Quicksort v1.6

Quicksort implementation for Operation Flashpoint: Resistance; by Faguss (ofp-faguss.com)

1. Overview

Quicksort is a widely used sorting algorithm. This package contains its implementation in the **Operation Flashpoint**'s / **ArmA: Cold War Assault**'s scripting language. It will order an array containing numbers (from the lowest to the highest) or strings (alphabetically).

2. Usage

Copy `quicksort.sqf` to the mission directory. Initialize function by adding the following line to the `Init.sqs`:

```
call preProcessFile "quicksort.sqf"
```

Function call syntax:

```
[<array to sort>, <optional parallel array>, <...>] call quicksort
```

It does not return anything. The original array is modified.

Example:

```
array = [5, 4, 1, 3, 2]
[array] call quicksort
hint Format ["%1", array]
```

Game will display:

```
[1, 2, 3, 4, 5]
```

Optionally you can pass any number of arrays (containing any type of data) and they will be reordered in the same manner as the first array. All arrays must be of the same size.

Example:

```
points = [100   , 10    , 15    , 0      , 50    ]
names  = ["Josh", "John", "Paul", "Jones", "Dave"]
[points, names] call quicksort
hint Format ["%1 %2", points, names]
```

Game will display:

```
[0,10,15,50,100] ["Jones","John","Paul","Dave","Josh"]
```

This function can also sort strings alphabetically. Requires Fwatch version 1.15 or newer. It automatically detects if input array contains strings or numbers. Case insensitive.

Example:

```
names  = ["Josh", "John", "Paul", "Jones", "Dave"]
points = [100   , 10    , 15    , 0      , 50    ]
[names, points] call quicksort
hint Format ["%1 %2", names, points]
```

Game will display:

```
["Dave","John","Jones","Josh","Paul"] [50,10,0,100,15]
```

When sorting alphabetically optionally you can pass a string at the end of the input array, that contains extra arguments.

Example:

```
[names, points, "caseSensitive:true natural:true reverseCase:true"]
call quicksort
```

Option caseSensitive will enable case sensitivity so that lowercase will appear first in the list. If you add `reverseCase` to it then capital letters will be given priority. Argument `natural` is for sorting strings that have numbers in them.

3. SQS variant

This package also contains `quicksort.sqs` which is a script variant. It works slower than `quicksort.sqf` but the game is not frozen while it's working and it's possible to abort it so it's more suitable for large arrays (tens of thousands of items).

Example:

```
IS_QUICKSORT_WORKING = true
[points, names, "IS_QUICKSORT_WORKING"] exec "quicksort.sqs"
@!IS_QUICKSORT_WORKING
```

Global variable has to be defined which indicates when the script has finished working. Its name has to be passed to the script (as the last item).

All other options (sorting multiple arrays, sorting alphabetically, string sorting options) are the same.

Script can be modified to display progress and check for a condition to quit (see `#Compare` label).

This implementation was based on https://www.geeksforgeeks.org/iterative-quicks-sort/

3. Version history

1.0 (08.09.10)

First release.

1.1 (02.09.11)

- added alphabetical sorting variants
- variant *quicksort2* renamed to *quicksort_MULTI*

1.2 (24.07.12)

- in alphabetical sorting you may pass extra param for case sensitivity (Fwatch 1.11)

1.3 (27.05.13)

- all local variables are now private
- function does not return anything
- global variable `QUICKSORT_RECURRENCE` is now used
- letter case in argument "`casesensitive`" does not matter
- not required anymore to pass array size numbers in the call to function

1.4 (01.06.16)

- `quicksortA` and `quicksortAM` use latest Fwatch 1.15
- `quicksortA` and `quicksortAM` syntax change – must pass array with string as a last item

1.5 (22.02.22)

Separate variants have been merged into a single a file with a unified usage:

- multi variant is default
- automatically detects if input is a number or a string
- it’s not required to pass string at the end when alphabetizing
- function variable is set when calling the file instead of setting it in the `Init.sqs`
- uses a series of functions `quicksort`, `quicksort_loop`, `quicksortABC_loop` instead of a global variable `QUICKSORT_RECURRENCE`
- works slightly faster

1.6 (27.09.22)

- added SQS variant
- function is executed only if input array size is larger than one (instead of larger than zero)