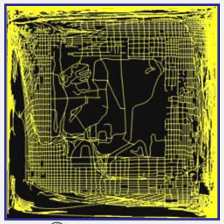
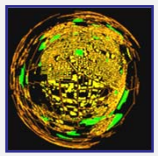
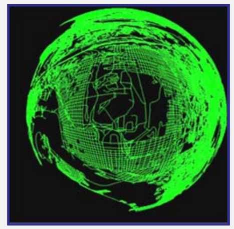
Hyperbolic Geometry for Visualization



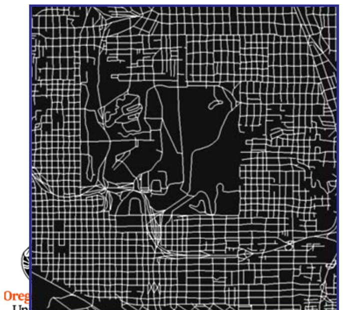


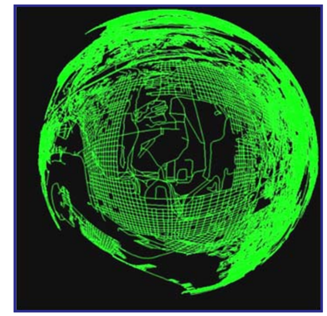


Oregon State University
Mike Bailey
mjb@cs.oregonstate.edu

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

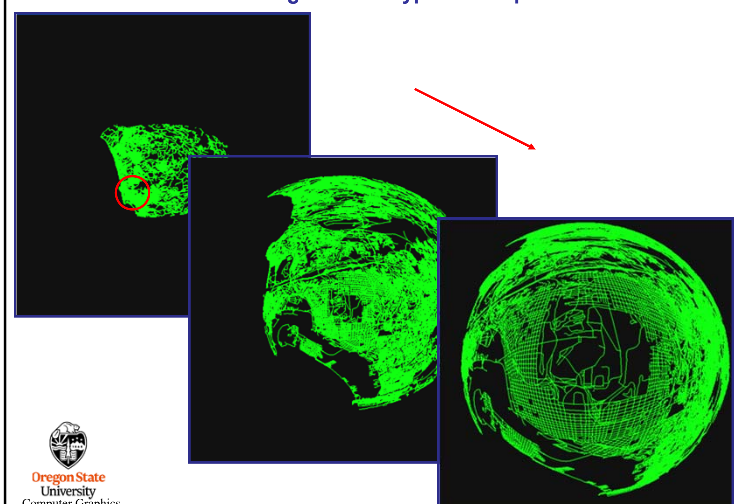




Usual Zooming in Euclidean Space

123,101 line strips
446,585 points

Zooming in Polar Hyperbolic Space



Polar Hyperbolic Equations

Overall theme: something divided by something else a little bigger

(X,Y) → R → R' = R / (R+K) → X' = R'cosΘ', Y' = R'sinΘ'

(X,Y) → Θ → Θ' = Θ → X' = R'cosΘ', Y' = R'sinΘ'

Because $R' = \frac{R}{R+K}$ **then:** $\boxed{\lim_{K \to 0} R' = 1}$ $\boxed{\lim_{K \to \infty} R' = 0}$

Polar Hyperbolic Equations Don't Actually Need to use Trig

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

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

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

Coordinates moved to outer edge when K = 0

Coordinates moved to 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 – Treat X and Y Independently

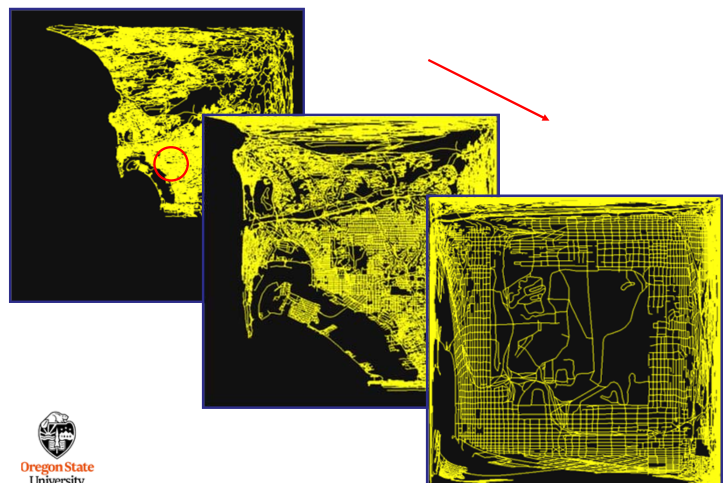
Polar
$$\begin{cases} X' = \dfrac{X}{R+K} \\ Y' = \dfrac{Y}{R+K} \end{cases}$$

Cartesian
$$\begin{cases} X' = \dfrac{X}{\sqrt{X^2 + K^2}} \\ Y' = \dfrac{Y}{\sqrt{Y^2 + K^2}} \end{cases}$$

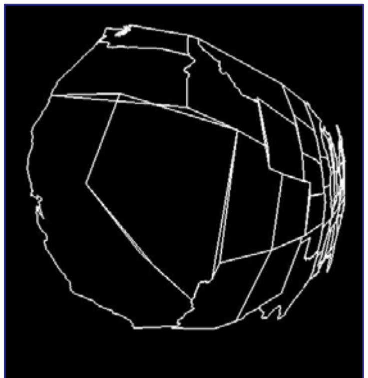
Coordinates moved to outer edge when K = 0

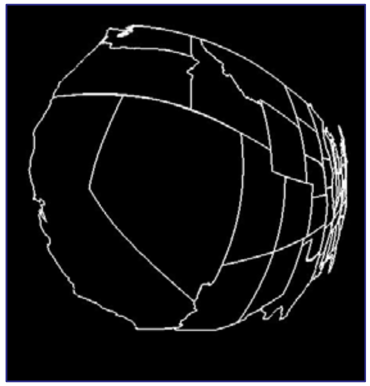
Coordinates moved to center when K = ∞

Zooming in Cartesian Hyperbolic Space



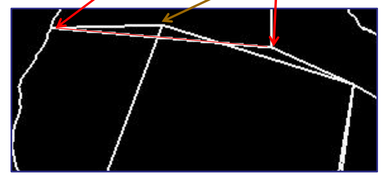
The Problem with T-Intersections





The Problem with T-Intersections

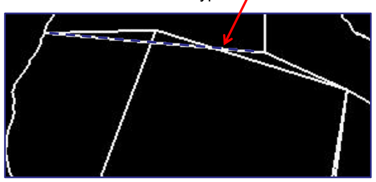
Your code computes the hyperbolic transformation here and here, and OpenGL draws a straight line between them. But, this point had its hyperbolic transformation computed separately, and doesn't match up with the straight line.



This kind of situation is called a T-intersection, and crops up all the time in computer graphics, even though we don't want it to. ☹

A Solution to the T-Intersection Problem

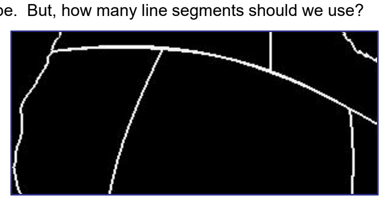
Break this line up into several (many?) sub-pieces, and perform the Hyperbolic Transformation on each intermediate point.



$$P(t) = (1-t)P_0 + tP_1$$

t = 0., .01, .02, .03, …

This makes that straight line into a curve, as it should be. But, how many line segments should we use?

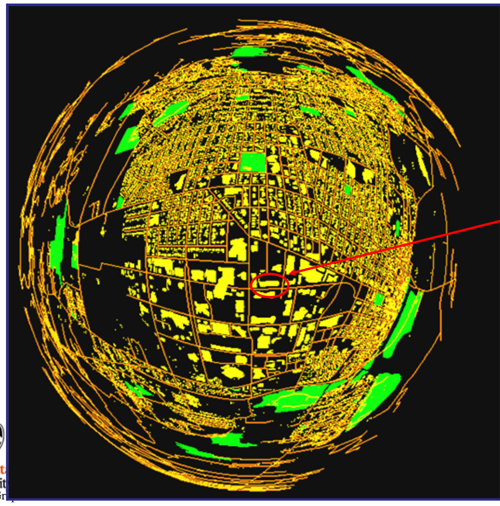


A More Elegant Approach is to Recursively Subdivide

void
DrawHyperbolicLine(P_0, P_1)
{
    Compute point $A = \frac{P_0 + P_1}{2.}$
    Convert point A to Hyperbolic Coordinates, calling it A'
    Convert P_0 and P_1 to Hyperbolic Coordinates P_0', P_1'
    Compute point $B' = \frac{P_0' + P_1'}{2.}$
    Compare A' and B
    if(they are "close enough")
    {
        Draw the line P_0'-P_1'
    }
    else
    {
        DrawHyperbolicLine(P_0, A);
        DrawHyperbolicLine(A, P_1);
    }
}

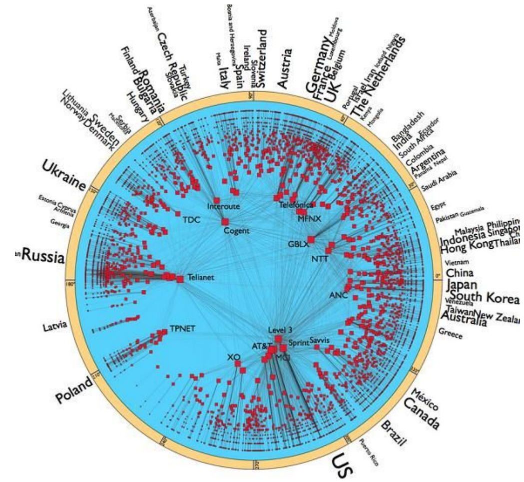
Subdividing to render a curve or surface correctly is a recurring theme in computer graphics.

Hyperbolic Corvallis (Streets, Buildings, Parks)



Kelley
Engineering
Center

Oregon State University
Computer Graphics



Oregon State University
Computer Graphics

http://www.sott.net/articles/show/215021-Hyperbolic-map-of-the-internet-will-save-it-from-COLLAPSE