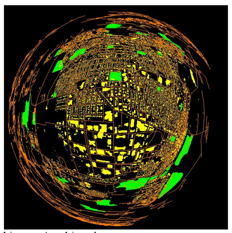
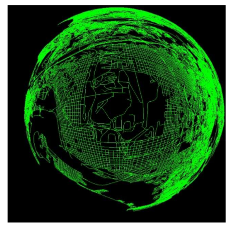
Using Vertex Shaders for Hyperbolic Geometry

Oregon State University

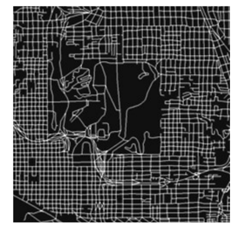
Mike Bailey
mjb@cs.oregonstate.edu

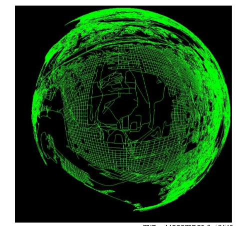




Zooming and Panning Around a Complex 2D Display

- Standard (Euclidean) geometry zooming forces much of the information off the screen
- This eliminates the context from the zoomed-in display
- This problem can be solved with *hyperbolic methods* if we are willing to give up Euclidean geometry
- At one time, this would have also meant severely giving up graphics performance, but not now (thanks to shaders)

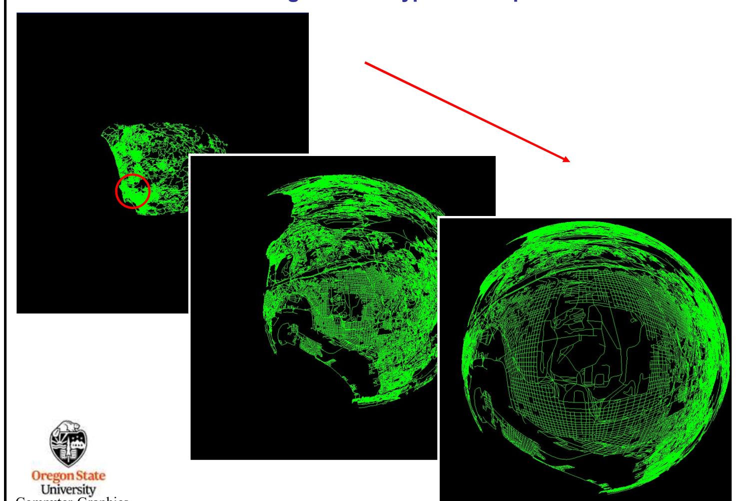




Zooming in *Euclidean* Hyperbolic Space

123,101 line strips
446,585 points

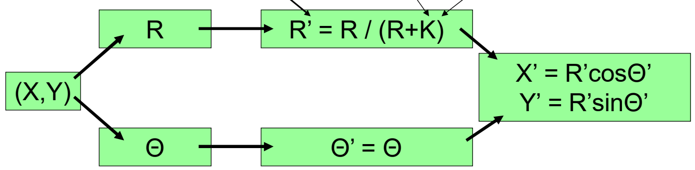
Zooming in *Polar* Hyperbolic Space



Polar Hyperbolic Equations

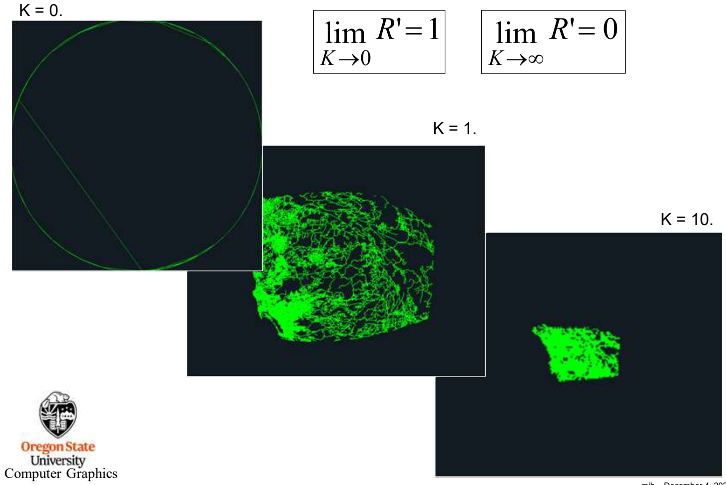
Overall theme: something divided by something a little bigger

$$\lim_{K \to 0} R' = 1 \qquad \lim_{K \to \infty} R' = 0$$



The Effect of K

$$\lim_{K \to 0} R' = 1 \qquad \lim_{K \to \infty} R' = 0$$



Polar Hyperbolic Equations

$$R = \sqrt{X^2 + Y^2}$$

$$\Theta = \tan^{-1}\left(\frac{Y}{X}\right)$$

But fortunately, we don't actually need to ever call the atan( ) function because there are shortcuts to get what we need:

$$\cos\Theta = \frac{X}{R} \qquad \sin\Theta = \frac{Y}{R}$$

$$R' = \frac{R}{R + K}$$

Coordinates moved to the outer edge when K = 0

Coordinates moved to the center when K = ∞

$$X' = R'\cos\Theta = \frac{R}{R+K} \times \frac{X}{R} = \frac{X}{R+K}$$

$$Y' = R'\sin\Theta = \frac{R}{R+K} \times \frac{Y}{R} = \frac{Y}{R+K}$$

Cartesian Hyperbolic Equations

Polar

$$X' = \frac{X}{R + K} \qquad Y' = \frac{Y}{R + K}$$

Coordinates moved to the outer edge when K = 0

Coordinates moved to the center when K = ∞

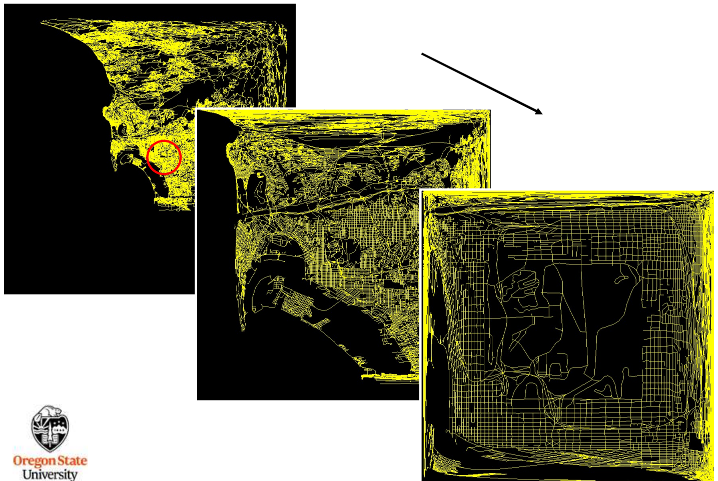
Cartesian

$$X' = \frac{X}{\sqrt{X^2 + K^2}} \qquad Y' = \frac{Y}{\sqrt{Y^2 + K^2}}$$

Coordinates moved to the outer edge when K = 0

Coordinates moved to the center when K = ∞

Zooming in Cartesian Hyperbolic Space



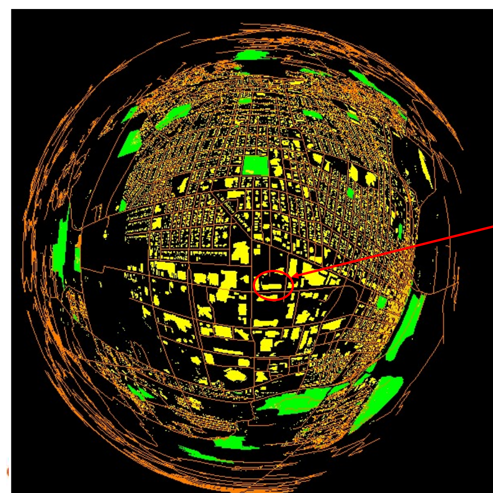
hyper.vert

```
#version 330 compatibility
uniform bool      uPolar;
uniform float     uK;
uniform float     uTransX;
uniform float     uTransY;
out vec3          vColor;

void
main( )
{
      vColor = gl_Color.rgb;

      vec2 pos = ( gl_ModelViewMatrix * gl_Vertex ).xy;
      pos += vec2( uTransX, uTransY );
      float r = length( pos );

      vec4 pos2 = vec4( 0., 0., -5., 1. );

      if( uPolar )
            pos2.xy = pos / ( r + uK );
      else
            pos2.xy = pos / ( pos*pos + uK*uK );

      gl_Position = gl_ProjectionMatrix * pos2;
}
```


hyper.frag

```
#version 330 compatibility
in vec3    vColor;

void
main(  )
{
      gl_FragColor = vec4( vColor, 1. );
}
```

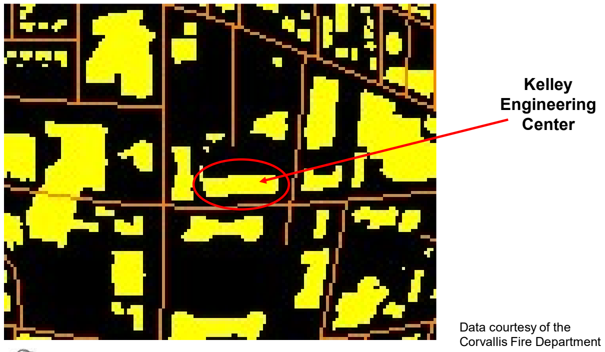

Corvallis Streets, Buildings, Parks



Kelley Engineering Center

Data courtesy of the Corvallis Fire Department

Kelley Engineering Center



Data courtesy of the
Corvallis Fire Department