RandomUtils

Set of utils for make work with random selections more comfortable.

Includes Randomizer and Weighted List.

Features

Randomizer

- Non-repetitions random.
- Flat-distributed random.

Weighted List

- Allow to set weights for list items, and then select randomly by weight.
- Inherits IList and IReadOnlyList interface - you can use it as usual List.
- Property drawer included to easy setup in the inspector.

Randomizer API

To use Randomizer, you need to create and store its instance:

```
_randomizer = new Randomizer(_prefabsList.Count);
```

It needs to be initialized with amount of items to select from. Randomizer is designed to work with arrays and lists, so, it's only returns integer indexes, limited in range from 0 to initialized amount.

To get indexes without repetitions, use next:

```
int index = _randomizer.SelectNoRepeat();
```

To get indexes flat-distributed:

```
int index = _randomizer.SelectFlatDistributed();
```

Both functions returns integer number in range from 0 to amount value, which constructor was initialized with.

WeightedList API

WeightedList<T> inherits standard C# interfaces IList<T> and IReadOnlyList<T>. Internally, it consists of the usual List<T>, and list of associated weights (List<float>). So, you can use it in your code exactly like List<T>, except you have to add weights, when its necessary:

Constructor (also parameter-less constructor available): WeightedList(List<T> objects, List<float> weights)

- int IndexOf(float weight) - get the first index of item with given weight.
- int IndexOf(T item) - get the first index of the item.
- void Insert(int index, T item) - inserts item at index with 0 weight.
- void Insert(int index, T item, float weight) - inserts item at index with given weight.
- void Add(T item, float weight)- adds item to the end of list with given weight.
- void Add(T item) - adds item to the end of list with 0 weight.

To keep usage safe, Weighted List implements read-only interface:

```
public interface IReadOnlyWeightedList<T>
  {
    T GetRandomByWeight();
    float GetWeightAtIndex(int index);
    float GetTotalWeight();
    float GetNormalizedWeightAtIndex(int index);
  }
```

Lets describe it:

- T GetRandomByWeight() - it's a main routine to use: it randomly selects an item from a list by its weight.
- float GetWeightAtIndex(int index) - allows you to know the weight at particular index.
- float GetTotalWeight() - returns the sum of all weights.
- float GetNormalizedWeightAtIndex(int index) - Normalized weights means the weight value, proportional to absolute value, but when the total weights sum equals 1. For example, we have two items, weights are 2 and 6. Normalized weights for these items are 0.25 and 0.75.

Additionally, WeightedList<T> provides next methods:

- void SetWeightAtIndex(int index, float weight) - sets weight for item with given index.
- void Normalize() - normalizes all weights (total weights sum will be 1).
- void SetWeightOf(T item, float weight) - sets weight for particular item (first occurance).

Weighted List property drawer

To make it comfortable to use Weighted List in unity, custom property drawer provided.

Usage:

1. Inherit concrete implementation of generic WeightedList (dont forget the Serializable attribute):

```
[Serializable] public class WeightedListOfPrefabs : WeightedList<CustomPrefab> { }
```

2. Inherit property drawer and mark it with attribute:

```
[CustomPropertyDrawer(typeof(WeightedListOfPrefabs))]
public class MyPropertyDrawer : WeightedListPropertyDrawer { }
```

Provided property drawer supports any basic type, and any type, inherited from UnityEngine.Object. Plain c# classes supported as read-only (but no limits to use plain classes in code).

Demo video

https://youtu.be/0Irz4ShEJbs

Contact

Mail: bryarey@gmail.com

Skype: m.khadzhynov

Support me on Patreon: https://www.patreon.com/user?u=26313020&fan_landing=true

Subscribe on Games Garden channel on YouTube:
https://www.youtube.com/channel/UCH6WybnFgT199Kkd6vm7-Lg

LinkedIn: https://www.linkedin.com/in/mykhaylo-khadzhynov-15635915/

Games Garden web site: https://www.gamesgarden.net/