



ALL PLATFORMS SAVE



WHY DO YOU NEED TO USE THIS PLUGIN ?



Easy to use: same line of code to save or load game data on all supported Unity platforms.



Game data can be saved using PlayerPrefs or external files.



Works on all Unity platforms without making any changes to the code.



Supports multiple save files.





CURRENTLY SUPPORTED ADVERTISERS



Jsonserializationfilesave

serializes data using built-in JSON serializer and saves the result into an external file



JSONSerializationPlayerPrefs

serializes data using built-in JSON serializer and saves the result as a string using PlayerPrefs.



BinarySerializationFileSave

serializes data using BinaryFormatter and saves the result into an external file.



BinarySerializationPlayerPrefs

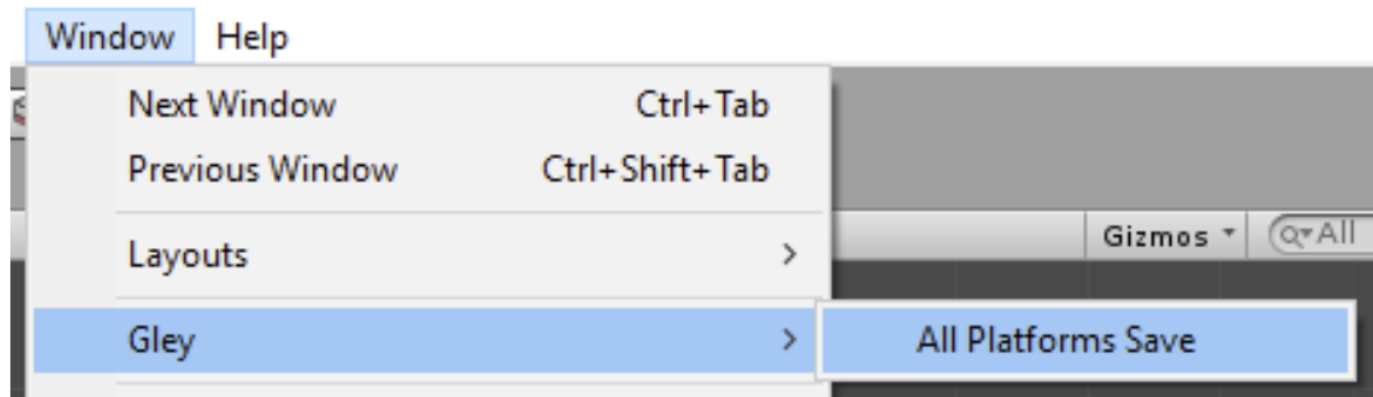
serializes data using the BinaryFormatter and saves the result as a string using PlayerPrefs



SETUP GUIDE

Import **GleySavePlugin** into Unity.

Go to **Window->Gley->All Platforms Save** to open the plugin settings window.



Click Add Build Target button and select all platforms you want to build for and for each set the desired save method.

Configure your save plugin from here:

Select your build target:

Select save method:

Select your build target:

Select save method:

Select your build target:

Select save method:

To remove the save support for a platform click the Remove Build Target button.

Make sure your current build target is added to Save Settings in order to work in Unity Editor. (ex: If Unity editor is set to Android you must add Android as one of your build targets in the Save Settings).

After all configurations are done press **Save** button to apply the settings.





USER GUIDE

Load

`SaveManager.Instance.Load<T> (fullPath, DataWasLoaded,encrypt);`

T -> a class in which saved data will be deserialized into.

fullPath-> full path to the file location.

DataWasLoaded -> method called when load process is done.

encrypt -> if true, data will be decrypted using an XOR algorithm.

//this method will be called after load process is done

```
private void DataWasLoaded(T data, SaveResult result, string message)
```

```
{
```

```
if (result == SaveResult.Success)
```

```
{
```

```
// do something with your data
```

```
}
```

```
}
```

data -> actual loaded data.

result -> Success/Error

message -> error message

Save

```
SaveManager.Instance.Save(T, fullPath, DataWasSaved, encrypt);
```

T -> an instance of any class marked as Serializable.

fullPath -> full path to the file location.

DataWasSaved -> method called when save process is done.

encrypt -> if true, data will be encrypted using an XOR algorithm.

```
//this method will be called when save process is complete  
private void DataWasSaved(SaveResult result, string message)  
{  
    if (result == SaveResult.Error)  
        //Do Something  
}
```

result -> Succes/Error

message -> error message

Clear specific save file

```
//clears the specified file name at path  
SaveManager.Instance.ClearFile(fullPath);
```

Clear all save files

```
//clears all files found at path  
SaveManager.Instance.ClearAllData(path);
```





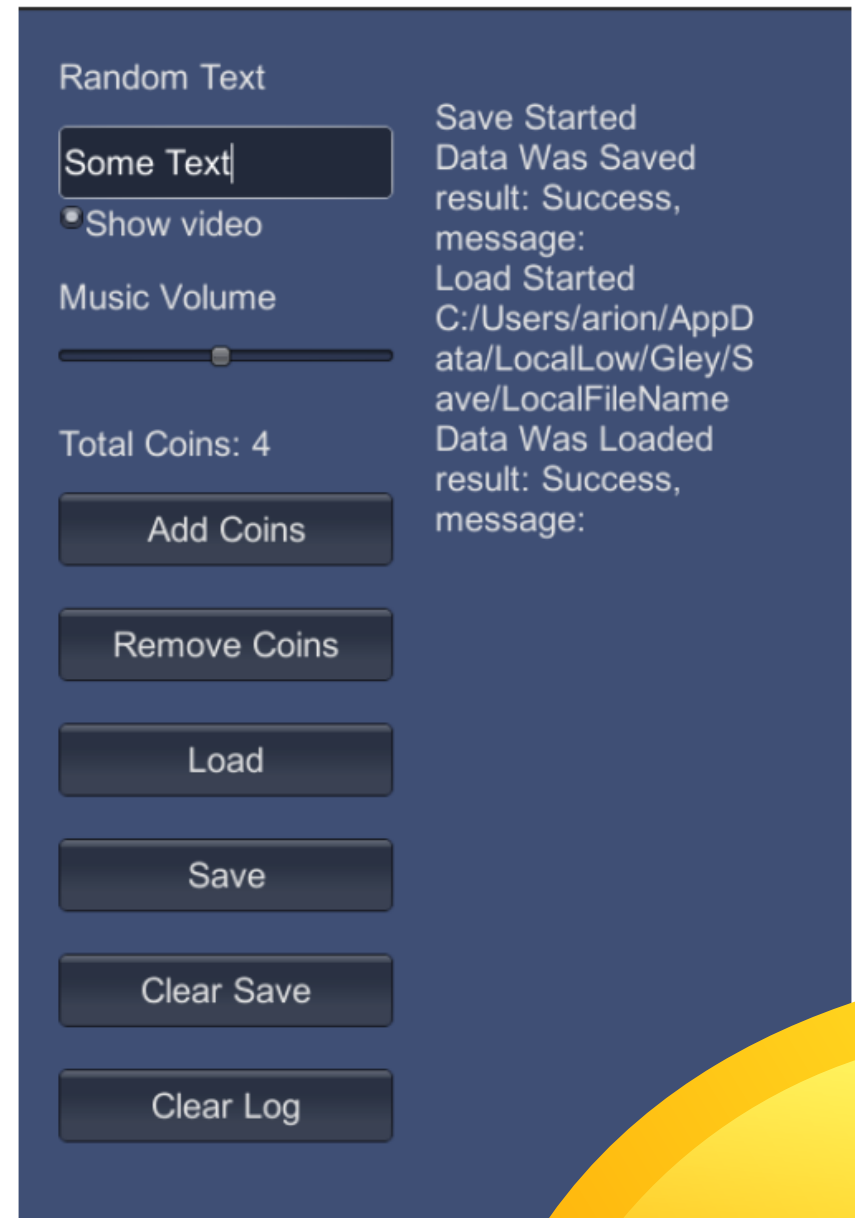
EXAMPLES

Simple Save Example

Can be opened from here:

Assets/GleyPlugins/Save/Example/Scenes/Simple-SaveExample.unity

Or by pressing Open Test Scene in settings window



This scene saves 4 properties:

```
public bool showVideo = true;  
public int totalCoins = 4;  
public float musicVolume = 0.5f;  
public string randomText = "Some Text";
```

Using the buttons, those properties can be changed, then saved and loaded. Also a log message appears on screen.

Any class marked as Serializable can be saved.

Variables need to be public

Any object of a serializable class can be serialized(saved).

Ex:

```
[System.Serializable]  
public class GameValues  
{  
    public double version=0;  
    public bool showVideo = true;  
    public int totalCoins = 0;  
    public float musicVolume = 1;  
    public string randomText = "Random Text";  
    public List<Level> levels = new List<Level>();  
}
```

Be aware that some Unity objects are not serializable like Transform, GameObject, Color, Vector3, etc. You have to use some standard type to serialize them:

Ex:

To serialize a Color the following class can be used

[System.Serializable]

public class SerializableColor

{

 public float r;

 public float g;

 public float b;

 public float a;

}





Version 1.0 / 2017