Computer Graphics

Shi-Min Hu

Tsinghua University

Texture

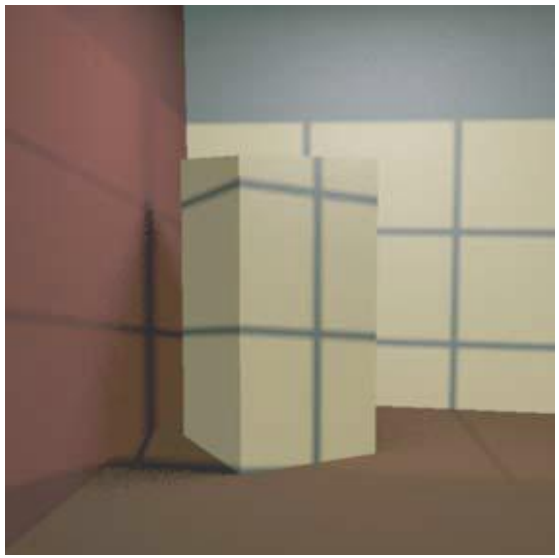
- **Texture**
  - **Texturing is a fundamental technique in Computer Graphics, allowing us to represent surface details without modeling geometric material details.**
  - **We will describe the basic ideas and applications of texture in this course**

Texture

- **Texture**

  Left Image (Without texture) ,Right Image (With texture) ,Note the differences between two images.

  The right one is more realistic.



Texture

- **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**

Texture

- **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.**

Texture

- **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).**

Texture

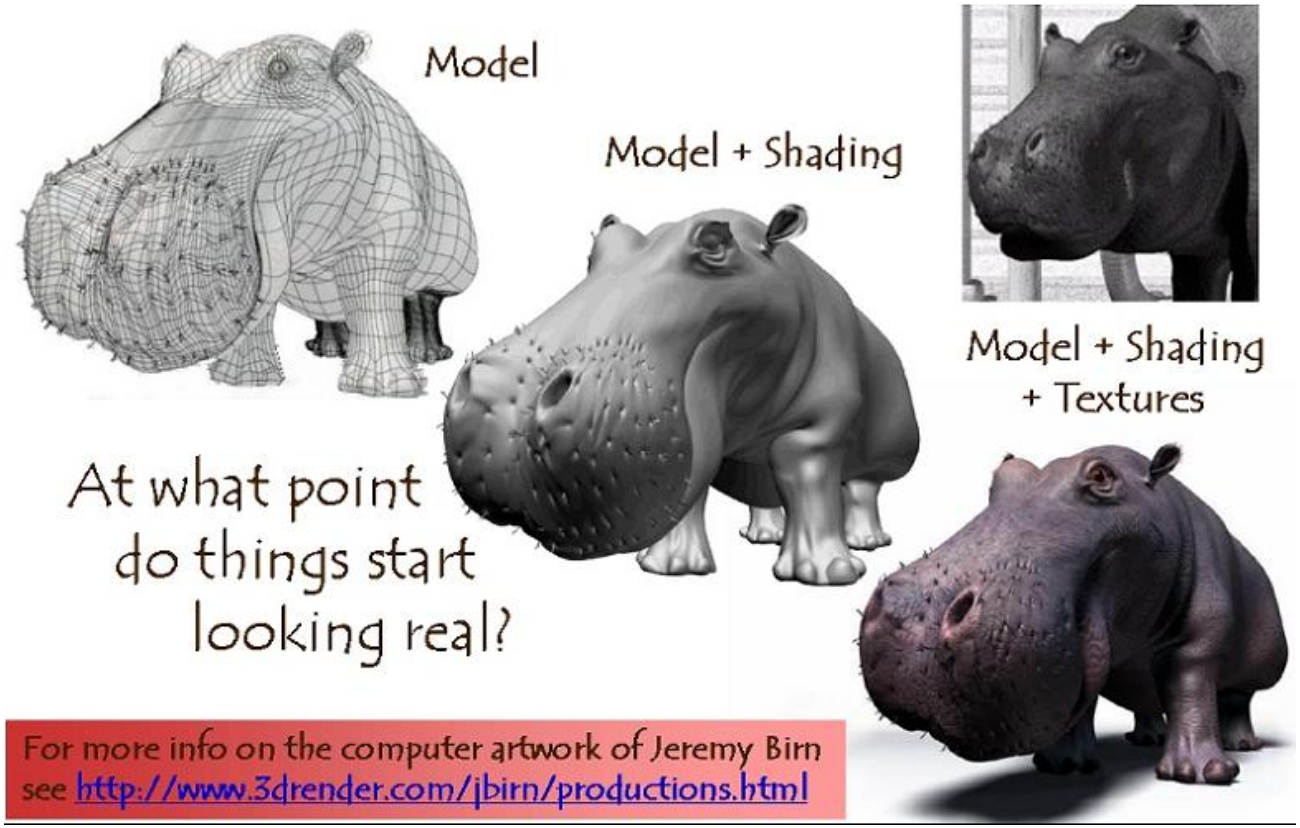
- **Why texture is useful?**
  - **Texturing resolves the two problems.**
    - **First, by representing the surface as a texture image, you don't have to painfully model all the geometric and material details. This saves time and resources and allow users to do other more important things.**

Texture

- **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, either via ray tracer or hardware . This saves computers time and resources.**

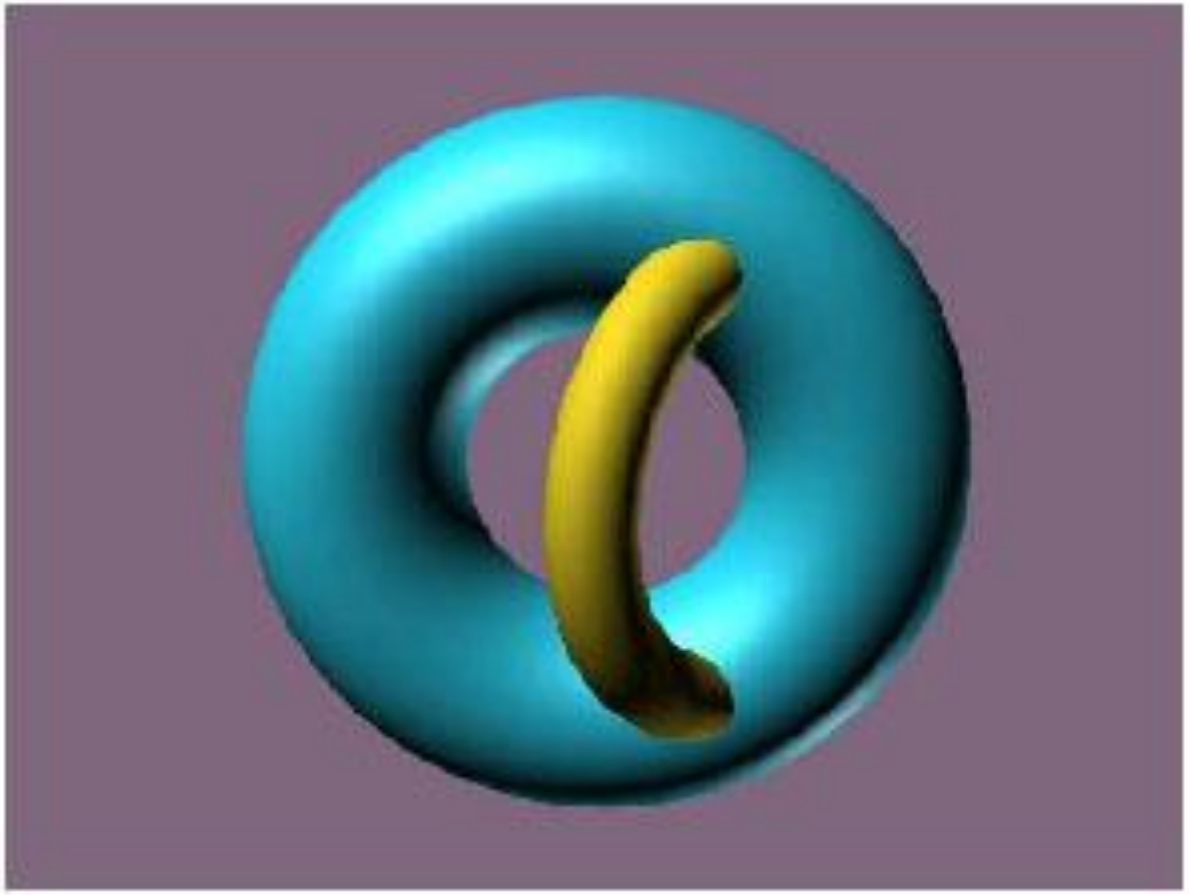
Texture

- **Why we need texture?**

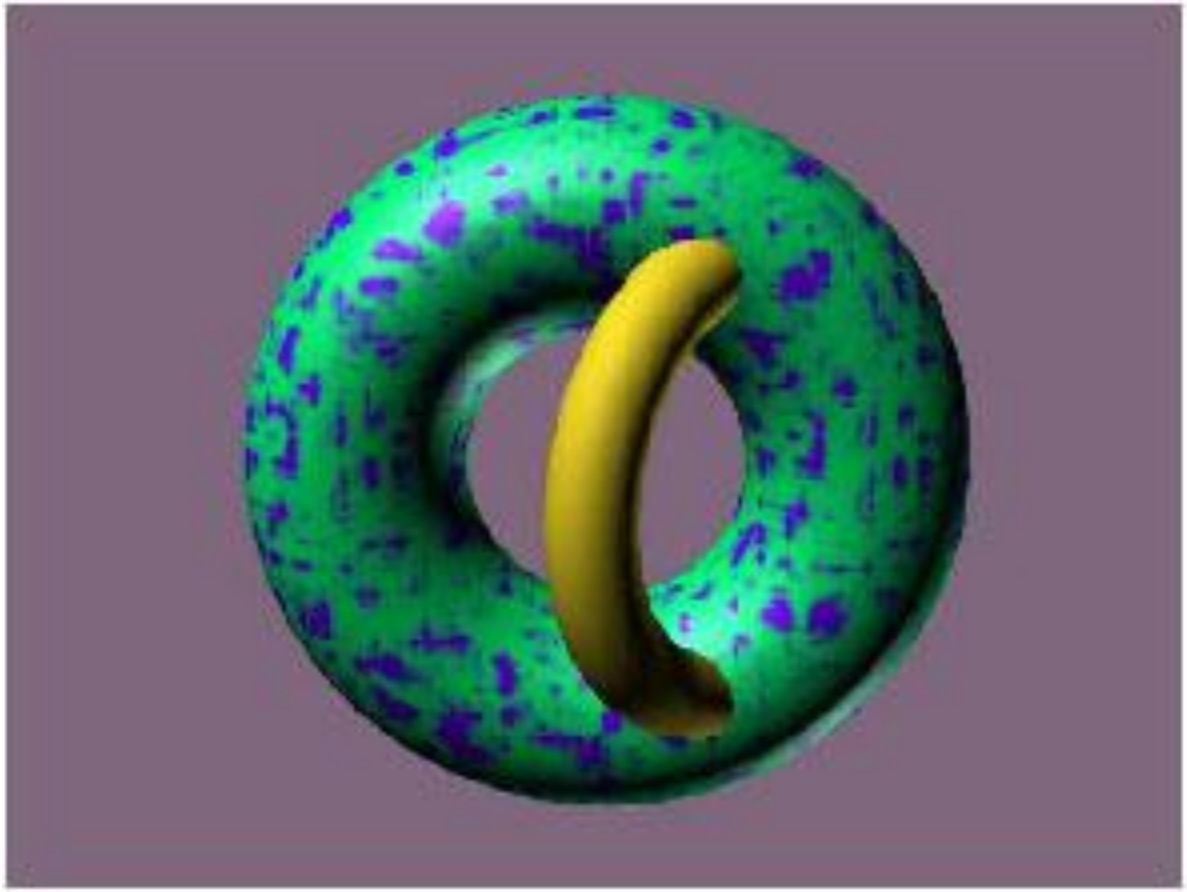


More texture example

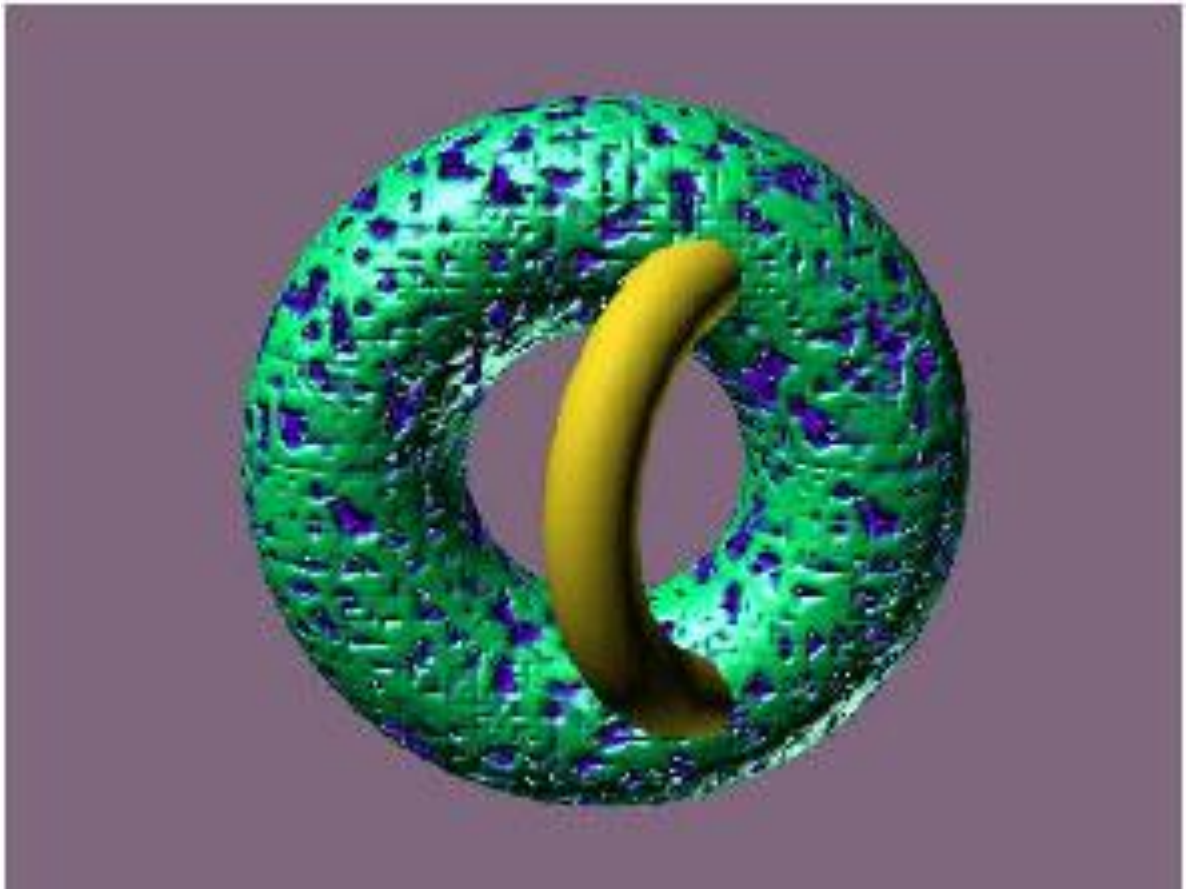
Original Model



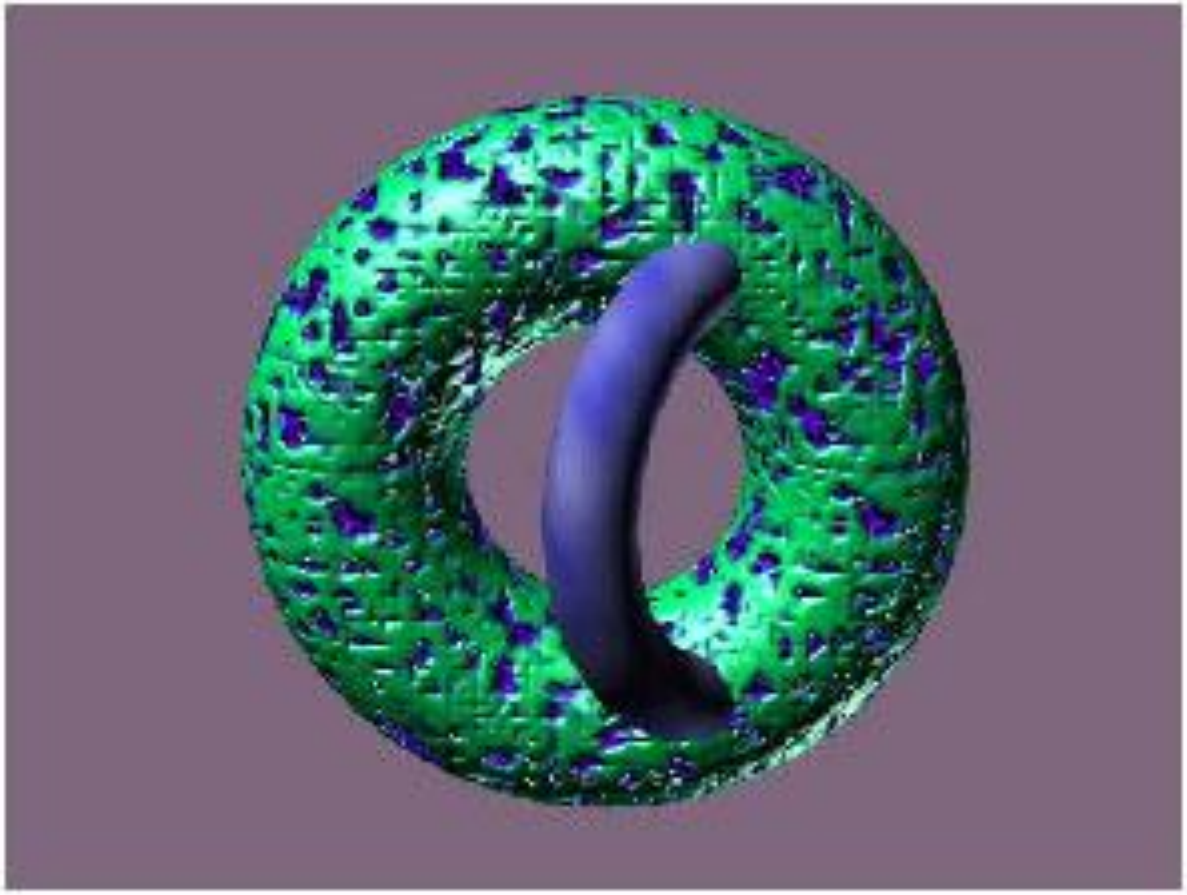
Two dimensional texture



Bump texture (几何纹理)



Combination



- Model and texture image

- Model with texture

- Another image

- With new texture

- With another texture

- Use bump texture related with the image

Texture

- **Texture usage**
  - **To use a texture, we usually need 3 steps**
    - **First, texture acquisition (获取). This could be done by manual drawing,**

      by taking photographs,

      by procedure texture(过程纹理)

      by texture synthesis (纹理合成)

Texture

- **Texture usage**
  - **To use a texture, we usually need 3 steps**
    - **Second, Texture mapping (贴图).**

      Given the texture image and a 3D model, you need to figure out how to map the texture onto the model. This could be done by specifying each vertex of 3D model a texture coordinate, and assigning the color from texture image using this coordinate.

Texture

- **Texture usage**
  - **To use a texture, we usually need 3 steps**
    - **Third, texture filtering(滤波). after you have decided how to map the texture onto the object, you will have to carefully sample the texture in the rendering process, otherwise, you may see some undesirable artifacts (人为痕迹) rooted in the signal processing theory**

Texture Acquisition

Texture Acquisition

- **The first task of texturing is texture acquisition. There are a variety of methods to generate the texture image.**
  - **You could simply draw a texture by hand.**
  - **You could simply take a photo of the material which you want to model.**
  - **Because we are neither artists(艺术家) nor camerist(摄影师), we will not focus on the two methods above, we will discuss two more technical methods, such as procedure texture(过程纹理) and 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

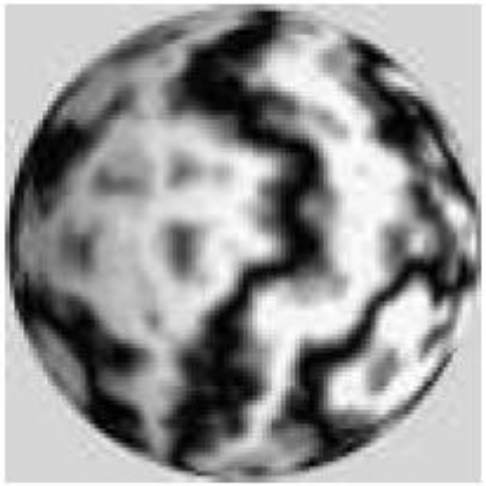
- **Advantages**
  - **They can be very compact, as they only need to store the procedure itself and some parameters.**
  - **By changing the parameters of the procedures, you can easily change the appearance of the materials, providing excellent controllability.**

Procedure Texture

- **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.**

Procedure Texture

- **The most popular procedure synthesis method : Perlin Noise [Perlin2002]**



marble

wood

Procedure Texture

- **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. Before introducing Perlin noise, we will first describe white noise.**

Procedure Texture

- **Purlin Noise**
  - **White Noise(白噪音)**
    - **A white noise is a signal that has uniform energy across all frequency bands; so the Fourier transform of a white noise will be rough flat.**
    - **It can be generated via a uniform random function.**
    - **For example, to generate a 2D white noise image, you could fill in each pixel with a value from a uniform random function in the range [0,1]**

Procedure Texture

- **Perlin Noise**
  - **Unlike white noise, perlin noise is a band limited function. It can be constructed as a summation of white noises at different frequency bands, as below:**

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

  - ***n* is the total number of band, *p* is the persistence, and *i* is the band number ,with *i=0* the lowest frequency band.**

Procedure Texture

- **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].**

Procedure Texture

- **Perlin Noise**

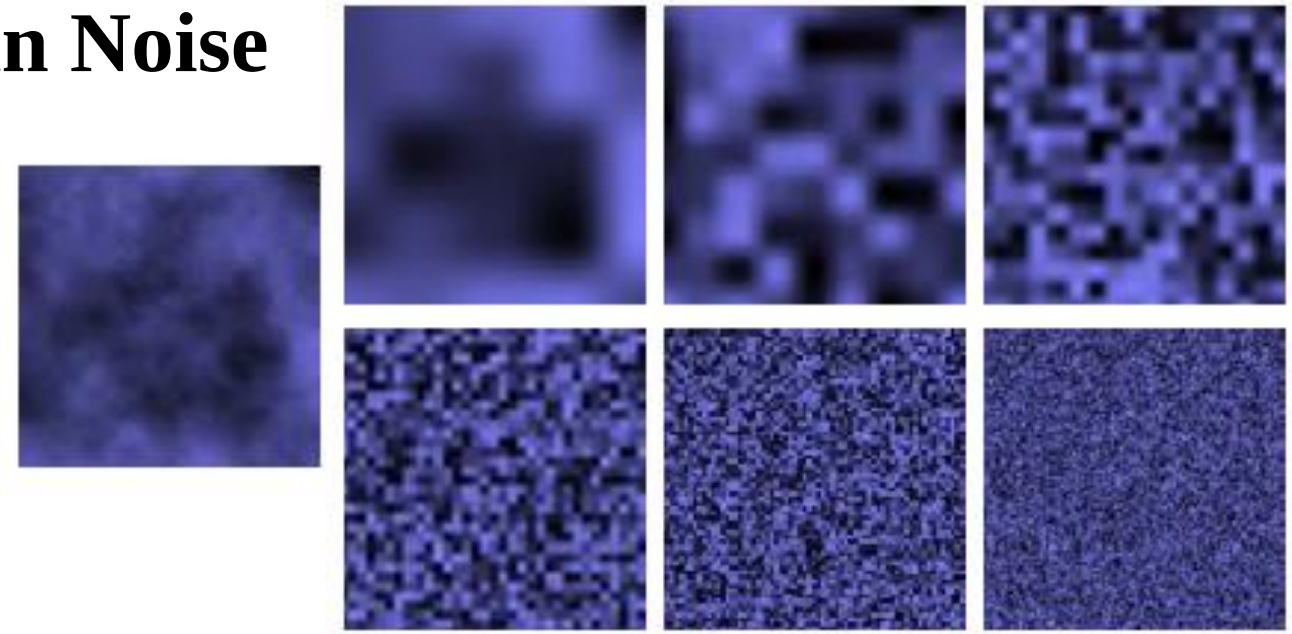


Figure 1: *2D Perlin noise example. Left: the Perlin noise image. Right: the individual noise bands, from low to high frequencies. Image courtesy of [Elias 2003].*

Procedure Texture

- **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 Synthesis

Texture Synthesis

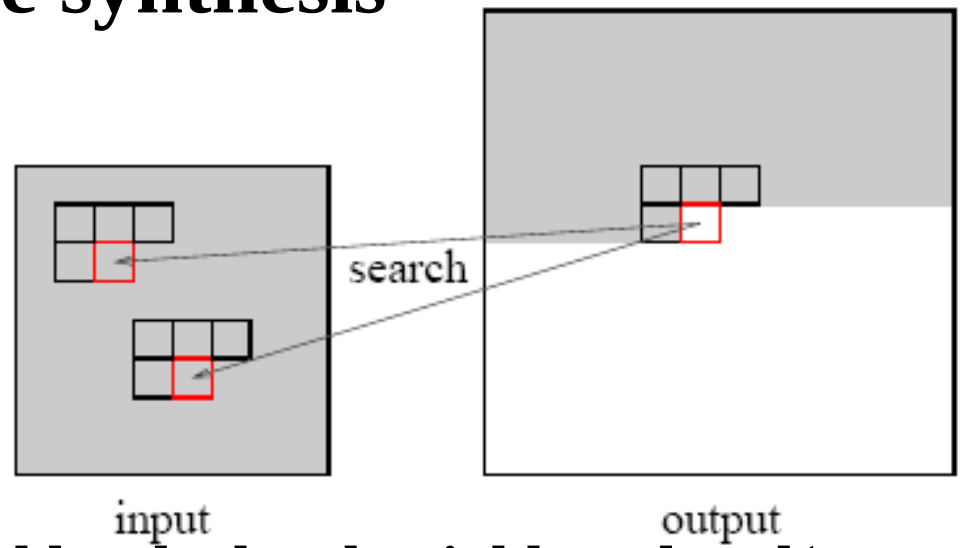
- **What is texture synthesis?**
  - **An alternative approach is to synthesize a new texture from a given example. This is certainly more user friendly, because to use the algorithm, all you need to do is to provide an image sample.**
  - **As much research has been devoted to texture analysis in both the computer vision and graphics community, so we have many practical algorithms.**

Texture Synthesis

- **In computer graphics, texture synthesis techniques could be classified into 2 classes:**
  - **Pixel-based texture synthesis**
  - **Patch-based texture synthesis**

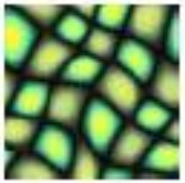
Texture Synthesis

- **Pixel based texture synthesis**
  - **Main idea**
    - **Synthesis a new texture pixel by pixel, where the value of each pixel is determined by the local neighbourhood(e.g. 3*3, 5*5), choosing the input pixel with a similar neighboorhood.**

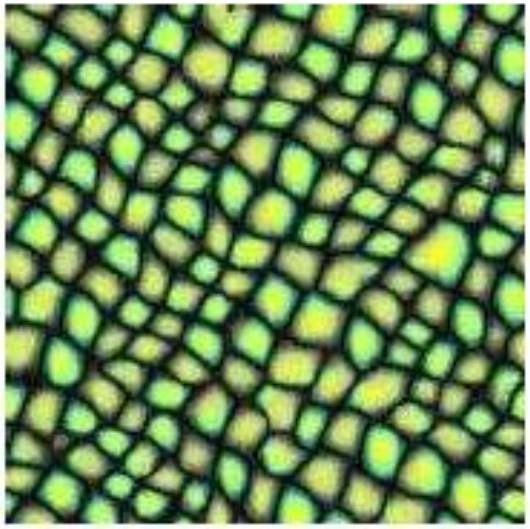


Texture Synthesis

- **Pixel based texture synthesis**
  - **These works include:**
    - **"Texture Synthesis by Non-parametric Sampling" [Efros99]**
    - **"Fast texture synthesis using tree-structured vector quantization" [Wei and Levoy00]**



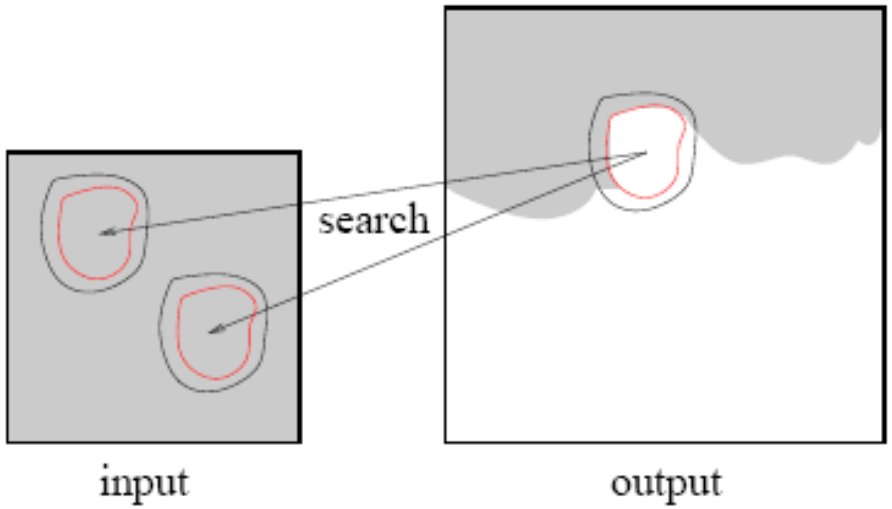
sample



result

Texture Synthesis

- **Patch-based texture synthesis**
  - **Pixel-based approaches could be improved by synthesis patches rather than pixels.**



  - **One of the most important algorithm is Graph-Cut [Kwatra03], which offers the best quality so far.**
  - **The patch is chosen also by matching neighborhoods.**

Texture Synthesis

- **Some result of Graph-Cut algorithm**

Input

Graph cut

Texture Synthesis

- **Some result of Graph-Cut algorithm**

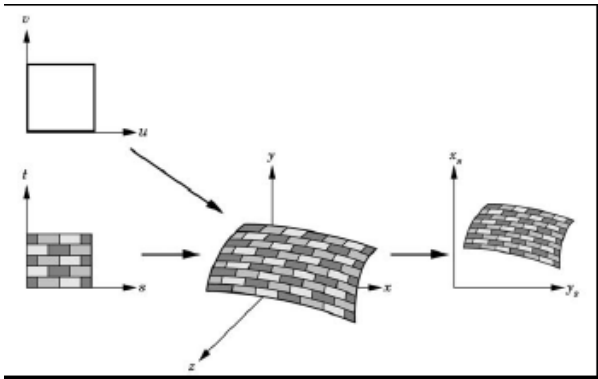
Sample Texture

Synthesized Texture

Texture Mapping

Texture Mapping

- **What is 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.**



Texture Mapping

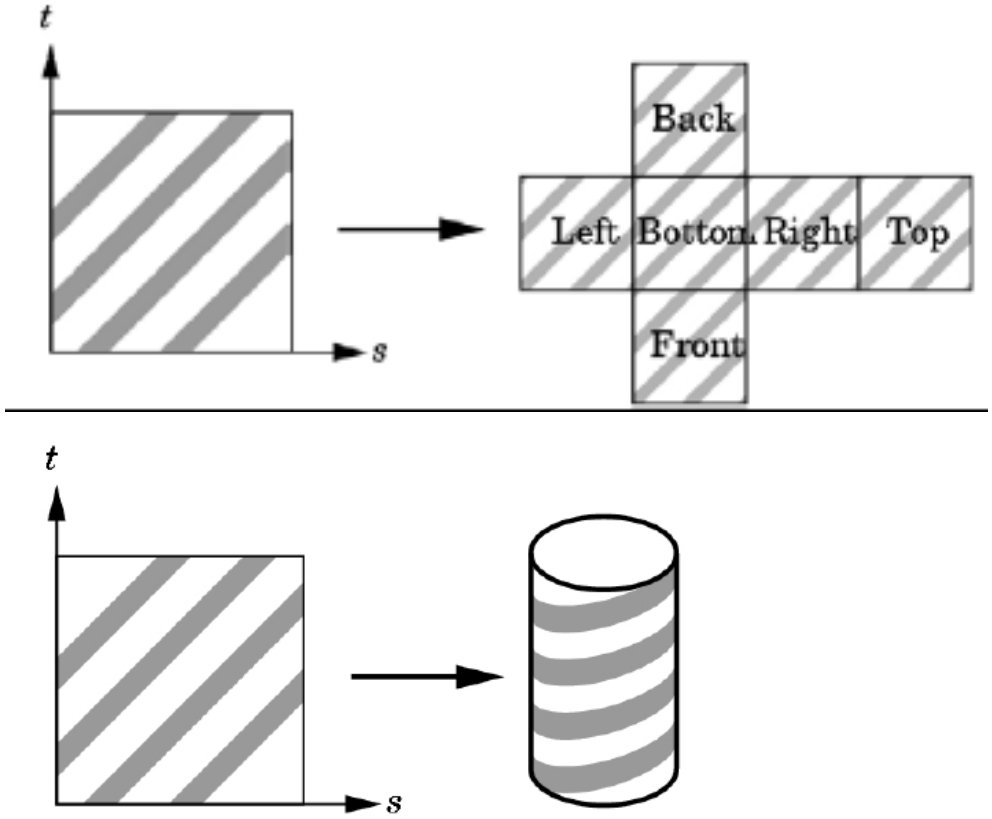
- **How to specify surface mapping function?**
  - **By natural parameterization**
    - **For regular objects, such as sphere, cube, cylinder**
  - **By manually specify texture coordinates**
    - **For complex objects, each vertex is specified a texture coordinate**

Texture Mapping

- **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.**

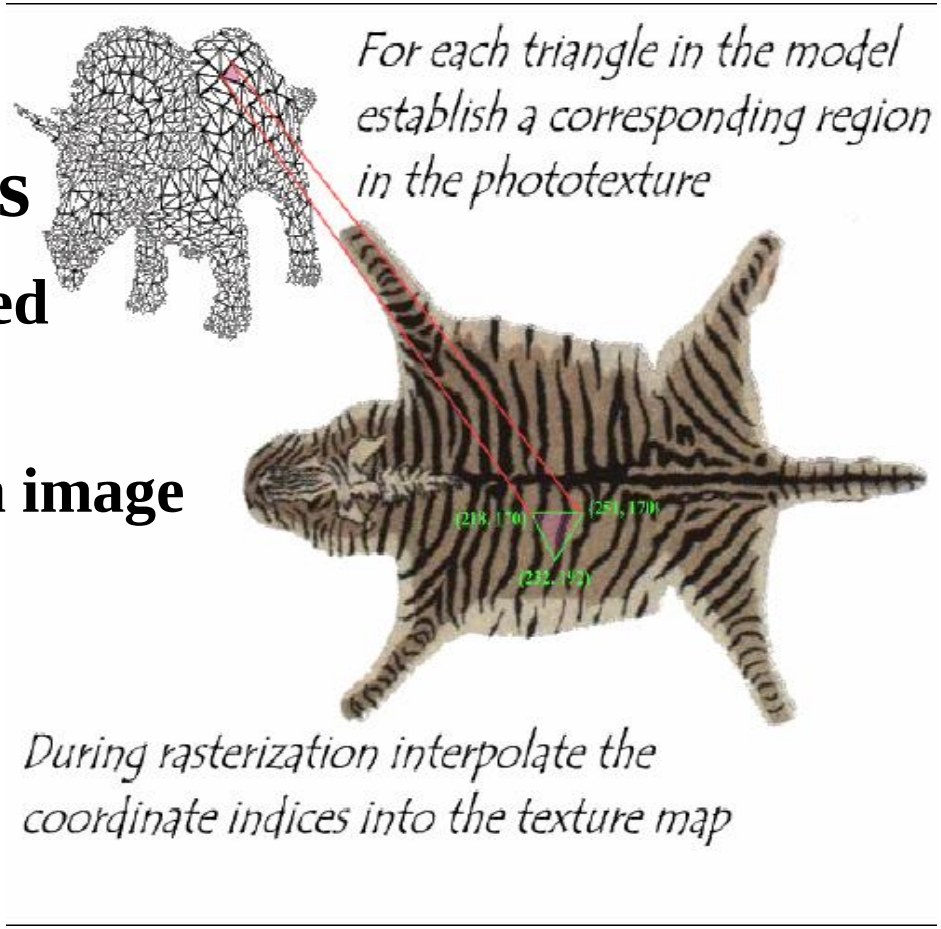
Texture Mapping

- **Natural parameterization**

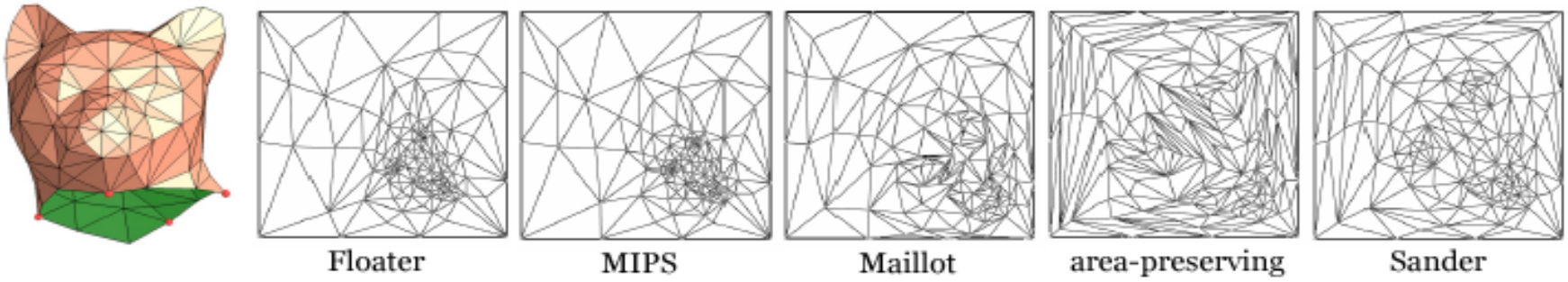


Texture Mapping

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

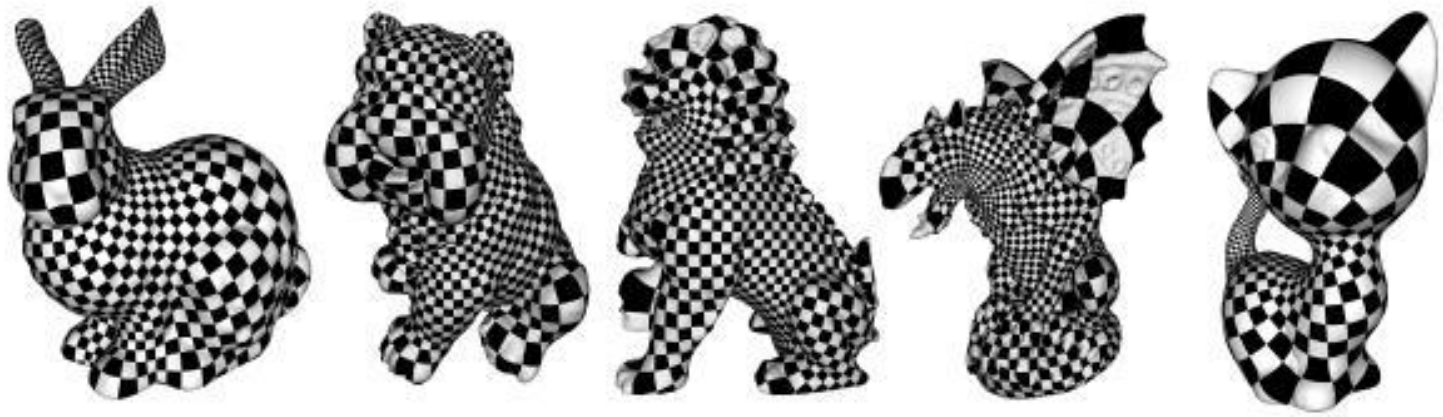


- Mesh parametrization
  - Construct a mapping from 3D mesh models to planar domain

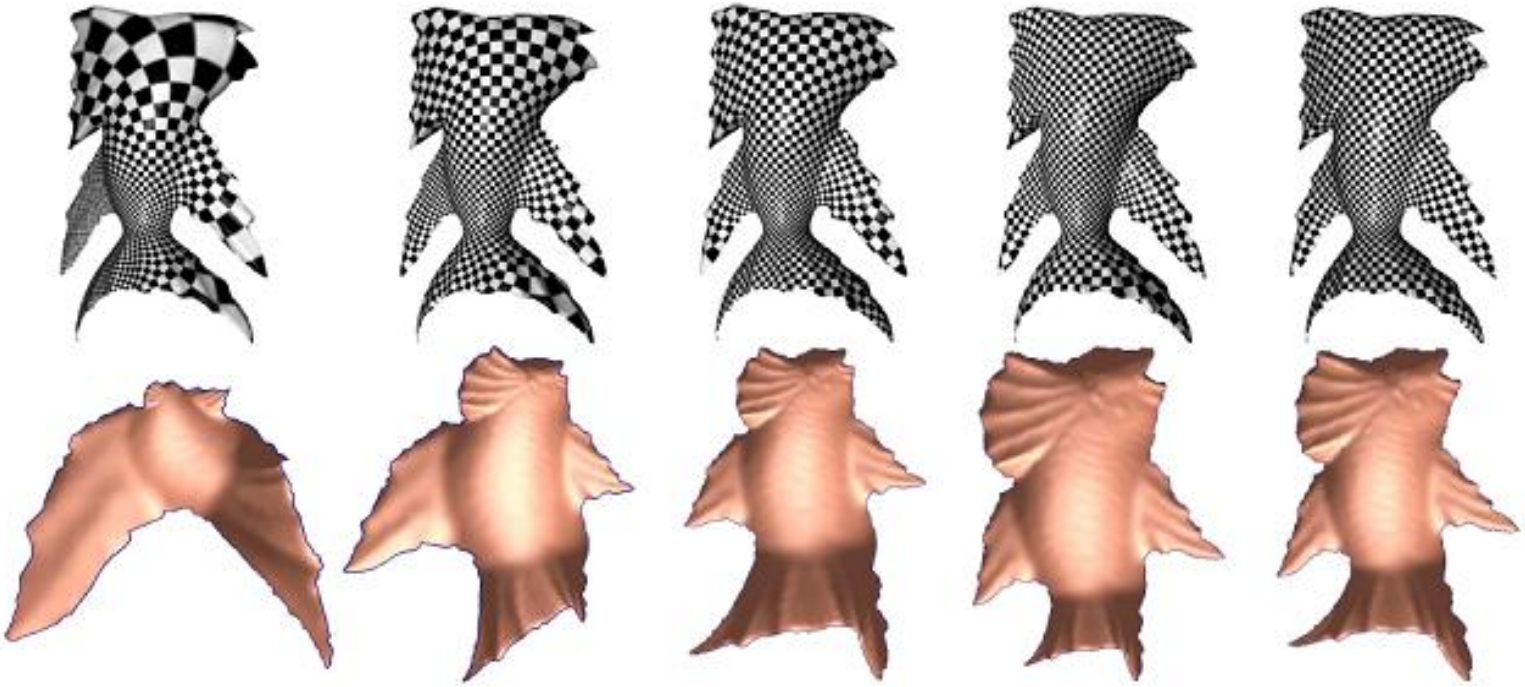


- Conformal parametrization and area preserving: inverse curvature map

在网格上从网格边的边长到网格顶点的高斯曲率的映射称之为曲率映射，利用共形几何理论，由网格曲面的曲率映射的切映射，可以证明这个映射是可逆的。



- Conformal parametrization and area preserving: inverse curvature map



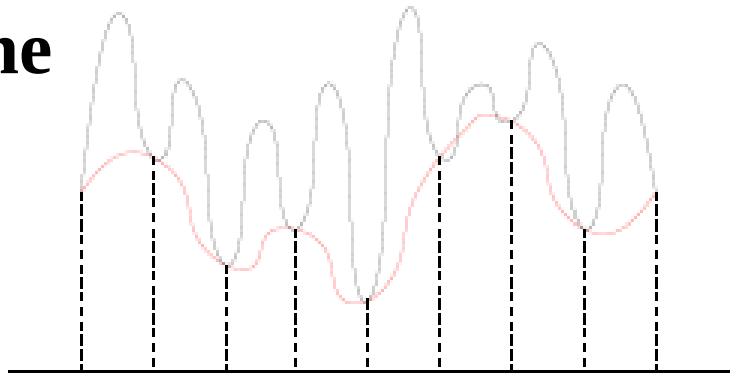
Texture Filtering

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)**

Texture Filtering

- **Aliasing(走样) and Anti-aliasing(反走样)**
  - **Generally, aliasing is caused when a signal is sampled at too low frequency rate. So that many high frequency features of the signal are missed.**
  - **e.g.: The black dash is the original signal, the red curve is the aliased signal**



aliasing case

Texture Filtering

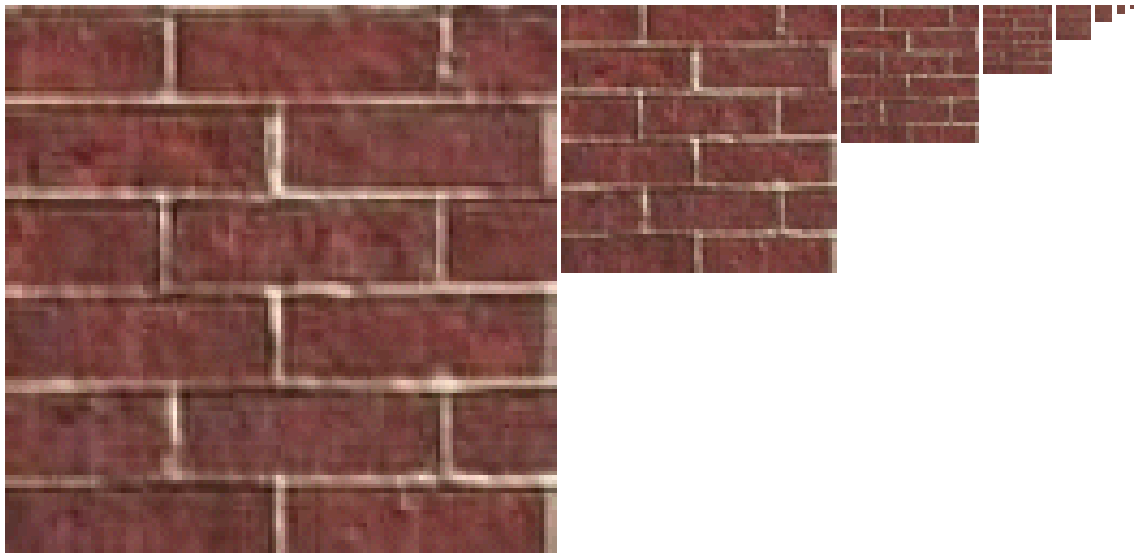
- To resolve this problem, a common method is to sample at a higher rate. However, this is not always feasible. For example, in graphics applications, the screen resolution limits sample rate.
- An alternative method would be to pre-filter the signal into a lower frequency one.

Texture Filtering

- **Isotropic(各向同性) filtering**
  - **Mipmapping :**
    - **Construct a pyramid of images that are pre-filtered 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.**

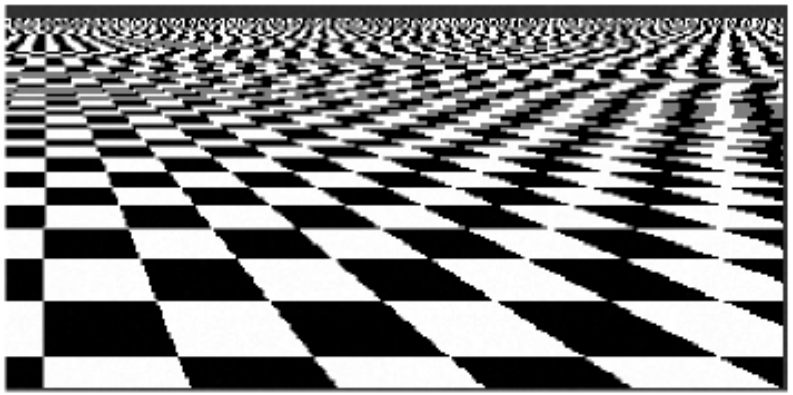
Texture Filtering

- **Mipmapping example**

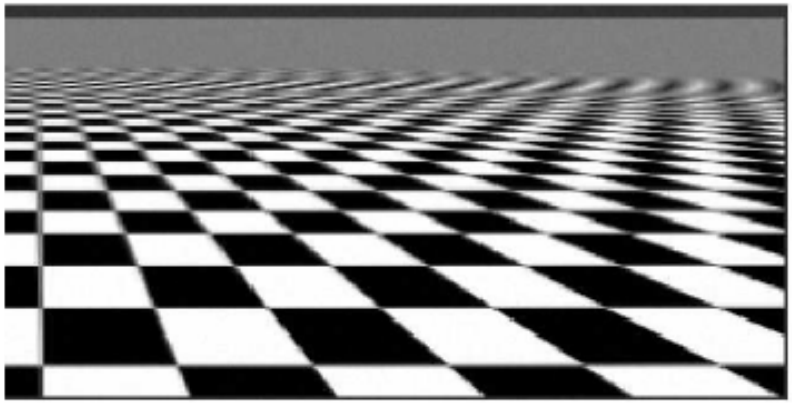


Texture Filtering

- **Anti-Aliasing Using Mip-mapping**



aliasing



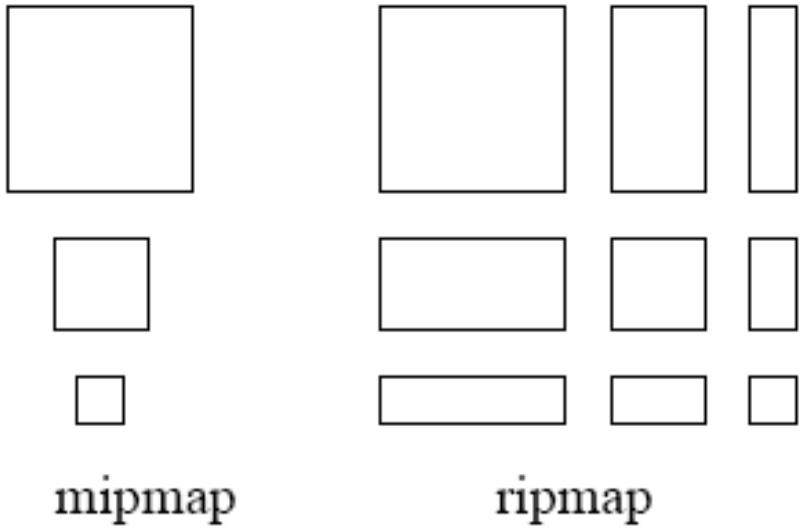
isotropic filtering

Texture Filtering

- **Anisotropic(各向异性) Filtering**
  - **However, one problem with mipmapping is that it can sometimes over-blur, as shown in the figure (last slide). When we build a mipmap, we always filter a higher resolution level isotropically. This can cause over-blurring when the desired filter is anisotropic (when viewed perspective). We will have to enclose it with a big isotropic footprint in order to avoid aliasing. However, this would also cause over blurring in the short axis of the anisotropic footprint.**

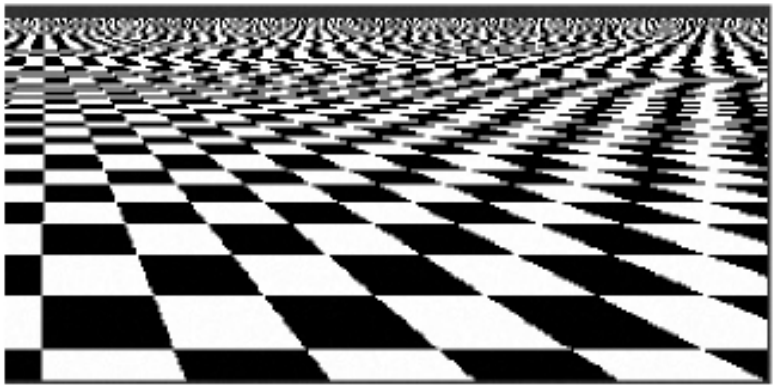
Texture Filtering

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

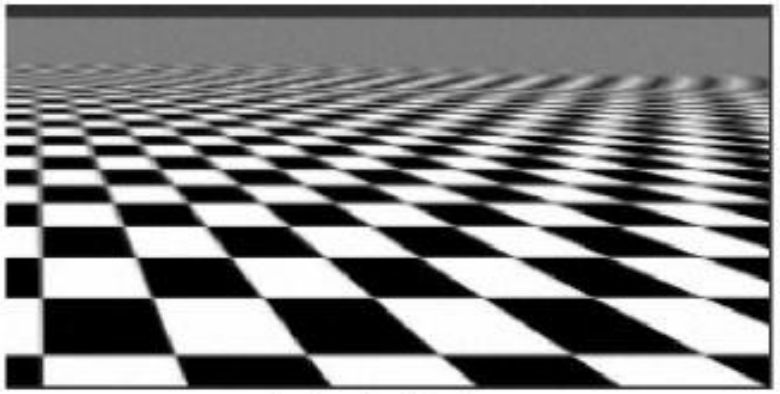


Texture Filtering

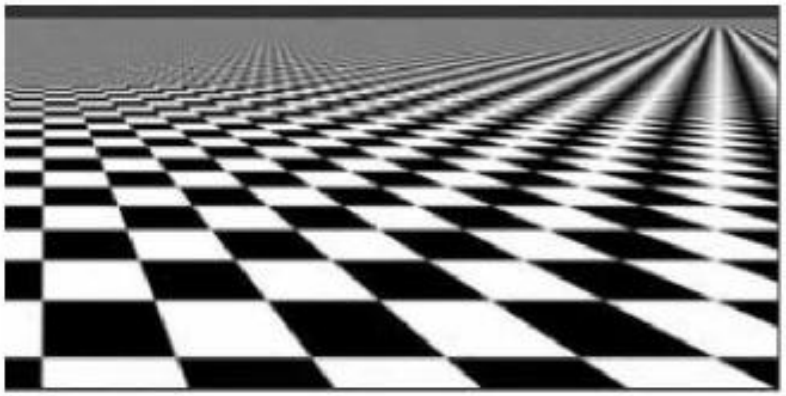
- **Anisotropic(各向异性) Filtering**



aliasing



isotropic filtering



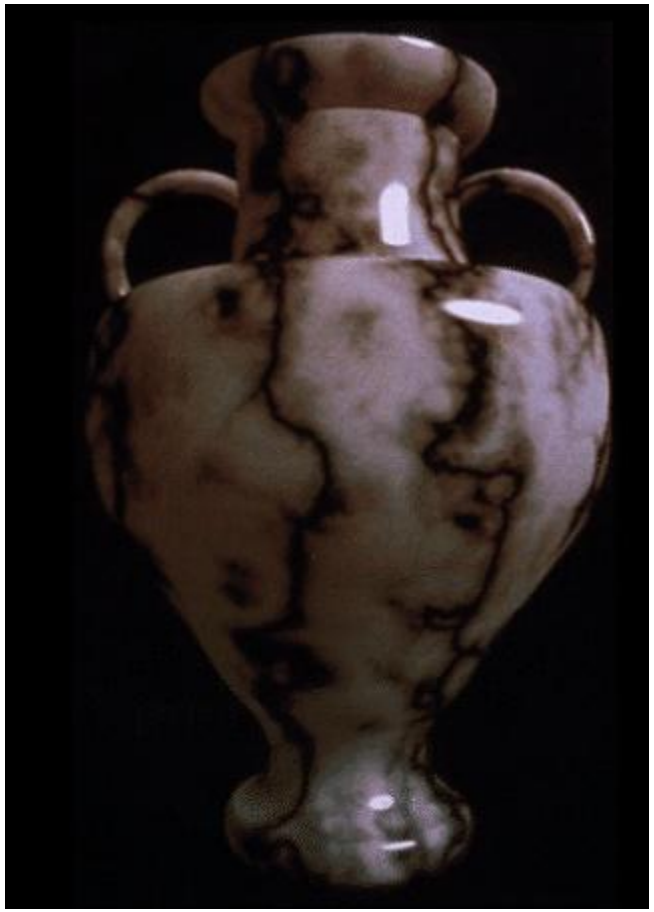
anisotropic filtering

Advanced Topics in Texture

- **Solid Textures**
- **Bump mapping**
- **Displacement mapping**

Advanced Topics in Texture

- **Solid Textures**
  - **Instead of a 2D texture image, we define texture by 3D texture volumes**
  - **creates a 3D parameterization (u,v,w) for the texture mapping onto the object**

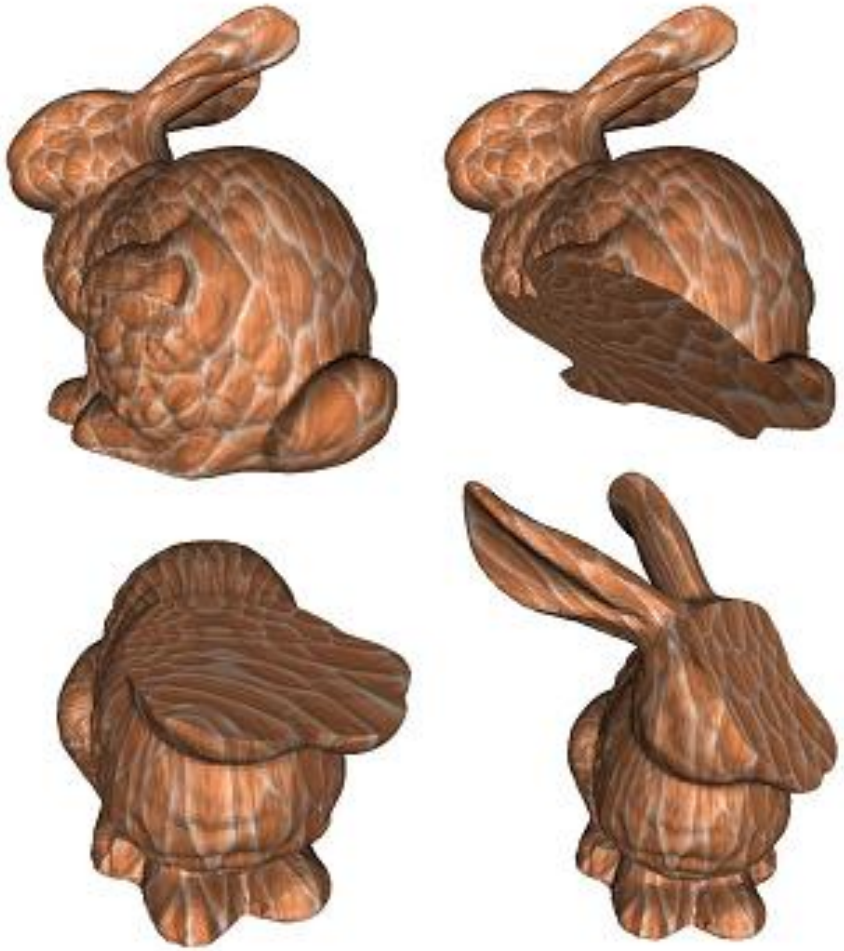


Advanced Topics in Texture

- **Solid Textures**
  - **Have a 3-D array of texture values (e.g., a block of marble).**
    - **Use a function [xyz] -> [RGB] to map colors to points**
  - **Such a 3D texture is called solid texture**
  - **In practice the map is often defined procedurally, no need to store an entire 3D array of colors,**
    - **Just define a function to generate a color for each 3D point**

Advanced Topics in Texture

- **Solid Texture Examples**



Advanced Topics in Texture

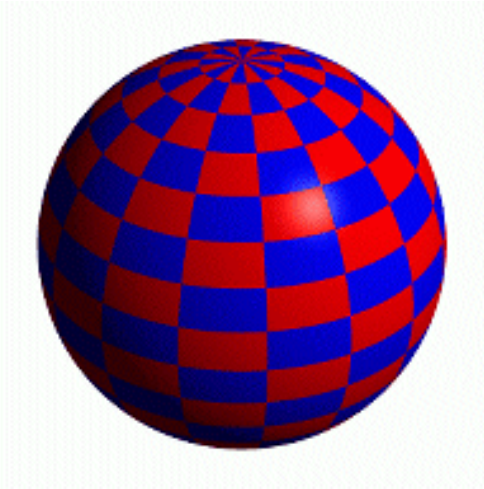
- **Map more than colors**
  - **Basic texture mapping applies colors to surfaces**
    - **but the method is much more general than this**
    - **can be used to map wide range of data onto surfaces**
  - **We will discuss a couple of examples today**
    - **Bump mapping**
    - **Displacement mapping**

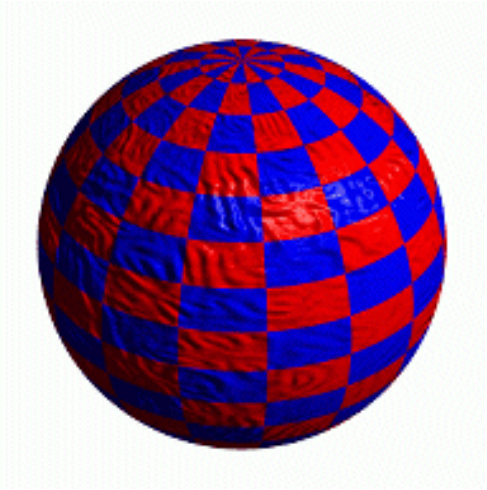
Advanced Topics in Texture

- **Map more than colors**
  - **Using textures to affect theses parameters**
    - **Surface color - common textures**
    - **Surface normal - bump mapping (Blinn 1978)**
    - **Geometry - displacement mapping**
    - **Light source radiance – environment mapping (Blinn 1978)**

Advanced Topics in Texture

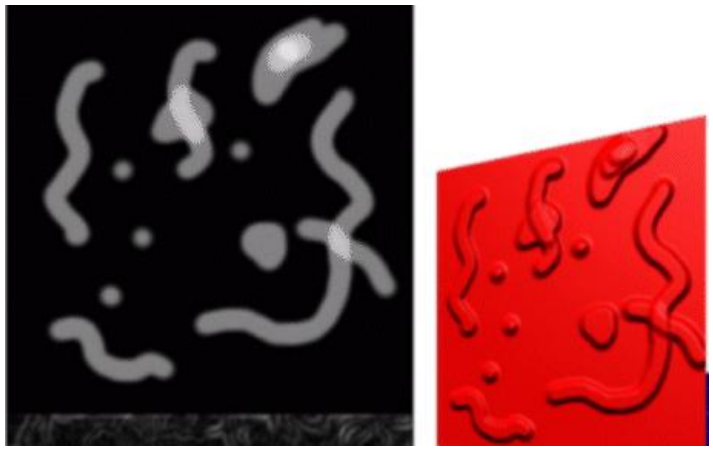
- **Bump Mapping**
  - **Use textures to alter the surface normal**
    - **It does not change the actual shape of the surface**
    - **Just shaded as if it were a different shape**

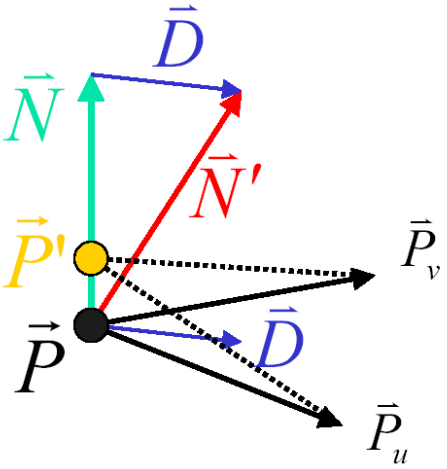




Advanced Topics in Texture

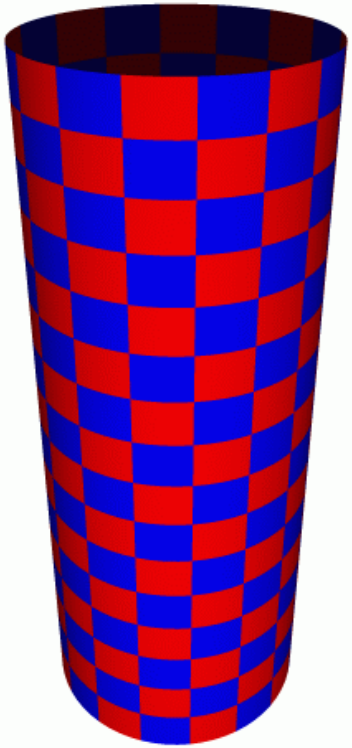
- **Bump Mapping**
  - Treat the texture as a single-valued height function
  - Compute the normal from the partial derivatives in the texture

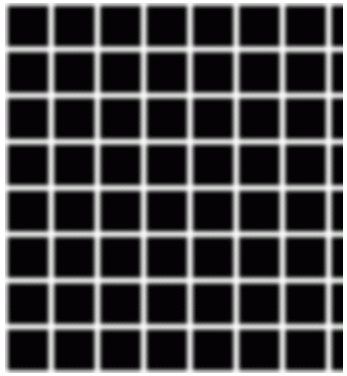




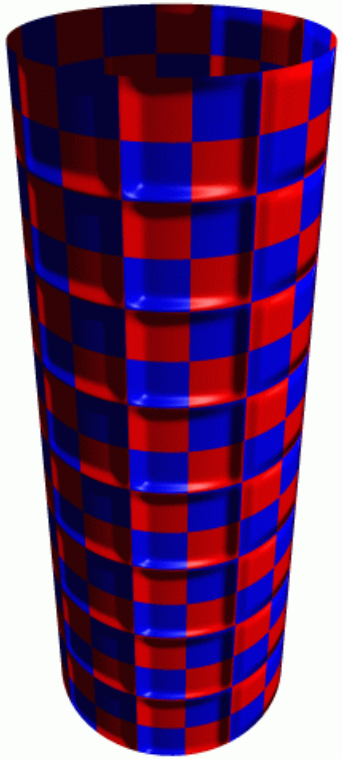
Advanced Topics in Texture

- **Another Bump Mapping Example**



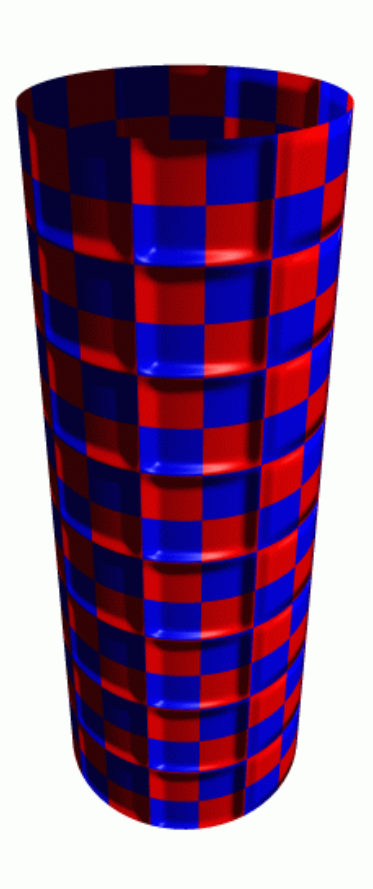


Bump Map



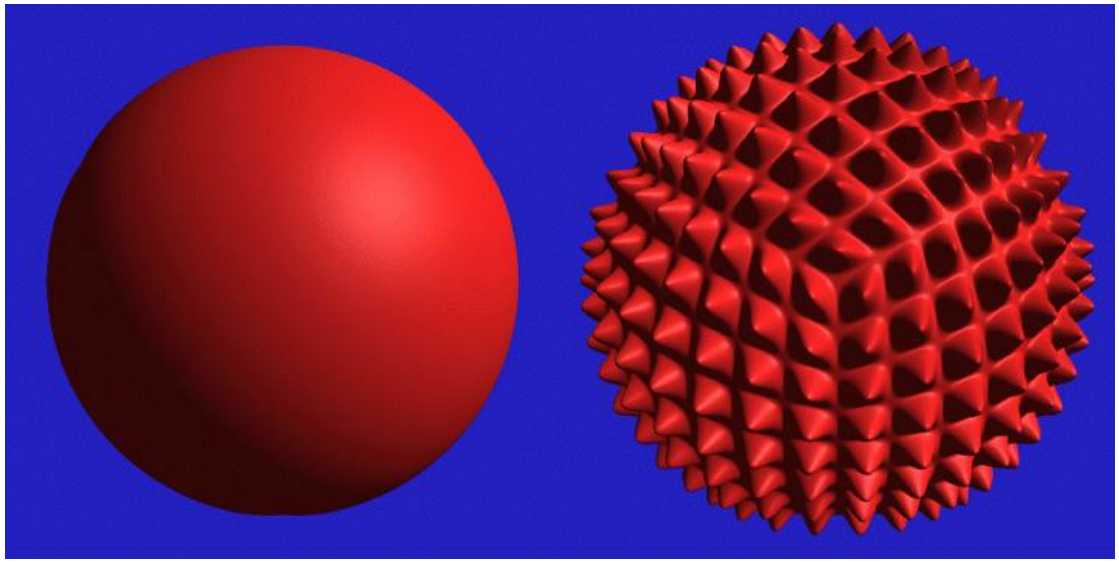
Advanced Topics in Texture

- **Bump Mapping**
  - **However, bump mapping does not allow silhouette(侧面轮廓) effects**
  - **It does not allow self-occlusion or self-shadowing**



Advanced Topics in Texture

- **Displacement Mapping**
  - **Use the texture map to actually move the surface point**
  - **The geometry must be displaced before visibility is determined**
  - **Must be done as preprocess, not in hardware**

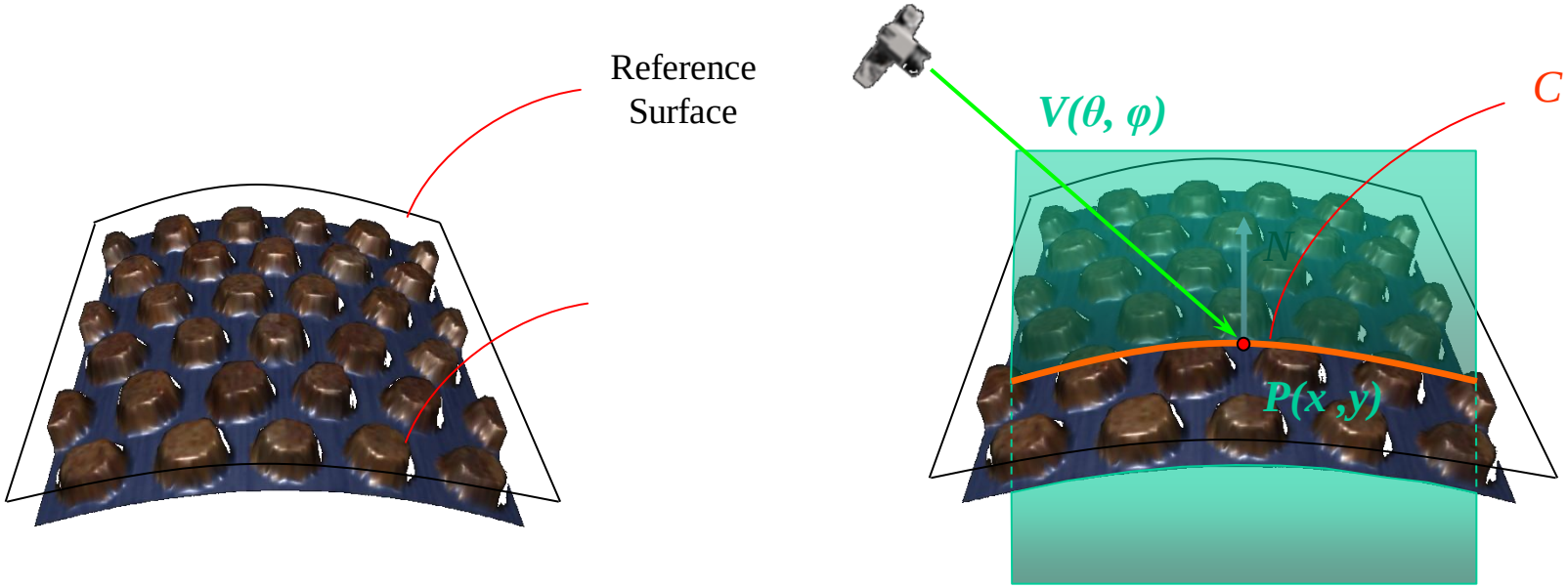


Advanced Topics in Texture

- **View-Dependent Displacement Mapping (VDM), Siggraph 2003, Wang Xi**
  - **Good quality and rich visual effects**
    - **Render all major effects associated with surface mesostructure**
    - **Self-shadow, self-occlusion and silhouette of fine-scale geometry**
  - **Real-time rendering for height field, high frame rate**
  - **Pixel-based processing**

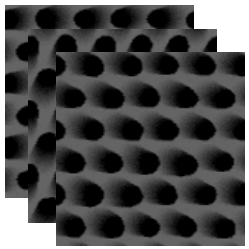
Ideas of VDM

- 5D Function $d_{\mathrm{VDM}}(x, y, \theta, \varphi, C)$

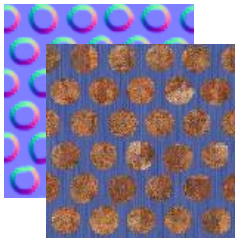


Ideas of VDM

- Through texture mapping transfer VDM to object surface

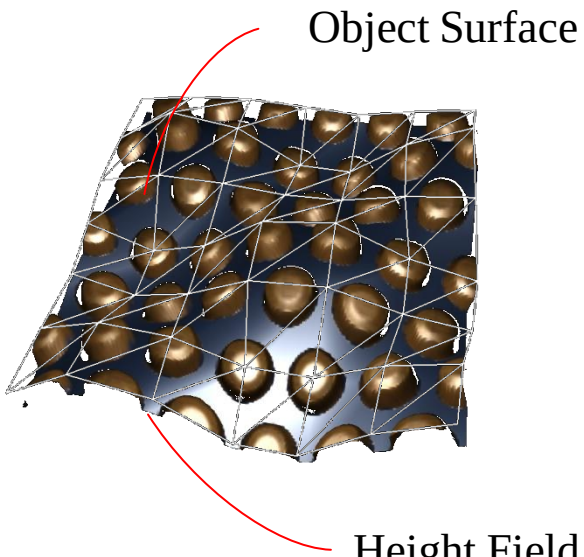


VDM



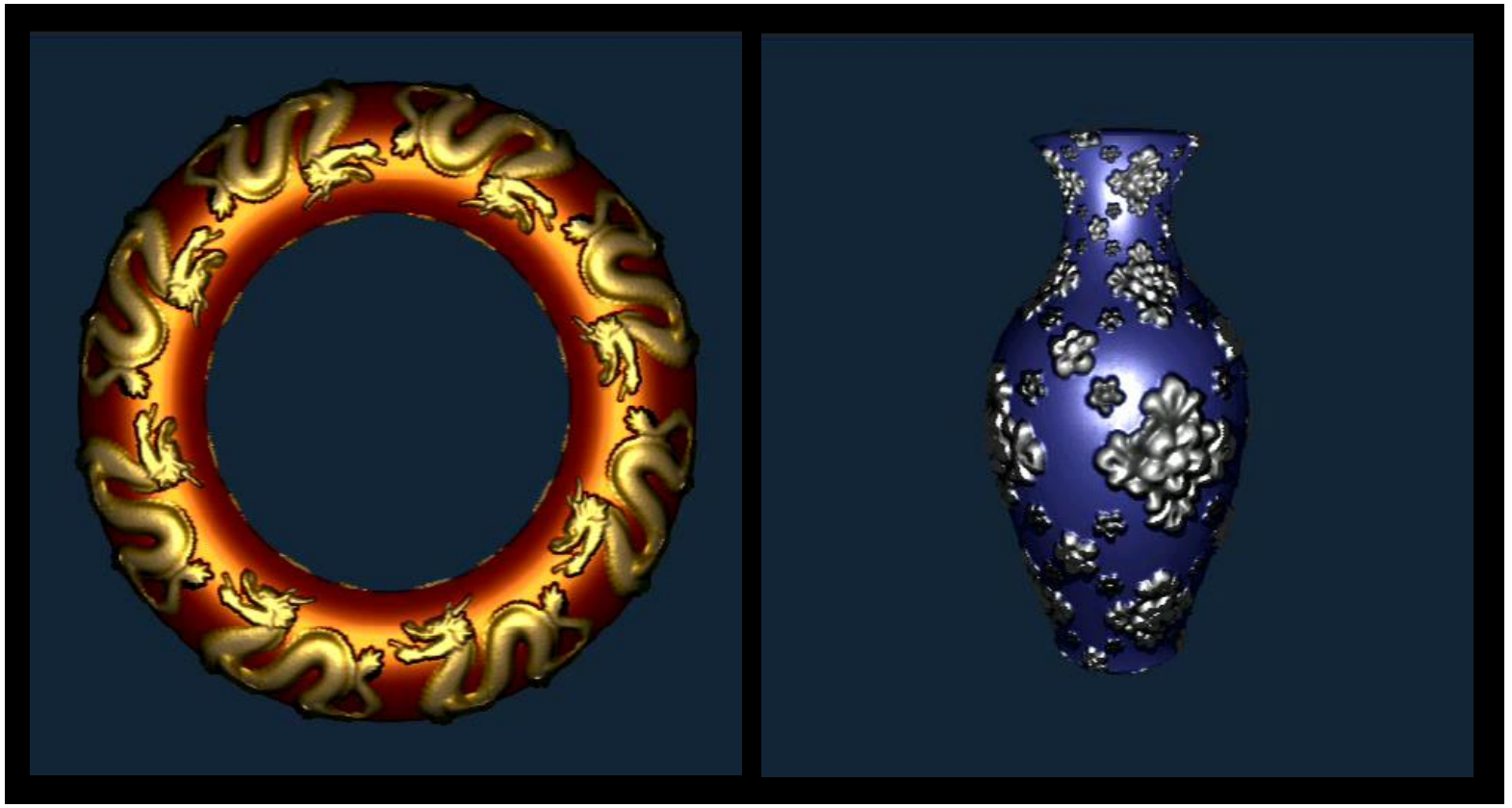
Property Textures

Texture Mapping



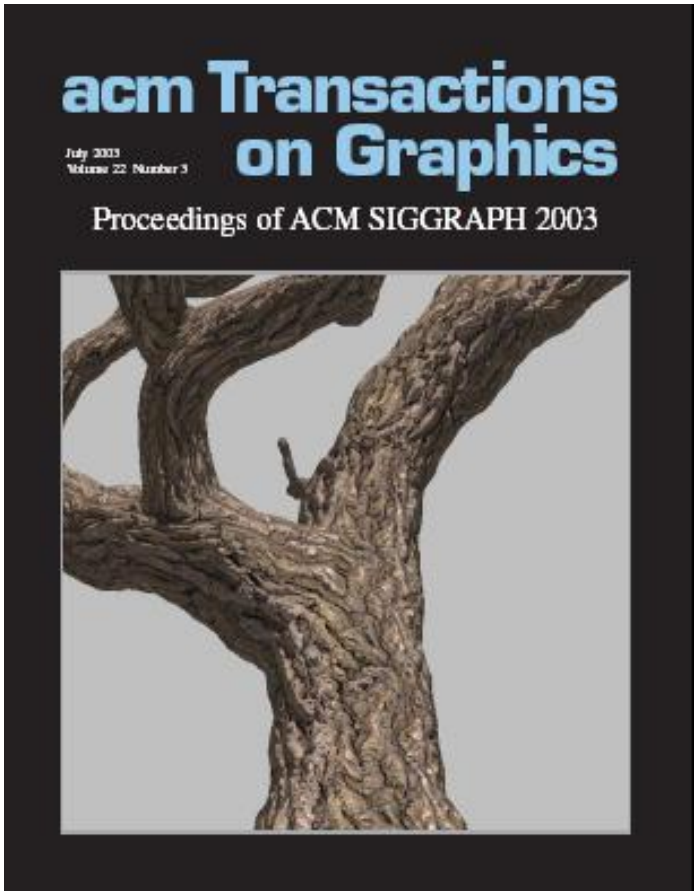
Advanced Topics in Texture

- **View-Dependent Displacement Mapping**



Advanced Topics in Texture

- **View-Dependent Displacement Mapping**



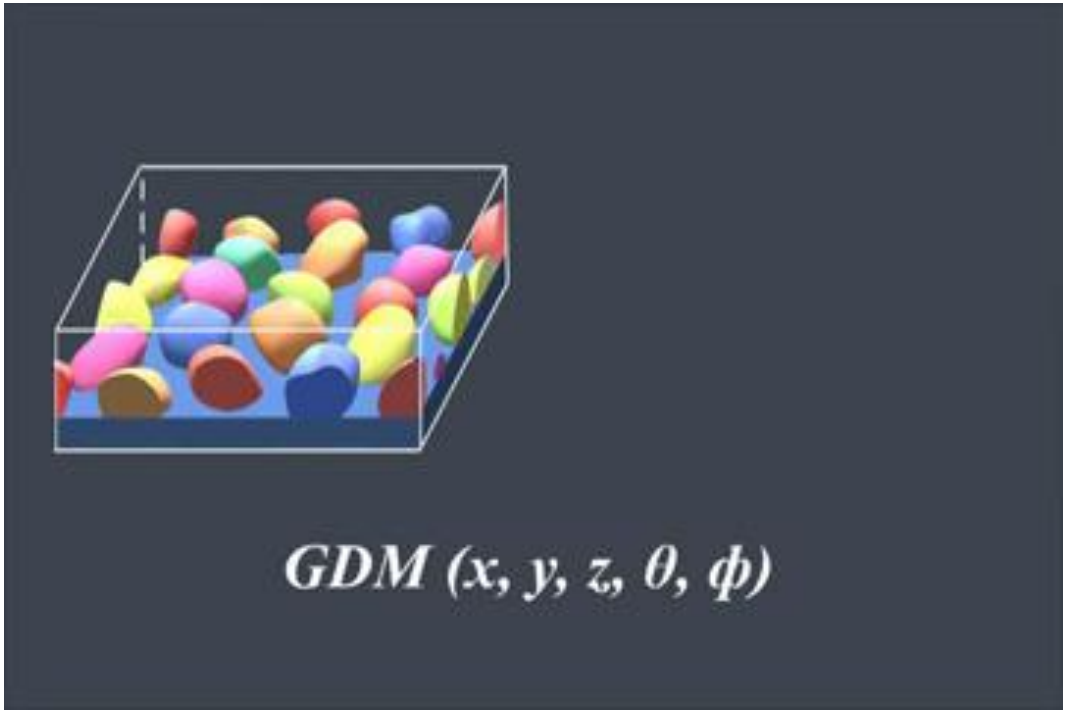
Generalized Displacement Mapping (GDM)

- **Generalized Displacement Mapping (GDM)**
  - **Improve VDM method**
    - **Pure volume texture representation**
    - **Preserve all merits of VDM**
  - **Power of GDM**
    - **NO constrain of mesostructure**
    - **Open surface silhouette**
    - **Robust under texture or surface distortion**

Xi Wang , **Generalized Displacement Maps,** ***Eurographics Symposium on Rendering, 2004***

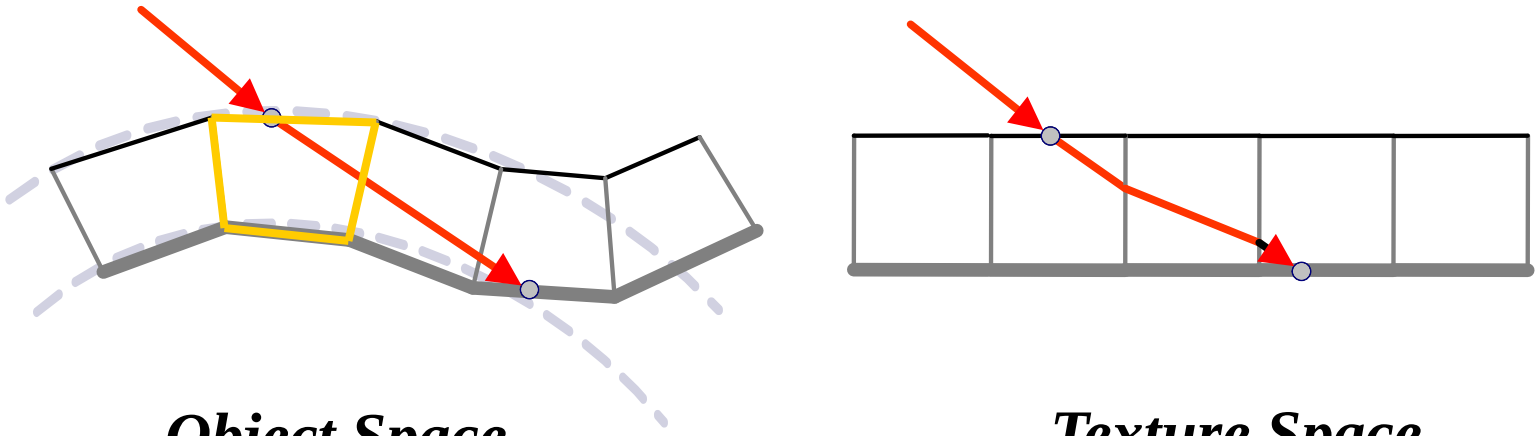
Ideas of GDM

- **GDM Define**



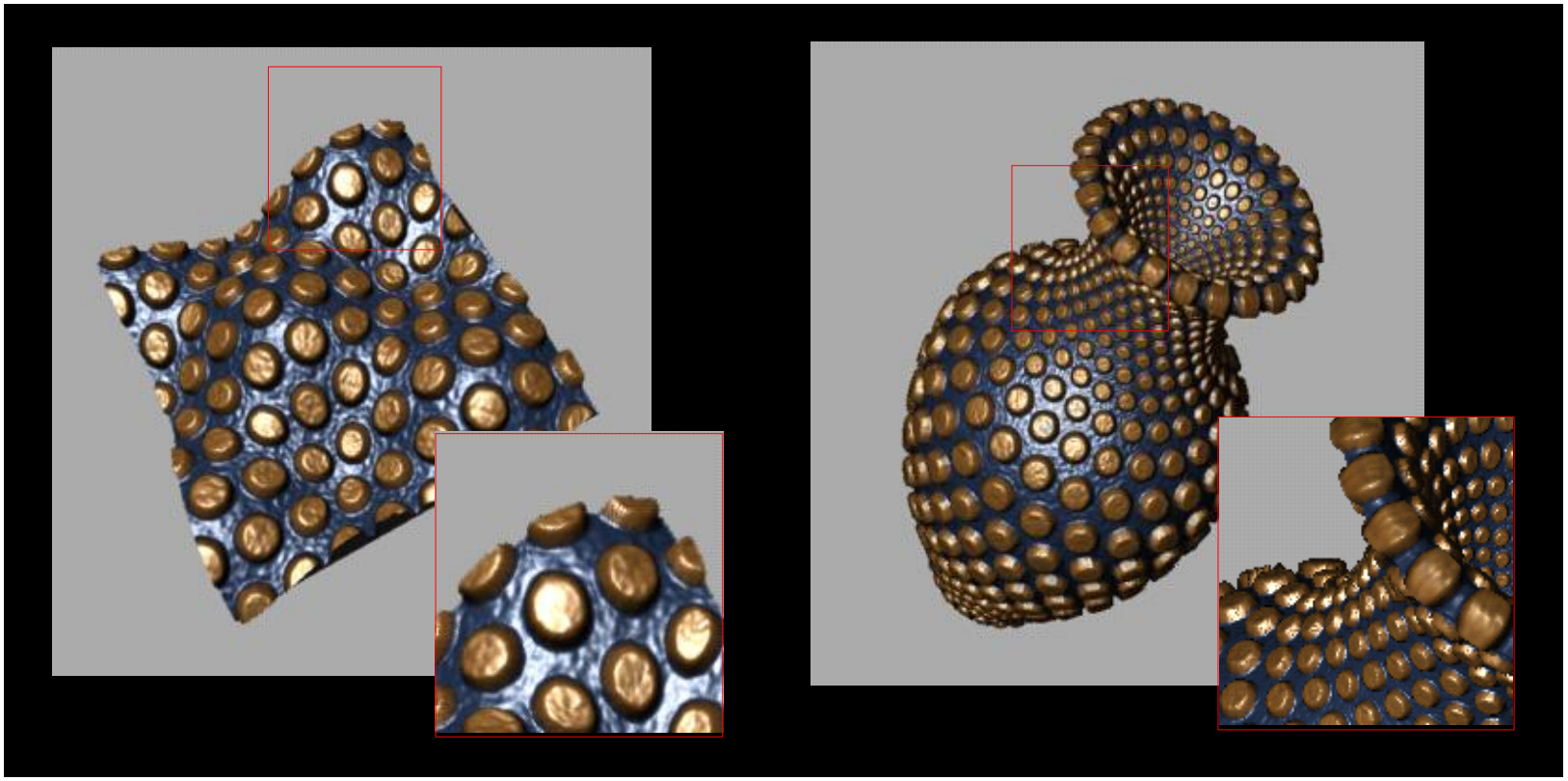
Ideas of GDM

- Basic idea
  - Transfer object space to texture space to get texture information



Object Space ***Texture Space***

Result Related



Result Related

Result Related

Eurograph Workshop on Rendering’04