A Variable-resolution BSDF Implementation

Greg Ward, Anyhere Software
Andrew McNeil, LBNL
Talk Overview

- Need for a variable-resolution representation
- Design considerations & solutions
- WINDOW 6 XML format extensions
- New `genBSDF` options
- An example or two
- Outstanding issues
BSDF Resolution Sensitivity Test for CFS

Is the Klems BSDF resolution enough to determine:

- Lighting energy use (workplane illuminance)
- Glare assessment
BSDF Resolutions

FULL KLEMS
145 PATCHES

2X KLEMS
580 PATCHES

4X KLEMS
2320 PATCHES
Model

FLOOR PLAN

SOUTH FACING ELEVATION
Model

WORK PLANE ILLUMINANCE GRIDS

VIEW POINTS AND DIRECTIONS

VIEW 1

VIEW 2

VIEW 3
Window Systems

SOUTH FACING ELEVATION

OPTICAL LIGHT SHELF

BLACK OUT
Static Simulations

- Ran sunny sky for 3 days (Dec, Mar, June) and 3 times (10:00, 12:00, 14:00)
- Sometimes good, sometimes bad
Best Match
(March, 10:00)

WINDOW OUTPUT
Worst Match
(Dec, 14:00)

WINDOW OUTPUT
Annual Simulations

※ Ran hourly analysis for Phoenix
Lighting Energy Use

<table>
<thead>
<tr>
<th>Zone</th>
<th>Full Klems</th>
<th>2x Klems</th>
<th>4x Klems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zone 1</td>
<td>72%</td>
<td>73%</td>
<td>74%</td>
</tr>
<tr>
<td>Zone 2</td>
<td>20%</td>
<td>21%</td>
<td>23%</td>
</tr>
<tr>
<td>Zone 3</td>
<td>8%</td>
<td>7%</td>
<td>7%</td>
</tr>
</tbody>
</table>
Daylight Glare Index

<table>
<thead>
<tr>
<th></th>
<th>Full Klems</th>
<th>2x Klems</th>
<th>4x Klems</th>
</tr>
</thead>
<tbody>
<tr>
<td>View 1</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>View 2</td>
<td>3%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>View 3</td>
<td>9%</td>
<td>3%</td>
<td>2%</td>
</tr>
</tbody>
</table>
Design Considerations

- Basic: capture peaks, compress smooth regions
- Scale input & output resolutions synchronously
- Require efficient sampling method
- Prefer compact disk/memory representation
- Optimize for isotropic and anisotropic distributions
Quantizing Directions

Altitude-Azimuth methods difficult to subsample
Shirley-Chiu Mapping

Maintains relative areas, important for hemispherical sampling

Cartesian Subdivision

Once we have mapped our directions to rectilinear coordinates, subdivision is straightforward.

Example method: Quadtree
Reason for Scaling Input Resolution with Output

- If we have a peak in a particular output direction, its position will shift in relation to the input direction.

- If we don’t scale resolutions together, we either need to record maximum resolution for all input directions, or deduce and reproduce each input-output relationship.
How It Works:

Take our output direction map:

Layer it for each input direction

Represent as octree

Anisotropic BSDF adds another dimension, making it a hextree

Resolution scales in all dimensions, minimizing footprint
Stratified Sampling in Multiple Dimensions

- Stratification spaces samples more evenly in domain.
- Normally, we would stratify N random variables.
- Coupled dimensions with variable resolution preclude this approach.
- Instead, we use a space-filling curve to traverse dimensions, maximizing neighbor relationships.
- Stratifying SF curve thus stratifies N-D domain.
Start with a probability density function, which we can think of as a 1-dimensional BRDF.

Accumulate densities and normalize to arrive at an invertible distribution.

**Example 1-D Probability Density Function**

Now we just call `rand()` and look up angles.

**Review of Monte Carlo Inversion**

Convert a uniform random variable, $X \in [0,1)$ into a properly distributed value on the sample domain.
MC Inversion in Higher Dimensions

- This gets a little tricky as we add dimensions
- One approach is to divide cumulative distribution into rank-N tensor (e.g., a matrix in 2-D domain)
- This runs into problems with variable resolution
- What if we could transform our N-Dimensional domain back into 1-D?
- Space-filling curves to the rescue!
Hilbert space-filling curves extend to any number of dimensions, maximizing neighbor relationships.

A subvoxel in our tree corresponds to a particular resolution of the Hilbert curve.

\(^\dagger\)H-3 means each dimension divided by \(2^3\)
Benefits of Hilbert Curve

- May be subdivided indefinitely to reach any point in the underlying N-Dimensional space.
- Nearby on 1-D curve implies nearby in other dimensions.
- Although the reverse cannot be said.
- Monte Carlo inversion works as if we had a 1-D PDF.
- We are free to vary function resolution based on PDF.
Variable Resolution Data

High resolution region

Low resolution region (nearly diffuse)

Spike in BSDF

Hilbert curve winds through our 2-D direction space & subdivides each region

Medium resolution region

Wednesday, August 24, 2011
Sampling Steps

1. Project incident vector to circle and map to square
2. Get cumulative table for this 2-D Cartesian position
3. Find nearest entry in cumulative distribution table based on the given random input \([0,1)\)
4. Interpolate the corresponding Hilbert index
5. Convert index to N-Dimensional Cartesian position
6. Map back to circle then to exiting direction vector
Details

- Cumulative tables are cached for efficiency
- Store cumulative distribution + Hilbert index correspondences rather than an inverse MC table
- Takes less space, slightly longer to sample
- Better accuracy & no resolution limit
- Isotropic case proved difficult to debug, but saves memory and time when applicable
/* Basic node structure for variable-resolution BSDF data */
typedef struct SDNode_s {
    short ndim;         /* number of dimensions */
    short log2GR;       /* log(2) of grid resolution (< 0 for tree) */
    union {
        struct SDNode_s *t[1];   /* subtree pointers */
        float v[1];              /* scattering value(s) */
    } u;                  /* subtrees or values (extends struct) */
} SDNode;

That's it.
Compare to BSDF Matrix Structure

/* Rectangular matrix format BSDF */
typedef struct {
    int ninc;    /* number of incoming directions */
    int nout;    /* number of outgoing directions */
    void *ib_priv; /* input basis private data */
    b_vecf *ib_vec; /* get input vector from index */
    b_ndxf *ib_ndx; /* get input index from vector */
    b_ohmf *ib_ohm; /* get input proj. SA for index */
    void *ob_priv; /* output basis private data */
    b_vecf *ob_vec; /* get output vector from index */
    b_ndxf *ob_ndx; /* get output index from vector */
    b_ohmf *ob_ohm; /* get output proj. SA for index */
    float bsdf[1]; /* scattering data (extends struct) */
} SDMat;
WINDOW 6 XML Format

- Added *IncidentDataStructure* types, “TensorTree3” for isotropic and “TensorTree4” for anisotropic data

- Added *AngleBasis* type, “LBNL/Shirley-Chiu”

- Scattering data has curly braces to delineate nodes

- Simplest possible example, perfect diffuser:
  
  ```xml
  <ScatteringData> { 0.3183 } </ScatteringData>
  ```
New \texttt{genBSDF} Options

- It was a lot of new code to add two little options:
  - \texttt{-t3 N} Isotropic BSDF at $2^N$ max. resolution
  - \texttt{-t4 N} Anisotropic BSDF at $2^N$ max. resolution

- Beware of \texttt{N} greater than 6 (64x64x64x64)

- Need better method to reach higher resolution

- The \texttt{-n} option has been improved to provide nearly linear speed-up for tensor tree construction
Simple Example

# A simple mirror

void metal mirror_mat
0 0
5 .8 .8 .8 1 0

mirror_mat polygon mirror
0 0
12
0 0 0
1 0 0
1 1 0
0 1 0
genBSDF -geom meter -t3 4 mirror.rad > mirror.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
    <!-- File produced by: genBSDF -t3 4 -mgf -geom meter mirror.mgf -->
    <WindowElementType System="WindowElement" Optical="Optical">
        <Layer System="Layer" Number="LayerNumber" Width unit="meter" Height unit="meter">
            <SourceSpectrum CIE Illuminant D65. imsp."SourceSpectrum">
                <Detector Spectrum ASTM E308. idsp."DetectorSpectrum">
                    <WavelengthData Direction = "Transmission">
                        <WavelengthDataDirection Transmission = "Transmission">
                            <IncidentDataStructure DeviceType = "DetectorSpectrum" IncidentDataStructure = "DetectorData">
                                <ScatteringDataType Reflection = "Reflection" Direction = "Reflection">
                                    <ScatteringData AngleBasis = "SLNL/Chiu-Chiu" ScatteringDataType = "BTDF"/>
                                </ScatteringData>
                            </IncidentDataStructure>
                            <WavelengthDataDirection Transmission = "Transmission">
                                <IncidentDataStructure DeviceType = "SourceSpectrum" IncidentDataStructure = "DetectorData">
                                    <ScatteringDataType Reflection = "Reflection" Direction = "Reflection">
                                        <ScatteringData AngleBasis = "Shirley-Chiu" ScatteringDataType = "BTDF"/>
                                    </ScatteringData>
                                </IncidentDataStructure>
                            </WavelengthDataDirection>
                        </WavelengthDataDirection>
                        <WavelengthDataDirection Transmission = "Transmission">
                            <IncidentDataStructure DeviceType = "SourceSpectrum" IncidentDataStructure = "DetectorData">
                                <ScatteringDataType Reflection = "Reflection" Direction = "Reflection">
                                    <ScatteringData AngleBasis = "Shirley-Chiu" ScatteringDataType = "BTDF"/>
                                </ScatteringData>
                            </IncidentDataStructure>
                            <WavelengthDataDirection Transmission = "Transmission">
                                <IncidentDataStructure DeviceType = "DetectorSpectrum" IncidentDataStructure = "DetectorData">
                                    <ScatteringDataType Reflection = "Reflection" Direction = "Reflection">
                                        <ScatteringData AngleBasis = "SLNL/Chiu-Chiu" ScatteringDataType = "BTDF"/>
                                    </ScatteringData>
                                </IncidentDataStructure>
                            </WavelengthDataDirection>
                        </WavelengthDataDirection>
                    </WavelengthDataDirection>
                    <WavelengthDataDirection Transmission = "Transmission">
                        <IncidentDataStructure DeviceType = "SourceSpectrum" IncidentDataStructure = "DetectorData">
                            <ScatteringDataType Reflection = "Reflection" Direction = "Reflection">
                                <ScatteringData AngleBasis = "Shirley-Chiu" ScatteringDataType = "BTDF"/>
                            </ScatteringData>
                        </IncidentDataStructure>
                    </WavelengthDataDirection>
                    <WavelengthDataDirection Transmission = "Transmission">
                        <IncidentDataStructure DeviceType = "SourceSpectrum" IncidentDataStructure = "DetectorData">
                            <ScatteringDataType Reflection = "Reflection" Direction = "Reflection">
                                <ScatteringData AngleBasis = "SLNL/Chiu-Chiu" ScatteringDataType = "BTDF"/>
                            </ScatteringData>
                        </IncidentDataStructure>
                    </WavelengthDataDirection>
                    <WavelengthDataDirection Transmission = "Transmission">
                        <IncidentDataStructure DeviceType = "SourceSpectrum" IncidentDataStructure = "DetectorData">
                            <ScatteringDataType Reflection = "Reflection" Direction = "Reflection">
                                <ScatteringData AngleBasis = "SLNL/Chiu-Chiu" ScatteringDataType = "BTDF"/>
                            </ScatteringData>
                        </IncidentDataStructure>
                    </WavelengthDataDirection>
                </Detector Spectrum ASTM E308. idsp."DetectorSpectrum">
            </SourceSpectrum CIE Illuminant D65. imsp."SourceSpectrum">
        </Layer>
    </WindowElement>
</WindowElement>
```

Only 8 non-zero reflectance values corresponding to i/o peaks

```plaintext
{ 0.000000e+00 0.000000e+00 6.8198e+01 0.000000e+00 6.5711e+01 0.000000e+00 0.000000e+00 0.000000e+00 }
```
Example from Previous Talk

Sawtooth material with alternating diffuse & mirror elements
Seeing the Whole BRDF

Each subimage is all the outgoing directions for a specific incident direction
Matrix BSDF

145 incident x 145 exitant directions using Klems coordinates
Low-resolution Tensor Tree

Maximum of 256 incident x 256 exitant directions
High-resolution Tensor Tree

Maximum of 4096 incident x 4096 exitant directions (400K samples per incident vector)
Full-resolution BRDF data

Same BRDF as Tensor Tree, but without simplification
Ground Truth

Mirror material with hundreds of virtual light sources
Proxy mode rendering using BSDF

Variable resolution version -- full res. looks about the same
XML File Sizes

*Klems Matrix file is 538 KB

*Low-resolution Tensor Tree is 110 KB

*High-resolution Tensor Tree is 17.6 MB

*Full-resolution data is 205 MB (16.7 million values)
## Calculation Times

<table>
<thead>
<tr>
<th>Resolution &amp; Type</th>
<th>genBSDF</th>
<th>rpict</th>
</tr>
</thead>
<tbody>
<tr>
<td>145x145 Klems</td>
<td>6 minutes</td>
<td>23 minutes</td>
</tr>
<tr>
<td>16x16 Tensor Tree</td>
<td>6 minutes</td>
<td>21 minutes</td>
</tr>
<tr>
<td>4Kx4K Tensor Tree</td>
<td>30 days</td>
<td>21 minutes</td>
</tr>
<tr>
<td>4Kx4K Full-res.</td>
<td>30 days</td>
<td>25 minutes</td>
</tr>
</tbody>
</table>
Outstanding Issues

• Higher-resolution BSDFs don’t always translate to better-looking results

• Difficult to sample highly directional indirect

• \textit{mkillum} can be used in CFS cases

• Can we use GPU to accelerate \texttt{genBSDF}?

• How best to reduce measured BSDF data?

• WINDOW 6 support?
Acknowledgements

- Doug Moore, Rice University for Hilbert curve code
- Peter Shirley, U. of Utah for disk↔square code
- Ian Ashdown, byHeart Software for porting work