



PATH PAINTER

By 3D Haven

Path Painter is a path painting system that enables the creation of gorgeous looking paths, roads, ramps, play areas, lakebeds and riverbeds on Unity terrains quickly and intuitively.

Version 2.1.2

Contents

About 3D Haven	3
Tutorials, Chat, Ticketed Support	4
Installation	5
What is Path Painter ?.....	6
Path Painter Systems and Components	7
Paint	7
Interface	7
Main Controls	8
Brush Settings.....	10
Shaping	12
Textures	14
Vegetation	17
Info, Tips and News	18
Status Bar.....	19
States.....	20
Paint Mode.....	20
Edit Mode	25
Auto Ramp Feature	27
Ramping Direction.....	27
Neighbour Terrain Painting	29
What are Neighbour Terrains and How Do They Work	29
Requirements	30
Texture Auto Propagation Feature.....	31
Project based settings	31
Limitations in older Unity versions.....	32
Help at Your Fingertips	32
Help System.....	32
Tooltips	33
Introductory Workflows	34
Opening Path Painter.....	34
The first path	35
Ramps, Straight Lines, Mixed Lines	43
Adding to Paths	47

Creating a mixed (straight and curved) path.	50
Connecting Elevated Paths, Riverbeds	51
Blending Embankment Textures	52
Path and Vegetation	54
MapScaler	60
Interface	60
Work Flow	60
More Tab	61
Paint API Usage	61
Paint Modes	62
Just Paint	62
Change All Settings and Paint	63
Change Any Settings and Paint	64
Bulk Painting	65
Line Painting	66
Real Life API Example	67
Using the API in Runtime (e.g. in-game)	68
Download Runtime API Demos	69
Troubleshooting	70
The shape of the path is jagged, not smooth	70
The painted texture has sharp edges	70
Grass clearing is inaccurate and disproportionate	70
Can't Nicely Connect Raised Paths or Lowered Riverbeds/Paths	70
The Embankment is Not Being Painted Randomly	70
Performance Is Not Ideal When Painting Large Scale	71
Terrain Editing Reset When I Update My Path	71
Path Painter is not painting on some of the terrains in the scene	71
Path Painter Layer Auto Propagation duplicates textures on the terrain	72
Path Painter doesn't remember my texture selection	72
There are visible texture seams where the terrains meet	72

About 3D Haven

Powerful, simple, beautiful. Friendly tools. Gorgeous games.

3D Haven empowers artists and developers to bring their vision to life by making it very easy and quick to create. Leverage our high-quality products to take the pain out of creation and focus on creating amazing games.

Learn more at our website: <http://www.3d-haven.com/>

Tutorials, Chat, Ticketed Support

Thanks for purchasing Path Painter!

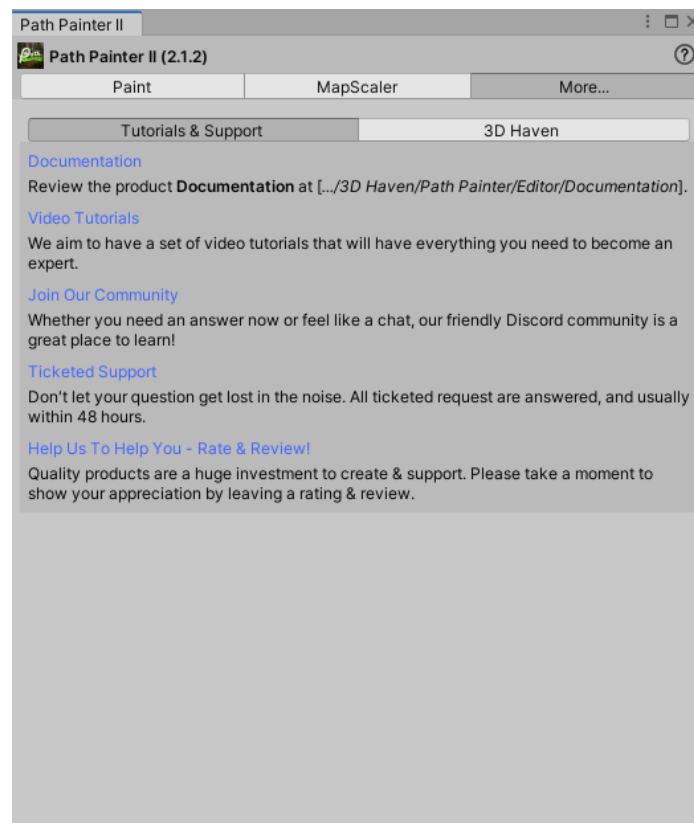
Sometime new tools can be a little overwhelming. To help you with this we have created an awesome support network for you. You can also get access to these links from the Path Painter menu in Unity (Window -> 3D Haven -> Path Painter II).

Tutorials: <https://bit.ly/PathPainterTuts>

Have A Chat: <https://bit.ly/3DHavenDiscord>

Lodge a Support Request: <https://bit.ly/3DHavenSupport>

Alternatively, the links can be found at More... -> Tutorials & Support in the Path Painter Window.



Installation

Installing Path Painter will create the following folder structure:

3D Haven

Path Painter

Editor: Path Painter editor components

Demo: Path Painter Demo environment

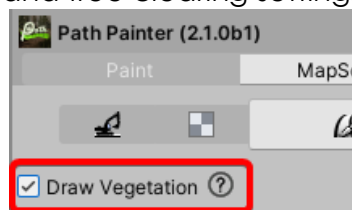
Documentation: Path Painter documentation

Engine: Path Painter engine components

Resources: Path Painter resources

Important Notes

- Path Painter normally exist only in the Editor scope. This means no part of it will be included in your builds and you don't need to remove any part of Path Painter to optimise your builds. This is the reason why the Demos and Documentations are located in the Editor folder.
- If you want to use the Path Painter API at runtime, use the build settings menu (Window -> 3D Haven -> Path Painter II -> Path Painter II Build Settings). For more information see [Using the API in Runtime \(e.g. in-game\)](#).
- Path Painter expects all terrain tiles to be square. If this condition is not met, Path Painter will deactivate and display a message to inform the user of this behaviour. Follow the link provided when this happens to get more information about this and how you can avoid pitfalls of non-square terrains.
- Path Painter operates in standard unity scenes and is subject to the usual performance constraints. Grass and trees can have huge impacts on performance. You can get around this by disabling "Draw Vegetation" after you find the desired grass and tree clearing settings for a set of paths.



Removing Path Painter

To remove Path Painter you can simply delete the Path Painter folder from your Assets. If there's nothing else in the 3D Haven folder, you can delete that folder all together instead.

What is Path Painter ?

Path Painter is a system that enables rapid and precise creation of gorgeous looking paths, roads, ramps, riverbeds, lakebeds, play areas, and possibly mountains on Unity terrains.

With Path Painter you will:

- Paint Paths on your terrains.
- Specify the shape of the path.
- Specify the texturing of the path.
- Specify how the path affects the surrounding vegetation.
- The MapScaler utility is also included to allow scaling of the different maps of the terrain in case they don't fit your path creation needs (Unity could not do this pre 2018.3).

Path Painter is different because:

- **It's standard** – There are no special shaders or other tricks that could cause compatibility problems (you can safely delete Path Painter after you have created your scene);
- **It's simple** – Path Painter does not have a steep learning curve and can be used out of the box by highly experienced level designers or anyone else;
- **It's fast** – Path Painter allows you to create all your paths for a map within minutes, with only minimal fine tuning after creating each. It is paint based, so any touch up needed can be achieved by painting over areas either with Path Painter itself, or one of the built-in Unity terrain tools depending on the situation;
- **It's powerful** – Live feedback gives a lot of power to the users when they are setting up for sets of paths, or finalising a particular one;

Path Painter Systems and Components

You can control Path Painter from the Path Painter Window. Open the Path Painter Window by selecting Window -> 3D Haven -> Path Painter -> Path Painter. The Path Painter Window is made up of the following tabs:

- **Path Painter Window:**
 - [Paint](#) – the Paint interface;
 - [MapScaler](#) – a handy utility included in Path Painter that allows you to scale the terrain maps to higher or lower resolutions (this was/is significant in earlier Unity version that could not do this);
 - [More...](#) – Quick access to more information;

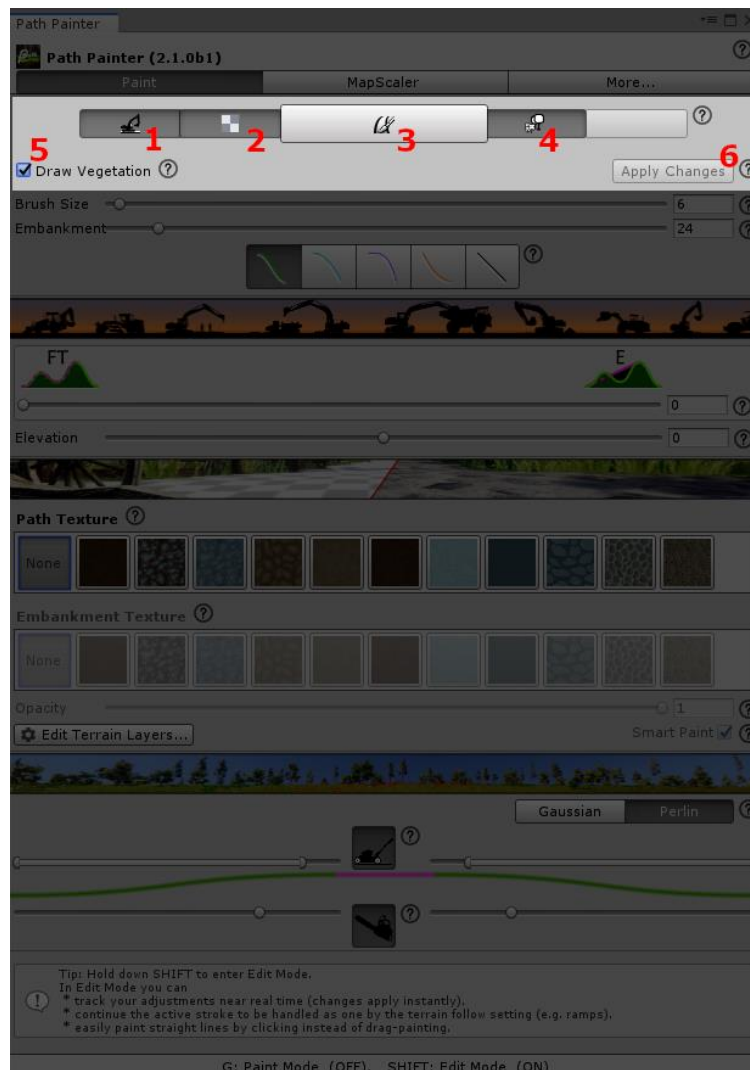
Paint

Interface

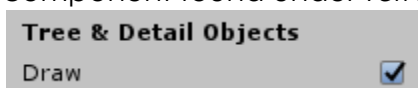
The Paint tab of the Path Painter Window is the main interface. It can be divided into 6 sections, each with their own scope. These are

- [Main Controls](#)
- [Brush Settings](#)
- [Textures](#)
- [Vegetation](#)
- [Info](#)
- [Status Bar](#)

Main Controls



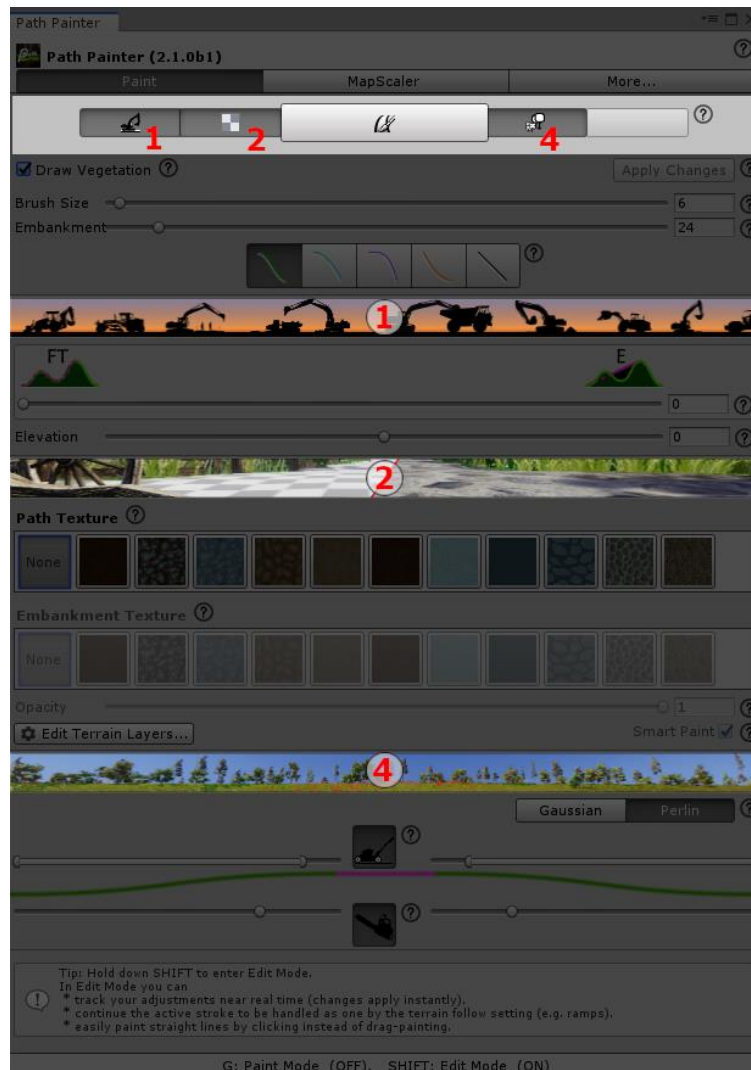
1. **Shaping** – enable/disable terrain shaping with this toggle;
2. **Texturing** – enable/disable path texturing with this toggle;
3. **Paint** – enable/disable painting with this toggle (or by pressing G);
4. **Vegetation Clearing** – enable/disable vegetation clearing with this toggle;
5. **Draw Vegetation** – is a handy shortcut for the same switch of the *Terrain* component found under *Terrain Settings*:



While creating paths people often find themselves in need of switching this on/off;

6. **Apply Changes** – is enabled when there are changes that you can apply to your active brush stroke. It is handy when you want to change a number of settings and apply them all at once to the active brush stroke;

Alternatively, you can click on the header image of each functionality to toggle them on/off.

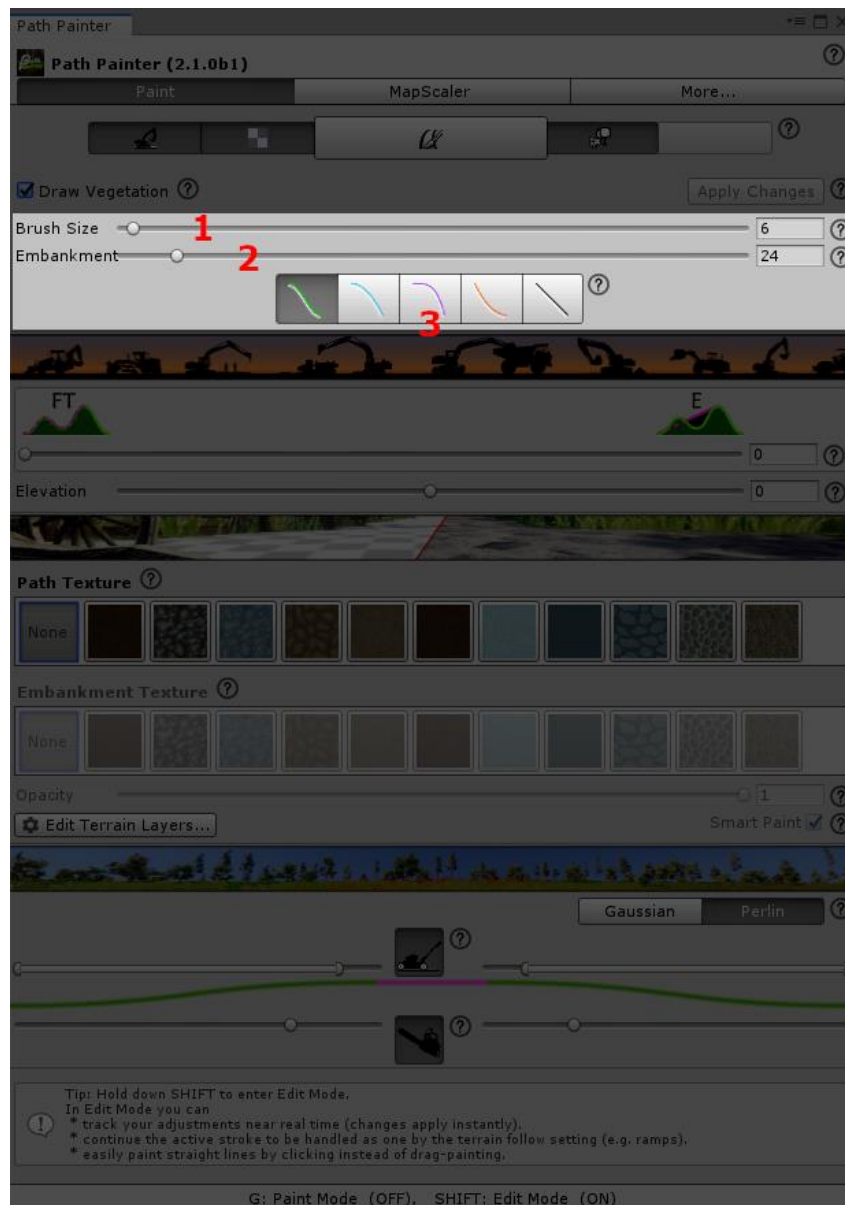


You can use any combination of **Shaping**, **Texturing** and **Vegetation Clearing**.

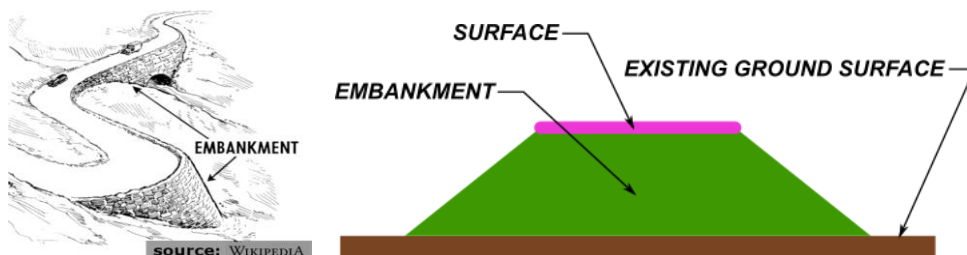
Path Painter operates in standard unity scenes and is subject to the usual performance constraints. Grass and trees can have huge impacts on performance. You can get around this by disabling **Draw Vegetation**. For example, if you are experiencing delays at the end of each painting action and when doing Undo, you can do the following:

1. When you are starting on a set of paths that will have the same vegetation settings, paint one path.
2. Setup everything to your liking.
3. Turn off **Draw Vegetation**.
4. Paint all the paths you need with these settings.
5. At the end you will only experience the delay once when you turn **Draw Vegetation** back on.

Brush Settings



1. **Brush Size** – is the size of the actual path (the surface) in meters (or units);
2. **Embankment (Size)** – is the overall width of the path in meters (or units). The embankment is the area which can be used to blend the path into the surrounding environment. The minimum **Embankment** size is slightly more than the **Brush Size** and the **Embankment** size automatically updates with the **Brush Size**. This keeps the proportions of the brush;



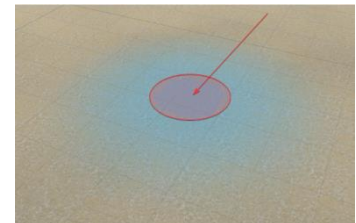
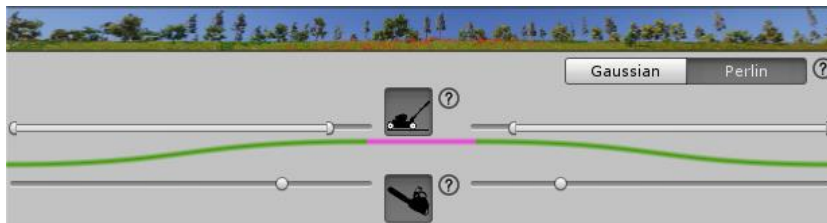
3. **Embankment Curves** – have an effect on the embankment shape and texture. Changing this can have a dramatic effect on how paths blend with the environment. You can monitor the visualisation in the [Vegetation](#) section to see how the embankment changes;

The **Brush Size** and **Embankment Size** are limited by the resolutions of the terrain maps. Fine details cannot be painted on a low-resolution canvas. The maximum size is also limited by the terrain's

- *Heightmap Resolution*, and
- *Control Texture Resolution* (if Texture painting is enabled).

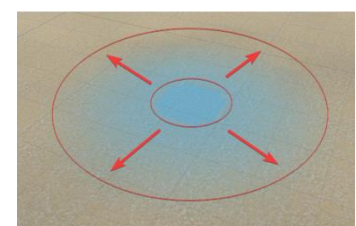
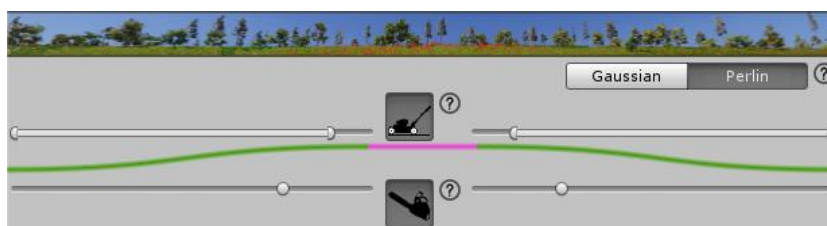
The **Brush/Surface** is

- shown purple in the visualisation found in the [Vegetation](#) section of the GUI.
- the purple centre of the brush as visualised on the terrain below your mouse when painting is enabled.



The **Embankment** is

- shown green in the visualisation found in the [Vegetation](#) section of the GUI.
- the outer, green ring of the brush as visualised on the terrain below your mouse when painting is enabled. It is fading out depending on your settings.

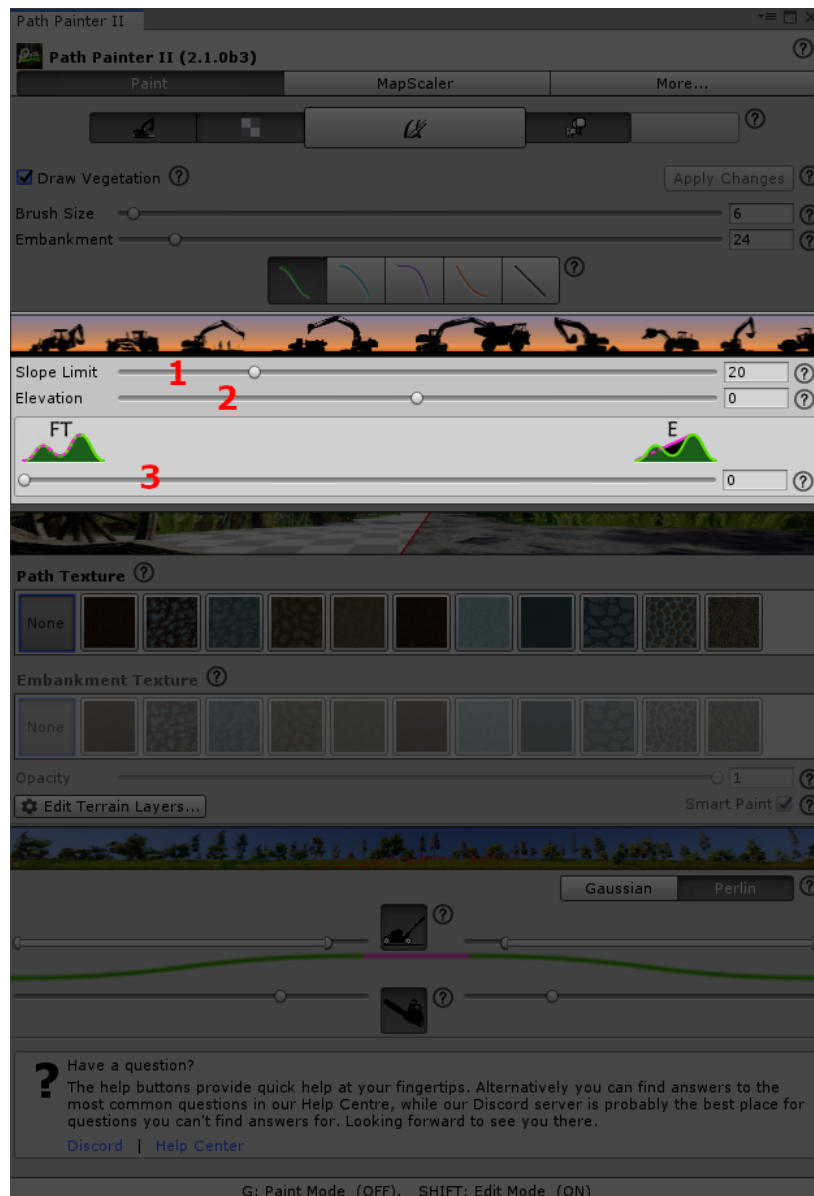


If you find yourself constrained by performance while painting something big, you can try to paint with

- Shaping/Texturing/Vegetation clearing disabled ([Main Controls](#))
- Turn **Draw Vegetation** off (checkbox in the [Main Controls](#) section)
- smaller brush size ([Brush Settings](#))

then apply your desired settings in one go to the path afterwards.

Shaping



1. **Slope limit** – Used by the **Auto Ramp** feature. Slope limit of the path. Auto Ramping is very useful to create paths that are not too steep for players, NPCs, etc. You just do the painting and Path Painter will automatically create ramps as needed. It's that easy. For example, use a few degrees below the *Slope Limit* value of your *FPSController* to ensure the player will be able to navigate the path.
2. **Elevation** – is the elevation of the path. This can be negative, which is a typical setting for riverbeds. Min/max elevation is affected by the embankment size to help keep things in proportion.

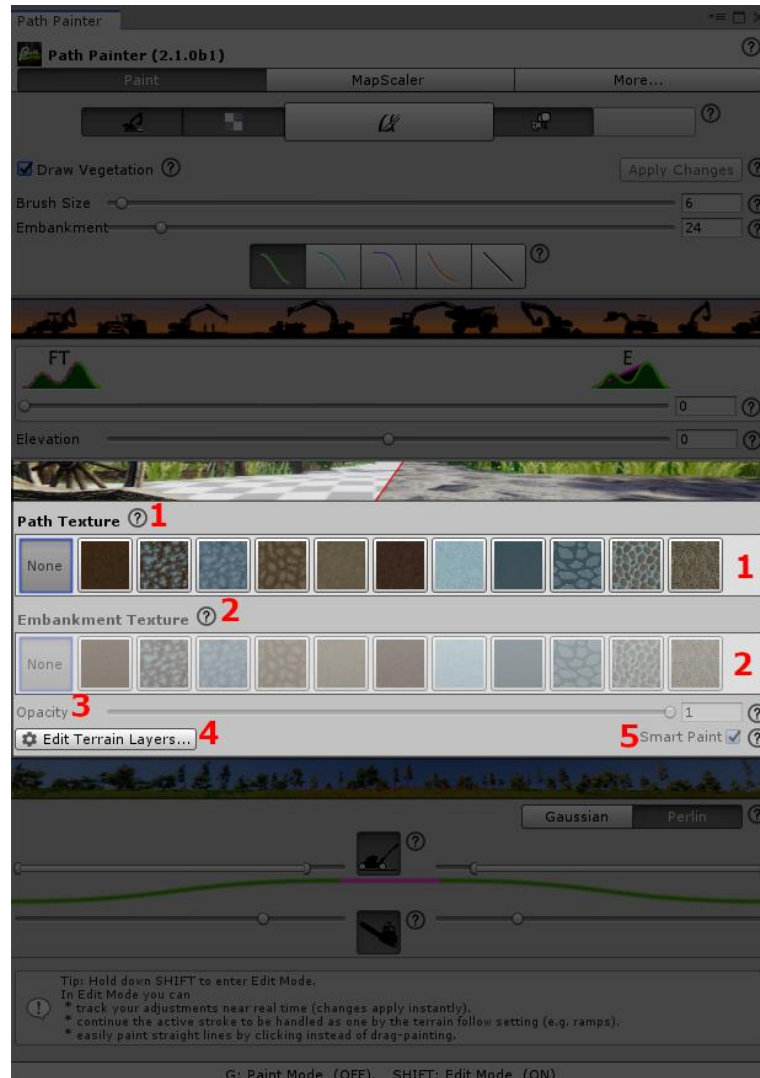
3. **Terrain follow** – With this setting you can make paths

- follow the terrain.
- have an even slope from start to end, and
- anywhere in between.

It comes handy when you want to create some specific ramps or other features. For example, connecting two mountains, islands, valleys, lakes or rivers. It helps making your paths as even as you want.

Textures

Path Painter can apply textures to your paths. In the **Textures** section you can select which of the terrain's textures are used on the path and a number of related things.

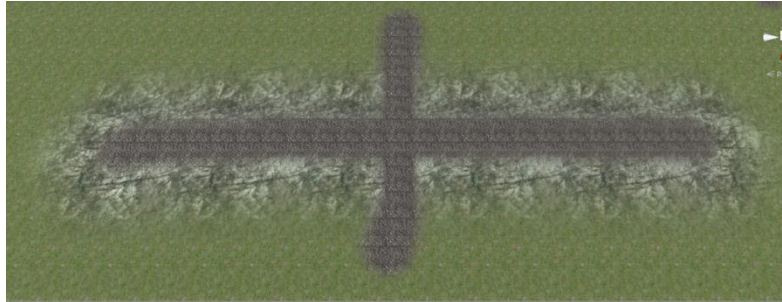


1. **Surface Texture** – is the texture to be applied to the surface of the path (see **Surface** in the previous section).
2. **Embankment Texture** – is the texture to be applied to the embankment of the path (see **Embankment** in the previous section).
3. **Opacity** – is the opacity, or target strength of the path texture, depending on the scenario and the settings. It can be used for CTS height blending or if you wish to paint with a certain opacity;
4. **Edit Terrain Layers** – button shows information about editing terrain layers;
5. **Smart Texture Paint** – will attempt to guess how you would like to apply textures to each path. When disabled Path Painter operates in texture over paint mode that's similar to ordinary painting in that the path will cover existing textures where possible.

Most of the time you will want **Smart Texture Paint** on. In some rare circumstances it might misjudge the situation and not paint the embankment texture. That is when you want painting to cover the existing textures. Switching **Smart Texture Paint** off will achieve that.

For example:

In this situation **Smart Texture Paint** assumes that you want to create a crossing (understandably):



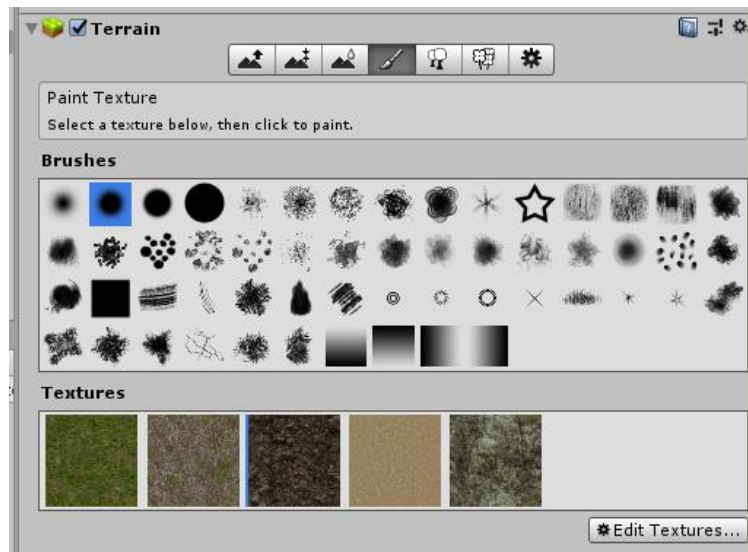
However, this might not always be the case:



Turn off **Smart Texture Paint** to achieve the desired effect:

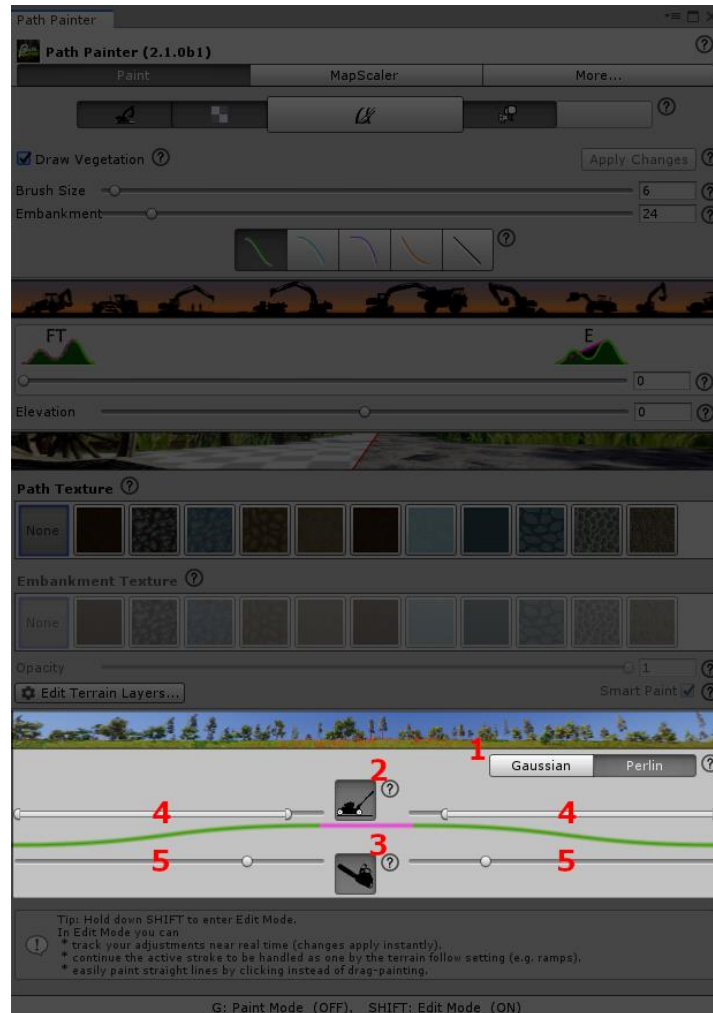


Note: If you need to edit textures of the terrain (or their settings), you can do that under the Paint Texture tool of the Terrain component as usual.



Vegetation

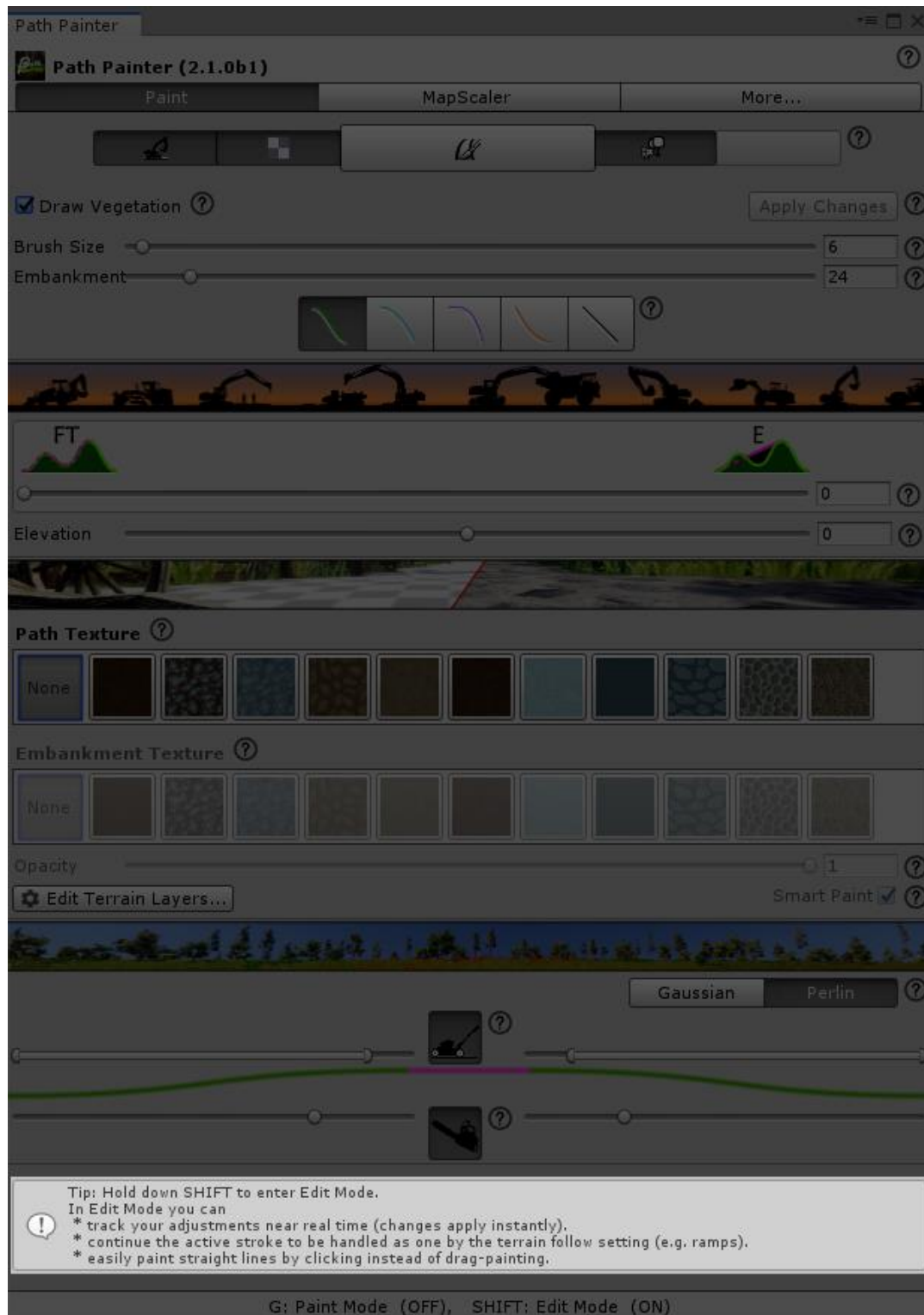
Path Painter makes it easy to clear vegetation in your path and achieve a natural effect.



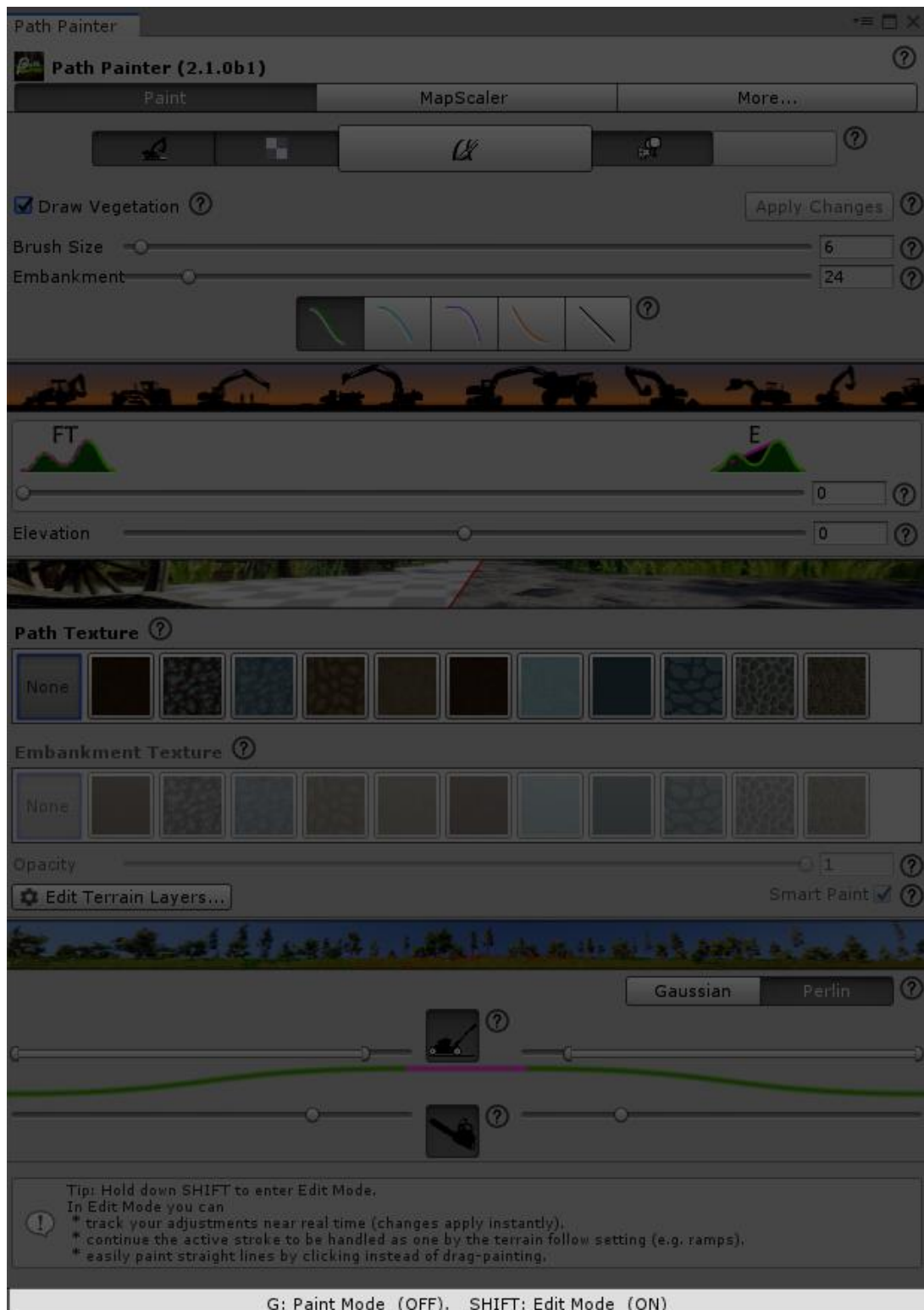
1. **Grass clearing noise** – the chosen type of noise will be applied to grass clearing for a more natural look.
2. **Grass clearing toggle** – Enable/disable grass clearing.
3. **Tree clearing toggle** – Enable/disable tree clearing.
4. **Grass clearing limit and thinning range** – See below the sliders the visualisation of the cross-section of the path and set them accordingly. Full grass clearing will be applied from the centre of the path to the inner knob of the range sliders. A gradual thinning with the selected noise is applied between the two knobs (in the range).
5. **Tree clearing limit** – See above the sliders the visualisation of the cross-section of the path and set them accordingly. Full tree clearing will be applied from the centre of the path to the knob of sliders.

Info, Tips and News

This area will display information, tips and news.



Status Bar

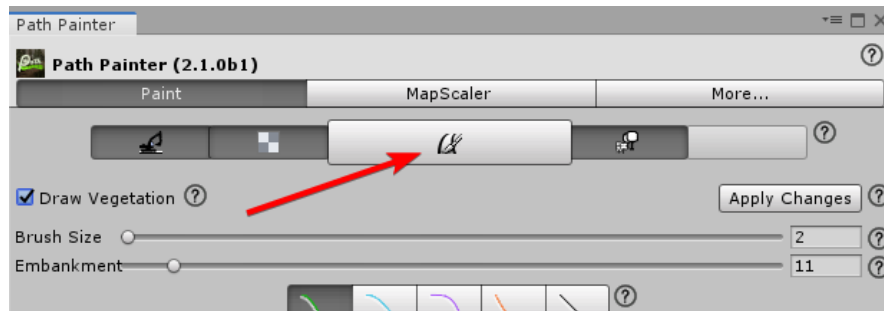


The **Status Bar** displays hotkeys and status information. It's visible here if *Paint Mode* or *Edit Mode* is active, and what functions are available. It is most beneficial while getting familiar with Path Painter.

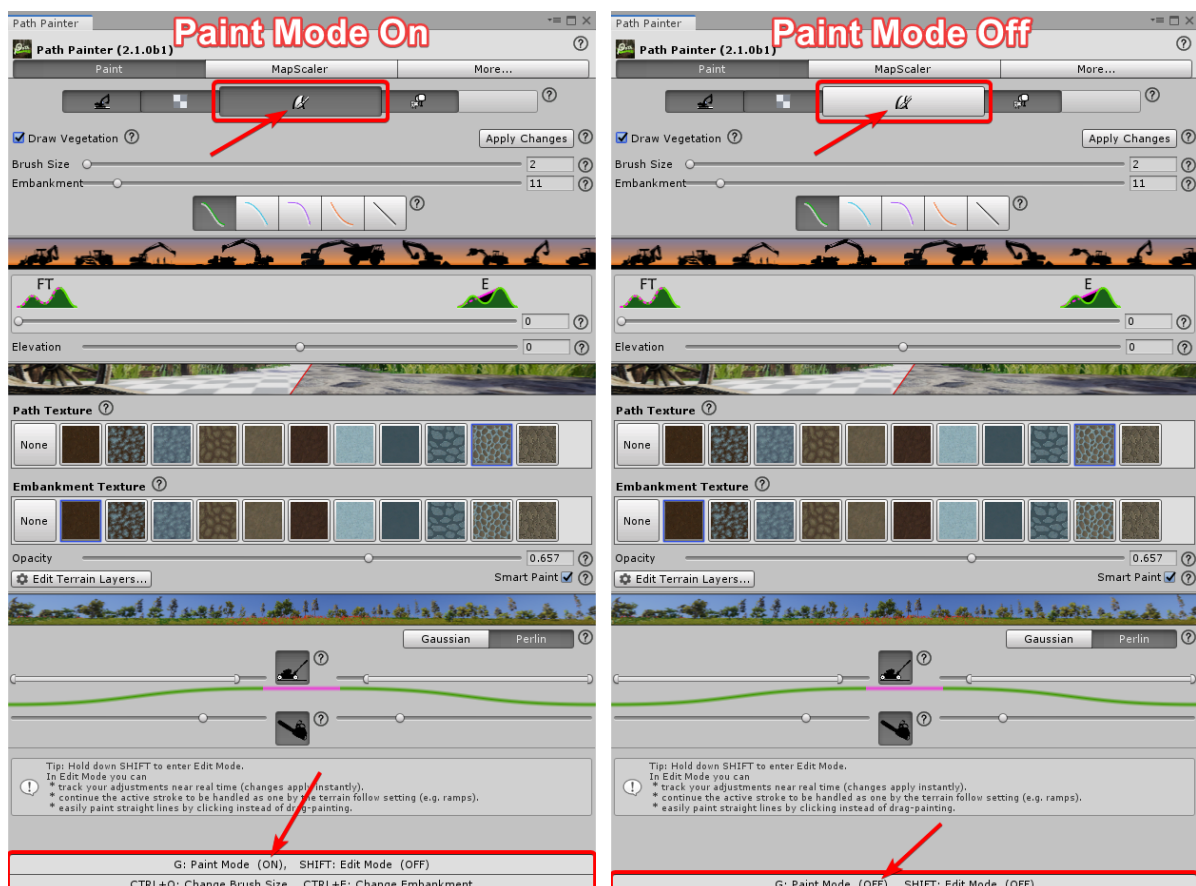
States

Paint Mode

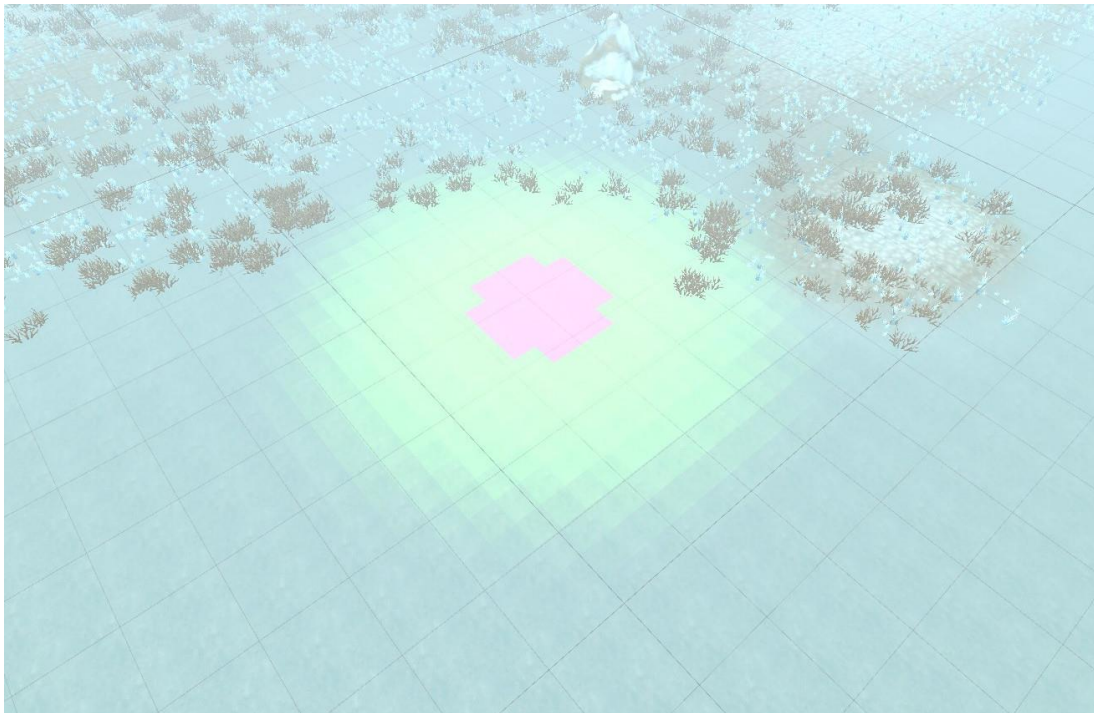
Painting is enabled in **Paint Mode**. You can enter paint mode by clicking the **Paint Button** or by pressing G.



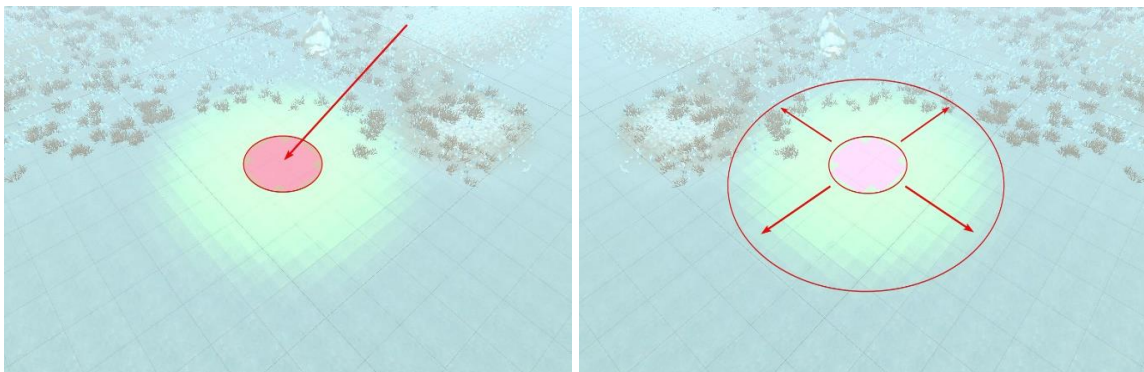
Both the button and the [Status Bar](#) shows when **Paint Mode** is active



and a brush visualisation will be visible on the terrain.

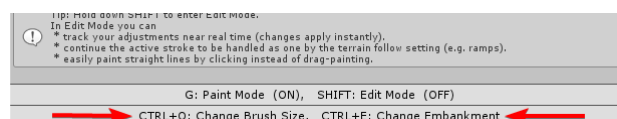


The visualisation displays both the surface and the embankment according to the settings.



The **Brush Size**, **Embankment Size**, and **Embankment Curve** are all visible on the visualisation.

In **Paint Mode** you can use hotkeys to [Change the Brush Size](#) and [Change the Embankment Size](#) in the Scene View. The [Status Bar](#) displays this:

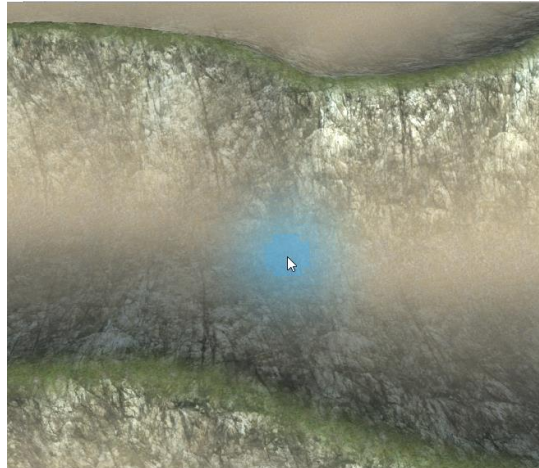


Changing Brush Size - in Scene View

Changing **Brush Size** in the Scene View can often come handy. Note: the same limits apply as in [Brush Settings](#) of the GUI.

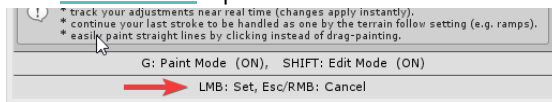
Press the appropriate hotkey (visible in the [Status Bar](#)) while

- the **Paint Mode** is active
- the Scene View is focused and
- the brush is visible on the terrain

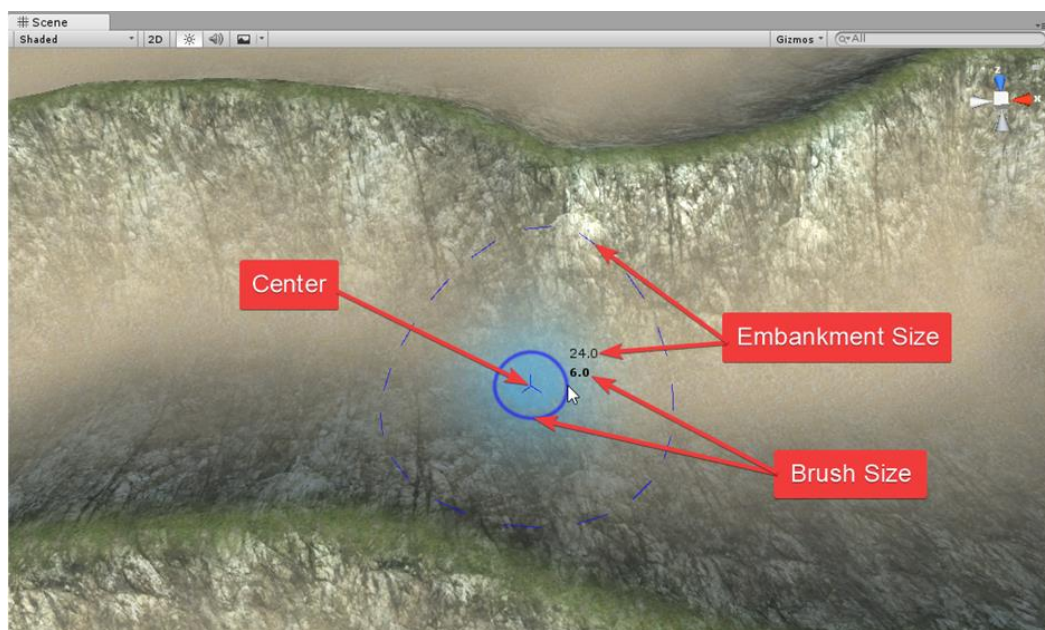


When you hit the hotkey, the following will happen:

- The [Status Bar](#) updates.



- Outlines of the brush appear.
- The mouse cursor snaps into place for the change.
- The current values will be displayed next to the mouse (**Brush Size** in bold).



As you move the mouse, **Brush Size** will update. You can now

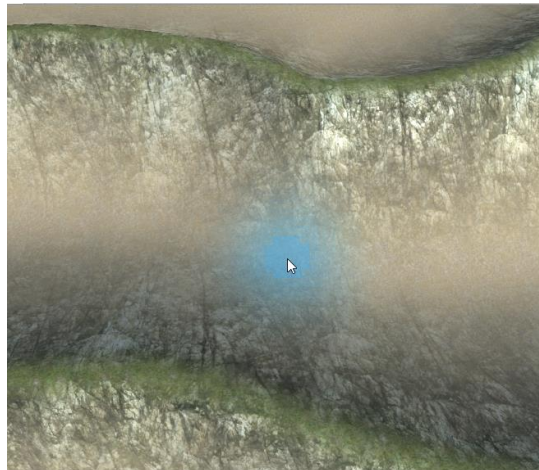
- set the **Brush Size** with the Left Mouse button once you are happy with it, or
- cancel with the Right Mouse button or by pressing Esc.

Changing Embankment Size - in Scene View

Changing **Embankment Size** in the Scene View can often come handy. Note: the same limits apply as in [Brush Settings](#) of the GUI.

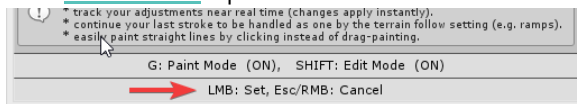
Press the appropriate hotkey (visible in the [Status Bar](#)) while

- the **Paint Mode** is active
- the Scene View is focused and
- the brush is visible on the terrain

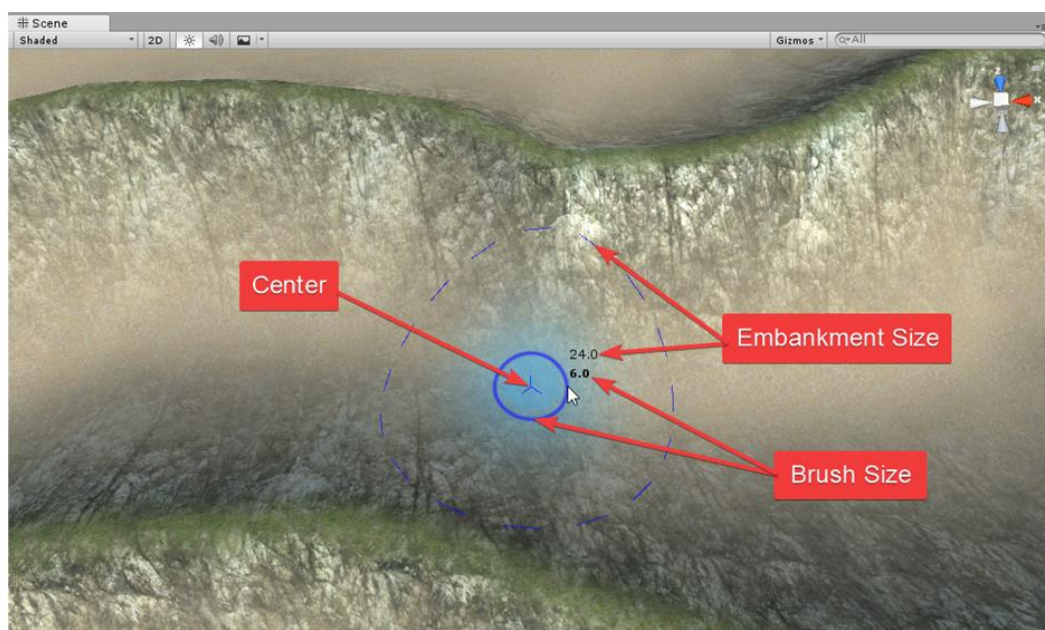


When you hit the hotkey, the following will happen:

- The [Status Bar](#) updates.



- Outlines of the brush appear.
- The mouse cursor snaps into place for the change.
- The current values will be displayed next to the mouse (**Brush Size** in bold).



As you move the mouse, **Embankment Size** will update. You can now

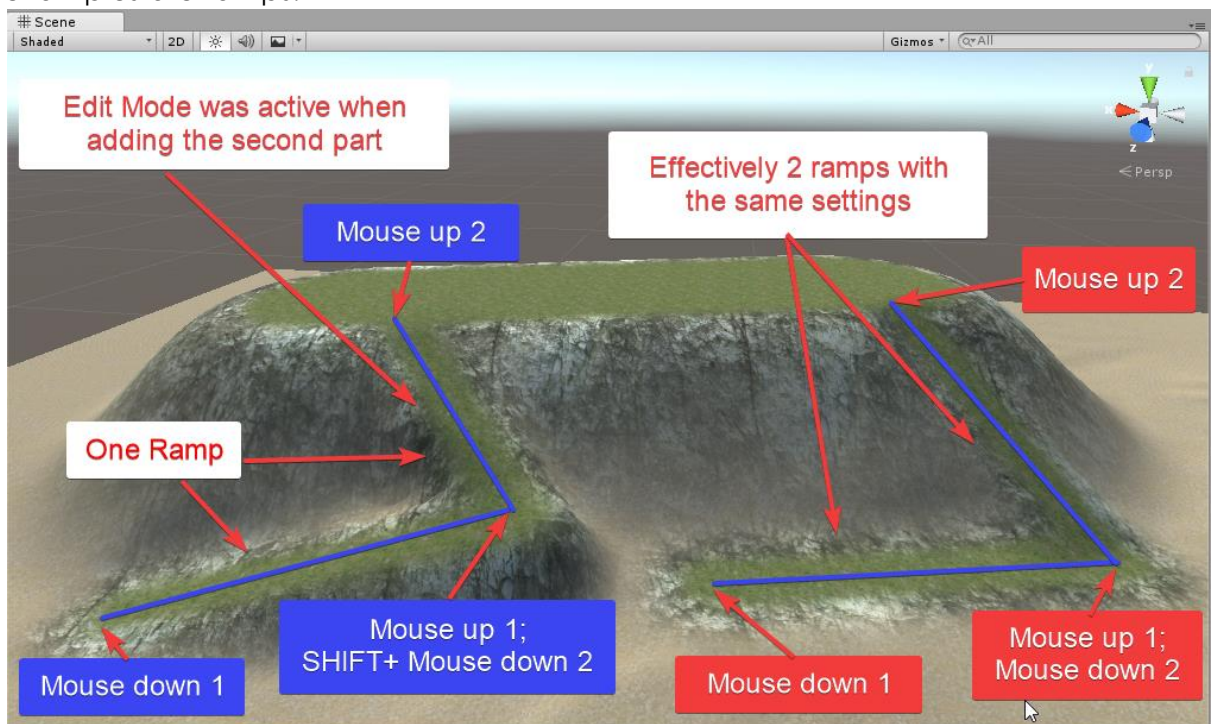
- set the **Embankment Size** with the Left Mouse button once you are happy with it, or
- cancel with the Right Mouse button or by pressing Esc.

Edit Mode

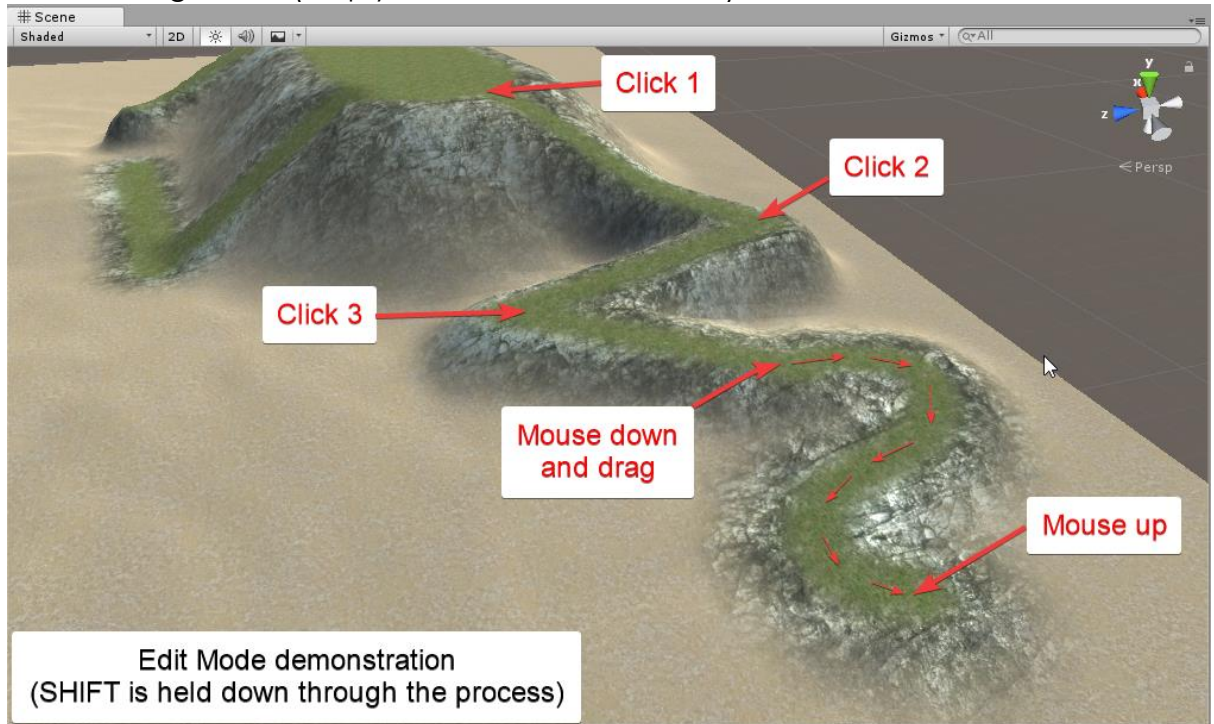
Edit Mode is a powerful feature of Path Painter. It modifies some of the functionality, much like the *SHIFT* key on your keyboard. It's easy to remember this, because by default **Edit Mode** is mapped to the *SHIFT* key. It comes very handy when tweaking and when working on more complex, long paths, that need a lot of viewport navigation during creation.

Edit Mode can be used to

- Edit the active stroke and get live visual feedback.
- Add to the active stroke. This can be important when strokes are needed to be handled as one by the **Terrain Follow** setting (see [Brush Settings](#)). Good examples are ramps.



- Create straight lines (simply click while in **Edit Mode**)



Edit Mode applies if there is an active stroke and it does the following

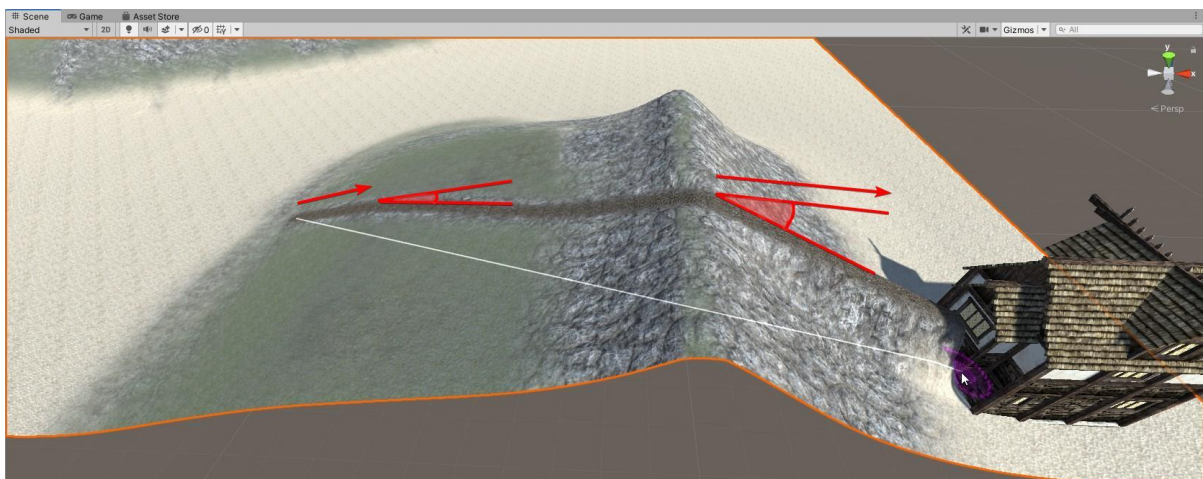
- Instead of starting a new line, the active stroke is going to be continued.
- Adjustments are instantly applied to the active stroke without the need to push the **Apply Changes** button.

Auto Ramp Feature

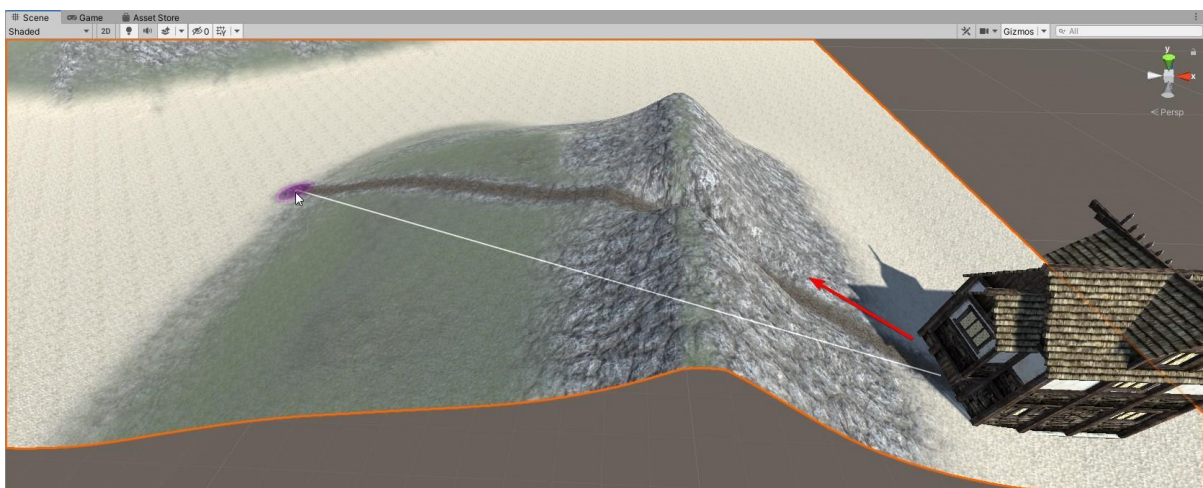
Auto Ramping is very useful to create paths that are not too steep for players, NPCs, and so on. You just do the painting and Path Painter will automatically create ramps as needed. It's that easy. For example, use a few degrees below the *Slope Limit* value of your *FPSController* to ensure the player will be able to navigate the path. This way of painting has been in the plans for about 5 years, from the very beginning, but got lost in the noise until now, when someone reminded me of it (thank you!).

Ramping Direction

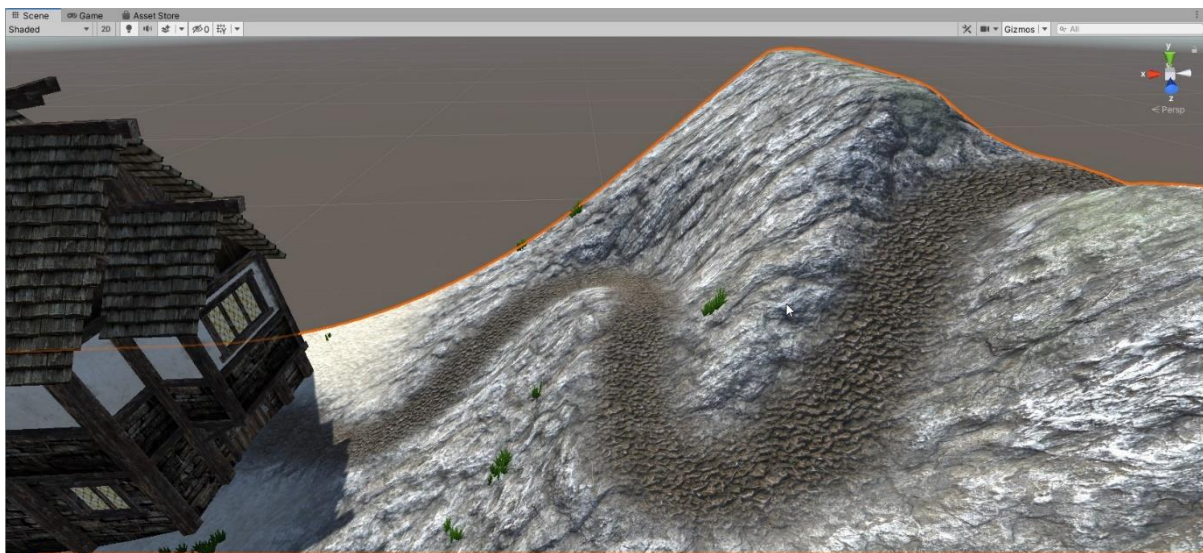
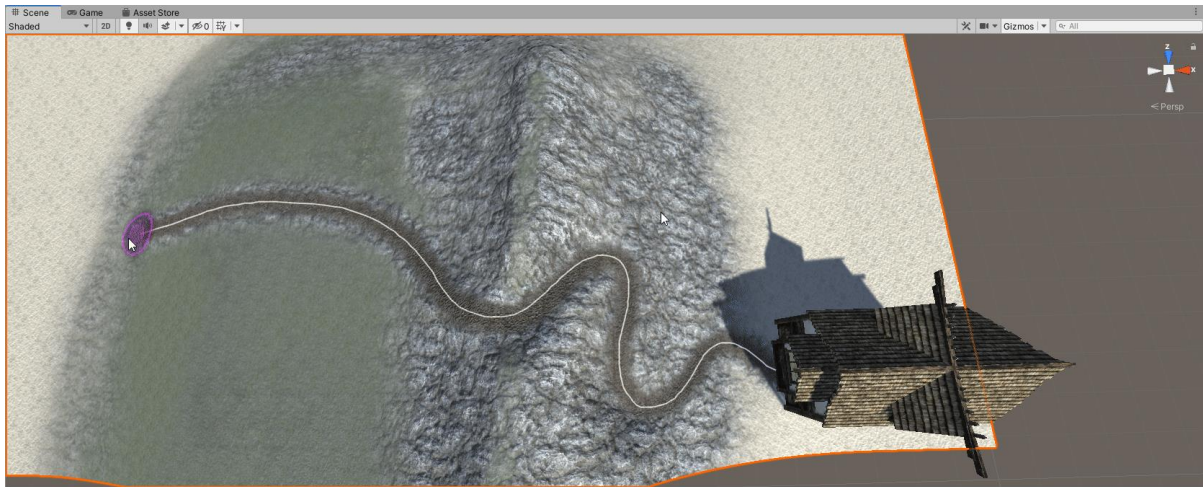
Ramps are created in the direction of your painting. This is worth keeping in mind to achieve precisely what you intend to (although a few paint-undo cycles always help to find the best track for your paths). For example, in situations where you arrive to a target location right next to a steep incline, or decline, it usually works best to start painting from the target location, because painting towards it might not have enough space for your ramp:



While painting the other direction ensures that your path correctly reaches the target:



Most often you probably won't paint your paths like in the example however. Winding serpentine always look cool after all.



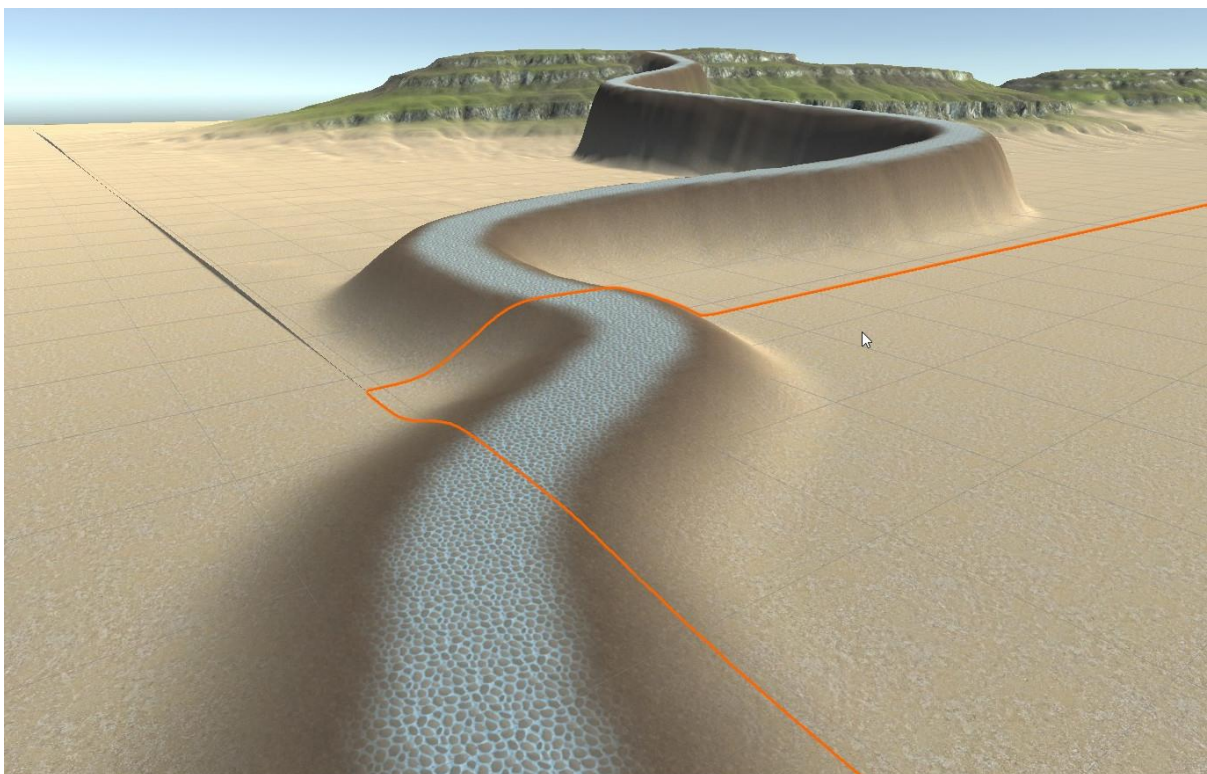
You will also want to consider if you want your ramp to carve into the terrain or rise above it and which direction gets you the result you are looking for. Hope you will have a lot of fun with this feature.

Neighbour Terrain Painting

Work started on Path Painter under Unity 5.6. This was nearly 3 years before neighbour terrain tiles appeared, and supporting such thing didn't even cross the author, Frank's mind, who was new to Unity at the time. This meant that adding this feature required a complete overhaul of Path Painter. **Path Painter II** is pretty much a complete rewrite (and in hindsight, it probably would have been quicker, cleaner and better to start with a clean plate).

What are Neighbour Terrains and How Do They Work

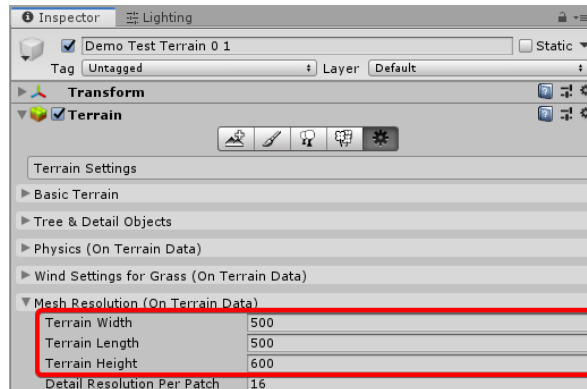
Neighbour terrains are useful if you want to create large worlds, or you want a terrain that's not square shaped. The reason for this is, because Unity terrains don't really like to be non-square, and making them in a rectangle shape can cause all sort of problems in your project down the line (for this reason Path Painter doesn't support non-square terrains to help you avoid the pitfall and avoid the overheads that would reduce its performance). Instead, you can use two or more terrains to create rectangle or other shapes desired, and end up with a scene that's less resource intensive and better for performance. Painting on multiple terrains can be done the same way as painting on one (If they are compatible. See [Requirements](#) to learn about Unity's requirements for terrains to be handled as one).



Requirements

In Unity there are some requirements for Neighbour terrains to be seamlessly painted on as if they were one. There are the following

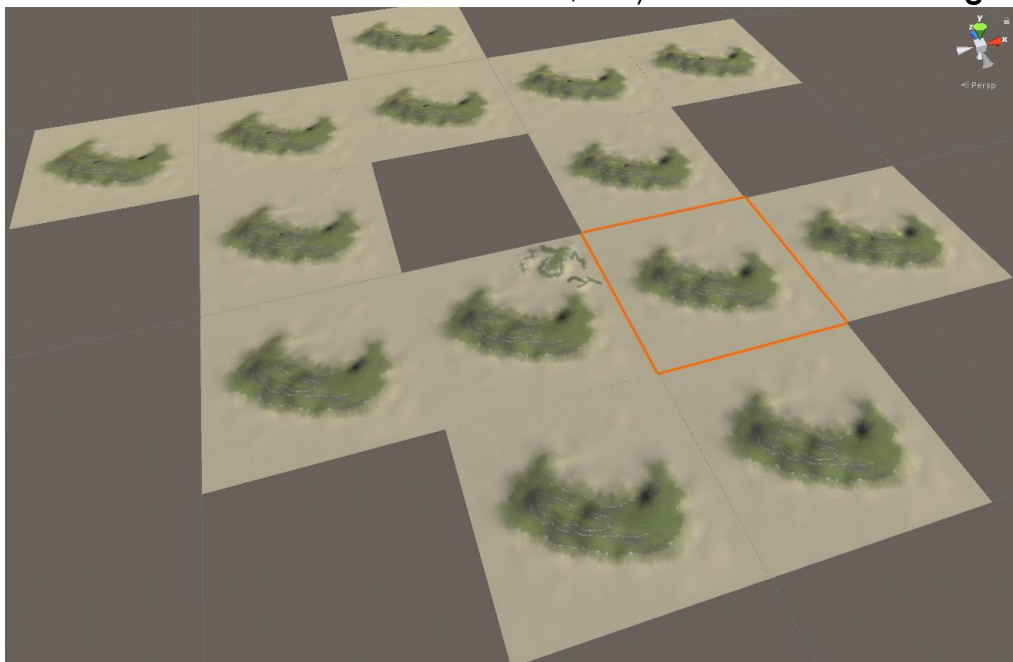
- **Width** and **Length** must be the same on all (you might possibly get away with different height terrains, but you will end up with terrain holes, so generally it's a bad idea).



- Relevant resolutions must be the same on all. In Path Painter's case these are **Heightmap Resolution**, **Control Texture Resolution** and **Detail Resolution**. You will get warnings if any of these are off. (Newer Unity which has neighbour terrain features will also give you warnings relevant to the different built-in terrain paint tools.)

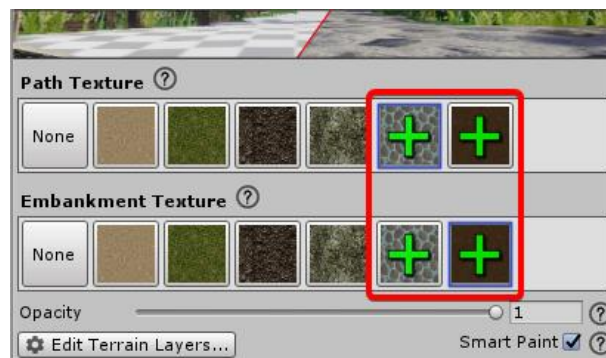


- The terrains must be correctly **Positioned**. This means that they are right next to each other and even when there are holes, they all still fit on the same **grid**.



Texture Auto Propagation Feature

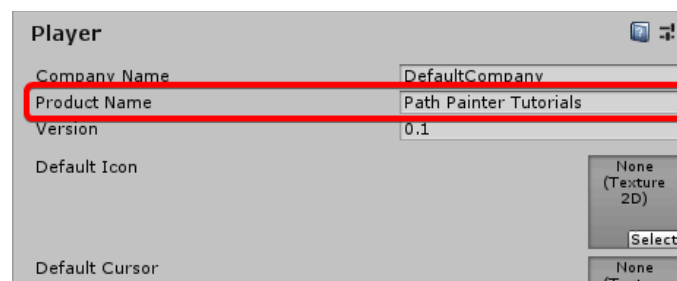
When painting texture Path Painter will automatically add *Terrain Layers* (textures) to the terrains if not already present. This means that you don't need to add the textures you would like to paint with to each terrain individually. In some cases, Path Painter may add a texture that's already on a terrain. The reason behind this is probably because the texture on the terrain have different settings from the one you are painting with. For example, in some situations, users want the same texture with different sizing and tiling on the same terrain. When you select a terrain where the *Layers* you are painting with are not present, you will see a plus sign over them, signifying that these layers will be added to the terrain if you paint on it while they are selected.



Since Path Painter remembers your settings for each project, this can be used to introduce the same *Terrain Layers* to different Scenes of your projects. (Older Unity versions (pre-2018) can't do this.)

Project based settings

Path Painter will remember your settings for each of your projects, so you can pick up where you left off. Path Painter identifies your projects by *Project Name* (Project Settings -> Player -> **Project Name**).




Limitations in older Unity versions

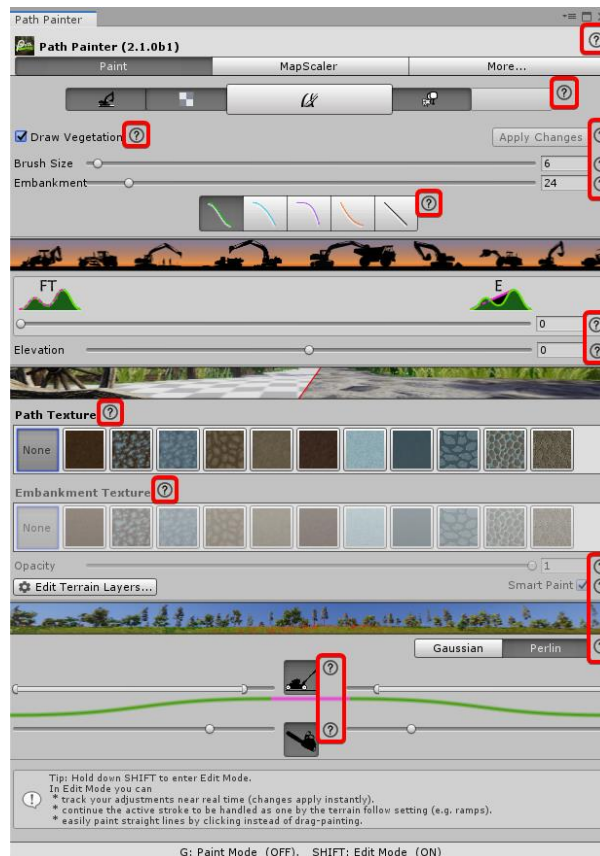
Older Unity versions (pre-2018.3) might not always remember your texture selections. So far, we know of two situations when your last texture selection might not be remembered in a project:

- The same terrain where the layer (that contains the texture) was selected, or the texture itself is not loaded in the scene. For example, if it's a new scene, so the terrain is not in it, or the textures changed on the terrain (where the texture was selected) and the texture is no longer on it, or it isn't at the same position (not under the same index).
- Texture selection from older Unity versions (pre-2018.3) get lost when Unity is updated, because Unity update scrambles that information.

Help at Your Fingertips

Help System

There is help provided all across Path Painter. Click on the  marks if you get stuck, or just need some extra information.



Tooltips

The tooltips also often contain useful information you're just looking for, so it's worth checking them out.

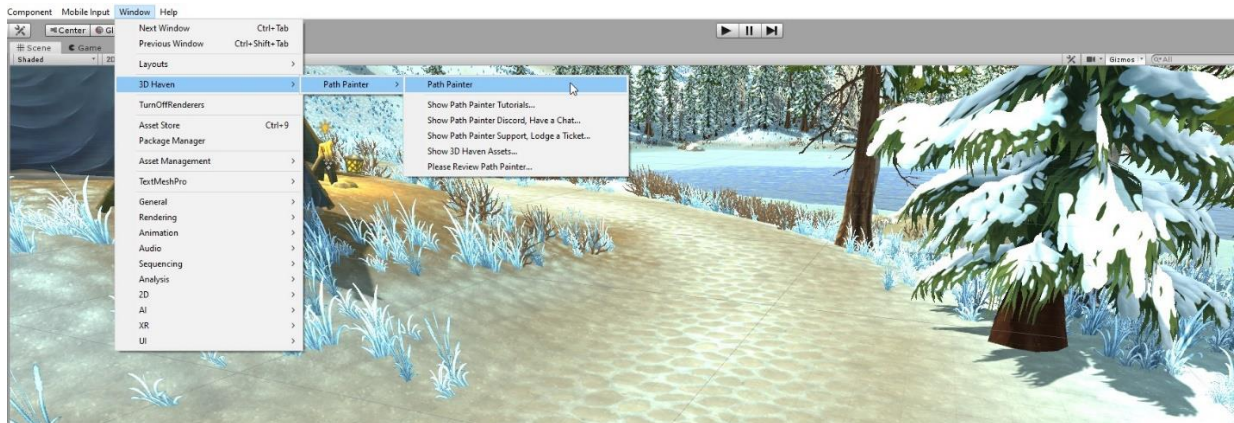


Introductory Workflows

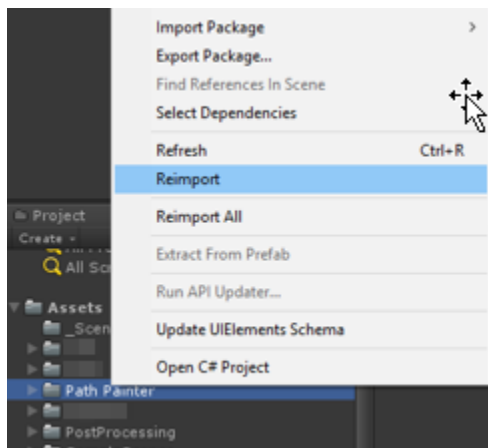
Opening Path Painter

When you first install Path Painter, please wait a little time for the import process to complete.

To open Path Painter select Window -> 3D Haven -> Path Painter -> **Path Painter**



If the menu is not there, it is possible that Path Painter has not yet been completely imported into your project. Please wait for a few moments. If it is still not there then try re-importing Path Painter.

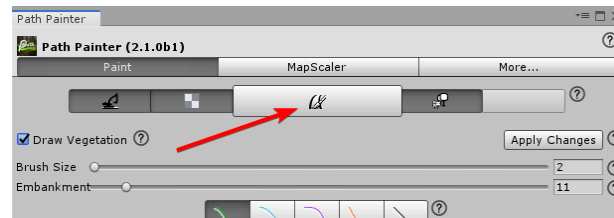


Note: These examples don't cover a lot of traditional path creation techniques. Their aim is to help the user understand how Path Painter works. After completing these, the user will be familiar enough with Path Painter to create any kind of paths. Turn the Auto Ramp feature off for these examples (set slope limit: 90) get learn how these features work. It will be easy to grasp working with the Auto Ramp feature after that and reading the information in the [Auto Ramp Feature](#) section of the documentation.

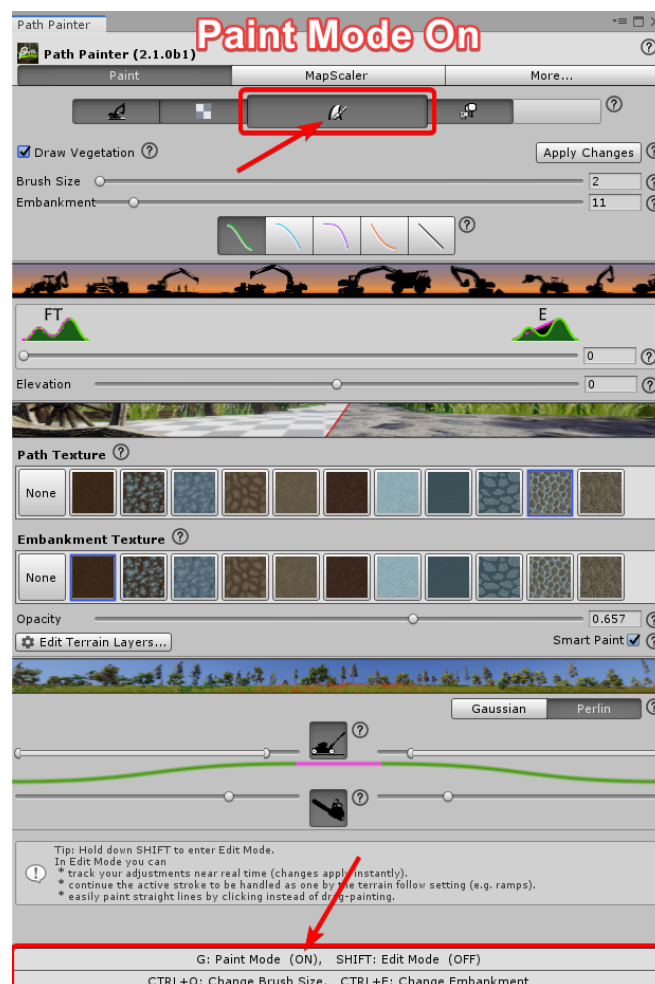
The first path

In this example we are going to be working on the *Demo Terrain* found in the *Demo* folder of the package (...3D Haven / Path Painter / Editor / Demo / **Demo Scene**).

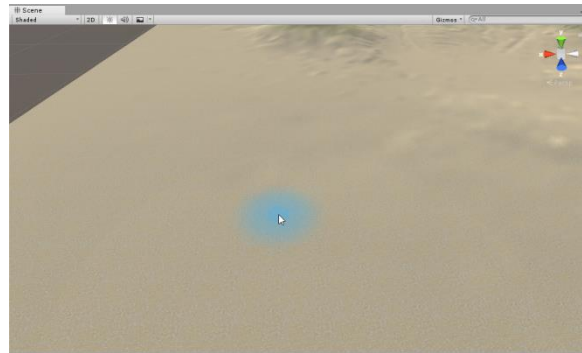
After opening Path Painter, activate **Paint Mode** by pushing the button (or press G).



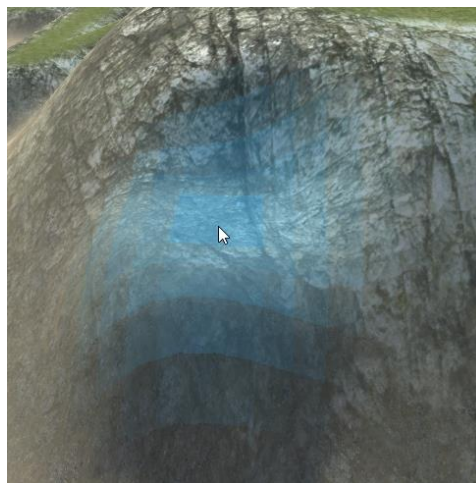
Both the button and the [Status Bar](#) will update to show that **Paint Mode** is active.



The brush will also be visible on the terrain

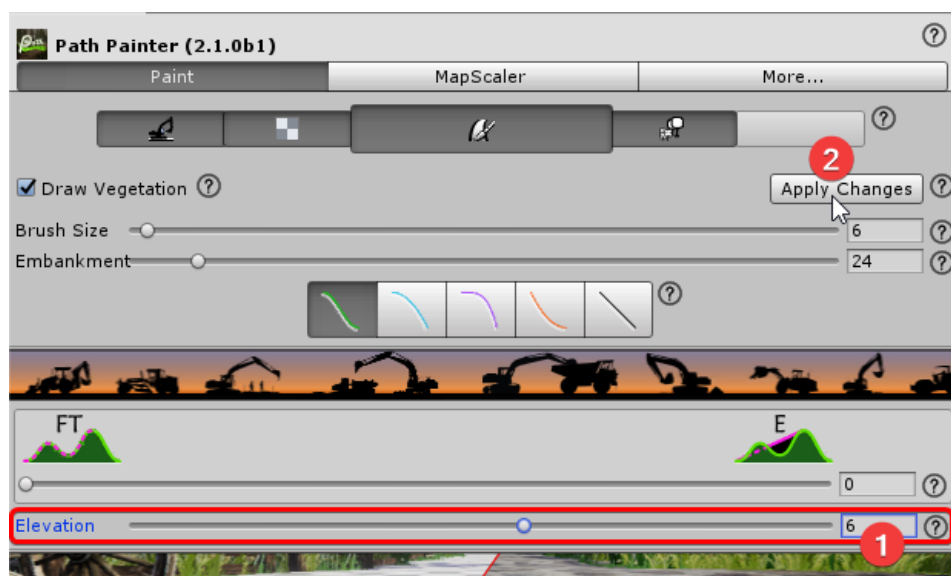


1. Now click on the terrain. - Nothing major will be visible on a flat terrain, but clicking does create a levelled flat surface and it can be used to create terrace spots.

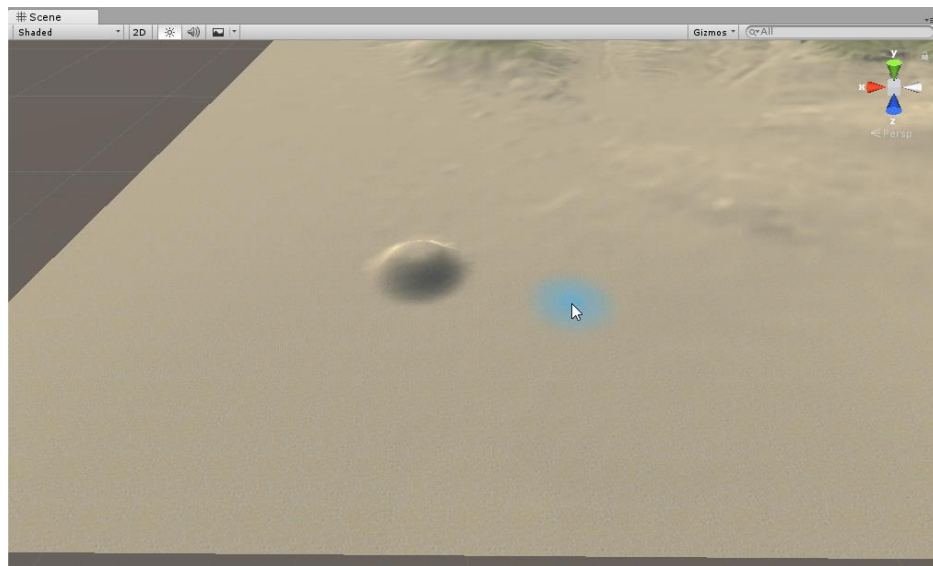


Back to our Demo terrain:

2. Set **Elevation** to **6** and press **Apply Changes**.



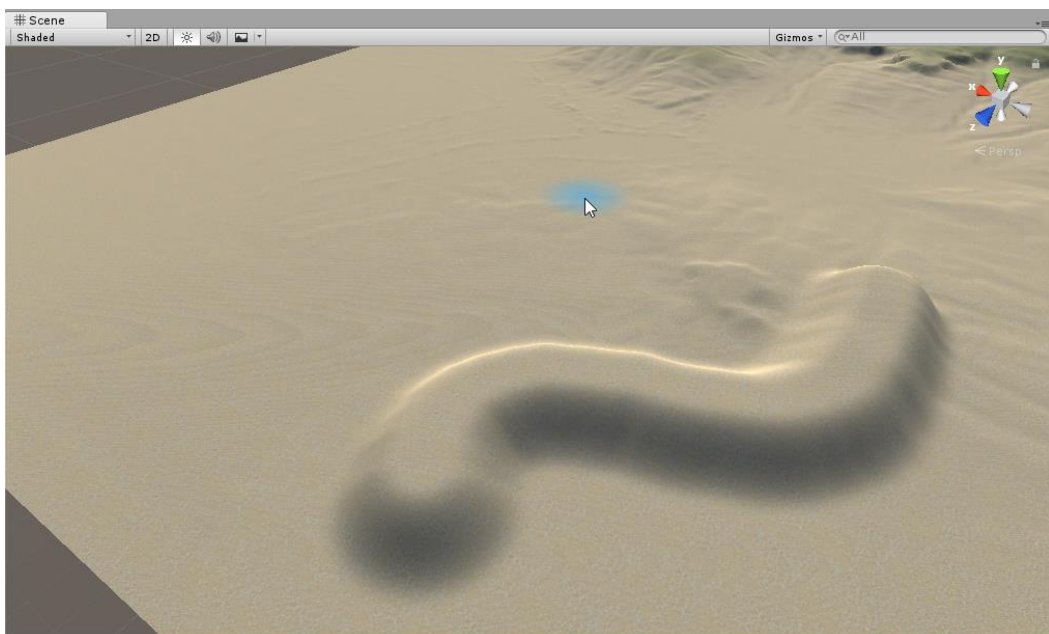
Congratulations! You created your first little hill.



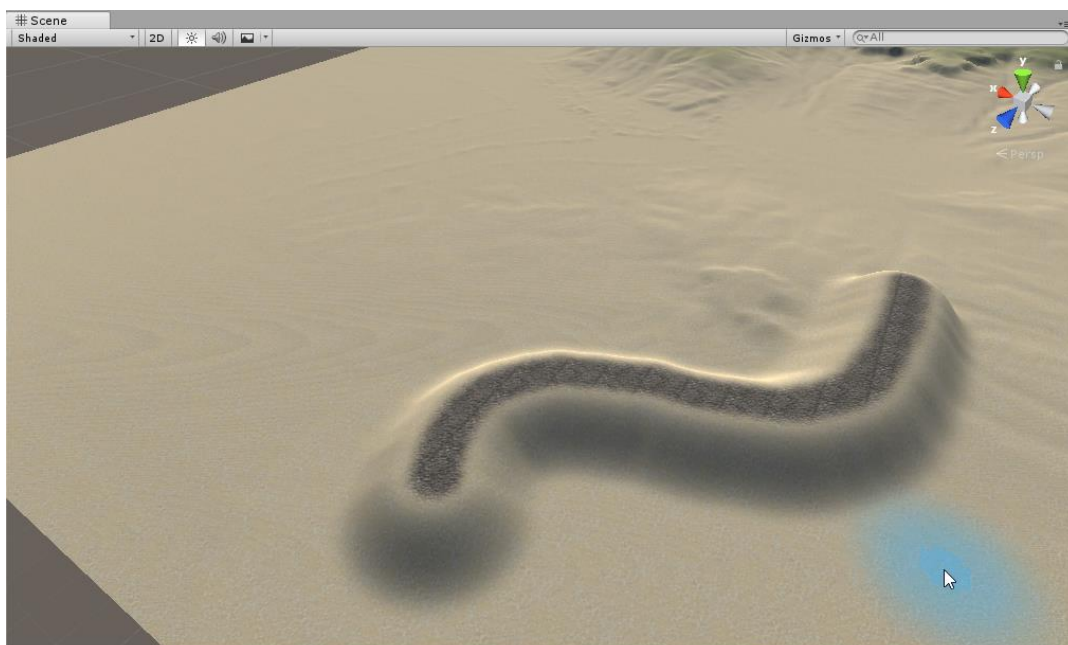
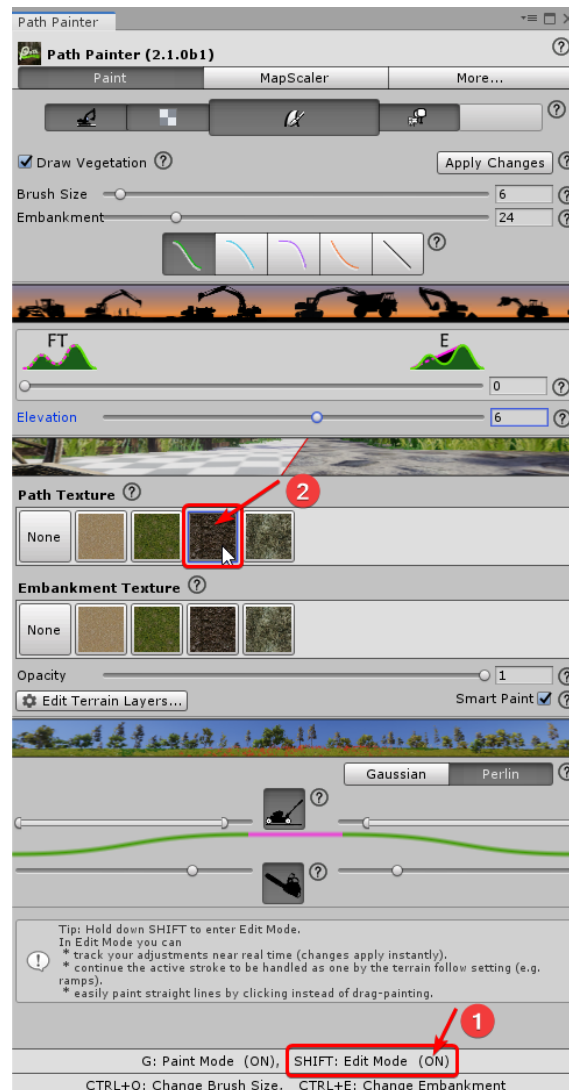
3. Let's get rid of it. **Press CTRL+Z**. You can find yourself doing this often.

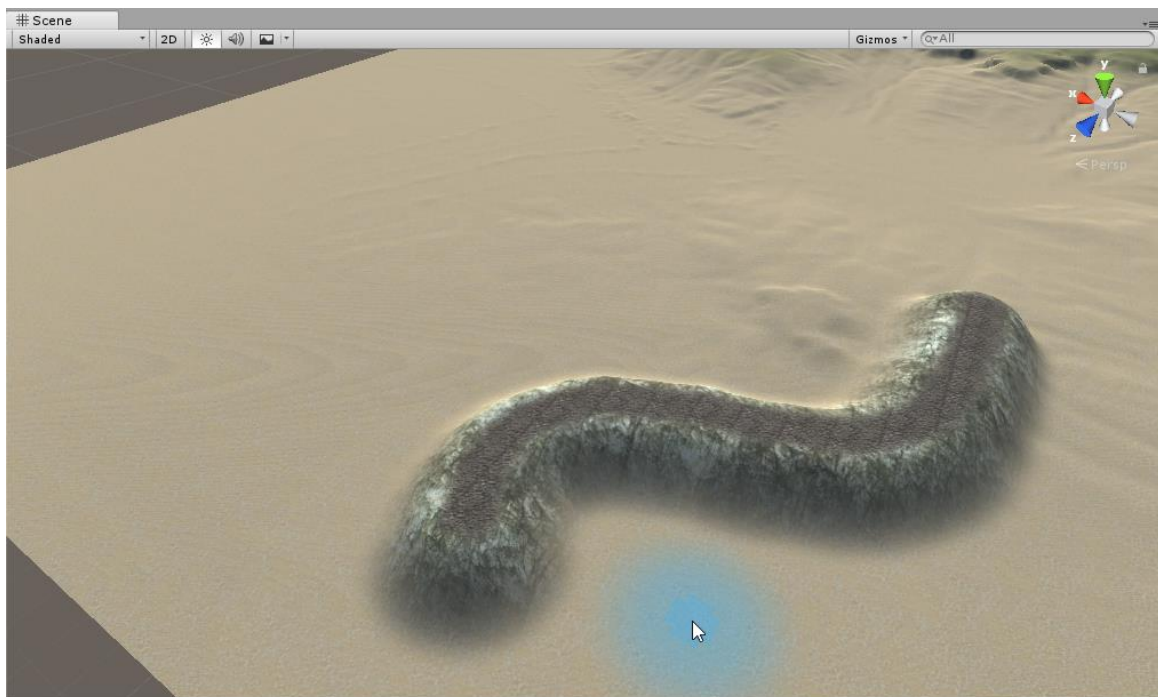
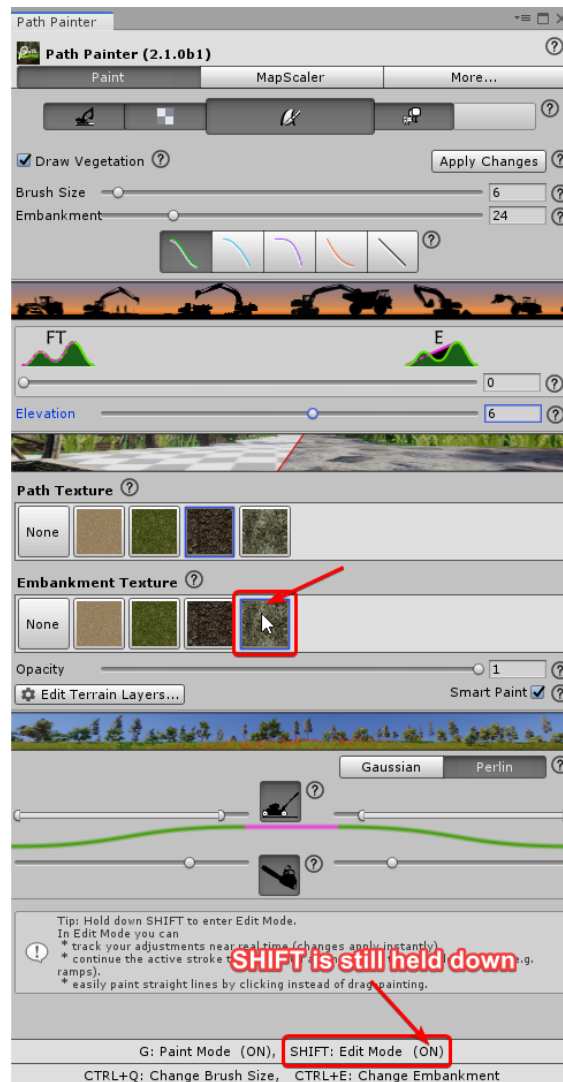
1. First painting a quick prototype to tweak your settings to what you want to do next.
2. Removing the prototype via Undo
3. and painting the paths you had in mind.

4. Now paint a line.



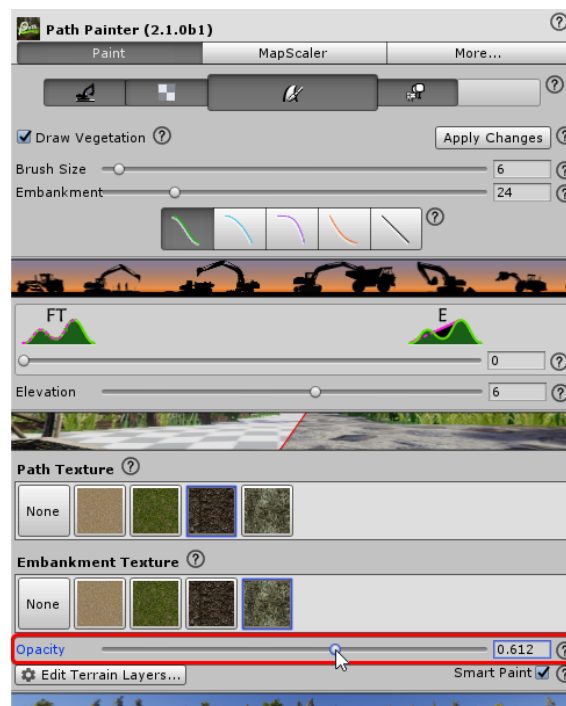
5. Let's add textures and use **Edit Mode** this time (hold down **SHIFT**) to avoid getting repetitive strain injuries from clicking the **Apply Changes** button all the time.



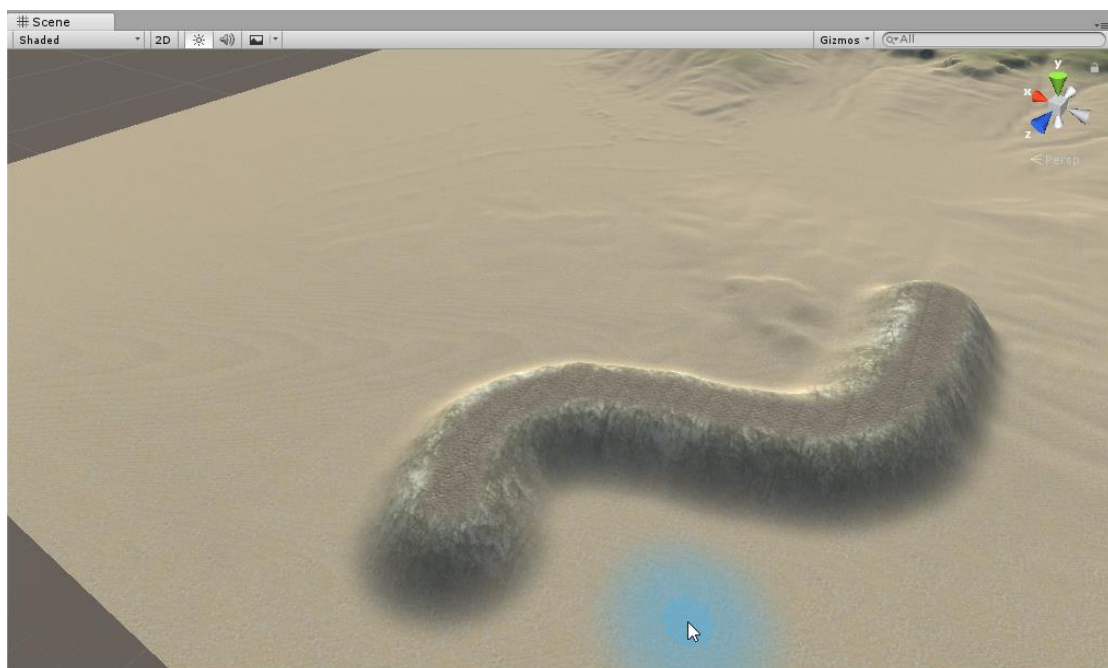
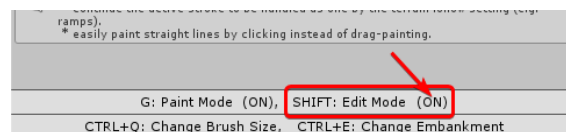


Well done.

6. Play with the **Texture Strength** slider to see what it does. Do this in **Edit Mode** to get live feedback.

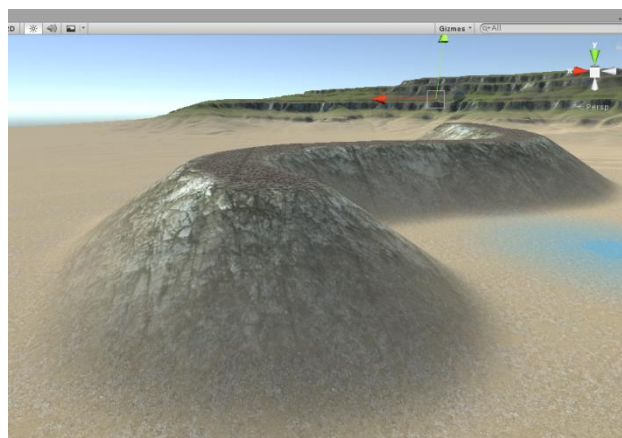
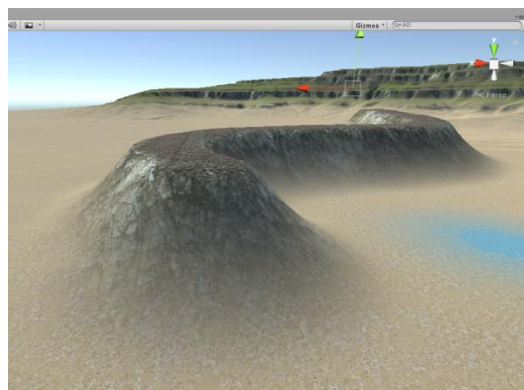
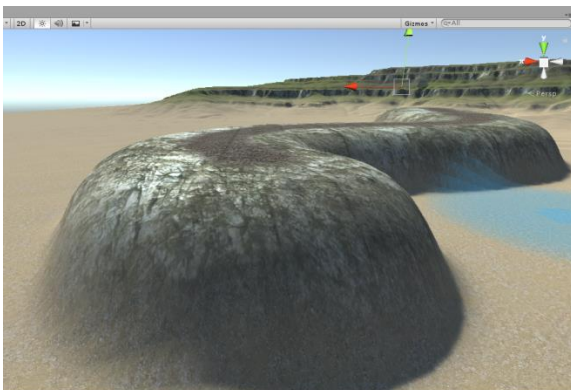
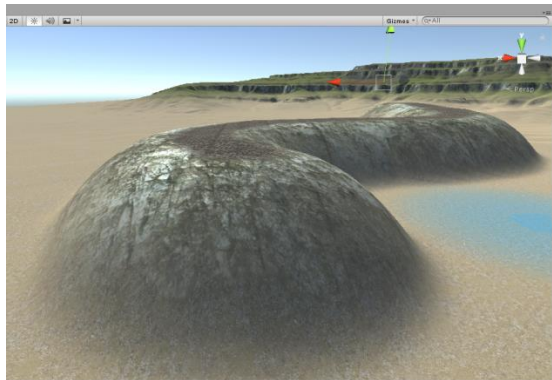
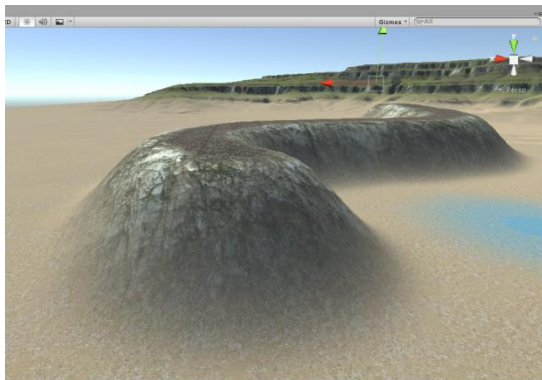
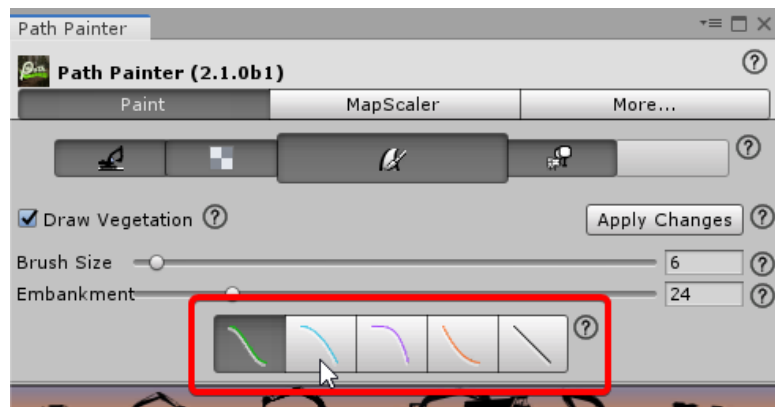


Hold down *SHIFT* to enter **Edit Mode** - you will see the **Status Bar** update



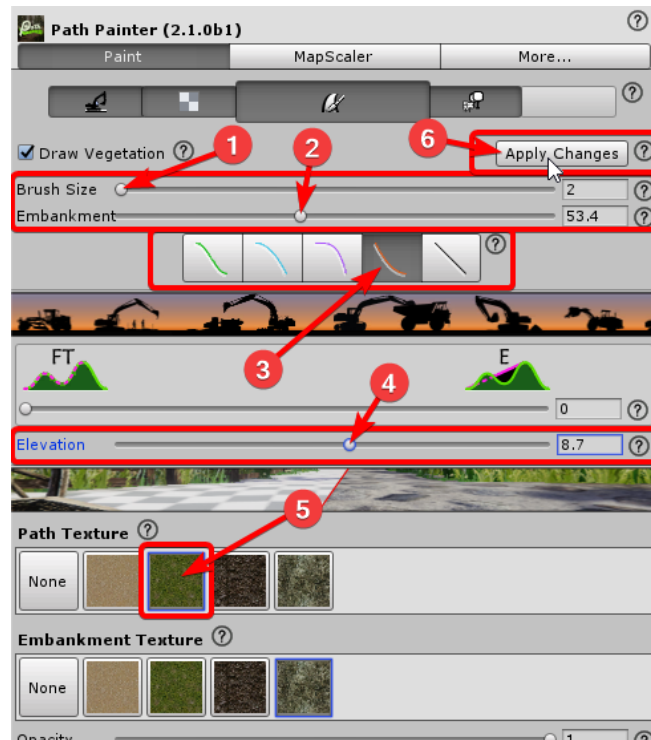
Set it back to 1 when you are done with it.

6. Play with **Embankment Curves** to see what they do. (Do this in **Edit Mode**).

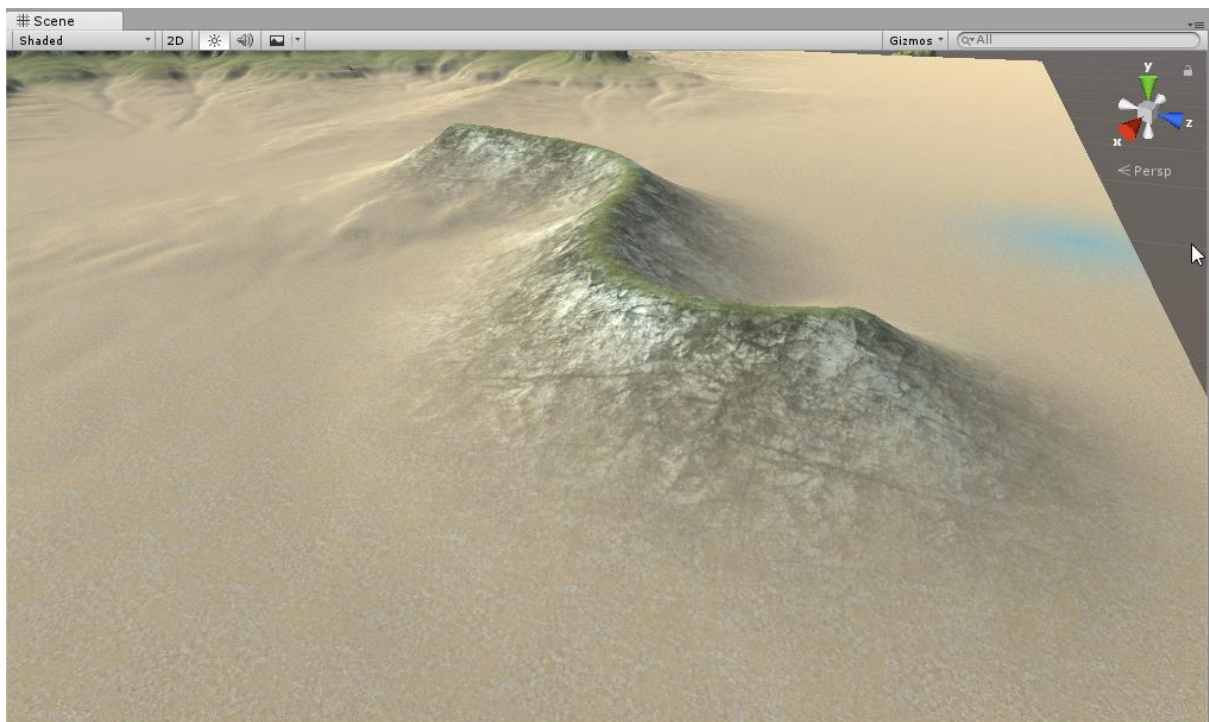


7. Change a few settings and see what we get. This time we will do the changes and apply them all at once (note: SHIFT not held done). Set

1. **Brush Size:** 2
2. **Embankment Size:** 50 - 55
3. **Embankment Curve:** Sharp (second from the right)
4. **Elevation:** 8 - 9
5. **Surface Texture:** Grass texture
6. And click **Apply Changes**.



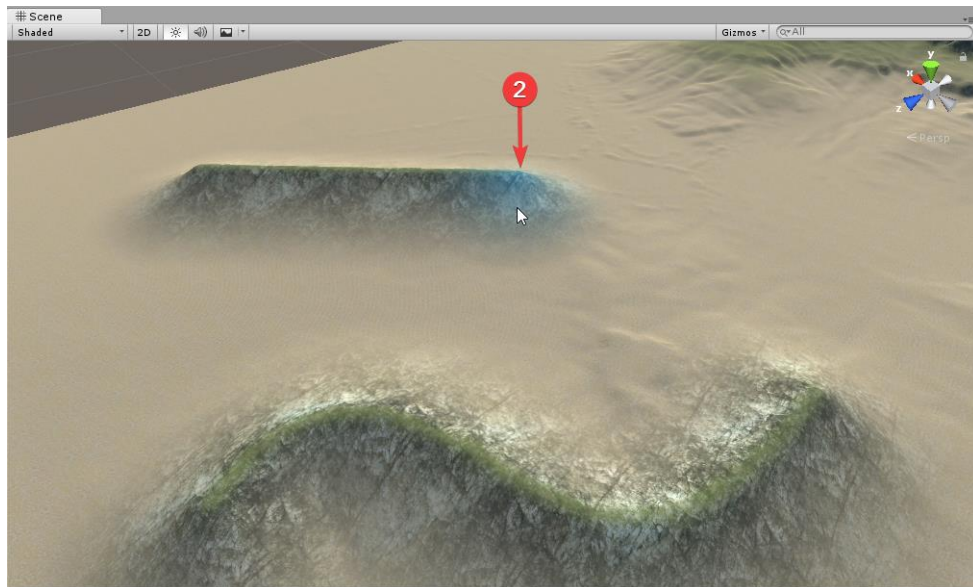
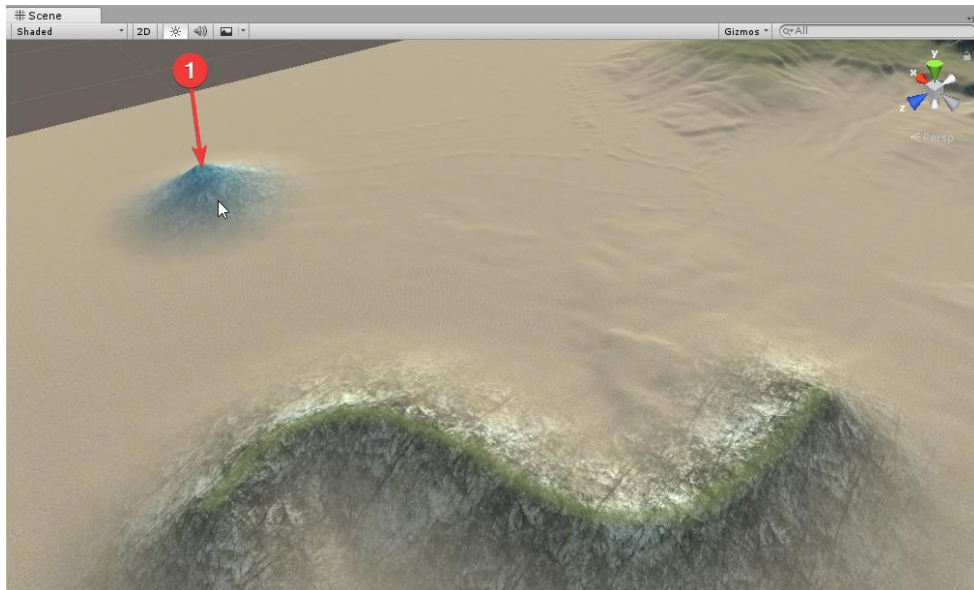
The result:



Ramps, Straight Lines, Mixed Lines

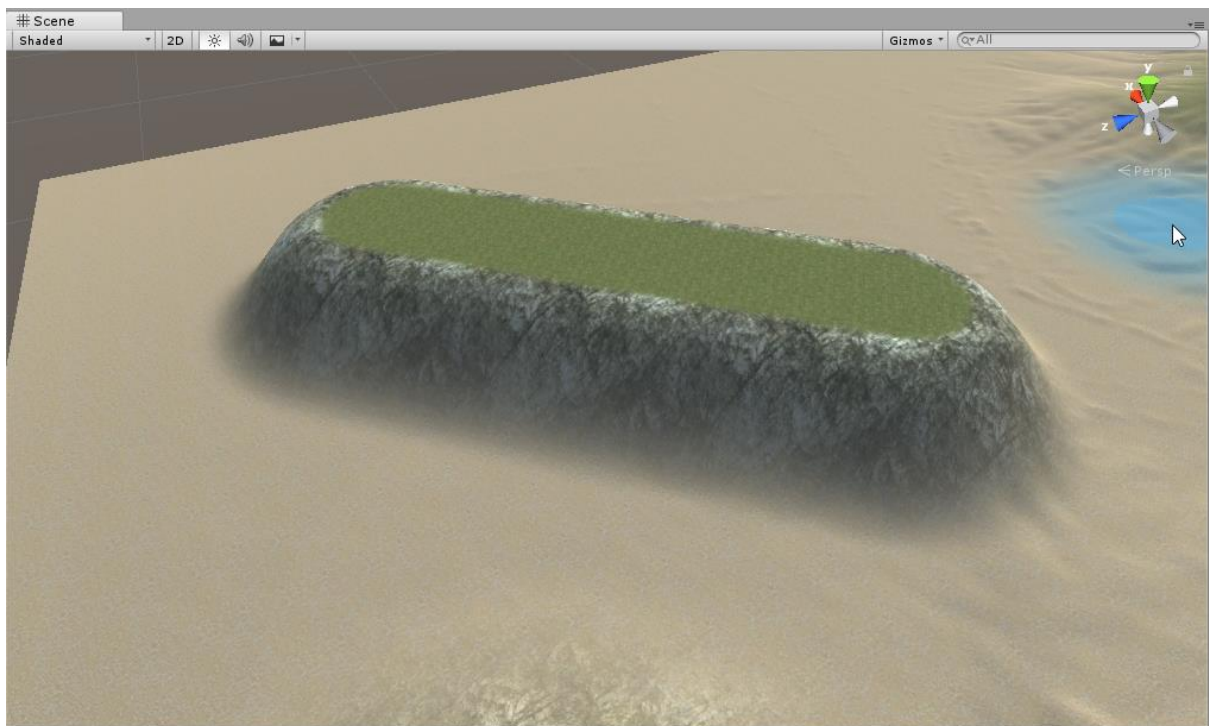
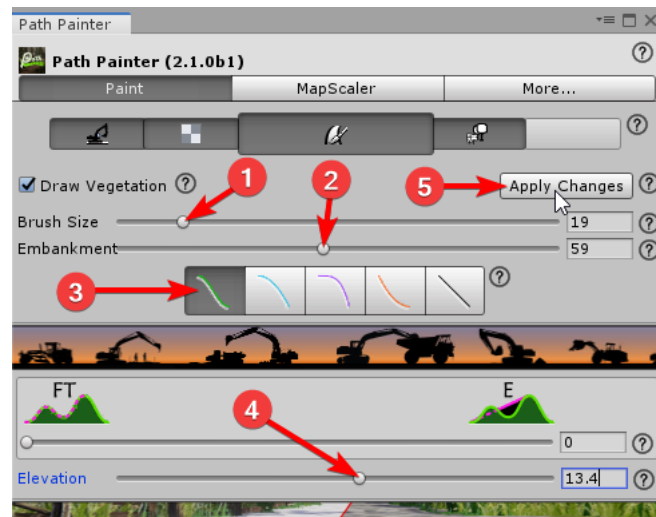
In this example we are going to be working on the *Demo Terrain* found in the *Demo* folder of the package (...[3D Haven](#) / [Path Painter](#) / [Editor](#) / [Demo](#) / **Demo Scene**).

1. Create a straight line by clicking a start and end point on the terrain in **Edit Mode** (We are adding to a point to get a straight line).



2. Update the line:

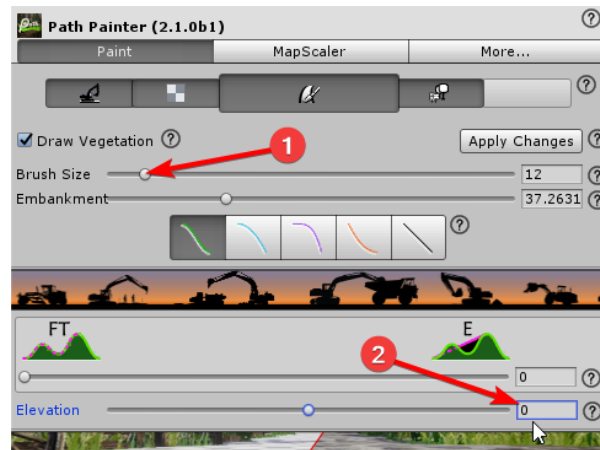
1. Brush Size: around 20
2. Embankment Size: around 60
3. Embankment Curve: Smooth (first on the left)
4. Elevation: 13-14
5. And click Apply Changes.



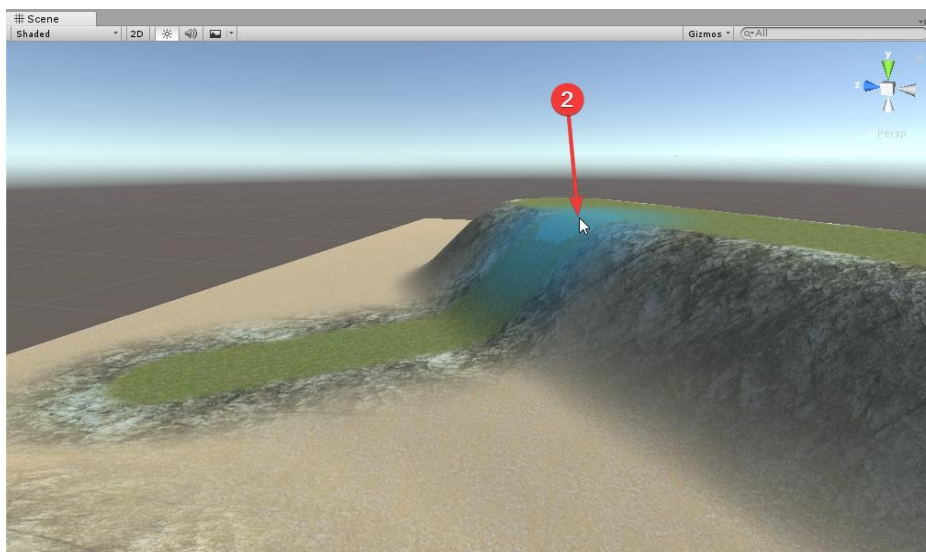
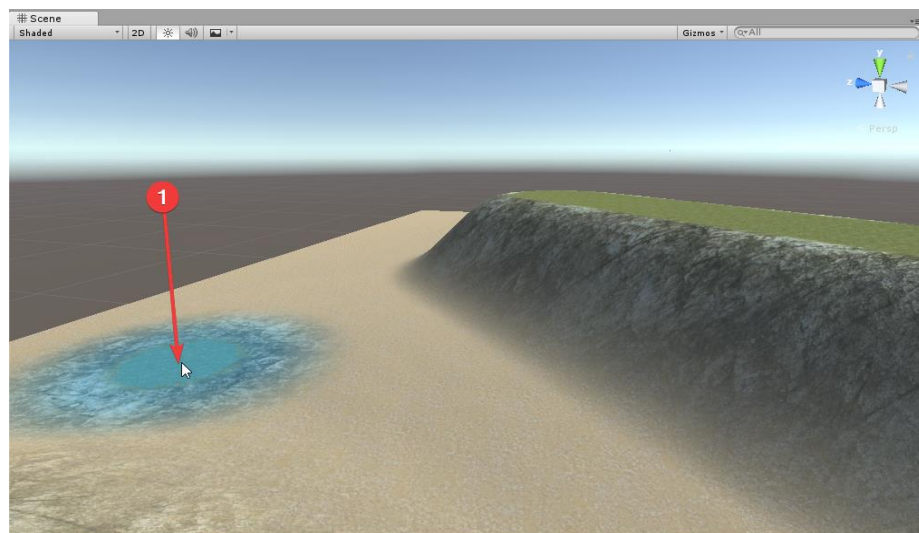
3. Creating a specific ramp - first step. Set

1. **Brush Size:** 12 (**Embankment Size** updates automatically)
2. **Elevation:** 0
3. **Slope limit:** 90 (Auto Ramp feature off)

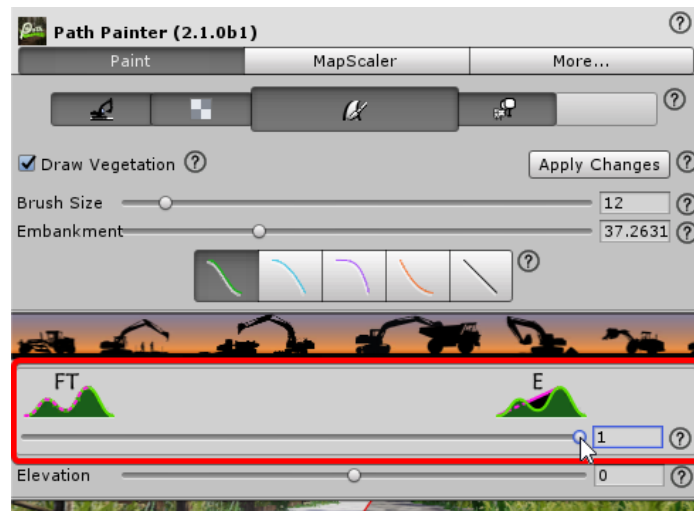
Do not **Apply Changes**. These settings are for the next path.



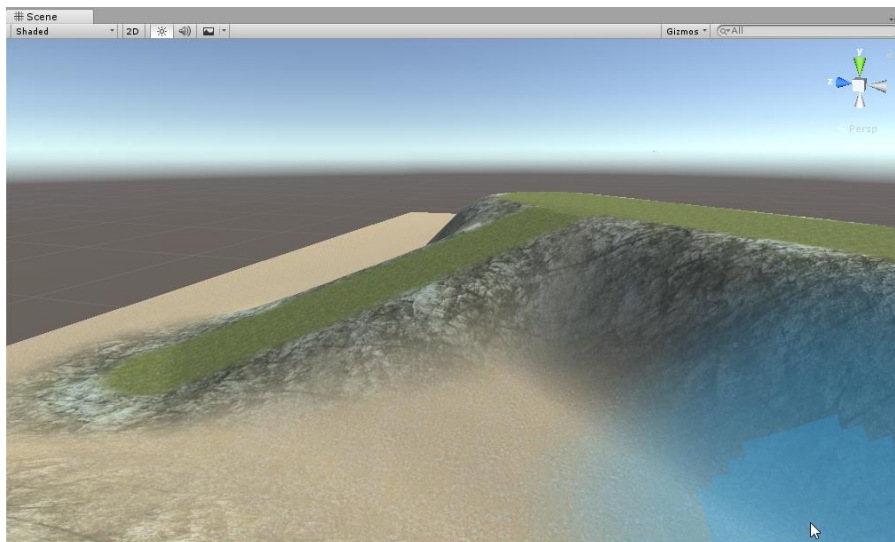
and paint a straight path up to the last one



4. Play with the **Terrain Follow** setting to see how it works and leave it at a setting you like (holding down *SHIFT* will help). In the rightmost, even slope position



this is what the path will look like:

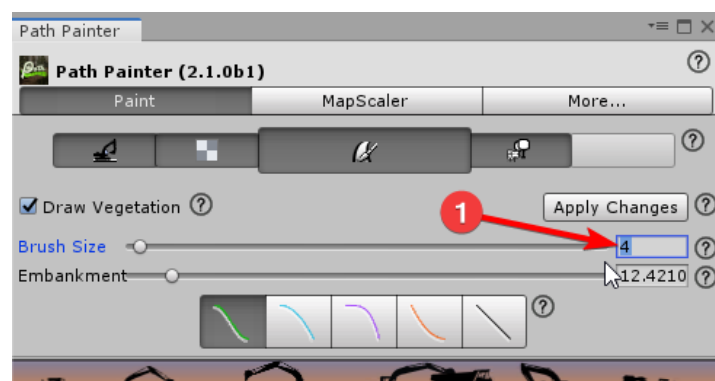


Adding to Paths

In this example we are going to be working on the *Demo Terrain* found in the *Demo* folder of the package (...[3D Haven](#) / [Path Painter](#) / [Editor](#) / [Demo](#) / **Demo Scene**).

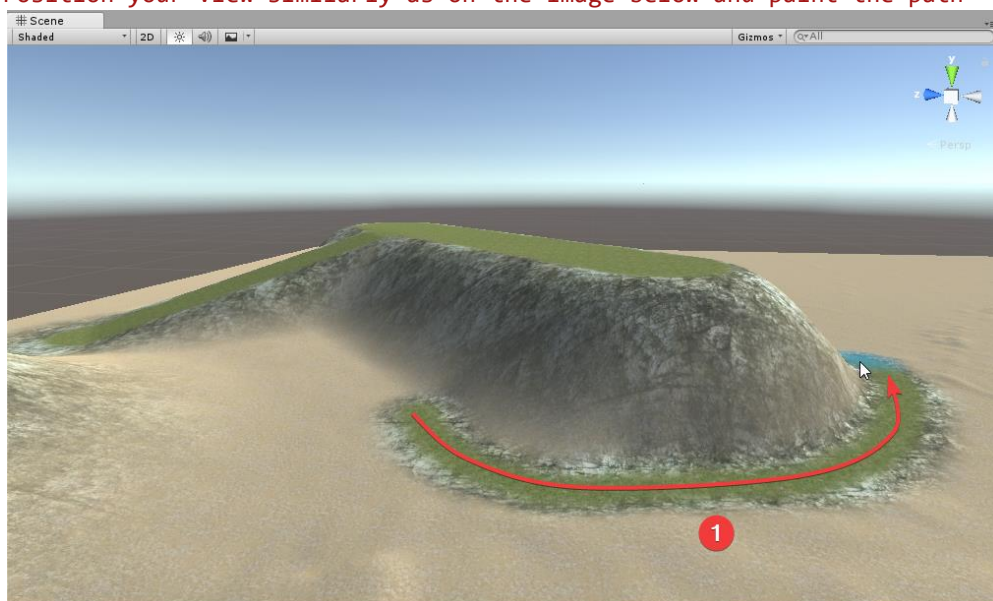
In the next steps we are going to look at when adding to a line comes especially handy. We are going to create a ramp with and without **Edit Mode** activated. In the first scenario we are going to try without using **Edit Mode**. We could fairly easily handle this example situation without **Edit Mode** by using the correct angle of view, but in real life scenarios this can be increasingly difficult.

1. Setup for thinner paths. Set **Brush Size** to 4.

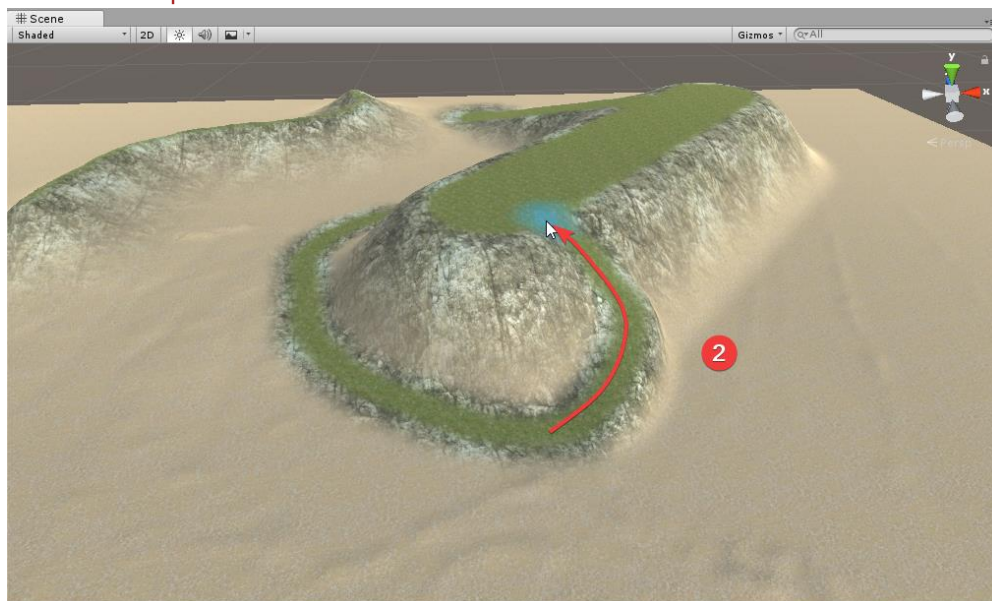


2. Paining the path in two.

1. Position your view similarly as on the image below and paint the path



2. Now you will need to rotate your camera to see the other side and paint the rest of the path.

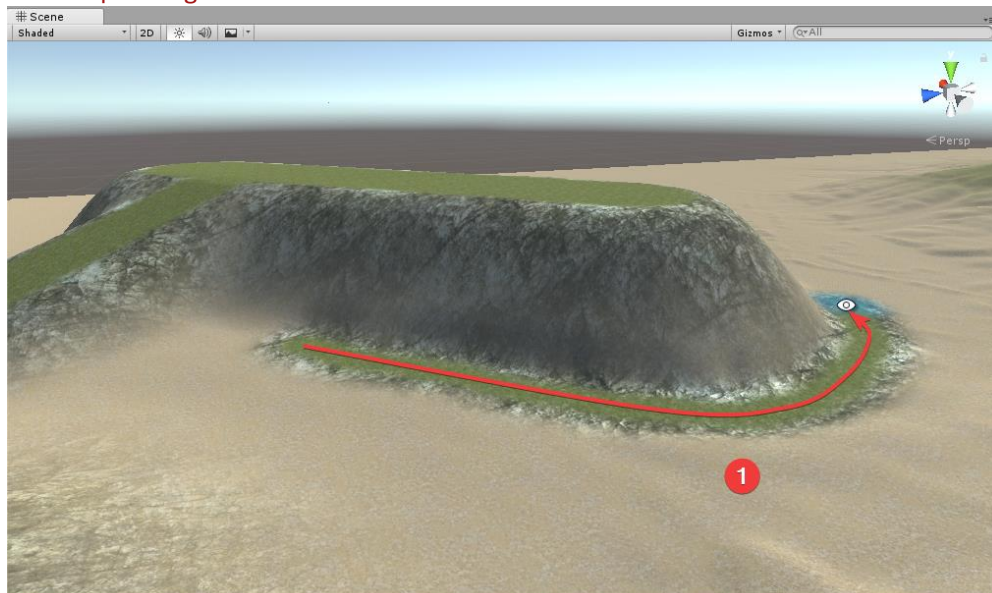


Notice where the actual ramp starts. In some cases, this is not desirable, especially if the ramp ends up to be too steep.

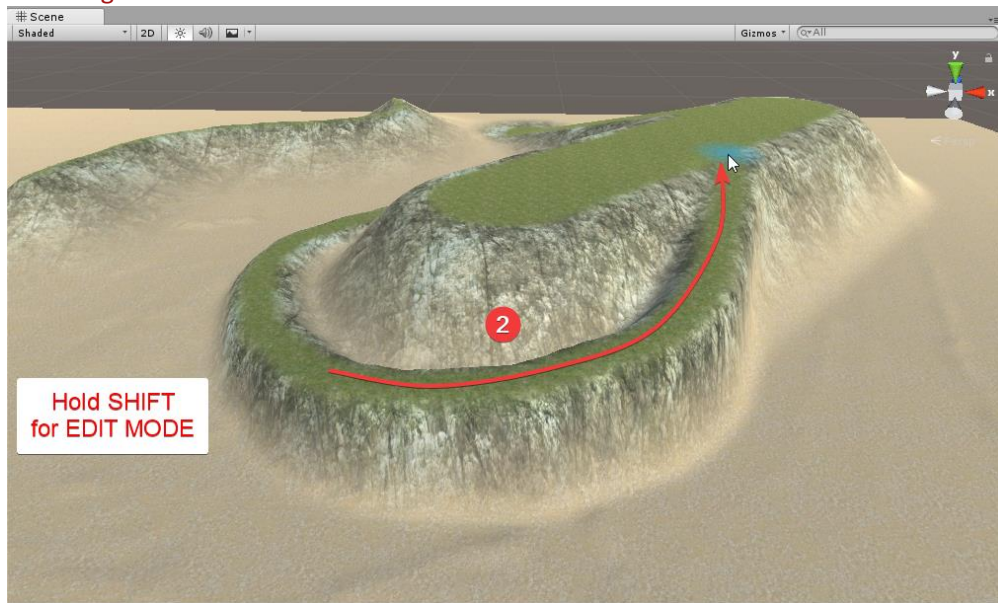
3. Use *CTRL-Z* twice to undo this before we try to do the same in Edit Mode.

2. Now let's imagine that we wanted the previously painter path to be a single ramp but the need to rotate the view made this difficult if not impossible.

1. Position your view similarly as on the image below and paint the first section of the path again.

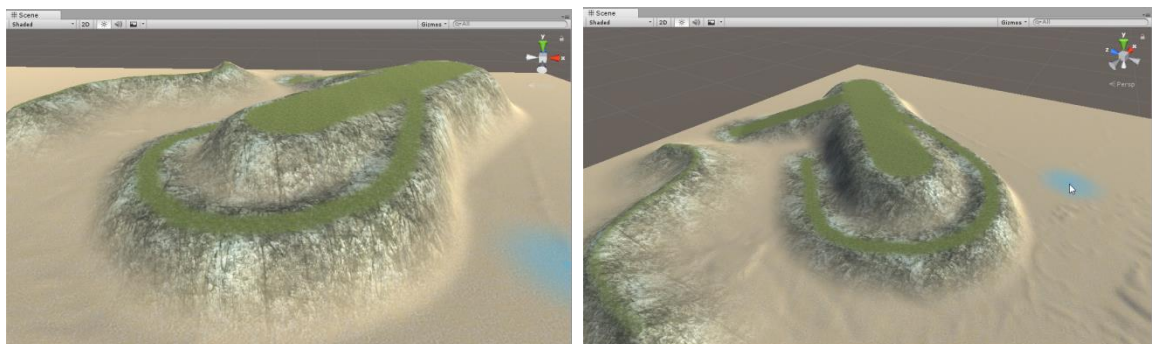
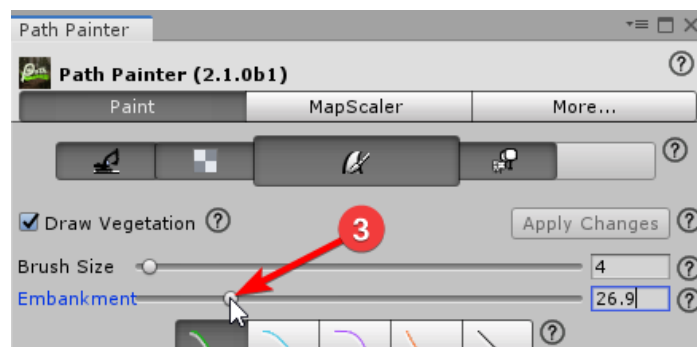


2. Rotate your camera to see the other side and paint the rest of the path, but this time hold down **SHIFT** for **EDIT MODE** to add to the first path instead of starting a new one.



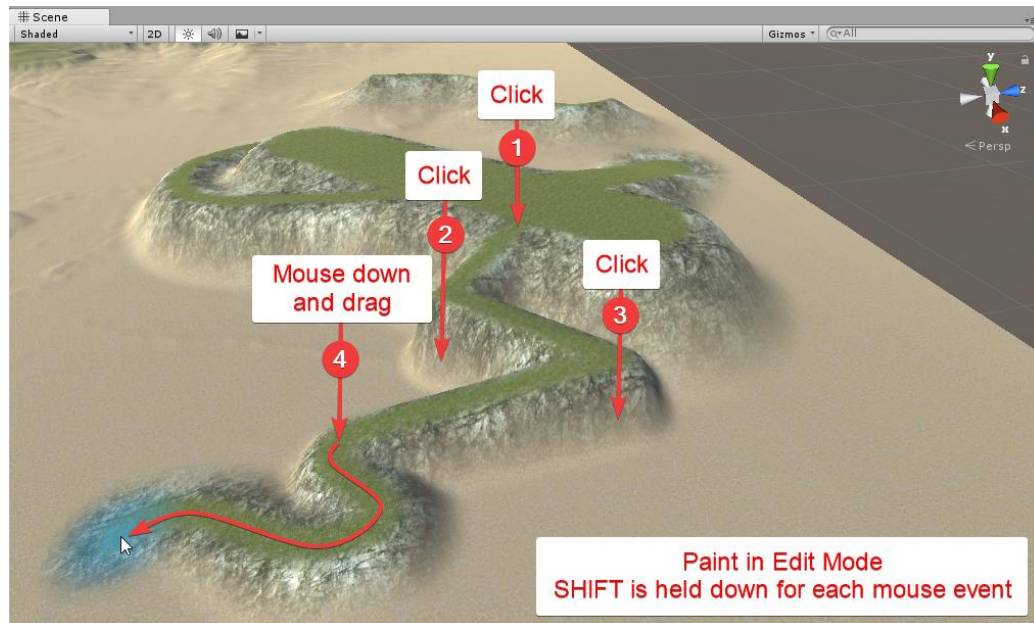
This time the whole path is a single ramp and it will be an easy walk up for characters.

3. You can increase the **Embankment Size** to make it seem like it's part of that terrain feature (still holding down **SHIFT** to see the result).

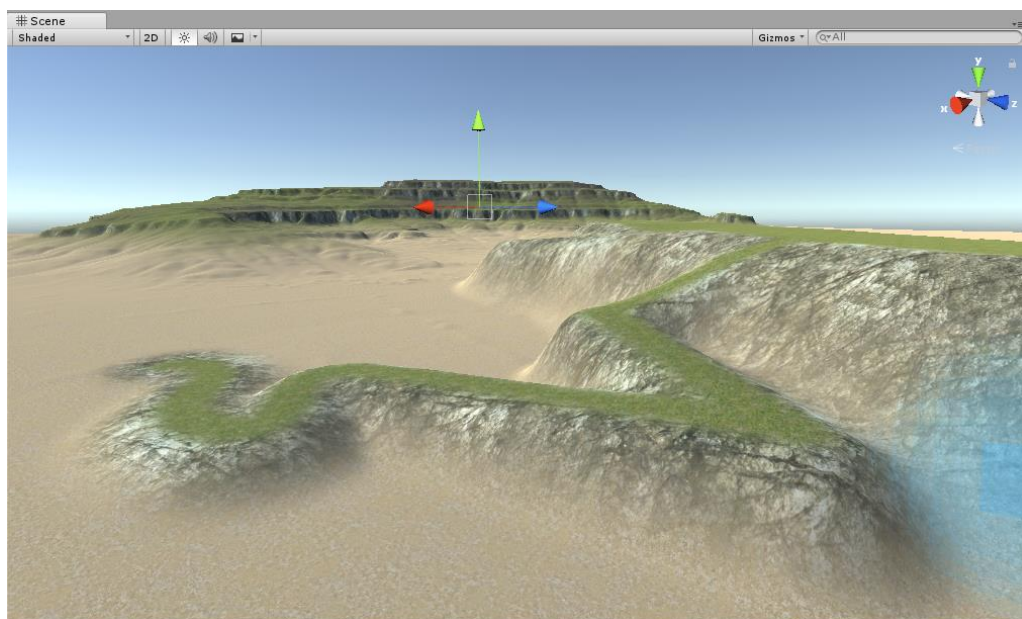
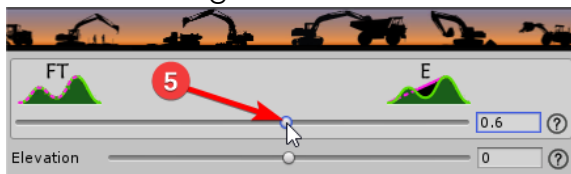


Creating a mixed (straight and curved) path.

1. Set the **Embankment Size** to 16 (don't **Apply Changes**).
2. Hold down **SHIFT** to add all the sections

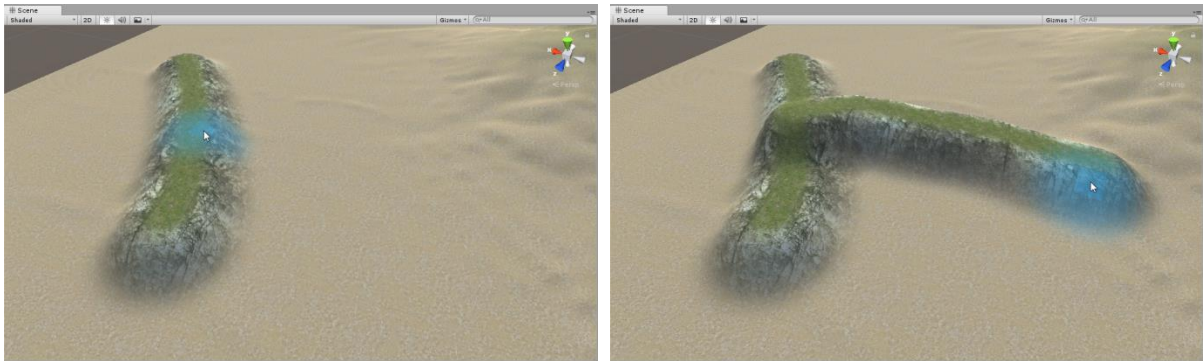


3. You can change the **Terrain Follow** setting to get a nicer look:



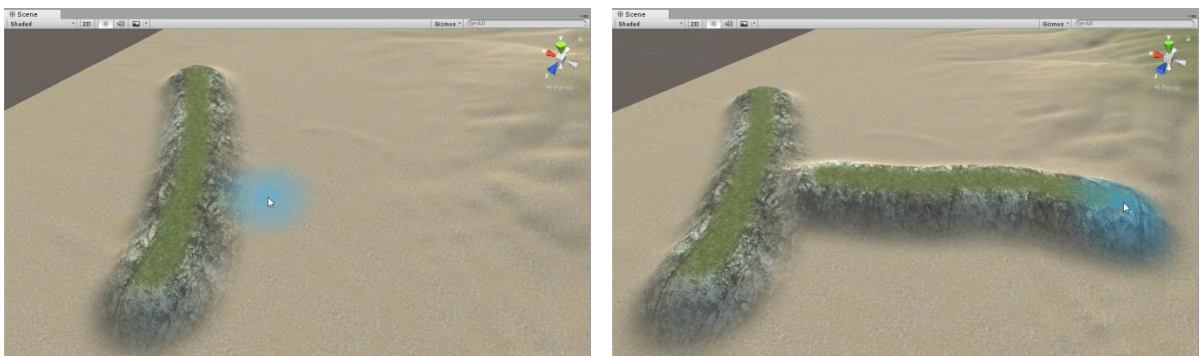
Connecting Elevated Paths, Riverbeds

Path Painter was created to paint. In this sense an existing elevated (or lowered) path/riverbed is just part of the canvas and the next path will apply its elevation according to this.



There are plans to improve this in the future. It's easy to get around this in the meantime. Even slope to the rescue! (CTRL-Z to Undo if you followed along)

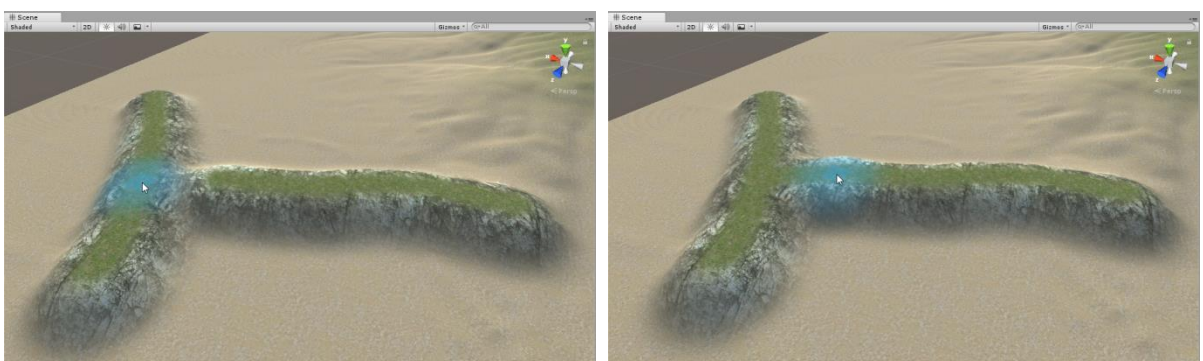
1. Create the paths so they don't cross one another.

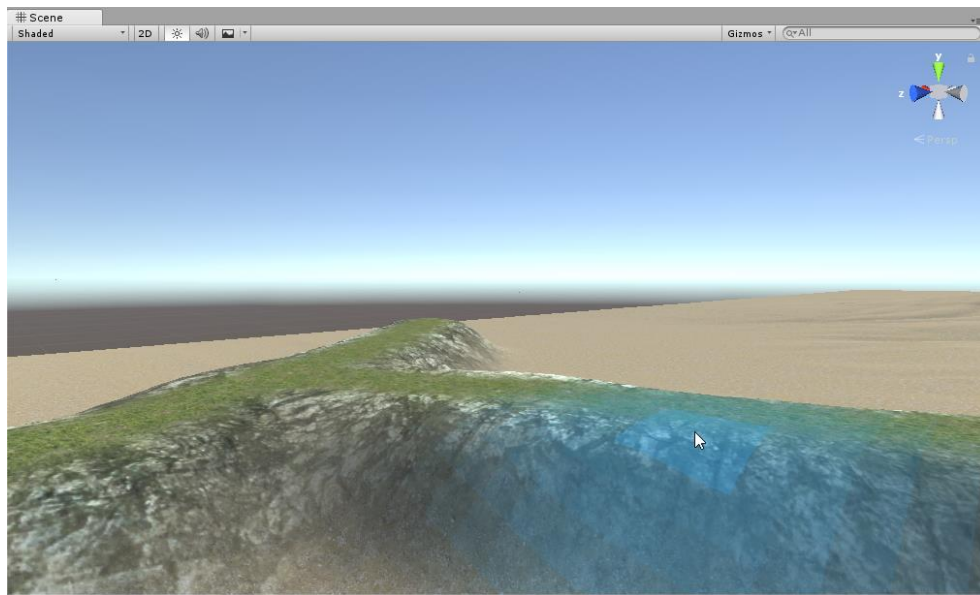


2. Set **Elevation zero** and **Even Slope** for your next step.



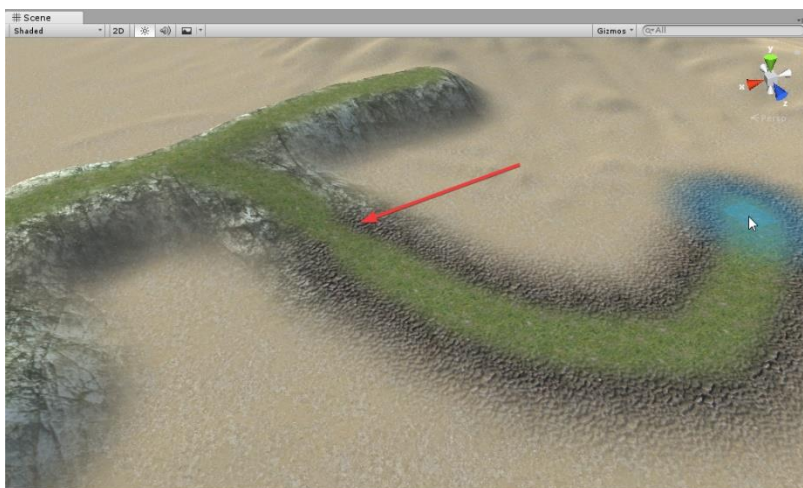
3. Connect the paths with this setting





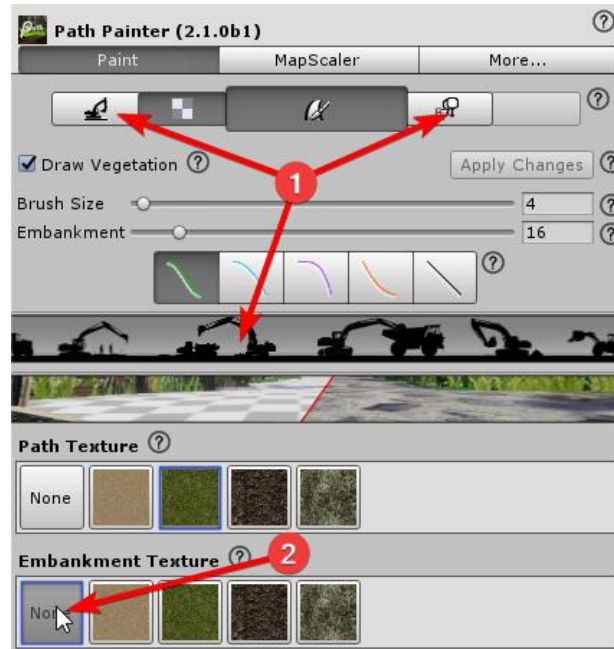
Blending Embankment Textures

When a path switch to a different embankment texture, a little texture blending can become necessary.

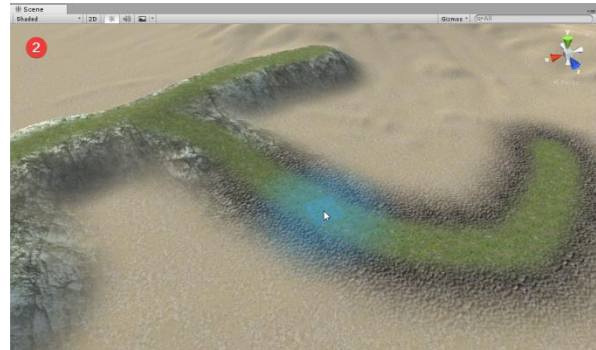
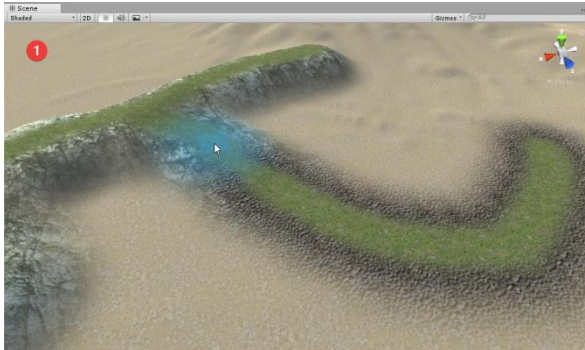


This can be easily achieved.

1. Switch **Shaping** and **Vegetation Clearing** Off by clicking on the toggles (or alternatively, you can click on their headers).
2. Set **Embankment Texture** to None.

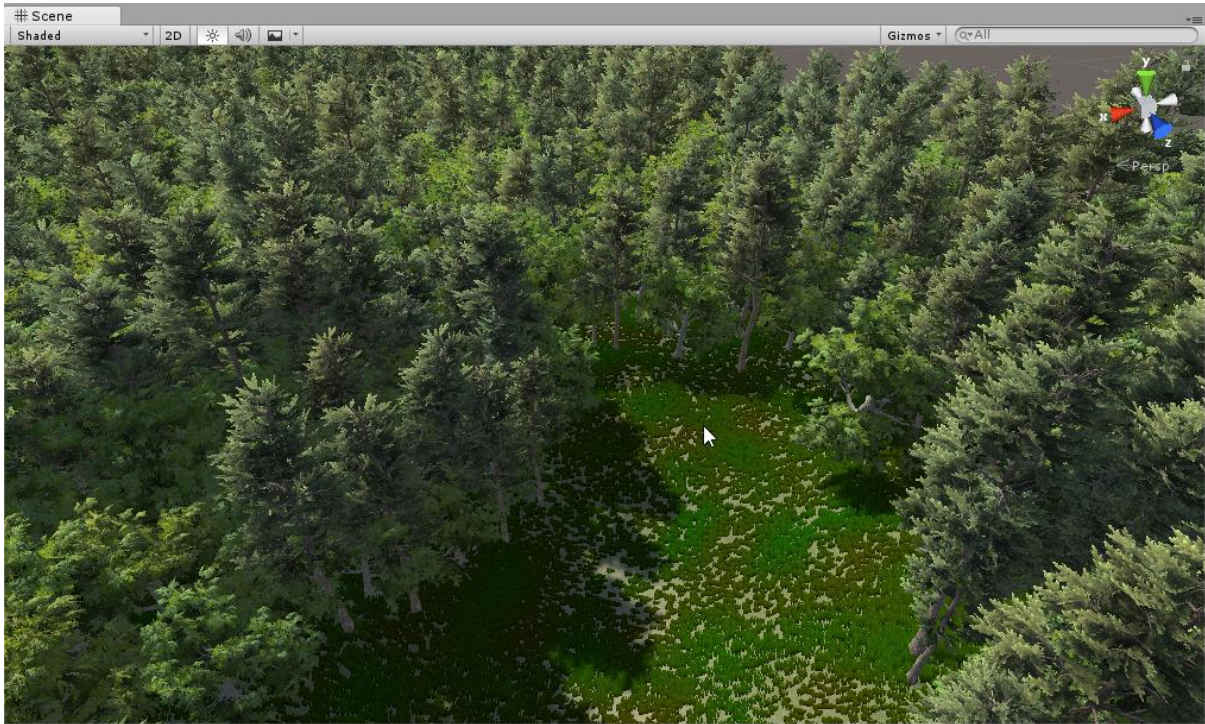


3. Paint over

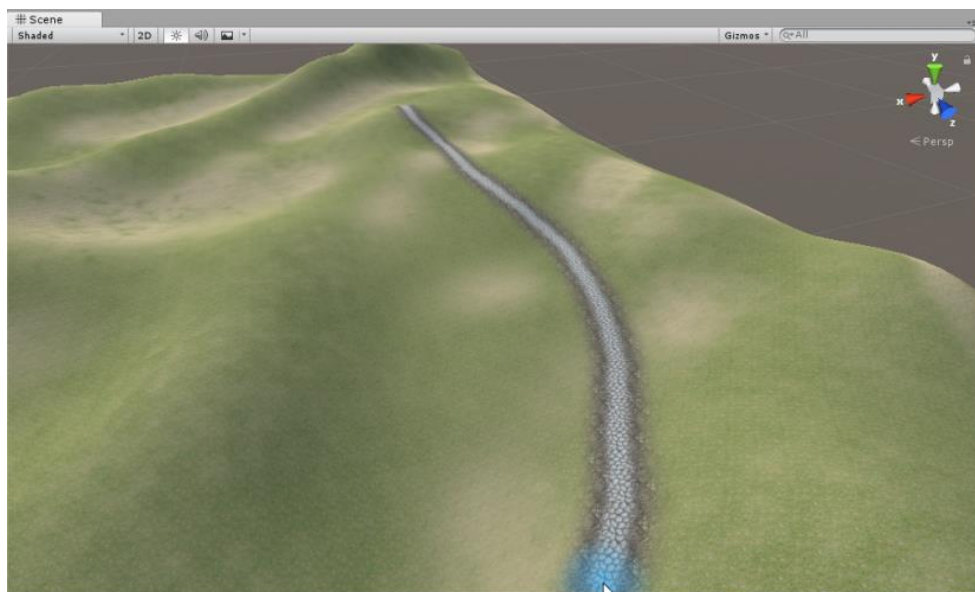


Path and Vegetation

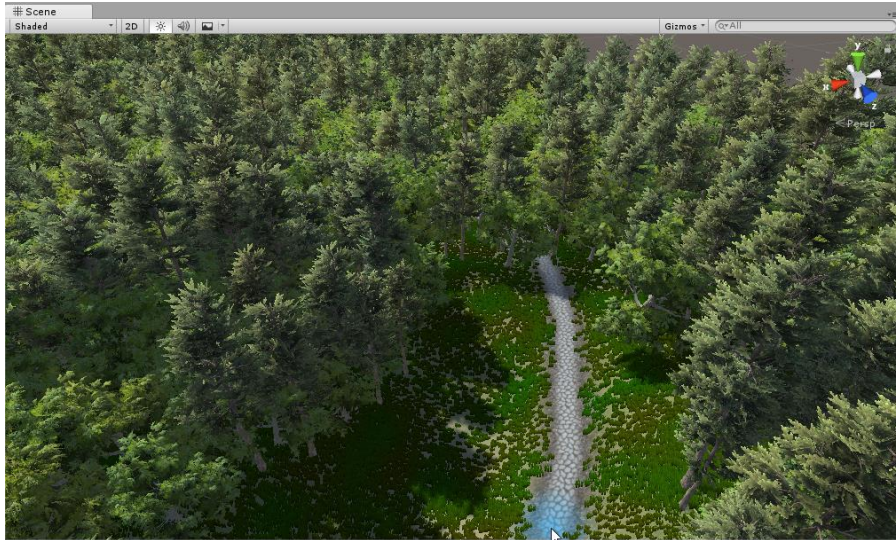
This example uses a terrain with some standard Unity terrain (details) grass and terrain trees. You can follow along if you have a similar terrain.



While painting, trees and grass are not visible to aid the painting process.

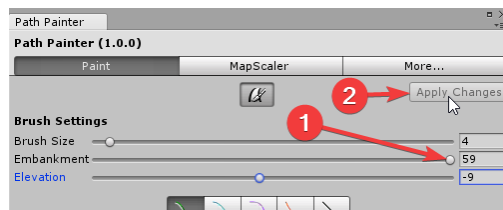


When painting is done, vegetation will be visible again (if drawing for them is enabled; see the [Vegetation](#) section).



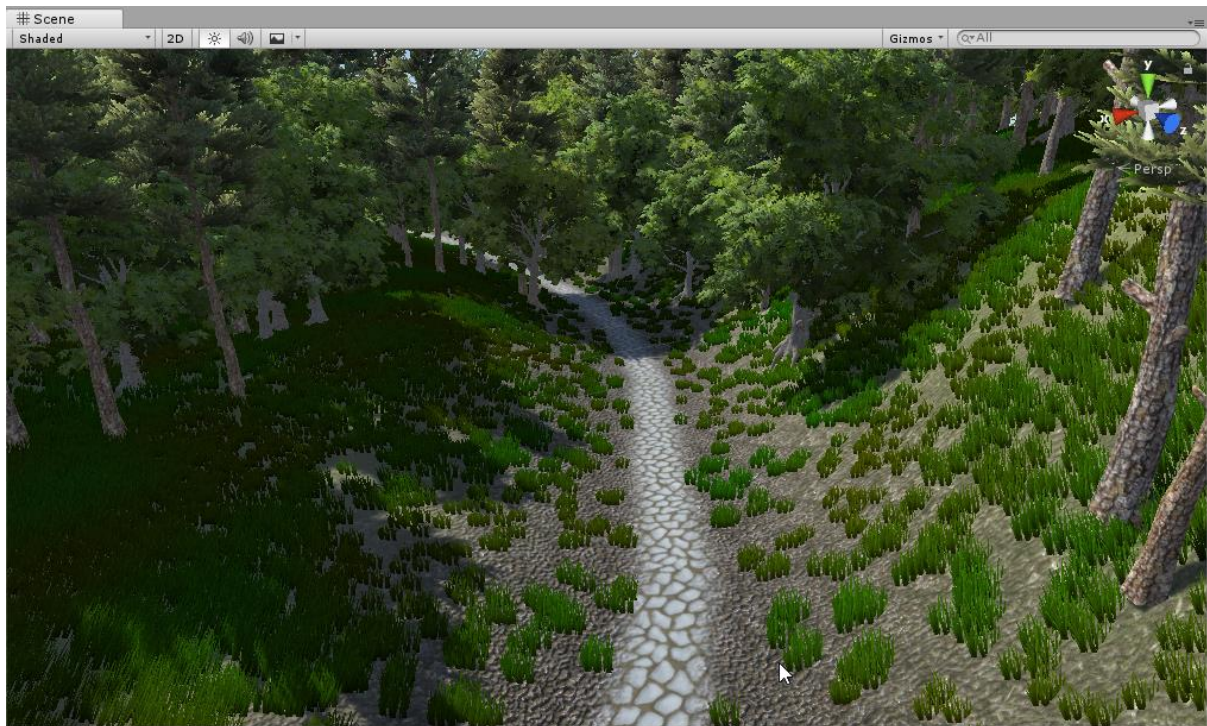
Edit Mode with **Paint Mode** activated will also hide vegetation to aid tweaking. An exception from this is when the vegetation settings themselves are being tweaked.

In this example the embankment of the path was set to 59 to better show the thinning and clearing options.

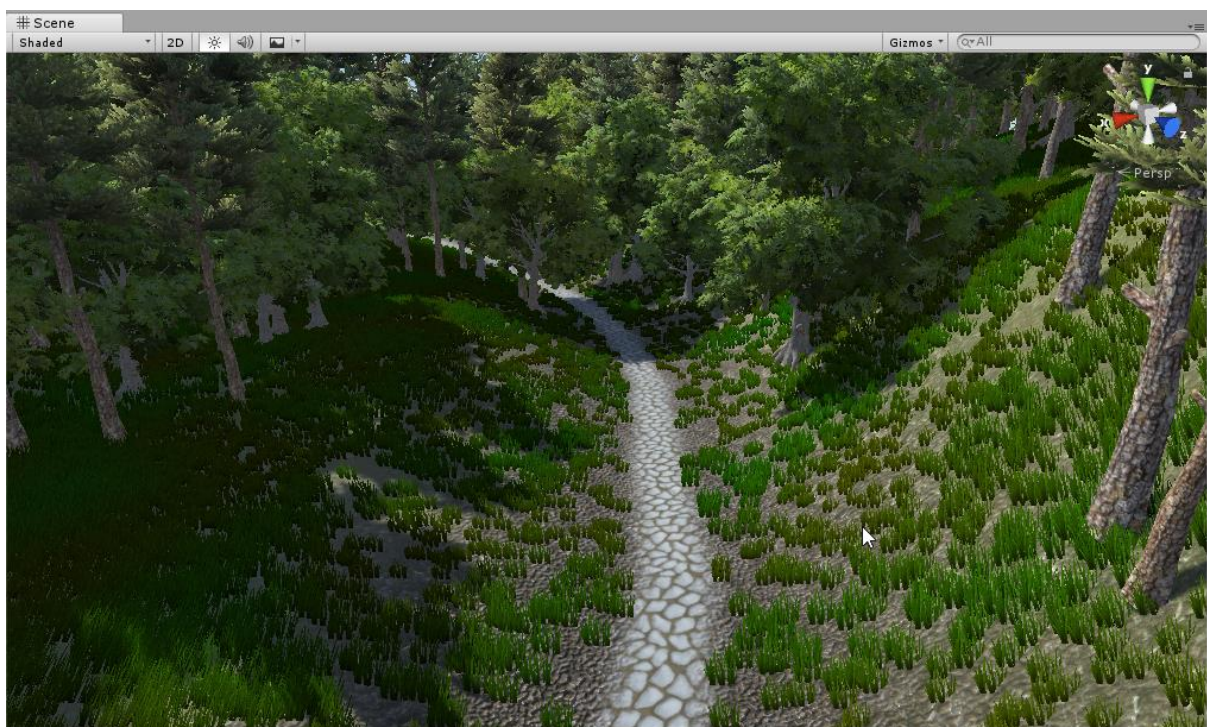
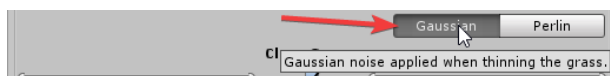


Do the grass tweaks in **Edit Mode** (**SHIFT** held down). The effect of clearing and thinning is easily visible that way.

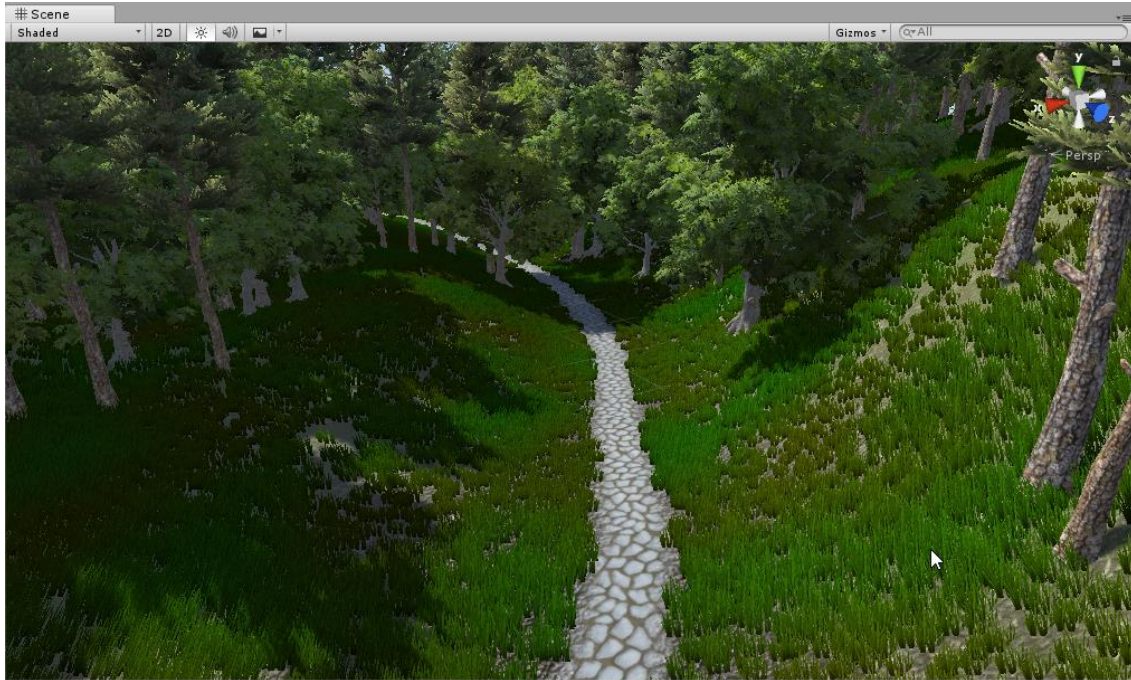
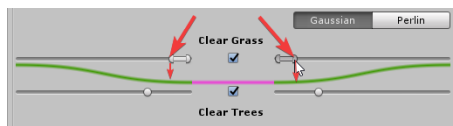
Grass thinning now have **Perlin** Noise applied



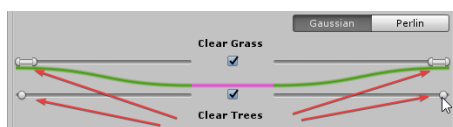
Change to **Gaussian**



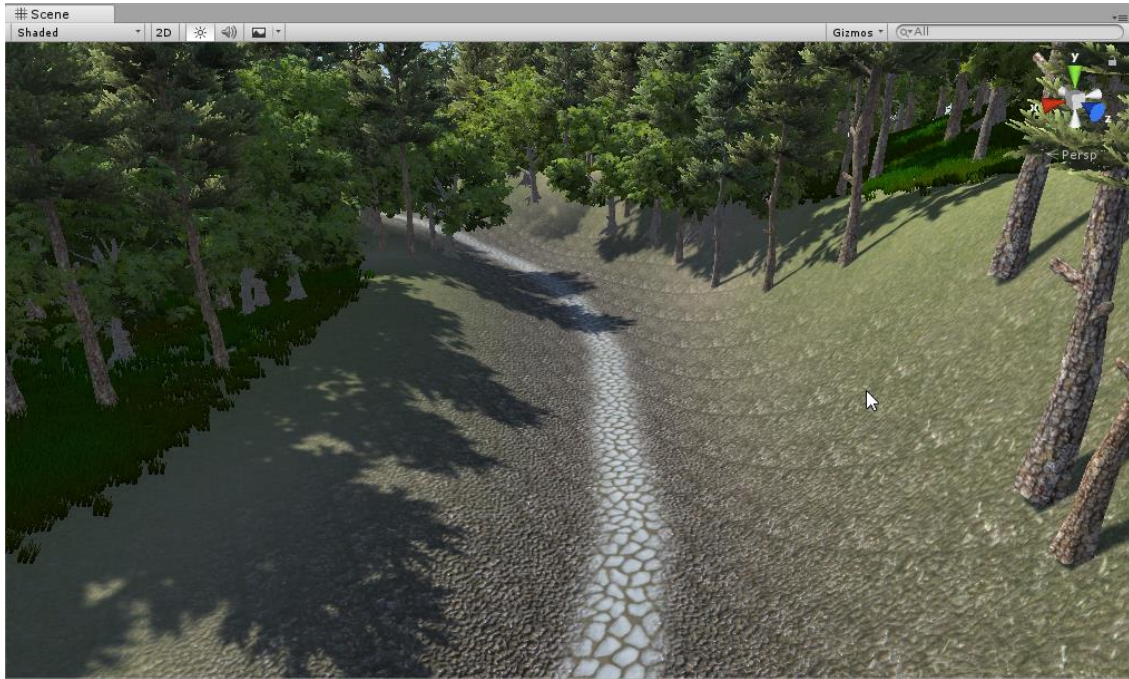
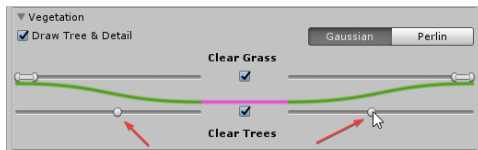
Reduce the thinning range



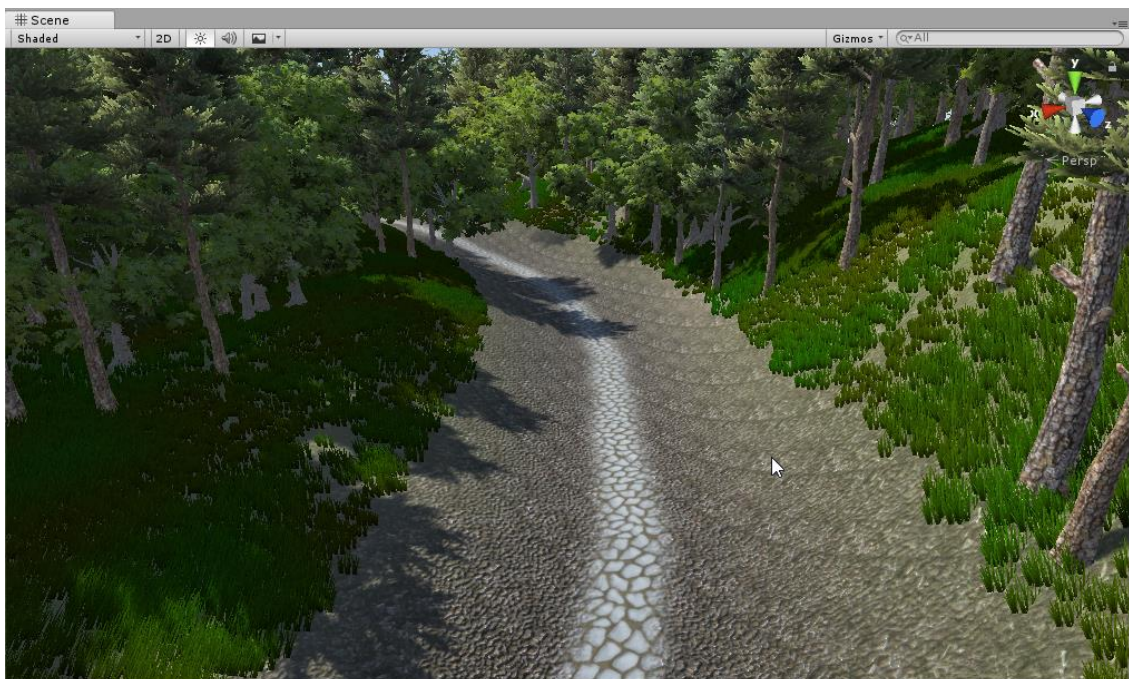
Grab and move the whole grass range to the outer edge of the embankment. Do the same with the tree slider.



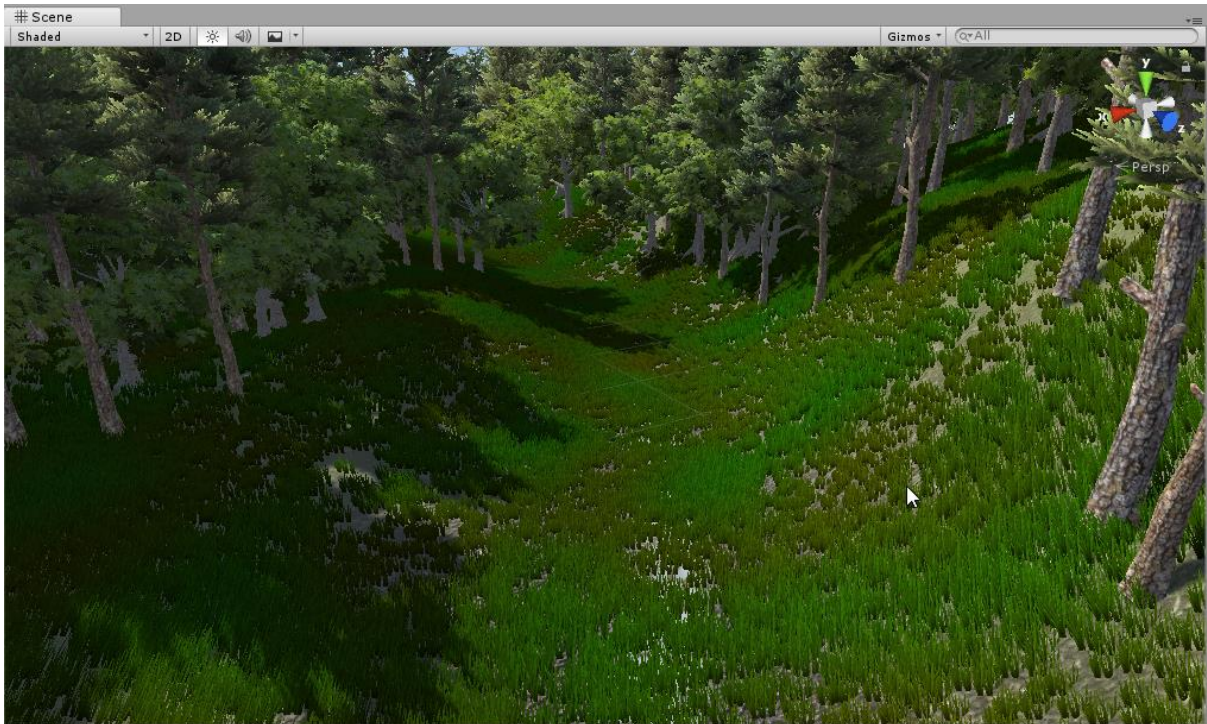
Click on the tree slider to move it back something like this.



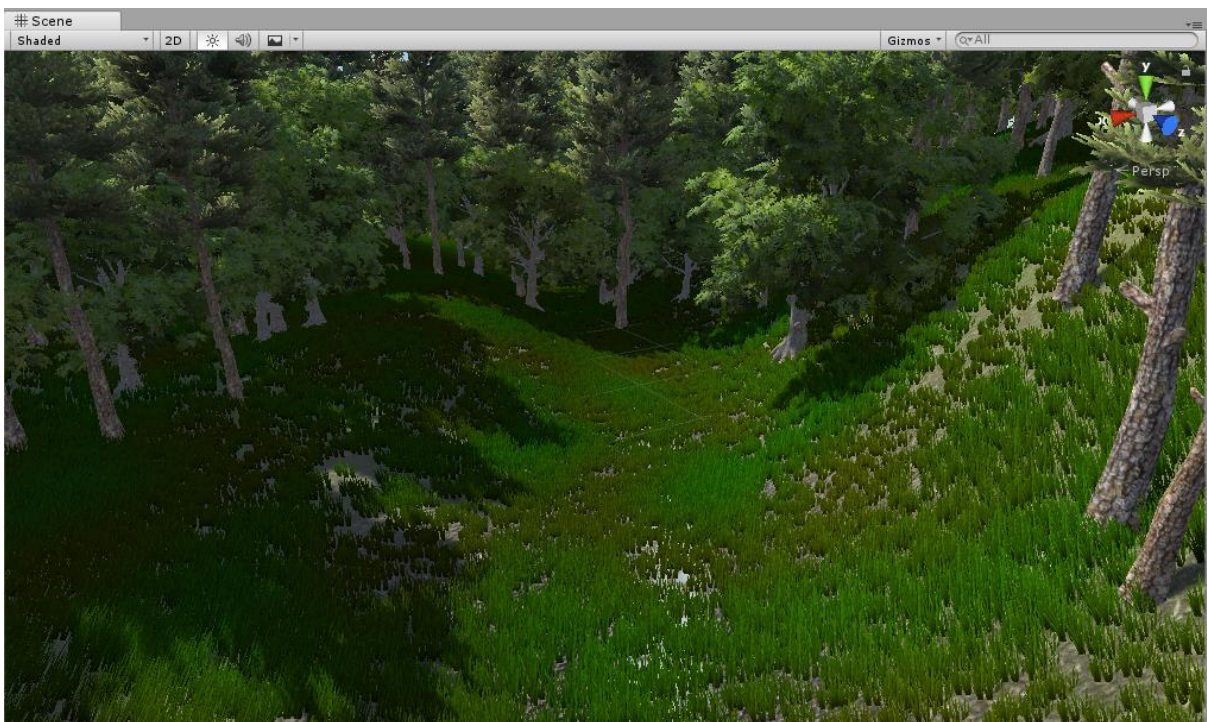
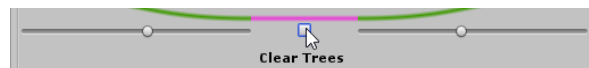
Pull the grass range back as well.



Disable Grass clearing:



And tree clearing:



This was a quick run through of the vegetation settings. It's recommended to play with them a little until it's clear what each of them does. You can also find more information in the [Vegetation](#) section.

MapScaler



Sometimes resolutions of terrains could turn out to be inadequate for the level of detail we would like to include in them. Imagine the scenario that you want to create 4-meter-wide paths on a 2 km by 2 km terrain but the *Heightmap* resolution is 513 and the *Control Texture* resolution is 512. You can't create the paths and attempts would look awful. This is where MapScaler comes handy and why it's included in Path Painter. You can scale up each map with a click of a button and you are good to go. (This was a significant feature in older Unity versions where this was not possible. Newer Unity versions allow a similar behaviour when changing map resolutions.)

Interface



The resolution of each map is displayed and they can be scaled up/down via the buttons.

Work Flow

Hit the / buttons at the map of your selection and acknowledge the action. MapScaler won't allow you to scale to resolutions that are not supported by Unity.

The resolution of each map is displayed and they can be scaled up/down via the buttons.

More Tab

The **More** tab provides quick access to more information.

Paint API Usage

You are going to be using the Path Painter API for this:

```
using Haven.API.PathPainter2;
```

The API expects a list or an array of world space points (`List<Vector3>` or `Vector3[]`) that determine points of the path to be painted (similar to how mouse movement could be used by recording a point every time the mouse moves a certain distance).

You will also need to instantiate a painter:

```
// Initialise a painter
Painter painter = new Painter();
// Initialise a painter and provide an action that records undo
Painter painter = new Painter(RecordUndoAction);
```

Signatures:

```
Painter()
```

```
Painter(System.Action<Terrain> recordUndoAction)
```

Tip: You can use different painters with different settings for different purposes. For example, you can have a script that creates rivers with a *Painter* member and another one that creates forest tracks with another one.

Paint Modes

There are two Paint Modes:

- **Paint()**: works the same way as painting with the Path Painter GUI, meaning that the height (or y positions) of the resulting path will be determined by the Auto Ramp feature(**SlopeLimit**) and **EvenRamp** (even ramp versus terrain follow) setting. Therefore, the y coordinates of the provided points are ignored.
- **Paint3D()**: is the behaviour we normally expect when using splines, meaning that the resulting path will stick to the provided points (including the y positions). The **EvenRamp** setting does not apply to this mode and it's ignored even if provided. Path elevation is relative to the points provided. Note: The Auto Ramp feature may still override your path in areas where it would be too steep. Set **SlopeLimit** to 90, if you don't want this behaviour.

There are a few options to paint using the API that apply to both modes (**Paint()** and **Paint3D()**). The only difference is that the **EvenRamp** setting is ignored by **Paint3D()** for obvious reasons. The below examples will use **Paint()** for example, but it very well could be **Paint3D()**.

Just Paint

The API stores its settings during a session. This means that a paint can be initiated without setting any options. The default settings are pretty similar to the ones in the GUI. The default settings can be restored any time by calling **painter.SetDefaults()**.

Signature:

```
void Paint(List<Vector3> points)
```

The below code will use the current settings of the *Painter* instance **painter** (instantiated earlier in [Paint API](#)) and paint the path determined by the nodes.

```
painter.Paint(points);
```

You can also change the settings in-between Paint calls. This is handy when paths are needed with very similar settings. Example:

```
painter.Paint(nodes);
painter.size = 5f;
painter.embankmentSize = 25f;
painter.Paint(nodes2);
painter.evenRamp = 0.3f;
painter.Paint(nodes3);
painter.embankCurve = Painter.EmbankmentCurve.Mound;
painter.Paint(nodes4);
```

Change All Settings and Paint

This can be used for explicit painting to ensure that all the settings are explicitly set for a call.

Signature:

```
void Paint(
    List<Vector3> points,
    float size,
    float embankmentSize,
    EmbankmentCurve embankmentCurve,
    bool shaping,
    float slopeLimit,
    float elevation,
    float evenRamp,
    float textureStrength,
#ifdef UNITY_2018_3_OR_NEWER
    TerrainLayer texture,
    TerrainLayer embankmentTexture,
#else
    SplatPrototype texture,
    SplatPrototype embankmentTexture,
#endif
    bool smartTexturePaint,
    bool clearGrass,
    float grassClearingDistance,
    float grassThinningDistance,
    Noise grassClearingNoise,
    bool clearTree,
    float treeClearingDistance,
    bool smoothPath = true
```

Example:

```
Painter.Paint(points, 6f, 24f,  
    Painter.EmbankmentCurve.Smooth, true, 25f, 0f, 0.1f,  
    1f, pathLayer, null, true,  
    true, 0.1f, 0.3f, Painter.Noise.Perlin, true, 0.2f);
```

Change Any Settings and Paint

This is handy to change only certain settings but otherwise use the current Painter settings. Use the `Haven.API.PathPainter2.Painter` class to pass in the options.

Signature:

```
void Paint(List<Vector3> points, params PaintOption[] options)
```

Bulk Painting

Bulk Painting can be used to avoid Unity's and other overheads when intending to paint several paths at once. Some projects may use more than one separate group of terrains. It only makes sense for a single bulk painting session to paint on a single compatible group of terrains, so attempting to paint on others will not have an effect. Bulk Painting can be done in-between `StartBulkPaint(Vector3 targetingPoint)` and `EndBulkPaint()`. For the `targetingPoint` you can pass in the first point to be painted (`points[0][0]`), or any point below or above the group of terrains you want to paint on. Use the painting methods in-between: `Paint()` or `Paint3D()`. Line painting doesn't make sense in a bulk painting context or vice versa. See [PPAPIExampleSpline](#) in the Runtime Demo for usage example.

Example:

```
/// <summary>
/// Paints paths from an array of point arrays provided.
/// Painting is done alternating between Paint and Paint3D.
/// Unsafe: Doesn't check for empty array, etc.
/// </summary>
void PaintfPaths(Vector3[][] paths)
{
    // Use the first point of the first path to initialise
    painter.StartBulkPaint(paths[0][0]);
    for (int i = 0; i < paths.Length; i++)
    {
        if (i % 2 == 0)
        {
            painter.Paint(paths[i]);
            continue;
        }
        painter.Paint3D(paths[i]);
    }
    painter.EndBulkPaint();
}
```


Line Painting

Line Painting can be used to paint paths/lines point by point. This is can be useful, for example, for hand painting. Start painting a new line with `NewLine()` or `NewLine3D()`. These have the same overloads and work the same way as `Paint()` and `Paint3D()` mentioned earlier, with the difference that in this mode you provide the points one by one, therefore you will only provide the first point of the line when calling `NewLine()` or `NewLine3D()`. Once you started a line, you can add points to it by calling `AddToLine(Vector3 point)`. Finally you can complete the line by calling `CompleteLine()`. You must complete lines before you can do non Line Painting with a painter instance.

It's possible to add more points to and complete a line several times over. This allows you to implement *continue line* and *two click straight line* painting. [See hand painting examples in the Runtime Demo scene.](#)

Example:

```
/// <summary>
/// Starts a new line when the user starts painting.
/// </summary>
void MouseDownOnTerrain(Vector3 point)
{
    painter.NewLine(point);
}

/// <summary>
/// Adds points to a new line while the user is painting.
/// </summary>
void MouseDraggedOnTerrain(Vector3 point)
{
    painter.AddToLine(point);
}

/// <summary>
/// Completes the line.
/// </summary>
void MouseUp()
{
    painter.CompleteLine();
}
```

Real Life API Example

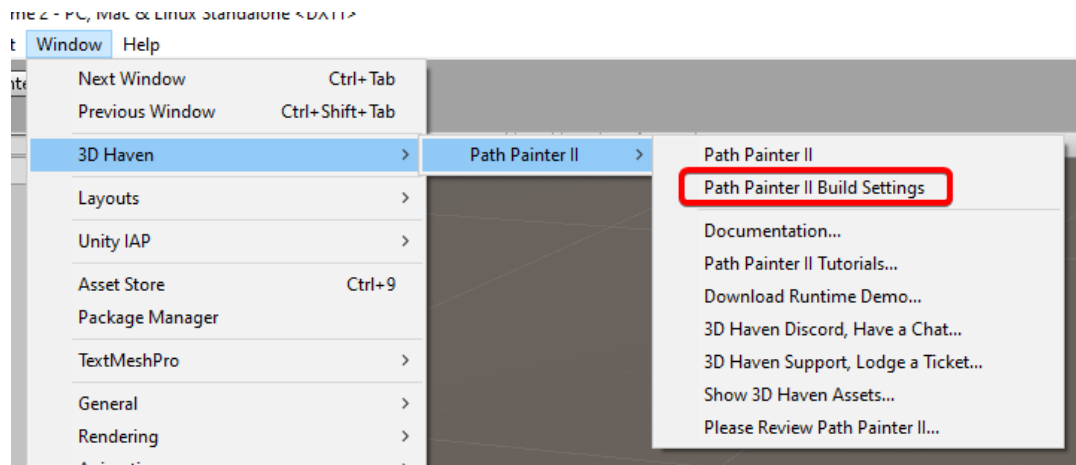
See and try out the examples provided in the Demo Scene. For example, the below shows the paint mechanism in the spline compatible example:

```
1  #define DLD_BEZ_SOL
2  using System.Collections.Generic;
3  using UnityEngine;
4  using Haven.API.PathPainter2;
5  using UnityEngine.UI;
6  #if DLD_BEZ_SOL
7  using BezierSolution;
8  #endif
9  // Used one of the free spline tools in this example. You can import that tool, uncomment the first
10 line (#define DLD_BEZ_SOL) and see how one could use a spline tool to drive Path Painter.
11 The tool that was used in this example is called Bezier Solution by yasirkula.
12 Get it here: http://bit.ly/3DHFreeSpln
13 You can also easily port to the spline tool of your choice.
14 Have fun!
15
16 namespace Haven.Demo
17 {
18     /// <summary>
19     /// A crude Path Painter API example that features a rudimentary undo system.
20     /// </summary>
21     public class PPAPIExampleSpline : MonoBehaviour
22     {
23         // Public, Inspector Members
24
25         // Enums
26
27         // Private Members
28
29         // Initialisation
30
31         // GUI Interface
32
33         // Point getters from lines/splines
34
35         #region Paint Mechanism
36
37         /// <summary>
38         /// Paint using the selected lines
39         /// </summary>
40         private void Paint(int lineSelection)
41         {
42             if (!m_validSetup)
43             {
44                 return;
45             }
46             m_activeLineSelection = lineSelection;
47             if (m_undoEnabled)
48             {
49                 m_undoSnapshot = new Dictionary<TerrainData, TerrainSnapshot>();
50                 m_undoStack.Push(m_undoSnapshot);
51             }
52             ApplyToLines(GetLines(m_activeLineSelection));
53         }
54
55         /// <summary>
56         /// Applies the selected painting following the selected lines
57         /// </summary>
58         private void ApplyToLines(List<List<Vector3>> lines)
59         {
60             // A better design would be to not care how many lines and always use Bulk Painting, but wanted to show the difference here
61             if (lines.Count == 1)
62             {
63                 ApplyToLine(lines[0]);
64             }
65             else
66             {
67                 mPainter.StartBulkPaint(TerrainsSelector.position);
68                 foreach (List<Vector3> line in lines)
69                 {
70                     ApplyToLine(line);
71                 }
72                 mPainter.EndBulkPaint();
73             }
74         }
75
76         /// <summary>
77         /// Applies the selected painting following the line
78         /// </summary>
79         private void ApplyToLine(List<Vector3> lines)
80         {
81             switch (m_paintMode)
82             {
83                 case PaintMode.Paint:
84                     mPainter.Paint(lines, Painter.Size(m_pathSize), Painter.EmbankmentSize(m_embankmentSize),
85                                     Painter.Texture(m_pathLayer), Painter.EmbankmentTexture(m_embankmentLayer), Painter.EvenRamp(m_ramp));
86                     break;
87                 case PaintMode.Paint3D:
88                     mPainter.Paint3D(lines, Painter.Size(m_pathSize), Painter.EmbankmentSize(m_embankmentSize),
89                                     Painter.Texture(m_pathLayer), Painter.EmbankmentTexture(m_embankmentLayer));
90                     break;
91                 default:
92                     Debug.LogErrorFormat("[PPAPIExample] Houston! I have no idea what to do with this: {0}", m_paintMode);
93                     break;
94             }
95         }
96     }
97
98     #endregion
99
100     // Undo System
101 }
```

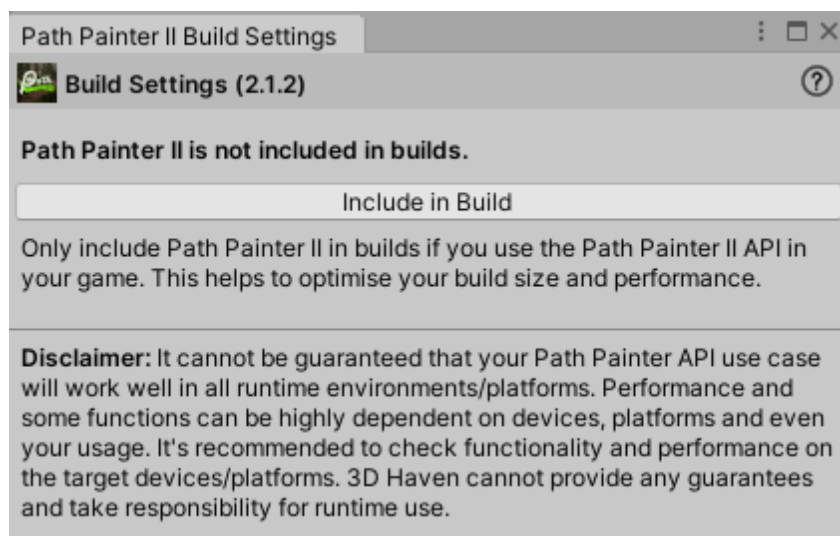
Using the API in Runtime (e.g. in-game)

Disclaimer: It cannot be guaranteed that your Path Painter API use case will work well in all runtime environments/platforms. Performance and some functions can be highly dependent on devices, platforms and even your usage. It's recommended to check functionality and performance on the target devices/platforms. 3D Haven cannot provide any guarantees and take responsibility for runtime use.

Use the Build Settings from the menu



to include/exclude Path Painter from your build.



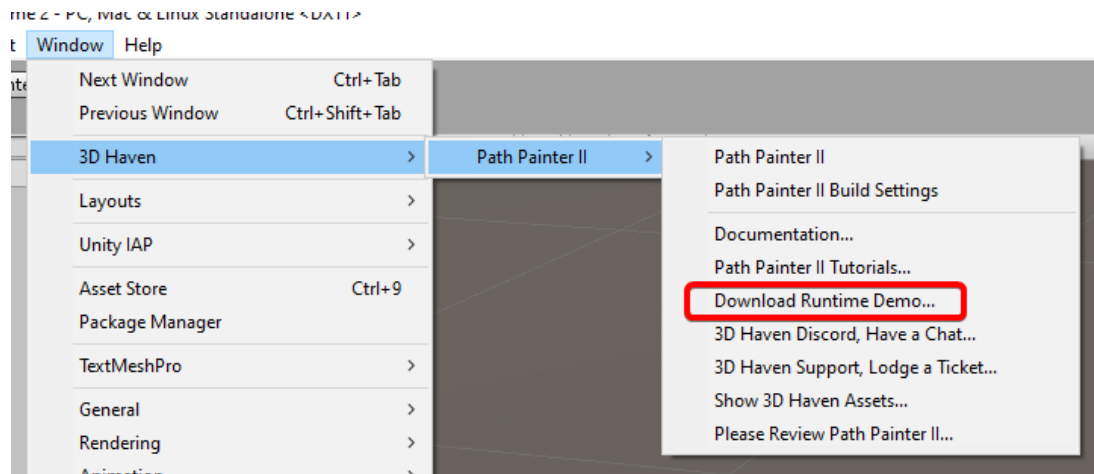
Using the Path Painter API in runtime effectively means that the API and Path Painters core is included in builds. From a build optimisation perspective, it's not recommended to include Path Painter in your builds if you don't use the API at runtime.

Download Runtime API Demos

Prerequisites: You need to include Path Painter in build to use the API in runtime. See [Using the API in Runtime \(e.g. in-game\)](#) to get more information about how to do this.

You can download the [Runtime API Demos from this link](#).

There is also an option in the Path Painter menu:



Troubleshooting

Below you will find some troubleshooting tips.

The shape of the path is jagged, not smooth

The possible reasons:

- Your *Heightmap Resolution* is too low for the fine details you are trying to create.
Solution: Increase your *Heightmap Resolution* [MapScaler](#).
- You are trying to create sharp edges or vertical walls that the Unity terrain can't handle.
Solution: Increase the Embankment Size until you are happy with the result.

The painted texture has sharp edges

The possible reasons:

- Your *Control Texture Resolution* is too low for the fine details you are trying to create.
Solution: Increase your *Control Texture Resolution* with [MapScaler](#).

Grass clearing is inaccurate and disproportionate

The possible reasons:

- Your *Detail Resolution* is too low for the fine details you are trying to create.
Solution: Increase your *Detail Resolution*.

Can't Nicely Connect Raised Paths or Lowered Riverbeds/Paths

See [Connecting Elevated Paths, Riverbeds](#) for a solution.

The Embankment is Not Being Painted Randomly

Try turning **Smart Texture Paint** off for those areas. See the [Textures](#) section for more information.

Performance Is Not Ideal When Painting Large Scale

Solution: If you find yourself constrained by performance while painting something large, you can try to paint with

- Shaping/Texturing/Vegetation clearing disabled ([Main Controls](#))
- Turn **Draw Vegetation** off (checkbox in the [Main Controls](#) section)
- smaller brush size ([Brush Settings](#))

then apply your desired settings in one go to the path afterwards.

Terrain Editing Reset When I Update My Path

The following workflow is not currently supported:

- Paint a path (and possibly realise something to be changed on the terrain outside of Path Painter for example a missing tree).
- Edit terrain outside of Path Painter.
- Tweak the path.

Solution: You can either

- finalise your path before making the other change, or
- create the path only after you made the other change.

See which solution best fits your situation.

Path Painter is not painting on some of the terrains in the scene

This is likely due to those terrains not being compatible with the currently selected one (see [Requirements](#) in [Neighbour Terrain Painting](#)). This might or might not be intentional.

Solution: Depending on that you may

- want to change their settings to be compatible, or
- if that's not what you want, just select the other type of terrains in your scene to paint on them.

Currently it's not possible in Unity to paint across non-compatible terrains and we don't know of a reason to do this. Visit [3D Haven Discord](#) in case you need to have path going across non-compatible terrains. Hopefully the community will be able to help you find a good solution.

Path Painter Layer Auto Propagation duplicates textures on the terrain

This generally means that the two layers are not the same. I.e., the textures have different settings. For example, in some situations, users want the same texture with different sizing and tiling on the same terrain.

Solution:

- If the two layers are meant to be the same, make sure that their settings are the same.
- Nothing special to do, if they in fact meant to be different and you want them both on the terrain. In this case, you can just allow Path Painter do its thing.

Path Painter doesn't remember my texture selection

Older Unity versions (pre-2018.3) might not always remember your texture selections. So far, we know of two situations when your last texture selection might not be remembered in a project:

- The same terrain where the layer (that contains the texture) was selected, or the texture itself is not loaded in the scene. For example, if it's a new scene, so the terrain is not in it, or the textures changed on the terrain (where the texture was selected) and the texture is no longer on it, or it isn't at the same position (not under the same index).
- Texture selection from older Unity versions (pre-2018.3) get lost when Unity is updated, because Unity update scrambles that information.

There are visible texture seams where the terrains meet

This can happen especially with older Unity versions (pre-2018.3), which were not made to support neighbour terrains.

Solution: Test if you can get the seams to disappear with Unity's built-in texture painter. If you can, please let us know, so we may be able to improve Path Painter. Unfortunately we haven't been able to figure out clear cut solutions to where Unity itself creates the seams (other than updating to newer Unity that was made with neighbour terrains in mind).