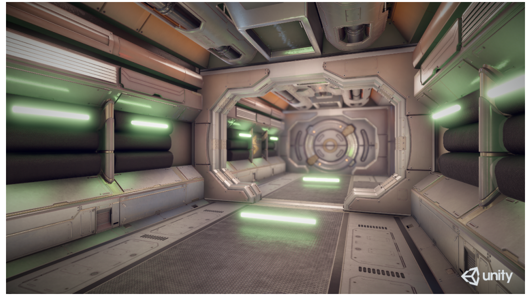
Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

Eric Heitz

unity

Stephen Hill

LUCASFILM Ltd

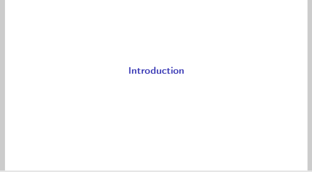


Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

⚠

This presentation contains many animated slides that do not display well with all PDF viewers. We recommend using Adobe Acrobat Reader.

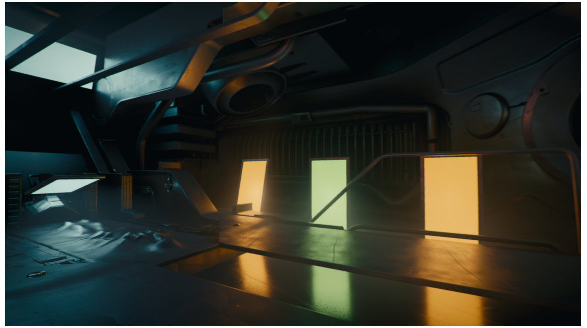
Introduction

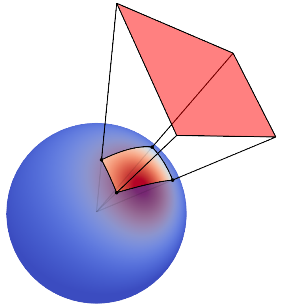


Introduction

Introduction

Last year: real-time polygonal-light shading with LTC

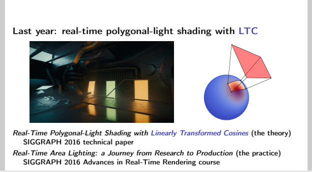




Real-Time Polygonal-Light Shading with Linearly Transformed Cosines (the theory)
SIGGRAPH 2016 technical paper

Real-Time Area Lighting: a Journey from Research to Production (the practice)
SIGGRAPH 2016 Advances in Real-Time Rendering course

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



Last year: real-time polygonal-light shading with LTC

Real-Time Polygonal-Light Shading with Linearly Transformed Cosines (the theory)
SIGGRAPH 2016 technical paper
Real-Time Area Lighting: a Journey from Research to Production (the practice)
SIGGRAPH 2016 Advances in Real-Time Rendering course

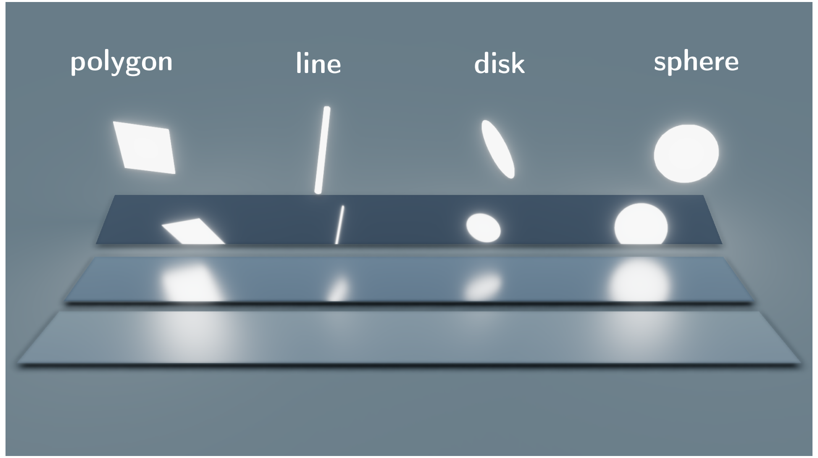
Last year, at SIGGRAPH 2016, we presented a new technique for real-time polygonal-light shading.

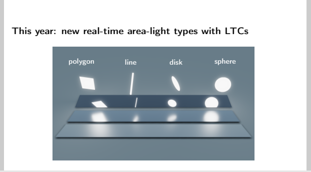
The cornerstone of this technique is the new spherical distribution Linearly Transformed Cosine (LTC), introduced in our technical paper:
`https://labs.unity.com/article/real-time-polygonal-light-shading-linearly-transformed-cosines`

Practical details and optimizations were discussed in our talk in the Advances in Real-Time Rendering course:
`http://blog.selfshadow.com/publications/s2016-advances/`

Introduction

This year: new real-time area-light types with LTCs



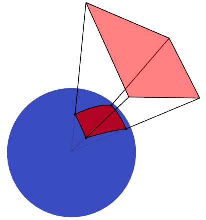


This year: new real-time area-light types with LTCs

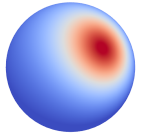
This year, we extend this area-lighting framework by introducing new area-light primitives such as lines, disks and spheres.

Introduction

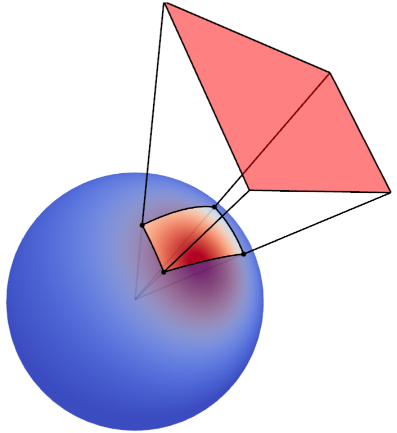
Last year: LTC integration for polygons



spherical polygon

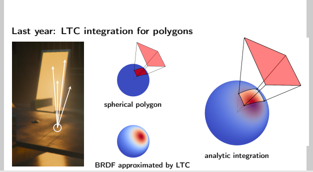


BRDF approximated by LTC



analytic integration

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

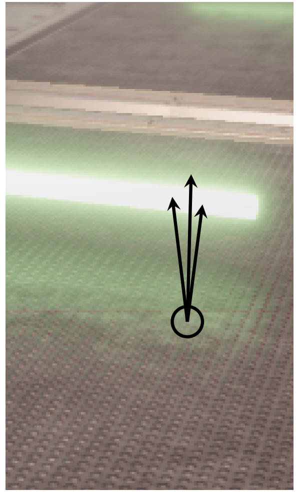


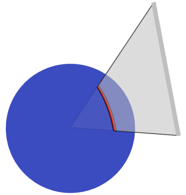
The shading is the computation of the integral of the BRDF over the spherical domain covered by the light. The main property of an LTC is that it can be analytically integrated over spherical polygons, which makes it a useful tool for polygonal-light shading.

This actually how the intuition and the formula of an LTC originated: we crafted this distribution to make sure that it would be easy to integrate over polygonal light sources. This distribution was originally meant for this specific light type.

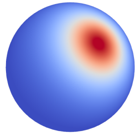
Introduction

This year: LTC integration for other primitives

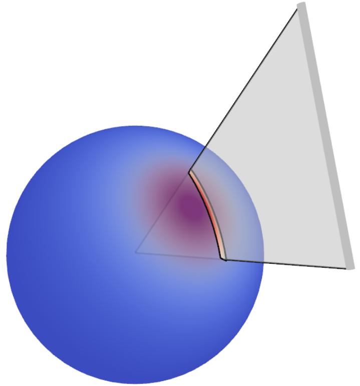




spherical segment

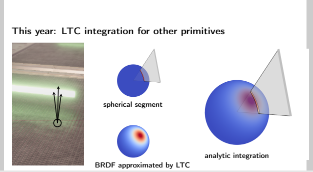


BRDF approximated by LTC



analytic integration

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



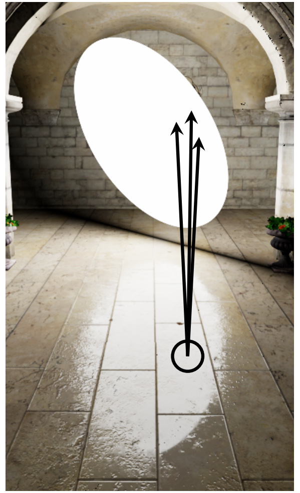
In order to support more light types, we had to investigate whether LTCs could also be integrated over other kind of spherical domains.

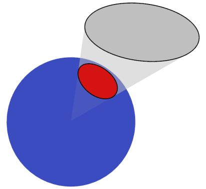
In this slide we can see the case of line lights, which produce spherical segments. Segments are conceptually close to polygons – they are defined by two vertices instead of more – but still, integrating an LTC over a line is not done in the same way as over a polygon.

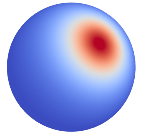
We were able to work out an analytic integration formula for this case, which looks like the integration formula for polygons but with some differences.

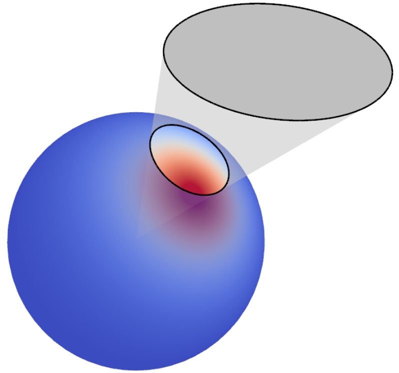
Introduction

This year: LTC integration for other primitives

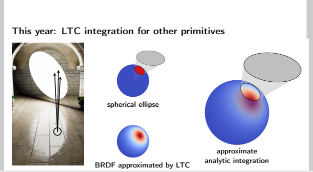



spherical ellipse


BRDF approximated by LTC


approximate
analytic integration

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines


This year: LTC integration for other primitives

spherical ellipse

BRDF approximated by LTC

approximate analytic integration

In the case of sphere or disk lighting, the spherical domain to consider is a spherical ellipse. Even though the problem looks very similar to the one for polygons or lines, spherical ellipses involve a very different kind of math.

In this case, we could not find an exact analytic integration formula, but we worked out one that we find accurate enough.

Organization of the talk

- Recap of LTCs (see material from last year for more details)
- Line lights
- Sphere/Disk lights

Organization of the talk

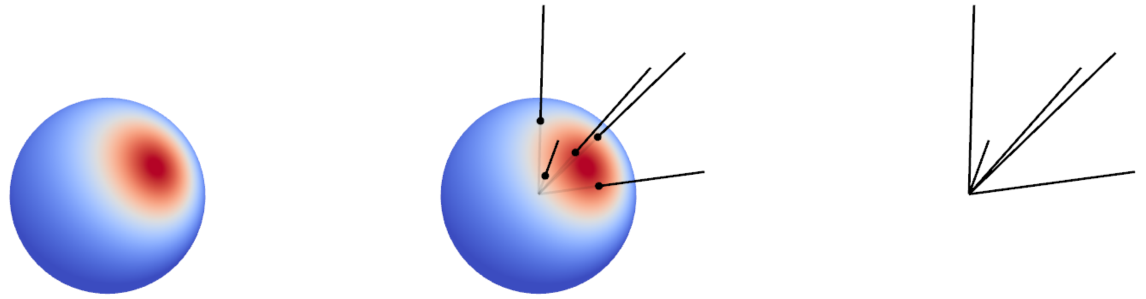
- Recap of LTCs (see material from last year for more details)
- Line lights
- Sphere/Disk lights

In this talk, we will start with by recapping the background of LTCs (check last year's material for more details) and their integration over polygons, which is required for understanding the new stuff. Then, we will talk about the new lights types: lines, spheres and disks.

Recap of LTCs

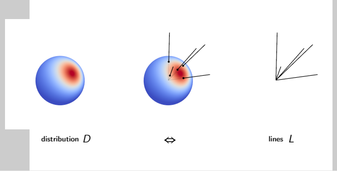
Recap of LTCs

Recap of LTCs



distribution D $\Leftrightarrow$ lines L

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



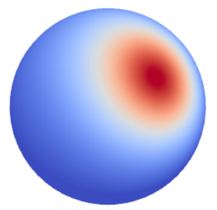
In order to understand how an LTC is defined, we use the following intuition: a spherical distribution is equivalent to the infinite set of samples that can be chosen from this distribution.

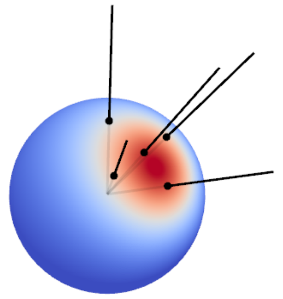
- If we have a spherical distribution, we can generate an infinity of samples in this distribution.
- If we have an infinity of samples, we can reconstruct the distribution to arbitrary precision.

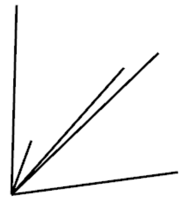
The distribution and the samples are simply two different descriptions of the same mathematical object.

Note: for illustration purposes, we've only drawn five samples in the figure, but remember that conceptually we are talking about an infinite set of samples. In the following, just assume that $5 = \infty$:-)

Recap of LTCs

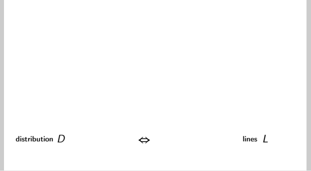






distribution D $\Leftrightarrow$ lines L

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



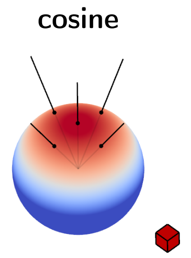
Since they are the same object, changing one changes the other. If we change the parameters of the distribution, the samples follow. Reciprocally, if we move the samples, the reconstructed distribution follows.

Because the distribution and the samples are dual descriptions of the same mathematical object, it means that a property associated with the distribution usually has a dual property associated with the samples, and vice versa.

The idea of LTCs is to use linear transformations (3×3 matrices) that change the samples' directions and hence the distribution.

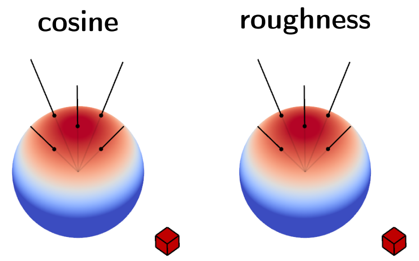
⚠This slide is animated (works with Acrobat Reader).

Recap of LTCs

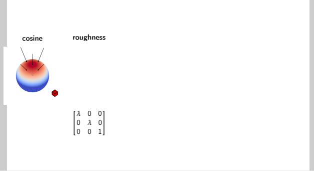


First, we start with this classic cosine distribution. In this figure we overlaid five samples to make it easier to visualize the effect of the linear transformation.

Recap of LTCs



$$\begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

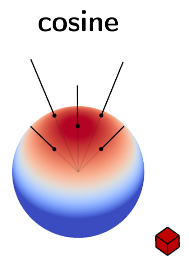


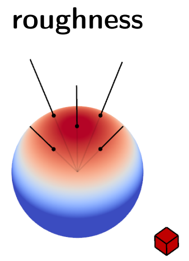
If we apply a scaling transform only on the *xy*-plane, we can see that the samples are going to be compressed towards the average direction of the distribution. This is how we create Phong-like distribution starting from a cosine.

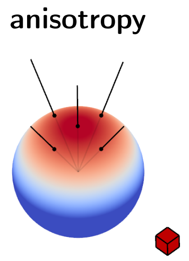
To aid the visualization, we also show a unit cube undergoing the same linear transformation.

⚠This slide is animated (works with Acrobat Reader).

Recap of LTCs

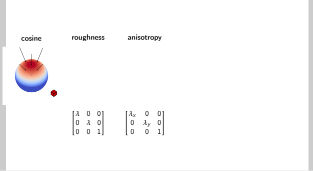






$$\begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} \lambda_x & 0 & 0 \\ 0 & \lambda_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

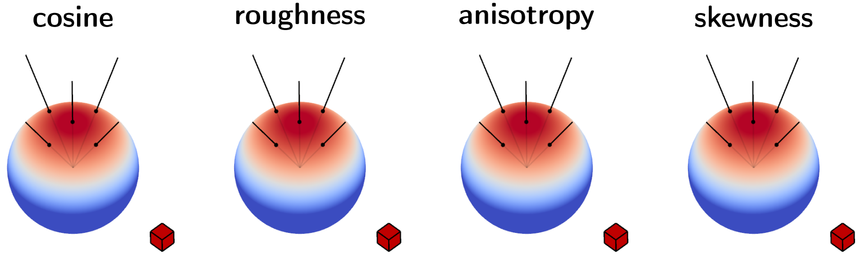
Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



Applying a scaling transform with different magnitudes in the x and y directions, we can see that the samples are going to be compressed more in one direction, which is introducing anisotropy to the distribution.

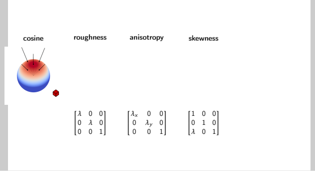
⚠This slide is animated (works with Acrobat Reader).

Recap of LTCs



$$\begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \lambda_x & 0 & 0 \\ 0 & \lambda_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \lambda & 0 & 1 \end{bmatrix}$$

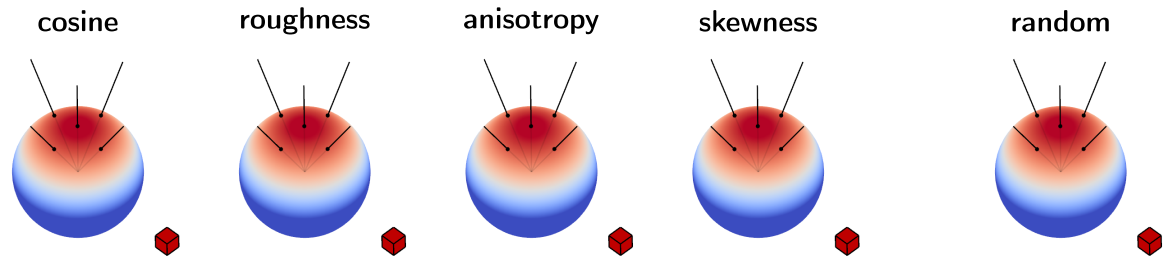
Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



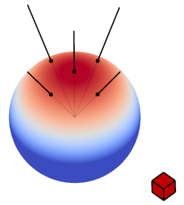
Applying a shear transform spreads the samples out on one side of the distribution and compresses them on the opposite side. This is introducing skewness to the distribution.

⚠This slide is animated (works with Acrobat Reader).

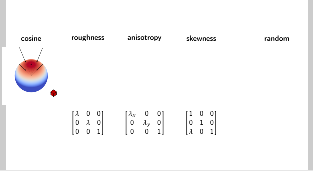
Recap of LTCs



$$\begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \lambda_x & 0 & 0 \\ 0 & \lambda_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \lambda & 0 & 1 \end{bmatrix}$$



Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



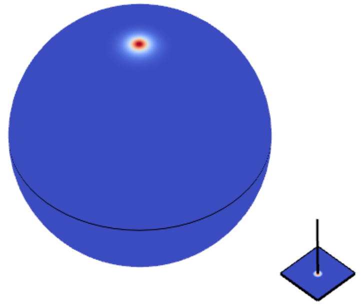
Finally, by combining all those effects in random matrices, we can create sophisticated spherical distributions with funny shapes.

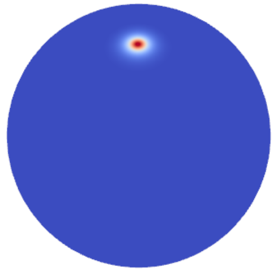
⚠This slide is animated (works with Acrobat Reader).

Recap of LTCs

BRDF (Smith GGX)

Linearly Transformed Cosines (fitted)





Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

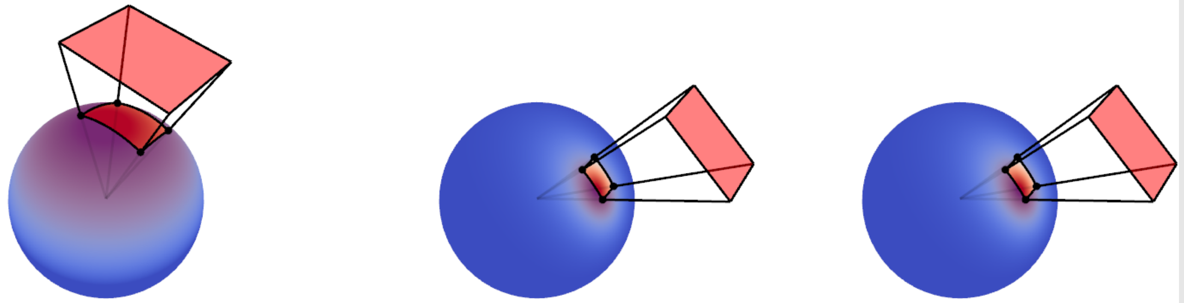
BRDF (Smith GGX) Linearly Transformed Cosines (fitted)

With this definition, Linearly Transformed Cosines offer a wide appearance space that can be used to efficiently approximate physically based BRDFs, especially the GGX one, which is used in many game engines today. Of course, the approximation is not perfect, but it reproduces the main features of the BRDF for different roughness and incidence configurations.

⚠This slide is animated (works with Acrobat Reader).

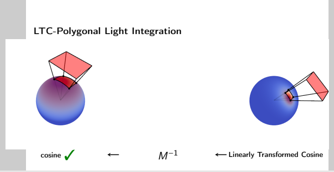
Recap of LTCs

LTC-Polygonal Light Integration



cosine ✓ $\longleftarrow$ M^{-1} $\longleftarrow$ Linearly Transformed Cosine

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

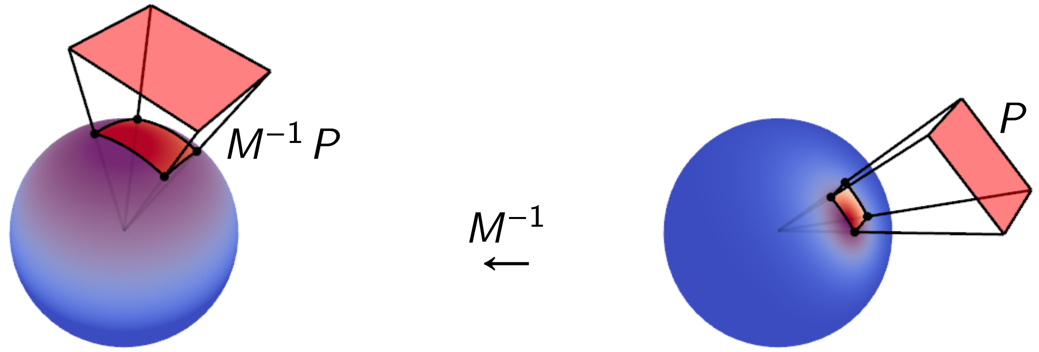


From this definition, we also obtain the algorithm for integrating LTCs over polygons. If M is the linear transformation used to obtain this LTC distribution from a cosine, we apply the inverse linear transformation M^{-1} on both the LTC distribution and the polygon. We obtain the original cosine and a new polygon.

⚠This slide is animated (works with Acrobat Reader).

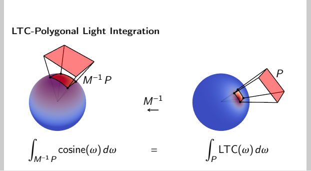
Recap of LTCs

LTC-Polygonal Light Integration



$$\int_{M^{-1}P} \text{cosine}(\omega)\, d\omega \quad = \quad \int_{P} \text{LTC}(\omega)\, d\omega$$

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

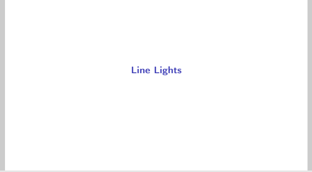


The important property is that the integral of the LTC over the polygon is exactly the integral of polygon transformed by M^{-1} and the cosine, which can be evaluated analytically (it's the irradiance of the new polygon and there is a closed-form expression for this).

That's it for the recap of LTCs as presented last year.

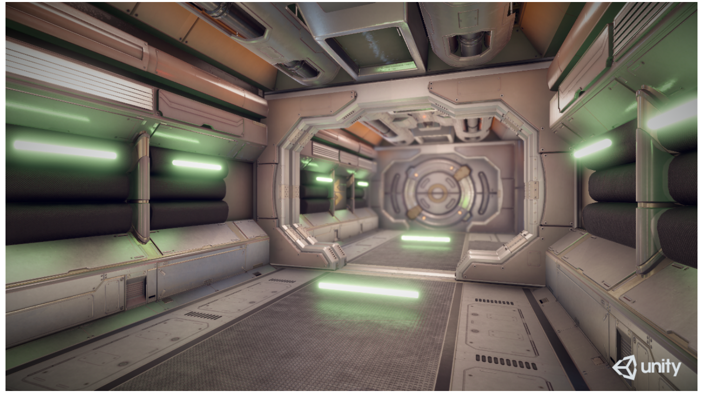
At this point, the open question was: can we do something similar for other light types?

Line Lights


Line Lights

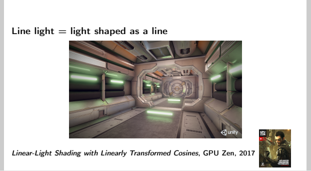
Line Lights

Line light = light shaped as a line



Linear-Light Shading with Linearly Transformed Cosines, GPU Zen, 2017

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines


Line light = light shaped as a line

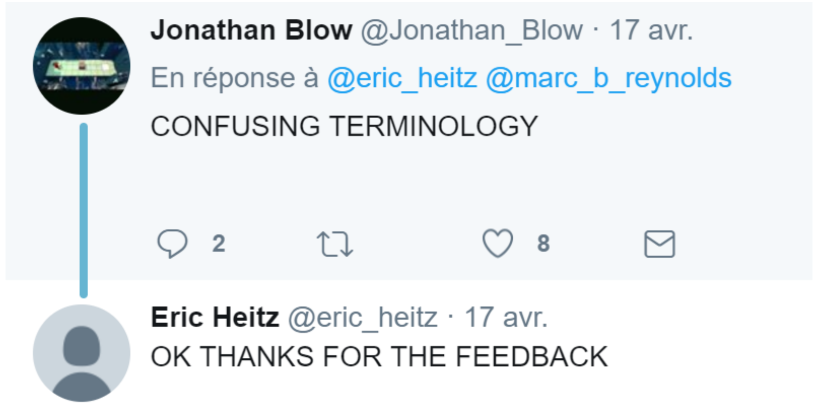
Linear-Light Shading with Linearly Transformed Cosines, GPU Zen, 2017

The first new light type we will discuss is the line light.

The part of this talk dedicated to line lights is also the topic of a chapter in the GPU Zen book. A free preprint of our chapter is available at `https://labs.unity.com/article/linear-light-shading-linearly-transformed-cosines`.

Line Lights

“Linear Lights” confusion (linear vs. gamma lighting)

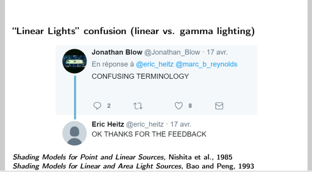


Jonathan Blow @Jonathan_Blow · 17 avr.
En réponse à @eric_heitz @marc_b_reynolds
CONFUSING TERMINOLOGY

2 8

Eric Heitz @eric_heitz · 17 avr.
OK THANKS FOR THE FEEDBACK

Shading Models for Point and Linear Sources, Nishita et al., 1985
Shading Models for Linear and Area Light Sources, Bao and Peng, 1993



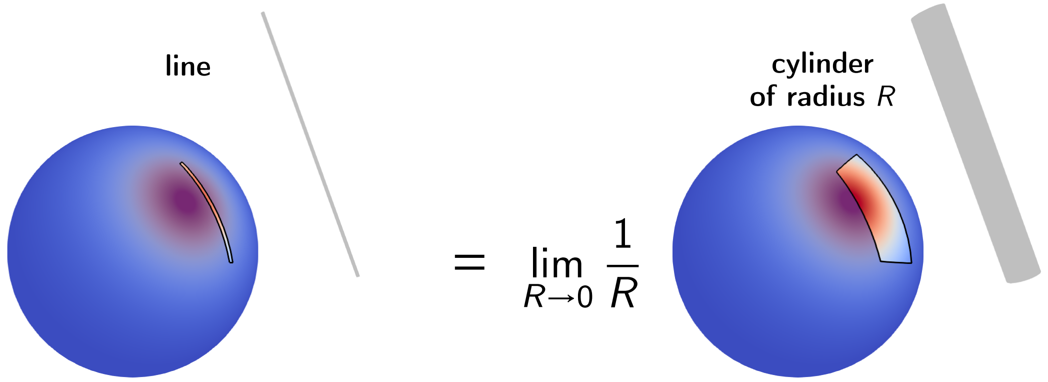
“Linear Lights” confusion (linear vs. gamma lighting)
Jonathan Blow @Jonathan_Blow · 17 avr.
En réponse à @eric_heitz @marc_b_reynolds
CONFUSING TERMINOLOGY
2 8
Eric Heitz @eric_heitz · 17 avr.
OK THANKS FOR THE FEEDBACK
Shading Models for Point and Linear Sources, Nishita et al., 1985
Shading Models for Linear and Area Light Sources, Bao and Peng, 1993

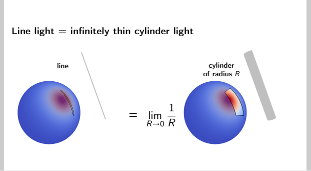
Note that in the book chapter we call them “linear lights” to be consistent with the existing literature on the topic. However, when we released it, several people complained that the name was confusing (as they understand linear lighting as opposed to gamma).

It’s too late to change that in the book chapter, but from now on we propose to call them “line lights” instead, to avoid this confusion.

Line Lights

Line light = infinitely thin cylinder light

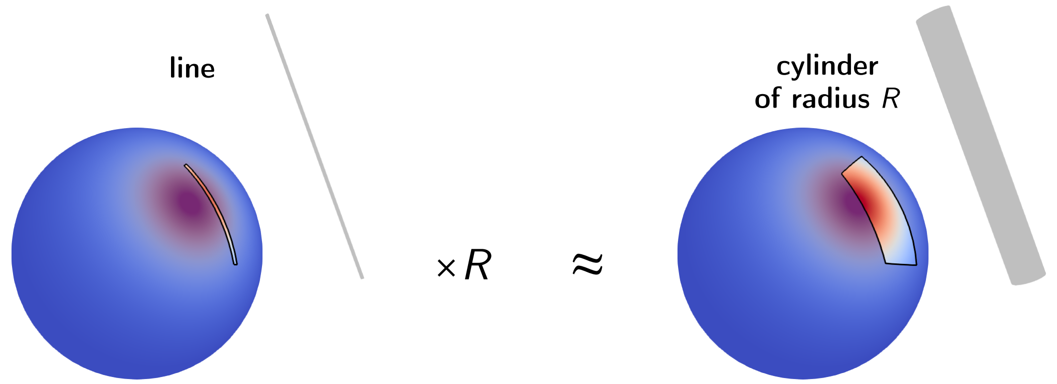




A line light is defined as an infinitely thin cylinder light. Formally, the shading obtained with a line light is the limit of the ratio of the shading obtained with a cylinder and the radius of the cylinder when the radius tends toward zero.

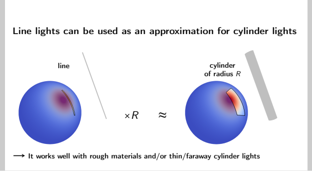
Line Lights

Line lights can be used as an approximation for cylinder lights



$\longrightarrow$ **It works well with rough materials and/or thin/faraway cylinder lights**

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

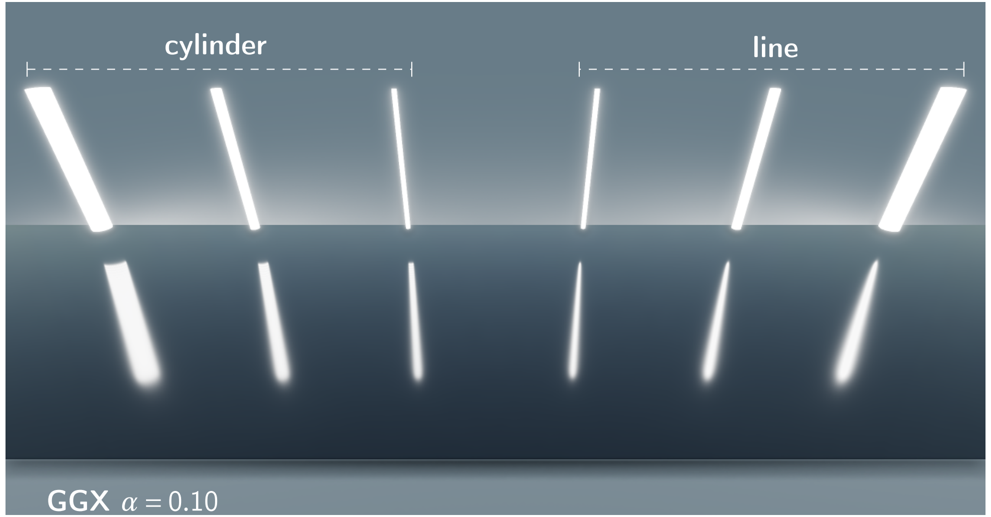

Line lights can be used as an approximation for cylinder lights
$\longrightarrow$ It works well with rough materials and/or thin/faraway cylinder lights

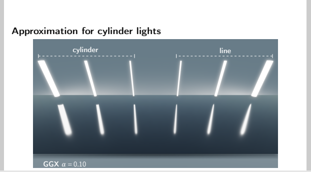
That means that the shading of a cylinder lights can be approximated by the shading of a line light multiplied by the radius of the cylinder. This is a first-order approximation of the cylinder's shading based on its values at the center of its spherical domain.

Hence, this approximation only works when the variation of the distribution is small inside the spherical domain covered by the cylinder, i.e. when either

- the cylinder is thin,
- the cylinder is faraway from the shading point,
- the material is rough.

Approximation for cylinder lights

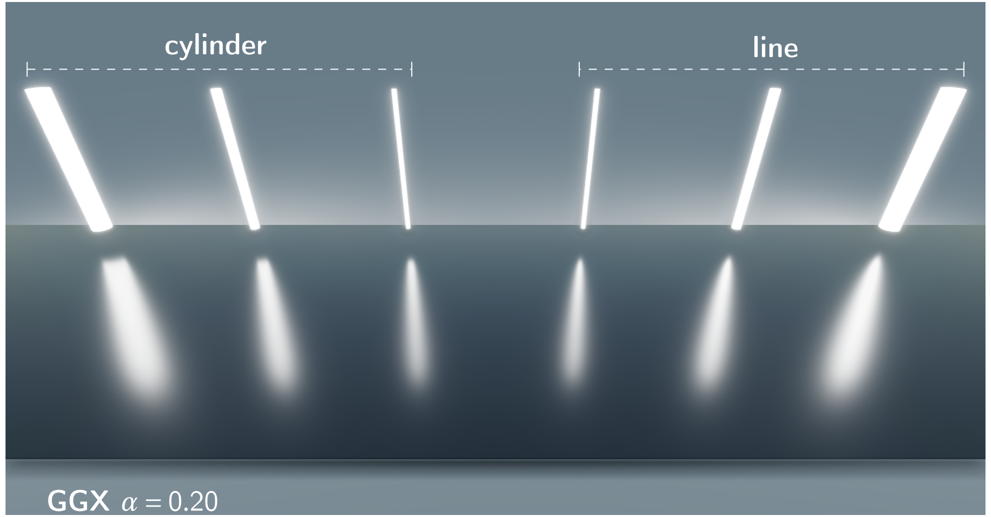


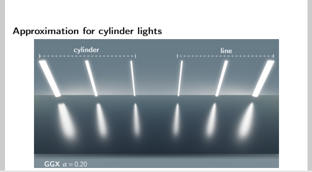


In the following examples, we compare a reference result obtained with MC integration over a cylinder and the analytic line approximation.

We can see that the rougher the material, the thinner the cylinder, or the farther the shading point, the better the approximation.

Approximation for cylinder lights



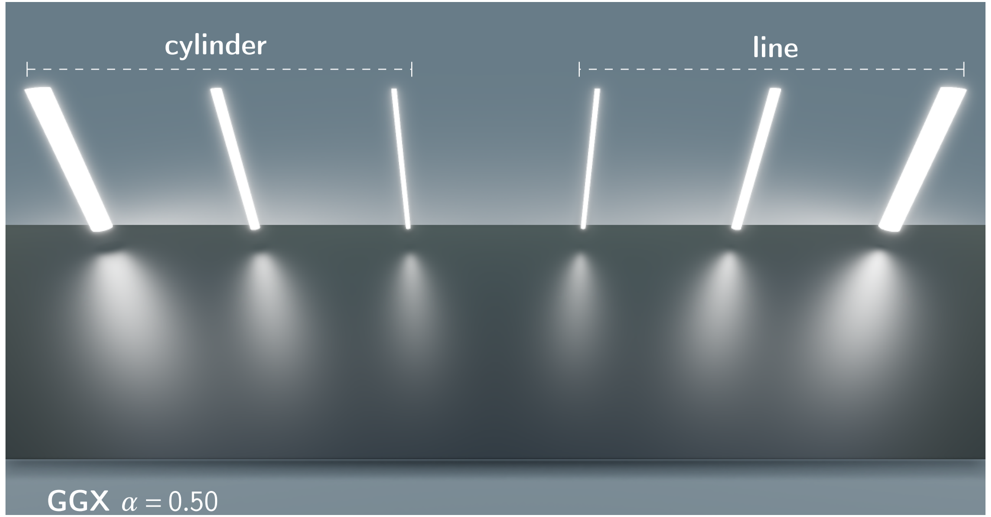


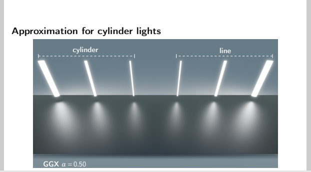
In the following examples, we compare a reference result obtained with MC integration over a cylinder and the analytic line approximation.

We can see that the rougher the material, the thinner the cylinder, or the farther the shading point, the better the approximation.

Line Lights

Approximation for cylinder lights

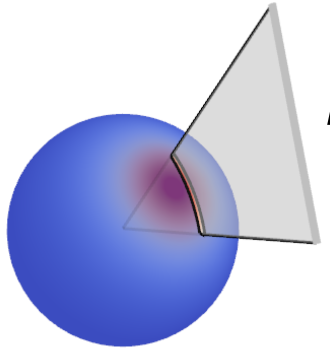




In the following examples, we compare a reference result obtained with MC integration over a cylinder and the analytic line approximation.

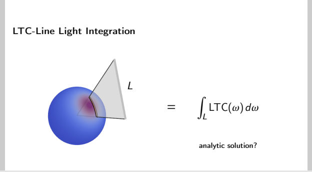
We can see that the rougher the material, the thinner the cylinder, or the farther the shading point, the better the approximation.

LTC-Line Light Integration



$$= \int_L \mathrm{LTC}(\omega)\, d\omega$$

analytic solution?



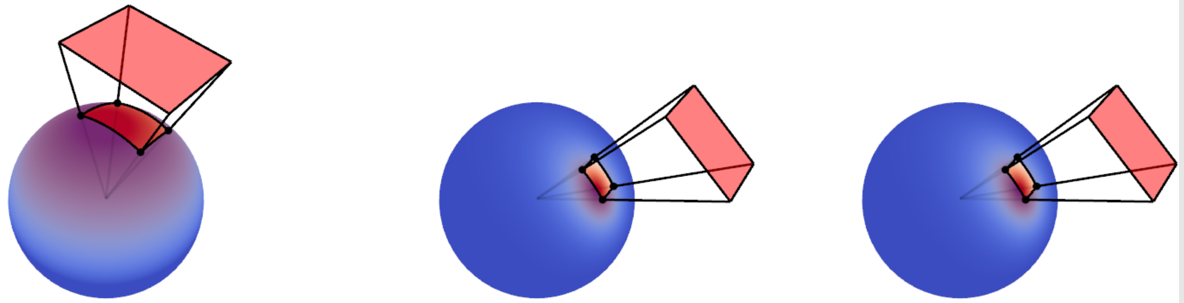
LTC-Line Light Integration

$$= \int_L \mathrm{LTC}(\omega)\, d\omega$$

analytic solution?

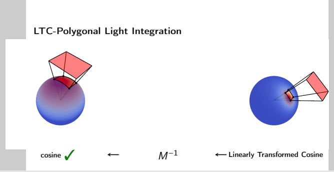
In order to use line lights in our LTC framework, the technical problem to solve is the integration of LTCs over line lights.

LTC-Polygonal Light Integration



cosine ✓ $\longleftarrow$ M^{-1} $\longleftarrow$ Linearly Transformed Cosine

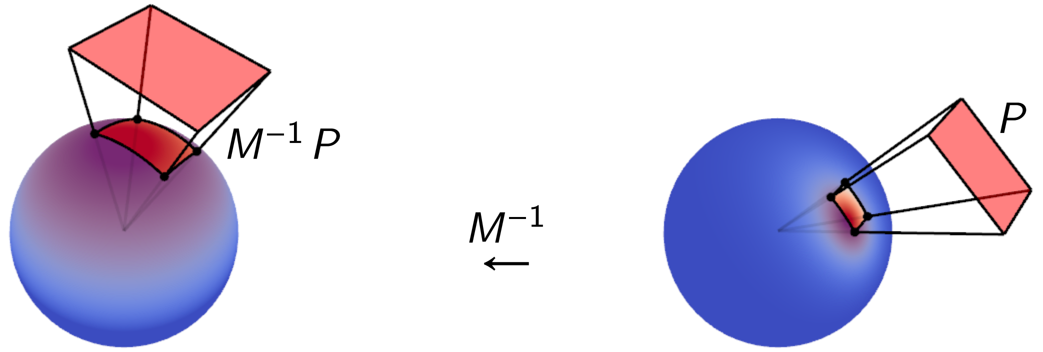
Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines


LTC-Polygonal Light Integration
cosine ✓
$\longleftarrow$
M^{-1}
$\longleftarrow$ Linearly Transformed Cosine

With polygonal lights, the trick is to multiply the polygon by the inverse matrix M^{-1} in order to go back to the original cosine configuration.

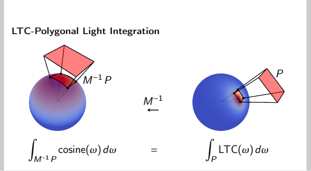
Line Lights

LTC-Polygonal Light Integration



$$\int_{M^{-1}P} \text{cosine}(\omega)\, d\omega \quad = \quad \int_{P} \text{LTC}(\omega)\, d\omega$$

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

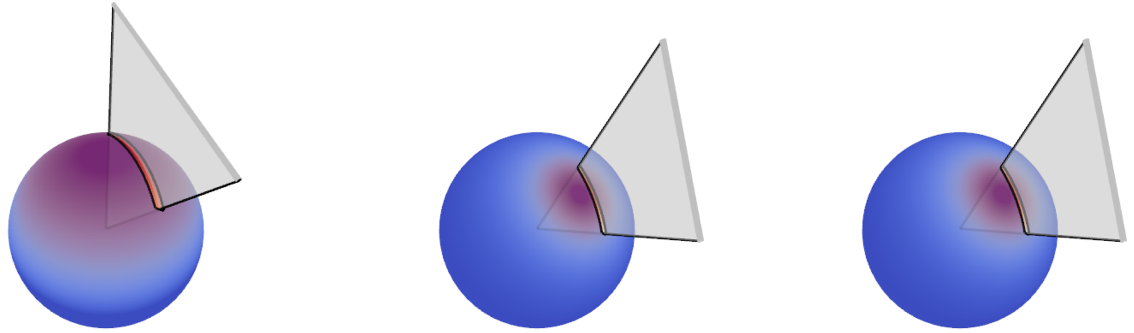

LTC-Polygonal Light Integration

$$\int_{M^{-1}P} \text{cosine}(\omega)\, d\omega = \int_{P} \text{LTC}(\omega)\, d\omega$$

In the original cosine configuration, we just need to integrate the polygon over the cosine, which is simple to do since there is an analytic solution for this.

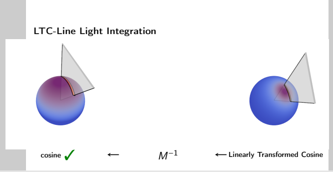
Line Lights

LTC-Line Light Integration



cosine ✓ $\longleftarrow$ M^{-1} $\longleftarrow$ Linearly Transformed Cosine

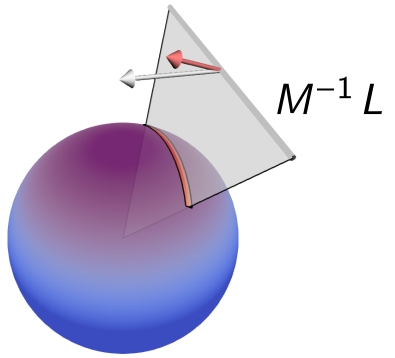
Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



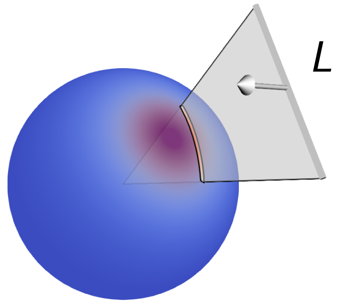
The integration over line lights is performed in the same way: we apply the inverse linear transformation to the line light and we obtain another line light in the cosine configuration.

Line Lights

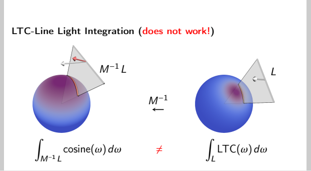
LTC-Line Light Integration (does not work!)



$$\xleftarrow{M^{-1}}$$



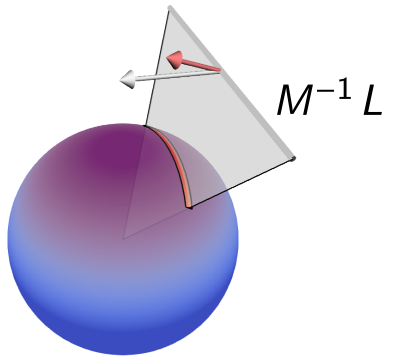
$$\int_{M^{-1}\,L} \text{cosine}(\omega)\, d\omega \neq \int_{L} \text{LTC}(\omega)\, d\omega$$



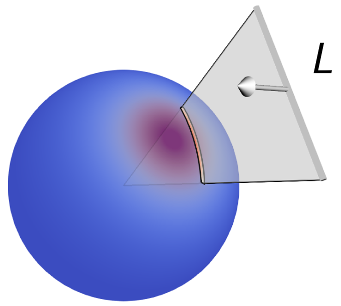
However, this does not work. The integral in the cosine configuration does not match the integral in the LTC configuration.

Line Lights

LTC-Line Light Integration (infinitesimal thickness change)

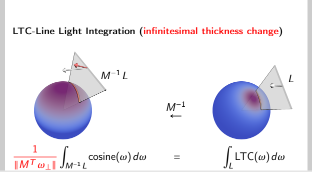


$\xleftarrow{M^{-1}}$



$$\frac{1}{\|M^T \omega_\perp\|} \int_{M^{-1} L} \text{cosine}(\omega)\, d\omega \quad = \quad \int_L \text{LTC}(\omega)\, d\omega$$

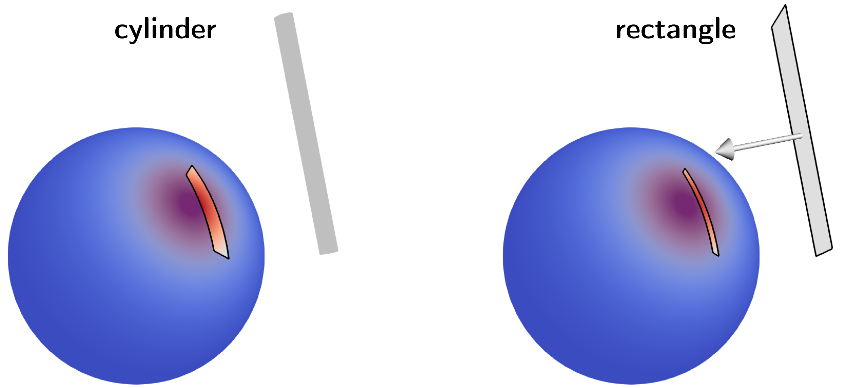
Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



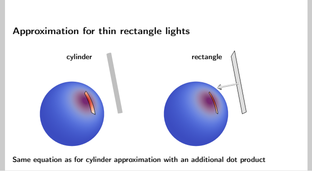
There is one subtlety: the result we are looking for is not just the integral of the cosine over the transformed line light. The problem is that even though the line light is infinitely thin, it still has a virtual infinitely small thickness, which is affected by the linear transformation. We need to account for this change to obtain the correct answer. Fortunately, it can be obtained by multiplying the result by the factor $\frac{1}{\|M^T \omega_\perp\|}$, where $\omega_\perp$ is the direction orthonormal to both the light and the direction towards the light. The amount this direction $\omega_\perp$ is affected by the linear transformation yields how much the infinitely small thickness changes.

More details about this result and its proof are provided in the associated book chapter.

Approximation for thin rectangle lights



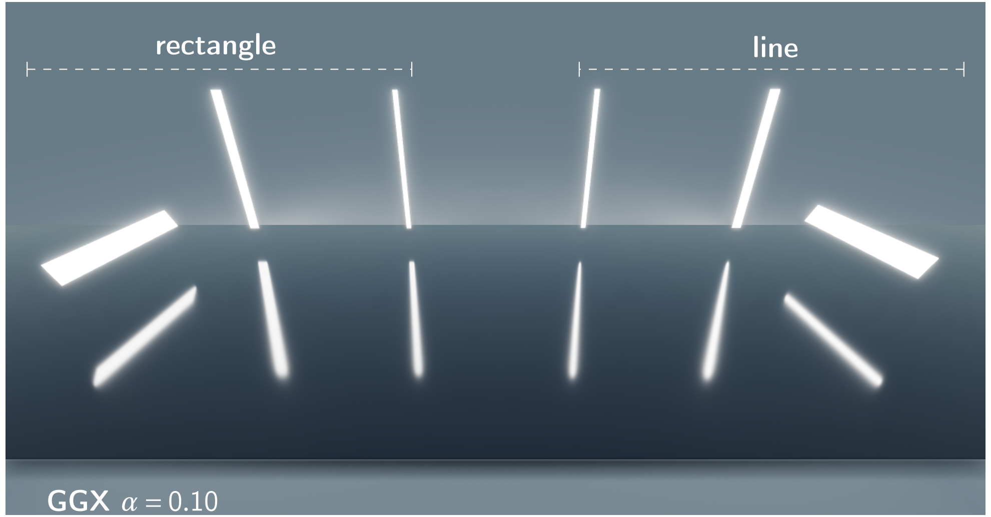
Same equation as for cylinder approximation with an additional dot product



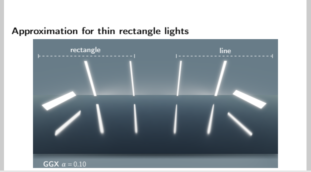
Besides cylinders, line lights can also be used as an approximation for thin rectangle lights. We show that the result is simply obtained by modulating the shading of the cylinder light with the cosine of the orientation of the thin rectangle light.

The line-light approximation for thin rectangle lights works best under the same conditions as for cylinders: high roughness, or thin lights, or faraway lights.

Approximation for thin rectangle lights



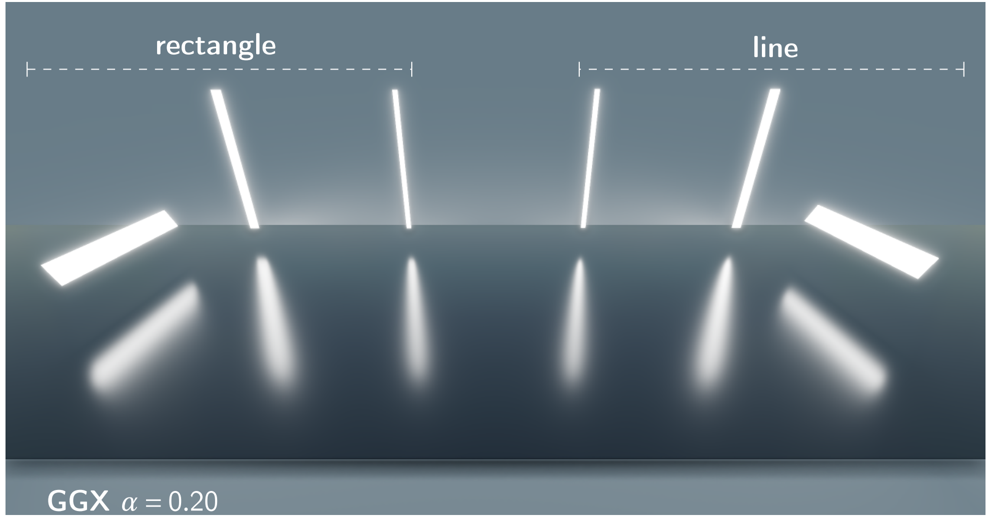
Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

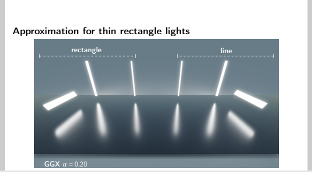


In the following examples, we compare a reference result obtained with MC integration over a rectangle light and the analytic line approximation.

We can see that the rougher the material, the thinner the rectangle, or the farther the shading point, the better the approximation.

Approximation for thin rectangle lights

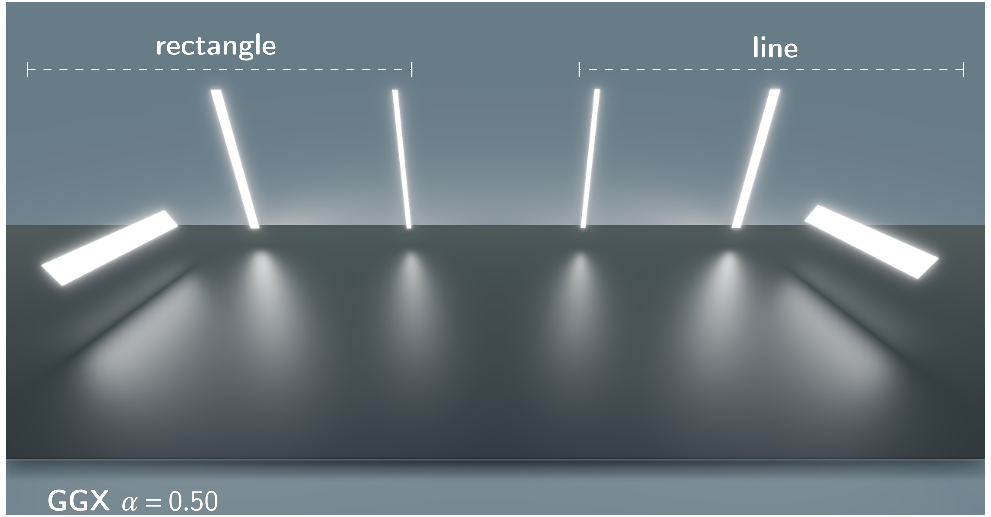


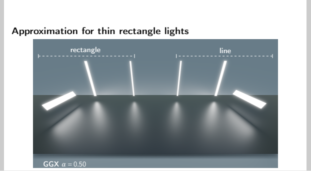


In the following examples, we compare a reference result obtained with MC integration over a rectangle light and the analytic line approximation.

We can see that the rougher the material, the thinner the rectangle, or the farther the shading point, the better the approximation.

Approximation for thin rectangle lights





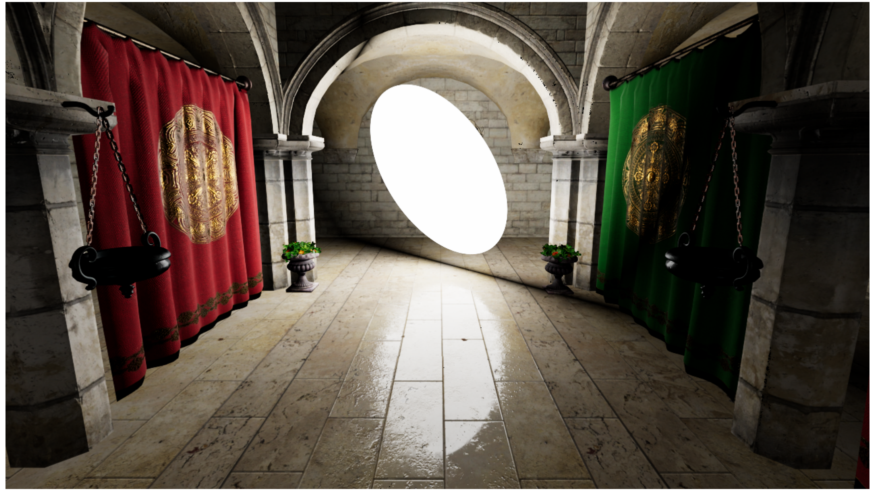
In the following examples, we compare a reference result obtained with MC integration over a rectangle light and the analytic line approximation.

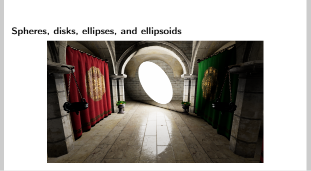
We can see that the rougher the material, the thinner the rectangle, or the farther the shading point, the better the approximation.

Sphere/Disk Lights

Sphere/Disk Lights

Spheres, disks, ellipses, and ellipsoids



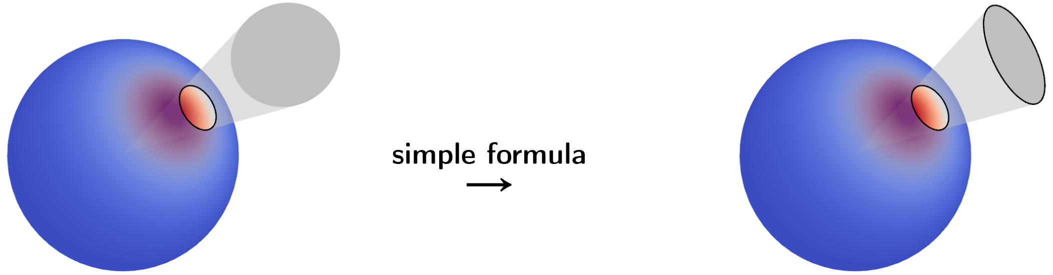


Spheres, disks, ellipses, and ellipsoids

The next light types we will investigate are spheres and disks.

Sphere/Disk Lights

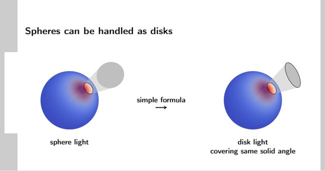
Spheres can be handled as disks



sphere light

disk light
covering same solid angle

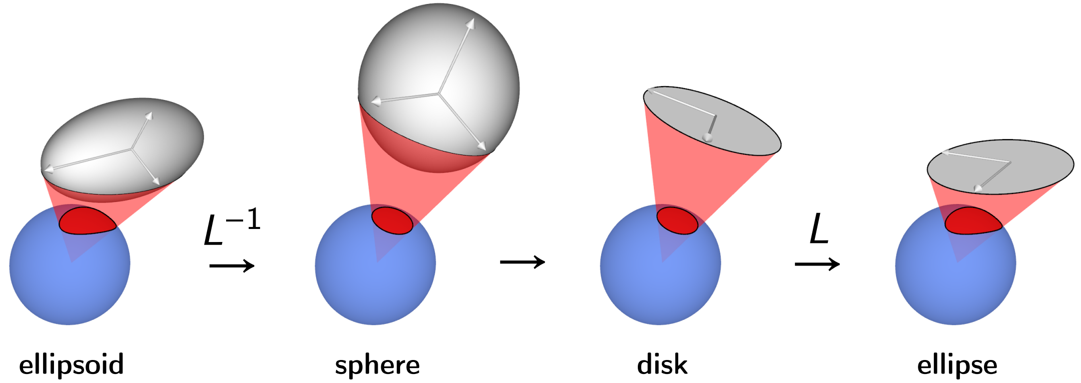
Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



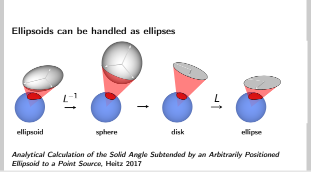
The first observation is that solving for sphere lights is a subset of solving for disk lights. Indeed, a sphere can always be replaced by a disk that covers the same solid angle. There is a simple analytic formula to do this.

Sphere/Disk Lights

Ellipsoids can be handled as ellipses



Analytical Calculation of the Solid Angle Subtended by an Arbitrarily Positioned Ellipsoid to a Point Source, Heitz 2017



Ellipsoids can be handled as ellipses

ellipsoid sphere disk ellipse

Analytical Calculation of the Solid Angle Subtended by an Arbitrarily Positioned Ellipsoid to a Point Source, Heitz 2017

The second observation is that, similarly, solving for ellipsoid lights is a subset of solving for ellipse lights. Indeed, an ellipsoid can always be replaced by an ellipse that covers the same solid angle.

We recently published our method for computing this ellipse in the journal *Nuclear Instruments and Methods in Physics Research*. A free preprint of this paper is available at

`https://labs.unity.com/article/analytical-calculation-solid-angle-subtended-arbitrarily-positioned-ellipsoid-point-source.`

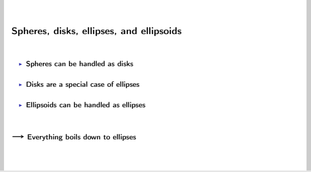
Note that the solution is directly inspired by our LTC intuition: transform linearly to a simpler problem where the solution is known, then transform back.

Sphere/Disk Lights

Spheres, disks, ellipses, and ellipsoids

- Spheres can be handled as disks
- Disks are a special case of ellipses
- Ellipsoids can be handled as ellipses

$\longrightarrow$ Everything boils down to ellipses


Spheres, disks, ellipses, and ellipsoids

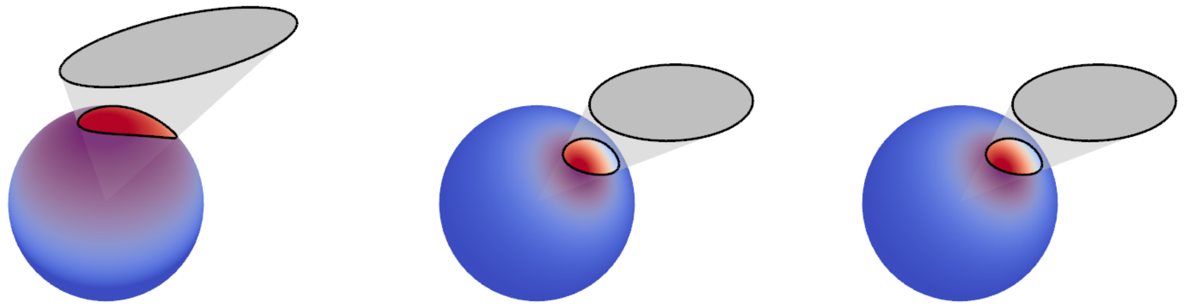
- Spheres can be handled as disks
- Disks are a special case of ellipses
- Ellipsoids can be handled as ellipses

$\longrightarrow$ Everything boils down to ellipses

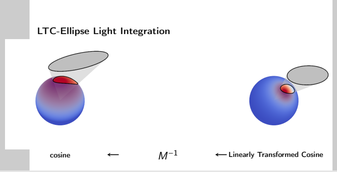
As a result, everything boils down to integrating LTCs over ellipses. This is the problem we will focus on.

Sphere/Disk Lights

LTC-Ellipse Light Integration



cosine $\longleftarrow$ M^{-1} $\longleftarrow$ Linearly Transformed Cosine

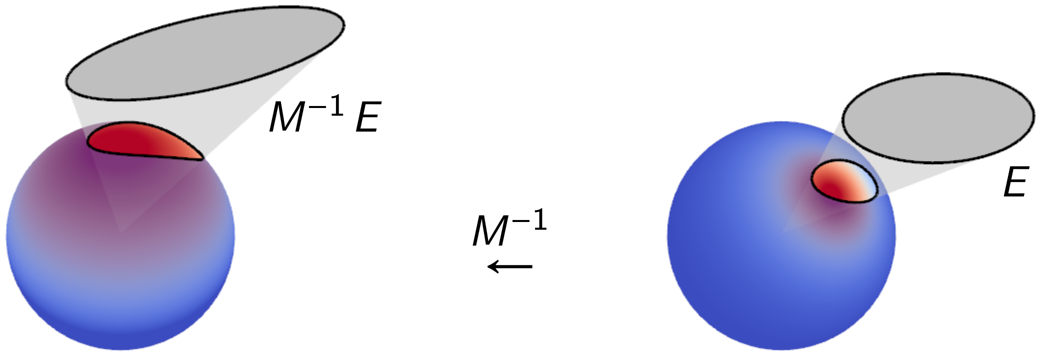


We use the same trick as for polygons and lines: we apply the inverse linear transform M^{-1} to the ellipse to go back to the original cosine configuration.

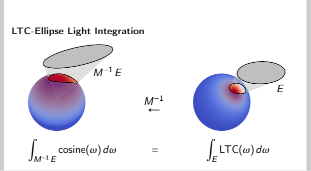
Note that a linearly transformed ellipse remains an ellipse.

⚠This slide is animated (works with Acrobat Reader).

LTC-Ellipse Light Integration



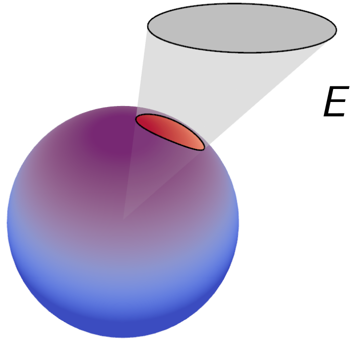
$$\int_{M^{-1}E} \text{cosine}(\omega)\, d\omega \quad = \quad \int_E \text{LTC}(\omega)\, d\omega$$



As for the polygon, the integral of the LTC over the ellipse is the integral of the cosine over the new ellipse, which defines a spherical ellipse.

Sphere/Disk Lights

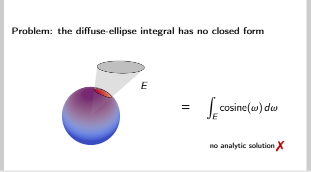
Problem: the diffuse-ellipse integral has no closed form



$$= \int_E \text{cosine}(\omega)\, d\omega$$

no analytic solution ✗

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

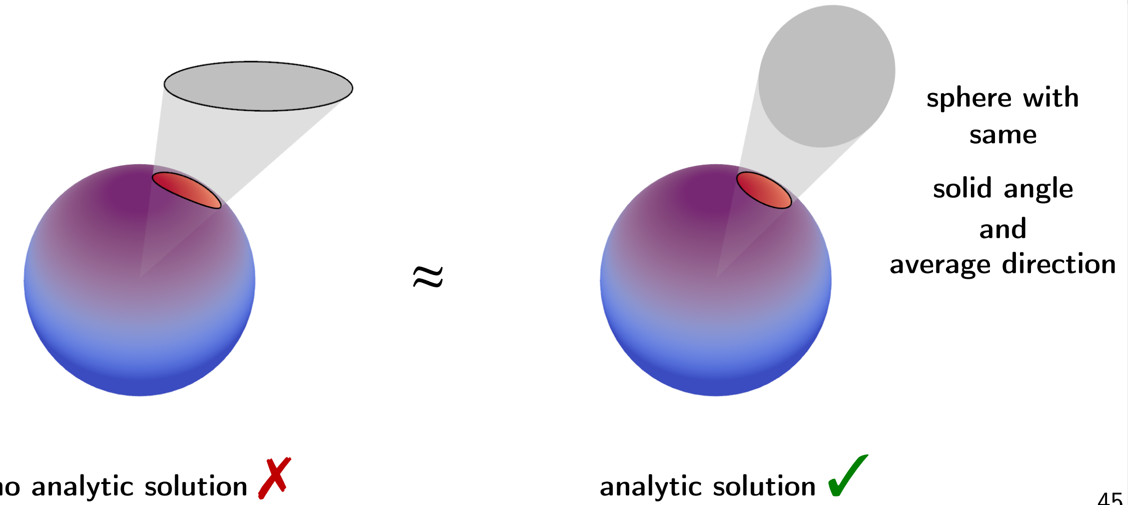


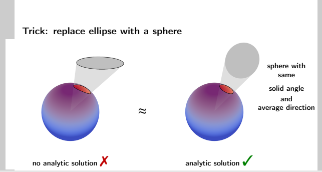
There is a problem though: the cosine distribution does not have an analytic integral over a spherical ellipse like it has for spherical polygons.

(Except in special cases – more on this later.)

Sphere/Disk Lights

Trick: replace ellipse with a sphere





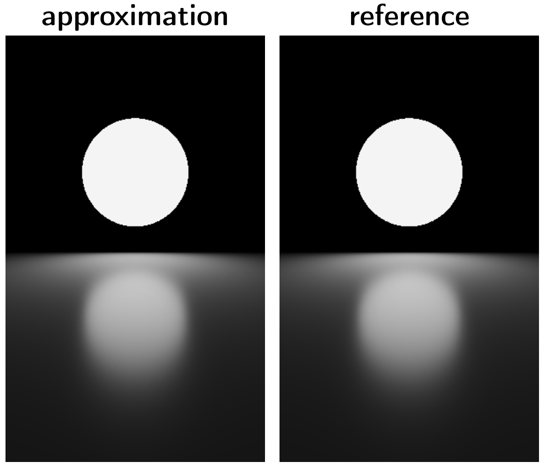
In order to overcome this problem, we approximate the ellipse with a sphere that covers a solid-angle domain of the same area and with the same average direction. The advantage is that the cosine can be integrated analytically over the sphere.

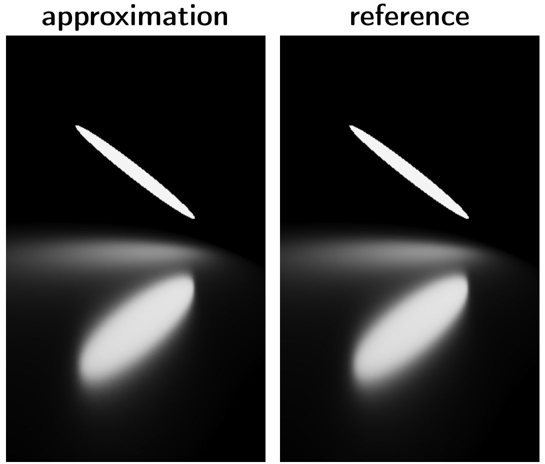
Note that this approximation needs to occur after the linear transformation of the ellipse by matrix M^{-1}, once we are solving a cosine-ellipse integration problem.

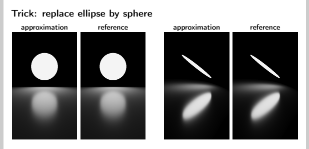
Note: for optimization purposes, we already used the same approximation for polygons, since it is cheaper than using their exact analytic integral (which, in the general case, involves clipping). For more details, see our presentation: `http://blog.selfshadow.com/publications/s2016-advances/`

Sphere/Disk Lights

Trick: replace ellipse by sphere





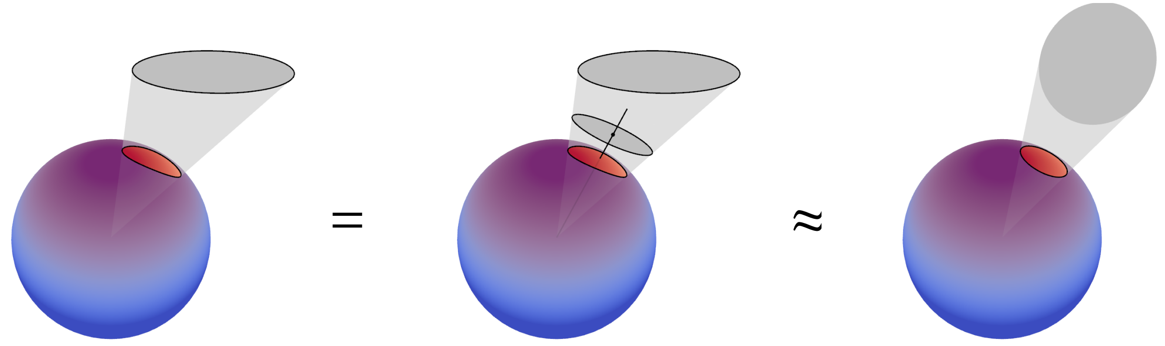


In practice, the bias introduced by this approximation is almost negligible.

This is because the cosine distribution is very low frequency. As such, the precise shape of its integration domain does not really matter so long as the location and area of the integration domain are preserved.

Sphere/Disk Lights

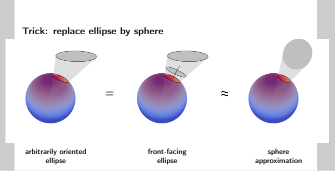
Trick: replace ellipse by sphere



arbitrarily oriented ellipse

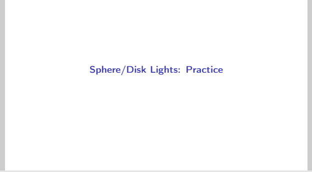
front-facing ellipse

sphere approximation



An intermediate step for computing the approximate sphere is to compute the front-facing ellipse that covers the same solid-angle domain.

Sphere/Disk Lights: Practice

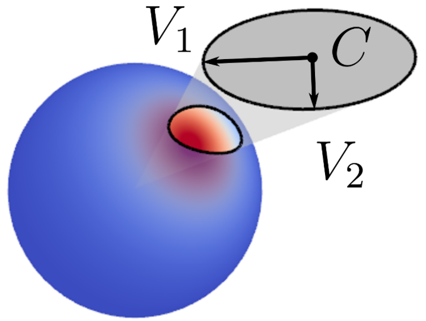

Sphere/Disk Lights: Practice

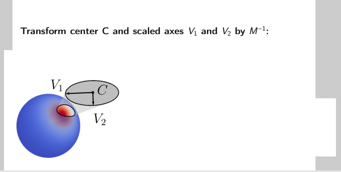
We will now cover disk lighting with LTCs in more detail.

First we'll review the complete process step by step, then we will cover a few practical issues that we hit and the solutions we found.

Sphere/Disk Lights: Practice

Transform center C and scaled axes V_1 and V_2 by M^{-1}:



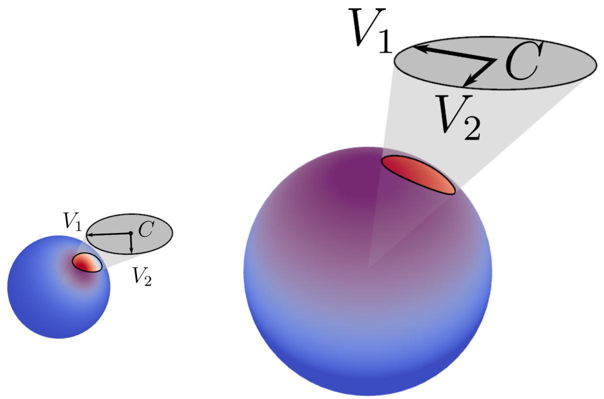


As we mentioned earlier, the first step in the process is to transform the disk by the inverse linear transform M^{-1}, after which we have an ellipse in the cosine configuration.

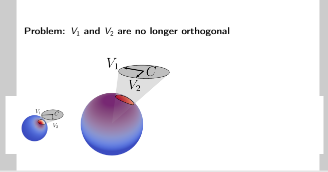
What we actually do here is to transform the two scaled axes of the ellipse, V_1 and V_2, and its center, C.

Sphere/Disk Lights: Practice

Problem: V_1 and V_2 are no longer orthogonal



Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

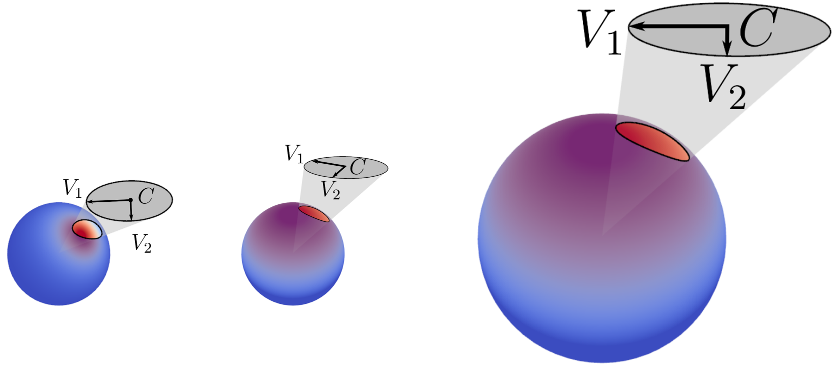


We're now in the cosine configuration – great!

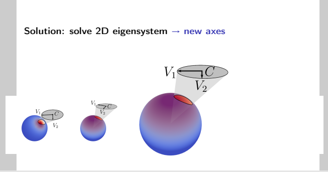
There's just one problem: the transformed vectors are no longer orthogonal.

Sphere/Disk Lights: Practice

Solution: solve 2D eigensystem → new axes



Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

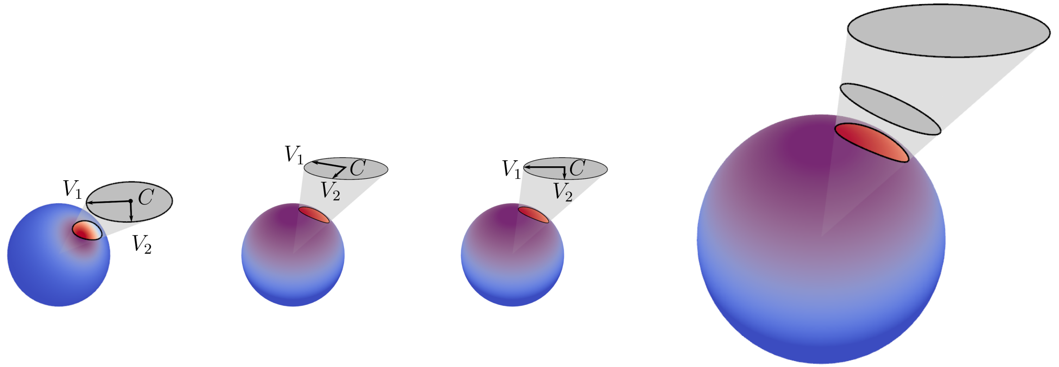


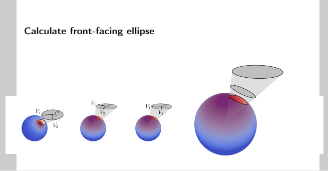
The solution is to solve a 2D eigensystem from the vectors.

From the resulting eigenvalues and eigenvectors, we can determine the axes of the new ellipse.

Sphere/Disk Lights: Practice

Calculate front-facing ellipse



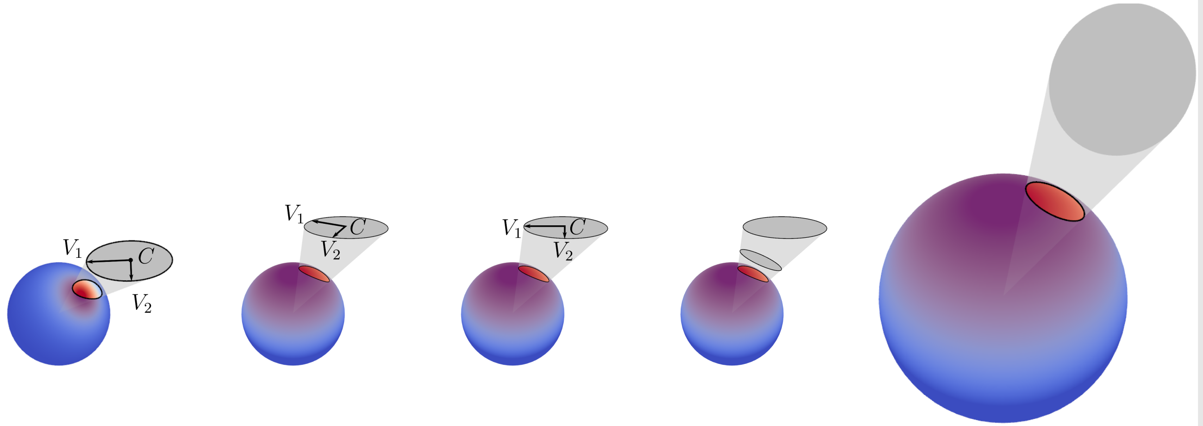


Next we find a front-facing ellipse that has the same solid angle as the previous ellipse.

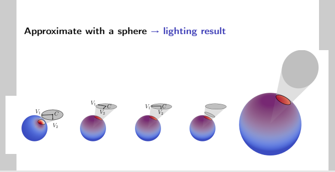
As we mentioned a little earlier, this is an intermediate step.

Sphere/Disk Lights: Practice

Approximate with a sphere → lighting result



Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

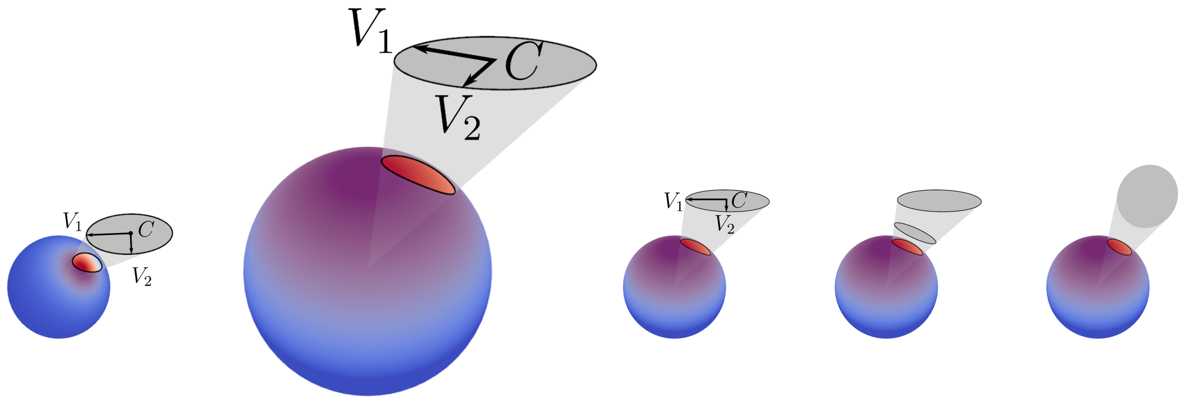

Approximate with a sphere → lighting result

The front-facing ellipse allows us to determine a sphere that (again) has the same solid angle, as well as the same average direction.

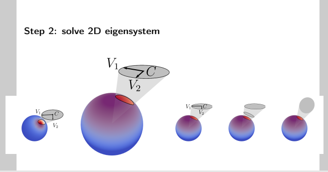
With this, we can calculate the LTC-ellipse light integral. (Or an approximation thereof.)

Sphere/Disk Lights: Practice

Step 2: solve 2D eigensystem



Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



Okay, so that was the complete process we need to go through to calculate the illumination from an elliptical light (or an ellipsoid, circular disk or sphere).

Now let's look at step 2 – solving a 2D eigensystem – in more detail.

Sphere/Disk Lights: Practice

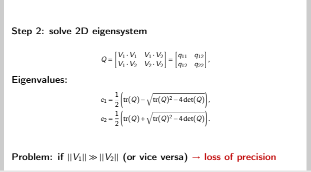
Step 2: solve 2D eigensystem

$$Q = \begin{bmatrix} V_1 \cdot V_1 & V_1 \cdot V_2 \\ V_1 \cdot V_2 & V_2 \cdot V_2 \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix},$$

Eigenvalues:

$$e_1 = \frac{1}{2}\left(\mathrm{tr}(Q) - \sqrt{\mathrm{tr}(Q)^2 - 4\det(Q)}\right),$$
$$e_2 = \frac{1}{2}\left(\mathrm{tr}(Q) + \sqrt{\mathrm{tr}(Q)^2 - 4\det(Q)}\right).$$

Problem: if $||V_1|| \gg ||V_2||$ (or vice versa) $\rightarrow$ loss of precision


Step 2: solve 2D eigensystem

$$Q = \begin{bmatrix} V_1 \cdot V_1 & V_1 \cdot V_2 \\ V_1 \cdot V_2 & V_2 \cdot V_2 \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix},$$

Eigenvalues:

$$e_1 = \frac{1}{2}\left(\mathrm{tr}(Q) - \sqrt{\mathrm{tr}(Q)^2 - 4\det(Q)}\right),$$
$$e_2 = \frac{1}{2}\left(\mathrm{tr}(Q) + \sqrt{\mathrm{tr}(Q)^2 - 4\det(Q)}\right).$$

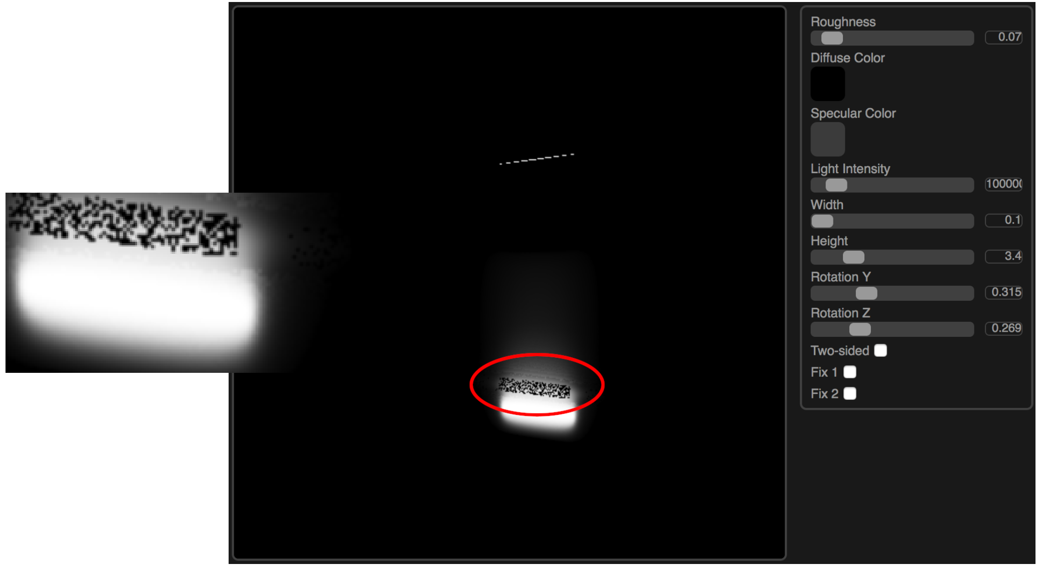
Problem: if $||V_1|| \gg ||V_2||$ (or vice versa) $\rightarrow$ loss of precision

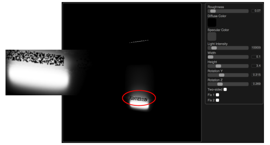
First we form a matrix of dot products from V_1 and V_2. After that we can calculate the eigenvalues and from there the eigenvectors (which we'll cover shortly). From these we can determine the new axes.

However, problems arise when the length V_1 is much greater than V_2, or vice versa. These lengths are squared, leading to many orders of magnitude difference between the leading diagonal elements of the matrix (q_{11} and q_{22}).

Chaos ensues, since the large value swamps the subsequent calculations. The end result is that the smaller eigenvalue can end up being zero instead of its true value.

Sphere/Disk Lights: Practice


Roughness 0.07
Diffuse Color
Specular Color
Light Intensity 10000(
Width 0.1
Height 3.4
Rotation Y 0.315
Rotation Z 0.269
Two-sided
Fix 1
Fix 2



Here's an example of artifacts that can occur in this situation.

In this case the light is almost perpendicular to the viewer.

Note: we've boosted the exposure to make the problems more obvious.

Sphere/Disk Lights: Practice

Solution: we can work with $\sqrt{Q}$ instead

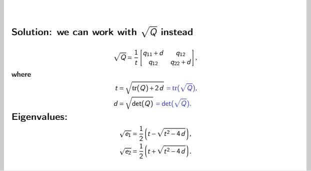
$$\sqrt{Q} = \frac{1}{t}\begin{bmatrix} q_{11}+d & q_{12} \\ q_{12} & q_{22}+d \end{bmatrix},$$

where

$$t = \sqrt{\operatorname{tr}(Q) + 2d} = \operatorname{tr}(\sqrt{Q}),$$
$$d = \sqrt{\det(Q)} = \det(\sqrt{Q}).$$

Eigenvalues:

$$\sqrt{e_1} = \frac{1}{2}\left(t - \sqrt{t^2 - 4d}\right),$$
$$\sqrt{e_2} = \frac{1}{2}\left(t + \sqrt{t^2 - 4d}\right).$$


Solution: we can work with $\sqrt{Q}$ instead

$$\sqrt{Q} = \frac{1}{t}\begin{bmatrix} q_{11}+d & q_{12} \\ q_{12} & q_{22}+d \end{bmatrix},$$

where

$$t = \sqrt{\operatorname{tr}(Q) + 2d} = \operatorname{tr}(\sqrt{Q}),$$
$$d = \sqrt{\det(Q)} = \det(\sqrt{Q}).$$

Eigenvalues:

$$\sqrt{e_1} = \frac{1}{2}\left(t - \sqrt{t^2 - 4d}\right),$$
$$\sqrt{e_2} = \frac{1}{2}\left(t + \sqrt{t^2 - 4d}\right).$$

Fortunately there's a neat trick we can use here: instead of extracting the eigenvalues from Q directly, we can use $\sqrt{Q}$ instead. Because Q is positive definite, there is one unique solution.

As you might expect, working with the square-root matrix brings the leading diagonal values much closer together, resulting in far more stable computations.

Afterwards, we simply need to square the eigenvalues of $\sqrt{Q}$ to obtain the eigenvalues of Q.

Fun fact: the intermediate terms here, t and d, are the trace and determinant of $\sqrt{Q}$. So you can actually express $\sqrt{Q}$ in terms of itself.

Sphere/Disk Lights: Practice

Faster and more stable:

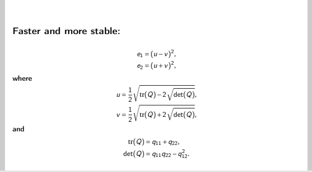
$$e_1 = (u - v)^2,$$
$$e_2 = (u + v)^2,$$

where

$$u = \frac{1}{2}\sqrt{\mathrm{tr}(Q) - 2\sqrt{\det(Q)}},$$
$$v = \frac{1}{2}\sqrt{\mathrm{tr}(Q) + 2\sqrt{\det(Q)}},$$

and

$$\mathrm{tr}(Q) = q_{11} + q_{22},$$
$$\det(Q) = q_{11}q_{22} - q_{12}^2.$$


Faster and more stable:

$$e_1 = (u - v)^2, \quad e_2 = (u + v)^2,$$

where

$$u = \frac{1}{2}\sqrt{\mathrm{tr}(Q) - 2\sqrt{\det(Q)}}, \quad v = \frac{1}{2}\sqrt{\mathrm{tr}(Q) + 2\sqrt{\det(Q)}},$$

and

$$\mathrm{tr}(Q) = q_{11} + q_{22}, \quad \det(Q) = q_{11}q_{22} - q_{12}^2.$$

In practice we use the following for computing e_1 and e_2, expressed in terms of the trace and determinant of the original matrix Q. This saves some operations and further improves precision.

Sphere/Disk Lights: Practice

Eigenvectors:

$$E_1 = q_{12} V_1 + (e_1 - q_{11}) V_2,$$
$$E_2 = q_{12} V_1 + (e_2 - q_{11}) V_2.$$

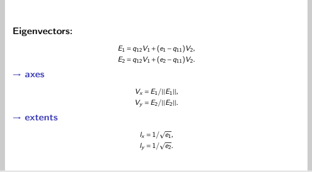
→ axes

$$V_x = E_1 / ||E_1||,$$
$$V_y = E_2 / ||E_2||.$$

→ extents

$$l_x = 1/\sqrt{e_1},$$
$$l_y = 1/\sqrt{e_2}.$$

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



Once we have the eigenvalues, we can then determine the eigenvectors.

From there it's a simple matter of extracting the axes and extents from the eigensystem.

Sphere/Disk Lights: Practice

More stable:

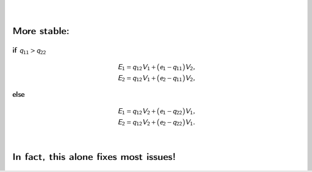
if $q_{11} > q_{22}$

$$E_1 = q_{12}V_1 + (e_1 - q_{11})V_2,$$
$$E_2 = q_{12}V_1 + (e_2 - q_{11})V_2,$$

else

$$E_1 = q_{12}V_2 + (e_1 - q_{22})V_1,$$
$$E_2 = q_{12}V_2 + (e_2 - q_{22})V_1.$$

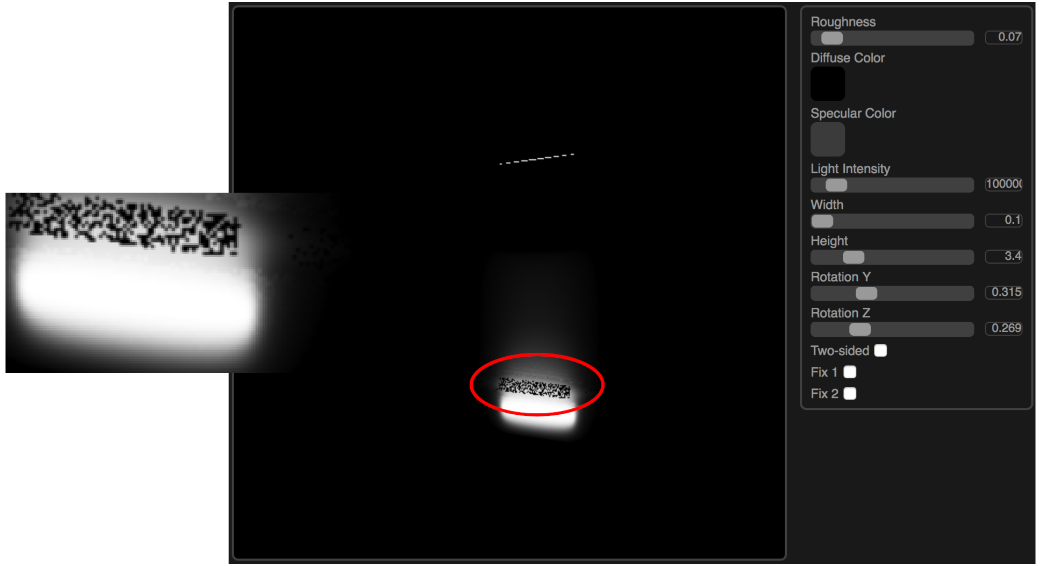
In fact, this alone fixes most issues!



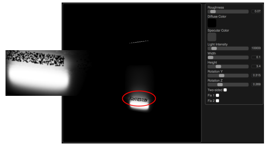
In practice, care is also needed when it comes to computing the eigenvectors. Depending on the relative magnitude of q_{11} and q_{22} we use one set of equations or another.

This alone can fix a lot of the stability issues we've observed with this stage of the process, but not the particular failure case we showed earlier.

Sphere/Disk Lights: Practice

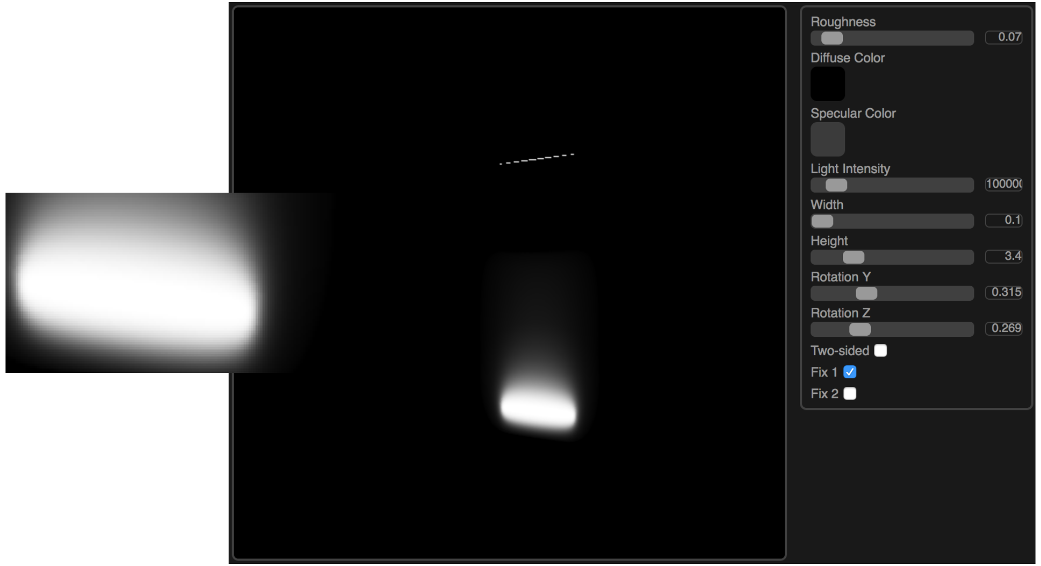


Roughness 0.07
Diffuse Color
Specular Color
Light Intensity 10000(
Width 0.1
Height 3.4
Rotation Y 0.315
Rotation Z 0.269
Two-sided
Fix 1
Fix 2



Speaking of which, here are the artifacts again...

Sphere/Disk Lights: Practice



Roughness 0.07

Diffuse Color ■

Specular Color ■

Light Intensity 10000

Width 0.1

Height 3.4

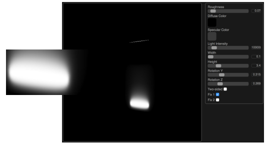
Rotation Y 0.315

Rotation Z 0.269

Two-sided ☐

Fix 1 ☑

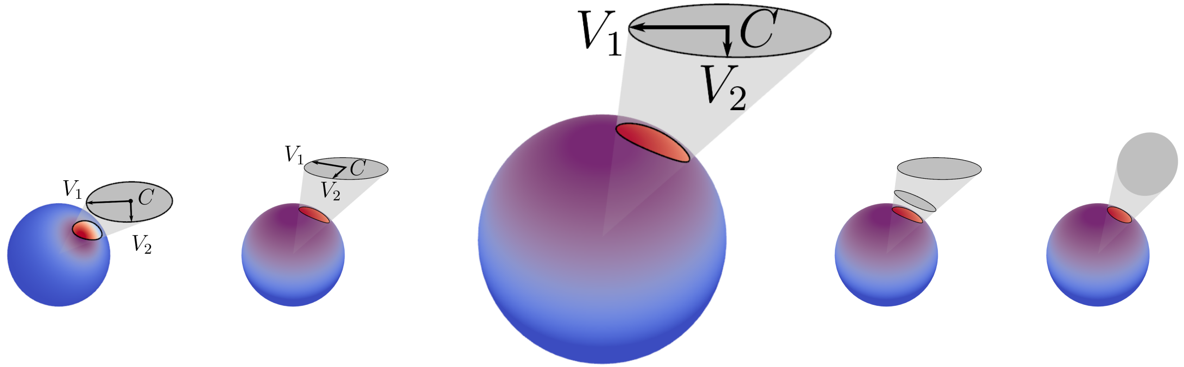
Fix 2 ☐



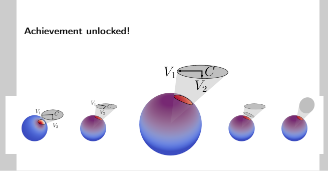
...and here is the result from using $\sqrt{Q}$. The failure cases are gone and we now have a smooth result, as expected.

Sphere/Disk Lights: Practice

Achievement unlocked!



Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



So, “achievement unlocked”: we now have a valid ellipse in the cosine configuration.

The next part of the process is to find a front-facing ellipse with the same solid angle. Both the current ellipse and the front-facing ellipse are conic sections of a cone (with the apex at the shading point), shown here in light grey.

Sphere/Disk Lights: Practice

Cone = spherical quadric

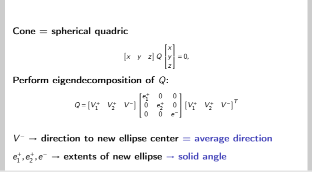
$$\begin{bmatrix} x & y & z \end{bmatrix} Q \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0,$$

Perform eigendecomposition of Q:

$$Q = \begin{bmatrix} V_1^+ & V_2^+ & V^- \end{bmatrix} \begin{bmatrix} e_1^+ & 0 & 0 \\ 0 & e_2^+ & 0 \\ 0 & 0 & e^- \end{bmatrix} \begin{bmatrix} V_1^+ & V_2^+ & V^- \end{bmatrix}^T$$

$V^- \rightarrow$ **direction to new ellipse center = average direction**

$e_1^+, e_2^+, e^- \rightarrow$ **extents of new ellipse → solid angle**



This cone is described by a spherical quadric, which again we'll call Q, just to confuse you. :-) We won't go into the details of calculating Q here since it's not particularly important for this presentation. Instead, please see the technical report referred to earlier and at the end of the slides.

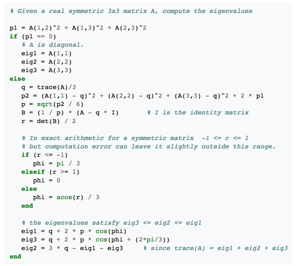
The important point is that if we again perform an eigendecomposition – this time on a 3×3 matrix – we can find the axes and extents of the front facing ellipse. However, instead of the axes, what we are interested in is the direction through the center of the front-facing ellipse, i.e. the average direction. This is given by a third eigenvector V^-.

From the extents we can calculate the solid angle of the ellipse. We then replace the front-facing ellipse with a sphere that has the same average direction and solid angle, from which we can compute the lighting integral.

Sphere/Disk Lights: Practice

Problem: eigendecomposition in a shader is tricky!

<u>Don't</u> copy code from the internet:



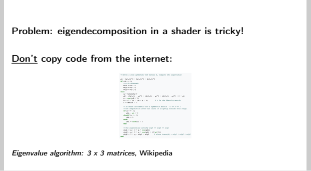
```
% Given a real symmetric 3x3 matrix A, compute the eigenvalues

p1 = A(1,2)^2 + A(1,3)^2 + A(2,3)^2
if (p1 == 0)
   % A is diagonal.
   eig1 = A(1,1)
   eig2 = A(2,2)
   eig3 = A(3,3)
else
   q = trace(A)/3
   p2 = (A(1,1) - q)^2 + (A(2,2) - q)^2 + (A(3,3) - q)^2 + 2 * p1
   p = sqrt(p2 / 6)
   B = (1 / p) * (A - q * I)       % I is the identity matrix
   r = det(B) / 2

   % In exact arithmetic for a symmetric matrix  -1 <= r <= 1
   % but computation error can leave it slightly outside this range.
   if (r <= -1)
      phi = pi / 3
   elseif (r >= 1)
      phi = 0
   else
      phi = acos(r) / 3
   end

   % the eigenvalues satisfy eig3 <= eig2 <= eig1
   eig1 = q + 2 * p * cos(phi)
   eig3 = q + 2 * p * cos(phi + (2*pi/3))
   eig2 = 3 * q - eig1 - eig3     % since trace(A) = eig1 + eig2 + eig3
end
```

Eigenvalue algorithm: 3 x 3 matrices, Wikipedia



Problem: eigendecomposition in a shader is tricky!

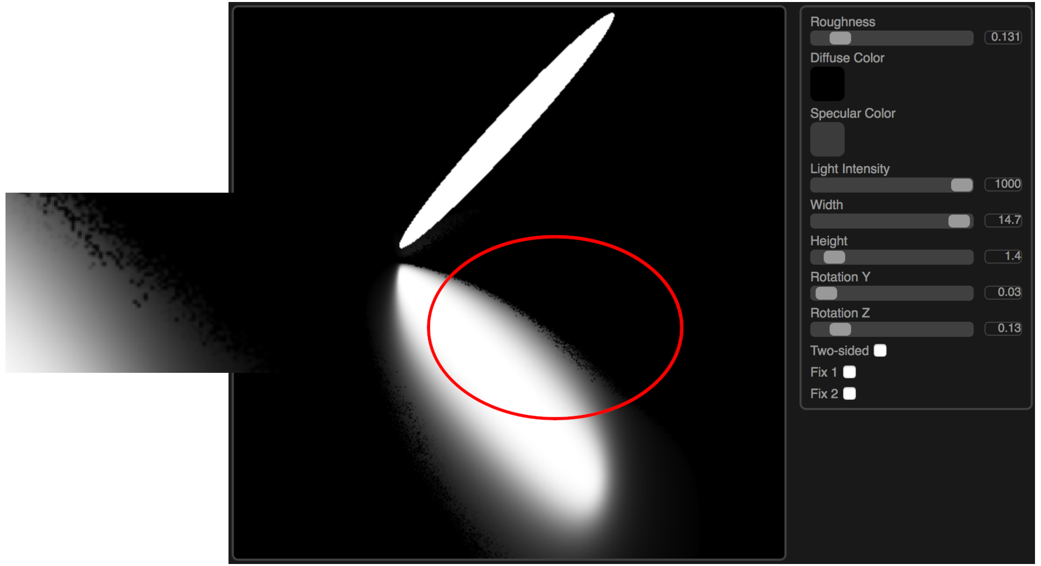
Don't copy code from the internet:

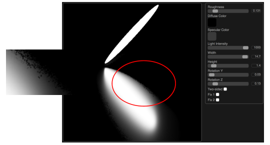
Eigenvalue algorithm: 3 x 3 matrices, Wikipedia

Unfortunately, 3D eigendecomposition is a lot less straightforward than the 2D case we dealt with earlier.

The snippet shown here – taken from Wikipedia – looked appealing as it's fairly compact compared to some other implementations we looked at. Sadly it wasn't robust in this context.

Sphere/Disk Lights: Practice


Roughness 0.131
Diffuse Color
Specular Color
Light Intensity 1000
Width 14.7
Height 1.4
Rotation Y 0.03
Rotation Z 0.13
Two-sided
Fix 1
Fix 2

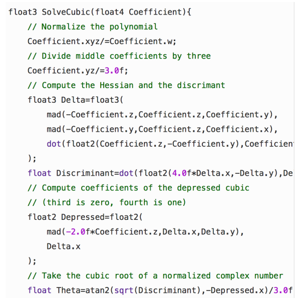


Here is an example of the kind of problems you can run into.

Sphere/Disk Lights: Practice

Solution: use a robust cubic solver

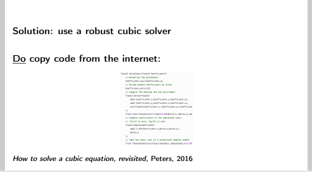
<u>Do</u> copy code from the internet:



```
float3 SolveCubic(float4 Coefficient){
    // Normalize the polynomial
    Coefficient.xyz/=Coefficient.w;
    // Divide middle coefficients by three
    Coefficient.yz/=3.0f;
    // Compute the Hessian and the discrimant
    float3 Delta=float3(
        mad(-Coefficient.z,Coefficient.z,Coefficient.y),
        mad(-Coefficient.y,Coefficient.z,Coefficient.x),
        dot(float2(Coefficient.z,-Coefficient.y),Coefficien
    );
    float Discriminant=dot(float2(4.0f*Delta.x,-Delta.y),De
    // Compute coefficients of the depressed cubic
    // (third is zero, fourth is one)
    float2 Depressed=float2(
        mad(-2.0f*Coefficient.z,Delta.x,Delta.y),
        Delta.x
    );
    // Take the cubic root of a normalized complex number
    float Theta=atan2(sqrt(Discriminant),-Depressed.x)/3.0f
```

How to solve a cubic equation, revisited, Peters, 2016

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



Solution: use a robust cubic solver

Do copy code from the internet:

How to solve a cubic equation, revisited, Peters, 2016

Our solution came from a blog post by Christoph Peters. He had used a cubic polynomial solver from Jim Blinn in the context of his *Moment Shadow Mapping* mapping work.

Blinn's approach – described in *How to Solve a Cubic Equation: Part 1...5* – was designed to handle tricky cases robustly and (relatively) efficiently on a GPU, where only single-precision floating point is currently feasible.

Sphere/Disk Lights: Practice

Solution: use a robust cubic solver

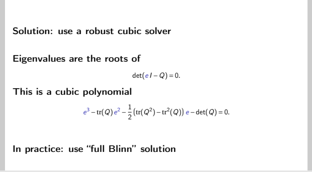
Eigenvalues are the roots of

$$\det(e\,I - Q) = 0.$$

This is a cubic polynomial

$$e^3 - \operatorname{tr}(Q)\,e^2 - \frac{1}{2}\left(\operatorname{tr}(Q^2) - \operatorname{tr}^2(Q)\right)e - \det(Q) = 0.$$

In practice: use “full Blinn” solution


Solution: use a robust cubic solver

Eigenvalues are the roots of

$$\det(e\,I - Q) = 0.$$

This is a cubic polynomial

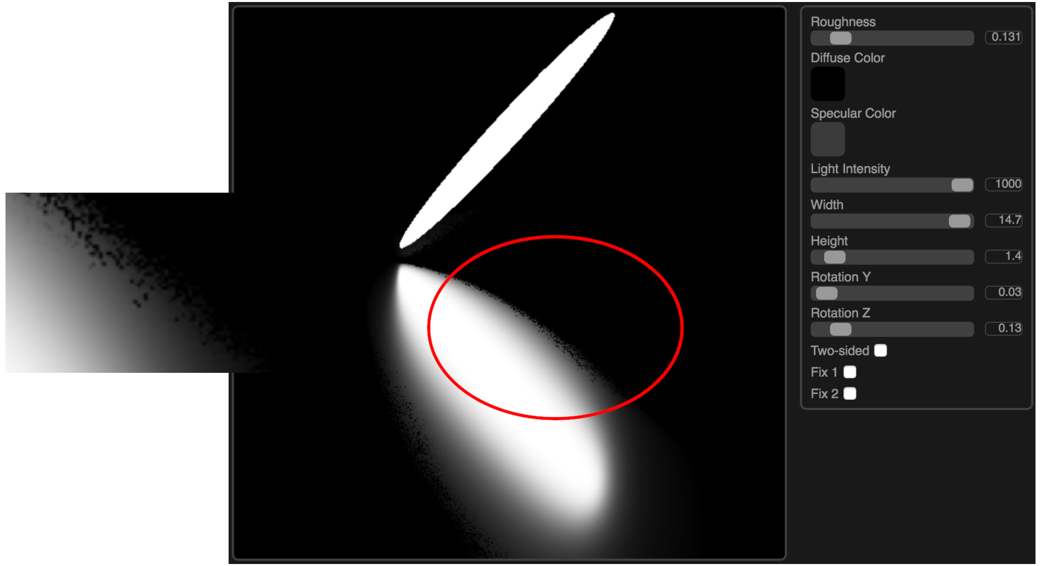
$$e^3 - \operatorname{tr}(Q)\,e^2 - \frac{1}{2}\left(\operatorname{tr}(Q^2) - \operatorname{tr}^2(Q)\right)e - \det(Q) = 0.$$

In practice: use “full Blinn” solution

The reason this can be used is that the eigenvalues are the roots of $\det(e\,I - Q)$, which if you expand it out is a cubic.

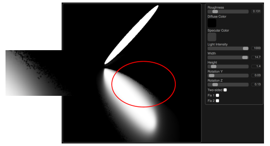
Christoph was able to take some shortcuts for his application, but we found that, for complete robustness, we needed to use the full algorithm.

Sphere/Disk Lights: Practice



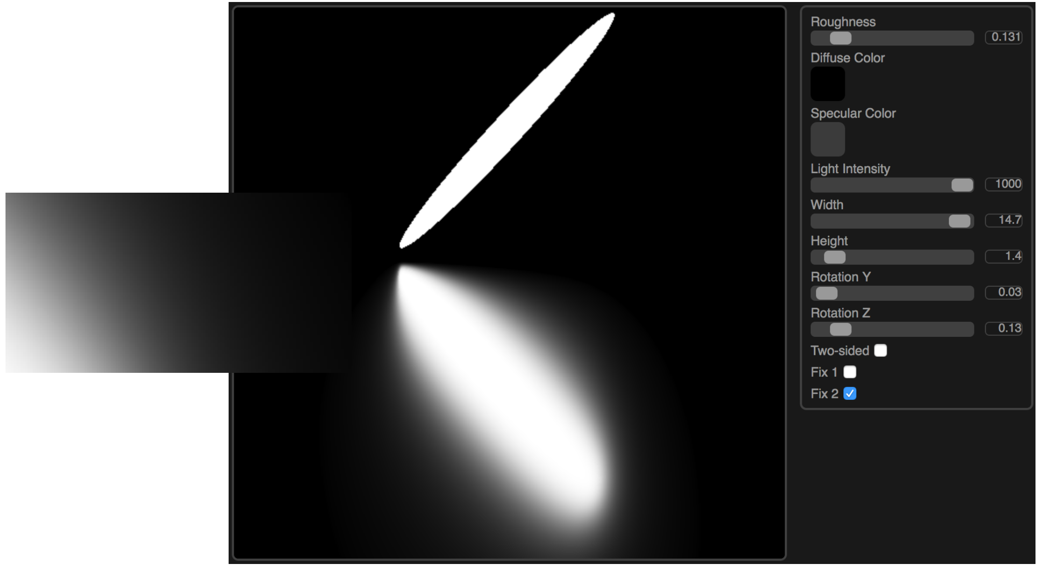
Roughness 0.131
Diffuse Color
Specular Color
Light Intensity 1000
Width 14.7
Height 1.4
Rotation Y 0.03
Rotation Z 0.13
Two-sided
Fix 1
Fix 2

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



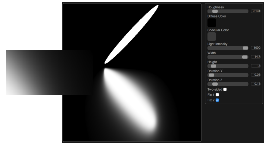
Here’s the image showing the artifacts again.

Sphere/Disk Lights: Practice



Roughness 0.131
Diffuse Color
Specular Color
Light Intensity 1000
Width 14.7
Height 1.4
Rotation Y 0.03
Rotation Z 0.13
Two-sided ☐
Fix 1 ☐
Fix 2 ✓

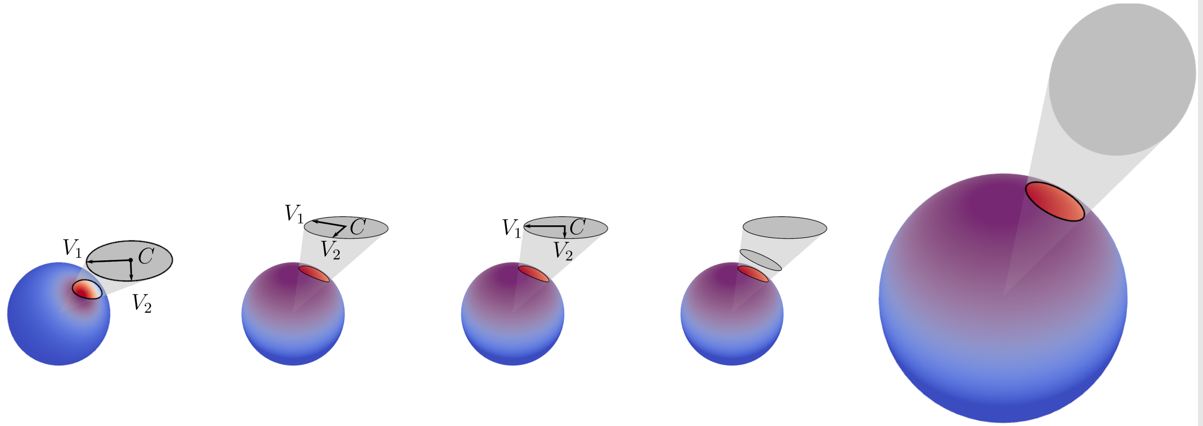
Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

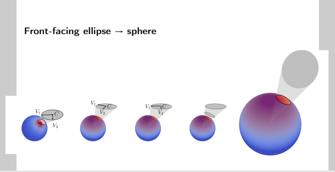


Now here's the same configuration using Blinn's cubic solver. Just as before, the results are now smooth.

Sphere/Disk Lights: Practice

Front-facing ellipse → sphere





The finish line is now in sight. We just need to replace the front-facing ellipse with a sphere and calculate the lighting. There are just a couple of details worth touching on here.

Sphere/Disk Lights: Practice

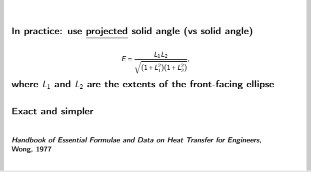
In practice: use <u>projected</u> solid angle (vs solid angle)

$$E = \frac{L_1 L_2}{\sqrt{(1+L_1^2)(1+L_2^2)}},$$

where L_1 and L_2 are the extents of the front-facing ellipse

Exact and simpler

Handbook of Essential Formulae and Data on Heat Transfer for Engineers, Wong, 1977


In practice: use projected solid angle (vs solid angle)

$$E = \frac{L_1 L_2}{\sqrt{(1+L_1^2)(1+L_2^2)}},$$

where L_1 and L_2 are the extents of the front-facing ellipse

Exact and simpler

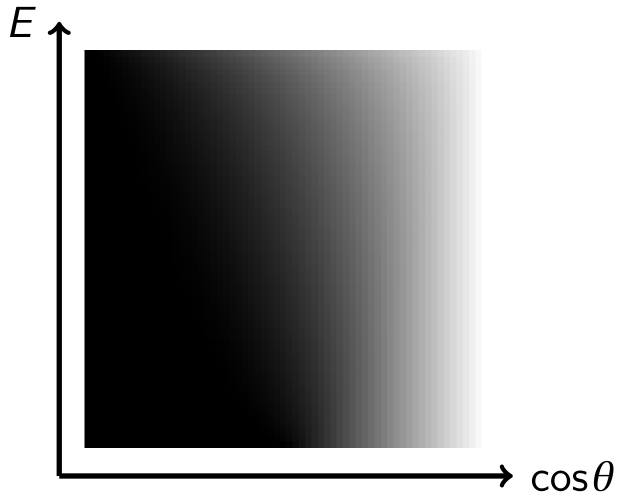
Handbook of Essential Formulae and Data on Heat Transfer for Engineers, Wong, 1977

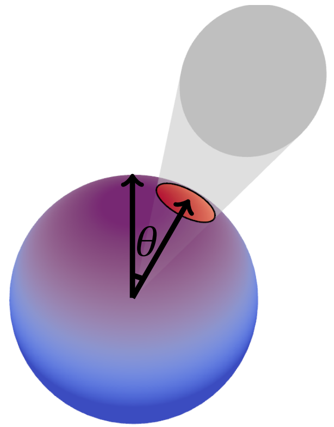
In practice we calculate the projected (cosine-weighted) solid angle E, relative to the average direction. This is because it has a closed form when the ellipse is parallel to a differential element, as is the case here with our front-facing ellipse.

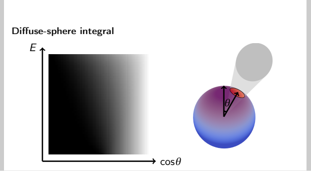
Source: `http://www.thermalradiation.net/sectionb/B-18.html`

Sphere/Disk Lights: Practice

Diffuse-sphere integral







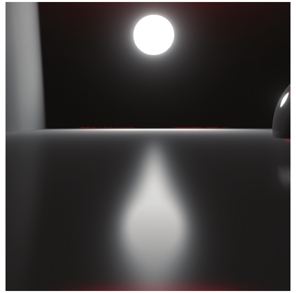
Once we have the projected solid angle, we can obtain the lighting integral for our shading point by looking up into a precomputed table parameterized by E and the angle between the average direction and the surface normal.

Because the function is very smooth, a 64×64 table is enough. This represents a total memory footprint of 8KB at half-float precision.

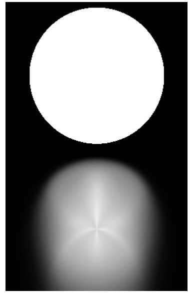
Sphere/Disk Lights: Practice

Comparison to previous work

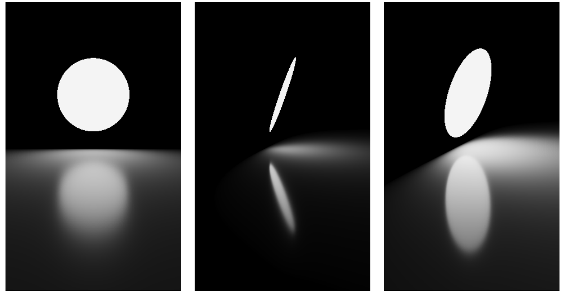
Drobot2014



Lecocq2017

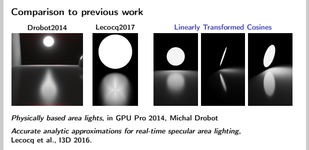


Linearly Transformed Cosines



Physically based area lights, in GPU Pro 2014, Michal Drobot

Accurate analytic approximations for real-time specular area lighting, Lecocq et al., I3D 2016.



Comparison to previous work

Drobot2014 Lecocq2017 Linearly Transformed Cosines

Physically based area lights, in GPU Pro 2014, Michal Drobot

Accurate analytic approximations for real-time specular area lighting, Lecocq et al., I3D 2016.

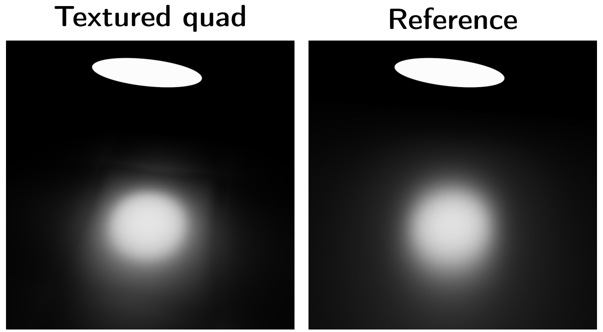
Previously, [Drobot2014] used a point approximation, which produced an incorrect highlight shape (shown here) with longer-tailed distributions such as GGX.

In the more recent work of Lecocq et al., the authors approximated a disk source with a 'spinning' polygon. While this is an ingenious idea, it has some artifacts in certain configurations, such as in the image shown here.

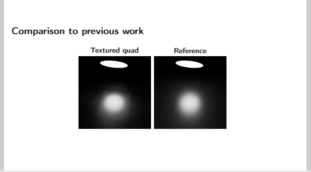
In comparison, we believe that our disk-light method is free of such limitations and produces accurate results in all configurations.

Sphere/Disk Lights: Practice

Comparison to previous work



Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines

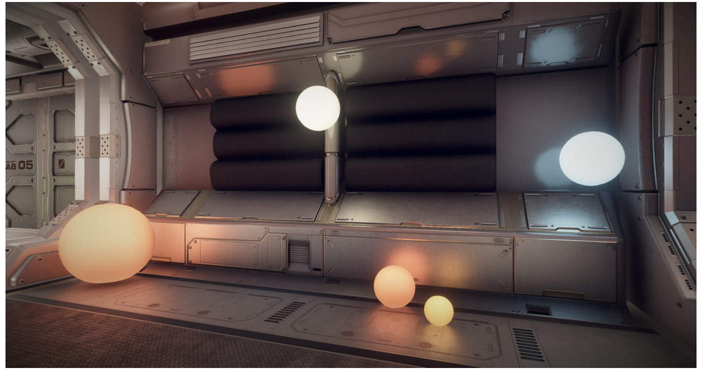


In our paper from last year, we showed a limitation of using a textured quad light to approximate a disk: it fails to produce the elongated highlights of the ground-truth version and there can also be some blockiness.

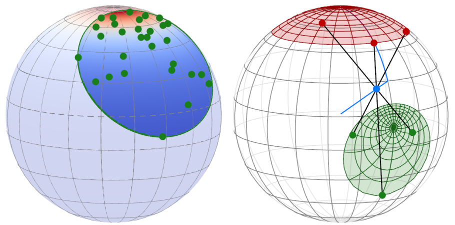
In contrast, our new disk-light method produces the correct highlight shape.

Related Technical Paper

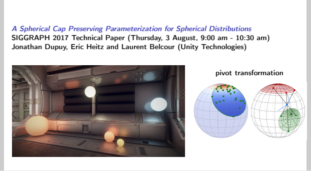
A Spherical Cap Preserving Parameterization for Spherical Distributions
SIGGRAPH 2017 Technical Paper (Thursday, 3 August, 9:00 am - 10:30 am)
Jonathan Dupuy, Eric Heitz and Laurent Belcour (Unity Technologies)



pivot transformation



Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines



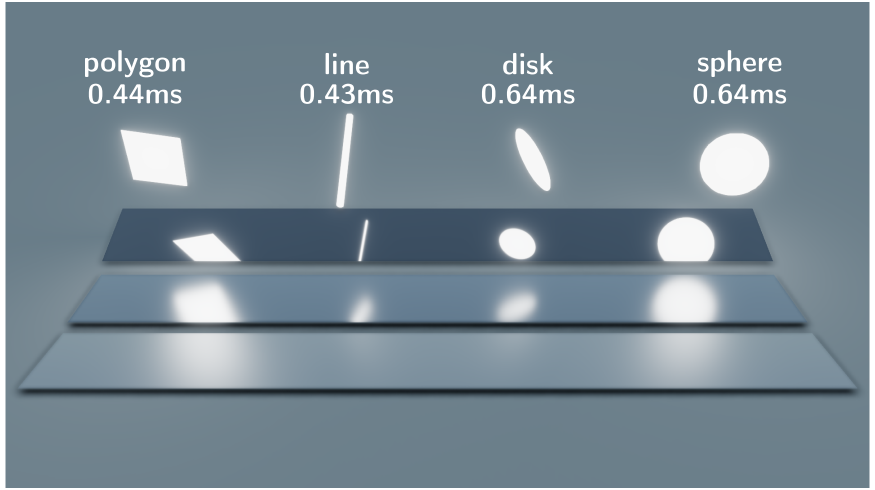
A Spherical Cap Preserving Parameterization for Spherical Distributions
SIGGRAPH 2017 Technical Paper (Thursday, 3 August, 9:00 am - 10:30 am)
Jonathan Dupuy, Eric Heitz and Laurent Belcour (Unity Technologies)
pivot transformation

This year at SIGGRAPH 2017, we also present a technical paper that offers another approach for sphere lights. This paper introduces a new transformation, based on conformal geometry, that preserves spherical caps. We use this transformation to parameterize spherical distributions, which can be integrated analytically over spherical caps.

This paper is in the same spirit as our LTC paper from last year, but with an emphasis on sphere lights instead of polygonal lights: the spherical distribution is crafted especially for offering specific properties for sphere lights. This is thus another complementary approach to this problem.

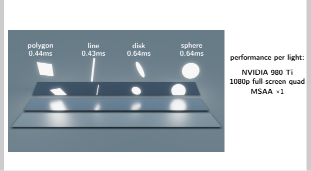
This paper and associated material can be found at
`https://labs.unity.com/article/spherical-cap-preserving-parameterization-spherical-distributions`

Performance



performance per light:

NVIDIA 980 Ti
1080p full-screen quad
MSAA $\times 1$



Here are some performance figures for each of the different light types.

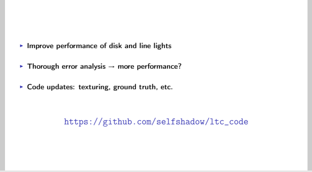
These numbers come from an updated version of our BGFX demo with the Sponza scene and single light. The scene is forward shaded (with a z-prepass), and the specular and diffuse components are rendered as separate passes. As such, the true per-pixel lighting cost is likely to be lower in practice. Still, the numbers are useful for relative comparison.

Note: the polygon timing is for a quad light, and this implementation includes the optimizations we presented in the Advances in Real-Time Rendering course at SIGGRAPH last year. (The unoptimized version is 0.58ms.)

Future Work

- Improve performance of disk and line lights
- Thorough error analysis → more performance?
- Code updates: texturing, ground truth, etc.

https://github.com/selfshadow/ltc_code



- Improve performance of disk and line lights
- Thorough error analysis → more performance?
- Code updates: texturing, ground truth, etc.

https://github.com/selfshadow/ltc_code

We believe that our current line and disk light implementations could be optimized – our focus up until now had been on accuracy and robustness.

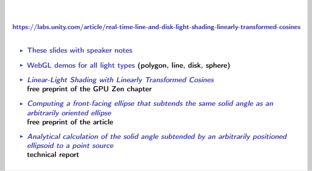
A more thorough analysis of the precision issues we encountered with disk lights might allow us to switch to faster methods in some cases, thereby further improving performance.

Finally, we have released code for all of our methods and hope to make additional updates to this repository in the future.

Associated Material

https://labs.unity.com/article/real-time-line-and-disk-light-shading-linearly-transformed-cosines

- These slides with speaker notes
- WebGL demos for all light types (polygon, line, disk, sphere)
- *Linear-Light Shading with Linearly Transformed Cosines*
  free preprint of the GPU Zen chapter
- *Computing a front-facing ellipse that subtends the same solid angle as an arbitrarily oriented ellipse*
  free preprint of the article
- *Analytical calculation of the solid angle subtended by an arbitrarily positioned ellipsoid to a point source*
  technical report



All associated material can be found at

`https://labs.unity.com/article/real-time-line-and-disk-light-shading-linearly-transformed-cosines.`