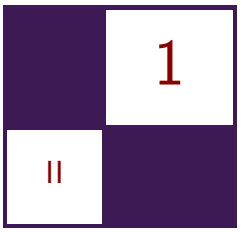

Physically Based Area Lights

Michal Drobot

1.1 Overview

This chapter presents the physically based area lighting system used in *Killzone: Shadow Fall*, developed by Guerrilla Games for Playstation 4 (see Figure 1.1).

We present a novel, real-time, analytical model for area lights, capable of supporting multiple light shapes. Each shape can be represented by simple 3D or 2D functions on a plane. Discussed applications include the following light shapes: sphere, disk, and rectangle.

The model supports diffuse and specular lighting. BRDF (bidirectional reflectance distribution function) implementation in *Killzone: Shadow Fall* rendering engine explicitly splits the material reflectance model from the lighting model itself. This allows a separation between different surface simulation algorithms and light types, which supply the former with required light quantities. To achieve this, we use the importance sampling principle to approximate non-finite or computationally expensive integrals in the material reflectance part of the BRDF. This chapter focuses only on the derivation of the proposed framework and actual light modeling part.

All lights in the rendering engine are physically based area lights, described by radiometric quantities such as light intensity in lumens, dimensions in meters, world orientation, and light shape type. We introduce a specialized description of surface roughness that is shared or adapted to match various surface reflectance models. The light model uses light description and surface roughness, at the point being shaded, to deliver light quantity arriving at the point—split between the light model and the surface reflectance model.

In addition we discuss integration of the proposed model into deferred renderer and how it was used as a principle for environmental probe generation and dynamic analytical texture-based area lights.

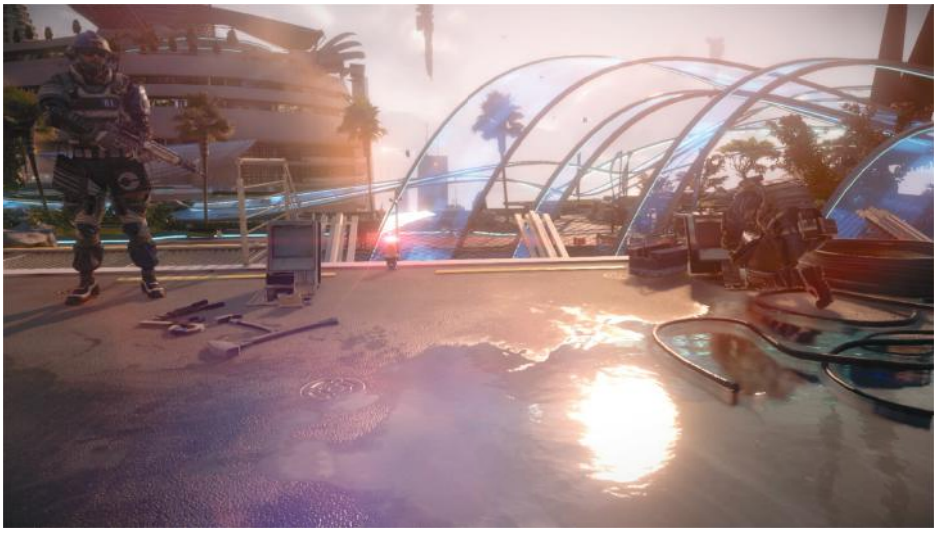

Figure 1.1. Usage of area lights in *Killzone: Shadow Fall*.

1.2 Introduction

The current game industry standard for lighting models is Blinn-Phong BRDF or models directly based on it. In recent years we have seen multiple advances extending the model to support more varied materials, surface properties, and physically based properties [McAuley et al. 13] or ways to tackle aliasing problems [Baker and Hill 12]. The result of this technological push is widespread access to an efficient, predictable, well-known lighting model, capable of capturing most material properties that we might observe in common scenarios. Most research focused on refining material interactions, including well-known geometric and fresnel terms proposed in the Cook-Torrance lighting model [Cook and Torrance 81]. However, a very basic constraint of the model still exists, as it can only simulate point-based lights. In almost every scenario, the source of light has a physical size, which in real life is reflected by the correct shape of specular reflection and diffuse lighting response. Using Blinn-Phong point lighting for dynamic lights proves inadequate in many situations, creating a visual disjoint between visible lights in the scene and lighting result (see Figures 1.2 and 1.3).

Several methods exist to tackle this issue. Some of them include pre-computing "light cards" or billboard reflections and raytracing them at runtime to simulate accurate specular reflections [Mittring and Dudash 11]. Unfortunately, this system is an addition on top of standard analytical, dynamic lighting that is point based. Moreover, it doesn't provide a solution for area-based diffuse lighting.

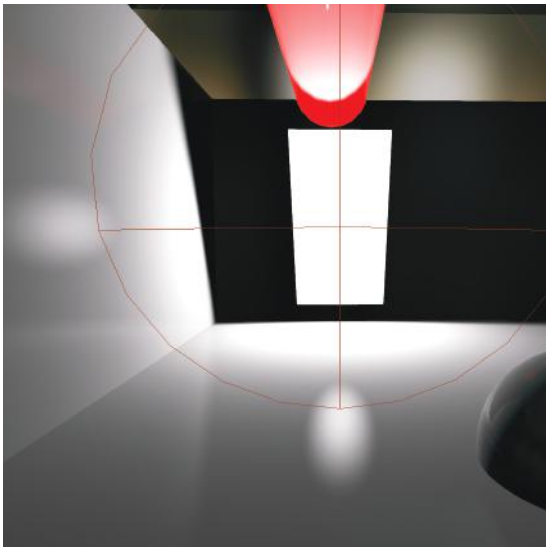

Figure 1.2. Visual disparity between the light shape and Blinn-Phong specular reflection.

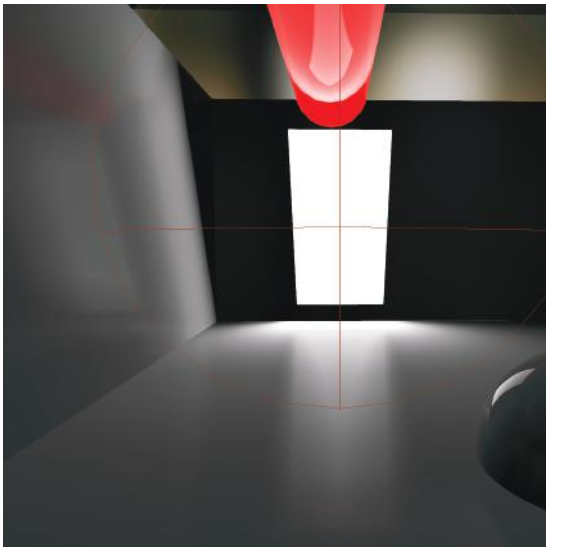

Figure 1.3. Specular reflection matches the light shape using the proposed model.

Another way of solving the problem involves switching to global illumination–based solutions. Several systems were already implemented in commercial engines, mostly voxel based [Mittring 12]; however, they can't fully substitute for analytical lights, due to stability, resolution, or quality difference.

During our research on a new iteration of the *Killzone* engine for next generation platform, we wanted to leverage current knowledge about lighting models and extend it to cover non-point-based lights. A unified way to deal with art production was also our priority. With our transition to physically based material modeling, we also wanted to have physically based lights, using real-world radiometric quantities, thus achieving a predictable shading model.

We decided to utilize existing BRDFs to model surface reaction to light and remodel the way lighting information is actually provided to those models. Standard BRDFs assume light incoming from only one direction with intensity given as a per-light set quantity. When dealing with an area-based light source, we would have to solve an integral of the lighting model over all points of the light shape. This can be achieved numerically, but unfortunately that proves unfeasible performance-wise in real-time applications. However, parts of that integral can be calculated analytically using radiometric integrals, while the rest can be efficiently approximated.

1.3 Area Lighting Model

1.3.1 Radiometric Integrals and BRDF Definition

In this section we introduce basic radiometric quantities such as light intensity, irradiance, and radiance [Pharr and Humphreys 04]. Then, we define radiometric integrals essential to solving area lighting models.

Let *intensity* be defined as the light flux density per solid angle:

$$I = \frac{d\phi}{d\omega},$$

where $d\phi$ is the light flux differential and $d\omega$ is the solid angle differential. Intensity is meaningful only for a point light source.

Irradiance defines the total amount of light flux per area:

$$E = \frac{d\phi}{dA},$$

where dA is the differential area receiving light flux.

Radiance describes the light flux density per unit area, per unit solid angle:

$$L = \frac{d\phi}{d\omega dA^{\perp}},$$

where $dA^{\perp}$ is the projected area of dA on a hypothetical surface perpendicular to the solid angle.

We also define radiance emitted $L_o(p, \omega)$ and incoming $L_i(p, \omega)$ to a point on a surface as a function of the point and direction.

Irradiance at point p with normal vector n would be defined as

$$E(p, n) = \int_{\mathcal{H}^2(n)} L_i\,(p, \omega)\, cos\theta d\omega,$$

where $\cos\theta d\omega$ is the projected solid angle $d\omega^{\perp}$, with θ the angle between ω and the surface normal n. This term comes from the definition of radiance. In other words, this equation integrates incoming light from all directions on a hemisphere, around a point of integration with a given normal, with respect to the projected solid angle.

A *bidirectional reflectance distribution function* (BRDF) defines a ratio of the light reflecting from a surface point, in the viewer's direction, and the amount of light incoming to a surface from a specific direction. Therefore, a basic definition of BRDF is

$$f_r\,(p, \omega_o, \omega_i) = \frac{dL_o\,(p, \omega_o)}{dE\,(p, \omega_i)}.$$

In the case of real-time graphics, we are interested in finding the integral of $L_o(p, \omega_o)$ over the whole hemisphere $\mathcal{H}$ set around point p with surface normal vector n. Therefore, we are looking for

$$L_o(p, \omega_o) = \int_{\mathcal{H}^2} f_r(p, \omega_o, \omega_i) L d\omega_i.$$

Using the definition of radiance L,

$$L_o(p, \omega_o) = \int_{\mathcal{H}^2} f_r(p, \omega_o, \omega_i) L_i(p, \omega_i) \cos\theta_i d\omega_i. \tag{1.1}$$

During rendering we evaluate a finite number of lights. Therefore, we are interested in expressing integrals over area. In the case of irradiance over area A, we can define

$$E(p, n) = \int_A L_i(p, \omega_i) \cos\theta_i d\omega_i.$$

In the simplified case of n contributing lights, we can express the integral from equation (1.1) as a sum of integrals over all area lights that are visible from point p:

$$L_o(p, \omega_o) = \sum_{1..n} \int_{A(n)} f_r(p, \omega_o, \omega_i) L_i(p, \omega_i) \cos\theta_i d\omega_{i(n)}. \tag{1.2}$$

Equation (1.2), our main lighting equation, will be the basis of all derivations.

For simplicity we can assume that the light source has uniform light flux distribution so $L(p, w)$ is constant and denoted L; therefore,

$$L_o(p, \omega_o) = \sum_{1..n} L_n \int_{A(n)} f_r(p, \omega_o, \omega_i) \cos\theta_i d\omega_{i(n)}.$$

The differential solid angle is also in relation with the differential area.

In the case of a light source defined on a quadrilateral, the differential solid angle can be expressed as a function of differential area of light:

$$d\omega = \frac{dA cos\theta_o}{r^2}, \tag{1.3}$$

where r is the distance from point p on the surface to point p' on dA and θ_o is the angle between surface normal dA at point p' and $\overrightarrow{p'p}$ (see Figure 1.4).

It is worth noting that the solid angle is well defined for multiple primitives that can be used as a light source shape, such as a disk and a rectangle.

To finalize, radiance at a point per single light is defined as

$$L_o(p, \omega_o) = \sum_{1..n} L_n \int_{A(n)} f_r(p, \omega_o, \omega_i) cos\theta_i \frac{dA cos\theta_o}{r^2}. \tag{1.4}$$

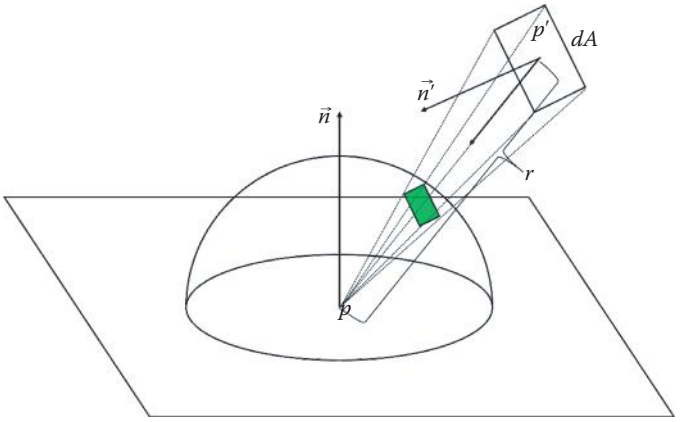

Figure 1.4. Solid angle of quadrilateral as visible from point p.

1.3.2 Material and Lighting Models

After deriving the principal integral (equation (1.4)) for light rendering, we assume that the light area is well defined and therefore possible to integrate. For simplification we restrict our reasoning to simple shapes—a sphere, a disk, and a rectangle—which are relatively easy to integrate and compute.

Our light is defined by the following parameters:

- position,
- orientation,
- outgoing light radiance in lumens,
- shape type (sphere, disk, rectangle),
- dimensions.

With those parameters we can instantly calculate L in lumens. We need to find a way to solve or approximate the integral from equation (1.4). To simplify the problem, let us look at the generalized physically based BRDF combining diffuse and specular reflectance models:

$$f_r\left(p, \omega_o, \omega_i\right) = k_d + k_s, \tag{1.5}$$

where k_d is the diffuse light model and k_s is the specular model.

We also need to set additional requirements to make this BRDF physically based:

$$\begin{gathered} f_r\left(p, \omega_o, \omega_i\right) \geq 0, \\ f_r\left(p, \omega_o, \omega_i\right) = f_r\left(p, \omega_i, \omega_o\right), \\ \bigvee_{\omega_o} \int_{\mathcal{H}^2} f_r\left(p, \omega_o, \omega_i\right) \cos\theta_i d\omega_i \leq 1. \end{gathered} \tag{1.6}$$

For our k_d we can use the standard Lambertian diffuse model [Lambert 60]. When expressed as a part of $f_r\left(p, \omega_o, \omega_i\right)$, it takes a very simple form:

$$k_d = C_d, \tag{1.7}$$

where C_d defines the surface diffusion color.

We choose the generalized Cook-Torrance BRDF [Cook and Torrance 81] for a base of our microfacet specular model:

$$k_s\left(p, \omega_o, \omega_i\right) = \frac{D(\vec{h})F(\omega_o, \vec{h})G(\omega_i, \omega_o, \vec{h})}{4(cos\theta_i)(cos\theta_o)}, \tag{1.8}$$

where $D(\vec{h})$ is the distribution of micro-facets around surface normal n'', $F(\omega_o, \vec{h})$ is the Fresnel reflectance function, and $G(\omega_i, \omega_o, \vec{h})$ is the geometric function. As previously defined, θ_i is the angle between $\vec{n}$ and ω_i, and θ_o is the angle between $\vec{n}$ and ω_o. Generally, $\vec{h}$ is called the *half vector*, defined as

$$\vec{h} = \frac{\omega_o + \omega_i}{\|\omega_o + \omega_i\|}. \tag{1.9}$$

We are interested in finding radiance in the direction of the viewer, per light, described as follows:

$$L_o\left(p, \omega_o\right) = \int_{A(n)} f_r\left(p, \omega_o, \omega_i\right) L_i\left(p, \omega_i\right) \cos\theta_i d\omega_{i(n)}.$$

For now, we can assume, as in equation (1.2), that $L_i\left(p, \omega_i\right)$ is constant over light:

$$L_o\left(p, \omega_o\right) = L_n \int_{A(n)} f_r\left(p, \omega_o, \omega_i\right) \cos\theta_i d\omega_{i(n)}. \tag{1.10}$$

Now substitute parts of equation (1.10) with equations (1.5), (1.7), and (1.8):

$$L_o\left(p, \omega_o\right) = L_n \int_{A(n)} \left(C_d + \frac{D(\vec{h})F(\omega_o, \vec{h})G(\omega_i, \omega_o, \vec{h})}{4(cos\theta_i)(cos\theta_o)}\right) cos\theta_i d\omega_{i(n)}.$$

In the end we can define two integrals:

$$Diffuse\,(p,\omega_o) = L_n \int\limits_{A(n)} C_d \cos\theta_i d\omega_{i(n)}, \tag{1.11}$$

$$Specular\,(p,\omega_o) = L_n \int\limits_{A(n)} \frac{D(\vec{h})F(\omega_o,\vec{h})G(\omega_i,\omega_o,\vec{h})}{4(cos\theta_o)} d\omega_{i(n)}. \tag{1.12}$$

To get the final form of specular integral, we need to choose functions DFG. There are multiple sources available that discuss the best choice for a specific use scenario [Burley 12].

Unfortunately, independent of the chosen function, the integrals from equations (1.11) and (1.12) are not easily solvable for shapes other than a sphere. Therefore, we will focus on finding a suitable approximation for light shapes that can be expressed as a 2D function on a quadrilateral.

1.3.3 Approximating Diffuse Integral

Monte Carlo methods and importance sampling. One of the known ways to solve an integral is numerical integration by discretized parts. There are multiple ways and techniques to accelerate this process. A particularly interesting one for us is the Monte Carlo technique, which in general can be described in the following steps:

1. Define a domain of possible inputs.
2. Generate inputs randomly from a probability distribution over the domain.
3. Calculate the function being integrated for given inputs.
4. Aggregate the results.

With a given probability distribution, the expected variance is also known as well as the estimator for minimal acceptable error. This process can be significantly sped up using importance sampling techniques [Pharr and Humphreys 04].

The principle of *importance sampling* is to prioritize samples that would have maximum impact on the final outcome. We can find such samples using spatial heuristics. Another solution is to run an initial pass of Monte Carlo integration with a low number of samples to estimate the result variance and therefore decide which regions of integration would use more important samples.

Importance sampling is an actively researched subject with multiple different solutions. The takeaway for our reasoning is that, in any integration, there are samples more important than others, therefore minimizing the error estimator for a given amount of samples.

Application to diffuse integral. We would like to apply the Monte Carlo method to our diffuse integral. In order to simplify that process, we assume that the light shape has uniform surface normal n. Another assumption is that dA does not change for integrated points, meaning the whole shape is always visible. Therefore, we can conceptually move $dAcos\theta_o$ out of our integral and compute it once for the entire domain:

$$dAcos\theta_o \int_A \frac{cos\theta_i}{r^2}. \tag{1.13}$$

The integrated function is dependent only on the distance and cosine between the surface normal and the directions subtended by the light shape. Those are continuous and well-defined functions.

Immediately we can see that the global minimum and maximum of the integrated function are inside the sampling domain defined by the light shape. Therefore, there is a single point, on the light shape, that is able to represent the integral, thus minimizing the error estimator.

If we could find this point, we could essentially solve the integral by approximation using importance sampling with one point of highest importance. We would just need to evaluate the function once for that specific point. However, we are only interested in points that are easy to find at GPU program runtime. The ideal candidate would be given by a function of light parameters, the position of point p, and the normal n. We need to find a function that returns a point minimizing the error estimator. We also need to minimize the function complexity, to maintain runtime performance. To simplify the problem's domain, we prefer to work in the 2D space of a light quadrilateral.

Our function of interest is bounded by the light shape and defined as

$$\frac{cos\theta_i}{r^2}. \tag{1.14}$$

We know that the function integral over area is bounded by local minima and maxima in limits of given area. The function from equation (1.14) has one global maximum and one global minimum at the shape bounds. Therefore, we know that a single point best representing the integral can be found on a segment between the maximum and minimum. In order to find it, we would have to calculate the global minimum and maximum, which we deemed to be too computationally expensive.

Instead, we decided to find an approximate important point in the proximity of the global maximum, accepting the risk of overestimation. In order to do so, we need to set boundary conditions for our search. The function from equation (1.14) is a component-wise multiplication of two functions. The maximum of their products can be found along a segment connecting their local maximums.

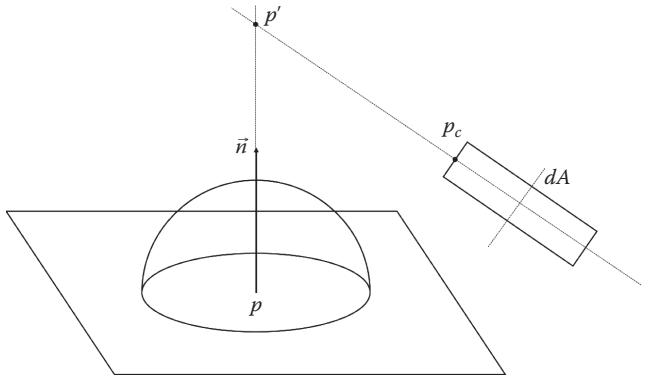

Figure 1.5. Geometric construction of point p_c.

We can easily find $cos\theta_o$ maximum on A geometrically by casting a ray from point p in the direction of the normal n of the surface, creating point p', intersecting with the light plane and finding the closest location on the shape to p', called p_c (see Figure 1.5). It is worth noting that in the case when the light plane is pointing away from the surface normal (i.e., $n \cdot n' > 0$), vector $\overline{pp'}$ should be skewed in the direction of the light plane in order to obtain intersection (see Figure 1.5).

A maximum of $\frac{1}{r^2}$ can be found by projecting point p on the light plane, creating point p_p, finding the closest point on positive hemisphere to p_p, called p'', and finally finding a closest point on the light shape to p_r (see Figures 1.6 and 1.7).

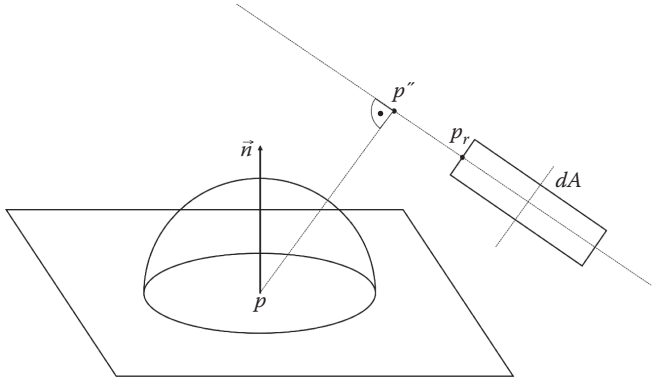

Figure 1.6. Geometric construction of point p_r when the light plane is pointing toward the surface.

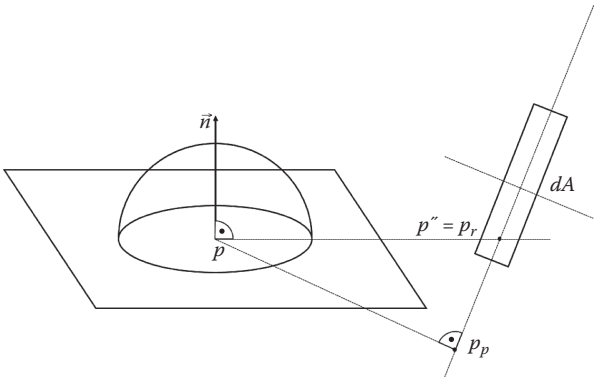

Figure 1.7. Geometric construction of point p_r when the light plane is pointing away from the surface.

As previously discussed, we were searching for the most important sample of the integral on a segment between points p_r and p_c representing component-wise local maximums (see Figure 1.8).

With those conditions set, we used a computational software package to numerically find a point on line $\overline{p_c p_r}$ that approximates the most important point as much as possible. We worked on a data set of several hundred area lights randomly generated as disk or rectangle light shapes. For every light shape we found a point on the plane that best resembles the full integral. Then, we computed the end points of our line between p_c and p_r (as if it would be calculated at runtime). Then, we numerically checked the points along the line, computing the lighting equation and comparing against a reference, using the least squares method to find point p_d, which would most accurately represent the integral.

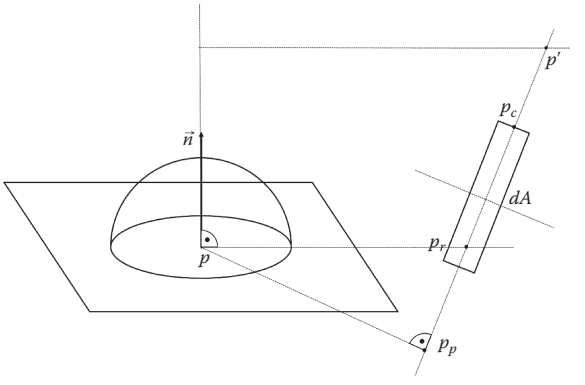

Figure 1.8. Geometric construction of line $\overline{p_c p_r}$.

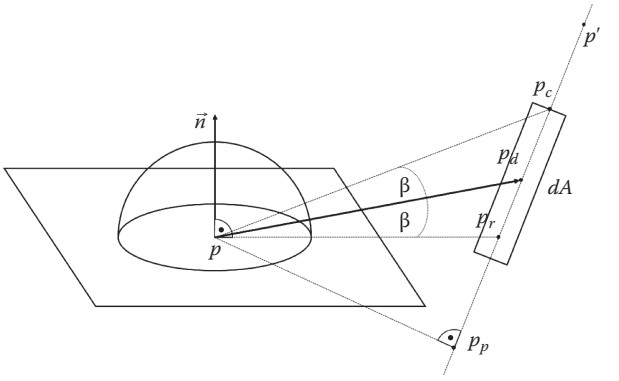

Figure 1.9. Geometric construction of point p_d.

After evaluating many test cases and several functions of various complexity, we noticed that intersecting a simple halfway vector between $\overrightarrow{pp_c}$ and $\overrightarrow{pp_r}$ with the light plane to create p_d (see Figure 1.9) works reasonably well in the majority of our cases, having a statistical error less than 0.05 against best-point approximating the integral.

After visual assessment we decided it works well enough in our use case (see Figure 1.10) and is visually indifferent from the raytaced solution (see Figure 1.11). It also proves simple enough for real-time per-pixel calculation on modern GPUs.

As another level of simplification, we used the halfway vector between $\overrightarrow{pp'}$ and $\overrightarrow{pp''}$. It allowed the important optimization of skipping the closest point to the shape calculation. Unfortunately, it yields a statistical error of more than 0.15. However, the disparity is mostly visible in an edge case scenario, where the light shape is pointing away from surface normal. There are no visual artifacts noticeable, and our artists decided it is good enough for our needs.

Area diffuse approximation algorithm. Our final approximation for diffuse integral can therefore be expressed by the following pseudo code:

- For point p with normal n:
  - Intersect a ray from p in direction n'' with light plane, creating new point p'.
  - Project point p on the light plane, creating point p''.
  - Create halfway vector $\overrightarrow{d_h}$ between $\overrightarrow{pp'}$ and $\overrightarrow{pp''}$:

$$\overrightarrow{d_h} = \frac{\overrightarrow{pp'} + \overrightarrow{pp''}}{\|\overrightarrow{pp'} + \overrightarrow{pp''}\|}.$$

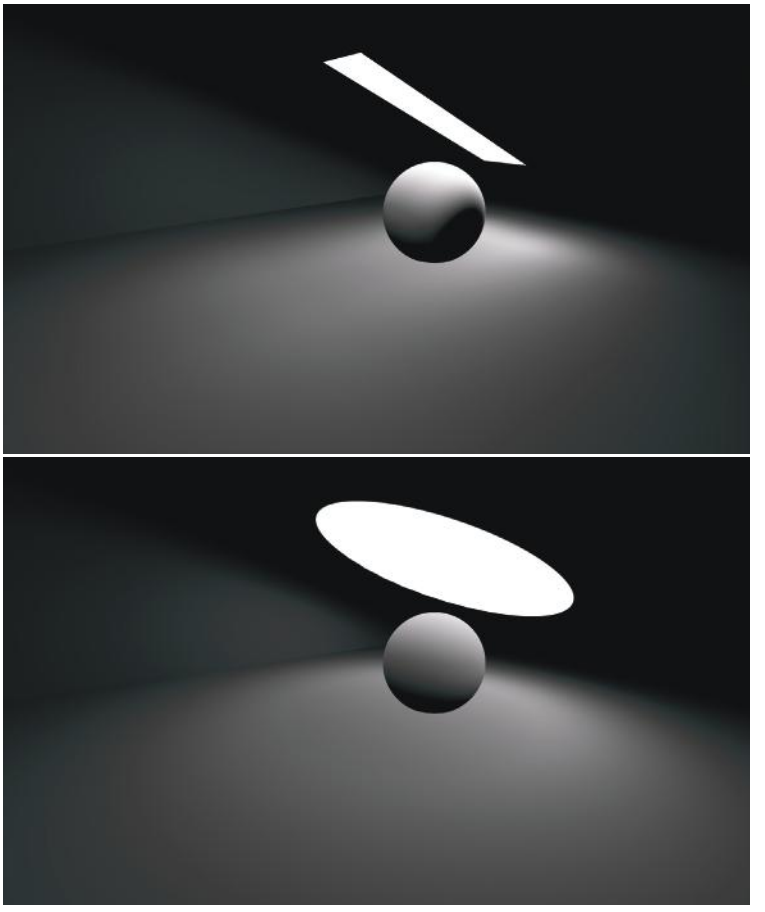

Figure 1.10. Correct wrapped diffuse lighting from rectangular (top) and disk (bottom) area light.

- Intersect a ray from p in direction $\vec{h}$ with the light plane creating point p_d. This is the most important point in terms of importance sampling.
- Treat vector $\overrightarrow{pp_d}$ as a light vector for the diffuse equation, effectively approximating the diffuse integral from equation (1.11):

$$\textit{Diffuse}\,(p,\omega_o) = L_n \int\limits_{A(n)} C_d \cos\theta d\omega_{i(n)} \sim L_n C_d \cos\theta_{\overrightarrow{pp_d}} d\omega_{\overrightarrow{pp_d}}. \quad (1.15)$$

In equation (1.15) we assume that L_n is constant for the whole area of the light. The angle between the new light vector $\overrightarrow{pp_d}$ and surface normal n is $\theta_{\overrightarrow{pp_d}}$.

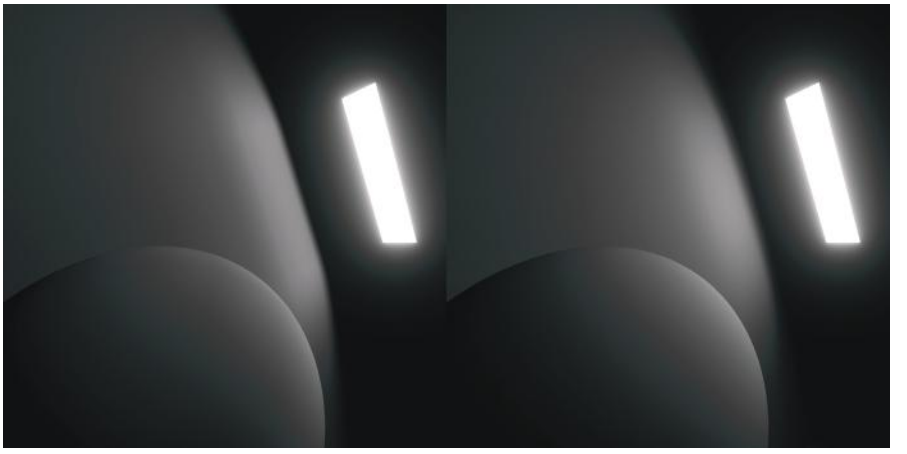

Figure 1.11. Comparison between Monte Carlo raytraced reference and proposed analytical solution: proposed model (left) and Monte Carlo sampled reference (right).

We can also express the differential solid angle as a function of distance and differential area using equation (1.3). Therefore, our final approximated diffuse integral is

$$Diffuse\,(p,\omega_o) \sim L_n C_d \cos\theta_{\overrightarrow{pp_d}} \frac{dAcos\theta_o}{r^2}, \tag{1.16}$$

where θ_o is the angle between the surface normal n and the light plane orientation normal n_l and dA is given by the light shape.

1.3.4 Approximating Specular Integral

Introduction. We followed the idea of leveraging importance sampling to estimate the specular integral. First we analyzed the behavior of typical specular lighting models relying on probability distribution functions (PDFs). This led us to the definition of a specular cone of importance sampling, used to find the most important point for the specular integral and the actual integration area.

Specular in importance sampling. If we were to render specular reflection using Monte Carlo methods, we would cast a ray in every direction around the point being shaded and evaluate specular BRDF for given functions DFG from equation (1.12). Also, every PDF used to model D depends on some kind of surface-roughness parameter, denoted g. This parameter describes how focused lighting remains after reflecting off the surface.

In the case of specular reflection, we can define a vector created by reflecting in the viewer's direction ω_o against the surface normal n. Such a vector is called the reflection vector $\vec{r}$.

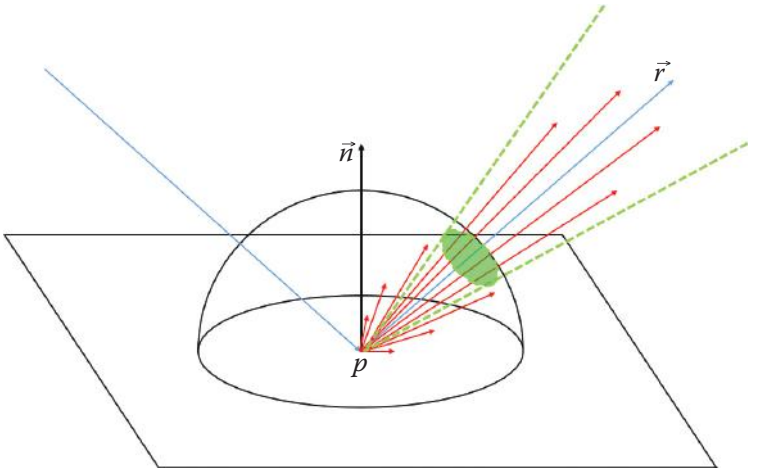

Figure 1.12. Visualization of reflection ray spread due to surface roughness.

Due to the nature of specular reflection, most samples that have meaningful weights would focus around $\vec{r}$. Their weights toward a final solution would be directly correlated to the angular distance of the ray being integrated to $\vec{r}$. They would also relate directly to material parameter g. Therefore, we can define a specular cone of importance sampling, centered around $\vec{r}$, encompassing all important ray samples (see Figure 1.12).

By the term *important*, we mean every ray that has absolute weight greater than a threshold σ (assuming that a ray shot in the direction of $\vec{r}$ would have a weight of 1.0). We can easily see that, with a constant σ, the cone apex angle α depends only on the surface glossiness factor g (see Figure 1.13). Therefore, we are interested in finding a function that calculates the specular cone angle from the surface glossiness, with a given specular model and constant σ.

As an example, we can apply this reasoning to find such a function for the Phong specular model:

$$k_{Phong} = (r \cdot n)^g \,. \tag{1.17}$$

We prepare data containing the specular cone angles α, at which $k_{Phong} > \sigma$. Then, for a given dataset, we find an approximation function. From possible candidates we pick a function with the lowest computational cost. In our case the function of choice is

$$\alpha\,(g) = 2\sqrt{\frac{2}{g+2}}. \tag{1.18}$$

It coincides with the Beckmann distribution definition of roughness m [Beckmann and Spizzichino 63], where m is given as the root mean square of the specular cone slope.

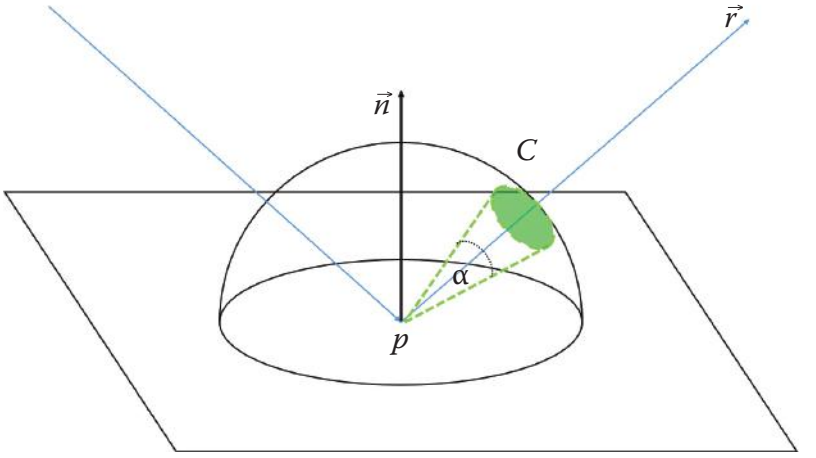

Figure 1.13. Visualization of cone of specular importance sampling.

We successfully applied this method to various specular models such as Phong, Blinn-Phong, Beckmann and GGX [Walter et al. 07]. However, it is worth noting that a cone shape is a relaxed bound on the actual distribution 3D shape, the complexity of which varies over different BRDF. Therefore, the cone shape is only approximate. In the case of Phong, the cone is actually a perfect match. However, in the case of Blinn-Phong, due to the use of the half vector, a pyramid with an ellipse base or even more complex shape would provide tighter, more accurate bounds.

Now we have an approximation model for the most important samples of our specular function. In the case of a single area light source, we would be interested in calculating the integral of our specular function over the portion of area of the light subtending the cone—defining integration limits. We can approximate the final solution by applying reasoning and methodology similar to Section 1.3.3. Again, a varied database was prepared, an integral numerically calculated, and the most important points estimated. In the end we found out that the geometric center of the integration limits area is a good point to estimate the unbounded specular integral. Let's call it p_{sc}. Therefore, to obtain the final result, we must calculate $Specular\,(p, \omega_o)$ for a single point light at p_{sc} and normalize the result by the integration limits area (see Figure 1.14).

To simplify our problem, we move to the light space (see Figure 1.15) by projecting our specular cone onto the light plane. In that case we are looking at a 2D problem, of finding the area of intersection between an ellipsoid (projected specular cone) and a given light shape (in our case a disk or rectangle, as spheres prove to be easily solvable working exclusively with solid angles).

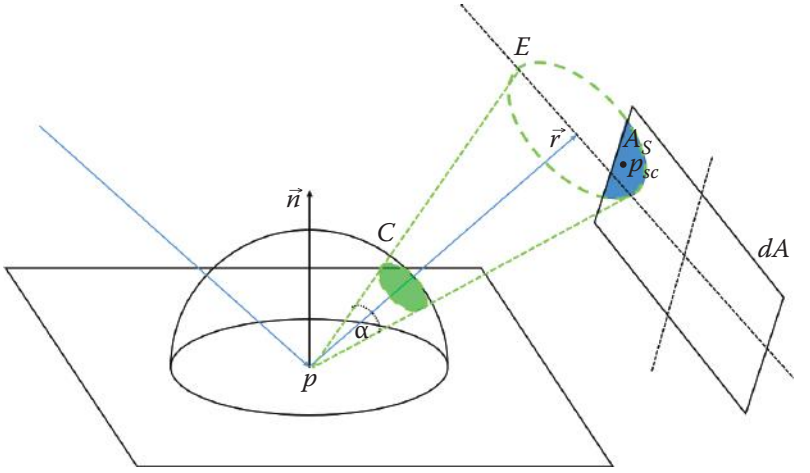

Figure 1.14. Construction of point p_{sc} and computing the intersection area A_S.

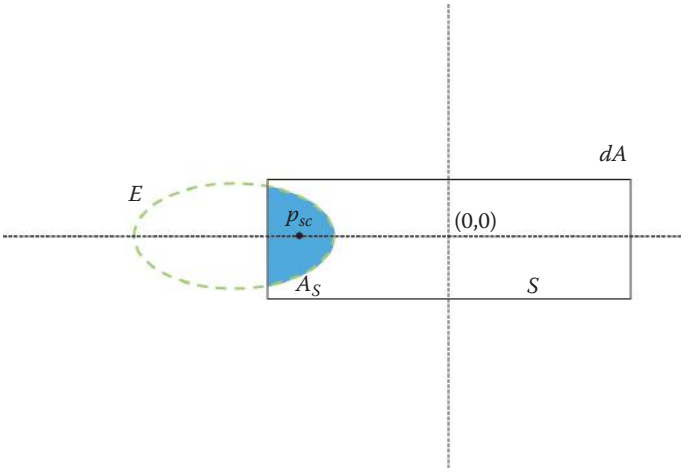

Figure 1.15. Situation from Figure 1.14 visible in 2D on the light plane.

Area specular approximation algorithm. Here is pseudo code of the algorithm to approximate area specular integral (see Figures 1.14 and 1.15 for reference):

- For every shaded point p, find reflection vector r.
- Calculate the specular importance cone apex angle $\alpha_{BRDF}(g)$ from point p and glossiness g, where $\alpha_{BRDF}(g)$ is defined per the BRDF used to model specular lighting.

- Create cone C with apex at point p and opening angle α, oriented around vector r.
- Project cone C onto the light plane of area light L, creating ellipsoid E.
- Find area A_S of the intersection between ellipsoid E and function S describing specular light shape.
- Find geometric center p_{sc} of A_S—the importance point of the specular integral.
- Calculate $Specular\,(p, \omega_o)$ as if p_{sc} was a point light, therefore vector $\overrightarrow{pp_{sc}}$.
- Normalize the result by the actual area of A_S as an effective solid angle differential.

Applying this to equation (1.12) results in

$$Specular\,(p, \omega_o) \sim L_n \frac{D(\vec{h})F(\omega_o, \vec{h})G(\omega_i, \overrightarrow{pp_{sc}}\vec{h})}{4(cos\theta_o)} d\omega_{\overrightarrow{pp_{sc}}}, \qquad (1.19)$$
$$\vec{h} = \|\omega_o, \overrightarrow{pp_{sc}}\|.$$

After substituting the solid angle differential from equation (1.19) by A_S normalization, we get

$$Specular\,(p, \omega_o) \sim L_n \frac{D(\vec{h})F(\omega_o, \vec{h})G(\omega_i, \overrightarrow{pp_{sc}}\vec{h})}{4(cos\theta_o)A_S}. \qquad (1.20)$$

When solving for specific light, we use the appropriate method per light type to calculate A_S. The following section focuses on approximations for various light-shape intersections with the specular importance cone.

Intersection area calculation. Finding an accurate area of intersection, between various 2D shapes, at runtime is not a trivial task. In our case we are looking at ellipse-disk or ellipse-rectangle intersections. Disks and rectangles are axis aligned in light space and centered at $(0, 0)$, as they represent the actual light shape. Ellipses can be rotated. See Figures 1.14 and 1.15 for reference. To minimize shader complexity, we decided to approximate an ellipsoid by a simple disk. Calculating the area of intersection of two disks, or an axis-aligned box and a disk, is a reasonably easy problem to solve. In the case of a disk-disk intersection, we use the known smooth step approximation proposed by [Oat 06]:

$$A_{D0D1} = A_{min} smoothstep\left(0, 1, 1 - \|c0 - c1\| - \frac{|r0 - r1|}{r0 + r1 - |r0 - r1|}\right), \qquad (1.21)$$

where $c0$, $r0$, $c1$, and $r1$ are the center and radius of disks $D0$ and $D1$, respectively (see Figure 1.16).

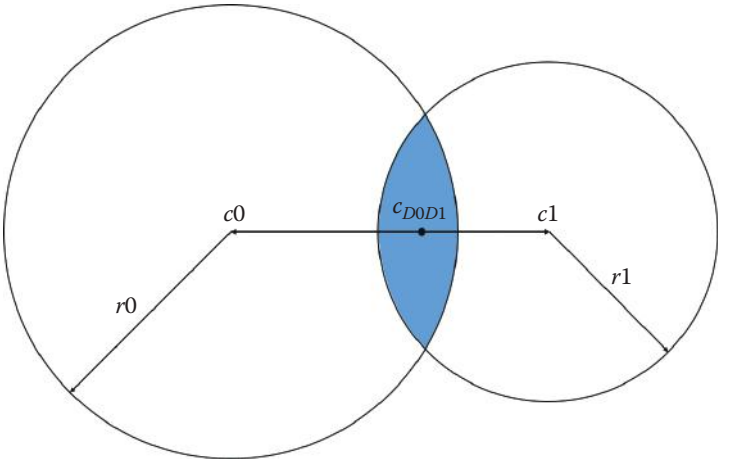

Figure 1.16. Finding area and center of intersection of disks.

The geometric center of intersection (if intersection exists) is given by

$$c_{D0D1} = c0 + (c1 - c0)\left(\frac{r0 - r1}{2\|c0 - c1\|} + 0.5\right). \tag{1.22}$$

Both equations (1.21) and (1.22) simplify under the assumption that $c0$ is at $(0, 0)$.

In the case of a rectangle-disk intersection, we treat the disk as a square adjusted to have the same area as the original disk. Therefore, we can resort to simple axis-aligned rectangle-rectangle intersection mathematics. A_{R0R1} and c_{R0R1} are given by

$$\begin{aligned} tl &= \max\left(c0 - \frac{d0}{2}, c1 - \frac{d1}{2}\right), \\ br &= \min\left(c0 + \frac{d0}{2}, c1 + \frac{d1}{2}\right), \\ A_{R0R1} &= \max\left(tl.x - br.x, 0\right)\max(tl.y - br.y, 0), \\ c_{R0R1} &= \frac{tl + br}{2}, \end{aligned} \tag{1.23}$$

where $c0$, $d0$, $c1$, and $d1$ are the center points and dimensions of rectangles $R0$ and $R1$, respectively, and tl and br are top left and bottom right intersection corners, respectively (assuming Euclidian space with inverse y; see Figure 1.17).

It is worth noting that accurate results can only be achieved for models based on radially symmetrical PDFs. If we strip away the projection part, by substituting the projected ellipse with a disk during our intersection tests and further

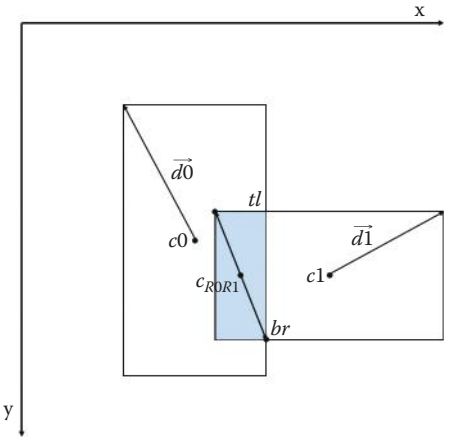

Figure 1.17. Finding the area and center of the intersection of rectangles.

integral calculation, we not only approximate the final result, but we also limit the model to radially symmetric PDFs. In the case of most microfacet BRDFs based on half vectors (equation (1.9)), the initial shape of the specular cone would be similar to an elliptical base, which would result in an ellipse shape on the light plane—thus an integral over an ellipse.

This is a rather crude approximation; however, it proved good enough in visual assessment of final results, when tested with Phong, Blinn-Phong, and GGX, using radially symmetrical light shapes (see Figures 1.18 and 1.19).

1.3.5 Nonuniform Light Sources

Up to this point we were only considering light sources with constant $L_i(p, \omega_i)$. We will now change this assumption and for simplicity assume that light intensity I is constant over the light and that $L_i(p, \omega_i)$ returns a normalized, wavelength dependent value. Looking at integrals from equations (1.11) and (1.12) and following intuitions from Sections 1.3.3 and 1.3.4, we can see that in order to acquire the correct result, we could pre-integrate equations (1.11) and (1.12) with varying $L_i(p, \omega_i)$ over the light source and then normalize by the source area. Assuming we can approximate diffuse and specular integrals at runtime, due to equations (1.16) and (1.20), we would just need to multiply those results by the pre-integrated, normalized full integral from equations (1.11) and (1.12) at our most important points for diffuse and specular integrals, respectively.

In the case of a diffuse (equations (1.11) and (1.16)), we need to integrate

$$Diffuse_{Lookup}(p_d, r) = \int_A L_i(p_d + rn_l, \omega_i)\, d\omega_i,$$

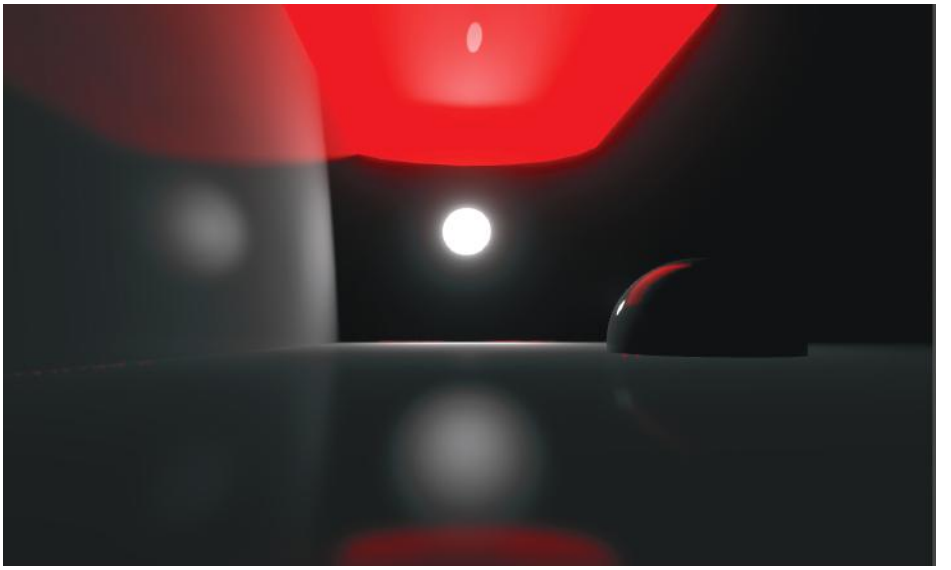

Figure 1.18. Disk area light fit to Phong distribution.

where p_d is a point on the light plane with normal n_l and r is the distance away from the light.

To simplify, we assume that occlusion does not influence the result. Then the integral can be pre-computed in a 3D lookup table indexed by $p_d.xy$ coordinates in the light space and by distance r between points p and p_d. The distance can be normalized against the maximum distance at which computing the integral still makes a visual difference. Such a distance maximum can be calculated by solving the light shape's solid angle (equation (1.3)) for r, irrespective of $cos\theta_o$, where σ sets the boundary importance of the solid angle weight:

$$r_{max} > \sqrt{\frac{dA}{\sigma}}.$$

For distances larger than r_{max}, the solid angle would be too small and, as a result, the integral would not further change visually.

Every Z layer of the lookup table would contain the diffuse integral for point p_d, set on coordinates $p_d.xy$ in the light space, $\max(r/r_{max}, 1)$ away from the light plane. Finally, the full diffuse integral approximation is

$$Diffuse\,(p, \omega_o) \sim IC_d \cos\theta_{\overrightarrow{pp_d}} \frac{dAcos\theta_o}{r^2} Diffuse_{Lookup}\,(p_d, r)\,. \tag{1.24}$$

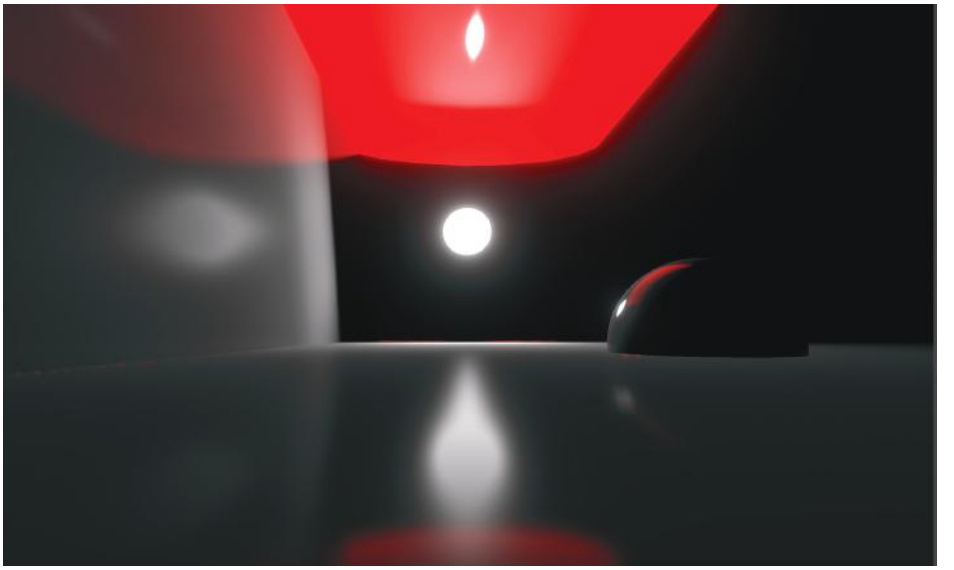

Figure 1.19. Disk area light fit to GGX distribution.

In the case of specular (equation (1.12)), we need to integrate

$$Specular_{Lookup}\left(\overrightarrow{p}\, p_{sc}, g\right) = \int_A L_i\left(p, \omega_i\right) \frac{D(\vec{h}, g) F(\omega_o, \vec{h}) G(\omega_i, \omega_o, \vec{h})}{4(cos\theta_o)} d\omega_i, \quad (1.25)$$

where p is the point being shaded and p_{sc} is the most important point of the specular integral with g defined as surface roughness. Unfortunately, such a representation would require a 4D lookup table. We would like to free ourselves from knowledge of point p. According to reasoning from Section 1.3.4, we know that the integral will effectively depend on the projected cone of importance sampling radius. We also know how to create functions to calculate the specular importance cone apex angle from the surface roughness for given BRDFs (equations (1.17) and (1.18)). The projected importance cone radius depends on distance r to the light and the cone opening angle α (see Figure 1.20). Therefore, we can calculate the integral from equation (1.25) for point p_{sc} and the new roughness g', where

$$g' = f_{BRDF}(g, r). \quad (1.26)$$

To make this assumption viable, we have to restrict only to radially symmetrical PDFs:

$$Specular_{Lookup}\left(p_{sc}, g'\right) = \int_A L_i\left(p, \omega_i\right) \frac{D(\vec{h}, g') F(\omega_o, \vec{h}) G(\omega_i, \omega_o, \vec{h})}{4(cos\theta_o)} d\omega_i, \quad (1.27)$$

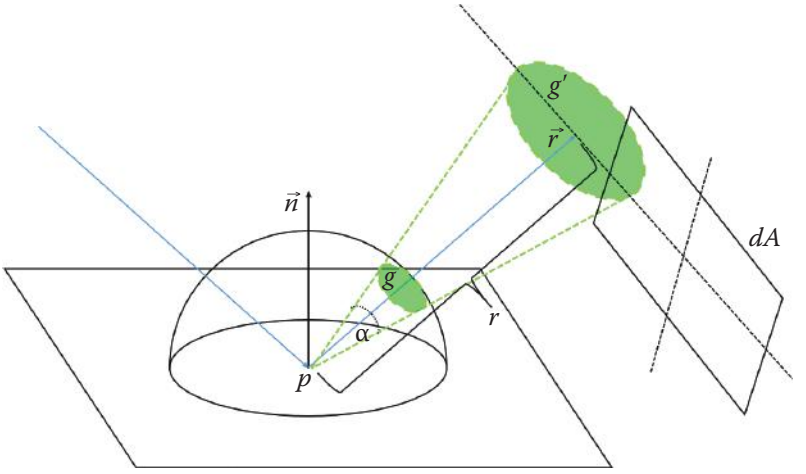

Figure 1.20. Relationship between cone angle, projected cone, and roughness.

where $\vec{h}$ is orthogonal to view direction ω_o. This effectively forces us to use Phong as our D function.

With these assumptions we can calculate equation (1.27) as a 3D lookup table indexed by $p_{sc}.xy$ coordinates in the light space and g' calculated using equation (1.26) derived for the used BRDF.

Finally, the full specular integral approximation is

$$Specular\,(p, \omega_o, g) \sim L_n \frac{D(\vec{h},)F(\omega_o, \vec{h})G(\omega_i, \overrightarrow{pp_{sc}}, \vec{h})}{4(cos\theta_o)A_S} Specular_{Lookup}(p_{sc}, g'). \tag{1.28}$$

It is worth noting that, based on choice, functions DFG might depend on g or other surface parameters. Every additional parameter, apart from already included g, would add one more dimension to our lookup table or would have to be factored out. This is entirely based on the final choices for the specular model.

1.3.6 Solving Area Light BRDF

We presented a framework for efficient derivation of area-based lighting models based on currently known and well-researched BRDFs. The final lighting model should follow all procedures from Section 1.3, with the choice of particular DFG functions and appropriate derivation of additional parameters based directly on them. Then, diffuse and specular lighting per point can be approximated for various light types (equations (1.16) and (1.20)), including colored, non-uniform lights (equations (1.24) and (1.28)). See Figures 1.21, 1.22, and 1.23.

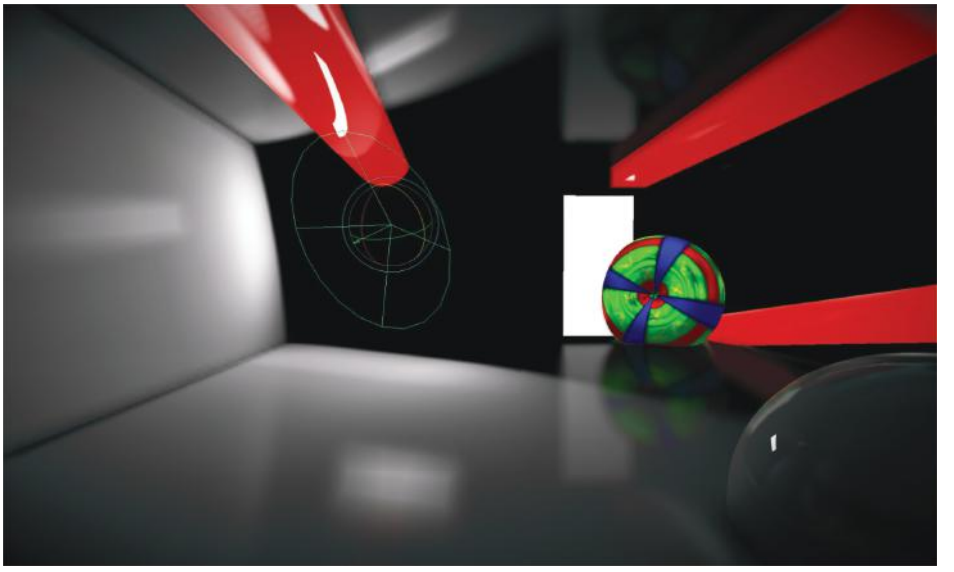

Figure 1.21. Rectangular area light.

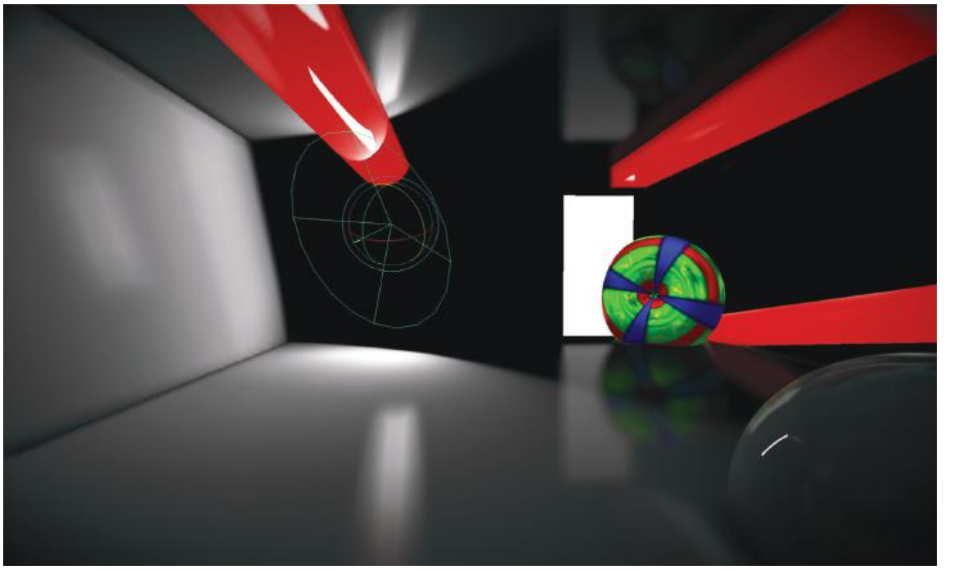

Figure 1.22. Rotated rectangular area light.

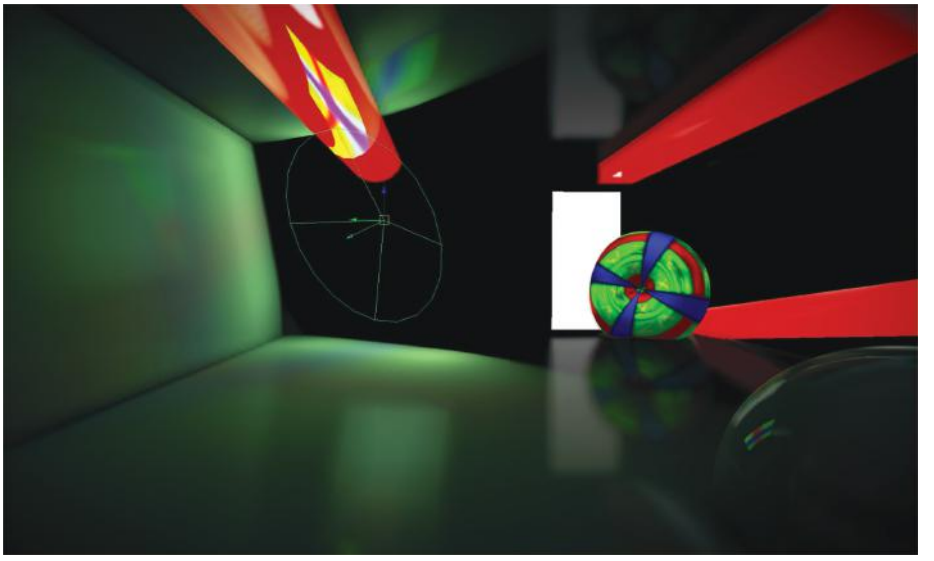

Figure 1.23. Textured rectangular area light.

1.4 Implementation

1.4.1 Introduction

Multiple ideas from this chapter were combined and used as the basis of the lighting model in *Killzone: Shadow Fall*. The light rendering engine relies heavily on real-time screen-space reflections and localized environmental reflection probes. Both solutions could efficiently support only radially symmetrical distribution-based specular (Phong). One of the dynamic texture-based lights requirements was to blend perfectly with existing image-based systems. Essentially we wanted a capability to swap dynamic area light, with image-based light card, with cube-map reflection at a distance without visual artifacts. We also wanted to support analytical shape lights allowing the highest visual quality and advanced material reflectance models.

1.4.2 Motivation

We used Cook-Torrance BRDF as the basis for our reflectance model. Normalized Blinn-Phong was used as the distribution function. Fresnel and geometric terms were calculated according to Smith-Schlick approximations [Schlick 94], matched and normalized to Blinn-Phong. We also prepared a radially symmetrical version of this BRDF. It was refitted to use Phong as the base, normalized and matched as closely as possible to reference solution.

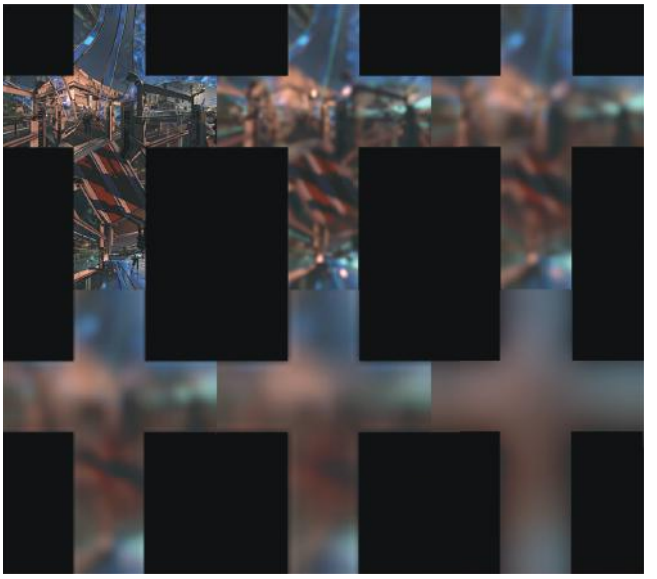

Figure 1.24. Pre-integrated integral of environmental cube-map.

All lights use the same diffuse integral approximation reasoning shown in Section 1.3.3. Every light, depending on the light shape function, implements equation (1.16) or equation (1.24) in the case of a textured light source, to approximate the diffuse integral. That includes sunlight, omnidirectional lights, and spot lights with sphere, disk, or rectangular light shapes.

Standard analytical dynamic lights, depending on the light shape function, implement equation (1.20) to approximate the specular integral. In the case of textured area lights, we support only rectangular lights with Phong-based BRDF.

Phong-based Cook-Torrance was also used for image-based lighting convolutions and texture-based area lights. Similar to already-known methods for cube map generation [Lazarov 11], we build our kernel based on our Phong-based BRDF and generate multiple mip-map levels of cube-maps for different specular cone widths (see Figure 1.24). Similar reasoning, described mathematically in Section 1.3.5, was applied to generate image-based light cards used with textured rectangular area.

Different light shape implementation. The *Killzone* renderer supports the following light shapes: sphere, rectangle, and disk. All lights follow the framework set by the disk light implementation. They require only minor changes to the code responsible for the closest point to the shape or normalized solid angle. Therefore, we were able to efficiently share code between different light shapes.

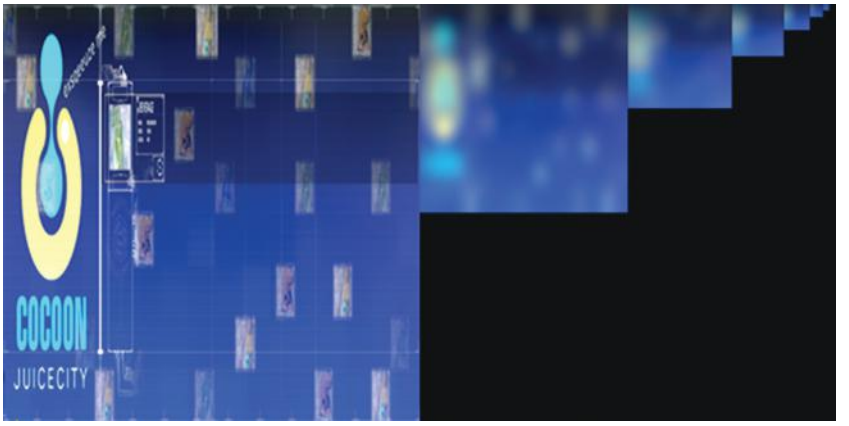

Figure 1.25. Pre-integrated specular integral of light card used with texture area lights.

It is also worth noting that sphere shape diffuse integral implementation can be vastly simplified by analytically integrating the integral as it has a closed form dependent only on the light position and radius. The sphere specular integral can also be optimized by working exclusively with the solid angle differential. There is no need to project computation into the light space, as the sphere shape is easily defined in 3D. Also, the projected solid angle is equal to the solid angle, as a sphere always subtends the same solid angle, independent of viewing direction and orientation.

Nonuniform light source implementation. We support texture-based lights using a rectangle shape as the base. We apply reasoning from Section 1.3.5 to directly compute and store pre-computed data in mip-maps of a 2D texture. Initial light intensity is provided by the texture generated by the in-game impostor rendering system or hand-painted by artists. Computed values are normalized and stored in two compressed RGB8 format textures (see Figures 1.25 and 1.26). At runtime, the rectangle light evaluates diffuse and specular integral, and multiplies the result by pre-computed partial integral from textures indexed in a similar fashion as described in Section 1.3.5

1.5 Results Discussion

Using area lights exclusively during the production of *Killzone: Shadow Fall* proved to be an important pillar of the physically based rendering pipeline. Due to the proposed model, we were able to unify lighting incoming from different sources—analytical or image based—keeping a similar response and quality. Our art production pipeline was able to produce assets in a hermetic review environ-

Figure 1.26. Pre-integrated diffuse integral of light card used with texture area lights.

ment, prepared as HDR pre-integrated BRDF cube-maps, and expect similar, high-quality results in game (see Figures 1.27 and 1.28).

Also due to the implicit split of material and lighting handling, teams could efficiently work on level assets, without interference between lighting and environmental departments. Previously, that kind of clash was unavoidable, as environmental artists had a tendency to tweak assets to simulate physically larger lights (i.e., by changing roughness). This habit resulted many times in duplicated assets per different in-game scenarios.

Figure 1.27. Typical usage of analytical area lights.

Figure 1.28. Typical usage of textured area lights.

The area lighting model proved to be efficient on the next generation console Playstation 4 GPU. The model itself is heavy on ALU operations (see Table 1.1), which in our case helped to amortize waiting time for multiple texture fetches from bandwidth intensive G-buffer and shadow maps. Therefore, we did not experience a noticeable slowdown in comparison with more standard point-based light models such as Blinn-Phong.

Unfortunately, several shortcuts we took during the model derivation resulted in artifacts, under certain circumstances, when using Blinn-Phong–based BRDF rectangular area lights. Due to those cases, we decided to use the rectangle shape exclusively with Phong-based BRDF, which is a better fit for the used approximations in the case of nonradially symmetrical light shapes.

Blinn-Phong Lights	Scalar ALU	% Against Base
Point	222	0%
Sphere	252	+13%
Disk	361	+62%
Rectangle	382	+72%
Texture Rectangle	405	+82%

Table 1.1. Comparision of Different light shaders assembly. Counts includes full deferred shadowed light code.

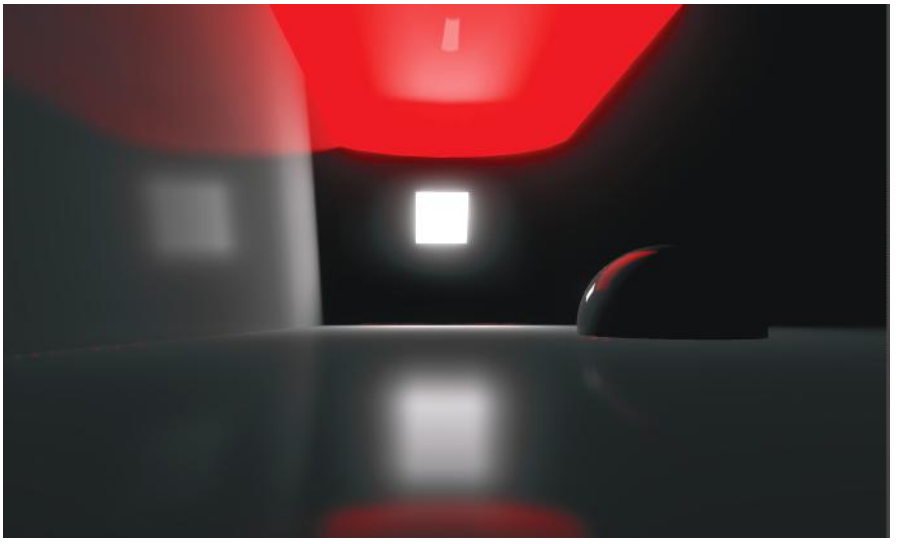

Figure 1.29. Square light shape using Phong distribution.

1.6 Further Research

In the near future we would like to focus on enhancing the normalization part of the specular integral, which currently, due to disk approximation (Section 1.3.4), is lacking in visual quality when compared to the reference solution. We would be especially interested in pursuing a good approximation for Blinn-Phong and therefore other microfacet distributions with a similar base such as the recently popular GGX.

In order to do so, we would need to find an efficient way of finding intersections between rotated ellipses and disks or rectangles.

We also feel that the rectangular lights specular approximation, when using nonradially symmetrical PDFs, needs more research. As of now, the current algorithm for the most important point might result in severe artifacts at a rotation angle of light that is orthogonal to the anisotropy direction of the PDF. A proper search algorithm would require taking anisotropy direction into account. It is worth noting that even in its current form, artifacts might be visually bearable for rectangular shapes close to square (see Figures 1.29 and 1.30).

One proposed heuristic would be to treat the projected specular cone as an elongated rectangle, oriented and extending in the direction of anisotropy. Such shape change could be dynamic, calculated for specific BRDFs. Then, the most important point for specular sampling would be the geometric center of the intersection between the specular rectangle and the light shape. The intersection area could also be used as A_s for normalization (see equation (1.20)).

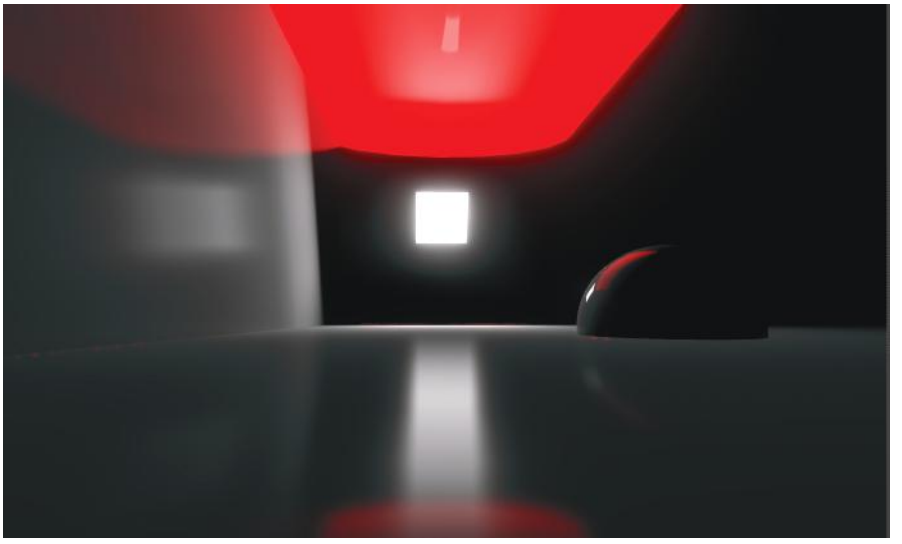

Figure 1.30. Square light shape using GGX distribution exhibiting minor visual artifacts.

We prototyped this approach, for rectangular lights, by analytically finding a polygon representing the light shape and the projected specular cone intersection. To find the intersection at runtime, we used a GPU-optimized version of the Liang-Barsky clipping algorithm [Liang and Barsky 84]. A light shape rectangle was treated as the clipping window, then a polygon representing the projected specular cone was clipped against it. The end result was an array of vertices representing a planar, non-self-intersecting polygon. In order to find the polygon's area, we used the Gauss formula for arbitrary polygon area.

Unfortunately, a busy production schedule has prevented further research in that promising direction. Figure 1.31 shows the proposed idea geometrically with reference shader-based solution results (Figure 1.32).

Another proposed area of research includes nonradially symmetrical PDFs support for texture-based area lights. We experimented with multisampling approaches and approximating the PDF at runtime; however, it proved to be too expensive for our real-time budgets.

1.7 Conclusion

We provided a tested framework for developing various physically based, area-based lighting models using currently researched BRDFs as the starting point. Implementation proved to be robust and efficient enough for the rendering engine of *Killzone: Shadow Fall*, a launch title for Playstation 4.

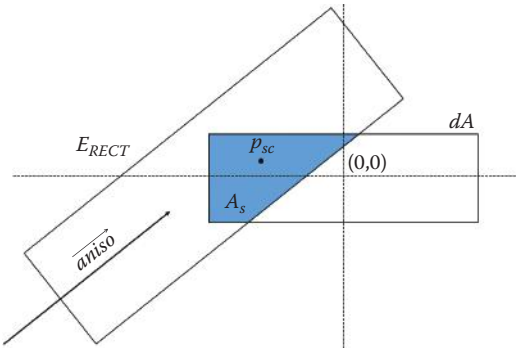

Figure 1.31. Geometric construction of point p_{sc} respecting distribution anisotropy direction. A_S due to Liang-Barsky clipping algorithm.

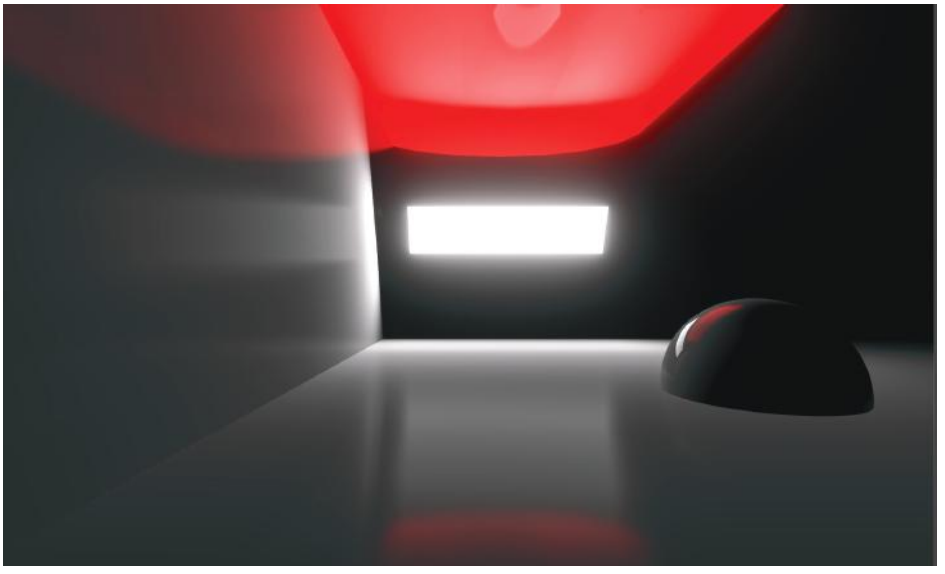

Figure 1.32. Runtime shader results using method from Figure 1.31.

We hope that the proposed ideas will provide other researchers and developers with tools to tackle problems of area lights and further extend the concept. In future, we hope that area-based lighting will become the de facto standard in modern video games, providing unified, predictable lighting solutions supplementing physically based material reflectance models.

Bibliography

[Baker and Hill 12] D. Baker and S. Hill "Rock-Solid Shading." SIGGRAPH 2012, Advances in Real-Time Rendering in 3D Graphics and Games Course, Los Angeles, CA, August 8, 2012.

[Beckmann and Spizzichino 63] Petr Beckmann and André Spizzichino. *The Scattering of Electromagnetic Waves from Rough Surfaces*. Oxford, UK: Pergamon Press, 1963. (Republished by Artech House in 1987.)

[Burley 12] B. Burley. "Physically-Based Shading at Disney." SIGGRAPH 2012, Practical Physically Based Shading in Film and Game Prduction Course, Los Angeles, CA, August 8, 2012.

[Cook and Torrance 81] R. Cook and K. Torrance. "A Reflectance Model for Computer Graphics." *Computer Graphics (Siggraph 1981 Proceedings)* 15:3 (1981), 301–316.

[Lambert 60] Johann Heinrich Lambert. *Photometria, sive De mensura et gradibus luminus, colorum et umbrae*. Augsburg, 1760.

[Lazarov 11] D. Lazarov. "Physically-Based Lighting in *Call of Duty: Black Ops*." SIGGRAPH 2011, Advances in Real-Time Rendering in 3D Graphics and Games Course, Vancouver, Canada, August, 2011.

[Liang and Barsky 84] Y.D. Liang and B. Barsky. "A New Concept and Method for Line Clipping," *ACM Transactions on Graphics* 3:1 (1984), 1–22.

[Mittring and Dudash 11] M. Mittring and B. Dudash. "The Technology Behind the DirectX 11 Unreal Engine 'Samaritian' Demo." Presentation, Game Developers Conference 2011, San Francisco, CA, March, 2011.

[Mittring 12] M. Mittring. "The Technology Behind the 'Unreal Engine 4 Elemental Demo'." SIGGRAPH 2012, Advances in Real-Time Rendering in 3D Graphics and Games Course, Los Angeles, CA, August, 2012.

[Oat 06] C. Oat. "Aperture Ambient Lighting." *ACM Siggraph 2006 Courses*, pp. 143–152. New York: ACM, 2006.

[Pharr and Humphreys 04] M. Pharr and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. San Francisco, CA: Morgan Kaufman, 2004.

[Schlick 94] C. Schlick. "An Inexpensive BRDF Model for Physically-Based Rendering." *Proc. Eurographics '94, Computer Graphics Forum* 13:3 (1994), 233–246.

[McAuley et al. 13] S. McAuley et al. "Physically Based Shading in Theory and Practice." In *ACM SIGGRAPH 2013 Courses*, Article no. 22. New York: ACM, 2013.

[Walter et al. 07] B. Walter, S.R. Marschner, H. Li, and K.E. Torrance. "Microfacet Models for Refraction through Rough Surfaces." In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, pp. 195–206. Aire-la-Ville, Switzerland: Eurographics Association, 2007.