Broccoli Tree Creator v1.9 Documentation

Concept

Pipeline

Every tree begins with a set of instructions about how to spawn branches and sprouts. A pipeline is a collection of elements holding these instructions. The pipeline can be seen as a chain of elements linked together to let data flow through this chain, each element modifying some aspect of the final tree product. It is mandatory for a pipeline to be complete in order to generate trees; this means having a starting (source) and an ending element (sink); a few other elements can be added in between too.

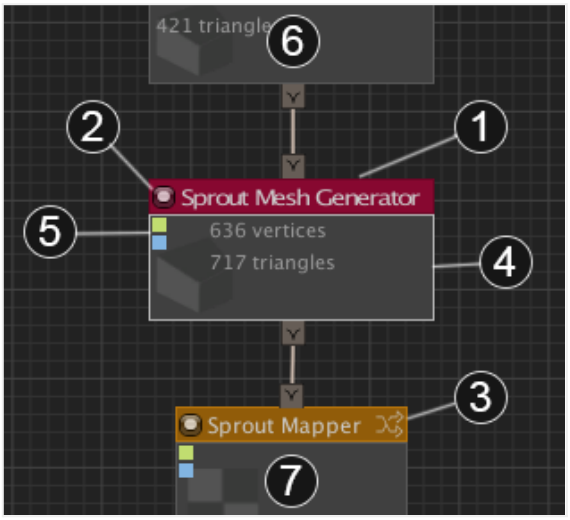
Elements

Pipeline elements have one specific function and are sequentially dependent between them. The first elements on a pipeline are structure generators, which create the basic tree data structure used by latter elements to apply meshes and texture maps. Elements are represented using nodes on the tree editor canvas view, selecting an element lets you modify its options on the inspector view. In the next section pipeline elements are described with more detail.

Pipeline Elements

Element Components

Pipeline elements are represented using nodes on the tree canvas. Each node has components to identify the element they represent, information about their configuration, pads to connect them to other nodes and controls to modify their role within the pipeline. The main components of nodes and their role as described here:



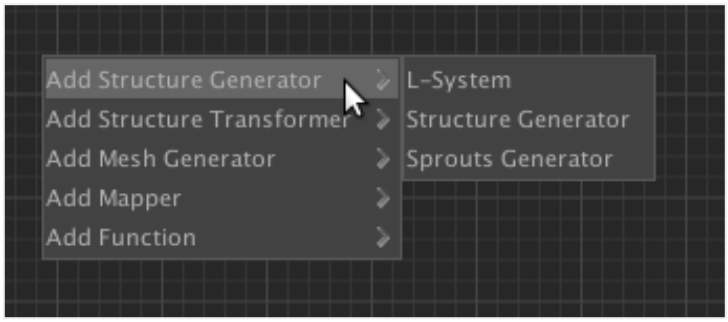
1. **Title bar:** displays the name of the element node. It has a color based on the type of aspect the element modifies on the tree.
2. **Enable/disable check:** some elements have the option to be disabled remaining part of the pipeline. Disabled elements don't participate in the processing pipeline; disabling elements is useful when testing the tree creation process.
3. **Randomized icon:** this icon indicates this pipeline element uses some level of randomization when processing a tree. This randomization is disabled if the option "Use fixed seed" is checked.
4. **Node body:** some nodes display information about the data processed by the element or its configuration.
5. **Sprout groups:** display specific configuratisons for sprout groups on elements that modify sprouts data.
6. **Source element:** the preceding element the selected element receives data from when it is connected to a pipeline.
7. **Sink element:** the following element connected at the sink pad of the selected element.

Node Operations

Nodes are entities created in the tree canvas context. When creating a new Tree Factory object its canvas comes with the most basic pipeline (or you can select one from the catalog); in order to extend any existing pipeline capabilities, more elements could be integrated to the pipeline or modify these elements properties. Additionally to the operations described below, the nodes are movable around the tree canvas.

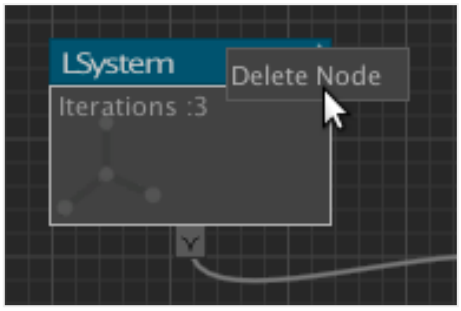
Create

Alt-click on an empty area of the tree canvas brings a context menu to select the elements available to add. The main menu options are the classes of elements available (see below for Elements by Class).



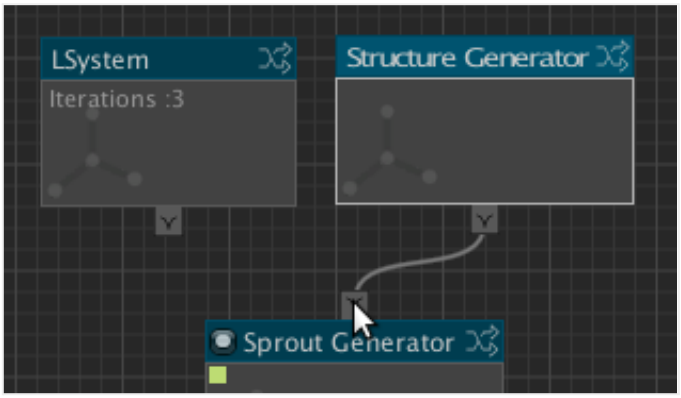
Delete

To delete a node within the pipeline alt-click on its title or its body, a context options is displayed to selected is deletion. Take account that deleting elements connected to other elements within a functional pipeline will render this pipeline incomplete.



Connect

Element nodes display pads either at their top border or at their bottom border. These pads are used to connect elements. To connect elements just click on the sink pad of an element node and drag the pointer to the source pad of the element node aiming to connect to; the same method is used in reverse to disconnect element nodes. Take account that elements have restrictions on the elements they can connect to, thus an invalid connect will not be applied.



Elements by Connection

There are three categories of element nodes when making connections on a pipeline.

Source Elements

These element nodes are the first node on a pipeline. Source elements provide the most basic data structure used later by other pipeline elements to build the final tree.

Transform Elements

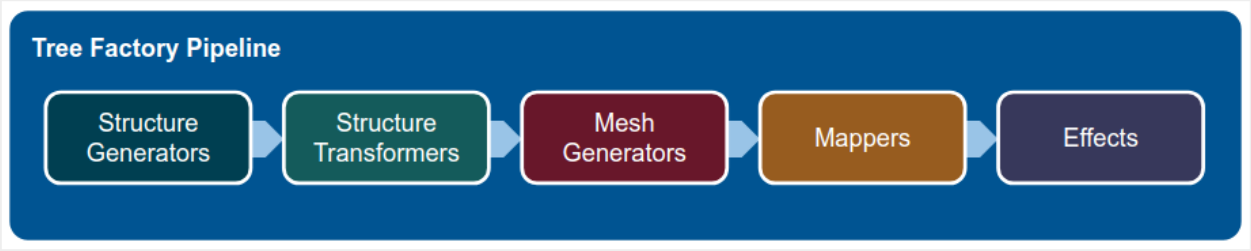
Transform elements comprise most of the element nodes within a pipeline. These elements apply changes and some add randomization to the final tree asset.

Sink Elements

These elements are the last node on a pipeline. Are used to place the final tree within the scene.

Elements by Class

Element nodes are placed within the pipeline using their class group as an order constraint. Elements that belong to the same class but have different tasks within the pipeline also have an order to follow; this is the global order of element nodes for a pipeline based on their class:



Structure Generators

Generators provide the basic data structure of branches and/or sprouts to be modified or used as input for other elements downstream. There are generators for branch hierarchy, for sprouts or generators that provide both entities.

Structure Transformers

Transformers take branches and/or sprout data as input and apply operations on them. These elements are useful to modify aspects of a tree, such as: branch length, branch girth, branch hierarchy, branch bending, sprout angle, sprout direction and sprout size.

Mesh Generators

These elements create the mesh around the branch and sprout data.

Mappers

These elements apply existing materials or create new ones from textures to be assigned to the tree mesh. Mappers mostly work with textures, materials, shaders and modify UV and color channels on the meshes.

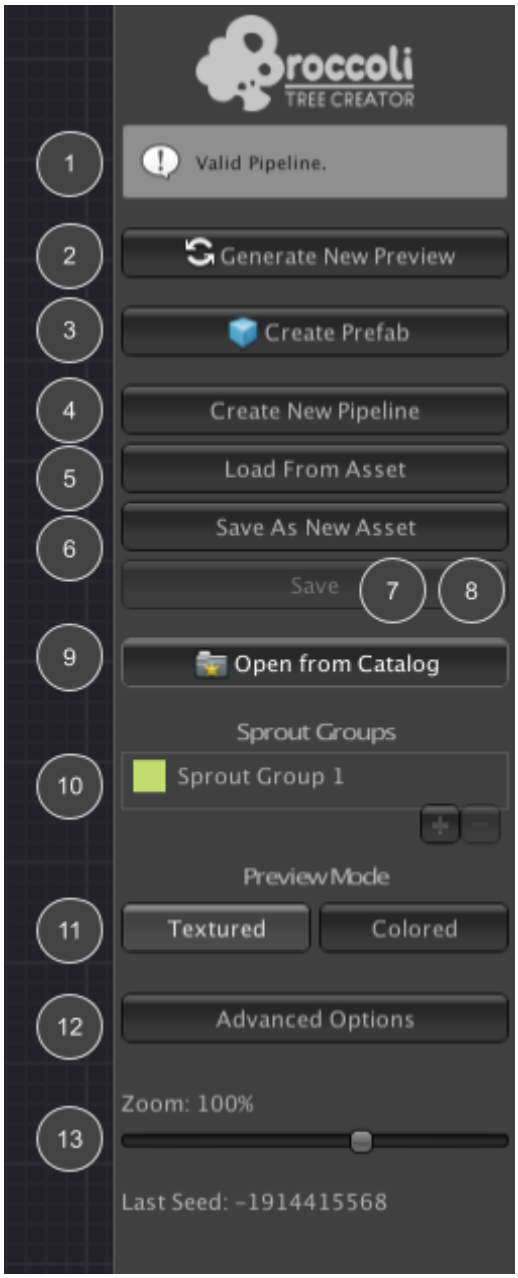
Effects

Apply vertex data and positioning on the final tree product.

To see all the available options on each pipeline element please refer to the node description section on the documentation.

Factory Options

Operations available for the tree factory are displayed on side panel next to the node canvas.



1. Pipeline Status: displays the status of the pipeline; every time a change is made on the pipeline the factory runs a validation on it and displays the result here. Only valid pipelines can produce tree assets.

2. Generate New Preview: if a valid pipeline is present at the tree node canvas this option comes available to produce a preview tree on the scene. Depending on the number of elements with randomization are connected in the producing pipeline, each time this button is pressed a different tree should be previewed.

3. Create Prefab: creates a prefab asset out of the current preview tree on the scene generated by a pipeline. Several prefab options are available at the Factory Options side panel.

4. Create New Pipeline: cleans the tree node canvas and initializes a new pipeline ready to be extended. Make sure you save changes if previously working on a pipeline you need to persist.

5. Load From Asset: opens a file dialog to select an asset file where a pipeline has been saved and loads it on the tree node canvas.

6. Save As New Asset: saves the current tree node canvas to a new asset file.

7. Save: when a tree node canvas has been loaded from an asset file this option persist any changes made to the pipeline to that particular asset file.

8. Asset Path: displays the path of the asset file the pipeline was loaded from.

9. Open from Catalog: opens the Catalog Side Panel displaying a collection of pipeline available to load on the tree node canvas.

10. Sprouts Groups: displays the list of sprout groups registered on the pipeline and provide options to add new ones or delete existing ones.

11. Preview Mode Options: two options are available when previewing trees:

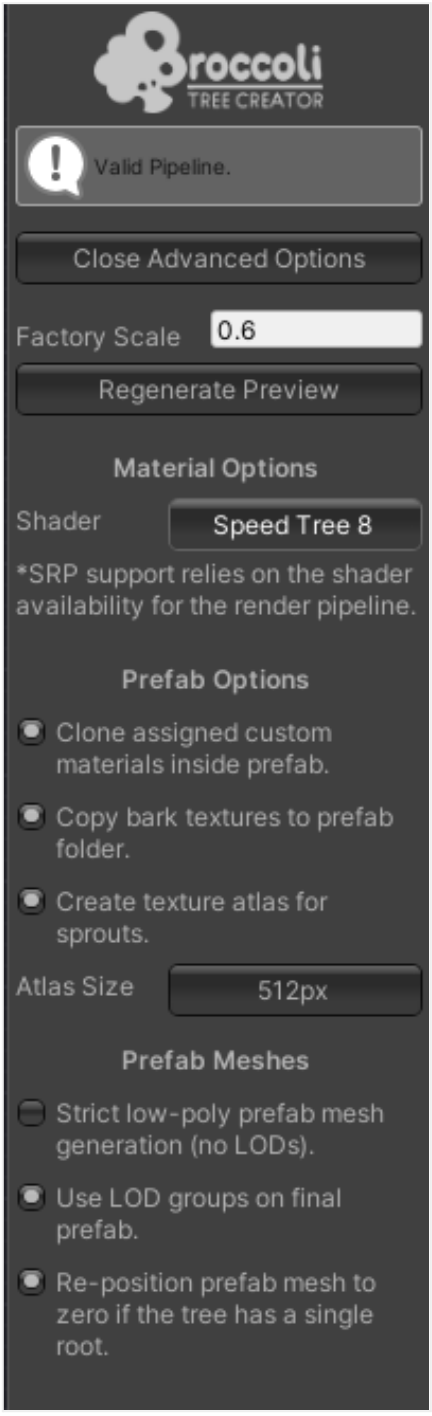
a) *Textured*: shows the preview tree with textures and materials applied to its submeshes.

b) *Colored*: submeshes on the preview tree are shown using colored areas (based on sprout groups).

12. Advanced Options: opens the Factory Advanced Options Side Panel with more advanced options for the factory.

13. Zoom: slider for the zoom factor on the tree node canvas area.

Advanced Factory Options



The Advanced Factory Options side panel is displayed when clicking on the Advanced Options button on the main side panel.

Advanced Options

Factory Scale: controls the scale of the produced tree. Values within the pipeline elements remain the same but the final tree mesh is multiplied by this factor.

Regenerate Preview: commands the factory to build anew the preview tree using the same set of instructions without applying randomization. For example, this is useful to build new materials when switching Scriptable Render Pipelines.

Prefab Options

These options describe how the prefab creation process should handle resources such as textures, materials and atlases.

1. *Shader*. Selects the prefered shader flavor to use on the materials. Either if you let the factory build the materials or you assign custom ones, the parameters to set on their shaders are set based on this option. The final shader flavour to use depends on your Unity version, if the one selected is available then that is the shader the factory will use, if the prefered shader flavor is not available then a fallback version is going to be used. The available shader flavors and their fallback order is:
    i. SpreedTree8 (or similar),
    ii. SpeedTree7 (or similar).

When selecting an option for a compatible shader type (like SpeedTree8 compatible) you should provide these shader in an additional field that will appear right beneath this option; the properties of this shader will be set just like if it was one of those that come with Unity (either a SpeedTree7 or SpeedTree8). One thing to consider is: although Broccoli Tree Creator supports SRP (Scriptable Render Pipeline) shaders, some of these shaders do not support global illumination on trees, so you need to use a custom shader to have proper illumination (otherwise the trees will have hard pitch black shadows).

Clone assigned custom materials inside prefab. When mapping elements within the pipeline have reference materials assigned to be applied to the final tree prefab, this option clones those materials and makes them part of the hierarchy of the resulting prefab.

Clone assigned custom materials inside prefab. When mapping elements within the pipeline have reference materials assigned to be applied to the final tree prefab, this option clones those materials and makes them part of the hierarchy of the resulting prefab.

Copy bark textures to prefab folder. If a texture is assigned to be applied to the tree trunk, this option copies that texture to the prefab folder. If the option is unchecked then the material used to create the bark points to the original assigned texture. Leaving this option unchecked is useful to let several tree prefabs share the same bark texture on their materials.

Create texture atlas for sprouts. When using several textures assigned to the sprout groups this option creates a texture atlas to be used by the prefab materials.

Atlas Size. Sets the dimensions for the sprout texture atlas.

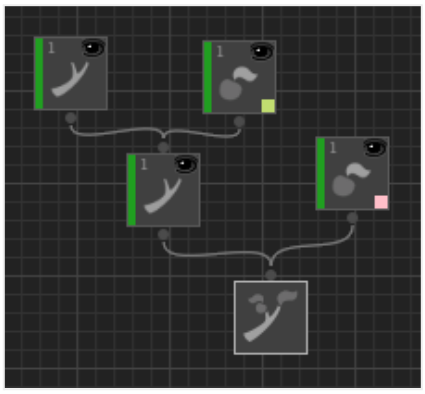
Prefab Meshes

Billboard Texture. Set the desired texture size for the billboard on the Prefab.

Re-position prefab mesh to zero if the tree has a single root. When using custom positions to spawn trees the mesh generated has an offset relative to the tree factory position; this option sets the tree mesh origin back to zero.

Structure Generator Node

This structure generator element contains nodes connected using a hierarchy setting rules to generate trees. The whole structure is described using level nodes in a hierarchy, each node contains properties to generate the branches at the assigned level. The structure here generated is taken as a base to be modified by other nodes downstream the pipeline, meshed and textured. Take this structure as the spatial data the tree will be built upon.



Root Node

Contains the properties to create the root branches that will sprout the subsequent branch levels.

Root Node

Max frequency

The maximum number of root branches to generate as the upper range limit on randomization mode. Each root branch follows the hierarchical structure described by the nodes.

Min frequency

The minimum number of root branches to generate as the lower range limit on randomization mode. Each root branch follows the hierarchical structure described by the nodes.

Max Length

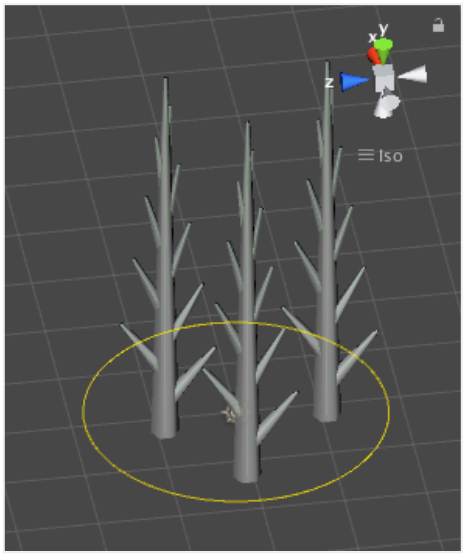
The maximum length value of any generated root branch as the upper range limit on randomization mode.

Min Length

The minimum length value of any generated root branch as the lower range limit on randomization mode.

Radius

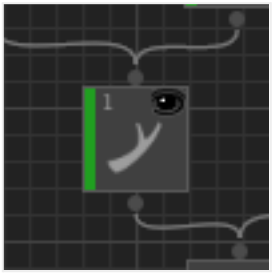
Radius for the circular area where the root branches will sprout.



Branch generation with radius = 1.7 and frequency = 3

Branch Node

Contains the properties to create children branches along the length of a parent branch. Multiple level nodes could be applied to any level node or root node, creating a parent-child relationship on the hierarchy. It is possible to disable a level node while still maintaining it in the hierarchy, this will disable all other nodes derived from it as well.



Branch Node

Max Frequency

Similar to the root node frequency option establishes the upper range limit of branches to generate on the parent branch.

Min Frequency

Similar to the root node frequency option establishes the lower range limit of branches to generate on the parent branch.

Probability

Establishes the odd of occurrence on the structure level. A value of 1 means the structure level will always be processed, a value of 0 means no processing at all. The probability value is display on the structure nodes at the top-left corner.

Distribution

Sets the modality of branch distribution along the parent branch. These modes are based on Unity's Tree Creator branch distribution options, so if you are familiar with it you'll be with these modes. There are four modes to select from:

Alternative

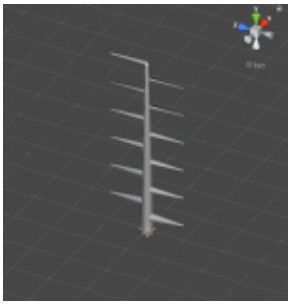
Children branches begin at the tip of the parent branch and are position-spaced along the parent branch towards its base. Each subsequent child branch takes the opposite direction of its predecessor branch.

Opposite

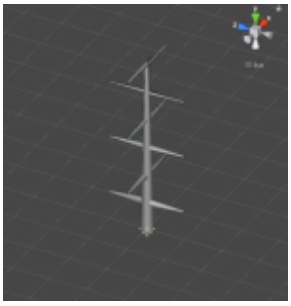
Branches come in pairs facing opposite direction at the same length position on the parent branch. The first pair begins at the tip of the parent branch.

Whorled

Children branches are grouped on nodes with n number of branches each. The **step** property sets the number of branches per group. The first group begins at the tip of the parent branch.

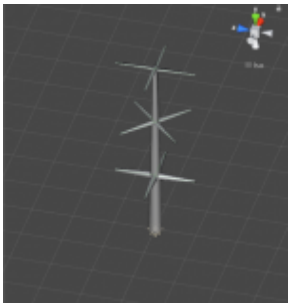


Alternative mode.



Opposite mode.

Whorled mode (step = 3).



Whorled mode (step = 4).

Distribution Spacing Variance

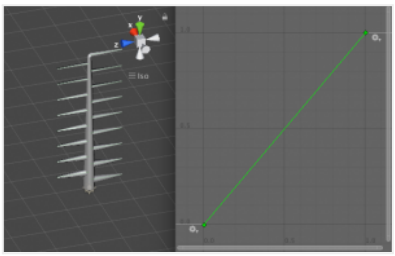
Adds a random displacement offset for the originally calculated position for a branch. The value 0 keeps the original position and a value of 1 randomizes the position within the neighbor branches limits.

Distribution Angle Variance

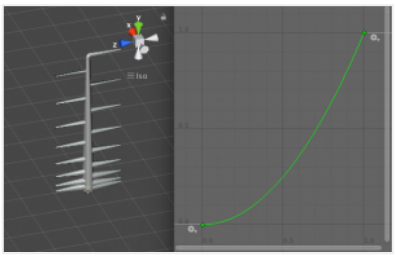
Adds a random displacement radial angle taking the parent branch as axis for the originally calculated direction for a branch. The value 0 keeps the original directional angle and a value of 1 completely randomizes the direction.

Distribution Curve

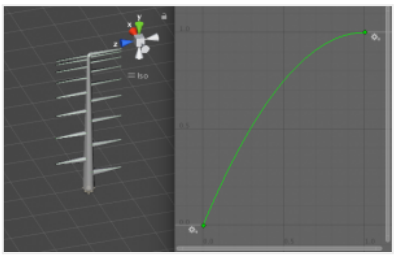
This curve controls the spacing of children branches or groups of branches along the parent branch; on the x axis 0 represents the base of the parent branch and 1 the tip of it.



Linear distribution curve.



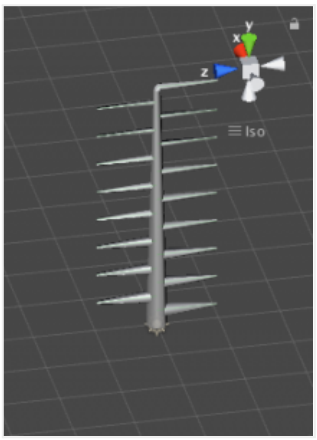
Distribution towards base.



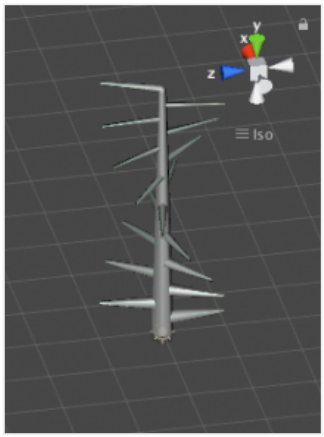
Distribution towards top.

Twirl

Rotation value for the children branches accumulative along and taking as axis the parent branch.



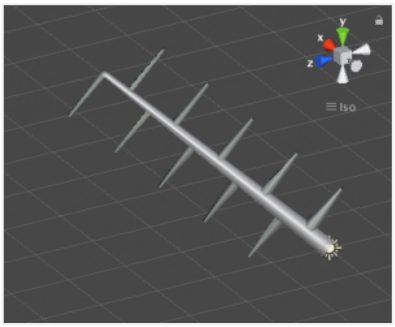
Branches with twirl = 0



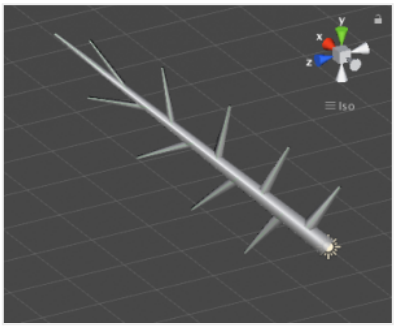
Branches with twirl = 0.15

Parallel Alignment (at Base, at Top, Curve)

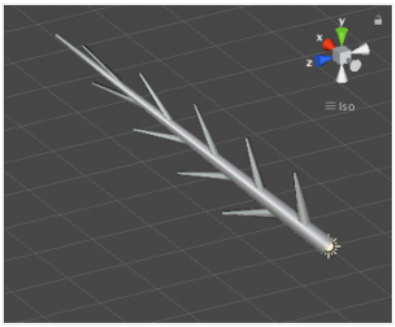
Interpolates the direction of a child branch with its parent direction. When 1 the child branch points exactly to the same direction as its parent branch does (thus the parallel name). Parallel alignment value requires a property that tells how much interpolation is going to be applied when branches are positioned at base and at top of the parent branch, positions in between are obtained using the parallel align curve property. Negative values on the properties point to the opposite direction of the parent branch.



Branches with no alignment modifications.



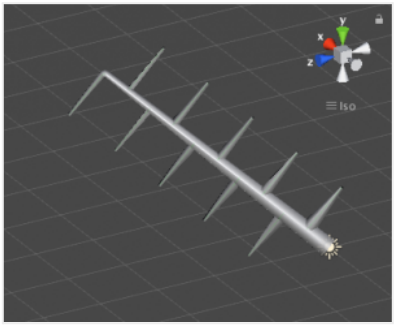
Branches with parallel align at top = 1



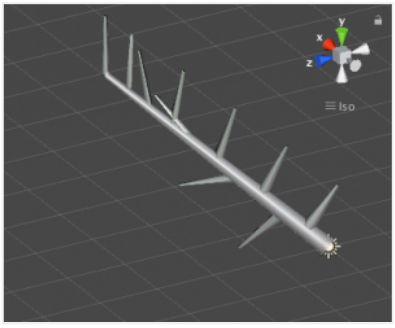
Branches with parallel align at top = 1 and at base = 0.5

Gravity Alignment (at Base, at Top, Curve)

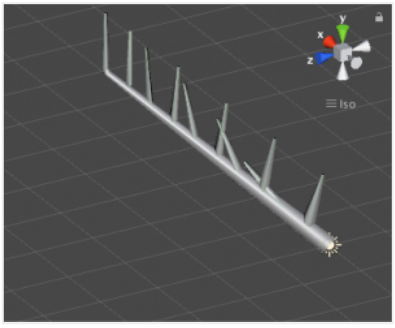
Interpolates the direction of a child branch with the vector against gravity (direction up). When value is set to 1 the child branch points up (by default the against gravity value is set to up). Gravity alignment value requires a property that tells how much interpolation is going to be applied when branches are positioned at base and at top of the parent branch, positions in between are obtained using the gravity align curve property. Negative values on the properties point to the gravity direction (down vector).



Branches with no alignment modifications.



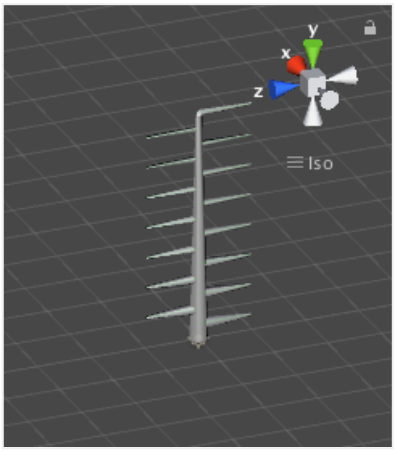
Branches with gravity align at top = 1



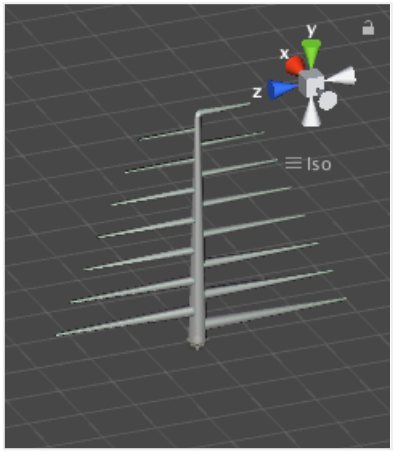
Branches with gravity align at top = 1 and at base = 0.5

Length (at Base, at Top, Curve)

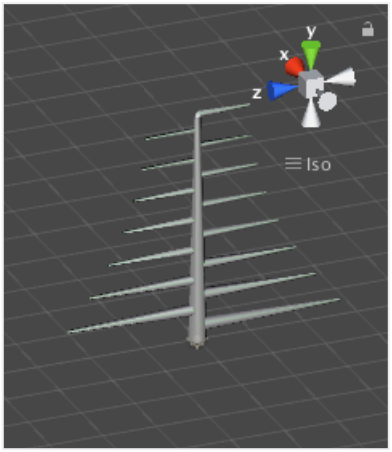
Controls the length of the children branches. A length value is required at the top (1) and at the base (0) position of the parent branch, any value in between is interpolated using the length curve property.



Branches with length at base = 1 and length at top = 1



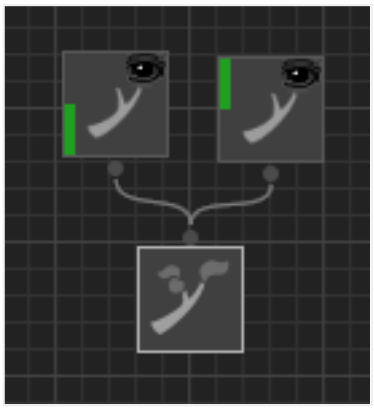
Branches with length at base = 1 and length at top = 3



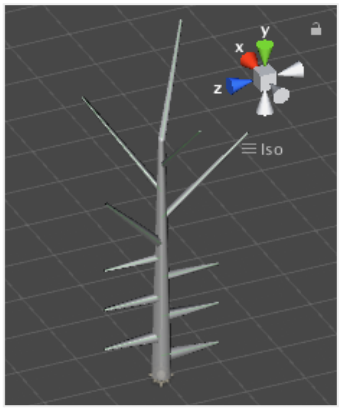
Branches with length at base = 3, length at top = 1 and length curve applied.

Action Range

This option lets you specify the range the level branches are going to be sprout along the parent branch. It is represented by a green bar at the left of the level node on the canvas view; the bottom of the bar represents the base of the parent branch and the top the tip of it. Level nodes action ranges can overlap on a parent; ideally it lets you have a higher degree of control on the level or detail or the topology of a tree. For example, in some cases you might want to have more detailed branches at the tree base level (near ground) and lower or more scarce branches at the top of it.



Level 1 nodes sharing 50% of the parent branch length.



Two different offspring on the parent branch.

Sprout Node

Contains the properties to create children sprouts along the length of a parent branch. In order to visualize sprouts on the scene view and apply a mesh to them, the node must be assigned to a sprout group; the assigned group is visible with a colored square on the bottom-right corner of the node. The sprout node is considered a terminal level on the tree structure hierarchy and is possible to disable it while still maintaining it in the hierarchy.

Sprout Node

Sprout Group

Assigns the generated sprouts to a sprout group, this is required for all sprouts to enable meshing and mapping on them.

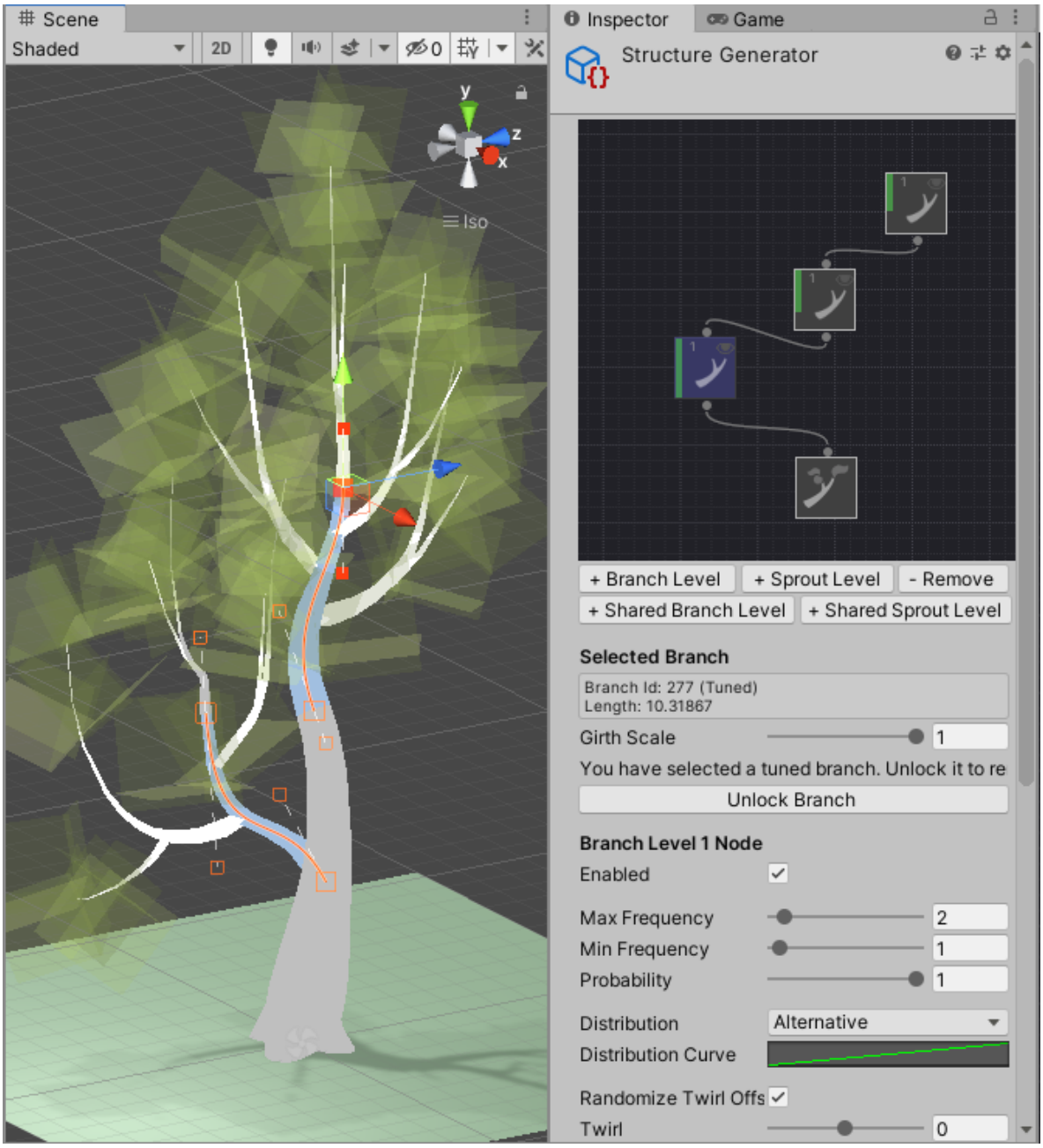
Max Frequency, Min Frequency, Probability, Distribution, Distribution Spacing Variance, Distribution Angle Curve, Distribution Curve, Twirl, Parallel Align (at Base, at Top, Curve), Gravity Align (at Base, at Top, Curve), Action Range

The same parameter effects on branches apply to sprout generation.

Branch Customization

Branches are modeled after bezier curves, Broccoli Tree Creator offers you the option to further tune the length and shape of a branch by editing its bezier curve manually.

Right after you select the Structure Generator Node the preview tree switches materials to better visualize the tree structure and displays editor gizmos to customize the branches.



Structure view on the preview tree with branch level 1 selected.

When on structure view the sprouts of the tree are rendered semi-transparent. Branches are rendered using colors depending on their state:

1. White: non customized branch, susceptible to change by randomization if a new version of the tree is generated.
2. Gray: customized (locked) branch, this means the bezier curve of the branch has been modified so that it is guaranteed it will be kept on the tree structure with the same length and shape after generating new tree previews.
3. Light blue: highlights branches that belong to a selected branch node, this lets you tune these branches using the bezier curve gizmos.

In order to tune branches you need to select the level structure node the branch belongs to and then click on any of the bezier nodes of the branch. Modifying the node position or its handles is an

undoable operation (using ctrl or cmd + z). Tuning also includes editing the branch girth scale, which is shown with a slider on the Structure Generator Node inspector; this scale is applied to all the descending branches of the tuned branch.

After a branch has been tuned it becomes ‘locked’ with all of its parents down to the root of the tree (locking the parents ensures all the lineage of branches gets spawned up to the tuned branch). Tuned branches can be brought back to randomized ones by clicking the “Unlock Branch” on the Structure Generator Node inspector.

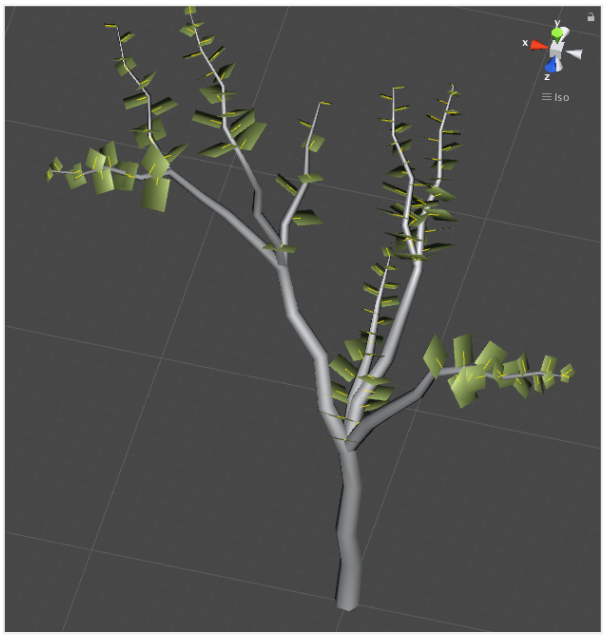
Sprout Generator Node

This generator spawns sprouts on an existing tree branch structure. Each sprout generator node on the pipeline specifies a pattern of distribution and alignment for the sprouts, which could be assigned to one or more sprout groups randomly. It should be noted that sprout orientation is not completely defined on this node, for other alignment properties might be set depending on the kind of mesh the sprout uses (these are defined on the sprout mesh generator node).

Max Frequency, Min Frequency

Sets the maximum and minimum number of sprouts to generate on a particular branch or sequence of branches. The final number of sprouts on every branch is a randomized number between max and min value.

Min and Max frequency = 6



Min frequency = 6 and max frequency = 20

Distribution

Refers to sprout distribution on sprouts nodes. Each node of n sprouts is positioned along its parent branch as a unit and these sprouts have an axial angle between them using the parent branch as axis.

Alternative

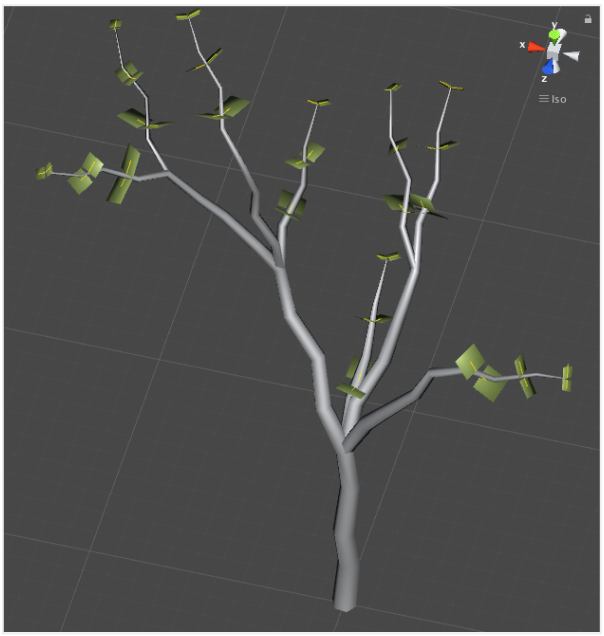
Each sprouts node has a single sprout, subsequent sprouts nodes have an angle offset of 180 degrees compared to its predecessor.

Opposite

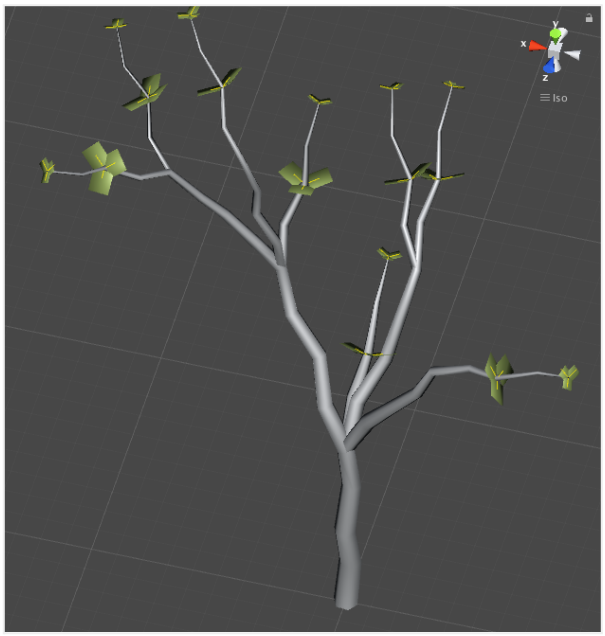
Each sprouts node has two sprouts, with an opposite direction taking the same position on its parent branch.

Whorled

The number of sprouts per sprouts node can be specified using the whorled steps value.



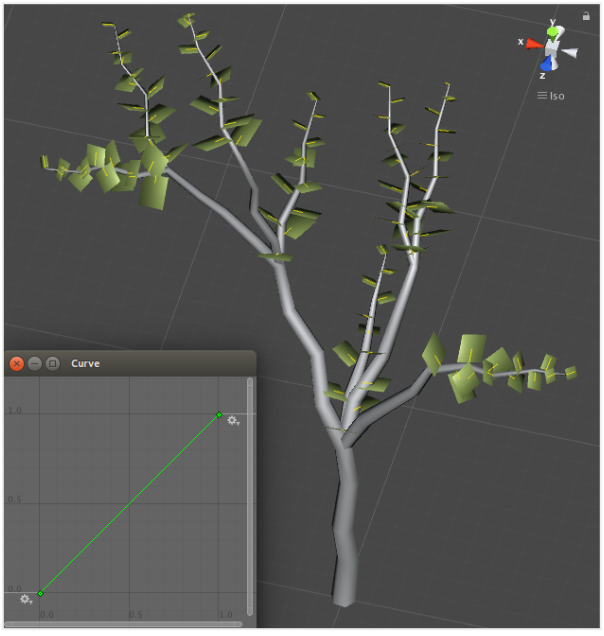
Opposite distribution.



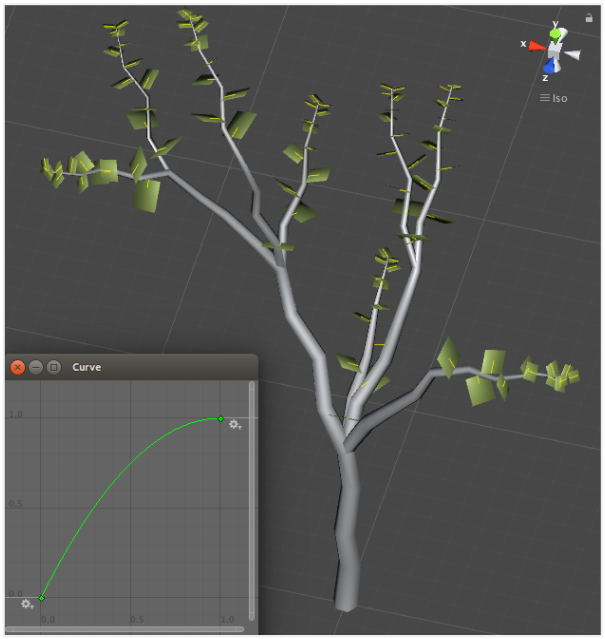
Whorled distribution with 3 steps.

Distribution Curve

This curves controls the spacing of sprouts along the parent branch; on the x axis 0 represents the base of the parent branch and 1 the tip of it.



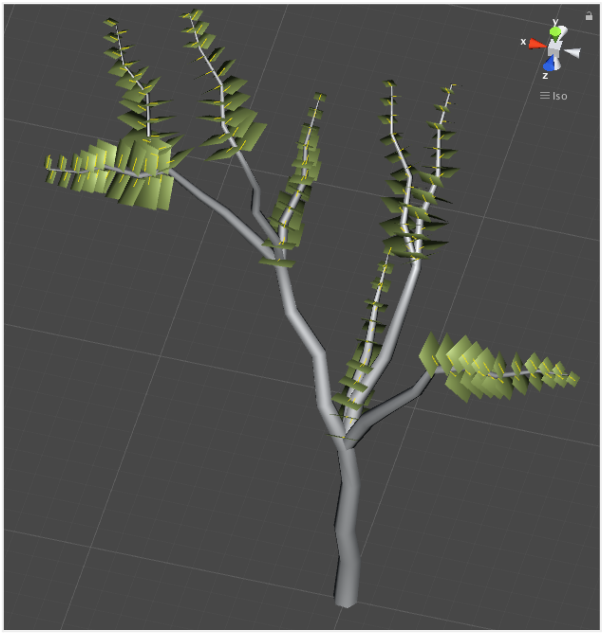
Linear distribution curve.



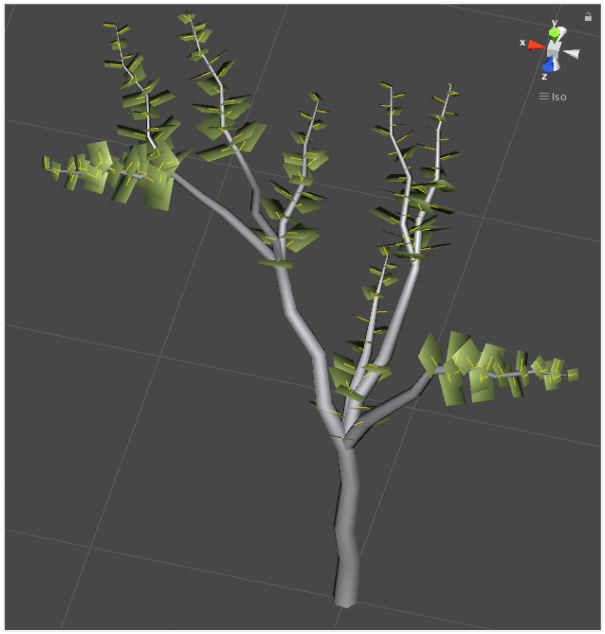
Distribution curve with sprouts toward the tip of the branch.

Twirl

Rotation value for the sprouts taking the parent branch as axis.



Twirl = 0



Twirl = 0.5

Alignment

Normally the sprouts point at a 90 degrees angle to their parent branch direction, alignment properties tune the sprout this direction in relation to the sprout position on its parent branch.

Parallel Align at Top, Parallel Align at Base, Parallel Align Curve

Interpolates the direction of the sprouts with its parent branch direction. When 1 the sprout points exactly to the same direction as its parent branch does (thus the parallel name). Parallel alignment value requires a property that tells how much interpolation is going to be applied when sprouts are positioned at base and at top of their parent branch, positions in between are obtained using the parallel align curve property. Negative values on the properties point to the opposite direction of the parent branch.

Gravity Align at Top, Gravity Align at Base, Gravity Align Curve

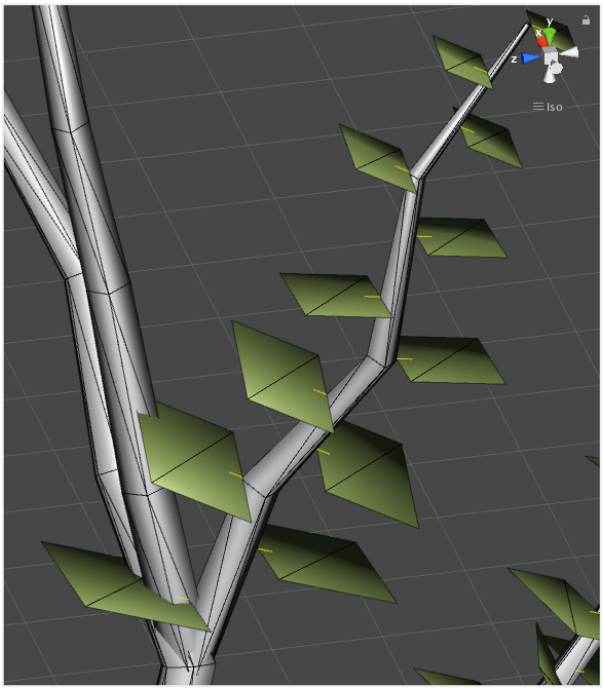
Interpolates the direction of the sprouts with the against gravity vector. When 1 the sprout points exactly against the gravity vector (upward). Gravity alignment value requires a property that tells how much interpolation is going to be applied when sprouts are positioned at base and at top of their parent branch, positions in between are obtained using the gravity align curve property. Negative values on the properties point to the gravity vector direction (downward).

All alignment values to zero and linear alignment curves.

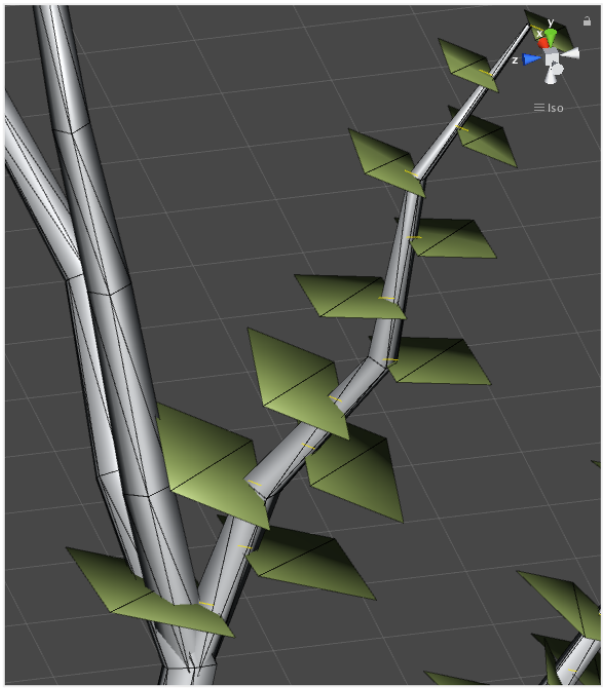
Parallel alignment at top = 1 and gravity alignment at top = 0.5

From Branch Center

When active the sprout mesh originates at the center of the branch mesh and not at its surface.



Sprout mesh has its origin at the branch surface.



Sprout mesh has its origin at the center of the branch.

Distribution Origin

Sets the origin of the sprout lineage in reference to the tree hierarchy.

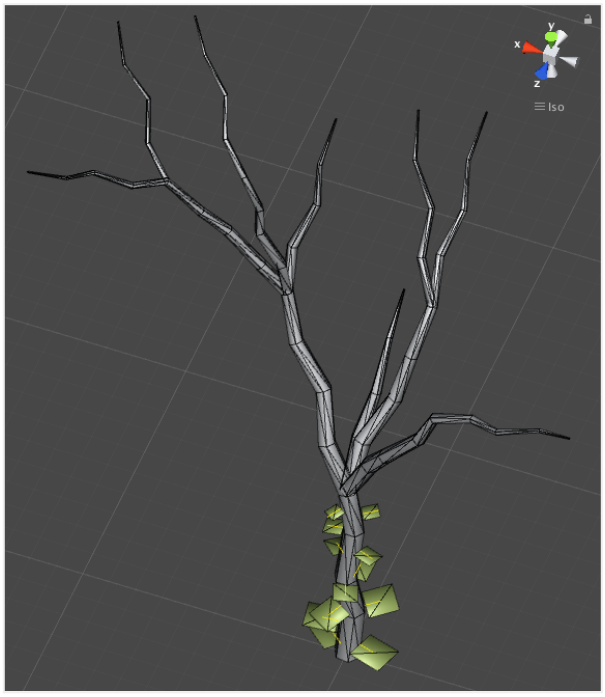
From Tip Branches

The default mode generates the sprout from the terminal branches on the tree.

From Trunk

The sprout lineage starts upwards from the tree trunk base.

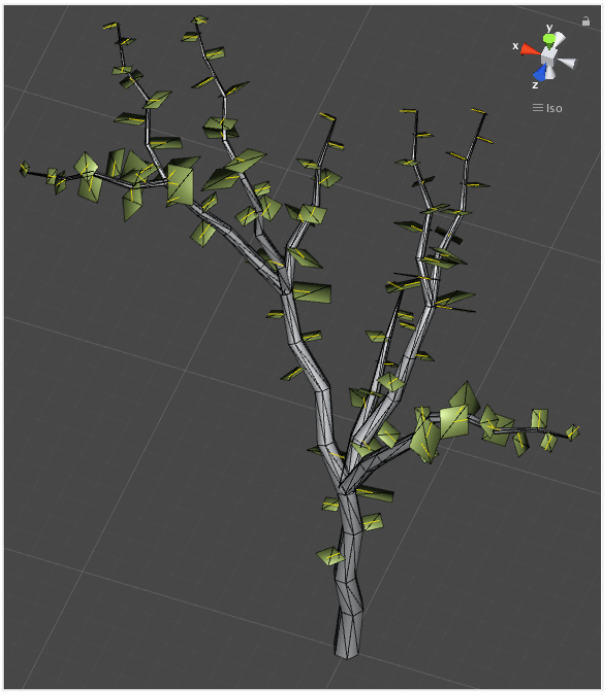
Distribution origin from tip branches.



Distribution origin from trunk.

Spread Enabled

If checked then the sprout lineage goes beyond their origin branch (depending on the distribution origin mode). The spread range value controls on how much of the tree hierarchy the sprouts are going to be placed; 0 for the point of origin and 1 for the whole hierarchy.



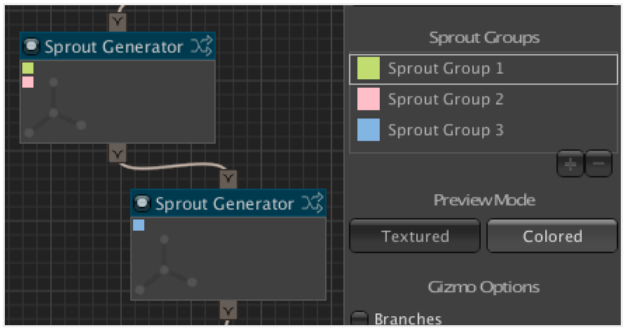
Spread enabled with range = 0.8.

Sprout Seeds

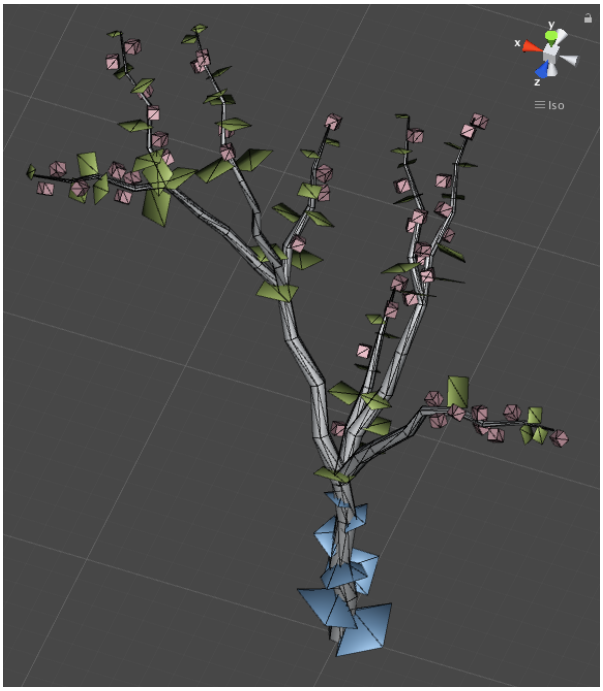
Controls the assignation of sprout groups on the lineage. The sprouts generated are assigned randomly between all the groups contained on the sprouts seeds value.

Sprout lineage assigned to two sprout groups.

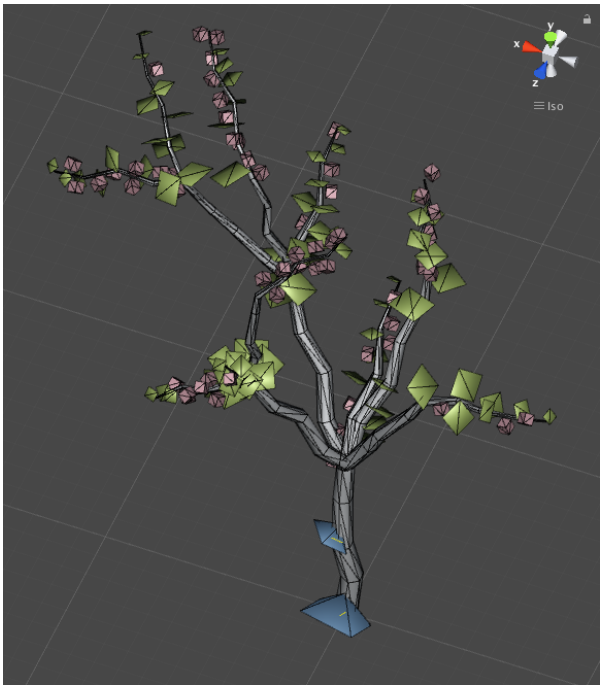
Since every sprout generator node creates a single lineage of sprouts, the pipeline allows the inclusion of multiple of such nodes to create a more modular and richer final tree.



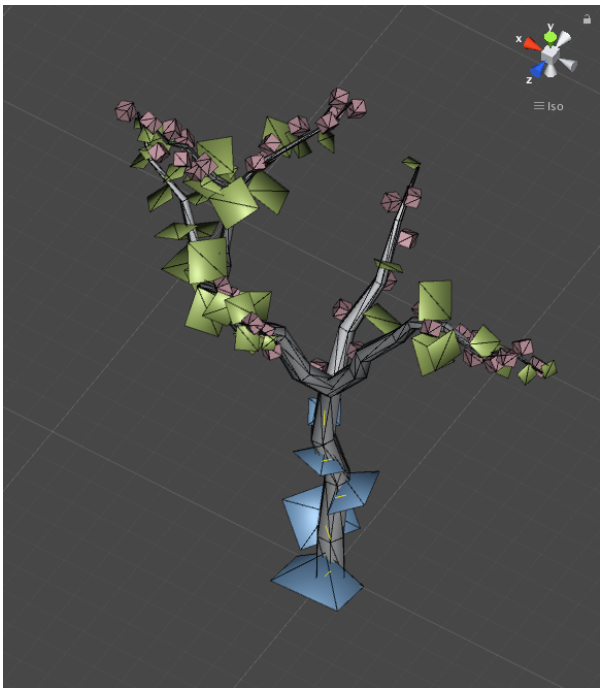
The following series comes from two sprout generator nodes, using two different sprouts lineages assigned to three sprout groups.



Sprout generation variant 1



Sprout generation variant 2



Sprout generation variant 3

Length Transform Node

The Length Transform Element is a structure transforming element on the pipeline; it applies modifications on the length and length scale of the existing branches on a tree. Each branch on the tree is given an absolute position value between 0 and 1 depending on their position relative to their parent and on their overall hierarchical position on the tree. This absolute position value is used to interpolate a length or a length factor to apply on the branch (0 for min and 1 for max value).

The resulting length applied to the branch is the result of the original length value multiplied by a factor. The factor value is the result of the interpolation of the absolute positional value on the min/max factor range.

MinFactor, MaxFactor

Min and max limit values to use on the range used to interpolate the absolute positional value of the branch; 0 takes the min factor value and 1 the max factor.

Curves

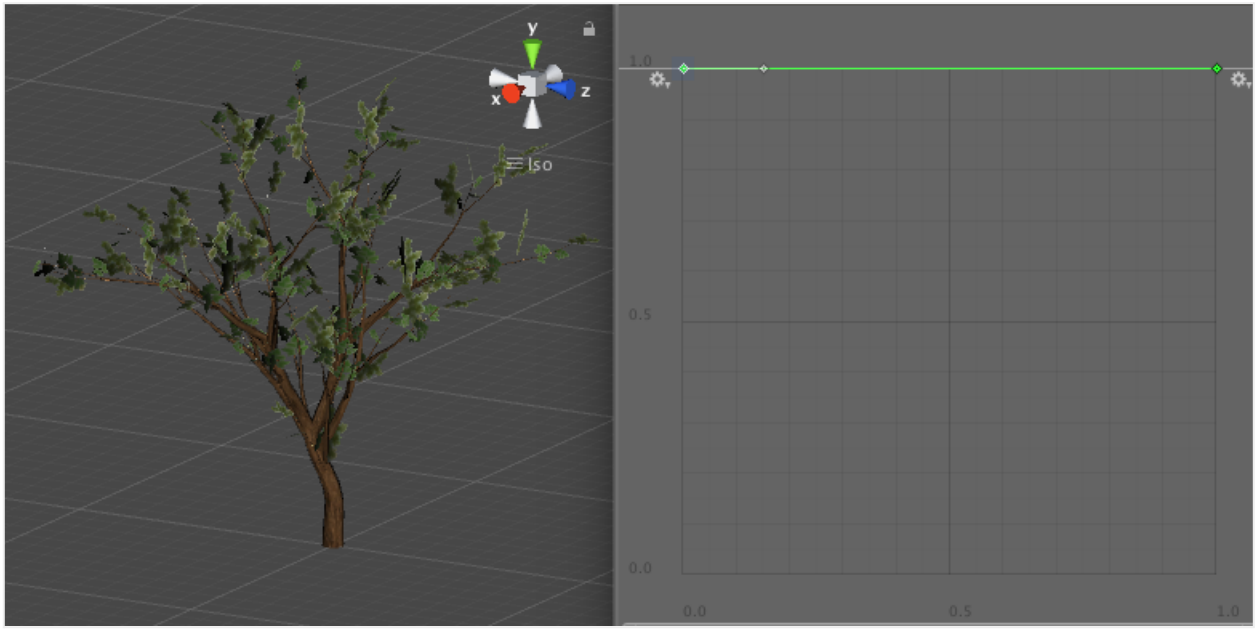
These curves are used to control the distribution of absolute positional value on its position and level components. The absolute positional value of the branch is the product of these components.

Position Curve

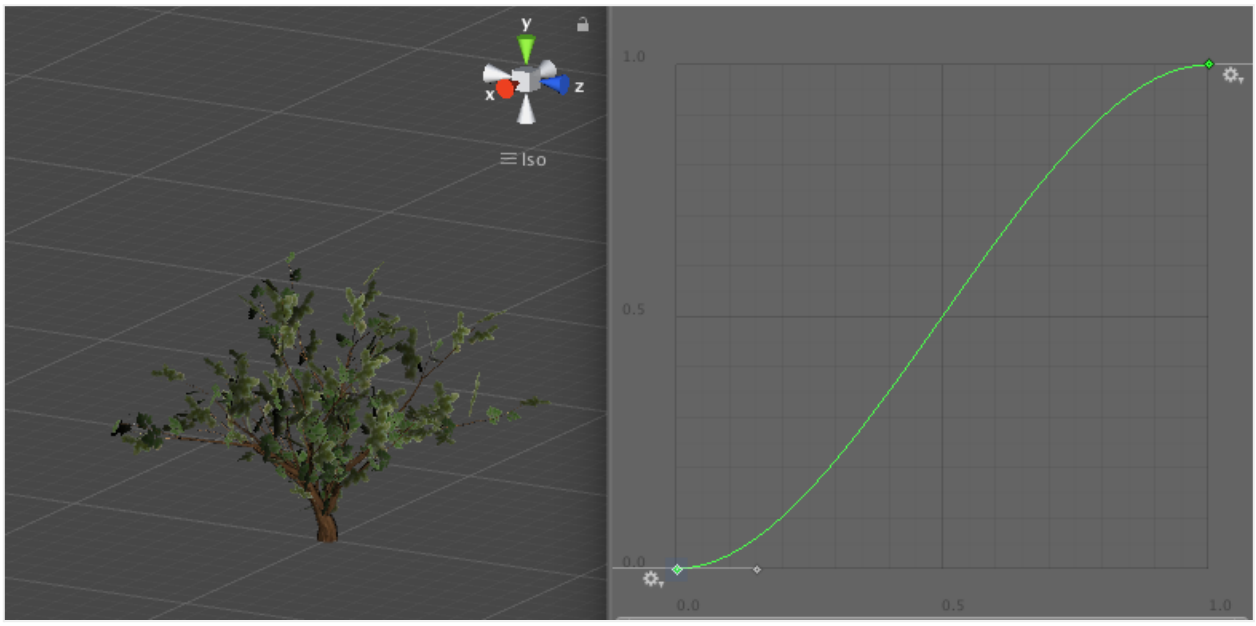
The x axis of the curve indicates the position of the branch along its parent length (0 at the base, 1 at top). The y axis stands for the positional component value, by default 1 for all x values so that the evaluation of the curve gives 1 at all times (thus setting max length/factor on the branch).

Level Curve

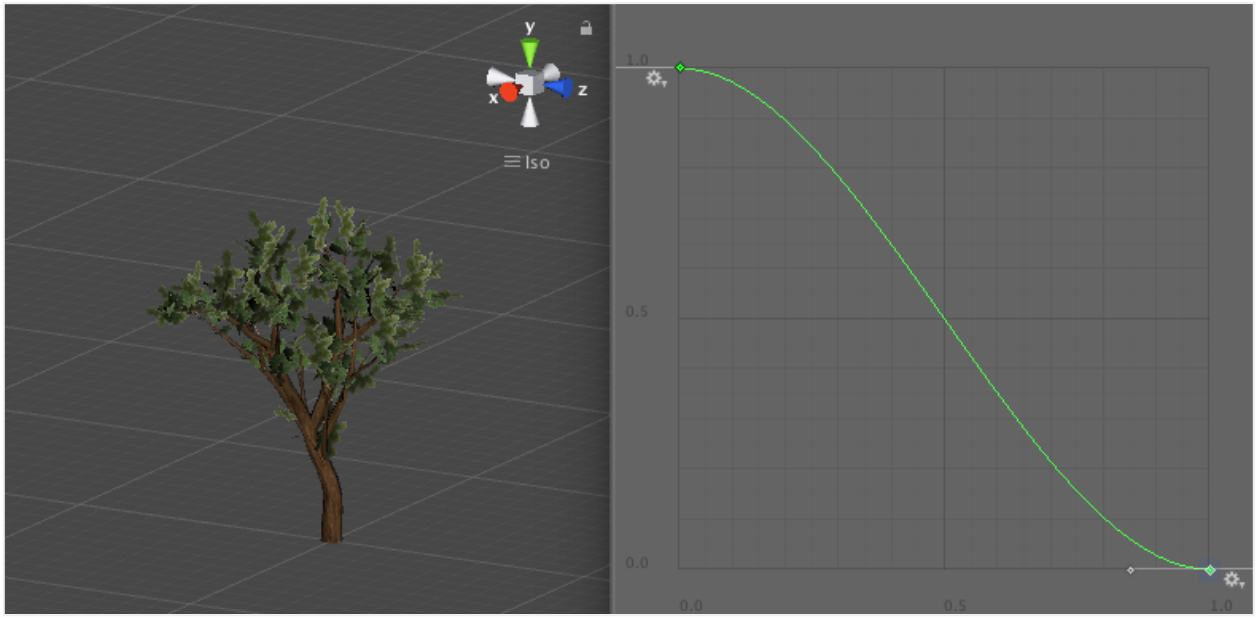
The x axis of the curve indicates the hierarchical position of the branch on the number of levels of the tree (0 for the root of the tree, 1 for tip branches). The y axis stands for the level component value, by default 1 for all x values so that the evaluation of the curve gives 1 at all times (thus setting max length/factor on the branch).



The level curve applies the factor or 4 (max) on all the branches of the tree.



The level curve applies a factor of 1 at the root branch (min factor) and a factor of 4 (max factor) on the tip branches. Branches in-between are given interpolated values.



The level curve applies a factor of 4 at the root branch (max factor) and a factor of 1 (min factor) on the tip branches. Branches in-between are given interpolated values.

Branch Bender Transform Node

The Branch Bender Element applies bending to the branches bezier curves.

Apply Joint Smoothing

If checked a curved smooth transition is applied between parent and follow-up branches.

Smoothing Strength

If 0 the transition between parent and child follow-up branch is kept at a sharp angle, going towards value 1 a curve is applied to the transition.

Apply Directional Bending

If checked a force is applied to bend the branches. Positive values go against gravity and negative values go in favor of gravity.

Force at Tips

The force applied to the tip of the branches of the tree. Positive values go against gravity and negative values go in favor of gravity.

Force at Trunk

The force applied to the branches near the base of the tree. Positive values go against gravity and negative values go in favor of gravity.

Hierarchy Distribution

Curve to modify the distribution of the applied force across the tree hierarchy of branches. The left side of the curve is for branches near the base, the right side is for terminal branches.

Apply Branch Noise

Adds Perlin noise to the branches of the tree, this gives them tortuosity to look more real (by applying a directional offset based on the noise).

Noise at Base

Magnitude for the directional offset to be applied at the base of the branches hierarchy (from the main trunk), the higher the more offset is applied at each branch bending point. This value is spread to the top of the branches (the tip of the branches).

Noise at Top

As described for the property Noise at Base, this is the value for the directional offset to be applied at the top of the branches hierarchy (from the tip of the terminal branches), the higher the more offset is applied at each branch bending point. This value is spread to the base of the branches (the tree trunk).

Noise Scale at Base

Perlin noise scale value at the base of the tree hierarchy (from the tree trunk spread to the terminal branches). The higher the value more bending points to apply a directional offset.

Noise Scale Top

Perlin noise scale value at the base of the tree hierarchy (from the tree trunk spread to the terminal branches). The higher the value more bending points to apply a directional offset.

Tree with no applied forces.

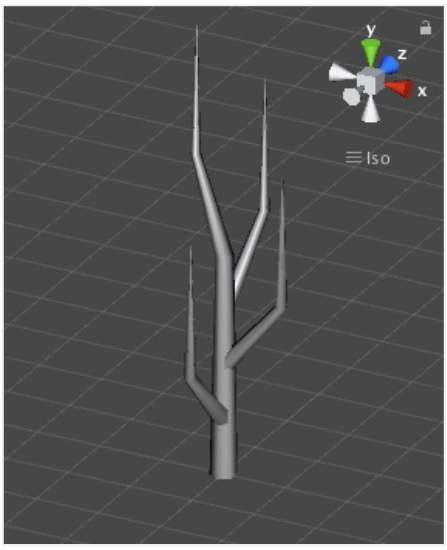
Tree with a factor force at tip = 0.5

Girth Transform Node

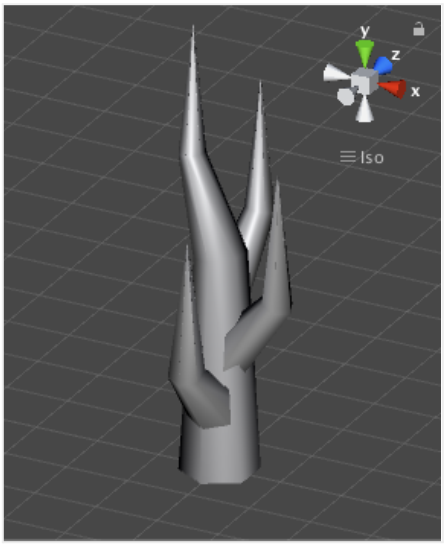
This element is used to assign a girth value to the tree branches. The girth value is used to create the mesh that represents the bark of the tree and is also the reference value to position sprouts on the surface of this barked is specified so.

Girth at Base

Value of girth at the trunk level of the tree (branch hierarchy 0).



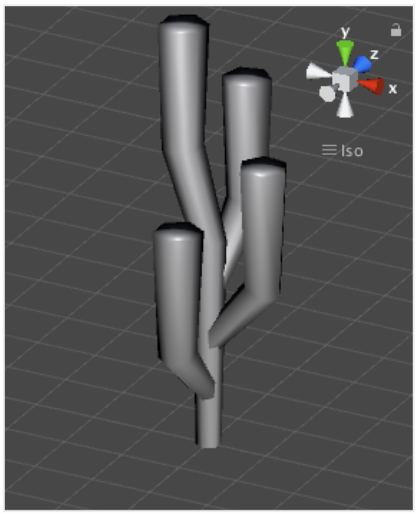
Tree with girth at base = 1



Tree with girth at base = 3

Girth at Top

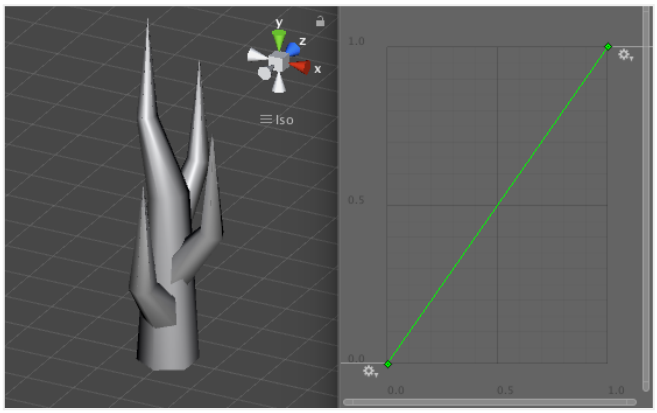
Value of girth at the top branches of the tree (last branch hierarchy).



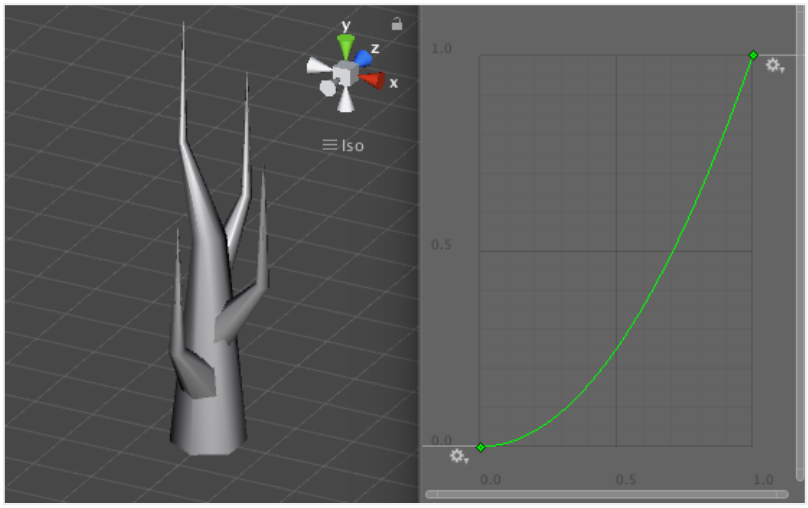
Tree with girth at top = 2 and girth at base = 1

Curve

Transition curve to control how the values from girth at base and girth at top is applied along the tree hierarchy. The left of the x axis represents the base of the tree, while the right side stands for the top branches. On the y axis the bottom is the girth at base value and the top the girth at top value.



Linear curve to transition from girth at base = 3 to girth at top = 0.05



Exponential curve to transition from girth at base = 3 to girth at top = 0.05

Hierarchy Scaling

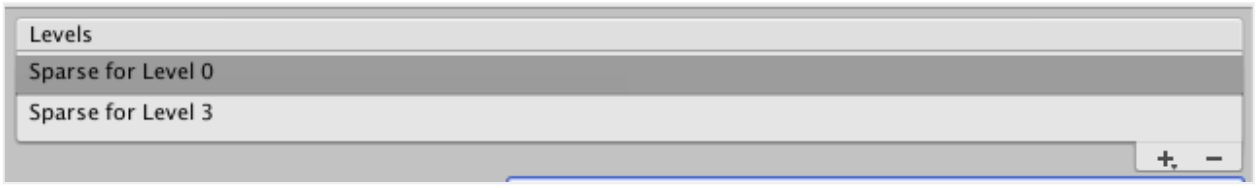
If enabled applies an scaling value to the girth of terminal branches children of the tree trunk. This options gives a more natural look to trees making these branches look as newer branches.

Sparse Transform Node

The sparse is used to modify already defined branch structures on a tree. The directives used are bound and are meant to be applied to a specific hierarchy level of the tree; for example, level 0 means the children branches of the root branch (es) are going to receive the modifications.

Levels List

Holds the directives listed per hierarchy level to be applied to. The list allows up to 5 hierarchy levels to be defined; if the hierarchy level is not present on the tree structure then that directive is simply not processed.



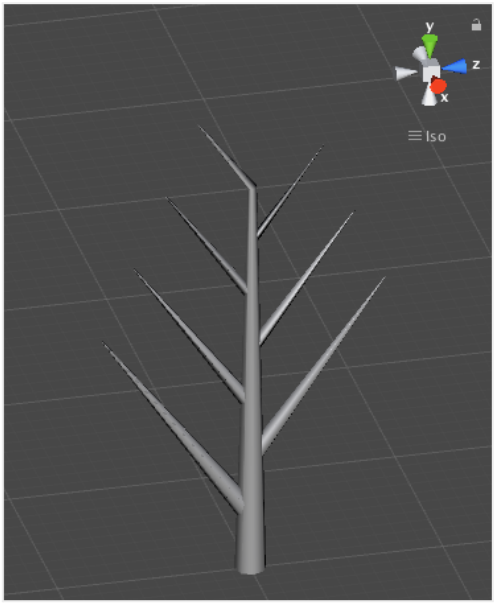
List of sparse properties per hierarchy level.

Reorder Mode

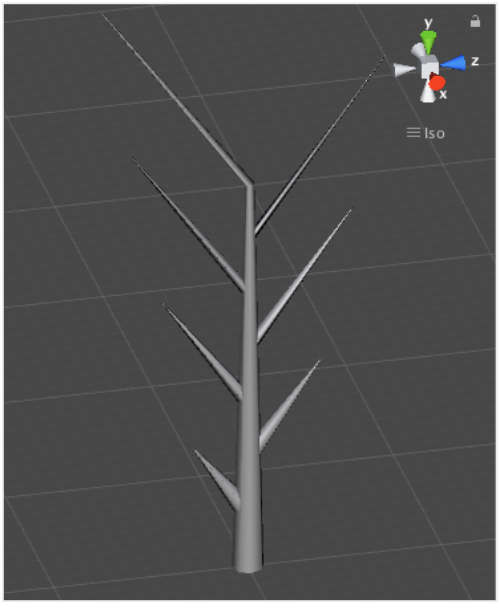
Switch the positions of the existing children branches on their parent, where the parent should match the hierarchy level specified on the list.

Reverse

Reverses the position of the children branches. Does not modify length or direction of the affected branches, but girth might get a new value as this property is position dependent.



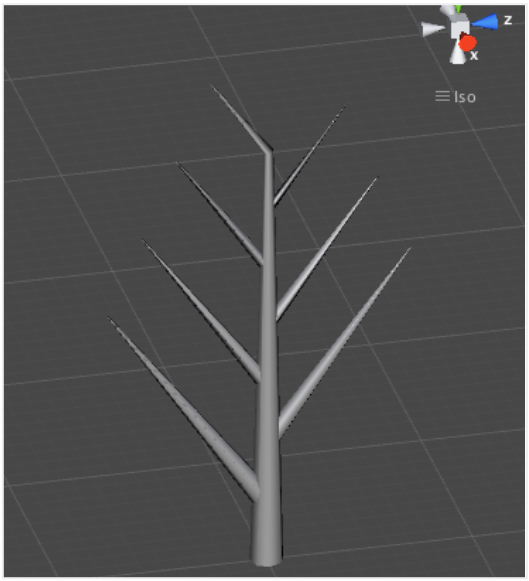
Base structure tree.



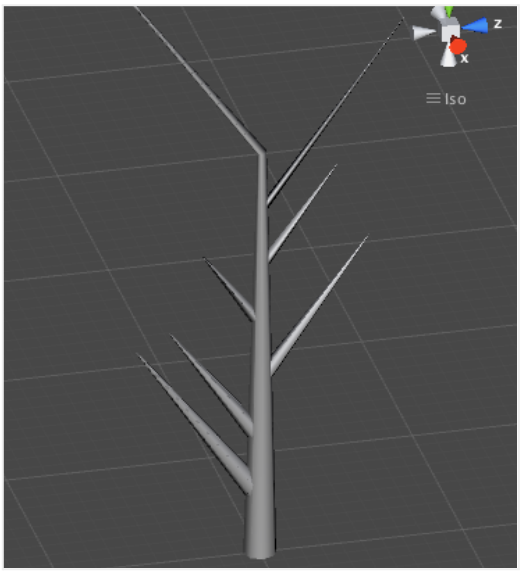
Tree with reversed child branches for hierarchy level 0.

Random

Randomizes the position of the children branches. This randomization could be fixed using a custom seed on this element.



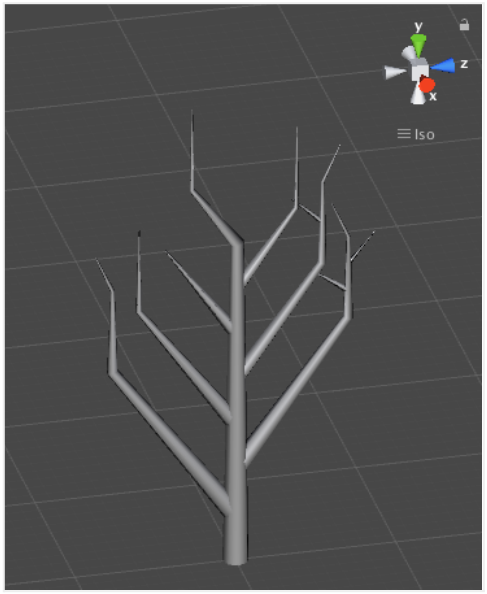
Base structure tree.



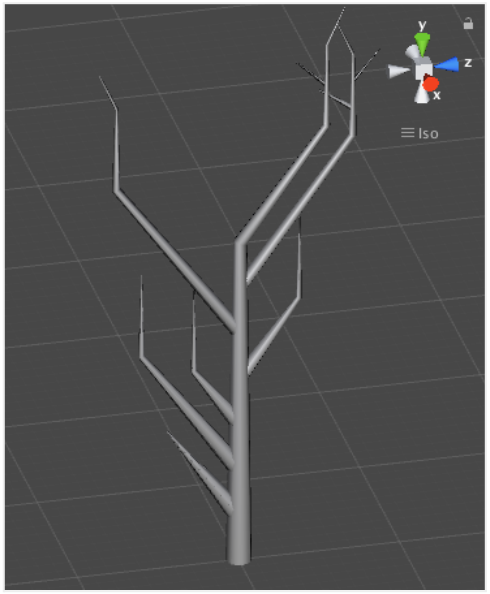
Tree with randomized child branches for hierarchy level 0.

Heavier at Top

Reorders the children branches of a hierarchy level putting the ones with more offspring levels at the top of their parent branch. The “heavier” term means only the levels beyond the affected branch.



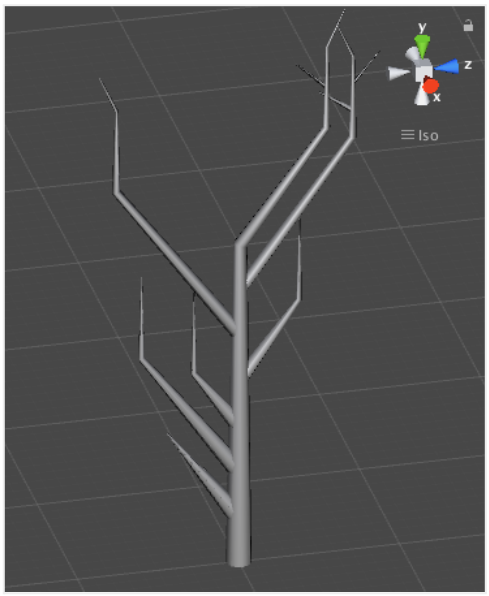
Base structure tree with branches up to 3 offspring levels.



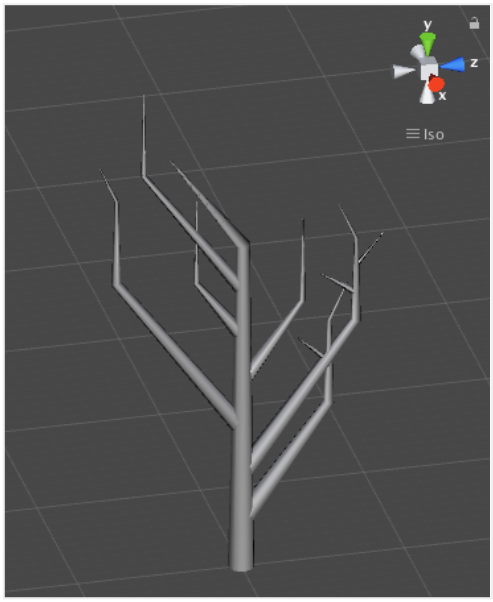
Tree with heavier branches at top for hierarchy level 0 children branches.

Heavier at Bottom

Reorders the children branches of a hierarchy level putting the ones with more offspring levels at the base of their parent branch.



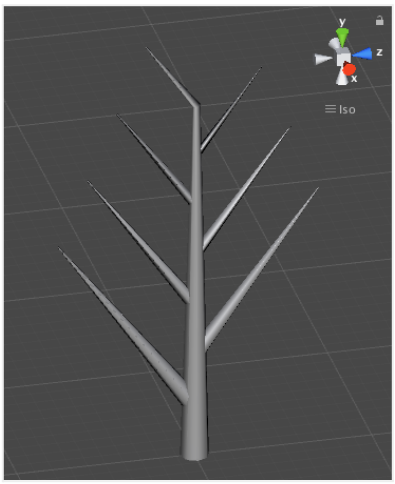
Tree with heavier branches at top for hierarchy level 0 children branches.



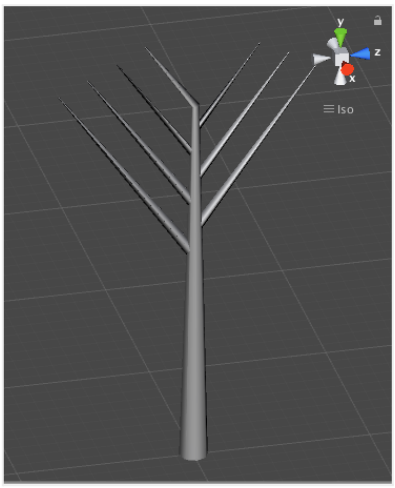
Tree with heavier branches at base for hierarchy level 0 children branches.

Length Sparsing Mode

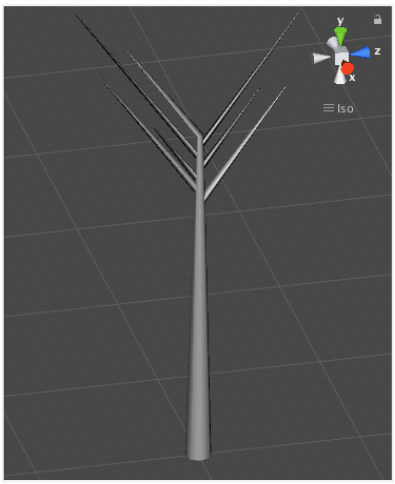
Spreads all the children branches of a hierarchy level branch along their parent using a reference value. When length = 1 it uses the whole parent branch, 0 positions all the children branches at top of their parent branch.



Base structure tree.



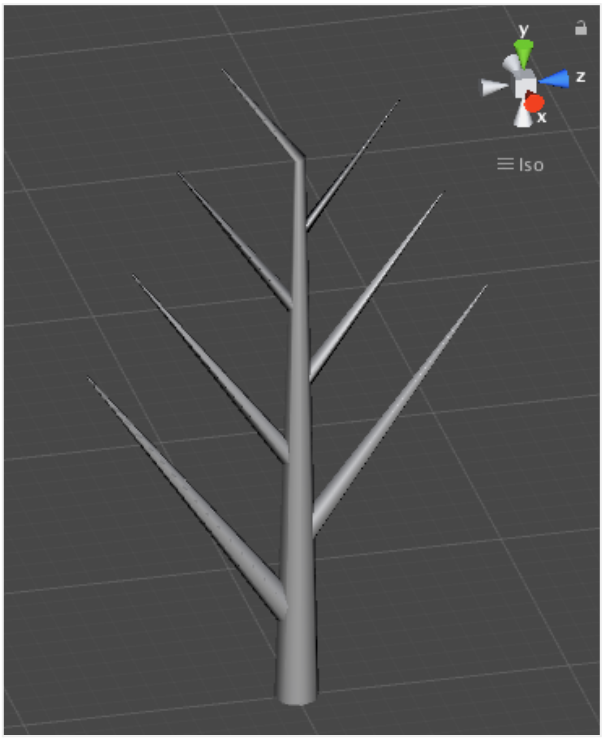
Children branches for level 0 repositioned occupying half its parent branch (value = 0.5).



Tree with children branches of level 0 randomized and with sparse length = 0.25.

Twirl Sparsing Mode

Adds a twirl to the children branches of a level branch around it.



Base structure tree.

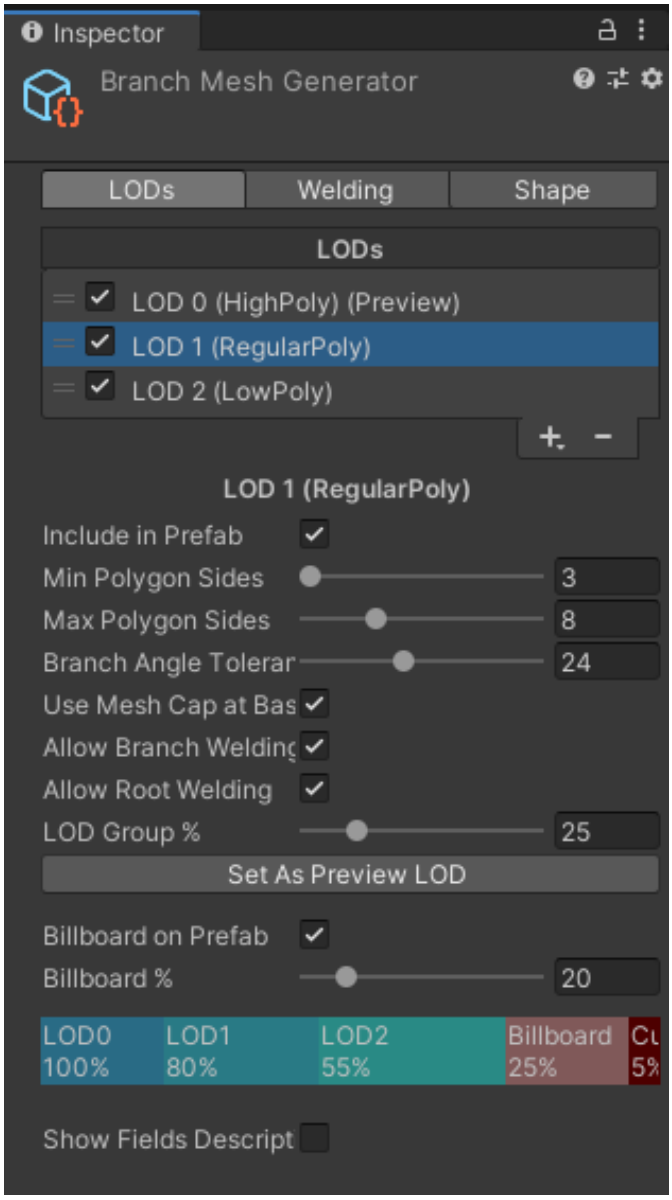
Children branches of level 0 with twirl sparse = 1.5

Branch Mesh Generator Node

This element creates the mesh for all the branch structure of the tree.

LODs

This panel provides options to control the resolution of the mesh when previewing the tree in the editor, when creating trees at runtime and on the LOD Group on the final Prefab.



The LODs panel displays a list of Level of Details definitions. The + button provides predefined LOD definitions, from Ultra High Poly, passing to Regular Poly and up to Ultra Low Poly. The values of each of these definitions are customizable.

Only the LOD definitions checked on the list will be included in the final Prefab in the order of the list (the list is reorderable and the bar below will display the final LOD Group settings). The LOD definition used in the Editor for previewing the trees and to use on the Runtime is marked with the (Preview) postfix.

Selecting a LOD definition from the list displays the options to customize its values:

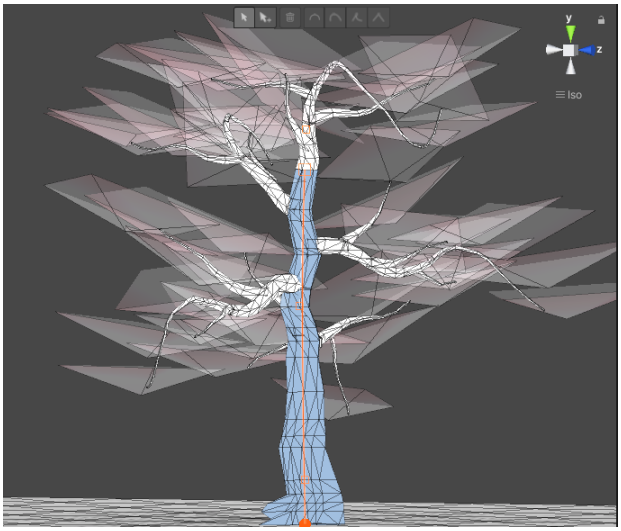
- **Include in Prefab:** check this to include the LOD definition on the final Prefab LOD Group.
- **Min/Max Polygon Sides:** minimum and maximum number of sides on each branch or root segment.
- **Branch Angle Tolerance:** the minimum the angle the more resolution on the curves of a branch or root.
- **Use Mesh Cap at Base:** adds additional geometry to the base of each branch and root, creating a cap of polygons right at the beginning of the branch/root.
- **Allow Branch Welding:** overrides the root welding settings, disabling this feature for this LOD definition.
- **Allow Root Welding:** overrides the root welding settings, disabling this feature for this LOD definition.
- **LOD Group %:** the percentage for this LOD definition to use on the final Prefab LOD Group.

- **Set as Preview LOD:** sets the selected LOD definition on the list ast the default LOD to be used on the Editor when creating tree previews and on the Runtime. It is not necessary for the LOD to be part of the LOD group (include in Prefab) to be the preview LOD.

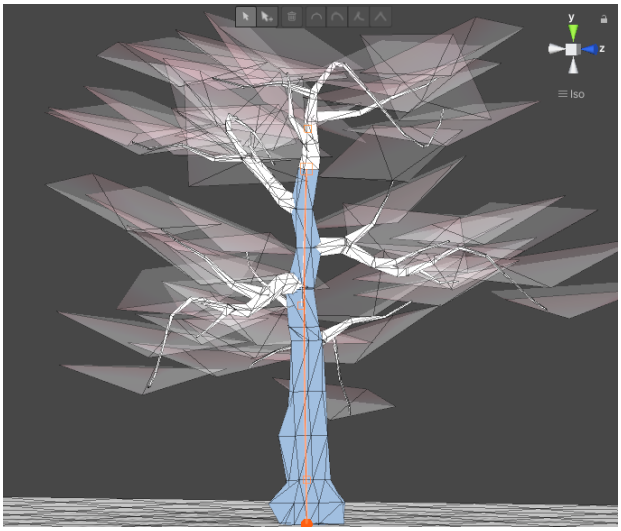
  The LODs panel also displays the option to use billboards on the final Prefab:

- **Billboard on Prefab:** check this to include a billboard asset on the final Prefab LOD Group.
- **Billboard %:** the percentage for the billboard to use on the final Prefab LOD Group.

The LOD bar displays how the LOD Group on the final Prefab should look like taking all the LOD definitions on the list and the billboard. It should display a warning if the sum of these LODs is over 100%.



Tree mesh with High Poly LOD.



Tree mesh with Low Poly LOD.

Branch Welding

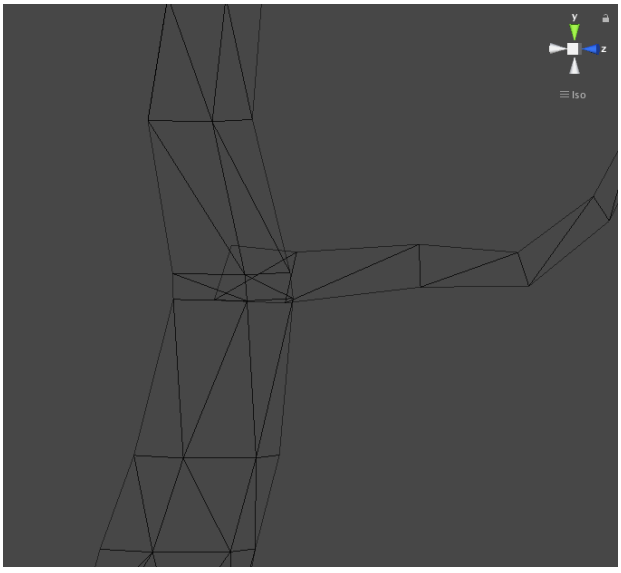
This feature uses a mesh collider when building children branches to set the base vertices ruth on the surface of their parent branch mesh, this creates a smooth transition between parent and children branches.

Branch welding off. The child branch mesh originates from the center of the parent branch mesh.

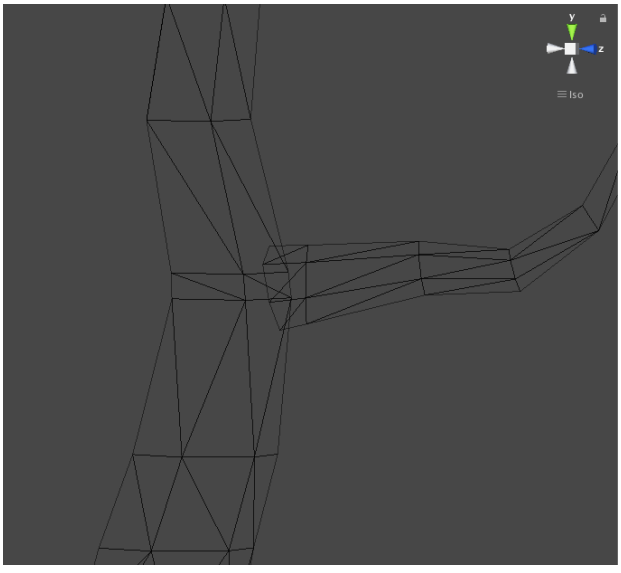
Branch welding on. The child branch mesh originates at the surface of the parent branch mesh.

Use Branch Welding

Flag to activate branch welding processing on the selected hierarchy range.



Branch welding off. The child branch mesh originates from the center of the parent branch mesh.



Branch welding on. The child branch mesh originates at the surface of the parent branch mesh.

Hierarchy Range

How far along the tree hierarchy children branches should have welding. The range goes from 0 at the base of the trunk to 1 at the tip of the farest branch. We recommend applying welding only to clearly visible branches (near the trunk); since terminal branches usually are obscured by foliage and require less mesh resolution, it is better to spare welding at these levels of the hierarchy.

Branch Welding Hierarchy Curve

Curve to apply lerp to diminish the amount of welding from the base of the hierarchy to the selected hierarchy range.

Branch Welding Curve

Controls the shape to add to the welding, from the base of the parent mesh to the welding distance segment.

Welding Distance

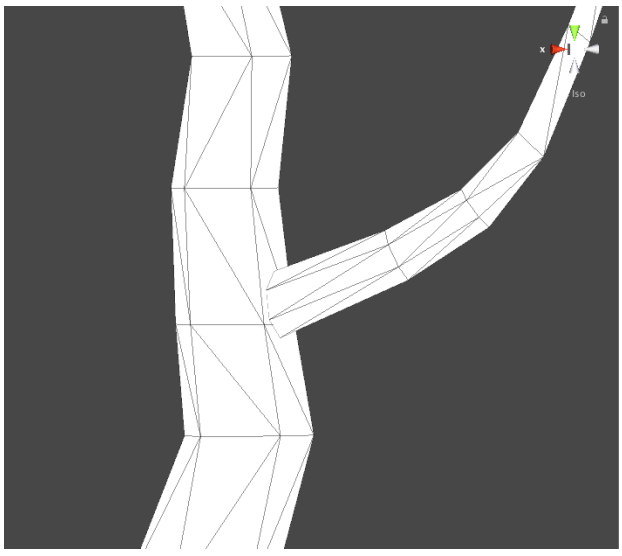
How long the welding transition should be applied along the length of the child branch. The value is a factor to multiply by the girth at the parent girth to get the welding length.

Additional Segments

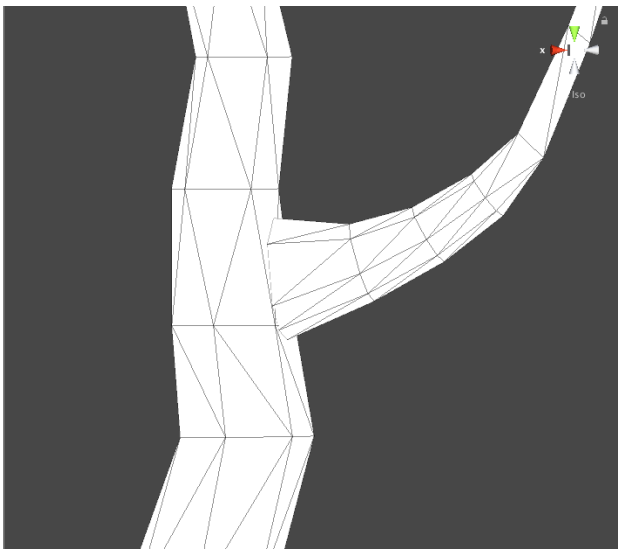
Adds additional segments to the welding length in order to increase its shape resolution.

Welding Upper Spread

How much the welding spreads towards the parent growth direction. The value is a factor that multiplies the girth of the parent at the child branch position to calculate the length of the spread.



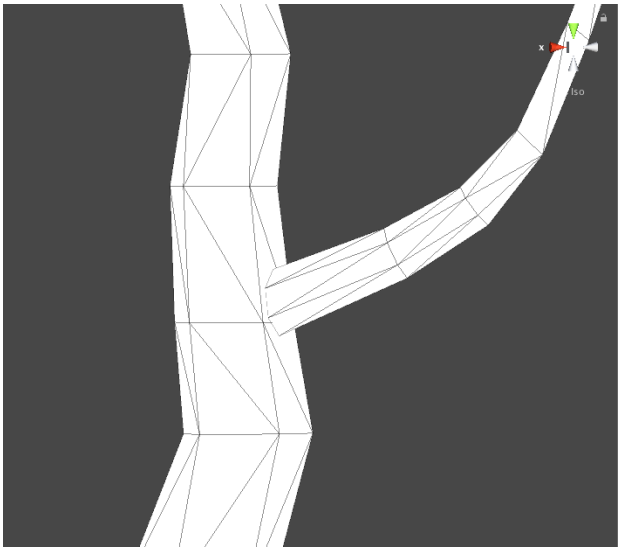
Welding with Upper Spread of 0.



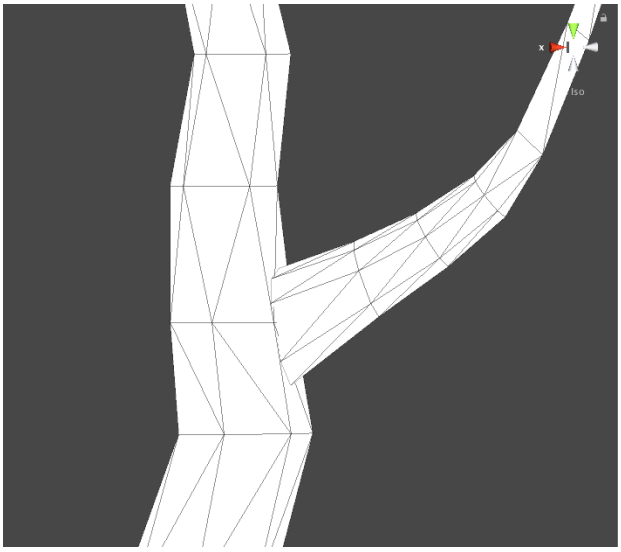
Welding with Upper Spread of 0.8.

Welding Lower Spread

How much the welding spreads against the parent growth direction. The value is a factor that multiplies the girth of the parent at the child branch position to calculate the length of the spread.



Welding with Lower Spread of 0.



Welding with Lower Spread of 0.8.

Trunk Mesh Generator Node

This element generates a stylized mesh on the main branch of a tree.

Spread

Length the mesh will take on the main branch of a tree, this value is taken randomly from the specified range. A value of 0 means no trunk mesh is applied, a value of 1 includes the whole length of the main branch.

Points

Number of points extruding out of the tree trunk mesh, this value is randomly generated from the range specified on this node inspector.

Angle Variance

Value for the angular distance between the extruded point in the tree trunk.

Twirl

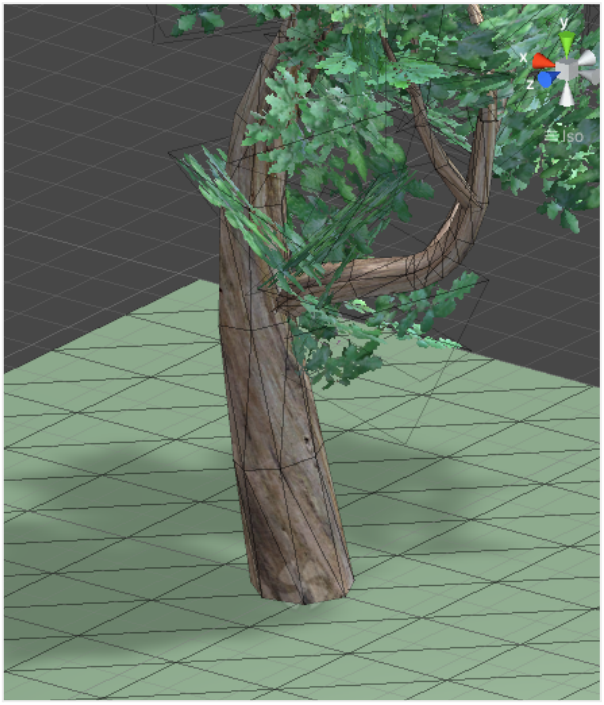
Twirl force applied to the mesh around the trunk axis, positive values result in a clockwise twirl and negative values in a counter-clockwise one.

Scale at Base

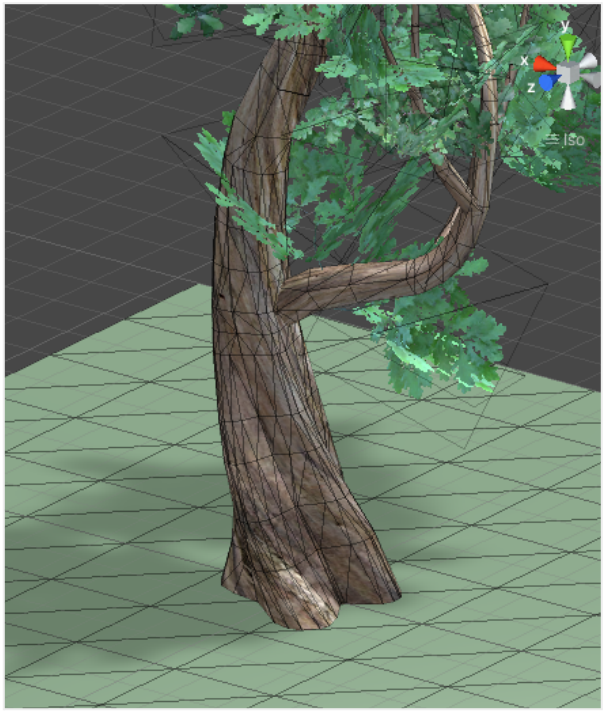
Scale on how much the points extrude from the main tree trunk mesh.

Scale Curve

Curve to control the scale value from the base of the trunk mesh to its top limit.



Tree with no trunk meshing.

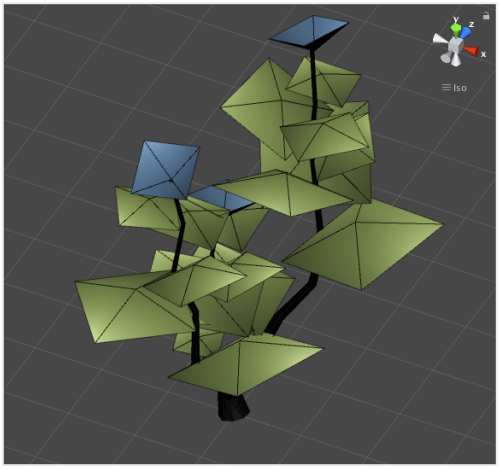


Tree with trunk meshing using twirl.

Sprout Mesh Generator Node

This element contains the instructions on how to mesh the various sprout groups on the tree structure.

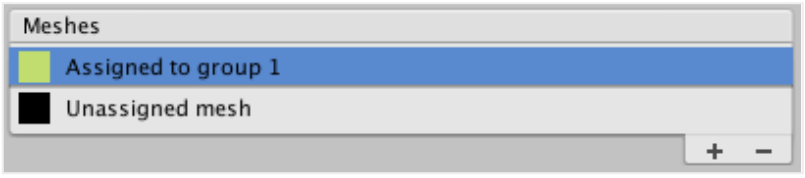
Tree with sprout meshes showing lighting and textures applied.



Underlying mesh structure of the tree with two sprout groups (green and blue).

Meshes

List for all the mesh directives on sprouts. These items must be assigned to a sprout group in order to produce meshes to the sprouts belonging to the group, the color of this group is shown on the left side of each item. Items could also be left unassigned. All properties related to a mesh item can be modified by selecting the element on the list.



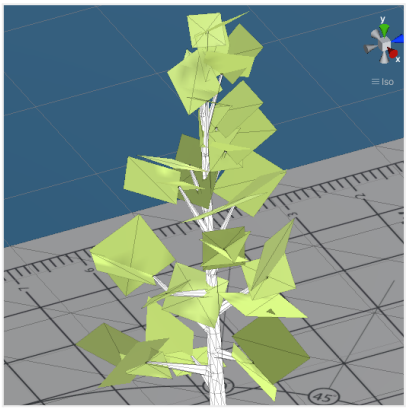
Sprout Group

List to change the assigned sprout group on a mesh item. Mesh items are required to be assigned to a sprout group in order to produce a mesh for sprouts belonging to the group.

Mode

Mesh modes provide different methods to create individual meshes for the sprouts. The mode available are:

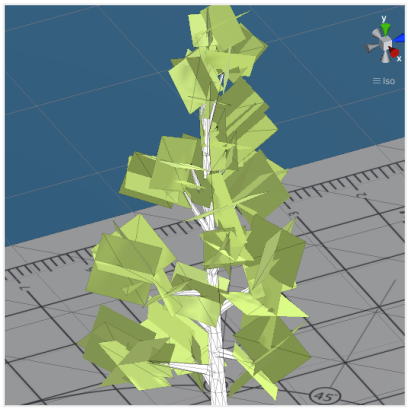
Plane



Plane mode.

The simplest mesh, it consists of a double-side plane for each sprout.

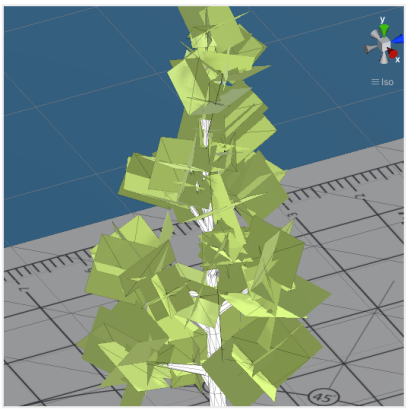
Cross



Cross mode.

Uses two double-side planes perpendicular to each other and intersected on their center.

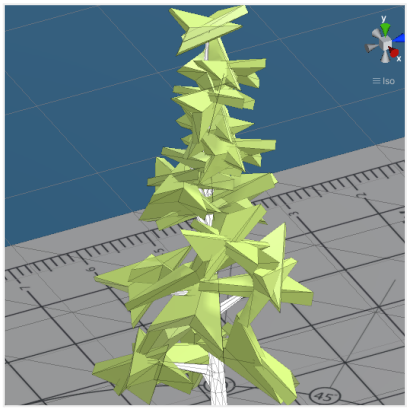
Tricross



Tricross mode.

Uses three double-side planes intersected on their center to create a mesh for each sprout.

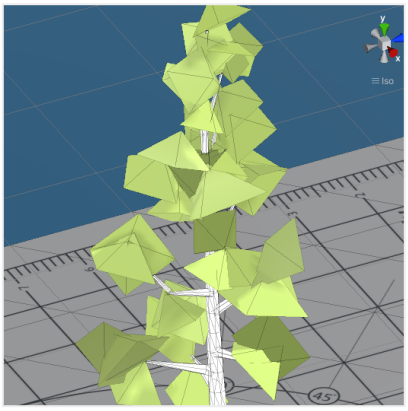
Mesh



Uses a GameObject containing a Mesh object as reference to create each sprout mesh.

Mesh mode.

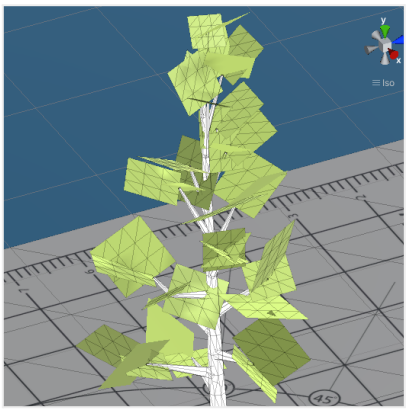
X Plane



Uses a plane that might turn into a pyramid shaped mesh assigning a depth value to its center.

X plane mode.

Grid Plane



Similar to the Plane mode but with resolution options to subdivide the mesh (better for bending sprouts).

X plane mode.

Common properties

The following properties are used by most of the mesh modes and are used basically to modify the dimensions of the mesh depending on the sprout position or their assigned texture.

Width and Height

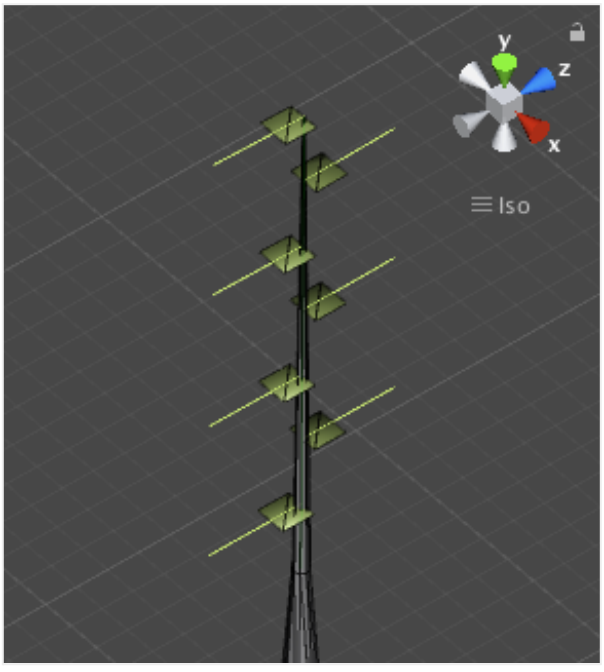
Dimensional values for the unscaled 2D plane used to build each sprout mesh. These properties are used on plane based modes (plane, cross, tricross, billboard and x plane).

Override with Texture Height

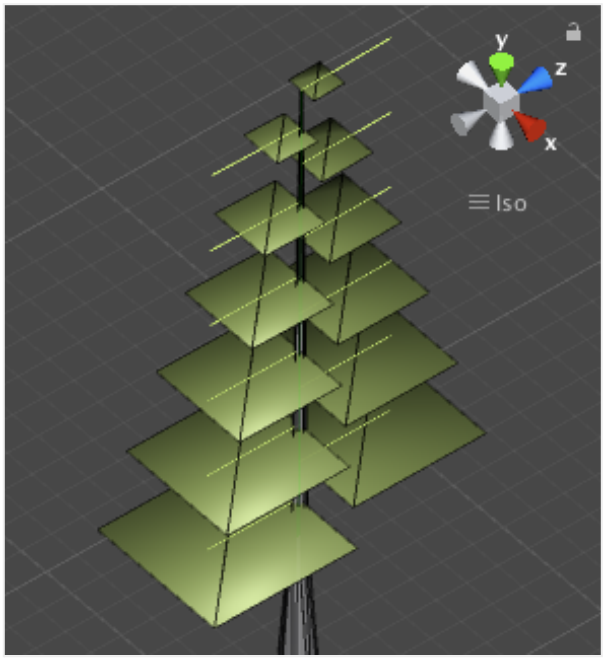
When a texture is assigned to a sprout group it might have a different width/height ratio than the one used at its mesh item. When the “override with texture height” is set to active then only the width dimensional value is taken to create the unscaled 2D plane on the meshes and the height dimensional value is calculated based on the assigned texture width/height ratio.

Scale at Base, Scale at Top and Scale Curve

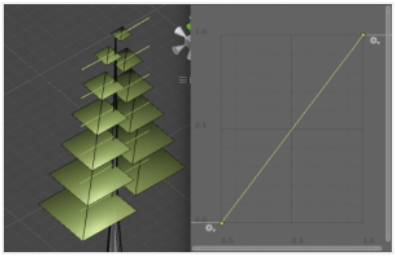
The scale value is used as a dimensional multiplier for all meshes. The scale value can be assigned for sprouts at the base or top of their parent branch, with a curve to control the transition between these two values.



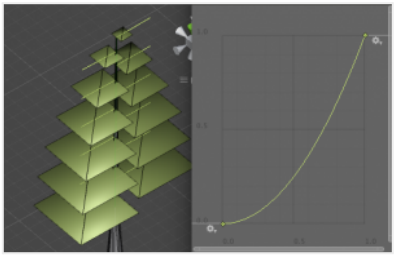
Plane mesh with width = 1, height = 1, scale at top = 1 and scale at base = 1.



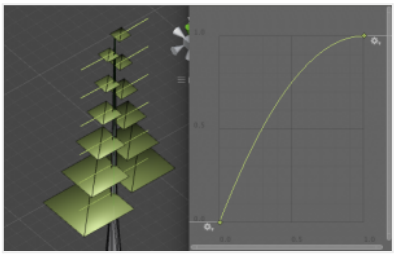
Plane mesh with width = 1, height = 1, scale at top = 1 and scale at base = 4.



Scale at base = 4, scale at top = 1. Transition with a linear curve.



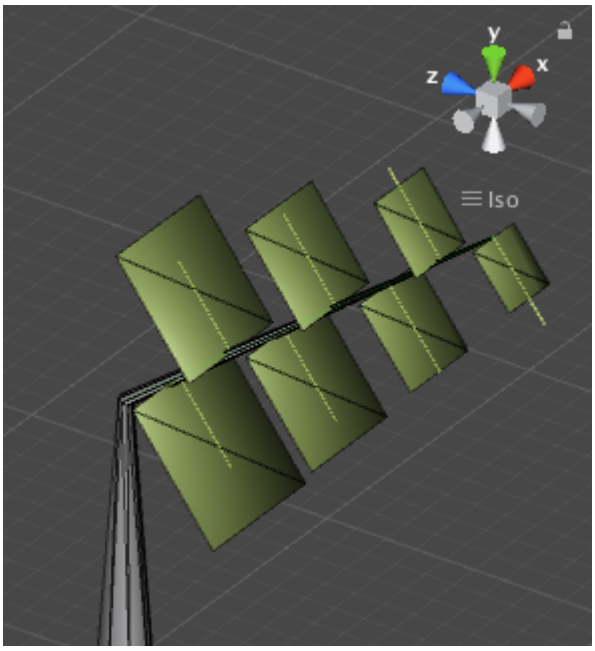
Scale at base = 4, scale at top = 1. Transition with a ease in curve.



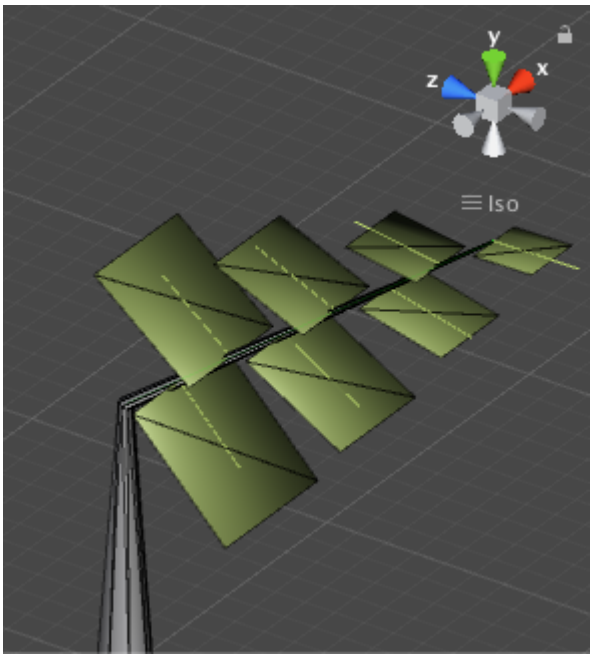
Scale at base = 4, scale at top = 1. Transition with a ease out curve.

Horizontal Align at Top, Horizontal Align at Top and Horizontal Align Curve

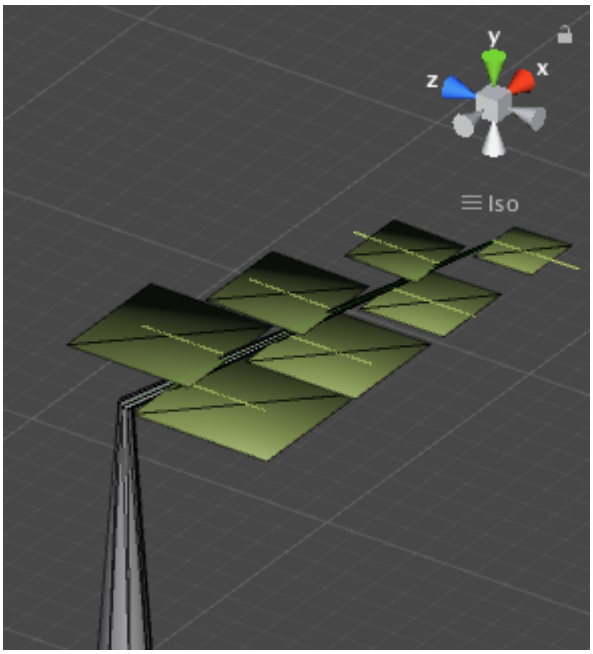
Usually sprout meshes are aligned to their parent branch direction as its upward vector. The horizontal align value is used to extrapolate the upward direction as the against gravity vector (for plane meshed it keeps them facing upwards). Horizontal alignment values could be assigned for sprouts at base or top of their parent branch and a curve is used to ease these values.



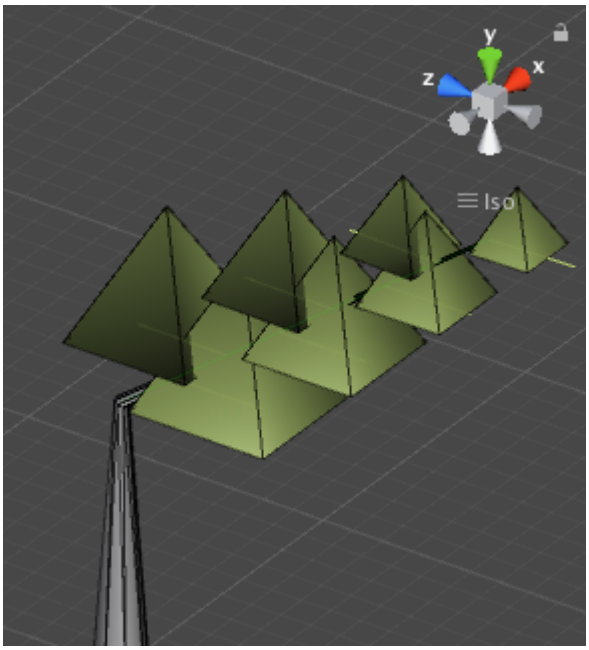
Horizontal align = 0, the sprout meshes point upwards to their parent branch direction.



Horizontal align at base = 0 and at top = 1, the sprout meshes at top point more towards the against gravity direction.



Horizontal align at base = 1 and at top = 1, the sprout meshes along the branch point more towards the against gravity direction.



X plane mode, horizontal align at base = 1 and at top = 1, the sprout meshes along the branch point more towards the against gravity direction.

Mesh Mode

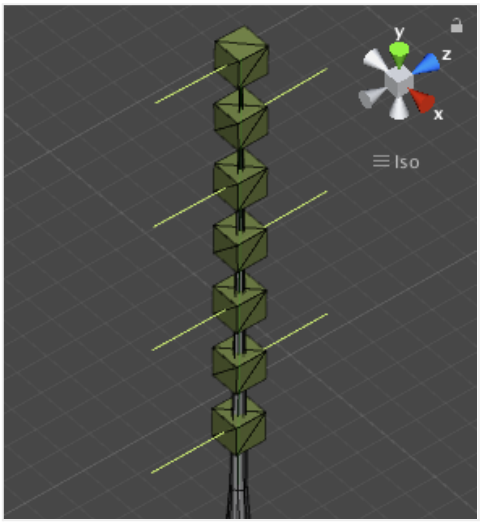
Takes a custom mesh as reference for building all the sprout meshes assigned to this group. The UV mapping from the reference mesh is taken as it is and copied to the new meshes. If the original material of the custom mesh has to be kept then it should be assigned so on the Sprout Mapper node on the pipeline.

Mesh

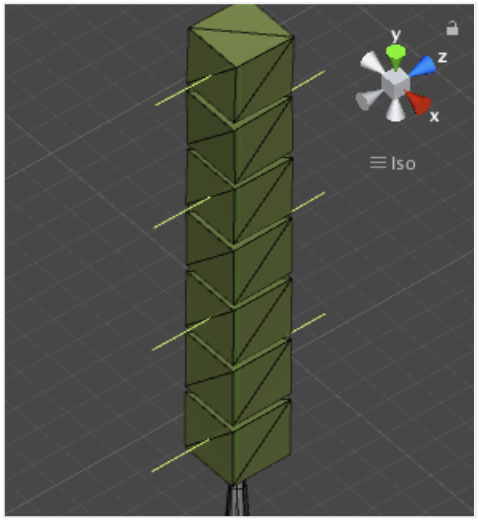
Game object containing the mesh to be used as reference to build the sprout meshes.

Mesh Scale

Scale value for the reference mesh, note that this value is going to add up to the “scale” value shared by all other meshing modes.



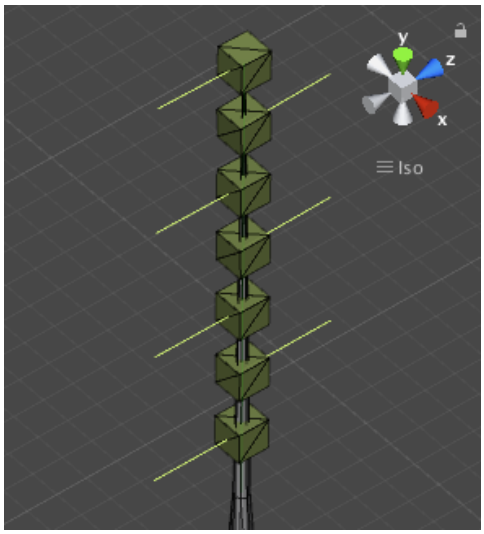
Custom mesh mode with mesh scale = 1.



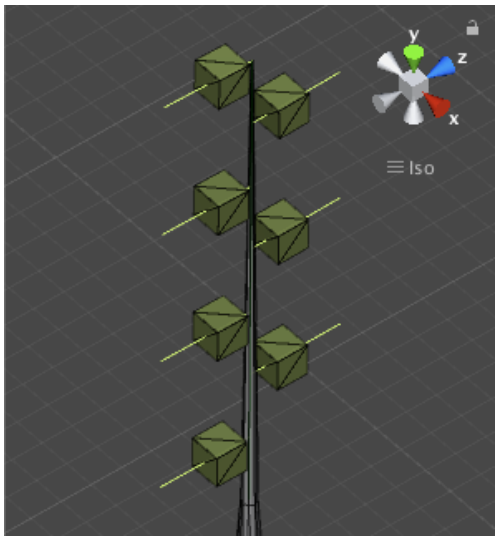
Custom mesh mode with mesh scale = 2.

Mesh Offset

A Vector3 value used to offset the center the custom mesh. Normally the absolute center of the custom mesh is used as the origin for the built sprout mesh, unless this offset is assigned. The offset values are affected by the common scale value shared by all other meshing modes.



Custom mesh with no offset.



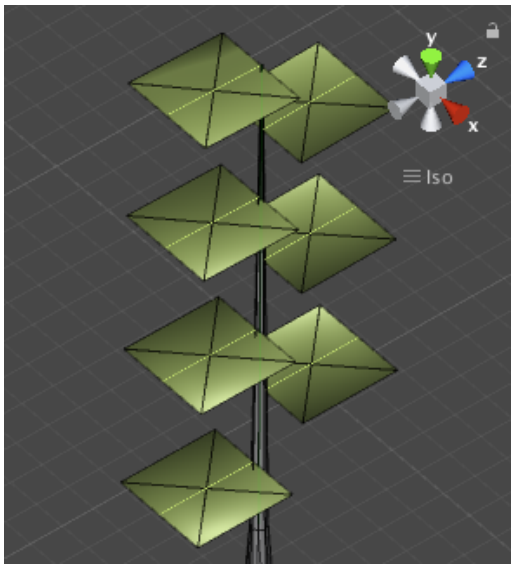
Custom mesh with offset on the z axis = 1.

X Plane Mode

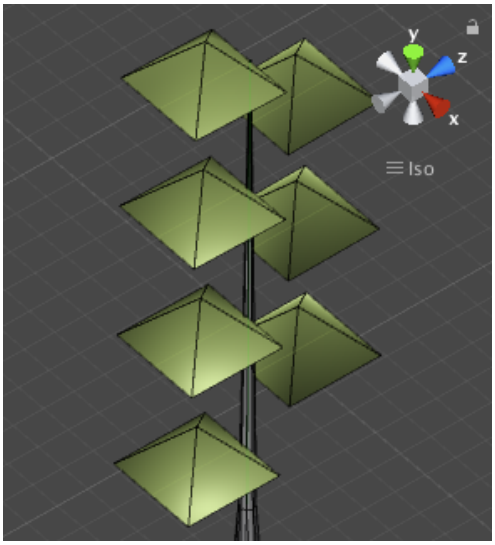
Uses a plane that might turn into a pyramid-shaped mesh assigning a depth value to its center.

Depth

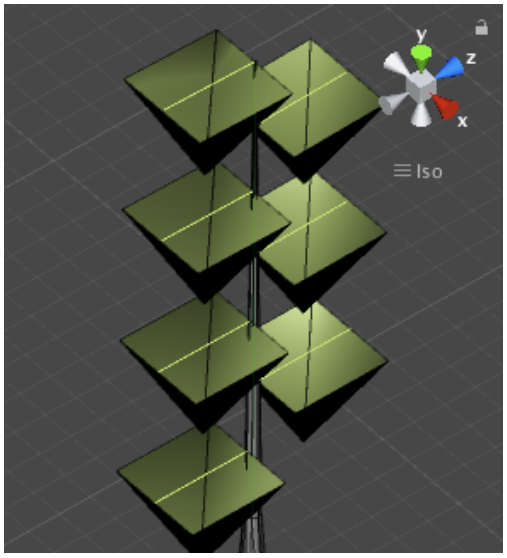
How deep the center of the plane is in reference to its borders.



X planes with depth = 0.



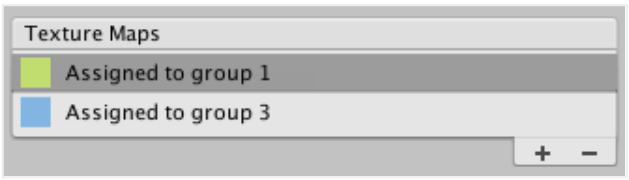
X planes with depth = -1.5



X planes with depth = 3

Sprout Mapper Node

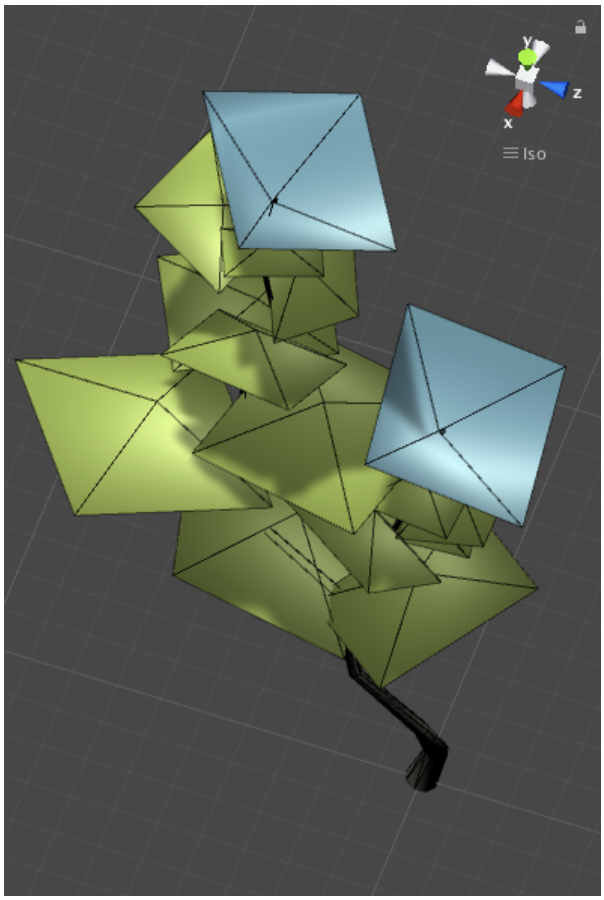
Sprout mappers apply materials to sprout meshes. The node has a list displaying the mappers, every mapper should be assigned to a sprout group in order to get all the meshes belonging to that particular group with a material.



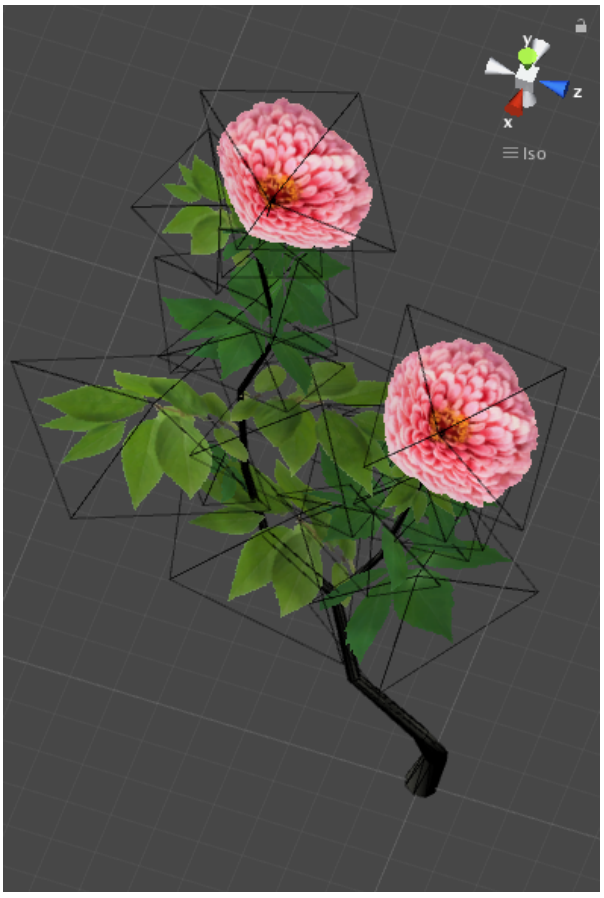
List of mappers for sprout groups.

Sprout Group

Assign the mapper to a specific sprout group. Then all directives on the mapper should be applied to sprout meshes belonging to that group. Two modes are available, custom material an texture mode.



Two sprout groups on colored preview mode.



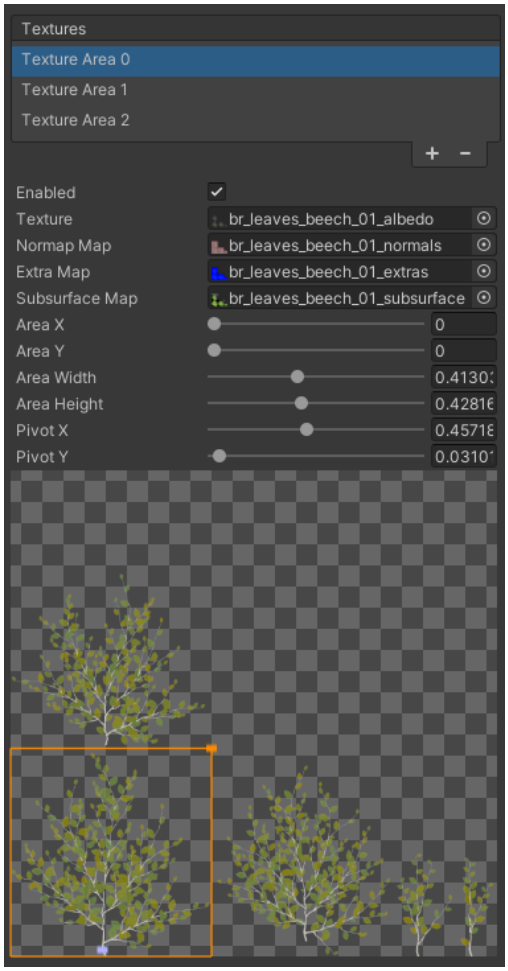
Two sprout groups with textured maps using 3 textures.

Texture Mode

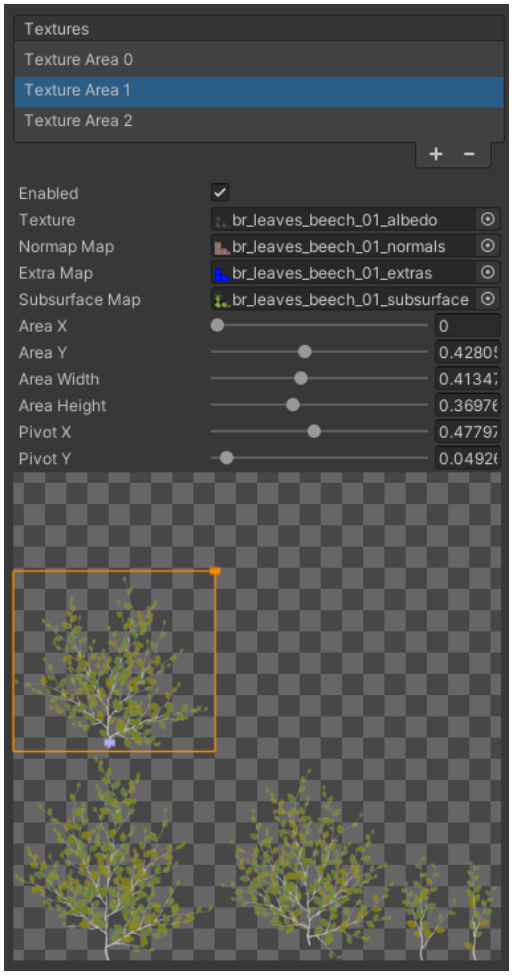
This modes take texture assets and define a region on them to create material for the sprout meshes.

Texture

Every texture mode mapper can have one or many textures assigned, these textures will be randomly assigned to each individual sprout mesh belonging to the group. A region should be defined per texture to set the boundaries of it (red square) and the origin of the sprout (blue dot).



Texture 1 assigned to sprout group 1.



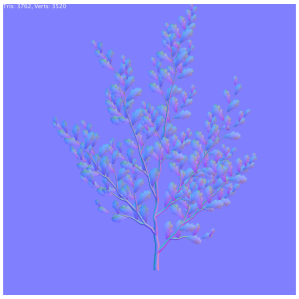
Texture 0 assigned to sprout group 1.

Each mapper can be assigned to 4 textures, each one containing values used by the PBR material:

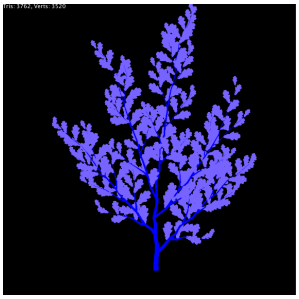
- **Texture (Albedo):** the sprout texture flat (unlit) colors.
- **Normals:** normal values in object space according to the orientation of branches and leaves.
- **Extras:** this special texture contains values for metallic, glossiness (smoothness) and ambient occlusion within a RGB color value. Red for metallic, green for glossiness and blue for AO.
- **Subsurface:** Subsurface scattering (SSS) is a mechanism of light transport in which light that penetrates the surface of a translucent object is scattered by interacting with the material and

exits the surface at a different point. It could be viewed as a map to specify the areas where the light will get through leaves.

Albedo Map



Normal Map

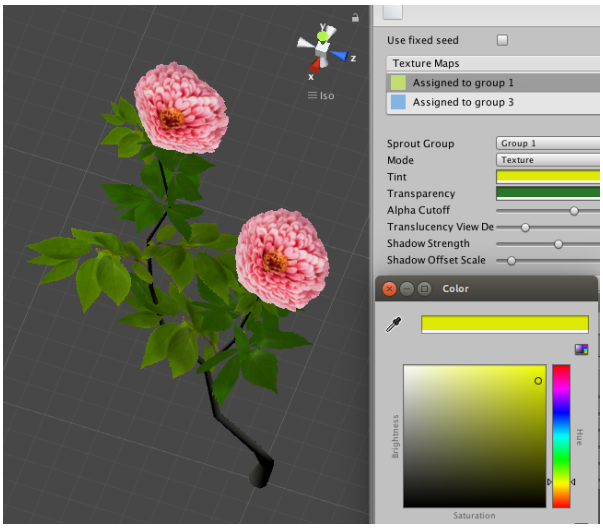


Extras Map

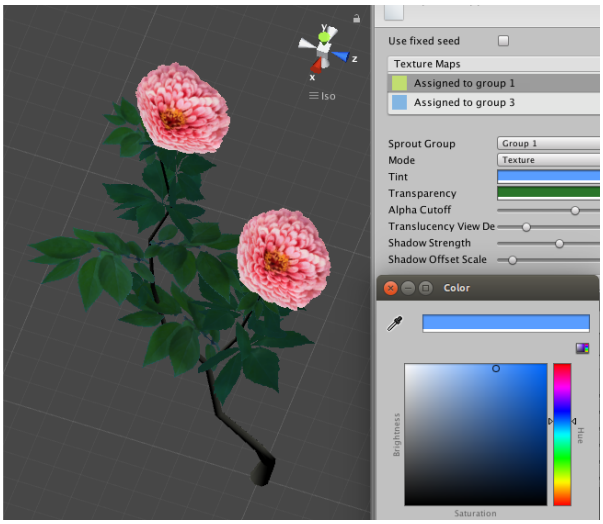
Subsurface Map

Tint

Adds a tinted color to the texture. White is the default value, meaning no tint.



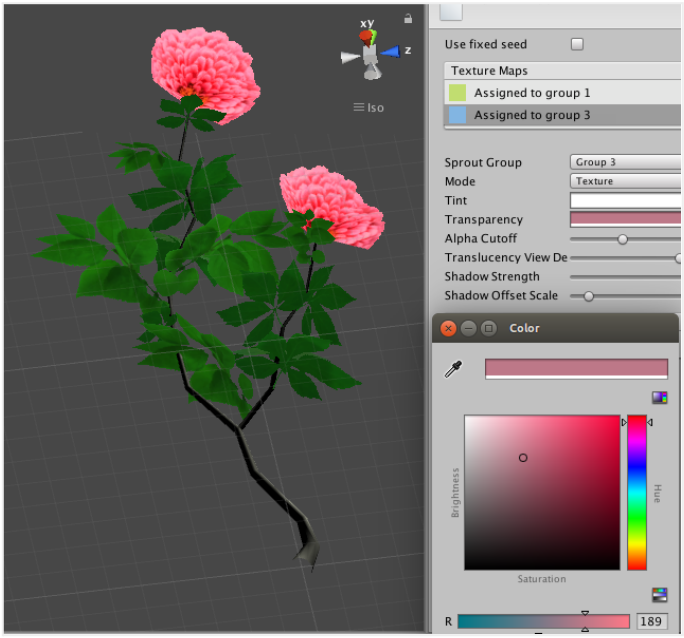
Sprout group 1 textures with a yellow tint applied.



Sprout group 1 textures with a blue tint applied.

Transparency

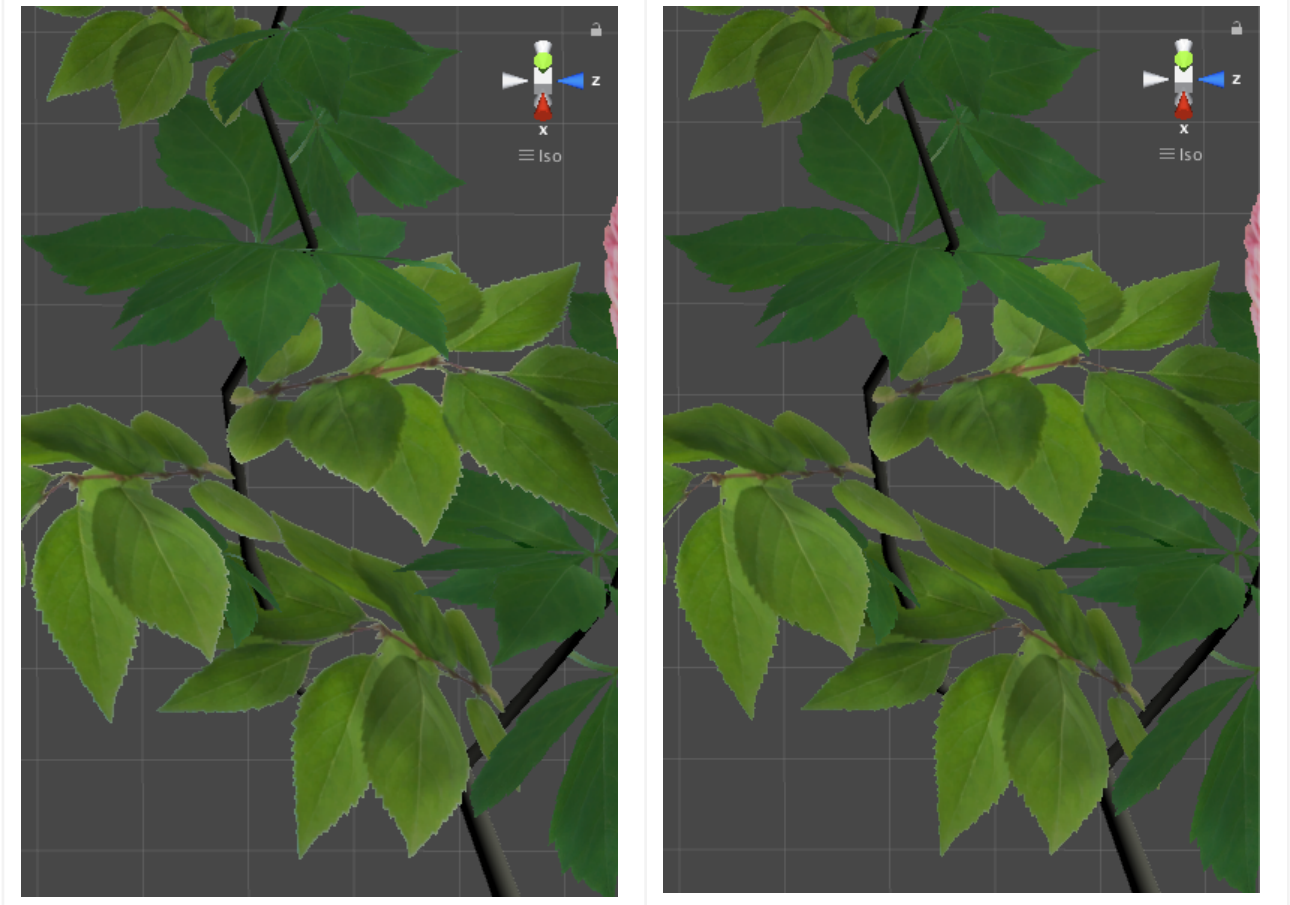
Transparency refers to the color the meshes let pass through from the light source when directly against it.



Meshes looked against the light source showing the transparency color set on them.

Alpha Cutoff

Defines how many of the semitransparent pixels on the texture make it to the final render based on its alpha value.



Border of the leaf textures with alpha cutoff = 0.2	*Border of the leaf textures with alpha cutoff = 0.6*

Subsurface Color and Value

This value refers to the color and quantity of light emitted by the mesh material from a light source.

Leaves bouncing little light from their side against the light source. Subsurface = 1

Leaves bouncing some light from their side against the light source. Subsurface = 0.5

Leaves bouncing all light from their side against the light source. Subsurface = 0

Glossiness and Metallic

The glossiness or roughness value controls the smoothness of the surface. It defines how blurry or sharp the reflections on the surface are. A glossiness value of 0 indicates a rougher surface with blurry reflections, scattering light in various directions. A glossiness value of 1 indicates a perfectly smooth surface with sharp reflections, similar to a mirror. This simulates the effect of microsurface imperfections that cause light to scatter instead of reflect directly.

The metallic value determines whether a material is more like a metal or a dielectric (non-metallic) surface. It controls how the material reflects and absorbs light. A metallic value of 0 indicates a dielectric surface (like plastic, wood, or glass) that has no specular reflections. A metallic value of 1 indicates a fully metal-like surface that reflects light in a mirror-like fashion and has specular reflections. These materials have a more reflective and glossy appearance.

These values are taken directly from the Extras Texture if set on the texture list of the mapper (red channel for metallic, green channel for glossiness); ignoring the values set on the glossiness and metallic on the slider controls.

Glossiness = 0.1, Metallic = 0.1

Glossiness = 0.75, Metallic = 0.1

Glossiness = 0.1, Metallic = 0.8

Glossiness = 0.75, Metallic = 0.8

Material Mode

A custom material could be applied directly to a sprout group mesh.

Custom material set on sprout group 3.

Branch Mapper Node

This mapper is used to set the UV mapping for the mesh coming from the branch structure of the tree. Mapping is necessary to get this mesh correctly rendered with a material.

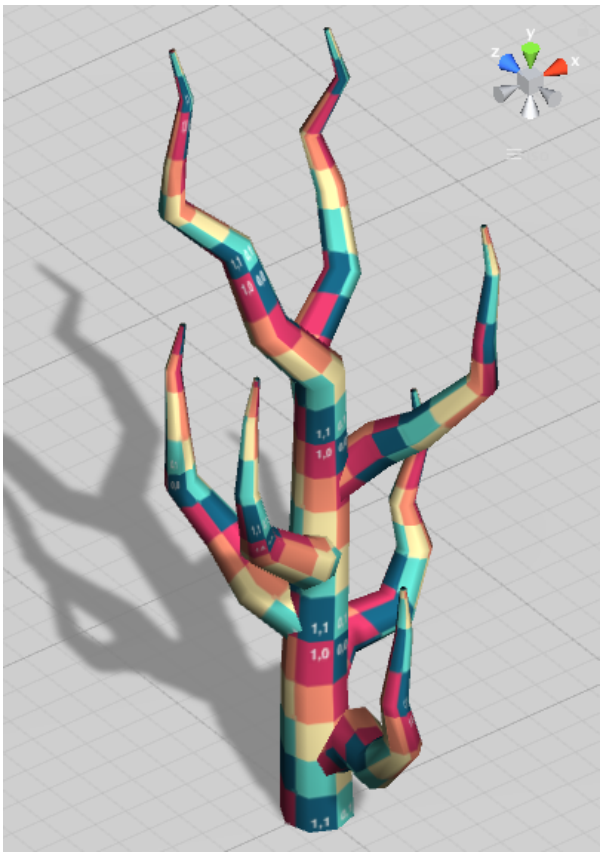
Use Custom Material

Checking this option allows the selection of a custom material to be applied to the branch structure mesh. If this option is not checked then the mapper will create a material from the textures set on them.

Mapping using a custom material on the mesh.

Main Texture

The main texture on the material. Is advisable that the texture is marked as readable for the texture atlas creation process when exporting a tree to a prefab.

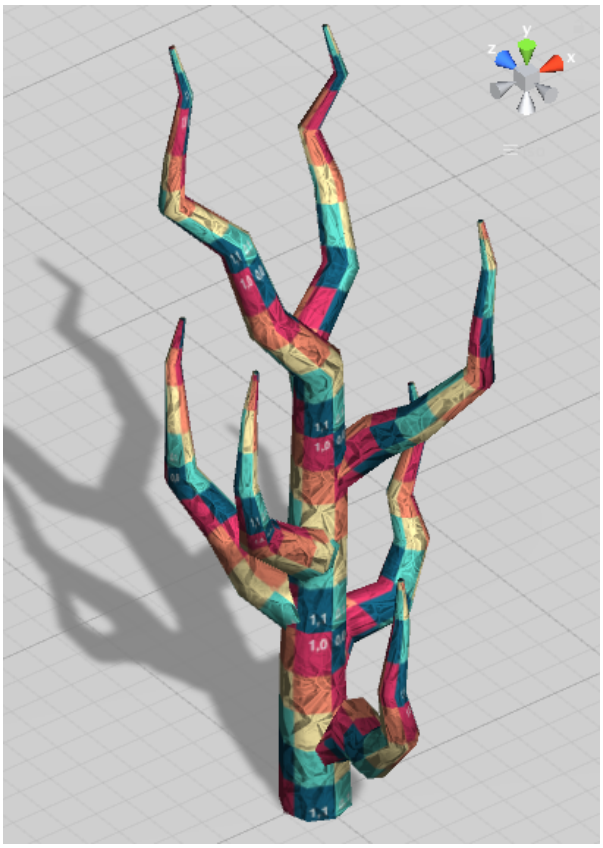


Mapping with main texture applied.

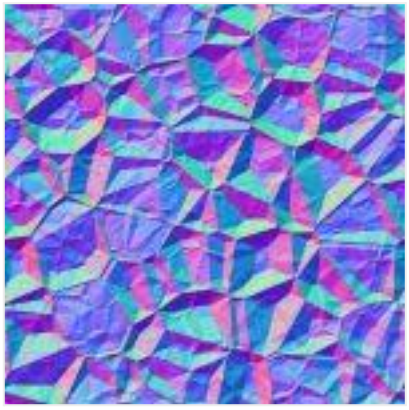
Main texture.

Normal Texture

The normal texture on the material. This texture should be marked as a normal map to be usable on the material and be readable for the texture atlas creation process when exporting a tree to a prefab.



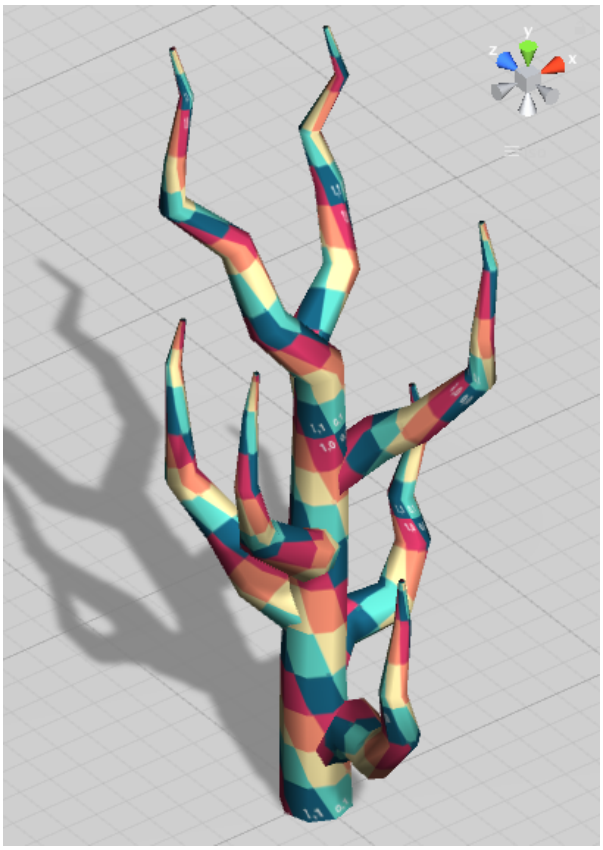
Mapping with main texture and normal texture applied.



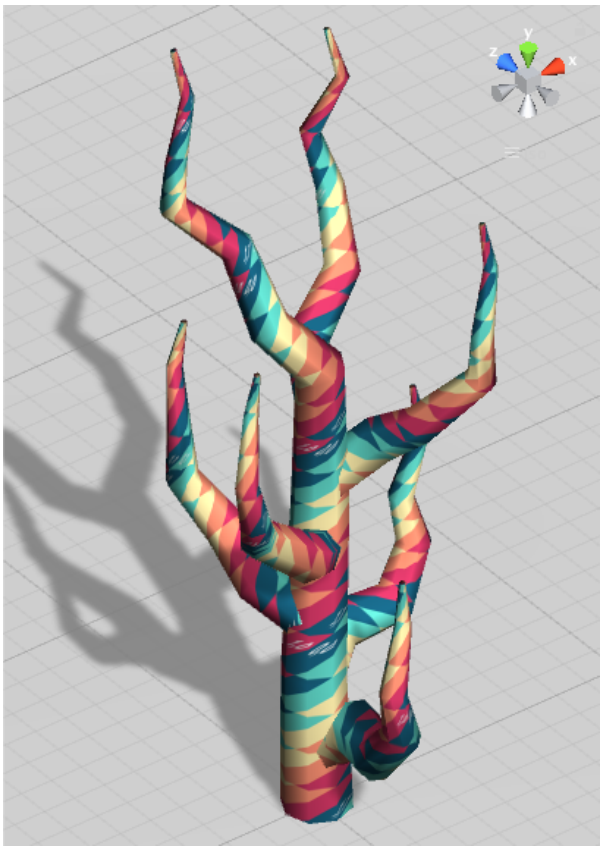
Normal texture.

Mapping X Displacement

The x displacement twists the UV mapping around the branches. This is useful to disrupt visible repeated patterns when using seamless textures or to get interesting effects with the texture.



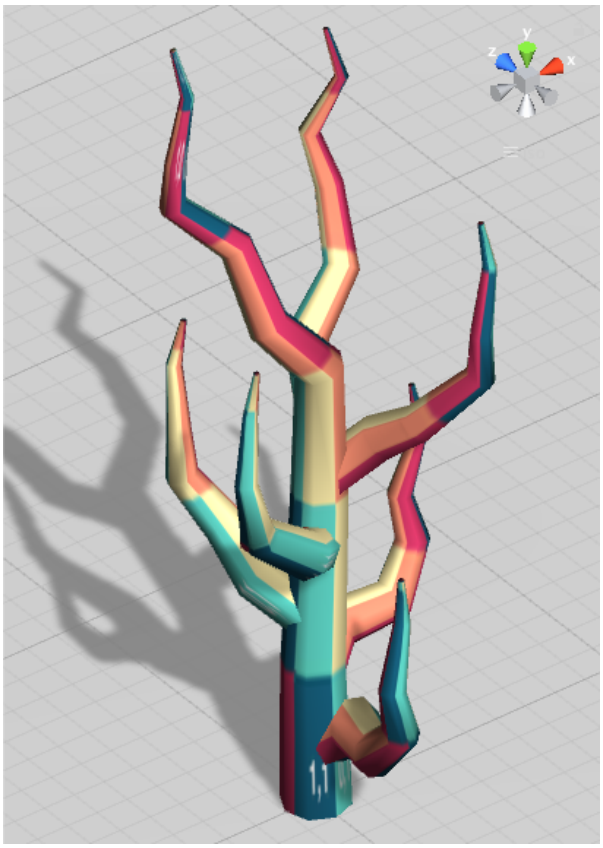
UV mapping with X displacement = 0.5



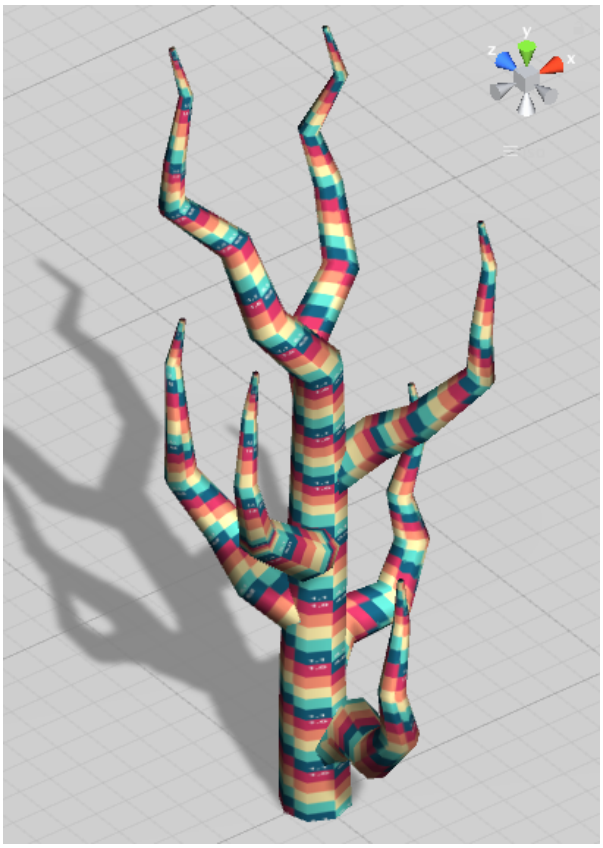
UV mapping with X displacement = -2

Mapping Y Displacement

The y displacement squeezes or lengthens the UV mapping along the branches. Both x and y displacement values can be combined to obtain a desired UV mapping on the branch structure mesh.



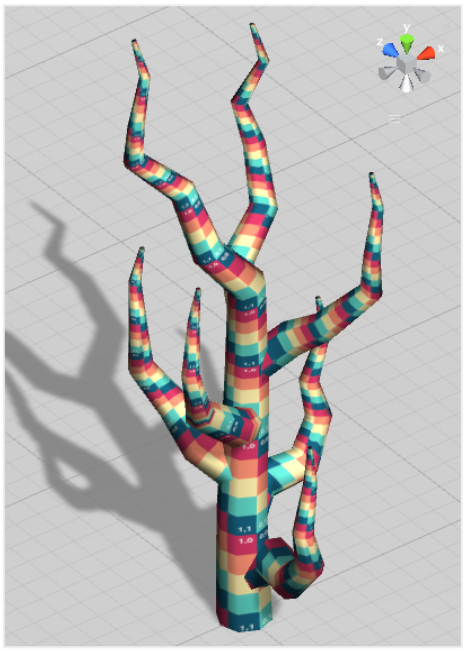
UV mapping with Y displacement = 3



UV mapping with Y displacement = -2.5

Is Girth Sensitive

Checking this option turns the UV mapping responsive to branch girth chances along the branch structure mesh.



UV mapping with girth sensitivity turned on.

Wind Effect Node

This element applies wind weight by applying values on the UV channels in the tree mesh. This enables the generated trees to be rendered using a shader that simulates wind (by default the SpeedTree8 shader that comes with Unity) and be reactive to WindZone objects on the scene (currently, the controller only supports WindZone Directional mode) or to be controlled/animated with a script.

When using Prefabs the wind effect is applied through a tree instance GameObject component: BroccoTreeController. When the scene starts the wind parameters used to calculate the wind effect are taken from two sources: a WindZone GameObject or set by script. You can choose the initialization method for these parameters on the BroccoTreeController component attached to the tree instance GameObject (nested LODs if working with level of detail) by setting the Wind Source variable:

- **WindZone**. The wind parameters (Main, Turbulence, Direction) are taken from the first active WindZone GameObject set to directional wind found in the scene.
- **Self**. The wind parameters are taken from either the localWindParams or globalWindParams structures on the component, depending if WindInstance is set to Local or Global (see below).

When using Prefab assets created to be reactive to WindZone components, please call BrocoTreeController UpdateWind method to update wind values on the tree instances. There are two modes for instancing wind on the trees set at the Wind Instance variable:

- **Global**. Wind values are calculated once per frame and applied to all the tree instances set to using global wind. This is preferred performance-wise. Every time you want to update wind parameters on these tree instances (from a WindZone or custom set) you need to call WindUpdate on just one BroccoTreeController component to be applied to all instances.
- **Local**. Wind values are calculated once per frame and applied exclusively to the tree instance holding the BroccoTreeController component being updated. This allows you to have trees with different wind parameters in the scene at the cost of individual calculations for wind.

Broccoli UV mapping for wind is tuned to work with the SpeedTree8 shader implementation for wind. Wind parameters (either taken from a WindZone or from script) are best tuned for values of WindMain from 0 to 4 and WindTurbulence from 0 to 3. Here are some references on the type of wind depending on these values:

- **Calm Breeze**. Main: 0.5, Turbulence: 0.5. Gentle movement of leaves and little swaying of branches.
- **Breeze**. Main: 1.0, Turbulence: 1.0. Noticeable swaying of leaves and branches.
- **Windy**. Main 2.0, Turbulence: 1.6. Some trunk bending occurs on young trees.
- **Strong Wind**. Main 3.0, Turbulence: 2.3. Strong leaves and branches swaying with some bending on sturdy tree trunks.
- **Stormy**. Main: 4.0, Turbulence: 3.0. Violent movement of leaves and branches with back and forth fast bending on young trees.

These wind modes are also available for you to test your tree's wind mapping on the Wind Effect Node.

Each type of tree has its own physical properties when it comes to interacting with wind; for example, a bamboo trunk should be more bendable than an old and sturdy oak tree. These properties are directly baked on the UV channels of the mesh and can be controlled by the following parameters using the Wind Effect Node.

Sprout 1 Sway, Sprout 2 Sway

This value controls how much the sprouts will move slowly or rhythmically backward and forward following the wind direction and force. There are two sprout sway patterns to apply according to the pattern assigned to the Sprout Group on the Sprout Generator Node.

Branch Sway

This value controls how much the branches will move rhythmically backward and forward following the wind direction and force. This property affects only the branches coming from the trunk and not the trunk itself.

Sturdy branches from old trees should (in theory) be less affected by wind and sway less, compared to young trees or thin main branches where the sway effect should be more noticeable. The recommended value for sturdy branches is 0.5 and for thin branches anywhere from 2.5 to 4.0.

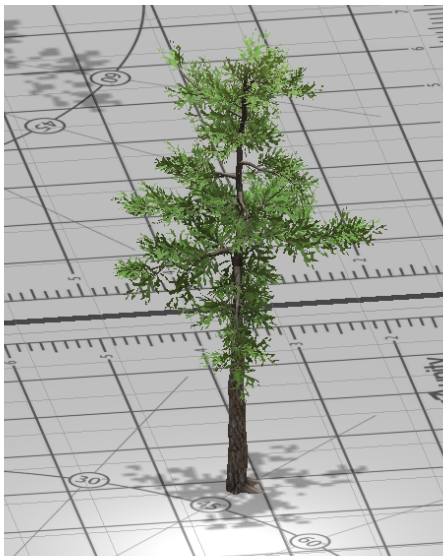
Branch Sway Flexibility

This value extends the Branch Sway property controlling how much the sway flexes the branch (like a whip) following the wind direction and force. This property affects only the branches coming from the trunk and not the trunk itself.

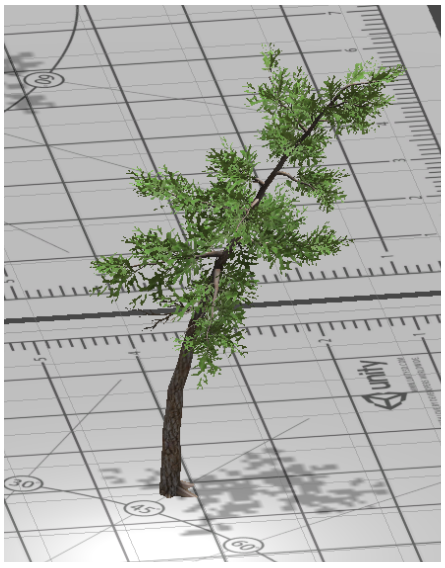
Trunk Bending.

The value controls the bending factor for the trunk of the tree. A value of 0 produces a tree whose trunk does not bend with the wind force (maintains its direction). The swaying effect of the children branches (coming from the trunk) does not depend on the trunk bending factor.

For old and sturdy trees a value of 0.5 is recommended, while for younger trees any value from 1.5 to 2.0.



Trunk bending = 0



Trunk bending = 2

Wind Quality

Sets predefined modes to apply wind to the tree according to the SpeedTree8 shader. A value of ‘best’ applies all the wind effects available on the shader, at the cost of being more graphical processing intensive; a value of ‘fastest’ applies only trunk bending, and it is preferred on trees that don’t require much wind detail (far away assets, low poly or low LOD assets).

Reset Wind

Sets default values for all the wind properties in the Wind Effect Node.

Preview Wind Mode

Allows you to test your wind parameters on various wind conditions. The selected wind mode does not get baked in the final tree mesh, as it serves only for testing purposes.

Preview Wind Always

Checking this option maintains the wind effect preview on the Scene Editor even when the Wind Effect node is not selected. Remember that in order to preview the wind effect, make sure the scene has a WindZone GameObject enabled (directional wind zones supported only) and the "Animated Materials" property enabled on the Scene View panel.

Wind for Unity Terrains

To apply the wind effect to Broccoli Trees painted on Unity Terrains, a BroccoTerrainController component needs to be added to the Terrain GameObject. This component updates the wind effect on the tree instances and can receive updates for the wind parameters using the UpdateWind method.

Positioner Node

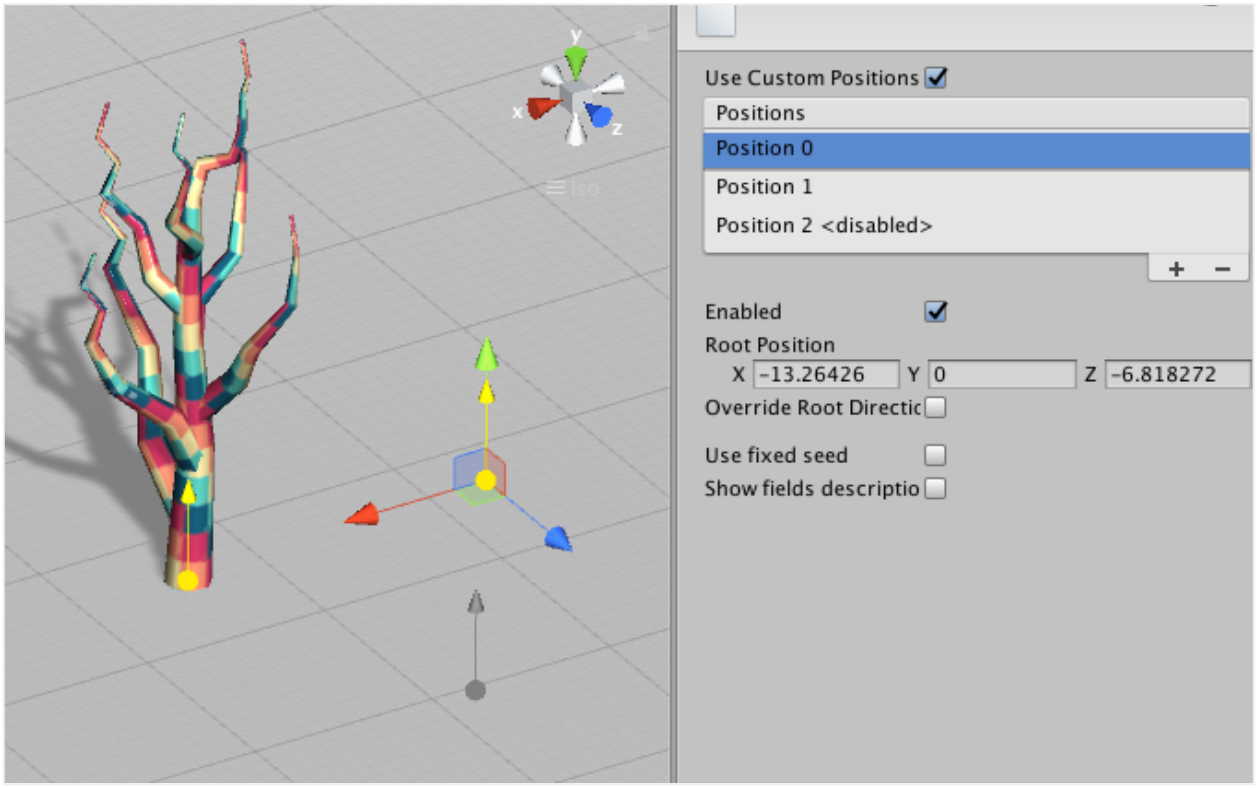
The positioner is a closing element for the pipeline, it positions the final tree on a location on the scene.

Use Custom Positions

By default the final location for a tree is the factory transform. If the “Use Custom Positions” is marked then the positioner randomly selects between a list of locations to position the final tree.

Positions List

The list of positions to place the tree. Selecting one of the items on the list displays this item’s properties.



Tree built on a custom position.

Enabled

Only enabled positions on the positioner are taken as possible locations for the tree.

Root Position

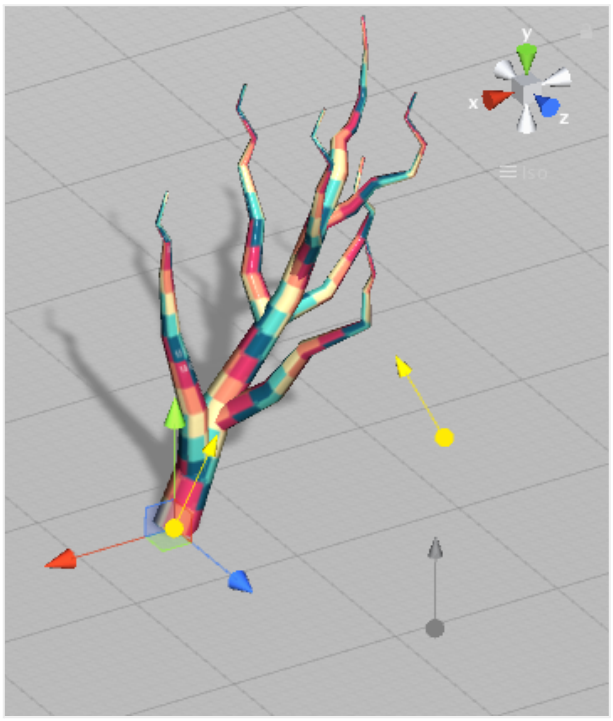
The Vector3 property that marks a possible origin for a tree. The point that displays the position is draggable on the scene view.

Override Root Direction

If this property is checked the direction of the tree trunk could be modified by the assigned position.

Root Direction

The Vector3 property for the direction of the tree trunk.



Tree tilted according to the custom location root direction.

Baker Node

Contains optional baking options and colliders for the final tree, either for previewing, runtime or prefab creation.

Add Collider

Adds a capsule collider at the base of each tree trunk.

Collider Scale

Scale to apply to the radius of the capsule collider. The base value for the radius is the max girth at the tree trunk.

Enable AO

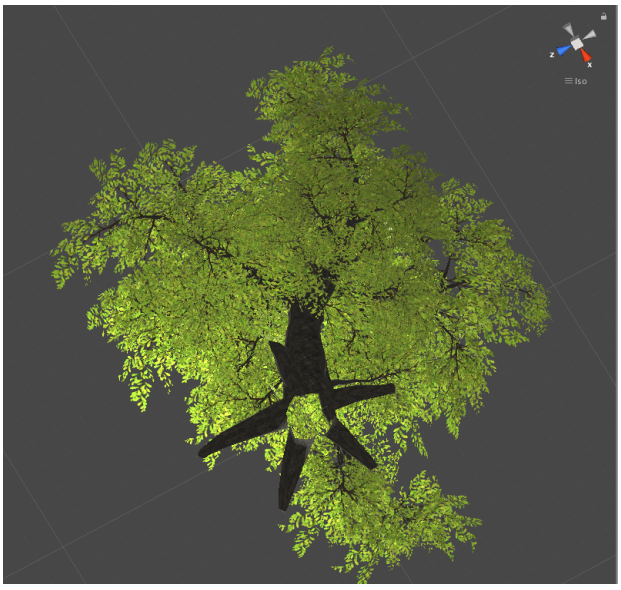
Applies ambient occlusion to the tree mesh, baking it in the color channel.

Samples AO

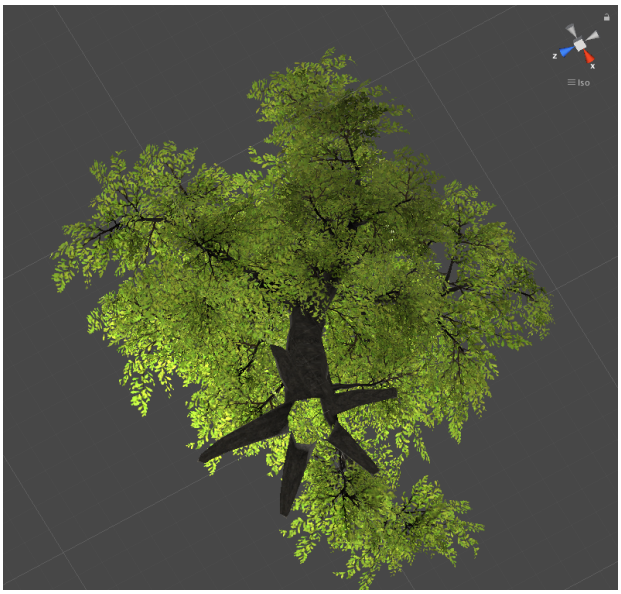
Number of samples for the ambient occlusion, the higher the number, the higher the fidelity of the baking.

Strength AO

Strength factor on the ambient occlusion calculation. From 0 to apply no occlusion at all to 1 to apply the full value.



No ambient occlusion baked on the mesh.



Ambient occlusion baked with strength=0.6.

Graphics

This structure generator element is modeled after Unity's Tree Creator component, so most of the properties described here should come familiar if you have used this tool before. The whole structure is described using level nodes in a hierarchy, each node contains properties to generate the branches at the assigned level. The structure here generated is taken as a base to be modified by other nodes downstream the pipeline, meshed and textured. Take this structure as the spatial data the tree will be built upon.

Universal Render Pipeline

SpeedTree shaders (v7 and v8) are supported within the Scriptable Render Pipeline package; however, they do not use global illumination, leading to dark hard shadows. To fix this you have the option to either use a custom URP shader compatible with SpeedTree properties (or TreeCreator) without this problem or you can patch the version of the URP on your Unity project by cloning Unity's Graphics repository, applying the patch manually and then installing the URP as a local package.

Here are the steps to proceed with the local patched URP package solution, this only applies to the SpeedTree v7 shader. (thanks to StaggartCreations for contributing with this! https://forum.unity.com/threads/still-no-speedtree-support.894025/#post-5946692):

1. Clone the Unity Graphics repository (https://github.com/Unity-Technologies/Graphics).

   git clone https://github.com/Unity-Technologies/Graphics

2. Go to the root folder of the repo and select the tag for the URP version your project is using (use the right tag for your package version).

   git checkout v7.3.1

3. Open the SpeedTree7CommonPasses.hlsl file at:

   /com.unity.render-pipelines.universal/Shaders/Nature/SpeedTree7CommonPasses.hlsl

4. Locate the following lines:

   ```
   float3 positionWS               : TEXCOORD7;
   float4 clipPos                  : SV_POSITION;
   ```

5. Add the TEXCOORD8 line:

   ```
   float3 positionWS               : TEXCOORD7;
   float4 lightmapUVOrVertexSH     : TEXCOORD8;
   float4 clipPos                  : SV_POSITION;
   ```

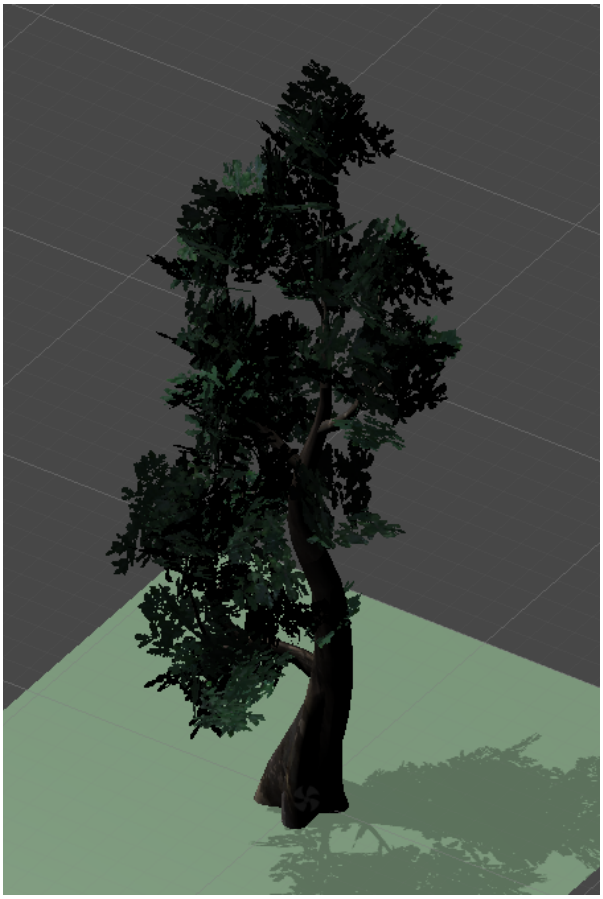
6. Locate the following lines:

inputData.fogCoord = input.fogFactorAndVertexLight.x;
inputData.vertexLighting = input.fogFactorAndVertexLight.yzw;
inputData.bakedGI = half3(0, 0, 0); // No GI currently.

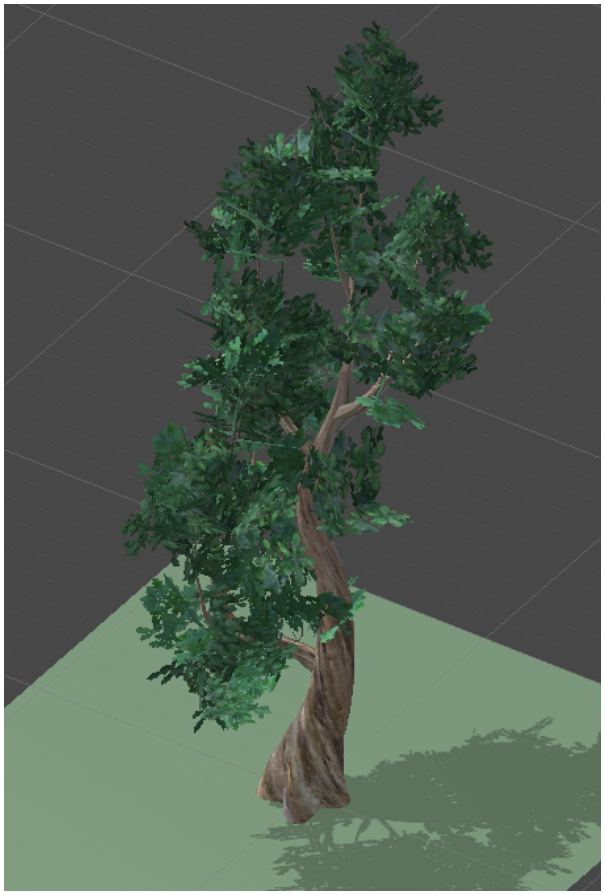
7. Add the OUTPUT_SH and inputData.bakedGI lines:

   OUTPUT_SH(inputData.normalWS.xyz, input.lightmapUVOrVertexSH.xyz);
   inputData.fogCoord = input.fogFactorAndVertexLight.x;
   inputData.vertexLighting = input.fogFactorAndVertexLight.yzw;
   inputData.bakedGI = SAMPLE_GI(input.lightmapUVOrVertexSH.xy,
   input.lightmapUVOrVertexSH.xyz, inputData.normalWS);

8. Save the file and install the patched version of the URP package with the Package Manager.



SpeedTree7 shader on URP (no fix).



SpeedTree7 shader on URP (fixed).

Please considering that there are still a lot of other issues with the SpeedTree shaders when using and Scriptable Pipeline (https://forum.unity.com/threads/case-1221827-urp-terrain-lightprobes-for-speedtree7-8-shaders-bake-black.833083/).