



## Anisotropic

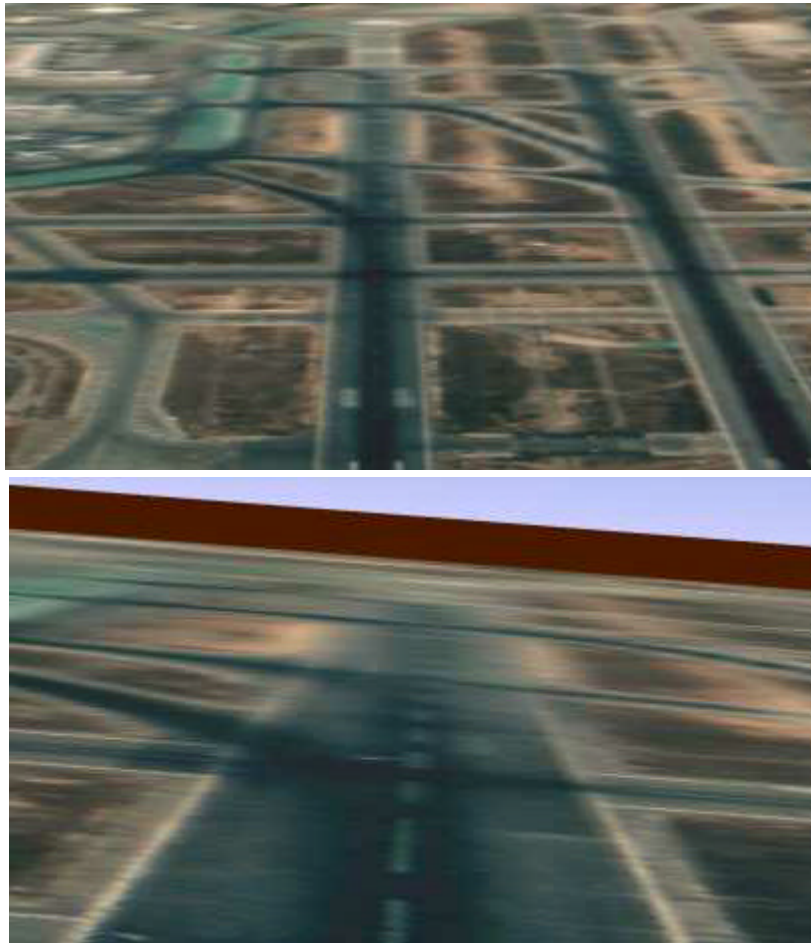
MIP map filters tend to trade image information for anti-aliasing due to the isotropic filter, this demo shows how to improve texturing to avoid losing information.



---

The use of MIP map filtering improves image quality by eliminating texture sample aliasing. It does this by storing prefiltered versions of the texture images with reduced resolutions for s & t dimensions, the image samples in the smaller textures represent the colour sampled over a region of the higher resolution original image. When a texture is applied to the image the resolution of texture used from memory is determined by the scale of the texture information w.r.t. the display resolution. Unfortunately the image sample generated always represents a region of the original high resolution image which is scaled symmetrically in s and t dimensions, however the s & t axes are not mapped symmetrically in the framebuffer, the mapping of the projected texture in the frame buffer is said to be anisotropic. With MIP mapping you must reduce the resolution of s & t symmetrically due to the texture foreshortening in a particular direction, however, typically at right angles to this axis you have no need to reduce resolution and you therefore loose information which might usefully represent information in the texture map, this causes the texture to appear blurred on the polygon. The greater the ratio of longest derivative/ shortest derivative the greater the apparent blurring effect will be. This can be seen in the images bellow where two views of a runway texture mapped to a ground plane exhibit different degrees of information loss. The image on the left blurs a little due to the texture derivative along the viewing vector projected onto the plane being much larger than the derivative at right angles to this. The MIP level is set high (low resolution) as a result, however the derivative at right angles is much shorter and ideally should map higher frequency information. Unfortunately the MIP level must be set by the other derivative to avoid the aliasing which would otherwise result. The image on the right exhibits much worse blurring because the

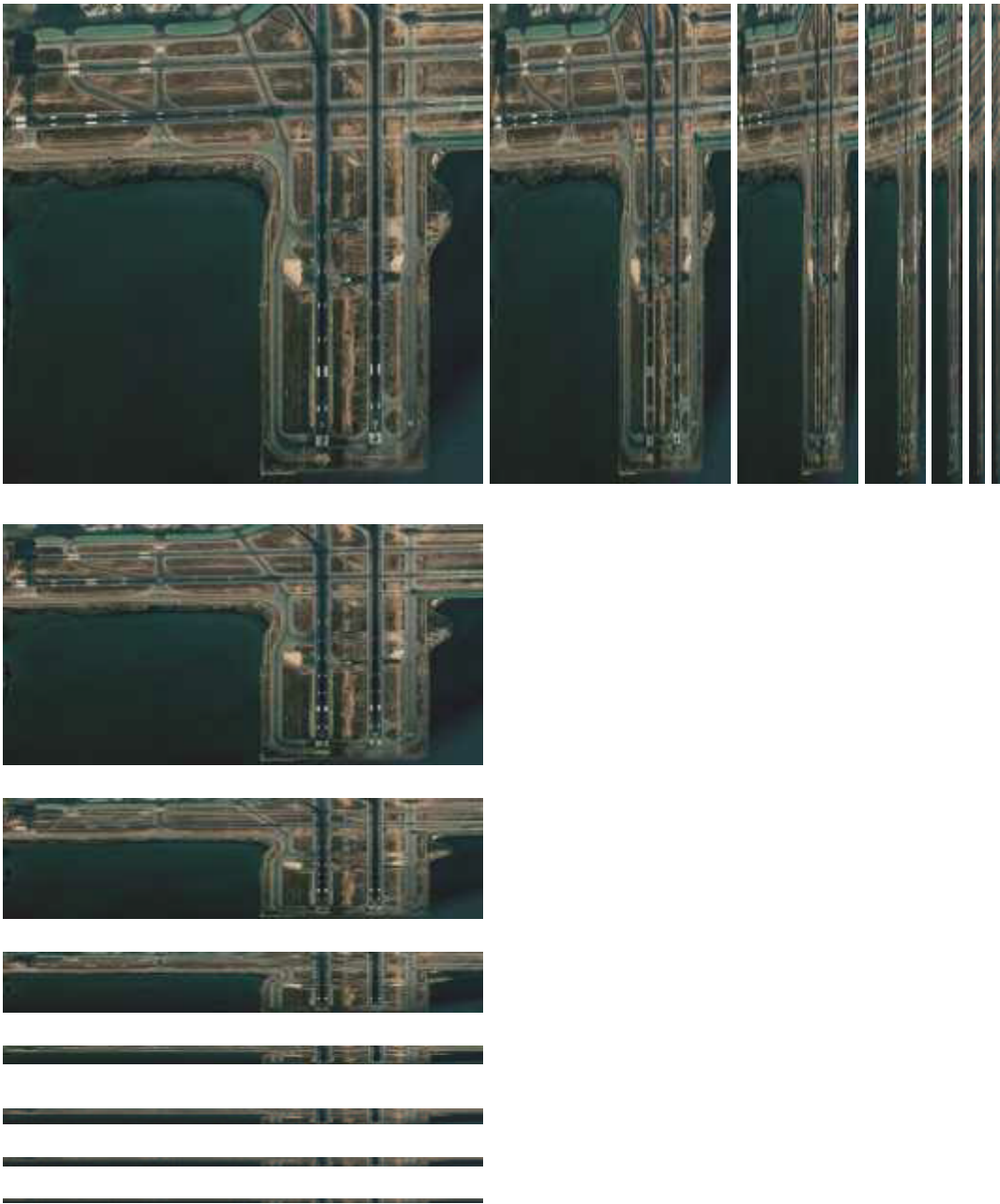
derivative along the viewing vector is even longer forcing an even higher level of MIP while the derivative at right angles is even shorter and really requires much higher frequency (resolution) information.



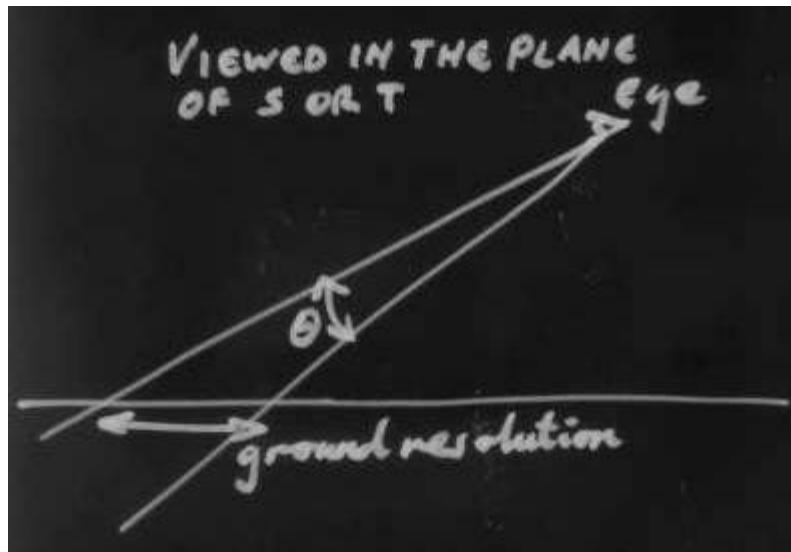
So what can be done, well ideally we want to sample the texture image asymmetrically along the appropriate derivative axes, this is known as anisotropic texture filtering. As you can appreciate there are many different schemes to perform this kind of texture sampling, and you should also realize that the ratio of derivative lengths supported is also critical, ie 2:1 anisotropic filtering would help a lot on the left image above but probably wouldn't improve the right hand image to an acceptable degree.

This example uses a sampling approach which modifies the ratio of texture filtering along the texture  $s$  &  $t$  axes using calculations performed in the application. The effectiveness of the technique relies on the derivative axes discussed above aligning with the  $s$  &  $t$  axes of the texture (at least for best results) but the approach can be extended to support other axes. It is worth noting that without this additional extension the approach could accurately be called RIP mapping. Although the sample code uses textures with prefiltered images which assume the  $t$  derivative is greater than  $s$  this is an arbitrary distinction since our example only needs to exploit this ratio, however with additional texture images the algorithm could support  $s$  derivatives greater than  $t$  with almost no modification.

So the basic idea is we take the image we normally MIP map and precompute versions of that image which are anisotropically filtered along each axis of interest, in this case I have shown both  $s$  &  $t$  axes prefiltering, in the example I actually use a  $t < s$  set of prefiltered of images only:



The next step is for each polygon in the database, select a texture ID from texture memory which best represents the ratio of  $s/t$  closest to those in the prefiltered data set. This is a simple calculation based on the eye position w.r.t. the polygon. My example computes the expected ground displacement resolution on a horizontal plane using some assumptions about the resolution and field of view, this gives the 'pixeltheta' value in the example code which is roughly the angle subtended by a screen pixel. I can use this value to compute an ideal ground resolution for the texture at the location of each polygon. Described another way, I compute the distance that a projected 'pixeltheta' will cover on the ground in the plane of each  $s$  &  $t$  axis and use the ratio of each to select a texture ID with the correctly shaped filter for the example.



```
// for this quad compute desired ground resolutions for
// this pixel angle and eye position
yrange = ((*coords1+i*4)[1] + ((*coords1+i*4+2))[1])* .5f - view.xyz[1];
xrange = ((*coords1+i*4)[0] + ((*coords1+i*4+1))[0])* .5f - view.xyz[0];
yincidence = atanf(view.xyz[2]/yrange);
xincidence = atanf(view.xyz[2]/xrange);
groundyres = (view.xyz[2] / tanf(yincidence-pixel_theta)) - yrange;
groundxres = (view.xyz[2] / tanf(xincidence-pixel_theta)) - xrange;
```

This code uses the ground resolution to select a texture ID, it works on the assumption of a 2:1 zoom at each texture id which would therefore half the assumed ground resolution for a particular axis at each iteration:

```
rip = 0;
while(groundyres > .002f)
{
groundyres *= .5f;
rip++;
}
while(groundxres > .0005f)
{
groundxres *= .5f;
rip--;
}
```

Because it is only ratios we are interested the code just decrements the texture ratio on one axis and increments for the other, the ground resolution is changes slightly for one axis to factor in a perspective foreshortening effect not accounted for by the axis aligned ground resolution computatiuon, this should really be computed accurately.



You can see that this technique gives hugely anisotropic filtering capabilities without recourse to special hardware and for constrained cases solves the problems like texture blurring in situations where MIPmap texture filters aren't good enough.

**Corporate Office**  
2011 N. Shoreline Boulevard  
Mountain View, CA 94043  
(650) 960-1980

U.S. 1(800) 800-7441  
Europe (44) 118-925.75.00  
Asia Pacific (81) 3-54.88.18.11  
Latin America 1(650) 933.46.37

Canada 1(905) 625-4747  
Australia/New Zealand (61) 2.9879.95.00  
SAARC/India (91) 11.621.13.55  
Sub-Saharan Africa (27) 11.884.41.47

Copyright © 1998 Silicon Graphics, Inc. All rights reserved.