Photometric lights

Punctual lights are an extremely practical and efficient way to light a scene but do not give artists enough control over the light distribution. The field of architectural lighting design concerns itself with designing lighting systems to serve humans needs by taking into account:

- The amount of light provided
- The color of the light
- The distribution of light within the space

The lighting system we have described so far can easily address the first two points but we need a way to define the distribution of light within the space. Light distribution is especially important for indoor scenes or for some types of outdoor scenes or even road lighting. Figure 41 shows scenes where the light distribution is controlled by the artist. This type of distribution control is widely used when putting objects on display (museums, stores or galleries for instance).

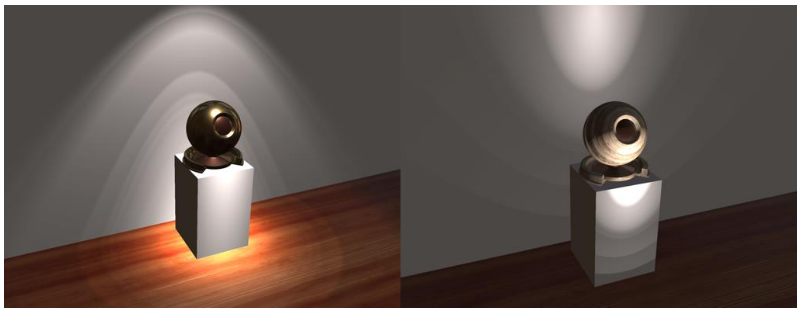
Figure 41: *Controlling the distribution of a point light*

Photometric lights use a photometric profile to describe their intensity distribution. There are two commonly used formats, IES (Illuminating Engineering Society) and EULUMDAT (European Lumen Data format) but we will focus on the former. IES profiles are supported by many tools and engines, such as Unreal Engine 4, Frostbite, Renderman, Maya and Killzone. In addition, IES light profiles are commonly made available by bulbs and luminaires manufacturers (Philips offers an extensive array of IES files for download for instance). Photometric profiles are particularly useful when they measure a luminaire or light fixture, in which the light source is partially covered. The luminaire will block the light emitted in certain directions, thus shaping the light distribution.

Example of a real world luminaires that can be described by photometric profiles

An IES profile stores luminous intensity for various angles on a sphere around the measured light source. This spherical coordinate system is usually referred to as the photometric web, which can be visualized using specialized tools such as IESviewer. Figure 42 below shows the photometric web of the XArrow IES profile provided by Pixar for use with Renderman. This picture also shows a rendering in 3D space of the XArrow IES profile by our tool `lightgen`.

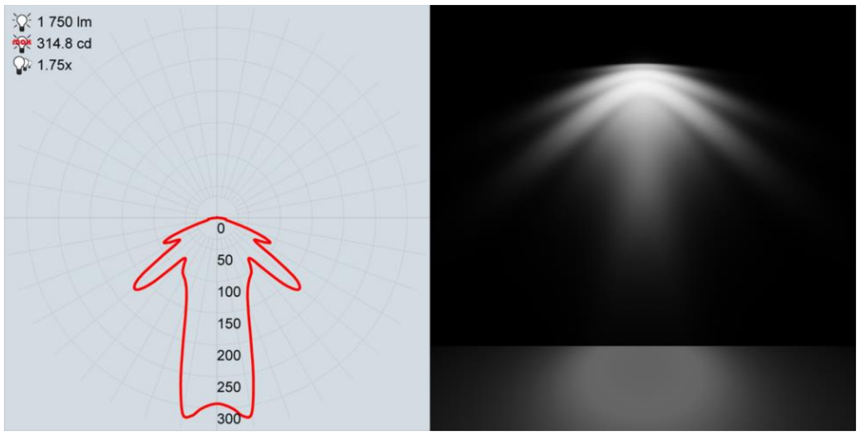

Figure 42: *The XArrow IES profile rendered as a photometric web and as a point light in 3D space*

The IES format is poorly documented and it is not uncommon to find syntax variations between files found on the Internet. The best resource to understand IES profile is Ian Ashdown's “Parsing the IESNA LM-63 photometric data file” document [Ashdown98]. Succinctly, an IES profiles stores luminous intensities in candela at various angles around the light source. For each measured horizontal angle, a series of luminous intensities at different vertical angles is provided. It is however fairly common for measured light sources to be horizontally symmetrical. The XArrow profile shown above is a good example: intensities vary with vertical angles (vertical axis) but are symmetrical on the horizontal axis. The range of vertical angles in an IES profile is 0 to 180° and the range of horizontal angles is 0 to 360°.

Figure 43 shows the series of IES profiles provided by Pixar for Renderman, rendered using our `lightgen` tool.

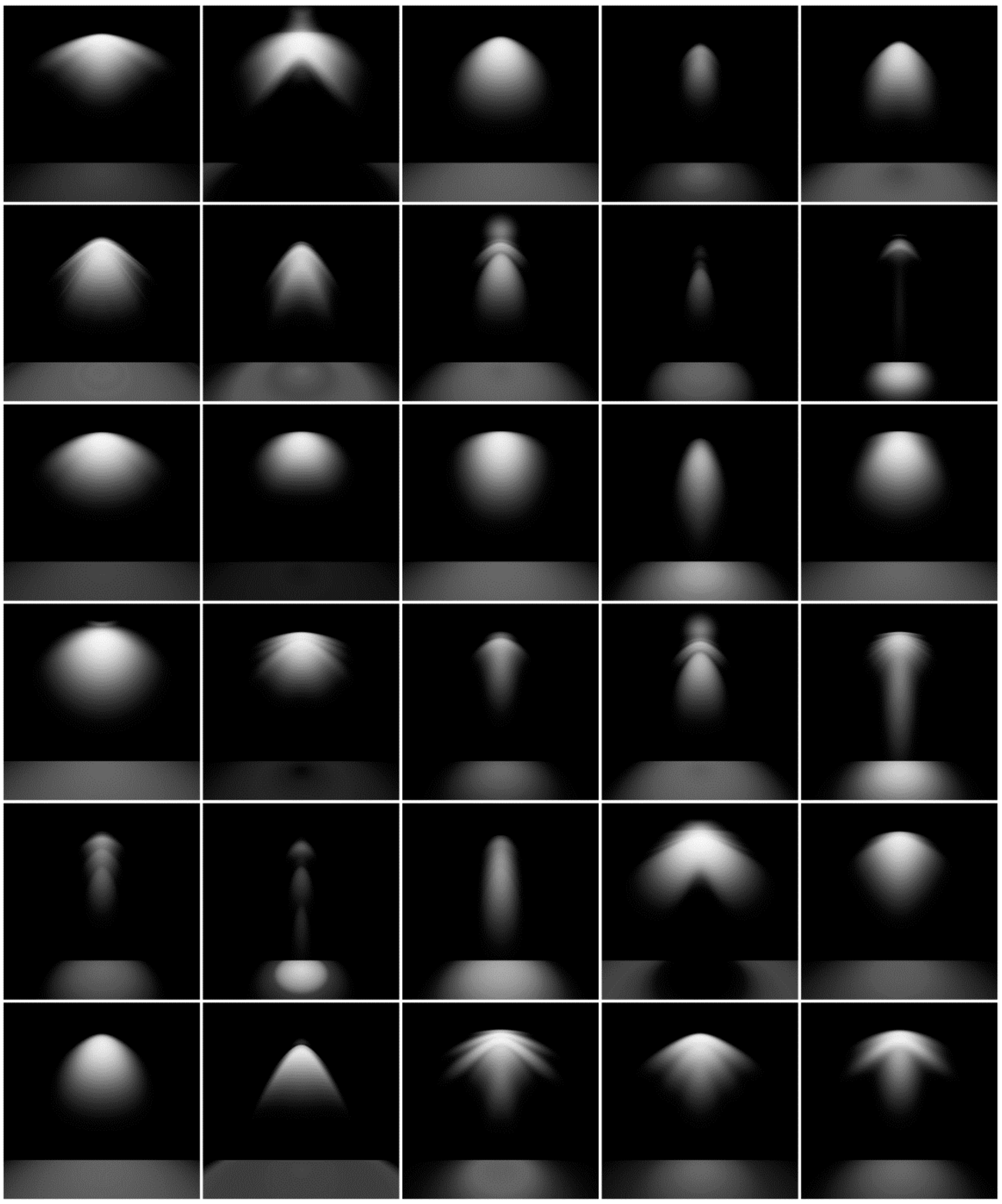
Figure 43: *Series of IES light profiles rendered with lightgen*

IES profiles can be applied directly to any punctual light, point or spot. To do so, we must first process the IES profile and generate a photometric profile as a texture. For performance considerations, the photometric profile we generate is a 1D texture that represents the average luminous intensity for all horizontal angles at a specific vertical angle (i.e., each pixel represents a vertical angle). To truly represent a photometric light, we should use a 2D texture but since most lights are fully, or mostly, symmetrical on the horizontal plane, we can accept this approximation. The values stored in the texture are normalized by the inverse maximum intensity defined in the IES profile. This allows us to easily store the texture in any float format or, at the cost of a bit of precision, in a luminance 8-bit texture (grayscale PNG for instance). Storing normalized values also allows us to treat photometric profiles as a mask:

Photometric profile as a mask

The luminous intensity is defined by the artist by setting the luminous power of the light, as with any other punctual light. The artist defined intensity is divided by the intensity of the light computed from the IES profile. IES profiles contain a luminous intensity but it is only valid for a bare light bulb whereas the measured intensity values take into account the light fixture. To measure the intensity of the luminaire, instead of the bulb, we perform a Monte-Carlo integration of the unit sphere using the intensities from the profile[4].

Photometric profile

The luminous intensity comes from the profile itself. All the values sampled from the 1D texture are simply multiplied by the maximum intensity. We also provide a multiplier for convenience.

The photometric profile can be applied at rendering time as a simple attenuation. The luminance equation 67 describes the photometric point light evaluation function.

$$L_{out}=f(v,l)\frac{I}{d^2}\langle n\cdot l\rangle\Psi(l) \qquad (67)$$

The term $\Psi(l)$ is the photometric attenuation function. It depends on the light vector, but also on the direction of the light. Spot lights already possess a direction vector but we need to introduce one for photometric point lights as well.

The photometric attenuation function can be easily implemented in GLSL by adding a new attenuation factor to the implementation of punctual lights (listing 21). The modified implementation is show in listing 22.

```glsl
float getPhotometricAttenuation(vec3 posToLight, vec3 lightDir) {
    float cosTheta = dot(-posToLight, lightDir);
    float angle = acos(cosTheta) * (1.0 / PI);
    return texture2DLodEXT(lightProfileMap, vec2(angle, 0.0), 0.0).r;
}

vec3 evaluatePunctualLight() {
    vec3 l = normalize(posToLight);
    float NoL = clamp(dot(n, l), 0.0, 1.0);
    vec3 posToLight = lightPosition - worldPosition;

    float attenuation;
    attenuation  = getSquareFalloffAttenuation(posToLight, lightInvRadius);
    attenuation *= getSpotAngleAttenuation(l, lightDirection, innerAngle,
outerAngle);
```

```
    attenuation *= getPhotometricAttenuation(l, lightDirection);

    float luminance = (BSDF(v, l) * lightIntensity * attenuation * NoL) *
lightColor;
    return luminance;
}
```

Listing 22: *Implementation of attenuation from photometric profiles in GLSL*

The light intensity is computed CPU-side (listing 23) and depends on whether the photometric profile is used as a mask.

```
float multiplier;
// Photometric profile used as a mask
if (photometricLight.isMasked()) {
    // The desired intensity is set by the artist
    // The integrated intensity comes from a Monte-Carlo
    // integration over the unit sphere around the luminaire
    multiplier = photometricLight.getDesiredIntensity() /
            photometricLight.getIntegratedIntensity();
} else {
    // Multiplier provided for convenience, set to 1.0 by default
    multiplier = photometricLight.getMultiplier();
}

// The max intensity in cd comes from the IES profile
float lightIntensity = photometricLight.getMaxIntensity() * multiplier;
```

Listing 23: *Computing the intensity of a photometric light on the CPU*

[4] The XArrow profile declares a luminous intensity of 1,750 lm but a Monte-Carlo integration shows an intensity of only 350 lm.