NAME

rholo - generate/view a RADIANCE holodeck

SYNOPSIS

rholo [**−n npr**][**−o dev**][**−w**][**−i**][**−f** | **−r**] **hdkfile** [**varfile** | **+** | **-** [**VAR=value ..**]]

DESCRIPTION

Rholo is a program for generating and viewing holodeck files. Similar to *rvu(1), rholo* can compute views interactively, but unlike *rvu,* it reuses any and all information that was previously computed in this or earlier runs using the given holodeck file, *hdkfile*.

The *−n* option sets the number of *rtrace(1)* processes to start for the calculation. It defaults to zero, which means that no new rays will be calculated. In general, it is unwise to start more processes than there are processors on the system. On a multiprocessing system with 4 or more processors, a value one less than the total should yield optimal interactive rates on a lightly loaded system.

The −o option sets the output device to use for display. Currently, there are at least two display drivers available, *x11* and *glx*. If no output device is specified, then *rholo* will start a global calculation of the holodeck, filling it in as time goes by. The quality of the final holodeck will depend on how long *rholo* runs before it is interrupted or runs out of file space or time, according to the variable settings described in the control variable section, below. If no output device and no processes are specified, *rholo* creates an empty holodeck using the given *varfile,* if present.

The *−i* option provides for reading from the standard input. Without a display driver, the input should consist only of views, which will be used to limit which parts of the holodeck are rendered in a batch calculation. With a display driver, most of the commands understood by the driver can be issued either from the operating window or the standard input. These commands are described together with their window equivalents in the display driver section following the control variable section.

The *−f* option permits the given holodeck to be clobbered. Without this option, giving both the holodeck file and a variable file (or "-") will result in an error message if the holodeck exists, since giving both implies that a new holodeck is being created. (When reusing an existing holodeck, the variable values are taken from the holodeck header, though some may be overriden by giving a "+" in place of the variable file.) Also, attempts to clear the holodeck using the interactive "clobber" command will be permitted only if the *−f* option is given on the initial command line.

The *−r* option tells *rholo* to open the holodeck file read-only, which is the default if there are no ray calculation processes. If one or more *rtrace* processes are started with the *−n* option and the *−r* option is given or the specified holodeck is not writable by the user, then any additional rays computed during the session will be discarded rather than saved to the holodeck file.

One or more holodeck section boundaries are defined along with other parameters in the holodeck file or, if the holodeck is being created, the *rholo* control variable file, *varfile*. These section boundaries define where you may move, or at least, where you will be able to see, since they determine where computed rays are stored. Additional variable settings may be added or overridden on the command line following *varfile*. If no *varfile* is needed, a holodeck may still be created by giving a "-" on the command line in place of the variable file. If you wish to override some of the variable settings in an existing holodeck, use a "+", followed by the new settings on the command line. Upper case variables specified more than once will result in a warning message (unless the *−w* option is present), and the last value given will be the one used, unless it would conflict with something in an existing holodeck that cannot be changed, such as the section boundaries. Changing section boundaries requires creating a new holodeck using *rholo* without a *−n* or *−o* option, then running *rhcopy(1)* to fill the new holodeck with the old holodeck's contents.

The *−w* option turns off warnings about multiply and misassigned variables.

Rendering variable assignments appear one per line in *varfile*. The name of the variable is followed by an equals sign ('=') and its value(s). The end of line may be escaped with a backslash ('\'), though it is not usually necessary. Variables that should have only one value are given in upper case. Variables that may have multiple values are given in lower case. Variables may be abbreviated by their first three letters. Comments in *varfile* start with a pound sign ('#') and proceed to the end of line.

CONTROL VARIABLES

The control variables, their interpretations and default values are given below.

OCTREE The name of the octree file. The default name is the same as *hdkfile* but with any suffix replaced by ".oct". This variable may also be read from *rad(1)* if the "RIF" variable is set. (See below.)

RIF This variable specifies a *rad* input file to use as a source of rendering options and other variable settings. If given, *rholo* will execute *rad* and get the rendering options to later pass to *rtrace*. Besides prepending the *render* variable, *rholo* will also extract default settings for the common "OCTREE" variable, and the "EYESEP" variable. Following the file name, overriding variable settings may be given, which will be passed to *rad* on the command line. Settings with spaces in them should be enclosed in quotes. The execution of *rad* will also update the contents of the octree, if necessary. There is no default value for this variable.

EYESEP The interocular spacing for stereo viewing. I.e., the world distance between the pupils of the left and right eyes. There is no default value for this variable.

section A section is a parallelepiped given by an origin and three axis vectors (i.e., 12 floating point values in world coordinates). The axis vectors define the three edges attached to the origin vertex, and the other edges and vertices are determined by the parallel wall constraint. A holodeck section is a region in which the user may freely move about to obtain a view of what is outside that region. In object rendering mode, a section may instead contain a detailed object to be viewed from the outside. The grid dimensions for each axis may also be given by three additional integer arguments following the three axes. Otherwise, if the grid dimensions are left out or any are unspecified or zero, the "GRID" variable will be used to determine them from the section axis lengths. (See below.) There is no default value for this variable, and it is required. If multiple values are given, they will be used for multiple rendering sections, which may or may not be connected, but should generally not overlap. The starting view for interactive display will be the center of the first section facing the positive X direction unless "OBSTRUCTIONS" is set to True, when the view will be placed outside the first section. (See below for this variable's definition.) The third axis of the first section is also used as the default "view up" vector.

geometry This variable is used to associate geometry from an octree file with one or more sections. The specified octree will be used by certain drivers (e.g., the "ogl" driver) to display simplified geometry using hardware lighting during motion. If this variable is not set, such drivers will use the main octree file, which contains all the scene geometry. This can be slow if the scene is complex, so use simplified geometry with portals (described below) or specify a non-existent file to turn geometry rendering off. If there is just one setting of this variable, it will be used for all sections. If there are multiple settings, they will correspond to multiple sections.

portals This variable is used to associate portal geometry with one or more sections, as required for simplified geometry in some drivers (e.g., "ogl"). The portal geometry itself is given in one or more RADIANCE scene files or quoted commands beginning with an exclamation mark ('!'), and the input may or may not include material definitons. (I.e., the surfaces may be modified by "void" if there are no materials.) A portal is an imaginary surface that intervenes between a view and some detailed geometry not included in the current section. (See the "geometry" variable definition, above.) Portals are often placed in doorways, windows and in front of mirrors. Portal geometry may also be placed around local geometry that has been culled due to its complexity. This specification is necessary in order that the detail geometry be drawn correctly, and that mirrors will work with virtual distances. (See the definition of "VDISTANCE," below.) The orientation of the portal surface geometry is ignored, so they have effect no matter which way they are facing. If there is just one setting of this variable, it will be used for all sections. If there are multiple settings, they will correspond to multiple sections.

GRID The default section grid size in world distance units. If any section axis grid is unspecified, the length of the axis will be divided by this number and rounded up to the next larger integer. The grid size is a very important determiner of holodeck performance, since the holodeck beam

index is proportional to average axis grid dimension to the fourth power! If the beam index is too large, poor file and memory performance will result. If the beam index is too small, the holodeck resolution will suffer and objects will tend to break up. In general, the grid size should divide each section wall into 64 or fewer cells for optimal performance. The default value for this variable is the maximum section axis length divided by 8.

OBSTRUCTIONS

This boolean variable tells *rholo* whether or not to compute intersections with objects inside holodeck sections. If it is set to "False", then only objects outside the holodeck sections will be visible. This is appropriate when you know all sections to be devoid of geometry, or when some secondary method is available for rendering geometry inside each section. If it is set to "True," all inside geometry will be visible. There is no default for this variable, which means that rays will be started at random points within each holodeck section, allowing interior geometry to be partially sampled.

VDISTANCE

This boolean variable determines whether the actual distance to objects is computed, or the virtual distance. If it is set to "True," the virtual distance will be used, which will make reflections and refractions through smooth, flat objects clear, but will blur the boundaries of those objects. Note that some drivers cannot render virtual samples without the proper placement of "portals" in the scene. (See above for the definition of the "portals" variable.) If it is set to "False," the reflections and refractions will be blurred, but object boundaries will remain sharp. The default value for this variable is "False."

CACHE The memory cache size to use for ray samples during interactive rendering, in Megabytes. This tuning parameter determines the tradeoff between memory use and disk access time for interactive display. This value will not affect memory use or performance for global holodeck rendering if there is no display process. The default cache is effectively set to 16 Megabytes. If this variable is set to zero, no limit will be placed on memory use and the process will grow to accommodate all the beams that have been accessed.

DISKSPACE

Specifies the maximum holodeck file size, in Megabytes. Once the holodeck file reaches this size, *rtrace* will exit. If there is no display process, *rholo* will also exit. The default value for this variable is 0, which is interpreted as no size limit.

TIME Sets the maximum time to run rtrace, in decimal hours. After this length of time, *rtrace* will exit. If there is no display process, *rholo* will also exit. If there is a display process, and *rtrace* is restarted with the "restart" command, then the time clock will be restarted as well. The default value for this variable is 0, which is interpreted as no time limit.

REPORT This variable may be used to specify a interval for progress reports in minutes. If this value is zero, then progress reports will not be given in intervals, but a final report of the file size and fragmentation will be issued when the program terminates, along with the number of rays and packets computed. If a filename is given after the interval, it will be used as the error file for reports and error messages instead of the standard error. There is no default value for this variable.

render This variable may be used to specify additional options to *rtrace*. These options will appear after the options set automatically by *rad,* and thus will override the default values.

DISPLAY DRIVER

Rholo may be started in interactive mode using the *−o* option to specify an output display driver. Currently, three drivers are supported on most machines, *glx, ogl* and *x11*. (In addition, there are variations on the first two drivers for stereo displays, local objects and human tone mapping. These are accessed with some combination of the 's', 'o' and 'h' suffixes, always in that order. E.g., the OpenGL stereo driver with human tone mapping would be "oglsh".) Each driver accepts simple one-character commands and mouse view control in its operating window. If the *−i* option is also given, then the driver will also listen for commands entered on the standard input. (It is unwise to use the *−i* option when rholo is run in the

background, because it will occassionally stop the process when input is available on the controlling terminal.) The commands and their single-key window equivalents are given below.

VIEW= (mouse)
Modify the current view with the specified parameters. (See the *−v** view options in the *rpict(1)* manual page for parameter details.) There is no one-character equivalent for this command in the display window. Instead, the mouse is used to control the current view in the following ways:

CONTROL		MOUSE ACTION
(none)	left	Move forward towards cursor position
(none)	middle	Rotate in place (usually safe)
(none)	right	Move backward away from cursor position
shift	left	Orbit left around cursor position
shift	middle	Orbit skyward
cntl	middle	Orbit earthward
shift	right	Orbit right around cursor position
cntl+shift	any	Frame focus by dragging rectangle

For all movements but rotating in place, the cursor must be placed over some bit of visible geometry, otherwise the driver has no reference point from which to work. It is best to just experiment with these controls until you learn to fly safely in your model. And if you run into trouble, the "last" command is very useful. (See below.)

last 'l' Return to the previous view. Some drivers will save up multiple views in a history, but you are guaranteed at least one.

where 'v'
Print the current view parameters to the standard output. This is useful for finding out where you are, or for saving specific views in a keyframe file for animations or returning to later.

frame 'f'
Change the calculation focus. If the "frame" command is given with no arguments on the standard input, it is equivalent to the interactive 'F' command, which releases the current calculation focus. If the "frame" command is followed by a relative horizontal and vertical position (specified as floating point values between 0 and 1), then the new focus is about this position on the screen (where 0 0 is at the lower left of the display). This is equivalent to the interactive 'f' command, which sets the focus about the current window cursor position. If four relative coordinates are given, they are assumed to mean the minimum horizontal and vertical positon, and the maximum horizontal and vertical position, in that order. This is equivalent to dragging the mouse over a rectangular area with the 'cntl+shift' keys held down.

pause 'p'
Pause the ray calculation temporarily.

resume <cr>
Resume the ray calculation.

redraw ^L
Redraw the current view from values calculated and stored in the holodeck. When executed from the display window via '^L', the effect may be slightly different, since all stored information will be flushed.

kill 'K' Terminate the ray calculation process. This is usually unnecessary, but is provided for special purpose applications.

restart 'R'
Restart the ray calculation process. If the "RIF" variable has been set, *rad* will be run first to assure that the octree is up to date.

clobber 'C'
Clobber the holodeck contents, deleting all that has been calculated before. To get an interactive dissolve of changes to the scene description, use the sequence "kill," "clobber," "restart." This command will be honored by *rholo* only if it was started with the *−f* option.

quit 'q' Quit *rholo*. The ray tracing calculation is terminated and all values are flushed to the holodeck file. This is the normal way to exit the program.

In addition to these standard commands, all drivers offer the following supplimentary controls.

'h' Fix the head height. All mouse-controlled view motions will be adjusted so that the head height does not change (where vertical is determined by the current view up vector).

'H' Release the head height, allowing it to change again during mouse-controlled movements.

^R Redraw the current view, recomputing the tone mapping in the process. This is useful if the current view is too light or too dark. (On an 8-bit display, it may be necessary to redraw the screen a couple of times to get the best image.) The "^L" command is a stronger type of redraw, since it will use only rays in the current view to determine the tone mapping, rather than a history of rays drawn from the *rholo* server.

EXAMPLES

The following shows a minimal holodeck control variable file:

```
RIF= sample.rif                 # rad input file
section= 2 2 4  5 0 0  0 7 0  0 0 3   # section parallelepiped origin & vectors
```

Technically, the "RIF" setting is not necessary, but the results are much better when *rholo* is used in association with *rad* to control the rendering parameters.

Here is a slightly more sophisticated example:

```
RIF=electric.rif
section= 7 4 3.5  15 0 0  0 10 0  0 0 5
GRID= .75
CACHE= 20       # cache size in megabytes
TIME= 120       # maximum time in hours
DISK= 200       # maximum file size in megabytes
REPORT= 60 elect.her
OBST= False
VDIST= False
```

We can invoke *rholo* on the above file to compute a hologram overnight in batch mode:

```
rholo −n 1 elect.hdk elect.hif TIME=12 &
```

This will report progress every hour to "elect.her".

The next morning, we can look at the holodeck interactively:

```
rholo−n 1 −o x11 elect.hdk &
```

If the previous command were still running, the above command would fail because the permissions on the holodeck would not grant access. To terminate *rholo* without losing any computed information, use the *kill(1)* command to send an interrupt or terminate signal to the *rholo* process listed by *ps(1)*. If the system goes down or something dire happens to *rholo,* it may be necessary to restore read/write permission on the holodeck using *chmod(1)*. Do not do this, however, unless you are absolutely sure that *rholo* is no longer running on the holodeck. (See the *ps* man page on how to check for running processes. The file modification date as reported by *ls(1)* is another clue.)

To view the holodeck without invoking a new ray calculation, leave off the *−n* option. To compute the holodeck with multiple processes on a multiprocessing system, use a higher number for the *−n* option. (Don't use more processes than you have processors, though, because you'll only slow things down.)

To allow interactive control of *rholo* from another process, the following invocation will override the file size limit and permit the holodeck to be clobbered by a command entered on the standard input:

rholo −n 1 −o x11 −i −f elect.hdk + DISK=0

To create an empty holodeck from settings on the command line:

rholo new.hdk - RIF=sample.rif "section=2 2 4 8 0 0 0 10 0 0 0 3"

NOTES

Each time rays are added to a beam, that beam's position in the holodeck file is released and a new position is found. After substantial computation on a holodeck, especially over several runs, the holodeck file may become fragmented, leaving holes that take up space without contributing useful information. The percentage fragmentation is reported when the REPORT variable is set and some calculation has taken place. When this percentage gets high on a large holodeck (above 15% or so), it is a good idea to run the *rhoptimize(1)* program once batch rendering is complete to close the gaps and collect beams into groups for quicker rendering access. Rholo will print periodic warnings when the fragmentation exceeds 20%.

AUTHOR

Greg Ward Larson

ACKNOWLEDGMENT

This work was supported by Silicon Graphics, Inc.

BUGS

Global participating media are not handled correctly, though local participating media will usually work.

SEE ALSO

chmod(1), ls(1), ps(1), rad(1), ranimate(1), rhcopy(1), rhinfo(1), rhoptimize(1), rhpict(1), rpict(1), rtrace(1), rvu(1)