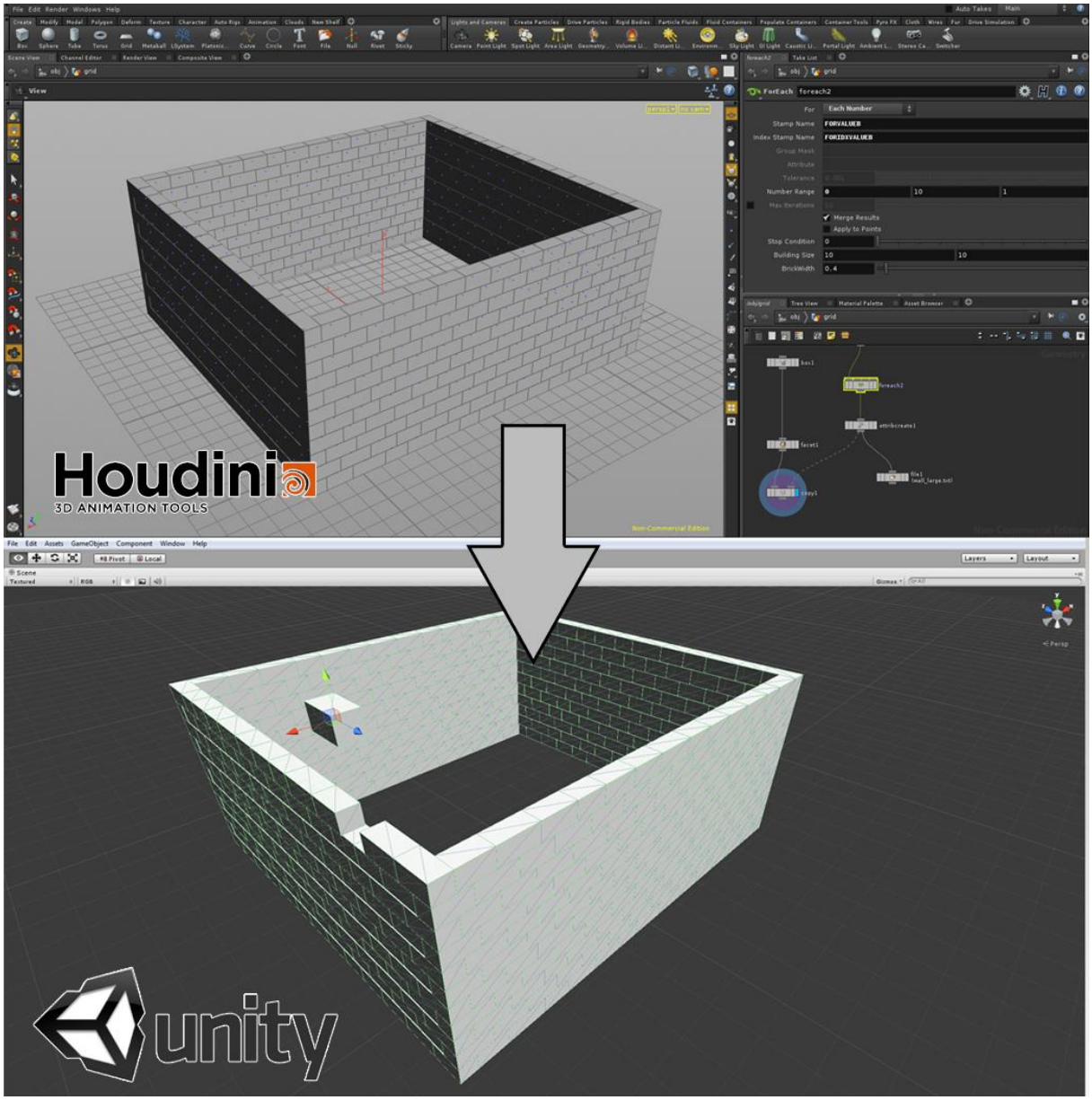
HOUDINI 12&13
POINTIMPORTER FOR UNITY

PROCEDURAL LEVEL GENERATION MADE EASY



A HOUDINI12 .GEO FILE PARSER FOR UNITY.

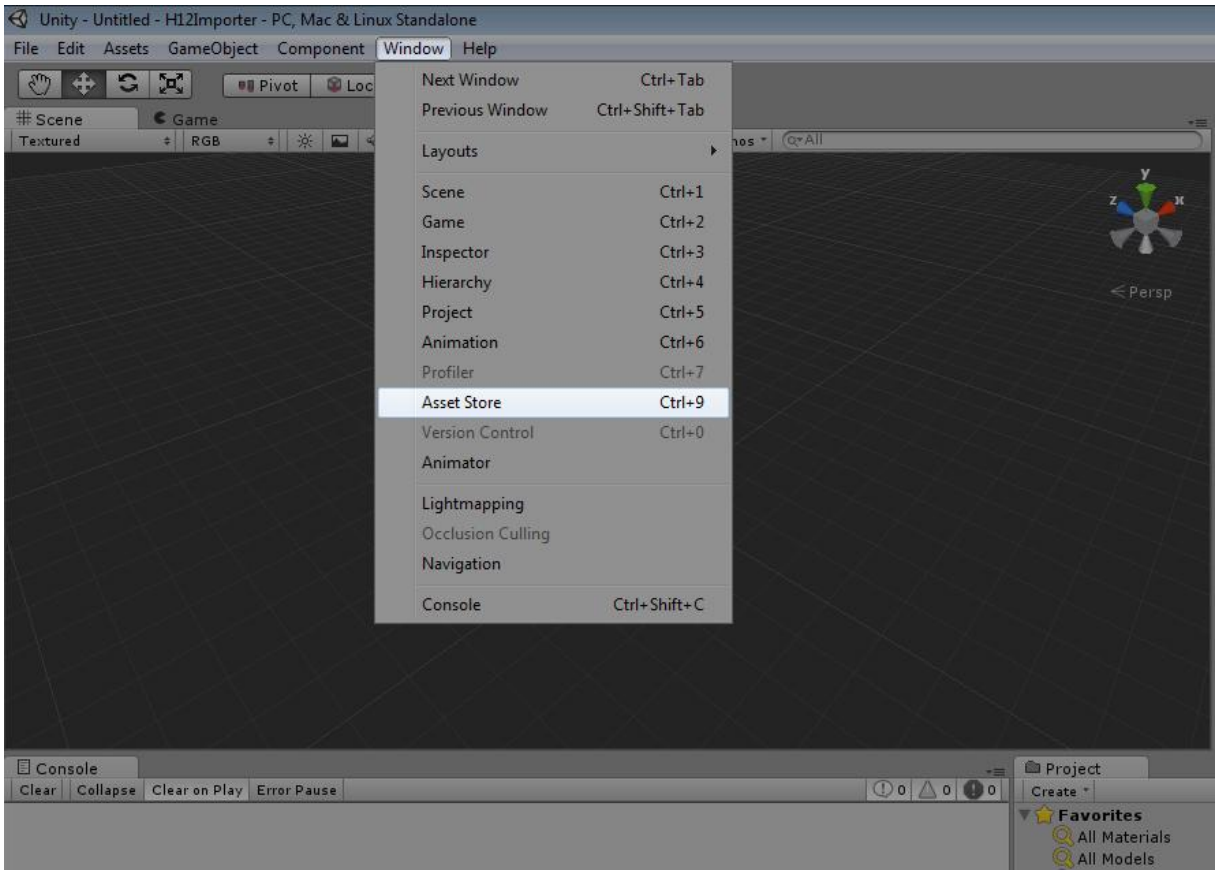
WRITTEN BY: TWAN DE GRAAF

TWANDEGRAAFTECHART@GMAIL.COM

CONTENTS

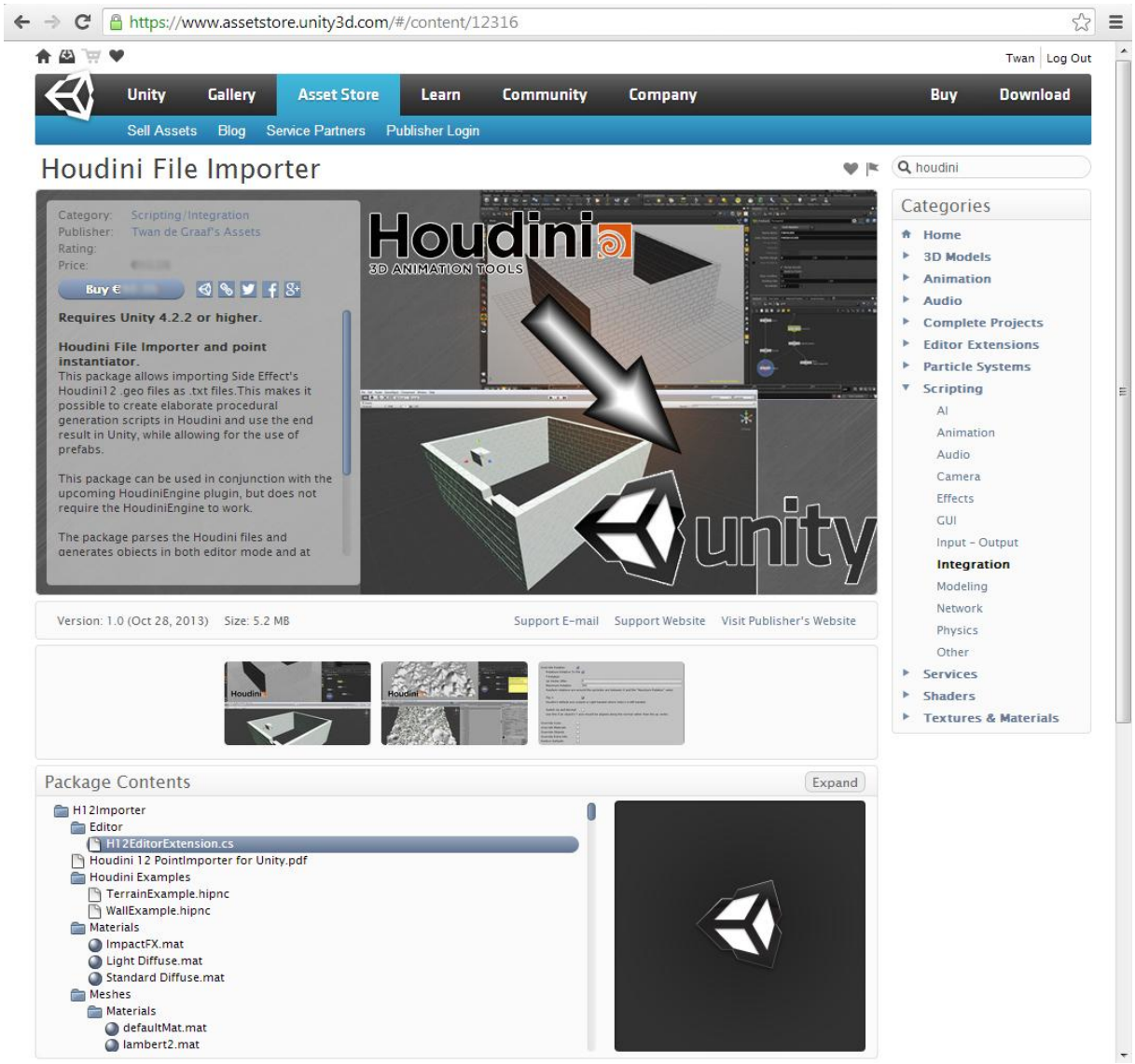
INSTALLATION

The installation of the tool inside unity is done via the Unity Asset Store



Inside the asset store you can browse for the file under Scripting/Integration, or you can simply use the search bar to find the asset package "Houdini File Importer". Via the Asset store you can download the package and install it in as much projects as you like.

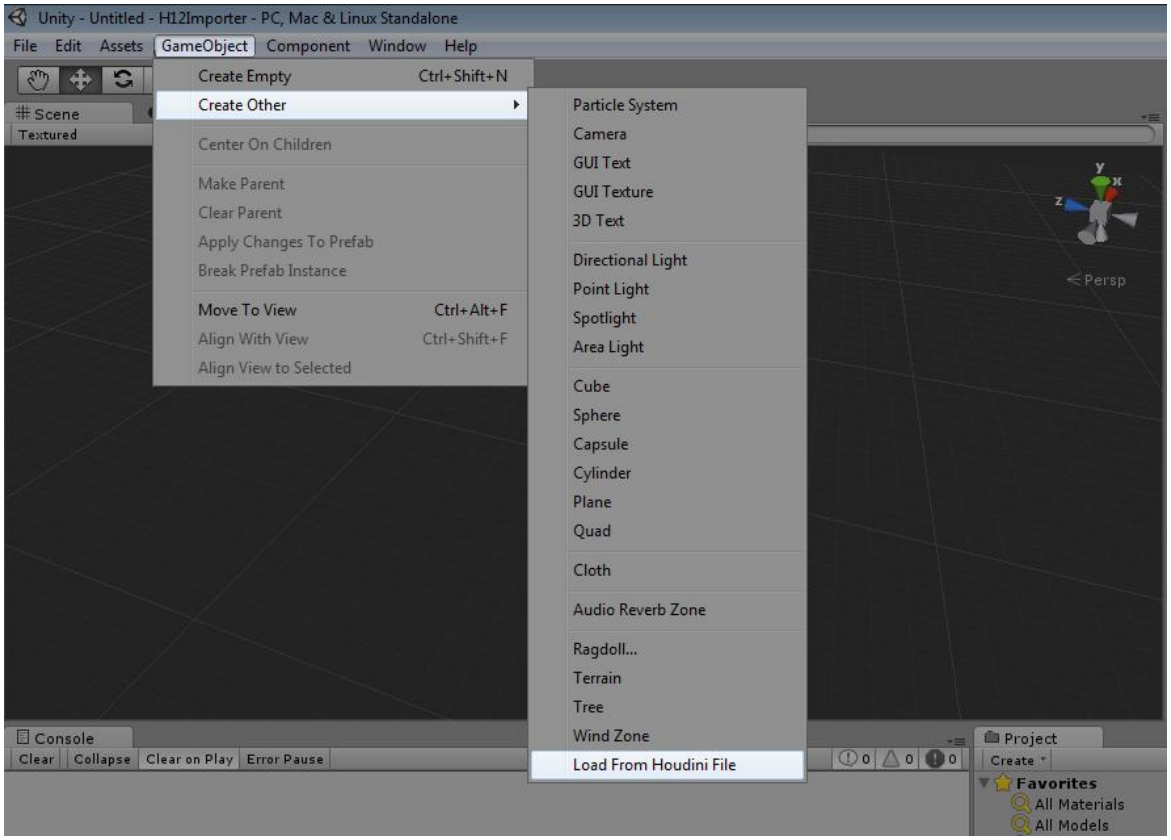
It is also possible to access the page from an internet browser.



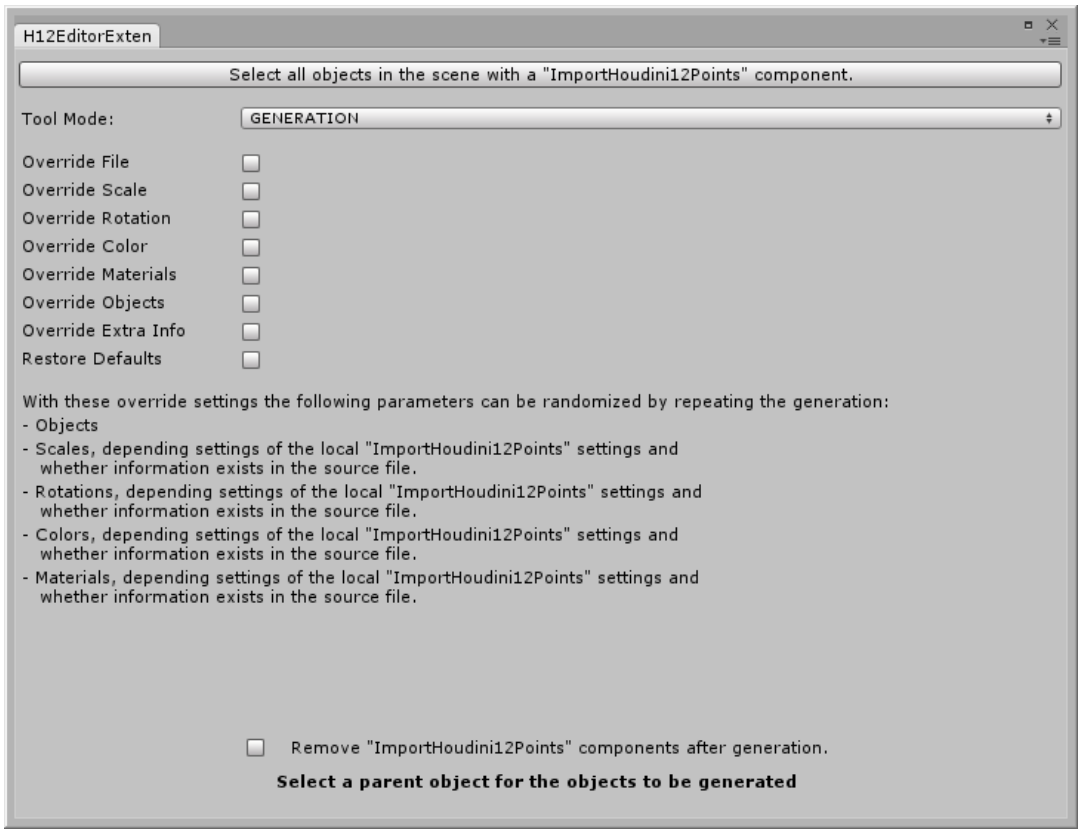
https://www.assetstore.unity3d.com/#/content/12316

USAGE IN UNITY EDITOR

Once the package is installed the tool can be accessed under GameObject/Create Other/Load From Houdini File.



This will open the following window:



SELECT ALL IMPORTHOUDINI12POINTS COMPONENTS

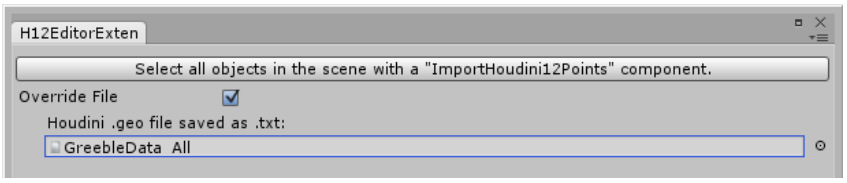
The H12EditorExtention will try add ImportHoudini12Points components to all selected objects, or use existing settings if selected objects already have such component. The top button can be used to select all objects in scene with an ImportHoudini12Points component.

TOOL MODE

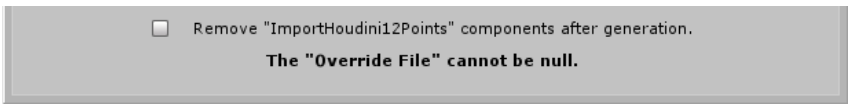
It is possible to use the window for the generation from a Houdini file in editor, or to snap existing objects to a path finding intersection. The generation mode is described first, for path finding mode see the end of this chapter.

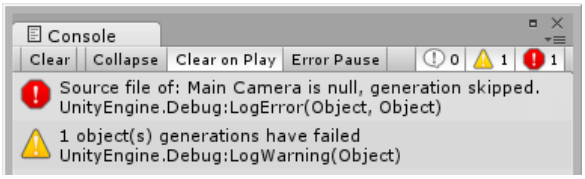
OVERRIDE FILE

Every ImportHoudini12Points, manually added in the editor, or added by the H12EditorExtention, requires at least a Houdini12 or later .txt file. Which in essence is a Houdini12 .geo file, but the Unity TextAsset object only accepts certain extensions.



The window will notify you if you forgot to add a valid Houdini file. It does so via the window itself, or via Debug.Warning depending on the situation.



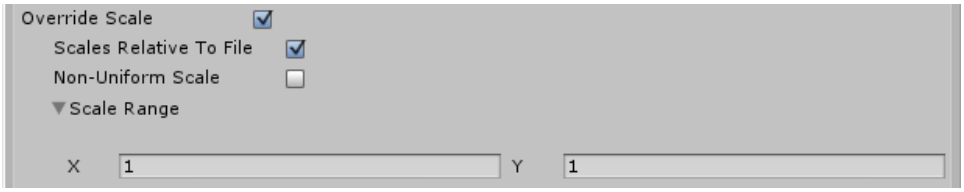


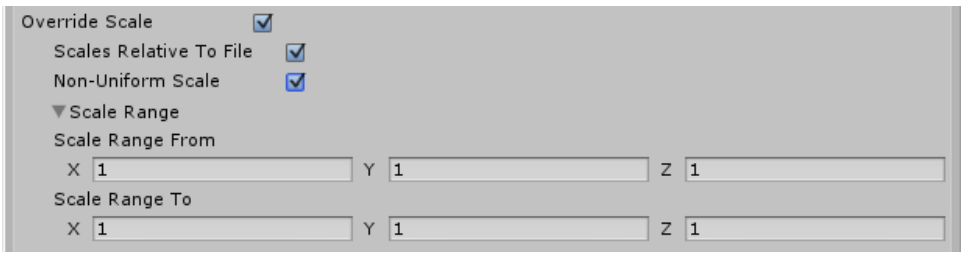
OVERRIDE SCALE

This the first of a series of variable overrides. The Houdini .txt file may contain scale data that can be used. With these settings this data can be overridden.

Scales can be relative to the data in the file. If this is selected, which it is by default, the random scale picked between the set ranges is multiplied with the scale data present in the file.

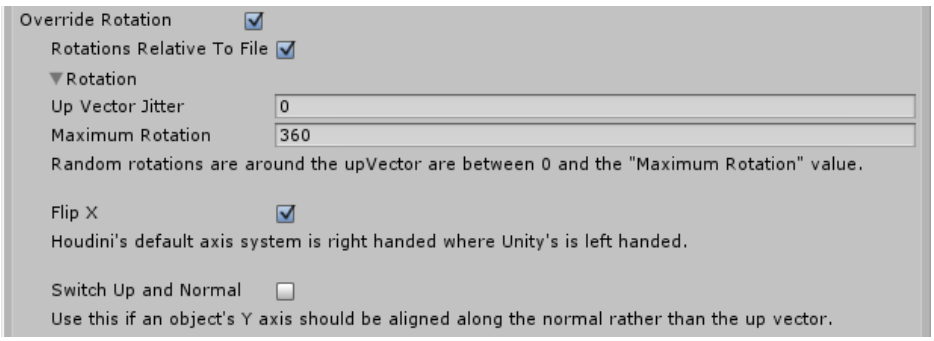
Scales can be uniform and non-uniform. Uniform scaling scales the generated objects a random value in the specified range, but uses the same value in all axes. With non-uniform scaling you can specify a range per axis.



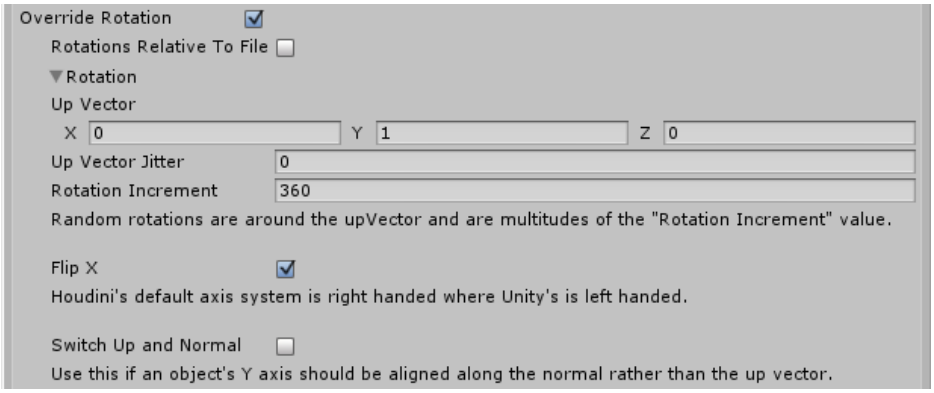


OVERRIDE ROTATION

Like the scale override the rotation override can also be relative to the data in the file. If this option is selected the rotation up vector or rotation axis is read from the file. The up vector can be jittered to randomize the pitch and jaw relative to the existing factor. After that the object is rotated around the up vector between 0 and (Maximum Rotation) degrees. If the Jitter and Maximum rotation are set to 0, the file's data will not be overridden.



If the rotation is not relative, an up vector must be specified. Also, instead of random rotation between two values, for this option the rotation is done in multitudes of this value, here called the "Rotation Increment" This way you can make sure object like buildings are always placed at 90 or 45 degree angles.



Flip X flips the rotation vectors and position across the X axis. This is done by default as Houdini uses a right handed and Unity a left handed coordinate system.

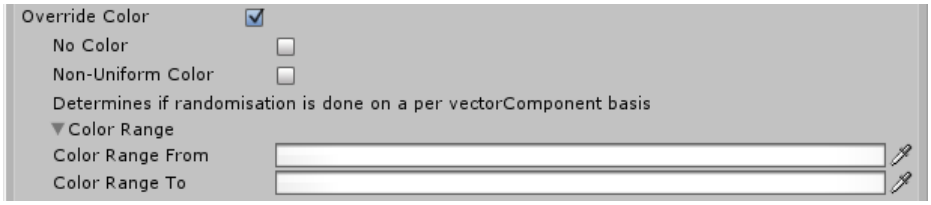
Switch Up and Normal allows swapping of the Up and Normal vectors. By default Houdini's copy node places object's forward axis (Z) along the normal and the object's up axis (Y) along the up vector. This can be switched using this toggle.

OVERRIDE COLOR

Override color overrides the material color the spawned objects or it can disable coloration if the Houdini file would otherwise introduce coloration.

Much like the Non-Uniform scaling option, the Non-Uniform Color option allows random values all for R,G and B axes of the color between the two specified colors. Uniform Color uses a Color.Lerp between the two specified colors.

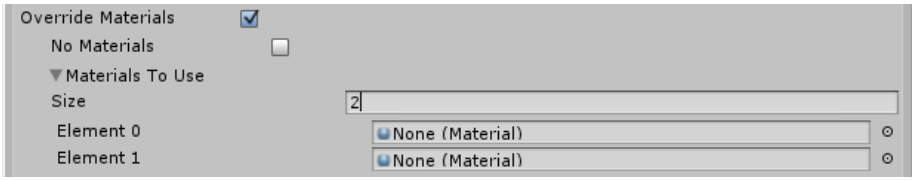
Note that unless specified in the ImportHoudini12Points materials are not allowed to leak into the scene. Instead, a script is added to colored objects that will create a material instance as soon as the application starts.



OVERRIDE MATERIALS

Override materials allows to override the prefab's default materials, note that these materials will be applied to all children, grand children etc. of each generated object. Much like the No color option, the No materials option can disable material modification if the Houdini file would otherwise modify the materials.

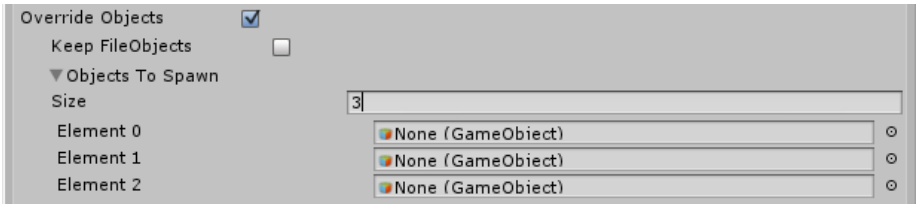
The generator will pick at random from the selected materials for each object. It is possible to specify a certain material for a specific object by adding it to the Houdini file's attributes.



OVERRIDE OBJECTS

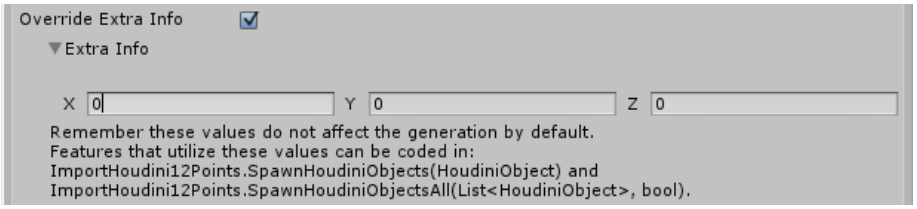
Override objects allows to choose which prefabs should be spawned. The generator will pick at random from the selected GameObjects for each position. It is possible to specify a certain GameObject for a specific position by adding it to the Houdini file's attributes.

Keep FileObjects will force the usage of the game objects when present in the file. This way the object selection will not be randomized each time the generator is re-run. This is useful if there are other values you do want to randomize.



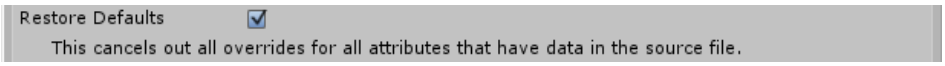
OVERRIDE EXTRA INFO

Overriding this data will not change the generation at first glance. However it is possible to implement features that use this data in the displayed functions. This can Extra Info can also be included in the Houdini file as a "extraData" attribute.



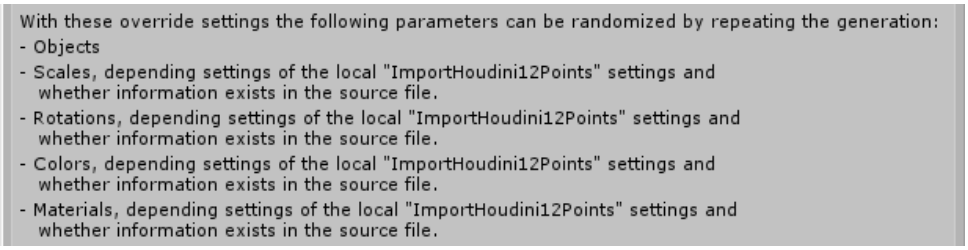
RESTORE DEFAULTS

Restoring defaults will restore all values to the attributes when present in the file. It will maintain overrides of which no data exists in the file.



RANDOMIZATION TIPS

When Restore Defaults and Remove "ImportHoudini12Points"components after generation are not checked, it will display a couple of reminders. These lines show which attributes will be randomized after re-running the generation.



REMOVE COMPONENTS AFTER GENERATION

By checking this box, the components will be removed after the generation, this way you can no longer review the values used, but it may slightly reduce memory usage in the scene.

If this option is not selected, the Components' Generation Setting is to NoGeneration. This way the generation is not repeated at runtime.

EXECUTING THE GENERATION IN EDITOR MODE

Once all overrides have been correctly set and the correct file is selected, either in the menu, or in manually added/edited ImportHoudini12Points components, you can generate the objects.

First make sure at least one object is selected:

If you have chosen to Override File, you must pick a file to use:

If these requirements are met, the generation may be executed:

If multiple objects are selected, the window will ask for conformation:

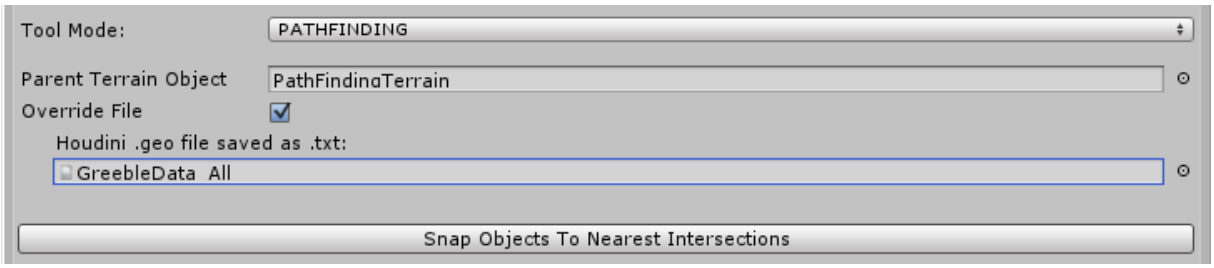
The generation will proceed. The generation can be near instantaneous till a few seconds. This mainly depends on the amount of components the prefabs to be generated have, especially collision and rigid body components may increase the duration of the generation. The duration of course will also depend on the amount of objects to be spawned.

Because of the way Unity handles physics, it is not advisable to instantiate too much objects with collision or rigid bodies under one single parent.

SNAP OBJECTS TO NEAREST INTERSECTIONS

When the tool mode is set to path finding it possible snap objects to their nearest path finding intersections. First a Parent Terrain Object must be selected. This object will receive an ImportHoudini12Points Component if it has not already. The Generation Setting will be set to PathFinding. Note that the Houdini file must contain neighbor* and preferably intersection* attributes for this to work. The Houdini file of the Parent Terrain can be overridden in the same way as with the generation mode.

When the parent has been set, select one or more objects in the editor and snap them by pressing the "Snap Objects To Nearest Intersections" button.



SETTING UP FOR USAGE AT RUNTIME IN UNITY

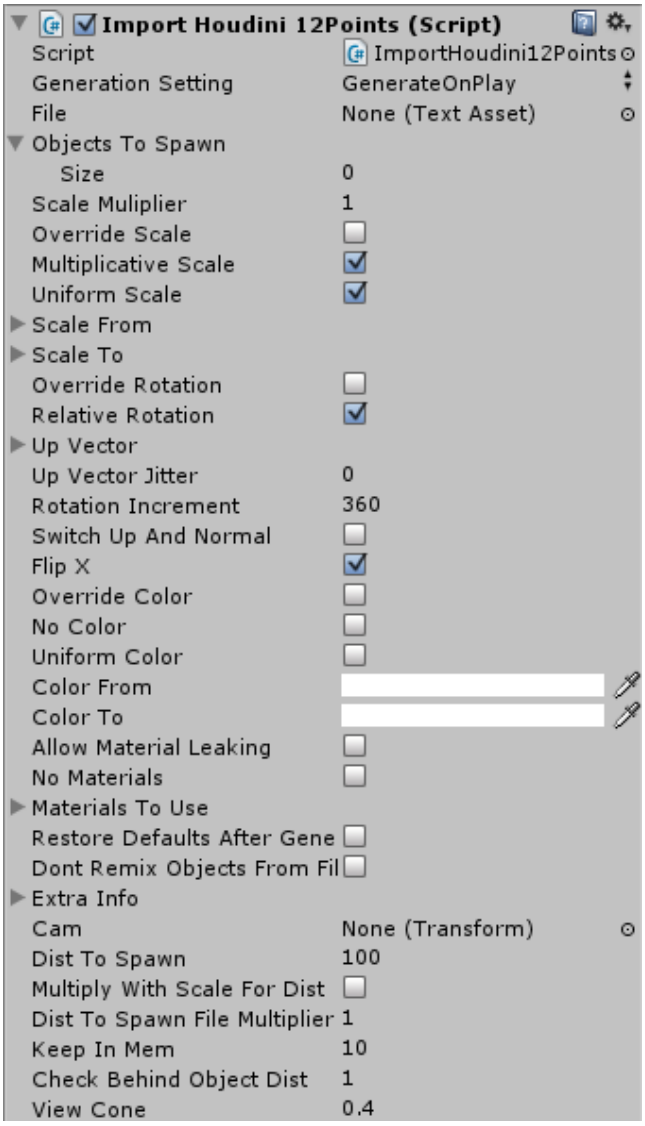
It is possible to use the ImportHoudini12Points component while bypassing the H12EditorExtention. This allows to either do the generation at the start of runtime or generate requested object depending on the distance to the camera and the direction of the camera.

The overrides set by the H12EditorExtention are saved in the ImportHoudini12Points components of the selected objects. Instead the overrides can be tweaked inside the components as well. Either by adding the component manually or adding it via the editor extension.

Also note that there are a couple of values and references that are not present in the editor extension. Most of these are associated with the DynamicGeneration option for the component.

THE IMPORTHOUDINI12POINTS COMPONENT

The ImportHoudini12Points component stores all the data set by the H12EditorExtension. For most of these values see the previous chapter. What follows is list of references not handled by the editor extension:



Generation Setting: Can be one of the following: GenerateOnPlay(default), NoGeneration, DynamicGeneration and PathFinding.

GenerateOnPlay will generate everything from the file at the start of runtime. With certain overriding settings, this may lead to a different generations each time.

NoGeneration will not parse or generate anything, the component is basically turned off, but allows for reviewing of all settings.

DynamicGeneration will parse the file at the start of runtime, but will only instantiate objects when they come into camera range and view, set in the Houdini file and/or values at the bottom of the component.

PathFinding will not generate any objects, just like NoGeneration, The file is however parsed to be usable for path finding at runtime.

Scale Multiplier: multiplies the scale of all objects generated, note that it already takes in account the scale of the parent object and of the prefab itself.

Allow Material Leaking: this allows material instance generation in editor mode. This option is not available in the editor extension as this is generally not desired.

Cam: Used for DynamicGeneration, it is the transform that is checked for distance and direction to each object. DynamicGeneration is skipped until a camera is assigned.

Dist To Spawn: Used for DynamicGeneration, distance in which objects are visible, if " Multiply With Scale For Dist" is enabled, this value is multiplied with the scale to calculate the final value.

Multiply With Scale For Dist: Used for DynamicGeneration, allows the usage of object scales to calculate generation distance

Dist To Spawn File Multiplier: Used for DynamicGeneration, A multiplier for generation distances as read by the Houdini file

Keep In Mem: Used for DynamicGeneration, amount of objects per type are kept in memory to be moved, instead of instantiated. Overwritten by the "renderData.y" attribute if it exists in the source file

Check Behind Object Dist: Used for DynamicGeneration, the distance that is checked behind the camera to make sure large objects are generated properly.

View Cone: Used for DynamicGeneration, the maximum dot product of the camera direction and direction to Object for an object to be generated.

AUXILIARY GENERATION SCRIPTS

The asset package allows dynamic instantiating of object depending on the camera. There are a couple of things to keep in mind when using this functionality, especially to prevent null reference exceptions. There are a couple of small monobehaviours that help with this:

DESTROYDYNAMICH12OBJECT

This behavior is automatically added to each object spawned using DynamicGeneration of ImportHoudini12Points. It allows the destruction and decoupling from the dynamic generation. Calling DestroyDynamicH12Object.DestroyDynamic() will correctly destroy an object handled by dynamic generation. It makes sure no null reference exceptions occurs when the object needs to be destroyed.

MAKENONDYNAMICH12OBJECT

This behavior should be added to prefabs generated with dynamic generation which should only be generated once, and should not be removed by the system once spawned, even when the object gets out of camera range or view. Examples of objects for which this may be useful are objects with non-kinematic rigid bodies or enemies.

H12EDITOREXTENSION

This is the editor script which houses the menu that allows in-editor generation.

HOUDINIOBJECT

This is a Struct, storing all the data needed for dynamic and non-dynamic generation from a Houdini12 .txt file.

HOUDINI12INSTANCESELFMATERIAL

This behavior is automatically added to each object and sub-object generated using the H12EditorExtention with a renderer. The component gets added when a material color modification is requested in editor mode. It sets up a material instance creation at the start of runtime for each object it has been added to.

THE HOUDINIPATHFINDINGAGENT COMPONENT

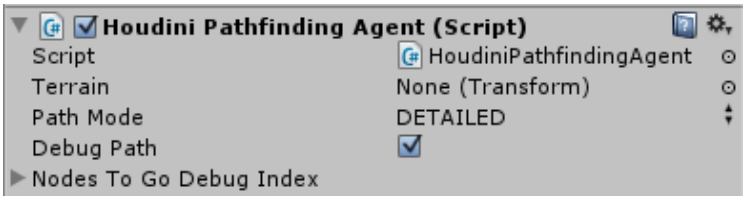
The HoudiniPathfindingAgent component stores paths generated by a linked ImportHoudini12Points component.

Terrain can be linked manually to each agent or can be set automatically, like by "HoudiniRTSMoveOrder" it is the object that is used to transform the imported path to the correct location and scale, this object should also hold the "ImportHoudini12Points" component that generates the path.

Path Mode can be set to FAST or DETAILED. The path mode FAST is the most basic mode. When the Houdini file has intersection* attributes, it will generate paths only connecting these intersections. When only neighbor* attributes are available it will use all points for path finding, which is slower. Path mode DETAILED creates a compromise, it will use the FAST method to find the intersections it needs to travel to, to reach the target, but will use the rest of the nodes to determine the path's between the intersections in a more efficient way. DETAILED mode requires neighbor* intersection* and pathArc* attributes. * Stands for a number wildcard so it possible to have neigbor0, neigbor1 and neigbor2 as separate attributes.

Debug Path, when enabled will display Debug.line()s in the editor along the calculated paths. Note that in DETAILED mode, when using debug path, it effectively creates the path twice due to the way the path's are processed.

Nodes To Go Debug Index, will show the index numbers of the nodes along the current path. These numbers correspond to the point numbers in de Houdini file.



PATH FINDING EXAMPLE SCRIPTS

The package includes a couple of example scripts of how the path finding capabilities could be used. They are showcased in the PathfinderTest scene.

AGENTMOVER

This script moves the rigid body of an object in the direction supplied by the HoudiniPathfindingAgent, which is required on the same object to work.

BLOCKPATHBYBOUNDS

This behavior allows the use of box shapes in the editor to toggle certain path nodes of a path generator. It could be used to create gates as shown in the demo.

HOUDINIRTSMOVEORDER

When attached to a camera and linked with a path generating terrain object and one or more agents, this class allows for the issuing of move orders to the assigned agents. Note that some sort of collision is required to calculate the 3d positions of mouse clicks.

AUXILIARY DEMO SCRIPTS

A couple of scripts have been added to demonstrate the workings of the asset package. They do however not take place in the generation process itself:

BALLIMPACT

Attached to the Projectiles fired by the FireBall script. When the rigid body of the object to which it is attached hits something, within a certain radius all objects generated using DynamicGeneration are removed. It also spawns a particle effect each time it hits.

DELETE HITTERS

Attached to the GroundKillPlane of H12PntImporterDemo. It removes all rigid bodies that hit it, this way projectiles are removed when they fall off the ground plane.

FIREBALL

Attached to the Main Camera of H12PntImporterDemo, it fires a projectile when the left mouse button is released in the direction the mouse at that time. The projectile fires faster when the button is pressed longer up to one second.

FRAMERATECOUNTER

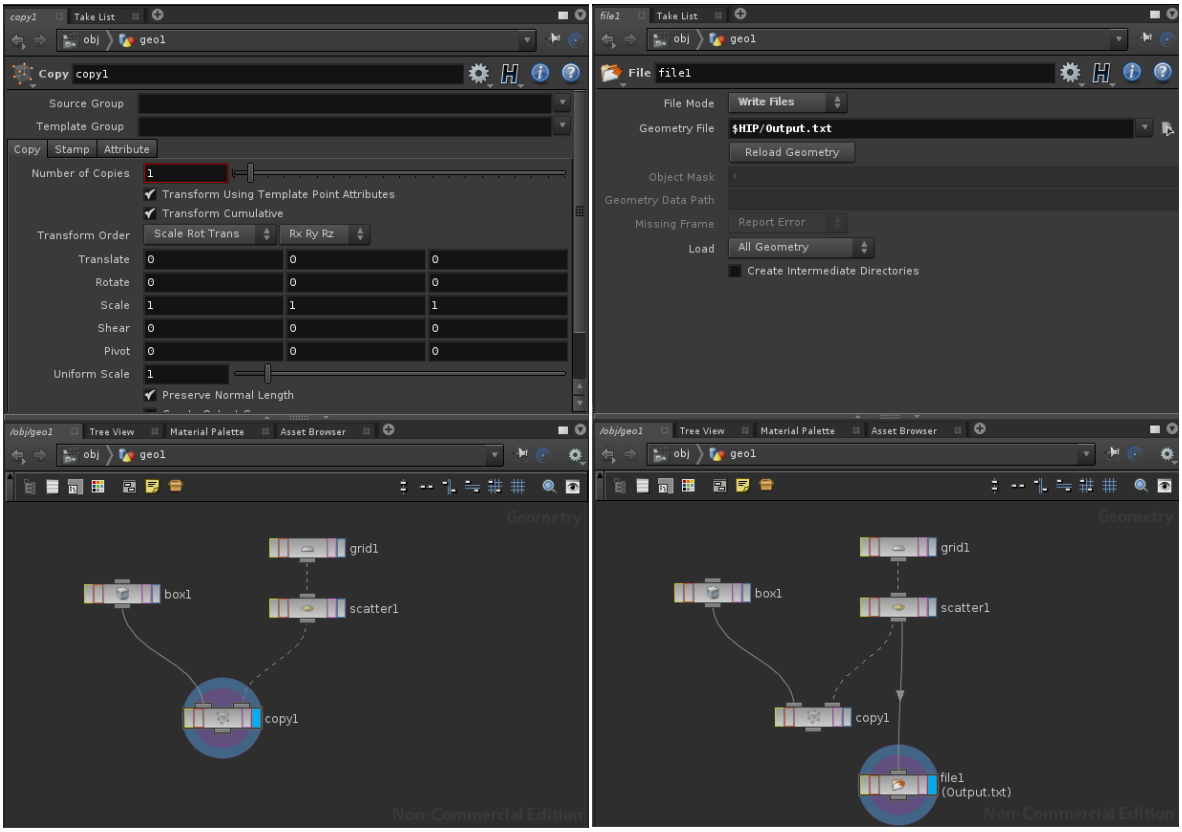
This behavior, when attached to a GUI Text, will display the current FPS as well as the lowest FPS after the first 30 frames.

ROTATEOBJECTOVERTIME

Attached to the Teapot of H12PntImporterDemo, it rotates the Teapot 10 degrees along the Y-axis each second. Note that the teapot has been placed at the generation range compared to the camera. This way only object that are on the side facing the camera are generated.

EXPORTING FROM HOUDINI

The purpose of this package is to import point data from Houdini. For those familiar with Houdini's procedural modeling, the entire parser acts like a Copy SOP does in Houdini, with stamping functionality.



The Houdini Copy node has two inputs, the left input has the objects that should be spawned, the right input the points at which to spawn these objects. This package allows importing the data you would normally insert into the right input of a Houdini Copy SOP (Surface Operator). Instead this information can be saved using a File SOP. In the example above this will only set point positions for the exported positions. Note that copy1 node gives exactly the same result as the file1 node's data being imported into Unity, when a cube is set as the object to spawn.

ATTRIBUTES

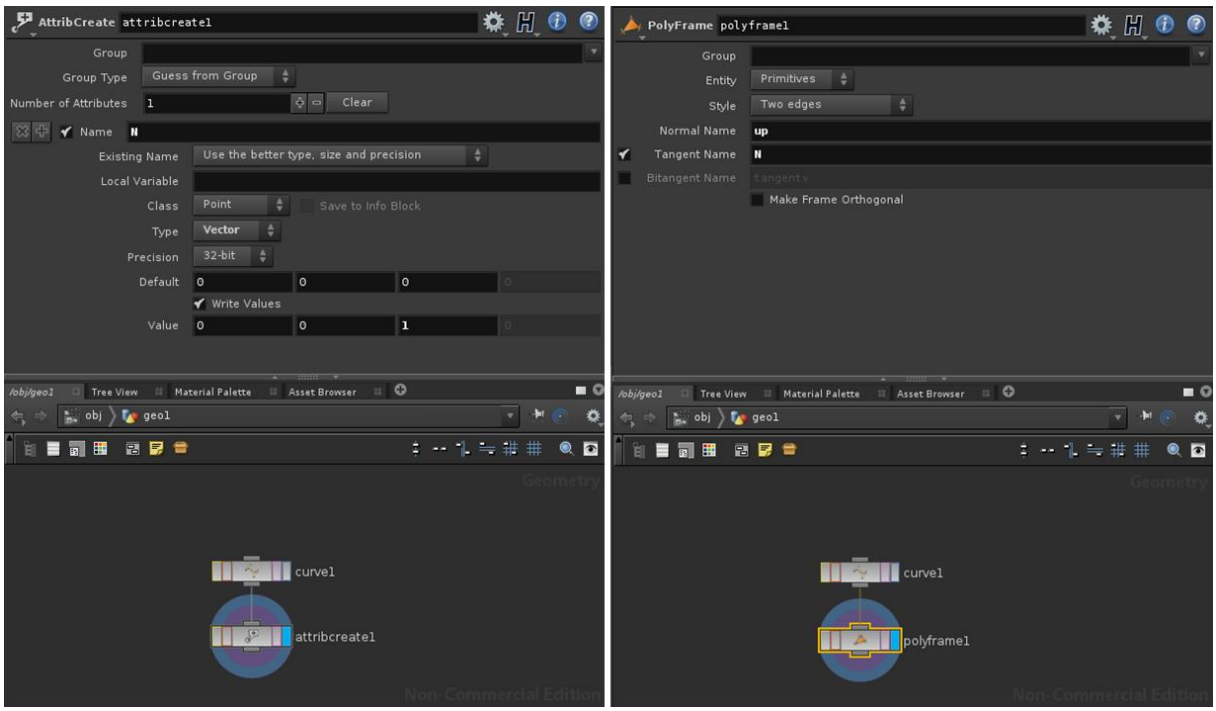
The importer features basic stamping functionality in Unity, this means storing additional information on the right input side of the copy node, for the left input side to use. Data that importer can read by default:

Information	Attribute name	Houdini Data type
Position	P	Vector4
Orientation of forward axis	N	Vector3
Orientation of up axis	up	Vector3
Scale	scale	Float or Vector3
Color	Cd	Float or Vector3
Object to Spawn	name	String
Material to use	material	String
Generation Distance	renderData	Vector3
Extra Data	extraData	Float or Vector3
Pathfinding Group	class	int
Neigboring Points	neigbor*	int
Neigboring Pntersections	intersection*	int
Pathlength to Neigboring Intersections	pathArc*	float

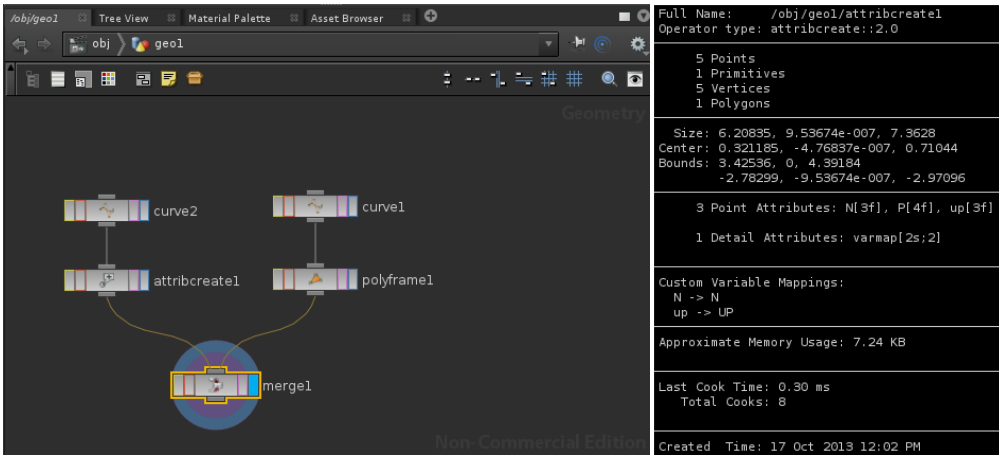
Note that these attribute names are case sensitive and those familiar with Houdini will see the standard Houdini attribute names have been used where possible.

Also note that for the bottom three path finding attributes, there can be multiple attributes, for example neigbor0, neigbor1 and neigbor2. It is required that these 3 have the same number of attributes with a -1 where no data exists, for instance if certain points have more neighbors than others.

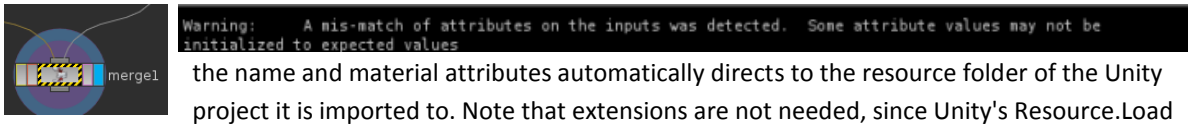
To add these attributes there are various ways. It possible to use AttribCreate SOPs to create set values for instance, but it is also possible to use the vast array of other SOPs, including VOP SOPs (VEX Surface Operators) to calculate the desired values. All attributes should be point attributes, either by creating them as such or by promoting them to point attributes using a AttribPromote SOP.



It also possible to merge groups of points together, each with their attributes. Remember to make sure that each group has the same attributes, otherwise Houdini will fill in the missing values with their default values. This can lead to unexpected behavior as N and up vectors should not have a (0,0,0) value. To check if both inputs of a merge node have the same attributes, you can use the middle mouse button on the connected node.



If inputs have mismatched attributes the merge node will show a yellow/black warning border. Using the middle mouse button on the merge node will display this error at the bottom of the window.



the name and material attributes automatically directs to the resource folder of the Unity project it is imported to. Note that extensions are not needed, since Unity's Resource.Load also does not require this. Extensions are stripped from the name attribute when the file is parsed.

If the name attribute is "house" for a certain point, it will look in the root of the resource folder for a GameObject named "house". If the name attribute is "houses/house1" Unity will look inside Resources/houses for a GameObject named "house1".

The material attribute handles exactly the same, with as a difference that it looks for Materials rather than GameObjects.

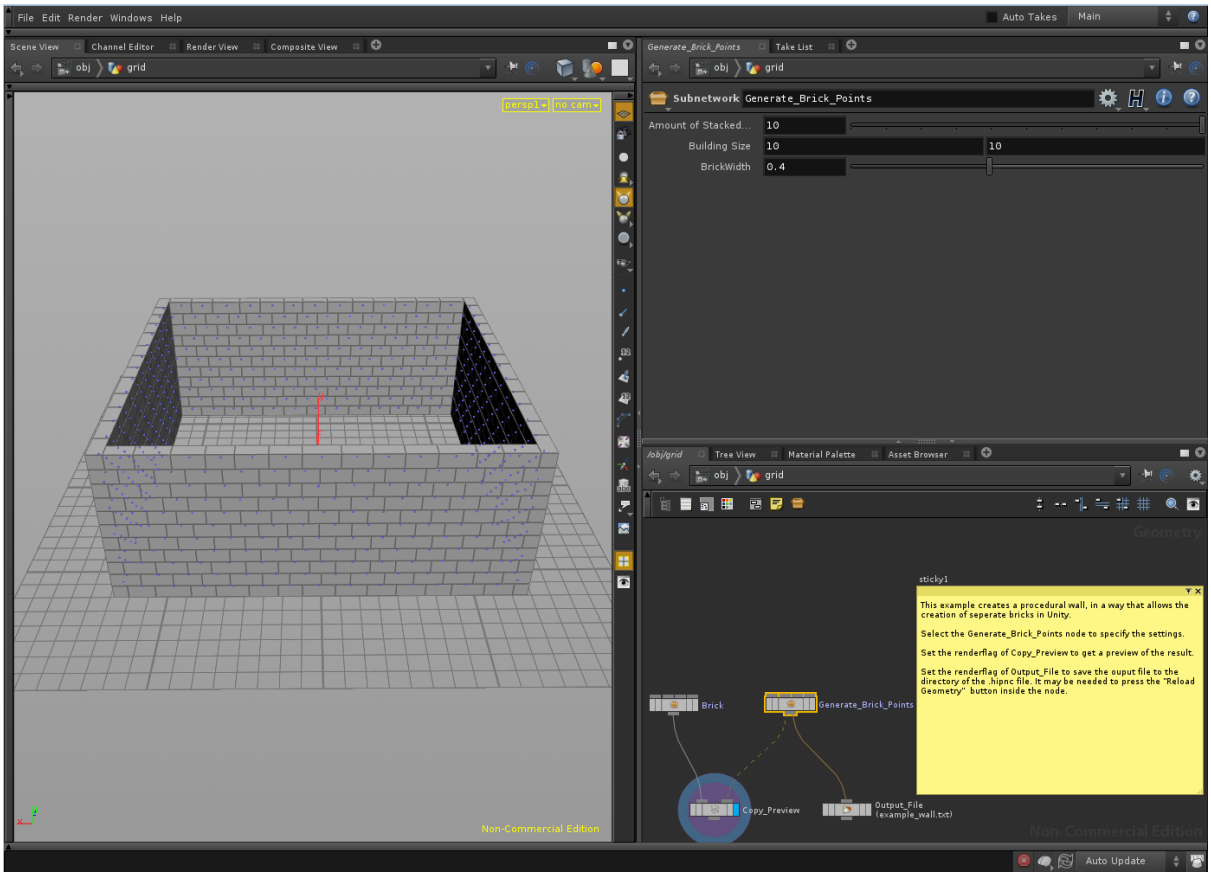
If the file parser cannot find a certain object from the name or material attribute, Unity will throw a Debug.Warning, showing which object it could not find, with the accompanying path.

HOUDINI EXAMPLE FILES

The package includes a couple of Houdini examples. These examples give an idea of how to best utilize the Importer. These examples have been created using Houdini 12.5.

WALLEXAMPLE

The Wall example, creates data for a rectangle building walls. The bricks are created independently and all slightly differ in size.



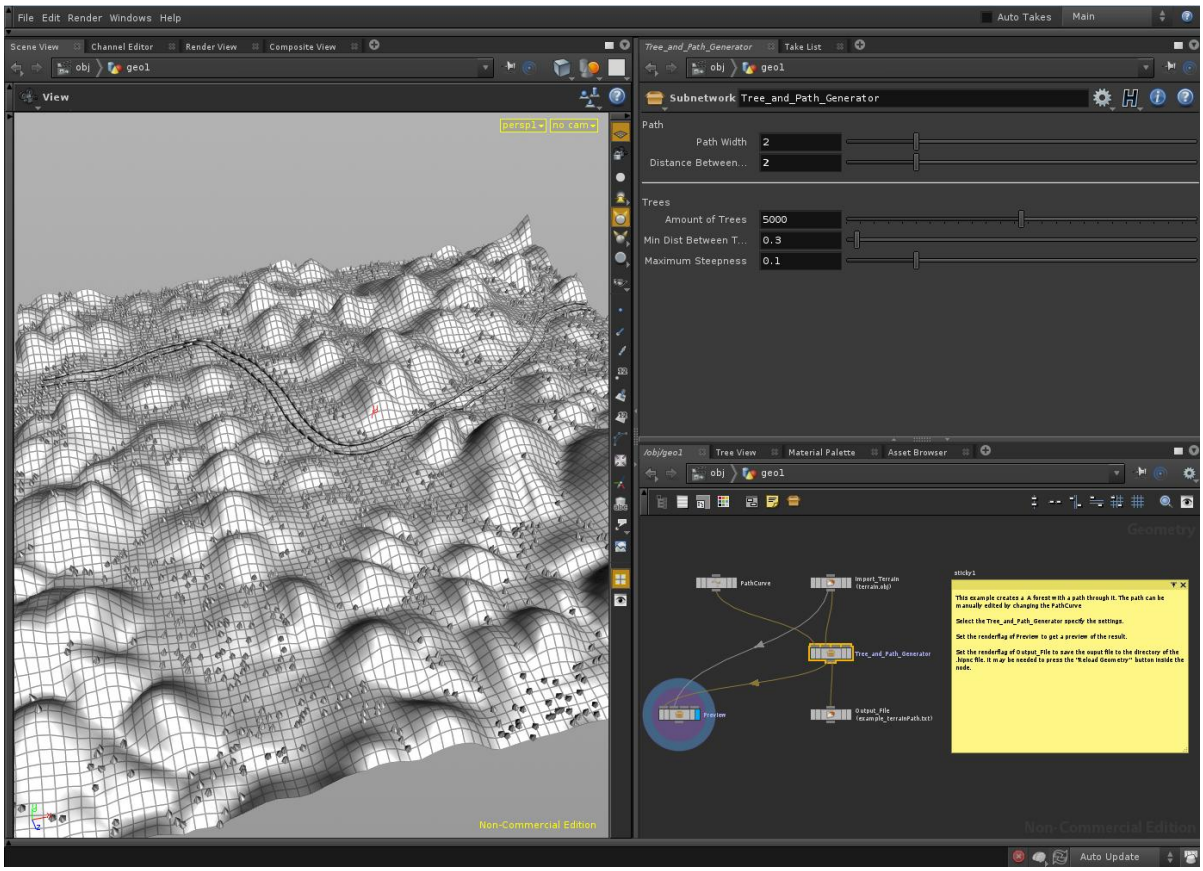
The example uses the following attributes: P, N, up and scale. When imported a 1by1by1 cube should be used as an object so it can be scaled according to the example.

Alternatively any 1by1by1 one object can be used, or various different ones to create a more varied wall.

```
Full Name:     /obj/grid/Generate_Brick_Points
Operator type: subnet

   572 Points
     0 Primitives
     0 Vertices

     4 Point Groups:
0 points in culledGrp
0 points in origrp
0 points in randomGrp
0 points in semiEdgePnt

  Size: 10.4, 4, 9.6
Center: 0, 2.2, 0
Bounds: 5.2, 4.2, 4.8
        -5.2, 0.2, -4.8

     4 Point Attributes: N[3f], P[4f], scale[3f], up[3f]

     1 Detail Attributes: varmap[2s;2]

Custom Variable Mappings:
  up -> UP
  scale -> SCALE

Approximate Memory Usage: 0.00 KB
Sub-Network output from: Generate_Brick_Points/up_vector

Last Cook Time: 0.04 ms
   Total Cooks: 4

Created  Time: 17 Oct 2013 01:12 PM
Modified Time: 17 Oct 2013 01:14 PM
Time Dependent: No
```

TERRAINEXAMPLE

The Forest Path example creates a forest with a path through it on a pre-generated terrain. The path can be manually edited. Other settings include how the forest placement reacts to the terrain and the width of the path.

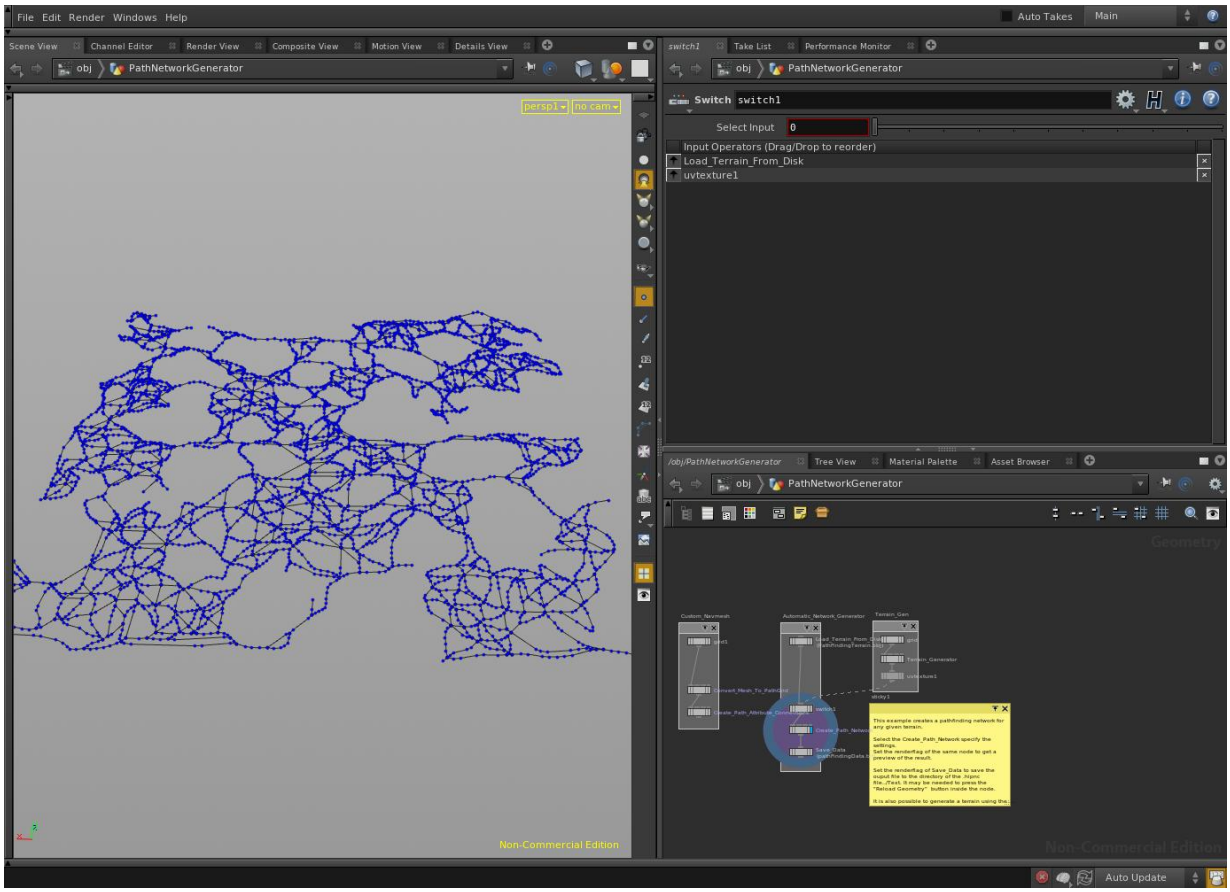


The example uses the following attributes: P, N, up, scale and name. When imported it will load the Fence and Tree models, from the Resources/Terrain folder inside the package.

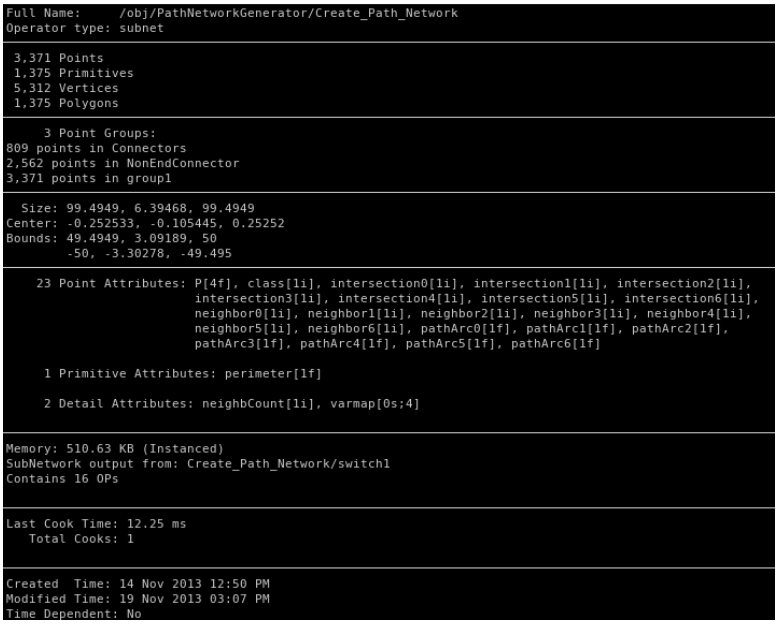
```
Full Name:     /obj/geo1/Tree_and_Path_Generator
Operator type: subnet

 1,824 Points
     0 Primitives
     0 Vertices

  Size: 99.987, 7.16608, 99.9482
Center: -0.00258255, 3.58508, -0.00948524
Bounds: 49.9909, 7.16811, 49.9646
        -49.9961, 0.00203688, -49.9836

     5 Point Attributes: N[3f], P[4f], name[1s;2], scale[3f], up[3f]

Approximate Memory Usage: 0.00 KB
Sub-Network output from: Tree_and_Path_Generator/merge1

Last Cook Time: 0.05 ms
   Total Cooks: 1

Created  Time: 17 Oct 2013 04:00 PM
Modified Time: 17 Oct 2013 08:35 PM
Time Dependent: No
```

PATHFINDINGEXAMPLE_H13

This example needs Houdini 13 or higher. It will automatically create a navigation mesh from a terrain input that can be imported by the package. It also contains a simple terrain generator and some nodes to convert a custom generated navigation mesh into something that can be used by the importer.



This example uses the following attributes: P, class, intersection0-6, neigbor0-6 and pathArc0-6. These attributes are specific to the pathfinder that is included in the package.



PACKAGE CONTENT OVERVIEW

Apart from the scripts, this package also contains a number of meshes, prefabs, materials etc. These can be used are used to demonstrate the package and are not need to be included in your project.

H12IMPORTER

- This Document

H12IMPORTER/EDITOR

- H12EditorExtension, allows using the H12Importer in editor mode.

H12IMPORTER/HOUDINI EXAMPLES

- PathfindingExample_H13, generates the path in the H13PathfinderDemo scene.
- TerrainExample, Shown in the background of the H12PntImporterDemo scene.
- WallExample, Shown in the H12PntImporterDemo scene, implemented with interactivity.

H12IMPORTER/MATERIALS

- BlockerDebug, shows the bounds overlay for the gate in the H13PathfinderDemo scene.
- Gate Diffuse, gives the gate in the pathfinder demo more contrast.
- ImpactFX, material for projectile particle effect.
- Light Diffuse, material used for various bright objects in the demo.
- Standard Diffuse, material used for darker grey objects in the demo.

H12IMPORTER/MESHES

- GateArc, shown in the H13PathfinderDemo scene.
- PanelTeapot, Shown in the H12PntImporterDemo, rotates around to demonstrate dynamic generation.
- PathFindingTerrain, used in the H13PathfinderDemo scene.
- TeapotCollision, Lower Resolution collision mesh for the PanelTeapot.
- Terrain, Shown in the background of the H12PntImporterDemo.
- TestArrow, Not used, can be chosen as an Object To Spawn using the ImportHoudini12Points script.
- Materials, Imported materials.

H12IMPORTER/PHYSIC MATERIALS

- BouncySphere, Used for the projectile in the demo.
- ConstructionBricks, Used for the wall object in the demo.

H12IMPORTER/PREFABS

- Arrow, Not Used, can be chosen as an Object To Spawn using the ImportHoudini12Points script.
- BallImpact, Particle effect shown on Projectile impact.
- Cube, Used for the bricks in the wall object.
- HoudiniAgent, used in the H13PathfinderDemo scene, moves along the generated paths
- Projectile, Used to Destroy the wall object and the greebles on the teapot in the demo.
- TestArrowCollision Not Used, can be chosen as an Object To Spawn using the ImportHoudini12Points script.

H12IMPORTER/SCENES

- H12PntImporterDemo, Shows the functionality of all the generation scripts in the package.
- H13PathfinderDemo, Shows the functionality of all the path finding scripts in the package.

H12IMPORTER/SCRIPTS

For descriptions see the concerned chapters.

- Demo Scripts
    - BallImpact
    - DeleteHitters
    - FireBall
    - RotateObjectOverTime
- DestroyDynamicH12Object
- Houdini12InstanceSelfMaterial
- HoudiniObject
- HoudiniPathfindingAgent
- ImportHoudini12Points
- MakeNonDynamicH12Object
- Pathfinding Example Scripts
    - AgentMover
    - BlockPathByBounds
    - HoudiniRTSMover

H12IMPORTER/TEXT

These are example Houdini12 .txt files that can be used with the importer

- example_terrainPath, generated by the supplied example Houdini project file. Generates the background in the demo.
- example_wall, generated by the supplied example Houdini project file.
- example_wallLarge, larger version of the above. Generates the wall in the demo.
- GreebleData_All, Example File using all available attributes.
- GreebleData_All+extra attribs-renderData, file used on the teapot in the demo.
- GreebleData_obj_scale_orient, Example File , using 3 of the available attributes.
- GreebleData_obj_scaleFloat_orient, Example File, using 3 of the available attributes. With scale as a float.
- GreebleData_positionOnly, Example File using 1 of the available attributes.
- GreebleData_scale_orient, Example File using 2 of the available attributes.

RESOURCES/GREEBLE

- 65 Greeble Prefabs, used by the teapot in the demo:
- Greeble00-Greeble64

RESOURCES/GREEBLEOBJ

- 65 Greeble Meshes, used by the teapot in the demo:
- Greeble00-Greeble64

RESOURCES/MATERIALS

- 5 Test Materials, used by the teapot in the demo:
- Test0-Test4

RESOURCES/TERRAIN

- Fence, mesh for the path in the background of the demo.
- Tree, mesh for the trees in the background of the demo.
- Materials, Imported materials.

REFERENCES

- Side Effects, Creators of Houdini, If you do not yet have Houdini, you can get a free version of Houdini 12 at their website: http://www.sidefx.com/
- Oddforce, if you have problems with Houdini, you can ask all questions on this forum: http://odforce.net/
- Unity Technologies, the creators of the Unity Engine, if you do not yet have Unity, you can get a free version of Unity at their website: http://unity3d.com/
- if you have specific questions about the importer itself, feel free to send me an email: TwandeGraafTechArt@gmail.com

SPECIAL THANKS TO

- Achraf Cherabi, for testing the package.
- Robin Marx, for teaching me the first steps of C# and during my Internship at LuGus studios we created the first steps that would eventually become this package.