Improved Alpha-Tested Magnification for Vector Textures and Special Effects

Chris Green, VALVE

VALVE

A Simple way to include high resolution line art in real time applications

- Fast
- Easy to implement
- Works on all graphics hardware
- Low texture memory usage (128x64 one channel):

VALVE

The problem

- We want sharp crisp edges even when viewed up close, without using tons of texture memory.
- Alpha testing gives us sharp edges, but also produces unpleasant false contours.

VALVE

Alpha blending

- Using alpha blending both gives us blurry edges and introduces unpleasant stair stepping due to false high frequencies created by bilinear reconstruction.

Previous solutions

- Multiple techniques have been published for storing line-art images in textures for reconstruction by the GPU.
- We wanted something that would:
  - Work on all hardware (dx8 / dx7)
  - Fit into existing shaders
  - Not require vector graphics for source art.
  - Be easy to code
  - Be high performance
  - Use a single rgba texture to encode the colored lineart.

VALVE

A cheaper solution

- The solution is to store values in the alpha channel that are not poorly reconstructed by bilinear interpolation

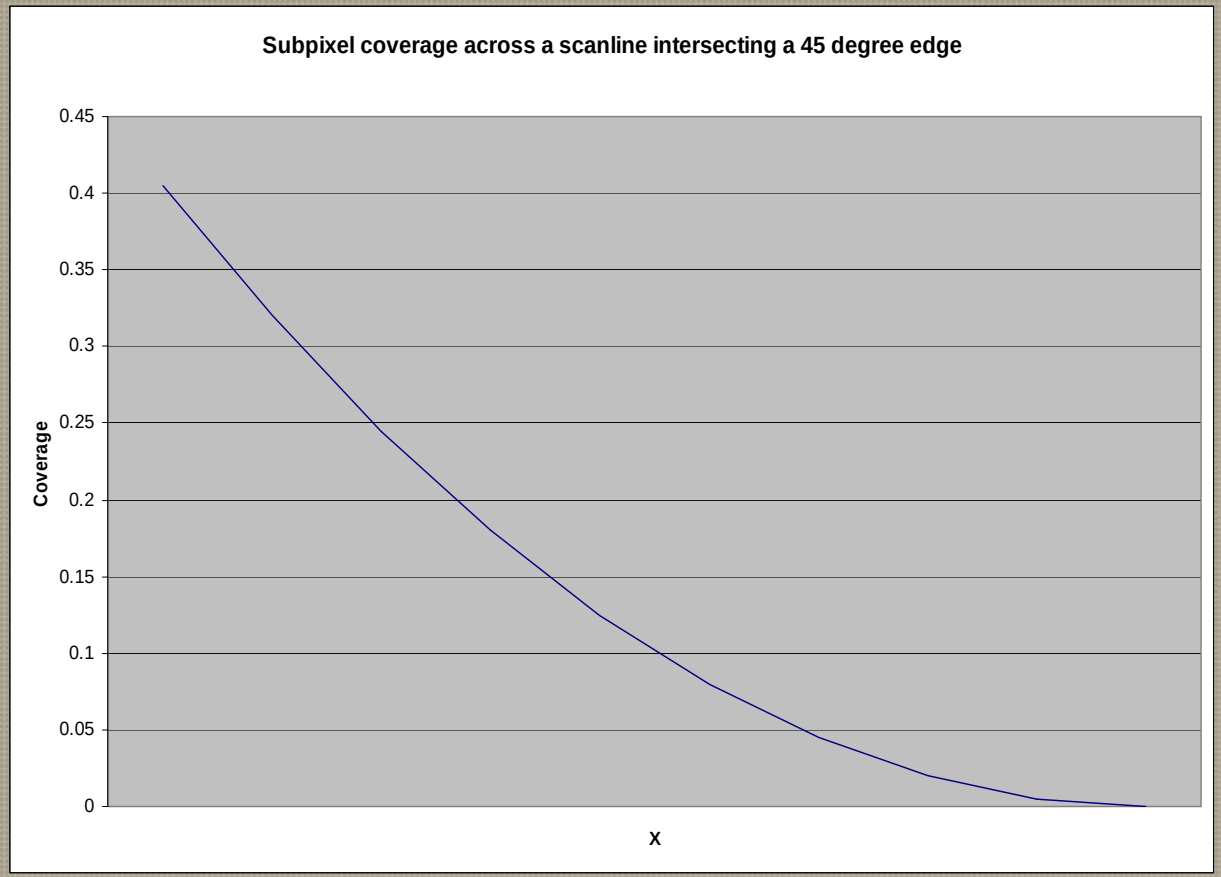
VALVE

Why do we get the false contours and stairsteps?

- Artist-drawn or rendered alpha channels contain “coverage” values.
- Textures are reconstructed by the GPU using bilinear interpolation.
- “Coverage” is not a linear function for interesting cases.
- Works for horizontal and vertical edges.
- Thresholding other edge types using interpolated low-resolution coverage will produce artifacts.

Texel Coverage

- Using alpha blending both gives us blurry edges and introduces unpleasant stair stepping due to false high frequencies created by bilinear reconstruction.



VALVE

So what **is** linear?

- The distance between a point and an infinite line segment is a 2d linear function.
- Distance = (P – P0) dot N.
- If we store the signed distance between the sample location and a line, bilinear filtering will properly reconstruct this distance.
- If we store the signed distance between the sample location and the shape, bilinear filtering will produce a piece-wise linear approximation of the original shape boundary.
- This produces “infinite resolution” linear edges (up to the interpolation precision of the GPU).

VALVE

Processing the source images

- The artist supplies a very high resolution image with a binary alpha channel. 4K x 4K typically.
- Our utility generates a low-resolution image (for example, 64x64), by storing a signed distance between the sample point and the closest pixel in the original image.
- Artist specifies a desired output resolution, and a “distance spread” (for special effects).

VALVE

Processing the source images

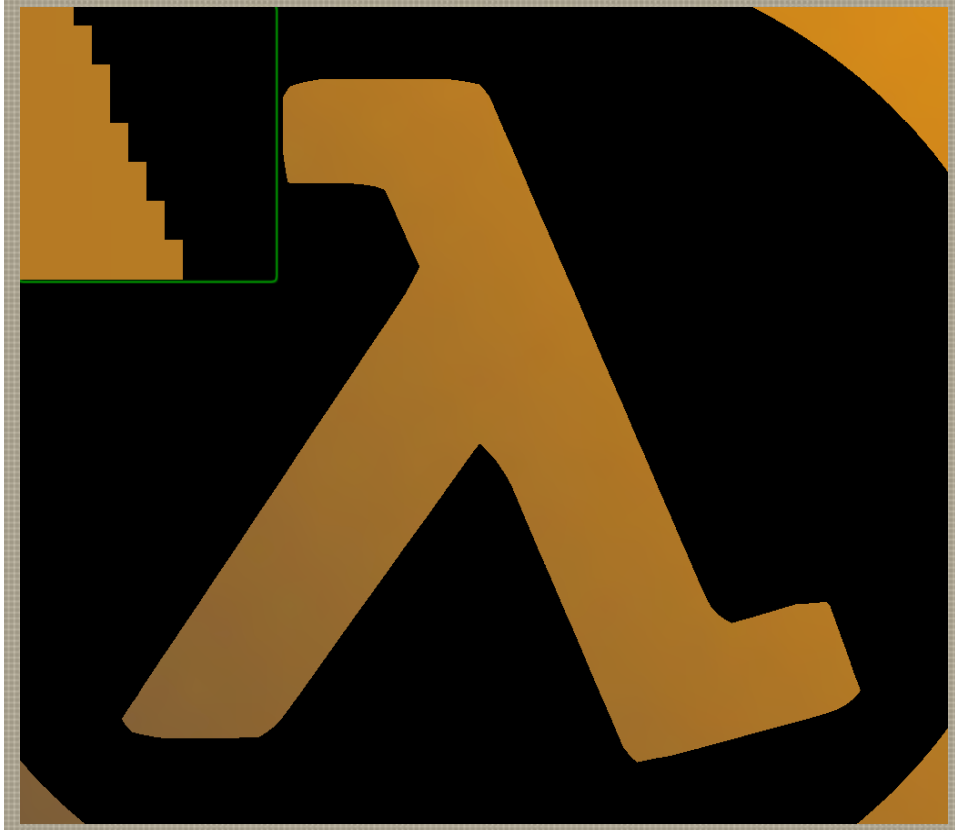
- While there are plenty of ways to accelerate the search for boundary texels, brute force neighborhood search works fine.
- Processing loop is only 40 lines of code.
- DXT compression can be used, but uncompressed usually looks better for a given texture budget.

64x64 Alpha channels

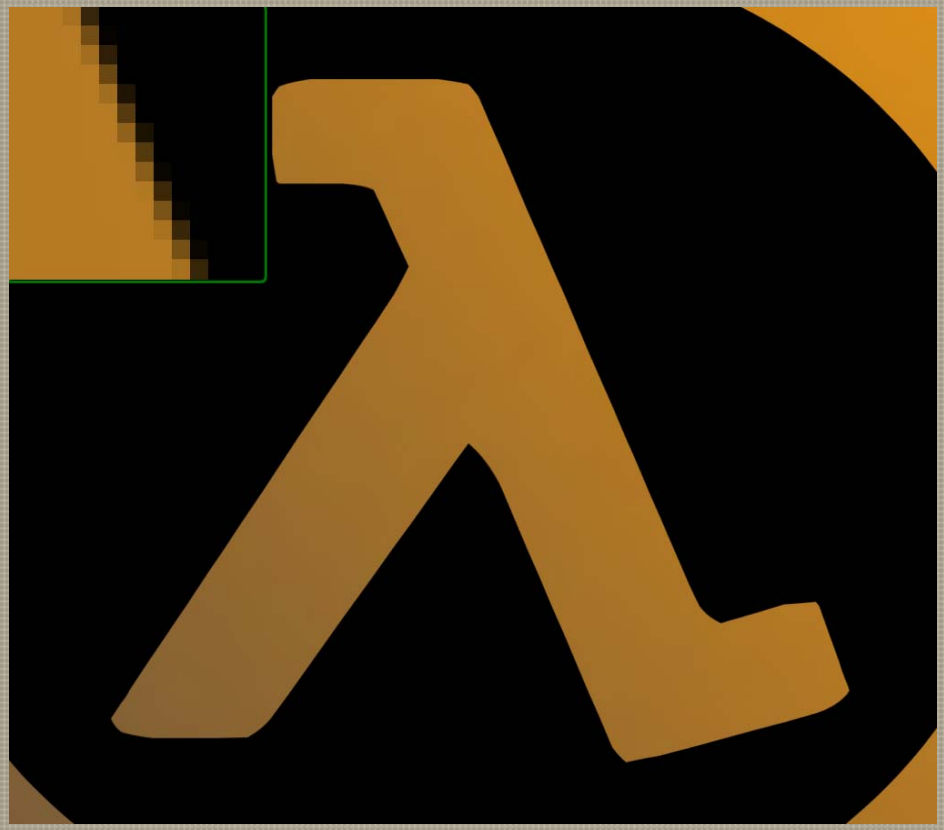
Alpha channel generated by rescaling 4kx4k image

Alpha channel generated by storing signed distance to 4kx4k image

Rendering



Rendered with alpha threshold 0.5, no shader.



Anti-aliased by using derivatives in the shader.

Rendering

Rendered blurry, using a soft threshold

A white outline applied by the pixel shader.

Rendering

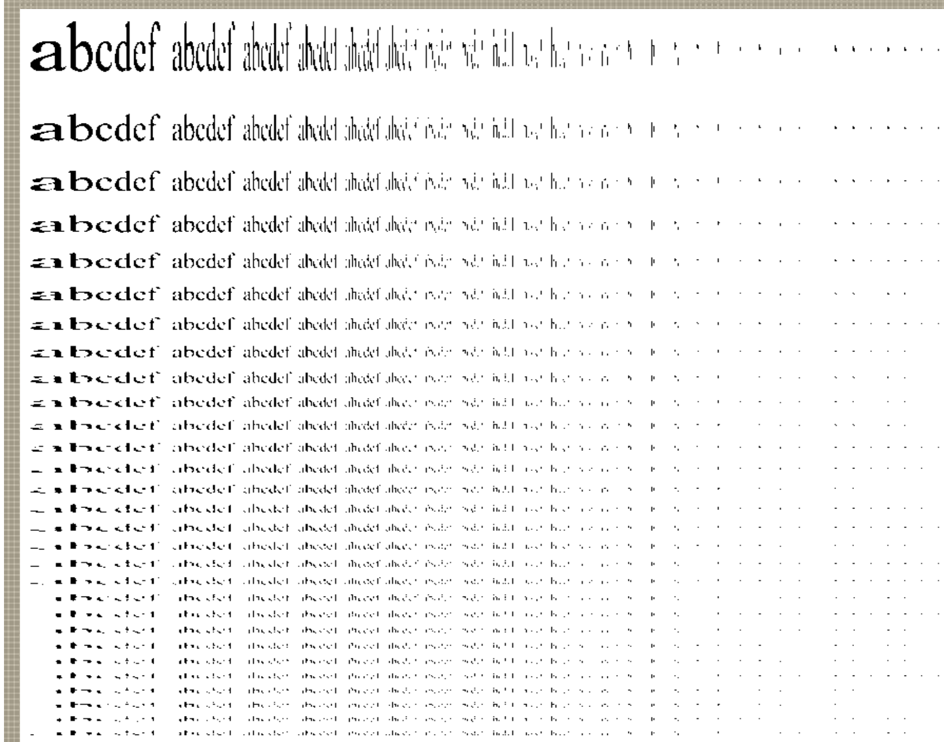
“Outer glow” added by pixel shader

By performing an extra fetch in the shader, we can add a soft drop shadow.

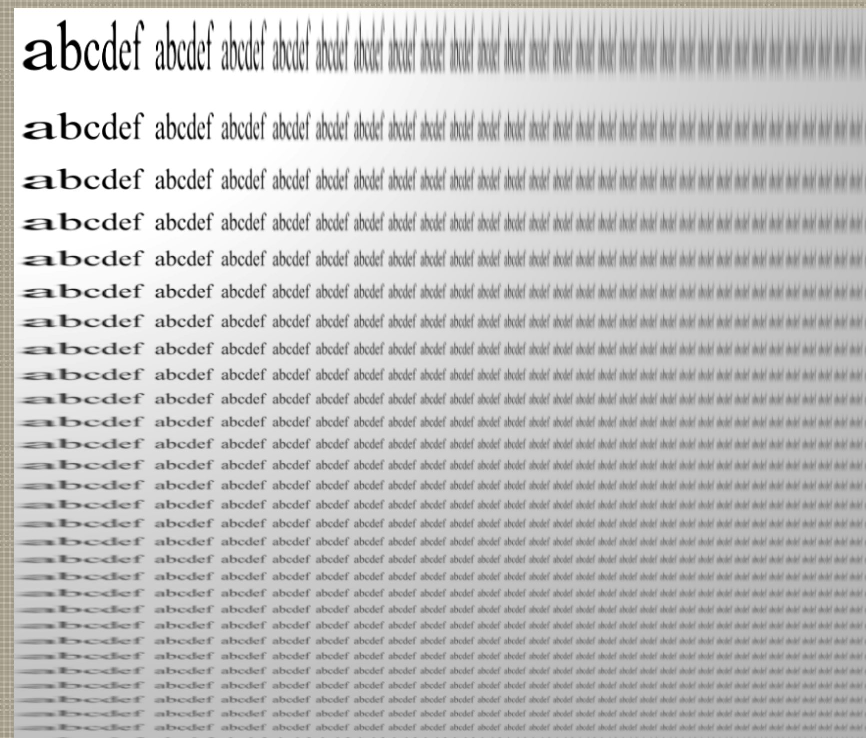
Uses

- Signs
- Text
- HUD/GUI elements

Anti-aliasing minified textures

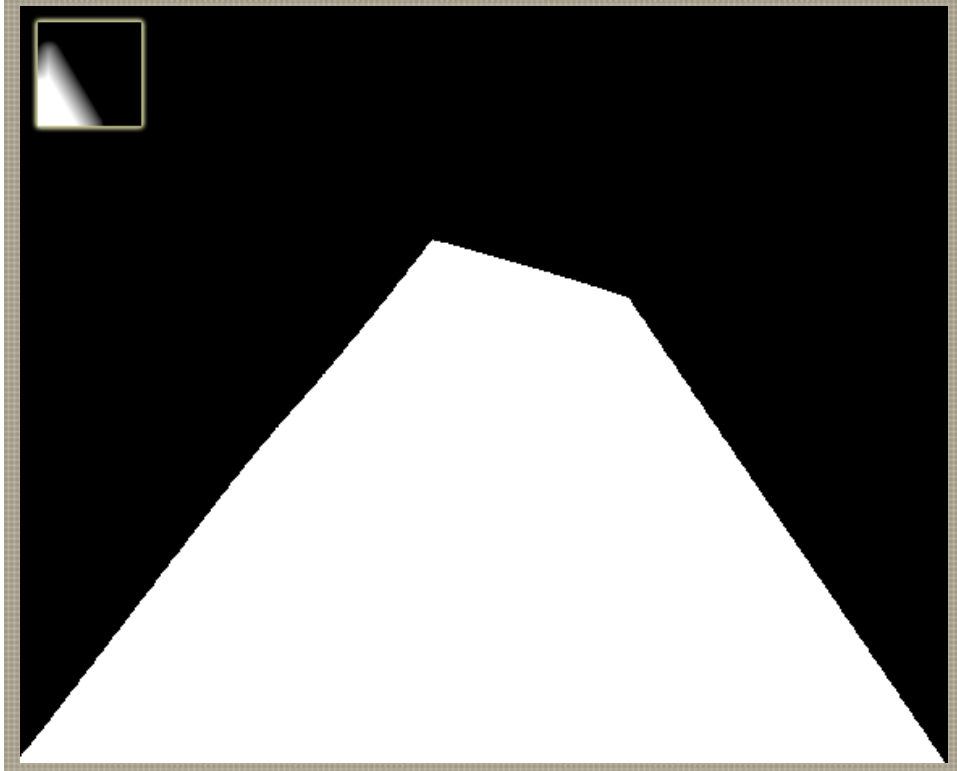


"Infinite" frequency edges cause aliasing, though you can get away with it in a lot of cases.

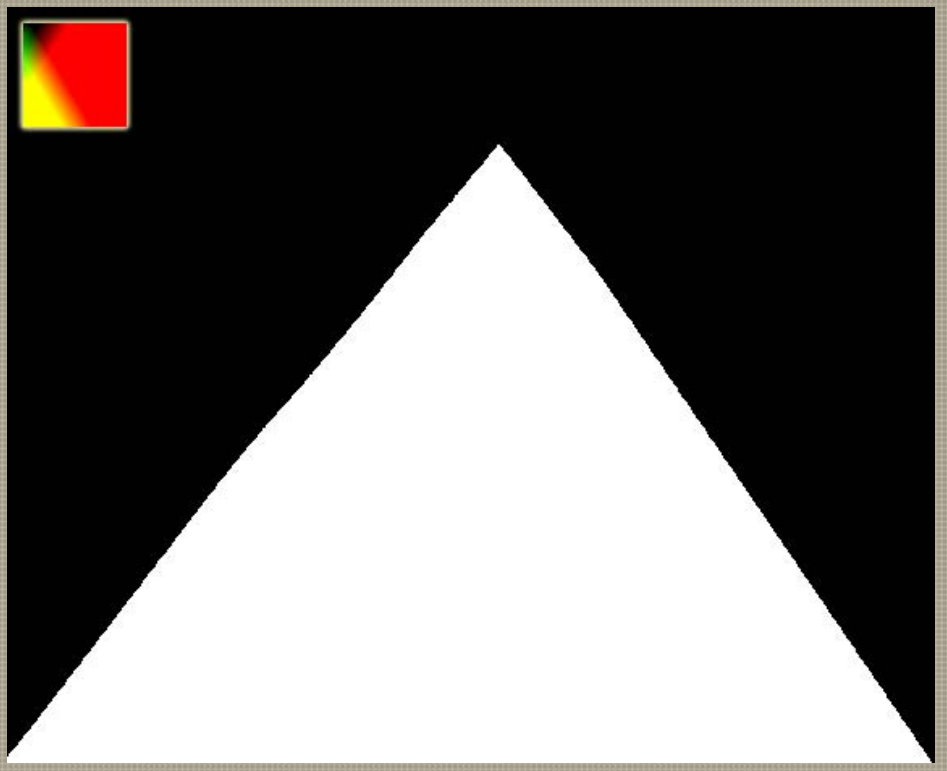


Varying the distance falloff based upon screen-space uv derivative reduces aliasing from minification

Artifacts



Corners will get chopped or rounded off as lower resolutions are used.



Either increase the resolution, or use more than one texture channel to encode more complex topology.

Questions?

cgreen@valvesoftware.com

http://valvesoftware.com/publications.html