Loading Wii Discs on ISO Loaders for Speedrunning

Guidelines on Ensuring Fair Load Times

Summary:

Presented and written by the Trauma Center speedrunning community:

- Thurler
- LeoKeidran

Special thanks to:

- Joselle for verifying load differences between loaders for Super Mario Galaxy
- Irisjoker and Cosimo for submitting video for loader and hardware comparisons
- XFlak for providing a plethora of information regarding the problem, and helping us draw the conclusion
- Other GBATemp users that provided useful information

One of the few things every speedrunning community will agree on is that load time differences between platforms and hardware revisions is a massive headache to deal with, when trying to ensure fair and balanced competition among players. In the Wii's case, there are at least 4 disc drive revisions, all of which can have an effect on a game's load times - since the bottleneck is, by far, the speed with which you read data from the disc.

An extreme aggravant in the Wii's case is the fact that it is region locked, and more often than not, players find themselves with a Japanese copy of their game and a North American / European console. Enter softmodding and homebrew, to try and circumvent the region lock in the system. The most common way to do it is to use what is known as an ISO loader - something that will jump execution from the system menu (or the homebrew app menu) to the game's boot sequence.

Since this is a custom app loading into the disc, one is left to wonder how (or if) this could affect the load times at all. It seems like such a trivial matter, but as it turns out, there is a huge detail that is central to ensuring fair load times when using these loaders that often gets overlooked.

In this document, we aim to cover some questions that might arise when pondering over this subject, and provide explanations for what is going on when using one of these loaders, as well as explain why they affect speedrunning at all. We will also provide a guide for setting up your homebrew environment in a way that ensures you can run Wii games fairly, as if you had a stock Wii using stock firmware to run stock games.

We will also go over some additional details to cover some optional video settings for specific Wii games that do not support progressive scan (480p), so that you can increase your video quality without compromising fairness in load times.

IMPORTANT: We only cover the cases where you are loading games from the Disc drive, rather than loading them from the USB slot. Since you can hook up any kind of hard drive to the Wii through USB, it's trivial to make a setup where you get instant loading in games by using an SSD or a high end HDD. Thus, loading games from USB should NOT be allowed to ensure fairness when playing. Disconnecting your USB drive is recommended to avoid confusion when loading games.

Section 1: Understanding the Problem

How does the Wii load games?

Keep in mind this will be a very simplified explanation, and more details can be found in the links provided in Section 6. I'm by no means an expert on the topic. That said, the Wii has 256 slots to store revisions of the "operating system" that it runs, and any game can request to run on any of those revisions, by specifying the slot it wants to load from. We will be calling each revision an IOS, and each slot an IOS slot, since it, well, stores an IOS. So when talking about IOS X, we mean whatever Nintendo (or your homebrew apps) stored in IOS slot X.

Most launch titles will request IOS 9, which contains the things Nintendo packed up with the Wii on launch back when the system menu was still in version 1.0. Later games required some software updates that brought in new features for the Wii's system, with new functionality and patches that affected both games and apps. Super Mario Galaxy, for example, requests IOS 33, since it was built to run in that revision of the system's software.

You might wonder why not just make every game load in whatever the most recent IOS is, but this presents a problem with backwards compatibility - can you be 100% sure that your future updates won't break something that old games relied on? Some obscure functionality that you may not be accounting for? By isolating the IOS slots and ensuring games always run on that slot, you can make sure they will run fine no matter what the future holds for the system.

That said, specific IOS can have updates within itself. IOS 9 for example has 10 official revisions, each building more functionality into that slot, sometimes backporting things that could be useful for older software, sometimes just fixing bugs and exploits.

It's important to note that the Wii can only have ONE active IOS and ONE active game/app at a time, so when you booted your launch day European Wii into the system menu, it was running IOS 10 and the system menu v1.0E. When you loaded a launch title, it switched to IOS 9 and whatever game you loaded. This is true even today, with homebrew apps and games loaded from Disc or USB.

How does homebrew fit into all of this?

The reality of the fact is that the single most important homebrew application is piracy. It's what the masses want, and plenty of homebrew developers will work tirelessly to ensure you can play as many games as possible without a legit copy of the game. We're not going to get into the morality of this, but anyway.

Piracy relies on loading a copy of a game from something that isn't the original disc, be it a writable DVD or a USB drive. Neither of those are supported by the Wii natively, so developers need to go around the system checks to forcefully load the game. This isn't as simple as just redirecting execution, so various patches need to be applied to the system files.

However, let's say someone messes up and patches faulty software into Nintendo's official IOS slots, and the system just stops working altogether, bricking your Wii. This is obviously a risk that everyone wants to avoid, so we need some way to isolate our custom code from Nintendo's official one.

Enter the arbitrarily high IOS slots, from slot 200 onwards. These were never used officially by Nintendo, except when they attempted to patch out homebrew software by overwriting those slots with useless code. The cat-and-mouse game is long over though, and most homebrew apps nowadays will run on IOS 249, which is the slot most people will install custom software to.

Commonly abbreviated as cIOS (custom IOS), this is a collection of patches from various developers that adds functionalities to go around system limitations (like region locks) and enhance already present functions (like add in support for USB loading). Since there's no need to reinvent the wheel, these are applied on top of an existing IOS, copied over into slot 249. This is referred to as the base IOS for the cIOS, so you can say your IOS 249 is a cIOS based on IOS 57, for example.

So what is the problem exactly with homebrew?

Since homebrew apps are designed to load backup of games from possibly non-standard mediums, it will require a cIOS to be present, based on a very recent IOS to maximise compatibility with games. This is another important point - most games will work just fine with a later IOS from the intended one they came out with. Some will stop working because of a small detail that the cIOS can patch in, others for more complex reasons that will require a second cIOS to be present, based on a different IOS.

The most common way to install homebrew nowadays is to have 3 cIOS installed in IOS 249, 250 and 251, to ensure compatibility with pretty much every Wii game out there. These are based on IOS 57, 56 and 38 respectively, as seen on the most popular homebrew guide out there today. This means that, by default, games will be using IOS 249 (based on 57) when loaded through homebrew, which is a much, much different IOS compared to the intended one (IOS 9 for launch titles, for example).

By itself, this wouldn't be a problem. However, as one might intuitively think, Nintendo optimised the way data is read from the disc as time went on, and these new changes were applied in newer IOS. Newer, bigger games could enjoy better load times as the amount of data that needed to be read from disc became more and more ambitious.

So what happens when you're now loading a game made for IOS 9 into a cIOS based on IOS 57? All of these optimisations get applied and the load times become faster than originally intended. This is great if you're playing casually, but it presents a problem for speedrunning, since it breaks fairness to someone playing with no homebrew.

And how can we fix it?

The problem stems not with homebrew itself, but the fact that we are loading the games with an IOS that is not the one intended for them, which in turn causes load times to act abnormal. Depending on the game, some other in-game behaviour could be abnormal as well, but as far as I know, no games have a noticeable difference.

Using an ISO Loader that actually loads the correct IOS for the game, rather than just default to IOS 249 can fix the issue immediately. One such loader is Gecko OS, that just loads the game like the disc channel would, just going around the region lock and any mandatory updates present in the disc.

ISO Loaders that default to IOS 249 can still be used, so long as they provide the means to override that behaviour and load the correct IOS. Neogamma has a built-in function to load the correct IOS for the game, and USB Loader GX allows one to load the game from arbitrary IOS slots, as long as it's a high slot (200+).

We will now go over the particularities of each loader, and how to ensure fairness with whatever setup you're running. For the sake of completion, we will also briefly go over the steps to install homebrew in your console, and help you understand what each step is doing to ensure you know your way around.

Section 2: Setting up Homebrew

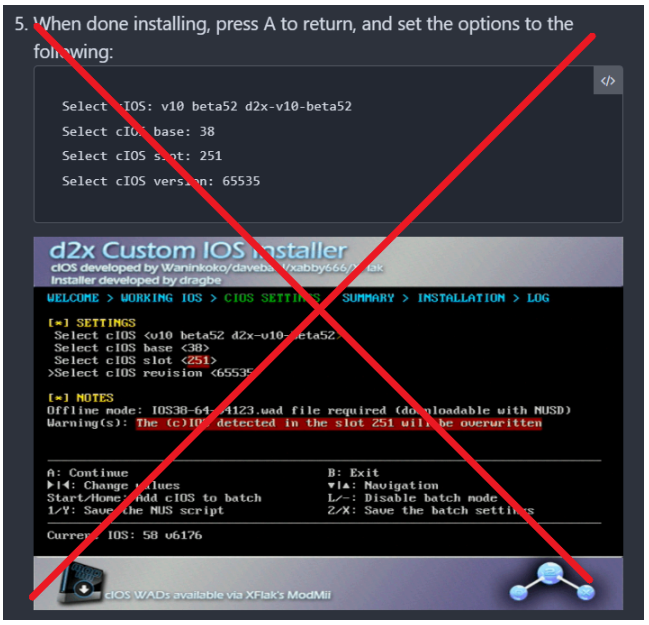
IMPORTANT: The steps below assume you have a regular Wii. If you are using a WiiU (vWii) or a Wii Mini, please refer to the internet for instructions on how to install homebrew, and how to organise your IOS slots / install cIOS.

To start off, you'll need to run an exploit on your Wii to load yourself into HackMii, a simple app that will install the basics of homebrew into your console. Namely, the Homebrew Channel (to load any custom app) and BootMii. Head over to https://wii.guide/get-started and find the best exploit for your setup.

Once you get into HackMii, you'll want to follow the steps over at https://wii.guide/hbc to install the Homebrew Channel and follow some other brick safety steps. This would install BootMii in IOS 254.

Keep in mind that installing cIOS is entirely optional when it comes to fairness in speedrunning. If you're only installing homebrew to circumvent region locking and video patching, this is completely optional. If you do get around to installing cIOS, follow https://wii.guide/cios, keeping note of which slot you're installing each revision to. The default installation goes as follows:

- IOS 249 becomes a patched IOS 57
- IOS 250 becomes a patched IOS 56
- IOS 251 becomes a patched IOS 38



Personally, I would forgo patching IOS 38 on IOS 251, since it only adds compatibility to very few games. Maybe look up if the games you want to play require that specific IOS to be installed. Either way, you'll want to know which of your IOS slots are vacant, and which aren't.

Figuring out what IOS a game runs on

If this information isn't quickly available through google search, or if you're uber sceptical, you can check it with Neogamma (download link in Section 4), as it will display this information when you are booting the game with its default settings. Simply launch Neogamma and hit "Load game from DVD", then stay on the lookout for this message:

In the above example, we are loading a Japanese copy of Trauma Center: Second Opinion, a launch title that uses IOS 9.

Copying stock IOS into arbitrary slots

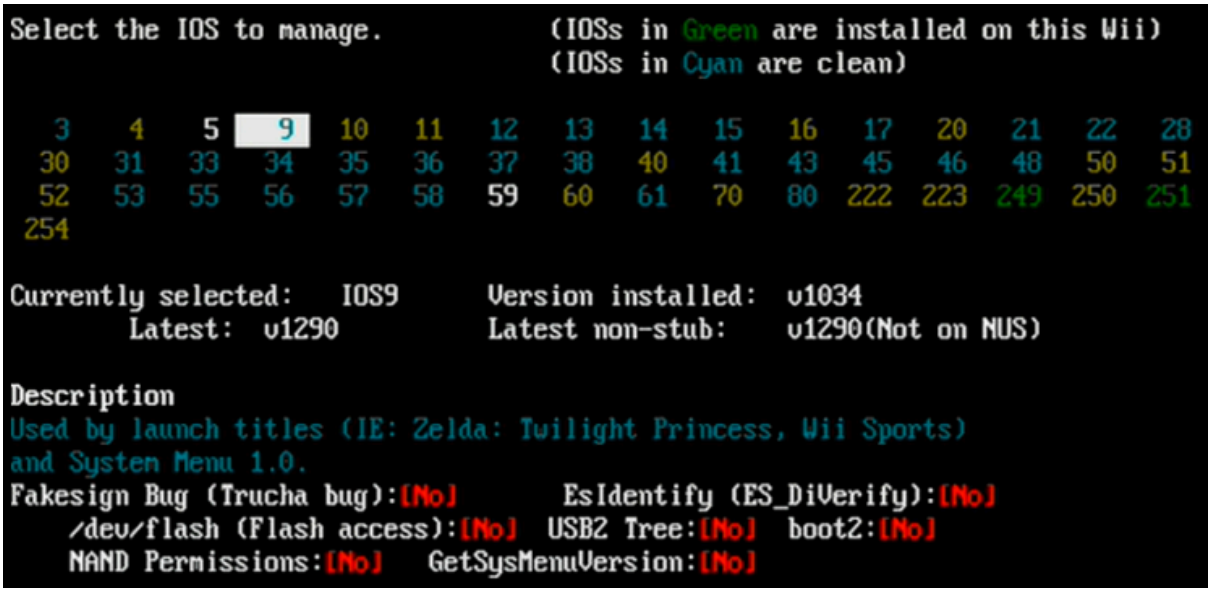
Some ISO loaders allow you to load games using arbitrary IOS slots, and specifically for USB Loader GX, it limits your choices to IOS 200 onwards. Why it does this is unknown to me, since if the user knows what they're doing they would know what to enter as the desired IOS. Regardless, this creates a need to copy over some very old IOS into arbitrarily high slots so we can use them, since we can't just patch cIOS on whatever IOS we desire.

We'll be taking Trauma Center: Second Opinion as our example here, it uses IOS 9. Refer to the previous subsection to figure out what IOS your game uses, and what you should be copying over. In order to do this copying, we'll be using Wii Mod Lite to install arbitrary things in arbitrary places, as it provides an interface to manage your IOS slots. A download link can be found below:

https://github.com/RiiConnect24/Wii-Mod-Lite/releases/download/v1.7/WiiModLite_v1.7.zip

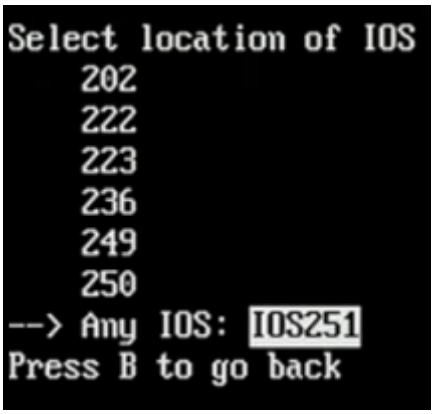
***IMPORTANT*: If you mess around in the following menus without knowing what you're doing, there is a chance of bricking your Wii. Please be very careful when installing arbitrary IOS. Remember this is fully optional, and only required for fairness with USB Loader GX - other loaders can load the correct IOS just fine.**

Once Wii Mod Lite loads, enter the IOSs menu, you'll be greeted by a grid of installed IOSs on your Wii. If your Wii is fully updated, you can navigate to whatever IOS your game uses and check the installed version, and some miscellaneous info about the IOS. Below is the example for IOS 9 on a 4.3U Wii:

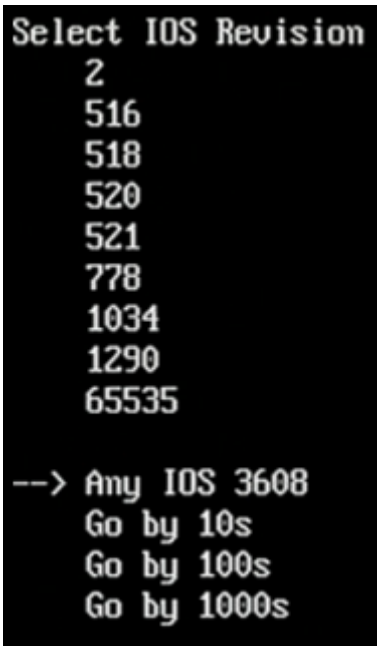


Hit A on the IOS for the game you want to load, you'll be brought to a menu for actions regarding that IOS. Head over to the Copy option to, well, copy its contents elsewhere:

It will then ask where you want to copy it to. The options it gives you are the most common targets, all of which are valid **so long as you don't have other homebrew installed there**. In order to select an arbitrary slot, head over to the "Any" option and hit left/right repeatedly until you have the right IOS slot highlighted. In this example, we're copying over IOS 9 to IOS slot 251. **BE VERY CAREFUL WHEN SELECTING ARBITRARY IOS SLOTS:**

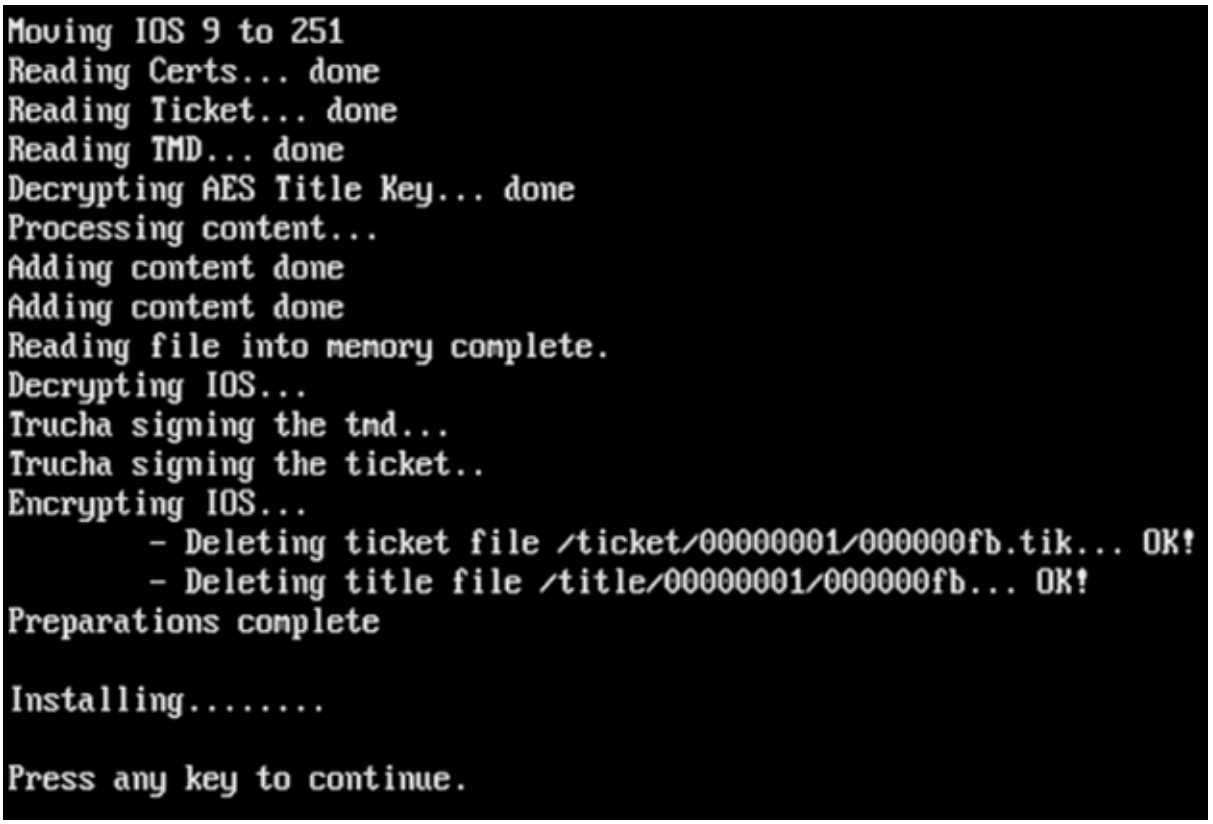


Next it will ask what revision you would like to install it as. The default is to use the same revision as the one you're copying, but you can use any number you want. Choosing 65535 will ensure it will override whatever is already present in that IOS slot, if you know what you're doing and overwriting things:



Finally, it will ask you what patches you want to apply to the IOS. Leaving it on all defaults is fine. If you know what you're doing or need a specific patch for your scenario to work, feel free to change these:

Now just wait for the system to patch the chosen IOS slot, shouldn't take too long. After this is done, you have successfully copied over the stock IOS into a higher slot visible to most homebrew apps:



Section 3: Disc Channel and Gecko OS

Loading games through the Disc Channel is the ground truth to all load times measurements for a given console, assuming of course nothing is modifying its behaviour, such as some Priiloader settings. It has no additional configurations to be wary of, just boot the game and go. It will not go around region locking, and for games that do not support progressive scan, it will force interlaced scan. If you own a Wii of the same region as the disc (such as a Japanese Wii for a Japanese game), there's usually no downside to using the disc channel.

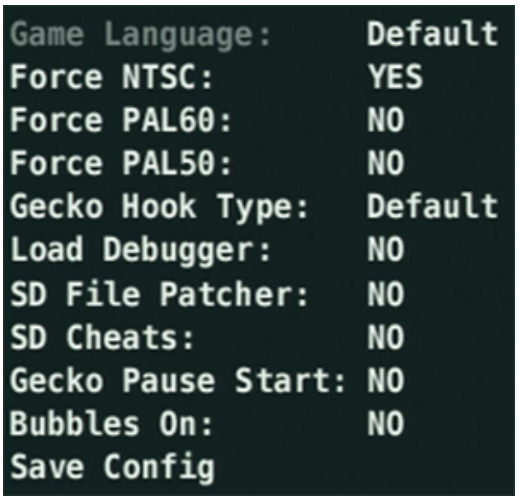
Video Patching?	Region Unlock?
No	No

Gecko OS is a very simple ISO loader that just acts like a Disc Channel with no region lock and a few video options to help some specific cases out. A link to its page can be found below, and there is a link to download it.

Download: https://wiibrew.org/wiki/Gecko_OS

Gecko OS is a safe and easy choice for most games, since there are no special concerns with IOS, it will always load the game's correct IOS, rather than rely on homebrew IOS. The video patching is kinda limited, as seen below you can force some settings, but far from ideal for games like Trauma Center: Second Opinion that do not support progressive scan:

Video Patching?	Region Unlock?
Kinda	Yes



Section 4: Neogamma

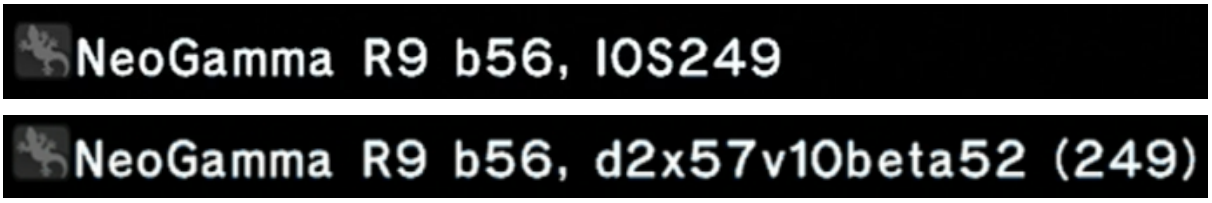
Neogamma will load games using IOS 249 as default. It will make no attempt to search for other available IOS in slots 200+, and cannot override the custom slot it uses to load games with. It can, however, force itself to load the correct IOS for each game, which is an amazing behaviour for speedrunning. It also provides a more complete video patching tool, allowing any game to use progressive scan.

Video Patching?	Region Unlock?
Yes	Yes

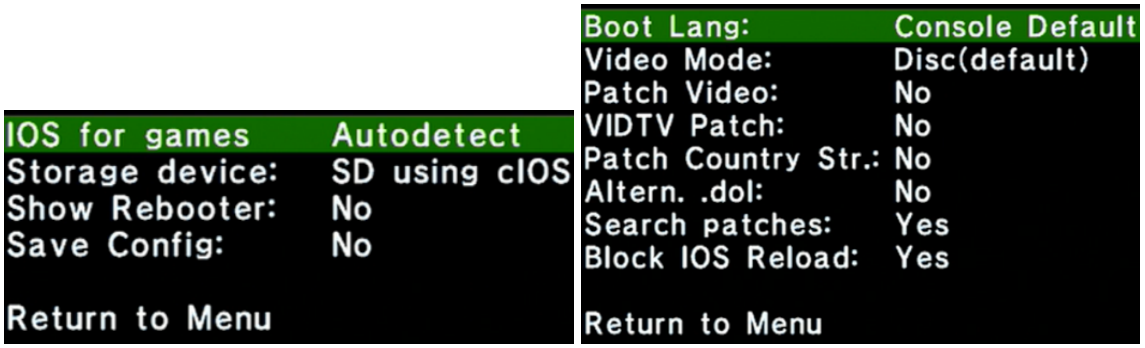
Download: https://gbatemp.net/download/neogamma.27066/download

One note regarding IOS 249 is that if there are no cIOS installed there, it will detect that it is a stub with no functionality, and default to using the correct IOS for the game. If you don't have a cIOS installed in slot 249, Neogamma is as reliable as Gecko OS, booting games fairly without the need to configure anything.

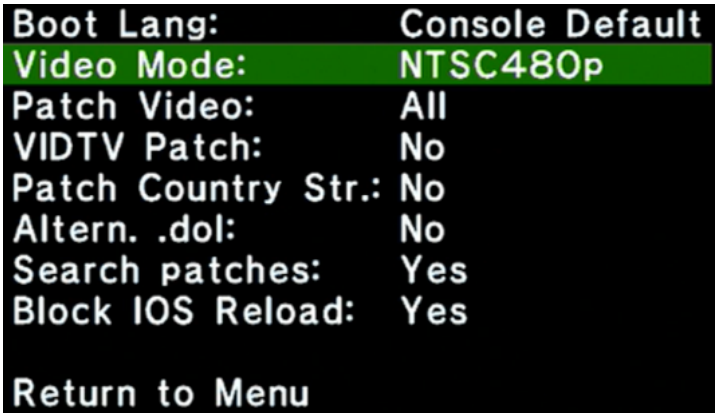
Neogamma will also tell you if it has detected any cIOS in IOS 249, note the difference in the screenshots below. This can be very useful when verifying that someone is running on a stub IOS 249, but keep in mind that this information can be easily tampered with, and should not be used as definitive evidence that someone's IOS 249 is a clean stub:



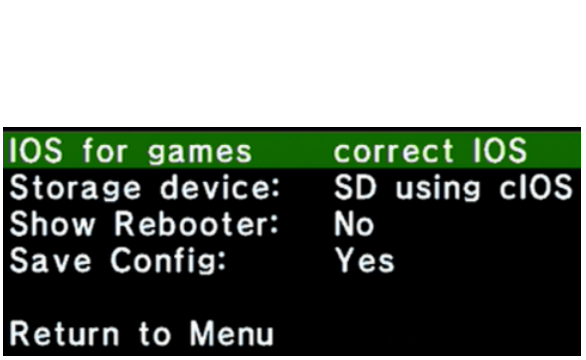
As mentioned, if running on a clean IOS 249, Neogamma already defaults to the correct IOS, so the General and Wii Options would default to something like this:

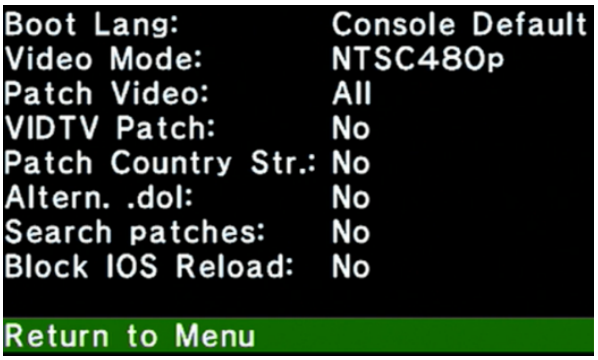


If running a game without progressive scan support, or if messing around with different video modes to better adapt to your situation, you can patch the video mode as such in the Wii options. These settings have been tested and do NOT impact load times in any meaningful way. In the example, we are patching a game to use NTSC 480p:



If you do have a cIOS installed in slot 249, you will need to tell Neogamma to use the correct IOS rather than its default one. This can be accomplished with the following settings on the General and Wii tabs. In the example we also patch video to NTSC 480p, and save config so we don't have to remember to change these settings every time we boot Neogamma:





The option "Block IOS Reload" is the one that allows Neogamma to change from IOS 249 to whatever the correct IOS is. Technically leaving the option "IOS for games" in "Autodetect" would work, but it's better to be safe than sorry.

Section 5: USB Loader GX

USB Loader GX, as the name implies, is primarily used to load games from USB storage. It features the most options out of all loaders mentioned in this document, and is built to ensure compatibility with whatever Wii is thrown in its way. Because of this, it's the one that takes the most setup to ensure fairness in, starting with the fact that we'll be using it to load a disc rather than USB.

Download: https://sourceforge.net/projects/usbloadergx/files/latest/download

For starters, make sure you have a copy of your game's IOS installed in a high slot (anything above 200). In the examples below, we'll assume a copy of IOS 9 is installed in IOS 251 in order to launch Trauma Center: Second Opinion. If you don't have this setup, refer to Section 2, specifically the Copying stock IOS into arbitrary slots section.

A warning for moderators: it is not possible to enforce this custom installation process, as there is no way to guarantee the correct IOS is installed in whatever arbitrary slot the player is using. Running a SysCheck will not show what is installed in such a slot, and even if it did, that information is easily spoofed by someone that knows what they're doing.

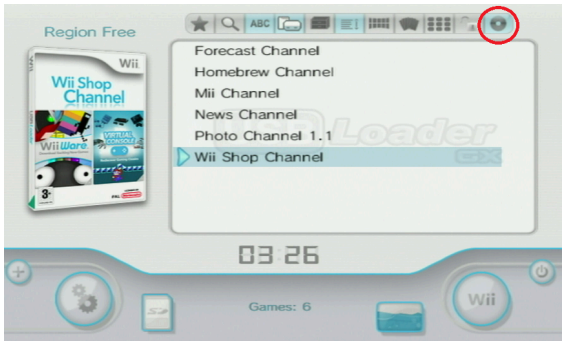
Video Patching?	Region Unlock?
Yes+	Yes

USB Loader GX offers the most options for video patching and region freeing, on top of offering plenty of quality of life features for casual play, making it a stellar option for those that do more than just speedrun on their Wii. As mentioned though, there is some considerable work to make it have fair load times, that might be too much trouble for some users.

Another important note for USB Loader GX: it specifically tries to load whatever is in IOS 249 on boot, and if it fails to find anything, it will scan for any valid IOS in slots 200-255. If it still fails to find anything, it will automatically load IOS 58 with a few custom patches into memory, so that even a Wii with no cIOS installed can run games from USB. Keep in mind that unlike Neogamma, it will always default to using IOS 58 for all games in this scenario.

Yet another important side effect to having no IOS installed on slot 249: if you copied over the system's stock IOS (like 9 or 33) into some high slot to load a game with fair load times, but not have anything in slot 249, USB Loader GX will try to load itself using the arbitrary slot you installed the stock IOS in, and most likely crash itself. So you kinda need to have a cIOS in slot 249 to make USB Loader GX behave properly for this setup.

Once everything is setup, open USB Loader GX and click the disc icon to mount the disc drive, then wait for the game to be recognized. Once loaded, head over to "Settings" and into "Game Load". You'll want to change the setting "Game IOS" to 251 (or whatever slot you used) as shown:



Finally, if you want to apply any video patching for games that don't natively support progressive scan, or just changing it to better fit your setup, you can do it in these settings as well:

Section 6: References and Further Reading

- WiiBrew page on IOS: https://wiibrew.org/wiki/IOS
- WiiBrew page on /dev/di, the Wii's disc driver: https://wiibrew.org/wiki//dev/di
- WiiBrew page on a specific IOS, for referencing revisions and other data: https://wiibrew.org/wiki/IOS9
- GBATemp thread where these issues were originally discussed: https://gbatemp.net/threads/d2x-cios-possibly-causing-faster-disc-load-times.606642/
- Homebrew guides with links to other documentation and support: https://wii.guide/get-started
- Joselle's spreadsheet containing timings for Super Mario Galaxy using different loaders: https://docs.google.com/spreadsheets/d/178cRnPLYlJc5nNz8FpLbkVuMuIQFC-U2RGZ665Lk5WY
- Trauma Center speedrunning discord server, where all this discussion started: https://discord.gg/h4qwmkY