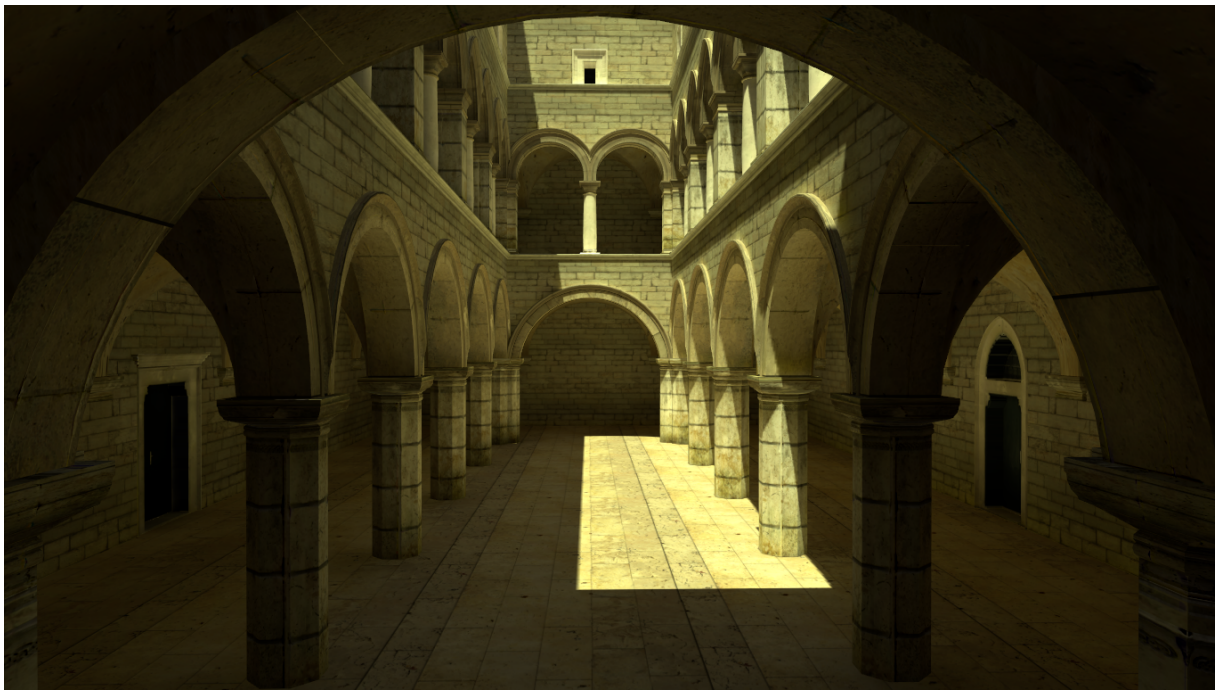
# Irradiance Normal Mapping

Denis Ishmukhametov



2011

# Contents

# Introduction

Cinematic image quality has always been one of the most important challenges of computer graphics. One of the key components, that realistic CG is comprised of, is illumination. In real-time graphics resources are limited, hence we cannot afford dynamic realistic physcially accurate lighting. However, we can precalculate high-quality static lighting - light mapping method. But we cannot use normal mapping with light mapping directly, because in light mapping, lighting is precalculated for one default normal direction whereas in normal mapping normals are defined in a hemispherical domain. Irradiance normal mapping methods overcomes such problems by using spherical or hemispherical basis functions to encode the irradiance signal.

Our task is to compare the existing approaches to irradiance normal mapping visually and analytically by implementing the irradiance normal mapping pipeline with the demo application that demonstrates different lighting methods and allows to switch between them in run-time.

Figure 1: A scene without albedo maps showing the difference between light mapping (left) and irradiance normal mapping (right).

# Chapter 1

# Light Transport in Computer Graphics

## 1.1 The Rendering Equation

**The rendering equation** is a formulation of the problem of producing images in computer graphics that is based only on physics. It is the gold standard by which all realisitic computer graphics lighting must be measured. It was simultaneously introduced into computer graphics by David Immel and James Kajiya in 1986 [8].

Firstly, we have to introduce some essential physical notions that are necessary for understanding the rendering equation.

**Radiance** is the emitted energy per unit time in a given direction from a unit area of an emitting surface. Radiance is a 5D function (3 spacial dimensions and 2 directional dimensions). The radiance of a purely diffuse surface is defined in terms of the surface's irradiance.

**Bidirectional reflectance distribution function (BRDF)** is a function that defines how light is reflected at an opaque surface. This function defines the material properties. For example, such surfaces as matte plastic or glossy metal will have different BRDFs.

We need to calculate the radiance at a surface point $x$ in direction $\omega$.

The total radiance is the sum of emitted and reflected radiance.

$$L\left(x, \vec{\omega}_o\right) = L_e\left(x, \vec{\omega}_o\right) + L_r\left(x, \vec{\omega}_o\right) \tag{1.1}$$

The rendering equation is an integral over a hemisphere of directions where L, the light intensity function we are looking to calculate, appears on both sides of the equation:

$$L\left(x, \vec{\omega}_o\right) = L_e\left(x, \vec{\omega}_o\right) + \int_S f_r\left(x, \vec{\omega}_i \to \vec{\omega}_o\right) L\left(x', \vec{\omega}_i\right) G(x, x') V(x, x') \mathrm{d}\omega_i \tag{1.2}$$

where

$L\left(x, \vec{\omega}_o\right)$ is the radiance reflected from position x in direction $\omega_o$
$L_e\left(x, \vec{\omega}_o\right)$ is the radiance emitted from x by the object itself
$f_r\left(x, \vec{\omega}_i \to \vec{\omega}_o\right)$ is BRDF of the surface at point x, transforming incoming light $\omega_i$ to reflected light $\omega_o$
$L\left(x', \vec{\omega}_i\right)$ is light from x' on another object arriving along $\omega_i$
$G(x, x')$ is the geometric relationship between x and x'
$V(x, x')$ is a visibility test, returns 1 if x can see x', 0 otherwise

The problem with the rendering equation is computational difficulty, it is not a real-time friendly operation. That is why a lot of different attempts were made to reach close results with the simplified equations.

## 1.2 Diffuse Surface Reflection Model

The simplest lighting model is the *diffuse surface reflection* model also known as dot product lighting. Assuming that we have $k$ light sources, for each light source the intensity $\vec{c}_i$ (expressed in 3 values corressponding to RGB colors) is multiplied by the scalar dot product between the unit surface normal $\vec{n}$ and the unit vector towards the light source $\vec{l}$. This value is then multiplied by the surface color $\vec{s}$ giving the final reflected result:

$$I = \vec{s} \sum_{i=1}^{k} \vec{c}_i \left( \vec{n} \bullet \vec{l}_i \right) \tag{1.3}$$

By looking at this formula we can say that it first calculates the total amount of incoming light from all directions and then scales it by the cosine of the angle between $\vec{n}$ and $\vec{l}$ and multiplies the result by the surface reflection function, which for a diffuse surface is just a constant color for all directions.

## 1.3 Irradiance Calculation

The radiance of a purely diffuse surface is defined in terms of the surface's irradiance. **Irradiance** is an integral of the field-radiance function multiplied by the Lambertian cosine term over a hemisphere. The irradiance of a surface point $\vec{x}$ with a surface normal $\vec{n}$ is defined as

$$E\left(\vec{x}, \vec{n}\right) = \int_{\Omega^+} L\left(\vec{x}, \vec{\omega}\right) \left(\vec{n} \bullet \vec{\omega}\right) d\vec{\omega} \tag{1.4}$$

The hemispheres are usually defined in tangent space, i.e. around the interpolated surface normal, so that we do not need to keep track of the orientation of the hemispheres at every point. Irradiance for all possible normal orientations is called **the irradiance distribution function**.

## 1.4 Light Mapping

Lightmapping is the most used technique of static precomputed lighting for static objects. Quake was the first computer game to use light mapping to speed up rendering. Lightmapping is used in most of the all first/third person games. The idea is to precalculate the lighting of the scene and store the irradiance values for the normal directions in vertices or low-resolution textures called light maps. When creating lightmaps, any precalculation method may be used such as ray tracing, photon mapping, radiosity etc. During run-time, irradiance values stored in lightmap textures are multiplied with the RGB color values stored in textures to obtain the final lit texel color. In case of irradiance values stored in vertices they are first interpolated and then mixed with the colors in textures.


Quake (1996)

Quake 2 (1997)

Half-Life (1998)

Figure 1.1: Light mapping examples

## 1.5 Normal Mapping

Normal mapping is a popular technique used for faking the lighting of bumps and dents. It is used to add details without using more polygons. A normal map is usually an RGB image that corresponds to the X, Y, and Z coordinates of a surface normal from a more detailed version of the object. A common use of this technique is to greatly enhance the appearance and details of a low polygon model by generating a normal map from a high polygon model. Normal mapping is used for such surfaces as brick walls, rocks, ground, concrete etc.

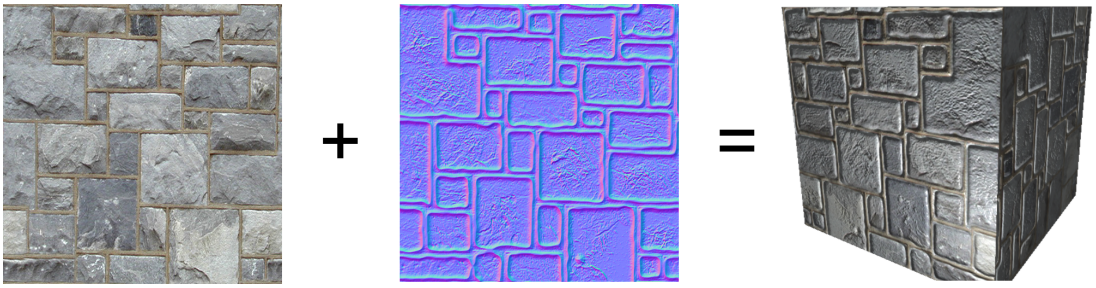


Figure 1.2: Normal mapping example

## 1.6 Irradiance Normal Mapping

By **irradiance normal mapping** we mean the group of methods that combine two popular techniques described above: light mapping and normal mapping. While light mapping stores global, low-frequency illumination at sparsely sampled points in the scene, normal maps provide local, high-frequency shading variation at a high resolution. The problem with combining the two methods is that light maps store lighting information for only one normal direction (the base surface normal) and therefore cannot be evaluated using the normals stored in a normal map. To overcome this problem, the irradiance (i.e., the incoming radiance integrated over the hemisphere) for all possible normal map directions has to be calculated and stored for each illumination sample point on the surface. Because the irradiance signal is low frequency in its directionality, it can be well represented by lower order basis functions.

## 1.7 Orthogonal Basis Functions

Basis functions are small pieces of signal that can be scaled and combined to produce an approximation to an original function, and the process of calculating how much of each

basis function to sum is called projection. To approximate a function using basis functions we must work out a scalar value that represents how much the original function $f(x)$ is like the each basis function $B_i(x)$. We do this by integrating the product $\int f(x)B_i(x)$ over the full domain of $f$.

Using this projection process over all our basis functions returns a vector of approxima-

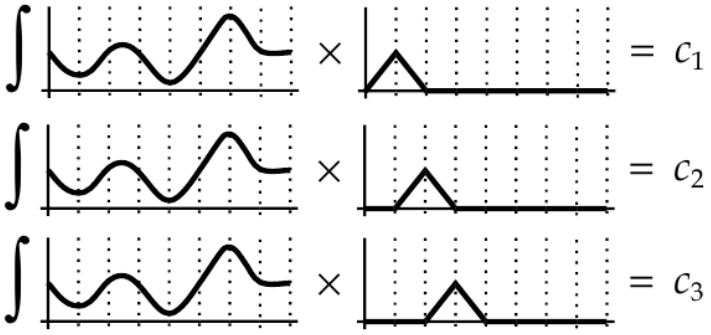



Figure 1.3: Function projection. Image from "Spherical Harmonic Lighting: The Gritty Details"[5]

tion coefficients. If we scale the corresponding basis function by the coefficients and sum the results we obtain our approximated function. In this example we have used a set of

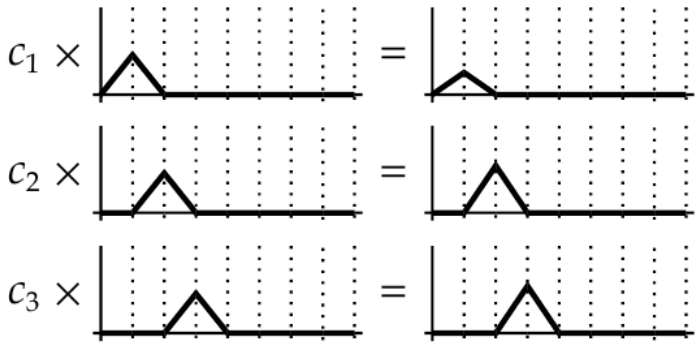



Figure 1.4: Scaling the basis functions by the corresponding coefficients. Image from "Spherical Harmonic Lighting: The Gritty Details"[5]

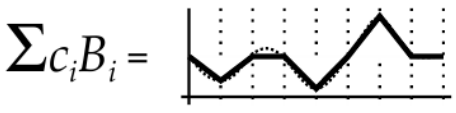



Figure 1.5: Approximated function is a result of the scaled basis functions summation. Image from "Spherical Harmonic Lighting: The Gritty Details"[5]

linear basis functions, giving us a piecewise linear approximation to the input function. There are many basis functions we can use, but some of the most interesting are grouped into a family of functions mathematicians call the *orthogonal polynomials*. Orthogonal polynomials are sets of polynomials that have an intriguing property - when we integrate the product of any two of them, if they are the same we get a constant value and if they are different we get zero which is the definition of orthogonality.

$$\int_{-1}^{1} F_m(x)F_n(x)\mathrm{dx} = \begin{cases} 0, \text{for } n \neq m \\ c, \text{for } n = m \end{cases} \tag{1.5}$$

We can also specify the more rigorous rule that integrating the product of two of these polynomials must return either 0 or 1, and this sub-family of functions are known as the orthonormal basis functions. Intuitively, it's a like the functions do not "overlap" each other's influence while still occupying the same space, the same effect that allows the Fourier transform to break a signal into it's component sine waves.

The problem is to select a good basis function set which could represent the irradiance signal accurately using reasonable amounts of memory.

## 1.8 Spherical and Hemispherical Basis Functions

### 1.8.1 Spherical Harmonics

Spherical harmonics are the angular portion of a set of solutions to Laplace's equation. They are known since 18th century. However, in computer graphics they were introduced realtively not long time ago in a paper at Siggraph 2002 by Sloan, Kautz and Snyder [12] in a technique for precalculated radiance transport (PRT). Before defining the spherical harmonics basis function set we should introduce the Legendre polynomials.

The Legendre polynomials, specifically the **Associated Legendre Polynomials** are an orthonormal basis functions. Traditionally represented by the symbol P, the associated Legendre polynomials have two arguments l and m, are defined over the range [-1,1] and return real numbers (as opposed to the ordinary Legendre Polynomials which return complex values). The two arguments l and m break the family of polynomials into bands of functions where the argument l is the band index and takes any positive integer value starting from 0, and the argument m takes any integer value in the range [0, l]. All polynomials are orthogonal with a constant term. We can diagram this as a triangular grid of functions per band, giving us a total of n(n+1) coefficients for an n band approximation:

$P_0^0(x)$
$P_1^0(x), P_1^1(x)$
$P_2^0(x), P_2^1(x), P_2^2(x)$
...

The process for evaluating Legendre polynomials turns out to be quite involved, which is why they're rarely used for approximating 1D functions. The usual mathematical definition of the series is defined in terms of derivatives of imaginary numbers and requires a series of cancellations of values that alternate in sign and this is not a floating point friendly process. Instead we turn to a set of recurrence relations (i.e. a recursive definition) that generate the current polynomial from earlier results in the series. There are only three rules we need:

1. $(l-m)P_l^m = x(2l-1)P_{l-1}^m - (l+m-1)P_{l-2}^m$

   The main term of the recurrence takes the two previous bands l-1 and l-2 and generates a new higher band l from them.

2. $P_l^m = (-1)^m(2m-1)!!\,(1-x^2)^{m/2}$

   The expression is the best place to start from as it is the only rule that needs no previous values. Note that x!! is the double factorial function which, as (2m-1) is always odd, returns the product of all odd integers less than or equal to x. We can use $P_0^0(x) = 1$ as the initial state for an iterative loop that hoists us up from 0 to m.

3. $P_{m+1}^m = x(2m+1)P_m^m$

This expression allows us to lift a term to a higher band.

The associated Legendre polynomials are at the heart of the Spherical Harmonics (SH), a mathematical system analogous to the Fourier transform but defined across the surface of a sphere. The SH functions in general are defined on complex numbers but we are only interested in approximating real functions over the sphere (i.e. irradiance), so here when we refer to an SH function we will only be talking about the *Real Spherical Harmonic functions*.

Given the standard parameterization of points on the surface of a unit sphere into spherical coordinates:

$$(\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta) \rightarrow (x, y, z) \tag{1.6}$$

the SH function is traditionally represented by the symbol y

$$y_l^m(\theta,\phi) = \begin{cases} \sqrt{2}K_l^m \cos(\mathrm{m}\phi)P_l^m(\cos\theta), m > 0 \\ \sqrt{2}K_l^m \sin(-\mathrm{m}\phi)P_l^{-m}(\cos\theta), m < 0 \\ K_l^0 P_l^0(\cos\theta), m = 0 \end{cases} \tag{1.7}$$

where P is the associated Legendre polynomials and K is just a scaling factor to normalize the functions:

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}} \tag{1.8}$$

In order to generate all the SH functions, the parameters l and m are defined slightly differently from the Legendre polynomials -l is still a positive integer starting from 0, but m takes signed integer values from -l to l.
The first 3 bands of SH functions (without Condon-Shortley phase) are:

$$Y_0^0 = \frac{1}{2}\sqrt{\frac{1}{\pi}} \tag{1.9}$$

$$Y_{-1}^1 = \frac{1}{2}\sqrt{\frac{3}{\pi}}\sin\phi\sin\theta = \frac{1}{2}\sqrt{\frac{3}{\pi}}y \tag{1.10}$$

$$Y_0^1 = \frac{1}{2}\sqrt{\frac{3}{\pi}}\cos\theta = \frac{1}{2}\sqrt{\frac{3}{\pi}}z \tag{1.11}$$

$$Y_1^1 = \frac{1}{2}\sqrt{\frac{3}{\pi}}\cos\phi\sin\theta = \frac{1}{2}\sqrt{\frac{3}{\pi}} \tag{1.12}$$

$$Y_{-2}^2 = \frac{1}{2}\sqrt{\frac{15}{\pi}}\sin 2\phi\sin^2\theta = \frac{1}{2}\sqrt{\frac{15}{\pi}}xy \tag{1.13}$$

$$Y_{-1}^2 = \frac{1}{2}\sqrt{\frac{15}{\pi}}\sin\phi\cos\theta\sin\theta = \frac{1}{2}\sqrt{\frac{15}{\pi}}yz \tag{1.14}$$

$$Y_0^2 = \frac{1}{4}\sqrt{\frac{5}{\pi}}\left(3\cos^2\theta - 1\right) = \frac{1}{4}\sqrt{\frac{5}{\pi}}\left(3z^2 - 1\right) \tag{1.15}$$

$$Y_1^2 = \frac{1}{2}\sqrt{\frac{15}{\pi}}\cos\phi\cos\theta\sin\theta = \frac{1}{2}\sqrt{\frac{15}{\pi}}zx \tag{1.16}$$

$$Y_2^2 = \frac{1}{4}\sqrt{\frac{15}{\pi}}\cos 2\phi\sin^2\theta = \frac{1}{4}\sqrt{\frac{15}{\pi}}\left(x^2 - y^2\right) \tag{1.17}$$

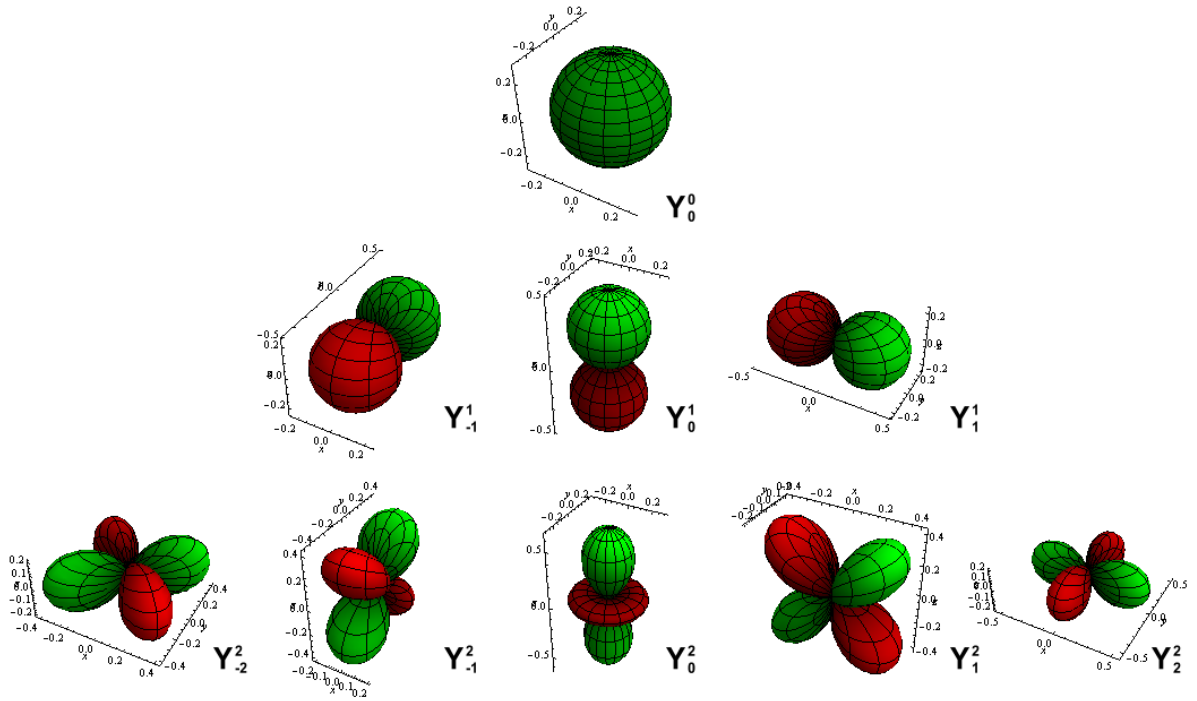



Figure 1.6: Graphical representation of Spherical Harmonics. Marked as green is a positive area, red - negative

### 1.8.2 Half-Life 2 Basis

Half-Life 2 basis [11] was developed by Valve for use in their Source engine and the game Half-Life 2 in particular. It is a function set that is orthonormal over the upper unit hemisphere $\Omega_+$ as well as full sphere $\Omega$. It is defined by 3 basis vectors $\vec{h_i}$:

$$\vec{h_1} = \left(\frac{-1}{\sqrt{6}}, \frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{3}}\right) \tag{1.18}$$

$$\vec{h_2} = \left(\frac{-1}{\sqrt{6}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}}\right) \tag{1.19}$$

$$\vec{h_3} = \left(\sqrt{\frac{2}{3}}, 0, \frac{1}{\sqrt{3}}\right) \tag{1.20}$$

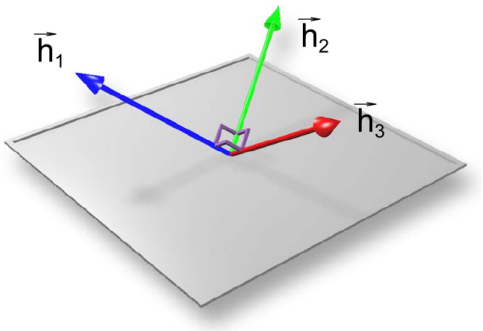



Figure 1.7: Half-Life 2 basis vectors

The explicit evaluation $E_{\mathrm{HL2}}$ is executed through the dot products of the evaluation

direction $\vec{n}$ and the basis directions weighted by their coefficients:

$$E_{\mathrm{HL2}} = \sum_{i=1}^{3} \sqrt{\frac{3}{2\pi}} c_i \vec{n} \cdot \vec{h_i} \tag{1.21}$$

where $c_i$ are the basis coefficients, $\sqrt{\frac{3}{2\pi}}$ is the normalization factor.

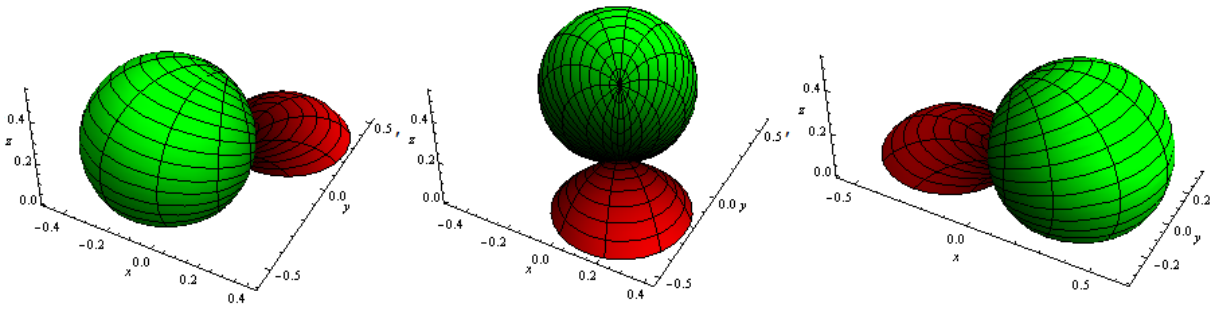


Figure 1.8: Graphical representation of Half-Life 2 basis

Though defined on the hemisphere, the three basis functions are, due to the linearity of the basis, a rotated and hemispherically normed version of linear Spherical Harmonics, which also consist of three cosine lobes.

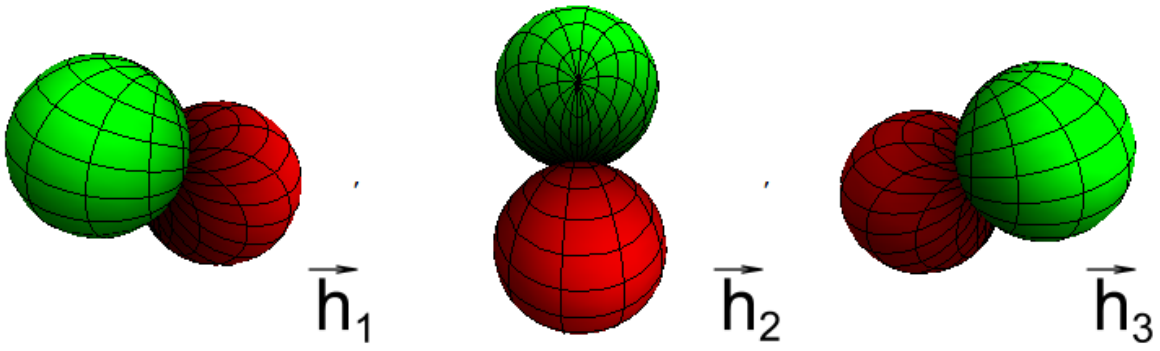



Figure 1.9: Graphical representation of Half-Life 2 basis over a full sphere domain, showing that they are rorated versions of the SH linear band.

### 1.8.3 Hemispherical Harmonics Basis

The hemispherical harmonics (HSH) basis was developed by P. Gautron in 2004 [4]. It is a hemispherically orthonormal basis that is constructed by mapping negative pole at (0,0,-1) to the border of the hemisphere, contracting the Spherical Harmonics functions to hemispherical ones through a shifting operation without destroying their orthogonality. This shifting is generated by replacing the $cos\theta$ term in the SH definition with $2cos\theta - 1$. Due to this shifting, the basis functions become strongly non-polynomial and therefore expensive to evaluate as they contain several square roots and divisions. This is important since for irradiance caching, the basis functions are evaluated directly in the pixel shader. Also due to the shifting, all basis functions are either 0 or constant at the border of the hemisphere, causing severe errors localized at the border. This effect diminishes with higher bands, but is dominant if only two or three bands are used since only one constant color can be represented at the border of the hemisphere. Due to their spherical

origin, their band structure is the same as Spherical Harmonics.

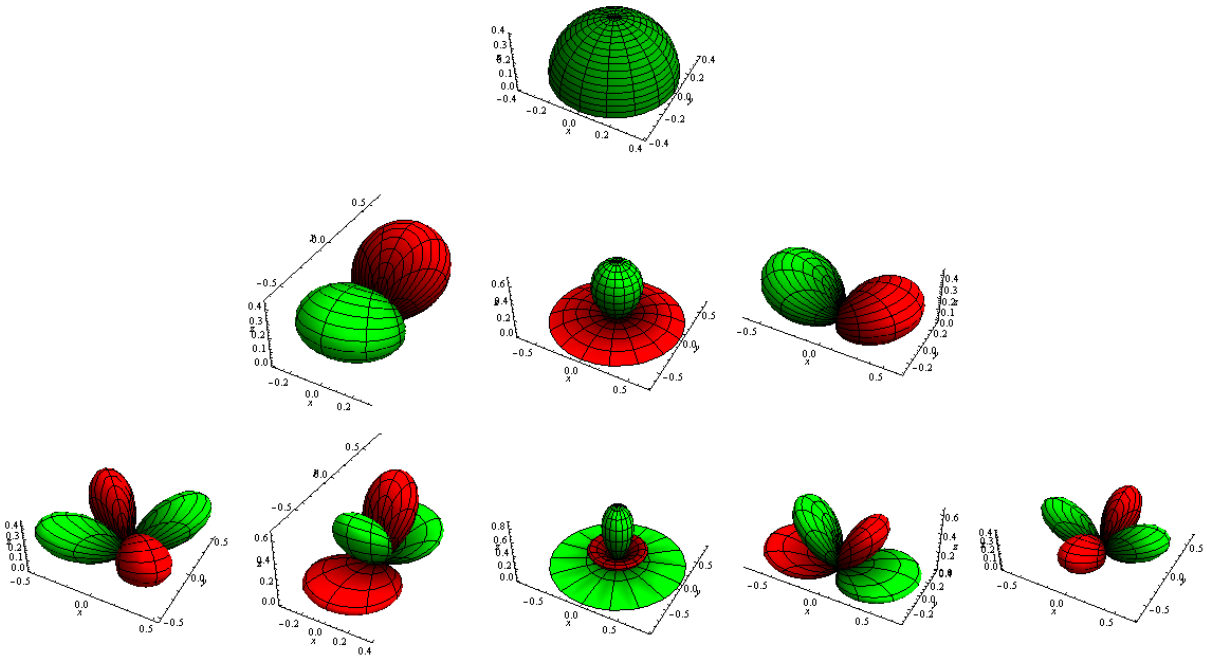


Figure 1.10: Graphical representation of the Hemispherical Harmonics

The first 3 bands of HSH functions are:

$$H_0^0 = \frac{1}{\sqrt{2\pi}} \tag{1.22}$$

$$H_{-1}^1 = \sqrt{\frac{6}{\pi}}\sqrt{(1-\cos\theta)\cos\theta}\sin\phi = -\sqrt{\frac{6}{\pi}}y\sqrt{\frac{z}{1+z}} \tag{1.23}$$

$$H_0^1 = \sqrt{\frac{3}{2\pi}}(2\cos\theta - 1) = \sqrt{\frac{3}{2\pi}}(2z-1) \tag{1.24}$$

$$H_1^1 = \sqrt{\frac{6}{\pi}}\cos\phi\sqrt{(1-\cos\theta)\cos\theta} = \sqrt{\frac{6}{\pi}}x\sqrt{\frac{z}{1+z}} \tag{1.25}$$

$$H_{-2}^2 = \sqrt{\frac{30}{\pi}}\left(\cos^2\theta - \cos\theta\right)\sin 2\phi = 2\sqrt{\frac{30}{\pi}}\frac{xyz}{1+z} \tag{1.26}$$

$$H_{-1}^2 = \sqrt{\frac{30}{\pi}}\sqrt{(1-\cos\theta)\cos\theta}(2\cos\theta - 1)\sin\phi \tag{1.27}$$

$$= \sqrt{\frac{30}{\pi}}y(2z-1)\sqrt{\frac{z}{1+z}} \tag{1.28}$$

$$H_0^2 = \sqrt{\frac{5}{2\pi}}\left(6\cos^2\theta - 6\cos\theta + 1\right) = \sqrt{\frac{5}{2\pi}}\left(6z^2 - 6z + 1\right) \tag{1.29}$$

$$H_1^2 = \sqrt{\frac{30}{\pi}}\sqrt{(1-\cos\theta)\cos\theta}(2\cos\theta - 1)\sin\phi \tag{1.30}$$

$$= \sqrt{\frac{30}{\pi}}x(2z-1)\sqrt{\frac{z}{1+z}} \tag{1.31}$$

$$H_2^2 = \sqrt{\frac{30}{\pi}}\cos 2\phi\left(\cos^2\theta - \cos\theta\right) = \sqrt{\frac{30}{\pi}}z\left(\frac{x^2-y^2}{1+z}\right) \tag{1.32}$$

### 1.8.4 Makhotkin Basis

This basis was developed by Oleg Makhotkin at Novosibirsk State University in 1996 [10]. Makhotkin basis is obtained by shifting Jacobi polynomials. This basis has the same problems as HSH i.e. non-polynomial nature and severe errors at the border of the hemisphere.
Makhotkin basis functions are:

$$M_0^0 = \frac{1}{\sqrt{\pi}}\sqrt{\cos\theta} = \frac{1}{\sqrt{\pi}}z \tag{1.33}$$

$$M_{-1}^1 = \sqrt{\frac{3}{\pi}}\cos\phi\sqrt{\cos\theta}\sqrt{1-(2\cos\theta-1)^2} = 2\sqrt{\frac{3}{\pi}}xz\sqrt{\frac{1}{1+z}} \tag{1.34}$$

$$M_0^1 = \frac{1}{\sqrt{2\pi}}\sqrt{\cos\theta}(3(2\cos\theta-1)-1) = \frac{1}{\sqrt{2\pi}}\sqrt{z}(6z-4) \tag{1.35}$$

$$M_1^1 = \sqrt{\frac{3}{\pi}}\sqrt{\cos\theta}\sqrt{1-(2\cos\theta-1)^2}\sin\phi = 2\sqrt{\frac{3}{\pi}}yz\sqrt{\frac{1}{1+z}} \tag{1.36}$$

$$M_{-2}^2 = \frac{1}{2}\sqrt{\frac{15}{\pi}}\cos 2\phi\sqrt{\cos\theta}\left(1-(2\cos\theta-1)^2\right) \tag{1.37}$$

$$= 2\sqrt{\frac{15}{\pi}}z\sqrt{z}\,\frac{x^2-y^2}{1+z} \tag{1.38}$$

$$M_{-1}^2 = \frac{1}{2}\sqrt{\frac{3}{\pi}}\cos\phi\sqrt{\cos\theta}(4+5(2\cos\theta-2))\sqrt{1-(2\cos\theta-1)^2} = \tag{1.39}$$

$$= \sqrt{\frac{3}{\pi}}xz(10z-6)\sqrt{\frac{1}{1+z}} \tag{1.40}$$

$$M_0^2 = \sqrt{\frac{3}{\pi}}\sqrt{\cos\theta}\left(1+4(2\cos\theta-2)+\frac{5}{2}(2\cos\theta-2)^2\right) \tag{1.41}$$

$$= \sqrt{\frac{3}{\pi}}\sqrt{z}\left(1+16(z-1)+10(z-1)^2\right) \tag{1.42}$$

$$M_1^2 = \frac{1}{2}\sqrt{\frac{3}{\pi}}\sqrt{\cos\theta}(4+5(2\cos\theta-2))\sqrt{1-(2\cos\theta-1)^2}\sin\phi = \tag{1.43}$$

$$= \sqrt{\frac{3}{\pi}}yz(10z-6)\sqrt{\frac{1}{1+z}} \tag{1.44}$$

$$M_2^2 = \frac{1}{2}\sqrt{\frac{15}{\pi}}\sqrt{\cos\theta}\left(1-(2\cos\theta-1)^2\right)\sin 2\phi = 4\sqrt{\frac{15}{\pi}}xy\sqrt{z}\frac{z}{1+z} \tag{1.45}$$

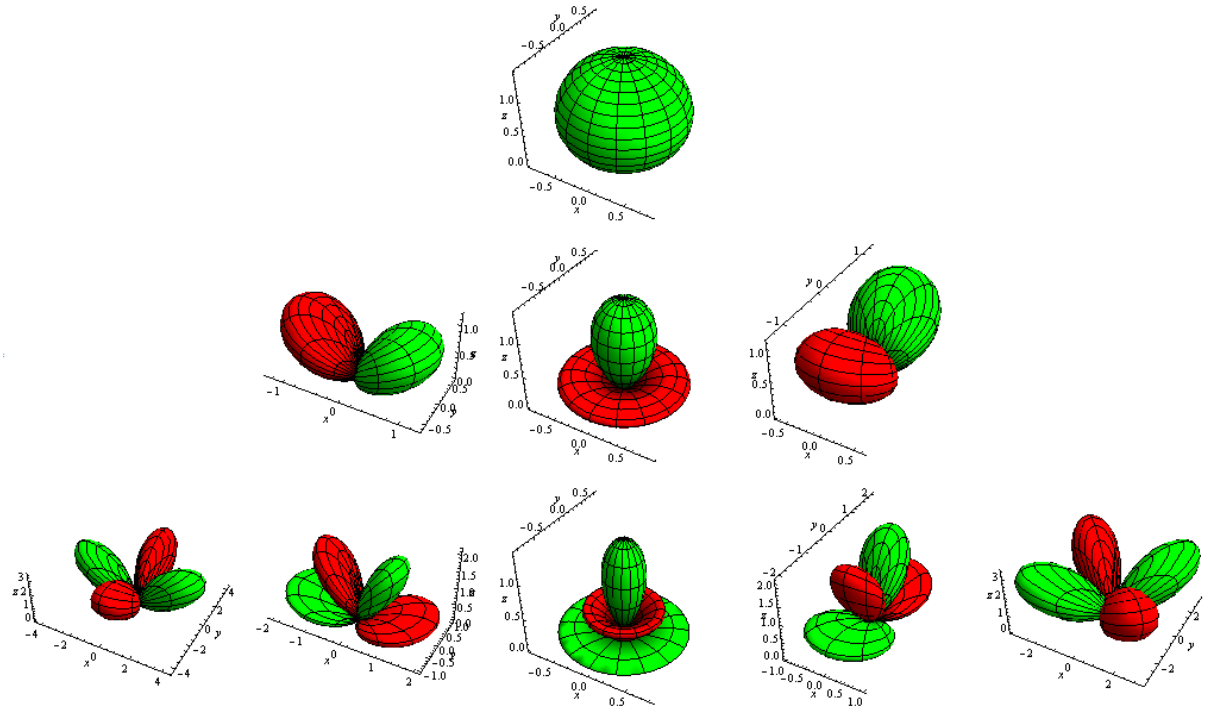


Figure 1.11: Graphical representation of Makhotkin basis

### 1.8.5 Zernike Basis

Zernike basis is an orthonormal polynomial set on the unit circle that have been adapted to the hemisphere by J. Koenderink in 1996 to represent BRDFs [9]. However, the adaptation introduces some square roors and divisions which are more expensive to evaluate in comparison with purely polynomial basis sets. Because of the their two-dimensional origin they feature a different basis structure having only 2 functions in first band and 3 functions in the second band, resulting in 3 coefficients up to the first band and 6 coefficients up to the second band. This makes Zernike basis interesting for an irradiance signal representation with only a few coefficients.
Zernike basis functions are:

$$Z_0^0 = \frac{1}{\sqrt{2\pi}} \tag{1.46}$$

$$Z_{-1}^1 = -\frac{2}{\sqrt{\pi}} \sin\phi \sin\frac{\theta}{2} = -\sqrt{\frac{2}{\pi}} y \frac{1}{\sqrt{1+z}} \tag{1.47}$$

$$Z_1^1 = \frac{2}{\sqrt{\pi}} \cos\phi \sin\frac{\theta}{2} = -\sqrt{\frac{2}{\pi}} x \frac{1}{\sqrt{1+z}} \tag{1.48}$$

$$Z_{-2}^2 = -2\sqrt{\frac{3}{\pi}} \sin 2\phi \sin^2\left(\frac{\theta}{2}\right) = -2\sqrt{\frac{3}{\pi}} \frac{xy}{1+z} \tag{1.49}$$

$$Z_0^2 = \sqrt{\frac{3}{2\pi}} \left(4\sin^2\left(\frac{\theta}{2}\right) - 1\right) = \sqrt{\frac{3}{2\pi}} (1-2z) \tag{1.50}$$

$$Z_2^2 = 2\sqrt{\frac{3}{\pi}} \cos 2\phi \sin^2\left(\frac{\theta}{2}\right) = \sqrt{\frac{3}{\pi}} \left(\frac{x^2-y^2}{1+z}\right) \tag{1.51}$$

Compared to shifted hemispherical bases, the Zernike basis does not suffer from severe errors at the border of the hemisphere, because the basis functions vary there.

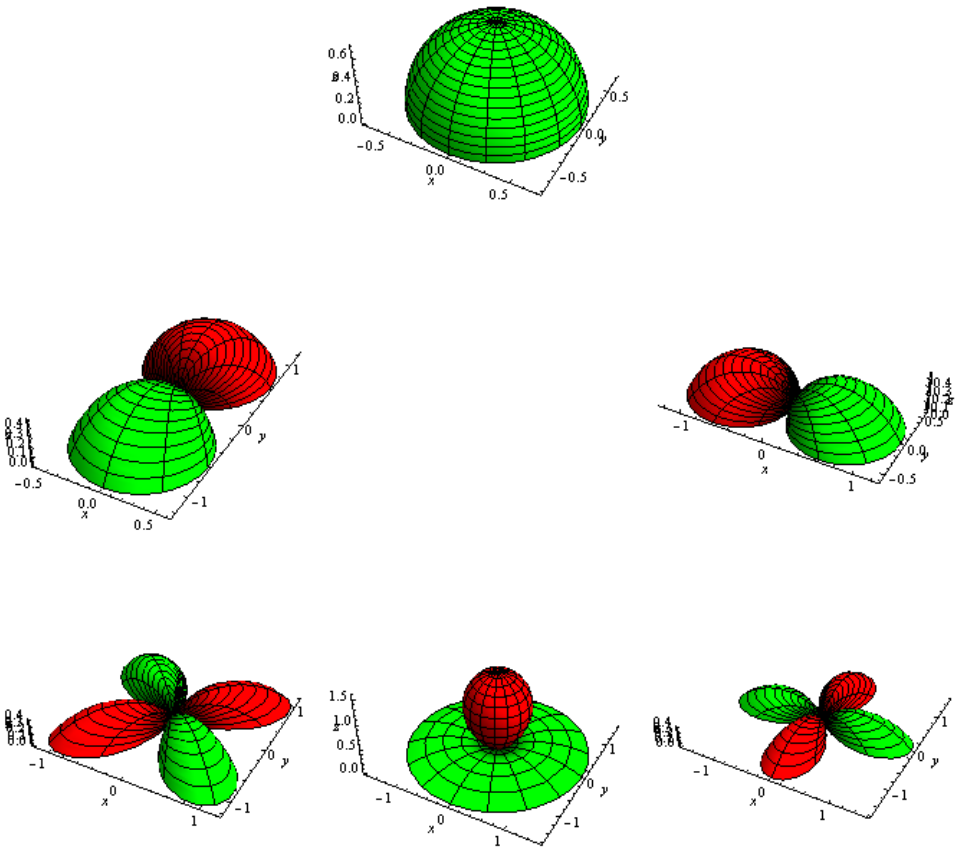


Figure 1.12: Graphical representation of Zernike basis

### 1.8.6 H-basis

The H-basis is the newest basis, it was introduced in 2010 by R.Habel and M.Wimmer [7]. The H-basis is an orthonormal hemispherical basis which consist of only polynomial basis functions up to a quadratic degree and therefore shares many properties with Spherical Harmonics. Some of the basis functions are actually the same basus functions as Spherical Harmonics, re-normalized on the hemisphere, that is why the H-basis can be seen as the counterpart of Spherical Harmonics on the hemisphere up to the quadratic band.

This basis is explicitly constructed to carry hemispherical directional irradiance signals and can provide similar accuracy with only 6 basis functions compared to 9 needed by Spherical Harmonics.
The basis functions are:

$$H^1 = \frac{1}{\sqrt{2\pi}} \tag{1.52}$$

$$H^2 = \sqrt{\frac{3}{2\pi}} \sin\phi \ \sin\theta = \sqrt{\frac{3}{2\pi}} y \tag{1.53}$$

$$H^3 = \sqrt{\frac{3}{2\pi}} (2\cos\theta - 1) = \sqrt{\frac{3}{2\pi}} (2z - 1) \tag{1.54}$$

$$H^4 = \sqrt{\frac{3}{2\pi}} \cos\phi \sin\theta = \sqrt{\frac{3}{2\pi}} x \tag{1.55}$$

$$H^5 = \frac{1}{2}\sqrt{\frac{15}{2\pi}} \sin 2\phi \sin^2\theta = \sqrt{\frac{15}{2\pi}} xy \tag{1.56}$$

$$H^6 = \frac{1}{2}\sqrt{\frac{15}{2\pi}} \cos 2\phi \sin^2\theta = \frac{1}{2}\sqrt{\frac{15}{2\pi}} \left(x^2 - y^2\right) \tag{1.57}$$

The band structure of the H-basis (similar to Spherical Harmonics) provides a natural

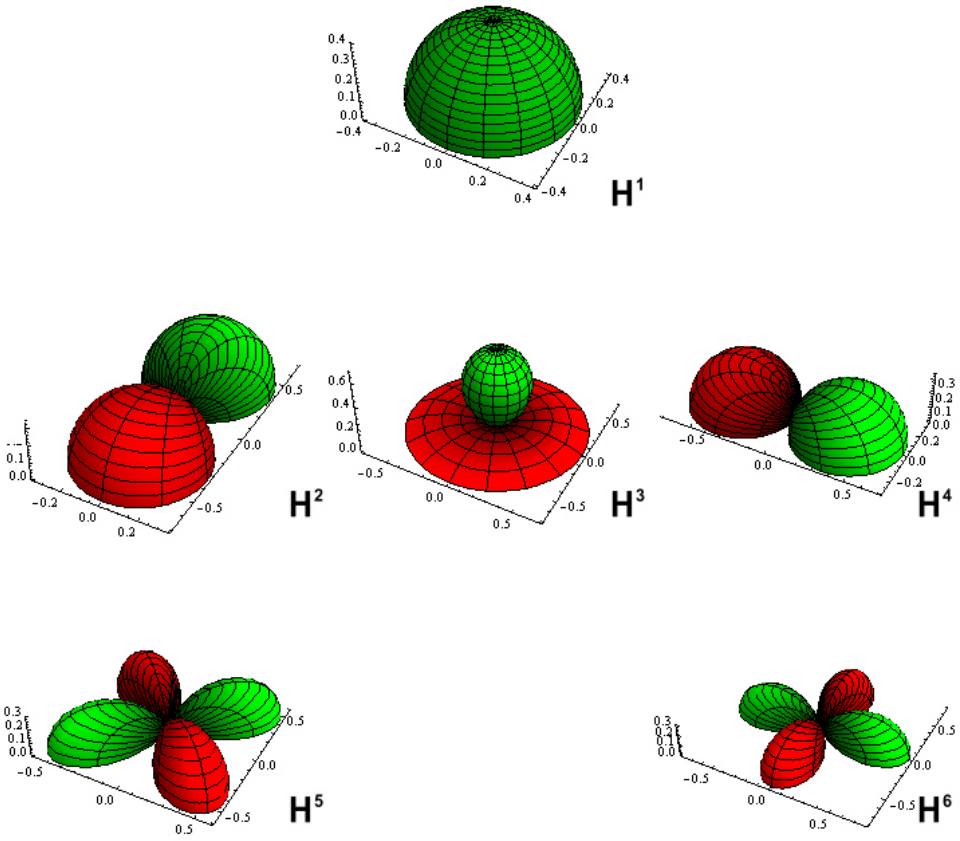



Figure 1.13: Graphical representation of H-basis

way for controlling the level of approximation of the directional irradiance signal. For example, given a normal from the normal map, we can use only the first 4 coefficients (corresponding to the constant and all linear basis functions) to calculate the irradiance. If a higher accuracy is required, we can simply use the two quadratic functions $H^5$ and $H^6$ in addition to the other 4 basis functions.

### 1.8.7 Modified H-basis

For the objects that are far away we may want to use only light mapping without normal mapping at all. In spherical harmonics constant term corresponds to a light map. But if we want to use light map in H-basis we need $H^1$ and $H^3$, because they evaluate to a non-zero value with a tangent space surface normal $n = (0, 0, 1)$. To overcome that problem H-basis was modified by changing the $H^3$ in the following way:

$$H^3_{\text{mod}} = \sqrt{\frac{3}{2\pi}}(1 - \cos\theta) = \sqrt{\frac{3}{2\pi}}(1 - z) \tag{1.58}$$

Orthonormality, which is not required for representing directional irradiance, of the basis set was sacrificed for this advantage.
Through this modification, the constant basis function $H^1$ now represents the traditional light map, while the high accuracy of the original H-basis is still maintained.

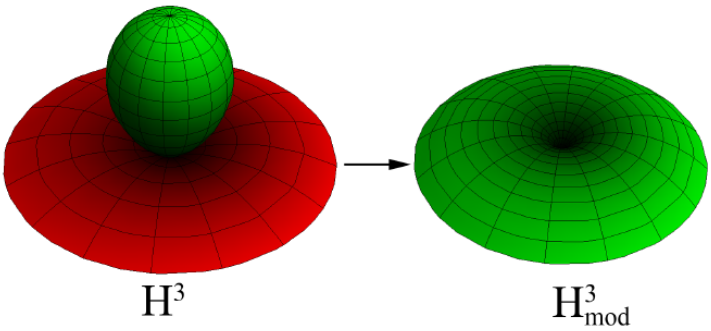



Figure 1.14: Function $H^3$ from H-basis and $H_mod^3$ from modified H-basis.

## 1.9 Surface and Volumetric Representation

There are two main approaches when using irradiance normal mapping: surface representation and volumetric representation.

**Surface representation** is a method of storing additional textures with encoded lighting information (using basis functions) for every object in the scene. For example, if we decide to use 3 bands of Spherical Harmonics for Irradiance signal encoding, we have to store 9 coefficient textures. Although, different objects in the scene may have the same diffuse texture, coefficient textures with lighting information has to be unique for every object because lighting may be different at the places where diffuse texture is the same. But surface representation can be used only for static objects, because lighting is precalculated and cannot be recalculated in real-time.

**Volumetric representation**, also called Irradiance Volumes [13] is a method for lighting of the dynamic objects in the scene. The basic idea is to construct a volumetric uniform grid of points in the scene and take Irradiance samples at every point and store them in Spherical Harmonics (or other spherical basis function set) during the precomputation step. Then in real-time for every dynamic object in their pixel shaders interpolate

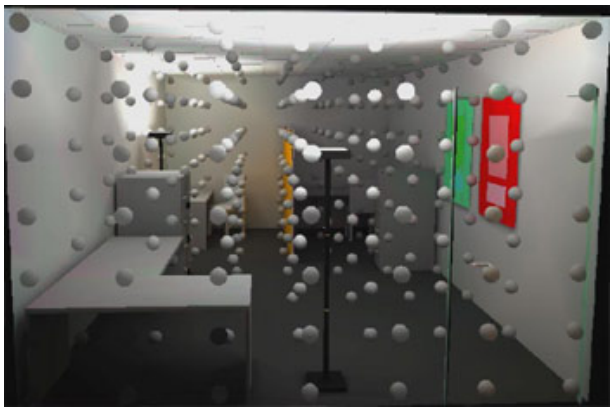


Figure 1.15: Volumetric irradiance sampling grid [6]

the SH coefficient values between closest samples and evaluate the Spherical Harmonics in the same way as in surface representation.
The volumetric data can be sparse, because generally irradiance does not change frequently in space.

Using this method global illumination effect could be acheived for the dynamic objects in the scene. However, lighting interactions of dynamic objects with static objects such as self-occlusion and shadows from dynamic objects are not provided.

In practice, combination of these two approaches can be used: surface representation for static objects and volumetric representation for dynamic objects.

# Chapter 2

# Implementation

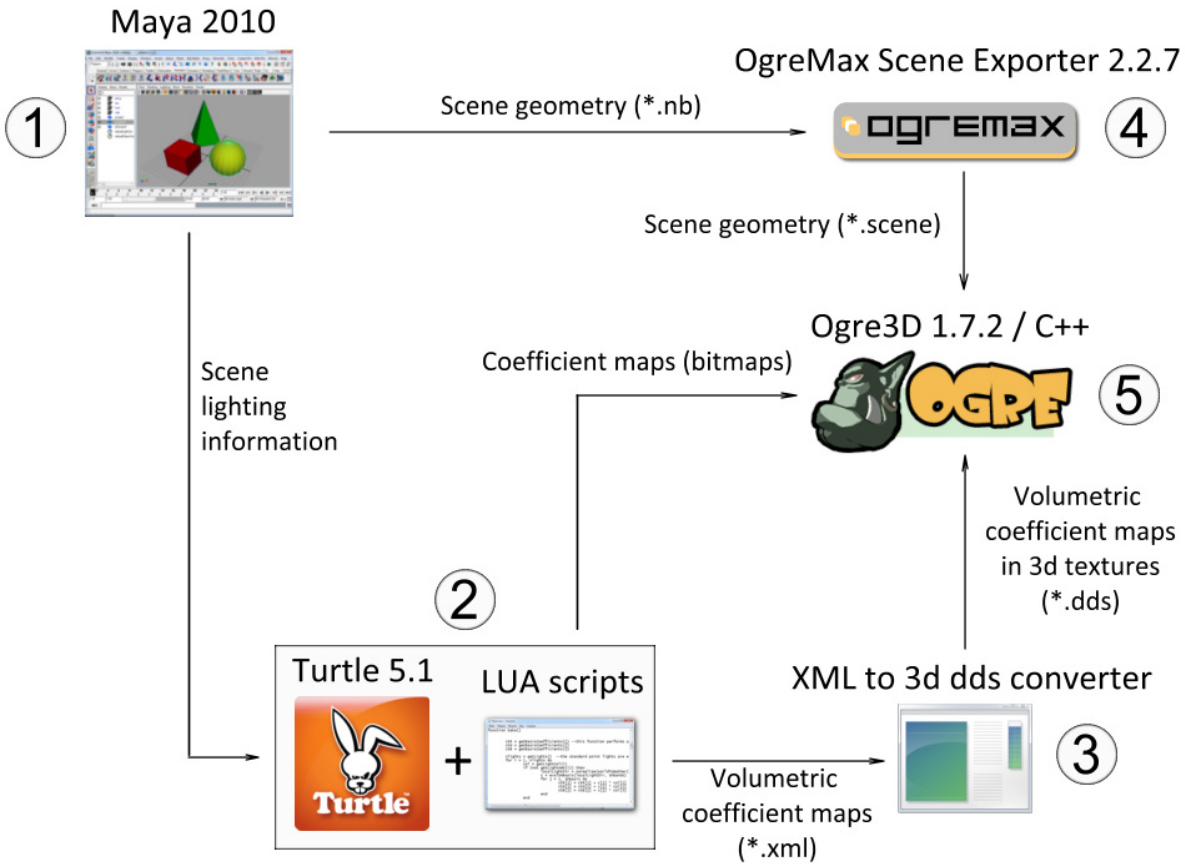



Figure 2.1: Implemented Irradiance Normal Mapping pipeline

The implemented Irradiance Normal Mapping pipeline consists of the following steps:

1. Scene creation and preparation (Autodesk Maya 2010)
2. Coefficient maps baking (Turtle 5.1 + LUA scripts)
3. Conversion of volumetric data in XML to 3D dds
4. Scene export to Ogre3D (OgreMax Scene Exporter 2.2.7)
5. Irradiance Normal Mapping application (Ogre3D 1.7.2)

## 2.1 Scene Creation and Preparation Step

During this step we create a scene in Autodesk Maya 2010 [1]. The complete scene is set up, including geometry, materials and UV paramterization as well as lighting. When creating materials the corresponding normal maps have to be added. Not only can we create static objects, but also animated objects.

### 2.1.1 Preparation for the 2D Coefficient Maps Baking

After the scene is created we have to prepare it for the coefficient maps baking. For every static object in the scene we have to create the secondary UV set in which every texture coordinate lies in the range [0, 1] and is fully non-overlapping. This UV set will be used for 2D coefficient maps.

### 2.1.2 Preparation for Irradiance Volumes

In the case of Irradiance Volumes we have to create a uniform volumetric sample grid. One way to do it in Maya 2010 is to create an array of realtively small quads and place them with a constant distance step. Later the irradiance will be sampled at the vertices of the quads. The number of the quads depends on the size of the scene and level of detalization that is required (e.g. 16x16x16 quads or 32x32x32 vertices). We have to make sure that the quads do not interact with other objects in the scene in terms of lighting i.e. they do not create any shadows or reflect any light. To ensure that we have to adjust the following properties of the quads. In object render stats we have only to leave the "Receive shadows" box checked. In the material settings: in the tab "Special effects" we check "Hide source" and in the tab "Matte Opacity" select Matte Opacity Mode "Black Hole".

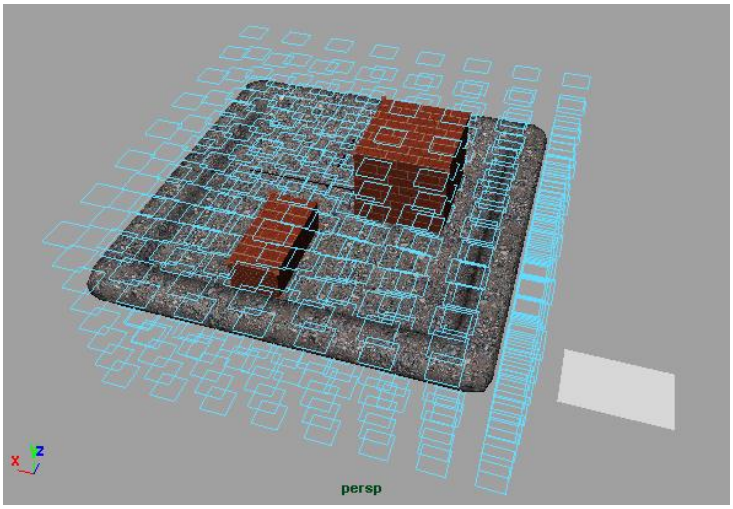



Figure 2.2: Example of irradiance sample grid

## 2.2 Coefficient Maps Baking

For 2D and volumetric coefficent maps baking we used the rendering and baking Maya plug-in Turtle 5.1 [2]. Turtle can be customised to bake out into any basis using LUA scripting.

### 2.2.1 Calculating Irradiance

To calculate Irradiance, using equation 1.4, we need to calculate the radiance $L\left(\vec{x},\vec{\omega}\right)$. Given the radiance $L\left(\vec{x},\vec{\omega}\right)$, calculating $E\left(\vec{x},\vec{n}\right)$ (equation 1.1) for a point $\vec{x}$ corresponds to filtering L with a diffuse (cut cosine) kernel. Doing this in Euclidean or spherical coordinates is prohibitively expensive because we have to filter a large number of surface points. Instead, we use Spherical Harmonics as an intermediate basis in which the filtering can be done much more efficiently.

As shown by R. Ramamoorthi and P. Hanrahan, a spherical directional irradiance signal is faithfully represented with 3 Spherical Harmonics bands (9 coefficients per color channel). Therefore, we only need to use Spherical Harmonics up to the quadratic band.

First, we rotate the sampled radiance of a surface point into the tangent space and expand it into Spherical Harmonics coefficients $s_m^l$ by integrating against the Spherical Harmonics basis functions $Y_m^l$ over the upper hemisphere $\Omega_+$:

$$s_m^l = \int_{\Omega_+} L(\vec{\omega}) Y_m^l(\vec{\omega}) d\vec{\omega} \tag{2.1}$$

In almost all cases, the coefficients are calculated using Monte Carlo Integration

$$s_m^l \approx \frac{2\pi}{N} \sum_{i=1}^{N} L(\vec{\omega_i}) Y_m^l(\vec{\omega_i}) \tag{2.2}$$

where N is the number of hemispherical equally distributed radiance samples $L(\vec{\omega_i})$. More advanced methods such as importance sampling can be applied, as long as a radiance estimate represented in Spherical Harmonics is calculated. The diffuse convolution reduces to an almost trivial step in this representation. Following R. Ramamoorthi and P. Hanrahan, applying the Funk-Hecke-Theorem, integrating with a cut cosine kernel corresponds to multiplying the coefficients of each band with a corresponding factor $a^l$:

$$a^0 = 1 \quad a^1 = \frac{2}{3} \quad a^2 = \frac{1}{4} \tag{2.3}$$

to arrive at the final directional irradiance signal represented in Spherical Harmonics:

$$E_{\mathrm{SH}}(\vec{n}) = \sum_{l,m} s_m^l Y_m^l(\vec{n}). \tag{2.4}$$

We have built the necessary division by $\pi$ for the exitant radiance into the diffuse kernel so we do not have to perform a division at run-time.

### 2.2.2 Irradiance Tangent Space

In irradiance normal mapping it may happen that both normal and coefficient map tangent spaces have different orientations and the normal map resides in a different tangent space than the irradiance.

Fortunately, both texture coordinate sets share the same vertices and geometric normals as well as interpolation in the rendering pipeline. By simply using the tangent and bitangent of the normal map tangent space in the irradiance texture coordinates during the precomputation, a tangent space for the the irradiance texture coordinates is constructed that is correctly oriented with the normal and albedo map tangent space.

Since both tangent spaces are aligned, we need neither the normals nor the (bi)tangents of any tangent space when evaluating the irradiance at run-time. However, we need them when using Irradiance Volumes method, because in that case we have to convert normals from normal map in tangent space to object space for Irradiance evaluation.

### 2.2.3 Texture Normalization and Range Compression

The output of the precomputation are пæЂoating point values which, can also be negative. Though we could use пæЂoating point texture formats to represent the sign and full dynamic range of the irradiance, this option is prohibitively expensive in most cases, though it leads to the best possible output.

We treat the map containing constant term (light map) differently, because it can never get negative, in contrast to other coefficients, Additionally, we change the color space from linear to sRGB by exponentiation with 1/2.2 to avoid quantization artifacts associated with an 8-bit linear representation. As with standard texture maps, we have to interpret the texture as sRGB data when evaluating it at run-time.

Because coefficients other than constant can be negative, we perform a similar range compression as done with normal maps, keeping them in linear color space. If constant coefficient is in the range [0..1], a rule of thumb is that the other coепæЋcients do not exceed the range of [-0,75..0.75]. However, the range compression factor of 0.75 may be changed according to the scene and lighting conditions to further increase the accuracy of the coепæЋcient representation over a range compression of [-1..1], as used in normal mapping.

### 2.2.4 Texture Compression

Similar to light maps, the coefficient maps can be compressed using DXT texture formats. Since we do not need any alpha information, DXT1 compression can deliver a compression rate of $6:1$. Since we епæЃectively add up several DXT compressed textures, compression artifacts may also add up to an intolerable level. These artifacts can be counteracted by choosing a higher coefficient map resolution or modulating the result with an albedo map to obfuscate the color shifts.

### 2.2.5 Baking Volumetric Data

To bake volumetric data we first select our volumetric quads grid in Maya, then select in Turtle to bake to vertices in XML format, using Spherical Harmonics LUA script. We use Spherical Harmonics because in this case we deal with volumetric data and we need to have Irradiance signal in a full sphere domain.

## 2.3 Conversion of Volumetric Data in XML to 3D dds

Volumetric data XML is converted into 3D dds format using a small console application, which we specially developed for that purpose. It uses tinyxml library to read and process XML data and then convert it to 3D dds using DirectX library.

Listing 2.1: Pseudo-code for calculating the coefficients for the modified H-basis including the texture optimizations.

```
for each surface point do {

  color SHc[9] //3 bands Spherical Harmonics coefficients

  //Monte-Carlo-Integration in tangent space over
  //the hemisphere to project into SH[] basis functions
  for each radiance sample L(direction) in N do {
    for each SHc do {
      SHc[]  +=  L(direction)*SH[](direction)
    }
  }
  SHc[] = (2*PI/N)*SHc[]

  //Diffuse convolution
  SHc[0] = SHc[0]
  SHc[1,2,3] = 2.0/3.0*SHc[2,3,4]
  SHc[4,5,6,7,8] = 1.0/4.0*SHc[4,5,6,7,8]

  //Projection into modified H-basis
  color modHc[6] // modified H-basis coefficients

  //Transform matrix
  for each color in modHc[] do {
    modHc[0] =  0.70711*SHc[0]+1.2247*SHc[2]+1.1859*SHc[6]
    modHc[1] =  0.70711*SHc[1]+0.59293*SHc[5]
    modHc[2] = -0.70711*SHc[2]-1.3693*SHc[6]
    modHc[3] =  0.70711*SHc[3]+0.59293*SHc[7]
    modHc[4] =  0.70711*SHc[4]
    modHc[5] =  0.70711*SHc[8]
  }

  //Convert first coefficient to sRGB
  modHc[0] = pow(modHc[0],1/2.2)

  //Range compress rest with global factor of 0.75
  for each modHc[] except modHc[0] do {
    modHc[] = modHc[]+0.75/(2*0.75)
  }

  write modHc[]
}
```

Listing 2.2: HLSL code for evaluating the modified H-basis, including a modulation with an albedo map. The different levels of detail are created by stopping the irradiance calculation at the shown points.

```
float3 n = 2*tex2D(normal,texUV)-1; //tangent space normal map

float3 irr =
  0.39894*tex2D(h1,lightUV) //is sRGB lookup ( x^2.2 )
  //stop here for lowest LOD (lightmap)
  +(2*0.75*tex2D(h2,lightUV)-0.75)*0.69099*n.y //not sRGB lookup
  +(2*0.75*tex2D(h3,lightUV)-0.75)*0.69099*(1-n.z)
  +(2*0.75*tex2D(h4,lightUV)-0.75)*0.69099*n.x
  //stop here for middle LOD
  +(2*0.75*tex2D(h5,lightUV)-0.75)*1.54509*n.x*n.y
  +(2*0.75*tex2D(h6,lightUV)-0.75)*0.77255*(n.x*n.x-n.y*n.y);
  //full evaluation

color = irr*tex2D(albedo,texUV) //is sRGB lookup ( x^2.2 );

//write color to sRGB frame buffer ( x^(1/2.2) )
```

## 2.4 Scene Export to Ogre3D

The scene is exported from Maya 2010 to Ogre3D [3] via Maya plug-in OgreMax Scene Exporter. Before exporting, in OgreMax scene settings chekboxes "Generate Binormals" and "Generate Tangents" have to be checked for use with Irradiance Volumes. If there is any animation in the scene it has to be added in the OgreMax object settings.

## 2.5 Run-time Evaluation

### 2.5.1 Evaluation with 2D Coefficient Maps

During run-time Irradiance values are obtained through basis function evaluation. This step is performed in the pixel shader that takes as an input corresponding coefficient maps, normal maps, albedo textures.

### 2.5.2 Evaluation with Irradiance Volumes

Difference in evaluation of Irradiance Volumes from 2D evaluation is that instead of 2D textures we have volumetric textures and additionaly normal, tangent and binormal vectors. The main idea is to align the precomputed volumetric grid with the scene objects in real-time application. To do that we have to know dimensions of the volumetric grid and origin offset of the center of the volumetric grid.

Listing 2.3: HLSL code for evaluating the Irradiance Volumes with SH, including a modulation with an albedo map.

```
// a normal from normal map in tangent space
float3 ts_normal = 2*tex2D(normalmap,normalUV)-1;

 // interpolated normal in world space
normal = mul(normal, toWorld);

// final normal in world space
normal = normalize(normal);
float3 ws_normal = tangent * ts_normal.x
                 + binormal * ts_normal.y
                 + normal * ts_normal.z;

 // widht, height, depth - dimensions of the volumetric grid
float3 width = float3(width, height, depth);

// centerx, centery, centerz - origin offset of the center
// of the volemetric gird
float3 offset = float3(centerx, centery, centerz);

float3 lightUVW = position3d + (width/2);
lightUVW += offset;

// volumetric texture coordinate normalization
lightUVW.x/=width.x;
lightUVW.y/=width.y;
lightUVW.z/=width.z;

lightUVW.z = -lightUVW.z;
lightUVW = lightUVW.zxy;
ws_normal.xyz = ws_normal.xzy;
ws_normal.y = -ws_normal.y;

// SH evaluation
float3 irr = tex3D(volume0,lightUVW)*0.28209479 //is sRGB lookup ( x^2.2 )
  +(2*tex3D(volume1,lightUVW)-1)*0.48860251*(ws_normal.y)
  +(2*tex3D(volume2,lightUVW)-1)*0.48860251*(ws_normal.z)
  +(2*tex3D(volume3,lightUVW)-1)*0.48860251*(ws_normal.x)
  +(2*tex3D(volume4,lightUVW)-1)*1.09254843*ws_normal.x*ws_normal.y
  +(2*tex3D(volume5,lightUVW)-1)*1.09254843*ws_normal.z*ws_normal.y
  +(2*tex3D(volume6,lightUVW)-1)*0.31539156*(3*ws_normal.z*ws_normal.z-1)
  +(2*tex3D(volume7,lightUVW)-1)*1.09254843*ws_normal.z*ws_normal.x
  +(2*tex3D(volume8,lightUVW)-1)*0.54627421*(ws_normal.x*ws_normal.x
  - ws_normal.y*ws_normal.y);

color = irr*tex2D(albedo,texUV) //is sRGB lookup ( x^2.2 );

//write color to sRGB frame buffer ( x^(1/2.2) )
```

# Chapter 3

# Results

## 3.1 Analytical Comparison

Since all described basis function sets can carry an approximation of the same irradiance signal we can directly compare them against each other to determine their accuracy. As an error metric, we use the integrated mean square error.

$$IMSE = \int_{\Omega} \left( \sum_{i=1}^{n} c_i B_i \left( \vec{x}, \vec{n} \right) - E \left( \vec{x}, \vec{n} \right) \right)^2 d\vec{n} \tag{3.1}$$

where $E\left(\vec{x}, \vec{n}\right)$ is the fully correct irradiance signal and $c_i B_i$ the corresponding weighted basis function of each basis. The IMSE of all bases with different numbers of coefficients averaged over 10,000 random irradiance signals is shown in Figure 3.1 and Table 3.1.

Table 3.1: Average of the integrated mean square error over 10,000 random irradiance functions grouped by number of coefficients.

| | **Number of basis functions** | | | |
|---|---|---|---|---|
| **Basis** | **3** | **4** | **6** | **9** |
| Spherical Harmonics | - | 0.02748 | - | 0.00231 |
| Hemispherical Harmonics | - | 0.03705 | - | 0.01114 |
| Makhotkin Basis | - | 0.06577 | - | 0.03030 |
| Zernike Basis | 0.05871 | - | 0.01010 | - |
| H-basis (modified H-basis) | - | 0.02779 | 0.00678 | - |
| Half-Life 2 Basis | 0.07718 | - | - | - |

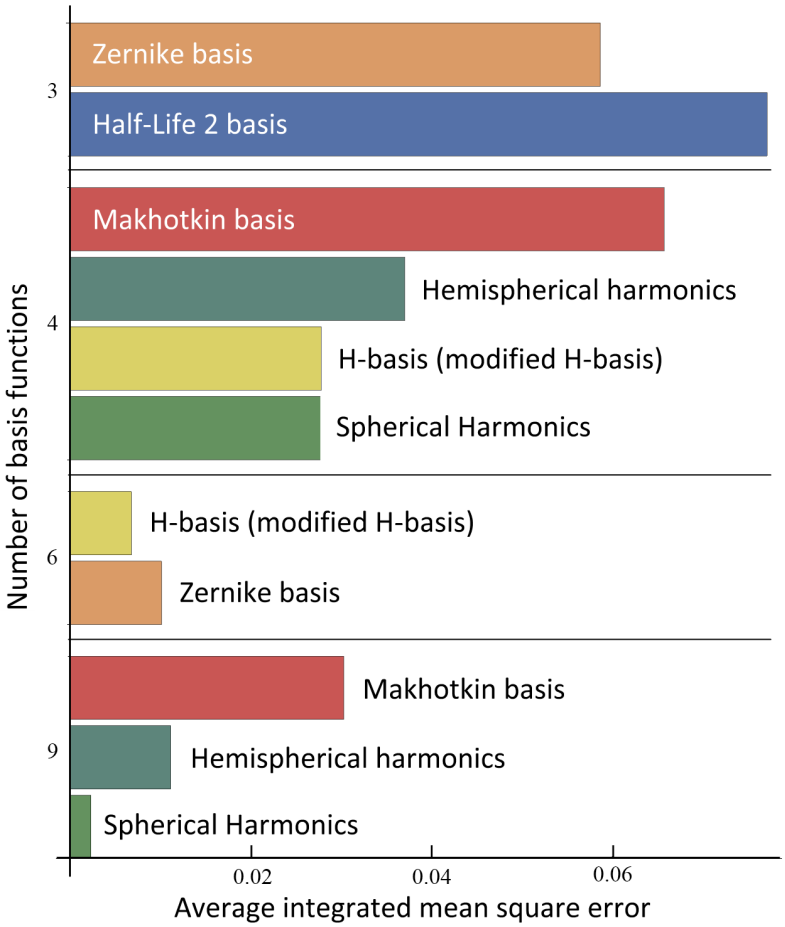



Figure 3.1: Average of the integrated mean square error over 10,000 random irradiance functions grouped by number of coefficients.

## 3.2 Visual Comparison

Figures 3.3 and 3.4 show a direct comparison of all bases using two or three bands under the same lighting conditions. The used lighting is an HDR environment map with the sun at grazing angle, which poses a worst case situation because the irradiance changes strongly in the longitudinal as well as latitudinal parameter. A typical in-game scene that shows different basis function sets is shown in Figure 3.5. According to the results of

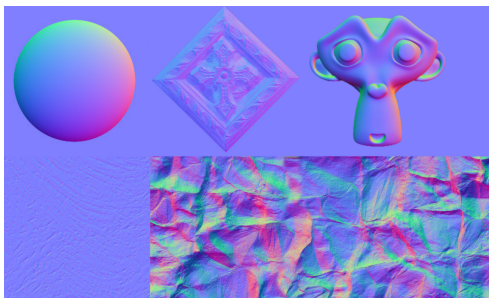


Figure 3.2: Normal map used for visual comparison of basis function sets.

the visual comparison Spherical Harmonics is the most accurate method, but at least 9 functions are needed. SH are purely polynomial and contain only multiplications, they can

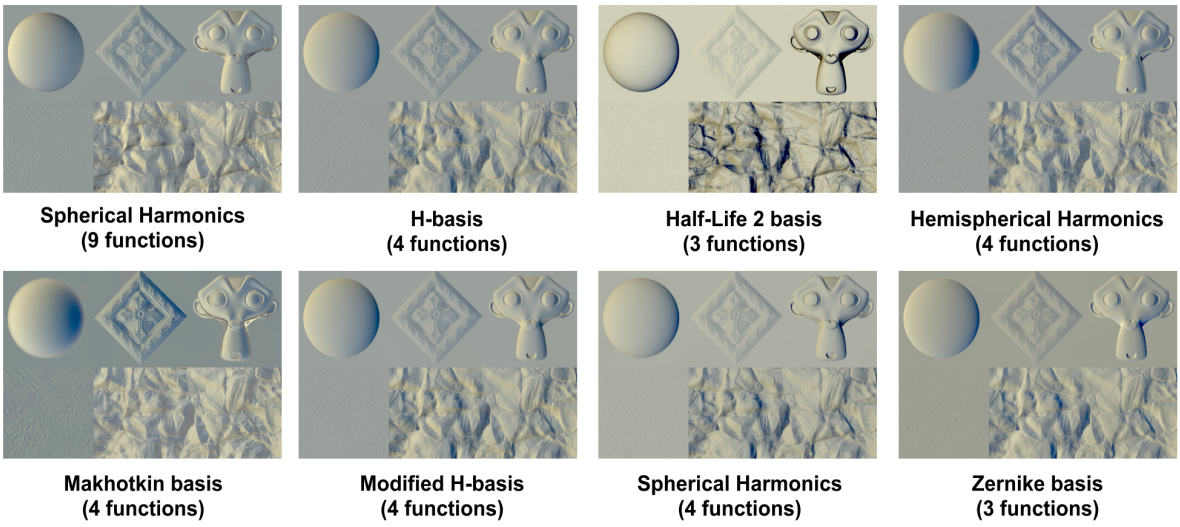



Figure 3.3: Visual comparison of linear bands of different basis function sets. SH with 9 functions are shown for clarity.

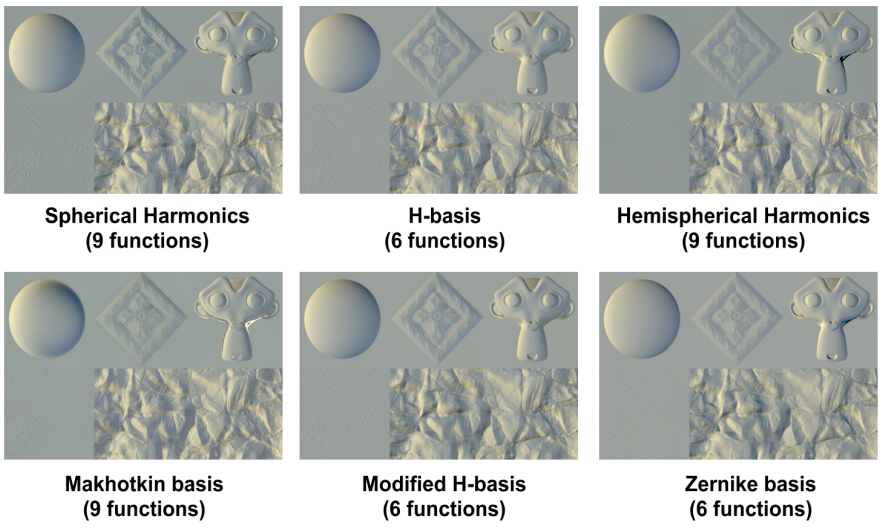



Figure 3.4: Visual comparison of quadratic bands of different basis function sets

be efficiently evaluated in the pixel shader. However, Irradiance in SH is not efficiently stored, because the data in lower hemisphere in never used, when using 2D coefficient maps.

Although, Hemispherical Harmonics unlike Spherical Harmonics contain information only about the upper hemipshere, due to the shifting operation they become strongly non-polynomial and expensive to evaluate in the pixel shader. In addition, there are severe errors at the border of the hemisphere, which is clearly noticeable in Figure 3.3. However, in Figure 3.4 we can see that error is decreasing with the increase of the number of functions.

Makhotkin basis is also non-polynomial due to the shifting operation as well as HSH. Furthermore, the error is higher than in HSH, which is clearly visible on Figures 3.3 and 3.4.

Half-Life 2 basis and Zernike basis are among the most inaccurate methods, but they have the least memory requirements вЃ Y only 3 functions are needed.

The most successful hemispherical basis is modified H-basis, because it's error with 6

functions is very close to SH with 9 functions. In case of 4 functions the error is almost the same as SH with 4 functions. Modified H-basis can be effectively evaluated in pixel shader due to it's purely polynomial structure. As a result when using modified H-basis with 6 functions we get the Irradiance signal representation which is mathematically very close to SH with 9 functions and visually indistinguishable from SH. In addition with modified H-basis we can easily control the level of detail of a scene by using only $H^1$ (light mapping) for objects that are far away, and then adding sequentially the linear and quadratic band.

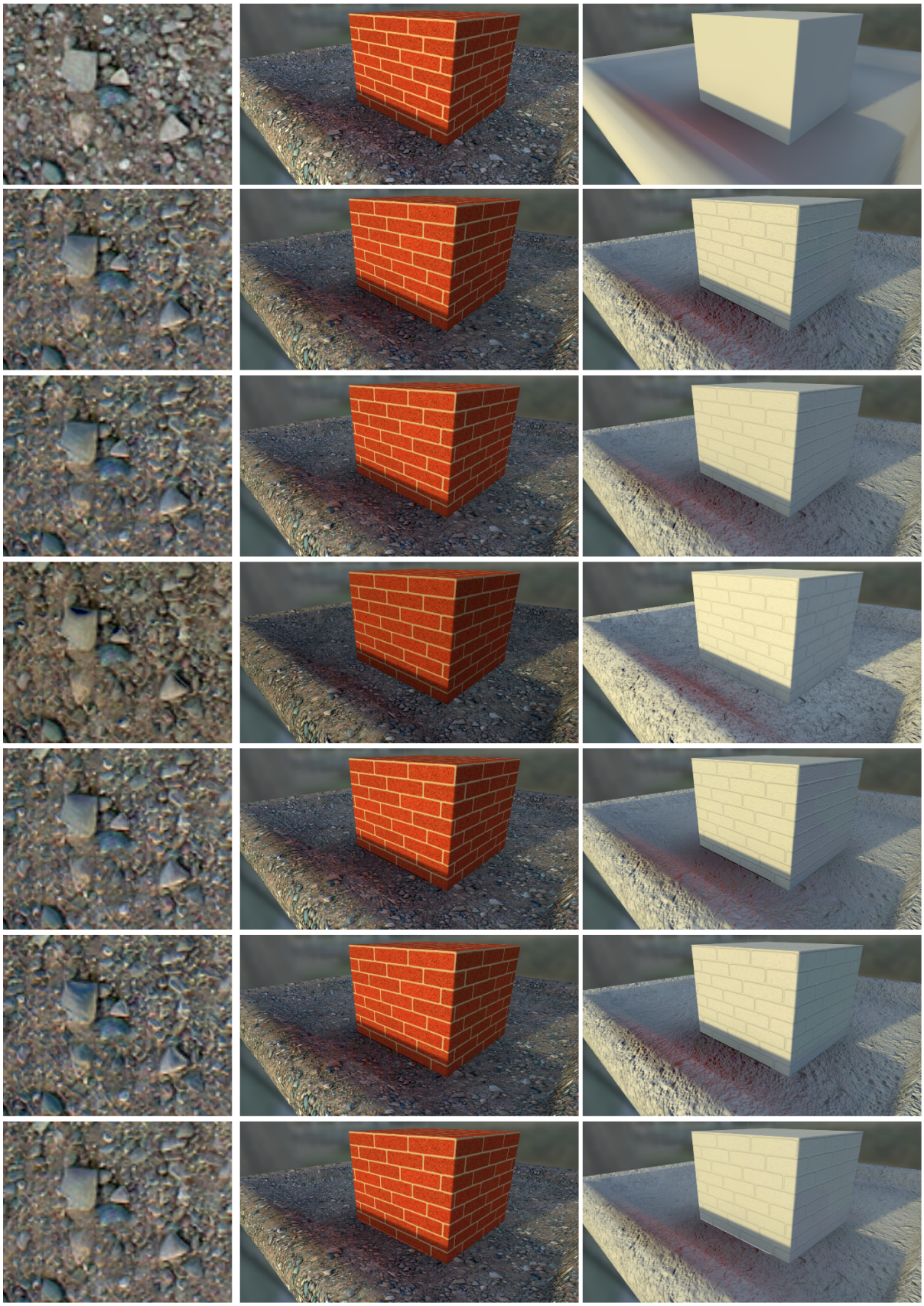


Figure 3.5: In-game scene showing different basis function sets. Top to bottom (1) Light Mapping, (2) Modified H-basis, (3) Spherical Harmonics, (4) Half-Life 2 basis, (5) Hemispherical Harmonics, (6) Makhotkin basis, (7) Zernike basis

Figure 3.6: In-game scene with Irradiance volumes. Here 9 volumetric textures with the resolution 16x16x8 are used to store 9 Spherical Harmonics coefficients

In Figure 3.6 we can see that results obtained with Irradiance Volumes differ from those with 2D coefficient maps. For example, several artifacts are caused by the interpolation of Irradiance samples which are inside objects (black) and outside objects (white) resulting in a gray color in lit object. However, such undesirable effect could be decreased by using higher resolution volumetric coefficient textures. Even better way would be to use adaptive subdivision, such as Octree, of Irradiance grid instead of uniform grid.

# Conclusion

We have implemented complete Irradiance normal mapping pipeline as well as discussed necessary theoretical background and implementation details. We have compared visually and analytically all current basis function sets used for Irradiance signal representation and approaches to Irradiance normal mapping: Irradiance volumes and 2D coefficient maps.

Figure 3.7: A scene with modified H-basis

Figure 3.8: The Sponza Atrium scene with modified H-basis. Top to bottom (1) withoud albedo, (2) with albedo, (3) with albedo and HDR tone mapping

# Bibliography


[1] Autodesk maya. maya is a registered trademark or trademark of autodesk, inc. in the usa and other countries, 2010. `www.autodesk.com`.

[2] Illuminate labs: Turtle for maya, 2010. `www.illuminatelabs.com`.

[3] Ogre graphics engine, 2010. `www.ogre3d.org`.

[4] Gautron, P., Krivánek, J., Pattanaik, S. N., and Bouatouch, K. A novel hemispherical basis for accurate and efficient rendering. In *Rendering Techniques* (2004), pp. 321–330.

[5] Green, R. Spherical harmonic lighting: The gritty details. *Archives of the Game Developers Conference* (March 2003).

[6] Greger, G., Shirley, P., Hubbard, P., and Greenberg, D. The irradiance volume. *IEEE Computer Graphics & Applications 18*, 2 (1998), 32–43.

[7] Habel, R., and Wimmer, M. Efficient irradiance normal mapping. In *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2010), ACM, pp. 189–195.

[8] Kajiya, J. T. The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM, pp. 143–150.

[9] Koenderink, J. J., Doorn, A. J. v., and Stavridi, M. Bidirectional reflection distribution function expressed in terms of surface scattering modes. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume II* (London, UK, 1996), Springer-Verlag, pp. 28–39.

[10] Makhotkin, O. A. Analysis of radiative transfer between surfaces by hemispherical harmonics. *Journal of Quantitative Spectroscopy and Radiative Transfer 56*, 6 (1996), 869–879.

[11] McTaggart, G. Half-Life 2/Valve Source Shading. Tech. rep., Valve Corporation, 2004.

[12] Sloan, P.-P., Kautz, J., and Snyder, J. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM, pp. 527–536.

[13] Tatarchuk, N. Irradiance volumes for games. In *GDC-E 2006* (London, England, 2006).