Using Vertex Shaders for Hyperbolic Geometry

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

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
Polar Hyperbolic Equations

Overall theme: something divided by something a little bigger

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

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

\( R' = R / (R+K) \)

\( (X,Y) \rightarrow X' = R' \cos \Theta' \)
\( Y' = R' \sin \Theta' \)

\( \Theta' = \Theta \)

\[ R = \sqrt{X^2 + Y^2} \]

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

\[ R' = \frac{R}{R+K} \]

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

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

Coordinates moved to outer edge when \( K = 0 \)

Coordinates moved to center when \( K = \infty \)
Cartesian Hyperbolic Equations

Polar

\[ X' = \frac{X}{K + R} \]
\[ Y' = \frac{Y}{K + R} \]

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

Cartesian

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

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

Zooming in Cartesian Hyperbolic Space
# hyper.vert

```glsl
#version 330 compatibility
uniform bool uPolar;
uniform float uK;
uniform float uTransX;
uniform float uTransY;
out vec4 vColor;

void main( void )
{
    vColor = aColor;
    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;
}
```

# hyper.frag

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

void main( )
{
    fFragColor = vColor;
}
```