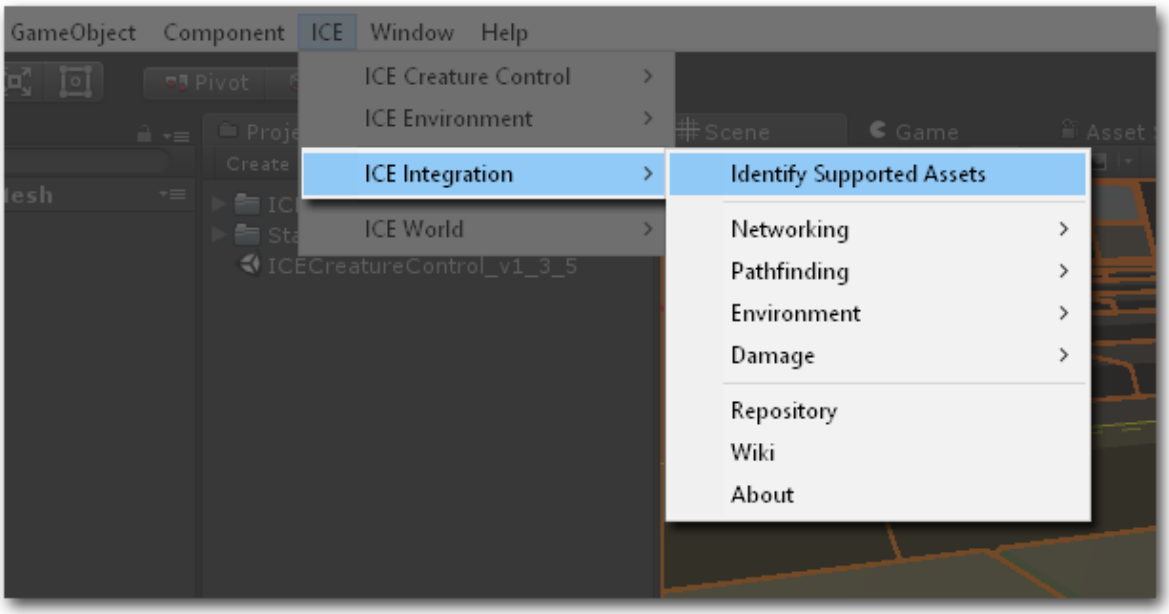
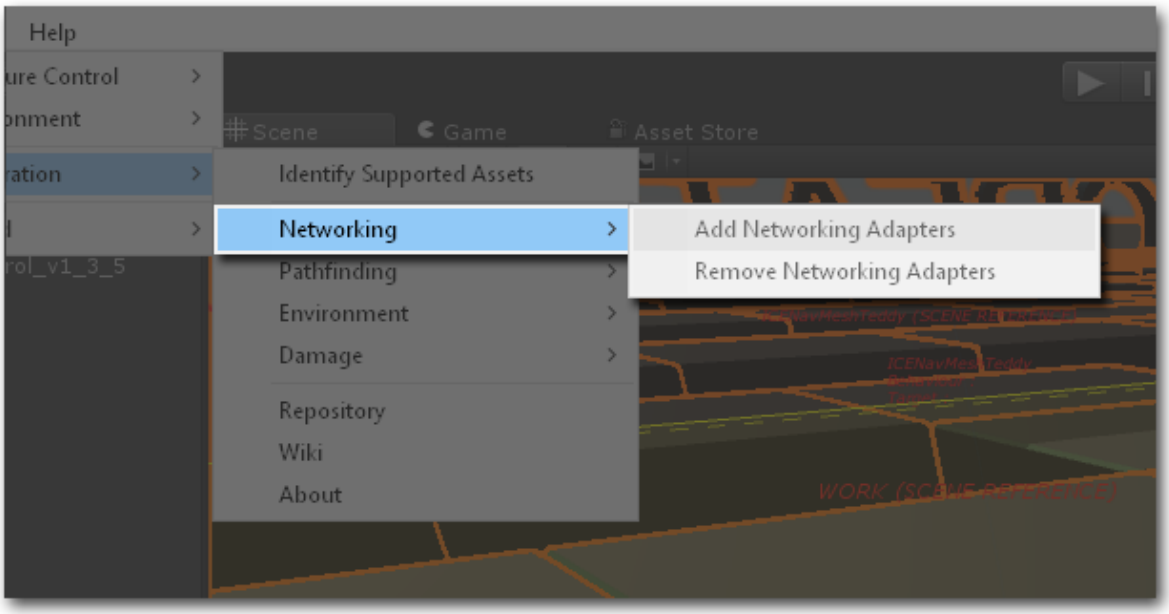
14 ICE INTEGRATION

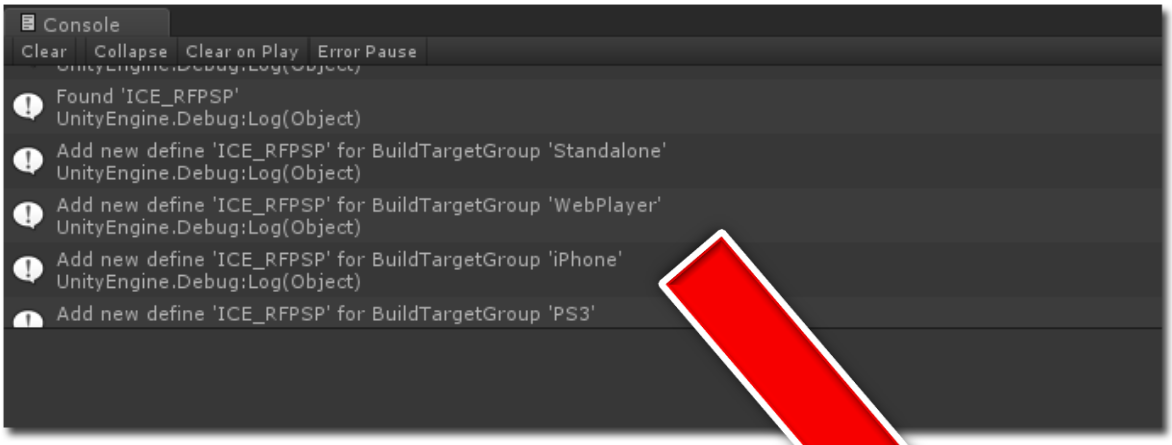
To support the smooth integration of 3rd Party Products ICE Integration comes with a couple of optional scripts which handles the integration automatically without custom code.

To combine ICE with supported third party products, just open the ICE Integration Menu and press ‘Identify Supported Assets’. ICE will scan now your project folder for supported files and will add a custom define for each detected and useable asset.

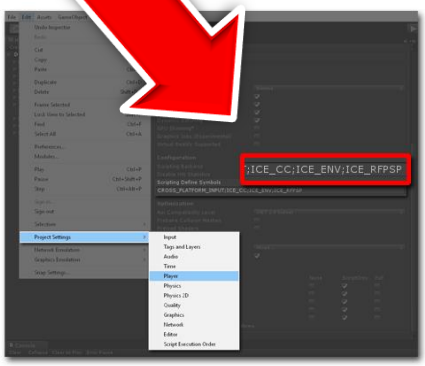


To integrate now specific assets, just select the desired menu group and press ‘Add Adapters’. ICE will scan now your hierarchy to find relevant Components by their type and will add the corresponding adapter to its GameObject. You can reverse this step at all times by pressing ‘Remove Adapters’.



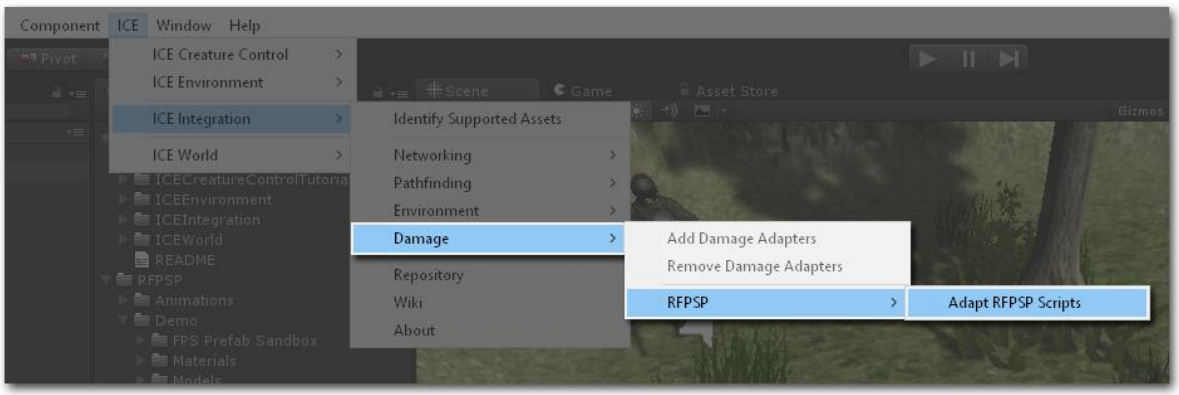


All steps and automatic procedures will be logged in the console, so you can follow and retrace all changes. Also you can open your Player Settings to see which Assets was detected. All ICE based defines will starting with 'ICE'. Please feel free to remove all needless defines as desired, this will avoid overhead by deactivating the unneeded code sections.

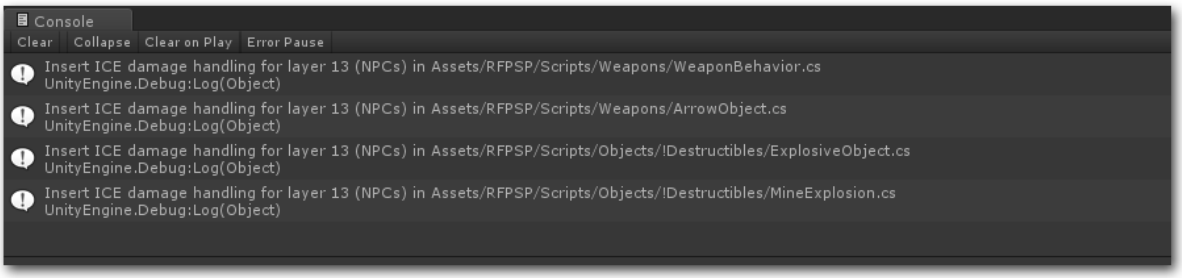


Basically the both above-mentioned steps will handle the complete integration process, so there is nothing else to do besides of the typical component settings. The most adapters do not need any additional configurations and will be ready to work as soon as they are added to their GameObject, but there are some asset packages we have to pay a bit more attention, because in addition to the adapters such packages requires also to adapt parts of their code. Basically I always try to avoid such modifications but in some cases there are no other ways to get access to the required class member, because of unfavourable code structures with missing access modifier or similar annoyances.

However, ICE can handle such required code modification for you. In cases an adapter requires code changes you will find one or more additional menu items within the Integration menu structure, just press the corresponding 'Adapt Script' item and ICE will doing the rest.



ICE will inform you also here about all changes in the console and if you want to check the modified code section after the adaptations you can searching for ‘MODIFIED BY ICE’ within your project scripts. ICE will comment each entry with a ‘// BEGIN …’ and ‘// END …’ and the above-mentioned ‘MODIFIED BY ICE’.



Btw. please consider that you will get errors or malfunctions while using an adapter who need such code adaptions without the required changes. Also make sure that you are working with the latest version of the asset and that the code is untouched. If ICE can’t find a required code sections you will get a warning and have to implement the changes by yourself.

Please note! Automated code adaptations are irreversible, so I would like to recommend you to back up your project before running the adaptations.

Well, if all this is done the integration process should be complete and your ICE product should be able to working with the 3rd party asset.

14.1 Third Party Support

Currently following assets are already supported by ICE or already requested and will coming soon.

14.1.1 Damage

14.1.1.1 Opsive's Ultimate FPS (UFPS)

14.1.1.2 Opsive's Third Person Controller (TPC)

14.1.1.3 Azuline Studio's Realistic FPS Prefab (RFPSP)

14.1.1.4 Hardworker Studio's UnitZ (UNITZ)

14.1.1.5 Gaming is Love's ORK Framework - RPG Engine (coming soon)

14.1.1.6 Calvin Weibel's Easy Weapons

14.1.2 Networking

14.1.2.1 Photon Unity Network (PUN)

14.1.3 Environment

14.1.3.1 Black Horizon Studio's UniStorm

14.1.4 Pathfinding

14.1.4.1 Aron Granberg's A Pathfinding Project*

14.1.4.2 Apex Game Tools' Apex Path (coming soon)

14.1.5 Animation

14.1.5.1 RootMotion's Final IK

14.1.5.2 Rune Skovbo Johanson's Locomotion System

14.1.6 Visual Scripting

14.1.6.1 Hutong Games LLC PlayMaker

14.1.6.2 Evasion Games' GameFlow

14.1.7 User Interface

14.1.7.1 Mad Pixel Machine's Energy Bar Toolkit

14.1.7.2 Devdog's Inventory Pro (coming soon)

Finally I would like to mention that ICE Integration and all the included scripts are completely optional and solely complementary solutions to facilitates potential integration tasks, they are NOT required to run ICECreatureControl and also NOT part of the core product, so please consider that I create and offer all these scripts for free to make the integration of 3rd Party Products easier to you. I

try to realize all requested adapters as soon as possible but please understand that there is no warranty or claim for benefits for such adapters.

14.2 Additional Supported Assets

14.2.1 Locomotion System

Maybe the free Unity Asset 'Locomotion System' by Rune Skovbo Johanson is a little bit aged but it is still perfect to enhance the quality of movements, especially if you have a creature with only some few legacy animations.

The Locomotion System and ICECreatureControl working absolutely fine together, in this combination the Locomotion System handles all ground animations while ICECreatureControl is working as character motor and defines the movements by following the given rules and execute all his other tasks.

To using the Locomotion System combined with ICECreatureControl just add the 'Leg Animator' and the 'Leg Controller' script of the Locomotion System in addition to ICECreatureControl. Configure the locomotion scripts by settings the required entries for your creature, such as the root and leg bones and the animations which are to be controlled by the locomotion system. If this is done, open the Behaviour Rules of ICECreatureControl and deactivate all animations which are controlled now by the Locomotion System – Note: don't delete the complete rule, just set the Animation Type to NONE, all other values and especially the movement settings are continue required.

Now you can press play, to check the movements. Of course you have to do the fine adjustment but you should see now, how your creature adapt his steps to the ground.

Note: To using the 'Locomotion System' with Unity 5 you have to adapt the code a little bit but the changes are easy.

14.2.1.1.1 LocomotionEditorClass.cs line 45

The 'Root Bone' Object should be a scene object, but the *ObjectField* parameter *allowSceneObjects* is flaged as *false*, so simply change it to *true*.

14.2.1.1.2 LegAnimator.cs line 252 and line 286

There are two code segments where the LegAnimator creates *new AnimationClips*, these new clips must be flagged as *legacy* otherwise the Animation Component can't handle these clips. You can fixed it easy by doing something like this:

AnimationClip _clip = new AnimationClip();

_clip.legacy = true;

GetComponent<Animation>().AddClip(_clip, "LocomotionSystem");

Note: The Locomotion System works currently only with legacy animations, therefore please consider that Unity intends to phase out the Legacy animation system over time and it is NOT advisable to use it for new and especially not for larger Projects.