

N I N T E N D O
NITRO-System

G2D Library

2D Graphics Overview

Version 1.0.0

**The contents in this document are highly
confidential. Keep it in the strictest confidence.**

Table of Contents

1	Introduction	5
2	NITRO-System 2D Development Environment	6
2.1	2D Development Flow	6
2.2	NITRO-CHARACTER	6
2.3	NITRO-CHARACTER Binary Files	7
2.4	G2D Runtime Binary Format	7
2.5	The Binary Converter	8
2.6	The G2D Library	8
3	Overview of the G2D Library	9
3.1	Library Functionality	9
3.1.1	Playback of NITRO-CHARACTER Animation Data	9
3.1.1.1	Animation Control Playback	9
3.1.1.2	Calling Callbacks at Various Times within the Animation	9
3.1.2	OAM Management	9
3.1.2.1	Manage the Registration of OAM Attribute and Affine Parameter	9
3.1.2.2	Drawing Software Sprite of the Registered OAM Attribute	9
3.1.2.3	Timeshare OBJ Display When There Are More OBJs than Used OAM Attributes	9
3.1.3	Drawing Sprites Using the 3D Graphics Engine	9
3.1.3.1	The Number of Sprites	10
3.1.3.2	Sprite Rotation and Scaling	10
3.1.3.3	Translucency Feature	10
3.1.4	Load Process of Image Data to VRAM	10
3.1.5	Complex Animations Composed of Pattern Animations	10
3.1.6	VRAM Transfer Animation	11
3.1.6.1	Avoid Unnecessary VRAM Transfer	11
3.1.7	Upper Drawing Module	12
3.1.7.1	Handling the Nintendo DS Main Screen and Sub Screen as One Contiguous Screen	12
3.1.7.2	Affine Transformation Parameter Management	12
3.1.7.3	Dynamically Switching the Drawing of Software Sprites and OBJs	12
3.1.7.4	Overwriting the OBJ Parameters for Drawing	12
3.1.8	BG Automatic Composition	12

Tables

Table 2-1	NITRO-CHARACTER Binary Format Types	7
Table 2-2	G2D Runtime Binary Format Types	7

Figures

Figure 2-1	The 2D Development Environment	6
------------	--------------------------------------	---

Revision History

Version	Revision date	Description of revisions
1.0.0	01/05/2005	Changed "NITRO" to "Nintendo DS."
0.8.0	12/06/2004	Standardized phrases, overall revision. Moved the description of details to API reference.
0.7.0	09/16/2004	Added BG functionality.
0.6.0	08/02/2004	Added support for August 2 version. Corrected typos.
0.5.0	07/20/2004	Added support for July 20 version.
0.4.0	06-22-2004	Added support for June 22 version.
0.3.0	06-10-2004	Added support for June 10 version. Correction of typos.
0.2.0	05-28-2004	Added support for May 28 version.
0.1.1	05-12-2004	Corrected typos. Corrected tables and figures.
0.1.0	05-10-2004	Initial release.

1 Introduction

The Nintendo DS allows you to draw OBJs and BGs with the 2D graphics engine and polygons with the 3D graphics engine. The NITRO-System 2D graphics library can use both the 2D and 3D graphics engines to support more complex 2D renderings.

2 NITRO-System 2D Development Environment

2.1 2D Development Flow

The 2D development process for the NITRO-System is shown in Figure 2-1. Graphic designers use NITRO-CHARACTER a 2D design tool to design characters and OBJ animations. The programmer uses a binary converter to convert the NITRO-CHARACTER binary files. These binary files are saved by NITRO-CHARACTER to runtime binary files that can be used by the G2D library of the 2D graphics library.

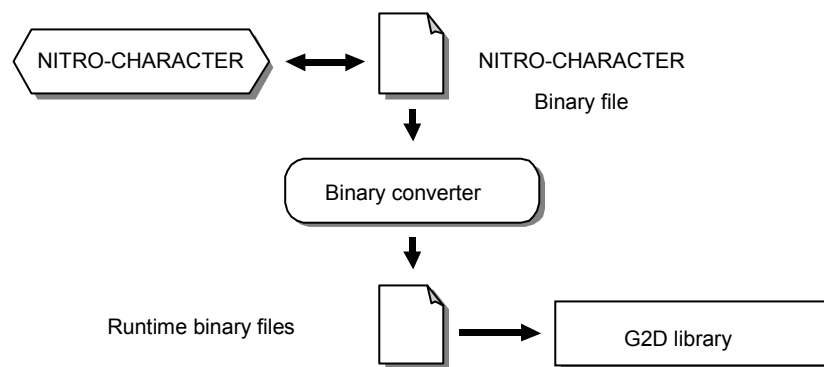


Figure 2-1 The 2D Development Environment

2.2 NITRO-CHARACTER

NITRO-CHARACTER is a Windows application that creates the character, screen, and OBJ data used to create 2D screens for the Nintendo DS (DS). NITRO-CHARACTER can create the following:

- Character data (drawing and editing of dot images)
- Screen data
- OBJ animation

2.3 NITRO-CHARACTER Binary Files

The NITRO-CHARACTER binary files are used as NITRO-CHARACTER saved file formats. The NITRO-CHARACTER binary files contain a variety of data used by the DS 2D graphics engine and data specific to NITRO-CHARACTER. Using NITRO-CHARACTER, designers save their work as NITRO-CHARACTER binary files that can be loaded back into NITRO-CHARACTER to restore source files. There are eight types of NITRO-CHARACTER binary files shown in Table 2-1.

Table 2-1 NITRO-CHARACTER Binary Format Types

File type	Extension	File Extension Explanation
Scene files	.nsn	Nitro S ce N e
Color palette files	.ncl	Nitro C o L or P alette
Character files	.ncg	Nitro C haracter G raphics
Cell files	.nce	Nitro C ell
Multicell files	.nmc	Nitro M ulti C ell
Screen files	.nsc	Nitro S creen
Panel files	.npl	Nitro P anel L
Panel screen files	.nps	Nitro P anel S creen

2.4 G2D Runtime Binary Format

G2D runtime binary format is the data format which the G2D library processes.

This data is generated using the binary converter described in 2.5 - The Binary Converter.

There are nine formats for G2D runtime binary formats as shown in Table 2-2.

Table 2-2 G2D Runtime Binary Format Types

File type	Extension	File Extension Explanation
Color palette files	.nclr	Nitro C o L or P alette for R untime
Character files	.ncgr	Nitro C haracter G raphics for R untime
Character files (bitmaps)	.ncbr	Nitro C haracter B itmap F ormat for R untime
Cell files	.ncer	Nitro C ell for R untime
Multicell files	.nmcr	Nitro M ulti C ell for R untime
Cell animation files	.nanr	Nitro C ell A nimation for R untime
Screen files	.nscr	Nitro S creen for R untime
Multicell animation files	.nmar	Nitro M ulticell A nimation for R untime
Entity files	.nenr	Nitro E ntity for R untime

2.5 The Binary Converter

The binary converter `g2dcvtr.exe` is a Windows application that converts the binary files generated NITRO-CHARACTER into runtime binary files; these files are used by the G2D library. For details about the binary converter and the runtime binary data format, see the following documents:

(NitroSystem\docs\G2D\g2dcvtr_Manual.pdf)

(NitroSystem\docs\G2D\NNS_G2DRuntime-BinaryFormat.pdf)

2.6 The G2D Library

The G2D library allows the user to generate DS 2D graphics easily. When G2D is used in conjunction with NITRO-CHARACTER (the 2D design tool), the G2D library provides the functions to create complex 2D graphics for the DS.

3 Overview of the G2D Library

3.1 Library Functionality

These sections describe the functionality of the G2D library and its capabilities.

3.1.1 Playback of NITRO-CHARACTER Animation Data

Using the animation data generated by NITRO-CHARACTER, you can create pattern animations for game characters.

3.1.1.1 Animation Control Playback

The playback mode (i.e., playback, reverse playback, slow playback, loop playback, etc.) can be configured in the program. This feature supports the various playback modes that can be set with NITRO-CHARACTER.

3.1.1.2 Calling Callbacks at Various Times within the Animation

You can call callback functions at various times during animation playback. It is also possible to register 4-byte user parameters with callback functions.

Related modules:

Cell animation, animation controller

3.1.2 OAM Management

This feature controls the OAM of the 2D graphics engine at a micro level.

3.1.2.1 Manage the Registration of OAM Attribute and Affine Parameter

Specify the area that the OAM manager manages.

3.1.2.2 Drawing Software Sprite of the Registered OAM Attribute

Using 3D polygons, draws the OBJ registered in OAM manager without registering it to OAM.

3.1.2.3 Timeshare OBJ Display When There Are More OBJs than Used OAM Attributes

Extended OAM Manager shares time with the display of OBJs when the number of OBJs is greater than the number of used OAM attributes. The display blinks when displayed.

Related modules:

OAM manager, extended OAM manager

3.1.3 Drawing Sprites Using the 3D Graphics Engine

The G2D library includes a feature called software sprites that draws sprites using square polygons with the DS 3D graphics engine.

Using a software sprite gives you a higher degree of freedom to draw an object than the 2D graphics engine.

G2D also includes the OAM software sprite drawing module that interprets the contents of the OAM attribute and provides the API that draws software sprites. With this module, you can draw the same OBJ data with software sprite or (hardware) OBJ. Due to the hardware restrictions, both the character data for software sprites and the character data for OBJs must be stored in VRAM when switching between software sprites and OBJs. Character data for software sprites and for OBJs are in different formats, but the feature that converts them is provided by using the converter.

The following sections describe the features of software sprites.

3.1.3.1 The Number of Sprites

The number of sprites that can be displayed with software sprites is larger than the number of OBJs displayed using the 2D graphics engine. Because the Nintendo DS displays each software sprite on a quadrilateral polygon, the total number of software sprites is limited only by the number of quads that the DS can draw. If polygons are used only for software sprites, the DS can display up to 1536 software sprites.

3.1.3.2 Sprite Rotation and Scaling

In the OBJ functionality of the 2D graphics engine, software sprites have no limit on the number of affine parameters. Also, characters can be scaled only to double their size with the OBJ functionality in the 2D graphics engine, but software sprites can be scaled to larger sizes and display properly.

3.1.3.3 Translucency Feature

The transparency level of software sprites ranges from 0 (zero) to 31. The sprite is opaque if the transparency level is set to 31. If the transparency level is set to 0 (zero), the sprite is transparent and does not appear on the screen.

Related modules:

Software sprite, OAM software sprite drawing

3.1.4 Load Process of Image Data to VRAM

The G2D library includes a feature to load image data to VRAM. A mechanism for managing loaded image data (e.g., image and image palette proxy) is included also.

Related modules:

Load Image

3.1.5 Complex Animations Composed of Pattern Animations

With NITRO-CHARACTER, you can use multicell animation to create large game characters from small components that does pattern animations with different frame counts. You can build complex game characters with pattern animation, using the various components and timing to build a game character.

The G2D library supports multicell data drawing and animation.

Related modules:

Multicell

3.1.6 VRAM Transfer Animation

The G2D library includes a system (VRAM Transfer Animation) which allows you to render animation by transferring and rewriting the image (VRAM) data referenced by the cell.

By using the VRAM transfer animation feature, animations with a large amount of image data can be created while reducing the image data capacity required by the hardware. The VRAM transfer animation is generally used to develop a main character with many animation patterns for 2D game development.

3.1.6.1 Avoid Unnecessary VRAM Transfer

The state of the VRAM transfer animation is managed by the cell VRAM transfer state object. The cell VRAM transfer state object stores state information (e.g., about whether the animation was switched or drawn), and the library uses this state information to do the VRAM transfer only as needed.

Related modules:

Cell animation, cell VRAM transfer state manager

3.1.7 Upper Drawing Module

The G2D library provides an upper drawing module for complex drawing processes.

3.1.7.1 Handling the Nintendo DS Main Screen and Sub Screen as One Contiguous Screen

Two screens can be treated as one large contiguous screen. The screen can be placed between the two screens at any location.

3.1.7.2 Affine Transformation Parameter Management

This feature monitors the affine transformation and sets the affine parameter automatically. If the affine parameters can be shared, set the affine parameters to `share` to reduce the number of affine parameters used. The flip conversion of the affine transformed game character is done by automatically calculating the affine matrix.

3.1.7.3 Dynamically Switching the Drawing of Software Sprites and OBJs

You can dynamically switch the drawing methods.

3.1.7.4 Overwriting the OBJ Parameters for Drawing

With the specified value, you can overwrite the OBJ parameter for drawing.

Related modules:

Renderer, Renderer core

3.1.8 BG Automatic Composition

G2D runtime screen data inherits information from the NITRO-CHARACTER binary data required to display the BG. The G2D library provides functionality to automatically carry out the BG control and load graphics data into VRAM using this information. The programmer can use this functionality to display BGs using common initialization tasks, allocating VRAM, and enabling the BG visibility state.

Related modules:

BG

© 2004-2005 Nintendo

The contents of this document cannot be duplicated, copied, reprinted, transferred, distributed or loaned in whole or in part without the prior approval of Nintendo.