Realtime Computer Graphics on GPUs

Effects I

Jan Kolomazník

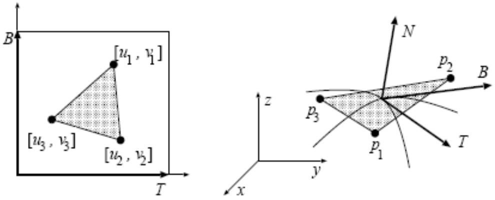
Department of Software and Computer Science Education
Faculty of Mathematics and Physics
Charles University in Prague

Computer Graphics Charles University

Surface Details

Tangent Space

- Local coordinate space
  - Z axis – normal N
  - X axis – tangent T (direction in which u coordinate changes)
  - Y axis – bitangent B (direction in which v coordinate changes)
- TBN matrix – transformation local tangent space to world space
- Orthonormal in texture space
- In general not othonormal in world space (only in special cases)



texture space local modeling space

Tangent Space – Mesh Preprocessing

- Tangent space computed for each vertex
  - Triangle P_1, P_2, P_3, relative coordinates Q_2, Q_3, relative tex. coordinates $[s_2, t_2], [s_3, t_3]$

$$Q_i = P_i - P_1$$

$$[s_i, t_i] = [u_i - u_1, v_i - v_1]$$

  - Solve:

$$Q_i = s_i T + t_i B$$

  - Average with incident triangles (like normals)
- Approximation by orthonormal space:
  - Easy inverse matrix computation
  - Less data transferred to GPU
  - Passing normal and 4D tangent (w used for handedness determination)

$$\mathbf{B} = T_w(\mathbf{N} \times \mathbf{T})$$

Tangent Space – Mesh Preprocessing

- Tangent space computed for each vertex
  - Triangle P_1, P_2, P_3, relative coordinates Q_2, Q_3, relative tex. coordinates $[s_2, t_2], [s_3, t_3]$

$$Q_i = P_i - P_1$$

$$[s_i, t_i] = [u_i - u_1, v_i - v_1]$$

  - Solve:

$$Q_i = s_i T + t_i B$$

  - Average with incident triangles (like normals)
- Approximation by orthonormal space:
  - Easy inverse matrix computation
  - Less data transferred to GPU
  - Passing normal and 4D tangent (w used for handedness determination)

$$\mathbf{B} = T_w(\mathbf{N} \times \mathbf{T})$$

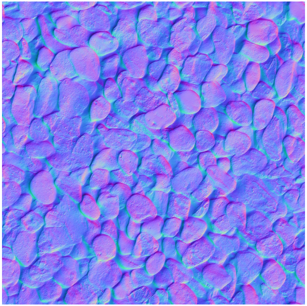
Tangent Space – Computation in Fragment Shader

- Current HW fast enough for on-the-fly computation
- Fast enough to also compute inverse matrix (no need for orthogonalization)
- How to compute differences from position and texture coordinates:

```
vec3 dp1 = dFdx( p );
vec3 dp2 = dFdy( p );
vec2 duv1 = dFdx( uv );
vec2 duv2 = dFdy( uv );
```

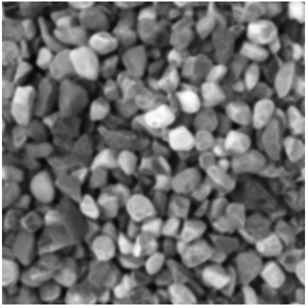
Bump Mapping

- Modulated normals in tangent space – normal map
  - normal $[0, 0, 1]$ mapped to RGB $[1/2, 1/2, 1]$
- Use TBN matrix to transform into world space for lighting computation



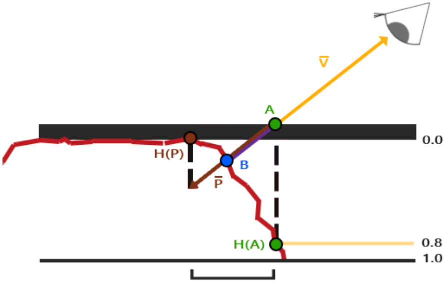
Parallax Mapping

- Bump map – no parallax for surface displacement
- Effect can be simulated by modifying texture coordinates using displacement map



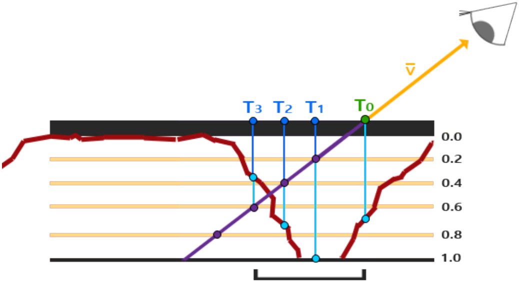
Parallax Mapping – Basic

- Computation in local tangent space
- Scale eye vector into $\mathbf{P}$ by $H(\mathbf{A})$
- Crude estimation of texture offset $\mathbf{P_{xy}}$
- Problematic for steep displacements and low viewing angles



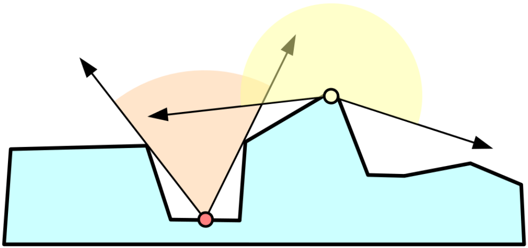
Parallax Mapping – Steep Parallax

- Better estimation of the texture offset
- Check multiple layers to detect intersection more precisely



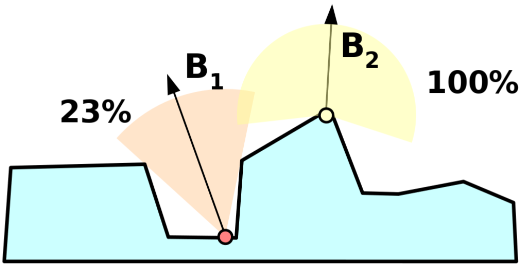
Ambient Occlusion

- constant *ambient term* not good enough
  - does not consider occlusion (even self-occlusion)
  - ridges are equally lighted as valleys
- pre-computed average (potential) contribution of surround light to the surface point



AMBIENT OCCLUSION

- for every surface point compute:
  - percentage of unoccluded rays from an environment (self-occlusion elimination) – *accessibility coefficient*
  - dominant light direction (best lit from) – B
  - technique: ray-casting from each point, counting rays without collision



Accessibility Map Utilization

- accessibility coefficient
  - multiplication factor for ambient light approximation (instead of the k_A constant)
- dominant vector B
  - addressing for the environment light map
  - map should be blurred in advance
  - texture data are multiplied by the accessibility coefficient as well

AO – Occlusion Coefficient

Figure: Phong

Figure: Occlusion coefficient

AO – Average Ray

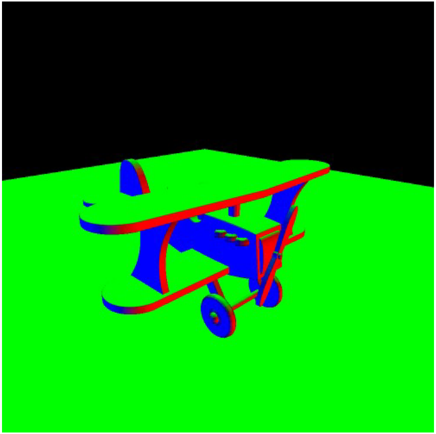


Figure: Normals

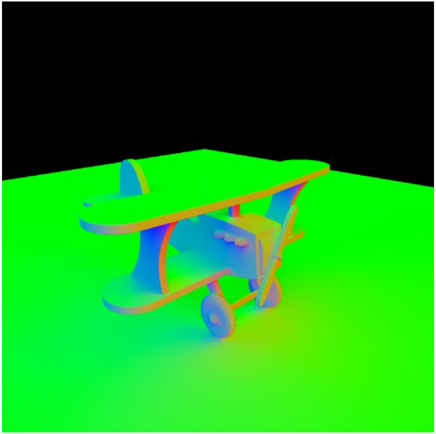


Figure: Average unoccluded rays

AO – Average Ray + Env. Mapping

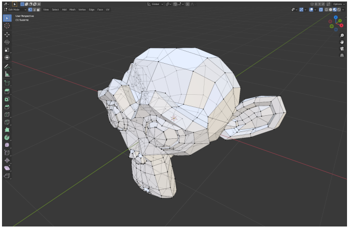
Figure: Phong

Figure: Ambient from env. map

Non-realistic Rendering

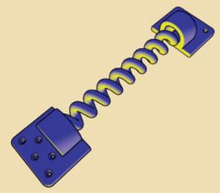
X-Ray Vision

- Highlight invisible objects (occluded by different object)
- CAD system – invisible components
- VR, Games – highlight objects of interest
- Possible approaches:
  - Select occluded objects, render without depth test after everything else
    - Selection by different means
    - Problematic partial occlusion
  - Second render pass for highlighted objects, inverted depth test
    - Works with partial occlusion



Cartoon (Cel) Shading

- goal: results similar to human 2D graphics
  - contour emphasis
  - pen-and-ink drawing simulation (hatching)
  - imitation of painting techniques (oil, watercolor)
  - cartoon-style shading
- approaches (techniques)
  - special textures (coarse shading tones, ..)
  - procedural textures (fragment shader)
  - post-processing (for specific painting techniques)
  - + combinations



Contour Rendering

- No need for explicit definition of contours
- Solids have to be regular (closed)
- Two phases:
  1. front-facing faces only
     - no special rendering style
     - back-face culling
  2. edges of back-facing faces only
     - more thick line (glLineWidth()) – contour lines will stick out
     - alternative – render backfaces of blown-up mesh (no scaling)

Cartoon Light Model

- light model similar to “Blinn-Phong“
  - diffuse term $cos\alpha$
  - optional specular term $cos^h\beta$
- diffuse term indexes simple ramp texture, or quantize the intensity
  - only small number of color tones
  - no texture filtering for sharp outlines
  - CAD applications – determination of plane orientation
- optional specular term with priority
  - thresholding for white-color highlight

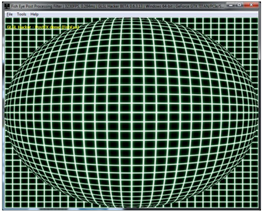
Postprocessing

BASIC POSPROCESSING OPERATORS

- Process outputs from deffered shading stage
- Texture coordinate transformation
- Spatial filtering – operations on pixel (texel) neighborhood
  - Linear filtering – convolution
    - Edge detection
    - Smoothing
    - Bluring
    - Bloom
  - Non-linear:
    - Morphological operations
    - Median filtering

Coordinate Transformation

- Transform input u, v coordinates ($f : [0, 1]^2 \rightarrow [0, 1]^2$)
- Warping
- Optical Effects
  - Fish eye lens
  - Barrel distortions
- Extreme stretching limited by number of texels
- Higher order interpolation – bicubic, …



Spatial Filtering

- Value of the pixel is updated by some function over the neighboring pixels
- Linear combination – convolution
  - Mask containing weights (kernel)
- Nonlinear operations – min, max, median, ...

Spatial Filtering – Implementation

- Fragment shader:
  - u_{step}, v_{step} – single texel offset in normalized texture space
  - texelFetch() – access via non-normalized coordinates
- Compute shader:
  - Better optimization options

Gaussian Smoothing

- Gaussian distribution (normal) – result of combined random processes
- Used for smoothing (bluring), noise reduction
- σ determines kernel radius – 68-95-99.7 rule
- Separable filter:
  - Equivalent to two pass filtering with horizontal and vertical 1D kernel
  - $2n$ instead of n^2 texture reads

Contour (Edge) Detection

- Edges in image – sharp changes in value
- Places with high gradient
- Alternative for cartoon shading

Numerical Differentiation

- Finite difference

$$\frac{f(x+h)-f(x)}{h}$$

- Symmetric difference

$$\frac{f(x+h)-f(x-h)}{2h}$$

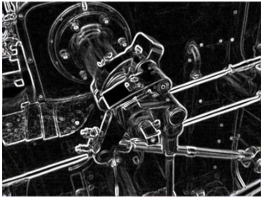
- Higher-order methods – increased numerical stability
- Differentiation increases noise

$$D_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad D_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Sobel Filter

- Numerical partial derivations with small smoothing
- Gradien magnitude – edge strength
- Threshold small values – filter out small fluctuations

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$



Discontinuities in G-buffers

- Z-buffer
  - Boundary between objects
  - Different parts of objects
- Normals
  - Strong edge without normal interpolation
  - Boundary between objects
- ID-buffer (stencil)
  - Boundaries only between different objects

Combined Contours

- Detected discontinuities in normals and depth
- Summ together – all important contours together