A Variable-resolution BSDF Implementation

Greg Ward, Anyhere Software
Andrew McNeil, LBNL

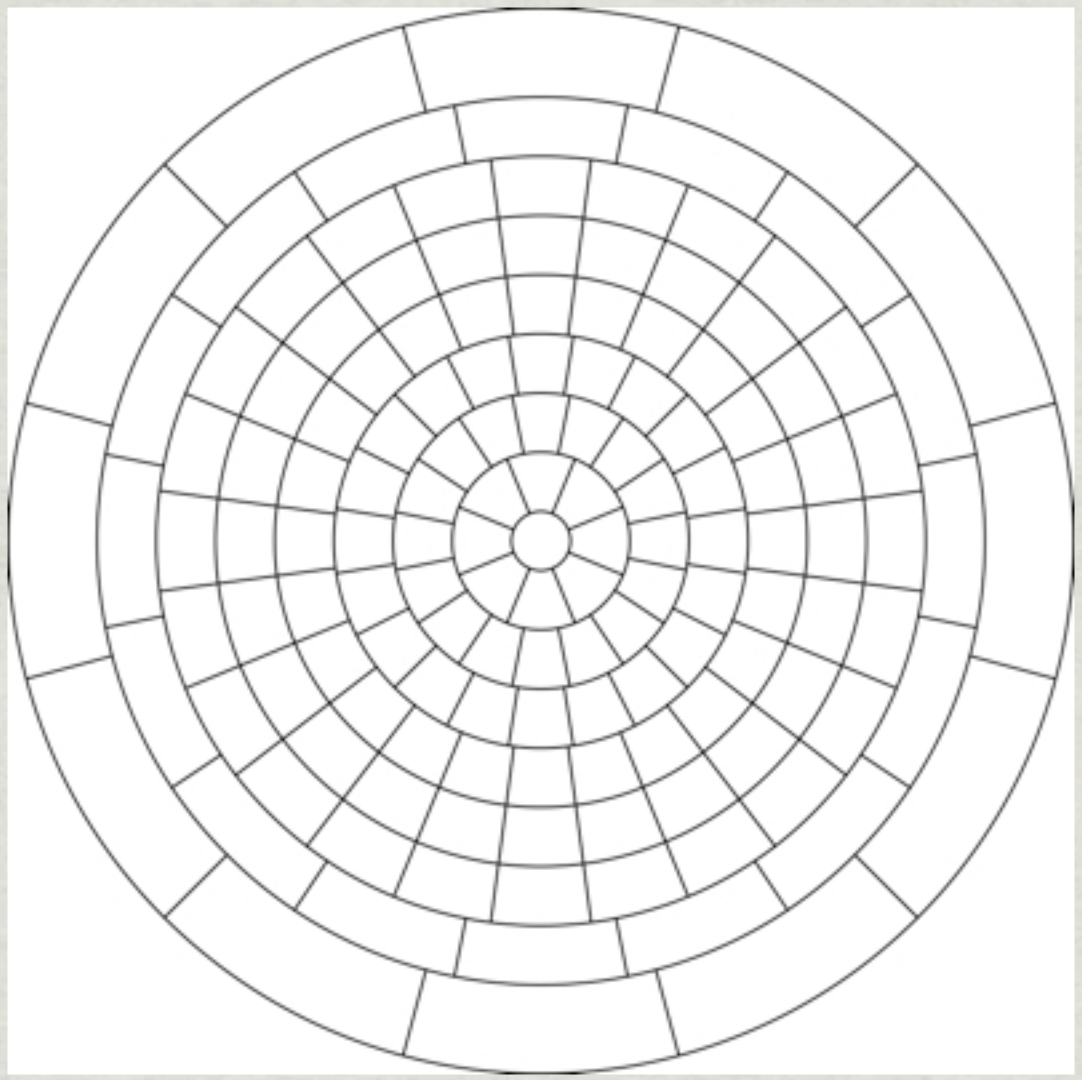
Talk Overview

- Need for a variable-resolution representation
- Design considerations & solutions
- WINDOW 6 XML format extensions
- New **genBSDF** options
- An example or two
- Outstanding issues

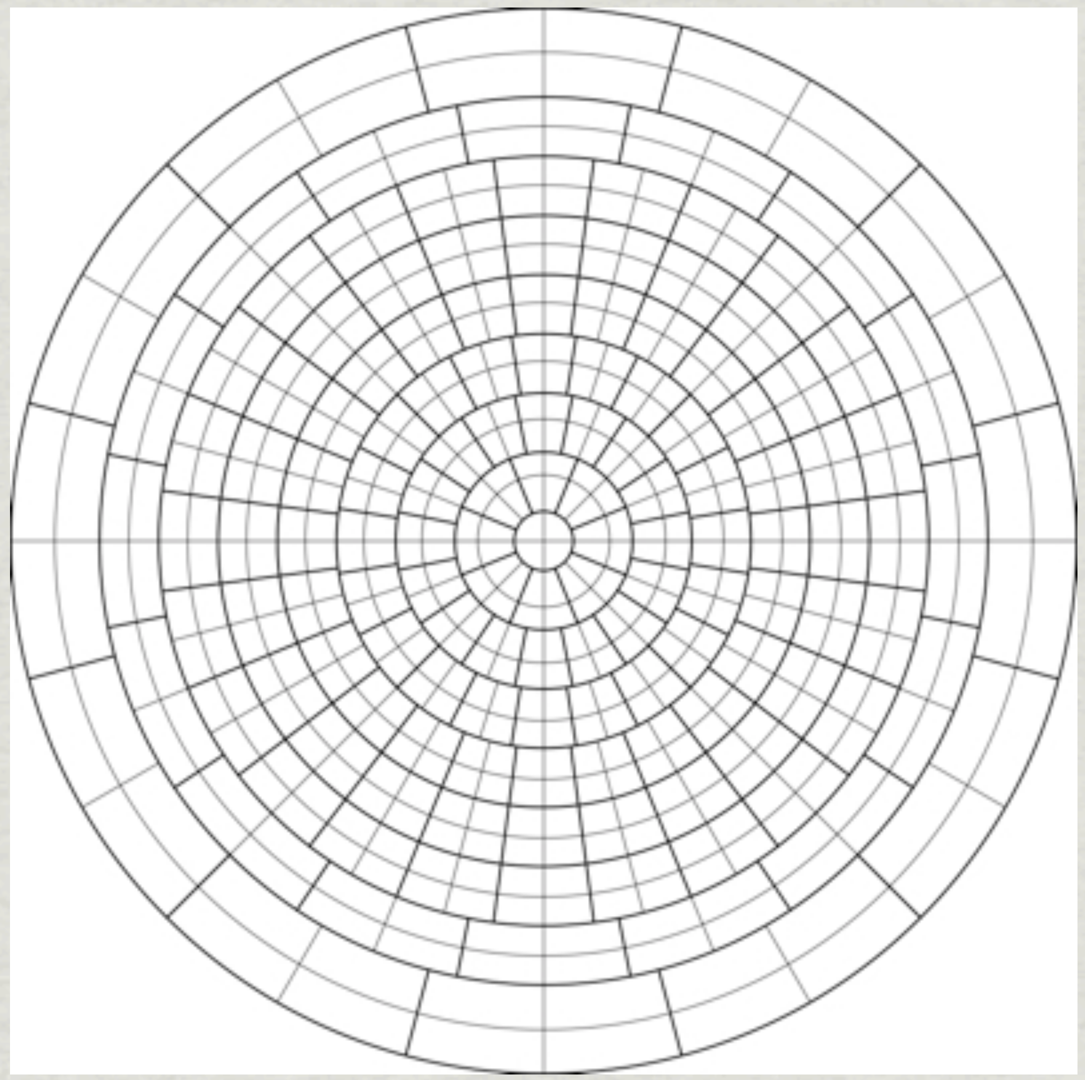
BSDF Resolution Sensitivity Test for CFS

- Is the Klems BSDF resolution enough to determine:
    - Lighting energy use (workplane illuminance)
    - Glare assessment

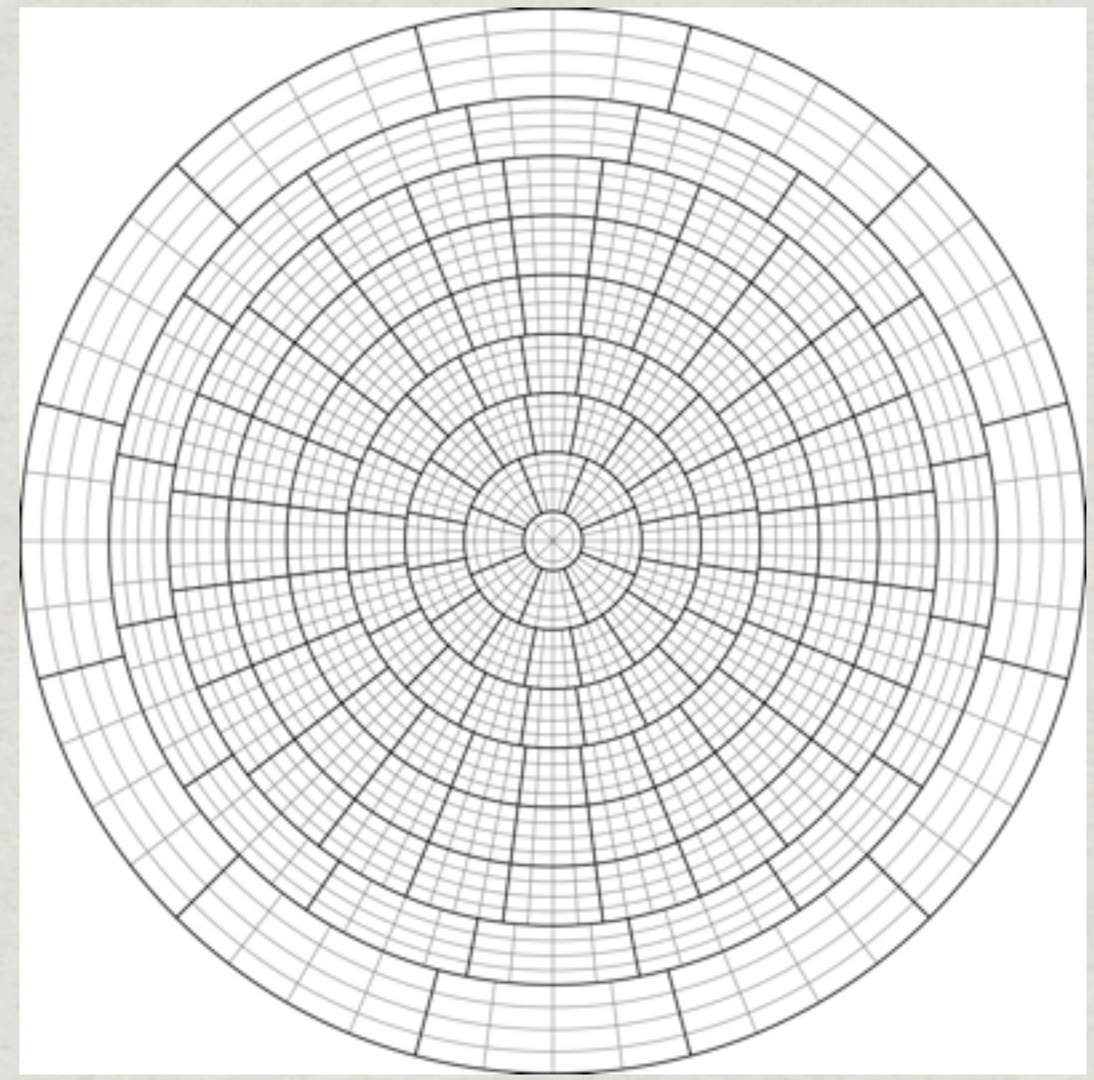
BSDF Resolutions



FULL KLEMS
145 PATCHES

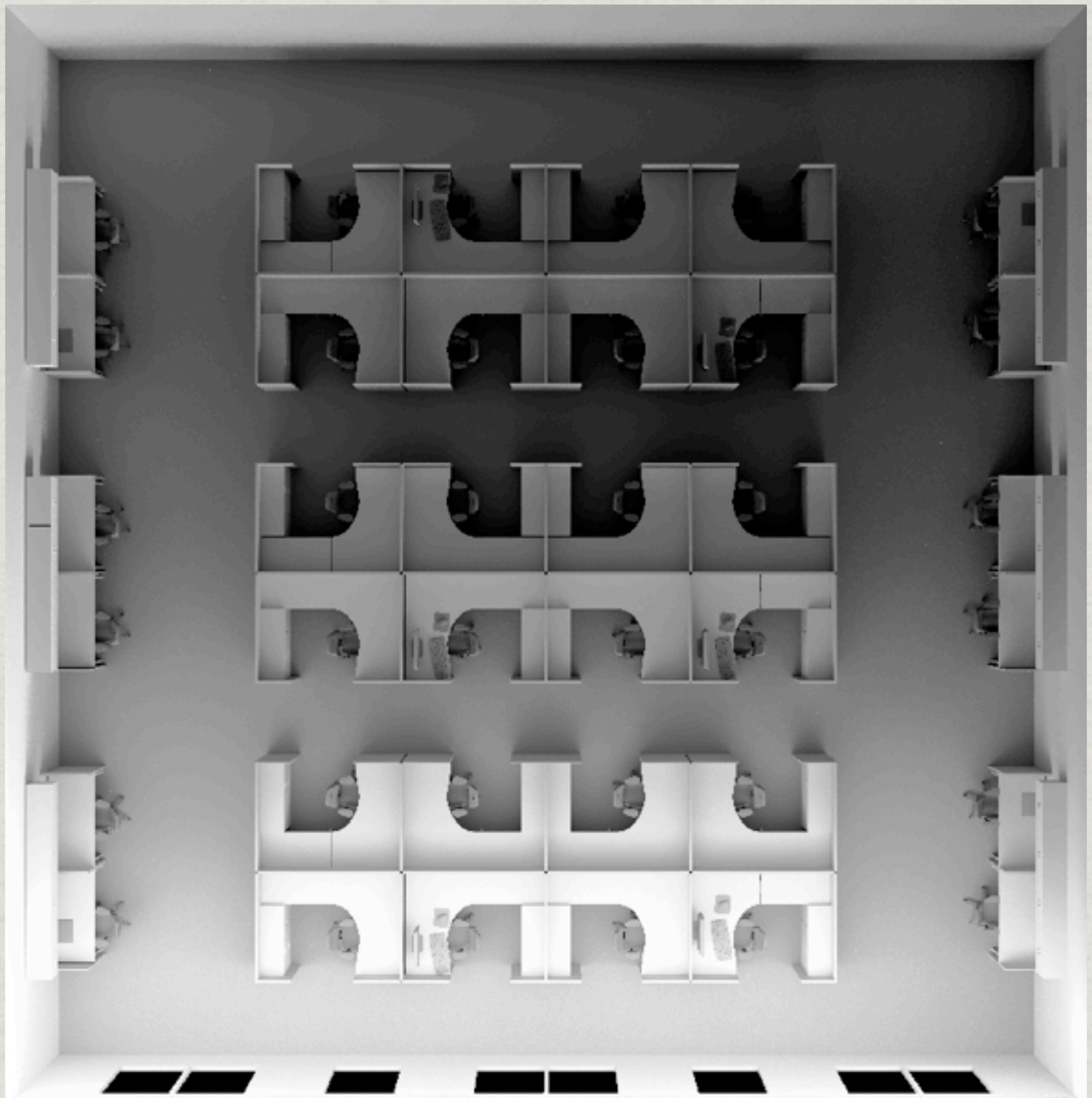


2X KLEMS
580 PATCHES

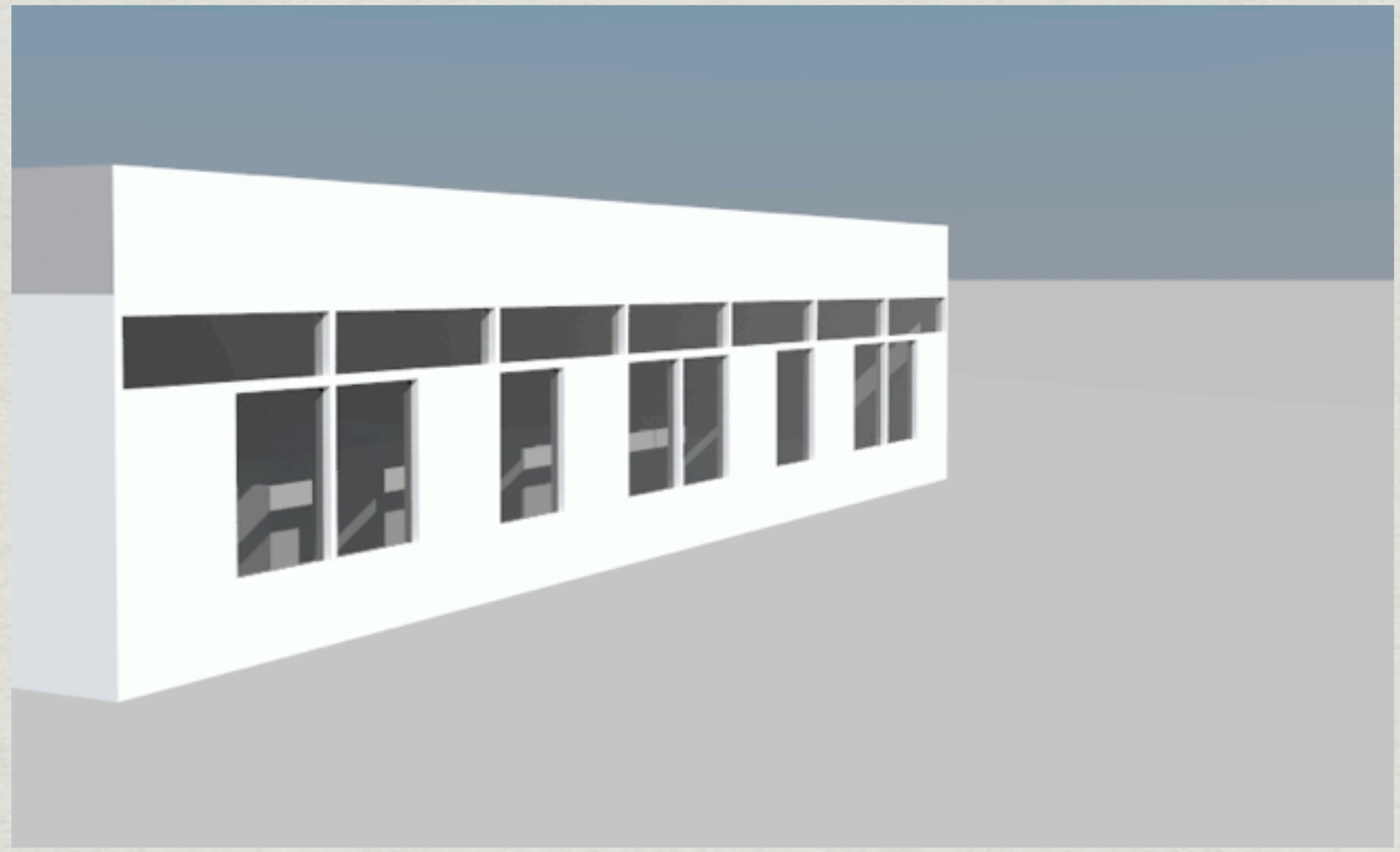


4X KLEMS
2320 PATCHES

Model

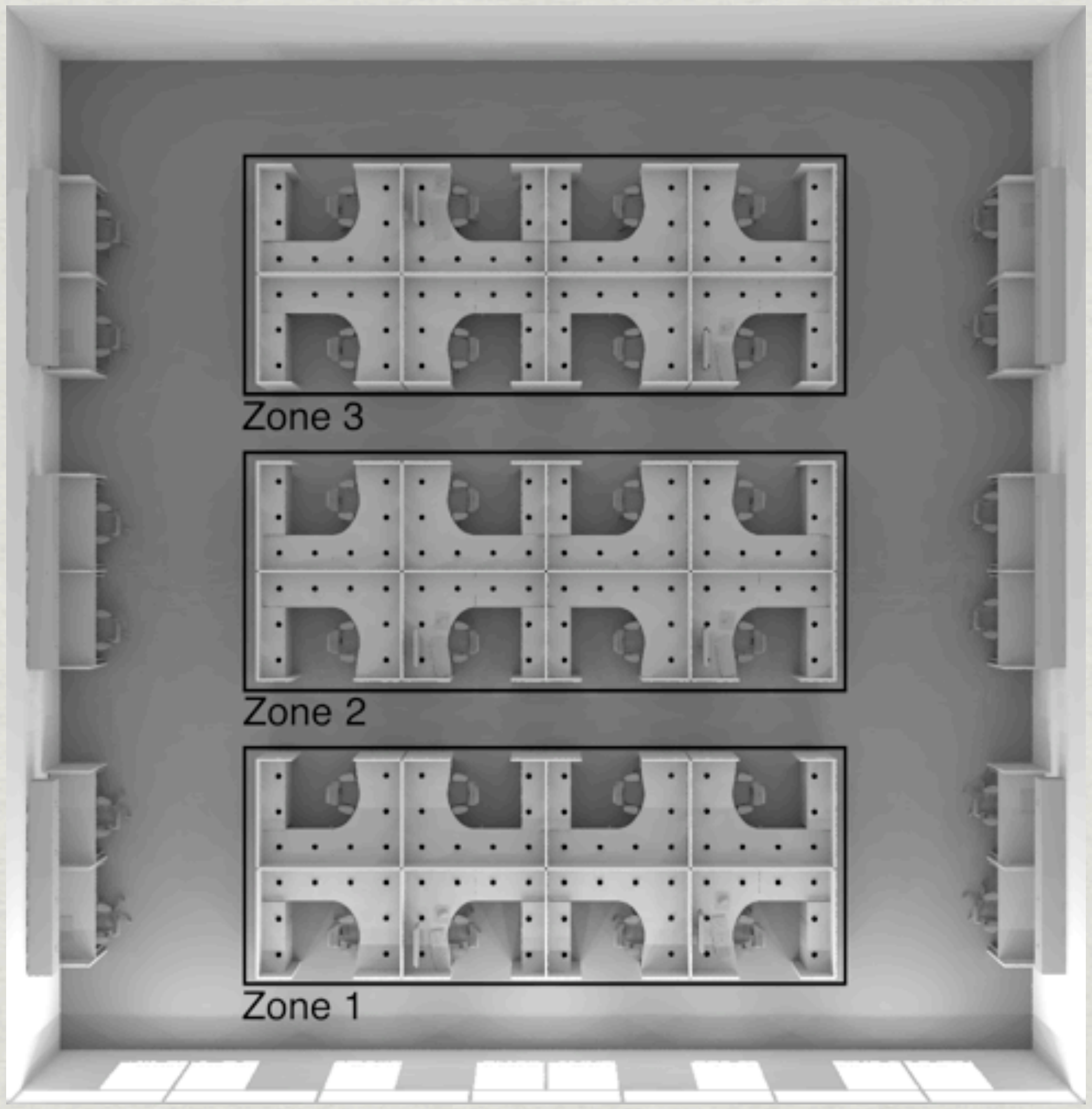


FLOOR PLAN

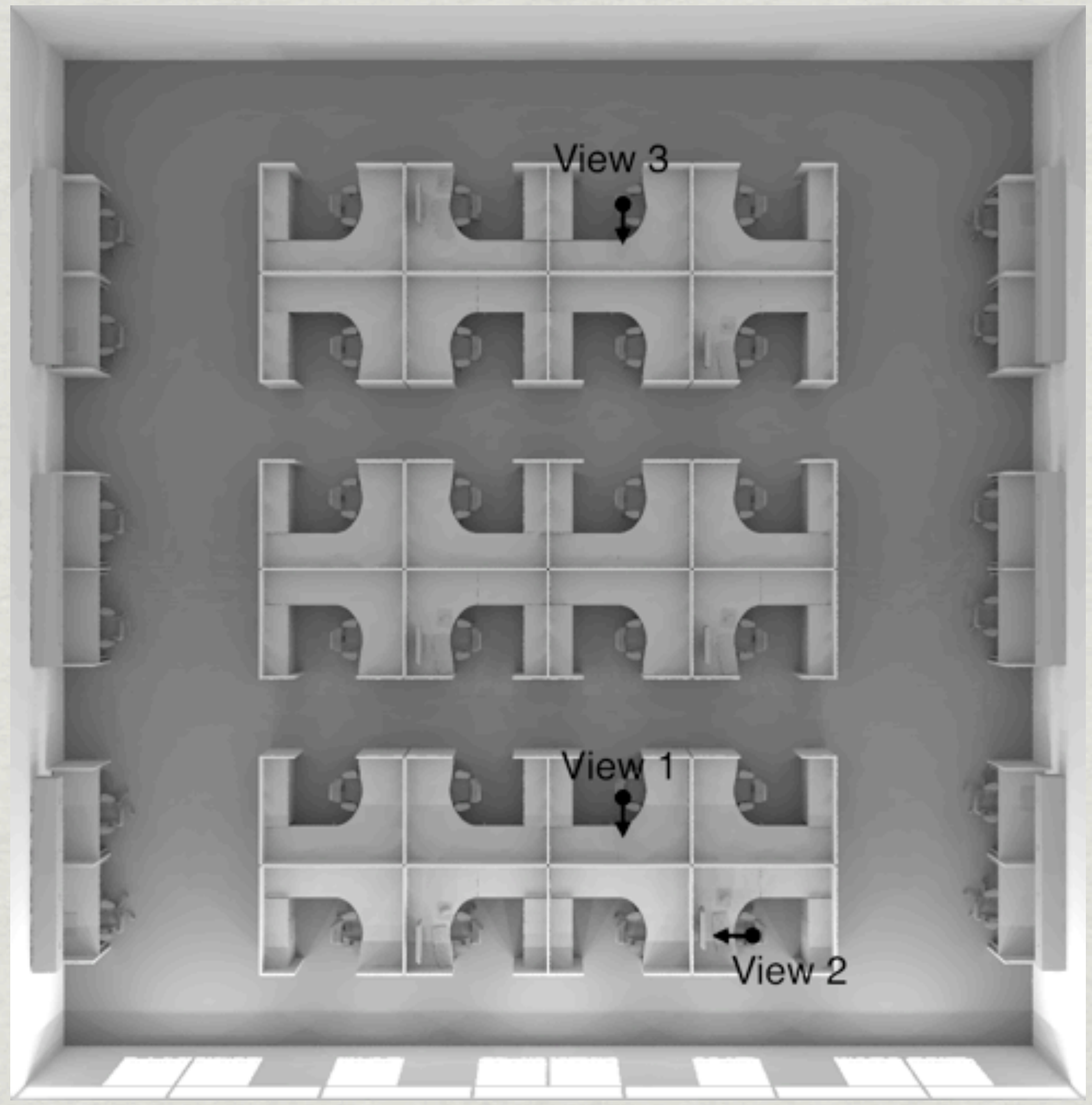


SOUTH FACING ELEVATION

Model



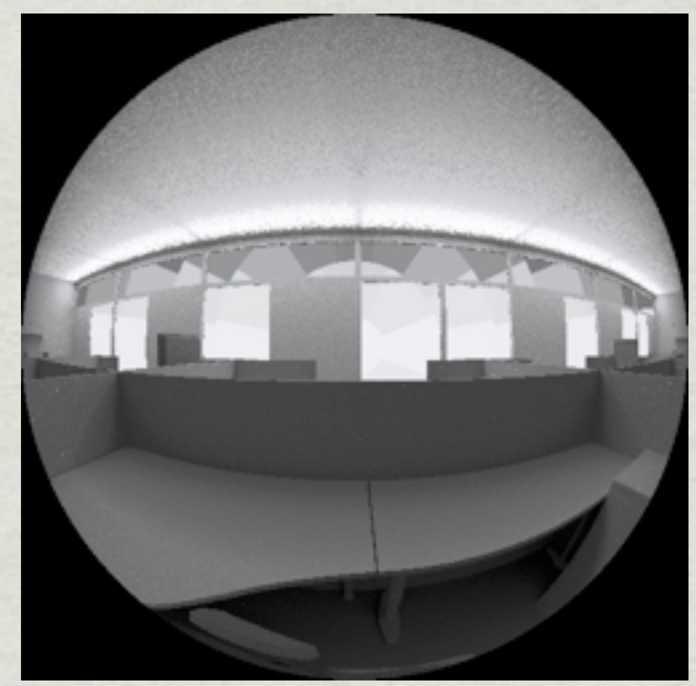
WORK PLANE ILLUMINANCE GRIDS



VIEW POINTS AND DIRECTIONS

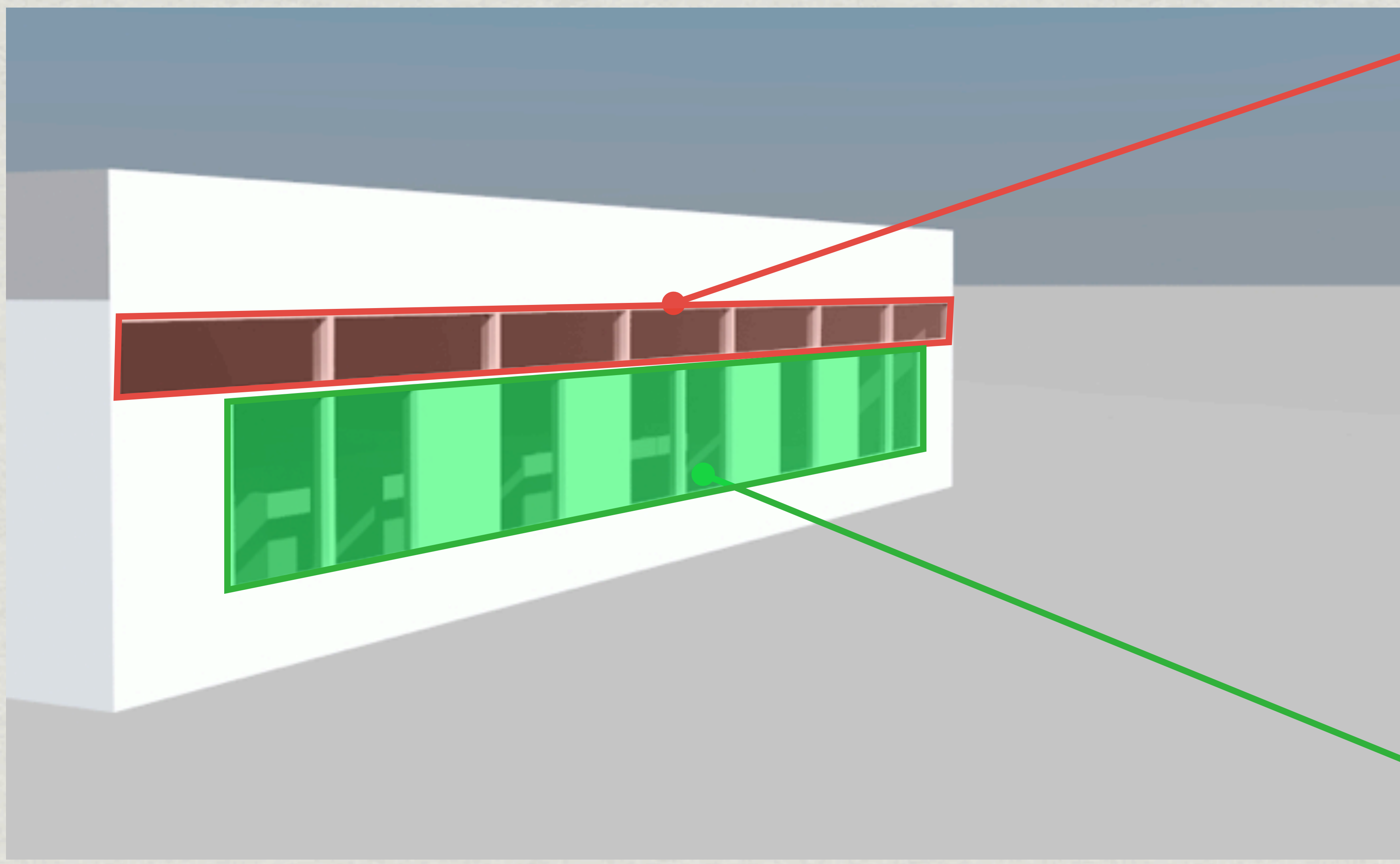
VIEW 1

VIEW 2



VIEW 3

Window Systems



SOUTH FACING ELEVATION

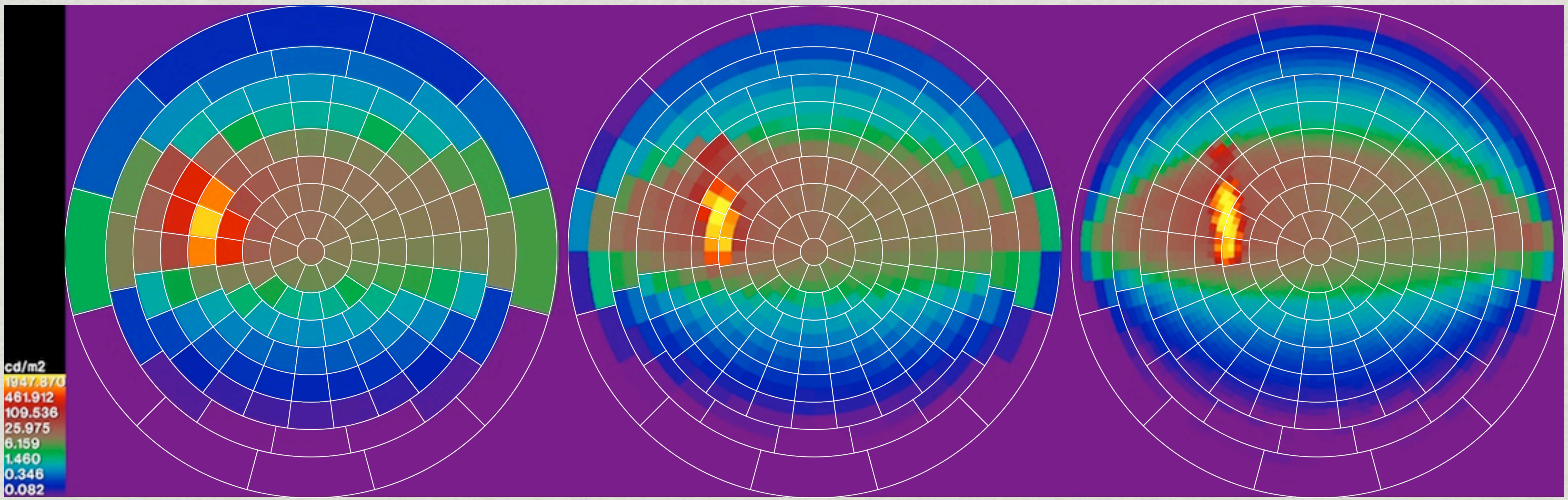
OPTICAL LIGHT SHELF

BLACK OUT

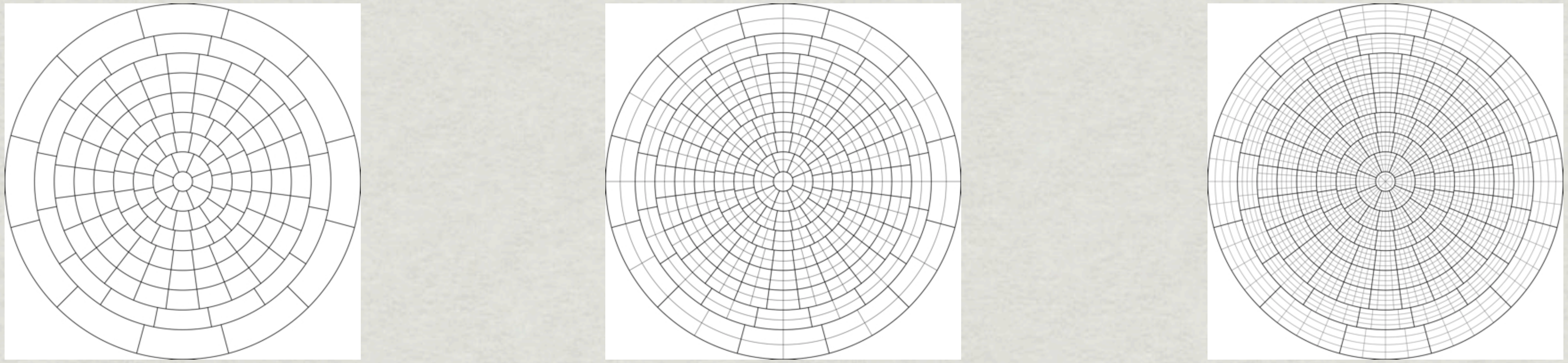
Static Simulations

- 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)

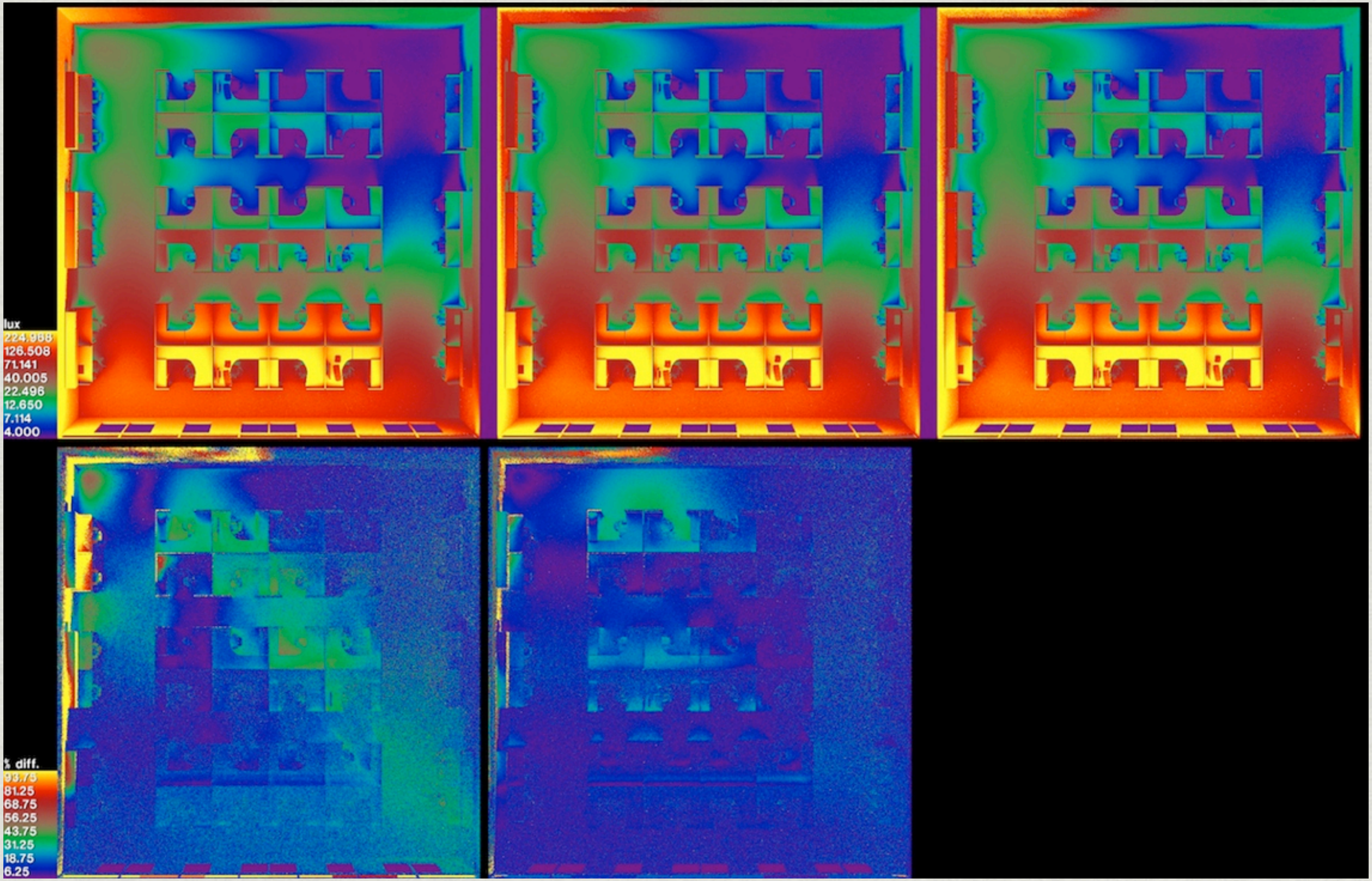


WINDOW OUTPUT

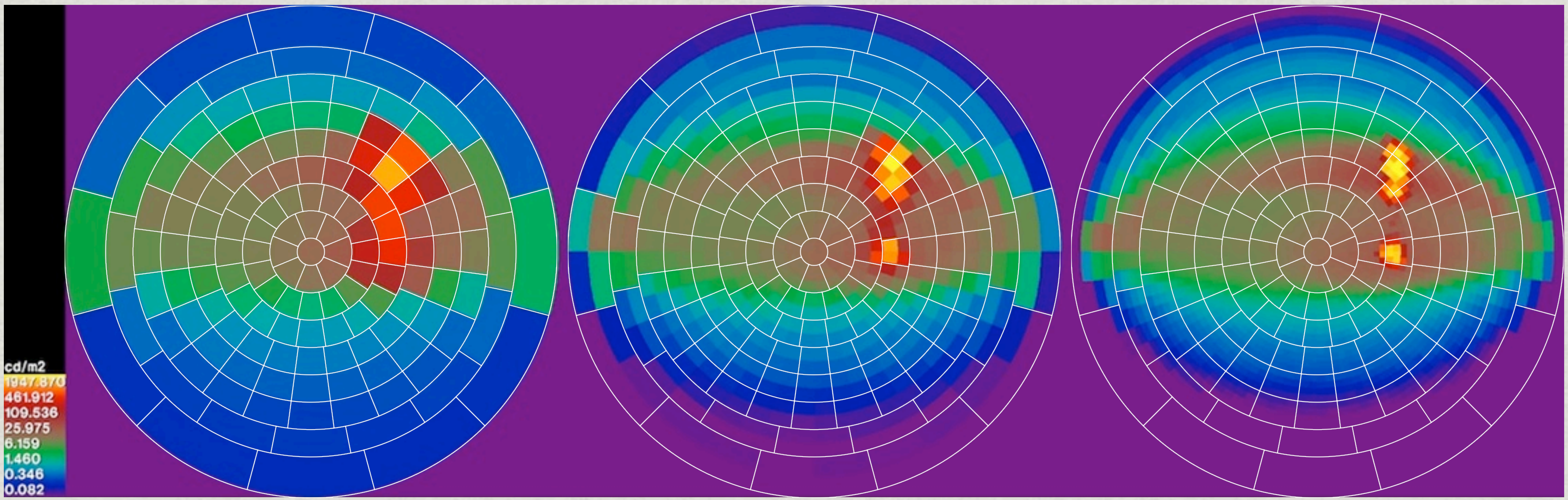


ILLUMINANCE [LUX]

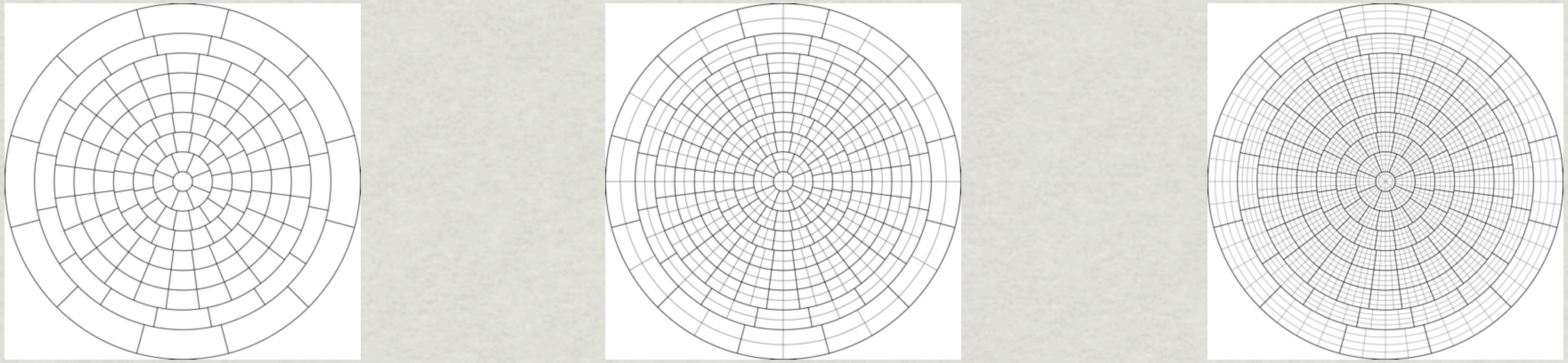
ABSOLUTE DIFFERENCE [%]
(FROM 4X)



Worst Match
(Dec, 14:00)

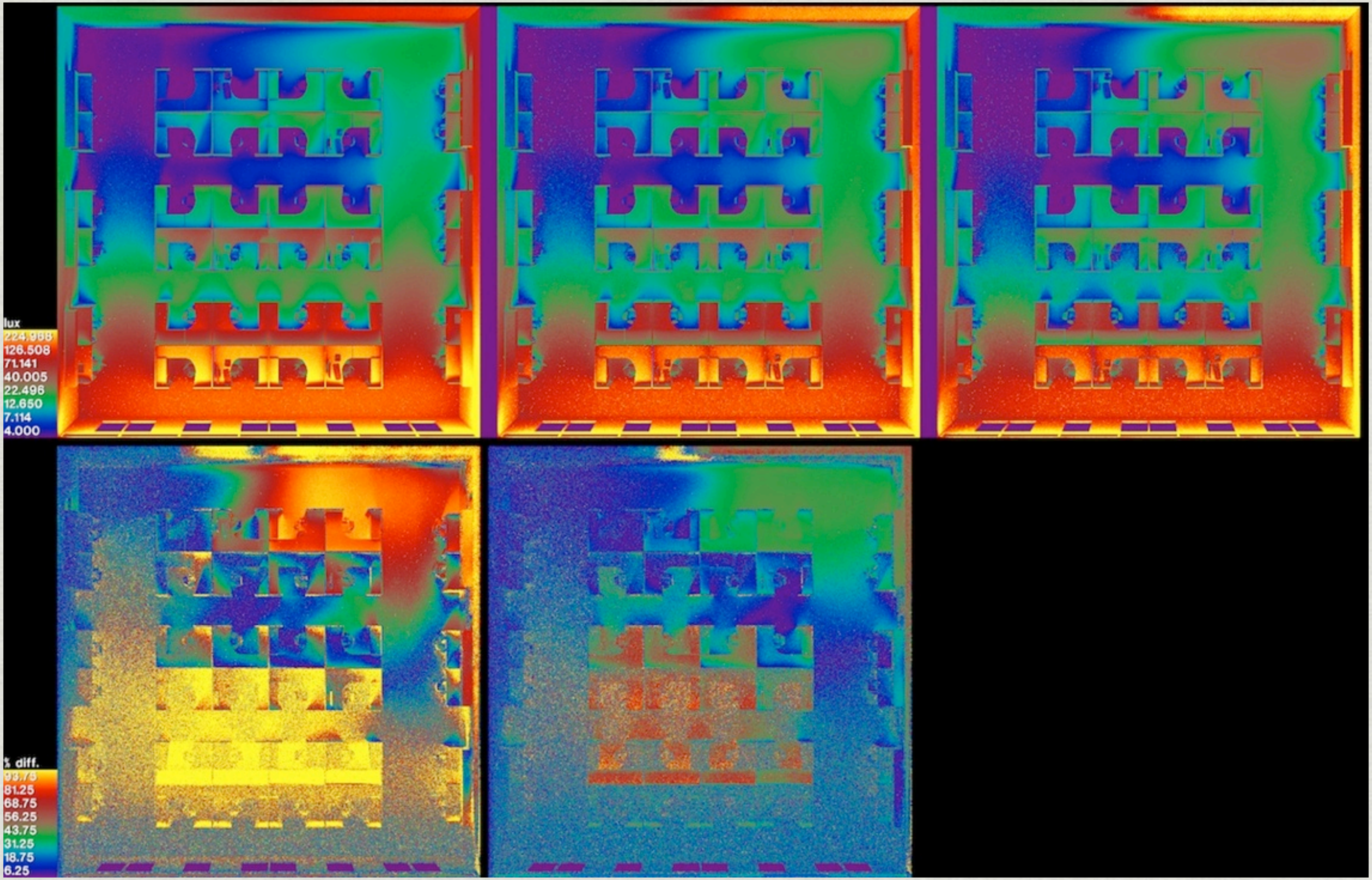


WINDOW OUTPUT



ILLUMINANCE [LUX]

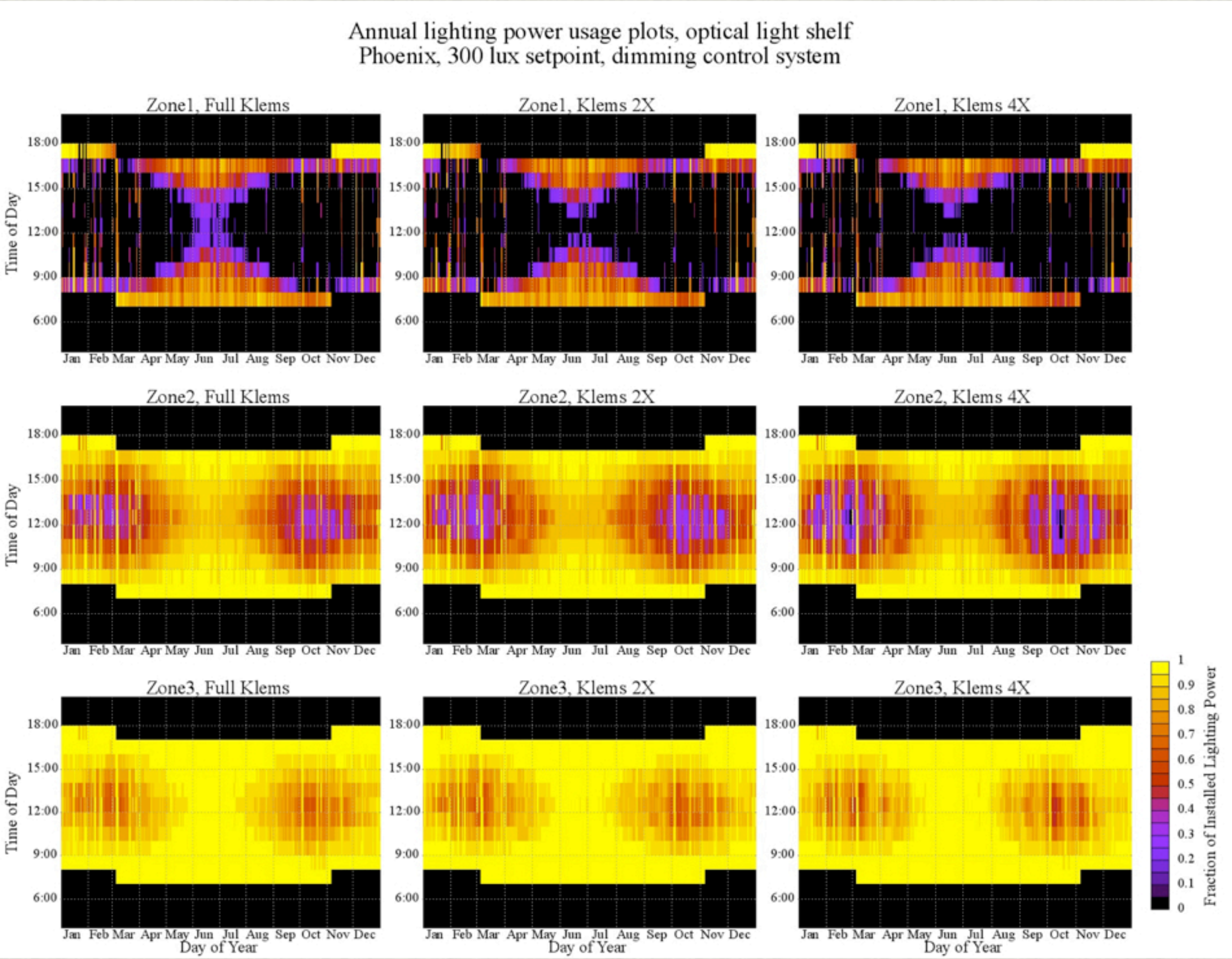
ABSOLUTE DIFFERENCE [%]
(FROM 4X)



Annual Simulations

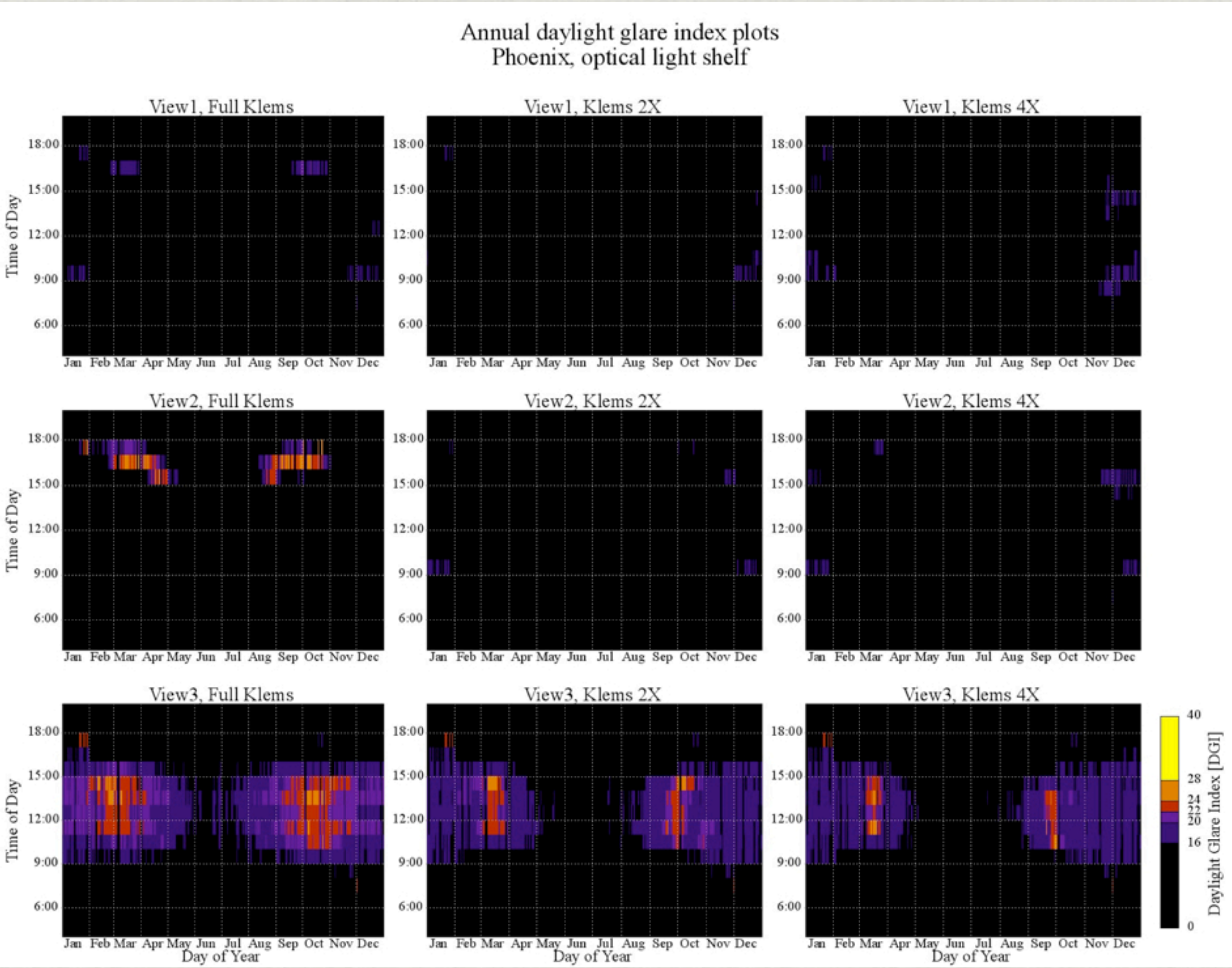
- Ran hourly analysis for Phoenix

Lighting Energy Use



	Full Klems	2x Klems	4x Klems
Zone 1	72%	73%	74%
Zone 2	20%	21%	23%
Zone 3	8%	7%	7%

Daylight Glare Index

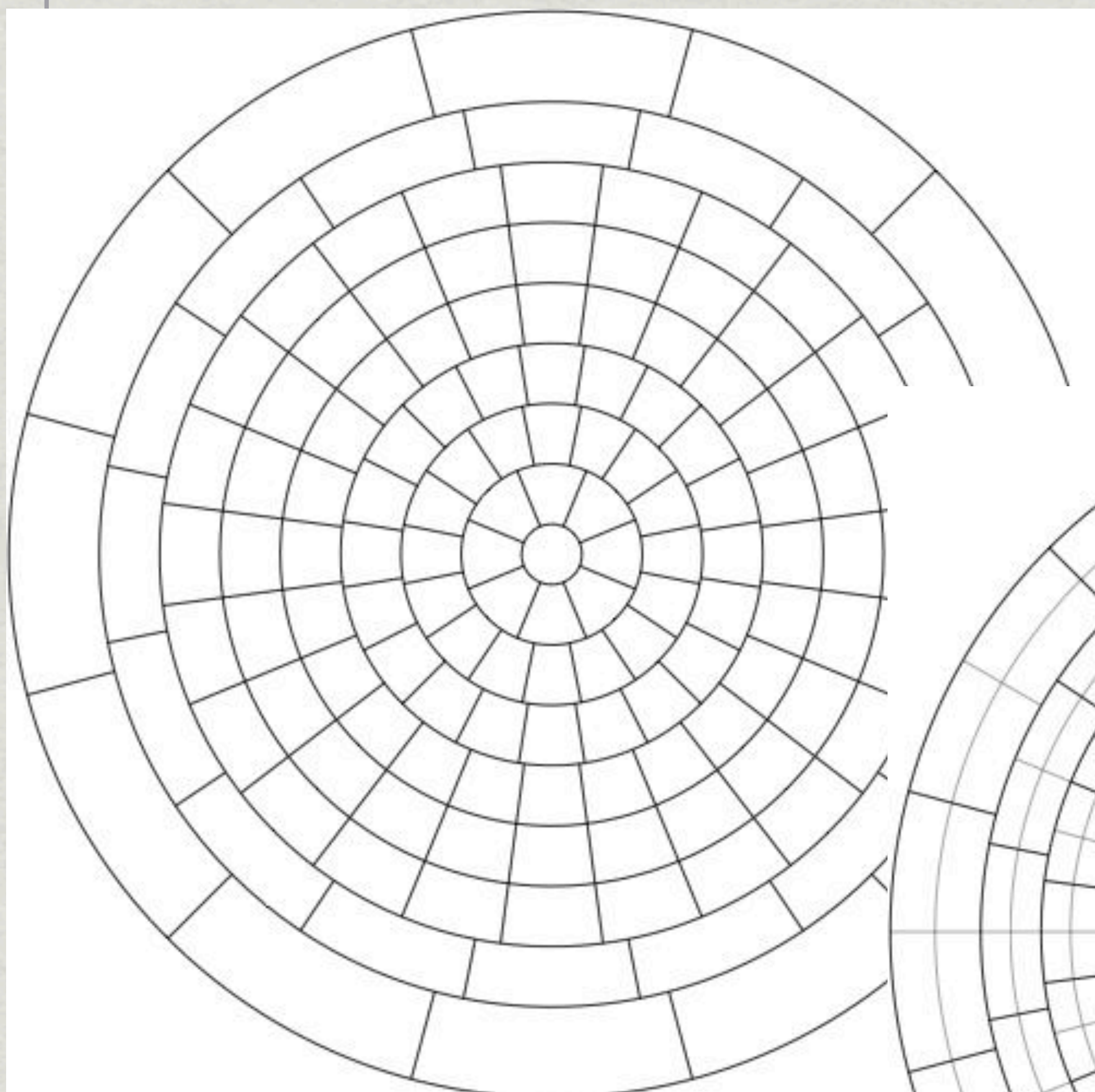


	Full Klems	2x Klems	4x Klems
View 1	0%	0%	0%
View 2	3%	0%	0%
View 3	9%	3%	2%

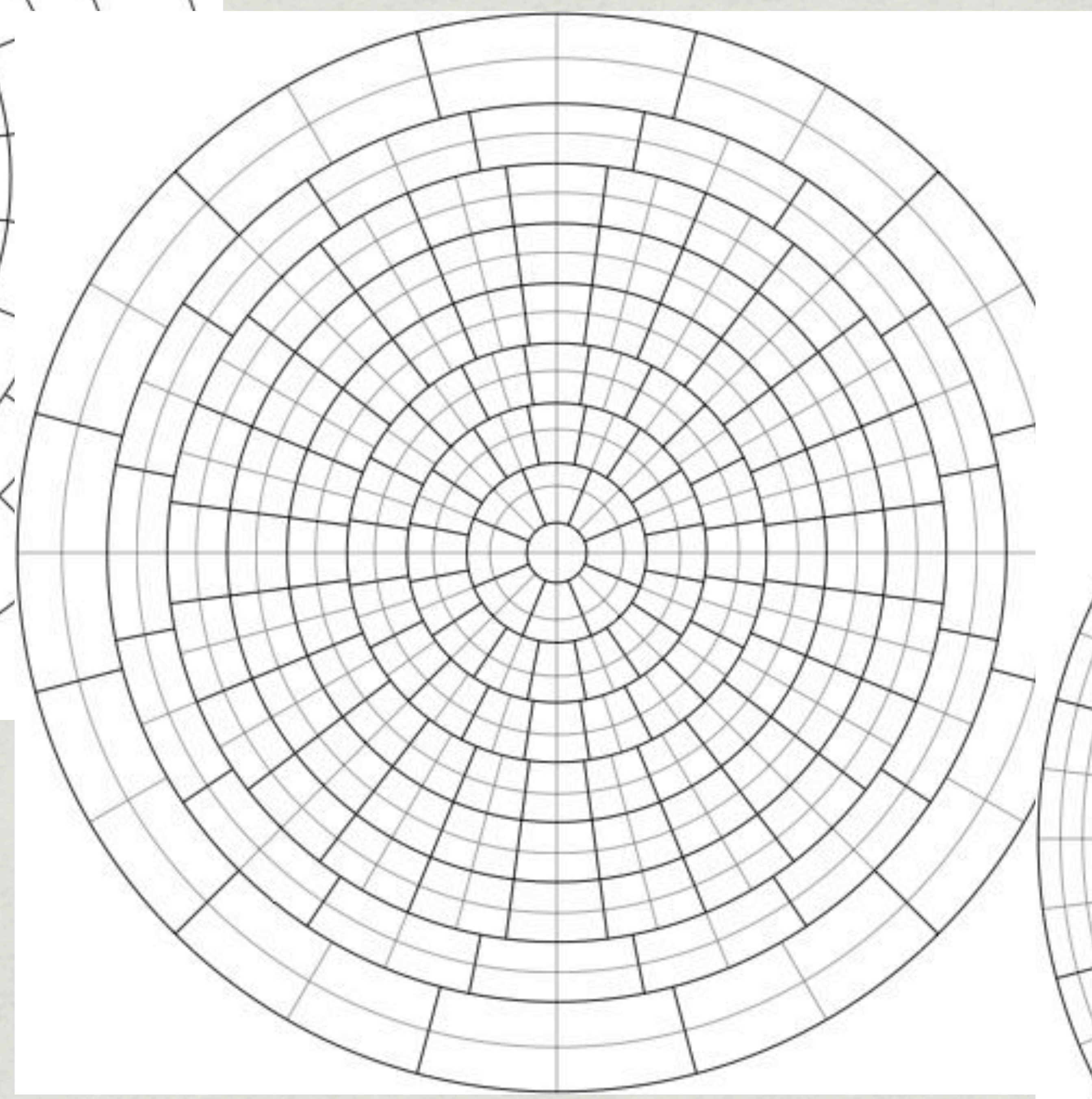
Design Considerations

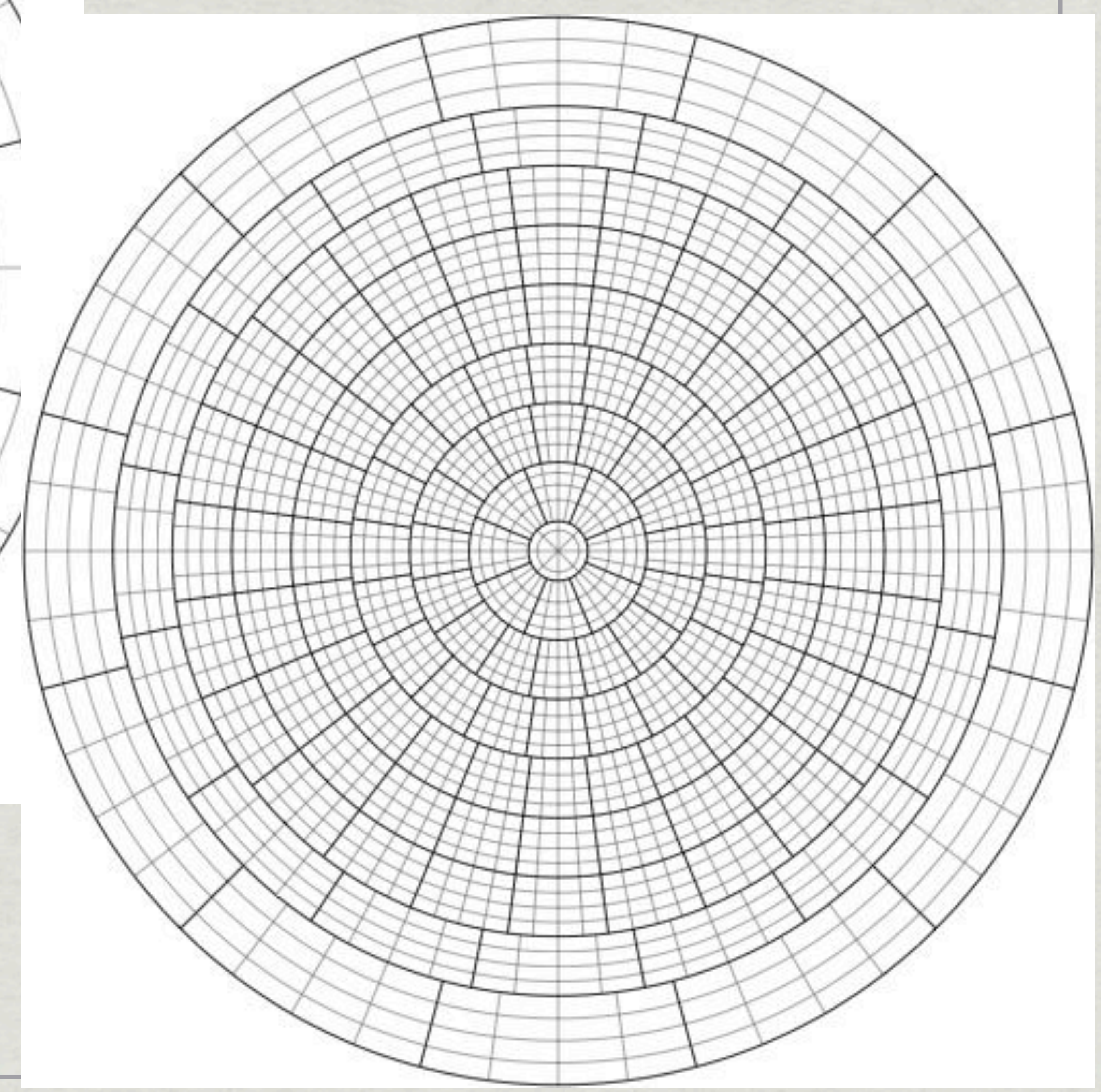
- Basic: capture peaks, compress smooth regions
- Scale input & output resolutions synchronously
- Require efficient sampling method
- Prefer compact disk/memory representation
- Optimize for isotropic and anisotropic distributions

Quantizing Directions

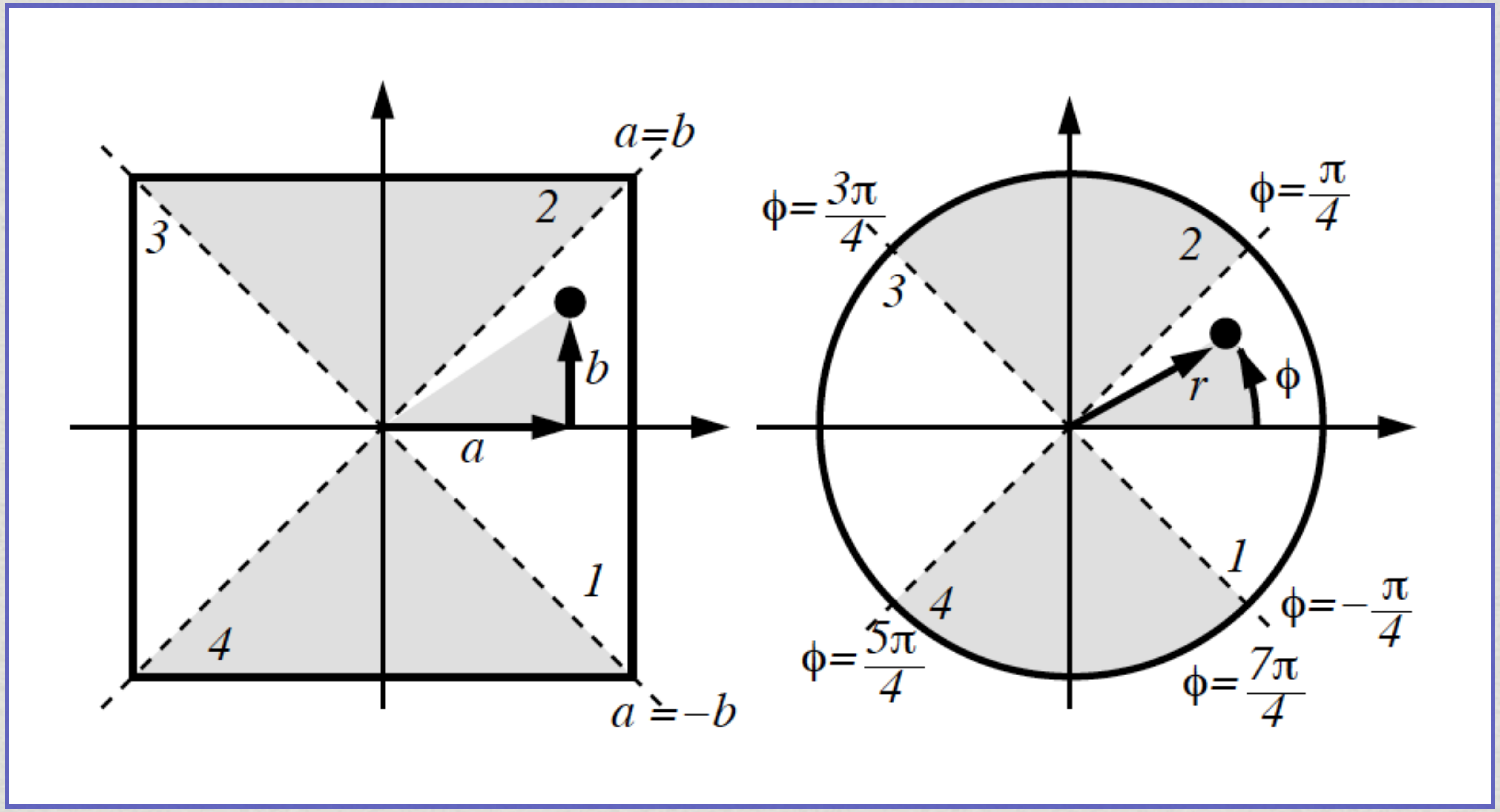


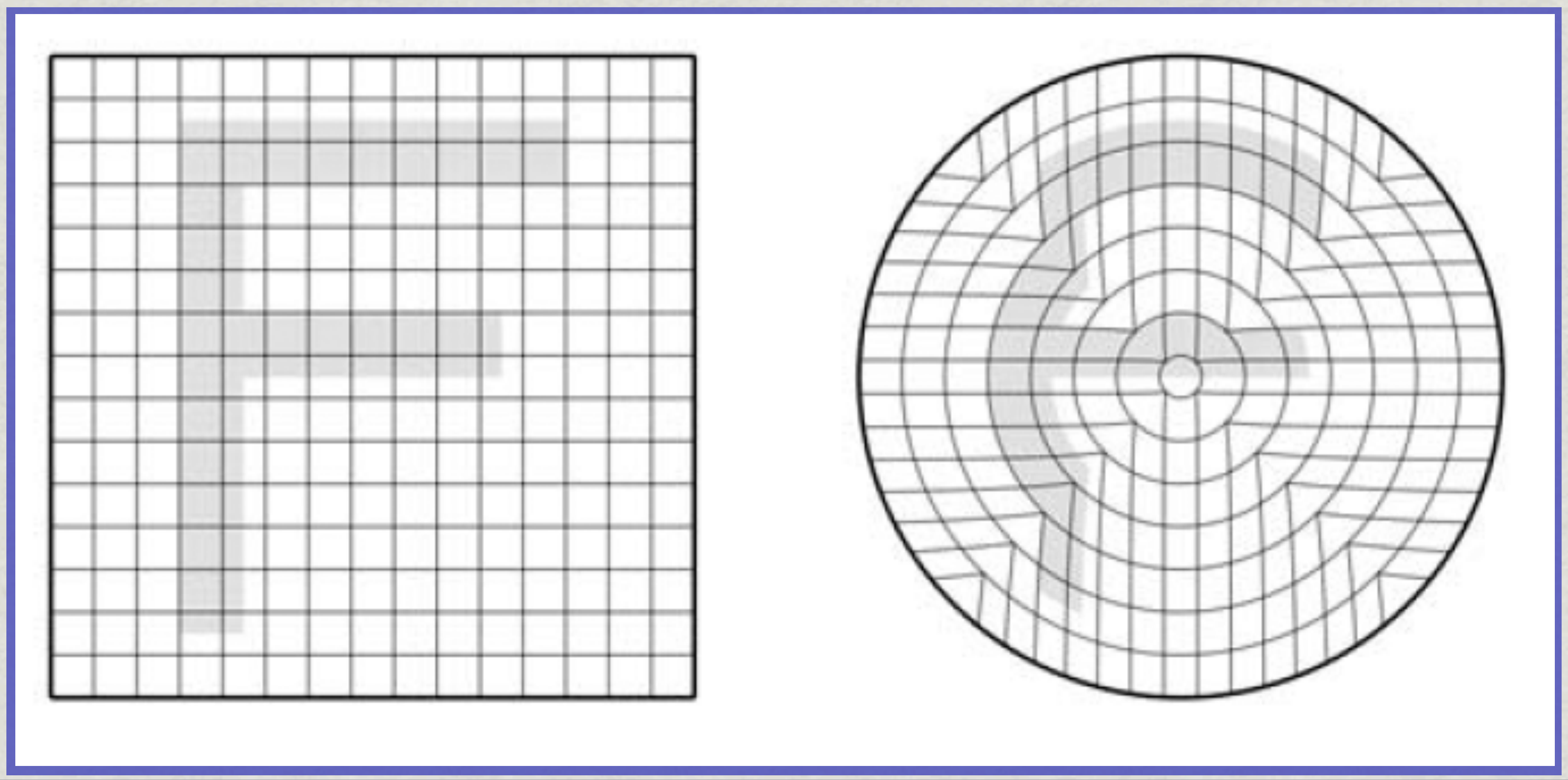
Altitude-Azimuth methods difficult to subsample





Shirley-Chiu Mapping



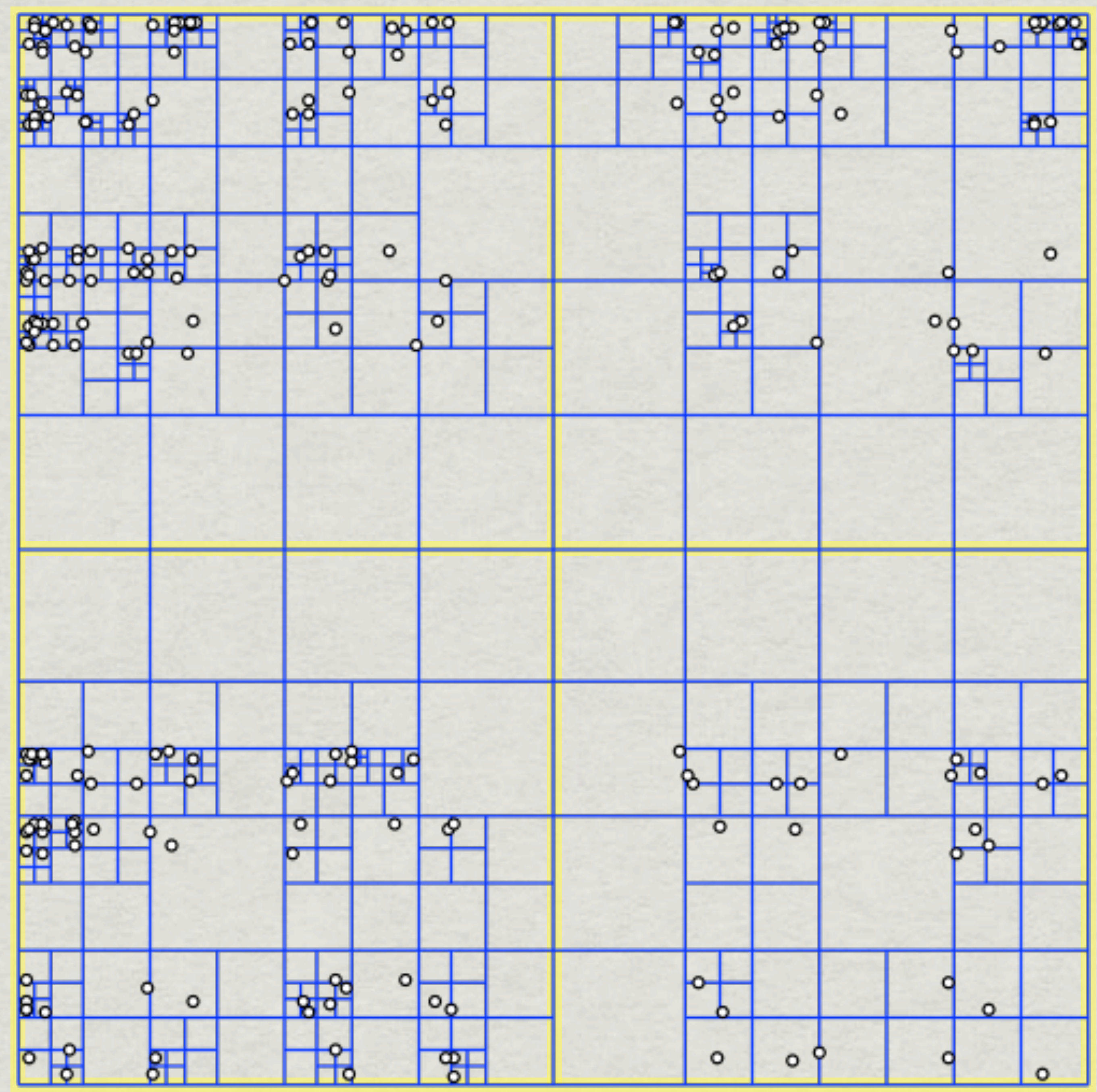


Maintains relative areas, important for hemispherical sampling

Peter Shirley and Kenneth Chiu, "A Low Distortion Map Between Disk and Square," JGT 2(3), 1997

Cartesian Subdivision

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

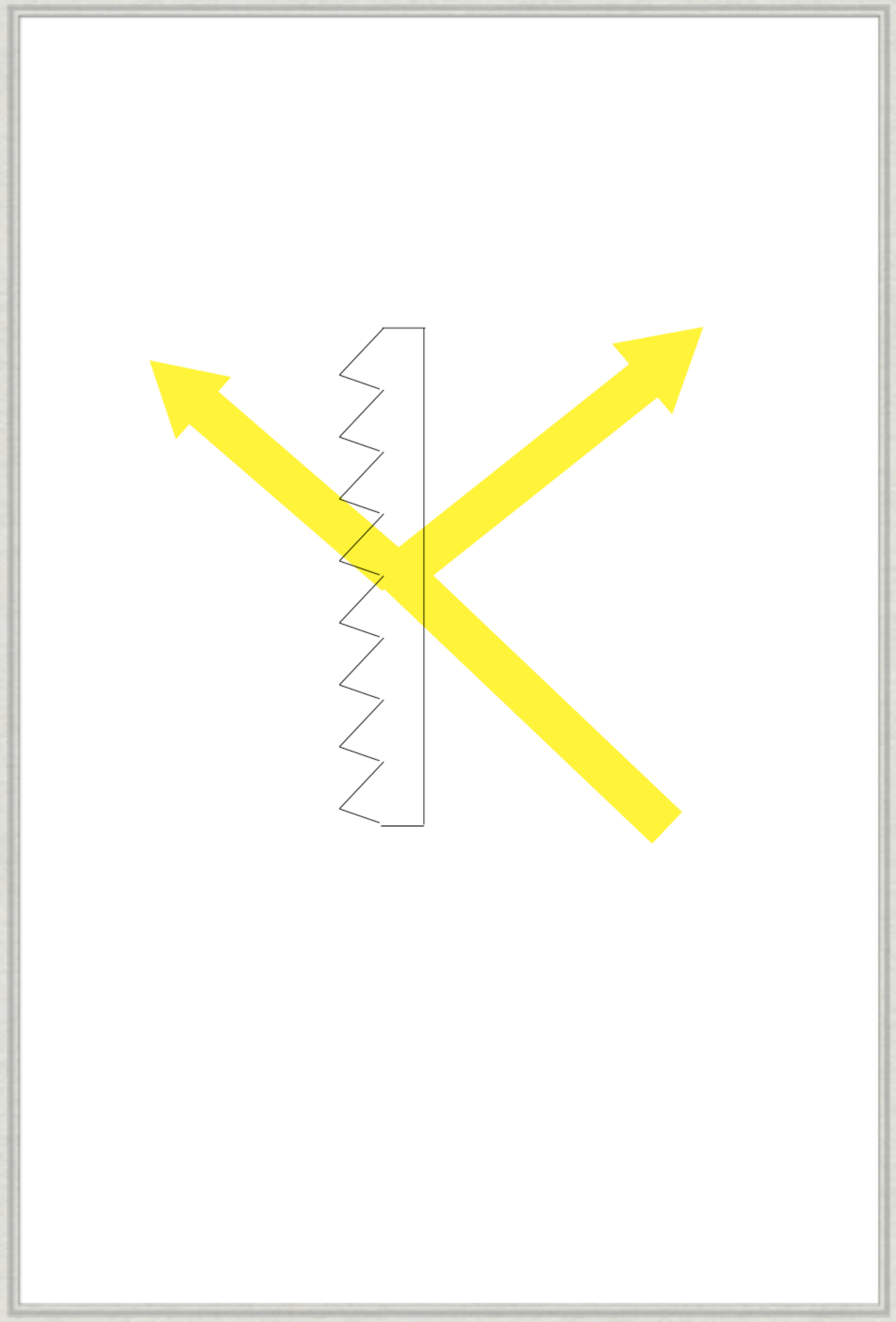


Example method: Quadtree

Reason for Scaling Input Resolution with Output

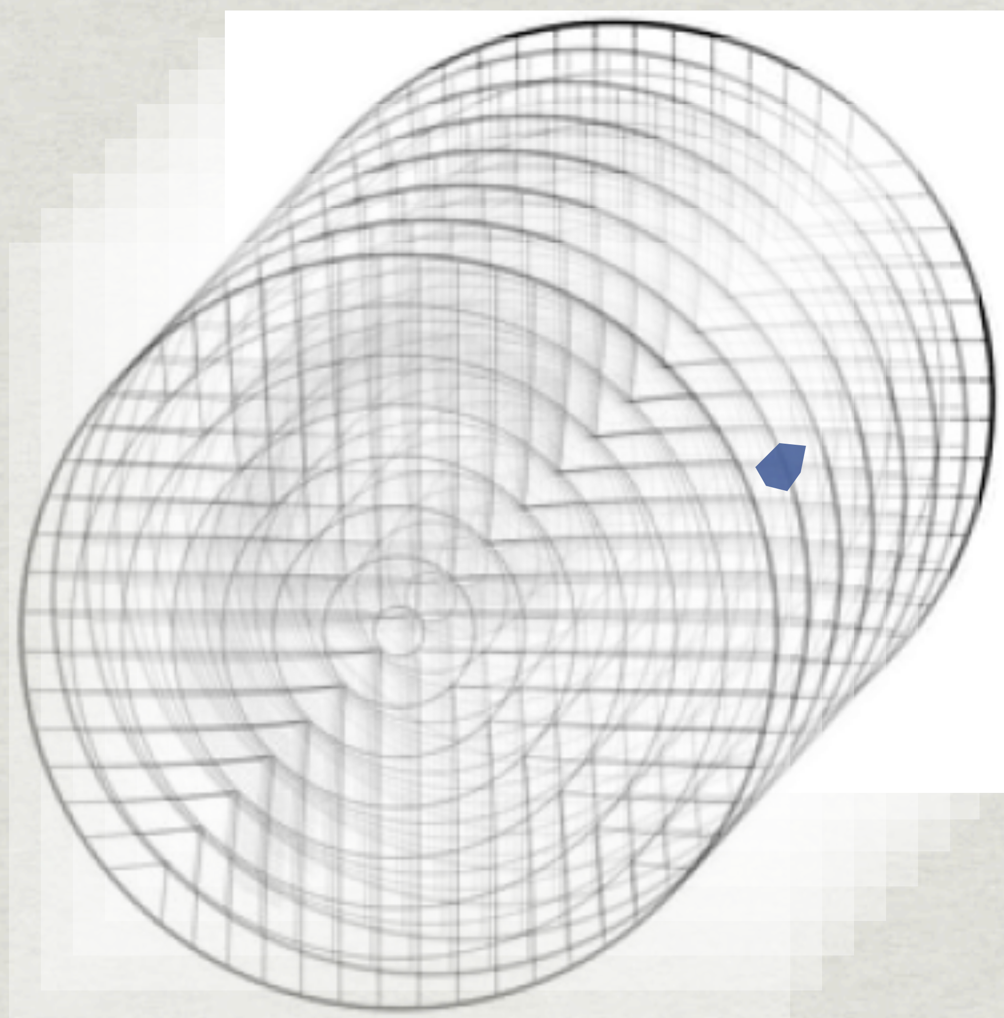
- If we have a peak in a particular output direction, its position will shift in relation to the input direction

- 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

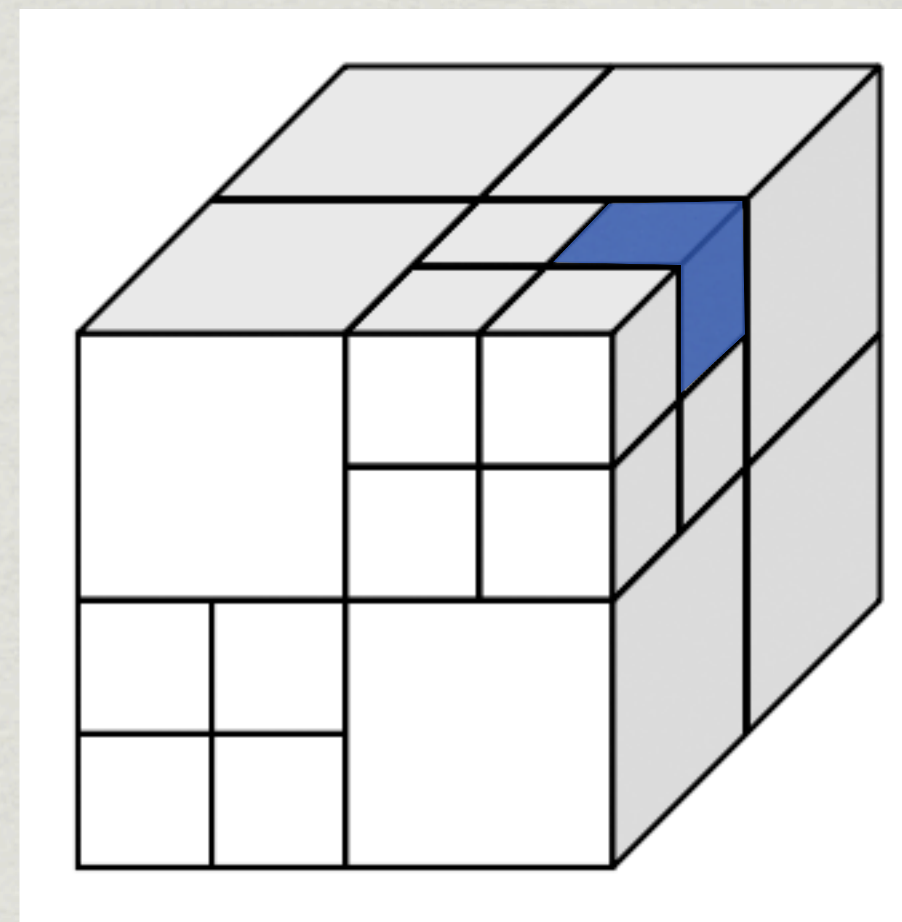


How It Works:

Take our output direction map:



Sample region



Layer it for each input direction **Represent as octree**

Anisotropic BSDF adds another dimension, making it a hextree

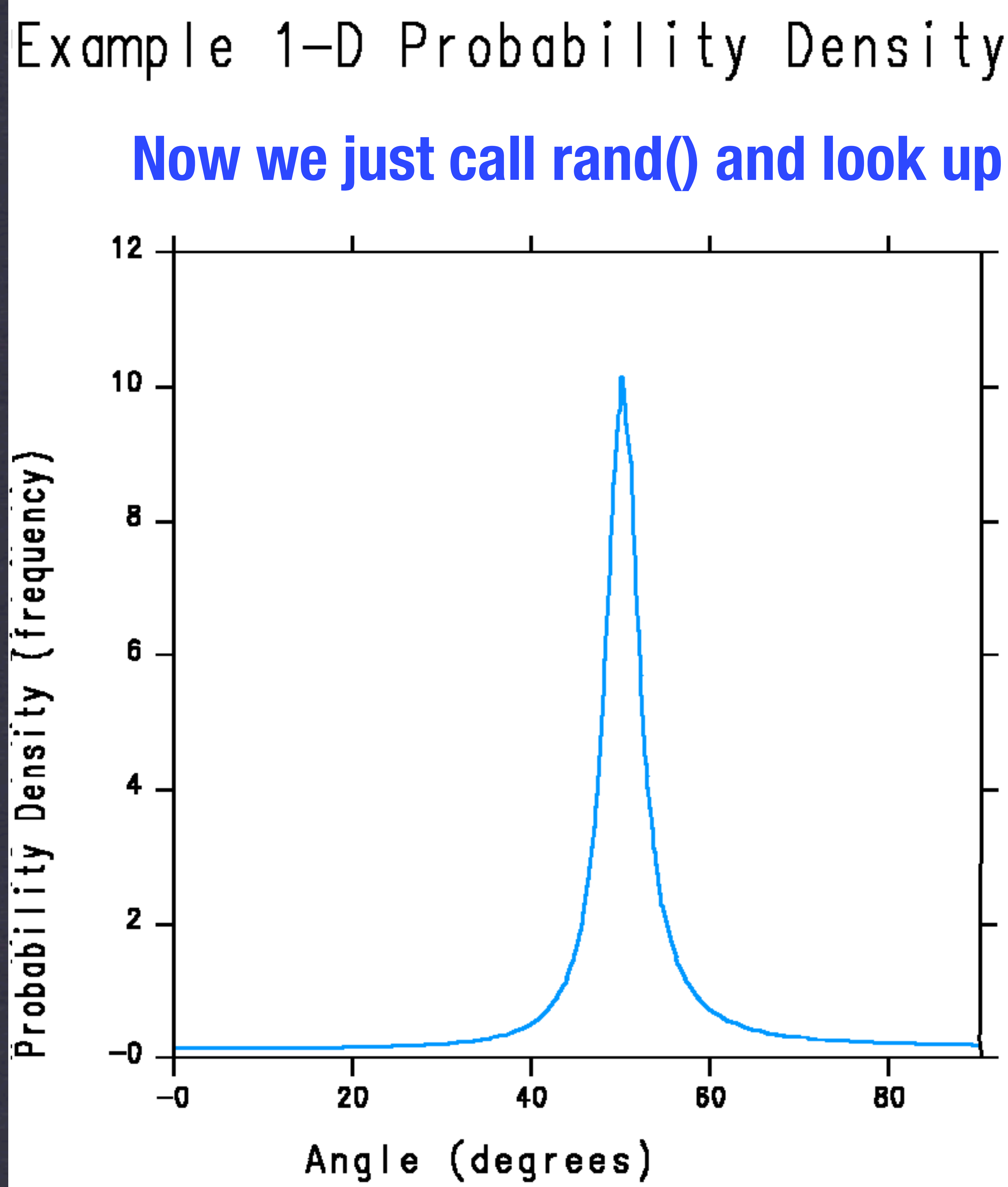
Resolution scales in all dimensions, minimizing footprint

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



Now we just call rand() and look up angles

Invert It

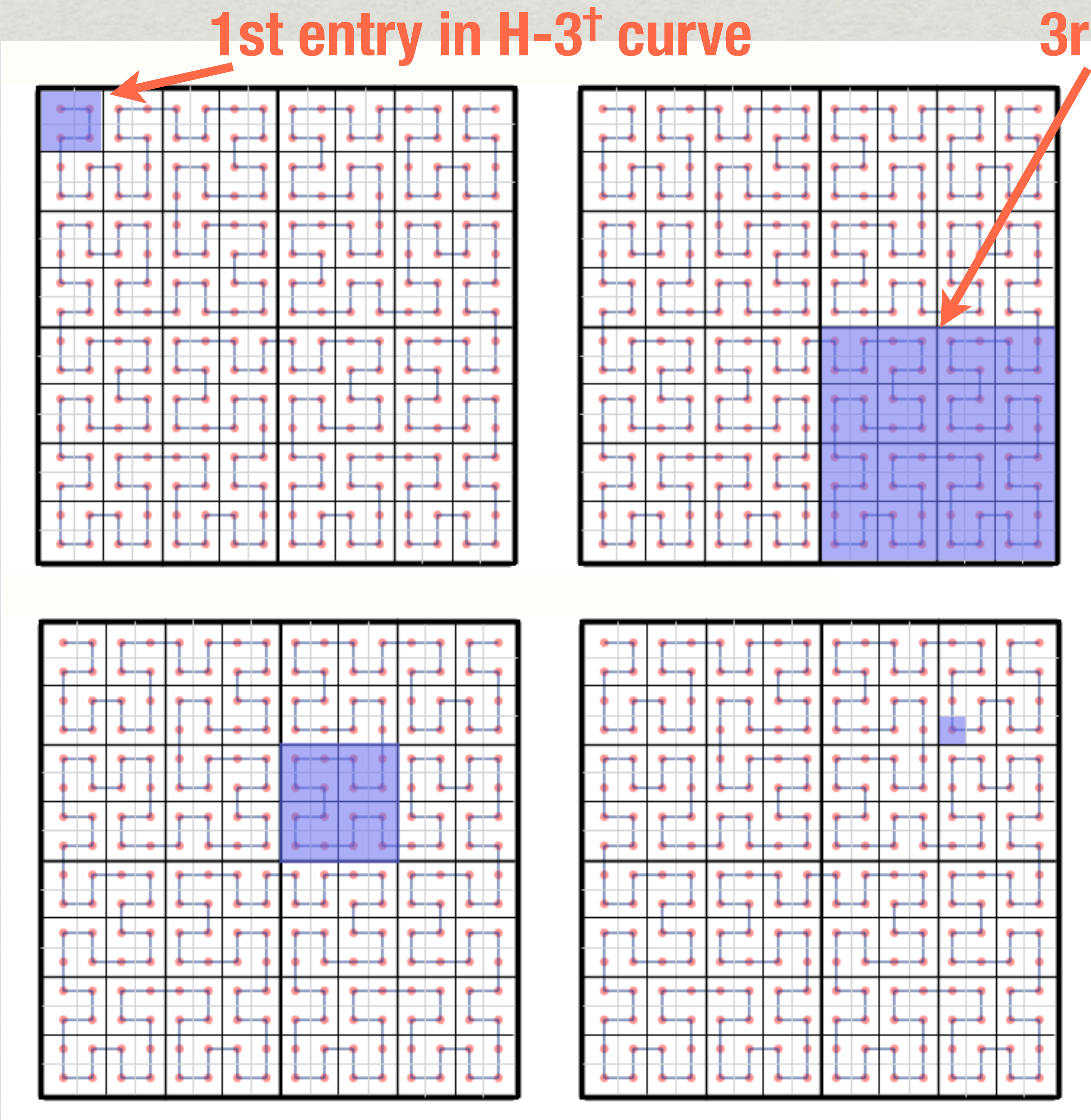
Review of Monte Carlo Inversion

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!

2-D Example



- 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

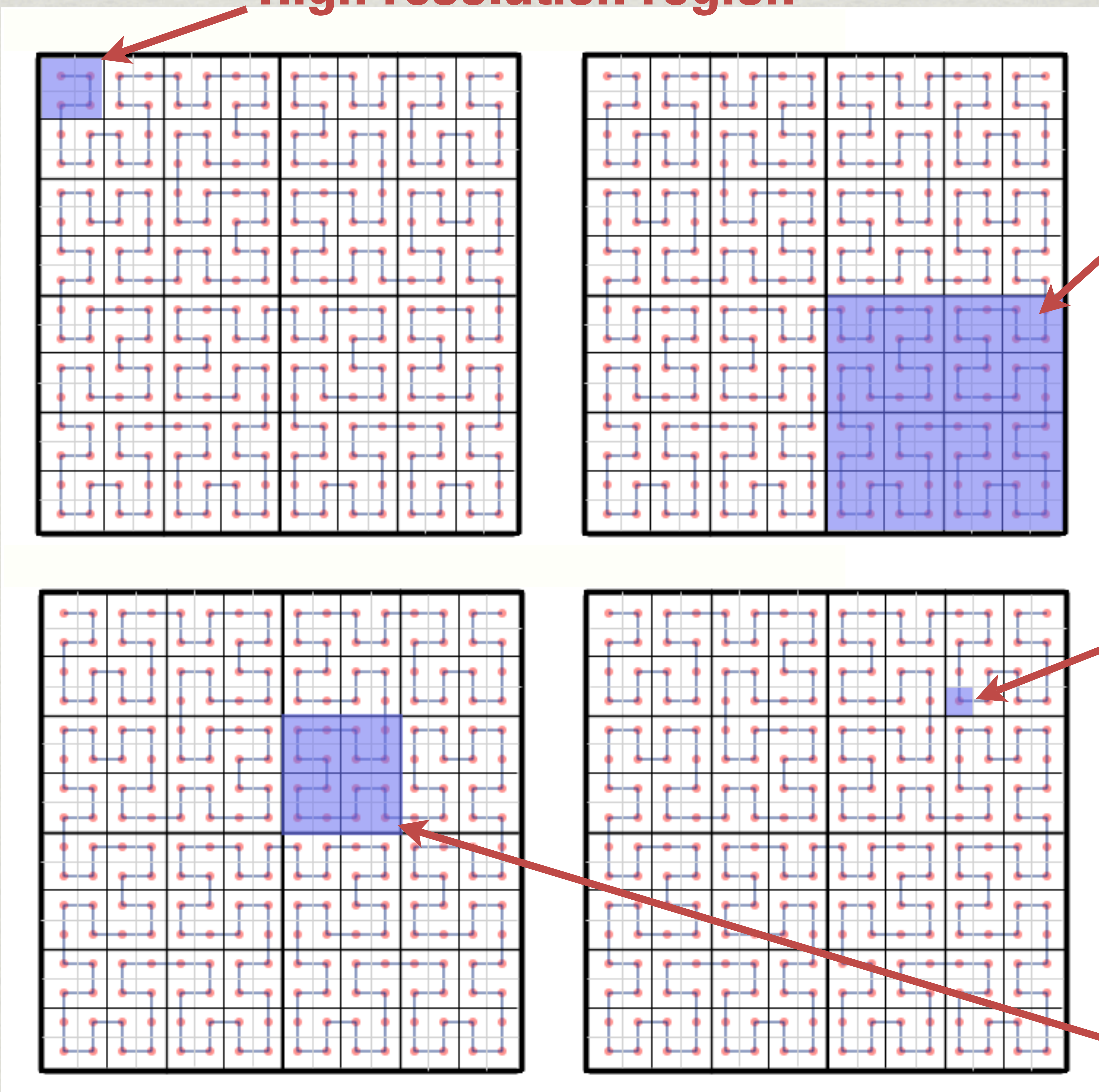
†H-3 means each dimension divided by 2^3

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

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

Tensor Tree Data Structure

```
/* 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: <ScatteringData> { 0.3183 } </ScatteringData>

New **genBSDF** Options

- It was a lot of new code to add two little options:
  -t3 N Isotropic BSDF at 2^N max. resolution
  -t4 N Anisotropic BSDF at 2^N max. resolution
- Beware of N greater than 6 (64x64x64x64)
  - Need better method to reach higher resolution
- The -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

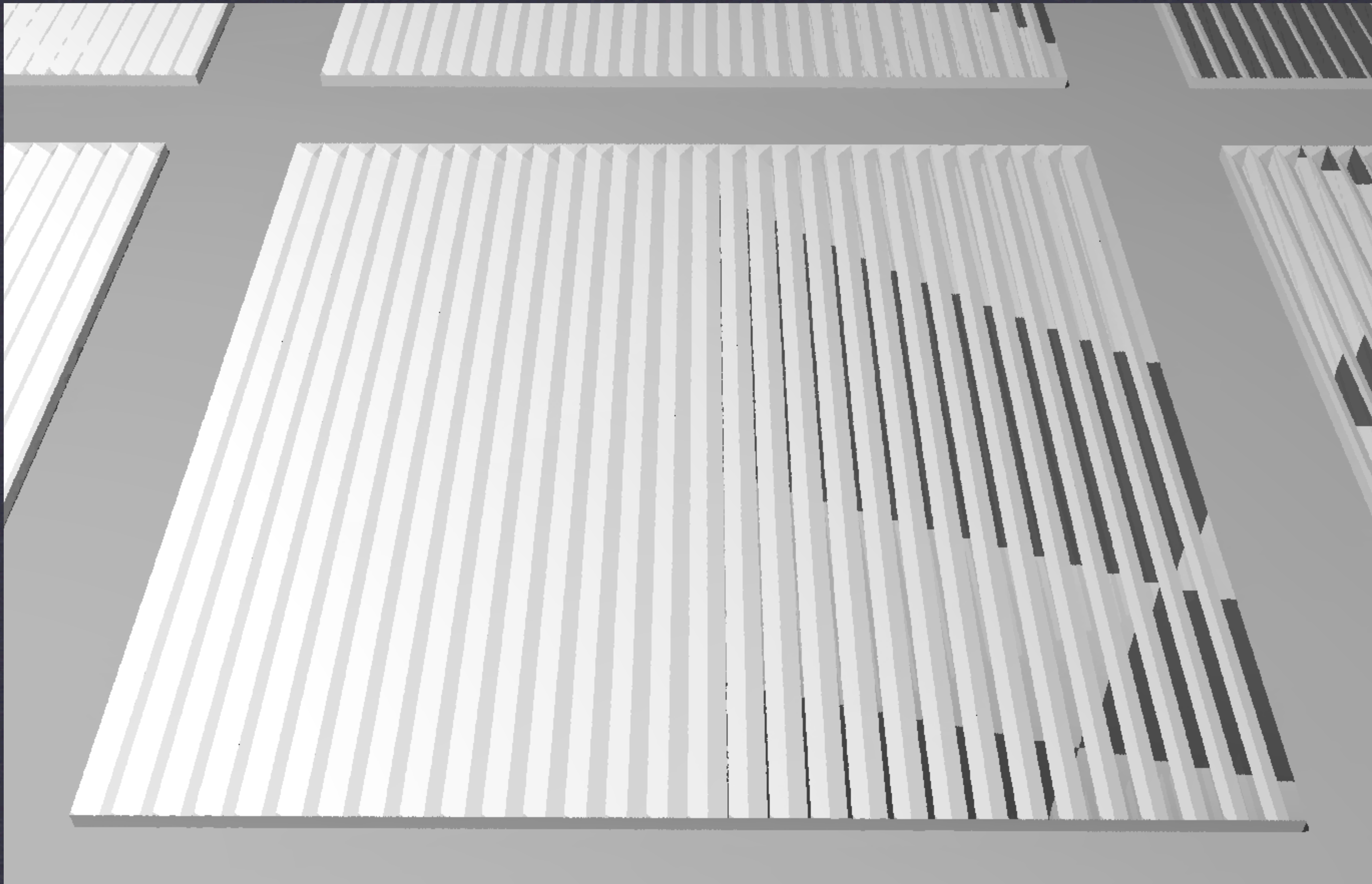

```xml
<?xml version="1.0" encoding="UTF-8"?>
<WindowElement xmlns="http://windows.lbl.gov" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://windows.lbl.gov/BSDF-v1.4.xsd">
<!-- File produced by: genBSDF -t3 4 +mgf -geom meter mirror.mgf -->
<WindowElementType>System</WindowElementType>
<Optical>
<Layer>
      <Material>
            <Name>Name</Name>
            <Manufacturer>Manufacturer</Manufacturer>
            <Thickness unit="meter">0.000</Thickness>
            <Width unit="meter">1.000</Width>
            <Height unit="meter">1.000</Height>
            <DeviceType>Integral</DeviceType>
      </Material>
      <DataDefinition>
            <IncidentDataStructure>TensorTree3</IncidentDataStructure>
      </DataDefinition>
      <WavelengthData>
            <LayerNumber>System</LayerNumber>
            <Wavelength unit="Integral">Visible</Wavelength>
            <SourceSpectrum>CIE Illuminant D65 1nm.ssp</SourceSpectrum>
            <DetectorSpectrum>ASTM E308 1931 Y.dsp</DetectorSpectrum>
            <WavelengthDataBlock>
                  <WavelengthDataDirection>Transmission</WavelengthDataDirection>
                  <AngleBasis>LBNL/Shirley-Chiu</AngleBasis>
                  <ScatteringDataType>BTDF</ScatteringDataType>
                  <ScatteringData>
{ 0.000000e+00 }
                  </ScatteringData>
            </WavelengthDataBlock>
      </WavelengthData>
      <WavelengthData>
            <LayerNumber>System</LayerNumber>
            <Wavelength unit="Integral">Visible</Wavelength>
            <SourceSpectrum>CIE Illuminant D65 1nm.ssp</SourceSpectrum>
            <DetectorSpectrum>ASTM E308 1931 Y.dsp</DetectorSpectrum>
            <WavelengthDataBlock>
                  <WavelengthDataDirection>Reflection Back</WavelengthDataDirection>
                  <AngleBasis>LBNL/Shirley-Chiu</AngleBasis>
                  <ScatteringDataType>BRDF</ScatteringDataType>
                  <ScatteringData>
{
    { 0.000000e+00 }
    { 0.000000e+00 }
    { 0.000000e+00 }
    { 0.000000e+00 }
    { 0.000000e+00 }
    { 0.000000e+00 }
    {
        { 0.000000e+00 }
        {
            { 0.000000e+00 }
            { 0.0000e+00 0.0000e+00 6.5195e+01 0.0000e+00 6.5191e+01 0.0000e+00 0.0000e+00 0.0000e+00 }
            { 0.0000e+00 0.0000e+00 6.5228e+01 0.0000e+00 6.5206e+01 0.0000e+00 0.0000e+00 0.0000e+00 }
            { 0.000000e+00 }
            { 0.000000e+00 }
            { 0.000000e+00 }
            { 0.000000e+00 }
            { 0.000000e+00 }
        }
        {
            { 0.000000e+00 }
            { 0.0000e+00 0.0000e+00 6.5380e+01 0.0000e+00 6.5275e+01 0.0000e+00 0.0000e+00 0.0000e+00 }
            { 0.0000e+00 0.0000e+00 6.8198e+01 0.0000e+00 6.5711e+01 0.0000e+00 0.0000e+00 0.0000e+00 }
            { 0.000000e+00 }
            { 0.000000e+00 }
            { 0.000000e+00 }
            { 0.000000e+00 }
            { 0.000000e+00 }
        }
        { 0.000000e+00 }
        { 0.000000e+00 }
        { 0.000000e+00 }
        { 0.000000e+00 }
        { 0.000000e+00 }
    }
    { 0.000000e+00 }
}
                  </ScatteringData>
            </WavelengthDataBlock>
      </WavelengthData>
</Layer>
</Optical>
</WindowElement>
```


```xml
<DataDefinition>
  <IncidentDataStructure>TensorTree3</IncidentDataStructure>
</DataDefinition>
```

```xml
<WavelengthDataBlock>
  <WavelengthDataDirection>Transmission</WavelengthDataDirection>
  <AngleBasis>LBNL/Shirley-Chiu</AngleBasis>
  <ScatteringDataType>BTDF</ScatteringDataType>
  <ScatteringData>
{ 0.000000e+00 }
  </ScatteringData>
</WavelengthDataBlock>
```

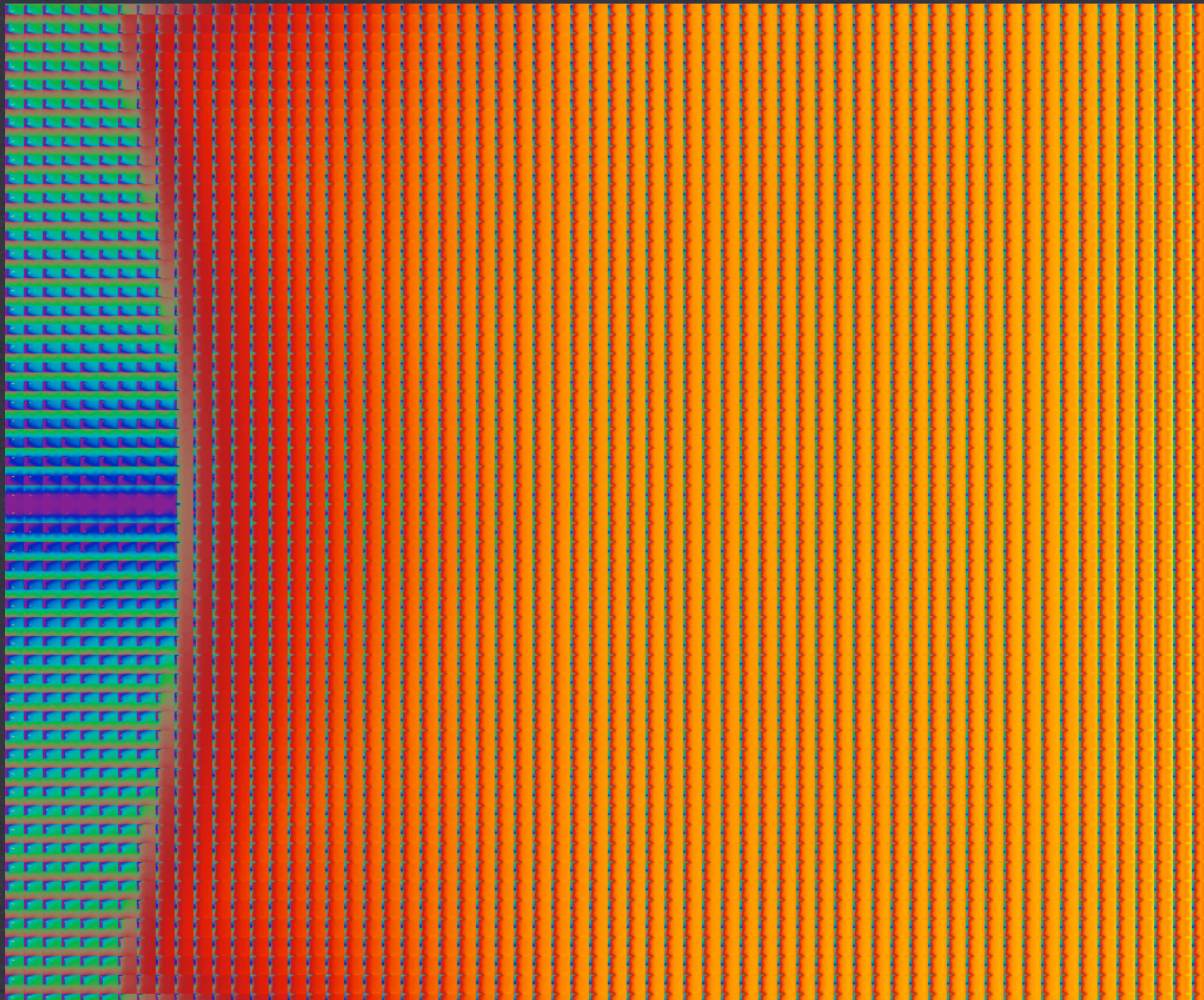
Only 8 non-zero reflectance values corresponding to i/o peaks

```
{ 0.0000e+00 0.0000e+00 6.8198e+01 0.0000e+00 6.5711e+01 0.0000e+00 0.0000e+00 0.0000e+00 }
```



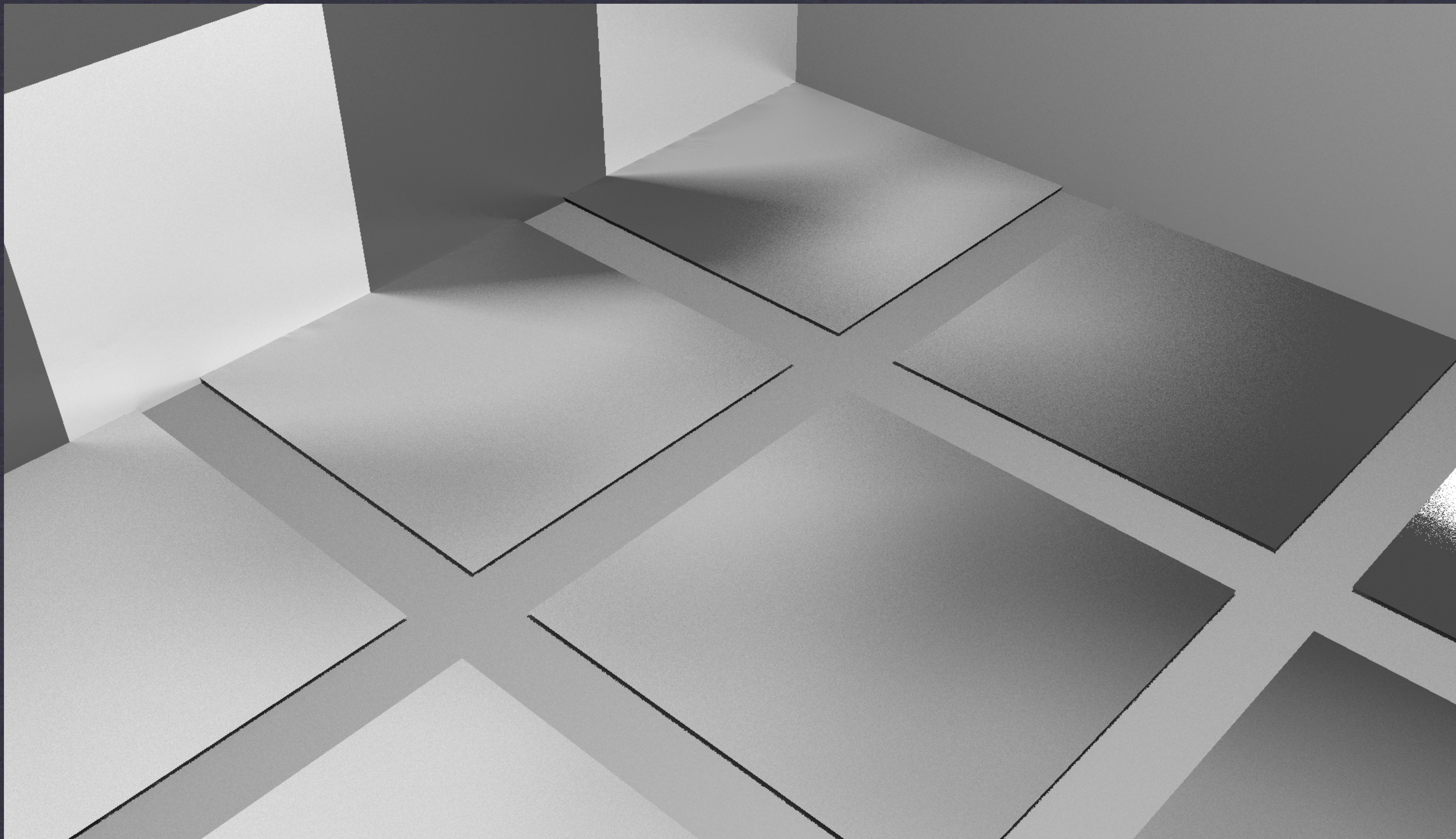
Example from Previous Talk

Sawtooth material with alternating diffuse & mirror elements



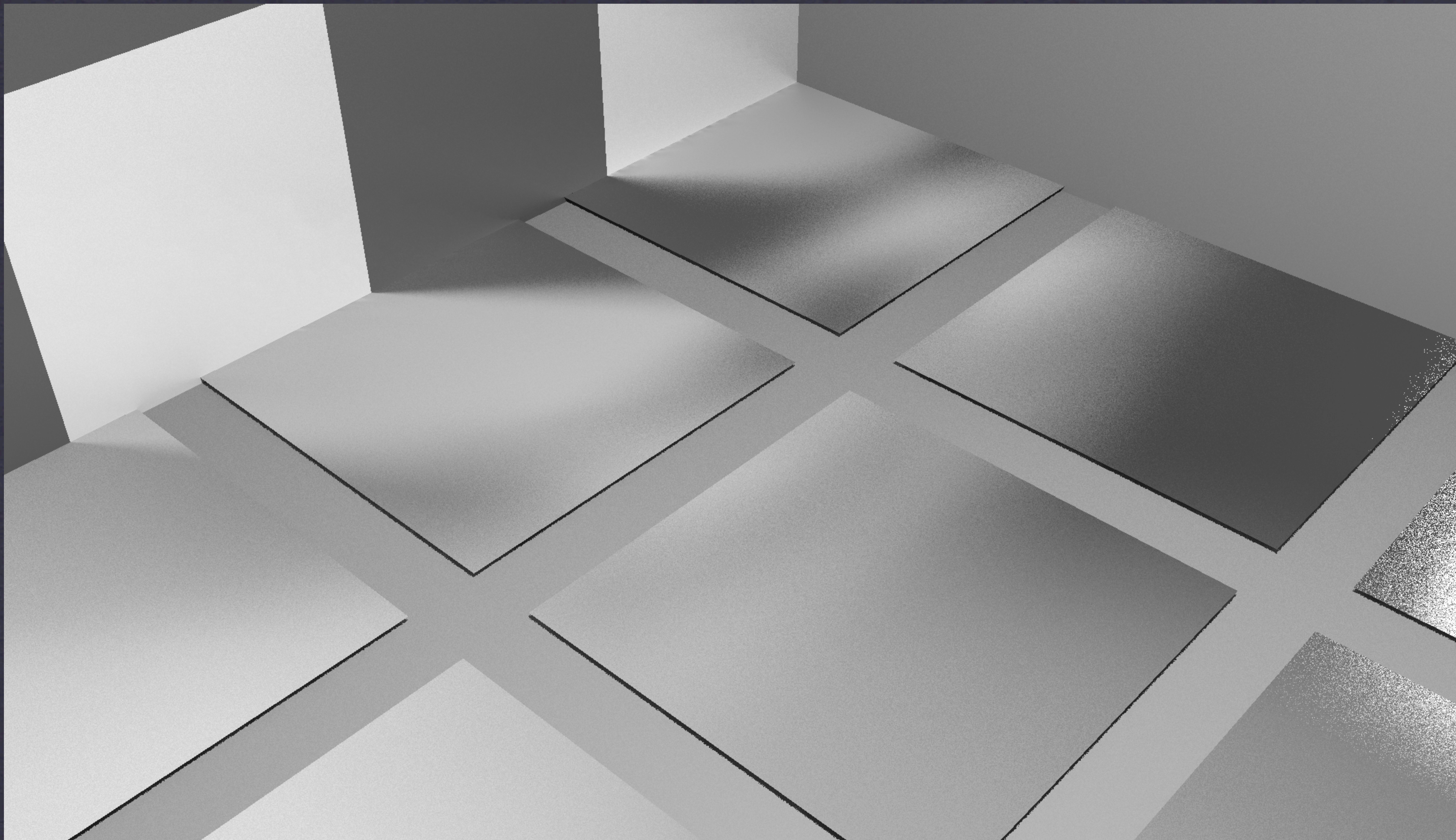
Seeing the Whole BRDF

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



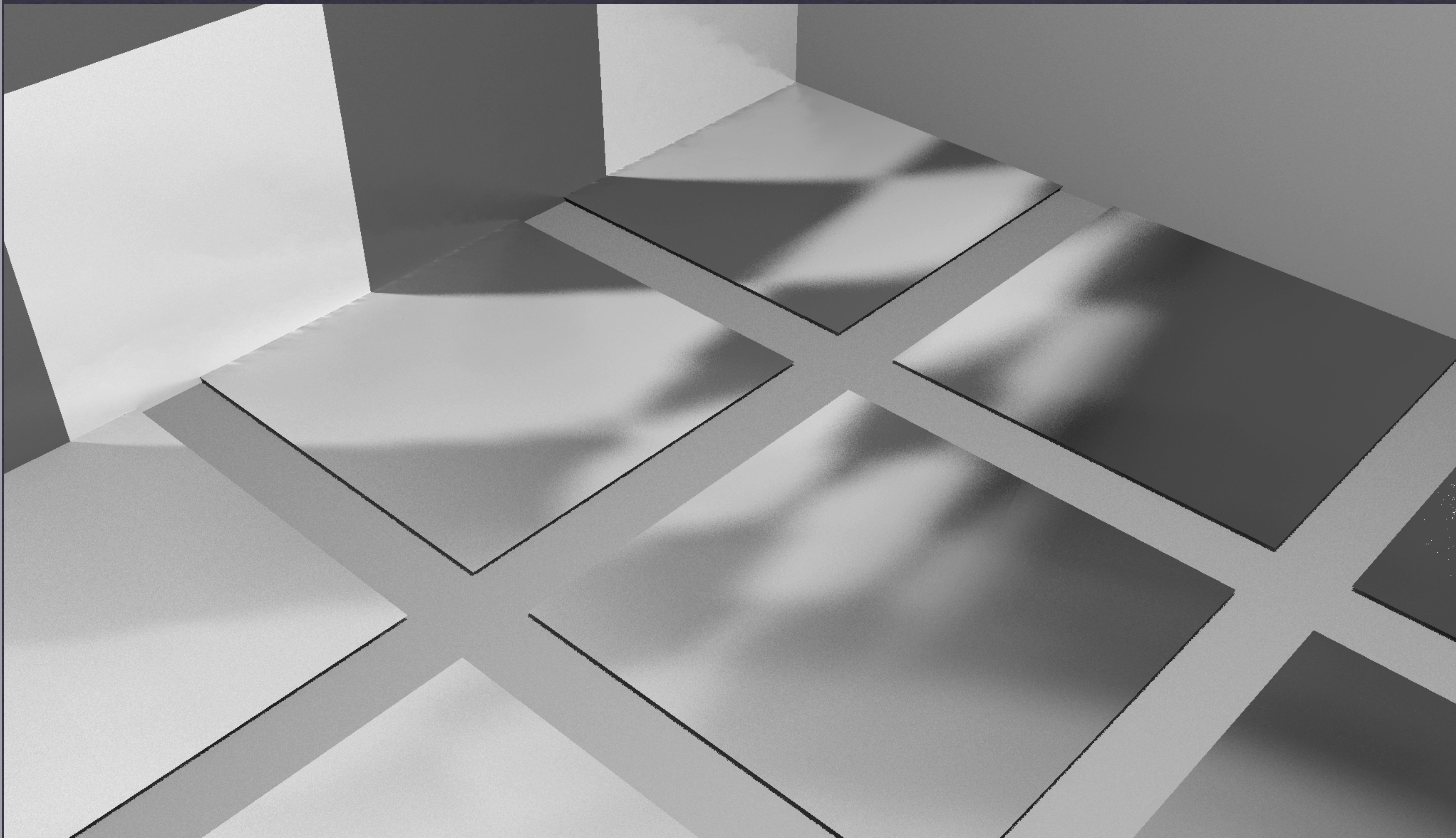
Matrix BSDF

145 incident x 145 exitant directions using Klems coordinates



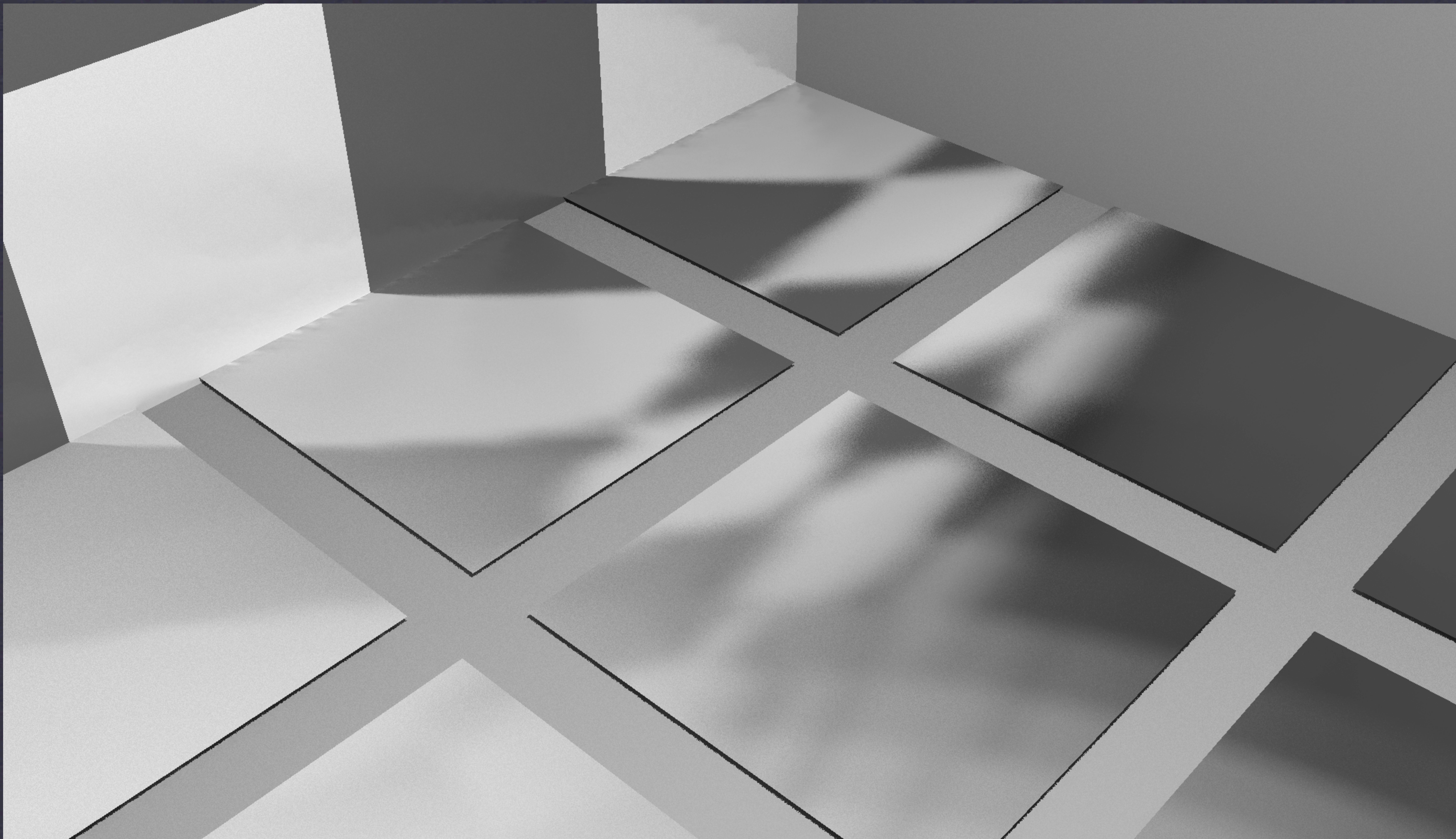
Low-resolution Tensor Tree

Maximum of 256 incident x 256 exitant directions



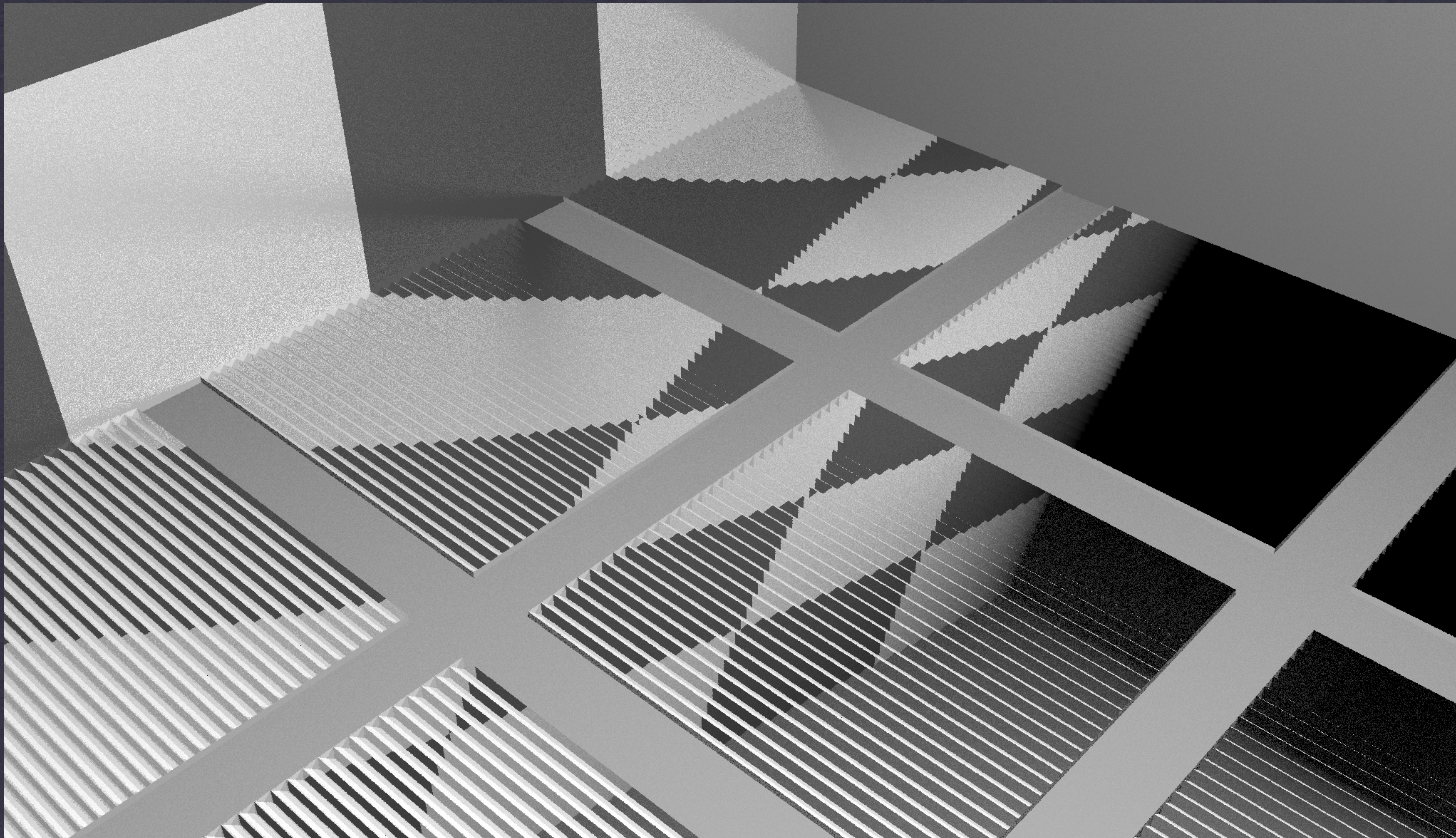
High-resolution Tensor Tree

Maximum of 4096 incident x 4096 exitant directions (400K samples per incident vector)



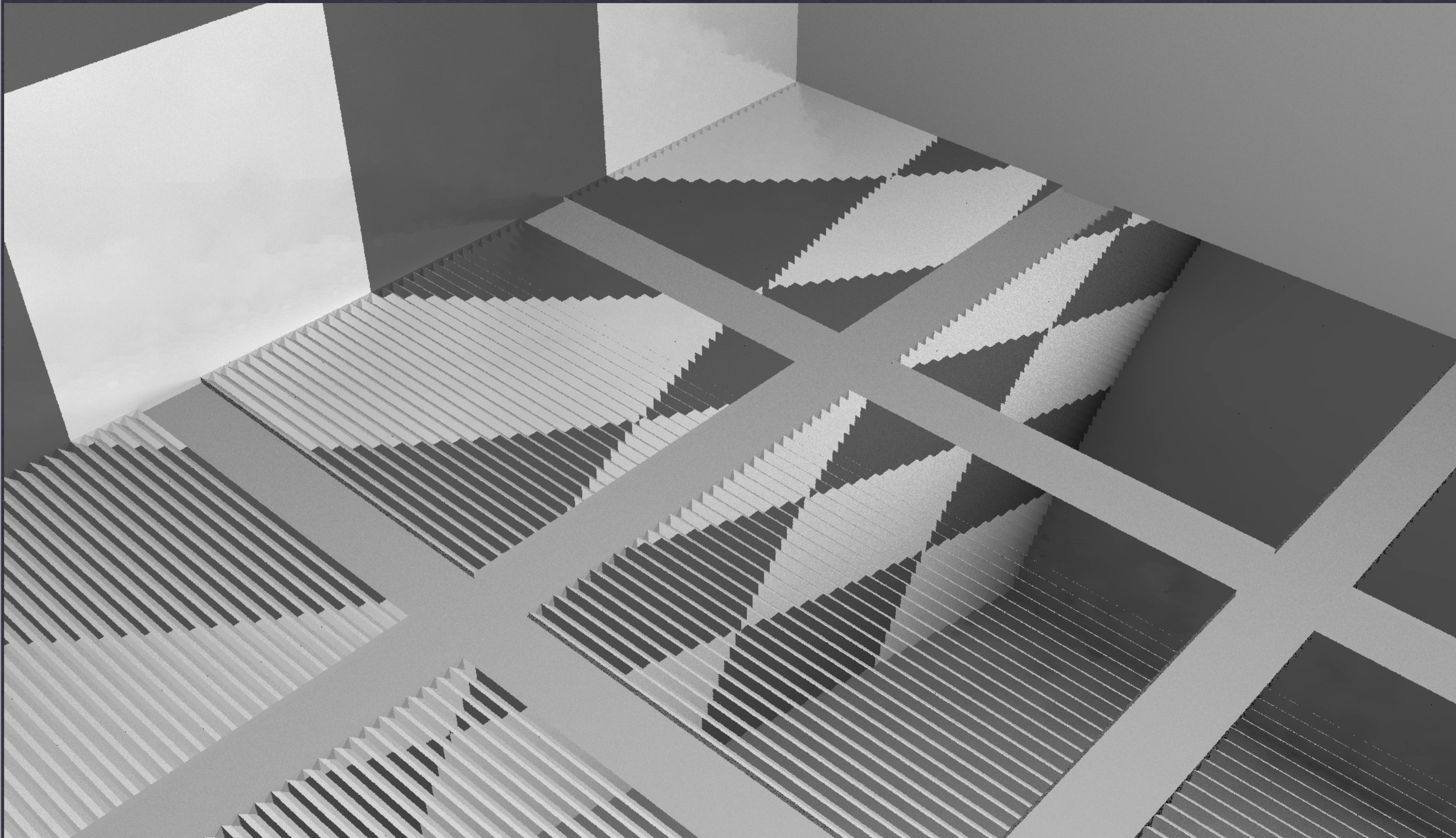
Full-resolution BRDF data

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

Resolution & Type	**genBSDF**	**rpict**
145x145 Klems	6 minutes	23 minutes
16x16 Tensor Tree	6 minutes	21 minutes
4Kx4K Tensor Tree	30 days	21 minutes
4Kx4K Full-res.	30 days	25 minutes

Outstanding Issues

- Higher-resolution BSDFs don’t always translate to better-looking results
  - Difficult to sample highly directional indirect
  - **mkillum** can be used in CFS cases
- Can we use GPU to accelerate **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