FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Realistic Local Lighting in Dynamic Height Fields

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Aaron Meier-Stauffer
Matrikelnummer 9927153

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Univ. Ass. Dipl.-Ing. Paul Guerrero

Wien, 13.03.2013

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Realistic Local Lighting in Dynamic Height Fields

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Aaron Meier-Stauffer
Registration Number 9927153

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Univ. Ass. Dipl.-Ing. Paul Guerrero

Vienna, 13.03.2013

(Signature of Author) (Signature of Advisor)

Erklarung zur Verfassung der Arbeit

Aaron Meier-Stauffer
Rechte Bahngasse 18/10, 1030 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Dedicated to Petra, Benjamin, Alicia, Helmut, Russell and my unborn twins

Abstract

This thesis presents a method to compute soft shadows from environment maps and local light sources on dynamic height fields, extending the work of Snyder et al. [29]. While direct illumination in static scenes is very common in video games and 3D applications, real-time global illumination methods supporting dynamic scenes and lights are still an active field of research.

In this work, a short general introduction to global illumination and spherical harmonics is presented as well as an overview of the state of the art methods in interactive global illumination for height fields.

In our method, visibility at each receiver point of a height field is determined by the visible horizon, which can be approximated efficiently using a multi-resolution sampling approach. Local light sources are represented by spherical lights and the incident radiance at receiver points is projected into the spherical harmonic basis. Hence, this method produces convincing shadows on dynamic height fields more efficiently than global illumination methods for general geometry.

Kurzfassung

Das Ziel dieser Diplomarbeit ist es, weiche Schatten von Umgebungslichtern und lokalen Lichtquellen auf dynamischen Höhenfeldern zu berechnen. Diese Arbeit erweitert die Arbeit von Snyder et al. [29] um lokale Lichtquellen. In heutigen 3D Anwendungen und Video Spielen werden üblicherweise direktionale Lichtquellen und Punktlichter für teils statische und dynamische Szenen verwendet. Die Entwicklung komplett dynamischer Szenen und dynamischer Lichtquellen stellt daher eine große Herausforderung dar.

Diese Diplomarbeit, gibt einen generellen Überblick über globale Beleuchtungsverfahren. Spherical Harmonics werden als Überblick im Kapitel State of the Art eingeführt, gefolgt von aktuellen interaktiven globalen Beleuchtungsverfahren für Höhenfeldern.

Der Kern dieser Arbeit ist für jeden Punkt auf dem Höhenfeld den sichtbaren Horizont zu berechnen. Dieser kann mittels einer Textur-Pyramide effizient angenähert werden. Die lokalen Lichtquellen werden mittels sphärischer Lichtquellen angenähert. Weiters wird die eingehende Strahlung von einer Lichtquelle an einem Punkt auf Kugelflächenfunktionen (Spherical Harmonics) abgebildet. Im Gegensatz zu generellen globalen Beleuchtungsverfahren berechnet unsere Methode weiche Schatten auf Höhenfeldern effizient und überzeugend.

Contents

CHAPTER 1

Introduction

Development of realistic shadowing algorithms for real-time computer graphics has been a major research topic in the past few years by the research community. Scenes without shadows look very poor and flat. Shadows help us perceive the shape of an object and the spatial relationship between objects in a 3D scene. We distinguish between two common shadow types, hard and soft shadows. Hard shadows are produced from infinitely small light sources, e.g. point lights, which do not exist in the real world. In case of hard shadows, a receiver point can either see the light source and is therefore unshadowed, or the receiver point is blocked by the geometry and is therefore fully shadowed, which gives a rather unrealistic look to an image. Soft shadows on the other hand are produced from extended light sources, e.g. the sky, resulting in soft shadows, because a receiver point can have a partial view of the light source. The area where a receiver point has partial view of the light source is called *penumbra*, and the area where the receiver point is totally occluded is called *umbra* (see Figure 1.1, comparing hard- and soft-shadows).

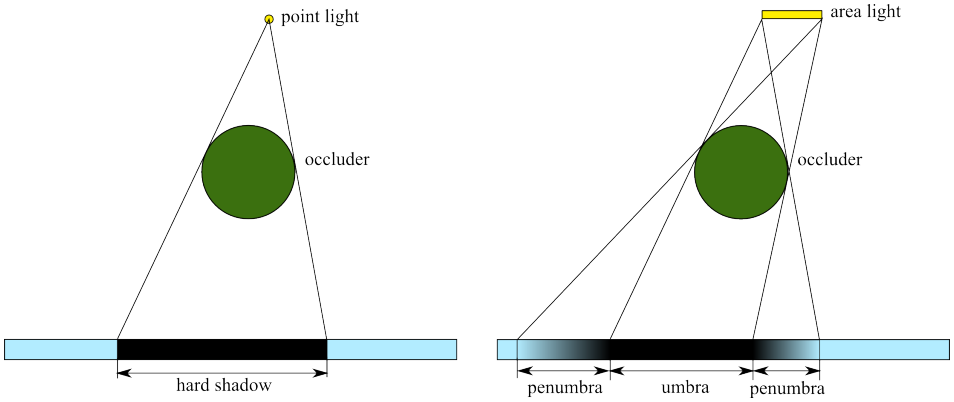


Figure 1.1: The image on left shows the geometry of hard shadows from a point light. In the right image, we can see that an extended light source casts soft shadows.

Soft shadows are especially important in large, open scenes like terrains. In our daily experience, we only see these types of scenes lit by large area lights like the sun and the sky, and we are therefore only used to seeing these scenes with soft shadows. In computer graphics, these scenes are usually represented by height-fields and used in many different applications like map visualizations (e.g. Google Earth [10]), movies, games and architectural visualizations, among others.

For instance, we could approximate the glow of a few buildings or an entire city at nighttime with several local light sources. Or a scene with houses and mountains could be lit by an environment map for the sky light and several local lights for the houses (see Figure 1.2). In a mapping visualization, shadows can provide visual cues that help the viewer understand the geometry of the terrain. The additional contours introduced by hard shadows can be confusing, therefore soft-shadows might be preferable. Height fields can also model smaller-scale surfaces, like panel surfaces in a car. When designing these surfaces, it might be interesting to get a quick approximation of their appearance when lit by local bright elements, e.g. LED lights or the light from a display (see Figure 1.2). A few lighting setups for height-fields can be seen in Figure 1.3.

1.1 Problem Statement

In real-time applications, soft shadows have to be computed very efficiently. In scenes with dynamic geometry or moving light sources, this is a significant challenge and an area of ongoing research. Using global illumination methods for general geometry would be either too slow or too inaccurate to solve this problem (in Chapter 3 we present some current global illumination methods for general geometry). In the special case of height-field geometry, simplifying assumptions can be made that result in increased efficiency to allow real-time soft shadows from dynamic geometry. A method that achieves this has already been developed by Snyder et al. [29], but only for infinitely distant light sources.

In our work, we want extend the method of Snyder et al. [29] to remove the limiting assumption of infinitely distant light sources, i.e. we want to compute soft shadows from *local lights* in dynamic height-fields. This goal presents us with additonal challenges, which can be broken down into two main research questions: namely, how to compute the following functions efficiently in real-time:

1. The unshadowed incident radiance from *local light sources* at receiver points on the height-field, i.e. unshadowed shadowed incident radiance that might be different for each point on the height-field.

2. The visibility of multiple local light sources, i.e. visibility of light sources that might only be occluded by a part of the height-field geometry.

To answer these research questions, our task was to develop methods and proof-of-concept implementations that compute these quantities. The real-time requirement is dependent on the complexity of the input scene. In this work, we take the scene complexity used by Snyder et al. [29] as reference, i.e. methods that are real-time on scenes of similar complexity are considered real-time methods.

(a) Lighthouse at night time (scene A)

(b) Panel surfaces in a car (scene B)

Figure 1.2: The images show two sample scenes (A and B). In both scenes, the geometry can be approximated by height-fields, the glow in scene A and the LED lights or light from display in scene B can be approximated by local light sources. Image courtesy of xdesktopwallpapers (scene A) - www.xdesktopwallpapers.com [3] and tutorialcenter (scene B) - www.tutorialcenter.net [2].

1.2 Contribution

This document presents a method to calculate real-time soft shadows from large, low-frequency area light sources like environment map and local light sources on dynamic height-fields based primarily on the work by Snyder et al. [29]. More specifically, we extend the work of Snyder et al. [29] for soft shadows on height-fields from infinitely distant light sources to handle local lights. For this purpose, we contribute two important methods:

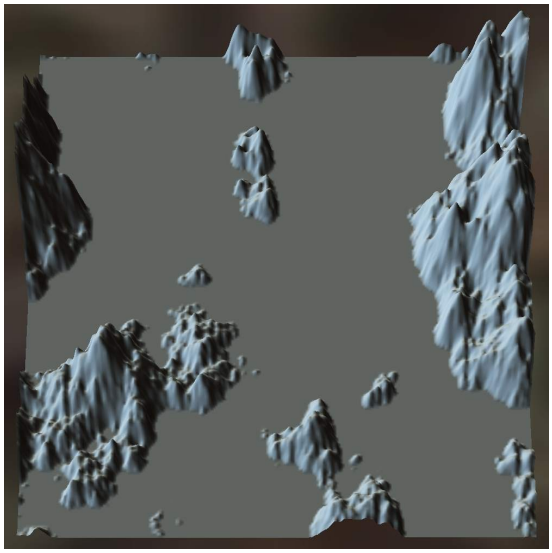
1. A method to efficiently compute the projection of a spherical local light source on the hemisphere of a receiver point in the Spherical Harmonics basis, approximating the incident radiance from the local light at the receiver point.

2. A method to efficiently compute the projection of the geometry blocking a local light on the hemisphere of a receiver point in the Spherical Harmonics basis, approximating the visibility of the local light.

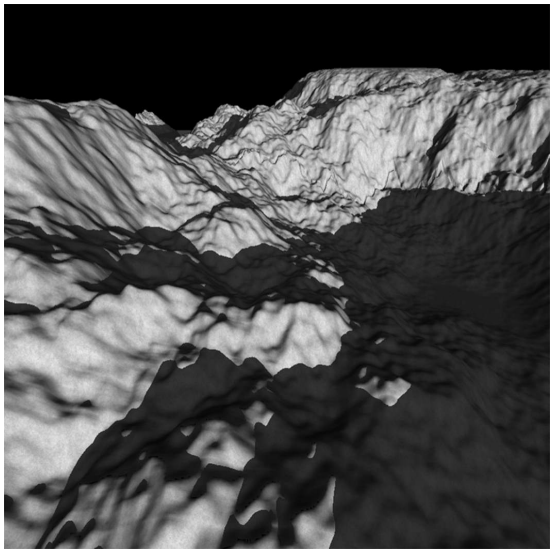
Using these methods, we are able to calculate soft shadows from one environment light and up to three local light sources. The height-field geometry and local light position, size and intensity can be changed at run-time.

1.3 Structure

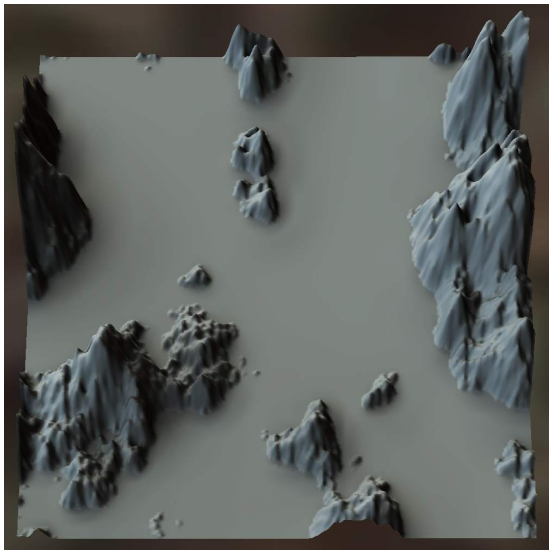
The structure of this document is as follows: In the following chapter, we will give a brief introduction to global illumination 2. Chapter 3 summarizes related methods to compute global illumination and gives a detailed description of spherical harmonics, which are commonly used in current state-of-the-art global illumination methods. In Chapter 4, we explain how to calculate soft shadows on dynamic height-fields lit by an environment map followed by our extended method which calculates soft shadows lit by local light sources. In Chapter 5, we show how to implement relevant parts of the method on the GPU. Finally, in Chapter 6 we show results generated with our method.



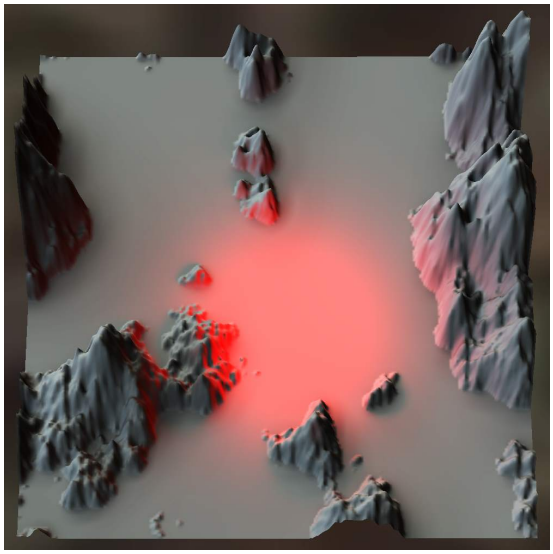
(a) Environment light only - unshadowed (scene A)



(b) Hard shadows only (scene B)



(c) Environment light only - shadowed (scene A)



(d) Local light + environment light - shadowed (scene A)

Figure 1.3: A set of images from two different dynamic height field scenes (A and B), rendered with two different shadowing methods: The top left image shows scene (A) lit by the environment map without shadows, the top right image shows scene (B) with hard shadows from a directional light source (Image courtesy of Sakalauskas [22]). The bottom left image shows the scene (A) with shadows from environment lighting and the bottom right image shows the scene (A) with shadows from one local light and environment lighting. Note how the terrain rendered with hard shadows introduces additional contours at the shadow borders that do not look natural and make it more difficult to understand the geometry of the terrain.

CHAPTER 2

Global Illumination

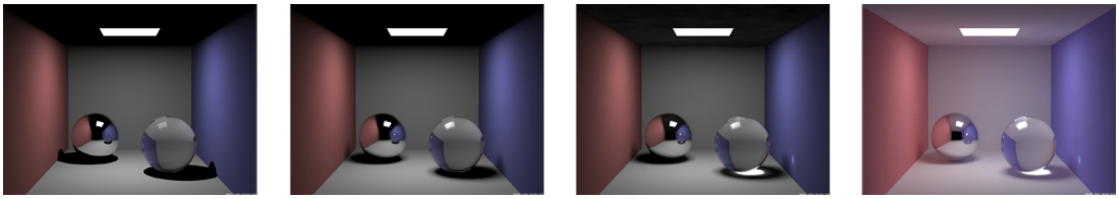


Figure 2.1: The fist image shows diffuse and specular reflections, rendered with a raytracer. In the second scene, soft shadows are added. The third image shows caustics, and in the last one, we can see indirect illumination. Image courtesy of `http://graphics.ucsd.edu/~henrik/images/cbox.html`

In this chapter, we want to give an overview of some of the concepts from global-illumination that are relevant for our method, including the light transpot notation, material properties and the light transport equation. Our method is not a global-illumination method in the true sense of the word, since we do not compute indirect illumination. However, we make use of the mathematical tools and terminology commonly used in global-illumination methods.

Intuitively, global-illumination can be described as a general scene being illuminated by lights (e.g. a room with mirrors, tables and chairs in it). We follow photons, emitted by the light sources. Photons while bounce around can be reflected, absorbed and refracted in different combinations.

Compared to local lighting algorithms (direct illumination), where photons travel from the light source to a surface and then to the eye, global-illumination algorithms take into account more complex cases, where photons emitted from the same light source are reflected by other objects, those reflected ones can then again be reflected by other objects and so on, until they reach the eye. Images generated by global-illumination algorithms will greatly increase realism compared to direct illumination algorithms.

A variety of photon path combinations exist where photons travel along a path from the light source, hitting zero or nearly an infinite number of diffuse or specular surfaces until they reach the eye along these more complicated paths. It would be nice to have a uniform scheme for describing the path combinations. This leads us to Paul Heckbert's [9] notational scheme, which he introduced in 1990 at SIGGRAPH.

2.1 The Light Transport Notation

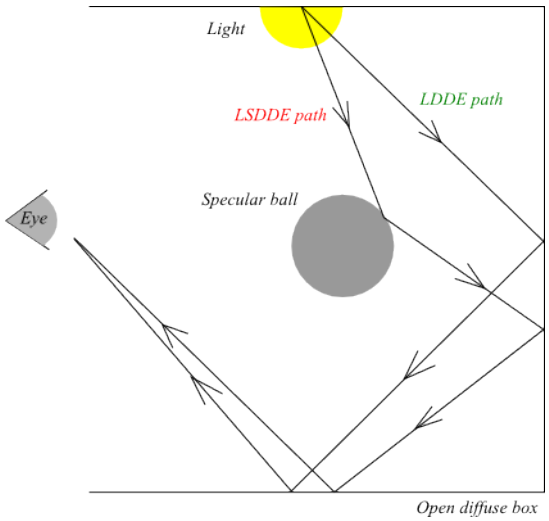


Figure 2.2: Example light paths in light transport notation

If we follow a photon walking through the scene from the light source to the eye, we can distinguish between a specular and a diffuse interaction a photon makes (see Figure 2.2). Moreover, additional surface types can be added, such as glossy surfaces to further refine the categories. To summarize all different kinds of interactions, regular expressions are mainly used. See Table 2.1.

Operator	Description	Example	Explanation
+	one or more	D+	one or more diffuse reflection
*	zero or more	S*	zero or more specular reflection
?	zero or one	S?	zero or one specular reflection
|	or	DD|S	two diffuse reflections or one specular reflection
()	group	(D|S)*	zero ore more diffuse ore specular reflections

Table 2.1: Light Transport Notation - D ... Diffuse, S ... Specular

Photons can take different paths from the light source to the eye, a various set of light paths exists therefore. The most general equation in global-illumination literature that describes the

energy transported along all kinds of light paths is called the light transport or render equation, which was formulated by Kajiya [14] in 1986. Using the light transport notation, we can sum up all relevant light path combinations by the expression L(S|D)*E to render a realistic image.

In the next section we will take a look at the material properties before explaining the render equation in detail.

2.2 Material Properties

The material properties describe how light is reflected and refracted at a surface, this function is called the *bidirectional scattering distribution function* (BSDF) and normally subdivided into reflected and transmitted parts, each of them are calculated separately. Reflection and Transmission of a surface is called *surface scattering*. As the name of the function implies, it is a function describing how light is scattered from a surface.

First of all, we will look at surface reflectance, which is described by the *bidirectional reflection distribution function* (BRDF), denoted f_r. It is defined as the ratio of reflected radiance dL_r along direction ω_o to the irradiance dE_i entering at the surface normal n from an incident direction ω_i.

$$f_r(p, \omega_i, \omega_o) = \frac{dL_r(p, \omega_o)}{dE_i(p, \omega_i)} \tag{2.1}$$

This function depends on the incident light direction ω_i and the exiting light direction ω_o. Each direction is parametrized through an elevation angle Θ and a rotation angle ϕ, both defined with respect to the surface normal n and a tangent vector t (see Figure 2.3). Further, for a surface point p the BRDF is a four-dimensional function and can be measured by a gonioreflectometer or described by empirical models like Phong, Lambertian, Ward, Oren-Nayar, Cook-Torrance and more elaborated ones.

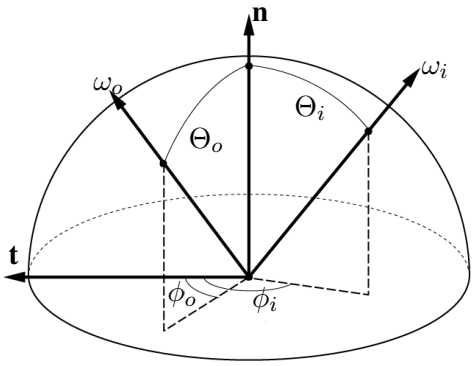


Figure 2.3: Rotational angles are given with respect to a given tangent vector t.

If we fix a particular direction ω_i with an infinitely small cone around ω_i, which occupies an infinitely small solid angle $d\sigma(\omega_i)$, the irradiance $dE(w_i)$ is the light entering through that cone at the surface point p.

$$dE(\omega_i) = L_i(\omega_i) d\sigma_\perp(\omega_i) \tag{2.2}$$

Where $d\sigma_\perp(\omega_i)$ is the differential projected solid angle and is defined by

$$d\sigma_\perp(\omega) = cos\Theta d\omega \tag{2.3}$$

Thus we can re-write Equation 2.1 as:

$$f_r(p, \omega_i, \omega_o) = \frac{dL_r(p, \omega_o)}{dE_i(p, \omega_i)} = \frac{dL_r(p, \omega_o)}{L_i(p, \omega_i) cos\Theta d\omega_i} \tag{2.4}$$

Transmissive surfaces are defined by the *bidirectional transmittance distribution function* (BTDF), denoted f_t. Since transmissive surfaces are hard to measure, perfectly specular reflection is often considered (see Figure 2.4).

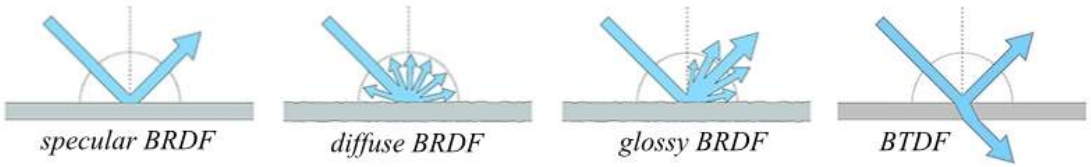


Figure 2.4: Different diagrams representing BRDFs and BTDF - Image courtesy of `http://randomcontrol.com`

When light interacts on a surface of an object, we can find different kinds of scattering. For instance, light can be reflected and refracted (see Figure 2.4). Further more, we can distinguish between different types of reflection and transmission called diffuse, specular and glossy reflection or transmission. For diffuse reflection or transmission, the BRDF or BTDF is constant, meaning radiance is equally distributed in all outgoing directions, specular reflection or transmission on the other hand, is scattered directionally over a narrow solid angle. Glossy reflection or transmission describes materials that are neither diffuse nor specular, it is still directional but more light is reflected on a restricted part of the hemisphere (see Figure 2.4).

Finally, the *bidirectional scattering distribution function* (BSDF), denoted by f_s, is the union of two BRDFs, one for each side of the surface, and two BTDFs describing the light transmission in each direction. With the BSDF its easier to define different kind of surfaces since it is defined over the whole sphere and we only have to deal with one function. In our context, BSDF functions are rarely needed because they are too complex for real-time applications.

2.3 The Light Transport Equation

This section presents the important *light transport equation* or *rendering equation*, which plays a major part in computer graphics, it describes the light transport in a scene. In general, the light transport equation can be described as follows. If we want to shade a surface location p we want to know the outgoing radiance at that location in view direction, which is equal to the sum of

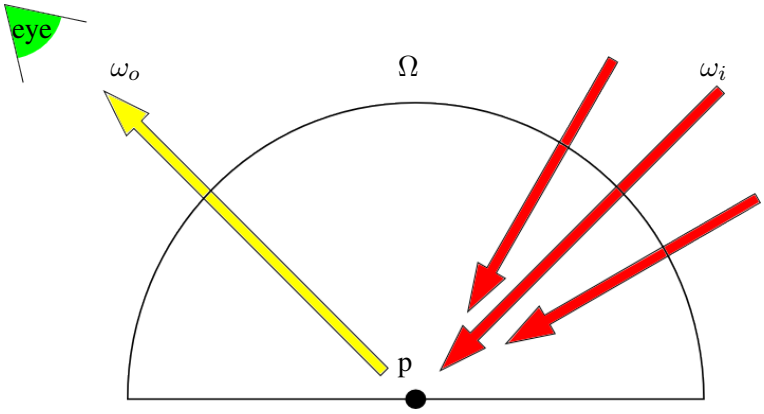


Figure 2.5: The light transport equation describes the total amount of radiance emitted from a point p along a particular viewing direction ω_o, given a incoming radiance and a BRDF

emitted, reflected and transmitted radiance (see Figure 2.5). In the following text we will use the BRDF to specify the surface properties, which excludes transmittance. Since we are interested in the extant part of radiance L_o, we can calculate the incident radiance L_i from

$$L_i(p, \omega) = L_o(Raycast(p, \omega), -\omega) \tag{2.5}$$

That means, incoming radiance arrives at location p coming from direction ω is equal to outgoing radiance L_o arriving from some other surface location p in the opposite direction $-\omega$. This other point is defined through a raycasting function, by shooting a ray from location p in direction ω returning the location of that point, which is visible from p. As we can see, the $L_o(Raycast(p, \omega), -\omega)$ states the fact that incoming radiance arriving at a point must be equal to outgoing radiance from some other point. Now we can express L_o as emitted radiance L_e plus reflected radiance L_r by using this formula:

$$L_o = L_e + L_{o,reflected} \tag{2.6}$$

$L_e(p, \omega)$ is a radiance function which represents all light emitted from a surface location p in direction ω in the scene and $L_{o,reflected}$ is defined through

$$L_{o,reflected}(p, \omega_o) = \int_\Omega f_r(p, \omega_i, \omega_o) L_i(p, \omega_i) cos\Theta d\omega_i \tag{2.7}$$

Ω is the hemisphere located above p. The term $f_r(p, \omega_i, \omega_o)$ is the BRDF, for incoming direction ω_i, and outgoing direction ω_o, $L_i(p, \omega_i)$ is the incoming radiance at the surface point p coming from direction ω_i. Furthermore Θ is the elevation angle between the incoming direction ω_i and the surface normal n at location p. If we put Equation 2.5, Equation 2.6 and Equation 2.7 together we can re-write the equation in its commonly used form:

$$L_o(p, \omega_o) = L_e(p, \omega_o) + \int_\Omega f_r(p, \omega_i, \omega_o) L_o(Raycast(p, \omega_i), -\omega_i) cos\Theta d\omega_i \tag{2.8}$$

which is the light transport equation. As we can see, the light transport equation is recursive since the outgoing light intensity function appears on both sides of the equation. This makes it hard to compute for real-time applications.

CHAPTER 3

State of the Art

3.1 Monte Carlo Integration

In Section 2.3, the light transport equation was introduced, which describes the outgoing light intensity. One way to solve the integral of the light transport equation is called Monte-Carlo integration, where random numbers are used to approximate the integral, which we will need when computing spherical harmonic coefficients (later on described in section 3.4). In this chapter, we will give a brief overview of Monte-Carlo Integration in the context of Computer Graphics. Readers interested in a more in-depth work should read: State of the art in monte carlo ray tracing for realistic image synthesis [13].

What we are looking for in *Monte-Carlo integration* is an estimate of the integral of a function. Basically, we take a large collection of samples of a function to estimate the integral, this estimate can be calculated by the Monte-Carlo estimator. It is defined by

$$\int f(x)d(x) \approx \frac{1}{N}\sum_{i=0}^{N} f(x_i)w(x_i), \tag{3.1}$$

where $\int f(x)d(x)$ is the integral, which we want to approximate, N is the number of samples we take, $f(x_i)$ represents one sample of the function, and $w(x_i)$ is a weighting function for each sample. The weighting function is $1/p(x)$, where $p(x)$ is the probability distribution of the samples. If we increase the number of samples, we can lower the approximation error.

Since we want to integrate over the surface of a sphere, we have to map the random numbers into spherical coordinates. To achieve an even distribution over the sphere, we take pairs of independent uniformly distributed random numbers ξ_x and ξ_y and map them into spherical coordinates by using this formula

$$(2\arccos(\sqrt{1-\xi_x}), 2\pi\xi_y) \rightarrow (\Theta, \phi). \tag{3.2}$$

If our points are evenly distributed, our weighting function $w(x_i)$ is a constant value and Equation 3.1 becomes:

$$\int f(x)d(x) \approx \frac{w}{N}\sum_{i=0}^{N} f(x_i). \tag{3.3}$$

Finally, when sampling uniformly over the unit sphere the probability $p(x)$ is always $\frac{1}{4\pi}$ since the integral of the probability density has to result in one and the area where we integrate over is 4π, which is the area of the unit sphere.

$$\int f(x)d(x) \approx \frac{4\pi}{N}\sum_{i=0}^{N} f(x_i) \tag{3.4}$$

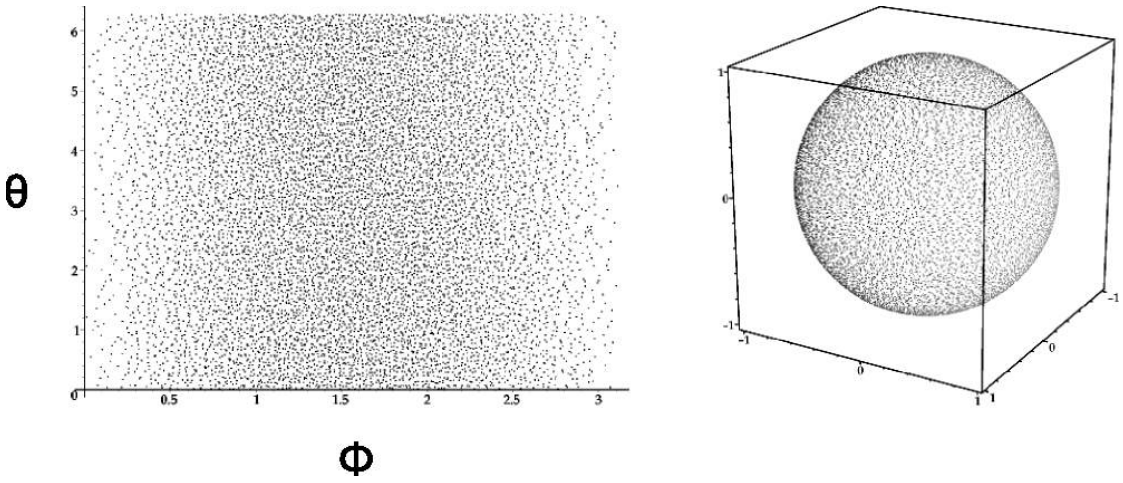


Figure 3.1: 10.000 samples generated using jittered stratification. The left image shows (Theta, Phi) angle space and on the right we can see the 3D projection. Image courtesy of [7].

To lower the variance from the sample distribution, we could generate a grid of jittered points (see Figure 3.1). This technique is called *Jittered Stratification*, where an input cell is divided into NxN cells and a random point is picked in each cell. The sum of variances of each cell will never exceed the variance of random samples over the whole sphere, and is often much lower.

3.2 Orthogonal Basis Functions

In this chapter, we will describe how a function $f(x)$ can be approximated by another function $B_n(x)$, which is a linear combination of basis functions. Orthogonal basis functions allow us to express a continuous function over a given domain as a linear combination of basis functions. These functions can be thought of as little pieces of information that when scaled and combined can produce an approximation of the original function $f(x)$. Since there are a lot of basis functions, which can be used, a good choice for basis functions are orthogonal polynomials.

They have a very important property, if we integrate the product of two orthogonal polynomials and they are different we get zero, if they are equal we get a constant value.

$$\int_{\Omega} B_m(x)B_n(x)dx = \begin{cases} 0 & \text{if } n \neq m \\ c_n & \text{if } n = m \end{cases} \tag{3.5}$$

Where Ω is the domain of the basis functions and the polynomials $B_k(x), k = 0, 1, 2, ...,$ are orthogonal. Examples for families of orthogonal polynomials are the *Legendre, Chebyshev, Laguerre, Jacobi and Hermite* polynomials [1, 30]. If $c_n = 1$ hold for all n, which is a stronger requirement, then the basis functions are also called orthonormal.

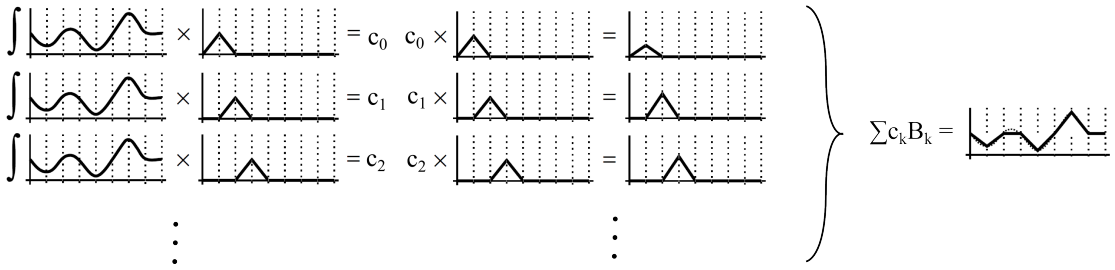


Figure 3.2: The left half of the image shows the projection of the polynomial into the first three coefficients. The right half of the image shows the reconstruction of the original function by the scaled coefficients. Image Image courtesy of [7].

An arbitrary function can be approximated by a linear combination of basis functions (see Figure 3.2):

$$f(x) \approx \sum_{i=0}^{N} \mathbf{c}_i b_i. \tag{3.6}$$

Where $\mathbf{c}$ is a vector of coefficients with one component $\mathbf{c}_i$ for each basis function. In the case of orthonormal basis functions, the coefficients $\mathbf{c}_i$ can be found by integrating the product:

$$\int_{\Omega} f(x)b_i(x)dx = \mathbf{c}_i. \tag{3.7}$$

For basis functions that are not orthonormal, finding the coefficients is more complicated, however we will only deal with orthonormal basis functions in this work. The process of finding the coefficients is called *projection* and the process of re-constructing a function with a linear combination of basis function is called *reconstruction*.

Another important operation is integrating the product of two arbitrary functions C and D. Using symbolic integration is not real-time friendly. The key is projection, where the functions C and D are projected into some orthonormal basis. We get the projected coefficients $\mathbf{c}_i$ and $\mathbf{d}_i$ and the orthonormality of the basis allows us to transform the integral into a simple dot product. This can be computed very fast.

$$\int \tilde{C}(x)\tilde{D}(x)dx = \sum_{i=0}^{N} \mathbf{c}_n \mathbf{d}_n \tag{3.8}$$

An interesting family of polynomials orthonormal in the interval -1 to 1 are the *Legendre Polynomials*, usually they are denoted by P_m and the *Associated Legendre Polynomials* P_l^m, which are generalizations of the *Legendre Polynomials* P_m. The Associated Legendre Polynomials return real numbers, in contrast to the Legendre Polynomials, which are defined over the field of the complex numbers. We will focus on the Associated Legendre Polynomials:

$$P_l^m(x) = \frac{(-1)^m}{2^l l!}\sqrt{(1-x^2)^m}\frac{d^{l+m}}{dx^{l+m}}(x^2-1)^l. \tag{3.9}$$

As we can see, the Associated Legendre Polynomials take two arguments m, l where $l \in \mathbb{N}_0$ is the *band index* and m takes any integer in the range of $[0, l]$. The band index is used to split the set of basis functions into bands, where inside a band the polynomials are orthogonal with respect to a constant. Between bands they are orthogonal with a different constant. Since Equation 3.9 is not computation friendly, we can also define a set of recurrence relations by

$$P_m^m(x) = (-1)^m(2m-1)!!(1-x^2)^{\frac{m}{2}} \tag{3.10}$$

$$P_{m+1}^m(x) = x(2m+1)P_m^m(x) \tag{3.11}$$

$$(l-m)P_l^m(x) = x(2l-1)P_{l-1}^m(x) - (l+m-1)P_{l-2}^m(x). \tag{3.12}$$

To evaluate P_l^m, we start the recursion by the term 3.10 since it needs no previous values, making our start condition $P_0^0 = 1$ and generating the P_m^m with highest possible m. Afterwards, if $l = m$ the final result has been computed or we have to raise a band for all remaining cases where $l < m$. For this purpose, Equation 3.11 is used to calculate the next band once until condition $l = m + 1$ is meet. Finally Equation 3.12 is iterated until the answer is found depending on two previous bands $l - 1$ and $l - 2$.

3.3 Spherical Coordinates

Because we will deal with spherical functions, it is useful to work in spherical coordinates rather than Cartesian coordinates. Normally, the spherical coordinate system is defined by two angles Θ and ϕ. Θ represents the elevation angle and is constrained to $0 \leq \Theta \leq \pi$ and ϕ, the rotational angle, is constrained to $0 \leq \phi \leq 2\pi$. If we want to convert between coordinate systems the following relations are used:

$$r = \sqrt{x^2 + y^2 + z^2} \tag{3.13}$$

$$\phi = tan^{-1}(\frac{y}{x}) \tag{3.14}$$

$$\Theta = cos^{-1}(\frac{z}{r}). \tag{3.15}$$

We can use these formulas to obtain the spherical coordinates (r, Θ, ϕ) of a point from Cartesian coordinates (x, y, z) and vice versa we can use these formulas:

$$x = r \sin\Theta cos\phi \tag{3.16}$$

$$y = r \sin\Theta sin\phi \tag{3.17}$$

$$z = r \cos\Theta. \tag{3.18}$$

3.4 Spherical Harmonics

Spherical harmonics lighting is a technique to render realistic looking images in real-time and was introduced by Sloan, Kautz and Snyder at Siggraph 2002. In this chapter, we will explain the definition of the spherical harmonics and some very important properties. Further on, we will describe the rotation of spherical harmonic functions including zonal harmonics, which is a restricted class of spherical harmonic functions. In this document, we are covering *Real Spherical Harmonics* only, since in computer graphics real-valued functions are mainly used.

3.4.1 Definition

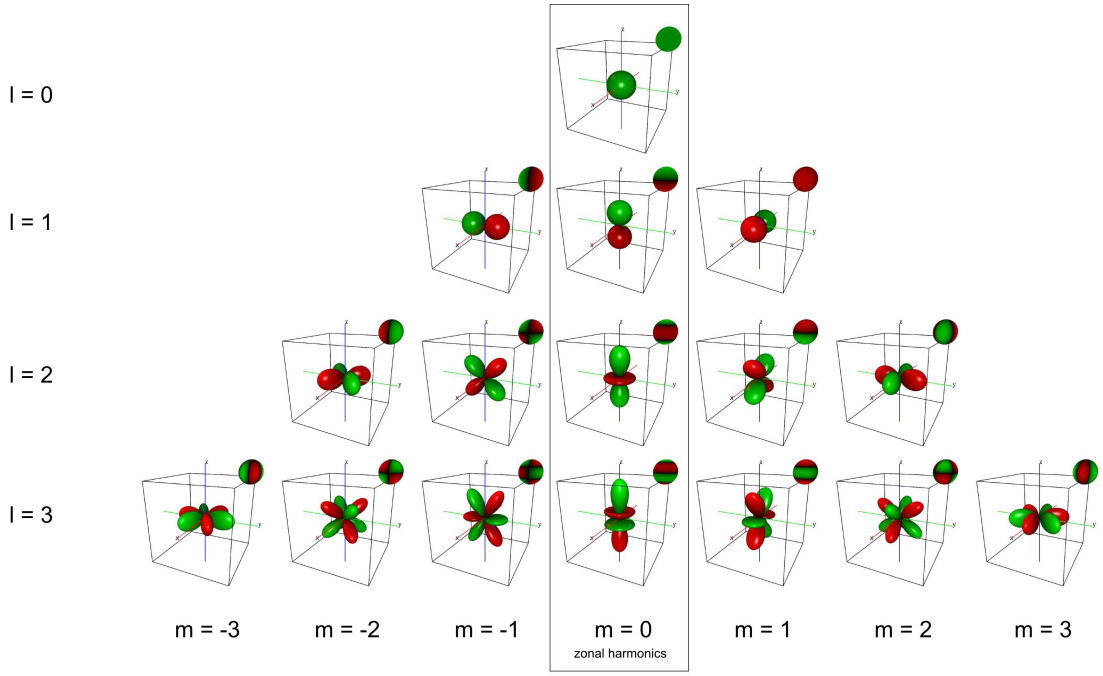


Figure 3.3: This figure illustrates the first 4 spherical harmonic bands l = 0...3. Green indicates positive extents and red negative extents. Image Image courtesy of [23]

Usually the spherical harmonic function is denoted by y. To parametrize a point on the unit

sphere, we will use spherical coordinates as explained in subsection 3.3.

$$y_l^m(\Theta,\phi) = \begin{cases} \sqrt{2}N_l^m cos(m\phi)P_l^m(cos\Theta) & \text{if } m > 0 \\ N_l^0 P_l^0(cos\Theta) & \text{if } m = 0 \\ \sqrt{2}N_l^m sin(-m\phi)P_l^{-m}(cos\Theta) & \text{if } m < 0 \end{cases} \tag{3.19}$$

$$N_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}} \tag{3.20}$$

Where N is a scaling factor to normalize the functions and P the Associated Legendre Polynomials explained in Section 3.2. Spherical harmonic functions are orthonormal and are basically a normalized adaption of the Legendre Polynomials mapped onto the unit sphere. For convenience, it is useful to distinguish three types of spherical harmonic functions: The set of spherical harmonics, which are latitudinally partitioned along the unit sphere and $m = 0$ are called *zonal harmonics*. The second class is called *sectoral harmonics*, which are of the form $y_{|m|}^m$ and are divided along the meridian. All other sets of spherical harmonics are called *tesseral harmonics* and are partitioned in longtitude and latitude. (see Figure 3.3).

When looking at Equation 3.19, we see that the function depends on two parameters m, l. For convenience, it is sometimes useful to flatten the two parameters m and l into one parameter i when generating the spherical harmonic functions. The ordered sequence is defined by

$$y_l^m(\Theta,\phi) = y_i(\Theta,\phi), \tag{3.21}$$

where the index $i = l(l+1) + m$. To speed up the calculation when generating the basis functions, the factorial calculation can be precalculated into a table.

3.4.2 Properties

In the next four sections, we will explain the most important properties and operations of spherical harmonics functions, *projection*, *convolution*, *products* and *rotational invariance*.

Projection and Reconstruction

With a given function, it is simple and straight forward to project the function into coefficients, as we have seen in Section 3.2. If we replace the arbitrary polynomial basis b_n of Equation 3.7 by the real spherical harmonics basis function $\mathbf{y}_i(s)$. We can then integrate the product of the function $f(s)$ and the SH function $y_i(s)$ to calculate the spherical harmonic coefficients.

$$\int_S f(s)y_i(s)ds = \mathbf{c}_i \tag{3.22}$$

$f(s)$ could be the incident light at a point in the scene, then the coefficient vector $\mathbf{c}_i$ provides a compact representation of this incident light. As we have already introduced the *Monte Carlo Estimator* 3.1, we are able to calculate a numerical solution for Equation 3.22 and it can be re-written as:

$$\frac{4\pi}{N}\sum_{k=1}^{N} f(x_k)y_i(x_k) = \mathbf{c}_i. \tag{3.23}$$

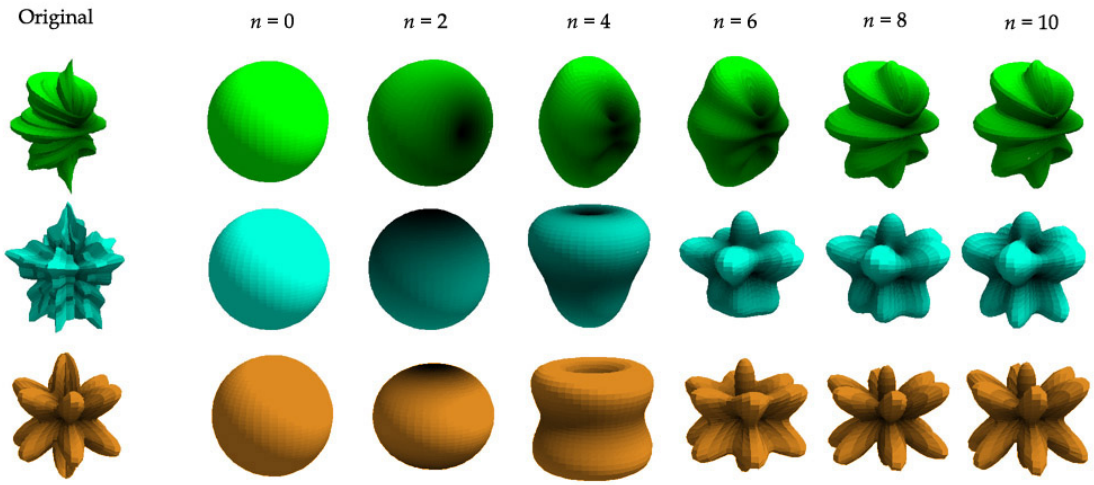


Figure 3.4: Projection of spherical functions with increasing order. Image courtesy of Green [7]

The result will be a band-limited approximation of the original function because the basis functions itself are band-limited, where band-limiting means to limit a deterministic signal to zero above a certain finite frequency. The quality of the approximation depends on the number of basis functions used, where $(n+1)^2$ coefficients are used for order-n harmonics. Using order-3 spherical harmonics will generate sixteen coefficients, which will in our case be sufficient to produce acceptable results (see Figure 3.4).

Convolution

It would be nice to apply filters to spherical functions as we know it from the Fourier theory, where convolution is a central concept. The main idea is that by using the spherical harmonics projection of a function f and a kernel function k instead of the original functions, the convolution can be computed more efficiently. A precondition is that the kernel function has to be circular symmetric since a non-symmetric convolution is not defined on the sphere. The spherical harmonic convolution can be defined in frequency domain by

$$(\mathbf{k} * \mathbf{c})_l^m = \sqrt{\frac{4\pi}{2l+1}} \mathbf{k}_l^0 \mathbf{c}_l^m, \tag{3.24}$$

where $\mathbf{k}$ is the coefficient vector of the kernel function and $\mathbf{c}$ the coefficient vector of f. $(\mathbf{k}*\mathbf{c})$ denotes the coefficient vector of the convolved function $(k*f)$. As we can see from Equation 3.24, we scale each band of c with each band of k where $m = 0$.

Products

This property is very important for calculating visibility and/or lighting. For interested readers, Green gives a detailed explanation in his paper [7]. For example, it would be nice express the

product of the visibility function and the incident light at some point in the spherical harmonics basis. This would reduce the triple product integral of light, BRDF and visibility often encountered in Global Illumination methods to the product of incident light and visibility followed by a simple dot product with the BRDF (recall from Section 3.2 that the product integral of two functions can be approximated by a dot product in any orthonormal basis). The goal is to compute an order-n coefficient vector $\mathbf{c} \bullet \mathbf{d}$ by projecting the product of two reconstructed functions $\tilde{f}$ and $\tilde{d}$ back to spherical harmonics.

$$(\mathbf{c} \bullet \mathbf{d})_i = \int_S \tilde{f}(s)\tilde{d}(s)y_i(s)ds \tag{3.25}$$

$$= \int_S (\sum_{j=0}^{N} c_j y_j(s))(\sum_{k=0}^{N} d_k y_k(s))y_i(s)ds \tag{3.26}$$

$$= \sum_{kj} c_j d_k \int_S y_j(s)y_k(s)y_i(s)ds \tag{3.27}$$

$$= \sum_{kj} c_j d_k \Gamma_{jki} \tag{3.28}$$

Where Γ_{jki} is called the *Triple Product Tensor* [26], a sparse, symmetric order-3 tensor. Since there is a lot of room for optimization when working with the sparse tensor, Snyder [28] wrote an article on optimizing low-order spherical harmonic products, where he explains in detail how to write a code generator. If one of the two functions, say f, is known in advance, it is possible to precompute a *transfer matrix* for that function

$$(\mathbf{M})_{ij} = \int_S f(s)y_i(s)y_j(s)ds \tag{3.29}$$

This matrix transforms a coefficient vector $\mathbf{d}$ of a second function d to the coefficient vector of the product of both functions

$$(\mathbf{c} \bullet \mathbf{d})_i = \sum_{j=0}^{N} \mathbf{M}_{ij} d_i. \tag{3.30}$$

For example, it is possible to map the projection of a light source into the projection of a shadowed light source through a transfer matrix if the visibility function is known in advance. The light function need not be known in advance and can change in each frame for the cost of a matrix product.

Rotational Invariance

A very interesting property of spherical harmonic functions is that they have *rotational invariance*, which describes the fact that if we take a copy c of an arbitrary rotated function d the following relation can be defined:

$$\tilde{c}(x) = \tilde{d}(R(x)). \tag{3.31}$$

Where R is a arbitrary rotation. This relation implies that projecting a rotated function c has the same result as projecting the original unrotated function d and rotating the input. This property is very important when we rotate or move lights, because it prevents the light source from causing artifacts like fluctuations in the computed shading.

3.4.3 Rotation

Since there are many ways to implement efficient spherical harmonic rotations [11] [17], we will give a short overview of spherical harmonic rotations based on Blanco's [4] paper and on Green's [7] supplementary notes. From the last section, we know that for each coefficient **c** that describes a function *f* there is a second coefficient vector **d** that exactly describes the function *f* rotated on the sphere. It would be nice to have a matrix that transforms **c** into **d**, like we know from Euler's rotation.

Since spherical harmonics are orthogonal, it is possible to find a linear transformation on the SH coefficients that transforms the coefficients of the unrotated function to the coefficients of the rotated function. Furthermore, because of the orthogonality property, we get a block-diagonal sparse matrix when compositing a SH rotation matrix (see Equation 3.32).

$$M = \left(\begin{array}{c|ccc|ccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ \hline 0 & X & X & X & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & X & X & X & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & X & X & X & 0 & 0 & 0 & 0 & 0 & \cdots \\ \hline 0 & 0 & 0 & 0 & X & X & X & X & X & \cdots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \cdots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \cdots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \cdots \\ 0 & 0 & 0 & 0 & X & X & X & X & X & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right) \tag{3.32}$$

It turns out that constructing a rotation matrix efficiently is not simple. However, for low order spherical harmonics it is still feasible to use symbolic integration on the product of a rotated spherical harmonic function with its unrotated copy.

$$M_{ij} = \int_S y_i(Rs) y_j(s) ds \tag{3.33}$$

In order to compose the rotation R it is useful to find the minimal number for rotations. An elegant approach is to use a ZYZ rotation, with its angular components (α, β, γ), which can be decomposed into a rotation of 90° about the x-axis, a rotation around the z-axis by β and a final rotation of -90° about the x-axis. The x-axis rotation is fixed in terms of the angle and therefore the matrix components of M can be precomputed (see Equation 3.34 for a 90°-rotation

example).

$$M = \left(\begin{array}{c|ccc|ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{\sqrt{3}}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{\sqrt{3}}{2} & 0 & \frac{1}{2} \end{array}\right) \tag{3.34}$$

The matrix for an arbitrary rotation can be calculated by

$$M(\alpha, \beta, \gamma) = M_\gamma M_{-90} M_\beta M_{90} M_\alpha \tag{3.35}$$

The next topic will cover a subset of spherical harmonic functions called the zonal harmonics, which can be rotated more efficiently.

3.4.4 Zonal Harmonics

Zonal Harmonics are the class of spherical harmonics basis functions, which are rotationally symmetric around the z-axis, and were first used in the context of computer graphics by Sloan [26] (see Figure 3.3). The zonal harmonics are denoted by y_l^0. Since we have only one non-zero entry per band a zonal harmonic function with the order n only has n coefficients. Zonal harmonics are very important when modelling light sources and are used later on in this document. Compared to the general spherical harmonic rotation, zonal harmonics are easier to rotate, because we only need to evaluate the spherical harmonic basis functions in the new direction v.

$$c_l = \sum_l z_l \sqrt{\frac{4\pi}{2l+1}} \sum_m y_l^m(d) y_l^m(v) \tag{3.36}$$

and c_l are the coefficients of the rotated function. z_l are the zonal harmonics coefficients of the unrotated function that is symmetric about the z-axis, d is the direction of the z-axis and v is the direction being rotated to.

3.5 Ambient Occlusion

Ambient occlusion is a soft shadowing technique, which adds more realism to a scene. Generally speaking, the ambient occlusion term describes how much of the hemisphere of a surface point is occluded. We want to give a short description on two methods, which extend previous papers on ambient occlusion by calculating the ambient occlusion term at run-time on the GPU. The method developed by Bunnell [5] can be used to compute the diffuse light transfer between surfaces in addition to the occlusion term for animated objects and scenes. The second method, presented here, was introduced by Mittring [18] and is called Screen Space Ambient Occlusion. The ambient occlusion is calculated in screen space.

Figure 3.5: The images from left to right show: A scene using environment lighting only, soft shadows using ambient occlusion and indirect lighting. Image courtesy of Bunnell [5]

Dynamic Ambient Occlusion and Indirect Lighting

Bunnell et al. [5] approximates a polygon mesh by surface elements. A surface element is an oriented disc, which has a position, normal and area. Each surface element can cast, reflect and transmit light. Surface elements are placed at each vertex and the area of the disc is calculated by adding one third of the area of each triangle shared by the vertex.

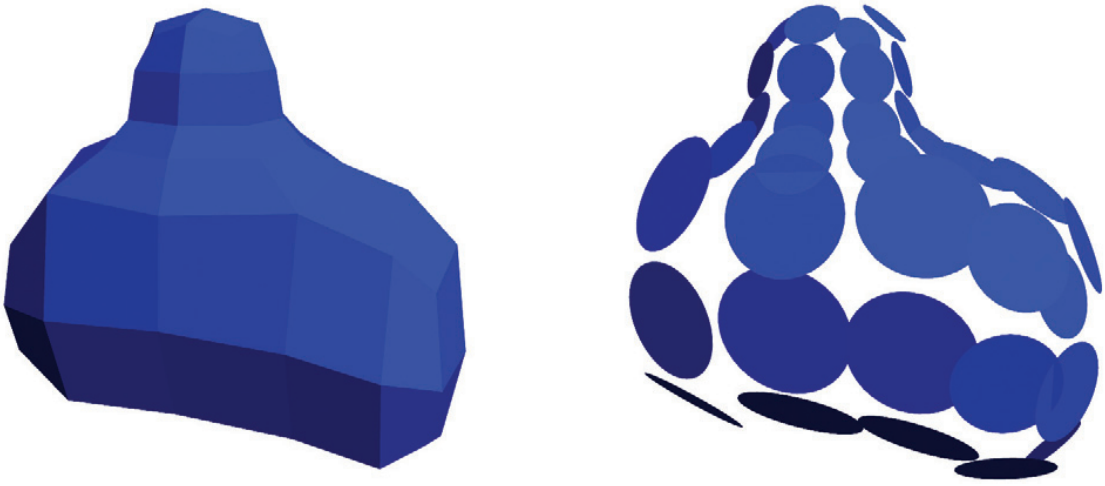


Figure 3.6: The image on left shows a polygon mesh, the image on the right shows the same mesh represented by surface disks. Image courtesy of Bunnell [5]

For calculating the ambient occlusion, we have to find out how much a receiver point is occluded by all other surface elements. This can be done based on the solid angle 3.7 between an emitter disc and receiver point. This determines how much the receiver point is occluded by the surface element.

$$O(r,e) = 1 - \frac{d_{re} cos\Theta_e max(1, 4cos(\Theta_r))}{\sqrt{\frac{A_e}{\pi} + d_{re}^2}} \tag{3.37}$$

Where Θ_e is the angle between the normal of the surface element and the vector from the center of the surface element to the receiver point on the receiver disc. Θ_r is the angle between the receiver normal and the vector from the receiver point to the center of the surface element. The term $max(1, 4cos(\Theta_r))$ is used to restrict occluding surface elements to the upper hemisphere. d_{er} is the distance from the receiver point to the center of the surface element and A_e is the area of the disc.

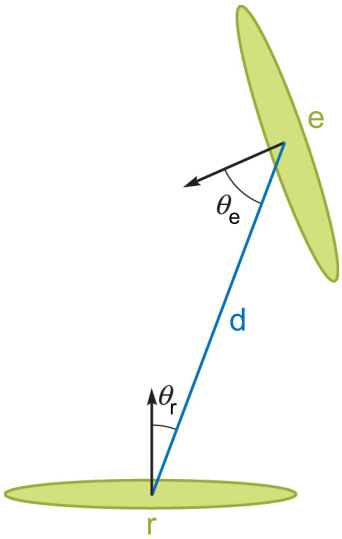


Figure 3.7: This image depicts a receiver disc and an surface element at distance d_{er}. Image courtesy of Bunnell [5]

The algorithm executes two main passes to calculate the ambient occlusion values. In the first pass, the ambient occlusion term of each disc in the upper hemisphere above a receiver point are summed up. After the first pass some surface areas are going to be too dark, since the overlap between surface elements is not accounted for. This problem is corrected by running a second pass with the same procedure, but now reducing the occlusion caused by each disc by a factor proportional to the occlusion of the disc itself. That results in shadows, which may be too light in some areas. To get a good estimate of the ambient occlusion values, a weighted average based on pass one and pass two is computed.

Indirect lighting can be added in an additional render pass. Bunnell [5] uses a slight modification of the ambient occlusion term mentioned before to calculate a single diffuse bounce of

indirect lighting. For this purpose, the solid angle function is replaced by a radiance transfer function calculated from disk to disk.

Additionally the disc is two-sided, the front face is used to emit and reflect light and the back face is used to transmit light and to cast shadows. All information like normal, position and area are stored in a texture map, which can easily be accessed by the GPU. To dynamically change geometry during animation, Bunnell [5] suggests using a fragment program to compute the animation and transformation of the surface elements.

Screen Space Ambient Occlusion

Figure 3.8: This image shows the effect of ambient occlusion on a video game scene. Image courtesy of Mittring [18]

Mittring et al. [18] introduced an improved Screen Space Ambient Occlusion technique (SSAO), which is currently used by state-of-the-art game engines. It is a method, where the ambient occlusion value is calculated for each pixel of a framebuffer. No pre-computation is needed and the method is therefore adequate for rendering dynamic scenes. Furthermore, the performance is independent of the scene complexity and only depends on the framebuffer resolution.

This technique uses a fragment program on the depth buffer. For each Pixel of the depth buffer nearby points are sampled. Afterwards those points are projected to screen space. The

depth of the sample is then compared against the depth of the pixel to identify if the surface at a sample occludes the surface at the pixel or not. The occlusion value for each pixel is computed by averaging the distances of the occluding samples. To speed things up, the authors developed a fast importance sampling to method to optimize sample placement.

3.6 Precomputed Radiance Transfer

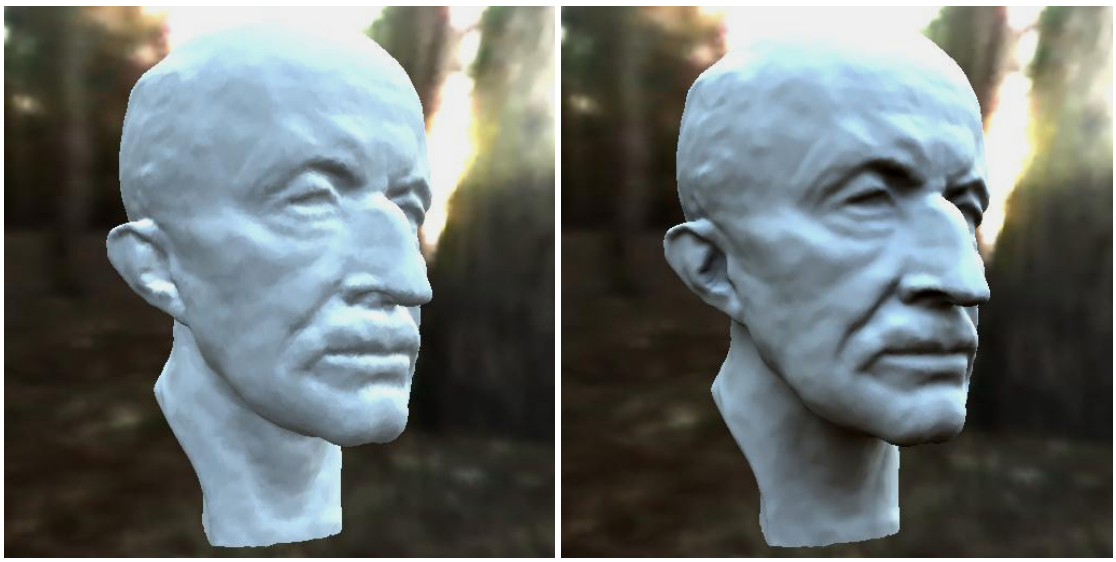


Figure 3.9: The image on the left depicts a head rendered with environment lighting only, the right image shows the head with self-shadows and interrefelctions using precomputed radiance transfer. Image courtesy of Sloan et al [27]

Precomputed Radiance Transfer (PRT) is a method to approximate global illumination effects. In PRT, the linear transformation (or transfer function) between radiance from a lighting environment and the incident radiance on a surface lit by this environment after a few indirect bounces is precomputed. Sloan et al. [27] use Spherical Harmonics (SH - see Section 3.4) to encode the huge amount of data resulting from the pre-computation. The basic idea is to approximate the incident radiance and the precomputed transfer function in the SH basis. During runtime the generated SH coefficients are used to approximate the render equation under changing environment lighting. With this method it is possible to calculate soft shadows and interreflections on diffuse and glossy objects in static scenes.

If we assume an infinitely distant light source like an environment map, we can rew-rite the incident light Equation 2.5 for an unshadowed surface point:

$$L_{i,unshadowed}(p, \omega_i) = L_{env}(p, \omega_i). \tag{3.38}$$

For the shadowed case, we add a visibility function V, which evaluates to 1, if we hit the light source and 0 otherwise.

$$L_{i,shadowed}(p, \omega_i) = L_{env}(p, \omega_i)V(p, \omega_i) \tag{3.39}$$

If we include indirect lighting, we replace V by the function $R(p, \omega_i)$ describing how much light from L_{env} reaches p through reflection.

$$L_{i,indirect}(p, \omega_i) = L_{env}(p, \omega_i)R(p, \omega_i) \tag{3.40}$$

If only diffuse objects are rendered then the light transport equation is simplified since light is reflected equally in all directions. In this case, the BRDF (see Equation 2.1) degenerates into a constant factor ρ with values ranging from 0 to 1 and we can re-write the light transport equation (see Equation 2.8) as

$$L_{o,*}(p, \omega_o) = \frac{\rho_{diffuse}}{\pi} \int_{\Omega} L_{i,*}(p, \omega_i) \cos(\theta) d\omega_i, \tag{3.41}$$

where $L_{i,*}$ is one of the three terms described in Equations 3.38 to 3.40. Notice how we can pull out the constant BRDF $\frac{\rho_{diffuse}(p)}{\pi}$ (see Equation 2.7) and integrate only the cosine term. The emittance term has been removed, because we assume the environment light is the only light source.

A key observation is that the function V and R only depend on the geometry of the scene. Therefore, in static scenes and for perfectly diffuse materials the product of the visibility function and the cosine term can be pre-computed, which results in a *transfer function*. At run-time, we now have two functions, the *lighting function* and the *transfer function*. To compute Equation 3.41, we need to quickly integrate the product of those two functions. For this purpose, we represent the lighting function and the transfer function in the spherical harmonic basis (see Section 3.4). The lighting function can be approximated by

$$L_{env} \approx \sum_k l_k \mathbf{y}_k \tag{3.42}$$

The spherical harmonics projection of the functions V and R are more involved (see [27] for details), but since they only depend on the scene geometry, they can be precomputed if the geometry is assumed to be static. The outgoing radiance $L_{o,*}$ can be computed very quickly because the integral of the product of two functions can be expressed by a simple dot product in the spherical harmonics basis (see Equation 3.8).

In summary, Sloan [27] demonstrated a technique which can compute dynamic lighting and viewing changes in static scenes.

3.7 Real-time Rendering of Dynamic Scenes under All-frequency Lighting using Integral Spherical Gaussian

Iwasaki et al. [12] introduced a new method for rendering dynamic scenes under all-frequency environment lighting in real-time.

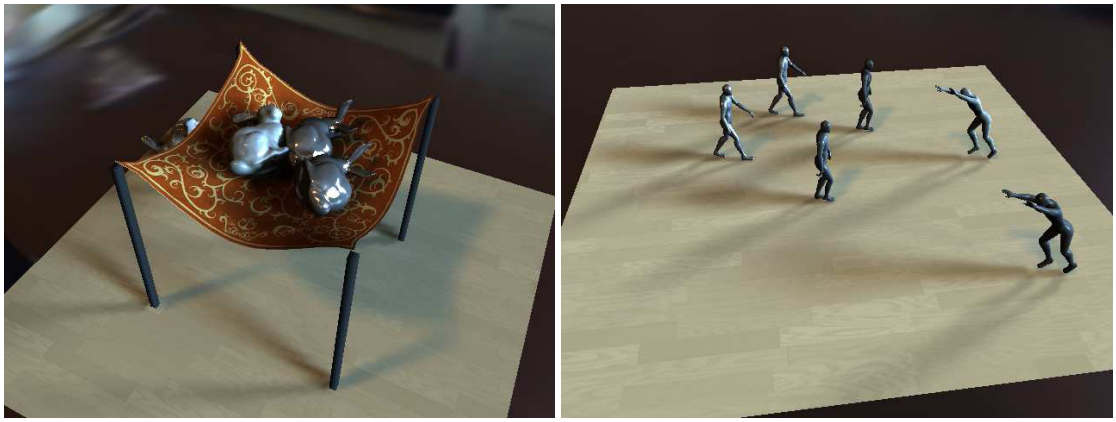


Figure 3.10: The two images show scenes which are completely dynamic and are rendered with all-frequency lighting. Image courtesy of Iwasaki [12]

To calculate outgoing radiance at a surface point the triple product integral of incident lighting, visibility and BRDF is computed. The authors approximate all three terms of the triple product integral by sums of *Spherical Gaussians* (see Section four in their paper [12]). A Spherical Gaussian is a type of spherical radial basis function suited for approximating low-frequency functions on the sphere. Environment lighting is approximated by the sum of Spherical Gaussians and diffuse BRDFs (including the cosine term) are represented by a single Spherical Gaussian. Glossy BRDFs need multiple Spherical Gaussians for a sufficiently accurate approximation. The visibility function for a receiver point is represented in the Spherical Gaussian basis as well.

To reduce the effort of calculating the visibility for a receiver point, the authors approximate all geometry by a set of spheres. Hence, it is easy to calculate the solid angles of an occluded region and represent it by a sum of Spherical Gaussians. To efficiently compute the product integral of the visbility function and the other terms of the triple product integral the authors introduce the *Integral Spherical Gaussians*. Integral Spherical Gaussians are an extension of the integral image defined over the 2D image domain to the Spherical Gaussians defined on the unit sphere. For a complete definition of the Integral Spherical Gaussians, see section four of the paper [12].

To calculate the integral of the Spherical Gaussians over the occluded regions, the hemisphere over a receiver point is discretized into small patches (see image (a) in Figure 3.11). Within an area limited by the angle γ the hemisphere is subdivided uniformly. Occluded areas on the hemisphere are determined by patches overlapping the projection of the blocking spheres (see image (b,d) in Figure 3.11). Finally the Integral Spherical Gaussian of the occluded regions are calculated representing the integral of the product of the visibility function, the incident lighting and the BRDF.

The authors demonstrated rendering dynamic scenes with changeable viewpoints and BRDFs under dynamic all-frequency lighting in real-time. Furthermore, they showed how to efficiently

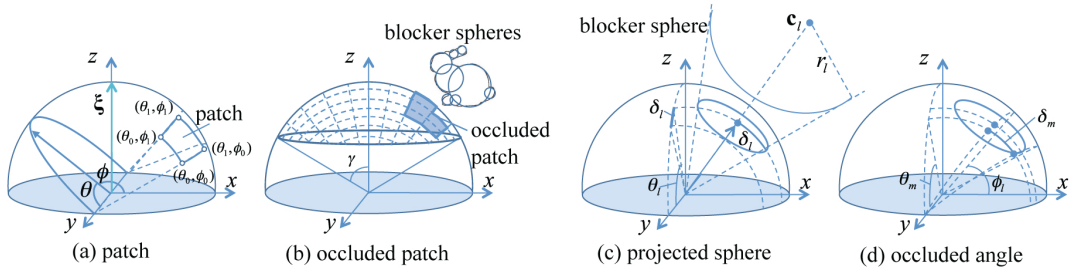


Figure 3.11: The four images show coordinate systems for *integral spherical Gaussian*. Image courtesy of Iwasaki [12]

calculate the triple product integral in the Spherical Gaussian basis using Integral Spherical Gaussians. However, due to the approximation of blocking geometry with spheres, their method is limited to relatively simple or sparse geometry. Approximating a complete height field accurately with spheres would be prohibitively expensive.

3.8 Cascaded Light Propagation Volumes for Real Time Indirect Illumination

Figure 3.12: The images show a scene rendered with indirect illumination in real-time. The complete scene, including lights, camera and geometry are fully dynamic. Image courtesy of A. Kaplanyan & C. Dachsbacherr [15]

Anton Kaplanyan and Carsten Dachsbacher [15] introduced a new technique to approximate direct and indirect illumination for fully dynamic scenes. Their method does not need any precomputation and can handle single and multiple bounce indirect illumination.

Radiance in a scene is sampled on a 3D-lattice, called *light propagation volume* (LPV). Each cell of the lattice stores the directional radiance distribution at the center of the cell represented in a spherical harmonic basis. Radiance is iteratively propagated from cell to cell until the radiance has travelled through the entire light propagation volume.

The method proceeds in four major steps. In the first step, the light propagation volume is initialized with the radiance distribution from a set of virtual point lights [16] representing direct and indirect light sources. In the second step a coarse approximation of the scene geometry is

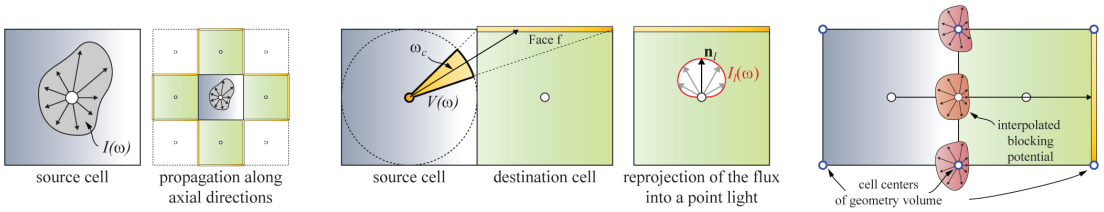


Figure 3.13: Left: The approximated radiance is propagated to adjacent cells. Center: The flux from the source cell through the destinations faces is calculated. Right: Blocking potentials are interpolated between cell centers of the geometry volume. Image courtesy of A. Kaplanyan & C. Dachsbacher [15]

computed which is needed for blocking information in the following step. Then radiance is propagated through the volume. During the final step the surface lighting is reconstructed using the final radiance distribution in the cells.

Initialization of the Light Propagation Volume

When initializing the LPV, direct and indirect light is converted into a set of virtual point lights (VPL), based on the work of Keller et al. [16]. VPLs representing the direct and indirect lighting contribution of each light source are created. Afterwards, the contribution of each VPL is used to compute the initial radiance distribution in each cell of the light propagation volume.

Scene Geometry Approximation

This step computes an approximation of the scene geometry for occlusion computation. This, done by sampling the scene surfaces and representing them as surfels (surface element). The authors model occlusions in the spirit of Sillion [25]. An accumulated blocking potential of the surfels is computed for each cell. This blocking potential tells us how much of the radiance passing through a cell is blocked in a particular direction. It is stored at the corners of each cell and is reconstructed to a spherical function that blocks the radiance flow between cells in the propagation step.

Iterative Light Propagation

This step computes the radiance distribution for the light propagation volume by iterating small local propagation steps until the light has travelled through the entire volume. In one iteration light can only travel to neighboring cells. Radiance is propagated from a source cell to its destination cells in six axial directions (see leftmost image in Figure 3.13). The amount of radiance transferred is determined by integrating the source intensity over the solid angle of the shared face f giving the flux across face f of the destination cell (see image two from the left of Figure 3.13). The radiance distribution at the center of the destination cell is updated with the

transferred radiance. This process is computed for all faces of a destination cell and the resulting SH coefficients are accumulated for the next iteration.

Rendering

Finally, incident radiance at a surface point is computed by trilinearly interpolating between the SH coefficients of neighboring cells. Outgoing radiance is computed by a product integral of the BRDF function and the incident radiance, approximated by a dot product in the spherical harmonics basis.

3.9 Fast Soft Self-Shadowing on Dynamic Height Fields

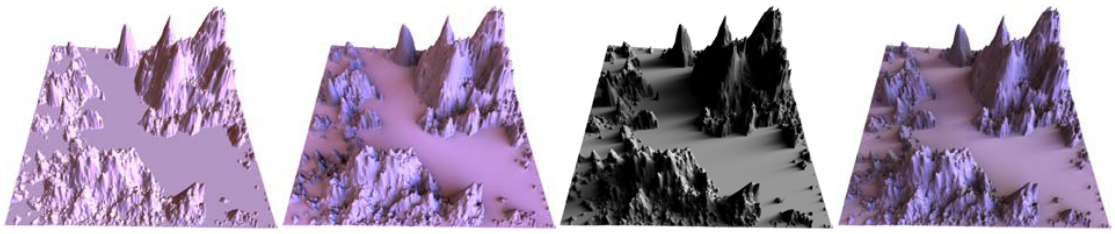


Figure 3.14: The images show different lighting and shadowing settings. The leftmost image shows a height field without shadows. The second image shows shadows from an environment light source. The third image shows a height field shadowed from a key-light. On the last image key light and environment light are combined giving a more natural look. Image courtesy of J. Snyder & D. Nowrouzezahrai [29]

John Snyder and Derek Nowrouzezahrai [29] present a method for calculating realistic soft shadows from large area light sources on dynamic height field data. This technique is the basis for our method.

To calculate soft self-shadowing on a height-field lit by an environment light, we have to find the directions in which the environment is visible from each receiver point. The visibility function is constructed by finding the maximum elevation angle in each azimuthal direction, i.e. the horizon at the receiver point. To reduce sampling overhead a multi-resolution pyramid of the height field is computed. The maximum blocking angle (horizon angle) in each azimuthal direction is found by sampling from finer to coarser levels with increasing distance from the receiver point. A continuous horizon function for a given receiver point can be reconstructed by finding the horizon angle in discrete, uniformly-spaced set of azimuthal directions and interpolating linearly in between. A visibility functions with values of 1 above the horizon and 0 below is projected to the spherical harmonics basis to get the total spherical harmonic visibility vector at a receiver point.

The method support two types of light sources. Key lights, which are similar to directional light sources (e.g. sunlight) and environmental lights (e.g sky). To calculate visibility for a given

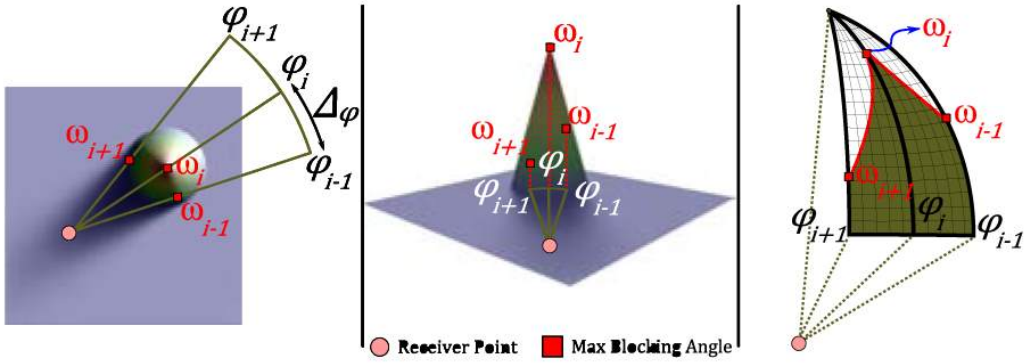


Figure 3.15: The images show a visibility wedge determined by adjacent blocking angles (ω_i, ω_{i+1}) and their corresponding azimuthal direction $(\varphi_i, \varphi_{i+1})$. Image courtesy of J. Snyder & D. Nowrouzezahrai [29]

key light, we only need to sample in a narrow range of azimuthal angles compared to environmental lights, where we have to sample the complete azimuthal extent ranging from 0 to 360 degrees. To obtain better soft shadows the authors compute finer transitions between neighboring pyramid levels resulting in extra levels between power-of-two reductions, also helping to decrease aliasing artefacts.

Finally, shading a point on the height field is done by computing the SH triple product integral of environment light, visibility and the BRDF. Since only diffuse surfaces are handled, the BRDF reduces to the clamped cosine function on the hemisphere. The triple product integral can be approximated by the dot product of environment light SH coefficient vector and the transfer vector. The transfer vector is the spherical harmonic product between the total visibility vector and the clamped cosine over the hemisphere.

In practice, the projection of the visibility function is the bottleneck of the method. To allow for real-time performance, the authors precompute a table of visibility function *wedges* between pairs of adjacent azimuthal directions. These wedges are only parametrized by the two adjacent horizon angles (Figure 3.15) and the coefficients of their SH projection can be precomputed and stored efficiently for all combinations of horizon angle pairs. At run-time the authors do a table lookup using the pairs of adjacent horizon angles to retrieve the SH coefficient vector followed by a fast z-rotation in the spherical harmonic basis to a given azimuthal direction. This can be done for all pairs of adjacent horizon angles and the sum of all rotated coefficient vectors is the projected visibility function.

In summary, we can say that with this method it is possible to render soft-shadows from key-lights and environment-lights. We will extend this method to render soft shadows from local lights, as explained in detail later in the text.

3.10 Scalable Height Field Self-Shadowing

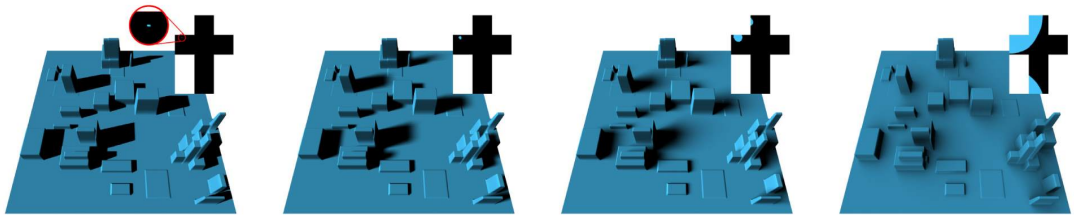


Figure 3.16: The images show, from left to right, a height-field lighted by environment-lights with increasing size. We can see sharper shadows for small environment-lights and softer shadows for large environment. Image courtesy of Timonen et al. [31]

Timonen [31] introduced a different approach to calculate soft-shadows on dynamic height fields. A main advantage compared to the method of Synder [29] is that they are able to render shadows with sharper edges, i.e. full-frequency visibility while still maintaining a good performance level.

Snyder et. al. [29] sample along discrete azimuthal directions for each receiver point on the height field. This is a redundant sampling strategy because the same positions on the height field can get sampled multiple times for different receiver points. Instead of processing each point on the hight field independently, Timonen et. al. [31] traverse the hight-field in parallel lines (see the leftmost image in Figure 3.17). Along each line one hight sample after another is added to the hight function to derive the horizon angle. The authors construct a convex hull iteratively, which represents the possible candidates for the horizon (points not on the convex hull can not be part of the horizon of any receiver point). The horizon angle at each height field sample is then determined by its direct neighbours in the convex hull (as illustrated in the middle image of Figure 3.17).

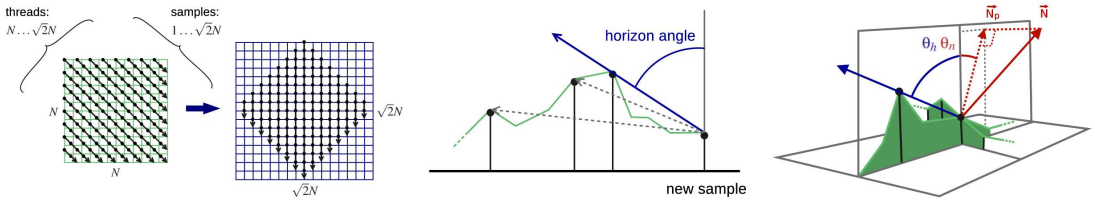


Figure 3.17: The left image shows, how many sampling directions are created and how many samples are processes in each directions for a given hight field resolution. The middle images depicts that for a new sample the horizon angles is calculated by the previous occluders in the convex hull. The rightmost image shows how lighting can be tabulated as a function of the horizon angle Θ_h and the angle Θ_n of the projected normal $\vec{N}_p$. Image courtesy of Timonen [31]

Arbitrary low-frequency and high-frequency environment-light sources can be handled by the method, allowing it to capture more shadow details than to the method of Snyder [29]. For a given sample point in direction d the sample point normal is projected onto the azimuthal plane and radiance incident from one azimuthal direction can be precomputed as a function of the maximum blocking angle θ_h and the angle θ_n for the projected normal (as illustrated in the rightmost image of Figure 3.17). The precomputed incident radiance can be stored in table in the form of SH coefficient vectors and looked up at run-time with the given blocking angles. Finally the complete incident radiance from all directions is approximated by accumulating the pre-calculated light function for each azimuthal direction times the magnitude of the corresponding projected normal N_p.

In summary we can say, that this is a very fast implementation of soft and hard shadows for dynamic height-fields running entirely on the GPU but not capable of rendering soft-shadows from local lights.

3.11 Fast Global Illumination on Dynamic Height Fields

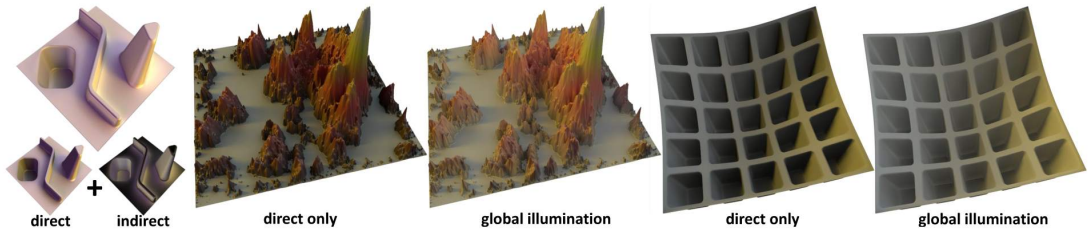


Figure 3.18: The images compare different scenes with global illumination and direct illumination only. Image courtesy of Nowrouzezahrai [19]

Nowrouzezahrai et al. [19] extended their previous method [29] presented in Section 3.9 to compute indirect lighting and to support glossy surfaces.

To compute direct shadowing and indirect lighting two multi-resolution pyramids are generated. One is used as in their last method for shadows of direct lighting and the other to calculate indirect lighting. In contrast to their previous method, where only the maximum blocking angle is captured, they compute visibility and incident radiance from the entire height-field (not from the environment light only) in discrete azimuthal directions, representing the binary visibility function and the incident radiance function of a single azimuthal direction analytically in the Normalized Legendre Polynomial (NLP) basis (see Section four in the paper [19]). Normalized Legendre Polynomials are used instead of spherical harmonic basis functions since they are used to describe the incident radiance and visibility of a single azimuthal direction, i.e. a one-dimensional function of the elevation angle, not a spherical function. Additionally, the order-n NLP basis only contains n basis functions compared to order-n SH basis containing n^2 basis functions.

Since visibility and incident radiance are stored in the NLP-basis for each azimuthal direction, the authors introduce blending matrices from NLP to SH to convert the 1D visibility and radiance functions to 2D visibility and radiance wedges on the sphere. This eliminates the need to precompute a 2D-table of wedge SH coefficient vectors for all combinations of pairs of horizon angles as they did it in their previous work. The total visibility vector can then be computed by summing over all wedges.

In summery we can say, this method represents visibility and incident radiance efficiently using the Normalized Legendre Polynomials. Furthermore, even sharper shadows are possible for key lights than in their previous method. However, it is still not possible to compute soft shadows from local light sources.

CHAPTER 4

Realistic Local Lighting in Dynamic Height Fields

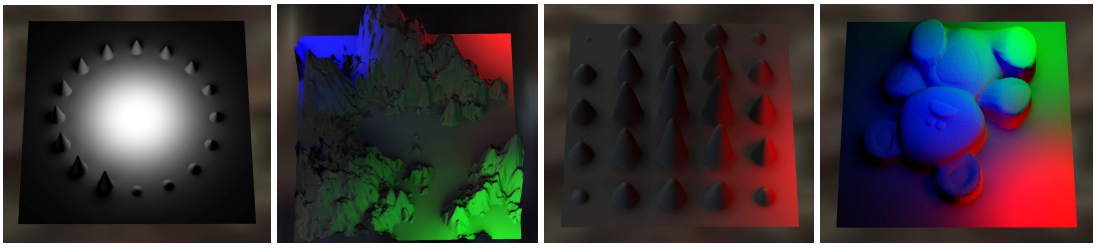


Figure 4.1: The images show different scenes and settings rendered with the method described in the following chapter. The scenes, including lights, camera and geometry are fully dynamic.

In the following chapter, we present a new method to approximate soft shadows and diffuse direct lighting from local lights and environment light sources on dynamic height-fields. The method extends the work of Snyder et al. [29], which we summarized in Section 3.9.

In our method, soft shadows from environment and local light sources are computed in two separate passes. To calculate soft shadows from environment light sources, the method described by Snyder et al. [29] is used. In our work, we extend this method to handle soft shadows from local light sources. We will describe our method in detail in this chapter.

This chapter is structured in two parts: In the first part 4.2, we explain how the authors [29] calculate soft shadows lit by an environment map. In the second part, we describe how to extend the method by Snyder et al. [29] to calculate soft shadows lit by local light sources and how to calculate total outgoing radiance at a receiver point from an environment light source and local light sources.

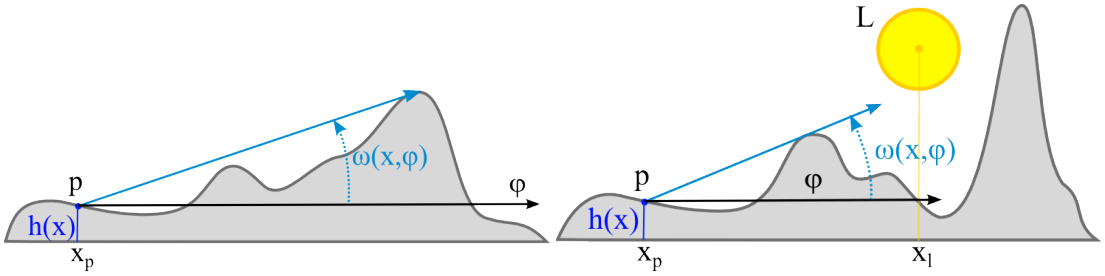


Figure 4.2: The image on the left shows the horizon angle at receiver point p for a single azimuthal direction lit by an environment map. The image on the right shows the horizon angle at receiver point p for a single azimuthal direction lit by a local light source L.

4.1 Overview

To calculate soft self-shadowing on a height-field lit by an environment light, we have to find the set of directions on the hemisphere in which the environment is visible from each receiver point. To do this, we find the maximum elevation angle of the height-field along each azimuthal direction, effectively defining the horizon as visible from a receiver point. A brute force approach requires sampling height differences between each receiver point and all height-field samples.

To reduce sampling, a *multi-resolution pyramid* is computed on the original height-field data. The multi-resolution pyramid is a stack of filtered versions of the original height-field data with increasing filter size. We sample detailed levels closer to the receiver point, and as the distance to the receiver point increases we sample from successively coarser levels. For each azimuthal direction from a given receiver point, the maximum elevation angle of the height-field is computed (see Figure 4.2) yielding a set of elevation angles, one for each azimuthal direction. A continuous horizon function is reconstructed from these elevation angles, describing the visibility of the environment from the receiver point. For efficiency, this function is stored in the spherical harmonics basis.

To compute the visibility of *local lights*, we have to find the maximum elevation angle for all azimuthal directions in the interval between the receiver point and the position of the light center, as shown in Figure 4.2.

Shading a point on the hight field is done by computing the spherical harmonics triple product integral of the lighting function – either the emittance of the environment or of the local lights – the visibility function and the BRDF, in our case a clamped cosine over the hemisphere.

4.2 Soft Shadows from Environment Lights

To calculate soft shadows from an environment light source, we use the method described in [29]. In this section, we will describe this method in detail, including how to calculate self-visibility for all receiver points on the height-field and how to calculate the final shading.

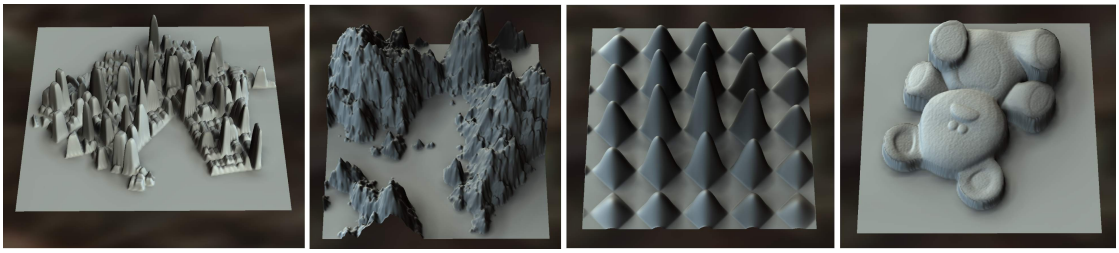


Figure 4.3: Images rendered using the method of Snyder et al. [29]

4.2.1 Self-Visibility of Height fields

A height-field is represented by a tabulated grid of scalar height values in two dimensions and is defined by

$$\{(x, y, h(x, y)) \mid (x, y) \in R^2\} \tag{4.1}$$

where $p = (x, y, h(x, y))$ is a three-dimensional point on the height-field and $h(x, y) \in R$ is a scalar height value at location (x, y).

To compute the visibility of the distant environment at a receiver point, we need to reconstruct the complete horizon, which is equivalent to finding the *horizon angle* in each azimuthal direction parametrized by $(cos(\varphi), sin(\varphi)), \varphi \in [0, 2\pi]$. The horizon angle represents the maximum angular elevation of the horizon above 0 degrees in one azimuthal direction φ from the receiver point p and is defined:

$$\omega_{x,y}(\varphi) = \max_{d \in (0,\infty)} atan\left(\frac{h(x + d\ cos(\varphi), y + d\ sin(\varphi)) - h(x, y)}{d}\right) \tag{4.2}$$

where $\omega_{x,y}(\varphi)$ represents the horizon angle or maximum elevation angle and d is the continuous distance along a azimuthal direction φ away from the receiver point (x, y).

The continuous visibility function on the hemisphere of a receiver point (x, y) with horizon angles $\omega_{x,y}(\varphi)$ is then

$$V_{x,y}(\theta, \varphi) = \begin{cases} 1, & \text{if } \theta > \omega_{x,y}(\varphi) \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where $\theta \in [0, pi/2]$ is the altitude angle and $\varphi \in [0, 2pi]$ the azimuthal angle. Informally, the environment is visible on those parts of the hemisphere that have an elevation angle above the horizon.

A brute-force sampling of the horizon angle function would require evaluating a prohibitive amount of samples. To reduce sampling, Equation 4.2 can be approximated using a multi-resolution pyramid.

Multi-resolution image pyramid computation

To reduce the sampling overhead along an azimuthal direction we sample more densely close to the receiver point and less densely as the distance away from the receiver point increases. We

sample more densely close to the receiver point because geometry closer to the receiver point has more influence to the final result. To avoid aliasing artefacts the original height-field data is pre-filtered with a *bicubic b-spline* [24] filter with a kernel size inversely proportional to the sampling density.

To reduce the processing overhead for each receiver point, the aforementioned filtering steps are computed once in each frame, resulting in a *multi-resolution image pyramid*.

The multi-resolution image pyramid represents a stack of filtered versions of the original height-field data with increasing filter size. Each level of the multi-resolution pyramid is denoted by $h_i(x, y),\ i \in \{0, 1, ..., n-1\}$ where i is the level index and n the total number of pyramid levels. The multi-resolution pyramid is constructed by iteratively decimating the original height data with a bilinear filter from finest to the coarsest level, which is called the *reduction step*, followed by bicubic b-spline reconstruction of each level back to the original resolution. We will give a detailed description on the bicubic b-spline interpolation method [24] in chapter 5.

The total number of pyramid level is

$$n = \left\lceil ls \left(\frac{\ln br}{\ln 2} \right) \right\rceil + 1 \tag{4.4}$$

where ls is called *level step* and br is the resolution of the original height-field. The level step is used to compute finer transitions between subsequent pyramid levels resulting in extra levels between power-of-two reductions. The reduction in resolution between two pyramid levels is given by $sr = 2^{(1/ls)}$. A level step of 1 gives power-of-two reductions, other values for the level step increase or decrease the reduction between consecutive pyramid levels.

In the next sections, we will describe how to use the multi-resolution image pyramid to approximate the horizon angle for one azimuthal direction at a receiver point.

Horizon angle approximation

When sampling the height-field along a single azimuthal direction away from a receiver point, the samples are placed at a fixed set of distances $d_1, ..., d_n$ from the receiver point. The spacing between the sample points increases with the distance from the receiver point. To avoid aliasing from undersampling, one sample is taken from each pyramid level. The sample closest to the receiver point is taken from the finest pyramid level, while the sample farthest away is taken from the coarsest level. The *sample spacing* $ss_i = d_{i+1} - d_i$ and the reduction between pyramid levels is directly related by $ss_i = sr^{-i} = 2^{(-i/ls)}$. This ensures that each sample is taken from an optimally pre-filtered version of the height-field. Setting $ls = 4$ produces visually fine transitions for soft shadows.

The elevation angle is approximated using the *multi-scale directional derivative* introduced by Snyder et. al. [29]. The multi-scale directional derivative is defined in the discrete setting using finite differences:

$$D_{x,y,i}(\varphi) = \frac{h_i(x + d_i cos(\varphi), y + d_i sin(\varphi)) - h_i(x, y)}{d_i} \tag{4.5}$$

where i is the a pyramid level, h_i the height value at (x, y) for pyramid level i, φ the azimuthal direction and d_i the sample distance.

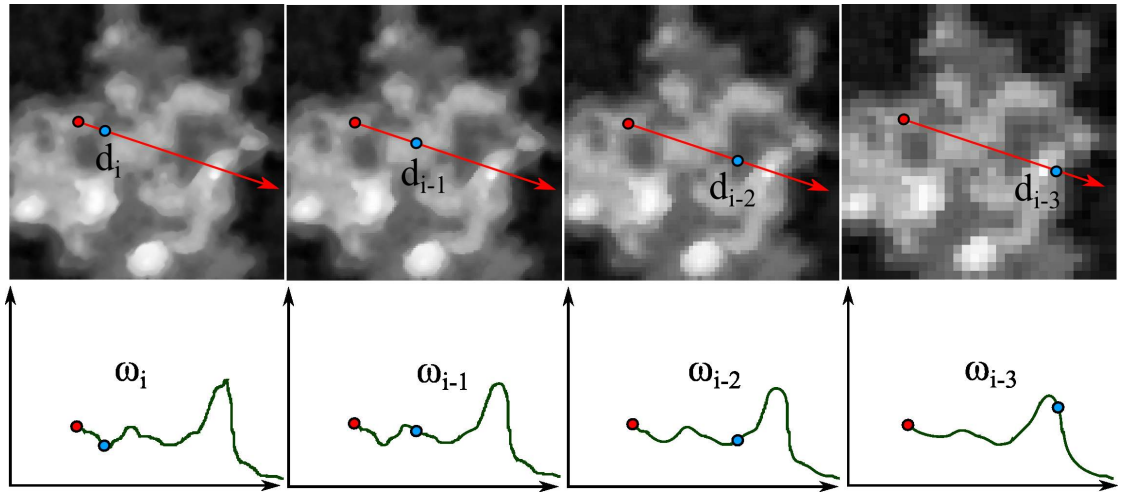


Figure 4.4: The top row shows filtered versions of the original height-field where height differences are sampled along the azimuthal direction at distance d_i. The bottom row shows pyramid slices of the filtered versions of the original height-field along with the associated samples along the azimuthal direction.

The elevation angle for a pyramid level i is defined by

$$\omega_{x,y,i}(\varphi) = atan(D_{x,y,i}(\varphi)) \tag{4.6}$$

and is computed at each pyramid level i, resulting in a set of elevation angles for each azimuthal direction of a height-field point (x, y). The authors compute the inverse tangent because it is much better to interpolate in the space of angles since the tangent is an unbounded function. If we would interpolate it directly, it would overaccentuate shadowing.

The next section explains how to reconstruct a continuous function of elevation angles from the computed samples $\omega_{x,y,i}$.

Continuous elevation function

To find the maximum elevation angle in a given azimuthal direction, we first reconstruct a one-dimensional continous function from the elevation angle samples, then we find the maximum of the function. A linear reconstruction would result in noticeable discontinuities in the shadows, therefore we reconstruct the continous function using splines resulting in smoother transitions (see Figure 9 in Snyders paper [29]).

For a given azimuthal direction φ_j, uniform cubic B-spline interpolation is used to reconstruct a continuous 1D elevation angle function denoted $\omega_{x,y}(\hat{d}, \varphi)$ from a set of elevation angle samples $\omega_{x,y,i}(\varphi)$ at each pyramid level i. $\hat{d}$ is the negative log distance away from the receiver point $\hat{d} = -log_{sr}d$, where sr is the reduction in resolution between two pyramid levels (see Section 4.2.1) and d is the continous distance from (x, y). The set of elevation angles is denoted

$$\Omega = \{\omega_{x,y,0}(\varphi_j), \omega_{x,y,1}(\varphi_j), ..., \omega_{x,y,n-1}(\varphi_j)\} \tag{4.7}$$

for one azimuthal direction. To reconstruct the continuous elevation function, each sequence of four subsequent samples $(\omega_{x,y,i-1}, \omega_{x,y,i}, \omega_{x,y,i+1}, \omega_{x,y,i+2})$ is interpolated using the following equation:

$$\omega_{x,y}(\hat{d}, \varphi) = b_0(t)\omega_{x,y,i-1} + b_1(t)\omega_{x,y,i} + b_2(t)\omega_{x,y,i+1} + b_3(t)\omega_{x,y,i+2}, \tag{4.8}$$

where the B-spline basis or filter weights b_i for cubic B-Splines consist of the following basis functions:

$$b_0(t) = \frac{(1-t)^3}{6} \tag{4.9}$$

$$b_1(t) = \frac{3t^3 - 6t^2 + 4}{6} \tag{4.10}$$

$$b_2(t) = \frac{-3t^3 + 3t^2 + 3t + 1}{6} \tag{4.11}$$

$$b_3(t) = \frac{t^3}{6} \tag{4.12}$$

The value $t = \hat{d} - \lfloor \hat{d} \rfloor \in [0, 1]$ is the blending distance between the values at $i = \lfloor \hat{d} \rfloor$ and $i + 1$. For cubic B-splines with uniform knot-vector, the blending functions can be precalculated and are equal for each segment.

The maximum elevation angle over all d for a receiver point along a single azimuthal direction is then approximated by

$$\omega_{x,y}(\varphi) \approx \max_{\hat{d} \in [0, n-1]} (\omega_{x,y}(\hat{d}, \varphi)) \tag{4.13}$$

Equation 4.13 is approximated by taking the maximum of a number of sample points on the B-Spline (for details see Section 5.2).

We constrain the function above to be non-negative, i.e. we disallow the horizon to be on the lower hemisphere of a receiver point. This reflects the fact that we assume the height-field to extend to infinity, just like real terrain seems to extend to infinity from the point of view of an observer.

To control the sharpness of the shadows, an additional parameter l_o called *level offset* is introduced to bias the pyramid level access when evaluating Equation 4.5. Height values are now looked up in finer pyramid levels, i.e. the pyramid level h_i in Equation 4.5 is replaced by $h_{[i+l_s*l_o]}$ where l_s is the level step which we already mentioned in section 4.2.1.

$$[i + l_s * l_o] = \min(\lceil i + l_s * l_o \rceil, n - 1) \tag{4.14}$$

With increasing l_o, the shadow gets less blurred at the cost of increased aliasing. Values between 2 and 4 produce nice results in terms of shadow sharpness.

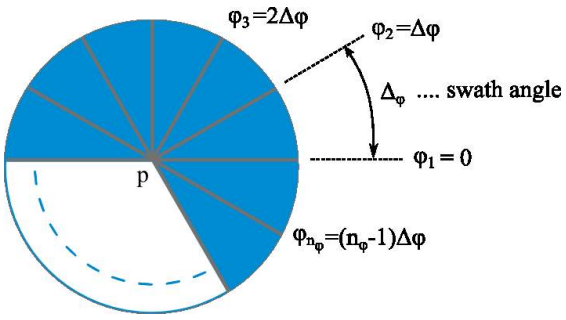


Figure 4.5: The figure shows a complete visibility swath, where the gray lines emanating form the center are representing the azimuthal sampling directions.

Visibility function

We now describe how to approximate total visibility at a receiver point p. In the last section, we described how to compute the horizon angle for a single azimuthal direction φ. Since we want to calculate the complete horizon for a receiver point p, in this section we first describe how to approximate the complete horizon at a receiver point p from a small set of horizon angles ω_i. Then we explain how to compute the complete visibility function from the horizon function.

To approximate the complete horizon at a receiver point, we compute the horizon angle for a small set of azimuthal directions φ_i, resulting a discrete set of horizon angles. This is done to minimize sampling overhead. To reconstruct the complete horizon, we linearly interpolate between adjacent pairs of horizon angles. The complete domain of azimuthal directions $\Phi = [0, 2\pi]$ is partitioned into n_φ equally spaced azimuthal directions $\{\varphi_1, \varphi_2, ..., \varphi_{n_\varphi}\}$ with spacing $\Delta\varphi = \varphi_{i+1} - \varphi_i = \Phi/n_\varphi$ (see Figure 4.5). The interval between two consecutive azimuthal directions is called partial swath. The complete swath is the union of n_φ partial swaths.

For each azimutahl direction φ_i, we compute the horizon angle ω_i as described in Section 4.2.1. Given a partial swath $[\varphi_i, \varphi_{i+1}]$ and the corresponding horizon angles ω_i and ω_{i+1}, a continous blocker function $\omega(\varphi)$ can then be reconstructed in the partial swath by

$$\omega(\varphi) = \omega_i + \frac{\varphi - \varphi_i}{\Delta\varphi}(\omega_{i+1} - \omega_i). \tag{4.15}$$

Next we define the spherical visibility function. Because we deal with spherical functions, we first define the space of directions on the unit sphere S. A unit direction vector $\hat{u}$ is parametrized by the complete azimuthal angle $\varphi \in [0, 2\pi]$ and the elevation angle $\theta \in [-\pi/2, +\pi/2]$. The space of directions is defined by

$$\hat{u}_S(\varphi, \theta) = (cos(\varphi)cos(\theta), sin(\varphi)cos(\theta), sin(\theta)). \tag{4.16}$$

A *spherical wedge* (see Figure 4.6) is the set of directions between two consecutive azimuthal directions $\{\hat{u}_s(\varphi, \theta) : \varphi \in [\varphi_i, \varphi_{i+1}], \theta \in [0, \pi]\}$. The spherical visibility function for a single visibility wedge determines if a ray emanating at receiver point p in a direction of the wedge is occluded by the height-field or if the environment is visible. The visibility function $v_i(\varphi, \theta)$ for spherical wedge i is defined as

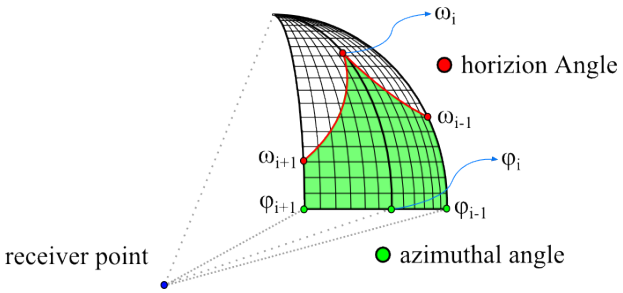


Figure 4.6: The visibility wedge geometry. The visibility function on a single hemispherical wedge is parameterized by two azimuthal directions $(\varphi_i, \varphi_{i+1}$ and its corresponding horizon angles (ω_i, ω_{i+1}). It is constructed by linearly interpolating the horizon function between two samples and projecting the directions onto the sphere.

$$v_i(\varphi, \theta) = \begin{cases} 1, & \text{if } \varphi \in [\varphi_i, \varphi_{i+1}] \ \ and \ \theta > \omega(\varphi) \\ 0, & \text{otherwise} \end{cases} \tag{4.17}$$

and evaluates to 1 if the environment is visible otherwise to 0. To approximate the complete visibility function $V_{x,y}$, we accumulate the visibility functions $v_i(\varphi, \theta)$ for each azimuthal direction.

The approximated total visibility $V_{x,y}$ at receiver point can now be projected to the spherical harmonic basis. In a pre-computation step, the spherical harmonic projection v_1 of the visibility function defined on the first wedge is precomputed for all possible combinations of angles ω_l and ω_r, where ω_l represents a discrete horizion angle sample along direction φ_0 and ω_r a discrete horizion angle sample along direction φ_1. In practice we sample n angles from ω_l and ω_r, for a total of $n \times n$ combinations. We denote the table of resulting spherical harmonic coefficients $\mathbf{v}_{\omega_l,\omega_r}$, each entry can be computed as:

$$\mathbf{v}_{\omega_l,\omega_r} = \int_H v_{\omega_l,\omega_r}(\varphi, \theta)\mathbf{y}(s)ds \tag{4.18}$$

where $v_{\omega_l,\omega_r}(\varphi, \theta)$ is the visibility function defined in 4.17 for a given pair of horizon angles ω_l and ω_r. $\mathbf{v}$ is a spherical harmonic coefficient vector and $\mathbf{y}$ are the spherical harmonics basis functions y_i arranged into a vector. We will give a detailed description in Section 5 on how this tabulation is implemented on the GPU.

At runtime, given two adjacent horizon angles $\omega_{x,y}(\varphi_i), \omega_{x,y}(\varphi_{i+1})$, we look up the spherical harmonic vector $\mathbf{v}_{\omega_l,\omega_r}$ in the wedge table with ω_l and ω_r as close as possible to $\omega_{x,y}(\varphi_i)$ and $\omega_{x,y}(\varphi_{i+1})$. Then we rotate $\mathbf{v}$ to direction φ_i in the spherical harmonic basis (see Figure 4.7). Sloan et al. [27] demonstrate that a z rotation in the spherical harmonic basis can be computed more efficiently. We do this for all azimuthal directions φ_i and accumulate the rotated vectors. The resulting total visibility coefficient vector $\mathbf{V}_{x,y}$ is then represented by

$$\mathbf{V}_{x,y} = R_{\varphi_0}(\mathbf{v}_{\omega_{l1},\omega_{r1}}) + R_{\varphi_1}(\mathbf{v}_{\omega_{l2},\omega_{r2}}) + ... + R_{\varphi_{n-1}}(\mathbf{v}_{\omega_{ln},\omega_{rn}}) \tag{4.19}$$

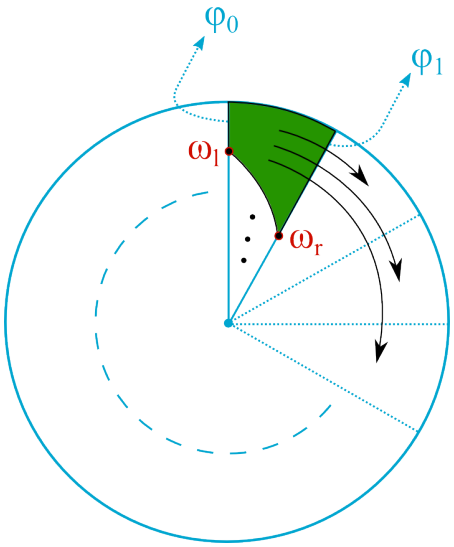


Figure 4.7: The visibility wedge geometry. On this image, we show that one spherical harmonic vector $\mathbf{v}_{\omega_l,\omega_r}$ (represented by the green wedge) from the wedge table is z-rotated to a given pair of horizion angle with their corresponding azimuthal directions.

where $\mathbf{v}_{\omega_l,\omega_r}$ is the spherical harmonic vector for a pair of horizon angles and ω_{li} and ω_{ri} are as close as possible to $\omega_{x,y}(\varphi_i)$ and $\omega_{x,y}(\varphi_{i+1})$, R_{φ_i} denotes the z-rotation in the spherical harmonic basis of $\mathbf{v}_{\omega_l,\omega_r}$ to direction φ_i.

4.2.2 Calculate shadowed outgoing radiance at receiver points

In this section, we describe how to calculate the shadowed outgoing radiance for each receiver point p lit by a environment map. Assuming we use only diffuse BRDFs, the shadowed exit radiance at each receiver point can be expressed via:

$$L^e_{x,y} = \frac{\rho}{\pi} \int_H L_{env}(\upsilon_i) V_{x,y}(\upsilon_i) C(n_{x,y}, \upsilon_i) d\upsilon_i \tag{4.20}$$

where L_{env} is the radiance from the environment light source as a function of direction $\upsilon_i \in \hat{u}_s(\theta, \phi)$ which is defined on the hemisphere, H the upper hemisphere sphere, $V_{x,y}(\omega_i)$ is the visibility function at receiver point (x, y) and $n_{x,y}$ denotes the unit normal vector. The diffuse BRDF term is a constant factor ρ and the clamped cosine hemisphere $C(n_{x,y}, \upsilon_i) = max(n_{x,y} \cdot \upsilon_i, 0)$.

We efficiently evaluate this integral in the spherical harmonics (SH) basis. The SH representation of $V_{x,y}$ is already known (see Section 4.2.1). The environment lighting SH coefficient vector $\mathbf{L}_{env}$ is computed once each frame from the given environment light source L_{env}. The SH coefficient vector $\mathbf{C}(n_{x,y})$ of the clamped cosine hemisphere is precomputed for a single normal direction and rotated at run-time to the actual normal direction using zonal harmonic rotation rules (see Section 3.4.4). The zonal harmonic coefficients for the clamped cosine function for a single normal direction can be computed using the method by Ramamoorthi et al. [20].

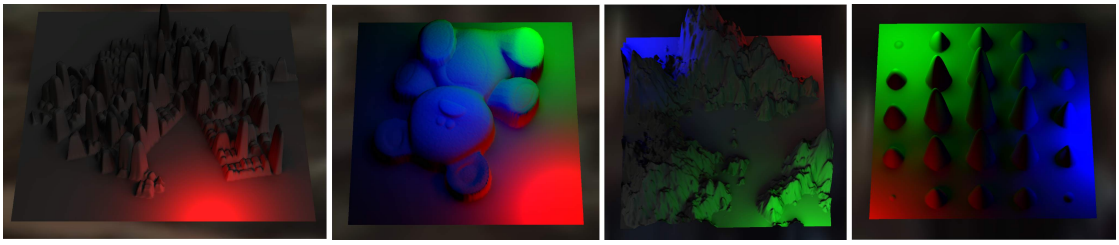


Figure 4.8: Images are rendered using our method for local lights.

The main challenge in dynamic scenes is finding the visibility function $V_{x,y}$ for each receiver point (x, y). In Section 4.2.1, we described how to approximate total visibility at receiver point (x, y) efficiently. The integral in Equation 4.20 can be approximated as

$$L^e_{x,y} \approx \rho * (\mathbf{L}_{env} \cdot (\mathbf{V}_{x,y} \bullet \mathbf{C}(n_{x,y}))) \tag{4.21}$$

where the binary spherical harmonic product $(\mathbf{V}_{x,y} \bullet \mathbf{C}(n_{x,y}))$ (see Section 3.4.2) is called the *transfer vector*. The final outgoing radiance $L^e_{x,y}$ is the dot product between $\mathbf{L}_{env}$ and the transfer vector.

Since this form of equation 4.20 can be evaluated at run-time, all terms, including the environment lighting, can change dynamically.

4.3 Soft Shadows from Local Lights

In this section, we will describe how to extend the method of Snyder et al. [29] presented in the last chapters to calculate soft shadows from local light sources. We use spherical lights as local light sources and explain how to approximate the horizon angle and the total visibility at a receiver point for these spherical lights. Then, we show how to approximate incoming radiance from a spherical light source at receiver point. Finally we describe how to calculate the shadowed outgoing radiance for all receiver points on the height-field.

4.3.1 Self-Visibility of Height-fields

The visibility of a local light is determined by the height-field geometry between the receiver point and the local light. Geometry at a larger distance from the receiver point than the local light can not occlude the local light. Therefore, we define the visibility function for a local light just as the visibility of the environment (see Equation 4.3) but only taking the geometry between the receiver point and the local light into account. The horizon angle for a local light is then the maximum elevation angle of the horizon for a single azimuthal direction φ from the receiver point to the local light position $(x, y)_l$ and is expressed via:

$$\omega^{l_i}_{x,y}(\varphi) = \max_{d \in [0, d_{l_i}]} atan\left(\frac{h(x + d\ cos(\varphi), y + d\ sin(\varphi)) - h(x, y)}{d}\right) \tag{4.22}$$

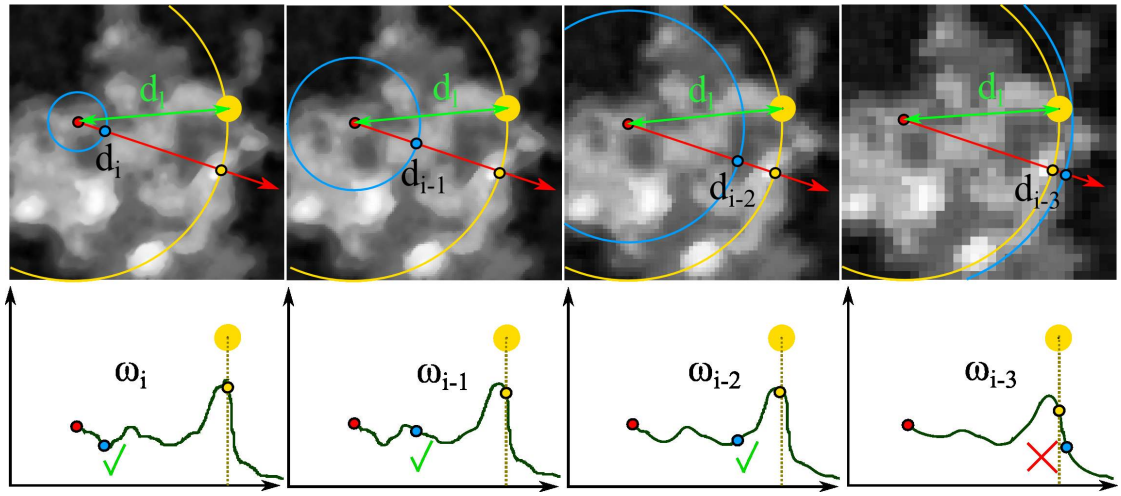


Figure 4.9: The top row shows filtered versions of the original height-field with height differences sampled along the azimuthal direction at distance d_i between the receiver point and the light source. The green d_l is the distance to the local light source. The bottom row shows slices of the filtered versions of the original height-field along with the associated samples along the parameter d_i between the receiver point and the local light source. Samples behind the local light source are not used.

where $\omega_{x,y}^{l_i}(\varphi)$ is the horizon angle and d is the distance between the receiver point and the local light source position along a azimuthal direction φ. In contrast to Section 4.2, the elevation angle can now take on values below zero degrees if the receiver point has a larger height than all other points on the azimuthal direction. This is due to the fact that the blocking geometry between the receiver point and the light source is no longer assumed to extend to infinity (see Section 4.2.1).

The continuous visibility function is defined on the sphere at a receiver point (x, y) in contrast to Section 4.2.1 where it was defined on the hemisphere. We define the continuous visibility function with horizon angles $\omega_{x,y}^{l_i}(\varphi)$ via

$$V_{x,y}^{l_i}(\theta, \varphi) = \begin{cases} 1, & \text{if } \theta > \omega_{x,y}^{l_i}(\varphi) \\ 0, & \text{otherwise} \end{cases} \tag{4.23}$$

with the altitude angle $\theta \in [-\pi/2, \pi/2]$ and $\varphi \in [0, 2\pi]$ the azimuthal angle. Because the altitude angle is in the range of $-\pi/2$ and $\pi/2$, we can also capture radiance from a local light sources beneath a receiver point.

Horizon angle approximation

To approximate the horizon angle for a local light source, the multi-resolution image pyramid described in Section 4.2.1 is used. When computing the continous elevation function (see Section 4.2.1), the set of elevation angle samples Ω in Equation 4.7 is reduced to the samples having

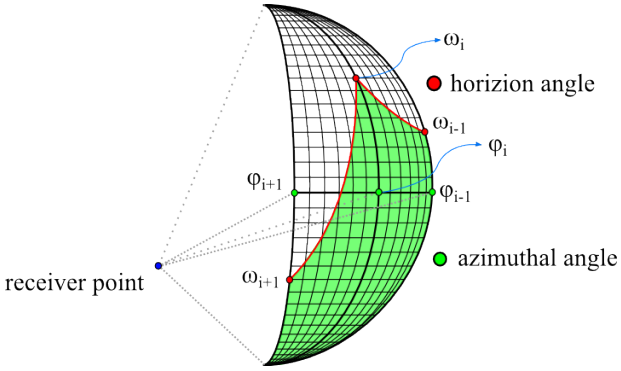


Figure 4.10: The visibility wedge geometry. The visibility wedge is parametrized by the azimuthal directions $(\varphi_i, \varphi_{i+1})$ and its corresponding horizon angles (ω_i, ω_{i+1}). By linearly interpolating the visibility function between the horizon angles ω_i and ω_{i+1} followed by projecting the interpolated values along the directions onto the sphere, we define a single visibility wedge.

a smaller distance to the receiver point than the local light. The distance d_{l_i} of the local light and the distance d_j of the elevation angle sample are both computed in the plane of the height-field (i.e., without taking height differences into account) and the distance d_{l_i} is computed from the center of the local light.

Let d_{k_i} denote the smallest sample distance in $\{d_1, ..., d_n\}$ larger than the local light distance d_{l_i}. The set of elevation angle samples for this local light is then

$$\Omega_{l_i} = \left\{ \omega^{l_i}_{x,y,0}(\varphi), \omega^{l_i}_{x,y,1}(\varphi), ..., \omega^{l_i}_{x,y,k_i}(\varphi) \right\} \tag{4.24}$$

i.e. the elevation angle samples closer to the receiver point than the local light. The last sample $\omega^{l_i}_{x,y,k_i}(\varphi)$ has a distance larger than the light, so the continous elevation function is defined at the distance d_{l_i} of the local light.

To reconstruct the continuous elevation function $\omega^{l_i}_{x,y}(\hat{d}, \varphi)$ for one azimuthal direction φ, we use the uniform cubic B-spline interpolation scheme described in Section 4.2.1 using the set of elevation angles from Equation 4.24 as input.

The maximum elevation angle for a local light source l_i along a single azimuthal direction is then approximated by

$$\omega^{l_i}_{x,y}(\varphi) \approx \max_{\hat{d} \in [0, \hat{d}_{l_i}]} (\omega^{l_i}_{x,y}(\hat{d}, \varphi)), \tag{4.25}$$

where $\hat{d}$ is the negative log distance away from the receiver point p as described in Section 4.2.1 and $\hat{d}_{l_i}$ is the negative log distance of the local light. As described in Section 4.2.1, we approximate the continous maximum by taking the maximum of a set of samples of the B-Spline function. For details see Section 5.2. As explained in Section 4.2.1, the sharpness of the shadows can also be controlled by biasing the pyramid level access when evaluating Equation 4.25.

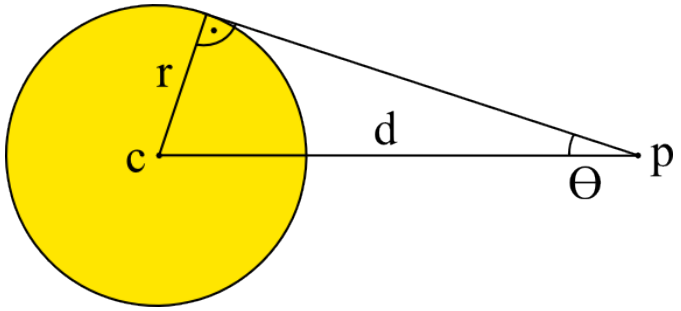


Figure 4.11: The spherical light source geometry. θ represents the half angle between the receiver point p and the spherical light source. Where $\sin(\theta) = \frac{r}{d}$ and c is the center of the spherical light source.

Visibility function

The visibility function is constructed from the elevation angles $\omega^{l_i}_{x,y,j}(\varphi)$ analogous to the environment lighting case (see Section 4.2.1), with the difference that the visibility function is defined on the complete sphere instead of the hemisphere. The visibility wedges are also defined on the complete range of elevation angles (see Figure 4.10) and the coefficient table for the wedges is precomputed as:

$$\mathbf{v}^{l_i}_{\omega_l,\omega_r} = \int_S v^{l_i}_{\omega_l,\omega_r}(\varphi,\theta)\mathbf{y}(s)ds \tag{4.26}$$

where $v^{l_i}_{\omega_l,\omega_r}(\varphi,\theta)$ is now defined on the sphere. Then at runtime we compute final total visibility coefficient vector $\mathbf{V}^{l_i}_{x,y}$ for a local light source exactly as described in Section 4.2.1.

Spherical light source

In our extended method, we model local lights as spherical lights sources. The spherical light source is parametrized by the center c_{l_i} and radius r_{l_i} (see Figure 4.11). We want to calculate the amount of incident radiance arriving from the light source at a receiver point. The incident radiance at the receiver point can be approximated by projecting the light source onto the unit sphere centered around the receiver point. This results in a function that is always circularly symmetric, therefore it can be approximated with a zonal harmonic vector that is rotated at runtime to align with the direction vector from receiver point to the light source using rotation rules for zonal harmonic vectors (see Section 3.4.4).

Sloan [26] computes the zonal harmonics coefficients in closed form as a function of the angle θ:

$$\mathbf{z}^{l_i}_k(\theta) = \int_{\theta=0}^{\theta_{l_i}} \int_{\varphi=0}^{2\pi} y^0_k d\varphi d\theta, \tag{4.27}$$

where k is the band index of the zonal harmonic basis function, $\mathbf{z}^{l_i}$ is the zonal harmonic coefficient vector representing the spherical light source, $\theta_{l_i} = asin(r_{l_i}/d_{l_i})$ the subtended angle, d_{l_i}

the distance of the local light from a receiver point and y_k^0 is a zonal harmonic basis function. See Appendix A.1 for the symbolic integrals of equation 4.27. At run-time, the zonal harmonics vector is rotated to the direction of the light source, resulting in the spherical harmonics coefficient vector $\mathbf{L}^{l_i}$, representing the radiance incident at a receiver point.

4.3.2 Calculate shadowed outgoing radiance at receiver points

In this section we compute shadowed outgoing radiance for each receiver point p lit by the local light sources. Like in Section 4.2.2, we assume only diffuse BRDFs. The shadowed exit radiance at receiver point can be expressed via:

$$L_{x,y}^{l} = \frac{\rho}{\pi} \sum_{j=0}^{n_l} \int_S L_{x,y}^{l_j}(\upsilon_i) V_{x,y}^{l_j}(\upsilon_i) C(n_{x,y}, \upsilon_i) d\upsilon_i \tag{4.28}$$

where n_l is the number of local lights, $L_{x,y}^{l_j}$ is the unoccluded incoming radiance as a function of direction υ_i from local light source l_j. $V_{x,y}^{l_j}$ is the visibility function corresponding to local light source l_j at receiver point (x, y). The clamped cosine hemisphere $C(n_{x,y}, \upsilon_i)$ is calculated as in Section 4.2.2 and the diffuse BRDF is a constant factor ρ.

In Section 4.3.1 we described how to approximate total visibility at receiver point efficiently and how to approximate incoming radiance from a local light source l_j at receiver point (x, y). These spherical harmonics approximations of the visibility function $\mathbf{V}_{x,y}^{l_j}$ and the incoming radiance $\mathbf{L}_{x,y}^{l_j}$ are then used to approximate the integral in Equation 4.28:

$$L_{x,y}^{l} = \sum_{j=0}^{n_l} \mathbf{L}_{x,y}^{l_j} \cdot (\mathbf{V}_{x,y}^{l_j} \bullet \mathbf{C}(n_{x,y})) \tag{4.29}$$

where $\mathbf{C}(n_{x,y})$ is the clamped cosine hemisphere around normal $n_{x,y}$. The spherical harmonic product $(\mathbf{V}_{x,y}^{l_j} \bullet \mathbf{C}(n_{x,y})$ are the transfer vectors for each light source l_j and the total outgoing radiance $L_{x,y}^{l}$ is the sum of each dot product between the $\mathbf{L}_{x,y}^{l_j}$ and the corresponding transfer vectors.

After the outgoing radiance at receiver points due to local lights $L_{x,y}^{l}$ from Equation 4.21 and the total outgoing radiance due to environment lighting from Equation 4.29 are found, we accumulate all contributions to get the final shadowed outgoing radiance at receiver point as

$$L_{x,y} = L_{x,y}^{e} + L_{x,y}^{l} \tag{4.30}$$

In the next section, we give some implementation details on the important parts of the method described so far.

CHAPTER 5

GPU Implementation

In our implementation, we used the OpenGL graphics library and the OpenGL shading language GLSL for rendering. The application code was written in C++ and compiled with the Visual Studio 2010 C++ compiler. A coarse overview of our implementation is given in Figure 5.1. In a global pre-computation step, the visibility wedge tables (see Sections 4.2.1 and 4.3.1) are created. Then, three steps are performed in each frame: A per-frame pre-computation step, creating the spherical harmonics representation of the light sources, the height field pyramid and height field normals, an environment lighting pass and a local lighting pass. For all spherical harmonic coefficient vectors used in our method, we use $n = 4$ spherical harmonic bands, which results in sixteen-component spherical harmonic vectors or four-component zonal harmonic vectors. This produces good results for low-frequency lighting and visibility while keeping memory and time requirements low. In the following sections, we describe important data structures which are used in the implementation and details of the algorithms used. First we describe the pre-computation step in Section 5.1, followed by the environment lighting and local lighting step in Sections 5.2 and 5.3.

5.1 Pre-Computation Step

Two types of light sources are supported in our implementation. The environment light source is sampled from a spherical High Dynamic Range (HDR) light probe [6] and represented as 3 spherical harmonic vectors, where each spherical harmonic vector represents one colour component from the RGB triple. A local light source is represented as a 4-component zonal harmonic vector. The local light source structure additionally stores the position and the radius of the light source.

For each light source type, we precompute a corresponding visibility wedge table on the CPU. We use Equation 4.18 to tabulate the low-order spherical harmonic projections of the visibility wedge function for the environment map, Equation 4.26 is used to compute the table for the local light sources. In our implementation, we pre-compute visibility wedges for 64 x 64

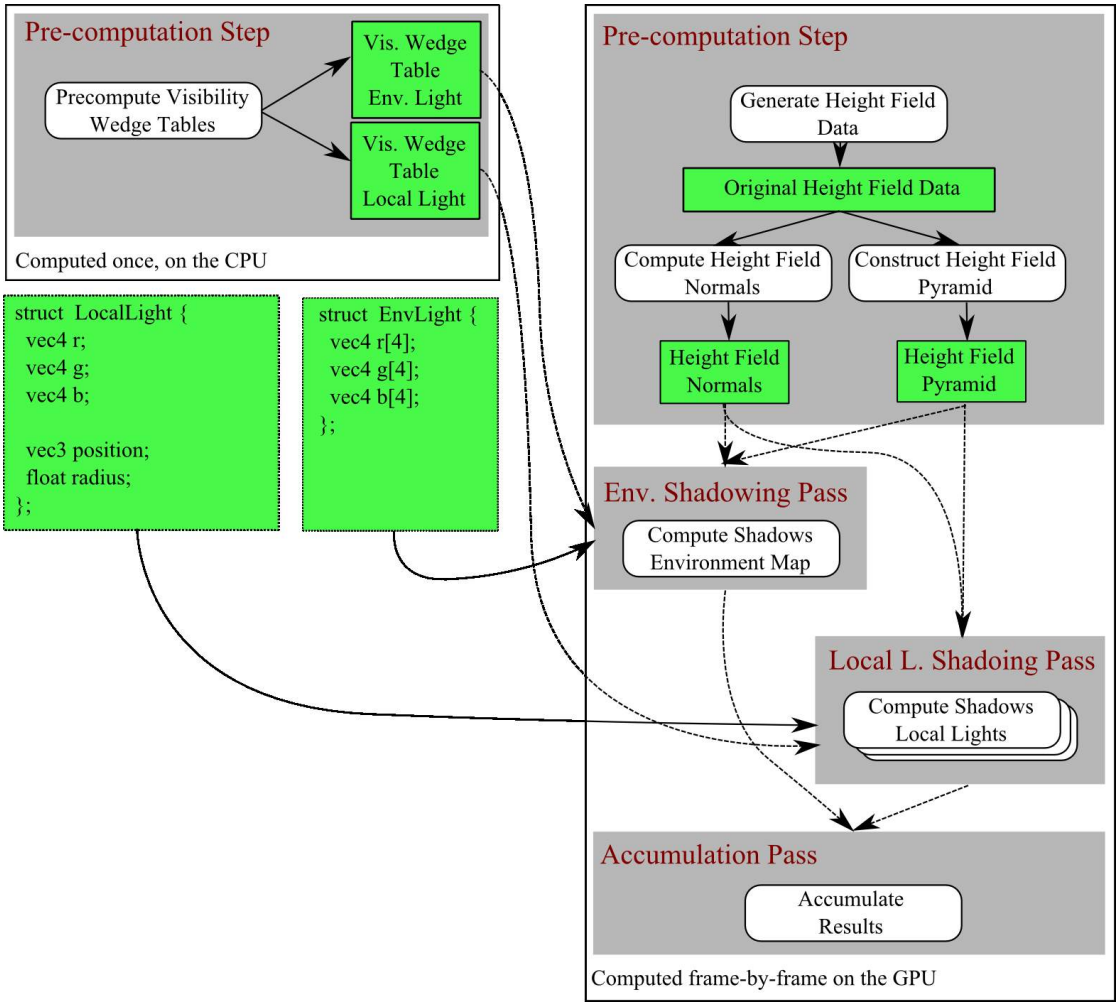


Figure 5.1: The green rectangles show data structures for the generate tables and light sources. Tables are stored in textures. The top left rectangle represents the pre-computation step of the visibility wedge tables, computed once on the CPU. The rectangle on the right shows the necessary steps to compute soft shadows from all light source types. This is done frame-by-frame on the GPU.

combinations of elevation angles and the tabulated information is stored in a OpenGL 2D texture of type RGBA float 32. Four texture layers are used to store four spherical harmonic coefficients in each layer (see Figure 5.3) for a total of 16 coefficients per pixel across all layers. Normals $\mathbf{N}_{x,y}$ are computed for each receiver point on the height field.

The height field pyramid is built from the original height field data. The texture size of the original data was set to 256×256 with one 16-bit float colour component to store the height value. The construction proceeds in two phases, a decimation phase and a reconstruction phase.

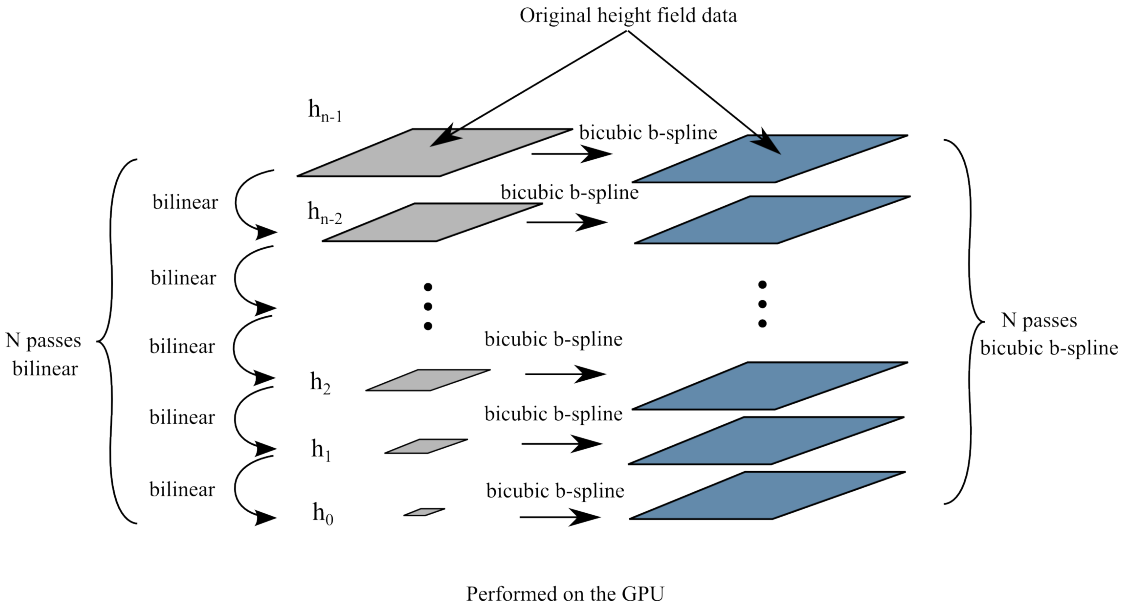


Figure 5.2: The left pyramid shows the decimated versions of the original height data. The right pyramid contains resampled versions using bicubic b-spline interpolation at each layer.

In the decimation phase, we iteratively run a bicubic down-sampling shader on the height field data, taking the result of the previous iteration as input for the next one. OpenGL frame buffer objects are used to render into the texture arrays. In OpenGL, a texture array can have one or more image layers, where each layer has the same resolution. When we start constructing the pyramid, we set the texture-array resolution to the original height field resolution, then the original height field data is copied into the top layer. Each decimation iteration results in a lower-resolution version of the data which is written at the beginning of the next image layer, leaving the rest of the layer empty. In the second phase, a second texture array is used to store the reconstructed smooth results from the first array. This is done using bicubic b-spline interpolation on each layer of the first texture array and storing the result in the second texture array using OpenGL frame buffer objects (see the processing scheme in Figure 5.2). Bicubic b-spline interpolation is used instead of bilinear interpolation, because this would produce discontinuities in shadows (see Figure 9 in the paper of Synder [29]).

5.2 Environment-light shadowing pass

In the environment-light shadowing pass, we compute the outgoing radiance at each height field point due to environment lighting. First, we construct the total visibility coefficient vector $\mathbf{V}_{x,y}$ (see Eq. 4.19), then we apply this visibility vector to the environment light to get the shadowed outgoing radiance.

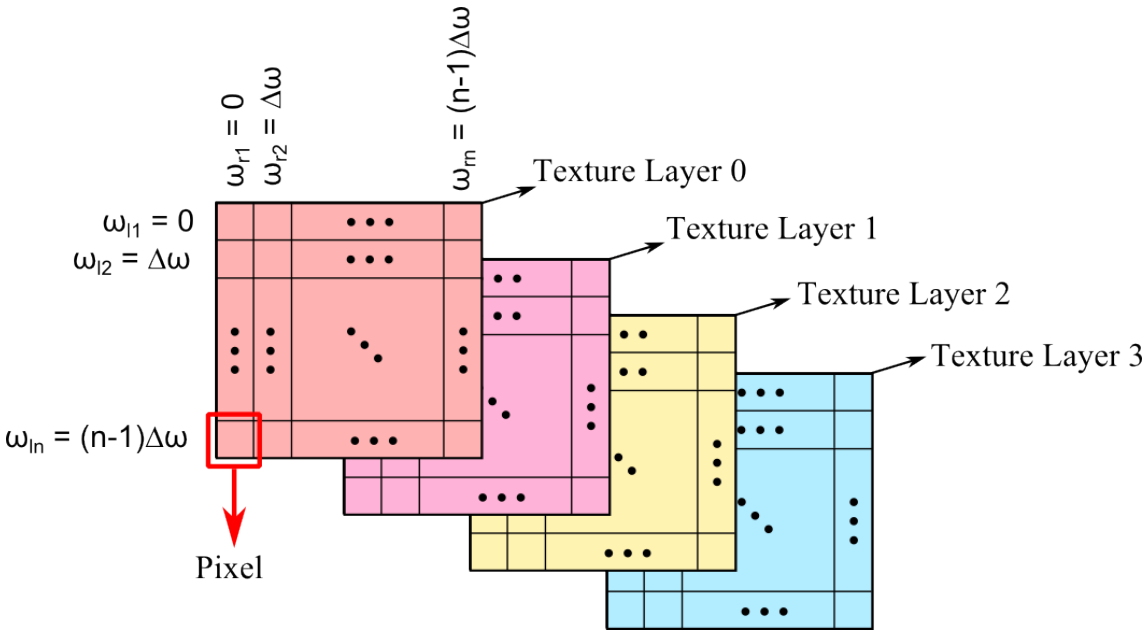


Figure 5.3: Four texture layers that store the visibility wedge table. Each pixel (RGBA float 32) of each layer stores 4 coefficients of the visibility wedge SH coefficient vectors.

Maximum horizon angle computation

The horizon function is approximated by calculating a set of horizon angles, using Equation 4.13, for a discrete set of azimuthal directions using the multi resolution height field pyramid. For improved efficiency, the azimuthal distances d_i in Equation 4.5 and their reciprocals $1/d_i$ are precomputed. To evaluate Equation 4.13, the b-spline function is evaluated at the elevation angle samples $\omega_{x,y,i}(\varphi)$ and at one additional point between every pair adjacent elevation angle samples. We can precompute the uniform b-splines weights, which are $\{1/6, 2/3, 1/6, 0\}$ for the knot points (elevation angle samples $\omega_{x,y,i}(\varphi)$) and $\{1/48, 23/48, 23/48, 1/48\}$ for the mid points (elevation angle sample between adjacent samples $\omega_{x,y,i+1/2}(\varphi)$).

Visibility function construction

In our implementation, we use $n_\varphi = 32$ azimuthal directions for the environment map when sampling visibility. The azimuthal directions $(cos(\varphi_i), sin(\varphi_i))$ are also precomputed. After calculating the set of horizon angles, we look up the projected visibility wedges by pairs of adjacent horizon angles and rotate them in the spherical harmonic basis using the zonal harmonics rotation technique by Sloan et al [27] to align them with each azimuthal direction. The 16-component spherical harmonic coefficient vectors for a pairs of horizon angles (ω_l, ω_r) are fetched from a texture array, where each texture layer stores four components of the spherical harmonic vectors (see Figure 5.3). Then we accumulate the rotated wedges to compute the total spherical harmonic visibility vector $\mathbf{V}_{x,y}$.

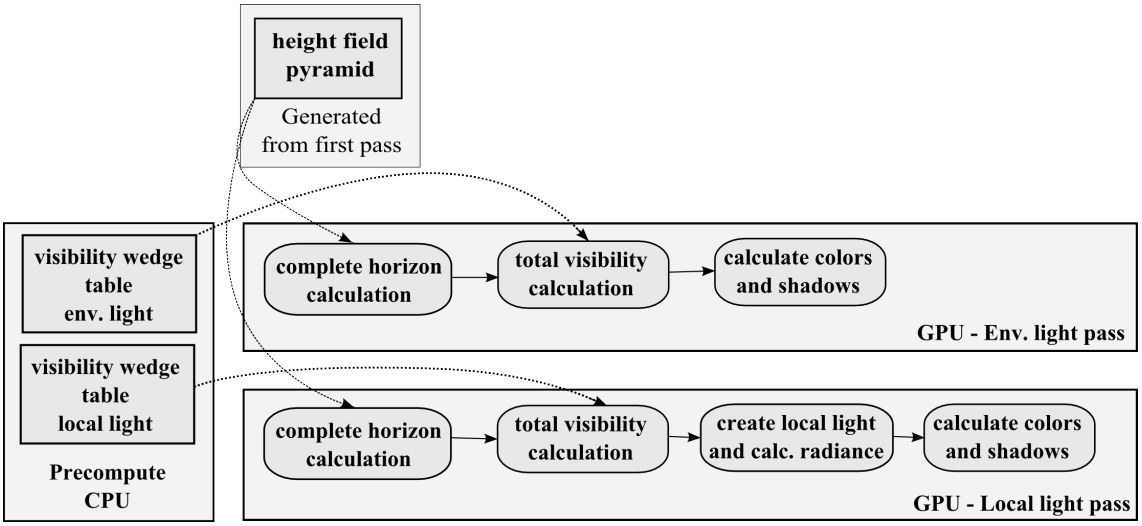


Figure 5.4: The main calculation steps of the environment shadowing pass and the local light shadowing pass (see Figure 5.1 for a complete overview).

Outgoing radiance

To compute outgoing radiance, we first align the clamped cosine function with the direction vector $\mathbf{N}_{x,y}$. The clamped cosine function is represented by the zonal harmonic vector [26] $1/\pi\,[0.886227, 1.02333, 0.495416, 0.0]$. Then the spherical harmonic product is formed with $\mathbf{V}_{x,y}$, resulting in the transfer vector $\mathbf{T}_{x,y}$. In the last step, the dot product of the spherical harmonic environment map vector and the transfer vector $\mathbf{T}_{x,y}$ is computed (see Figure 5.4 for an overview of the shader computation pipeline). The final shadowing is rendered to texture using OpenGL framebuffer objects.

5.3 Local light shadowing pass

To calculate shadows from multiple local light sources, as described in Section 4.2, one pass for each local light source is computed and rendered to texture using OpenGL framebuffer objects. The result is accumulated in an additional pass. We decided to use one shader per local light instead of a single shader for all lights, since at the time of writing the software, we were limited by the number of available temporary GPU registers and multiple short shaders would usually perform better than a single long one.

First, for each pass, i.e. each local light source, the azimuthal distance for each receiver point to the light source position is computed. We use a dedicated shader to compute the distances for each light source to all receiver points and store them to texture using OpenGL framebuffer objects. Each colour component stores one distance between the light sources position and the receiver point.

The implementation of the visibility vector construction for a local light is very similar to the visibility vector construction for the environment lighting. The only difference is that the visibility wedges are defined on the complete sphere instead of the hemisphere. The total visibility vector $\mathbf{V}_{x,y}^{l_i}$ is computed as described in Section 4.3.1 and stored in a 16-component spherical harmonic vector.

To calculate incoming radiance from local light sources, the solid angle subtended by the local light source on the sphere around the receiver point is computed as a function of distance and radius (see Section 4.3). As described in Section 4.3.1, we find the spherical harmonics representation of the incident radiance using a rotated zonal harmonics vector. We compute the zonal harmonic coefficients analytically by evaluating the symbolic integral of zonal harmonics basis functions and incident radiance as a function of the solid angle subtended by the light source. The rotated zonal harmonics vector $\mathbf{L}_{x,y}^{l_i}$ and the visibility SH vector $\mathbf{V}_{x,y}^{l_i}$ are used to compute outgoing radiance at a receiver point.

The outgoing radiance is computed from the environment map and from each local light source in separate passes and rendered to texture using OpenGL framebuffer objects. The contributions from all passes are accumulated in an additional shading pass to render the final shadowing from all light sources. Different lighting setups where tested and are shown in the results chapter.

CHAPTER 6

Results

We tested our method on a computer with an Intel Core i7 3.6Ghz processor, 16 Gb RAM and an NVidia 560 GTX graphics card with 1 Gb RAM. We implemented our method in OpenGL and used GLSL to do computation on the GPU. Frames where rendered at a resolution of 1920x1080.

The tests where performed on six scenes with varying amount of detail. The teddy, city and eggtray are static scenes. The cones scene (both versions), sine grating and mountain scenes have dynamically changing geometry. We used several different lighting configurations including three local volume lights and an environment light in each scene. See the individual scenes for more detailed information. All lights are fully dynamic.

To evaluate the influence of the level step ls and swath number n_φ on the computation time and the quality of the soft shadows, we rendered the cones scene (version two) with different parameter settings (see Figures 6.1 and 6.2). In the cones scene, cones with different sizes are positioned on a circle, and a single light source is positioned at the center of the height field, slightly above the ground. First, we changed only the level step value ls, which adds additional levels to the multi-resolution pyramid, and left the number of partial swaths $n_\varphi = 32$ unchanged (see Table 6.1). In the second test, we increased the number of partial swaths from 8 to 32 and left the level step $ls = 4$ unchanged (see Table 6.2).

We can see that if we increase the number of partial swaths to 32 and the level step ls to 4, we have smooth frame rates of more than 30fps and high-quality soft shadows (see Table 6.2). A loss of quality is noticeable if we set n_φ to 8 or 12 and ls to one or two, which can be seen in Table 6.1. Figures 6.1 and 6.2 compare results from the cones scene (version two).

The scenes in Figures(6.3, 6.4 and 6.5) show all six test scenes with different lighting conditions. The parameter ls was set to 4 and n_φ to 32. Table 6.3 shows the performance in all scenes using only local lights, Table 6.4 shows the performance using local lights and environment lighting. As a reference, Tabel 6.5, we show results from all scenes, rendered without lighting. The timings for each step of our method in all scenes, exept the cones scene, are shown in Table 6.6. Finally, Figure 6.6 compares results applying the different lighting types (local light and environment map) to dynamic height fields.

Cones scene v2				
	$ls = 1$	$ls = 2$	$ls = 3$	$ls = 4$
$n_\varphi = 32$	80fps	62fps	47fps	34fps

Table 6.1: This table shows the Cones scene v2 with one local light with increasing level step value.

Cones scene v2					
	$n_\varphi = 8$	$n_\varphi = 12$	$n_\varphi = 16$	$n_\varphi = 24$	$n_\varphi = 32$
$ls = 4$	94fps	85fps	75fps	62fps	34fps

Table 6.2: This table shows the Cones scene v2 with one local light with increasing number of partial swaths value.

Test scenes rendered without environment lighting			
	1 *Local Light*	2 *Local Lights*	3 *Local Lights*
City	38fps	27fps	21fps
*Cones v*1	41fps	28fps	23fps
Moutains	38fps	27fps	23fps
Eggtray	37fps	27fps	25fps
Sine grating	38fps	26fps	23fps
Teddy	39fps	30fps	23fps

Table 6.3: This table shows scences rendered with different number of local lights in fps.

Test scenes rendered with environment lighting				
	0 *Local Lights*	1 *Local Light*	2 *Local Lights*	3 *Local Lights*
City	113fps	32fps	21fps	15fps
*Cones v*1	112fps	33fps	22fps	17fps
Moutains	102fps	31fps	21fps	15fps
Eggtray	108fps	31fps	21fps	16fps
Sine grating	100fps	32fps	21fps	16fps
Teddy	103fps	33fps	22fps	16fps

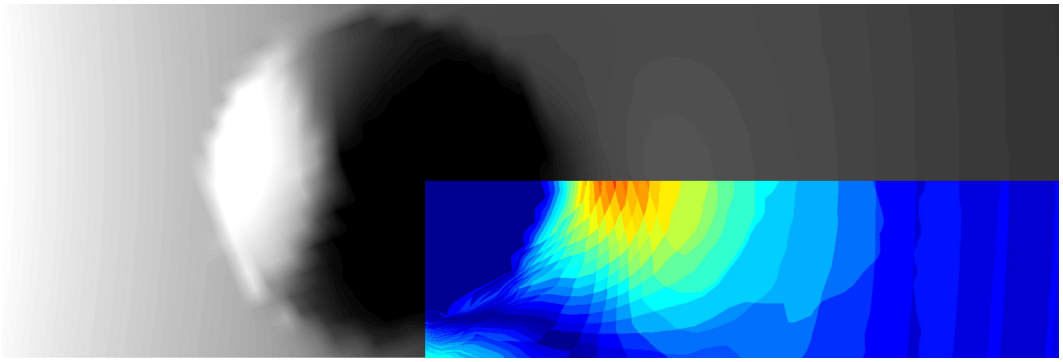
Table 6.4: This table shows scences rendered with different number of local lights and an environment map in fps.

Test scenes rendered without lighting						
	City	*Cones v*1	*Moutains*	*Eggtray*	*Sine grating*	*Teddy*
fps	1060	1125	905	1006	1100	1108

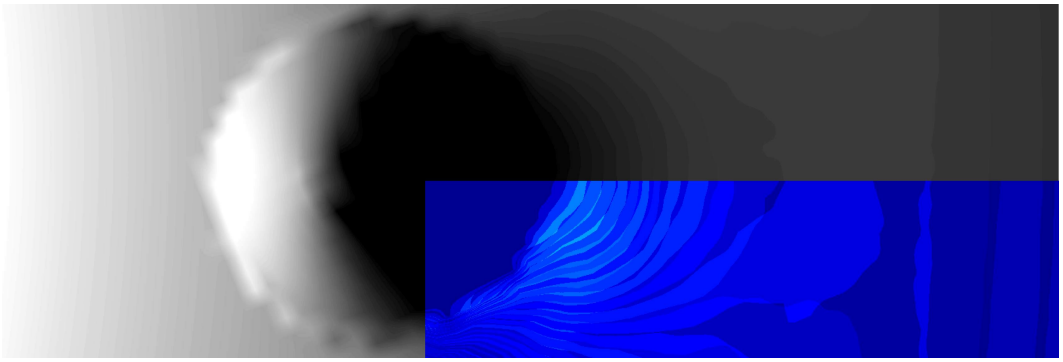
Table 6.5: This table shows the scences from table 6.4, rendered without environment light and local lights (i.e. just standard OpenGL lighting) in fps.

Test scenes time measurements					
	$Pyramid$	$Env.\ Light$	$1st\ L-Light$	$2nd\ L-Light$	$3rd\ L-Light$
$City$	1.71 ms	6.71 ms	16.33 ms	16.19 ms	16.23 ms
$Cones\ v1$	1.75 ms	6.69 ms	15.06 ms	15.47 ms	15.64 ms
$Moutains$	1.72 ms	7.15 ms	15,64 ms	16.49 ms	15.79 ms
$Eggtray$	1.71 ms	7.05 ms	15,8 ms	16.34 ms	16.23 ms
$Sine\ grating$	1.69 ms	7.18 ms	15,96 ms	16.44 ms	16.63 ms
$Teddy$	1.77 ms	6.82 ms	15,74 ms	16.17 ms	16.25 ms

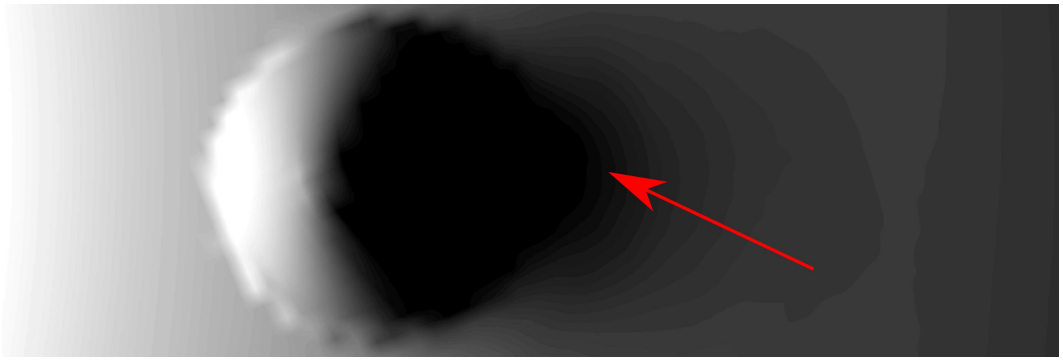
Table 6.6: This table shows the computation times in milliseconds per frame, for each pass and each scene.



(a) $n_\varphi = 4$ and $ls = 4$

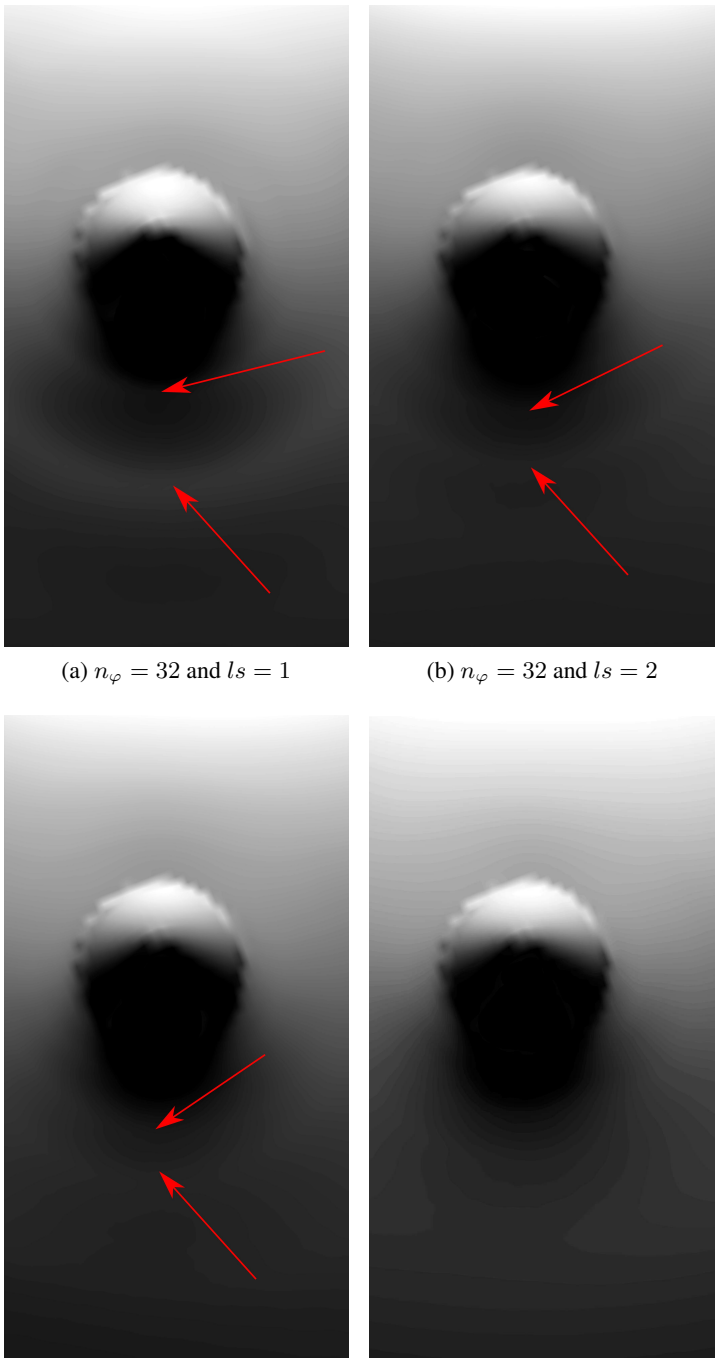


(b) $n_\varphi = 8$ and $ls = 4$



(c) $n_\varphi = 32$ and $ls = 4$

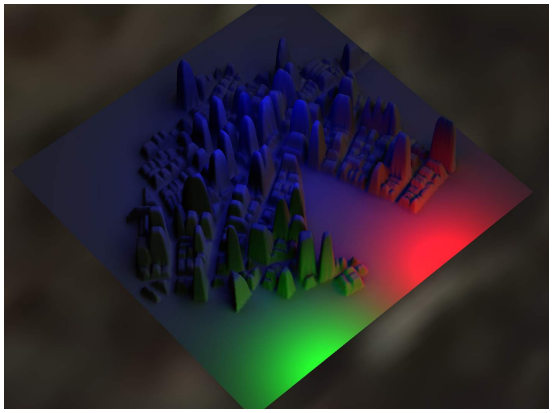
Figure 6.1: Details from the cones scene (version two) rendered with different number of partial swaths. Each image shows one cone, viewed from the top. We can see that we already get nice results with a low number of swaths, due to the smoothing in the height-field pyramid and the low-order SH basis. By taking additional samples we get a more accurate shadow, compared to just 4 samples, which produces no shadow at all. This is because the cone falls between sampling directions. To better distinguish soft-shadows between the images, we increased the contrast, resulting in noticeable quantisation artefacts (lines bewteen colours). The areas in the lower right of the Image a and Image b show the differences with Image c. The red color componet represents the maximum difference, with 0.5 times the maximum brightness of image.



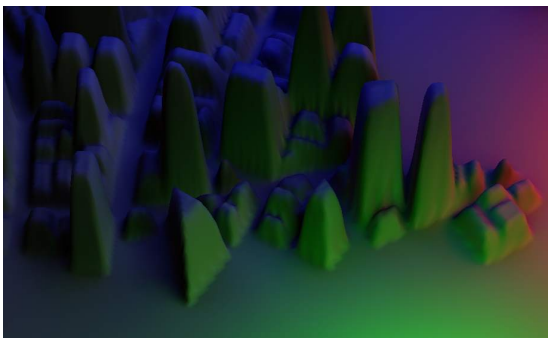
(a) $n_\varphi = 32$ and $ls = 1$ (b) $n_\varphi = 32$ and $ls = 2$

(c) $n_\varphi = 32$ and $ls = 3$ (d) $n_\varphi = 32$ and $ls = 4$

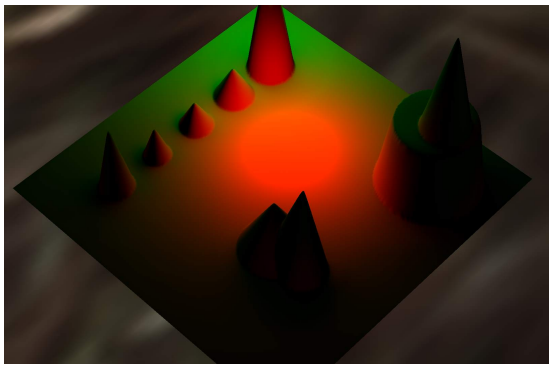
Figure 6.2: Image area snippets from the cones scene (version two) rendered with different number of level steps values. Each image shows one cone, viewed from the top. The red arrows point out artifacts in the shadows. The ringing seen in the shadows arises due to undersampling of the height field in the distance direction. We can see that increasing the sampling density by increasing the level step parameter eliminates the artifacts and results in smooth shadows.



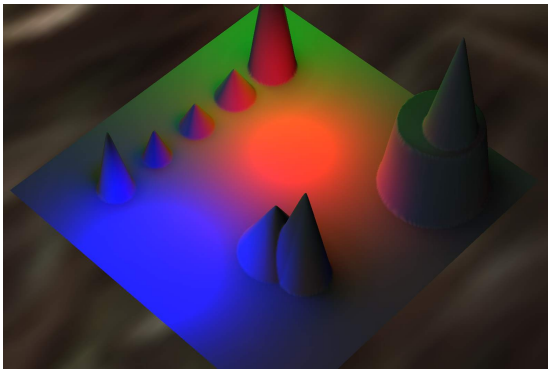
(a) City scene (3 local lights, 1 environment light)



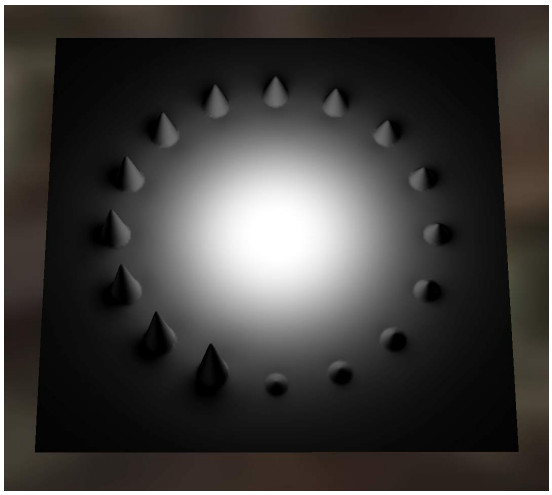
(b) Detailed view of scene 6.3a



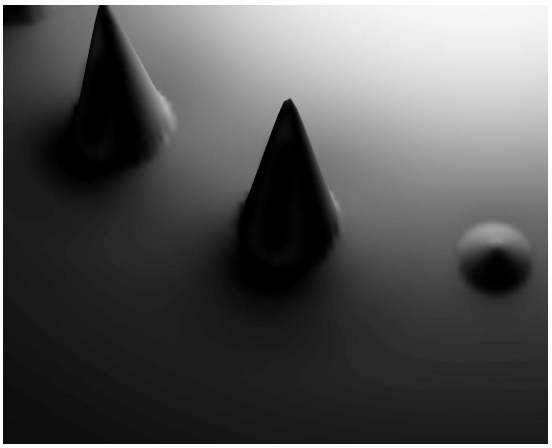
(c) Cones scene V1 (2 local lights)



(d) Cones scene V1 (3 local lights, 1 environment light)

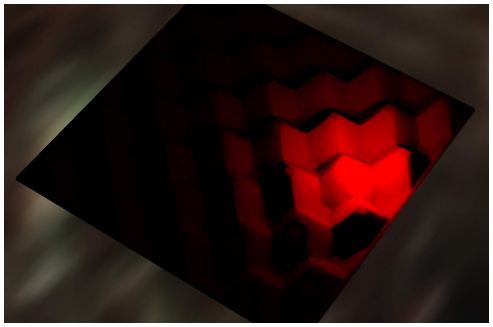


(e) Cones scene V2 (1 local light)

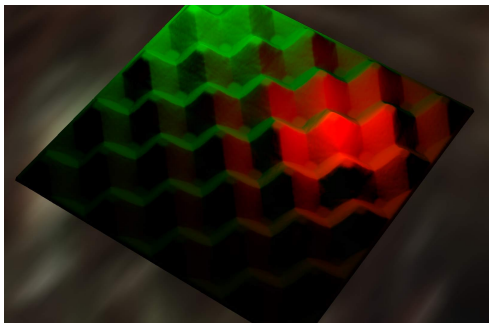


(f) Detailed view of 6.3e

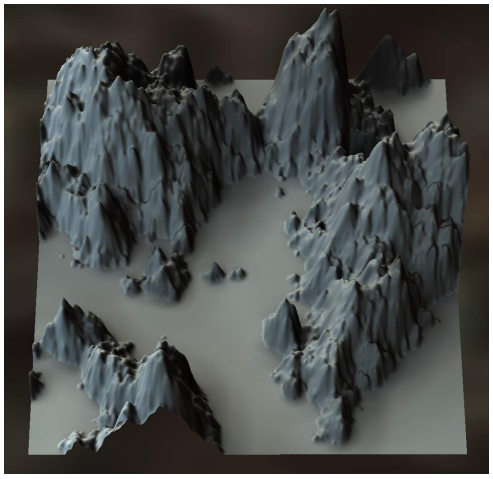
Figure 6.3: A set of images from the city, cones V1 and cones V2 scene: The parameterization for the hightfield pyramid: $ls = 4$, $lo = 4$, number of partial swaths is 32 and the texture resolutin is 256×256. All scenes are dynamic.



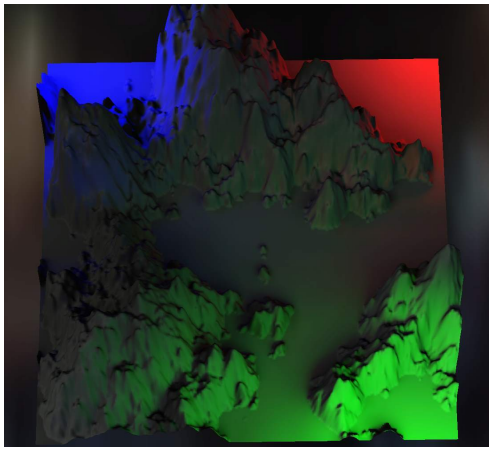
(a) Eggtray scene (1 local light)



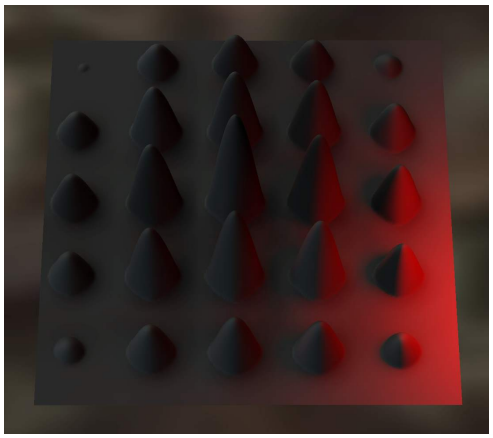
(b) Eggtray scene (2 local lights)



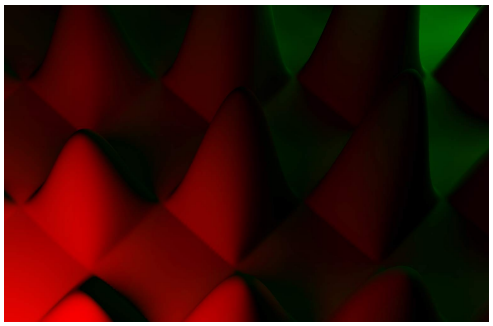
(c) Mountain scene (1 environment light)



(d) Mountain scene (3 local lights, 1 environment light)

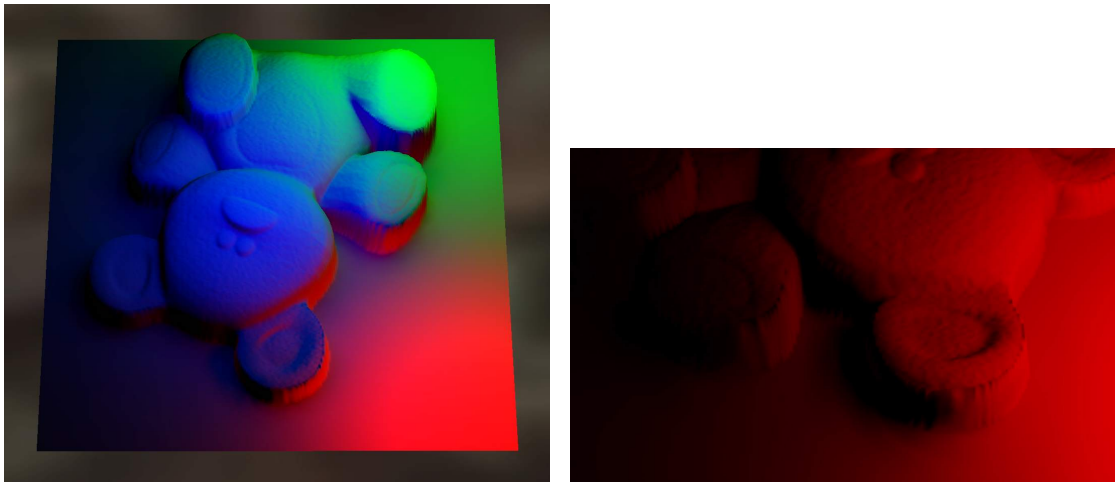


(e) Sine grating scene (1 local light, 1 environment light)



(f) Detail of the sine grating scene (2 local lights)

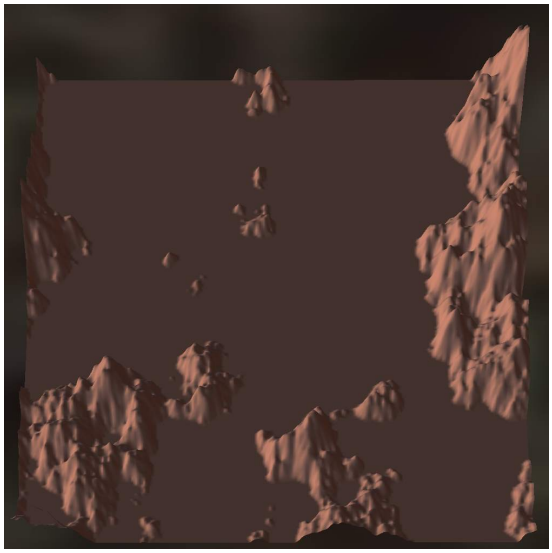
Figure 6.4: A set of images from the eggtray, mountain and sine grating scene: The parameterization for the hightfield pyramid: $ls = 4$, $lo = 4$, the number of partial swaths is 32 and the texture resolutin is $256x256$. All scenes are dynamic.



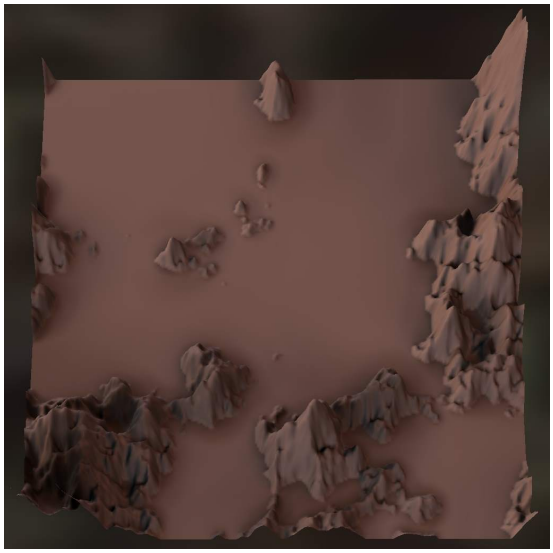
(a) Teddy scene (3 local lights)

(b) Detail of the teddy scene (1 local light)

Figure 6.5: A set of images from the teddy scene: The parameterization for the hightfield pyramid: $ls = 4$, $lo = 4$, number of partial swaths is 32 and the texture resolutin is 256×256. All scenes are dynamic.

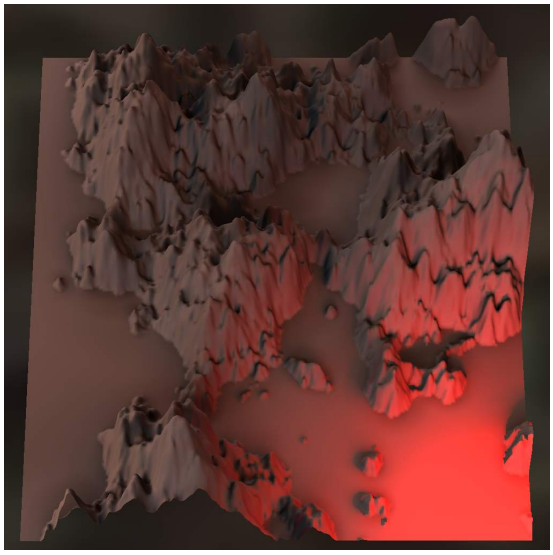


(a) Environment light only - unshadowed



(b) Environment light only - shadowed

(c) Local light only - shadowed)



(d) Local light and environment light - shadowed

Figure 6.6: A set of images from the dynamic height field scene: The top left image shows the scene lit by the environment map without shadows, the top right image shows the shadowed scene using only environment lighting. The bottom left image shows the shadowed scene using one local light only and the bottom right image shows the scene shadowed from the environment light and one local light source. The parameterization for the hightfield pyramid: $ls = 4$, $lo = 4$, number of partial swaths is 32 and the texture resolutin is $256x256$.

CHAPTER 7

Conclusion

In this work, we describe a novel method to calculate soft shadows from local light sources on dynamic height fields. Due to the complexity of computing physically based soft-shadows, most current real-time graphics applications, like video games, suffer from poor soft shadow quality, or they even do not support them. Another limitation seen in video games that include open worlds like mountains is the limitation of shadow casting lights to strong directional light sources like sun light and also infinitely distant light sources. In the special case of height field geometry, simplifying assumptions can be made that allow real-time soft shadows from dynamic geometry.

In this thesis, we have tried to contribute to the development of realistic soft shadows on dynamic height fields. We present a method to approximate realistic soft shadows from local light sources in dynamic height fields by extending the method by Snyder et al. [29]. We achieve real-time frame rates for up to three local lights and an environment light source on consumer graphics hardware.

Several extensions to this method are possible. First, we could precompute the visibility wedge table for different angles between two azimuthal directions as a function of the size of the spherical light source. This would reduce the sample requirement for the complete swath to just a partial swath for a small set of azimuthal directions covering the range of incident light directions from a local light source. Another possible extension is to support diffuse and glossy inter-reflections. Basically, this could be done by creating a multi-resolution pyramid on the radiance and geometry data to reduce sampling overhead or by extending the method by Nowrouzezahrai et al. [19] to local light sources.

The method can also be combined with methods [8, 21] to support shadows cast by more general geometry onto the height field.

Acknowledgements

Many, many thanks to Dipl. Ing. Paul Guerrero supporting me and giving me useful tips. I would also like to thank the technical staff at the Institute of Computer Graphics and Algorithms at the Vienna University of Technology for supporting me with necessary harware to test the method.

APPENDIX A

Zonal Harmonics Coefficients for Local Light Sources

The symbolic integrals for a spherical light source l_i that substends an angle Θ_{l_i} in radians. The first 4 bands, where k is the band index, are:

$$\begin{aligned} k = 0 &: -\sqrt{\pi}\ (-1 + cos(\Theta_{l_i})) \\ k = 1 &: \frac{1}{2}\sqrt{3\pi}\ sin(\Theta_{l_i})^2 \\ k = 2 &: -\frac{1}{2}\sqrt{5\pi}\ cos(\Theta_{l_i})\ (-1 + cos(\Theta_{l_i}))\ (1 + cos(\Theta_{l_i})) \\ k = 3 &: -\frac{1}{8}\sqrt{7\pi}\ (-1 + cos(\Theta_{l_i}))\ (1 + cos(\Theta_{l_i}))\ (-1 + 5cos(\Theta_{l_i})^2) \end{aligned} \tag{A.1}$$

Bibliography

[1] Orthogonal polynomials. Wolfram Mathematica Documentation Center.

[2] *Tutorialcenter: tutorialcenter.net*. http://tutorialcenter.net/create-stylish-sports-car-dashboard-areas-detailed-realism/, 2013. Accessed on April 19, 2013.

[3] *Xdesktopwallpapers: xdesktopwallpapers.com*. http://xdesktopwallpapers.com/christmas-light-house-in-moutains-11510.php, 2013. Accessed on April 19, 2013.

[4] M Blanco. Evaluation of the rotation matrices in the basis of real spherical harmonics of the rotation matrices in the basis of real spherical harmonics. *Journal Of Molecular Structure Theochem*, 419(1-3):19–27, 1997.

[5] Michael Bunnell. *Dynamic Ambient Occlusion And Indirect Lighting*.

[6] Paul Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 189–198, New York, NY, USA, 1998. ACM.

[7] Robin Green. Spherical harmonic lighting: The gritty details. *GDC 2003*, January 2003. Sony Computer Entertainment America.

[8] Paul Guerrero. Approximative real-time soft shadows and diffuse reflections in dynamic scenes. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 10 2007.

[9] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH*, pages 145–154, 1990.

[10] Google Inc. Google earth, March 2013.

[11] Joseph Ivanic and Klaus Ruedenberg. Rotation matrices for real spherical harmonics. direct determination by recursion. *Journal of Physical Chemistry*, 100(15):6342–6347, 1996.

[12] Kei Iwasaki, Wataru Furuya, Yoshinori Dobashi, and Tomoyuki Nishita. Real-time rendering of dynamic scenes under all-frequency lighting using integral spherical gaussian. *Comp. Graph. Forum*, 31(2pt4):727–734, May 2012.

[13] Pat Hanrahan (Stanford University) Henrik Wann Jensen (Stanford University) Don Mitchell Matt Pharr (Exluna) Peter Shirley (University of Utah) James Arvo (Caltech), Marcos Fajardo (ICT/USC). State of the art in monte carlo ray tracing for realistic image synthesis. Course.

[14] James T. Kajiya. The rendering equation. In *SIGGRAPH*, pages 143–150, 1986.

[15] Anton Kaplanyan and Carsten Dachsbacher. Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, I3D '10, pages 99–107, New York, NY, USA, 2010. ACM.

[16] Alexander Keller. Instant radiosity. In *SIGGRAPH*, pages 49–56, 1997.

[17] Jaroslav Křivánek, Jaakko Konttinen, Kadi Bouatouch, Sumanta Pattanaik, and Jiří Žára. Fast approximation to spherical harmonic rotation. In *SCCG '06: Proceedings of the 22nd spring conference on Computer graphics*, pages XXX–XXX, New York, NY, USA, 2005. ACM Press.

[18] Martin Mittring. Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 97–121, New York, NY, USA, 2007. ACM.

[19] Derek Nowrouzezahrai and John Snyder. Fast global illumination on dynamic height fields. *Computer Graphics Forum: Eurographics Symposium on Rendering*, June 2009.

[20] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 497–500, New York, NY, USA, 2001. ACM.

[21] Zhong Ren, Rui Wang, John Snyder, Kun Zhou, Xinguo Liu, Bo Sun, Peter-Pike Sloan, Hujun Bao, Qunsheng Peng, and Baining Guo. Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. pages 977–986, New York, NY, USA, 2006. ACM. SIGGRAPH 2006.

[22] Tomas Sakalauskas. Hybrid terrain shadow ray casting. 2008.

[23] Volker Schönefeld. Spherical harmonics, July 2005.

[24] C. Sigg and M. Hadwiger. Fast third-order texture filtering. In *GPU Gems 2*, chapter 20. 2005.

[25] François X. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):240–254, September 1995.

[26] Peter-Pike Sloan. Stupid spherical harmonics (sh) tricks, 2008.

[27] Peter-Pike J. Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH*, pages 527–536, 2002.

[28] John Snyder. Code generation and factoring for fast evaluation of low-order spherical harmonic products and squares, 2 2006.

[29] John Snyder and Derek Nowrouzezahrai. Fast soft self-shadowing on dynamic height fields. *Computer Graphics Forum: Eurographics Symposium on Rendering*, 27(4):1275–1283, June 2008.

[30] Gabor Szegö. Orthogonal polynomials. *Colloquim Publications*, 23.

[31] Ville Timonen and Jan Westerholm. Scalable height field self-shadowing. *Computer Graphics Forum (Proceedings of Eurographics 2010)*, 29(2), May 2010.