Serialised Data Structures

By The Black Cat

Content

Read Before Everything!!

This is very important. When you are using the serialised data structures, if you want to store lists,
DO NOT!

What you want to do is to write a **wrapper** for the list and store the **wrapper** instead.

NO!

```
public SerializedQueue<List<int>> queuedLists;
```

YES~

```
public SerializedQueue<ListWrapper> queuedLists;

[System.Serializable] ← Remember this line, I always forget
1 個參考
public class ListWrapper
{
    public List<int> list;
}
```

Serialised Queue

Instead of **Queue<>**, you can use the **SerializedQueue<>** in the namespace **TheBlackCat.SerialisedDS** to serialise your queue.

```
public SerializedQueue<string> queuedNames;
```

After that, you will be able to view the queue in the inspector.

← Next Dequeue

Elements are enqueued to the bottom and dequeued from the top, in this direction

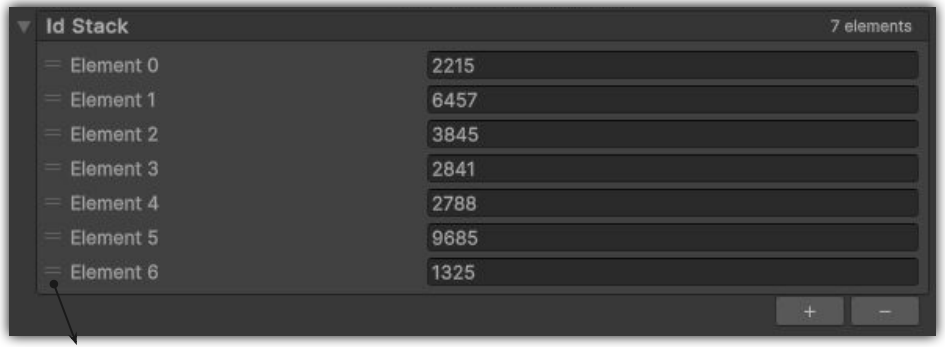
← Last Enqueued

Serialised Stack

Instead of **Stack<>**, you can use the **SerializedStack<>** in the namespace **TheBlackCat.SerialisedDS** to serialise your stack.

```
public SerializedStack<int> idStack;
```

After that, you will be able to view the stack in the inspector:



Will become undraggable in Play Mode

Elements are pushed to the bottom and popped from there.

← Last Pushed, Next Pop

Serialised Priority Queue - Declaration

I found that I still can't use the Priority Queue in unity, so I also implemented one myself. You can use the **SerializedPriorityQueue<,>** in the namespace **TheBlackCat.SerialisedDS** where the 1st parameter is the element and the 2nd parameter is the priority.

```
public SerializedPriorityQueue<char, int> charPriorityQueue;
```

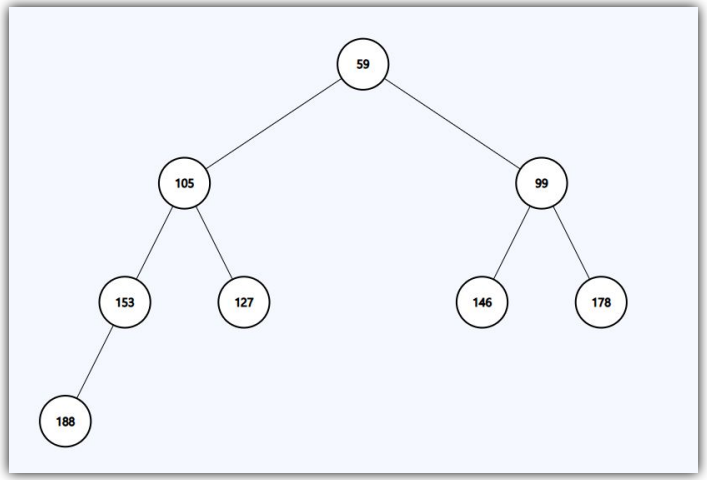
Here, the char is the element and the int determines the priority.

*The priority queue can only be modified by code. You can only view it in the inspector.
*The priority queue makes use of the min-heap data structure, the elements are **NOT** sorted.

Serialised Priority Queue - Example

← Added elements in code, elements are sorted by priorities

Visualisation of the queue below ↓



Serialised Priority Queue - Comparer

You can assign a comparer when creating a priority queue. This is useful when you are using custom class objects for the priority and you want custom comparing behaviour.

I wrote a custom comparer that inverts the comparison, the **InverseComparer<>**. The parameter is the type you use for the priority.

To use the inverse comparer, you can instantiate a new comparer:

```
public SerializedPriorityQueue<char, int> charPriorityQueue = new (new InverseComparer<int>());
```

Since this priority queue is a **min-heap**, you can use this comparer to invert the min-comparison if you want a **max-heap** instead. You can pass your comparer into the inverse comparer so it inverts your custom comparer or call **Invert()** directly from the comparer.

```
public SerializedPriorityQueue<char, int> charPriorityQueue = new (your_comparer.Invert());
```

Serialised Tuples

It's tricky to serialise tuples, so I created structs that act like tuples.

These are the serialised tuples available:

Pair<,> for 2 items
Triplet<,,> for 3 items
Quartet<,,,> for 4 items
Quintet<,,,,> for 5 items

*If you've got more than 5 items in a tuple…I recommend using a custom class instead.

Serialised Tuples - Example

Say you want to declare a tuple with 3 items, you can use Triplet:

```
public Triplet<string, int, bool> tupleExample = Triplet<string, int, bool>.Of("haha", 36, true);
```

*Use **Of()**, not **new()**.

And it will be displayed in the inspector

If you need to access them, use **tupleExample.Item1/item2/item3/etc.** just like how you would access items in a normal tuple.

Serialised Tuples - Display Names

If you want custom display names, you can use the **SerializedTuple** attribute. Pass in the display names for every item in the tuple (the number of arguments must be exactly the same as the number of items).

```
[SerializedTuple("I have", "No", "Idea")]
public Triplet<string, int, bool> tupleExample = Triplet<string, int, bool>.Of("haha", 36, true);
```

And the display names will be change →

*Only works for **globally** declared tuples.

Huge thanks for noticing my assets, huger thanks for even reading until here.

This asset is not perfect. Any suggestions, bug reports, reviews and advice will be highly appreciated.

If you have any problems, please contect me through:
Email: heinokchow314@gmail.com
Github: https://github.com/The-best-cat/Unity_asset-Serialised_data_structures

Good luck with your projects, and have a nice day.