

Luxor

Page for Luxor documentation.

- [Luxor 1 / Amun Rising](#)
 - [Getting Started with Luxor 1/Amun Rising Modding](#)
 - [*.psys files](#)
 - [Creating Maps](#)
 - [Tuning Level Settings](#)
- [Luxor 2](#)
 - [State File Modification](#)
 - [UI File Modification](#)
 - [GVF Types](#)
 - [General GVF Syntax](#)
- [Other](#)
 - [Limitations](#)
 - [Enabling Cheats](#)
 - [Mod Debugging & Errors](#)
- [Obtaining Legal Copies of Luxor Games](#)

Luxor 1 / Amun Rising

Documentation for Luxor 1 / Amun Rising

Getting Started with Luxor 1/Amun Rising Modding

So, you decided on making a Luxor 1 / Amun Rising mod? This page is dedicated to newcomers to modding the game. Luxor Amun Rising is often the modded game of choice due to it being an improvement (and added features) over Luxor 1, and has the same engine as said game.

What You Need

- A copy of the game. ([Steam \(might be broken for you\)](#) | [Big Fish Games \(is in a bundle pack\)](#))
- QuickBMS

Extraction

Luxor 1 and Amun Rising are packed within a MJZ file, which can be opened with QuickBMS. Luxor 1 and Amun Rising share the same engine, although Amun Rising is more recommended to be modded because of it's difficulties and it's ability to use `\n` in dialog boxes.

1. Download QuickBMS from [here](#)
2. Download the MumboJumbo script from [here](#)
3. Open QuickBMS and select the mumbojumbo script, and hit Select.
4. Go to your Luxor 1 (or Amun Rising) folder and select **data.mjz** in the `data` folder, then hit Open.
5. When another file select box appears to ask where to extract, **go up one folder** (this will take you to the game folder itself) and hit Save.
6. Repeat step 3.
7. Repeat step 4, but this time, **select english.mjz** and hit Open.
8. Repeat step 5.

Video explanation:

<https://www.youtube.com/embed/CBskMSDVUHI>

Now that is out of the way, you can either:

- Have separate folders for each mod and copy the game files there
- Have mods be in their own folder and swap mods by renaming data folders

If you opt for the first option, make sure to copy Steam.dll for each copy (Steam), or copy the bundle pack executable and launch Luxor 1/Amun Rising from there. (Bundle Pack)

If you opt for the first option, simply rename the data folder to something else, or take it out, then add the mod's data folder.

If you opt for the second option, here is a visual guide: [luxor-installation.png](#)

What's Next?

Know your basic goals. If you're making a mod, at least have a general idea of where to go, and what to do.

Before you just jump in thinking you're going to make the next high quality mod. **stop**. Those aren't easy to do. First mods are often mediocre, and that's okay! What's important is that you learn and improve over time.

Start small. Level design is one of the most important aspects of most mods, as they aim to create a new experience for the player, so start with that. Since this is likely your first mod, try aiming for the standard vanilla level order: 25 maps, 88 levels. Focus on the extra things (UI, menus, sound) later on.

Assuming that you are reading this because you are a new modder in the scene, try making maps first as practice, but keep these in mind when you finally make the decision to create your very first mod.

Now that you know what to do, it's time to learn about [making maps](#).

*.psys files

PSYS files are the visual effect files used in the Luxor 1 engine (used in 1 and Amun Rising). They can make or break the game's aesthetics, and if done right, can add to a level / the game, and can be stunning.

How to include effects to a ui file

Create a new `uiParticleSystem` child in your ui file (yes, this means that you can add effects to a level!).

```
Child child_name = uiParticleSystem
{
    X = 0
    Y = 0
    Depth = YourDepthHere
    File = your\path\here.psys
}
```

Make sure your `uiParticleSystem` child name is unique. Not doing so could make the game load the wrong `uiParticleSystem` child.

One thing to note is that unlike sprites, which have their anchor point in the top-left corner, their anchor point is at the center. This also goes for the emitter in the psys file.

For example, setting an effect to `x0 y0` will still go to the top-left corner as any other child does, but only the lower-right portion of the effect will be displayed in that position.

An easier way of saying this is: Treat psys file positioning as you would position the frog in Zuma.

Anatomy of a .PSYS File

We will look at a specific part of the "extra life" psys that happens when you receive an extra life.

```
Emitter ankh
{
    Type = Sprite
    Flags = EF_NONE
    StartParticles = 2
    MaxParticles = 2
    ParticleRate = 0.000000
    Sprite = data\sprites\particles\ankh.spr
    ColorRate = 0.000000
    AnimMin = 0
    AnimMax = 0
    AnimRate = 30
    FadeInEndTime = 0.000000
    FadeOutStartTime = 0.800000
    LifespanMin = 2.000000
    LifespanMax = 2.000000
    RenderDelay = 0.000000
    FirstFrameDelay = 0.000000
    PosX = 0.000000
    PosY = 0.000000
    SpawnRadiusMin = 0.000000 0.000000
    SpawnRadiusMax = 0.000000 0.000000
    StartVelMin = 0.000000 -25.000000
    StartVelMax = 0.000000 -25.000000
    Acc = 0.000000 0.000000
    DevDelay = 0.000000
    DevAngle = 0.000000 0.000000
    EmitterVelMin = 0.000000 0.000000
    EmitterVelMax = 0.000000 0.000000
    EmitterAcc = 0.000000 0.000000
    EmitterStart = 0.000000 0.000000
    EmitterLifespan = 0.250000 0.250000
}
```

At first glance it looks slightly simple - and it is simple!

Type

Not touched too much yet, keep it as Sprite.

Flags

These flags determine what this effect should do.

- `EF_NONE` - If your particle does not need any of these flags, use this flag.
- `EF_SPRITE_ANIM_LOOP` - makes the animated sprite loop, if it is animated.
- `EF_SPRITE_RANDOM_FRAME` - takes a random frame from the sprite, if it is animated.
- `EF_VEL_POSRELATIVE` - Particles' velocity will depend on initial particle position.
- `EF_LIFESPAN_INFINITE` - makes the particles' lifespan infinite.
- `EF_POS_RELATIVE` - If this flag is set then position of all particles will all the time be relative to emitter's position (for example, gem shadow moves together with gem, but sparks may not).
- `EF_ELIFESPAN_INFINITE` - Makes the emitter lifespan infinite.
- `EF_USE_COLOR_RATE` - makes the sprite use the palette from an image file (for example: the wild powerup's symbol changes colors taken from the data\bitmaps\powerups\wild_pal.jpg file)
- `EF_PALETTE_LOOP` - makes the palette loop back to the start. Used with `EF_USE_COLOR_RATE`.
- `EF_VEL_ORBIT` - makes the particles move along a circular trajectory.
- `EF_VEL_DEVIATION` - allows for DevAngle and DevDelay to be used.
- `EF_SOFTWARE` - unknown
- `EF_HARDWARE` - unknown

The Rest

- `StartParticles` - the amount of particles it will start with.
- `MaxParticles` - particle cap
- `ParticleRate` - $\text{ParticleRate} = 1/\text{dt}$, where dt is time period (in seconds) between the particle spawning. If `ParticleRate=1` then particle will be spawned each second. If `ParticleRate=10` then particle will be spawned each 100 milliseconds (10 times per second).
- `Sprite` - the path to the spr file
- `Palette` - if `EF_USE_COLOR_RATE` is enabled, this will make the sprite color itself to the palette file defined here
- `Spline` - Requires a valid .obj path to be used. When it is used, the emitter will travel on the path specified. It does not loop and will immediately stop movement at the path's end.
- `ColorRate` - if `EF_USE_COLOR_RATE` is enabled, this will set the speed of the color change
- `AnimMin` - unknown/not touched yet
- `AnimMin` - unknown/not touched yet

- `AnimMax` - this will make the palette loop back to the start. Used with `EF_USE_COLOR_RATE`.
- `AnimRate` - animation speed
- `FadeInEndTime` & `FadeOutStartTime` - Both between 0 and 1. Relatively to the particle lifespan. For example, particle lifespan time is 5 seconds, `FadeInEndTime`=0.2 and `FadeOutStartTime`=0.6. Then the real (not relative to the lifespan) time before the Fading In End and Fading Out Start will be $0.25=1$ and $0.65=3$ seconds respectively.
- `LifespanMin` & `LifespanMax` - minimum lifespan (in seconds) of a single particle
- `RenderDelay` - untouched, likely self-explanatory
- `FirstFrameDelay` - Delay before the animation start in milliseconds.
- `PosX` & `PosY` - X and Y positions of the emitter.
- `SpawnRadiusMin` & `SpawnRadiusMax` - The minimum/maximum (in separate X and Y percentages) of the spawn radius.
- `StartVelMin` & `StartVelMax` - The minimum/maximum (in separate X and Y percentages) of the velocity (in where do the particles go).
- `Acc` - The speed + relative position (in X and Y percentages) of how fast will the particles accelerate.
- `RotMin` & `RotMax` - The minimum/maximum degree of rotation.
- `RotVelMin` & `RotVelMax` - The minimum/maximum speed of how fast it rotates.
- `DragMin` & `DragMax` - unknown
- `DevDelay` & `DevAngle` - Works only if `EF_VEL_DEVIATION` flag is set. The particles' moving direction will start to be changing after the `DevDelay` time will pass.
- `EmitterVelMin` & `EmitterVelMax` - Emitter velocity. If there is a set Spline, these will be the speeds of the emitter moving along the spline.
- `EmitterAcc` - Emitter acceleration. Usually, this is double of the `EmitterVelMin` and `EmitterVelMax` values.
- `EmitterStart` - Emitter spawn delay.
- `EmitterLifespan` - Emitter lifespan.

Examples

When done right, you can do awesome effects like these:

An extra from Luxor Mod 2020:

https://www.youtube.com/embed/IIEvL_YCh_8

Sniper Bullet:

<https://www.youtube.com/embed/TrQIfTqQ2Jw>

Keep In Mind

- If it's something such as a ball collapse, make sure the particle count isn't high, as too much can overload the game. Keep it at the default value if you're not too sure!
- Luxor 1/AR does not have blending modes, so try to make things look not weird.
- The Scorpion shares a collapse with the Wild Ball, so try to make the wild ball collapse psys not colored.
- When adding temporary particles in a ui file, such as a dialog box, keep in mind that particle effects that don't have the `EF_LIFESPAN_INFINITE` flag do not reload.

Creating Maps

Now that you have decided to create a mod (or make one or two levels), it's time to create a map.

What You'll Need

- A graphics editor (Paint.net, GIMP, Adobe Photoshop, just something that you can do well in.)
- Nocturnal Owl De-vertice Editor, found [here](#)
- Some form of text editor. Notepad can do, although you may want to use Notepad++ for clarity.

Creating your Path Concept

Luxor spheres are 32 x 32 pixels thick. You can either have a 32 px thick stroke, or thinner. It doesn't matter, as long as the player has a general idea of how the path goes. Here's an image of a simple map: rows, but it's not plain rows. [luxor-path-example-1.png](#)

Now, there are a few things to note when making paths. You should avoid going down 500px or lower, as this is where the shooter area is. Clipping between the shooter and the spheres is close here.

Now that that's out of the way...

Creating the Background

You don't need to be a god at art to create a background. Just make sure it's decent and all. There are plenty of tutorials on the internet if you aren't much of an art prodigy - but try experimenting with plugins and effects, and you'll probably have something going.

The path should be visible to the player, so that they have a general idea of how does the path go.

For demonstration purposes, we'll go with a dirt and rock background.

[luxor-path-example-1-decorated.png](#)

Creating the Level Folder

1. Navigate to `data\maps` of your game. You should see a bunch of folders with no spaces and full of level names.
2. Create a new folder and name it to your level's name. For this tutorial, we'll name it `RockyRoad`.
3. Export the background you just made as a JPG and name it `background.jpg`.

Creating the path in N.O.D.E

Now is the time to download and extract N.O.D.E if you haven't already. If you haven't already, get it [here](#).

Before all of that, it's recommended to edit the background a little bit to guide you on what are you doing. Make the path lighter and add a 40px section (not including rounded tips) to the end of the path, as shown: [luxor-path-example-1-decorated-nodeready.png](#) Save this version of the background to the N.O.D.E folder, replacing `background.jpg` in the same folder as N.O.D.E.

This is where the tool comes in. Either open the exe file, the swf file or the html file.

4. Click on Start, and if it is your first time booting, enter a username. Ignore all of the other buttons, you will want to click on **New Path**. [NODE-tutorial-1.png](#)
5. Disable hotkeys, then name the level in the "Map Name" box. Enable the hotkeys again and focus on a text box that only requires numbers (such as the X/Y manual adding). You will want to remember 2 hotkeys: **H** for making the next node hidden / visible, and **Del** for deleting your last placed vertice. [QuNODE-tutorial-2.png](#)
6. Now trace the path. Use Magnifier and lower your mouse DPI if you need to do so. (For the 40px ending. N.O.D.E automatically hides your last two vertices on saving the path because of how the hidden nodes function work in Luxor. For demonstration purposes, here it is with the two vertices already placed in as hidden.) [NODE-tutorial-3.png](#)
7. Click on Save and you should see what looks like an obj file. Copy the contents. [NODE-tutorial-4.png](#) Get a plaintext editor, paste the contents there and save it as `path.obj` in your map folder, in this case `data\maps\RockyRoad`.
8. Back to N.O.D.E, click on **Toggle OBJ & UI/GVF**. This changes the dialog box's layout. Click on the pyramid that suits the level's ending best. In this case, we'll go with the right-facing pyramid, because the danger zone (the ending area of a path) goes to the left. [NODE-tutorial-5.png](#) Here's a handy protip: Virtually all Luxor 1/AR files are in **plaintext** - this includes UI, level settings, sprite file settings, et cetera.

- Copy the top text box's contents into a plaintext editor. You will want to remove this line.
Save this file as `map.ui` in the level folder.

```
GLSprite = 0 0 GameBackgroundSprites data\maps\RockyRoad\mask.spr
```

- Now, create a new file. Paste this in, replacing `RockyRoad` with your folder name. Save this in your level folder as `background.spr`.

```
data\maps\RockyRoad\background.jpg  
none  
800 600  
1  
1  
0 0
```

Applying and Testing your Level File

- Head over to `data\levels` and open `level_1_1` (Stage 1-1, or the first level). Replace "DIE KHUFU DIE" (or "URAEUS NEFERTARI") with your folder name **in all capital letters with spaces**. [level-apply.png](#)
- Now open up the game, choose a difficulty if you're in Amun Rising and click on Start. You should now see your new level. [rocky-road.png](#)

And that's it! You've created your first Luxor level. Congratulations! You probably want to have an in-depth look of the lvl file format, which can be found [HERE](#).

Tuning Level Settings

Level files are what loads the map and difficulty settings, They can be found in `data\levels`. You will want to look at the `level_x_x.lvl` files.

Level Files

Here is an example: Luxor Amun Rising's 1-1.

```
mapFile = "DIE KHUFU DIE"
```

```
// Ball Colors
```

```
spawnColor_1 = true
```

```
spawnColor_2 = true
```

```
spawnColor_3 = true
```

```
spawnColor_4 = true
```

```
spawnColor_5 = false
```

```
spawnColor_6 = false
```

```
spawnColor_7 = false
```

```
spawnStreak = 225
```

```
// Powerups
```

```
powerup_reverse = true
```

```
powerup_slow = true
```

```
powerup_stop = true
```

```
powerup_shotspeed = true
```

```
powerup_lightning = true
```

```
powerup_bomb = true
```

```
powerup_colorbomb = true
```

```
powerup_wild = true
```

```
powerup_scorpion = true
```

```
// Rewards
```

```
reward_gem_1    = true
reward_gem_2    = true
reward_gem_3    = true
reward_gem_4    = false
reward_gem_5    = false
reward_gem_6    = false
reward_gem_7    = false
reward_gem_8    = false
reward_gem_9    = false
reward_gem_10   = false
reward_gem_11   = false
reward_gem_12   = false
reward_gem_13   = false
reward_gem_14   = false
reward_gem_15   = false

// Win Condition (collapses and board clear)
winCondition = 100

// Vise Groups
viseGroupCount = 30

// Vise Speed
viseMaxSpeed = 800.0
viseMidMaxSpeed = 100.0
viseMidMinSpeed = 100.0
viseMinSpeed = 5.0
viseSpeedMaxBzLerp = 0.9 0.9
viseSpeedMidBzLerp = 0.25 0.75
viseSpeedMinBzLerp = 0.25 0.75

// Path Distances
viseSpawnDistance_1 = 0.6
midStartDistance_1 = 0.2
midEndDistance_1 = 0.6
```

Parameters

- `mapFile` - the folder name of your map with spaces and all in capital letters
- `spawnColor_n` - which sphere colors are activated. 1 is blue, 2 is yellow, 3 is red, 4 is green, 5 is purple, 6 is white and 7 is black.
- `spawnStreak` - This was supposed to increase / decrease the chances of clusters in a pusher train, though this parameter is **hardcoded**.
- `powerup_x` - Which powerups spawn. If you want to change a chance of a powerup spawning, edit `powerups.txt`.
- `reward_gem_x` - Which gems to spawn if a pusher train has been destroyed (not merged). For a reference of which gem gives a specific amount of points, refer to `powerups.txt`.
- `winCondition` - The amount of **spheres destroyed** to fill up the progress bar.
- `viseMaxSpeed` - the maximum speed of the spheres
- `viseMidMaxSpeed` - the middle-maximum speed of the spheres
- `viseMidMinSpeed` - the middle-minimum speed of the spheres
- `viseMinSpeed` - the minimum speed of the spheres, which is triggered on danger zone.
- `viseSpeed(Max/Mid/Min)BzLerp` - transitions(?)
- `viseSpawnDistance_n` - from 0 (0%) to 1 (100%), if there are no spheres in this area, spawn a new pusher train
- `midStartDistance_n` - from 0 (0%) to 1 (100%), trigger `viseMidMaxSpeed` here.
- `midEndDistance_n` - from 0 (0%) to 1 (100%), end `viseMidMinSpeed` here and trigger the `viseMinSpeed`.

For each path (up to 2), you must have a `viseSpawnDistance`, `midStartDistance` and `midEndDistance`. If there are no values for those, default values will be applied (unknown).

A better visual explanation: [Luxer.lvl demo.png](#) Image not found or type unknown

Every level slot you define in `stage_select.uis` must have their `level_x_x.lvl` file. The main menu level file is `level_0_0.lvl`.

powerups.txt

This txt file handles global powerup spawn chances and gem scoring. Everything here is self-explanatory.

```
// Powerups File
// Defines global powerup spawning chances and scoring

// Powerup Spawning
spawn_reverse    = 1000
spawn_slow       = 1000
spawn_stop       = 1000
spawn_speed_shot = 1000
```

```
spawn_lightning = 500
spawn_bomb      = 500
spawn_color_bomb = 500
spawn_wild       = 1000
spawn_scorpion  = 500
```

```
// Powerup Scoring
```

```
scoring_coin      = 250
scoring_gem_1     = 1000
scoring_gem_2     = 2000
scoring_gem_3     = 3000
scoring_gem_4     = 4000
scoring_gem_5     = 5000
scoring_gem_6     = 6000
scoring_gem_7     = 7000
scoring_gem_8     = 8000
scoring_gem_9     = 9000
scoring_gem_10    = 10000
scoring_gem_11    = 11000
scoring_gem_12    = 12000
scoring_gem_13    = 13000
scoring_gem_14    = 14000
scoring_gem_15    = 15000
```


Luxor 2

Documentation for Luxor 2

State File Modification

State files control certain functions of UI screens in games using the Luxor 2 engine. They have an extension of `.sm` and can be found in `data/state`.

For Luxor 2 (not HD), you may want to use a pre-fixed plaintext version of the state folder that should work with all versions [HERE](#).

General Ideas

Here's some basic ideas you could do with state files:

- Simple screen additions, such as a credits screen without the need for replacing something else for it, such as the Enable Cheats dialog
- "Are you sure you wish to quit?"
- Being able to assign **MORE** Map IDs in the Achievements screen, which **might** give way for mods with 26+ maps that properly function in Survival/Challenge/Gauntlet modes

“ **Remark:** I noticed that it seems like Luxor AR HD (or Luxor 1 HD?) used to be a 2-in-one pack of sorts, you can check `achievements.sm` and notice how there's "Luxor Completion" stuff commented out and the map counts starting from **26**, this is obvious when playing a map in Survival/Practice mode.

Remark 2: It might be already possible to have 26+ maps in the Survival/Challenge/Gauntlet maps, due to the fact that these levels start from the index of 26.

- Splitting achievements screen into two (achievements and personal stats), but I dunno why would you do that.
- Set music playlists for each screen (without the need for hacky workarounds, such as playing an `eiSound` in the "Danger" track and setting "Normal" track volume to 0)

But before you get too excited, know that you can't do these:

- An actual cutscene system (damn, I wish)
- Conditional statements
- A fix for 15+ stages (see: DRTPIAB's Epic Wins/Fails counter attempt, ask anyone lol)
- Implement things from other Luxor 2-based engine games (LXE, NCQ, MM, etc)

- Per-level tracks

“ **Remark:** It seems like Luxor Evolved-specific features are in Luxor Amun Rising HD, and possibly other HD remasters. While looking at it's memory allocation in HxD, I noticed Luxor Evolved specific things like `uiSpectrumFrame` and `Song` are in the game. Sadly, it is not possible to assign level tracks this way. It just seems like those were just sitting in the code, and the devs only worked on those when Evolved was next to be developed. It only makes sense considering Luxor HD remasters and Evolved were released during 2012, and the other HD releases never needed such functionality.

Basic State File Modification

Creating Simple Screens

Let's say we'd like to create a simple confirmation screen for when the player wants to exit. We don't need to worry much about it - the UI system takes care of our UI elements, and the state system takes care of the "programming".

First we need to look at `vars_pc.gvf` (Don't mind the iOS and x360 files, unless you're over your head and want to create a Luxor AR HD mod for Android devices, then yes, go ahead).

```
7 global TOPLEVEL_UI = ~data/scripts/interface/toplevel.ui
8 global TRANSITION_UI = ~data/scripts/interface/transition.ui
9 global GAME_UI = ~data/scripts/interface/game.ui
10 global DIALOG_OPTIONS_UI = ~data/scripts/interface/dialog_options.ui
11 global DIALOG_MODE_SELECT_UI = ~data/scripts/interface/dialog_mode_select.
```

As you can see, the `vars` pretty much define UI files for state machine files. We want to define a new UI here, as shown:

```
7 global TOPLEVEL_UI = ~data/scripts/interface/toplevel.ui
8 global TRANSITION_UI = ~data/scripts/interface/transition.ui
9 global GAME_UI = ~data/scripts/interface/game.ui
+ 10 global DIALOG_CONFIRM_EXIT = ~data/scripts/interface/dialog_confirm_exit.ui
11 global DIALOG_OPTIONS_UI = ~data/scripts/interface/dialog_options.ui
```

We're going to the ui files now. I'm not going to talk much about it, but we're mostly interested in the `Command` property.

```
.. UIButton Button_ActuallyQuit
.. {
..   Position = [ 108.000000, 390.000000, 0.000000 ]
..   AnchorHorz = CENTER
+ ..   Command = "QuitForReal"
..
..   uiTextWidget Text
..   {
..     Text = T( "Yes" )
..   }
..
..   uiSprite Icon
..   {
..     Style = "Icon"
..     Position = [ 48.000000, 48.000000, 0.000000 ]
..     Color = [ 1.000000, 1.000000, 1.000000, 0.908038 ]
..     AnchorHorz = CENTER
..     AnchorVert = CENTER
..     Sprite = ~data/scripts/interface/styles/dialog/button_ipad/sprites/icons/quit_game.png
..     BlendMode = ADD
..   }
.. }
```

The command name does not matter as long as it's readable and it's the exact same name you'll be using in the state file.

The state machine file we're interested in is `mainmenu.sm`.

```
37 Command "Profiles"   StateMachine::csPushState "Profile_Manage"
38 Command "Play"       StateMachine::csPushState "ModeSelect"
39 Command "Options"    StateMachine::csPushState "Options"
40 Command "HighScores" StateMachine::setState "HighScores"
41 Command "Instructions" StateMachine::setState "Instructions"
42 Command "Achievements" StateMachine::setState "Achievements"
43 Command "MoreGames"  enClientLocal::invoke_3rdPartyMoreGamesURL
44 Command "QuitMenu"   StateMachine::setState "MainMenuQuit"
```

The string after the keyword "Command" is what triggers the function/callback/whatever in the right. We want to change line 44 to something like:

```
37 Command "Profiles"    StateMachine::csPushState "Profile_Manage"
38 Command "Play"        StateMachine::csPushState "ModeSelect"
39 Command "Options"     StateMachine::csPushState "Options"
40 Command "HighScores"  StateMachine::setState "HighScores"
41 Command "Instructions" StateMachine::setState "Instructions"
42 Command "Achievements" StateMachine::setState "Achievements"
43 Command "MoreGames"   enClientLocal::invoke_3rdPartyMoreGamesURL
! 44 Command "QuitMenu"   StateMachine::csPushState "ConfirmExit"
```

We want to "push" the state and not "set" it, because our exit confirmation dialog is a pop-up, not full-screen. All we need to do now is to create the new state!

```
+ .. StateDialog ConfirmExit : BaseState
+ .. {
+ ..     Dialog = DIALOG_CONFIRM_EXIT
+ ..
+ ..     Command "Cancel"        StateMachine::popState
+ ..     Command "QuitForReal"   StateMachine::setState "MainMenuQuit"
+ .. }
```

You might be able to understand quickly how this works now. The "Cancel" command just "pops" the state and goes back to the previous state (which is the main menu). The "QuitForReal" command sets the state to "MainMenuQuit", which is already defined, which well... exits the game!

Think of the "Command" property as an alias to a function, with the state file providing the "link" to the function.

Assigning Music to Screens

Let's say we want to change the adventure Stage Map to play something else instead of the same old boring main menu theme. First we need to modify `data\game\music.gvf`.

We just need to add a new objEffectMap, like so:

```
1 objEffectMap Menu
2 {
3     Effect = ~data/scripts/effects/sound/music/menu.ofx
```

```

4  }
5
+ 6  objEffectMap Map
+ 7  {
+ 8    Effect = ~data/scripts/effects/sound/music/map.ofx
+ 9  }
10
11  objEffectMap Level
12  {
13    Effect = ~data/scripts/effects/sound/music/level.ofx
14  }

```

The OFX file handles the actual music track, we just need to create one, like so:

```

1  objEffect MapMusic
2  {
3    Loop = true
4    ObjectOffset = "Normal"
5
6    eiSound Song
7    {
! 8      File = ~data/music/sparkleunleashed-map.ogg
9      HandleGroup = "Music"
10   }
11  }

```

Then we edit `data\state\common\gcladventure.sm` like so:

```

31  StateDialog StageMap_Adventure : BaseState
32  {
33    Dialog = DIALOG_STAGEMAP_ADVENTURE_LUXOR
34    TrTransInDone = gameClientLocal_Luxor::trigger_advanceLevelCb
35
36    Init = gameClientLocal_Luxor::init_stageMapAdventureCb
+ 37    Init = enClientLocal::setMusicPlaylist "Map"
38
39    Command "Cancel"  gameClientLocal_Luxor::command_quitGame "StageMap_QuitGame"
40    Command "Start"  gameClientLocal_Luxor::command_startGameCb
41    Command "SignOut" StateMachine::setState "TransToMainMenu"
42  }

```

And now when you enter the stage map (**not** the Challenge of Horus stage map), the new map music track should play. It should also stop playing when you return to the Main Menu.

Documentation, So Far

Common Properties

Init

```
Init = namespace::Function
```

A function is called on state initialization.

Term

```
Term = namespace::Function
```

A function is called on state termination.

Functions

StateMachine

csPushState

```
StateMachine::csPushState "StateLabel"
```

Pushes a state into the current stack. Used for modal dialogs (such as confirmations).

csPopState

```
StateMachine::csPopState
```

Pops the current state pushed by `csPushState` and returns to the previous state.

csPopPushState

```
StateMachine::csPopPushState "StateLabel"
```

Pops the current state, then pushes a state into the current stack.

setState

```
StateMachine::setState "StateLabel"
```

Sets the state. Used for screens, like the High Scores screen or the Achievements screen.

enClientLocal

setMusicPlaylist

```
enClientLocal::setMusicPlaylist "Label"
```

Sets the currently playing music to the objEffectMap defined in `music.gvf`.

stopMusic

```
enClientLocal::stopMusic
```

Stops any playing music.

gameClientLocal_Luxor

command_gamePauseCb

```
gameClientLocal_Luxor::command_newGameOrContinueGameCb "Pause" | "Menu"
```

- `"Pause"` will pause the game with the "Paused: Press Spacebar to Continue" dialog.
- `"Menu"` will pause the game and open the pause menu that allows you to quit/exit/change options.

command_newGameOrContinueGameCb

“ ⚠ **Luxor Evolved-specific.**


```
gameClientLocal_Luxor::command_newGameOrContinueGameCb "DifficultySelect"
```

Checks if there's a currently running game. Else, show the difficulty selector.

State Types

StateDialog

Represents a dialog UI.

```
StateDialog Label : BaseState
{
    ...
}
```

Properties

Dialog

```
Dialog = NAME_OF_DIALOG
```

`NAME_OF_DIALOG` must be defined in the target vars file (usually `vars_pc.gvf`).

DialogTransIn / Out

```
DialogTransIn = true
DialogTransOut = false
```

Both properties are boolean. Tells the game if the dialog should transition in or out.

EffectTransIn / Out

```
EffectTransIn = "LabelOfObjEffectMap"
EffectTransOut = "AnotherObjEffectMap"
```

Both properties take a string. Uses these effects as transin/out effects.

Command

```
Command "Label" namespace::Function
```

A function is called when the command is invoked. **Must be defined in the ui file.**

StateServer_Luxor

Used in `gsv_*.sm` files. I don't recommend you poking around `gsv_*.sm` and `gcl_*.sm` files, since they pretty much handle the game itself.

```
StateServer_Luxor Label : Server_Common
{
    ...
}
```

Properties

MusicState

“ ⚠ **Luxor Evolved-specific.**

```
MusicState = "NameOfMusicState"
```

Sets the music state of the level.

UI File Modification

UI files control the layout of a specific element in the game - including level backgrounds.

Objects

Every object should have it's type and an optional label, with 3 spaces as indentation:

```
uiFrame theLabel
{
    uiSprite theLabel
    {

    }
}
```

uiFrame

A uiFrame is a container, usually used for dialog boxes. uiFrames can be adjusted with a uiFlowLayout.

uiScrollFrame

A uiScrollFrame is just like a uiFrame. However, overflow content is clipped and scrollbars appear.

uiContainer

A uiContainer is usually used for adding props, or as an anchor for OFX files.

uiInputFrame

A `uiInputFrame` is a container usually used for buttons or controls to change it's contents or icon depending on the active input device, such as a mouse or a controller.

uiDialog

A `uiDialog` is the root of either a dialog box, or a dialog screen.

uiSpectrumFrame

“ ⚠ **Luxor Evolved-specific.**

A `uiSpectrumFrame` is a container for `uiSpectrumChannels`. However, it is not a direct replacement for a `uiFrame`. A `uiSpectrumFrame` still needs to be wrapped in a `uiFrame`.

uiFlowLayout

A `uiFlowLayout` controls the layout of a `uiFrame`. This can be used to create grids, for example.

objEffectMap

- `Effect` `<Path>` The OFX file to use.

An `objEffectMap` applies an OFX file to an object. It takes the object label as the event.

Example:

```
objEffectMap Idle
{
    Effect = ~data/maps/town/idle.ofx
}
```

Valid labels are:

- `TransIn`
- `TransOut`

- Idle
- ShowMap
- ShowMap_Bonus
- HideMap
- HideMap_Bonus
- ContinueIn
- NewLife
- Announce_PU_<powerup> - LXE specific, <powerup> must be a powerup ID
- Milestone_Combo_<n> - LXE specific, <n> must be of: 6, 9, 12

uiBackground

A uiBackground defines the background of an object. It can be a fixed size or a 9-slice background.

uiTextWidget

A uiTextWidget inserts text into a container.

uiButton

A uiButton represents a button contained by it's elements.

uiProgressBar

A uiProgressBar represents a progress bar, usually used in the level HUD to indicate a rough amount of spheres left to destroy, or in the splash screen to indicate loading progress.

uiSprite

A uiSprite adds a sprite to a container.

Common Properties

GVF Types

Simple Types

- Number: A literal number. The game engine can usually take either a float or an int.
- Int64: Ending the number with an `i64` means it's an int and is usually used for score values.
- String: A literal string encased in double quotation marks (`"`).

Localized String

```
T( "string" )
```

Often used in UI elements.

Tuple

```
[ 1, 2, 3<,4> ]
```

So far only used for numbers. Commonly used in UI and OFX.

General GVF Syntax

GVF files are (presumably) "Game Variable Files" that also take on different extensions depending on the usage:

- `.gvf` - Game variables
- `.ui` - UI elements
- `.uis` - UI snippets
- `.ofx` - Object effects
- `.sm` - State Machine files

Objects

```
objectName
{

}
```

Includes

```
# path/to/whatever/you/need.txt
```

It doesn't *have* to end with txt.

Variables/Properties

```
varname = value
```

To assign it globally, add `global` before the variable name. This is only done in `vars.gvf`:

```
global varname = value
```


Other

Other for any non-specific documentation.

Limitations

Each game engine (and some specific games) have their own limitations which you should keep in mind:

Luxor 1/AR

Engine Specific

- `data/uiscrypt/depths.txt` has a hard limit of 69 layers. If this is exceeded, layers starting from the 70th will be ignored.

Luxor 1

- Stage 2-4 (and by extension, 5-4, 8-4, 11-4) do not prioritize colors in the danger zone, presumably due to the vanilla level being Khufu's Revenge.

Luxor 2

Engine Specific

- Going over the maximum Map ID (25 for Luxor 2) crashes the game.
- Only a maximum of ~100 images can be loaded. Any more and the logs will show an "Out of Factory Nodes" error and any other images past it will **not be loaded**. To work around this, you may want to place images in atlases, or simply don't go overboard.
- You may want to look at the logs every time you crash after toggling Fullscreen. A potential reason for this is that the game reloads every single texture *including unused ones*.

- Setting the BlendMode to `NONE` will completely ignore the alpha. Omit the BlendMode property altogether if you wish to include it.

Enabling Cheats

When developing mods, you may want to enable built-in cheats to test a specific level.

Luxor 1/AR

- Pass `-unlocklevels` and `-unlockstages` to the executable.
 - `-unlocklevels` allows you to select a specific level from the currently selected stage when starting a new game.
 - `-unlockstages` allows you to select a specific stage when starting a new game.

Luxor 2 (and games that use it's engine)

Open it's Options menu and press `PgUp` + `PgDown` at the same time. This will open a dialog asking if you want to enable cheats **permanently** for the current profile.

Once enabled, you can go to Options > Cheats to access the following cheats:

- **Unlock All Stages** - Similar to `-unlockstages`. This will allow you to select a specific stage when starting a new game. This also unlocks all maps in Survival, Practice and Pharaoh's Challenge (Luxor 2).
- **Unlock All Levels** - Similar to `-unlocklevels`. allow you to select a specific level from the currently selected stage when starting a new game.
- **Unlock Challenge of Horus** - Unlocks the Challenge of Horus mode.
- **Enable Super Easy Mode** - Only initially available on the HD remasters and Evolved, but is present in Luxor 2 via state machine and UI editing. This will set all levels' winCondition to 1, and set all scarab train sphere counts to 1.
- **Time Scale** - Only available on the HD remasters and Evolved. This adjusts the speed of the game, so don't put it too high or too low.

Mod Debugging & Errors

You're in the middle of testing and you just made something new. And then when it loads... BAM! Your game explodes and you have no idea where to look.

- **Luxor 1/AR:** Check the luxor.log or luxorAR.log file and scroll to the bottom. Look out for any warnings about missing files, invalid characters/syntax, etc.
- **Luxor 2 and games that use it's engine:** Check your Documents/MumboJumbo folder.

Each game has it's own folder:

- Luxor 2: luxor2
- Luxor 1 HD: Luxor 1 HD
- Luxor 2 HD: Luxor 2 HD
- Luxor AR HD: Luxor Amun Rising HD
- Luxor Evolved: Luxor Evolved
- Myth Match: mythmatch
- Neopets Codestone Quest: neopets

Obtaining Legal Copies of Luxor Games

Since Luxor games are pretty quite dated at this point (2005), you may think that none of these games work anymore. However, there are still proven ways to obtain a working copy of these.

- **Big Fish Games**

- **Luxor Bundle Pack**

- Hands down the best deal available on Big Fish Games for *both* Luxor 1 and Amun Rising.
 - Will require you to open the Bundle Pack EXE first. If you wish to have a unique game copy per mod, you will need to copy the executable to the mod folder, which is marked hidden by default.

- **Luxor 2**

- Works on modern Windows machines, but may not launch from the launcher.
 - You can launch the game by opening it's hidden game executable on the installation directory.
 - If you are using a new account, you can enter the code `NEW299` to pay for your first game for 3 USD instead of the usual price.

- **GameHouse**

- **Luxor**

- **Luxor Amun Rising**

- **Luxor 2**

- Reported to work on modern Windows machines.

- **Steam (NOT RECOMMENDED)**

- **Luxor**

- **Luxor Amun Rising**

- **Luxor 2**

- Reported to **NOT WORK** on modern Windows machines.
 - Reported to work on Linux via Proton.