Array help sheet

Array type comparison

Array type	Resizable? *Can I make array bigger/smaller?*	Mixed variable types? *Can I mix different variable types?*	Search method *How can i get values out?*	Appears in Unity? *Can I set values in the Editor?*
Built-in	no*	no	index-based	yes
ArrayList	yes	yes	index-based	no
Hashtable	yes	yes	key-value pairs	no
List	yes	no	index-based	yes
Dictionary	yes	no	key-value pairs	no

*can be resized using the “System.Array.Resize” method to a specific size

---

Explanation of terms

Resizable

The array can have space added for extra values, or space removed making the array size smaller.
The Built-in array type will not add more values past its set size, unless the array is re-created with a new set size.
This re-creation of the array is encapsulated in the method “System.Array.Resize” which will resize the array to a new pre-determined size.

Mixed variable types

Some arrays allow different types of variables to be added as values.
For example, an ArrayList can have an int as one value and a Vector3 as another value:

```
ArrayList myArrayList = new ArrayList();                // create the ArrayList
myArrayList.Add( 5 );                                   // add an integer value
myArrayList.Add( new Vector3( 0.0f, 0.0f, 10.0f ));     // add a Vector3 value
```

Search method

The way we can search for values in the array.
There are 2 ways:

- index-based
- key-value pairs

index-based

Each value is sorted by having an index, or number given to it.
You can find a value by giving the array the number (contained in square brackets), and it will return the value:

```
int[] intArray = new int[2];            // create the built-in array
intArray[0] = 5;                        // add an integer value, note the use of square brackets for the index
intArray[1] = 53254;                    // add another value at the next index
print( intArray[0] );                   // get the value from the array using the index
```

key/value pairs

Each array value is split into a key and a value pair:
array.Add(KEY, VALUE);
The key is mainly used for searching, while the value can be searched too.
Unlike the index based array, the key can be other types of variable, like strings.
This can be useful for naming array values to make them more visible in your code:

```
Hashtable hashtable = new Hashtable();                          // create the built-in array
hashtable.Add( "position", new Vector3( 10.0f, 5.0f, 0.0f ));   // add a key/value
hashtable.Add( 5.7f, 100 );                                     // add another key/value of different types
print( hashtable[5.7f] );                                       // get the value from the array using the key
```

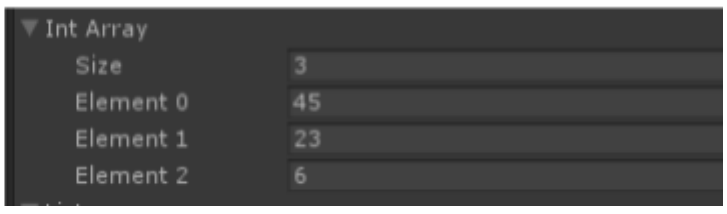
Appears in Unity

Some array types can be added to in the Unity Editor.
The built-in type is the most common.
The built-in array can be resized from the Unity Editor:

only 2 values allowed in the script

```
public int[] intArray = new int[2];
```

Resizable in the editor



Built-in array

The most basic type of array used in Unity.
They only accept one type of variable per array (e.g. an array of ONLY int or ONLY float values)
They are of a fixed size when created.
They can be resized using the “System.Array.Resize” method.
Built-in arrays are the fastest performing type of array, because they have a fixed size and only contain one type of variable

Syntax

Declaring

Most variable types and components in Unity can be created as an array.
Create a variable and add “[]” square brackets after declaring the type to turn the variable into an array

```
float[] myFloatArray;
```

Creating

Next, create the array using the “new” operator and declare the size of the array in the square brackets
This float array has a size of 5

```
float[] myFloatArray = new float[5];
```

Referring

To enter a value for the array, you must use its index to refer to it
This array refers to the first value in the “MyFloatArray” using the square brackets.
The number “0” is the first index in the array

```
myFloatArray[0]
```

Setting

To set the value, simply use the “=” operator like any other variable

```
myFloatArray[0] = 0.5f;
```

Getting the size

To see how big the array is, use the “Length” property
The “Length” property will always be an int

```
int arrayLength = myFloatArray.Length;
```

Setting the size

To set a new size for the array, use the “System.Array.Resize” method.
Remember to use the “ref” modifier before entering the array
The array has now been resized to 25 values

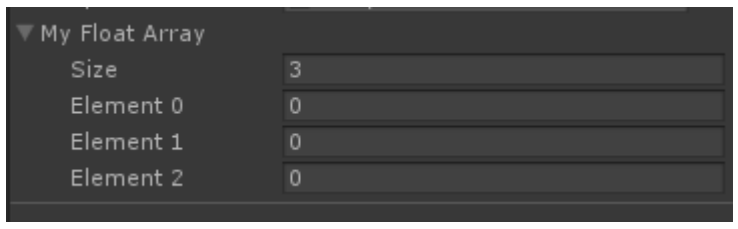
```
System.Array.Resize( ref intArray, 25 );
```

In the Unity Editor

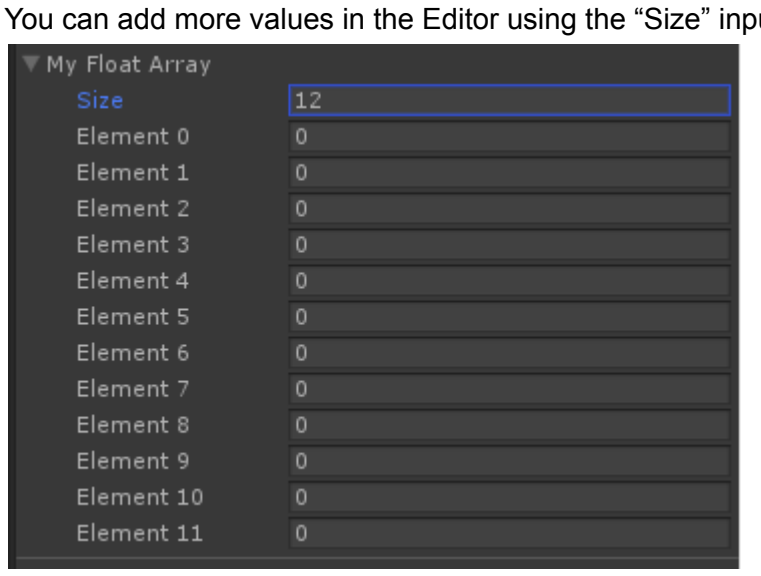
If you create an array with a size of 3 like so:

```
public float[] myFloatArray = new float[3];
```

The Unity Editor will display space for 3 values:



You can add more values in the Editor using the “Size” input field.



Changing the size in the Unity Editor will not show in the code.

Some useful methods

Syntax	Returns	Description
array[int index]	value	returns the value at the index
array.Length	int	get the size of the array
System.Array.Resize(ref array, int new size)	nothing	resizes the array

---

ArrayList

An array that allows mixed variable types
You can add and remove new values easily
You cannot edit values in the Unity Editor

Syntax

Declaring

Most variable types and components in Unity can be used in an ArrayList
Declare the ArrayList as follows

```
ArrayList myArrayList;
```

Creating

Use the “new” operator like other variables

```
ArrayList myArrayList = new ArrayList();
```

Referring

As long as it contains a value at the requested index, you can refer to a value in the ArrayList like so:
The index being the number in the square brackets

```
myArrayList[0]
```

Setting

To enter a new value, use the “Add” method on the ArrayList
This will add another value to the end of the ArrayList

```
myArrayList.Add( “a value” );
```

Getting the size

ArrayLists use the “Count” property to get their size

```
int arrayListLength = myArrayList.Count;
```

Setting the size

ArrayLists set their own size according to the values contained.

In the Unity Editor

ArrayLists are not available to set in the Unity Editor without some scripting techniques outside the mandate of this help sheet

Some useful methods

Syntax	Returns	Description
ArrayList[int index]	value	returns the value at the index
ArrayList.Count	int	gets the size the the ArrayList
ArrayList.Add(object value)	nothing	adds a new value to the end of the ArrayList
ArrayList.RemoveAt(int index)	nothing	removes the value at the index the ArrayList will resize itself to “close the gap”

Hashtable

A resizable array that stores key/value pairs
Mixed value types are allowed
You cannot edit values in the Unity Editor

Syntax

Declaring

Most variable types and components in Unity can be used in a Hashtable
Declare the Hashtable as follows

```
Hashtable hashtable;
```

Creating

Use the “new” operator as follows

```
Hashtable hashtable = new Hashtable();
```

Referring

If a Hashtable value were created with a string key and an int value
The key would be how to refer to the value

```
hashtable.Add( "health", 100 );           // create a key/value pair
int healthvalue = hashtable["health"];    // get the value (100) using the key “health”
```

Setting

Use the “Add” method to add a value to the hashtable using the syntax:
Hashtable.Add(object KEY, object value);

```
hashtable.Add( "health", 100 );           // create key/value pair
```

Getting the size

Hashtables use the “Count” property to get their size

```
int length = hashtable.Count;
```

Setting the size

Hashtables set their own size according to the values contained.

In the Unity Editor

Hashtables are not available to set in the Unity Editor without some scripting techniques outside the mandate of this help sheet

Some useful methods

Syntax	Returns	Description
Hashtable[object key]	object	returns the value at the specified key
Hashtable.Count	int	gets the size of the Hashtable
Hashtable.Add(object key, object value)	nothing	creates a new value in the Hashtable
Hashtable.Remove(object key)	nothing	removes a value specified by the key

---

List

Also known as a “Generic List”
Lists are resizable
In order to use a List, add the following to the top of your C# class file:

```
using System.Collections.Generic;
```

They only accept one type of variable per array (e.g. an array of ONLY int or ONLY float values)
Lists are index-based
Lists can be adjusted in the Unity Editor

Syntax

Declaring

Most variable types and components in Unity can be used in a List
Declare the List as follows

```
List<int> listArray;
```

Note the angle brackets “<>” surrounding the type of variable

Creating

Use the “new” operator as follows

```
List<int> listArray = new List<int>();
```

note the angle brackets are also used after the new operator, along with normal brackets

Getting the size

Lists use the “Count” property to get their size

```
int length = listArray.Count;
```

Setting the size

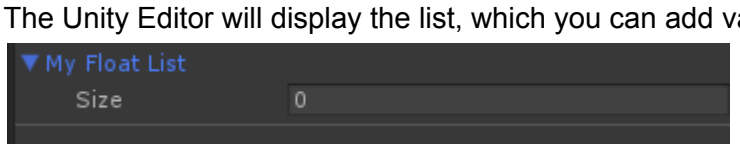
Lists set their own size according to the values contained.

In the Unity Editor

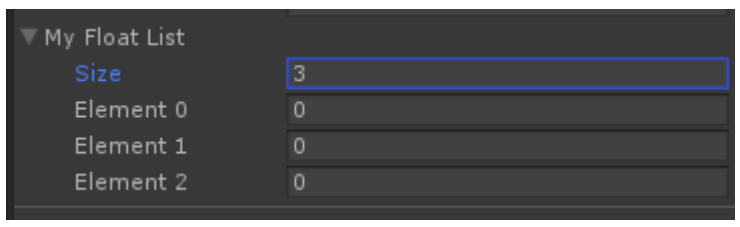
If you create a List as a public property like so:

```
public List<float> myFloatList = new List<float>();
```

The Unity Editor will display the list, which you can add values to



Like so:



Changing the size in the Unity Editor will not show in the code.

Some useful methods

Syntax	Returns	Description
List[int index]	object	returns the value at the specified key
List.Count	int	gets the size of the List
List.Add(object value)	nothing	creates a new value in the List
Hashtable.RemoveAt(int index)	nothing	removes a value specified by the index

---

Dictionary

A resizable array that stores key/value pairs
They only accept one type of variable per array (e.g. an array of ONLY int or ONLY float values)
In order to use a Dictionary, add the following to the top of your C# class file:

```
using System.Collections.Generic;
```

You cannot edit values in the Unity Editor

Syntax

Declaring

Most variable types and components in Unity can be used in a Dictionary
Dictionaries declare a KEY and VALUE like so:
Dictionary<KEY, VALUE>
Declare the Dictionary as follows

```
Dictionary<string, Vector3> myDictionary;
```

note the angle brackets surrounding the key and value types

Creating

Use the “new” operator as follows

```
Dictionary<string, Vector3> myDictionary = new Dictionary<string, Vector3>();
```

note the angle brackets are also used after the new operator, along with normal brackets

Getting the size

Dictionaries use the “Count” property to get their size

```
int length = myDictionary.Count;
```

Setting the size

Dictionaries set their own size according to the values contained.

In the Unity Editor

Dictionaries are not available to set in the Unity Editor without some scripting techniques outside the mandate of this help sheet

Some useful methods

Syntax	Returns	Description
Dictionary[object key]	object	returns the value at the specified key
Dictionary.Count	int	gets the size of the Dictionary
Dictionary.Add(object key, object value)	nothing	creates a new value in the Dictionary
Dictionary.Remove(object key)	nothing	removes a value specified by the key

---

Useful links

Arrays - Video
ArrayList - MSDN Manual
Hashtable - MSDN Manual
Lists and Dictionaries - Video
Lists - MSDN Manual
Dictionaries - MSDN Manual