NAME

pinterp - interpolate/extrapolate view from pictures

SYNOPSIS

pinterp [view options][**−t threshold**][**−z zout**][**−f** *type*][**−B**][**−alq**][**−e exposure**][**−n**] **pictfile zspec ..**

DESCRIPTION

Pinterp interpolates or extrapolates a new view from one or more RADIANCE pictures and sends the result to the standard output. The input picture files must contain correct view specifications, as maintained by *rpict(1), rvu(1), pfilt(1)* and *pinterp.* Specifically, *pinterp* will not work on pictures processed by *pcompos(1)* or *pcomb(1).* Each input file must be accompanied by a z specification, which gives the distance to each pixel in the image. If *zspec* is an existing file, it is assumed to contain a short floating point number for each pixel, written in scanline order. This file is usually generated by the *−z* option of *rpict(1). Pinterp* also accepts an encoded depth buffer, as produced by *rtpict(1)* or *rcode_depth(1).* If *zspec* is a positive number rather than a file, it will be used as a constant value for the corresponding image. This may be useful for certain transformations on "flat" images or when the viewpoint remains constant.

The *−n* option specifies that input and output z distances are along the view direction, rather than absolute distances to intersection points. This option is usually appropriate with a constant z specification, and should not be used with *rpict(1)* z files.

The *−z* option writes out interpolated z values to the specified file. Normally, this information is thrown away.

Pinterp rearranges the pixels from the input pictures to produce a reasonable estimate of the desired view. Pixels that map within the *−t* threshold of each other (.02 times the z distance by default) are considered coincident. With the *−a* option, image points that coincide will be averaged together, giving a smooth result. The *−q* option turns averaging off, which means that the first mapped pixel for a given point will be used. This makes the program run faster and take less memory, but at the expense of image quality. By default, two or more pictures are averaged together, and a single picture is treated with the faster algorithm. This may be undesirable when a quick result is desired from multiple input pictures in the first case, or a single picture is being reduced in size (anti-aliased) in the second case.

Portions which were hidden or missing in the input pictures must be "filled in" somehow, and a number of methods are provided by the *−f* option. The default value for this option is *−fa,* which results in both foreground and background filling. The foreground fill algorithm spreads each input pixel to cover all output pixels within a parallelogram corresponding to that pixel's projection in the new view. Without it, each input pixel contributes to at most one output pixel. The background algorithm fills in those areas in the final picture that have not been filled with foreground pixels. It does this by looking at the boundary surrounding each blank area and picking the farthest pixels to each side, assuming that this will make a suitable background. The *−ff* option tells the program to use only the foreground fill, the *−fb* option says use only background fill, and the *−f0* option says not to use either fill algorithm.

Even when both fill algorithms are used, there may still be some unfilled pixels. By default, these pixels are painted black and assigned a z distance of zero. The *−fc* option can be used to change the color used for unfilled pixels, and the *−fz* option can be used to set the z distance (always along the view direction). Alternatively, the *−fr* option can be used to compute these pixels using *rtrace(1).* The argument to this option is a quoted string containing arguments for *rtrace.* It must contain the octree used to generate the input pictures, along with any other options necessary to match the calculation used for the input pictures. The *−fs* option can be used to place a limit on the distance (in pixels) over which the background fill algorithm is used. The default value for this option is 0, which is interpreted as no limit. A value of 1 is equivalent to turning background fill off. When combined with the *−fr* option, this is roughly equivalent to the *−ps* option of *rpict(1).*

In order of increasing quality and cost, one can use the *−fa* option alone, or the *−fr* option paired with *−fs* or *−ff* or *−f0.* The last combination will result in the recalculation of all pixels not adequately accounted for in the input pictures, with an associated computational expense. It is rare that the *−fs* option results in appreciable image degradation, so it is usually the second combination that is used when the background fill algorithm results in objectionable artifacts.

The −*B* option may be used to average multiple views read from the standard input into a single, blurred output picture. This is similar to running *pinterp* multiple times and averaging the output together with a program like *pcomb(1).* This option is useful for simulating motion blur and depth of field. (See *pmdblur(1).)* The input views are reported in the information header of the output file, along with the averaged view. The picture dimensions computed from the first view will be the ones used, regardless whether or not the subsequent views agree. (The reported pixel aspect ratio in the output is determined from these original dimensions and the averaged view.) Note that the expense of the *−fr* option is proportional to the number of views computed, and the −*z* output file will be the z-buffer of the last view interpolated rather than an averaged distance map.

In general, *pinterp* performs well when the output view is flanked by two nearby input views, such as might occur in a walk-through animation sequence. The algorithms start to break down when there is a large difference between the view desired and the view(s) provided. Specifically, obscured objects may appear to have holes in them and large areas at the image borders may not be filled by the foreground or background algorithms. Also, specular reflections and highlights will not be interpolated very well, since their view-dependent appearance will be incompletely compensated for by the program. (The −*a* option offers some benefit in this area.)

The −*e* option may be used to adjust the output image exposure, with the same specification given as for *pfilt*. The actual adjustment will be rounded to the nearest integer f-stop if the −*q* option is in effect (or there is only a single input picture).

EXAMPLE

To interpolate two frames of a walk-through animation, anti-alias to 512x400 and increase the exposure by 2.5 f-stops:

```
pinterp −vf 27.vf −a −x 512 −y 400 −e +2.5 30.hdr 30.z 20.hdr 20.z > 27.hdr
```

To extrapolate a second eyepoint for a stereo pair and recalculate background regions:

```
pinterp −vf right.vf −ff −fr "−av .1 .1 .1 scene.oct" left.hdr left.z > right.hdr
```

To convert an angular fisheye to a hemispherical fisheye:

```
pinterp −vf fish.hdr −vth -ff fish.hdr 1 > hemi.hdr
```

AUTHOR

Greg Ward

SEE ALSO

getinfo(1), pdfblur(1), pfilt(1), pmblur(1), pmdblur(1), rpict(1), ranimate(1), rcode_depth(1), rtpict(1), rtrace(1), rvu(1)