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

Texture

- Texture mapping(texturing)is a fundamental technique in Computer Graphics, allowing us to represent surface details without modeling geometric material details.
- We will describe the basic ideas and applications of texture in this course

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

Why we need texture?

- Modeling surface details is one of the most important tasks for rendering realistic images.
  - For example, if we want to model a brick wall.
  - One option: use a huge number of polygons with appropriate surface coloring and reflectance characteristics to model the surface details

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
  - Landforms
  - Skin
  - Leaf
  - Hair
  - Fire and Water
  - ……

Limitations

- Representing all detail in an image with polygons would be cumbersome

Specific details

Structured noise

Pattern w/ randomness

Section through volume

Bumps

3

Why we need texture?

- However, in most applications, few people would care much about the details of the brick wall and would normally view the wall from a distance.
- In this situation, we probably don’t need to know all the details of the brick wall. Instead, we can simply model the wall as a big flat polygon, and paste onto the polygon a wall image so that the polygon looks like a real brick wall.

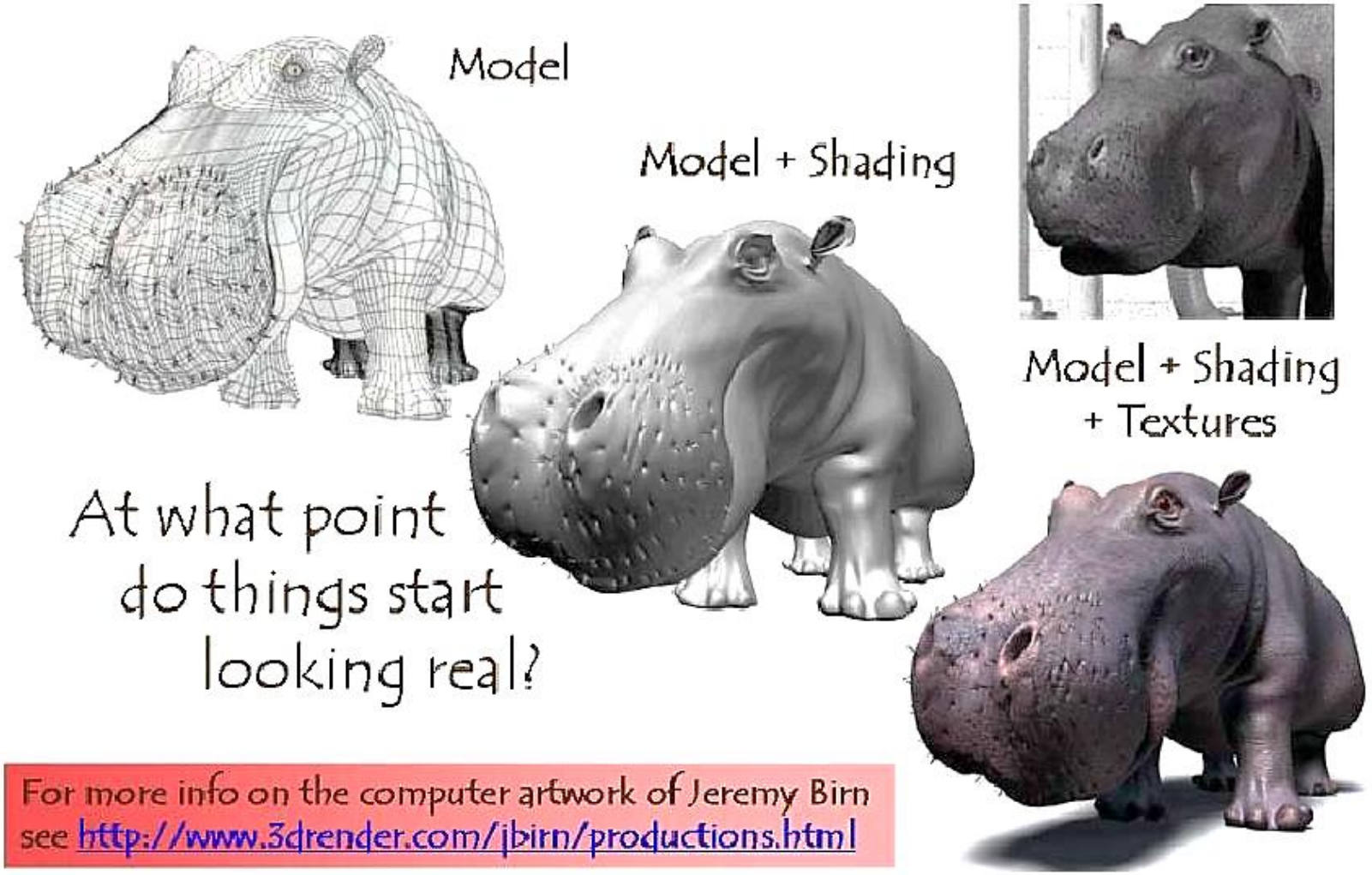
What is texture?

- This image, which is an approximation to the real brick wall, is called texture in graphics
- The process of applying the texture to an object surface is called texture mapping (纹理 映射) or texturing (贴纹理).

Why Texture is Importance ？

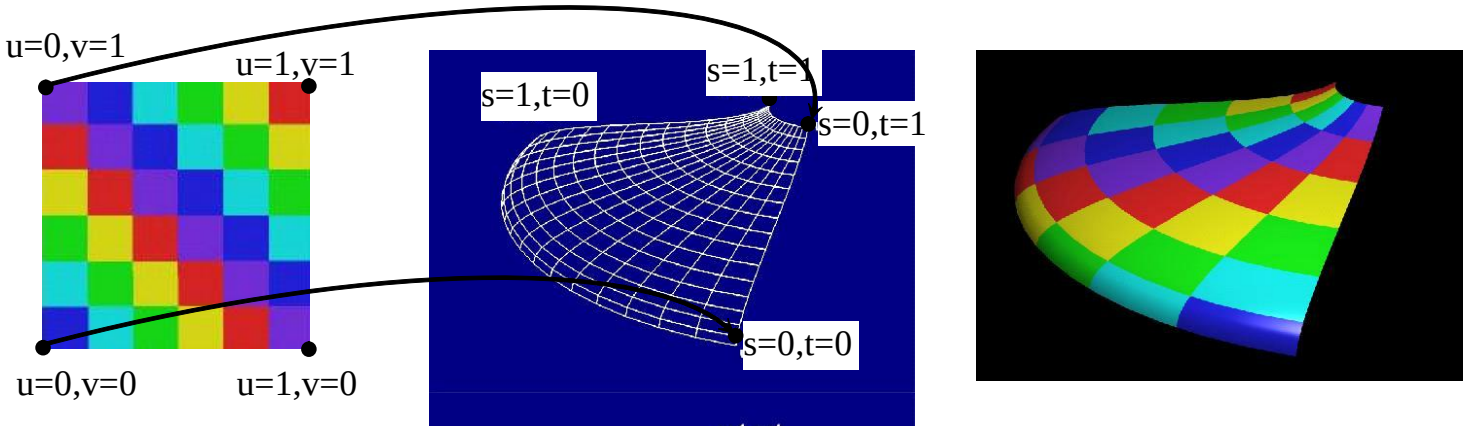
- Texturing resolves the two problems：
  - First, by representing the surface as a texture image, you don’t have to painfully model all the geometric and material details. This saves time and resources and allow users to do other more important things.
  - Second, by rendering a rough polygonal model (e.g. a single square polygon for a brick wall) and a texture instead of a detailed geometrical model with different BRDFs, the rendering can be done much more efficiently. This saves computers time and resources.

The Quest for Visual Realism



Definition of texture mapping

- Texture mapping is the process of transforming a texture on to a surface of a 3D object（adding surface detail by mapping texture patterns to the surface）.
- Developed by Catmull(1974),Blinn and Newell(1976) , and others

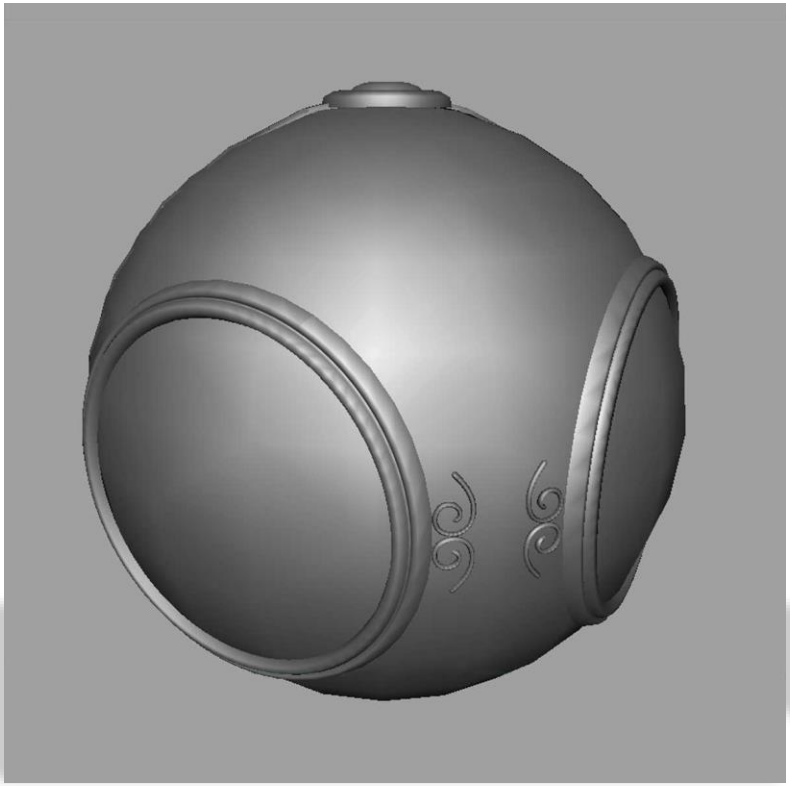


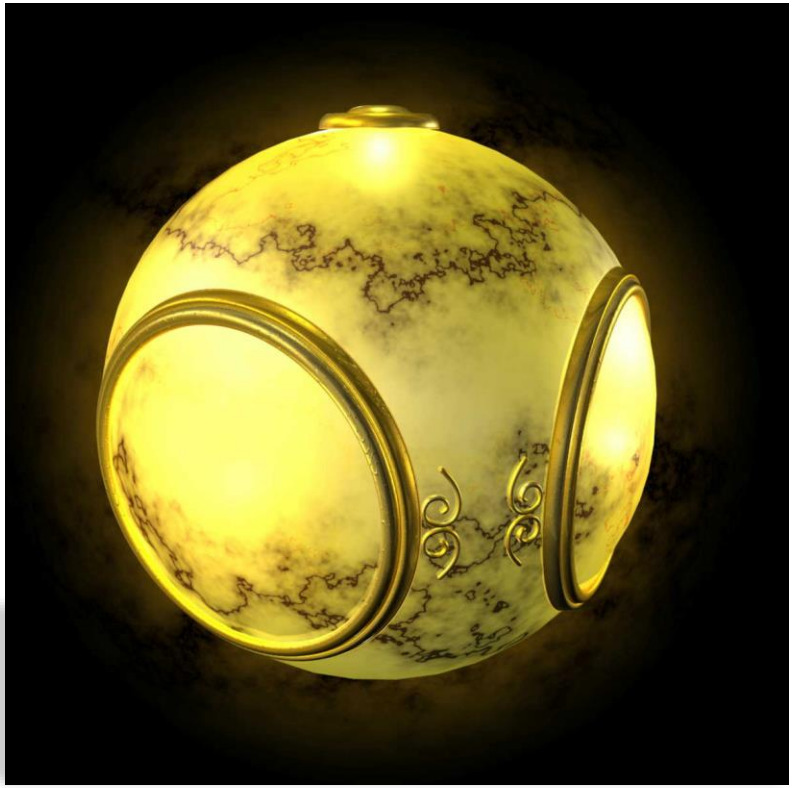
(a) 纹理空间　（b） 景物空间的曲面片　(c) 纹理映射后的曲面片

Three Types of Mapping

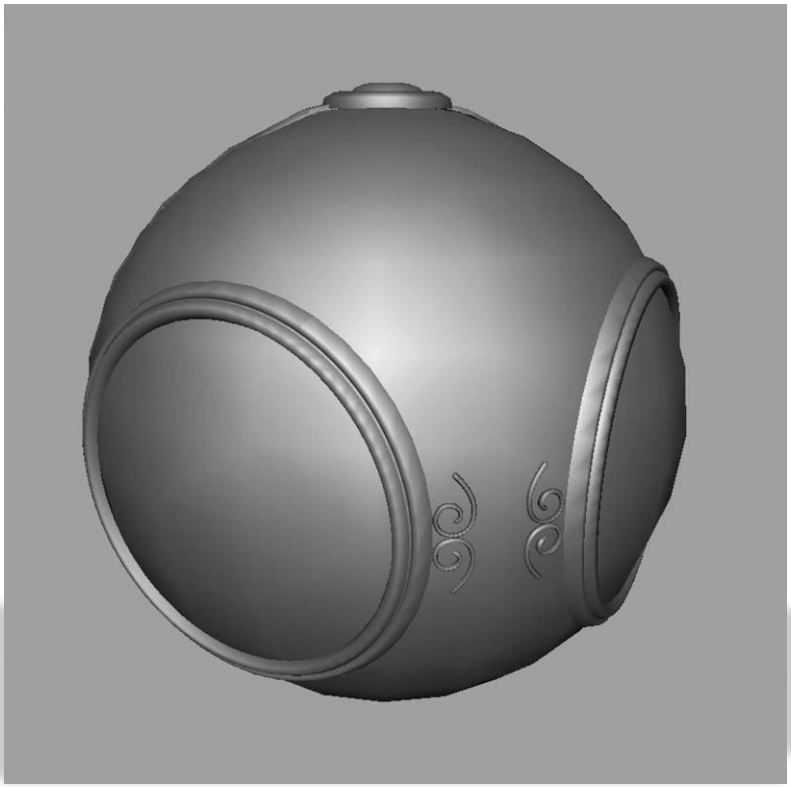
- Texture Mapping
  - Uses images to fill inside of polygons
- Bump Mapping
  - Emulates altering normal vectors during the rendering process
- Environment (reflection mapping) Mapping
  - Uses a picture of the environment for texture maps
  - Allows simulation of highly specular surfaces
  - Simulate complex mirror-like objects

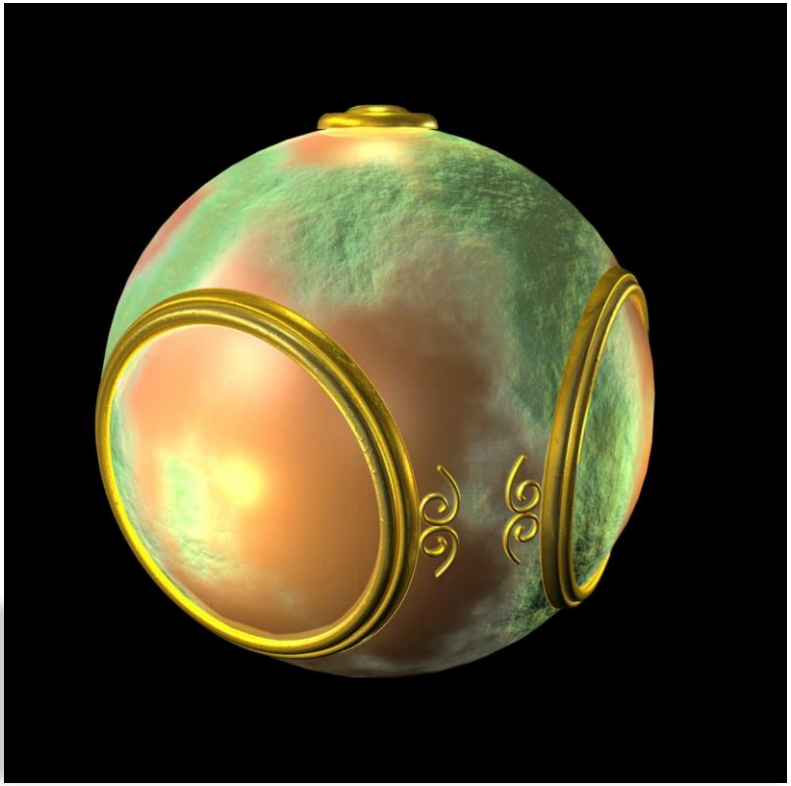
Texture Mapping



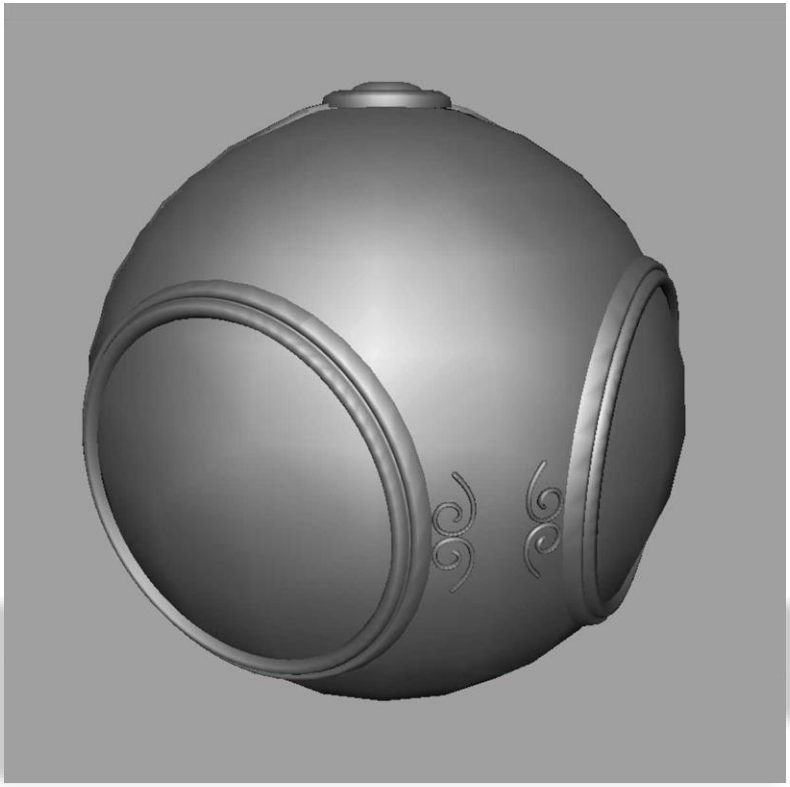


Bump Mapping





Environment Mapping



Texture usage

- To use a texture, we usually need 3 steps：
  - First，texture acquisition (获取).
  - Second，texture mapping (贴图).
  - Third，texture filtering(滤波).

Texture acquisition

- The first task of texturing is texture acquisition. There are a variety of methods to generate the texture image.
  - **Manual drawing:** You could simply draw a texture by hand.
  - **Taking photographs:** You could simply take a photo of the material which you want to model.
  - **Procedure texture**(过程纹理)
  - **Texture synthesis** (纹理合成)

Procedure texture

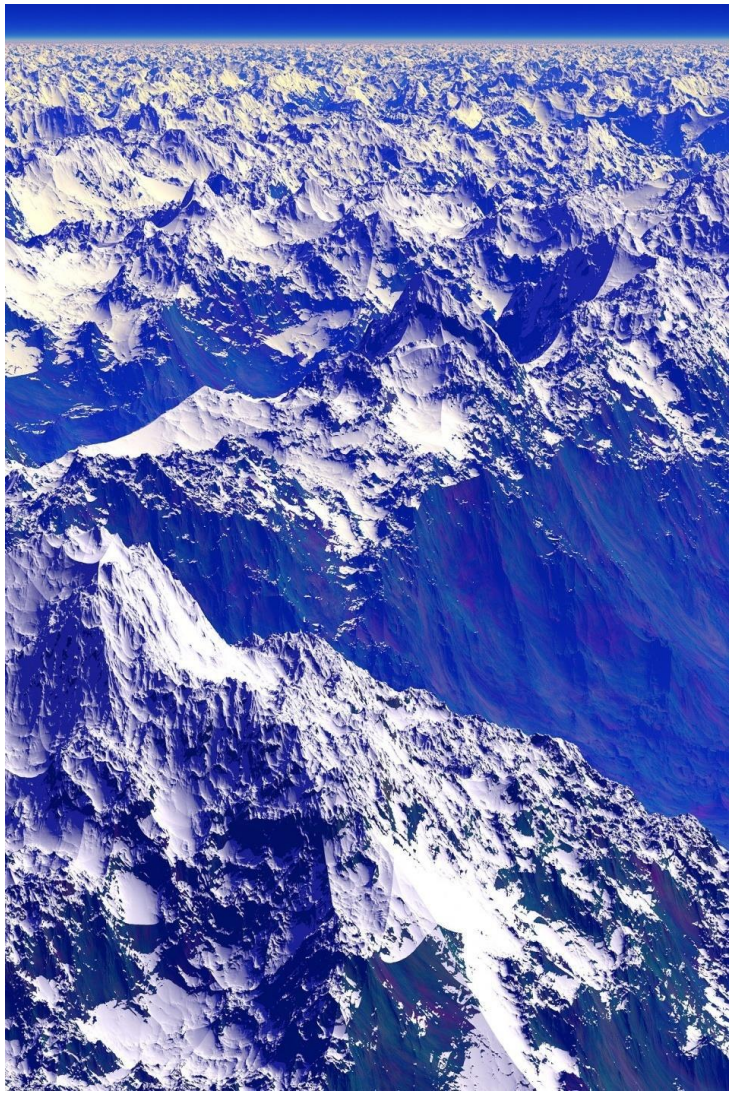
- Synthesis textures by writing special procedures which simulate either physical formation process or simply the appearance of material.
- For example, certain patterns such as marble or wood, they can be easily simulated by very simple functions.

Procedure texture

- Advantages:
  - They can be very compact, as they only need to store the procedure itself and some parameters.
  - By changing the parameters of the procedures, you can easily change the appearance of the materials, providing excellent controllability.

- Disadvantages
  - However, procedural synthesis can only be applied to some specific class of textures. – For a texture that has no known procedural code, it is not able to synthesis it.

Procedural Texture Gallery



Procedure texture

- The most popular procedure synthesis method : Perlin Noise
- From 1985, Perlin Noise has been widely adopted as the standard for procedural texture generation.
- The basic idea of Perlin noise is simple and elegant.

- **Perlin Noise**

$$perlin = \sum_{i=0}^{n-1} interpolate(white_i) \times p^i$$

  - **In the above equation, white noise at band *i* is simply a white noise with specific image size, has size 2^i. Because different bands have different sizes, we need to interpolate(插值) between them before summation.**
  - **Persistence is a user-specified parameter, which simply controls the relative weight of different frequency bands. Usually, it is within [0,1].**

Purlin Noise

- **Perlin Noise**

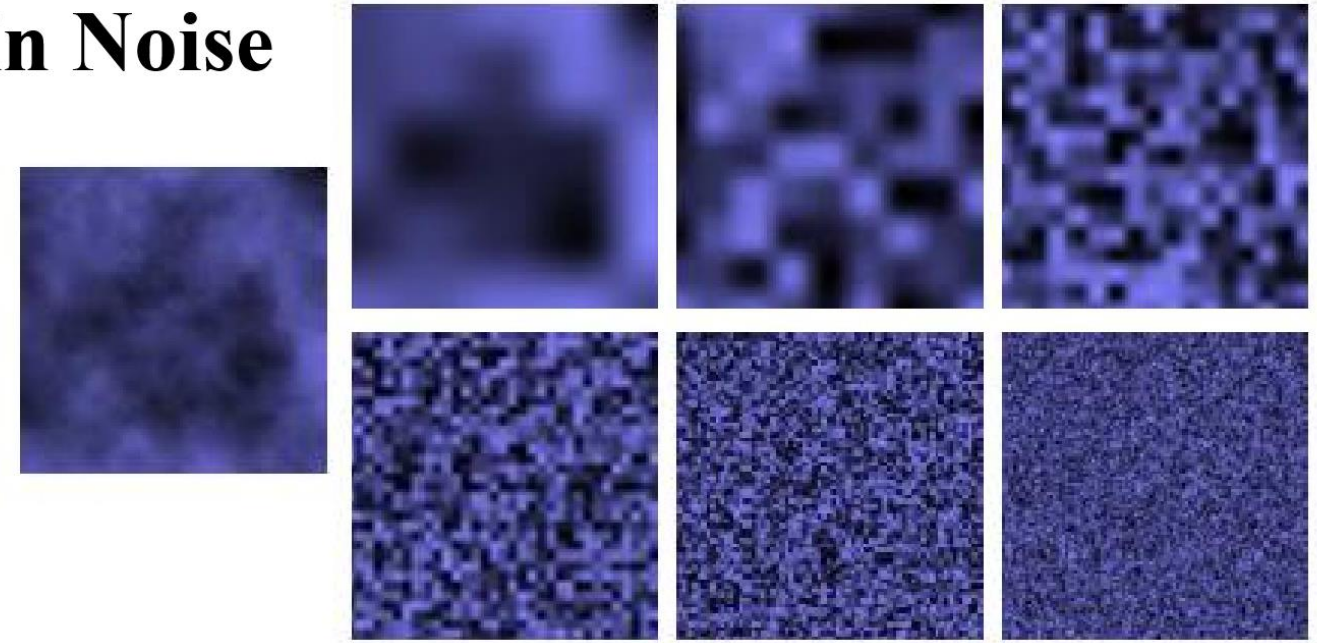


Figure 1: *2D Perlin noise example. Left: the Perlin noise image. Right: the individual noise bands, from low to high frequencies. Image courtesy of [Elias 2003].*

Purlin Noise

- **Perlin Noise**
  - **Based on Perlin noise, a variety of textures can be synthesized by proper procedures, such as marble and wood. Their formulas are as follows:**

$$
\begin{aligned}
marble &= cosine(x + perlin(x, y, z)) \\
g &= perlin(x, y, z) * scale \\
wood &= g - int(g)
\end{aligned}
$$

Texture mapping

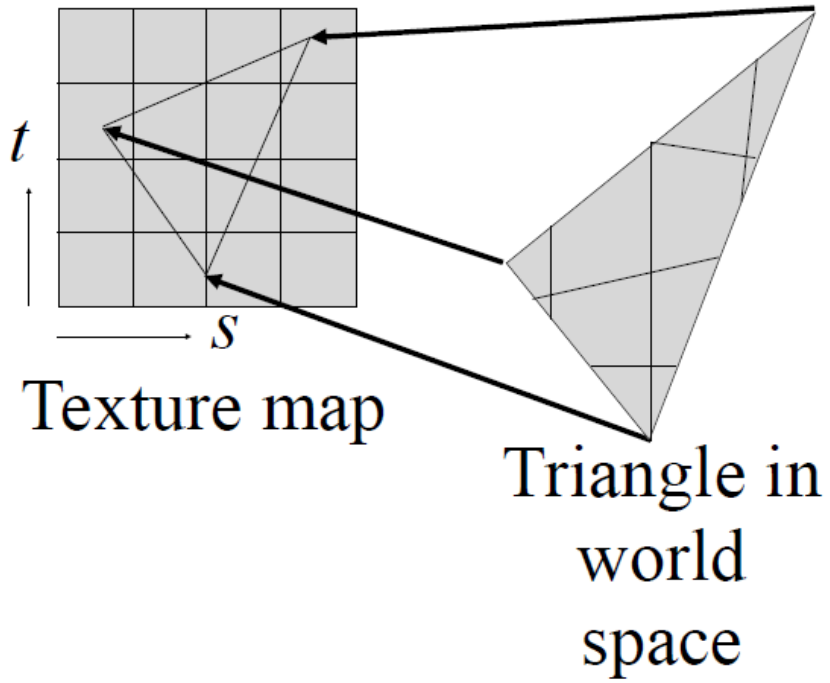
- Given a model, and a 2D texture image.
- Map the image onto the model:
  - By a function which maps a point on the model onto (u,v) image coordinates, this function is called surface mapping function
- When shading a point on the model, we look up the appropriate pixel from the 2D texture, and use that to affect the final color.

Assigning Texture Coordinates

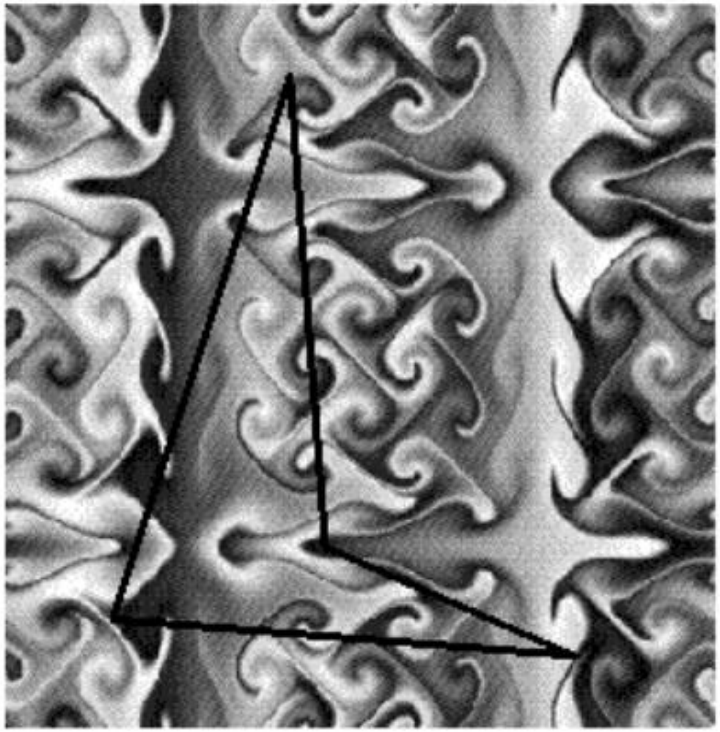
- You must provide texture coordinates for each vertex.
- The texture image itself covers a coordinate space between 0 and 1 in two dimensions usually called *u* and *v* to distinguish them from the *x*, *y* and *z* coordinates of 3D space.
- A vertex's texture coordinates determine which texel(s) are 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

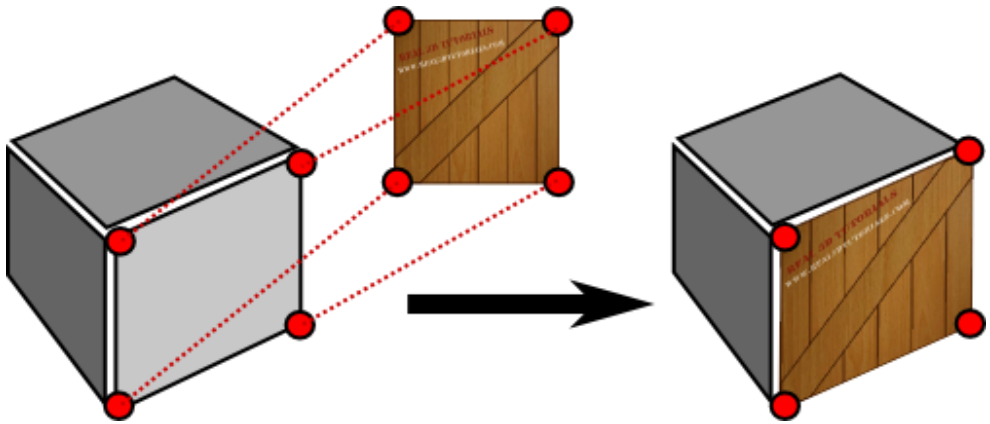


Polygonal texture mapping

- 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



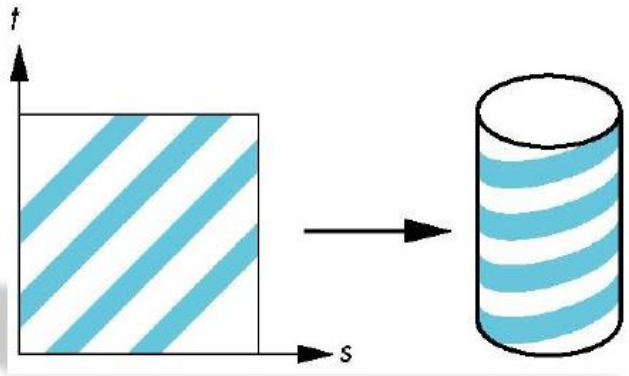
Establish Correspondences

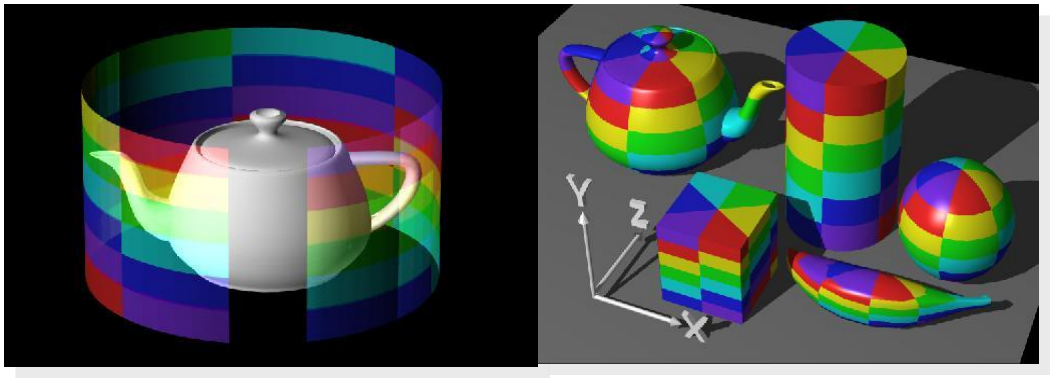
- How to specify surface mapping function?
- By natural parameterization
  - For regular objects, such as sphere, cube, cylinder
- By manually specify texture coordinates
  - For complex objects, each vertex is specified a texture coordinate

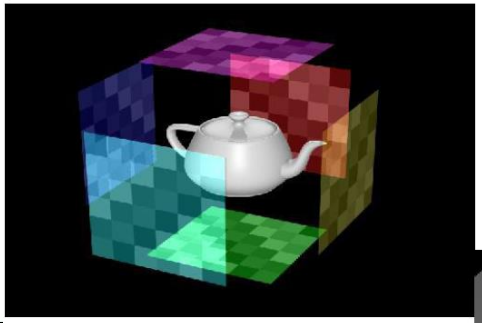
By natural parameterization

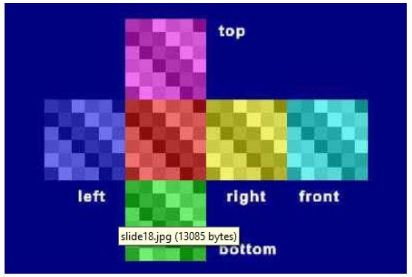
- **Natural parameterization**
  - **Sphere:**
    - **You could use spherical coordinates** $(\theta, \phi) = (\pi u, 2\pi v)$
  - **Cylinder:**
    - **You could use cylinder coordinates** $(u, \theta) = (u, 2\pi v)$
  - **Cube**
    - **Each face could directly use its face coordinates.**

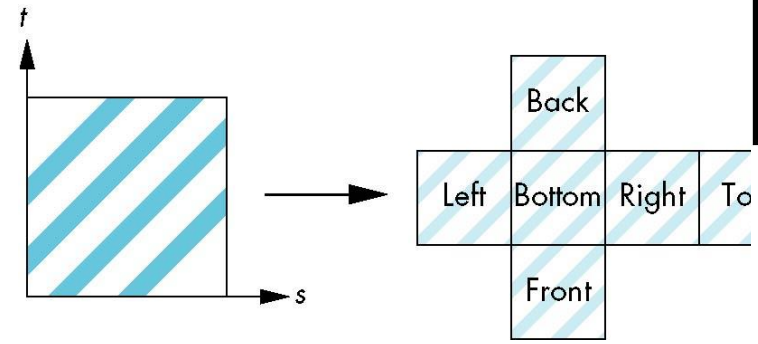
Natural parameterization

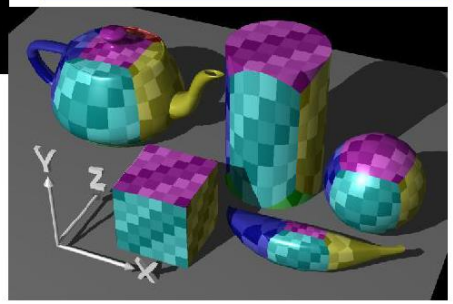






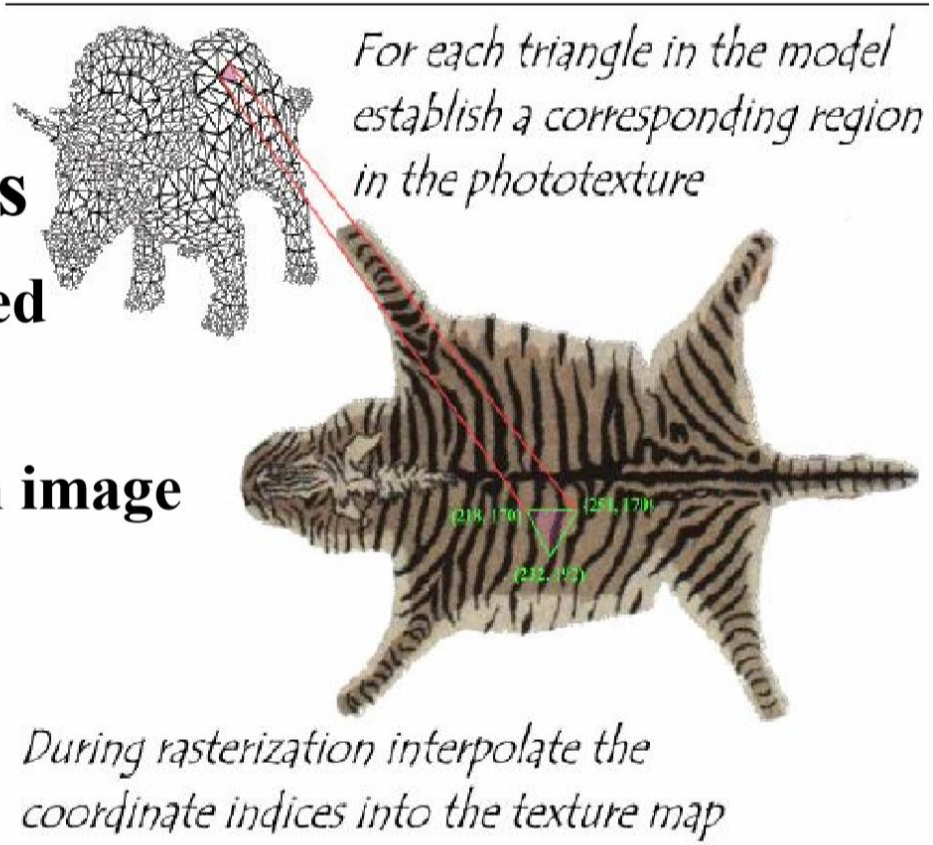






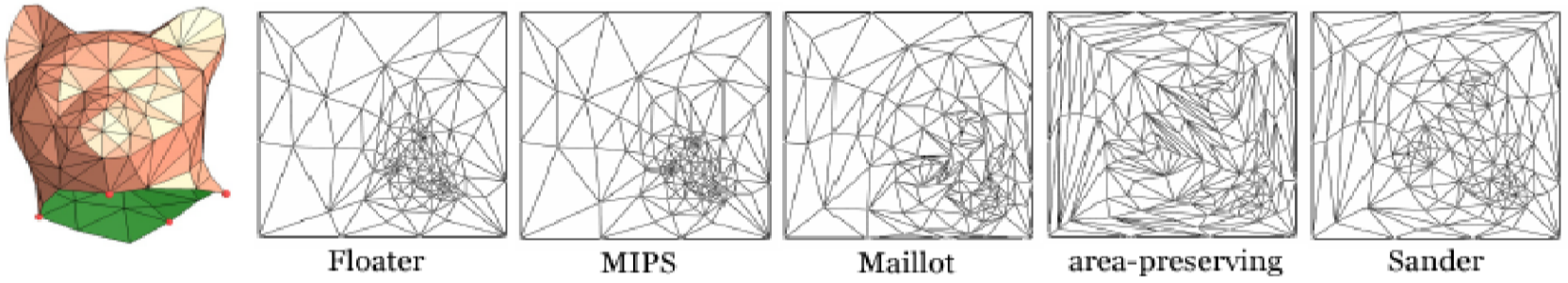
Manually Specify texture coordinates

- **Manually Specify texture coordinates**
  - **Each vertex is specified a texture coordinates**
  - **Mapping a triangle in image space to object space**

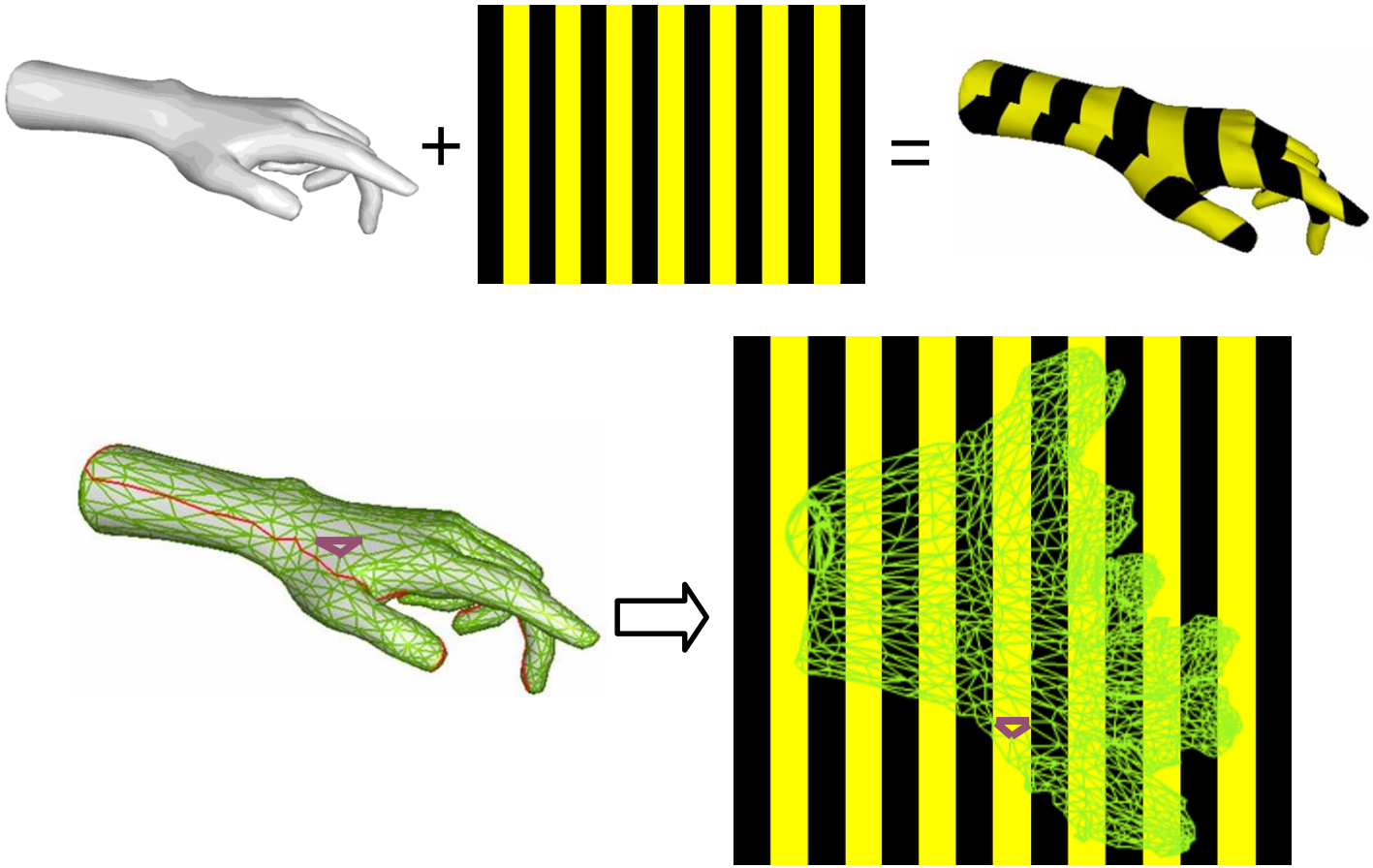


Manually Specify texture coordinates

- Mesh parametrization
  - Construct a mapping from 3D mesh models to planar domain



Manually Specify texture coordinates



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.

Crude texture mapping code

```
// for each vertex we have x,y,z and u,v
for (x=xleft; x < xright; x++) {
    if (z < zbuffer[x][y] ){
        z[x][y] = z;
        raster[x][y] = texture[u][v]; // replacing color
    }
    z=z+dz;
    u=u+dv;
    v=v+dv;
}
```

- Note that instead of replacing color as done in code, you can also modulate the color
- Instead of color, you can of course choose to modify some other property of surface

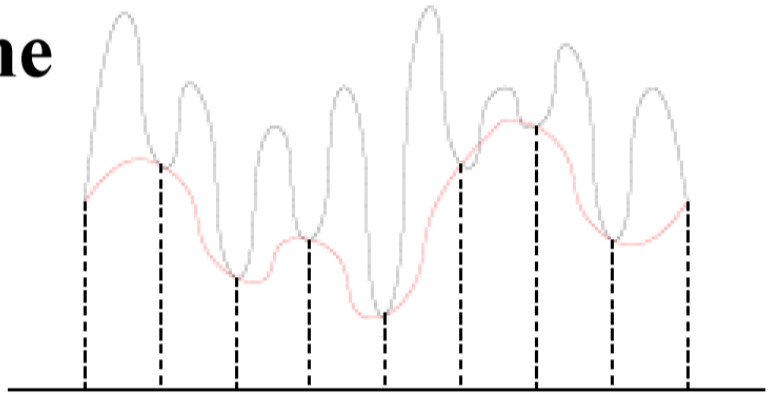
Attributes modulated for texture

- *Surface color* **(diffuse reflection coefficients)**
  - –most commonly used parameter for texture mapping. It is used like wrapping an image on to a surface, like a label on a bottle.
- *Specular and diffuse reflection* **(environment mapping)**
  - –used to capture reflection of the environment on to a surface. Commonly used in exhibiting shiny metallic surfaces.
- *Normal vector perturbations* **(bump mapping)**
  - –used to generate a rough surfaces like that of an orange.

Texture Filtering

- Now we know how to generate texture and how to map it onto an object. The next step is to render it.
- However, if you just do this directly, you would encounter some artifacts. These artifacts are caused by signal aliasing(走样)

  - **e.g.: The black dash is the original signal, the red curve is the aliased signal**



Texture Filtering

- To resolve this problem, a common method is to sample at a higher rate. However, this is not always feasible. For example, in graphics applications, the screen resolution limits sample rate.
- An alternative method would be to prefilter the signal into a lower frequency one.

Optimization

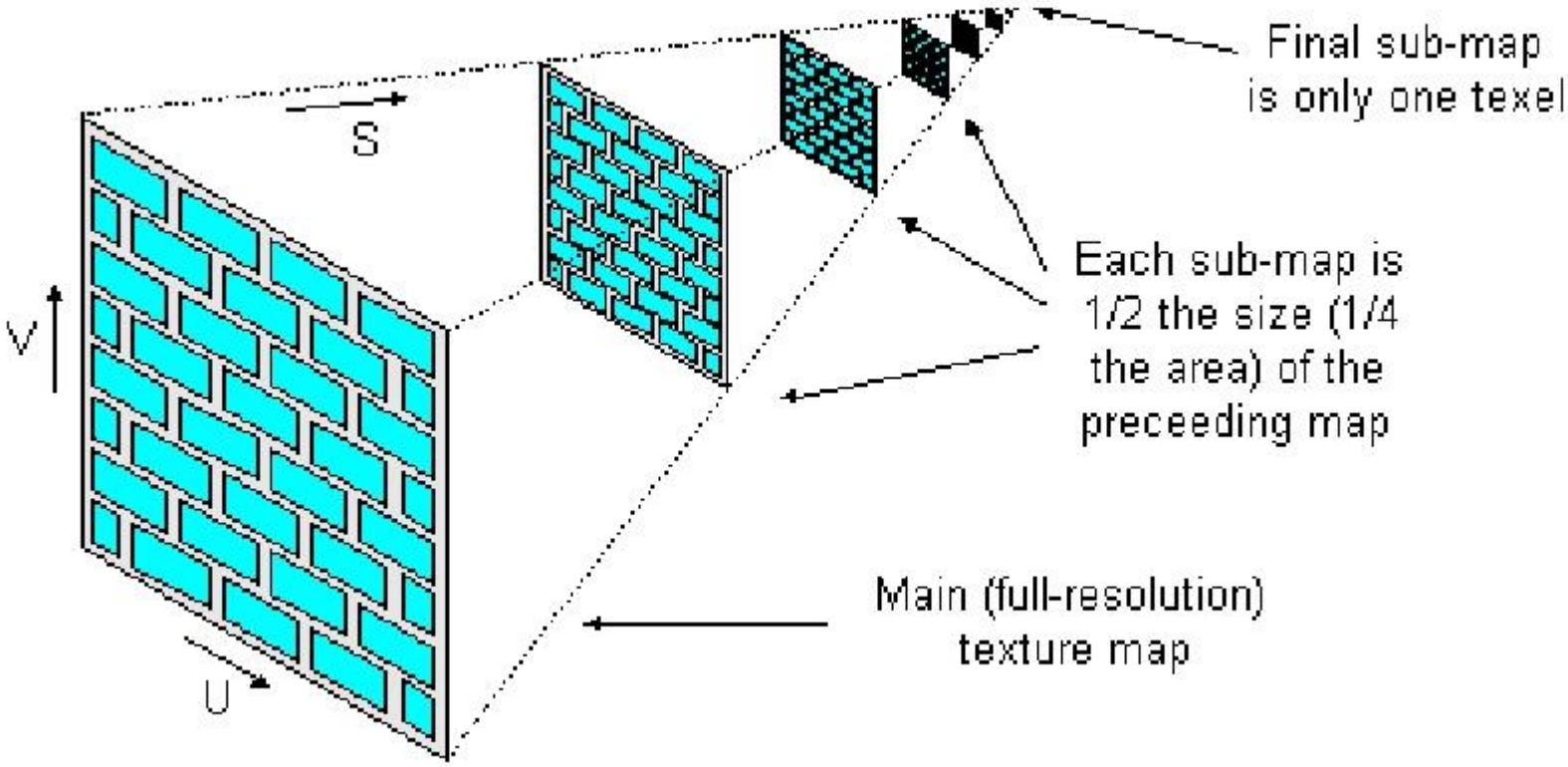
- Since most of the times, textures are known a priori, we can create various levels of these prefiltered textures in a preprocess.
  - Construct a pyramid of images that are prefiltered and re-sampled at 1/2, 1/4, 1/8, etc., of the original image's sampling
- During rasterization, we compute the index of the image that is sampled at a rate closest to the density of our desired sampling rate.
- 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?

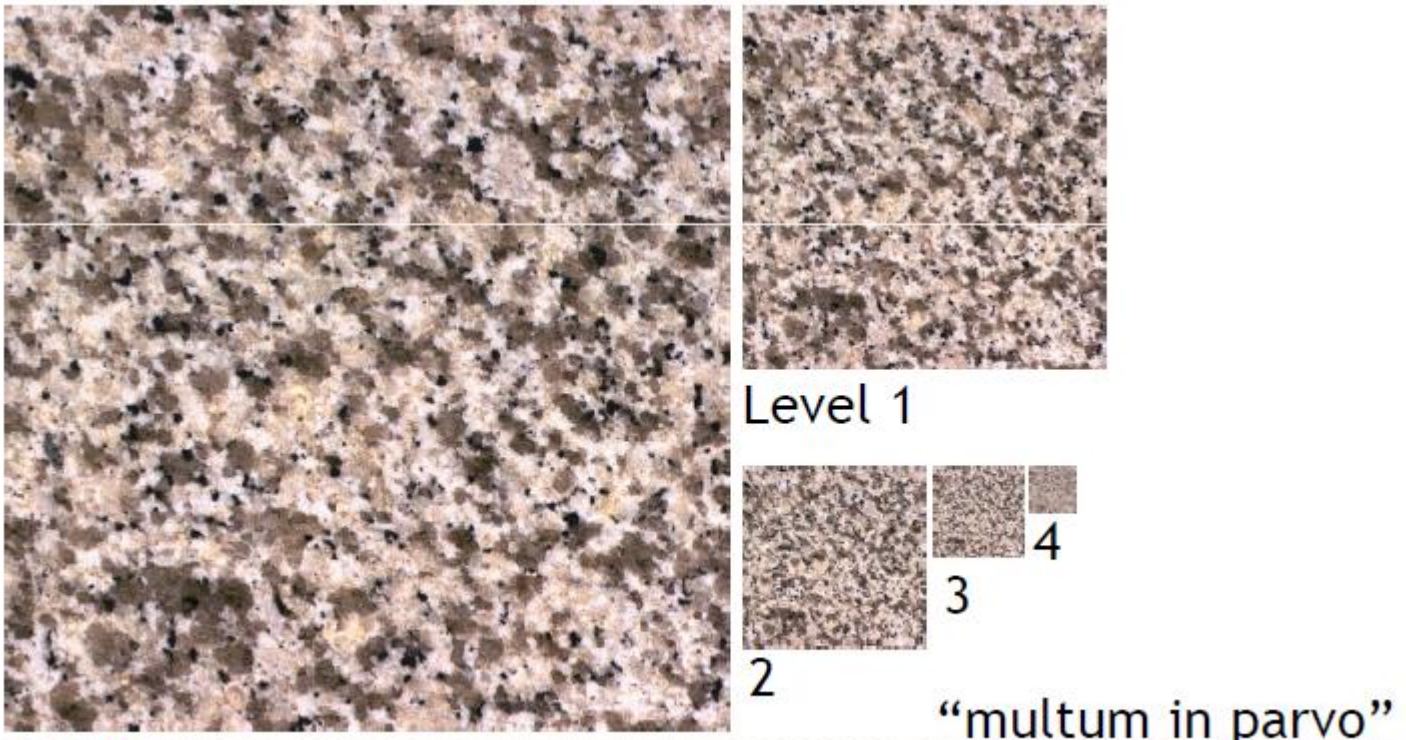
Mipmapping

- The basic idea is to construct a pyramid of images that are prefiltered and down sampled.



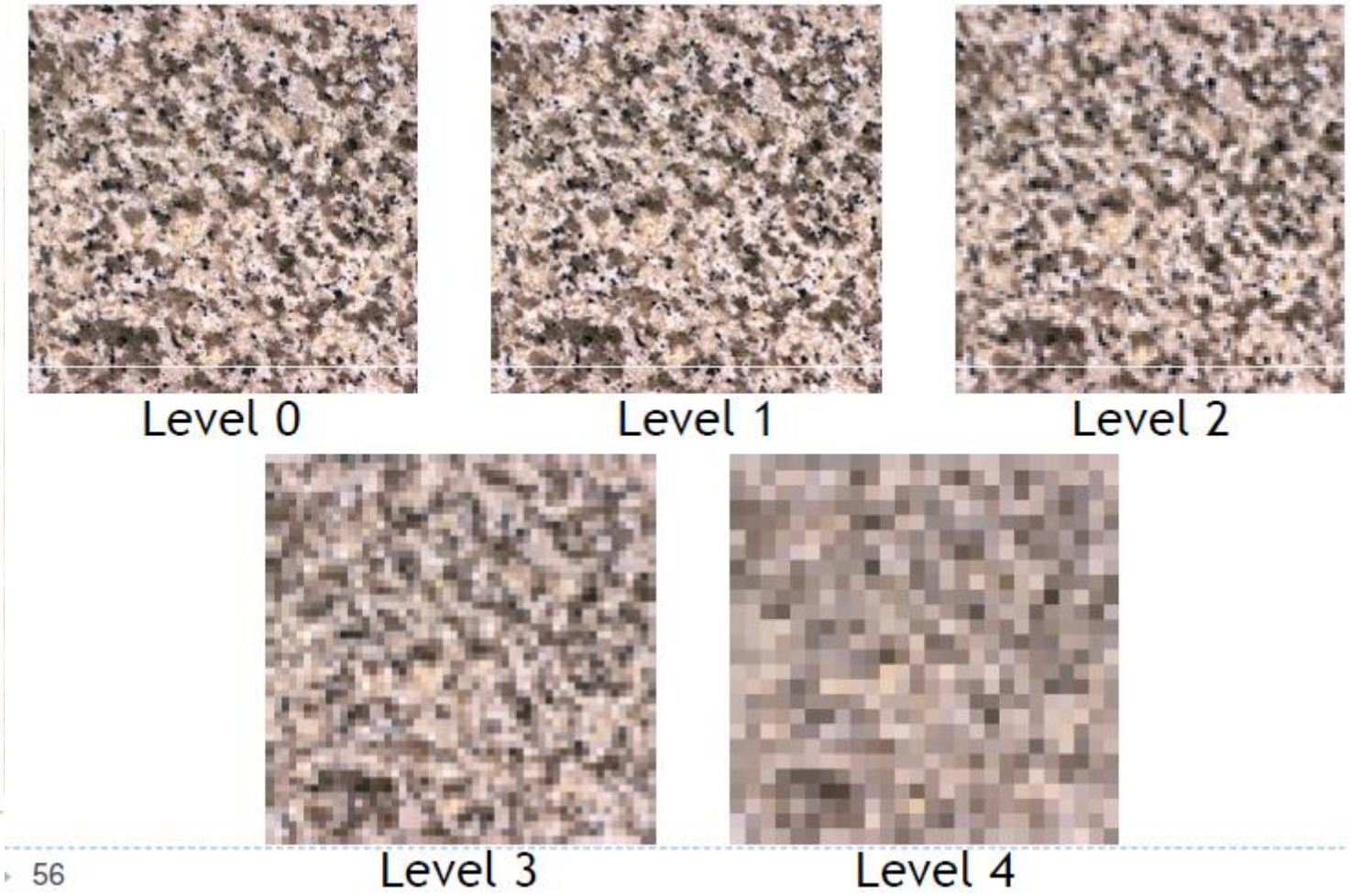
Mipmapping (continued)

Example: resolutions 512x512, 256x256, 128x128, 64x64, 32x32 pixels.

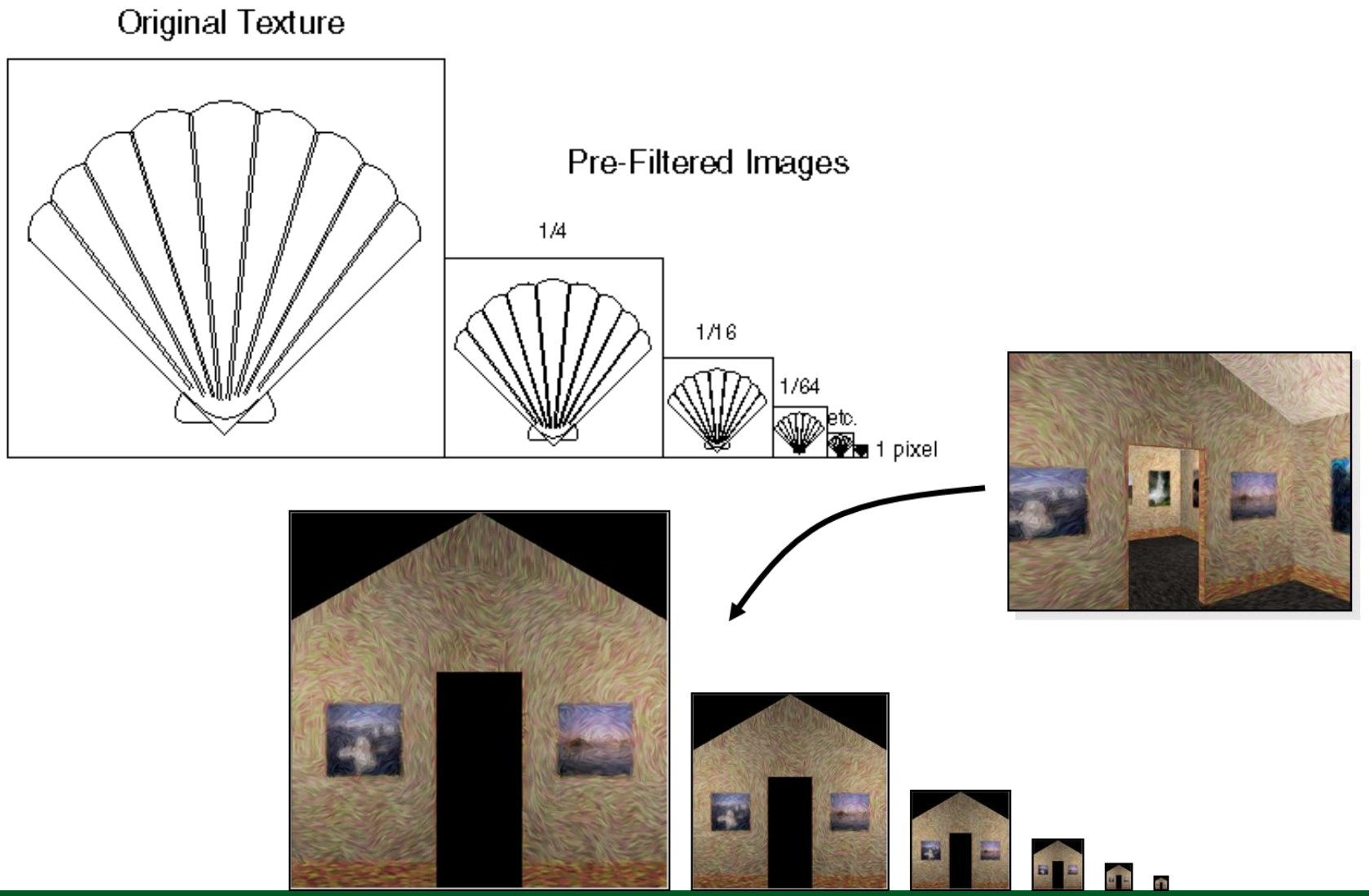


Mipmapping (continued)

- One texel in level 4 is the average of 256 texels in level 0



MipMapping Example



MIP-map Example

- No filtering:

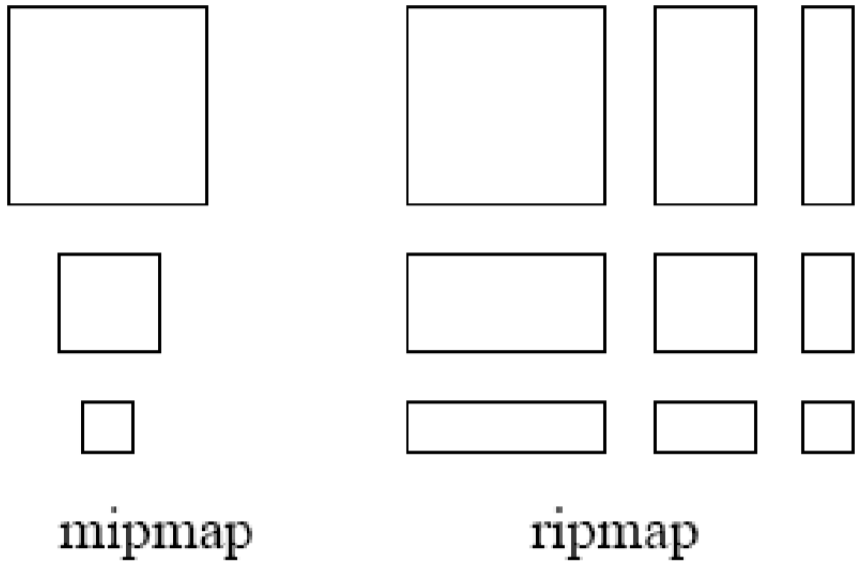
AAAAAAAGH
MY EYES ARE BURNING

- MIP-map texturing:

Where are my glasses?

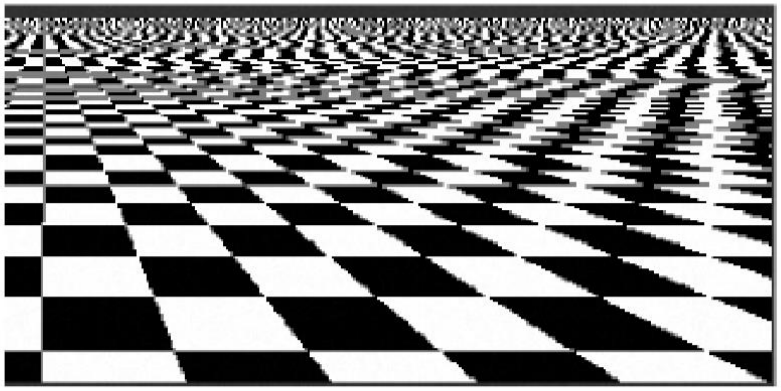
Mipmapping

- **Anisotropic(各向异性) Filtering**
  - **Build a ripmap instead of mipmap by pre-computing the anisotropic filtering.**

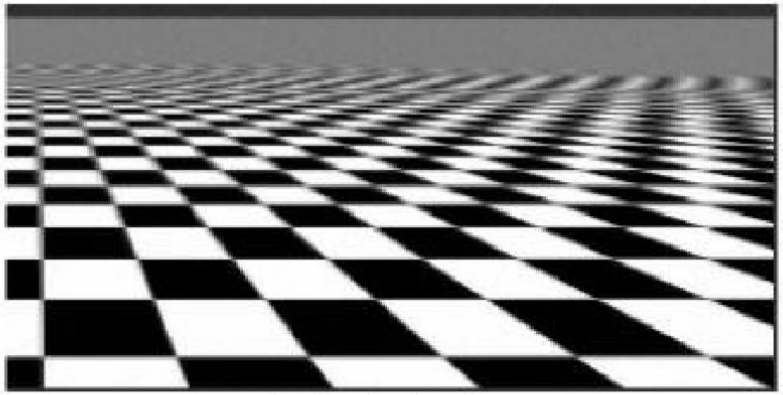


Mipmapping

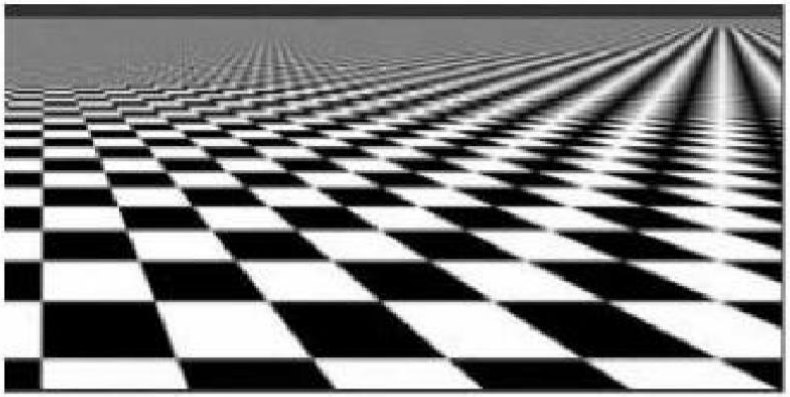
- **Anisotropic(各向异性) Filtering**



aliasing



isotropic filtering



anisotropic filtering

Texture Mapping in OpenGL

- Create and bind texture objects
- Load textures
- Set texture filter and wrap mode
- Pass textures to OpenGL
- Enable texturing
- Supply texture coordinates for vertex

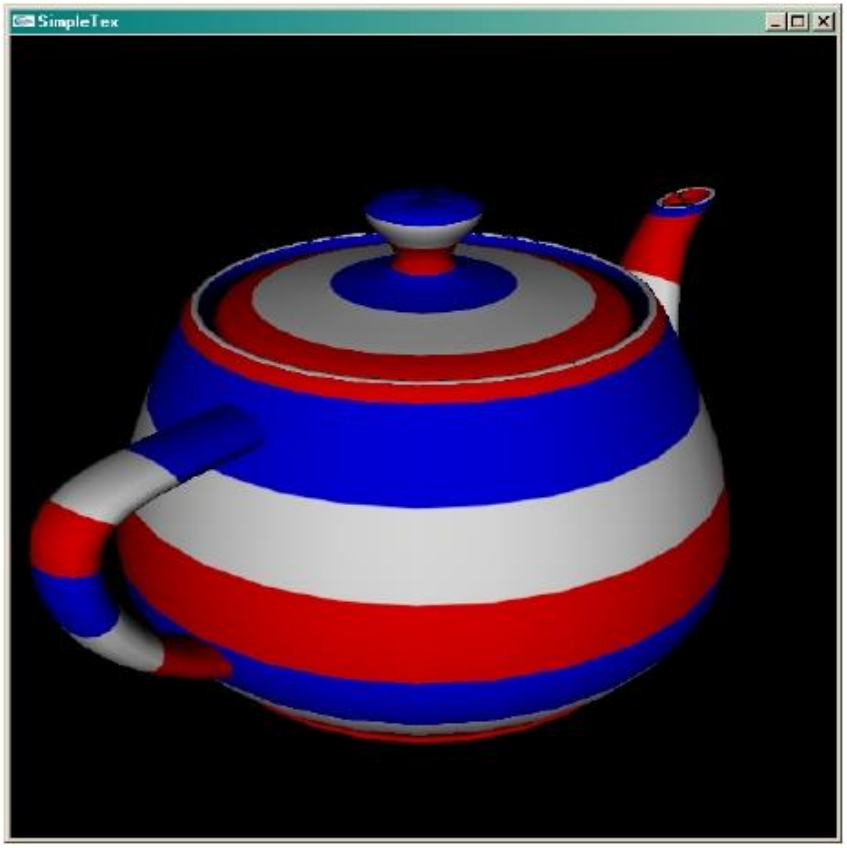
Texture Mapping in OpenGL

```
// In the init() routine
 glBindTexture (GL_TEXTURE_2D, texObjects[textureID] );

// Load the image
Image.ReadImage("grass.tga");

// Set up texture environment
   glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WARP_S,REPEAT);
   glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WARP_T,REPEAT);
   glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG FILTER, GL_LINEAR);
   glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN FILTER, GL_LINEAR);

// Pass texture to OpenGL
glTexImage2D(GL_TEXTURE_2D, 0, GLRGBA8, image.GetWidth(), image.GetHeight(),0,
GL_RGBA, GL_UNSIGNED_BYTE, (const void*)image.GetImageBuffer());
```

Texture Mapping in OpenGL

```
void display(void)
{
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glEnable(GL_DEPTH_TEST);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_TEXTURE_2D);
   glBindTexture (GL_TEXTURE_2D, texObjects[textureID] );
   glPushMatrix();
   glTranslatef(0.0, 0.0, -50.0);
   glColor3f(1.0, 1.0, 1.0);
   // Draw the scene with geometric and texture coordinates
   glBegin(GL_QUADS);   // Draw a rectangle
  // Assign texture coordinates to each vertex
  glTexCoord2f(0.0, 0.0); glVertex2f(-1.0, -0.5);
  glTexCoord2f(1.0, 0.0); glVertex2f( 2.0, -0.5);
  glTexCoord2f(1.0, 1.0); glVertex2f( 2.0,  1.0);
  glTexCoord2f(0.0, 1.0); glVertex2f(-1.0,  1.0);
   glEnd();
        glPopMatrix();
        glutSwapBuffers();   }
```

Texture Mapping in OpenGL

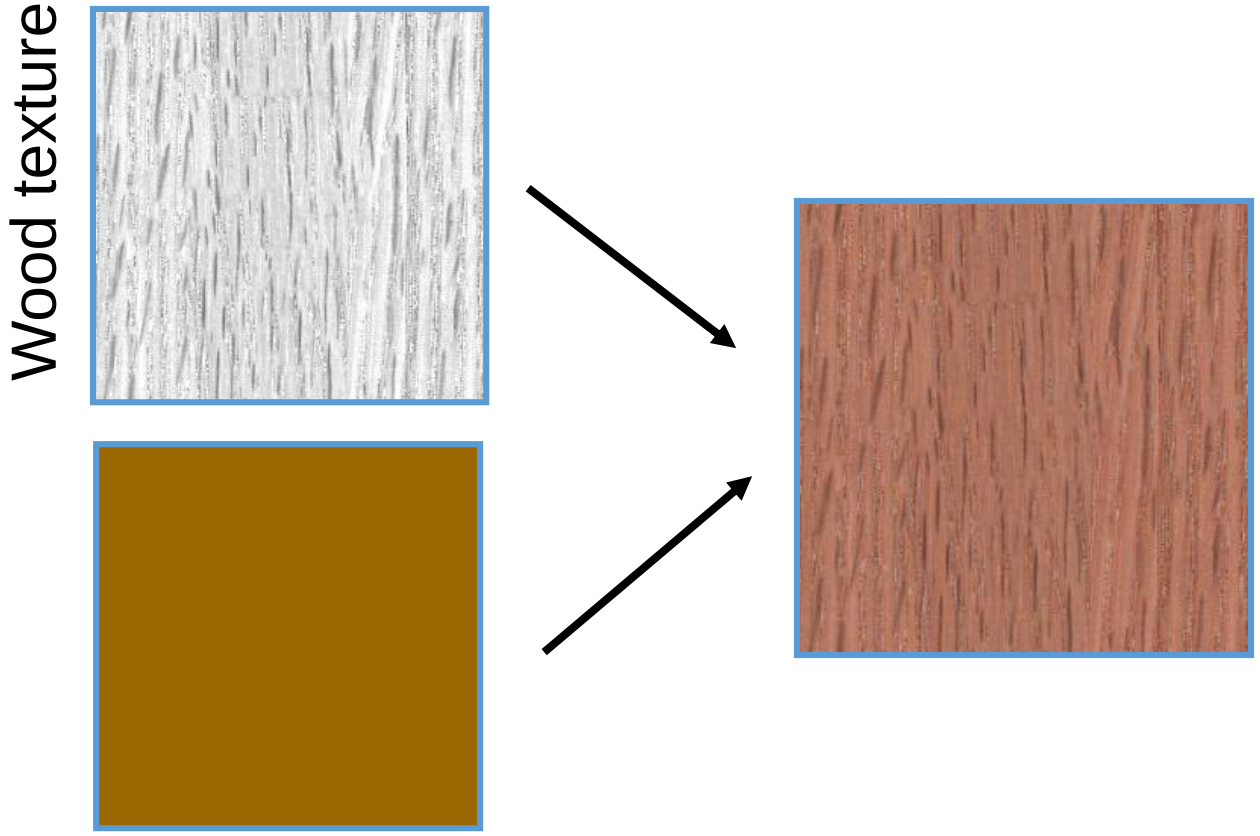


Texture Mapping Applications

- Modulation, light maps
- Bump mapping
- Displacement mapping
- Illumination or Environment Mapping
- And many more

Modulation textures

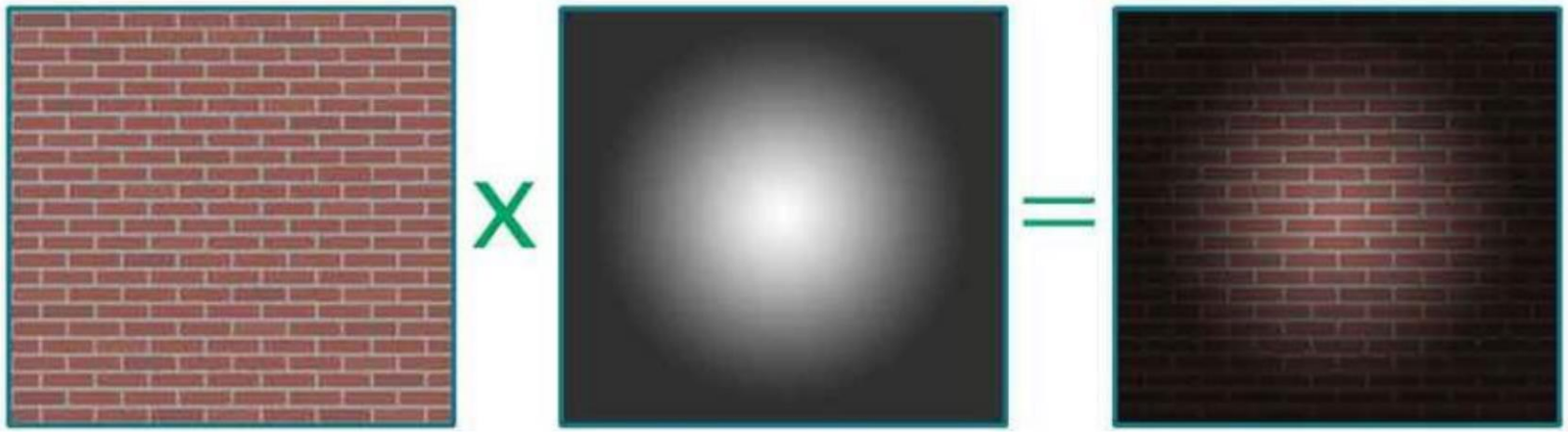
Map texture values to scale factor



$$I = T(s,t)(I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L + K_T I_T + K_S I_S)$$

Light Maps

- Instead of wallpapering a polygon with the texture's colors, we can blend the texture with the existing colors.
- A "light map" is such a black-and-white texture; white texels will cause the underlying pixels to appear brighter and shinier, and vice versa.



diffuse object color
(color texture)

light map
(light texture)

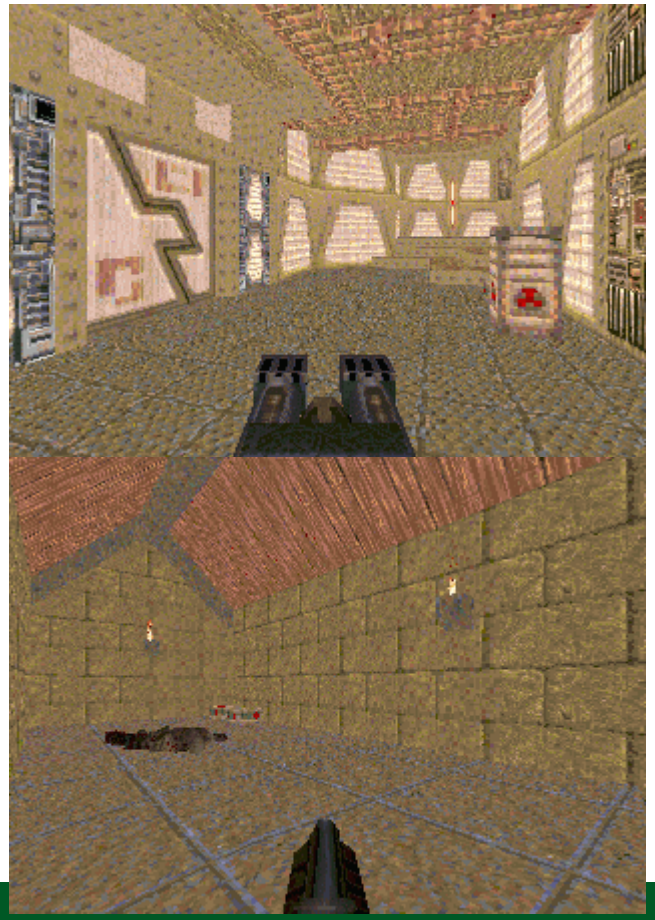
diffuse reflection component

Light Maps

- Light maps are used to store pre-computed illumination

Light map image by Nick Chirkov

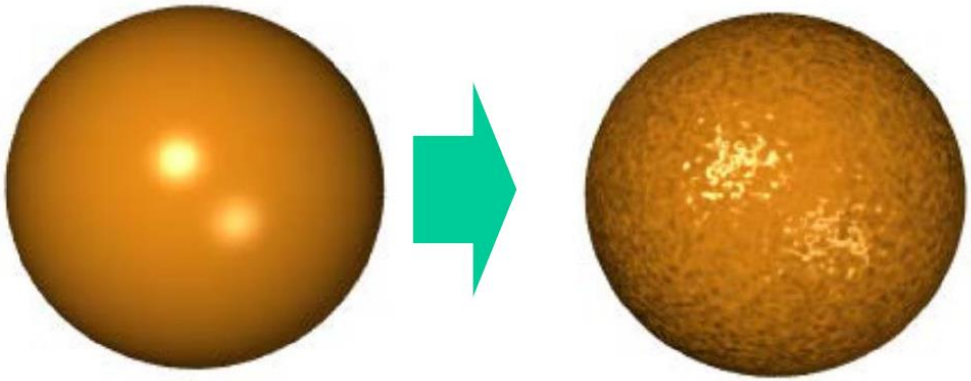
Texture only



Texture map+ light map

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**



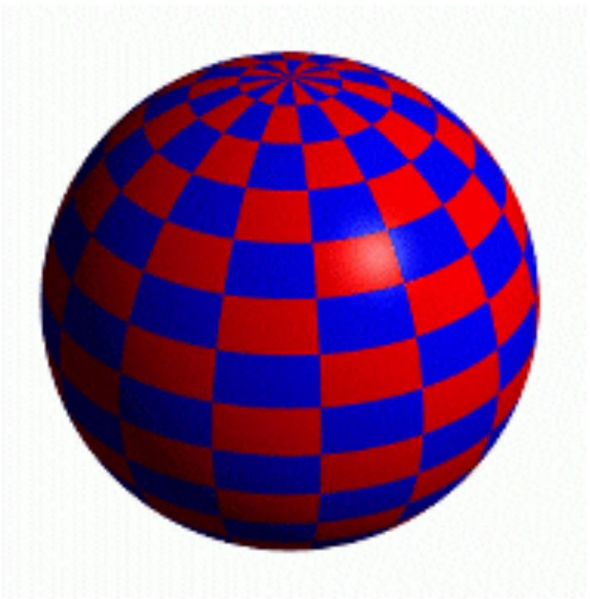
如此得到的图像就会显现出形状变化的错觉

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。

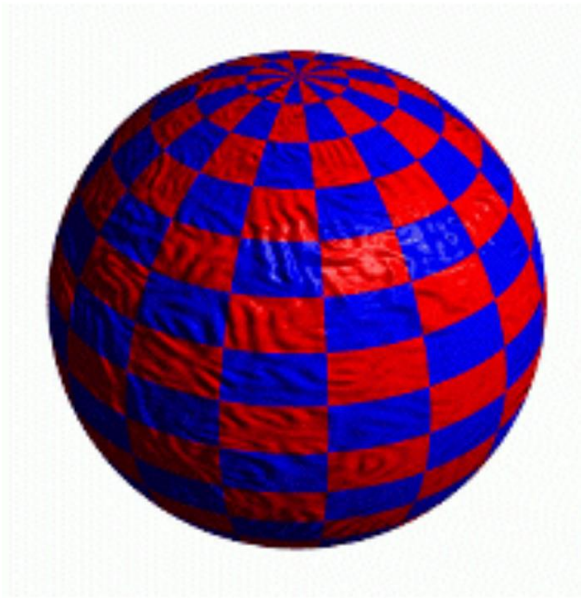
Bump Mapping (凹凸贴图)

Bump Mapping assumes that the Illumination model is **applied at every pixel** (as in Phong Shading or ray tracing).



Sphere w/Diffuse Texture

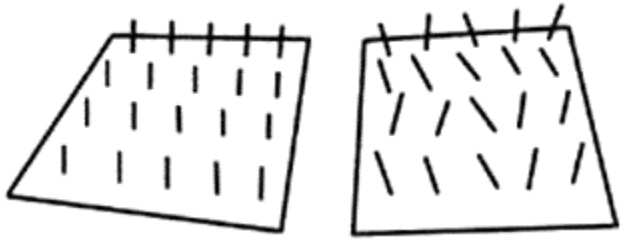
Swirly Bump Map



Sphere w/Diffuse Texture & Bump Map

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

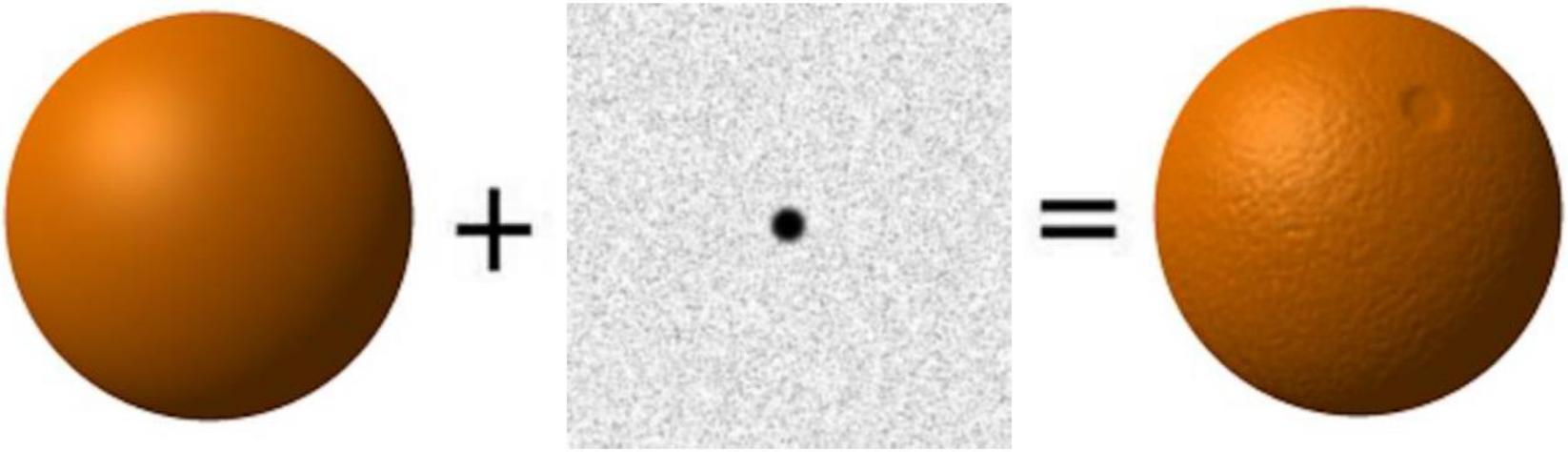
$$\mathbf{p}'_u = \mathbf{p}_u + \frac{\partial d}{\partial u}\mathbf{n} + d(u, v)\mathbf{n}_u$$

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

Method of perturbation(3)

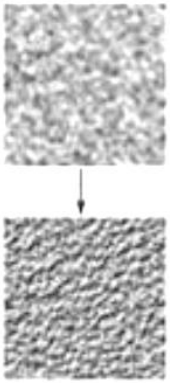
- **To obtain the approximate perturbed normal:**

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

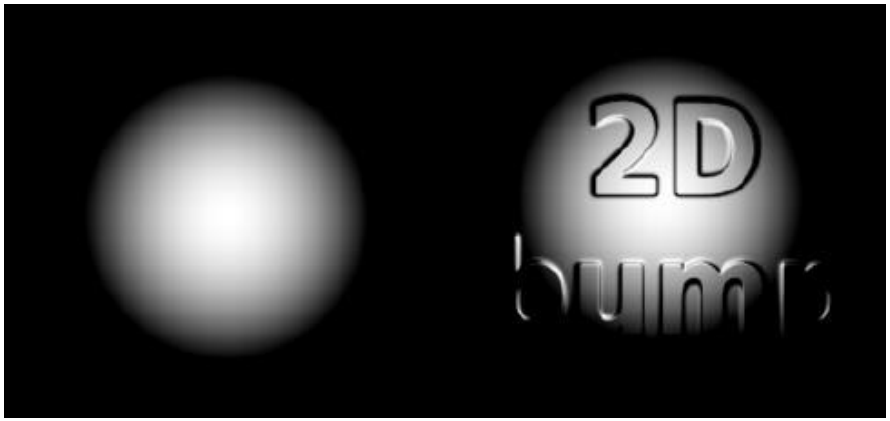


Example

http://www.paulsproject.net/tutorails/simblebump/simblebump.html



Example



http://www.paulsprojects.net/tutorials/simplebump/simplebump.html

Displacement Mapping (置换贴图)

- Bump mapping adds realism, but it only changes the appearance of the object.
- We can do one better, and actually change the geometry of the object
  - This is displacement mapping
- 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**.

Displacement Mapping

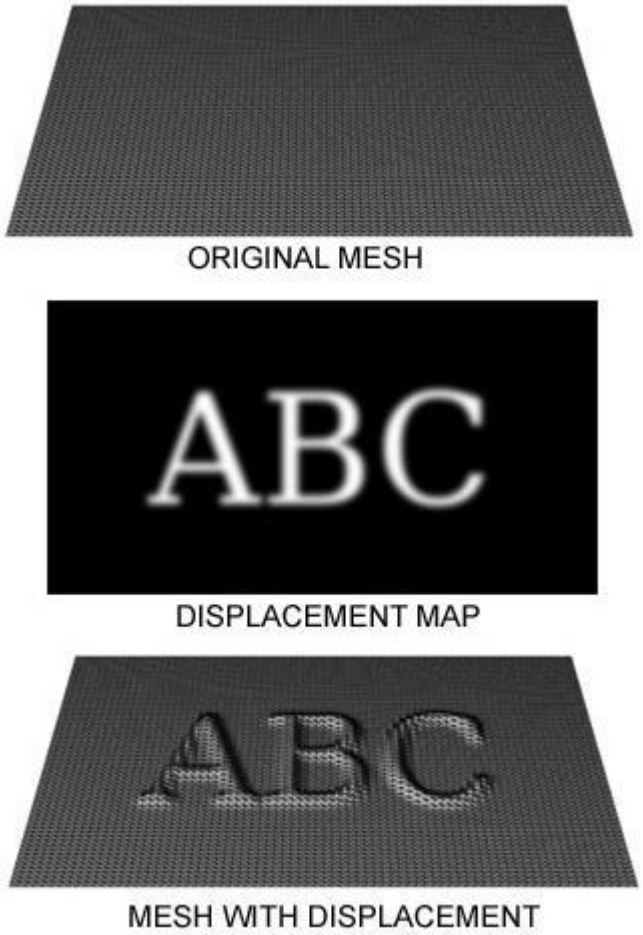
- Displacement mapping shifts all points on the surface in or out along their normal vectors
  - Assuming a displacement texture *d,*

    $\mathbf{p'} = \mathbf{p} + d(\mathbf{p}) * \mathbf{n}$

- Note that this actually changes the vertices, so it needs to happen in geometry processing

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

Displacement Mapping

Bump Mapping

Displacement Mapping

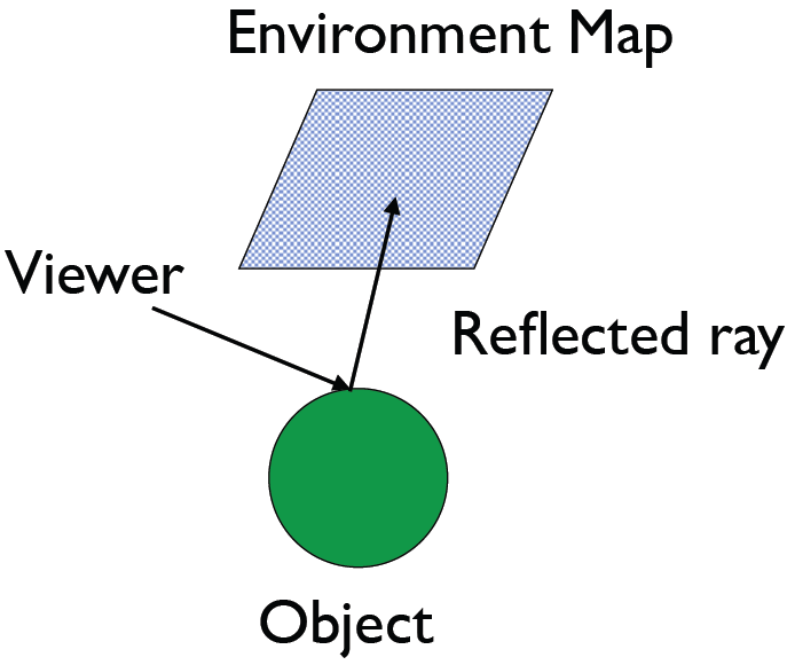
Environment Mapping

- Simulate complex mirror-like objects
  - Use textures to capture environment of objects
  - Use surface normal to compute texture coordinates

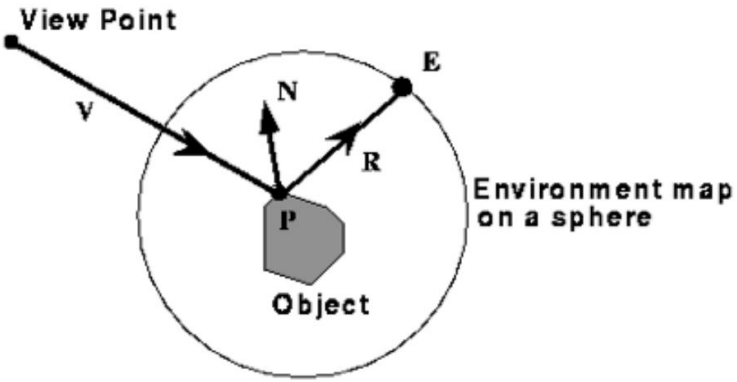
Environment Mapping

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



- We use the *direction* of the reflected ray to index a texture map.
- We can simulate reflections. This approach is not completely accurate. It assumes that all reflected rays begin from the same point, and that all objects in the scene are the same distance from that point.

Environment Mapping

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

Solution

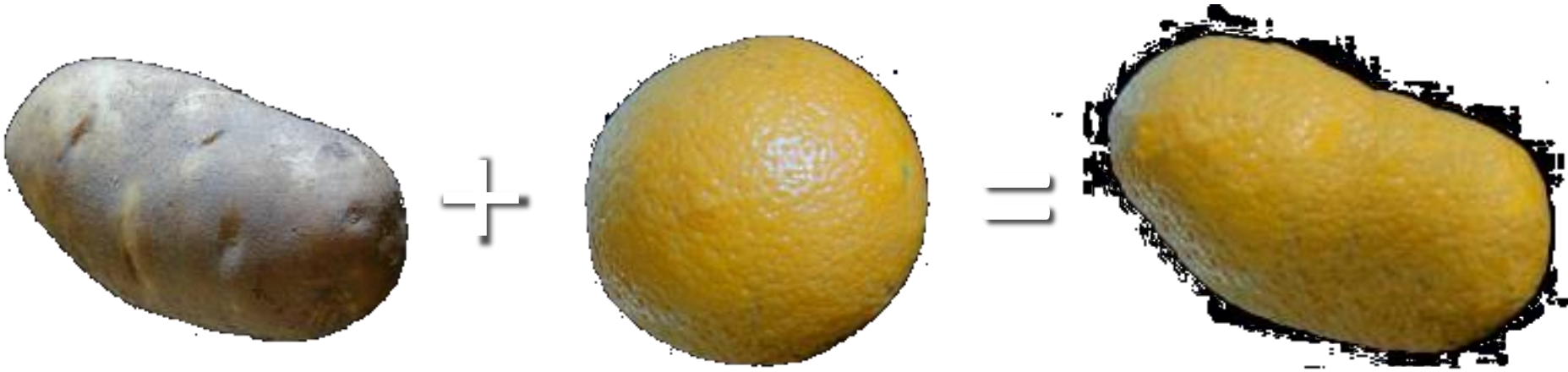
- A physically based rendering method, such as a ray tracer, can produce this kind of image.
- Ray-tracing calculations are too time-consuming to be practical for real-time applications
- Expand the texture mapping method：can give approximate results that are visually acceptable through environment maps.

Texture Application: Synthesis

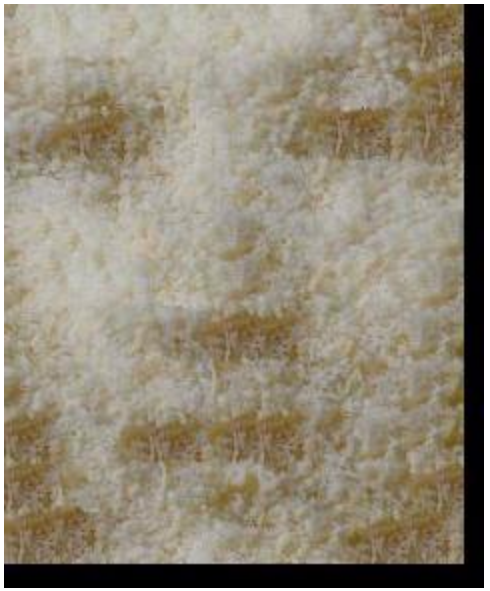
Texture Application: Synthesis

Application: Texture Transfer

- Try to explain one object with bits and pieces of another object:

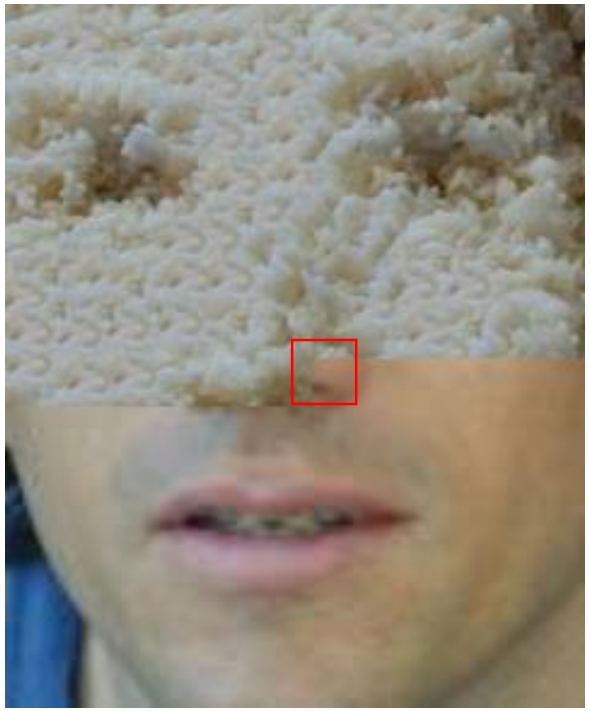


Texture Transfer



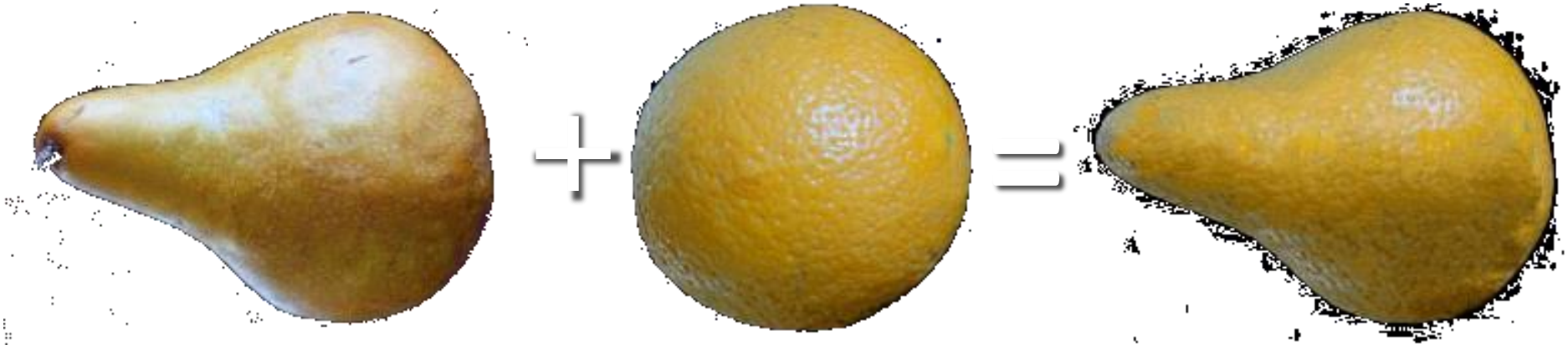
Texture Transfer

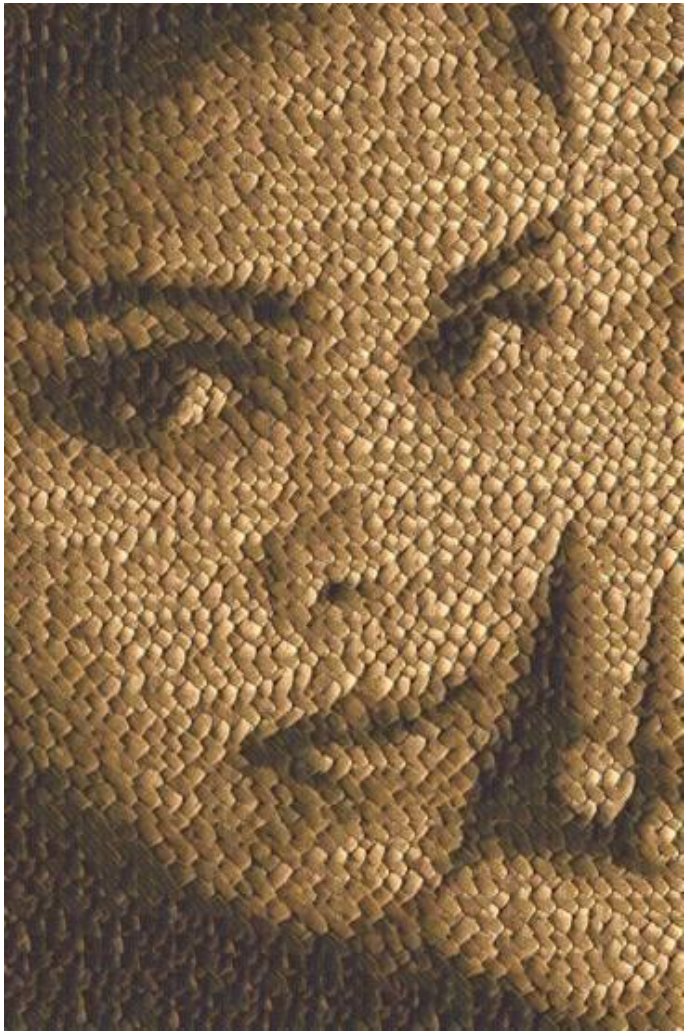
- Take the texture from one image and “paint” it onto another object



Same as texture synthesis, except an additional constraint:

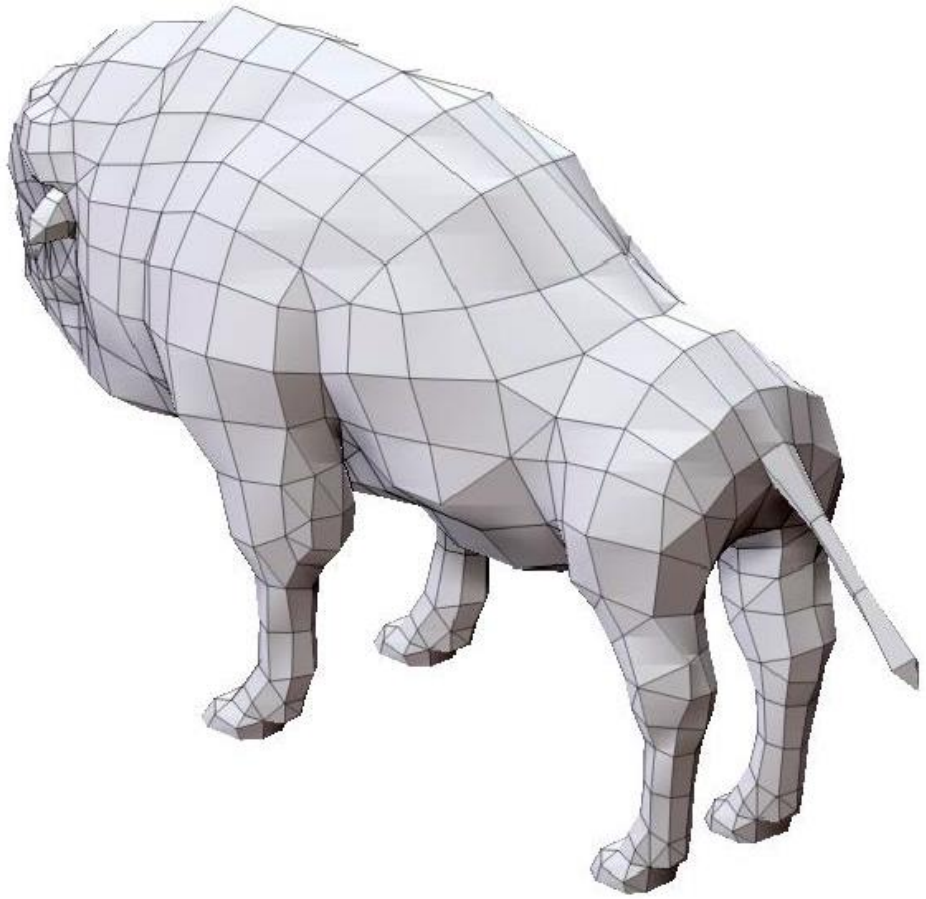
1. Consistency of texture
2. Similarity to the image being “explained”





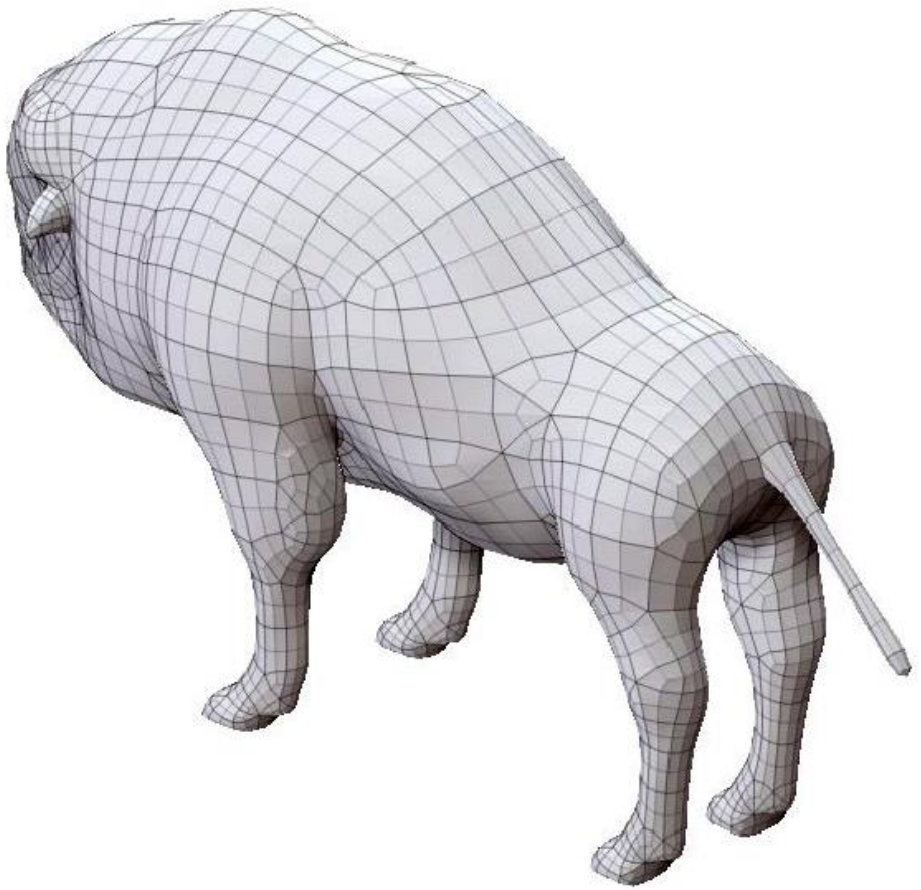
Coarse-to-Fine Sculpting Modeling

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



Coarse-to-Fine Sculpting Modeling

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

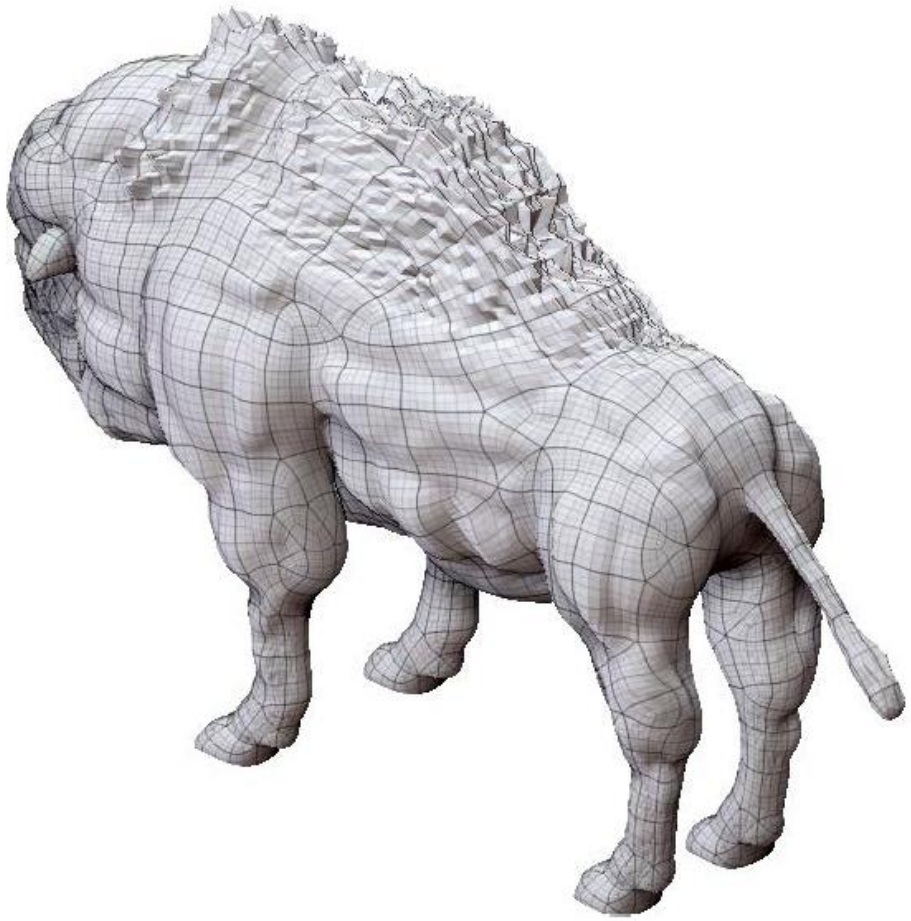


Coarse-to-Fine Sculpting Modeling

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

Coarse-to-Fine Sculpting Modeling

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

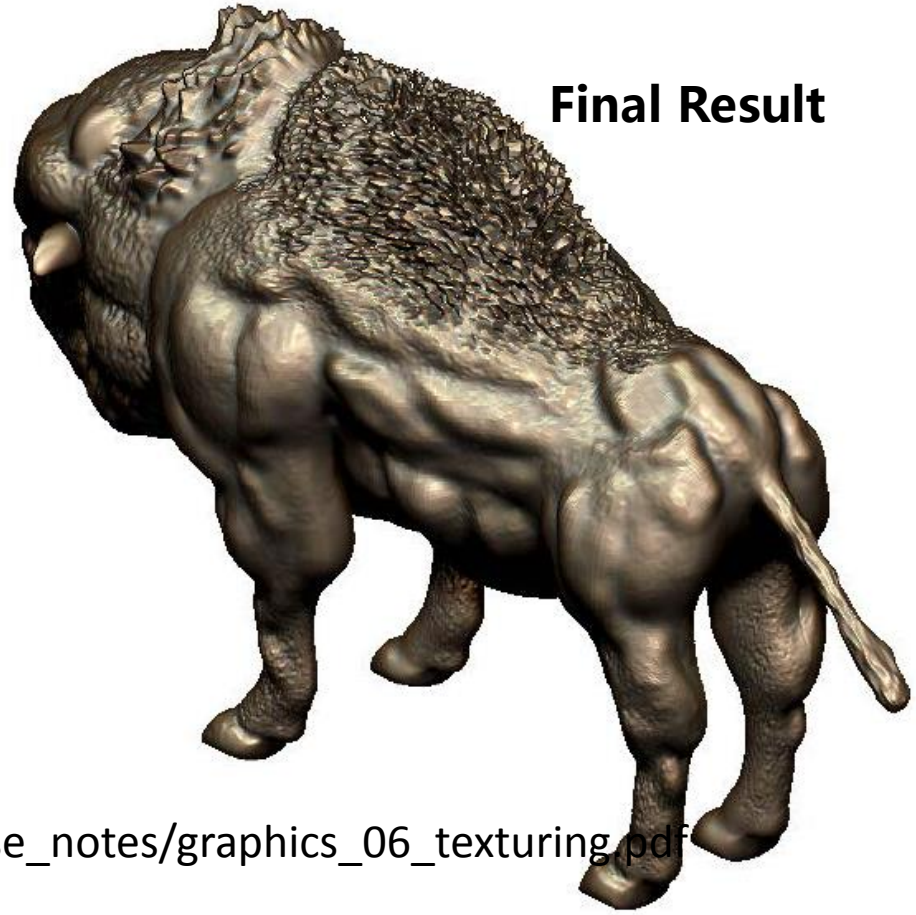


Coarse-to-Fine Sculpting Modeling

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

Coarse-to-Fine Sculpting Modeling

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



Final Result

http://cg.informatik.uni-freiburg.de/course_notes/graphics_06_texturing.pdf