Icon and Banner Authoring Manual

2012/06/27

Version 1.3

Table of Contents

Code

Tables

Figures

1 Introduction

This document provides application developers with information related to icons and banners, which are required for CTR applications.

1.1 How to Read This Document

First, read Chapter 2 Icon/Banner Overview for an overview of icons and banners. Determine the types of icons and banners that are necessary based on your application's specifications.

Next, start reading at Chapter 3 How to Create CTR Icons to learn how to create each of the various icons and banners and to understand the related precautions. This document does not give detailed explanations of the tools used to create icons and banners. See the relevant manual for each tool.

Read Chapter 10 Notes and Precautions for supplementary information related to bugs and other topics. Refer to this as necessary.

2 An Overview of Icons and Banners

This chapter gives an overview of icons and banners by explaining which types are used by CTR applications as well as where and when they are used.

This chapter also describes the types of tools related to creating icons and banners in order to explain the procedures given in the following chapters.

2.1 Types of Icons and Banners

The icons and banners used by CTR applications can be grouped into the following seven categories.

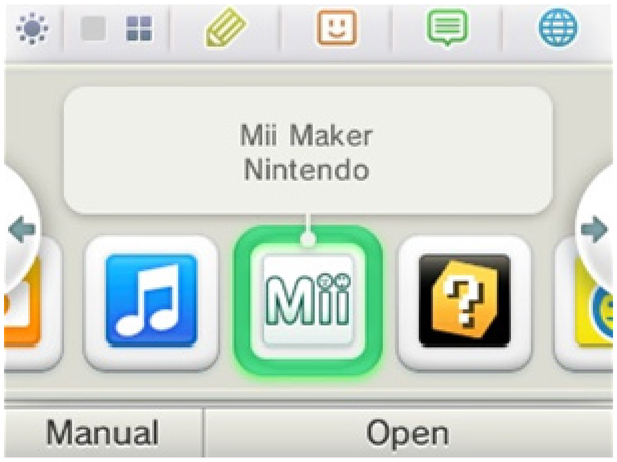
- CTR icons
- CTR title banners
- CTR contextual banner data
- Expanded save data icons
- Icons for Download Play child programs
- Icons for StreetPass boxes
- Icons for StreetPass messages

The following sections explain where and how these are used.

2.1.1 CTR Icons

The HOME Menu displays these icons on the lower screen. A CTR icon image includes both 48x48 and 24x24 image data, which are switched out smoothly according to the situation. They also include titles and other text information.

Figure 2-1 CTR Icons



CTR icons are used not only by the HOME Menu but also by the friend list and Activity Log as well as in various other situations.

Table 2-1 Locations That Use CTR Icons

Application	Location
HOME Menu	• The lower screen of the HOME Menu • The upper screen when an application is suspended
Friend List	• When the Favorite Title selection screen is displayed • When a game is displayed under Currently Playing
Notification Applet	• SpotPass notifications and notifications from general applications
Activity Log	• Activity Log • Software library
System Settings	• Software Management screen
StreetPass Mii Plaza	• Most Recent Software
Game Notes	• The image that appears for the suspended software
E-manual	• The top page of the e-manual
Nintendo 3DS System Transfer	• The list of software titles

Note: As with CTR icons, different-sized images are also used in printed materials for card-based software packages. A resolution of 384x384 (72 pixels/inch) is required for printed material. Use the same design as you do within your application. For details, see the Printed Matter Guidelines for Software.

Icons were animated for Nintendo DSi software but not for CTR software. These cannot be switched by expanded save data like CTR title banners.

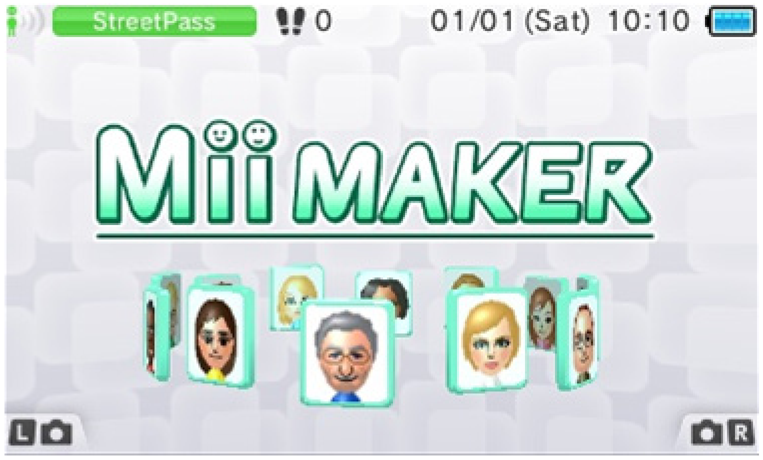
CTR icons must be created for every title.

2.1.2 CTR Title Banners

These title banners are shown on the upper screen of the HOME Menu. 3D imagery is used to make the 3D models appear to be displayed in a showcase. The 3D model appears in the middle of a rotating oblate spheroid. The model rotates slowly. Rotation speed responds to microphone input audio.

When you select a CTR icon from the lower screen on the HOME Menu, the title banner is displayed on the upper screen and a banner sound is played back.

Figure 2-2 A CTR Title Banner



CTR contextual banner data (described later) allows you to replace some elements in the display.

CTR title banners must be created for all titles.

2.1.3 CTR Contextual Banner Data

You can replace some of the CTR title banner display elements with the content of CTR contextual banner data. You can replace textures and the scrolling text displayed at the bottom of the upper screen.

You cannot switch CTR icons and banner sounds.

There are two types of CTR contextual banner data: local contextual banners, which are created from applications, and downloadable contextual banners, which are downloaded via SpotPass.

2.1.3.1 Local Contextual Banners

Applications can save these banners within contextual banner data to apply them. These banners allow you to apply images that reflect in-game progress as well as content specific to individual players.

2.1.3.2 Downloadable Contextual Banners

This contextual banner data is applied after it is distributed via SpotPass. Unlike local contextual banners, you can configure these to expire.

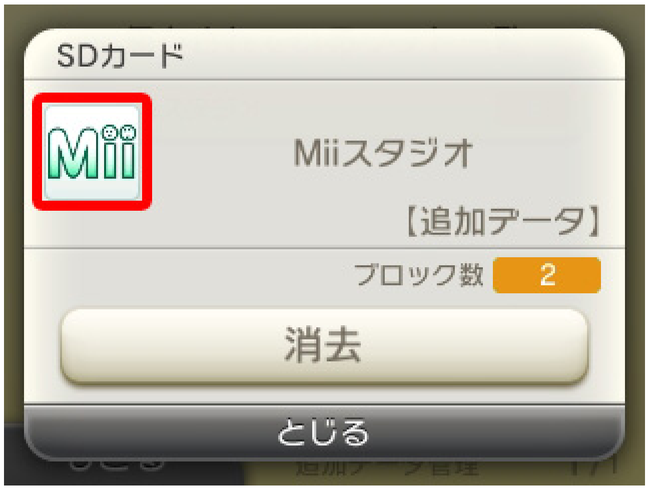
Local contextual banners and downloadable contextual banners can generally be applied together if they both exist at the same time, but downloadable contextual banners have higher priority when both banners are configured to switch the same content. For details, see Chapter 5 How to Create CTR Contextual Banner Data.

You can use CTR contextual banner data however you like, including using it to change CTR title banners.

2.1.4 Expanded Save Data Icons

These icons are displayed under Extra Data Management in System Settings.

Figure 2-3 Expanded Save Data Icons

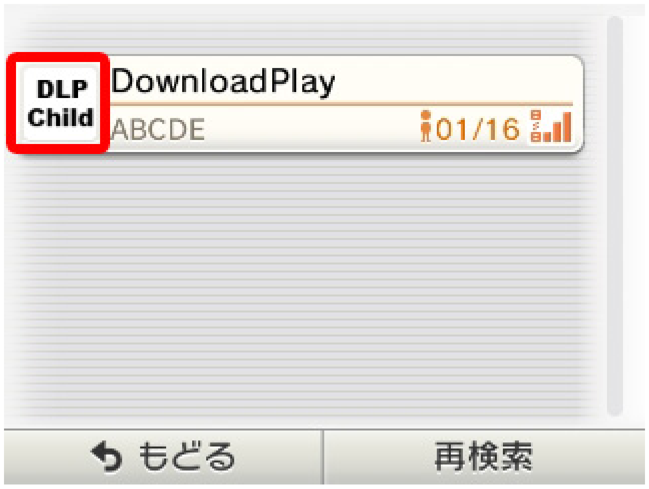


You must create these for titles that use expanded save data.

2.1.5 Icons for Download Play Child Programs

These icons are displayed on systems that download child programs via Download Play.

Figure 2-4 Icons for Download Play Child Programs

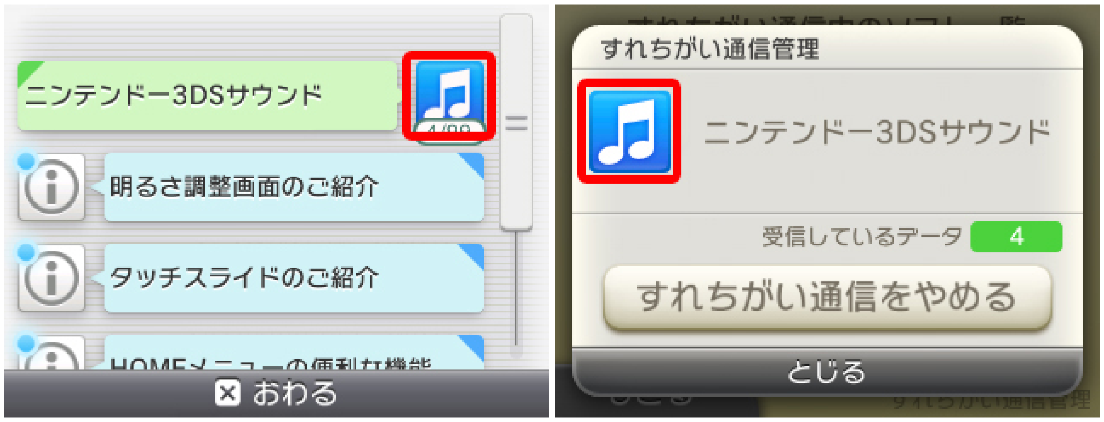


You must create these for applications that use Download Play. Embed them for use in the Download Play child programs.

2.1.6 Icons for StreetPass Boxes

These icons appear in the StreetPass Management screen and Notifications List. Set them when you create StreetPass boxes for performing StreetPass communication.

Figure 2-5 Icons for StreetPass Boxes

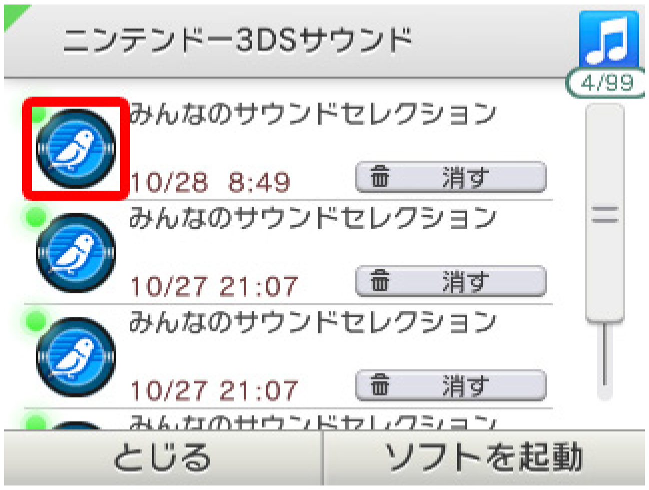


You must create these for titles that use StreetPass.

2.1.7 Icons for StreetPass Messages

These icons are attached to messages exchanged via StreetPass. They appear under the Notifications List in the HOME Menu.

Figure 2-6 Icons for StreetPass Messages



You must create these for titles that use StreetPass.

2.2 Tools Required to Create Icons

The CTR-SDK and a special version of NintendoWare for CTR ("NW4C for Banner") provide the tools used to create icon and banner data. For details on each of these tools, see the documentation in their respective packages.

2.2.1 CTR-SDK

The CTR-SDK provides converter tools for various types of data.

2.2.1.1 ctr_TexturePackager

The `ctr_TexturePackager` tool loads TGA image files, converts them into a (compressed or uncompressed) texture format that can be used by CTR systems, and then creates a package file that combines multiple textures. This generates a texture package file with the `.ctpk` extension. This file is used when an icon is created.

Note: From CTR-SDK 3.2 onward, you don't need to use this tool because the `ctr_makebanner` tool can use NW4C TGA files and CTPK files directly.

You can also remove the header and padding region(s) from an NW4C TGA file to output it as binary data. You can use this binary data in icons for StreetPass boxes and StreetPass messages.

2.2.1.2 ctr_BannerModelConverter

The `ctr_BannerModelConverter` tool creates model data for CTR title banners. Convert the intermediate files output by NW4C for Banner's CreativeStudio to generate BCMDL data. Use the generated BCMDL data with `ctr_makebanner`.

2.2.1.3 ctr_WaveConverter

The `ctr_WaveConverter` tool converts ordinary waveform file formats (`.wav`, `.aif`, `.aiff`) into a CTR waveform file format (`.bcwav`). Use these to create banner sounds for CTR title banners.

2.2.1.4 ctr_makebanner

The `ctr_makebanner` tool creates icon and banner data. In addition to the required texture, model and sound data, prepare the title information that will be displayed with the icon and a BSF file with various settings denoted for when you generate icon and banner data. Use `ctr_makerom` to embed this generated icon and banner data into the executable binary.

2.2.1.5 ctr_makerom

The `ctr_makerom` tool creates executable binaries for CTR applications. Pass CTR icon and banner data as arguments to embed CTR icons and banners into executable binaries.

2.2.1.6 ctr_make_ex_banner

The `ctr_make_ex_banner` tool creates contextual banner data. Both common and language-specific BIN data are generated.

2.2.1.7 ctr_makedlexbanner

The `ctr_makedlexbanner` tool generates downloadable contextual banners. BIN data generated by `ctr_make_ex_banner` is converted into data that will be downloaded.

2.2.2 NintendoWare for CTR for Banner (Package for Creating Banner Data)

You need to use a specific version of NW4C to create CTR title banners. To distinguish this from the normal NW4C package, it is provided under the name "NW4C for Banner" (the banner creation package). These tools are based on NW4C 1.3.2.

2.2.2.1 CreativeStudio for Banner

CreativeStudio is a graphics IDE for the CTR system. With it, you can edit 3D model data output by Maya and other DCC tools through special plug-ins.

2.2.2.2 Adobe Photoshop Plug-Ins

These plug-ins create NW4C texture files from Adobe Photoshop. These output TGA files with additional, NW4C-specfic information (called "NW4C TGA files").

2.2.2.3 Autodesk Maya, 3ds Max, and Softimage Plug-Ins

These plug-ins output 3D intermediate files from general-purpose 3D graphics tools: Autodesk Maya, 3ds Max, and Softimage. Edit these intermediate files in CreativeStudio.

2.2.2.4 Viewer (Hardware Viewer)

With Viewer, you can use a CTR system to display data created by the tools and plug-ins provided by NW4C. After you have edited data with the various tools, you can transfer it to Viewer through HIO-Daemon and then preview it.

Note: The HOME Menu is implemented using NW4C 1.2.23 and might therefore exhibit slight disparities with the results that have been confirmed using the NW4C for Banner package based on NW4C 1.3.2.

Although no such differences have been found in the displayed results, always use the HOME Menu for your final checks. To learn more about how to check data in the HOME Menu, see the manuals included with the CTR-SDK.

3 How to Create CTR Icons

Create a CTR icon as follows.

1. Prepare both small and large image data to use in the CTR icon.
2. Use `ctr_TexturePackager` to convert the CTR icon image data into the CTPK format.
3. Pass the image data to `ctr_makebanner` to create ICN data.
4. Pass the ICN data to `ctr_makerom` to embed it in a ROM.

The following sections explain each of these steps.

3.1 Prepare the Icon Images

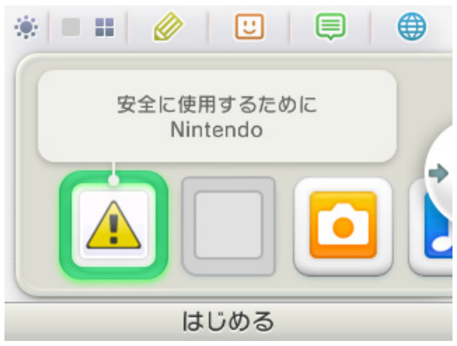
To create a CTR icon for a CTR application, you need to provide both a large (48x48) CTR icon and a small (24x24) CTR icon in the TGA format.

Note: You can use NW4C TGA files along with CTR-SDK version 3.2 or later to reduce the amount of work you must do. Nintendo therefore recommends that you use NW4C for Banner's Photoshop plug-in to save files in the NW4C TGA format.

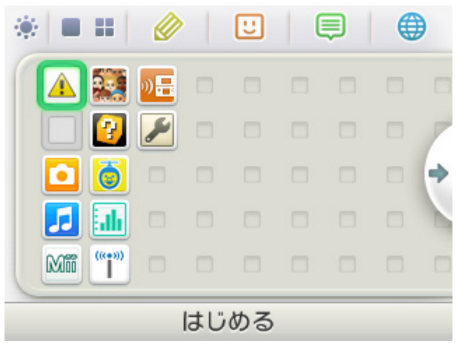
Each of the two image types is used to represent the application with an appropriate size. These are generally assumed to be the same image, so treat the small CTR icon as simply a miniaturized version of the large CTR icon.

For example, CTR icons shown on the HOME Menu will grow and shrink as the HOME Menu layout is changed on the lower screen due to user operation. Something would seem amiss if the large and small CTR icons were to use entirely different images.

Figure 3-1 Sample CTR Icon Display



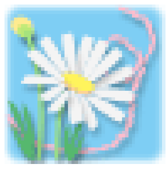
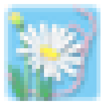
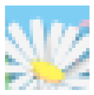
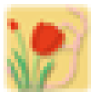
(a) 1 行表示バージョン
(48px * 48px)



(b) 5 行表示バージョン
(24px * 24px)

When you shrink a large CTR icon to create a small one, you may touch up a few pixels in the small icon to get rid of moiré patterns and other artifacts. However, you cannot use a small icon that is just a part of the large icon, or one that has different colors or shapes.

Table 3-1 Sample Created CTR Icons

Icon	Size	Allowed?	Description
	Large	Yes	The original (large) icon.
	Small	Yes	An icon that has been shrunk and touched up by hand.
	Small	No	An icon that is only part of the original image.
	Small	No	An icon with different colors and shapes than the original icon.

Note: As with CTR icons images, different-sized icons are also used in printed material for card-based software packages. A resolution of 384x384 (72 pixels/inch) is required for printed material. Use the same design as you do within your application. For details, see the Software Printed Matter Guidelines.

Note: Icons for expanded save data and downloadable child programs use the same data format as CTR icons. The small images used for these icons are assumed to be scaled-down—but otherwise identical—versions of the large images and must comply with the restrictions above.

Note: Parts of a CTR icon are sometimes not displayed correctly. For details, see section 10.1 Parts of an Icon Are Sometimes Displayed Incorrectly.

3.2 Convert and Compress the Images

Use `ctr_TexturePackager` to convert both the large and small (TGA) CTR icon image data created in the previous section into the CTPK format. For example, you could run the command as follows.

Code 3-1 Sample Invocations of `ctr_TexturePackager`

```
% ctr_TexturePackager.exe -l icon_large.tga -dsl
% ctr_TexturePackager.exe -l icon_small.tga -dsl
```

Note: You don't need to do this if you are using at least version 3.2 of the CTR-SDK and have saved your image data in the NW4C TGA format.

3.3 Create the Icon Data

Pass the image data created in the last section to `ctr_makebanner` to create an ICN file. When you do so, fill out a configurations file called a banner spec file (BSF).

3.3.1 Configure the Banner Spec File

Within the BSF file is the basic application information, such as its title and information on the publisher. This information appears in the HOME Menu, friend cards, the Activity Log, and elsewhere. The BSF file has other information as well, some of which is related to an application's specifications. This affects Parental Controls and other behavior.

This section only describes image data settings related to the procedure up to the previous section. For information on other settings, see `$CTR_SDK/documents/tools/ctr_makebanner.html` in the CTR-SDK. Use settings that are relevant to your application's specifications.

Set the Image Data

Specify the image data that you created in the last section to the BSF file.

Code 3-2 Sample BSF File Content

```
BigIconFile:    icon_large.ctpk    # Filename for the 48x48 icon
LittleIconFile: icon_small.ctpk    # Filename for the 24x24 icon
```

Note: If you are using version 3.2 or later of the CTR-SDK, you can specify an NW4C TGA file directly. If you are using an earlier version of `ctr_makebanner`, specify the CTPK file created in the previous section.

3.3.2 Use ctr_makebanner

Once you have prepared a BSF file, use `ctr_makebanner` to create an ICN file.

Code 3-3 Sample Invocation of the `ctr_makebanner` Command

```
ctr_makebanner32 -nobnr BannerSpec.bsf icon_file.icn
```

Note: For the sake of simplicity, the command here only creates CTR icon data. You would ordinarily create both CTR icon and banner data at the same time.

3.4 Embed Into the ROM

Use `ctr_makerom` to embed the ICN data that you created in the last section into a ROM.

Code 3-4 Sample Invocation of the `ctr_makerom` Command

```
% ctr_makerom -icon icon_file.icn
```

You are now done creating and embedding a CTR icon.

3.5 Icon Database and Cache

The CTR system has an icon database that saves information on CTR icons. Once you register a CTR icon's information to the icon database, the Activity Log and other built-in system applications can use it. CTR icon information is registered to the icon database at the following times.

- Icon data is obtained from an application and then saved when the application is launched on the system.
- Icon data is saved when it is received via StreetPass.
- Icon data is downloaded from an icon database server on the network and saved on a system when that system has never launched a Favorite Title specified by a friend in the friend list. If a friend has set an application from another region, that region's icon data is saved. An icon is not saved in the icon database for the game that a friend is currently playing.

Icon data is registered to the icon database on the network when an application passes Lotcheck. If the same title is released in more than one region, Lotcheck accepts each region's title as a separate application. As a result, the same icon does not appear in all regions until after the title has passed Lotcheck for all of those regions.

If you change the system region on development hardware or elsewhere with version 3 or earlier of the SDK, the icon data that had been registered to the icon database for the previous region will be preserved unless you initialize the system. (Version 4.0 and later of the SDK force a system initialization when the region is changed and thus clear the icon database.)

Once an icon has been registered to the icon database, its application's icon information can be referenced from the icon database. In other words, if in the process of creating an icon you modify it and attempt to view the result, you will see the icon that was registered to the database before. To see your modifications to an icon during debugging or at some other time, you need to clear the icon database. You can clear the icon database by holding down the X and Y Buttons when you launch the HOME Menu. If the database has been cleared successfully, a dialog box appears after the HOME Menu was launched to tell you that save data is being initialized.

The icon cache is a collection of icon data that is temporarily saved on an SD Card to allow the HOME Menu to load icons more quickly. Once icon data is saved in this cache, it will be used until the cache is explicitly cleared. You can clear the icon cache just like the icon database, by holding down the X and Y Buttons when you launch the HOME Menu.

4 How to Create CTR Title Banners

Create a CTR title banner as follows.

1. Create banner model data.
2. Create banner sound data.
3. Pass the model and sound data to `ctr_makebanner` to generate BNR data.
4. Pass the BNR data to `ctr_makerom` to embed it in a ROM.
5. Use the HOME Menu's debugging features to check the processor load.

The following sections explain each of these steps.

4.1 Create Banner Model Data

After model data has been created by CreativeStudio and converted into BCMDL files by `ctr_BannerModelConverter`, the resulting binary data is used for CTR title banners.

Note: You must use model data that has been created with the NW4C for Banner package.

4.1.1 Rotate the Model

3D models rotate clockwise (once every ten seconds) around the screen's vertical axis. If you don't want to rotate a node, either use billboarding or rotate it in the opposite direction (counter-clockwise at the same speed).

The rotation speed varies with the input microphone volume (under the assumption that players will blow into the microphone). When this happens, the rotation will not affect nodes that are being used as billboards but it will affect nodes that are rotating in the opposite direction.

4.1.2 Adjust the View Range and Parallax

When you create a CTR title banner, configure the camera's parameters with the values given in Table 4-1.

Table 4-1 Camera Parameters for Banners

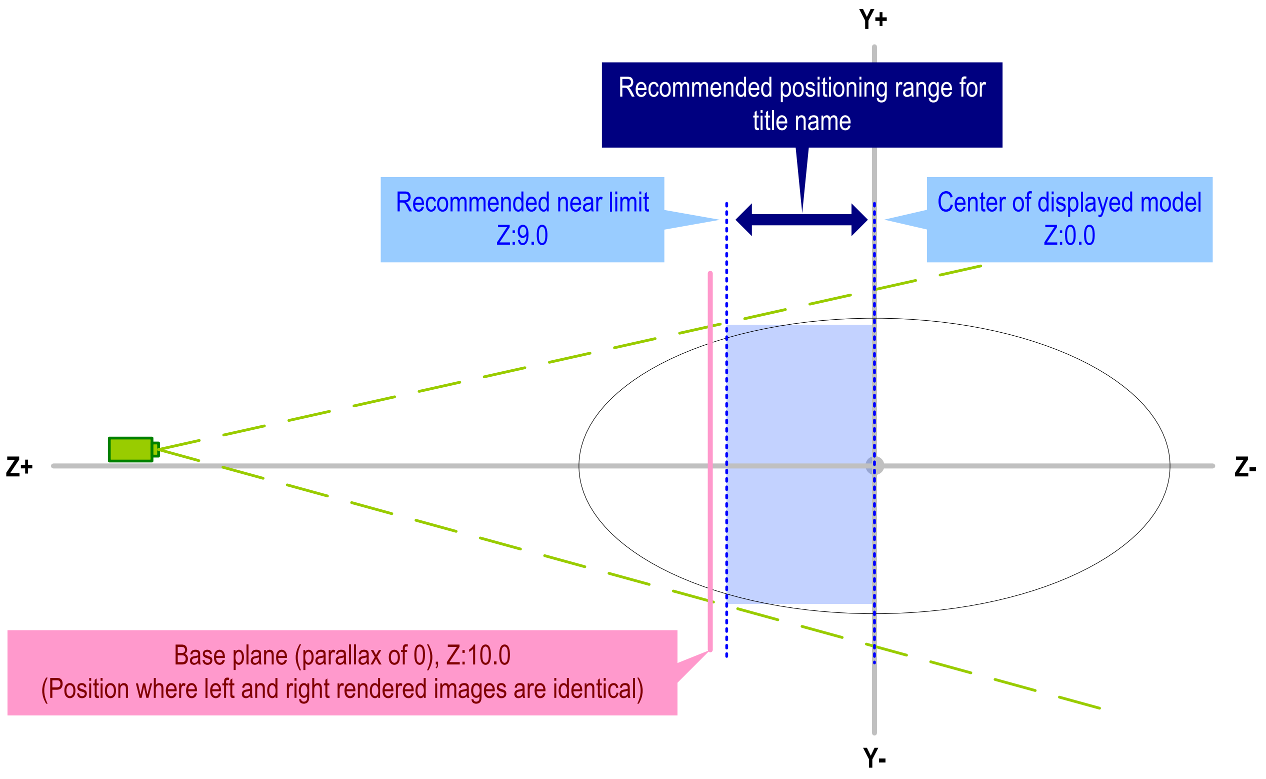
Parameter	Value
Position	X : 0, Y : 1, Z : 44.786
Focal point	X : 0, Y : 1, Z : 0
Mode	Persp
Fovy	30
Aspect	1.666667
Near	26.5

Parameter	Value
Far	10000
ULCD range	34.786 (+10 from the origin is the base plane)
ULCD factor	1.0

3D models placed outside of the space of the spheroid will not be visible. Do not allow your model to hit the edge of the screen when the 3D depth slider is moved to its maximum value.

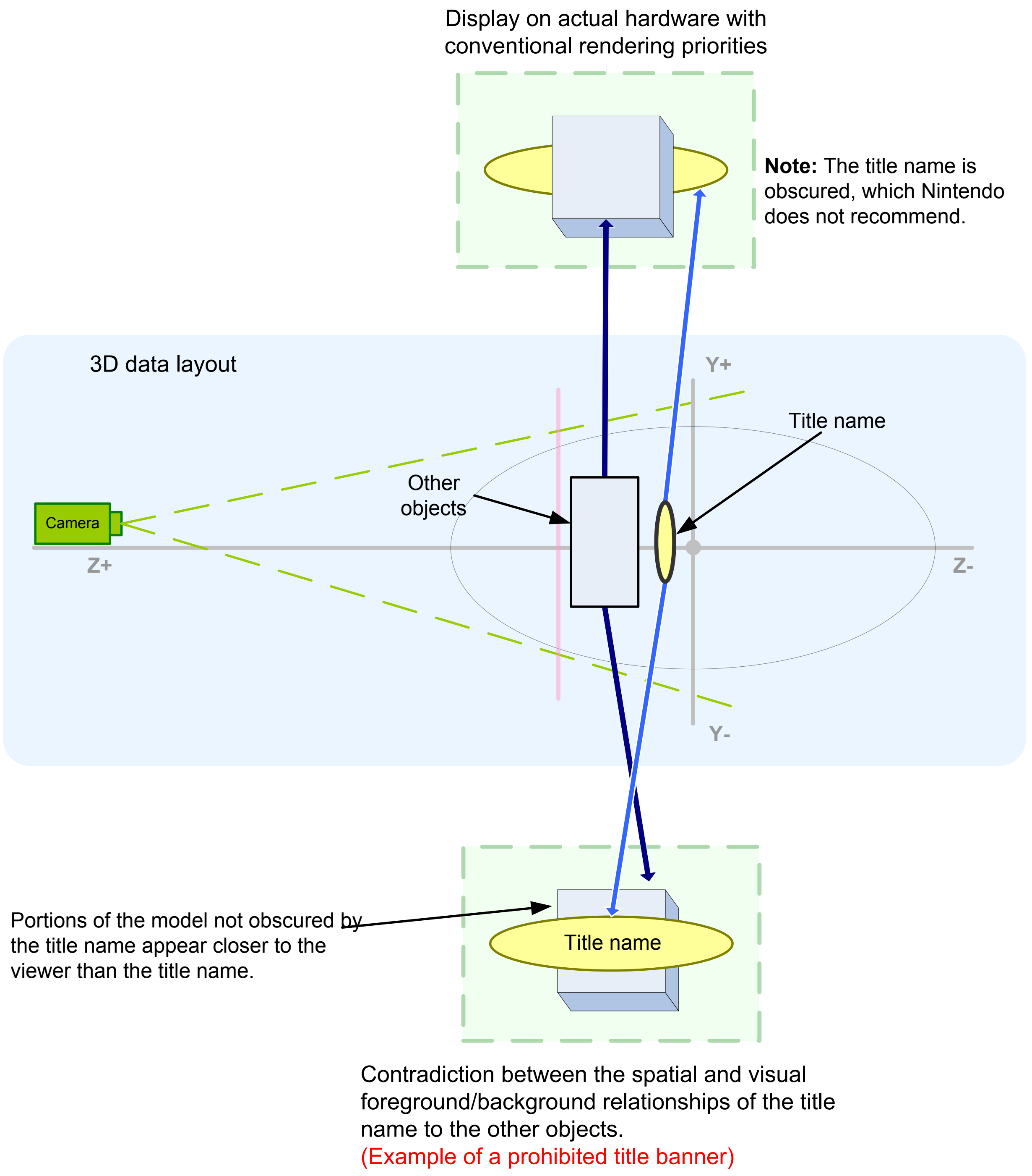
The HOME Menu switches the title banner shown on the upper screen when the user selects a different application on the lower screen. The user would find it uncomfortable to see the parallax vary widely from one application to the next. Nintendo therefore recommends placing the 3D model's title—which the user is particularly likely to focus on—somewhere between 0.0 and 9.0 on the z-axis, just like the built-in applications. The base plane, and by extension the CTR system's screen, is located at a z-coordinate of 10.0.

Figure 4-1 Recommended Placement Range of the Title Portion


Y+
Recommended positioning range for title name
Recommended near limit Z:9.0
Center of displayed model Z:0.0
Z+
Z-
Base plane (parallax of 0), Z:10.0
(Position where left and right rendered images are identical)
Y-

Some rendering priority settings for the 3D model may cause contradictions in the relative depths of objects in the 3D model, either spatially through their placement in 3D space or visually through the screen. Showing 3D models that use parallax and have these contradictions will cause the user discomfort. To avoid this, prevent both spatial and visual contradictions in the relative depths of objects.

Figure 4-2 Examples of Spatial and Visual Contradictions Arising in the Relative Depths of Objects in a 3D Model


Display on actual hardware with conventional rendering priorities
Note: The title name is obscured, which Nintendo does not recommend.
3D data layout
Y+
Title name
Other objects
Camera
Z+
Z-
Y-
Portions of the model not obscured by the title name appear closer to the viewer than the title name.
Title name
Contradiction between the spatial and visual foreground/background relationships of the title name to the other objects.
(Example of a prohibited title banner)

4.1.3 Support for Language Settings

3D models have both common data that is shown for all language settings and language-specific data that varies with individual language settings. You must prepare language-specific data for all languages that are valid in your software's market.

If you have prepared different 3D models for the common data and the language-specific data, both will be shown. If both language-specific data and common data have a texture with the same name, the language-specific version is shown. In other words, you need to use language-specific data to show different models for different languages and to paste different textures on the common data model.

Be careful if you are considering a demo for a networked interactive retail display (IRD). For details, see section 10.2 Bug in Menus Displayed for Demos on Networked Interactive Retail Displays.

4.1.4 Title Name Notation

Title banners are used to show the user which type of application is selected in the lower screen. You need to include your application's title in its 3D model because the HOME Menu will not show the application's title on the lower screen when the user has rearranged the layout to use two or more rows of CTR icons.

This restriction is intended to prevent the user from losing track of which application is selected. It is not necessary, therefore, to show the precise title correctly. You may, however, use abbreviations and change some notation (from "2" to "II", for instance).

Nintendo recommends, but does not require, that you show the title in each of the languages supported by your application. If you will only use a single 3D model to support every language, use the file specified as the English title in the BSF file.

Furthermore, to make the title more recognizable, Nintendo recommends configuring the title as a non-rotating billboard that is easier to see than other parts of the 3D model.

4.1.5 Data Naming Conventions

Data is named using a combination of your software's market and language. With the exception of consecutively-numbered texture data, data for the same market and language must use the same name for all file extensions.

Table 4-2 Data Names with Corresponding Markets and Languages

Name	Market	Language
`COMMON`	(Common to all markets)	(Common to all languages)
`JPN_JP`	Japan	Japanese
`USA_EN`	The Americas	North American English
`USA_FR`	The Americas	French (Canada)

Name	Market	Language
`USA_SP`	The Americas	Spanish (Latin America)
`USA_PO`	The Americas	Portuguese (Brazil)
`EUR_EN`	Europe	British English
`EUR_FR`	Europe	French
`EUR_GE`	Europe	German
`EUR_SP`	Europe	Spanish
`EUR_IT`	Europe	Italian
`EUR_DU`	Europe	Dutch
`EUR_PO`	Europe	Portuguese
`EUR_RU`	Europe	Russian
`CHN_CN`	China	Simplified Chinese
`KOR_KR`	Korea	Korean
`TWN_TW`	Taiwan and Hong Kong	Traditional Chinese

4.1.6 Intermediate File Restrictions

The intermediate files included the model data have the following restrictions. These restrictions apply to models that are displayed when there is a combination of common data and data specific to one language. You cannot use any intermediate data with an extension that is not in the table because it is unsupported. The default shader is used to display the model.

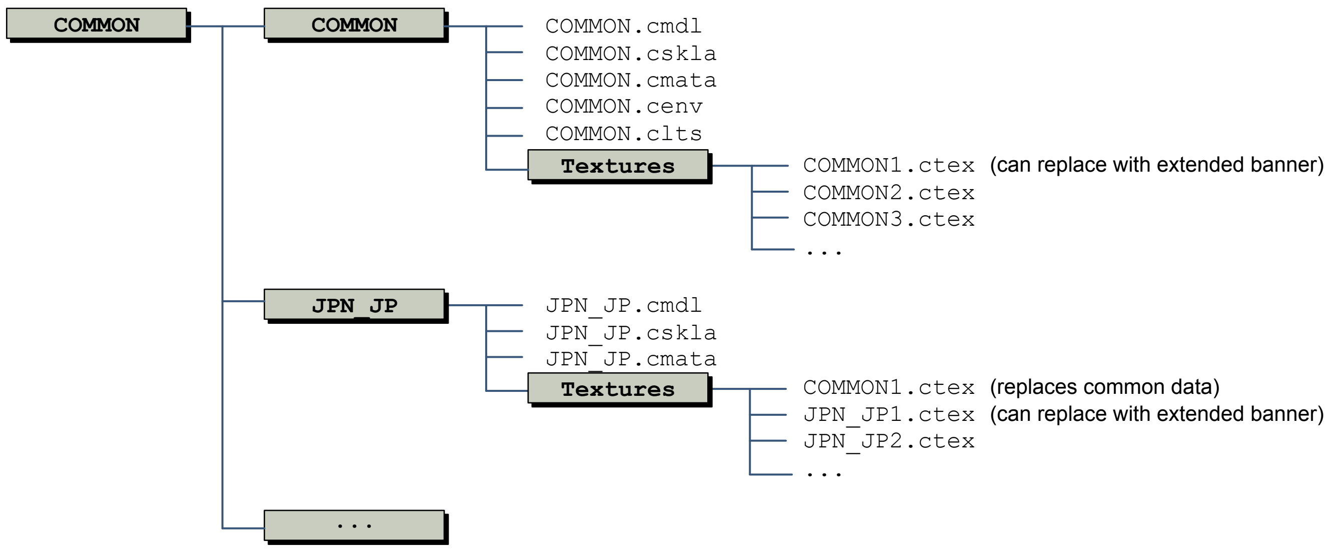
Table 4-3 Restrictions and Specifications for Intermediate Files Included in Model Data

File Extension	Restrictions and Specifications
`cmdl`	Create models with no more than 3000 polygons, 5 bones, and 5 materials. You can choose from four layer configurations (with one cycle): 0, 1, 2, and 3. You cannot use stencil tests.
`ctex`	Apart from capacity restrictions, you can use as many of these as you want. Textures in the language-specific data replace textures with the same name in the common data that are pasted onto models.
`cskla`	Create a looping animation of no more than 600 frames. This is rendered at 60 frames per second.
`amata`	Create a looping animation of no more than 600 frames. This is rendered at 60 frames per second.
`cenv`	Do not configure more than three lights, three cameras, and one fog object.
`clts`	You can use up to three lookup tables.

File Extension	Restrictions and Specifications
`cptl`	Particles are not supported. You don't need this file.
`csdr`	User shaders are not supported. You don't need this file.
`clgt`	Use `cenv` for light settings. Light animations are not supported. You don't need this file.
`ccam`	Use `cenv` for camera settings. Camera animations are not supported. You don't need this file.
`cres`	Consolidated files are not supported. You don't need this file.

Create directories for each market/language combination and then place intermediate files under them, as follows. Place language-specific texture data in the `Textures` directory—separately from the language-specific model data—using filenames like `COMMON1.ctex`. This language-specific texture data replaces textures in the common data.

Figure 4-3 Directory Structure for Intermediate Files and Model Data



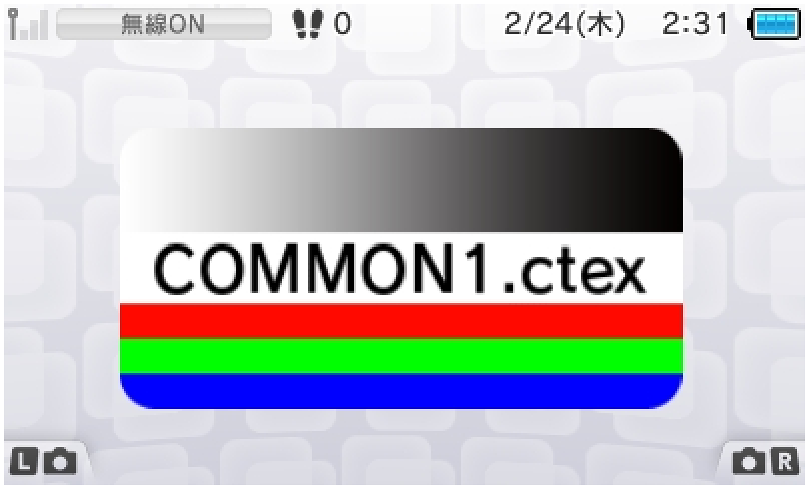
A 3D model can use any number of textures in any format, but the total size of the uncompressed model data (the common data and data for a single language) cannot be more than 256 KB.

All of the compressed data (the entire banner, including common data, all language-specific data, and banner sounds) must not exceed 1 MB.

4.1.7 Miscellaneous

Nintendo is providing a flat polygon as a template for developers who are not familiar with how to create 3D models. You can simply create a texture and paste it onto the polygon to display it.

Figure 4-4 Template That Displays a Flat Polygon



4.1.8 Convert the Intermediate Data Into Model Data

Once you have finished editing the model data in CreativeStudio, use `ctr_BannerModelConverter` to create CBMDL data. Specify the name of the folder where the model data's intermediate files are stored as an argument.

Code 4-1 Convert to Model Data

```
ctr_BannerModelConverter32.exe BannerModelFolder
```

4.2 Create Banner Sound Data

When you select an application's CTR icon from the HOME Menu, that application's banner appears on the upper screen and a banner sound is played. This indicates to the user that the application is selected and can be launched. You must therefore play an application-specific sound when its title banner appears. You cannot register sample data provided by Nintendo or files without sound.

To create a banner sound, use `ctr_WaveConverter` to convert audio data that conforms to the following restrictions into the BCWAV format.

Your banner sound must use approximately the same volume as system applications. You cannot synchronize banner sound playback with the 3D model's movements. The banner sound is played back slightly after the banner appears, so you don't need any blank space at the beginning of the sound. There is only one type of banner sound; it is not language-dependent.

Note: A banner sound is played back in front bypass mode. It is therefore played back in stereo even when surround sound is not configured in System Settings.

The following restrictions apply to the audio data used by banner sounds.

Table 4-4 Restrictions on Audio Data Used by Banner Sounds

Property	Restrictions
Length	No more than three seconds
Source file format	WAV or AIF
Number of channels	Stereo only
Compression method	DSP ADPCM, PCM16, or PCM8
Sampling rate	Any
Miscellaneous	No looping

4.3 Create Banner Data

Use `ctr_makebanner` to combine the banner model data and banner sound data that you have created in the previous sections to create BNR data. Specify the filenames for the banner model data and banner sound data in your BSF file.

Code 4-2 Sample BSF File Denotation

```
ModelFile:      banner_model.cbmd    # Banner model filename
SoundFile:      banner_sound.bcwav   # Sound filename
```

These are the only settings in the BSF file that you need to create banner data, but you must still configure other settings in the BSF file as appropriate. For details, see `$CTR_SDK/documents/tools/ctr_makebanner.html` in the CTR-SDK. Use settings that are relevant to your application's specifications.

Once you have prepared a BSF file, use `ctr_makebanner` to create a BNR file.

Code 4-3 Sample Invocation of the `ctr_makebanner` Command

```
ctr_makebanner32 -noicn BannerSpec.bsf banner_file.bnr
```

Note: For the sake of simplicity, the command here only creates banner data. You would ordinarily create both icon and banner data at the same time.

4.4 Embed in the ROM

Pass the BNR data that you created in the last section as an argument to `ctr_makerom` when you build your application.

Code 4-4 Sample Invocation of the `ctr_makerom` Command

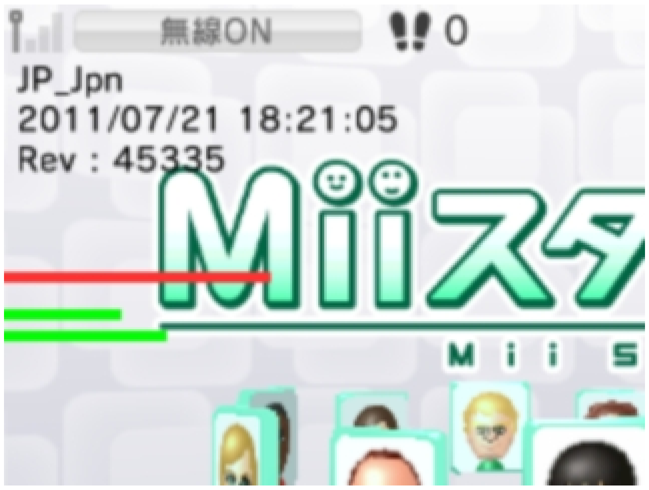
```
% ctr_makerom -banner banner_file.bnr
```

You are now done creating and embedding a CTR title banner.

4.5 Check the Processing Load

By pressing the X and Y Buttons simultaneously at the HOME Menu, you can show a banner that measures banner processing on the upper screen.

Figure 4-5 Banner With Processing Load Measurement Bars



The length of the first bar on top represents 100% processor use in a single frame (for comparison). The second bar represents processing with the CPU and the third bar represents processing with the GPU. A green bar indicates less than 75% processing, a yellow bar indicates 75–80% processing, and a red bar indicates more than 80% processing. Keep these bars green and yellow while your application's banner is shown. Revise your banner if any of the bars turn red.

Check the processing load in the following situations, because it may vary.

- When the model rotates slowly, changing its appearance (to face backwards, for example).
- When a model rotates quickly in response to microphone input.
- When textures are changed and text displayed by contextual banner data.

5 How to Create CTR Contextual Banner Data

There are two types of CTR contextual banner data: local contextual banners and downloadable contextual banners.

To create a local contextual banner, you can either create the banner data in advance and then write it to expanded save data, or your application can dynamically create the banner data and then write it to expanded save data.

You follow almost the same steps to create downloadable contextual banner data as you do to create local contextual banner data in advance.

5.1 Replaceable Data

CTR contextual banners allow you to modify elements of your CTR title banners: you can replace the texture data used in your 3D models and add text that scrolls across the bottom of the upper screen.

CTR contextual banner data can store one instance of texture data and one instance of text data. You can also use just one or the other. Texture data also includes information that specifies which texture it will replace. Text data does not include information that specifies what it will replace, because it only replaces a single location at the bottom of the upper screen.

There are two types of CTR contextual banner data: local contextual banners, which are created from applications, and downloadable contextual banners, which are downloaded via SpotPass. When CTR contextual banner data replaces elements of a CTR title banner, local contextual banners and downloadable contextual banners are applied independently. Downloadable contextual banners take priority over local contextual banners and are applied when both banners are supposed to replace the same thing.

Like banner model data for CTR title banners, CTR contextual banner data comprises both common data for all language settings and language-specific data for individual language settings. When CTR contextual banner data replaces elements of a CTR title banner, common data and language-specific data are applied independently. Language-specific data takes priority over common data and is applied when they are both supposed to replace the same thing.

When you use local contextual banners, downloadable contextual banners, common data, and language-specific data together, they are given the following priorities when replacing the same element.

Table 5-1 Order in Which Contextual Banner Data Is Applied

Priority	Applied Data
1	Language-specific data in downloadable contextual banners.
2	Common data in downloadable contextual banners.
3	Language-specific data in local contextual banners.

Priority	Applied Data
4	Common data in local contextual banners.
5	The original data in the CTR title banner.

On the other hand, if different elements are being replaced, the contextual banner data at priorities 1–4 above are replaced simultaneously. Up to four textures can be replaced at once.

Note: Be careful when you are considering replacing the texture data in the contextual banner. For details, see section 10.3 Bug That Causes the Home Menu to Freeze When Replacing Texture Data in the CTR Contextual Banner.

5.2 How to Create Replacement Data

5.2.1 Replacement Texture Data

Prepare replacement texture data as binary data in the following formats. The texture data must have the same resolution as the texture it is replacing, with a width and height that are both powers of 2 from 8 to 1024.

To create binary data, output NW4C TGA files with NW4C for Banner's Adobe Photoshop plug-in and then remove the header and padding region by specifying the `-dsl` option to `ctr_TexturePackager`.

Note: Starting with version 3.2 of the CTR-SDK, `ctr_make_ex_banner` can directly use NW4C TGA files as texture data.

Table 5-2 Texture Data Formats That Can Be Used by Contextual Banner Data

Format	Description
`RGBA8`	4 bytes per pixel, with 8-bit RGBA channels.
`RGB8`	3 bytes per pixel, with 8-bit RGB channels.
`RGBA5551`	2 bytes per pixel, with 5-bit RGB channels and a 1-bit alpha channel.
`RGB565`	2 bytes per pixel, with a 5-bit R channel, 6-bit G channel, and 5-bit B channel.
`RGBA4`	2 bytes per pixel, with 4-bit RGBA channels.
`LA8`	2 bytes per pixel, with 8-bit alpha and luminance channels.
`HILO8`	2 bytes per pixel, with 8-bit X and Y channels.
`L8`	1 byte per pixel, with an 8-bit luminance channel.
`A8`	1 byte per pixel, with an 8-bit alpha channel.
`LA4`	1 bytes per pixel, with 4-bit alpha and luminance channels.
`L4`	4 bits per pixel, with a 4-bit luminance channel.
`A4`	4 bits per pixel, with a 4-bit alpha channel.

Format	Description
`ETC1`	A compressed format with the equivalent of 4 bits per pixel.
`ETC1A4`	A compressed format with the equivalent of 8 bits per pixel.

If your application creates local expanded save data dynamically, arrange it using a data format above. These are native hardware formats and can be used by images (textures) that are rendered to the framebuffer.

5.2.2 Replacement Text

Replacement text can have up to 255 characters. It appears unchanged if it is short enough to fit on-screen and it scrolls if it is too long.

Note: If the text is just barely too long to fit on-screen, it may not scroll and be partially hidden. This behavior is dictated by the HOME Menu's specifications. To work around this, avoid using strings that are just barely too long to fit on-screen or add some extra characters, such as blank spaces, at the end of your strings.

For more information on which replacement characters can be used, see the reference page for the `ctr_make_ex_banner` tool (`$CTR_SDK/documents/tools/Exbanner.html`).

5.3 Statically Creating Local Contextual Banner Data

One way to create local contextual banner data is to create a binary with the contextual banner data in advance and save it in your application's ROM, then write the binary to expanded save data.

Use the `ctr_make_ex_banner` tool. Specify the required settings in an EBSF file and then use `ctr_make_ex_banner` to output binary data (a BIN file). You can set an expiration date in your EBSF file, but local contextual banners will ignore it.

Code 5-1 Sample Invocation of the `ctr_make_ex_banner` Command

```
% ctr_make_ex_banner32.exe EX_BANNER_SPEC_FILE [OUTPUT_DIRECTORY]
```

Code 5-2 Sample Settings in an EBSF File

```
FileName:JPN_JP
TexWidth:64
TexHeight:64
TexFormat:RGBA4
TextureName:sampleTexture
Text: Sample text
TextureDataFileName:./img/sample.tex
```

BIN files are created for either common or language-specific data. You do not need to create contextual banner data for all the languages that are supported in your market; you may simply create contextual banner data for the languages that you want to replace. This is not required for common data, either. Each BIN file that you use as a local contextual banner must be 128 KB or less.

To use these BIN files in your application, place them under a top-level directory named `ExBanner` in your expanded save data. This allows the HOME Menu to search for banners in expanded save data and then replace the banner that is displayed.

Change the BIN files in the expanded save data to change which data is replaced. To stop replacing data, remove the corresponding BIN file from the expanded save data.

5.4 Dynamically Creating Local Contextual Banner Data

The previous section explained how to create binary data in advance with the `ctr_make_ex_banner` tool. Applications can also dynamically create binary data to use as local contextual banner data. This allows you to apply user names, in-game screenshots, and other user-specific data to a banner.

Follow the same steps as you would to create local contextual banner data statically, up through creating expanded save data and the "ExBanner" directory. Save binary data for the structure shown in Code 5-3 (and defined by `$CTR_SDK/include/nn/pl/CTR/pl_ExBanner.h`) in BIN files in that directory. Fill unused elements in the structure with zeros. A variable-length array starts at `tex_data`. Use it to place the texture data described in section 5.2.1 Replacement Texture Data.

As when you create data statically, you must create a BIN file for common or language-specific data, each BIN file must be no larger than 128 KB, and BIN files must be modified or deleted to change the content that is replaced or to stop replacing it altogether.

Code 5-3 Expanded Banner Data Structure

```
struct ExBanner
{
  u16 tex_width;              // Texture width
  u16 tex_height;             // Texture height
  u8 tex_format;              // Texture format
                              // (nn::pl::CTR::ExBannerTexFormat)
  u8 pad[3];

  s32 limit_year;             // Expiration date (at 11:59 p.m.)
  s32 limit_month;            // However, this is ignored by local
                              // contextual banners
  s32 limit_date;

  char texture_name[16];      // Texture name to overwrite
  wchar_t text[256];          // Contextual banner message
  void *tex_data;             // Variable-length; the beginning of the
                              // texture data
};
```

5.5 Creating Downloadable Contextual Banner Data

You can use the `ctr_makedlexbanner` tool to combine the BIN data you created for individual languages in section 5.3 Statically Creating Local Contextual Banner Data and create downloadable contextual banner data. The downloadable contextual banner data must be no larger than 1 MB.

Code 5-4 Sample Invocation of the `ctr_makedlexbanner` Command

```
% ctr_makedlexbanner32.exe EXBNR_BIN_DIRECTORY_PATH [OUTPUT_FILE_NAME]
```

For more information on using `ctr_makedlexbanner`, see `$CTR_SDK/documents/tools/ctr_makedlexbanner.html` in the CTR-SDK.

You can register the BIN files that you create with the BOSS server, using contextual banner data to change the banner when your application initiates a download.

5.6 Banner and Application Settings

If you use a CTR contextual banner to switch CTR title banner elements, you must set the `ExBanner` flag in your BSF file when you create the CTR title banner. For more details, see `$CTR_SDK/documents/tools/ctr_makebanner.html` in the CTR-SDK.

Your application must create expanded save data when it uses local contextual banners. For more details, see the *CTR Programming Manual: System*. If you are using expanded save data, you must configure `UseExtSaveData` and `ExtSaveDataNumber` in your RSF file when you build your application. For more details, see `$CTR_SDK/documents/tools/ctr_makerom.html` in the CTR-SDK.

Your application must be able to handle BOSS download tasks when it uses downloadable contextual banners. For more details, see the *CTR Programming Manual: Wireless Communication*. You must also apply to use the BOSS server. For information on how to apply, contact support@noa.com.

6 How to Create Expanded Save Data Icons

You can create icons for expanded save data using the same procedure as you would use for CTR icons. See Chapter 3 How to Create CTR Icons for specific instructions. This chapter explains how expanded data icons differ from CTR icons, as well as how to implement expanded save data icons.

6.1 Creating Expanded Save Data Icons

As explained above, you can create icons for expanded save data using the same procedure as you would for a CTR icon. As a result, you may use the same data as you would for CTR title icons if you don't have a problem with the title and other information that is displayed.

Note: Parts of an expanded save data icon are sometimes not displayed correctly. For details, see section 10.1 Parts of an Icon Are Sometimes Displayed Incorrectly.

On the other hand, you can also create and use data that differs from the data for CTR title icons. Although you may change the icon's design, title, and other properties to fit the nature of the expanded save data, Nintendo recommends that you make it easy to determine which application the data is associated with. This is desirable because expanded save data is managed independently of applications; it must be easy to determine whether it is okay to delete the expanded save data.

In particular, when multiple game titles share expanded save data, Nintendo recommends configuring the expanded save data icon with a design and name that is easy to associate with its game title.

6.2 Implementing Expanded Save Data Icons

Store the ICN data for the expanded save data icon within the application’s ROM data and then use it when you create the expanded save data.

Code 6-1 Sample Implementation of an Expanded Save Data Icon

```
// Read the ICN file from the ROM archive
nn::fs::FileInputStream fileStream(L"rom:/ExtSaveData.icn");
size_t iconSize = fileStream.GetSize();
void* iconDataBuffer = s_ExtHeap.Allocate(iconSize);
fileStream.Read(iconDataBuffer, iconSize);

// Create the expanded save data
nn::fs::ExtSaveDataId extSaveDataId = 0xFFFFF;
u32 maxDirectories = 8;
u32 maxFiles = 8;
result = nn::fs::CreateExtSaveData(extSaveDataId, iconDataBuffer, iconSize,
maxDirectories, maxFiles);
if(result.IsFailure())
{
    // Handle any errors that occurred in the process of creating expanded
    // save data
}
```

You can only configure expanded save data icons when you create the data. You cannot change them later.

7 How to Create Icons for Download Play Child Programs

You can create icons for Download Play child programs using the same procedure as you would use for CTR icons. See Chapter 3 How to Create CTR Icons for specific instructions. This chapter explains how icons for Download Play child programs differ from CTR icons at creation.

7.1 Creating Icons for Download Play Child Programs

As explained above, you create icon data using the same procedure as you would for a CTR icon.

A CTR icon's title (name) must be set properly in all regions because StreetPass may cause CTR icons to appear on CTR systems from other regions. However, Download Play is not established between a parent and child system from different regions (the child system will be unable to find the parent system while scanning). As a result, the titles (names) set in an icon for a Download Play child program can simply handle the languages for the region supported by that software.

Note: Parts of an icon for a Download Play child program are sometimes displayed incorrectly. For details, see section 10.1 Parts of an Icon Are Sometimes Displayed Incorrectly.

7.2 Implementing Icons for Download Play Child Programs

Pass the icon as an argument to `ctr_makerom` when you build the Download Play child program.

7.3 Banner Data for Download Play Child Programs

Download Play child programs are never launched directly from the HOME Menu and, as a result, CTR title banners do not have the opportunity to be displayed. For this reason and to reduce the download size, Nintendo recommends against embedding banner data in your ROM.

If you omit the `-banner` option when you run `makerom`, banner data will not be embedded in your ROM.

The following warning message will appear if banner data is embedded in your Download Play child program when you check it with Master Editor. As explained in the warning message, you may include banner data in your ROM if you want to.

Code 7-1 Warning Message in Master Editor

```
This ROM has BNR data. Download Play child programs do not need BNR data.
Although you may choose to include this data, it is unnecessary and will
not be used. Ignore this message if you want to include the BNR data.
```

8 How to Create Icons for StreetPass Boxes

StreetPass boxes use normal texture data for their icons. Set these icons when applications create StreetPass boxes.

8.1 Creating Icons

Prepare a 48x48-pixel texture in the (little-endian) RGB565 native PICA format to use as an icon. To create texture data, output NW4C TGA files in the format specified by NW4C for Banner's Adobe Photoshop plug-in and then remove the header and padding region by specifying the `-dsl` option to `ctr_TexturePackager`.

Note: For information on native PICA formats, see the *CTR Programming Manual: Basic Graphics*.

Although you would ordinary use an icon of your game title, you may also use an icon that represents a series of game titles when your StreetPass box is shared by multiple games in a series.

Note: Parts of a StreetPass box's icon are sometimes displayed incorrectly. For details, see section 10.1 Parts of an Icon Are Sometimes Displayed Incorrectly.

8.2 Implementing Icons

8.2.1 Creating a StreetPass Box

Call the `nn::cec::MessageBox::CreateMessageBox` function to create a StreetPass box. Pass your icon's texture data as an argument to this function.

Code 8-1 Sample Implementation when Creating a StreetPass Box

```
// Prepare the icon data
u16 iconData[48*48];
memset(iconData, 0xf0f0, sizeof(iconData)); // Create a simple texture with
                                            // only one color

// Create a StreetPass box
u32 cecTitleId = nn::cec::MakeCecTitleId(0x12345); // StreetPass ID
u32 privateId = 0xaabbccdd; // Private ID (the box's key)
char hmacKey[] = "12345678901234567890123456789012";  // Message key
wchar_t boxName[] = L"SampleMessageBox"; // Box name

result = cecMessageBox.CreateMessageBox(
            cecTitleId, privateId, hmacKey,
            iconData, sizeof(iconData),
            boxName, sizeof(boxName) );
```

For simplicity, the sample code above generates a texture with only one color. In your actual application, load and use appropriate image data.

8.2.2 Changing Icons

Once you create a StreetPass box, you can change its icon at any time. Pass the texture data for the icon you want to change to the `nn::cec::MessageBox::SetMessageBoxIcon` function.

9 How to Create Icons for StreetPass Messages

StreetPass messages use normal texture data for their icons. Set these icons when you create StreetPass messages.

9.1 Creating Icons

Prepare a 40x40-pixel texture in the (little-endian) RGB565 native PICA format to use as an icon. To create texture data, output NW4C TGA files in the format specified by NW4C for Banner's Adobe Photoshop plug-in and then remove the header and padding region by specifying the `-dsl` option to `ctr_TexturePackager`.

Note: For information on native PICA formats, see the *CTR Programming Manual: Basic Graphics*.

Configure the design of the icons however you like according to the messages' content. Icons cannot have user-generated content (UGC) unless the UGC does not have to be supported by the local blacklist.

A CTR system receives StreetPass data along with its icon and then shows the icon in the Notifications List. The Notifications List does not support the local blacklist as instructed by the *CTR Guidelines: UGC*. If icons for StreetPass data were to use images that are classified as UGC, users could see UGC images sent by other users whom they found offensive in the past.

For more information on UGC—and UGC that needs to be supported by the local blacklist—see the separately distributed *CTR Guidelines: UGC*.

9.2 Implementing Icons

Create a new message with the `nn::cec::CTR::Message::NewMessage` function and then set the icon data with the `nn::cec::CTR::Message::SetIcon` or `nn::cec::CTR::Message::SetExHeader` function.

Code 9-1 Sample Implementation of StreetPass Messages

```
// Create a message
nn::cec::Message cecMessage;
cecMessage.NewMessage( cecTitleId, groupId, messFlag,
                       sendMode, sendCount, propCount );

// Set the message body
char messageBody[] = "Hello!";
cecMessage.SetMessageBody(messageBody, sizeof(messageBody));

// Set the icon
u16 iconData[40*40];
memset(iconData, 0xf0f0, sizeof(iconData)); // Create a simple texture with
                                            // only one color
cecMessage.SetIcon(iconData, sizeof(iconData));
```

```
// Set the message title
char messageTitle[] = "Message received.";
cecMessage.SetInfoText(messageTitle, sizeof(messageTitle));

// Save the message to the outbox
result = cecMessageBox.WriteMessage(cecMessage, nn::cec::CEC_BOXTYPE_OUTBOX,
retMessId);
```

For simplicity, the sample code above generates a texture with only one color. In your actual application, load and use appropriate image data.

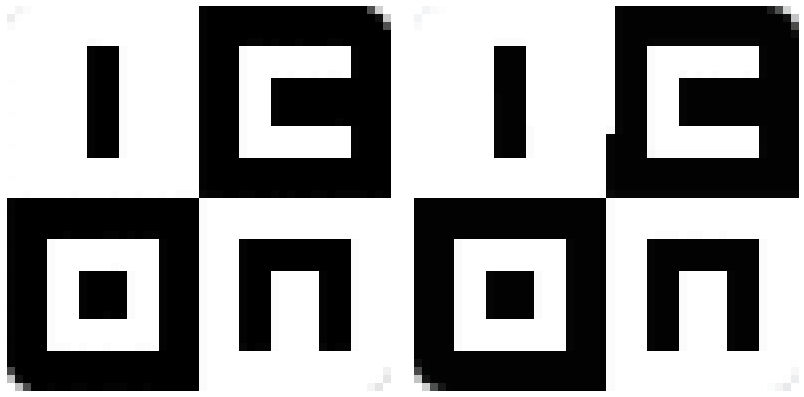
For more details, see the *CTR Programming Manual: Wireless Communication*.

10 Notes and Precautions

10.1 Bug Where Parts of an Icon Are Displayed Incorrectly

As shown in Figure 10-1, some built-in system applications do not display all icons properly.

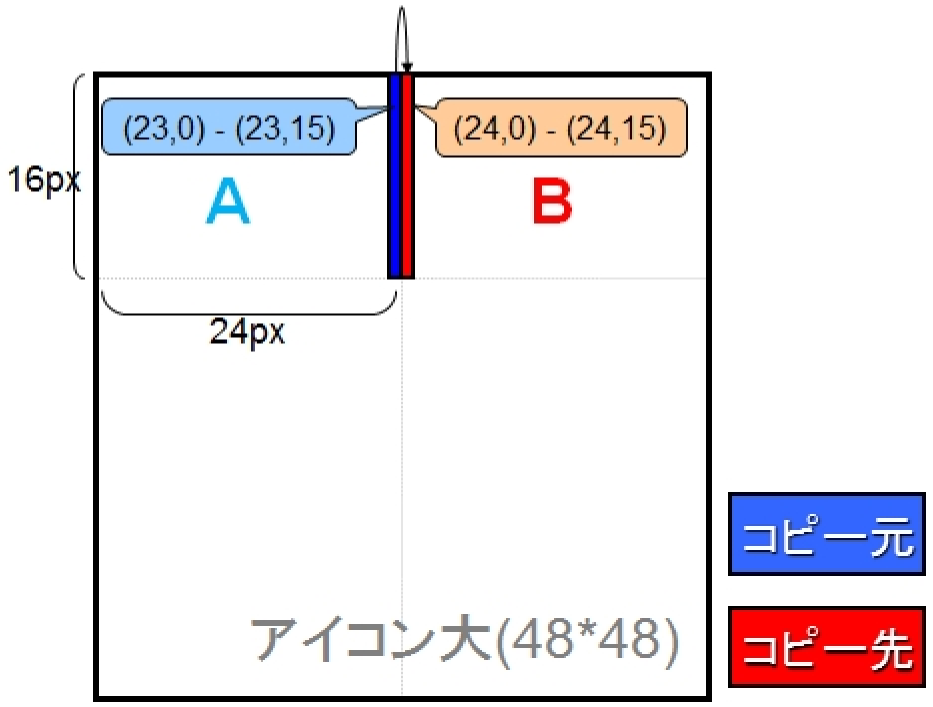
Figure 10-1 Close-Up of an Icon That Is Partially Corrupted



The icon on the left is rendered correctly and the icon on the right is partially corrupted. There is an area at the top of the icon, in the middle, that is white even though it should be black.

This is a symptom of a bug in the built-in system applications. As shown in Figure 10-2, part of area A in the 48x48-pixel icon is copied to area B.

Figure 10-2 Underlying Cause of the Symptom



To prevent this symptom from affecting your icons, design them so that the area that is copied (as shown above) is the same or similar to the area that it is copied to.

Table 10-1 shows which built-in system applications exhibit this behavior and at what times. It also shows when this behavior is scheduled to be fixed. Built-in applications for the Chinese, Korean, Taiwanese, and Hong Kong regions do not exhibit this behavior.

Table 10-1 Applications Affected by This Bug and Scheduled Fixes

Affected Applications	Timing	Affected Icons	Scheduled Fixes and Workarounds
System Settings	When downloadable applications' icons are displayed in the Software Management screen	CTR icons	This has been fixed in CTR-SDK 3.2.4. If you are using an older development environment, Nintendo recommends that you check your icons in the development version of System Settings because this symptom may occur on Nintendo 3DS systems in the marketplace.
	When icons for expanded save data are displayed in the Extra Data Management screen	Expanded save data icons	
	When icons for StreetPass boxes are displayed in the StreetPass Management screen	Icons for StreetPass boxes	

Affected Applications	Timing	Affected Icons	Scheduled Fixes and Workarounds
StreetPass Mii Plaza	When icons are displayed for Favorite Titles	CTR icons	This has been fixed in a second network update. The fix will only be applied through a network update, though, and thus not necessarily to all Nintendo 3DS systems in the marketplace. Nintendo recommends that you check icons in the development version of Activity Log.
Activity Log	When icons are displayed in the Software Library and elsewhere (excluding Page View)	CTR icons	
Download Play	When downloadable software is selected from a child device and an icon is displayed	Icons for Download Play child programs	This has been fixed in a third network update. However, because an icon for a child device program is also being displayed with the unmodified Download Play version, Nintendo recommends that you check the display on the Download Play development version.

10.2 Bug in Menus Displayed for Demos on Networked Interactive Retail Displays

If there is no texture data for a given language, a networked interactive retail display will search for and use texture data for another language when it shows banners in its menu. This means that even if you only wanted to use a special texture for a particular language and a common texture by default, your special texture will end up being used for all languages.

As a result, if you want to change the texture from one language to the next, you need to set a texture for every language.

Both menus in networked interactive retail displays and the tool that checks those menus—which is provided by the package for creating demos for networked interactive retail displays—have this bug. Nintendo does not plan to fix it. Please work around it by following the steps above.

10.3 Bug That Causes the Home Menu to Freeze When Replacing Texture Data in the CTR Contextual Banner

When texture data that references from multiple materials is replaced in the contextual banner, an internal memory leak occurs when said banner is displayed in the HOME Menu.

For this reason, when another application on the HOME Menu is selected, and after another banner is displayed initially, the HOME Menu freezes after operations such as display of a banner with replaced texture data are repeatedly performed.

This bug existed in the HOME Menu, but was fixed with 3.2NUP. However, please try to avoid this bug when distributing contextual banners to systems before 3.2NUP.

Note that when replacing texture data, you can check that this bug does not occur when repeatedly displaying the contextual banner in the HOME Menu multiple times. The following is the replacement texture size and recommended number of debugging tries

- 80 times when texture size is 32 KB
- 40 times when texture size is 64 KB
- 20 times when texture size is 128 KB
- 10 times when texture size is 256 KB

Revision History

Version	Revision Date	Category	Description
1.3	2012/06/27	Changed	Updated the description in 10.3 Bug That Causes the Home Menu to Freeze When Replacing Texture Data in the CTR Contextual Banner.
1.2	2012/04/27	Changed	Added explanation of CTR icon information used in the notification applet in table 2.1 of section 2.1.1. Updated information for “Bug where Parts of an Icon Are Displayed Incorrectly” in table 10, section 10.1.
		Added	Added an explanation in section 10.3 Bug That Causes the Home Menu to Freeze When Replacing Texture Data in the CTR Contextual Banner.
1.1	2012/02/24	Changed	Revised some expressions and terminology.
		Added	In section 10.1 Parts of an Icon Are Displayed Incorrectly, noted that this bug does not occur in the China, Korea, and Taiwan regions.
1.0	2012/01/17	—	Initial version.