Introduction to Computer Graphics with WebGL

Ed Angel

Professor Emeritus of Computer Science

Founding Director, Arts, Research, Technology and Science Laboratory

University of New Mexico

Reflection and Environment Maps

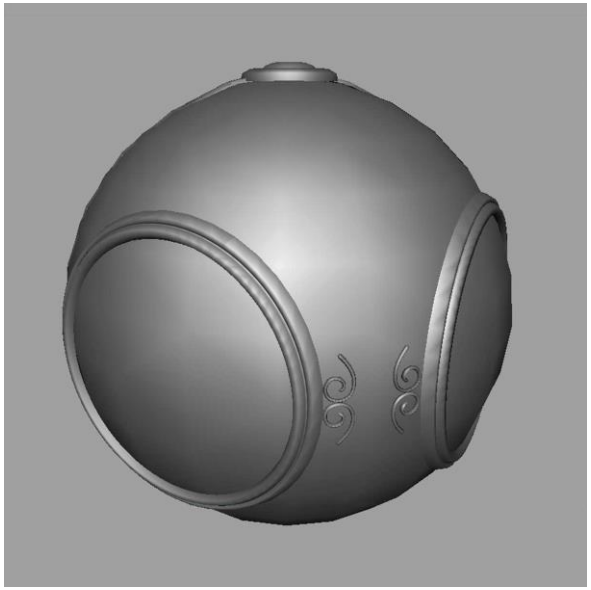
Ed Angel
Professor Emeritus of Computer Science
University of New Mexico

Objectives

- Texture Mapping Applications
- Reflection (Environment) Maps
  - Cube Maps
  - Spherical Maps
- Bump Maps

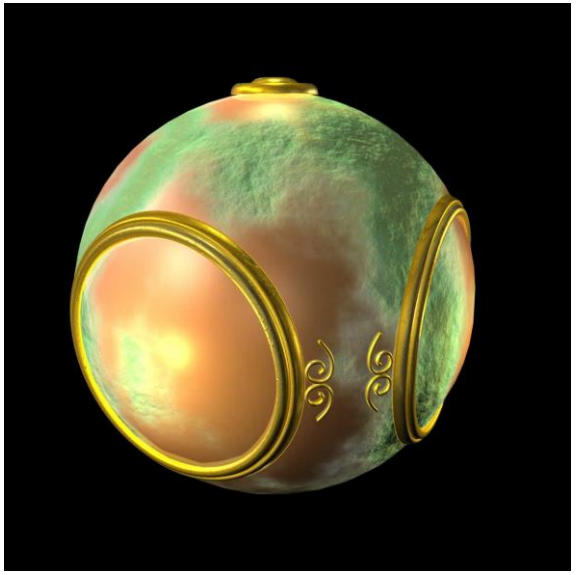
The University of New Mexico

Mapping Variations



smooth shading

environment mapping

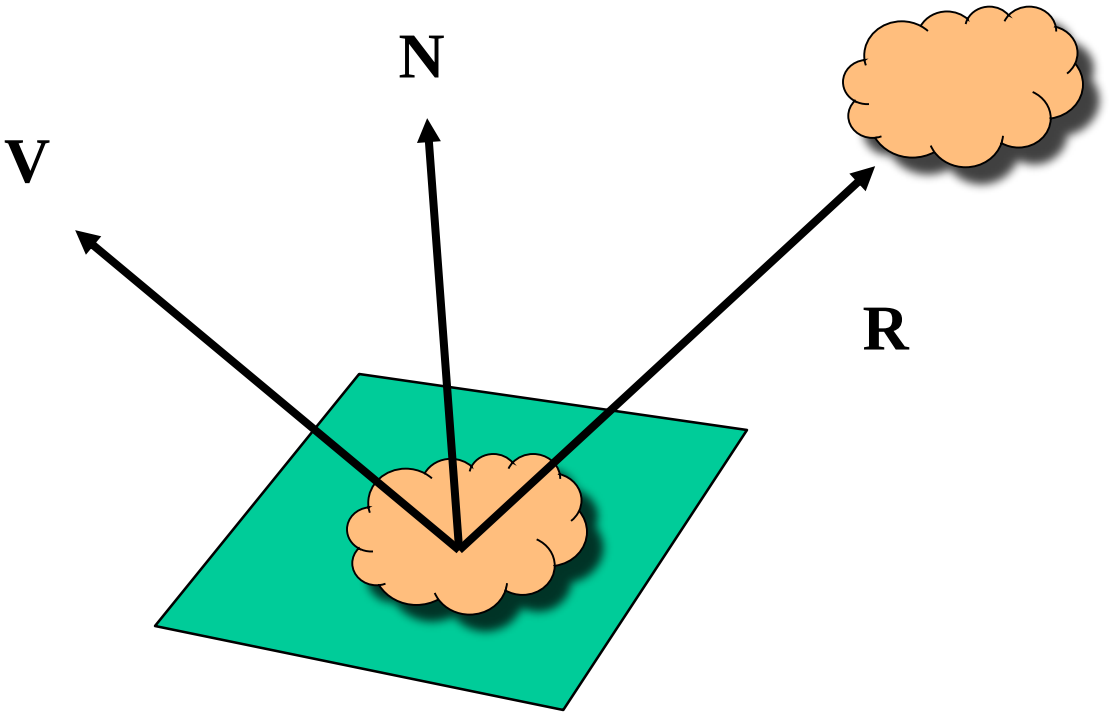


bump mapping

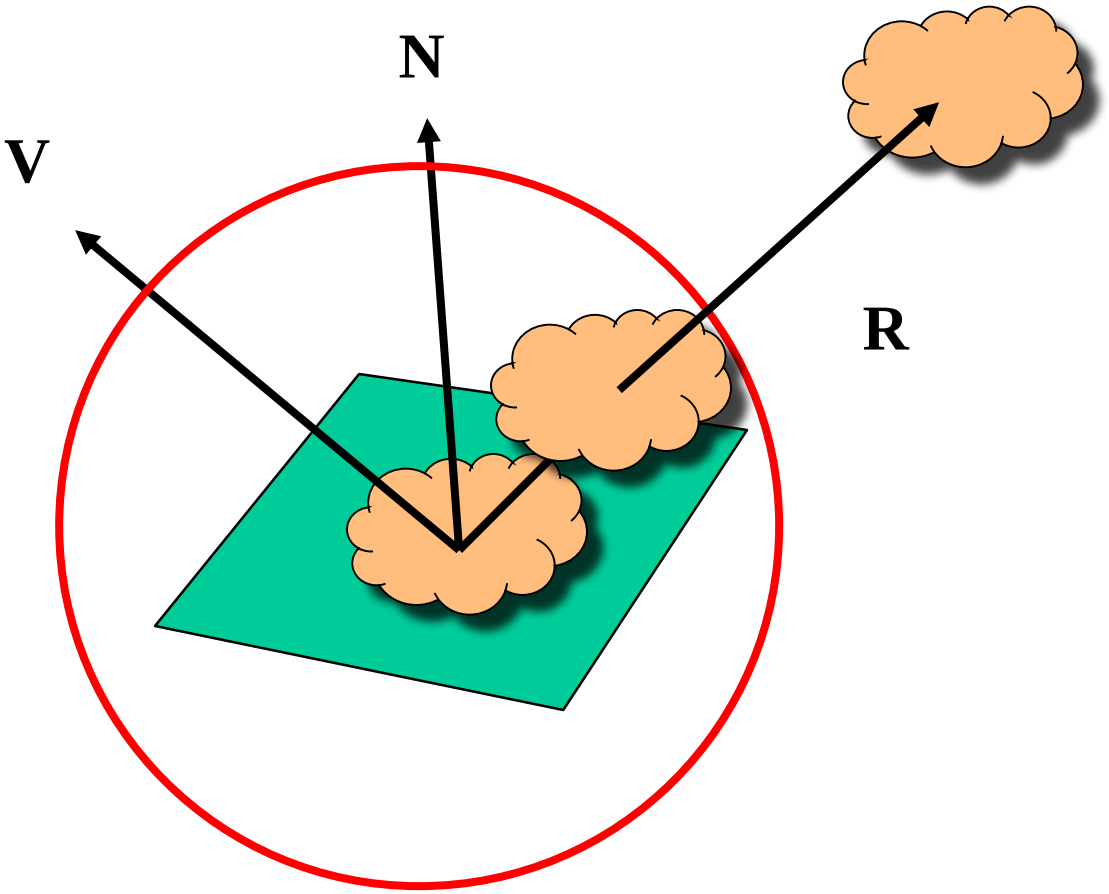
Environment Mapping

- Environmental (reflection) mapping is way to create the appearance of highly reflective surfaces without ray tracing which requires global calculations
- Introduced in movies such as The Abyss and Terminator 2
- Prevalent in video games
- It is a form of texture mapping

Reflecting the Environment



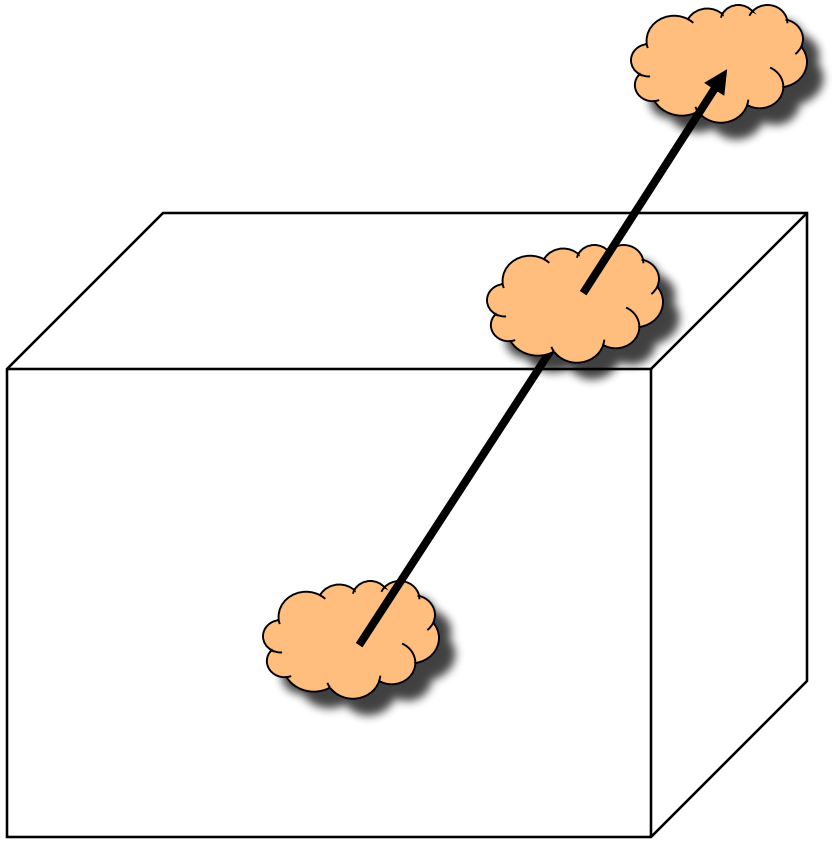
Mapping to a Sphere



Hemisphere Map as a Texture

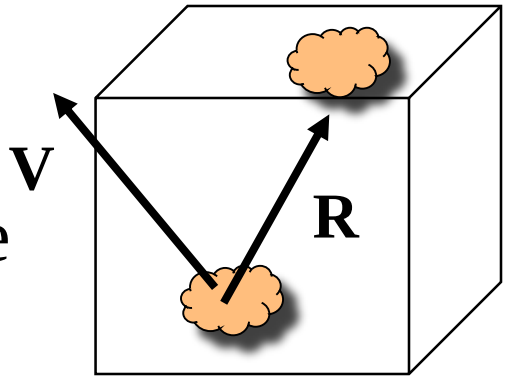
- If we map all objects to hemisphere, we cannot tell if they are on the sphere or anywhere else along the reflector
- Use the map on the sphere as a texture that can be mapped onto the object
- Can use other surfaces as the intermediate
  - Cube maps
  - Cylinder maps

Cube Map



Indexing into Cube Map

- Compute $\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{V})\mathbf{N} - \mathbf{V}$
- Object at origin
- Use largest magnitude component of R to determine face of cube
- Other two components give texture coordinates



OpenGL Implementation

- WebGL supports only cube maps
  - desktop OpenGL also supports sphere maps
- First must form map
  - Use images from a real camera
  - Form images with WebGL
- Texture map it to object

Cube Maps

- We can form a cube map texture by defining six 2D texture maps that correspond to the sides of a box
- Supported by WebGL through cubemap sampler

  vec4 texColor = textureCube(mycube, texcoord);
- Texture coordinates must be 3D

  - usually are given by the vertex location so we don't need compute separate tex coords

Environment Maps with Shaders

- Environment maps are usually computed in world coordinates which can differ from object coordinates because of the modeling matrix
  - May have to keep track of modeling matrix and pass it to the shaders as a uniform variable
- Can also use reflection map or refraction map for effects such as simulating water

The University of New Mexico

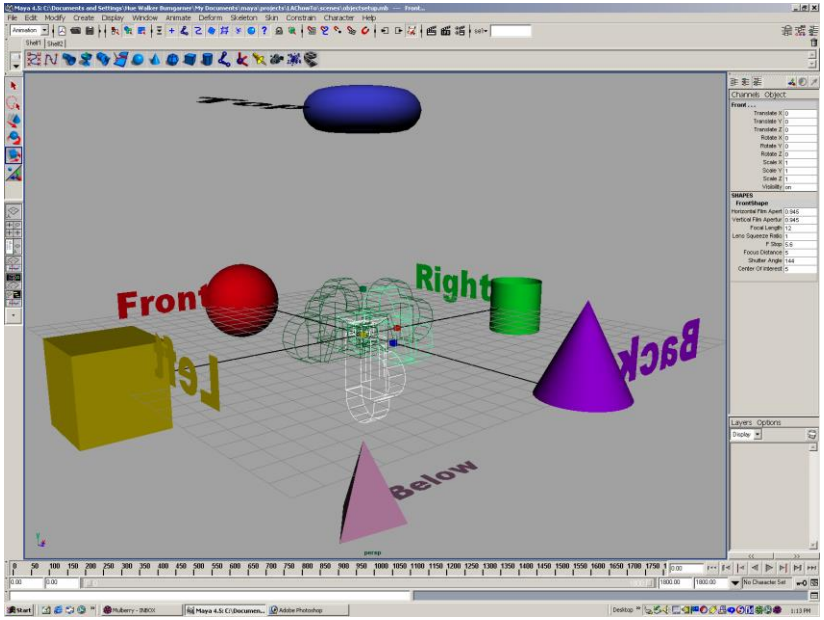
Issues

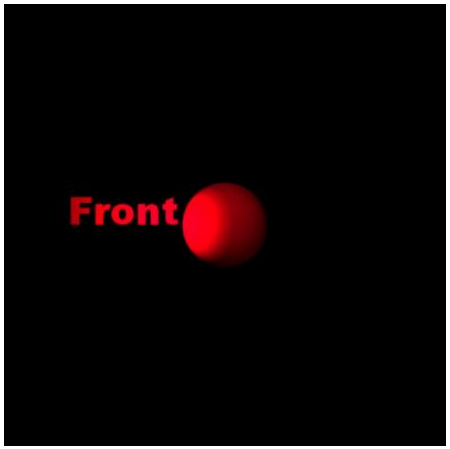
- Must assume environment is very far from object (equivalent to the difference between near and distant lights)
- Object cannot be concave (no self reflections possible)
- No reflections between objects
- Need a reflection map for each object
- Need a new map if viewer moves

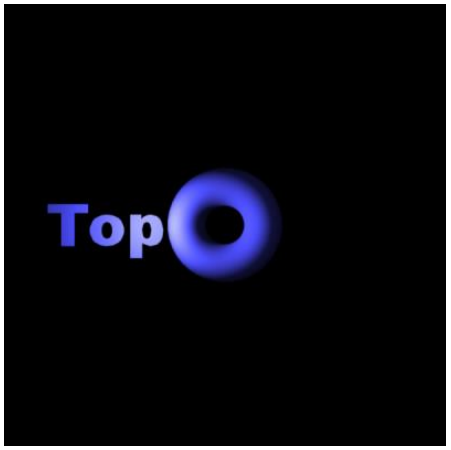
The University of New Mexico

Forming Cube Map

- Use six cameras, each with a 90 degree angle of view

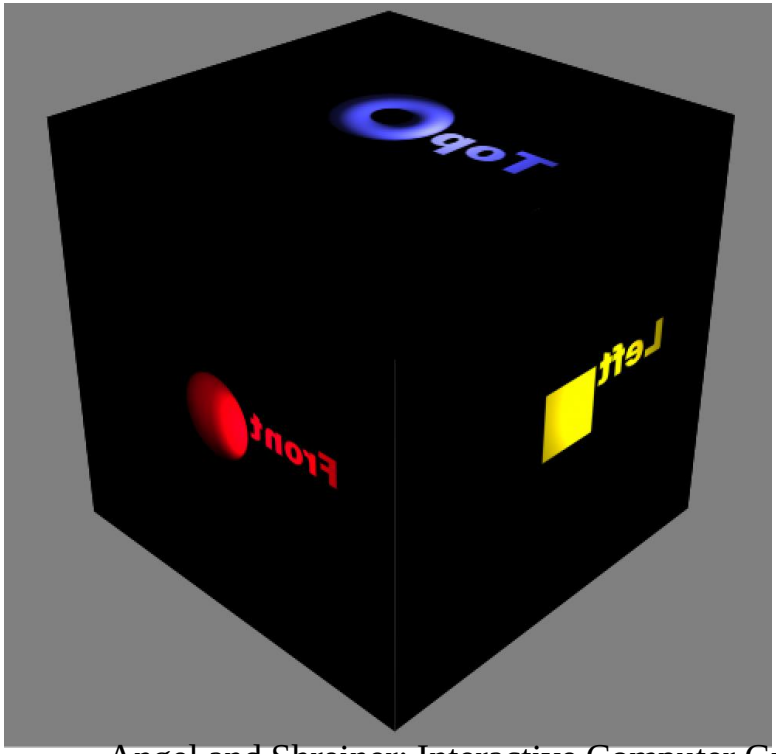


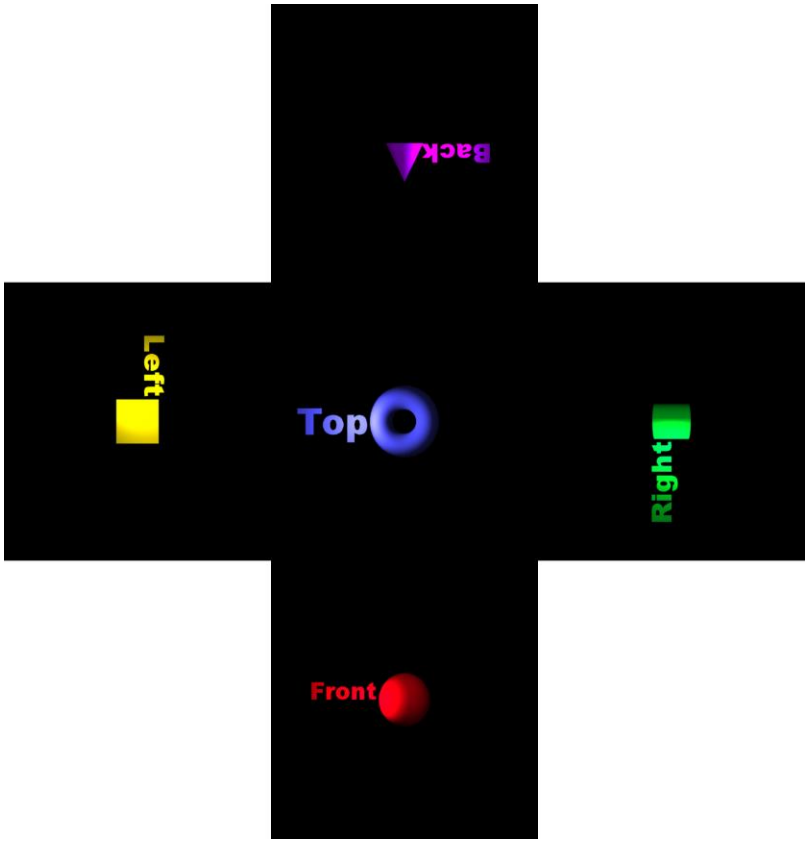




The University of New Mexico

vs Cube Image





Doing it in WebGL

```
gl.textureMap2D(
    gl.TEXTURE_CUBE_MAP_POSITIVE_X,
    level, rows, columns, border, gl.RGBA,
    gl.UNSIGNED_BYTE, image1)
```

- Same for other five images
- Make one texture object out of the six images

Example

- Consider rotating cube inside a cube that reflects the color of the walls
- Each wall is a solid color (red, green, blue, cyan, magenta, yellow)
  - Each face of room can be a texture of one texel

```
var red = new Uint8Array([255, 0, 0, 255]);
var green = new Uint8Array([0, 255, 0, 255]);
var blue = new Uint8Array([0, 0, 255, 255]);
var cyan = new Uint8Array([0, 255, 255, 255]);
var magenta = new Uint8Array([255, 0, 255, 255]);
var yellow = new Uint8Array([255, 255, 0, 255]);
```

Texture Object

```
cubeMap = gl.createTexture();
gl.bindTexture(gl.TEXTURE_CUBE_MAP, cubeMap);
gl.texImage2D(gl.TEXTURE_CUBE_MAP_POSITIVE_X, 0, gl.RGBA,
      1, 1, 0, gl.RGBA,gl.UNSIGNED_BYTE, red);
gl.texImage2D(gl.TEXTURE_CUBE_MAP_NEGATIVE_X, 0, gl.RGBA,
      1, 1, 0, gl.RGBA,gl.UNSIGNED_BYTE, green);
 gl.texImage2D(gl.TEXTURE_CUBE_MAP_POSITIVE_Y, 0, gl.RGBA,
      1, 1, 0, gl.RGBA,gl.UNSIGNED_BYTE, blue);
 gl.texImage2D(gl.TEXTURE_CUBE_MAP_NEGATIVE_Y, 0, gl.RGBA,
      1, 1, 0, gl.RGBA,gl.UNSIGNED_BYTE, cyan);
 gl.texImage2D(gl.TEXTURE_CUBE_MAP_POSITIVE_Z, 0, gl.RGBA,
      1, 1, 0, gl.RGBA,gl.UNSIGNED_BYTE, yellow);
  gl.texImage2D(gl.TEXTURE_CUBE_MAP_NEGATIVE_Z, 0, gl.RGBA,
      1, 1, 0, gl.RGBA,gl.UNSIGNED_BYTE, magenta);
  gl.activeTexture( gl.TEXTURE0 );
  gl.uniform1i(gl.getUniformLocation(program, "texMap"),0);
```

Vertex Shader

```
varying vec3 R;
attribute vec4 vPosition;
attribute vec4 vNormal;
uniform mat4 modelViewMatrix;
uniform mat4 projectionMatrix;
uniform vec3 theta;
void main(){
    vec3 angles = radians( theta );
    // compute rotation matrices rx, ry, rz here
    mat4 ModelViewMatrix = modelViewMatrix*rz*ry*rx;
   gl_Position = projectionMatrix*ModelViewMatrix*vPosition;
   vec4 eyePos  = ModelViewMatrix*vPosition;
   vec4 N = ModelViewMatrix*vNormal;
    R = reflect(eyePos.xyz, N.xyz);   }
```

Fragment Shader

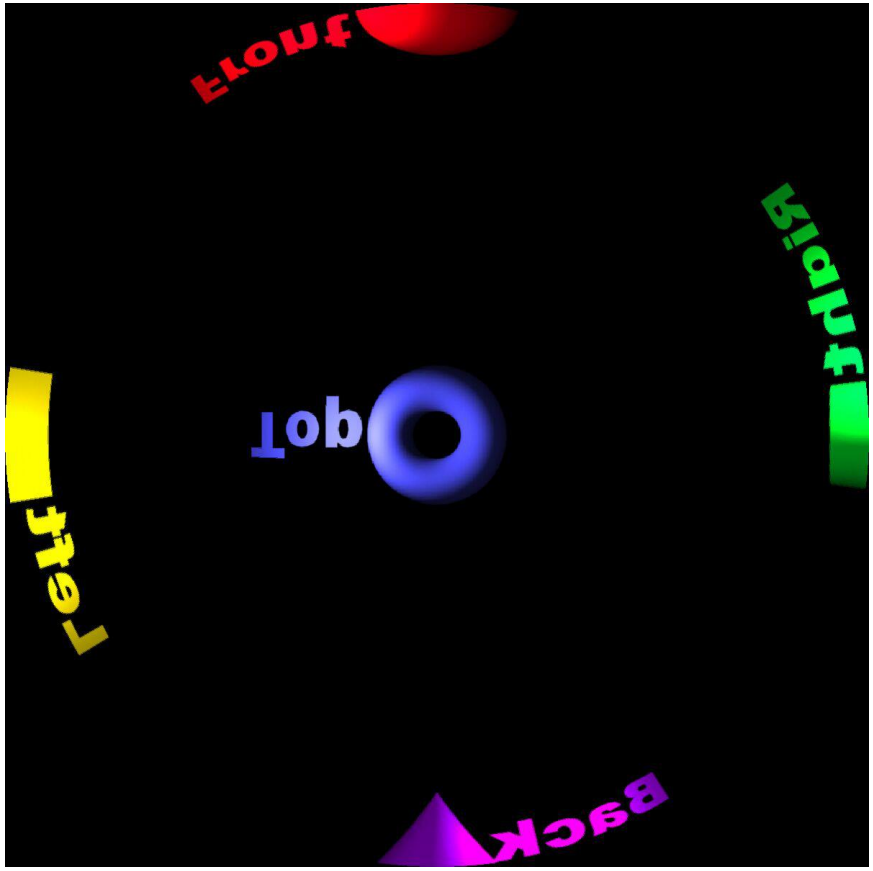
```
precision mediump float;

varying vec3 R;
uniform samplerCube texMap;

void main()
{
    vec4 texColor = textureCube(texMap, R);
    gl_FragColor = texColor;
}
```

Sphere Mapping

- Original environmental mapping technique proposed by Blinn and Newell based in using lines of longitude and latitude to map parametric variables to texture coordinates
- OpenGL supports sphere mapping which requires a circular texture map equivalent to an image taken with a fisheye lens

The University of New Mexico

Sphere Map



Introduction to Computer Graphics with WebGL

Ed Angel

Professor Emeritus of Computer Science

Founding Director, Arts, Research, Technology and Science Laboratory

University of New Mexico

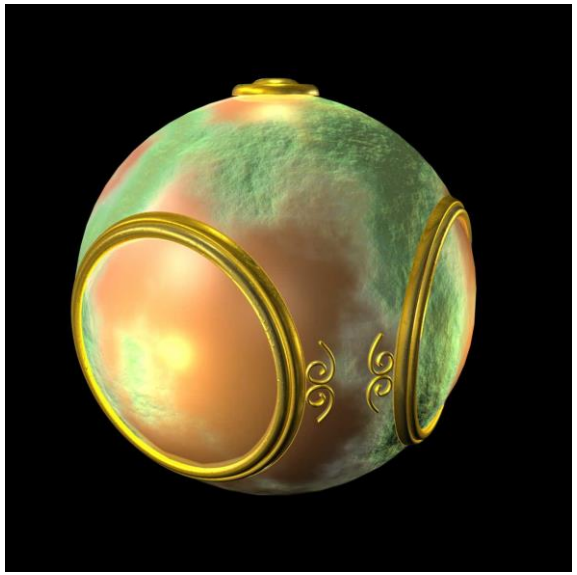
Bump Maps

Ed Angel
Professor Emeritus of Computer Science
University of New Mexico

The University of New Mexico

Objectives

- Introduce bump mapping

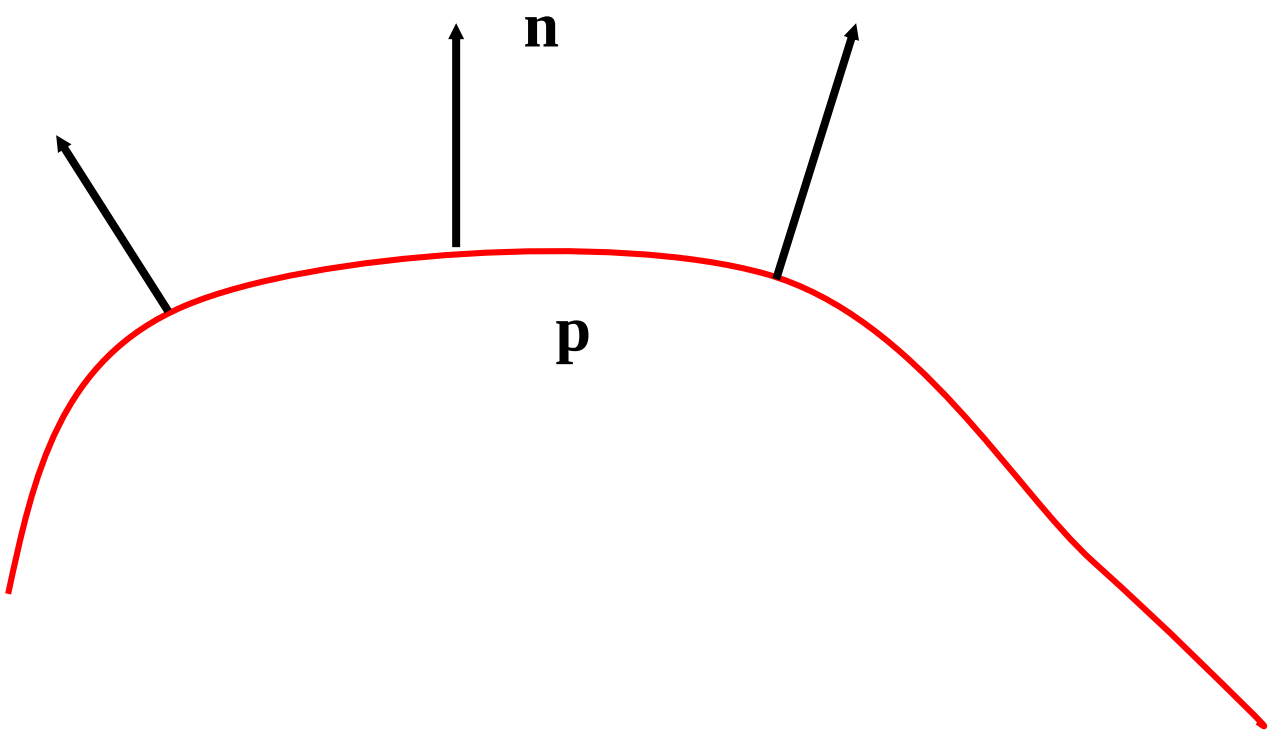


Modeling an Orange

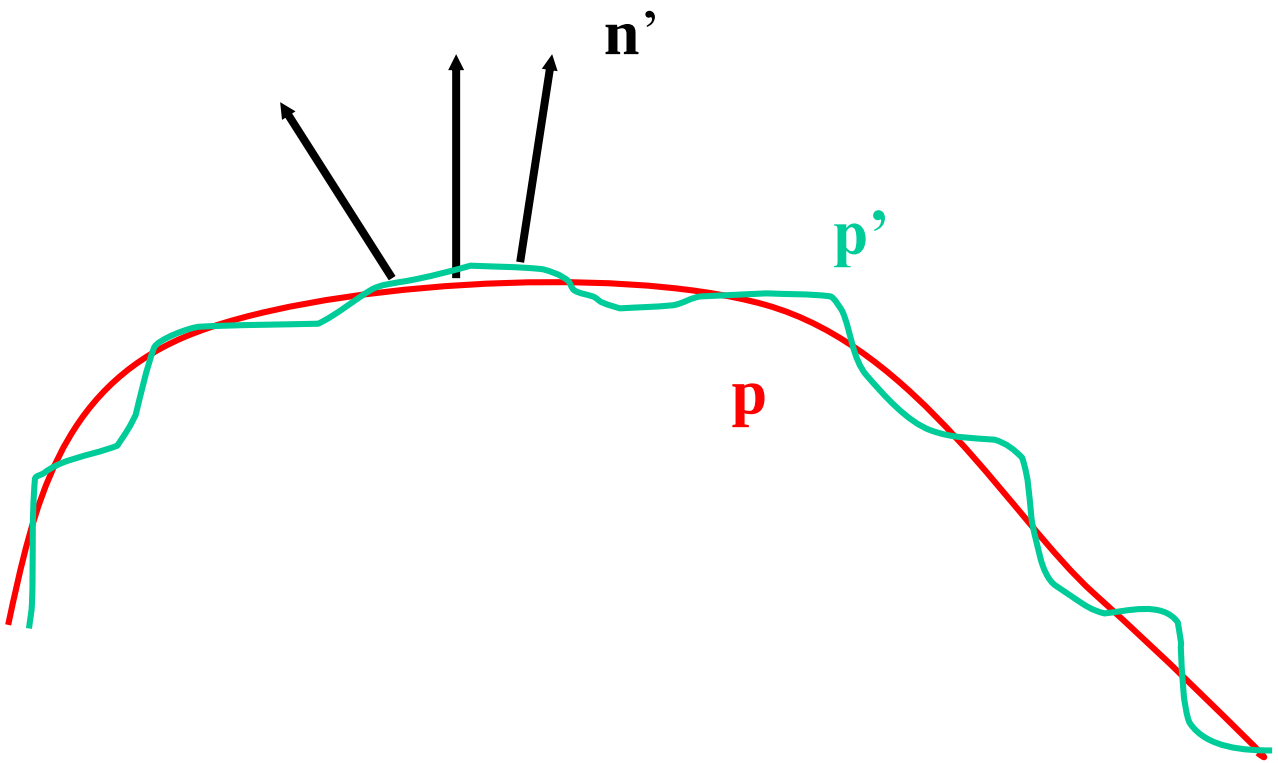
- Consider modeling an orange
- Texture map a photo of an orange onto a surface
  - Captures dimples
  - Will not be correct if we move viewer or light
  - We have shades of dimples rather than their correct orientation
- Ideally we need to perturb normal across surface of object and compute a new color at each interior point

Bump Mapping (Blinn)

- Consider a smooth surface

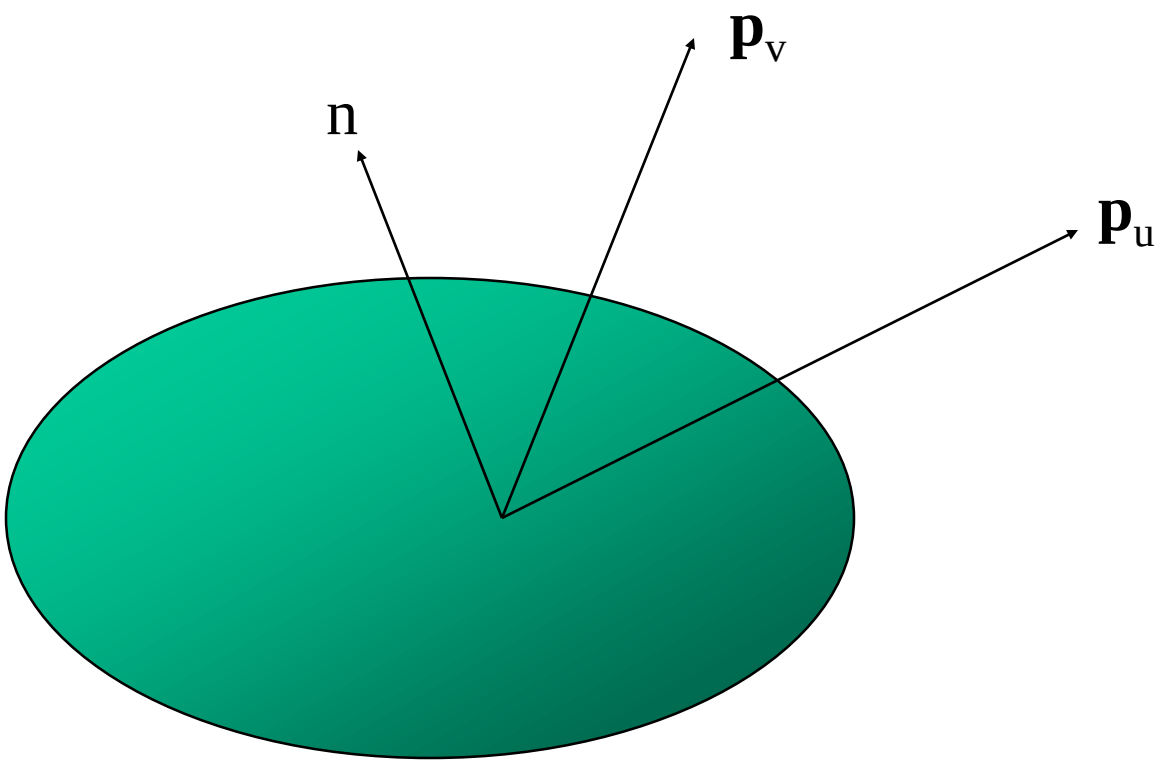


Rougher Version



The University of New Mexico

Tangent Plane



Equations

$$\mathbf{p}(u,v) = [x(u,v), y(u,v), z(u,v)]^T$$

$$\mathbf{p}_u = [\partial x/\partial u, \partial y/\partial u, \partial z/\partial u]^T$$

$$\mathbf{p}_v = [\partial x/\partial v, \partial y/\partial v, \partial z/\partial v]^T$$

$$\mathbf{n} = (\mathbf{p}_u \times \mathbf{p}_v) / |\mathbf{p}_u \times \mathbf{p}_v|$$

Displacement Function

$$\mathbf{p}' = \mathbf{p} + d(u,v)\,\mathbf{n}$$

d(u,v) is the bump or displacement function

$|d(u,v)| << 1$

Perturbed Normal

$$\mathbf{n}' = \mathbf{p}'_u \times \mathbf{p}'_v$$

$$\mathbf{p}'_u = \mathbf{p}_u + (\partial d/\partial u)\mathbf{n} + d(u,v)\mathbf{n}_u$$

$$\mathbf{p}'_v = \mathbf{p}_v + (\partial d/\partial v)\mathbf{n} + d(u,v)\mathbf{n}_v$$

If d is small, we can neglect last term

Approximating the Normal

$$\mathbf{n}' = \mathbf{p}'_u \times \mathbf{p}'_v$$

$$\approx \mathbf{n} + (\partial d/\partial u)\mathbf{n} \times \mathbf{p}_v + (\partial d/\partial v)\mathbf{n} \times \mathbf{p}_u$$

The vectors $\mathbf{n} \times \mathbf{p}_v$ and $\mathbf{n} \times \mathbf{p}_u$ lie in the tangent plane

Hence the normal is displaced in the tangent plane

Must precompute the arrays $\partial d/\partial u$ and $\partial d/\partial v$

Finally,we perturb the normal during shading

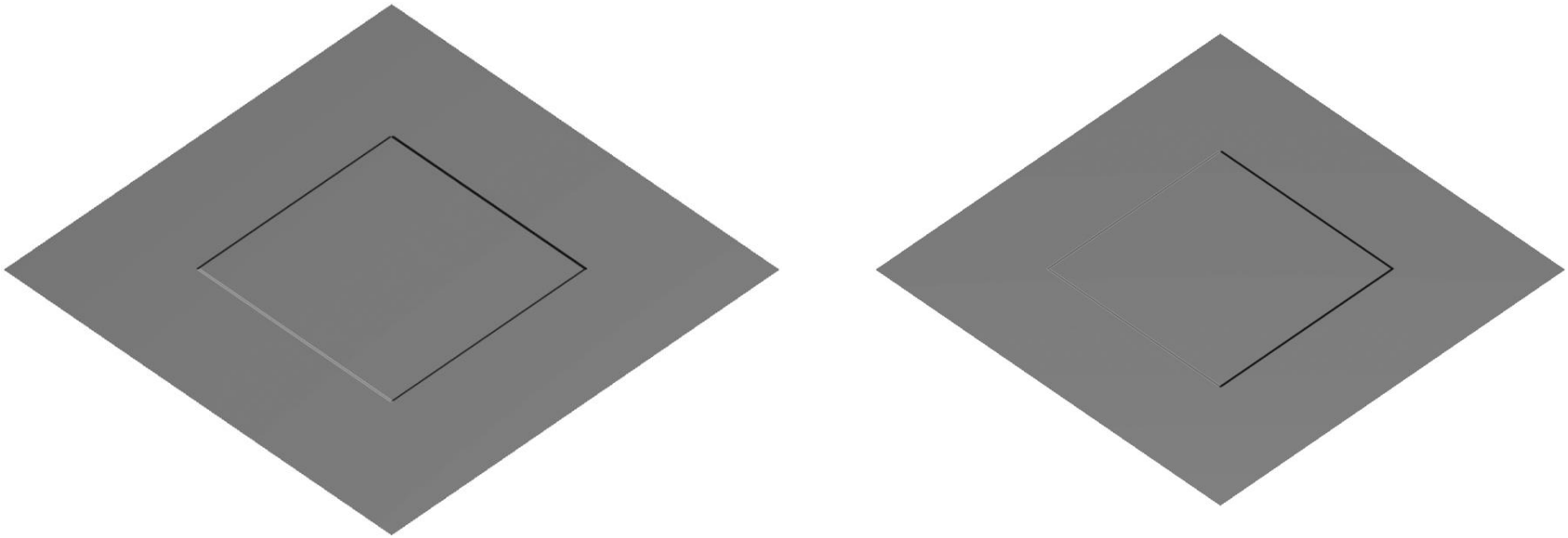
Image Processing

- Suppose that we start with a function d(u,v)
- We can sample it to form an array $D=[d_{ij}]$
- Then $\partial d/\partial u \approx d_{ij} - d_{i-1,j}$

  and $\partial d/\partial v \approx d_{ij} - d_{i,j-1}$
- **Embossing**: multipass approach using floating point buffer

Example

Single Polygon and a Rotating Light Source



How to do this?

- The problem is that we want to apply the perturbation at all points on the surface
- Cannot solve by vertex lighting (unless polygons are very small)
- Really want to apply to every fragment
- Can’t do that in fixed function pipeline
- But can do with a fragment program!!
- See bumpmap.html and bumpmap.js