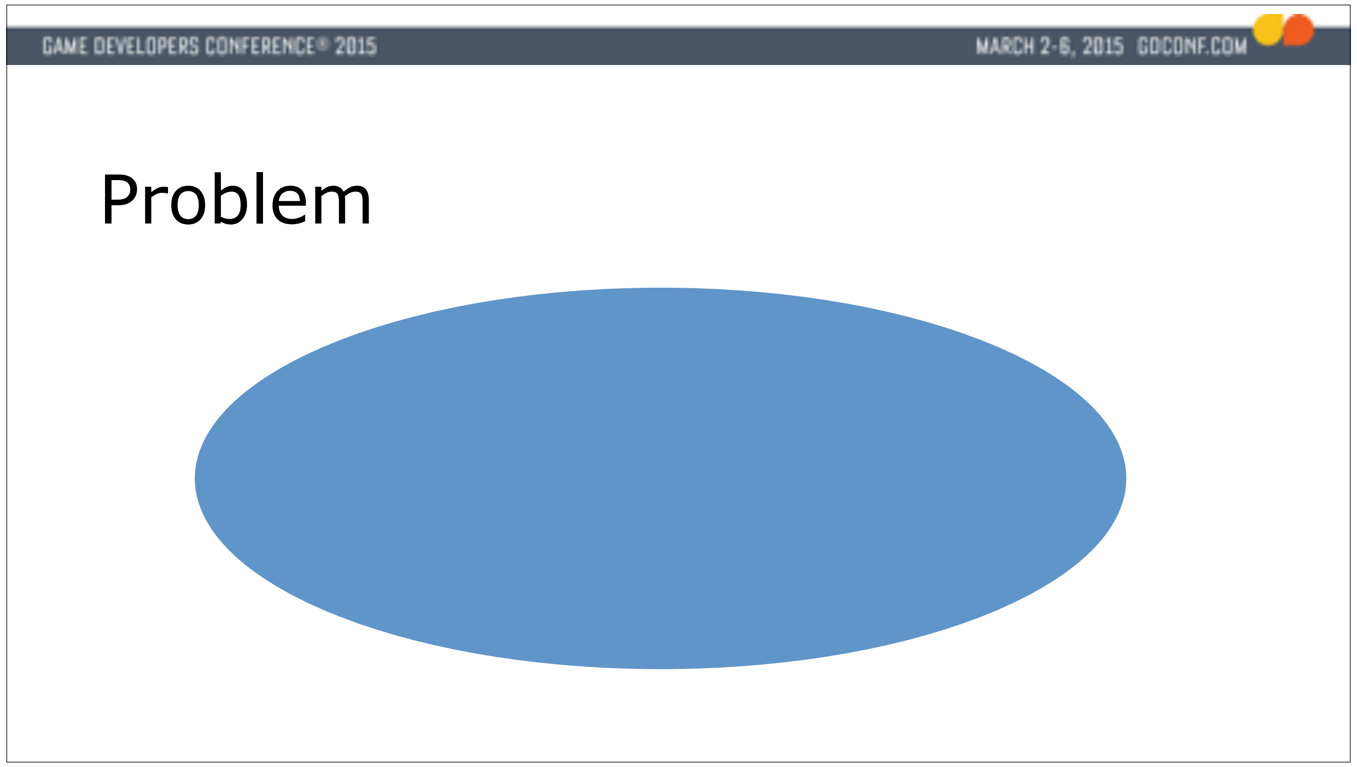
Drawing an Antialiased Ellipse

Jim Van Verth
Software Engineer, Google
skia.org

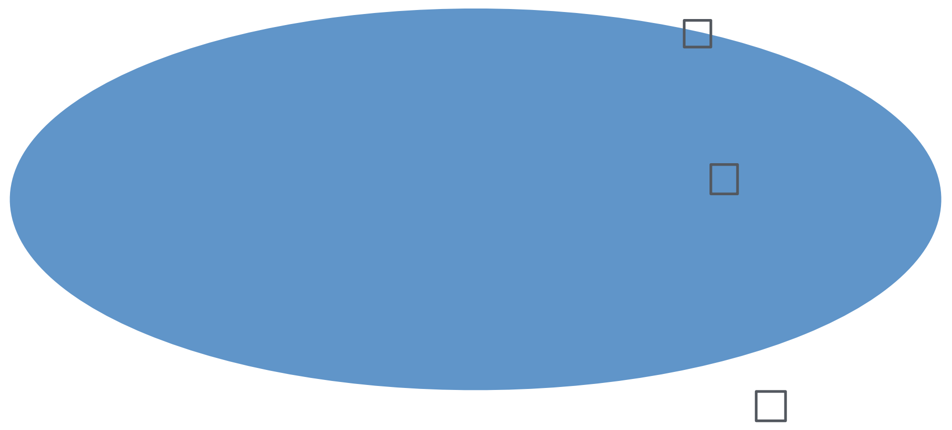
GAME DEVELOPERS CONFERENCE®
MOSCONE CENTER · SAN FRANCISCO, CA
MARCH 2-6, 2015 · EXPO: MARCH 4-6, 2015



Problem

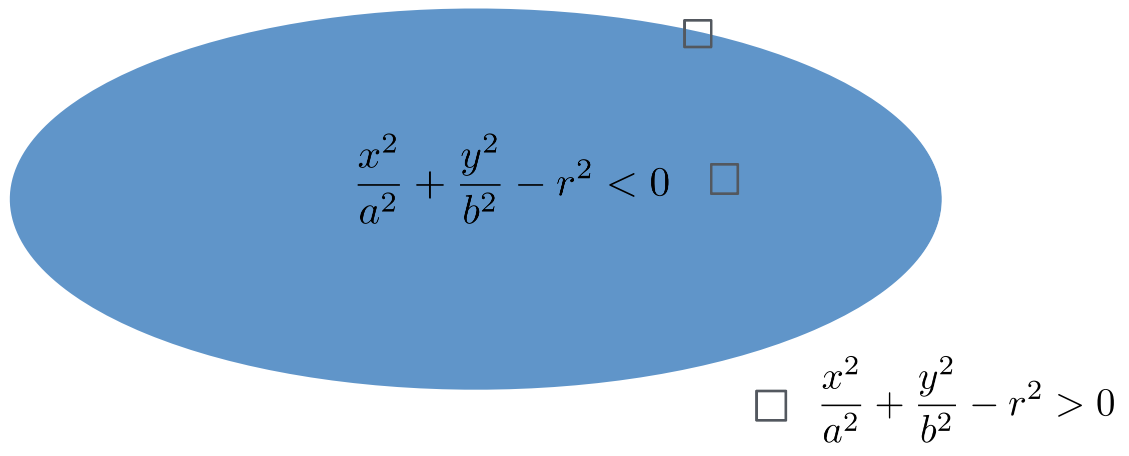
The goal is that we have an ellipse, and want to render a nice smooth edge.

Problem



For proper AA, want to compute coverage of ellipse on each pixel. (We could use MSAA, but let’s assume it’s too slow or poorly supported)

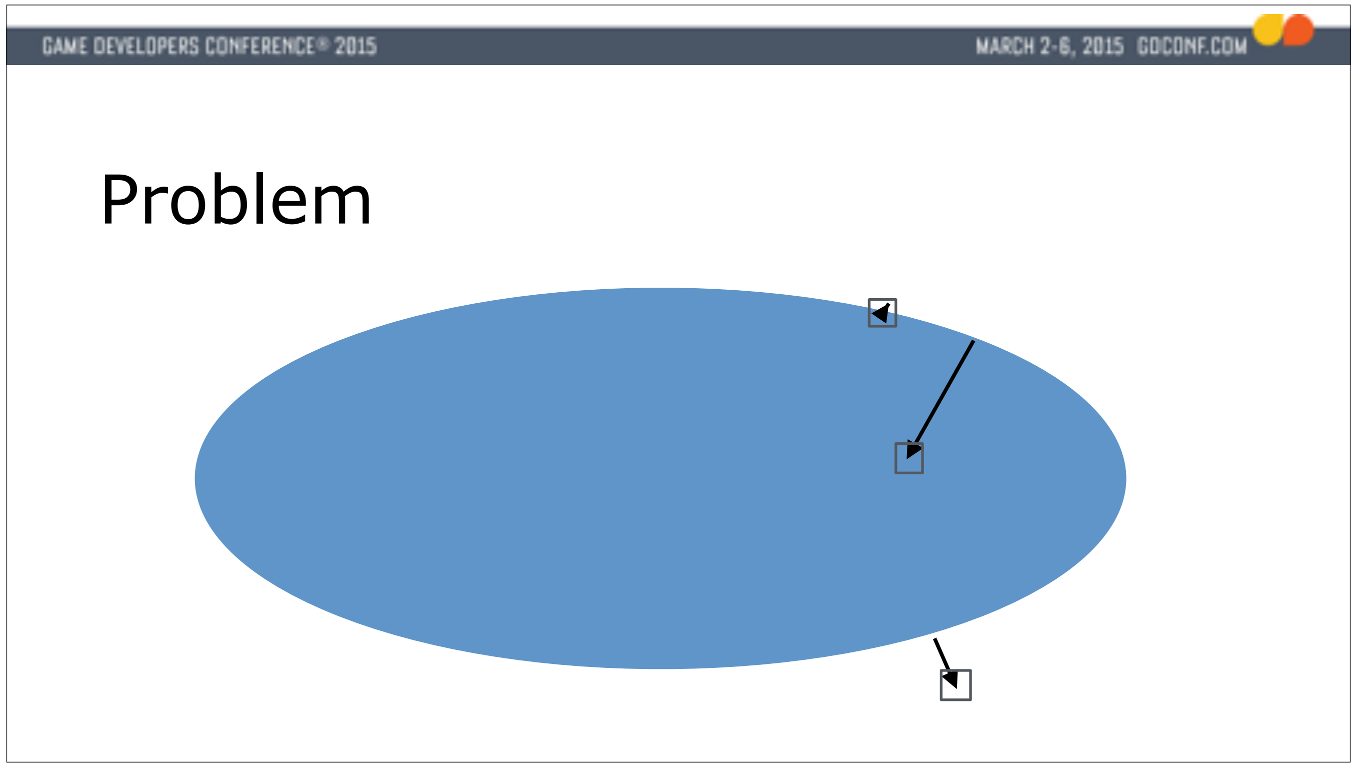
Problem



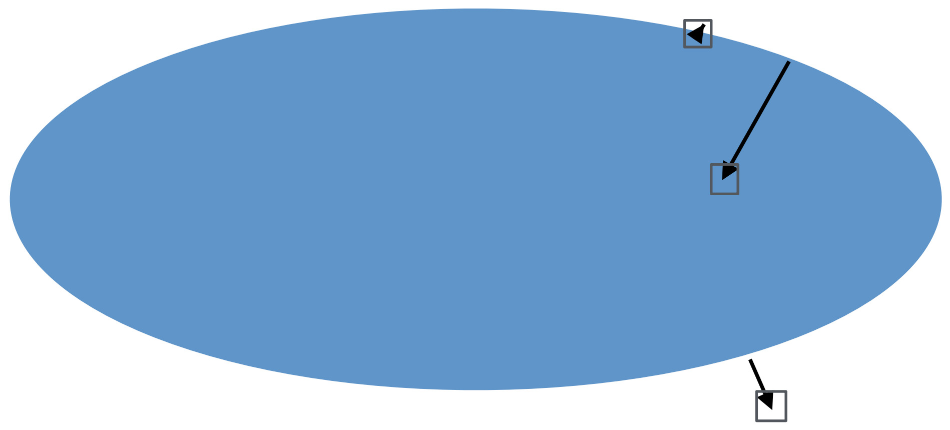
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - r^2 < 0$$

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - r^2 > 0$$

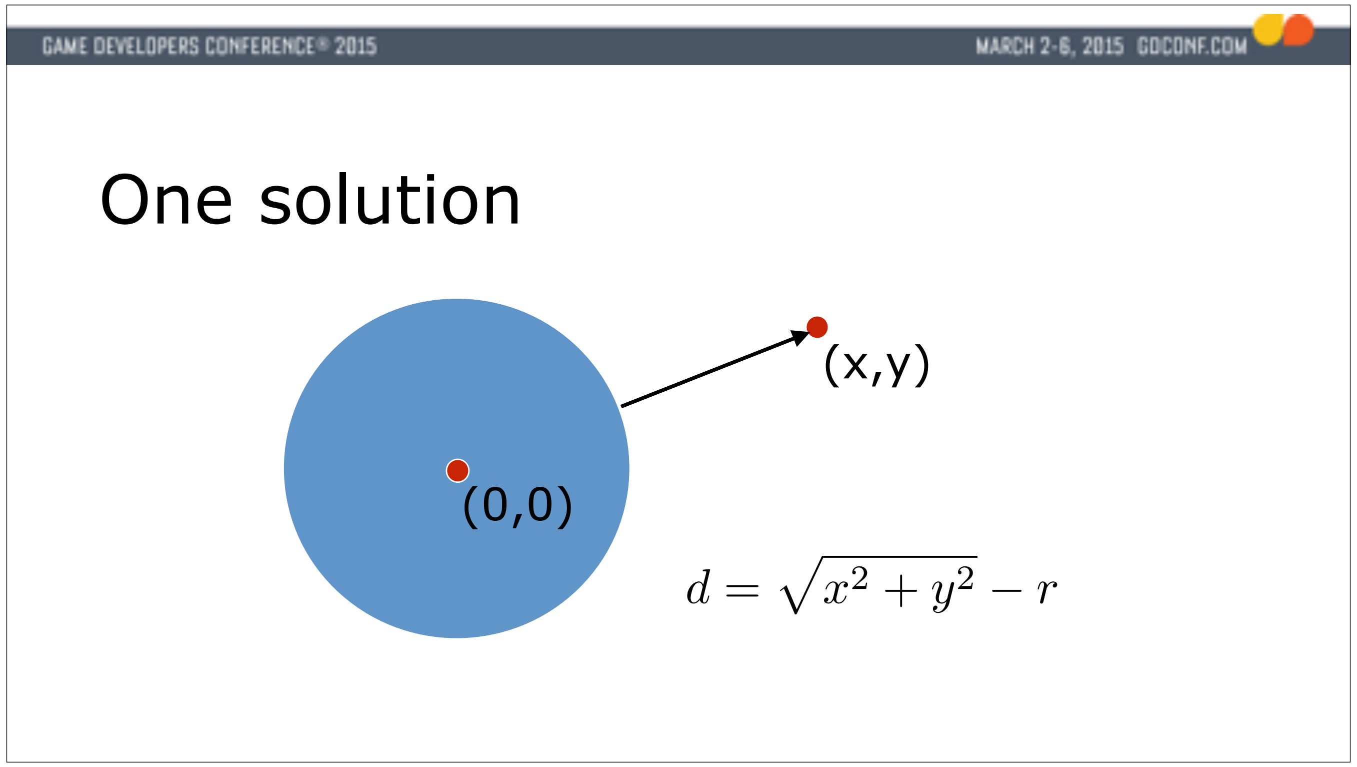
Could use an implicit function. Positive means outside the ellipse, negative inside, 0 on the edge. But that just tells us if the pixel center is in or out, not coverage.



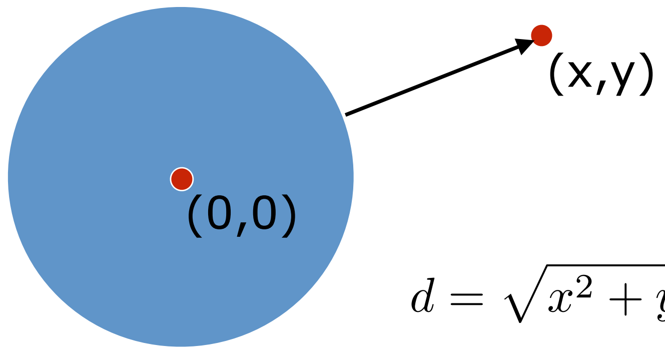
Problem



Can use (1/2 – signed distance) from edge to center of pixel, clamped between 0 and 1, to approximate coverage. Problem — general distance from an ellipse is an intractable problem.

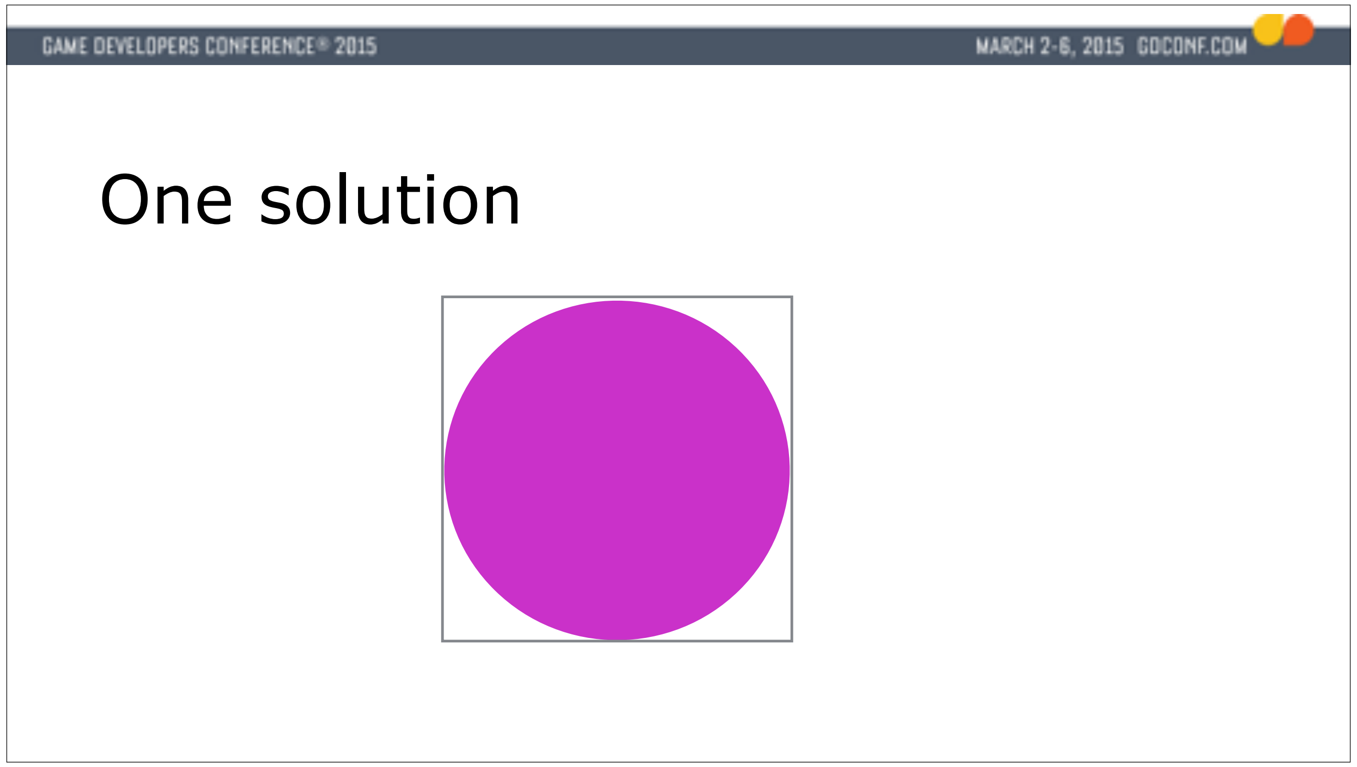


One solution

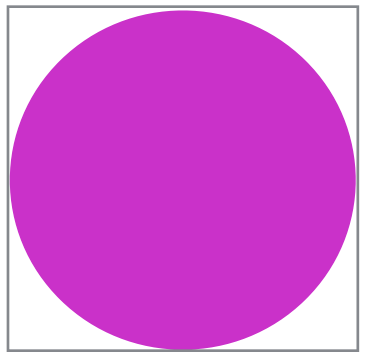


$$d = \sqrt{x^2 + y^2} - r$$

Signed distance to a circle (at origin) is easy to compute

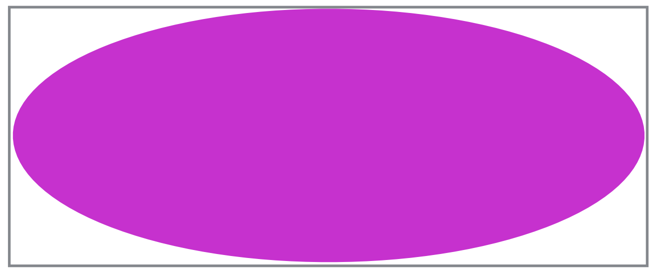


One solution



So write a shader that renders a circle in a quad

One solution

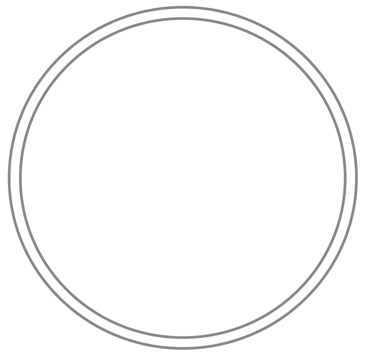


And stretch and squash the quad to get an ellipse. This is effectively the same as rendering a square quad with a non-uniform scale transformation.

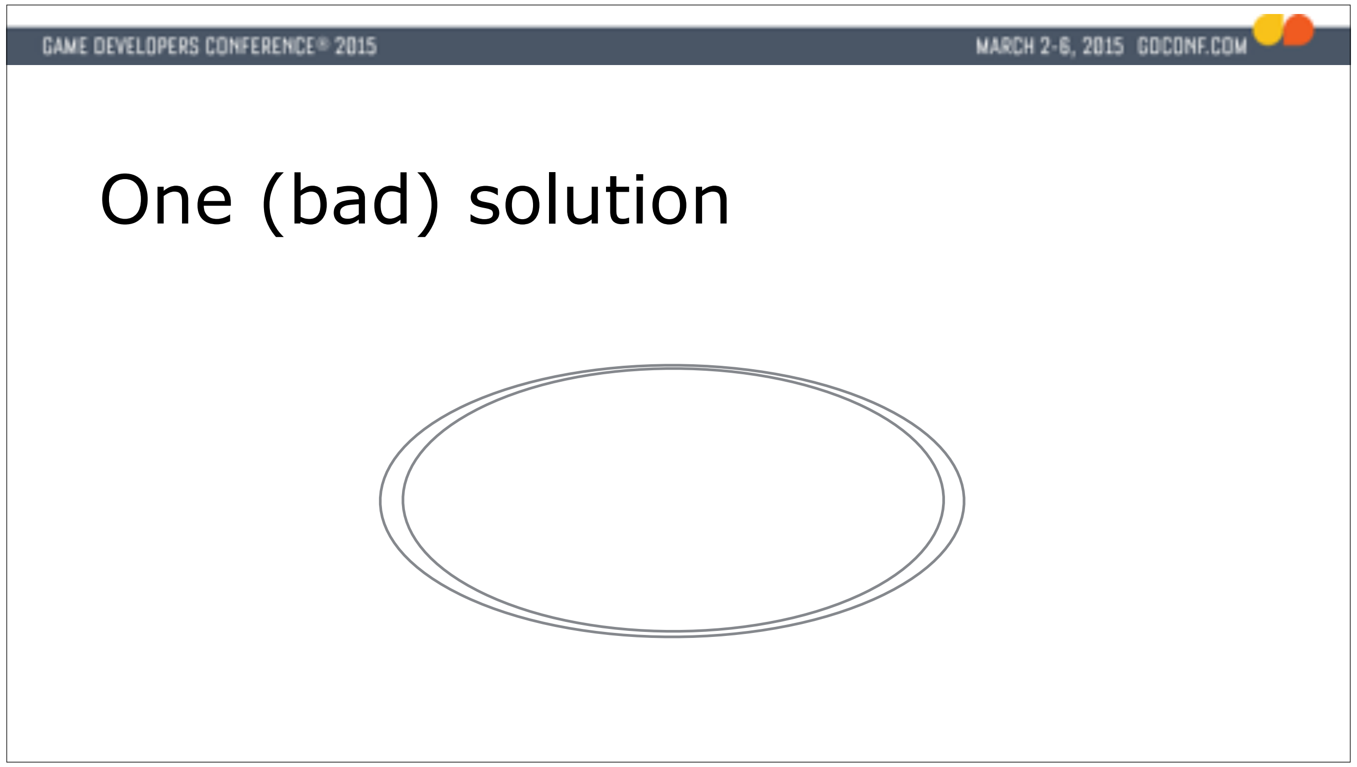
One (bad) solution

The problem is that then you end up with blurred edges in the stretched direction, and overly sharp edges in the shrunken direction. Here I'm using a unit circle, so everything is blurred.

One (bad) solution

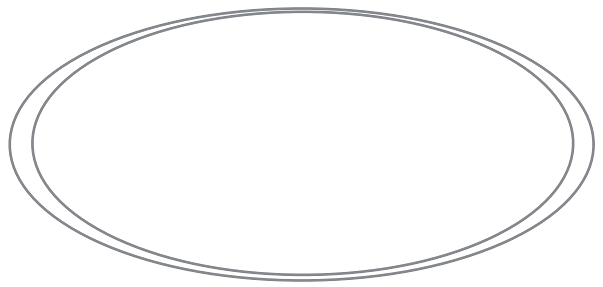


If we take some isodistance curves, we can see what's going on.

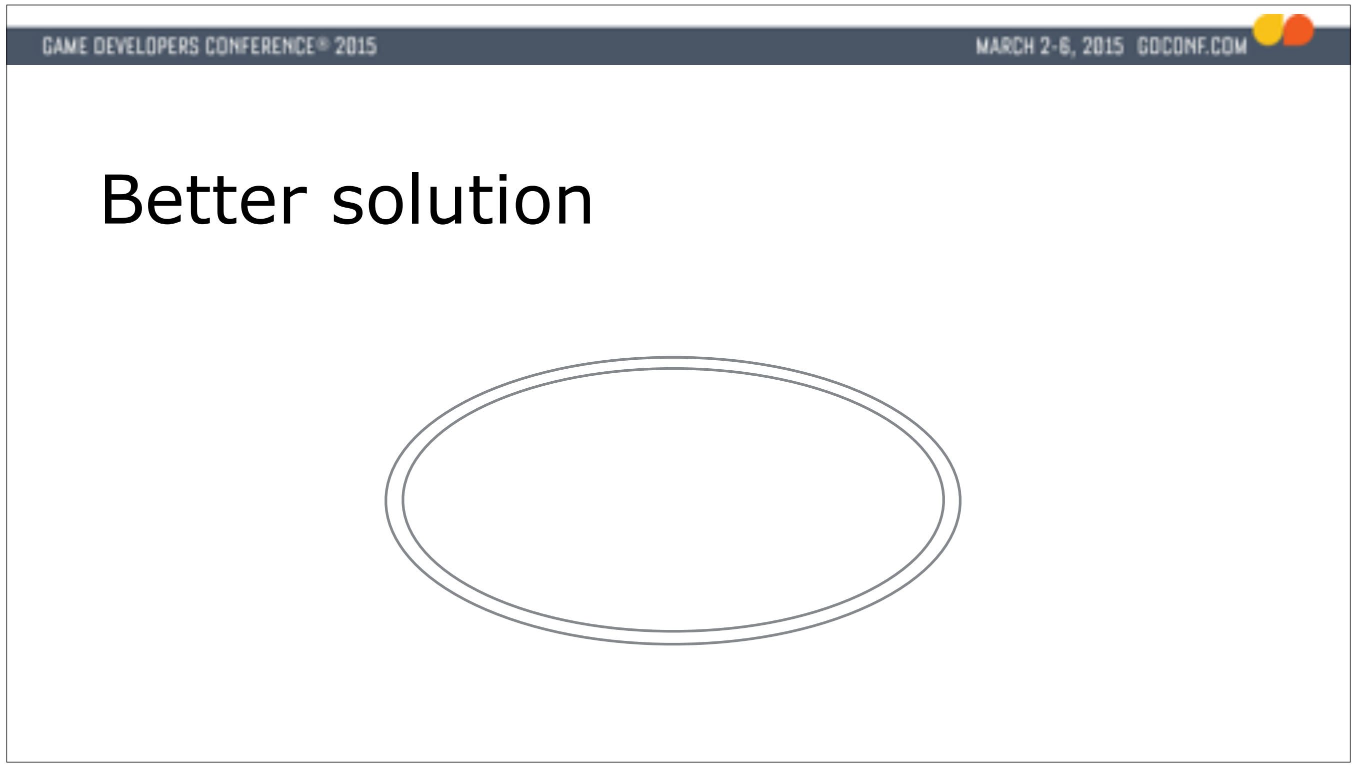


GAME DEVELOPERS CONFERENCE® 2015 MARCH 2-6, 2015 GDCONF.COM

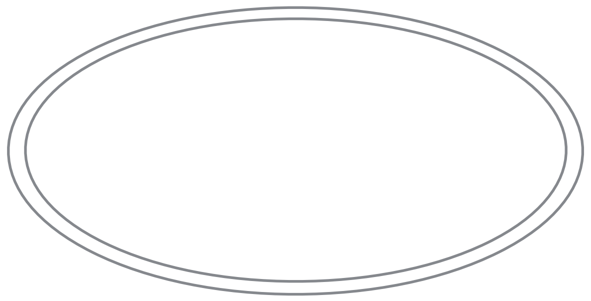
One (bad) solution



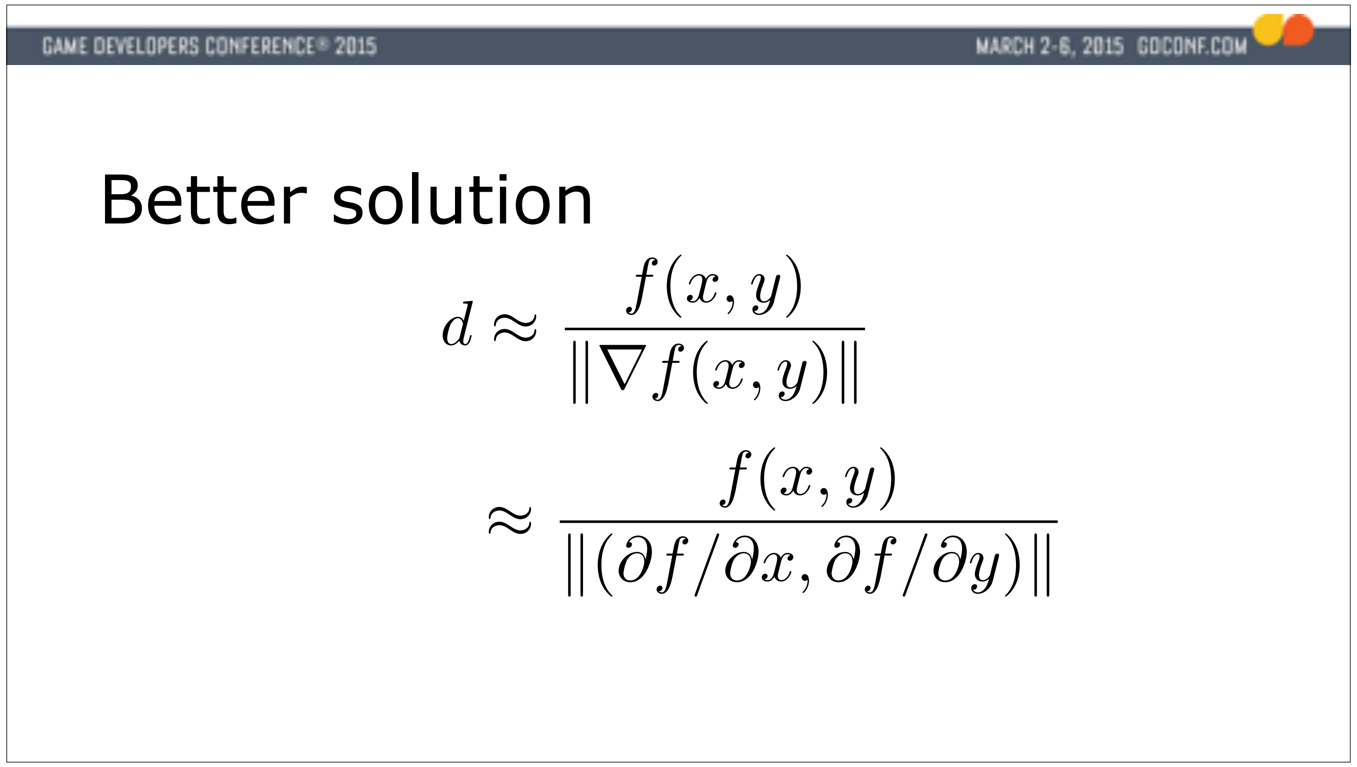
After we squash/stretch our quad, we end up non-uniformly scaling our distance values as well.



Better solution



What we’d like to do is apply the inverse of the transformation to the distance curves, to correct for this. But what would we apply our transformation to? We don’t have a vector to transform.

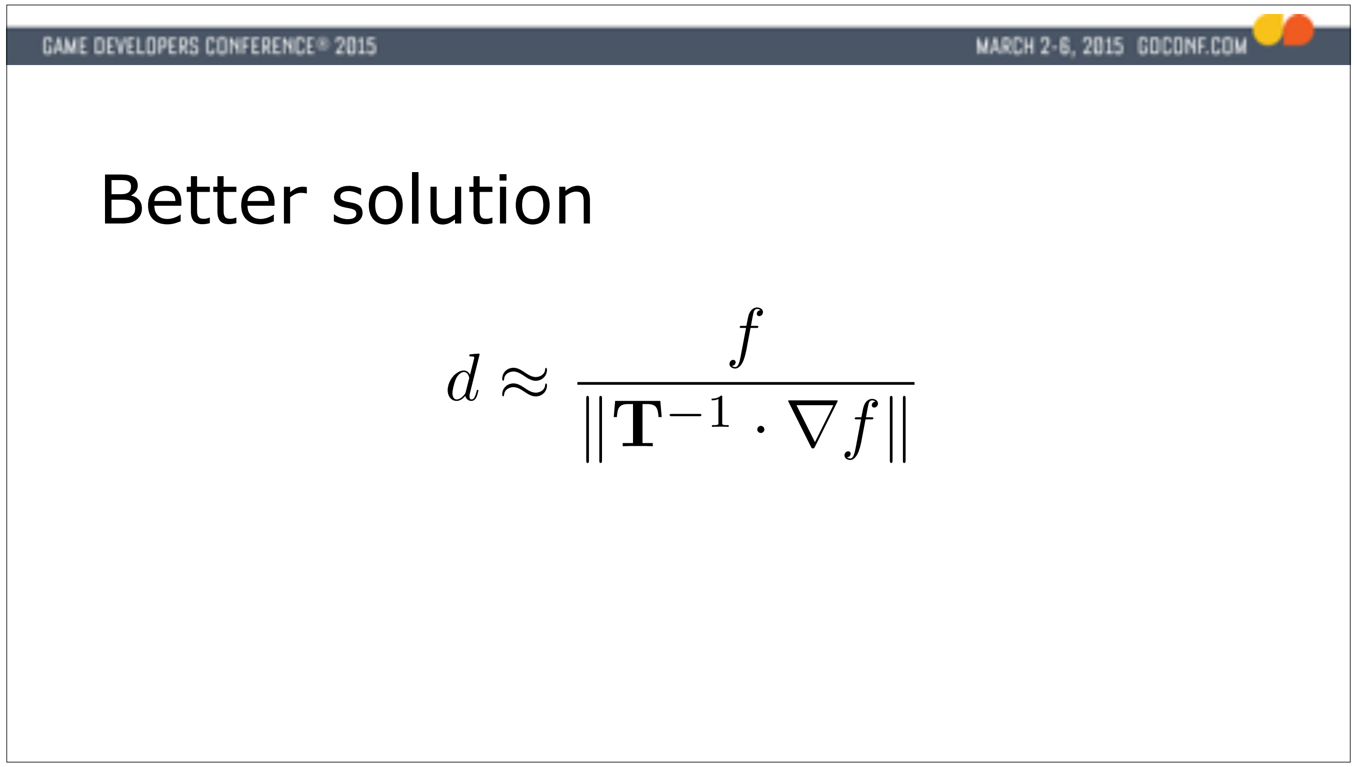


Better solution

$$d \approx \frac{f(x,y)}{\|\nabla f(x,y)\|}$$

$$\approx \frac{f(x,y)}{\|(\partial f/\partial x, \partial f/\partial y)\|}$$

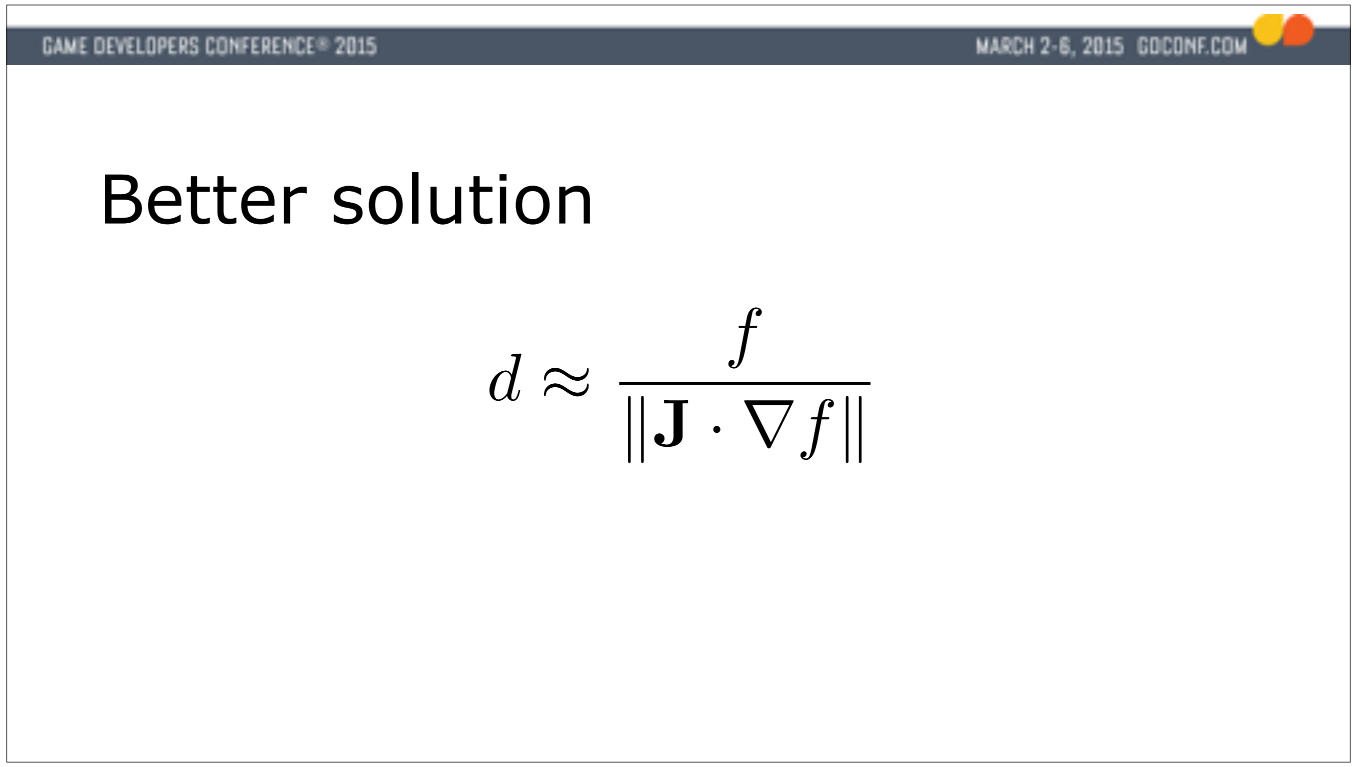
However, there is an approximation to the distance for any implicit function (via Gabriel Taubin), which is to divide the function value by the length of the gradient at that point. This approximation is only good within a distance of about 1 – but that's okay, because that's all we need to measure with some accuracy. Beyond that, we know that a pixel is either all in or all out.



Better solution

$$d \approx \frac{f}{\|\mathbf{T}^{-1} \cdot \nabla f\|}$$

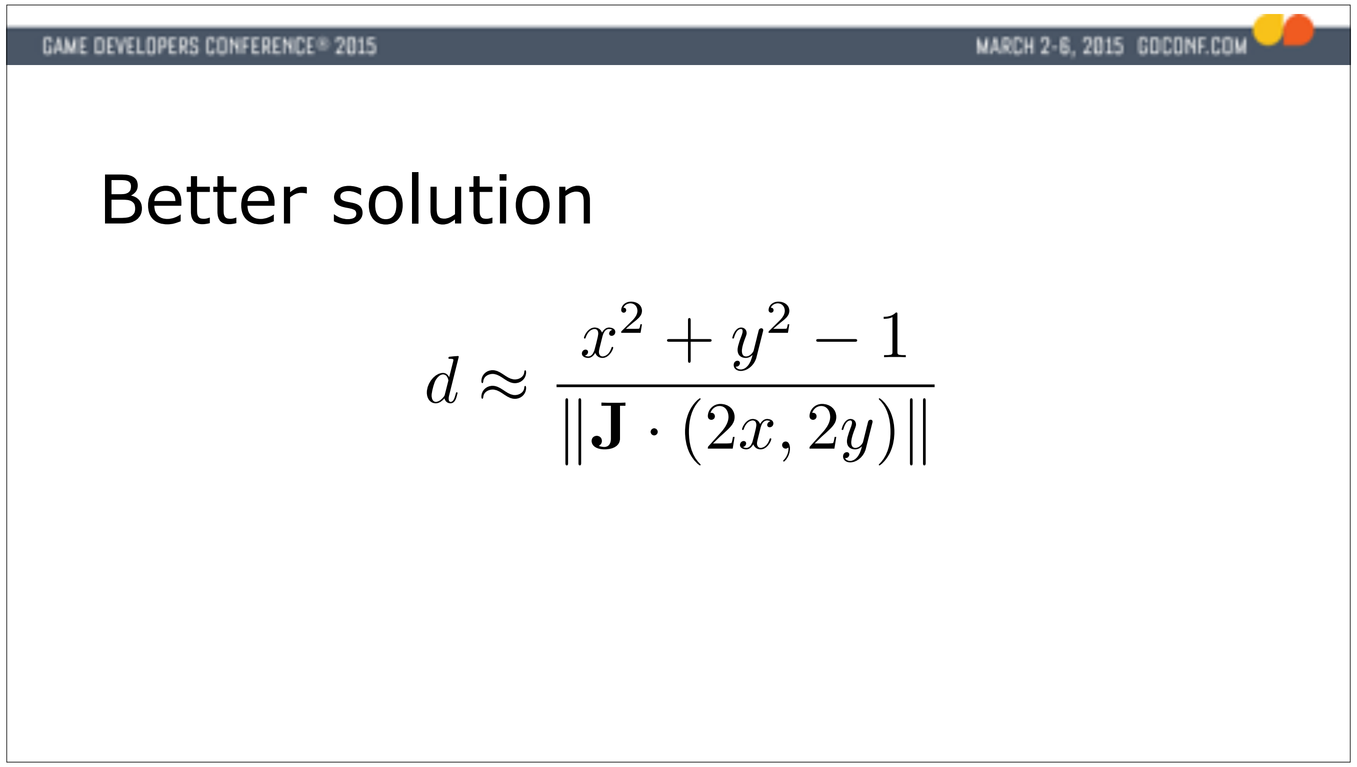
If we have a transformation T, we can transform the gradient by the inverse to get a corrected distance.



Better solution

$$d \approx \frac{f}{\|\mathbf{J} \cdot \nabla f\|}$$

For non-affine transformations, we can’t get a general inverse across the entire quad, so we can compute the Jacobian at each point (using the shader differential operators).

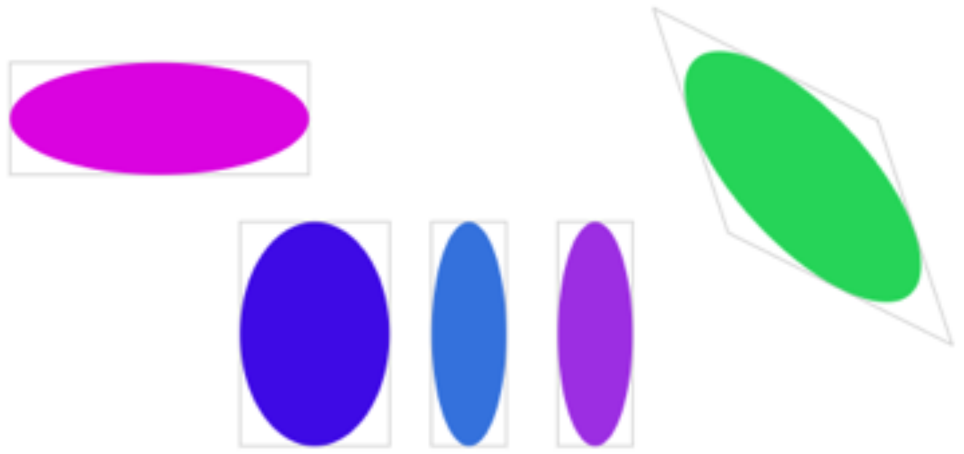


Better solution

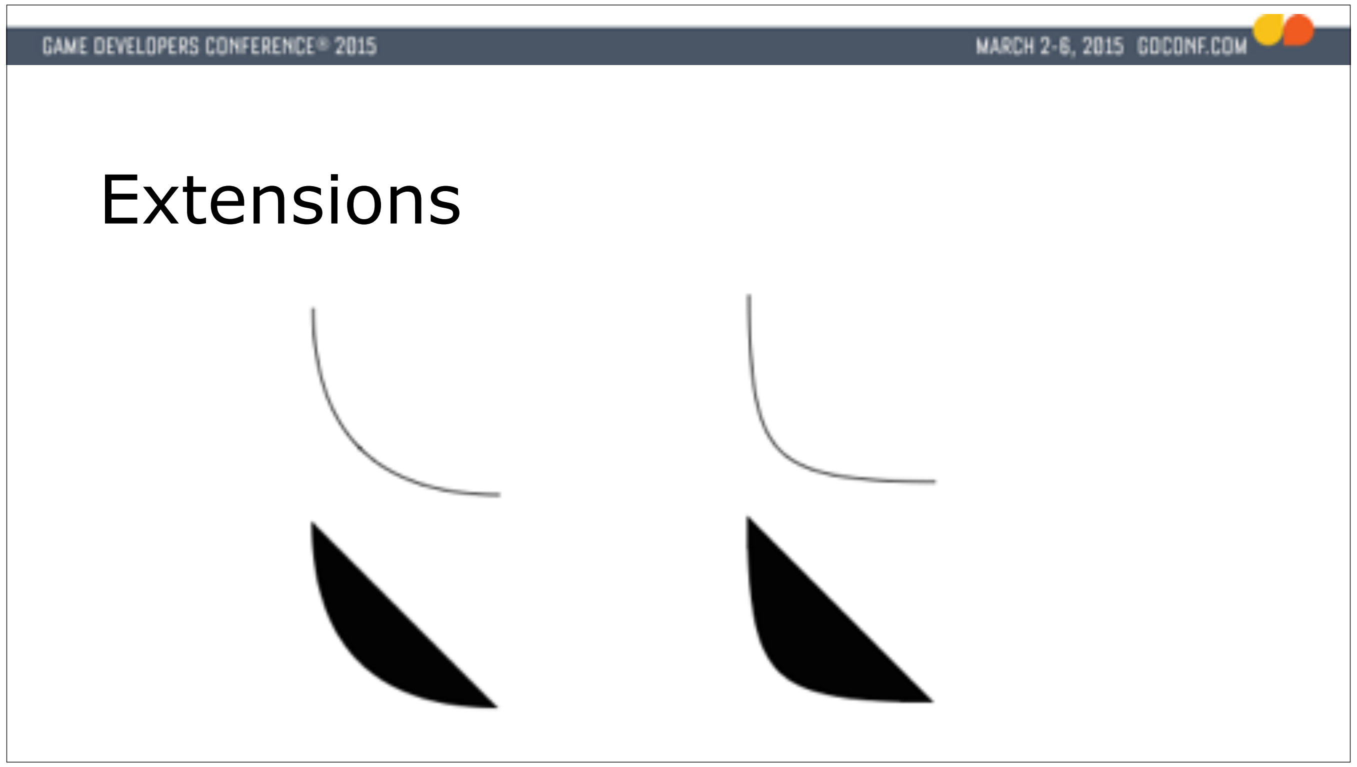
$$d \approx \frac{x^2 + y^2 - 1}{\|\mathbf{J} \cdot (2x, 2y)\|}$$

So for our ellipse or general circles, it looks like this. This is the formula for a unit circle. We apply a transformation to it to stretch into whatever circle or ellipse we want.

Better solution

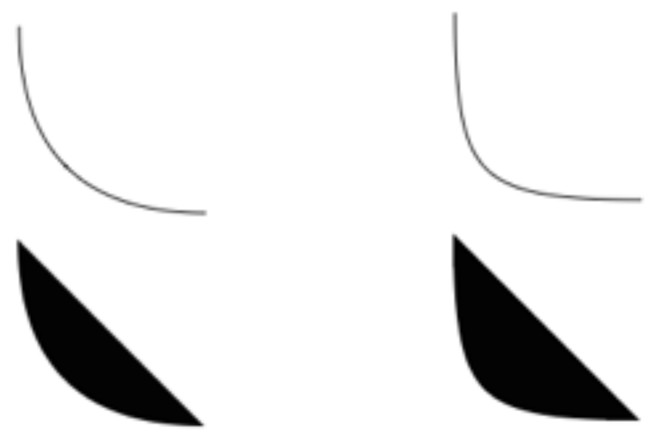


And we can use this to get all sorts of ellipses

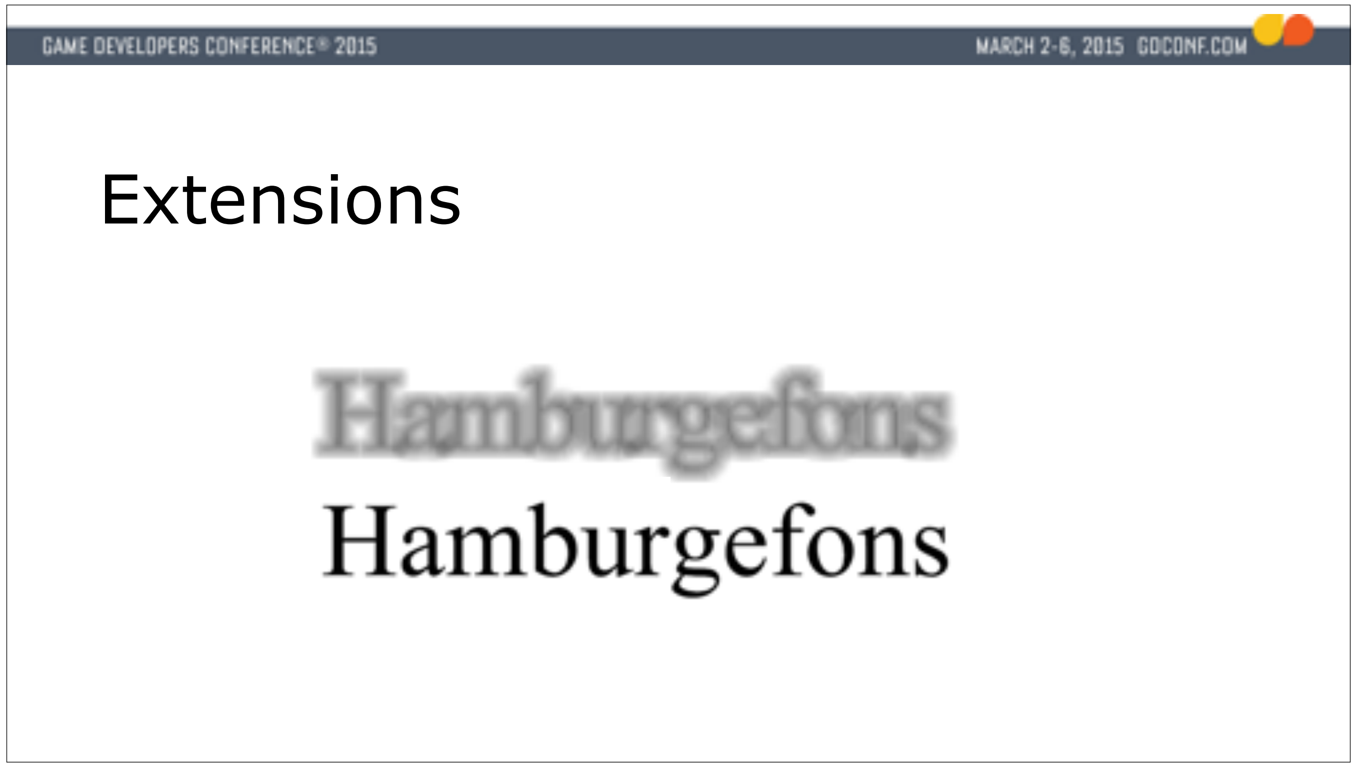


GAME DEVELOPERS CONFERENCE® 2015 MARCH 2-6, 2015 GDCONF.COM

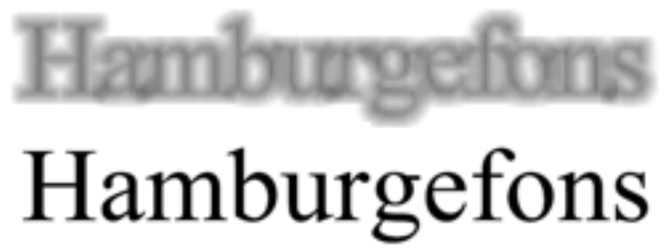
Extensions



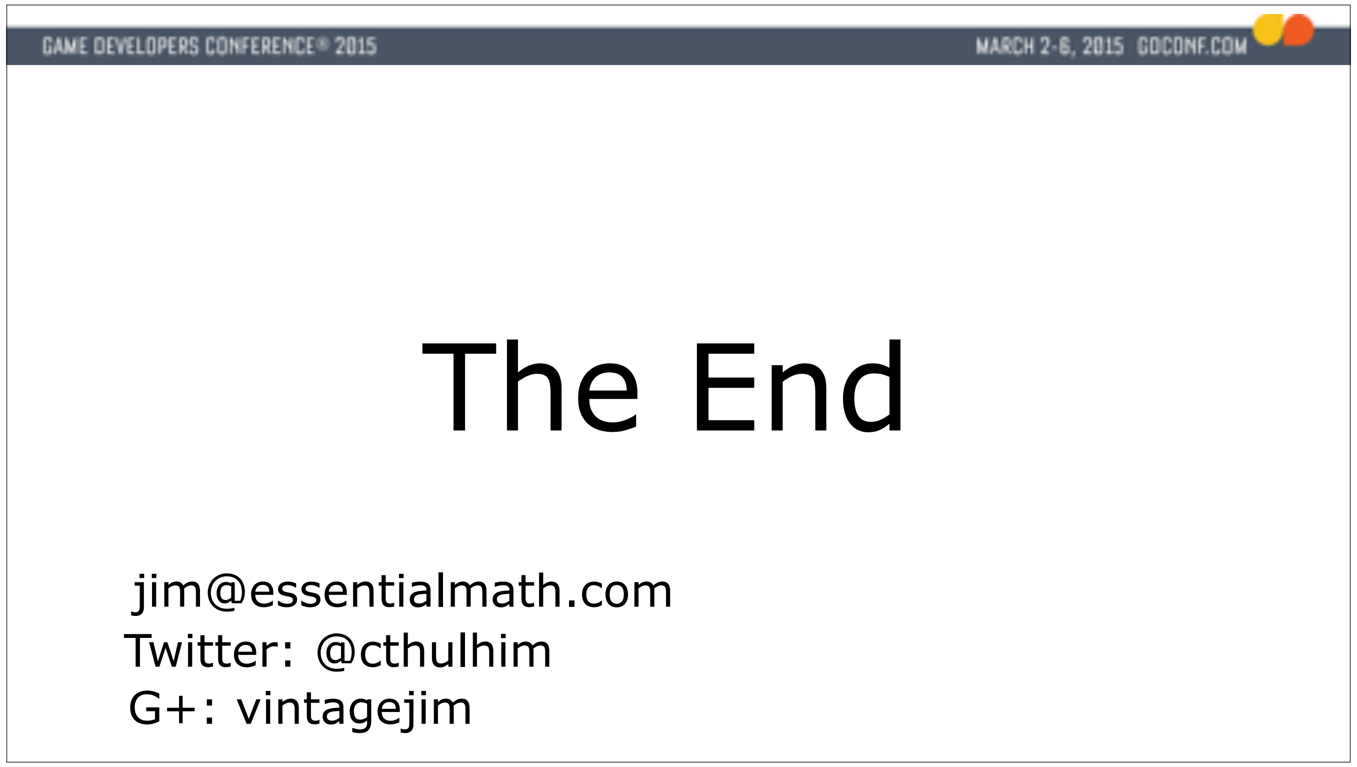
Can use this trick for any implicit function. Can use it for space curves (see Loop-Blinn)



Extensions



Implicit function can also be stored in a texture, i.e. distance field (see Valve). Note: all of the shaders I've seen do the correction improperly. See Skia code for correct antialiasing.



The End

jim@essentialmath.com
Twitter: @cthulhim
G+: vintagejim

We’re hiring!