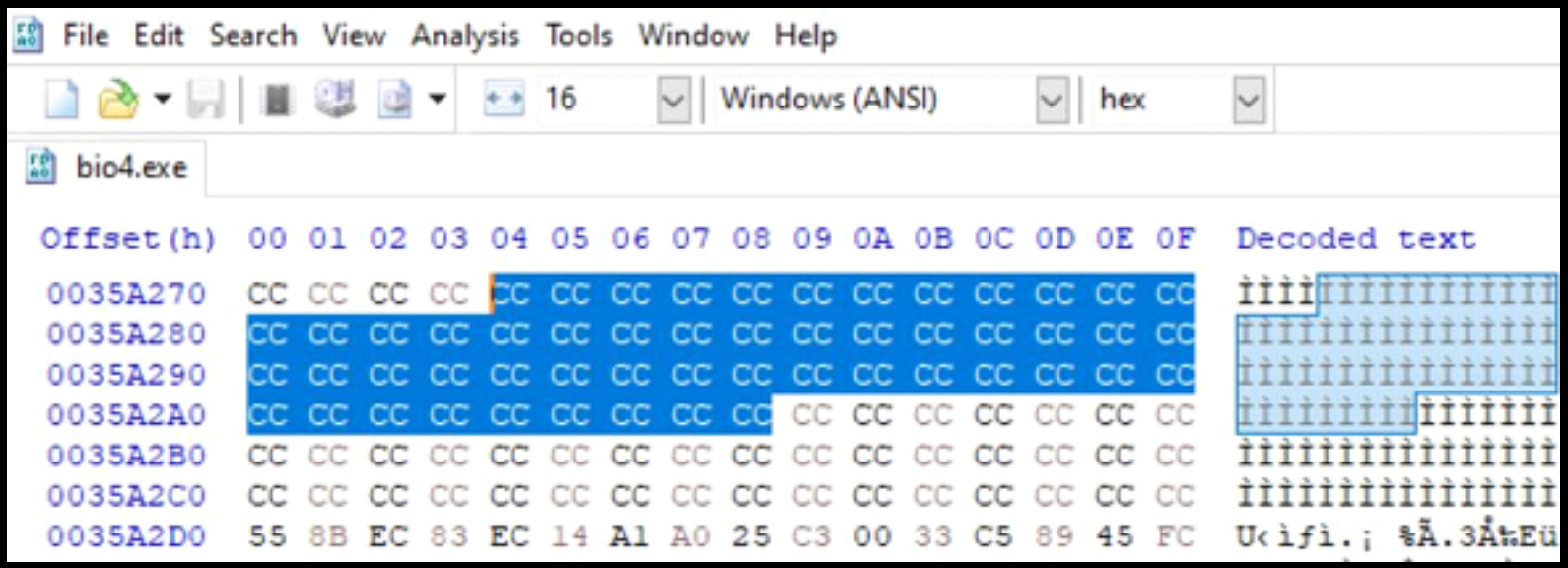
BIO4.EXE MODIFICATIONS (v.9.0)

Hardcoding edits into the bio4.exe

Research & Development by AnonymousUser & Mr.Curious
Guide by Mr.Curious





Mr. Curious
@MrCuriousModding
2.61K subscribers

HOME VIDEOS PLAYLISTS COMMUNITY CHANNELS ABOUT

Users can check out other video tutorials on my YouTube Channel HERE
Users can also see a full list of my available tutorials HERE.

IMPORTANT UPDATE NOTICES

PINNED NOTE* SANITY CHECK

Any edit above version V.7.2 must have the Sanity Check edit installed to work.

PINNED NOTE* INTERDEPENDENT EDITS

It is important to understand that there are several edits (older and newer), that require each other to be installed to function properly. These are what we call ‘Interdependent Edits’ that share code. It is REALLY IMPORTANT that these Interdependent Edits are installed in the correct order. Here is a list of Interdependent Edits and their dependencies. I highly recommend simply installing all of these to ensure that everything works correctly.

Here is the list. All items in this list are hyperlinks to the detailed edits lower down in this document. These edits should be installed in linear order, from the top of this list to the bottom with their **required dependencies installed first**, because some edits will overwrite parts of the next edit, so **doing this in the right order is imperative**.

—----------

FSE - AEV link

AEV-ESE Toggle
- **REQUIRES**
  FSE - AEV link

AEV - Auto-Door Block
- **REQUIRES**
  - **FSE - AEV link**
  - **AEV-ESE Toggle**

AEV-CAM Toggle
- **REQUIRES**
  - **FSE - AEV link**
  - **AEV-ESE Toggle**
  - **AEV - Auto-Door Block**

AEV-Checkpoint (added 1-17-2024)
- **REQUIRES**
  - **FSE - AEV link**
  - **AEV-ESE Toggle**
  - **AEV - Auto-Door Block**
  - **AEV-CAM Toggle**

AEV CHAIN
- **REQUIRES**
  - **FSE - AEV link**
  - **AEV-ESE Toggle**
  - **AEV - Auto-Door Block**
  - **AEV-CAM Toggle**
    - **AEV-Checkpoint** (added 1-17-2024)

—----------

Spawn Enemies from ITA (including ETM) models
- **REQUIRES**
  - **Sanity Checking Function**

—----------

AEV-ITA Enemy Spawn Persistence & ETS ID's
- **REQUIRES**
  - **Sanity Checking Function**

Skip down to **TABLE OF CONTENTS** (link)

WHAT'S NEW

(links to new edits and updates to existing edits will be posted here)

What's new in v..9.0 (1-16-2024)

AEV-CAM Toggle (update)
There is a small update to this edit that allows for the functionality of another new edit (AEV Checkpoint, see directly below this). Since these edits are interdependent, it is best to simply install both edits. Check this important update **HERE**.

AEV Checkpoint
New system of setting checkpoints (without the use of a TypeE in previous edits). Can disable checkpoints on AEV events (like doorways to other rooms) or add checkpoints to other events. **IMPORTANT NOTICE ABOUT THIS EDIT** - This edit has prerequisites so please be sure to see the updated version of the Interdependent Edits section above.)If you want to install this edit you will need to update the AEV-CAM Toggle edit (as noted above)

Ashley Jumps Alone (new info in - this edit should be working now)
We can not set Ashley to jump without the assistance of Leon having to catch here. Check the edit **HERE**.

Ashley Water Effect in any room
Presently even if we port over all the water slashing effects from another room to a custom room, Ashley will not make any ripples or splashes in the water because of some code that checks if we are in room r10a or r11a. If we are not in these rooms then the water splashing effects will not work. This edit fixes that. Check the edit **HERE**.

What's new in v.8.9 (1-10-2024)

AUTO Message/CAM/Sound (Update)
There was a bug with flag Offset[56] = 0x8 not working to trigger custom prompts. This has been patched now. See the update **HERE**.

What's new in v.8.8 (1-07-2024)

Discard AEV (update)
Added functionality now allows us to have more than one 'Success' and 'Failure' AEV ouputs. This is a game changing feature that allows us to toggle multiple AEV events at once, without having to use a chain. A more descriptive overview has been created below. View the update **HERE**.

AUTO Message/CAM/Sound
This edit allows us now to trigger Type5 events automatically without a prompt, without the use of the Companion DLL. Very useful for triggering sounds, cameras, and messages instantly without having to mess around with the events.cfg file. View this new edit **HERE**

AEV Options (update)
- Fixed a bug where AEV options would eventually cause MDT's to bug out and become erratically spaced.
Update can be viewed **HERE**.

What's new in v.8.7 (1-04-2024)

Discard AEV
Allows you to take items away from the player and activate/deactivate AEV's accordingly. View this new edit **HERE**

AEV Options (update)
- Fixed a bug where AEV options would eventually cause MDT's to bug out and become gibberish. Update can be viewed **HERE**.

102 - No Teleport After Cutscene
This edit prevents the player from being teleported after the Merchant cutscene in r102. View this new edit **HERE**.

This edit prevents the Merchant from opening the store after the cutscene. View this new edit **HERE**.

r320 - No Teleport After Cutscene

This edit prevents the player from being teleported after the first cutscene in r320 (**r320s00.evd**). View this new edit **HERE**.
QuickTip: To prevent the second cutscene from playing directly after the first, simply nul the file r320s01.evd

---

What's new in v.8.6 (12-29-2023)

Part 1: Spawn Enemies from ITA (including ETM) models (update)

- Added functionality. Respawned enemies can now drop items infinitely by linking the AuraType to an ITA with the specified Item. Update can be viewed **HERE**. Instructions can be found **HERE**.

AuraType of items when shot from a high place

Provided bio4.exe address and value of the default AuraType used for falling items. This edit can be viewed **HERE**.

Custom CES (Chapter Ending Screens)

Update to relocate the CodeCave used for user data. Previous CC only allowed for a limited number of entries. This edit can be viewed **HERE**.

What's new in v.8.5 (11-27-2023)

AEV Options (update)

Fixed a bug where hardcoded choices would trigger the wrong choice.
- Fixed a bug where if the first choice doesn't have a linked AEV then the rest of the choices won't trigger their respective AEVs regardless of selection.
See this edit **HERE**.

Part 1: Toggling Effects with an AEV event (update)

- Fixed EAR's disappearing when walking out of other AEV-EARs. Update can be viewed HERE.

Whats new in v.8.4 (11-07-2023)

Part 11: AEV Options

Fixed a bug where upgrading an item from the merchant causes the game to crash.

Whats new in v.8.3 (11-04-2023)

Renaming of Part 9

There was some confusion as to which edits were needed for ITA-AEV spawns, so the edit **Part 9: Spawning**

Enemies, Items, Treasures & Weapons with AEV has been renamed to **Part 9: AEV-ITA Functionality**. This renaming is to simplify the explanation of what this edit does; it simply links the AEV and ITA together.

Part 1: Spawn Enemies from ITA (including ETM) models (update)
There was a small piece of code missing from the previous update of this edit, which should fix some scaling issues with some of the non-Ganado enemies. The update for this edit can be viewed **HERE**.

What's new in v8.3 (10-15-2023)

Very important updates in this release that essentially fixes some Interdependent Edits, that weren't previously added, or linked here.I highly recommend reading the new Pinned Section above called Interdependent Edits to understand how things have changed.

Part 11: AEV Options
Allows for use of MDT generated choice menus to toggle AEV events. The previous release had a mistake (conflict with another edit), so it is now an independent edit and works as standalone.

Part 12: AEV - Auto-Door Block
Allows for enabling/disabling the functionality of ETM Auto Doors. NEEDS TO BE INSTALLED FOR OTHER EDITS TO WORK. See **Interdependent Edits** section above.
Small update to part 2 of the ITA EVENT MODIFICATIONS. Basically fixes a crash when the amount of enemies spawned exceeds the limit set in the CNS file.

What's new in v8.2 (9-24-2023)

Part 9d - Universal ETS ID Access
With this edit we are now able to toggle AEV and ITA events from one room to another.

Companion DLL Automatic Key Unlock
If the first ItemID in any events.cfg entry is 0xFFFF00xx (XX= [ITEMID] then it will unlock without opening your inventory,

DLL Key Unlock No Message No cam
Allows for preventing messages and cameras from being assigned in the events.cfg. Previous methods of simply assigning values of unused CAM entries resulted in camera jitter. The message part of this makes it so we do not need to create blank MDT messages for CAM shots without text.

Keep EAR Visible After Death
After Leon Dies, the game disables the visibility of EARs, this fixes that.

Parasite Plaga ITA Respawn Fix
Fixes the issue of Spiders persisting when they shouldn't with the ITA edits

r320 No Explosion Sound
Removes the explosion sound that occurs when you enter r320 for the first time.

Luis Never Responds to Persistent Attacks Outside of r11c
When not in the cabin fight, Luis will become unresponsive after shooting him 5 times, this edit prevents this.

What's new in v8.1 (9-13-2023)
SWAP / CLONE ROOM INIT
Added tutorial on how to clone / swap room INIT instructions. This newly learned ability allows users to clone the instructions from room to room. Imagine having two rooms with all the same functions, but with a totally different layout! See the ROOM INIT MODIFICATIONS Guide below

SWAP / CLONE ENEMY AI
This recently discovered functionality allows us to put the AI (enemy behavior) or one enemy onto another. See full description and tutorial HERE.

What's new in v8 (7-31-2023)
All camera edits have now been moved to their own dedicated section of this guide, and are now labeled as Part 1, Part 2 etc. Edits do not appear in order of creation or publication, so be sure to check this section for any edits that are new.

CAM EVENT MODIFICATIONS - Part 1:Control Camera Visibility/Freezes
This edit will allow for total control of what is visible to type6 cameras. This includes player visibility, effect visibility, Allows for freezing player, freezing AEV, freezing timers and much more. This is the motherload edit for the CAM file and essentially makes the previously published CAM edits that control visibility of some game elements redundant. That being said, this edit requires that users hex edit their CAM files to control what is visible, so for simple visibility (like effects during cam), some of the previous edits are easier to use.

CAM EVENT MODIFICATIONS Part 5: Effects Keep Playing During Cam
This edit will force the game to keep effects present when a type6 camera is triggered

CAM EVENT MODIFICATIONS Part 6: Drawn Enemies During Camera Events
Allows for whitelisting enemies for type6 cameras. This edit is somewhat redundant considering this can be accomplished using the Control Camera Visibility/Freezes edit above, but allows for some flexibility.

What's new in v7.9 (7-23-2023)

Allow EM's to climb AEV ladders
For whatever reason the devs of RE4 thought it would be useful to make it so that some enemies (Ganados, Zealots, and Islanders) should be restricted from using wall ladders (these are to be distinguished from movable ETM ladders that we can kick down and raise up). These are the Type 10 AEV ladders that enemies and the player can climb straight up (an example of this is the ladder in the tower in r101). AnonymousUser has identified the original table that handles which enemies are allowed to use ladders and the values associated with them.

Allow Additional EM's to climb AEV ladders
Since the original table mentioned above only deals with enemies from em10 through em3f, AnonymousUser has expanded the table to include all enemies up to FF (the original table starts at em10 and goes up to em3F). By default,

any EM higher than 3F will NOT be able to climb ladders, but with this edit the functionality of the original table is extended by simply extending the table.

Change Memory Allocation Amount
This is a guide to help along with user-end editing of the bio4.exe to manage memory usage. Changing how much memory is allocated can often help with additional requirements for heavy modding that demands more memory (high resolution textures, combined with over a dozen different types of enemies, along with sideloaded textures can really add to the limit).

What's new in v7.8 (7-16-2023)
As part of an effort to keep things organized, several edits have been split, with another being deprecated. The 'Auto Destruct' edit has been deprecated as recent testing showed the intended effect of skipping a prompt worked, but this was later realized to be redundant with the ability to use Type 6 events. Basically what happened is that at some point code was introduced with the AutoCam edit that made Type 6 AEV events no longer require the AD byte. This was an unintended functionality, but nonetheless, makes the AutoDestruct redundant. This part of the AutoCam edit has been migrated to a new edit called 'Type 6 AEV Toggle'. (more on these two edits below).

Type 6 AEV Toggle
This edit allows Type 6 AEV's to activate automatically when stepped on (without a prompt). Basically used as a trigger in a chain, or as a trigger for ITA Enemy Spawns.

CAM EVENT MODIFICATIONS - Part 4: No Escape Auto Cam (Type 2 events)
This edit allows us to set a flag to prevent users from escaping auto-cam shots. This edit requires that the previously posted 'Type 6 AEV toggle' edit be installed.

Updated Edits in v7.8 (7-04-2023)
Part 1: Spawn Enemies from ITA (including ETM) models
It was recently pointed out to us that only ganados, zealots and islander enemies were properly scaling when spawning them via ITA, so this is essentially a fix that allows proper scaling/speed of enemies when spawning via ITA.
VIEW UPDATE HERE

Part 9a: AEV-ITA Enemy Spawn Persistence & ETS ID's
Special enemies like 3F and 32 used to immediately trigger AEV's linked to their ETS IDs when they spawned, however this update fixes that. There is a varying byte in the edit depending on which RE4Tweaks setting is used (see note in edit). RE4Tweaks is **NOT** required for this edit, but if it is used use assigned byte)
VIEW UPDATE HERE

Part 1: FSE - AEV link
Updated the FSE edit to include being able to toggle multiple FSE entries with a single AEV event. This makes it possible to stop one song and start another with a single AEV event. Also this edit makes it so we can use custom fight music when an enemy is spawned (via ITA-AEV spawn) and end the song when the enemy is killed. There was an issue that prevented this working when using certain emxx.udas (mostly bosses like em32 and em3F) but this has been resolved in the 'AEV-ITA Enemy Spawn Persistence & ETS ID's update just above this so be sure to install that update as well.
VIEW UPDATE HERE or HERE (both the ESE and FSE edits share this block of data)

Higher EM XWB/XSB Limit
Update that fixes XSB/XWB's not unloading when returning to the main menu which caused a slow memory leak leading to crashes.

What's new in v7.7 (6-17-2023)

- Fixes for Companion DLL issue of loading sidechained .pack files in the 'TEXTURES' folder twice upon retrying
- Removes Companion DLL persistent check of the "TEXTURES" folder.
- Fixes shared EFF textures incorrectly loading with some sidechained enemies.

Companion DLL No 2x Allocation

Fixes the Companion DLL doubling the memory allocation of certain files, which wastes memory.

Higher EM XWB/XSB Limit (new)

Allows for more XSB/XWB files to be loaded into memory than previously allowed.

Changelog:

* Some kind of intermittent sound data needed more space, changed size from 0.06 MB to 1 MB (Global MEM) (FIXES save screen)
* New XSB/XWB's weren't replacing old ones correctly causing sound issues like wrong guns sounds and no "RESIDENT EVIL 4" voice and other mishaps

Night Time Eyes Always On

Red eyes of Ganados are now activated in all daytime rooms,

What's new in v7.6 (5-17-2023)

Small update to the **EM Compatibility** edit. Users will néed to revert the old edit and install the new one. Improves performance and fixes a small bug.

What's new in v7.5 (3-29-2023)

ESE Paused Instead of Removed

Fixes issue of ambient environmental sounds, as well as ESE triggered sounds from stopping after an .SFD movie is triggered via the Companion DLL

CAM EVENT MODIFICATIONS Part 2: ITM Shown in Type 6 Camera Shots

Fixes issue of ITA spawned items not being shown during Type 6 camera events.

AEV Merchant

Turns AEV event into Merchant shop.

EM Incompatibility Issue Fix

Allows for placing enemies in any order in ESL

XWB/XSB Sideload by Quinsheng (Companion DLL version)

Adds functionality of side-loading xwb & xsb files

EM Sideload by Quinsheng (Companion DLL version)

Adds functionality of side-loading emxx.udas files as well as xwb & xsb files

CAM MODIFICATIONS Part 3: AEV-CAM Toggle

Allows Type 8 CAM entries to be toggled with AEV Type2 events.

What's new in v7,4 (3-25-2023)

Part 9c - EM ITA UDAS Pre-Load

Removes lag when spawning an enemy for the first time via ITA. This issue was fixed for ESL type entries in RE4Tweaks, but this fixes the issue for ITA spawns. Also, if the ITA event that spawns the enemy has already triggered, it will no longer pre-Load. This edit requires the **Sanity Check** edit to work.

Updated Edits in v7.4 (3-25-2023)

There are a few important updates to several of the Enemy ITA Spawn related edits:

1 - Update to **Part 1: Spawn Enemies from ITA (including ETM) models** - fixes issue when an enemy respawns the speed/scale wouldn't be applied if using AppearanceType = 0x3. Enemies will now persistently spawn (if they are not killed), when re-entering the room. Update can be viewed **HERE**.

2 - Update to **ITA Enemies Persistence & ETS ID's** can be seen **HERE.** (compatibility with 9c edit above).

IMPORTANT - **It should be understood that all these ITA updates, and edits need to be installed to work properly. Installing some of them, or not updating them could cause them not to work properly. Also, don't forget to have the 'Sanity Check' installed for all the ITA edits and updates.**

What's new in v7,3 (2-13-2023)

Sanity Check

This is a function that will heavily reduce chances of crashing when future edits are made. If we are using any of the newer ITA bio4 edits, this **must** be installed. Also, **all new edits from this point on will require this edit**.

Updated Edits in v7.2 (1-13-2023)

Part 1: Spawning Enemies from ITA (including ETM models)

Enemies spawned via AEV-ITA now have more options (scale, speed, sight range etc).

Part 9a: AEV-ITA Enemy Spawn Persistence & ETS ID's

Enemies spawned via AEV-ITA will be persistently spawned if we re-enter the room. Also, Enemies spawned via AEV-ITA can now be assigned an ETS_ID, so we are able to use their deaths as toggles as the ability to assign ammo/item/weapon/treasure drops from them.

MISC MODIFICATIONS

Verdugo No Teleport (Every Room Except r221)

What's new in v7,2 (1-08-2023)

Part 8b: DLL Key inconsistency Issue Fix ***updated 1-12-2023***

Lock states will now save when jumping into the same room. (tested with debug jump and AEV Jump).

Part 8c: Companion DLL Keyless Unlock
Allows events paired with Type 11 events (linked with an **events.cfg file**), to **not require** keys to have their states saved. This is used in conjunction with the DLL continuity edit to allow other AEV's to have persistent states after returning to room. Does not require that these events be in a chain. See more below.

Updated Edits in v7.2 (1-07-2023)
In this version of the guide (v.7.2) there is an important update to the **AEV Event State Continuity** edit. The previous version did not account for players landing in TriggerZones as soon as the room loads, so the new patch will ensure that continuity bytes are loaded before the execution of any AEV entries in the room. The previous version of this edit remains in this document for users that may have already installed it, as you **must revert** the old edit prior to installing the new one.

Part 1: Spawning Enemies from ITA (including ETM models) *updated 2-2023*
Added support for enemies who throw weapons infinitely. See section below for more details.

Updated Edits in v7.1 (12-23-2022)
In this version of the guide (v.7.1) there is an important update to the AEV-ITA enemy spawn edits. Some extra functionality was added (being able to spawn enemies via ITA with outfits, weapons and health). It is important to note that if you have already put these edits to use you will need to edit your ITA files. I suggest anyone that has installed any of the edits below to reinstall them. Also please see the updated documentation as some offsets have been changed:

Part 1: Spawning Enemies from ITA (including ETM models)
Part 9: Spawning Enemies, Items, Treasures & Weapons with AEV

It should be noted that neither of these edits requires the Companion DLL.

Updated Edits in v7,0 (10-31-2022)
In this version of the guide (v.7) there have been some important updates to some of the edits. The most important update is the Event State Continuity edit. If you already have installed any of the edits below then you should definitely update the section in those edits that are shown with a red caption of the update (instead of updating the entire edit).

Event State Continuity Chain
ESE-AEV link
Custom CES (Chapter Ending Screens)

What's new in v7 (10-31-2022)
Use Ashley Buttons in r20e
Leon Can Do Ashley Actions in r20d

Updated Edits in v6 (9-07-2022)

In this version of the guide (v.6) there have been some important updates to some of the edits. The most important update is the AEV Chain edit, which now requires AEV-FSE **and** AEV-ESE edits to work. These were updated to fix overlapping function calls, and to add functionality for older edits to work with newer ones.

AEV-FSE Link
AEV Chain
AEV Timer

What's new in v6
Lever Timer
Allow Custom CES (Chapter Ending Screens)
Adding Additional BGM Files
Doorway AEV's Can Spawn/Despawn Ashley
r106 EM Sound Fix
Drawn Enemies During Camera Events
U3 No Transform and allows Death
Added ESE-AEV link
FSE TriggerZones can be Overlapped
EM18 Invincible to Incendiary Grenades
Merchant Initialization (r100)

TABLE OF CONTENTS

(sections in **red** should not be skipped over, as they contain important information)

IMPORTANT UPDATE NOTICE

PREAMBLE

HOW TO COPY & PASTE INTO THE BIO4.EXE

POINTER EDIT

DEMO VIDEO

AEV EVENT MODIFICATIONS

Part 1: Toggling Effects with an AEV event

Part 2: Setting a timer on an AEV event

Part 3: Adding player mortality to AEV 'Type A' events

Part 4a: Changing the Parameters of some AEV Type A events

Part 4b: Changing the Parameters of some AEV Type A events

Part 5: AEV Type A 'Damage Type' changes from 0A to 08 upon reaching 1 health

Part 6a: Adding ETM Lever functionality to toggle AEV events

Part 6b: Adding ETM Lever Timer

Part 7: Chaining AEV events

Part 8: Event State Continuity Chain

Part 9: AEV–ITA Functionality

Part 9a :AEV-ITA Enemy Spawn Persistence & ETS ID's

Part 9c - EM ITA UDAS Pre-Load

Part 9d - Universal ETS ID Access

Part 10: Type 6 AEV Toggle

Part 11: AEV Options

Part 12: AEV - Auto-Door Block

Part 13: AEV - Discard AEV

Part 14: AUTO Message/CAM/Sound)

Part 15: AEV Checkpoint

ITA EVENT MODIFICATIONS

Part 1: Spawning Enemies from ITA & ETM models

FSE EVENT MODIFICATIONS

Part 1: FSE - AEV link

ESE MODIFICATIONS

Part 1: ESE - AEV Toggle

Part 2: ESE's Paused Instead of removed on Companion DLL Movie

CAM MODIFICATIONS

Par1: Control Camera Visibility/Freezes

Part 2: ITM Shown in Type 6 Camera Shots

Part 3: AEV-CAM Toggle

Part 4: No Escape Auto Cam (Type 2 events)

Part 5: Effects Keep Playing During Cam

Part 6: Drawn Enemies During Camera Events

MISC MODIFICATIONS

ROOM INIT MODIFICATIONS

OUTRO

PREAMBLE

AnonymousUser (who has asked to remain anonymous for now), and myself have been hard at work modifying the binary **bio4.exe** to work some extra functionality with the game. It started with some simple things, and has progressed

into large scale functionality that never before existed for modding this game. In the spirit of mutual progress this document aims to share this extra functionality with other modders in the modding community.

What can this do?

Basically, we will learn in this guide how to edit the bio4.exe to gain added functionality. Examples include toggling effects, sounds, and enemy spawns with AEV events, putting timers on AEV events, or removing game functions like removing Leon's model from the inventory window and so on.

How is this done without a DLL?

Normally when we think of added functionality, we think of a .DLL file that is used with a config file, like the Companion DLL, or the RE4Tweaks DLL. These DLL files inject code into memory to alter the functions of the game. Another way to do this is to simply edit the code directly into the bio4.exe file.

While I do not know of all the sorcery involved, I can explain some things in simple terms. Before starting all these edits, I informed AnonymousUser about huge swaths of unused data in the bio4.exe. I noticed huge sections of just **CC CC CC CC** (called Code Caves), and so he was able to use Cheat Engine to make many of the functions, and then hash out raw HEX for me to paste into the bio4.exe in the unused sections. Many of the edits in this guide will use custom flags in the game files to activate certain functionality which will be explained in detail for each edit.

Some other methods involve piggybacking memory space that is used by Raz0r's DLL, but I really don't know how all that works. What I do know is that this level of wizardry by AnonymousUser is unparalleled, even with some of the accomplishments by the master modder Raz0r.

Prerequisites for edits

For this tutorial we will be editing the **1.0.6 version** of the bio4.exe file. Users should already have a basic understanding of the concepts of the individual tools as described in my previous tutorials, especially those described in the **Universal Concepts Guide**.

Also, many of the AEV functions utilize a toggling method, so users should have an understanding of this. Please see my **Toggling AEV Events** tutorial

Also - It should be noted that many of these edits write to memory that is accessed by the Companion DLL, so that DLL is required for most edits (any edit with **00 0E 2E 10**).

Also - Make sure to read the pinned section above about 'Interdependent Edits', as there are several important edits that need to be installed in the correct order.

HOW TO COPY & PASTE INTO THE BIO4.EXE

Important info about editing the bio4.exe

First of all, you can use any hex editor for these edits, but for the purpose of this guide I will explain how to make these edits using the freeware HxD Hex editor.

It should be noted that while making these edits to the bio4.exe that it is **imperative** that we **do not change the overall length** (file size) in any way. This is important because we do not want to change any of the addresses of the pointers to the functions. Please make sure to read this section thoroughly before making any edits, and please also remember to

BACK UP YOUR bio4.exe before making and edits. If you happen to forget, you can download the bio4.exe (1.0.6) HERE.

To ensure that we do not change the overall length of the file we should be especially careful while pasting, always remembering to ensure that what we are pasting is the same length as the field we are pasting into.

How to properly copy & paste in the hex editor

There are two types of edits that we will encounter in this guide;

1 - '**Change to**'
2 - '**Find & Paste**'

Now let's look at the steps that are involved with each method described above:

'Change to' Edits

With edits that ask us to '**Change to**' I simply copy the provided array and search for it in HxD. When the search results come up, the string is highlighted in HxD, so I simply paste what is in the clipboard while the original string is highlighted in HxD. The newly pasted string will appear in red to indicate that bytes have been replaced (but not yet saved).

'Find & Paste' Edits

With edits that ask us to '**Find**: 004BA690 (for example), then the method is to:
1 - use (CTRL-G in HxD program) which brings us to this address. At this point the cursor will be blinking at the address where we need to paste the array of bytes.

2 -The next step is to copy the provided array, and to paste it into a new document in HxD using CTRL + B. **NOT CTRL V - DOING THIS WILL RESULT IN YOUR EXE BEING THE WRONG LENGTH.** Using **CTRL + B** inserts the code over the existing code, and will not change the length of your bio4.exe.

POINTER EDIT

Important Pointer Edit before all other edits

(The Most Important Edit to Have)

The following edit essentially updates a pointer that is used in almost all of the edits provided in this document. If an edit below already contains this edit, you can skip that part. Any edit that has **00 0E 2E 10** needs this edit. Also it should be noted that any edit that has **00 0E 2E 10** also needs the Companion DLL.

HEX CODE TO ACTIVATE POINTER FUNCTIONALITY

83 C4 08 81 60 54 FF FF FF EF 5F 5E 5B 8B E5 5D C3 CC CC CC CC CC
Change To:
A3 00 0E 2E 10 83 C4 08 81 60 54 FF FF FF EF 5F 5E 5B 8B E5 5D C3

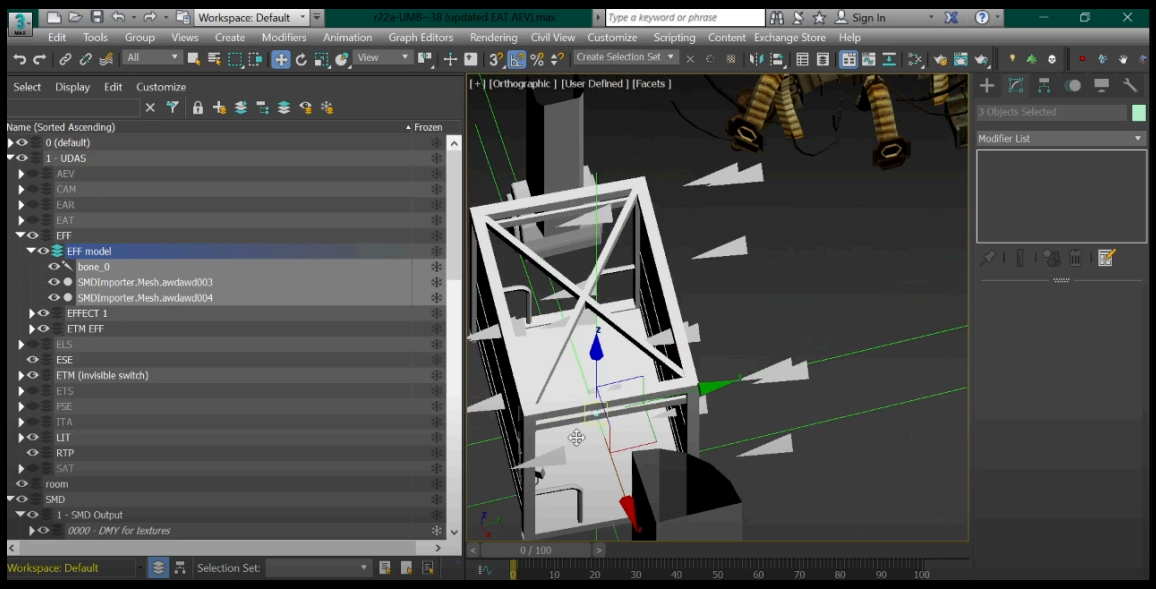
Sanity Checking Function (IMPORTANT)

This is a function that will heavily reduce chances of crashing when future edits are made. If we are using any of the newer ITA bio4 edits, this **must** be installed. Also, all new edits from this point on will require this edit.

Find:offset **001B2368**
Paste:
51 8B 5C 24 08 8B 44 24 0C 85 DB 74 20 0F B6 4B 06 8D 5B 10 39 D8 72 15 69 C9 9C 00 00 00 8D 9C 19 9C 00 00 00 39 D8 77 04 59 B3 01 C3 59 30 DB C3

DEMO VIDEO

This video is a little older, and contains examples using older edits. While not all information in the video is current, as newer techniques & edits have been developed, many of the concepts are the same. Click on the image below to open the video



AEV EVENT MODIFICATIONS

AEV EVENT MODIFICATIONS

Part 1: Toggling Effects with an AEV event

Some of the edits done to the binary include interaction with the files we use, one of which is the AEV. Some added functionality will work if we edit certain bytes that were not being used before. In this first example we will explore the functionality of the **EAR** file, which essentially allows us to toggle effects when entering certain predetermined TriggerZones. Normally this function is simply a matter of walking into an area to toggle the effect, but with this edit we are able to add a layer of functionality by linking the EAR TriggerZone with the AEV file. In this example we will be looking at the AEV and EAR files as extracted by the **Son of Persia 2018 toolset.**

Offset[140..143] = 0xEBXX (where **XX** is we put the EAR **Offset[2]** value.
Using this offset in an AEV entry will link the AEV event to an EAR entry that both have overlapping TriggerZones. What this allows us to do is toggle effects with an AEV. So for example, we can make a modified type 2 TriggerZone that will trigger the AEV event without a prompt, and when it is triggered it will load all the effects that are in a specified **EFFECT1\Group X** folder.

Why is this important if we can already do this with an EAR TriggerZone? Well, we know now that we can toggle AEV events by destroying enemies or ETM models, and so it follows that activating an AEV this way (or deactivating), will in turn toggle the effect. So an example of having an AEV entry with the **Offset[140..143] = 0xEBXX** would be:

1 - player walks into an EAR\AEV zone. Nothing happens
2 - player shoots enemy coming at him
3 - a wall of fire erupts (EFF folder is activated) and kills the player (AEV Type A is activated).

Linking an AEV event with the EAR entries

With this edit the AEV and EAR files are linked in several ways that work in tandem:

- The AEV and EAR TriggerZones need to overlap (the AEV can be smaller and in some situations we will want the EAR TriggerZone to be the size of the entire room). -
- The AEV entry will need to have the **custom flag** at the right offset to activate the link between the files (**Offset[140..143] = 0xEBXX**).

This being said, the byte that we chose in the EAR entry will determine which effect is activated when the AEV is activated. This is done with the EAR entry **Offset[2] = 0xXX** The value must be in the range of **11** to **1D**.

The second part to this is the linkage between the EAR entry and the **EFFECT1\Group X** folder. This is done by editing the **Table_2.txt** file. In the example below each Entry corresponds to an Effect Group folder, (for example Entry_1 correlates to all the effects in Effect Group 1). Let's look at an example of a Table_2.txt file:

EntryCount = 4

Entry_0_index = 0x0
Entry_0_unknown = 0xFFFF (this means the effects in Effect Group 0 folder will be seen anywhere, so no EAR link)

Entry_1_index = 0x1
Entry_1_unknown = 0x1 (this means the effects in Effect Group 1 folder will be seen when the player enters any EAR zone with **Offset[2] = 0x1**)

Entry_2_index = 0x2
Entry_2_unknown = **0x12** (this links all the effects in **EFFECT1\Group 2** folder to any EAR entry with: **Offset[2] = 0x12**)

Entry_3_index = 0x3
Entry_3_unknown = **0x13** = (this links all the effects in EFFECT1\Group 2 folder to any EAR entry with: **Offset[2] = 0x13**)

And now let's look at the linking AEV events:

Event 39
Type 2 - Fire Effect - EAR 12 link

ID = 0x1
TriggerZoneHeightBoundary1 = 1082.875000
TriggerZoneHeightBoundary2 = 2169.542725

UnknownValue = 750.000000
TriggerZoneCorner0_X = 106117.984375
TriggerZoneCorner0_Y = -9084.941406
TriggerZoneCorner1_X = 110010.875000
TriggerZoneCorner1_Y = -9163.832031
TriggerZoneCorner2_X = 109901.187500
TriggerZoneCorner2_Y = -18232.496094
TriggerZoneCorner3_X = 106172.046875
TriggerZoneCorner3_Y = -18263.816406
Offset[52] = 0x2
Type = 0x2
Index = 0x34
Offset[55] = 0x0
Offset[56] = 0x1
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
PromptMessage = 0x0
Offset[75] = 0x0
# Cut Scene Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0x0
Offset[140..143] = 0xEB12 - (this is value we set in Entry_2 in the **Table_2 EAR** file above)
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

And the same kind of entry would follow for the next event, but using **Offset[140..143] = 0xEB13** that links with Entry_3 in the EAR)

Summing it up

Any AEV event with the **Offset[140..143] = 0xEBXX** byte and a TriggerZone that overlaps an EAR TriggerZone will load whatever effects are triggered by the EAR for that TriggerZone , **ONLY WHEN THAT AEV EVENT is active** and the player is within the boundaries of the TriggerZone. It should be noted that for these custom EAR/AEV triggers we can only use values of **11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C,** and **1D**.

HEX CODE TO ACTIVATE FUNCTIONALITY

D3 E0 09 45 EC F6 46 32 01 74 0F A1
Change To:
EB 75 09 45 EC F6 46 32 01 74 0F A1
---

09 10 8A 4E 35 80 F9 02 75 0A 83 7E 40 00 0F 84
Change To:
E9 14 03 00 00 80 F9 02 75 0A 83 7E 40 00 0F 84

Find: offset 002BA3D0
Paste:
80 BE 8C 00 00 00 FC 75 06 80 7E 34 03 74 0A 09 10 8A 4E 35 E9 D3 FC FF FF C6 05 10 0E 2E 10 01 EB ED

Find: 001FE960
Paste:
~~80 F9 **11** 7C 05 80 F9 **1D** 7E 07 D3 E0 E9 7A FF FF FF 80 3D 10 0E 2E 10 00 75 02 B1 00 C6 05 10 0E 2E 10 00 EB E5~~

UPDATED - 12/2/2023
80 F9 **11** 7C 05 80 F9 **1D** 7E 07 D3 E0 E9 7A FF FF FF 50 53 BB 10 0E 2E 10 31 C0 88 C8 80 3C 18 00 75 0A B1 00 C7 44 24 04 00 00 00 00 C6 04 18 00 5B 58 EB D6

---
11 to **1D** is the range of values mentioned above.

AEV EVENT MODIFICATIONS

Part 2: Setting a timer on an AEV event

Special Update Note:
This edit has been updated since the previous versions, and will replace any previously published edits for this. It is really important to update this and any other previously installed edits.

About Using Timers

It is now possible to set timers on any AEV event so that after a certain amount of time the TriggerZone will become permanently inactive (for now this only applies to when we are in the room, as the timer presently resets when re-entering the room). This edit essentially instructs the event to 'time out' or stop working once the set timer value runs out. This is useful for temporary events, say for example, a **Type A** damage event like an explosion that only hurts the player for a short duration of time. This is done with entering custom flags into two offsets of the extracted AEV .txt file:

Offset[144..147] = 0xFD000000 (This simply activates the timer functionality for the entry).
Offset[148..151] = 0x257 (this is the amount of time until the AEV event activation expires.

The value of the timer amount is shown as a 4 Byte decimal number. For reference:
900 = 15 Seconds
1800 = 30 Seconds
3600 = 60 Seconds

It should be noted that an event's timer is only started once it is activated. So if you have an external method of triggering an AEV event (say by breaking an ETM model), then the timer starts as soon as the object is broken. If there is no external mechanism controlling the activation status of the AEV event, then the timer will start as soon as the room is loaded. It should also be noted that if a timer has started and we leave the room, then the timer goes straight to 0 and resets when re-entering the room.

HEX CODE TO ACTIVATE FUNCTIONALITY

F6 40 34 01 0F 84 F7 FB FF FF 38 58 35 0F 85 EE

Change To:
F6 40 34 01 E9 58 05 00 00 90 38 58 35 0F 85 EE

F6 47 34 01 74 58 0F B6 4F 36 51 E8 B4 44 D4 FF
Change To:
E9 A1 08 00 00 90 0F B6 4F 36 51 E8 B4 44 D4 FF

Find: 002BD530
Paste:
66 9C 80 B8 93 00 00 00 FD 75 1E 80 78 34 03 75 18 81 B8 94 00 00 00 00 00 00 00 7F 06 C6 40 34 02 EB 43 FF 88 94 00 00 00 66 9D 0F 84 6F F6 FF FF E9 73 FA FF FF 80 BF 93 00 00 00 FD 74 15 75 04 80 40 01 01 F6 47 34 01 0F 84 9F F7 FF FF E9 42 F7 FF FF 80 78 01 02 75 E7 C7 87 94 00 00 00 00 00 00 00 EB DF 51 52 53 31 DB 8A 58 36 6A 00 53 E8 B6 45 D4 FF 83 C4 08 5B 5A 59 EB AB

02 = number of events
if the number is set to say 4 for example, all events with timer will require 4 events to have the fix work. ???

AEV FLAGS
Offset[144..147] = 0xFD000000 must be used to activate
Offset[148..151] = 0xXXXX is timer amount

Calculation Example (say you want 45 seconds):
1800 = **30** Seconds
(1800/30)*45 = **2700**

AEV EVENT MODIFICATIONS

Part 3: Adding player mortality to AEV 'Type A' events

While there are some AEV 'Type A' events that will kill Ashley, there do not seem to be any that kill Leon. There is one that is a 'soft kill' that totally immobilizes Leon, but the '**You Are Dead**' screen is not triggered and we are forced to reload. So adding the custom flag **Offset[152..155] = 0xFE** to any TYPE A event will eventually kill Leon forcing the 'You Are Dead' screen to load, and prompt for continue.

This applies to Ashley as well, but the event must be set to affect her (see **AEV EVENT ACTIVATION RESTRICTIONS**).

In my experience it is best to always make two separate AEV Type A events when dealing with Ashley; One for her, and one for Leon. If we use the Activation types that allow both of them to be affected it often damages both players, even if just one of them is in the event TriggerZone. For example, the player is far away from Ashley on the other side of the room while Ashley walks through a Type A event TriggerZone that kills her, and this in turn kills Leon.

HEX CODE TO ACTIVATE FUNCTIONALITY
8B 45 0C 6A 01 6A 00 50 56 E8 72 87 D8 FF 0F B6
Change To:
E9 9C 01 00 00 6A 00 50 56 E8 72 87 D8 FF 0F B6

Find: offset 0027E000
Paste:
8B 45 0C 8B 0C 24 83 F9 00 74 09 80 B9 98 00 00 00 FE 74 07 6A 01 E9 49 FE FF FF 6A 00 EB F7

AEV EVENT MODIFICATIONS

Part 4a: Changing the Parameters of some AEV Type A events

It should be noted that while in some ‘Type A’ event TriggerZones, Ashley will be continually stunned with her stun animation repeating much too often. This next edit allows us to change the frequency of the intervals at which she is stunned.

Ashley Stun Fix

(changes time interval between damage while in TriggerZone)
Time for Ashely to be stunned then return to normal animations **1E** = 30

Find: 0035FC97
1E
Change To:
53

AEV EVENT MODIFICATIONS

Part 4b: Changing the Parameters of some AEV Type A events

This next edit allows us to change the Activation Type of AEV events for Ashely which changes which animations are used when she is killed. There are several different animations (some with different camera angles) that are used depending on the Activation Type used. With this edit we can only choose one type that is used, to be determined by the byte seen in **red** below.

Ashley AEV Type A Kill Function

Changes Type A activation type when Ashley reaches 1 health

HEX CODE TO ACTIVATE FUNCTIONALITY

8B 45 0C 6A 01 6A 00 50 56 E8 25 69 CA FF D9 45
Change To:
E9 7F FE FF FF 6A 00 50 56 E8 25 69 CA FF D9 45
---
74 06 8A 4E 5C 88 4D BC 51 0F B6 4E 60 F6 C2 02 8B
Change To:
74 06 8A 4E 5C 88 4D BC E9 0D 01 00 00 F6 C2 02 8B

Find: 0035FB30
Paste:
8B 45 0C 8B 0C 24 83 F9 00 74 09 80 B9 98 00 00 00 FE 74 07 6A 01 E9 66 01 00 00 53 8B 1D 00 0E 2E 10 66 83 BB B8 4F 00 00 01 5B 7F E7 6A 00 EB E5
—---
Find: 002B2DF0
Paste:
51 80 BE 98 00 00 00 FE 74 09 0F B6 4E 60 E9 E0 FE FF FF 50 A1 00 0E 2E 10 66 83 B8 B8 4F 00 00 01 58 75 E6 B1 **09** EB E6
(**09** = AEV Type A activation type it changes to when Ashley reaches 1 health)

AEV EVENT MODIFICATIONS

Part 5: AEV Type A ‘Damage Type’ changes from 0A to 08 upon reaching 1 health

(this allows us to use one type of animation for regular damage, and another custom one for death)

Normally when we take damage there is a certain animation used, and with the added functionality of mortality in AEV Type A events (described in Part 3 above), we can set a different animation to be used when dying. This simple edit allows us to switch the animation that is seen when we die from a Type A event.

HEX CODE TO ACTIVATE FUNCTIONALITY

82 A3 00 00 00 D9 45 18 53 8A 5D 08 D9 C0 DD 05

Change To:

82 A3 00 00 00 D9 45 18 53 E9 FA 11 00 00 DD 05

Find: 0027F090

Paste:

8A 5D 08 D9 C0 66 3D 00 00 75 07 80 FB 0A 75 02 B3 08 E9 EF ED FF FF

AEV EVENT MODIFICATIONS

Part 6a: Adding ETM Lever functionality to toggle AEV events

While we now know that most AEV events can be toggled by killing an enemy or breaking an ETM model (lever functions are for some reason not recognized to toggle AEV events in the vanilla game. With the following edit we can use levers to toggle AEV events using the following custom flags at these offsets:

Offset[52] = 0x3 (**0x3** is default active while **0x2** is default inactive)
Offset[68..71] = 0x22020008 (**22**= ETS ID)
(In the example above we would see the AEV event disabled when the ETM model linked to ETS ID **22** is broken. This can be done in vanilla, but now works for levers).

Now we have the same functionality as other breakable ETM models with the lever, however we have the ability to turn it back on (or turn it off again)! Another feature that was added to toggling on and off is that we also have added a '**one time use only**' functionality with the following custom flags at these offset values:

Offset[52] = 0x0 makes lever operable one time only (event is disabled by default)
Offset[52] = 0x1 makes lever operable one time only (event is enabled by default)

Easter Egg: Did you know that doors we can kick down also can be used as a trigger to toggle events? A kicked down door is registered by the game in the same way that a broken vase is! (this can be done without this edit).

HEX CODE TO ACTIVATE FUNCTIONALITY

83 7D 10 00 74 1A 8B C6 E8 80 B4 FF FF 50 56 68

Change To:

EB 2B 90 90 74 1A 8B C6 E8 80 B4 FF FF 50 56 68

---

89 90 14 02 00 00 C7 81 FC 00 00 00 01 02 00 00

Change To:

E9 D7 FE FF FF 90 C7 81 FC 00 00 00 01 02 00 00

---

89 90 14 02 00 00 C7 81 FC 00 00 00 01 01 00 00

Change To:

E9 45 FF FF FF 90 C7 81 FC 00 00 00 01 01 00 00

Find: 0021B438 -- (There's multiple hits of this next part in green so make sure it's at this offset)
5F 5E 89 0A 5B 5D C3 C7 02 00 00 00 00 39 04 8D
Change To:
5F 5E E9 B1 04 00 00 C7 02 00 00 00 00 39 04 8D

Find: 0021B850
Paste:
83 FB 65 75 07 BB 0E 00 00 00 EB BE 83 7D 10 00 EB C5

Find: 0021B8F0
Paste:
89 0A 83 FB 0E 74 03 5B 5D C3 50 56 31 C0 8B 75 14 8A 5E 36 8A 7E 34 8D 89 48 06 00 00 38 1C 08 74 08 66 81 3C 08 00 00 75 2A 66 89 1C 08 80 7E 34 02 7D 25 81 79 C0 00 00 00 00 75 1C 81 BE 94 00 00 00 00 00 00 00 74 10 8B 9E 94 00 00 00 89 59 C0 EB 05 83 C0 02 EB C4 5E 58 EB AA

Find: 001DD6A0
Paste:
66 83 B8 40 02 00 00 00 75 14 89 90 14 02 00 00 83 FA 00 0F 84 11 01 00 00 E9 9E 00 00 00 50 51 52 31 C9 8D 80 40 02 00 00 51 50 31 D2 F6 44 08 01 01 75 01 42 80 7C 08 01 02 7D 07 C7 40 F0 00 00 00 00 6A 00 6A 00 52 8A 14 08 52 E8 6B 44 E2 FF 83 C4 10 58 59 83 C1 02 66 81 3C 08 00 00 75 C8 5A 59 58 EB A4

AEV EVENT MODIFICATIONS

Part 6b: Adding ETM Lever Timer

NOTE: This edit requires the previous Lever edit 6a to work

This is a simple edit that gives us the ability to add a timer (or interval rate), at which the levers are allowed to function. This was added because without it, levers can be pulled at will and looks quite odd when attached to door events etc. This could also come in handy if we are trying to create scenarios in the same room where we have to come back to the lever some time later. This makes it so there is an interval rate at which the levers can be pulled.

AEV Activation byte needs to be **Offset[52] = 00/01**, and then we need to put in the AEV timer:
Offset[148..151] = **0x8C** (or value shown below)

The timer starts after the lever stops turning.
500(DEC) = **000001F4** (HEX) = 8.3 Seconds

HEX CODE TO ACTIVATE FUNCTIONALITY

83 BE 18 06 00 00 00 0F 84 80 01 00 00 8B 0D D4
Change To:
E9 42 FF FF FF 90 90 0F 84 80 01 00 00 8B 0D D4

Find: 001DDBB8
Paste:
81 BE 08 06 00 00 00 00 00 00 75 0C 83 BE 18 06 00 00 00 E9 A8 00 00 00 81 BE 38 06 00 00 00 00 00 00 75 E8 FF 8E 08 06 00 00 81 BE 08 06 00 00 00 00 00 00 75 D6 C7 86 38 06 00 00 00 80 BB 44 EB CA

AEV EVENT MODIFICATIONS

Part 7: Chaining AEV events

IMPORTANT - Updatated Edit Notice:
AEV Chain now requires AEV-FSE and AEV-ESE and AEV Checkpoint edits to work
——----------

This particular edit is really useful for creating complex scenarios that involve multiple AEV events that work together. What this edit allows us to do is link AEV events so that activating one can activate a 'chain' of other events. The way this works is simple; a 'leader' is set with a certain custom byte and the rest of the linked events are set to be 'followers' that are set with another byte.

It should be noted that the '**leader**' and '**followers**' need to always be in a certain order (that is to say, we can not skip over entries in the .txt file... all linked entries must be next to one another in the .txt file). This being said we can set the direction to be forward or backward (up or down).

So let's look at a practical use for this; in the following example we will see that the first event is triggered by a lever, and then the rest of the chained events are activated as a result:

Event 1
Blocking for Door 1 (**THIS IS THE LEADER that other events will be activated from**)

ID = 0x1
TriggerZoneHeightBoundary1 = 582.492676
TriggerZoneHeightBoundary2 = 3970.652344
UnknownValue = 750.000000
TriggerZoneCorner0_X = -1264.050781
TriggerZoneCorner0_Y = 88988.484375
TriggerZoneCorner1_X = 1516.228516
TriggerZoneCorner1_Y = 86914.539063
TriggerZoneCorner2_X = 1210.981445
TriggerZoneCorner2_Y = 86530.375000
TriggerZoneCorner3_X = -1530.459961
TriggerZoneCorner3_Y = 88673.101563
Offset[52] = 0x2 (this value makes this entry **inactive** until the **ETM model** in **Offset[68..71]** is broken. See more HERE
Type = 0xB
Index = 0x20
Offset[55] = 0x1
Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..71] = 0x03020008
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
# Map Blocking Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0

Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0xFB - '**FB**' value sets this entry as the 'leader' and instructs the next entry forward to be included in the chain.
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..15t5] = 0x0

Event 2
Blocking for Door 2 (this is the first 'follower' event. Since this entry uses **Offset[52] = 0x2** it is only activated when the previous event is activated)

ID = 0x1
TriggerZoneHeightBoundary1 = 582.492676
TriggerZoneHeightBoundary2 = 3970.652344
UnknownValue = 750.000000
TriggerZoneCorner0_X = -1264.050781
TriggerZoneCorner0_Y = 88988.484375
TriggerZoneCorner1_X = 1516.228516
TriggerZoneCorner1_Y = 86914.539063
TriggerZoneCorner2_X = 1210.981445
TriggerZoneCorner2_Y = 86530.375000
TriggerZoneCorner3_X = -1530.459961
TriggerZoneCorner3_Y = 88673.101563
Offset[52] = 0x2 (entry is **inactive** but becomes activated by the previous entry in the chain that used **Offset[136..139] = 0xFB**
Type = 0xB
Index = 0x21
Offset[55] = 0x1
Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
# Map Blocking Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0

Offset[136..139] = 0xFF (this sets the **next** entry to be a 'follower of the this entry, and keeps this entry as part of the chain)
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Event 3
Blocking for Door 3

ID = 0x1
TriggerZoneHeightBoundary1 = 582.492676
TriggerZoneHeightBoundary2 = 3970.652344
UnknownValue = 750.000000
TriggerZoneCorner0_X = -1264.050781
TriggerZoneCorner0_Y = 88988.484375
TriggerZoneCorner1_X = 1516.228516
TriggerZoneCorner1_Y = 86914.539063
TriggerZoneCorner2_X = 1210.981445
TriggerZoneCorner2_Y = 86530.375000
TriggerZoneCorner3_X = -1530.459961
TriggerZoneCorner3_Y = 88673.101563
Offset[52] = 0x3 (active by default but is deactivated and is part of the chain as the previous entry uses **Offset[136..139] = 0xFF**
Type = 0xB
Index = 0x22
Offset[55] = 0x1
Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
# Map Blocking Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0x0 (The final 'follower' in the chain does not and **SHOULD NOT** have **FF**). Leaving the final entry at **0x0** ends the chain, but this event is still linked from the previous event using **Offset[136..139] = 0xFF**
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Breakdown of values

Offset[136..139] = **0xFA** = Sets entry as 'leader' for previous events (we would go backwards up the AEV file with this as the leader)
Offset[136..139] = **0xFB** = Sets entry as 'leader' for next events (we would go forward down the AEV file with this as the leader, as in example above)
Offset[136..139] = **0xFF** = Sets entry as 'follower' and continues the chain
Offset[136..139] = **0x0** = Sets entry as 'follower' and stops the chain

As you can see, the chain can go in either direction by using either **0xFA** or **0xFB** for the 'leader'. That is to say, in the previous example, the 'leader' was the first entry, and the 'followers' in the chain followed with each new entry, but we can create chains that go backwards as well, staring with **0xFA** for the leader value, and work our way backwards (or up in the .txt file)

Note: Do not try to chain back the first AEV entry, and try not to chain forward the last AEV entry

Unidirectional Chains

Beside creating one way chains, we can also create a **unidirectional chain**, or a '**Twin**' chain that consists of two leaders that are linked. This allows us to create a '**toggle**' for two linked events. In the following example, we have two events that turn each other on and off, each with their own separate lever. In this example we have an two separate bloom effects that are toggled (one red and one green):



In this scenario BOTH switches can activate and deactivate the bloom effects. Since one event has **Offset[52] = 0x3** and the other has **Offset[52] = 0x2** this means only one effect is on at a time. Whenever either of the levers is operated, the effects are toggled (these effects are toggled via EAR link.. see Part 1 of this guide for more detailed information on how to link effects to AEV events).

Twin Chain example:

Event 3
Twin Chain - Red Bloom *linked with next event * FA

ID = 0x1
TriggerZoneHeightBoundary1 = 1082.875000
TriggerZoneHeightBoundary2 = 2169.542725
UnknownValue = 750.000000
TriggerZoneCorner0_X = 106117.984375
TriggerZoneCorner0_Y = -9084.941406
TriggerZoneCorner1_X = 110010.875000
TriggerZoneCorner1_Y = -9163.832031
TriggerZoneCorner2_X = 109901.187500
TriggerZoneCorner2_Y = -18232.496094
TriggerZoneCorner3_X = 106172.046875
TriggerZoneCorner3_Y = -18263.816406
Offset[52] = 0x2 (event is disabled by default)
Type = 0x2
Index = 0x3
Offset[55] = 0x1
Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..71] = 0x26020008 (**26** is the **ETS ID** of the ETM lever on the left)
Offset[72] = 0x0
Offset[73] = 0x2D
PromptMessage = 0x0
Offset[75] = 0x0
# Cut Scene Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0xFB sets the event as a 'leader' and will toggle the next entry
Offset[140..143] = 0xEB14 This is the EAR link that toggles the effect (see documentation Part 1)
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Event 4
Twin Chain - Green Bloom

ID = 0x1
TriggerZoneHeightBoundary1 = 1082.875000
TriggerZoneHeightBoundary2 = 2169.542725
UnknownValue = 750.000000
TriggerZoneCorner0_X = 106117.984375
TriggerZoneCorner0_Y = -9084.941406
TriggerZoneCorner1_X = 110010.875000
TriggerZoneCorner1_Y = -9163.832031

TriggerZoneCorner2_X = 109901.187500
TriggerZoneCorner2_Y = -18232.496094
TriggerZoneCorner3_X = 106172.046875
TriggerZoneCorner3_Y = -18263.816406
Offset[52] = 0x3 (Activation State - Event is enabled by default, but is deactivated by the chain)
Type = 0x2
Index = 0x4
Offset[55] = 0x1
Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..71] = 0x39020008 (**39** is the **ETS ID** of the ETM lever on the right)
Offset[72] = 0x0
Offset[73] = 0x2D
PromptMessage = 0x0
Offset[75] = 0x0
# Cut Scene Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0xFA also sets the event as a 'leader' and toggles previous entry
Offset[140..143] = 0xEB15 This is the EAR link that toggles the effect (see documentation Part 1)
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

To see this scenario in action please check out this YouTube Video:
(simply click the image below for link)



Customizable AEV Trigger (used in a chain)

Includes custom prompt word

I recently discovered that instead of using an ETM lever, the breaking of an ETM object, or death of an enemy, to toggle an AEV event, we can use a specialized AEV event in a chain that will allow us to toggle an event (while also using a customizable prompt word). In this next example we will use a customized AEV event to start a very short chain of events. The 'trigger' event (the one that starts the chain) will only work **once**.

In the example below we can see that we are using a '**Type 6**' AEV event as the trigger, which by default does not appear in any examples in the vanilla game. This type can be used as a general type (like **Type 2** events). In the example below the chain will start at the end (bottom part of the .txt file and work backwards):

```
# Event 19
# Type 2 - EAR link

ID = 0x1
TriggerZoneHeightBoundary1 = 1082.875000
TriggerZoneHeightBoundary2 = 2169.542725
UnknownValue = 750.000000
TriggerZoneCorner0_X = 106117.984375
TriggerZoneCorner0_Y = -9084.941406
TriggerZoneCorner1_X = 110010.875000
TriggerZoneCorner1_Y = -9163.832031
TriggerZoneCorner2_X = 109901.187500
TriggerZoneCorner2_Y = -18232.496094
TriggerZoneCorner3_X = 106172.046875
TriggerZoneCorner3_Y = -18263.816406
Offset[52] = 0x2
Type = 0x2
Index = 0x37
Offset[55] = 0x0
Offset[56] = 0x81
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
PromptMessage = 0x0
Offset[75] = 0x0
# Cut Scene Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
```

Offset[136..139] = 0x0 (this entry does not need ‘**FF**’ to be a 'follower' as there are no more events in this chain). It is still linked though by next event.
Offset[140..143] = 0xEB13 (this is the link to the EAR entry with **Offset[2] = 0x13**)
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Event 20
Type 6 - Trigger to activate EAR in previous entry

ID = 0x1
TriggerZoneHeightBoundary1 = 133.498444
TriggerZoneHeightBoundary2 = 719.622314
UnknownValue = 750.000000
TriggerZoneCorner0_X = -45753.839844
TriggerZoneCorner0_Y = 9086.477539
TriggerZoneCorner1_X = -43502.746094
TriggerZoneCorner1_Y = 8998.590820
TriggerZoneCorner2_X = -43522.011719
TriggerZoneCorner2_Y = 7060.516602
TriggerZoneCorner3_X = -45833.609375
TriggerZoneCorner3_Y = 6889.056152
Offset[52] = 0x3
Type = 0x6
Index = 0x47
Offset[55] = 0x1
Offset[56] = 0x81
Offset[57] = 0x1 (These parameter in green make the event single use only, until we reload the room)
Offset[68..71] = 0x8
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x11 (custom prompt word. In this case it is ‘SWIM’)
Offset[75] = 0x0
# Map Blocking Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0xFA (as mentioned above, this is the flag to set this entry as the 'leader', but is working backwards to the previous entry)
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

What this setup does is acts as a basic trigger (with a prompt) to activate another event in a chain. So in this example above the first entry could be anything. It could be a cutscene that is supposed to play. It could be a teleport to another room. It could be a Type A that causes damage to enemies only. The possibilities are plentiful.

HEX CODE TO ACTIVATE FUNCTIONALITY

8A 48 35 80 F9 03 74 1D 80 F9 0B 75 0E 56 8B F0
Change To:
E9 D6 04 00 00 90 74 1D 80 F9 0B 75 0E 56 8B F0

Find: 002B8920
Paste:

50 53 51 31 DB 80 B8 88 00 00 00 FA 74 22 80 B8 88 00 00 00 FB 74 20 EB 09 80 B8 88 00 00 00 FF 74 1A 59 5B 58 8A 48 35 80 F9 03 E9 FB FA FF FF BB 64 FF FF FF EB 05 BB 9C 00 00 00 8D 04 18 F6 40 34 01 75 06 83 48 34 01 EB 04 80 60 34 FE 80 78 35 0B 75 1C 50 51 52 56 8B F0 F6 40 34 01 74 07 E8 DA AB FF FF EB 05 E8 13 AE FF FF 5E 5A 59 58 E8 98 11 ED FF E8 2D F1 FF FF EB 9C

AEV EVENT MODIFICATIONS

Part 8: Event State Continuity Chain (updated 10-29-2022)

Enables game to save state of an AEV entry (enable/disabled) when used in a chain with Type 11 DLL entry. This edit will allow for lockable AEV (Companion DLL lock) to be used in a chain which will in turn make the game 'remember' what state certain events in the chain are in when re-entering the room.

Why this edit was created

With my recent discovery of how to **make locks that go over ETS events** (like ETM doors), came some interesting challenges.There were two issues that came about:

Issue 1
While being able to place a lock over ETS events (like an ETM doorway), I noticed something right away; enemies were still capable of opening the door, so all we had to do to get through one of these locked doors was to wait for an enemy to open it for us. So my solution to this was to make three separate, but linked events:

1 - a blocking Type B that prevents the enemies from going through the doorway.
2 - a lock on the door that prevents the player from accessing the doorway 'OPEN' prompt.
3 - a key mechanism that unlocks the door, while also canceling out the blocking Type B event.

But even with this setup there was another issue:

Issue 2
After unlocking events in a chain that toggled other events (like toggling Type B blocking events), certain events would toggle again when re-entering the room. This was problematic as the lock mechanism would be gone (since the Companion DLL tells the game to remember the lock state), but the activation state (**Offset[52]**) of linked events in the chain was not remembered, so in this case, the blocking part would return.

The solution
The solution was to enable Type 11 events to be used in a chain, and to create an '**Event State Continuity**' byte that would essentially tell the game to preserve the activation state (**Offset[52]**) of linked events in the chain.

Working Example of Event State Continuity

Before we proceed I recommend that you first read about my ETS lock guide (linked above), and also read the **events.cfg tutorial**
To have a DLL locking event, we will need to use the **bio4.exe** edit below, as well as have the Companion DLL with a properly configured events.cfg file.

In the following example we are making a DLL lock overtop of an ETS event to prevent the player from opening the ETM door, while also using a Type B event to prevent enemies from opening the ETM door. We are also making it so that we re-enter the room that the state of all events after unlocking are preserved. **It should be noted that these kinds of chains should be created to work backwards.** Let's have a look at a working example:

Event 17
Type B - Blocking for Enemies

ID = 0x1
TriggerZoneHeightBoundary1 = 5.132019
TriggerZoneHeightBoundary2 = 1000.000000
UnknownValue = 750.000000
TriggerZoneCorner0_X = -12675.734375
TriggerZoneCorner0_Y = 8806.472656
TriggerZoneCorner1_X = -8175.763672
TriggerZoneCorner1_Y = 8819.949219
TriggerZoneCorner2_X = -8175.763672
TriggerZoneCorner2_Y = 7319.955566
TriggerZoneCorner3_X = -12575.735352
TriggerZoneCorner3_Y = 7306.942383
Offset[52] = 0x3
Type = 0xB
Index = 0x53
Offset[55] = 0x0
Offset[56] = 0x1
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
# Map Blocking Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x400000 (these bytes make it so that only enemies are blocked)
Offset[108..111] = 0x0
Offset[112..115] = 0x1
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x54 (this is the '**Event State Continuity**' byte. See more below)
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0x0 (this byte does not need to be set to '**FF**' to become a 'follower', as it is the **last** entry in the chain)
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0

Offset[152..155] = 0x0

Event 18
Type 2 (Trigger event for chain)

ID = 0x1
TriggerZoneHeightBoundary1 = -191.578247
TriggerZoneHeightBoundary2 = 2665.466309
UnknownValue = 750.000000
TriggerZoneCorner0_X = 1473.864258
TriggerZoneCorner0_Y = 30606.484375
TriggerZoneCorner1_X = 4016.215820
TriggerZoneCorner1_Y = 30606.484375
TriggerZoneCorner2_X = 4016.215820
TriggerZoneCorner2_Y = 29338.554688
TriggerZoneCorner3_X = 1473.864258
TriggerZoneCorner3_Y = 29338.554688
Offset[52] = 0x3 (Activation Status is ON)
Type = 0x2
Index = 0x54 (this value of this byte must match **Offset[124..127] = 0x54** above and the '**EventIndex**' value in the '**events.cfg**' file below)
Offset[55] = 0x1
Offset[56] = 0x8
Offset[57] = 0x1
Offset[68..71] = 0x8
Offset[72] = 0x0
Offset[73] = 0x2D
PromptMessage = 0x10
Offset[75] = 0x0
# Cut Scene Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0xDD (custom byte that enables the Type 11 DLL **Continuity Link**. Links previous event that uses **Offset[124..127] = 0x54**)
Offset[136..139] = 0xFA (custom byte that makes this entry the 'leader' in the chain - Chaining AEV edit must be installed)
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Event 19
Type 11 - DLL link

ID = 0x1
TriggerZoneHeightBoundary1 = -191.578247
TriggerZoneHeightBoundary2 = 2665.466309
UnknownValue = 750.000000
TriggerZoneCorner0_X = 1473.864258
TriggerZoneCorner0_Y = 30606.484375
TriggerZoneCorner1_X = 4016.215820
TriggerZoneCorner1_Y = 30606.484375
TriggerZoneCorner2_X = 4016.215820
TriggerZoneCorner2_Y = 29338.554688
TriggerZoneCorner3_X = 1473.864258
TriggerZoneCorner3_Y = 29338.554688
Offset[52] = 0x3
Type = 0x11
Index = 0x55
Offset[55] = 0x1
Offset[56] = 0x1
Offset[57] = 0x1
Offset[68..71] = 0x2
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
# Item Dependent Event Parameters
NeededItemNumber = 0x3B (this byte must match the '**ItemID**' in the events.cfg file below)
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0x0
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Now let's look at the corresponding events in the **events.cfg file** (placed in in the **FILES\Stage\rxxx** directory):

event_lock0 %ETS_DOOR_LOCK% (yes you can make comments in **events.cfg file** the so long as they are between %with_no_spaces%
{
    ItemID 0x3B
    Message 0x08
    Sound 0x02
    Unknown0 0xFF
    Unknown1 0x00
    EventIndex 0x54

```
    event_unlock
    {
        ItemID 0x3B
        Sound 0x0D
        Message 0x09
        LockEventIndex 0x54
        Unlocked false
    }
}
```

Summary of Example

So in the above example, # Event 18 is locked by # Event 19 (which prevents the player from accessing the OPEN door function of the ETM model), but also acts as a trigger to toggle # Event 17 Blocking event. Once the key item is provided for the locking event the # Event 17's **Activation Status** is toggled OFF **Offset[52] = 0x2.** and # Event 19's locking mechanism is disabled.

There was **ONE PROBLEM WITH THIS SETUP** (now resolved). The blocking event would normally become enabled again when we re-enter the room. This was because by default the entry is set to be enabled with **Offset[52] = 0x3**, so when we re-enter the room, the game reads the AEV data from **file, not from memory** when it was last interacted with. To compensate for this we developed the use of **Offset[124..127] = 0x54** in # Event 17. This is called the '**Event State Continuity**' byte.

Essentially this byte allows us to preserve the state (whether an AEV entry is disabled or enabled) after it has been toggled while connected to a chain with a Type 11 (like the example above). This tells the game what state **Offset[52] = 0x3** (enabled), or **Offset[52] = 0x2** (disabled) was at after the Type 11 is activated. This simply 'records' the Activation State **(Offset[52]**) so that it does not change when we reload the room. Remember that to make all this work, the trigger must use **Offset[132..135] = 0xDD** to start the **Event State Continuity** chain.

How the Event State Continuity Chain works

The way it does this is the instruction checks to see if the Type 11 data is present when the room is reloaded (it gets wiped once we unlock something with a key and stored into memory) and if that Type 11 data is not present when we re-enter the room, the instruction tells the game to **load the opposite** value of the Activation State **Offset[52]** to preserve the state.

We can use multiple instances of this in the same chain, or skip entries. The important thing to remember is that the value in **Offset[124..127] = 0xXX** must be the index of the trigger event.

Besides the scenario illustrated above, there are many other uses for this feature. One of these can be seen in **this video**. As you can see in the video, after the lock is disabled by providing the key, there are some effects **<3** that are enabled. This is using the same type of setup above where we have an AEV event that is activated in the chain after a key event is completed. In this scenario, when we re-enter the room, the events are no longer active.

So you see that while my first example is not perfect, it does still provide some functionality, and now we see there are other scenarios in which this new function can be useful.

HEX CODE TO ACTIVATE FUNCTIONALITY

Key Toggle AEV Chain (Type 11)

88 50 3A 89 50 40 5D C3 CC CC CC
Change To:
88 50 3A 89 50 40 E9 E2 07 00 00

Find: 002B42D0

Paste:
80 B8 84 00 00 00 DD 74 09 80 B8 84 00 00 00 4D 75 1E 51 6A 00 6A 00 F6 40 34 01 75 04 6A 01 EB 02 6A 00 8A 48 36 51 E8 60 D8 D4 FF 83 C4 10 59 5D C3

HEX CODE TO ACTIVATE FUNCTIONALITY

(updated January 7 2023 -MUST revert edit below before applying)
AEV Event State Continuity

~~74 05 E9 99 90 D4 FF C3 CC CC CC CC CC CC CC CC~~
~~**Change To:**~~
~~74 05 EB 17 90 90 90 C3 CC CC CC CC CC CC CC CC~~
~~---~~

~~**Find:** 002C3FB8~~
~~**Paste:**~~
~~E8 C3 31 44 00 E9 7B 90 D4 FF EB E0~~
~~---~~

~~**Find:** 00707180 (this last part of the edit was updated 10-29-2022)~~
~~**Paste:**~~
~~50 53 51 52 56 57 8B 15 00 0E 2E 10 8B BA 38 CE 00 00 EB 74 90 31 DB 31 F6 83 C7 10 43 80 7F 7C 00 90 90 90 74 02 57 46 38 D8 74 08 81 C7 9C 00 00 00 EB E8 83 FE 00 74 48 8B BA 38 CE 00 00 83 C7 10 31 DB 8B 0C 24 90 8A 49 7C 90 90 90 81 7F 40 00 00 00 00 75 19 3A 4F 36 75 14 59 4E F6 41 34 01 75 06 80 49 34 01 EB 04 80 61 34 FE EB C4 43 38 D8 75 04 59 4E EB BB 81 C7 9C 00 00 00 EB CD 5F 5E 5A 59 5B 58 C3 83 FF 00 74 F4 8A 47 06 EB 83~~

NEW 1-2023 *be sure to revert all text with ~~strikethrough~~ above first*

E8 5F 37 D4 FF B9 10 43 C6 00
Change To:
EB 79 90 90 90 B9 10 43 C6 00

Find: 002C3FB8
Paste:
E8 E4 36 D4 FF E8 BE 31 44 00 E9 7B FF FF FF

Find: 00707180
Paste:
50 53 51 52 56 57 8B 15 00 0E 2E 10 8B BA 38 CE 00 00 EB 74 90 31 DB 31 F6 83 C7 10 43 80 7F 7C 00 90 90 90 74 02 57 46 38 D8 74 08 81 C7 9C 00 00 00 EB E8 83 FE 00 74 48 8B BA 38 CE 00 00 83 C7 10 31 DB 8B 0C 24 90 8A 49 7C 90 90 90 81 7F 40 00 00 00 00 75 19 3A 4F 36 75 14 59 4E F6 41 34 01 75 06 80 49 34 01 EB 04 80 61 34 FE EB C4 43 38 D8 75 04 59 4E EB BB 81 C7 9C 00 00 00 EB CD 5F 5E 5A 59 5B 58 C3 83 FF 00 74 F4 8A 47 06 EB 83

AEV EVENT MODIFICATIONS

Part 8b: DLL Key inconsistency Issue Fix

One rarely reported issue with the Companion DLL is that some users reported key locks not being consistent in their states after returning to a room. That is to say, sometimes users would unlock a type11 companion lock, and come back to the room and the lock would be enabled again.

The issue with the DLL was that its function to look at the room number gets updated on a different thread (multi-thread tasks) than the instructions that update the AEV's with DLL locks at the start of every room. This means that we have a

chance for the instructions to be looking at the locks of the previously loaded room when entering the next room.

Hows does this work?

This edit fixes that by not allowing the instructions to continue until the DLL's room number matches the game's room number.

HEX CODE TO ACTIVATE FUNCTIONALITY

0F B7 88 AC 4F 00 00 83 C4 08 51 B9 D4 3C
Change To:
E8 6E 01 00 00 90 90 83 C4 08 51 B9 D4 3C

Find: 002C40A0
Paste:
0F B7 88 AC 4F 00 00 50 31 C0 66 39 0D 94 01 2F 10 74 0A 3D FF FF FF 0F 73 03 40 EB ED 58 C3 *updated 2-08-2023*

New part *updated 1-12-2023*

66 8B 90 AC 4F 00 00 66 89 90 B0 4F 00 00 A1 3C 5F C0 00 0F B6 88 AE 4F 00 00
Change To:
66 8B 90 AC 4F 00 00 53 E8 54 FD FF FF 5B A1 3C 5F C0 00 0F B6 88 AE 4F 00 00
---
66 89 41 28 8B 15 3C 5F C0 00 C6 42 2A 00 C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
Change To:
66 89 41 28 8B 15 3C 5F C0 00 C6 42 2A 00 53 52 66 8B 92 B0 4F 00 00 66 89 C3 E8 22 8E 00 00 5A 5B C3 CC CC CC CC CC CC CC CC CC CC CC CC CC

E8 57 9A D2 FF 8B E5 5D
Change To:
E9 F5 04 00 00 8B E5 5D

Find: 002B6F18
Paste:
66 89 90 B0 4F 00 00 83 FE 00 74 12 66 8B 5E 6C 86 FB 66 39 D3 75 07 C6 05 96 01 2F 10 01 C3

Find: 002D8E38
Paste:
E8 5D 95 D2 FF 81 3D B5 C4 28 10 13 58 06 00 74 7F C7 05 B5 C4 28 10 13 58 06 00 C6 05 B4 C4 28 10 E9 C7 05 C8 1C 2F 10 28 DC 5F 02 C7 05 CC 1C 2F 10 E8 BF A5 F9 C7 05 D0 1C 2F 10 FF 0F B7 C0 C7 05 D4 1C 2F 10 8B CE 66 89 C7 05 D8 1C 2F 10 46 54 80 3D C7 05 DC 1C 2F 10 96 01 2F 10 C7 05 E0 1C 2F 10 01 0F 85 DB C7 05 E4 1C 2F 10 A7 F9 FF C6 C7 05 E8 1C 2F 10 05 96 01 2F C7 05 EC 1C 2F 10 10 00 E9 D5 C7 05 F0 1C 2F 10 A7 F9 FF 00 E9 76 FA FF FF

AEV EVENT MODIFICATIONS

Part 8c: Companion DLL Keyless Unlock

Allows events paired with Type 11 events (linked with an **events.cfg file**), to **not require** keys to have their states saved. This is used in conjunction with the DLL continuity edit to allow other AEV's to have persistent states after returning to room. Does not require that these events be in a chain.

This edit is really one of the best I have used yet. Essentially what this does is allows us to use a Type11, (along with whatever AEV event it is paired with), to serve as a memory bank of activation states for other events. That is to say, the paired event will become a ‘master’ that other events will link to to get their activation state upon returning to the room. For example:

We walk through a Type11 event that is paired to a DMY Type2 event that simply serves as a trigger. This gives us access to the DLL **events.cfg file** which we can use to either post a message, trigger a sound, or use a camera. We can also use it to post a blank message with no sound and no camera, so essentially it doesn’t really do anything but toggle itself. This then serves as an event that other events can link to to get their states. The next part would be to link other events to this DMY Type 2 (using the continuity byte). So for example, let's say we link a doorway to this DMY Type2 to. Now whenever we reload the room, the activation state of the doorway will be toggled.

Let's look at a working example. (In this example the events are not necessarily side by side in the AEV .txt file):
In this example we are just going to make a message that plays once, and when we return to the room, will not show again, EVER.

 # Event 3
Message

ID = 0x1
TriggerZoneHeightBoundary1 = -0.004333
TriggerZoneHeightBoundary2 = 1000.000000
UnknownValue = 750.000000
TriggerZoneCorner0_X = -711.072754
TriggerZoneCorner0_Y = 61234.445313
TriggerZoneCorner1_X = 788.927307
TriggerZoneCorner1_Y = 61234.445313
TriggerZoneCorner2_X = 788.927307
TriggerZoneCorner2_Y = 59734.445313
TriggerZoneCorner3_X = -711.072754
TriggerZoneCorner3_Y = 59734.445313
Offset[52] = 0x3
Type = 0x5
Index = 0x1
Offset[55] = 0x0
Offset[56] = 0x81
Offset[57] = 0x1 (these bytes make the event only usable once while in the room)
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
# Message Event Parameters
Offset[92..93] = 0x0
RoomMessage = 0x1
MessageCutSceneID = 0x0
MessageID = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0

Offset[124..127] = **0x07** (this is the continuity byte which is the index of the paired event with the Type11)
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0x0
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

skipping some events…

Event 16
Door Way (Room Change) to r326

ID = 0x1
TriggerZoneHeightBoundary1 = -485.628906
TriggerZoneHeightBoundary2 = 4420.777344
UnknownValue = 750.000000
TriggerZoneCorner0_X = 1112.949219
TriggerZoneCorner0_Y = 53097.511719
TriggerZoneCorner1_X = 1859.098145
TriggerZoneCorner1_Y = 53097.511719
TriggerZoneCorner2_X = 1859.097900
TriggerZoneCorner2_Y = 51798.886719
TriggerZoneCorner3_X = 1112.948975
TriggerZoneCorner3_Y = 51798.886719
Offset[52] = 0x3
Type = 0x1
Index = 0x7 (paired event index)
Offset[55] = 0x1
Offset[56] = 0x8
Offset[57] = 0x1
Offset[68..71] = 0x8
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x10
Offset[75] = 0x0
# Door Way (Room Change) Event Parameters
DestinationPoint_X = -3980.755859
DestinationPoint_Y = -5038.460938
DestinationPoint_Z = 3298.453125
DestinationFacingAngle = -0.172178
DestinationStage = 0x3
DestinationRoom = 0x26
Offset[110..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0x0
Offset[140..143] = 0x0

Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Event 17
Type 11 - Keyless Lock for continuity

ID = 0x1
TriggerZoneHeightBoundary1 = -485.628906
TriggerZoneHeightBoundary2 = 4420.777344
UnknownValue = 750.000000
TriggerZoneCorner0_X = 1102.697266
TriggerZoneCorner0_Y = 53098.343750
TriggerZoneCorner1_X = 1848.846191
TriggerZoneCorner1_Y = 53098.343750
TriggerZoneCorner2_X = 1848.845947
TriggerZoneCorner2_Y = 51799.718750
TriggerZoneCorner3_X = 1102.697021
TriggerZoneCorner3_Y = 51799.718750
Offset[52] = 0x3
Type = 0x11
Index = 0x8
Offset[55] = 0x1
Offset[56] = 0x1
Offset[57] = 0x1
Offset[68..71] = 0x2
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
# Item Dependent Event Parameters
NeededItemNumber = 0x117 (an unobtainable key ID, which prevents DLL from asking for an actual key)
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0x0
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

As we can see in the example above there are 3 events, which are not all side by side (only the last 2 are). This means that we can play the continuity byte in any AEV, wherever it is located in our .txt file and it will still be linkable with the continuity byte.

Now let's have a look at the corresponding entry in the **events.cfg file**. In the **events.cfg file**, we set **ItemID** to anything ranging from **111-FFFE** (**FFFF** will automatically unlock the AEV upon room entry). In this example we are using **ItemID 0x117:**

```
event_lock0 %Event_Continuity_for_message%
{
	ItemID 0x117
	Message 0x08 (links to entry in MDT with message that is blank. Use {0x0000}{0x0100} )
	Sound 0xFF (links to non-existent sound Cue)
	Unknown0 0xFF  (links to non-existent Cam Entry)
	Unknown1 0x00
	EventIndex 0x07
	event_unlock
	{
		ItemID 0x117
		Sound 0xFF (links to non-existent sound Cue)
		Message 0x08 (links to entry in MDT with message that is blank. Use {0x0000}{0x0100} )
		LockEventIndex 0x07
		Unlocked false
	}
}
```

So we see that setting **ItemID** to something 111 or higher is an unobtainable item, which will prevent the DLL lock from asking us for an actual key. The great part about this though is that it will still register the linked event (**EventIndex 0x07** the doorway) in the **save.extend** file and write to our save file as well. We also see that the continuity byte in **#Event 3** is the index of the paired event, so this means that when we leave and return to this room, that the state of the message should be off permanently.

What this will allow us to do then is to toggle any AEV event when we come back to a room if it is linked to an event in the method shown above.

—----------

HEX CODE TO ACTIVATE FUNCTIONALITY

8B 41 1C 8B 51 14 C7 45 FC 00 00 00 00 85 C0 7E

Change To:

E9 3F FA FF FF 90 C7 45 FC 00 00 00 00 85 C0 7E

Find: 003045F8

Paste:

81 7D 04 12 A5 27 10 75 0D 8B 45 08 66 3D 10 01 7E 04 66 89 41 08 8B 41 1C 8B 51 14 E9 A1 05 00 00

AEV EVENT MODIFICATIONS

Part 9: AEV-ITA Functionality

(previously named Spawning Enemies, Items, Treasures & Weapons with AEV)

Prerequisite Edits: **Part 10: Type 6 AEV Toggle** for this to work

Exciting new developments that now allow us to spawn enemies, items, treasures & weapons by AEV are here! What is now possible is that a player can walk through an AEV TriggerZone and the magic happens! Imagine a whole inventory of weapons appearing if you walk through a magical gateway, or perhaps a terrifying ambush of 20 hungry wolves if you happen to walk in the wrong place! There are endless possibilities!

Hows does this work?

Essentially this edit works by linking the AEV file and the ITA file together using custom values in each file. The AEV serves as a TriggerZone that activates a dormant ITA entry. Once theTriggerZone of the AEV is entered & activated, the ITA entry becomes active. This function is based on the function of the snake being able to be spawned via ITA, but builds off of that and gives us more options. This edit simply serves as a way to toggle ITA entries.

Spawning Enemies with AEV

Once we have a good understanding of how to use the AEV & ITA tools, we can now look at an example of how this works. Let's start of by looking at a scenario where we will trigger an enemy to appear, first looking at the ITA entry that will be activated by the AEV event:

Working Example - ITA entry

Item Data 4 AEV enemy spawn

Category = 0x2
TriggerZoneHeightBoundary1 = 68.316528
TriggerZoneHeightBoundary2 = 1000.000000
UnknownValue = 500.000000
TriggerZoneCorner0_X = -43901.687500
TriggerZoneCorner0_Y = -5248.697266
TriggerZoneCorner1_X = 0.000000
TriggerZoneCorner1_Y = -0.000000
TriggerZoneCorner2_X = 0.000000
TriggerZoneCorner2_Y = -0.000000
TriggerZoneCorner3_X = 0.000000
TriggerZoneCorner3_Y = -0.000000
Offset[52] = 0x2

DataBlockType = 0x3
DataBlockIndex = 0x4
Offset[55] = 0x1
Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..69] = 0x0
AppearanceType = 0x0
ETS_ID = 0x0
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
Offset[76..79] = 0x010413 (these are the bytes that determine the enemy data.. more on this in the ITA section)
Offset[80..83] = 0x1050007A (these are the bytes that determine the enemy data.. more on this in the ITA section)
Offset[84..87] = 0x3E8 (these are the bytes that determine the enemy data.. more on this in the ITA section)
Offset[88..91] = 0xEA15 (these are the bytes that link the sister bytes in the corresponding AEV entry) This is called the **Group ID.**
ItemPosition_X = 105206.734375
ItemPosition_Y = -14144.859375
ItemPosition_Z = 2167.744141
ItemNumber = 0x0
Randomness = 0x10 (these bytes must be used as seen here in this example.. they are the same as spawning a snake)
Offset[122..123] = 0x0
Amount = 0x0

Offset[126..127] = 0x5
AuraType = 0x2
Offset[130..131] = 0xxx Behavior (Chase, Stay, Rush, etc) more on this in the ITA section)
Offset[132..135] = 0x00390032 —---------- **Scale Speed**—---------- more on this in the ITA section)
Offset[136..139] = 7499.00 (Sight Range) (float value) —---------- more on this in the ITA section)
ItemAngle_X = 0.295800
ItemAngle_Y = -57.295800 (these bytes determine the facing angle of enemies, or rotation of items, more on his below).
ItemAngle_Z = 90.295798
Offset[152..155] = 0x0

Working Example - AEV entry

The next part is creating an AEV entry that will have the sister bytes that will link to the previous example entry of the ITA file. Let's have a look at an example of an AEV entry that we can use to trigger our enemy:

Event 34
Enemy ENEMY Spawn EA15

ID = 0x1
TriggerZoneHeightBoundary1 = 133.498444
TriggerZoneHeightBoundary2 = 719.622314
UnknownValue = 750.000000
TriggerZoneCorner0_X = -45753.839844
TriggerZoneCorner0_Y = 9086.477539
TriggerZoneCorner1_X = -43502.746094
TriggerZoneCorner1_Y = 8998.590820
TriggerZoneCorner2_X = -43522.011719
TriggerZoneCorner2_Y = 7060.516602
TriggerZoneCorner3_X = -45833.609375
TriggerZoneCorner3_Y = 6889.056152
Offset[52] = 0x3
Type = 0x6 (this Type 6 is not a vanilla Type but is effective for creating one-time AEV events that don't use prompts)
Users will need **THIS** edit for this to work
Index = 0x5
Offset[55] = 0x0
Offset[56] = 0x81 (These bytes in all 3 offsets combined make the event happen without a prompt, and only once).
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0

Offset[136..139] = 0x0
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0xEA15 (Sister byte that links this event to **any other entry** in the ITA that has the **Group ID** value of **Offset[88..91] = 0xEA15**

AEV Data Overview

We should understand that more than one entry in the ITA can use the same value of **Offset[88..91] = 0xEA15** which would allow us to spawn multiple enemies from one AEV TriggerZone. For each enemy we wish to spawn with the same AEV TriggerZone, we simply create another ITA entry (as shown above), with the same **Offset[88..91] = 0xEA15** which is the **Group ID**.

We can also make more than one AEV event like the one above, but use different **Group IDs.** This way we can have a variety of AEV TriggerZones around our rooms, triggering different **groups of enemies**. Valid values for **Group IDs** range from **01** to **FE**. (I haven't tested anything with zeros, like a value of **01, 02, 03** etc.. You could safely start at **10** and go up from there).. Values for this offset are in **HEX**.

Using with Companion DLL enemy_swap.ini

If users are familiar with the Companion DLL and the **enemy_swap.ini** feature, they may use it with this new AEV functionality. Instead of simply spawning an enemy with **Offset[76..79]** in the ITA as described above, users can simply spawn snakes as usual and replace their enemies using the **enemy_swap.ini** file. Doing things this way may add some variety if there are multiple AEV events triggering enemies), and allow users to edit additional parameters like health, outfits etc. Presently there is no tutorial on how to use this **enemy_swap.ini** feature, but I will eventually get around to writing one, and will add links to this tutorial in the future.

Spawning Items, Treasures & Weapons with AEV

Like the examples above, we can use an AEV event to spawn an item, treasure or weapon, or multiple items (I haven't tested spawning multiple items, but it should work in principle). The concept is almost exactly the same, except for the ITA entry for items, treasures, and weapons will not require us to use any bytes in the Offset[76..79] offset. Let's look at an example you can copy and paste for future use:

Working Example - ITA entry

Item Data 3 - ITA SPAWN

Category = 0x2
TriggerZoneHeightBoundary1 = 2076.125000
TriggerZoneHeightBoundary2 = 3162.792725
UnknownValue = 500.000000
TriggerZoneCorner0_X = 97841.507813
TriggerZoneCorner0_Y = -10652.327148
TriggerZoneCorner1_X = 101971.656250
TriggerZoneCorner1_Y = -10762.014648
TriggerZoneCorner2_X = 102002.976563
TriggerZoneCorner2_Y = -15618.452148
TriggerZoneCorner3_X = 97762.625000
TriggerZoneCorner3_Y = -15672.514648
Offset[52] = 0x2
DataBlockType = 0x3
DataBlockIndex = 0x3
Offset[55] = 0x1

Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..69] = 0x0
AppearanceType = 0x0
ETS_ID = 0x0
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
Offset[76..79] = 0x0 (notice that we do not need to add anything to this offset).
Offset[80..83] = 0x0
Offset[84..87] = 0x0
Offset[88..91] = 0xEA14 (This is the sister byte that links the corresponding AEV entry).
ItemPosition_X = 100305.531250
ItemPosition_Y = -13256.824219
ItemPosition_Z = 3635.744141
ItemNumber = 0x18
Randomness = 0x0
Offset[122..123] = 0x0
Amount = 0x12
Offset[126..127] = 0x5
AuraType = 0x0
Offset[130..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 1200.000000
ItemAngle_X = 0.000000
ItemAngle_Y = -0.000000
ItemAngle_Z = 0.000000
Offset[152..155] = 0x0

ITA Data Overview

Notice in this ITA entry that the **Group ID** is different from that of the ITA entry that is used for the enemy in the first example. In this example we are spawning a shotgun when we enter the TriggerZone of the linked AEV event. To make this work, we would simply need to create another AEV event (like the first example), with the corresponding **Group ID**. I have not tested this idea, but we should also be able to spawn combinations of enemies, weapons, treasures and items using the same **Group ID**. Feel free to test this out for yourselves.

About Item and Enemy placement

For those using 3dsMax, we can use the sub-object triangle models that are generated as part of the OBJ file when extracting the ITA file to help place items, weapons and treasures in locations. For users that do not have 3dsMax, you can simply delete the extracted OBJ file and use the coordinates in these offsets to place your enemy, item, treasure or weapons:

ItemPosition_X = 100305.531250
ItemPosition_Y = -13256.824219
ItemPosition_Z = 3635.744141

As you can see there are many options available to us now, especially when used in conjunction with some of the other parameters outlined in other posts in this guide. Imagine using an AEV event to spawn weapons, but that the AEV entry is only activated once an enemy is killed. This essentially means we can make enemy drops that have multiple rewards. You could also make it so that when one enemy is defeated, another is spawned immediately, or a whole group is spawned immediately. Like I said, there are now many more possibilities to create more dynamic interactions with the game play.

HEX CODE TO ACTIVATE FUNCTIONALITY

74 1D 80 F9 0B 75 0E 56 8B F0 80 FA 01
Change To:
E9 B0 1B 41 00 75 0E 56 8B F0 80 FA 01

Find: 006CA000
Paste:
66 9C 80 B8 99 00 00 00 EA 75 49 53 51 52 56 66 8B 98 98 00 00 00 8B 0D 00 0E 2E 10 8B 89 40 CE 00 00 31 D2 8A 51 06 31 F6 83 C1 10 46 66 39 59 58 75 13 C6 41 34 03 50 51 52 51 E8 60 F8 BE FF 83 C4 04 5A 59 58 81 C1 9C 00 00 00 39 D6 75 DC 5E 5A 59 5B 66 9D 0F 84 0E E4 BE FF 80 F9 0B E9 EC E3 BE FF

AEV EVENT MODIFICATIONS

Part 9a: AEV-ITA Enemy Spawn Persistence & ETS ID's

Part of this edit varies depending on if "EnableModExpansion" is set in RE4Tweaks - see note below

Previously when spawning enemies with the method shown above (via AEV-ITA), ETS_IDs were not assigned to enemies, thus making them unable to be linked for other AEV event toggles (using the **Offset[68..71]** offsets). With this new edit we accomplish several things:

1 - Enemies spawned via AEV-ITA can now be assigned an **ETS_ID**, so we are able to use their deaths as toggles.
2 - Enemies spawned via AEV-ITA can now be assigned an **ETS_ID,** so we are able to assign ammo/item/weapon/treasure drops from them.
3 - We did not know this until recently, but when an enemy was spawned via AEV-ITA, and we re-entered the room WITHOUT killing that enemy, the AEV event would not spawn the enemy again. If we simply use **Offset[126..127] = 0x0** in the ITA entry for the spawned enemy, the enemy will persistently spawn after re-entering the room, EVEN if we kill them.

What we could essentially do here is create an infinite enemy spawn (something like Mercenaries), by activating enemy spawns from the death of enemies. This could be done with groups of enemies as well, so for example, every enemy that dies, another 3 enemies spawn!

How to use this edit

In this scenario we are simply going to use the same setup as the previous edit above, and just add the following data into the ITA file:

AppearanceType = 0x3 (this ensures that an ITA-spawned enemy will respawn when we re-enter the room)
ETS_ID = [0x01 - 0xFE] (range of ETS ID's available to use. Make sure not to use ID's already in use by other ITA-ETS entries)

HEX CODE TO ACTIVATE FUNCTIONALITY

C6 86 A0 03 00 00 FF DB 45 08 DE C9 D9 9E 9C 00
Change To:
E9 88 00 00 00 90 90 DB 45 08 DE C9 D9 9E 9C 00

0F 82 D1 FD FF FF 5F 5E 5B 8B E5 5D C3 CC CC CC CC CC
Change To:
0F 82 D1 FD FF FF 5F 5E 5B 8B E5 5D E8 A6 13 11 00 C3

NOTE FOR NEXT PART OF EDIT IF USING RE4Tweaks

If "EnableModExpansion = true", change byte highlighted in red to **1C**
If "EnableModExpansion = false", change to **0C**
RE4Tweaks is **NOT** required for this edit, but if it is used use assigned byte)

Find: 001B1EC0 updated 7-16-2023
Paste:
50 53 8B 45 **1C** 8B 1D 00 0E 2E 10 8B 9B 40 CE 00 00 50 53 E8 90 04 00 00 84 DB 5B 58 74 2F 80 78 46 03 75 22 0F B6 40 47 88 86 A0 03 00 00 F7 86 CC 03 00 00 20 00 00 00 75 13 81 8E CC 03 00 00 20 00 00 00 EB 07 C6 86 A0 03 00 00 FF 5B 58 E9 26 FF FF FF

(updated 3-25-2023)
Find 002C2D90
Paste:
60 A1 00 0E 2E 10 8B 98 40 CE 00 00 85 DB 74 7B 31 D2 0F B6 4B 06 8D 5B 10 39 D1 74 6E 80 7B 46 03 75 5F 50 51 52 8D 43 5C E8 02 18 FF FF 5A 59 84 C0 58 74 58 0F B6 73 47 51 52 0F B6 90 B3 4F 00 00 8B CE C1 E9 05 83 E6 1F 8D 14 D1 8B 94 90 2C 50 00 00 BF 00 00 00 80 8B CE D3 EF 21 FA 85 D2 5A 59 75 1D 0F B7 73 7D 66 C7 43 7D 00 00 50 51 52 53 E8 98 6A FF FF 83 C4 04 5A 59 58 66 89 73 7D 8D 9B 9C 00 00 00 42 EB 8E 61 C3 60 0F B6 43 4C 85 C0 74 0D 6A 00 6A 00 50 E8 1F 4A D4 FF 83 C4 0C 61 EB DC

AEV EVENT MODIFICATIONS

Part 9c - EM ITA UDAS Pre-Load

Removes lag when spawning an enemy for the first time via ITA. This issue was fixed for ESL type entries in RE4Tweaks, but this fixes the issue for ITA spawns. Also, if the ITA event that spawns the enemy has already triggered, it will no longer pre-Load.

HEX CODE TO ACTIVATE FUNCTIONALITY
6A 00 6A 00 6A 24 E8 99 BA D4 FF 83 C4 0C 56 E8
Change To:
E9 50 0C 00 00 90 E8 99 BA D4 FF 83 C4 0C 56 E8

Find 002BCA00
Paste:
60 8D 46 5C E8 B7 7B FF FF 84 C0 75 1E 80 7E 4C 00 74 06 0F B6 46 4C EB 05 B8 24 00 00 00 6A 00 6A 00 50 E8 27 AE D4 FF 83 C4 0C 61 E9 88 F3 FF FF

AEV EVENT MODIFICATIONS

Part 9d - Universal ETS ID Access (linking events from room to room)

With this edit we are now able to toggle AEV and ITA events from one room to another. When using the **Offset[68..71]** offsets as described in the **Toggling AEV Events** section of the AEV tutorial along with another offset for the linked room, we are now no longer limited to linking events from the same room.

As described in the tutorial at the previously link, we use

Offset[68..71] = 0x43010008
Offset[120..123] = 0x01130000 (more on this below)

43 is the **ETS ID**
01 is the **interaction type**
08 is the **priority** (more on this later)

We use:
interaction type 01 for enemies
interaction type 02 for breakable ETM items

To link events from other rooms we now use the AEV **Offset[120..123] = 0x01130000 w**here 0**113** is the Room iD. Let's look at a working example of two rooms working together. For this example we will have rooms **r113** and **r320** working in tandem:

Scenario Description:

We have a Type 1 doorway event in **r320** that we wish to toggle with something in r113. Let's look at the extracted files:

Extracted AEV from r320

Offset[68..71] = 0x43010008
This sets an **enemy ID** of **43** to be used to toggle the event. Normally this would use the Enemy ID linked to the ESL for the room **r320**, but we can assign it to another room using:

Offset[120..123] = 0x01130000 (used in the same AEV entry as the **Offset[68..71]** above).This sets the target room for the **Offset[68..71] = 0x43010008** data.

So this has two possible scenarios:
0x43 is the **ETS ID** used in the vanilla ESL that is loaded for r113 **or…**
0x43 is the **ETS ID** used in the custom ITA entry that spawns an enemy in r113. (see AEV Enemy Spawns in Part 9 above).

So what this does is when the enemy in r113 with the **ETS ID 43** is killed, it will toggle the linked AEV event in room r320 the next time it is loaded.
Alternatively we can use ETM objects as the linking thing instead of enemies. We would simply use **Offset[68..71] = 0xxx020008 (02 instead of 01) and use the ETS ID of the ETS entry for the ETM object.**

Examples of Use

There are countless ways in which this new function can benefit us to create more dynamic scenarios. One example that user Kteo mentioned was to spawn an enemy with a 3F flag (we can do this in the vanilla ESL or in an ITA generated enemy spawn). Spawning an enemy with the 3F flag basically spawns the enemy dead (for Ganados you will see them lying on the floor). We can use the ETS ID of this dead enemy to act as a toggle for other events using the method described above. So instead of having to kill an enemy in another room to toggle the event, the event toggles automatically since the enemy spawns with the dead flag. This can be a very useful way to 'flip' bytes in AEV chains, to help with continuity of events between rooms, and on and on.

Another example that comes to mind is the spawning of enemies when we re-enter a room. So it might look something like this:

Kill enemy 04 in Room A = Spawn enemy 56 in Room B
Kill enemy 04 in Room A = Spawn enemies 56, 57, 89, 9A in Room B (this would be done by adding the toggle bytes to the AEV enemy spawn entry and all enemies would be assigned to the same enemy group).

Examples of use for ITA spawned items

Not only can we use this function to toggle AEV events but we can also use this to toggle ITA Item spawns. That is to say, we can use events from Room X to spawn ITA items in Room Y. For this we do not edit the AEV file, but instead edit the ITA. Since the ITA and AEV are almost identical in structure, they have the same functions at the same offsets. Let's look at how we can edit our ITA entries to use the same principles as the AEV above to toggle an item spawn:

Offset[52] = 0x2 *set this to 0x2 to keep spawn disabled until the ETS ID is broken/killed

AppearanceType = **0x1** (same **interaction type** function described above)
ETS_ID = 0x67 **enemy or breakable ETM model ID**

Offset[122..123] = 0x**0113** Room number (this is where we set the room number of the target room.

HEX CODE TO ACTIVATE FUNCTIONALITY
0F B7 91 AC 4F 00 00 52 50 E8 25 12 D5 FF 83 C4 08
Change To
E9 C5 28 00 00 90 51 52 50 E8 25 12 D5 FF 83 C4 0C

8A 80 B3 4F 00 00 0F B6 4F 47 84 C0 0F 88
Change To
E9 BD 27 00 00 90 0F B6 4F 47 84 C0 0F 88

Find: 002BF570

Paste:
0F B7 57 7A 85 D2 75 0C 0F B7 91 AC 4F 00 00 E9 28 D7 FF FF 51 52 50 E8 4D E9 D4 FF 85 C0 0F 85 20 D7 FF FF 58 5A 59 EB DF 50 0F B7 47 7A 85 C0 74 19 51 52 50 E8 D3 36 D4 FF 83 C4 04 5A 59 85 C0 78 08 83 C4 04 E9 22 D8 FF FF 58 8A 80 B3 4F 00 00 EB F2

AEV EVENT MODIFICATIONS

Part 10: Type 6 AEV Toggle

This edit allows Type 6 AEV's to activate automatically when stepped on (without a prompt). Basically this is used as a trigger in a chain or as a trigger for ITA Enemy Spawns.
Note: There are two edits that require this edit to function:
USAGE:

Offset[55] = 0x1
Offset[56] = 0x81
Offset[57] = 0x1

HEX CODE TO ACTIVATE FUNCTIONALITY
8A 48 35 80 60 34 FE 80 F9 03 74 15 80 F9 0B 0F 85 97 FC
Change To
E9 0F F5 FF FF 90 90 80 F9 03 74 15 80 F9 0B 0F 85 97 FC

Find: 002B9858
Paste:
8A 48 35 52 31 D2 8A 50 36 6A 00 52 E8 F3 82 D4 FF 83 C4 08 5A 80 B8 98 00 00 00 CF 75 06 FE 05 F0 0D 2E 10 E9 CA 0A 00 00

AEV EVENT MODIFICATIONS

Part 11: AEV Options

Allows for use of MDT generated choice menus to toggle AEV events.
- Supports 42 Options of which each can have an unlimited amount of AEVs to trigger.
- Multiple AEVs can now be put into a single option, for example:
{0x0700}{0xFE21}{0xFE3A}{0xFF41}

USAGE:

Users must create an AEV entry with the following parameters:

Event X
Message - (Choices)

ID = 0x1
TriggerZoneHeightBoundary1 = 426.320313
TriggerZoneHeightBoundary2 = 2668.497070
UnknownValue = 750.000000
TriggerZoneCorner0_X = 18517.935547
TriggerZoneCorner0_Y = -10223.799805
TriggerZoneCorner1_X = 19072.451172
TriggerZoneCorner1_Y = -10243.449219
TriggerZoneCorner2_X = 19065.455078
TriggerZoneCorner2_Y = -10696.993164
TriggerZoneCorner3_X = 18495.556641
TriggerZoneCorner3_Y = -10714.681641
Offset[52] = 0x3
Type = 0x5
Index = 0x2
Offset[55] = 0x1
Offset[56] = 0x8
Offset[57] = 0x1
Offset[68..71] = 0x2
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x14
Offset[75] = 0x0
# Message Event Parameters
Offset[92..93] = 0x0
RoomMessage = 0x1 (this is the entry in the MDT that will have the choices you set up)
MessageCutSceneID = 0x4 (use this for a camera if you want)
MessageID = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0

Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0x0
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Next, let's look at the code we will enter in the MDT to toggle the other AEV events in our AEV file:

In your MDT, for each option your entry must include:
{0x0700}{0xFFxx}TEXT_LABEL **or**
{0x0700}{0xFExx}TEXT_LABEL

{0x0700} = Option
xx = AEV index to be toggled - - **NOTE YOU CAN NOT USE INDEX 00**
FF = shuts off the AEV after the player selects the specified option.
FE = keeps the AEV on after the player selects the specified option.

You can mix and match, having certain options keep the AEV on and certain options turn it off, BUT it should be noted that any use of **FF** will prevent further use of the Type5 AEV event from operating again.

Example of MDT entry with choices:

{0x0000}{0x0400}{0x0700}{0xFE54}Operate door{0x0700}{0xFF55}Spawn Item A{0x0700}{0xFF56}Spawn Item B{0x0700}Exit{0x0800}{0x0100}

Definitions:

{0x0400} - Adds need for player input, This is placed in this entry to prevent a bug where pressing 'E' on keyboard auto selects the first option.
{0x0700} - Sets the first option
{0xFE54} - FE sets the AEV Type5 event to be allowed for future use. 54 is the AEV index that is toggled when this choice is selected.
Operate door - Option label (you can leave this blank if you like, there will still be an option)
…… {0xFF55} - Sets this option to prevent further use of the AEV Type 5 choice entry.

{0x0700} - the last entry has no index attached to it, so it will simply close out the MDT display when selected
{0x0800}{0x0100} - closes the MDT entry

For more information about related parameters please read:
Toggling AEV Events tutorial
MDT Tool Tutorial

HEX CODE TO ACTIVATE FUNCTIONALITY

E9 43 BA CE FF E9 BF 5B CF FF E9 CB AA CE FF E9 56 01 CF FF 83 C0 02 89
Change To
E9 31 FF FF FF E9 BF 5B CF FF E9 CB AA CE FF E9 56 01 CF FF 83 C0 02 89

8A 86 FA 00 00 00 FE C0 88 86 F9 00 00 00 C7 85 78 FF FF FF 00 00 00 00

Change To
E9 37 01 00 00 90 FE C0 88 86 F9 00 00 00 C7 85 78 FF FF FF 00 00 00 00

CC CC CC CC CC CC CC CC 55 8B EC 8B 45 08 50 E8 68 8D D4 FF
Change To
55 8B EC 83 EC 04 EB 0C 55 8B EC 8B 45 08 50 E8 68 8D D4 FF

Find: 00317EE0
Paste:

~~53 0F B7 58 02 80 FB FE 72 21 51 52 8D 89 30 4B 00 00 0F B6 91 C8 B5 FF FF 83 FA 54 7D 0B 66 83 3C 51 00 75 04 66 89 1C 51 5A 59 5B E9 E1 BA CE FF~~

~~53 0F B7 58 02 80 FB FE 72 1F 51 52 8D 89 30 4B 00 00 0F B6 91 C8 B5 FF FF 83 FA 2A 7D 09 83 3C 91 00 75 03 89 04 91 5A 59 5B E9 E3 BA CE FF CC CC~~

Updated 11-27-2023
53 0F B7 58 02 80 FB FE 72 2A 51 52 8D 89 30 4B 00 00 0F B6 91 C8 B5 FF FF 83 FA 2A 7D 14 83 3C 91 00 75 0E 89 04 91 83 39 00 75 06 C7 01 CC CC CC CC 5A 59 5B E9 D8 BA CE FF

Find: 00316A10
Paste:
~~8A 86 FA 00 00 00 66 83 BE 30 4B 00 00 00 74 76 51 52 0F B6 D0 8D 8E 30 4B 00 00 0F B7 14 51 80 FA FE 72 53 60 0F B6 D6 50 52 E8 45 A7 CE FF 83 C4 04 85 C0 74 15 F6 40 34 01 75 04 6A 01 EB 02 6A 00 52 E8 B0 19 FA FF 83 C4 08 58 61 80 FA FE 74 25 60 8B 81 34 3F 03 00 85 C0 74 19 0F B6 50 36 F6 40 34 01 75 04 6A 01 EB 02 6A 00 52 E8 85 19 FA FF 83 C4 08 61 50 6A 54 51 E8 E7 4E CF FF 83 C4 08 5A 59 58 E9 3F FE FF FF~~

~~8A 86 FA 00 00 00 66 83 BE 30 4B 00 00 7F 0F 86 B6 FE FF FF 51 52 0F B6 D0 8D 8E 30 4B 00 00 8B 14 91 85 D2 74 63 8D 52 02 52 0F B7 12 80 FA FE 72 56 60 0F B6 D6 50 52 E8 37 A7 CE FF 83 C4 04 85 C0 74 15 F6 40 34 01 75 04 6A 01 EB 02 6A 00 52 E8 A2 19 FA FF 83 C4 08 58 61 80 FA FE 74 25 60 8B 81 34 3F 03 00 85 C0 74 19 0F B6 50 36 F6 40 34 01 75 04 6A 01 EB 02 6A 00 52 E8 77 19 FA FF 83 C4 08 61 5A EB 9E 5A 50 6A 54 51 E8 D5 4E CF FF 83 C4 08 5A 59 58 E9 2D FE FF FF~~

Updated 11-27-2023

8A 86 FA 00 00 00 83 BE 30 4B 00 00 7F 0F 86 B7 FE FF FF 51 52 0F B6 D0 8D 8E 30 4B 00 00 8B 14 91 85 D2 74 6B 81 FA CC CC CC CC 74 63 8D 52 02 52 0F B7 12 80 FA FE 72 56 60 0F B6 D6 50 52 E8 30 A7 CE FF 83 C4 04 85 C0 74 15 F6 40 34 01 75 04 6A 01 EB 02 6A 00 52 E8 9B 19 FA FF 83 C4 08 58 61 80 FA FE 74 25 60 8B 81 34 3F 03 00 85 C0 74 19 0F B6 50 36 F6 40 34 01 75 04 6A 01 EB 02 6A 00 52 E8 70 19 FA FF 83 C4 08 61 5A EB 9E 5A 50 6A 54 51 E8 CE 4E CF FF 83 C4 08 58 5A 59 E9 26 FE FF FF

New Addition to Edit 1-04-2024

83 C0 80 0F B7 C0 66 85 C0
Change To

E9 CA 01 00 00 90 66 85 C0

Find: **00318698**

Paste:

3C FE 0F 83 65 FE FF FF 83 C0 80 0F B7 C0 E9 24 FE FF FF

New Addition to Edit 1-06-2024

66 0F BE 4C 10 01 66 0F BE 14 10 66 2B CA

Change To

E9 26 12 00 00 90 66 0F BE 14 10 66 2B CA

Find: 00318A88

Paste:

3C FC 72 0A 31 C0 66 31 C9 66 31 D2 EB 0B 66 0F BE 4C 10 01 66 0F BE 14 10 E9 C2 ED FF FF

AEV EVENT MODIFICATIONS

Part 12: AEV - Auto-Door Block

With this simple edit we can toggle whether Auto-Doors will actually move when we walk in their radius. By default, when the player walks within a certain distance of an auto-door ETM model, the door will move. This edit allows us to use an AEV event to turn them off and back on again (using the AEV Toggle functions). Usage is as follows:

Offset[128..131] = 0xDB2A
DB = DoorByte (activation)
2A = ETS ID

Example of use:

 # Event 22
 # Cancel Auto Door - (left side)

ID = 0x1
TriggerZoneHeightBoundary1 = 133.498444
TriggerZoneHeightBoundary2 = 719.622314
UnknownValue = 750.000000
TriggerZoneCorner0_X = -45753.839844
TriggerZoneCorner0_Y = 9086.477539
TriggerZoneCorner1_X = -43502.746094
TriggerZoneCorner1_Y = 8998.590820
TriggerZoneCorner2_X = -43522.011719
TriggerZoneCorner2_Y = 7060.516602
TriggerZoneCorner3_X = -45833.609375
TriggerZoneCorner3_Y = 6889.056152
Offset[52] = 0x2 (off by default, but when toggled, will prevent the auto-door ETM object with ETS iD 36 from functioning.
Type = 0x6
Index = 0xE4
Offset[55] = 0x1

Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0xDB36
Offset[132..135] = 0x0
Offset[136..139] = 0x0
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

HEX CODE TO ACTIVATE FUNCTIONALITY
Find: 002B7AC8
Paste:
E8 6B D1 11 00 E8 BE FE 10 00 C3

Find: **:** 003C7990
Paste
80 B8 81 00 00 00 DB 74 01 C3 60 0F B6 80 80 00 00 00 8B 1D 00 0E 2E 10 8D 9B 30 E2 FF FF 8D 0C 40 8D 0C 8B F6 01 01 74 33 8B 49 08 80 B9 54 06 00 00 01 75 09 C6 81 54 06 00 00 00 EB 1E 80 B9 48 06 00 00 01 75 0E C6 81 FE 00 00 00 00 C6 81 FD 00 00 00 02 C6 81 54 06 00 00 01 61 C3

AEV EVENT MODIFICATIONS

Part 13: AEV - Discard AEV

Allows you to take items away from the player and activate/deactivate AEV's accordingly

This is a VERY powerful edit that allows for the use of AEV (alongside the OSD file) to take items from the players inventory, and then activate AEV entries based on if the player has the item or not. For better ease of use, we decided to use the OSD file (which is hardly used by the game, if at all). We will need to hex edit this file to contain the parameters of our values. Let's look at the specifics:

OSD FILE

(users will need to hex edit the OSD file located in the extracted UDAS directory). Example:

44 69 73 63 26 02 23 00 01 00 04 00 10 00 01 21 22

44 69 73 63 "Disc" Header: **MUST ALWAYS BE INCLUDED** at the start of the file in order for the code to read it.

26 - AEV Index: AEV used to kick off the discard instructions. This would Typically be a Type6 that we stand on like this:

Event 39
Type 6 - Discard 1

ID = 0x1
TriggerZoneHeightBoundary1 = 4770.539551
TriggerZoneHeightBoundary2 = 4955.585938
UnknownValue = 750.000000
TriggerZoneCorner0_X = 26346.667969
TriggerZoneCorner0_Y = -1468.177246
TriggerZoneCorner1_X = 31712.880859
TriggerZoneCorner1_Y = -1466.146973
TriggerZoneCorner2_X = 31691.603516
TriggerZoneCorner2_Y = -8282.834961
TriggerZoneCorner3_X = 26228.367188
TriggerZoneCorner3_Y = -8313.178711
Offset[52] = 0x2
Type = 0x6
Index = 0x26
Offset[55] = 0x1
Offset[56] = 0x81
Offset[57] = 0x1
Offset[68..71] = 0x1
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0x0
Offset[140..143] = 0x0

Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

02 - # of Different Items. Consider this the entry count number which is basically the number of different items to check for and then discard. In the example above we have **02** entries:
23 00 01 00
04 00 10 00
If we wanted to have more entries it would look like this using **05** entries: **44 69 73 63 26 05 23 00 01 00 04 00 10 00 18 00 0E 00 20 00 32 00 A3 00 01 00 21 22**

23 00 - Item ID.
01 00 - Item Amount

So, observing the second example above, we see that the code is checking the player inventory for:

23 00 01 00 checks the player inventory for 1 (**01** hex) Handgun **(23 00).**
04 00 10 00 checks the player inventory for 16 (**10** hex) units of the Handgun ammo **(04 00).**
18 00 0E 00 checks the player inventory for 14 (**0E** hex) units of the Shotgun ammo **(18 00).**
20 00 32 00 checks the player inventory for 50 (**32** hex) units of the TMP ammo **(20 00).**
A3 00 01 00 checks the player inventory for 1 **01** hex) gallery key **(A3 00).**

If the value is "**FF FF**" it will discard ***everything*** for that item ID, and return 'success' even if you don't have any.
WARNING: Do not put "**00 00**", or it will crash if the player does not possess the item.
ALL OTHER VALUES will have the code specifically check if you have **at least** the amount specified, if not, the discard will not take place and will return 'failure' (Even if the rest of the entries satisfy).

05 - Amount of '**Success**' and '**Failure**' AEVs (see below)

21 - Index(s) of '**Success**' AEV(s). Here we place the AEV index (or indexes if more than one) for events that will get toggled (either ON or OFF) when the items are successfully discarded. (For example, a TypeB blocking AEV is toggled off and allows the player to proceed further after a key item is successfully discarded).

22 - Index(s) of '**Failure**' AEV(s).Here we place the AEV index (or indexes if more than one) for events that will get toggled (either ON or OFF) when one of the conditions does not satisfy and none of the items get discarded. (For example, this could be a Type5 message that alerts the player that they do not possess the item).

Using more than one '**Success**' and '**Failure**' AEV

There may be scenarios in which we wish to toggle more than one AEV event when we successfully (or unsuccessfully) interact with a Discard AEV. In the example below we have increased the a**mount of** '**Success**' and '**Failure**' AEVs to **05.** We see then that the 5 indexes that follow are the indexes of the '**Success**' AEV entries, then followed by another 5 indexes of the **Failure**' AEVs. For example:

44 69 73 63 26 02 23 00 01 00 04 00 10 00 05 21 22 23 24 26 33 34 35 36 37

We can also mix and match the number of working outcomes, but must remember that the amount must match. That is to say, if there are 5 **Success**' AEV entries, there must be 5 '**Failure**' entires. So if we want to have 5

working ‘**Success**’ AEV entries, but only 1 **Failure**’ entry, we can simply use ‘**FF’** in the remaining 4 slots. For example:

44 69 73 63 26 02 23 00 01 00 04 00 10 00 05 21 22 23 24 26 21 FF FF FF FF

The general rule is, for **Success**’ and **Failure**’ AEV indexes in the OSD file, we can use a value of "**FF"** if we don't want an AEV to be toggled for each one.

Some practical applications for this edit.

So we know now that with this edit we can use an AEV Type6 event to check the player for possession of certain items, and if those requirements are not met, then we can direct the OSD to toggle other AEV events. An interesting scenario that came to mind was to set the needed item to something unobtainable (or something that the player will never possess) like Item ‘**EF - maria**”. The Type6 event would check if the player had this item (which the player would not), so we could set the **Failure**’ entries in the OSD file to perform whatever we wish. Individual events in the AEV file could be toggled all at once, without having to chains which require that all related events are in linear order in the extracted .txt file.

EXTRA NOTE 1

- Discard AEV indexes can **NOT** be DLL Key AEVs or Hardcoded AEVs, as their functions will override the discard AEV's functions.

EXTRA NOTE 2

- There can be multiple discard AEVs to a room, here's what one looks like with three:

44 69 73 63 5E 01 D1 00 01 00 21 22 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 (padding for visual ease.. Not required)
44 69 73 63 5F 01 D2 00 01 00 21 22 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 (padding for visual ease.. Not required)
44 69 73 63 60 01 D3 00 01 00 21 22 00 00 00 00

Recommendation

Put at least 4 "CD"s in sequence at the end of the OSD file. This tells the code when to stop looking for Discard AEV entries. This is not required due to the fact that the file usually after the OSD file in the UDAS file, "BLK" has CD's at the end.

HEX CODE TO ACTIVATE FUNCTIONALITY

81 A0 CC 52 00 00 7F FF FF FF E8 48 50 D4
Change To
E9 89 2E 00 00 90 90 90 90 90 E8 48 50 D4

Find: 002C6D88
Paste:
81 A0 CC 52 00 00 7F FF FF FF 60 8B 98 34 4F 00 00 85 DB 74 39 81 3B CD CD CD CD 74 31 81 3B 44 69 73 63 74 03 43 EB ED 8D 5B 04 0F B6 0B 50 51 52 51 E8 C5 A3 D3 FF 83 C4 04 5A 59 8B C8 58 85 C9 74 E2 8D 90 40 27 D1 FF 89 51 40 EB D7 61 E9 28 D1 FF FF

Find: 00568A20
Paste:

~~60 8B 82 F4 27 2F 00 81 38 CD CD CD CD 0F 84 D8 00 00 00 81 38 44 69 73 63 74 03 40 EB E9 0F B6 4E 36 3A 48 04 75 F4 8D 40 05 0F B6 18 31 C9 0F B7 74 88 01 0F B7 7C 88 03 50 51 52 8D 8A 34 41 30 00 56 E8 9A 58 AA FF 5A 66 81 FF FF FF 74 09 66 39 F8 0F 82 89 00 00 00 59 58 41 39 D9 72 CF 31 C9 0F B7 74 88 01 0F B7 7C 88 03 50 51 52 8D 8A 34 41 30 00 6A 01 56 E8 4B 22 AA FF 5A 5A 66 81 FF FF FF 74 21 66 3B 78 02 7D 09 66 29 78 02 66 31 FF EB 1A 66 2B 78 02 50 51 52 50 E9 4B 00 00 00 5A 59 58 EB 08 85 C0 75 EE 59 58 EB 07 59 58 66 85 FF 75 B6 41 39 D9 7C A7 0F B6 74 98 01 56 E8 9E 86 A9 FF 5E 85 C0 74 20 F6 40 34 01 75 04 6A 01 EB 02 6A 00 56 E8 5F 90 A9 FF 83 C4 08 EB 09 59 58 0F B6 74 98 02 EB D5 61 C3 E8 3A CD A9 FF E8 67 96 A9 FF EB A9~~

New Addition to Edit 1-07-2024
60 8B 82 F4 27 2F 00 81 38 CD CD CD CD 0F 84 F3 00 00 00 81 38 44 69 73 63 74 03 40 EB E9 0F B6 4E 36 3A 48 04 75 F4 8D 40 05 0F B6 18 31 C9 0F B7 74 88 01 0F B7 7C 88 03 50 51 52 8D 8A 34 41 30 00 56 E8 9A 58 AA FF 5A 66 81 FF FF FF 74 09 66 39 F8 0F 82 9D 00 00 00 59 58 41 39 D9 72 CF 31 C9 0F B7 74 88 01 0F B7 7C 88 03 50 51 52 8D 8A 34 41 30 00 6A 01 56 E8 4B 22 AA FF 5A 5A 66 81 FF FF FF 74 21 66 3B 78 02 7D 09 66 29 78 02 66 31 FF EB 1A 66 2B 78 02 50 51 52 50 E9 66 00 00 00 5A 59 58 EB 08 85 C0 75 EE 59 58 EB 07 59 58 66 85 FF 75 B6 41 39 D9 7C A7 0F B6 7C 98 01 8D 44 98 02 85 FF 74 3E 0F B6 74 38 FF 50 53 57 56 E8 8E 86 A9 FF 5E 85 C0 74 15 F6 40 34 01 75 04 6A 01 EB 02 6A 00 56 E8 4F 90 A9 FF 83 C4 08 5F 5B 58 4F EB CE 59 58 0F B6 7C 98 01 8D 44 98 02 8D 04 38 EB BE 61 C3 E8 1F CD A9 FF E8 4C 96 A9 FF EB 8E

AEV EVENT MODIFICATIONS

Part 14: AEV - AUTO Message/CAM/Sound

This edit allows us now to trigger Type5 Messages automatically without a prompt. Normally any Type5 message AEV requires that the player interact with a prompt to activate the message. This can be a real pain because sometimes we want to access a CAM shot accompanied by a sound but we don't want to have to press a button to make it happen. For the past several years I have been using the Companion DLL to execute what I call 'AUTO-CAMS' and 'AUTO-Messages' that play instantly as soon as the player walks into a TriggerZone (or by other means, like having a CAMERA activate when another event is toggled). While this was indeed effective, there were certain limitations, specifically that there are only 10 entries allowed in each events.cfg file.

With this new edit we can tell the game to make only certain Type5 events play automatically, while the default will remain with prompts. We have used a special byte to act as a flag to tell the code when to be used.

The normal behavior for the game when loading an AEV is to check this offset:

Offset[56] = 0x81

In type 5 AEVs if the yellow bit is **less than 8**, then the game will force the byte to become 8 after the flag is read. This is the vanilla behavior.

Offset[56] = 0x81 (this important byte governs if an event is one-time only, or sometimes works as a flag to allow custom prompt messages for the event. In any case, it was recently discovered that the game code checks this byte for type5 AEVs, and if the second bit of the byte is lower than 8, it will set it to 8 in memory (after parsing any parameters that were activated by lower values, like custom prompts etc). So this behavior

explains why we can't simply use **Offset[56] = 0x5** for messages, since this value is below 8, it gets rewritten into memory.

USING THE CUSTOM FLAG FOR PROMPTLESS TYPE5 AEV EVENTS

Offset[56] = 0xFx

Example:
Offset[55] = 0x1
Offset[56] = 0xF1
Offset[57] = 0x1

So the way the code works is simple; If the **F** is present in a type 5 AEV, x bit will not be modified after being put into memory.

HEX CODE TO ACTIVATE FUNCTIONALITY
8A 46 38 A8 08 0F 85 64 FF FF FF 24 88 0C 08 88 46
Change To:
~~**8A 46 38 A8 08 0F 84 7B 7E 00 00 24 88 0C 08 88 46**~~
8A 46 38 A8 08 E9 7C 7E 00 00 90 24 88 0C 08 88 46 **(updated 1-10-2024)**

Find: 002C3B48
Paste:
~~**A8 F0 0F 85 E1 80 FF FF E9 78 81 FF FF**~~
A8 F0 0F 85 E1 80 FF FF A8 08 0F 85 D9 80 FF FF E9 70 81 FF FF **(updated 1-10-2024)**

AEV EVENT MODIFICATIONS

Part 15: AEV Checkpoint

IMPORTANT NOTICE ABOUT THIS EDIT - This edit has prerequisites so please be sure to see the updated version of the **Interdependent Edits** section).

This edit allows us to set checkpoints (without the use of a TypeE in previous edits). We can now disable checkpoints on AEV events (like doorways to other rooms) or add checkpoints to other events.

Usage

Offset[75] = **0xC0** - Checkpoint gets set when AEV is toggled.
Offset[75] = **0xC1** - Checkpoint gets removed from room teleport AEV.
Offset[75] = **0xC2** - Both 0xC1 and 0xC2.

In the case of using **0xC2** we can imagine this scenario; When the Type1 AEV gets toggled on/off it sets a checkpoint, but using the same Type1 to go to a different room does **NOT** set a checkpoint.

HEX CODE TO ACTIVATE FUNCTIONALITY
88 50 29 0F B6 4E 6D 8B 15 3C 5F
Change To:
E9 EB 0C 00 00 90 90 8B 15 3C 5F

Find: 002B7ED0
Paste:
88 50 29 0F B6 4E 6D 80 7E 4B C1 74 06 80 7E 4B C2 75 0A 81 88 28 50 00 00 00 00 00 80 E9 F5 F2 FF FF

Find: 002BE4B8
Paste:
80 78 4B C0 74 06 80 78 4B C2 75 07 60 E8 74 0A 12 00 61 C3

ITA EVENT MODIFICATIONS

ITA EVENT MODIFICATIONS

Part 1: Spawn Enemies from ITA (including ETM) models

(updated with new functionality 2-2023)
These parameters are also explained in **Part 9** section above

AuraType = 0xFE is needed to activate scaling and speed options
—————

This edit allows us to spawn any type of enemy from an ETM model when it is broken. Users should have a basic understanding of the ITA file as described HERE.

Just as in the case of the snake we will still need to use the following values in the extracted ITA file:
Randomness = 0x10
Offset[122..123] = 0x0

As described in earlier tutorials, the ETM object the enemy will spawn from is determined by the **ETS_ID** value in the ITA file.

As noted in the previous edit (Spawning Items, Treasures, & Weapons with AEV), If we wish to change the **EM_UDAS** in the ITA file (not to be confused with the **ESL ID**), we can simply add these custom values to the following offset in the extracted .txt file of the ITA:

Offset[76..79] = 0x01032B

01 - (Action) This is what is referred to as 'Enemy Behavior' in my **ESL TUTORIAL** and determines how the enemy will behave.
03 - (Sub-Type) This is the Sub-Type for enemies. Each of these use their own BIN file. Ganados, Zealots, and Commandos usually have 4 different Sub-Types that are used)
2B - (EM_UDAS) This is the corresponding **emxx.udas** file is loaded (but not always).**THIS IS NOT THE ESL ID**

(the example above spawns an El Gigante as **2B = EM_UDAS)** which is em2b.udas **THIS IS NOT THE ESL ID.**

To have the enemy persistently spawn (even after re-entering the room use):
AppearanceType = 0x3

ETS_ID = 0xXX

—----------

The following bytes are used for determining extra particulars like attire and weapons (for ganados, zealots and islanders)

Offset[80..83] = 0x1050007A

10 - (Weapon) Weapon enemy will equip. **Note:** Weapon will be disabled if **Equipment 1** has a value unless **Equipment1** value is a weapon!)
A full list of equipment descriptions and combinations can be found ~~HERE~~ (thank you Biohazard X)!

50 - (Equipment 2) Mostly head attire.
50 - (Equipment 1) Some head attire and weapons (like torches and knives).
~~**7A - (SightRange)** Distance until enemy reacts to player presence. This is the proximity of the player to the enemy.~~
The last byte we thought for years was responsible for SightRange but have since learned it does not, and has some other unknown function.

Offset[84..87] = (HealthPoints) Amount of health the enemy spawns with. **Example: (1000 = 0x3E8)** Max = **FFFF**

Offset[88..91] = 0xEAXX (Group ID)
These are the bytes that link the sister bytes in the corresponding AEV entry) This is called the **Group ID.** In the AEV we use **Offset[152..155] = 0xEAXX** to link the AEV event to the ITA entry (or entries). Whenever the AEV event with a **Group ID** at offset **Offset[152..155]** is activated, any ITA entry with the same **Group ID** at **Offset[88..91]** will be activated. To make this work the ITA entry (or entries) must also use **Offset[52] = 0x2** to keep the event dormant until activation.

Also, it should be noted that we can trigger multiple ITA entries for enemy spawns with **one AEV event.** For example, we can have an AEV event with:
Offset[152..155] = 0xEA34 that triggers multiple ITA entries who all use **Offset[88..91] = 0xEA34**.

Offset[130..131] = 0x0x Behavior (Chase, Stay, Rush, etc)
This is a two bit integer - Must use **0x0x NOT 0xx**
(example : Offset[130..131] = 0x02)
More can be read about this function of the ESL in my tutorial **HERE**

Offset[132..135] = 0x00390032 —---------- **Scale Speed**
Values are in HEX, so a value of **0064** will be 100% (normal)

Offset[136..139] = 7499.00 (Sight Range) (float value)
This is not to be confused with 'Reaction' in the Crzosk tool. We thought that was used for enemy sight range but we now know that it is not.
—----------

Special Note It should be noted that if we repack this ITA file with the Son of Persia tools, and then extract it again, the tools will remove the first **0** after the x, so it will look like this: **0x1032B**. You will notice that the first **0**

is missing, so make sure to put that back in your ITA .txt file if you are repacking the file after extracting. Same thing goes for **Offset[132..135]**

About Item and Enemy placement

For those using 3dsMax, we can use the sub-object triangle models that are generated as part of the OBJ file when extracting the ITA file to help place items, weapons and treasures in locations. For users that do not have 3dsMax, you can simply delete the extracted OBJ file and use the coordinates in these offsets to place your enemy, item, treasure or weapons:

ItemPosition_X = 100305.531250
ItemPosition_Y = -13256.824219
ItemPosition_Z = 3635.744141

One more valuable thing to know is the offset **Offset[132..135] = 0x1** . This will allow us to place the enemy anywhere in the room instead of spawning where the ETM item is placed. With this byte activated we can simply use the coordinates in the .txt file or the sub-object in the OBJ file in 3dsMax to position where we want the enemy to spawn.

Facing Angle of Enemy/ Item Rotation

This is a tricky thing to get right most of the time, as we have discovered that each of the XYZ angles affect each other in some way, and if there aren't the right values in one of the offsets, the rest won't work. I know that in the example below that these DO WORK, so simply copy and paste the values below into your extracted ITA .txt file, and make your adjustments to the **Z offset only**:

ItemAngle_X = 0.295800
ItemAngle_Y = -57.295800
ItemAngle_Z = 90.295798

About Enemy who throw weapons infinitely

For those wishing to use enemies that throw items infinitely, we would normally use **Offset[80..83] = 0xXX11XXXX,** but for some reason if any of the other 3 bytes are populated, it will ignore the game function and activate as normal. So if you ONLY want that 2nd byte to be something like 11, try **Offset[80..83] = 0x0011FF00. FF** might not be assigned to anything and will probably result in nothing for that equipment if thats want you want, but idk

Notes

I think the parasite doesn't spawn at all if the **Action** value is set to **00**. If you are having trouble with an enemy not spawning try other values for the **Action** byte.

About Respawning Enemies Infinetly and assigning items

For those wishing to spawn enemies infinitely, we can simply edit out ITA entry to include:

Offset[126..127] = 0x0 (or Offset[126..127] = 0x0000)

We can also now link enemy drops to these infinitely spawned enemies via the AuraType offset. Lets look at a working example:

Extracted ITA .txt FILE

Item Data 1 - Shotgun AMmo

Category = 0x2
TriggerZoneHeightBoundary1 = 2076.125000
TriggerZoneHeightBoundary2 = 3162.792725

UnknownValue = 500.000000
TriggerZoneCorner0_X = 97841.507813
TriggerZoneCorner0_Y = -10652.327148
TriggerZoneCorner1_X = 101971.656250
TriggerZoneCorner1_Y = -10762.014648
TriggerZoneCorner2_X = 102002.976563
TriggerZoneCorner2_Y = -15618.452148
TriggerZoneCorner3_X = 97762.625000
TriggerZoneCorner3_Y = -15672.514648
Offset[52] = 0x3
DataBlockType = 0x3
DataBlockIndex = 0x2 (we use this value in the **AuraType** of the enemy spawned via ITA)
Offset[55] = 0x1
Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..69] = 0x8
AppearanceType = 0x1 (we spawn the item/weapon/ammo via the normal method of enemy drops)
ETS_ID = 0x2B (we use the ETS_ID of the enemy in the entry with the enemy spawned via ITA)
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
Offset[76..79] = 0x0
Offset[80..83] = 0x0
Offset[84..87] = 0x0
Offset[88..91] = 0x0
ItemPosition_X = 100305.531250
ItemPosition_Y = -13256.824219
ItemPosition_Z = 3635.744141
ItemNumber = 0x18
Randomness = 0x0
Offset[122..123] = 0x0
Amount = 0x12
Offset[126..127] = 0x0000
AuraType = 0x1
Offset[130..131] = 0x100
Offset[132..135] = 0x0
Offset[136..139] = 1000.000000
ItemAngle_X = 0.000000
ItemAngle_Y = -0.000000
ItemAngle_Z = 0.000000
Offset[152..155] = 0x0……

(skipping entries is ok).....

Item Data 16 - Enemy Spawn EA24 - Creature E - Regen

Category = 0x2
TriggerZoneHeightBoundary1 = 68.316528
TriggerZoneHeightBoundary2 = 1000.000000
UnknownValue = 500.000000
TriggerZoneCorner0_X = -43901.687500
TriggerZoneCorner0_Y = -5248.697266

TriggerZoneCorner1_X = 0.000000
TriggerZoneCorner1_Y = -0.000000
TriggerZoneCorner2_X = 0.000000
TriggerZoneCorner2_Y = -0.000000
TriggerZoneCorner3_X = 0.000000
TriggerZoneCorner3_Y = -0.000000
Offset[52] = 0x2
DataBlockType = 0x3
DataBlockIndex = 0x22
Offset[55] = 0x1
Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..69] = 0x8
AppearanceType = 0x3
ETS_ID = 0x2B
Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
Offset[76..79] = 0x000736
Offset[80..83] = 0x0
Offset[84..87] = 0x100
Offset[88..91] = 0xEA24 (AEV Link)
ItemPosition_X = 143085.093750
ItemPosition_Y = 16506.539063
ItemPosition_Z = 3309.244141
ItemNumber = 0x0
Randomness = 0x10
Offset[122..123] = 0x0
Amount = 0x0
Offset[126..127] = 0x0
AuraType = 0x02FE (The FE is used to activate other functionality such as weapons, outfits, and size & speed. This is optional. Users can simply use **AuraType = 0x02** if needed).
Offset[130..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 8200.000000
ItemAngle_X = 0.000000
ItemAngle_Y = -90.000000
ItemAngle_Z = 180.000000
Offset[152..155] = 0x0

HEX CODE TO ACTIVATE FUNCTIONALITY

66 C7 45 DD 24 00 DD 05 28 D8 B0 00 C6 45 DF 00 DC F9 C7 45 E0 01 00
Change To:
E9 1E FF FF FF 90 DD 05 28 D8 B0 00 90 90 90 90 DC F9 C7 45 E0 01 00

B9 01 00 00 00 52 66 89 45 E4 66 89 4D F6 E8 E6
Change To:
E9 8B 02 00 00 52 66 89 45 E4 66 89 4D F6 E8 E6

Find: 002B5420
Paste:

66 81 7A F0 00 00 75 0F 66 C7 45 DD 24 00 C6 45 DF 00 E9 CC 00 00 00 50 66 8B 42 F0 66 89 45 DD 31 C0 8A 42 F2 88 45 DF 58 EB E7

Find: 002B5848

Paste:

B9 01 00 00 00 83 7D CC 00 74 1C 52 56 8B 75 CC 8B 56 50 83 FA 00 74 03 89 55 E0 8B 56 54 83 FA 00 74 02 8B C2 5E 5A E9 49 FD FF FF

F6 47 52 01 74 07 0F B6 4F 36 51 EB A5 53 33 DB

Change To:

F6 47 52 01 74 07 E9 E8 F7 FF FF EB A5 53 33 DB

Find: 002B9110

Paste:

66 83 7F 50 00 75 10 80 7F 53 00 75 0A 0F B6 4F 36 51 E9 A8 07 00 00 E9 FE 07 00 00

PART 2 BELOW DEPRECATED as of 7-08-2023

please revert this part of the edit in **red** and install the update below.

~~8A 86 01 01 00 00 C7 86 AC 02 00 00 78 2E~~

~~**Change To:**~~

~~E9 28 08 00 00 90 C7 86 AC 02 00 00 78 2E~~

Find: **00061228**

Paste:

~~8A 86 01 01 00 00 50 53 8B 85 18 01 00 00 8B 1D 00 0E 2E 10 8B 9B 40 CE 00 00 50 53 E8 1F 11 15 00 84 DB 5B 58 0F 84 E8 00 00 00 0F B7 98 80 00 00 00 66 81 FB FE 00 0F 85 CF 00 00 00 0F B6 98 82 00 00 00 85 DB 74 06 88 9E D8 03 00 00 8B 98 88 00 00 00 85 DB 74 06 89 9E D4 03 00 00 0F B7 98 84 00 00 00 85 DB 74 27 50 B8 64 00 00 00 66 0F 6E C8 F3 0F E6 C9 58 66 0F 6E C3 F3 0F E6 C0 F2 0F 5E C1 66 0F 5A C0 F3 0F 11 86 98 02 00 00 0F B7 98 86 00 00 00 85 DB 74 71 50 B8 64 00 00 00 66 0F 6E C8 F3 0F E6 C9 66 0F 6E C3 F3 0F E6 C0 F2 0F 5E C1 66 0F 5A C8 0F 28 C1 F3 0F 59 86 AC 00 00 00 F3 0F 11 86 AC 00 00 00 0F 28 C1 F3 0F 59 86 B0 00 00 00 F3 0F 59 86 B4 00 00 00 F3 0F 11 86 B0 00 00 00 F3 0F 11 8E B4 00 00 00 F3 0F 7E 86 AC 00 00 00 8B 86 B4 00 00 00 66 0F D6 86 98 04 00 00 89 86 A0 04 00 00 58 5B 58 E9 C6 F6 FF FF 8B 85 D8 00 00 00 50 53 E8 20 10 15 00 84 DB 5B 58 74 E6 E9 00 FF FF FF~~

----------Updated 7-08-2023----------

32 C0 E9 05 02 00 00 6A 00 57 56 6A

Change To:

90 90 E9 4F BC DA FF 6A 00 57 56 6A

Find: **00061228**

Paste:

~~CC CC 8B F0 30 C0 50 53 8B 85 9E 00 00 00 8B 1D 00 0E 2E 10 8B 9B 40 CE 00 00 50 53 E8 1F 11 15 00 84 DB 5B 58 0F 84 E8 00 00 00 0F B7 98 80 00 00 00 66 81 FB FE 00 0F 85 CF 00 00 00 0F B6 98 82 00 00 00 85 DB 74 06 88 9E D8 03 00 00 8B 98 88 00 00 00 85 DB 74 06 89 9E D4 03 00 00 0F B7 98 84 00 00 00 85 DB 74 27 50 B8 64 00 00 00 66 0F 6E C8 F3 0F E6 C9 58 66 0F 6E C3 F3 0F E6 C0 F2 0F 5E C1 66 0F 5A C0 F3 0F 11 86 98 02 00 00 0F B7 98 86 00 00 00 85 DB 74 71 50 B8 64 00 00 00 66 0F 6E C8 F3 0F E6 C9 66 0F 6E C3 F3 0F E6 C0 F2 0F 5E C1 66 0F 5A C8 0F 28 C1 F3 0F 59 86 AC 00 00 00 F3 0F 11 86 AC 00 00 00 0F 28 C1 F3 0F 59 86 B0 00 00 00 F3 0F 59 86 B4 00 00 00 F3 0F 11 86 B0 00 00 00 F3 0F 11 8E B4 00 00 00 F3 0F 7E 86 AC 00 00 00 8B 86 B4 00 00 00 66 0F D6 86 98 04 00 00 89 86 A0 04 00 00 58 5B 58 E9 A5 44 25 00 8B 85 54 00 00 00 50 53 E8 20 10 15 00 84 DB 5B 58 74 E6 E9 00 FF FF FF~~

----------Updated 10-2023----------
Find: 00061228
Paste:

~~CC CC 8B F0 30 C0 50 53 8B 85 9E 00 00 00 8B 1D 00 0E 2E 10 8B 9B 40 CE 00 00 50 53 E8 1F 11 15 00 84 DB 5B 58 0F 84 F0 00 00 00 85 F6 0F 84 E1 00 00 00 0F B7 98 80 00 00 00 66 81 FB FE 00 0F 85 CF 00 00 00 0F B6 98 82 00 00 00 85 DB 74 06 88 9E D8 03 00 00 8B 98 88 00 00 00 85 DB 74 06 89 9E D4 03 00 00 0F B7 98 84 00 00 00 85 DB 74 27 50 B8 64 00 00 00 66 0F 6E C8 F3 0F E6 C9 58 66 0F 6E C3 F3 0F E6 C0 F2 0F 5E C1 66 0F 5A C0 F3 0F 11 86 98 02 00 00 0F B7 98 86 00 00 00 85 DB 74 71 50 B8 64 00 00 00 66 0F 6E C8 F3 0F E6 C9 66 0F 6E C3 F3 0F E6 C0 F2 0F 5E C1 66 0F 5A C8 0F 28 C1 F3 0F 59 86 AC 00 00 00 F3 0F 11 86 AC 00 00 00 0F 28 C1 F3 0F 59 86 B0 00 00 00 F3 0F 59 86 B4 00 00 00 F3 0F 11 86 B0 00 00 00 F3 0F 11 8E B4 00 00 00 F3 0F 7E 86 AC 00 00 00 8B 86 B4 00 00 00 66 0F D6 86 98 04 00 00 89 86 A0 04 00 00 58 5B 58 E9 9D 44 25 00 8B 85 54 00 00 00 50 53 E8 18 10 15 00 84 DB 5B 58 74 E6 E9 F8 FE FF FF~~

----------Updated 11-2023----------

~~CC CC 8B F0 30 C0 50 53 8B 85 9E 00 00 00 8B 1D 00 0E 2E 10 8B 9B 40 CE 00 00 50 53 E8 1F 11 15 00 84 DB 5B 58 0F 84 05 01 00 00 85 F6 0F 84 F6 00 00 00 0F B7 98 80 00 00 00 66 81 FB FE 00 0F 85 E4 00 00 00 0F B6 98 82 00 00 00 85 DB 74 06 88 9E D8 03 00 00 8B 98 88 00 00 00 85 DB 74 06 89 9E D4 03 00 00 0F B7 98 84 00 00 00 85 DB 74 27 50 B8 64 00 00 00 66 0F 6E C8 F3 0F E6 C9 58 66 0F 6E C3 F3 0F E6 C0 F2 0F 5E C1 66 0F 5A C0 F3 0F 11 86 98 02 00 00 0F B7 98 86 00 00 00 85 DB 0F 84 82 00 00 00 50 B8 64 00 00 00 66 0F 6E C8 F3 0F E6 C9 66 0F 6E C3 F3 0F E6 C0 F2 0F 5E C1 66 0F 5A C8 0F 28 C1 F3 0F 59 86 AC 00 00 00 F3 0F 11 86 AC 00 00 00 0F 28 C1 F3 0F 59 86 B0 00 00 00 F3 0F 59 86 B4 00 00 00 F3 0F 11 86 B0 00 00 00 F3 0F 11 8E B4 00 00 00 F3 0F 7E 86 AC 00 00 00 8B 86 B4 00 00 00 66 0F D6 86 98 04 00 00 89 86 A0 04 00 00 8B 06 85 C0 74 0B 8B CE 8B 40 0C 85 C0 74 02 FF D0 58 5B 58 E9 88 44 25 00 8B 85 54 00 00 00 50 53 E8 03 10 15 00 84 DB 5B 58 74 E6 E9 E3 FE FF FF~~

—------Updated 12-29-2023----------
CC CC 8B F0 30 C0 50 53 8B 85 9E 00 00 00 8B 1D 00 0E 2E 10 8B 9B 40 CE 00 00 50 53 E8 1F 11 15 00 84 DB 5B 58 0F 84 32 01 00 00 85 F6 0F 84 23 01 00 00 0F B7 98 80 00 00 00 80 FB FE 53 90 0F 85 E4 00 00 00 0F B6 98 82 00 00 00 85 DB 74 06 88 9E D8 03 00 00 8B 98 88 00 00 00 85 DB 74 06 89 9E D4 03 00 00 0F B7 98 84 00 00 00 85 DB 74 27 50 B8 64 00 00 00 66 0F 6E C8 F3 0F E6 C9 58 66 0F 6E C3 F3 0F E6 C0 F2 0F 5E C1 66 0F 5A C0 F3 0F 11 86 98 02 00 00 0F B7 98 86 00 00 00 85 DB 0F 84 82 00 00 00 50 B8 64 00 00 00 66 0F 6E C8 F3 0F E6 C9 66 0F 6E C3 F3 0F E6 C0 F2 0F 5E C1 66 0F 5A C8 0F 28 C1 F3 0F 59 86 AC 00 00 00 F3 0F 11 86 AC 00 00 00 0F 28 C1 F3 0F 59 86 B0 00 00 00 F3 0F 59 86 B4 00 00 00 F3 0F 11 86 B0 00 00 00 F3 0F 11 8E B4 00 00 00 F3 0F 7E 86 AC 00 00 00 8B 86 B4 00 00 00 66 0F D6 86 98 04 00 00 89 86 A0 04 00 00 8B 06 85 C0 74 0B 8B CE 8B 40 0C 85 C0 74 02 FF D0 58 5B 84 FF 74 28 0F B6 DF 81 C3 80 00 00 00 53 E8 1F FE F9 FF 83 C4 04 85 C0 74 12 0F B6 58 46 53 50 56 E8 0F 05 FA FF 5E 58 5B 88 58 46 5B 58 E9 5B 44 25 00 8B 45 54 50 53 E8 D9 0F 15 00 84 DB 5B 58 74 E9 E9 B9 FE FF FF

Parasite Plaga ITA Respawn Fix

Fixes the issue of Spiders persisting when they shouldn't with the ITA edits

----
66 89 86 24 03 00 00 6A 00 56 E8 30 84 F6
Change To:
EB 3E 90 90 90 90 90 6A 00 56 E8 30 84 F6

Find: 0009DC18

Paste:
66 89 86 24 03 00 00 50 8A 86 A0 03 00 00 3C FF 58 74 07 56 E8 AF D8 F6 FF 5E EB AB

FSE EVENT MODIFICATIONS

FSE EVENT MODIFICATIONS

Part 1: FSE - AEV link

This allows for toggling the activation of an FSE event with the toggle of an AEV entry.

Note about Update
In the previous versions of the BIO4 Modifications Guides there was an edit that linked ETS events to FSE, offering us the ability to toggle FSE events when ETM models were broken. This edit is now deprecated and replaced with a newer version that links the AEV to the FSE, which gives us some more flexibility and functionality, while preserving the ability to use ETM model destruction as a source of toggling the FSE.
Please remove all previous FSE edits before installing this one in your bio4.exe

Prerequisites & Dependencies

Users should have a basic understanding of the FSE file. A full tutorial can be viewed or downloaded HERE.
This edit requires the Companion DLL
Also, as mentioned above, the AEV-Chain edit requires this edit to function.

Working with the FSE & AEV files

Working these scenarios is very simple. We only needs to do two things:

- make sure that the TriggerZones of both the AEV and the FSE overlap (unless directed otherwise by the example)
- in the FSE we put the index of the AEV trigger event at **Offset[124..127]**

Let's look at a working example of these two files working together. This scenario does the following (in order):

- Player enters room and **Song1** is played by FSE Event 0
- Player walks through AEV Event 2 - This plays a vanilla cutscene and starts chains forward (FB):
  Chain toggles **Song1** started by FSE Event 0 **OFF**, while also toggling **ON Song 2.**
- When **enemy is killed**, AEV Event 4 chains backwards (FA) and toggles AEV Event 3

Part 1 - AEV EVENTS

This is a two way chain that:
- starts music when an event is started,
- ends when the enemy is killed **(Offset[68..71] = 0x28010008)**

Event 2 - (vanilla event)
Cut Scene (starts chain forward)
(you could make this a Type 6 if this type2 doesn't work)

ID = 0x1
TriggerZoneHeightBoundary1 = 2251.394043
TriggerZoneHeightBoundary2 = 1000.000000
UnknownValue = 750.000000
TriggerZoneCorner0_X = 122561.710938
TriggerZoneCorner0_Y = -8151.380859
TriggerZoneCorner1_X = 125461.492188

TriggerZoneCorner1_Y = -8175.436523
TriggerZoneCorner2_X = 125391.023438
TriggerZoneCorner2_Y = -13774.835938
TriggerZoneCorner3_X = 122496.398438
TriggerZoneCorner3_Y = -13450.831055
Offset[52] = 0x3
Type = 0x2
Index = 0x2
Offset[55] = 0x0
Offset[56] = 0x81
Offset[57] = 0x1
Offset[68..71] = 0x2
Offset[72] = 0x0
Offset[73] = 0x2D
PromptMessage = 0x0
Offset[75] = 0x0
# Cut Scene Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0xFB
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Event 3
Type 2 - FSE LINK - (kills song1 that playing in the room when you first enter, then starts song2)
 # THIS EVENT TRIGGERZONE IS OUT OF BOUNDS

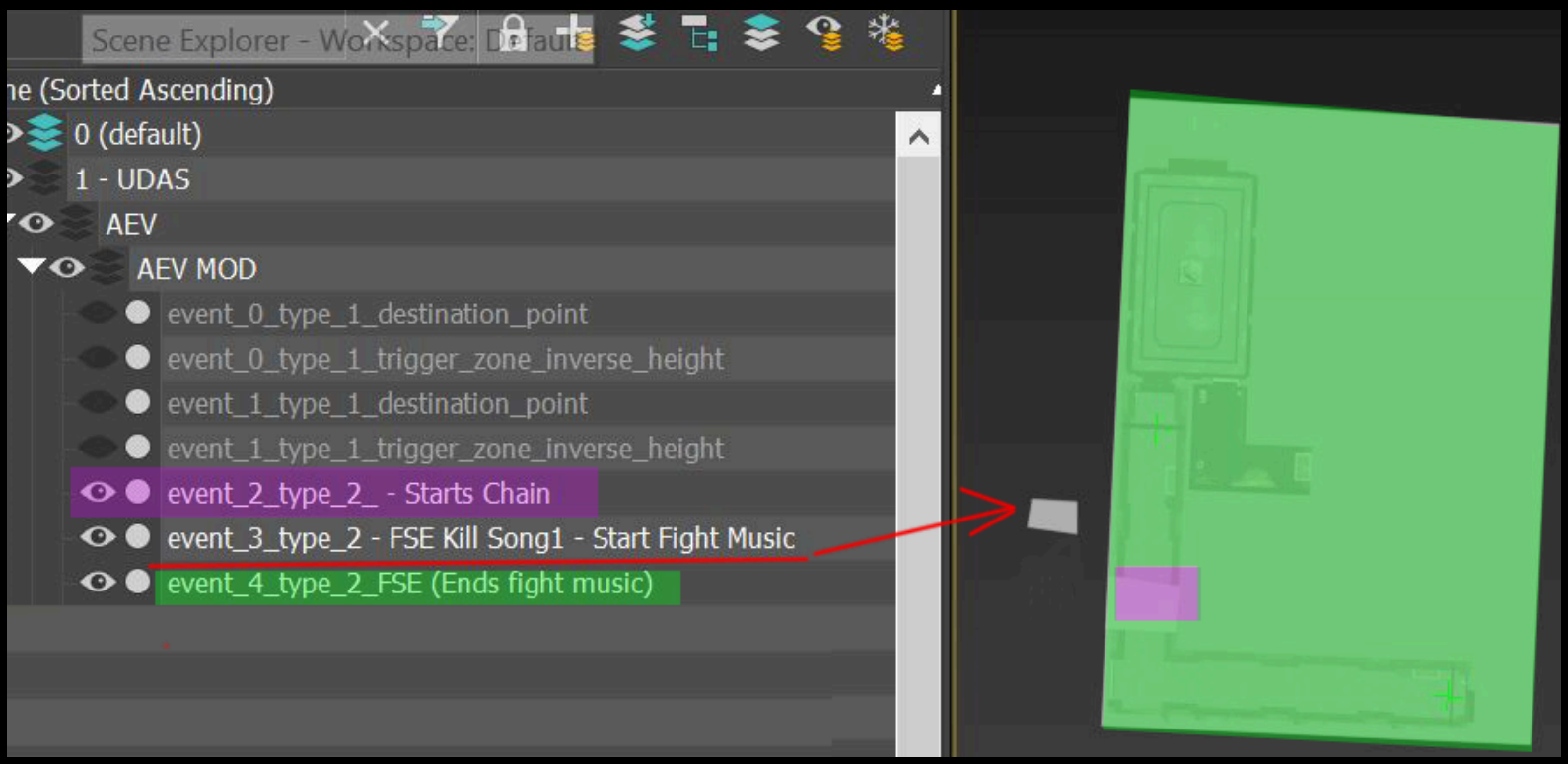
ID = 0x1
TriggerZoneHeightBoundary1 = 1082.875000
TriggerZoneHeightBoundary2 = 2169.542725
UnknownValue = 750.000000
TriggerZoneCorner0_X = 106117.984375
TriggerZoneCorner0_Y = -9084.941406
TriggerZoneCorner1_X = 110010.875000
TriggerZoneCorner1_Y = -9163.832031
TriggerZoneCorner2_X = 109901.187500
TriggerZoneCorner2_Y = -18232.496094
TriggerZoneCorner3_X = 106172.046875
TriggerZoneCorner3_Y = -18263.816406
Offset[52] = 0x3
Type = 0x2

Index = 0x1D (**0x1D** is the linking byte in **Offset[124..127]** in the FSE .txt file)
Offset[55] = 0x1
Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
PromptMessage = 0x0
Offset[75] = 0x0
# Cut Scene Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0x0
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Event 4
Type 2 - FSE Toggle 2 (Ends fight music)
Starts the chain backwards to toggle the previous event
 # THIS EVENT COVERS THE ENTIRE AREA OF THE FIGHT

ID = 0x1
TriggerZoneHeightBoundary1 = 133.498444
TriggerZoneHeightBoundary2 = 719.622314
UnknownValue = 750.000000
TriggerZoneCorner0_X = -45753.839844
TriggerZoneCorner0_Y = 9086.477539
TriggerZoneCorner1_X = -43502.746094
TriggerZoneCorner1_Y = 8998.590820
TriggerZoneCorner2_X = -43522.011719
TriggerZoneCorner2_Y = 7060.516602
TriggerZoneCorner3_X = -45833.609375
TriggerZoneCorner3_Y = 6889.056152
Offset[52] = 0x2
Type = 0x6
Index = 0x26
Offset[55] = 0x1
Offset[56] = 0x81
Offset[57] = 0x1
Offset[68..71] = 0x28010008 (ETS ID of enemy, that when killed, toggles this event)
Offset[72] = 0x0

Offset[73] = 0x2D
Offset[74] = 0x0
Offset[75] = 0x0
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0xFA
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0



PART 2 - FSE EVENTS

EntryCount = 2
Type = 1

Entry 0 - First song when you enter room

ID = 0x203

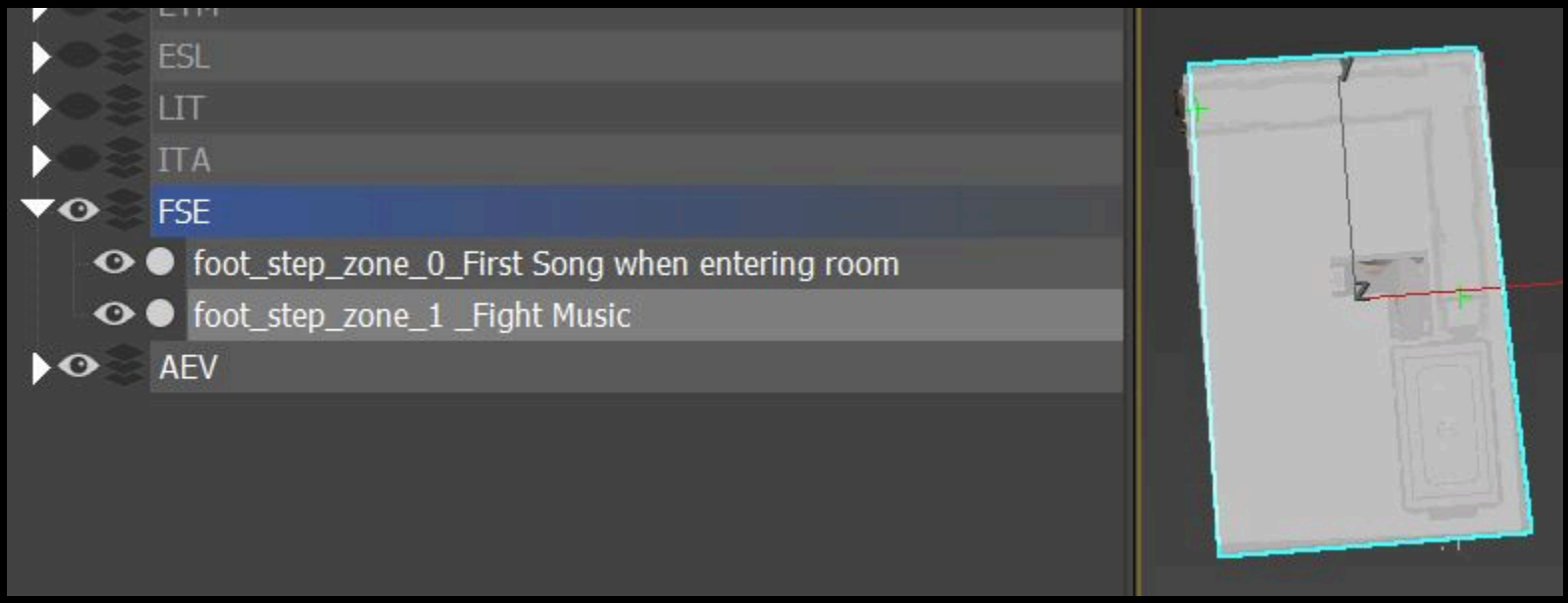
Index = 0x86
Offset[4..7] = 0x8
Offset[20] = 0x1
Offset[21] = 0x1
FootStepZoneHeightBoundary1 = -0.001831
FootStepZoneHeightBoundary2 = 1000.000000
UnknownValue = 500.000000
FootStepZoneCorner0_X = -13879.056641
FootStepZoneCorner0_Y = 17291.673828
FootStepZoneCorner1_X = -12603.895508
FootStepZoneCorner1_Y = 16900.998047
FootStepZoneCorner2_X = -13556.769531
FootStepZoneCorner2_Y = 14822.679688
FootStepZoneCorner3_X = -14687.317383
FootStepZoneCorner3_Y = 15348.590820
Offset[68] = 0x10
Offset[69] = 0x10
Offset[70] = 0x0
Offset[71] = 0x0
Offset[72..75] = 0x0
Offset[76..79] = 0x0
Offset[80..83] = 0x210000 (Song 1)
Offset[84..87] = 0x0
Offset[88..91] = 0x0
Offset[92..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x1D
Offset[128..131] = 0x0

Entry 1 Data - FIGHT music START

ID = 0x203
Index = 0x87
Offset[4..7] = 0x8
Offset[20] = 0x1
Offset[21] = 0x1
FootStepZoneHeightBoundary1 = 7253.291016
FootStepZoneHeightBoundary2 = 18067.361328
UnknownValue = 500.000000
FootStepZoneCorner0_X = -18588.453125
FootStepZoneCorner0_Y = -21045.888672
FootStepZoneCorner1_X = -12890.717773
FootStepZoneCorner1_Y = -21421.236328
FootStepZoneCorner2_X = -13079.687500
FootStepZoneCorner2_Y = -28411.601563
FootStepZoneCorner3_X = -18622.851563
FootStepZoneCorner3_Y = -28540.109375

Offset[68] = 0x10
Offset[69] = 0x0
Offset[70] = 0x0
Offset[71] = 0x0
Offset[72..75] = 0x0
Offset[76..79] = 0x0
Offset[80..83] = 0x4C0000 (song 2)
Offset[84..87] = 0x0
Offset[88..91] = 0x0
Offset[92..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x1D (**0x1D** is the linking byte to the AEV. We put the AEV index of the trigger event here).
Offset[128..131] = 0x0

(the TriggerZones for both FSE events cover the entire area of the battle with the enemy).



Removing Ambient Environmental Sounds with the FSE

Another handy feature of this edit is the ability to cancel all ambient environmental sounds using the FSE. To do this we simply use:

Offset[128..131] = 0xAF in **any** FSE entry. The player does not even need to walk through the TriggerZone of this event to make use of it. We can simply make a dummy event somewhere out of reach, or simply just add the **0xAF** byte to any entry to make it work.

FSE TriggerZones Can be Overlapped

And one more part of this edit that adds really useful functionality to FSE editing is we now have the ability to overlap FSE events that trigger music. Previously the TriggerZones for FSE entries that played music from the bio4bgm file would need to be isolated from other similar events. This part of the edit makes it so that at least two of them can overlap (perhaps

more than two, but I have not tested). This edit can be particularly useful if we are using the FSE to trigger secondary BGM sounds, while simultaneously triggering music from the bio4bgm.

HEX CODE TO ACTIVATE FUNCTIONALITY

FSE AEV Functionality

E8 68 8D D4 FF 83 C4 04 85 C0 75 11 68 C4 F5

Change To:

E8 68 8D D4 FF E9 F7 16 ED FF 75 11 68 C4 F5

---

83 C4 08 F6 00 01 0F 84 70 FE FF FF

Change To:

83 C4 08 F6 00 01 E9 B1 FB FF FF 90

Find: 002BC870

Paste:

0F 84 BA 02 00 00 50 53 51 56 8B 35 00 0E 2E 10 8B B6 68 ED FF FF 8A 5E 06 84 DB 74 30 31 C9 8A 47 36 84 C0 74 27 83 C6 10 41 38 46 7C 75 12 80 7E 45 10 74 06 C6 46 45 10 EB 12 C6 46 45 00 EB 0C 38 D9 74 08 81 C6 84 00 00 00 EB DC 5E 59 5B 58 E9 FA 03 00 00

Updated 7-16-2023 (this edit is also present in the ESE - AEV link edit as both edits share this block of data - Be sure to install these in the right order - See Interdependent Edits section that is pinned at the top of this document

Find: 00189B18

Paste:

83 C4 04 85 C0 0F 84 00 E9 12 00 E8 06 00 00 00 E9 F4 E8 12 00 CC 66 9C 50 53 51 56 8B 35 00 0E 2E 10 8B B6 68 ED FF FF 8A 5E 06 84 DB 74 30 31 C9 8A 40 36 84 C0 74 27 83 C6 10 41 38 46 7C 75 12 80 7E 45 10 74 06 C6 46 45 10 EB 06 C6 46 45 00 90 90 38 D9 74 08 81 C6 84 00 00 00 EB DC 5E 59 5B 58 66 9D C3

HEX CODE TO ACTIVATE FUNCTIONALITY

No Ambience on FSE byte Weather (also disables wind in r226)

Use Offset[128..131] = 0xAF in the FIRST ENTRY of your FSE to disable default whether sounds

---

0F B7 C8 52 51 E8 15 D3 A9 FF 83 C4 18

Change To:

0F B7 C8 52 51 E9 2D FF FF FF 83 C4 18

Find: 0056CDE8

Paste:

50 A1 00 0E 2E 10 81 B8 AC 4F 00 00 26 02 00 00 58 75 11 83 FA 01 77 05 E9 B6 00 00 00 E8 C6 D3 A9 FF EB F4 50 A1 00 0E 2E 10 8B 80 68 ED FF FF 83 F8 00 74 0C 80 B8 90 00 00 00 AF 58 74 D9 EB DC 58 EB D9

HEX CODE TO ACTIVATE FUNCTIONALITY

FSE TtriggerZones Can Be Overlapped

---

81 C7 84 00 00 00 3B D9 72 A8 5F 5E 33 C0 5B 8B

Change To:

EB 30 90 90 90 90 3B D9 72 A8 5F 5E 33 C0 5B 8B

C6 5E 33 CD 5B E8 80 27 43 00 8B E5 5D C3

Change To:

C6 5E 33 CD 5B EB 1B 90 90 90 8B E5 5D C3

Find: 00238570
Paste:
81 C7 84 00 00 00 80 3E FE 75 03 C6 06 03 EB C4 CC CC 80 38 FE 74 03 C6 00 FE E8 5B 27 43 00 EB D9 CC CC

ESE MODIFICATIONS

ESE EVENT MODIFICATIONS

Part 1: ESE - AEV Toggle

Very much like the previous edit in this guide, this edit enables us to use AEV to toggle sounds events, in particular, ESE events. Users should have a basic understanding of the principles of modding sound which can be read or downloaded HERE. Also a reminder that the SoP ESE tool works fine, except that the extracted .obj file does not function at all for repacking, so coordinates for ESE events must be inputted manually in the extracted ESE **.txt** file.

Prerequisites & Dependencies

Users should have a basic understanding of the ESE file. A full tutorial can be viewed or downloaded HERE.
This edit requires the Companion DLL
Also, as mentioned above, the AEV-Chain edit requires this edit to function.

Working with the ESE & AEV files

As with the FSE-AEE, working these scenarios is very simple. We only needs to do two things:

- create an AEV event that will act as a toggle for the ESE event.
- in the ESE we put the AEV index of the AEV trigger event at **Offset[40..43] = 0x32EE** (this **'EE'** must be added after the AEV index)

Let's look at two different examples of this at play. The first is simply the destruction of an ETM model that, in turn toggles an ESE event, and the second example will be a little more complex, with a lever being used to toggle an AEV event, which in turn will toggle the ESE event:

Example #1

AEV EVENT

Event 42
Kill ESE Trigger

ID = 0x1
TriggerZoneHeightBoundary1 = 133.498444
TriggerZoneHeightBoundary2 = 719.622314
UnknownValue = 750.000000
TriggerZoneCorner0_X = -45753.839844
TriggerZoneCorner0_Y = 9086.477539
TriggerZoneCorner1_X = -43502.746094
TriggerZoneCorner1_Y = 8998.590820
TriggerZoneCorner2_X = -43522.011719
TriggerZoneCorner2_Y = 7060.516602

TriggerZoneCorner3_X = -45833.609375
TriggerZoneCorner3_Y = 6889.056152
Offset[52] = 0x2 (event is disabled by default, but will be toggled by Offset[68..71]
Type = 0x6
Index = 0x6A (this value must be placed at **Offset[40..43]** in the ESE… see below)
Offset[55] = 0x0
Offset[56] = 0x81
Offset[57] = 0x1
Offset[68..71] = 0x**22**020008 (AEV event is activated by the destruction of ETM model linked to **ETS_ID 22**) see **Toggling AEV Events** section of the AEV Tutorial for more info).
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x11
Offset[75] = 0x0
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0x0
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

ESE EVENT

EntryCount = 1
Type = 0

Effect Sound Entry 0 Data - Electricity

ID = 0x3
Index = 0x1
Position_X = -15147.885
Position_Y = -28253.703
Position_Z = 9327.895
Offset[16..19] = 0x6
Type = 0x6
Offset[24..27] = 0x0
Offset[28..31] = 0x10000
Offset[32..33] = 0x0
Offset[34..35] = 0x0
Offset[36..39] = 0x0
Offset[40..43] = 0x6AEE

So in the above example, we have a simple setup that has an AEV trigger that cancels the ESE event, but the AEV itself is disabled by default, leaving the ESE to play normally until the AEV is toggled on by the destruction of the ETM model linked to **ETS_ID 22.**

Example #2

AEV EVENTS

Event 36
Kill ESE Trigger (Auto Destruct)

ID = 0x1
TriggerZoneHeightBoundary1 = 133.498444
TriggerZoneHeightBoundary2 = 719.622314
UnknownValue = 750.000000
TriggerZoneCorner0_X = -45753.839844
TriggerZoneCorner0_Y = 9086.477539
TriggerZoneCorner1_X = -43502.746094
TriggerZoneCorner1_Y = 8998.590820
TriggerZoneCorner2_X = -43522.011719
TriggerZoneCorner2_Y = 7060.516602
TriggerZoneCorner3_X = -45833.609375
TriggerZoneCorner3_Y = 6889.056152
Offset[52] = 0x3
Type = 0x6
Index = 0x31
Offset[55] = 0x1
Offset[56] = 0x89
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x10
Offset[75] = 0x0
# Map Blocking Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0xAD (auto destructs the event after use) See H ERE for bio4.exe edit.
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0xFB (sets this entry as the ‘leader’ in the AEV chain going in a forwards direction).). See **HERE** for bio4.exe edit.
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0

Offset[152..155] = 0x0

Event 37
Type 2 - Kill ESE Trigger 2

ID = 0x1
TriggerZoneHeightBoundary1 = 1000.000000
TriggerZoneHeightBoundary2 = 26327.994141
UnknownValue = 750.000000
TriggerZoneCorner0_X = -1116.245239
TriggerZoneCorner0_Y = 24868.894531
TriggerZoneCorner1_X = 10586.791016
TriggerZoneCorner1_Y = 46732.250000
TriggerZoneCorner2_X = 22776.101563
TriggerZoneCorner2_Y = 24093.476563
TriggerZoneCorner3_X = 5210.395020
TriggerZoneCorner3_Y = 17294.962891
Offset[52] = 0x3 (this event is Active by default, and turns off the ESE event as soon as we enter the room, making the sound be disabled by default).
Type = 0x2
Index = 0x32 (this index byte is also present in the ESE entry below. This is the event that is linked to the ESE)
Offset[55] = 0x1
Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
PromptMessage = 0x0
Offset[75] = 0x0
# Cut Scene Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0x0
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Event 38
Message “elevator power”

ID = 0x1
TriggerZoneHeightBoundary1 = 6025.960938

TriggerZoneHeightBoundary2 = 9084.715820
UnknownValue = 750.000000
TriggerZoneCorner0_X = -12130.857422
TriggerZoneCorner0_Y = -30284.125000
TriggerZoneCorner1_X = -10935.885742
TriggerZoneCorner1_Y = -30252.406250
TriggerZoneCorner2_X = -10935.885742
TriggerZoneCorner2_Y = -32879.890625
TriggerZoneCorner3_X = -12130.857422
TriggerZoneCorner3_Y = -32911.609375
Offset[52] = 0x3
Type = 0x5
Index = 0x33
Offset[55] = 0x1
Offset[56] = 0x8
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
Offset[74] = 0x1
Offset[75] = 0x0
# Message Event Parameters
Offset[92..93] = 0x0
RoomMessage = 0x9
MessageCutSceneID = 0x0 - (chooses which **.MDT** message to display when this event is active)
MessageID = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0xFF (sets this entry as a ‘follower’ in the AEV chain, this time from the other direction - not linked to previous chain above)
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Event 39
Type 2 - Control Room Electricity EAR 1B link

ID = 0x1
TriggerZoneHeightBoundary1 = 1082.875000
TriggerZoneHeightBoundary2 = 2169.542725
UnknownValue = 750.000000
TriggerZoneCorner0_X = 106117.984375
TriggerZoneCorner0_Y = -9084.941406
TriggerZoneCorner1_X = 110010.875000
TriggerZoneCorner1_Y = -9163.832031

TriggerZoneCorner2_X = 109901.187500
TriggerZoneCorner2_Y = -18232.496094
TriggerZoneCorner3_X = 106172.046875
TriggerZoneCorner3_Y = -18263.816406
Offset[52] = 0x2
Type = 0x2
Index = 0x34
Offset[55] = 0x0
Offset[56] = 0x1
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
PromptMessage = 0x0
Offset[75] = 0x0
# Cut Scene Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0xFF (sets this entry as a 'follower' in the AEV chain)
Offset[140..143] = 0xEB1B (sets this entry to toggle a visual effect linked to an EAR entry) See **HERE** for bio4.exe edit.
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

Event 40
Type 2 - Electricity Chain Toggle

ID = 0x1
TriggerZoneHeightBoundary1 = 1082.875000
TriggerZoneHeightBoundary2 = 2169.542725
UnknownValue = 750.000000
TriggerZoneCorner0_X = 106117.984375
TriggerZoneCorner0_Y = -9084.941406
TriggerZoneCorner1_X = 110010.875000
TriggerZoneCorner1_Y = -9163.832031
TriggerZoneCorner2_X = 109901.187500
TriggerZoneCorner2_Y = -18232.496094
TriggerZoneCorner3_X = 106172.046875
TriggerZoneCorner3_Y = -18263.816406
Offset[52] = 0x2
Type = 0x2
Index = 0x35
Offset[55] = 0x1

Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..71] = 0x2A020008 (links event to ETM lever in **ETS_ID 2A)**
Offset[72] = 0x0
Offset[73] = 0x2D
PromptMessage = 0x0
Offset[75] = 0x0
# Cut Scene Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0
Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0x0
Offset[132..135] = 0x0
Offset[136..139] = 0xFA (sets this entry as the 'leader' in the AEV chain, going in a backwards direction).
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

ESE EVENT

Effect Sound Entry 0 Data

ID = 0x3
Index = 0x1
Position_X = 1883.645
Position_Y = 20942.723
Position_Z = 27698.689
Offset[16..19] = 0x6
Type = 0x9
Offset[24..27] = 0x0
Offset[28..31] = 0x10000
Offset[32..33] = 0x0
Offset[34..35] = 0x0
Offset[36..39] = 0x0
Offset[40..43] = 0x32EE (**32** is the index of the event that toggles the sound. **'EE'** must be added after the AEV index).

Have a quick look at this video to see how this event plays out in the game (double click image to watch video)



So this chain of events is a little more complex, but works well for certain scenarios involving levers. In this scenario we are using a lever to start a chain of events that toggles an ESE event. In this setup we have the ESE disabled as soon as we enter the room so that the sound controlled by the ESE is OFF when we first enter the room. There are also other events in the chain that get toggled, like a message that blocks the elevator access, and the visual effect of the electricity.

HEX CODE TO ACTIVATE FUNCTIONALITY

AEV-ESE Toggle

75 11 68 C4 F5 B1 00 50 50 E8 49 3E D5 FF
Change To:
75 D5 68 C4 F5 B1 00 50 50 E8 49 3E D5 FF

Find: 002B83F8
Paste:
E8 CB F6 FF FF EB 35

Find: 002B7AC8
Paste:
E8 6B D1 11 00 C3

Find: **003D4C38** **(Updated with a fix 3-29-2023)**
Paste:
60 8A 40 36 8B 1D 00 0E 2E 10 8B 8B 84 48 5E 00 8D 9B D8 E9 B7 FF 0F B6 51 06 31 F6 8D 49 10 39 D6 74 5A 80 79 28 EE 75 4E 3A 41 29 75 49 50 52 51 F6 01 01 74 0F 80 21 FE 8B C3 0F B7 51 14 0F B7 49 10 EB 1E 80 09 01 66 81 79 1E FF FF 75 24 8D 41 04 F6 41 02 01 74 02 31 C0 0F B7 51 14 0F B7 49 10 6A 00 6A 00 6A 00 50 52 51 E8 27 55 C3 FF 83 C4 18 59 5A 58 46 8D 49 2C EB A2 61 C3 CC 00 00 80 7F 00 00 80 7F 00 00 80 7F

NEXT PART IS ALSO IN THE AEV-FSE EDIT (you can overwrite this or leave it if you already have that edit installed)

Updated 7-16-2023 (this edit is also present in the FSE - AEV link edit as both edits share this block of data)
Find: 00189B18
Paste:
83 C4 04 85 C0 0F 84 00 E9 12 00 E8 06 00 00 00 E9 F4 E8 12 00 CC 66 9C 50 53 51 56 8B 35 00 0E 2E 10 8B B6 68 ED FF FF 8A 5E 06 84 DB 74 30 31 C9 8A 40 36 84 C0 74 27 83 C6 10 41 38 46 7C 75 12 80 7E 45 10 74 06 C6 46 45 10 EB 06 C6 46 45 00 90 90 38 D9 74 08 81 C6 84 00 00 00 EB DC 5E 59 5B 58 66 9D C3

ESE EVENT MODIFICATIONS

Part 2: ESE's Paused Instead of removed on Companion DLL Movie

Fixes issue of ambient environmental sounds, as well as ESE triggered sounds from stopping after an .SFD movie is triggered via the Companion DLL.

HEX CODE TO ACTIVATE FUNCTIONALITY

E8 6B E5 D3 FF 5F 5B B0 01 5E 5D C2 08 00 E8 A5
Change To:
EB 1B 90 90 90 5F 5B B0 01 5E 5D C2 08 00 E8 A5

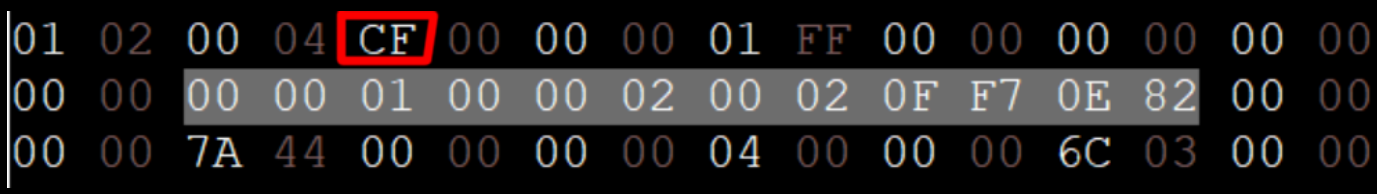
Find:: 002CD3E8
Paste:
E8 4E E5 D3 FF 81 7D 04 00 00 00 10 7C DA 50 53 51 31 C9 A1 00 0E 2E 10 8B 80 84 48 5E 00 0F B6 58 06 8D 40 10 39 D9 74 19 F6 00 01 74 0E 66 81 78 1E FF FF 75 06 66 C7 40 1E 01 00 41 8D 40 2C EB E3 59 5B 58 EB A1

CAM MODIFICATIONS

CAM EVENT MODIFICATIONS

Part 1: Control Camera Visibility/Freezes

needs activation byte **CF** and data array



Up until now, when observing the game through a camera (usually a Type 6), there were certain elements that remained invisible, like items, enemies, player etc. There are good reasons why certain elements remain off screen for some camera types, like say enemies for example. In most circumstances, we don't want enemies moving around and attacking us while we are examining an item to pick up as by default, we have no control over the player while in a Type 6 camera. So by default, most elements are removed from Type 6 cameras for one reason or another, but what is interesting is that the developers hardcoded exceptions to this for some scenarios.

For example, when we first enter r104, there is a camera shot that pans over the room which includes enemy visibility. This is done for cinematic purposes. Another example is how we see items when we open a cabinet or drawer, locker, chest etc. Normally ITM models are invisible to Type 6 cameras, but there are hardcoded exceptions for each room. These are linked via the ITA indexes, so if we were to change the ITA index, the item would no longer remain visible to the camera. The reason for this last example eludes me, but we now have more options.

What this edit does

With some hardcoded edits to the bio4.exe, we can now control what elements are visible in our cameras by injecting flags into memory. We store the data in unused arrays of the CAM file itself, with a designated flag to activate the code. The CF flag is used as there were a couple of entries we noticed that had data in the byte arrays we set aside for our data, so without the CF flag these default data sets are preserved.

It should be noted that the CAM tool by MarioKartN64 will preserve and import and edits made to these byte arrays and store them in the ‘unk0xx’ fields. It was smart of MarkioKart64n (aka Data) to preserve this information.

Explaining the data sets

(this is a rough definition that may need updating at some point)

Since these arrays of bytes are populated with several flags, we should take note of what each flag is used for.

1 2 3 4 5 6 7 8 9 10 11 12
00 00 06 03 16 03 08 01 00 01 40 90

If any of the first 4 bytes are left blank it will use default

---

1 - ??? (No Change Detected)

2 - Visibility of Screen Filters and GUIs (HUD, Film Grain)

3 - Visibility of Lights (includes shadows)

4 - Visibility of Player/Ashley/ITA/ETM/Enemy/World Objects

5 - Setting to **BA** or Higher disables certain Effects. Setting to **C0** or Higher disables skybox

6 - ??? (No Change Detected)

7 - ??? (No Change Detected)

8 - Setting to **10** Freezes/Makes Invisible Player/Ashley/ITA/Enemies Unless whitelisted
(Has priority over 2,3,4)
Setting to **0D** Creates Heat Signature Filter

9 - Freezes Effects

10 - Freezes Prompts-player movement (**0F** keeps breathing animation)
(if you're near a button or item and it says to pick it up, and you're in a camera it stops this)

11 - Freezes AEV/CAM Timers/Effects

12 - Freezes Player/CAM/Ashley/ITA/ETM/Enemy/World Objects
(Every **20**, freezes Leon's movement but not his animation
And odd freezes him completely, example **10 30 50 70**)

PRESETS

Default Key Camera:
00 00 00 60 00 02 00 12 0F F7 0E 92

Default Key Camera (No Whitelist):
00 00 00 60 00 02 00 02 0F F7 0E 92

Opening Object To (Whitelisted) Reveal Item:
00 00 00 60 00 02 00 10 0F F7 0E 92

Opening Object To (No Whitelist) Reveal Item:
00 00 00 60 00 02 00 00 0F F7 0E 92

Interaction W/Text (Freezes Everything):
00 00 00 00 00 02 00 00 BF FF FF FF

r104 Enemy Camera:
00 00 06 03 16 03 08 01 00 01 40 90

HEX CODE TO ACTIVATE FUNCTIONALITY

E8 20 F2 D0 FF EB 02 33 C0 89 86 10 07 00 00 89
Change To:
E9 4F 01 00 00 EB 02 33 C0 89 86 10 07 00 00 89
---
04 F6 05 38 40 C1 00 01 75 ED C3 CC CC CC CC CC CC CC CC CC CC CC CC CC
Change To:
04 F6 05 38 40 C1 00 01 75 ED E9 78 08 00 00 C3 CC CC CC CC CC CC CC
---
80 79 34 05 74 03 B0 01 C3 8B 81 10 07 00 00 83 B8 FC 00 00 00 01 0F 94 C0 C3 CC CC CC CC CC CC
Change To:
80 79 34 05 74 04 B0 01 EB BE 8B 81 10 07 00 00 83 B8 FC 00 00 00 01 0F 94 C0 EB AC CC CC CC CC

Find: 002F3258
Paste:
84 C0 74 05 E8 3F B5 FC FF C3

Find: 002BE7A0
Paste:
50 53 A1 00 0E 2E 10 8D 98 DA 51 00 00 83 FB 00 74 1B C7 40 58 00 00 00 00 C7 80 20 50 00 00 00 02 00 00 C7 80 70 01 00 00 00 00 00 00 5B 58 C3

Find: 002F3CD8
Paste:
8D 81 00 A0 FE FF 51 52 57 8D 90 DA 51 00 00 8B 4B 12 31 FF 83 F9 00 74 04 47 89 48 58 8B 4B 16 83 F9 00 74 07 47 89 88 20 50 00 00 8B 4B 1A 83 F9 00 74 07 47 89 88 70 01 00 00 89 3A 5F 5A 59 E8 8C F0 D0 FF E9 6B FE FF FF

Part 2: **ITM Shown in Type 6 Camera Shots**

Offset[52] = 0x1 in ITA file.
Fixes issue of ITA spawned items not being shown during Type 6 camera events. This edit is somewhat redundant considering this same thing can be accomplished with the Control Camera Visibility/Freezes edit, but there are times users may wish to simply add this byte in their ITA without having to hex edit the CAM file.

HEX CODE TO ACTIVATE FUNCTIONALITY

8B D6 E8 65 FE FF FF B0 01 5E 5D C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC
Change To:
80 7E 34 01 7F 07 81 49 04 00 08 00 00 8B D6 E8 58 FE FF FF B0 01 5E 5D C3

Part 3: **AEV-CAM Toggle**

(This edit requires all other AEV/AEV-Chain edit)

Allows Type 8 CAM entries to be toggled with AEV Type2 events.
(use Offset[128..131] = 0xCC (CAM Number *not index*) Example in AEV entry:

Event 41
Type 2 - CAM toggle

ID = 0x1
TriggerZoneHeightBoundary1 = -1377.125000
TriggerZoneHeightBoundary2 = 3634.695313
UnknownValue = 750.000000
TriggerZoneCorner0_X = -29255.898438
TriggerZoneCorner0_Y = 71853.406250
TriggerZoneCorner1_X = -27366.824219
TriggerZoneCorner1_Y = 71859.468750
TriggerZoneCorner2_X = -27418.507813
TriggerZoneCorner2_Y = 70027.046875
TriggerZoneCorner3_X = -29215.789063
TriggerZoneCorner3_Y = 69998.218750
Offset[52] = 0x2
Type = 0x2
Index = 0x39
Offset[55] = 0x1
Offset[56] = 0x2
Offset[57] = 0x1
Offset[68..71] = 0x0
Offset[72] = 0x0
Offset[73] = 0x2D
PromptMessage = 0x0
Offset[75] = 0x0
# Cut Scene Event Parameters
Offset[92..93] = 0x0
Offset[94..95] = 0x0
Offset[96..99] = 0x0
Offset[100..103] = 0x0
Offset[104..107] = 0x0
Offset[108..111] = 0x0
Offset[112..115] = 0x0

Offset[116..119] = 0x0
Offset[120..123] = 0x0
Offset[124..127] = 0x0
Offset[128..131] = 0xCC01
Offset[132..135] = 0x0
Offset[136..139] = 0x0
Offset[140..143] = 0x0
Offset[144..147] = 0x0
Offset[148..151] = 0x0
Offset[152..155] = 0x0

HEX CODE TO ACTIVATE FUNCTIONALITY

Find: 002B7AC8
Paste:
~~E8 6B D1 11 00 E8 BE FE 10 00 E8 31 B6 E6 FF C3~~
E8 6B D1 11 00 E8 BE FE 10 00 E8 31 B6 E6 FF E8 DC 69 00 00 C3 (new 1-17-2024)

Find: 00123108
Paste:
80 B8 81 00 00 00 CC 74 01 C3 60 0F B6 98 80 00 00 00 A1 00 0E 2E 10 8B 80 28 4F 00 00 0F B6 48 05 85 DB 74 12 39 CB 7F 0E 6B DB 10 8B 44 18 08 85 C0 74 03 80 30 01 61 C3

CAM EVENT MODIFICATIONS

Part 4: **No Escape Auto Cam** (Type 2 events)

Used in conjunction with Type 2 Auto events (linked with Type 11 event - requires DLL)
Makes it so that event automatically closes when it's complete and you can't escape from it.
use **Offset[152..155] = 0xCF** in AEV
Prerequisites & Dependencies 'Type 6 AEV Toggle' Edit

HEX CODE TO ACTIVATE FUNCTIONALITY

8D A4 24 00 00 00 00 6A 01 E8 F9 55 D4 FF 83 C4
Change To: :
E9 2A 03 00 00 90 90 6A 01 E8 F9 55 D4 FF 83 C4

Find: 002BE238
Paste:
8D A4 24 00 00 00 00 80 3D F0 0D 2E 10 01 0F 8C C4 FC FF FF 8B 0D 58 00 2F 10 E8 E2 74 D4 FF 6A 01 E8 B2 52 D4 FF 83 C4 04 8B 0D 58 00 2F 10 E8 CD 74 D4 FF 84 C0 74 E7 50 A1 00 0E 2E 10 83 80 E8 D1 FB FF 02 58 FE 0D F0 0D 2E 10 C3 CC

CAM EVENT MODIFICATIONS

Part 5: **Effects Keep Playing During Cam**

Must use by **CF** (Cam Flag) **Offset[152..155] = 0xCF** in any Auto Cam entry (AEV)

This edit forces cam entries with the designated flag to preserve effect visibility during the cam shot. This edit is somewhat redundant considering this same thing can be accomplished with the Control Camera Visibility/Freezes edit

HEX CODE TO ACTIVATE FUNCTIONALITY

C7 82 70 01 00 00 FF FF FF FF A1 3C 5F
Change To: :
C7 82 70 01 00 00 0F FF 0F FF A1 3C 5F

CAM EVENT MODIFICATIONS

Part 6: **Drawn Enemies During Camera Events**

In most circumstances (with some exceptions), enemies do not appear in cameras. While lots of study was put into the CAM file, we have not determined what actually allows enemies to be drawn during camera events (perhaps this is a hardcoded limitation). There are logical reasons why we would not always want this to be the case. For example, if we have a Type 6 camera that focuses on a drawer we are opening, we certainly do not wish to have enemies on screen attacking us while we are in this first person camera view.

There are times however, for cinematic reasons, that we may wish to include enemies in our camera shots (like when creating a cinematic sequence for an upcoming battle, or perhaps an opening sequence when first loading a room). For this purpose, this edit will come in handy, but it has limitations. We are only able to use 4 separate rooms with this edit, so choose them wisely! The rooms in this edit are shown in color below. Feel free to edit those to match the IDs of the rooms you wish to have this feature in.

This edit is somewhat redundant considering this same thing can be accomplished with the Control Camera Visibility/Freezes edit above.

HEX CODE TO ACTIVATE FUNCTIONALITY

---
81 88 20 50 00 00 00 00 00 10 8B 15 3C 5F
Change To:
E9 D9 00 00 00 90 90 90 90 90 8B 15 3C 5F

Find: 002BDBE8
Paste:
53 8B 98 AC 4F 00 00 66 81 FB **25 03** 74 1F 66 81 FB **1C 03** 74 18 66 81 FB **1A 02** 74 11 66 81 FB **0B 03** 74 0A 81 88 20 50 00 00 00 00 00 10 5B E9 F9 FE FF FF

r325
r31c
r21a
R30b

ROOM INIT MODIFICATIONS

SWAPPING / CLONING ROOM INIT INSTRUCTIONS

In this section we will explore how to swap or clone room INIT functions. What is this you ask? The room INIT is essentially the hardcoded scripts that get loaded when the room loads. These scripts are instructions on how the room will work. These instructions might include the following:

- IDs for moving models (like cabinets, lockers etc). There would be hardcoded CAM links tied to these IDs
- IDs for AEV events that make SMD models move (like when a stage model is moved when we interact with a certain

AEV event).
- IDs for enemies that cause certain other events to transpire (like when we kill enemy X, Y happens).
- Timers for events, directions for moving models, special effects linked to AEV events (like the r307 puzzle). The list is basically too long to write out. These are just examples of events in rooms that take place because there are instructions to make them work. These are in the room INIT.

So it was brought up to me by user Zatarita (and then later by AnnonymousUser), that there are tables that contain the IDs of the INIT functions, and that we can essentially clone an INIT from one room, and use it in another. For example, I was successful in creating another r307 room EXACTLY as the real one in r10b. Everything was the same; the puzzle with the effects, the moving door, the camera for the Regenerator on the table etc. This is a very exciting development for stage modding because we can:

- repace INITs that are useless (like r316 there is a cutscene, then the map gets forced loaded, then there is another cutscene event when we open the ETM door to downstairs). With cloning INIT files we can replace all of these instructions with those of another room. For example, if we wanted to just wipe the slate clean with minimal scripts, we could clone r312 to to r316 so that r316 wouldn't have all these crazy scripts. We could do the same for r100 so there are no cutscenes, no moving truck, no hardcoded scripts. THe possibilities are exciting too because we can now have multiple rooms with the same INIT. For example, I have taken the r307 puzzle and placed it in a whole bunch of other rooms to utilize the functionality of the puzzle elsewhere.

We could do the same with other things like the cranks Ashley has to use to open the gates. Imagine being able to use this in 10 different rooms, and not be limited to just the original room. Very cool!

GAME FILE DEPENDENCIES

If we are to clone INITs from one room to another it is really important to understand that there are **DEPENDENCIES**. The very fact that scripts call on objects means that if we clone an INIT, and the dependent objects are not present in the UDAS, or other game files, the room may well crash. This is why when first cloning a room (while you learn the basics), that all the dependencies are present.

For example, when I cloned r307 to r10b, I had to ensure that all the parts were there if I wanted it to work. That meant basically porting all of the UDAS files from r307 so everything would work. Later on, I could simply use the parts that I wanted, and leave the rest. The important thing to note is that we need to understand what are the essential elements (or used objects) of the cloned room. For example, if I wanted to make sure that the puzzle from r307 worked in r10b, I had to make sure that all the effects that the puzzle used were present in the new room. I also had to make sure that all the textures were present etc. The easiest way to do all of this is to simply rename the source room UDAS to that of the target room, and then update any TPL calls to the new pack file (or simply side-chain the original room pack file).

I have not extensively tested all types of dependencies, like enemies for example. I have mostly spent my time testing things out with AEV events, SMD models and EFFECTS that are dependencies. I imagine that if there were ESL entries linked to AEV events that the IDs would need to be the same as the original room for things to work properly.

INIT TABLES

So let's get down to how this works. There are essentially three tables that have been mapped out so far: St1, St2, and St3. It is likely that there are Tables rooms in the other campaigns and Mercs, but I have not tested (It might be possible to create more Mercs rooms in the Main campaign with this, but who knows). For now let's look at the tables, and then observe how we calculate the INIT IDs.

Each Stage has a table (or array of bytes) that starts at a certain Offset (or **Base Address**, or Prolog.. whatever you want to call it). In this guide below we will see a value that represents part of a calculation (left columns). This value is the '**Room Value**'. This value will be part of a calculation we will see later on. For now here is the list of the three stages and their '**Room Values**':

Stage 1 **Base Address** (prolog): **@Offset 4DE450**
Stage 2 **Base Address** ((prolog): **@Offset 4A9E70**
Stage 3 **Base Address** ((prolog): **@offset 43E530**

ST1 - @Offset 4DE450	ST2 - @Offset 4A9E70	ST3 - @offset 43E530
0 100	0 200	0 300
1 101	1 201	1 301
2 102	2 202	2 303
3 103	3 203	3 304
4 104	4 204	4 305
5 105	5 205	5 306
6 106	6 206	6 307 (used in example below)
7 107	7 207	7 308
8 108	8 208	8 309
9 109	9 209	9 30a
A 10a	A 20a	A 30b
B 10b	B 20b	B 30c
C 10c	C 20c	C 30d
D 10d	D 20d	D 30e
E 10e	E 20e	E 30f
F 10f	F 20f	F 310
10 111	10 210	10 311
11 112	11 211	11 312
12 113	12 212	12 315
13 117	13 213	13 316
14 118	14 214	14 317
15 119	15 215	15 318
16 11a	16 216	16 31a
17 11b	17 217	17 31b
18 11c	18 218	18 31c
19 11d	19 219 (used in example below)	19 31d
1A 11e	1A 21a	1A 320
1B 11f	1B 21b	1B 321
1C 120	1C 21d	1C 325
1D 121	1D 220	1D 326
1E 122	1E 221	1E 327
1F 123	1F 222	1F 328
20 124	20 223	20 329
21 125	21 224	21 330
22 126	22 225	22 331
23 127	23 226	23 332
24 128	24 227	24 333
25 129	25 228	25 335
26 12a	26 229	26 336
27 12b	27 22a	27 337
28 12c	28 22b	28 338
29 12d	29 22c	29 339
2A 12e	2A 22e	2A 340
2B 12f	2B 22f	2B 341
2C 130	2C 230	2C 342
2D 131	2D 231	2D 343
2E 132	2E 232	2E 344
2F 133	2F 233	2F 345

30	134
31	135
32	136
33	137

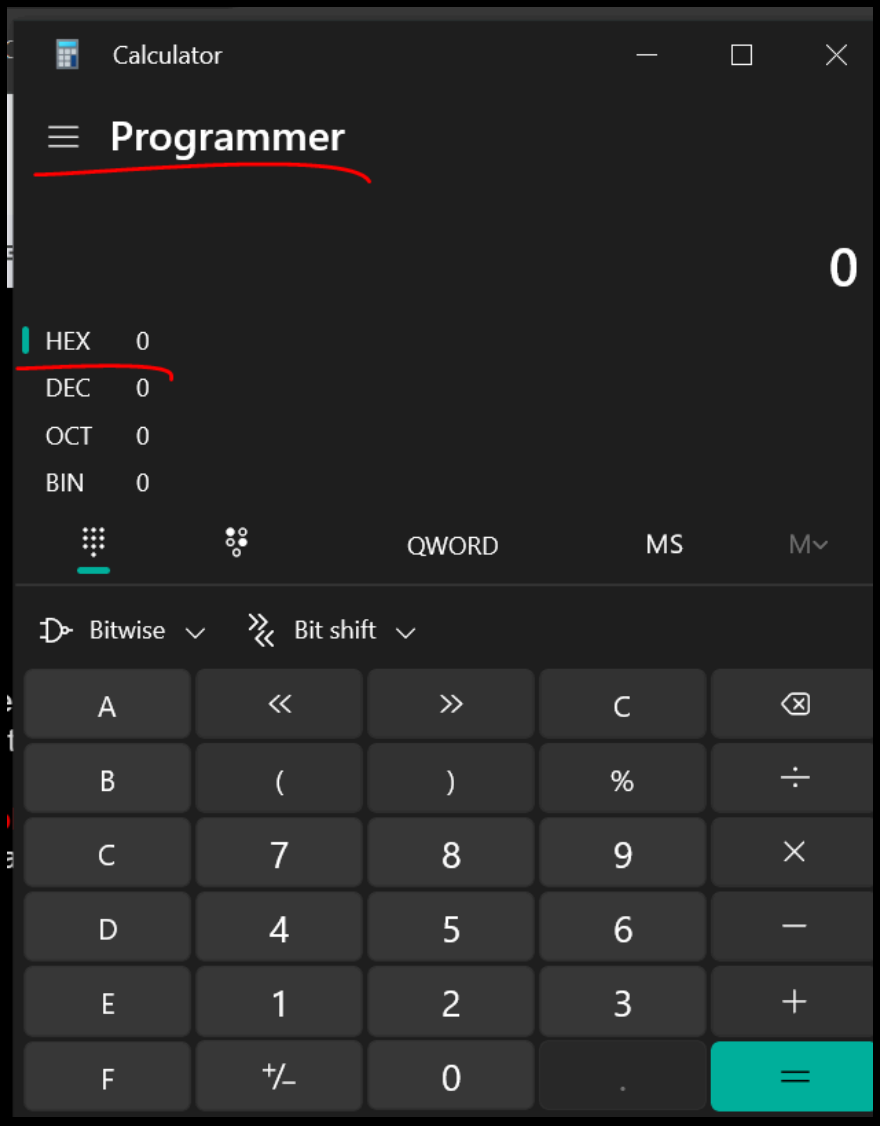
30	234
31	235
32	236
33	237
34	238
35	239
36	240
37	241
38	242
39	243
3A	244
3B	245

30	346
31	347
32	348
33	349
34	350
35	351
36	352
37	353
38	354
39	355
3A	356
3B	357
3C	358
3D	359

HOW TO CALCULATE THE ADDRESS FOR A ROOM'S INIT ID

As described above, each stage (ST1, ST2, ST3), has a **Base Address** (or offset) where the Table for the instructions for the room of that stage's instructions start.
Each set of individual Room INIT instructions has a **UNIQUE** **four byte INIT ID** which can be swapped out with other bytes to clone a room.

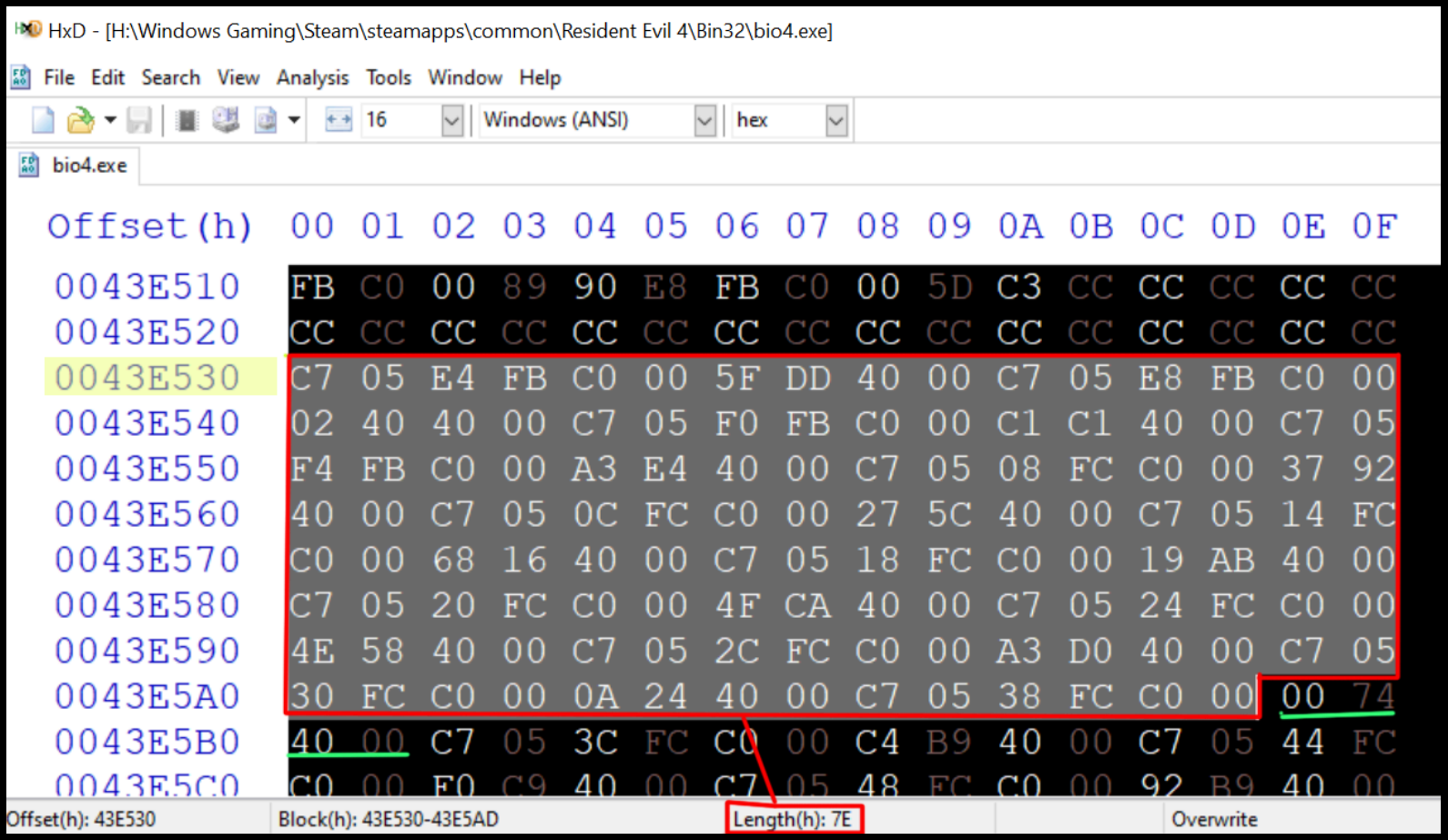
To get the **INIT ID** values for any given room, we take the **'Room Value'** from the table above, and **multiply it by 14, and add 6**. This calculated value will be the **length of the string from the start of the base address of the stage**, to the start of the **INIT ID** for the room. These values are **HEXIDECIMAL**, so it will need to be calculate in 'Programmer' mode in WIndows Calculator under the HEX:



For the sake of example let's say we are going to clone **r307** over to **r219**, so we will do the following:

1 - Get the Room **INIT ID** for r307. To do this we take the **'Room Value'** of **6** (derived from the table above) and **multiply it by 14 + 6**:
6 x 14 + 6 = **7E** . 7E is the **total length in bytes** from the start of the base address for ST3 to where the r307 **INIT ID** is (four bytes)

2 - Next we open our bio4.exe in a hex editor and navigate to **@offset 43E530** (use CTRL G to bring up the offset navigator in HxD):



So the area surrounded in red shows a total length of **7E** from the start of the **Base Address** for ST3 to the value underlined in green which is
00 74 40 00. This is the INIT ID value for r307.

So now that we have our **INIT ID** for r307, we do the same calculation for our target room, r219:

19 219 (room r219)
19 x **14** + **6** = **1FA**

From the base address for St2 (ST2 - **4A9E70)** we go **1FA** bytes forward. This lands us at: @4AA06A. There we see the 4 byte array: **2F 86 40 00** this is the room **INIT ID** for r219.

So now the last step is to simply replace the r219 **INIT ID** with that of r307 (pasting **0 74 40 00** over **2F 86 40 00**). While this is the last step in actually cloning the INIT, we must remember that there are almost always **DEPENDENCIES** that will need to be copied over as well. This could be any of the files included in the UDAS. Remember all of the instructions in these INITs are very specific about which IDs they use for events, enemies, models, and effect groups. Changing any of these could result in the scripts not working, or the room crashing. Feel free to experiment with each room to see what works for you.

EXCEPTIONS FOR CERTAIN INSTRUCTIONS

While swapping out the INIT we are usually going to get most of the functionality of the original room, however there are some instructions that are independent of the room INIT though, which are loaded for particular rooms no matter which INIT we use. For example, when I was assigning various INITs to r316 last night, the result was as expected; the cutscene did not play right away, and the map was not forced to load as it normally does **BUT.....** the interaction with the first ETM door still triggers an AEV event like it does in the vanilla game. So there are some instructions that will prevail even if we swap out the INIT. Users will need to be aware that this could happen and that some instructions can not be ported with this INIT swap.

Not all rooms may work. I have only tested this in a limited capacity. Feel free to experiment!

SWAPPING / CLONING ENEMY AI

Allows for swapping out Enemy Behavior

This tutorial will show users how to swap enemy behavior from one enemy to another. An example of this would be taking
TO BE CONTINUED

MISC MODIFICATIONS

AEV Merchant

Turns AEV event into Merchant shop.
Offset[128..132] = BB in AEV

HEX CODE TO ACTIVATE FUNCTIONALITY

E8 54 79 E7 FF 8B 56 10 8B 46 08 8B 4E 04 52 50
Change To:
E9 B1 00 00 00 8B 56 10 8B 46 08 8B 4E 04 52 50

Find: 00189FF8
Paste:
E8 9E 78 E7 FF 80 BB 80 00 00 00 BB 75 0C 6A 00 6A 10 E8 A0 2C E8 FF 83 C4 08 E9 30 FF FF FF

EM Incompatibility Issue Fix

Allows for placing enemies in any order in ESL

HEX CODE TO ACTIVATE FUNCTIONALITY

~~74 13 81 C1 98 00 00 00 40 81 F9 60 02 00 00 72 E8 33 C0 5D C3 69 C0 98 00 00 00 05 20 38 C6 00 5D C3 CC CC CC CC CC CC CC CC CC CC CC CC CC~~
~~**Change To:**~~
~~74 13 81 C1 98 00 00 00 40 81 F9 60 02 00 00 72 E8 33 C0 5D C3 69 C0 98 00 00 00 05 20 38 C6 00 60 8B 80 88 00 00 00 8B 48 34 FF D1 61 5D C3~~

NEW EM Incompatibility Issue Fix (5-2023)
---
74 13 81 C1 98 00 00 00 40 81 F9 60 02 00 00 72 E8 33 C0 5D C3 69 C0 98 00 00 00 05 20 38 C6 00 5D C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
Change To:
74 13 81 C1 98 00 00 00 40 81 F9 60 02 00 00 72 E8 33 C0 5D C3 69 C0 98 00 00 00 05 20 38 C6 00 60 8B 80 88 00 00 00 85 C0 74 09 8B 48 34 85 C9 74 02 FF D1 61 5D C3

XWB/XSB Sideload by Qingsheng (Companion DLL version)

Adds functionality of side-loading xwb & xsb files. Files are loaded into:

\Resident Evil 4\BIO4\Data\rxxx where xxx is room number.

HEX CODE TO ACTIVATE FUNCTIONALITY

E8 3E FA FF FF 8B 4D FC 83 C4 10 5F 33 CD 5E E8 09 59 0F 00 8B E5 5D C3 CC CC CC

Paste:
50 8B 44 24 04 8D 80 E2 3A 60 00 C7 80 2B 91 7A 00 30 30 30 2E C7 80 1B 91 7A 00 42 49 4F 34 C7 80 1F 91 7A 00 2F 44 61 74 C7 80 23 91 7A 00 61 2F 72 31 C7 80 27 91 7A 00 30 30 2F 30 58 E9 FD FB FF FF

Find: 002ACD92
Paste:
60 A1 00 0E 2E 10 C7 80 2F 91 7A 00 64 61 73 00 8B 7F 03 89 B8 2A 91 7A 00 6A 00 6A 01 6A 03 6A 00 6A 01 68 00 00 00 80 8D 80 1B 91 7A 00 50 E8 8F 36 D5 FF 83 C4 1C 8B D8 50 E8 19 8D D5 FF 83 C4 04 83 FB 00 61 7E 22 8D BA 20 91 7A 00 57 E8 BA F0 EF FF 50 A1 00 0E 2E 10 C7 80 2A 91 7A 00 30 30 30 30 58 E9 C2 FC FF FF 57 E8 9E F0 EF FF 50 A1 00 0E 2E 10 C7 80 2A 91 7A 00 30 30 30 30 58 E9 A6 FC FF FF

Change Memory Allocation Amount

This is a guide to help along with user-end editing of the bio4.exe to manage memory usage. Changing how much memory is allocated can often help with additional requirements for heavy modding that demands more memory (high resolution textures, combined with over a dozen different types of enemies, along with sideloaded textures can really add to the limit).

Currently the total amount of memory the game is able to commit is around 3.5GB of memory. That means if the resources we add to the game exceed this amount the game will certainly crash. To monitor memory usage in real-time, see this tutorial on how to use MSI Afterburner/RivaTuner to add an overlay to your screen.

Find: 005728AA
Find: 005728BE
Find: 005728DF
Find: 0057290D
Find: 0025555D

At each of the offsets above you will paste one of the presets below (or make your own). **The values entered to all of the offsets above must match.**
After running several tests it was observed that anywhere from 100 to 250mb is a good setting for this. Anything higher will push the amount used closer to the 3.5GB memory limit. Original values are:

00 00 D0 01 = 30408704(DEC) (**29.00 MB**)

Preset Values

MEM2 MOD A= **00 00 C0 03** = 62914560 (DEC) (**60.00 MB**)
MEM2 MOD B = **00 00 40 06** = 104857600 (DEC) (**100.00 MB**)
MEM2 MOD C = **00 00 D0 07** = 131072000 (DEC) (**125.00 MB**)
MEM2 MOD D = **00 00 40 0B** = 188743680 (DEC) (**180.00 MB**)
MEM2 MOD E = **00 00 A0 0F** = 262144000 (DEC) (**250.00 MB**)
MEM2 MOD F = **00 00 40 1F** = 524288000 (DEC) (**500.00 MB**)

Companion DLL Double File Load Fix (new)

- Fixes Companion DLL issue of loading sidechained .pack files in the ‘TEXTURES’ folder twice upon retrying
- Removes Companion DLL persistent check of the "TEXTURES" folder.
- Fixes shared EFF textures incorrectly loading with some sidechained enemies.

(if you have an older version of this edit, make sure to revert that edit before installing this).

HEX CODE TO ACTIVATE FUNCTIONALITY
55 8B EC 83 EC 44 6A 29 68 FF 00 00 00

Change To:
55 8B EC 83 EC 44 6A 29 E9 8B 09 00 00

Find: 002D8A18
Paste:
C7 05 EF C8 26 10 E9 F7 01 00 C7 05 F3 C8 26 10 00 90 83 7E C7 05 61 B5 26 10 E9 C5 18 00 C7 05 65 B5 26 10 00 90 90 75 C7 05 EE 17 27 10 E9 7D 0A 08 C7 05 F2 17 27 10 00 83 BE 28 C7 05 70 22 2F 10 E8 0B 92 F7 C7 05 74 22 2F 10 FF 6A 00 6A C7 05 78 22 2F 10 00 6A 01 8B C7 05 7C 22 2F 10 CE E8 FE 91 C7 05 80 22 2F 10 F7 FF E9 6C C7 05 84 22 2F 10 F5 F7 FF 90 68 FF 00 00 00 E9 F3 F5 FF FF

Companion DLL No 2x Allocation

Fixes the Companion DLL doubling the memory allocation of certain files, which wastes memory.

HEX CODE TO ACTIVATE FUNCTIONALITY
BF 10 00 00 00 8B 45 08 8D B4 38 07 01 00 00 68
Change To:
BF 10 00 00 00 8B 45 08 EB B8 90 90 90 90 90 68

Find: 002AA220
Paste:
8D B4 38 07 01 00 00 50 A1 00 0E 2E 10 85 C0 75 03 58 EB 39 8D 80 10 C7 00 00 89 04 24 83 E6 80 EB 33

Companion DLL Automatic Key Unlock

If the first ItemID in any **events.cfg** entry is **0xFFFF00xx** (**xx= [ITEMID]** then it will unlock without opening your inventory WILL NOT TAKE YOUR ITEM AWAY, only serves as an item check.

Example:

```
event_lock0 %Auto_Key_01%
{
    ItemID 0xFFFF0091
    Message 0x1
    Sound 0x2
    Unknown0 0x00
    Unknown1 0x00
    EventIndex 0x20
    event_unlock
    {
        ItemID 0x91
        Sound 0xFF
        Message 0x03
        LockEventIndex 0x20
        Unlocked false
    }
}
```

HEX CODE TO ACTIVATE FUNCTIONALITY
66 8B C7 5F 5E 5B 8B E5 5D C2 04

Change To: :
E9 A7 FA FF FF 5B 8B E5 5D C2 04

Find: 003046A0

Paste:
66 89 F8 66 85 C0 74 18 81 7D 04 12 A5 27 10 75 0F 66 81 7D 0A FF FF 75 07 66 31 C0 66 89 59 08 5F 5E E9 32 05 00 00

DLL Key Unlock No Message No cam

Allows for preventing messages and cameras from being assigned in the events.cfg. Previous methods of simply assigning values of unused CAM entries resulted in camera jitter. The message part of this makes it so we do not need to create blank MDT messages for CAM shots without text.

If "Message" equals 0xFFFFFFFF in the events.cfg, then it will not play a message and prevent the game from freezing for a few frames.
If "Unknown0" equals 0xFFFFFFFF in the events.cfg, then no camera will play and prevent the game from jittering the camera around (It tries to center the camera to where the player is facing).

HEX CODE TO ACTIVATE FUNCTIONALITY

B8 01 00 00 00 F6 C3 01 75 05 B8 02 00
Change To: :
EB 96 90 90 90 F6 C3 01 75 05 B8 02 00

8B 90 28 4F 00 00 8B 41 2C 89 51
Change To: :
8B 90 28 4F 00 00 EB 2B 90 89 51

Find: 002F58D8

Paste:
8B 41 2C 81 3C 24 1F A5 27 10 75 CA 81 7D F4 FF FF FF FF 75 C1 C2 04 00

Find: 002C14F0

Paste:
81 7D 08 FF FF FF FF 74 07 B8 01 00 00 00 EB 5D 5E 5B 5D C3

Higher EM XWB/XSB Limit (updated 7-16-2023)

Allows for more XSB/XWB files to be loaded into memory than previously allowed.
Changelog:
* Some kind of intermittent sound data needed more space, changed size from 0.06 MB to 1 MB (Global MEM) (FIXES save screen)
* New XSB/XWB's weren't replacing old ones correctly causing sound issues like wrong guns sounds and no "RESIDENT EVIL 4" voice and other mishaps

HEX CODE TO ACTIVATE FUNCTIONALITY

85 5C 96 0C 74 15 0F B6 8C 06 18 09 00 00 3B CF 74 06 40 83 F8 06 7C D6 83 C8 FF 5F 5E 5B 5D C3 CC CC CC CC CC CC CC

Change To:
85 5C 96 0C 74 1C 0F B6 8C 06 18 09 00 00 39 F9 74 0D 40 83 F8 06 7C D6 E8 50 D3 FF FF EB CF 83 C8 FF 5F 5E 5B 5D C3
---
88 9C 01 18 09 00 00 EB 0D 8B 45 14 8A 5D 10 8D
Change To:
E9 D6 39 00 00 90 90 EB 0D 8B 45 14 8A 5D 10 8D

E8 18 1B A9 FF 83 C4 14 5F 5E 5D C3 - **revert if already installed**
Change To:
E9 C2 8D FF FF 83 C4 14 5F 5E 5D C3
---
50 C7 03 00 00 00 00 C7 00 00 00 00 00
Change To:
E9 0E FC FF FF 90 90 C7 00 00 00 00 00

E8 39 FD FF FF 85 C0 74 1B 68 60 5C
Change To:
E9 E1 FB FF FF 85 C0 74 1B 68 60 5C

E8 B3 FD FF FF 85 C0 74 1B 68 30 5C
Change To:
E9 C2 FB FF FF 85 C0 74 1B 68 30 5C

0F B7 06 B9 FF FF 00 00 66 3B C1 0F 84
Change To:
E9 18 FF FF FF 90 90 90 66 3B C1 0F 84

8B 04 CD 7C 4C 27 01 8B 4D D8 8B 10 8B 52 10 6A 00 6A 00 51 50 FF D2
Change To:
8D 04 CD 7C 4C 27 01 E9 AB 00 00 00 8B 52 10 6A 00 6A 00 51 50 FF D2

B9 01 00 00 00 66 89 08 BA 00 01 00 00 66 @ 00579D17
Change To:
E8 B4 01 00 00 89 08 90 BA 00 01 00 00 66

BA 01 00 00 00 89 50 14 0F B7 71 08
Change To:
E8 0E EF FF FF 89 50 14 0F B7 71 08
---
05 00 00 01 00 68 01 80 00 00 81
Change To:
05 00 00 10 00 68 01 80 00 00 81

Find: 00568900
Paste:
83 F8 06 7C 2C 83 BE E8 96 03 00 00 75 1B 89 86 EC 96 03 00 60 68 00 10 00 00 6A 40 FF 96 14 12 4D 00 89 86 E8 96 03 00 61 8B B6 E8 96 03 00 31 C0 C3

Find: 0056EFD0
Paste:

50 52 8B 81 EC 96 03 00 83 F8 06 7C 1B 8B 91 E8 96 03 00 85 D2 74 11 88 9C 10 18 09 00 00 FF 81 EC 96 03 00 5A 58 EB 09 5A 58 88 9C 08 18 09 00 00 E9 05 C6 FF FF

Find: 0056E9C8 Revert this edt to CC CC CC CC …..

Paste:

~~E8 51 8D A9 FF 60 8B 44 24 20 8B 80 9C A7 67 00 8B 98 E8 96 03 00 85 DB 74 5A 8B 88 EC 96 03 00 83 F9 07 7C 34 83 E9 07 31 D2 50 53 51 52 53 C7 03 00 00 00 00 C7 43 0C 00 00 00 00 E8 7C 76 A9 FF 8D 5B 0C 53 E8 7E 9D A9 FF 83 C4 08 5A 59 5B 58 42 8D 5B 18 39 CA 7E D1 31 D2 89 90 EC 96 03 00 8B B0 E8 96 03 00 89 90 E8 96 03 00 56 FF 90 18 12 4D 00 61 E9 C4 71 00 00~~

Find: 00575A80

Paste:

83 38 00 74 31 52 51 8B 97 38 12 00 00 85 D2 74 23 8B 88 A0 00 00 00 39 D1 75 19 8B 8F 3C 12 00 00 83 F9 07 7C 0E 8D 49 F9 6B C9 18 8D 14 11 8B C2 8D 5A 0C 59 5A C3 E8 C4 FF FF FF 50 C7 03 00 00 00 00 E9 E3 03 00 00 50 53 8B C6 E8 AF FF FF FF 8B F0 5B 58 E8 46 01 00 00 E9 08 04 00 00 50 53 8B C6 E8 98 FF FF FF 8B F3 5B 58 E8 DF 01 00 00 E9 27 04 00 00

Find: 0056BF58

Paste:

0F B7 06 66 3D FF FF 75 43 53 56 57 B8 FF FF FF FF 8B 99 E8 96 03 00 85 DB 74 2C 8B B1 EC 96 03 00 83 FE 07 7C 21 BF 06 00 00 00 47 38 94 3B 17 09 00 00 75 0E 8B C7 69 C0 00 00 01 00 66 B8 08 00 EB 04 39 F7 7C E4 5F 5E 5B 89 06 B9 FF FF 00 00 E9 95 00 00 00

Find: 00575760

Paste:

50 53 51 8D 0C CD 00 00 00 00 29 C8 8B 98 50 01 00 00 83 FB 07 7C 2A 8B 80 4C 01 00 00 85 C0 74 20 0F B7 5E 16 83 FB 07 7C 17 8D 5B F9 6B DB 18 8D 5B 14 8B 04 18 85 C0 74 07 59 5B 83 C4 04 EB 05 59 5B 58 8B 00 8B 4D D8 8B 10 E9 05 FF FF FF

Find: 00579ED0

Paste:

8B 4D 08 66 B9 01 00 C3 8B 11 66 BA 01 00 C3

NEW (updated 7-16-2023)

---

83 C4 0C 38 5A 17 74 08 81 08 00 00 00 40 EB 06 81

Change To:

EB 50 90 38 5A 17 74 08 81 08 00 00 00 40 EB 06 81

E8 18 1B A9 FF 83 C4 14 5F 5E 5D C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC

Change To:

E8 18 1B A9 FF E8 BD 8D FF FF 83 C4 14 5F 5E 5D C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC

Find: 00252AB0

Paste:

FF B2 1D F9 D1 FF E8 0D BF 31 00 83 C4 10 EB A1

Find: 0056E9C8

Paste:

60 8B 44 24 24 8B 80 9C A7 67 00 8B 98 E8 96 03 00 85 DB 74 5A 8B 88 EC 96 03 00 83 F9 07 7C 34 83 E9 07 31 D2 50 53 51 52 53 C7 03 00 00 00 00 C7 43 0C 00 00 00 00 E8 81 76 A9 FF 8D 5B 0C 53 E8 83 9D A9 FF 83 C4 08 5A 59

5B 58 42 8D 5B 18 39 CA 7E D1 31 D2 89 90 EC 96 03 00 8B B0 E8 96 03 00 89 90 E8 96 03 00 56 FF 90 18 12 4D 00 61 C3 CC CC CC CC CC CC CC CC CC

Allow EM's to climb AEV ladders

(this is not an edit, but an explanation)

The summary below explains how to edit the original table that governs which enemies will be allowed to climb Type10 AEV ladders.

ORIGINAL TABLE @offset **000467E0**
01 = Can't climb AEV ladders
00 = Can climb AEV ladders

00 00 00 00 00 01 00 00 01 00 00 00 00 00 00 00 00 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 00 00 00 00 00 00 00 00 00 00 00 01 00 00 01 00

The first **00** is em10, second is em11, the third is em12 etc. The first restriction **01** in this table appears for the entry used for em15. If we navigate all the way to the end of the table we arrive at the entry for em3f. The original ladder restriction table spans from em10 to em3f in ascending sequential order.

So what do we do for enemies with IDs outside the range of this table (like ema5 for example, which is an extra enemy version of em15 that is used in other scenarios)? Well, AnonymousUser has extended the table with a small edit. See edit below:

Allow Additional EM's to climb AEV ladders

Extends the ladder restriction table to include enemies from em40 to emff.

By default, any enemy with an ID higher than 3F will NOT be able to climb ladders, and will bug out if they don't have the supporting animations. Use whatever ID you use to spawn in the ESL and subtract it by 10 in hex. For example:

ema5 uses ESL ID 65, so we would subtract 10 to get 55 (HEX VALUE). So we would go to 00046820 and then move 55 steps forward to change the value for ema5.

-For enemies with and ESL ID less than 10 the math goes as follows:

Example 07
07 minus 10 = FA
then: 000467E0 + FA

HEX CODE TO ACTIVATE FUNCTIONALITY
0F B6 87 00 01 00 00 83 C0 F0 83 F8 3F 77 E1
Change To:
0F B6 87 00 01 00 00 83 C0 F0 0F B6 C0 90 90

Find: 00046820
Paste:
**01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01

Note:
(setting all these to **01** will default all enemies beyond em3f to NOT access fixed AEV Type 10 ladders. Setting all these values to **00** will default them all to be able to use ladders.

IMPORTANT NOTE ABOUT ENABLING LADDER ACCESS TO ENEMIES:
It should be noted that the enemies who by default are not allowed to access ladders are missing the animation files in their udas files to actually climb the ladders. By default, any EM higher than 3F will not be able to climb ladders and will be missing these animations. If we simply change their value in the table without adding the supporting animation files, the enemy will bug out when they climb ladders (they will glitch out and eventually disappear from the room).

ADDING MISSING ANIMATIONS
- To add the supporting animations we simply copy the files from an emxx.udas that has them (em12 or em13 will work. See list of files below).
- next paste the renamed files into the extracted udas folder for the enemy we are editing.
- next we need to update the **.idx** file to point to these new files instead of the **.DMY** files that were present beforehand. For example:

File_193 = ema5\ema5_193.DMY
File_194 = ema5\ema5_194.DMY
File_195 = ema5\ema5_195.DMY
File_196 = ema5\ema5_196.DMY
File_197 = ema5\ema5_197.DMY

Change To:

File_193 = ema5\ema5_193.SEQ
File_194 = ema5\ema5_194.FCV
File_195 = ema5\ema5_195.SEQ
File_196 = ema5\ema5_196.FCV
File_197 = ema5\ema5_197.SEQ

It should be noted that some enemies will simply not be able to climb ladders because they are not designed to, like wolves for example.

Keep EARs Visible After Death

After Leon Dies, the game disables the visibility of EARs; this fixes that.

E8 16 60 DC FF A1 3C 5F C0 00 81 88 70 01 00 00 00 00 40 00
Change To:
E8 16 60 DC FF A1 3C 5F C0 00 90 90 90 90 90 90 90 90 90 90

Verdugo No Teleport (Every Room Except r221)

No companion needed (see Companion Edit below this)

HEX CODE TO ACTIVATE FUNCTIONALITY
D8 9F 78 03 00 00 DF E0 F6 C4 41 75 10 33 C0 8B
Change To:
EB A8 90 90 90 90 DF E0 F6 C4 41 75 10 33 C0 8B

Find: 000C2440

Paste:
8B 4D 04 8D 89 F1 B5 78 00 66 81 B9 AC 4F 00 00 21 02 75 4F D8 9F 78 03 00 00 EB 40

COMPANION VERSION:

Verdugo No Teleport (Every Room Except r221)

HEX CODE TO ACTIVATE FUNCTIONALITY

Verdugo No Teleport (Every Room Except r221)

D8 9F 78 03 00 00 DF E0 F6 C4 41 75 10 33 C0 8B
Change To:
EB A8 90 90 90 90 DF E0 F6 C4 41 75 10 33 C0 8B

Find:: 000C2440
Paste:
50 A1 00 0E 2E 10 66 81 B8 AC 4F 00 00 21 02 75 09 D8 9F 78 03 00 00 58 EB 42 58 EB 46

Allow Custom CES (Chapter Ending Screens) While Excluding Normal CES

This edit will allow users to set custom locations for Chapter End Screens. These will work as normal, with updated game stats and all. If we are using the Companion DLL we must set **SkipEndScreen** to false in the **dinput8.cfg.** More instructions on how to use this edit are included at the bottom of the edit.

HEX CODE TO ACTIVATE FUNCTIONALITY

E8 9E 3B D4 FF 68 00 00 00 80 53 6A 0A 8B C3 50
Change To:
E9 45 01 00 00 68 00 00 00 80 53 6A 0A 8B C3 50
---
88 81 9A 4F 00 00 A1 3C 5F
Change To:
E9 03 08 00 00 90 A1 3C 5F

F6 C4 41 75 33 53 68 00 00 00 20 E8 28 C4 D4 FF
Change To:
E9 03 05 00 00 53 68 00 00 00 20 E8 28 C4 D4 FF

E8 BC 86 D4 FF F6 05 38 40
Change To:
E9 B2 04 00 00 F6 05 38 40

E8 2B 0B D4 FF A0 21 41
Change To:
E9 D9 02 00 00 A0 21 41

E8 E4 03 D4 FF A0 21 41
Change To:
E9 81 02 00 00 A0 21 41

E8 CA 15 D4 FF 6A 28 E8 B7 67 D4 FF 83 C4 20 8B
Change To:
E9 2E 02 00 00 6A 28 E8 B7 67 D4 FF 83 C4 20 8B

0F BE 45 FF 48 5E 75 12 6A 0A 53 E8 2A FC D3 FF
Change To:
E9 21 02 00 00 5E 75 12 6A 0A 53 E8 2A FC D3 FF

E9 8E 00 00 00 6A 05 53 E8 32 83 D4 FF 83 C4 18
Change To:
E9 1D 03 00 00 6A 05 53 E8 32 83 D4 FF 83 C4 18

Find: 002C20F8
Paste:
50 A1 00 0E 2E 10 80 B8 A1 D4 00 00 FF 58 0F 85 1C FB FF FF F6 C4 41 0F 85 13 FB FF FF E9 DB FA FF FF E8 05 82 D4 FF 50 A1 00 0E 2E 10 80 B8 A1 D4 00 00 FF 58 0F 85 51 FB FF FF E9 30 FB FF FF E8 4D 08 D4 FF 50 A1 00 0E 2E 10 80 B8 A1 D4 00 00 FF 58 0F 85 2A FD FF FF E9 09 FD FF FF E8 5E 01 D4 FF 50 A1 00 0E 2E 10 80 B8 A1 D4 00 00 FF 58 0F 85 85 FD FF FF E9 61 FD FF FF 50 A1 00 0E 2E 10 80 B8 A1 D4 00 00 FF 58 75 05 E8 87 13 D4 FF E9 B8 FD FF FF 50 A1 00 0E 2E 10 80 B8 A1 D4 00 00 FF 58 5E 75 0A 0F BE 45 FF 48 E9 C5 FD FF FF 0F BE 45 FF 48 E9 CF FD FF FF 50 A1 00 0E 2E 10 80 B8 A1 D4 00 00 FF 58 0F 85 DA FC FF FF E9 58 FD FF FF 50 A1 00 0E 2E 10 81 B0 1C 50 00 00 00 00 00 10 58 E8 43 3A D4 FF E9 A0 FE FF FF 88 81 9A 4F 00 00 81 89 1C 50 00 00 00 00 00 10 E9 E9 F7 FF FF

Custom CES (Chapter Ending Screens)

0F 8C A8 00 00 00 A1 D4 37
Change To:
E9 A9 00 00 00 90 A1 D4 37

E8 18 A6 D4 FF A1 3C 5F
Change To:
E9 7C ED FF FF A1 3C 5F

Find: 002C2CC8 (this code below updated 10-31-2022)
Paste:

~~60 A1 00 0E 2E 10 8D 98 70 CA A6 FF 8B 88 AC 4F 00 00 66 8B 90 B0 4F 00 00 31 F6 66 81 7C 33 02 CC CC 74 3D 66 3B 4C B3 02 75 31 66 3B 14 B3 75 2B 50 51 8D 88 F4 CD 00 00 E8 B1 74 D4 FF F7 00 00 00 80 00 75 26 58 90 6A 02 56 C6 80 A1 D4 00 00 FF E8 85 00 D4 FF 83 C4 08 EB 05 46 90 90 EB BA 61 E8 35 B8 D4 FF E9 18 12 00 00 66 81 38 25 03 75 11 F7 00 00 00 00 10 75 09 81 08 00 00 00 10 58 EB C4 58 EB DA~~

—————**Updated 12-29-2023**——————————

60 A1 00 0E 2E 10 8D 98 98 C2 EA FF 8B 88 AC 4F 00 00 66 8B 90 B0 4F 00 00 31 F6 66 81 7C 33 02 CC CC 74 3D 66 3B 4C B3 02 75 31 66 3B 14 B3 75 2B 50 51 8D 88 F4 CD 00 00 E8 B1 74 D4 FF F7 00 00 00 80 00 75 26 58 90 6A FF 56 C6 80 A1 D4 00 00 FF E8 85 00 D4 FF 83 C4 08 EB 05 46 90 90 EB BA 61 E8 35 B8 D4 FF E9 18 12 00 00 66 81 38 25 03 75 11 F7 00 00 00 00 10 75 09 81 08 00 00 00 10 58 EB C4 58 EB DA

~~**Find: 002C2D50 (important)**~~

Find: 00702578 (important) New CodeCave address ———Updated 12-29-2023———

In this section of the paste, we enter the Room# in pairs of **Last Entered** ---> **Destination**
CES will happen only once in a destination room and then never again.

Note: For this to work properly, when you go into the destination room, it must be the first time you enter.
You can reuse **Last Entered** rooms but not **Destination Rooms**
1st pair will use 1st CES, 2nd will use 2nd CES, etc.

Example Paste (this is code for REmix):

20 02 0A 02 FF FF FF FF FF FF FF FF 0C 01 13 02 FF FF FF FF FF FF FF FF 07 02 07 01 FF FF FF FF FF FF FF FF FF FF FF FF 25 02 1D 02 FF FF FF FF FF FF FF FF 21 02 10 02

use **FF FF FF FF** to skip over a subchapter

r220 ---> **r20a**
r10c ---> **r213**
r207---> **r107**
r225 ---> **r21d**
r221--- > **r210**

Adding Additional BGM files

This edit will allow for the use of additional BGM files to be called on, and will be loaded into memory at startup. For a more detailed tutorial on it's use please see **THIS** section of the RE4UHD Sound Tutorial 2022.

HEX CODE TO ACTIVATE FUNCTIONALITY

AEV-ESE Toggle
7C BC 5F 5E 8B E5 5D C3
Change To:
7C BC E9 EA FA FF FF C3
---
83 C4 10 89 45 F8 33 C9 33 F6 B8 44 7E
Change To:
83 C4 10 E9 D8 01 00 00 90 90 B8 44 7E
---
57 F6 C1 01 0F 84 09 01 00 00 8B 04 9D 00 02
Change To:
57 F6 C1 01 E9 A4 01 00 00 90 8B 04 9D 00 02

3C FF 0F 84 EC 00 00 00 0F BE F8 8B 04 9D 00 02
Change To:
3C FF E9 6B 01 00 00 90 0F BE F8 8B 04 9D 00 02

Find: 0056DDB0
Paste:
0F 84 60 FF FF FF 83 FB 02 0F 8C 4E FE FF FF A1 00 0E 2E 10 8B 80 20 93 5C 00 E9 45 FE FF FF 74 80 83 FB 02 0F 8C 8B FE FF FF 0F BE F8 A1 00 0E 2E 10 8B 80 20 93 5C 00 E9 82 FE FF FF

Find: 005771A0
Paste:

89 45 F8 31 F6 8B 0D 00 0E 2E 10 8D 89 20 69 F3 FF 80 39 00 74 06 46 8D 49 40 EB F5 31 C9 A1 00 0E 2E 10 8D 80 64 0F 62 00 56 6B F6 5C 8D 04 30 5E E9 FA FD FF FF

Find: 00577168

Paste:

3B B6 3F E0 D4 FF 75 1C 50 53 8D 5E FC 8D 86 CC 35 F6 FF 80 38 00 74 0A 89 03 8D 40 20 8D 5B 04 EB F1 5B 58 83 3E 00 0F 85 A0 04 00 00 5F 5E 8B E5 5D C3

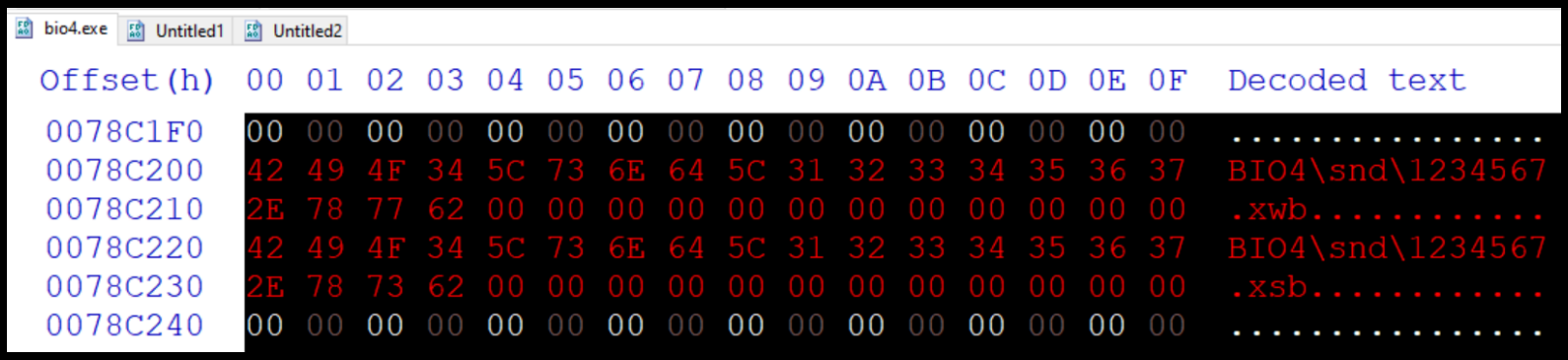
Find: **0078C200**

Start Entering String Directories at this address

For Example:

BIO4\snd\1234567.xwb

BIO4\snd\1234567.xsb



If we are to use more than one extra BGM file they will need to be separated by 20 bytes (hex) to separate each new set of XWB and XSB. The order **MUST** be xwb and xsb respectively. The maximum string length of each entry is 32(DEC)| or 20(HEX) bytes long.

Example Paste:

42 49 4F 34 5C 73 6E 64 5C 31 32 33 34 35 36 37 2E 78 77 62 00 00 00 00 00 00 00 00 00 00 00 00 42 49 4F 34 5C 73 6E 64 5C 31 32 33 34 35 36 37 2E 78 73 62 00 00 00 00 00 00 00 00 00 00 00 00

IN YOUR FSE ENTRY:

Directly at **Offset[88]:**

0 = bio4bgm

1 = bio4evt

2 = 1st new XSB/XWB

3 = 2nd new XSB/XWB

Example:

(this is an entry that plays a song from the first extra BGM file with):

Offset[80..83] = 0x**1D**0002

1D = song ID

2 = BGM file ID.

Doorway AEV's Can Spawn/Despawn Ashley

Use on Type 1 doorway events. Use Offset[52] = **0x7** for Despawn, **0x9** for Spawn

---

88 88 B2 4F 00 00 0F B6 56 6C A1 3C 5F
Change To:
E9 E2 02 00 00 90 0F B6 56 6C A1 3C 5F

Find: 002B74B8
Paste:
88 88 B2 4F 00 00 80 7E 34 07 75 0C 81 A0 28 50 00 00 FF FF FF FB EB 10 80 7E 34 09 75 0A 81 88 28 50 00 00 00 00 00 04 E9 F2 FC FF FF

---

r106 EM Sound Fix

Removes sound for spiders and bats that are hardcoded

E8 ED 77 B3 FF 6A 00 6A 00 6A 29 E8 E2 77 B3 FF 6A 00 6A 00 6A 2A E8 D7 77 B3 FF 6A 00 6A 00 6A 2E E8 CC 77 B3 FF
Change To:
90 90 90 90 90 6A 00 6A 00 6A 29 90 90 90 90 90 6A 00 6A 00 6A 2A 90 90 90 90 90 6A 00 6A 00 6A 2E 90 90 90 90 90
---
75 05 E8 0D EB FF FF E8 C8 FB FF FF 6A 01 8D 4D
Change To:
EB 0A E8 0D EB FF FF E8 C8 FB FF FF 6A 01 8D 4D

U3 No Transform and allows Death

This edit will prevent U3 from transforming into the part with the large claw. It will also add mortality to U3 as normally when we spawn U3 it will not be mortal.

---
80 BE 9B 0D 00 00 02 74 05 B9 01 00 00 00 51 53
Change To:
80 BE 9B 0D 00 00 02 EB 05 B9 01 00 00 00 51 53

(You're going to find 2 hits, **change both**)
75 0F C7 86 FC 00 00 00 01 00 00 00 E9 34 02 00
Change To:
EB 0F C7 86 FC 00 00 00 01 00 00 00 E9 34 02 00
---
56 E8 06 1E F0 FF 83 C4 34 6A 05 8B CE E8 2D C7
Change To:
56 90 90 90 90 90 83 C4 34 6A 05 8B CE E8 2D C7

8B CE E8 87 26 F0 FF D9 EE B9 50 CA
Change To:
8B CE 83 C4 04 90 90 D9 EE B9 50 CA

Disable Buy & Sell from Merchant

Disables Buying & Selling from Merchant

NOTE - This was not tested to see if we can still upgrade weapons with this modification so feel free to test it out.

FE C9 88 4B 11 EB 34 8B D0 81 E2 00 00 00 04 33 FF 0B D7 74 07 FE C1 88 4B 11

Change To:
B1 02 88 4B 11 EB 34 8B D0 81 E2 00 00 00 04 33 FF 0B D7 74 07 B1 02 88 4B 11

C6 46 11 01 8B C6 5E C3 CC CC CC CC CC CC CC CC
Change To:
C6 46 11 02 8B C6 5E C3 CC CC CC CC CC CC CC CC

Merchant Initialization (r100)

This edit solves the issue that makes us need to travel to r104 (the first encounter with the Merchant), in order to properly initialize our interactions with the Merchant. This edit does not provide the supplementary hardcoded edits that are required to make certain items available right away. More information can be found on the modding forums ~~HERE~~ but please do not ask me how to explain any of this).

8D 88 FE FE FF FF 83 F9 10 0F 87 FB 00 00 00 0F
Change To:
E9 5B FF FF FF 90 83 F9 10 0F 87 FB 00 00 00 0F

Find: 00313868
Paste:
3D 00 01 00 00 75 05 B8 04 01 00 00 8D 88 FE FE FF FF E9 8F 00 00 00

Always XL Attache Case

Even if users force a small attache case in the Ultimate Trainer it will still provide an XL sized case.



0F BE 8F AA 02 00 00 51 8B CB 89 9F AC 02 00 00
Change To:
B9 03 00 00 00 90 90 51 8B CB 89 9F AC 02 00 00

0F BE 86 AA 02 00 00 83 F8 03 77 50 FF 24 85 BC
Change To:
B8 03 00 00 00 90 90 83 F8 03 77 50 FF 24 85 BC

0F BE 96 AA 02 00 00 52 89 8E AC 02 00 00 E8 B9

Change To:
BA 03 00 00 00 90 90 52 89 8E AC 02 00 00 E8 B9

No Opening Cutscenes

Removes the car scene with Leon driving down the road.

C1 E9 05 83 C4 20 F6 C1 01 74 0C A1 3C 5F
Change To:
C1 E9 05 83 C4 20 F6 C1 01 90 90 A1 3C 5F

Keep Type 6 Lights During Cam events

(Prevents Type 6 lights from disappearing during Type 6 Cam events)

---
74 0D 85 C0 74 09 F7 40 04 00 08 00 00 74 25 8B
Change To:
EB 0D 85 C0 74 09 F7 40 04 00 08 00 00 74 25 8B

Ashley Elevator (r225, r21d)

Normally Ashley is not drawn by the game when she is following us and we ride this elevator. With this edit she will be seen riding the elevator with us. Do not ask about other elevators, as this edit only seems to work for these rooms.

E8 CC 54 B7 FF 8B 0E 83 C4 0C 8D 91 94 00 00 00
Change To:
E9 C3 01 00 00 8B 0E 83 C4 0C 8D 91 94 00 00 00

B9 FF FC 00 00 66 21 8E CE 02 00 00 81 4E 04 00
Change To:
B9 01 16 00 00 66 89 8E CE 02 00 00 81 4E 04 00

Find: 00492078
Paste:
E8 04 53 B7 FF 50 53 A1 00 0E 2E 10 66 8B 98 AC 4F 00 00 66 81 FB 25 02 74 0E 66 81 FB 1D 02 74 07 5B 58 E9 15 FE FF FF 8B 80 00 C9 FF FF 83 F8 00 74 EE 89 90 98 00 00 00 66 F7 80 CE 02 00 00 00 01 74 09 66 C7 80 CE 02 00 00 01 16 F7 40 04 00 08 00 00 75 CB 81 48 04 00 08 00 00 EB C2

Raz0r DLL Overlay Removal

This edit removes the silly splash page and coordinates overlay in game. This will also remove the room ID overlay when pressing F8



---

00 00 30 00 5E C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC 55 8B EC

Change To:

00 00 30 00 5E C3 8B EC C7 05 E9 88 24 10 90 90 90 90 C7 05 ED 88 24 10 90 8B C8 83 EB 03 55 EB E5

Ashley UpSkirt Reaction Removal

This edit removes Ashley's reaction 'You Pervert'



0F 84 3C 01 00 00 3C 02

Change To:

E9 3D 01 00 00 90 3C 02

Ashley No Scare

Removes Ashley's crippling reaction to enemies forcing her to the floor shaking in fear.

5F C7 86 FC 00 00 00 00 07 00 00 5E 5B 8B E5 5D

Change To:

5F 90 90 90 90 90 90 90 90 90 5E 5B 8B E5 5D

C7 86 FC 00 00 00 00 07 00 00 5E 5B 8B 4D FC 33

Change To:

90 90 90 90 90 90 90 90 90 90 5E 5B 8B 4D FC 33

Infinite Parasite Lifetime

Disables the timer on parasite death

7A 0F D9 86 90 07 00 00

Change To:

EB 0F D9 86 90 07 00 00

Crouch to Checkpoint - EDIT NOW DEPRECATED - PLEASE SEE NEW EDIT HERE

Sets AEV Type **0x0E** = Crouch to set a checkpoint.

~~66 C7 80 FD 00 00 00 11 00 88 88 FF 00 00 00 C3 CC CC CC CC CC CC~~

~~**Change To:**~~

~~66 C7 80 FD 00 00 00 11 00 88 88 FF 00 00 00 31 C0 E9 79 1C 04 00~~

~~53 53 53 88 45 FF 53 3C 01 75 10 6A 04 53 E8 42~~

~~**Change To:**~~

~~E9 70 E2 FB FF 90 53 3C 01 75 10 6A 04 53 E8 42~~

~~**Find:** 002800F0~~

~~**Paste:**~~

~~83 FB 00 75 0B 53 53 53 88 45 FF E9 81 1D 04 00 C3~~

No Melee for Ganados

Disables Leon's mêlée ability when interacting with Ganados

0F 84 33 02 00 00 D9 86 98

Change To:

E9 64 02 00 00 90 D9 86 98

Find: 00060800

Paste:

80 BE 02 01 00 00 22 74 05 E9 8F FD FF FF 3C 1A 0F 84 87 FD FF FF EB B8

No Leon Model In Inventory

Removes Leon's model from the Inventory screen

E8 C0 E8 C8 FF A0 84 74

Change To:

90 90 90 90 90 A0 84 74

E8 81 98 C7 FF 83 C4 10 8B

Change To:

90 90 90 90 90 83 C4 10 8B

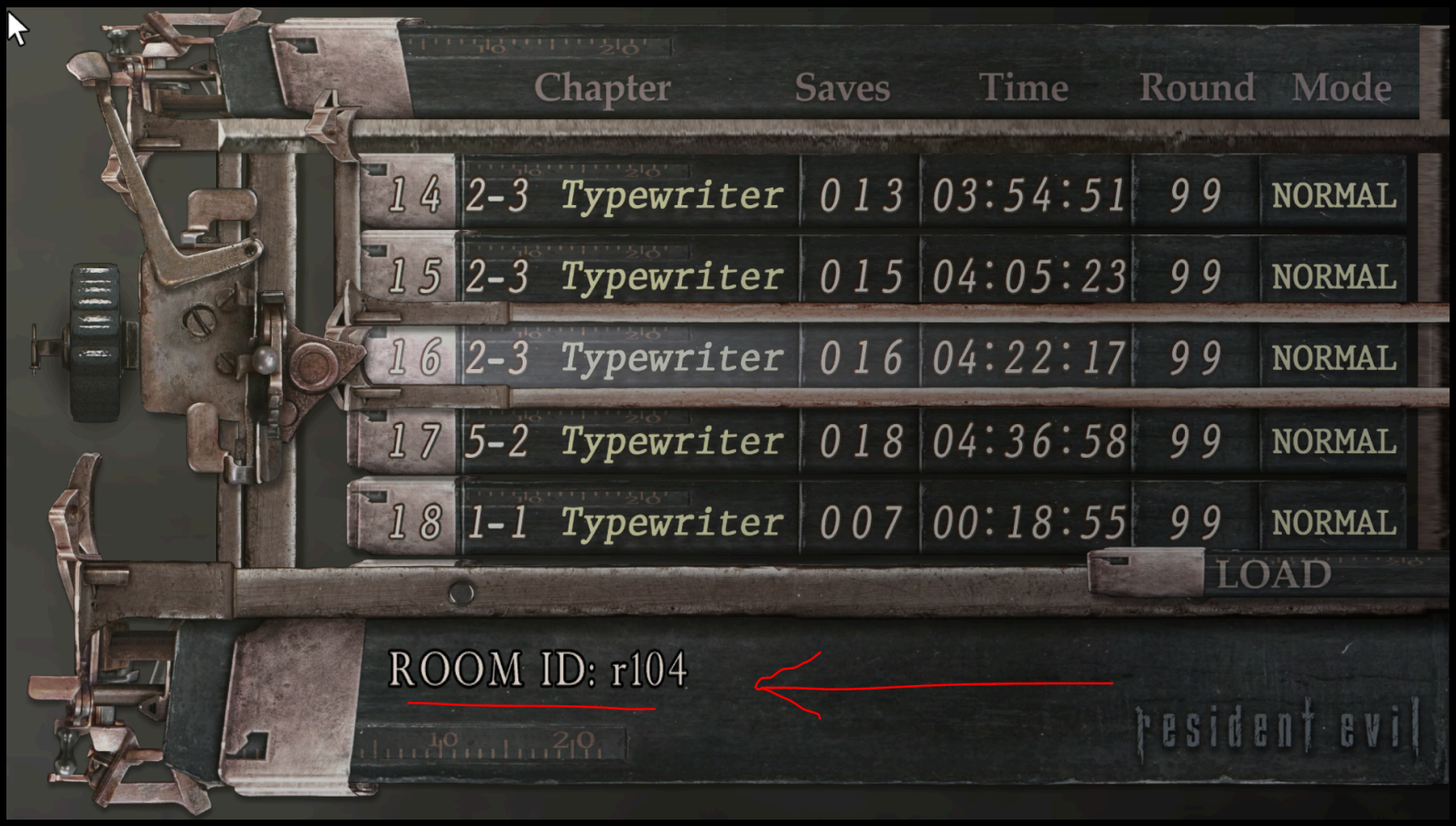
Transparency Pick Up Item fix

fixes transparency overlay when picking up items. This edit is already fixed with the RE4 Tweaks DLL
@ offset 007FF480

8B 50 58 8B 7D D0 C7 40 58 FF FF FF FF A1 3C 5F
Change To:
8B 50 58 8B 7D D0 90 90 90 90 90 90 90 A1 3C 5F

Show Room ID at Save & Load Screen

This edit will add extra data to be shown on the Save & Load Screen (Typewriter) which shows which room the save was created in.
Users must also use modified **memcard.udas** which can be downloaded **HERE**



0F B7 48 40 33 DB EB 35 56 6A 03 8B CF E8 D7 D6
Change To:
E9 2F 06 00 00 90 EB 35 56 6A 03 8B CF E8 D7 D6

8B 45 08 8D 14 40 8B 04 8D 6C 8E
Change To:
E9 47 02 00 00 90 8B 04 8D 6C 8E

8A 46 1D 84 C0 79 0D B8 13
Change To:
E9 C0 05 00 00 79 0D B8 13

Find: 0055B960

Paste:
0F B7 48 40 31 DB 50 53 51 56 66 8B 58 50 A1 00 0E 2E 10 8B 88 A4 7B 5C 00 8A 49 1D 3A 4D D8 75 27 BE 4F 00 00 00 8D 88 F0 56 D0 FF 46 66 3B 19 74 0C 83 C1 02 66 81 39 CC CC 74 0C EB EE 6A 00 6A 00 56 E8 2C 07 AB FF 5E 59 5B 58 E9 81 F9 FF FF

Find: 0055EB78
Paste:
8B 45 08 8D 14 40 83 F8 4F 0F 8C AB FD FF FF 8B 15 00 0E 2E 10 81 EA 20 E3 02 00 C7 02 87 00 0E 01 66 89 42 04 8B C2 31 D2 E9 93 FD FF FF

Find: 00561688
Paste:
66 9C 53 51 52 A1 00 0E 2E 10 2D C0 2E 04 00 8B C8 6A 00 E8 47 AB AA FF 5A 59 5B 66 9D 8A 46 1D 84 C0 E9 19 FA FF FF

Find: 0055B9D0 (This is a list of all the room numbers in ascending order)
Paste:
02 00 03 00 04 00 15 00 00 01 01 01 02 01 03 01 04 01 05 01 06 01 07 01 08 01 09 01 0A 01 0B 01 0C 01 0D 01 0E 01 0F 01 11 01 12 01 13 01 17 01 18 01 19 01 1A 01 1B 01 1C 01 1D 01 1E 01 1F 01 20 01 00 02 01 02 02 02 03 02 04 02 05 02 06 02 07 02 08 02 09 02 0A 02 0B 02 0C 02 0D 02 0E 02 0F 02 10 02 11 02 12 02 13 02 14 02 15 02 16 02 17 02 18 02 19 02 1A 02 1B 02 1D 02 20 02 21 02 22 02 23 02 24 02 25 02 26 02 27 02 28 02 29 02 2A 02 2B 02 2C 02 00 03 01 03 03 03 04 03 05 03 06 03 07 03 08 03 09 03 0A 03 0B 03 0C 03 0D 03 0E 03 0F 03 10 03 11 03 12 03 13 03 14 03 15 03 16 03 17 03 18 03 1A 03 1B 03 1C 03 1D 03 20 03 21 03 22 03 23 03 24 03 25 03 26 03 27 03 28 03 29 03 30 03 31 03 32 03 33 03 00 04 02 04 03 04 04 04 05 04 06 04 0A 04 0B 04 0C 04 0D 04 0E 04 0F 04 10 04 11 04 00 05 01 05 02 05 03 05 04 05 05 05 06 05 07 05 08 05 09 05 0A 05 0B 05 0C 05 0D 05 0E 05 0F 05 10 05 11 05 12 05 13 05 14 05 15 05 16 05 17 05 18 05 19 05 1A 05 1B 05 1C 05 1D 05 1E 05 1F 05 30 05 31 05 32 05 33 05 34 05 00 06 01 06 02 06 03 06 04 06 05 06 06 06 07 06 08 06 09 06 0A 06 0B 06 0C 06 0D 06 0E 06 0F 06 10 06 11 06 12 06 13 06 14 06 15 06 16 06 17 06

EM18 Invincible to Incendiary Grenades

Disabled the ability to kill em18 Merchant

66 89 8E 24 03 00 00 E8 4B 9D F8 FF 83 C4 04 C7 86 FC 00 00 00 03 00 00 00
Change To:
90 90 90 90 90 90 90 90 90 90 90 90 83 C4 04 C7 86 FC 00 00 00 01 00 00 00

Use Ashley Buttons in r20e

Allows for Leon to operate the buttons in r20e which is normally only accessible when playing Ashley

80 B8 C8 4F 00 00 01 0F 85 DA 00 00 00
Change To:
80 B8 C8 4F 00 00 01 90 90 90 90 90 90

Night Time Eyes Always On

BA 08 05 00 00 66 3B C2 1B C0 40 C3 33 C0 C3
Change To:
BA 08 05 00 00 66 3B C2 1B C0 40 C3 EB B0 C3

Leon Can Do Ashley Actions in r20d

Allows for Leon to operate the cranks, levers etc in r20d which is normally only accessible when playing Ashley. He can also throw the lanterns.

0F 85 82 01 00 00 8B 56 40 8B 8A 00 01 00 00 8B
Change To:
90 90 90 90 90 90 8B 56 40 8B 8A 00 01 00 00 8B

80 B8 C8 4F 00 00 01 0F 85 B5 00 00 00 0F B7 80
Change To:
80 B8 C8 4F 00 00 01 90 90 90 90 90 90 0F B7 80

---

r320 No Explosion Sound

Removes the explosion sound that occurs when you enter r320 for the first time.

6A 14 E8 F3 94 C1 FF 83 C4 18 6A 01
Change To:
6A 14 90 90 90 90 90 83 C4 18 6A 01

Luis Never Responds to Persistent Attacks Outside of r11c

(REQUIRES DINPUT8.DLL)

-When not in the cabin fight, Luis will become unresponsive after shooting him 5 times, this edit prevents this.
Note about vanilla behavior:
After 5 hits, an anti-shoot barrier will be formed around Luis which can help prevent the player from shooting Luis further.

FE 8E 7A 09 00 00 8A 86 7A 09 00 00 75 09 80 8E
Change To:
E9 F3 00 00 00 90 8A 86 7A 09 00 00 75 09 80 8E

Find: 004E6030

Paste:
80 BE 7A 09 00 00 01 74 08 FF 8E 7A 09 00 00 EB 21 50 A1 00 0E 2E 10 66 8B 80 AC 4F 00 00 66 3D 1C 01 74 07 C6 86 7A 09 00 00 05 FF 8E 7A 09 00 00 58 E9 D7 FE FF FF

AuraType of items when shot from a high place

Find: 002BD177

OG value is = 02

---

r102 - NoTeleport after Cutscene

E8 4F 92 B3 FF D9 EE D9 55 F0 8D 45 F0 D9 05 A0 D0 B2 00 50 D9 5D F4 8B CE D9 5D F8 E8 98 AA B3 FF
Change To:
83 C4 04 90 90 D9 EE D9 55 F0 8D 45 F0 D9 05 A0 D0 B2 00 50 D9 5D F4 8B CE D9 5D F8 83 C4 04 90 90

r102 - No Merchant Shop After Cutscene

E8 64 4E B4 FF 6A 01 6A 01 6A 01 6A
Change To:
90 90 90 90 90 6A 01 6A 01 6A 01 6A

---

r320 - No Teleport after Cutscene

E8 CE 73 C1 FF D9 EE 8B 0D D4 37 C5 00 D9 55 F0 8D 45 F0 D9 05 EC 85 B2 00 50 D9 5D F4 D9 5D F8 E8 13 8C C1 FF
Change To:
83 C4 04 90 90 D9 EE 8B 0D D4 37 C5 00 D9 55 F0 8D 45 F0 D9 05 EC 85 B2 00 50 D9 5D F4 D9 5D F8 83 C4 04 90 90

Ashley Jumps Alone (BROKEN FOR NOW)

We can not set Ashley to jump without the assistance of Leon having to catch her. By default, Ashley will always jump alone IF she is before a certain height. By adjusting the height threshold in the exe, we can set her to jump alone.

Ashley Fall Height The max distance Ashley can jump down before asking for help.

Find:: **00724E8C**
OG = 00 40 35 45 = 2900 (float)
MOD = 00 C0 0F 46 (9200)

Ashley Fall Down Animation Time

Seems to be how long Ashely is immune to collision during the falling animation. Without altering this (while also using the required animations described below), Ashley will fall through the floor. This next part of the edit seems to simply prevent this from happening. I am unsure of which values

Find: **0070C294**
OG = 00 00 20 42 = 40 (float)
MOD = 00 00 F0 41 = 30 (float) (feel gree to try other values. Setting this too high will result in Ashley going through the floor. A value of 30 seems to work well for both high and low elevations.

IMPORTANT NOTE

This mod requires the use of Ada's Jump down FCV, as the default for Ashley does not contain enough frames and will result in Ashley hanging mid-air when jumping from higher elevations. BiohazardX did create a new animation for Ashely that was longer, but it did not look as good as Ada's original.

Download Ada's FCV 'Jump Down' file **HERE.** Users will need to **copy** the contents of this FCV in a hex editor and **paste** it into **pl11_070.FCV** and repack the pl11 UDAS or:
simply rename the downloaded file to **pl11_070.FCV** and repack the pl11 UDAS

In the video seen on my Youtube Channel, we can see a height of 9200.
(Special Thanks to BiohazardX for all the animation research)!

Ashley Water Effect in any room

Presently even if we port over all the water splashing effects from another room to a custom room, Ashley will not make any ripples or splashes in the water. This edit simply removes that check from the instructions.

Users may need to port the ‘EFFECT 0’ folder from r10a EFF file to whatever other room they are working on. If there is a conflict with the need to preserve the original EFFECT 0 group folders, then it might still be possible. The water splashing effects are anywhere from Group 0 - 5 (there might be more of I don’t know but things seem ok with just these 5 folders).

---

66 3B C1 74 0E BA 1A 01 00 00 66 3B C2 0F

Change To:

66 3B C1 EB 0E BA 1A 01 00 00 66 3B C2 0F

OUTRO

This concludes this RE4UHD - BIO4 Modifications Guide. While there are certainly more bio.exe edits out there that other users have documented exclusively, my goal was not to make a comprehensive list of all edits I know that exist. This guide was created to share new edits and how to use them in game. My deepest thanks to AnonymousUser for all the help with these amazing new features.

Happy Modding!

Mr.Curious
4-2022

Back to Top



Mr. Curious
@MrCuriousModding
2.61K subscribers

HOME VIDEOS PLAYLISTS COMMUNITY CHANNELS ABOUT

Users can check out other video tutorials on my YouTube Channel **HERE**
