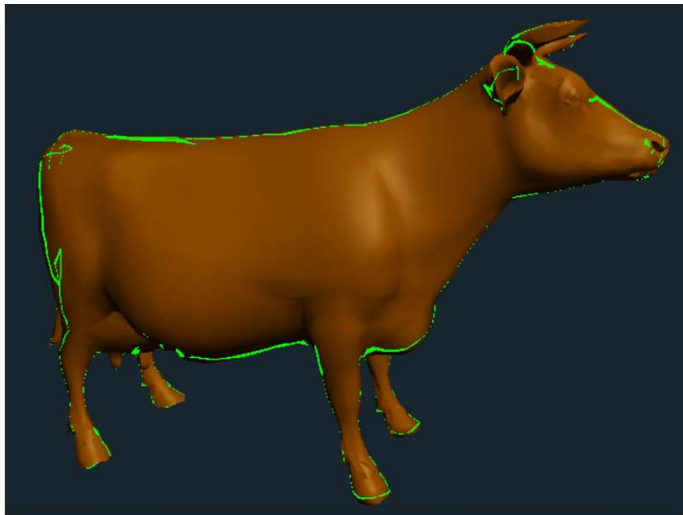


Fragment Shader Silhouettes



Oregon State
University

Mike Bailey

mjb@cs.oregonstate.edu



Oregon State
University
Computer Graphics

Fragment Shader Silhouettes

fsilh.glib

```
##OpenGL GLIB
Perspective 70
LookAt 0 0 10  0 0 0  0 1 0

Vertex    fsilh.vert
Fragment  fsilh.frag
Program   FragSilhouette

        uColor {0.50 0.25 0. 1.}
        uSilh <false>
        uDeltaZ <0. 0.001 0.002>
        uOpaque <true>
        uTol <0. 0.04 0.5>
        uKa <0. 0.1 1.>
        uKd <0. 0.8 1.>
        uKs <0. 0.1 1.>
        uShininess <2. 10. 50.>
```

Obj cow.obj

Fragment Shader Silhouettes

fsilh.vert

```
#version 330 compatibility

out vec2  vST;           // texture coords
out vec3  vN;           // normal vector
out vec3  vL;           // vector from point to light
out vec3  vE;           // vector from point to eye
out float  vNz;          // z-component of normal

const vec3 LIGHTPOS = vec3( 5.,10.,10.);

void
main()
{
    vST = gl_MultiTexCoord0.st;
    vN = normalize( gl_NormalMatrix * gl_Normal );           // normal vector
    vNz = normalize( gl_NormalMatrix * gl_Normal ).z;
    vec4 ECposition = gl_ModelViewMatrix * gl_Vertex;
    vL = LIGHTPOS - ECposition.xyz;           // vector from the point
                                           // to the light position
    vE = vec3( 0., 0., 0. ) - ECposition.xyz;           // vector from the point
                                           // to the eye position
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

Fragment Shader Silhouettes

fsilh.frag, I

```
#version 330 compatibility

uniform vec4  uColor;
uniform bool   uSilh;
uniform float  uDeltaZ;
uniform bool   uOpaque;
uniform float  uTol;
uniform float  uKa, uKd, uKs;      // coefficients of each type of lighting
uniform float  uShininess;        // specular exponent

in  vec2  vST;                    // texture coords
in  vec3  vN;                     // normal vector
in  vec3  vL;                     // vector from point to light
in  vec3  vE;                     // vector from point to eye
in  float  vNz;

const vec3 SPECULARCOLOR = vec3( 1., 1., 1. );
const vec3 SILHCOLOR     = vec3( 0., 1., 0. );

void
main()
{
    vec3 myColor = uColor.rgb;

    // per-fragment lighting:
    vec3 Normal = normalize(vN);
    vec3 Light  = normalize(vL);
    vec3 Eye    = normalize(vE);

    vec3 ambient = uKa * myColor;
```

Fragment Shader Silhouettes

fsilh.frag, II

```

float d = max( dot(Normal,Light), 0. );    // only do diffuse if the light can see the point
vec3 diffuse = uKd * d * myColor;

float s = 0.;
if( d > 0. )          // only do specular if the light can see the point
{
    vec3 ref = normalize( reflect( -Light, Normal ) );
    float cosphi = dot( Eye, ref );
    if( cosphi > 0. )
        s = pow( max( cosphi, 0. ), uShininess );
}
vec3 specular = uKs * s * SPECULARCOLOR.rgb;

float deltaz = 0.;
if( uSilh && ( abs(vNz) <= uTol ) )
{
    gl_FragColor = vec4( SILHCOLOR, 1. );
    deltaz = -uDeltaZ;
}
else
{
    if( ! uOpaque )
        discard;

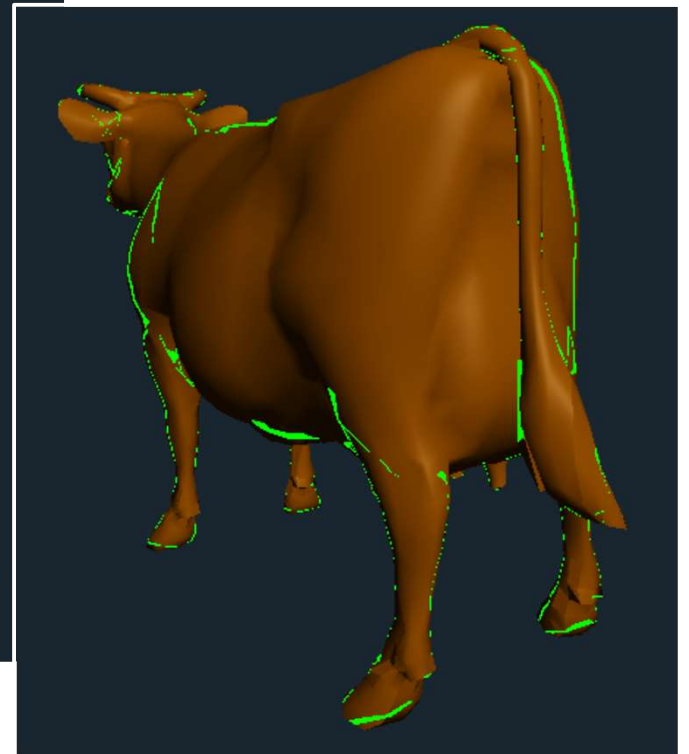
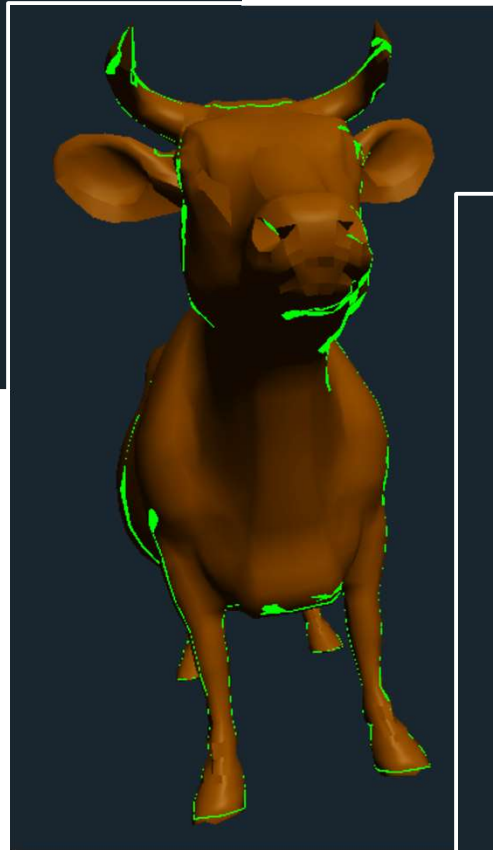
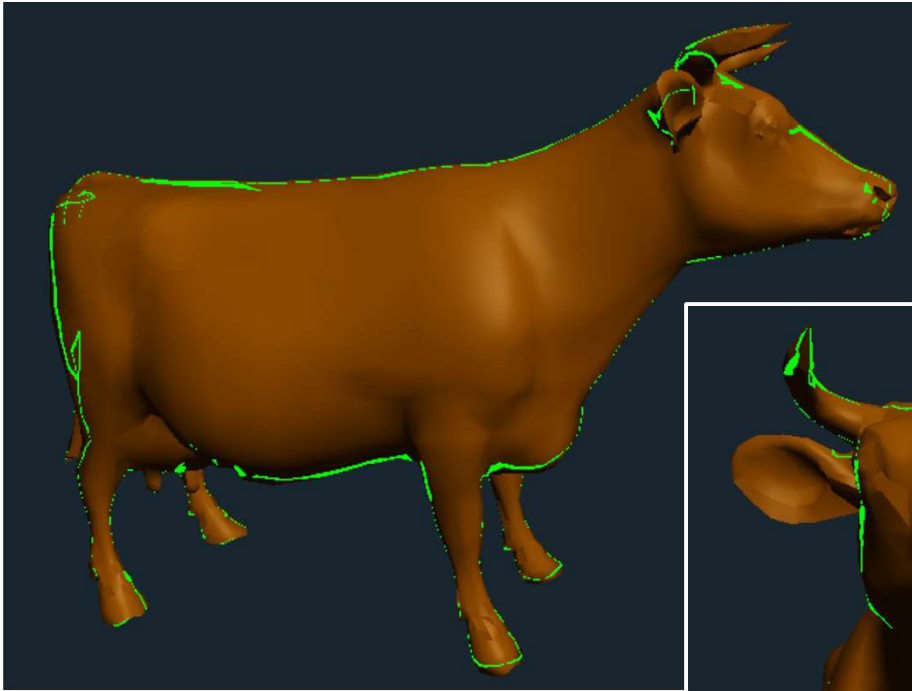
    gl_FragColor = vec4( ambient + diffuse + specular, 1. );
}
gl_FragDepth = gl_FragCoord.z + deltaz;
}

```

gl_FragCoord gives the x and y pixel coordinates and the z-buffer value of this fragment.

gl_FragDepth is used to change the z-buffer value of this fragment.

Fragment Shader Silhouettes



Fragment Shader Silhouettes



Something Fun – Only Draw the Silhouette Fragments

8

