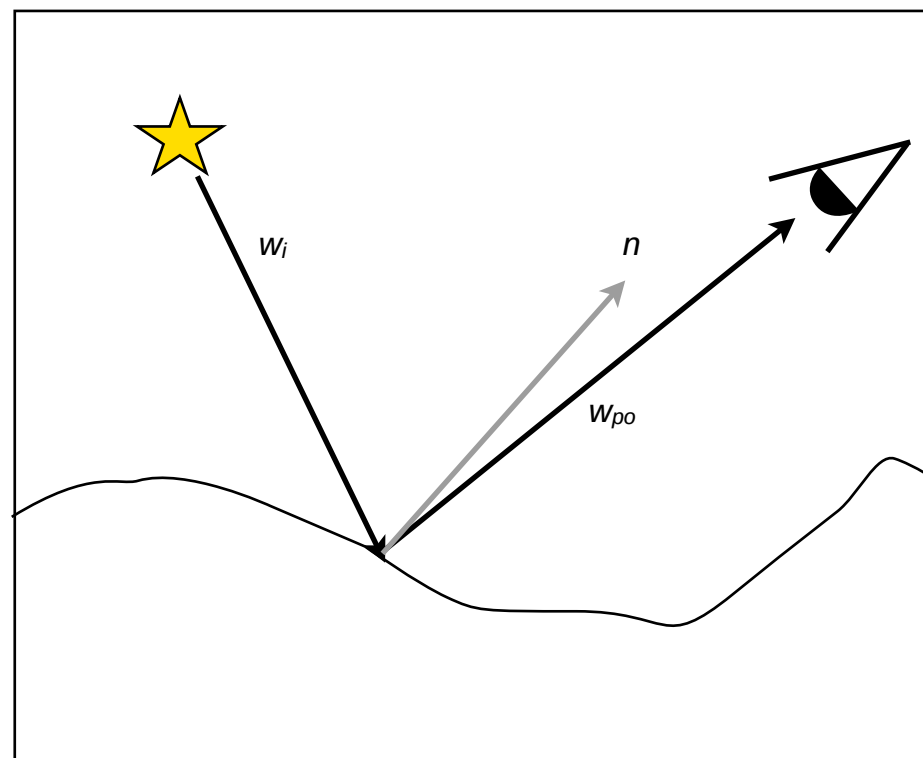


Implementing BRDF in ICC Labs

James Vogh
X-Rite

BRDF for ICC Labs

- **What is a Bidirectional Reflectance Distribution Function (BRDF)?**
A BRDF is a function that specifies the reflectance of a surface for a particular light (position & color) and a viewer (position)



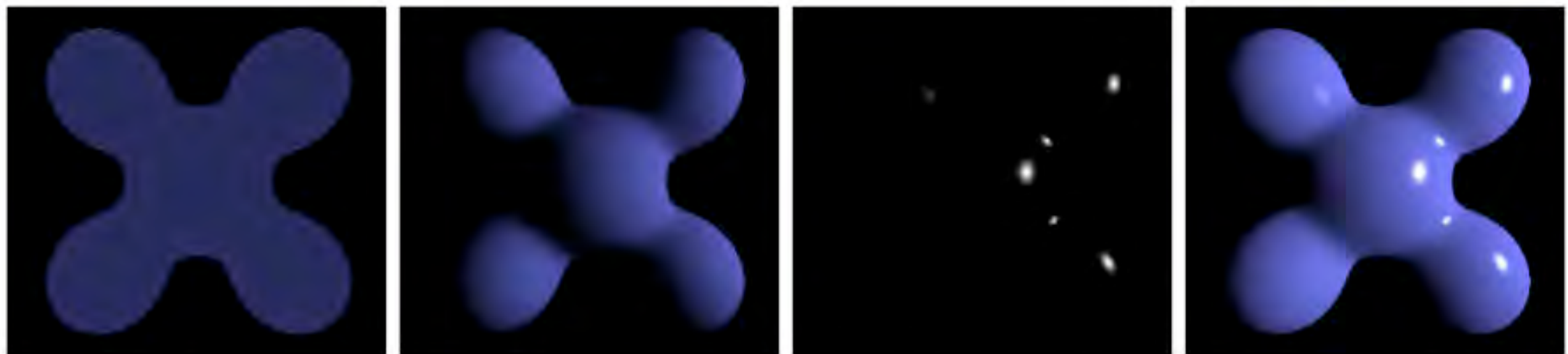
- **A BRDF allows an ICC profile to provide more information about overall appearance**

BRDF Demo

- Compare glossy paper to matte paper

BRDF Models

- Many BRDF models have been developed over the years; can be very simple, can be very complex
- Phong is an early BRDF model



Ambient + Diffuse + Specular = Phong Reflection

- More complex (and accurate models)
—Ward, Lafortune, etc.

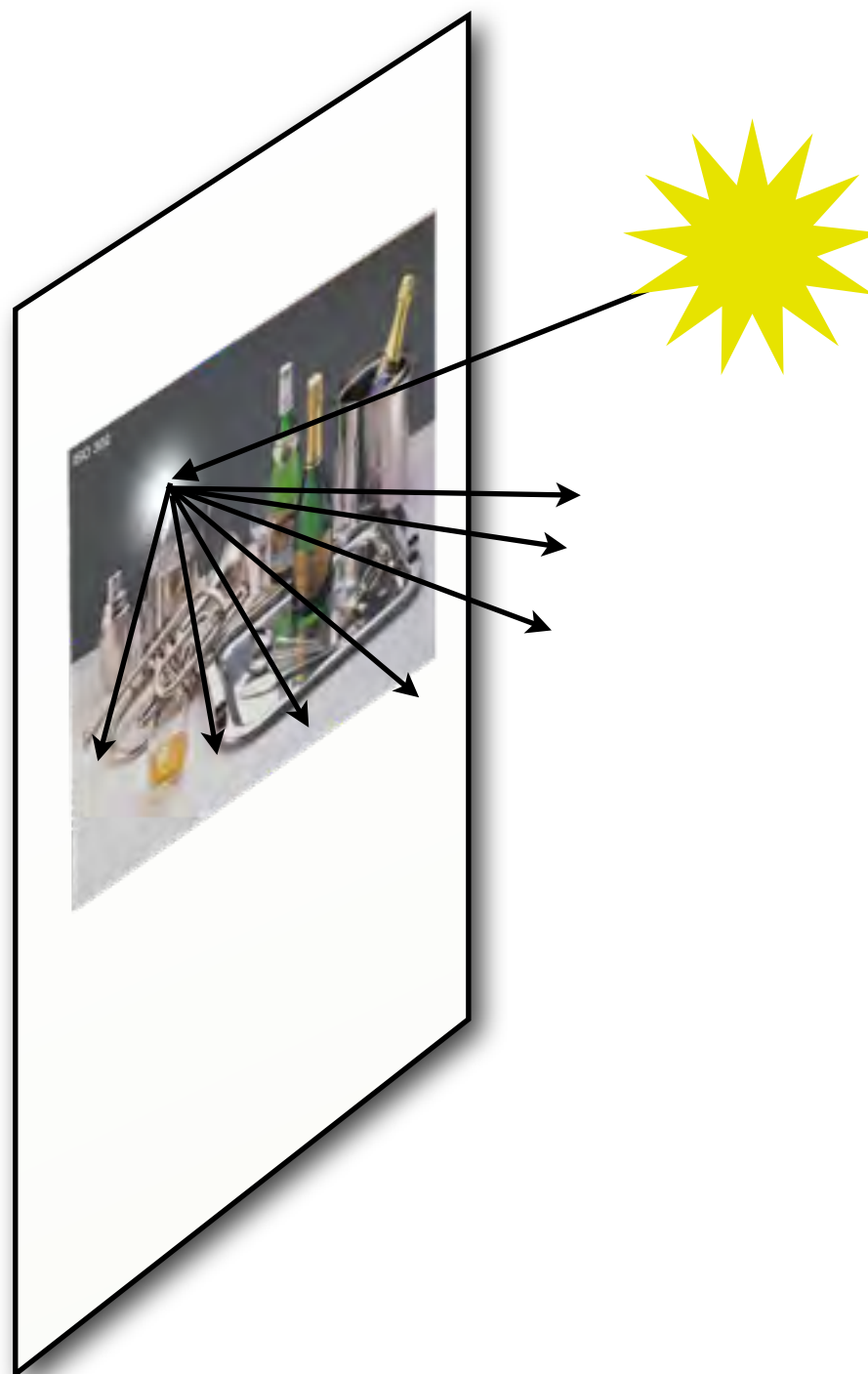
What Does BRDF Do in ICC Labs?

- In this initial implementation BRDF information is allowed for output class profiles
- The BRDF information is in the form of BRDF parameters for the Ward model.
- These parameters are intended to be used by an external 3D rendering engine.

What Does BRDF Do in ICC Labs?



What Does BRDF Do in ICC Labs?



What Does BRDF Do in ICC Labs?



What Does BRDF Do in ICC Labs?



What Does BRDF Do in ICC Labs?



What Does BRDF Do in ICC Labs?

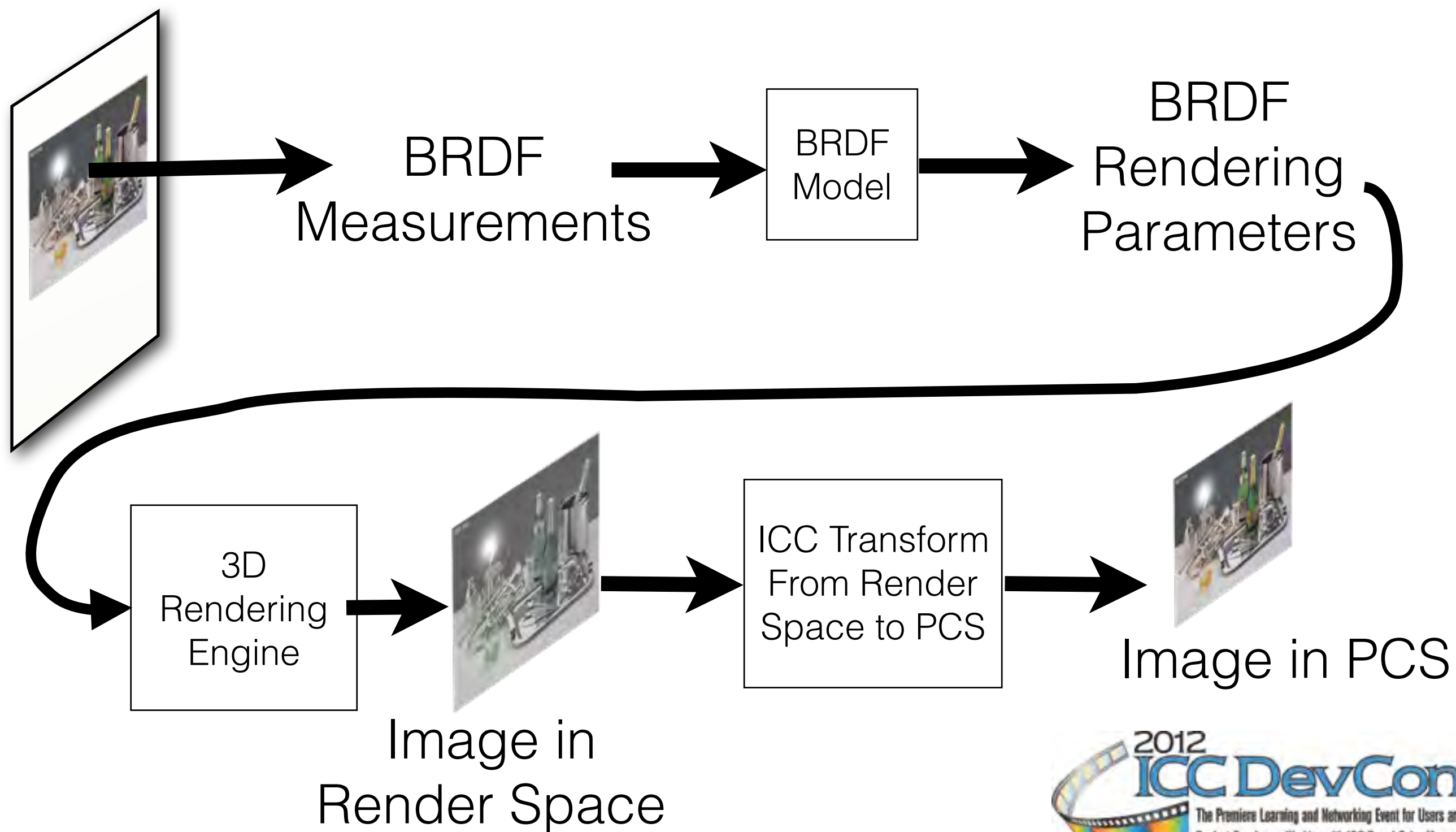


What Does BRDF Do in ICC Labs?



BRDF
Measurements

What Does BRDF Do in ICC Labs?



What Does BRDF Support in ICC Look Like?

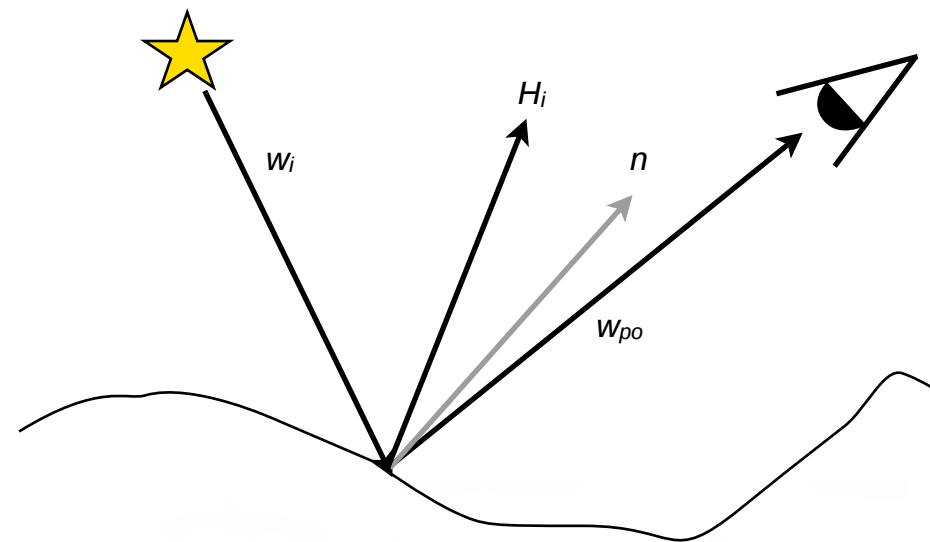
- **Two Basic Types of Transform**
 - **Colorimetric**
 - **Spectral**
- **Profile can contain both types of transforms**
- **Mirrors the implementation of A2B & D2B tags**
 - **Uses same observer and illuminant information**

Color and BRDF

- **BRDF can be implemented to apply per color channel or equally across all color channels.**
- **Some materials can be adequately represented with separate color and BRDF parameters.**
 - Many papers can probably be adequately represented in this manner.
- **Note: The actual implementation is a little more complicated, but the reduction in data is still significant.**

Ward BRDF Model

- Ward is simple and good enough for many applications.



$$\rho_{bd,iso}(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{\rho_d}{\pi} + \rho_s \frac{1}{\sqrt{\cos\theta_i \cos\theta_r}} \cdot \frac{e^{-\frac{\tan^2\delta}{\alpha^2}}}{4\pi\alpha^2}$$

ρ_d = diffuse reflectance

ρ_s = specular reflectance

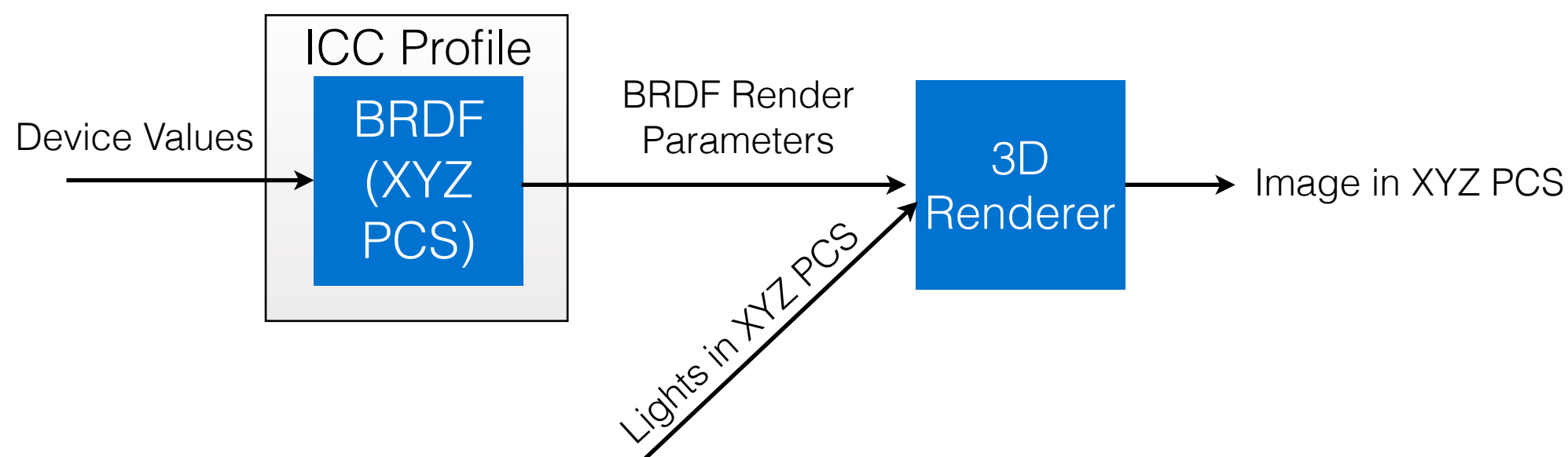
α = width of specular lobe

δ = angle between half vector and normal

Flexible Color Space Support

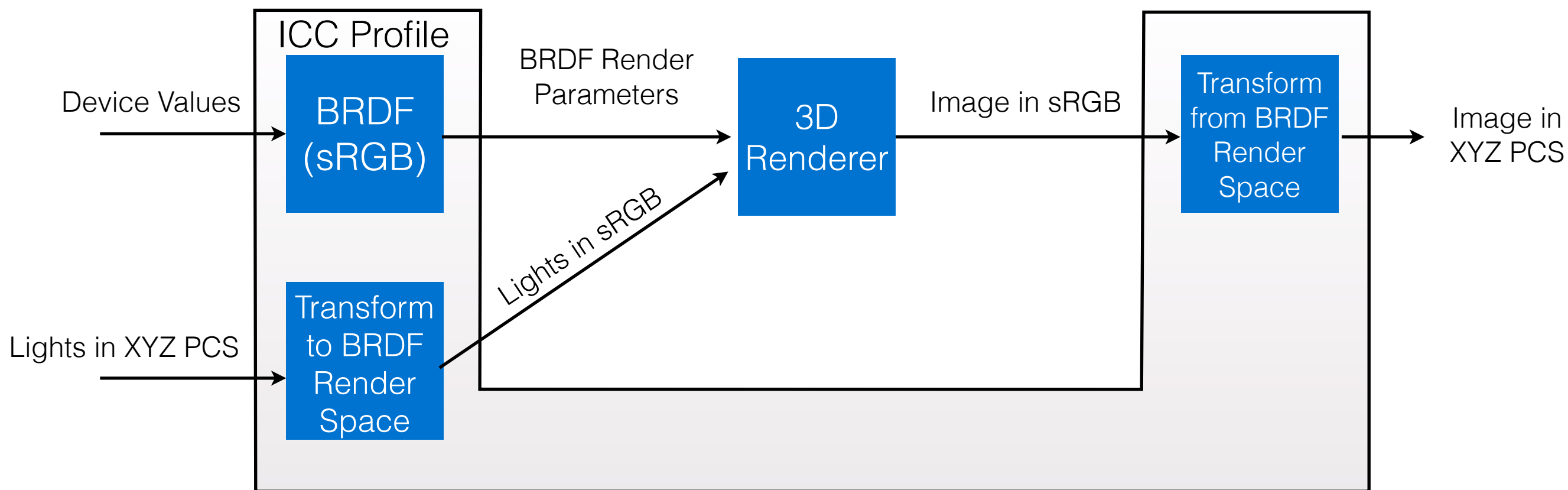
- The BRDF render parameters don't have to yield XYZ PCS values when the BRDF model is applied.
- A separate transform can convert the output of the BRDF model to PCS.
- A 3D rendering engine can render an image in BRDF render space and then convert it to PCS with the separate BRDF render space to PCS transform.
- The lights (or environment map, etc.) Must be converted from PCS to the BRDF render space with a transform that goes from PCS to BRDF render space.
- Permits flexibility in how BRDF parameters are selected.
- Allows for dimensional reduction for spectral.

BRDF Space to PCS Conversions Example: BRDF rendering space is XYZ PCS



BRDF Render Space to PCS Conversions

Example: BRDF rendering space is sRGB



Implementation Details

- **BRDF is implemented as a tag structure**
 - **The BRDF tag structure contains the following required sub-tags**
 - `icSigTypeBrdfMember`
 - Signature type, specifies the type of BRDF.
 - `icSigBRDFTTypeWardIsotropic` & `icSigBRDFTTypeWardAnisotropic` are current possible values.
 - `icSigFunctionBrdfMember`
 - Signature type, specifies if BRDF is full color or global.
 - `icSigBRDFFunctionMonochrome` & `icSigBRDFFunctionColor` are current possible values.
 - `icSigNumParamsBrdfMember`
 - The number of output values from the BRDF transform.
 - `icSigNumRenderSpace`
 - `icSigTransformBrdfMember`
 - An MPE tag that returns the BRDF render parameters.
 - **The following are optional tags**
 - `icSigPCSToBRDFTTransformBrdfMember`
 - An MPE tag that transforms from PCS to BRDF render space.
 - Used for lights, etc. Also used to transform A2B output for BRDFs
 - `icSigBRDFTToPCSTransformBrdfMember`
 - An MPE tag that transforms from BRDF render space space to PCS.

Full Color Ward BRDF

- Individual Ward parameters for each device value

$$\rho_{bd,iso}(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{\rho_d}{\pi} + \rho_s \frac{1}{\sqrt{\cos\theta_i \cos\theta_r}} \bullet \frac{e^{\frac{-\tan^2\delta}{\alpha^2}}}{4\pi\alpha^2}$$

- ρ_d and ρ_s become vectors with length equal to the number of output channels
- For XYZ space there are 7 parameters
 - $\rho_{d,X}$ $\rho_{d,Y}$ $\rho_{d,Z}$ $\rho_{s,X}$ $\rho_{s,Y}$ $\rho_{s,Z}$ α

Global Ward BRDF

- One set of device values for all device values
- Good enough for many applications
- The complete set of parameters used is:
 - The global BRDF parameters
 - The PCS values for a device value that are obtained from the A2B transform. Transformed through light transform if BRDF is not PCS.
- For XYZ the parameters are:
 - From the A2B transform:
 - X_{A2B} Y_{A2B} Z_{A2B}
 - From the BRDF Transform
 - α A_d B_s X_s Y_s Z_s
 - $\rho_d = A_d(X_{A2B}, Y_{A2B}, Z_{A2B})$
 - $\rho_s = B_s(X_{A2B}, Y_{A2B}, Z_{A2B}) + (X_s, Y_s, Z_s)$

Implementation Example: Create BRDF Tag

- Show how to add BRDF tag to an existing profile using the ReflccLabs C++ API

```
CiccProfile *pIcc;  
  
pIcc = ReadIccProfile(argv[nArg]);  
  
// create the tag  
CiccTagStruct* tagStruct = new CiccTagStruct;  
  
// create the tag parts  
// Type tag  
CiccTagSignature* brdfTypeTag = (CiccTagSignature*)CiccTagCreator::CreateTag(icSigSignatureType);  
brdfTypeTag->SetValue(icSigBRDFTTypeWardIsotropic);  
tagStruct->AttachElem(icSigTypeBrdfMember, brdfTypeTag);  
  
// Function tag  
CiccTagSignature* brdfFunctionTag = (CiccTagSignature*)CiccTagCreator::CreateTag(icSigSignatureType);  
brdfFunctionTag->SetValue(icSigBRDFFunctionColor);  
tagStruct->AttachElem(icSigFunctionBrdfMember, brdfFunctionTag);  
  
// number of parameters tag  
CiccTagUInt32* brdfNumParamTag = (CiccTagUInt32*)CiccTagCreator::CreateTag(icSigUInt32ArrayType);  
(*brdfNumParamTag)[0] = 8;  
tagStruct->AttachElem(icSigNumParamsBrdfMember, brdfNumParamTag);
```

Implementation Example: Create BRDF Tag (Cont.)

```
// The BRDF parameters
const int numParams = 7;
CIccTagMultiProcessElement* brdfParamsTag = new CIccTagMultiProcessElement(icGetSpaceSamples(
                                                                    pIcc->m_Header.colorSpace), numParams);

CIccBasicMpeFactory mpeFactory;
CIccMpeCLUT* lut = (CIccMpeCLUT*)mpeFactory.CreateElement(icSigCLutElemType);
CIccCLUT* clut = new CIccCLUT(icGetSpaceSamples(pIcc->m_Header.colorSpace), numParams);
clut->Init(2);
for (int i=0; i<16; i++)
{
    (*clut)[i*numParams] = printerXYZ[i*3];
    (*clut)[i*numParams+1] = printerXYZ[i*3+1];
    (*clut)[i*numParams+2] = printerXYZ[i*3+2];
    (*clut)[i*numParams+3] = 0.267; // a
    (*clut)[i*numParams+4] = 0.19; // specular_X
    (*clut)[i*numParams+5] = 0.19; // specular_Y
    (*clut)[i*numParams+6] = 0.19; // specular_Z
}
lut->SetCLUT(clut);
brdfParamsTag->Attach(lut);
tagStruct->AttachElem(icSigTransformBrdfMember, brdfParamsTag);

// Add the struct to the profile
pIcc->AttachTag(icSigBRDFColorimetric1Tag, tagStruct);

// save the profile
SaveIccProfile(argv[2], pIcc);
```

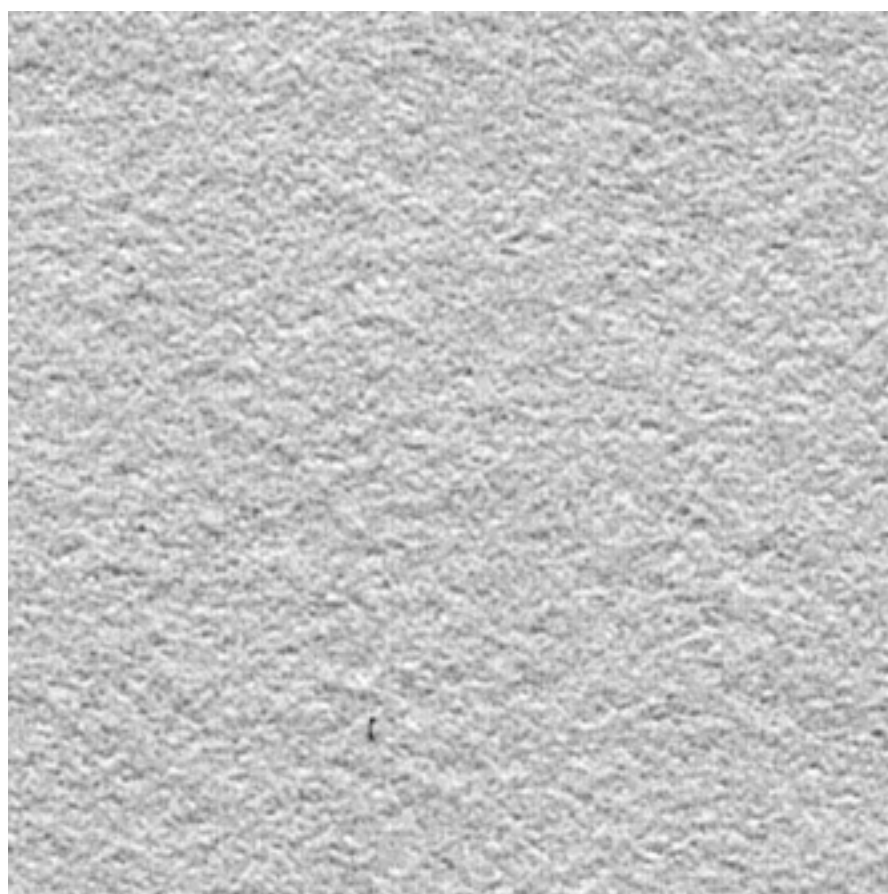
Implementation Example: Use BRDF Tag

- Show how to use ReflICCLab to get BRDF parameters

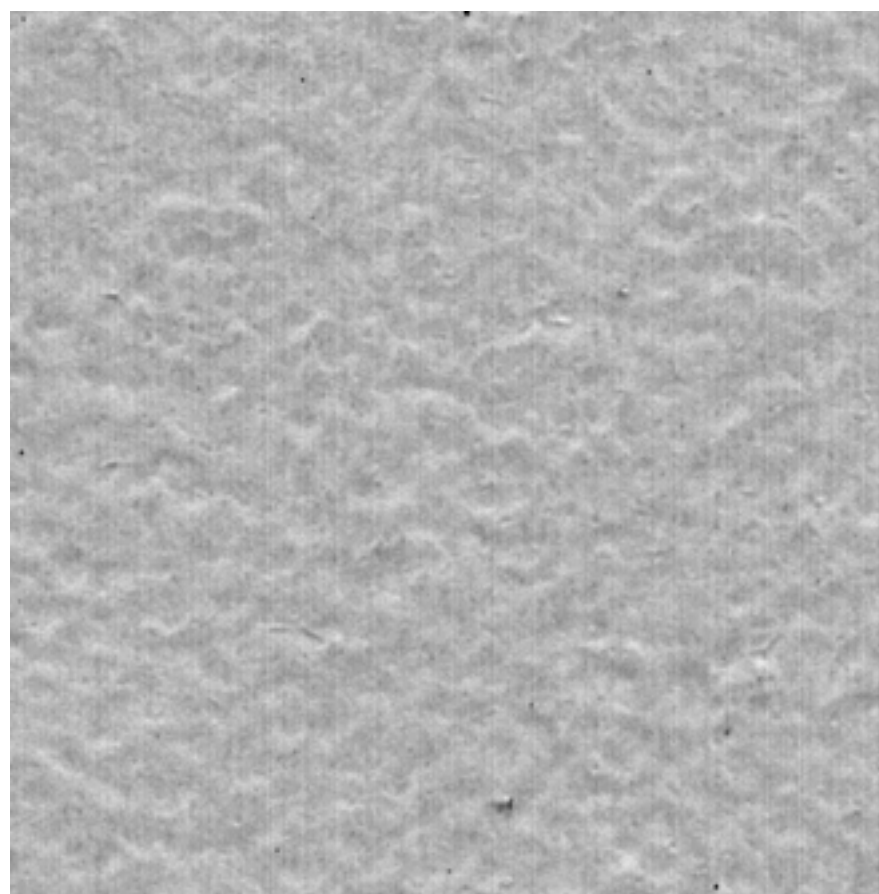
```
pIcc = ReadIccProfile(argv[2]);  
  
// apply the profile to get BRDF info  
CIccCmm cmm;  
  
cmm.AddXform(pIcc, icRelativeColorimetric, icInterpLinear,  
            NULL, icXformLutBRDFModel, false);  
  
cmm.Begin();  
  
icFloatNumber outNum[numParams];  
icFloatNumber inNum[] = {0,0,0,0};  
  
cmm.Apply(outNum, inNum);
```

Texture

- BRDF doesn't provide information about the texture of a substrate.
- Example substrate textures (enhanced contrast)



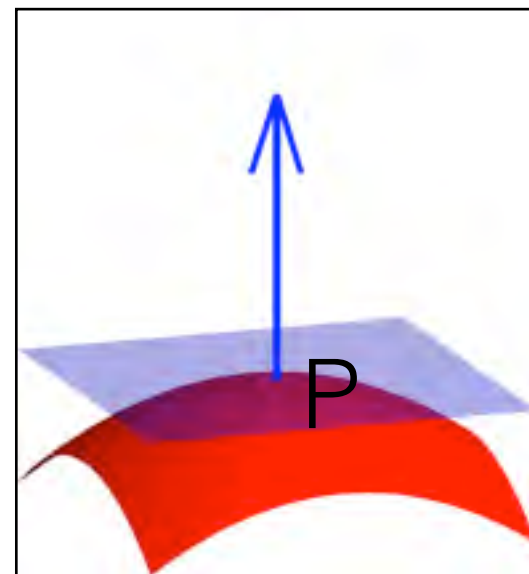
Matte Paper



Premium Luster Paper

Normal Map

- The surface texture of a substrate can be represented with a normal map
 - A surface normal is the vector that is perpendicular to the tangent plane of a surface at point P



- A normal map is a set of surface normals across a surface
- Normal map represents how the normal varies across the surface

Normal Map

- In ICC labs a normal map is optional, even if the profile contains BRDF information.
- Stored as a 2D array of vectors
 - Contains spatial dimensions of the map for correct rendering
 - Can be compressed



Demo

Normal Map Consideration

- The normal map and the BRDF aren't independent from each other.
- Notice how the specular lobe is enlarged by the normal map.
- The BRDF parameters and normal map should be calculated together to get the correct results.

Future Directions

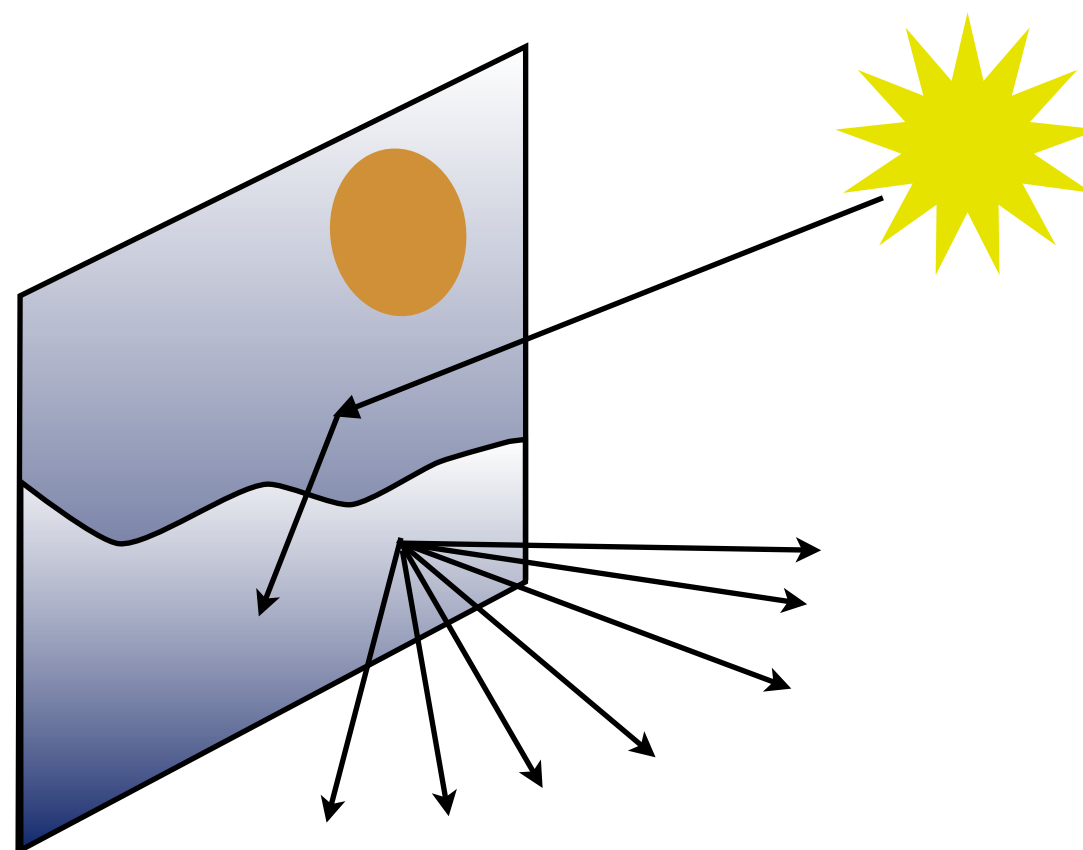
- This is a first step in putting more appearance information in a profile.
- Extended support can come in future versions.

Types of BRDF Models

- This initial implementation uses the Ward model.
- Additional models can be added.
- BRDF information can also be in the form of measurement data
 - The amount of data might be very large.
- May want to establish a BRDF registry where third parties could specify BRDF models and allow others to use them.

What about Other Types of Profiles Such as Display Devices?

- Can characterize viewing angle behavior of LCD
- Can characterize how light reflects off the surface of an LCD



BRDF PCS?

- Do not have much control over appearance with current output devices. Can control gloss layer on some devices.
- Complex BRDF models would be very difficult to support.
- Currently an item for future consideration.

Thank You!

Questions?