InfiniTree: Procedural Tree generation, growth, dynamics & management system

This is a pack devoted to procedural tree and plant generation and the first major module of the **INfiniDy Suit**. The goal is to offer an as optimized and versatile as possible system to create unique and controllable trees. The new version 1.6 brings a lot of performance enhancements, more control over tree growth and new options that allow the system to be used for grass and decals besides trees and forests.

Instructions

In order to use the **INfiniDy Tree** system, the user must add the "**INfiniDyForest"** script to a gameobject. This system is also available as **prefab that can be instantiated in real time**.

Basic setup

The "**INfiniDyForest"** script will generate a tree in its gameobject position. The "Create forest" option will add the tree to the forest pool gameobject, if it is not selected the object will be self batched in a newly created gameobject.

Options:

Max Health (v1.6): Maximum health of the tree, reduced at each axe hit.

Chop (v1.6): New chop tree behavior, enable to cut down trees with the axe. This replaces the standard interactive tree behavior (leaf and branch motion). The tree must be set to "Interactive" for this option to have an effect.

Max Fall amount (v1.6): The total amount to fall.

No shadows on dynamic (v1.6): Remove shadows when the tree enters the dynamic phase. Use to decrease draw calls during dynamics.

Remove shadows above level (v1.6): Remove shadows above the chosen growth level.

No leaf shadows (v1.6): Remove leaf shadows on dynamic phase.

Remove receive - cast (v1.6): Remove receive and/or cast shadows for the above options.

Range leaf scale (v1.6): Range of leaf scaling, use for varying leaf size.

Leaf scale range (v1.6): Low and high leaf scale for the above option.

Fall max time (v1.6): Time to fall.

Fall speed (v1.6): Speed of tree fall after chop.

Rotate towards normal (v1.6): Rotate tree based on the "direction normal" vector.

Rotation speed (v1.6): Vary adjustment to normal speed.

Direction normal (v1.6): Normal vector to rotate tree towards.

Max rotation time (v1.6): Maximum time to rotate tree towards surface normal on growth.

Hero: Assign the hero or camera that will move in relation to the trees. The tag "Hero" can also be used in order to get the instantiated hero during gameplay.

Interaction threshold: The distance at which the interactive trees will react to the hero. At this point the trees are de-combined and draw calls will increase locally depending on the type of the trees in the group.

Interaction offset: Use this offset to smooth the transition between combined and de-combined state. Must be above zero to function properly.

Enable LOD: Enable the LOD system for the tree and its group. Enabled by default.

Grow in editor: An option to grow the tree in editor and keep the scripted nature in play mode. Make sure the tree is running properly in play mode before use. Check the option to create the tree in editor. **NOTE:** Break prefab instance before entering this mode and make sure prefab does not get updated after growth. Also this option can be checked only once, so create a copy of the system before applying.

Insta-grow leaves: Grow leaves instantly at start.

Leaves local space: When checked, leaves will be parented to their branches, otherwise will be parented to the "leaf pool" object.

Is grass: Use the system for grass. This option will parent the branches to one holder, spread them around the tree and regulate their positioning based on the terrain height.

Start radius: The starting radius of branches and trunk.

Start length: The starting length of branches and trunk.

Length Scale: Scaling for the branches/bark placement.

Extend Scale: Scale of the branch/bark prefab gameobject.

Min-Max Spread: Spread of branches angle.

Spread Y-Z separate: Define spread angles individually for each axis.

Min-Max spread Y-Z: The angle range to spread branches in the Z-Y axis.

Min-Max Branching: Amount of child branches.

Max branches per level (v1.6): Define the maximum amount of grown branches allowed per level of growth.

Min-Max Radius: Min and max of branch radius.

Lower length by: The percent to lower the successive child branch length.

Lower radius by: The percent to lower the successive child branch radius.

Grow tree: Enable the gradual growth of the tree.

Start tree scale: The starting scale of the tree, when growing option is active.

End scale: The desired final scale to reach.

Grow tree speed: The speed at which the tree will grow towards its target global scale.

Update every: Time between two updates. Use lower values for smoother results, higher for performance gain.

Bark prefab: The prefab name to be used as bark. It must be placed in a folder called "Resources".

Bark prefab Object (v1.6): The prefab to be used as bark. Directly assign a prefab from the project. The "Bark Prefab" string parameter must be left empty for this to work.

Leaf prefabs: The prefab names to be used as leaves. They must be placed in a folder called "Resources".

Branch prefabs: The prefab names to be used as branches. They must be placed in a folder called "Resources".

Leaf prefabs Object (v1.6): The prefabs to be used as leaves. Directly assign a prefab from the project. The "Leaf Prefabs" string parameters list must be left empty (set size to zero) for this to work.

Branch prefabs Object (v1.6): The prefabs to be used as branches. Directly assign a prefab from the project. The "Branch Prefabs" string parameters list must be left empty (set size to zero) for this to work.

Alternate leaves (REMOVED in v.1.6): If checked, the system will randomly choose from the leaves list, otherwise will use only the first prefab in the list. **(REMOVED in v.1.6 – leaves will alternate if the leaf prefabs list size is bigger than one, automatically)**

Branch prefab per level (REMOVED in v.1.6): Use a different prefab for branches, based on the branch level. Useful for LOD implementation in only certain branches. **(REMOVED in v.1.6 – branches will alternate if the branch prefabs list size is bigger than one, automatically)**

Branch ID per level: The ID of the branch prefab in the prefabs list to use for the equivalent to this list level. For example use 0 in the 3rd item of the list, in order to use the first prefab in the branch prefab list for the 3rd level of branch growth.

Grow leaves slow factor: The larger the value, the slower the leaves will grow.

Grow speed: The speed of individual leaf and branch growth.

Grow start division: Starting scale at branch prebaf instantiation, the scaling will continue from that starting scale.

Growth angle division: Starting angle at branch prebaf instantiation, the rotation will continue from that starting scale.

Leaf start division: Starting scale at leaf prebaf instantiation, the scaling will continue from that starting scale.

Smooth leaf growth: Make tree leaves grow gradually.

Leaf level min-max: Grow leaves between these branch growth levels.

Leaves per branch: Number of leaves to assign to each branch.

Use height: Use height when assigning leaves to the branches.

Leaf height min-max: Grow leaves only between these height levels.

Leaf distance factor (v1.6): Control the distance between leaf placement.

Create forest: **Parents newly created trees to the gameobject with tag "INfiniDyForest"** or creates it if it does not exist. This will batch all trees under one script and will drastically lower draw calls. It is recommended to always use this option. If not checked the batching will be applied per individual tree.

Interactive tree: Make this tree interactive, when hero comes close. This type of tree will be parented in the lower number tree groups, to make the de-combination of trees smoother and draw calls lower while de-combined.

Max tree distance: The max distance between the closest tree group and the new tree at which the tree will be included in the group. If no group is close, a new one will be created.

Grow angles: Make branches rotate towards their final positions from a lower angle.

Leaf pool: The gameobject to hold the leaves. If not defined as a scene object, one will be created.

Height levels: The times to repeat the initial branching on top of the initial one.

Height separation: The separation of the height segments. The layers above the first layers of branches will be created at this interval from each other.

Height reduce: Reduction of scale on each consecutive tree height layer as height increases. Use for pine trees.

Level 1 Decline: Degrees to turn each of the main bark segments when more than one height levels are defined.

Reduce angle with height: Reduce branch spread angle as height increases.

Multithreaded: Use an optimized multithreaded version for the batching.

Height offset: Offset the extra segments created when more than one height is defined. Use to adjust placement of height segments based on the prefab shape.

Decouple from bark: Avoid parenting of new height segments to the previous segment. If checked all height segments will be parented to the same starting point (bark).

Forest Holder: The gameobject to hold the forest trees. If not defined as a scene object, one will be created.

Leaf Material: The material used for the leaves. Define in order to enable wind in leaves. **The material must have a "_Scale" parameter.**

Enable Wind: Wind effect is applied the tree leaves.

Wind frequency: Frequency of the wind motion.

Wind strength: The amplitude of the wind.

Batch ended (v1.6): Signal the end of batching procedure for the tree. **(debug)**

Is moving: Flag to show the dynamic tree is active. **(debug)**

Health (v1.6): Current tree health **(debug)**

Start falling (v1.6): Flag tree fall after it has been chopped. **(debug)**

Fall amount (v1.6): The current fall amount. **(debug)**

Fall ended - Passed chop (v1.6): Flags for chop stages. **(debug)**

Root tree (v1.6): Holds the created tree, which is added to the batch pool, for quick reference on click. **(debug)**

Grow tree ended: Flag to show tree growth has ended **(debug)**.

Branch Grew (v1.6): Flag grown branches **(debug).**

Grow level: For debug purposes, the plant growth stage. **(debug)**

Growth over (v1.6): Flag showing the completion of tree growth **(debug)**

Combiner (v1.6): Reference to the script that takes care of the automatic batching **(debug)**

NOTE: The system demo uses prefabs from Unity tree creator, which is affected by windzones. If a windzone affects the tree material, the batching system may introduce a radical change to the combined mesh. The effect is instant and can be seen as a shape change to the tree after the batching is done. Disable windzone for the prefabs or use prefabs that are not connected to Unity tree creator if a windzone must exist in the scene. Another solution is a shader that is not affected by windzones.

NOTE: The debug parameters can also be used to synchronize external scripts with the system.

Batching (static and dynamic) for trees.

The batching system is applied to the generated trees by default. The script allows for static batching and also dynamic batching can be used when the system has the “Self Dynamic Enable” option checked.

Options:

Save name: The name of the tree when a prefab is created.

Save directory: The directory to save the tree meshes and prefab.

Added item count: The tree items added to this combiner.

Added items: The transforms of the trees added to the combiner.

Added items handles: The scripts that control each tree added to the above list.

Max items: The maximum number of trees that can be assigned to this combiner. This number will be defined by the tree script, depending on its type (static of interactive).

Deactivate Hero distance: Distance at which the LOD system and mesh group will be enabled. This parameter must be set in code in order to apply to all items.

Deactivate Hero offset: A small offset distance to secure a smooth transition in the boundary of the above transition. This parameter must be set in code in order to apply to all items.

Deactivate Hero distance cutoff (v1.6): Distance at which the LOD system will make items disappear. This parameter must be set in code in order to apply to all items.

Deactivate LOD: Enable the LOD system for this combiner. Enabled by default.

LOD Material (v1.6): NOT USED. Reserved for future updates.

Generate triangle strip: Option for mesh combination.

Auto disable: Disable automatically after enabled. Use when an external script needs to activate and not have to check if the script got disabled. Used in Dynamic Batching by Propagator script to control the use of the script, so it can be applied only when objects have moved to maximize performance.

Skip every N frame: Performance control.

Multithreaded: Use the multithreaded system for extra performance. Enabled by default.

MT Method 2: Enable this option to completely de-combined the meshes before recombination, in multithreaded mode. It is recommended leave this option **unchecked** for performance gain. Disabled by default.

Is moving: Flag to show the system is in interactive mode (externally controlled by the tree script) **(debug)**

Doing batching: Flag used to synchronize the system's batching procedure with the tree scripts **(debug)**

Make Active: Set script as active. The script will combine meshes to a single mesh at every update and recombine.

Self Dynamic Enable: Allow the script to be used to any gameobject pool for dynamic batching. The script checks for position changes to activate the re-batching. The trees must not be in "Interactive" mode and this mode will not work in multithreaded mode.

Self Dynamic Check Rotation: Dynamically re-batch items when rotation is sensed.

Self Dynamic Check Scale: Dynamically re-batch items when scale is sensed.

Is active: Dynamically re-batch items when scale is sensed.

Hero: The hero (with the camera), use the "Hero" tag to grab during gameplay.

Min distance: Min distance from original position at which dynamic bathing will re-batch objects. Use to increase performance and avoid re-batching when minor motion takes place.

Stop Self Dynamic After: Stop dynamic batching after the specific number of seconds. Use to allow objects to settle before they are combined.

Decombine: Restore objects to their initial state before batching.

Decombined: Flag to show the meshes have been de-combined and are ready to be interacted with **(debug)**

Erase tree: Check to erase the trees defined in the Erase Tree IDs list.

Erase tree IDs: The numbers of the trees to erase, according to the added items list.

Batching initialized: Flag used to synchronize the system's batching procedure with the tree scripts **(debug)**

LOLed: Flag to show the system is in LOD mode **(debug)**

NOTE: The **LOD distance parameters can be declared inside the script** in order to have a global effect in the automatic forest creation & handling. Use the "Deactivate_Hero_..." parameters in "ControlCombineINdiniDy.cs" script to define the required distances.

SAVE TREE to disk

The system offers the option to save a created tree as a prefab on the hard disk. The combiner must have only one tree, not be in LOD mode and not be in interactive mode. The name and folder of the saved tree are pre-defined and can be changed in the combiner script.

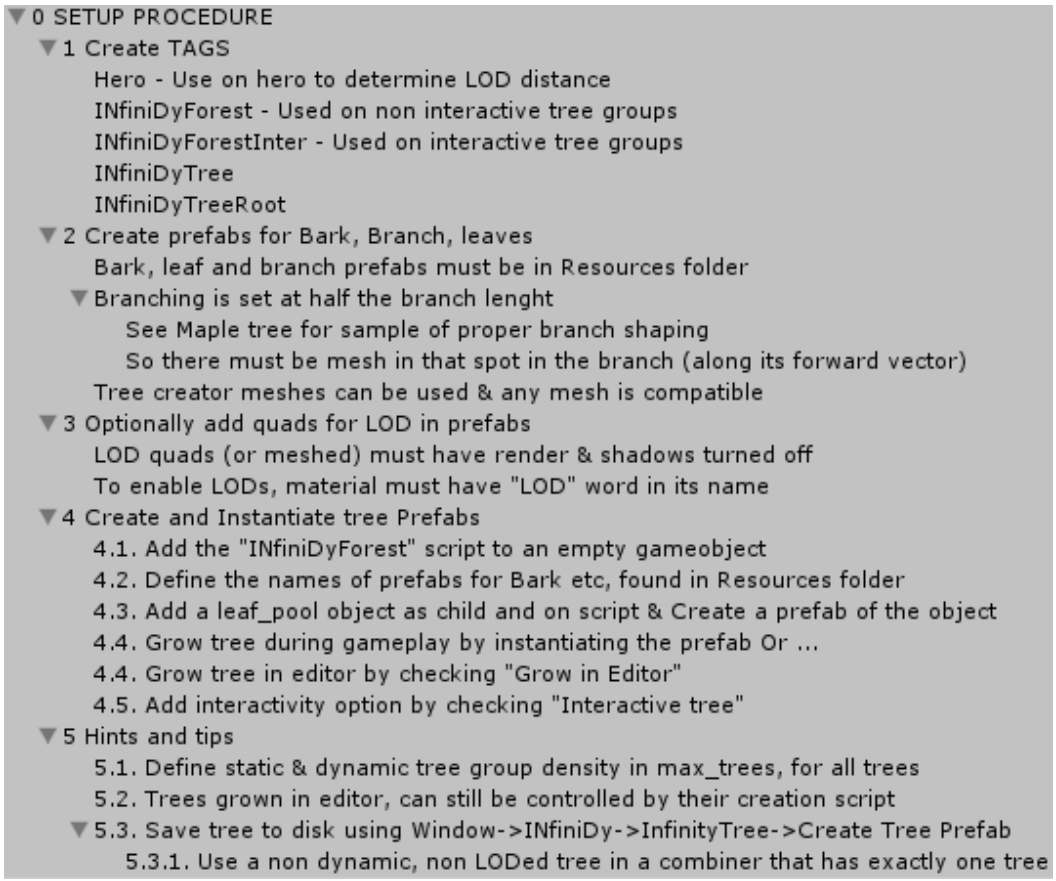
Select the combiner script with the above features and use the Window-> Infinidy -> InfiniTree -> Create tree prefab command to create the tree meshes and prefab to the desired directory.

SAVE TREE as Obj file

The script "ObjExporter.cs" in Version 1.5 folder will export selected items as *.obj file. The command can be found in File -> Export -> Wavefront Obj.

Step by step Setup Procedure

The setup process for a new tree is described below. The procedure is also included in the demo scene (image below), so can be easily copied and used in new scenes for reference.



1. Create required Tags

The following tags need to exist in the project for the system to function.

Hero - Use on hero to determine LOD distance.

INfiniDyForest - Used on non interactive tree groups.

INfiniDyForestInter - Used on interactive tree groups.

INfiniDyTree

INfiniDyTreeRoot

2. Create tree parts

At the next stage the prefabs for Bark, Branch and leaves must be created, either using custom meshes or Unity Tree Creator. Samples for bark and branches with the Tree Creator are included in all the library trees. The prefabs that get instantiated must be in the Resources folder, at its root. There can be multiple Resource folders, at any directory depth, but the prefabs must be in their root to be found.

Note 1: The branching is done at half the branch length, so that part of the branch must have a visible mesh (along its forward vector), the rest of the branch can be shaped at will. The starting part of the branch must also have mesh visible. Make sure the origin and the mid way mesh areas are aligned across the branch vector.

Note 2: See Maple Tree from the tree library, for a sample of proper branch shaping with a non straight branch.

3. Create LOD for the tree

Optionally, quads (or any other custom mesh) can be added to any tree part, representing the LOD mesh of the part.

To activate the LOD, the material of these meshes must have the “LOD” word inside it (and likewise other materials must not contain it, or will be used as LOD parts). LOD quads (or meshed) must have their renderer & shadows turned off in the tree part prefab.

4. Create tree generator prefab

Add the "INfiniDyForest" script to an empty gameobject.

Define the names of prefabs for Bark etc, found in Resources folder.

Add a leaf pool object as child to the gameobject, add its transform to the corresponding tree script parameter & create a prefab of the object in a new folder in the Tree Library.

Grow the tree during gameplay by instantiating the prefab (sample code in the script attached to the camera, in the demo scene), or placing it on the scene at the desired point or grow the tree in editor by checking "Grow in Editor". The tree may be altered in the editor (moved, rotated etc for both whole tree and individual branches) and will be controlled by the tree script during gameplay.

Add interactivity option by checking "Interactive tree", this will allow the tree to move or change shape when the hero is close. The script has a sample motion for the branches and leaves, but any kind of interaction is possible through extra scripting. More options and a library of interactions will be available in later updates.

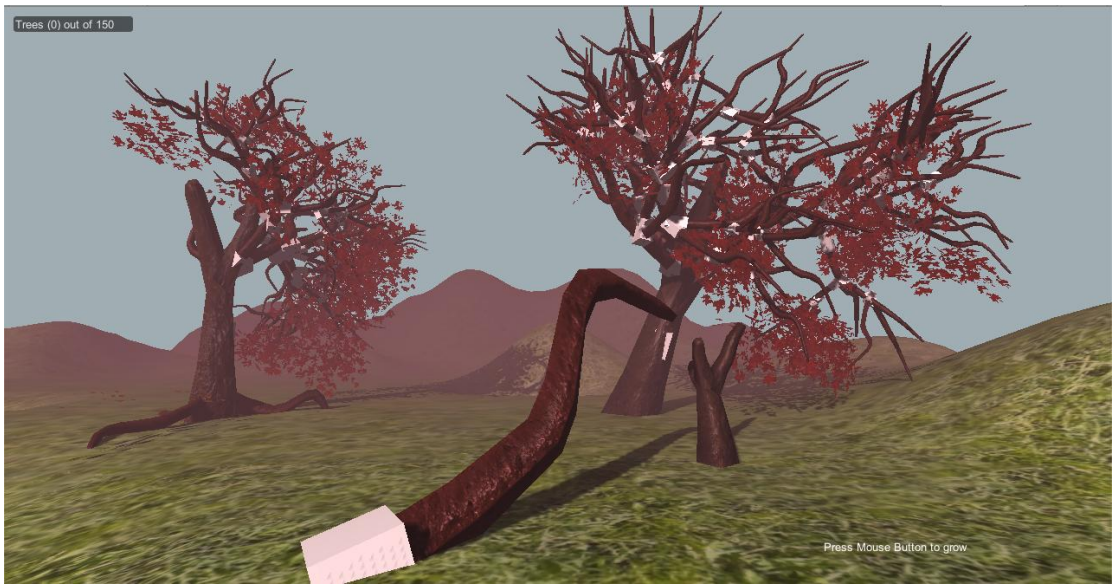
Tree Grouping parameters

It is possible to define how many trees will go into each group (static and dynamic) so these values are used globally when the systems are created on the fly.

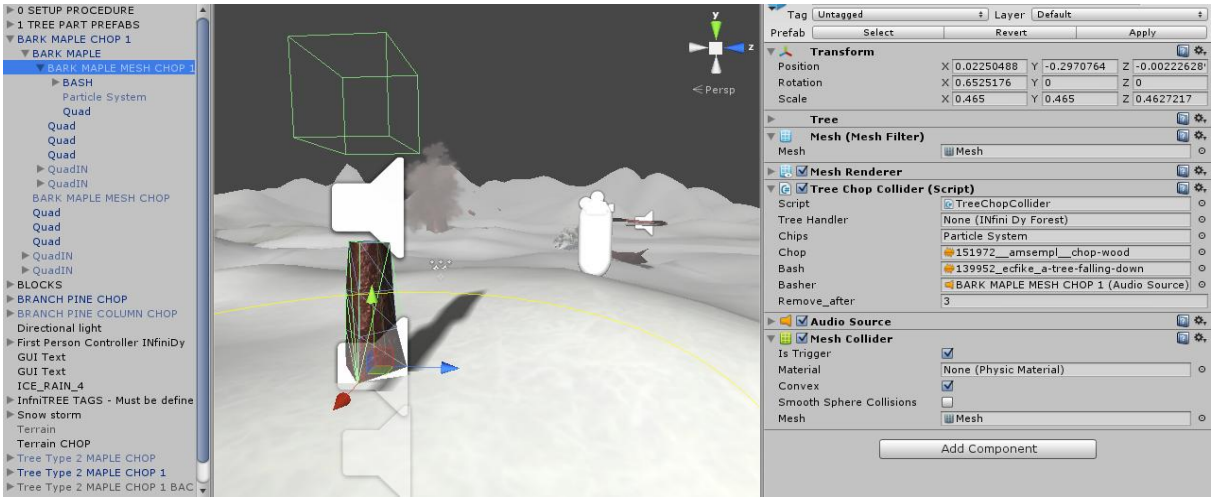
Use the **“Max_trees_per_group”** (static trees - default:12) and **“Max_interact_holder_item”** (interactive trees - default:2) in **INfiniDyForest.cs** script to control how many trees will be allowed in static and interactive tree cases. These groups are opened each time a new tree is inserted for static trees (cheap operation with multithreading, possible to use a large number depending on the target system) and while interactive trees are in dynamic phase (expensive and depending on tree complexity – assets, constantly open when in constant dynamics like the wind motion sample, cheaper in the tree chop case).

Sample of tree leaf combinations. Multiple types of leaves can be provided individually and Unity Tree Creator leaves integrated on the branches can be used as well.

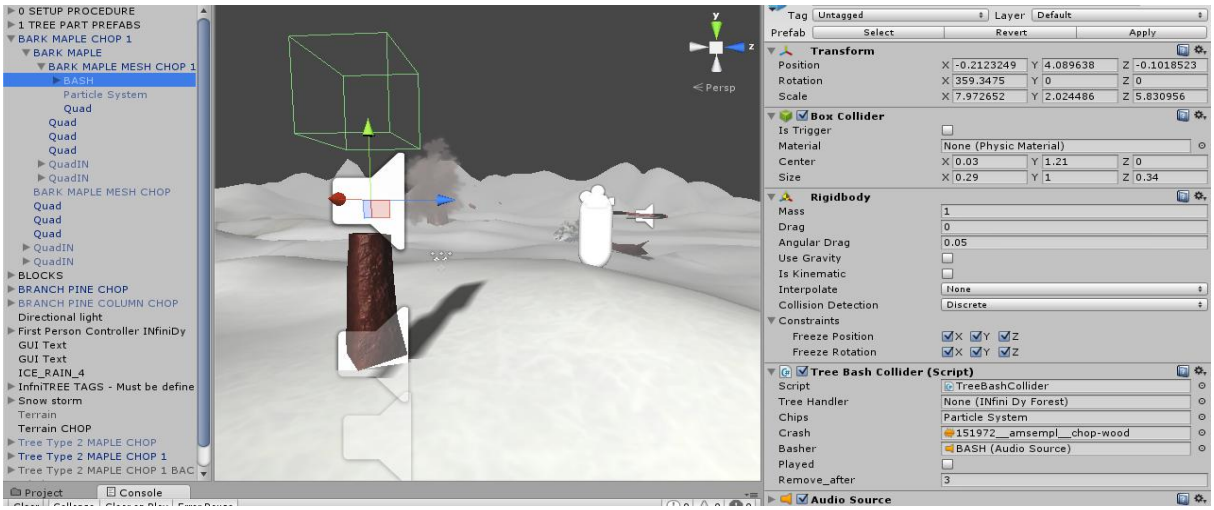
Sample of tree bark and branch prefabs and the resulting tree.



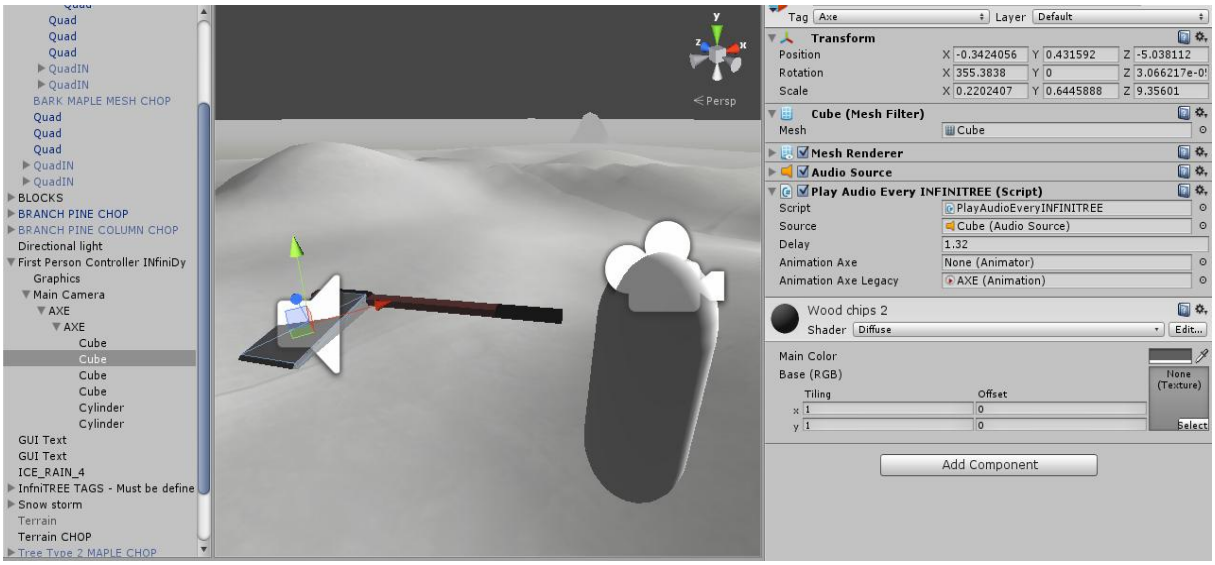
Tree chopping



Tree bark setup for axe chopping. Define sounds for the chop, bash and audio source.



Tree bark setup for fall and crash on ground. The box collider will enable the crash sound. Fall sound will play when tree starts falling.



Axe setup for swing sound and chop. “Axe” tag is used to distinguish collision from other objects and reduce tree health accordingly.

As of version 1.6, a complete tree chopping sample and behavior is included in the pack. The included demo and prefabs show the proper setup to handle axe chopping, tree health reduction and controls for the tree fall**. There are two collision/behavior control scripts**, one that reduces tree health (TreeChopCollider.cs), using trigger collision and one that handles the crash to the ground and fall sound (TreeBashCollider.cs) using normal collision. The Axe collider must have the “Axe” tag in order to reduce the tree health properly and the tree must have a proper collider to receive the axe chops. Also it has the “PlayAudioEveryINFINITREE.cs” script attached in order to play the swing sound.

The demo also showcases how to use the rotation along normal when trees are instantiated, so they can follow the terrain. The collision normal on mouse click is inserted in the main script as normal vector and rotation is activated towards that normal for a specified time before the tree is batched.

Work in progress folder

This folder contains a few samples of the work in progress for future InfiniTREE systems, that include dynamic infinite grass, dynamic mesh generation and growth directed with user input, physics and splines and more. These systems are meant for preview and experimentation only and are not yet polished for production. The scripts and systems in this folder may go under major changes on the release versions.


Trees (4) out of 150
Maple Tree
Maple(Roots)
Snow Tree
Pine Tree
Thorn Bush
Thorns
Current: Thorns
Press Right Mouse Button to chop
Press Left Mouse Button to grow trees