ZSerializer

Quick Start Guide

Contents

Setup

When you first install ZSerializer you will be greeted by this big Setup button. When you click it, it'll generate the necessary files you need to start using the tool.

There are several things you can serialize using this tool:

Serializing Components

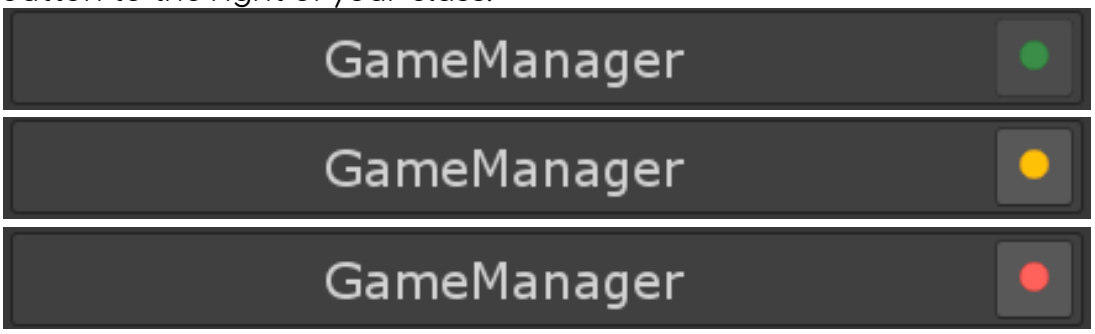
Let's say you have a component you want to serialize, for example, a Game Manager that holds information about the state of the game:

```
public class GameManager : MonoBehaviour
{
    public int highScore;
    public int currentScore;
    public string playerName;
    public Vector3 playerPosition;
}
```

If you add make your class inherit from ***PersistentMonoBehaviour*** (you will need to add the ZSerializer namespace for it to be available), it will be recognized as a ***Persistent Component***.

Whenever you go back to Unity, on the ZSerializer Menu, you'll see the name of your class appear on that menu. This means your class is now recognized as Persistent.

Depending on your settings, you might see either a green, yellow, or red button to the right of your class.

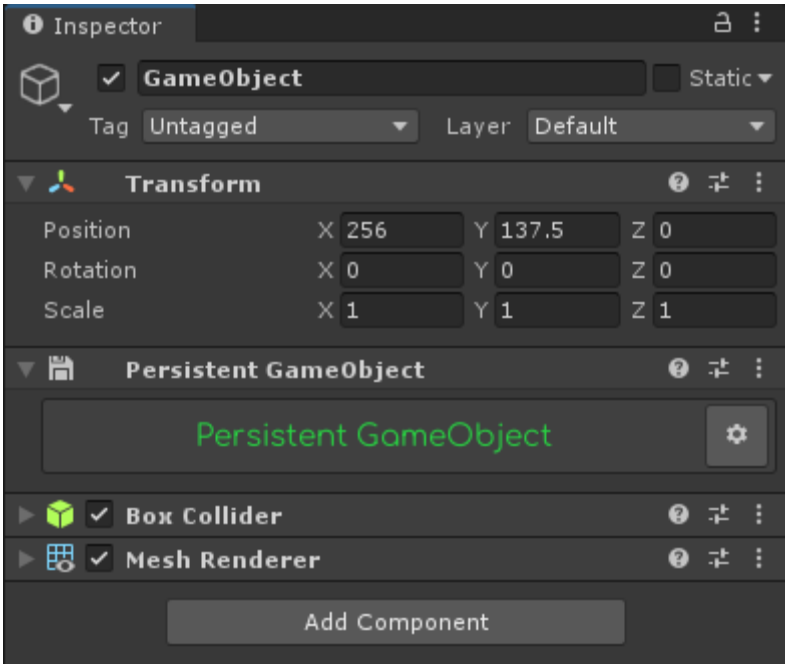


- Green means everything's fine, the class instances will be saved and loaded when prompted.
- Yellow means the files needed for serialization have been created, but they need to be rebuilt, because the class itself was changed.
- Red means there's no files and they need to be built.

Clicking the button in either Yellow or Red will fix these problems.

Serializing GameObjects

Making GameObjects serializable is even easier than componentsl, just add the ***Persistent GameObject*** component to your gameObject, and it will be saved and loaded when prompted.



The ***Persistent GameObject*** component will save the GameObject's name, tag, layer, static state, transform, and each and every one of the Unity Components that are attached to it.

(That includes Colliders, Renderers, and everything else under the UnityEngine namespace.)

You'll see a cogwheel icon to the right of where it says "***Persistent GameObject***" this will display a menu with every component that's currently attached to the GameObject, and it will give you the option to Serialize it or not.

Saving and Loading.

This part is also pretty easy, you just need to call:

```
ZSave.SaveAll();
ZSave.LoadAll();
```

ZSave.SaveAll();

1. It will serialize every instance of a ***Persistent Component*** you have on your scene.
2. It will serialize every instance of a ***Persistent GameObject*** along with its selected Unity Components and ***Persistent Components***.
3. (Optional) It will encrypt your data using the Rijndael Algorithm.
4. It will save your data based on which Save File you've selected and which scene you're currently in when performing the Save.

ZSave.LoadAll();

1. It will deserialize every ***Persistent GameObject***, and instantiate new ones in case they're not already in the scene, preserving all of the gameObject's data along with their hierarchy.
2. It will deserialize every one of Unity's Components and add them into their respective Game Objects.
3. It will deserialize every one of your custom ***Persistent Components*** and add them into their respective Game Objects
4. It will restore the references and values that each component had at the time of saving.

It will however NOT:

1. Restore a custom ***Persistent Component***'s GameObject if said GameObject doesn't have a ***Persistent GameObject*** Component attached to it.
2. Restore an instance of a Material, Mesh and similar assets that have been modified from the original.

OnSave and OnLoad Events

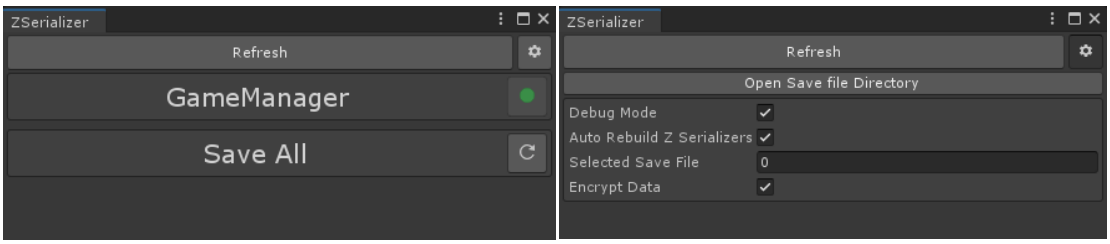
If you want to execute some code right before or after a Save or a Load, you can override any of the following methods like this:

```
public override void OnPreSave()
{
    //Your code goes here...
}

public override void OnPostSave()
{
    //Your code goes here...
}
```

```
public override void OnPreLoad()
{
    //Your code goes here...
}

public override void OnPostLoad()
{
    //Your code goes here...
}
```

The ZSerializer Menu

This menu contains information about your current project's ***Persistent Components*** as well as some settings.



The first panel will show you which classes have been marked as ***Persistent*** in your Project.

When you press on the cogwheel to the right, you'll see the Settings menu:

Open Save File Directory

When you click this button, the folder in which save files will be created will open, and you will be able to explore it.

The file structure of this folder will be the following:

<SelectedSaveFileIndex>/<CurrentSceneBuildIndex>/GameManager.save

Debug Mode

This will dump information into the console whenever a save or a load occurs, useful for debugging in case something breaks.

Auto Rebuild ZSerializers

This will rebuild the ZSerializers of your Persistent Components, so that you don't have to click that filthy yellow or red button ever again.

Selected Save File

The selected Save File in which the saving and loading will be done. There can technically be any amount of save files you want, although it may take a toll on disk space on very large games if it's not used correctly, so limiting it is a good idea.

Encrypt Data

Determines whether or not your data will be encrypted using the Rijndael Algorithm.