Revolution

THP Library Guide

Version 1.00

© 2006 Nintendo

"Confidential"

These coded instructions, statements, and computer programs contain proprietary information of Nintendo of America Inc. and/or Nintendo Company Ltd., and are protected by Federal copyright law. They may not be disclosed to third parties or copied or duplicated in any form, in whole or in part, without the prior written consent of Nintendo.

© 2006 Nintendo

TM and ® are trademarks of Nintendo.

Dolby, Pro Logic and the Double-D symbol are trademarks of Dolby Laboratories.

IBM is a trademark of International Business Machines Corporation.

Roland GS Sound Set is a trademark of Roland Corporation U.S.

All other trademarks and copyrights are property of their respective owners.

Table of Contents

Code Examples

© 2006 Nintendo
CONFIDENTIAL

Tables

Figures

Equations

© 2006 Nintendo
CONFIDENTIAL

Revision History

Revision No.	Date Revised	Items (Chapter)	Description
1-Mar-2006	3/1/2006	-	First release by Nintendo of America Inc.

© 2006 Nintendo
CONFIDENTIAL

© 2006 Nintendo
CONFIDENTIAL

1 Introduction

1.1 Overview

The THP library is designed for the playback of movies on Revolution. By using the THP library, movie data comprising interleaved video data and audio data (henceforth called THP movie data) can be played on Revolution.

The format of the video data handled by the THP library (henceforth called THP video data) is customized for rapid decoding by the Revolution. The THP library has been optimized to a high extent for decoding of this THP video data.

Note: For increasing speed, the THP library uses certain functions unique to the Broadway CPU (i.e., locked cache as well as GQR). Please be careful when using the THP library.

Because THP movie data can be decoded so rapidly, it is extremely well suited for applications inside the game. For example, a movie displayed to a screen size of 640 x 480 pixels at a frame rate of 60 frames per second that has been compressed to around 1 bit per pixel, requires around a 40% load on the CPU when it is being decoded. If the frame rate is 30 frames per second, the compression ratio can be lowered to 5 or 6 bits per pixel (thereby raising picture quality).

Note: The actual compression ratio is influenced by limits on the speed at which data is read from the optical disc. The slowest optical disc read speed is 3 Mbytes per second. Accordingly, for a movie displayed at 30 frames per second, you should compress the data to 102.4 K or less per frame.

By utilizing high-speed THP movie data, you can decode and display a movie in real-time while rendering objects in front of the movie.

THP video data is created from the conversion of normal JPEG data. The JPEG data is converted into THP video very quickly and with no loss in picture quality from the original JPEG data.

Through the use of THP movie data, the developer can estimate how much time is needed to decode movie data and what kind of picture quality will be realized on the Revolution when the data is actually decoded.

The THP movie data's audio data (henceforth called the THP audio data) is compressed in the same format as the Revolution audio system's `DSPADPCM`. This THP audio data is appropriately processed in order to synchronize the video data and the audio data as required for the movie.

In summary, use of the THP library enables developers to playback high picture-quality and sound-quality movies with only a minimum burden on the CPU.

1.2 Sample Program

Every frame of THP video data is compressed independently of every other frame. Because of this, movie playback is easy to realize with the THP library compared to other movie libraries, which require that information from prior and subsequent frames be referenced when decoding the present frame.

However, a slightly complicated procedure is required, in order to achieve appropriate synchronization of the video data and the audio data during movie playback.

To avoid complexity, two movie players have been prepared that hide this procedure of the THP library, making the playback of movies even simpler.

The first movie player is an extremely simple player designed to provide a basic understanding of how to use the THP library.

The second movie player is an advanced player that assumes use in applications. This advanced player has been created with flexibility to meet the various needs of the game with regard to movie playback.

© 2006 Nintendo
CONFIDENTIAL

The THP library was intentionally created so that movies could be actively utilized inside even ordinary game scenes. However, the game system can have a large effect on the movie player when it is used in this way. Thus, even though the THP movie player library is easy to use, the developer should modify the configuration of the player to suit the environment.

Source code is distributed along with the movie players. Feel free to modify this source to suit the THP movie data playback environment.

1.3 The Organization of this Document

This document is organized as follows:

- Chapter 2, Using the Simple Player to Play Back THP Movie Data, explains the minimal procedure required for playing THP movie data on the Revolution.
- Chapter 3, How to Create THP Movie Data, explains how to create THP movie data.
- Chapter 4, Cautions, reviews the use of the THP library.
- Appendix A. includes a list of API functions for the THP library.

© 2006 Nintendo
CONFIDENTIAL

2 Using the Simple Player to Play Back THP Movie Data

The THP simple player (henceforth called the Simple Player) is a THP movie player that provides the minimal set of functions needed for playing THP movie data on the Revolution.

This section explains how to use the Simple Player.

2.1 Source Files

The Simple Player is supplied as the sample program `THPSimple` in the THP library (`build/demos/thpdemo`). The Simple Player is comprised of the following source files:

- `src/THPSimple/THPSimple.c`

Describes the various API functions of the Simple Player. The player calls the THP library's low level API function `THPVideoDecode` when decoding THP video data and the low level API function `THPAudioDecode` when decoding THP audio data.

- `src/THPSimple/THPDraw.c`

The `THPVideoDecode` function, which is called by the Simple Player when decoding THP video data, outputs the decoded data in YUV texture format. This file describes the functions for drawing this data to the EFB via the graphics processor.

- `src/THPSimple/main.c`

This is a simple sample of THP movie data for playback using the Simple Player.

- `include/THPSimple.h`

Contains the definitions for the various API functions of the Simple Player.

- `include/THPDraw.h`

Contains the definitions for the functions used for drawing the decoded YUV texture format data to the EFB via the graphics processor.

© 2006 Nintendo
CONFIDENTIAL

2.2 How to use the Simple Player

Below is a simple program using the Simple Player for playback of THP movie data. This program is described in "`build/demos/thpdemo/src/THPSimple/main.c`".

Code 2-1 Example of Simple Player Use (main.c)

```
1:      #include <demo.h>
2:      #include <stdlib.h>
3:      #include <string.h>
4:      #include "THPSimple.h"
5:      #include "axseq.h"
6:
7:      void main(void)
8:      {
9:          u32             size, x, y, count;
10:         s32             frame, start, vol, audioTrack;
11:         u16             buttonDown, button;
12:         u8              *buffer;
13:         GXRenderModeObj *rmode;
14:         THPVideoInfo    videoInfo;
15:         THPAudioInfo    audioInfo;
16:
17:         DEMOInit(&GXNtsc480Int);
18:
19:         AIInit(NULL);
20:
21:         AXSeqSetup();
22:
23:         THPSimpleInit(THP_MODE_WITH_AX);
24:
25:         GXSetDispCopyGamma(GX_GM_1_0);
26:
27:         // Open the THP File
28:         if (THPSimpleOpen("thpdemo/fish_mlt.thp") == FALSE)
29:         {
30:             OSHalt("THPSimpleOpen fail");
31:         }
32:
33:         THPSimpleGetVideoInfo(&videoInfo);
34:         THPSimpleGetAudioInfo(&audioInfo);
35:
36:         rmode = DEMOGetRenderModeObj();
37:
38:         x = (rmode->fbWidth   - videoInfo.xSize) / 2;
39:         y = (rmode->efbHeight - videoInfo.ySize) / 2;
40:
41:         // Set aside work area
42:         size = THPSimpleCalcNeedMemory();
43:
44:         buffer = (u8 *)OSAlloc(size);
45:
46:         if (!buffer)
47:         {
48:             OSHalt("Can't allocate the memory\n");
49:         }
50:
51:         THPSimpleSetBuffer(buffer);
52:
53:         OSReport("\n########################################\n");
54:         OSReport("Push A button     : restart the movie, select audio track at random\n");
55:         OSReport("Push B button     : play/stop midi file using AX\n");
56:         OSReport("Push Start button : application end\n");
57:         OSReport("Pad up            : volume up\n");
```


© 2006 Nintendo
CONFIDENTIAL

```
58:         OSReport("Pad down          : volume down\n");
59:         OSReport("########################################\n");
60:
61:     RESTART:
62:         // Load for playback
63:         if (THPSimplePreLoad(THP_PLAY_LOOP) == FALSE)
64:         {
65:             OSHalt("THPSimplePreLoad fail");
66:         }
67:
68:         audioTrack = (s32)(OSGetTick() % audioInfo.sndNumTracks);
69:
70:         start = 1;
71:
72:         count = VIGetRetraceCount();
73:
74:         while(1)
75:         {
76:             DEMOPadRead();
77:
78:             buttonDown = DEMOPadGetButtonDown(PAD_CHAN0);
79:             button     = DEMOPadGetButton(PAD_CHAN0);
80:
81:             DEMOBeforeRender();
82:
83:             // Decode 1 frame of THP data
84:             if (THPSimpleDecode(audioTrack) == VIDEO_DECODE_FAIL)
85:             {
86:                 OSHalt("Fail to decode video data");
87:             }
88:
89:             // Draw decoded THP video data
90:             frame = THPSimpleDrawCurrentFrame(rmode, x, y, videoInfo.xSize, videoInfo.ySize);
91:
92:             while (VIGetRetraceCount() < count + 1)
93:             {
94:                 VIWaitForRetrace();
95:             }
96:
97:             DEMODoneRender();
98:
99:             count = VIGetRetraceCount();
100:
101:            if (start)
102:            {
103:                // Enable audio output
104:                THPSimpleAudioStart();
105:                start = 0;
106:            }
107:
108:            if (buttonDown & PAD_BUTTON_A)
109:            {
110:                // Stop playback and restart
111:                THPSimpleAudioStop();
112:                THPSimpleLoadStop();
113:                goto RESTART;
114:            }
115:
116:            if (buttonDown & PAD_BUTTON_START)
117:            {
118:                // End playback, and leave main loop
119:                THPSimpleAudioStop();
120:                THPSimpleLoadStop();
121:                THPSimpleClose();
122:
```

© 2006 Nintendo
CONFIDENTIAL

```
123:                OSFree(buffer);
124:
125:                break;
126:            }
127:
128:            if (buttonDown & PAD_BUTTON_B)
129:            {
130:                if (GetSeqState())
131:                {
132:                    SeqStop();
133:                }
134:                else
135:                {
136:                    SeqPlay();
137:                }
138:            }
139:
140:            if (button & PAD_BUTTON_UP)
141:            {
142:                vol = THPSimpleGetVolume() + 1;
143:                if (vol > 127)
144:                {
145:                    vol = 127;
146:                }
147:                THPSimpleSetVolume(vol, 0);
148:            }
149:
150:            if (button & PAD_BUTTON_DOWN)
151:            {
152:                vol = THPSimpleGetVolume() - 1;
153:                if (vol < 0)
154:                {
155:                    vol = 0;
156:                }
157:                THPSimpleSetVolume(vol, 0);
158:            }
159:        }
160:
161:        THPSimpleQuit();
162:
163:        VIWaitForRetrace();
164:
165:        OSHalt("application end\n");
166:
167:        return;
168:    }
```

Line 14

The `THPVideoInfo` structure is declared. This structure maintains the vertical and horizontal size of the THP video data. The members of this structure are set by the `THPSimpleGetVideoInfo` function (see line 33).

Line 19:

Before initializing one of the Revolution audio libraries, AX library, the `AIInit` function is called to initialize the audio interface.

Line 21:

The `AXSeqSetup` function initializes AX internally, and makes preparations for the use of the AX sequencer.

© 2006 Nintendo
CONFIDENTIAL

Line 23:

The `THPSimpleInit` function must be called first, before using any of the Simple Player API functions. In addition to initializing the Simple Player's control structure (the `THPSimple` structure), the `THPSimpleInit` function enables locked cache and calls the THP library's low level API function `THPInit`. It also registers a callback function in the Revolution audio interface for the playback of THP audio data. When used at the same time as AX, initialization functions for the audio library (`AXInit`) need to be called before `THPSimpleInit` is called. When used at the same time as AX, `THPSimpleInit` needs to be called while sound output of AX is set to produce no sound. This sample program specifies **`THP_MODE_WITH_AX`** for the argument of `THPSimpleInit` for simultaneous use of AX.

Line 28:

This opens the THP movie data ("`fish_mlt.thp`") specified by the argument. The `THPSimpleOpen` function calls the `DVDOpen` function to open the THP movie data specified by the argument, and calls the `DVDRead` function to read the header portion of that data. After that, the `THPSimpleOpen` function analyzes the header portion to check that the data specified by the argument is valid THP movie data.

Line 33:

This obtains information regarding THP video data from the header portion of the THP movie data that was loaded into main memory in line 28, and stores it in the `THPVideoInfo` structure specified by the argument. The information acquired here is utilized by the application (in this case, the sample program described in `main.c`).

Line 34:

This obtains information regarding THP audio data from the header portion of the THP movie data that was loaded into main memory in line 28, and stores it in the `THPAudioInfo` structure specified by the argument. The information acquired here is utilized by the application (in this case, the sample program described in `main.c`).

Lines 38, 39:

Calculates the draw-location (upper left corner coordinates) of the decoded data so the THP video data will be drawn centered in the screen when the THP movie data is played back.

© 2006 Nintendo
CONFIDENTIAL

Line 42:

Calculates the size of the work region to be used by the Simple Player. The `THPSimpleCalcNeedMemory` function returns a value that is the sum of the size of the buffer used when the THP movie data is read from the optical disc, the size of the YUV texture buffer used when the THP library's low level API function decodes the THP video data, the size of the buffer used by the THP audio data, and the size of the work region used by the THP library's low level API functions. The formula used for this calculation is given below:

Equation 2-1 Size of the Work Region for the THP Simple Player

```
// Size of buffer for reading the THP movie data
size  = OSRoundUp32B (Maximum data size of THP frame data) * READ_BUFFER_NUM;

// Size of Y texture buffer
+ OSRoundUp32B (THP video data horizontal size x vertical size)

// Size of U texture buffer
+ OSRoundUp32B (THP video data horizontal size x vertical size/4)

// Size of V texture buffer
+ OSRoundUp32B (THP video data horizontal size x vertical size/4)

// Size of buffer for THP audio data
+ OSRoundUp32B (THP audio data's maximum number of samples x 4) x  AUDIO_BUFFER_NUM

// Size of work region used by THP library's low level API functions
+ THP_WORK_SIZE;
```

Notes:

- The `THPConv` tool, which is used to create THP movie data, automatically sets the THP frame data's maximum size when it is creating THP movie data. This maximum size differs for every set of THP movie data. In addition, for movies that play back a long time or have frequently changing screens, this value will differ largely from the average size that can be calculated from the THP movie data file size. For example, in the sample data that comes with the THP library (`rebirth.thp` (`dvddata/thpdemo/rebirth.thp`), the average size is 39,552 bytes, and the maximum size is set to 77,728 bytes. This type of localized fluctuation in data size can have a large influence not only on the required buffer size for the Simple Player, but also on the data transfer speed from the optical disc and the CPU load required for decoding. You need to work to have a good balance when creating movies.
- The `THPConv` tool also automatically sets the maximum size for the THP audio data. Unlike for the video value, this value for the THP audio data does not fluctuate largely in every frame.
- The values for `READ_BUFFER_NUM` and for `AUDIO_BUFFER_NUM` are defined in the THP Simple Player's header file `THPSimple.h` (`build/demos/thpdemo/include/THPSimple.h`).

Code 2-2 Definition of `READ_BUFFER_NUM / AUDIO_BUFFER_NUM` in THP Simple Player

```
#define READ_BUFFER_NUM  10
#define AUDIO_BUFFER_NUM 3
```

Note: The value of `THP_WORK_SIZE` is defined in the THP library's header file `thp.h` (`include/dolphin/thp.h`).

Code 2-3 Definition of `THP_WORK_SIZE` in THP library

```
#define THP_WORK_SIZE 0x1000
```


© 2006 Nintendo
CONFIDENTIAL

Line 44:

Allocates a work region for the Simple Player.

Line 51:

Registers in the Simple Player the work region that was allocated in line 44.

Line 63:

Several frames worth of THP movie data are read from the optical disc before the THP movie data begins to play. The `THPSimplePreLoad` function calls the `DVDRead` function to read the `READ_BUFFER_NUM` number of frames of THP movie data. The method of playback for the THP movie data is set in the argument of the `THPSimplePreLoad` function. `THP_PLAY_LOOP` is specified to loop the THP movie data, and `THP_PLAY_ONESHOT` is specified to play the THP movie data only once in **one shot**. Both `THP_PLAY_LOOP` and `THP_PLAY_ONESHOT` are defined in the header file of the Simple Player `THPPlayerCommon.h` (`build/demos/thpdemo/include/THPSimple.h`).

Code 2-4 Definitions of the Argument of the `THPSimplePreLoad` function

```
#define THP_PLAY_ONESHOT 0x00
#define THP_PLAY_LOOP    0x01
```

Line 68:

Determines the audio track played back from the time base register value.

Line 70:

The **`start`** flag is set, in order to get the timing for the start of THP audio data playback (see lines 101-106).

Line 84:

Decodes the THP video data. The decoded data is stored in YUV texture format in a buffer maintained internally by the Simple Player. If audio data is interleaved in the THP movie data, then the audio data from the audio track specified with `audioTrack` is also decoded.

Line 90:

Draws the THP video data that was decoded in line 84. For drawing, the `THPSimpleDrawCurrentFrame` function takes the YUV texture-format data decoded by the `THPSimpleDecode` function and pastes it onto polygons.

Lines 92-97:

Writes from the EFB to the XFB and waits for two retraces.

When using the Simple Player, the loop portion in which the various Simple Player API functions are called and the frame buffer is drawn (i.e., the portion between lines 74-159), must have the same frame rate as that of the THP movie data being played. For example, if the Simple Player is incorporated into a loop where objects are being drawn at a frame rate of 60 frames per second, then the frame rate for that object rendering must drop to 30 frames per second, which is the frame rate for playback of THP movie data.

Note: This sample program is for the playback of THP movie data at a frame rate of 29.97 frames per second.

Lines 101-106:

Allows start of playback of THP audio data. If start of playback is allowed by the `THPSimpleAudioStart` function, decoded THP audio data is sent to the Revolution audio interface. When audio data plays back interleaved THP movie data, `THPSimpleAudioStart` needs to be called once immediately after the start of playback.

© 2006 Nintendo
CONFIDENTIAL

The best location for the call to this function is immediately after the first frame (line 90) of THP movie data drawn by the `THPSimpleDrawCurrentFrame` function is displayed on the screen (at or after line 97).

Lines 108 - 114

Stops playback of THP movie data and restarts playback of that data at the beginning. To do this, first the `THPSimpleAudioStop` function (line 111) is called to cancel permission to play the audio data. When the denial is received, decoded THP audio data will not be sent to the Revolution audio interface. Next, the `THPSimpleLoadStop` function (line 112) is called to stop preloading of THP movie data and to initialize the `THPSimple` structure. Doing so returns the Simple Player to its initial status.

Lines 116 - 126:

Terminates playback of THP movie data. First the `THPSimpleAudioStop` function (line 119) is called to stop playback of audio data. Next the `THPSimpleLoadStop` function (line 120) is called to return the Simple Player to its initial status. Then the `THPSimpleClose` function (line 121) is called to close the THP file.

Lines 128 - 138:

Uses AX to play back or stop MIDI file.

Lines 140 - 148:

Turns up the playback volume of THP audio data.

Lines 150 - 158:

Turns down the playback volume of THP audio data.

Line 161:

Terminates the Simple Player. The `THPSimpleQuit` function calls the `LCDisable` function to disable locked cache and to return the Simple Player's internal state to the status it was in before the `THPSimpleInit` function was called. To use the Simple Player again, the process must begin with the calling of the `THPSimpleInit` function. When using AX simultaneously, call the `THPPlayerQuit` function while sound output of AX is set to not produce any sound.

© 2006 Nintendo
CONFIDENTIAL

3 How to Create THP Movie Data

3.1 Overview

The format of THP movie data has specifications unique to the Revolution. THP movie data is composed of THP video data and THP audio data, and both types of data are interleaved in every frame. THP movie data has high extensibility, and can also incorporate a third type of data besides THP video data and THP audio data.

A unique format is employed for the THP video data. It has been customized for rapid decoding by the Revolution. The JPEG data is converted into THP video data very quickly, with no loss in picture quality from the original JPEG data.

The THP audio data is compressed in the Revolution audio system's DSP ADPCM format. There are no restrictions on playback frequency. This THP audio data supports both monaural and stereo playback.

The utility `THPConv` is available for use in creating THP movie data. The `THPConv` tool supports sequential JPEG files (see section 3.2.1). The tool also has a function that can be used for replacing the audio data that is already incorporated in the THP movie data (see section 3.2.2).

3.2 Usage

The `THPConv` tool is a Win32 console application. The tool is used differently depending on the format of the input data. The following sections explain how to use the THPConv tool for each separate input data format.

Addendum: The usage can be displayed by executing the `THPConv` tool without any options.

3.2.1 Converting Serial JPEG files into THP movie data

Use `THPConv` tool as follows when converting serial JPEG files to THP movie data.

Code 3-1 `THPConv` Syntax when Converting Serial JPEG files

```
THPConv.exe -j <*.jpg> -d <outputfile> [-<option> <argument>]
```

Table 3-1 `THPConv` Arguments for Serial JPEG Files

Argument	Description
`-j` `<*.jpg>`	Specifies the input files (serial JPEG files). Wildcard characters (*) can be used. This argument must be specified.
`-d` `<outputfile>`	Specifies the output file (the THP file). This argument must be specified.

The `THPConv` tool supports the options in :

Table 3-2 `THPConv` Options for Conversion of Serial JPEG Files

Option	Description
`-s` `<wavefile> <wavefile> <wavefile> ... ... ...`	Specifies an audio file (a wav file) to convert into THP audio data. If multiple files are specified in the argument, the data is allocated to the various audio tracks in the order specified.

© 2006 Nintendo
CONFIDENTIAL

Option	Description
-r <framerate>	Specifies the movie frame rate. The frame rate can be set in the range of 1.0 to 59.94. If no value is specified, then the default value (29.97) is utilized.
-o	With THP movie data, the offset to each frame's data can be stored as a table (see section 3.3.3.3). When this argument is specified, the THPConv tool creates an offset table inside the THP data. If this argument is not specified, then an offset table will not be created inside the THP data.
-non or **-odd** or **-even**	Specifies the format of the THP video data (see THPVideoInfo). If the video data is in interlaced format and starts from an odd-numbered field, specify –odd. If the video data is in non-interlaced format and starts from an even-numbered field, specify –even. If this argument is not specified, then the THP video data format is automatically set as non-interlaced (-non).
-on	Specify this option to convert THP audio data in real time to a playback frequency compatible with Revolution (see section 3.3.2) when the THP movie data is played on the Revolution, rather than when the data is created.
-v	Enables verbose mode.

In the example below, the THPConv tool is used to convert sequential JPEG files into THP movie data

```
THPConv -j test*.jpg -d output.thp
```

When this is executed on the command line, the sequential JPEG files located in the current directory (`test001.jpg`, `test002.jpg`, `test003.jpg` ...) are individually converted into THP video data, and then collected together in one THP movie data file named `output.thp`. See Serial JPEG Files for information about sequential JPEG files.

Note: When converting sequential JPEG files to THP movie data, make sure the extension `.jpg` or `.jpeg` with for the file name of the input file specified.

Note: When playing THP movie data created with the `-on` option specified, please use **`THPPlayerStrmAX`**.

© 2006 Nintendo
CONFIDENTIAL

3.2.2 Replacing or Adding THP Audio Data within THP Movie Data

Code 3-2 `THPConv` Syntax when Replacing or Adding THP Audio Data

```
THPConv.exe -c <inputfile> -s <wavefile> <wavefile>... -d <outputfile> [-<option> <argument>]
```

Table 3-3 `THPConv` Arguments when Replacing or Adding THP Audio Data

Argument	Description
`-c` `<inputfile>`	Specifies the input file (a THP file). This argument must be specified.
`-s` `<wavefile> <wavefile> <wavefile> ... ... ...`	Specifies the sound file (a wav file) that will be substituted. If multiple files are specified in the argument, the data is allocated to the various audio tracks in the order specified. If a THP file format not supporting multi-tracks is input, the format is changed to provide this support.
`-d` `<outputfile>`	Specifies the output file (a THP file). If this argument is not specified, then the THPConv tool will *overwrite the input file*.

© 2006 Nintendo
CONFIDENTIAL

The THPConv tool supports the options shown in Table 3-4:

Table 3-4 `THPConv` Options when Replacing or Adding THP Audio Data

Option	Description
-r <framerate>	Specifies the movie frame rate. The frame rate can be set in the range of 1.0 to 59.94. If no value is specified, then the default value (29.97) is utilized.
-o	With THP movie data, the offset to each frame's data can be stored as a table (see section 3.3.3.3). When this argument is specified, the `THPConv` tool creates an offset table inside the THP data. If this argument is not specified, then an offset table will not be created inside the THP data.
-non or **-odd** or **-even**	Specifies the format of the THP video data (see THPVideoInfo). If the video data is in interlaced format and starts from an odd-numbered field, specify `-odd`. If the video data is in non-interlaced format and starts from an even-numbered field, specify `-even`. If this argument is not specified, then the THP video data format is automatically set as non-interlaced (`-non`).
-trk <track No.> <wavefile>	You can use this option instead of `-s` when you only want to substitute one audio data track. In the first argument you specify the number of the audio track you want to replace (0 or higher). In the second argument the sound file (WAV file) that carries out the substitution is specified. If this option is used, `-r`, `-o`, `-non`, `-odd`, and `-even` are disabled. In addition, the `-on` option is ignored and the playback frequency conversion is automatically set to match the other audio tracks. This option can only be used with THP file formats that supports multiple tracks.
-on	Specify this option to convert THP audio data in real time to a playback frequency compatible with Revolution (see section 3.3.2) when the THP movie data is played on the Revolution, rather than when the data is created.
-v	Turns the verbose mode on.

© 2006 Nintendo
CONFIDENTIAL

Notes:

- When the `THPConv` tool replaces the audio data that is already inside the THP movie data, the old THP audio data is deleted. Once the THP audio data has been deleted it cannot be recovered. Moreover, if an output file is not specified when the audio data is replaced, then the `THPConv` tool will overwrite the existing THP movie data. Thus, when replacing audio data, take the precaution of creating a backup file or specifying an output file, so you do not inadvertently lose data.
- If the playback duration of the wav file is longer than that of the THP movie data, then when the wav file is substituted for the existing audio data, that portion of the wav file that is longer than the THP movie data will not be converted. Conversely, if the playback duration of the THP movie data is longer than that of the wav file, then there will be no sound during that extra time.
- If the THP movie data that has been specified as the input file does not contain audio data, this operation will add THP audio data to the specified file.
- If you are including multiple audio data files in the THP movie data, the number of channels for the audio data and the playback frequencies must be consistent.
- When playing THP movie data created with the `-on` option specified, please use **`THPPlayerStrmAX`**.

3.3 Data Formats

3.3.1 Serial JPEG Files

The `THPConv` tool can convert serial JPEG files (a group of JPEG files, with consecutive numbers at the end of the files that indicate the ordered sequence for playback) into THP movie data. If audio data is interleaved in the serial JPEG files, either a wav file must be specified with the `-s` option during the conversion process, or the WAV file must be added after the sequential JPEG files have been converted.

The following restrictions apply to the serial JPEG files that can be handled by the `THPConv` tool:

- Consecutive numbers corresponding to the frame numbers must be at the end of the serial JPEG files.
- The JPEG file that corresponds to the first frame of the THP movie data can have any number, but the files thereafter must be numbered consecutively from that number.
- The frame number at the end of the filename should have the same number of digits as the final frame, with zeros attached to the front.
- All of the JPEG files that are collected together into a single set of THP movie data must have the same number of vertical pixels and the same number of horizontal pixels.

In the example below, sequential JPEG files are used to create 4 seconds worth of THP movie data at a frame rate of 30 frames per second. The sequential JPEG files (`testxxxx.jpg`) are numbered as shown below:

Code 3-3 Using Serial JPEG Files to Create THP Movie Data

```
Frame     1 : test001.jpg
Frame     2 : test002.jpg
Frame     3 : test003.jpg
   :            :
Frame    51 : test051.jpg
Frame    52 : test052.jpg
   :            :
Frame   118 : test118.jpg
Frame   119 : test119.jpg
Frame   120 : test120.jpg
```

The following restrictions apply to the individual JPEG files that comprise the serial JPEG file group:

- Only the JPEG baseline DCT format is supported
- Only sequential encoding is supported
- Only 4:2:0 sub-sampling is supported
- The pixel count must be a multiple of 16 both vertically and horizontally
- The maximum number of pixels in the horizontal direction is 672. There is no restriction in the vertical direction.

If the serial JPEG files do not fully meet these restrictions, then the `THPConv` tool will output an error and the process will stop.

Note: Serial JPEG files need to have the extension `.jpg` or `.jpeg`.

3.3.2 WAV Files

The THPConv tool can convert a common wav file into THP audio data, and interleave it into the THP movie data. However, there are certain restrictions as to what kinds of WAV files the THPConv tool can convert:

- Must be uncompressed 16-bit PCM data
- Monaural (1 channel) and stereo (2 channels) are supported

If the WAV file does not fully meet these restrictions, then the THPConv tool will output an error and the process will stop.

Although there are no playback frequency restrictions on wav files converted with `THPConv`, please keep the following in mind:

- When the playback frequency is set to 48,000 Hz, Revolution outputs audio data at 48,043 Hz; when the playback frequency is set to 32,000 Hz, it outputs audio data at 32,028 Hz. To maintain strict synchronization between the video data and audio data when playing THP movie data, you must consider these characteristics of Revolution. Specifically, when creating THP movie data, you should either convert the audio data playback frequency in advance so that it matches the Revolution's characteristics, or perform the conversion using the audio library (AX) functionality at the time the THP movie data is played.
  If you want to convert the THP audio data playback frequency in advance, then when creating the THP movie data, run THPConv without the `-on` option. You do not need to prepare a WAV file converted to the Revolution's playback frequencies.
  If you want to convert the THP audio data playback frequency on the Revolution in real time, then when creating the THP movie data, run `THPConv` with the `-on` option. When playing THP movie data that was created with the `-on` option specified, use `THPPlayerStrmAX`.
- WAV files with playback frequencies other than 32,000 Hz and 48,000 Hz can also be converted with `THPConv`. When playing THP movie data created with this type of WAV file, use `THPPlayerStrmAX`.
  Even when you created THP movie data with a WAV file having a playback frequency of 32,000 or 48,000 Hz, if the Revolution-side playback frequency setting differs from the WAV file playback frequency, use `THPPlayerStrmAX`.

Note: If you are including multiple audio data files in the THP movie data, the number of channels for the audio data and the playback frequencies must be consistent.

Note: WAV files need to have the extension `.wav`.

© 2006 Nintendo
CONFIDENTIAL

3.3.3 THP Movie Data

The following sections explain the format of the THP movie data that is output by the `THPConv` tool.

3.3.3.1 THPHeader

`THPHeader` is situated at the beginning of the THP movie data. It holds information for the proper initialization of the THP Player.

`"THP\0" (magic[4])` and the version number are stored at the start of `THPHeader`, in order to identify the THP movie data. The upper two bytes of the version number indicate the major number and the lower 2 bytes the minor number. The `THPConv` tool automatically sets the version number. `THP_VERSION` is defined in the header file (`THPSimple.h`, `THPPlayer.h`) of both Players.

Following this information inside `THPHeader` is information about the maximum frame data size (`bufSize`) and the maximum number of audio samples (`audioMaxSamples`), which together are used by the THP Player to calculate the size of the work region.

After this comes other information, including the THP movie data's frame rate (`frameRate`) and the total number of frames (`numFrames`).

The THP movie data format is designed with extensibility in mind, and can accommodate additional data. The offset to this additional data also can be stored in `THPHeader` (beginning from `finalFrameDataOffsets`).

3.3.3.2 THPFrameCompInfo

Data can be interleaved in frames and stored inside the THP movie data. In the THP library, these interleaved data are referred to as components. THP video data and THP audio data are also components. Designed with extensibility in mind, the THP movie data can store components other than video and audio.

`THPFrameCompInfo` holds `numComponents`, which is the number of components included in the THP movie data, and `frameComp[]`, an array showing the order of the various components stored in each frame. This array stores the THP Components descriptors (see table below) in the order in which the components are interleaved.

Table 3-5 THP Component Descriptors

Descriptor	Order
Video Comp	0
Audio Comp	1
Undefined	2
Undefined	3
Undefined	4
Undefined	5
Undefined	6
Undefined	7
-	8
.	9
.	10
.	11
.	12
.	13
.	14
.	15
Nothing	FF

For example, if the first component of the frame is THP video data and the second component is THP audio data, then `THPFrameCompInfo` would look like this:

Code 3-4 THPFrameCompInfo

```
numComponents       = 2    : Number of components
frameComp[0]        = 0    : Video data descriptor
frameComp[1]        = 1    : Audio data descriptor
frameComp[2..15]    = FF   : No data
```

After `THPFrameCompInfo` comes information on each component (`THPVideoInfo`, `THPAudioInfo`, etc.). This latter component information must also be in the order of the THP Components descriptors stored in the frameComp array.

© 2006 Nintendo
CONFIDENTIAL

(1) THPVideoInfo

`THPVideoInfo` holds the THP video data's vertical & horizontal size information (`xSize`, `ySize`). This information is used for proper playback of the THP video data.

Next, the THP video data format (videoType) is stored. The following values are stored for the video format based on the options when using `THPConv`.

Table 3-6 THP Video Data Format Descriptors

Video Format	THPConv Option	videoType
Non-interlaced	None or [-non]	0
Interlaced starting from odd-numbered fields	[-odd]	1
Interlaced starting from even-numbered fields	[-even]	2

(2) THPAudioInfo

`THPAudioInfo` holds the number of channels (`sndChannels`), the playback frequency (`sndFrequency`), and the total number of audio samples (`sndNumSamples`) of THP audio data. It also holds the number of audio data tracks (`sndNumTracks`) included in the THP audio data. This data is used for proper playback of the THP audio data.

3.3.3.3 THPFrameOffsetData

The THP movie data can also maintain a data table with the offset to the start of every frame. When the `THPConv` tool is used with the `-o` option, `THPFrameOffsetData` is created in the THP movie data. If the `-o` option is not specified, then `THPFrameOffsetData` will not be created. Use this offset data table for special playback purposes, such as to start movie playback from any frame.

3.3.3.4 MovieData

`MovieData` holds the interleaved component data for each frame.

(1) FrameHeader

`FrameHeader` is situated at the front of the `MovieData` data for every frame.

Stored at the start of `FrameHeader` are the size of the previous frame (`frameSizePrevious`) and the size of the next frame (`frameSizeNext`). For the very first frame, the size of the very last frame is stored in `frameSizePrevious`. Similarly, for the very last frame, the size of the very first frame is stored in `frameSizeNext`.

Following these two values, `FrameHeader` stores the data size information in each component interleaved in that frame. This size information must be stored in the order in which the components are interleaved, as defined in `THPFrameCompInfo`.

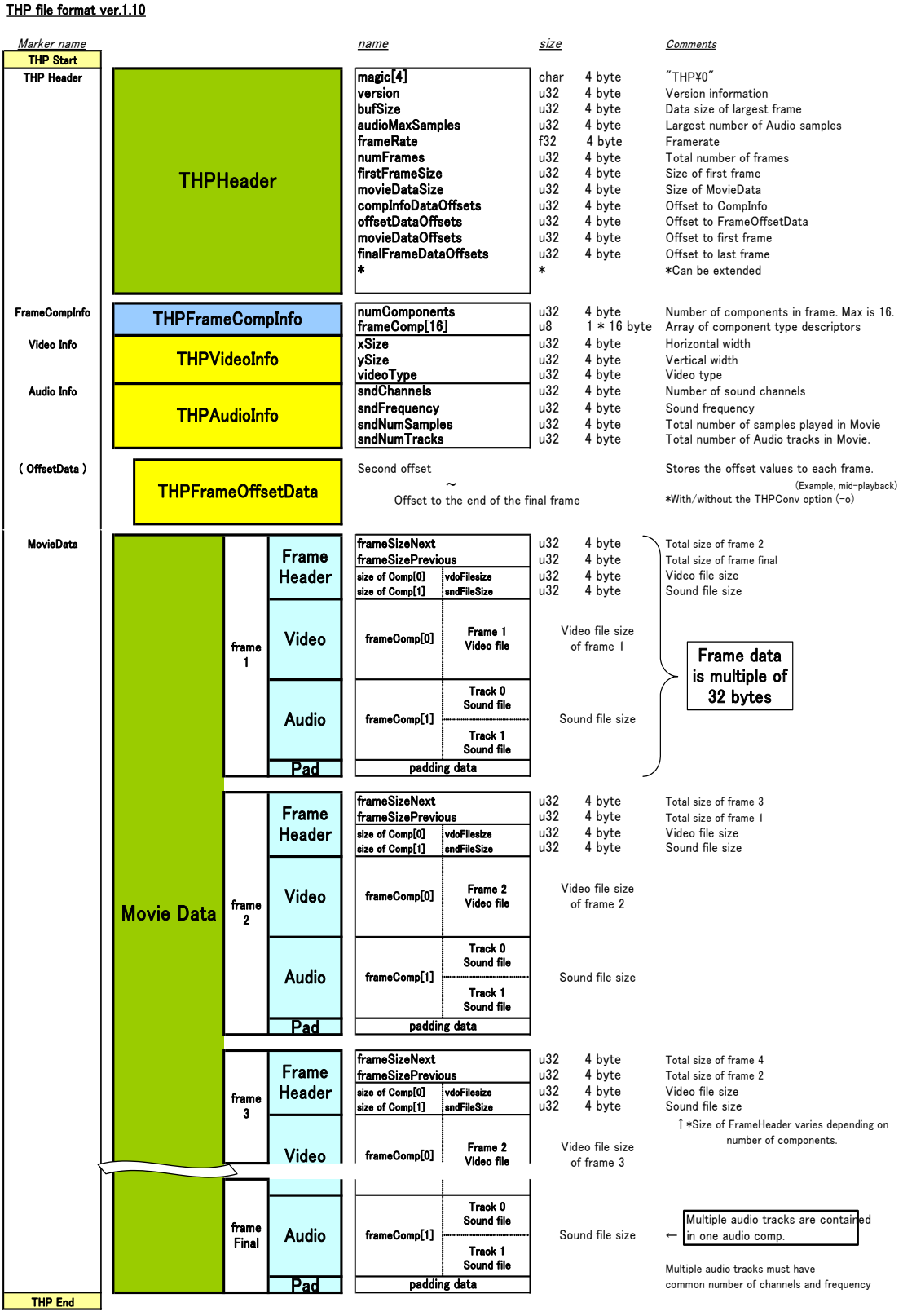
The size of each frame must be a multiple of 32 bytes. Frames should be padded with 0, as needed, at the end of the frame data to meet this requirement.

Notes:

- The component data sizes included in `FrameHeader` store the size information for each component. However, in the case of audio components, the data size of one audio track is stored.
- The THP audio data is compressed in the Revolution audio system's DSP ADPCM format. Because of this, the number of audio data samples stored in every frame (with the exception of the last frame) must be a multiple of 14.

© 2006 Nintendo
CONFIDENTIAL

Figure 3-1 The THP File Format



© 2006 Nintendo
CONFIDENTIAL

3.4 When Creating THP Movie Data

The following items should be noted when using the `THPConv` tool.

3.4.1 Specifying the Path

The `THPConv` tool makes use of `dsptool(D).dll`, which comes with the Revolution SDK. In order to use `THPConv` tool, you must use the path: \(Root)\x86\bin.

3.4.2 Sufficient Available Hard-disk Space

The THP movie data output by the `THPConv` tool is nearly the same size as the original data before the conversion. When the `THPConv` tool converts video data and audio data into THP movie data, it creates temporary files in the current directory. These temporary files are also around the same size as the original data.

Be sure to confirm that there is sufficient available space on the hard disk when using the `THPConv` tool.

Equation 3-2 Required Available Hard-disk Space for `THPConv`

Necessary free hard-disk space = Size of original data x 3

When the `THPConv` tool creates temporary files, it names the temporary file for video data `__tmp_VD` and the temporary file for audio data `__tmp_AD`. If files with these same filenames exist in the current directory, the `THPConv` tool will overwrite them.

3.4.3 The Execution Environment for the `THPConv` Tool

If the `THPConv` tool is executed on bash, sometimes the tool will not get the input file that was specified by the argument and the THP file will not be created as anticipated. For this reason, be sure to execute the `THPConv` tool from the command line.

4 Cautions

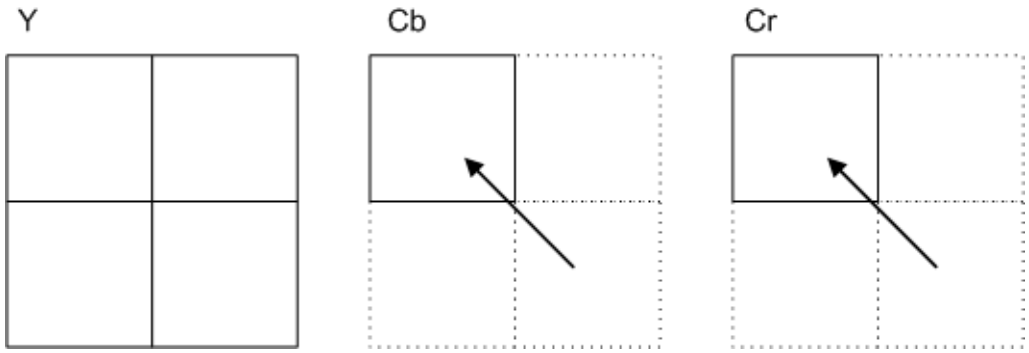
4.1 General Cautions

4.1.1 Sub-sampling

The THP library only supports 4:2:0 sub-sampling.

For a JPEG file with 4:2:0 sub-sampling, the Cb and Cr components are culled as shown in the figure below in regards to the Y component:

Figure 4-1 4:2:0 Sub-Sampling



If the input JPEG data does not have 4:2:0 sub-sampling, then the `THPConv` tool will output an error and the conversion process will stop. If it is not clear what kind of sub-sampling is performed on the JPEG data output by a commercial graphics application, you can determine whether the THP library will be able to decode the data by seeing what happens when the file is input to the `THPConv` tool.

4.1.2 The Players and the GX settings

The THP Simple Player and the THP Player both utilize the THP library's low-level API function, `THPVideoDecode`, to decode the THP video data. This function outputs the decoded data in YUV texture format. Both Players draw to the screen by pasting this YUV texture format decoded data to polygons.

Both Players effect major changes to the GX settings when drawing the decoded data. For applications that make use of THP movie data, and especially for applications that play movies and draw objects at the same time, be sure to rectify the GX settings that are used for the drawing of objects, fixing them in each frame after the movie has been drawn and before the object is drawn.

4.1.3 Output of THP Audio Data for Players

When the THP Simple Player and THP Player (`THPPlayer`) are used at the same time as AX, they mix audio output data from the audio libraries with THP audio data using the CPU, and send it to the audio interface.

In contrast, the THP player that uses AX audio streaming (THPPlayerStrmAX) sends the THP audio data to the audio library and any subsquent processing (until audio data is output from the Revolution) is handled by the audio library.

© 2006 Nintendo
CONFIDENTIAL

4.1.4 Sampling Rate of THP Audio data with Simultaneous Use of Audio Libraries

The sampling rate for THP audio data needs to be 32khz when the THP Simple Player and the THP Player are used with AX simultaneously.

4.1.5 Monitoring the Optical Disc Drive during Streaming Playback

When streaming THP movie data from the optical disc for playback, please monitor the state of the disc drive from inside the main loop that calls the THP Player's various API functions and draws the frame buffer, and perform the proper process for the given disc drive state. (The code shown in Code 4-1 is an example of such processes; the exact same procedure need not be followed.)

Code 4-1 Example: Monitoring the Drive within the Main Loop

```
switch(DVDGetDriveStatus())
{
    case DVD_STATE_FATAL_ERROR:
        Stop playback
        break;
    case DVD_STATE_NO_DISK:
        Stop playback
        break;
    case DVD_STATE_COVER_OPEN:
        Pause playback
        break;
    case DVD_STATE_WRONG_DISK:
        Stop playback
        break;
    case DVD_STATE_RETRY:
        Stop playback
        break;
 }
```

4.1.6 Handling Sample Data

The sample data that accompanies this library (`rebirth.thp`) is provided strictly for reference purposes. The misappropriation of this sample data and its duplication or alteration without the permission of Nintendo of America is prohibited.

Sample data[a]	`rebirth.thp`	Copyright © 2000 Nintendo
	Created by:	mix core: http://www.mix-core.com
	CG Designers:	Shinichi Shirai, Yutaka Nishikawa, Makoto Nishibori, Takahiro Onishi, Hiroyuki Kusukawa
	Sound Composer:	Masaya Tsunemoto
	Violinist:	Yoko Yoshida

a. All queries regarding this sample data should be directed to Nintendo of America, Inc. (support@noa.com).

© 2006 Nintendo
CONFIDENTIAL

4.2 Regarding the THP Player

4.2.1 The Main Loop

When playing THP movie data, be sure to program your application so that the main loop that draws to the screen and calls the THP Player loops once per `vsync`.

Code 4-2 Restriction on Main Loop when Using the THP Player

```
THPPlayerPlay()

while(1)
{
            .
            .

    THPPlayerDrawCurrentFrame();
            .
            .

    VISetNextFrameBuffer();
    VIFlush();
    VIWaitForRetrace();
}
```

The THP movie data may not play smoothly if the main loop is set to loop at anything other than once per `vsync`.

4.2.2 The `THPPlayerDrawCurrentFrame` function

Sometimes the call to the `THPPlayerDrawCurrentFrame` function may fail (returned value = -1) even after the call to the `THPPlayerPlay` function has succeeded. The reason is because, internally, playback has been delayed past the proper timing for the start of play.

Essentially, the first frame of decoded data is fetched by the THP Player's VI post callback function at the time of the vsync right after the `THPPlayerPlay` function is called (though in the case of an interlaced movie, this action could be delayed due to restrictions on the first field). It is after this process that drawing of video data by the `THPPlayerDrawCurrentFrame` function becomes possible.

If you are drawing movies and objects at the same time, please check the value returned by the `THPPlayerDrawCurrentFrame` function and confirm that it is some frame number other than -1 (failure) before displaying the object.

© 2006 Nintendo
CONFIDENTIAL

If it is not an interlaced movie, then you can use the procedure in Code 4-3 to make certain that the `THPPlayerDrawCurrentFrame` function is called successfully:

Code 4-3 Making Certain the Call to the `THPPlayerDrawCurrentFrame` Function Succeeds

```
THPPlayerPlay();

VIWaitForRetrace(); <------- At the time of this vsync, the first frame is
                        fetched by the VI post callback.
while(1)
{
        .
        .

    THPPlayerDrawCurrentFrame();
        .
        .

    VISetNextFrameBuffer();
    VIFlush();
    VIWaitForRetrace();
}
```

4.2.3 VI Post Callback

The THP Player utilizes VI post callbacks in order to control movie playback. Callbacks that are registered before the registration of the THP Player's VI post callback function are called via the Player's VI post callback. At the end of playback (i.e., when `THPPlayerStop` is called), the Player returns the VI post callback to its origin.

4.2.4 Interlaced Movies

The THP Player supports the playback of interlaced movies, but only if the format is such that the data in every frame contains even fields and odd fields that are alternately interleaved in each scan. The even and odd fields should be set up so that each field comes 1/59.94 seconds after the previous field for NTSC and MPAL, and 1/25 seconds for PAL

When the data exists in this format, the timing for the start of playback is affected by whether the data in each frame begins with the even field or the odd field.

The THP Player uses the second argument of the `THPPlayerPrepare` function to get information on the scanning order of the interlaced movie, and to automatically adjust the timing of the start of playback.

There are no restrictions on the screen size for interlaced movies that can be played by the THP Player. However, whether each frame begins with an even field or an odd field places restrictions on the drawing location when the movie is drawn to the screen. The developers must adjust the drawing location themselves.

When an interlaced movie is scaled, the position of the even and odd lines inside the frame will be misaligned, and it may not be possible to play the movie back correctly. For this reason, do not scale interlaced movies for playback.

© 2006 Nintendo
CONFIDENTIAL

Below is a list of the restrictions that apply to interlaced movies that can be played with the THP Player:

- The even fields and the odd fields must be interleaved.
- The frame rate must be 29.97 frames per second for NTSC and MPAL, and 25 frames for PAL.
- The developer must specify in the THP Player whether the frame data that comprises the interlaced movie is scanned in the order of even field → odd field, or odd field → even field.
- There are no restrictions on screen size, but the drawing location must be adjusted.
- The decoded data cannot be scaled.

4.2.5 Displaying THP Movie Data Created for NTSC on a PAL System

When creating software versions for foreign markets, you may want to repurpose THP movie data created in the NTSC format for release in a PAL format.

When THP movie data created for NTSC televisions at a rate of 29.97 FPS is displayed on an NTSC television, the movie will play correctly if the screen is refreshed every 2 Vsync. If the same data is displayed on a PAL television and the screen is refreshed every 2 Vsync, because PAL televisions display 25 frames (50 fields) per second, not all of the frames will be displayed, and the movie will not appear smooth (when using THP player).

The easiest way to avoid this lack of smoothness in display is to change the screen refresh rate to every 1 Vsync. This allows all the frames in the THP movie data to be displayed on the television and reduces the loss of fluidity.

The critical thing here is when to display each frame. THP player monitors the time since the playback started and it determines the frame to draw with the THPPlayerDrawCurrentFrame function. Accordingly, the user does not need to worry about the timing and can simply call the THPPlayerDrawCurrentFrame function each Vsync.

In contrast, the THP Simple Player does not internally monitor the time since the start of playback. The user must monitor the time by himself or herself and must decode and display the frame at the right time.

Note: The above method can only be applied to NTSC movie data that is not interlaced and has a frame rate below 50 FPS.

Appendix A. Function List

A.1 THP Low-level API Functions

API	Process
`THPInit`	Carries out preparation for decoding with THP library.
`THPVideoDecode`	Decodes THP video data.
`THPAudioDecode`	Decodes THP audio data.

A.2 THP Simple Player API Functions

API	Process
`THPSimpleInit`	Initializes the THP Simple Player.
`THPSimpleQuit`	Quits the THP Simple Player.
`THPSimpleOpen`	Opens THP movie data.
`THPSimpleClose`	Closes THP movie data.
`THPSimpleCalcNeedMemory`	Obtains the size of the work area that the THP Simple Player uses.
`THPSimpleSetBuffer`	Sets the work area in the THP Simple Player.
`THPSimplePreLoad`	Starts the pre-load of the THP movie data.
`THPSimpleLoadStop`	Stops the pre-load of the THP movie data.
`THPSimpleDecode`	Decodes THP video and audio data.
`THPSimpleDrawCurrentFrame`	Draws the decoded THP video data immediately after calling the THPSimpleDecode function.
`THPSimpleAudioStart`	Starts playback of THP audio data.
`THPSimpleAudioStop`	Stops playback of THP audio data.
`THPSimpleSetVolume`	Sets the playback volume for the THP audio data.
`THPSimpleGetVolume`	Obtains the current volume setting of the THP audio data.
`THPSimpleGetVideoInfo`	Obtains information for the THP video data.
`THPSimpleGetAudioInfo`	Obtains information for the THP audio data.
`THPSimpleGetFrameRate`	Obtains the frame rate for the THP movie data.
`THPSimpleGetTotalFrame`	Obtains the total number of frames for the THP movie data.

© 2006 Nintendo
CONFIDENTIAL

A.3 THP Player API Functions

API	Process
`THPPlayerInit`	Initializes the THP Player.
`THPPlayerQuit`	Quits the THP Player.
`THPPlayerOpen`	Opens THP movie data.
`THPPlayerClose`	Closes THP movie data.
`THPPlayerCalcNeedMemory`	Obtains the size of the work area that the THP Player uses.
`THPPlayerSetBuffer`	Sets the work area in the THP Player.
`THPPlayerPrepare`	Prepares playback of the THP movie data.
`THPPlayerPlay`	Starts playback of the THP movie data.
`THPPlayerStop`	Stops playback of the THP movie data.
`THPPlayerPause`	Pauses playback of the THP movie data.
`THPPlayerSkip`	Skips one frame of THP movie data.
`THPPlayerDrawCurrentFrame`	Draws THP video data that needs to be displayed, synchronizing with playback of THP audio data.
`THPPlayerDrawDone`	Use this function instead of GXDrawDone when playing back THP movie data.
`THPPlayerSetVolume`	Sets the playback volume for the THP audio data.
`THPPlayerGetVolume`	Obtains the current volume setting of the THP audio data.
`THPPlayerGetVideoInfo`	Obtains information for the THP video data.
`THPPlayerGetAudioInfo`	Obtains information for the THP audio data.
`THPPlayerGetFrameRate`	Obtains the frame rate for the THP movie data.
`THPPlayerGetTotalFrame`	Obtains the total number of frames for the THP movie data.
`THPPlayerGetState`	Obtains the current status of the THP Player.

© 2006 Nintendo
CONFIDENTIAL

© 2006 Nintendo
CONFIDENTIAL