Welcome To Dictator v1.0

Forewords

Unity3d has great integration for most common class types in the inspector, except for Dictionary (AKA: Map).

If you are reading this, it means you already understand the frustration with Unity's Dictionary support, they don't show up in the inspector and they don't serialize

While there are many plugins out there that claims to solve this problem, most of them are flawed in one of the following ways

1. Hard to setup - many solution requires you to write another class and put it in a specific folder to enable the inspector, that is too troublesome
2. Not a real Dictionary - many solutions uses a list of a Struct to imitate Dictionary. unfortunately, one of the property of Dictionary is that keys must be unique, this solution does not guarantee that. Furthermore these solution requires you to learn it's own API, as it's not really a Dictionary
3. Limited - many solutions has limitations, such as:
    - you cannot use GameObject as key
    - No nesting is allowed
4. Ugly looking - many solutions that will not display properly when using complex types or nesting

And that is why I created the Dictator, this should be the easiest, least intrusive and good looking Dictionary solution on the market right now.

Features

1. Custom Dictionary in inspector in 2 lines of code

   You don't need to write a separate class to enable inspector, just subclass Dictator with [Serializable] property, that's it!
2. Almost the same API as native Dictionary, you don't need to learn any new API
3. Same performance as native Dictionary, it uses native Dictionary under the hood
4. Serialize and De-serialize
5. Compact property drawer
6. Slider to balance the display size between key and value
7. Drag and drop to reorder entries
8. Nesting is supported, only selected entry will show nested elements to save display space
9. Smartly handles duplicate keys

Installation

Move the Dictator folder to your unity assets folder, it is recommended to place it under "Plugins" directory

Usage

Define a custom Dictionary

```
[Serializable]
public class MyCustomDict : Dict<type, type>{}
```

If `[Serializable]` is showing an error, add `using System;` to the top of your file OR change it to `[System.Serializable]`

Unity Inspector

- clicking on variable name to show or hide the content
- the slider controls the display ratio between key and value, the position is saved
- click the "+" button to add an entry
- if an entry turned red, then it is invalid in runtime, however your other values will still be present in runtime, it is usually because
  - there are duplicate keys
  - null value for Object type keys
- to delete an entry, click on the "=" to select it, then click the "-" button
- drag an entry by the "=" to reorder

Note: invalid entries are still serialized and deserialized, but they will not exist in runtime, only in the editor

Using Dictator in Script

Please see Dictionary API at MSDN

Notable usages are:

loops

```
foreach(KeyValuePair<type, type> item in MyDict)
{
    item.Key;//do stuff with key
    item.Value;//do stuff with value
}
```

or for loop with LINQ

```
using System.Linq;

for(var i = 0; i < MyDict.Count; i++)
{
    MyDict.ElementAt(i);//do stuff with the keyvaluepair
}
```

Get Set and Remove

```
MyDict["StringKey"];//get the corresponding value by key
MyDict.ContainsValue(val);//return true if the dictionary contains that value
MyDict.ContainsKey(key);//return true if the dictionarry contains that key

float fvalue;
if(MyDict.TryGetValue(key, out fvalue))
{
    //try to get the value by key
    //do something with fvalue
}

//adding key value pairs to dict
MyDict.Add(key, val);
MyDict.Add(new KeyValuePair<type, type>(key, val));

MyDict.Remove(key);
```

Note: when adding key value pair to Dictionary, if it already contains that key, then an error will be thrown. check to see if a key exists before adding

Using LINQ expressions

Dictator supports LINQ just like native Dictionary

for example

```
var dupeValues =    from x in MyDict.Values
                    group x by x into grouped
                    where grouped.Count() > 1
                    select grouped.key;
```

Support

email me at wuhao@astrokoala.com