Real-time Shading System and Global Illumination

Christian Luksch
Overview

- Scene Content Definition
  - Physically plausible materials
  - Physical light source description

- Putting Everything Together
  - Shading System
  - Solving complexity

- Global Illumination
  - Baking
  - Real-time
Rendering System

Scene
- Geometries
- Materials
- Lights

Global Illumination
- Lightmaps
- Light Probes

Real-time Shading
- IBL
- VCT
- HDR
- GGX
- CSM
- PCSS
- SSS
- SSAO

Camera
## Shader Modules Overview

<table>
<thead>
<tr>
<th>Geometry</th>
<th>Light</th>
<th>Visibility</th>
<th>Filtering</th>
<th>BRDF</th>
<th>Post Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Displacement</td>
<td>Point</td>
<td>Cubemap</td>
<td>PCF</td>
<td>Diffuse</td>
<td>Fog</td>
</tr>
<tr>
<td>Tessellation</td>
<td>Spot</td>
<td>Perspective</td>
<td>Poisson</td>
<td>Phong</td>
<td>Scattering</td>
</tr>
<tr>
<td>Animation</td>
<td>Area</td>
<td>Paraboloid</td>
<td>CSS</td>
<td>Ward</td>
<td>Tonemap</td>
</tr>
<tr>
<td></td>
<td>Directional</td>
<td>Cascaded</td>
<td>PCSS</td>
<td>GGX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Photometric</td>
<td>Imperfect</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Disk</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Visibility</th>
<th>Filtering</th>
<th>Area Light Approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>+Image-based Lighting</td>
<td>BRDF</td>
<td></td>
</tr>
<tr>
<td>+Global Illumination</td>
<td>BRDF</td>
<td></td>
</tr>
</tbody>
</table>
Forward Rendering Setup

Light setup:

- All lights global
  - Culling in shader by light radius
- Assign lights to objects
  - Culling on CPU
  - Expensive uniform updates
  - Should avoid shader switches

-> Uber shader required
Deferred Shading

- Generate G-Buffer with all information required for shading
  - Position, Normal, Material Color, Metalness, Roughness, ...
- For each light:
  - Render light emission hull (depth test for first culling)
  - Reconstruct world position / normal
  - Read material attributes
  - Accumulate light contribution
Deferred Shading

● **Advantages**
  ○ Low Shading Costs: only pixels visible & potentially within light radius
  ○ Less complex shaders
  ○ All information for post-processing

● **Disadvantages / Limitations**
  ○ Transparency
  ○ Antialiasing
  ○ BRDF model restricted
  ○ Memory consumption / bandwidth
# Forward / Deferred Comparison

<table>
<thead>
<tr>
<th></th>
<th>Forward</th>
<th>Deferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex Lighting</td>
<td>-</td>
<td>+ Cheap</td>
</tr>
<tr>
<td>Antialiasing</td>
<td>+ MSAA++</td>
<td>- Post FX AA</td>
</tr>
<tr>
<td>Transparency</td>
<td>o Forward</td>
<td>- Painful</td>
</tr>
<tr>
<td>Material Models</td>
<td>+ Flexible</td>
<td>- Restricted</td>
</tr>
<tr>
<td>Overdraw</td>
<td>+ Early Z</td>
<td>+ Included</td>
</tr>
<tr>
<td>Performance</td>
<td>- Complex Shader</td>
<td>- Memory Hungry</td>
</tr>
</tbody>
</table>
G-Buffer Example Setup

<table>
<thead>
<tr>
<th>RGBA Float16</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth Float32</td>
<td>Depth</td>
</tr>
<tr>
<td>RGBA Float16</td>
<td>Normal XYZ / Roughness</td>
</tr>
<tr>
<td>RGBA Float16</td>
<td>BaseColor / Metalness</td>
</tr>
</tbody>
</table>

- Reconstruction of Position from projective depth
  - Investigate on precision
  - Optimize number of instructions
- Simple representation of Normals
  -> Potential to optimize!

```hlsl
float3 GetFragmentWorldPos(float2 fragcoord)
{
    float depth = tex2D(g_sDepth, fragcoord).r;
    // clip-space coordinate
    float4 vp = float4(fragcoord * float2(2, -2) - float2(1, -1) + float2(-1, 1) / g_RTSsize, depth, 1);
    // transform to world
    float4 v = mul(vp, g_matInvViewProj);
    return v.xyz / v.w;
}
```

HLSL, D3D 9

No fragment center offset in modern OpenGL or DX10+
G-Buffer Optimizations

Keep G-Buffer as small as possible

- Optimization of depth representation
  
  Attack on the Depth Buffer, Matt Pettineo
  https://mynameismjp.wordpress.com/2010/03/22/attack-of-the-depth-buffer/

- Packing of normals
  
  Survey of Efficient Representations for Independent Unit Vectors, JCGT 2014,
  Zina H. Cigolle, Sam Donow, Daniel Evangelakos, Michael Mara, Morgan McGuire, and Quirin Meyer,
  http://jcgt.org/published/0003/02/01/paper.pdf
Transparency

● Render as separate forward pass
  ○ Restricted lighting complexity

● Order-independent transparency
  ○ Possible implementations:
    ■ Depth Peeling
    ■ Pixel Linked List
      http://developer.amd.com/wordpress/media/2013/06/2041_final.pdf
  ○ - High memory consumption / bandwidth
  ○ + Correct transparency
  ○ + Final composition allows for exact blending with volumetric effects based on depth ranges
Multi-Sample Antialiasing

- **No AA**
- **2x MSAA** (4 coverage samples)
- **4x MSAA** (8 coverage samples)
- **8x MSAA** (16 coverage samples)

- **2x EQAA** (4 coverage samples)
- **4x EQAA** (8 coverage samples)
- **8x EQAA** (16 coverage samples)

**Legends:**
- Red Circle: Color Sample Location
- Yellow Circle: Coverage Sample Location
- Blue Rectangle: Pixel Boundary
Multi-Sample Antialiasing

- Fragment shader usually executed only once per pixel
  - Depth/Stencil test performed per sample
  - Color stored per sample

- Using system variables allows per sample evaluation
  - gl_SampleID
  - gl_SamplePosition
  - gl_SampleMaskIn

- MSAA not practical for Deferred Shading
Post Processing Antialiasing

No AA | MLAA | SMAA 1x | SMAA S2x | MSAA 4x | MSAA+MLAA | MSAA+FXAA

[Images of different antialiasing techniques]
Post Processing Antialiasing

- Practical AA for Deferred Shading
- Can be combined with MSAA in general
- Additional Depth input for better reconstruction
- Temporal reprojection and sub-pixel jittering
Post Processing Antialiasing Solutions

- **FXAA: Fast Approximate Antialiasing**
  Code: [https://github.com/NVIDIAGameWorks/GraphicsSamples/blob/master/samples/es3-kepler/FXAA/FXAA3_11.h](https://github.com/NVIDIAGameWorks/GraphicsSamples/blob/master/samples/es3-kepler/FXAA/FXAA3_11.h)

- **MLAA: Morphological Antialiasing**

- **SRAA: Subpixel Reconstruction Antialiasing**

- **SMAA: Enhanced Subpixel Morphological Antialiasing**
  Code: [https://github.com/iryoku/smaa](https://github.com/iryoku/smaa)

- **Temporal Reprojection Antialiasing**
  UE4: [https://de45xmedrsdbp.cloudfront.net/Resources/files/TemporalAA_small-59732822.pdf](https://de45xmedrsdbp.cloudfront.net/Resources/files/TemporalAA_small-59732822.pdf)
  Unity/INSIDE: [https://github.com/playdeadgames/temporal](https://github.com/playdeadgames/temporal)
Going Back Towards to Forward

- Deferred Lighting

1. G-Buffer generation pass of opaque geometry
   - everything required for light calculation (no material information)
2. Deferred lighting pass
   - Generate diffuse and specular illumination
3. Forward render pass
   - Read lighting information from G-Buffer
4. Forward render transparency pass with direct lighting
Deferred Lighting

+ Reduction of shading cost for opaque surfaces compared to forward
+ MSAA
  + Forward pass could be MSAA
  + Upsample G-Buffer illumination

Remaining issues:

- Complex lighting setup for transparent surfaces
- Restricted BRDF model
Clustered Shading / Forward+

1. Z-Pass / Deferred
2. Compute Shader Light Culling
   a. Clustering: Screen Tiles + Depth Tiles
   b. Generate list of visible lights per cluster
3. Forward or Deferred lighting pass using light indices
   
   ● The next level: Tiled Light Trees
      ○ Adaptive tree instead of static tiling
Clustered Shading - Tiled Light Tree

Shaded Scene

Clustered Tiles

Tiled Light Tree

Tiled Light Trees, Yuri O’Donnell and Matthäus Chajdas
Forward+

- All light parameters in global buffer
- Uber Shader
- Allows complex lighting
- Gets around some drawbacks of Deferred Shading
  - No G-Buffer memory overhead
  - MSAA
- Transparency?
Additional Resources

Forward+: Bringing Deferred Lighting to the Next Level,
Takahiro Harada, Jay McKee, and Jason C. Yang, Eurographics 2012
https://takahiroharada.files.wordpress.com/2015/04/forward_plus.pdf

Clustered Deferred and Forward Shading
Ola Olsson, Markus Billeter, and Ulf Assarsson, HPG 2012

Tiled Light Trees,
Yuriy O'Donnell and Matthäus Chajdas, I3D 2017
Conclusion

● **Deferred Shading**
  ○ Solutions for most drawbacks
  ○ Good antialiasing is a challenge
  ○ Order-independent transparency great if suitable
  ○ Long way from first sketch to final system

● **Forward Rendering**
  ○ Lighting complexity can be tackled
  ○ Flexible BRDF
  ○ All antialiasing techniques
  ○ No way around uber shader
How far are we?
How far are we?
Global Illumination
Baking Global Illumination

How to render?
- Path Tracing
- Instant Radiosity
- ...

Where to store?
- Lightmaps
- Light Probes
- Voxels
- ...

Possibility to update?
Lightmaps
UV-Parameterization

a) Use modelling software to generate

b) Simple Approach:
   ○ Build mesh topology
   ○ Grow face clusters with similar normals
   ○ Do not cross border edges
   ○ Start new cluster if no faces can be found

c) Advanced methods overview:
   Mesh Parameterization Methods and Their Applications, Alla Sheffer, Emil Praun and Kenneth Rose, CGV 2006
Lightmap Atlas

- NP-hard problem
- Greedy algorithms
  - Shelf
  - Guillotine
  - Skyline / Horizon

“A thousand ways to back a bin”

- Solutions implemented in modeling programs & tools
Fast Light-Map Computation with Virtual Polygon Lights

Christian Luksch, Robert F. Tobler, Ralf Habel, Michael Schwärzler, Michael Wimmer,
Instant Radiosity
Instant Radiosity
Instant Radiosity
Instant Radiosity
Instant Radiosity
Algorithm Overview

virtual light creation
- photon simulation
- light clustering

light-map rendering
- shadow-map computation
- light accumulation

primary lights

virtual lights
Photon Simulation
Light Clustering
Light Clustering
Light Clustering
Polygon Light Evaluation
Polygon Light Evaluation

1. PCSS visibility test
Polygon Light Evaluation

2. polygon clipping
Polygon Light Evaluation

3. contour integral
Polygon Light Evaluation

= texel illumination
Illumination Error
Illumination Error
Illumination Error
Illumination Error
Illumination Error
Lightmap Rasterization

light-map atlas
Lightmap Rasterization

light-map atlas
Lightmap Rasterization

light-map atlas
uv-Rasterization
uv-Rasterization
uv-Rasterization
uv-Rasterization

\[ p_{10}, p_8, p_6, p_4, p_2 \]
uv-Rasterization
uv-Rasterization
uv-Rasterization
uv-Rasterization
Global Illumination Results
Global Illumination Results
Lightmaps

+ Efficient to use
+ Complex Lighting
  - Parameterization + packing complex
    - Add/Remove during runtime
    - Geometry dependent
    - Prone to visible seams
  - No surface details (e.g. normal maps)
Directional Lightmaps
Directional Lightmaps

Irradiance normal mapped
Directional Lightmaps

Irradiance normal mapped no albedo

Ralf Habel
Directional Lightmaps

- Irradiance -> low frequency signal
- Representation of hemispherical function
  - Half-Life Basis
  - Spherical Harmonics
  - H-Basis

Efficient Irradiance Normal Mapping
Ralf Habel, Michael Wimmer, SIGGRAPH 2010
Directional Lightmaps

- 4 coefficients can represent directional irradiance
  - $2^{nd}$ 4-channel texture layer
  - or 3 layers total with coefficients color channel
- 6 coefficients numerically accurate

Further reading:

Efficient Irradiance Normal Mapping
Ralf Habel, Michael Wimmer, SIGGRAPH 2010
Global Illumination with Light Probes

- Geometry independent representation
- Probes need to represent change of local light
  - Different to probes intended for reflections
- Higher spatial resolution instead of angular resolution
  - Can use similar representations as irradiance lightmaps
Global Illumination with Light Probes

Global Illumination with Light Probes

- Independent from scene geometry
  - Sparse grid / BSP tree search
- Lighting from multiple probes
  - Sample on interpolated probe
  - Or interpolate probes and sample once
- Light leaks problematic
  - Occlusion planes
  - Occlusion shadow map
BRDF Evaluation
BRDF Evaluation

● Want to solve:

\[ L_o(\mathbf{v}) = \int_{\Omega} f(\mathbf{l}, \mathbf{v}) \otimes L_i(\mathbf{l})(\mathbf{n} \cdot \mathbf{l}) d\omega_i \]

● Image-Based Lighting
   ○ \( L_i \) represented by Environment-Map

● Approximate convolution of \( f \) and \( L_i \)
   ○ Difficulty: \( f \) dependent on \( \mathbf{l} \) and \( \mathbf{v} \)
Visualization of BRDF
Real-time IBL Techniques

Single Sample:
- Split Sum Approximation (Unreal Engine 4)

General Approximations:
- Filtered Importance Sampling (Colbert and Krivánek)
- Regular Sampling (Luksch et al.)
  [Link](https://www.cg.tuwien.ac.at/courses/RendEng/2015/RendEng-2016-01-11-GlossyMaterials.pdf)
- Multiple Lobes (Activision)
  [Link](http://c0de517e.blogspot.co.at/2016/07/siggraph-2015-notes-for-approximate.html)
Split Sum Approximation

- Assume microfacet BRDF with roughness and IOR
- Reformulation of IBL rendering equation using importance sampling

\[
\int_{H} L_i(l) f(l, v) \cos \theta_1 dl \approx \frac{1}{N} \sum_{k=1}^{N} \frac{L_i(l_k) f(l_k, v) \cos \theta_{1k}}{p(l_k, v)}
\]

- Split sum into a product of two independent sums
  - First dependent on the image
  - Second dependent on the BRDF

\[
\frac{1}{N} \sum_{k=1}^{N} \frac{L_i(l_k) f(l_k, v) \cos \theta_{1k}}{p(l_k, v)} \approx \left( \frac{1}{N} \sum_{k=1}^{N} L_i(l_k) \right) \left( \frac{1}{N} \sum_{k=1}^{N} \frac{f(l_k, v) \cos \theta_{1k}}{p(l_k, v)} \right)
\]
Split Sum Approximation

● Precompute first sum:

\[
\left( \frac{1}{N} \sum_{k=1}^{N} L_i(1_k) \right) \left( \frac{1}{N} \sum_{k=1}^{N} \frac{f(1_k, \mathbf{v}) \cos \theta_{1_k}}{p(1_k, \mathbf{v})} \right)
\]

○ Assume view from 90° top \( \mathbf{v} = \mathbf{n} \)
○ Sum environment image for each direction \( \mathbf{v} \) by importance sampled vectors \( l_k \)
○ Repeat for different roughness values and store in different mip-levels

● First part can be evaluated by single sample into mip-map level based on roughness
Split Sum Approximation

- Substitute in Schlick Fresnel Approximation into second part

\[
\int_{H} f(1, v) \cos \theta_1 dl = F_0 \int_{H} \frac{f(1, v)}{F(v, h)} \left(1 - (1 - v \cdot h)^5\right) \cos \theta_1 dl + \int_{H} \frac{f(1, v)}{F(v, h)} (1 - v \cdot h)^5 \cos \theta_1 dl
\]

- \(F_0/R_0\) moved out of integral
- Remaining integral depending on roughness and \(\cos \theta_v\)
  - Pre-compute integral into 2D-LUT with scale for \(F_0\) in R and bias in G channel
Split Sum Approximation

Reference (top), Split Sum Approximation (middle), Full Split Sum Approximation with $n = v$ (bottom)
Filtered Importance Sampling

- Use importance transformation of BRDF to sample $L_i$ with a fixed random pattern
- Apply mip-bias based on PDF
Regular Sampling

- Optimally placed regular samples based on BRDF
Regular Sampling
Regular Sampling
Activisions Approach

● Pre-convolve environment map to pyramid of Spherical Gaussians
● Approximate BRDF by a few SG lobes
Activisions Approach

Common Approximation  Multiple Lobes  Reference
BRDF Evaluation with IBL

- Single sample approximation possible
  - Plausible at first look
  - No elongated reflections at glancing angles

- Multi-sample solutions
  - With and without pre-processing
  - Good and flexible approximation
  - Anisotropic BRDFs
Environment Probe Placement
Environment Probe Placement
Environment Probe Placement

- Automatic placement might not necessarily work
  - Allow manual assignment / optimization
- Baked and custom probes
- Additional options for interpolation / blending
- Box shape probes
Real-time Global Illumination Techniques

Instant Radiosity:

- **Reflective Shadow Maps**
  Carsten Dachsbacher, Marc Stamminger, I3D 2005

- **Imperfect Shadow Maps**
  T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, J. Kautz, SIGGRAPH Asia 2008

- **Sequential Monte Carlo Instant Radiosity**
  Peter Hedman, Tero Karras, Jaakko Lehtinen, I3D 2016
  [http://visual.cs.ucl.ac.uk/pubs/smcir/](http://visual.cs.ucl.ac.uk/pubs/smcir/)
Real-time Global Illumination Techniques

Voxel-based:

- **Cascaded Light Propagation Volumes**
  Anton Kaplanyan and Carsten Dachsbacher, I3D 2010
  Video: [https://www.youtube.com/watch?v=vPQ3BbuYVh8](https://www.youtube.com/watch?v=vPQ3BbuYVh8)

- **Voxel Cone Tracing**
  Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, Elmar Eisemann, I3D 2011
Real-time Global Illumination Techniques

Light Field Probes:

● Precomputed Light Field Probes
  Morgan McGuire, Mike Mara, Derek Nowrouzezahrai, David Luebke, I3D 2017

Real-time lightmap updates:

● Precomputed Local Reconstruction from Sparse Radiance Probes
  Ari Silvennoinen and Jaakko Lehtinen, SIGGRAPH Asia 2017
  https://users.aalto.fi/~silvena4/Projects/RTGI/index.html
Have fun coding!

Thank you for your attention!