User Tools

Site Tools

Table of Contents

Proxies

Proxies are placeholder objects primarily used in world building.

Overview

“Proxies” serve as stand-ins for other objects in world-building. Once proxies have been created, their transform properties (Position, Rotation, and Scale) can be exported in an ASCII format to be read in by other apps. Proxies are created by proxy generators, and generally are used as children of zone generators. Each proxy generator can be assigned a list of proxy meshes instead of only a single type.

World Building

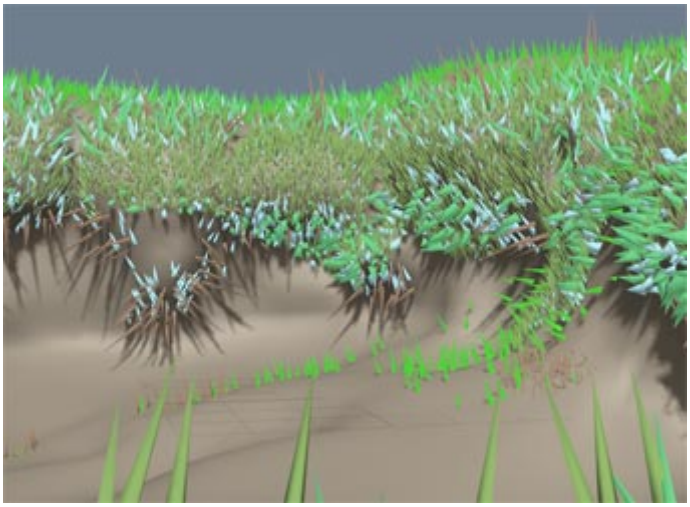
Proxies only function in a world building setup. Please see the world building tutorial for a detailed walk through of the world building pipeline:

Spikes

Each of these proxies, by default, are displayed as “spikes”. The spikes shown in the viewport are used to approximate the volume based on a Spike height and Spike radius property. Besides these transform properties, proxies hold very little additional information. Spikes can be given more identity by assigning a material to each spike entry. In that way, the transforms of the spikes can be exported by the assigned material name, or by the name of the proxy generator that created the spike node. If you find yourself in need of more distinction between spikes, you'll probably want to load mesh assets in place of spikes.

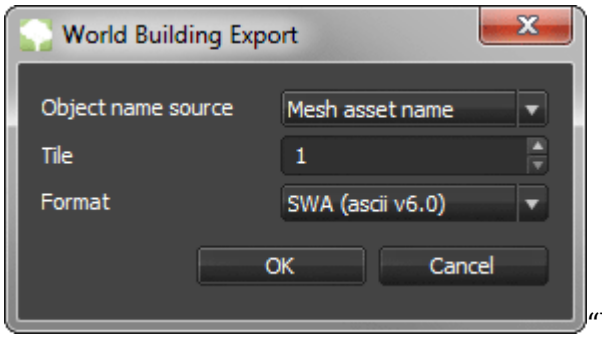
Draw Proxies As Spikes (Degradation Setting)

The Draw proxies as spikes option overrides the rendering of the mesh asset assigned to each proxy in the 'Tree Window' with a spike stand-in. This is useful when world-building with many proxy nodes using actual meshes as proxies, which can quickly bring your system to a grind. With this Degradation setting, the proxy generator can be assigned specific mesh assets for export reasons, but only show simple spikes in the Tree Window. The spikes will take the diffuse color of the color set associated with the material assigned to the proxy type to make it easier to tell the types apart.



An entire forest depicted as spike proxies via degradation.

Exporting Proxy Data



“World building data” refers to the transform properties (location, rotation, and scale) of all proxies in the scene. To export world building data, choose the menu option “**File → Export world building data**”. This will pop up the dialog depicted to the right, which provides the export options listed in this section.

Controlling which proxies will get exported
Any proxy that isn't "hidden" will get exported, regardless of its place in the tree hierarchy. To omit a particular proxy generator or node from being exported, hide the generator or nodes in question.

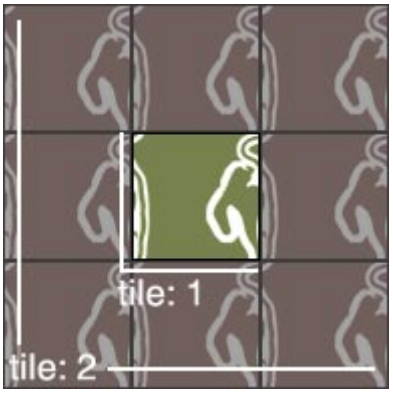
Object Name Source

This property dictates the naming scheme used in the world building file. Entries can be grouped based on the following criteria:

- **Mesh asset filename** - world building data is grouped by the filename on disk associated with each mesh asset. This is most useful when the world building data is to be read by the SpeedTree SDK (which requires actual filenames), since real-time SpeedTree meshes (SRT files) can be used directly in the world building process and their actual filenames exported with the SWA file. Full path filenames are not supported in the older STF format.
- **Mesh asset name** - world building data is grouped by the name of the mesh asset assigned to each proxy entry.
- **Material name** - world building data is grouped by the materials assigned to the proxies. The same material can be applied to multiple proxy entries, which will get grouped together during export.
- **Generator name** - world building data is grouped by the name of the generator housing each proxy node.

Tile

If your scene requires a great deal of proxy instances with random placement (such as grass blades), use the "Tile" multiplier to extend the area populated in the world building file. A value of "1"' results in the exact number of proxies in your scene being exported over the area of your scene. A value of "2" will result in nine total tiles, with the center tile being comprised of what actually exists in the scene. Each increment higher will add a ring around the last group of tiles in the same manner, as depicted below.

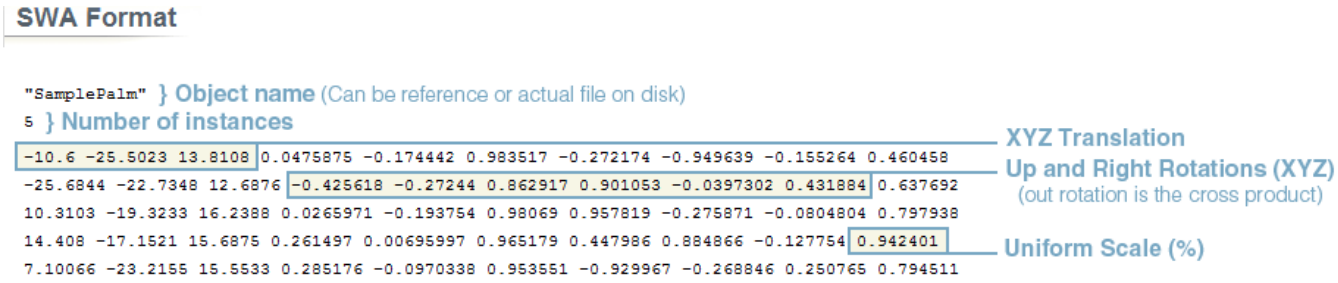


Format

World building data can be exported in two formats, SWA (SpeedTree World Building ASCII) format, or optionally the older STF (SpeedTree Forest) format. STF is deprecated, but is still supported in this release. The differences between the formats is as follows:

- SWA files can contain arbitrary rotations as well as absolute or relative path filenames. The filenames can be encompassed by quotation marks, allowing for filenames with spaces in them.
- STF files only contain one rotation angle, around the Z (up) axis. Filenames must not contain spaces, or backwards directory notations ("../"). STF should only be used if the format is already a part of your existing pipeline.

The following is a breakdown of the data stored in a standard SWA file:



World Building Tutorial

Follow the steps in this tutorial to get world building data between this and other apps: http://www.speedtree.com/demo-videos.php

---

Using Mesh and SRT Files

Meshes

Instead of simply using spikes as proxy objects, imported mesh assets can be assigned to entries in a proxy generator. The actual mesh asset will appear in the 'Tree Window' in place of the default spike. Using meshes opens up two extra options for exporting tree locations – by "mesh asset name" or by "mesh asset filename". The "mesh asset filename" option is best when using SpeedTree run-time (SRT) mesh assets.

SpeedTree Run-Time (SRT) FIles

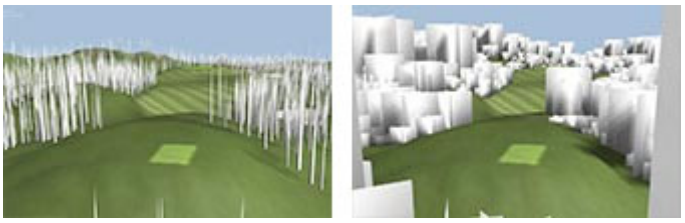
SpeedTree for Games Only: SRT files can be loaded as mesh assets and in turn used as proxies. SRT files are generated by the SpeedTree Compiler (a separate app). The Compiler is the conduit between the Modeler and the SpeedTree SDK. Once a forest of tree assets has been finalized, the Compiler will combine all textures into a single atlas, generate billboards, and convert the procedural tree models (SPM files) into efficient binary run-time versions (SRT files). Once the SRT files have been generated, they can be loaded back into the Modeler as mesh assets. SRT files are shown as cross-hatches instead of full-blown models, which can utilize billboard atlases for

low-detail representations of each tree.

Advantages of Using SRT Files as Proxies

There are a few added benefits to using SRT files as proxies, as outlined below:

- SRT proxies are resized based on the height of each SRT file. A single proxy generator may contain both an SRT file of a small shrub and one of a tall conifer, and each will be sized relatively to one another. If using spikes, both spikes would by default appear the same size in the Tree Window.



Above: (*left*) spike proxies, (*right*) SRT proxies exhibiting correct scale.

- SRT proxies can be assigned a material with the billboard map in the diffuse channel. Once assigned to an SRT proxy material slot, the correct billboard image will be displayed on the proxy cross-hatch.

Above: the billboard image is automatically texture mapped onto SRT proxies.

- SRT files can use their own collision during proxy collision detection. This is the most ideal way to compute collision detection, as only intersecting collision volumes will result in a cull, as opposed to simply the bounding box of the object. Additionally, SRT meshes have special collision properties located only on the meshes tab. For detailed info on proxy collision, see the following section.

Above: pre-culling (*left*), post-culling (*right*).

- SRT filenames should have a one-to-one correspondence with the tree assets used by the SDK. Tree locations can be exported as world building files, and read directly by the SDK, various 3D content creation packages, or a 3D game engine.

Above: The same scene with actual tree models being navigated in real-time through the SpeedTree Reference App.

Proxy Collision Detection

When using proxies for world building, the need will inevitably arise to perform collision detection on all proxies sharing a zone. If you do not use proxy collision detection, it is likely that proxies will overlap one another.

Above: pre-culling (*left*), post-culling (*right*).

Proxy collision is controlled via the global 'Tree Properties'. Much like leaf collision, proxy collision has three culling functions:

Automatic
Enabling this checkbox tells the app to perform proxy collision detection every time that proxy nodes are recomputed. This option may be too slow with scenes containing thousands of nodes. When off, collision detection will need to be recomputed each time the file is loaded.

Cull
Performs the culling process. Overlapping proxies will be removed based on their weight scale and proximity to other nodes.

Restore Culled
Restores all previously culled proxy nodes.

Collision Scale

Proxy collision detection is performed with a hierarchical approach to ensure that both simple and complex collision schemes can be utilized. Proxies are assigned both a collision weight and a collision scale through generator properties. SRT mesh assets have an extra layer of collision control (more on that later).

Each proxy generator has a group called “Collision” at the bottom of it's property list. Here, you can alter the size of each proxy's extents.

Global Scale

Global scale affects how proxy nodes from other generators will collide with the nodes from a specific generator.

(*left*) standard collision, (*right*) increased “global” collision on the pines.

As a practical example, one proxy generator may contain pine trees, while another may contain shrubs. Setting the “global scale” of the pine tree generator to “0.5” would result in all other generators (including the shrubs) being able to get within the default bounds of each pine tree – closer to the trunk by half. At the same time, each pine tree will still collide normally with its siblings, ensuring no overlap of pine branches.

Self Scale

Self scale is the counterpart to “global scale”. Instead of affecting how other generators will collide with it, “self scale” affects how nodes within a single generator collide with each other.

(*left*) standard collision, (*right*) increased “self” collision on the pines.

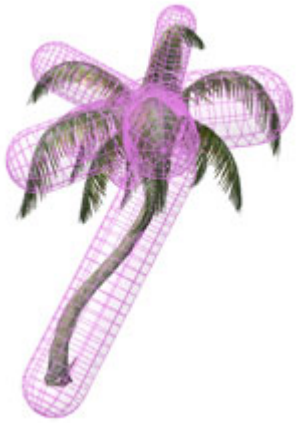
As with the above example of pine trees, the pine tree generator could have a “self scale” of “2.0”, meaning that each pine tree would be spaced out twice as far only from every other pine tree in the same generator.

Weight Scale

Weight scale provides preference to which proxy nodes are more likely to “survive” the culling process. Higher weight values mean a higher likelihood of survival. Weight can be used to place importance on more dominant tree types.

With the same pine tree example, the shrubs should never "win" over a pine tree during collision, since the culling process could conceivably remove all of the pine trees, resulting in a barren-looking scene. Instead, if a weight scale of "2.0" was assigned to the pine tree generator, the pine trees would always survive the culling process.

SRT Mesh Collision



SpeedTree for Games Only: SRT mesh assets provide a much more accurate way of colliding proxy nodes. Since SRT files contain the same collision volumes that exist in the procedural version of the tree (the SPM file), collision can be performed in a way that matches the actual geometry of the real tree. For instance, this allows for trees with wide canopies to extend branches over other shorter trees without clearing a broad area below their canopies.

SRT Mesh Asset Collision Properties

Each mesh asset that is assigned an SRT filename is provided two additional properties: a mesh level "collision scale" and "weight scale".

Changing the "collision scale" will scale the size of each individual collision volume within the SRT file. For instance, the palm tree to the left has seven capsules for collision, and each individual capsule would be scaled by the collision scale factor.

"Weight scale" works the same as the "weight scale" property of proxy generators (described in the previous section) – higher weight values take precedence over lower weight values during the culling process. The added advantage of setting the weight scale on the mesh asset rather than on the generator alone is that proxies with varying weights could exist within a single generator.

-
- Questions?