Using Vertex Shaders for Hyperbolic Geometry

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

Zooming in Euclidean Space
123,101 line strips
446,585 points

Zooming in Polar Hyperbolic Space
Coordinates moved to outer edge when \( K = 0 \)
Coordinates moved to center when \( K = \infty \)

Polar Hyperbolic Equations

Overall theme: something divided by something a little bigger

\[
\lim_{K \to 0} R' = \frac{1}{1+K} \\
\lim_{K \to \infty} R' = 0
\]

\[
X' = R' \cos \Theta' \\
Y' = R' \sin \Theta'
\]

Polar Hyperbolic Equations

\[
R = \sqrt{X^2 + Y^2} \\
\Theta = \tan^{-1} \left( \frac{Y}{X} \right) \\
R' = \frac{R}{R+K} \\
X' = R' \cos \Theta = \frac{R X}{R+K} \\
Y' = R' \sin \Theta = \frac{R Y}{R+K}
\]
Cartesian Hyperbolic Equations

Polar

\[ X' = \frac{X}{\sqrt{1 + K^2}} \]
\[ Y' = \frac{Y}{\sqrt{1 + K^2}} \]

Coordinates move to center when \( K = \infty \)
Coordinates move to outer edge when \( K = 0 \)

Cartesian

\[ X' = \frac{X^2 + Y^2}{2K} - \frac{XK}{2} \]
\[ Y' = \frac{Y}{\sqrt{1 + K^2}} \]

Coordinates move to center when \( K = \infty \)
Coordinates move to outer edge when \( K = 0 \)

Zooming in Cartesian Hyperbolic Space

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

void main( void )
{
  vColor = aColor.rgb;
  vec2 pos = ( uModelViewMatrix * aVertex ).xy;
  pos += vec2( uTransX, uTransY );
  float r = length( pos.xyz );
  vec4 pos2 = vec4( 0., 0., -5., 1. );
  if( uPolar )
    pos2.xy = pos / ( r + uK );
  else
    pos2.xy = pos / ( pos*pos + uK*uK );
  gl_Position = uProjectionMatrix * pos2;
}
```

```
#version 330 compatibility
in vec3 vColor;
out vec4 fFragColor;

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