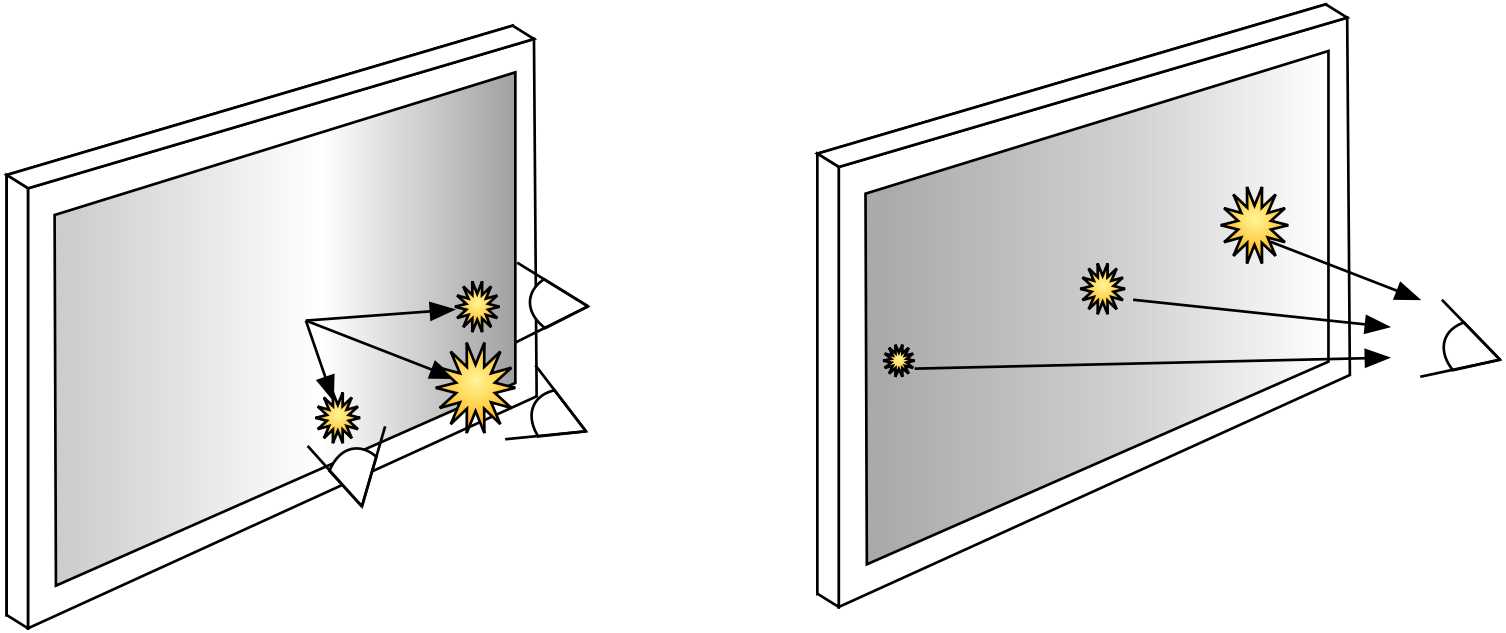
Correction of Display Viewing Angle with iccMAX

James Vogh
X-Rite

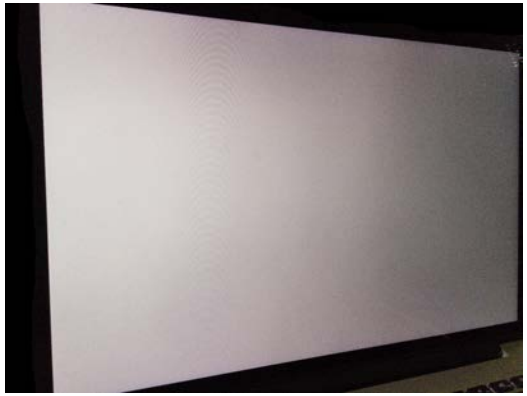
Angular Behavior of LCD Display

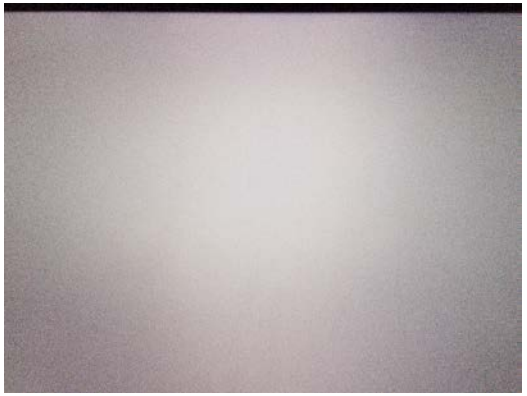
Intensity and color of light from LCD display typically varies by viewing angle



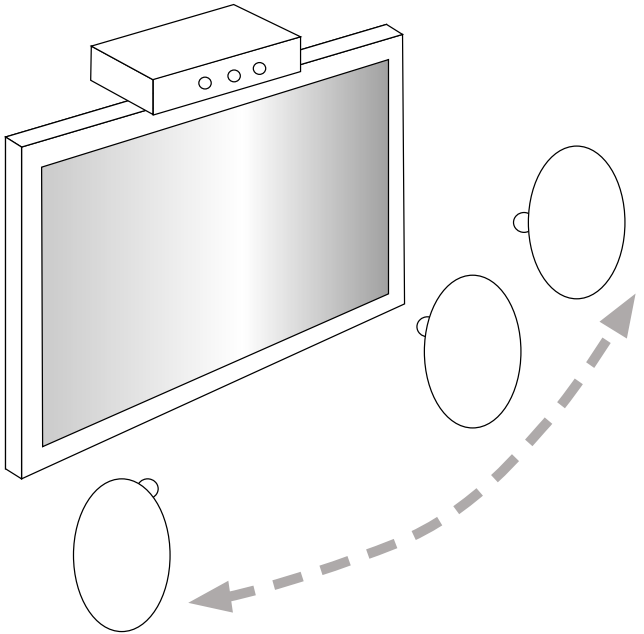
Viewing Angle Demonstration

Brightest part of screen changes as viewing position changes





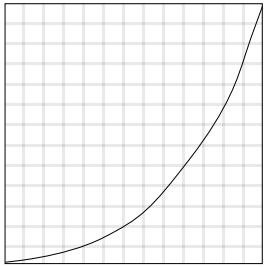
Using face tracking

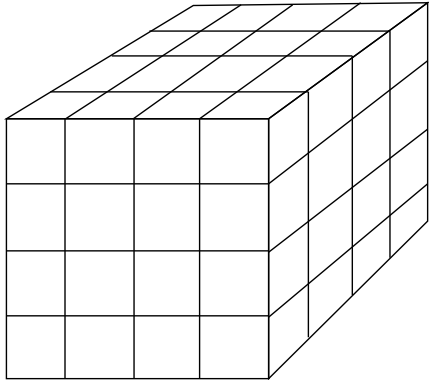


ICC v4 Display Profile

- Has no support for angular characterization
- Typically assumes angle perpendicular to the screen
- Profile is typically Matrix/TRC or LUT based

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

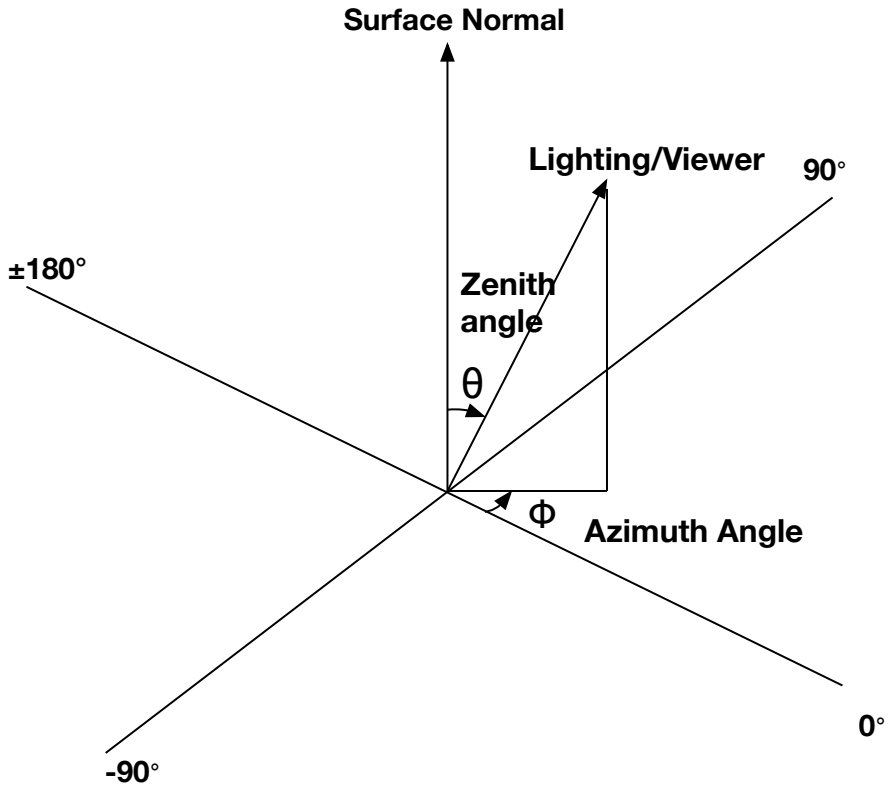




iccMAX

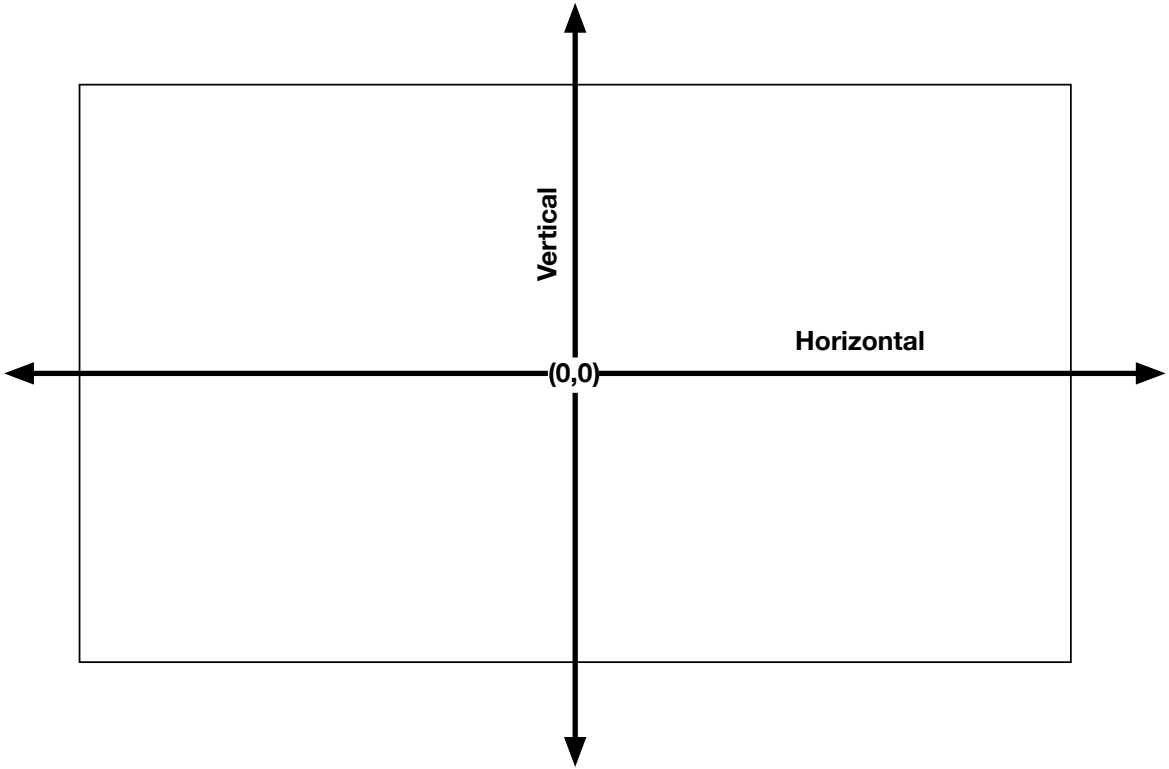
- Adds support for directional AToB, DtoB, BtoA, and BToD tags.
- Mirrors functionality of AtoB, DtoB, BtoA, and BToD tags with additional inputs of:
  - Horizontal and vertical location
  - Azimuth and Zenith specify viewing angle to surface
- Implemented as multiProcessElementType
  - May contain any valid component including LUTs and calculator elements

Azimuth & Zenith



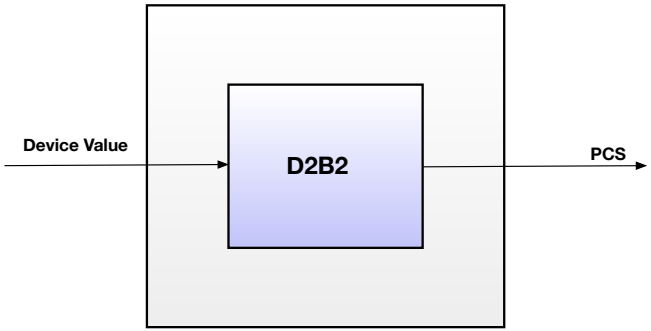
REAL WORLD SOLUTIONS & APPLICATIONS

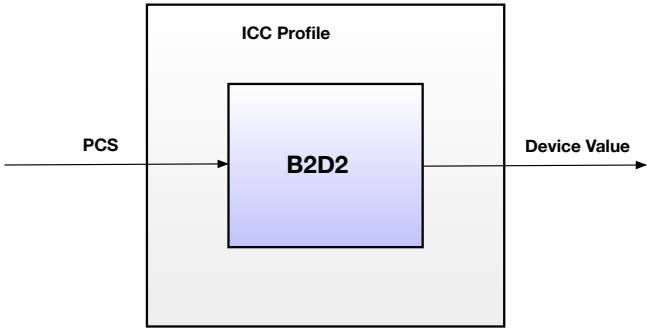
Horizontal & Vertical



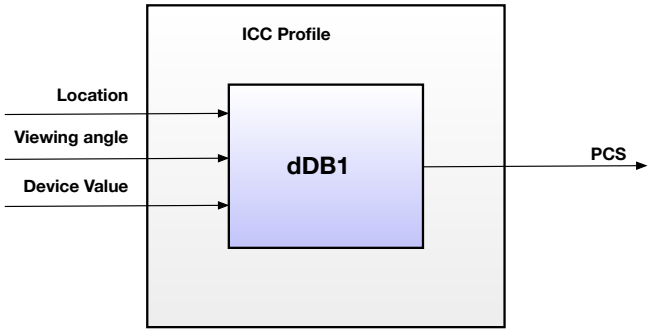
Non-Directional and Directional tags

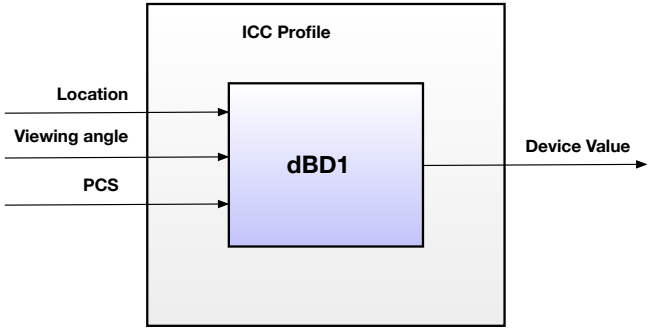
Non-directional





Directional





Screen Uniformity

Directional tags can also correct screen uniformity

(Contrast enhanced)

Directional Tags in iccMAX

- A single profile can correct viewing angle and screen uniformity
- Directional tags can correct for viewing angle when combined with information on viewer location
  - Viewing angle combinations that produce the brightest image can be scaled down to match darker viewing angle combinations so that all viewing angle combinations produce the same brightness.
  - Since multiprocessing tags are used, can be very complex and correct for differences in gamma, primary chromaticities, etc. that might vary by angle.

Directional Tags in iccMAX

- Location information can be used to correct for non-uniformity in the screen.
  - Doesn’t require face tracking
  - Can be high resolution
  - Can correct for spatial differences in
    - Gamma
    - White point
    - Chromaticites
    - Black point
    - etc.

Demonstration of the Creation of a Simple Angular Display Profile

- Will cover the steps of
  - Measuring the display
  - Creating the iccMAX tags
  - Applying the tags with an OpenGL shader

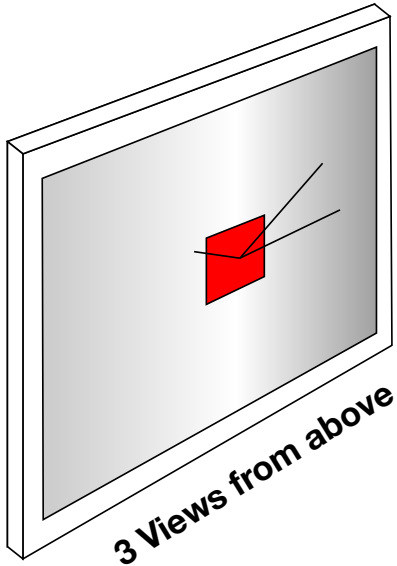
The profile demonstrated here is not meant to be the highest quality profile.

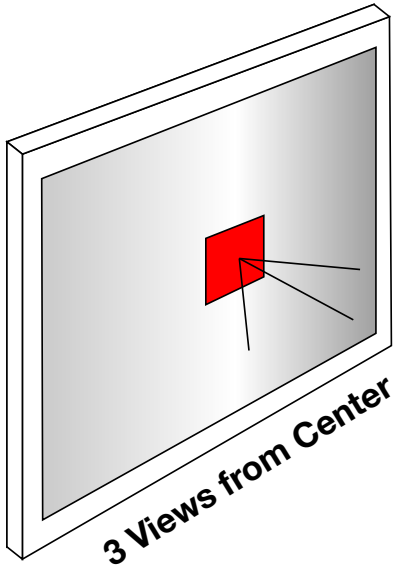
Measurement Process

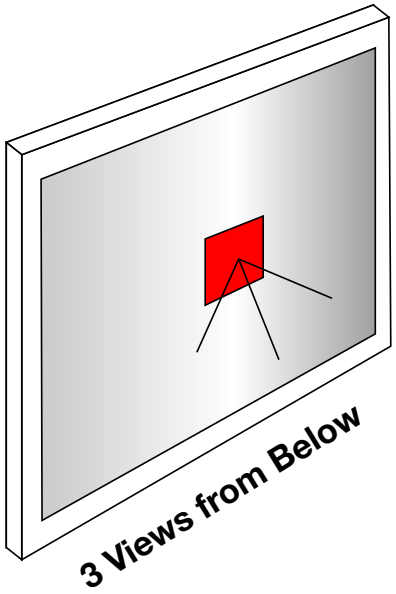
Typical display profiles are created by measuring a set of color patches with measurement device and computing a profile from the measurements

The angular profile measures the same set of colors, but measures the colors from different angles

9 Measurement Angles







iccMAX Profile

- The profile will have the normal A2B and B2A tags that a typical display profile would have.
  - These tags will be matrix TRC profiles built from the zero angle measurements
- The profile will also have the directional A2B & B2A tags.
- Directional tags can be Colorimetric and/or specular. The profile for this demonstration will be Colorimetric
- Profiles can be for different adapted white points and observers. This demonstration profile will be D50/2deg

iccMAX Profile Directional B2A tags

- The B2A tags will be designed to maintain brightness for all typical viewing angles of the display
  - Brightness of angle 0,0 will be decreased in order to maintain consistent brightness.
  - For very steep viewing angles, the brightness may not be maintained since the maximum brightness at that angle from the display may be too low
    - Software that was utilizing the profile could provide the equivalent of a out of gamut warning for these viewing angles
  - For a display with poor viewing angle behavior, a compromise between consistent brightness and sufficient overall brightness will need to be found

iccMAX Profile Directional A2B Tag

- The A2B directional tags specify what the display will look like from any viewing angle
- Inputs to the tag are full device values

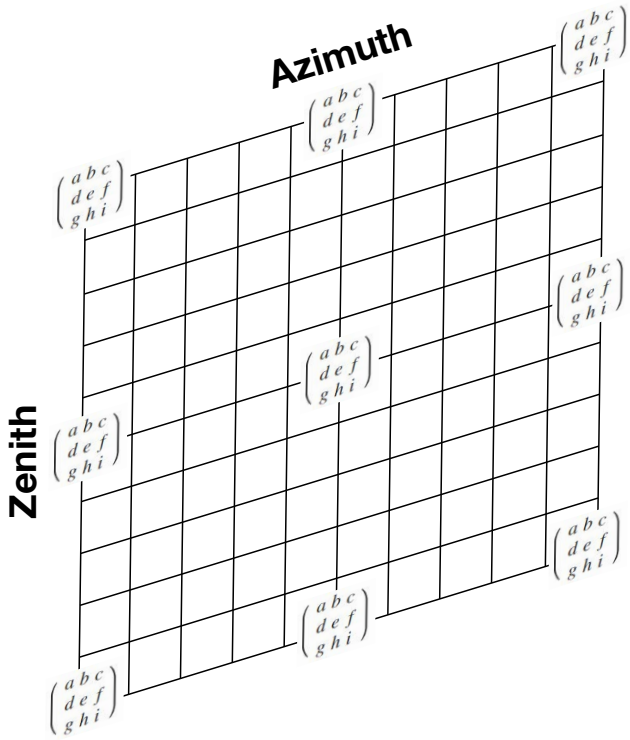
Construction of iccMAX Directional Tags

- The multiprocessing element gives a great deal of flexibility in the implementation
- The implementation chosen for this demonstration uses a 3D LUT that implements tone curves and a 2D table of matrix values.
    - The 2 LUTs are used in conjunction with a few simple calc elements
- The elements of the Tags can easily be extracted and used by a GPU based code
- Both A2B and B2A tags are created

Creation of Profile

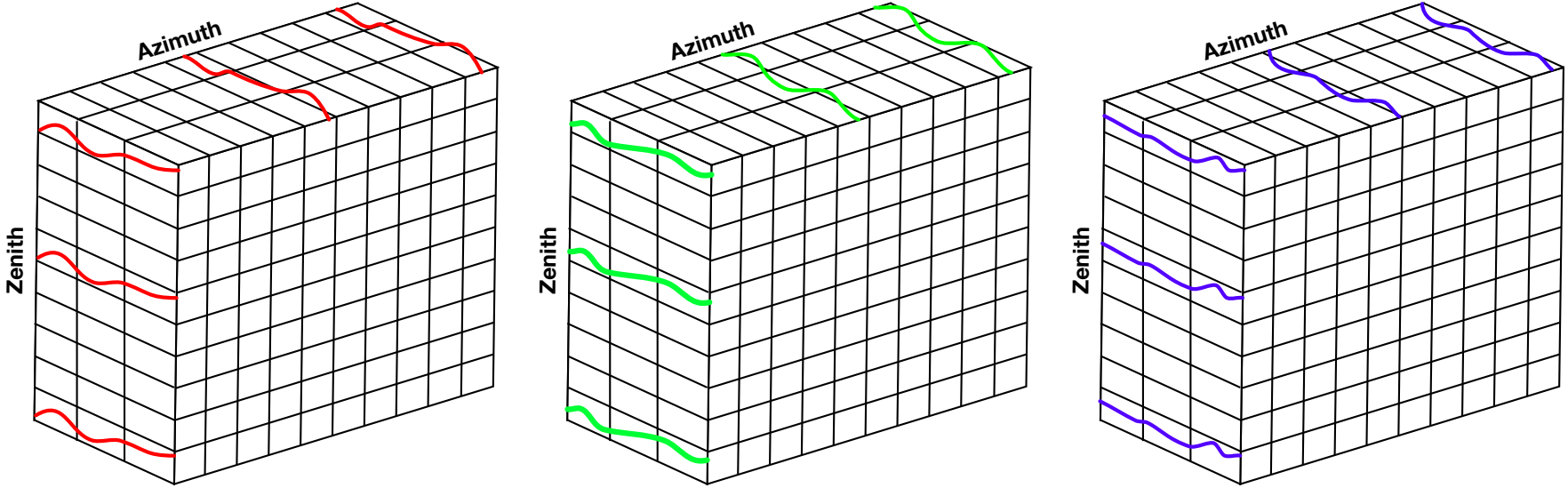
- Interpolate the 9 sets of measurements to create a 15x15 array of measurements
- For each of 225 measurements calculate the max brightness that is achievable. Set this brightness as the overall screen brightness
- Create 225 Matrix/TRC profiles with the desired screen brightness.
    - Brightness level is set with the B2A tone curve

Creation of matrix LUT



Creation of Tone Curve LUT

- LUT is stored as 3 3D LUTs. R,G,B are all stored separately.



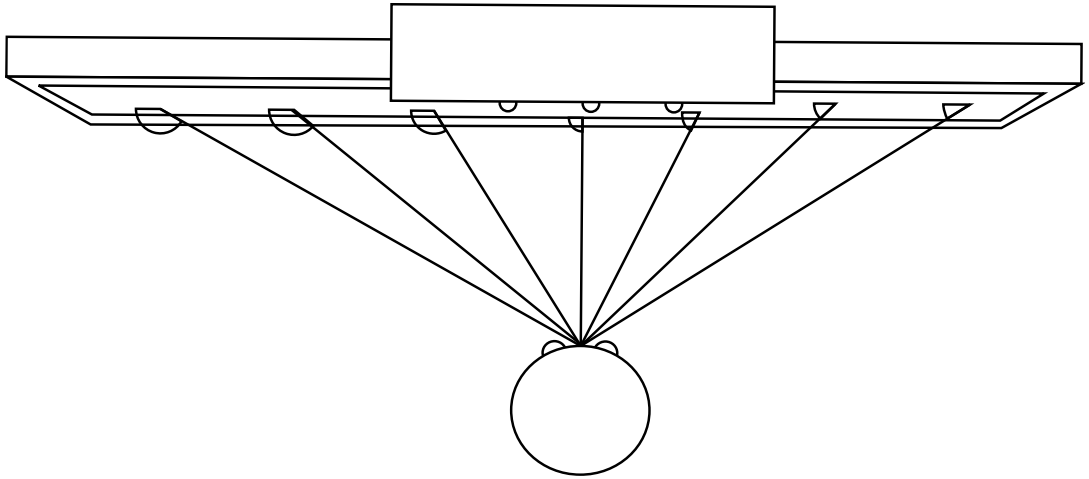
Processing Order

The following are the processing steps in the directional B2A tag:

1. Interpolate 2D Matrix LUT to get 3x3 matrix
    1. Input to interpolation is (azimuth,zenith)
2. Use calc element to multiply input RGB by the 3x3 matrix to get R',G',B'
3. Use calc element to interpolate the 3D tone curve LUTs 3 times
    1. Input is (azimuth,zenith,R'), (azimuth,zenith,G'), & (azimuth,zenith,B')

Applying the Profile

- For best results, the profile should be applied on a per pixel basis
- When viewer is close, viewing angle to different locations on the screen is very different



Applying the Profile

- Movement of the viewer causes significant change in viewing angle
- The application of the directional profile needs to track the user in real time with very little delay
- A GPU can apply the profile in real time
  - The profile can be applied by using a shader
  - A shader can be created by parsing a multiprocessing element tag and converting it to shader code.
  - For this demonstration, a simple shader will use two LUTs extracted from the multiprocessing element
    - This type of extraction can be standardized with a ICS

OpenGL Textures

- The desired B2A tag is loaded and the 2 LUTs are extracted from the multiProcessElementsType tag
- The tone curve LUT is stored as a 3D LUT in the tag with 3 channels (RGB)
  - The OpenGL texture is also a 3D texture. One dimension is azimuth, the other is zenith, and the last is the input device value.
- The matrix LUT is stored as a 2D LUT in the tag
  - In OpenGL it is stored as a Sampler Array

OpenGL example shader code

```
uniform vec3 screen_corner;
uniform vec3 screen_x;
uniform vec3 screen_y;
uniform vec3 viewer_pos;

uniform sampler2DArray profile_matrix;
uniform sampler3D profile_LUTs;

void main()
{
		//get pixel value (assumed to be in XYZ space)
		vec4 pixel = inputFn();

		// get the texture coordinates for the current screen position and views position
		vec2 screen_a_z = getAZTexCoords();

		mat3x3 matrix;
		matrix[0] = texture(profile_matrix,vec3(screen_a_z,0.0)).xyz;
		matrix[1] = texture(profile_matrix,vec3(screen_a_z,1.0)).xyz;
		matrix[2] = texture(profile_matrix,vec3(screen_a_z,2.0)).xyz;

		vec3 rgb = matrix*pixel.rgb;

		//apply LUT
		for (int i=0; i<3; ++i)
		{
		   rgb[i] = texture(profile_LUTs,vec3(screen_a_z,rgb[i]))[i];
		}


		// Apply any final processing
		FragColor.xyz = finalColor(rgb);

		FragColor.a = pixel.a;

}
```

JVO
GH

Conclusion

- Directional tags can be used to make color and luminance invariant across viewing angles and screen location
- The simple MatrixTRC profile can be extended to a profile that characterizes and corrects for viewing angle
- The profile can be applied in real time with the help of a GPU

Questions

Thank You!