Revolution SDK

Texture Palette Library (TPL) Files and TexConv

Version 1.1

2008/05/20

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

RVL-06-0357-001-B
Released: February 3, 2010

Table of Contents

Code

Tables

Figures

Revision History

Version	Revision Date	Description
1.1	2008/05/20	• Updated the document title from TPL Format Description to TPL Files and TexConv. • Deleted references to trademarks not mentioned in document. • Added details to Chapter 1 Introduction specific to what this document sets out to explain. • Updated the title of Chapter 2 from Features of TPL Format to Features and Advantages of TPL Files and TexConv. Also clarified the difference in formats supported by TPL files and TexConv. • Rewrote the explanation regarding TGA in section 3.1.2
1.0	2008/03/18	Initial version.

1 Introduction

The Texture Palette Library (TPL) image format for textures was part of the character pipeline SDK provided with the Nintendo GameCube development environment. The format is highly versatile and is even used for some of the GX library's sample demos. Although a small library was provided in the Revolution SDK to access TPL files, no information was given on creating new TPL files.

This document contains explanations of the following.

- The advantages of TPL files
- How to use the texture conversion application TexConv to convert data to the TPL file format

2 Features and Advantages of TPL Files and TexConv

TPL files and TexConv offer the following features and advantages.

- They can speed up disc read times by storing multiple textures in a single file.
- They can improve efficiency by storing multiple files that share temporal locality for an application.
- They can store many different texture maps, and color lookup tables for different formats, in a single file.
- TPL files can contain all texture formats supported on the Wii console (RGBA, IA, I, CI, CMPR) except for the Z texture format. (TexConv does not support certain formats.)
- All the texture data in a TPL file is 32-byte tiles, which can be used directly by the GX library.
- They can convert the input palette into the RGB565 and RGB5A3 output formats.
- They can generate mipmaps.
- They can compress S3TC textures.
- They can perform automatic color conversion (for example, from RGBA to IA, CI to RGB, or RGB to RGBA).

2.1 Optimizations for Texture File Creation

When creating textures, designers will want to add many detailed revisions to the image file of each original picture until they are satisfied with the final appearance. To make the working process more efficient, it is necessary to remove the overhead of re-creating an entire TPL file every time the designer adds a small change to a texture.

The `TexConv` texture converter has a feature to improve conversion speed by updating only those parts of a TPL file that have changed since `TexConv` was last run. When `TexConv` is run, it compares each of the new images and palettes contained in the TPL file that was created most recently. If a data block matches, the currently existing one will be copied directly to a new TPL file, skipping the conversion process.

2.2 Support for Color Index Textures

Compression and animation are two important reasons to use a color index (CI) texture. However, it is likely that many developers use the CMPR (S3TC) texture compression format instead of the CI format because the former can provide high-quality compressed textures from true color images. Consequently, `TexConv` was designed without support for creating CLUTs (color lookup tables) from true color images. Do not confuse this with the `TexConv` feature that automatically converts from a CI input format to an RGB output format.

In addition, `TexConv` does not support encoding methods for CI texture animations. This is because it is difficult to define a CLUT encoding method for CI texture animations. For the most part, these encoding methods are unique to the animation being requested, and the relevant animation techniques themselves are unique to each game. The result of all this is to show developers how, as a result of the simple act of supporting CI textures in `TexConv` alone, CLUT and CI textures can be taken and used on the Wii console; this is not intended to be the ideal way to create CI textures specialized for games.

`TexConv` only supports CI8, but internally it stores color index textures in 16-bit format. The output index is copied directly from the input index. No bit masks or bit shifts are used. Consequently, it is the developer's responsibility to ensure that the input color index falls precisely within the output bit range. For example, if the output format is CI8, the input index must be between 0 and 255.

`TexConv` stores palette entries in RGBA format. Palettes can be output in either RGB565 or RGB5A3 format. IA8 is not supported as a palette entry.

3 Converting to TPL Files

`TexConv` converts to TPL files according to script file specifications.

3.1 Script File

The following is a sample texture conversion script file.

Code 3-1 Texture Conversion Script

```
                                ; Comments begin with a semicolon and continue
                                ; until the end of the line

path = c:/level1/mario/         ; Set the directory path at which files exist
                                ; If path = NULL, the full path is required in
                                ; the file name
                                ; path is pre-concatenated to all subsequent
                                ; file names
                                ; path can be set on any line
                                ; path can be absolute or relative

file   0  = marioHead_rgba8.tga ; The complete file name is
                                ; c:/level1/mario/marioHead_rgba8.tga
image  0  = 0, 0, RBGA8         ; With image0[RGB]=file0 and
                                ; image0[A]=file0, convert to RGBA8
texture 0  = 0, x               ; texture 0 of the TPL file
                                ; The image index is image0
                                ; The CLUT index is [x] because
                                ; it is not a color-indexed texture

path      =  d:/temp/mario/     ; Change the path
file   1  =  marioArm_rgb565.tga ; The complete file name is
                                ; d:/temp/mario/marioArm_rgb565.tga
image  1  =  1, x, RGB565, 0, 3, 0
                                ; Create mipmaps 0 to 3 (four LODs),
                                ; and remap to 0
texture 1  =  1, x

file   2  =  marioFoot_ci8.tga
image  2  =  2, x, CI8          ; Convert to CI format
palette 0  =  2, RGB565         ; Create palette0 from image2
                                ; Convert palette entries to RGB565 format
texture 2  =  2, 0              ; texture 2[RGB] = image 2,
```

```
                                ; texture 2[CLUT] = palette 0

path       = NULL               ; Unspecified path
file   3   =  c:/marioBody_i8.tga
                                ; The full path is now required
image  3   =  3, x, I8, GX_REPEAT, GX_REPEAT
                                ; TexConv will automatically convert
                                ; from RGB input to intensity output.
                                ; Wrap mode can be set in S and T
                                ; to repeat.
                                ; Wrap mode can be set in the same way
                                ; as mipmap arguments.
texture 3  =  3, x
```

3.1.1 Basic Formatting Rules

The following are basic rules for script files.

- Spaces are ignored. “image1=1” and “image 1 = 1” are considered to be identical.
- Text is parsed line-by-line. A line ends with a carriage return and can be no longer than 255 characters.
- Images, textures, palettes, and their indices can be listed in any order. `TexConv` sorts each list in ascending order before conversion.
- Comments begin with a semicolon and continue until the end of the line.
- Unspecified elements are depicted by “x.” For example, “image 1 = 1, x, RGB565” describes an image without an alpha plane. A texture without a palette is described by “texture 1 = 1, x”.
- The directory path is specified either as an absolute path or as a relative path from the current directory.

3.1.2 Commands

These are explanations of script file commands recognized by `TexConv`.

3.1.2.1 Command to Change the Reference Directory Path

```
Path = directory_path
```

Input files exist in the `directory_path` directory. All file names are attached to the end of `directory_path`.

The content of `path` will be maintained until changed. Set `path` equal to 0 or NULL to disable the path.

3.1.2.2 Command to Specify the Reference File

```
File file_id = file_name
```

The `file_id` is the number referred to by the `file_name` file used by the image and texture commands. The number is valid for any positive integer, including 0.

When path is disabled, `file_name` must include the full path. The path is given either as an absolute path or a relative path from the current directory.

When path is configured, `directory_path` is connected to `file_name` in advance.

3.1.2.3 Command to Specify the Reference Image

```
Image image_id =  rgb_image_file_id, alpha_image_file_id,
                  format [,start_lod, end_lod, remap_lod] [,wraps, wrapT]
```

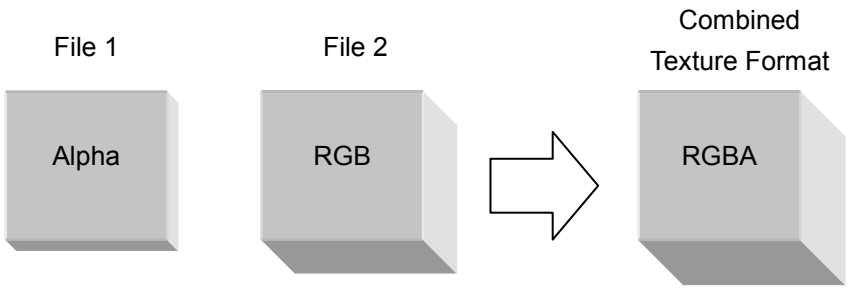
The `image_id` is the number the image used by the texture command will refer to. The number is valid for any positive integer, including 0.

The RGB portion of this image is contained in the `rgb_image_file_id` file.

The alpha portion of this image is contained in the `alpha_image_file_id` file.

As described above, the image command is extremely powerful. An output RGBA image can be constructed from two different files using `rgb_image_file_id` and `alpha_image_file_id`. TGA has complete support for alpha planes, so this feature will probably not be used.

Figure 3-1 Creating an RGBA Image from Two Different Files



The output texture format is specified in `format`.

Specifiable formats are shown in Table 3-1.

Table 3-1 List of Specifiable Texture Formats

Format	Format Details
I4	4-bit intensity
I8	8-bit intensity
IA4	8-bit intensity + alpha (4 + 4)
IA8	16-bit intensity + alpha (8 + 8)
CI8	8-bit color index

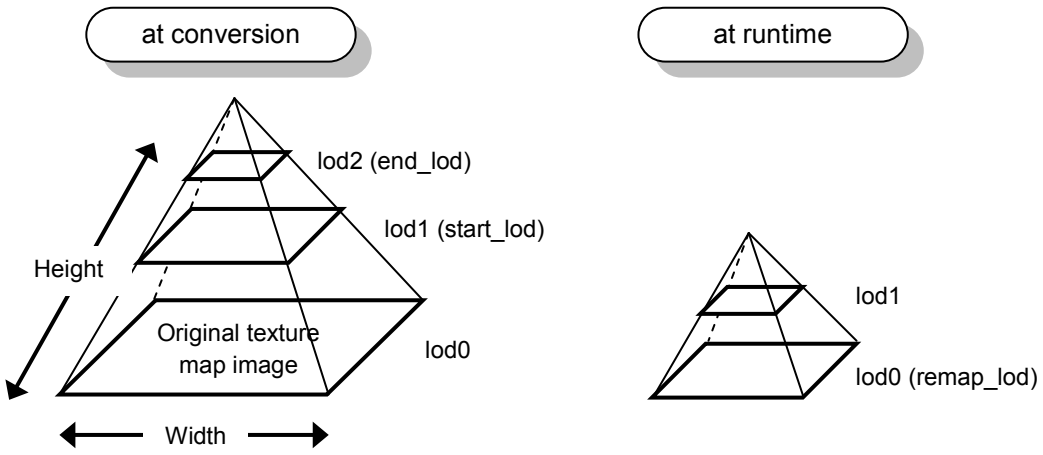
Format	Format Details
RGB565	16-bit RGB
RGB5A3	16-bit RGB + alpha (RGB555 or RGBA4443)
RGBA8	32-bit RGBA
CMPR	Compressed format with 4 bits per texel

The following are optional arguments and only required when creating mipmaps:

- `[, start_lod, end_lod, remap_lod]`. The mipmap levels that must be generated by `TexConv` are specified by `start_lod` ; `end_lod`. `remap_lod` specifies how to map to the LOD at runtime. In general, `remap_lod` is 0.
- The optional `[, wraps, wrapT]` arguments are added to overwrite wrap mode in the s and t directions. The following values can be applied: `GX_REPEAT`, `GX_CLAMP` and `GX_MIRROR`. By default, when the height and width dimensions are a power of 2, this is `GX_REPEAT`; otherwise it is `GX_CLAMP`.

Figure 3-2 shows how to generate 2 LOD levels and remap to `lod0` at runtime.

Figure 3-2 Generating and Remapping LOD Levels



3.1.2.4 Command to Specify Generated Palette Entries

```
palette CLUT_palette_id = CLUT_file_id, CLUT_format
```

`CLUT_palette_id` is the number of the palette entry used by the texture command. The number is valid for any positive integer, including 0.

`CLUT_file_id` is the number of the image to use when creating this palette entry.

The two palette entry formats supported by `CLUT_format` are RGB565 and RGB5A3.

3.1.2.5 Command to Specify Generated Textures

```
texture texture_id = image_id, CLUT_palette_id
```

The `texture` command precisely defines a texture in the TPL file. The value of `texture_id` must be

a positive integer of 0 or greater.

This texture represents an image with number `image_id`.

`CLUT_palette_id` is the number of the palette entry used by this texture. This parameter is `x` for textures that do not use the color index format.

3.2 List Order

When script file parsing begins, all image, palette, and texture lists are started as well. The first word (for example, image, palette, or texture) on a text line is recognized, and a new element is added to the appropriate list.

`TexConv` sorts each list in ascending order when it has finished parsing the script file. As a result, images, textures, palettes, and their indices can be listed in any order within the script file. In other words, although the images, textures, and palettes are listed interchangeably in Code 3-1, all the images could instead be listed as a group after all of the palettes.

Palettes and texture images are each stored in the TPL file in the order that they occur in their respective sorted lists.

Normally, the list's index matches the position in the sorted list, so that an index of 0 is the start of the list, and an index of (n-1) is the end of the list. Although this is convenient for commenting out several elements in a script file when determining the optimal TPL file structure, be aware that the script file may specify an index that does not match the position in the ordered list. Once the final TPL file structure has been fixed, the script file indices must be changed so that they are continuous from 0 to (n-1). Even though this is not mandatory, it is strongly recommended to avoid confusion when accessing textures at runtime.

3.3 Usable Input Image Files

`TexConv` uses image files in the Truevision Graphics Adapter (TGA) format as its input.

The TGA format was selected as the format to support because it has a long history, can be imported and exported by many graphics tools, and can store many texture formats, including true color, intensity, intensity alpha, color-indexed images, and 256-color palettes.

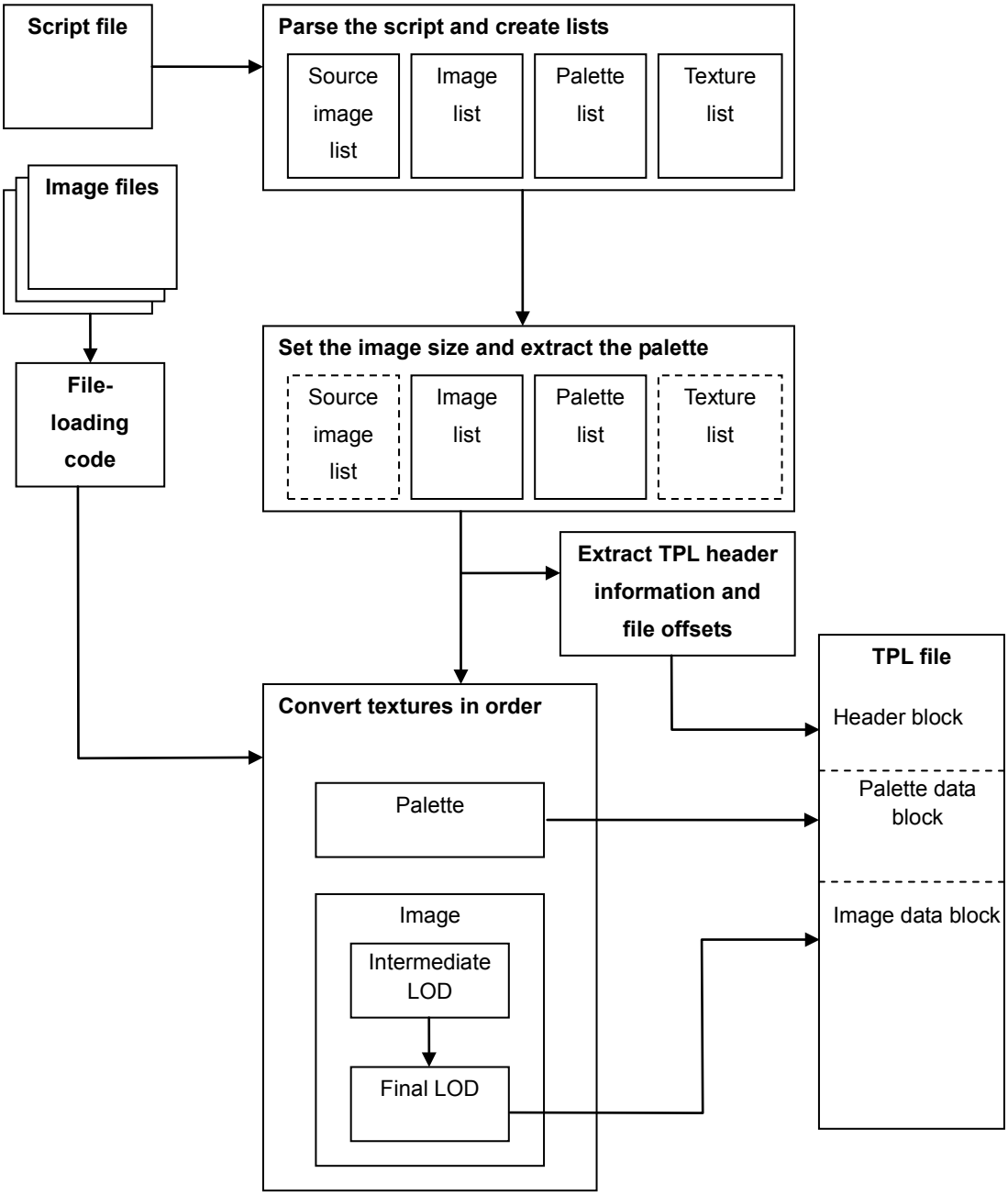
For more information on TGA files, see the *Encyclopedia of Graphics File Formats, 2nd Ed.* (ISBN 1-56592-161-5), pp. 860-879.

3.4 Conversion Process Flow

This section provides the complete source code for `TexConv` and an overview of the major processes performed by `TexConv`, from the original TGA file to conversion into a TPL file.

Figure 3-3 shows the logical flow of TPL file creation with `TexConv`.

Figure 3-3 Conversion Process Flow for TPL Files



TPL file conversion can be broken up into three steps.

3.4.1 Input

The script file is loaded and parsed. Error checking confirms that all of the data is consistent and all of the requested files exist. If there is a problem with any of these, the program will output an error message and quit.

3.4.2 Setup

`TexConv` collects input data and organizes it for conversion. `TexConv` creates and manages the following four information lists in this step.

- **Source image list**

  List of all of the original input files, along with size and format information.

- **Image list**

  Contains information on the final images, such as how to combine different files. For example, a single image can take RGB information from one source and alpha information from another source.

- **Palette list**

  Maintains palette information from the source data. Palettes are extracted from the source files and stored here.

- **Texture list**

  Contains combined information for making an image with a palette in the output TPL file.

3.4.3 Conversion and Exporting

`TexConv` reads a texture list from beginning to end and converts the original data into a format that can be handled by the Wii. Preconverted data will be used if it exists, skipping the conversion process. `TexConv` generates mipmap levels as necessary during conversion. A new TPL file will be output each time a texture has finished converting and is generated. `TexConv` will close the TPL files after it is done processing all textures in the texture list. This completes runtime library preparations.

3.5 Cache File

In the process of creating a texture, designers will repeatedly edit, convert, and then preview. In this case, almost all of the textures within the TPL file will probably be unchanged since the previous conversion results. `TexConv` saves conversion time by making block copies of unchanged textures from the previous TPL file and converting only the modified textures.

In order to determine if an image or palette needs to be converted, `TexConv` maintains a cache file (CSF) that describes the content of the previous TPL file. `TexConv` recognizes whether there are changes by using a CSF file, the previous TPL file, and the source image data; if nothing has changed, it copies the data that exists in the cache file.

The CSF file is saved in `C:/Temp/tplCache.csf`.

The CSF file itself does not need to be read or modified. However, if this is necessary, the CSF file format is entirely explained in comments and structures at the beginning of the following file.

```
$(REVOLUTION_SDK_ROOT)/build/libraries/tc/src/TCTPLToolBox.cpp
```

The following situations will result in a complete reconversion.

- The CSF file does not exist.
- The previous TPL file does not exist.
- The previous TPL file has been changed since it was created.
- The version number of the previous TPL file does not match the current code version number.
- The previous TPL file and the new TPL file do not have any common images or palettes.

When determining whether to convert or to copy data blocks, `TexConv` checks whether the previous TPL file has image or palette blocks with the same file name and attributes as the images or palettes to convert. This means that even if the script file index or TPL filename is changed between conversions, the converted data can be used.

Table 3-2 lists situations that will result in a partial reconversion.

Table 3-2 Causes for Partial Reconversion

Situation	TexConv Behavior
There was a change to the last update time of the source image file.	Reconvert images and palettes related to this file.
Changes were made to the script file content.	Reconvert the relevant images and palettes.
Additions were made to the script file content.	Convert new images and palettes.

Note: To determine whether data can be reused, `TexConv` checks script file tokens and the timestamp at which changes were added to a file. The check does not extend to the actual bits in the source image. This will cause errors if the conversion code is rewritten between consecutive calls. In this case, the previous TPL data can no longer be synchronized with the current conversion code. To force a complete reconversion, delete either the CSF file or the previous TPL file.

3.6 How to Use TPL Files

Use the Revolution SDK's texture palette library for applications that use textures stored in TPL files.

For more information on the texture palette library, see the *Revolution Function Reference Manual*.