List Serializer

Copyright 2015 Pigasus Games
Version 1.0.0
petrucio@pigasusgames.com

Thank you for buying the List Serializer! It let's you export lists of items from the inspector into XML files in order to make them easier to edit outside of Unity, and import them back into your objects.

Getting Started

Seting up:

- Select the object containing the script that you want to export a list of things (strings, vectors, ...) in the inspector.
- Add the script ListSerializer to it
- Select the list base type (string, Vector3D, ...) in the "List Type" dropdown
- Add these basic export and import methods to the script that contains the target list:

```
public class YourAwesomeClass : MonoBehaviour {
	public string[] yourListOfStrings;

	void ListImport(string[] values) {
		this. yourListOfStrings = values;
	}
	void ListExport(Pigasus.ListSerializer serializer) {
		serializer.Export<string>(this.yourListOfStrings);
	}
}
```

(Replace this.yourListOfStrings with the name of the list you want to serialize, and <string> with the base type of your list. You can also change the name of the Import and Export methods if you change their default values in the Options inspector fold)

- Add the [ExecuteInEditMode] marker before the definition of the class that holds your list:

```
[ExecuteInEditMode]
public class YourAwesomeClass : MonoBehaviour {
	//...
```

Using:

- Click on "Export Values Now" to save the values to an XML file. (The default name is NameOfParentObject_ListSerializer.xml, saved on the project root. You can override this is the Options)

- Edit the values of your list using an external text edito

- Click on "Import Values Now" to load the values from the XML file into the target list.

Can I export things other than strings, vectors, and numbers?

Yes, you can easily expand it to export your own data types, as long as they do not inherit from IEnumerable (like references to GameObjects or Transforms).

Simply open the ListSerializer.cs script, and add a new entry into the ListType enum, and it's Import switch. The proper places hae already been marked with an "`// Add custom types here:`" comment.

This is what it will look like on the ListSerializer.cs script:

```
public enum ListType {
     // ...
     // Add custom types here:
     MyType,        // Name this whatever you want
}

public void Import()
{
     switch (this.listType)
     {
     // ...
     // Add custom types here:
     case ListType.MyType: this.Import<MyType>(); break;
     // ...
     }
}
```

Setting up and using it is the same as covered in the previous section. Here's an example usage of a custom type:

```
[System.Serializable]
public class MyType {
     public string title;
     public float  a_float;
     public Vector3[] values;
}
public MyType[] myTypeList;   // Horrible to edit in inspector

void MyTypeImport(MyType[] values) {
     this.myTypeList = values;
}
void MyTypeExport(Pigasus.ListSerializer serializer) {
     serializer.Export<MyType>(this.myTypeList);
}
```