Einführung in Visual Computing
186.822

Textures

Werner Purgathofer
Surface-Rendering Methods

- polygon rendering methods
- ray tracing
- global illumination
- environment mapping
- texture mapping
- bump mapping
Textures in the Rendering Pipeline

scene objects in object space

transformed vertices in clip space

scene in normalized device coordinates

raster image in pixel coordinates

object capture/creation

modeling

viewing

projection

clipping + homogenization

viewport transformation

rasterization

shading

vertex stage („vertex shader“)

pixel stage („fragment shader“)
Environment Mapping Principle

- reflection mapping
- defined over surface of an enclosing universe (sphere, cube, cylinder)
Environment Mapping Examples

© Haeberli/Segal

© Jim Blinn
Environment Mapping Calculation

- information in the environment map
  - intensity values for light sources
  - sky
  - background objects

- pixel
  - projected onto surface
  - reflected onto environment map
Environment Mapping Calculation

- information in the environment map
  - intensity values for light sources
  - sky
  - background objects

- pixel area
  - projected onto surface
  - reflected onto environment map
Environment Mapping Example
Environment Mapping Filtering

environment maps may be filtered for not so reflective surfaces
define textures on 6 sides of a cube, each parameterized in \((u,v)\)

direction vector \(\mathbf{r}\) starts at \((0,0,0)\)
Environment Mapping with Cube-Map (2)

find \((u,v)\) from the direction vector \(r(x_r,y_r,z_r)\):

if \(x_r > |y_r|\) and \(x_r > |z_r|\) then “front face”

\[
    u = (y_r + x_r)/2x_r
\]

\[
    v = (z_r + x_r)/2x_r
\]

analogous formulas for the other 5 faces

\((-x>|y| \land -x>|z|, \quad y>|x| \land y>|z|, \quad -y>|x| \land -y>|z|, \quad z>|x| \land z>|y|, \quad -z>|x| \land -z>|y|)\)
calculation of the direction vector $\mathbf{r}$:

at a pixel:

viewing direction $\mathbf{v}$ and normal vector $\mathbf{n}$ \Rightarrow

$$\mathbf{r} + \mathbf{v} = (2\mathbf{n} \cdot \mathbf{v})\mathbf{n}$$

$$\mathbf{r} = (2\mathbf{n} \cdot \mathbf{v})\mathbf{n} - \mathbf{v}$$
Environment Mapping Example
most objects do not have smooth surfaces
- brick walls
- gravel roads
- carpets

surface texture required
Adding Surface Detail

- modeling surface detail with polygons
  - small polygon facets
    (e.g., checkerboard squares)
  - facets overlaid on surface polygon (parent)
- parent surface used for visibility calculations
- facets used for illumination calculations
- impractical for complicated surface structure
Texture Mapping: Principle

- texture patterns mapped onto surfaces
  - texture pattern:
    - raster image
    - or procedure (modifies surface intensities)

![Diagram of texture mapping process]

- Texture space
- Texture-surface transformation
- Object space
- Viewing & projection transformation
- Image space
Texture Mapping: Transformation

- texture mapping
  - texture scanning \((u,v) \rightarrow (x,y)\)
  - inverse scanning \((x,y) \rightarrow (u,v)\)

**Texture space:** (\(u,v\)) array coordinates

**Object space:** \((u^*,v^*)\) surface parameters

**Image space:** \((x,y)\) pixel coordinates

**Texture-surface transformation**

\[
\begin{align*}
    u^* &= u^*(u,v) = a_{u^*}u + b_{u^*}v + c_{u^*} \\
    v^* &= v^*(u,v) = a_{v^*}u + b_{v^*}v + c_{v^*}
\end{align*}
\]
projecting pixel areas to texture space = inverse scanning \((x,y) \rightarrow (u,v)\)

- calculation of \(M_{VP}^{-1} M_T^{-1}\)
- anti-aliasing with filter operations
Texture-Mapping: Cylindrical Surface

- $M_{VP}$
  - $u^* = \theta$, with $0 \leq \theta \leq \pi/2$
  - $v^* = z$, with $0 \leq z \leq h$

  - $x = r \cdot \cos u^*$
  - $y = r \cdot \sin u^*$
  - $z = v^*$

- $M^{-1}_{VP}$
  - pixel $\rightarrow$ surface point $(x,y,z)$
  - $(x,y,z) \rightarrow (u^*,v^*)$: $u^* = \cos^{-1}(x/r)$, $v^* = z$
Texture-Mapping: Cylindrical Surface

- \( M_T \) \[ u^* = u \cdot \pi / 2, \quad v^* = v \cdot h \]

- \( M^{-1}_T \) \[ u = 2u^* / \pi, \quad v = v^* / h \]
  \[ (u^* = \cos^{-1}(x/r), \quad v^* = z) \]

\[ u = \cos^{-1}(x/r) / \pi, \quad v = z / h \]
Texture Mapping: Anti-aliasing

- anti-aliasing with filter operations
  - project pixel area into texture space and take average texture value

- speed ups:
  - mip-mapping
  - summed-area table method
Perspective Correct Textures on Triangles

- mapping a texture on a triangle with barycentric coordinates:
  \[ p(\alpha,\beta,\gamma) = \alpha \cdot a + \beta \cdot b + \gamma \cdot c \]
  \[ \text{color}(x,y) = \alpha \cdot t_0 + \beta \cdot t_1 + \gamma \cdot t_2 \]

- fails after perspective transform!

![Diagram showing the mapping of textures on triangles before and after perspective transform.](image)
solution: keep homogeneous weights $h_0, h_1, h_2$ of $a, b, c$ and correct the barycentric values

\[
\begin{align*}
    d &= h_1 h_2 + h_2 \beta (h_0 - h_1) + h_1 \gamma (h_0 - h_2) \\
    \beta_w &= h_0 h_2 \beta / d \\
    \gamma_w &= h_0 h_1 \gamma / d \\
    \alpha_w &= 1 - \beta_w - \gamma_w \\
    u &= \alpha_w u_0 + \beta_w u_1 + \gamma_w u_2 \\
    v &= \alpha_w v_0 + \beta_w v_1 + \gamma_w v_2 \\
    \text{color}(x,y) &= t(u,v)
\end{align*}
\]

instead of

\[
\begin{align*}
    u &= \alpha u_0 + \beta u_1 + \gamma u_2 \\
    v &= \alpha v_0 + \beta v_1 + \gamma v_2 \\
    \text{color}(x,y) &= t(u,v)
\end{align*}
\]
Solid Texturing

- texture defined in 3D
- every position in space has a color
- coherent textures across corners
Solid Texturing Examples

examples for application of 3D textures on a scull and a face

© Univ.Swansea

© Univ.Swansea
Procedural Texturing

- procedural texture definition
  - texture-function $t(x,y,z)$ returns intensity
  - avoid $M_T$
- 2D (surface texturing) or 3D (solid texturing)
- stochastic variations (noise function)
- examples
  - wood grains
  - marble
  - foam
Bump Mapping Principle

bumps are visible because of shading

modeling of bumps is very costly.

*trick: insert a detail structure* $b$:

\[ b \]
Bump Mapping Examples
Reminder: Derivation of a Function

The derivation of a function in one variable is the slope of the function. The formula for the derivative is:

\[ f'(x) = \frac{df(x)}{dx} = \frac{df}{dx} \]
Reminder: Partial Derivation of a Function

Partial derivations of a function in two variables are the slopes of the functions when you keep one of the variables fixed, they are *slopes of tangents*.
Bump Mapping Calculation

- surface roughness simulated
- perturbation function varies surface normal locally
- bump map \( b(u,v) \)

\[
p(u,v) \quad \text{surface point}
\]

\[
\mathbf{n}^* = \mathbf{p}_u \times \mathbf{p}_v \quad \mathbf{n} = \mathbf{n}^*/|\mathbf{n}^*| \quad \text{surface normal}
\]

\[
p'(u,v) = p(u,v) + b(u,v) \cdot n
\]

= modified surface point
Bump Mapping Calculation

\[ p'(u,v) = p(u,v) + b(u,v) \cdot n \]

\[ n' = (p'_u \times p'_v) \]

\[ p'_u = \frac{\partial (p + bn)}{\partial u} = p_u + b_u n + b n_u \]

\[ p'_u \approx p_u + b_u n \quad p'_v \approx p_v + b_v n \]

because \( b \) is very small

\[ n' = (p_u + b_u n) \times (p_v + b_v n) \]

\[ n' = p_u \times p_v + b_v (p_u \times n) + b_u (n \times p_v) + b_u b_v (n \times n) \]

\[ n \times n = 0 \]

\[ n' = n^* + b_v (p_u \times n) + b_u (n \times p_v) \]
Bump Mapping Representation

- bump map \( b(\mathbf{u}, \mathbf{v}) \) defined as raster image
- \( b_u, b_v \): approximated with finite differences
Bump Mapping Problems

- sources of error
  - distortions at grazing angles
  - wrong silhouette (geometry is not changed!)
  - wrong shadows
  - missing bump shadows
  - light effects on back side
Bump Mapping: Grazing Angles

red buttons appear too flat, although they are shaded in 3D
Bump Mapping Problems

- sources of error
  - distortions at grazing angles
  - wrong silhouette (geometry is not changed!)
  - wrong shadows
  - missing bump shadows
  - light effects on back side
Bump Mapping: Wrong Silhouette

bump mapping

Correct geometry
Bump Mapping Problems

- sources of error
  - distortions at grazing angles
  - wrong silhouette (geometry is not changed!)
  - wrong shadows
  - missing bump shadows
  - light effects on back side
Bump Mapping: Missing Bump Shadows

bump mapping

correct shadows
Bump Mapping Problems

- sources of error
  - distortions at grazing angles
  - wrong silhouette (geometry is not changed!)
  - wrong shadows
  - missing bump shadows
  - light effects on back side
Bump Mapping: Back Side Light Effects
Bump Mapping Problems

- sources of error
  - distortions at grazing angles
  - wrong silhouette (geometry is not changed!)
  - wrong shadows
  - missing bump shadows
  - light effects on back side
- there exist special algorithms to repair each error
Displacement Mapping

- “correct version of bump mapping”
- surface points are moved from their original position
- outline of object changes
- much harder to implement than bump mapping
  - rare in practice
- latest hardware partially supports it
Multitexturing: Combination of Mappings

- 2 or more textures applied to a surface
- examples:
  - texture + dirt
  - texture + light map
  - texture + bump map
  - photo + annotations
  - ...

```
+ =
```

![Image of wood texture, light map, and combined image]
Multitexturing: Combination of Mappings

bump mapping
& environment mapping
& texture mapping