WebGL - Fun

A “Funday” Project

From Workiva

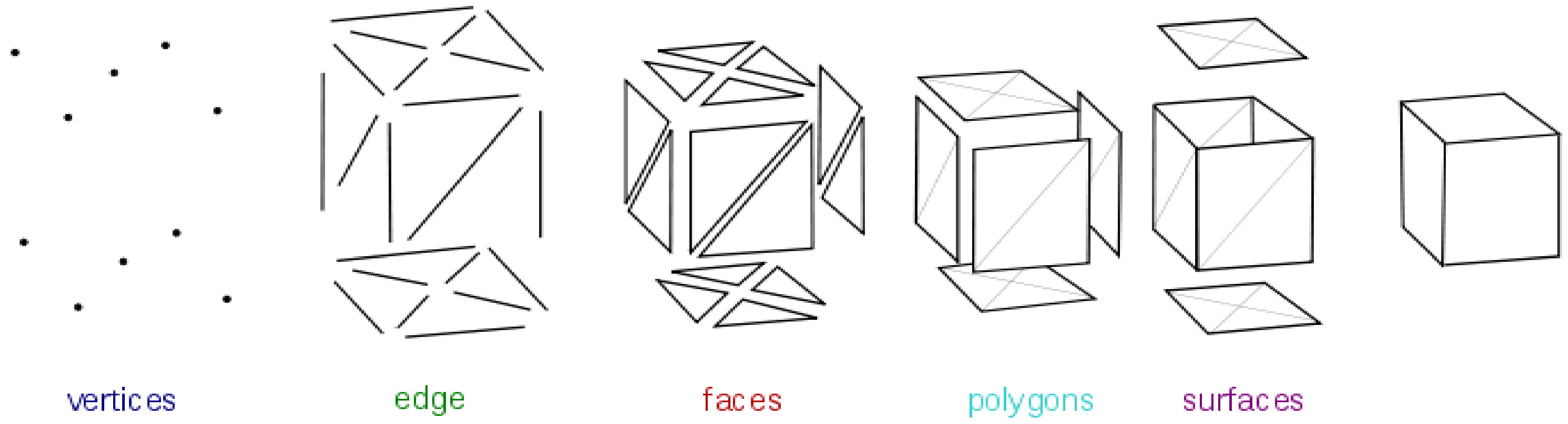
By Grant Nelson

Introduction

- All code and data can be found at github.com/grantnelson-wf/webgl-fun
- Cube maps from www.humus.name/index.php?page=Textures
- Other images found via Google image search.
- Some diagrams in this document are from Wikipedia.

These slides are not meant to teach shaders with but is an outline for the presentation on the research project.

Data View



- Data: Pos, Clr3, Clr4, Norm, Txt, Cube, Binm, Wght, Adj1, Adj2
- Math: objMat, viewMat, projMat
- Creating shapes and render targets.
- This presentation is focused on the shaders.

Color

Vertex Shader

```
uniform mat4 objMat;
uniform mat4 viewMat;
uniform mat4 projMat;

attribute vec3 posAttr;
attribute vec3 clr3Attr;

varying vec4 vColor;

void main()
{
  gl_Position = projMat*viewMat*objMat*vec4(posAttr, 1.0);
  vColor = vec4(clr3Attr, 1.0);
}
```

Color

Fragment Shader

```
precision mediump float;

varying vec4 vColor;

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

Fog

Vertex Shader

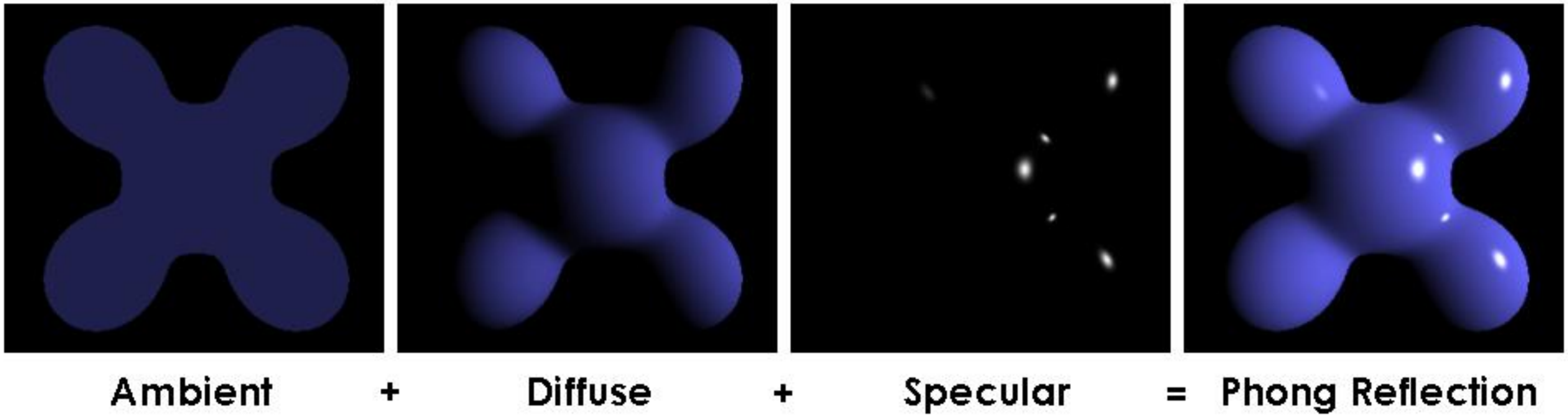
```
uniform mat4 objMat;
uniform mat4 viewMat;
uniform mat4 projMat;

attribute vec3 posAttr;

varying float depth;

void main()
{
  vec4 pos = viewMat*objMat*vec4(posAttr, 1.0);
  depth = pos.z;
  gl_Position = projMat*pos;
}
```

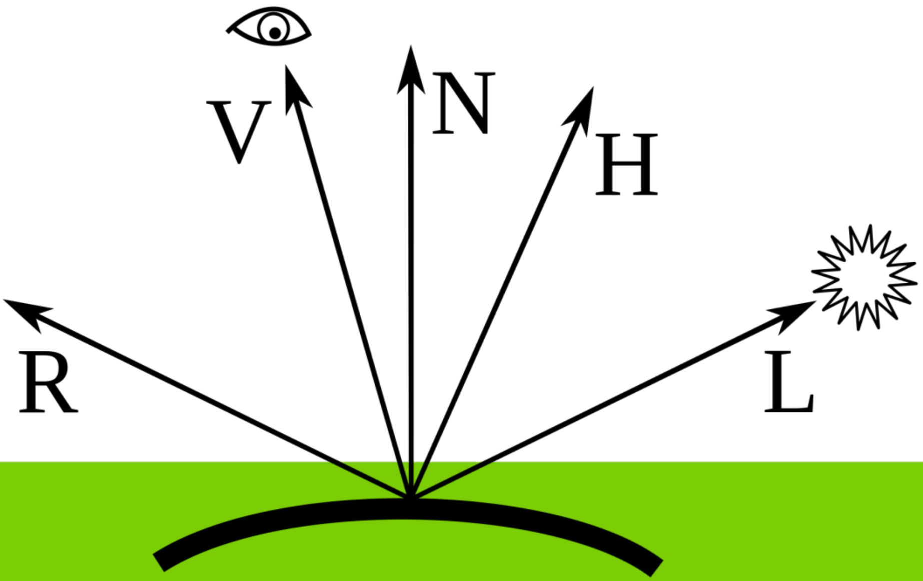
Fog

Fragment Shader

```
precision mediump float;

uniform vec3 objClr;
uniform vec3 fogClr;
uniform float fogStart;
uniform float fogStop;

varying float depth;

void main()
{
    float factor = (depth-fogStop)/(fogStart-fogStop);
    factor = clamp(factor, 0.0, 1.0);
    gl_FragColor = vec4(mix(fogClr, objClr, factor), 1.0);
}
```

Directional Light





Directional Light

Vertex Shader

```
uniform mat4 objMat;
uniform mat4 viewMat;
uniform mat4 projMat;
uniform vec3 lightVec;

attribute vec3 posAttr;
attribute vec3 normAttr;

varying vec3 normal;
varying vec3 litVec;
varying vec3 camPos;

void main()
{
  camPos = (viewMat*vec4(0.0, 0.0, 0.0, -1.0)).xyz;
  normal = normalize(objMat*vec4(normAttr, 0.0)).xyz;
  litVec = normalize((viewMat*vec4(lightVec, 0.0)).xyz);
  gl_Position = projMat*viewMat*objMat*vec4(posAttr, 1.0);
}
```

Directional Light

Fragment Shader

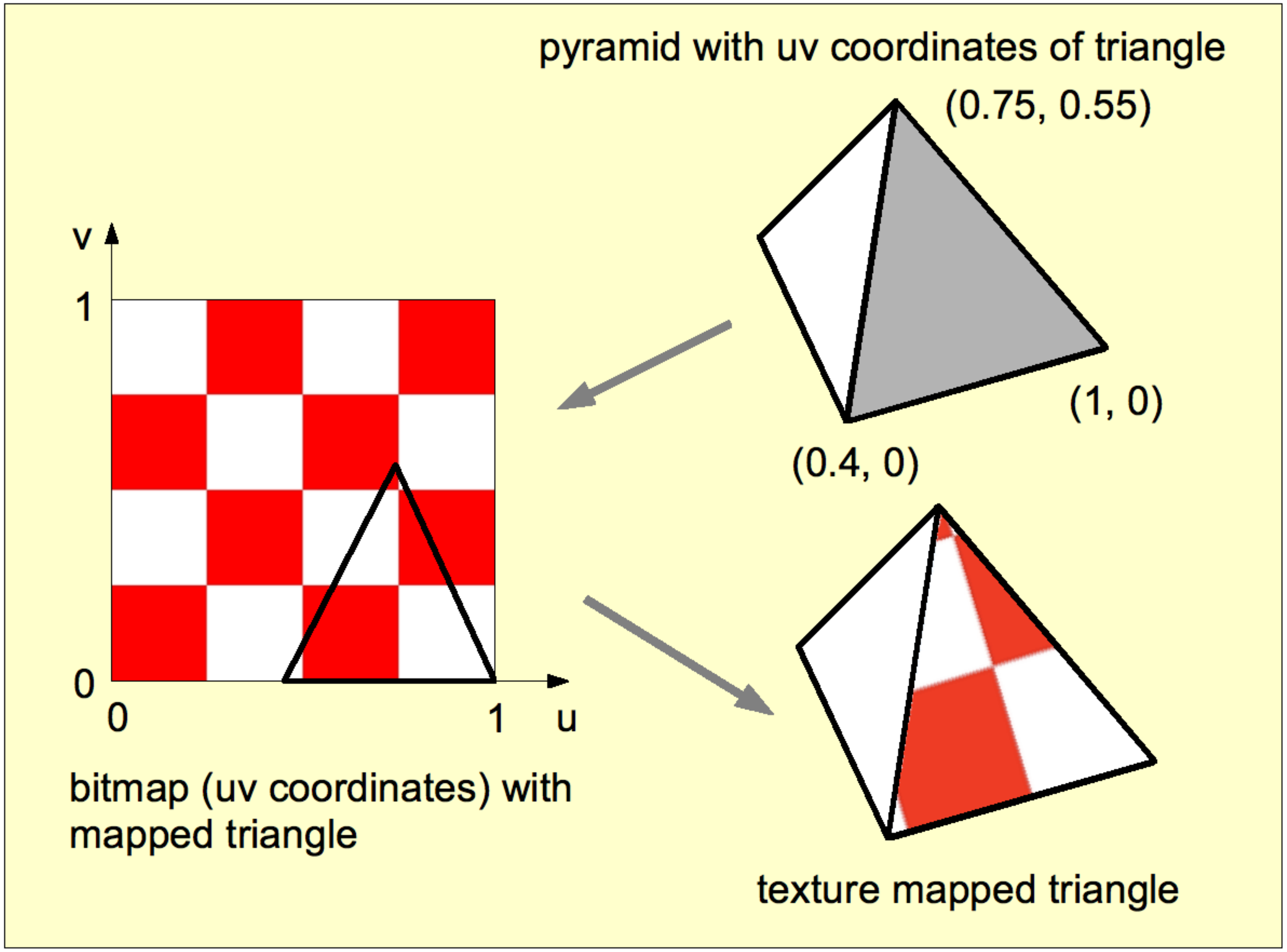
```
precision mediump float;

uniform vec3 ambientClr;
uniform vec3 diffuseClr;
uniform vec3 specularClr;
uniform float shininess;

varying vec3 normal;
varying vec3 litVec;
varying vec3 camPos;

-------------------------------------------------------------------


void main()
{
   vec3 norm = normalize(normal);
   gl_FragColor = vec4(ambientClr +
                       diffuse(norm) +
                       specular(norm), 1.0);
}
```

Directional Light

Fragment Shader continued

```

vec3 diffuse(vec3 norm)
{
   float scalar = dot(norm, litVec);
   return diffuseClr*max(scalar, 0.0);
}

vec3 specular(vec3 norm)
{
   vec3 lightRef = normalize(reflect(-litVec, norm));
   float scalar = dot(lightRef, normalize(camPos));
   if(scalar > 0.0)
      return specularClr*max(pow(scalar, shininess), 0.0);
   else
      return vec3(0.0, 0.0, 0.0);
}

```

Texture 2D


pyramid with uv coordinates of triangle
(0.75, 0.55)
(1, 0)
(0.4, 0)
v
1
0
0
1
u
bitmap (uv coordinates) with mapped triangle
texture mapped triangle

Texture 2D

Vertex Shader

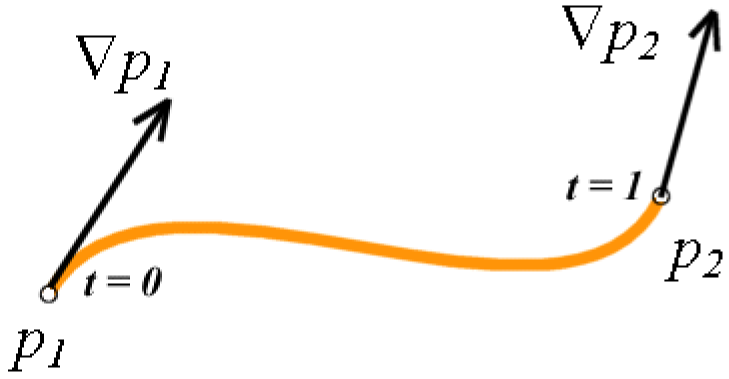
```
uniform mat4 objMat;
uniform mat4 viewMat;
uniform mat4 projMat;

attribute vec3 posAttr;
attribute vec2 txtAttr;

varying vec2 vTxt;

void main()
{
  gl_Position = projMat*viewMat*objMat*vec4(posAttr, 1.0);
  vTxt = txtAttr;
}
```

Texture 2D

Fragment Shader

```
precision mediump float;

varying vec2 vTxt;

uniform sampler2D txtSampler;

void main()
{
    gl_FragColor = texture2D(txtSampler, vTxt);
}
```

Texture Flatten



Hermite Specification

- Vertex manipulation between two positions.

Texture Flatten

Vertex Shader

```
uniform mat4 viewMat, projMat;
uniform float flatten, magnifier;

attribute vec3 posAttr, normAttr;
attribute vec2 txtAttr;

varying vec2 vTxt;

void main()
{
  mat4 hermite = mat4( 2.0, -3.0,  0.0,  1.0,
                      -2.0,  3.0,  0.0,  0.0,
                       1.0, -2.0,  1.0,  0.0,
                       1.0, -1.0,  0.0,  0.0);
  float flatten2 = flatten*flatten;
  float flatten3 = flatten2*flatten;
  vec4 iter = vec4(flatten3, flatten2, flatten, 1.0);
  float flatx = txtAttr.x*2.0-1.0;
  float flatz = txtAttr.y*2.0-1.0;
  mat4 pov = mat4(posAttr.x, flatx, normAttr.x*magnifier, 0.0,
                  posAttr.y, 0.0,   normAttr.y*magnifier, magnifier,
                  posAttr.z, flatz, normAttr.z*magnifier, 0.0,
                  1.0,       1.0,   0.0,                  0.0);
  vec4 final = iter*hermite*pov;
  gl_Position = projMat*viewMat*objMat*final;
  vTxt = txtAttr;
}
```

Texture Flatten

Vertex Shader

```
precision mediump float;

varying vec2 vTxt;

uniform sampler2D txtSampler;

void main()
{
    gl_FragColor = texture2D(txtSampler, vTxt);
}
```

SkyBox

Vertex Shader

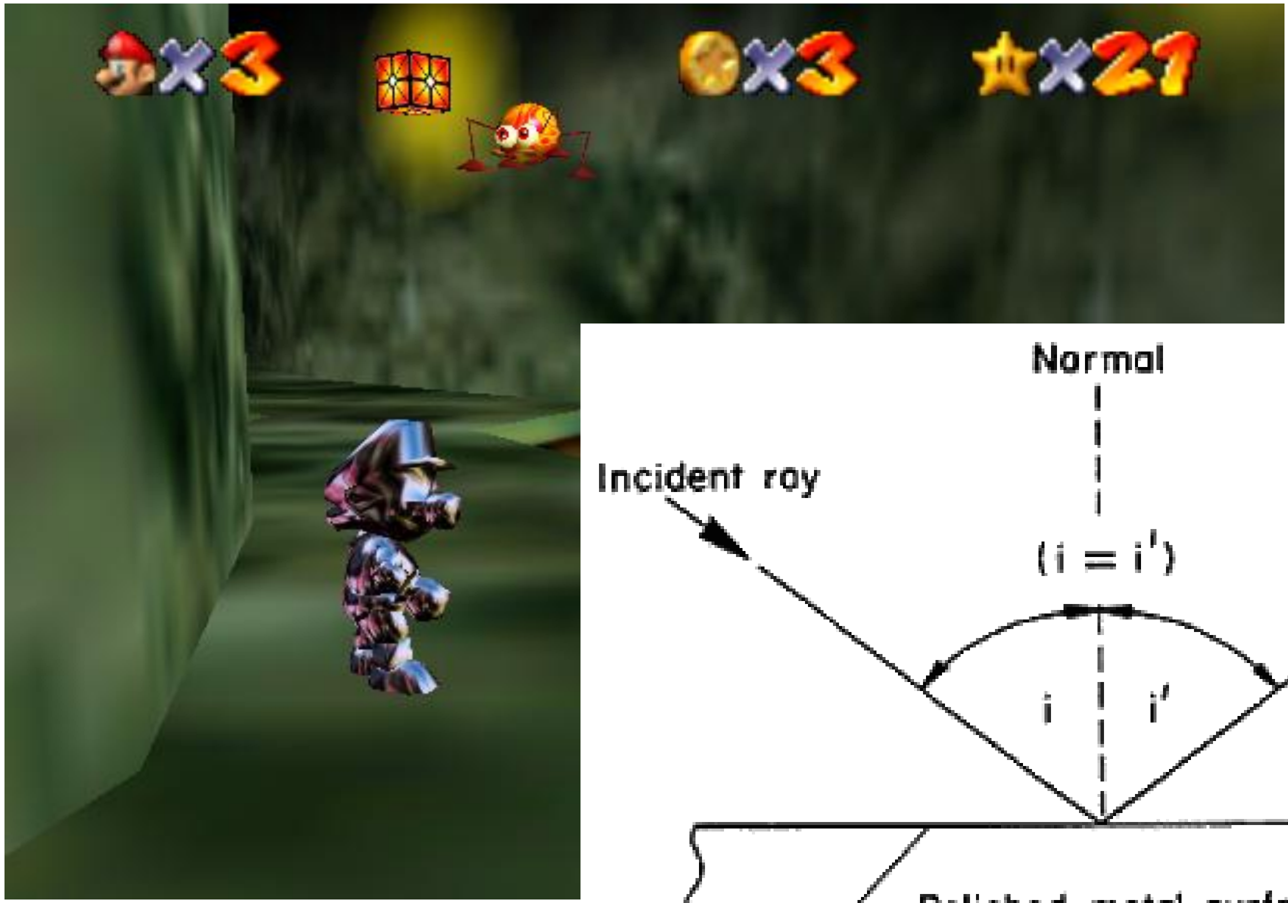
```
uniform mat4 objMat;
uniform mat4 viewMat;
uniform mat4 projMat;

attribute vec3 posAttr;
attribute vec3 cubeAttr;

varying vec3 vCube;

void main()
{
  gl_Position = projMat*viewMat*objMat*vec4(posAttr, 1.0);
  vCube = cubeAttr;
}
```

SkyBox

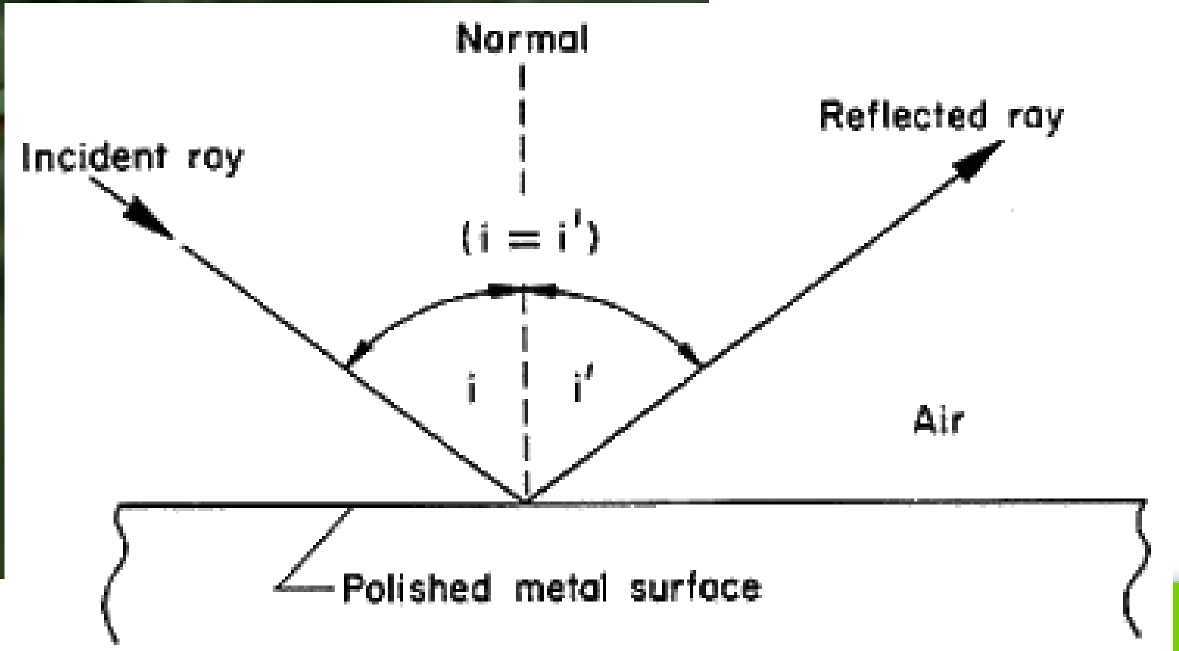
Fragment Shader

```
precision mediump float;

varying vec3 vCube;

uniform samplerCube txtSampler;

void main()
{
    gl_FragColor = textureCube(txtSampler, vCube);
}


```

Metal



×3
×3
×21



Normal
Reflected ray
Incident ray
(i = i')
i
i'
Air
Polished metal surface

Metal

Vertex Shader

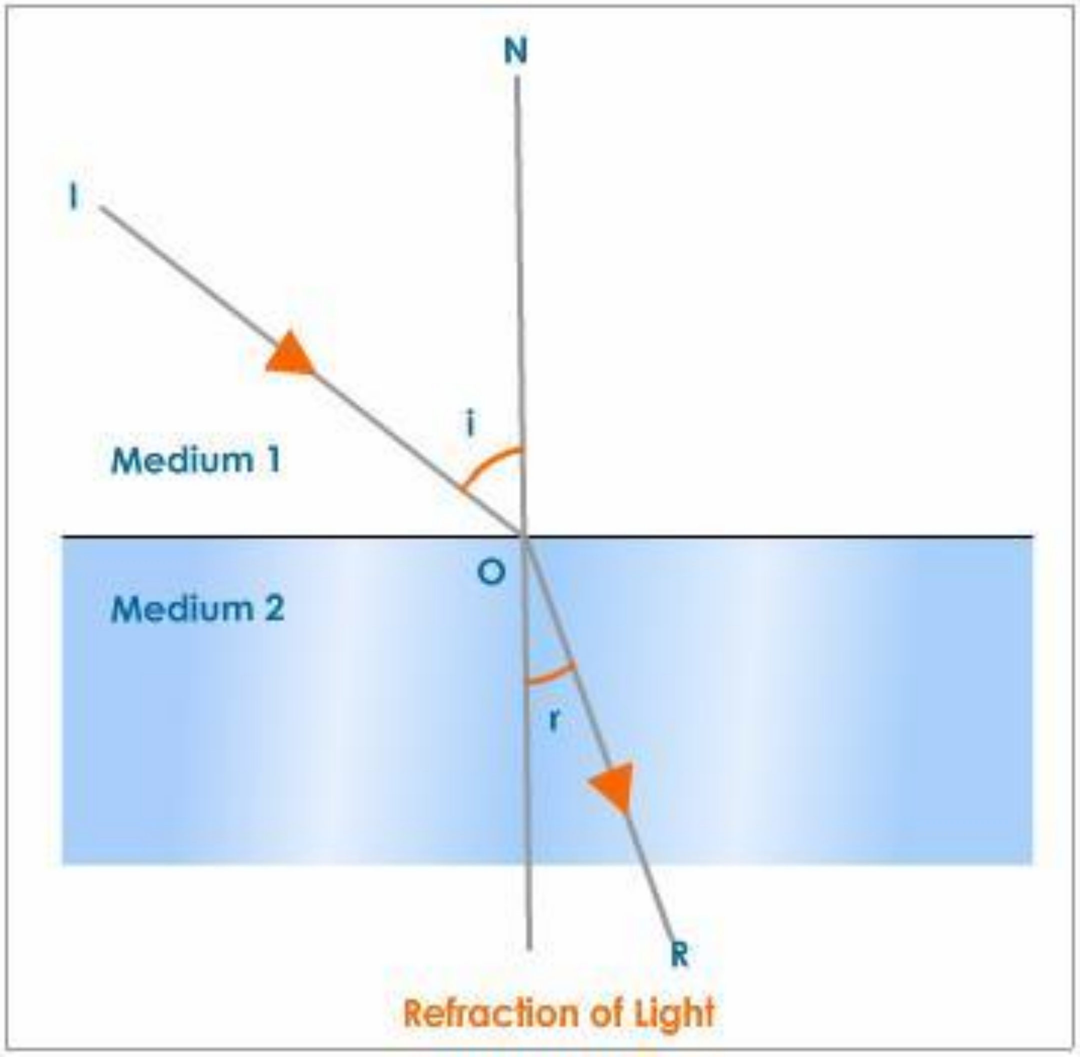
```
uniform mat4 objMat;
uniform mat4 viewMat;
uniform mat4 projMat;

attribute vec3 posAttr;
attribute vec3 normAttr;

varying vec3 vNorm;
varying vec3 vView;

void main()
{
  vec4 eyeCoords = viewMat*objMat*vec4(posAttr, 1.0);
  gl_Position = projMat*eyeCoords;
  vView = -vec3(eyeCoords);
  vNorm = vec3(viewMat*objMat*vec4(normAttr, 0.0));
}
```

Metal

Fragment Shader

```
precision mediump float;

uniform mat4 invViewMat;
uniform samplerCube txtSampler;

varying vec3 vNorm;
varying vec3 vView;

void main()
{
   vec3 refl = reflect(normalize(vView), normalize(vNorm));
   refl = vec3(invViewMat*vec4(refl, 0.0));
   gl_FragColor = textureCube(txtSampler, refl);
}
```

Glass



Glass

Vertex Shader

```
uniform mat4 objMat;
uniform mat4 viewMat;
uniform mat4 projMat;

attribute vec3 posAttr;
attribute vec3 normAttr;

varying vec3 vNorm;
varying vec3 vView;

void main()
{
  vec4 eyeCoords = viewMat*objMat*vec4(posAttr, 1.0);
  gl_Position = projMat*eyeCoords;
  vView = vec3(eyeCoords);
  vNorm = -vec3(viewMat*objMat*vec4(normAttr, 0.0));
}
```

Glass

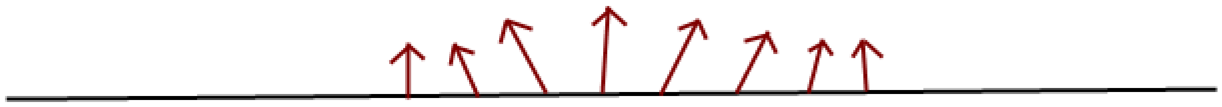
Fragment Shader

```
precision mediump float;

uniform mat4 invViewMat;
uniform samplerCube txtSampler;
uniform float reflWeight;

varying vec3 vNorm;
varying vec3 vView;

void main()
{
   vec3 refr = reflect(normalize(vView), normalize(vNorm));
   refr = mix(refr, -vView, reflWeight);
   refr = vec3(invViewMat*vec4(refr, 0.0));
   gl_FragColor = textureCube(txtSampler, refr);
}
```

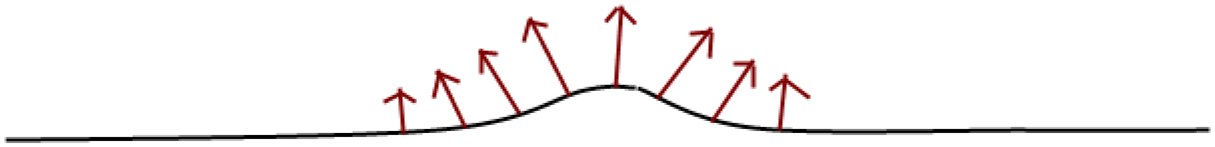
Bubble

- A mix of a vertex shader which “wobbles” the vertices and a fragment shader which combines reflection and refraction.

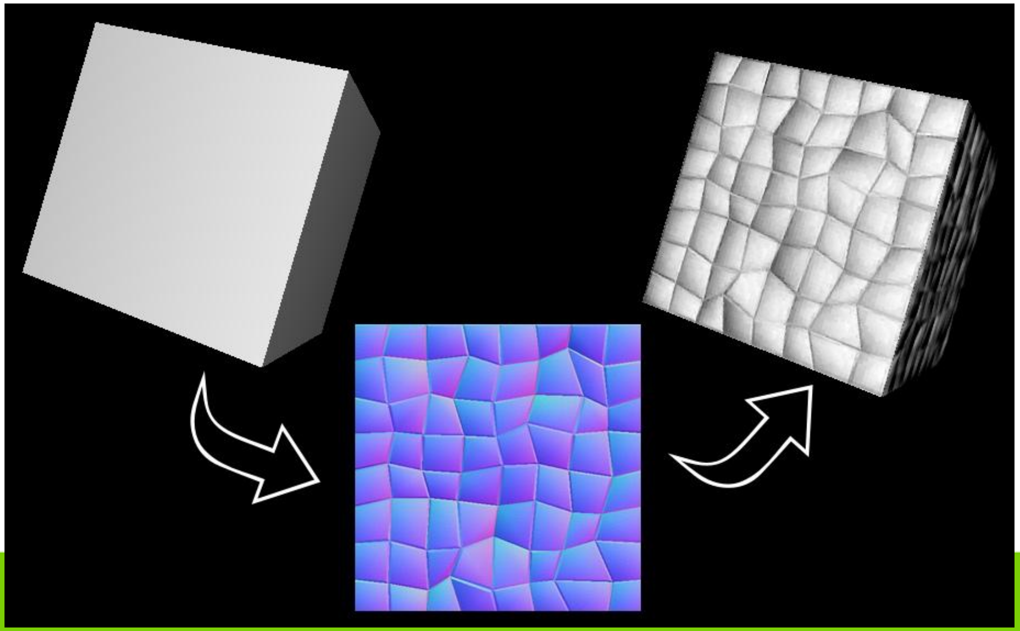
Bump Map



Approximation using bump mapping on a planar surface



Geometry of a bumpy surface



Bump Map

Vertex Shader

```
uniform mat4 objMat;
uniform mat4 viewMat;
uniform mat4 projMat;

attribute vec3 posAttr, normAttr, binmAttr, txtAttr;

varying vec3 vView, vNorm, vBinm;
varying vec2 vTxt;

void main()
{
  vec4 eyeCoords = viewMat*objMat*vec4(posAttr, 1.0);
  vView = -(eyeCoords.xyz);
  vNorm = (viewMat*objMat*vec4(normAttr, 0.0)).xyz;
  vBinm = (viewMat*objMat*vec4(binmAttr, 0.0)).xyz;
  vTxt  = txtAttr;
  gl_Position = projMat*eyeCoords;
}
```

Bump Map

Fragment Shader

```
precision mediump float;

uniform mat4 invViewMat;
uniform samplerCube cubeSampler;
uniform sampler2D bumpSampler;

varying vec3 vView, vNorm, vBinm;
varying vec2 vTxt;

void main()
{
   vec3 n = normalize(vNorm);
   vec3 b = normalize(vBinm);
   vec3 c = -cross(n, b);
   b = -cross(c, n);
   mat3 m = mat3(c.x, c.y, c.z,
                 n.x, n.y, n.z,
                 b.x, b.y, b.z);

   vec3 bump = texture2D(bumpSampler, vTxt).rbg;
   bump = m * normalize(2.0*bump - 1.0);
   vec3 refl = reflect(normalize(vView), bump);
   refl = vec3(invViewMat*vec4(refl, 0.0));
   gl_FragColor = textureCube(cubeSampler, refl);
}
```

Height Map

Vertex Shader

```
uniform mat4 objMat, viewMat, projMat;
uniform float maxHeight;

attribute vec3 posAttr;
attribute vec3 normAttr;
attribute vec2 txtAttr;

varying vec4 color;

uniform sampler2D txtSampler;

void main()
{
  color = texture2D(txtSampler, txtAttr);
  float height = (color.x + color.y + color.z)/3.0;
  vec4 pos = height*vec4(normAttr, 0.0)*maxHeight
             + vec4(posAttr, 1.0);
  gl_Position = projMat*viewMat*objMat*pos;
}
```

Height Map

Fragment Shader

```
precision mediump float;

varying vec4 color;

void main()
{
   gl_FragColor = color;
}
```

Red/Blue Image

- A multi-pass shader.

Red/Blue Image

Vertex Shader

```
uniform float dx;
uniform float dv;
uniform float width;
uniform float height;
uniform float aspect;

attribute vec3 posAttr;
attribute vec2 txtAttr;

varying vec2 vTxt;

void main()
{
  gl_Position = vec4((posAttr.x*width + dx)*aspect,
                     posAttr.y*height, posAttr.z, 1.0);
  vTxt = vec2(txtAttr.x*0.5 + dv, txtAttr.y);
}
```

Red/Blue Image

Fragment Shader

```
precision mediump float;

uniform vec3 color;

varying vec2 vTxt;

uniform sampler2D txtSampler;

void main()
{
   gl_FragColor = texture2D(txtSampler, vTxt) * vec4(color, 1.0);
}
```

Cartoon

Cartoon - Filler

Vertex Shader

```
uniform mat4 objMat;
uniform mat4 viewMat;
uniform mat4 projMat;
uniform vec3 lightVec;

attribute vec3 posAttr;
attribute vec3 normAttr;

varying vec3 normal;
varying vec3 litVec;

void main()
{
  normal = normalize(objMat*vec4(normAttr, 0.0)).xyz;
  litVec = normalize((viewMat*vec4(lightVec, 0.0)).xyz);
  gl_Position = projMat*viewMat*objMat*vec4(posAttr, 1.0);
}
```

Cartoon - Filler

Fragment Shader

```
precision mediump float;

uniform float ambient;
uniform float diffuse;
uniform vec3 lightClr;
uniform vec3 darkClr;
uniform float slices;

varying vec3 normal;
varying vec3 litVec;

void main()
{
   vec3 norm = normalize(normal);
   float diff = diffuse*max(dot(norm, litVec), 0.0);
   float shade = 1.0 - clamp(ambient + diff, 0.0, 1.0);
   shade = floor(shade * (slices+1.0)) / (slices+1.0);
   gl_FragColor = vec4(mix(lightClr, darkClr, shade), 1.0);
}
```

Cartoon - Outline

Vertex Shader

```
uniform mat4 objMat, viewMat, projMat;
uniform float thickness, edgeLimit;

attribute vec3 posAttr, normAttr;
attribute vec3 adj1Attr, adj2Attr;
attribute float wghtAttr;

void main()
{
  mat4 viewObjMat = viewMat*objMat;
  if (wghtAttr < 0.5) {
    gl_Position = projMat*viewObjMat*vec4(posAttr, 1.0);
  } else {
    vec4 pos       = viewObjMat*vec4(posAttr,  1.0);
    vec4 norm      = viewObjMat*vec4(normAttr, 0.0);
    vec4 faceNorm1 = viewObjMat*vec4(adj1Attr, 0.0);
    vec4 faceNorm2 = viewObjMat*vec4(adj2Attr, 0.0);

    if (dot(faceNorm1, faceNorm2) <= edgeLimit) {
      pos += norm*thickness;
    } else {
      float dot1 = dot(pos, faceNorm1);
      float dot2 = dot(pos, faceNorm2);
      if ((dot1 * dot2) < 0.0) {
        pos += norm*thickness;
      }
    }
    gl_Position = projMat*pos;
  }
}
```

Cartoon - Outline

Fragment Shader

```
precision mediump float;

uniform vec3 color;

void main()
{
   gl_FragColor = vec4(color, 1.0);
}
```

Fireflies

- Multiple point lights via multiple passes, multiple shapes, and bump mapping with textures and diffuse lighting.

Questions