UNIVERSE

TECHNICAL DOCUMENTATION (v1.0)

Contents

Introduction

One of the first problem someone encounter while developing with Unity is what could be called "level of persistence". At the root, you have the Unity Player which is never unloaded; it can be seen as being always there. On top of the Player, there's a collection of Unity managers, such as the Sound Manager and the Input Manager. They are also persistent and are never unloaded as long as the game is running.

The developer doesn't have access to that Player layer. Instead, he has two "virtual" levels which are the scenes and the GameObject. You usually load and unload a scene as the context of the game change. As for GameObject, you create and destroy them all the time following the current gameplay.

However, it happens quite often you would need a way to have a fully persistent manager on your own. Sadly, Unity doesn't give you access at creating your own manager the same way they did for their own. This package is our solution to this issue.

Universe

In other game engine we have used, the layers of persistence were separated as followed;

- **GameObject**; The lowest item, with a very short life span.
- **World**; Scene that holds GameObject or other object. Are loaded and unloaded once in a while when the context of the game change. It has a medium life span.
- **Universe**; Is loaded when the game start, and stays as long as the game is running. It has the longest life span.

In Unity, the concept of Universe is not accessible. However, it is possible to build one for our needs. Usually, one will go towards the singleton or the static pattern. Both may work, but they each have their flaws and shortcoming.

A static class will lack any parameters and proper serialization.

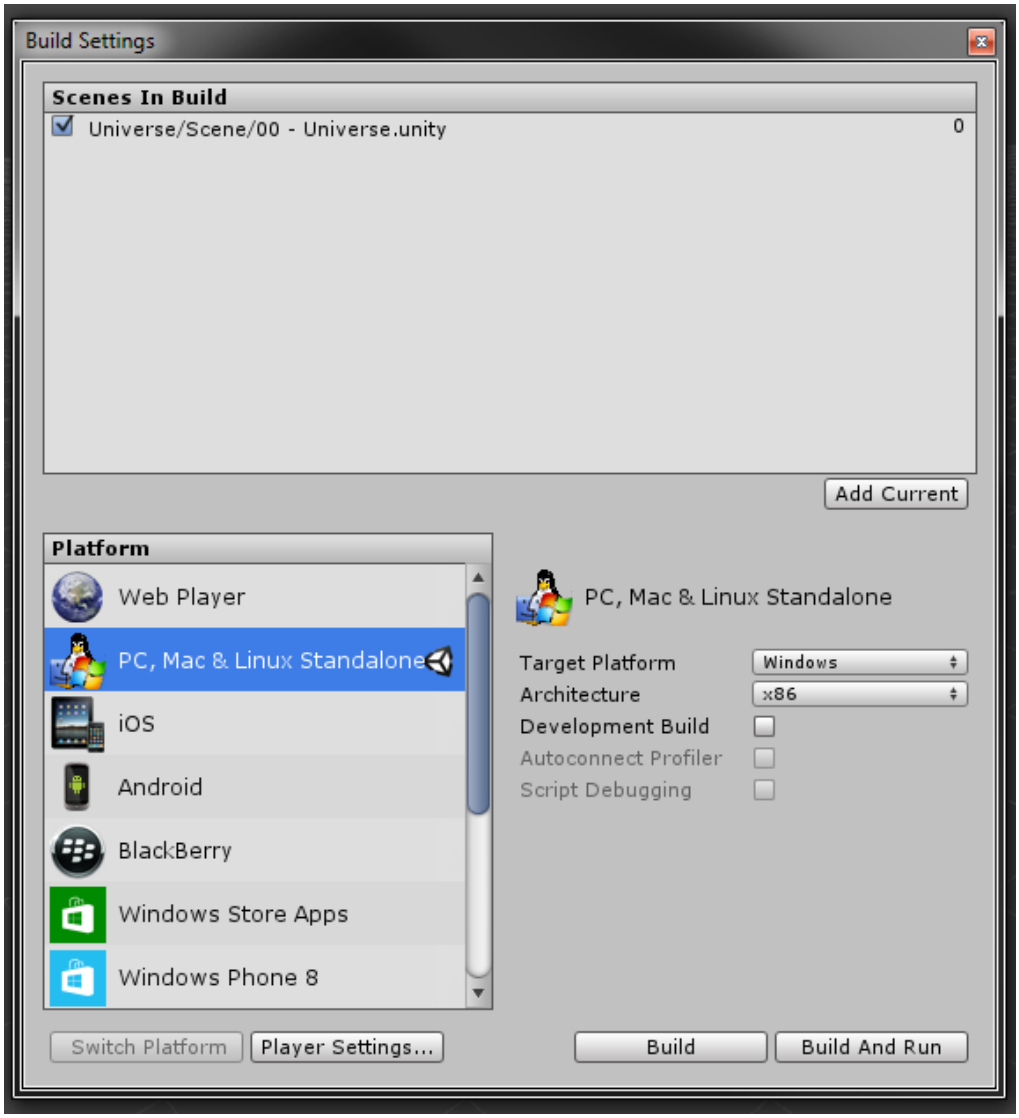
A singleton pattern requires maintenance and to be setup in scene to work properly.

The Universe in this package works with a self-referencing singleton pattern, coupled to a serialization-by-reflection. There's many advantages to this;

- No maintenance; no need to add or update script, GameObject or prefab in all your scene. No need to create any prefab, the package does it for you.
- All Scenes work; create an empty scene, press play, and it works! Managers are automatically loaded.
- Accessibility; Each manager has its own prefab to hold its data in the Resource folder. They are all at the same place, and easy to edit.
- Message; a static class cannot receive external Message, which is a must for platform like iOS/Android. Being on a prefab, every manager is ready to receive your messages.
- Auto-implementation; the singleton pattern is automatically implemented, you don't need to do anything.

How To Use

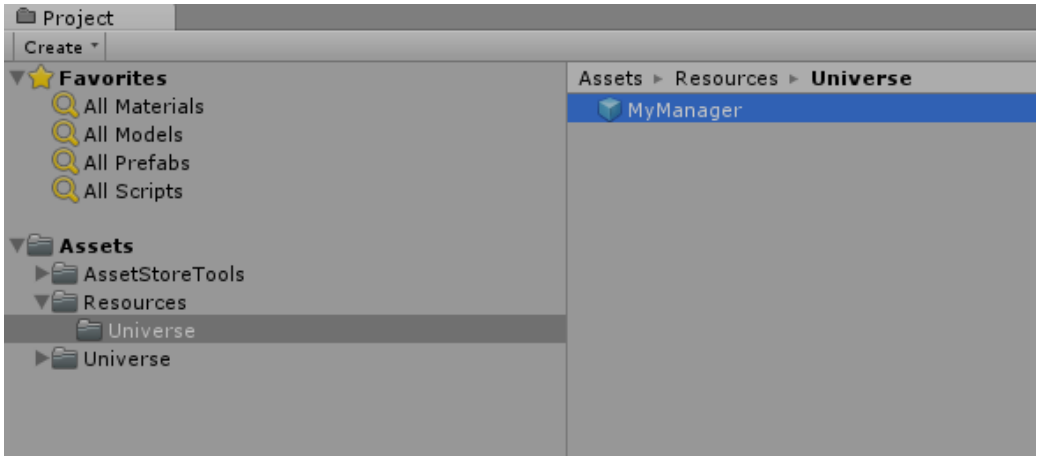
Using the Universe package is extremely easy. First, you need to set the first loaded scene of your build as being "00 - Universe", a scene that is provided in this package. This scene has a single GameObject with the Universe script on it. Its only task is to load all the prefabs of all the managers and keep them alive.



The second step is simply to have your manager you wish to exist for the lifespan of the whole game to derive from the Manager<> class;

```
public class MyManager : Manager<MyManager>
{
    public float myValue;
}
```

The next time you press Play in Unity editor, a new prefab will be created;



This prefab is named the same way as your manager class. Please do no rename it, otherwise a new prefab will be created.

That's it! All your managers will always be loaded, no matter which scene you use, in the editor or in your build. Added a new manager is as easy as creating the class and pressing play.

Contact

While the Universe is a fairly simple tool, it surely still has room for improvement. To send us any bug report or feature request, contact us at;

Email : admin@lightstrikersoftware.com

Website : www.lightstrikersoftware.com

Package Revision

1.0

- Release