GRIFFIN

USER GUIDE

(v2.4.8)

Next-gen Low Poly Terrain Engine for Unity

PROLOGUE

Griffin is a complete solution for creating stunning large scale stylized environment, especially built for Unity developer. This is a must-have asset which saves you thousands hours of work and tons of effort, give 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 release log, please visit this page.

For business, please contact: hello@pinwheel.studio

For support, report a bug or request a feature, please contact: support@pinwheel.studio

RELEASE LOG

V2.4.8

NEW FEATURES

- Adding support for MicroSplat as an extension.

IMPROVEMENTS

- New splat selector for texturing tool, now display as a grid if multiple texture is used in the same layer.

FIXES

- Fix Texture Stamper height blend precision.

DEPRECATED

- Remove all assembly definition files as they introduce too many constraints.

===

V2.4.7

IMPORTANT!

- Please BACK UP your project and perform a clean update. This is required due to major changes in package structure.

NEW FEATURES

- Compile Runtime and Editor code in their own assemblies.

- Adding some test case to use with Unity Test Runner to quickly detect package misconfigure issues.

- Adding sample assets from 2 talented publishers Total Game Assets and Distant Lands. See ThirdPartiesNotice.txt for detail.

IMPROVEMENTS

- Re-organize package files for better assemblies managements.

- Many code clean up.

===

V2.4.6

NEW FEATURES

- Adding the Builtin RP Support extension to reset material and shaders without the need of re-import the package.

- Adding Polygon Distribution property, which allows you to select Dynamic or Uniform distribution.

IMPROVEMENTS

- Terrain chunk's position will be place at the center of their mesh for better level streaming and baking.

- Optimize geometry generation process.

- Strip off vertex color and tangent channel from geometry meshes if it's not necessary for the shader.

- Improve grass batching performance, reduce grass mesh storage.

- Optimize grass shader.

- Change height map format to RGBA32 that greatly reduce texture memory.

- History now records faster, reduce lagging on mouse down and up when painting.

CHANGES

- Height map now use RGBA32 format, with height encoded to RG channels, B for sub-division and A for visibility.

FIXES

- Fix some bug that prevent terrain from being spawn from scratch at runtime via code.

- Fix Live Preview Z-fight.

===

V2.4.5

IMPROVEMENTS

- Refactoring scripting API, adding Scripting API doc.

- Adding per-prototype grass color.

- Optimize grass shader.

FIXES

- Fix conflict between demo water shader and Poseidon water shader.

- Fix LWRP Support module not appeared in Extension window.

===

V2.4.4

IMPROVEMENTS

- Minor editor GUI improvements.

===

V2.4.3

IMPROVEMENTS

- Terrain will filled with the first splat texture on create.

- Geometry - Texture Painter perform better smooth operation between terrain tiles.

- Tree Prototypes: Adding Base Scale and Base Rotation properties.

FIXES

- Fix Texture Stamper live preview when editing multiple terrain.

- Fix Unity Terrain Data Importer bug.

- Billboard Creator now render alpha channel correctly.

- Fix Subdiv and Visibility Live Preview on Mac Editor.

===

V2.4.2

IMPROVEMENTS

- Texture Importer/Exporter: adding import/export visibility map.

- Faster grass preview on painting.

- Group Tool: Adding bulk data import, export.

- Group Tool: Adding some context menus.

- Group Tool: Adding more overridable properties for Foliage settings.

- Geometry Stamper: Adding ability to stamp to visibility channel, use Falloff curve to blend with source geometry.

- Stamper Tools: Adding Snap To Terrain and Snap To Level Bounds action.

CHANGES

- LWRP and URP support now provided in separated packages. Install them using the Extension Window.

===

V2.4.1

NEW FEATURES

- Adding support for Universal Render Pipeline.

- Adding Extension System for extent Polaris functionalities using third parties assets.

- Adding Height Map Filter component, apply blur and step effect on height map sampling.

- New terrain callbacks: PreProcessHeightMap and PostProcessHeightMap

IMPROVEMENTS

- Shading settings GUI improvements.

FIXES

- Reset Shading setting doesn't flush out terrain material anymore.

===

V2.4.0

NEW FEATURES

- Geometry & Texture Painters Live Preview: now you can see preview of painter effect on terrain surface.

- New set of Geometry Painter: including Terrace, Remap and Noise painter.

- New set of Terrain Shader: Lambert, BlinnPhong and PBR Vertex Color.

- Support for Amplify Shader Editor: now you can visually edit terrain shader to fit your special needs using ASE. Support for both Builtin RP and LWRP.

- Terrain Wizard: Adding new option to select number of splats and normal maps when using Splats texturing model.

- Terrain Wizard: Support for Vertex Color texturing model.

- Interactive Grass: Now support for both Builtin RP and LWRP with better interaction.

- Wind Zone: a new component which allow you to control global wind effect on grass.

IMPROVEMENTS

- Adding new noise type for noise generator, including Billow and Value noise.

- Terrain & Foliage shader is completely remade for both Builtin RP and LWRP, using Amplify Shader Editor.

- Terrain Converter: now pick an appropriate terrain shader depend on number of terrain layers. (4 or 8 layers)

- Editor GUI: Display terrain shader's name under Material slot in the Inspector.

- Re-organize shader family and shader files.

- Improve tree and grass painting performance.

FIXES

- Terrain Wizard will force using PBR lighting model when using LWRP.

DEPRECATED

- Drop support for Unity Shader Graph, most shader now use Amplify Shader Editor.

===

V2.3.2

NEW FEATURES

- Interactive Grass (touch bending): Breath life into your scene by making grass react to moving agent. Support for unlimited agent. This feature is still experimental, LWRP not supported yet.

IMPROVEMENTS

- Grass Prototype: adding Pivot Offset and Bend Factor properties, support for custom mesh for grass shape.

- Prototype Group: cache prefab asset path to avoid reference lost when down version the editor.

- Terrain Wizard: now you can open the wizard in Shading context menu to set terrain shader, by setting Lighting model and Texturing model.

FIXES

- Terrain Pinning: now only accept left mouse click.

===

V2.3.1

FIXES

- Minor fix for main thread checking.

===

V2.3.0

NEW FEATURES

- Object Painters: paint objects as prefab instance onto terrain surface.

- Spline based Object Spawner/Remover: randomly spawn/remove game object as prefab instance along a path.

- Object Stamper: stamp game object as prefab instance using mask.

- Object Helper: Utility component for managing spawned game objects.

- Prefab Prototype Group: a new data container for ease of managing prefab variation.

IMPROVEMENTS

- Foliage Filters System: now become Spawn Filters System, which will work with both Foliage Painters and Object Painters.

- Painters Tools: tree prototype/grass prototype/object selector now support multi-selection mode, with different probability for each prototypes.

- Texture Creator: support multi-selection mode for Foliage Distribution Map Generator.

- Foliage/Object snapping: now you can choose to snap them to terrain surface, or world objects with collider (filtered by layer mask).

- Geometry background generation: geometry mesh now generated in background thread in some case to avoid hanging the editor.

FIXES

- Rename geometry root object from "_Geometry" to "~Geometry".

- Terrain Pinning: newly pinned terrain will copy main data (dimension, material properties, etc.) from source one.

===

V2.2.1

FIXES

- Terrain Tools now work as expected in Linear color space.

- Billboard Creator now render atlas correctly in LWRP.

- Foliage have correct position after importing Unity Terrain Data.

===

V2.2.0

NEW FEATURES

- Texture Creator: a new way for creating and authoring terrain textures and advanced mask creation (up to 4K), support a wide variety of texture types: Height Map, Stamp Mask From Mesh, Sharp/Interpolated/Per-Pixel Normal Map, Steepness Map, Noise Map, Color Map, Blend Map, Foliage Distribution Map, and more interesting effect with Filters Stack and Live Preview.

FIXES

- Fix Texture Stamper & Path Painter live preview bug.

- Snap trees and grasses onto terrain surface after heightmap importing.

- Fix NULL error when enter/leave play mode, switch scene with a stamper/spline selected in the Inspector.

- Minor Editor GUI fixes.

- Minor bug fixes.

IMPROVEMENTS

- Simplify Terrain Data ID.

- Terrain Generated Data assets now display some statistic in the Inspector.

- Simplify Billboard Editor GUI, adding dropdowns to select main texture and main color properties from the shader.

- Data Tools: data importers now create Backup before and after the process.

- Terrain Tools undo/redo refactoring.

- Texture Stamper, Foliage Stamper: now you can choose to use sharp/interpolated/per-pixel normal map for Slope Blending.

- Automatically refresh terrain material when select a terrain or texturing-related tools.

- Group Tool: now it only overrides terrain material's shader instead of replacing the material.

- Editor GUI: Adding some instruction and warning for correct usage.

- Tree rendering: adding Pivot Offset to bring the tree up/down from the surface, no need to mess with modeling software.

- Terrain Pinning: newly pinned terrain will have the same shader with the previous one.

===

V2.1.4

IMPROVEMENTS

- Paint tools now able to show paint mode as grid or dropdown.

- Terrain related assets now have different icons in the editor.

- Newly created terrains will be placed under a root game object for better scene structure.

- Terrain pining mode UX improved.

FIXES

- Gradient Lookup shaders now use pre-multiply albedo blending to get rid of dark borders.

===

V2.1.3

IMPROVEMENTS

- Adding some demo tree prefabs. Updating demo scenes.

FIXES

- Scene navigation with Alt key issue when using Paint tools.

===

V2.1.2

FIXES:

- Terrain tool shaders lost at runtime.

===

V2.1.1

FIXES:

- Geometry static flags and lightmap baking result getting reset when reload scenes.

===

V2.1.0

NEW FEATURES:

- Unity Undo compatible: Undo/Redo work seamlessly with Backup system by Ctrl+Z and Ctrl+Y.

- Terrain Pinning & Auto Connect: create neighbor terrain and connect adjacent tiles in one click.

- Detail Mesh: painting and rendering a huge amount of environment detail like rocks, debrises, etc.

- Brush Dynamic: more control over brush stroke like radius jitter, opacity jitter, scatter, etc.

- Data Importer: import data from Unity terrain data, Polaris V1 data, RAW files and textures.

- Data Exporter: export data to Unity terrain data, RAW files and textures.

- Unity Terrain Converter: convert a group of Unity Terrain to low poly, migrate from 3rd parties' tools like Gaia, MapMagic and TerrainComposer is easier than ever!

- Debug Mode: visualize recently updated chunks in the Scene View, and more.

IMPROVEMENTS:

- Height Map: now switch to RGBAFloat format for more precise elevation data.

- Grass/Detail: now can align to surface normal vector.

- Paint Tool cursor: now follow geometry contour for more comfortable experience.

- Editor GUI revision: more consistent and organize design.

- Grass mesh batching: now run asynchronously in the background.

- Generated data storage: more efficient and compact.

- Stampers: smoother live preview, faster stamping process.

- Foliage rendering & Tree Collider: improve performance.

- Geometry generation: reduce memory usage.

FIXES:

- Fix tree/grass incorrect position after geometry modification.

- Fix foliage rendering culling issues.

INTRODUCTION

Griffin is a user friendly terrain engine dedicated to help you create gorgeous landscapes, deeply focused on low-poly and stylized world, that can run well on both Mobile and Desktop applications, save you a lot of time and effort!

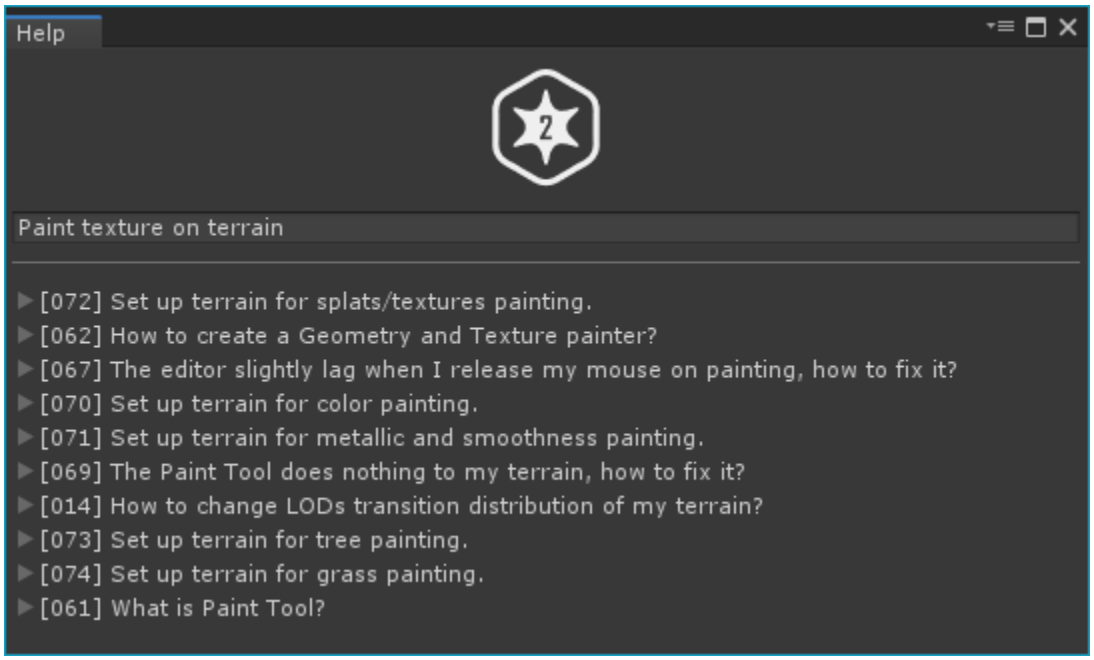
Griffin is carefully designed to totally eliminate the frustration of complexity, giving you the fastest and easiest experience of exploring and creating, empowered by the latest techniques from the inside, giving the highest quality of achievement. Even if you are a beginner or an advanced creator, it is just the right tool for you!

Different from other stylized terrain engine, Griffin is the first one which support for multi-terrains editing, empowered by GPU technologies, providing the smoothest experience. It also gives you the most flexible workflow by combining the traditional sculpting or the modern procedural method, or somewhere in between.

Griffin also comes with sample assets and example scenes for you to play with. You can even use them in your commercial projects. Have fun!

GETTING HELP WITHOUT LEAVING THE EDITOR

Griffin comes with a handy Help Tool which let you search for solutions right in the editor. Instead of composing an email and wait days for respond, now you can have the answer instantly. Go to **Window>Polaris V2>Learning Resources>Help** to open it.



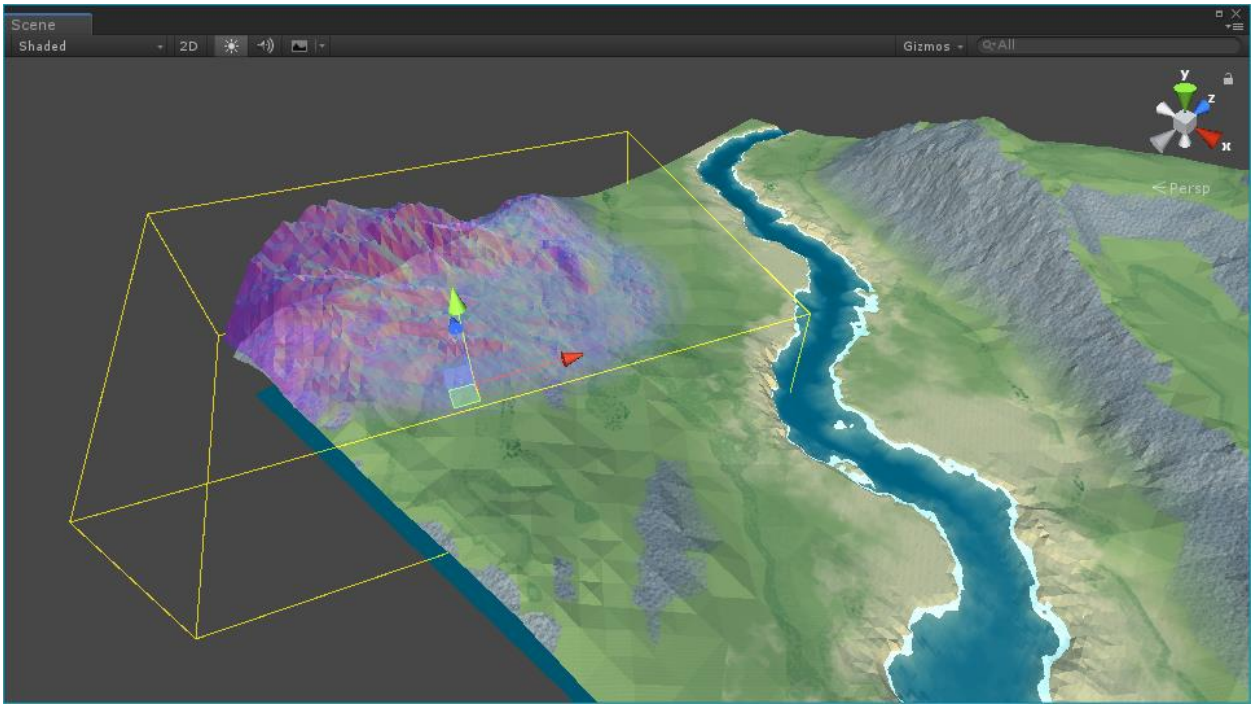
QUICK START

Before going deeper into every aspect of the tool, let's talk a little bit about its main structure.

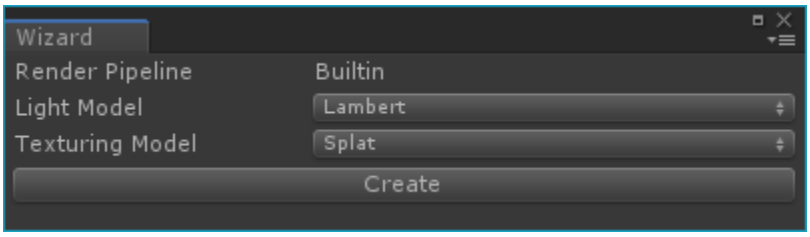
The core architecture of Griffin is designed to mainly support for multi-terrains editing and GPU processing. Every terrain in a scene consist of 2 parts: the terrain component and the terrain data asset, where the terrain component responsible for utilize the data and create the actual terrain.

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:

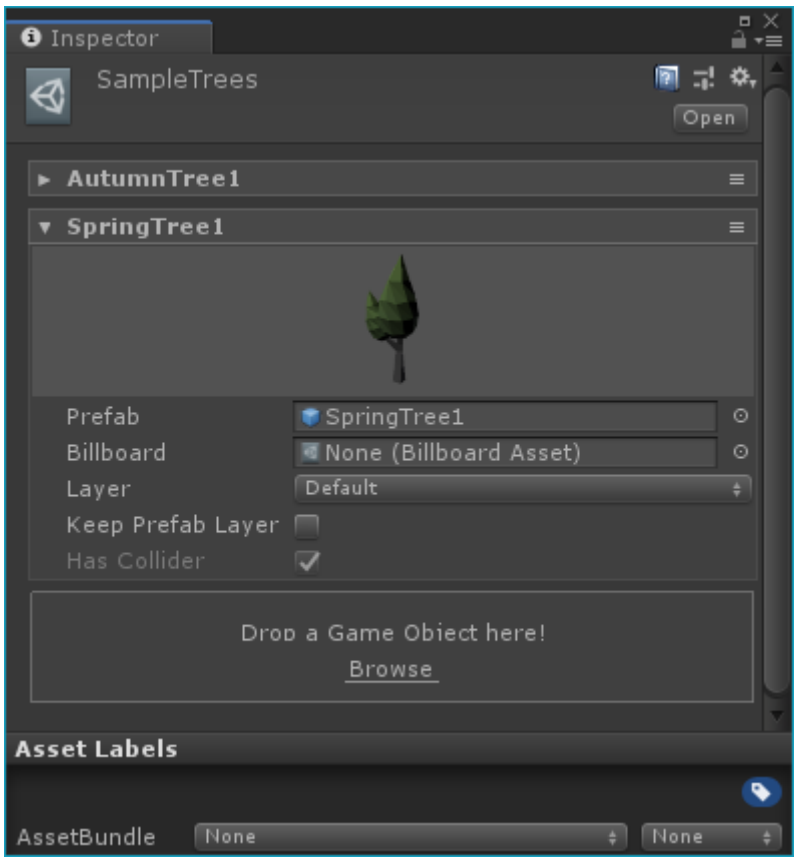
- Geometry & Texture Painter: Sculpt geometry and paint color, metallic, smoothness, splats, etc. onto the surface.
- Foliage Painter: Paint trees and grasses.
- Spline: Create spline and do some tasks like make ramps, paint paths, clear foliage, etc.
- Geometry Stamper: Stamp features onto the surface 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 method like height-based, slope-based or noise.
- Foliage Stamper: Similar to Texture Stamper, but instead of stamping texture, it stamps trees and grasses.



Griffin support a wide variety of shading styles, from geometry-based (height, slope) gradient lookup, albedo metallic, to splat map blending. To have your terrain shaded correctly, and the tools to work correctly, you have to pick the right material for it. There is a wizard tool to help you to configure in the first time (of course you can change your mind later, just select another shader for the material in the Inspector)



Since Griffin supports for 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:



Griffin provides a custom Backup System to help you manage editing history better. From v2.1.0, Backup System also works seamlessly with Unity built-in Undo System. See Backup Tool for more detail.

FREQUENTLY USED EDITOR MENUS

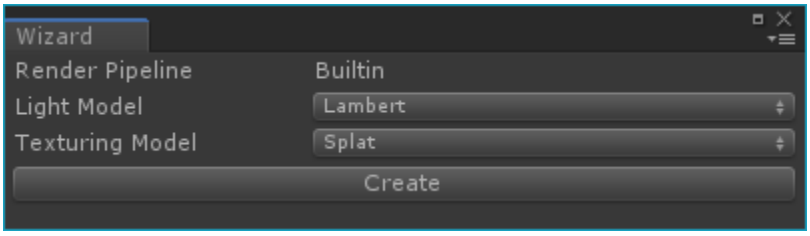
You can easily find Griffin functionalities in these editor menus:

- **Assets>Create>Polaris V2>…**: Create Griffin specific assets like Terrain Data, Splat Prototypes Group, Foliage Prototypes Group, etc.
- **GameObject>Create>3D Object>Polaris V2>...**: Create terrains and terrain tools in the scene.
- **Window>Polaris V2>...**: Open additional editor window or configure Griffin global settings.

CREATE AND CONFIGURE TERRAIN SETTINGS

Create Terrain game object

To create a new terrain, go to **GameObject>3D Object>Polaris V2>Low Poly Terrain (Wizard)**, this will bring up a window for material config:

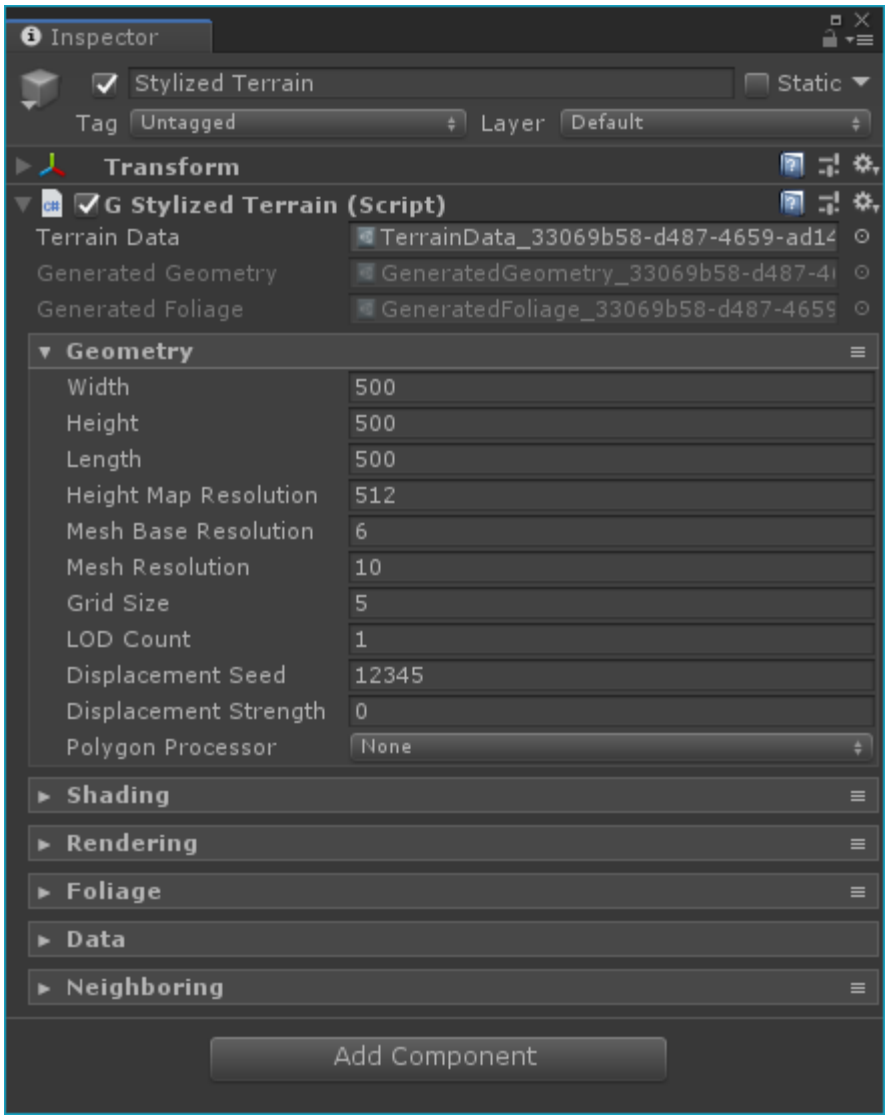


Depend on your art style to select appropriate option, then hit Create:

- Render Pipeline: this will be automatically config based on you Graphics Settings, currently only support for Built-in and Lightweight Pipeline.
- Lighting Model: select between Lambert (diffuse), Blinn Phong (specular) or PBR (physical based)
- 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.**
- Splats Model: choose the maximum splat layers to apply for the terrain, if using Splat texturing model.

The terrain data, its material and generated data container will be shown in the Assets/ directory. You should store them somewhere else depend on your project.

Also, a new game object will be added into the scene. Select the game object, you should see the terrain component with various settings:



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 chunk 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 LOD to generate. This should be 1 during editing to save some processing power, then you can try a higher value when editing is done. Actually it will not affect much because lower LODs are generated on another thread.
- Displacement Seed: a random seed for vertex displacement.
- Displacement Strength: Strength of vertex displacement on XZ plane.]
- Polygon Distribution: How to distribute vertices on terrain surface. Dynamic distribution will put more vertices on rough area, less on plain sight; while Uniform distribution will place vertices with the same density for all region.
- Polygon Processor: Additional processing after polygon generation such as custom tessellation, convert albedo to vertex color, etc.

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 (32) 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 (32) 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.
- Splats: 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 (32) to save memory if you don't use it.
- Properties under Advanced section: These define the actual properties name in terrain shaders and tell Griffin 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 Foliage: determine whether to draw trees & grasses or not.
- Enable Instancing: Enable GPU Instancing for rendering trees. Note that this option only work in Play mode and required tree material to have Enable Instancing flags on, otherwise it will fallback to normal rendering.
- Billboard Start: distance from the camera to render trees as billboard. If billboard is not available for that tree, it will not be rendered.
- Tree Distance: the maximum distance from the camera to render trees.
- Grass Distance: the maximum distance from the camera to render grasses.

For Foliage settings:

- Trees: a collections 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 tree can snap on.
- Tree Instance Count: the total number of tree instance on the terrain.
- Grasses: 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 patch along each X and Z axis for grass instance batching. The number of patch is a square of this. Higher patch count requires more draw calls but allow denser field.
- Interactive Grass: Enable touch bending for grass, make them react to moving object in the scene. See Setting up Interactive Grass for more detail.
- Vector Field Map Resolution: size of the vector field map, higher value make more precise grass bending, but cost more performance.
- Bend Sensitive: how fast grass react to moving object.
- Restore Sensitive: how fast grass return to their natural position when the moving object go away.
- Grass Instance Count: the total number of grass instance on the terrain.

For Data Settings:

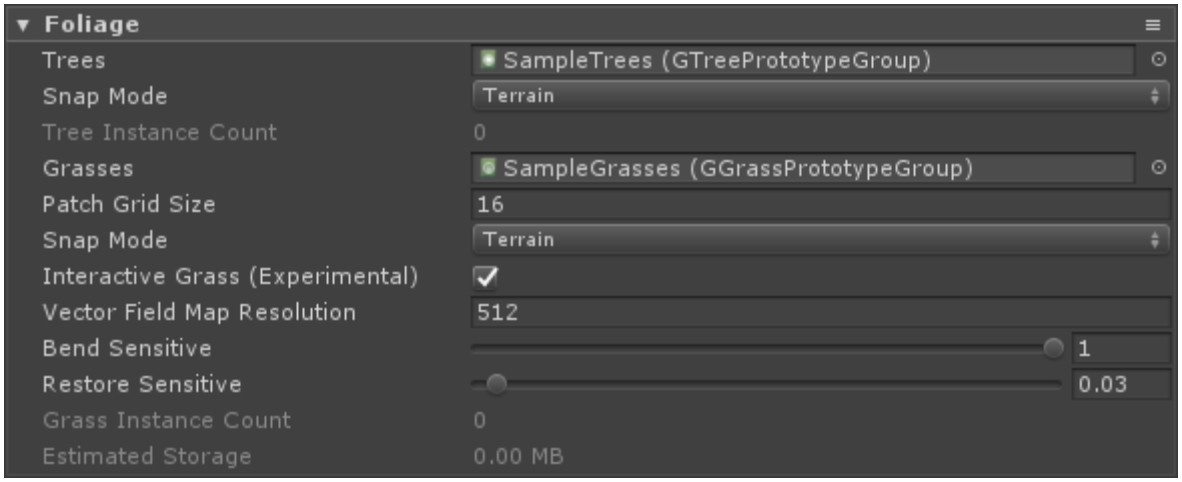
- Import: allow you import data from external sources like Unity Terrain Data, RAW files, etc.

- Export: allow you to export data to Unity Terrain Data, RAW files, etc.

For Neighboring Settings:

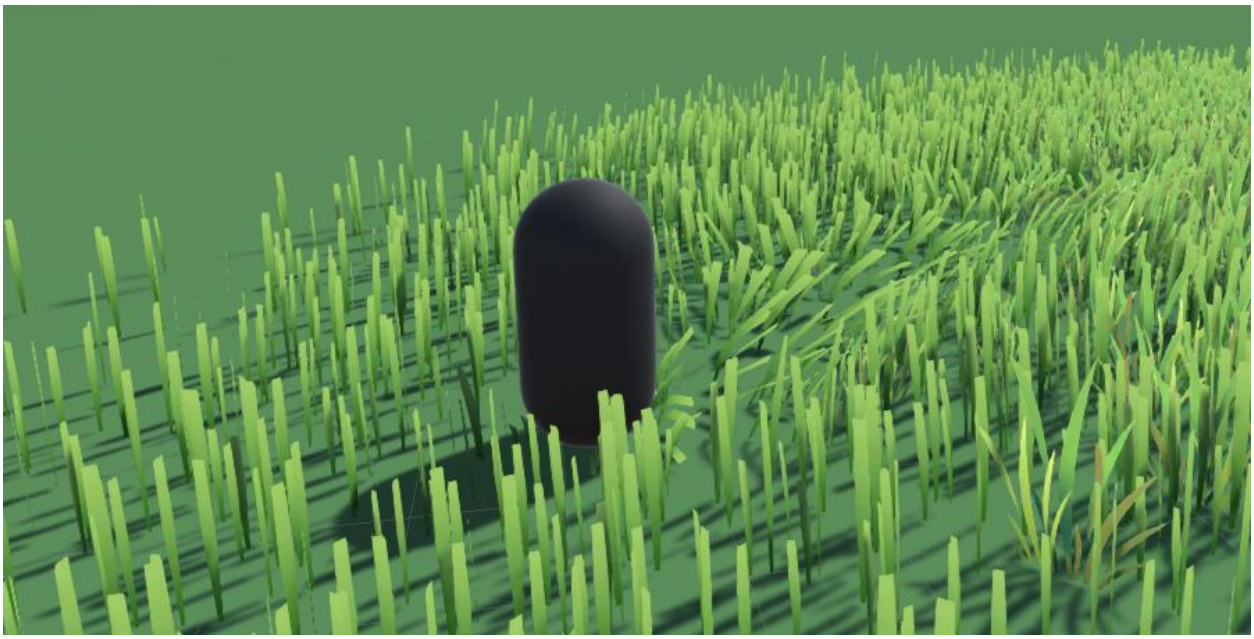
- Auto Connect: indicate that the terrain should be connect automatically to its adjacent neighbors when a new terrain is added into the scene.
- Group Id: an integer number to group terrains in the scene. This will affect how terrain tools work. For example, you can have terrains in the play area with 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 match up.

Setting up Interactive Grass



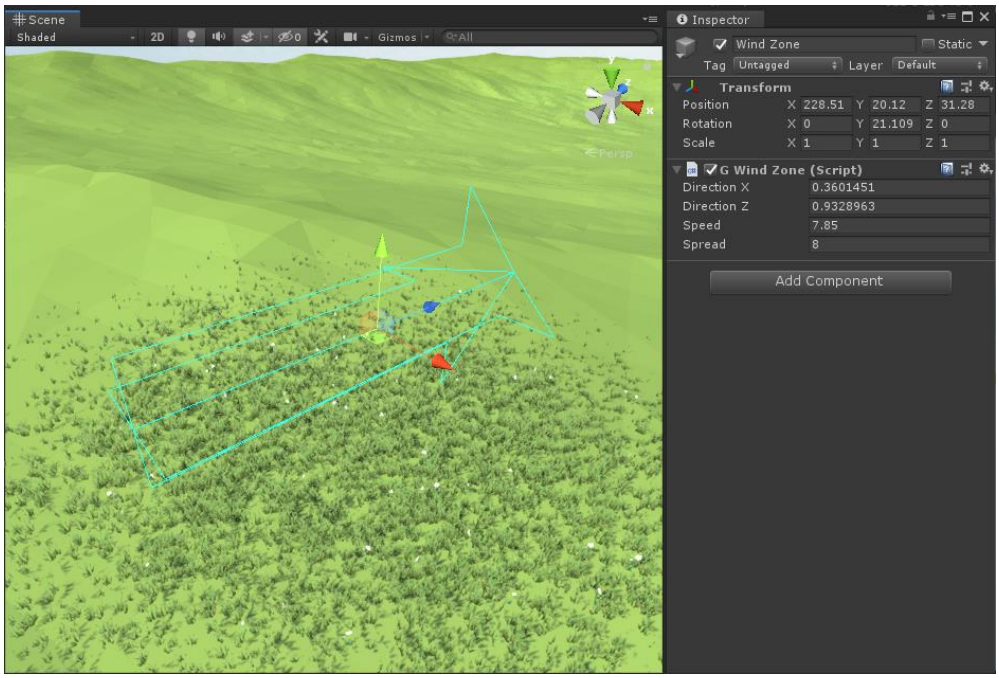
The set up process is quite simple. Select a terrain, in the Inspector, under Foliage fold out, check on the Interactive Grass toggle, then choose appropriate values for other properties like Vector Field Map Resolution, Bend Sensitive and Restore Sensitive. See Foliage settings for more detail.

Next, the system need to know who is the character so they can react correctly. Select you character, then add a GInteractiveGrassAgent component, pick a appropriate Radius value. You can add as many agent 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 V2>Wind Zone** to add one.



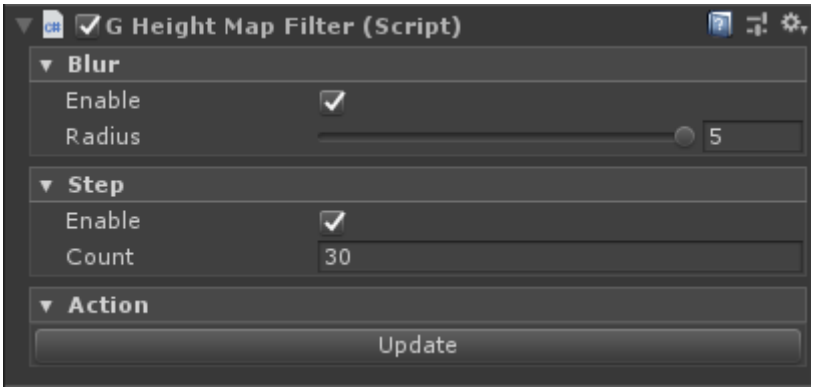
In the Inspector, there are some properties to define the wind effect:

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

Adding Height Map filters

You can add filters for height map sample to create interesting geometry effect such as smoothing and terrace.

In the Inspector, click on **Geometry>CONTEXT>Add Height Map Filter**. A new component will be added to the terrain game object:

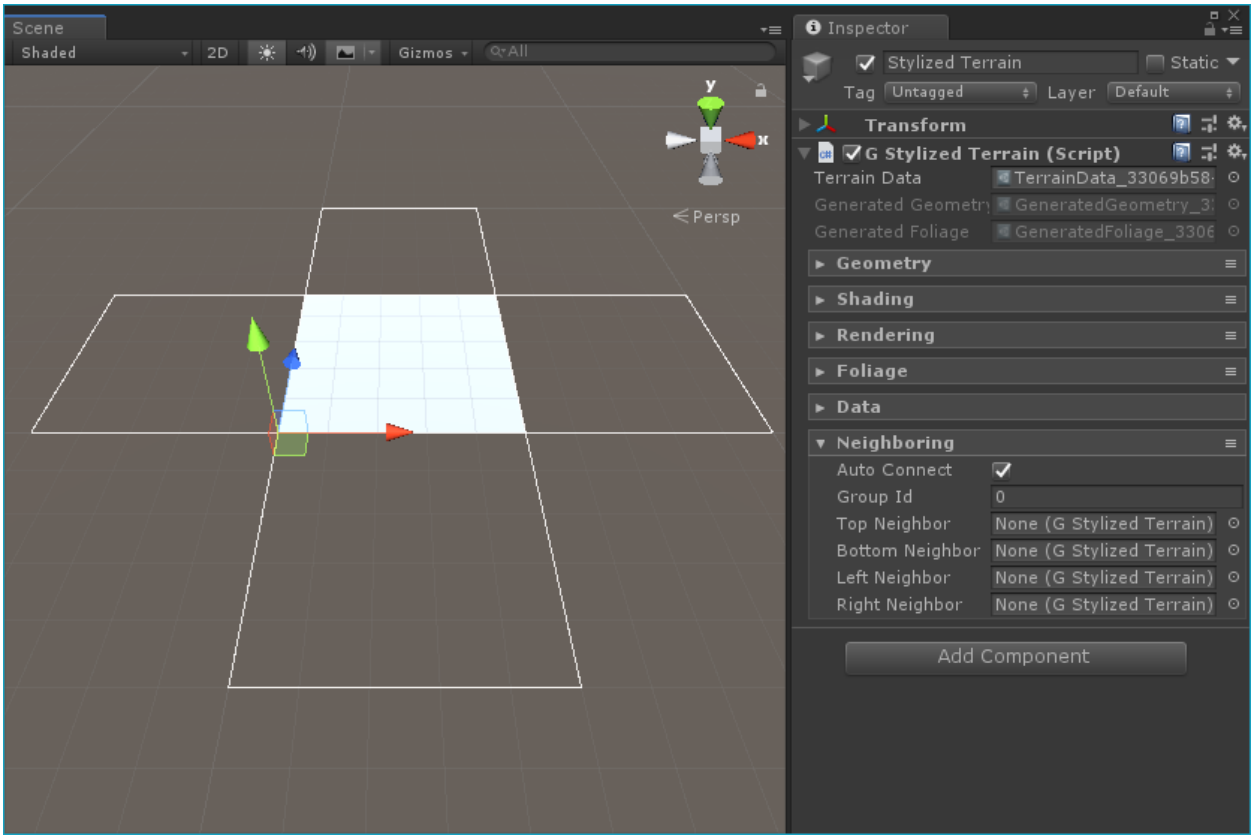


These filters will be apply top to bottom upon geometry painting (stamping, etc). Sometimes you will want to click on Update to re-generate the whole terrain.

This is a non-destructive operation, which mean your raw height map data will remain intact.

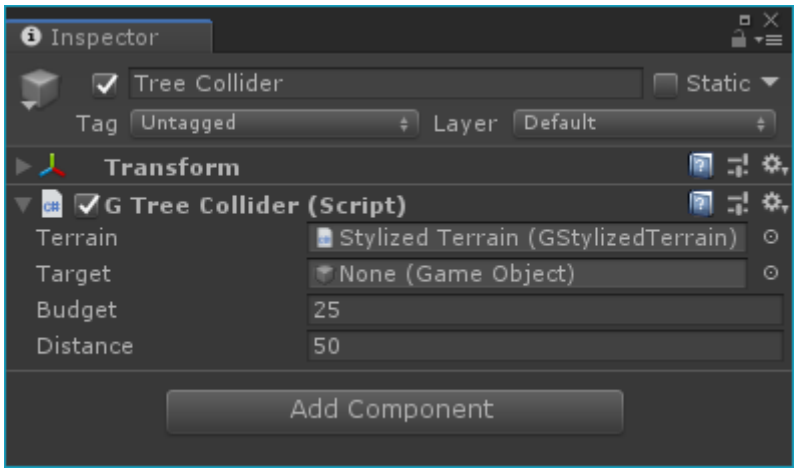
Create and connect neighbor terrains

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.



Create Tree Collider

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



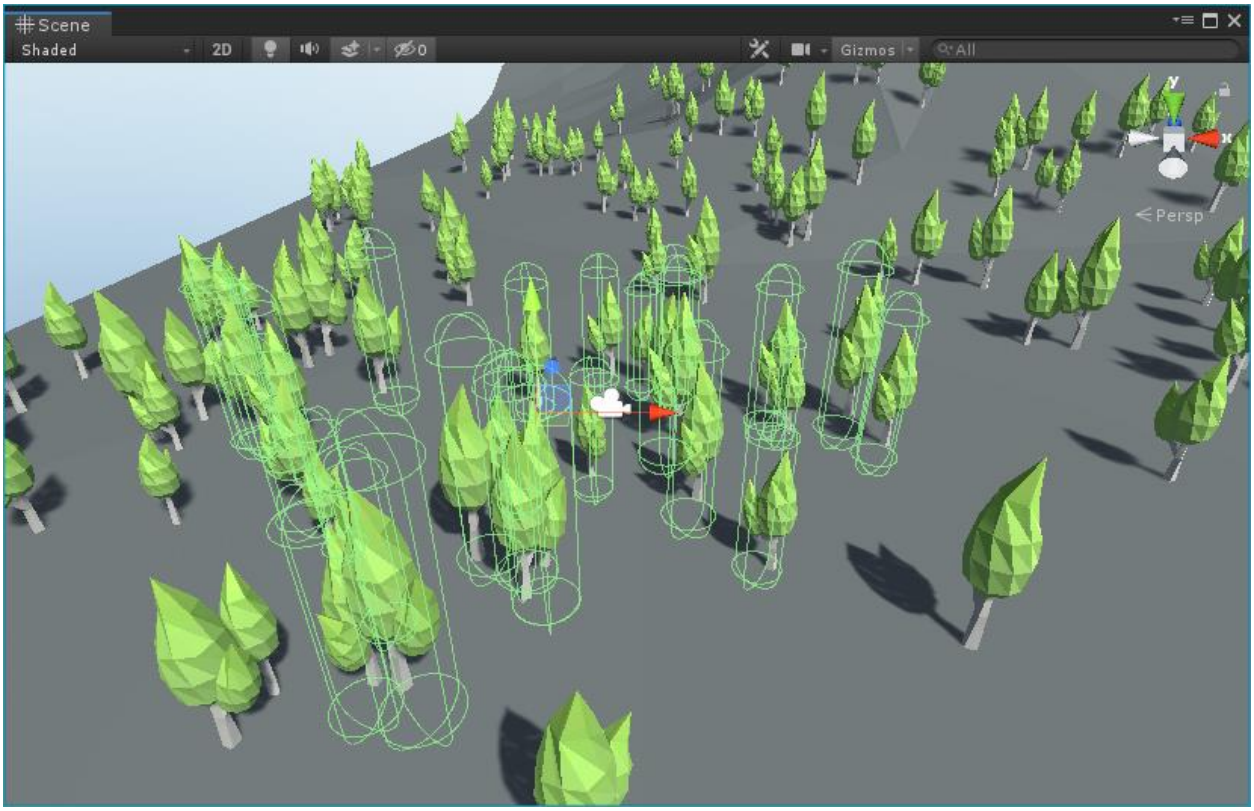
Internally, Tree Collider component store a list of Capsule Collider, which will be update every frame depend on its target position in the scene. Each Tree Collider will track only one target, to

have more target, you can add another Tree Collider by go to **GameObject>3D Object>Polaris V2>Tree Collider**.

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

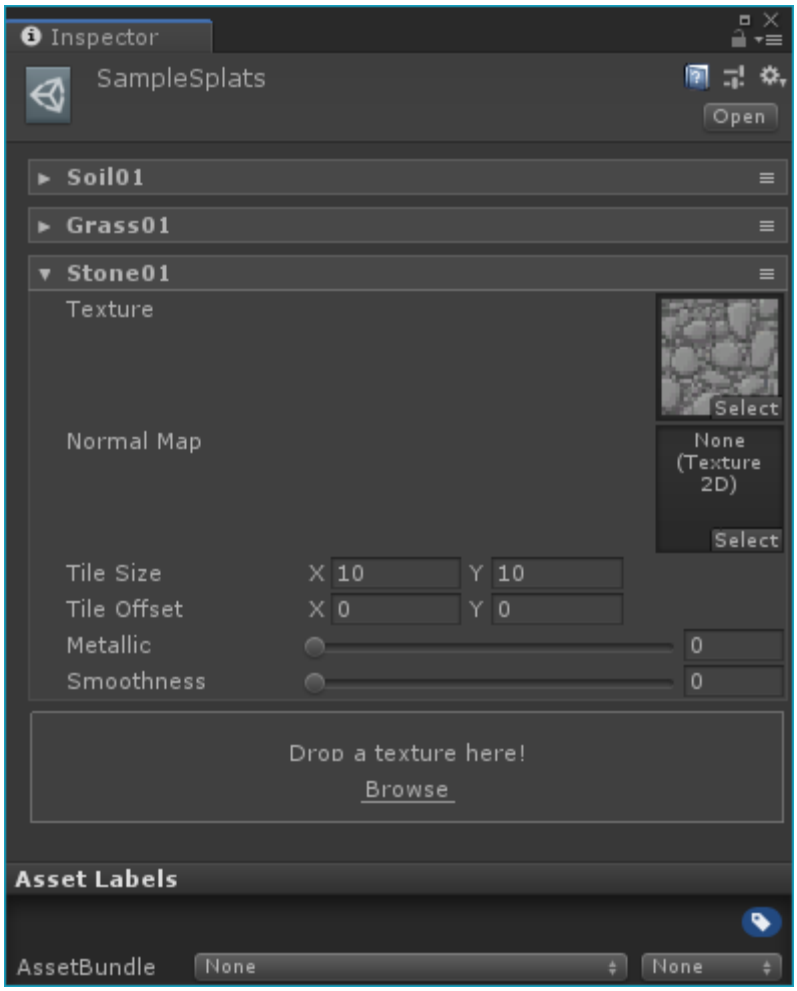
Tree Collider have the following properties:

- Terrain: the parent terrain which contain information of tree instances.
- Target: the target for it to track, if null, it will track the Main Camera.
- Budget: how many collider game object to create and cache.
- 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 to the terrain. Go to **Assets>Create>Polaris V2>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:

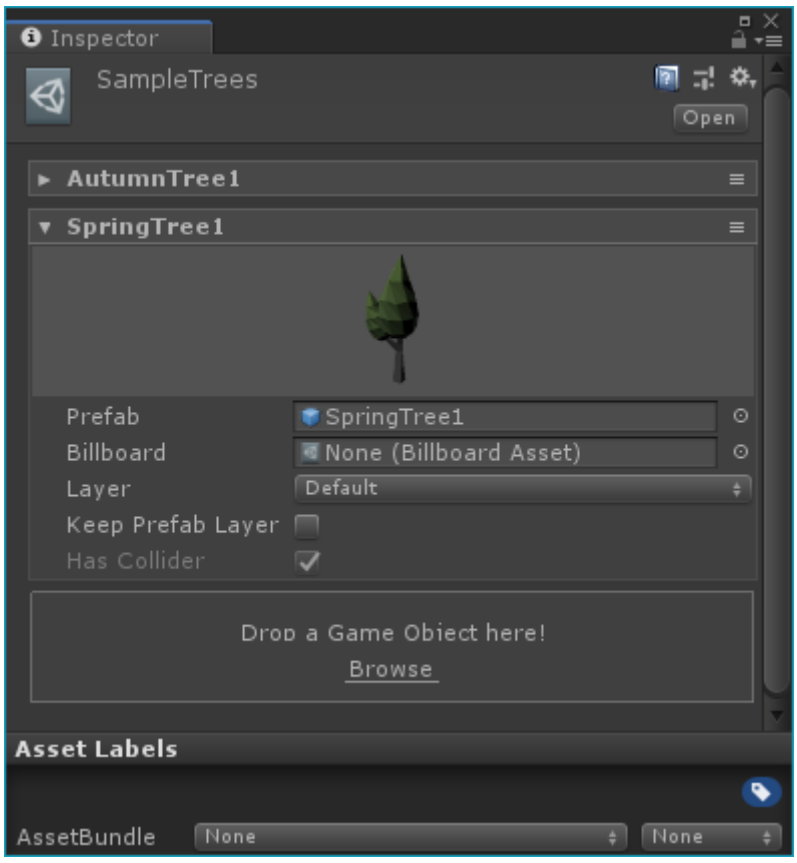


For each prototype, there are several settings:

- 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.
- Layer: determine which camera layer to render the tree.
- Keep Prefab Layer: if check on, the tree will be render in the same layer as the source prefab.
- 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

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 V2>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:

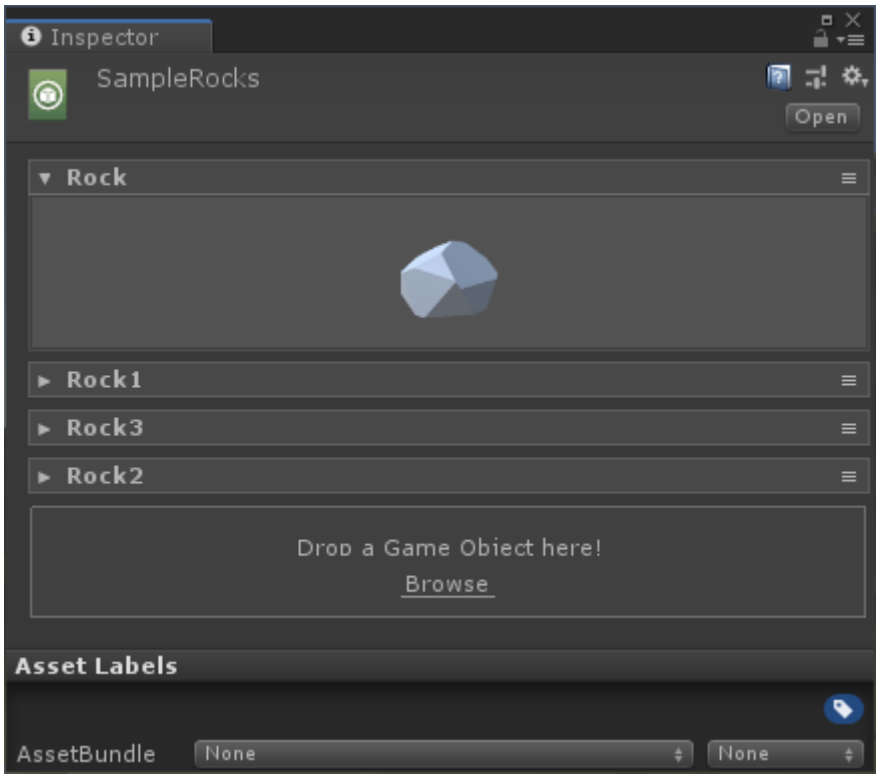
Each prototype has several settings:

- Texture: main texture represents the grass.
- 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.
- Bend Factor: how much it react to wind and other bending effects.
- Pivot Offset: A small position offset to move the instance up/down when batching.

- Layer: determine which camera layer to render grass instances.
- Shape: determine the mesh used to draw each instance, including Quad, Cross, TriCross, Custom Mesh and Detail Object.
- Align To Surface: make the grass/detail instance up vector to snap with surface normal vector (need re-update to take effect).

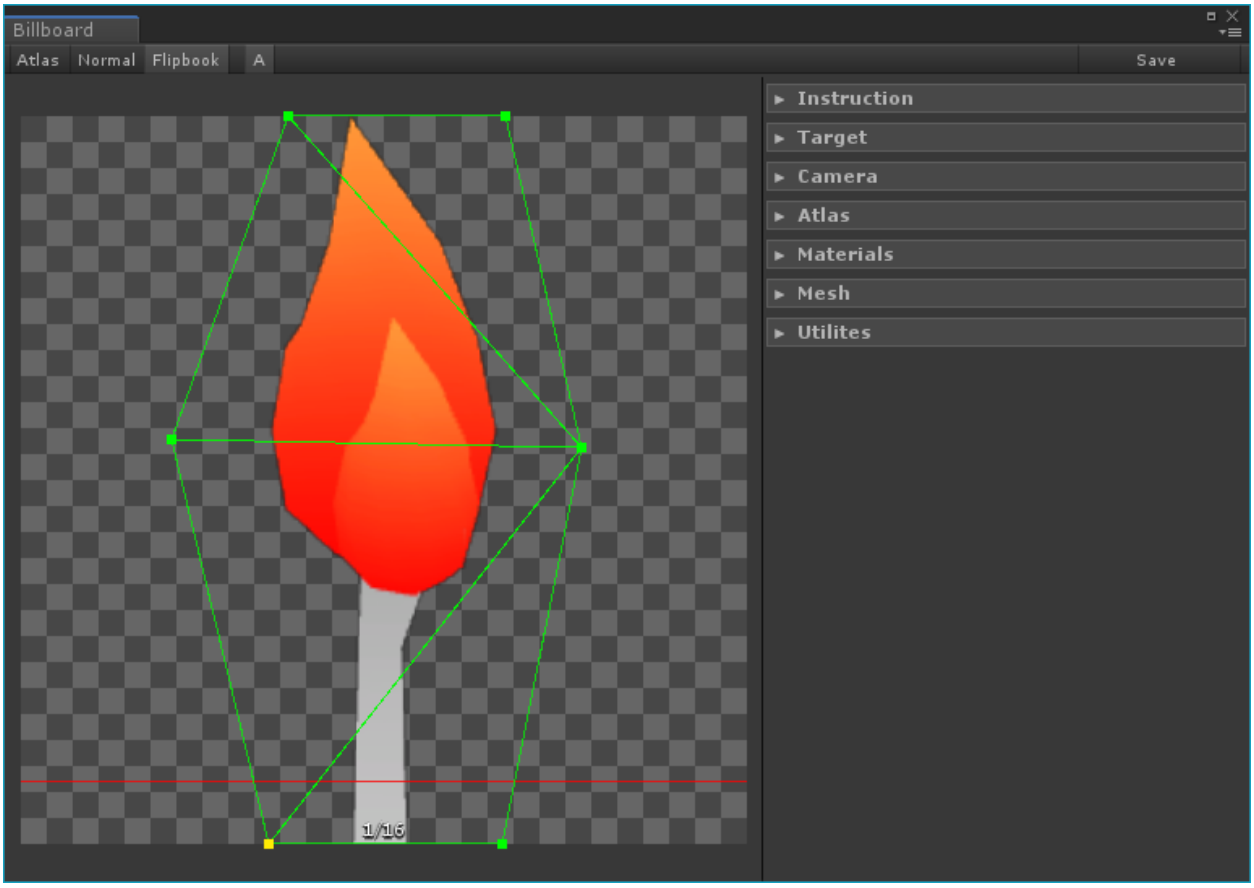
Create Prefab Prototypes Group

Prefab Prototypes Group doesn't bound to any specific terrain. Instead, it is used to store prefab variations then use later with other terrain tools. Go to **Assets>Create>Polaris V2>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

Griffin comes with a handy Billboard Asset creator to use with tree rendering. Go to **Window>Polaris V2>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 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

Griffin 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, any resource intensive operation such as texture modification should be done on the GPU side. Remember that when making your own tool!

Some actions like geometry painting require terrain neighboring to be set up correctly, so that their edges can match up and eliminating the gap between them.

Terrains in the scene can be group together for different purpose, 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 Griffin use the multi-terrains workflow, sometime it requires properties must be identical across tiles for it to work correctly. It would be time consuming to iterate and change things one by one.

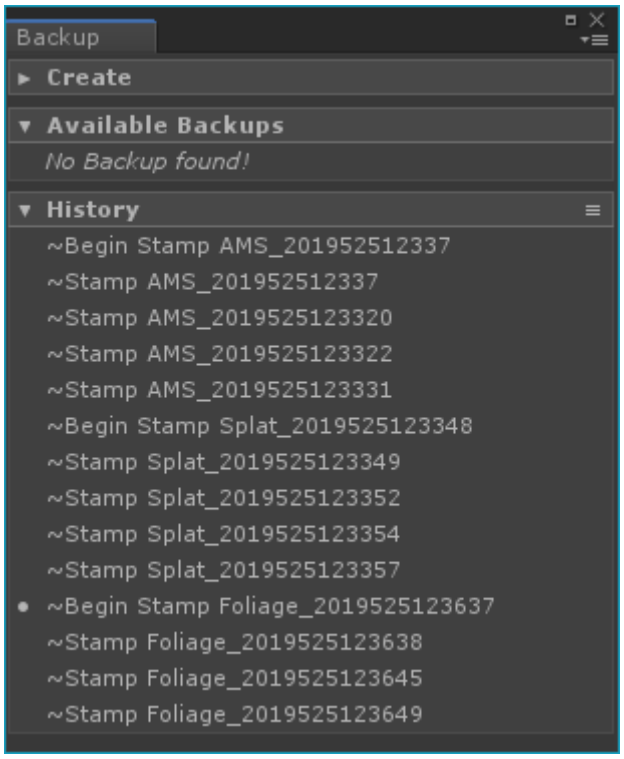
Group Tool provides a simple and easy way to override properties across tiles in a group, selectively, at once!

Usually, Group Tool is automatically created when you add the first terrain into the scene. If not, go to **GameObject>3D Object>Polaris V2>Tools>Group** to add one.

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.

BACKUP TOOL

Backup Tool provides a robust, non-linear way to store/restore editing history. To open the editor, go to **Window>Polaris V2>Tools>Backup**.



There are 2 types of backup:

- Long Lives Backup: terrains in the selected group will be fully backed up and store 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 store on disk. This type is created automatically by the tool, only data channel affected by the tool will be backed up, and will NOT remain between editor session. However, you can choose to keep them by tweaking Griffin global settings.

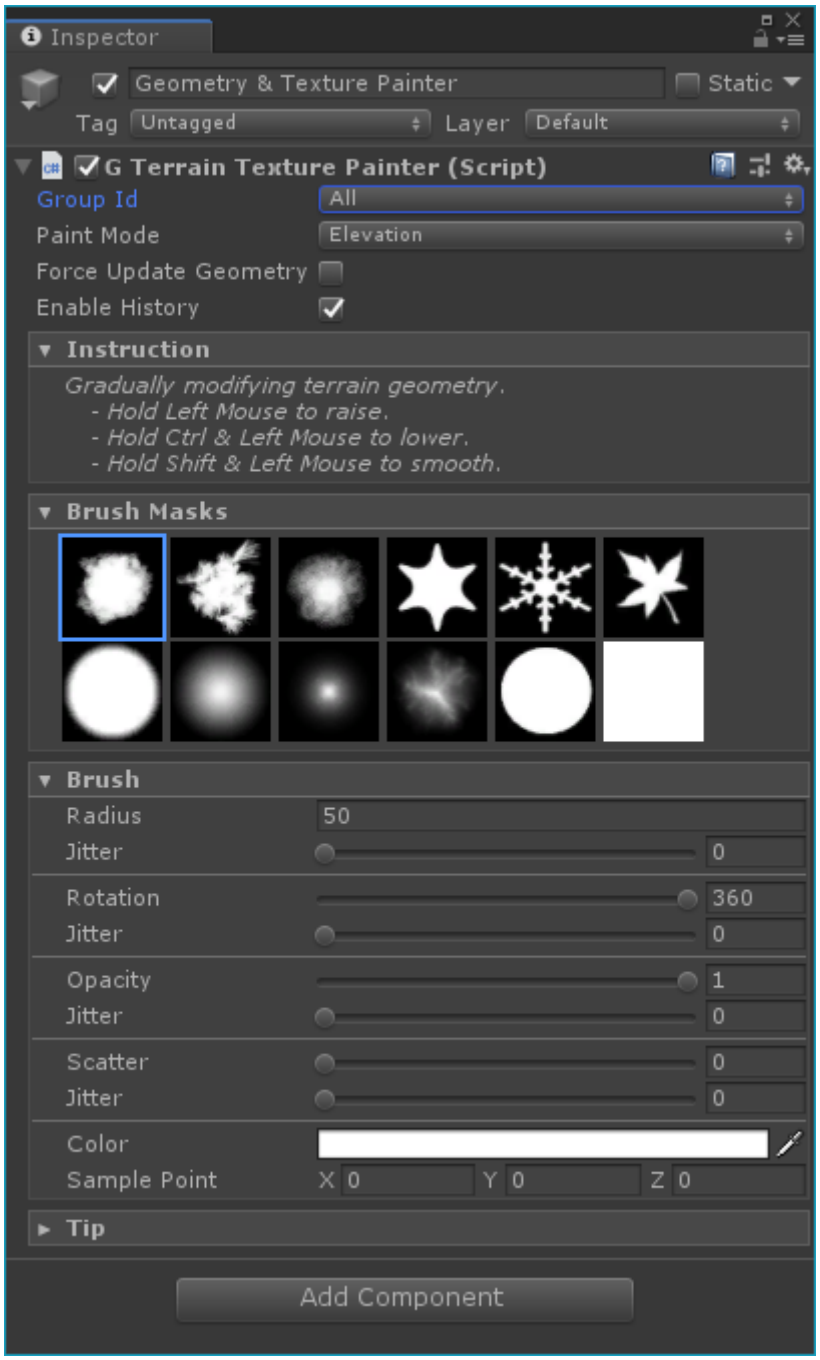
To restore a backup, simply click on the listed entry. For additional action like deleting the entry, right click on it. The small dot (●) indicates the last active Backup/History entry.

From v2.1.0 and above, 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 terrain surface. Usually, it will be automatically created when you add the first terrain into the scene. If not, go to **GameObject>3D Object>Polaris V2>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.
- 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. See Polygon Processor for more info.
- 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.
- 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.

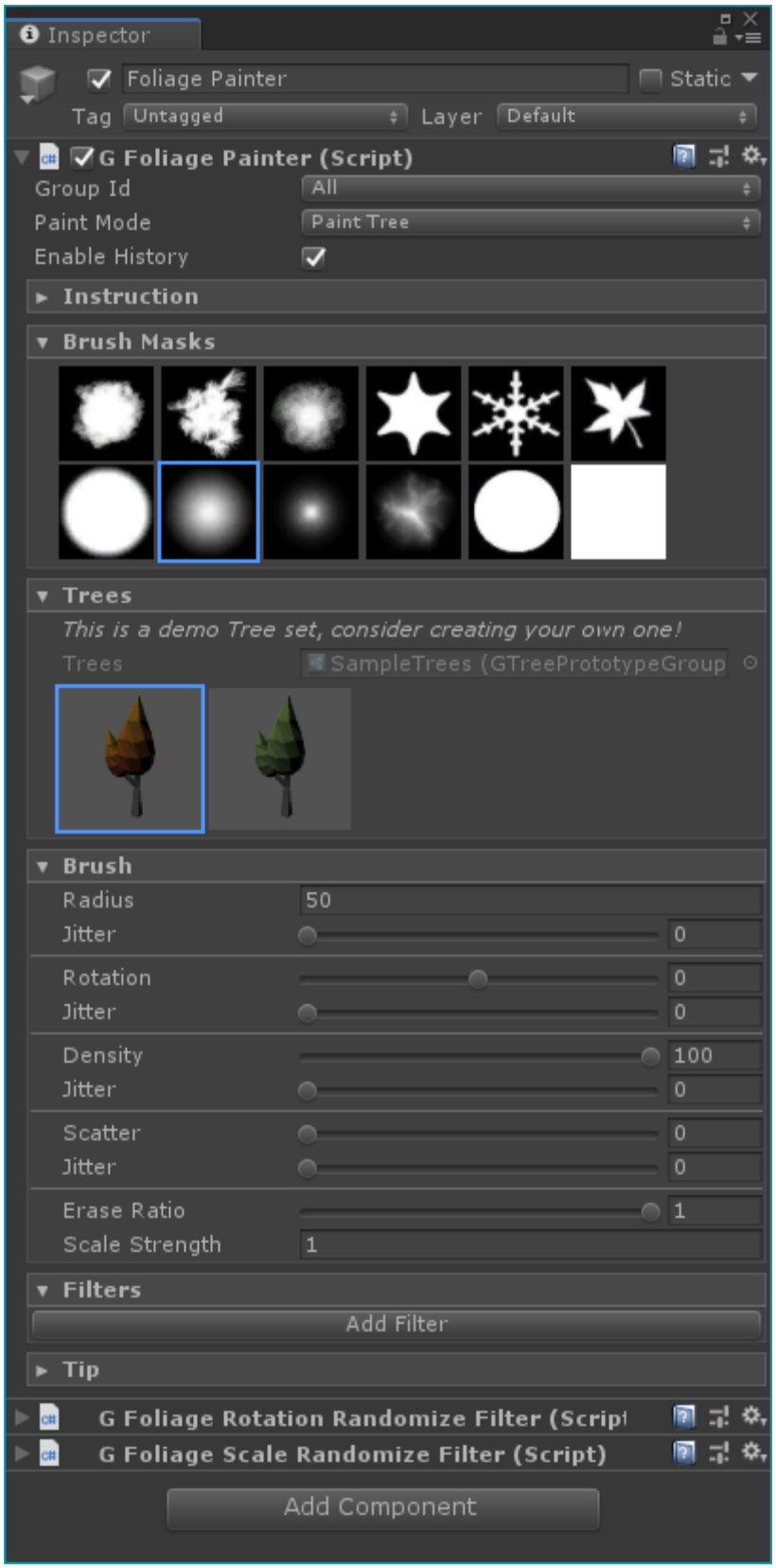
When the Scene View is focused, you can use the minus/plus, square brackets, semicolon/quote button to gradually change brush radius, rotation and opacity, respectively.

To add custom brush mask, simply put your texture into the **Assets/.../Resources/Brushes** 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. See this section for more detail.

FOLIAGE PAINTER

This tool is used to paint tree and grass instances onto the environment. Usually, it is automatically created when you add the first terrain. If not, go to **GameObject>3D Object>Polaris V2>Tools>Foliage Painter** to add one.



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

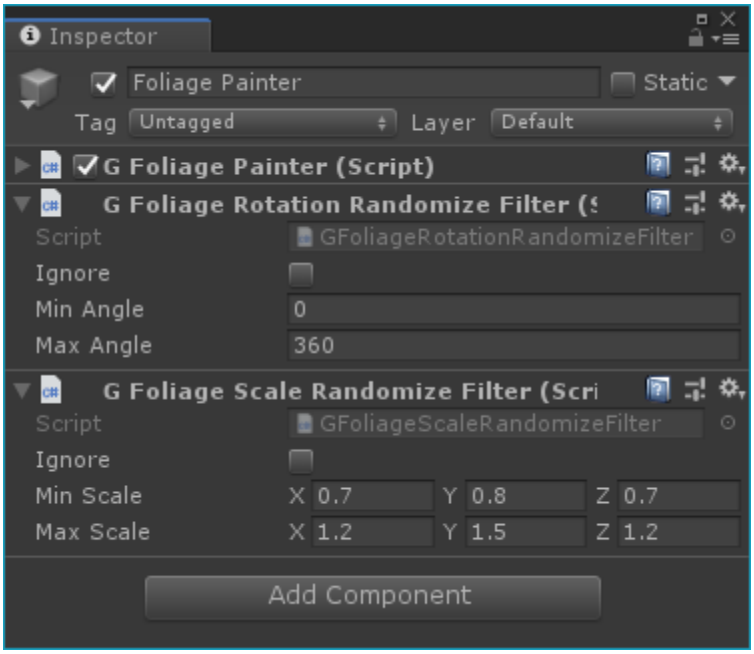
- Paint Tree: paint or erase tree instances.
- Scale Tree: modify tree instances size.
- Paint Grass: paint or erase grass instances.
- Scale Grass: modify grass instances size.
- Custom: other custom functionalities, defined by user code.

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 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 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 filter are automatically added when you create the painter.

OBJECT PAINTER

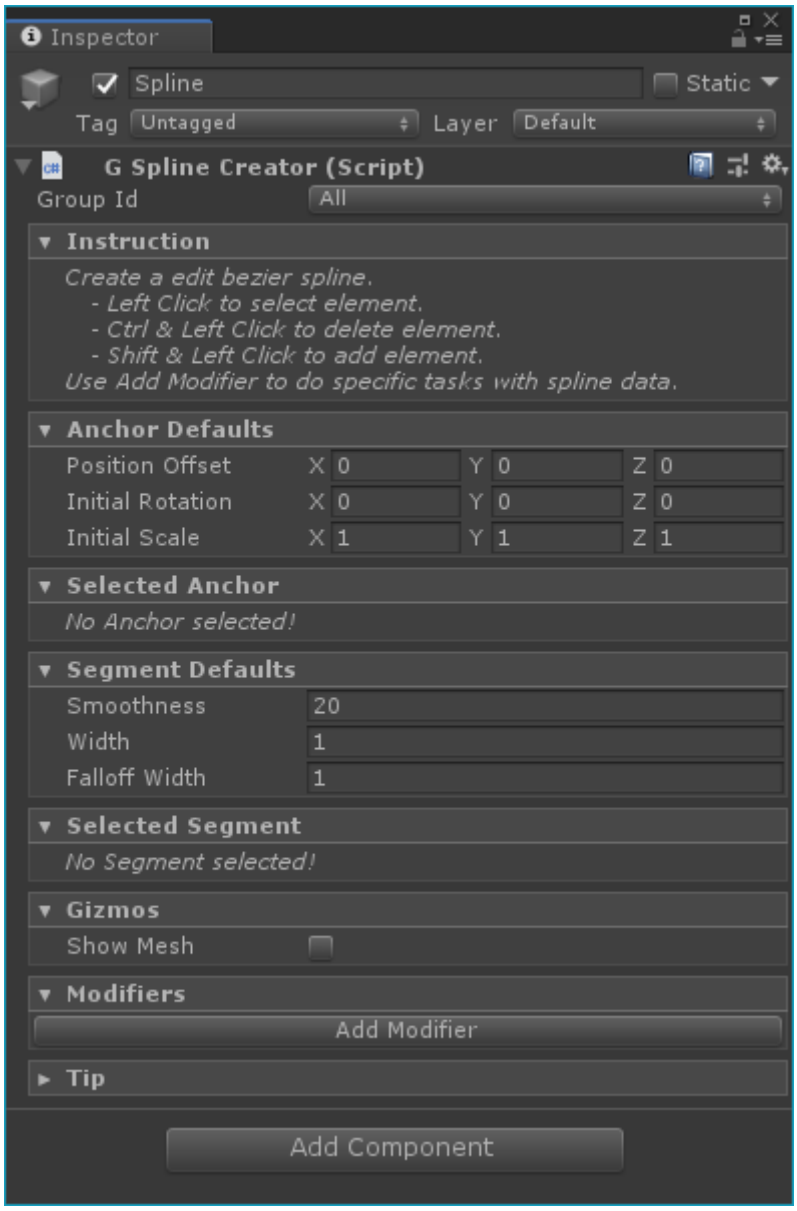
This tool is similar to Foliage Painter, but it work with prefabs instead. Usually, it is automatically created when you add the first terrain. If not, go to **GameObject>3D Object>Polaris V2>Tools>Object Painter** to add one.

See FOLIAGE PAINTER for more detail.

SPLINE CREATOR

Spline Creator is a handy tool to perform some specific operation 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 V2>Tools>Spline**.

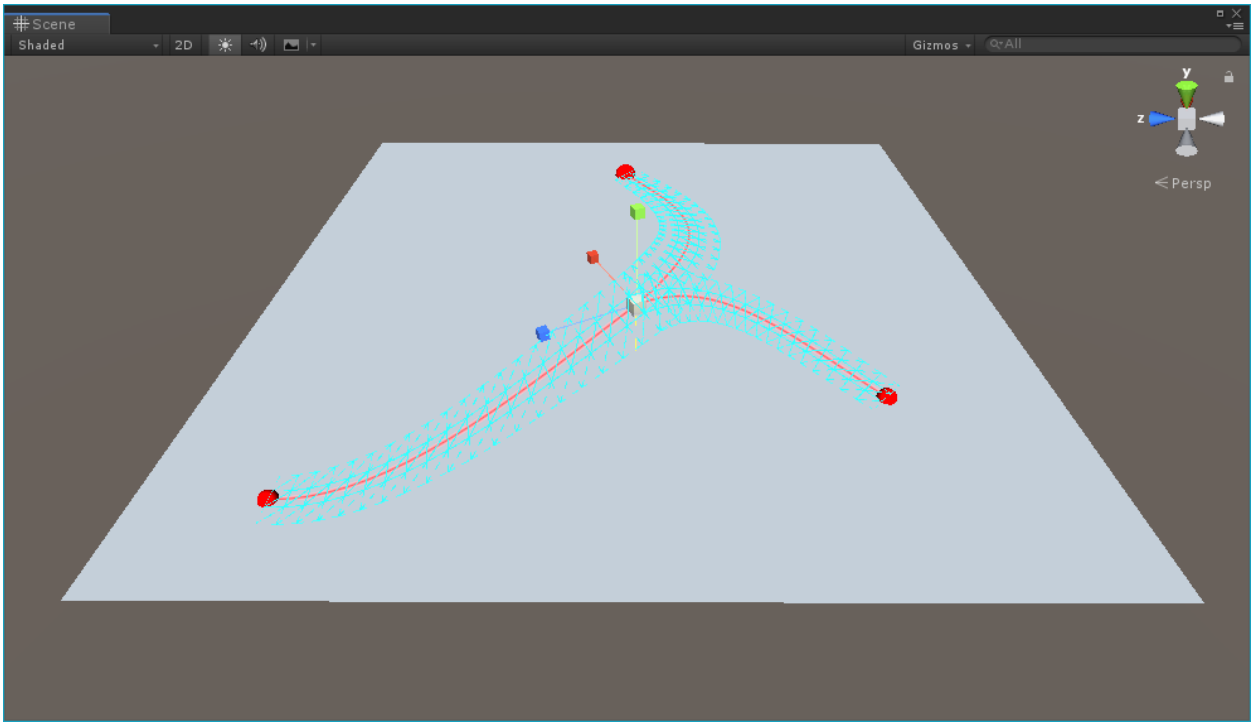


The spline holds a list of anchor points where it goes through and some segments to connect their anchors. There are some actions when working with spline:

- 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 surface.

Each anchor has its own position, rotation and scale. You can modify these information using 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 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 Anchor Defaults:

- Position Offset: an offset to add to 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.
- Rotation: rotation of the anchor.
- Scale: scale of the anchor.

For Segment Defaults:

- Smoothness: spline subdivision factor, larger number 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:

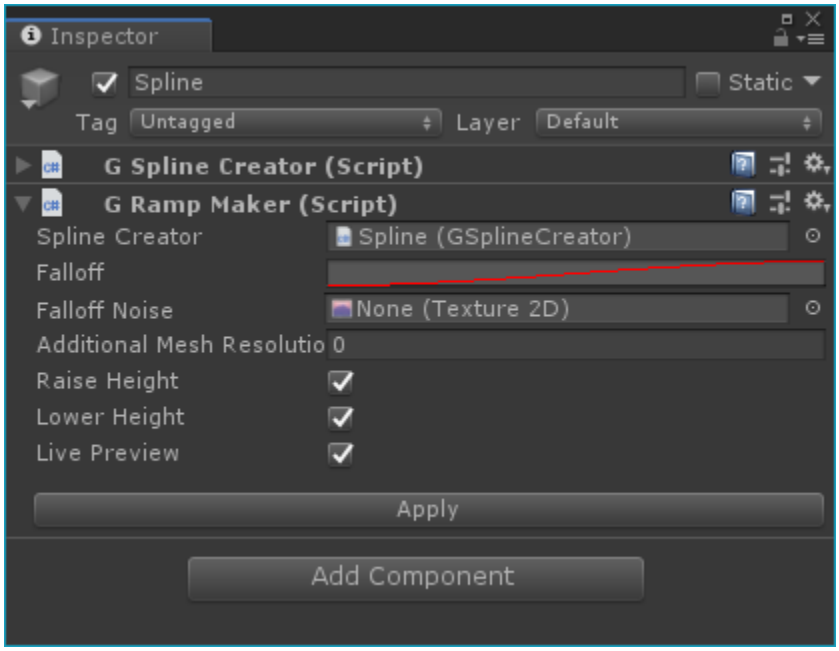
- Start Tangent: position of the first tangent relative to its anchor.
- End Tangent: position of the second tangent relative to its anchor.

For Gizmos:

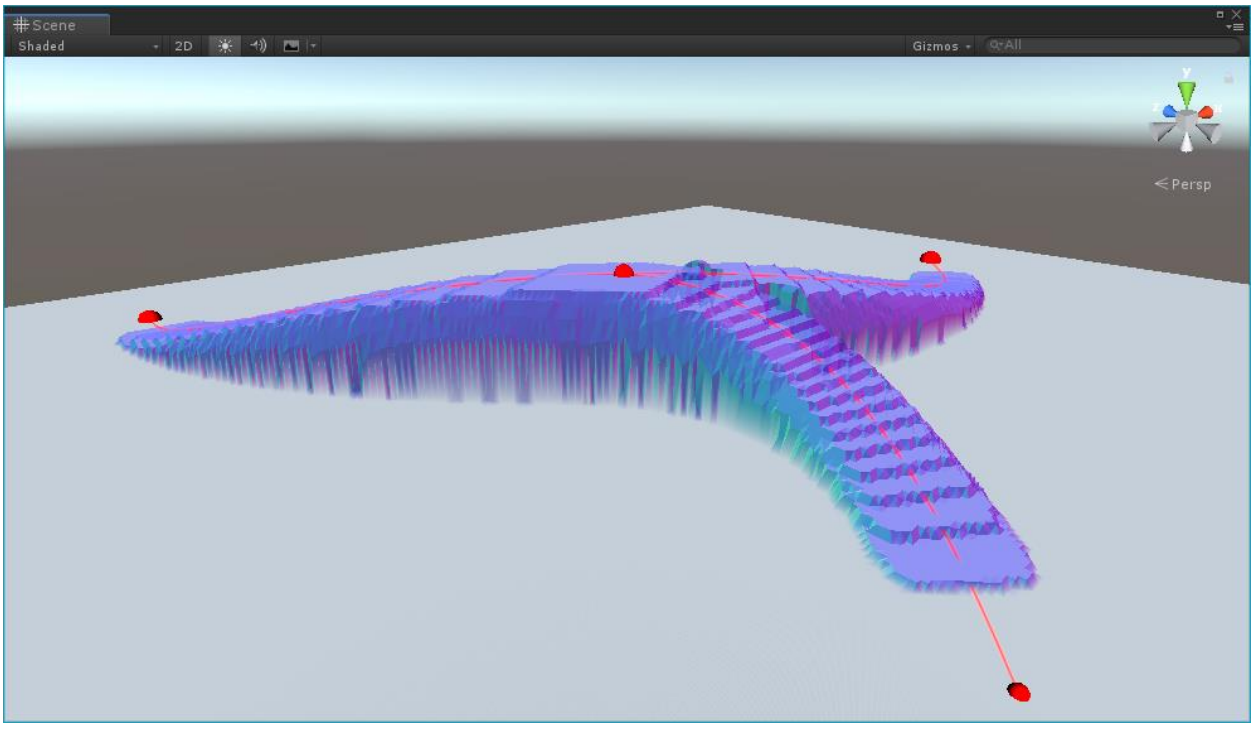
- Show Mesh: determine whether to visualize the spline mesh in the Scene view or not.

The Spline Creator 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 modify 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.



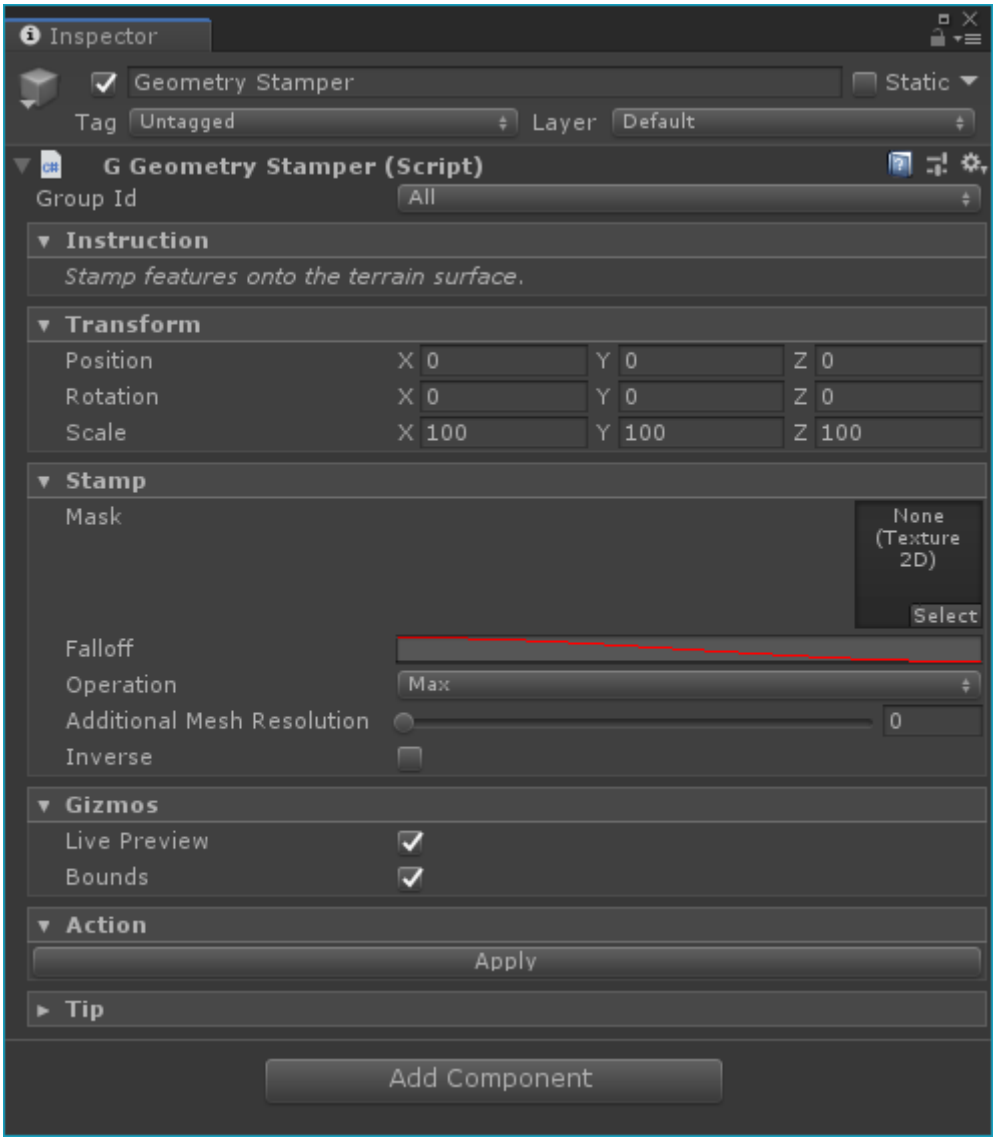
A special thing about this tool is that it allows you to see the result before apply it to the terrain, minimize the trial-and-error process.



Explore the parameters to see how it work. 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 V2>Tools>Stampers**, this will create other kind of stamper like Texture Stamper and Foliage Stamper, etc.



Detail of each property of a Geometry Stamper:

For Transform:

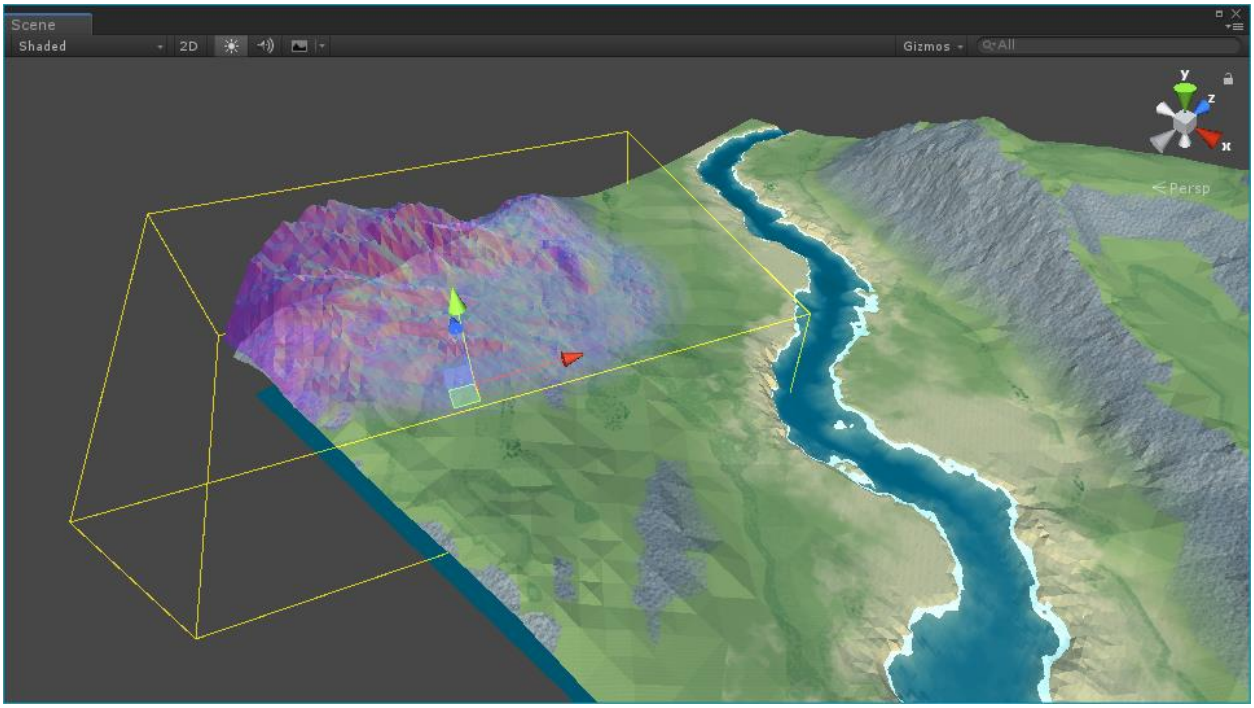
- Position: position of the stamper in the scene.
- Rotation: rotation of the stamper.
- Scale: size of the stamper.

For Stamp:

- Stamp: a mask texture represents the geometry feature to stamp
- 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.

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 mode. Depend on what you want to make, choose an appropriate operation:

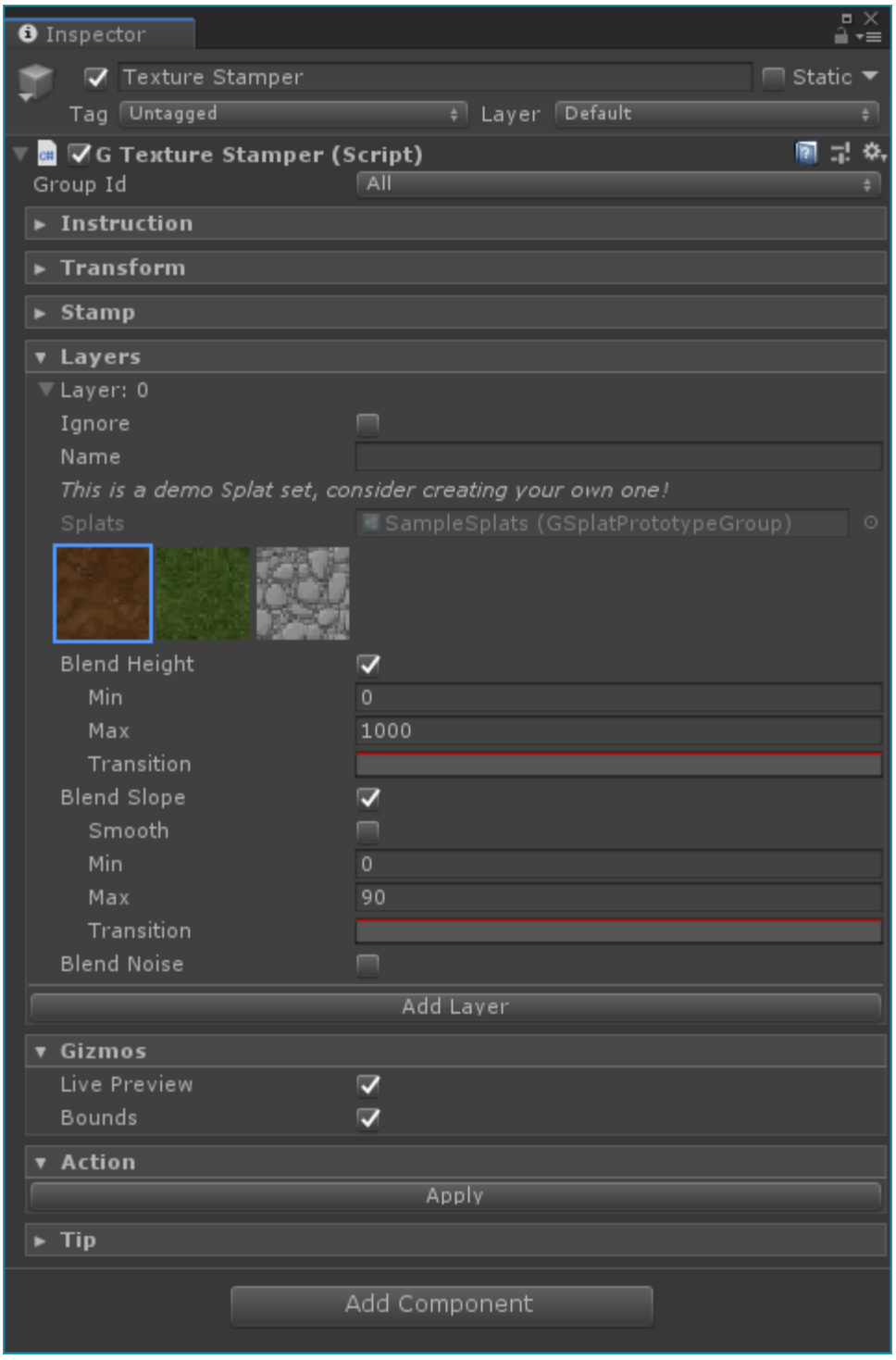
Let the result is "**h"**, current height is "**a"** and stamper height is "**b"**:

- 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 let you perform procedural texturing on terrain surface. To add a Texture Stamper, go to **GameObject>3D Object>Polaris V2>Tools> Texture Stampers**.



Texture Stamper also has basic properties as described in Geometry Stamper to define the shape. In additional, it consists of multiple stamp layer, which will be apply one by one onto the surface. To add a stamp layer, click on Add Layer.

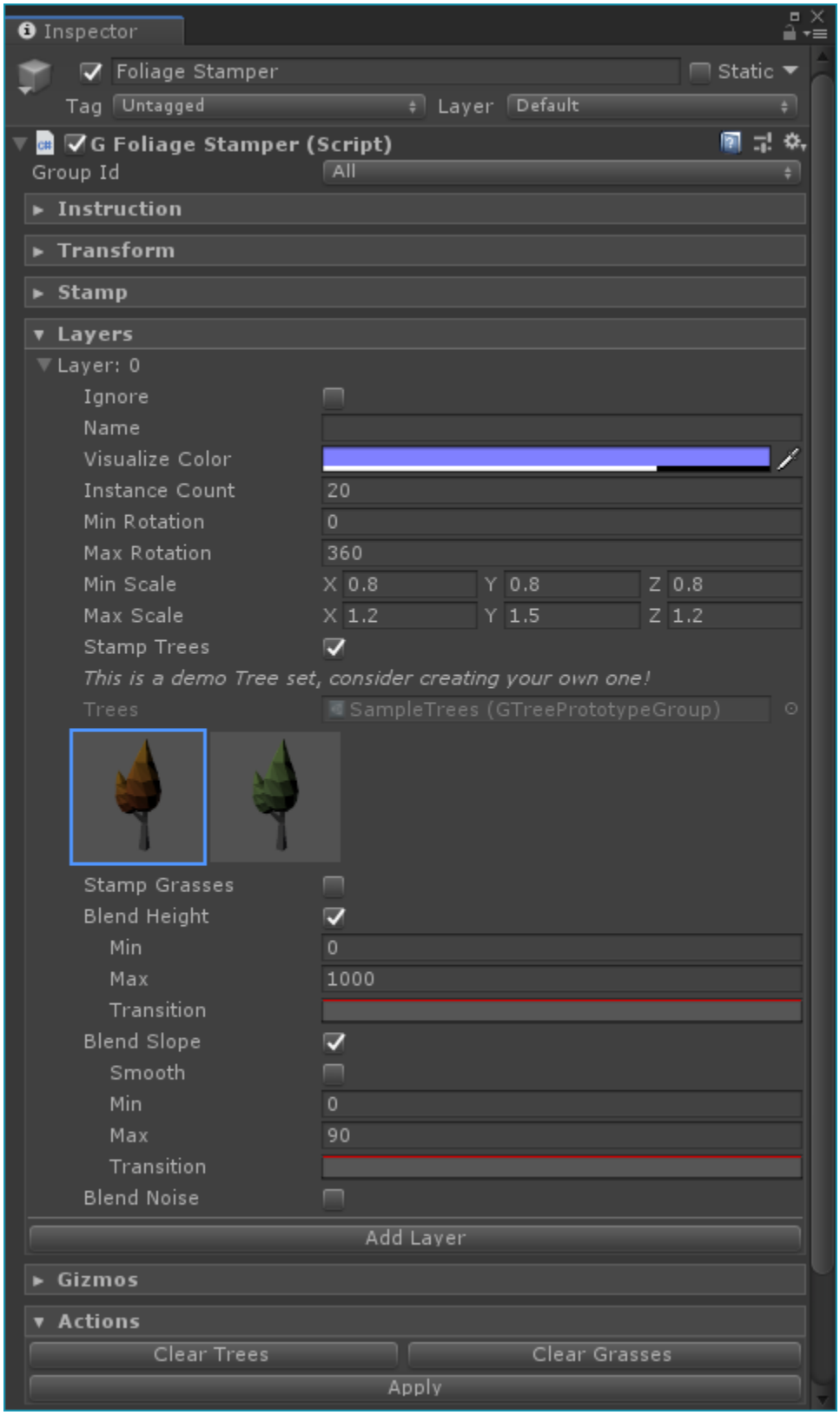
Each stamp layer has its own properties:

- 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 explanation 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.

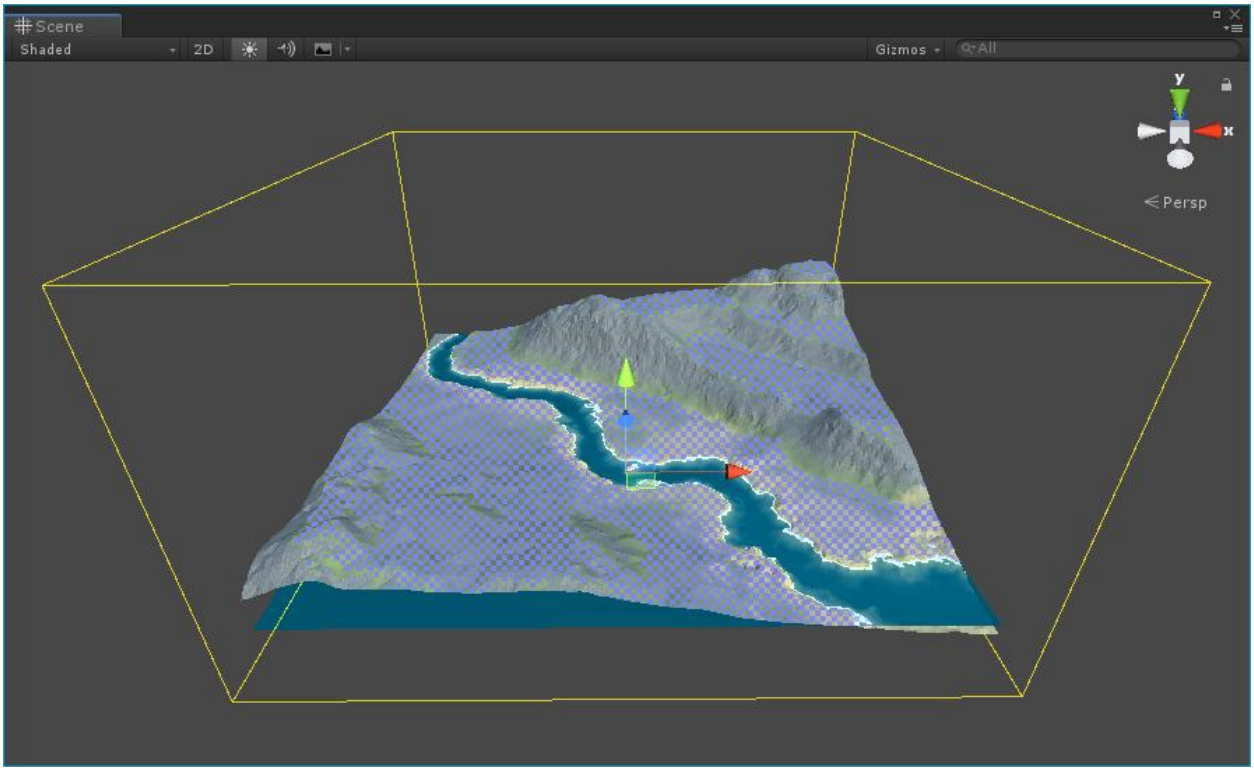
FOLIAGE STAMPER

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 V2>Tools> Foliage Stampers**.



Each stamp layer has some additional properties:

- Visualize Color: a color to visualize its mask on terrain surface.
- Instance Count: the number of instance 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 time to add even more instances into the scene.

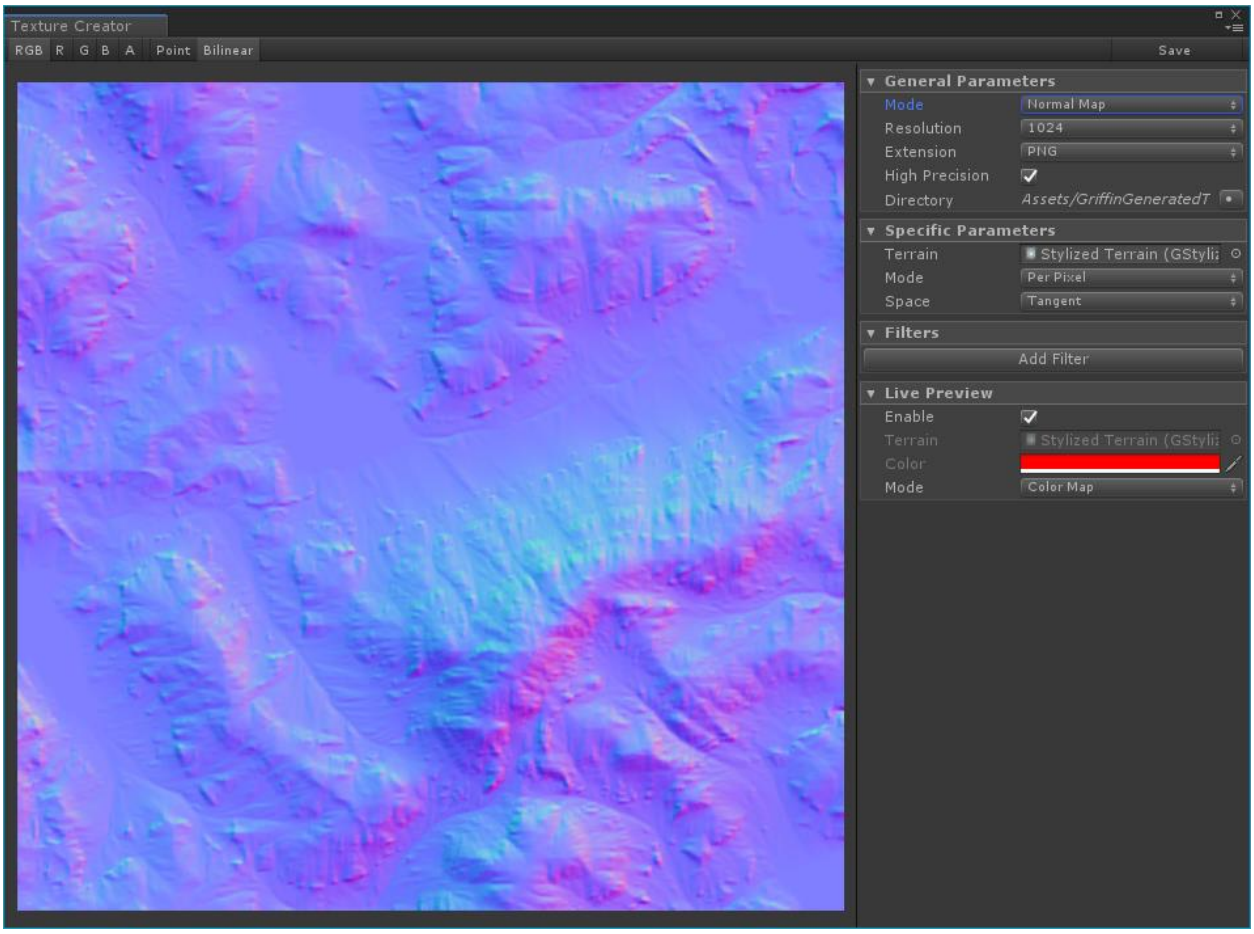
OBJECT STAMPER

This tool is similar to Foliage Stamper, but it work with prefab instead. To add one, go to **GameObject>3D Object>Polaris V2>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 V2>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 is 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 enable 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 is 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 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 adding on top of each others.
- 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 step.

Warp

Apply a normal map based image warping.

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

Live Preview

This section control 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 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

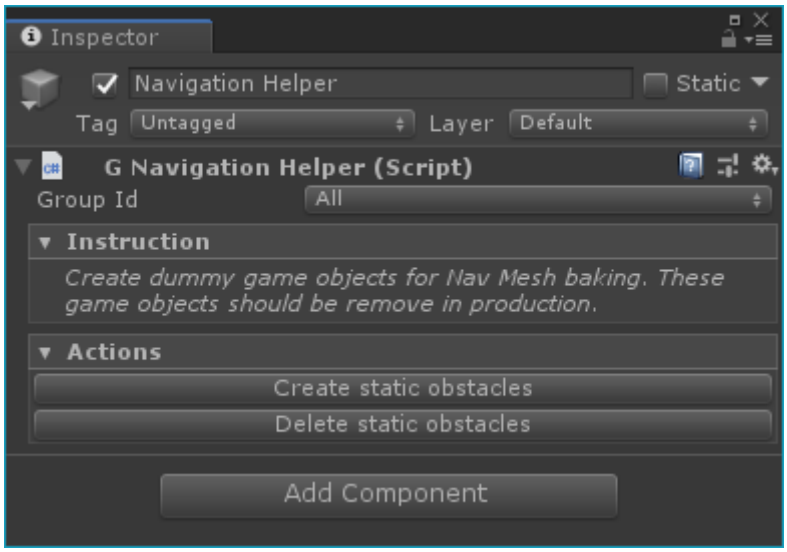
Mark geometry as static

Before perform 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 kind of component like Tree Collider.

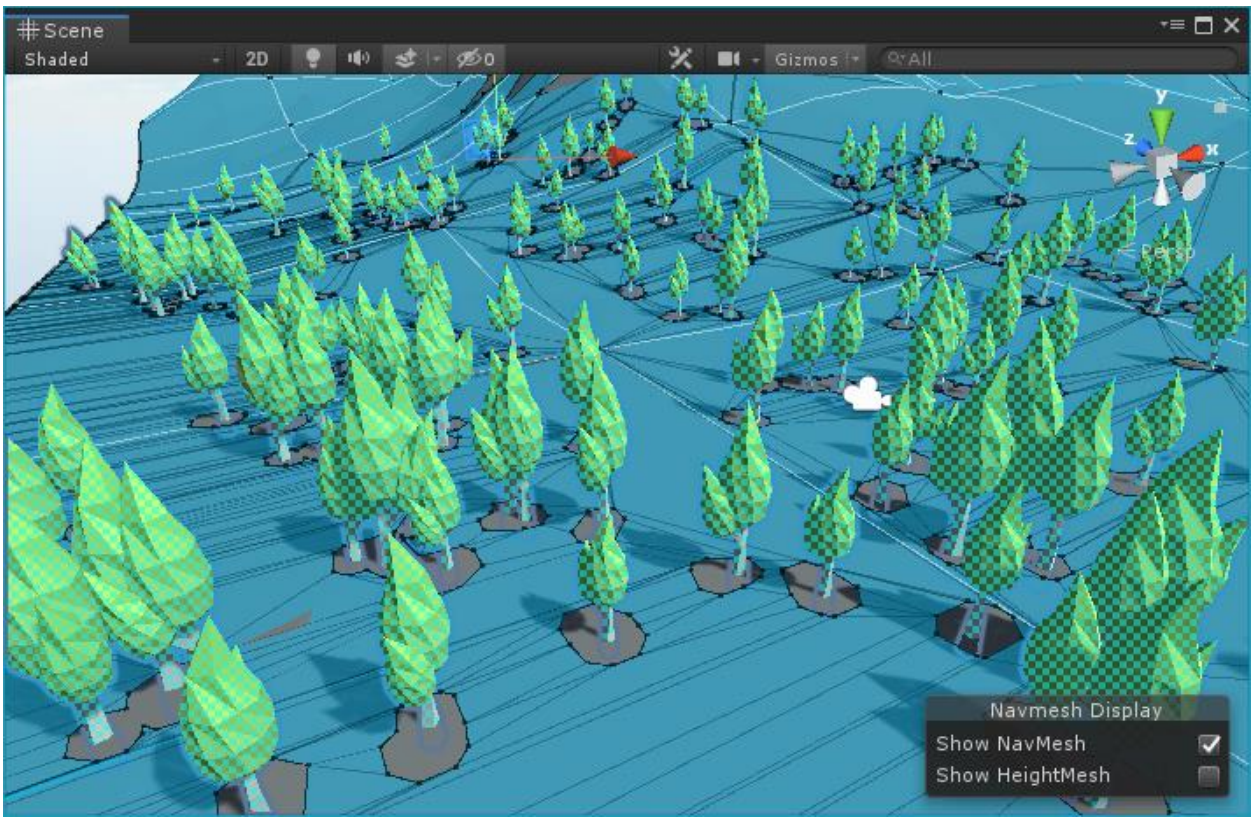
To properly mark geometry as static, go to **Window>Polaris V2>Project>Settings**, then check on **Show Geometry Chunks In Hierarchy**. A child game object named **_Geometry** will appear, **mark it as static then select "Yes, change children".**

Nav Mesh baking

Terrain geometry is automatically marked as Navigation Static so it can be grab by Unity Navigation System. However, trees rendered with Griffin are not game objects, so Unity don't know how to carve holes on the nav mesh for more precise result. Luckily, Griffin provide a helper tool to help you deal with this. Go to **GameObject>3D Object>Polaris V2>Tools>Navigation Helper** to create it.



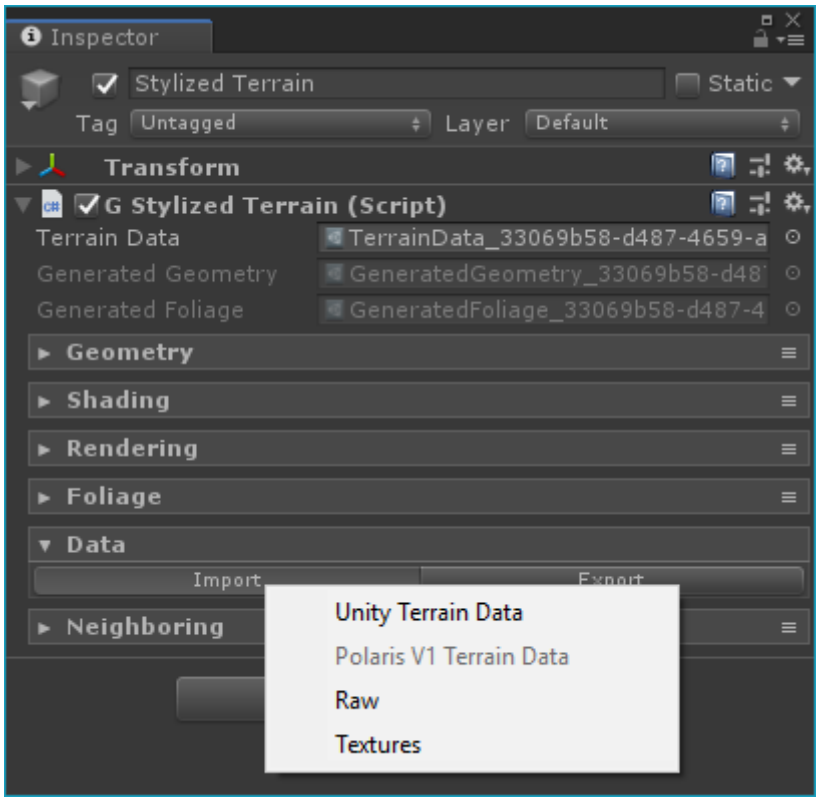
Before baking nav mesh, click on **Create static obstacles**, it will create some dummy game objects where your trees are, so Unity know how to carve holes on the nav mesh.



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.
- Polaris V1 Terrain Data: this option need Polaris V1 to be presented in your project.
- 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.

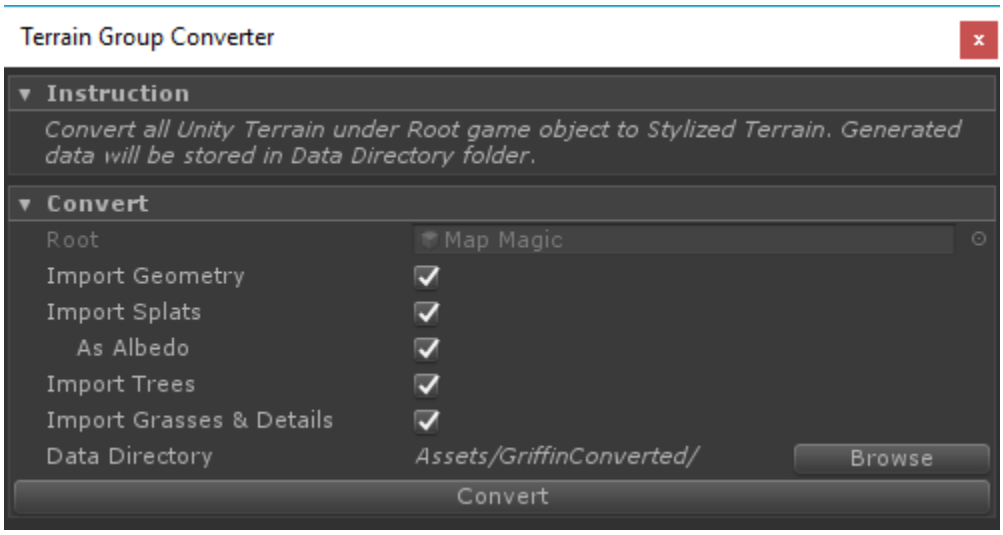
Export:

- 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

Griffin provides a handy tool to help you to easier migrate from 3rd parties' tools like Gaia, MapMagic and TerrainComposer by convert a Unity terrain group to low poly. Right click on the root game object of the terrain group, select **3D Object>Polaris V2>Convert From Unity Terrain**:



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

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

LIGHTWEIGHT/UNIVERSAL RENDER PIPELINE UPGRADE GUIDE

Griffin also support for Lightweight/Universal Render Pipeline. There is something you have to know when working with it. Note that render pipeline other than the built-in and LWRP/URP is not supported. You have to write your own shaders and upgrade it manually.

Update project dependencies

To work with LWRP/URP, you need to import the following package:

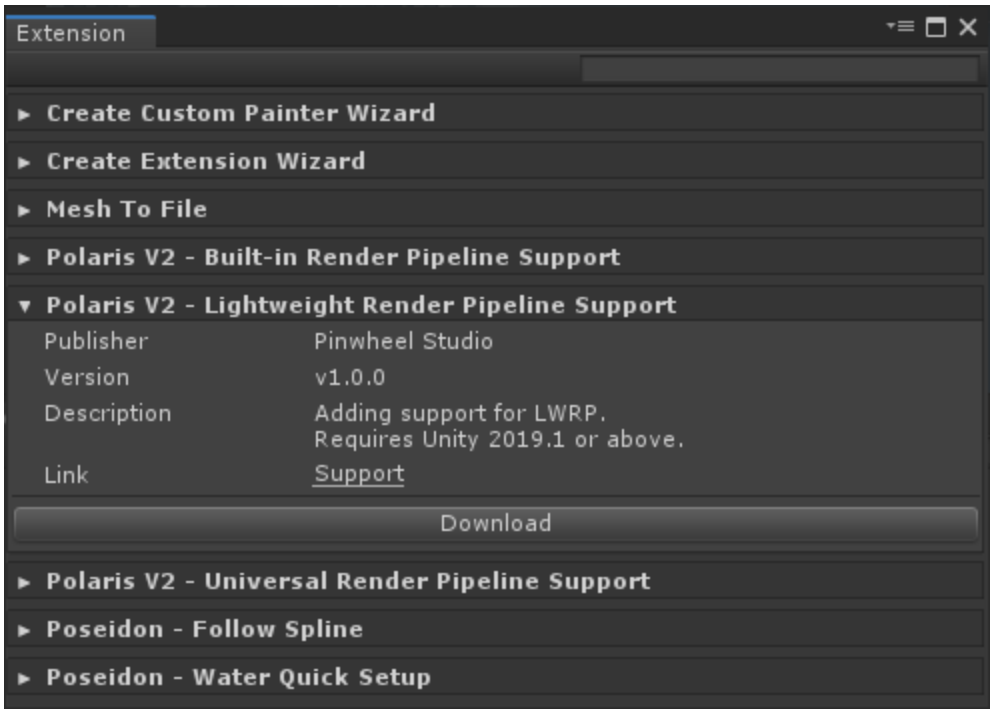
- Lightweight RP (com.unity.render-pipelines.lightweight), or
- Universal RP (com.unity.render-pipelines.universal)

Open the Package Manager to include them. After that, go to **Window>Polaris V2>Project>Update Dependencies** to re-configure Griffin.

Install LWRP/URP Support Extension

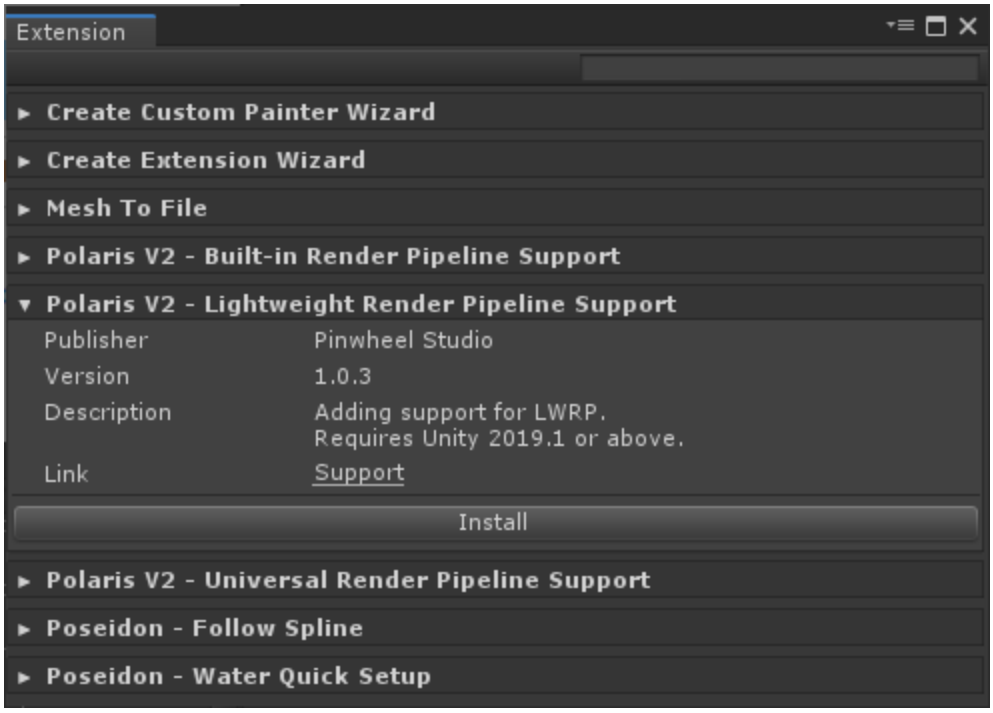
Materials and Shaders for LWRP/URP are provided in separated packages. Go to **Window>Polaris V2>Tools>Extensions** to open the Extension Window, then look for **Lightweight Render Pipeline Support** or **Universal Render Pipeline Support** extension.

Click on Download button to open the Asset Store and download the extension.



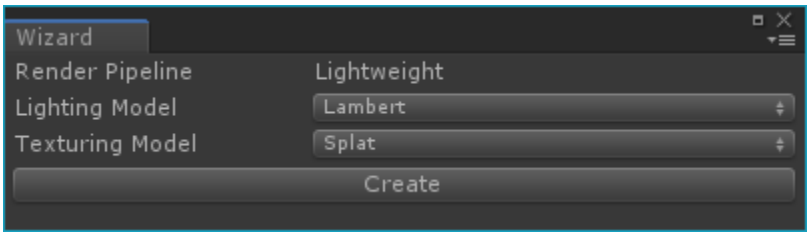
After the download is complete, import the package into your project.

Then, **re-open the Extension window and click Install**. This step is important, you will receive many error by skipping this.



Creating new terrain

It's better to use the Wizard to create a new terrain. The Wizard automatically detect render pipeline currently in use and pick an appropriate material for you:



Upgrade previously created terrain

For terrains that 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 pick up depend on your render pipeline, so nothing to worry about it.

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

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

Upgrade art asset materials

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

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

EXTENSION SYSTEM

Griffin provide a system that let you write code to add extent/utility tasks, integrate your asset, etc. and display information in a hub. Go to **Window>Polaris V2>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 V2

- Lightweight Render Pipeline Support: adding materials and shaders for LWRP.
- 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.

Writing your own extension

To be detected as an extension, your script must be placed 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 getting compiled when Griffin is imported and 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 function must have exact the same name and signature as above, and return the appropriate value to display in the extension info section. If you failed 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:

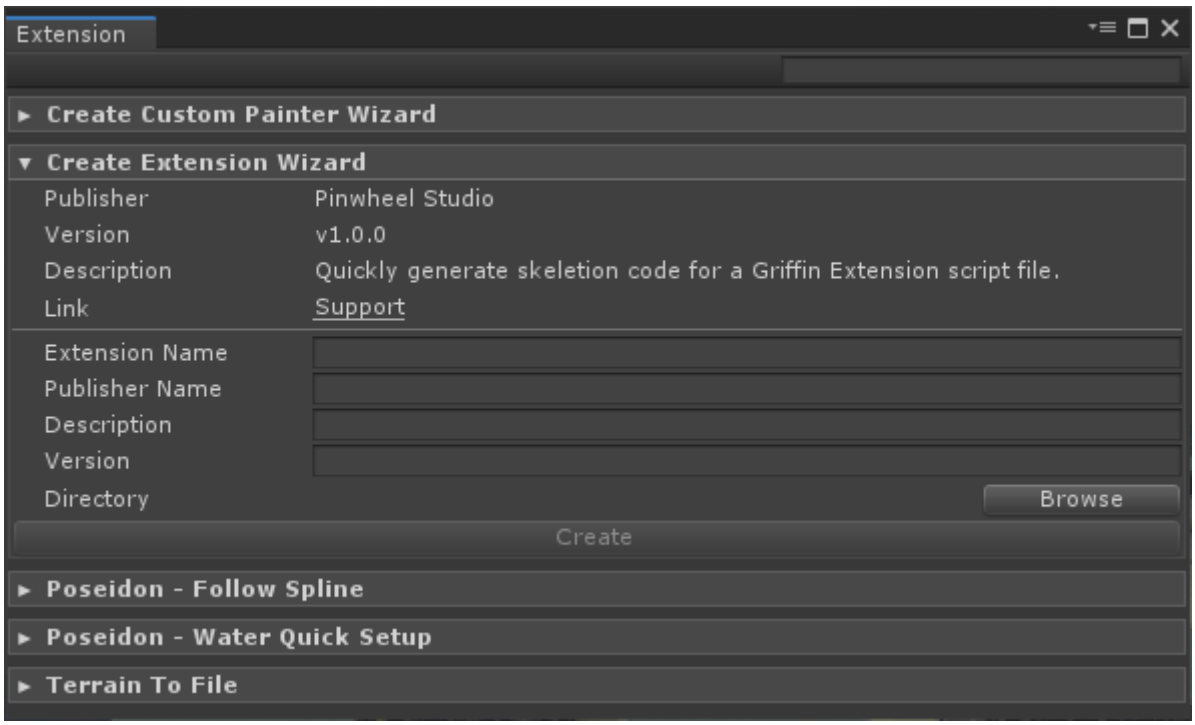
```
public static void Button_OpenConverter() {…}
```

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

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

Your GUI block will be display 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:



THIRD PARTIES ASSET SUPPORT

Amplify Shader Editor

Most of the terrain and foliage shader are made with Amplify Shader Editor (ASE), so you can visually edit and tailor them as 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

Work in progress…

MicroSplat

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

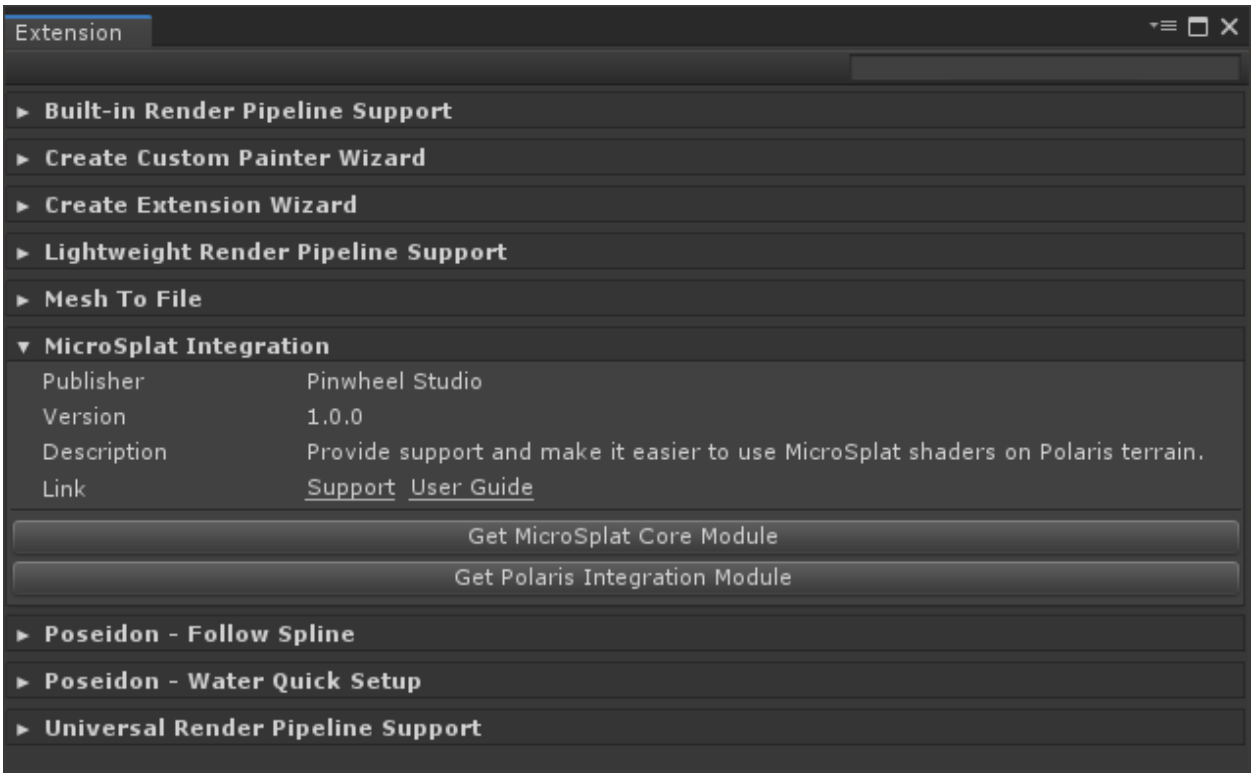
The integration also provide tools for quickly setup your terrain to be used with MicroSplat, as well as texture synchronization between the two systems, so texturing tools will work as usual.

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

Installing MicroSplat and the extension

Open the Extension window by going to **Window>Polaris V2>Tools>Extensions**.

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

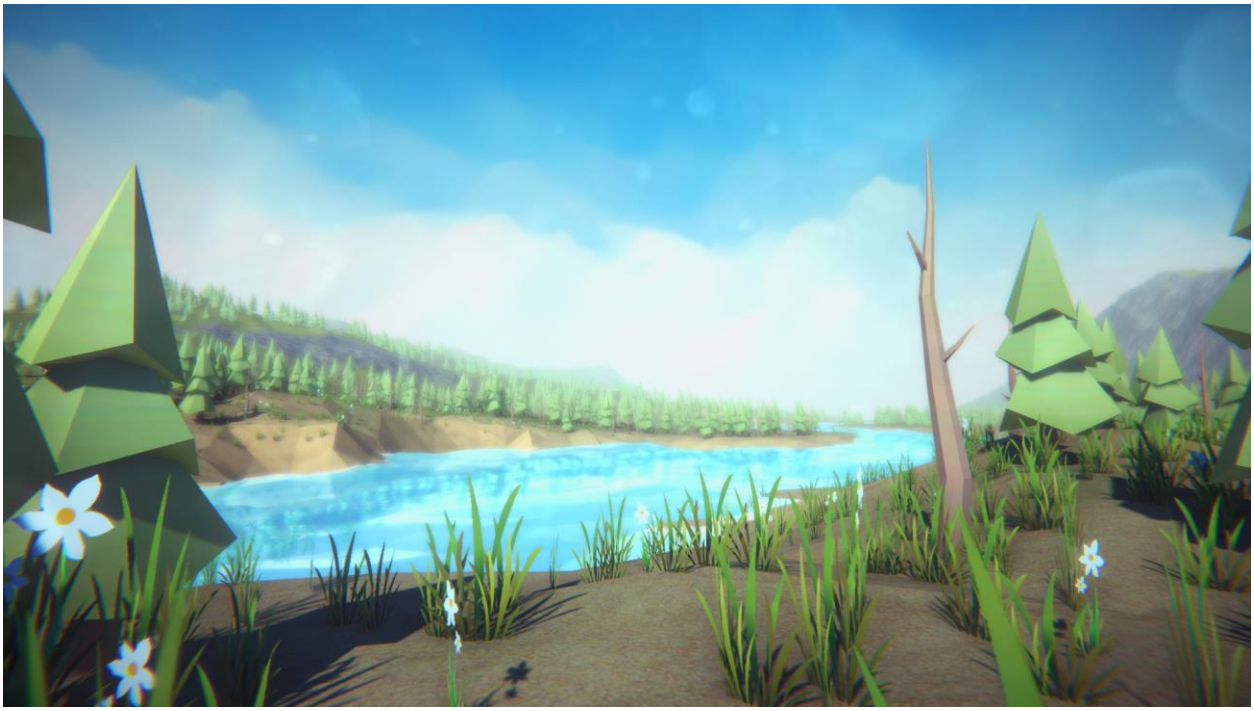


Click on **Get MicroSplat Core Module** and **Get Polaris Integration Module** to download and import necessary packages. Or you can use the following links directly:

- MicroSplat Core Module.
- Polaris Integration Module.

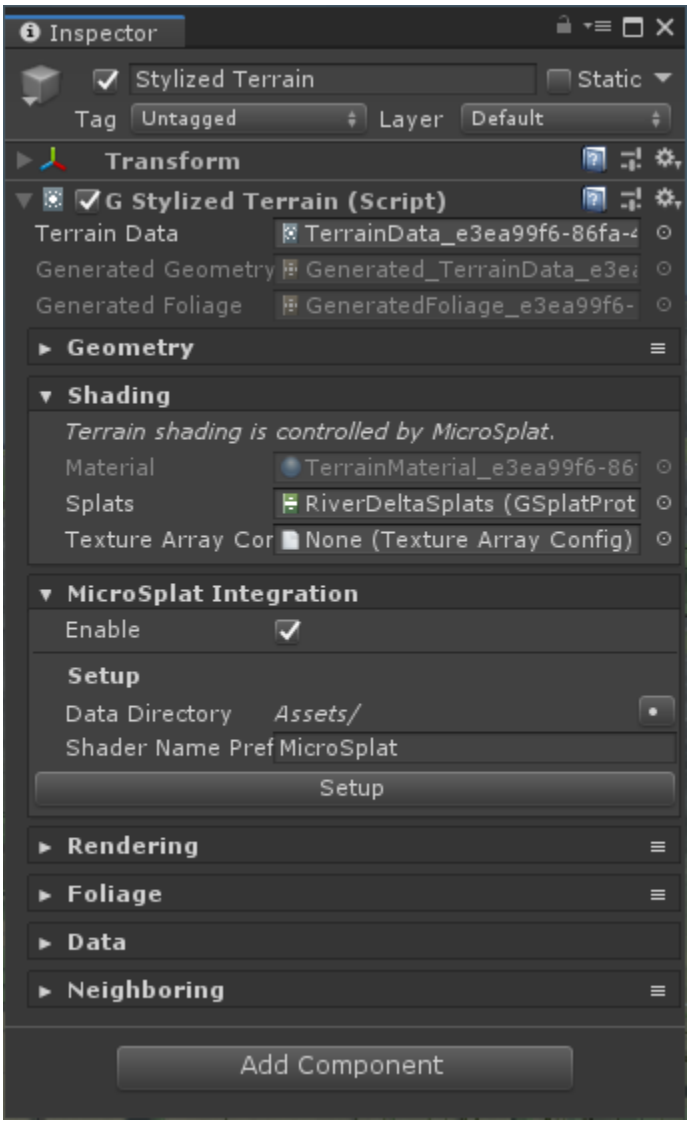
Setting up your terrain

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, and the Shading will be a bit different:



Under the MicroSplat Integration section, check on **Enable**, so the terrain will keep synchronizing with MicroSplat and not to interfere the shading process.

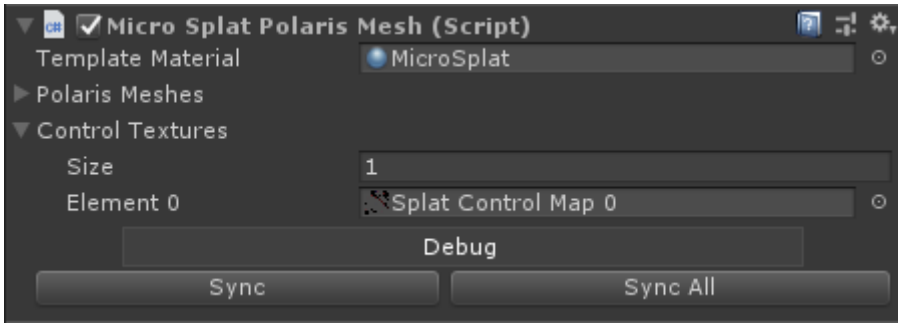
MicroSplat need a folder to store its configuration as well as generated shaders and material. Use the folder selector to pick one, it's better to select a folder outside of MicroSplat and Polaris folder.

- Data Directory: Where MicroSplat will store its data such as Texture Array Config, shaders, material, etc.
- Shader Name Prefix: A string to append to the head of shader name.

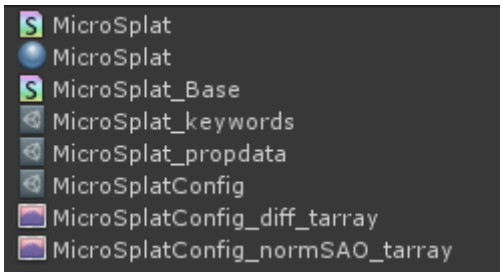
Then, click **Setup**.

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

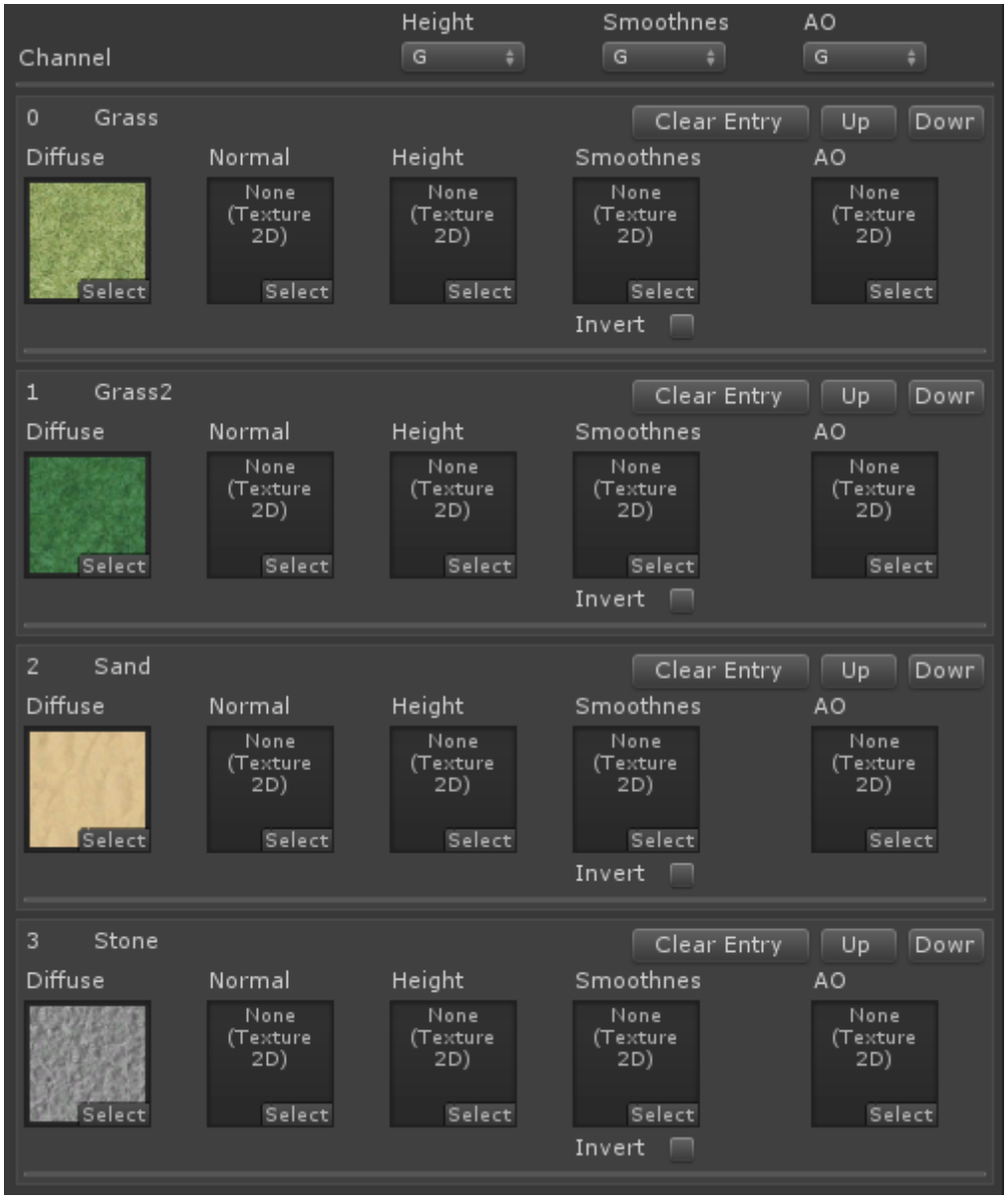
- A new component Micro Splat Polaris Mesh is added to the terrain, with the splat control map(s) assigned.



- 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 first time setup, if you want to add more or edit splat layers later, you have to use MicroSplat editor instead.

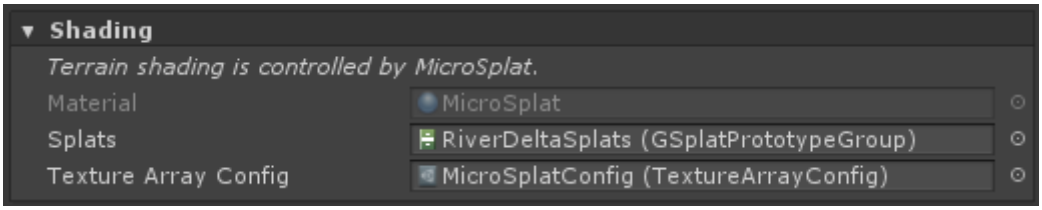


At this point, Polaris has been linked with MicroSplat for render the terrain. All tasks related to splat layers, material editing, etc. should be performed using MicroSplat editor and follow its instructions.

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.

Synchronize between Polaris texturing tools and MicroSplat

To make sure texturing tools such as painter and stamper to work as usual, simply assign a Splat Prototype Group to its slot under Shading group.



At this time, the Splat Prototype Group only act as a medium to link between the two systems and fetching preview for texturing tools. Splat layers config should be done on MicroSplat side.

Also, make sure the Texture Array Config is assigned, if not, drag and drop it there.

Now you can use Polaris texturing tool to edit the terrain as normal.

Result

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