NintendoWare for CTR

Texture File Additional Information

2010/07/29

Version 1.1

The content of this document is highly confidential and should be handled accordingly.

CTR-06-0316-001-A
Released: January 27, 2014

Table of Contents

Tables

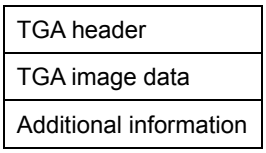
Figures

1 Introduction

The texture files that NintendoWare for CTR applies to 3D models use the special TARGA (TGA) file format, which includes additional information such as texture format information and texture data for NintendoWare. If the texture file has this additional information, the 3DCG tool export plug-in reads the additional information and includes it when the intermediate file is exported.

The current version for this additional information is 1.0. Figure 1-1 outlines the structure of the texture file.

Figure 1-1 Structure of the Texture File



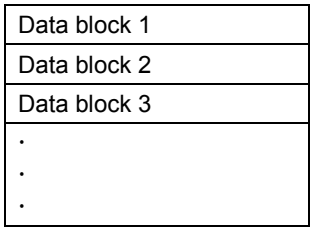
When mipmap setting is enabled, only the highest-level data is stored in the TGA image data area. The remaining mipmap levels of data are stored in the additional information area.

2 Additional Information Structure

As shown in Figure 2-1, the additional information comprises a number of data blocks containing various kinds of binary data.

The byte order of the binary data is always little-endian.

Figure 2-1 Additional Information Structure



Each data block starts with a data block header as shown in Table 2-1. The block header, which is 12 bytes long, stores an identifier indicating the data type and the data block size.

Table 2-1 Data Block Header

Type	Description	Size
`char[8]`	Identifier that indicates the type of data in the data block. Always stores eight ASCII characters.	8 bytes
`u32`	Data block size. This size includes the 12 bytes of the data block header. Note that the size does not need to align to any particular address boundary.	4 bytes

The data types used in this document are shown in Table 2-2.

Table 2-2 Data Types

Data Type	Description
`u8`	8-bit unsigned integer
`u16`	16-bit unsigned integer
`u32`	32-bit unsigned integer

3 Data Blocks

3.1 Data Block Types

Table 3-1 shows the data block types.

Certain data block types may not appear in the additional information area under some conditions.

The `nw4c_tfm` block must always be the first data block in the additional information area and the `nw4c_end` block must always be the last data block. Apart from these restrictions, the other data blocks in the additional information area can appear in any order.

For the string type data, the "¥0" (0x00), which indicates the end of the string, is unnecessary.

Table 3-1 Data Block Types

Identifier	Data Type	Description
`nw4c_tfm`	String	Texture format name • `L4`: 4-bit luminance • `L8`: 8-bit luminance • `A4`: 4-bit alpha • `A8`: 8-bit alpha • `La4`: 4-bit luminance and 4-bit alpha • `La8`: 8-bit luminance and 8-bit alpha • `Hilo8`: 8-bit R,G • `Rgb565`: 5-bit R, 6-bit G, 5-bit B • `Rgb8`: 8-bit R,G,B • `Rgb5_a1`: 5-bit R,G,B and 1-bit alpha • `Rgba4`: 4-bit R,G,B,A • `Rgba8`: 8-bit R,G,B,A • `Etc1`: RGB compression • `Etc1_a4`: RGB compression and 4-bit alpha
`nw4c_cub` Present when there are cube maps.	No data	This data block is just a header The size of a single texture for a cube map is as follows. • Its width is 1/4 the TGA image width • Its height is 1/3 the TGA image height
`nw4c_mps` Present if mipmaps are enabled.	String	The number of mipmap levels This is either 2, 3, 4, 5, 6, 7, or 8.

Identifier	Data Type	Description
`nw4c_txd`	u8 array	Texture data This is the native format that can be used by the CTR system. Cube maps are saved as six consecutive textures in the +X, -X, +Y, -Y, +Z, and -Z directions, respectively. When mipmaps are enabled, data is stored consecutively for all levels. For example, data is stored in the following order when there are two mipmap levels. • +X (level 0) • +X (level 1) • -X (level 0) • -X (level 1) • +Y (level 0) • +Y (level 1) • -Y (level 0) • -Y (level 1) • +Z (level 0) • +Z (level 1) • -Z (level 0) • -Z (level 1)
`nw4c_udt` Present if user data is configured.	String	This is saved in the same format as NintendoWare intermediate files
`nw4rnw4c_mpi` Present if mipmaps are enabled but not when `Generate MipMap` is specified in Adobe Photoshop.	u8 array	All the image data except the highest level of the mipmap The image data section of the TGA file stores only the highest level mipmap data; all other levels of data are stored in this data block. For details, see section 3.1.1 About the Data in the nw4c_mpi Data Block.
`nw4c_gnm`	String	Name of the tool that exports the texture file Example: "NW4C_Tga for Photoshop 11.0.1"
`nw4c_gvr`	String	Version name of the tool that exports the texture file Example: "1.0.0"
`nw4c_psh` Present if data is exported by Adobe Photoshop.	Set of data blocks	Data specific to Adobe Photoshop For details, see section 3.2 Data Specific to Adobe Photoshop.
`nw4c_end`	No data	End of the additional information area This data block contains only a header that indicates the end of the additional information area. Ignore any data blocks located after this data block.

3.1.1 About the Data in the nw4c_mpi Data Block

Each texel contains 3 bytes (RGB) or 4 bytes (RGBA) of run-length compressed data. All the RGB or RGBA data are compressed into a group instead of channel by channel. Whether the data are RGB or RGBA can be determined from the header section of the TGA file.

Compressed data is stored sequentially from the second-highest level to the lowest level for each image.

Similarly to the `nw4c_txd` block, compressed image data in each direction of a cube map is stored sequentially from the second- highest level to the lowest level.

The original data represents the image from top to bottom. Each line of data represents the image from left to right.

The run-length compressed data is compressed one line at a time. Each compressed line consists of a group containing one control byte and a group of one or more sets of texel data. The most significant bit (MSB) of the control byte indicates whether the block contains a succession of texel data sets. The lower seven bits of the control byte represent the value (group length -1). If the MSB of the control byte is 1, the block contains only one set of texel data, and the set of data continues for just the group length. If the MSB of the control byte is 0, the block contains multiple sets of texel data that are equal in number to the group length, and the unmodified texel data is used.

3.2 Data Specific to Adobe Photoshop

Data that is unique to Adobe Photoshop and information related to filter plug-ins cannot be saved to the TGA file. Export plug-ins skip this data block.

This data block actually consists of a set of many data blocks. The format of the data block header is the same as the format described in Chapter 2 Additional Information Structure

The psh_pver block must always appear first. The other data blocks can appear in any order.

Table 3-2 Data Specific to Adobe Photoshop

Identifier	Data Type	Description
`psh_pver`	String	Version of data specific to Adobe Photoshop. This is currently set to 1.0.
`psh_gray` Present if the Photoshop image mode is grayscale	No data	This data block contains only a header. If this data block is present, the image mode will appear in grayscale when Photoshop opens the texture file.

Identifier	Data Type	Description
`psh_ctbl` Present if the Photoshop image mode is index color	u8 array	Photoshop color table data. This stores the color data in RGB order for the applicable number of colors. Each color consists of 24 bits of color data (8 bits for each R, G, and B). The size of this block (excluding the header) divided by three equals the number of colors in the color table. If this data block is present, the image mode will use index color when Photoshop opens the texture file.
`psh_cidx` Present if the Photoshop image mode is index color	u8 array	Color index data for each texel in Photoshop. (See the next section for more details.)
`psh_xpid` Present if the Photoshop image mode is index color with transparent color	u8 (1 byte)	Transparent color index (0 through 255) of the Photoshop color table. This data block is not present if a transparent color is not specified.
`psh_xppl` Present in an image that has only layers but no background or alpha channel.	No data	This data block is just a header. If this data block exists, the alpha values from TGA image data are applied to the layer transparency when a texture file is load by Adobe Photoshop.
`psh_gmip` Present when `Generate MipMap` is specified.	String	The number of mipmap levels to generate. This can be `all`, `2`, `3`, `4`, `5`, `6`, `7`, or `8`.
`psh_cldp` Present if a color depth filter is applied and the texture format is the same type as the color depth filter	String	Types of color depth filters: • `L4`: 4-bit luminance • `A4`: 4-bit alpha • `La4`: 4-bit luminance and 4-bit alpha • `Rgb565`: 5-bit R, 6-bit G, 5-bit B • `Rgb5_a1`: 5-bit R,G,B and 1-bit alpha • `Rgba4`: 4-bit R,G,B,A
`psh_etco` Present if the texture format is `ETC1` or `ETC1_A4`.	u8 (1 byte)	ETC compression method. • `0x00` is Fast • `0x01` is Medium • `0x02` is Slow • `0x03` is Fast Perceptual • `0x04` is Medium Perceptual • `0x05` is Slow Perceptual

Identifier	Data Type	Description
`psh_etc1` Present if the `ETC` filter is applied and the texture format is `ETC1`.	No data	This data block is just a header.
`psh_etca` Present if the `ETC` filter is applied and the texture format is `ETC1_A4`.	No data	This data block contains only a header.

3.2.1 Details of the psh_cidx Data Block

Each texel contains 1 byte of run-length compressed data. The entire image in Photoshop is compressed as a single image (this does not compress each mipmap level). The original data represents the image from top to bottom. Each line of data represents the image from left to right.

The run-length compressed data is compressed one line at a time. Each compressed line comprises a one control byte and a group of one or more texel data sets. The MSB of the control byte indicates whether the block contains a succession of texel data sets. The lower seven bits of the control byte represent the (group length -1). If the MSB of the control byte is one, then the block contains only one set of texel data, and the set of data continues for just the group length. If the MSB of the control byte is 0, the block contains multiple sets of texel data equal in number to the group length, and the texel data is used as is.

4 TGA File Restrictions

4.1 ID Field

The TGA file can store an ID field that contains arbitrary data after the 18-byte TGA header. The size of this ID field ranges from 0 through 255 bytes.

A TGA file that has additional information should include the 20 bytes of data shown in Table 4-1 at the beginning of the ID field.

Table 4-1 TGA File ID Field

Type	Description	Size
`char[16]`	Identifier and additional information version. Currently, this is always set to `NW4C_Tga Ver1.0`, with the last byte set to 0x00 The export plug-in checks for the identifier and version to determine whether there is additional information.	16 bytes
`u32`	The offset (little-endian) from the start of the additional information file. By using this offset, you can access the additional information without analyzing the TGA file image data.	4 bytes

The size of the ID field is specified in the first byte of the TGA header.

4.2 Image Data

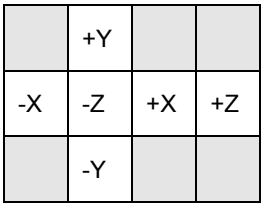
TGA file image data can be in several different formats, including RGB, index color, and grayscale.

However, NintendoWare for CTR uses only TGA files where the image data is in the RGB format since some 3DCG tools cannot use TGA files if the image data is in any of the other formats. Consequently, you should export files using the RGB format even when exporting from a paint tool that supports the index color format or the grayscale format.

The bits-per-pixel value should be set to either 24-bit RGB or 32-bit RGBA. The compression method should be set to either no compression or run-length compression.

Image data for a cube map is arrayed with the image for each view direction in a horizontal cross as shown in Figure 4-1.

Figure 4-1 Cube-Map Arrangement of TGA Image Data



The following table shows the coordinate axes for the TGA image data in each view direction.

Table 4-2 Cube-Map Coordinate Axes of TGA Image Data

View Direction	Up	Right
+X	+Y	+Z
-X	+Y	-Z
+Y	+Z	+X
-Y	-Z	+X
+Z	+Y	-X
-Z	+Y	+X

On a CTR system, the -Y axis points upward for images in the +X, -X, +Z, and -Z view directions. As a result, these are rotated by 180 degrees when they are stored as additional texture data (in `nw4c_txd`).

Revision History

Version	Revision Date	Classification	Description
1.1	2010/07/29	Changed	• Changed the format of the document.
1.0	2009/10/30		• Initial version.