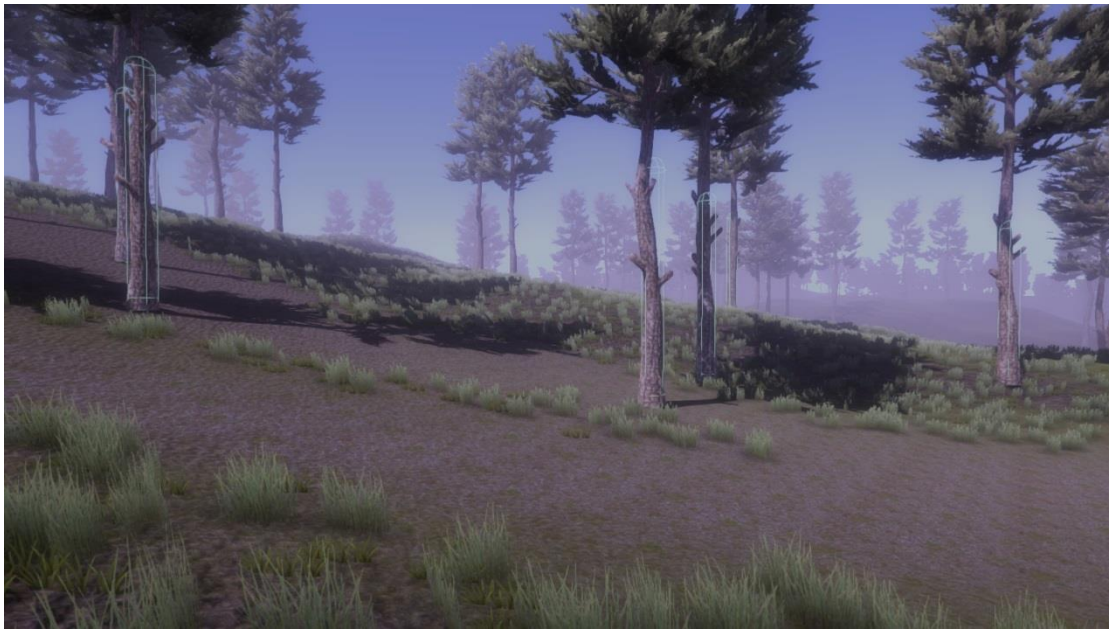
TreesManager

Advanced Tree AOI Solution

&

Including BOLT Support



About TreesManager:

First of all thank you for purchasing TreesManager, TreesManager is an Area Of Interest solution for terrain trees, and in total a Tree solution for Unity, its well optimized and still being optimized and also extremely easy to use.

TreesManager allows you to have an enormous amount of intractable trees without having any significant impact on your game performance, with TreesManager you can use terrain TreeInstances and still have them intractable, TreesManager allows you to have more than 200k trees (As long as unity can stand it) intractable trees on your scene quite easily and also has a feature to have the trees respawn after they die.

TreesManager also supports multiple networking libraries like : "BoltEngine" and "Photon", "UNet" and also "Forge", and allows you to easily implement synchronization to your game, and updating late connectors without having the trees being actual entities. That way you can save an enormous amount of bandwidth, the synchronization is happening server-sidedly to provide authoritative solution for all networking solutions.

The Area Of Interest that TreesManager offers is working by snapping colliders only to the trees surrounding you instead of instead having all the trees colliders working at the same time, that way you can still have trees intractable without having a huge fps impact.

TreesManager offers several scenes (sp demo, mp demo, world trees demo, photon demo) that you can check and work with, You can go ahead and check each one and learn how to use the system feature from both our demos and our documentation and also see the system capability.

TreesManager also offers a demo and good to use tree chop solution for you to work with, its very stable and does all the dirty job for you!, you can make your own but make sure you make it work with our TreesManager script ☺.

Content:

Getting started :

So lets get started with the basics, Go ahead and import the asset.

After you have imported the asset make sure you create the next Tags&Layers:

Layers :

Player

Tags:

Tree

Terrain

After you have created those Tags&Layers, Enter the new imported folder "TreesManager" and enter the "Scenes" folder, and then enter the "DemoSceneSP" scene.

After you have entered the scene, play the game and enable Gizmos on the Game window, and on the hierarchy, click on the new instantiated object "AOI Colliders parent", and try walking around.

You should be able to see how the Area Of Interest system procedurally snap colliders around you as you go around the map.

Try hitting a try and you should see a new prefab instantiating on the hierarchy, that prefab is a converted gameObject from the terrain's tree instances. Try hitting it a couple more times and you should be able to see the tree falling down and at the end disappears, Note that the tree fall and so are basic and you can edit them as you want, its further down the docs.

Scripts:

After seeing the demo, we will now go over some important scripts on the asset and what are they responsible for, this does NOT include demo scripts explanation.

TreesManager (TreesManager->Scripts->Manager->TreesManager)

This script for example handles the whole tree converting, Apply damage to trees, Trees restoration, Tree killing, Tree grids sectors, World trees management, Tree Respawning, Tree Converting and also if you use the bolt extension, it takes care of sending global events to all player and taking care of all the GameObject faking.

TreeAOI (TreesManager->Scripts->Player->TreeAOI)

This script needs to be assigned to every prefab that needs to have Area Of Interest on. that means – the players, and all other stuff that can move and need to have tree collision.

HittableInterface (TreesManager->Scripts->Manager->HittableInterface)

This interface is basically a way for the TreesManager to detect hits, it can be replaced by any other interface you have in the weapon/ dmg system but you might have to edit the scripts abit. (Delete the script and then fix all references with your interface, altho make sure that the die functions does the same as the demo script).

BoltTreesCallbacksManager (TreesManager->Scripts->Bolt->BoltTreesCallbacksManager)

This script is only used by the bolt extension, this script basically gets all events of the trees that have been killed and updates the tree on each client, but also on the server-side when a client has connected it will update him with the current dead trees.

AOITreeDetection (TreesManager->Scripts->Tree->AOITreeDetection)

This script is used on the collision capsules of the Area Of Interest, the TreeAOI scripts automatically assigns that and what it does is just having the "HittableInterface" on it to detect the hits and send them to the TreesManager.

As always you can delete that script and replace it with what you like, again you will have to update the references.

TreeAOIEditor (TreesManager->Scripts->Editor->TreeAOIEditor)

This script is basically an editor extension of the "TreeAOI" script and its used as a custom editor to show the editable variables

TreeScript (TreesManager->Scripts->Tree->TreeScript)

This script is a basic TreeScript we provide you with that is also being used in the Demo, The script needs to be attached to the prefab of the terrain treeInstance in order for it to receive damage of the tree, send updates to the TreesManager and also on need handle all the kill mechanics (Add rigidbody, Make him fall on a certain direction and when he collides make all the renderers and colliders disabled).

PhotonTreesCallbacksManager (TreesManager->Scripts->Networking->PhotonTreesCallbacksManager)

This script is only used by the photon extension, this script basically gets all events of the trees that have been killed and updates the tree on each client, but also on the server-side when a client has connected it will update him with the current dead trees.

WorldTreeScript (TreesManager->Scripts->Manager->WorldTreeScript)

This script is a way for the "TreesManager" script to know what Objects needs to be transformed into terrain trees. On this script you will be able to see an Int field.

"TreePrototype" This field needs to be assigned with the terrain tree prototype that this tree needs to be (for example 0 for the first prototype, This is explained further down in the documentation).

IPollable (TreesManager->Scripts->Poling->IPollable)

This interface is used for certain things that needs a specific poll restart, for example trees needs to remove their rigidbody, need to stop all the coroutines etc, and with pollable you can do that as its being called when that tree is returning to the poll.

PollManager (TreesManager->Scripts->Poling->PollManager)

This script handles all the polling in the system, it manages polling of trees and colliders and save you the whole instantiate and destroy methods which are very expensive, and that way it saves a lot of performance.

ForgeTreesCallbacksManager (TreesManager->Scripts->Networking-> ForgeTreesCallbacksManager)

This script is being used by the forge extension and it is used to receive and send all auth data to the rest of the clients and also late joiners. It will also handle the auth and un auth filters for the players.

UNetTreesCallbackManager (TreesManager->Scripts->Networking-> UNetTreesCallbackManager)

This script is being used by the UNet extension and it is used to hold information about who is the server and that way decide the control variable on the client and server.

UNetTreesNetworkingManager (TreesManager->Scripts->Networking-> UNetTreesNetworkingManager)

This script is being used by the UNet extension and it’s the networking manager we made for you so you will have all the callbacks needed and it will handle late joiners, syncing data and receiving data accordingly.

Using Your Own Resources:

In this section we will learn how to implement your own tree, or player, or terrain into the TreesManager asset.

First of all go to your "Playing" scenes and then go to the terrain and untick the "Enable Tree Colliders" variable on the "Terrain Collider".

You will also need to create a new empty GameObject and add the "TreesManager" script to it, or you can drag the "TreesManager" prefab from (TreesManager->Prefabs->TreeManager).

You have to make sure all of your playing scenes have that asset if you want TreeAOI to work In them.

Also, make sure that each terrain that has environment on it is tagged as "Terrain" so the TreesManager will identify it.

Next, go to your player prefab and add the "Tree AOI" script, you might want to leave the settings as they are but let me explain on each one.

"AOI Colliders Amount": This variable is the amount of colliders you will have on that certain prefab. Its recommended to leave that on the default but it can be changed according to the density of your trees.

"AOI Max Finding Range": This variable is the range of the Area Of Interest detection, the higher your detection range should be, the higher it should be as well (for example if you have extremely long weapon you should get it higher altho default is more than enough). Notice that sometimes it wont be as long as you made it to be because of your Colliders Amount, you will need to extend them as well, altho on most cases the default amount of Colliders should be enough.

"AOI Updating Frequency": That variable determines the frequency of the tree Area Of Interest range sorting, and re assigning of colliders. The lower you take it, the faster it will update the AOI but there will be a bigger performance impact depending on your tree amount, if you higher it then it will update slower but also have better performance. The AOI Updating Frequency needs to be assigned by seconds. (Update each second, each two seconds etc.)

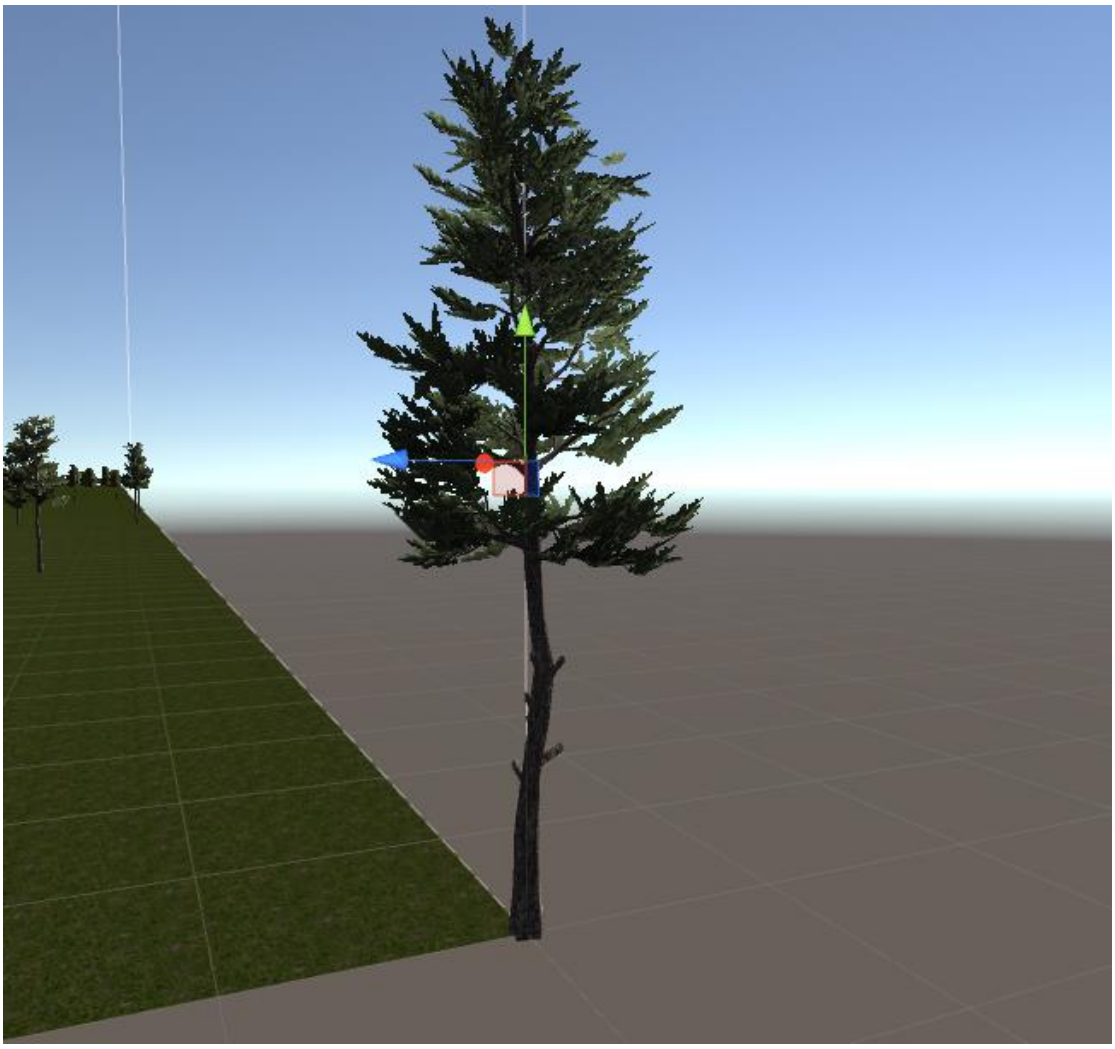
After you have assigned the "Tree AOI" script and made sure the variables are to your liking, go to your terrain tree prefabs that you **WANT** the Area Of Interest to detect and assign them the "Tree" tag. **Only** those trees will get collision, that means you can still put the tag on the not-intractable trees if you want them to have collision, and as long as they don’t have an actual Script to handle damage on their prefab like on your intractable trees they wont get chopped down.

on the intractable trees prefabs you will need to add a "capsule collider" or any other of a collider on your tree parent and make it be on the tree and make it trigger so you can receive raycast hits on that prefab.

you will also want to add some sort of "Receiving damage mechanism and Die functionality", you can use the script that we made for you - "TreeScript" (TreesManager->Scripts->Tree->TreeScript) if you don’t have one.

You will see a variable called "TopTreeTransform", This variable is used for the detection of your tree, when the tree dies it checks if your tree has hit the terrain according to the position of the transform.

To assign the variable you will need to create a new gameObject, make it a child of the tree prefab and place it somewhere high on the tree, like on the picture :



After that, save your tree prefab, play the game and you should be able to see the Tree Area Of Interest generating colliders on your trees as you walk near them if you enable Gizmos and click on the new Created " AOI Colliders parent" on your hierarchy. The generation is limited to the distance you have assigned on your player "Tree AOI" script, if its too low you can try extending "AOI Max Finding Range" variable and if it still doesn’t help you will need to stop the game and increase the "AOI Colliders Amount", Do not extend that variable too much though to save performance.

And that’s it, that's how simple it is to implement that asset into your game.

Tree Respawns:

TreesManager includes a feature for death tree respawning, by default Tree Respawns will be disabled, although if you want to disable it or change the respawn time you can do that.

Go over to your TreesManager script on your scene, and you should see the next several variables:

"Respawn Trees": This variable decides if trees will respawn after they die. The respawn time is decided by the "Respawn Time" variable.

"Respawn Time": This variable is the amount of time the tree will respawn from after he dies. That means that for example if its 5, then 5 minutes after he dies he should go back to be a terrain tree by default.

"Check Trees Frequency": this variable decides the frequency in seconds of checking the trees time (checking if the tree should be respawned or restored). You probably should leave that as default.

"Restore Idle Trees Time": this variable decides the amount of time it will take the tree to go back to a terrain tree from when he was last hit. For example if you hit a tree and 2 min will go by (by default) he will go back and be a terrain tree again for performance reasons.

To make trees be able to respawn you should tick "Respawn Trees" and make Respawn time to be as you like. (by default : 5 min).

Implement Bolt:

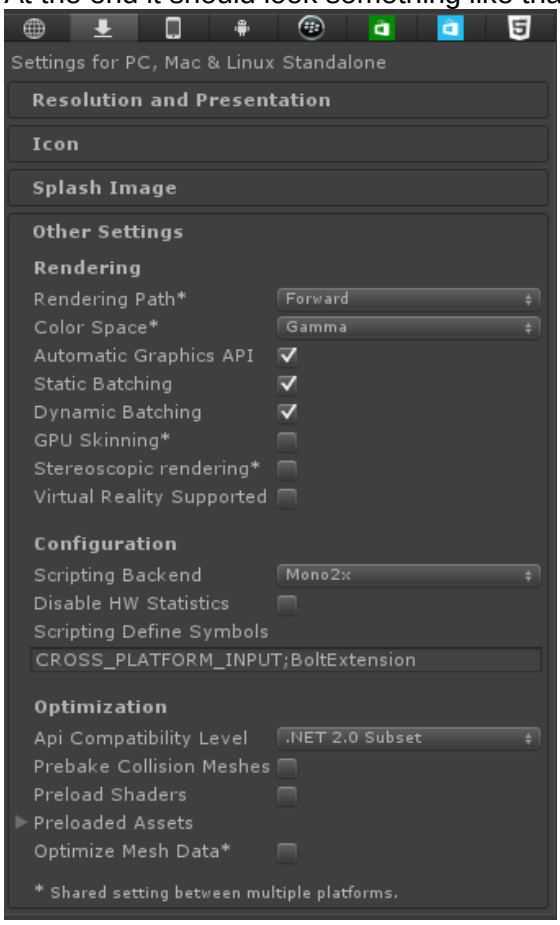
TreesManager supports bolt. If you would want to implement bolt on the TreesManager System follow the next steps.

Firstly you would want to import bolt into your project if you haven't already and install it appropriately.

Then to go to (Edit->Project Settings->Player) and then go over to the "Other Settings" section and in the "Scripting Define Symbols" add that : ";BoltExtension" and press the Keyboard key "Enter" while you are on the "Scripting Define Symbols".

That will make TreesManager compile the asset to work with bolt. You **WONT** be able to make bolt work with the asset without adding that.

At the end it should look something like that :

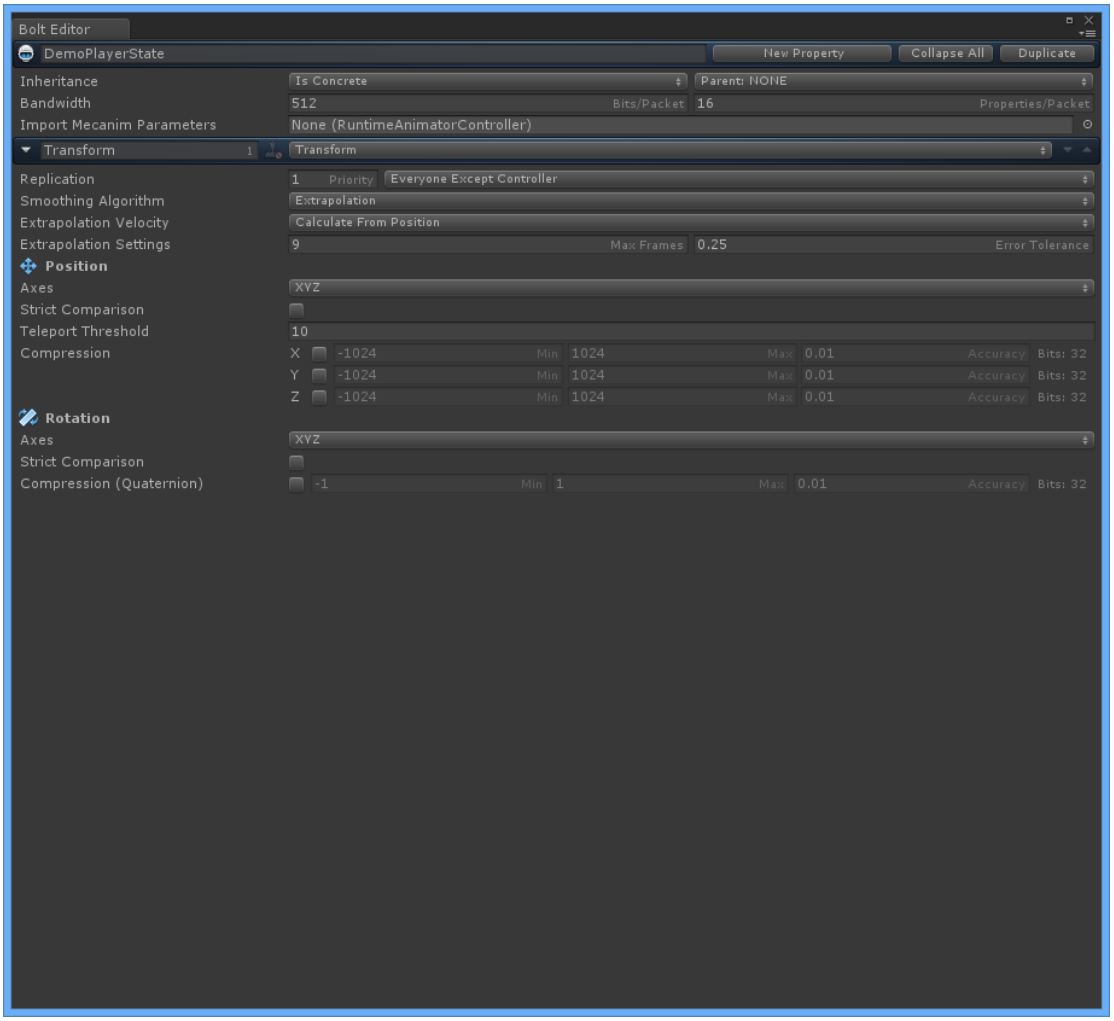


After you added that, you will need to go to "Using Your Own Resources" Stage and apply that on your actual networked player if you haven't.

Now, Go to the bolt asset and create the next state (for the demo scene, if you don't do that the demo scene will not work and you will have to remove those demo assets to fix the errors. You obviously don't have to add those if you aren't going to use the

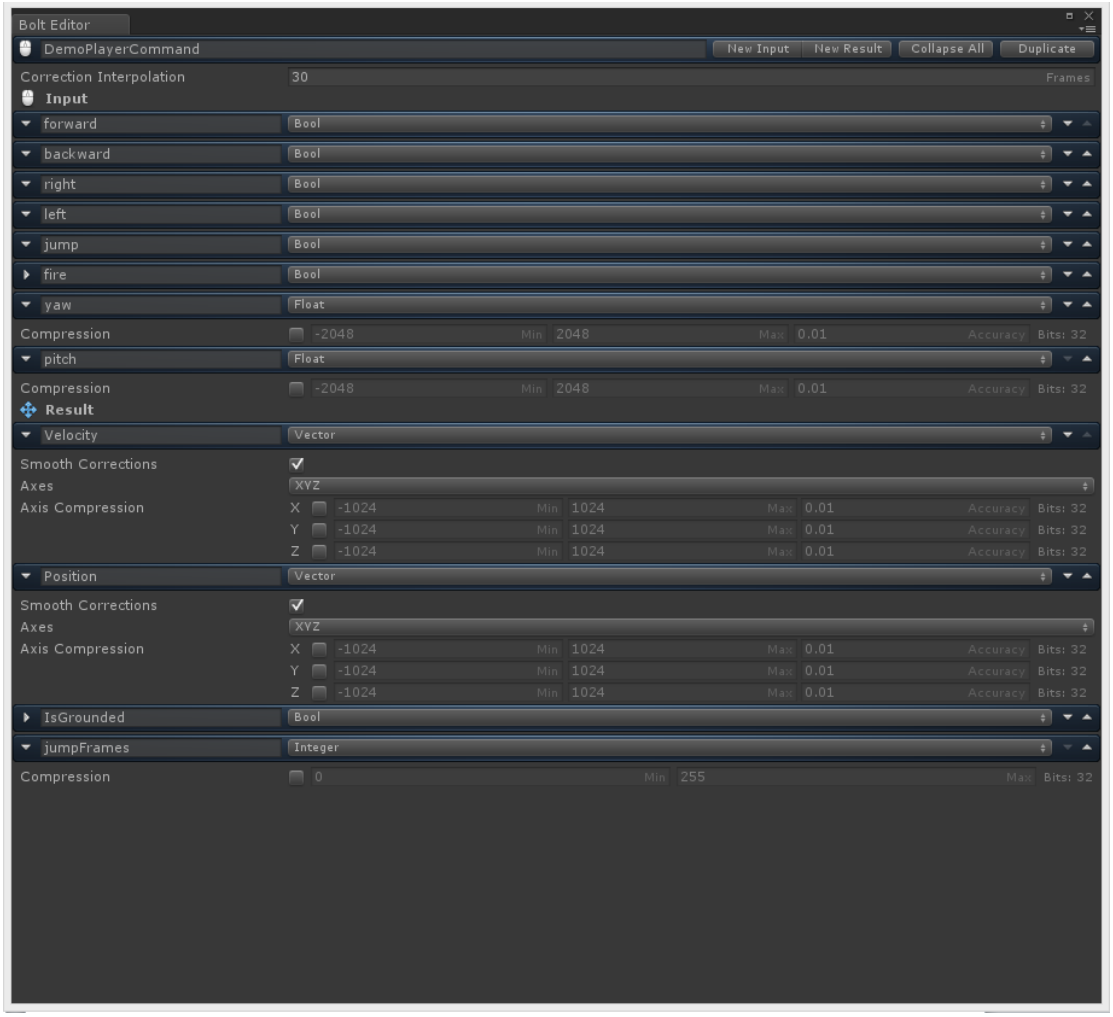
demo scene you can just delete the demo [TreesManager->Scripts->Demo] and skip to the next stage).

Create a new bolt state called "DemoPlayerState" and copy the same variables from here:



After you have done so, create a new Bolt Command called "DemoPlayerCommand"

And fill it up like here :



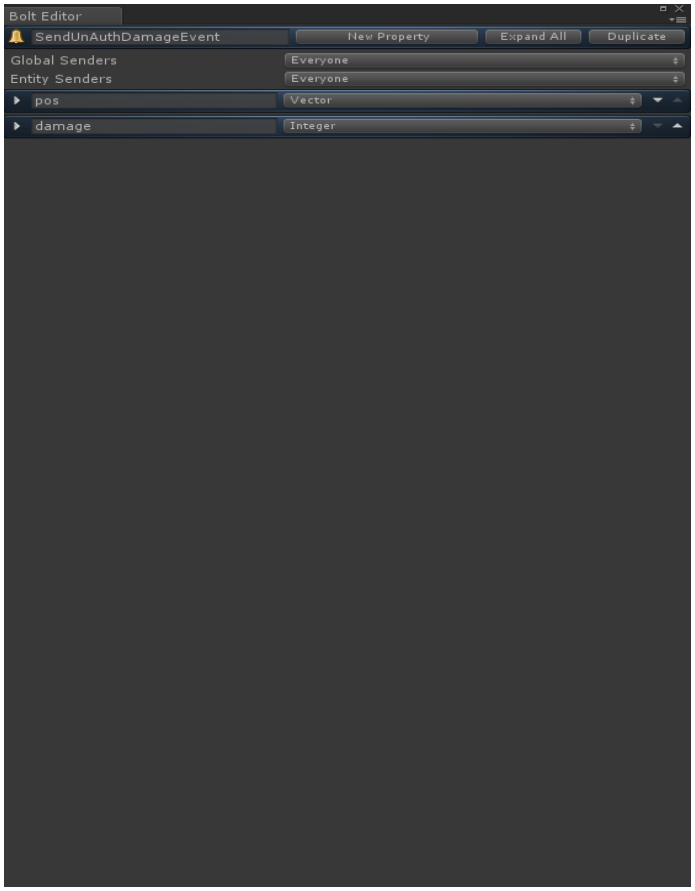
NOTE: THE NEXT BOLT EVENT IS A MUST. IT ISNT A DEMO EVENT.

The next and the last bolt asset you will need to create is the "SendResourceUpdate" event.

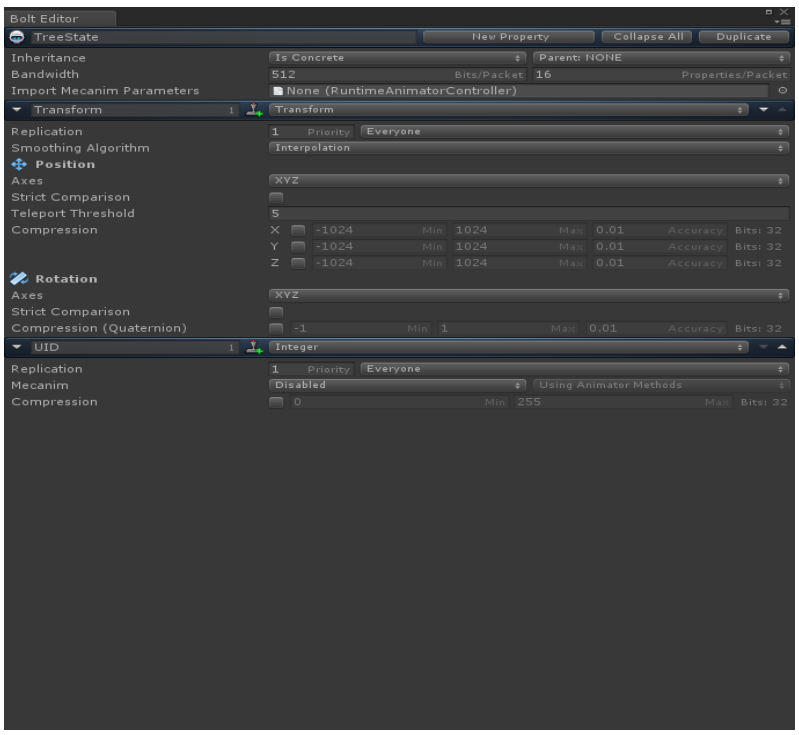
this event basically updates the clients about the tree status (dead, restore, and if its force) and with that all the clients will get that information and be able to receive all the needed information.

So create that event named "SendResourceUpdate" and make the variables exactly like the picture : (Next Page).

After that you need to create an another event that will be used for un-auth damage registration, even if you use an auth approach you will still need to create this, so create an event named "SendUnAuthDamageEvent" and create the variables like this:



Now, you will also have to create a certain state for the "Networked Objects" feature or you will have some errors. So create a new state called "TreeState" and create the variables like here:



After you have done all that, compile your bolt and all the errors should be gone.

Now all you need to do is connect with a friend to a server and check it out. (you can use the Bolt_Demo if you have made the demo stages.) to use the demo you will have to go back to the MP_Player prefab on TreesManager->Resources->Players -> Bolt_Player and assign the IDemoPlayerState to the entity state.

And that’s it, after all that the TreesManager will support bolt and sync all the death trees, and so on automatically for you.

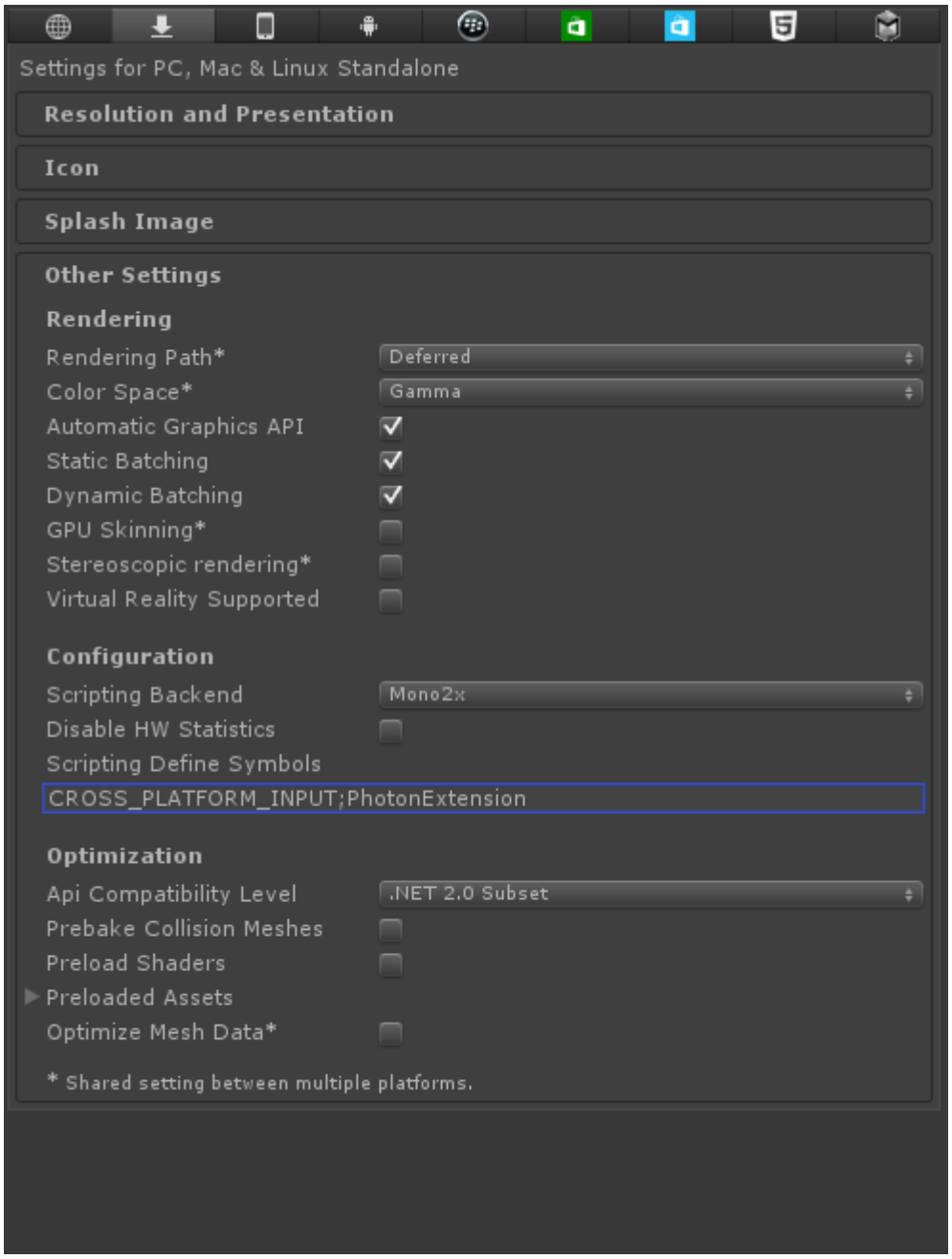
Implement Photon:

TreesManager has built-in support for photon, to use this extension accordingly, Please follow this documentation.

First of all, import photon from the asset store and sort out all the App ID stuff.

After you have done so go to (Edit -> Project Settings -> Player)

And in the "Scripting Define Symbols" add that : ";PhotonExtension" and make sure you press enter while you are at the input field:



After you have done that, Go ahead and open the "DemoScenePhoton" Scene (Scenes - > DemoScenePhoton) and look at the "TreeManager" GameObject.

This object includes 4 scripts:

* TreesManager – Most important script, YOU MUST have this on your scene, EVERY Scene for you to have TreesManagerSystem working.

* PhotonView – This script is only used on the photon extension, its used on this case for sending events between client to master client and sync data (dead tree, restore tree and so on).

* PhotonTreesCallbacks – This script is also only used on the photon extension and it's a MUST on a photon extension-using scene. Including the photonView and treesManager (you would want them 3 on the same object on the scene).

* DemoPhotonServerConnector – This is a demo script used on the photon extension to connect to a random server upon startup.

NOTE : On every photon-extension using scene you must have : TreesManager,PhotonView and PhotonTreesCallback on the scene and on the same GameObject.

Now go ahead and run the scene and play around, you can also build the scene and try it with a friend!, but first make sure the scene is in your building menu.

Using World Objects Trees:

TreesManagerSystem also now support world objects and can convert them for you and make them intractable and batched into the terrain and also networked.

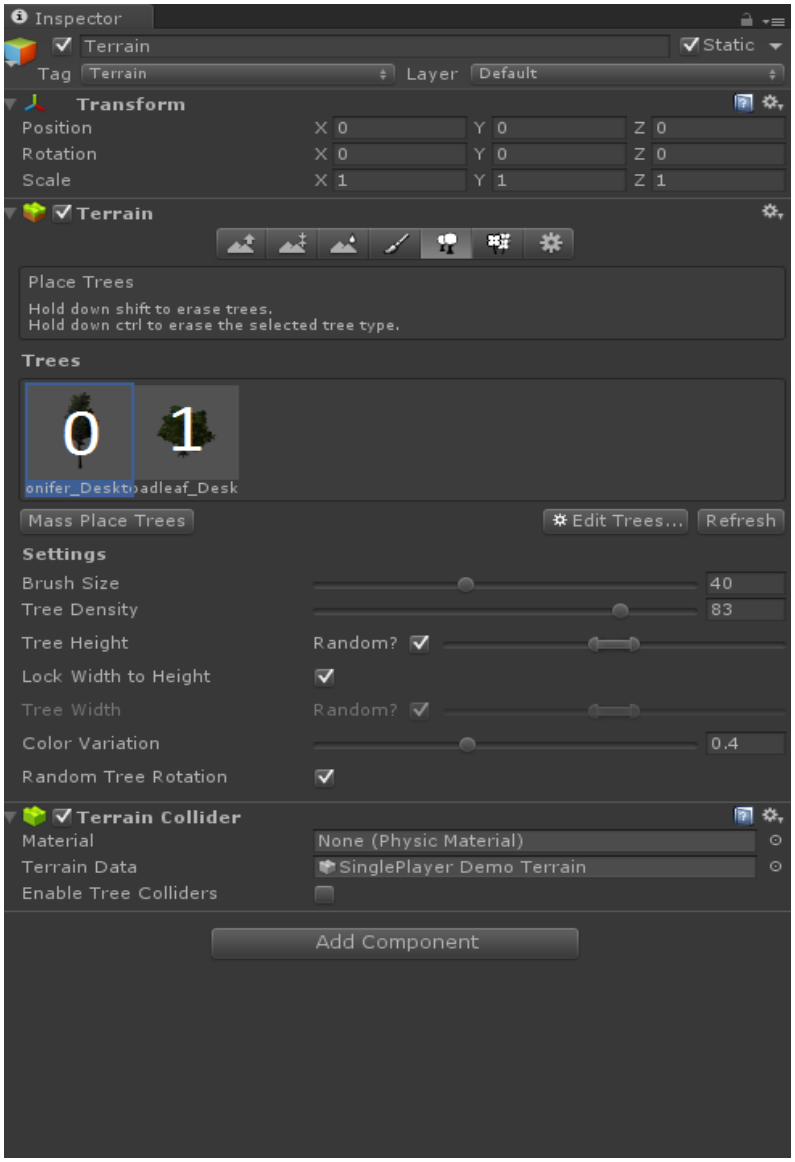
So lets get started shall we?

First of all pick the tree (you can choose multiple trees together) and add a script to them called "WorldTreeScript" (Scripts -> Manager -> WorldTreeScript).

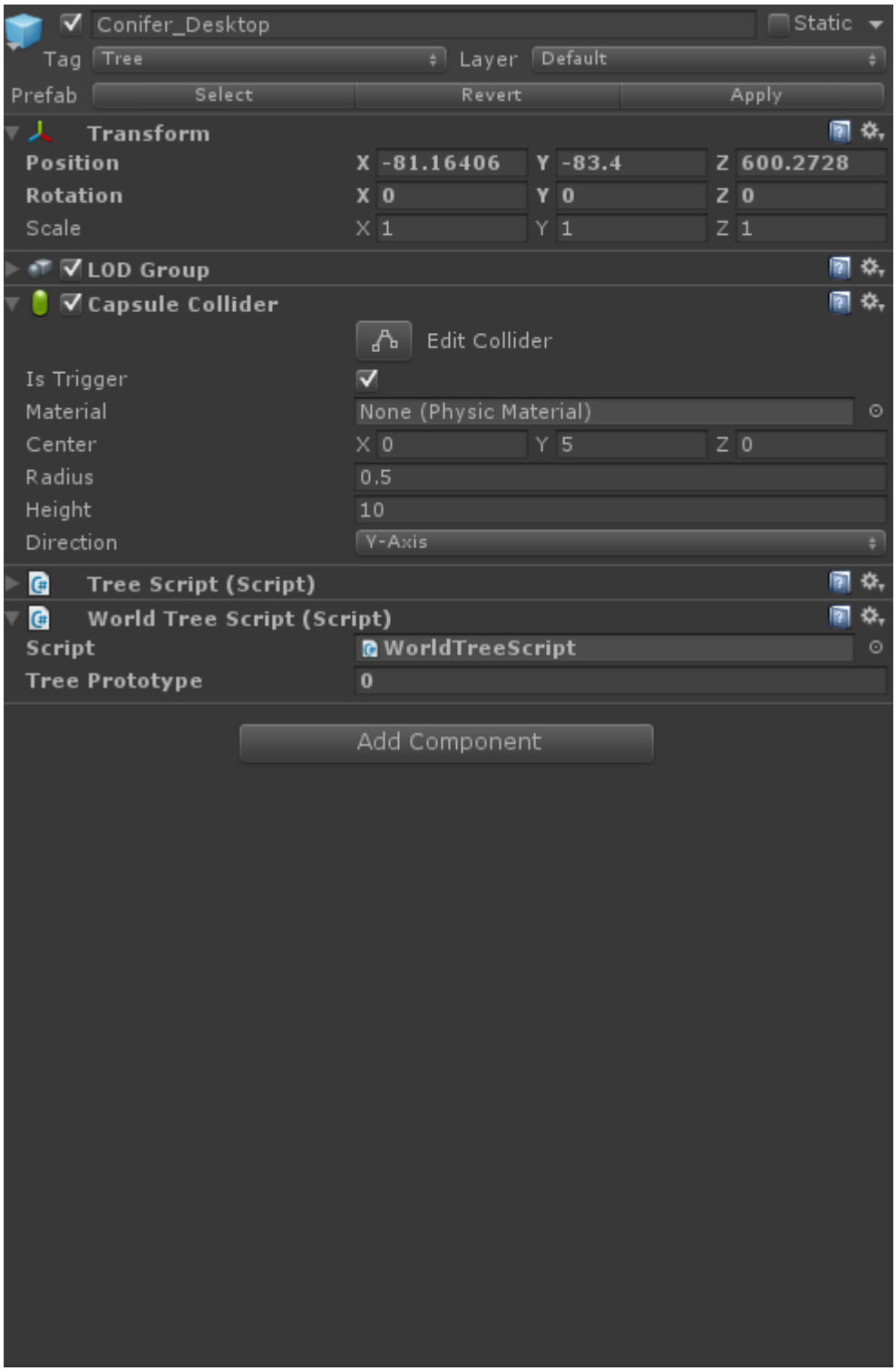
And you should see an int field on it called "treePrototype".

TreePrototype is the prototype that will be used from the terrain as a prefab from that tree, basically it should be the same prefab but set up as tree prototype.

So how do we set this up? Well, it should be something like that :



As you can see in the image above its being numbered just like array, according to that place your prototype number. So if I am using the first prototype (Confier_Desktop) in the scene It will be setup like this:



And then run the scene and that should work perfectly for you (it should batch them, and make them intractable and networked). You can also check the "

DemoSceneSP_WorldTrees" Scene (Scenes -> DemoSceneSP_WorldTrees), for further understanding of world trees and how to set them up. You can try running around and cut down trees, you can also go ahead and look at the trees that are parented to the GameObject "Tree" and see how they are setup to the first prototype.

And that should be it for World Trees.

Implement UNet:

TreesManager has built-in support for the new unity networking framework, to use this extension accordingly, Please follow this documentation:

First of all, Go over to the Player settings (Edit -> Project Settings -> Player),

And in the "Scripting Define Symbols" add that : ";UNetExtension" and make sure you press enter while you are at the input field:

Afterwards go ahead and edit the build settings (File -> Build Settings) and add the 2 unet demo scenes to the build settings.

First scene would be (TreesManagerAsset -> Scenes -> UNet -> UNetLoadingScene) and also (TreesManagerAsset -> Scenes -> UNet ->

DemoSceneUNet) after you do so run the "UNetLoadingScene" scene and that will allow you to check out the demo scene of UNet.

If you want to implement it to your own game, you need to make sure you have the "UNetTreesCallbackManager" script attached to the TreesManager object and also make sure you have a networkIdentity on it with localPlayerAuthority enabled.

After you do that, you will need to replace your NetworkManager with ours (Scripts -> Networking -> UNetTreesNetworkingManager) or you could use your own and call our functions when someone connects into the scene.

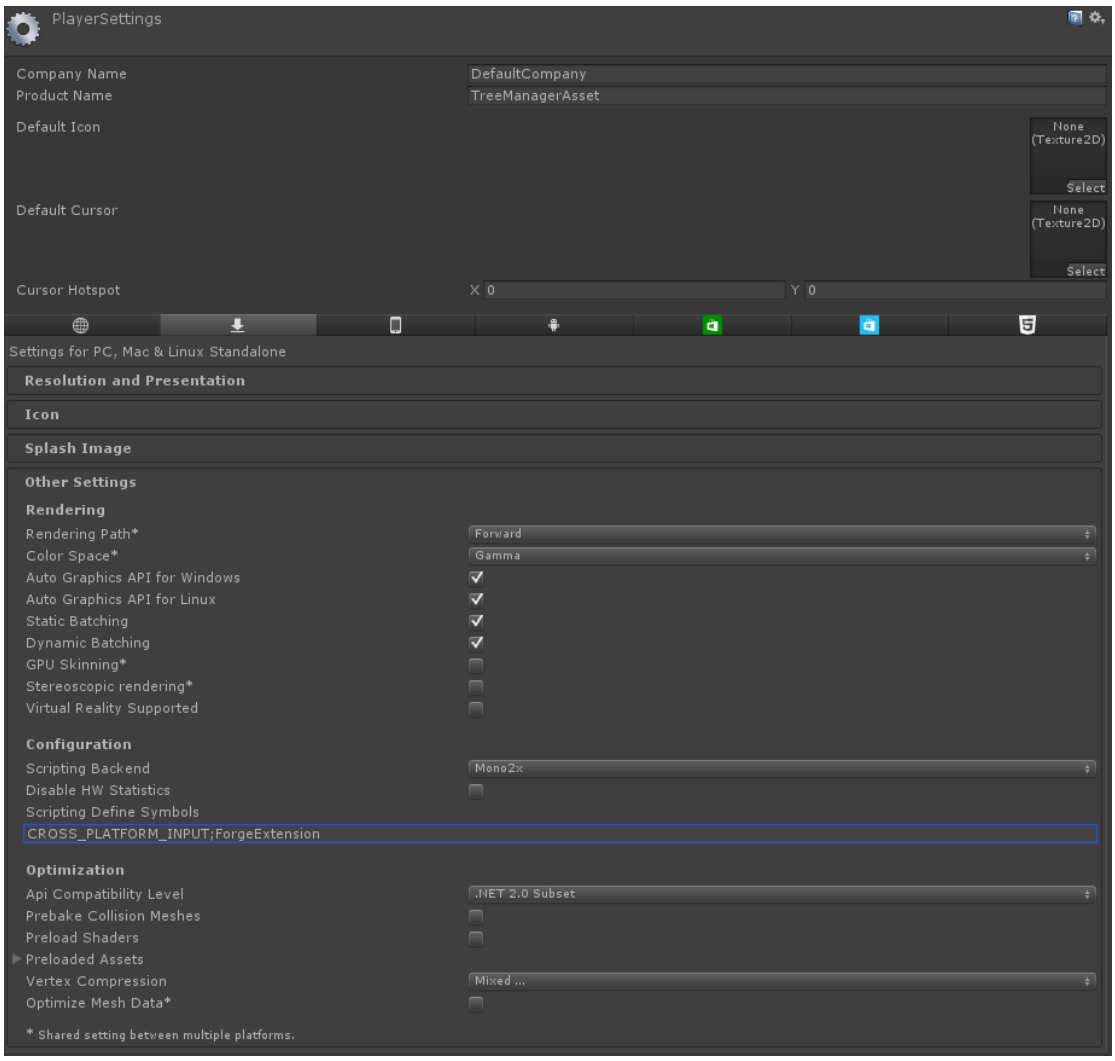
And that’s all for the UNet implemention, if you have any issues or bugs or suggestions contact us on email ☺.

Implement Forge:

Other than Photon, UNet and BoltEngine TreesManager also has built-in support for forge networking. To implement that extension make sure you have forge imported into your project. (You should pull the latest version from the website as the website always has the updated version way ahead of the asset store).

After you have imported forge go into the Player settings (Edit -> Project Settings -> Player),

And in the "Scripting Define Symbols" add that : ";ForgeExtension" and make sure you press enter while you are at the input field:



After you do so, go ahead and add the demo forge scenes into your build settings (File -> Build Settings) :

First scene would be (TreesManagerAsset -> Scenes -> Forge -> ForgeQuickStartMenu) and also (TreesManagerAsset -> Scenes -> Forge ->

DemoSceneForge) after you do so run the " ForgeQuickStartMenu " scene and that will allow you to check out the demo scene of Forge.

If you want to implement it to your own scene, you need to make sure you have the " ForgeTreesCallbacksManager" script attached to the TreesManager object.

After you do that, hit play and it should work perfectly for you and the Forge extension would be enabled and working on your scene.

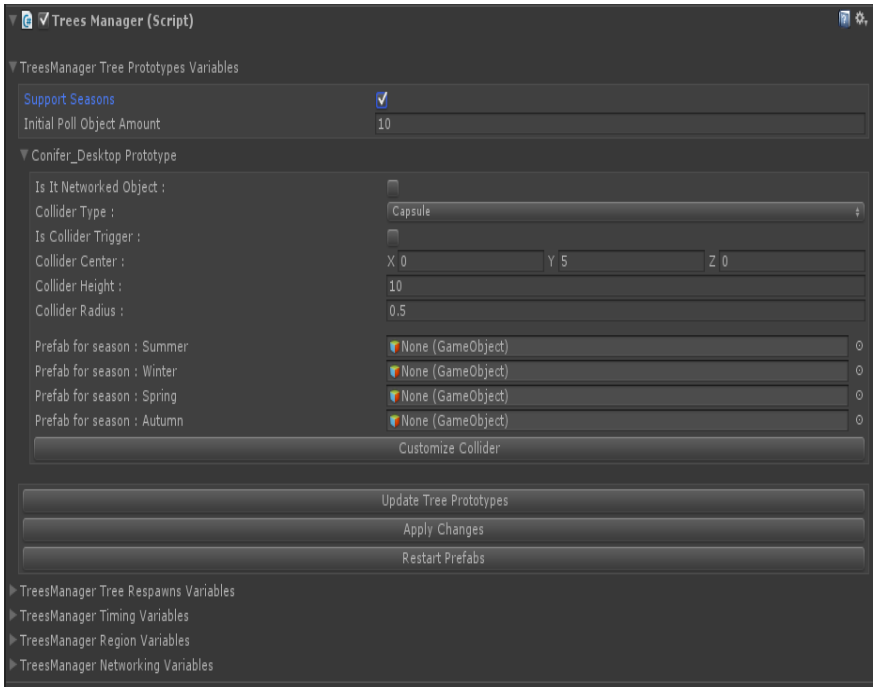
Seasons Support:

TreesManager has a built-in support for seasons, that means that by changing 1 variable certain trees that you choose will switch their model and will switch from lets say a summer tree to a snow tree.

So what are we waiting for, lets get started.

Go into your TreesManager script in your play scene and open up the " TreeManager Tree Prototypes Variables" tab and enable the "Support Seasons".

Now you should be able to see that each tree prototype in the whole scene is listed under the "Support Seasons" Toggle, try to open one of them and you should see that :



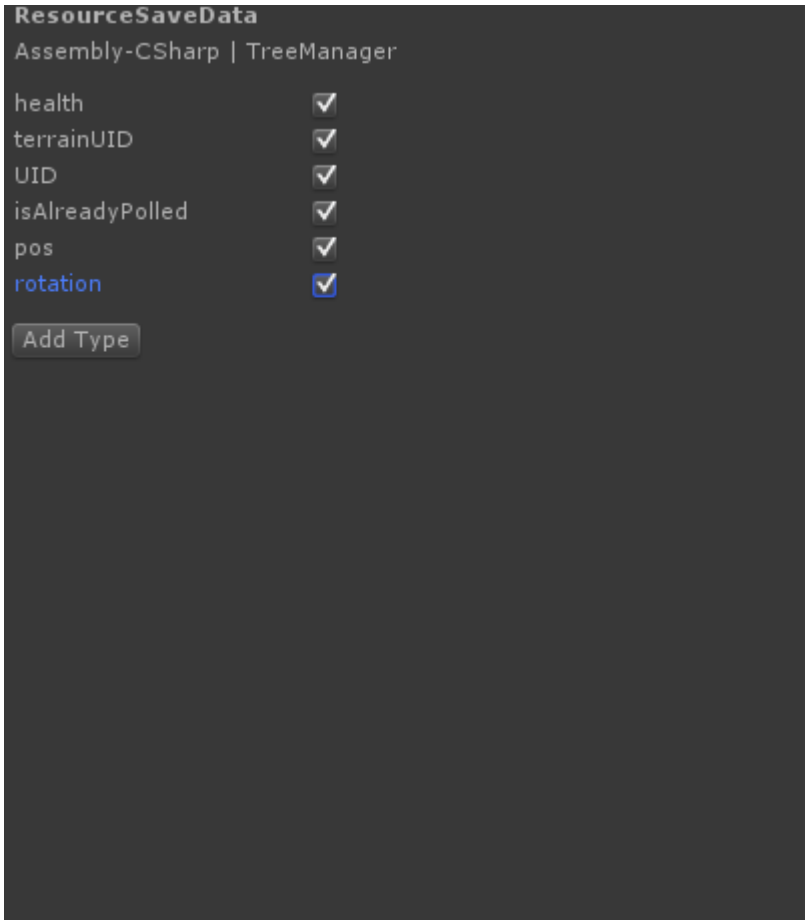
Now, you need to assign for each prototype a tree for each season (Summer, Winter, Spring and Autumn), if you don’t want a tree to be changed in a certain season then leave the certain season null, if you don’t want the tree to change at all then leave all of his seasons null.

Now just pop in game and try changing the TreesManager static variable "currentSeason" to a certain season and see how it changes the trees accordingly.

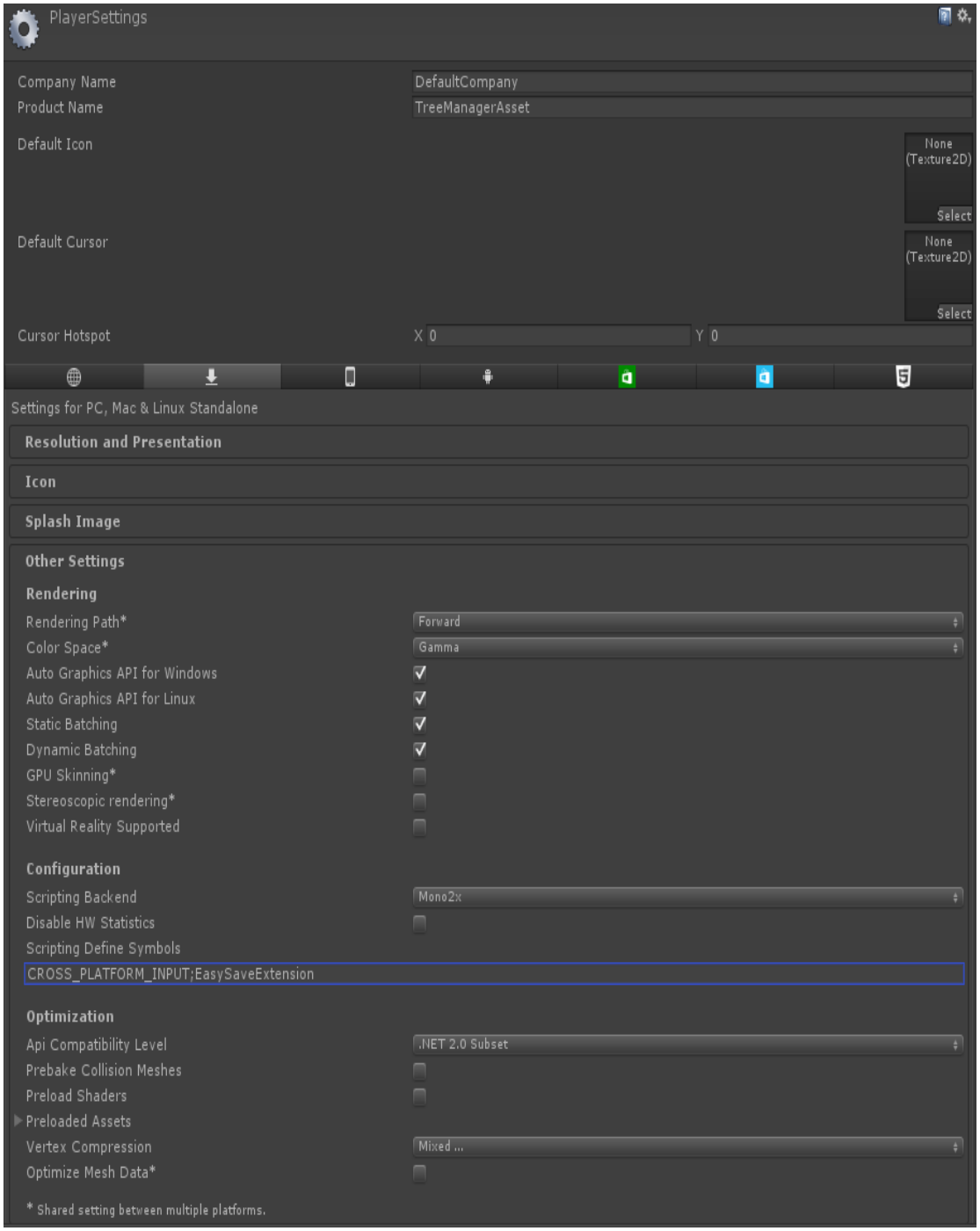
That’s all for the seasons, its really simple but powerful.

Easy-Save Support:

TreesManager has a built-in support for easy-save, using that will make it so when ever you close the game, it will save all the trees in the map for you and load it to you when you open the game again, in order to use that feature you must first import the "Easy-Save" asset and then go to "Assets->Easy Save 2->Manage Types" And look for "ResourcesSaveData", click on it and mark all the variables on it, like that :



And click on "Add Type", After you have done that, you will need to add the "EasySaveExtension" into the "Scripting Define Symbols"

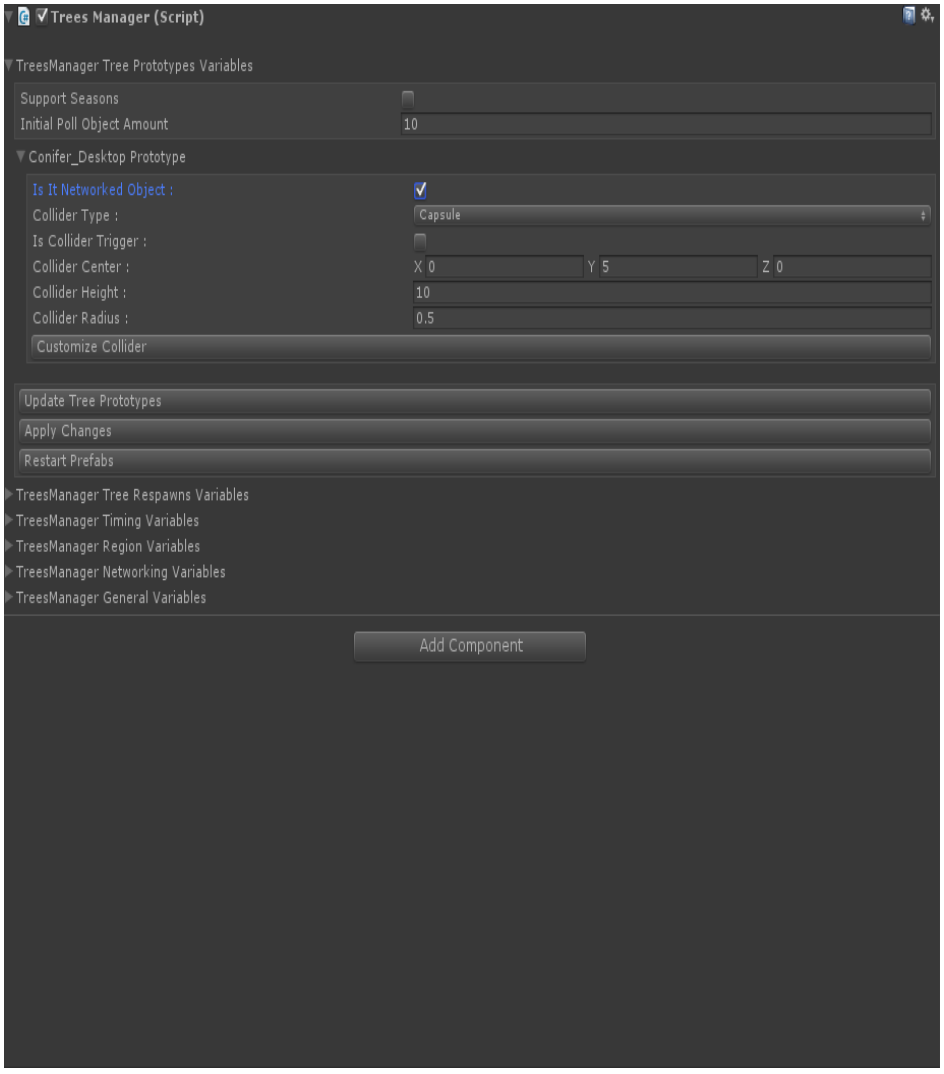


After you have done so, go over to the TreesManager script in your scene and enable "Save Resources" Toggle inside the "TreesManager General Variables" tab. After you have done that apply the changes and that’s it. Go over to the game and test it out ☺.

Networked Objects Support:

TreesManager supports the ability to use the networking library's sync method that you are using. So lets say you are using bolt that would be "Bolt Entity" if you are using Photon that would be "Photon view".

Implementing this feature is mostly the same for each networking library so lets get started. Firstly go over to your TreesManager script in your scene and then open up the "TreesManager Tree Prototypes Variables" tab, inside that tab open up the tab of your tree prototype that you want to implement that feature on (you don't have to use that feature for every tree, you can choose for which tree you want to do it and which trees you don't), and after you have opened it, toggle the "Is It Networked Object" on and click on the "Apply Changes" button. So something like that :



After you have done that, go over to that prototype prefab and add to him your networking library sync method (for example, bolt – bolt entity, photon – photon view and so on), and also add your position sync to it. Now lets work on the actual Networked Object feature.

So to start that, please note that if you are a Bolt Engine user you will have to use the TreeScript state on the prototype prefab that you created when implementing that extension or edit that in the script that i will explain about in a minute.

So, in your prototype prefab add a new script, the script will be determined of your networking library:

Photon Networking – ("Scripts ->Tree ->NetworkedTreeScripts -> PhotonNetworkedObjectAttacher").

Bolt Networking - ("Scripts ->Tree ->NetworkedTreeScripts -> BoltNetworkedObjectAttacher").

UNet Networking - ("Scripts ->Tree ->NetworkedTreeScripts -> UNetNetworkedObjectAttacher").

Forge Networking - ("Scripts ->Tree ->NetworkedTreeScripts -> ForgeNetworkedObjectAttacher").

Now, after you have done that, build the game and test it with another client and check how it works ☺.

Designing a collider:

TreesManager system supports custom colliders designing, to use this feature, go over to your TreesManager script in your scene and click on "TreesManager Tree Prototypes Variables" and open up the prototype that you want to edit the collider of.

In there you have several variables:

Collider Type : The collider type of the collider (could be Box or Capsule).

Is Collider Trigger : Is the collider trigger ?, can be used for stuff that needs to be intractable and also you can go through them (For example wheat farms) .

Center, Height, Center, Radius and Size are all just shape variables.

To customize the collider you can put the variables in manually or click on the "Customize Collider" variable.

There just edit the collider as you like in real time and edit it as you like, when you are finished click on the "Save Collider Data" button and it will be saved into the prototype. Now Apply changes on the TreesManager script and now you have a custom collider ☺.

Support, Improvements and Bug Reports:

In case you have found any bugs, or need support or having any improvements to offer please don’t be shy to send us a mail to :

EEProductionsSupp@gmail.com

Or contact us on the support section on our site.

CLICK HERE

And we will be more than pleased to help you.