

G2D Library

Release Notes

Version 1.2.3

**The contents in this document are highly
confidential and should be handled accordingly.**

Table of Contents

1	The G2D Library	6
1.1	The Runtime Library	6
1.2	Multithread Operations	6
1.3	The Binary Converter	6
2	Major Revisions	7
2.1	Changes from Version 1.2.2	7
2.1.1	General	7
2.1.2	Data Structures	7
2.1.3	Converter	7
2.1.4	Animation Controller	8
2.1.5	Cell animations	8
2.1.6	Multicell Animations	8
2.1.7	OAM Manager	8
2.1.8	Extended OAM Manager	8
2.1.9	Renderer	8
2.1.10	BG	9
2.1.11	Drawing Text	9
2.2	Changes from Version 1.2.1	9
2.2.1	General	9
2.2.2	Cell animations	9
2.2.3	Drawing OAM software sprites	9
2.2.4	Multicell animations	10
2.2.5	Renderer and renderer core	10
2.2.6	Converters	10
2.3	Changes from Version 1.1.0	11
2.3.1	Data Structures	11
2.3.2	Multicell Animation	11
2.3.3	Oam Software Sprite Rendering	11
2.3.4	Converter	11
2.4	Changes from Version 1.0.0	12
2.4.1	General	12
2.4.2	Multicell Animation	12
2.4.3	Extended OAM manager	12
2.4.4	Software Sprite	12
2.4.5	Image Load Related	12
2.4.6	Converter	12
2.5	Changes from Version 0.9.5	13
2.5.1	General	13

2.5.2	Cell	13
2.5.3	Software Sprite	13
2.5.4	OAM Software Sprite Rendering	13
2.5.5	Image Load Related	13
2.5.6	Renderer.....	14
2.5.7	Converter.....	14
2.6	Changes from Version 0.9.0	14
2.6.1	General.....	14
2.6.2	Cells.....	14
2.6.3	Software Sprites	14
2.6.4	Image Loading.....	14
2.6.5	Renderer.....	15
2.6.6	Converter.....	15
2.7	Changes from Version 0.8.0	16
2.7.1	Overall	16
2.7.2	OAM Manager	16
2.7.3	Extended OAM Manager	16
2.7.4	Renderer.....	16
2.7.5	Converter	16
2.8	Changes from Version 0.7.0	17
2.8.1	OAM Manager	17
2.8.2	Renderer.....	17
2.8.3	Cell Animation	17
2.8.4	Multicell Animation.....	17
2.8.5	Converter.....	17
2.9	Changes from Version 0.6.0	18
2.9.1	Animation	18
2.9.2	Converters	18
2.10	Changes from Version 0.5.0	19
2.10.1	General.....	19
2.10.2	Loading Files	21
2.10.3	OAM Manager	21
2.10.4	Software Sprites	22
2.11	Changes from Version 0.4.0	22
2.11.1	General.....	22
2.11.2	Cell Animation	22
2.11.3	Multicell.....	22
2.11.4	OAM Manager	22
2.11.5	Converter	22
2.12	Changes from Version 0.3.0	23
2.12.1	General.....	23

2.12.2	Animation Controller	23
2.12.3	Extended OAM Manager	23
2.12.4	Renderer Visibility	23
2.13	Changes from Version 0.2.0	24
2.13.1	File Loading	24
2.13.2	OAM Manager	24
2.13.3	Software Sprite	24
2.13.4	General	24
2.14	Changes from Version 0.1.0	24
2.14.1	File Loading	24
2.14.2	Animation Controller	24
2.14.3	OAM Manager	25
2.14.4	Extended OAM Manager 2	25
2.14.5	Image Load	25
2.14.6	Entity	25
2.14.7	Renderer	25
2.14.8	Sample	25
2.14.9	General	25
3	Known Problems	25
4	Future Plans	25

Revision History

Version	Revision Date	Description
1.2.3	09/01/2005	Updated to support the September 01 version.
1.2.2	06/06/2005	Updated to support the July 06 version.
1.2.1	03/28/2005	Updated to support the March 28 version.
1.2.0	01/31/2005	Updated to support the January 06 version.
1.1.0	12/06/2004	Updated to support the December 06 version.
1.0.0	11/10/2004	Updated to support the November 10 version.
0.9.5	10/12/2004	Updated to support the October 12 version: <ul style="list-style-type: none">• Added new items to schedule• Added a warning about multi-thread operations
0.9.0	09/16/2004	Updated to support the September 16 version.
0.8.0	09/02/2004	Updated to support the September 2 version.
0.7.0	08/10/2004	Updated to support the August 10 version.
0.6.0	08/02/2004	Updated to support the August 2 version.
0.5.0	07/20/2004	Updated to support the July 20 version.
0.4.0	6/22/2004	Updated to support the June 22 version.
0.3.0	6/10/2004	Updated to support the June 10 version.
0.2.0	5/28/2004	Updated to support the May 28 version.
0.1.0	5/10/2004	Initial version.

1 The G2D Library

1.1 The Runtime Library

The NITRO-System G2D library offers features that allow high-quality 2D display using both the Nintendo DS 2D graphics engine and the Nintendo DS 3D graphics engine. The G2D library can also use animation data that is output from the NITRO-CHARACTER 2D tool that is supplied with NITRO-System to pattern-animate characters.

See the *G2D Library 2D Graphics Overview* (`NitroSystem\docs\G2D\G2D_Overview.pdf`) or the *Function Reference* for details on the G2D library.

1.2 Multithread Operations

NITRO-System library was not designed to be fundamentally thread-safe (to support multithreading). Note that G2D library APIs called from interrupt handlers and differing threads may not always work correctly.

1.3 The Binary Converter

In order to generate data from the binary data files that are output from NITRO-CHARACTER that the G2D runtime can interpret, you must convert the binary files with a binary converter. `g2dcvtr.exe` is used for this conversion.

See `NitroSystem\docs\G2D\g2dcvtr_Manual.pdf` for details on how to use `g2dcvtr.exe`.

2 Major Revisions

2.1 Changes from Version 1.2.2

2.1.1 General

- The character data file formats NCCR and NCBR have been extended and the character location information block has been added. The function `NNS_G2dGetUnpackedCharacterPosInfo()` has also been added to access this data block.
- The sample demo `Renderer_CharChange` has been added. This demo switches between multiple images for the renderer. It also shows how to use the partial character output option `-br`.
- The sample demo `UserExAttribute` has been added. This demo uses user extended attributes.
- The sample demo `MultiCell_UILayout` has been added. This demo does user interface processing for game applications using multicells.

2.1.2 Data Structures

- An accessor API has been added to enable use of user extended attributes.
- `NNS_G2dGetAnimSequenceIndex()` has been added.

2.1.3 Converter

- The option `-br/` has been added and enables the conversion process to be done by specifying a rectangular portion of the character file.
- An unnecessarily large transfer size was specified when VRAM transfer information is output. This problem has been corrected.
- During the normal conversion process, an invalid warning message was output. The warning displays information about an invalid type conversion that causes truncation when converting variable types within the converter. In previous versions, even valid type conversions outputted this warning. This problem has been corrected.
- Screens with a height that were not a multiple of 256 were not properly converting. This problem has been corrected.
- The rectangular region size was output one size larger than the actual region size. This problem has been corrected.
- The `-rtp` option has been added to remove transparent pixels from the boundary region before calculating boundary region information.
- The `-oua` option has been added to extract comment strings in extended comment fields that are added in nce file v1.04 as extended user attribute information. (For details, see the `g2dcvtr` manual.)

2.1.4 Animation Controller

- The following new API functions have been added:

`NNS_G2dGetAnimCtrlCurrentAnimSequence()` and `NNS_G2dGetAnimCtrlCurrentElemIdxVal()`.

2.1.5 Cell animations

- The rendering of animation sequences that specified a zero display time for the leading animation would fail. This problem was corrected.

2.1.6 Multicell Animations

- Animation updating did not perform correctly because the multicell Node attribute and `NNS_G2D_MCANIM_PLAYMODE_CONTINUE` were not set.

Note: The fix for this problem has slightly increased the processing load to update multicell animations. Specifically, the average execution time of `NNS_G2dTickMCAnimation()` has increased by 19% (from 57 μ seconds to 68 μ seconds) in the `Renderer_PerfCheck` sample. The increase in the processing load depends on the data.

The characteristics of an increased processing load are as follows:

- Processing occurs during multicell switching.
 - No processing occurs if `NNS_G2D_MCANIM_PLAYMODE_CONTINUE` has not been set.
 - The processing load increases if the total number of animation frames is large in the cell animation referenced by multicell.
- The following API functions have been added: `NNS_G2dTraverseMCCellAnims()` and `NNS_G2dTraverseMCNodes()`.

2.1.7 OAM Manager

- A problem has been corrected for an assert that fails if the transformation enable flag is not set for the OBJ and the OBJ is registered by the OAM manager with an affine reference number. (The operations were changed to continue processing without setting the affine number.)

2.1.8 Extended OAM Manager

- When setting a registration function with the extended OAM manager using `NNS_G2dSetOamManExEntryFunctions()`, an erroneous assert warning was output when the affine parameters were not used but `NNSG2dOamExEntryFunctions.getAffineCapacity` and `NNSG2dOamExEntryFunctions.funcs.entryNewAffine` were set to NULL. This has been corrected.

2.1.9 Renderer

- A problem has been corrected in which cells were not drawn correctly when using OBJ drawing without using affine transformation after drawing a cell with the renderer using software sprites that use affine transformation.

2.1.10 BG

- A problem has been corrected in `NNS_G2dBGSetup` and `NNS_G2dBGLoadElements` in which the extended palette was being loaded into the wrong slot with the sub-screen as the target.
- A problem has been corrected in `NNS_G2dBGSetup` which resulted in the wrong BG mode when reading an affine-extended BG into a BG3 that is an affine-extended BG.
- Support was added for loading compressed palettes and partial characters.
- Support was added for loading free size screens.

2.1.11 Drawing Text

- A problem has been corrected in `NNS_G2dTextCanvasDrawTaggedText` in which changes made using a tag process callback to `TextCanvas` were not reflected in the rendered drawing.
- A function was added for creating cells that display `CharCanvas` using the renderer.

2.2 Changes from Version 1.2.1

2.2.1 General

- Cells with OBJs positioned outside of the range -128 and 127 were drawn incorrectly. This problem was fixed. OBJs in cells can be positioned from -256 to 255. You can revert to the previous drawing functionality by defining `NNS_G2D_LIMIT_CELL_X_128` in `include/fmt/g2d_Cell_data.h` and recompiling the library.
- The direction of flat surfaces is reversed on square polygons in software sprites. As a result of this change, sprites will display correctly even when the culling mode is set to `GX_CULL_BACK`.
- We added functions to manipulate fonts and text character/text string drawings.
- We added a font converter (`fontcvtr.exe`) that creates font resources for the aforementioned functions. For instructions on use, see `NitroSystem\docs\G2D\fontcvtr_Manual.pdf`.

2.2.2 Cell animations

- A problem has been corrected for the animation sequence rendering that would fail when the lead animation frame was specified as having a display time of zero in the cell animation. We added an assert statement that treats animation sequences having only animation frames with a display time of zero as invalid sequences.
- The OBJ position correction process that was performed when the double-size affine flag was enabled in `NNS_G2dMakeCellToOams()` was changed to perform only when the OBJ's double-size affine flag actually changes.

2.2.3 Drawing OAM software sprites

- We added support for drawing OBJs with an enabled double-size affine flag.

2.2.4 Multicell animations

- Added `NNS_G2dSetMCAnimationCellAnimFrame()`, which sets the playback animation frame for each cell animation in a multicell animation.

2.2.5 Renderer and renderer core

The OBJ position correction processing that is used when the double-size affine flag overwrite is specified using `NNS_G2dSetRendererAffineOverwriteMode()` will be performed only when the OBJ's double-size flag is actually changed.

The processing method used to draw with 2D graphics engine OBJs that have had their double-size affine flag enabled by NITRO-CHARACTER was changed. In the new version, when OBJs with an enabled double-size affine flag are entered, based on the assumption that the tool has performed OBJ position correction, $\text{OBJ size} * 1/2$ will be added to the OBJ position calculation, and the correction value for double-size affine OBJ is canceled before performing the process.

(If you want to revert to the previous operation, comment out the

`NNS_G2D_ASSUME_DOUBLEAFFINE_OBJPOS_ADJUSTED` definition in `g2d_config.h`.)

2.2.6 Converters

- We added the `-ncn` option, which uses the filenames of nce for the filenames of the NCGR (NCBR) output. The `-ncn` option is used when several 1D mapping format nce files are referencing a single ncg file.
- We corrected problems in `BuildNENR.exe`. (Mainly problems related to input path interpretation.)
- A warning process was added to prevent the output of invalid data. (For animation sequences that do not contain any animation frames, and sequences that contain only animation frames having a display time of zero.)

2.3 Changes from Version 1.1.0

2.3.1 Data Structures

- Added the `NNS_G2dCalcAnimSequenceTotalVideoFrames()` function, which calculates the total video frame length of the animation sequence data.

2.3.2 Multicell Animation

- Added `NNS_G2dResetMCCellAnimationAll()`, which resets the animation frames for the cell animation that makes up the multicell entity.
- Changed the multicell animation initialization functions. The sample demo that showed how to use the functions was also revised. The usage method has been simplified, and depending on the characteristics of the multicell animation data, improved performance and memory efficiency can be anticipated (this is effective in the case where data references the same cell animation). The existing functions also remain.

2.3.3 Oam Software Sprite Rendering

- Fixed a bug in the calculation method of the UV value used in software sprite rendering when a 1D mapping with a 256-color character was being used.

2.3.4 Converter

- Fixed a bug where file conversion failed for files with related files in the link file name data that use absolute path names.
- Made changes so that the extended comment information in the label definition header file output with the `-lbl` option is inserted as a C comment.
- Fixed a bug where the cell animation playback mode information was not configured properly when converting multicell data.
- Deleted the feature for creating output files in C source code format.

2.4 Changes from Version 1.0.0

2.4.1 General

- Updated the contents of the technical document and moved a section of the overview manual to API reference.

2.4.2 Multicell Animation

- The `NNS_G2dGetMCBankNumNodesRequired()` function was added.

2.4.3 Extended OAM manager

- With the lower priority OAM (drawn in the back), the problem that caused the number of drawing times to be set to a fewer number was fixed.

2.4.4 Software Sprite

- When `NNS_G2dSetSpriteAttrEnable()` was executed the second or later times, the value was not updated properly. This problem was solved.

2.4.5 Image Load Related

- With `NNS_G2dLoadImage*()` function and such, the problem that did not flash the cache contents before the DMA transfer was solved.

2.4.6 Converter

- The following options were added to the converter `BuildNENR.exe`.
 - `-o/...` Output destination folder specification
 - `-src` Source file output specification
- Added options (`-ai`, `-aisrt`, `-ait`) that specify the method of output animation element in `g2dcvtr.exe`.

2.5 Changes from Version 0.9.5

2.5.1 General

- A renderer core module was added. This is a module created by separating from the renderer module only the essential parts necessary for render processing. It was created with the assumption that users want to substantially customize their process, or speed up simple processes. The renderer module uses the renderer core module internally.
- Six samples have been added.

2.5.2 Cell

- `NNS_G2dGetCellAnimationCurrentCell()` has been added.
- A process has been added in cell (multicell) SRT animation for data using only translational animation. Previously, even when using only translational animation, scale and rotation was set to cell (multicell) animation. With this addition, unnecessary affine transformation settings are avoided.

2.5.3 Software Sprite

- `NNNS_G2dDrawSprite*Fast()` function has been added that does not save the current matrix of the 3D graphics engine. It works faster because the Push and Pop operations are not performed on the matrix stack.

2.5.4 OAM Software Sprite Rendering

- `NNNS_G2dDrawOneOam3DDirect*Fast()` function has been added that does not save the current matrix of the 3D graphics engine. It works faster because the Push and Pop operations are not performed on the matrix stack.
- Added an API to increase the efficiency of the rendering process by caching the UV parameter calculation results (`NNS_G2dSetOamSoftEmuSpriteParamCache()` and `NNS_G2dDrawOneOam3DDirectUsingParamCacheFast()`). The usable parts are limited, but it operates about 50% faster than the normal `NNS_G2dDrawOneOam3DDirect*Fast` function.

2.5.5 Image Load Related

- Fixed the bug in `NNS_G2dLoadPaletteEx()` function where the palette was not loaded correctly.
- Added a process that writes the memory cache of the transfer source data region back to main memory before data transfer to VRAM with `NNS_G2dLoadPalette()` and `NNS_G2dLoadPaletteEx()`.

2.5.6 Renderer

- Separated the rendering process part from matrix stack management and parameter management, and made the rendering process parts into a separate renderer core module.
- Process efficiency has improved.

Comparing with the multicell rendering process (see sample demo `Renderer_PerfCheck`):

- When the optimization method is not set, OBJ rendering is about -40% and software sprite rendering is -30%
- When the optimization method is set, OBJ rendering is about -50% and software sprite rendering is -40%
- `NNS_G2dOptimizeRenderer()` has been removed. `NNS_G2dBeginRenderingEx()` has been added as a alternative API.
- `NNS_G2dSetTrans()` has been added.
- Items were added that the renderer can specify in the parameter overwrite process performed for OAM (such as `NNS_G2dSetRendererOverwriteEnable/Disable()`).

2.5.7 Converter

In cell SRT animation, we added animation result types supporting data using only translational animation. We also added support for that output. With this extension, animation data capacity was reduced, and unnecessary affine transformation settings during execution were avoided.

2.6 Changes from Version 0.9.0

2.6.1 General

- The DMA channels used by the G2D library are now the same ones used by the GX library.
- To improve processing efficiency, made some of the library functions to inline functions.

2.6.2 Cells

- Added support for cell information that has rectangle information as cell region information. Also, the pre-existing internal release functions (such as `NNS_G2dGetCellBoundingRect()`) that obtain bounding sphere information were made into external release.

2.6.3 Software Sprites

- Fixed the bug that disabled the functions (such as `NNS_G2dSetSpriteCurrentPolygonID()`) used to set polygon IDs used while drawing software sprites.
- `NNS_G2dDrawSprite*()` no longer internally emulates wrapping when an off-screen position is specified.

2.6.4 Image Loading

- Added a palette load function (`NNS_G2dLoadPaletteEx()`) that loads only the portion of the palette that is actually used. To use this functionality, you must use the converter's `-pcm` option to

create compressed palette data.

- Fixed the bug in which the loading destination address value was not reflected when using `NNS_G2dLoadPalette()` function to load 256-color extended palette data for use with the 2D graphics engine.

2.6.5 Renderer

- Added functions (such as `NNS_G2dSetRendererOverwriteEnable()` and `NNS_G2dSetRendererOverwrite*()`) that overwrite the drawing OBJ parameters and are easier for the programmer to manipulate.
- You can now control the OBJ affine transform mode rewrite method (such as `NNS_G2dSetRendererAffineOverwriteMode()`) used when the renderer draws OBJs using the 2D graphics engine.
- Changed the argument type used by callbacks before and after drawing OBJs.
- Fixed the bug in which cell data meeting both of the following conditions were not drawn properly:
 - The load destination address of the image proxy character data is a non-zero value.
 - The character data uses a 256-color palette format or a mapping mode other than 1D-32K.

2.6.6 Converter

- The option `-pcm` was added. This option outputs compressed palette data that holds only the used palette numbers as data. Palettes can be partially loaded using `NNS_G2dLoadPaletteEx()` to load the compressed palette data.
- Error messages are now output to standard error. Also, the user can control the output of the converter's operation messages. The default value halts output of the converter's operation messages. If option `-v` is used, the operation messages are sent to standard output.
- The converter's execution result is now returned as a return value. A 0 is returned when the execution succeeded. A non-zero is returned when the execution failed.
- A bug caused the exported data to be corrupted if object compression was turned OFF when converting 256-color palettes and 1D mapping characters. This was fixed.
- The option `-br` was added. This option outputs cell data that contains rectangular area information.

2.7 Changes from Version 0.8.0

2.7.1 Overall

- Added the feature that displays BG semi-automatically, using screen data information.

2.7.2 OAM Manager

- Changed the way in which double-size affine transformation mode is applied forcibly to an OBJ that is registered using `NNS_G2dEntryOamManagerOamWithAffineIdx()`. Only the affine parameter numbers are rewritten. The affine transformation mode will not be changed.

The same processing must be performed outside the manager. (When using `NNS_G2dMakeCellToOams()` and the renderer, you do not need to be concerned with this. In the next release, we plan to release an API that controls the rewriting operations that are performed in the renderer's affine transformation mode.)

2.7.3 Extended OAM Manager

- Removed the ability to set the OBJ affine transformation mode (normal or double-size). The following functions were deleted:

- `NNS_G2dSetOamManExDoubleAffineFlag()`
- `NNS_G2dGetOamManExDoubleAffineFlag()`

The same processing must be performed outside the manager. (When using `NNS_G2dMakeCellToOams()` and the renderer, you do not need to be concerned with this.)

2.7.4 Renderer

- Implemented processing that supports the switching of the active state of the render surface. This feature allows you to forcibly prohibit drawing on specific surfaces. The following accessor functions were added:

- `NNS_G2dSetRendererSurfaceActiveFlag()`
- `NNS_G2dGetRendererSurfaceActiveFlag()`

2.7.5 Converter

- Added support for the conversion of BG data.

2.8 Changes from Version 0.7.0

2.8.1 OAM Manager

- Added the feature to shift the z value automatically and to draw sprites when the software sprite drawing feature is used. (`NNS_G2dSetOamManagerSpriteZoffsetStep()`)

2.8.2 Renderer

- Added the feature to shift the z value automatically and to draw sprites when the software sprite drawing feature is used. (`NNS_G2dSetOamManagerSpriteZoffsetStep()`)
- Examined the renderer setting status, and implemented the feature that optimizes the rendering process. (`NNS_G2dOptimizeRenderer()`)
- Implemented the flip drawing feature. (`NNS_G2dSetRendererFlipMode()`)
- Made it possible to register the callback functions that are used before and after drawing a cell or an OBJ.

2.8.3 Cell Animation

- Added the support for SRT (scale, rotate, and translate) animation.

Note: Many of the NCE data that was created with previous versions of NITRO-CHARACTER contain animation frames that include an incorrect scale zero value. Such data cannot be drawn properly with the new runtime.

Reconvert the data and replace scale zero, or save the data again with the latest version of NITRO-CHARACTER. (this applies to the multicell animation that is described below.)

2.8.4 Multicell Animation

- Added the support for SRT (scale, rotate, and translate) animation.
- Fixed the bug that did not reset the cell animation playback status when multicells were switched. Now the multicell animations that use the cell animations in stop playback mode can be played back properly.

2.8.5 Converter

- Implemented the feature that replaces the scale value with a value of "1" if the scale value is specified as "0" for the SRT animation data.

2.9 Changes from Version 0.6.0

2.9.1 Animation

- Fixed the problem that caused the animation frame of frame 0 to be set regardless of the frame number entered with `NNS_G2dSetAnimCtrlCurrentFrame()`.
- Fixed the problems with:
 - `NNS_G2dSetMCAnimationCurrentFrame()`
 - `NNS_G2dSetCellAnimationCurrentFrame()`
 - `NNS_G2dSetCellAnimationCurrentFrameNoReset()`(Internally, these all call `NNS_G2dSetAnimCtrlCurrentFrame()`)

2.9.2 Converters

- Fixed the bug that caused parameter interpretation to fail with entered file names that do not include a `'\'` or `'/'` in their file path.
- Added the output option `-bg` for character (ncg) data used for BG. When using the `-bg` option, no nce (cell definition) file is necessary during conversion.
- Added option `-lbl`, which outputs the label name definition header file. The label name definition header file defines label text strings linked to animation sequences as aliases for label numbers.
- The converter log message was changed to output more data.

2.10 Changes from Version 0.5.0

2.10.1 General

- The name of the converter program was changed from `G2DConv.exe` to `g2dcvtr.exe`, and the features were enhanced. (See the converter manual for details.)
- Changed function names. Function names in past versions have been aliased and still work. Please use the new functions for the new projects. The following functions have changed.

Animation

Old	New
<code>NNS_G2dGetCurrentElement</code>	<code>NNS_G2dGetAnimCtrlCurrentElement</code>
<code>NNS_G2dGetNextElement</code>	<code>NNS_G2dGetAnimCtrlNextElement</code>
<code>NNS_G2dGetNormalizedTime</code>	<code>NNS_G2dGetAnimCtrlNormalizedTime</code>
<code>NNS_G2dSetCallBackFunc</code>	<code>NNS_G2dSetAnimCtrlCallBackFunc</code>
<code>NNS_G2dSetCallBackFuncAtAnimFrame</code>	<code>NNS_G2dSetAnimCtrlCallBackFuncAtAnimFrame</code>
<code>NNS_G2dGetAnimSpeed</code>	<code>NNS_G2dGetAnimCtrlSpeed</code>
<code>NNS_G2dSetAnimSpeed</code>	<code>NNS_G2dSetAnimCtrlSpeed</code>
<code>NNS_G2dResetAnimationState</code>	<code>NNS_G2dResetAnimCtrlState</code>
<code>NNS_G2dInitCallBackFunc</code>	<code>NNS_G2dInitAnimCallBackFunc</code>
<code>NNS_G2dBindAnimController</code>	<code>NNS_G2dBindAnimCtrl</code>
<code>NNS_G2dGetAnimControllerType</code>	<code>NNS_G2dGetAnimCtrlType</code>
<code>NNS_G2dInitAnimController</code>	<code>NNS_G2dInitAnimCtrl</code>
<code>NNS_G2dIsAnimControllerActive</code>	<code>NNS_G2dIsAnimCtrlActive</code>
<code>NNS_G2dStartAnimController</code>	<code>NNS_G2dStartAnimCtrl</code>
<code>NNS_G2dStopAnimController</code>	<code>NNS_G2dStopAnimCtrl</code>
<code>NNS_G2dTickAnimController</code>	<code>NNS_G2dTickAnimCtrl</code>
<code>NNS_G2dInitAnimControllerCallBackFunc</code>	<code>NNS_G2dInitAnimCtrlCallBackFunc</code>

Cell Animation

Old	New
<code>NNS_G2dSetCellAnimSpeed</code>	<code>NNS_G2dSetCellAnimationSpeed</code>
<code>NNS_G2dGetCellAnimAnimCtrl</code>	<code>NNS_G2dGetCellAnimationAnimCtrl</code>
<code>NNS_G2dInitializeCellAnimation</code>	<code>NNS_G2dInitCellAnimation</code>
<code>NNS_G2dInitializeCellAnimationVramTransferred</code>	<code>NNS_G2dInitCellAnimationVramTransferred</code>

Multicell Animation

Old	New
NNS_G2dInitializeMCAnimation	NNS_G2dInitMCAnimation
NNS_G2dInitializeMCInstance	NNS_G2dInitMCInstance
NNS_G2dSetMCAnimSpeed	NNS_G2dSetMCAnimationSpeed

Image Load Related

Old	New
NNS_G2dInitializeImageProxy	NNS_G2dInitImageProxy
NNS_G2dInitializeImagePaletteProxy	NNS_G2dInitImagePaletteProxy

Entity

Old	New
NNS_G2dInitializeEntity	NNS_G2dInitEntity
NNS_G2dSetCurrentAnimation	NNS_G2dSetEntityCurrentAnimation

OAM Manager

Old	New
NNS_G2dInitializeOamManager	NNS_G2dInitOamManagerModule
NNS_G2dEntryNewOam	NNS_G2dEntryOamManagerOam
NNS_G2dEntryNewOamWithAffineIdx	NNS_G2dEntryOamManagerOamWithAffineIdx
NNS_G2dEntryNewOamAffine	NNS_G2dEntryOamManagerAffine
NNS_G2dEntryOamAffine	NNS_G2dSetOamManagerAffine
NNS_G2dApplyToHW	NNS_G2dApplyOamManagerToHW
NNS_G2dApplyToHWSoftEmu	NNS_G2dApplyOamManagerToHWSprite
NNS_G2dResetBuffer	NNS_G2dResetOamManagerBuffer
NNS_G2dApplyToHWAndReset	NNS_G2dApplyAndResetOamManagerBuffer
NNS_G2dGetOamAttrCapacity	NNS_G2dGetOamManagerOamCapacity
NNS_G2dGetOamAffineCapacity	NNS_G2dGetOamManagerAffineCapacity

Expanded OAM Manager

Old	New
NNS_G2dEntryNewOamEx	NNS_G2dEntryOamManExOam
NNS_G2dEntryNewOamWithAffineIdxEx	NNS_G2dEntryOamManExOamWithAffineIdx
NNS_G2dEntryNewAffineParamsEx	NNS_G2dEntryOamManExAffine
NNS_G2dApplyToBaseModuleEx	NNS_G2dApplyOamManExToBaseModule
NNS_G2dResetOamInstanceEx	NNS_G2dResetOamManExBuffer
NNS_G2dGetNewOamInstanceEx	NNS_G2dGetOamManExInstance
NNS_G2dSetOamEntryFunctionsEx	NNS_G2dSetOamManExEntryFunctions

Renderer

Old	New
NNS_G2dAddTargetSurface	NNS_G2dAddRendererTargetSurface
NNS_G2dSetCurrentImageProxy	NNS_G2dSetRendererImageProxy

Software Sprites

Old	New
NNS_G2dSetCurrentPolygonID	NNS_G2dSetSpriteCurrentPolygonID
NNS_G2dSetDefaultAttr	NNS_G2dSetSpriteDefaultAttr
NNS_G2dGetDefaultAttr	NNS_G2dGetSpriteDefaultAttr

2.10.2 Loading Files

Improved debug character string output of the color palette.

2.10.3 OAM Manager

- Added `NNS_G2dGetNewOamManagerInstance()`, a new API that generates the OAM Manager entity.
- Added `NNS_G2dGetNewOamManagerInstanceAsFastTransferMode()`, an API that generates the OAM Manager entity and that can transfer the internal buffer at high speed using DMA.
- Added support for when the OAM Manager entity does not control the affine parameter
- Fixed the problem where `affine` parameters outside the control range were incorrectly reset by `NNS_G2dResetBuffer()` (internal buffer reset process)
- Fixed the problem of incorrect drawing from occurring outside of the double `affine` mode when the software sprite drawing function is used.

2.10.4 Software Sprites

- Changed the parameters of the 3D graphics engine camera settings for software sprites. Fixed the problem of disparities in the image output between 2D and 3D drawing.

2.11 Changes from Version 0.4.0

2.11.1 General

- Added support for multicell.
- Changed the version of every file format to 1.0. (Old data cannot be used. Please use the new converter to reconvert.)
- Added CellVRAM transfer state manager module. (Used in VRAM transfer animation.)
- Drawing functionality using OAM software sprites separated from OAM Manager. OAM software sprite drawing module added.

2.11.2 Cell Animation

- Added support for VRAM transfer animation.

2.11.3 Multicell

- Changed the name of the `NNS_G2dGetNumNodesRequired()` function to `NNS_G2dGetMCNumNodesRequired()`.
- Changed the name of the `MultiCellDataToMCInstance()` function to `NNS_G2dSetMCDataToMCInstance()`.

2.11.4 OAM Manager

- Separated the drawing function using OAM software sprites into its own module. (The behavior of `NNS_G2dDrawOneOam3DDirect()` was changed accordingly. For details, see the API reference.)

2.11.5 Converter

- Implemented version checking of NITRO-CHARACTER files. Old version files can no longer be used. (Old data must be resaved using the new NITRO-CHARACTER.)
- Various output options were added. (See the converter documentation for details.)
- Added support for the output of various 1D mapping modes.
- Added support for conversion of data using the character compression functionality.

- Made it possible to output VRAM transfer animation data.
- Added support for output of animation play method information.

2.12 Changes from Version 0.3.0

2.12.1 General

- Combined the Extended OAM Manager and the Extended OAM Manager 2 into the Extended OAM Manager. The API uses the Extended OAM Manager 2 format.
- Changed the format of the Cell definition data NCER file. The data from previous versions cannot be used. (Re-conversion is necessary.)
- Renamed `NNSG2dCellData.cellType` to `NNSG2dCellData.cellAttr`.
- The function group that outputs runtime library debug data is defined as a dummy inline function in the FINALROM build.

2.12.2 Animation Controller

- Converted the accessor group functions that were defined in the header to inline functions.

2.12.3 Extended OAM Manager

- Fixed the bug that caused improper rendering when registering OAM without using affine parameters.
- Can specify the method (normal or double-sized affine) for affine transformations that are used by the OAM Manager. (`NNS_G2dSetOamManExDoubleAffineFlag()`)
- Can no longer configure the default rendering registration functions.
`NNS_G2dSetOamEntryFunctionsEx()` must be executed.

2.12.4 Renderer Visibility

- Added a Visibility Culling feature to the Renderer. (Data must be reconverted with the new converter.)
- Corrected the problem that prevented proper rendering when the OBJ position was $y \leq -64$ in the local coordinate system in `Cell`.
- Added a function to initialize the render surface. (`NNS_G2dInitRenderSurface()`)
- Added a pointer to the function that performed Visibility Culling on the render surface member.

2.13 Changes from Version 0.2.0

2.13.1 File Loading

- In the `NNS_G2dPrintAnimContents()` function, the display of debug animation data that was created using the SRT (Scale, Rotate, and Translate) animation feature failed. This unexpected behavior has been fixed.

2.13.2 OAM Manager

- Fixed the bug that caused improper operation when `NNS_G2D_OAMTYPE_SOFTWAREEMULATION` was specified as a manager type and the OAM management region start index was specified as a value other than 0.
- Fixed the bug that caused improper operation when the affine parameter management region start index was specified as a value other than 0.

2.13.3 Software Sprite

- Fixed the bug that caused improper rendering after a sprite with an alpha value of 0 was rendered.
- Fixed the bug that caused improper rendering when the `NNS_G2D_SPRITEATTR_UV` attribute was not used.
- Corrected the problem of a difference in the behavior of the 2D graphics engine and software sprite processing when scrolling.

2.13.4 General

- Added converter support for specifications to allow multiple OBJs in a single `Cell` to share the character region during 1D mapping.

2.14 Changes from Version 0.1.0

2.14.1 File Loading

- Added the `NNS_G2dGetUnpackedXXX()` function group to simplify the description of user code.
- Added the `NNS_G2dIsBinFileValid()` function to verify file extensions and versions.

2.14.2 Animation Controller

- Added playback control functions (`NNS_G2dStartAnimController()`, `NNS_G2dStopAnimController()`, etc.)
- Added functions to allow the programmer to change the playback mode. (`NNS_G2dSetAnimCtrlPlayModeOverridden()`, etc.)
- Renamed `NNS_G2dSetCallBackFuncAtSpecifiedFrame()` to `NNS_G2dSetCallBackFuncAtAnimFrame()`.

2.14.3 OAM Manager

- Changed the arguments for `NNS_G2dApplyToHWSoftEmu()` and `NNS_G2dDrawOneOam3DDirect()`.

2.14.4 Extended OAM Manager 2

- Changed the arguments for `NNS_G2dGetNewOamInstanceEx2()`.

2.14.5 Image Load

- Added a convenient function group to simplify the loading of data to an image proxy.

2.14.6 Entity

- Provided the `BuildNENR.exe` tool to create entity run-time binary files.

2.14.7 Renderer

- Changed the arguments for drawing methods. The matrix cache for the `NNS_G2dDrawXXX` internal data structure has been hidden.
- Changed the arguments for `NNS_G2dPopMtx()`.

2.14.8 Sample

- Revised the sample to support SDK 1.2 and load sample data from the file system.
- Added character data display samples for each format (256-color palette, extended palette, etc.).

2.14.9 General

- Allowed the compiler switch to turn on or turn off debug output in the library. Debug output is enabled when `NNSI_G2D_DEBUG` is defined.
- Changed the file format and converter. Reconvert previously created files.

3 Known Problems

None.

4 Future Plans

None.

© 2004-2005 Nintendo

No part of the contents of this document may be reproduced, copied, transferred, distributed, or given without the permission from Nintendo.