

MOHAA Script Framework

(By: \$or)

Admittedly, I've been all talk lately, so I thought it's time to come out with some results. I've spent the last few weeks rewriting what I've had so far. At first, I thought the framework ought to provide common mods and all the features most modders would want. Things like a teambalancer, spawn protection.. and an event handler, maths library etc...

Before long, the low-level stuff for the basic libraries like array, string, maths... was going slowly. It's tedious, boring stuff and it made me realize that I hadn't been programming all those years, I had been scripting.

Luckily, I have started a programming course at college this year and I really got into programming! I've been researching a lot on datastructures, algorithms and design patterns in my free time. Also, the ongoing tedious work and screw-up gained me a lot of experience



I rewrote a majority of what I had and completely finished it. It went pretty smoothly. If it weren't for the blanks it'd be a finished product.

Concretely

The framework will simply be a set of libraries and available functionality. It, by itself, doesn't do anything. You have to use it. It's structured as a hierarchy of classes. The lower you go, the more specific the functionality. A 'class' is simply an object with function pointers and/or properties.

There are also 'enums', there not much different from classes except that they only contain properties (camelCase) and constants (CAPCASE). Classes on the other hand primarily consist of function pointers. All pointers point to threads in the source file that the class represents.

Usage

The entire framework will be initialized before the map's script file is executed by the engine.

The class objects and their structural hierarchy are kept globally in a game variable.

To access their functionality, you can consult a property or execute a function (PascalCase):

Code:

```
local.words = waitthread game.System.String.Split "hello world"  
println (local.words[1] + " " + local.words[2]) //prints 'hello world'
```

That's a lot of typing! In addition, there are some classes that may not be fully initialized yet. Such classes may perform background work or need to utilize framework resources in order to allow the user access to all its functionality. At the moment there are none such classes.

Include system

To 'finalize' their initialization or access the class via a shorthand, you have to 'include' them. It's like C or C++'s 'include' or C#'s 'using' but with a twist. The twist essentially boils down to "if you include a class, you also include all of its children classes and their children class and..."

Thus, by that logic, including the root class 'System', should include each class and make the framework fully functional. Overlapping includes (from multiple mods, for instance,) are handled.

Code:

```
main:
thread game.Include game.System
//or, alternatively:
//game.System thread game.Include
level waittill prespawn
local.words = waitthread $String.Split "hello world"
println (local.words[1] + " " + local.words[2]) //prints 'hello world'
//..
end
```

You should be able to include classes as soon as the map script is executed. If you include a class before 'level waittill prespawn' however, the inclusion won't take effect until after prespawn. In this case, though, the function will return early with a boolean that indicates whether or not the inclusion will take place.

If called at any other time, it should be called with 'waitthread' as the class won't be included until after Include() returns.

Code:

```
main:
level waittill spawn
local.arr = "pears"::"oranges"::"plums"::"grapes"::"apples"::"peaches"::"lemons"::"apricots"

waitthread game.Include game.System.Array

waitthread $Array.Sort local.arr 1
waitthread $Array.Print local.arr
/* prints
"[1] = plums
[2] = pears
[3] = peaches
[4] = oranges
[5] = lemons
[6] = grapes
[7] = apricots
[8] = apples"
*/
end
```

In order to view the structure of the entire framework, just do:

Code:

```
waitthread game.System.Ls  
// or  
// waitthread $System.Ls
```

and check out the console.

If you see an asterisk next to a component's name, that means it is an enum and not a class.

Classes & Enums

At the moment we have

- 5 finished classes: 'Maths', 'String', 'Char', 'Array', 'Time';
- 5 enums: 'Int', 'Float', 'SystemObject', 'VarType', 'ArrayType'
- and classes 'Common' and 'Map' which have working functionality

I will incrementally update the framework with finished classes and fixed bugs only.

All scripts written with it now will have 100% forward compatibility so any compatibility problem is simply a matter of updating to the latest version.

I haven't had time to write user-friendly and extensive documentation but the source codes of the aforementioned classes and enums have all functions sufficiently and concisely documented.

If, by any chance, a function fails for you, try setting level.DEBUG to 1. It is a global variable that allows most framework functions to intercept and/or print debug on common errors.

Future

Thanks to the very near **exam period** (sorry, used bold to remind myself of that fact), expect the next update near end of June or early July. I expect the entire IO part to be finished, being the Path, File, Directory and IO classes, and the much wanted 'standard event handler' as the Event class which will be particularly flexible.

I know the juicy stuff is missing, but apart from the i/o, all the boring stuff is done and over now while laying a solid groundwork for the things I'm well-versed at: the mohaa stuff. Things will start going more quickly, though, I suspect the HUD and map location logic will be an annoying pain but no real problem.

When the framework is in its final state, it will be included, with other fixes and content to the reborn_pak8.pk3 file and released officially. This released file should override all previous framework versions installed unless the file name has gained a few more z's (obviously).

Download

Just pop the pk3 in the main/ directory. This doesn't interfere with dmprecache or anything else.

[zzzzzzzzzzzz ScriptFramework 0822BETA.pk3](#)

The latest reborn path must be installed, i.e. Reborn RC3.5.116 (yes, the test version).
If you have any question feel free to ask!

P.S.: Git bash and I aren't getting along right now, so I'm not able to update the repository with the source code.