POLARIS

USER GUIDE

2020.1.4

PROLOGUE

Polaris is a complete solution for creating stunning large scale stylized environments, especially built for Unity developers. This is a must-have asset which saves you thousands hours of work and tons of effort, and gives you more time and resources to focus on other aspects of your game.

This User Guide is created to help you get started with the tool, including the most basic information about its features and workflow, as well as best practices, tips and tricks from its creator and showcasing amazing work from other successful developers.

To see the release log, please visit this page.

For business or support request, please contact: support@pinwheel.studio

RELEASE LOG

V2020.1.2

NEW FEATURES

- Adding support for Unity 2020

- Adding grass distance fade.

- Adding new callback when rendering grass to inject custom shader data.

- Adding Height Offset and Step properties for Ramp Maker.

- Adding Auto Tangent mode for Spline tool.

IMPROVEMENTS

- Optimizing terrain GetMesh function.

- Unload grass cell if it's inactive for a period of time.

- URP Support Extension now embedded in the package.

FIXES

- Many bugs fixing.

===

V2020.1

NEW FEATURES

- New geometry generation functions that leverage C# Job System and Burst Compiler.

- New job-based Foliage Renderer with instanced rendering, cell-based, volume-based, frustum-based culling.

- New job-based tree collider system.

- Billboard mode for grass.

- Adding support for MicroSplat (extension module published by Jason Booth).

- Adding support for Vegetation Studio Pro (provided as separated free package).

- Adding direct option for Vertex Color from Albedo Map.

- Adding option for time-sliced mesh generation, in game and in editor.

- Adding option to save geometry mesh to asset, or re-generate on enable, saving storage space.

- New Mask settings for Terrain Data.

- New Mask painter which lock a particular region of the terrain from being edited, and for other purpose such as VSP vegetation mask.

- New Wizard window that easier to setup and get started.

- New grass shape Clump for a denser look and reduce instance count.

- Convert from Tree/Grass prototype group to Prefab prototype group and vice versa.

IMPROVEMENTS

- Many editor refinements.

- New grass serialization format, reduce storage size.

- Expose more option for tree shadow, billboard shadow and grass shadow.

- Split Draw Foliage to Draw Trees and Draw Grass options.

- Improve Unity Terrain Data Importer/Converter performance.

- Allow Unity Terrain Converter to use a Polaris Terrain Data as template.

- Reduce shadow pop-in and out artifact when render trees.

- Better tree billboard shader.

- Allow using transform gizmos to move/rotate/scale the whole spline.

- Tone down geometry painter sensitivity.

- Adding Target Strength for more control over painter sensitivity.

- Refine smooth painting strength/kernel.

- Faster Undo recording for Paint Tools, Spline Tools and Stamp Tools.

- Display Splat/Tree/Grass selector as grid when there are many prototype group used in the scene.

- Adding undo/redo for vertex color painting.

- Faster grass painting, no more green-quad preview and batching.

FIXES

- Texture stamper multi-terrain and precision bug.

- Fix hardcoded path in C# and shader code.

- Foliage have correct position after convert from Unity terrain.

- Gradient lookup texture have correct color in linear mode.

- Fix visibility stamp bug when there is no stamp texture assigned.

- Fix all reference from "Griffin" to "Polaris" in document, help database and shaders to avoid confusion.

- Change all OnSceneGUI callback to DuringSceneGUI.

- Many minor bug fixings.

DEPRECATED

- Remove support for LWRP.

- Remove batched grass rendering (use instanced rendering instead).

- Remove Grass Generated Data asset.

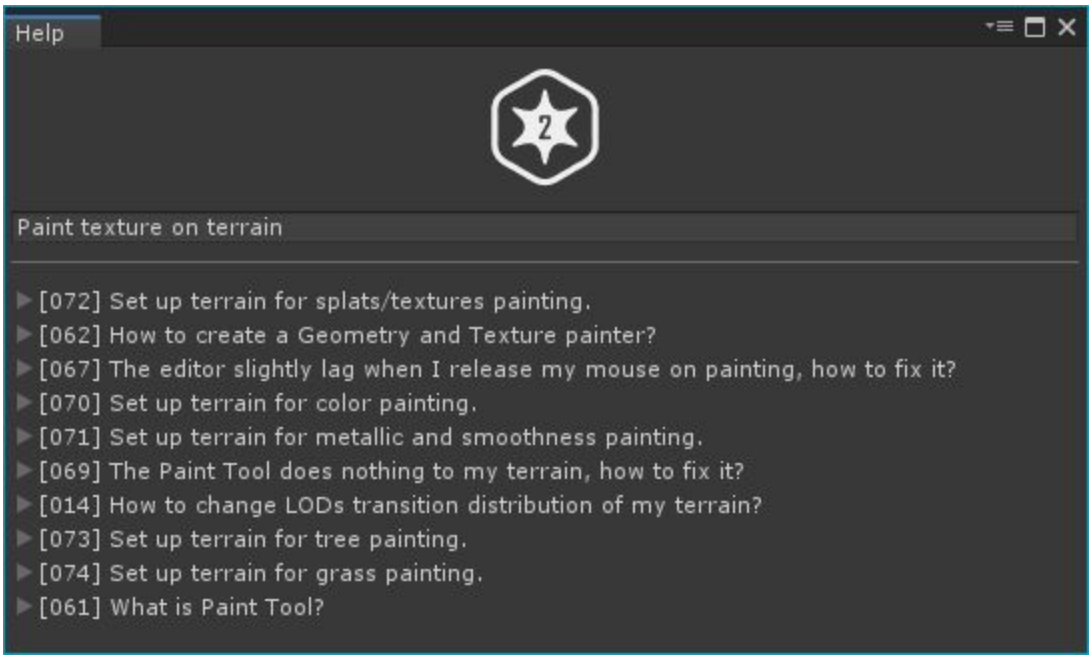
- Remove Polygon Processor option.

- Remove Polygon Distribution option.

- Remove support for Unity 2018, now requires 2019+.

FAQ

Polaris comes with a handy Help Tool which lets you search for solutions right in the editor. Instead of composing an email and waiting days for response, now you can have the answer instantly. Go to **Window>Polaris>Learning Resources>Help** to open it.



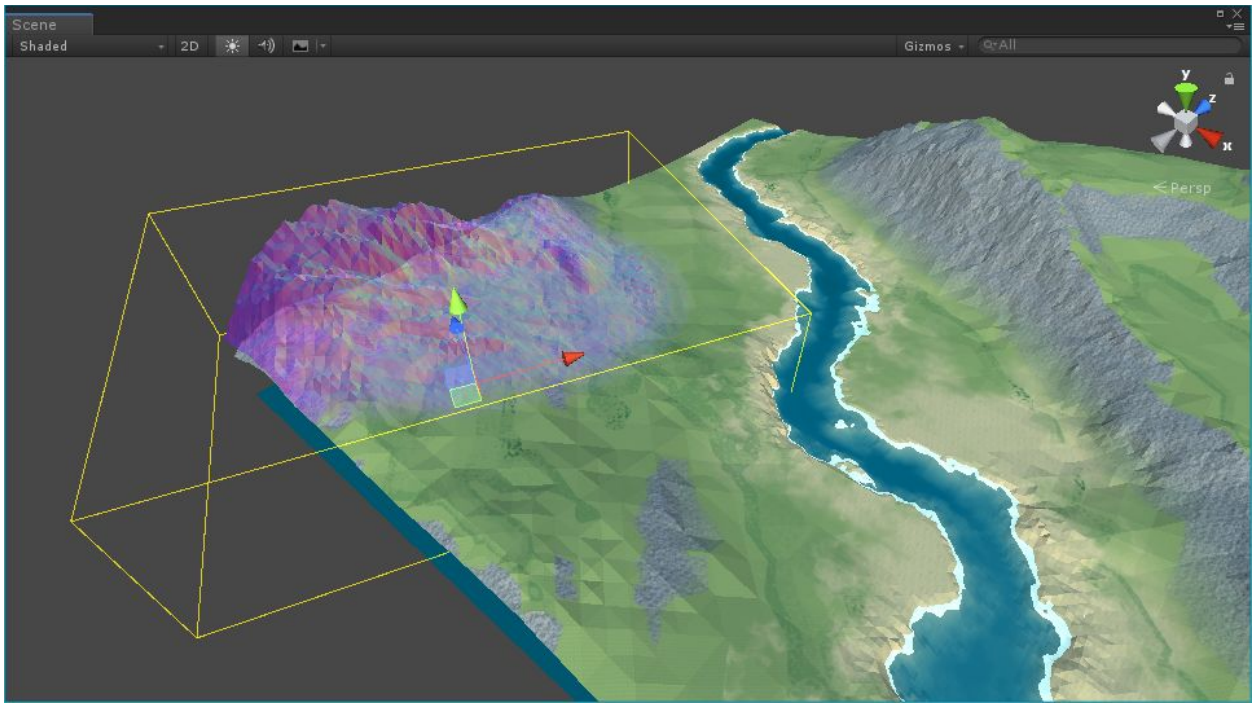
QUICK START

Main structure

The core architecture of Polaris is designed to mainly support multi-terrains editing and GPU processing. Every terrain in a scene consists of 2 parts: the terrain component and the terrain data asset, where the terrain component is responsible for utilizing the data and creating the actual terrain.

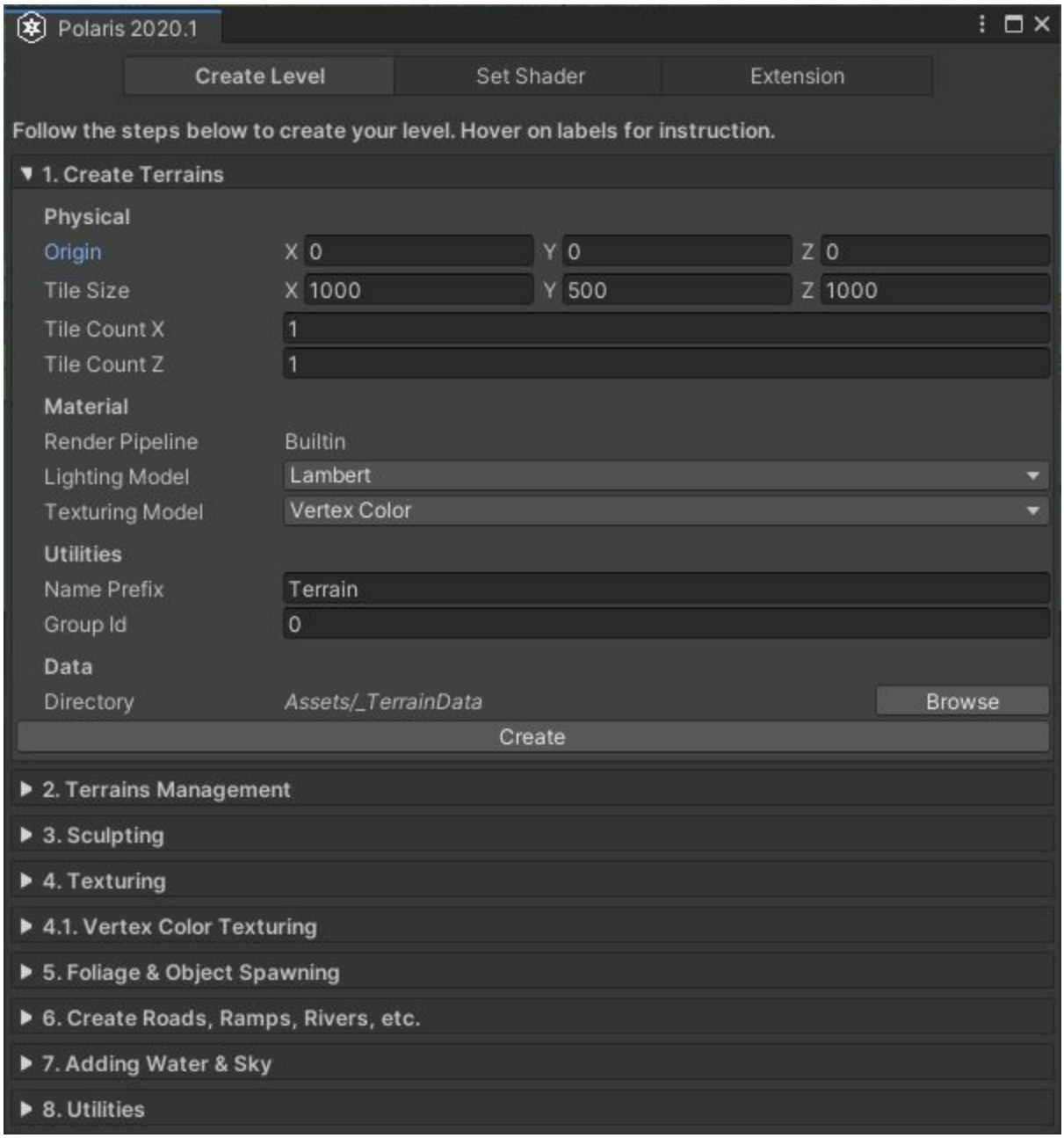
Terrain editing tools are not combined with the terrain component. They are built as separate components and only mean to modify terrain data. There are some pre-made tools for you to use:

- Geometry & Texture Painter: Sculpt geometry and paint color, metallic, smoothness, splats, etc. onto the surface.
- Foliage Painter: Paint trees and grasses.
- Object Painter: Spawn game objects into the scene.
- Spline: Create spline and do some tasks like make ramps, paint paths, clear foliage, etc.
- Geometry Stamper: Stamp features onto the surface geometry, using some basic math operation to blend the result like add, subtract, min, max, linear interpolation, etc.
- Texture Stamper: Stamp texture/color onto the surface, with additional procedural blending methods like height-based, slope-based or noise.
- Foliage Stamper: Similar to Texture Stamper, but instead of stamping texture, it stamps trees and grasses.
- Object Stamper: Spawn game objects into the scene, with some procedural rules.



The Wizard

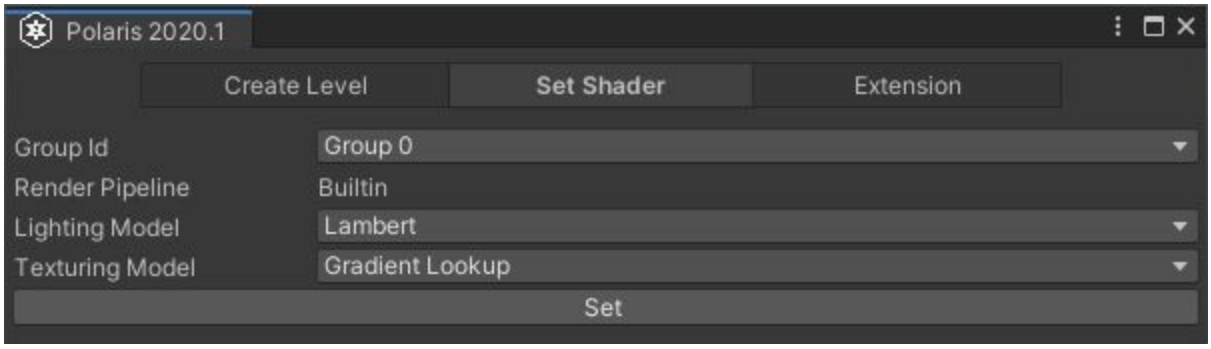
Polaris provides a Wizard window to help you get started with the most used parts of the tool. Right click on the Hierarchy, select **3D Object>Polaris>Terrain Wizard** to open it.



You should follow these steps from top to bottom. In each step there will be some instruction to help you go faster.

Shading styles

Polaris supports a wide variety of shading styles, from geometry-based (height, slope) gradient lookup, albedo metallic, splat map blending, to vertex color. Some terrain tools only work with a specific shading style. There is a wizard tool to help you to configure the material, we will talk about it later.



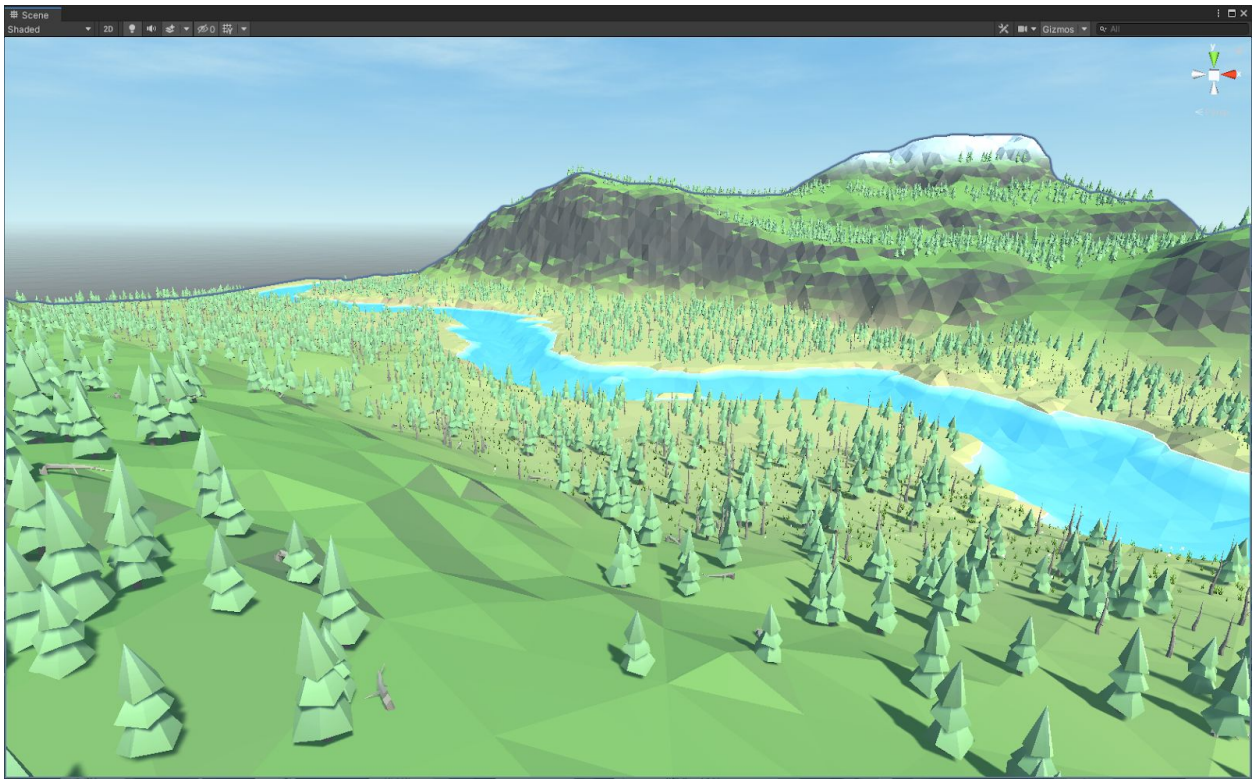
There are 3 lighting model included:

- PBR: physical based, the highest visual quality and the most expensive model.
- Lambert: simple lighting, without specular.
- Blinn Phong: simple lighting, with specular.

Gradient Lookup

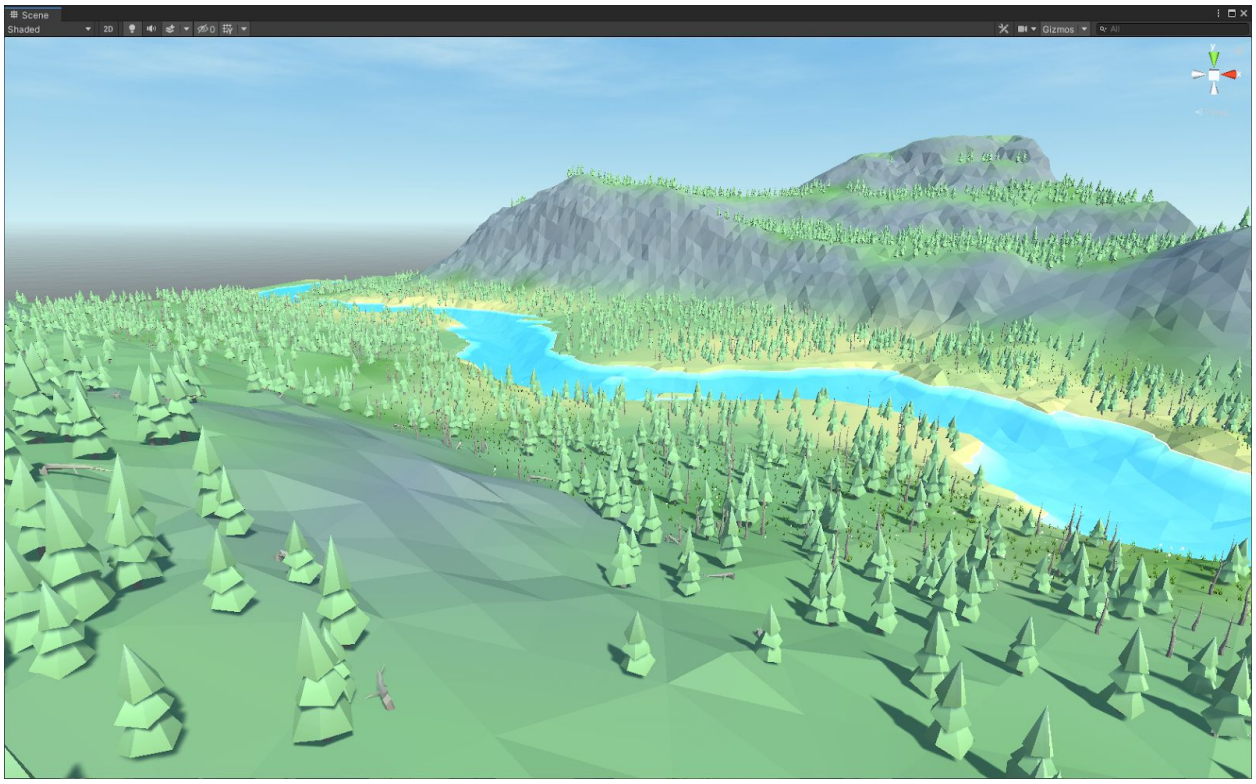
This model uses 2 gradients to shade the terrain based on vertex height and normal vector. The 2 gradients are blended together using a curve.

This is the easiest way to color your terrain procedurally. You can also paint an additional albedo layer on top, for things like roads and fine details.



Color Map

This model uses a single albedo map for the whole terrain. Suitable for small terrains that don't require much detail.



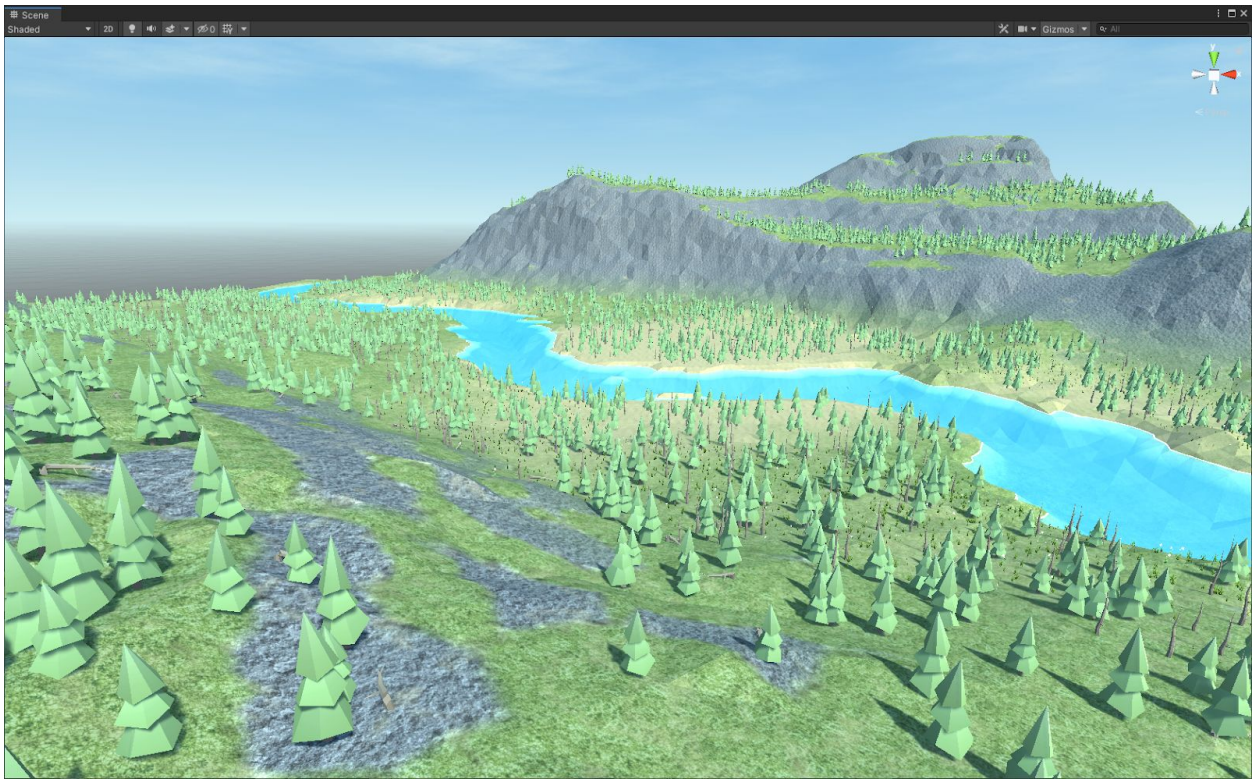
Splats

Similar to Unity terrain, it uses multiple textures stacked on top of each other, blended by control/alpha maps. This model gives the highest detail level, especially at close up distant.

There are 3 sub-model:

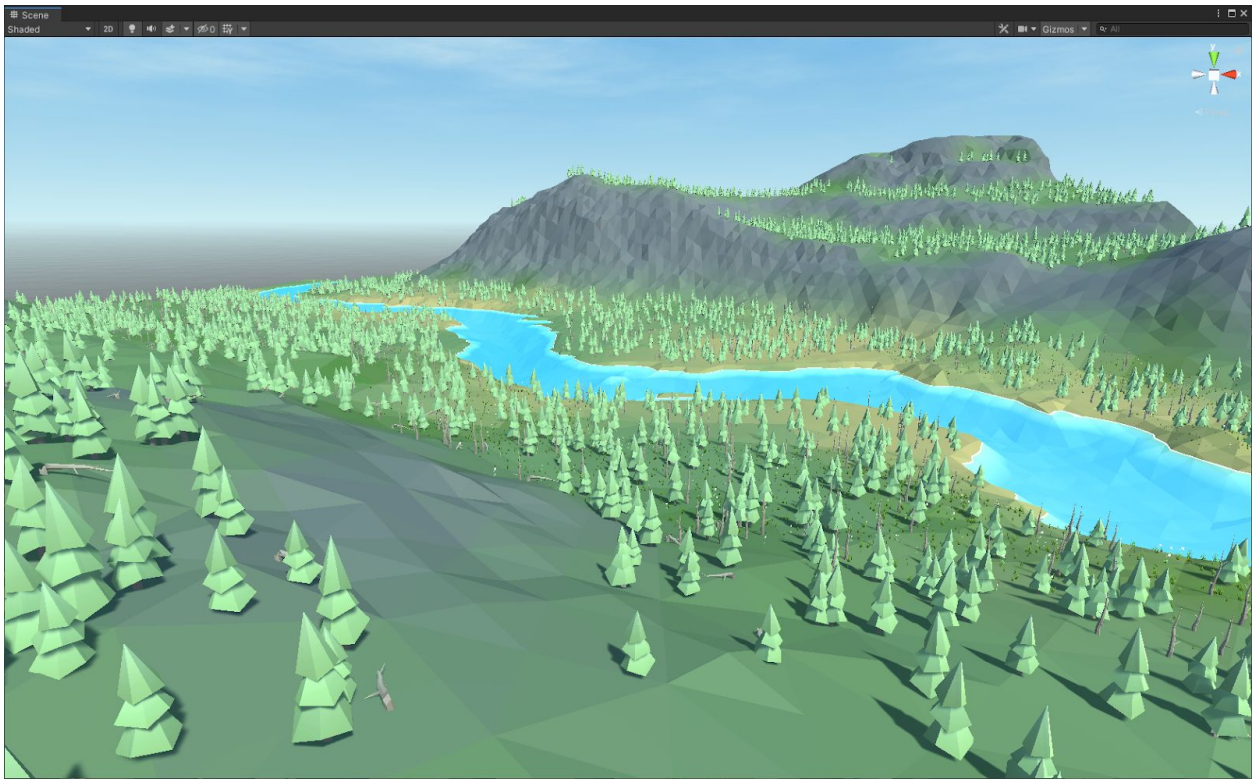
- 4 Splats: use up to 4 splat textures.
- 4 Splats 4 Normals: use up to 4 splat textures with normal maps.
- 8 Splats: use up to 8 splat textures.

You can use more splat textures with more complex shading effect with MicroSplat Integration.



Vertex Color

This one uses vertex color channel for the terrain. Vertex colors are read from the albedo map when generating terrain geometry. It can yield sharper color for each triangle, compared to Color Map mode.



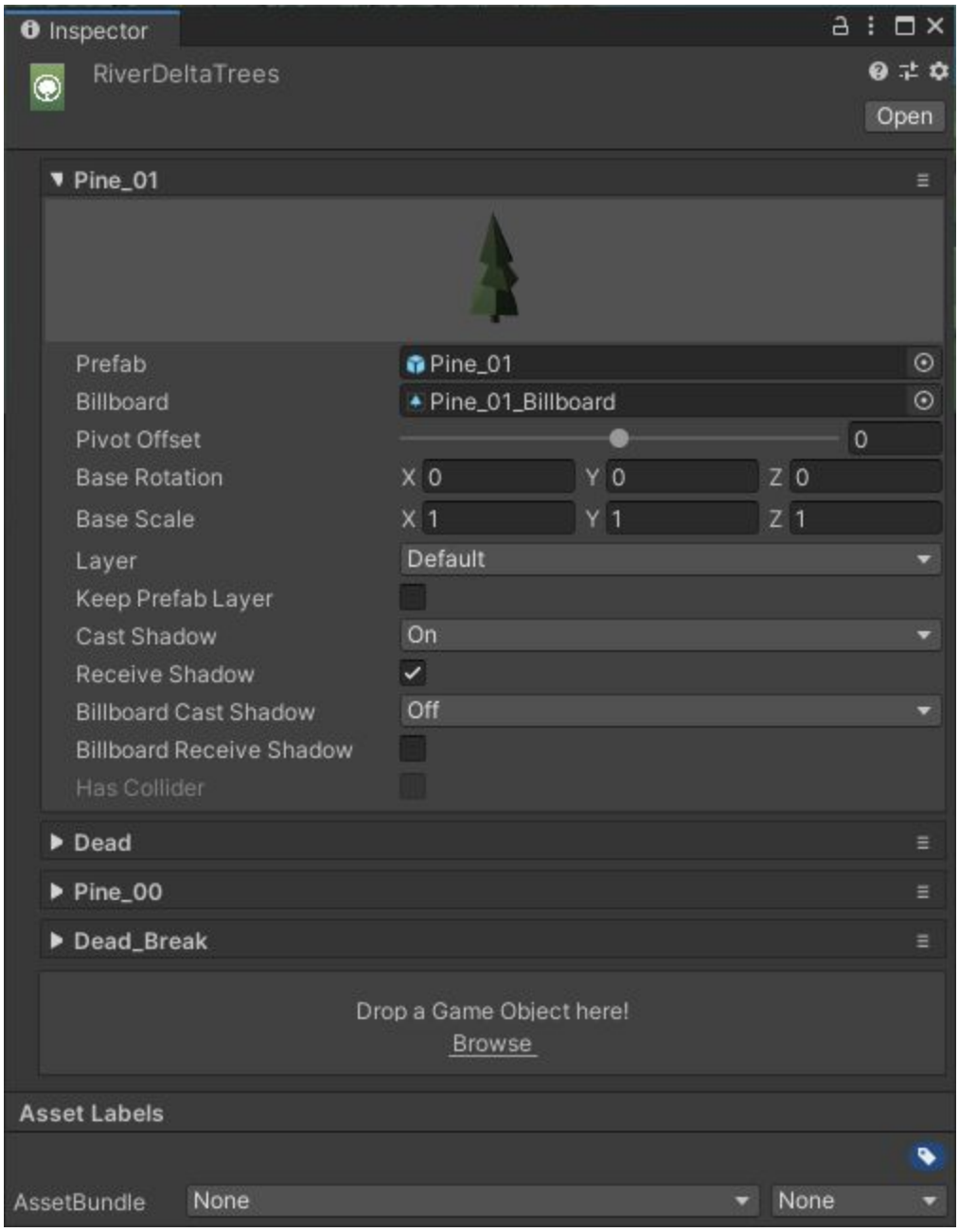
Main assets

There are some asset types you need to care about:

- Terrain Data: contains data related to a single terrain, such as its height map, material textures, foliage instances and other settings. A new instance will be created when you create a terrain. Each instance will be assigned with an ID for some purpose. **Do NOT duplicate this asset**, it will not work. Instead using data export/import for copying data.

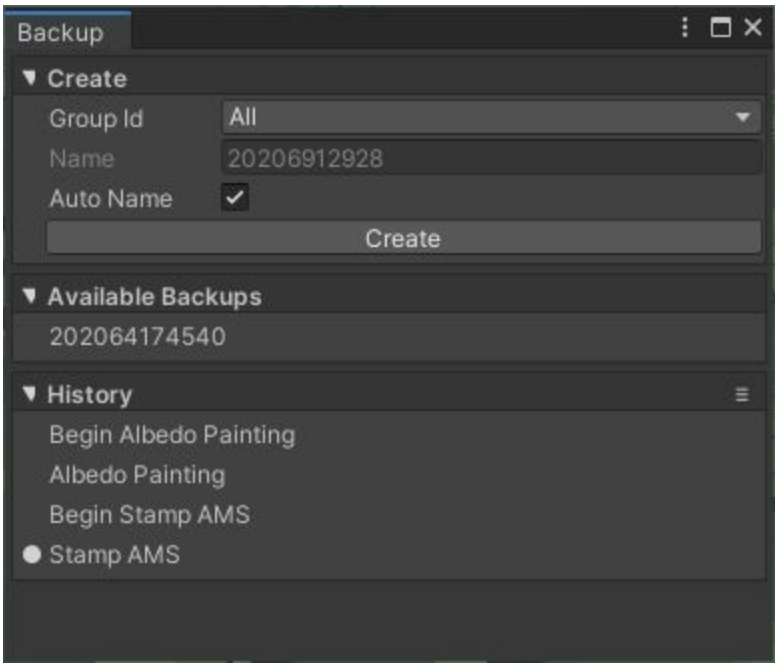
- Terrain Generated Data: contains generated geometry meshes. You don't need to commit this asset using a version control, since it can be re-generated using terrain data. This asset is created automatically.
- Splat Prototype Group: contains a set of splat texture, can be shared between multiple terrains. Go to **Project Window>Create>Polaris>Spat Prototype Group** to create one.
- Tree Prototype Group: contains a set of tree prefabs and related settings, can be shared between multiple terrains. Go to **Project Window>Create>Polaris> Tree Prototype Group** to create one.
- Grass Prototype Group: contains a set of grass texture and detail object prefabs with related settings, can be shared between multiple terrains. Go to **Project Window>Create>Polaris> Grass Prototype Group** to create one.

Since Polaris supports multi-terrain editing, some properties like splat textures and foliage prototypes are not bound to a single terrain. Instead they're configured in an asset object as a gallery, then that asset will be assign to multiple terrain in the scene:



Undo and Backup

Polaris provides a custom Backup System to help you manage editing history better. Go to **Window>Polaris>Tools>Backup** to open the Backup editor.



Polaris will record a snapshot of your terrains after a modification. Each snapshot will have a name depending on which action you did. Note that this system is non-linear, which means you can restore data at any point in the History list by clicking on that entry. History entries are cleared after the Unity editor is closed.

You can also use Ctrl+Z and Ctrl+Y to perform undo/redo as usual.

Different from History snapshot, a Backup will write all* data onto disk, which will remain across editor sessions. Use the Create foldout to do so. Backup files are stored in the GriffinBackup folder, right next to your Assets folder.

See Backup Tool for more detail.

*** This system only records textures and foliage data, other numeric values will not be recorded.**

Frequently used editor menus

You can easily find Polaris functionalities in these editor menus:

- **Assets>Create>Polaris>…**: Create Polaris specific assets like Terrain Data, Splat Prototypes Group, Foliage Prototypes Group, etc.
- **GameObject>Create>3D Object>Polaris>...**: Create terrains and terrain tools in the scene.
- **Window>Polaris>...**: Open additional editor window or configure Polaris global settings.

Maximize performance

Polaris can function fine without the need of installing any additional package/asset, except the extension for URP support.

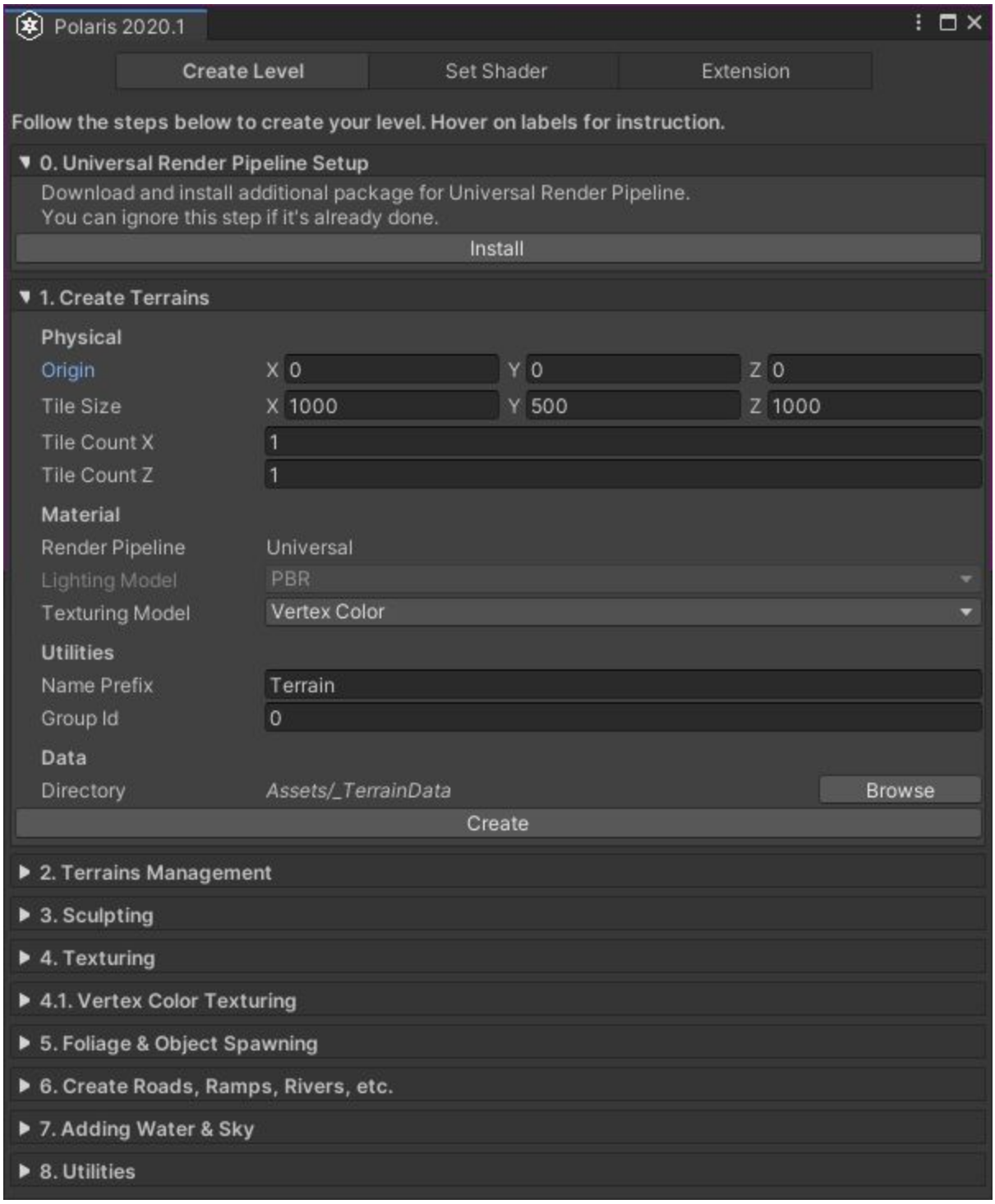
However, to maximize its performance and possibility, consider installing these packages/assets:

- Unity Burst Compiler: install via the Package Manager (com.unity.burst). You will get faster geometry generation and foliage rendering with this.
- Editor Coroutines: install via the Package Manager (com.unity.editor-coroutines). This will enable time-sliced generation in editor mode.
- Shader Graph: install via the Package Manager (com.unity.shadergraph). This will let you customize shaders for URP.
- Amplify Shader Editor: customize shaders for Builtin RP
- MicroSplat and MicroSplat-Polaris Integration: adding more advanced terrain shaders. See this page for setup guide.
- Vegetation Studio Pro: more option for spawning and rendering foliage, instead of using the default one. See this page for setup guide.

CREATE AND CONFIGURE TERRAIN SETTINGS

Create terrain game objects

Go to **GameObject>3D Object>Polaris>Terrain Wizard**, this will bring up the Wizard window:



Make sure you're on the Create Level tab.

This tab is a step by step instruction for you to create terrain game objects, managing and editing your level. Follow the steps from top to bottom to get used to the tools and its workflow.

If you're using Universal Render Pipeline, there will be an additional step to help you download and setup necessary shaders and materials.

To create new terrains, expand the **1. Create Terrains** foldout. The parameters are described as follow:

Physical:

- Origin: position of the first terrain in the new terrain grid.
- Tile Size: size (WxHxL) of each terrain tile in world space.
- Tile Count X/Z: number of terrain tiles along X/Z axis.

Material:

- Render Pipeline: the render pipeline currently in use. Only support Builtin and Universal.
- Lighting Model: select between Lambert (diffuse), Blinn Phong (specular) or PBR (physical based). In URP, only PBR model is supported.
- Texturing Model: choose how to apply texture to the terrain, gradient lookup, albedo metallic, or splat map blending, etc. **These settings may affect some terrain tools.** See Shading styles for more details.
- Splats Model: choose the maximum splat layers to apply for the terrain, if using Splat texturing model.

Utilities:

- Name Prefix: a string to append to the beginning of terrain's name. Useful for some level streaming system.
- Group Id: an integer to gather terrains into a group, which will affect seam stitching and multi-terrains editing.

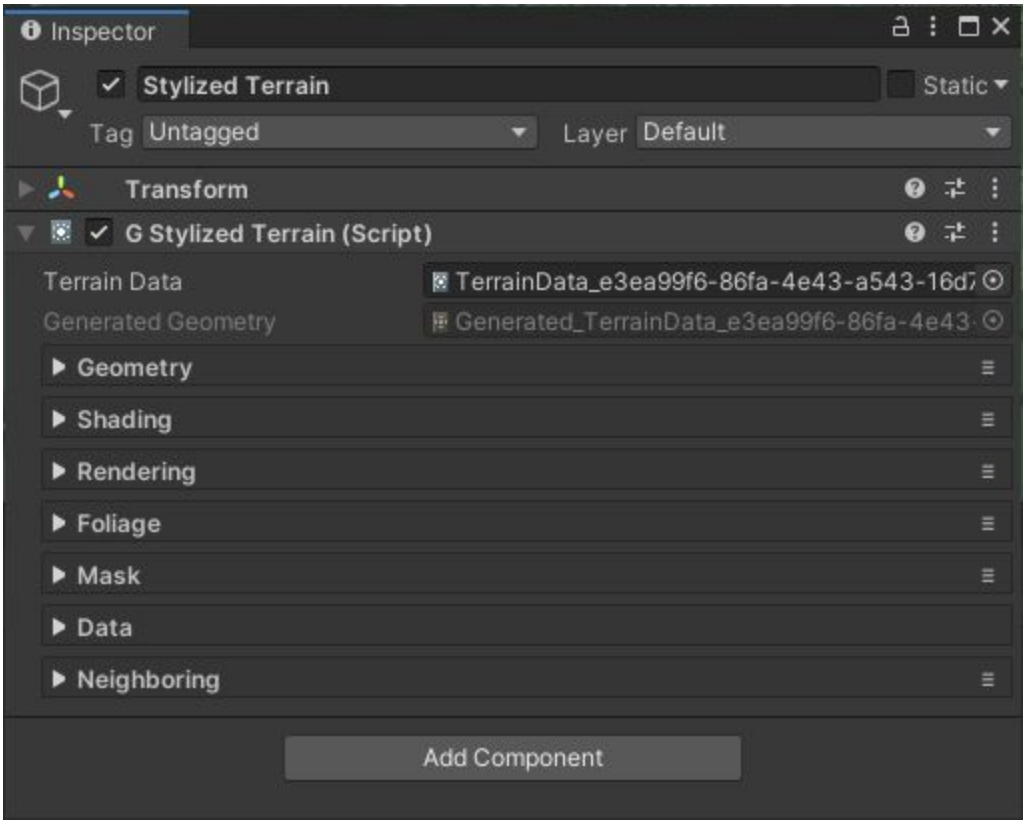
Data:

- Directory: a folder to store all generated assets. There are many assets to be created, so a subfolder of Assets/ is preferred.

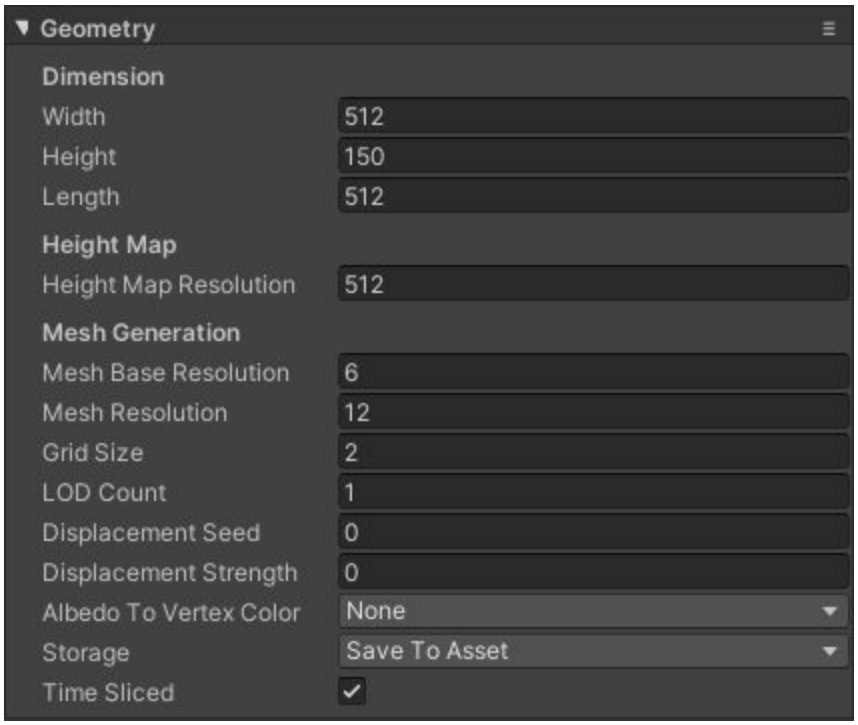
Fill up these parameters with your value, then hit Create.

Configure terrain settings

Select a terrain in the Hierarchy, in the Inspector you will see there are many settings:



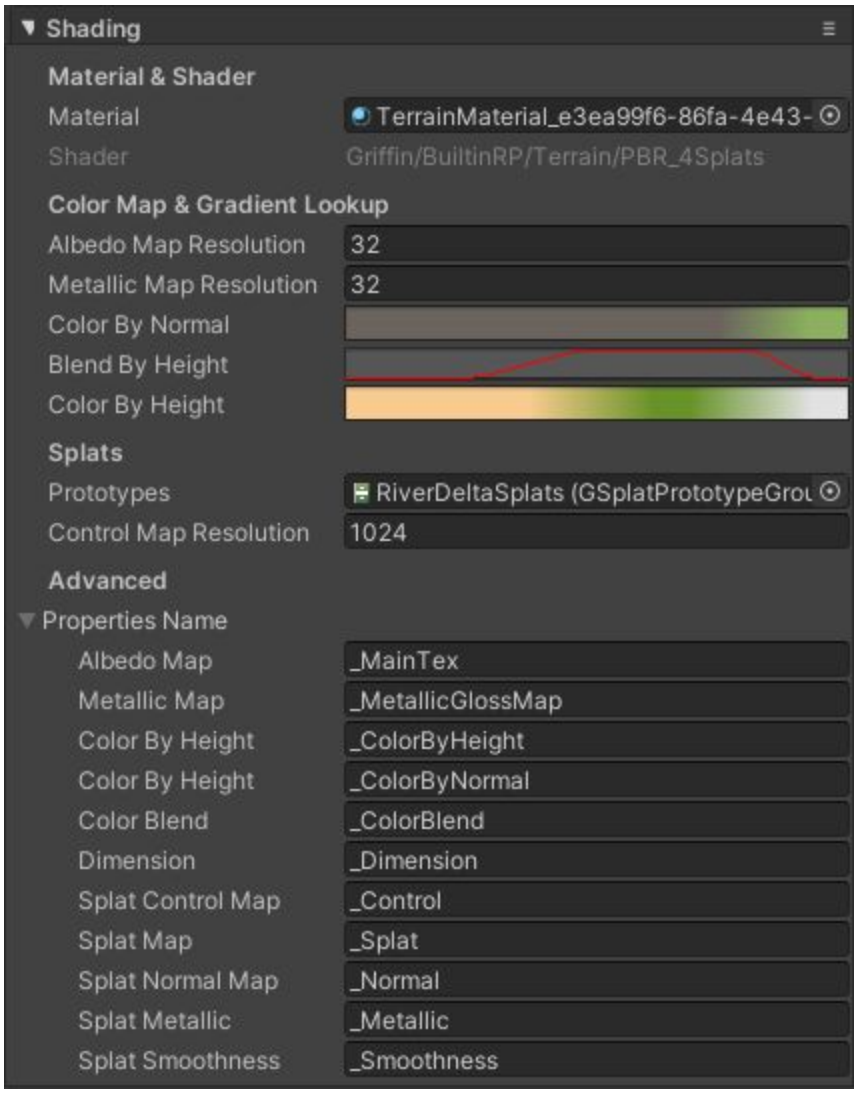
The Terrain Data and Generated Data asset is shown on top of the component.



For Geometry settings:

- Width: size of the terrain on X-axis in local space.
- Height: size of the terrain on Y-axis in local space.
- Length: size of the terrain on Z-axis in local space.
- Height Map Resolution: size of the height map texture in pixel.
- Mesh Base Resolution: the minimum resolution of geometry mesh.
- Mesh Resolution: the maximum resolution of geometry mesh where it has densest vertices distribution.
- Grid Size: how many chunks for the terrain mesh to split up on each X and Z axis. The total chunk is a square of this.
- LOD Count: the total LODs to generate. This should be 1 during editing to save some processing power, then you can try a higher value when editing is done.
- Displacement Seed: a random seed for vertex displacement.
- Displacement Strength: Strength of vertex displacement on XZ plane.
- Albedo To Vertex Color: how to fetch vertex color from albedo map.
- Storage: choose whether to write generated mesh to asset, or re-generate on enable to save some storage.

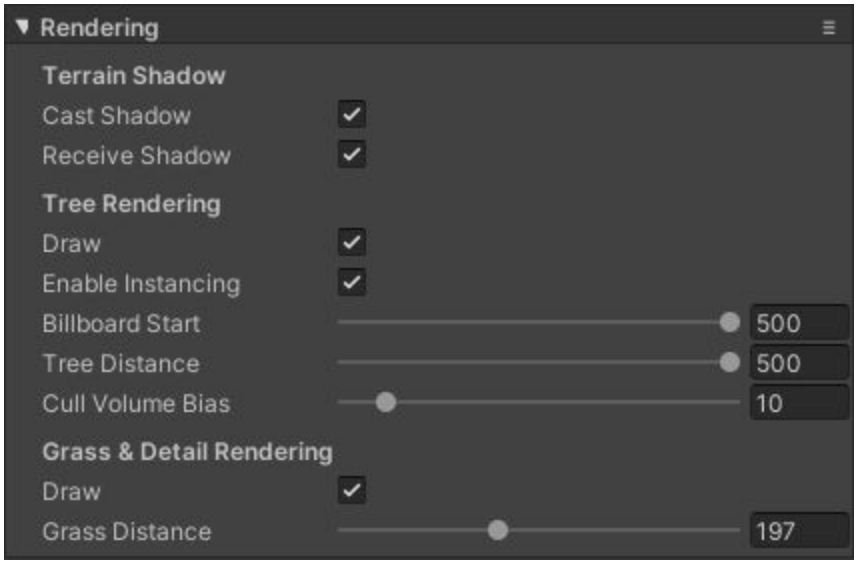
- Time Sliced: toggle time-sliced generation. If turned on, it will generate only one chunk per frame if possible. Some operations still generate all chunks at once.



For Shading settings:

- Material: the material to render the terrain.
- Albedo Map Resolution: size of the Albedo Map in pixel. Set this to a small number (1) to save memory if you don't use it.
- Metallic Map Resolution: size of the Metallic Map in pixel. Set this to a small number (1) to save memory if you don't use it.
- Color By Height, Color By Normal: gradients to shade the terrain based on vertex height and slope. Only affect Gradient Lookup shaders.

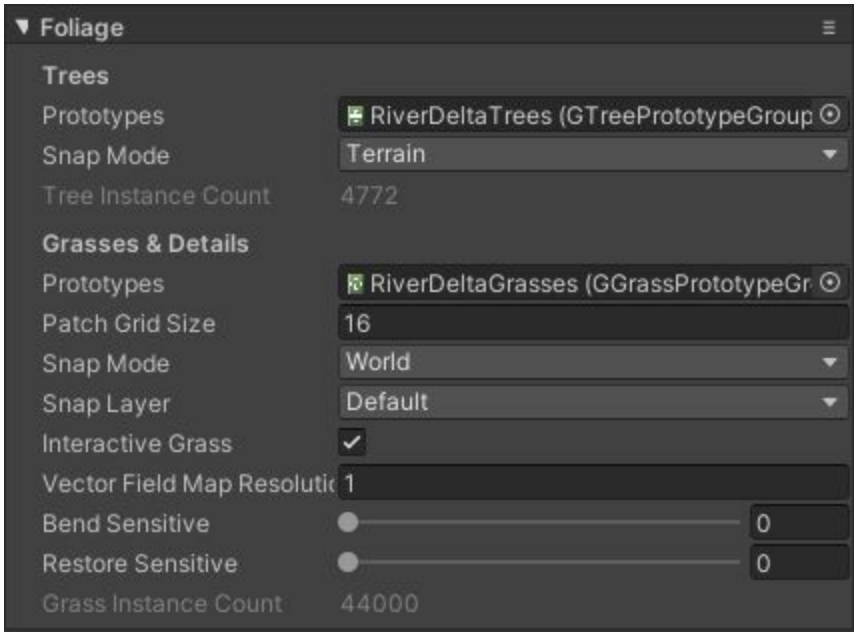
- Color Blend: blend fraction to interpolate gradients based on vertex height. Only affect Gradient Lookup shaders.
- Splat Prototypes: a collection of terrain textures to apply onto the surface. Only affect Splat shaders. See Create Splat Prototypes Group for more detail.
- Splat Control Resolution: size of the splat control maps in pixel. Set this to a small number (1) to save memory if you don't use it.
- Properties under Advanced section: These define the actual properties name in terrain shaders and tell Polaris how to bind data to terrain materials. Leave it as default if you don't use custom shaders.



For Rendering settings:

- Cast Shadow: should the terrain cast shadow?
- Receive Shadow: should the terrain receive shadow?
- Draw Tree: should it draw tree instances?
- Enable Instancing: Enable GPU Instancing for rendering trees. Note that this option requires tree materials to have Enable Instancing flag on, otherwise it will fallback to normal rendering. You can see some flickering in the Scene view sometimes, disable Dynamic Clipping for the Scene view camera will solve the issue.
- Billboard Start: distance from the camera to render trees as billboard. If the billboard is not available for that tree, it will not be rendered.
- Tree Distance: the maximum distance from the camera to render trees.

- Cull Volume Bias: adjust this property to prevent shadow pop in artifact. This is a global value.
- Draw Grass: should it draw grass instances?
- Grass Distance: the maximum distance from the camera to render grasses.



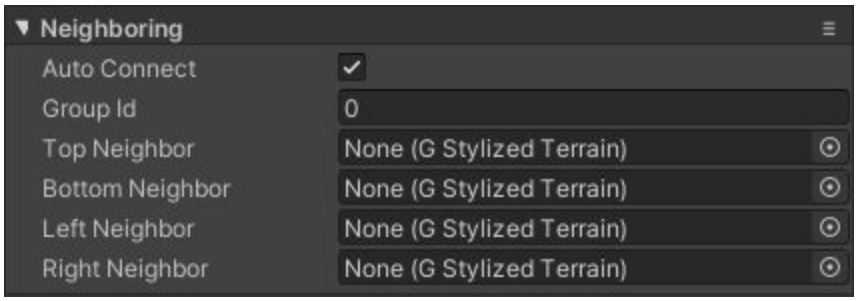
For Foliage settings:

- Tree Prototypes: a collection of tree prototypes to render. See Create Tree Prototypes Group for more detail.
- Tree Snap Mode: chose to snap tree to terrain surface or world objects.
- Tree Snap Layer Mask: a mask to filter out world objects which trees can snap on.
- Tree Instance Count: the total number of tree instances on the terrain.
- Grass Prototypes: a collection of grass prototypes to render. See Create Grass Prototype Group for more detail.
- Grass Snap Mode: chose to snap grass to terrain surface or world objects.
- Grass Snap Layer Mask: a mask to filter out world objects which grass can snap on.
- Patch Grid Size: determine the number of patches along each X and Z axis for grass instance batching. The number of patches is a square of this. Higher patch count requires more draw calls but allows a denser field.
- Interactive Grass: Enable touch bending for grass, make them react to moving objects in the scene. See Setting up Interactive Grass for more detail.

- Vector Field Map Resolution: size of the vector field map, higher value makes more precise grass bending, but cost more performance.
- Bend Sensitive: how fast grass reacts to moving objects.
- Restore Sensitive: how fast grass returns to its natural position when the moving object goes away.
- Grass Instance Count: the total number of grass instances on the terrain.

For Data Settings:

- Import: allow you import data from external sources like Unity Terrain Data, RAW files, etc.
- Export: allow you to export data to Unity Terrain Data, RAW files, etc.

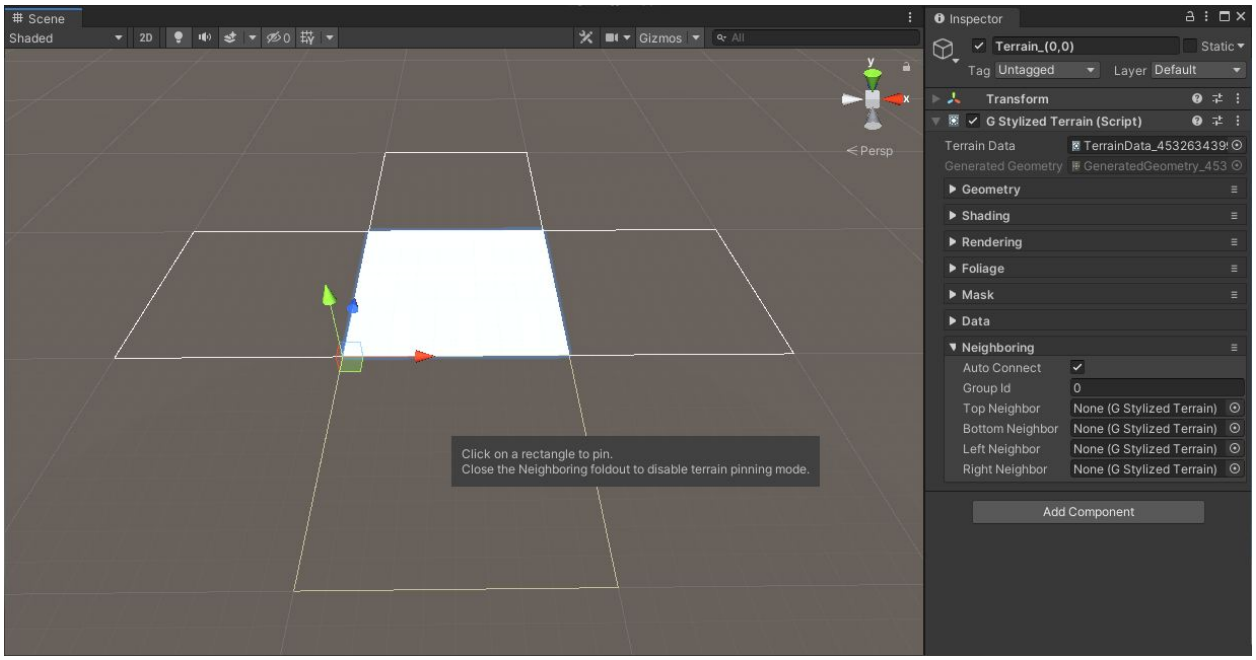


For Neighboring Settings:

- Auto Connect: indicate that the terrain should be connected automatically to its adjacent neighbors when a new terrain is added into the scene.
- Group Id: an integer number to group terrains in the scene. This will affect how terrain tools work. For example, you can have terrains in the play area with Group Id 0, and terrain in the background with Group Id 1, etc.
- Top/Bottom/Left/Right Neighbor: connect adjacent terrain together, their geometry will be matched up.

Create and connect neighbor terrains

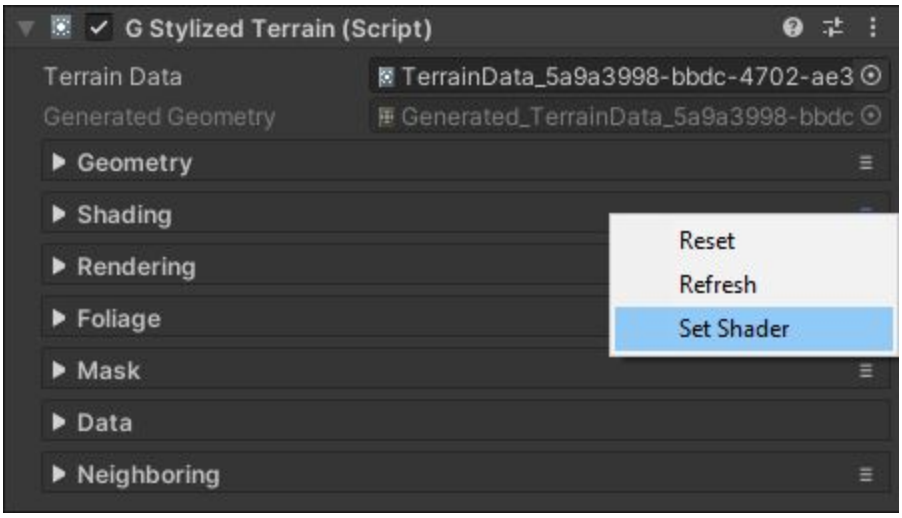
You can quickly create and connect adjacent terrains tiles using its Terrain Pinning & Neighboring feature. Select a terrain, expand it Neighboring foldout, you can see some square button next to it, click on the button and the new terrain will be added to the scene with neighboring configuration set.



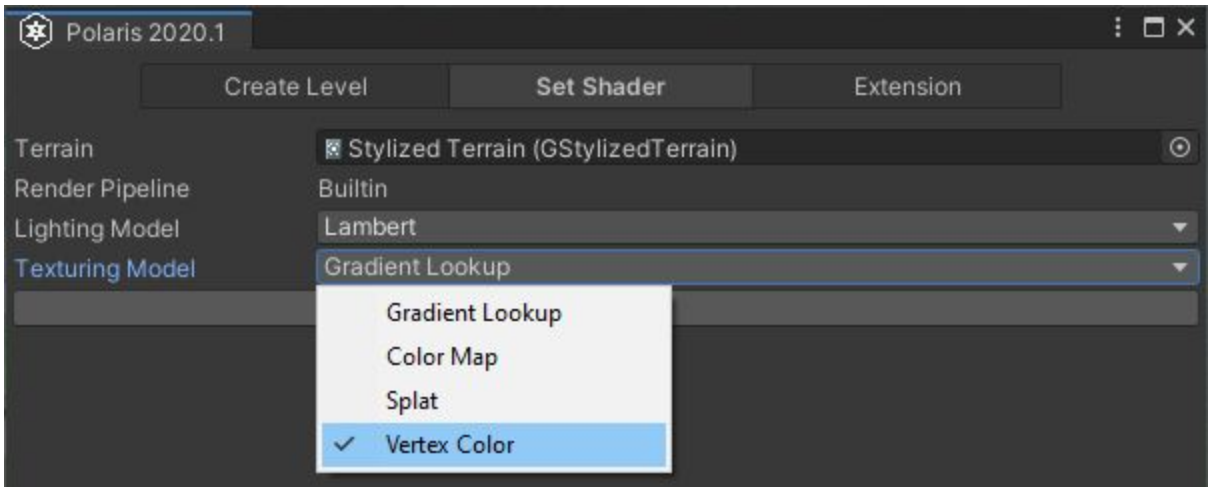
Changing terrain material

You can change terrain's material using the Wizard.

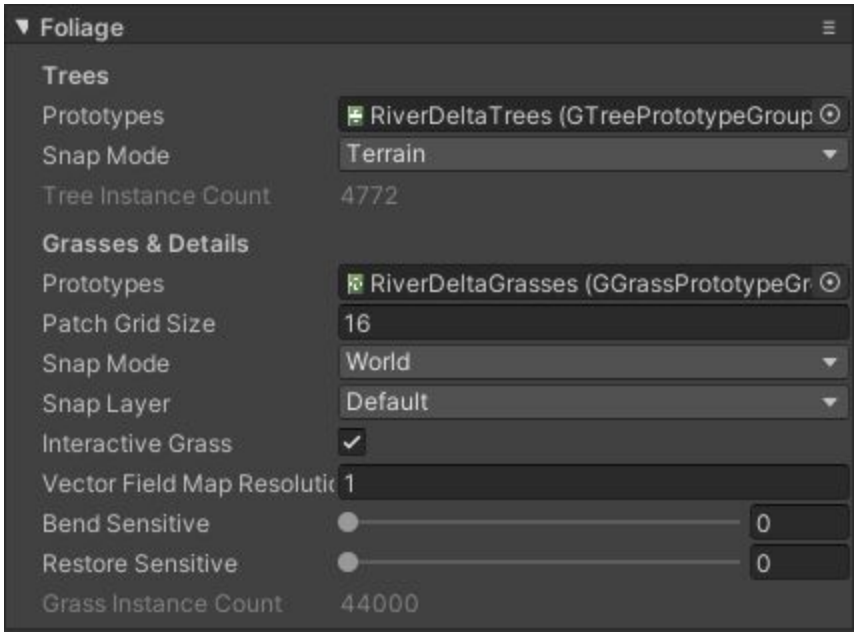
In the terrain Inspector, go to Shading>CONTEXT (☰)>Set Shader, it will bring up the Wizard window.



Select your desired Lighting Model, Texturing Model (and Splat Model as well), then click Set.

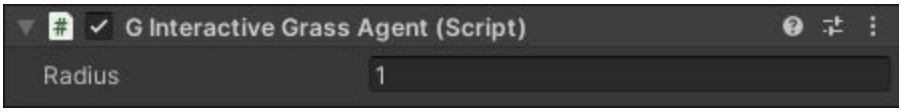


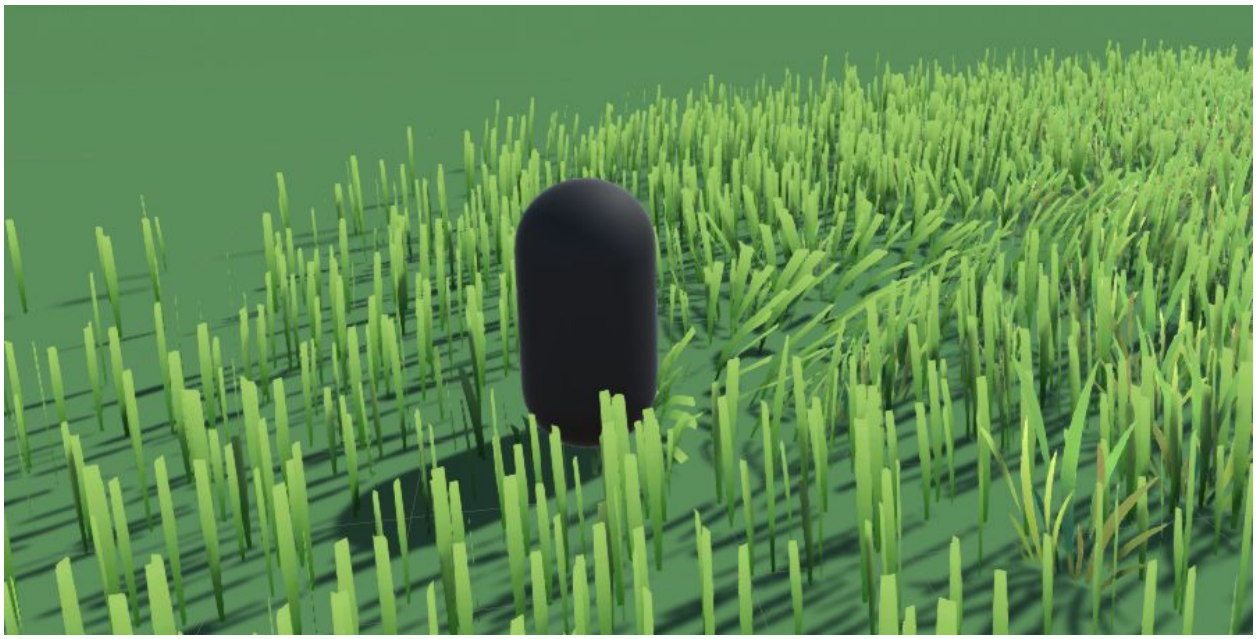
Setting up Interactive Grass



The setup process is quite simple. Select a terrain, in the Inspector, under Foliage fold out, check on the Interactive Grass toggle, then choose appropriate values for other properties like Vector Field Map Resolution, Bend Sensitive and Restore Sensitive. See Foliage settings for more detail.

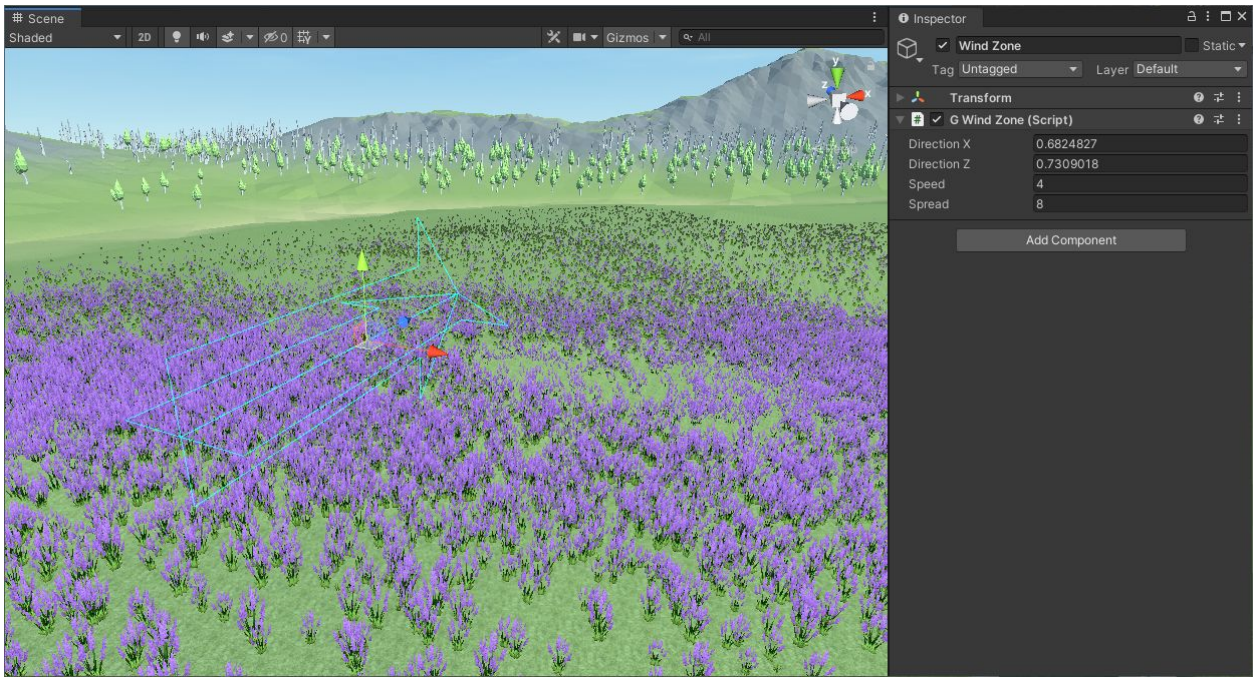
Next, the system needs to know who is the character so they can react correctly. Select your character, then add a GInteractiveGrassAgent component, pick an appropriate Radius value. You can add as many agents as you want.





Setting up Wind Zone

You can control global wind effect on grass in the entire scene by adding a Wind Zone game object. Go to **GameObject>3D Object>Polaris>Wind Zone** to add one.

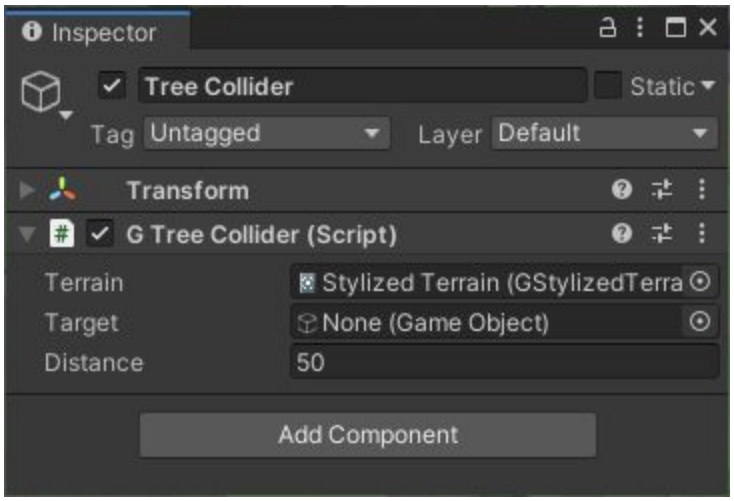


In the Inspector, there are some properties to define the wind effect:

- Direction X: direction of the wind on X-axis.
- Direction Z: direction of the wind on Z-axis.
- Speed: how fast the wind blows.
- Spread: Control wind spread/turbulence.

Create Tree Collider

A Tree Collider component will be automatically added in a child game object of the terrain when you create it.

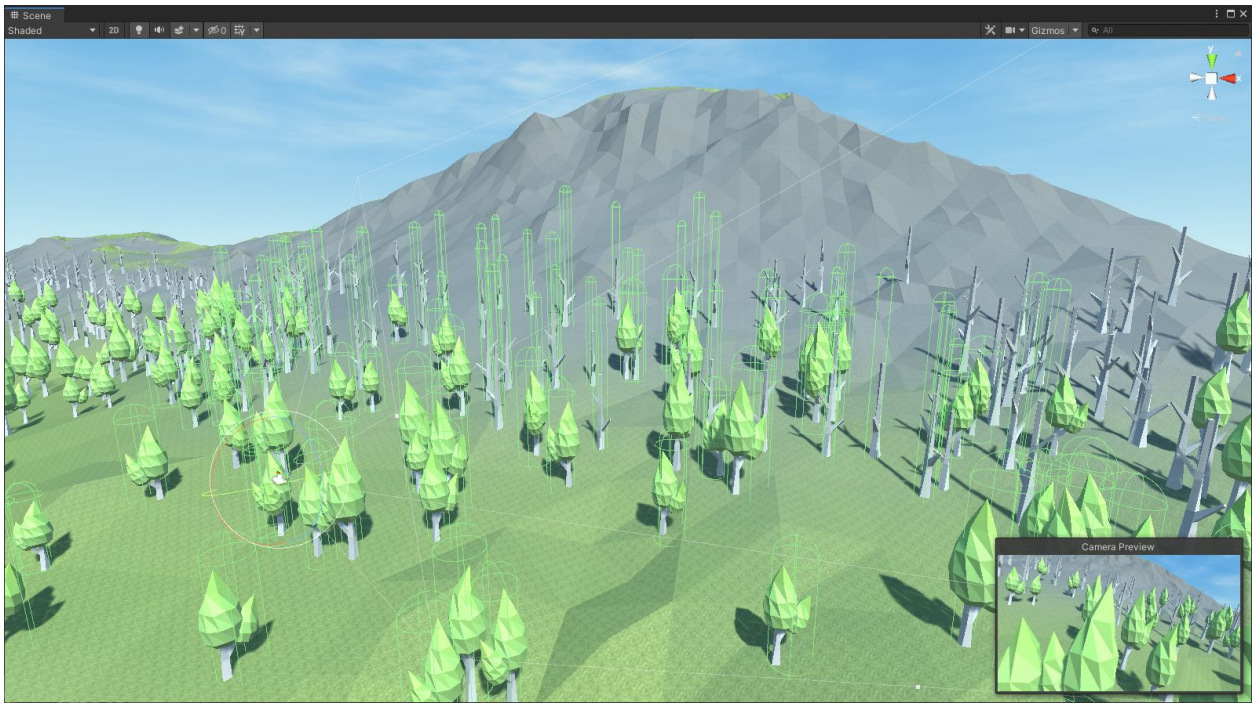


Internally, Tree Collider component stores a list of Capsule Collider, which will be updated every frame depending on its target position in the scene. Each Tree Collider will track only one target, to have more targets, you can add another Tree Collider by going to **GameObject>3D Object>Polaris>Tree Collider**.

To enable collider for a Tree Prototype, you must add a Capsule Collider to its prefab.

Tree Collider have the following properties:

- Terrain: the parent terrain which contains information of tree instances.
- Target: the target for it to track, if null, it will track the Main Camera.
- Distance: Maximum distance from the tree to its target.



Create Splat Prototypes Group

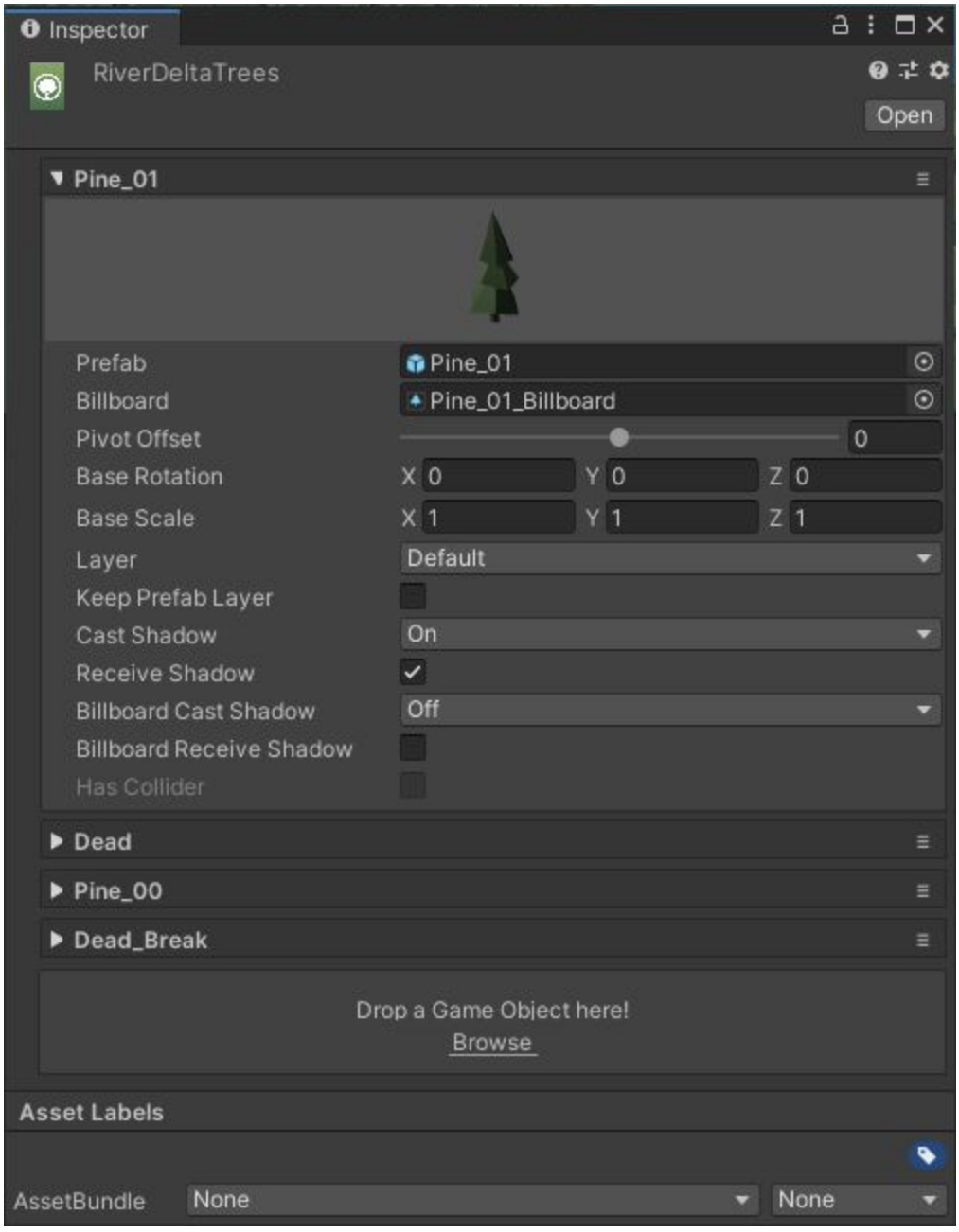
To apply splat textures to the terrain, you have to create a Splat Prototypes Group asset and assign it to the terrain. Go to **Assets>Create>Polaris>Splat Prototypes Group** and give the asset an appropriate name. Select the asset, in the Inspector, try drop a texture into the selector box, you should see the prototype appear:

For each prototype, there are several settings:

- Texture: the main texture of the prototype.
- Normal Map: a normal map for fine detail.
- Tile Size: size of the repetition pattern.
- Tile Offset: offset of the repetition pattern.
- Metallic/Smoothness: physical properties of the prototype, only used in PBR shader.

Create Tree Prototypes Group

To spawn and render trees on the terrain, you have to create a Tree Prototypes Group asset and assign it to the terrain. Go to **Assets>Create>Polaris V2>Tree Prototypes Group** and give the asset an appropriate name. Select the asset, in the Inspector, try dropping a prefab into the selector box, a prototype should appear:

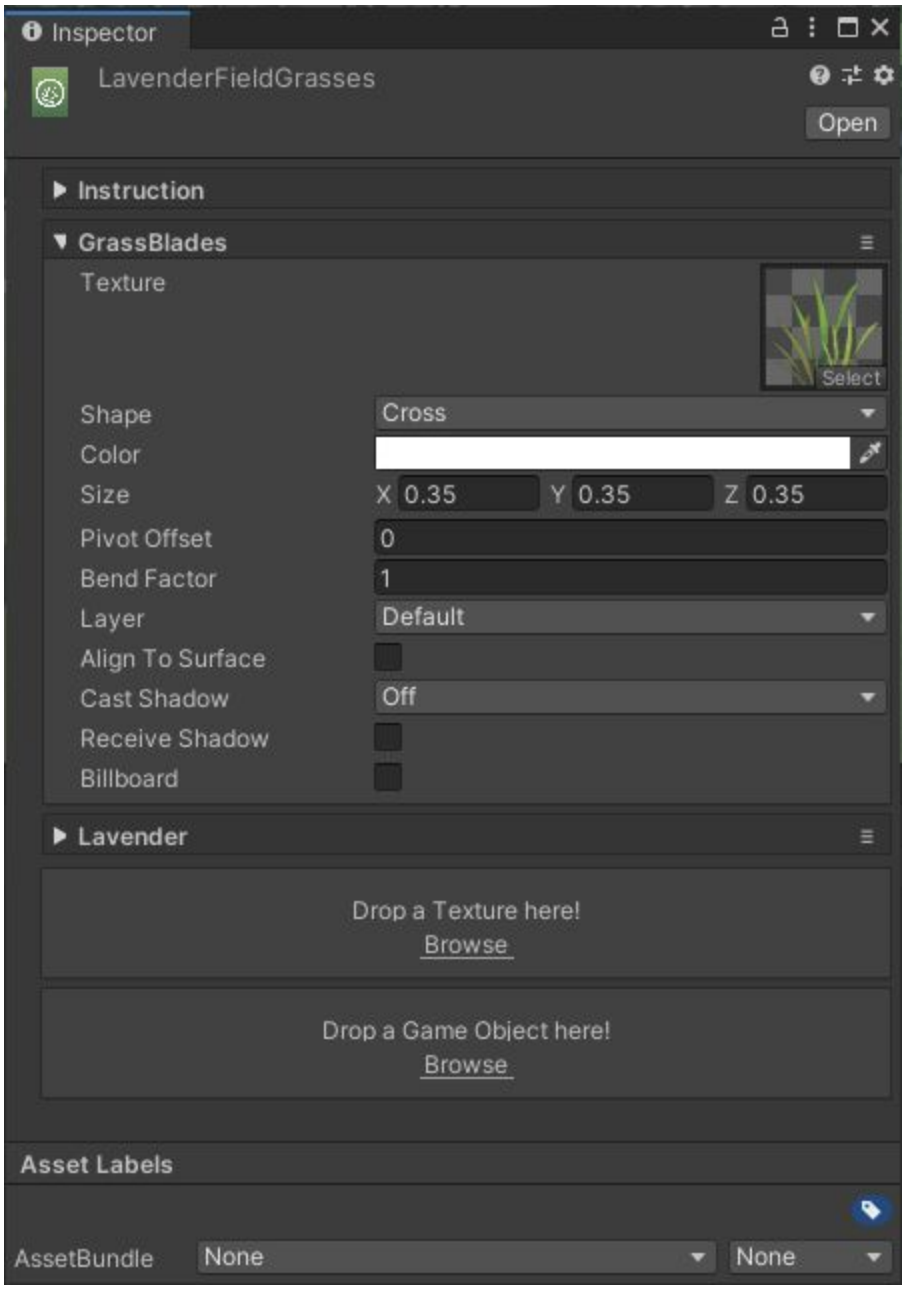


Each prototype has several settings:

- Prefab: the source prefab to copy data from.
- Billboard: the billboard asset used to render the tree as billboard. See Create Billboard Asset for more detail.
- Pivot Offset: Offset the tree position on Y-axis
- Base Rotation: rotation offset to fix mis-align issue from tree models.
- Base Scale: initial size of the tree.
- Layer: determine which camera layer to render the tree.
- Keep Prefab Layer: if check on, the tree will be rendered in the same layer as the source prefab.
- Cast Shadow/Receive Shadow: should it cast/receive shadow?
- Billboard Cast/Receive Shadow: should its billboard cast/receive shadow?

Create Grass/Detail Prototype Group

To spawn and render grasses/details on the terrain, you have to create a Grass Prototypes Group asset and assign it to the terrain. Go to **Assets>Create>Polaris>Grass Prototypes Group** and give the asset an appropriate name. Select the asset, in the Inspector, try dropping a texture/prefab into the selector box, a prototype should appear:



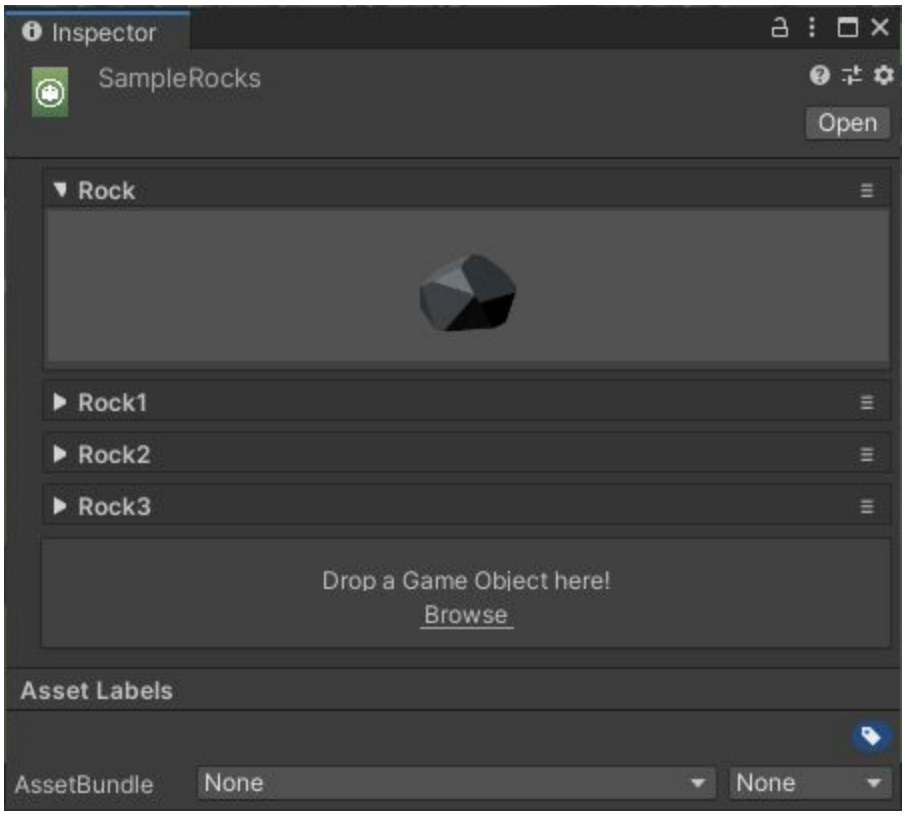
Each prototype has several settings:

- Texture: main texture represents the grass.
- Shape: determine the mesh used to draw each instance, including Quad, Cross, TriCross, Custom Mesh and Detail Object.
- Prefab: prefab game object for details.
- Mesh: mesh used in Custom Mesh shape.

- Color: Tint color for this prototype.
- Size: physical size of grass instances.
- Pivot Offset: A small position offset to move the instance up/down when batching.
- Bend Factor: how much it reacts to wind and other bending effects.
- Layer: determine which camera layer to render grass instances.
- Align To Surface: make the grass/detail instance up vector to snap with surface normal vector (need re-update to take effect).
- Cast/Receive Shadow: should it cast/receive shadow?
- Billboard: should it be rendered as a billboard? Note that Interactive Grass won't work when Billboard is enabled.

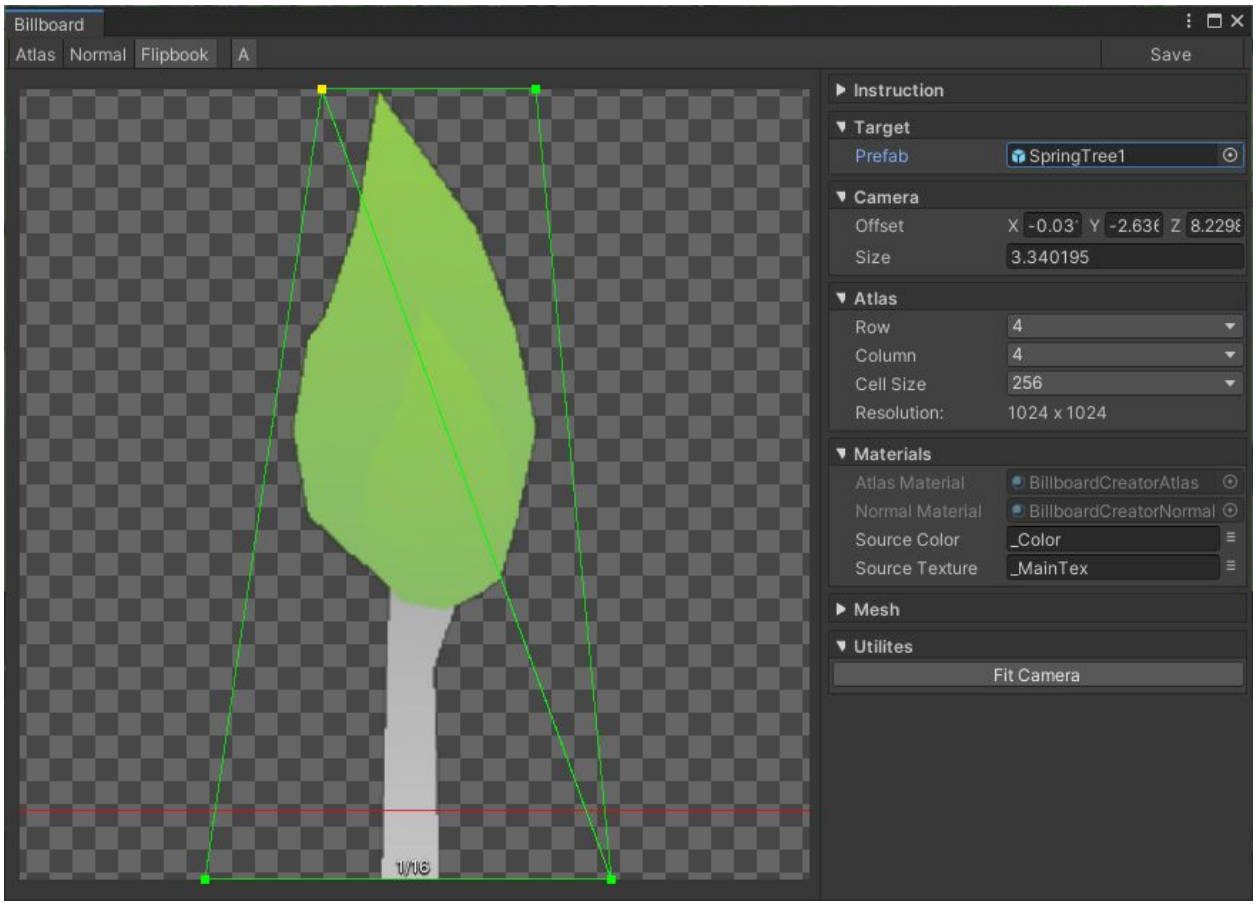
Create Prefab Prototypes Group

Prefab Prototypes Group isn't bound to any specific terrain. Instead, it is used to store prefab variations then used later with other terrain tools. Go to **Assets>Create>Polaris>Prefab Prototype Group** and give the asset an appropriate name. Select the asset, in the Inspector, try dropping a prefab into the selector box, a prototype should appear:



Create Billboard Asset

Polaris comes with a handy Billboard Asset creator to use with tree rendering. Go to **Window>Polaris>Tools>Billboard Creator** to open the editor.



The editor has 3 render modes:

- Atlas: preview billboard atlas.
- Normal: preview billboard normal map.
- Flipbook: preview billboard transition between cells and editing billboard mesh.

On the right pane you should see a section that looks like the Inspector to set properties for the billboard asset, these settings are clearly described on Unity Documentation.

Hit Save to create the billboard asset and write it to disk.

MULTI-TERRAINS WORKFLOW

Polaris design and architecture comes with the “multi-terrains” philosophy in mind, which means terrain tools should be separated from the terrain itself, and it should work seamlessly across multiple terrain without manual iteration between them. To achieve the smoothest experience, resource intensive operations such as texture modification will be done on the GPU side.

Some actions like geometry painting require the neighboring terrain to be set up correctly, so that their edges can match up and eliminate the gap between them. See this page for more details.

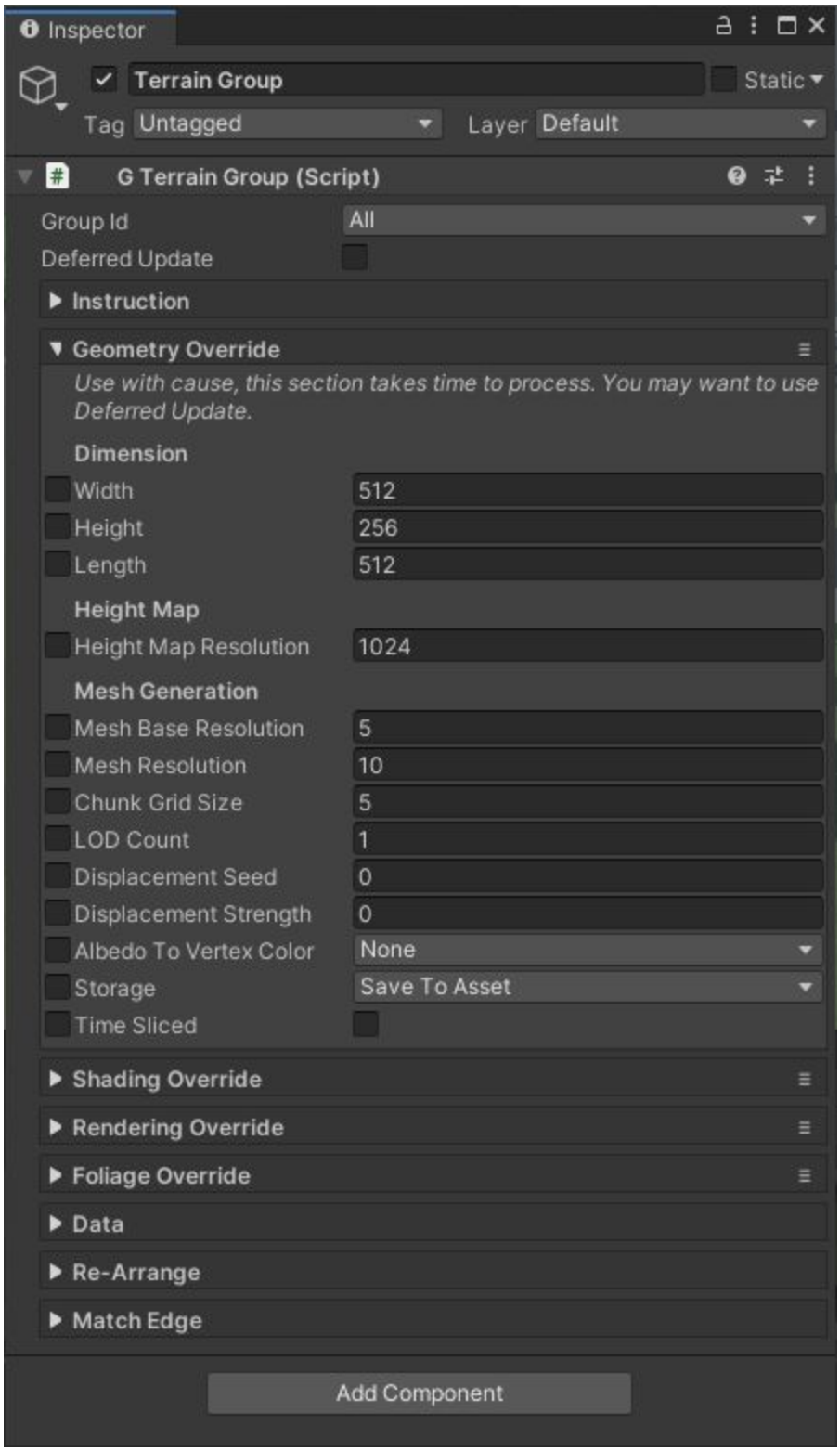
Terrains in the scene can be grouped together for different purposes, such as play area and background, or different biomes, using a simple integer as Group Id. This Id will tell terrain tools which one to apply the action on, and which one not. See this section for more information.

GROUP TOOL

Since Polaris uses multi-terrains workflow, sometimes properties must be identical across tiles for it to work correctly. It would be time consuming to iterate and change things one by one.

Group Tool provides a simple and easy way to override properties across tiles in a group, selectively, at once!

Go to **GameObject>3D Object>Polaris>Tools>Group** to add one.


Inspector
Terrain Group
Static
Tag Untagged
Layer Default
G Terrain Group (Script)
Group Id All
Deferred Update
Instruction
Geometry Override
Use with cause, this section takes time to process. You may want to use Deferred Update.
Dimension
Width 512
Height 256
Length 512
Height Map
Height Map Resolution 1024
Mesh Generation
Mesh Base Resolution 5
Mesh Resolution 10
Chunk Grid Size 5
LOD Count 1
Displacement Seed 0
Displacement Strength 0
Albedo To Vertex Color None
Storage Save To Asset
Time Sliced
Shading Override
Rendering Override
Foliage Override
Data
Re-Arrange
Match Edge
Add Component

The properties are quite similar to terrain data, with an additional checkbox on the left of each to determine whether to override that property or not.

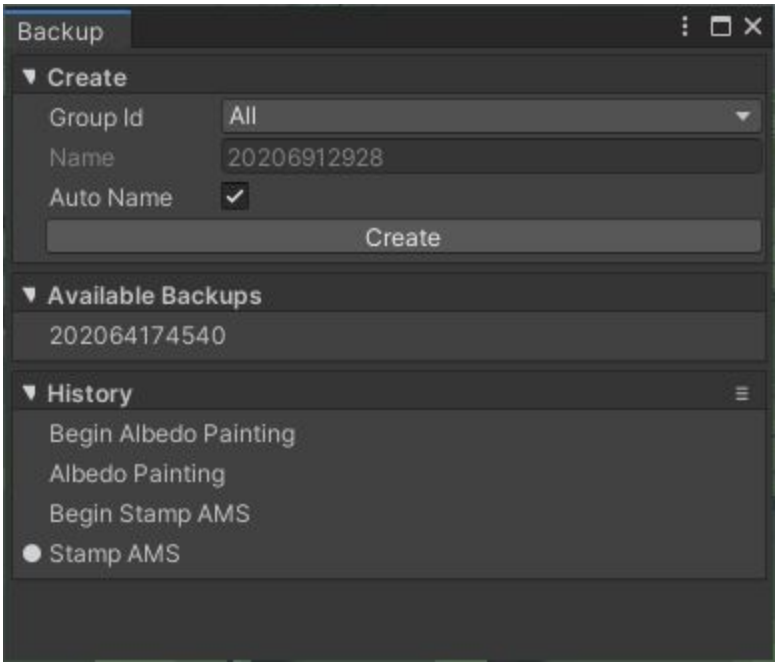
To override a property, turn on the toggle on the left of it in the Inspector, then set a value.

If you want to override many properties at once before re-generating terrains, turn on Deferred Update, override them, then click Update.

You can also perform some operations on selected terrain groups such as bulk data export/import, rearrange position or match terrain edges.

BACKUP TOOL

Backup Tool provides a robust, non-linear way to store/restore editing history. To open the editor, go to **Window>Polaris>Tools>Backup**.



There are 2 types of backup:

- Long Lives Backup: terrains in the selected group will be fully backed up and stored on disk, which mean everything from textures to foliage will be scan, and the entry remain across editor session, or transfer via network. This type is created manually by the user.
- Short Lives History: terrains in the selected group will be partially backed up and stored on disk. This type is created automatically by the tool, only the data channel affected by the tool will be backed up, and will NOT remain between editor sessions.

To restore a backup, simply click on the listed entry. For additional action like deleting the entry, right click on it. The small dot (●) indicates the last active Backup/History entry.

You can press Ctrl+Z and Ctrl+Y to perform restoring a backup like a regular Undo/Redo.

Backup data is stored in **Project Folder/GriffinBackup** directory (the same level as Assets/).

GEOMETRY & TEXTURE PAINTER

This tool is used to sculpt geometry and apply color, textures onto the terrain surface. Go to **GameObject>3D Object>Polaris>Tools>Geometry - Texture Painter** to create one.

The tool consists of several painting modes, with similar parameters and mainly processing on terrain data type of texture such as height map, splat map, etc. To select paint mode, click on the Paint Mode dropdown, or pressing F1, F2, ..., Fx key on the keyboard.

- Elevation: Raise, lower or smooth height map.
- Height Sampling: set height map value to match a pre-sampled point.
- Terrace: create steppy effect on height map.
- Remap: remap height map value using a curve.
- Noise: add small detail to height map.
- Sub Division: adding more resolution to a particular region on the surface.
- Visibility: mark a particular area on the surface as visible or not, useful when making a cave entrance.
- Albedo: paint color on Albedo Map.
- Metallic: paint on the Metallic Map R channel.

- Smoothness: paint on the Metallic Map A channel.
- Splat: paint blend weight on Splat Control Maps.
- Mask: paint on mask texture.
- Custom: other custom functionalities, defined by user code.

Each paint mode will have different actions and hotkeys to use, see the Instruction section in the Inspector for more detail.

These paint modes have some similar parameters:

- Force Update Geometry: force the terrain to update its geometry even when the painter doesn't modify its height map, use full when you want to convert texture data to mesh data, such as from albedo map color to vertex color.
- Enable History: determine whether to store a history entry for undo when painting.
- Enable Live Preview: let you see a preview of the painter effect on the terrain. This option will use more resources on your computer.
- Enable Terrain Mask: use mask texture R channel to lock particular regions of the terrain from being modified.
- Brush Mask: a texture to define the stroke shape.
- Splat: determine which splat prototype to paint. Only visible in Splat & Custom mode. See Create Splat Prototypes Group for more detail.
- Radius: radius of the brush.
- Rotation: rotation of the brush.
- Opacity: strength to apply operations.
- Scatter: randomize brush position by an offset.
- Jitter: control brush stroke randomness.
- Color: color of the brush.
- Sample Point: the pre-sample point when in Height Sampling mode.

When the Scene View is focused, you can use the minus/plus, square brackets, semicolon/quote button to gradually change brush radius, rotation and opacity, respectively.

To add a custom brush mask, simply put your texture into the **Assets/.../Resources/PolarisBrushes** directory, then disable-enable the painter component for the brushes to be reloaded.

Important: Sometime the result doesn't show up when painting, this is caused mainly by incorrect material setup, where the material doesn't use the texture which the painter is working on. For example: painting splat on a gradient lookup material doesn't show anything.

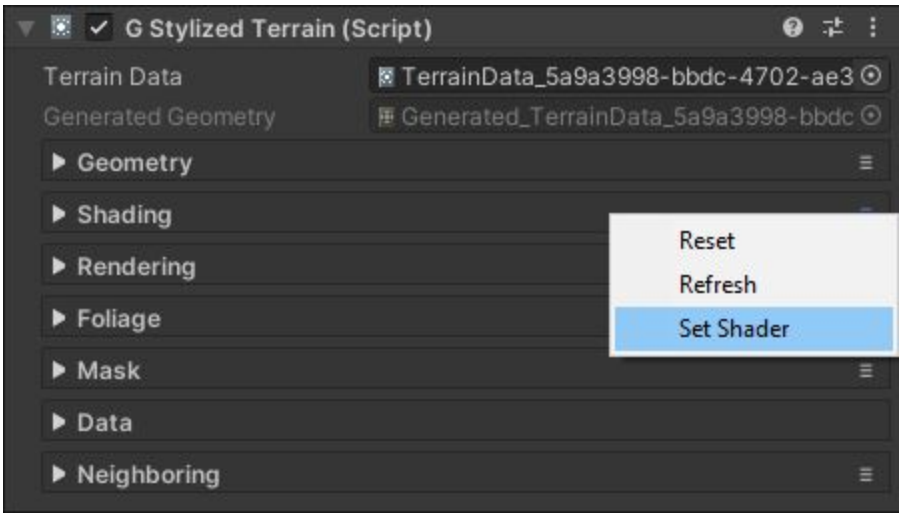
Below are the list of paint mode and the texture they work on:

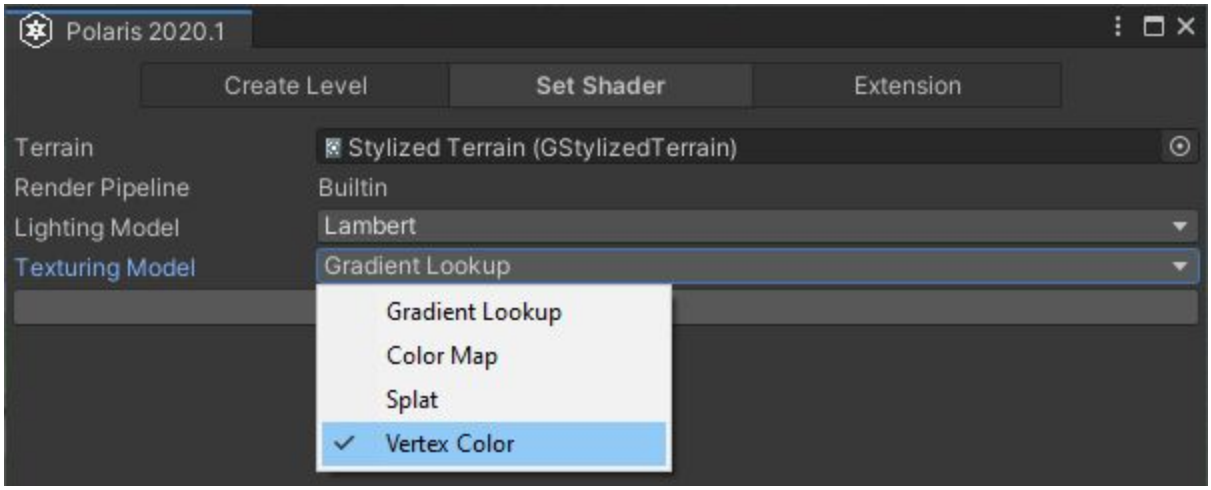
	Height Map	Albedo Map	Metallic Map	Splat Control Maps	Mask Map
Elevation	RG				
Height Sampling	RG				
Terrace	RG				
Remap	RG				
Noise	RG				
Sub Division	B				
Visibility	A				
Albedo		RGBA			
Metallic			R		
Smoothness			A		
Splats				RGBA	
Mask					RGBA

Vertex Color painting

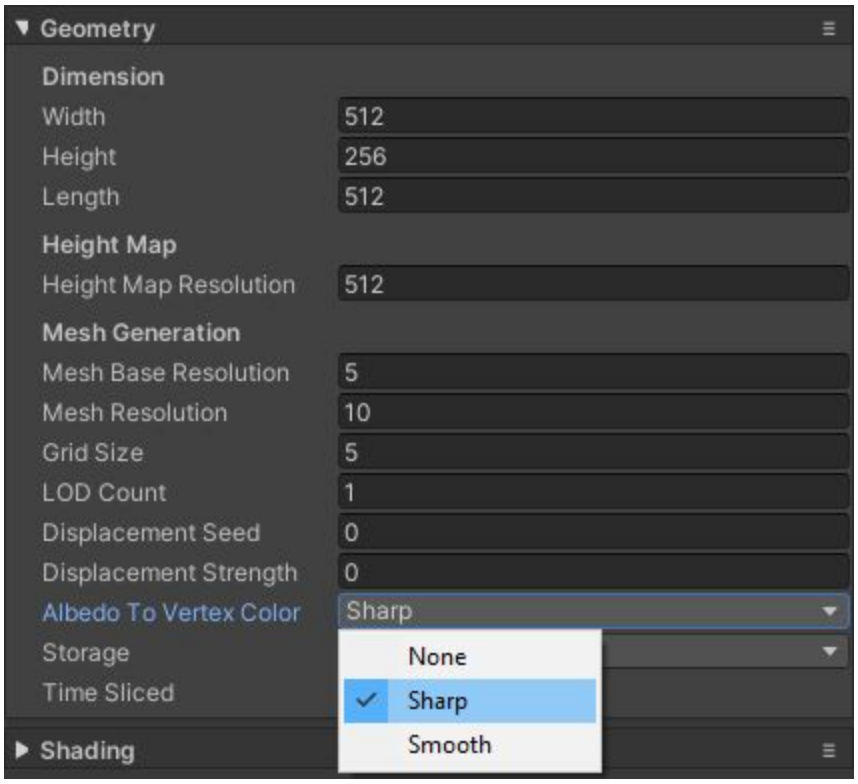
To paint vertex color, follow these steps:

1. Make sure your terrains are using Vertex Color shader variant, you can select the variant when creating them using the Wizard window. If they're using a different variant, and you want to switch to vertex color, then open the Wizard by go to terrain Inspector>Shading>CONTEXT (☰)>Set Shader. Then select Vertex Color for Texturing Model, and click Set.





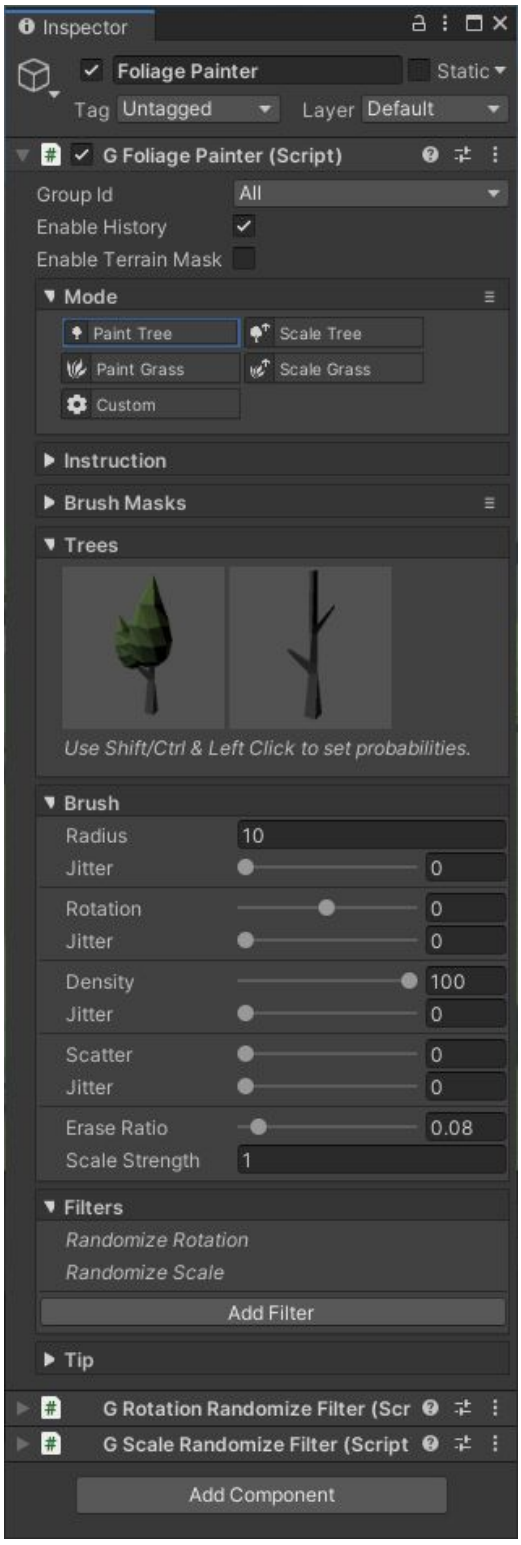
2. Under Geometry foldout, set Albedo To Vertex Color to Sharp or Smooth, depend on your art style.



3. Select the Geometry - Texture Painter, turn on **Force Update Geometry**.
4. Select **Albedo** mode and begin to paint.

FOLIAGE PAINTER

This tool is used to paint tree and grass instances onto the environment. Go to **GameObject>3D Object>Polaris>Tools>Foliage Painter** to add one.



Similar to the Geometry & Texture Painter, this tool consist of several painting modes:

- Paint Tree: paint or erase tree instances.
- Scale Tree: modify tree instances size.
- Paint Grass: paint or erase grass instances.
- Scale Grass: modify grass instances size.
- Custom: other custom functionalities, defined by user code.

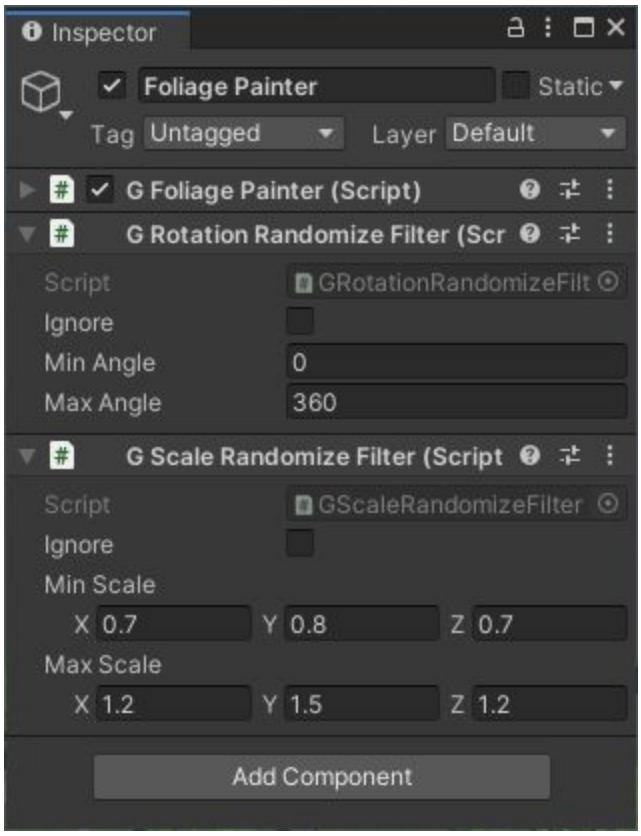
For the Trees/Grasses selector to show up, you have to create appropriate prototype group assets and assign it to the terrain. See Create Tree Prototypes Group and Create Grass Prototype Group for more detail.

Foliage Painter has some additional parameters:

- Density: determine how many instances to spawn.
- Erase Ratio: use this multiplier if you just want to snip off trees in an area, not erase them all.
- Scale Strength: strength of the brush in Scale modes.

Initially, painted instances will have default transform properties (rotation of 0 and scale of 1),

this will introduce a repetitive pattern. To add some randomness, you can use some filters. To add a filter, click on the Add Filter button and select one of them.



There are several types of filter:

- Height Constraint: prevent the instance from being spawned based on the altitude range.
- Slope Constraint: prevent the instance from being spawned based on surface angle.
- Randomize Rotation: rotate the instance randomly.
- Randomize Scale: scale the instance randomly.
- Clamp Scale: prevent the instance from being too big or too small by clamping it scale.

Randomize Rotation and Randomize Scale filters are automatically added when you create the painter.

OBJECT PAINTER

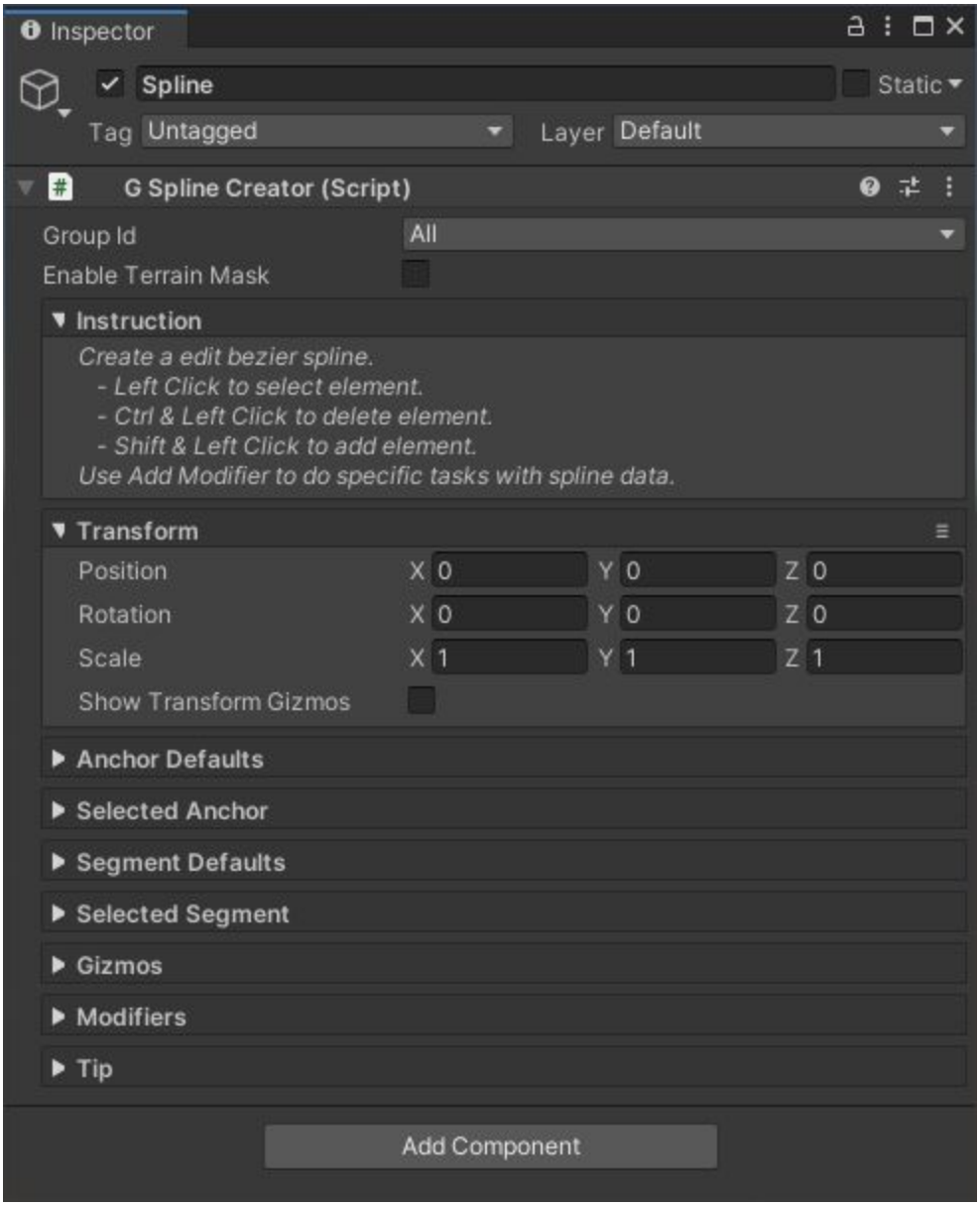
This tool is similar to Foliage Painter, but it works with prefabs instead. Go to **GameObject>3D Object>Polaris>Tools>Object Painter** to add one.

See FOLIAGE PAINTER for more detail.

SPLINE CREATOR

Spline Creator is a handy tool to perform some specific operations along a spline like making ramp, painting path or spawning trees, etc.

To add a Spline Creator into your scene, go to **GameObject>3D Object>Polaris>Tools>Spline**.

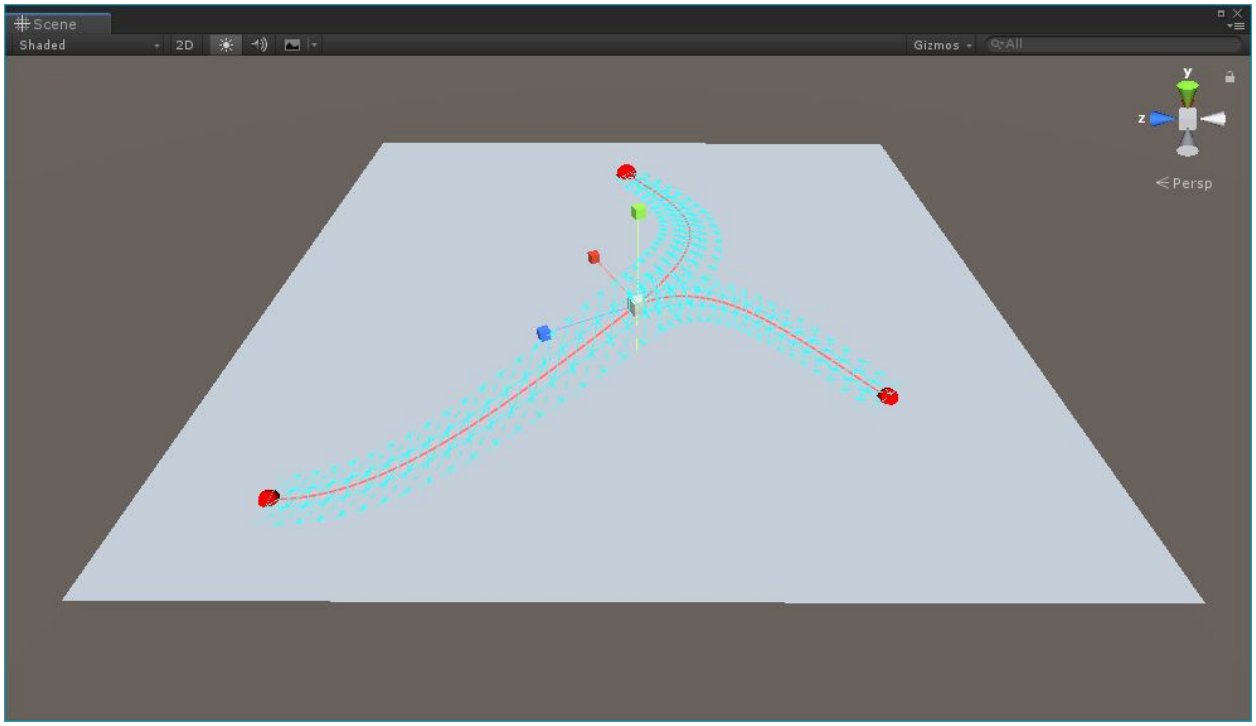


The spline holds a list of anchor points where it goes through and some segments to connect their anchors. There are some actions when working with spline:

- Left Click: select an Anchor or Segment
- Shift Click: Add a new Anchor, this anchor will be automatically connected to the selected one and may create a new branch.
- Shift Click between 2 anchors: connect 2 anchors.
- Ctrl Click: remove an Anchor or Segment.
- Ctrl Click on the dotted line: snap the Anchor onto the surface.

Each anchor has its own position, rotation and scale. You can modify these information the same way when working with Game Object (using the transform tool in the scene view, use WER to switch between modes). If you want an exact value, use the properties listed under the Selected Anchor section in the Inspector.

Spline tangents are per-segment basis, since it supports for branching. Select a segment and move its tangents to bend the spline. If you want an exact value, use the properties listed under Selected Segment section in the Inspector.



Detail for each property in the Inspector:

For Transform:

- Position/Rotation/Scale: transformation of the Spline object itself.
- Show Transform Gizmos: show gizmos to translate, rotate or resize the whole spline.

For Anchor Defaults:

- Position Offset: an offset to add to the anchor position when it is created, something like “create a new anchor at 1 meter above the ground”.
- Initial Rotation: rotation of the anchor when it is created.
- Initial Scale: scale of the anchor when it is created.

For Selected Anchor:

- Position: position of the anchor in spline local space.
- Rotation: rotation of the anchor in spline local space.
- Scale: scale of the anchor in spline local space.

For Segment Defaults:

- Smoothness: spline subdivision factor, larger numbers produce smoother spline but take more time to process.
- Width: Width of the spline where operations take full strength.
- Falloff Width: Width on the 2 sides of the spline where operations begin to fade.

For Selected Segment:

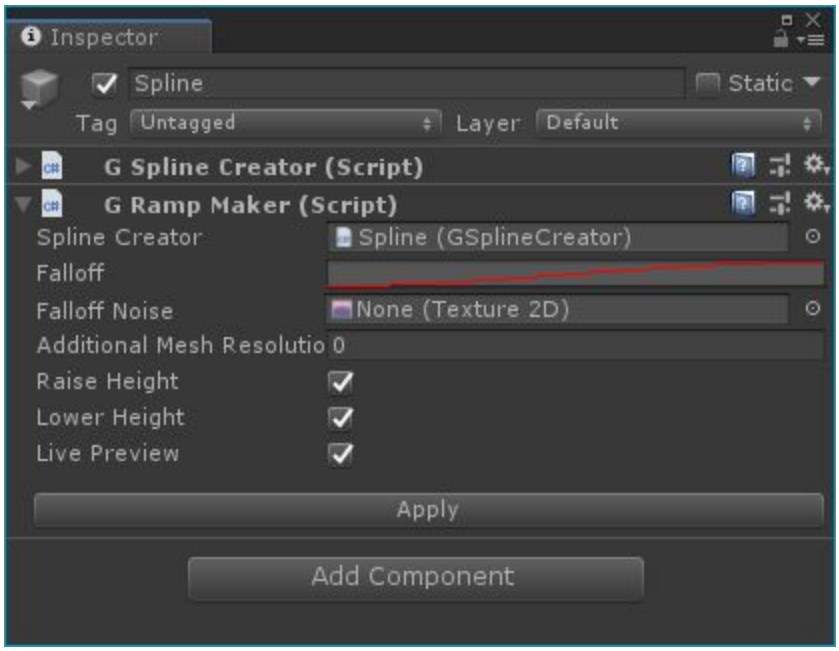
- Start Tangent: position of the first tangent in spline local space.
- End Tangent: position of the second tangent in spline local space.

For Gizmos:

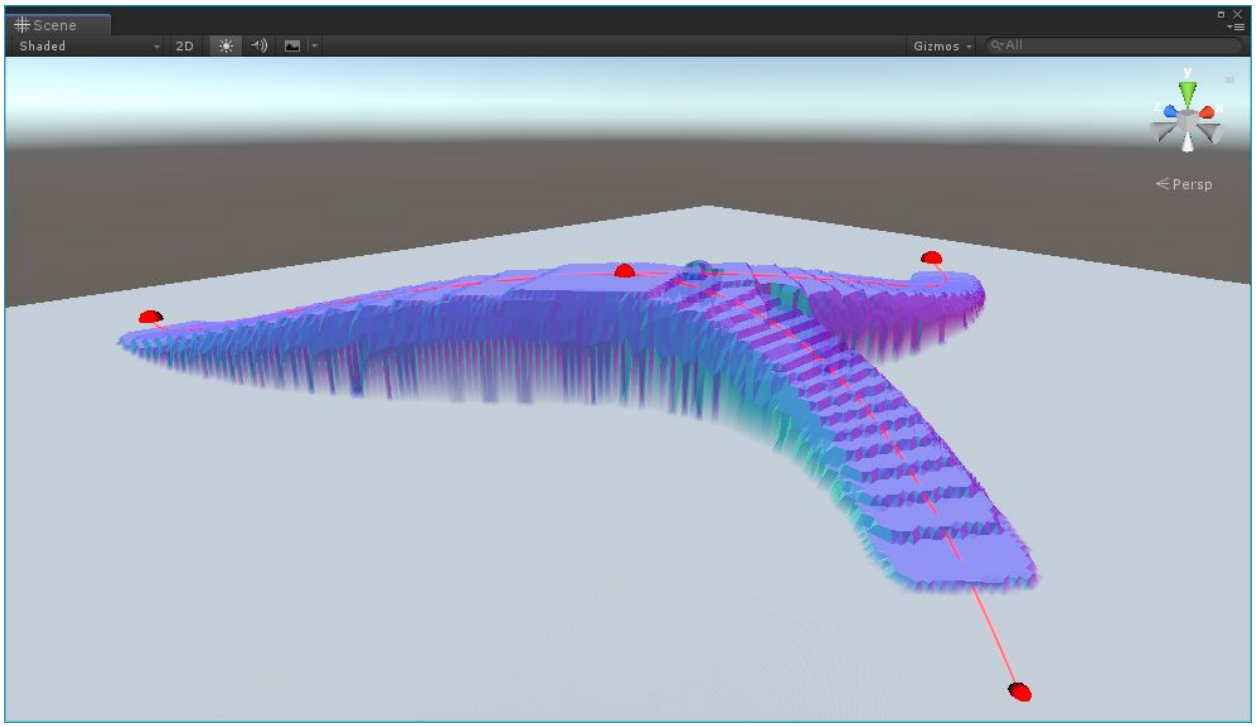
- Show Mesh: determine whether to visualize the spline mesh in the Scene view or not.

The Spline Creator is only responsible for creating base spline. To apply an operation based on that spline, you have to add a Spline Modifier, by click on Add Modifier:

- Ramp Maker: making a ramp by modifying terrain Height map.
- Path Painter: making a path by painting on Albedo or Splat map.
- Foliage Spawner: spawn trees and grasses along the path.
- Foliage Remover: clear trees and grasses along the path.
- Object Spawner: spawn prefab instances along the path.
- Object Remover: remove prefab instances along the path.



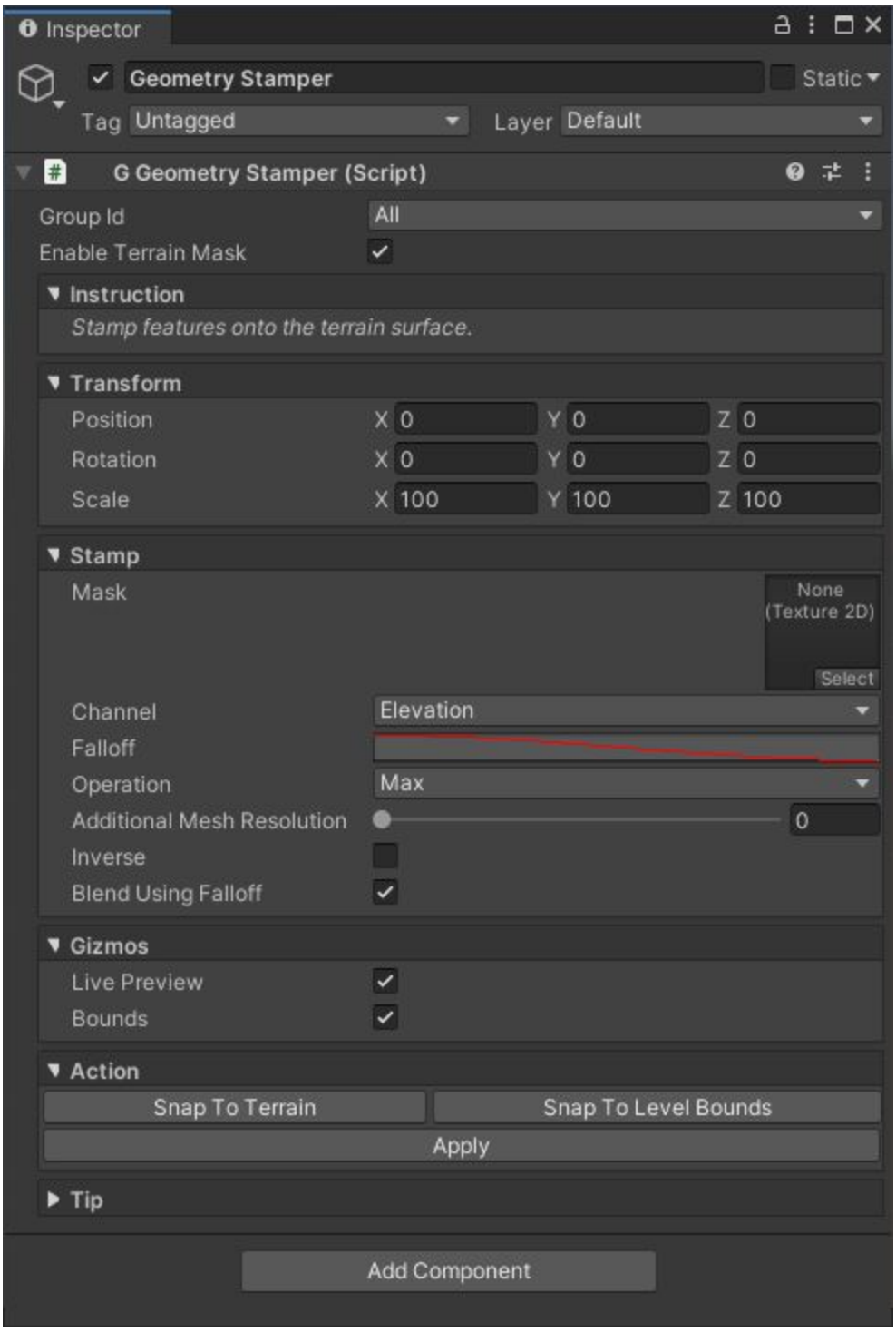
A special thing about this tool is that it allows you to see the result before applying to the terrain, minimizing the trial-and-error process.



Explore the parameters to see how it works. Once you're happy with it, hit Apply and done!

GEOMETRY STAMPER

This tool stamp features onto terrain geometry, using some special math operations to blend the result. To create a Geometry Stamper, go to **GameObject>3D Object>Polaris>Tools>Geometry Stamper**.



Detail of each property of a Geometry Stamper:

For Transform:

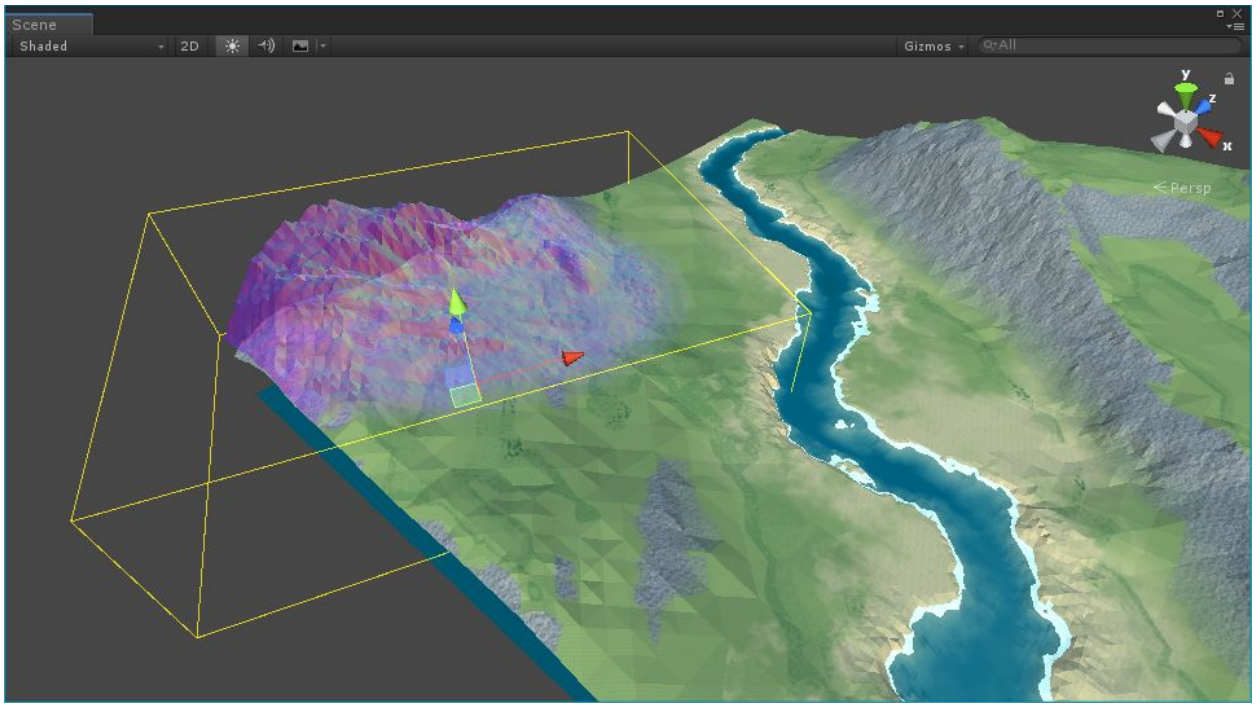
- Position: position of the stamper in the scene.
- Rotation: rotation of the stamper.
- Scale: size of the stamper.

For Stamp:

- Mask: a mask texture represents the geometry feature to stamp.
- Channel: which channel of the height map to stamp on (elevation or visibility).
- Falloff: fade the stamp mask from the center.
- Operation: the math operation to apply.
- Additional Mesh Resolution: increase the level of subdivision at stamp position.
- Inverse: inverse the stamp mask color.
- Lerp Factor: linear interpolation factor, only available in Lerp operation.
- Blend Using Falloff: use Falloff curve to blend result with original terrain, to have smoother transitions.

For Gizmos:

- Live Preview: whether to draw live preview in the Scene.
- Bounds: whether to draw stamper bounding box in the Scene.



In the Scene view, you can use the transform tool to move, rotate and scale the stamper, or use WER to switch between modes. Depend on what you want to make, choose an appropriate operation:

Let the result is "**h"**, current height is "**a"** and stamper height is "**b"**:

- Add: Additive blending, use this mode to raise terrain up. The result is not less than current height. The formula is something like this: h = max(a, a+b)
- Subtract: Subtract stamper height from current height. Use this mode to lower terrain. The result is not larger than current height. The formula looks like this: h = min(a, a-b)
- Reverse Subtract: Subtract current height from stamper height. The formula looks like this: h = min(b, b-a)
- Max: Take the larger height. The formula looks like this: h = max(a,b)
- Min: Take the smaller height. The formula looks like this: h = min(a,b)
- Lerp: Linear interpolation from current height to stamper height based on Lerp Factor. The formula looks like this: h = lerp(a, b, Lerp Factor)
- Difference: Take the difference between current height and stamper height. The formula looks like this: h = absolute(a - b)

Play around with different settings, once you are happy with the preview, hit Apply and done!

TEXTURE STAMPER

Instead of painting by hand, Texture Stamper lets you perform procedural texturing on a terrain surface. To add a Texture Stamper, go to **GameObject>3D Object>Polaris>Tools> Texture Stampers**.

Texture Stamper also has basic properties as described in Geometry Stamper to define the shape.

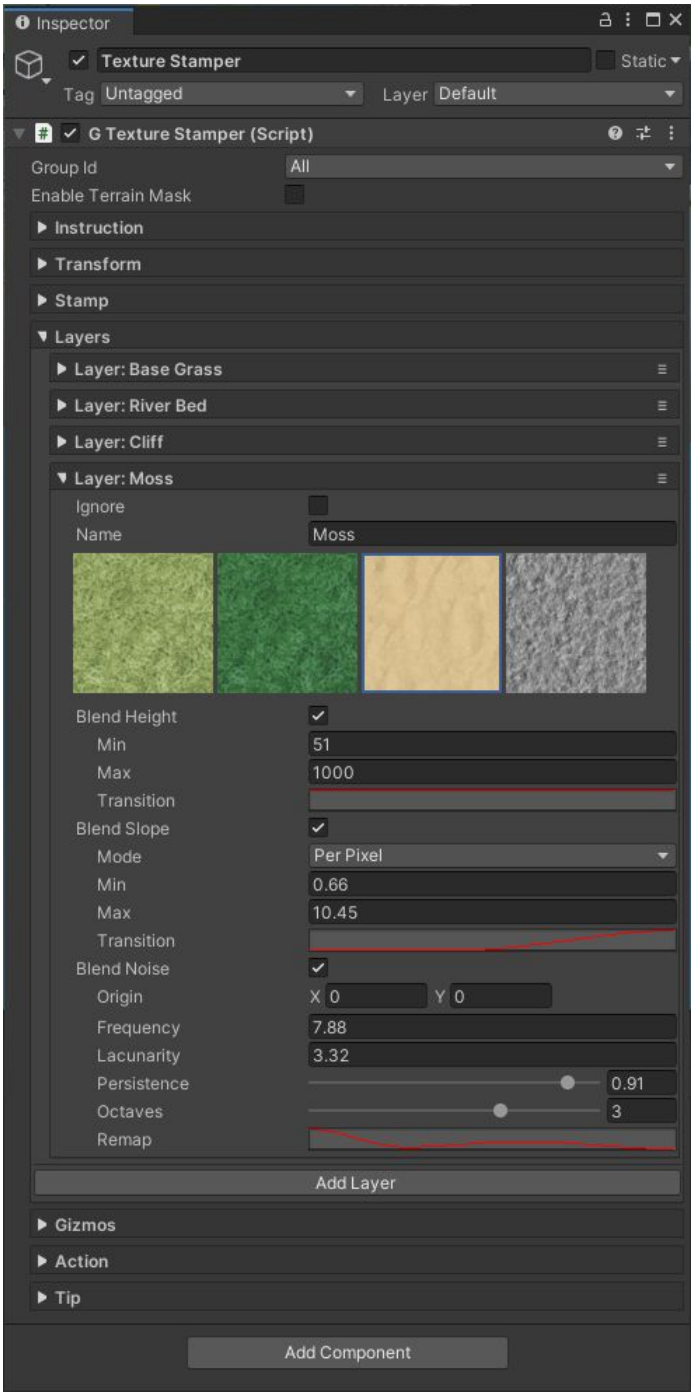
The stamper allows you to stamp on one of the two channels:

- Albedo Metallic Smoothness
- Splat

In addition, it consists of multiple stamp layers, which will be applied one by one onto the surface. To add a stamp layer, click on Add Layer.

Each stamp layer has its own properties:

- Ignore: skip the current layer.
- Name: name of the layer, for better managing.
- Color: color to apply, only available in Albedo Metallic Smoothness mode.
- Metallic: metallic value to apply, only available in Albedo Metallic Smoothness mode.



- Smoothness: smoothness value to apply, only available in Albedo Metallic Smoothness mode.
- Splat: splat texture to apply, only available in Splat mode.
- Blend Height: determine whether to use height based blending or not.
- Height Min/ Height Max/ Height Transition: define the region which satisfies height condition.
- Blend Slope: determine whether to use slope based blending or not.
- Slope Mode: choose between Sharp, Interpolated, Per-Pixel normal map
- Slope Min/ Slope Max/ Slope Transition: define the region which satisfies slope condition.
- Blend Noise: determine whether to generate some Perlin noise to add some randomness or not.
- Origin/ Frequency/ Lacunarity/ Persistence/ Octaves: basic noise properties, you can find a bunch of explanations of them on the Internet.
- Remap: remap noise value to create more interesting effects.

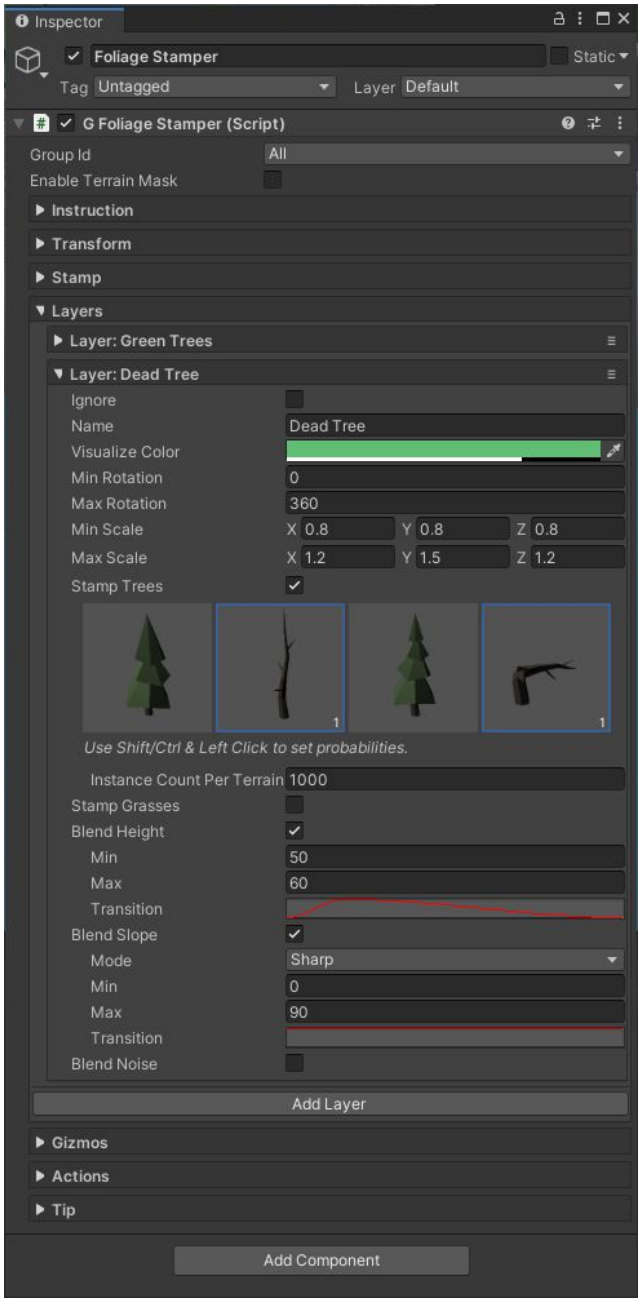
You can have more than one layer, play around with them, once you're happy with the result, hit Apply and done.

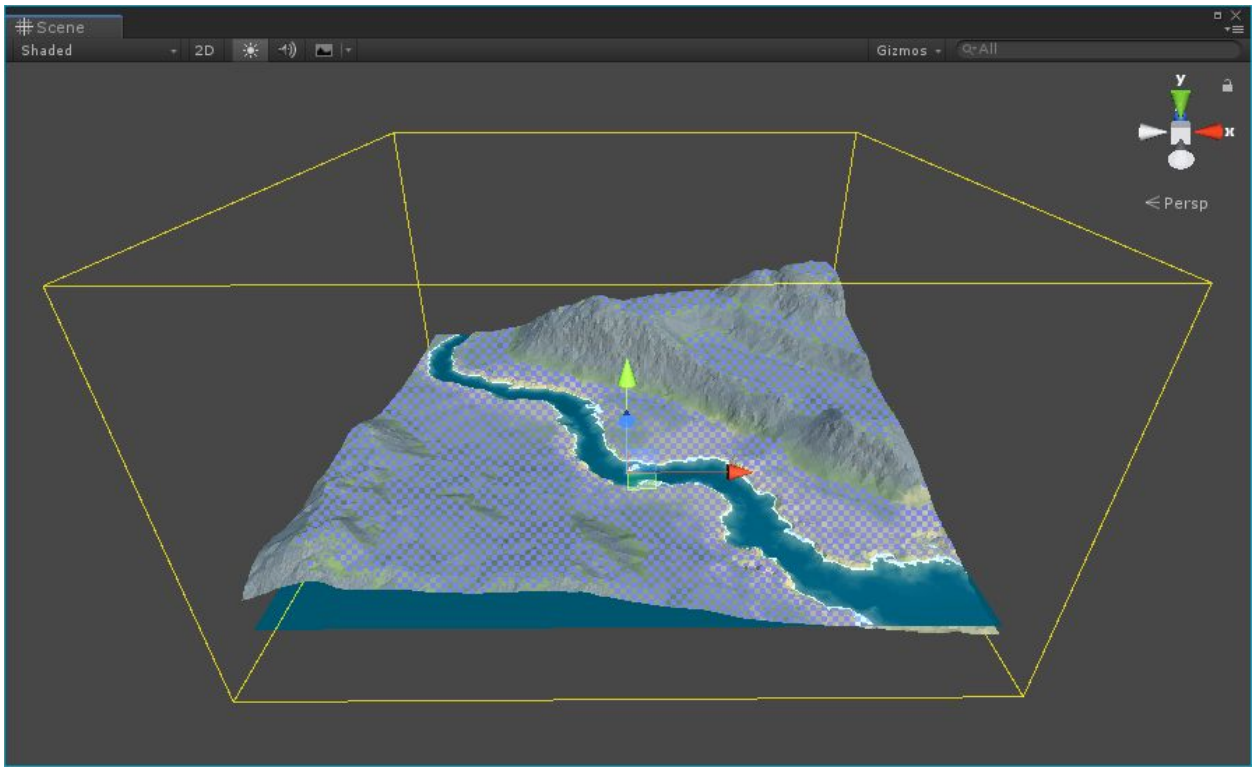
FOLIAGE STAMPER

Similar to Texture Stamper, Foliage Stamper let you perform procedural foliage placement instead of painting by hand. To add one, go to **GameObject>3D Object>Polaris>Tools>Foliage Stampers**.

Each stamp layer has some additional properties:

- Visualize Color: a color to visualize its mask on the terrain surface.
- Instance Count Per Terrain: the number of instances to spawn.
- Min Rotation/Max Rotation: rotation range of each instance being spawned.
- Min Scale/Max Scale: scale range of each instance being spawned.
- Stamp Trees: determine whether to spawn trees or not.
- Tree Index: determine which tree to spawn.
- Stamp Grasses: determine whether to spawn grasses or not.
- Grass Index: determine which grass to spawn.





Play around with these properties until you're happy with it, then hit Apply to stamp. You can hit Apply multiple times to add even more instances into the scene.

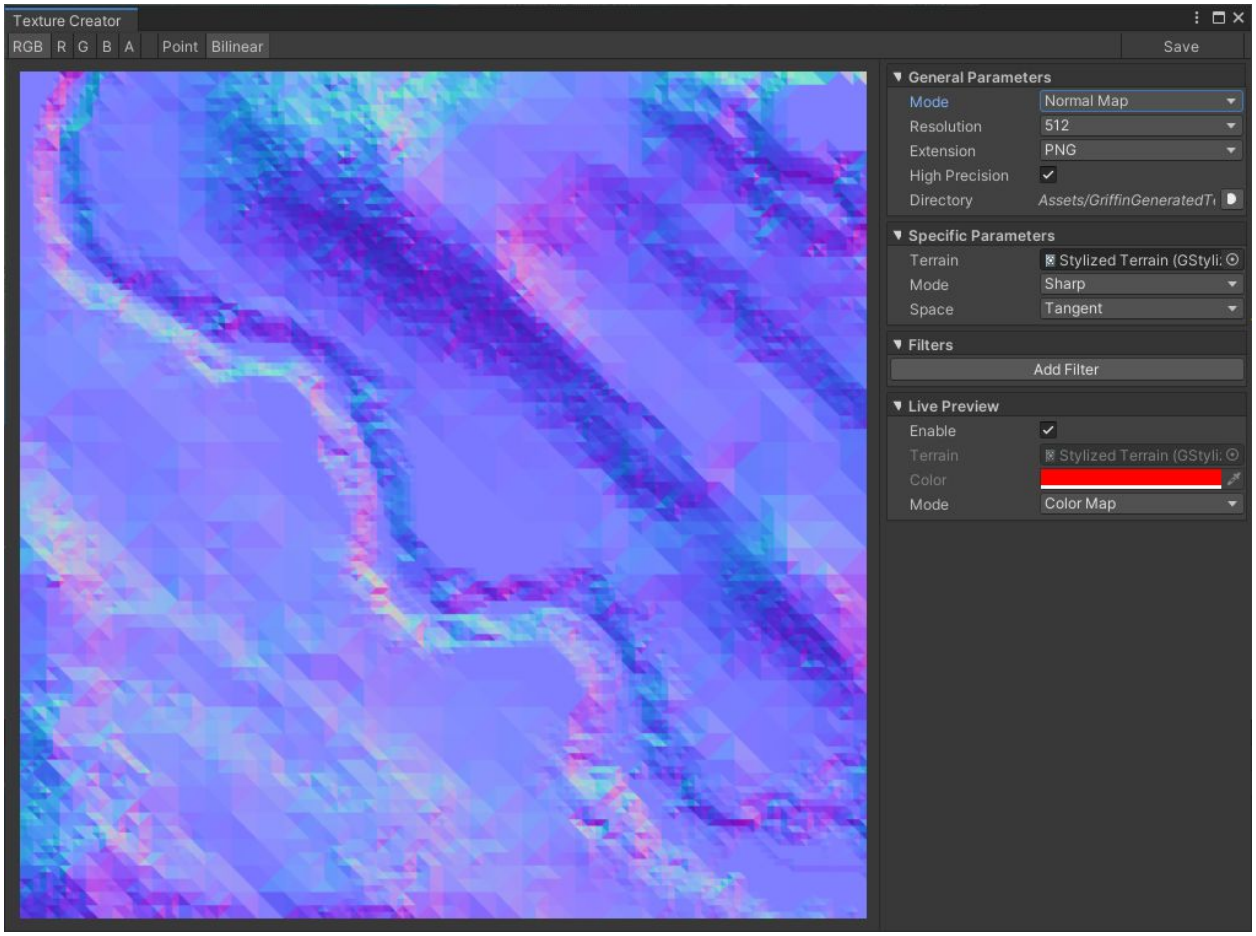
OBJECT STAMPER

This tool is similar to Foliage Stamper, but it works with prefab instead. To add one, go to **GameObject>3D Object>Polaris>Tools>Object Stamper**.

See FOLIAGE STAMPER for more detail.

TEXTURE CREATOR

Texture Creator is an intuitive way to create and author terrain textures, as well as advanced masking. Go to **Window>Polaris>Tools>Texture Creator** to open the window.



Toolbar

The toolbar is on top of the window, where you can toggle between view mode and save the texture to asset.

- RGB/R/G/B/A: Toggle between full color or per channel view.
- Point/Bilinear: Toggle between point or bilinear filter mode.
- Save: Save the texture to asset.

General Parameters

This section determines generation mode and some properties which are shared between modes.

- Mode: Which kind of texture to generate, this will be described more later.
- Resolution: Size of the texture.
- Extension: Image file extension to save to asset (PNG, JPG, EXR, TGA)
- High Precision: Determine whether to use floating point (HDR) texture or not, should be enabled for generating height map, mask and normal map, which contains non-color data.
- Directory: The location to save the image.

Specific Parameters

This section displays properties which are specific to each generation mode.

Height Map

Extract elevation data from a terrain.

- Terrain: The terrain to extract its height map, elevation (RG channel) only, sub-division and visibility are ignored.
- Real Geometry: Check to extract a sharp edged height map.

Height Map From Mesh

Generate a height map using custom mesh, useful for stamper mask, or third-parties' terrain meshes.

- Mesh: The mesh to extract from.
- Offset: Position of the mesh related to the camera.
- Rotation: Rotation of the mesh.
- Scale: Scale of the mesh.

- Depth: Control the strength of the grayscale range.
- Fit Camera: fit the mesh to the texture automatically.

Normal Map

Extract a normal map from terrain geometry or height map.

- Terrain: The terrain to extract.
- Mode: Choose between Sharp, Interpolated or Per-Pixel normal map.
- Space: Choose between Local or Tangent space.

Steepness Map

Extract slope steepness data from a terrain.

- Terrain: The terrain to extract.
- Mode: Choose between Sharp, Interpolated or Per-Pixel normal vectors.

Noise Map

Generate a noise map, useful for adding fine detail to your terrain or use as a height map.

- Type: The noise type, Perlin, Ridged, Voronoi, etc.
- Origin: Offset the noise coordinate.
- Frequency: Determine size of noise detail, higher frequency produce smaller detail.
- Lacunarity: Determine the increment of frequency of each layer/octave.
- Persistence: Determine the decrement of amplitude of each layer/octave.
- Octaves: Number of noise layers added on top of each other.
- Seed: A random seed.

Color Map

Generate a color map from splat data, as known as Base Map, or Mega Texture.

- Terrain: The terrain to extract.

Blend Map

Perform some basic texture blending operation for masking. Click on Add Layer to add a new blend layer. For each layer:

- Source: Blend data source, from a Texture, or a constant Number, or a Vector.
- Texture: The texture to blend.
- Number: The constant number to blend.
- Vector: The vector to blend.
- Operation: The operation to apply.
- Factor: The factor to interpolate between 2 layers, in Lerp mode.
- Mask: The mask to interpolate between 2 layers, in Lerp mode.

Foliage Distribution Map

Extract a distribution map from trees and grasses.

- Terrain: The terrain to extract.
- Brush: Brush texture to paint at each location.
- Opacity: Brush opacity.
- Size: Size of the brush, in normalized space.
- Rotation Min/Max: Rotation of the brush.
- Trees: Tree prototypes to extract.
- Grasses: Grass prototypes to extract.

Filters

This section contains a stack of image filters for manipulating the generated textures. The stack is applied top to bottom, each layer can be ignored by uncheck the Enable box. Click Add Filter to add a layer.

Curve

Remap color data using curves.

- Master: The master curve, apply after remapping each channel.
- Red/Green/Blue/Alpha: Remap value on each channel.

Blur

Apply a Gaussian blur effect. This filter is expensive.

- Radius: The blur radius.

Invert

Invert the color on each channel (1-value).

- Red/Green/Blue/Alpha: Invert each channel, selectively.

Step

Apply a value stepping/snapping effect, useful for creating a terraced heightmap, for example.

- Count: The number of steps.

Warp

Apply a normal map based image warping.

- Mask: The normal map to use.
- Is Normal Map: Is the mask already a normal map? Uncheck to use a grayscale image.
- Strength: Strength of the effect.

Live Preview

This section controls Live Preview in the scene view.

- Enable: Toggle Live Preview on/off.
- Terrain: The terrain to apply the texture on, will be picked automatically in some modes.
- Color: Color of the mask in Mask mode.

- Mode: Render mode, Mask, Color Map or Geometry.

Texture Importing

Saved textures are imported using the default importer, which is considered as a color texture. For specific usage, such as normal map or height map, you have to adjust the importer manually.

The recommended format for height maps and masks is **R16**.

PREPARE FOR BAKING

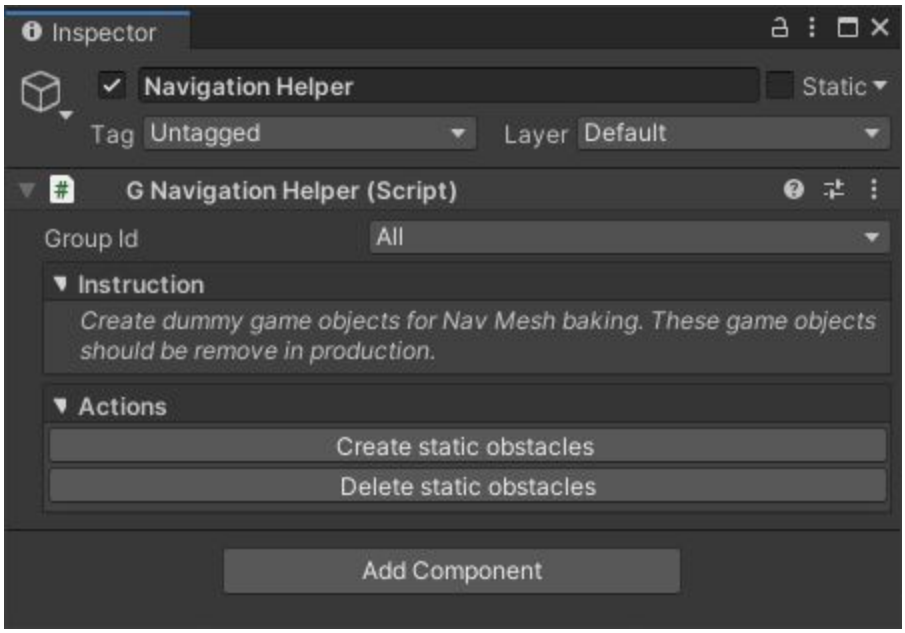
Mark geometry as static

Before performing baking in the editor (Lightmap baking for example), you have to mark terrain geometry as static. However, game objects which contain geometry mesh are hidden from the Hierarchy to prevent clustering, and **you should NOT mark the terrain game object as static**, since it also contains other kinds of components like Tree Collider.

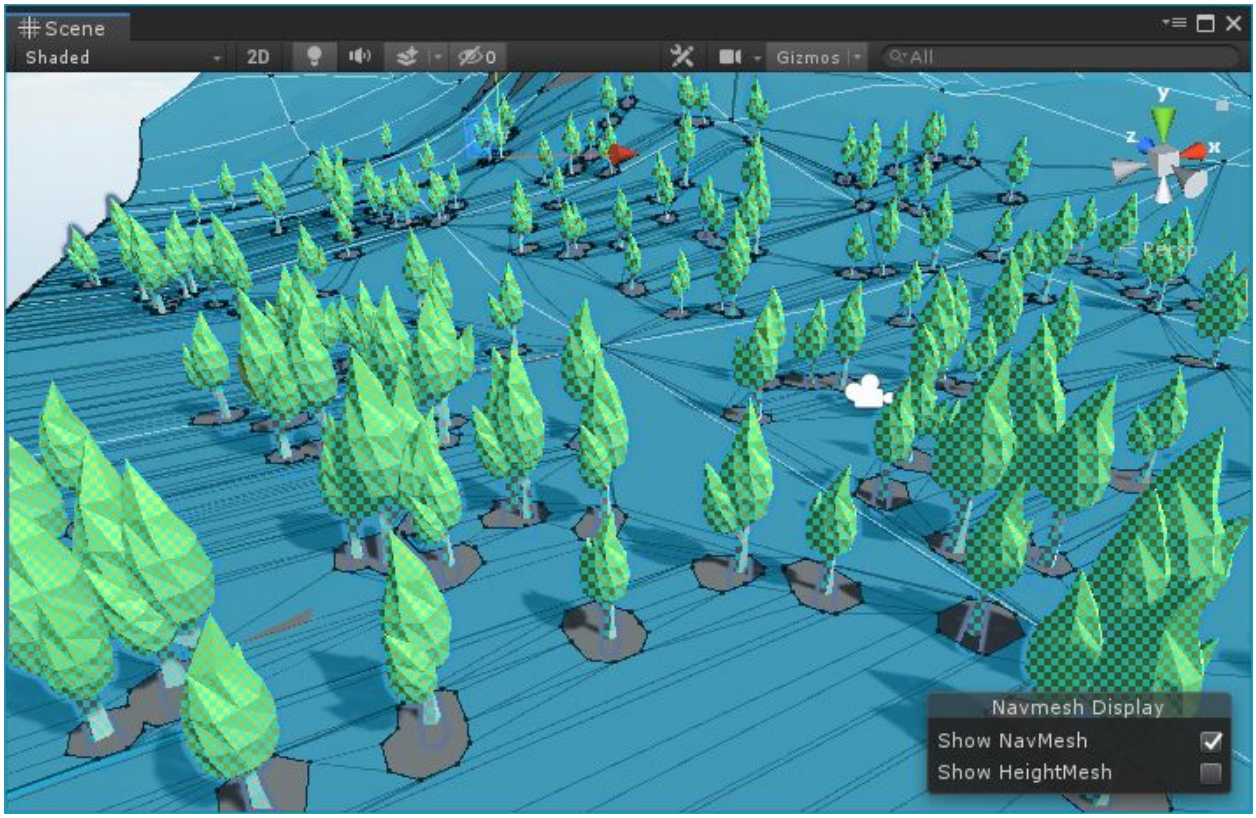
To properly mark geometry as static, go to **Window>Polaris>Project>Settings**, then check on **Show Geometry Chunks In Hierarchy**. A child game object named **~Geometry** will appear, **mark it as static then select "Yes, change children".**

Nav Mesh baking

Terrain geometry is automatically marked as Navigation Static so it can be grabbed by Unity Navigation System. However, trees rendered with Polaris are not game objects, so Unity don't know how to carve holes on the nav mesh for more precise results. Luckily, Polaris provides a helper tool to help you deal with this. Go to **GameObject>3D Object>Polaris>Tools>Navigation Helper** to create it.



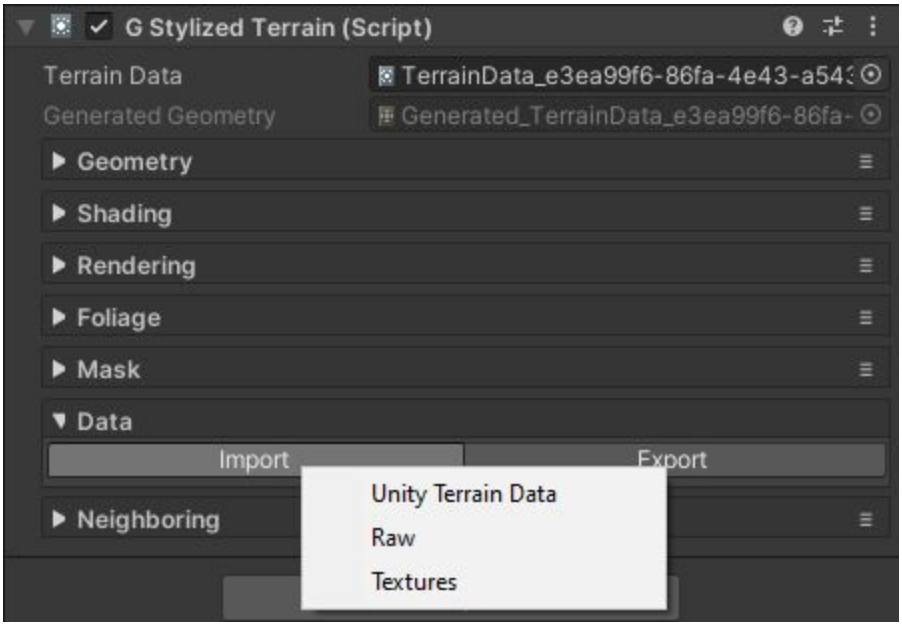
Before baking nav mesh, click on **Create static obstacles**, it will create some dummy game objects where your trees are, so Unity knows how to carve holes on the nav mesh.



Once you’re satisfied with the result, remember to click on **Delete static obstacles** to delete those dummy game objects.

IMPORT AND EXPORT DATA

You can import/export data by selecting a terrain, expand its Data section, click on Import/Export button and select an appropriate data type:



Currently support:

Import:

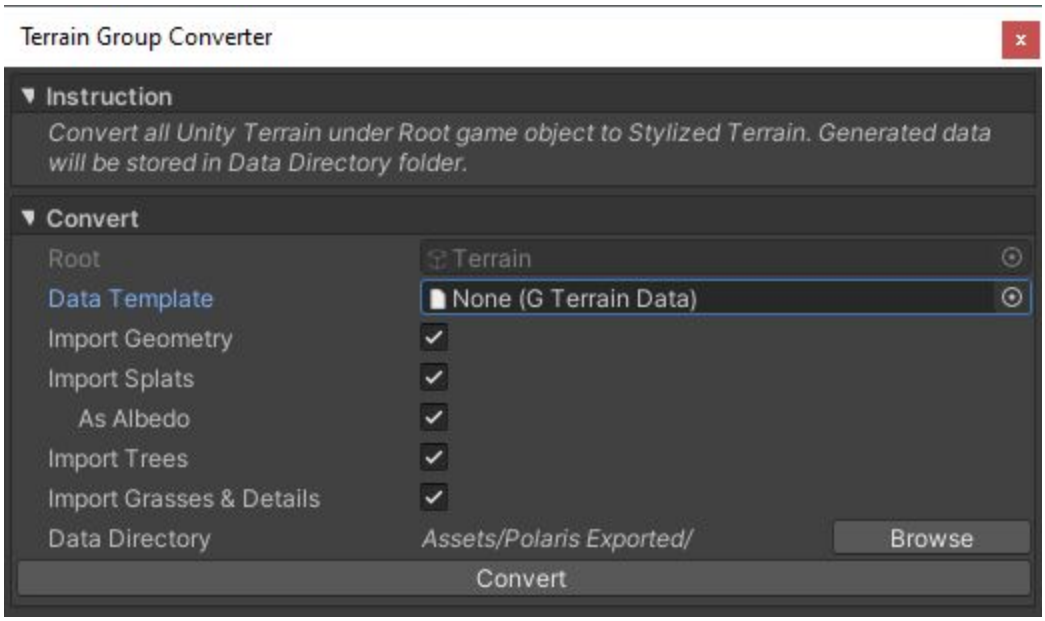
- Unity Terrain Data: easier to migrate from other tools like Gaia, MapMagic and TerrainComposer.
- Raw: import height map from .raw and .r16 files, generally created with software like World Machine.
- Textures: import textures to use as its own internal data for painting and stamping, etc.

Export:

- Unity Terrain Data: for further polishing using other terrain tools.
- Raw: export height map to .raw, .r16 files.
- Textures: export copies of its textures.

CONVERT UNITY TERRAIN TO LOW POLY

Polaris provides a handy tool to help you to easier migrate from 3rd parties' tools like Gaia, MapMagic and TerrainComposer by converting a Unity terrain group to low poly. Right click on the root game object of the terrain group, select **3D Object>Polaris>Convert From Unity Terrain**:



Select what you want to import and where to store converted data, then hit Convert.

You can assign Polaris Terrain Data as a template for converted terrain, where you can set custom properties such as texture resolution, vertex color mode, etc. instead of taking default value.

Note: many files will be created during the converting process, depending on how many terrains in the group and their complexity, it's better to select a new, empty folder for storing data before converting.

UNIVERSAL RENDER PIPELINE UPGRADE GUIDE

Polaris also supports the Universal Render Pipeline. There is something you have to know when working with it. Note that render pipelines other than the built-in and URP is not supported. You have to write your own shaders and upgrade it manually.

Install URP

To work with URP, you need to import the following package using Package Manager:

- Universal RP (com.unity.render-pipelines.universal)

Open the Package Manager to include them. After that, go to **Window>Polaris>Project>Update Dependencies** to re-configure project.

Install URP Support Extension

Materials and Shaders for URP are kept inside a package to avoid compilation errors.

The package will be automatically extracted and installed when you open the Wizard window.

If you're using URP (URP Asset assigned in Project Settings), it will try to upgrade material for all terrains in your project. You can also open the Extension window to do this step later.

Creating new terrain

The Wizard automatically detects the render pipeline currently in use and picks an appropriate material for you.

Upgrade previously created terrain

For terrains that were previously created in built-in render pipeline, you can use the Wizard to re-configure its material. In the Inspector, click on **Shading>CONTEXT>Set Shader** to do so,

Upgrade Grass and Tree Billboard materials

Grass material is automatically picked up depending on your render pipeline, so nothing to worry about.

For new billboard assets, the corresponding material will be automatically configured to fit your render pipeline.

For previously created billboard material, you have to switch to **Griffin>Universal RP>Foliage>TreeBillboard** shader.

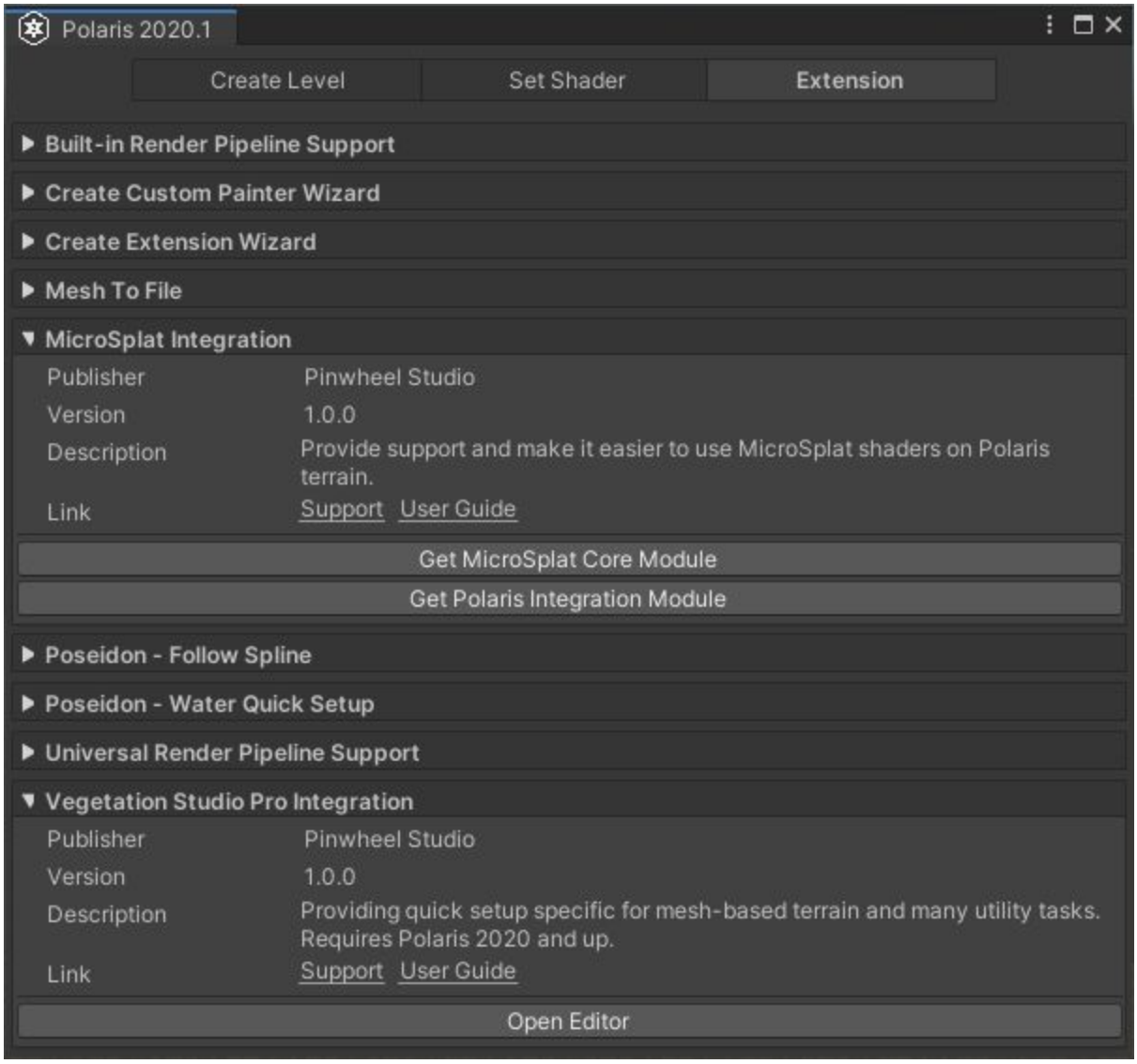
Upgrade art asset materials

For other kinds of art assets such as trees purchased on the Asset Store, you have to switch to an appropriate shader provided under **Universal Render Pipeline>...** shader group.

Polaris can’t handle any issues with these assets, please contact their publisher directly for better support.

EXTENSION SYSTEM

Polaris provides a system that lets you write code to add extent/utility tasks, integrate your asset, etc. and display information in a hub. Go to **Window>Polaris>Tools>Extension** to open it.



These extensions can be provided by many publishers, which requires you to import their assets before use.

Verified extensions

There are some available extension provided by Pinwheel Studio, integrated with our product:

Polaris

- Universal Render Pipeline Support: adding materials and shaders for URP.

CSharp Wizard

- Create Custom Painter Wizard: generate skeleton code for making a custom terrain painter.
- Create Extension Wizard: generate skeleton code for an extension entry.

Poseidon

- Poseidon - Water Quick Setup: quickly add low poly water using presets.
- Poseidon - Follow Spline: add water tiles follow a spline.

Mesh To File

- Terrain To File: export terrain geometry to Obj or Fbx file.

There are also some extensions to integrate with other publishers' product:

MicroSplat Integration: advanced terrain shader with high fidelity and more artistic control. Requires MicroSplat Core Module and MicroSplat Polaris Integration Module.

Vegetation Studio Pro Integration: advanced procedural foliage placement and rendering. Requires Vegetation Studio Pro and Vegetation Studio Pro Integration Extension.

Writing your own extension

To be detected as an extension, your script must be placed in a namespace which ends with ".GriffinExtension", the recommended pattern is <Company>.<Package>.GriffinExtension.

To avoid compilation error, you should wrap the entire code file inside a **#if GRIFFIN && UNITY_EDITOR** and **#endif** block, which means it only gets compiled when Griffin (Polaris internal name) is imported and the user is working with the editor. For example:

```
#if GRIFFIN && UNITY_EDITOR
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using UnityEditor;

namespace Pinwheel.Griffin.GriffinExtension
{
    public static class ExampleExtension
    {
…
    }
}
#endif
```

You also have to define some mandatory functions for the extension to work, they are:

```
public static string GetPublisherName() {…}

public static string GetExtensionName() {…}

public static void OpenSupportLink() {…}
```

These functions must have exactly the same name and signature as above, and return the appropriate value to display in the extension info section. If you fail to define them, the extension will not be detected!

There are some optional functions to provide additional information:

```
public static string GetDescription() {…}
```

```
public static string GetVersion() {…}

public static void OpenUserGuide() {…}
```

The Extension System provide a default GUI for every extension, which display a vertical list of button corresponding to each function in the script, to expose your function to the GUI, append the function name with “Button_” prefix, and it should be static, have void return type and zero parameter. For example:

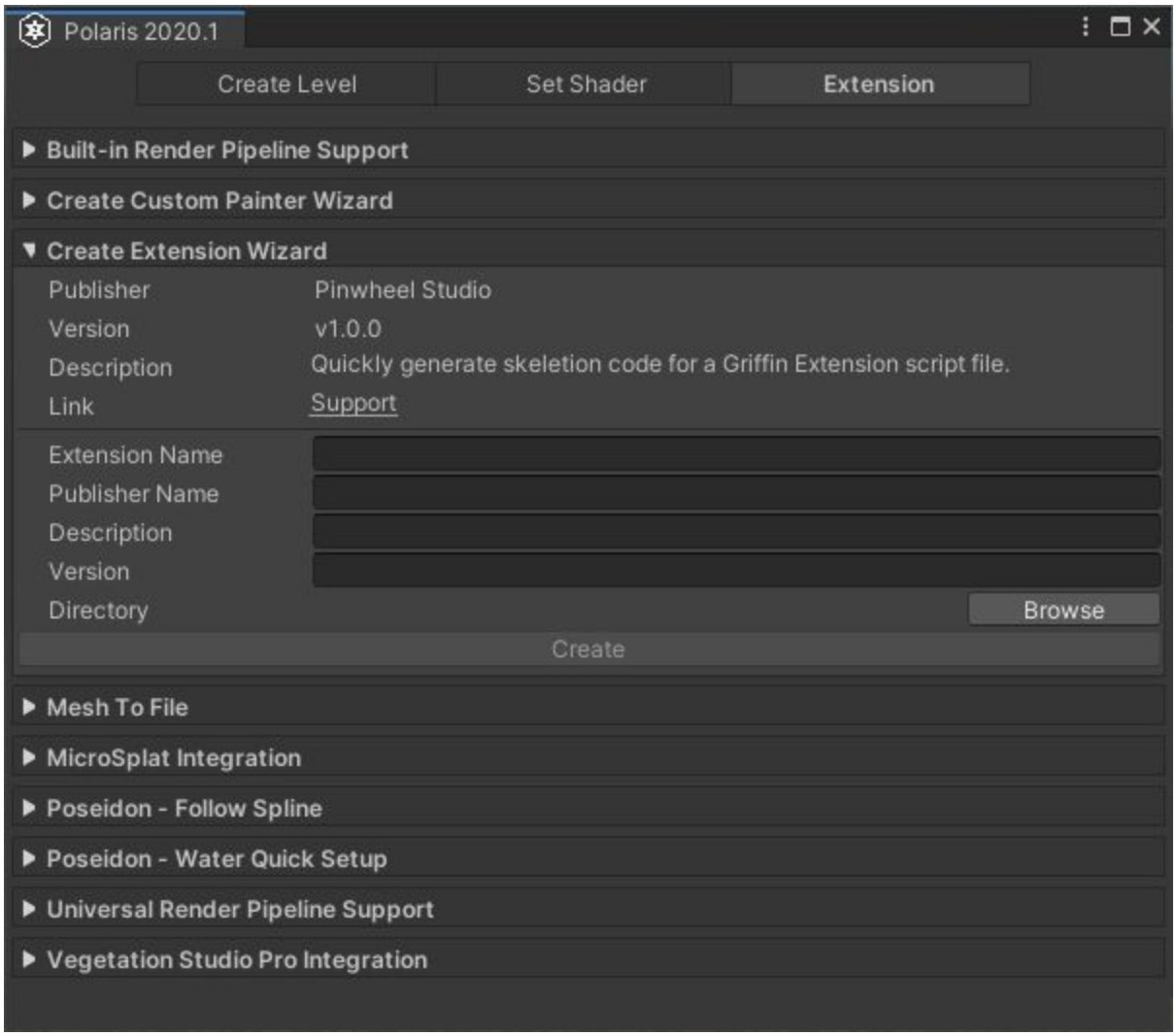
```
public static void Button_OpenConverter() {…}
```

You can create a custom GUI by providing a function as below:

```
public static void OnGUI() {…}
```

Your GUI block will be displayed right below the extension info section.

Note: You can quickly generate skeleton code for an extension by using the Create Extension Wizard, after importing CSharp Wizard:


Polaris 2020.1
Create Level
Set Shader
Extension
Built-in Render Pipeline Support
Create Custom Painter Wizard
Create Extension Wizard
Publisher
Pinwheel Studio
Version
v1.0.0
Description
Quickly generate skeletion code for a Griffin Extension script file.
Link
Support
Extension Name
Publisher Name
Description
Version
Directory
Browse
Create
Mesh To File
MicroSplat Integration
Poseidon - Follow Spline
Poseidon - Water Quick Setup
Universal Render Pipeline Support
Vegetation Studio Pro Integration

THIRD PARTIES ASSETS SUPPORT

Amplify Shader Editor

Most of the terrain and foliage shader in Builtin RP are made with Amplify Shader Editor (ASE), so you can visually edit and tailor them to your special needs.

To open these shaders as graphs, you need to import ASE first.

If you don't have ASE yet, get it here: http://bit.ly/2Bq79CX

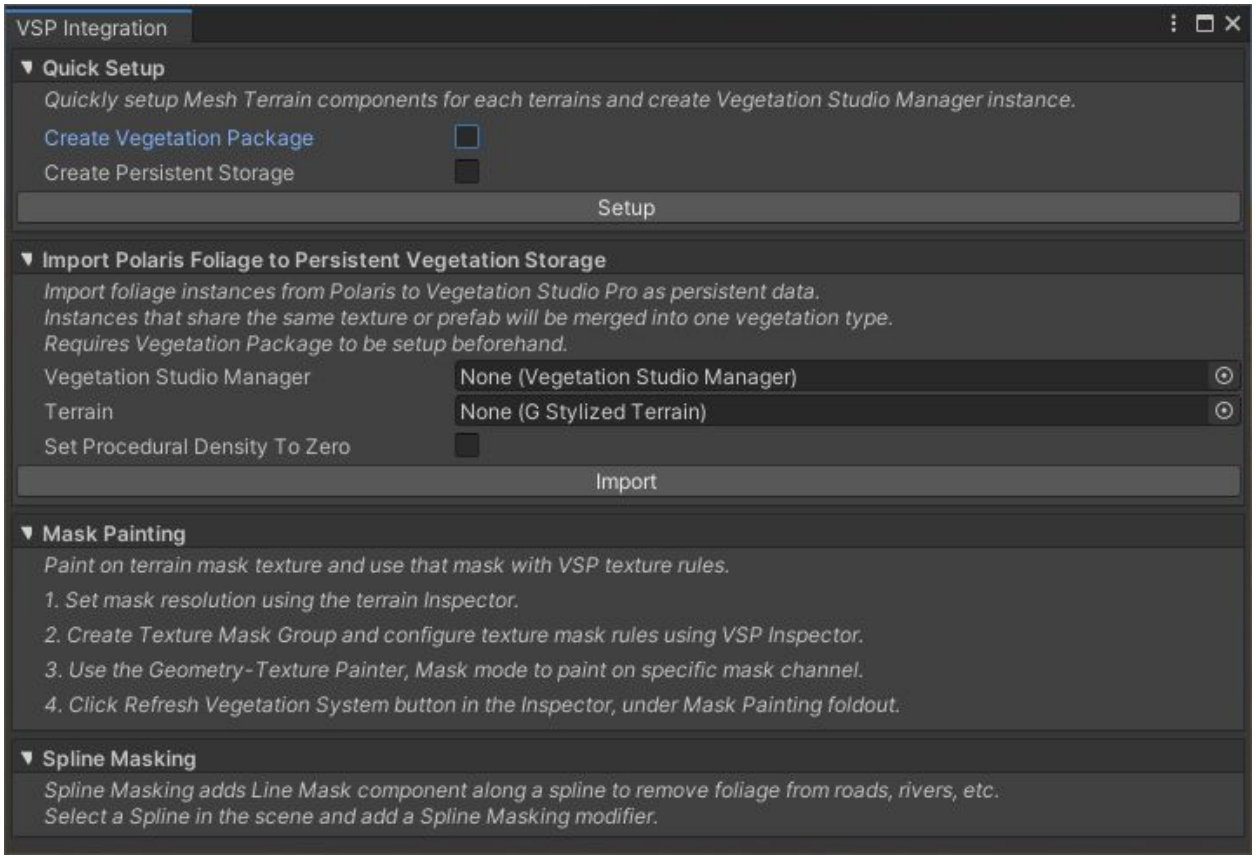
Vegetation Studio Pro

Vegetation Studio Pro integration provides handy wizards and tools to help you switch from Polaris foliage renderer to VSP.

Installing VSP and the extension

Download VSP and the VSP Integration extension package via the Asset Store.

Go to **Window>Polaris>Tools>Vegetation Studio Pro Integration** to open the editor.



Follow the instructions provided in the window to perform integration tasks.

MicroSplat

MicroSplat integration enhances your terrain shading with several techniques that offer high visual fidelity and performance.

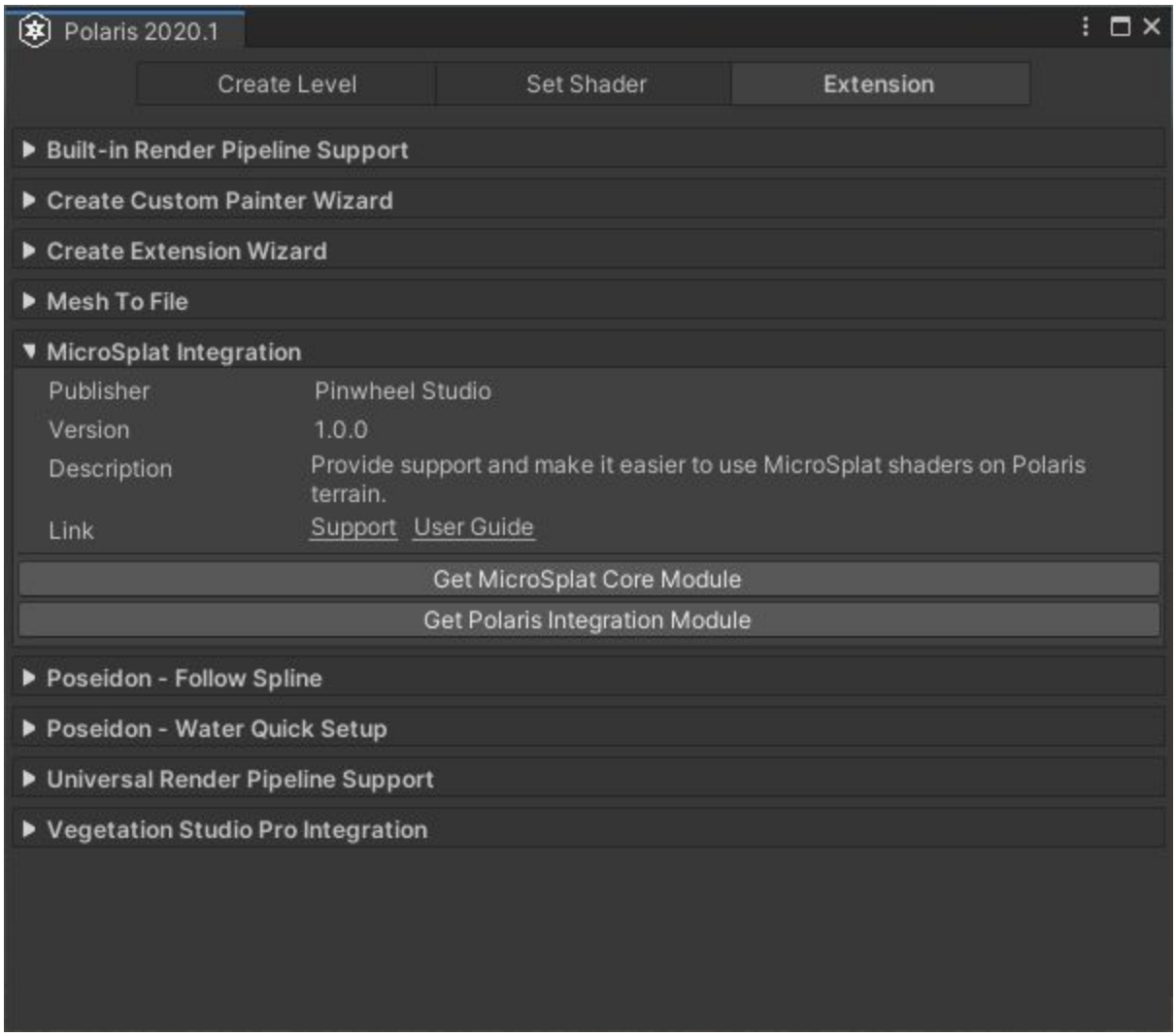
The integration also provides tools for quickly setting up your terrain to be used with MicroSplat, as well as texture synchronization between the two systems, so texturing tools will work as usual.

Please carefully read and follow the instructions below for correct usage.

Installing MicroSplat and the extension

Open the Extension window by going to **Window>Polaris>Tools>Extensions**.

In the Extension window, look for the **MicroSplat Integration** entry:

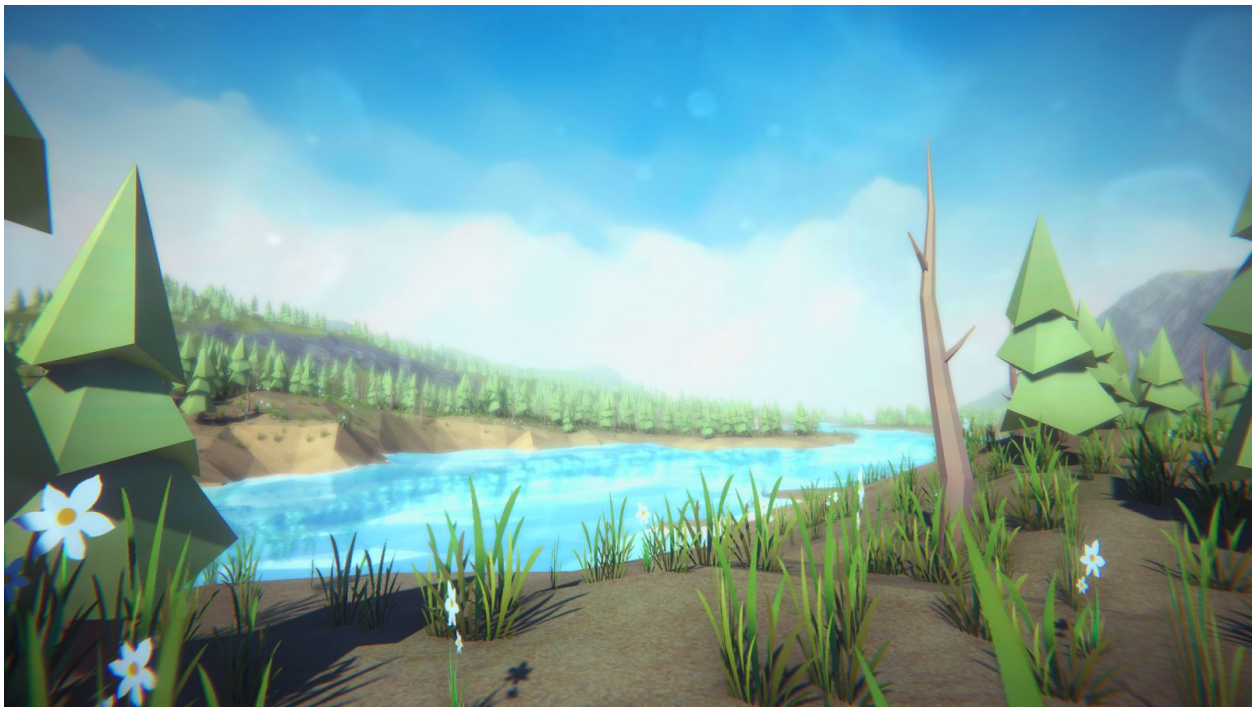


Click on **Get MicroSplat Core Module** and **Get Polaris Integration Module** to download and import necessary packages. Or you can use the following links directly:

- MicroSplat Core Module.
- Polaris Integration Module.

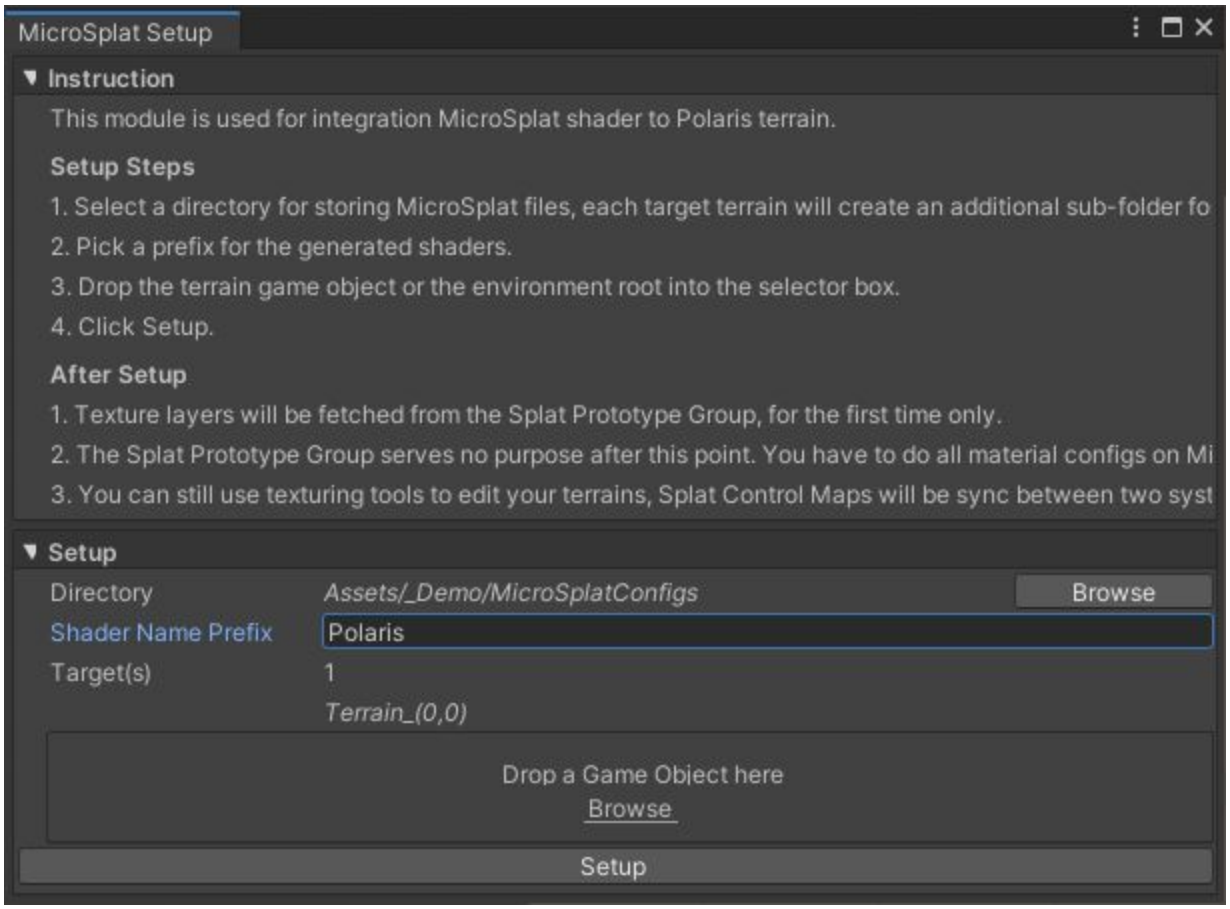
Setting up your terrain

Next step is to connect Polaris terrain to MicroSplat system. Let's take the Demo_00 scene as an example.



Open the Demo_00 scene and select the terrain game object in the Hierarchy.

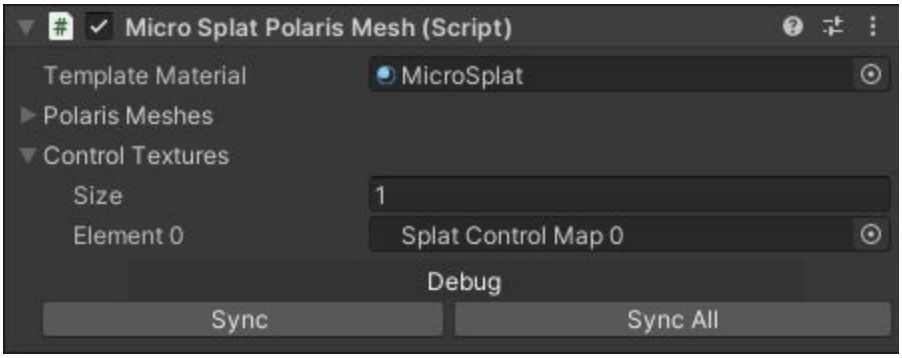
In the Inspector, you will there is an additional section called MicroSplat Integration, expand it and click Open Editor:



Select a directory for storing MicroSplat data, type in a prefix for new shaders, drop your terrains into the box, then click Setup.

After the process, make sure you see the following things:

- A new component Micro Splat Polaris Mesh is added to the terrain, with the splat control map(s) assigned.



- In the selected directory, there are several files created.

- Select the MicroSplatConfig asset, the terrain splat layers are set. Note that they only be set for the first time setup, if you want to add more or edit splat layers later, you have to use MicroSplat editor instead.

At this point, Polaris has been linked with MicroSplat to render the terrain. All tasks related to splat layers, material editing, etc. should be performed using MicroSplat editor and follow its instructions.

Note: In case you want to restart from the beginning, make sure to delete the Micro Splat Polaris Mesh component and the files generated by MicroSplat before proceeding.

Result

The images below utilize MicroSplat Core module and Low Poly Look module to achieve its style.