WoG 3.59 Engine Features

by Jakub Grzana

Links

- Project on Github: ORIGINAL MY-VERSION
- ERM/Lua Help: ORIGINAL MY-VERSION
- How to build project: CLICK
- Alpha Releases: ORIGINAL

Introduction

This document is dedicated for modders: explains most important features of WoG as Engine rather than standalone expansion. Content features will NOT be explained here.

Version 3.59 was meant to be not only mod, but platform that was meant to host many different mods – pretty much like Era II. It is opensource and written in C, C++, Lua and Assembler. Alpha 8.0 released by GrayFace is very stable and definitely shouldn't be so marginalised. Let's start presentation of new possibilities.

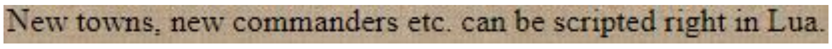
Additionally, purpose of this documemt isn't detailed explanation of each feature, but short description of all, so if you're after modding tutorial – again, wrong address.

Installation

There is no proper installation app, cause it's still alpha.
- Install WoG 3.58f
- Make sure Data folder doesn't contain these files: ARTEVENT.TXT, ARTRAITS.TXT, CRANIM.txt, CRTRAITS.TXT, SpTraits.txt, ZCRTRAIT.TXT, ZELP.TXT
- Unpack WoG 3.59 release archive inside main Heroes 3 folder.

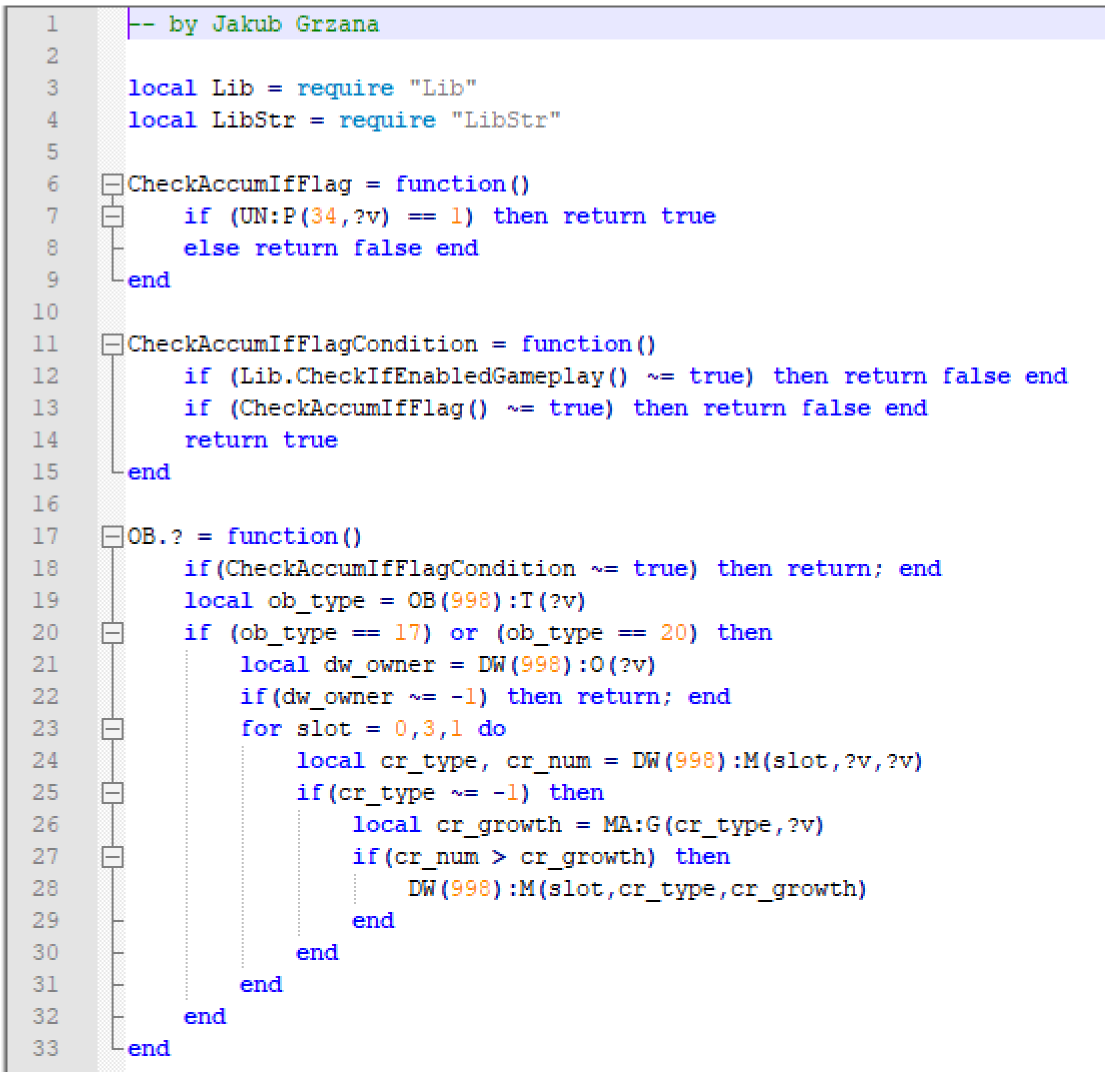
Lua scripting language

Despite vicious rumors, Lua language introduced in WoG 3.59 **is NOT** “just a thin wrapper” for ERM. Lua scripts are part of engine and grants low-level modding tools, like memory edit or hooks. Mod Support *(that will be explained later)*, new dialog system, “WoG Options” dialog, support for new campaigns and *(unfinished)* new town support are written in Lua. According to GrayFace itself:

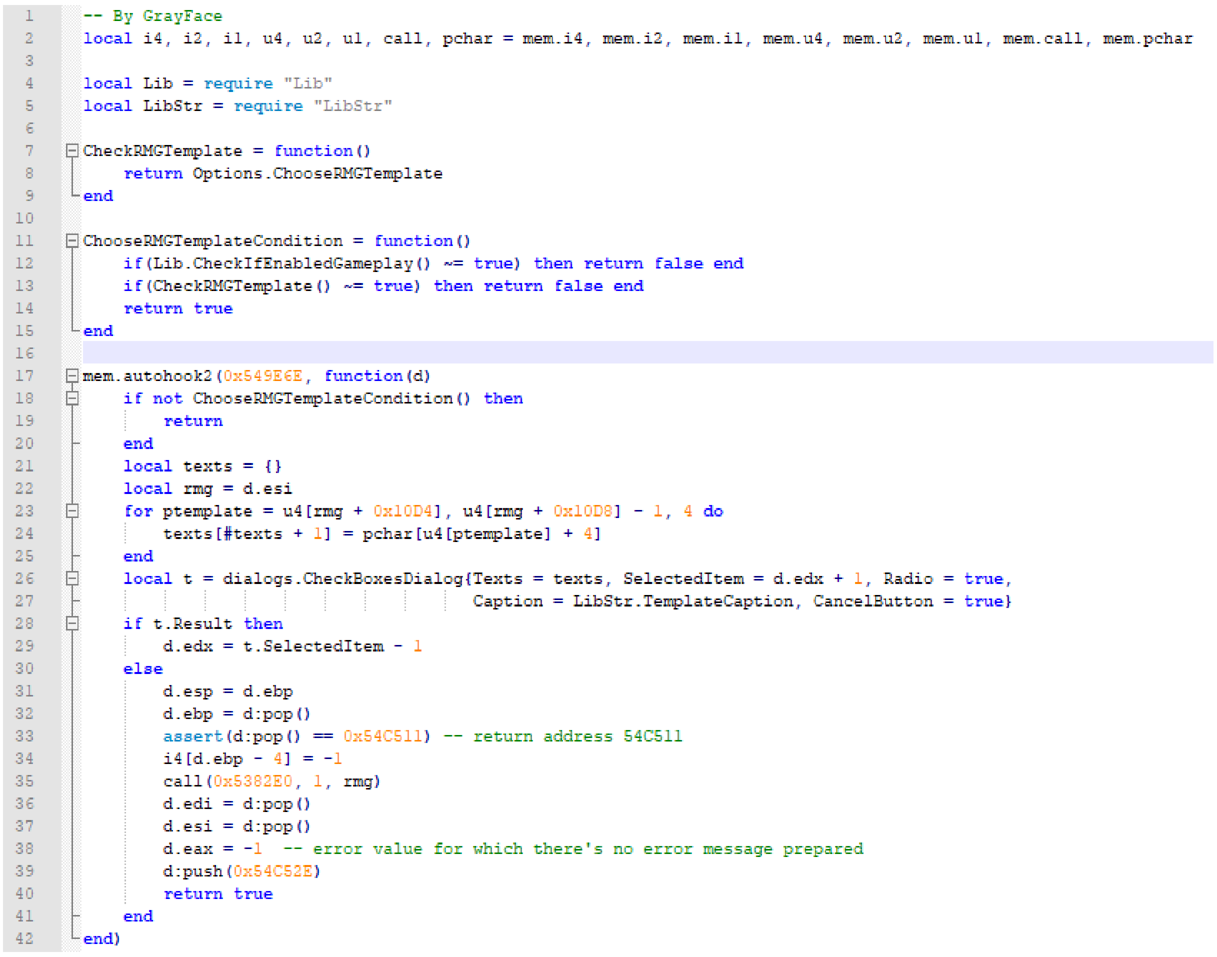

New towns, new commanders etc. can be scripted right in Lua.

Although it **IS** truth that all ERM receivers and triggers are accessible from Lua, so you can use it as wrapper.

This isn’t place to discuss Lua syntax, I will focus on unique possibilities it grants. First, Lua allows you to use **named variables and functions** with various arguments. Scripts and their environment (including variables) are stored in savefile, so effectively there is **no limit of resources** and **no risk of reusing resource**, you can just declare your own vars and functions at will. Furthermore, syntax itself is much **easier to read** than ERM. Here is example of *“dwelling accum cr only if flagged”* script:

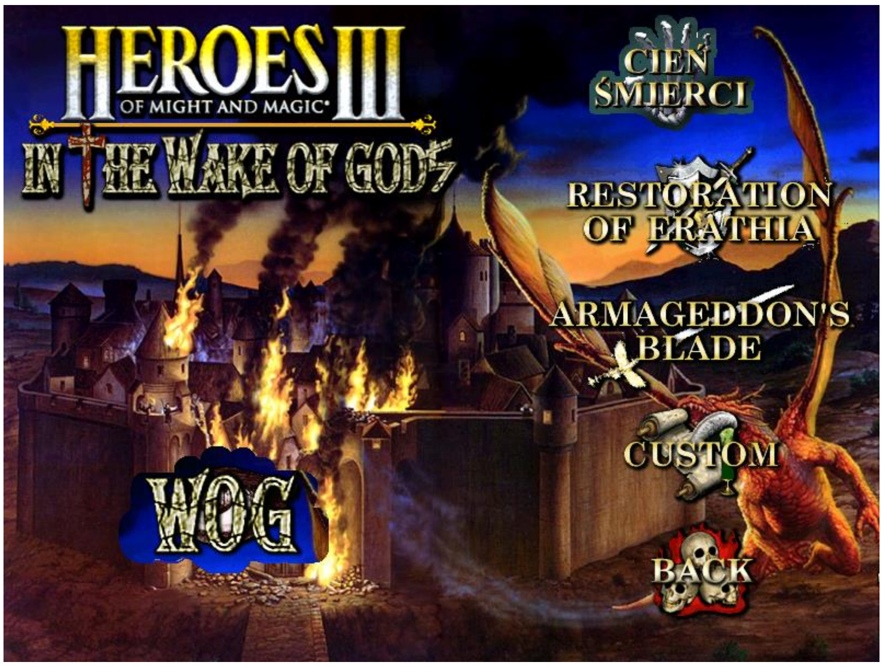

```lua
-- by Jakub Grzana

local Lib = require "Lib"
local LibStr = require "LibStr"

CheckAccumIfFlag = function()
    if (UN:P(34,?v) == 1) then return true
    else return false end
end

CheckAccumIfFlagCondition = function()
    if (Lib.CheckIfEnabledGameplay() ~= true) then return false end
    if (CheckAccumIfFlag() ~= true) then return false end
    return true
end

OB.? = function()
    if(CheckAccumIfFlagCondition ~= true) then return; end
    local ob_type = OB(998):T(?v)
    if (ob_type == 17) or (ob_type == 20) then
        local dw_owner = DW(998):O(?v)
        if(dw_owner ~= -1) then return; end
        for slot = 0,3,1 do
            local cr_type, cr_num = DW(998):M(slot,?v,?v)
            if(cr_type ~= -1) then
                local cr_growth = MA:G(cr_type,?v)
                if(cr_num > cr_growth) then
                    DW(998):M(slot,cr_type,cr_growth)
                end
            end
        end
    end
end
```

If you are into low-level programming, you can use **mem** global variable. It stores **reference to memory** of game and allows **using hooks** - intercepting function calls. Here is example of *“choose RMG template”* script.



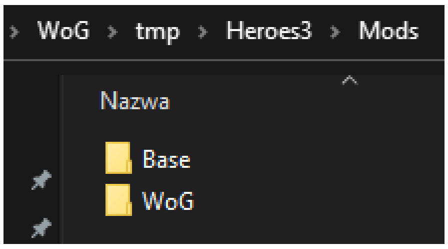
```lua
-- By GrayFace
local i4, i2, i1, u4, u2, u1, call, pchar = mem.i4, mem.i2, mem.i1, mem.u4, mem.u2, mem.u1, mem.call, mem.pchar

local Lib = require "Lib"
local LibStr = require "LibStr"

CheckRMGTemplate = function()
    return Options.ChooseRMGTemplate
end

ChooseRMGTemplateCondition = function()
    if(Lib.CheckIfEnabledGameplay() ~= true) then return false end
    if(CheckRMGTemplate() ~= true) then return false end
    return true
end

mem.autohook2(0x549E6E, function(d)
    if not ChooseRMGTemplateCondition() then
        return
    end
    local texts = {}
    local rmg = d.esi
    for ptemplate = u4[rmg + 0x10D4], u4[rmg + 0x10D8] - 1, 4 do
        texts[#texts + 1] = pchar[u4[ptemplate] + 4]
    end
    local t = dialogs.CheckBoxesDialog{Texts = texts, SelectedItem = d.edx + 1, Radio = true,
                                       Caption = LibStr.TemplateCaption, CancelButton = true}
    if t.Result then
        d.edx = t.SelectedItem - 1
    else
        d.esp = d.ebp
        d.ebp = d:pop()
        assert(d:pop() == 0x54C511) -- return address 54C511
        i4[d.ebp - 4] = -1
        call(0x5382E0, 1, rmg)
        d.edi = d:pop()
        d.esi = d:pop()
        d.eax = -1  -- error value for which there's no error message prepared
        d:push(0x54C52E)
        return true
    end
end)
```

Some Lua scripts are executed not on map setup *(like in ERM)* but on GAME setup, so you can modify game behaviour in main menu. For example, add new buttons or new campaign.



Mod support

While WoG 3.58f was pretty good as mod-hosting platform, 3.59 is designed for that purpose and mod support is much more advanced here. No installation process is required – mods are just folders that you put in /Mods/ catalog:



› WoG › tmp › Heroes3 › Mods

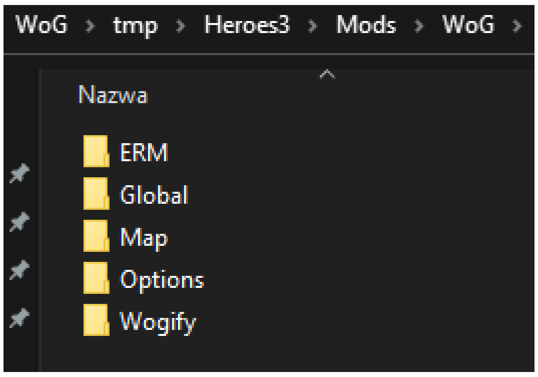
Nazwa

Base

WoG

By default, Alpha 8.0 offers WoG mod containing all external scripts, and Base mod containing many cool scripts in Lua made by GrayFace. Like I said, we won't discuss content.

Inside mod folder, there might be 5 folders with automatically executed scripts



WoG › tmp › Heroes3 › Mods › WoG ›

Nazwa

ERM

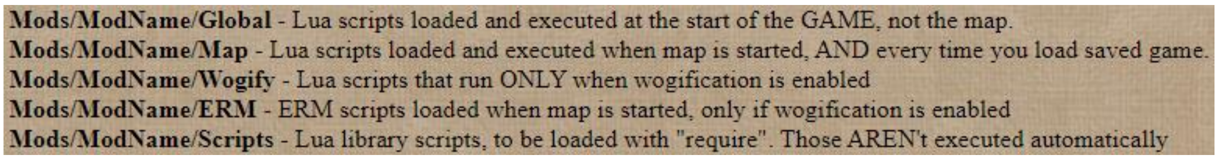
Global

Map

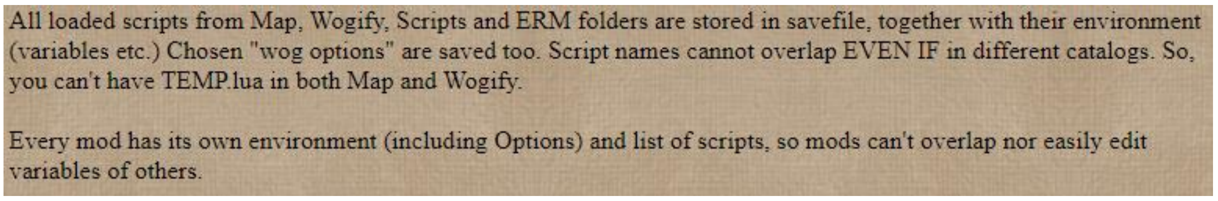
Options

Wogify

Every folder can contain different scripts. Their purpose is explain in ERM/Lua Help. Note that scripts from "Global" folder are NOT stored in savefile.



Mods/ModName/Global - Lua scripts loaded and executed at the start of the GAME, not the map.
Mods/ModName/Map - Lua scripts loaded and executed when map is started, AND every time you load saved game.
Mods/ModName/Wogify - Lua scripts that run ONLY when wogification is enabled
Mods/ModName/ERM - ERM scripts loaded when map is started, only if wogification is enabled
Mods/ModName/Scripts - Lua library scripts, to be loaded with "require". Those AREN't executed automatically



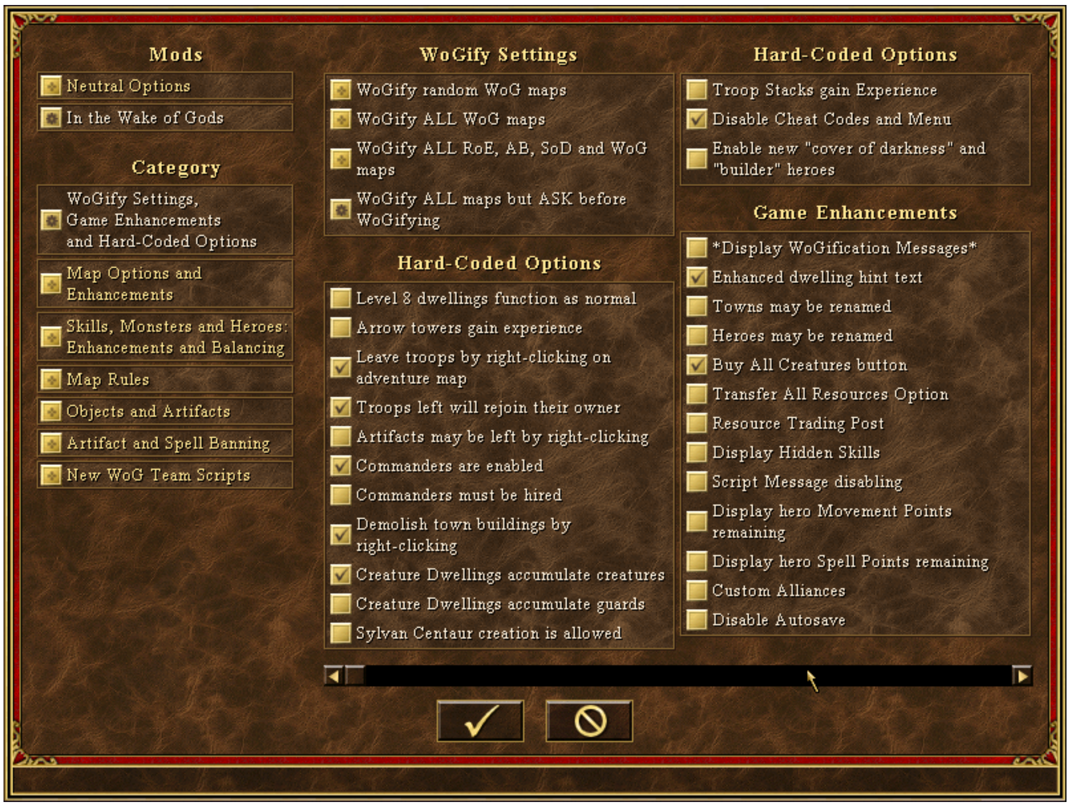
All loaded scripts from Map, Wogify, Scripts and ERM folders are stored in savefile, together with their environment (variables etc.) Chosen "wog options" are saved too. Script names cannot overlap EVEN IF in different catalogs. So, you can't have TEMP.lua in both Map and Wogify.

Every mod has its own environment (including Options) and list of scripts, so mods can't overlap nor easily edit variables of others.

Scripts from "Scripts" folder aren't automatically executed. Instead you can manually execute them with "require" keyword, and assign their environment to local variable. Usually it is used to make Library scripts, and localization support. *Note there is proper localization support by GrayFace – but isn't documented and I don't know how to use it.*

Scripts from "Global" folder are executed at GAME start. You can't use ERM here cause ERM parser doesn't work in main menu. Inside those scripts you might add new *campaigns (like WoG campaign – there might be more!)* new buttons for main menu and so on.

That’s not everything though – there is “Options” folder. This is where you keep entry for “WoG options” dialog, similar to old ZSETUP.txt. In game, it looks like this:



Mods

- Neutral Options
- In the Wake of Gods

Category

- WoGify Settings, Game Enhancements and Hard-Coded Options
- Map Options and Enhancements
- Skills, Monsters and Heroes: Enhancements and Balancing
- Map Rules
- Objects and Artifacts
- Artifact and Spell Banning
- New WoG Team Scripts

WoGify Settings

- WoGify random WoG maps
- WoGify ALL WoG maps
- WoGify ALL RoE, AB, SoD and WoG maps
- WoGify ALL maps but ASK before WoGifying

Hard-Coded Options

- ☐ Level 8 dwellings function as normal
- ☐ Arrow towers gain experience
- ✓ Leave troops by right-clicking on adventure map
- ✓ Troops left will rejoin their owner
- ☐ Artifacts may be left by right-clicking
- ✓ Commanders are enabled
- ☐ Commanders must be hired
- ✓ Demolish town buildings by right-clicking
- ✓ Creature Dwellings accumulate creatures
- ☐ Creature Dwellings accumulate guards
- ☐ Sylvan Centaur creation is allowed

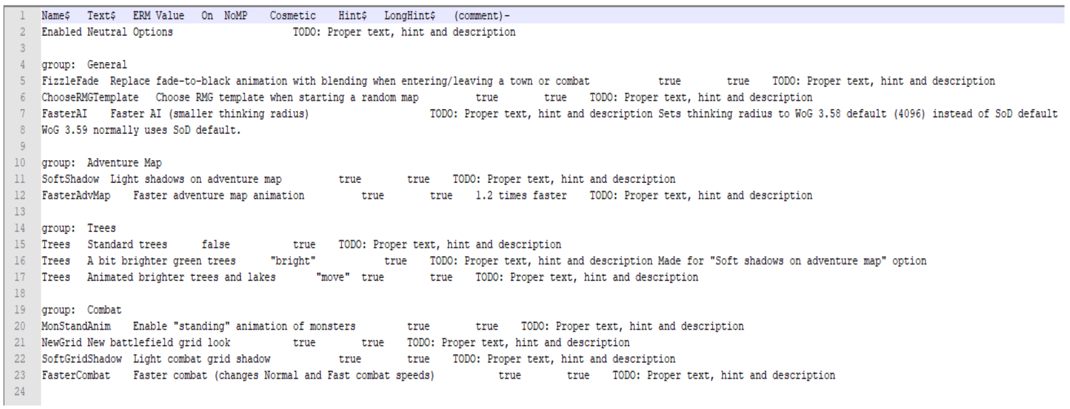
Hard-Coded Options

- ☐ Troop Stacks gain Experience
- ✓ Disable Cheat Codes and Menu
- ☐ Enable new "cover of darkness" and "builder" heroes

Game Enhancements

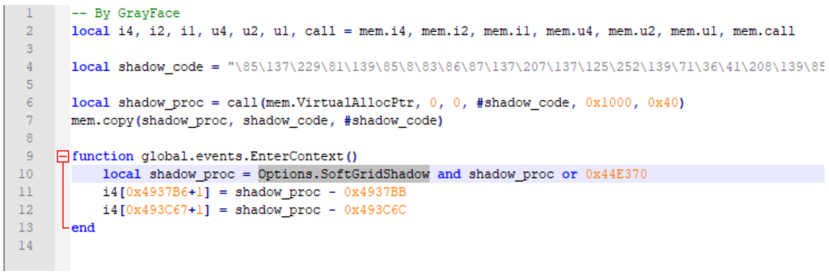
- ☐ *Display WoGification Messages*
- ✓ Enhanced dwelling hint text
- ☐ Towns may be renamed
- ☐ Heroes may be renamed
- ✓ Buy All Creatures button
- ☐ Transfer All Resources Option
- ☐ Resource Trading Post
- ☐ Display Hidden Skills
- ☐ Script Message disabling
- ☐ Display hero Movement Points remaining
- ☐ Display hero Spell Points remaining
- ☐ Custom Alliances
- ☐ Disable Autosave

File itself looks like this:



```
Name$   Text$   ERM Value   On  NoMP    Cosmetic    Hint$   LongHint$   (comment)-
Enabled Neutral Options                     TODO: Proper text, hint and description

group:  General
FizzleFade  Replace fade-to-black animation with blending when entering/leaving a town or combat            true        true    TODO: Proper text, hint and description
ChooseRMGTemplate   Choose RMG template when starting a random map          true        true    TODO: Proper text, hint and description
FasterAI    Faster AI (smaller thinking radius)                     TODO: Proper text, hint and description Sets thinking radius to WoG 3.58 default (4096) instead of SoD default
WoG 3.59 normally uses SoD default.

group:  Adventure Map
SoftShadow  Light shadows on adventure map          true        true    TODO: Proper text, hint and description
FasterAdvMap    Faster adventure map animation          true        true    1.2 times faster    TODO: Proper text, hint and description

group:  Trees
Trees   Standard trees      false           true    TODO: Proper text, hint and description
Trees   A bit brighter green trees      "bright"            true    TODO: Proper text, hint and description Made for "Soft shadows on adventure map" option
Trees   Animated brighter trees and lakes       "move"  true        true    TODO: Proper text, hint and description

group:  Combat
MonStandAnim    Enable "standing" animation of monsters         true        true    TODO: Proper text, hint and description
NewGrid New battlefield grid look           true        true    TODO: Proper text, hint and description
SoftGridShadow  Light combat grid shadow            true        true    TODO: Proper text, hint and description
FasterCombat    Faster combat (changes Normal and Fast combat speeds)           true        true    TODO: Proper text, hint and description

```

And this is where you can notice another important feature: **named WoG Options**! Chosen options are stored in variable “Options” and you can access them inside your scripts. For example to access SoftGridShadow option, type “Options. SoftGridShadow”.



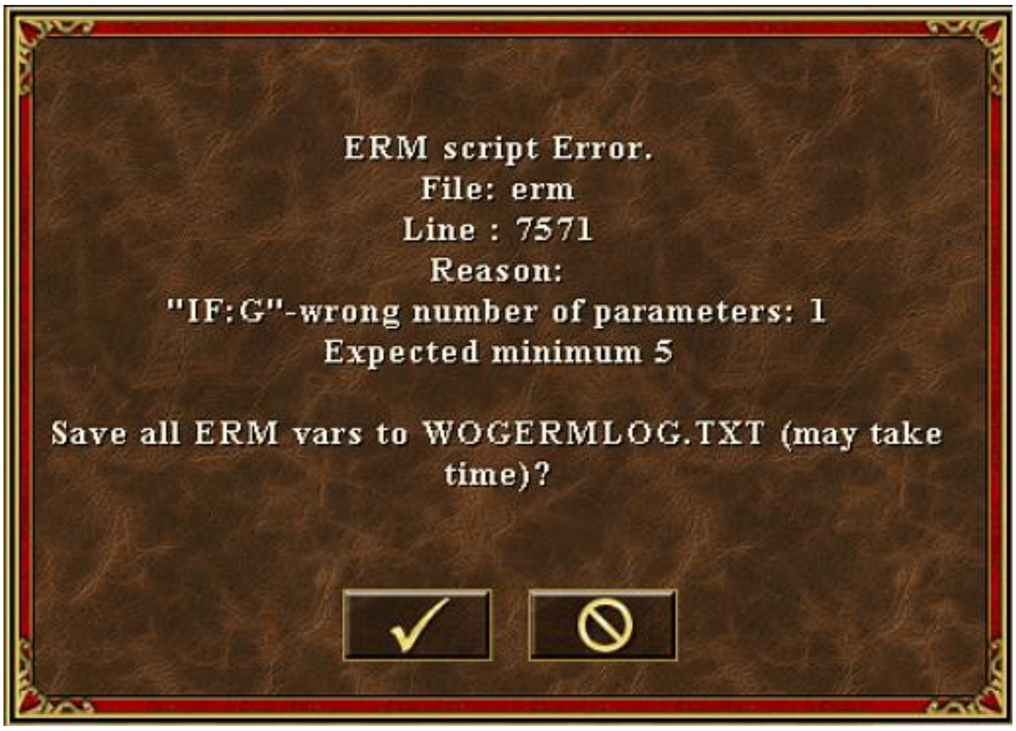
```lua
-- By GrayFace
local i4, i2, i1, u4, u2, u1, call = mem.i4, mem.i2, mem.i1, mem.u4, mem.u2, mem.u1, mem.call

local shadow_code = "\85\137\229\81\139\85\8\83\86\87\137\207\137\125\252\139\71\36\41\208\139\85

local shadow_proc = call(mem.VirtualAllocPtr, 0, 0, #shadow_code, 0x1000, 0x40)
mem.copy(shadow_proc, shadow_code, #shadow_code)

function global.events.EnterContext()
    local shadow_proc = Options.SoftGridShadow and shadow_proc or 0x44E370
    i4[0x4937B6+1] = shadow_proc - 0x4937BB
    i4[0x493C67+1] = shadow_proc - 0x493C6C
end

```

Variable “Options” is individual for every mod and is stored in savefile.

Better error messages

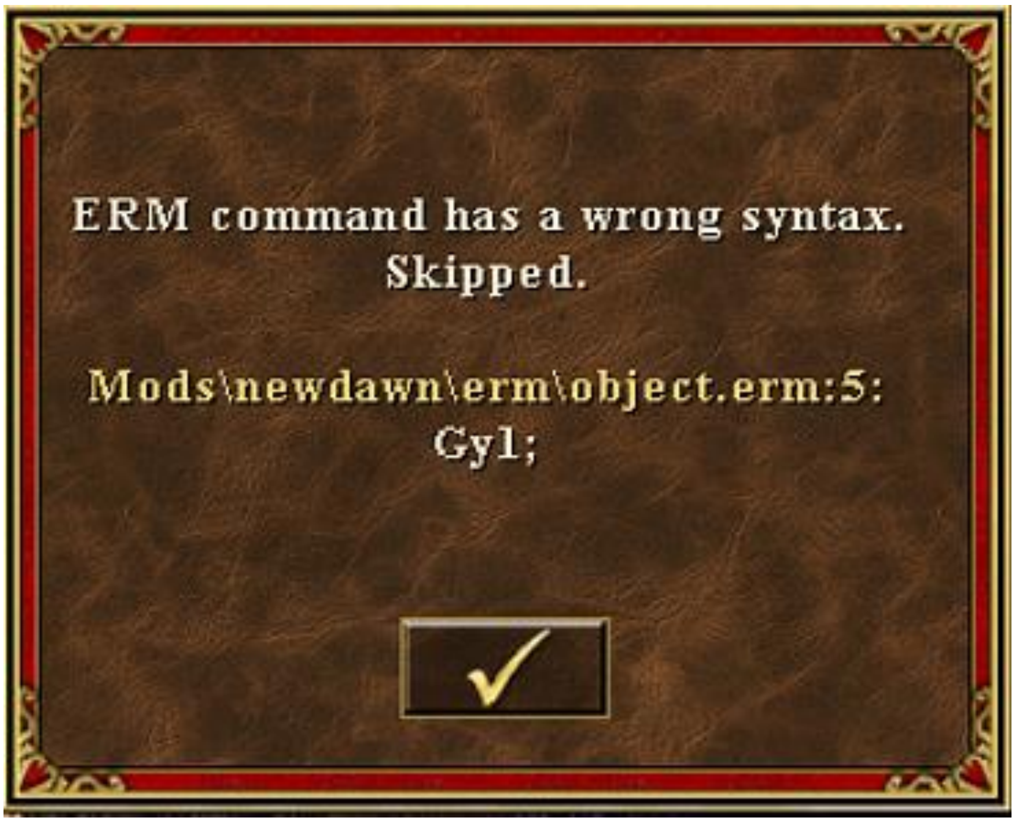
Old WoG 3.58f error messages were not very clear and often you could have troubles with finding which script is broken, and which line is problematic.

New error messages fixes that issue, showing exact mod, script and line where error occurred. Might sound like not-a-big-deal, but it really makes scripting much easier.



ERM script Error.
File: erm
Line : 7571
Reason:
"IF:G"-wrong number of parameters: 1
Expected minimum 5

Save all ERM vars to WOGERMLOG.TXT (may take time)?

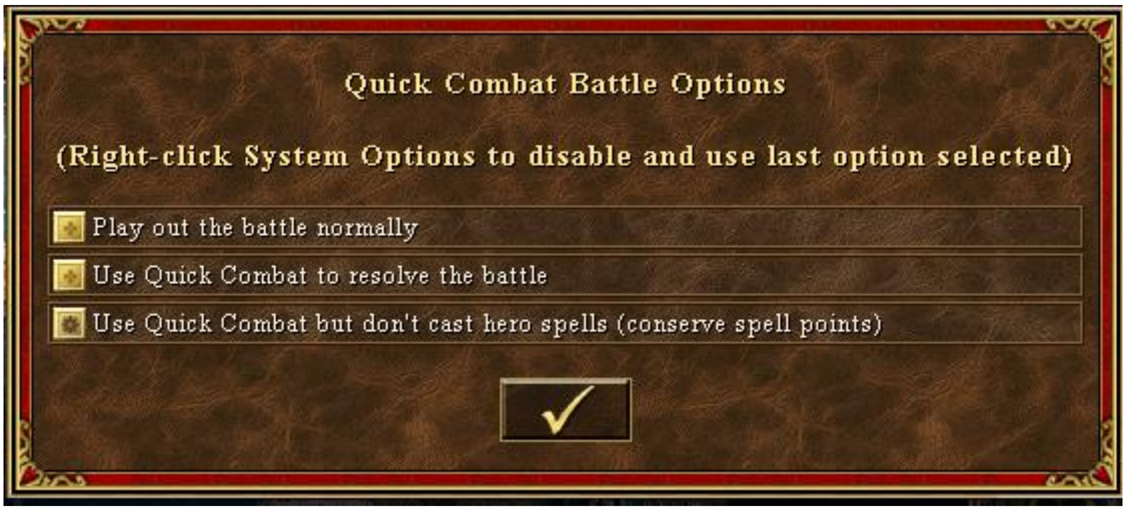


ERM command has a wrong syntax. Skipped.

Mods\newdawn\erm\object.erm:5:
Gy1;

New dialogs

You might know DL-based dialog system from Era. It was originally introduced in 3.59 alpha, but soon it was nearly abandoned for something much better: Lua-based dialogs. Many standard WoG dialogs are remade using Lua. Example:

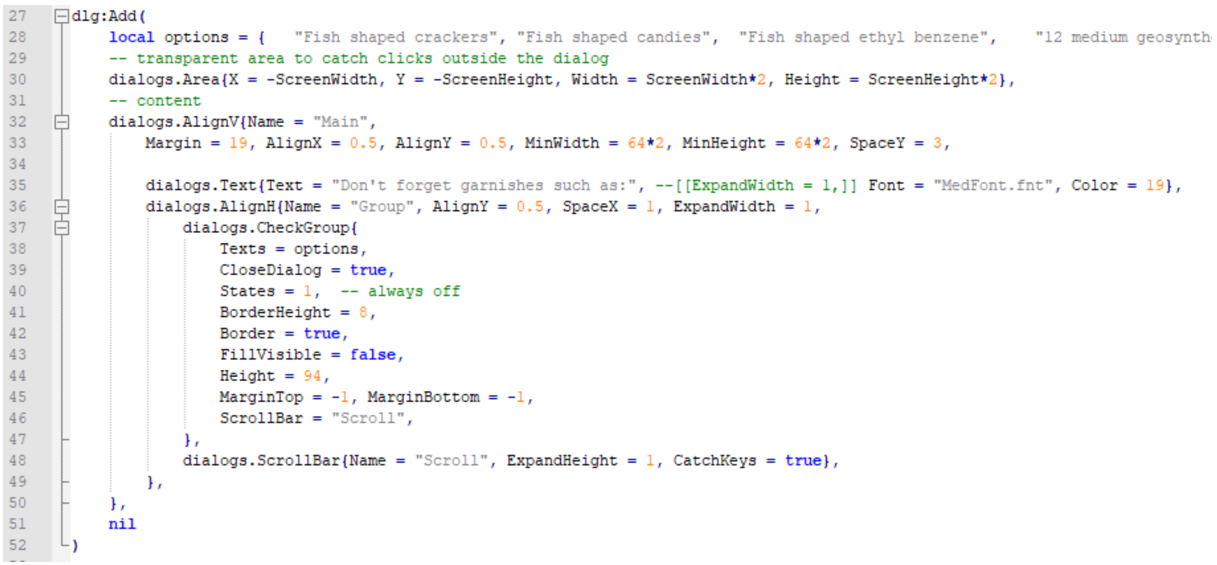


Quick Combat Battle Options

(Right-click System Options to disable and use last option selected)

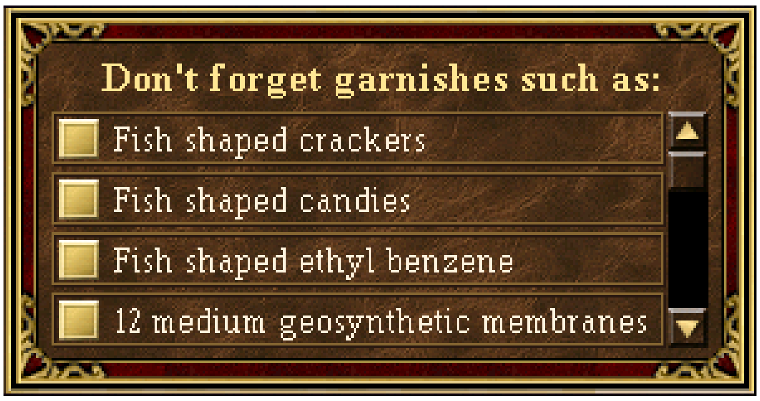
- Play out the battle normally
- Use Quick Combat to resolve the battle
- Use Quick Combat but don't cast hero spells (conserve spell points)

Of course, you can make your own dialogs. There are no limit, no "slots" for dialogs, and thousands of options to use.



```lua
dlg:Add(
    local options = {   "Fish shaped crackers", "Fish shaped candies",  "Fish shaped ethyl benzene",    "12 medium geosynth
    -- transparent area to catch clicks outside the dialog
    dialogs.Area{X = -ScreenWidth, Y = -ScreenHeight, Width = ScreenWidth*2, Height = ScreenHeight*2},
    -- content
    dialogs.AlignV{Name = "Main",
        Margin = 19, AlignX = 0.5, AlignY = 0.5, MinWidth = 64*2, MinHeight = 64*2, SpaceY = 3,

        dialogs.Text{Text = "Don't forget garnishes such as:", --[[ExpandWidth = 1,]] Font = "MedFont.fnt", Color = 19},
        dialogs.AlignH{Name = "Group", AlignY = 0.5, SpaceX = 1, ExpandWidth = 1,
            dialogs.CheckGroup{
                Texts = options,
                CloseDialog = true,
                States = 1,  -- always off
                BorderHeight = 8,
                Border = true,
                FillVisible = false,
                Height = 94,
                MarginTop = -1, MarginBottom = -1,
                ScrollBar = "Scroll",
            },
            dialogs.ScrollBar{Name = "Scroll", ExpandHeight = 1, CatchKeys = true},
        },
    },
    nil
)
```

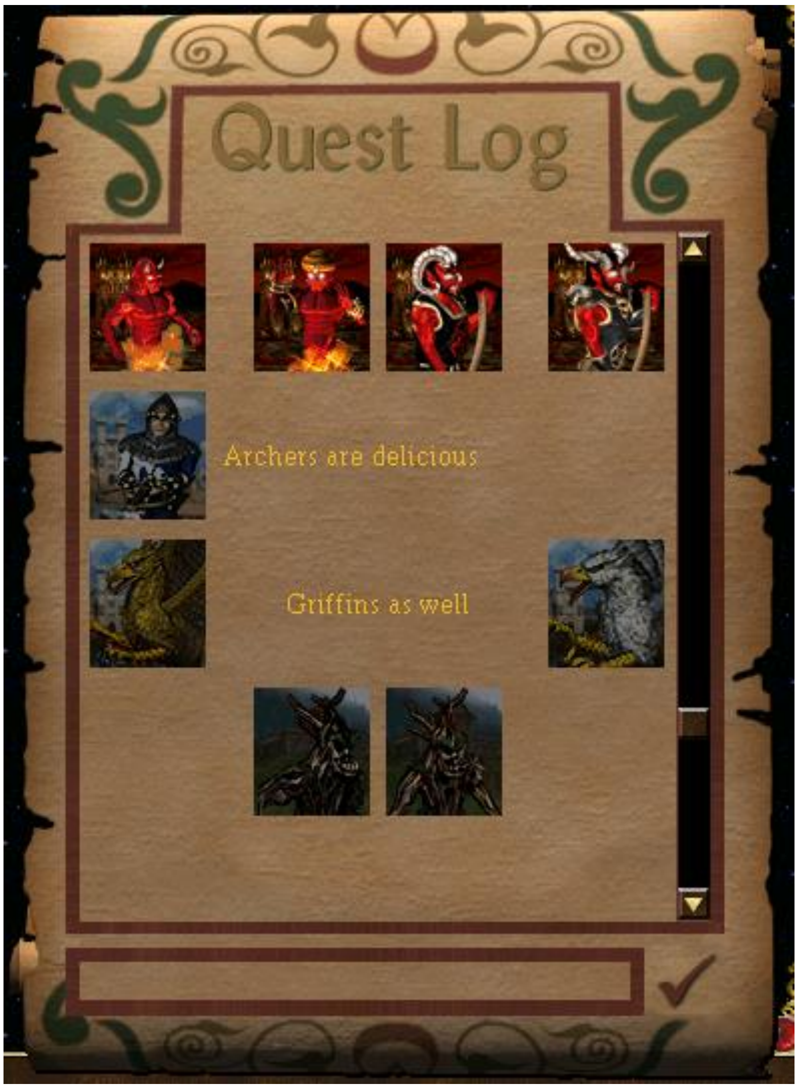
Output looks like this



Don't forget garnishes such as:

- Fish shaped crackers
- Fish shaped candies
- Fish shaped ethyl benzene
- 12 medium geosynthetic membranes

Another example. QuestLog.pcx is the background picture. Size of the dialog is derived from the picture This – and much more – can be seen in ERM/Lua Help. Lua -> Lua dialogs: Old tutorial



The thing that is mind-blowing in my opinion is: WoG Options dialog is actually written in this Lua-based dialog system!

Expanded ERM

WoG 3.59 introduce new receivers: CI, DG, DL, FC, HD, LD, SS, TL, UX, and few new syntax options. Here I will present only the ones I consider most important.

AI trigger and receiver

!?AI is called when Ai calculates weight of "interesting object" on adventure map. Each object within AI thinking radius is analyzed from perspective of every hero (you can refer to him via HE-1) Coords are stored in v998/v999/v1000

Inside block of this trigger you can use new receivers: !!AI:W that lets you get/set weight. and !!AI:M allowing to get/set movement points required to reach given square.

This has great potential to improve game AI, but making a proper formula to calculation of weight correction is problematic. There are many, many parameters to consider.

LD receiver

!!LD receiver allows to load/unload LOD archives. This might be used to replace graphics (trees for example) or add new ones for custom objects.

HD trigger and receiver

!?HD is triggered any time game Display Hint for object on adventure map. Coords of object entrence are stored in v998/v999/v1000.

Everybody who has ever tried to make dynamic hint for object will appreciate this.

Strings: internal buffer

If you pass a string constant (^hi!^) or a z-string (global or local), the value is copied into internal buffer (you don't need to use up a z variable for each object which hint you change). If you use get syntax for a string (?z1), it's set to the current value (even if the value wasn't customized)

Get address syntax: d?

In many commands you can use d? to get address of variable. It can later be used with UN:C receiver. If command doesn't support this syntax, 0 is returned.

Example: !#MA:P13/d?v1; [get addres of Archangels hitpoints]

Local funtioms

In every ERM script you have 100 local functions to be used. They are indexed FU-1 to FU-100, and their scope is limited to current scripts - so can be called only within this script.

Thus, using !!FU-10:P; inside script01.erm will trigger only !?FU-10; inside script01.erm.

If-elseif-else statement

Lack of proper conditional expression used to be biggest gimmick of ERM. Not anymore. !!if:;, !!el:; and !!en:; receivers.- note it's lowercase - allow to make if-elseif-else statements. You can freely nest them too.

Go-To statement

While usually you first learn to "not use goto" and the what goto does, for plain ERM it's wonderful tool. You can, for example, easily make while/for loops without reserving function for them.

Receiver UX - Universal Extended

New receiver that allows to : get mouse position in pixels, set/get many internal strings, disable custom grail effect and more.

Receiver SS - Spells Support

Well-known thanks to ERA, allows to change any spell's tier, magic school, name, mana cost, AI flags and much more.

Improved sound support

!?SN trigger runs every time sound is loaded instead of played. This allowed trigger to be called for nearly all sounds in the game - including loop sounds (or chanting, if you prefer)