Weaver Change Log

Weaver 3.0

Weaver Lite

- You can create up to 5 procedural assets (was Pro-Only).
- Asset Linker can link no more than 10 assets (was unlimited).
- The source code for Kybernetik.Core is now included.

Procedural Assets

- New experimental feature: Auto Generate in Play Mode. Disabled by default.
- New Pro-Only feature: Procedural Asset Collections.
- New feature: Auto Generate on Build.
- Procedural asset generation now takes place in a temporary scene, so any un-parented objects accidentally left in the scene will be properly destroyed (with a log message to let you know).
- Massive performance optimizations to the sub-asset saving system. A large object hierarchy that previously took 4.2 seconds to save its sub assets now takes 0.3 seconds.
- Sub-assets can now be referenced multiple times throughout the asset (instead of only keeping the first reference).
- Improved the way compile errors affect automatic regeneration of the procedural scripts.

Procedural Asset Manager

- You can now generate an asset or group by double clicking it (or middle clicking it, or using the right click context menu).
- Improved the reliability with which asset previews and icons are generated, particularly for assets with sub-assets.
- Improved text colours when using the Unity Dark Skin.

Asset Generator Window

- Fixed a bug during procedural asset generation that would delete objects from the currently open scene when the hierarchy of a prefab changes. Asset generation now takes place in a temporary scene.
- When adding the asset generation time to the end of an asset label, it now also adds the time the generator method itself took to run in brackets,
  I.E. GeneratorMethod() -> AssetPath: timeToGenerateAndSave ms (timeToGenerate).
- When an exception occurs during procedural asset generation, you can choose to Retry, Skip, Abort, or open the source file, and it shows the exception message in the generator window itself (as well as logging it).
- When the system restarts generation to include new dependencies, it now keeps the existing list and adds an entry to indicate where the restart occurred instead of clearing the list entirely.

Asset Linker

- Added menu items "Assets/Add Selection to Asset Linker" and "Assets/Remove Selection from Asset Linker" so you can add and remove asset links without opening the Asset Linker window.
- Added a project window overlay for linked assets. Middle click it to open the Asset Linker.
- Asset Linker initialisation is now split over multiple frames so the editor doesn't lock up when opening the window in a large project.
- Instead of regenerating the Asset Linker script immediately whenever you add/delete/move/rename a linked asset, it will now wait until you click on a different EditorWindow to take focus away from the Project Window. Likewise, if the game is playing, it will wait until you stop playing.
- Improved auto-generate conditions to be more reliable and to include changes to any elements of an asset collection.
- Fixed a bug that prevented procedural assets from being excluded when they are first generated after having their path changed.

Layer Linker

- Added a toggle for all layers at once.
- Added a comment field for each custom mask.
- Added support for layer names starting with numbers (the script constant will simply start with an underscore).
- Improved the layout of the procedural Layers script to be more useful when viewed through intellisense.

User Interface

- Added asset type icons to the buttons in the Procedural Asset Manager and Asset Linker when they're showing previews rather than thumbnails. And also to the type picker for asset collections.
- You can now drag and drop the asset buttons out of the Procedural Asset Manager and Asset Linker windows into the scene or hierarchy to instantiate objects as if you were dragging their assets out of the Project Window.
- Added Weaver settings tab to the Edit/Preferences menu (moved from the Procedural Asset Manager context menu and added some extra options).
- Improved asset icon loading system so that windows will keep repainting until Unity finishes generating previews.

Public API

- Added and improved comments.
- Exposed the sub-asset saving system as Kybernetik.Utils.SaveSubAssets.
- Refactored the Weaver.Asset<T>.Instantiate overloads which take position and/or rotation to turn them into extension methods in Weaver.WeaverUtils. This way they will only be available for GameObjects since it doesn't make sense to use them on other asset types.
- Added new Instantiate overloads which take a parent transform.
- Renamed ProceduralResourceAttribute to AssetModifierAttribute and added a property to disable Play Mode Auto-Generate for a specific asset.
- Changed Kybernetik.Reflection.Assemblies to use partial assembly names for everything. Damn Unity for using two different versions of mscorlib.dll interchangeably.

- Added Kybernetik.Utils.LogErrorIfModified for "locking" a collection which you don't want modified without allocating a new read-only collection. Note that this only logs an error, it doesn't actually prevent modifications.
- Added [assembly: InternalsVisibleTo("Weaver.AssetLinker")] attribute to Weaver.Editor.dll and removed a few public methods that were only used to give the Asset Linker internal access to procedural assets.
- Removed AssetGenerator.IsGeneratorMethodMandatory as it was basically pointless.
- Changed Kybernetik.TypeField into ObjectPickerField and made it generic so it can be used to pick any kind object (not just System.Type).
    - Significantly optimised the GUI code of the picker window so that it no longer lags when searching through a large list.
    - Added support for picking null (if there is a null value in the list).
    - It now sets GUI.changed = true when an object is picked so it can work properly with EditorGUI.BeginChangeCheck.
    - Opening the picker window now automatically scrolls to the selected object.
- Added extension methods Weaver.WeaverUtils.LoadAll and UnloadAll for loading and unloading entire asset collections.

Serialization

- Fixed an issue when changing the procedural script directory which caused it to sometimes fail to move files.
- Fixed the Procedural Asset Manager and Asset Linker to properly save the collapsed state of folders in the asset selector.
- Removed the serialization of some unnecessary fields in the Asset Collection settings.
- Optimised settings file serialization so the imperceptible load times are imperceptibly faster.

Weaver 2.0

- Sub assets are now saved automatically.
- Added context menus to assets and folders in the Procedural Asset Manager and Asset Linker.
    - Added a function to open the source script of a procedural asset.
    - When you try to select or open an asset which doesn't exist in the Procedural Asset Manager or Asset Linker, it will now target the closest existing parent folder.
- Refactored assembly structure.
    - Renamed all assemblies to use the MSDN recommended style: Weaver.AssetLinker instead of Weaver-AssetLinker.
    - Changed Weaver.Runtime.dll and Weaver.Editor.dll to both be called Weaver.dll and have the same GUID to avoid having the assembly name change between runtime and editor code. The runtime DLL is now in a sub-folder called Runtime.
    - Renamed Weaver.Utils into Kybernetik.Utils, split it into separate Runtime and Editor DLLs, and moved them to Assets/Plugins/Kybernetik so they can be shared by various Kybernetik products.
    - Removed reference to System.dll from all assemblies.
    - Moved MeshBuilder and MeshBuilderUtils into Kybernetik.Core.dll.
- Reworked settings files.
    - Instead of using ScriptableObjects, all settings are now serialized using procedural codecs generated by Super Serial.

    - Initialisation code is now much simpler and more reliable because the settings files can be loaded immediately instead of waiting for Unity to deserialize and initialise the ScriptableObjects.
    - The Weaver Procedural Examples are now able to properly add themselves to the Asset Linker on initialisation (previously it did not work if there was an already existing settings file).
    - Deletion of procedural assets when their source files are deleted is much more reliable.
    - Settings files are saved in the root directory of the project (next to the Assets folder) with the ".ss" file extension.
    - Removed the stupid hack that was required to prevent the settings ScriptableObjects from being automatically created when trying to upload to the Asset Store (because including those files would overwrite every user's settings when they upgraded).
    - Removed SerializedType and SerializedField since Super Serial is able to serialize types and fields directly.
    - Removed StaticScriptableObject since the system no longer uses ScriptableObjects.
- Changed the project window overlay for procedural assets to use an icon instead of text and improved the tooltips.
- Greatly improved Layer Linker window GUI.
- Improved comments in the example scripts.
- Improved Weaver Lite warning messages.
- Generating procedural assets no longer requires the scene to be saved. Weaver now cleans up spilled assets itself.
- Added ScriptGenerator.AliasAttribute to change the name procedural scripts to use in their "generated by ..." comment.
- Added UnityConsole class with HasCompileError method for checking if there are any compile errors.
    - The Asset Linker and Layer Linker now check if there are compile errors before automatically regenerating their scripts since such errors would often lead to the scripts being generated incorrectly and causing additional errors.
    - Weaver won't ask if you want to delete missing assets while there are compile errors because such errors often prevented it from finding any assets and led to it asking if you want to delete everything.
- Added ScriptGenerator.SaveMessage so you can add additional details to the log message when a procedural script is generated.
- Removed shared asset caching system as it was actually hindering performance rather than improving it.
- Fixed a bug in PreBuildAttribute which sometimes caused it to trigger methods when pressing play in the editor rather than only when building.
- Renamed AppDetails to BuildDetails and removed its AppName and CompanyName fields (since they are now available at runtime in the Application class).
- Changed usage of StringBuilders to chain text.Append(...).Append(...)... etc in various areas to improve source code readability.
- The Procedural Asset Manager now fits into its window properly when docked. Tip: when creating the GUI for an EditorWindow, use its position property to get its dimensions rather

than Screen.width and Screen.height because the latter values are slightly off when the window gets docked.
- Removing a layer no longer causes an exception when the Layers script is regenerated.
- Fixed a naming error in the Asset Linker relating to assets in the root Assets folder.

Weaver 1.1

- Fixed serialization related errors in Unity 5.4.
- Fixed a bug with Asset Linker Collections that prevented them from being included in the generated script under certain conditions.
- Moved all classes into the Weaver namespace.
- The Asset Linker and Layer Linker give you the choice between the global namespace or Weaver.

Weaver 1.0

- New Asset Linker system.
- Massive changes and improvements in all areas from the Procedural Asset Framework v2.0.

Procedural Asset Framework 2.0

- Upgraded to Unity 5.3.
- Improved the Procedural Asset Manager:
    - Improved the grid selector with asset grouping and multi-selection.
    - Compact mode now shows the selector in a popup window.
    - When generating assets, the full list is shown instead of just the name of the currently generating asset on a progress bar.
    - Type dependencies are now automatically detected, but you still need to add a [TypeDependency] attribute for them to be enforced.
    - Settings are now saved in a ScriptableObject instead of a text file.
    - Old sub assets are now removed more reliably.
- Improved the examples and added a procedural texture and material (WoodTexture and WoodMaterial, used by Barrel).
- Added [PreBuild] attribute for methods to automatically be called when you execute a build.
- Added AppDetails class which automatically numbers and dates builds, and gives runtime access to the product name and company name.
- If you change the path of an asset class, the generated asset is automatically moved to maintain prefab links and avoid leaving clutter in the project.
- If you delete an asset class, a dialog box asks if you want to also delete the generated asset.
- When the game starts (in the editor), it automatically checks for any procedural assets that haven't been generated. If there are any, it asks if you want to generate them. If yes, it stops the game, generates them, and starts the game again.
- Removed the AutoGenerate system for procedural scripts and added a log message when they are generated.
- Added the ability to set a different path for procedural scripts if you don't want them in Resources/Code.
- Removed the procedural Scenes class because the new SceneManager makes it obsolete.
- Source code is included in the Pro version.

Procedural Asset Framework 1.0

Initial release.