Computer Graphics

Texture Mapping

Teacher: A.prof. Chengying Gao(高成英)

E-mail: mcsgcy@mail.sysu.edu.cn

School of Data and Computer Science

Basic Conception of Texture Mapping

Limitations

- So far, every object has been drawn either in a solid color, or smoothly shaded between the colors at its vertices.
  - Similar to painting

Generated with Blue Moon Rendering Tools — www.bmrt.org

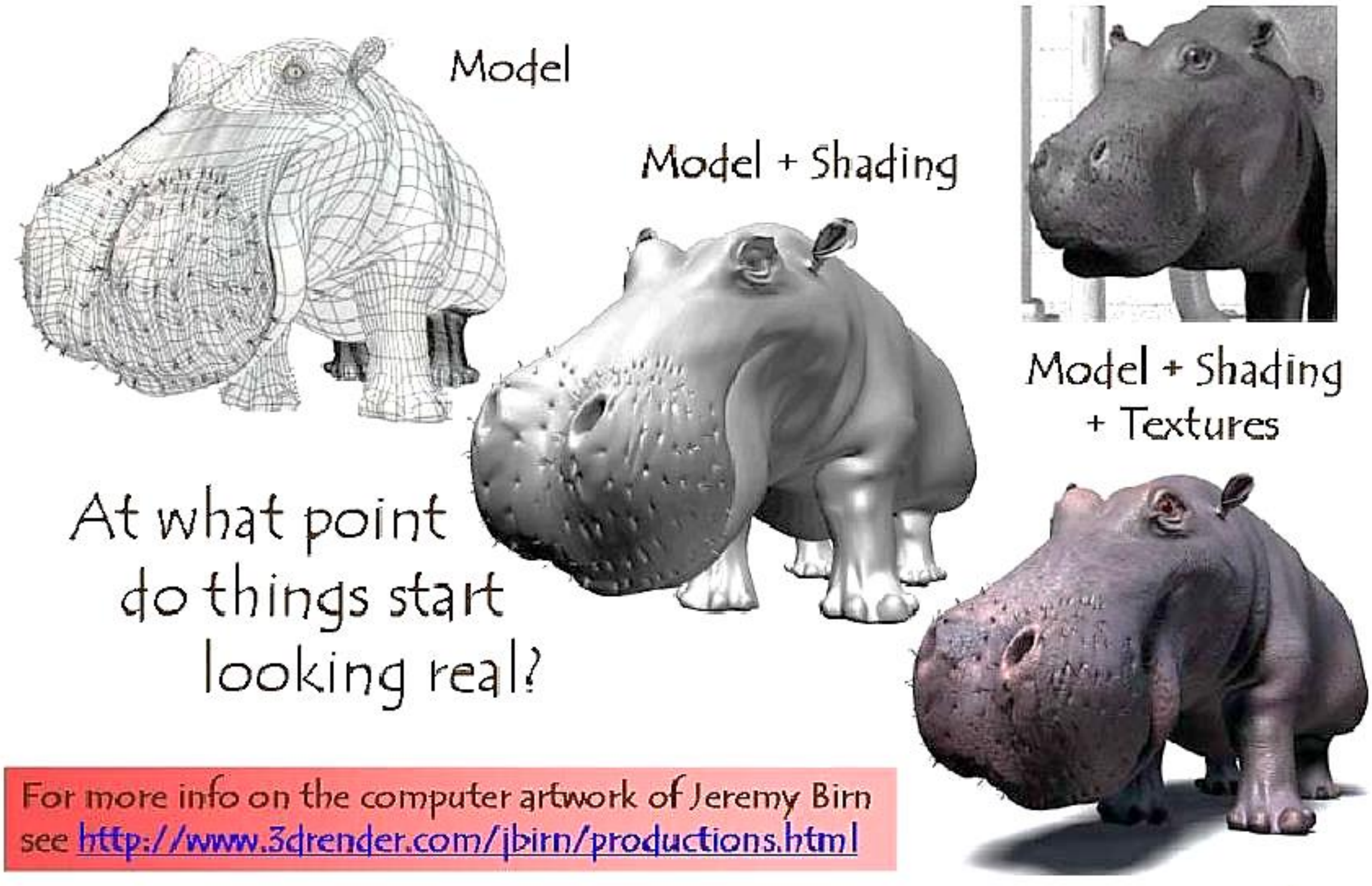
Limitations

- Even though the graphics card can display up to ten million polygons per second,It's difficult to simulate all the phenomenon in nature.
  - Cloud
  - Grass
  - Leaf
  - Landforms
  - Hair
  - Fire and Water

The Importance of Surface Texture

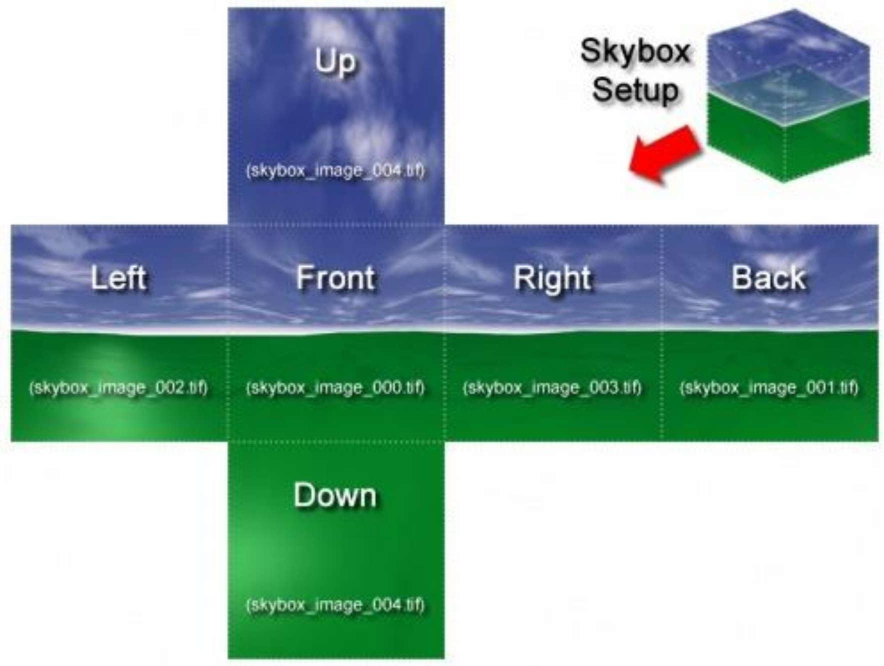
- So far, every object has been drawn either in a solid color, or smoothly shaded between the colors at its vertices.
  - –Similar to painting
- Texture Mapping applies a variation to the surface properties of the object instead
  - –Similar to surface finishing, example like wallpaper on a wall surface or sticking a label onto a bottle.

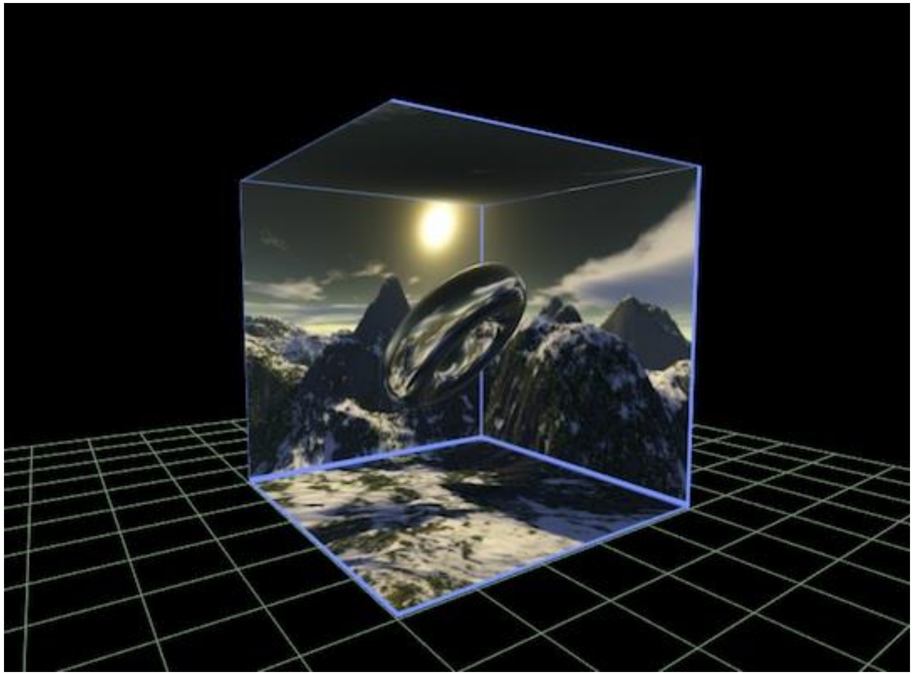
The Quest for Visual Realism



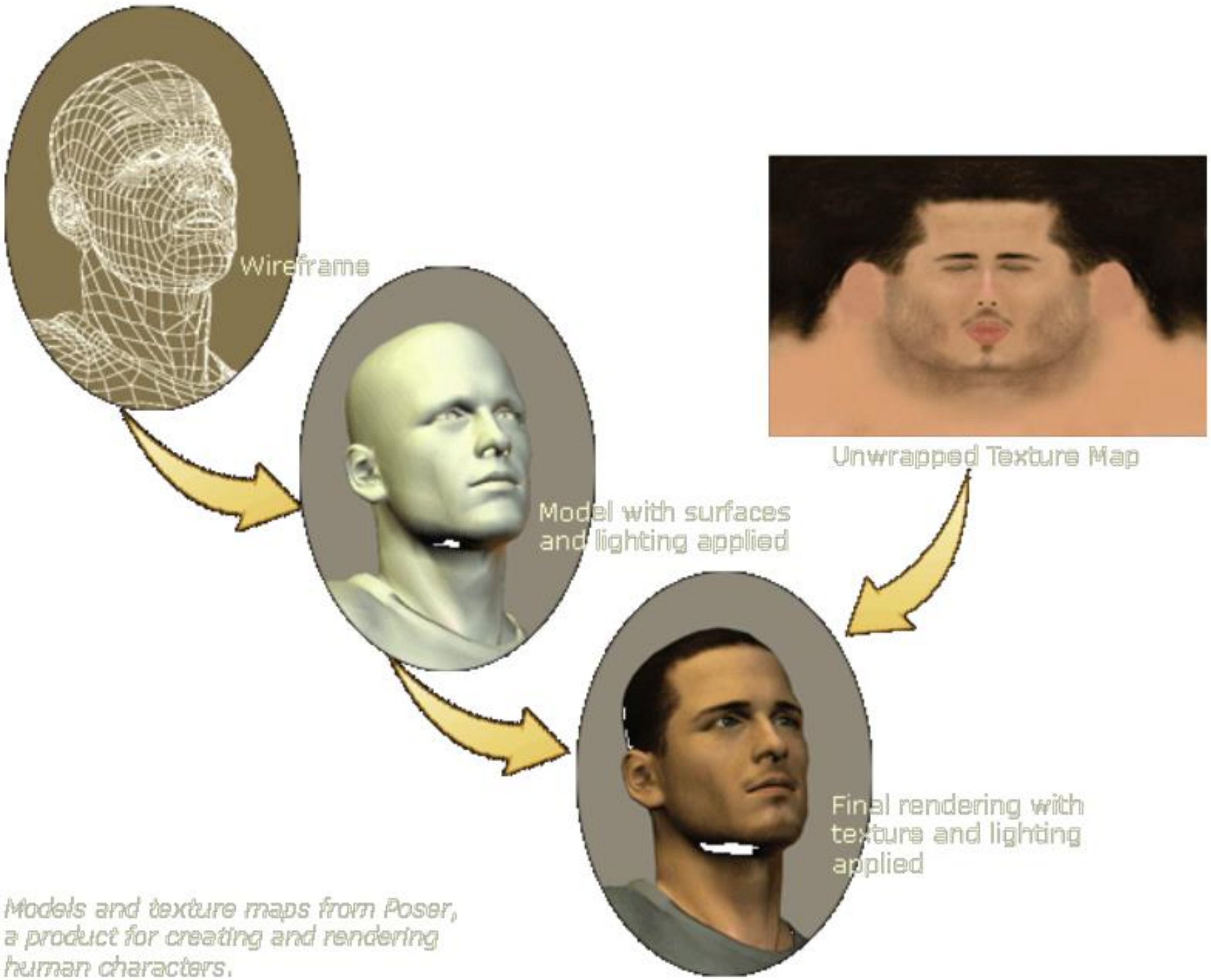
Texturing Example

- SkyBox





Texturing Example



Model of Orange

- Consider the method to create model of orange (or some of other fruits).
- Coloring orange in a simple sphere?
  - Not reality
- Instead more complicate shape of sphere?
  - Too many polygons to display the dimples in the surface

Why Texture Mapping?

- Texture is variation in the surface attributes
  - like color, surface normals, specularity, transparency, surface displacement etc.
- Computer generated images look more realistic if they are able to capture these complex details
- It is difficult to represent these details using geometric modeling because they heavily increase computational requirements
- Texture mapping is an effective method of “faking” surface details at a relatively low cost

What is texture mapping

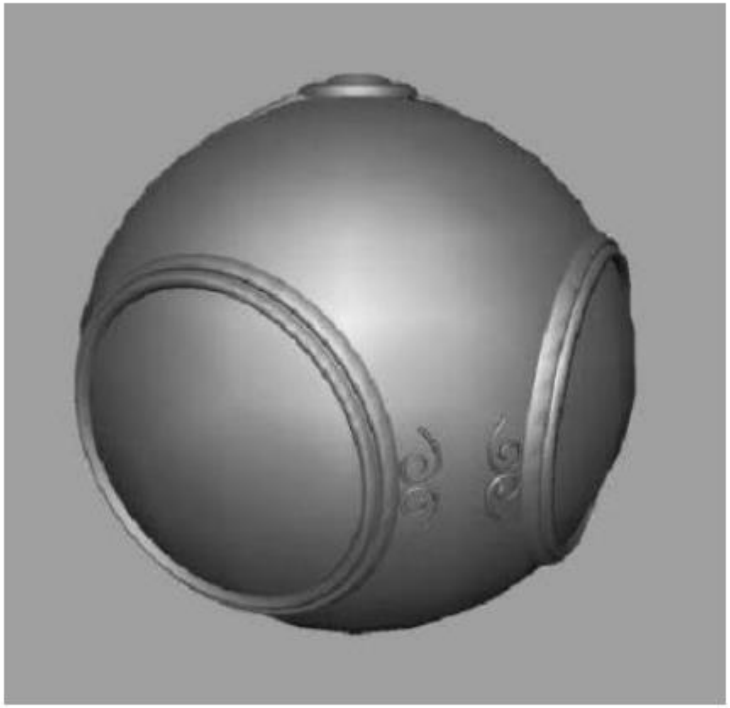
- Texture mapping is the process of transforming a texture on to a surface of a 3D object.
- It is like mapping a function on to a surface in 3D
  - the domain of the function could be 1D, 2D or 3D
  - the function could be represented by either an array or it could be an algebraic function.

Mapping Methods

- **Texture Mapping (纹理贴图)**
  - Using images to fill polygons
- **Environment Mapping (Reflection Mapping, 反射贴图)**
  - Using images of environment to fill polygons
  - Simulate specular surfaces
- **Bump Mapping (凹凸贴图)**
  - Can change the normal vector in displayed model.

Mapping Methods

- Texture Mapping

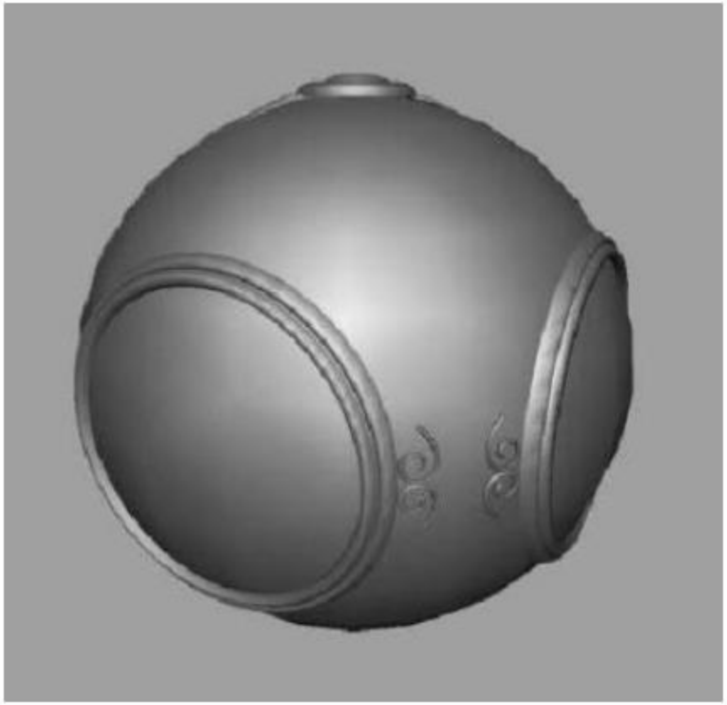


Before Mapping

After Mapping

Mapping Methods

- Environment Mapping (Reflection Mapping)

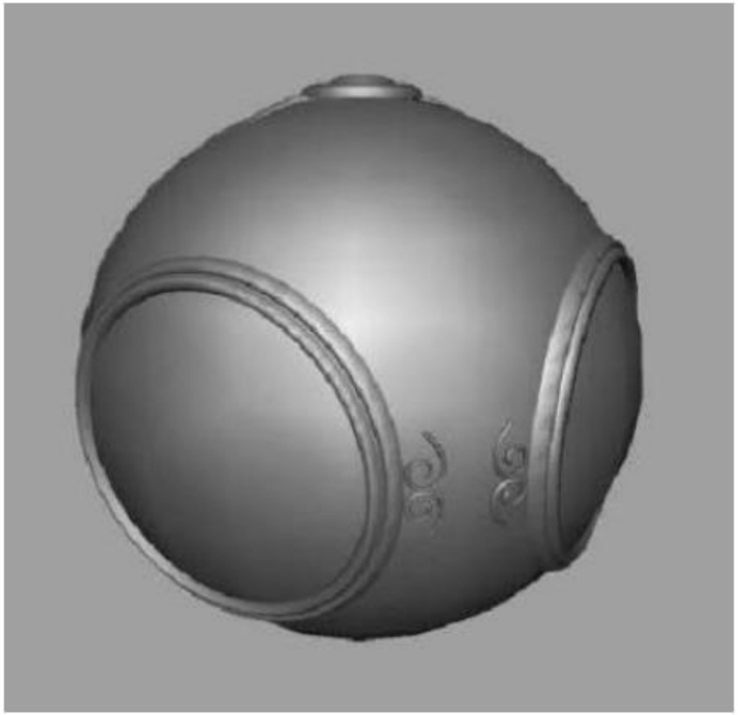


Before Mapping

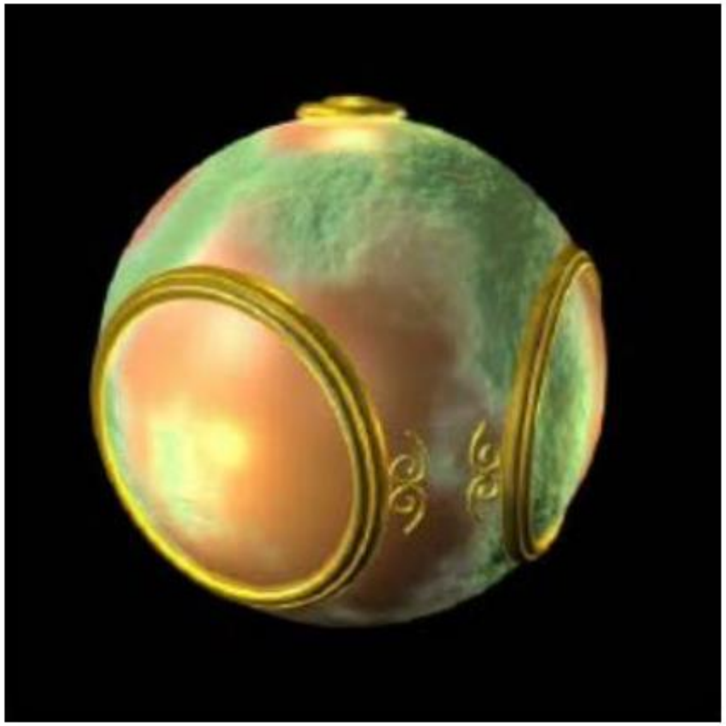
After Mapping

Mapping Methods

- Dump Mapping



Before Mapping



After Mapping

What kind of objects?

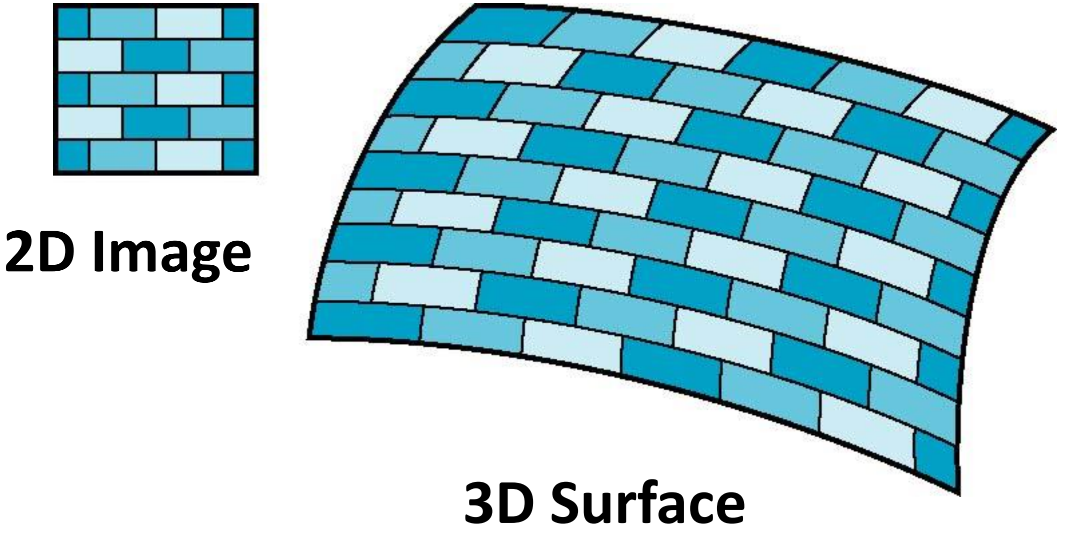
- In general Texture mapping for arbitrary 3D objects is difficult
  - E.g. Distortion (try mapping a planer texture onto sphere)
- It is easiest for polygons and parametric surfaces
- We limit our discussion to Texture mapping polygons

What is typical 2D texture?

- A 2D function represented as rectangular array of data
  - Color data
  - Luminance data ?
  - Color and alpha data
- Each data element in a texture is called a texel (纹素); on screen, a texel may be mapped to
  - A single pixel
  - Part of a pixel (for small polygons)
  - Several pixels (if the texture is too small or the polygon is viewed from very close)

Is it easy？

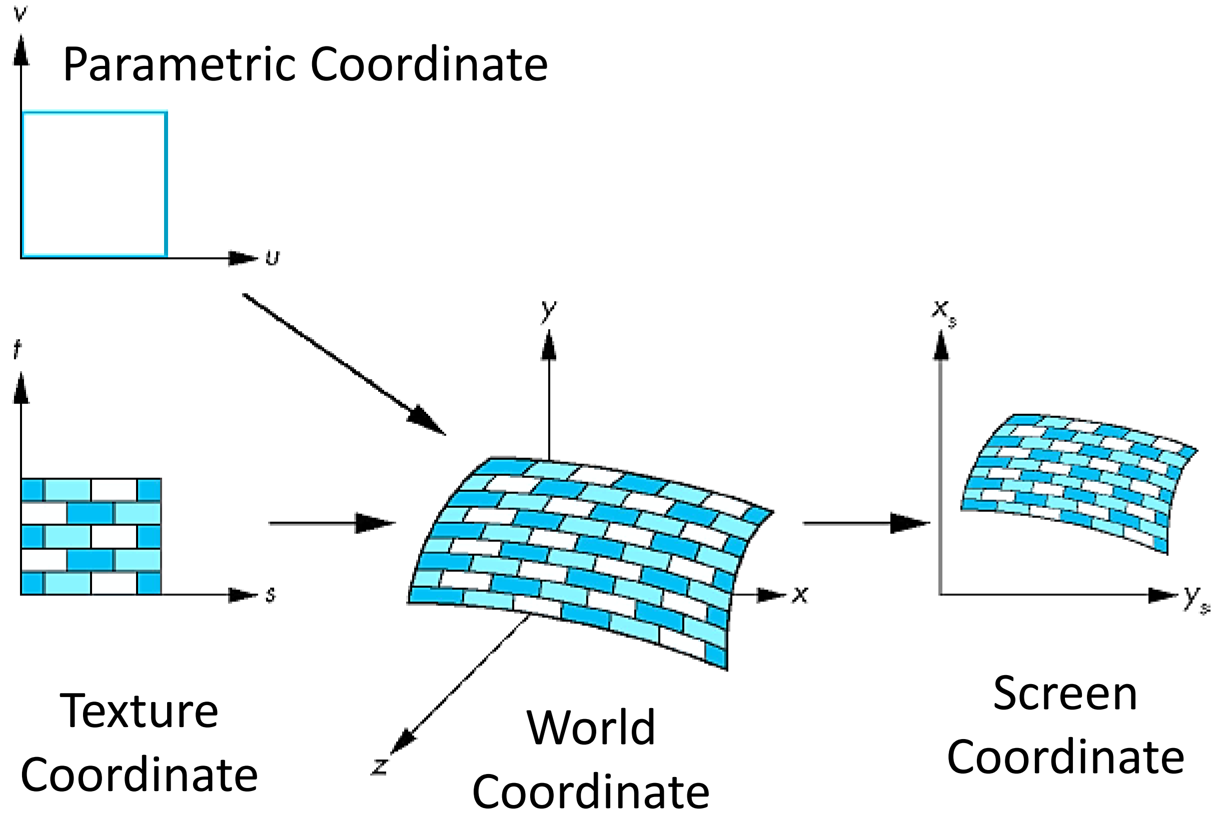
- Although the idea is simple, it requires a sequence of steps that involve mappings among three or four different coordinate systems.



Coordinate systems

- parametric coordinates
  - which we use to help us define curved surfaces
- Texture coordinates
  - which we use to locate positions in the texture
- World coordinate
  - Where we describe the objects upon which the textures will be mapped
- Screen coordinates
  - Where the final image is produced

2D Texture Mapping Frame

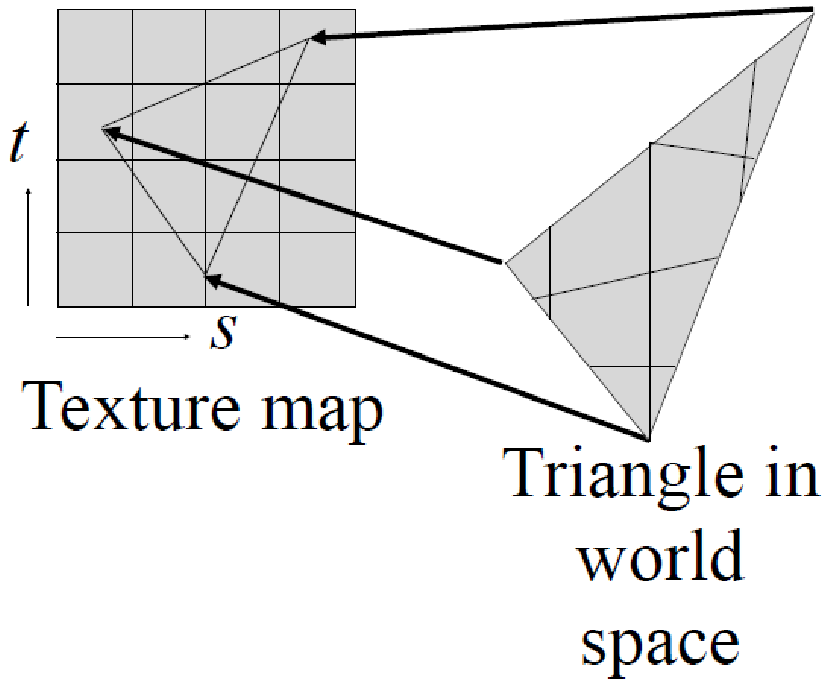


Assigning Texture Coordinates

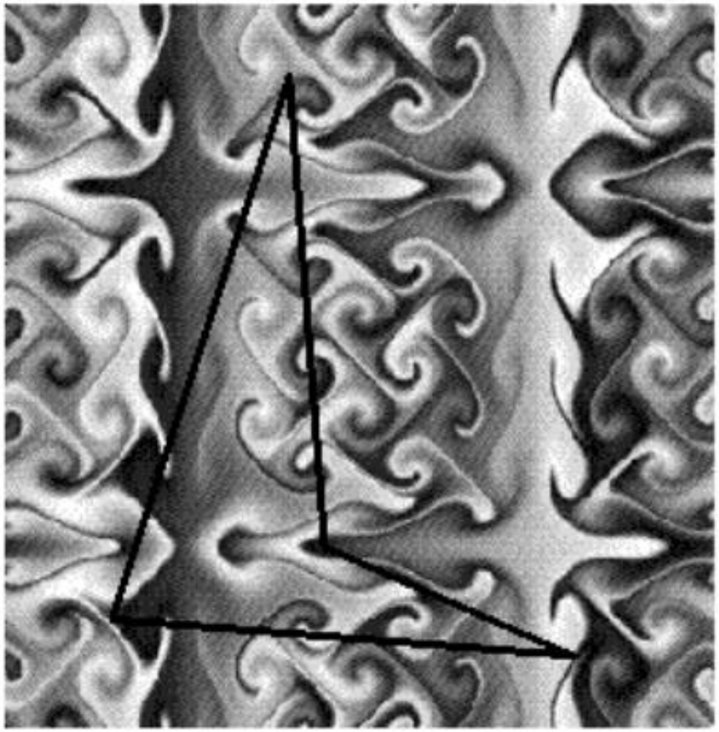
- You must provide texture coordinates for each vertex.
- The texture image itself covers a coordinate space between 0 and 1 in two dimensions usually called *s* and *t* to distinguish them from the *x*, *y* and *z* coordinates of 3D space.
- A vertex's texture coordinates determine which texel(s) are mapped to the vertex.
- Texture coordinates for each vertex determine a portion of the texture to use on the polygon.
- The texture subset will be stretched and squeezed to fit the dimensions of the polygon.

Texture Interpolation

- Specify where the vertices in world space are mapped to in texture space
- Linearly interpolate the mapping for other points in world space
  - Straight lines in world space go to straight lines in texture space



Texture Example

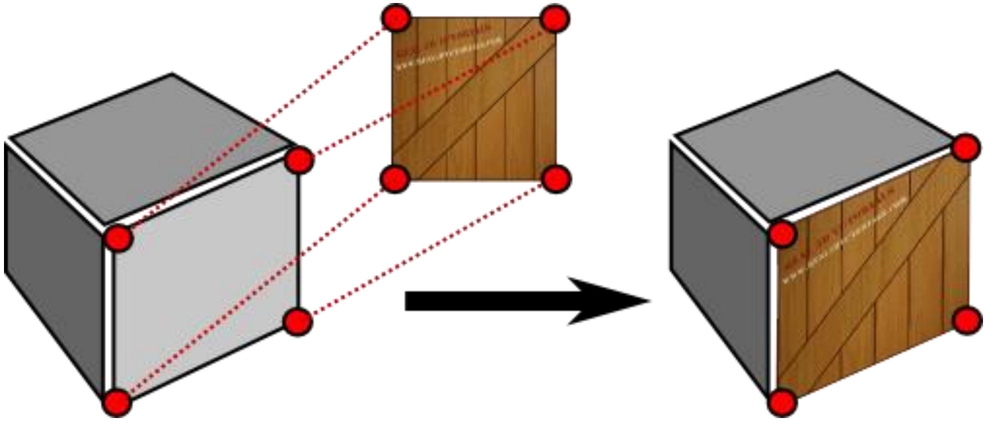


Polygonal texture mapping

- Establish correspondences
- Find compound 2D-2D mapping
- Use this mapping during polygon scan conversion to update desired attribute (e.g. color)

Establish Correspondences

- Usually we specify texture coordinates at each vertex
- These texture coordinates establish the required mapping between image and polygon

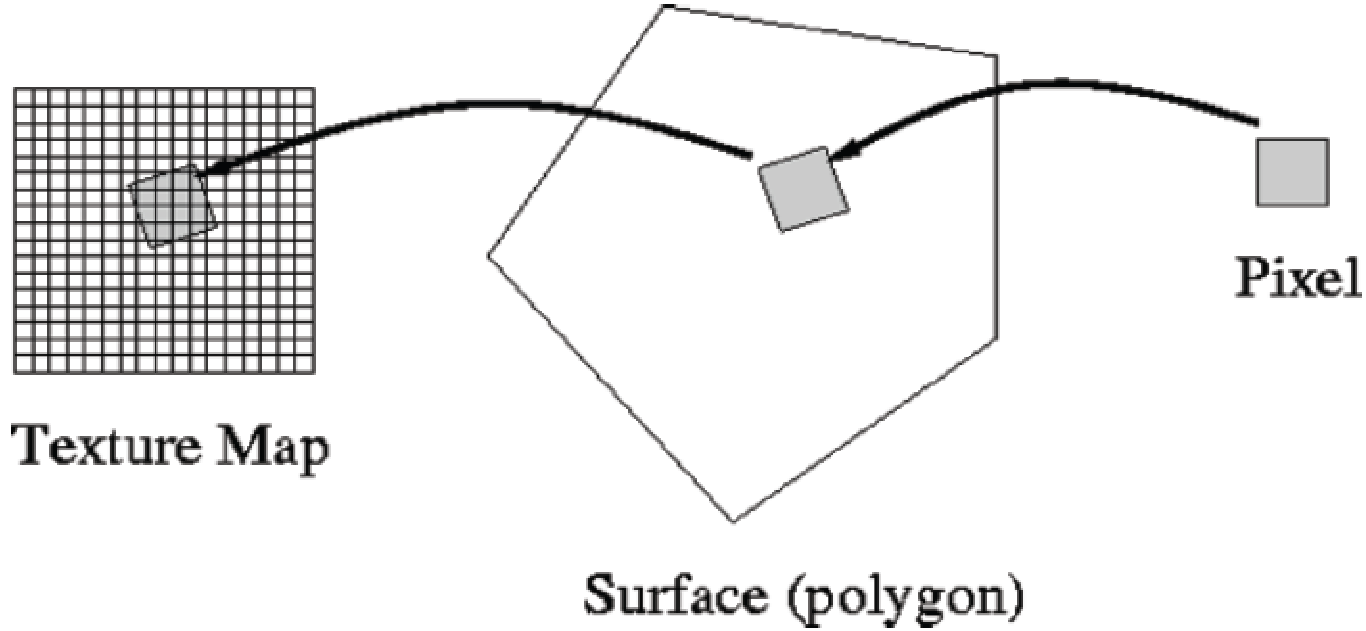


Find compound 2D-2D mapping

- Since the texture is finally seen on screen which is 2D, it makes sense to combine two mappings (from image to 3D space and then from 3D to screen space) into single 2D-2D mapping
- This avoids texture calculations in 3D completely
- This simplifies hardware implementation of graphics pipeline.

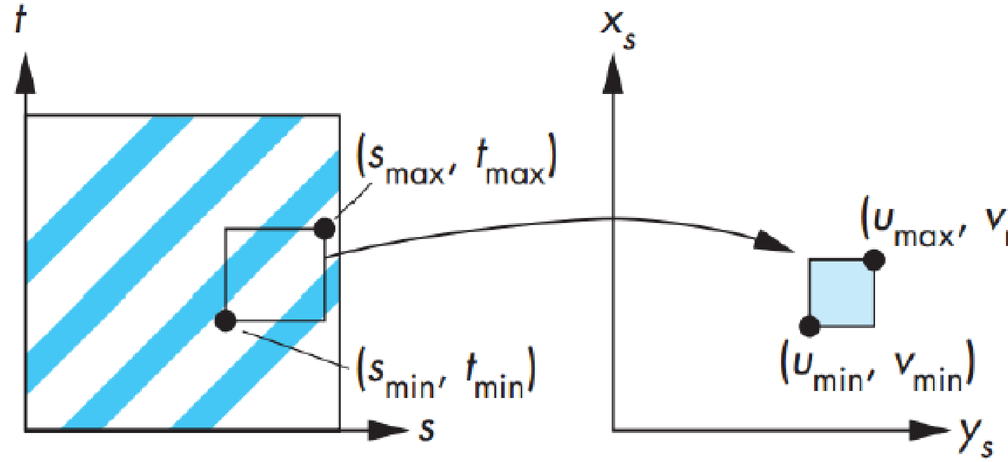
Computing Color in Texture mapping

- Associate texture with polygon
- Map pixel onto polygon and then into texture map
- Use weighted average of covered texture to compute color.



Mapping Problem

- The patch determined by the corners(s_{min},t_{min}) and (s_{max}, t_{max}) corresponds to the surface patch with corners(u_{min}, v_{min}) and (u_{max}, v_{max}).



$$u = u_{min} + \frac{s - s_{min}}{s_{max} - s_{min}}(u_{max} - u_{min}),$$

$$v = v_{min} + \frac{t - t_{min}}{t_{max} - t_{min}}(v_{max} - v_{min}).$$

Characteristics of this method

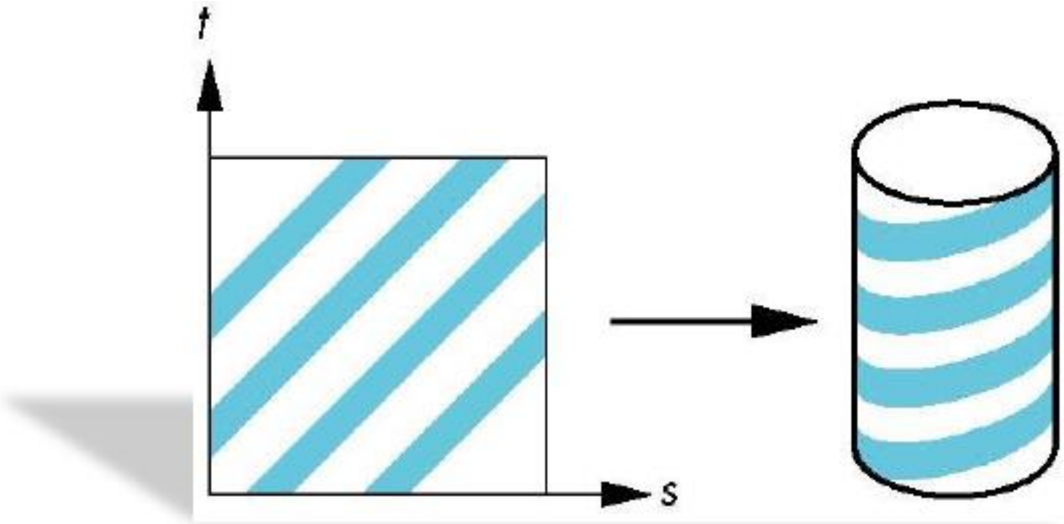
- This mapping is easy to apply.
- But it does not take into account the curvature of the surface.
- Texture patches must be stretched to fit over the surface patch.

Solution

- Another approach to the mapping problem is to use a two-part mapping.
  - The first step maps the texture to a simple three-dimensional intermediate surface.
  - In the second step, the intermediate surface containing the mapped texture is mapped to the surface being rendered

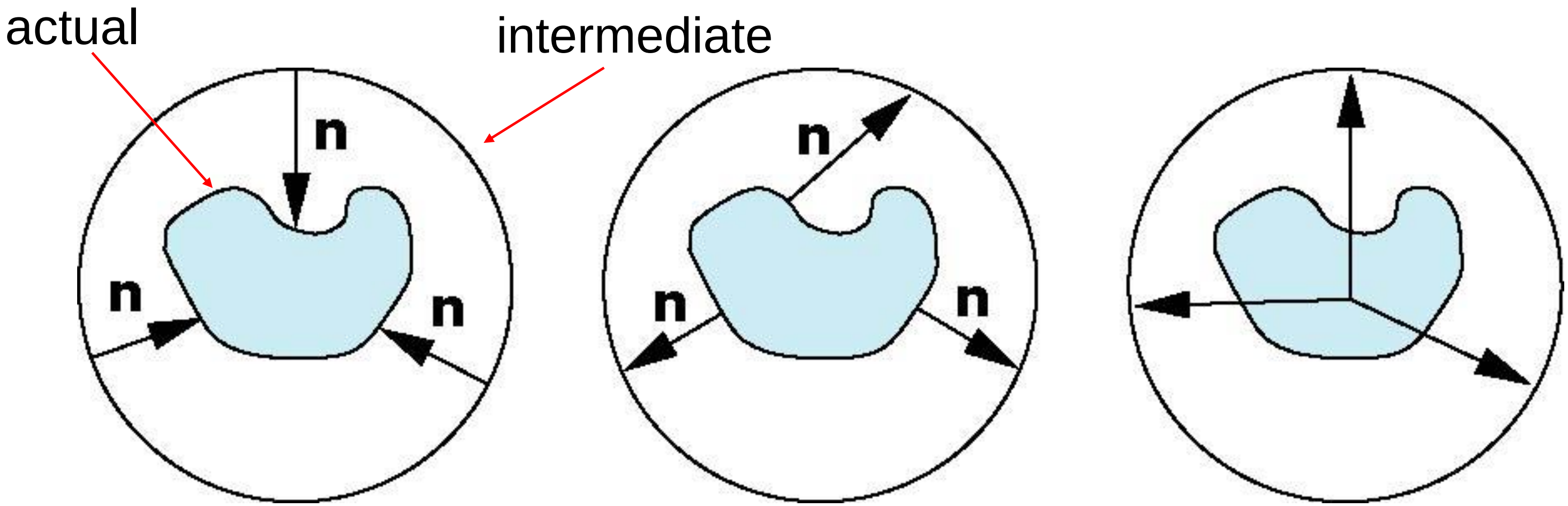
Two-part mapping

- One solution to the mapping problem is to first map the texture to a simple intermediate surface
- Example: map to cylinder



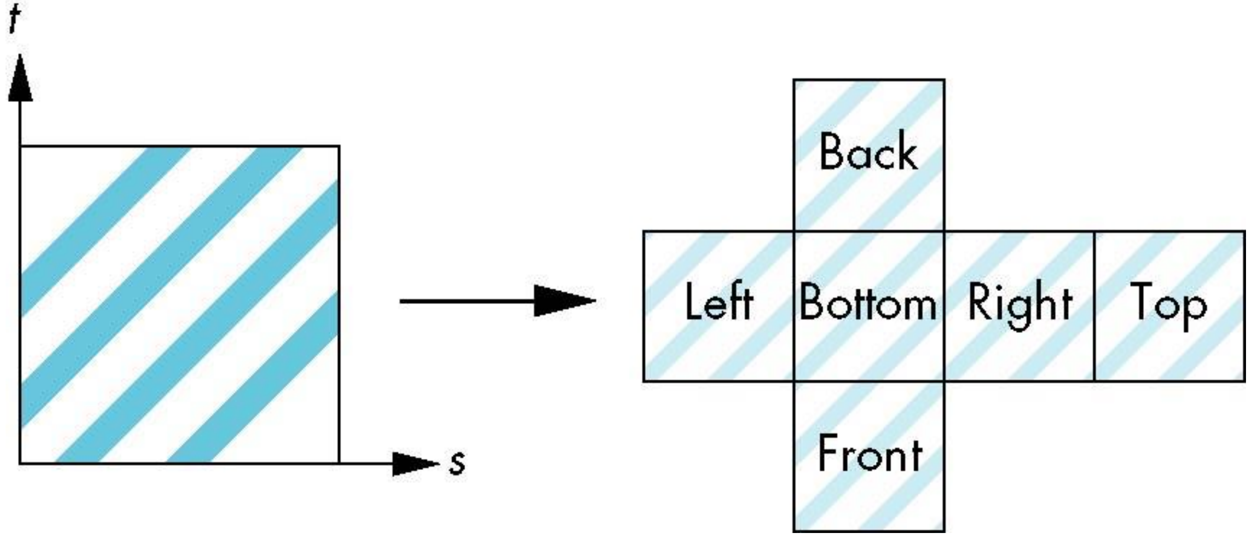
Second Mapping

- Map from intermediate object (中间对象) to actual object (实际对象)
  - Normals from intermediate to actual
  - Normals from actual to intermediate
  - Vectors from center of intermediate

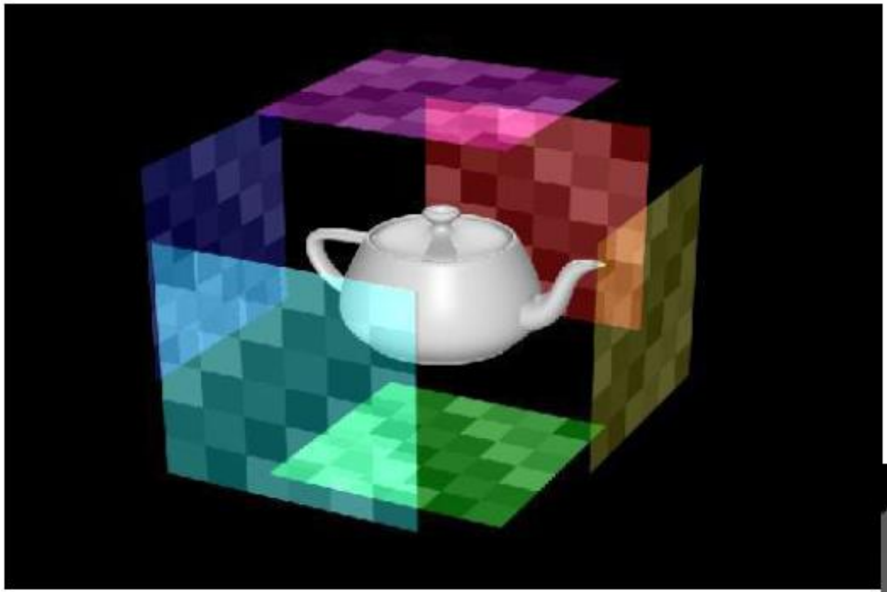


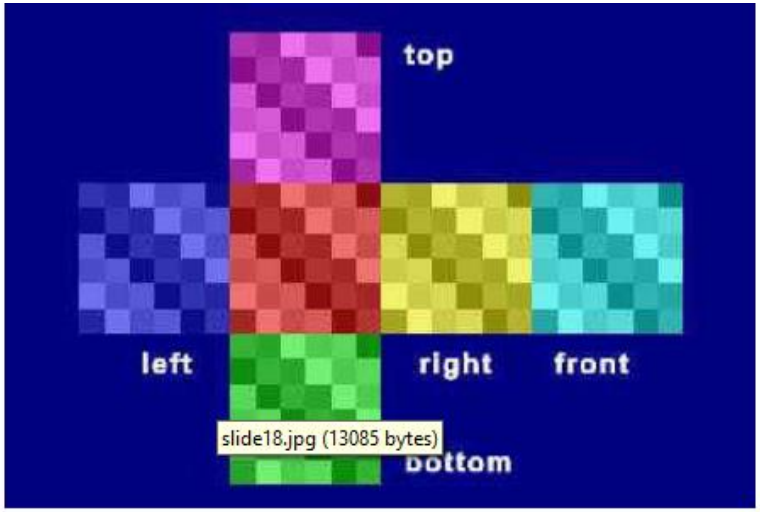
Cube Mapping

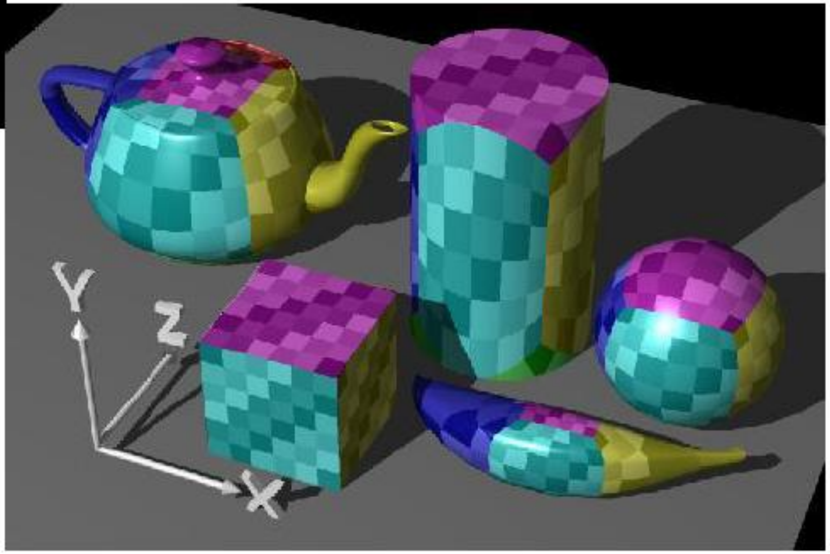
- Easy to use with simple orthographic projection
- Also used in environment maps



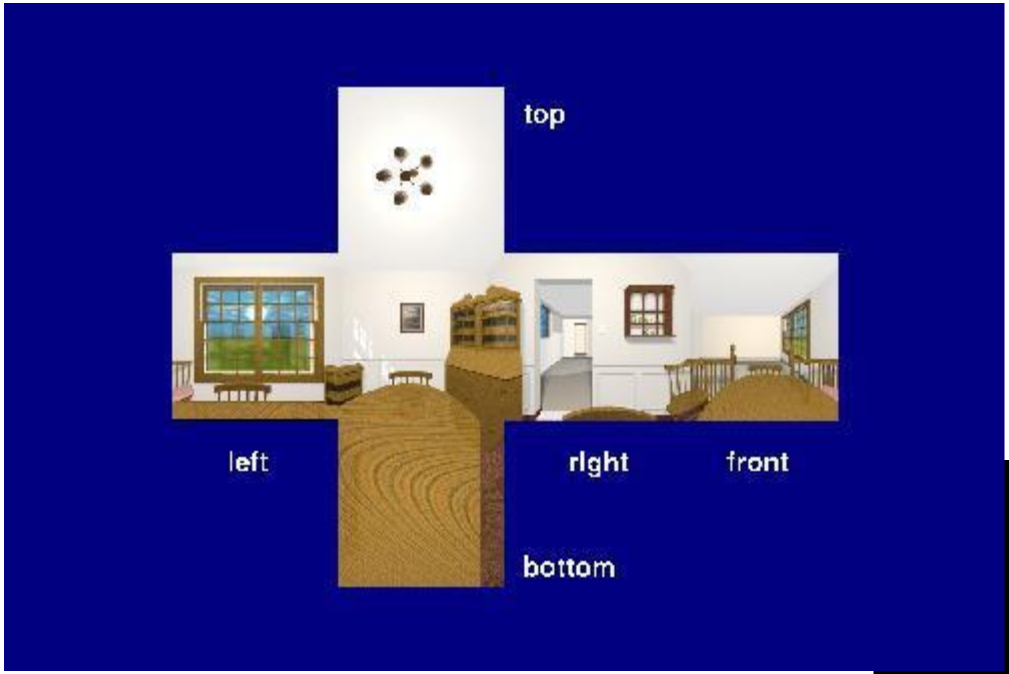
Cube Mapping







Cube Mapping



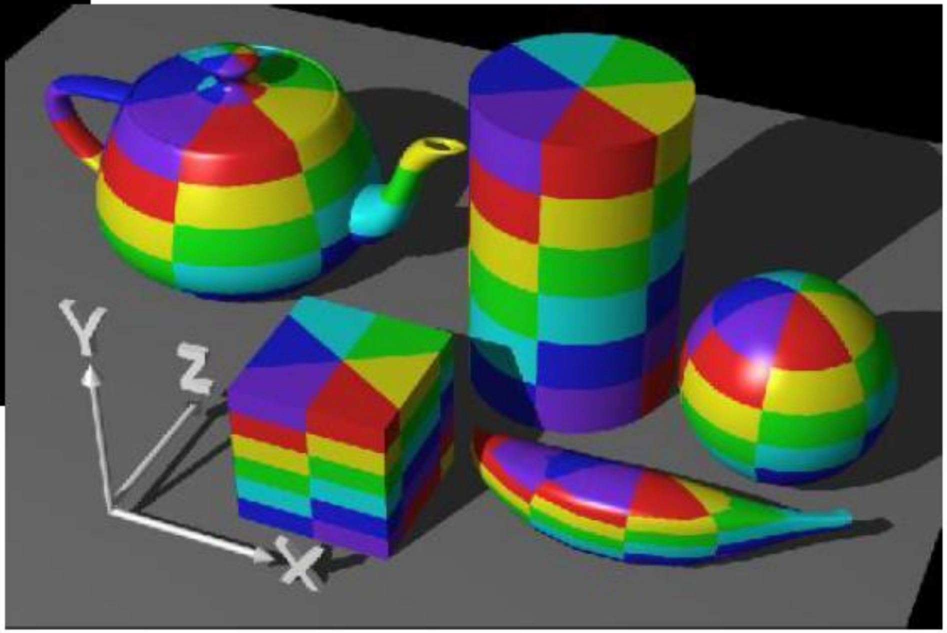
Cylinder Mapping

- 假设纹理坐标在单位正方形$[0,1]^2$内变化，圆柱高h，半径r
- 那么圆柱的参数方程为

  $x = r\cos(2\pi s)$, $y = r\sin(2\pi s)$, $z = ht$
- 从纹理坐标到圆柱面上没有变形
- 适合于构造与无底的圆柱面拓扑同构的曲面上的纹理

Cylindrical Mapping

- **Cylinder: r, θ, z with (s,t) = (θ/(2π),z)**
  - ✓ Note seams when wrapping around (θ = 0 or 2π)



Sphere Mapping

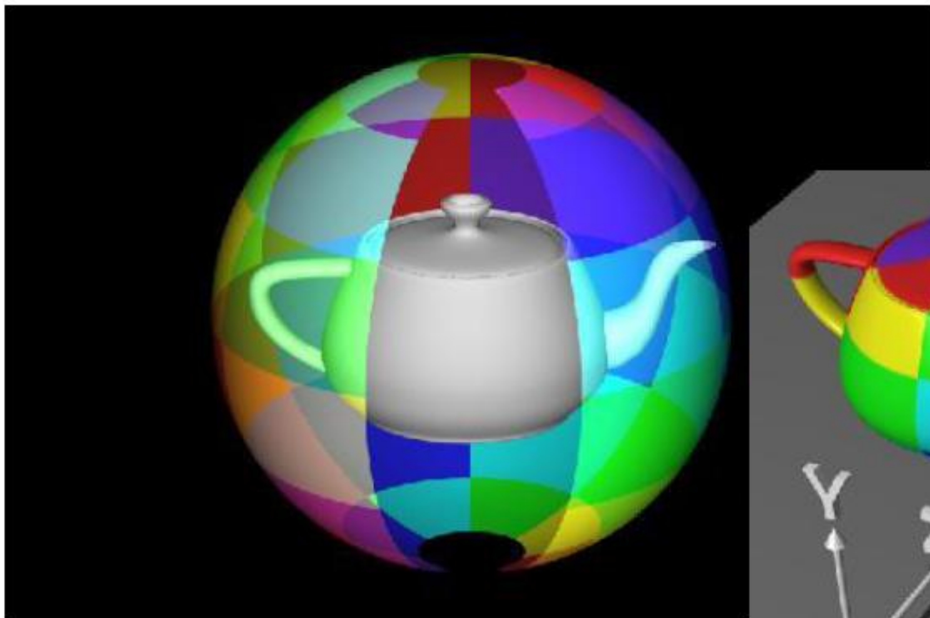
- 球的参数方程

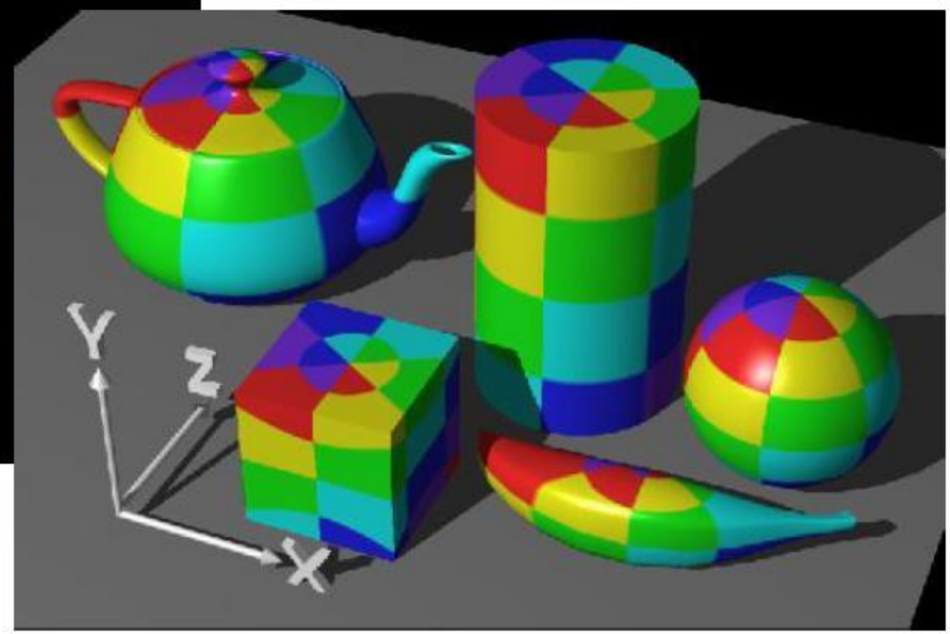
  $x = r\cos(2\pi s),\ y = r\sin(2\pi s)\cos(2\pi t),\ z = r\sin(2\pi s)\sin(2\pi t)$

- 类似于地图绘制中的映射
  - 肯定有变形
- 用在环境映射中

Spherical Mapping

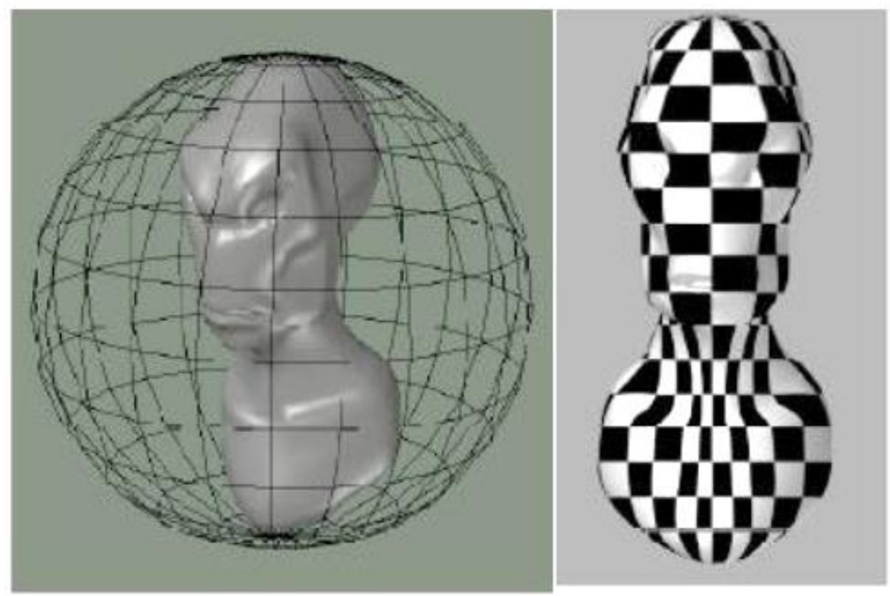
- **Convert to spherical coordinates: use latitude/long**
  - Singularities at north and south poles





Spherical Mapping

- Use, e.g., spherical coordinates for sphere
- Place object in sphere
- "shrink-wrap" sphere to object



Problems in Texture Mapping

- Aliasing artifacts
  - Due to point sampling
- Perspective distortions
  - Since we did not take into account perspective transformation

Taking care of aliasing

- What causes aliasing artifacts
  - High frequency signals
- Typical solutions
  - Sample at higher rates
  - Prefilter textures using low pass filters

Optimization

- Since most of the times, textures are known a priori, we can create various levels of these prefiltered textures in a preprocess.
- Then at run time, we fetch the required level of mipmap and apply that texture.
- This is known as mipmapping (分级细化贴图)

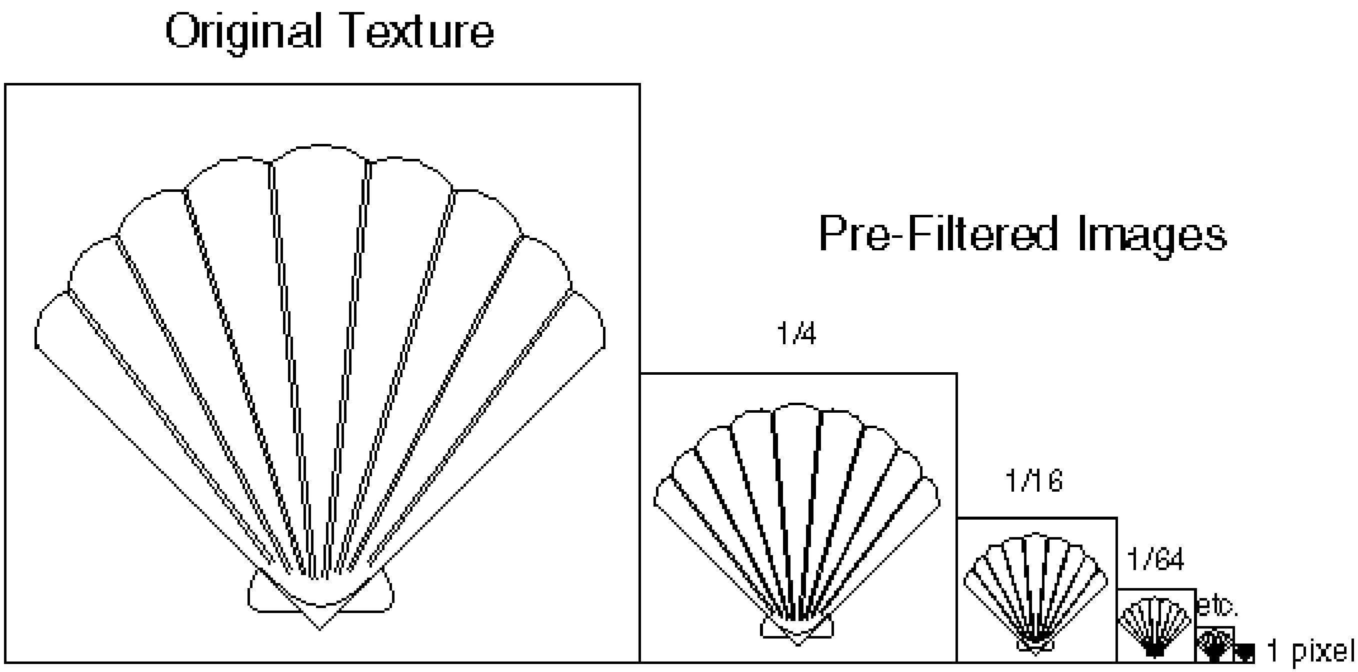
MipMapping

- Like any other object, a texture mapped object can be viewed from many distances.
- Sometimes, that causes problems.
  - A low-resolution texture (say, 32x32) applied to a big polygon (say, one that occupies a 512x512 area on screen) will appear blocky.
  - Conversely, if you apply a high-resolution texture to a small polygon, how do you decide which texels to show and which to ignore?

Mipmapping (continued)

- One solution is to provide multiple levels of detail for the same texture and use the one that best matches the polygon’s apparent size on screen.
- This technique is called Mipmapping（分级细化贴图), and each texture in the hierarchy is called a mipmap.
- OpenGL can compute mipmaps automatically, and it can also accept mipmaps provided from different files.

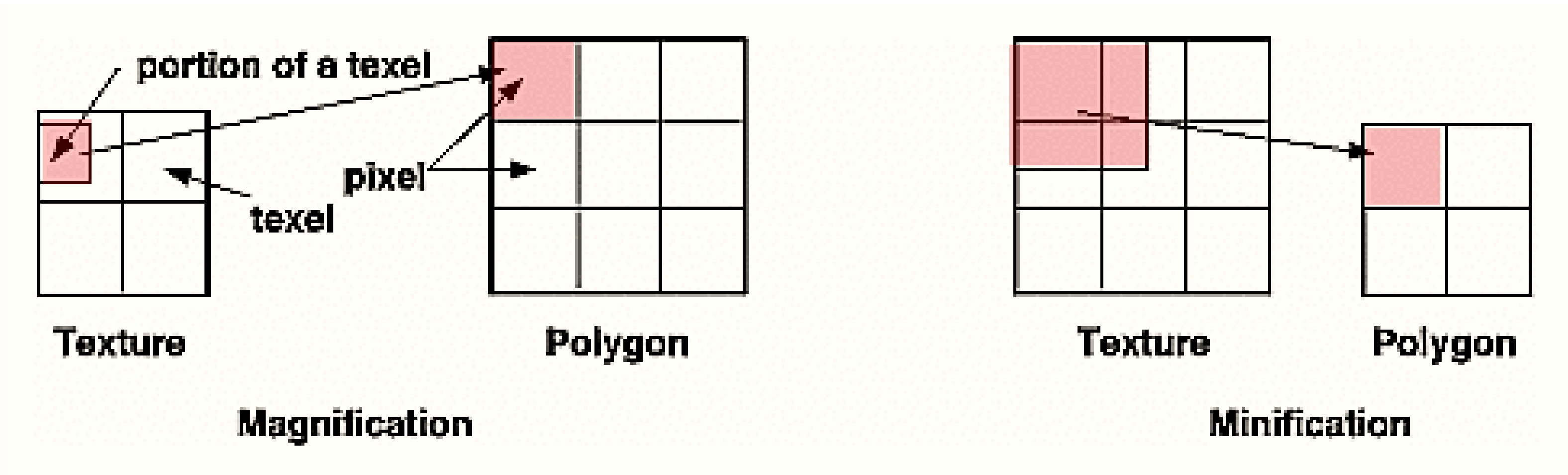
MipMapping Example



Texture Filtering

- When the texture is mapped to a polygon, a single texel rarely matches a single pixel exactly.
- If a pixel matches only a portion of a texel, the texel must be magnified.
- If a pixel matches more than one texel, the texels must be minified.

Filtering Example

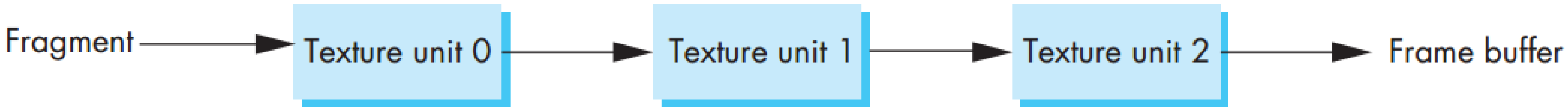


Texture Matrix

- Loading a texture onto the graphics card is very expensive
- But once there, a *texture matrix* can be used to "transform" the texture
  - For example, changing the translation can select different parts of the texture
- If the texture matrix is changed from frame to frame, the texture will appear to move on the object
- This is particularly useful for things like flame, or swirling vortices, …

Multi-texturing

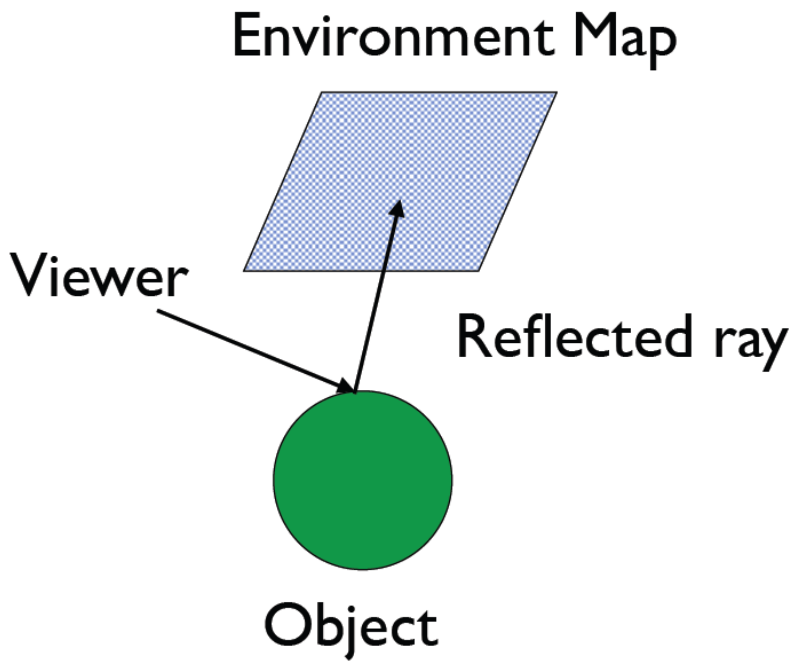
- It is sometimes possible to apply more than one texture to a polygon
- Examples: Light Maps, Texture Blending
- NOTE:
  - Most, but not all, implementations of OpenGL support multitexturing!



Other Methods of Texture Mapping

Environment Mapping

- Environment mapping produces reflections on shiny objects
- Texture is transferred in the direction of the reflected ray from the environment map onto the object
- Reflected ray:
  $R=2(N\cdot V)N-V$



Environment Mapping

- **Highly reflective surfaces are characterized by specular reflections that mirror the environment.**
  - In a distorted form
  - Requires global information

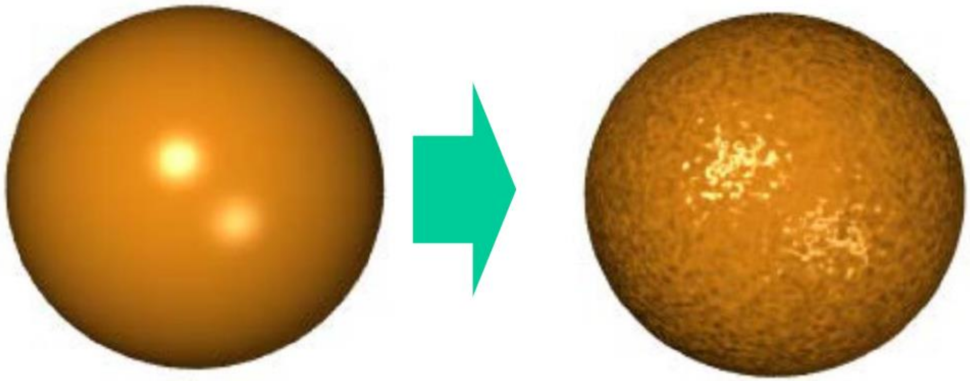
Environment Mapping

Solution

- A physically based rendering method, such as a ray tracer, can produce this kind of image.
- Ray-tracing calculations are too time-consuming to be practical for real-time applications
- Expand the texture mapping method：can give approximate results that are visually acceptable through environment maps or reflection maps.

Bump Mapping (凹凸贴图)

- **The technique of bump mapping varies the apparent shape of the surface by perturbing the normal vectors as the surface is rendered.**
  - **The colors that are generated by shading then show a variation in the surface properties**



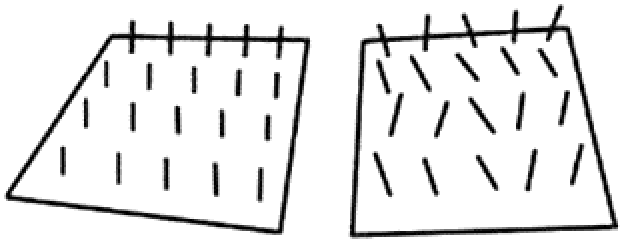
如此得到的图像就会显现出形状变化的错觉

凹凸贴图是指计算机图形学中在三维环境中通过纹理方法来产生表面凹凸不平的视觉效果。

主要的原理是通过改变表面光照方程的法线，而不是表面的几何法线来模拟凹凸不平的视觉特征，如褶皱、波浪等等。

Finding Bump Maps

- The normal at any point on a surface characterizes the orientation of the surface at that point.
- If we perturb the normal at each point on the surface by a small amount, then we create a surface with small variations in its shape.



如果在生成图像时进行这种扰动，那么就会从光滑的模型得到具有复杂表面模型的图像。

Method of perturbation (1)

- We can perturb the normals in many ways.
- The following procedure for parametric surfaces is an efficient one.
  - Letp(u, v)be a point on a parametric surface and the partial derivatives at the point:

$$\mathbf{p}_u = \begin{bmatrix} \frac{\partial x}{\partial u} \\ \frac{\partial y}{\partial u} \\ \frac{\partial z}{\partial u} \end{bmatrix}, \qquad \mathbf{p}_v = \begin{bmatrix} \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial v} \end{bmatrix}$$

  - The unit normal at that point:

$$\mathbf{n} = \frac{\mathbf{p}_u \times \mathbf{p}_v}{|\mathbf{p}_u \times \mathbf{p}_v|}$$

Method of perturbation (2)

- **Displace the surface in the normal direction by a function called the bump,or displacement function *d(u, v)*, The displaced surface:**

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

- **The normal at the perturbed pointp is given by the cross product:**

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

  where

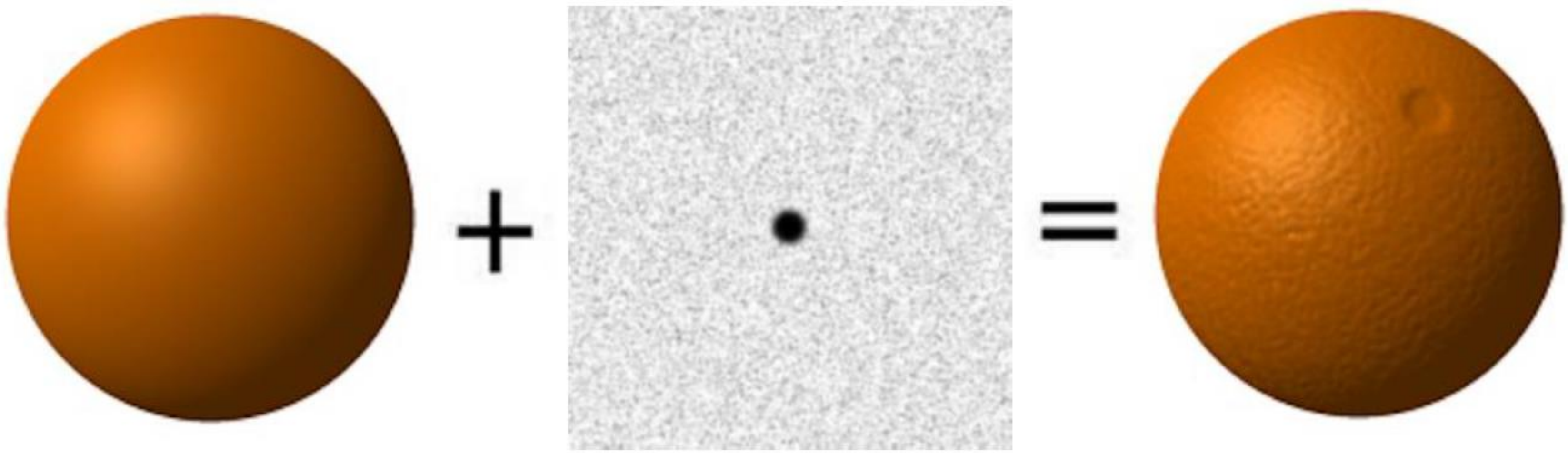
$$\mathbf{p}'_u = \mathbf{p}_u + \frac{\partial d}{\partial u}\mathbf{n} + d(u, v)\mathbf{n}_u$$

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

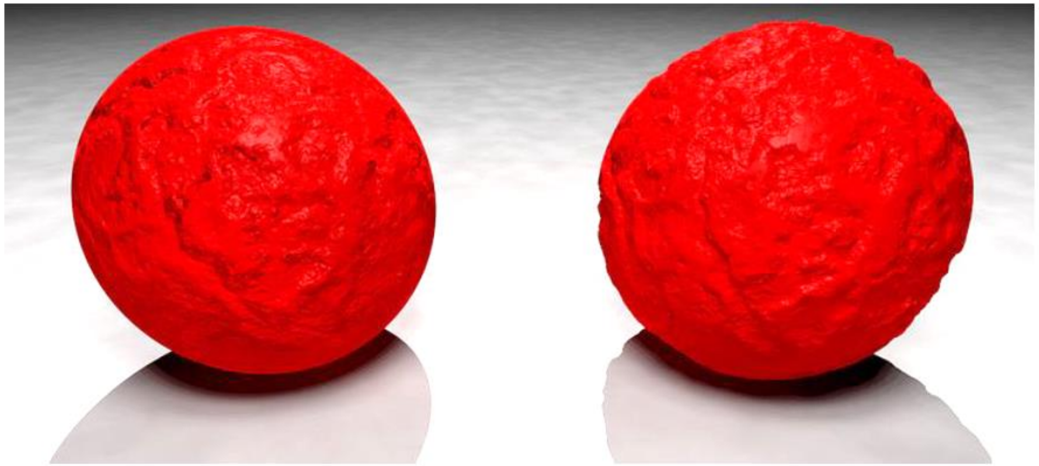
Method of perturbation(3)

- **To obtain the approximate perturbed normal:**

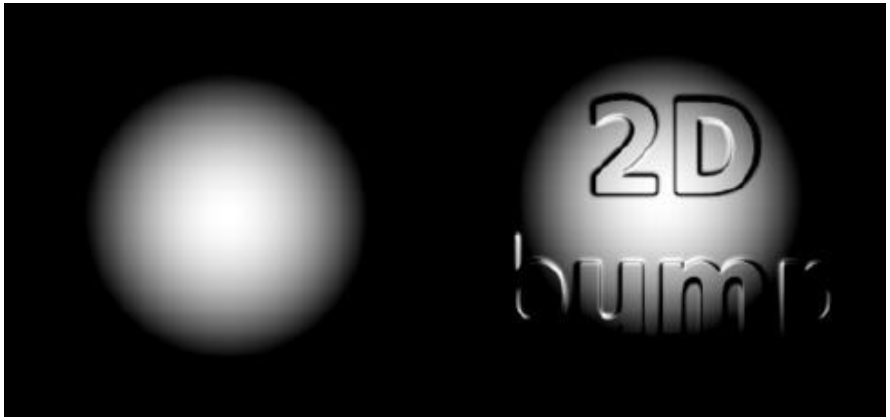
$$\mathbf{n}' \approx \mathbf{n} + \frac{\partial d}{\partial u}\mathbf{n} \times \mathbf{p}_v + \frac{\partial d}{\partial v}\mathbf{n} \times \mathbf{p}_u$$



Example



Example

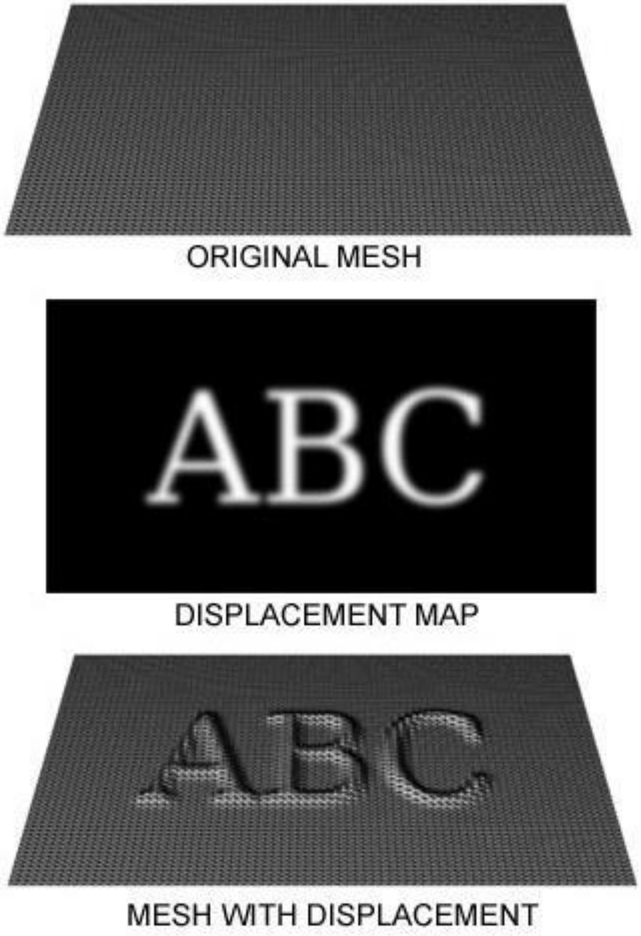


Displacement Mapping (置换贴图)

- Displacement mapping is an alternative computer graphics technique in contrast to bump mapping.
- Using a (procedural-) texture- or height map to cause an effect where **the actual** geometric position of points over the textured surface are **displaced**.
- 置换贴图这种效果通常是让点的位置沿面法线移动一个贴图中定义的距离。它使得贴图具备了表现细节和深度的能力，且可以同时允许自我遮盖，自我投影和呈现边缘轮廓。

Displacement Mapping

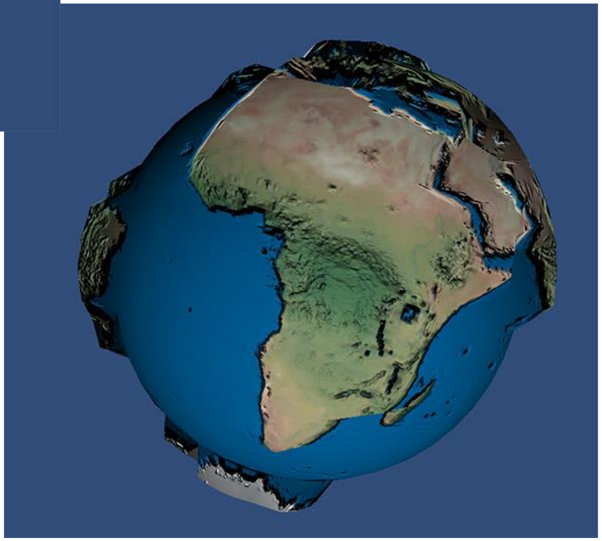
- It gives surfaces a great sense of depth and detail, permitting in particular self-occlusion, self-shadowing and silhouettes.
- On the other hand, it is the most costly of this class of techniques owing to the large amount of additional geometry.



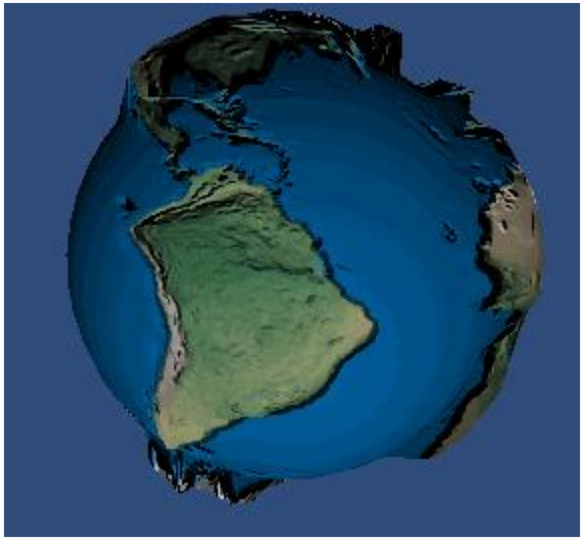
Example

Before Displacement

After Displacement

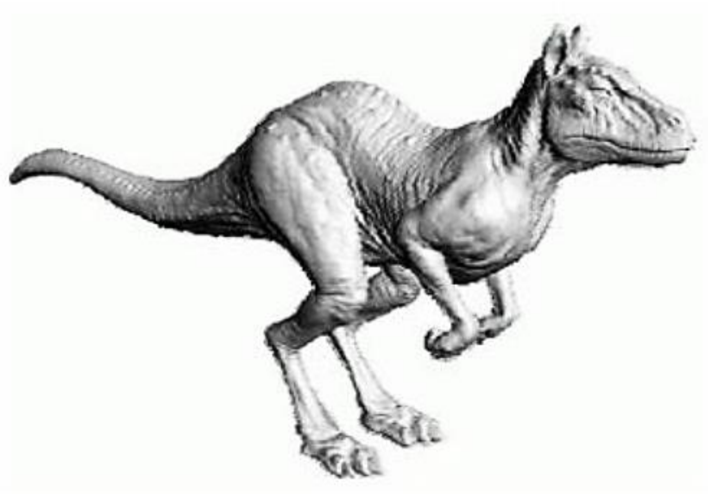


Animation

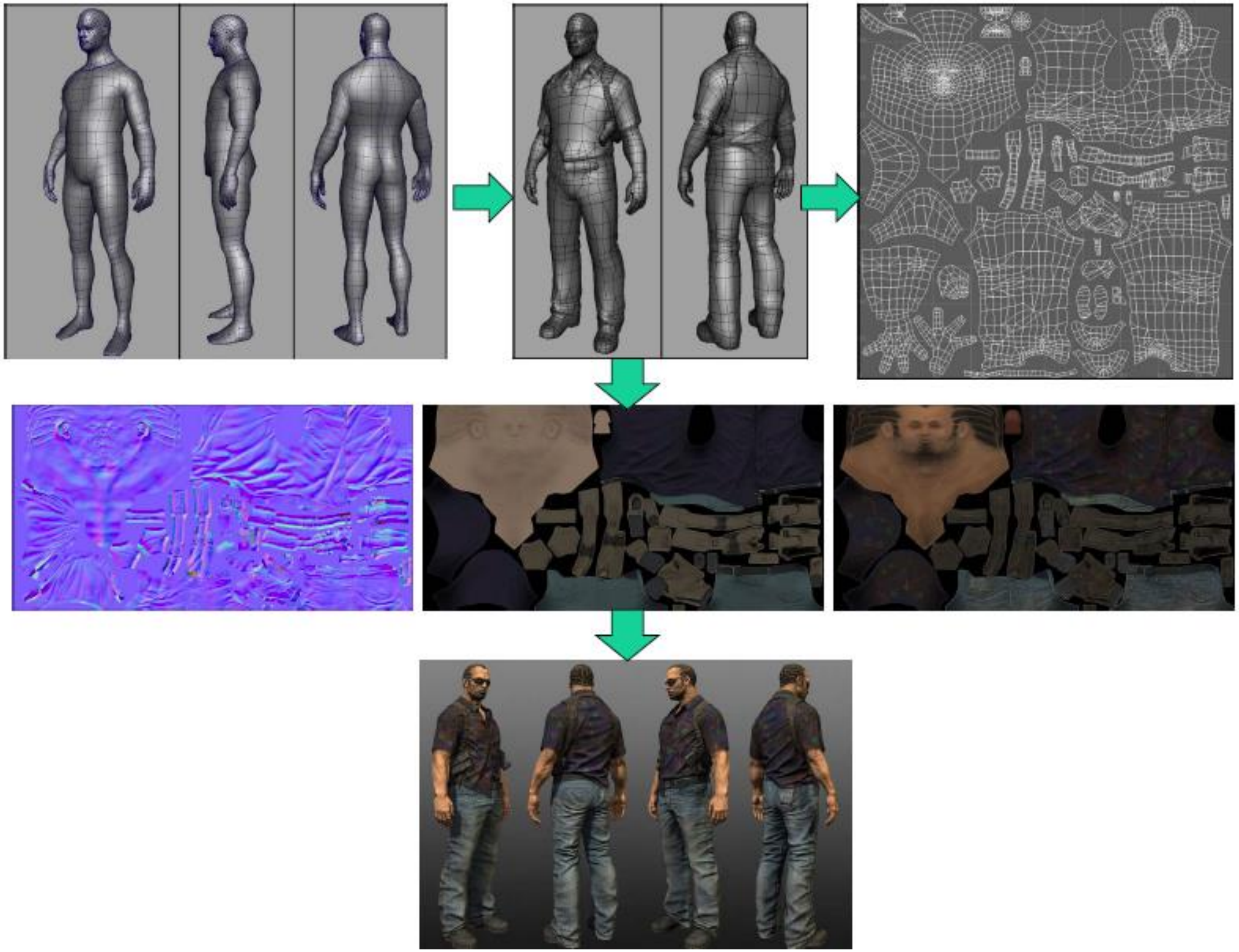


Example

Example



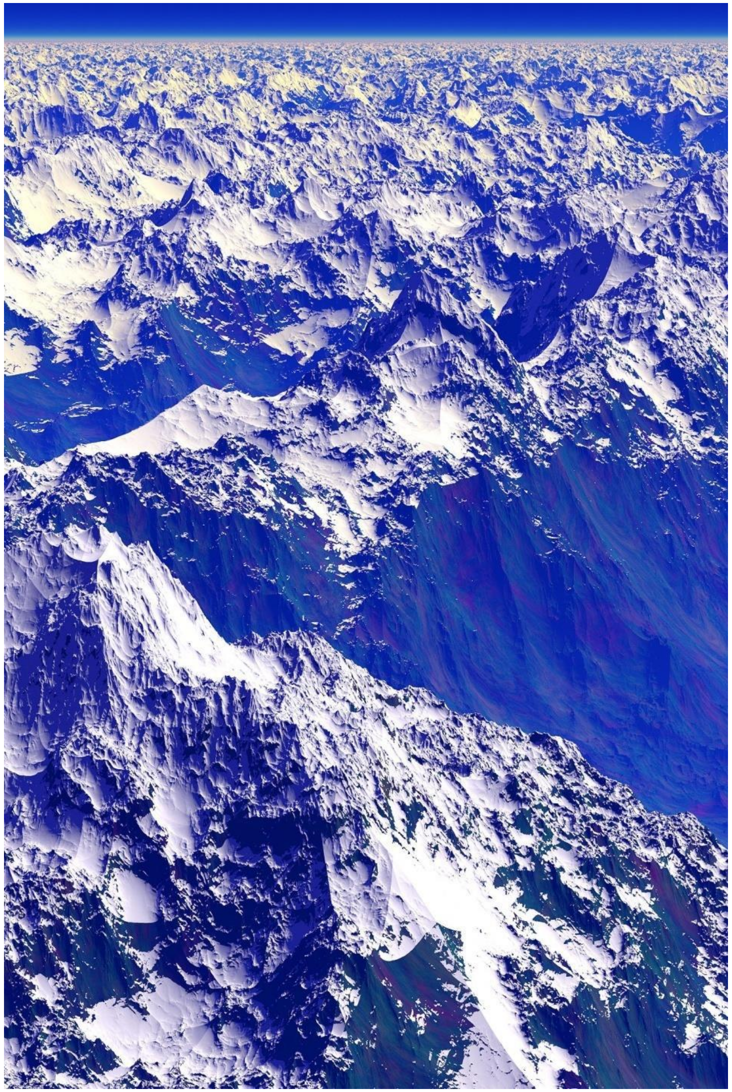
Example



Procedural Texture Mapping

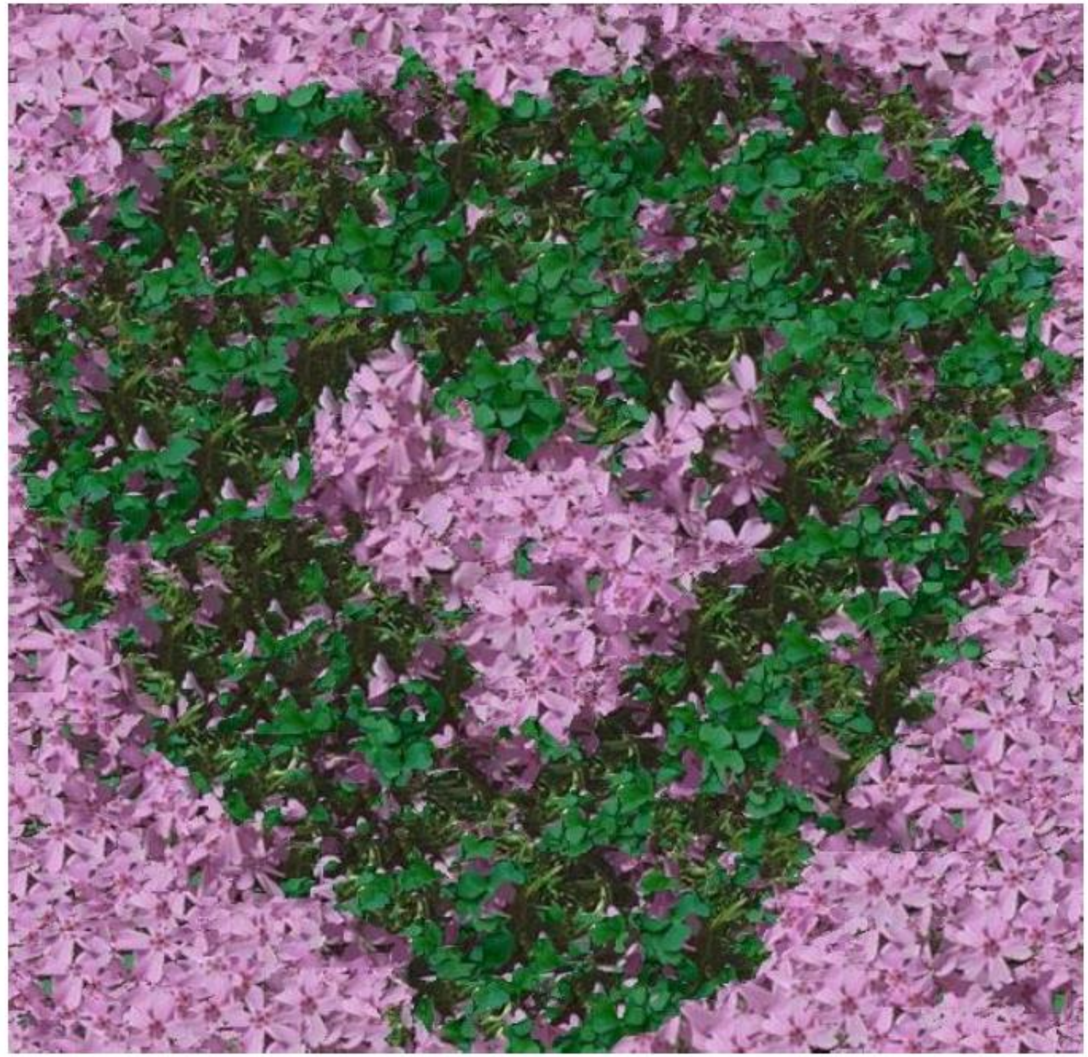
- Instead of looking up an image, pass the texture coordinates to a function that computes the texture value on the fly
  - Renderman, the Pixar rendering language, does this
  - Available in a limited form with vertex shaders on current generation hardware
- Advantages:
  - Near-infinite resolution with small storage cost
  - Idea works for many other things
- Has the disadvantage of being slow in many cases

Procedural Texture Gallery



Texture Application: Synthesis

Texture Application: Synthesis



Texture Mapping
In OpenGL

Three steps to applying a texture

1. Specify the texture
   - read or generate image
   - assign to texture
   - enable texturing
2. Assign texture coordinates to vertices
   - Proper mapping function is left to application
3. Specify texture parameters
   - wrapping, filtering

Specifying a Texture Image

- **Define a texture image from an array of *texels* (texture elements) in CPU memory**

  Glubyte my_texels[512][512];

- **Pixel maps which are applied for define a texture image**
  - Scanned image
  - Generate by application code
- **Enable texture mapping**

  glEnable(GL_TEXTURE_2D);

  - OpenGL supports 1-4 dimensional texture maps

Define Image as a Texture

void glTexImage2D(target, level, components, w, h, border, format, type, *texels);

- **target**: type of texture, e.g. GL_TEXTURE_2D
- **level**: used for mipmapping
- **components**: elements per texel 每个纹理元素的分量数
- **w, h**: width and height of texels in pixels

  texels中以像素为单位的宽度与高度
- **border**: used for smoothing
- **format and type**: describe texels
- **texels**: pointer to texel array

Typical Usage

```
glTexImage2D(GL_TEXTURE_2D, 0, 3, 512, 512, 0, GL_RGB,
        GL_UNSIGNED_BYTE, my_texels);
```

- **target**: GL_TEXTURE_2D
- **level:** no mipmapping
- **components:** R, G, B
- **w, h:** 512 X 512
- **border:** no border
- **format and type:** GL_RGB and unsigned byte (0-255)
- **texels:** actual data!

Specifying Mipmaps

- You need to specify the levels for which mipmapping is being used.
- You can also specify scale factor (between the texture image and size of polygon).
- You also need to supply images for all levels.
- You use glTexParameteri() and glTexImage2D()
  - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_BASE_LEVEL, 2);
  - glTexImage2D(..,2,…)

Automatic mipmap generation

- If you have highest resolution image, OpenGL can generate low level prefiltered images automatically.
- gluBuild2DMipmaps()
- gluBuild2DMipmapsLevels()
  - To build only a subset
- See book for detailed description of parameters

OpenGL filtering

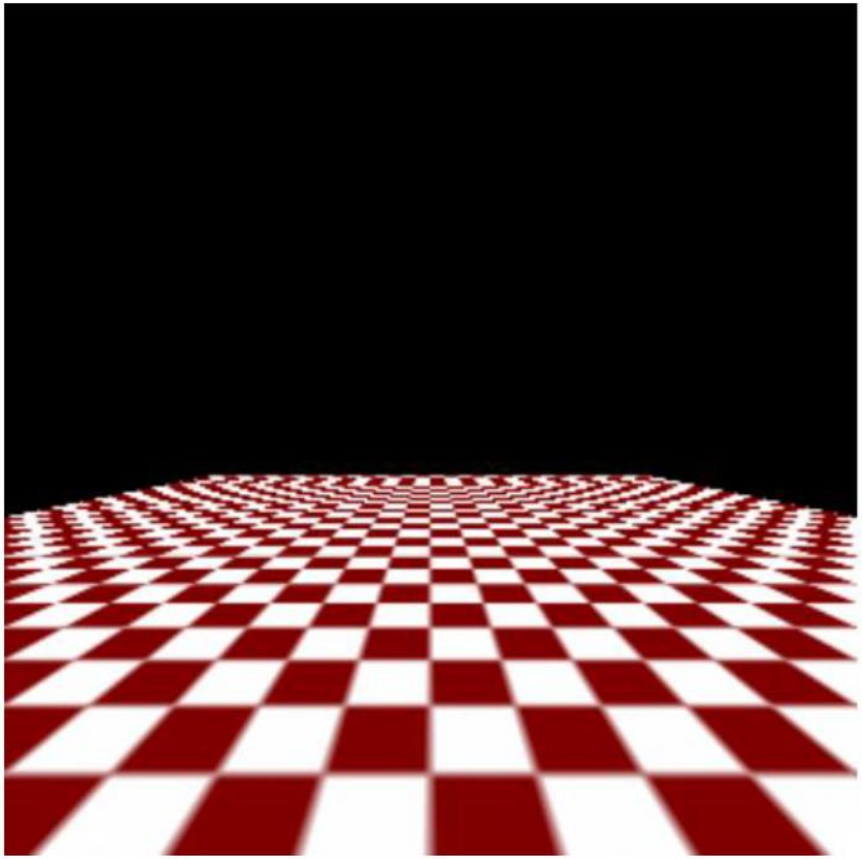
- We can specify the kind of filter to be applied
- OpenGL filtering is crude but fast
- Specify filter using glParameteri() with
  - GL_TEXTURE_MAG_FILTER and GL_TEXTURE_MIN_FILTER
- filters :
  - GL_NEAREST, GL_LINEAR, GL_NEAREST_MIPMAP_LINEAR (for minification)

  - glParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);

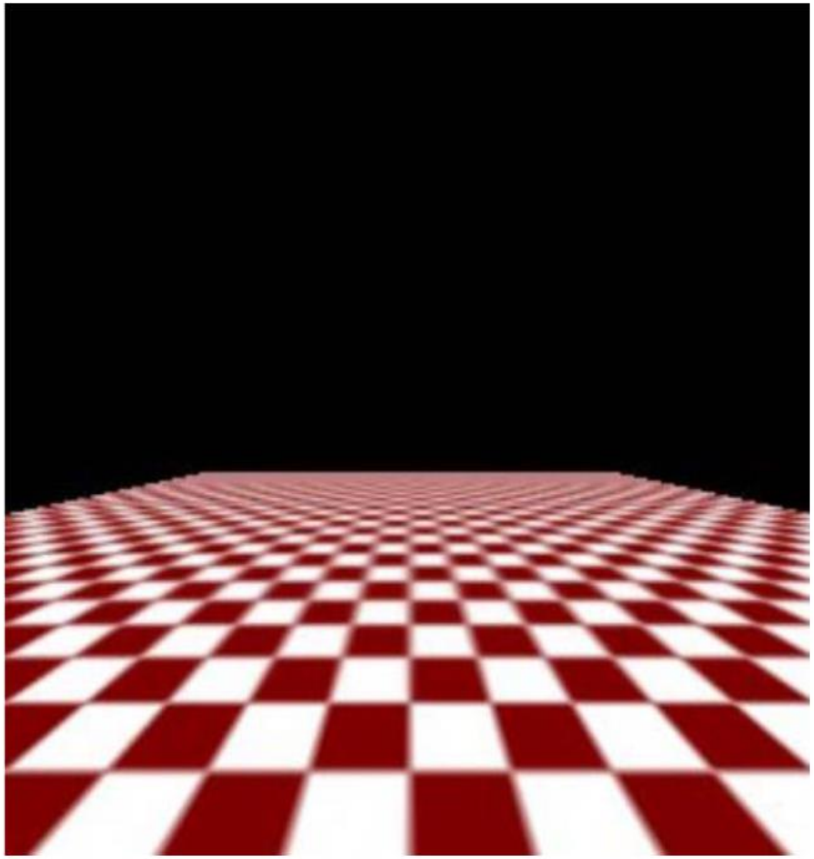
Modulating Surface Properties

- Instead of replacing color, you can also modulate color or some other property of polygon.
- glTexEnv(GL_TEXTURE_ENV, pname, param);
- Pname and param specfiy how texture affects surface
- E.g.
  glTexEnv(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_MODULATE);

Example

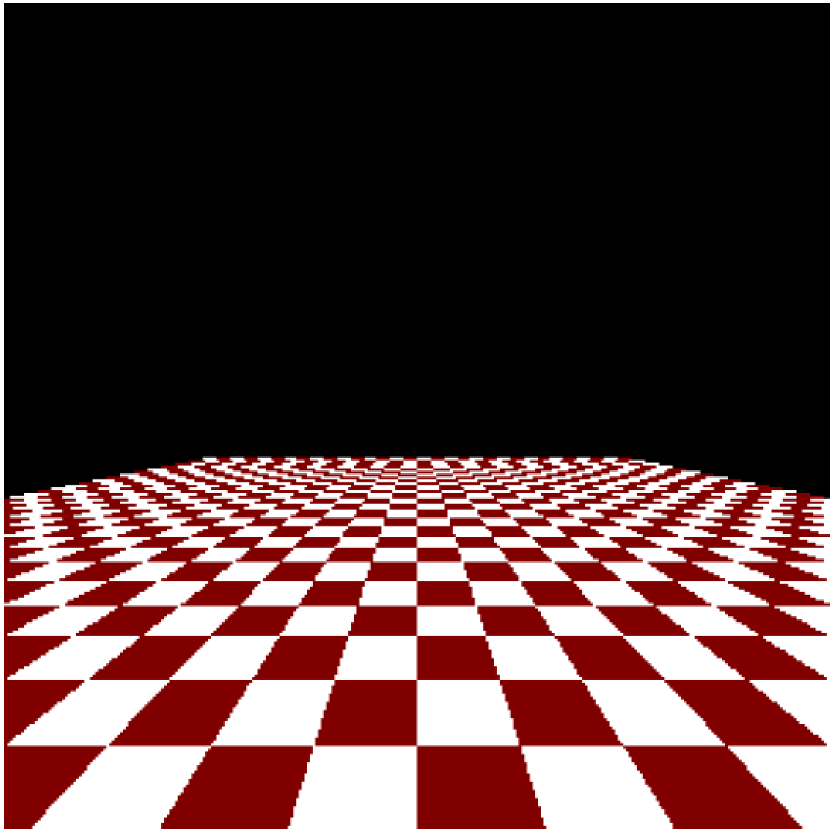


without mipmap

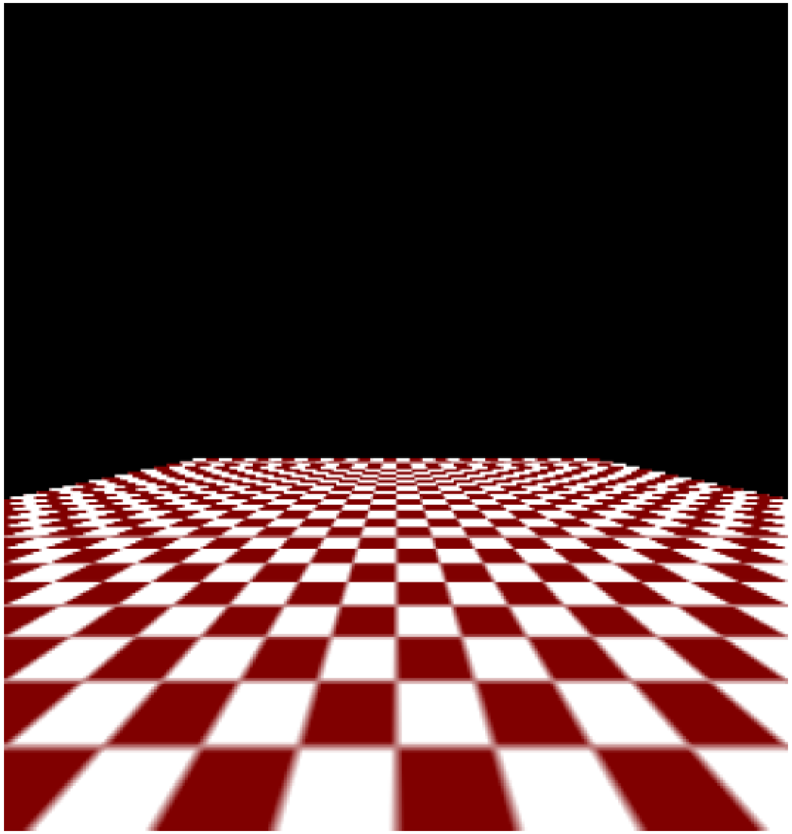


with mipmap

Example



GL_NEAREST



GL__LINEAR

Tricks with Textures

- If the texture is not large enough to cover a polygon, you can tile it using the GL_REPEAT parameter
  - glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);

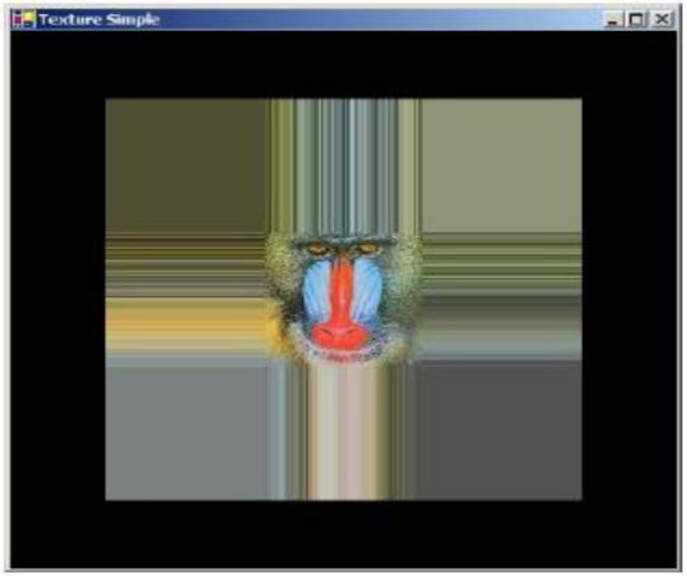
- You can also clamp the texture (i.e., stretch the last pixel to cover everything)

Wrapping Mode

- You can control what happens if a point maps to a texture coordinate outside of the texture image
  - All textures are assumed to go from (0,0) to (1,1) in texture space
- Repeat: Assume the texture is tiled
  - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT)
- Clamp: Clamp to Edge: the texture coordinates are truncated to valid values, and then used
  - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP)
- Can specify a special border color:
  - glTexParameterfv(GL_TEXTURE_2D, GL_TEXTURE_BORDER_COLOR, R,G,B,A)

Example

GL_REPEAT



GL_CLAMP

GL_REPEAT: Repeat texture in all directions

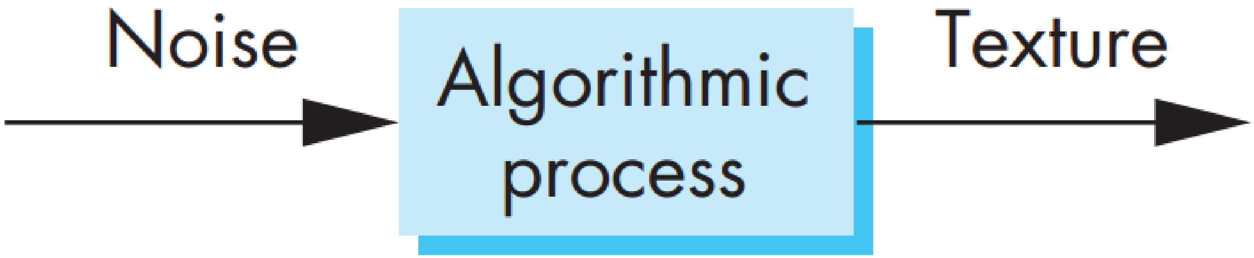
GL_CLAMP: Clamp the texture to the last value

Texture Generation

- **One of the most powerful uses of texture mapping is to provide detail without generating numerous geometric objects.**
- **High-end graphics systems can do three dimensional texture mapping in real time.**
  - 对于每一帧图像，纹理被映射到对象上，几乎与不进行纹理映射的对象显示频率相同
- **Graphics boards for personal computers now contain a significant amount of texture memory and allow game developers to use texture mapping to create complex animated environments.**

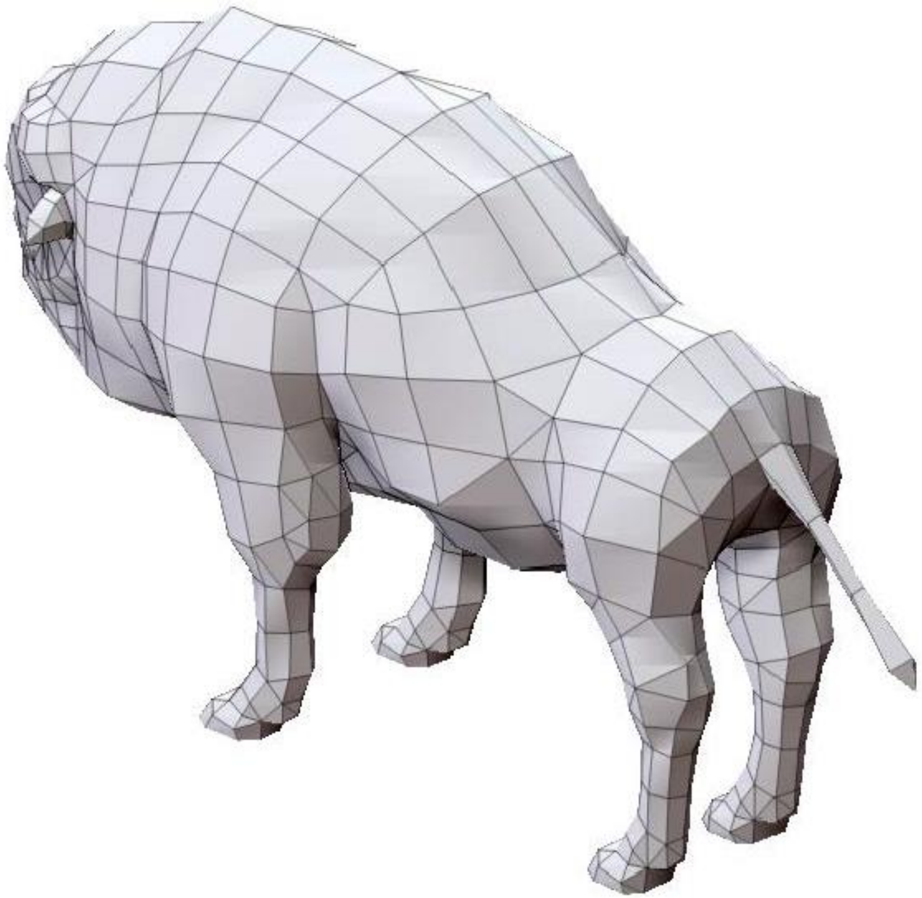
Texture Generation

- Some textures show both structure (regular patterns) and considerable randomness.
  - Sand
  - Grass
  - Minerals
- Most approaches to generating such textures algorithmically start with a random-number generator and process its output.



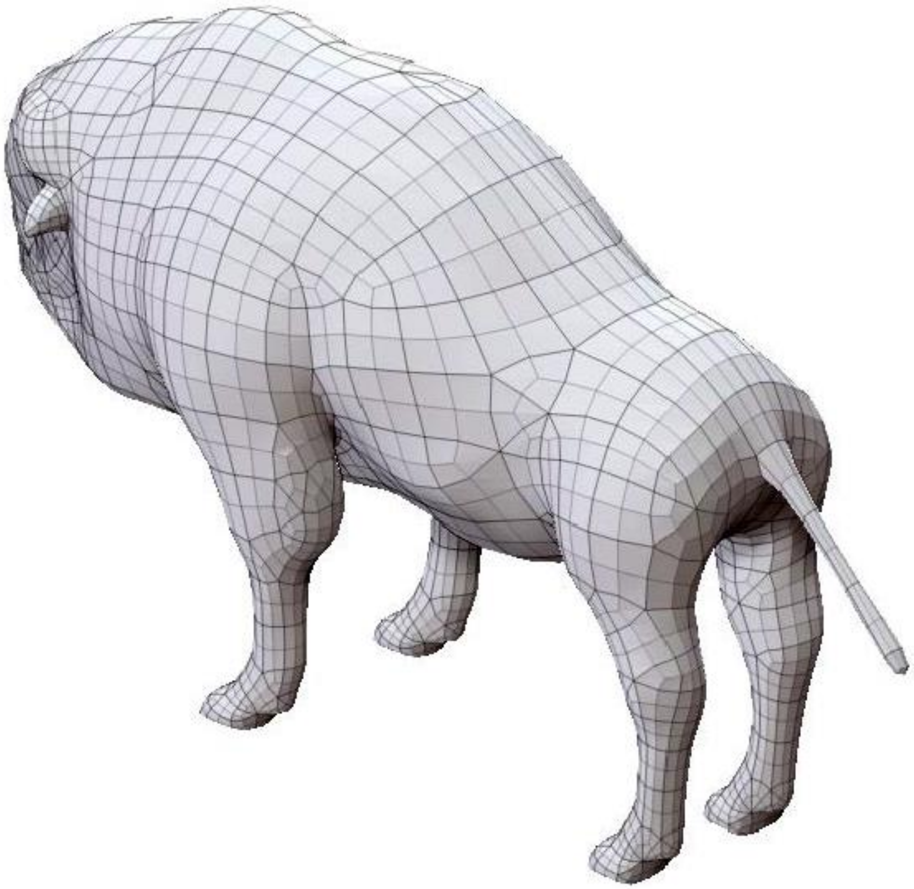
Coarse-to-Fine Sculpting Modeling

- Base mesh
- Subdivision
- Sculpting
- Subdivision
- Sculpting
- Iteration



Coarse-to-Fine Sculpting Modeling

- Base mesh
- Subdivision
- Sculpting
- Subdivision
- Sculpting
- Iteration

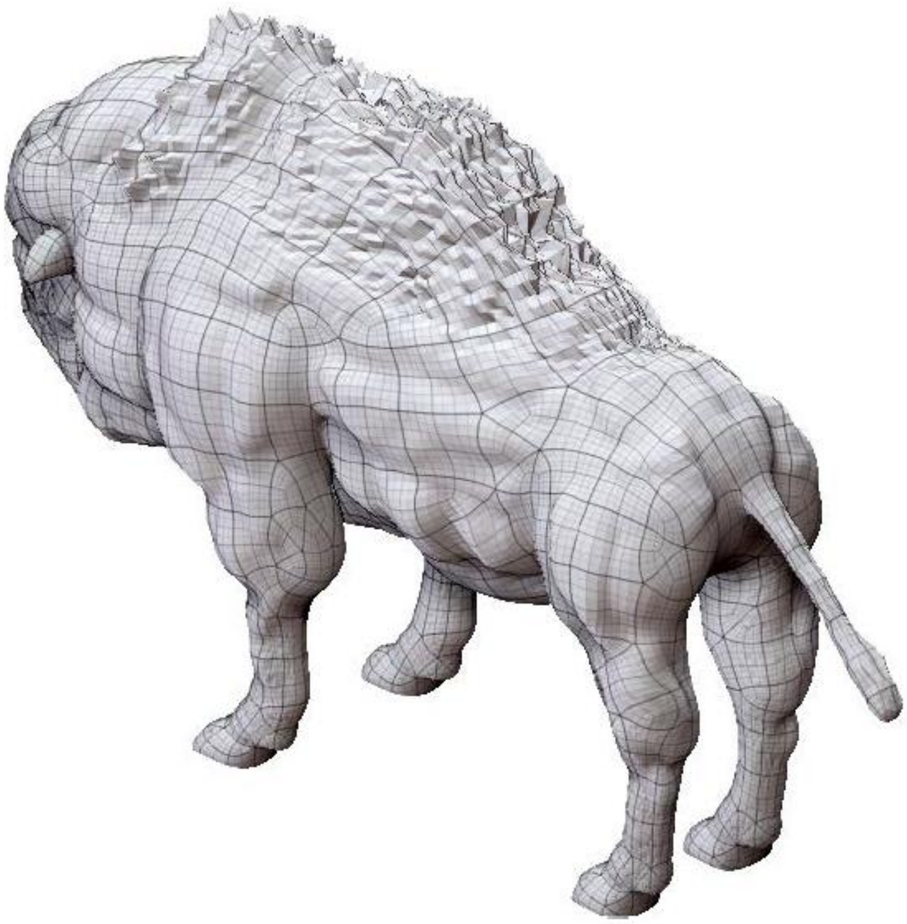


Coarse-to-Fine Sculpting Modeling

- Base mesh
- Subdivision
- Sculpting
- Subdivision
- Sculpting
- Iteration

Coarse-to-Fine Sculpting Modeling

- Base mesh
- Subdivision
- Sculpting
- Subdivision
- Sculpting
- Iteration

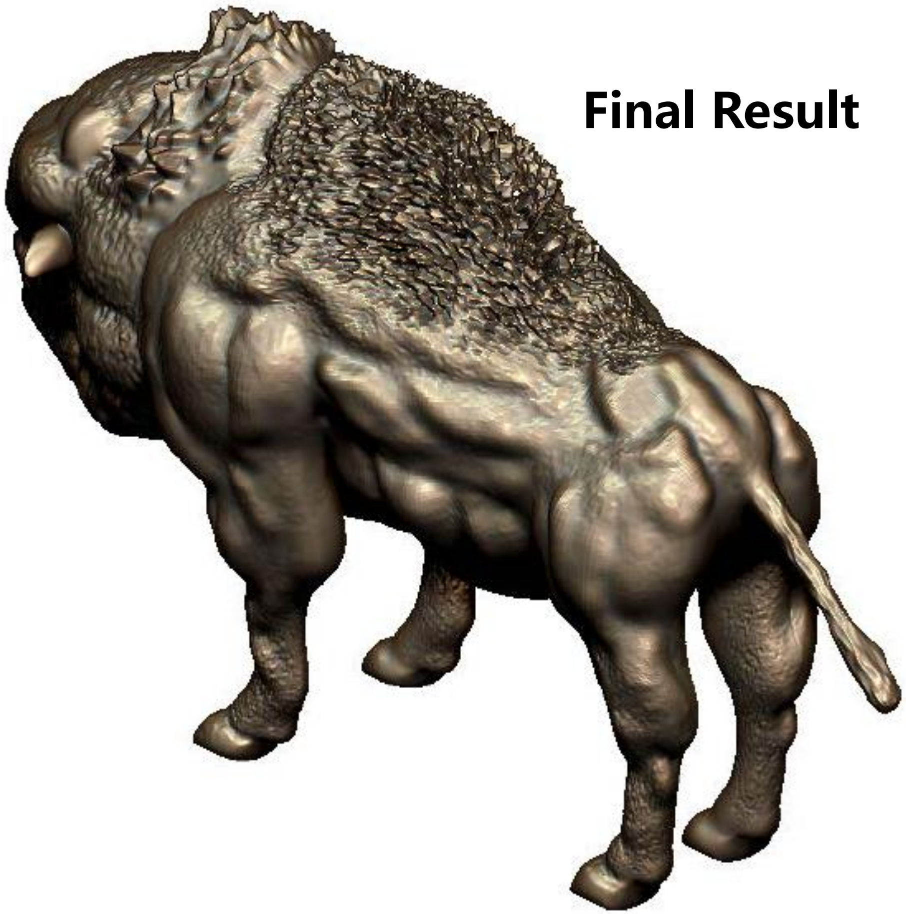


Coarse-to-Fine Sculpting Modeling

- Base mesh
- Subdivision
- Sculpting
- Subdivision
- Sculpting
- Iteration

Coarse-to-Fine Sculpting Modeling

- Base mesh
- Subdivision
- Sculpting
- Subdivision
- Sculpting
- Iteration



Final Result