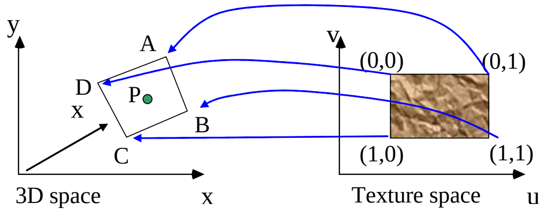
Ray Tracing
Texture Mapping

Kadi Bouatouch
IRISA
Email: kadi@irisa.fr

Texture Mapping

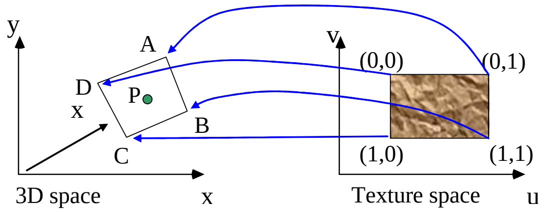
- Texture = 2D Image (we do not consider 3D textures)
- Texture : represented by a 2D array of RGB triplets
- Triplet: Color, Normal or another thing
- Normalized texture space: (u,v), u et v ranging from 0 to 1
- For an intersection point P: compute its texture coordinates either directly or by interpolation



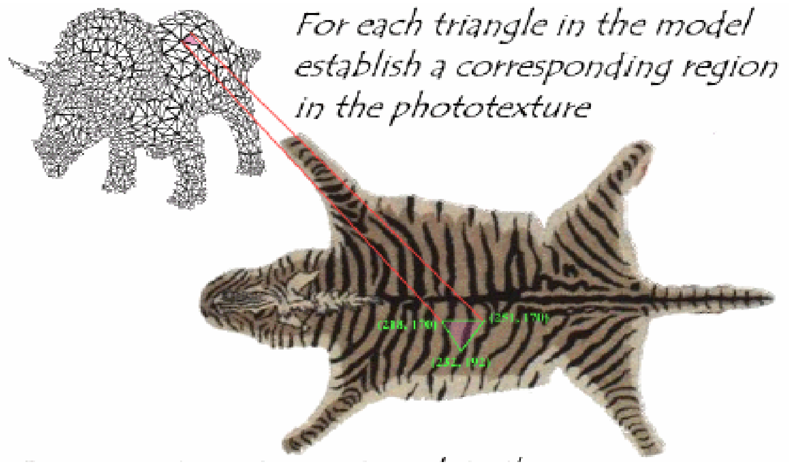
Texture Mapping

Polygon: Triangle or quadrilateral

- Each vertex of the 3D polygon is assigned texture coordinates
- For an intersection point P: compute its texture by bilinear interpolation of the texture coordinates of the polygon's vertices

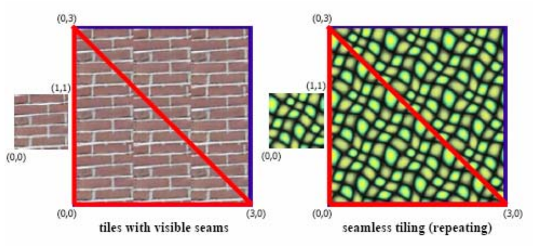


Texture Mapping


For each triangle in the model
establish a corresponding region
in the phototexture

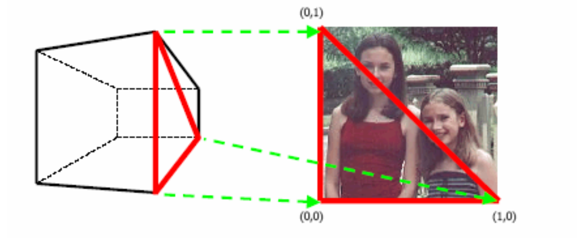
Texture Mapping

- Specify a texture coordinate (u,v) at each vertex
- Canonical texture coordinates (0,0) → (1,1)

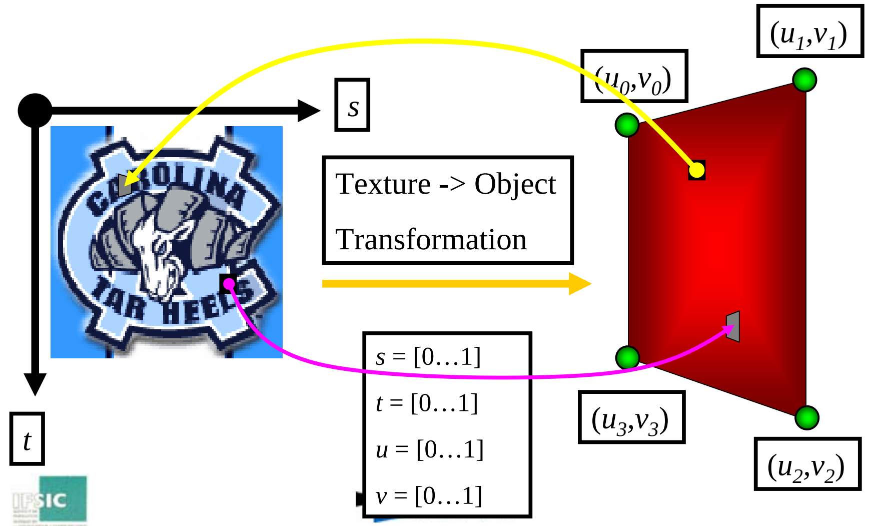


Texture Mapping: Interpolation

- Specify a texture coordinate (u,v) at each vertex
- Interpolate the texture values of intersection points lying on the polygon using those of its vertices



Texture Mapping: Interpolation



Texture Mapping

- Common texture coordinate mapping
  - planar
  - Cylindrical
  - Spherical

Texture Mapping

- Planar: Triangle
  - Barycentric coordinates: a way to parameterize a triangle with P_0, P_1 and P_2 as vertices
  - Let P be a point on the triangle and $\beta_0, \beta_1, \beta_2$ its barycentric coordinates

-Thus: $P = \beta_0 P_0 + \beta_1 P_1 + \beta_2 P_2$

$$P = (1-\beta_1-\beta_2)P_0 + \beta_1 P_1 + \beta_2 P_2$$

Since: $\beta_0 + \beta_1 + \beta_2 = 1$

Texture Mapping

- Planar: Triangle
  - Given an intersection P lying on a triangle
  - Compute its texture coordinates (s,t), say (β_1,β_2) by solving the following linear system:

$$P = (1-\beta_1-\beta_2)P_0 + \beta_1 P_1 + \beta_2 P_2$$

$$P - P_0 = \left(P_2 - P_1\ ,\ P_1 - P_0\right)\begin{pmatrix}\beta_2\\ \beta_1\end{pmatrix}$$

  - Unknowns: β_1 and β_2

Texture Mapping

- Planar: Quadilateral
  - Given an intersection P lying on a triangle

P(u,v)=(1-u)(1-v)P_0 + u(1-v)P1 + uvP_2 + (1-u)vP_3

  - Compute its texture coordinates (s,t) by solving the following linear system (for each coordinate):

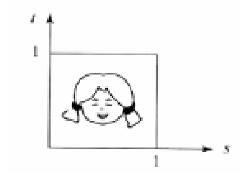
P=(1-u)(1-v)P_0 + u(1-v)P1 + uvP_2 + (1-u)vP_3

  - Unknowns: u and v

Texture Mapping

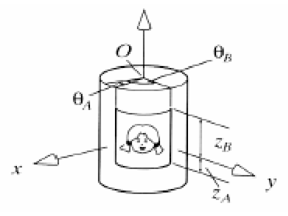
Cylinder

- P(x,y,z): point on the cylinder
- (s,t): texture coordinates



$$\theta \in [\theta_A, \theta_B],\ \ z \in [z_A, z_B]$$

$$x = r\cos\theta,\ y = r\sin\theta,\ z \in [z_A, z_B]$$



$$s = \frac{\theta - \theta_A}{\theta_B - \theta_A} \qquad t = \frac{z - z_A}{z_B - z_A}$$

Texture Mapping

Cylinder

- How to computeTexture coordinates
- Given an intersection P lying on a cylinder
- Compute its texture coordinates (s,t) as:

$$s = (a\cos(x/r) - \theta_A)/(\theta_B - \theta_A)$$
$$t = (z - z_A)/(z_B - z_A)$$

Texture Mapping

Sphere

- P=(x,y,z) on the sphere
- (s,t): texture coordinates

$$x = r\sin\theta\cos\varphi,\ \ y = r\sin\theta\sin\varphi,\ \ z = r\cos\theta$$

$$\theta \in [0,\pi], \varphi \in [0,2\pi] \qquad s = \frac{\theta}{\pi},\ t = \frac{\varphi}{2\pi}$$

Texture Mapping

Sphere

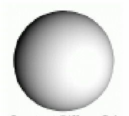
- How to compute Texture coordinates
- Given an intersection P lying on a sphere
- Compute its texture coordinates (s,t) as:

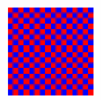
$$s = a\cos(z/r)/\pi$$
$$t = a\cos(\text{x}/(r\sin(\pi\ \text{s})))/2\pi$$

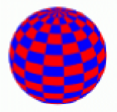
Texture Mapping & Illumination

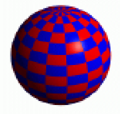
- Texture mapping can be used to alter some or all of the constants in the illumination equation:
  - pixel color, diffuse color, alter the normal, ….
- Classical texturing: diffuse color k_d changes over a surface and is given by a 2D texture which is an image

$$I_{local} = \sum_{i=0}^{nbLum} I_i \times \frac{vis(i)}{d_i^2} \times \left(k_d\left(\vec{N}\cdot\vec{L_i}\right) + k_s\left(\vec{R_i}\cdot\vec{V}\right)^n\right)$$


Constant Diffuse Color

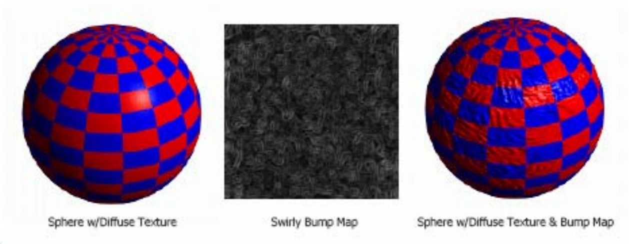

Diffuse Texture Color


Texture used as Label


Texture used as Diffuse Color

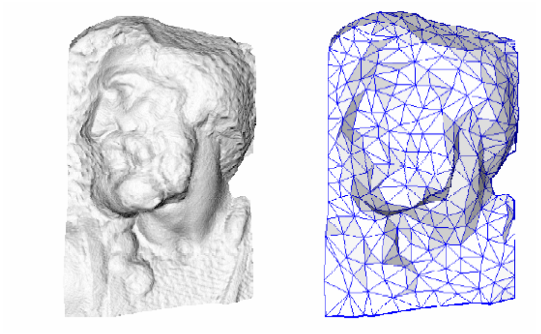
Bum Mapping

- Use textures to alter the surface normal
  - Does not change the actual shape of the surface
  - Just shaded as if it was a different shape

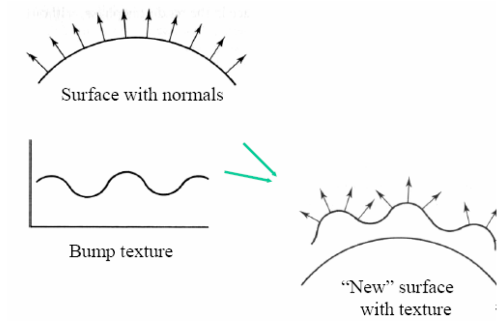


Bum Mapping

- Add more realism to synthetic images without adding a lot of geometry



Bum Mapping



Bum Mapping

- Normal of bumped surface, so-called *perturbed* normal:
- Derivation can be found in "Simulation of Wrinkled Surfaces"

  James F. Blinn

  SIGGRAPH '78 Proceedings, pp. 286-292, 1978

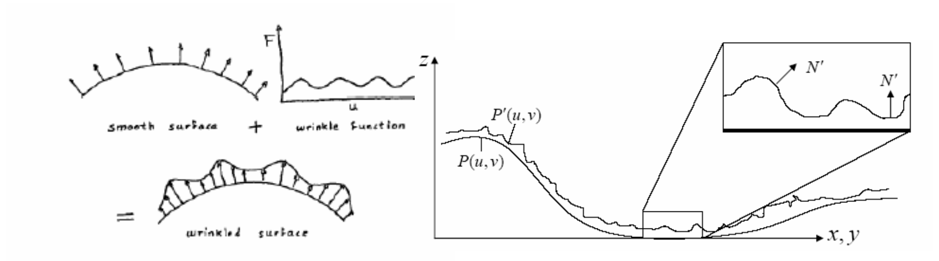
  (Pioneering paper...)
- Use texture to store either:
  - perturbed normal map
  - bump–map itself

Bum Mapping

- The light at each point depends on the normal at that point.
- Take a smooth surface and perturb it with a function B.
- But we don’t really perturb that surface (that is not displacement mapping).
- We modify the normals with the function B(u,v), measuring the displacement of the irregular surface compared to the ideal one.
- we are only shading it as if it was a different shape! This technique is called bump mapping.
- The texture map is treated as a single-valued height function.
- The value of the function is not actually used, just its partial derivatives.

Bum Mapping

The partial derivatives tell how to alter the true surface normal at each point on the surface to make the object appear as if it was deformed by the height function.



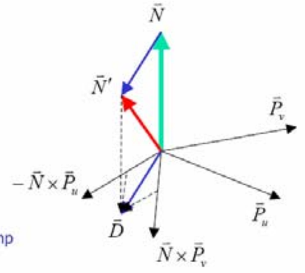
Bump mapping

General case

$$\bar P = [x(u,v), y(u,v), z(u,v)]^T$$ Initial point

$$\bar N = \bar P_u \times \bar P_v$$ Normal

$$\bar P' = \bar P + \frac{B(u,v)\bar N}{\|\bar N\|}$$ Simulated elevated point after bump



$$\bar N' \approx \bar N + \underbrace{\frac{B_u\left(\bar N \times \bar P_v\right) - B_v\left(\bar N \times \bar P_u\right)}{\|\bar N\|}}_{\bar D}$$

Variation of normal in u direction

$$B_u = \frac{B(s-\Delta,t) - B(s+\Delta,t)}{2\Delta}$$

$$B_v = \frac{B(s,t-\Delta) - B(s,t+\Delta)}{2\Delta}$$

Variation of normal in v direction

Compute bump map partials by numerical differentiation

Bump mapping derivation

$$\bar P' = \bar P + \frac{B(u,v)\bar N}{\|\bar N\|}$$

$$\bar P'_u = \bar P_u + \frac{B_u \bar N}{\|\bar N\|} + \overset{\approx 0}{\cancel{\frac{B\bar N_u}{\|\bar N\|}}}$$

Assume B is very small...

$$\bar P'_v = \bar P_v + \frac{B_v \bar N}{\|\bar N\|} + \overset{\approx 0}{\cancel{\frac{B\bar N_v}{\|\bar N\|}}}$$

$$\bar N' = \bar P'_u \times \bar P'_v$$

$$\bar N' \approx \bar P_u \times \bar P_v + \frac{B_u(\bar N \times \bar P_v)}{\|\bar N\|} + \frac{B_v(\bar P_u \times \bar N)}{\|\bar N\|} + \frac{B_u B_v(\bar N \times \bar N)}{\|\bar N\|^2}$$

But $\bar P_u \times \bar P_v = \bar N$, $\bar P_u \times \mathrm{N} = -\bar N \times \bar P_u$ and $\bar N \times \bar N = 0$ so

$$\bar N' \approx \bar N + \frac{B_u(\bar N \times \bar P_v)}{\|\bar N\|} - \frac{B_v(\bar N \times \bar P_u)}{\|\bar N\|}$$

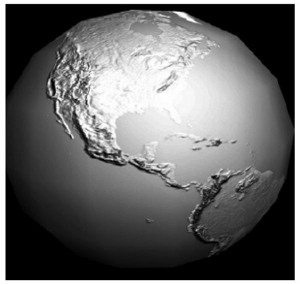
Bum Mapping

Choice of function B(u,v)

- Blinn has proposed various techniques:
- B(u,v) defined analytically as a polynomial with 2 variables or a Fourier serie (very expensive approach)
- B(u,v) defined by 2-entry table (poor results, requires large memory)
- B(u,v) defined by 2-entry table smaller and an interpolation is performed to find in-between values

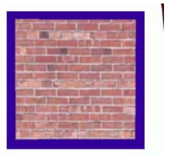
Bum Mapping

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



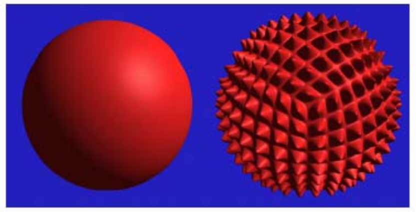
Bump Mapping

- There are no bumps on the silhouette of a
  bump-mapped object
- Bump maps don't allow self-occlusion or self-shadowing
- Problem solved with Displacement Mapping



Displacement Mapping

- Use the texture map to actually move the surface point along the normal to the intersected point.
- The geometry must be displaced before visibility is determined, which is different from bump mapping



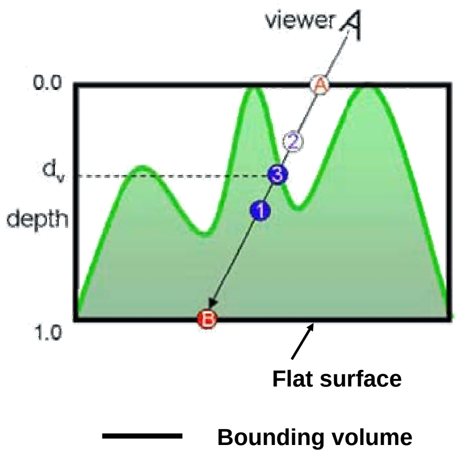
Displacement Mapping

•Compute intersection between ray and bounding volume

•Result: points A and B

•Height (or depth) is stored in a texture

•Use a search technique for the first intersection point: here point 3



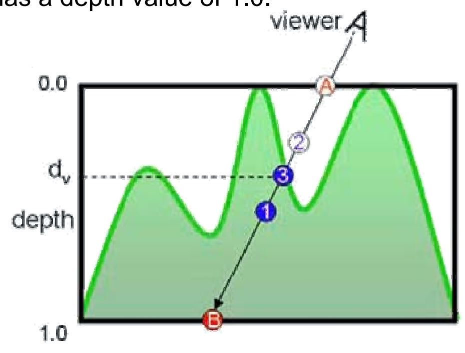
Displacement Mapping

- *A* has a depth vale of 0 and *B* has a depth value of 1.0.

•At each step, compute the midpoint of the current interval and assign it the average depth and texture coordinates of the end points. (used to access the depth map).

•If the stored depth is smaller than the computed value, the point along the ray is inside the height-field surface (point 1).

•In this case it takes three iterations to find the intersection between the height-field and the ray



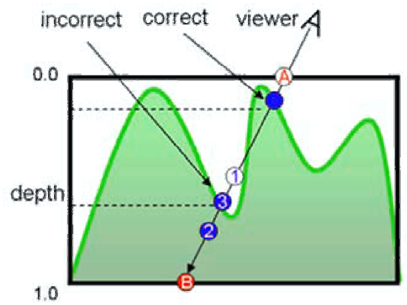
•However, the binary search may lead to incorrect results if the viewing ray intersects the height-field surface more than once

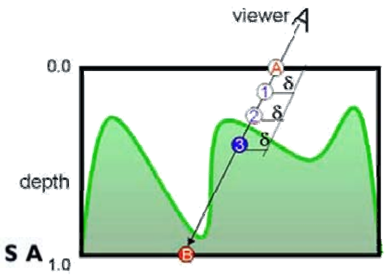
Displacement Mapping

•However, the binary search may lead to incorrect results if the viewing ray intersects the height-field surface more than once:

•In this situation, since the value computed at 1 is less than the value taken from the height-field, the search will continue down.

•In order to remedy this, the algorithm starts with a linear search

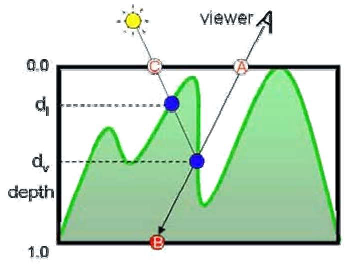




Displacement Mapping

- The technique can also handle surface self-shadowing:

•We must decide if the light ray intersects the height-field surface between point C and the point where the viewing ray first hits the surface.

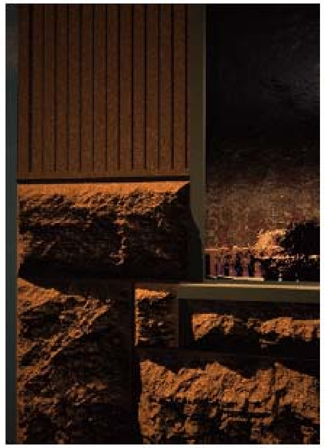


Displacement Mapping

- Image from:

  Geometry Caching for

  Ray-Tracing Displacement Maps
- by Matt Pharr and Pat Hanrahan.
- *note the detailed shadows cast by the stones*



IFSIC

IRISA

Displacement Mapping

- Bump Mapping combined with texture

Schweppes
Black
Cola

IFSIC

IRISA