrain editor and generator for **Unity 3.4**. It will guide you through basic steps of terrain creation and editing, and explain the features of the tool.

The manual is intended for users of the tool. If you're looking for the source code reference you should read the Doxygen documentation. If you want to know how the terrain is implemented and how the generator works inside read the thesis.

1.1 Installation

The first thing to do is to download Unity itself. After it is installed, open your project or create a new one. Then, double-click the **e2d_demo.unitypackage** file containing the tool and several examples. If you only want the tool itself open **e2d.unitypackage** instead. Click **Import**. The package with all script files will be imported into your project in the **e2d** folder, and you can start using it.

1.2 Getting Started

If you haven't done so yet you should read the Learning the Interface section of the Unity manual. It will show you how to control the editor and work in the environment.

Now, take a look at the Project window. In the **e2d demo** folder you can find few sample scenes with 2D terrain generated by the tool. You can open **basic.unity**, start the action and see something similar to Figure 1. The other sample scenes are **lerpz.unity** and **car.unity**.

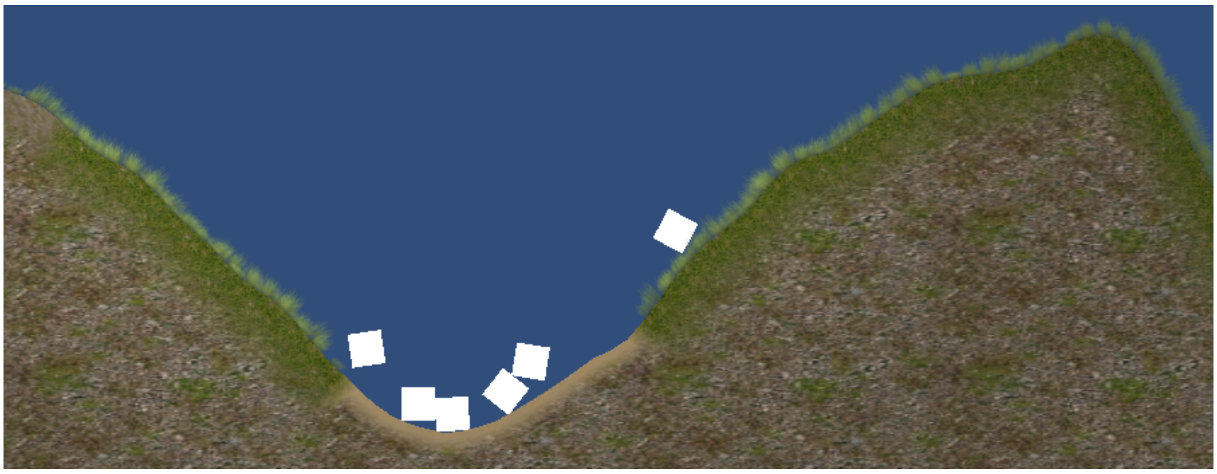


Figure 1: *Demo scene with sample terrain in action.*

The tools and the terrain itself are all in the **e2dTerrain** game object. If you want to play with the tools select the object, and then you can see all the tools in the inspector. But don't spend too much time there! We'll head on to explain you the tools one after each other.

1.3 Creating New Terrain

The demo scene is nice, but it's time to create your own terrain. First create a new empty scene. Then there are two ways of bringing the terrain tools into the scene:

1. Dragging the **e2dTerrain** prefab in the **e2d** folder into your scene.
2. Creating new game object and dragging the **e2dTerrainGenerator** and **e2dTerrain** scripts onto it.

The first method brings into the scene predefined terrain, but otherwise both of them have the same effect. To create your own terrain select the terrain object and in the inspector click on **Nodes**. This will select the tool for editing nodes. If you have used the first method delete the terrain data by clicking on **Reset Terrain**. Now the terrain is clear and we can start adding new nodes.

2 Terrain Editor

The terrain editor consists of tools located in the inspector of the **E2d Terrain** component. Select the game object with the terrain to see the inspector.

2.1 Editing Nodes

The surface of the terrain is composed of terrain nodes. They are basically 2D points linked together to form a polygonal chain. You can edit them by clicking on **Nodes** in the inspector. Now you can add new nodes by **left-clicking** anywhere in the scene (except other nodes), but be careful not to place any node, so that the surface curve intercrosses itself! If you do an error message will appear (see Figure 3). Then either undo your action or move the nodes, so that they don't intercross themselves anymore.

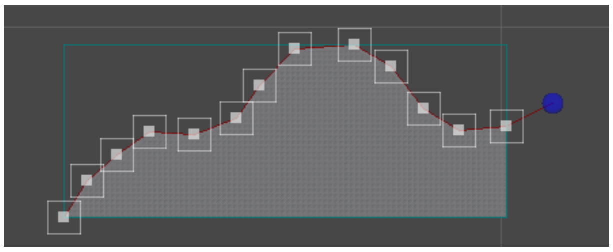


Figure 2: *Creating terrain nodes with the mouse.*

Normally, new nodes are added between two closest nodes or appended at the end of the terrain surface. But if you want to force the tool to append the node to the end, then hold **ctrl** while placing the node. You can **drag** each of the nodes to change their position. And if you want to remove any node hold **shift** and click on the node.

The **Reset Terrain** button clears all terrain data. Of course, you can undo this action, so don't worry! Clicking **Rebuild Data** will force the component to rebuild runtime data in case something went wrong. The **Close Curve** parameter is used to create terrain without any ground — like a floating island of land! See Figure 4 for example of this. If **Plastic Edges** is checked the terrain surface normals are modified, so that

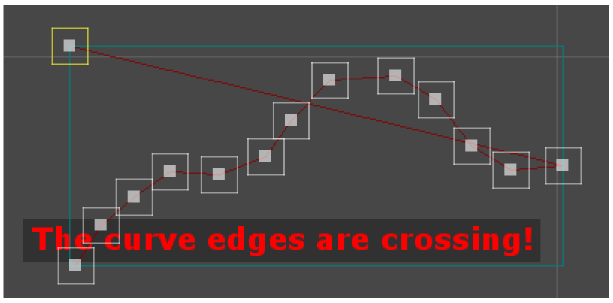

The curve edges are crossing!

Figure 3: *Terrain nodes are intercrossing themselves. The terrain cannot be rendered.*

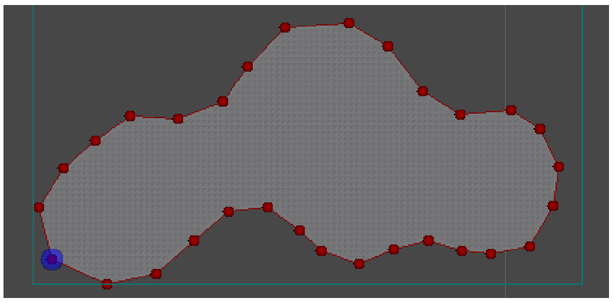


Figure 4: *Closed terrain forming an island of land floating in the air.*

it looks more plastic when there's a light hovering above the terrain. An example using lights with 2D terrain is in the **lerpz.unity** scene.

2.2 Height Brush

Now click on **Brush** to select the height brush tool. The height brush is used to move many nodes at once. They are moved in the direction you define — not just upwards! The affected terrain nodes are marked in the scene by blue glow, and the direction of movement is displayed by a red arrow (see Figure 5). As you increase the size of the brush more and more nodes will be marked blue and will be affected by the brush. To use the brush **left-click** in the scene or sweep the mouse around a bit.

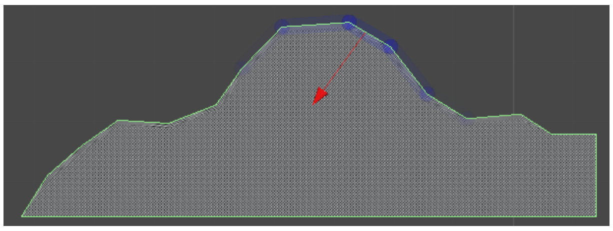


Figure 5: *Height brush before being applied.*

There are also some handy shortcuts you can use to change the properties of the brush. While holding **shift** the mouse movement changes the brush size. While holding **ctrl** the brush direction changes. And holding both **shift and ctrl** changes the strength (opacity) of the brush. The brush parameters have the following meaning:

- **Brush Size** — size of the brush in world units.
- **Brush Opacity** — influences the strength of the brush; the higher the more movement will be done by the brush.
- **Brush Angle** — specifies the direction of the movement.

2.3 Terrain Filling

The area of the terrain is defined by the surface nodes (as we have created them earlier) and the boundary rectangle. By adjusting the rectangle you can define the shape of the filling of the terrain. You can do this if you click on **Filling** and then drag the handles in the scene. You can also change the texture of the filling in the inspector. In the **e2d demo/Terrain Textures** folder there are some default textures you can use. They are supplied with Unity and are free to use. The terrain should look similar to Figure 6. The blue rectangle shows the boundary rectangle of the terrain. The parameters have the following meaning:

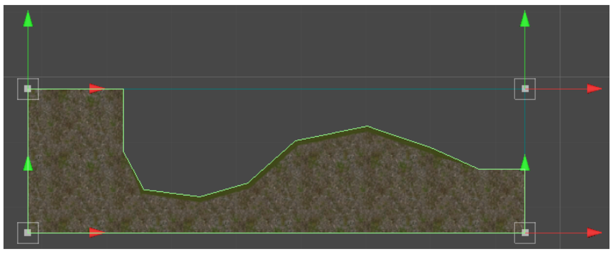


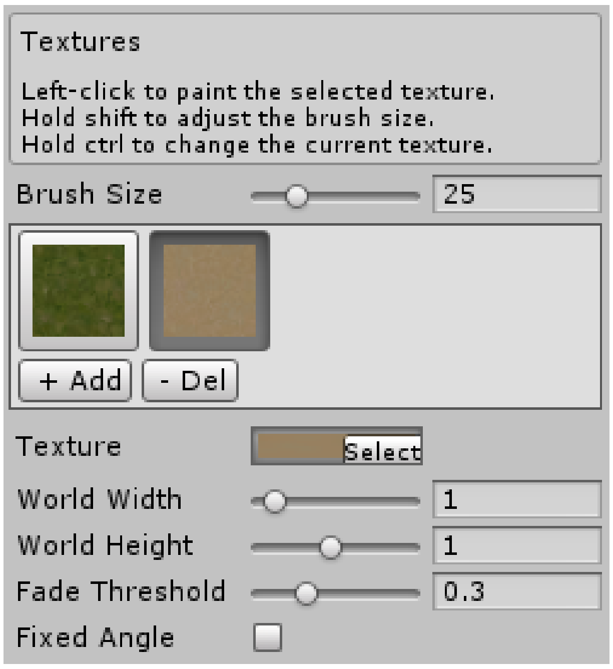
Figure 6: *Defining the fill area of the terrain by dragging boundary handles.*

- **Fill Texture** — texture used to cover the inside area of the terrain.
- **Tile Width** — scale of the width of the fill texture in world units.
- **Tile Height** — scale of the height of the fill texture in world units.
- **Tile Offset X** — x offset of the fill texture in world units.
- **Tile Offset Y** — y offset of the fill texture in world units.
- **Boundary Rect** — boundary rectangle which is displayed in the scene as a blue rectangle with handles in the corners.

2.4 Surface Textures

Click on **Textures** to select the tool for editing and painting surface textures. The terrain surface is covered by a thin stripe of grass, sand, etc. Using this tool you can specify the texture used for each particular node of the terrain.

But first you have to define some textures. To do so click **Add** to add new texture to the list. You can delete it afterwards by clicking on **Del**. When you select a texture from the list you can paint it in the scene but also edit its parameters. The most important is the texture itself. Change it by dragging a texture asset from the Project view to the **Texture** field. In the **e2d demo/Terrain Textures** folder you can find several default textures you can use. The tool should look similar to Figure 7. After you're done, you can change other parameters from the following list:



Textures

Left-click to paint the selected texture.
Hold shift to adjust the brush size.
Hold ctrl to change the current texture.

Brush Size 25

Texture

World Width 1

World Height 1

Fade Threshold 0.3

Fixed Angle ☐

Figure 7: *Settings of the texturing tool.*

- **Texture** — the texture asset handle.
- **World Width** — width of one tile before it is repeated (world units).
- **World Height** — height of one tile before it is repeated (world units). It also influences the size of the terrain surface mesh.
- **Fade Threshold** — determines where the texture starts to blend into the fill texture.
- **Fixed Angle** — if checked the texture is not aligned to the terrain surface.

When you're done select the texture you want to paint, set the **Brush Size** and hover your mouse over the terrain. As in case of the height brush you should see blue highlights above nodes around your mouse. When you click the mouse the texture will be painted on the highlighted nodes, as in Figure 8.

Again, there are some shortcuts for adjusting the brush properties. When you hold **shift** then by moving your mouse you can change the brush size. While holding **ctrl** you change the currently selected texture.

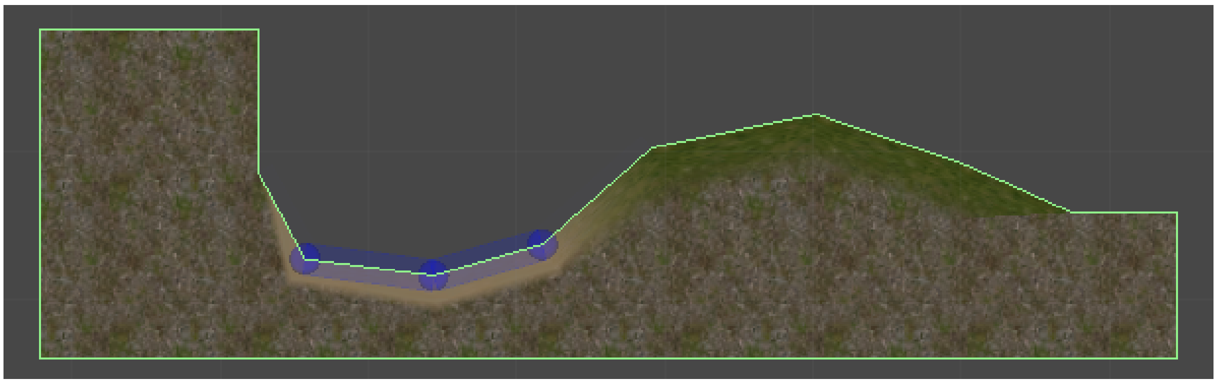


Figure 8: *During the process of painting the sand texture on the terrain.*

2.5 Grass

When the terrain surface itself is ready it helps a great deal to add some vegetation on top of it. To do so select the **Grass** tool. The inspector looks similar to the one of the Textures tool. Again, there are textures used for grass painting, so the first thing to do is to click the **Add** button and add some grass textures. By clicking on **Del** you can delete the selected grass texture.

While painting the terrain surface you select specific texture to apply it to the terrain. But when you're painting grass the textures are mixed together automatically. Instead, by left-clicking on the terrain surface you increase the density of grass. If you hold **ctrl** the density is decreased. As in case of texture painting or the height brush the area to be influenced by the brush is highlighted in blue color in the scene. Also, there are some shortcuts adjusting the brush properties. When you hold **shift** then by moving your mouse you can change the brush size. While holding **shift and ctrl** you change the brush opacity. The opacity influences how much the grass density will be changed.

The final look of the grass is influenced by several parameters. There are some global parameters valid for all grass textures:

- **Random Scatter** — random displacement of individual grass clumps.
- **Wave Speed** — influences how fast the grass waves in wind.

Each of the grass textures has its own list of parameters influencing its look:

- **Texture** — the texture asset handle.
- **Wave Strength** — influences how much the grass waves in wind.
- **Width Random** — randomness of the width of grass clumps.
- **Height Random** — randomness of the height of grass clumps.
- **World Size** — standard size of grass clumps (world units).

The grass when set properly should look similar to Figure 9. Note that the grass waves only when the game action is running.

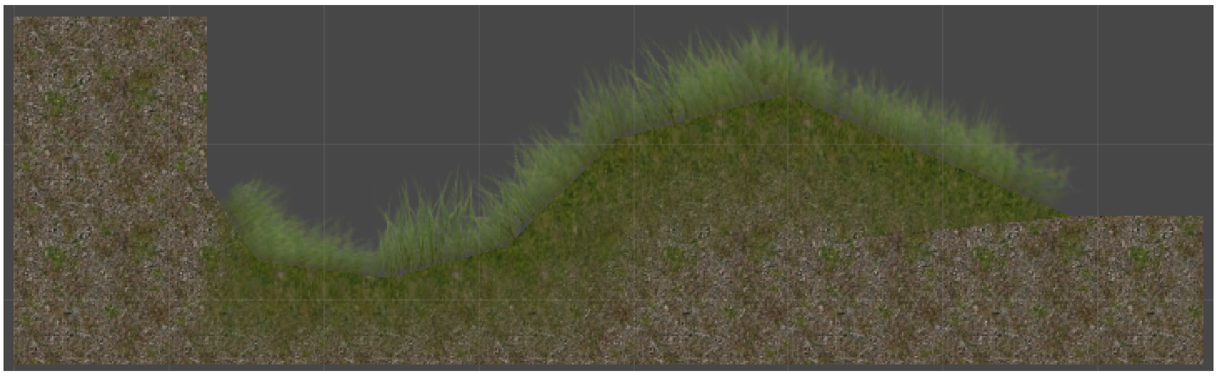


Figure 9: *Sample terrain with some grass painted on it.*

3 Terrain Generator

Now that we know how to manually edit the terrain we can talk about the generator. The generator automatically produces the terrain (or part of it), so that you have some basis to start your work from. It is not intended to be used completely independently, and some manual work might be necessary.

3.1 Surface Curve Generator

To select the curve generator tool click on **Create** in the inspector of the **E2d Terrain Generator** component. Two things will happen: you'll see a blue rectangle in the scene and a lot of parameters to set in the inspector. The blue rectangle is called **target area** and defines the space where the generator works. It can either grab existing nodes from the area and adjust them, or (as in case of the curve generator) put the generated terrain there. Note that the target area can be rotated and scaled, and it also affects the result of the generator. Basically, you can generate the terrain upside down if you want. It can be handy for caves. The target area can be adjusted by dragging handles in the scene (see Figure 10) or by adjusting its parameters in the inspector. There are two global parameters:

- **Rebuild All** — if checked the whole terrain is rebuilt from scratch. Otherwise only the part in the target area is deleted and recreated.
- **Step Size** — space between two nodes (along the X axis) of the generated terrain. This basically defines the density of the nodes and can lead to better looking terrain but worse performance.

Below that are tabs with settings of different methods of the generator. It is important that all of the methods are mixed together to produce the final result. The way they are mixed is influenced by the **Blend Weight** parameter of each of the method. You can set all of them to zero, but nothing will be really generated then.

The parameters of the methods can be quite cryptic at times but each of them has some presets predefined which you can use instead. By selecting a preset the parameters are set, so that the generator produces something sensible. For example, a preset called „Rolling Hills" produces smaller smooth hills. Try it by yourself, but don't forget to increase the **Blend Weight** parameter or otherwise nothing will be generated!

3.1.1 Peaks

As you can see the last tab is titled **Peaks**. It is not generating anything. Instead, it is used to define your own peaks you would like to have in your terrain. If you select the tab, you can then click in the target area and red dots will appear. They represent the peaks (or hills, mountains, etc.). You can drag them around or hold **shift** and delete them. The peaks are used in some of the generator methods mentioned below.

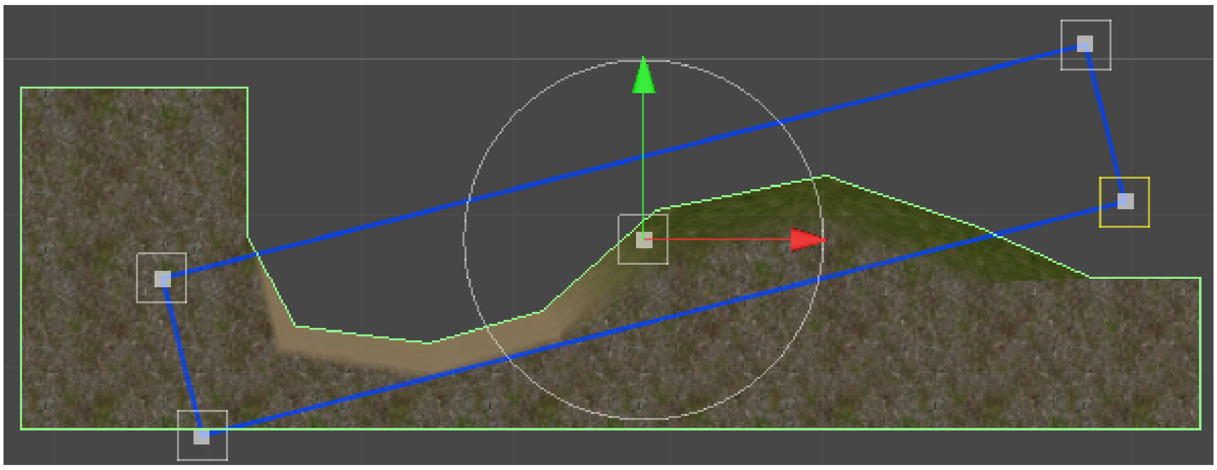


Figure 10: *Target area defining space where the generator works.*

3.1.2 Perlin Generator

The Perlin generator works by producing noise and then creating the heightmap based on the noise. The noise can be controlled by the following parameters[1]:

- **Freq. / Meter** — the higher the more hills will appear in the terrain. More hills also mean that they will be more steep.
- **Octaves** — influences the level of detail. Increase if you want to have small bumps in the terrain.
- **Persistence** — the amount of detail at lower levels. Increase if you want the small bumps to be bigger in the height.

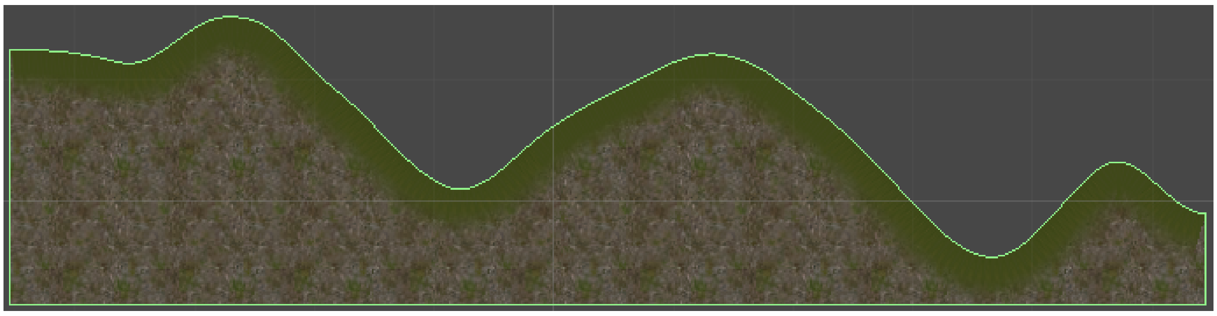


Figure 11: *Terrain created using the Perlin generator.*

3.1.3 Midpoint Generator

The midpoint generator produces similar kind of terrain as the Perlin generator, but it is generally more spiky and jagged. The advantage is that it attempts to satisfy the peaks you have defined. However, this works only when **Freq. / Meter** is high enough. The parameters are:

- **Freq. / Meter** — the higher the more hills will appear in the terrain. More hills also mean that they will be more steep.
- **Roughness** — the higher the more jagged and spiky the terrain is.
- **User User Peaks** — if checked the user-defined peaks are taken into account.

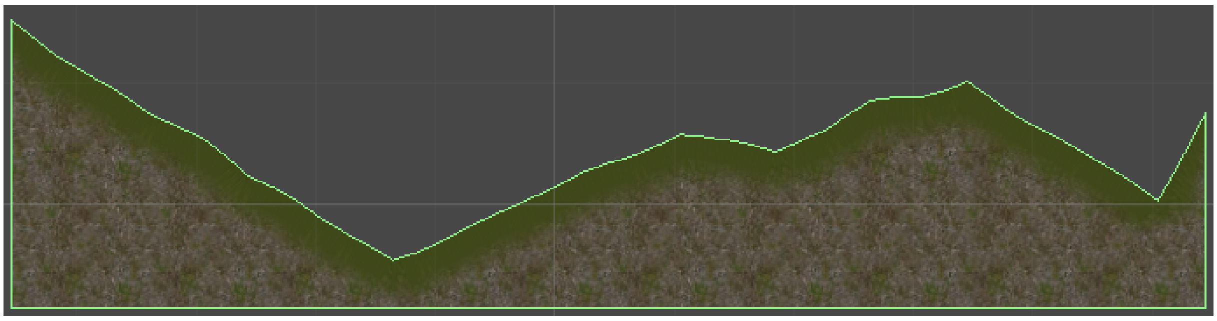


Figure 12: *Terrain created using the midpoint generator.*

[1]The parameters are explained in detail here: `http://freespace.virgin.net/hugo.elias/models/m_perlin.htm`

3.1.4 Voronoi Generator

This generator randomly defines several peaks (points in the target area similar to the user peaks), and then creates the heightmap by interpolating different curves through the peaks. The advantage is that you can very well use peaks defined by you (check **Use User Peaks**). The parameters are:

- **Peak Type** — shape of the hills. It can be straight, round or plateaus.
- **Freq. / Meter** — amount of peaks generated.
- **Peak Ratio** — ratio of non-zero peaks. Lower if you want more plains.
- **Peak Width** — influences the width of the hill-tops.
- **Use User Peaks** — if checked the user-defined peaks are taken into account.

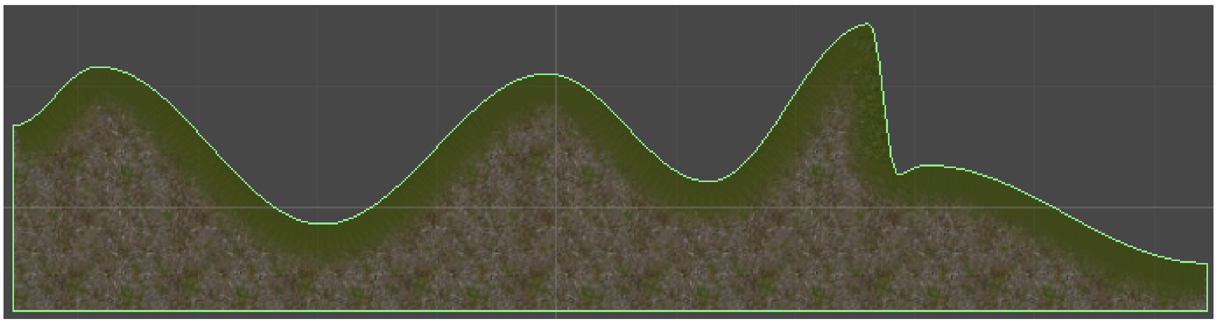


Figure 13: *Terrain created using the Voronoi generator with round peaks.*

3.1.5 Walk Generator

This is a bit different generator. Instead of trying to produce the height values directly it simulates a person walking up or down a hill. He starts walking straight ahead, and every so often (based on **Freq. / Meter**) makes a big turn up or down. This way the generator produces terrain which is more suitable for characters walking on it. It is controlled by the following parameters:

- **Freq. / Meter** — number of points per meter where a bigger turn is performed.
- **Change / Meter** — magnitude of the turns.
- **Cohesion** — controls the overall steepness of the terrain. Lower if you want the terrain to be more tight.

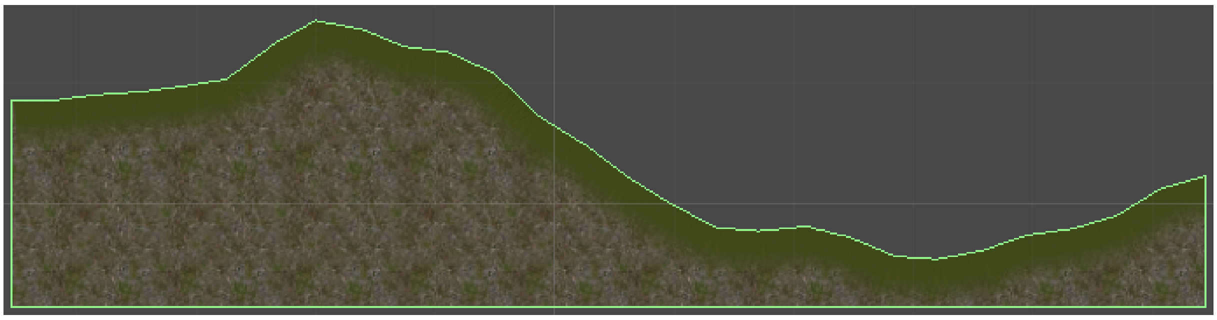


Figure 14: *Terrain created using the walk generator.*

3.2 Smoothing

The terrain after it is automatically generated or manually created can have many spikes or sharp edges. However, for games it is usually better if it is smoother, so that characters can walk on it. If you select the **Smooth** tool you can smooth the surface curve of the terrain. Again, the area to be smoothed is specified by the target area. Note that the angle of the target area influences the direction under which the height values are smoothed. The only parameter controlling the smoothing is **Iterations** which influences the amount of smoothing to be done. See Figure 15 to compare a terrain before and after smoothing.

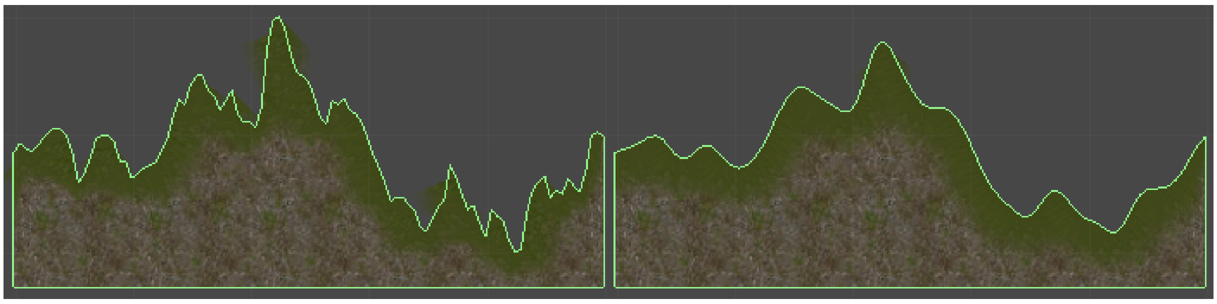
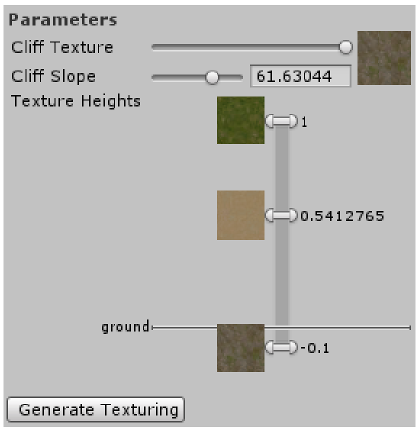


Figure 15: *Terrain before (left) and after (right) smoothing.*

3.3 Texturing

In many cases it is necessary to paint the terrain textures manually to reflect the needs of the level design. But it's always good to have something to start from. That's the aim of the **Texture** tool which applies texturing to the terrain based on the height and on the slope of the terrain. The generator uses the target area definition to know where and how to apply the texturing. A height limit is assigned to each texture defined in the **Textures** tool of the base **E2d Terrain** component. It is defined using a vertical slider, as seen at Figure 16. Then, when the node is above this limit the particular texture is assigned to it. However, if the slope of the terrain around the node is higher than **Cliff Slope**, then the **Cliff Texture** is used instead. See Figure 17 for an example.



Parameters
Cliff Texture
Cliff Slope 61.63044
Texture Heights
1
0.5412765
ground
-0.1

Figure 16: *Defining height limits for textures.*

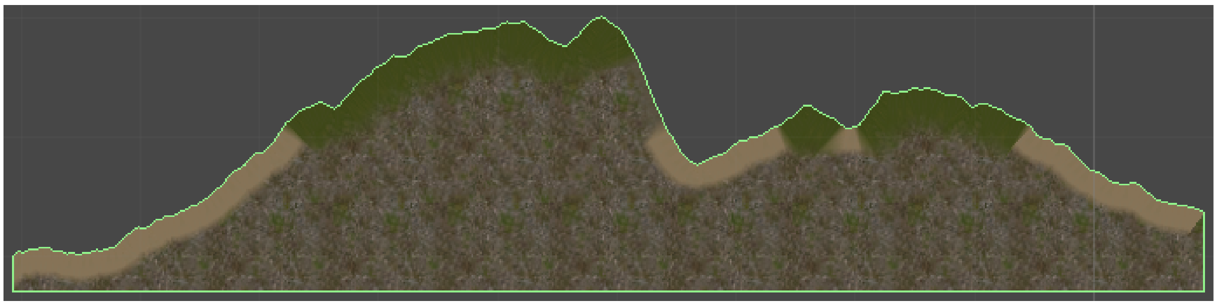


Figure 17: *Terrain with automatic texturing applied.*

3.4 Grass

While it is possible to paint grass manually using the terrain editor, it is easier to use the generator when the terrain too large. The generator sets the density of grass to the terrain based on the height and on the slope of the terrain. The generator uses the target area definition to know where and how to paint the grass. Only nodes whose height is between **Min Height** and **Max Height** are assigned any grass. Also, the slope of the terrain around the node must not exceed **Cliff Slope** – we don't want any grass on cliffs! The density of grass to be set is specified by the **Density** parameter.

4 Conclusion

Now you should be able to use both the editor to manually create and adjust the terrain and the generator to automatically produce the shape and texturing of the terrain. The terrain and the generator components can also be used from scripts. You can access them as any other components and their methods. However, you should know that if you manually change the terrain data (such as `TerrainCurve`) you must rebuild the runtime data by calling `RebuildAllMaterials()` and `RebuildAllMeshes()`. When you're using the generator you don't have to do that. To see what the methods do read the Doxygen documentation.