Geometric Morphing with the Vertex Shader

Oregon State University

Mike Bailey

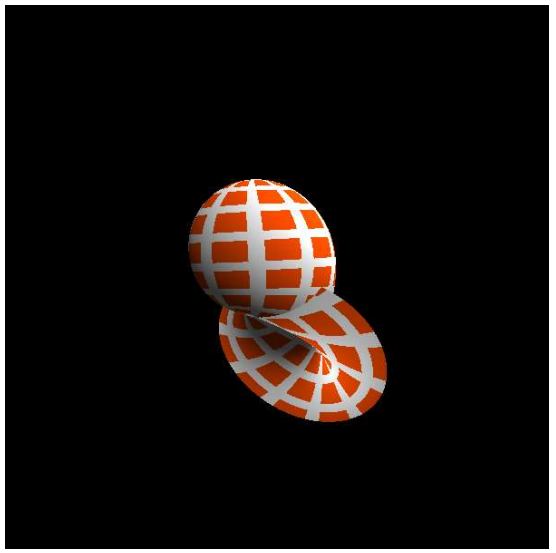
mjb@cs.oregonstate.edu

Oregon State University
Computer Graphics

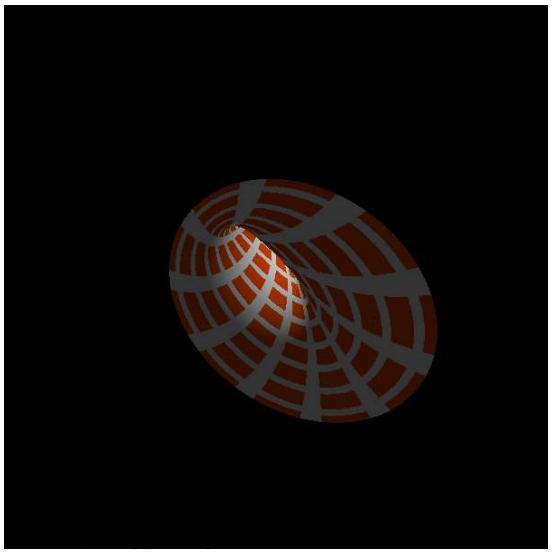
Morphing a Sphere into a Circle

Blend = 0.00

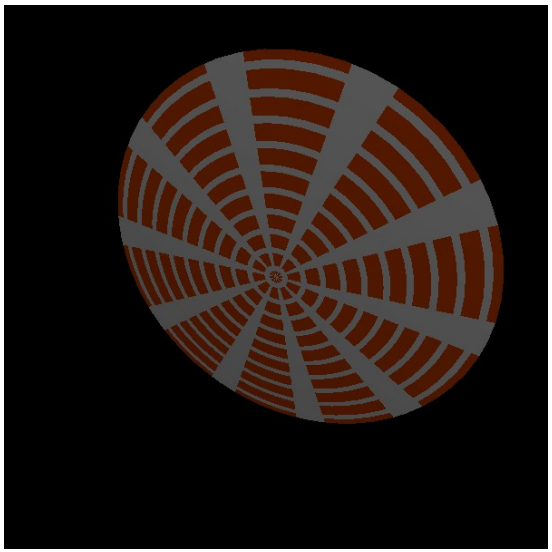
Blend = 0.25



Blend = 0.50



Blend = 0.75



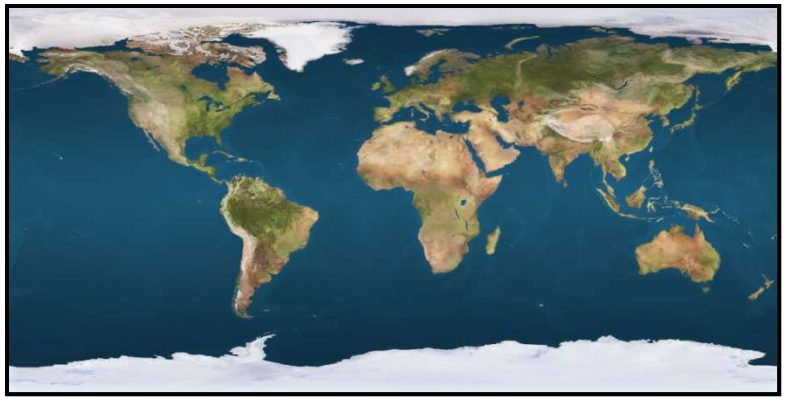
Blend = 1.00

vertex shader:

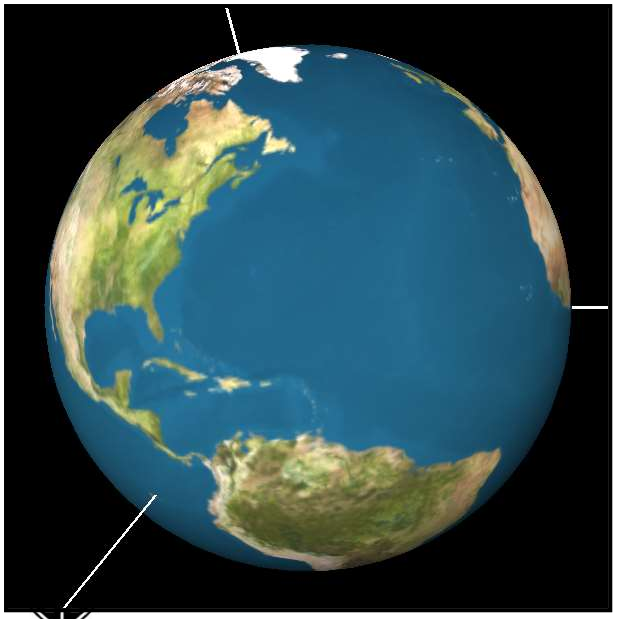
```
uniform float               uOffsetS, uOffsetT, uBlend;
out vec2                    vST;
out float                   vLightIntensity;
out vec3                    vColor;
const float TWOPI = 2.*3.14159265;
const vec3 LIGHTPOS = vec3( 5., 10., 10. );

// original model coords  of the sphere:

vec4 vertex0  = gl_Vertex;
vec3 norm0   = gl_Normal;

// circle coords:

vST= gl_MultiTexCoords0.st;
float radius = 1 . -  vST.t;
float theta = TWOPI * vST.s;
vec4  circle = vec4( radius*cos(theta), radius*sin(theta), 0., 1. );
vec3  circlenorm = vec3( 0., 0., 1. );                    // normal vector

vST += vec2( uOffsetS, uOffsetT );

// blend the 2 shapes:
vec4 theVertex   = mix( vertex0, circle, uBlend );
vec3 theNormal = normalize(  mix( norm0, circlenorm, uBlend )  );

// do the lighting:
vec3 tnorm      = normalize( vec3( gl_NormalMatrix * theNormal ) );
vec3 ECposition = vec3( gl_ModelViewMatrix * theVertex );
vLightIntensity  = abs( dot( normalize(LIGHTPOS - ECposition), tnorm ) );

vColor = gl_Color.rgb;
gl_Position = gl_ModelViewProjectionMatrix * theVertex;
```

A possible vis application ??

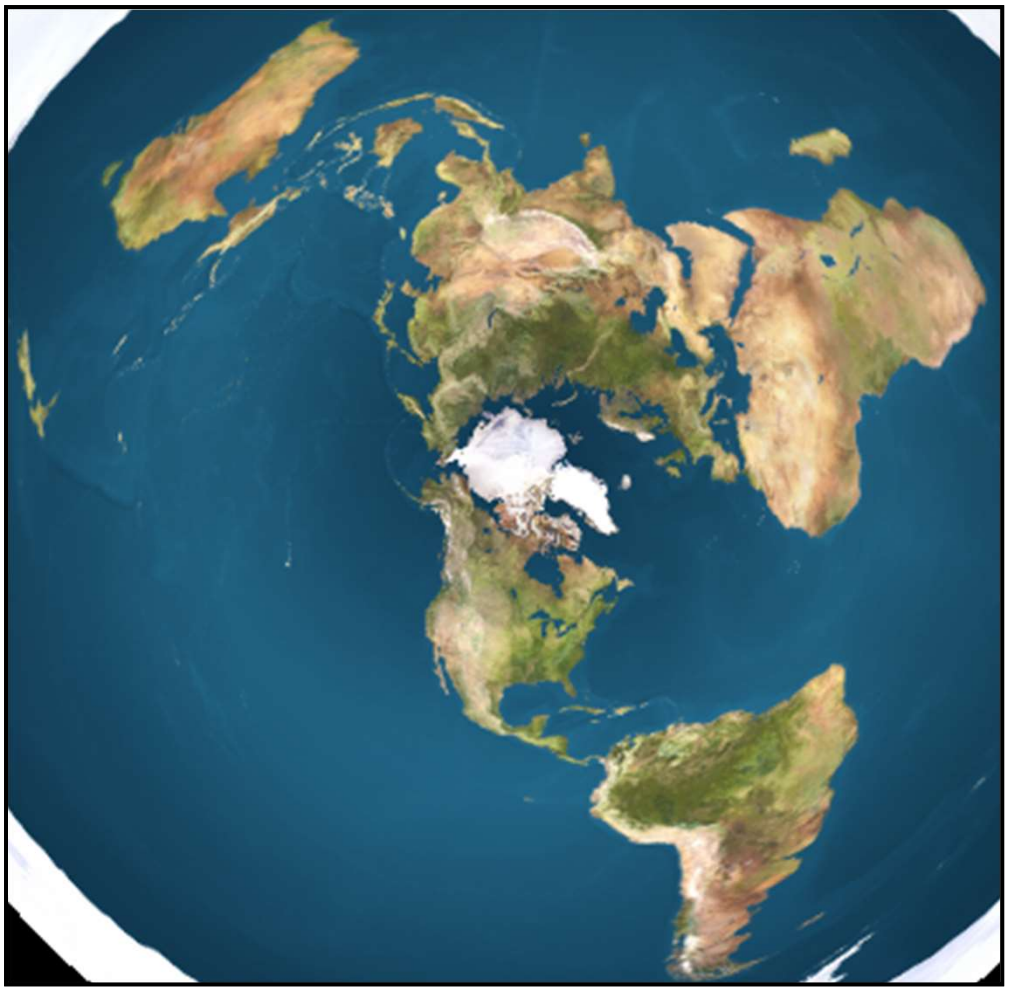
What an interesting 2D view of Earth!



Original texture map

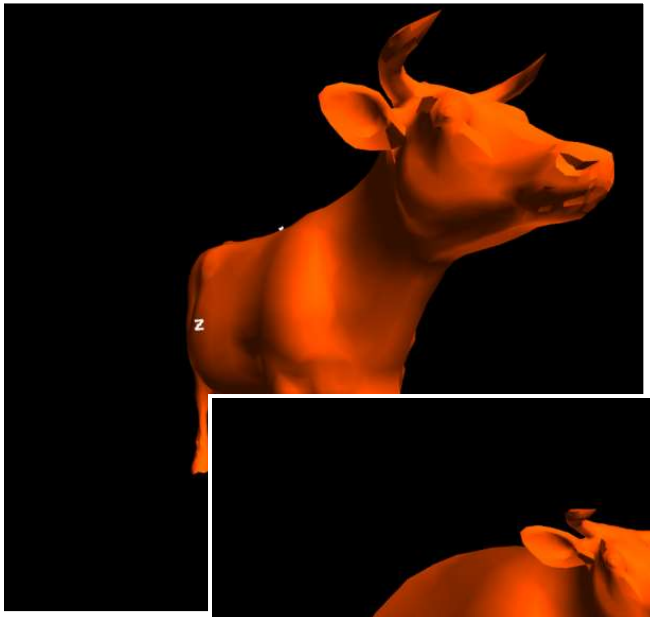


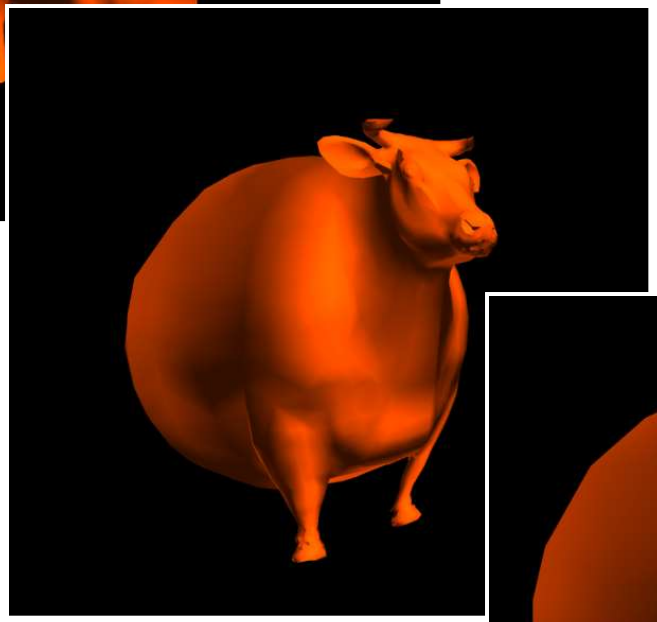
Mapped onto a Sphere

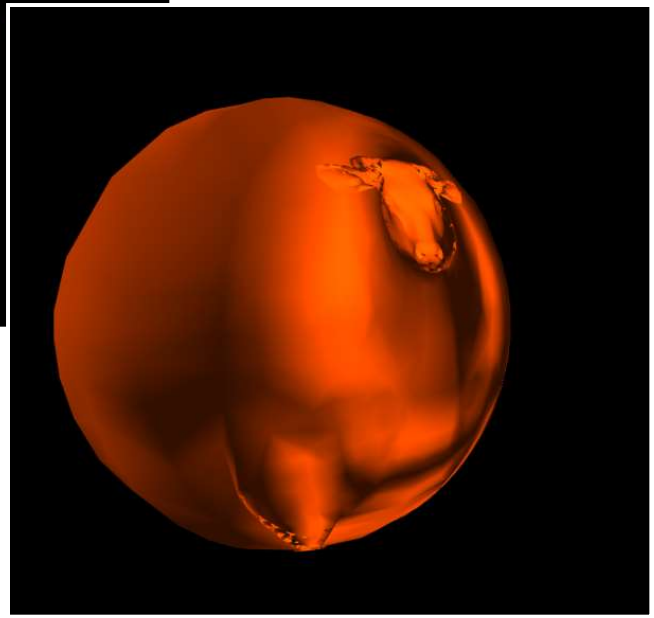


Morphed into a Circle

Morphing a Cow into a Sphere





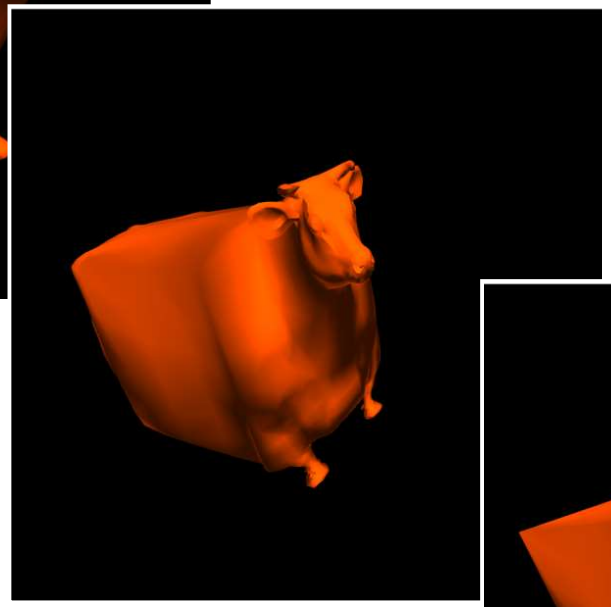


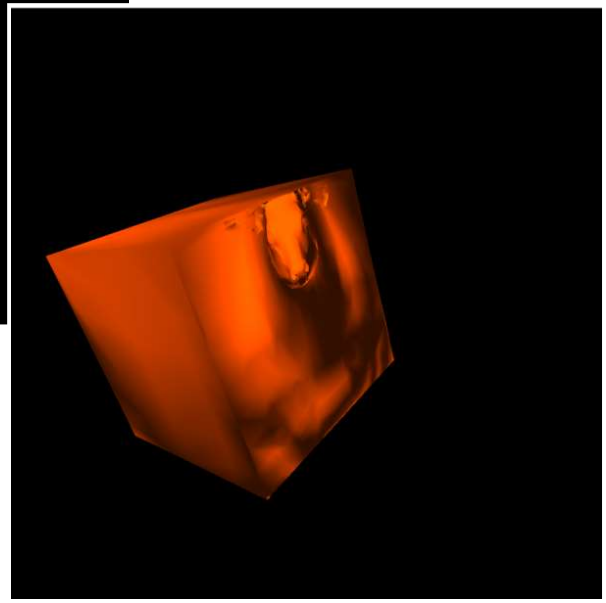
```
vec4 vertex = gl_Vertex;
vertex.xyz *= 4. / length(vertex.xyz);
```

Basically scale all vertex coordinates to be a distance of 4.0 from the origin.

Note: the “face” in the sphere cow is there because the normals were not morphed into sphere normals – they were left as cow normals

Morphing a Cow into a Cube





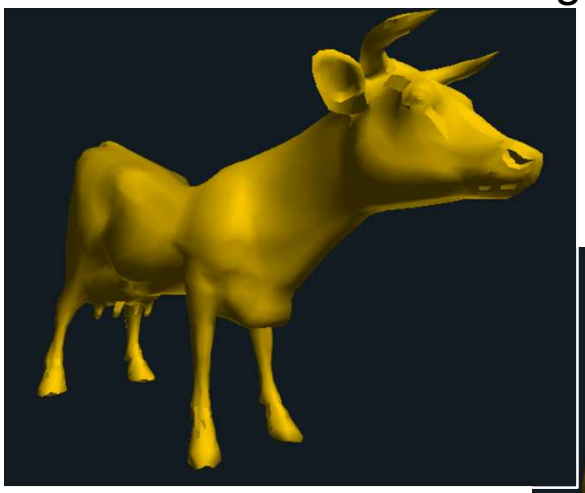
```
const float SIDE = 2.;

vec4 vertex = gl_Vertex;
vertex.xyz *= 4. / length(vertex.xyz);
vertex.xyz = clamp( vertex.xyz, -SIDE, SIDE );
```

Basically scale all vertex coordinates to be a distance of 4.0 from the origin, then clamp them to be in the range –2. to +2.

Note: the “face” in the cube cow is there because the normals were not morphed into cube normals – they were left as cow normals

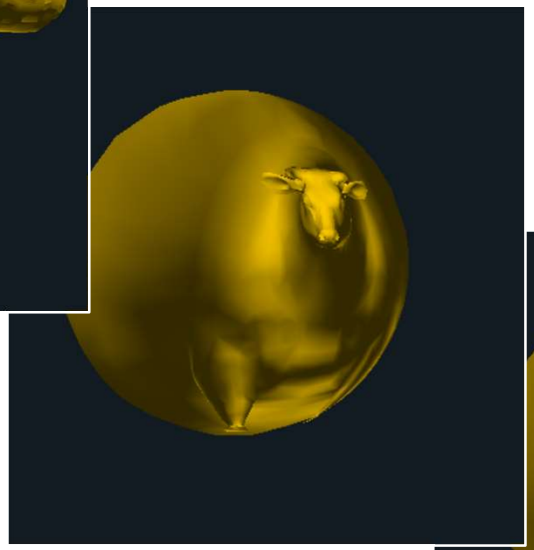
What If We Go Outside the 0. to 1. Range for t?

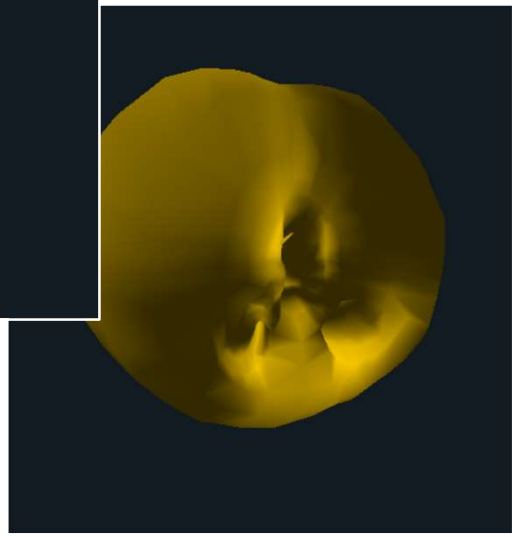
t = -1. : more cow and negative sphere



t = 0. : all cow and zero sphere

t = +1. : zero cow and all sphere





t = +2. : negative cow and more sphere

For you movie fans, what about "Real Morphing"?

"Real Morphing" involves interpolating key points from one object into key points in another. This flies in the face of graphics hardware's philosophy of dealing with one triangle and then getting rid of any record of it. Movies do this in software. We got away with it in our class because we knew the equation of a disk, a sphere, and a cube and so could interpolate in a vertex shader.

The first movie-morphing I remember seeing is from the fantasy movie *Willow:*

https://www.youtube.com/watch?v=IKzbsDG58pc

The "making of" video for this is here:

https://www.youtube.com/watch?v=kxVwNIZDQJ0

But, my nomination for #1 morphing ***ever*** is in Michael Jackson's *Black or White* video:

https://www.youtube.com/watch?v=F2AitTPI5U0

The great morphing starts at around 05:30.