Updated 18-November-2014

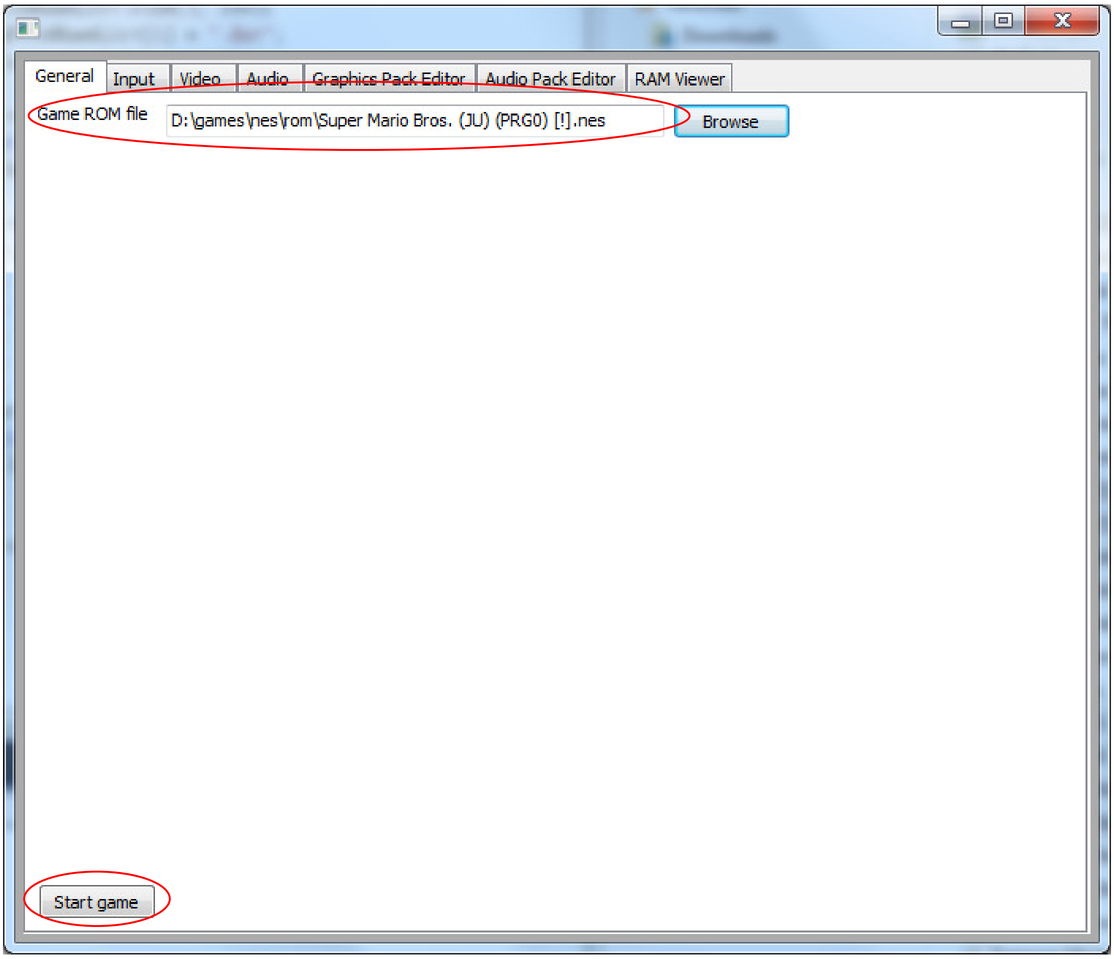
Index

About HDNes

HDNes is an emulator of the Nintendo Entertainment System with the ability of replacing graphics of the game with custom-made graphics pack and it comes with an editor for creating them. Recently the ability of replacing music is also added but this feature requires the user to have in-depth knowledge of the game. Currently, only mapper 000-004,007,009,010, 016, 153, 159 are supported at the moment. And the emulator requires graphics card glsl version 330 or above to run.

Running a game

To start a game, choose a ROM file and press the “Start game” button. To end the game, press close button or press the key assigned. The default is “ESC”.

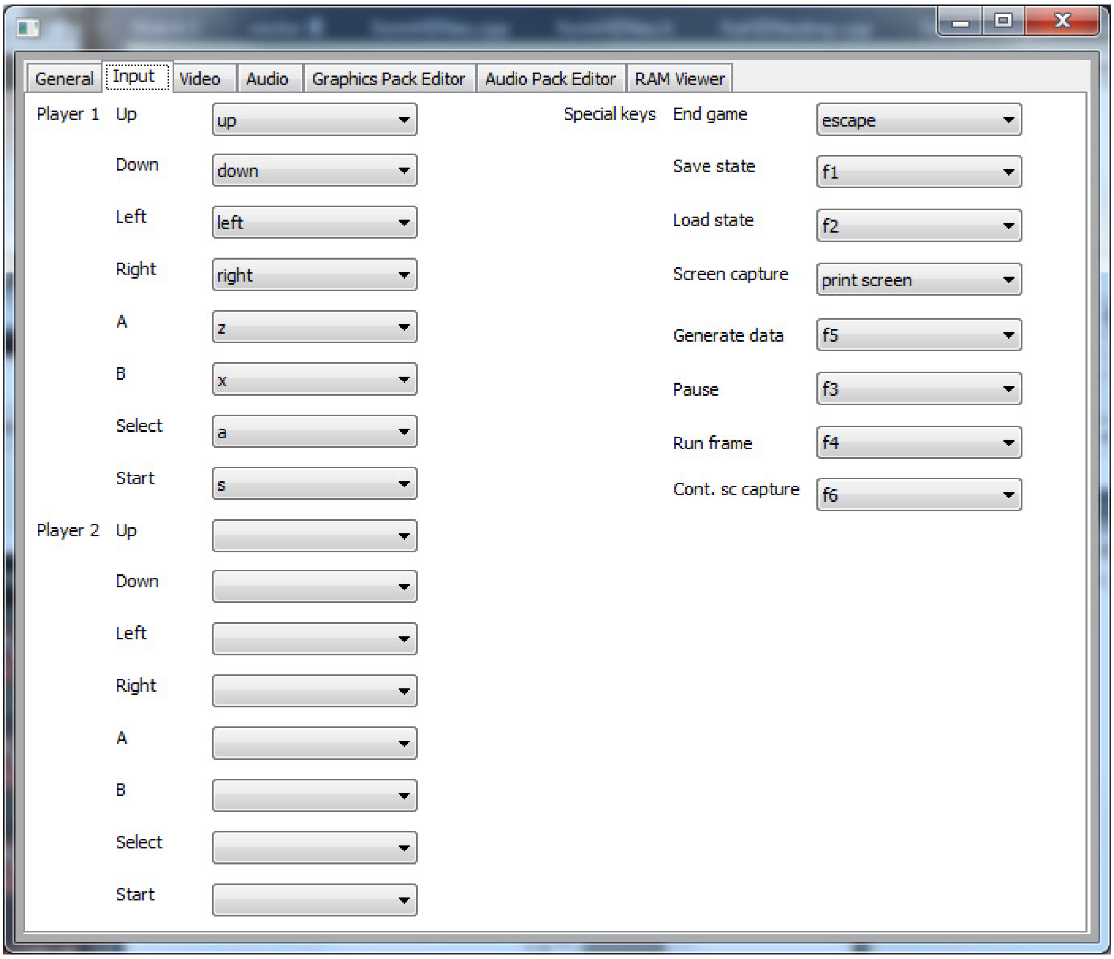


Game ROM file D:\games\nes\rom\Super Mario Bros. (JU) (PRG0) [!].nes

Options

Key assignments

To change a key assignment, goto the “Input” tab and use the drop-down box to assign a new key.



Player 1
Up: up
Down: down
Left: left
Right: right
A: z
B: x
Select: a
Start: s

Player 2
Up:
Down:
Left:
Right:
A:
B:
Select:
Start:

Special keys
End game: escape
Save state: f1
Load state: f2
Screen capture: print screen
Generate data: f5
Pause: f3
Run frame: f4
Cont. sc capture: f6

There are several special keys and they are as follows:

a) “End game” this will quit the game that is currently running.

b) “Save state” this will mark that point of time which the player can return to at a later time. Notice that the battery backup will also revert to that point.

c) “Load state” return the game to a previous marked point.

d) “Screen capture” save the current game screen into a file in PNG format. The file is stored in the “screen” folder inside the HDNes folder.

e) “Generate data” check the current screen for new tiles. See the “Graphics pack editor” section for more detail.

f) “Pause” Pause the game or resume a paused game. This will as update the RAM Viewer.

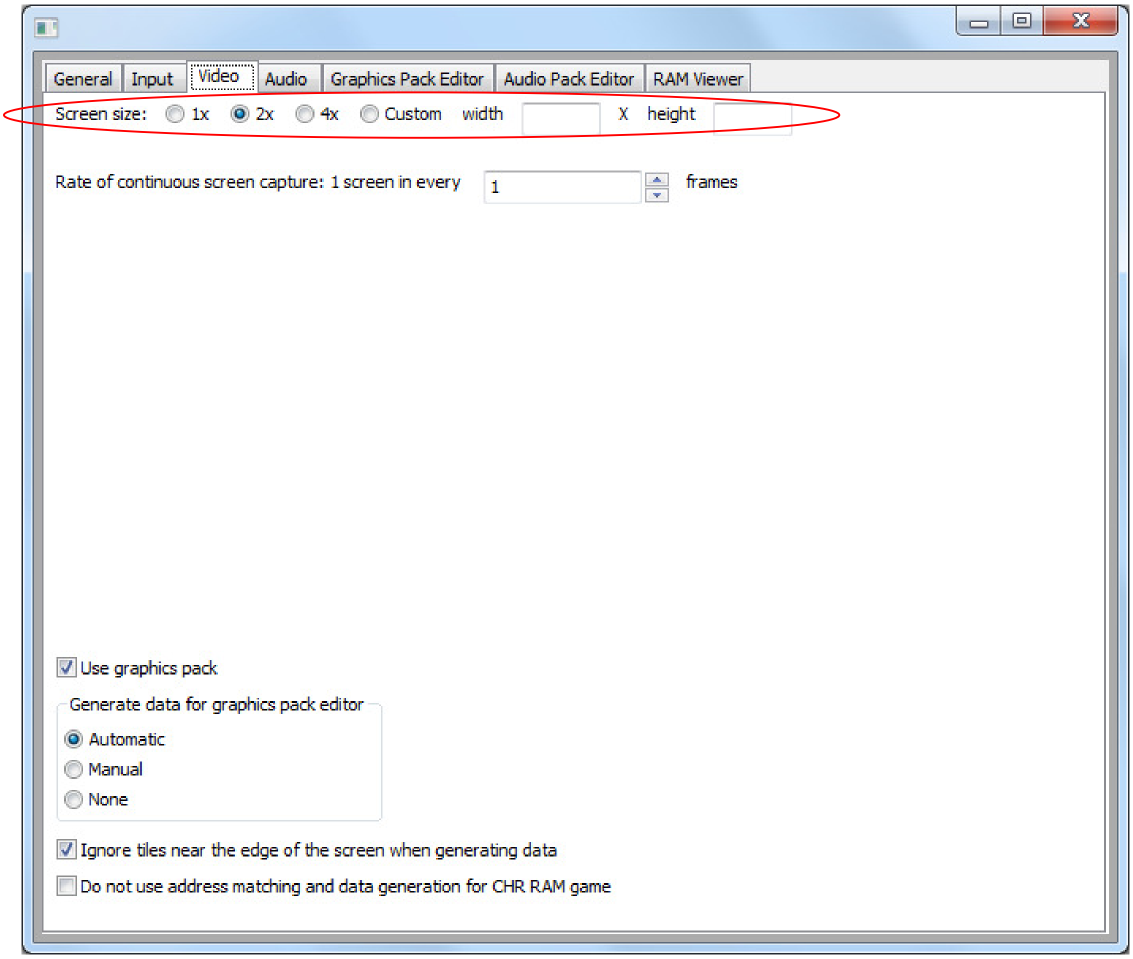
g) “Run frame” Advance the game by 1 frame and pause the game. Use the

"Pause" key to resume the game.

h) "Cont. sc capture" Start capturing the screen using a defined rate and the basic screen size (256 x 240). The emulator runs game at 60 frames per second. See video options for more information.

Video options

The basic screen size is 256 x 240. "2x" is 512 x 480 and "4x" is 1024 x 960. The user can choose to use a custom screen size, but if the screen size is not a multiple of the basic screen size, the game screen will have glitches.



Screen size: ○ 1x ● 2x ○ 4x ○ Custom width X height

Rate of continuous screen capture: 1 screen in every 1 frames

☑ Use graphics pack

Generate data for graphics pack editor

● Automatic

○ Manual

○ None

☑ Ignore tiles near the edge of the screen when generating data

☐ Do not use address matching and data generation for CHR RAM game

The rate of continuous screen capture can also be changed here. The value here acts as a divider to the video frames. A value of 5 means the emulator will capture the screen once every 5 frames, generating 12 frames per second. The value entered must be between 1 and 999.

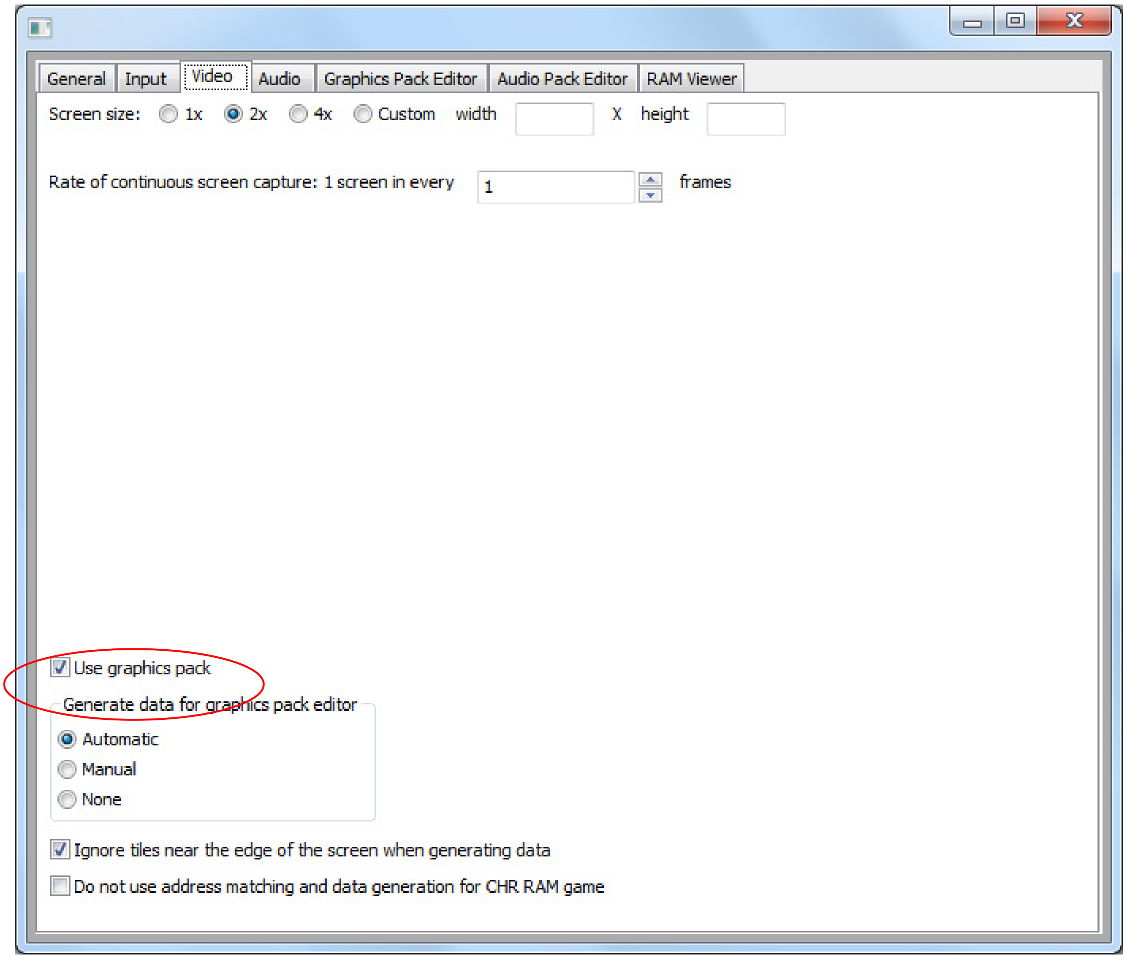
Graphics pack

The concept

A game screen is composed of square tiles of 8 x 8 in size and the tiles are defined by a pattern and a colour palette. Many games store the patterns inside a section (known as CHR ROM) within the game ROM and is read directly by the system. So by knowing the location of the a pattern being used inside the ROM and the colour palette being used to render it, it is possible to uniquely identify a tile. In a nutshell, a graphics pack is a list of tiles and what graphics are to be used to replace the tiles.

Usage

If the "Use graphics pack" option is checked, the emulator will look for graphics pack data inside a subfolder with the same name as the game ROM file under the ROM folder. The emulator will apply the pack to the game automatically.



Screen size: ○ 1x ● 2x ○ 4x ○ Custom width X height

Rate of continuous screen capture: 1 screen in every 1 frames

☑ Use graphics pack

Generate data for graphics pack editor

● Automatic

○ Manual

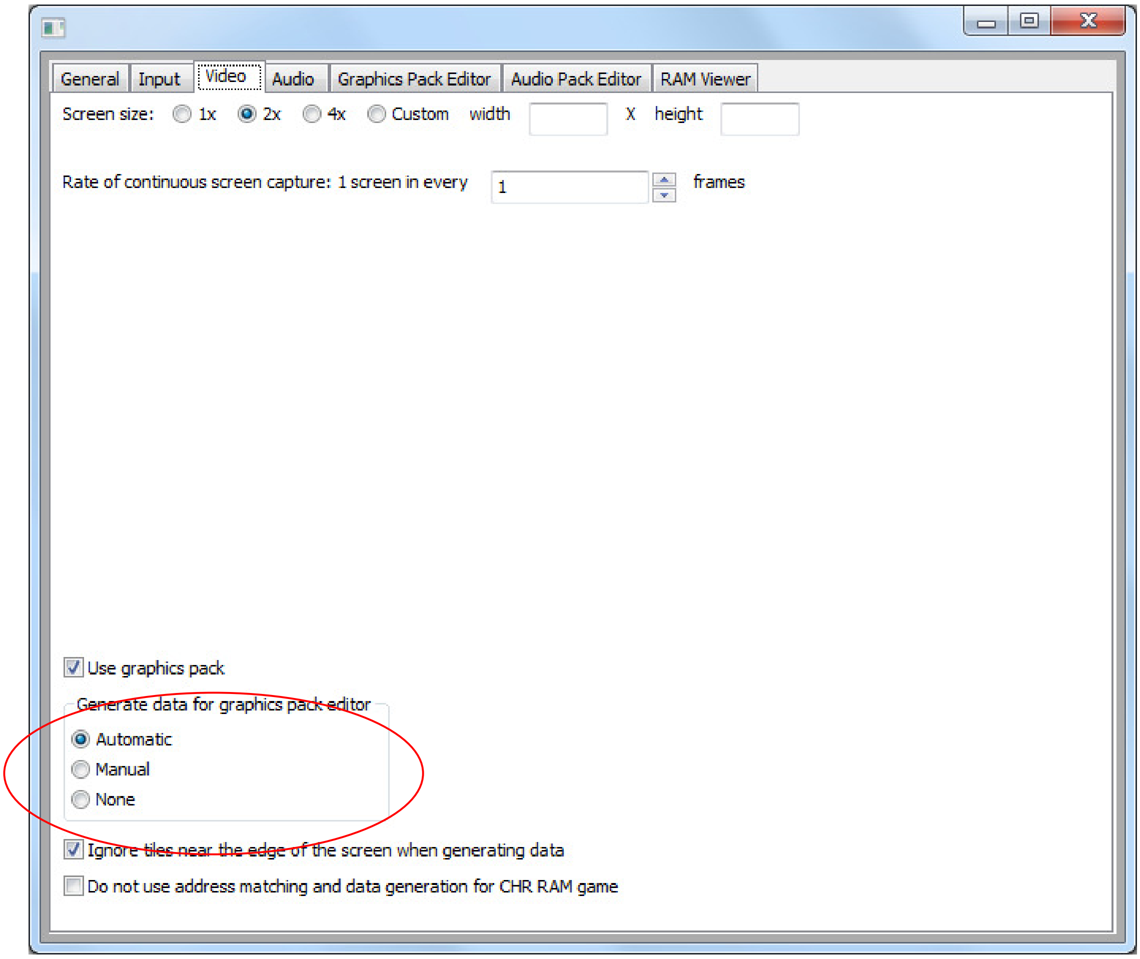
○ None

☑ Ignore tiles near the edge of the screen when generating data

☐ Do not use address matching and data generation for CHR RAM game

Making a graphics pack

To create a graphics pack, the emulator must first run the game and collect a list of tiles being used. The emulator can either collect the data automatically or manually. When “Automatic” is selected, the emulator will check each frame for new tiles and add them into a list along with a screen shot. This option may cause the emulator to slow down. When “Manual” is selected, the emulator will only perform the checking when the user presses the “Generate data” key. The collected data is stored inside a subfolder with the same name as the ROM file under the “edit” folder within the same folder as the emulator.



Screen size: ○ 1x ● 2x ○ 4x ○ Custom width X height

Rate of continuous screen capture: 1 screen in every 1 frames

☑ Use graphics pack

Generate data for graphics pack editor

● Automatic

○ Manual

○ None

☑ Ignore tiles near the edge of the screen when generating data

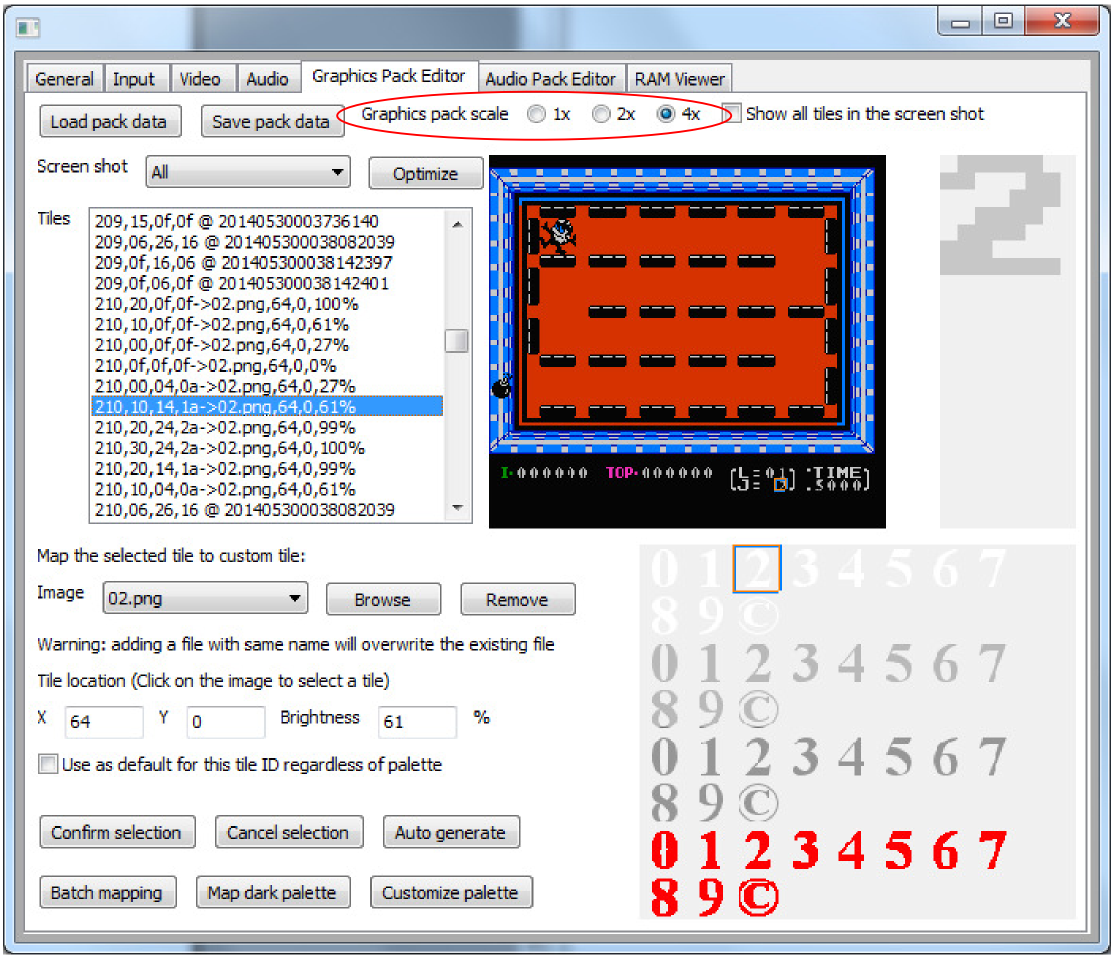
☐ Do not use address matching and data generation for CHR RAM game

The next option is to select whether the tile must be completely in view before being considered by the emulator. It will be difficult to identify a tile from a screen shot if it is only partially in view and many games have junk tiles along the edge of the screen. So selecting this option is recommended.

Lastly, for games which do not use CHR ROM, they write the patterns into RAM (known as CHR RAM) and use that to render the screen. Some of these games still store the pattern inside the game ROM and they copy them into the CHR RAM. In that case it is possible to use graphics pack for these games too and the last option needs to be unselected. For these games, the emulator will also store the whole tile pattern and compare it as well as the colour palette.

Using the Graphics Pack Editor

After running the game long enough to collect the tile data, the first step is to choose a scale for the graphics pack. The scale determines the size of the tiles in the pack. 4x means the tiles are 32 x 32 pixels both the width and height - 4 times the original 8 x 8 in both dimensions. On the other hand, if a pack already existed, the emulator will load it automatically as it runs the game or the data can be loaded manually.



Graphics pack scale ○ 1x ○ 2x ● 4x

☐ Show all tiles in the screen shot

Screen shot: All

Tiles

```
209,15,0f,0f @ 20140530003736140
209,06,26,16 @ 20140530038082039
209,0f,16,06 @ 20140530038142397
209,0f,06,0f @ 20140530038142401
210,20,0f,0f->02.png,64,0,100%
210,10,0f,0f->02.png,64,0,61%
210,00,0f,0f->02.png,64,0,27%
210,0f,0f,0f->02.png,64,0,0%
210,00,04,0a->02.png,64,0,27%
210,10,14,1a->02.png,64,0,61%
210,20,24,2a->02.png,64,0,99%
210,30,24,2a->02.png,64,0,100%
210,20,14,1a->02.png,64,0,99%
210,10,04,0a->02.png,64,0,61%
210,06,26,16 @ 20140530038082039
```

Map the selected tile to custom tile:

Image: 02.png

Warning: adding a file with same name will overwrite the existing file

Tile location (Click on the image to select a tile)

X: 64 Y: 0 Brightness: 61 %

☐ Use as default for this tile ID regardless of palette

The editor lists all collected screen shots in the dropdown box. When a screen shot is selected, the screen shot is displayed on the top right area. And the tiles, which appear on this screen, (for the first time or not depends on the value of the checkbox on the top right of the editor) are listed below the dropdown box. The tiles are listed as 4 numbers: the pattern address and the ID of the 3 colours used in the palette. When a tile is selected in the list, the editor displays a box around the location of that tile on the screen shot and shows an enlarged version on the right.

Often, when a large object moves into the screen, the emulation will generate one screen shot per row of tiles as they appear. But only the last screen shot is needed because it contains all the tiles which the large object is consisted of. The remove those redundant screen shots, use the “Optimize” button next to the dropdown box, and the emulator will search for screen shots which have all their listed tiles available in other screen shots and remove them from the screen shot dropdown box.

Replacement tiles

Below the list of tiles, is a dropdown box containing the list of image files, which hold the replacement tiles, currently inside the graphics pack. To add new files to the pack, use the “Browse” button to select a file. The file will be copied to the pack folder and added to the dropdown box. Then when a file is selected with the dropdown box, the image is displayed on the bottom right area. Use the “Remove” button the remove it from the list.

To assign a replacement tiles, select a screen shot and a tile from the list. Then select an image from the dropdown box. Select the replacement tile by clicking it on the image. The editor assumes the tiles are located on a grid and snaps the mouse click to the grid. If the tiles are not aligned to the grid, input the x y coordinate of the top right corner of the tile manually in the two text boxes. A brightness value can be entered here to darken the replacement tile. Press “Confirm selection” to add the assignment to the list. A replacement tile is listed as 3 numbers: the index of the image file and the x y coordinate of the tile. In some cases, it is simpler to have a replacement tile to act as the default for a particular pattern. To do this, select the check box below before adding the assignment. Be sure to use the “Save pack data” button to save any changes to the pack.

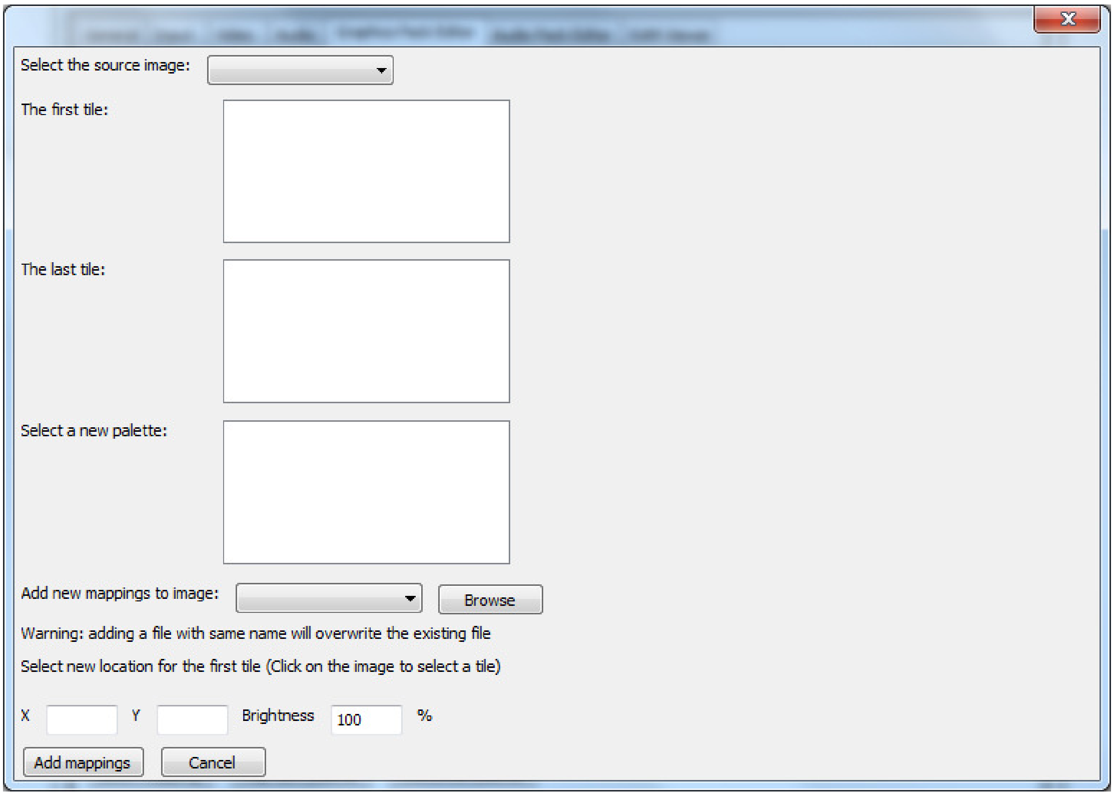
Automatically generate image files

The system uses the tiles in two ways. The first is to use them for the background where they are used to cover the screen and do not overlap one another. Sometimes the tiles combine into one bigger picture. In that case, select the screen shot with those tiles and press the “Auto Generate” button, the editor will create an image file with those tiles placed in the same location as in the screen shot and the editor will add the image to the pack with all the tile assignments. The image file can then be edited directly.

Sometimes it is difficult to distinguish the dividing line where two tiles are joined together. And also tiles are used to represent moving objects on the screen known as “sprites”. These sprite tiles can move freely around the screen, overlap other tiles and flip horizontally or vertically. Since the emulator requires the orientation and the edge of the replacement tiles to match those of the original inside the ROM, it is important to have those details on hand. By selecting “All” in the screen shot dropdown box and press the “Auto Generate” button, the editor will create images files with replacement tiles for all unassigned ones along with assignments and add them to the pack. This way, the image files created can be used as reference for those details.

Handling palette swap

Frequently, a game uses the same pattern with different palette at various situations and in some occasions a group of patterns with continuous pattern addresses uses the same set of palettes. In this case, by placing the replacement tiles for those patterns of the same palette inside the same image file and preserving the relative position of the replacement tiles across different palettes, the “Batch mapping” button provides a quick way to copy assignments of one palette to another.



Warning: adding a file with same name will overwrite the existing file

Select new location for the first tile (Click on the image to select a tile)

Brightness 100 %

First, select an image file containing replacement tiles of a tile group. Then pick the group of tiles by selecting the first those tiles, i.e. the tile with the lowest pattern address, and the last tile from the lists below. Next, select the new palette and the image file containing the replacement tiles for the same group with the new palette. Lastly, select the location of the first tile in the image and press the “Add mappings” button to copy the assignments of the first palette to the new palette.

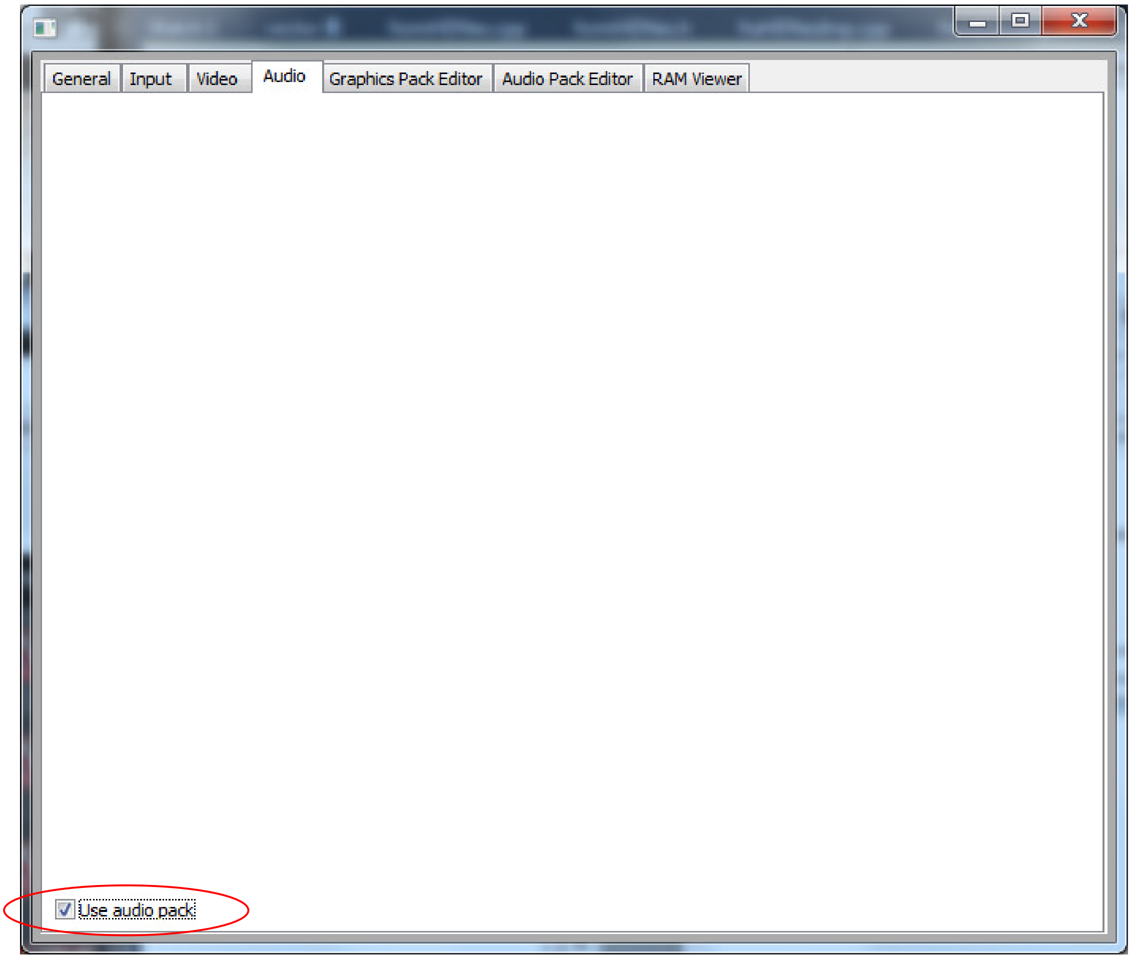
Audio pack

Concept

When the system runs a game, the game continuously feeds audio data to the audio. The design of HDNes assumes that when a game wants to play some audio data, the game will write the ID of that data into a certain RAM address before calling the sound engine to read the audio data and one ID value represents silence. And so by looking for writes to that address and checking the value being written to, the emulator can find out when the game wants to play which audio data. Then the emulator can replace the audio data ID in the write with the silence ID and play a sound file instead.

Usage

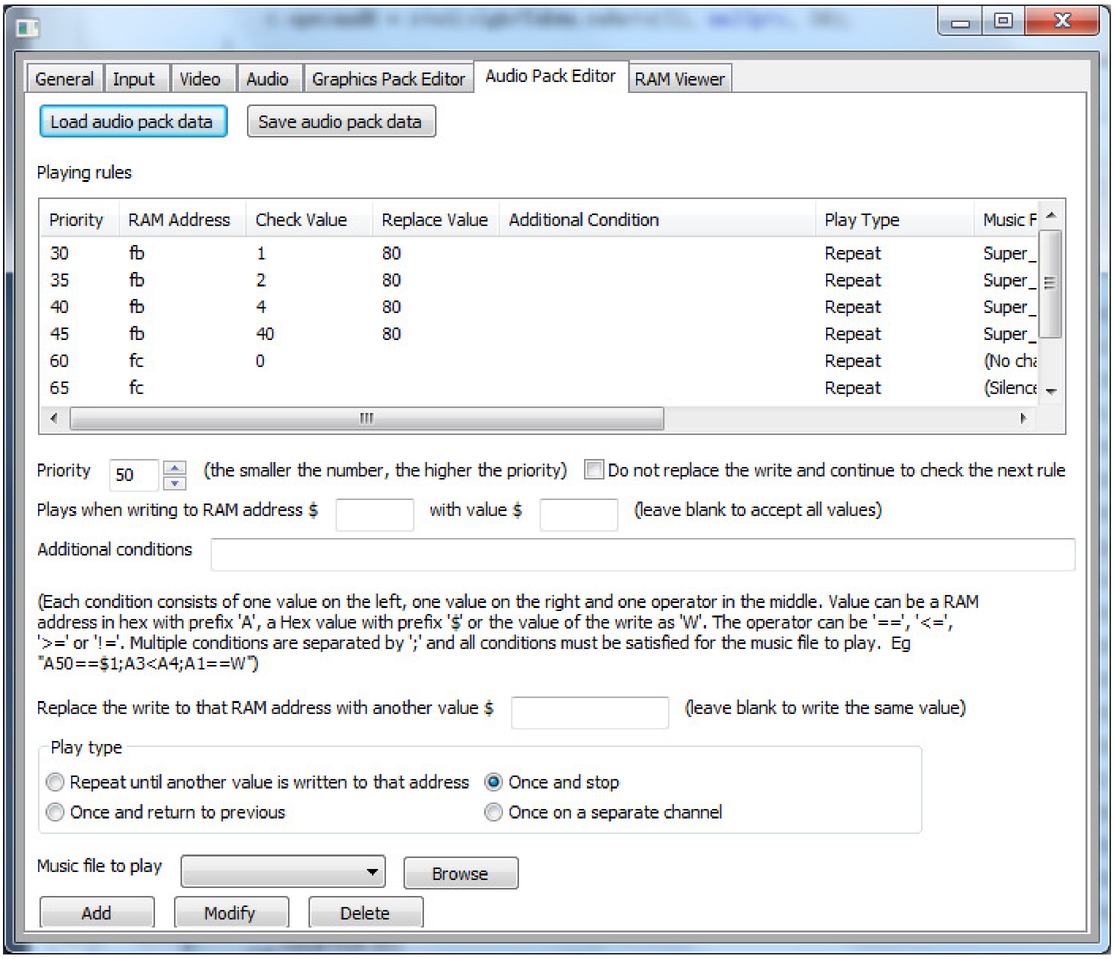
Select the check box at the bottom of the audio page to enable the audio pack. The emulator will look for the audio pack inside a subfolder with the same name as the game ROM file under the ROM folder.



☑ Use audio pack

Using the Audio Pack Editor

The audio pack consists of a list of rules, which are checked every time a write to the RAM is performed. Each rule has a priority number, which determines the order of which the rules are checked. First, the RAM address must matches the one specified by the rule. Then the value of the write must also matches if one is entered. Both the RAM address and the value must be expressed in hexadecimal form.



Playing rules

Priority	RAM Address	Check Value	Replace Value	Additional Condition	Play Type	Music F
30	fb	1	80		Repeat	Super_
35	fb	2	80		Repeat	Super_
40	fb	4	80		Repeat	Super_
45	fb	40	80		Repeat	Super_
60	fc	0			Repeat	(No ch
65	fc				Repeat	(Silenc

Priority 50 (the smaller the number, the higher the priority) ☐ Do not replace the write and continue to check the next rule

Plays when writing to RAM address $ with value $ (leave blank to accept all values)

Additional conditions

(Each condition consists of one value on the left, one value on the right and one operator in the middle. Value can be a RAM address in hex with prefix 'A', a Hex value with prefix '$' or the value of the write as 'W'. The operator can be '==', '<=', '>=' or '!='. Multiple conditions are separated by ';' and all conditions must be satisfied for the music file to play. Eg "A50==$1;A3<A4;A1==W")

Replace the write to that RAM address with another value $ (leave blank to write the same value)

Play type

○ Repeat until another value is written to that address
● Once and stop
○ Once and return to previous
○ Once on a separate channel

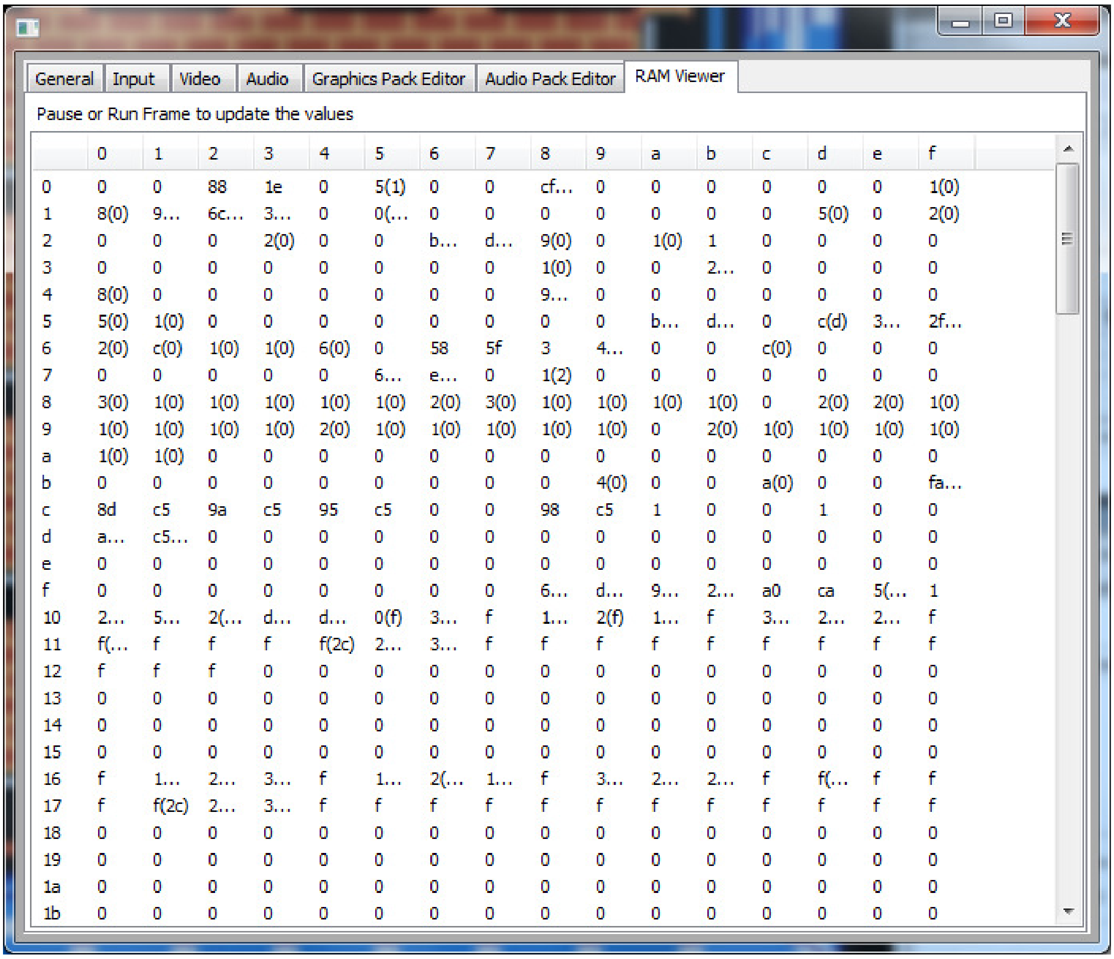
Music file to play

Additional conditions

After this, any additional conditions are checked too. The additional conditions are useful when the game uses more than 1 value to identify the audio data. For example, the game may use another value to change the tempo of the background music. In that case, depending one whether the tempo value or the music ID is written to the RAM first, additional conditions can be used to distinguish which audio file should be played. An additional condition can compare the value of a RAM location with a fixed value, the value of another RAM location or the value of the write. The symbol between the two values determines which type of comparison to use. The "==" symbol means the two value must be equal. The "<=" symbol means the value on the left must be less than or equal to the right. The ">=" symbol means the value on the left must be larger than or equal to the right. The “!=” symbol means the two values must be different.

Resulting actions

If the check succeeds, the emulator will perform several actions. First, if an audio file or “Silence” is selected, and the play type is not “Once on a separate channel”, the emulator will stop playing any audio file. Then it will play an audio file with the selected play type. The audio file must be use the “ogg” format. “Once on a separate channel” means the audio file will be played concurrently with others and is useful for replacing sound effects. Finally it will either move to the next rule or return a replacement value for the write.



Pause or Run Frame to update the values

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	0	0	88	1e	0	5(1)	0	0	cf...	0	0	0	0	0	0	1(0)
1	8(0)	9...	6c...	3...	0	0(...	0	0	0	0	0	0	0	5(0)	0	2(0)
2	0	0	0	2(0)	0	0	b...	d...	9(0)	0	1(0)	1	0	0	0	0
3	0	0	0	0	0	0	0	0	1(0)	0	0	2...	0	0	0	0
4	8(0)	0	0	0	0	0	0	0	9...	0	0	0	0	0	0	0
5	5(0)	1(0)	0	0	0	0	0	0	0	0	b...	d...	0	c(d)	3...	2f...
6	2(0)	c(0)	1(0)	1(0)	6(0)	0	58	5f	3	4...	0	0	c(0)	0	0	0
7	0	0	0	0	0	6...	e...	0	1(2)	0	0	0	0	0	0	0
8	3(0)	1(0)	1(0)	1(0)	1(0)	1(0)	2(0)	3(0)	1(0)	1(0)	1(0)	1(0)	0	2(0)	2(0)	1(0)
9	1(0)	1(0)	1(0)	1(0)	2(0)	1(0)	1(0)	1(0)	1(0)	1(0)	0	2(0)	1(0)	1(0)	1(0)	1(0)
a	1(0)	1(0)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b	0	0	0	0	0	0	0	0	0	4(0)	0	0	a(0)	0	0	fa...
c	8d	c5	9a	c5	95	c5	0	0	98	c5	1	0	0	1	0	0
d	a...	c5...	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f	0	0	0	0	0	0	0	0	6...	d...	9...	2...	a0	ca	5(...	1
10	2...	5...	2(...	d...	d...	0(f)	3...	f	1...	2(f)	1...	f	3...	2...	2...	f
11	f(...	f	f	f	f(2c)	2...	3...	f	f	f	f	f	f	f	f	f
12	f	f	f	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	f	1...	2...	3...	f	1...	2(...	1...	f	3...	2...	2...	f	f(...	f	f
17	f	f(2c)	2...	3...	f	f	f	f	f	f	f	f	f	f	f	f
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1a	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1b	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RAM Viewer

The emulator provides a RAM viewer for checking the values in the RAM. The values are updated when the “Pause” key or the “Run frame” key is pressed. If a value changes between two updates, then the old value is shown next to it. Each row, 16 consecutive addresses are listed. The actual address can be found by combining the number at the left end of the row and the number at the top of the column.

Other uses

Besides using the audio pack for replacing audio data, one side effect is that the writes to the RAM are intercepted by the emulator and in some cases replaced. So it is possible to use it as a cheat code system.

Example:

Super Mario Bros
From "A COMPREHENSIVE SUPER MARIO BROS. DISASSEMBLY" by doppelganger (http://www.romhacking.net/documents/344/), there are two relevant RAM addresses:
AreaMusicQueue = $fb
EventMusicQueue = $fc
And the game uses following values:
Silence = $80
StarPowerMusic = $40
PipeIntroMusic = $20
CloudMusic = $10
CastleMusic = $08
UndergroundMusic = $04
WaterMusic = $02
GroundMusic = $01
TimeRunningOutMusic = $40
EndOfLevelMusic = $20
AltGameOverMusic = $10
EndOfCastleMusic = $08
VictoryMusic = $04
GameOverMusic = $02
DeathMusic = $01

Special thanks

Quietust for helping with EEPROM of Bandai FCG board

rainwarrior for fixing a bug in fragment shaders