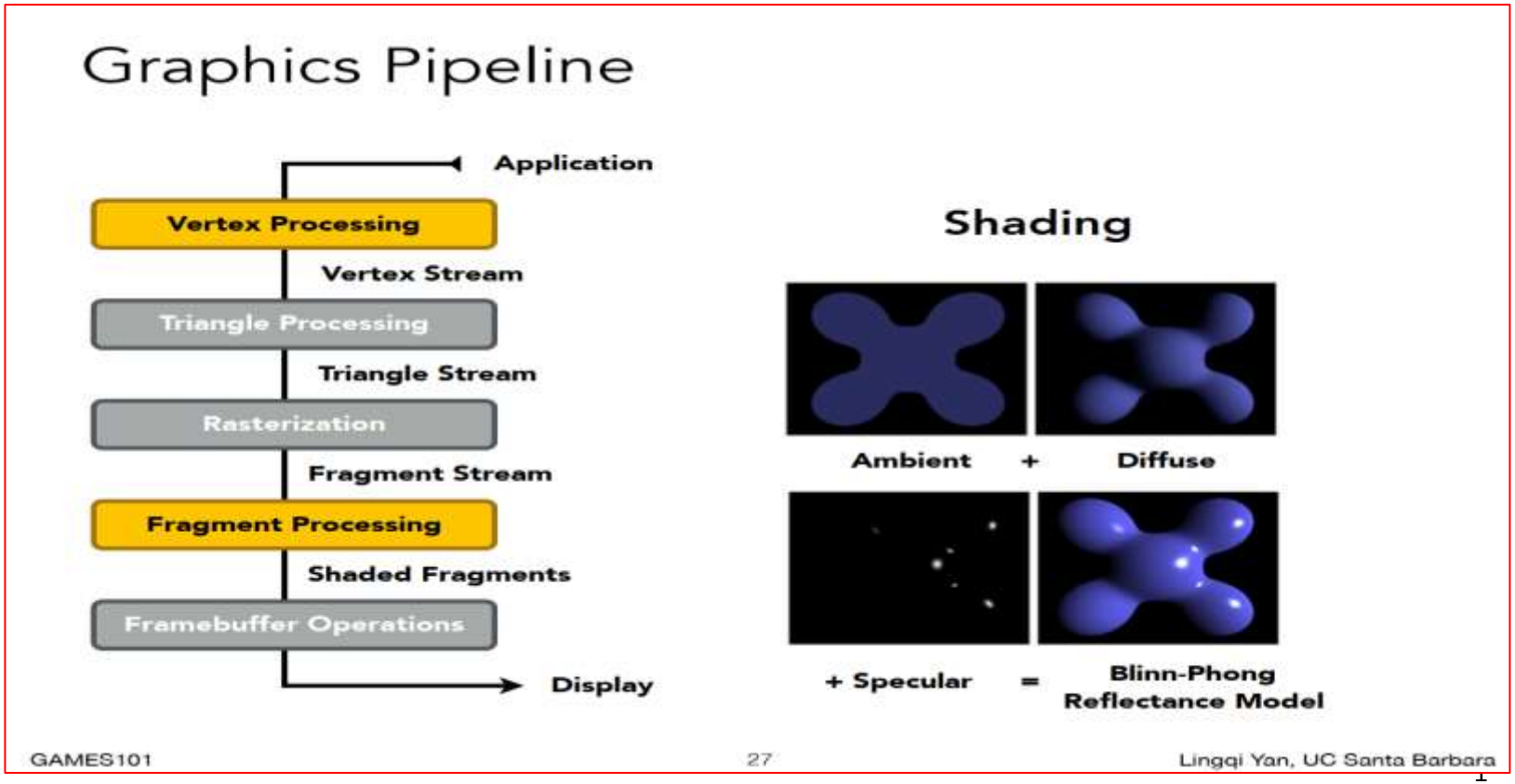
The University of New Mexico

Recap

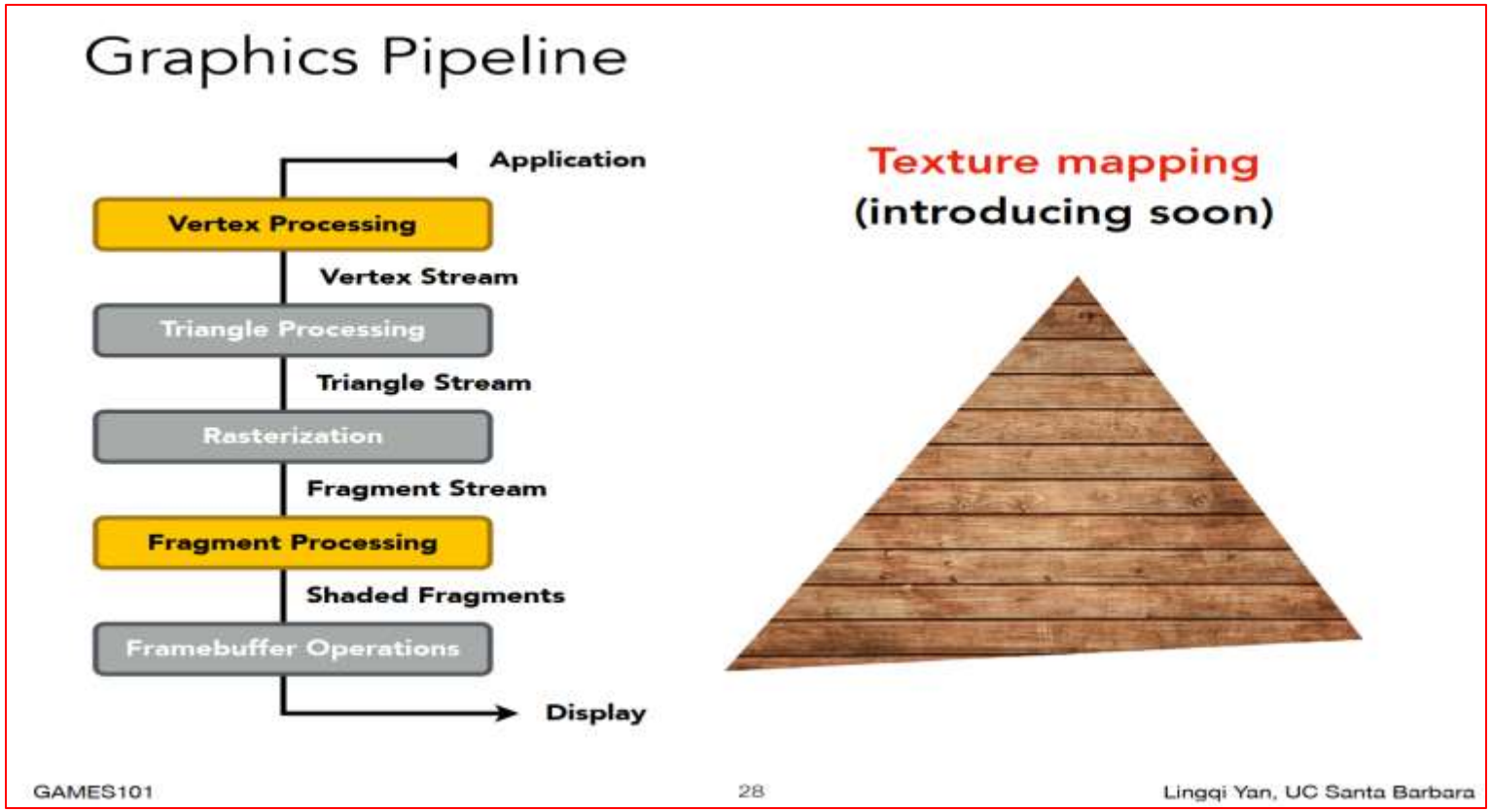
- Shading(Local Lighting)
  - Vertex and Fragment Processing



The University of New Mexico

Today's Theme

- Texture Mapping
  - Vertex and Fragment Processing



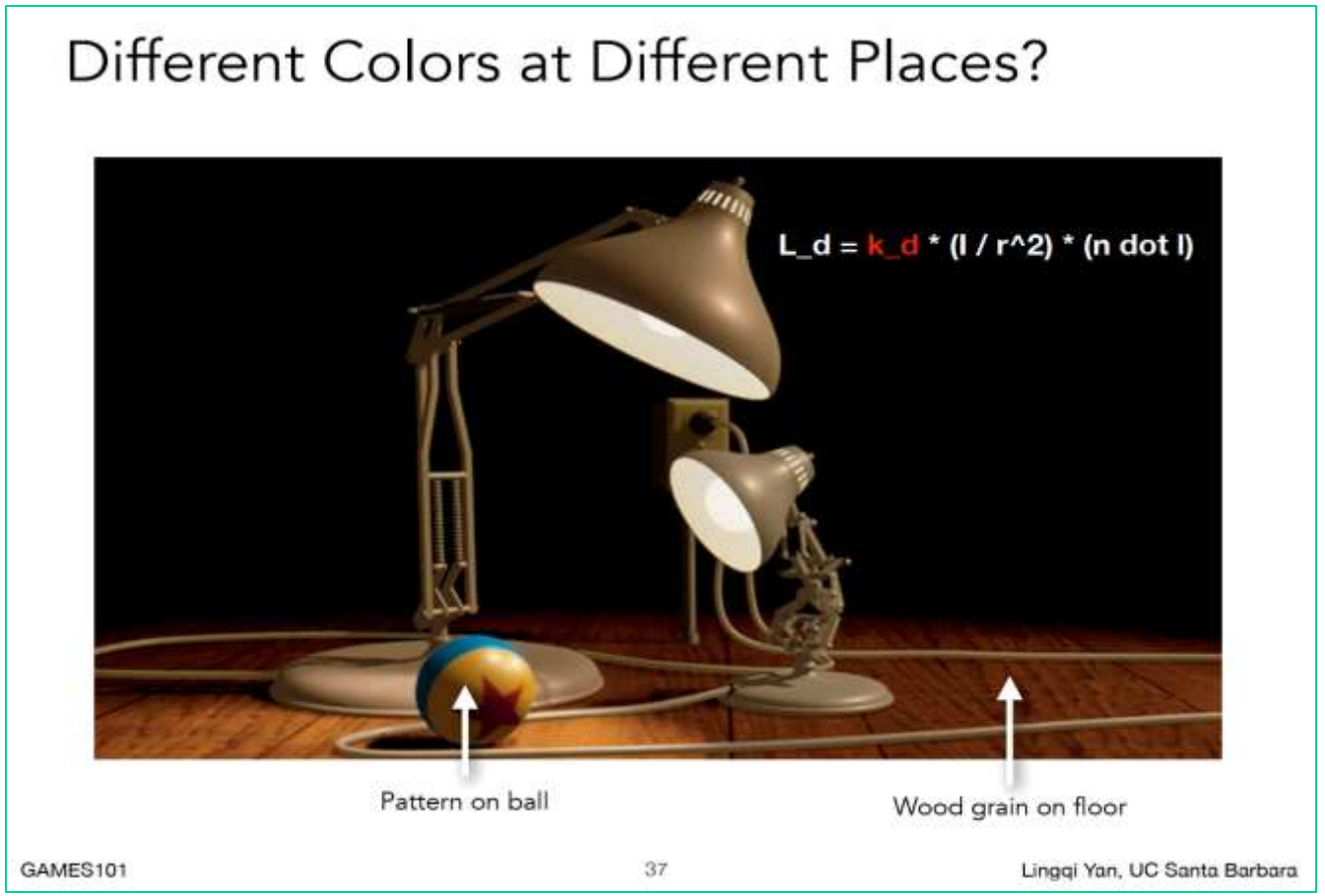
Outline

- **Why Texture Mapping**
- **What is Texture Mapping**
- Basic Texture Mapping Strategy
  - **Coordinates Mapping**
    - Forward vs. Backward Mapping
    - 2D vs. 3D Mapping
  - **Texture Sampling**
    - Point Sampling vs. Area Sampling
    - Magnification vs. Minification
- Implementation Examples
  - Color Texture Mapping
  - Reflection Texture Mapping

The University of New Mexico

Why Texture Mapping?

- Lighting is difficult to describe the surface details
  - the reflective properties of each point on the surface of an object

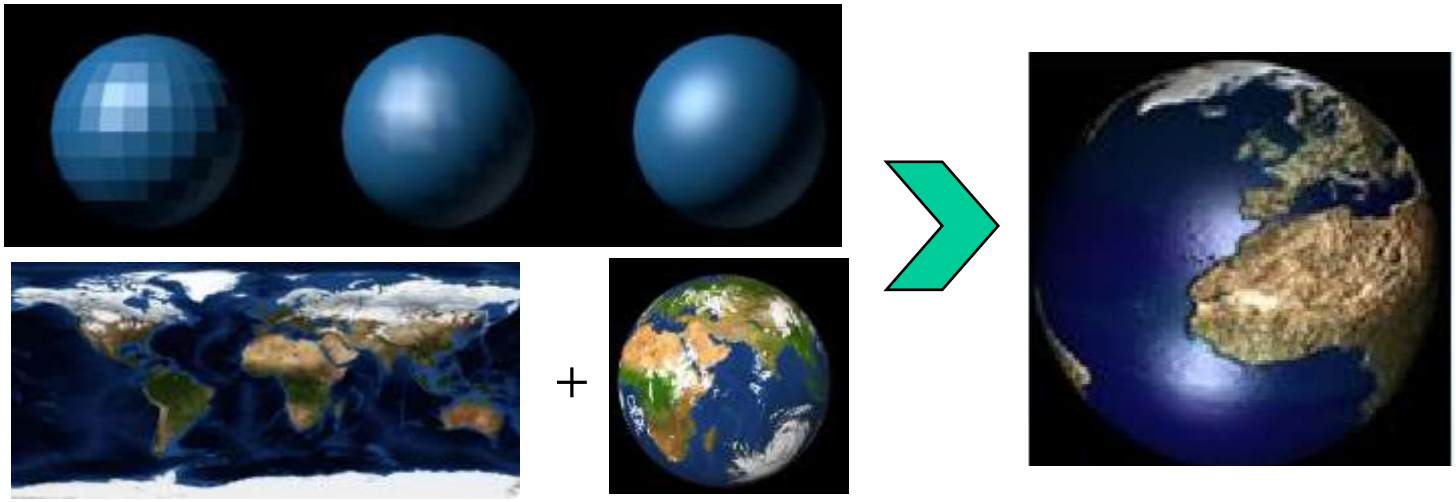


The University of New Mexico

Why Texture Mapping?

Reality： Lighting and Texture Mapping

- Lighting: Describes the lighting effect of the scene, but the surface properties of the object are monotonous单调的.
- Texture Mapping: describes the surface details of the object, enhancing the sense of reality on the surface.



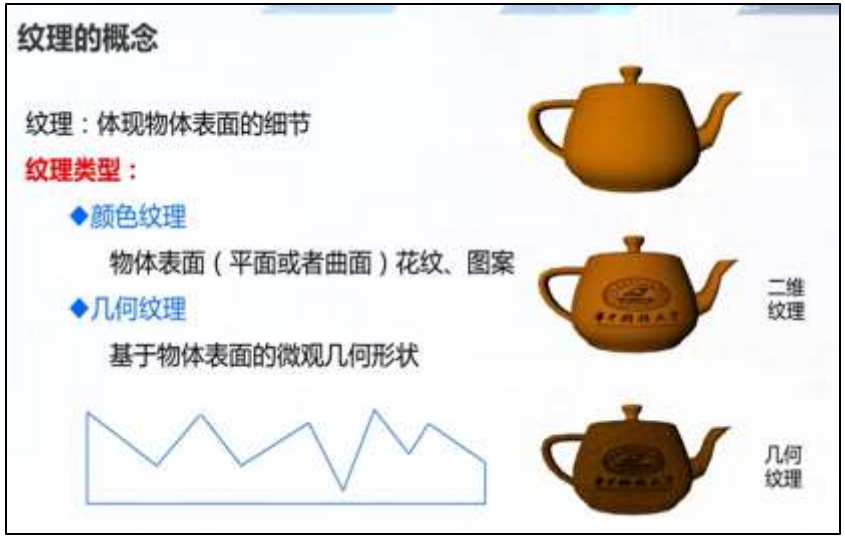
Outline

- Why Texture Mapping
- **What is Texture Mapping**
- Basic Texture Mapping Strategy
  - **Coordinates Mapping**
    - Forward vs. Backward Mapping
    - 2D vs. 3D Mapping
  - **Texture Sampling**
    - Point Sampling vs. Area Sampling
    - Magnification vs. Minification
- Implementation Examples
  - Color Texture Mapping
  - Reflection Texture Mapping

The University of New Mexico

What is Texture Mapping

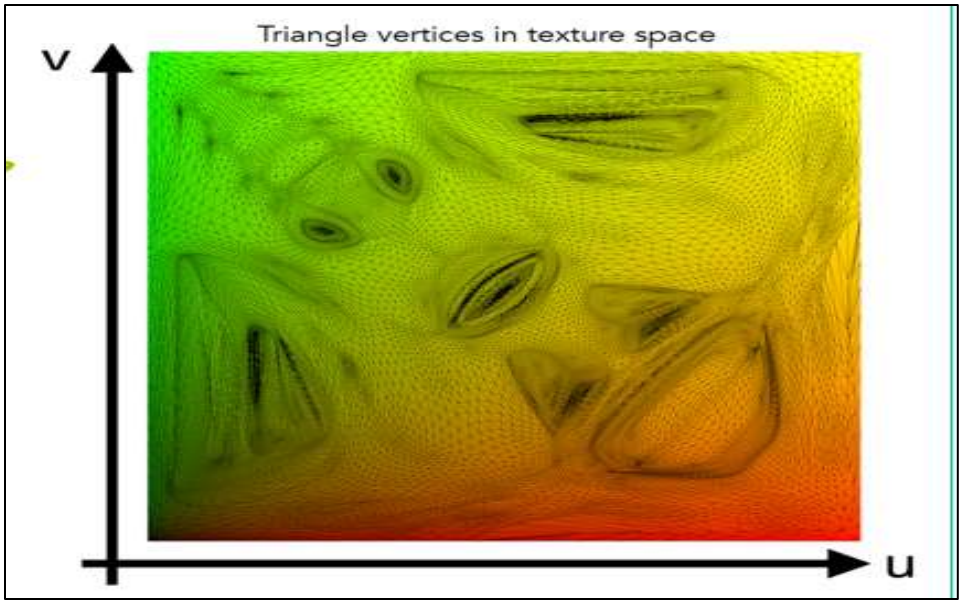
- **Texture 纹理**
  - **纹理通常是一个二维数组，通常是一个图片文件（如PNG或JPEG），它包含了颜色信息，这些信息会被映射到三维模型的表面上。**
  - 可以看作是定义在纹理空间上的函数 f(u,v)，一般为颜色纹理（图像），也可以是几何纹理（如微观几何形状参数）
- **Texel 纹素 (Texture Element)**
  - 纹理的基本组成元素。对于2D图像纹理，每个像素就是该纹理的纹素。

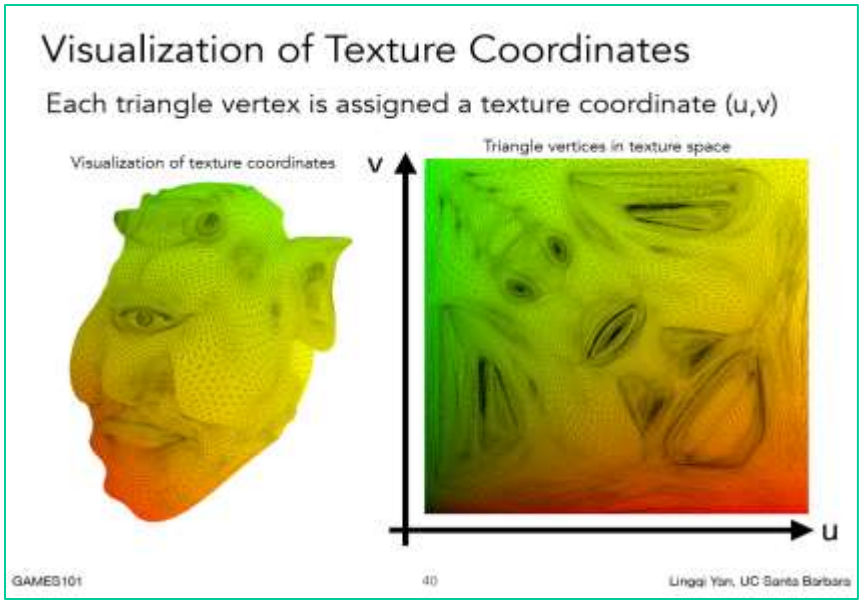


The University of New Mexico

What is Texture Mapping(cont.)

- **Texture Space 纹理空间**
  - 纹理空间通常是指纹理坐标系统。纹理空间范围通常是[0, 1]标准化坐标，如二维纹理f(u,v)，通常定义在单位正方形区域，$0 \le u \le 1$，$0 \le v \le 1$。
- **Texture Coordinate纹理坐标**
  - 纹素在纹理空间中的位置就是纹理坐标。对于2D纹理就是(u，v)值。

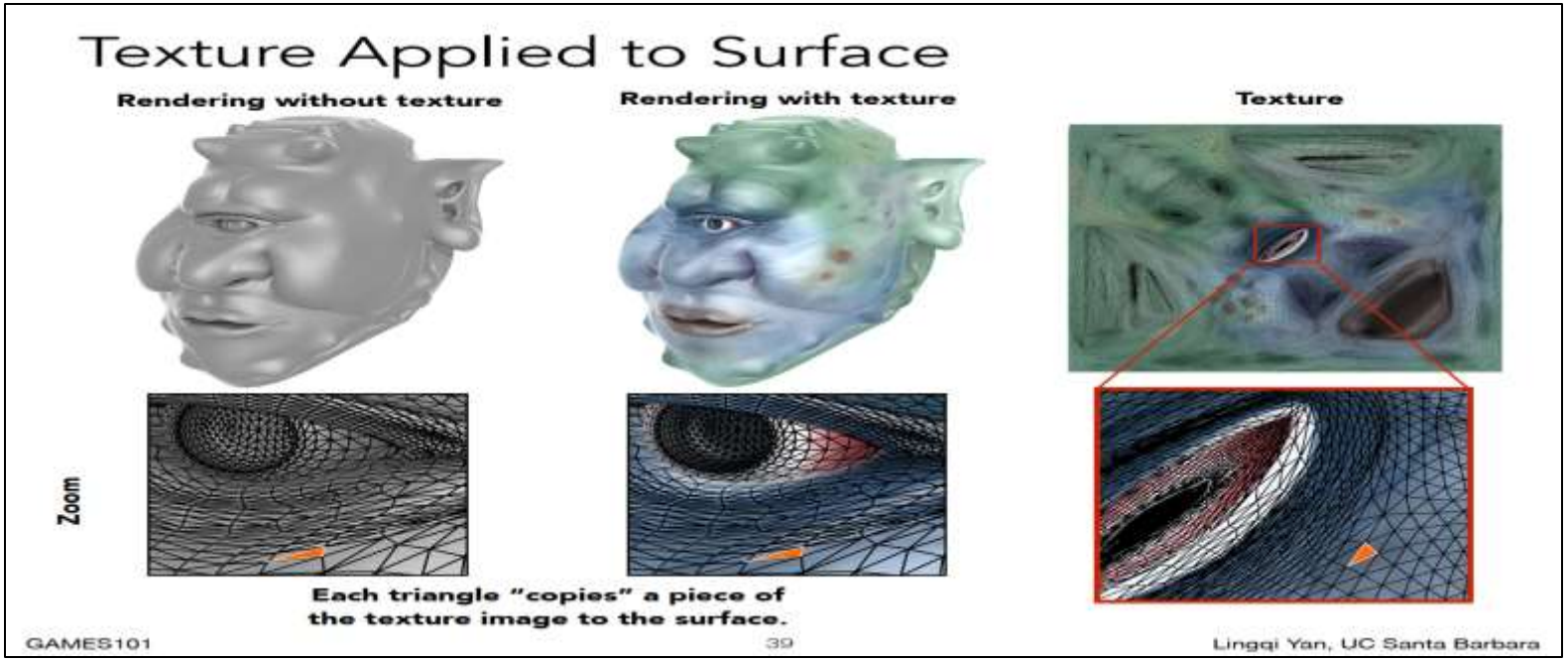




What is Texture Mapping(cont.)

The University of New Mexico

- Texture Mapping（纹理映射）
  - 纹理映射是将纹理坐标（通常是二维的UV坐标）与三维模型上的顶点相关联的过程；这些坐标决定了纹理的哪一部分将被应用到模型的每个三角形面上。
  - 三维建模中，“UV 图”是“立体模型”的“皮肤”。

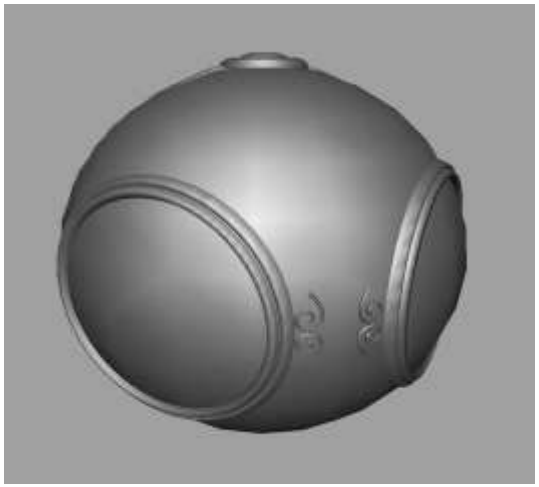


The University of New Mexico

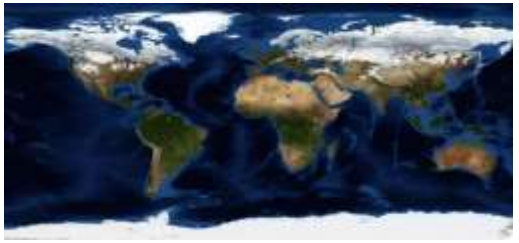
What is Texture Mapping（cont.)

- **Texture Mapping（cont.）**
  - Type1: **Color Mapping（颜色映射/贴图Diffuse Maps）**

- Uses images to fill surface of polygons



geometric model



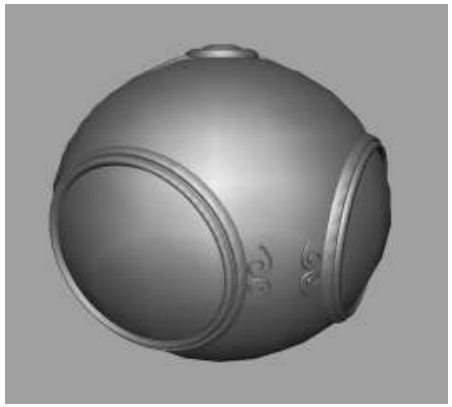
texture mapped

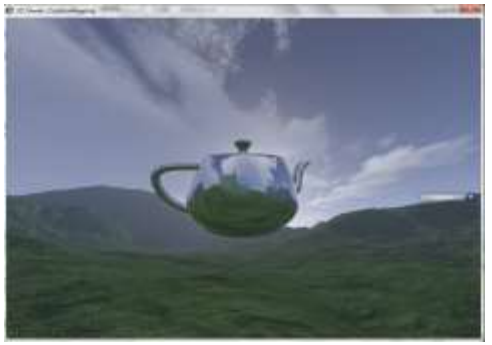
The University of New Mexico

What is Texture Mapping(cont.)

➢Texture Mapping（cont.）

- Type2: **Environment (reflection mapping)环境（反射）映射**
- Uses a picture of the environment for texture maps
  - Allows simulation of highly specular surfaces(高镜面表面）





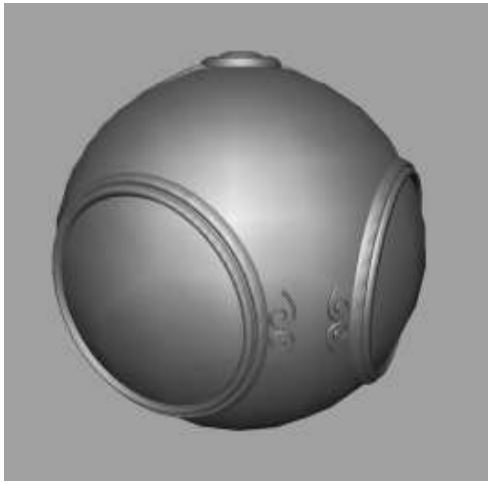
geometric model　　texture mapped

The University of New Mexico

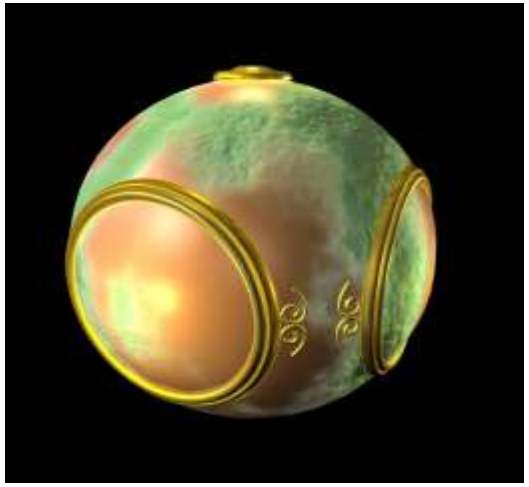
What is Texture Mapping(cont.)

➢Texture Mapping（cont.）

- Type3: Bump Mapping(凹凸映射)：**模拟表面凹凸不平的效果**
  - **法线纹理（Normal Maps）**：它通过在纹理中存储法线信息，而不是直接在模型上增加额外的几何细节，从而在视觉上增强表面的凹凸效果。



geometric model



texture mapped

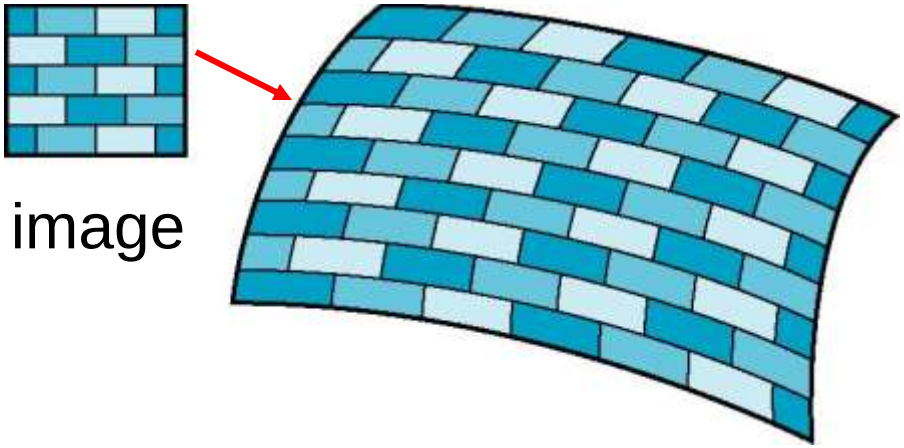
Outline

- Why Texture Mapping
- What is Texture Mapping
- **Basic Mapping Strategy**
  - **Coordinates Mapping**
    - Forward vs. Backward Mapping
    - 2D vs. 3D Mapping
  - **Texture Sampling**
    - Point Sampling vs. Area Sampling
    - Magnification vs. Minification
- Implementation Examples
  - Color Texture Mapping
  - Reflection Texture Mapping

The University of New Mexico

Coordinates Mapping

- Although the idea is simple , but not easy!
  - ➢ Basic problem is how to find the maps!



2D image

3D surface

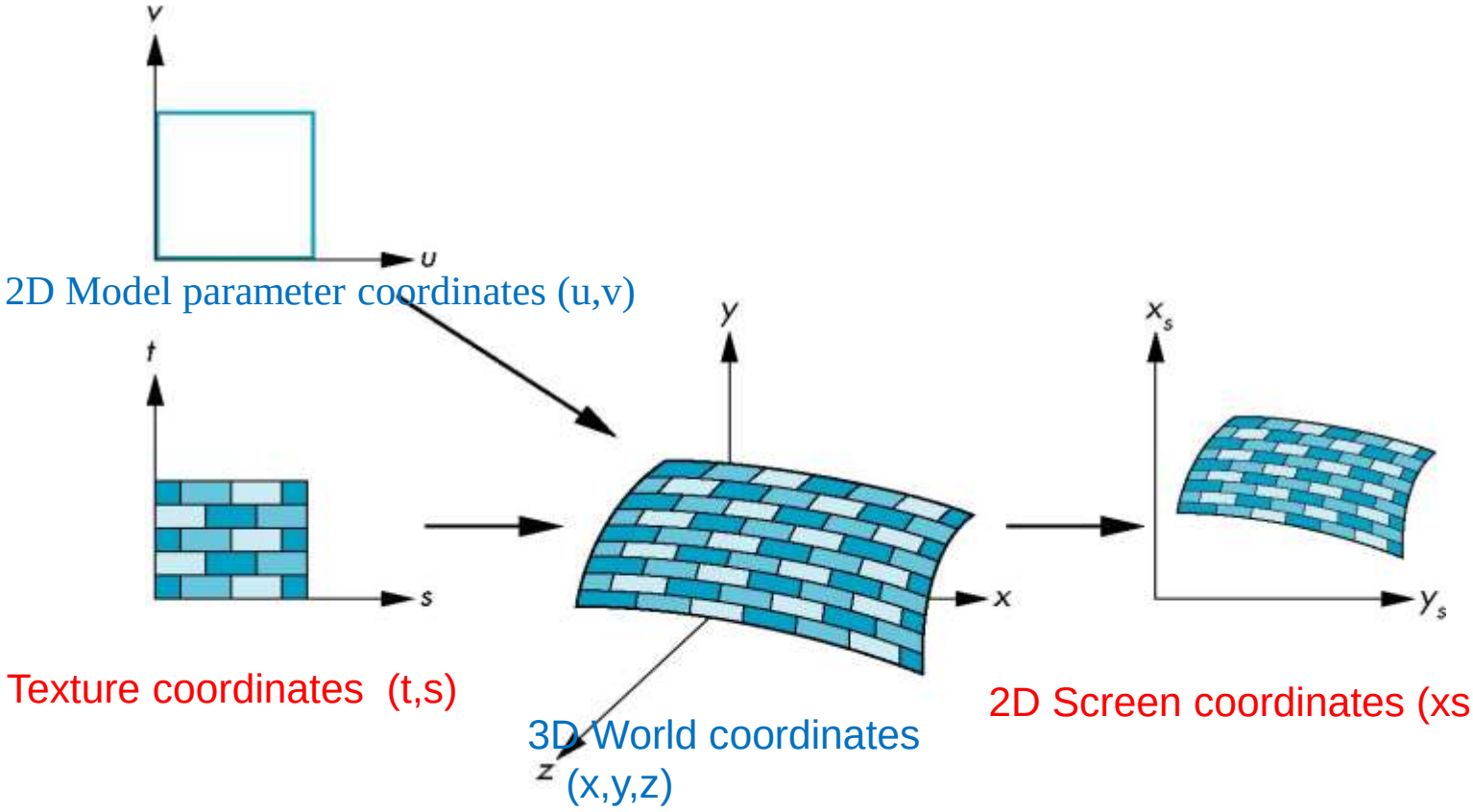
blueflower2.JPG

The University of New Mexico

Coordinates Mapping(cont.)

- Coordinate Systems:

  Map an image to a Surface: there are 3 or 4 coordinate systems involved



The University of New Mexico

Coordinates Mapping(cont.)

➢Forward vs. Backward Mapping

- Forward Coordinates Mapping

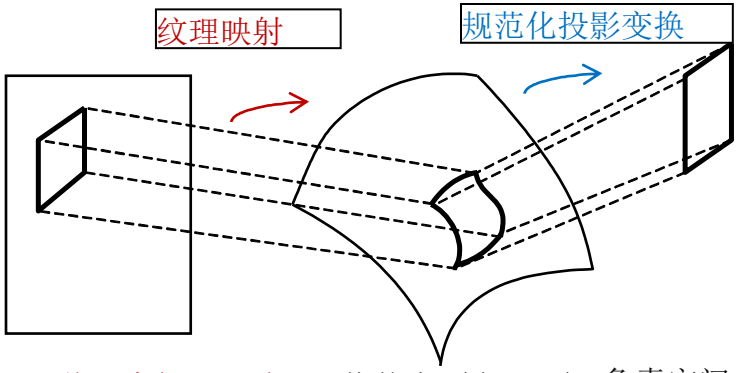
Consider mapping from texture coordinates to a point a surface ,

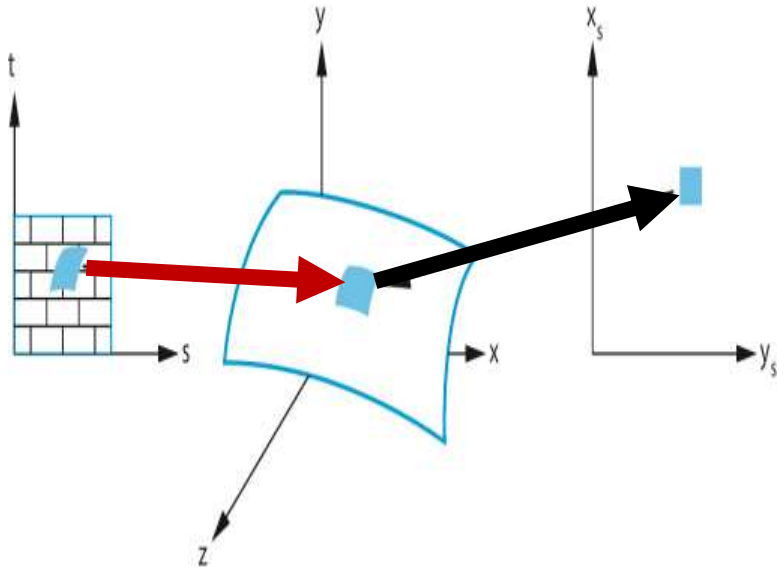
Appear to need a map of the form :

$x = X(t,s)$

$y = Y(t,s)$

$z = Z(t,s)$





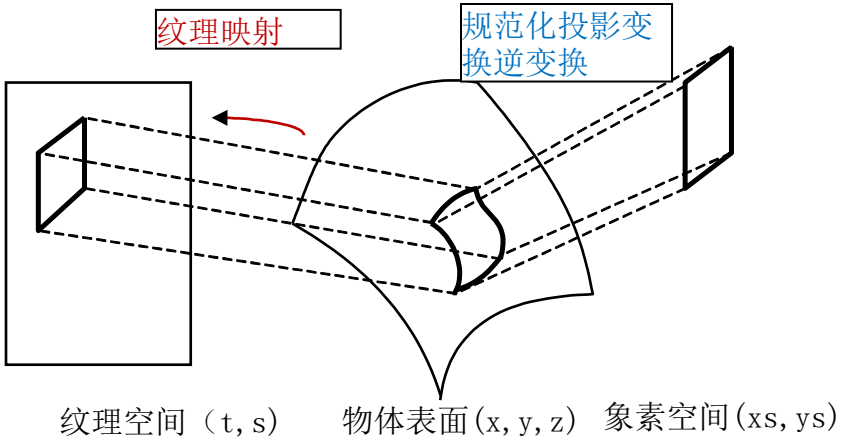
The University of New Mexico

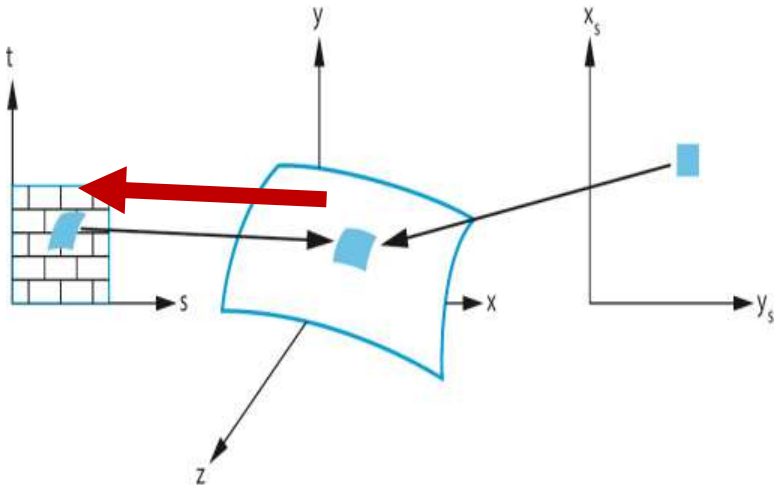
Coordinates Mapping(cont.)

➢Forward vs. Backward Mapping(cont.)

➢Backward Coordinates Mapping

- Given a point on an object, we want to know to which point in the texture it corresponds, Need a map of the form
  - t = T(x,y,z)
  - s = S(x,y,z)





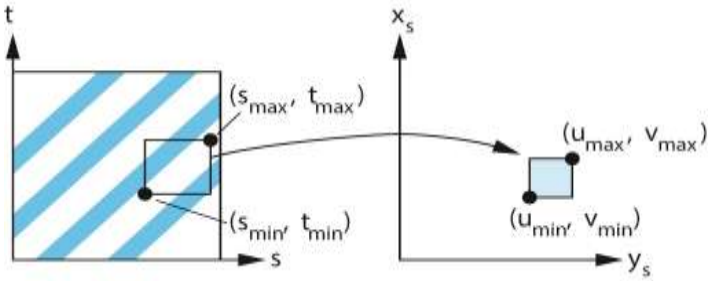
✓采用逆向Backward映射策略，因为只有少量多边形通过了裁剪和光栅化，而转换为片元; 而正向Forward坐标映射有太多映射是浪费了。

Coordinates Mapping(cont.)

➢ 2D vs. 3D Mapping

➢ 2D Mapping

- Direct find 2D coordinates mapping function: (xs,ys)~ (s,t)
  - **xs=Umin+((s-Smin)/(Smax-Smin))*(Umax-Umin)**
  - **ys=Vmin+((t-Tmin)/(Tmax-Tmin))*(Vmax-Vmin)**



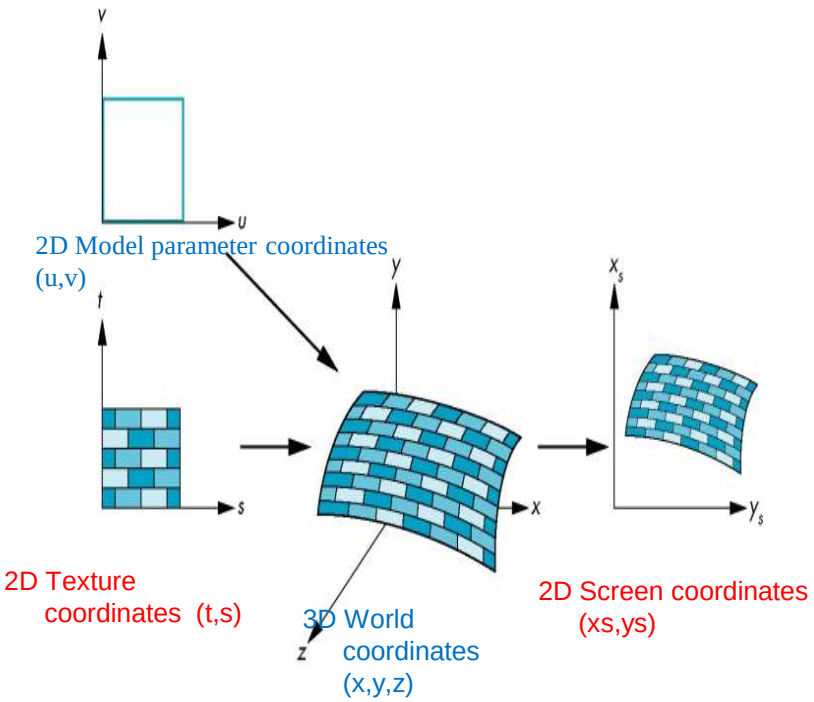
- Advantage：映射容易，可直接将纹理映射到平面区域，或者参数曲面的一组片面上。
- Disadvantage：对于弯曲表面，贴图不真实。

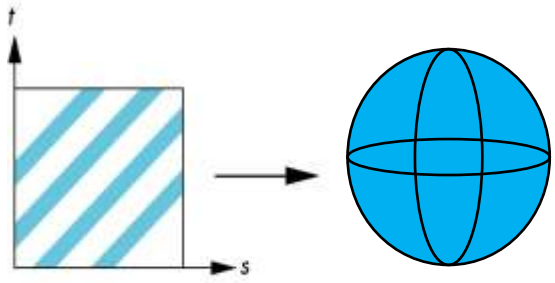
Coordinates Mapping(cont.)

➢ 2D vs. 3D Mapping(cont.)

➢Non-Linear Mapping: 在3D对象坐标空间找映射关系

➢ 例如将纹理贴到球面上





(s,t)~ (u,v)~(x,y,z)~(xs,ys)

s = u
t = v

X' = r sin πu cos 2πv
Y' = r sin πu sin 2πv
Z' = r cos πu

(xs,ys) = f(x,y,z)

Outline

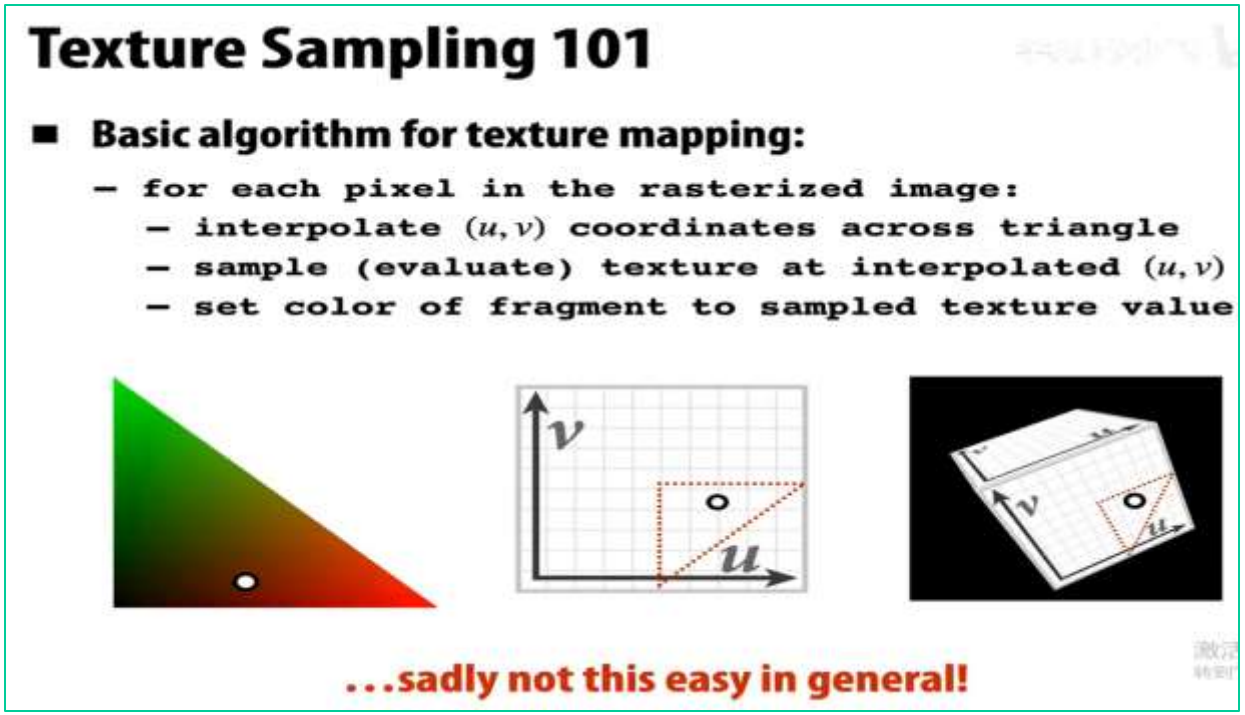
- Why Texture Mapping
- What is Texture Mapping
- **Basic Mapping Strategy**
  - **Coordinates Mapping**
    - Forward vs. Backward Mapping
    - 2D vs. 3D Mapping
  - **Texture Sampling**
    - Point Sampling vs. Area Sampling
    - Magnification vs. Minification
- Implementation Examples
  - Color Texture Mapping
  - Reflection Texture Mapping

The University of New Mexico

Texture Sampling

- 纹理采样Texture Sampling

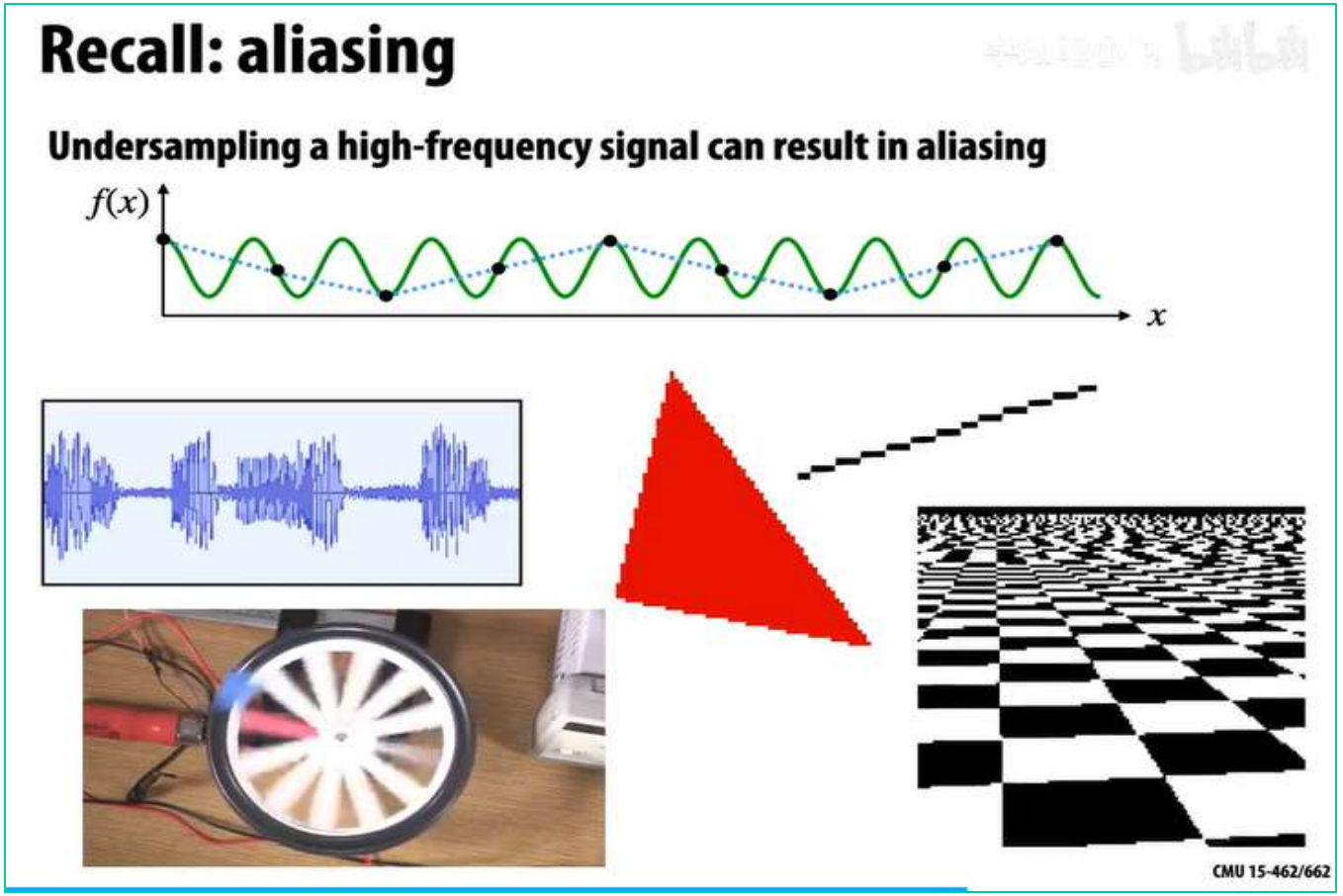
参B站CMU计算机图形学
https://www.bilibili.com/video/BV1LyYveiEB2/?spm_id_from=333.337.search-card.all.click&vd_source=c93f9d5c2c6ee8043fe0db22203390ee



The University of New Mexico

Texture Sampling（cont.）

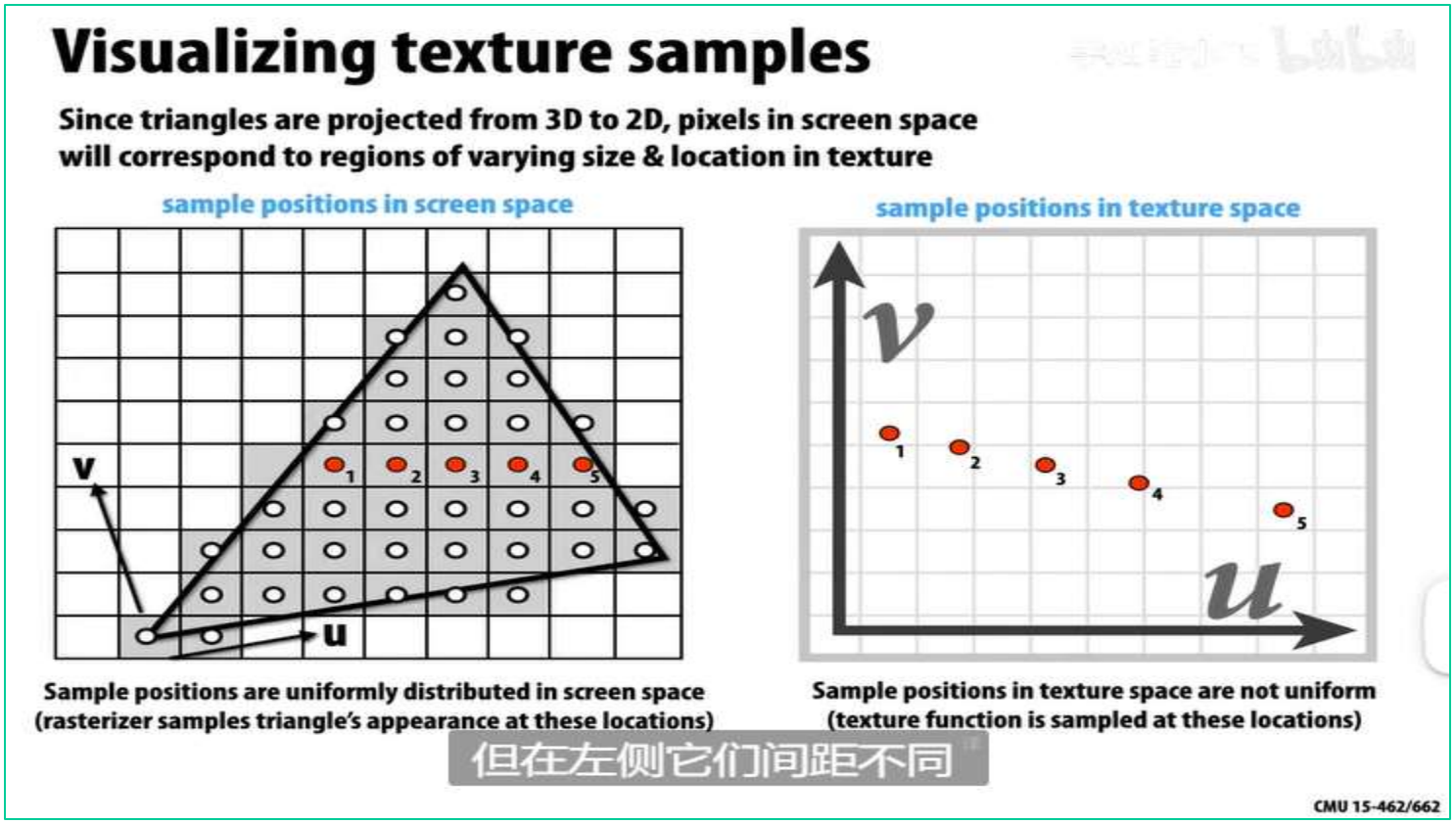
- 纹理采样会走样aliasing



The University of New Mexico

Texture Sampling（cont.）

- 纹理采样例子
  - 因为三角形从3D投影到2D，屏幕空间的像素对应纹理空间对应会有变化的大小和位置，该如何采样呢？



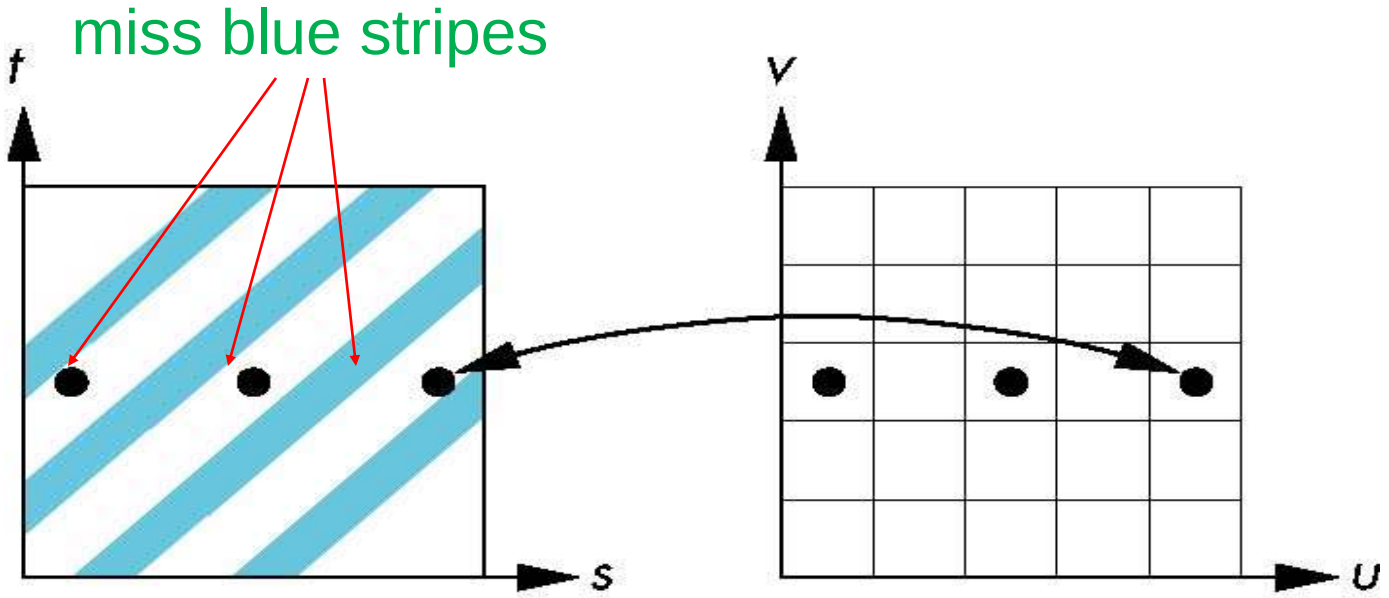
The University of New Mexico

Texture Sampling（cont.）

➢Point Sampling vs. Area Sampling

a. Point sampling 点采样:“Nearest”: 将像素中心逆向投影到纹理坐标空间，得某个“最靠近”纹理坐标的“纹素”，作为该片元的颜色

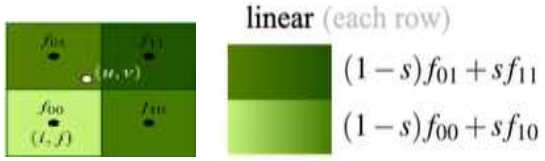
- can lead to aliasing errors



The University of New Mexico

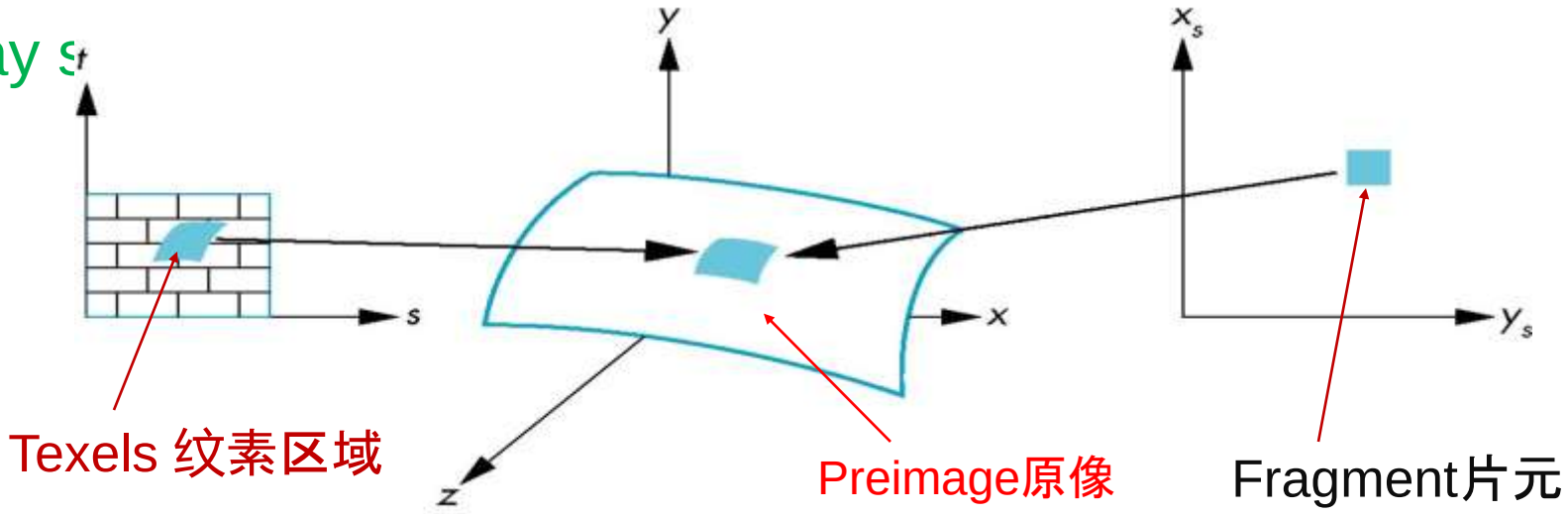
Texture Sampling(cont.)

➢Point Sampling vs. Area Sampling



b. Area Sampling 区域采样 “Linear”: A better but slower option is to use *area averaging*(**计算“原像preimage”所对应的多个“纹素”值的平均值/加权平均值, 赋给片元)**

- may s

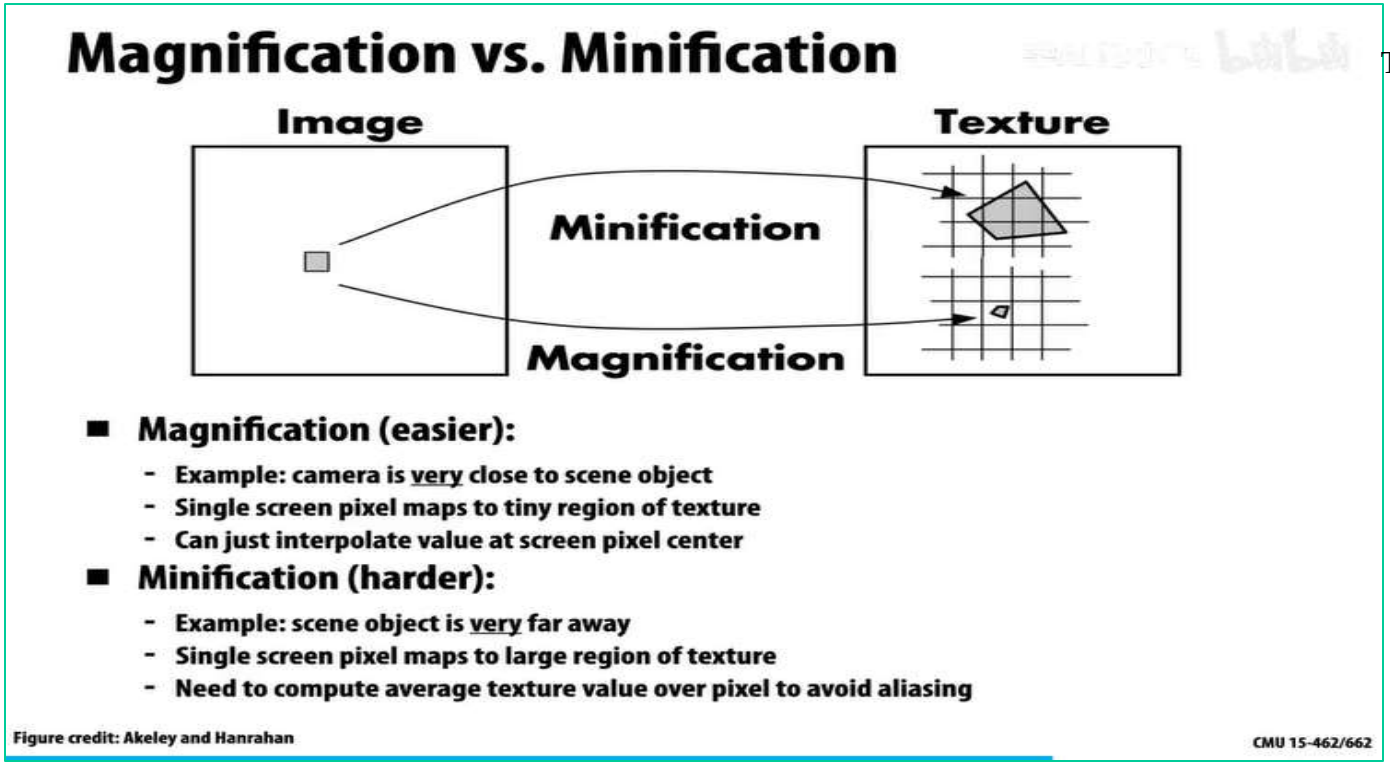


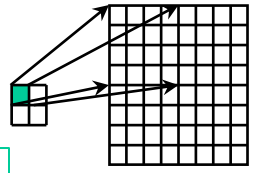
Note: that preimage(原像) of fragment is curved

The University of New Mexico

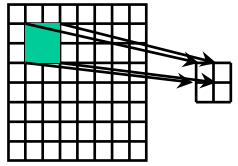
Texture Sampling（cont.）

- Magnification vs. Minification
  - 放大：从纹理到像素是放大，从像素到纹理是缩小
  - 缩小：从纹理到像素是缩小，从像素到纹理是放大





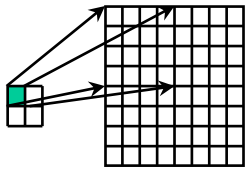
Texture Polygon
Magnification



Texture Polygon
Minification

The University of New Mexico

Texture Sampling（cont.）



➢Magnification vs. Minification（cont.）

➢Magnification is easier

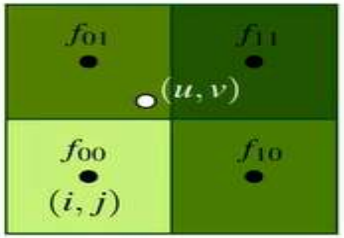
- **Magnification (easier):**
  - Example: camera is very close to scene object
  - Single screen pixel maps to tiny region of texture
  - Can just interpolate value at screen pixel center

➢点采样nearest：fast but ugly,

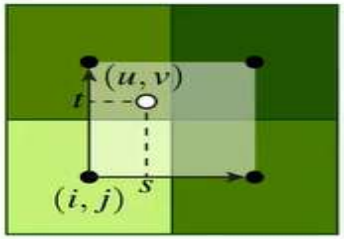
➢线性插值采样linear: slow but smooth

Bilinear interpolation (magnification)

How can we "look up" a texture value at a non-integer location (u, v)?



$$i = \lfloor u - \tfrac{1}{2} \rfloor$$
$$j = \lfloor v - \tfrac{1}{2} \rfloor$$



$$s = u - (i + \tfrac{1}{2}) \in [0,1]$$
$$t = v - (j + \tfrac{1}{2}) \in [0,1]$$

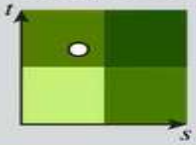
linear (each row)

$$(1-s) f_{01} + s f_{11}$$
$$(1-s) f_{00} + s f_{10}$$

bilinear

$$(1-t)\left((1-s) f_{00} + s f_{10}\right) + t\left((1-s) f_{01} + s f_{11}\right)$$

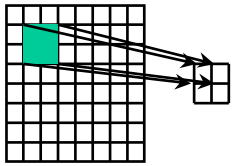
nearest neighbor



fast but ugly:
just grab value of nearest "texel" (texture pixel)

The University of New Mexico

Texture Sampling（cont.）



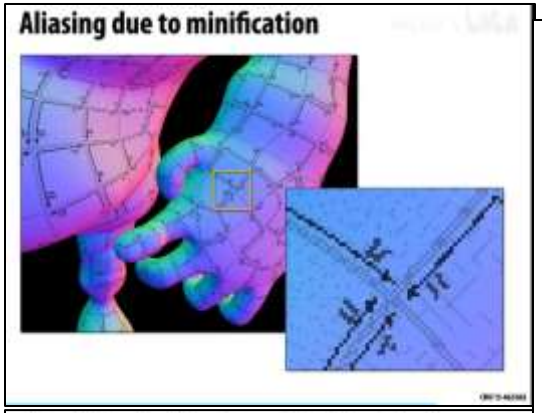
Texture Polygon
Minification

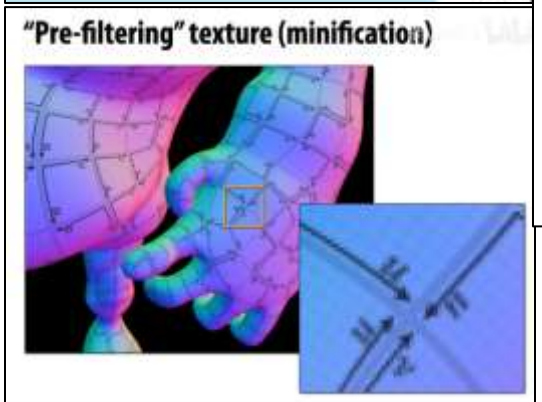
➢Magnification vs. Minification（cont.）

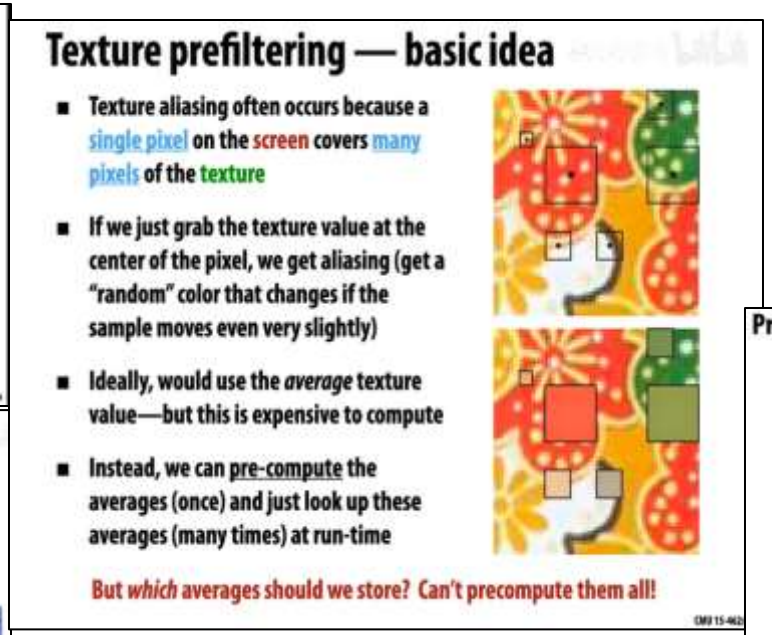
➢Minification is harder：aliasing may heavy

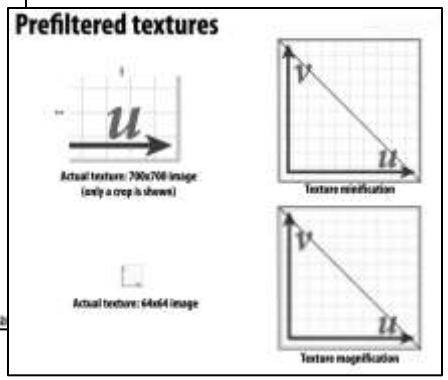
➢解决方案： 预过滤/预滤波/前置滤波“pre-filtering”

Idea:若简单点采nearest会容易aliasing,而区域平均linear又计算昂贵;所以采用”预滤波”方法，即将纹理预计算为不同的分辨率（resolution）纹理保存并匹配使用





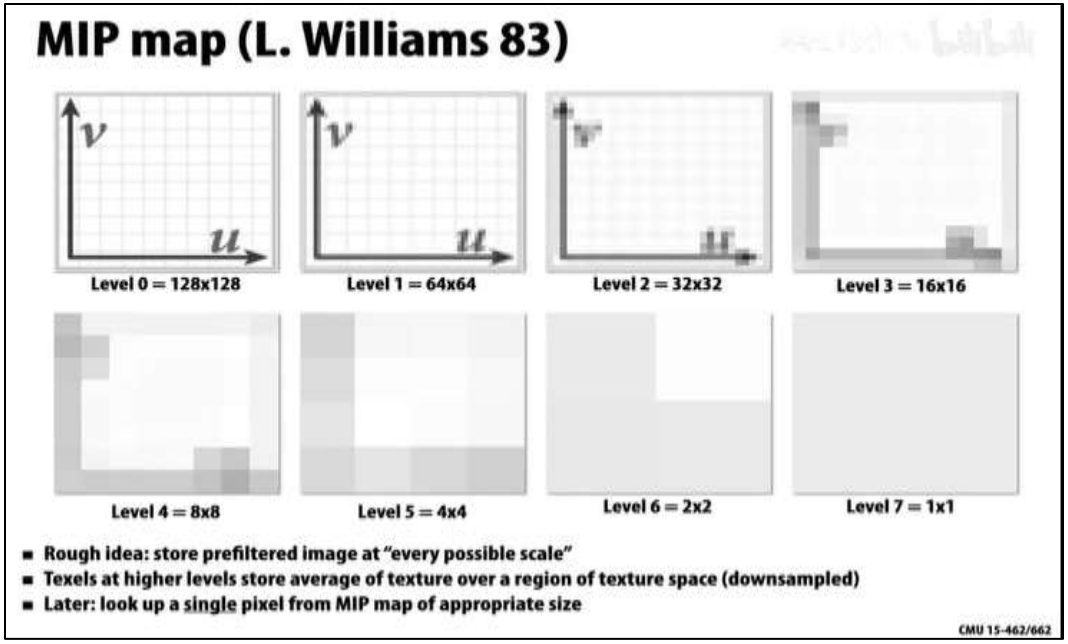


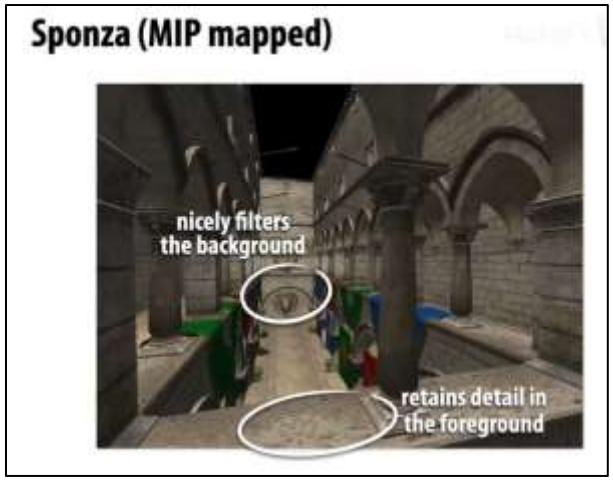


Texture Sampling（cont.）

- Magnification vs. Minification（cont.）
  - Minification is harder： aliasing may heavy
    - 预过滤/预滤波/前置滤波“pre-filtering”(cont.)
    - **MIP map(Multidimensional Interface Pattern mapping)**

Mipmap 是一种被广泛采用的图形优化技术，它通过牺牲一定的存储空间来提高“渲染效率”和“图像质量”。Mipmap 通过预先生成一系列逐渐缩小的纹理图像来工作，这些图像被称为层级（levels），每一层级都是前一层级纹理大小的一半。

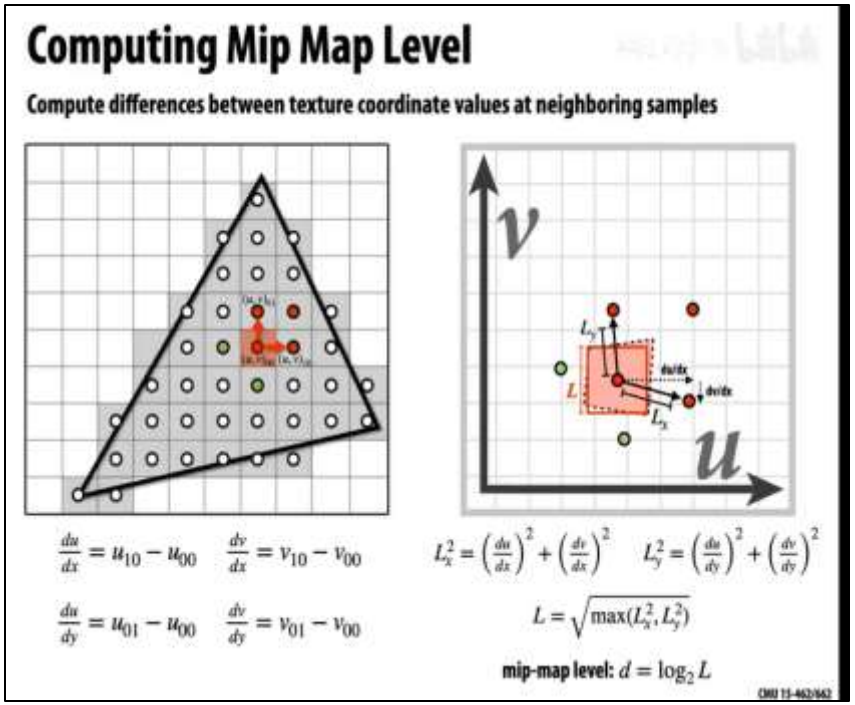


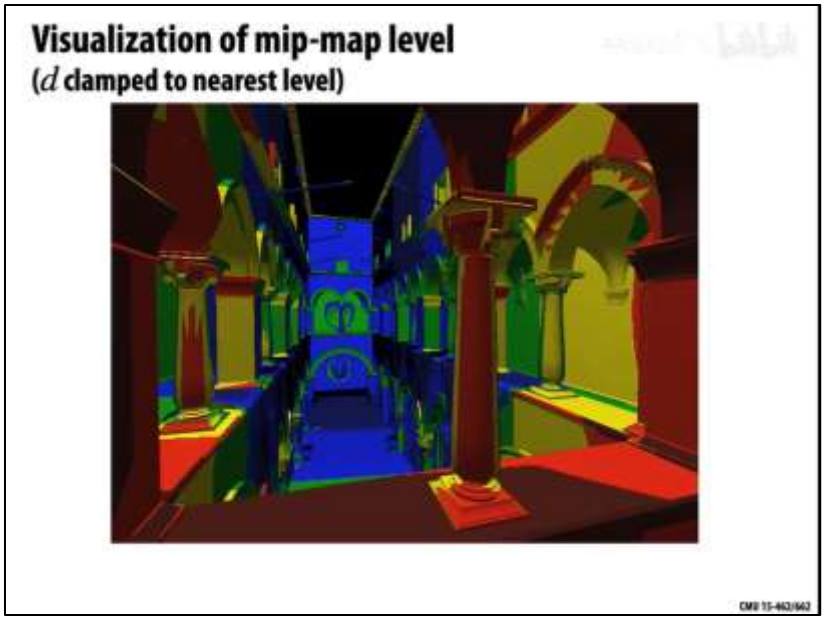


The University of New Mexico

Texture Sampling（cont.）

- Magnification vs. Minification（cont.）
  - Minification is harder： aliasing may heavy
    - 预过滤/预滤波/前置滤波“pre-filtering”(cont.)
    - **MIP map(Multidimensional Interface Pattern mapping)**
      - **如何计算得到MIP map level**





Outline

- Why Texture Mapping
- What is Texture Mapping
- **Basic Mapping Strategy**
  - **Coordinates Mapping**
    - Forward vs. Backward Mapping
    - 2D vs. 3D Mapping
  - **Texture Sampling**
    - Point Sampling vs. Area Sampling
    - Magnification vs. Minification
- Texture Mapping Implementation
  - Color Texture Mapping
  - Reflection Texture Mapping

Texture Mapping Implementation

➢Where does mapping take place?

- Mapping techniques are implemented at the end of the rendering pipeline
  - Very efficient because few polygons make it past the clipper (在片元处理里进行，因为只有少量多边形通过了裁剪和光栅化)
- Images and geometry flow through separate pipelines that join during fragment processing,
  - "complex" textures do not affect geometric complexity(纹理复杂性不影响几何复杂性)

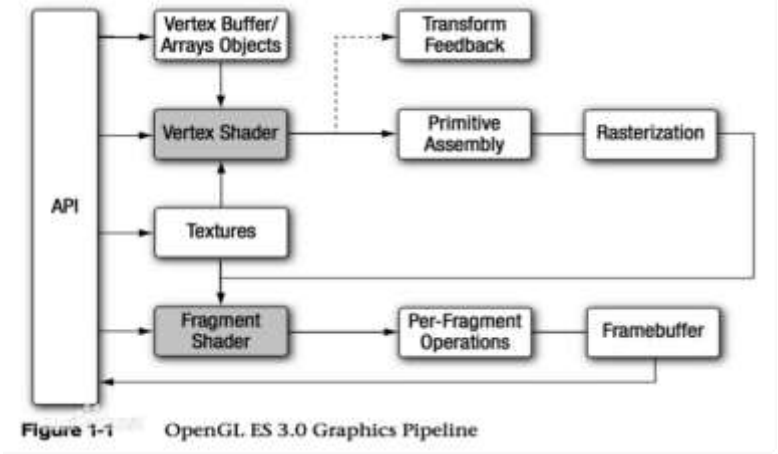
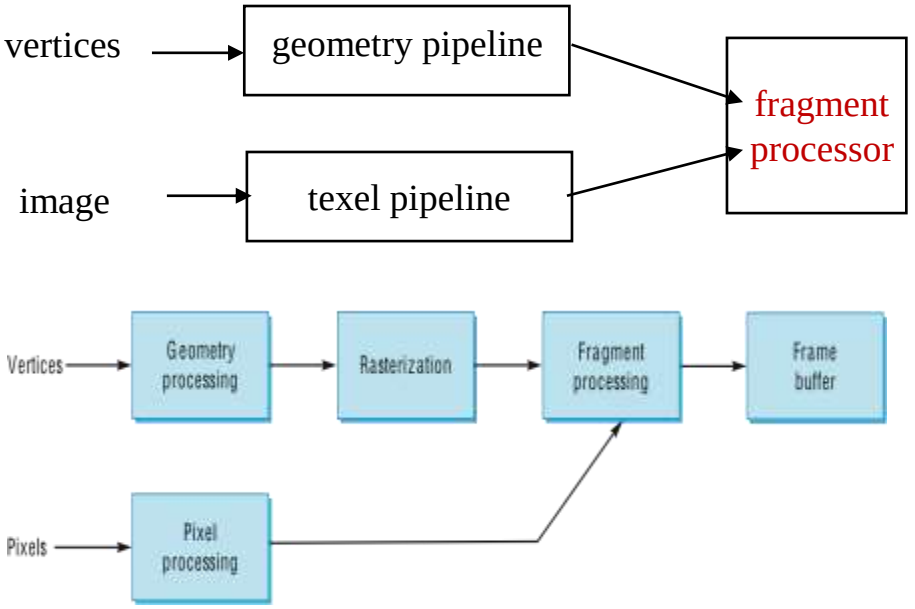
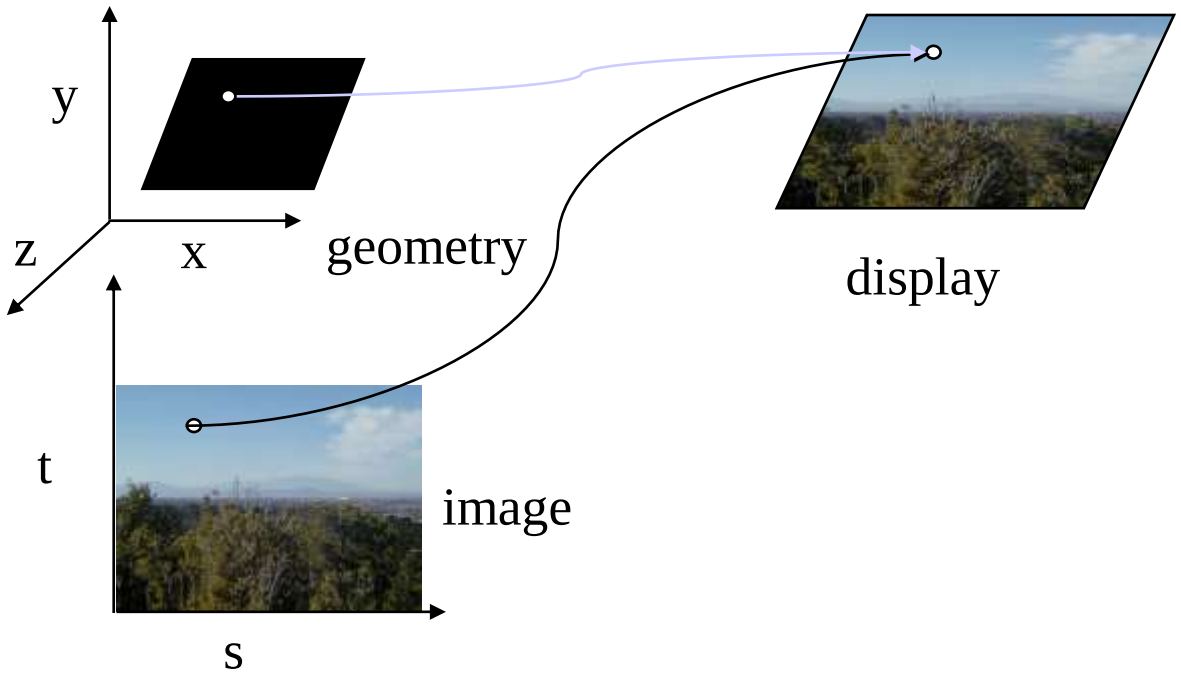


Figure 1-1 OpenGL ES 3.0 Graphics Pipeline



Texture Mapping Implementation **(cont.)**

- The texture (below) is a 256 x 256 image that has been mapped to a rectangular polygon which is viewed in perspective



Outline

- Why Texture Mapping
- What is Texture Mapping
- **Basic Mapping Strategy**
  - **Coordinates Mapping**
    - Forward vs. Backward Mapping
    - 2D vs. 3D Mapping
  - **Texture Sampling**
    - Point Sampling vs. Area Sampling
    - Magnification vs. Minification
- Texture Mapping Implementation
  - Color Texture Mapping
  - Reflection Texture Mapping

Color Texture Mapping

纹理贴图的主要工作：

1. **specify the texture object 创建纹理对象**
   - ◆ read or generate image读取或创建数字图像
   - ◆ assign image to texture object创建纹理对象

2. **Texture coordinates to Vertex coordinates 建立坐标映射**
   - ◆assign texture coordinates to vertices 建立纹理和对象的坐标映射
   - ◆ Texture coordinates passing传递纹理坐标给着色器(顶点着色器中)

3. **Texture sampling纹理采样**
   - ◆ assign sampler to texture object 设置采样器给纹理对象
   - ◆ specify texture sampling parameters 设置纹理对象的采样参数
   - ◆ sampling in fragment shader用采样器进行采样(在片元着色器中)

实例代码参考: \Angle8E Code\07\ textureCube.**

Color Texture Mapping（cont.)

1. specify the texture（cont.)

- read or generate image读取或创建数字图像
- assign image to texture object创建纹理对象并关联该数字图像

//specify image in HTML file with <img> tag

Ref: \Angle8E Code\07\ textureCube1.js

Reading a image

```
<img id = "texImage" src = "tiger.png" hidden></img>
```

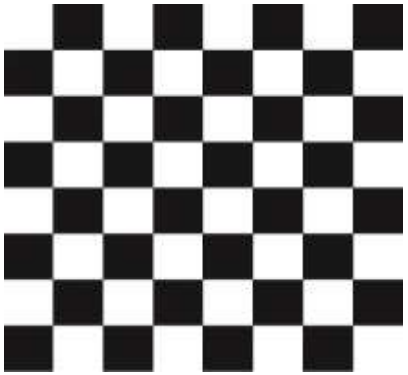
Color Texture Mapping（cont.)

1. specify the texture（cont.)

◆read or generate image读取或创建数字图像（cont.)

◆assign image to texture object创建纹理对象并关联该数字图像

```
var texSize = 64;
// Create a checkerboard pattern using floats
var image1 = new Array()
    for (var i =0; i<texSize; i++)  image1[i] = new Array();
    for (var i =0; i<texSize; i++)
        for ( var j = 0; j < texSize; j++)
           image1[i][j] = new Float32Array(4);
    for (var i =0; i<texSize; i++) for (var j=0; j<texSize; j++) {
        var c = (((i & 0x8) == 0) ^ ((j & 0x8)  == 0));
        image1[i][j] = [c, c, c, 1];
    }
```



Generate A Checkerboard Image

\Angle8E Code\07\ textureCube2.js

The University of New Mexico

Color Texture Mapping（cont.）

1. specify the texture(cont.)

- read or generate image读取或创建数字图像
- assign image to texture object创建纹理对象并关联该数字图像
  - ✓每个纹理对象包含“纹理数组”和各种“纹理参数”（控制如何进行纹理采样）
  - ✓可以创建多个纹理对象，但是只有一个是当前纹理对象.

```
var image = document.getElementById("texImage");
configureTexture(image);
```

```
image.onload = function() {// configureTexture( image );
function configureTexture( image ) {
    texture = gl.createTexture();
    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGB, gl.RGB, gl.UNSIGNED_BYTE, image);
    gl.generateMipmap(gl.TEXTURE_2D);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER,gl.NEAREST_MIPMAP_LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.NEAREST);
    gl.uniform1i(gl.getUniformLocation(program, "uTexMap"), 0);
}
```

The University of New Mexico

Color Texture Mapping（cont.)

2. Map Texture coordinates to vertex coordinates

◆assign texture coordinates to vertices 建立纹理和对象的坐标映射关系

◆Texture coordinates passing传递纹理坐标给着色器

```
var texCoord = [
    vec2(0, 0),
    vec2(0, 1),
    vec2(1, 1),
    vec2(1, 0)
];
```

➢ 定义纹理空间的纹理坐标[0,1]

```
function quad(a, b, c, d) {
    positionsArray.push(vertices[a]);
    colorsArray.push(vertexColors[a]);
    texCoordsArray.push(texCoord[0]);

    positionsArray.push(vertices[b]);
    colorsArray.push(vertexColors[a]);
    texCoordsArray.push(texCoord[1]);

    positionsArray.push(vertices[c]);
    colorsArray.push(vertexColors[a]);
    texCoordsArray.push(texCoord[2]);

    positionsArray.push(vertices[a]);
    colorsArray.push(vertexColors[a]);
    texCoordsArray.push(texCoord[0]);

    positionsArray.push(vertices[c]);
    colorsArray.push(vertexColors[a]);
    texCoordsArray.push(texCoord[2]);

    positionsArray.push(vertices[d]);
    colorsArray.push(vertexColors[a]);
    texCoordsArray.push(texCoord[3]);
}
```

➢ *顶点具有纹理坐标属性*

- *纹理坐标数组texCoordsArray*

Color Texture Mapping（cont.)

2. Texture coordinates to vertex coordinates(cont.)

◆assign texture coordinates to vertices 建立纹理和对象的坐标映射

◆Texture coordinates passing传纹理坐标给着色器

```
var tBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, tBuffer);
gl.bufferData(gl.ARRAY_BUFFER, flatten(texCoordsArray), gl.STATIC_DRAW);

var texCoordLoc = gl.getAttribLocation(program, "aTexCoord");
gl.vertexAttribPointer(texCoordLoc, 2, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(texCoordLoc);
```

- ✓ **App**将纹理坐标作为顶点属性数组传送给顶点着色器 **Vertext shader**

```
in vec2 aTexCoord;
out vec2 vTexCoord;

vTexCoord = aTexCoord;
```

- ✓ **Vertext shader** 定义属性变量vTexCoord接收顶点的纹理坐标，并且传到片元着色器fTexCoord（插值得到片元纹理坐标）

```
in vec2 vTexCoord;

fColor = texture(uTextureMap, vTexCoord);
```

- ✓ Fragment shader 接受插值后的纹理坐标，进行后续纹理采样用。

Color Texture Mapping（cont.）

3. Texture sampling纹理采样

- assign sampler to texture object 设置采样器给纹理对象
- specify texture sampling parameters 设置纹理对象的采样的参数
- sampling in fragment shader用采样器进行采样(在片元着色器中)

```
function configureTexture( image ) {
    texture = gl.createTexture();
    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGB, gl.RGB, gl.UNSIGNED_BYTE, image);
    gl.generateMipmap(gl.TEXTURE_2D);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER,gl.NEAREST_MIPMAP_LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.NEAREST);
    gl.uniform1i(gl.getUniformLocation(program, "uTexMap"), 0);
}
```

Color Texture Mapping（cont.）

3. Texture sampling纹理采样

- ◆**assign sampler to texture object 设置采样器给纹理对象**
- ◆**specify texture sampling parameters 设置纹理对象的采样的参数**
- ◆**sampling in fragment shader用采样器进行采样(在片元着色器中)**

```
function configureTexture( image ) {
    texture = gl.createTexture();
    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGB, gl.RGB, gl.UNSIGNED_BYTE, image);
    gl.generateMipmap(gl.TEXTURE_2D);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER,gl.NEAREST_MIPMAP_LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.NEAREST);
    gl.uniform1i(gl.getUniformLocation(program, "uTexMap"), 0);
}
```

Color Texture Mapping（cont.）

3. Texture sampling纹理采样

- assign sampler to texture object 设置采样器给纹理对象
- specify texture sampling parameters 设置纹理对象的采样的参数
- sampling in fragment shader用采样器进行采样(在片元着色器中)

```
<script id="fragment-shader" type="x-shader/x-fragment">
#version 300 es
precision mediump float;
in vec4 vColor;
in vec2 vTexCoord;
out vec4 fColor;
uniform sampler2D uTextureMap;

void
main()
{
    //fColor = vColor * texture(uTextureMap, vTexCoord);
    fColor = texture(uTextureMap, vTexCoord);
}
</script>
```

Color Texture Mapping（cont.）

- 单纹理 vs.多纹理
  - textureCube1: texture map of a gif image onto cube. 纹理图是读取”图像文件”，表面着色采用**“贴花”**

```
fColor = texture(uTextureMap, vTexCoord);
```

  - textureCube2: texture mapping checkerboard onto cube纹理图案换成”黑白棋盘:，表面着色采用“**调制”**: 颜色*纹理色

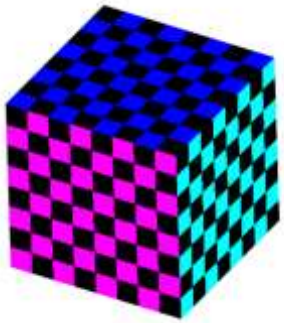
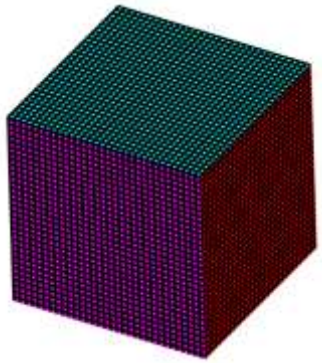
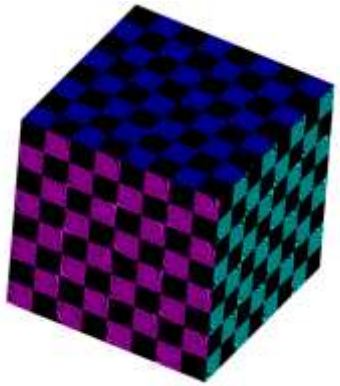
```
fColor = vColor*texture(uTextureMap, vTexCoord);
```

  - textureCubev3: texture map onto cube using two texture images multiplied together in fragment shader. 颜色*纹理色1* 纹理色2

```
fColor = vColor*(texture(uTex0, vTexCoord)*texture(uTex1, vTexCoord));
```

  - textureCube4: texture map with two texture units. First applies checkerboard, second a sinusoid.颜色*纹理色1* 纹理色2

```
fColor = vColor*(texture(uTex0, vTexCoord)*texture(uTex1, vTexCoord));
```

Outline

- Why Texture Mapping
- What is Texture Mapping
- **Basic Mapping Strategy**
  - **Coordinates Mapping**
    - Forward vs. Backward Mapping
    - 2D vs. 3D Mapping
  - **Texture Sampling**
    - Point Sampling vs. Area Sampling
    - Magnification vs. Minification
- Texture Mapping Implementation
  - Color Texture Mapping
  - **Reflection Texture Mapping**

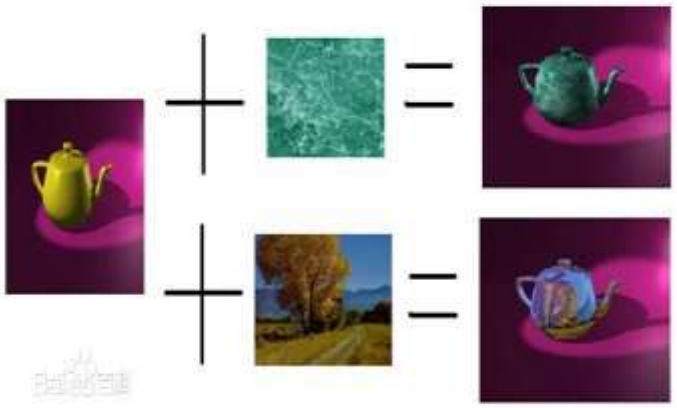
Reflection /Environment Maps（反射/环境映射）

- Uses a picture of the environment for texture maps
- Allows simulation of highly specular surfaces

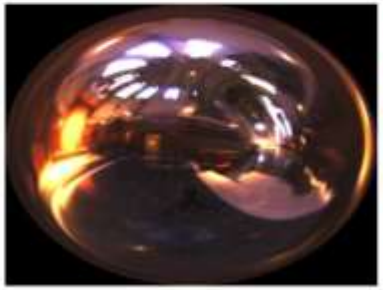
Light from the environment

Rendering with the environment

[Blinn & Newell 1976]



Environmental Lighting

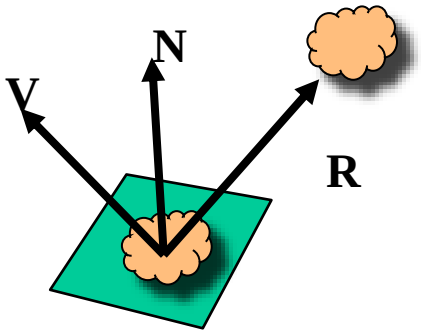


Environment map (left) used to render realistic lighting

The University of New Mexico

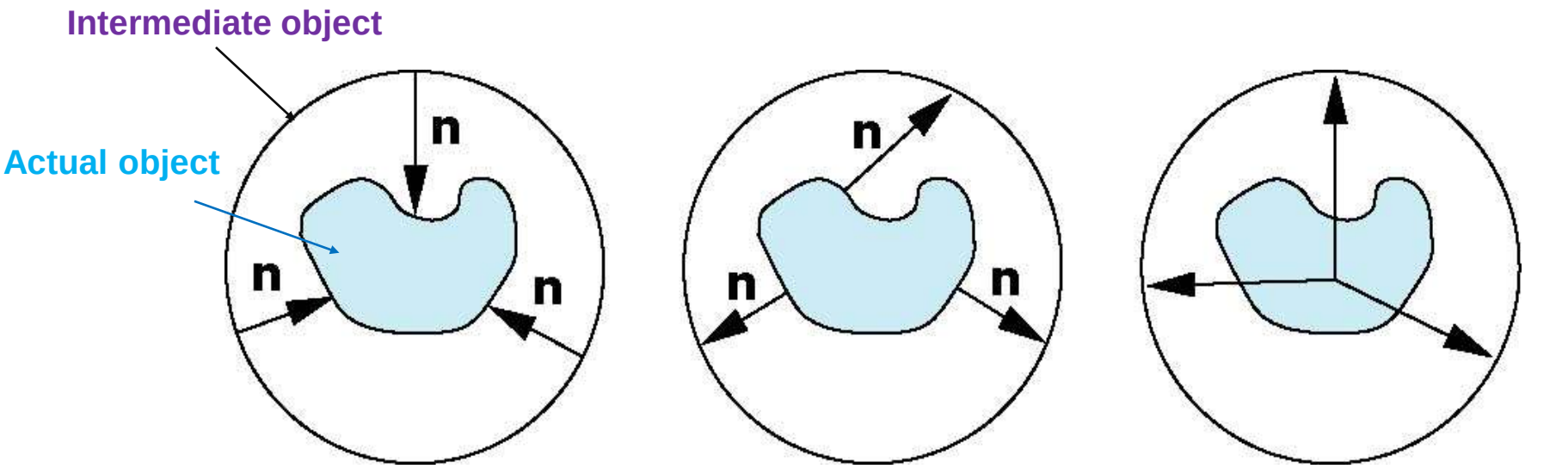
Reflection /Environment Maps（cont.)

- 高抛光表面具有镜面反射特点，可反射周围的环境，
- **环境贴图/反射贴图可替代光线跟踪产生视觉上可以接受的近似效果且计算量小**
- **映射方法：Two-part mapping**
  - 根据物体表面P点视线V和法向量N，可计算得到反射方向R。
  - 跟踪反射方向R直到它与环境对象相交为止,将该“环境纹理”贴到物体表面。



Reflection /Environment Maps（cont.)

- **Two-part mapping**
  - ➢ Second Mapping: 纹理从“中间对象表面”映射到“实际对象”
    - Map from intermediate object to actual object



normals from intermediate object to actual object

normals from actual object to intermediate object

vectors from center of intermediate object

Two-part mapping:First: (s,t)~(x',y',z') , second (x',y',z') ~ (x,y,z) // (x',y',z') is intermediate object

Reflection /Environment Maps（cont.)

- **纹理坐标的映射方式：**
  - **Two-part mapping（两阶段映射）**

    First Mapping: 将“纹理”映射到“中间对象表面”

    map the texture to a simple intermediate surface（such as 圆柱面Cylindrical ,球面Spherical ,盒面Box）.

    Find f: (s,t)~(x’,y’,z’)

    Second Mapping: 纹理从“中间对象表面”映射到“实际对象”

    Map from intermediate object to actual object :

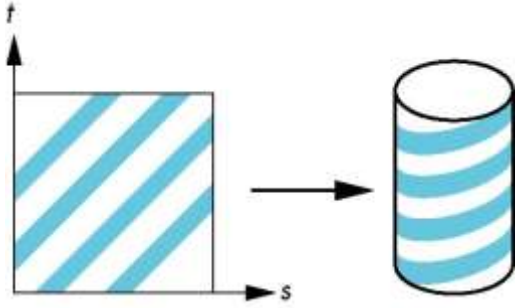
    Find f:(x’,y’,z’) ~（x,y,z）

Reflection /Environment Maps（cont.)

The University of New Mexico

➢Two-part mapping(cont.)

- ➢First Mapping: 将“纹理”映射到“中间对象表面”
  - ➢a simple intermediate surface is parametric cylinder参数 圆柱面



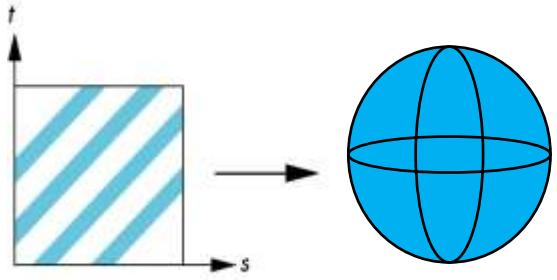
- ◆ (s,t) ~ (u,v) ~（x’,y’ ,z’）
  - ➢ maps rectangle in u,v space to cylinder of radius r and height h in world coordinates
    - X’ = r cos 2π u
    - Y’ = r sin 2πu
    - Z’ = v/h
  - ➢ maps from texture space
    - s = u
    - t = v

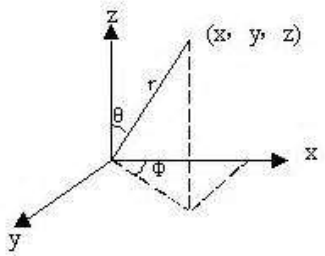
Reflection Texture Mapping（cont.)

➢ Two-part mapping(cont.)

➢First Mapping: 将“纹理”映射到“中间对象表面”

- a simple intermediate surface is parametric sphere(参数球面）





(s,t)~(u,v)~（x’,y’,z’）

$$X' = r \sin \pi u \cos 2\pi v$$
$$Y' = r \sin \pi u \sin 2\pi v$$
$$Z' = r \cos \pi u$$

$$s = u$$
$$t = v$$

$$x(u,v) = r \sin \phi \cos \theta$$
$$y(u,v) = r \sin \phi \sin \theta$$
$$z(u,v) = r \cos \phi$$

$360 \geq \theta \geq 0$

$180 \geq \phi \geq 0$

θ constant: circles of constant longitude

ϕ constant: circles of constant latitude

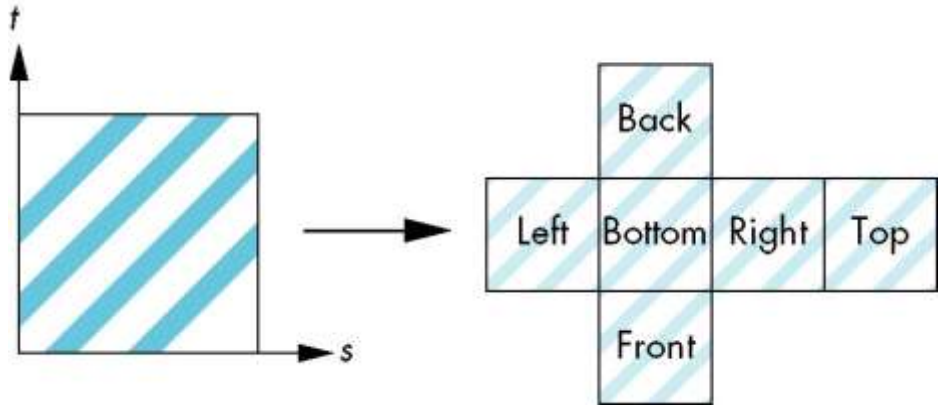
The University of New Mexico

Reflection /Environment Maps（cont.)

➢ Two-part mapping(cont.)

➢First Mapping: 将“纹理”映射到“中间对象表面”

- a simple intermediate surface is Box（盒子表面）
  - ***Easy to use with simple orthographic projection***
  - ***used in environment maps***



(s,t)~(u,v)~（x’,y’,z’）

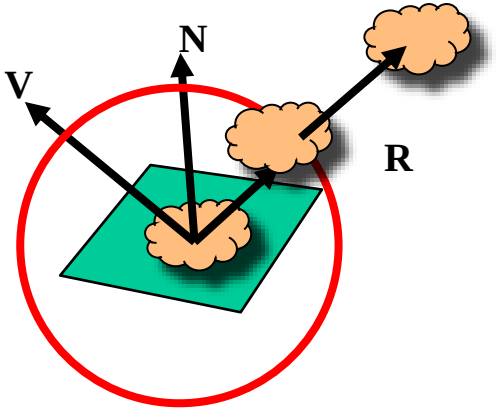
Example: front face:
X’ =au
Y’ = bv
Z’=c //a,b,c are constants

s = u
t = v

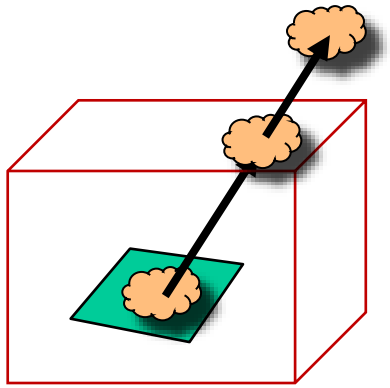
Reflection /Environment Maps（cont.)

➢采用两阶段映射方法实现环境映射

- 第一阶段:把环境模拟成图像纹理，并映射到中间表面（柱面，球面，立方体表面）
- 第二阶段:把中间表面上的纹理“环境映射”到物体表面（R求交）



Mapping to Hemisphere



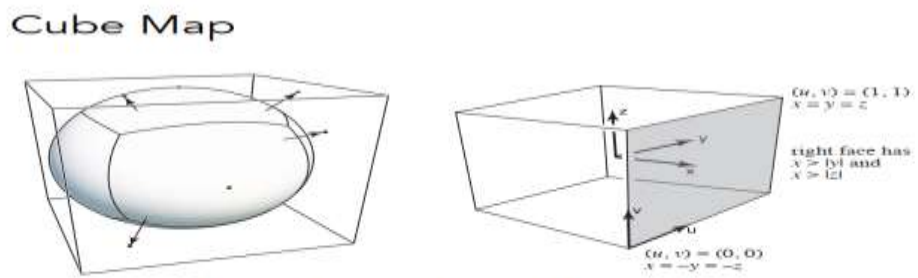
Mapping to Cube

webGL只支持cubeMap(立方体贴图)

Reflection /Environment Maps（cont.)

The University of New Mexico

- **采用两阶段映射方法实现环境映射~实例**
  1. Create a cubeMap texture object (创建立方体纹理对象）
  2. Map the cubeMap texture to the mirror object surface according on a vector(根据向量R，将立方体纹理映射到镜面物体表面)



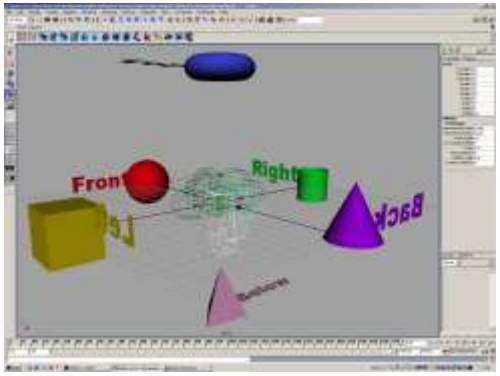
Reflection/Environment Maps（cont.

The University of New Mexico

➢**采用两阶段映射方法实现环境映射~实例（cont.)**

step1. Create a cubeMap texture object(**创建立方体纹理对象**)

立方体包围盒B有6个镜面，需要相机朝6个方向摆放6次才能得到立方体包围盒6个镜面的图像



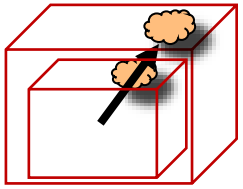

```
/*********************生成立方体纹理对象*********************************/
function configureCubeMap() {
    gl.activeTexture(gl.TEXTURE0);

    cubeMap = gl.createTexture();
    gl.bindTexture(gl.TEXTURE_CUBE_MAP, cubeMap);
    gl.texParameteri(gl.TEXTURE_CUBE_MAP, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
    gl.texParameteri(gl.TEXTURE_CUBE_MAP, gl.TEXTURE_MIN_FILTER, gl.LINEAR);

    gl.uniform1i(gl.getUniformLocation(program, "cubeSampler"), 0);

    var faces = [
        ["./skybox/right.jpg", gl.TEXTURE_CUBE_MAP_POSITIVE_X],
        ["./skybox/left.jpg", gl.TEXTURE_CUBE_MAP_NEGATIVE_X],
        ["./skybox/top.jpg", gl.TEXTURE_CUBE_MAP_POSITIVE_Y],
        ["./skybox/bottom.jpg", gl.TEXTURE_CUBE_MAP_NEGATIVE_Y],
        ["./skybox/front.jpg", gl.TEXTURE_CUBE_MAP_POSITIVE_Z],
        ["./skybox/back.jpg", gl.TEXTURE_CUBE_MAP_NEGATIVE_Z]
        ];

    for (var i = 0; i < 6; i++) {
        var face = faces[i][1];
        var image = new Image();
        image.src = faces[i][0];
        image.onload = function (cubeMap, face, image) {
            return function () {
                gl.texImage2D(face, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, image);
            }
        }(cubeMap, face, image);
    }
}
```

Reflection/Environment Maps（cont.)

- **采用两阶段映射方法实现环境映射~实例（cont.)**
  - Step2：Map the cubeMap texture to the mirror object surface according on a vector(**根据某向量将立方体纹理映射到某镜面物体表面**)
  - Example1: mapping cubeMap to a box to form “skybox”
  - Example2: reflection mapping to a mirror ball
  - Example3: both Ex1 and Ex2

Reflection/Environment Maps（cont.)

- Example1: 立方体贴图绘制“天空盒”
  - “天空盒”:是场景的边界背景，是包裹着相机和其它场景物体的大盒子
  - 把立方体纹理映射到一个立方体上作为场景的边界（背景），即:“天空盒”
  - 此时直接取”观察向量V,对 立方体纹理进行采样 (下面代码用R表示V)



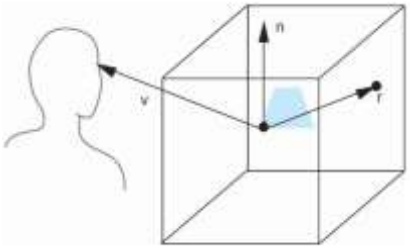
```
colorCube(10);
```

```
if(u_flag==0){//画天空盒
    mat4 skyViewMatrix=mat4(mat3(u_ViewMatrix));
    gl_Position=u_ProjectionMatrix * skyViewMatrix  * vPosition;
    //!计算得到WC下的向量R
    pos =vPosition.xyz;
    R=normalize(pos);
};
```

```
gl_FragColor=textureCube(cubeSampler, R); //采样
```

Reflection/Environment Maps（cont.）

- Example2: 立方体贴图映射到 镜面物体
  - Map the cubeMap texture to the mirror object surface according to R
  - R=**reflect（V,N)**　**//R = 2(N·V)N-V，GLSL support function relect**



```
if(u_flag==1){//画中间物体
    gl_Position=u_ProjectionMatrix*u_ViewMatrix*u_ModelMatrix*vPosition;
    //计算得到WC下的向量L,N,V
    pos = (u_ModelMatrix * vPosition).xyz;
    V =-normalize(pos);
    N= normalize(u_NormalMatrix*vNormal).xyz;//N=inverse(transpose(u_ModelMatrix))
    R=reflect(V,N);
};
```

```
<script id="fragment-shader" type="x-shader/x-fragment">
    precision mediump float;
    varying vec4 fColor;

    uniform samplerCube  cubeSampler;//盒子纹理采样器
    varying vec3 R; //反射向量

    void main() {

            gl_FragColor=textureCube(cubeSampler, R);  //反射采样
    }
</script>
```

The University of New Mexico

Reflection/Environment Maps（cont.)

- Example3:“天空盒”和“镜面物体”都绘制

```
colorCube(10);
sphere(0.5);
```

```
void main() {
   vec3 pos, N, L, V;
   if(u_flag==0){//画天空盒
      mat4 skyViewMatrix=mat4(mat3(u_ViewMatrix));
      gl_Position=u_ProjectionMatrix * skyViewMatrix  * vPosition;
      //直接取反射向量R=v2
      pos = vPosition.xyz;
      R=normalize(pos);
   };

   if(u_flag==1){//画中间物体
      gl_Position=u_ProjectionMatrix*u_ViewMatrix*u_ModelMatrix*vPosition;
      //计算得到vc下的向量R=reflect(V,N)
      pos = (u_ModelMatrix * vPosition).xyz;
      V =-normalize(pos);
      N= normalize(u_NormalMatrix*vNormal).xyz;//N=inverse(transpose(u_ViewMatrix))
      R=reflect(V,N);
   };
```

```
<script id="fragment-shader" type="x-shader/x-fragment">
    precision mediump float;
    varying vec4 fColor;
    varying vec3 R; //反射向量
    uniform samplerCube  cubeSampler;//立方体纹理采样器
    void main() {

            gl_FragColor=textureCube(cubeSampler, R); //采样
    }
</script>
```

Reflection/Environment Maps（cont.)

绘制“天空盒”需注意的问题

- 首先，天空盒不需要进行模型变换M。
  - 解决方案：天空盒不进行模型变换

- 其次，相机不管如何移动，可以始终保持在天空盒的中心点，所以需要修改viewMatrix（这点非必须）
  - 的位移，而保存相机视角的变换

$$\begin{bmatrix} a & 0 & 0 & Tx \\ 0 & b & 0 & Ty \\ 0 & 0 & c & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ax + Tx \\ by + Ty \\ cz + Tz \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \Rightarrow \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ax \\ by \\ cz \\ 1 \end{bmatrix}$$