Display Drivers

Display drivers are plugins that receive image data from the renderer and output it in some fashion. Generally, they will either send it to a file or display it immediately to the screen.

In terms of files, the two main drivers for writing shallow images for production use in RenderMan write OpenEXR and TIFF images, respectively. Both of these drivers can create files with pixels ranging from 8-bit quantized all the way up to full floating point HDRI, and with any number of channels per pixel. Both drivers can write out periodic checkpoint images for incremental rendering, though currently PRMan can only restore from OpenEXR images.

For screen display, a good choice is the "it" driver for sending images to it3 for display with RenderMan installed. This flexible viewer lets you easily scroll, zoom, adjust exposure, and review previous renders among other things.

Finally, the deepexr driver can produce deep images in the OpenEXR 2.0 format.

openexr

This driver supports OpenEXR, a high dynamic-range image, floating point file format developed by Industrial Light & Magic.

The driver supports an optional `"int asrgba"` option. If set to a non-zero value this causes the driver to take the first four channels and always calls them RGBA. Try using this if your image viewer or compositing software swaps around channels (such as red or blue) incorrectly.

This display driver also supports the output of image channels other than `"rgba"` using the Arbitrary Output Variable mechanisms.

This driver maps RenderMan's output variables to image channels as follows:

RenderMan Output Variable Name	Image Channel Name	Type
r	R	preferred type
g	G	preferred type
b	B	preferred type
a	A	preferred type
z	Z	float
other	same as output variable name	preferred type

By default, the "preferred" channel type is the value specified for `/display/openexr/type` in `rendermn.ini`, which can be either `half` (16-bit) or `float` (32-bit). If not specified therein, it defaults to `float`. The preferred type can be changed by adding a `"type"` argument to the `Display` command in the RIB file. For example:

```
# Store point positions in HALF format
Display "gnome.points.exr" "openexr" "P" "string type" "half"
```

The default compression method for the image's pixel data is the value specified for `/display/openexr/compression` in `rendermn.ini`, which may take the value of `none`, `rle`, `zip`, `zips`, `pixar`, `b44`, or `piz`; if not specified therein it defaults to `zips`. You can select a different compression method by adding a `"compression"` argument to the Display command. For example:

```
# Store RGBA using run-length encoding
Display "gnome.rgba.exr" "openexr" "rgba" "string compression" "rle"
```

Note, the OpenEXR display driver also supports the `-recover` option to PRMan which allows PRMan to restart rendering where it aborted if the same file is rendered a second time.

This driver supports both scanline and tile-based file output by adding a `"storage"` argument to the Display command that can be either `"scanline"` or `"tiled"`. For example:

```
# Store RGBA using in a tiled format
Display "gnometiled.rgba.exr" "openexr" "rgba" "string storage" "tiled"
```

The default is scanline-based output, which is the format written by previous versions of PRMan. The new tiled format optimizes the amount of internal buffering needed, which can reduce the display buffer memory footprint, especially for large images with many display channels and non-horizontal bucket orders. Note that the `-recover` option works only for horizontal and vertical bucket orders with the tiled format.

Finally, this driver also supports the injection of arbitrary metadata. This is done by passing named parameters with the form `"type exrheader_$key"`, where *$key* will be the name of the attribute as it shows up in the OpenEXR header. For example,

```
Display "myfile.exr" "openexr" "rgba" "string exrheader_author" ["Bob"]
```

Will result in `exrinfo myfile.exr` reporting:

```
author (type string): "Bob"
```

Any number of attributes of this form can be passed, with any inline type (e.g. `string`, `int`, `int[2]`, `float`, `float[4]`, etc.).

There is also the option for autocrop, which will reduce the data window to an area that bounds the non-zero alpha of the image. This can reduce the size and processing times of EXRs.

tiff

This driver produces an output file in Tagged Image File Format (TIFF). It can be used to store R, RGB, RGBA images in 8-, 16-, or floating point 32-bit-per-component resolutions. Alternately, it can store an arbitrary number of AOVs. The TIFF file created employs LZW compression; 8, 16, or 32 bits per sample; and a planar contiguous configuration.

The tiff driver also has special configuration options to control the resolution information written into the output file as well as the compression used for output. These may be set from the Display parameter list or through the configuration file.

The TIFF compression scheme can be controlled by setting the parameter `"compression"`, which takes a string value that should be one of `"lzw"`, `"packbits"`, `"deflate"`, `"pixarlog"`, or `"none"`. The default value for the compression scheme may be set by adding the following line:

```
/display/tiff/compression compression-scheme
```

where compression-scheme is is one of the above strings. If no compression scheme is specified, LZW compression is used.

The TIFF resolution unit can be controlled by adding the parameter `"resolutionunit"`, which takes a string value that should be one of `"inches"`, `"centimeters"`, or `"none"`. The default value for the resolution unit may be set by adding the following line to your `rendermn.ini`:

```
/display/tiff/resolutionunit unit
```

where unit is is one of the above strings. If no units are specified the default is `"none"`.

If the TIFF resolution unit has been defined as above, the TIFF resolution values can be controlled by setting the parameter `"resolution"`, which takes a value that is an array of two floats specifying the resolution in the x and y directions. The default value for the resolution values may be set by adding the following line:

```
/display/tiff/resolutionunit xresolution,yresolution
```

where *xresolution,yresolution* are floating-point numbers separated by a single comma. There should be no spaces between the comma and *xresolution*. If no values are specified, values of 100.0 are used, except in the case where no units are specified, in which case values of 1.0 are used.

The default image size and pixel aspect ratio are set through the configuration file as follows:

```
/display/tiff/xres xresolution
/display/tiff/yres yresolution
/display/tiff/par pixelaspectratio
```

it

This driver will launch the image tool ("it") image viewer if it is not already open and send the render to it. This is a good choice for interactive use as it lets you see the image in progress as it renders, pan, zoom, adjust exposure or gamma, compare renders, take notes, and do other useful things. "It" also supports interactive IPR rendering when used from RfM, RfK, or another digital content creation application.

deepexr

This driver writes out deep images in the OpenEXR 2.0 format. Like the "openexr" driver for shallow images, it supports multiple channels (either basic RGBA or arbitrary output variables), a choice between half and float for storage, different compression settings, scanline and tiled images, and arbitrary metadata. It also supports the same mapping between the RenderMan channel names and the OpenEXR names for R, G, B, and A, as well as the `asrgba` option.

One of the main differences are that it always automatically includes Z and ZBack channels for depth information (see the Interpreting Deep Pixels white-paper from ILM for more details on these) and these two channels are always stored as the full float type regardless of the choice of type made for the other channels. The default type may be set using the `/display/deepexr/type` line in `rendermn.ini`. Also, the choice of compression methods is limited to `none`, `rle`, and `zips`. The default for this can be set in the `/display/deepexr/type` line in `rendermn.ini`. If not set there, it defaults to `zips`. Finally, noted that the default storage type for the deepexr driver is `tiled` rather than `scanline` like the default in the shallow openexr driver.

You can use the render option for deepshadowerror to control the merging of samples where larger values merge more samples but may introduce precision errors in compositing.

Other drivers

- targa – writes a Truevision, Inc. file of type 2, which is an uncompressed RGB or RGBA image.
- png - writes a PNG file with transparency, RGBA format often used on the web
- texture – produces an output file that will be accepted directly by the renderer for use in a texture node.