

GB ENHANCED+ USER MANUAL



Version 1.3
D.S. Baxter
April 1st, 2019

1. Foreword	4
2. Getting Started	5
2.1 :: Obtaining, Compiling, & Installing GBE+	6
2.2 :: Command-Line Options	7
3. Configuration	9
3.1 :: General Settings	9
3.2 :: Display Settings	15
3.3 :: Sound Settings	17
3.4 :: Controls Settings	18
3.5 :: Netplay Settings	22
3.6 :: Paths Settings	24
3.7 :: The .ini File	26
3.8 :: The Cheat File	27
4. Custom Graphics (CGFX)	28
4.1 :: Overview	28
4.2 :: Dumping Graphics	29
4.3 :: Layers - BG Meta Tile	30
4.4 :: OBJ Meta Tile	32
4.5 :: Manifest Tab	33
4.6 :: Configuration Tab	34
4.7 :: Using the Advanced Menu	35
4.8 :: The Manifest File	36
4.9 :: Editing and Loading Graphics	39
5. Debugging	40
5.1 :: Memory Mapped I/O Registers	40
5.2 :: Palettes	41
5.3 :: Memory Viewer	42
5.4 :: Disassembly	43
5.5 :: OBJ Tiles	45
5.6 :: BG Tiles	46
5.7 :: The Command-Line Debugger	47
6. Netplay	53
6.1 :: Netplay Guide	53
6.2 :: DMG-07 Setup	54
6.3 :: Net Gate	55

7. Game Specific Instructions	56
7.1 :: Barcode Boy Games	56
7.2 :: Barcode Taisen Bardigun	56
7.3 :: Boktai Games	56
7.4 :: Battle Chip Gate Games	56
7.5 :: Bomberman Max Red/Blue, Mission Impossible	57
7.6 :: Chee Chai Alien	57
7.7 :: Drill Dozer	57
7.8 :: GBC Infrared Games	57
7.9 :: Gyogun Tanchiki: Pocket Sonar	57
7.10 :: Kirby Tilt 'n' Tumble	58
7.11 :: Legendz: Isle of Trial/Sign of Nekrom	58
7.12 :: Mobile Adapter GB Games	58
7.13 :: Pokemon Gold, Silver, and Crystal	58
7.14 :: Pokemon Ruby, Sapphire, and Emerald	58
7.15 :: Sakura Taisen GB	59
7.16 :: WarioWare: Twisted	59
7.17 :: Yoshi Topsy Turvy/Yoshi's Universal Gravitation	59
7.18 :: Zok Zok Heroes	59
8. FAQ	60
8.1 General FAQ	60
8.2 GBE+ Roadmap	62
9. Credits & Acknowledgments	63

1. Foreword

This manual will attempt to cover all the functions users may encounter while operating GB Enhanced+. The information contained herein doubles as both an operating guide and reference to the emulator's various features.

GB Enhanced+ is the successor to the original GB Enhanced project (note the shiny "+"). It is a Game Boy, Game Boy Color, Game Boy Advance, and NDS emulator that aims to provide as many enhancements as reasonably possible. Although there is much work to be done in later releases, such as GBA netplay and e-Reader emulation, GBE+ is dedicated to pursuing these types of things.

Perhaps most importantly, GBE+ supports an emerging emulation technique for 2D games, what this project refers to as **Custom Graphics**, or **CGFX** for short. Like HD textures for N64 games, CGFX lets users replace in-game graphics with their own pixels, from simple recolors to full-blown high-definition versions. While replacing graphics for 2D systems is not common among other emulators, GBE+ is proud to push the boundaries in this field of emulation.

Additionally, GBE+ aims to be at the forefront of video game preservation by emulating a vast array of the Game Boy's many accessories. From the widely-known Game Boy Camera and Printer, to the exotic and obscure Barcode Boy, one of GBE+'s primary goals is to recreate all of these devices through software. History deserves to be documented and remembered, and to that end, GBE+ plays all sorts of games that used specialty hardware or weird features. With any luck, eventually every crazy add-on imaginable will added to the project.

What started as an idle dream to make my own NDS emulator has become something much more than the humble project I founded in 2012. There is still a lot more to go through before I can see my vision completed, however, I believe this project is on the right path. In the coming years, we'll see where the road takes this emulator. In the meantime, I hope this manual will give users an insight to how the programs works, what it's capable of, and where it's going.

D.S. Baxter - aka Shonumi

2. Getting Started

Getting started with GBE+ is relatively simple. The emulator does not have many requirements to build from source, and installation should be simple for most operating systems. Please consider, however, that GBE+ has not been tested on OS X/macOS in any way, shape, or form. This is due to the lack of access to the operating system. Users can still build it themselves on OS X/macOS.

Currently, GBE+ has minimal hardware requirements. Any recent computer should be able to run the emulator just fine. Certain tasks, such as processing large amounts of CGFX, require more single-threaded processing power. Under some circumstances CGFX may also benefit from more RAM. Generally, however, these scenarios are reserved for intense use of HD graphics. Otherwise GBE+ is not a demanding emulator.

It should be noted, however, that as of 1.3, GBA games in general may eat up a lot of CPU resources. This is due to inefficiencies in the GBA core that will be addressed in the very near future. Most mid-range computers and above won't notice anything performance-wise, but weaker computers, particularly those with Ultra-Low Voltage CPUs, may experience slowdowns with the GBA core.

GBE+ aims to build and run with minimal software dependencies. The recommended minimum version of OpenGL is 3.3. Any computer released in the past decade should support this without any trouble. Future versions of GBE+ may add support for Vulkan.

Currently, GBE+ supports both 32-bit and 64-bit systems. For the foreseeable future, this will remain the case. For CPU emulation, any dynamic recompilers added to later releases, will only target x64 systems. Nevertheless, GBE+ will technically continue to support 32-bit systems at that time through CPU interpreters.

For a general roadmap of where GBE+ will go from here, please see the FAQ in **Section 8** for more details.

2.1 Obtaining, Compiling, & Installing GBE+

For Windows users who do not want to build from source code, please visit the project's GitHub page and check out the 1.2 release on the Release page. Download the zip file and extract it. Simply double-click the executable file *gbe_plus_qt.exe* to run the GUI version of the emulator. For those interested in the command-line version of GBE+, run the *gbe_plus.exe* file from the command prompt. The majority of this manual focuses on the Qt version of GBE+, however, please refer to **Section 2.2** for more details about running the command-line version.

Linux users have to compile the source code themselves. The build process on Windows is virtually unchanged when using the MSYS2 toolchain and msysgit. Compiling from source requires prior installation of the following programs and libraries:

- GIT
- CMake
- SDL 2.0
- SDL_net 2.0 (optional, for netplay support)
- OpenGL
- Qt4 or Qt5 (optional, for the GUI)
- GLEW (Windows only)

CMake will check to make sure it can find all of the necessary dependencies before the build process begins. Before that happens, however, GIT must retrieve the source, or the source tarball from the Release page must be downloaded and extracted. The following terminal instructions detail how to download the source code through GIT, compile it, and install the emulator:

```
git clone https://github.com/shonumi/gbe-plus.git
cd gbe-plus
mkdir build && cd build
cmake ..
make && make install
```

Note that this will install the very latest source code. GIT can check out specific revisions based on the hash of that commit. Consult the GIT documentation for checking out revisions and the GBE+ GitHub repository for the appropriate hash. Once CMake installs GBE+, the emulator can be called via *gbe_plus* for the command-line version, or *gbe_plus_qt* for the Qt version.

2.2 Command-Line Options

The command-line version of GBE+ accepts several parameters. Below are all the valid arguments for the emulator along with a short description of what they do:

-b or --bios [FILE]:

This instructs GBE+ to boot a system's BIOS or Boot ROM with the provided file when loading a game. The second argument is the exact path for the BIOS or Boot ROM on the user's computer.

-d or --debug:

This starts GBE+ in debug mode. It will pull up the command-line debugger. See **Section 5.7** for details on how to use properly use this version of the debugger.

--opengl:

This forces GBE+ to use OpenGL for all drawing/blitting operations instead of SDL.

--2x, --3x, --4x, --5x, --6x:

Scales the screen by a given factor. Only applicable when OpenGL is enabled.

--sys-auto:

Sets the emulated system type to AUTO. In this mode, GBE+ will automatically determine what system to emulate based on the game.

--sys-dmg:

Sets the emulated system type to DMG (old black and white Gameboy). This option is not valid when running GBA games and is ignored by the emulator.

--sys-gbc:

Sets the emulated system type to GBC. This option is not valid when running GBA games and is ignored by the emulator. DMG games will run as if on a GBC.

--sys-gba:

Sets the emulated system type to GBA. DMG/GBC games will run as if on a GBC, however, like a real GBA, the screen can be stretched horizontally by pressing the L and R triggers.

--sys-nds:

Sets the emulated system type to NDS.

--sys-rgb:

Sets the emulated system type to SGB. Allows GBE+ to display Super Game Boy borders. Borders can be toggled by pressing the L and R triggers.

--sys-rgb2:

Sets the emulated system type to SGB2. Allows GBE+ to display Super Game Boy and Super Game Boy 2 borders. Borders can be toggled by pressing the L and R triggers.

--mbc1m:

Forcibly emulates MBC1 games as MBC1M variants. Use this option for games like Mortal Kombat I & II, or Bomberman Collection.

--mbc1s:

Forcibly emulates an MBC1S cartridge used for the Pocket Sonar.

--save-auto:

Automatically detects GBA save type.

--save-none:

Disables all GBA game saves.

--save-sram:

Forces GBA save type to SRAM.

--save-eprom:

Forces GBA save type to EEPROM.

--save-flash64:

Forces GBA save type to FLASH RAM (64KB).

--save-flash128:

Forces GBA save type to FLASH RAM (128KB).

--h, or --help:

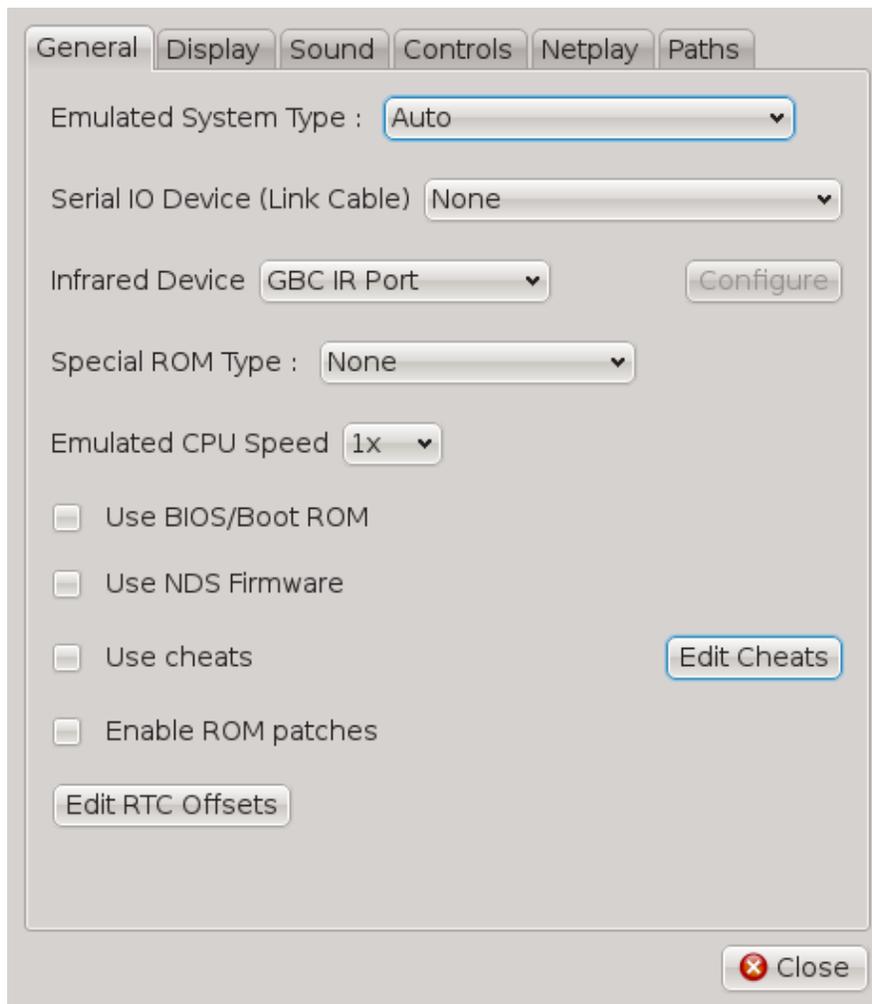
Displays a brief help message explaining all of the above options.

3. Configuration

GBE+ has many options that will affect how the program runs games. There are 6 major areas of configuration: General Settings, Display Settings, Sound Settings, Control Settings, Netplay, and Paths. The following sections detail what these options do for the Qt version of the emulator.

3.1 General Settings

This section deals with miscellaneous settings that don't belong to any particular category.



Emulated System Type:

Forces GBE+ to emulate a certain system. The following options are:

Auto: In this mode, GBE+ will automatically determine what system to emulate based on the game.

Game Boy [DMG]: Emulates the DMG (old black and white Gameboy). This option is not valid when running GBA games and is ignored by the emulator.

Game Boy Color [GBC]: Emulates the GBC. This option is not valid when running GBA games and is ignored by the emulator. DMG games will run as if on a GBC.

Game Boy Advance [GBA]: Emulates the GBA. DMG/GBC games will run as if on a GBC, however, like a real GBA, the screen can be stretched horizontally by pressing the L and R triggers.

Nintendo DS [NDS]: Emulates the NDS. Most NDS games are detected automatically, so this setting does not need to be explicitly changed. NDS emulation is fairly experimental.

Super Game Boy [SGB]: Emulates the Super Game Boy. The border can be disabled by pressing the input assigned to the R trigger, and disabled by pressing the input assigned to the L trigger.

Super Game Boy 2 [SGB2]: Emulates the Super Game Boy 2. This is basically the same as the Super Game Boy, except some games such as Tetris DX will show different borders.

Serial IO Device (Link Cable):

Specifies the type of device that should be connected to the emulated Game Boy. The following options are:

None: Emulates a regular Game Boy with nothing attached to it.

GB Link Cable: Emulates the GB Link Cable used for GB and GBC games. This option is necessary for netplay to work. See **Section 6.1** for more details on setting up netplay.

GB Printer: Emulates the GB Printer. Images generated by the GB Printer are automatically saved to the screenshot directory. See **Section 3.6** for more details on setting up the screenshot directory.

GB Mobile Adapter: Emulates the GB Mobile Adapter used in a limited number of Japan-only games. Support is currently limited to booting GB Mobile Trainer.

Bardigun Barcode Scanner: Emulates the card scanner for Barcode Taisen Bardigun. See **Section 7** for more details.

Barcode Boy: Emulates the Barcode Boy card scanner. See **Section 7** for more details.

DMG-07: Emulates the DMG-07 4-player adapter. See **Section 6.1** for general netplay configuration and specifically **Section 6.2** for configuring netplay for the DMG-07.

GBA Link Cable: Emulates the GBA Link Cable used for GBA games. This option is still experimental and not currently functional.

GB Player Rumble: Emulates the special rumble feature some GBA games unlock when using the Game Boy Player. Mostly experimental and only verified to work with homebrew software. The rumble feature must also be enabled in the controller settings. See **Sections 3.4** for more details.

Soul Doll Adapter: Emulates the Soul Doll Adapter for the GBA games Legendz: Isle of Trial and Legendz: Sign of Nekrom. See **Section 7** for more details.

Battle Chip Gate: Emulates the Battle Chip Gate used in various Mega Man games on the GBA. See **Section 3.4** and **Section 7** for more details.

Progress Chip Gate: Emulates the Progress Chip Gate used for Megaman Battle Network 5/Rockman.EXE 5. See **Section 7** for more details.

Beast Link Gate: Emulates the Beast Link Gate used for Megaman Battle Network 6/Rockman.EXE 6. See **Section 7** for more details.

Please note that when changing the Serial IO Device (Link Cable) option during gameplay, it will only take effect when resetting or booting a new game.

Infrared Device:

Specifies the type of infrared device that should be connected to the emulated Game Boy Color. The following options are:

GBC IR Port: Emulates the standard GBC IR port, used in various GBC-to-GBC communications. If a game does not use the GBC IR port (e.g. older DMG games) this option has no effect.

Full Changer: Emulates the Full Changer accessory used with Zok Zok Heroes. Click the Configure button for more options. See **Section 7** for more details.

Pokemon Pikachu 2: Emulates IR communications with the Pokemon Pikachu 2. Click the Configure button for more options. See **Section 7** for more details.

Pocket Sakura: Emulates IR communications with the Pocket Sakura. Click the Configure button for more options. See **Section 7** for more details.

TV Remote: Emulates random IR signals as if from a TV remote. See **Section 7** for more details.

Constant IR Light: Emulates a constant IR light in Chee Chai Alien. See **Section 7** for more details.

Special ROM Type:

Some games have unique cartridges that are not easily detected automatically. In these cases, GBE+ can manually specify what type of cartridge to emulate. The following options are:

None: GBE+ will not emulate any of the supported special cartridge types. If unsure, leave this as the default option.

DMG - MBC1M: Emulates MBC1 multicart variants known as MBC1M. Use this for games like Mortal Kombat I & II or Bomberman Collection.

DMG - MBC1S: Emulates the MBC1S cartridge used for Pocket Sonar. See **Section 7** for more details.

DMG - MMM01: Emulates MMM01 multicarts. Use this for games like Taito Variety Pack or Momotarou Collection 2.

AGB - RTC: Emulates the Real-Time Clock found on some GBA games. Use this for games like Pokemon Ruby, Sapphire, and Emerald, as well as the RTC can be artificially manipulated via RTC offsets. See more about that option further down in the section.

AGB - Solar Sensor: Emulates the Solar Sensor found in the Boktai games. The intensity of the sunlight can be controlled dynamically via Context Up and Context Down controls. See **Sections 3.4** and **Section 7** for more details.

AGB - Rumble: Emulates the rumble found in Drill Dozer. *This is not to be confused with the rumble feature used for Game Boy Player games like Pokemon Pinball: Ruby & Sapphire or Mario & Luigi: Superstar Saga (which is mostly unsupported in GBE+).* The rumble feature must also be enabled in the controller settings. See **Sections 3.4** and **Section 7** for more details.

AGB - Gyro Sensor: Emulates the gyroscope sensor in WarioWare: Twisted. The game can be played with Context Left and Context Right controls. See **Sections 3.4** and **Section 7** for more details.

AGB - Tilt Sensor: Emulates the tilt sensor in Yoshi Topsy-Turvy / Universal Gravitation. This game can be played with Context Left and Context Right controls. See **Sections 3.4** and **Section 7** for more details.

Emulated CPU Speed:

Alters the CPU's speed by 1x, 2x, 4x, or 8x. Only works for the DMG/GBC/SGB core!

Use BIOS/Boot ROM:

This instructs GBE+ to boot a system's BIOS or Boot ROM. When checking this option, users must ensure that the proper paths to the DMG, GBC, or GBA files are configured. See **Section 3.6** for more details on configuring BIOS and Boot ROM files.

User NDS Firmware:

This instructs GBE+ to load and boot the NDS core using a given firmware file. Currently does not work.

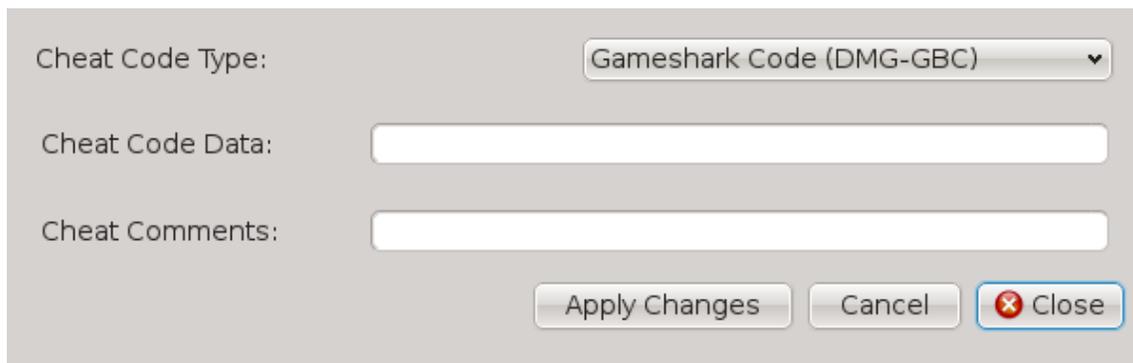
Use cheats:

Enables cheats for DMG and GBC games via GameShark and Game Genie codes. Both code types can be used simultaneously. To add, delete, and modify cheats, click on the Edit Cheats button to enter the following menu:



The screenshot shows a window titled "Edit Cheats" with a list of cheats. The first cheat is "Gameshark Code: 12345678" with description "GS Code #1". The second cheat is "Game Genie Code: 123456789" with description "Game Genie Code #1" and is highlighted in blue. Below the list are four buttons: "Edit Cheat", "Delete Cheat", "Add Cheat", and "Close".

Current cheats are displayed in a list like such. To edit or delete a specific cheat, click on it and highlight it as the screenshot above demonstrates, then click the **Edit Cheat** or **Delete Cheat** button. **Add Cheat** will bring up a menu such as this:



The screenshot shows a dialog box for adding a cheat. It has three fields: "Cheat Code Type" with a dropdown menu set to "Gameshark Code (DMG-GBC)", "Cheat Code Data" with an empty text box, and "Cheat Comments" with an empty text box. At the bottom are three buttons: "Apply Changes", "Cancel", and "Close".

Select the cheat code type, then enter the cheat code data. Comments can be added to each cheat. If cheat code data format is incorrect, GBE+ will set it to 0, in which case, please edit the cheat to correct it. Cheats are loaded from and saved to the specified cheat file if available. See **Section 3.8** for more info.

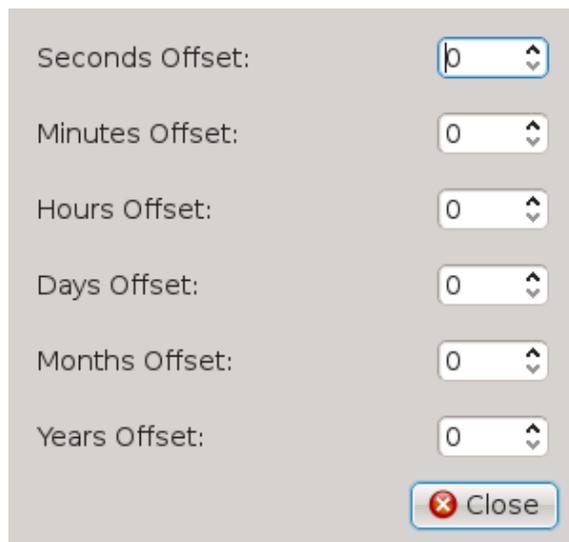
For the GBA, GBE+ supports V1 and V2 GameShark/Action Replay codes.

Use ROM patches:

Applies IPS or UPS patches when booting a game. The patch must match the ROM filename, except for the extension. For example, the patch file for **Super_Mario.gb** would be **Super_Mario.ips** or **Super_Mario.ups**.

Edit RTC Offsets:

Some games - such as Pokemon Gold, Silver, Crystal, Ruby, Sapphire, and Emerald - take advantage of special cartridges that have real-time clocks in them. GBE+ will use the current system time for this feature, however, users can manually adjust the clock without having to change their system clock. GBE+ will add values to the current time; this makes it possible to jump from day to night and back again in the Pokemon games with minimal effort. The following screen is used for editing RTC offsets:



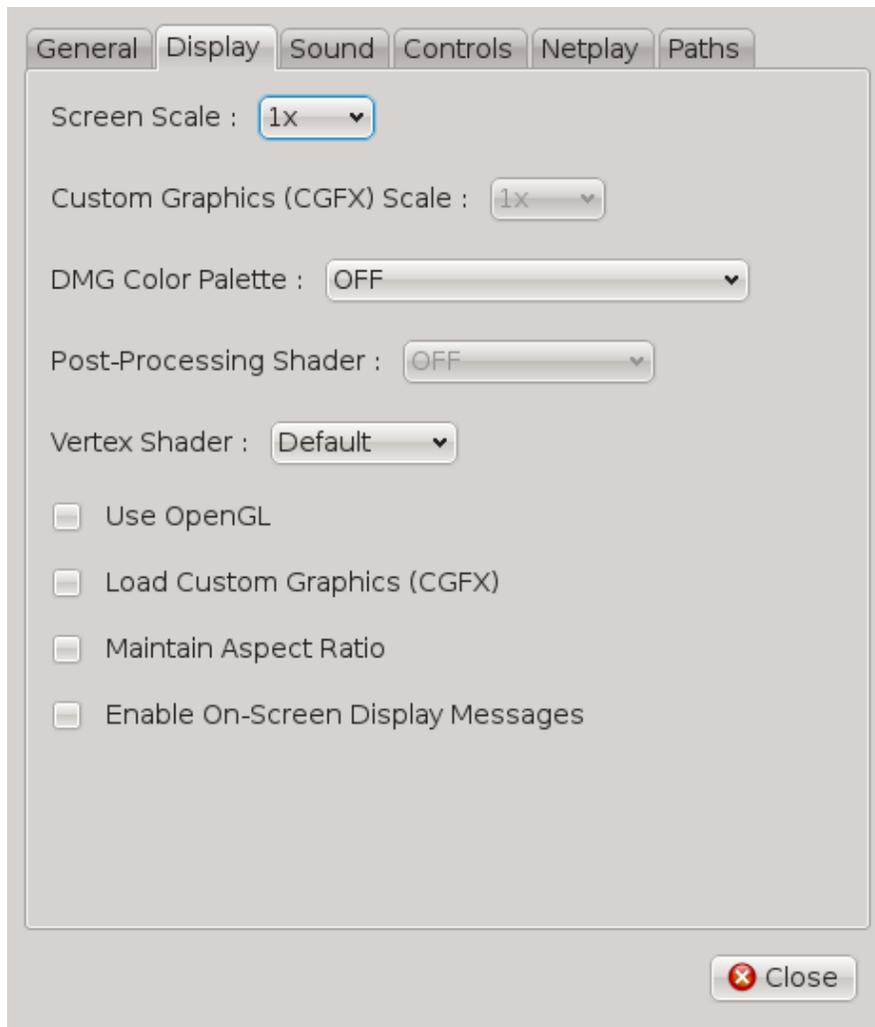
The screenshot shows a settings window with a light gray background. It contains six rows of labels and spinners:

- Seconds Offset: spinner with value 0
- Minutes Offset: spinner with value 0
- Hours Offset: spinner with value 0
- Days Offset: spinner with value 0
- Months Offset: spinner with value 0
- Years Offset: spinner with value 0

At the bottom right of the window is a button with a red 'X' icon and the text "Close".

3.2 Display Settings

Display settings deal with how GBE+ draws things on screen.



Screen Scale:

Determines what factor to scale the original game screen. Factors of 1x to 10x are applicable. Unlike the command-line version, the Qt version can scale the image regardless of whether OpenGL is used or not.

Custom Graphics (CGFX) Scale:

Determines what the input scale for CGFX is. *When loading CGFX into the emulator, this option must be set to the correct scale.* For example, if a user makes HD graphics that are 4x the size of the original graphics, this option must be set to 4x. Mismatching the scale will result in graphical errors when using CGFX. This scale is also multiplied by the Screen Scale to get the final screen size.

DMG Color Palette:

DMG games can be colorized with special palettes. **OFF** emulates standard grayscale colors. **DMG - Classic Green** emulates the old-school green LCD. The other options emulate palettes from the GBC Boot ROM.

Post-Processing Shader:

Applies various shaders (if any) to the final output image. Used to create different special effects or for scaling algorithms. OpenGL must be selected to use this option. Changing this option during gameplay takes effect immediately.

Vertex Shader:

Applies a vertex shader to the final output image. Used for unique transformations, such as inverting the image on the X-axis.

Use OpenGL:

Use OpenGL for all drawing/blitting operations. This option is faster than using software, especially when increasing the Screen Scale.

Load Custom Graphics (CGFX):

Loads Custom Graphics into the emulator. This option must be enabled in order to play using CGFX. Additionally, a manifest file must be selected in Paths. For full details on how to set up CGFX, refer to **Section 4.8**.

Maintain Aspect Ratio:

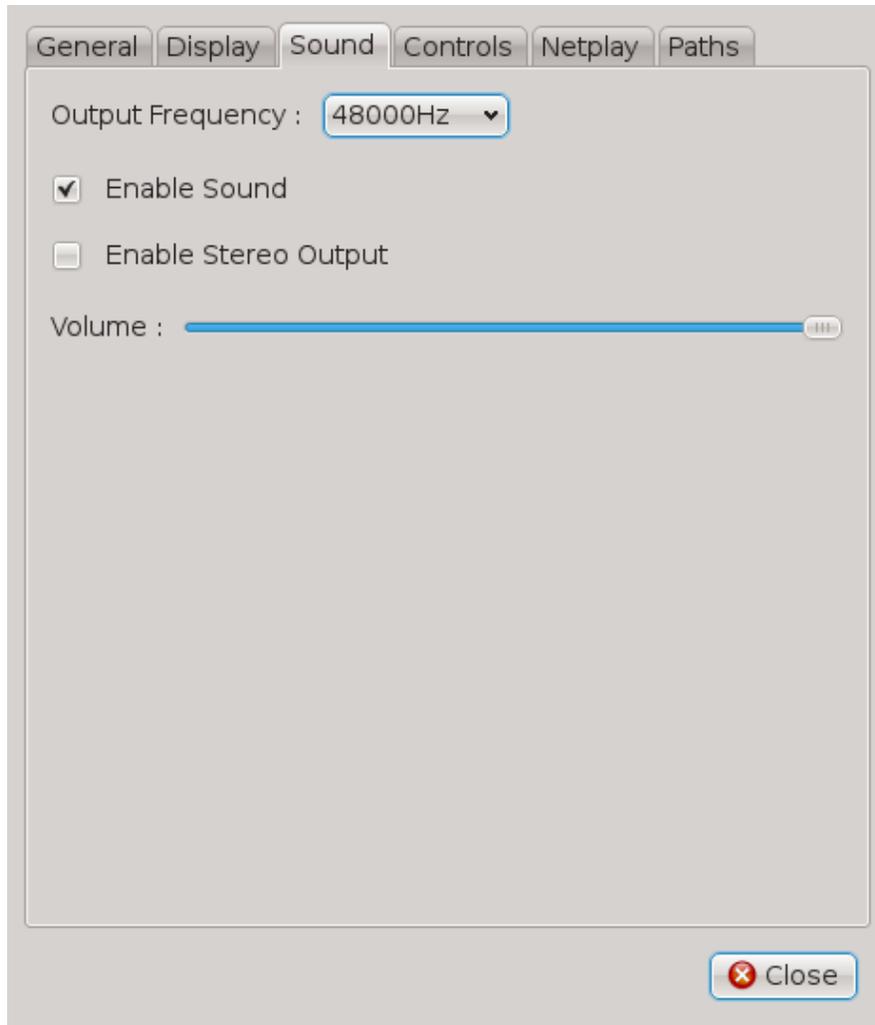
Forces GBE+ to maintain the same aspect ratio as the original system regardless of the window size. Leaving this option unchecked will let GBE+ fill in the available window space.

Enable On-Screen Display Messages:

Allows GBE+ to draw brief text messages on-screen when certain actions occur, such as loading/saving save states or taking a screenshot.

3.3 Sound Settings

Sound settings determine how GBE+ will process audio.



Output Frequency:

Determines the final output frequency of all sound. The available options are **11025Hz**, **22050Hz**, **44100Hz**, and **48000Hz**. With higher frequencies, GBE+ will produce better audio quality. Please note, changing this option only takes effect when booting or resetting a game. If unsure, please leave this option at its default setting.

Enable Sound:

Checking this option enables sound output. Unchecking this option will mute any sounds from GBE+.

Enable Stereo Output:

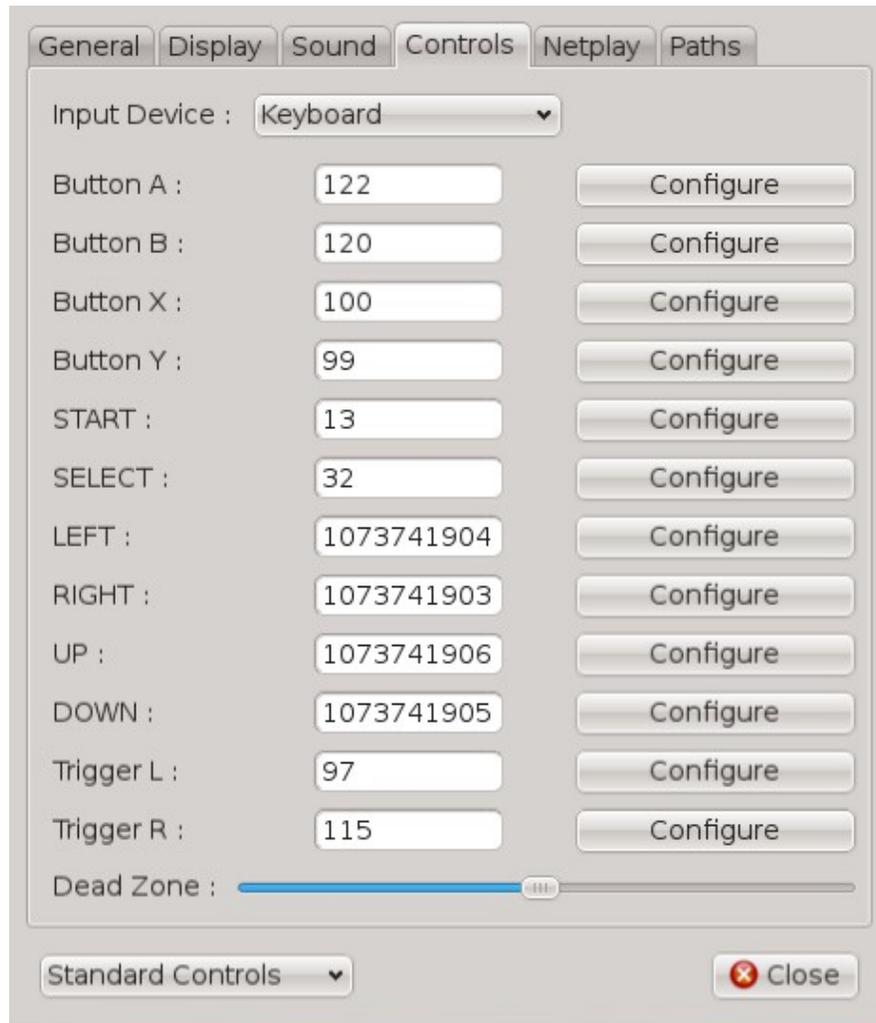
Checking this option enabled stereo output for DMG/GBC/SGB games. Currently not supported for the GBA and NDS cores.

Volume:

This slider controls the master volume for all sound output from GBE+. Turning it all the way down to zero will effectively mute the emulator.

3.4 Controls Settings

Controls settings allow users to configure input for the emulator via the keyboard or joysticks. There are two sections for input configuration, Standard Controls and Advanced Controls. Featured below are settings for the Standard Controls.



Input Device:

Selects the input device to configure. Both keyboards and joysticks can be configured. This option will contain a list of all available joysticks GBE+ can detect. Please note that both keyboard and joysticks can be used at the same time, regardless of what this option is set to.

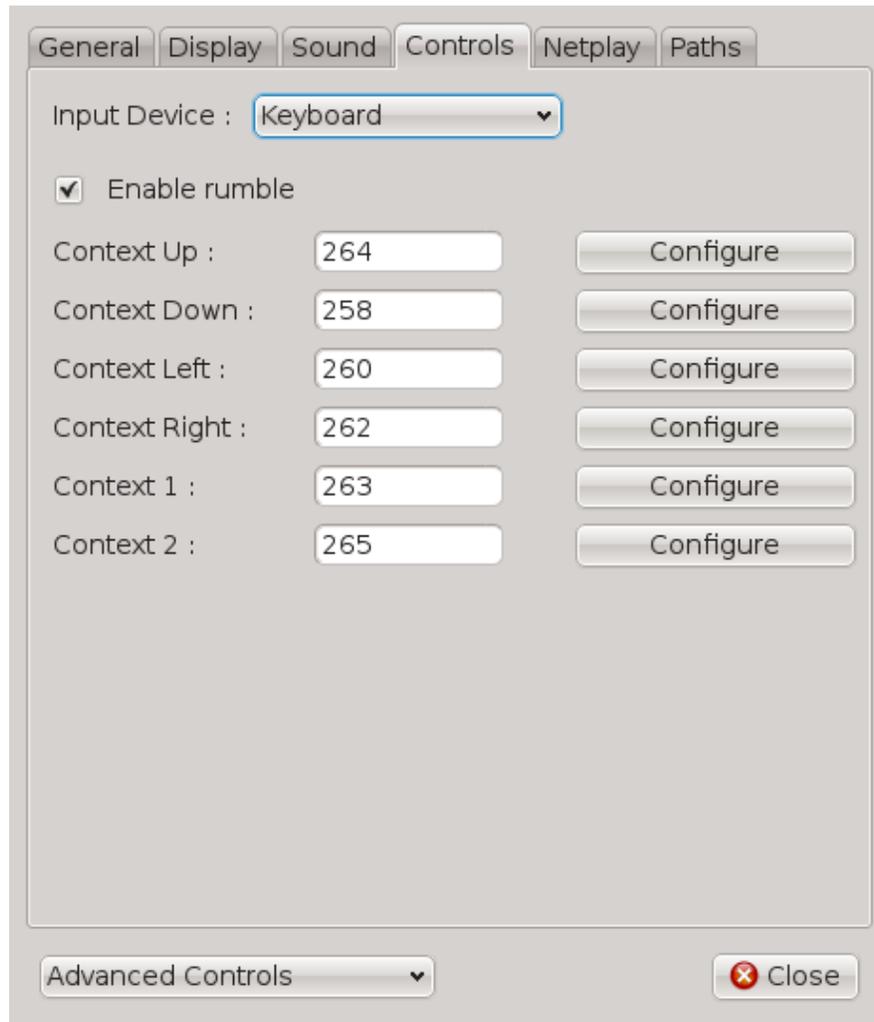
Buttons:

These are the individual buttons that can be configured. Press the **Configure** button for GBE+ to map input from a device. For joysticks, there is a 3 second delay before configuration is processed. During this time, please continue holding down on the joystick for GBE+ to finish mapping.

Dead Zone:

Configures the dead zone for joystick axes. A smaller dead zone means less tilt on an axis is needed to trigger an input. A larger dead zone means more tilt on an axis is needed to trigger an input. If unsure, please leave this option at its default setting.

Advanced Controls cover areas like enabling rumble and the Context Buttons.



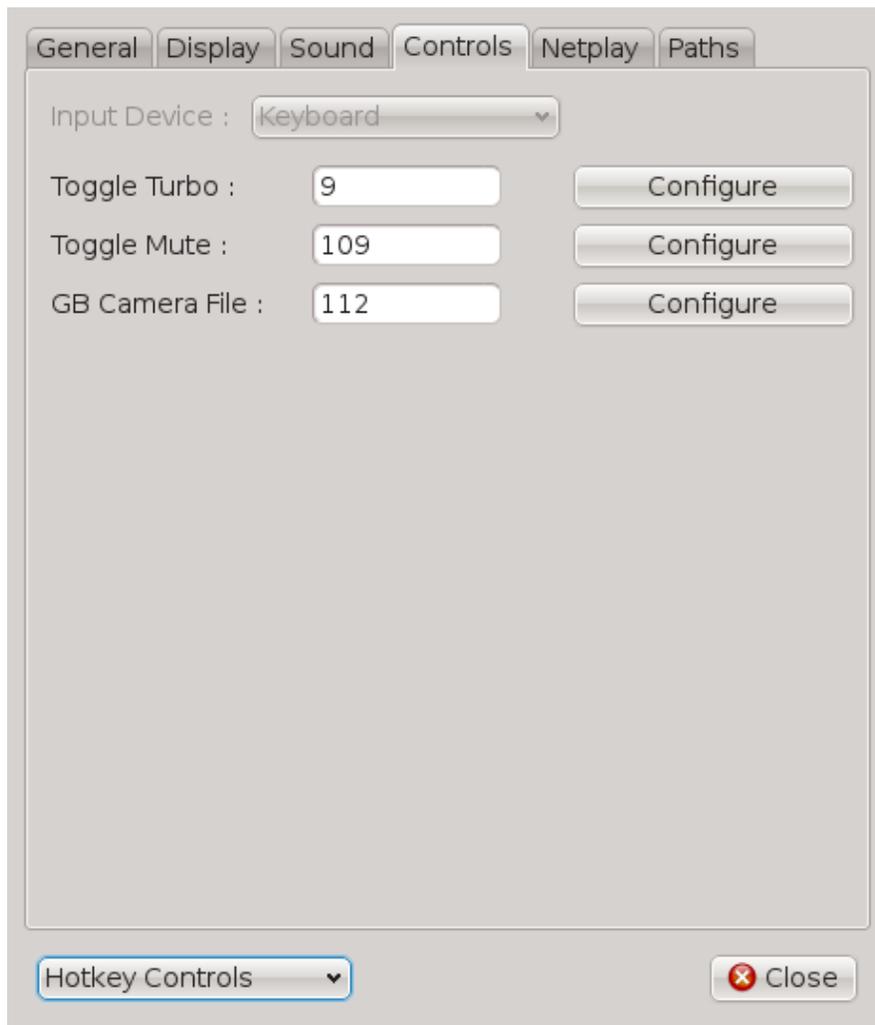
Enable rumble:

Turns on haptic feedback (rumble support) for various games that use rumble carts such as GBC titles like Pokemon Pinball or GBA titles like Drill Dozer or GBA software designed for the Game Boy Player. This option only works for joysticks that support some sort of haptic feedback.

Context Buttons:

Context buttons are meta-buttons that serve different roles depending on certain games. For example, in Kirby Tilt 'n' Tumble, the context buttons are used to move “tilt” the emulated GBC up, down, left or right. For WarioWare: Twisted, Context Left and Context Right determine which direction to spin the emulated GBA. For the Boktai games, Context Up and Context Down dynamically control the level of sunlight hitting the sensor. Context Left, Right, Up, and Down are used for the Battle Chip Gate controls on the following page. These buttons are configured exactly like the buttons in Standard Controls.

Hotkey Controls cover certain configurable shortcuts to different actions within GBE+. Currently, hotkeys only apply to keyboard input.

**Toggle Turbo:**

Lets GBE+ run at maximum speed while this key is held down. Defaults to the TAB key. It should be noted that many OpenGL drivers limit the FPS in some way. Properly configure the OpenGL to allow higher FPS, or simply disable OpenGL in GBE+.

Toggle Mute:

Disables/enables all sound coming from GBE+. Defaults to the M key.

GB Camera File:

Loads an arbitrary BMP file from the user's computer into VRAM while GBE+ emulates the GB Camera. Defaults to the P key. This basically inserts any image into the GB Camera to take pictures with.

Battle Chip Gate Controls allow players to map specific Battle Chips to the Context Buttons.

**Battle Chip Gate Type:**

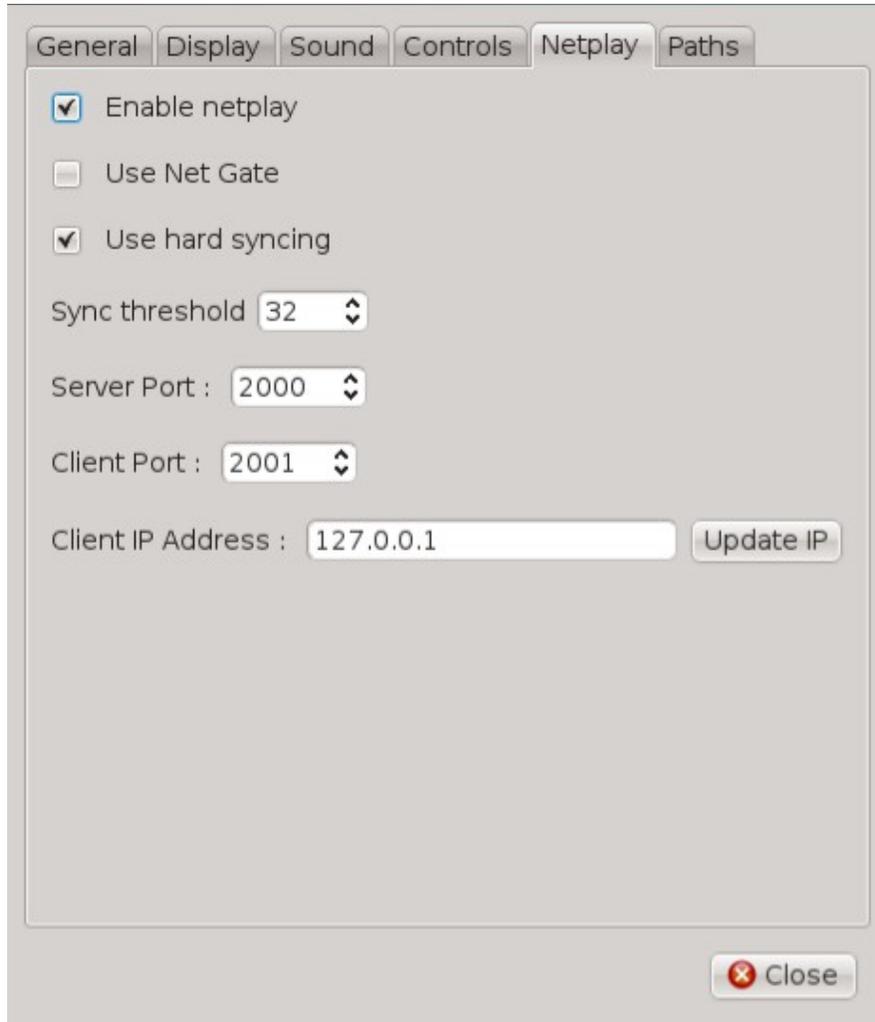
Chooses between Battle Chip Gate (the original model), Progress Chip Gate, and Beast Link Gate. Changing this option updates the list of available Battle Chips specific to each model. This option does not enable emulation of the Battle Chip Gate accessory. Enabling the Battle Chip Gate must be done through the Serial IO Device option. See **Section 3.1** for more details.

Battle Chip 1-4:

Chooses the Battle Chip to use. Battle Chips 1 through 4 are mapped directly to the Context Left, Right, Up, and Down buttons respectively. See **Section 7** for details on using the Battle Chip Gate in-game.

3.5 Netplay Settings

Netplay settings configure networking information necessary for two instances of GBE+ to connect with one another for multiplayer games. For a guide on setting up netplay in GBE+, refer to **Section 6**.

**Enable netplay:**

Turns on netplay in GBE+.

Use Net Gate:

Allows an external program to connect to GBE+ to send Battle Chips when emulating one of the Battle Chip Gates. See **Section 6.3** for more details.

Sync threshold:

Sets the number of CPU cycles before two instances of GBE+ are synchronized. Used only when hard syncing is enabled. Default of 32 cycles is recommended for most DMG and GBC games, and lower values may be necessary for IR communications.

Use hard syncing:

Forces two instances of GBE+ to synchronize periodically. This may cause slowdowns, but it is also necessary for some games. If this option is unchecked, GBE+ may de-sync or fail to connect properly. If unsure, leave checked.

Server Port:

The port of GBE+ will send data to. This must match the Client Port of another instance of GBE+.

Client Port:

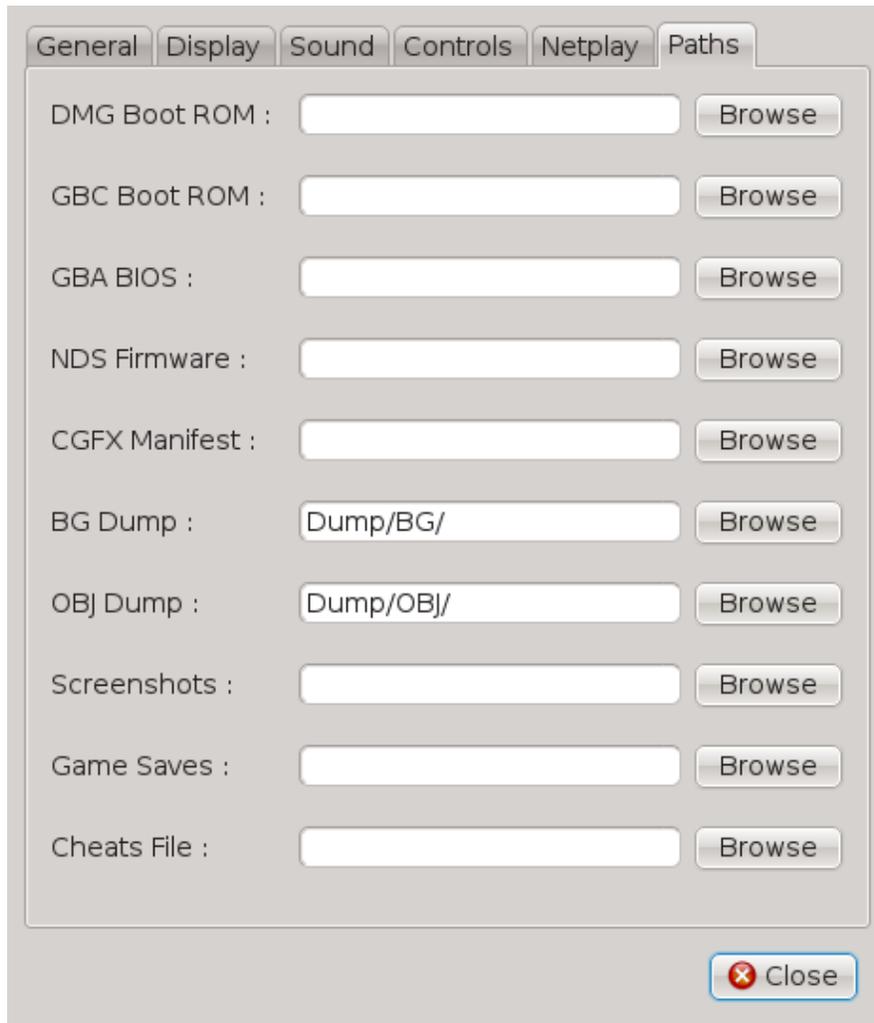
The port GBE+ will receive data from. This must match the Server Port of another instance of GBE+.

Client IP Address:

The IPv4 address of the client, a.k.a, the other instance/player GBE+ will connect to.

3.6 Paths Settings

Paths settings determine the location GBE+ will look to for important files such as screenshot directories, or the location of BIOS or Boot ROMs. To set a location, click the **Browse** button and choose a folder or file from the pop-up.



DMG Boot ROM:

This is the original Game Boy's Boot ROM file. Although booting this will have no effect on overall emulation, this will let users see the scrolling "Nintendo" logo.

GBC Boot ROM:

This is the Game Boy Color Boot ROM file. Booting up this will let users choose color palettes when running DMG games, and some DMG games (like *Metroid II: Return of Samus*) will have custom palettes.

GBA BIOS:

This is the Game Boy Advance BIOS file. Booting this up will let users see the GBA's "Nintendo" logo and the boot animation. Currently, all BIOS functions are high-level emulated, so GBE+ does not actually run the code from the GBA BIOS. In the future, low-level emulation of the BIOS will be possible.

NDS Firmware:

This is the firmware file GBE+ will use when booting up the NDS core. Booting from firmware is not possible yet.

CGFX Manifest:

This is the manifest file GBE+ will use when loading Custom Graphics or when dumping them via the Advanced Menu. See **Section 4** for more details about Custom Graphics.

BG Dump:

This is the folder GBE+ will look to when loading or dumping Background Tiles for Custom Graphics. By default, this is located in the **data** folder for GBE+. This folder *must* be located within the **data** folder for GBE+ to correctly find the image files.

OBJ Dump:

This is the folder GBE+ will look to when loading or dumping Sprite Tiles for Custom Graphics. By default, this is located in the **data** folder for GBE+. This folder *must* be located within the **data** folder for GBE+ to correctly find the image files.

Screenshots:

This is the folder GBE+ will use to store screenshots.

Game Saves:

This is the folder GBE+ will use for all game saves. If nothing is specified for this path, GBE+ will look for the save file in the same directory as the ROM.

Cheats File:

This is the file GBE+ will use to look up cheats. If nothing is specified for this path, cheats can still be used (by adding them through **Edit Cheats** in the General tab, see **Section 3.1**), however GBE+ will not be able to save them. For more details about the format of the cheat file itself, see **Section 3.8**.

3.7 The .ini File

GBE+ can be configured through a .ini text file. This file includes nearly all of the same options as the Qt options previously described. For both the SDL and Qt versions of GBE+, the .ini file will be loaded and automatically set up any options. The .ini file and its format are self-documented, so please refer to the default .ini file to see how it works and how to edit it.

GBE+ will always search for a .ini file in the same folder as the emulator itself. If no such file exists, GBE+ will search in the **data** folder for its .ini file. If no .ini file exists in the **data** folder either, GBE+ will use its default settings.

For the Qt version, GBE+ will always update and save the .ini file. This way, changes to settings are remembered during the next play session.

3.8 The Cheat File

Below is the format for the cheat file. While user can manage cheats exclusively through the GUI, they can also edit the cheat file manually. Each cheat code has three components:

Cheat Type:

This is a simple ID, either “GG” for Game Genie or “GS” for GameShark on the DMG/GBC or “GSA1” for GameShark/Action Replay V1 and V2 on the GBA.

Cheat Code Data:

This is the actual cheat code (with no spaces).

Comment:

Any comments about the cheat.

The following is an example of what a cheat code file looks like as a text file:

```
[GG:123456789:Test Code]
[GG:DEADBEEF:Test Code 2]
[GS:FEEDFACED:Another Test]
[GS:00000001:More]
[GG:11111111:This had better work]
[GSA1:0123456789ABCDEF:Game Shark Advance Code]
```

4. Custom Graphics (CGFX)

Custom Graphics (referred to as CGFX) are an exciting new way of modding in-game graphics, much like custom textures used on other platforms. This section details how GBE+ can extract and replace tiles for backgrounds and sprites.

4.1 Overview



CGFX works by first grabbing tile data from a game, which GBE+ offers numerous tools for achieving. Once the graphics have been extracted or “dumped”, they are edited to look like something else. The changes can be simple 1:1 recolors of black-and-white-only games such as the screenshot above, or they can be bigger, high-definition versions. Once those changes are made, GBE+ can automatically load the graphics and draw them on-screen when running the game.

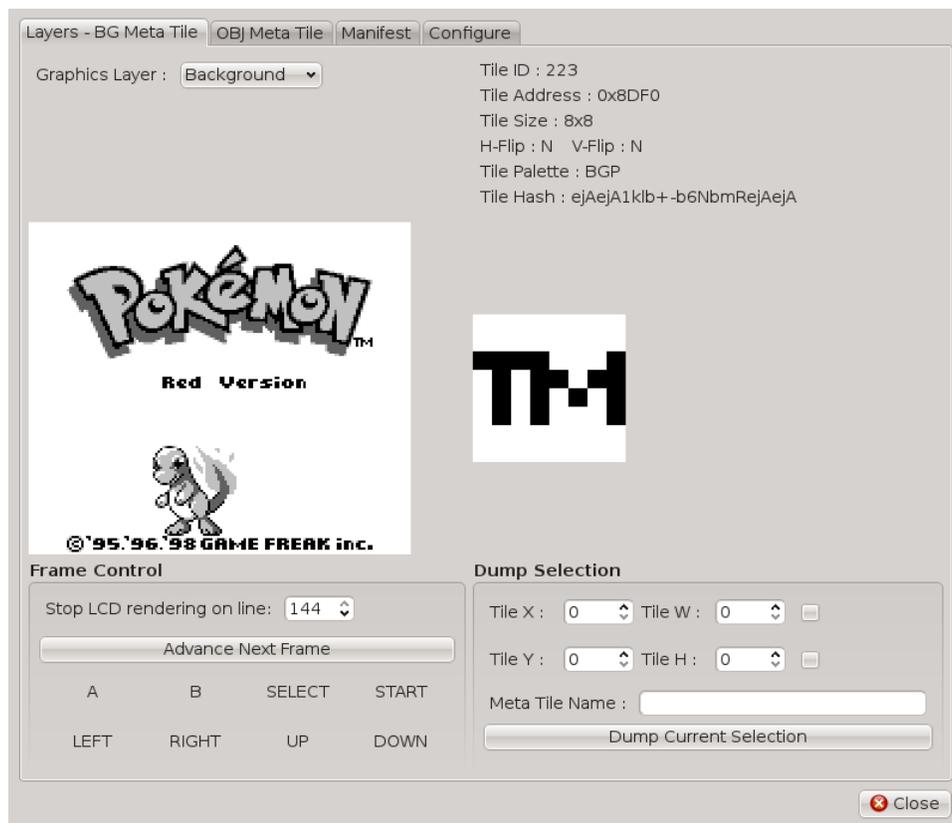
4.2 Dumping Graphics

In order to replace a game's original graphics with CGFX, the original graphics must first be extracted from the game. This process is known as "dumping" or "ripping". GBE+ provides a dedicated user interface for this very purpose. **At this time, it is important to note that CGFX only work for GB and GBC games.** Support for GBA games will come in later releases. The guide is applicable to the Qt version only.

Before beginning, please make sure the paths for the BG and OBJ dump folders are correct. Go to **Options -> Paths** to check or change any settings. **The BG and OBJ dump folders must always be in GBE+'s data folder.** By default, these two folders are data/Dump/BG and data/Dump/OBJ, but they can be changed, moved or renamed (so long as they remain in the data folder).

When dumping custom graphics, a manifest file should be specified. This is simply a text file that will tell GBE+ where to look for custom graphics and how it should load and handle them. Go to **Options -> Paths** and make sure the CGFX Manifest points to a valid text file. When starting new CGFX projects, the manifest may be left as an empty text file. GBE+ will add data to the manifest automatically.

To get started, load a game from the menu. Once the game starts, click on **Advanced -> Custom Graphics...** The screen below will appear:

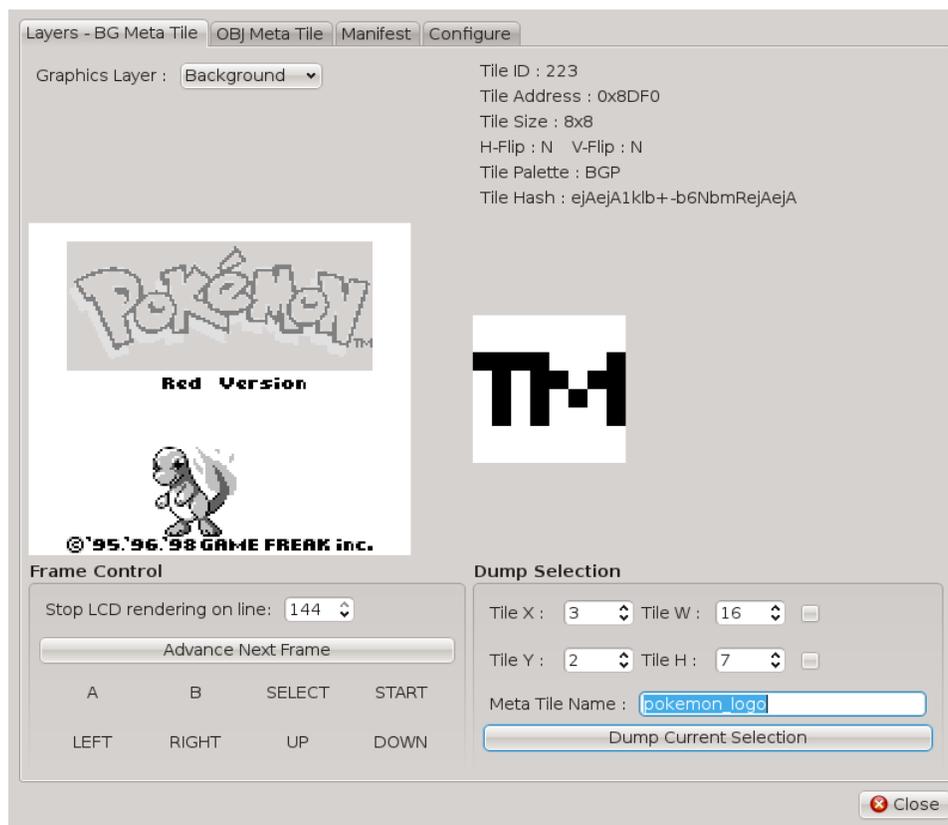


4.3 Layers - BG Meta Tile

The Layers - BG Meta Tile tab (pictured on the previous page) allows users to switch between the Game Boy's Background, OBJ, and Window layers. The current in-game screen will be displayed. Use the mouse to hover over a specific tile. A magnified version will appear on the right along with detailed data about the tile itself. Clicking here will dump the tile. This method is useful for dumping one tile at a time. However, it's possible to dump many tiles at once.

Meta-tiles offer users a way to dump a section of the screen all at once, then save the results as a single image. Take, for example, the POKEMON logo from Pokemon Red. The logo itself is just short of 100 8x8 background tiles. Normally, it would take many people hours to dump each tile and edit them by hand. As a convenience, GBE+ will let users save that logo as a single bitmap file which can be edited and loaded later.

Meta-tiles can be created for background tiles and sprite tiles. Meta-tiles for sprites have their own separate menu described later. Below the BG meta-tile menu is explained.



To create a BG meta-tile, go to **Dump Selection**. In this area are parameters to control which section will be dumped. The X and Y boxes determine where the on-screen selection will begin. The W and H boxes determine the dimensions of the selection. Clicking and dragging the mouse will also create a selection. The selection will automatically highlight the area to be dumped. Additionally, **EXT_VRAM_ADDR** and **EXT_AUTO_BRIGHT** options are available (see **Section 4.7** for more details).

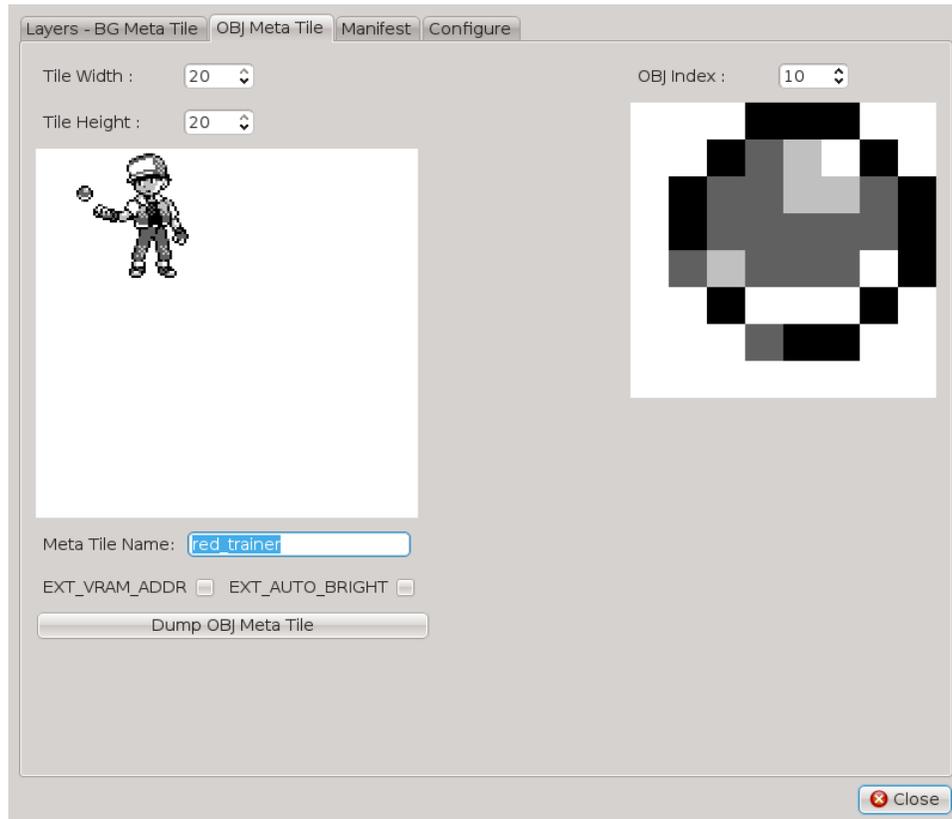
Once the selection has been made, enter in a name for the file. **This file will automatically appear in the BG Dump folder and will automatically be saved as a .BMP file.** All the necessary manifest entries are written when saving as well.

GBE+ provides extra tools under Frame Control to aid tile dumping. The Game Boy's screen is 160x144, and GBE+ can stop rendering after a specific line. This is useful because some games (such as Mega Man I) change their graphics once they reach a specific line.

Advancing frame-by-frame is also possible when clicking on the Advance Next Frame button. Furthermore, GBE+ can feed input into the emulated game. This allows users to perform actions such as walking around so that different animations can be dumped. Clicking on an input will highlight it and emulates pressing that button on a Game Boy. The button is pressed down as long as the input in this menu remains highlighted. Clicking on an input a second time removes highlighting and emulates releasing that button on a Game Boy.

4.4 OBJ Meta Tile

The OBJ Meta Tile tab provides users with tools to create meta-tiles for OBJs, also known as sprites. Unlike backgrounds meta-tiles, OBJ meta-tiles require a separate process for dumping. This tab contains an editor on the left for crafting sprite-sheets. On the right is a preview of the currently selected OBJ to be added to the sprite-sheet.

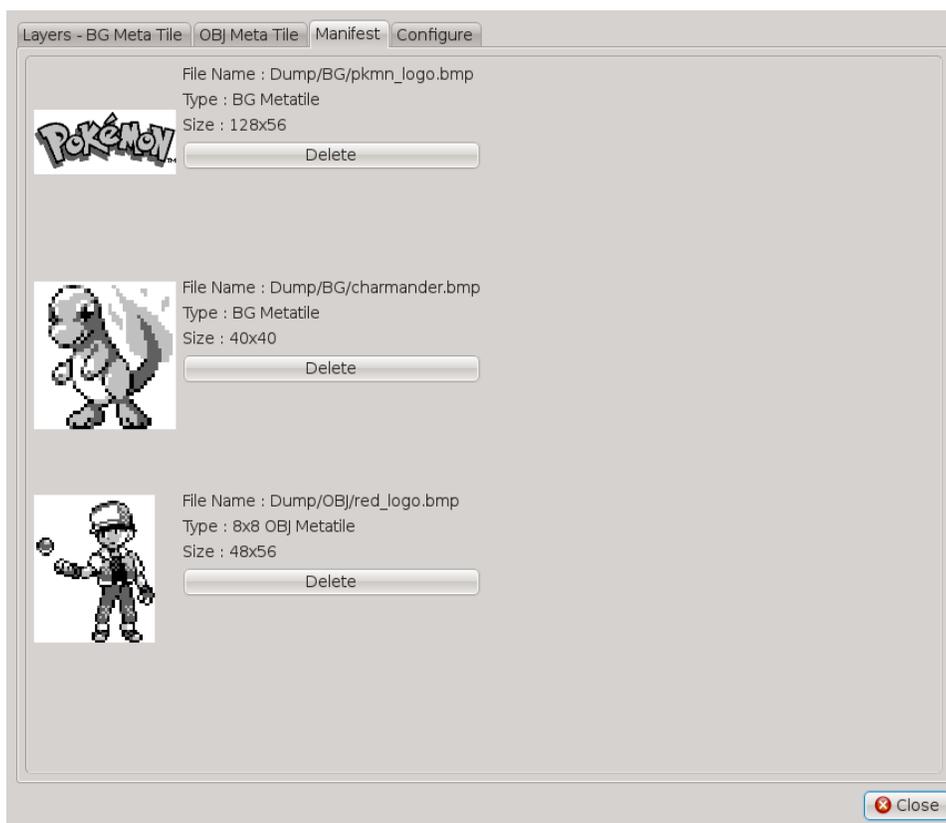


First, select an OBJ from the index (0-39), then hover the mouse over the meta-tile. The current tile will be highlighted. Left-clicks add graphics to the sprite-sheet while right-clicks delete them. The size of the meta-tile will change based on the Tile Width and Tile Height values (20 max for each).

Like BG meta-tiles, OBJ meta-tiles have **EXT_VRAM_ADDR** and **EXT_AUTO_BRIGHT** options; see [Section 4.7](#) for details. Once the OBJ meta-tile is complete, press the Dump OBJ Meta Tile button to save the meta-tile as an image file and generate manifest entries. **This file will automatically appear in the OBJ Dump folder and will automatically be saved as a .BMP file**

4.5 Manifest Tab

The manifest tab provides a visual overview of all graphics stored in the manifest file. Here, users can view each entry as a list, complete with a miniature preview along with important details such as the file path to the replacement image, image size, and the type of manifest entry. Use this tab to keep track of what graphics have currently been dumped. Items from the manifest can be deleted as well.



4.6 Configuration Tab

This tab contains extra settings for advanced CGFX users. Do not use these unless you know what you're doing. If unsure, do not touch these.



Use advanced menu:

Every time graphics are dumped, GBE+ will let users specify additional options through an advanced menu. See [Section 4.7](#) for an extended explanation of the advanced menu and its features. **Enabling this option is the default. It is highly recommended that you do not turn it off.**

Ignore blank/empty tiles when dumping:

Useful for the above automatic dumping options. Sometimes games generate graphics that are blank tiles (solid colors), that GBE+ may mistake for graphics that might be relevant. This option will prevent GBE+ from dumping them at all.

Automatically add transparency color when dumping OBJs:

OBJs have their own transparency color according to the DMG or GBC palette being used by a game. Sometimes these colors are quite obvious, but other times it can be confusing or ambiguous. Enabling this option allows GBE+ to use the transparency color (specified in the .ini file) and automatically add it to any OBJ tiles dumped. This is a great time saver in many cases.

4.7 Using the Advanced Menu

Manifest files are necessary for properly loading custom graphics in GBE+. They act like lists that describe what the emulator should do when loading custom graphics. GBE+ will let users automatically add entries into a manifest file via the Advanced Menu. Simply check the above Use advanced menu option, and every time a tile is dumped, GBE+ will write the manifest file for you. The following screen will pop up every time a tile is dumped:



The screenshot shows a dialog box with a light gray background. At the top, there is a label "Destination Folder :" followed by a text input field containing "Dump/BG/" and a "Browse" button to its right. Below this is a label "Tile Name :" followed by an empty text input field and another "Browse" button. Underneath are two checkboxes: the first is labeled "EXT_VRAM_ADDR" and the second is labeled "EXT_AUTO_BRIGHT"; both are currently unchecked. At the bottom right, there are two buttons: "Dump Tile" (highlighted with a blue border) and "Cancel".

Destination Folder:

This will specify what folder to use to dump this tile. By default, it is either the BG or OBJ dump folder specified in **Options** -> **Paths**.

Tile Name:

This gives the tile a unique name. By default, GBE+ will name the graphics after its hash. Use this to give it a descriptive name like "Mario_Hat_1.bmp" or "Link_Sword.bmp"

EXT_VRAM_ADDR:

Checking this will allow GBE+ to associate this tile with its VRAM address. This is useful for dumping graphics that have the same hash but are located in different parts of VRAM. Often, games like Super Mario Land used a blank, white tile for the "sky" and the "ground". Ordinarily, GBE+ thinks these two are the same (they have the same data after all), but since they are located in different areas of VRAM, they can be replaced separately. This is an advanced option; if unsure, leave unchecked!

EXT_AUTO_BRIGHT:

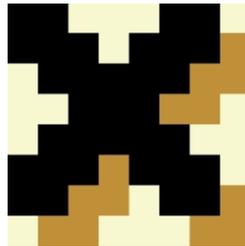
This option is generally only applicable to GBC games. This will force GBE+ to automatically adjust the brightness of this tile based on the game's original colors. With the GBC, many games had "fade in" and "fade out" effects programmed. When EXT_AUTO_BRIGHT is enabled for a tile, GBE+ will try to lighten or darken an image appropriately when this happens in a game. Generally, this can be reserved for things like intro logos or title screen elements. This is an advanced option; if unsure, leave unchecked!

Manifest files can be edited manually however. See **Section 4.8** for more info about the manifest file.

4.8 The Manifest File

Manifests describe what files GBE+ needs to load when using CGFX. The manifest basically acts like a detailed list, pointing the emulator to the correct images. Entries within the manifest may also provide additional information on how to process the images. The primary advantage of manifests is that it reduces the time it takes for GBE+ to search for the correct CGFX assets and it gives users greater flexibility for organizing their projects.

Manifests allow the user to associate a hash with any file name they want. Even a relatively simple looking tile like this:



could end up having a complex hash like this: `0630_07507J05X04T05X07J07503-`

That's not very descriptive, and it's hard for most people to remember. Manifests allow GBE+ to associate that hash with a simpler name like "X_FONT.BMP" or whatever users want, as long as that name points to a valid image file. The format of manifest files is simple. Each entry is enclosed in brackets ([]), and each parameter is separated by a colon (:).

4.8.1 Tile Entries

Manifest files are made up of 2 different kinds of entries, **Tile Entries** and **Meta-tile Entries**. **Tile Entries** describe the individual tiles that are loaded up, while **Meta-tile Entries** point to a bitmap image made up of multiple tiles which are then broken down into smaller, individual tiles. Below is the format for a **Tile Entry**

1. GBE+ hash:

This is the hash GBE+ computes (obtained while dumping graphics), such as

```
"0630_07507J05X04T05X07J07503-"
```

2. Image file name:

This is the file name for the image associated with the above hash. Use a path relative to GBE+'s data folder. **Enclose path in single-quotes!** When a **Tile Entry** is used in conjunction with a **Meta-tile Entry**, this parameter refers to the Meta-tile's **Meta Name** parameter followed by its ID in the form of "_123".

3. Image type:

This determines what type of image the file represents.

1 = DMG Sprite, 2 = GBC Sprite, 10 = DMG Background Tile, 20 = GBC Background Tile.

4. EXT_VRAM_ADDR:

The address of a tile in VRAM, necessary for CGFX under certain scenarios. GBE+ will generate this. If uncertain, set to 0!

5. EXT_AUTO_BRIGHT:

Setting this to 1 will let GBE+ automatically correct the brightness of CGFX to match the original game's brightness. If uncertain, set to 0!

4.8.2 Meta-tile Entries

Meta-tiles (from **Section 4.3**) are simply large bitmaps that consist of multiple tiles. GBE+ will grab this image and automatically break it down into separate tiles. For example, if you are editing a title screen that is 80x40 pixels, GBE+ will load this and break it down into 8x8 tiles. If you are doing a 3x HD version with a 320x120 image, GBE+ will also break it down accordingly into 24x24 tiles.

Meta-tile Entries specify where to look for this bitmap, and how it should be broken down. A **Meta-tile Entry** simply gives this bitmap an ID called the **Meta Name**, and the original dimensions of each tile. In order for GBE+ to actually use any of the tiles from the Meta-tile bitmap, a regular **Tile Entry** must follow and use the **Meta Name** to grab a specific tile number. Below is the format for a **Meta-tile Entry**.

1. Image file name:

This is the file name for the image associated with the above hash. Use a path relative to GBE+'s data folder. **Enclose path in single-quotes!**

2. Meta Name:

This is a unique text-based ID for a specific meta-tile. It can be named anything, however, **do not use quotes (single or double) or colons in the name.**

3. Meta Type:

Specifies the base tile type of meta-tile. This represent the tile's original format, not the modified format. 0 = 8x8 Background Tile, 1 = 8x8 Sprite Tile, 2 = 8x16 Sprite Tile.

4.8.3 Example Manifest File

A working manifest file would look something like this:

```
#Comments look like this

#This is a basic Tile Entry
#Font stuff below, pulling data from our Font folder, these are sprites btw
#The CGFX folder is in GBE+'s "data" folder
[07507J05X04T05X07J07503-:'CGFX\Font\X.bmp':1:0:0]
[07X07507507X07507507503-:'CGFX\Font\R.bmp':1:0:0]
[05X07506-05X04507505X03-:'CGFX\Font\S.bmp':1:0:0]

#You can even divide up graphics into your own categories for easier editing!
#Imagine we're loading Mario sprites or something here
[10102128173--1312123211:'CGFX\Mario\M_1.bmp':1:0:0]
[101053232223--333123211:'CGFX\Mario\M_2.bmp':1:0:0]

#This is a Meta-tile Entry
#It specifies a group tiles to extract from a single, large bitmap
#The Meta Name is "T_SCREEN"
['CGFX\Screens\Title.bmp':T_SCREEN:0]

#After the Meta-tile Entry, populate the manifest with regular Tile Entries
#Instead of a file path, the second parameter is "T_SCREEN_XXX"
[eJafdsHadd-adfdsgfdsaaaq:T_SCREEN_0:10:0:0]
[ejAejAejAejAejAejAejAejA:T_SCREEN_1:10:0:0]
[AddaAddaDDapp-4322102934:T_SCREEN_2:10:0:0]
```

Obviously, editing the manifest file by hand can be a bit daunting, therefore, it is only recommended for people who know what they are doing, or for troubleshooting problems with CGFX.

4.9.1 Editing Graphics

Once the graphics have been dumped/ripped, the next step is to edit them. Locate the files you have recently dumped (in GBE+ data folder) and open them in an image editor of your choice. Decide on what scale you want, e.g. 1x for 1:1 replacements or something like 3x for HD replacements. Scale the source image appropriately by that factor, then make any change necessary. At this time, GBE+ only supports 24-bit BMPs for dumping and loading. Please make sure the images saved are in the correct format.

For sprites, GBE+ supports a transparency color. By default, this is ugly green (#00FF00) so use that color to make transparent areas. This color can be changed by editing the gbe.ini file.

4.9.2 Loading Graphics

When loading custom graphics, a manifest file **must** be specified. This is simply a text file that will tell GBE+ where to look for custom graphics and how it should load and handle them. Go to **Options -> Paths** and make sure the CGFX Manifest points to a valid text file. This manifest file will automatically contain entries if these graphics were dumped through the Advanced Menu of the Custom Graphics window. Otherwise, users will have to manually create the manifest file themselves. Read more about manifest files here.

Before loading a game, go to **Options -> Display**. Change the **Custom Graphics (CGFX) Scale** to match the scale factor of the graphics you will be loading. Lastly, check the **Load Custom Graphics** option. Enjoy the results of your work.

5. Debugging

GBE+ offers a wide range of debugging options for those interested in what makes games run. Currently, GBE+ supports command-line debugging features for both the DMG and GBA cores. Additionally, GBE+ has a specific interface for debugging DMG/GBC games through the Qt GUI. Below the GUI version is detailed first. To access this, run a DMG or GBC game and go to **Advanced** -> **Debugger**. When opening the debugger, the game will automatically pause; it will resume when the window is closed.

5.1 Memory Mapped I/O Registers

This is the first tab of the debugger. It shows the current state of various MMIO registers.

Register Name	Value	Register Name	Value	Register Name	Value	Register Name	Value
0xFF40 - LCDC:	0xC3	0xFF12 - NR12:	0x08	0xFF22 - NR43:	0x00	0xFF69 - BCPD:	0xFF
0xFF41 - STAT:	0x41	0xFF13 - NR13:	0x00	0xFF23 - NR44:	0x80	0xFF6A - OCPS:	0x80
0xFF42 - SY:	0x00	0xFF14 - NR14:	0x80	0xFF24 - NR50:	0x77	0xFF6B - OCPD:	0x00
0xFF43 - SX:	0x00	0xFF16 - NR21:	0x00	0xFF25 - NR51:	0xFF	0xFF70 - SVBK:	0x00
0xFF44 - LY:	0x90	0xFF17 - NR22:	0x08	0xFF26 - NR52:	0x80	0xFF00 - P1:	0xFF
0xFF45 - LYC:	0xC7	0xFF18 - NR23:	0x00	0xFF4D - KEY1:	0x80	0xFF01 - SB:	0xE1
0xFF46 - DMA:	0xCB	0xFF19 - NR24:	0x80	0xFF4F - VBK:	0x00	0xFF02 - SC:	0x80
0xFF47 - BGP:	0xE4	0xFF1A - NR30:	0x00	0xFF51 - HDMA1:	0x00	0xFF04 - DIV:	0x84
0xFF48 - OBP0:	0xE4	0xFF1B - NR31:	0xFF	0xFF52 - HDMA2:	0x00	0xFF05 - TIMA:	0xAE
0xFF49 - OBP1:	0x6C	0xFF1C - NR32:	0x9F	0xFF53 - HDMA3:	0x00	0xFF06 - TMA:	0x77
0xFF4A - WY:	0xC7	0xFF1D - NR33:	0x00	0xFF54 - HDMA4:	0x00	0xFF07 - TAC:	0x04
0xFF4B - WX:	0xC7	0xFF1E - NR34:	0xBF	0xFF55 - HDMA5:	0x00	0xFF0F - IE:	0x0F
0xFF10 - NR10:	0x08	0xFF20 - NR41:	0xFF	0xFF56 - RP:	0x00	0xFFFF - IF:	0x00
0xFF11 - NR11:	0x00	0xFF21 - NR42:	0x08	0xFF68 - BCPS:	0x80		

Refresh Close

5.2 Palettes

The Palettes tab displays the current Background and Object palettes. On DMG games, there is only 1 BG palette, and 2 OBJ palettes, however on the GBC there are 8 for both. The relative RGB values for each color (0 - 31) will be displayed when clicking on a color. An enlarged preview of the color will appear on the right-hand side as well.



5.3 Memory

The Memory tab will display the entire contents of the memory map (0x0 through 0xFFFF on the DMG/GBC). All the values can be scrolled through and examined in detail. Additionally, as a convenience, the memory contents are also displayed as ASCII characters on the right-hand side.

The screenshot shows a debugger window with the 'Memory' tab selected. The interface includes tabs for 'I/O', 'Palettes', 'Memory', 'Disassembly', 'Obj Tiles', and 'BG Tiles'. The main area is divided into two columns: the left column displays hexadecimal memory addresses and their corresponding byte values, and the right column displays the ASCII characters corresponding to those bytes. A vertical scrollbar is visible on the right side of the ASCII column. At the bottom right, there are 'Refresh' and 'Close' buttons.

Address	Hex	ASCII
0x0000	87 E1 85 6F 30 01 24 2A 66 6F E9 00 00 00 00	...o0.\$*fo.....
0x0010	85 6F D0 24 C9 00 00 00 C5 4F 06 00 09 09 C1 C9	.o.\$.....0.....
0x0020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0040	F5 C5 D5 E5 C3 D4 09 00 F5 E5 C3 22 0B 00 00 00"
0x0050	FB F5 C5 D5 E5 C3 E3 0C F5 C3 19 0C 00 00 00 00
0x0060	D9 00 00 00 00 00 00 00 83 5F D0 14 C9 81 4F D0_.....0.
0x0070	04 C9 E5 87 6F 3E 00 8F 67 19 5D 54 E1 C9 E5 87o>..g.]T....
0x0080	6F 3E 00 8F 67 09 4D 44 E1 C9 FA 97 FF F5 7B EA	o>..g.MD.....{.
0x0090	97 FF EA 22 22 CD A0 00 F1 EA 97 FF EA 22 22 C9"....."
0x00A0	E9 42 44 46 47 48 4A 4C 4D 13 BD 12 11 23 4E 51	.BDFGHJLM....#NQ
0x00B0	52 53 54 57 59 21 10 7E 7F 2B 2D 62 64 66 67 68	RSTWY!.~.+-bdfgh
0x00C0	6A 6D 24 2A 2F 3A 5C 6E 71 72 73 74 77 79 3F 25	jm\$*/:\nqrstwy?%
0x00D0	26 3C 3D 3E 32 33 34 35 36 37 38 39 15 16 17 18	&<=>23456789....
0x00E0	40 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00	@.....
0x00F0	00 00 00 00 00 00 00 00 01 02 04 08 10 20 40 80@.
0x0100	00 C3 50 01 CE ED 66 66 CC 0D 00 0B 03 73 00 83	..P...ff.....s..
0x0110	00 0C 00 0D 00 08 11 1F 88 89 00 0E DC CC 6E E6n.
0x0120	DD DD D9 99 BB BB 67 63 6E 0E EC CC DD DC 99 9Fgcn.....
0x0130	BB B9 33 3E 5A 45 4C 44 41 20 44 49 4E 00 00 41	..3>ZELDA DIN..A
0x0140	5A 37 45 C0 30 31 00 1B 05 02 01 33 00 EE FE 8F	Z7E.01.....3....
0x0150	00 F3 FE 11 3E 00 20 07 3C CB 40 28 02 3E FF E0>. .<.@(>..
0x0160	96 3E 37 E0 94 3E 0D E0 95 31 10 C1 3E 03 E0 97	.>7...>...1...>...
0x0170	EA 22 22 C3 00 40 06 00 87 30 01 04 B7 20 F9 78	."..@...0... .x
0x0180	C9 7E 81 27 22 7E 88 27 32 D0 3E 63 22 32 C9 7E	~.'~.'2.>c"2.~
0x0190	91 27 22 7E 98 27 32 D0 AF 22 32 37 C9 1E 08 06	.'~.'2.."27....
0x01A0	00 68 60 29 87 30 01 09 1D 20 F8 C9 CB 37 47 E6	.h`.0... ..7G.
0x01B0	F0 4F 78 E6 0F 47 C9 CB 37 0F 47 E6 F8 4F 78 E6	.0x..G..7.G..0x.
0x01C0	07 47 C9 06 00 87 CB 10 87 CB 10 4F C9 06 FF CB	.G.....0....

5.4 Disassembly

The Disassembly tab shows instructions and allows the user to step through game code in real-time. Below are descriptions of all of the functions in this tab:

The screenshot shows the Disassembly tab of a debugger. The instruction list is as follows:

0x0040	0xF5 --> PUSH AF
0x0041	0xC5 --> PUSH BC
0x0042	0xD5 --> PUSH DE
0x0043	0xE5 --> PUSH HL
0x0044	0xC3 --> JP 0x09D4
0x0045	0xD4 --> CALL NC, 0x09
0x0046	0x09 --> ADD HL, BC
0x0047	0x00 --> NOP
0x0048	0xF5 --> PUSH AF
0x0049	0xE5 --> PUSH HL
0x004A	0xC3 --> JP 0x0B22
0x004B	0x22 --> LDI (HL), A
0x004C	0x0B --> DEC BC
0x004D	0x00 --> NOP
0x004E	0x00 --> NOP
0x004F	0x00 --> NOP
0x0050	0xFB --> EI
0x0051	0xF5 --> PUSH AF
0x0052	0xC5 --> PUSH BC
0x0053	0xD5 --> PUSH DE
0x0054	0xE5 --> PUSH HL
0x0055	0xC3 --> JP 0x0CE3
0x0056	0xE3 --> UNK INSTR
0x0057	0x0C --> INC C
0x0058	0xF5 --> PUSH AF
0x0059	0xC3 --> JP 0x0C19
0x005A	0x19 --> ADD HL, DE
0x005B	0x0C --> INC C
0x005C	0x00 --> NOP

Register window values:

- A: 0xC7 F: 0xC0
- B: 0x00 C: 0x1C
- D: 0xC4 E: 0x9D
- H: 0xC4 L: 0x9D
- SP: 0xC10E
- PC: 0x40
- Flags: ZN..

Buttons: Next, Continue, Set Breakpoint, Clear Breakpoints, Reset, Reset + Run, Refresh, Close.

Next:

Jumps to the next instruction and waits for further input. When running in **Continue** mode, this will pause the emulator and resume debugging.

Continue:

Runs the emulator normally until it hits a breakpoint or the **Next** button is pressed. In either case, the emulator will pause and resume debugging.

Set Breakpoint:

Creates a break point on the current PC. Any breakpoints are highlighted in yellow. A breakpoint will force the emulator to stop when running in **Continue** mode.

Clear Breakpoints:

Erases all existing breakpoints from the debugger.

Reset:

Resets emulation back to the beginning and pauses.

Reset + Run:

Resets emulation back to the beginning and runs in **Continue** mode.

5.5 OBJ Tiles

The OBJ Tiles tab displays all of the sprites currently in use by a game. Each sprite can be clicked on, and then GBE+ will display an enlarged preview on the right-hand side along with detailed information about the sprite's OAM data.



The screenshot shows the OBJ Tiles tab in the GBE+ interface. The interface has a tabbed menu at the top with the following tabs: I/O, Palettes, Memory, Disassembly, OBJ Tiles (selected), and BG Tiles. The main area displays a grid of 80 small sprite tiles arranged in 5 rows and 8 columns. The first three rows contain various patterns of green and blue pixels. The fourth row contains the letters 'P', 'R', 'E', 'S', 'S', 'S' in yellow on a blue background. The fifth row contains the letters 'T', 'A', 'R', 'T' in yellow on a blue background, followed by five blue tiles with a small yellow character. To the right of the grid is a larger, detailed preview of a green and blue pixelated character. Below the preview, the following OAM data is displayed:

- X: 114 Y: 65
- Tile #: 86
- Tile Address: 0x8560
- Tile Size: 8x16
- H-Flip : N V-Flip : N
- Tile Palette : OCP4
- VRAM Bank: 0

At the bottom right of the window, there are two buttons: 'Refresh' and 'Close'.

5.6 BG Tiles

The BG Tiles tab displays the background tiles currently in use by a game. Each BG tile can be clicked on, and then GBE+ will displayed an enlarged preview on the right-hand side. Please note that this feature is not complete at this time and may experience issues for some GBC games.



5.7 The Command-Line Debugger

GBE+ offers another debugger for the SDL version. Although not as powerful as the Qt GUI version, it offers many of the features found in the disassembler such as breakpoints and viewing instructions. To enter into debugging mode use the **-d** or **--debug** arguments when running GBE+ from the command-line.

GBE+ will start off paused, and from the command-line, debugging commands are entered. GBE+ will print information to the command-line console as necessary. The DMG and GBA cores both have this debugging functionality, unlike the Qt version which currently only works with DMG and GBC games. The debugging commands vary slightly between the DMG and GBA cores. Below are the all the commands for the DMG core:

- n:**
Run next Fetch-Decode-Execute stage.
- c:**
Continue until next breakpoint.
- bp:**
Set breakpoint, format 0x1234ABCD.
- bc:**
Set breakpoint when a specific memory address equals a certain value. format 0x1234 for addr, 0x12 for value.
- bw:**
Set breakpoint when any value is written to the specified address, format 0x1234. Only available when setting the **ADVANCED_DEBUG** CMake flag when building GBE+ from source code.
- br:**
Set breakpoint when any value is read from the specified address, format 0x1234. Only available when setting the **ADVANCED_DEBUG** CMake flag when building GBE+ from source code.
- del:**
Delete all current breakpoints.
- u8:**
Show BYTE @ memory, format 0x1234.
- u16:**
Show WORD @ memory, format 0x1234.
- w8:**
Write BYTE @ memory, format 0x1234. Prompts for address first, then value to write.
- w16:**
Write HALFWORD @ memory, format 0x1234. Prompts for address first, then value to write.

- reg:** Alters the value of a register. Prompts for an index (0 - 9) first, then the value. Index values of 0 - 6 correspond to registers A - L. Index values of 7 - 9 correspond to the CPU Flags, SP, and PC registers respectively.
- rom:** Displays current MBC ROM bank, if any.
- ram:** Displays current MBC RAM bank, if any.
- dz:** Disassembles 16 GBZ80 instructions from the specified address, format 0x1234.
- dq:** Quit the debugger, return to normal emulation.
- dc:** Toggle CPU cycle display.
- cr:** Reset CPU cycle counter.
- rs:** Reset emulation.
- pa:** Toggles printing all instructions to the screen.
- pc:** Toggles printing the all Program Counter values to the screen.
- ls:** Loads a given save state (0-9)
- ss:** Saves a given save state (0-9)
- q:** Quit GBE+.
- h:** Display the above help messages.

Below are all the commands for the GBA core:

- n:**
Run next Fetch-Decode-Execute stage.
- c:**
Continue until next breakpoint.
- bp:**
Set breakpoint, format 0x1234ABCD.
- bc:**
Set breakpoint when a specific memory address equals a certain value. format 0x1234 for addr, 0x12 for value.
- del:**
Delete all current breakpoints.
- u8:**
Show BYTE @ memory, format 0x1234ABCD.
- u16:**
Show HALFWORD @ memory, format 0x1234ABCD.
- u32:**
Show WORD @ memory, format 0x1234ABCD.
- w8:**
Write BYTE @ memory, format 0x1234ABCD. Prompts for address first, then value to write.
- w16:**
Write HALFWORD @ memory, format 0x1234ABCD. Prompts for address first, then value to write.
- w32:**
Write WORD @ memory, format 0x1234ABCD. Prompts for address first, then value to write.
- reg:**
Alters the value of a register. Prompts for an index (0 - 36) first, then the value. Index values 0 - 15 correspond to r0 - r15 (User Mode), 16 for CPSR, and 17 - 36 the rest of the registers for FIQ, SVC, ABT, IRQ, and UND modes
- dq:**
Quit the debugger, return to normal emulation.
- dc:**
Toggle CPU cycle display.
- cr:**
Reset CPU cycle counter.

- rs:** Reset emulation.
- pa:** Toggles printing all instructions to the screen.
- pc:** Toggles printing the all Program Counter values to the screen.
- ls:** Loads a given save state (0-9).
- ss:** Saves a given save state (0-9).
- q:** Quit GBE+.
- h:** Display the above help messages.

Below are all the commands for the NDS core:

- n:**
Run next Fetch-Decode-Execute stage.
- c:**
Continue until next breakpoint.
- sc:**
Switch current CPU core to debug (NDS7 or NDS9).
- bp:**
Set breakpoint, format 0x1234ABCD.
- bc:**
Set breakpoint when a specific memory address equals a certain value. format 0x1234 for addr, 0x12 for value.
- del:**
Delete all current breakpoints.
- u8:**
Show BYTE @ memory, format 0x1234ABCD.
- u16:**
Show HALFWORD @ memory, format 0x1234ABCD.
- u32:**
Show WORD @ memory, format 0x1234ABCD.
- w8:**
Write BYTE @ memory, format 0x1234ABCD. Prompts for address first, then value to write.
- w16:**
Write HALFWORD @ memory, format 0x1234ABCD. Prompts for address first, then value to write.
- w32:**
Write WORD @ memory, format 0x1234ABCD. Prompts for address first, then value to write.
- dq:**
Quit the debugger, return to normal emulation.
- da:**
Disassembles 16 ARM instructions from the specified address, format 0x1234ABCD.
- dt:**
Disassembles 16 THUMB instructions from the specified address, format 0x1234ABCD.

rs:

Reset emulation.

pa:

Toggles printing all instructions to the screen.

pc:

Toggles printing the all Program Counter values to the screen.

q:

Quit GBE+.

h:

Display the above help messages.

6.1 Netplay Guide

GBE+ supports netplay for the DMG and GBC systems only. GBA and SGB support are not functional at this time. Standard linking between two Game Boys is emulated. The 4-player adapter - used for games like Wave Race or Faceball 2000 - is also supported (see **Section 6.2** for more details). Additionally the GBC's infrared port is emulated with only a few minor incompatibilities (see **Section 7** for more details). This section will go over the basics of configuring netplay.

First, gather the necessary network information. Each player must have the IPv4 address of the other to correctly connect. If netplay will only involve one machine (connecting two instances of GBE+ on the same computer), only that machine's current IPv4 address is needed. In that case, it is recommended that the **localhost** address (127.0.0.1) be used.

Next, all players must agree to use the same ports. Player A's Server Port **must match** the Player B's Client Port, Player A's Client Port **must match** Player B's Server Port. Effectively, only 2 ports are necessary, but they must be configured correctly like such:

Player A		Player B	
Server Port	2000	Server Port	2001
Client Port	2001	Client Port	2000

Additionally, both players must agree whether or not to turn on hard syncing. For most network connections outside of the same machine, the latency is far too high and will desync netplay without hard syncing. Although it is very slow, hard syncing is recommended. Due note that the real-world latency outside of LAN or small networks makes hard syncing unbearably slow, so netplay across the internet is a bit impractical. For local networks, however, the speed should be fine.

To start netplay, the emulated SIO device should be set to GB Link Cable for both players for linked gameplay. For the GBC IR port, this step is unnecessary. Each player must then start a netplay session while running the game by pressing hotkey F5. GBE+ will pause for 10 seconds and wait to establish a connection. If no connection is detected, GBE+ will unpauses and users can try again. Once connected, both instances of GBE+ will continuously communicate with each other. To disconnect from a netplay session, press the F6 hotkey. After any disconnection (by hotkey or network error) GBE+ must restart the game to properly reconnect. Currently there are no GUI indications of netplay connection status, but there are messages printed to the terminal when launch the SDL or Qt version from the command-line.

6.2 DMG-07 Setup

To emulate the DMG-07, first make sure every instance of GBE+ has selected **DMG-07** as the option in **Serial IO Device (Link Cable)** from General Settings (see **Section 3.1** for more details). After that, adjust the netplay settings as described in the previous section. The only difference now is that 3 or 4 players must now be configured as such:

Player A		Player B		Player C		Player D	
Server Port	2000	Server Port	2001	Server Port	2001	Server Port	2001
Client Port	2001	Client Port	2000	Client Port	2000	Client Port	2000

Player A's Server Port **must match** the other players' Client Ports, and Player A's Client Port **must match** the other players' Server Port. Player A's Server Port must be lower than any other players in order for GBE+ to detect it as the DMG-07 host. Additionally, ensure that all instances of GBE+ have access to following ports:

Player A Server Port

Player A Server Port + 2

Player A Server Port + 4

Player A Client Port

Player A Client Port + 2

Player A Client Port + 4

Finally, hard syncing is required for GBE+ to properly emulate DMG-07. Make sure every instance of GBE+ has this options turned on and that they use the same threshold value. **Currently, GBE+ only supports localhost for the DMG-07**, so all instances of GBE+ need to be running on the same IP address, i.e. the same computer.

To begin connecting instances of GBE+, let Player A begin the netplay session by pressing the F5 key. Afterwards, other instances of GBE+ can connect to Player A by pressing F5 as well. Player IDs are assigned based on whoever connects first (e.g. Player B is the 1st to connect, Player C is 2nd, and Player D is 3rd). If any instance of GBE+ disconnects, all others will also exit netplay.

6.3 Net Gate

The Net Gate is a simple protocol that allows an external application to communicate with GBE+ when emulating the Battle Chip Gate, Progress Chip Gate, or Beast Link Gate accessories. GBE+ will wait on a network to receive a specific Battle Chip ID to emulate inserting the chip into the slot. The chip is automatically removed after a brief period of time, so the external application only deals with sending data back to the emulator. The protocol is only 3 bytes sent over TCP and is as follows:

Byte 1: 0x80

Byte 2: High-byte of the 16-bit Chip ID

Byte 3: Low-byte of the 16-bit Chip ID

It is important to note that GBE+ **will not return a TCP response**. The application should essentially “fire and forget”. The application must connect to GBE+ on the emulator’s server port. See **Section 3.5** and **6.2** for more information on GBE+’s server port. The Net Gate option must be enabled, and one of the Battle Chip Gate models must be selected as the Emulated Serial IO Device.

Besides connecting via GBE+ through a raw TCP socket, there are no other requirements. The interface used to present and send Battle Chips to GBE+ is up to this external application.

7. Game Specific Instructions

Due to the fact that many Game Boy games have different accessories or utilize hardware in unique ways, some titles require additional configuration in order to run properly. Below, each section details how to play various games correctly or how to emulate edge cases.

7.1 Barcode Boy Games

To emulate the Barcode Boy scanner, set the **Serial IO Device (Link Cable)** from General Settings to **Barcode Boy**. Next, select **File -> Select Card File** from the GBE+ Qt menu, or manually specify the card file in the .ini file (see the [#card_file] field). When running a Barcode Boy compatible game, wait for the game to ask for the player to swipe a card. At this time, press the F3 key, and GBE+ will begin emulating the Barcode Boy.

GBE+ comes with binary card data pre-packaged in the **data/cards/bcb** folder.

7.2 Barcode Taisen Bardigun

To emulate the Barcode Taisen Bardigun scanner, set the **Serial IO Device (Link Cable)** from General Settings to **Bardigun Barcode Scanner**. Next, select **File -> Select Card File** from the GBE+ Qt menu, or manually specify the card file in the .ini file (see the [#card_file] field). When running Barcode Taisen Bardigun, wait for the game to ask for the player to swipe a card. At this time, press the F3 key, and GBE+ will begin emulating the Barcode Taisen Bardigun scanner.

GBE+ comes with binary card data pre-packaged in the **data/cards/btb** folder.

7.3 Battle Chip Gate Games

To emulate the Battle Chip Gate, Progress Chip Gate, or Beast Link Gate, set the **Serial IO Device (Link Cable)** from General Settings to **Battle Chip Gate**, **Progress Chip Gate**, or **Beast Link Gate**. Afterwards, the Battle Chip Gate Controls menu and select the appropriate Chip Gate. Choose up to 4 Battle Chips to map to Context inputs. Be sure to configure Context inputs in the Advanced Controls menu. Pressing any of the Context inputs will emulate inserting a Battle Chip, and releasing any of the Context inputs will emulate extracting a Battle Chip.

7.4 Boktai Games

To emulate the Boktai games, set the **Special ROM Type** in General Settings **AGB - Solar Sensor**. In the Advanced Controls menu, be sure to set up Context Up and Context Down inputs. When running any Boktai game, press Context Up to increase the amount of sunlight, and Context Down to decrease the amount of sunlight.

7.5 Bomberman Max Red/Blue, Mission Impossible

These GBC games use TV remote input on the IR sensor for unlockables or extras. set the **Infrared Device** in General Settings to **TV Remote**. When playing these games, press F3, and GBE+ will emulate random IR signals (note, this is functionally useless for Mission Impossible's TV remote programmer but it still "works").

7.6 Chee Chai Alien

To emulate using a lamp or some other light source in Chee Chai Alien, set the **Infrared Device** in General Settings to **Constant IR Light**. There are two modes of operation which can be chosen by clicking the "Configure" button. Static means the IR sensor always detects light without any user intervention. Interactive lets users control the light by pressing the Context Up button.

7.7 Drill Dozer

To emulate the rumble functionality, set the **Special ROM Type** in General Settings to **AGB -> Rumble**. Additionally, be sure to enable rumble support in the Advanced Controls menu.

7.8 GBC Infrared Games

To emulate most GBC games that use the IR port for GBC-to-GBC communications, set the **Infrared Device** in General Settings to **GBC IR Port**. Make sure netplay settings are properly adjusted for 2-player linking (see **Section 6.1** for more details). **Be sure to enable hard sync and set the Sync threshold to 16**. When playing a game, connect two instances of GBE+ via netplay, then try to initialize IR communications through the game. GBE+ will emulate the IR signals automatically.

Some game require both sides to initialize IR communications around the same time. GBE+ is not equipped to handle ones that require extreme precision. As a result Pokemon TCG (non-Japanese versions) and Bomberman Max Red/Blue are currently incompatible. Other should work as intended.

7.9 Gyogun Tanchiki: Pocket Sonar

To emulate the Pocket Sonar, set the **Special ROM Type** in General Settings to **DMG -> MBC1S**. Next, select **File -> Select Image File** from the GBE+ Qt menu, or manually specify the image file in the .ini file (see the [#image_file] field). This is a 160x96 BMP image of the sonar data that will be displayed when using the cart's sonar features.

The image should only uses 4 colors: #000000, #606060, #C0C0C0, and #FFFFFF. The image should typically be broken into 2 parts: above and below the floor for a body of water. Above the floor, use #606060 to indicate debris or other objects and #FFFFFF for open water. Use #000000 to indicate the floor and #C0C0C0 for spaces underneath it.

Fish can be drawn using #C0C0C0. A simple 8x1 line will suffice. These must be drawn above the floor. The floor should be several pixels thick GBE+ comes with an example image in the **data** folder under **misc**.

7.10 Kirby Tilt ‘n’ Tumble

To emulate Kirby Tilt ‘n’ Tumble, go to the Advanced Controls menu and configure the Context Up, Context Down, Context Left, and Context Right buttons. Each button will move Kirby up, down, left, and right respectively.

7.11 Legendz: Isle of Trial/Sign of Nekrom

To emulate the Soul Doll Adapter, set the **Serial IO Device (Link Cable)** from General Settings to **Soul Doll Adapter**. Next, select **File -> Select Data File** from the GBE+ Qt menu, or manually specify the Soul Doll file in the .ini file (see the [#data_file] field). Custom Soul Doll data will now be loaded and saved to specified file. GBE+ provides blank Soul Doll data for various Legendz in the **data** folder under **bin/soul_dollz**.

To emulate the RTC in Legendz: Isle of Trial/Sign of Nekrom, set the **Special ROM Type** in General Settings to **AGB -> RTC**. The clock can dynamically be adjusted with RTC Offsets from General Settings.

7.12 Mobile Adapter GB Games

To emulate the Mobile Adapter GB, set the **Serial IO Device (Link Cable)** from General Settings to **GB Mobile Adapter**. GBE+ will use an internal server (located in the **data** folder under **gbma**) to emulate Mobile Adapter servers. The following games and features are currently supported.

Hello Kitty Happy House: Receives and email from GBE+ with free furniture.

Mobile Trainer: Small demo of the web browser and receives email from GBE+.

Net de Get: Downloads a small “minigame” demo that soft resets the game.

7.13 Pokemon Gold, Silver, Crystal

To emulate Pokemon G/S/C’s GBC-to-GBC IR communications (for Mystery Gifts), see **Section 7.8** above. To emulate Pokemon Pikachu 2 communications (again, for Mystery Gifts), set the **Infrared Device** in General Settings to **Pokemon Pikachu 2**. Use the “Configure” button to choose the level of watts to receive. When running Pokemon G/S/C, go to the Mystery Gift option and press F3 to receive an item from Pikachu. Make sure the **pokemon_pikachu_db.bin** file exists in the **data** folder.

7.14 Pokemon Ruby, Sapphire, Emerald

To emulate the RTC in Pokemon Ruby, Sapphire, and Emerald, set the **Special ROM Type** in General Settings to **AGB -> RTC**. The clock can dynamically be adjusted with RTC Offsets from General Settings.

7.15 Sakura Taisen GB

To emulate GBC-to-GBC IR communications (for swapping Points), see **Section 7.8** above. To emulate Pocket Sakura communications (for sending points), set the **Infrared Device** in General Settings to **Pocket Sakura**. Use the “Configure” button to choose the level of point to receive. When running Sakura Taisen GB, go to the extra menu where Points from the Pocket Sakura can be received, then press F3 when the game prompts the Pocket Sakura to send IR signals. Make sure the **pocket_sakura_db.bin** file exists in the **data** folder.

7.16 WarioWare: Twisted

To emulate the gyroscope in WarioWare: Twisted, set the **Special ROM Type** in General Settings to **AGB -> Gyro Sensor**. Configure the Context Left and Context Right buttons in the Advanced Controls menu to emulate tilting the GBA left or right. Additionally, be sure to enable rumble support in the Advanced Controls menu as well.

7.17 Yoshi Topsy Turvy/Yoshi's Universal Gravitation

To emulate Yoshi Topsy Turvy/Yoshi's Universal Gravitation, set the **Special ROM Type** in General Settings to **AGB -> Tilt Sensor**. Configure the Context Left and Context Right buttons in the Advanced Controls menu to emulate tilting the GBA left or right.

7.18 Zok Zok Heroes

To emulate Zok Zok Heroes, set the **Infrared Device** to **Full Changer**. Press the “Configure” button to choose the Cosmic Character to transmit. When prompted by the game to use the Full Changer, press F3 to begin sending IR signals.

8.1 General FAQ

Q) Will there be an Android port of GBE+?

A) Unlikely in the foreseeable future. My focus has always been to target “desktop” usage. Android just isn’t something I have my sights set on. Having said that, GBE+ runs on very little dependencies outside of SDL, OpenGL, and Qt. It would be fairly easy for someone to port even a basic version of Android and refine it from there. If others work on it, I’d be more than happy to answer any questions and offer advice.

Q) Will there be a Libretro port of GBE+?

A) Unlikely in the foreseeable future. It’s just not something I’m currently interested in. I have a dozen other priorities at the moment (GBA netplay, e-Reader, and NDS support). As I said above, if anyone else is attempting to work on it, I would be more than happy to answer questions about GBE+’s code and give advice.

Q) Will you make Custom Graphics (CGFX) for [insert random game console]?

A) No. I am only interested in the Game Boy, Game Boy Color, Game Boy Advance, and Nintendo DS, and I am only pursuing CGFX on those systems. NES, Genesis, SNES, Wonderswan, etc are not my thing. Learning the ins and outs of [insert random game console] is not easy at any rate; it takes a lot of time. Regardless, I will be publishing my work on CGFX so other emulator developers can implement in themselves.

Q) What about GBE+ and emulation accuracy?

A) For GBE+, accuracy is a long-term sub-goal. This means that accuracy gets handled as something incremental, but things like cycle or per-pixel accuracy are not the main focus of this project. That is not to say GBE+ does not care about accuracy. Eventually, these things should be pursued, however other features have priority.

Q) But... accuracy!

A) That’s not a question ;)

As said above, accuracy will be tackled in due time, step-by-step. General game compatibility is very important, but extreme, detailed hardware accuracy comes at its own pace.

Q) What is the general state of emulation in GBE+?

A) As of 1.3, support for DMG and GBC games is very high. Only a few dozen games on each system still exhibit graphical or music errors or freezes. The vast majority will play without issue as the DMG/GBC core is quite mature at this point.

The situation on the GBA is better than 1.0. Most games will play fine, notwithstanding the occasional non-game-breaking graphical glitch. The GBA core still needs more work done to it, especially with regards to speed. Overall, the core itself is stable and acceptable to play many games from start to finish, but GBE+ probably won’t replace your default GBA emulator any time soon.

The NDS core is still under constant development. It runs many homebrew, and boots a few commercial titles.

Q) Will Custom Graphics (CGFX) come to other systems supported by GBE+?

A) Yes. The current plan is to rewrite most of the GBA LCD rendering code to be faster, and then attempt to do CGFX on the GBA. In the future, GBE+ aims to emulate the NDS as well. CGFX should be applicable to the NDS as well, at least for the 2D LCD engine (which is strikingly similar in many ways to the GBA). Having said that, getting CGFX running on the GBA core will not be an easy task and is going to take some time.

Q) When will the next version of GBE+ come out?

A) When it's done, and not a moment sooner. April 1st of every year.

Q) Why are there so many .dll files in the GBE+ 1.3 download?

A) I have not yet found a good way to get mingw and CMake to statically link some libraries. This is necessary to reduce the number of .dll files (since that code will be put into the .exe files instead).

Q) Why doesn't my DS3 controller work in GBE+?

A) This is a known issue with various driver reports input from the controller to SDL. On Linux, the DS3 seems to indicate that everything is pressed instead of an individual joystick or button. For those curious, this affects other emulators such as PPSSPP. It's something either SDL or the driver will have to fix.

Q) Will GBE+ support Vulkan?

A) Possibly in the future, yes, but at this time there's no need to rush to Vulkan. GBE+ currently only uses OpenGL to draw a single texture on-screen, and optionally add some shaders. Vulkan won't improve much in that regard. Vulkan would be a good candidate once a 3D hardware renderer for the NDS core is implemented, but that is probably years and years down the road.

Q) Where can I find ROMs online?

A) lol, no.

8.2 GBE+ Road Map

Here is a rough roadmap of where GBE+ is heading from 1.1 to the expected 2.0 version. Note that various other versions may be released before 2.0, and the goals here may change.

- * Improve the Qt UI to list games with user specified icons.
- * Improve the NDS core to a reasonable level of maturity and game compatibility.
- * Refactor memory and LCD emulation in GBA core to improve speed.
- * Further improve DMG/GBC game compatibility.
- * Add GBA full netplay support.
- * Add e-Reader support.
- * Add support for more obscure Game Boy hardware.

9. Credits & Acknowledgments

ADormant - For issue reporting and some feature requests.

AnyOldName3 - For initial suggestion of CGFX as a feature I should look into.

AWJ - For helping solve MBC1M emulation.

benderscruffy - For extensive compatibility testing. Great job!

BlaXunSlime - Inspiration for the Net Gate.

bunnei (from the Citra dev team) - For being something of a mentor, colleague, and in general a cool guy.

byuu (higan author) - For bringing up several interesting edge cases about GB emulation.

CaptainCaffeine (chroma author) - Fixes for the GUI debugger + helpful tips on DMG bugs.

endrift (mGBA author) - General contributions to getting accurate GB and GBA emulation. MBC6 and Battle Chip Gate research.

exophase (Drastic author) - For being encouraging + helping me with the GBA VBlankIntrWait SWI.

frozenLake - CGFX testing and issue reporting.

Game Boy Online Restoration Project - Lots of info + collaboration + research on the Mobile Adapter GB.

gbdev.gg8.se - Great dev community.

gekkio (mooneye-gb author) - General contributions to getting accurate GB emulation, MBC1M emulation.

GregoryMcGregerson - For being an awesome tester! Some CGFX stuff would still be broken if not for him.

Háčky - Initial documentation of the Mobile Adapter GB + Pokemon Crystal servers.

JappaWakka - CGFX testing and issue reporting.

LJI32 (SameBoy author) - General contributions to accurate GB emulation.

lioncash (Dolphin and Citra dev) - Code contributions to GBE+.

mars - Battle Chip Gate research.

NESdev - Great dev community.

PSISP - (CorgiDS/DobieStation author) - Tips on NDS emulation + great blog entries.

/r/emulation and /r/emudev - The defacto emulation and emudev communities. 'Nuff said.

[/r/GameBoy](#) - Hey guys!

[sczther](#) - Compatibility input on some tough to emulate games.

[StapleButter](#) - (melonDS author) - Tips on NDS emulation + great blog entries.