Diffraction Shader

Bachelorarbeit

der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Michael Single

2014

Leiter der Arbeit:
Prof. Dr. Matthias Zwicker
Institut für Informatik und angewandte Mathematik

Abstract

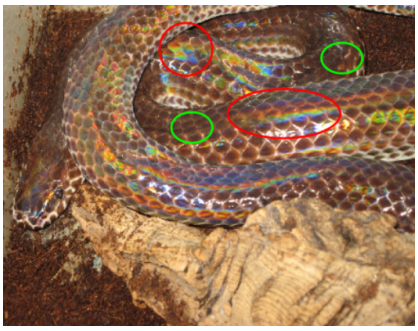
Inhaltsverzeichnis

Kapitel 1

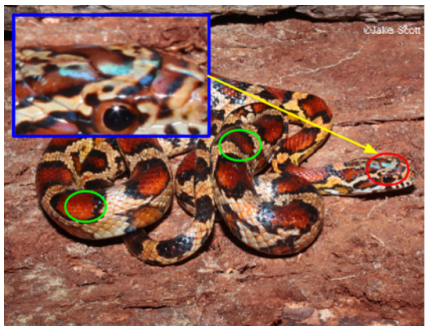
Introduction

1.1 Motivation

As a human being, we visually perceive and experience our whole world in terms of colors, resulting from various kinds of physical light interaction phenomenons. Particularly in biology, there are basically two main causes for color production. Firstly, due to pigmentation, which occures since certain molecules in a biological structure selectively absorb or reflect specific wavelengths from an incident light source. And secondly because of structural colors which are the result of physical interaction of light with a nanostructure, exclusively relying on the structring of the material and not any other property. One particular example for such biological color production are the colors we can see when having a closer look at the illuminated skin of snake species, as shown in figure 1.1.



(a) Xenopeltis snake



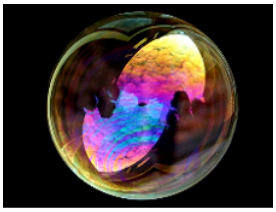
(b) Elaphe Guttata snake

Abbildung 1.1: Examples of pigmentation color (green circles) and structural color (red circles) on snake species[1].

Comparing both snake species with each other, we observe that the Xenopeltis species expresses structural colors in form of iridescent patterns along its scales way stronger than the Elaphe species does. The reason for this lies on the

[1] image source of figure 1.1(a) `http://www.snakes-alive.co.uk/gallery_5.html` and figure 1.1(b) `http://www.the-livingrainforest.co.uk/living/view_price.php?id=464`

nanostructure of their skins. A natural diffraction grating is a semitransparent layer of biological nanostructures which exhibits a certain degree of regularity to produce structural colors by diffracting an incident light source. There are a vast amount of additional reasons for producing structural colors in nature, such as thin film interference, intra-cellular photonic crystyls or diffraction gratings. More detailed examples are listed in figure 1.2.



(a) Thin Film Inteference soapbubble

(b) Multilayer Interference on abdomen of beetle

(c) Photonic crystals in Wings of buttefly

(d) Scattering of light on butterfly wing

(e) Artifical Diffraction Grating on CD

(f) Natural Diffraction Grating on snake skin

Abbildung 1.2: Examples[2] for structural colors on the wings and the abdomen of insects, liquids, synthetic structures, and on scales on the skin of reptiles.

Already in the 17th century Robert Hooke was able to relate the cause of structural colors to the the microstructure of a material. During his examinations of peacock feathers he found that the colors could be made disappear by wetting the feathers and further observerd that the feathers are made of tiny ridges. Building on top of the latest knowledge about interference, Newton related structural colors with wave interference. Nowadays, in the field of computer graphics, many researchers have been attempted rendering structural colors but unforunately, most currently available approaches are not able to perfrom interactive rendering or are oversimplified and thus cannot model accurately the effect of diffraction.

[2]image source of figure:

- 1.2(a): `http://www.ualberta.ca/~pogosyan/teaching/PHYS_130/FALL_2010/lectures/lect33/lecture33.html`
- 1.2(b): `http://www.itp.uni-hannover.de/~zawischa/ITP/multibeam.html`
- 1.2(c): `http://upload.wikimedia.org/wikipedia/commons/a/a4/Parides_sesostris_MHNT_dos.jpg`
- 1.2(d): From paper [al.10], figure 6.
- 1.2(e): `http://cnx.org/content/m42496/latest/?collection=col11428/latest`
- 1.2(f): `http://www.snakes-alive.co.uk/gallery_5.html`

1.2 Goals

The purpose of this thesis is firstly, to simulate physically accurate structural colors caused by the effect of diffraction on various biological structures and secondly implement this simulation as a renderer with interactive behavior. We mainly focus on structural colors generated by natural diffraction gratings. In particular the approach presented in this thesis applies to surfaces with quasiperiodic structures at the nanometer scale which can be represented as heighfields stored in gray-scale images.

Natural gratings like this are found on the scale of reptiles, wings of butterflies or the bodies of various insects but we restrict ourself and focus on snake skins. The heightfield data of used snake skins are acquired using atomic force microscopy and provided by LANE lab at Genava[3].

The renderer discussed in this thesis is based on the pioneering work of J. Stam about diffraction shaders [Sta99] in which he formulated a BRDF modeling the effect of diffraction. Nevertheless we have to adapt his BRDF model since his model assumes, that a given surface of a grating can either be formulated by an analytical function, and has therefore an closed form solution or is that simple that it satifies relying on statistical methods in order to model the surface. However, in our case, we are dealing with natural diffraction grating, given as an explicit heightfield. Hence we neither can rely on image statistics, since the details of natural gratings are to complex nor are we able to formulate an closed form solution. Therefore, this results of this thesis can be considered as an extension of J. Stam's work for the case someone is provided by a explicit height field of a quasiperiodic surface.

1.3 Previous work

stam, hooke, see our paper, see stams paper, see own research.

Robert Hooke = observed connection between microscopic structures and colorisation wave nature of light led to conclusion that the cause for the coloration lies in wave interference.

previous

In computer graphics literature, Stam was the first to develop reflection models based on wave optics called diffraction shaders, that can produce colorful diffraction effects. His approach is based on a far field approximation of the Kirchhof integral. He shows that for surfaces representeted as nanoscale heightfieds it is possible to derive their BRDF as the Fourier transformation of a function of the heightfield. Nevetheless, this formulation is not immediately useful for efficent rendering of measured complex nanostructures since this would require the on-the-fly evaluation of and and integration over Fourier transforms of the heightfield that depend on the light and viewing geometry. In his derivations, Stam models heightfields as probabilistic superpositions of bumps forming periodic like structures. This provides him an analytical identity for this class of heightfields. However, boplogocal nanostructures are way more complex and do not lend themself to this simplified statistical model.

follow ups

[3]Laboratory of Artificial and Natural Evolition. Website:`www.lanevol.org`

1.4 Overview

The reminder of this thesis is organized as the follows: due to the fact that this thesis has a rather advanced mathematical complexity the first part of chapter 2 introduces some important definitions which are required in order to be able to follow the derivations in the last third of chapter 2. Before starting the derivations, a brief summary of J. Stam's Paper about diffraction shaders is provided since this whole thesis is based on his BRDF representation. Our derivations itself are listed step-wiese, whereas there is a final representation provided by the end of chapter 2. Chapter 3 addresses the practical part of this thesis, the implementation of our diffraction model, explaining all precomputation steps and how rendering is preformed in our developed framework for this thesis. Chapter 4 gives some further insight about diffraction by explaining the topic about diffraction grating in depth. Furthermore, within this chapter we evaluates the qualitative validity of our BRDF models applied on different surface gratings by computing their reflectance and comparing this to the grating equation under similar conditions. Chapter 5 presents our rendered results, first the so called BRDF maps for all our gratings and shading approaches under various shading parameters and then the actual renderings on a snake mesh. Chapter 6 contains the conclusion of this thesis which starts by a review briefly discussing what has been achieved in this thesis and the drawbacks. There are also some words about my personal experience during this thesis.

Kapitel 2

Theoretical Background

2.1 Basics in Modeling Light in Computer Graphics

2.1.1 Radiometry

One purpose of Computer Graphics is to simulate the interaction of light on a surface and how a real-world observer, such as a human eye, will perceive this. These visual sensations of an eye are modeled relying on a virtual camera which captures the emitted light from the surface. The physical basis to measure such reflected light depicts radiometry which is about measuring the electromagnetic radiation transfered from a soruce to a receiver.

Fundamentally, light is a form of energy propagation, consisting of a large collection of photons, whereat each photon can be considered as a quantum of light that has a position, direction of propagation and a wavelength λ. A photon travels at a certain speed $v = \frac{c}{n}$, that depends only the speed of light c and the refractive index n through which it prograte. Its frequency is defined by $f = \frac{v}{\lambda}$ and its carried amount of energy q, mearsured in the SI unit Joule, is given by $q = hf = \frac{hv}{\lambda n}$ where h is the Plank's constant. The total energy of a large collection of photons is hence $Q = \sum_i q_i$.

2.1.2 Spectral Energy

It is important to understand that the human eye is not equally sensitive to all wavelength of the spectrum of light and therefore responds differently to specific wavelengths. Remember that our goal is to model the human visual perception. This is why we consider the energy distribution of a light spectrum rather than considering the total energry of a photon collection since then we could weight the distribution according the human visual system. So the question we want to answer is: How is the energy distributed across wavelengths of light?

The idea is to make an energy histrogram from a given photon collection. For this we have to order all photons by their associated wavelength, discretize wavelength spectrum, count all photons which then will fall in same wavelength-interval, and then, finally, normalize each interval by the total energy Q. This will give us a histogram which tells us the spectral energy Q_λ for a given discrete

λ interval and thus models the so called spectral energy distribution [1].

2.1.3 Spectral Power

Rendering an image in Computer Graphics corresponds to capturing the color sensation of an illuminated, target scene at a certain point in time. As previousely seen, each color is associated by a wavelength and is directly related to a certain amount of enegry. In order to determine the color of a to-be-rendered pixel of an image, we have to get a sense of how much light (in terms of energy) passes through the area which the pixel corresponds to. One possibility is to consider the flow of energy $\Phi = \frac{\Delta Q}{\Delta t}$ transferred through this area over a small period of time. This allows us to measure the energy flow through a pixel during a certain amount of time.

In general, power is the estimated rate of energy production for light sources and corresponds to the flux. It is measured in the unit Watts, denoted by Q. Since power is a rate over time, it is well defined even when energy production is varying over time. As with Spectral Energy for rendering, we are really interested in the spectral power $\Phi_\lambda = \frac{\Phi}{\lambda}$, measured in Watts per nanometer.

2.1.4 Spectral Irradiance

Before we can tell how much light is reflected from a given point on a surface towards the viewing direction of an observer, we first have to know how much light arrives at this point. Since in general a point has no length, area or even volume associated, let us instead consider an infinitimal area ΔA around a such a point. Then, we can ask ourself how much light falls in such a small area. When further observing this process over a short period in time, this quantity is the spectral irradiance E as illustrated in figure 2.1. Summarized, this quantity tells us how much spectral power is incident on a surface per unit area and mathematically is equal:

$$E = \frac{\Phi_\lambda}{\Delta A} \tag{2.1}$$

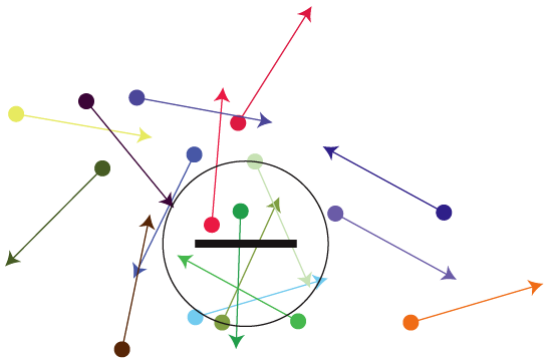
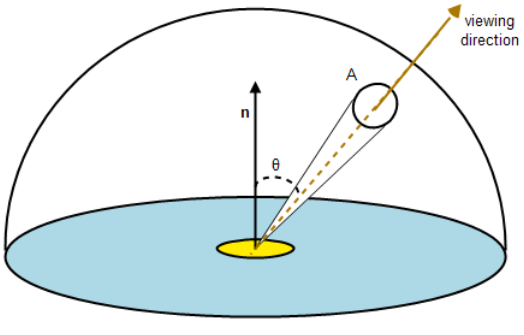


Abbildung 2.1: Irradiance is the summed up radiance over all directions

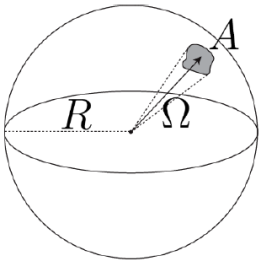
[1] Intensive quantities can be thought of as density functions that tell the density of an extensive quantity at an infinitesimal point.

2.1.5 Spectral Radiance

When rendering an image we have to determine the color of each pixel of the image. Although irradiance tells us how much light is arriving at a point as illustrated in figure 2.1, it tells us little about the direction that light comes from. This relates to how the human eye perceives the brightness of an illuminated objects when looking at it in a certain direction.



(a) Radiances is the density of photons per area per solid angle



(b) Solid angle is the area of a surface patch on a sphere with radius R which is spanned by a set of directions

This concepted is described by the radiometric quantity radiance. Basically, this is a measure of light energy passing through or is emitted off from a small area around a point on a surface towards a given direction during a short period in time. More formally this is the spectral power emerging from an arbitrary point (an infinitimal area around this point) and falls within a given solid angle (see figure[2] $2.2(b)$)in specific direction (usually towards the observer) as shown in figure $2.2(a)$. Formally, this leads us to the following mathematical formalism:

$$L_\lambda(\omega) = \frac{d^2\Phi_\lambda}{dAd\Omega} \approx \frac{\Phi_\lambda}{\Omega A} \tag{2.2}$$

where L is the observed spectral radiance in the unit energy per unit area per solid angle, which is $Wm^{-}2sr^{-}1$ in direction ω which has an angle θ between the surface normal and ω, Θ is the total flux or power emitted, θ is the angle between the surface normal and the specified direction, A is the area of the surface and Ω is the solid angle in the unit steradian subtended by the observation or measurement.

It is useful to distinguish between radiance incident at a point on a surface and excitant from that point. Terms for these concepts sometimes used in the graphics literature are surface radiance L_r for the radiance *reflected* from a surface and field radiance L_i for the radiance *incident* at a surface.

2.1.6 BRDF

In order to render the colorization of an observed object, a natural question in computer graphics is what portion of the reflected, incident light a viewer will

[2]Similar figute like used in computer graphics class 2012 in chapter colors

receive, when he looks at an illuminated object. Therefore for any given surfaces which is illuminated from a certain direction ω_i, we can ask ourself how much light is reflected off of any point on this surface towards a viewing direction ω_r. This is where the Bidirectional Reflectance Distribution Function (short: BRDF) comes into play, which is a radiometric quantity telling us how much light is reflected at an opaque surface. Mathematically speaking, the BRDF is the ratio of the reflected radiance pointing to the direction ω_r to the incident irradiance comming from the inverse direction of ω_i as illustrated in figure 2.2. Hence the BRDF is a four dimensional function defined by four angles θ_i, ϕ_i, θ_r and ϕ_r.

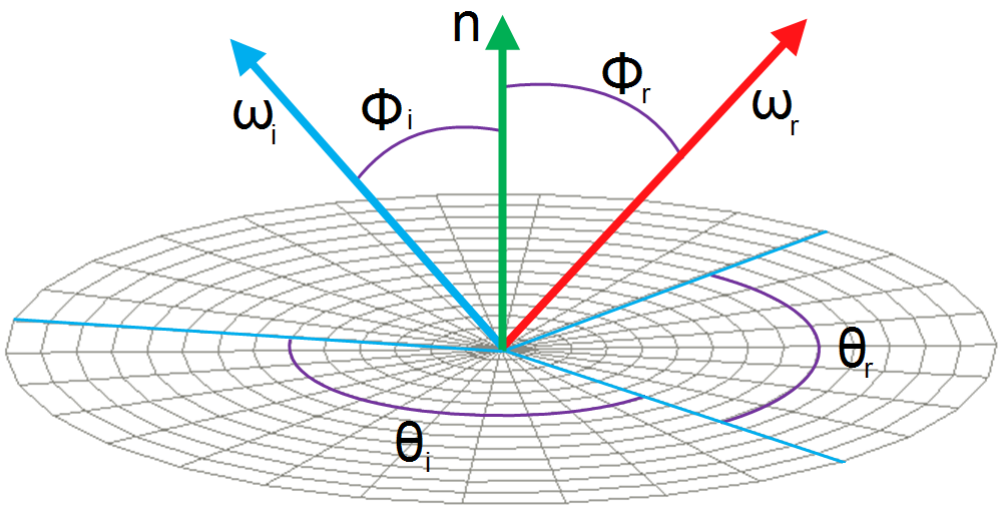


Abbildung 2.2: Illistration of the BRDF model, where ω_i is pointing to the light source and the existing direction is denoted by ω_r. Both direction unit direction vectors defined w.r.t to a surface normal **n** for every point on the surface.

Which formally is for any given wavelength λ equivalent to:

$$\begin{aligned} BRDF_\lambda(\omega_i, \omega_r) &= \frac{dL_r(\omega_r)}{dE_i(\omega_i)} \\ &= \frac{dL_r(\omega_r)}{L_i(\omega_i)cos(\theta_i)d\omega_i} \end{aligned} \tag{2.3}$$

Where L_r is the reflected spectral radiance, E_i is the spectral irradiance and θ_i is the angle between ω_i and the surface normal **n**.

2.1.7 Wavespectrum and Colors

In order to see how crucial the role of human vision plays, let us consider the following definition of color by *Wyszechkiu and Siles*[3] stating that *Color is the aspect of visual perception by which an observer may distinguish differences*

[3]mentioned in Computer Graphics Fundamentals Book from the year 2000

between two structure-free fields of view of the same size and shape such as may be caused by differences in the spectral composition of the radiant energy concerned in the observation. Therefore, similarly like the humans' perceived sensation of smell and taste, color vision is just another individual sense of perception giving us the ability to distinguish different frequency distribution of light experienced as color.

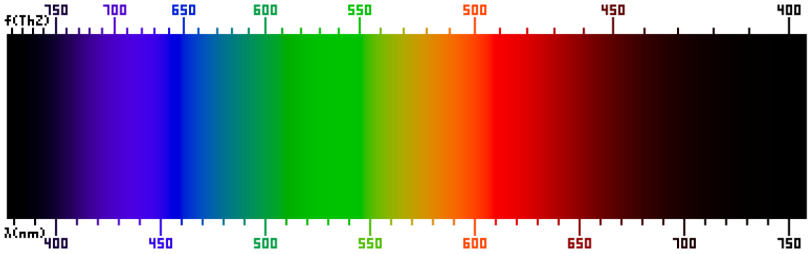


Abbildung 2.3: Frequency (top) and wavelenght (bottom) of colors of the visible light spectrum[4].

In general an eye consists of photoreceptor cells which are responsible for providing ability of color-perception. A schematic of an eye is illustratied in figure 2.4. Basically, there are two specialized types of photoreceptor cells, cone cells which are responsible for color vision and rod cells, which allow an eye to perceive different brigthness levels.

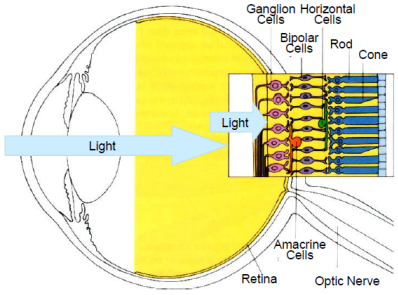


Abbildung 2.4: Schematic[5] of photoreceptor cells, cones and rods, in human eye

A human eye is made of three different types of cone cells, having their peak sensivity in sensing color at different wavelength ranges. More precisely, there are cone cells most sensitive to short wavelengths which are between $420nm$ and $440nm$, those which are most sensitive in the middle range between $530nm$ and $550nm$ and those which have their peak in the long range, from $560nm$ to $580nm$. In principle, any color sensation in human color perception as shown in figure 2.3 can therefore be described by just three parameters, corresponding to levels of stimulus of the three types of cone cells.

[4]Similar figute like used in computer graphics class 2012 in chapter colors
[5]image of illustration has been taken from wikipedia

2.1.8 Colorspace

In order to render accurately images of how a human observer sees its world, a mathematical model of the human color perception is required. Remember that color sensation is due to a visual stimulus processed by cone cells in an eye. A human eye contains three different types of cone cells. Therefore, one possible approach is to describe each kind of these cone cells as a function of wavelength, returning a certain intensivity. In the early 1920, from a series of experiments the so called CIE XYZ color space was derived, describing response of cone cells of an average human individual, the so called standard observer. Basically, a statistically sufficiently large number of probands were exposed to different target light colors expressed by their wavelength. The task of each proband was to reproduce these target colors by mixing three given primary colors, red-, green- and blue-light. The strength of each primary color could be manually adjusted by setting their relative intensivity. Those adjustment weights have been measured, aggregated and averaged among all probands for each primary color. This model describes each color as a triple of three real valued numbers[6], the so called tristimulus values.

Pragmatically speaking, color spaces describes the range of colors a camera can see, a printer can print or a monitor can display. Thus, formally we can define it as a mapping a range of physically produced colors from mixed light to an objective description of color sensations registered in the eye of an observer in terms of tristimulus values.

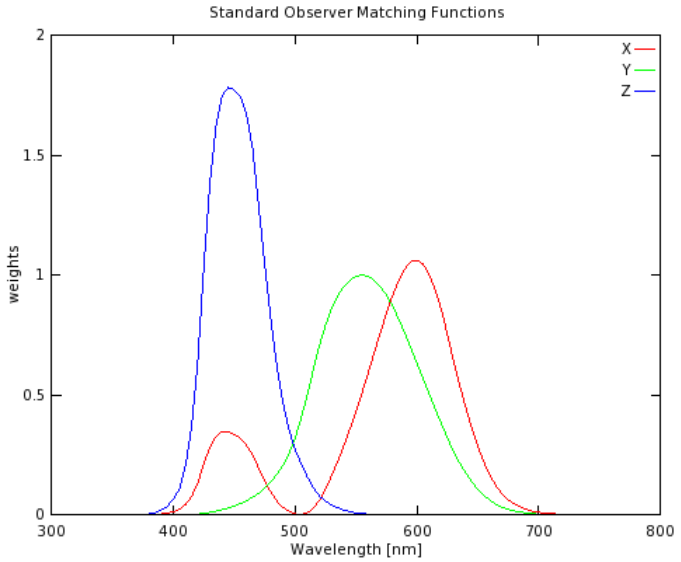


Abbildung 2.5: Plots of our color matching functions we used for rendering

[6]note that there are negative color weights possible in the CIE XYZ colors space. This is why some human perceived color sensations could not be reconstructed using just an additive color model (adding three positively weighted primary values). Therefore, a probabant was also allowed to move one of the primary colors to the target color and instead was supposed to reproduce this new color mix using the two remaining primaries (subtractive model). The value of the selected, moved primary was then interpreted as beeing negative weighted in an additive color model.

Interpolating all measured tristimuli values gives us three basis functions, the CIE color matching functions $\overline{x}(\lambda)$, $\overline{y}(\lambda)$, $\overline{z}(\lambda)$. In figure 2.5 are the numerical description of the chromatic response of the observer. They can be thought of as the spectral sensitivity curves of three linear light detectors yielding the CIE Tristimulus values X, Y and Z.

The tristimulus values for a color with a spectral power distribution $I(\lambda)$, are given in terms of the standard observer by:

$$\begin{aligned} X &= \int_{\Lambda} I(\lambda)\,\overline{x}(\lambda)\,d\lambda \\ Y &= \int_{\Lambda} I(\lambda)\,\overline{y}(\lambda)\,d\lambda \\ Z &= \int_{\Lambda} I(\lambda)\,\overline{z}(\lambda)\,d\lambda \end{aligned} \tag{2.4}$$

Where λ, is the wavelength of the equivalent monochromatic light spectrum $\Lambda = [380nm, 780nm]$. Note taht it is not possible to build a display that corresponds to the CIE XYZ colorspace. For this reasons it is necessary to design other color spaces, which are physical realizable, offers efficient encoding, are perceptual uniform and have an intuitive color specification. There are simple conversions between XYZ color space, to other color space described as linear transformations.

2.1.9 Spectral Rendering

When rendering an image, most of the time we are using colors described in a certain RGB color space. However, a RGB colorspace results from a colorspace transformation of the tristimulus values, which themself are inherent to the human visual system. Therefore, many physically light phenomenon are poorly modeled when always relying on RGB colors for rendering. Using only RGB colors for rendering is alike we would assume that a given light source emits light of only one particular wavelength. But in reality this is barely the case. Spectral rendering is referring to use a certain wavelength spectrum, e.g. the human visible light spectrum, instead simply using the whole range of RGB values in order to render an illuminated scene. This captures the physical reality of specific light sources way more accurate. Keep in mind that, even when we make use of a spectral rendering approach, we have to convert the final spectra to RGB values, when we want to display an image on an actual display.

2.2 Wave Theory for Light and Diffraction

2.2.1 Basics in Wave Theory

In order prepare the reader for physical relevant concepts used during later derivations and reasonings within this thesis, I am going to provide a quick introducation to the fundamental basics of wave theory and related concepts. In physics a wave describes a disturbance that travels from one location to another through a certain medium. The disturbance temporarly displaces the particles in the medium from their rest position which results in an energy transport

along the medium during wave propagation. Usually, when talking about waves we are actually refering to a complex valued function which is a solution to the so called wave equation which is modeling how the wave disturbance proceeds in space during time.

There are two types of waves, mechanical waves which deform their mediums during propagation like sound waves and electromagnetic waves consiting of periodic oscilations of an electromagnetic field such as light for example. Like simplified illustrated in figure 2.6, there are several properties someone can use and apply in order to compare and distinguish different waves:

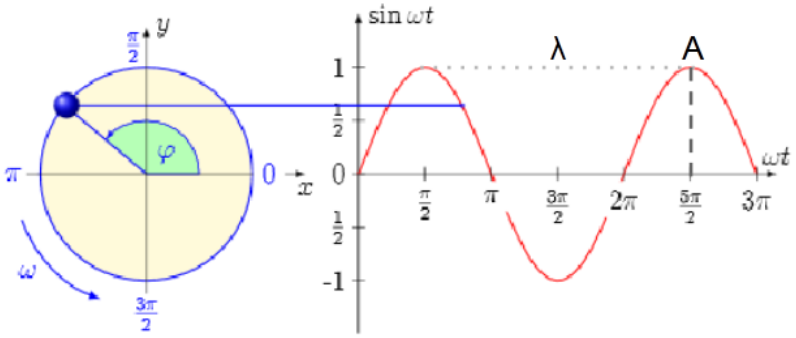


Abbildung 2.6: Simplified, one dimensionaly real valued wave function[7], giving an idea about some important wave properties. We denote the crest of a wave as the hightest point relative to the equilibrium line (zero height along time axis) and similarly the trough as the lowest point.

Wavelength: Is usually denoted by λ and is a measure for the spatial distance from one point to another until the shape of a wave repeats

Amplitude: Is denoted by A and there are two possible interprations: First, it is a measure of the height from the equilibrium point to the heighest point of a crest on the wave or the lowest point of a trough. This mean the amplitude can be positive or negative. However, usually, someone is just interested the absolute value of an amplitude, the magnitude of a wave. For light waves it is a relative measure of intensity or brightnes to other lught waves of the same wavelength. And secondly, it can be interpreted as a measure how much energy a wave carries wherate the greater the absoulte aplitute value, the bigger the amount of energy being carried.

Frequency: Is a measure of the number of waves which are passing through a particular point in the propagation medium during a certain time and is denoted by f.

Phase: Is denoted by ϕ. Describes either the offset of initial position of a wave or the relative displacement between or among waves having the same frequency. Two waves two waves with same frequency are denoted by being in phase if they have the same phase. This means they line up everywhere. As a remark, we denote by ω the angular frequency which is equal $2\pi f$.

[7]Image source: http://neutrino.ethz.ch/Vorlesung/FS2013/index.php/vorlesungsskript

A geometrical property of waves is their wavefront. This is either a surface or line along the path of wave propagation on which the disturbance at every point has the same phase. Three are basically three types of wavefronts: spherical-, cylindrical- and plane wavefront. If point in a isotropic medium is sending out waves in three dimensions, then the coresponding wavefronts are spheres, centered on the source point. Hence spherical wavefront is the result of a spherical wave, also denoted as a wavelet. Note that for electromagnetic waves, the phase is a poisition of a point in time on a wavefront cycle (motion of wave over a whole wavelength) whereat a complete cycle is defined as being equal 360 degrees.

2.2.2 Wave Interference

Next, after having seen that a wave is simply a traveling disturbance along a medium, having some special properties, someone could ask what happens when there are several waves travaling on the same medium. Especially, we are interested how these waves will interact with each other. In physics the term interference denotes the interaction of waves when they encounter each other at a point along their propagation medium. At each point where two waves superpose, their total displacement at these points is the sum of the displacements of each indiviudal wave at those points. Then, the resulting wave is having a greater or lower amplitude than each seperate wave and this we can interprete the interference as the addition operator for waves. Two extreme scenarios are illustrated in figure 2.7. There are basically three variants of interferences which can occur, depending on how crest and troughs of the waves are matched up:

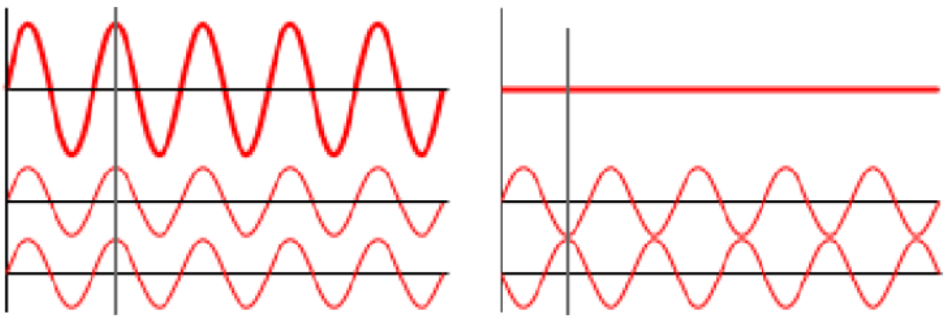


Abbildung 2.7: Interference scenarios[8] when two waves waves meet: On the left handside, there is constructive interference and on the right handside there is destructive interference illustrated.

- Either a crest of a wave meets a crest of another wave or similarly a trough meets a trough of another wave. This scenario is denoted as constructive interference and occurs at any location along the medium where the two interfering waves have a displacement in the same direction. This is equivalent like saying that the phase difference between the waves is a multiple of 2π. Then the resulting amplitude at that point is being much larger

[8]Image source: `http://en.wikipedia.org/wiki/Interference_(wave_propagation)`

than the amplitude of an individual wave. For two waves with an equal amplitude interfering constructively, the resulting amplitude is twice as large as the amplitude of an individual wave.

- Either a crest of a wave meets a trough of another wave or vice versa. This scenario is denoted as destructive interference and occurs at any location along the medium where the two interfering waves have a displacement in the opposite direction. This is like saying that the phase difference between the waves is an odd multiple of π. Then the waves completely cancel each other out at any point they superimpose.
- If the phase difference between two waves is intermediate between the first two scenarios, then the magnitude of the displacement lies between the minimal and maximal values which we could get from constructive interference.

Keep in mind that when two or more waves interfere which each other, the resulting wave will have a different frequency. For a wave, having a different frequency also means having a different wavelength. Therefore, this directly implies that a light of a different color, than its source waves have, is emitted.

2.2.3 Wave Coherence

Concept of Coherence

When considering waves which are traveling on a shared medium along the same direction, we could examine how their phase difference is changing over time. Formulating the changement of their relative phase as a function of time will provides us a quantitative measure of the synchronism of two waves, the so called wave coherence. In order to better understand this concept, let us consider a perfectly mathematical sine wave and second wave which is a phase-shifted replica of the first one. A property of mathematical waves is that they keep their shape over an infinity amount of moved wavelengths. In our scenario, both waves are traveling along the same direction on the same medium, like exemplarily illustrated in figure 2.8.

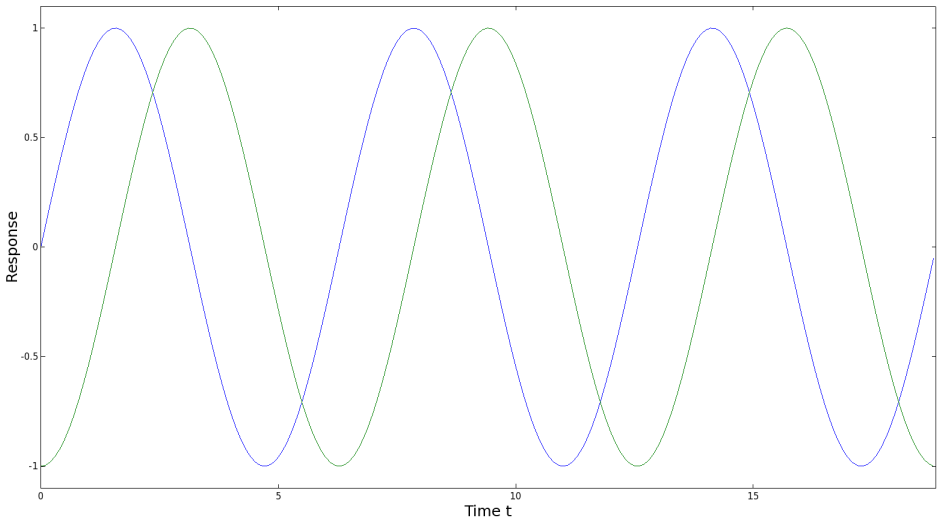


Abbildung 2.8: Two mathematical sine waves which are perfectly coherent which means that their phase difference is constant for every point in time.

Taking the difference between the two sine waves from the previous figure yields always a constant number. Therefore, those two waves are said to be coherent and hence perfectly synchronous over time. Notice that this scenario is completly artificial since in nature there are no mathematical sine waves. Rather, the phase difference is then a fuction of time $p(t)$. The more coherent two waves are, the slower this function will change over time. In fact, two waves are said to be coherent if they are either of the same frequency, temporally in phase or have the same amplitude at every point in time. Thus two waves are coherent if they are generated at the same time, having the same frequency, amplitude, and phase. Reversely, Waves are considered incoherent or also asynchronous if they have no stable phase difference. This means $p(t)$ is heavly varying over time. Coherence describes the effect of whether waves will tend to interfere with each other constructively or destructively at a certain point in time and space. Thus this is a property of waves that enables stationary interference. The more correlated two waves are, the higher their degree of coherence is. In physics coherence between waves is quantified by the corss-correlation function, which basically predicts the value of a second wave using the value of the first one. There are two basic coherence classifications:

- Spatial coherence is dealing with the question of what is the range of distance between two points in space in the extend of a wave for which there is occuring a significant effect of interference when averaged over time. This is formally answered by considering the correlation between waves at different point in space. The range of distance is also denoted as the coherence area.
- Temporal coherence examines the ability of how well a wave will interfer with itself at different moments in time. Mathematically, this kind of coherence is computed by averaging the measured correlation between the value of the wave and the delayed version of itself at different pairs of

time. The Coherence time denotes the time for which the propagating wave is coherent and we therefore can predict its phase using the correlation function. The distance a wave has traveled during the coherence time is denoted as the coherence length.

Derivation

inject contribution here

2.2.4 Huygen's Principle

Besides from a wave's phase and amplitude, also its propagation directly affects the interaction between different waves and how they could interfere with each other. This is why it makes sense to formulate a model which allow us to predict the position of a moving wavefront and how it moves in space. This is where Huygen's Principle comes into play. It states that any each point of a wavefront may be regarded as a point source that emits spherical wavelets in every direction. Within the same propagation medium, these wavelets traval at the same speed as their source wavefront. The position of the new wavefront results by superimposing all of these emitted wavelets. Geometrically, the surface that is tangential to the secondary waves can be used in order to determine the future position of the wavfront. Thesrefore, the new wavefront encloses all emitted wavelets. Figure 2.9 visualizes Huygen's Prinicple for a wavfront reflected off from a plane surface.

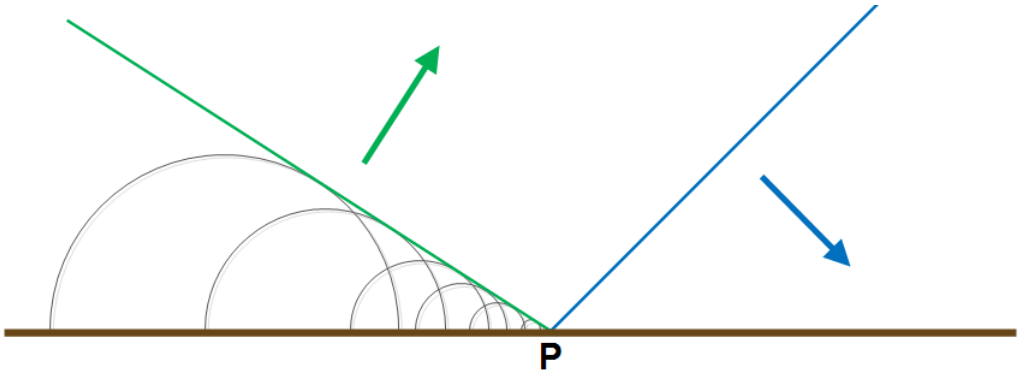


Abbildung 2.9: A moving wavefront (blue) encounters an obstacle (a surface in brown colors) and produces a new wavefront (green) as a result of superposition among all secondary wavelets.

2.2.5 Waves Diffraction

Revisting Hugen's Principle we know that each point on a wavefront can be considered as a source of a spherical wavelet which propagates in every direction. But what exactly happens when a wave's propagation direction is occluded by an object? What will be the outcome when applying Huygen's Principle for that case? An example scenario for this case is shown in figure 2.10.

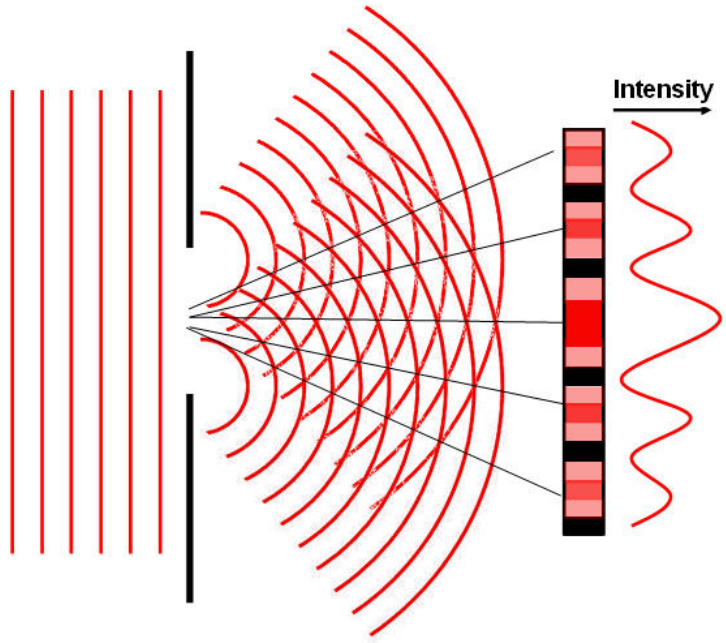
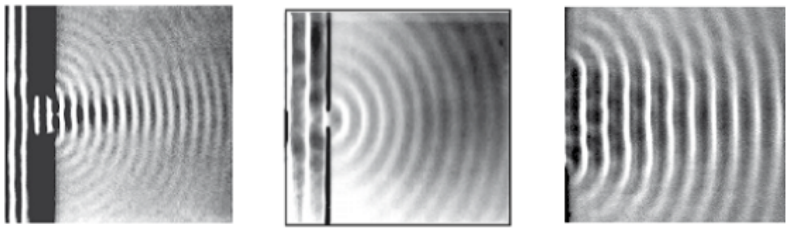


Abbildung 2.10: Illustation[9] of a diffraction scenario in which a plane wavefront passes through a surface with a certain width and how the wave will be bent, also showing the intensity of the resulting wave.

Whenever a propagating wavefront is partially occluded by an obstacle, the wave is not only moving in the direction along its propagation, but is also bent around the edges of the obstacle. In physics, this phenomenon is called diffraction. Waves are diffracted due to interference which occures among all wavelets when applying Huygen's Principle for the case when a wavfront hits an obstacle. Generally, the effect of diffraction is most expressed for waves whose wavelength is roughly similar in size to the dimension of the occluding object. Conversely, if the wavelength is hardly similar in size, then there is almost no wave diffraction perceivable at all. This relationship between the strength of wave diffraction and wavelength-obstacle-dimensions is conceptually illustrated in figure 2.11 when a wave is transmitted through a surface. A reflective example is provided in figure 2.9.

[9]Image source:`http://cronodon.com/images/Single_slit_diffraction_2b.jpg`



(a) W $\ll \lambda$ (b) W $\approx 2\lambda$ (c) W $\approx 6\lambda$

Abbildung 2.11: Illustration[10] of how the effect of diffraction changes when a wave with wavelength λ propagates through a slit of width equal W.

In everday's life, we can see the direct outcome of the effect of wave diffraction in form of structural colors. There are examples from nature such as the irridecent colors on various snake skins as well as artificial examples such as the colorful patterns notable when having a close look at an illuminated compact disc. All in common having having a surface made of very regular nanostructure which is diffracting an incident light. Such a nanostructure which exhibits a certain degree of regularity is also denoted as diffraction grating. More about this in section 5.2.

2.3 Stam's BRDF formulation

In his paper about Diffraction Shader, J. Stam derives a BRDF which is modeling the effect of diffraction for various analytical anisotropic reflexion models relying on the so called scalar wave theory of diffraction for which a wave is assumed to be a complex valued scalar. It's noteworthy, that Stam's BRDF formulation does not take into account the polarization of the light. Fortunately, light sources like sunlight and light bulbs are unpolarized.

A further assumption in Stam's Paper is, the emanated waves from the source are stationary, which implies the wave is a superposition of independent monochromatic waves. This implies that each wave is associated to a definite wavelength lambda. However, sunlight once again fulfills this fact.

In our simulations we will always assume we have given a directional light source, i.e. sunlight. Hence, Stam's model can be used for our derivations.

For his derivations Stam uses the Kirchhoff integral (ADD REF TO WIKI), which is relating the reflected field to the incoming field. This equation is a formalization of Huygen's well-known principle that states that if one knows the wavefront at a given moment, the wave at a later time can be deduced by considering each point on the first wave as the source of a new disturbance. Mathematically speaking, once the field $\psi_1 = e^{ik\mathbf{x}\cdot\mathbf{ss}}$ on the surface is known, the field ψ_2 everywhere else away from the surface can be computed. More precisely, we want to compute the wave ψ_2 equal to the reflection of an incoming planar monochromatic wave $\psi_1 = e^{ik\omega_i * x}$ traveling in the direction ω_i from a surface S to the light source. Formally, this can be written as:

[10] Image taken from:`http://neutrino.ethz.ch/Vorlesung/FS2013/index.php/vorlesungsskript`, chapter 9, figure 9.14

$$\psi_2(\omega_i, \omega_r) = \frac{ike^{iKR}}{4\pi R}(F(-\omega_i - \omega_r) - (-\omega_i + \omega_r)) \cdot I_1(\omega_i, \omega_r) \tag{2.5}$$

with

$$I_1(\omega_i, \omega_r) = \int_S \hat{\mathbf{n}} e^{ik(-\omega_i - \omega_r)\cdot \mathbf{s}} d\mathbf{s} \tag{2.6}$$

In applied optics, when dealing with scattered waves, one does use differential scattering cross-section rather than defining a BRDF which has the following identity:

$$\sigma^0 = 4\pi \lim_{R\to\infty} R^2 \frac{\langle |\psi_2|^2 \rangle}{\langle |\psi_1|^2 \rangle} \tag{2.7}$$

where R is the distance from the center of the patch to the receiving point x_p, $\hat{\mathbf{n}}$ is the normal of the surface at s and the vectors:

The relationship between the BRDF and the scattering cross section can be shown to be equal to

$$BRDF = \frac{1}{4\pi}\frac{1}{A}\frac{\sigma^0}{cos(\theta_i)cos(\theta_r)} \tag{2.8}$$

where θ_i and θ_r are the angles of incident and reflected directions on the surface with the surface normal n. See 2.12.

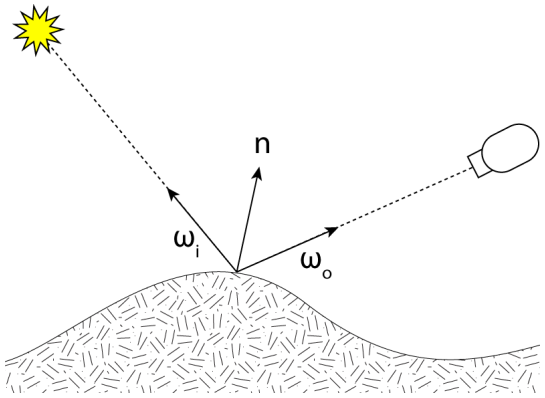


Abbildung 2.12: ω_i points toward the light source, ω_r points toward the camera, n is the surface normal

The components of vector resulting by the difference between these direction vectors: In order to simplify the calculations involved in his vectorized integral equations, Stam considers the components of vector

$$(u, v, w) = -\omega_i - \omega_r \tag{2.9}$$

explicitly and introduces the equation:

$$I(ku, kv) = \int_S \hat{\mathbf{n}} e^{ik(u,v,w)\cdot \mathbf{s}} d\mathbf{s} \tag{2.10}$$

which is a first simplification of 2.6. Note that the scalar w is the third component of 2.9 and can be written as $w = -(cos(\theta_i) + cos(\theta_r))$ using spherical coordinates. The scalar $k = \frac{2\pi}{\lambda}$ represent the wavenumber.

During his derivations, Stam provides a analytical representation for the Kirchhoff integral assuming that each surface point $s(x,y)$ can be parameterized by $(x,y,h(x,y))$ where h is the height at the position (x,y) on the given (x,y) surface plane. Using the tangent plane approximation for the parameterized surface and plugging it into 2.10 he will end up with:

$$\mathbf{I}(ku,kv) = \int\int(-h_x(x,y),-h_y(x,y),1)e^{ikwh(x,y)}e^{ik(ux+vy)}dxdy \tag{2.11}$$

For further simplification Stam formulates auxillary function which depends on the provided height field:

$$p(x,y) = e^{iwkh(x,y)} \tag{2.12}$$

which will allow him to further simplify his equation 2.11 to:

$$\mathbf{I}(ku,kv) = \int\int\frac{1}{ikw}(-p_x,-p_y,ikwp)dxdy \tag{2.13}$$

where he used that $(-h_x(x,y),-h_y(x,y),1)e^{kwh(x,y)}$ is equal to $\frac{(-p_x,-p_y,ikwp)}{ikw}$ using the definition of the partial derivatives applied to the function 2.12.

Let $P(x,y)$ denote the Fourier Transform (FT) of $p(x,y)$. Then, the differentiation with respect to x respectively to y in the Fourier domain is equivalent to a multiplication of the Fourier transform by $-iku$ or $-ikv$ respectively. This leads him to the following simplification for 2.11:

$$\mathbf{I}(ku,kv) = \frac{1}{w}P(ku,kv)\cdot(u,v,w) \tag{2.14}$$

Let us consider the term $g = (F(-\omega_i-\omega_r)-(-\omega_i+\omega_r))$, which is a scalar factor of 2.5. The dot product with g and $(-\omega_i-\omega_r)$ is equal $2F(1+\omega_i\cdot\omega_r)$. Putting this finding and the identity 2.14 into 2.5 he will end up with:

$$\psi_2(\omega_i,\omega_r) = \frac{ike^{iKR}}{4\pi R}\frac{2F(1+\omega_i\cdot\omega_r)}{w}P(ku,kv) \tag{2.15}$$

By using the identity 2.8, this will lead us to his main finding:

$$BRDF_\lambda(\omega_i,\omega_r) = \frac{k^2F^2G}{4\pi^2Aw^2}\langle|P(ku,kv)|^2\rangle \tag{2.16}$$

where G is the so called geometry term which is equal:

$$G = \frac{(1+\omega_i\cdot\omega_r)^2}{cos(\theta_i)cos(\theta_r)} \tag{2.17}$$

Kapitel 3

Derivations

3.1 Adaption of Stam's BRDF

3.1.1 BRDF formulation

Lets assume we have given an incoming light source with solid angle ω_i, θ_i is its angle of incidence, ω_r is the solid angle for the reflected light. Further let λ denote the wavelength and Ω is the hemisphere we of integration for the incoming light. Then, we are able to formulate a BRDF by using its definition 2.3:

$$\begin{aligned} f_r(\omega_i, \omega_r) &= \frac{dL_r(\omega_r)}{L_i(\omega_i)cos(\theta_i)d\omega_i} \\ \Rightarrow f_r(\omega_i, \omega_r)L_i(\omega_i)cos(\theta_i)d\omega_i &= dL_r(\omega_r) \\ \Rightarrow \int_\Omega f_r(\omega_i, \omega_r)L_i(\omega_i)cos(\theta_i)d\omega_i &= \int_\Omega dL_r(\omega_r) \\ \Rightarrow L_r(\omega_r) &= \int_\Omega f_r(\omega_i, \omega_r)L_i(\omega_i)cos(\theta_i)d\omega_i \end{aligned} \tag{3.1}$$

The last equation is the so called rendering equation. We assume that our incident light is a directional, unpolarized light source 4.6 like sunlight and therefore its radiance is given as

$$L_\lambda(\omega) = I(\lambda)\delta(\omega - \omega_i) \tag{3.2}$$

where $I(\lambda)$ is the intensity of the relative spectral power for the wavelength λ. Since all light rays are parallel, whenever we are provided by a directional light source and we can think of radiance as a measure of the light emitted from a particular surface location into a particular direction, above's radiance identity will follow immediately. By plugging the identity 3.2 into our current rendering equation 3.1, we will get:

$$\begin{aligned} L_\lambda(w_r) &= \int_\Omega BRDF_\lambda(\omega_i, \omega_r)L_\lambda(\omega_i)cos(\theta_i)d\omega_i \\ &= BRDF_\lambda(\omega_i, \omega_r)I(\lambda)cos(\theta_i) \end{aligned} \tag{3.3}$$

where $L_\lambda(\omega_i)$ is the incoming radiance and $L_\lambda(\omega_r)$ is the radiance reflected by the given surface. Note that the integral in equation 3.3 vanishes since $\delta(\omega - \omega_i)$ is only equal one if and only if $\omega = \omega_i$.

We are going to use Stam's main derivation (2.16) for the $BRDF(\omega_i, \omega_r)$ in 3.3 by applying the fact that the wavenumber is equal $k = \frac{2\pi}{\lambda}$:

$$\begin{aligned} BRDF(\omega_i, \omega_r) &= \frac{k^2 F^2 G}{4\pi^2 A w^2} \langle |P(ku, kv)|^2 \rangle \\ &= \frac{k^2 F^2 (1 + \omega_i \cdot \omega_r)^2}{cos(\theta_i) cos(\theta_r) 4\pi^2 A w^2} \langle |P(ku, kv)|^2 \rangle \\ &= \frac{4\pi^2 F^2 (1 + \omega_i \cdot \omega_r)^2}{cos(\theta_i) cos(\theta_r) 4\pi^2 A \lambda^2 w^2} \langle |P(ku, kv)|^2 \rangle \\ &= \frac{F(w_i, w_r)^2 (1 + \omega_i \cdot \omega_r)^2}{cos(\theta_i) cos(\theta_r) A \lambda^2 w^2} \langle |P(ku, kv)|^2 \rangle \end{aligned} \tag{3.4}$$

Going back to the definition 2.9 of $(u, v, w) = -\omega_i - \omega_r$ and using spherical coordinates B.2, we get for w the following identity

$$\begin{aligned} w &= -\omega_i - \omega_r \\ &= -(\omega_i + \omega_r) \\ &= -\left(cos(\theta_i) + cos(\theta_r)\right) \end{aligned} \tag{3.5}$$

and therefore w^2 is equal $(cos(\theta_i) + cos(\theta_r))^2$ This new fact will allow us to get even further:

$$\begin{aligned} L_\lambda(\omega_r) &= \frac{F(\omega_i, \omega_r)^2 (1 + \omega_i \cdot \omega_r)^2}{A\lambda^2 cos(\theta_i) cos(\theta_r) (cos(\theta_i) + cos(\theta_r))^2} \left\langle \left| P_{cont}\left(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}\right)\right|^2 \right\rangle cos(\theta_i) I(\lambda) \\ &= I(\lambda) \frac{F(\omega_i, \omega_r)^2 (1 + \omega_i \cdot \omega_r)^2}{\lambda^2 A (cos(\theta_i) + cos(\theta_r))^2 cos(\theta_r)} \left\langle \left| P_{cont}\left(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}\right)\right|^2 \right\rangle \\ &= I(\lambda) \frac{F(\omega_i, \omega_r)^2 (1 + \omega_i \cdot \omega_r)^2}{\lambda^2 A (cos(\theta_i) + cos(\theta_r))^2 cos(\theta_r)} \left\langle \left| T_0^2 P_{dtft}\left(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}\right)\right|^2 \right\rangle \end{aligned} \tag{3.6}$$

Where P_{cont} denotes the continuous inverse Fourier-Transform A.2 for the Taylor-Series A.8 of our height field representing the nano-scaled surface structure, i.e. $P(k,l) = \mathcal{F}^{-1}\{p\}(k,l)$ and P_{dtft} is the inverse Discrete Time Fourier Transform A.3 of $p(x,y) = e^{ikwh(x,y)}$. Furthermore T_0 the sampling distance for the discretization of $p(x,y)$ assuming equal and uniform sampling in both dimensions x and y.

3.1.2 Relative BRDF

In this section we are going to explain how to scale our BRDF formulation such that all of its possible output values are mapped into the range $[0,1]$. Such a

relative BRDF formulation will ease our life for later rendering purposes since usually color values are within the range $[0,1]$, too. Furthermore, this will allow us to properly blend the resulting illumination caused by diffraction with a texture map.

Let us examine what $L_\lambda(\omega_r)$ will be for $\omega_r = \omega_0 := (0,0,*)$ i.e. specular reflection case, denoted as $L_\lambda^{spec}(\omega_0)$. When we know the expression for $L_\lambda^{spec}(\omega_0)$ we would be able to compute the relative reflected radiance for our problem 3.6 by simply taking the fraction between $L_\lambda(\omega_r)$ and $L_\lambda^{spec}(\omega_0)$ which is denoted by:

$$\rho_\lambda(\omega_i,\omega_r) = \frac{L_\lambda(\omega_r)}{L_\lambda^{spec}(\omega_0)} \tag{3.7}$$

But first, let us derive the following expression:

$$\begin{aligned} L_\lambda^{spec}(\omega_0) &= I(\lambda)\frac{F(\omega_0,\omega_0)^2(1+\begin{pmatrix}0\\0\\1\end{pmatrix}\cdot\begin{pmatrix}0\\0\\1\end{pmatrix})^2}{\lambda^2 A(cos(0)+cos(0))^2 cos(0)}\langle\left|T_0^2 P_{dtft}(0,0)\right|^2\rangle \\ &= I(\lambda)\frac{F(\omega_0,\omega_0)^2(1+1)^2}{\lambda^2 A(1+1)^2 1}\left|T_0^2 N_{sample}\right|^2 \\ &= I(\lambda)\frac{F(\omega_0,\omega_0)^2}{\lambda^2 A}\left|T_0^2 N_{sample}\right|^2 \end{aligned} \tag{3.8}$$

Where $N_{samples}$ is the number of samples of the DTFT $A.3$. Thus, we can plug our last derived expression 3.8 into the definition for the relative reflectance radiance 3.7 in the direction w_r and will get:

$$\begin{aligned} \rho_\lambda(\omega_i,\omega_r) &= \frac{L_\lambda(\omega_r)}{L_\lambda^{spec}(\omega_0)} \\ &= \frac{I(\lambda)\frac{F(\omega_i,\omega_r)^2(1+\omega_i\cdot\omega_r)^2}{\lambda^2 A(cos(\theta_i)+cos(\theta_r))^2 cos(\theta_r)}\langle\left|T_0^2 P_{dtft}(\frac{2\pi u}{\lambda},\frac{2\pi v}{\lambda})\right|^2\rangle}{I(\lambda)\frac{F(\omega_0,\omega_0)^2}{\lambda^2 A}\left|T_0^2 N_{sample}\right|^2} \\ &= \frac{F^2(\omega_i,\omega_r)(1+\omega_i\cdot\omega_r)^2}{F^2(\omega_0,\omega_0)(cos(\theta_i)+cos(\theta_r))^2 cos(\theta_r)}\langle\left|\frac{P_{dtft}(\frac{2\pi u}{\lambda},\frac{2\pi v}{\lambda})}{N_{samples}}\right|^2\rangle \end{aligned} \tag{3.9}$$

For simplification and a better overview, let us introduce the following expression, the so called gain-factor:

$$C(\omega_i,\omega_r) = \frac{F^2(\omega_i,\omega_r)(1+\omega_i\cdot\omega_r)^2}{F^2(\omega_0,\omega_0)(cos(\theta_i)+cos(\theta_r))^2 cos(\theta_r) N_{samples}^2} \tag{3.10}$$

Using this substitute, we will end up with the following expression for the relative reflectance radiance from equation 3.9:

$$\rho_\lambda(\omega_i,\omega_r) = C(\omega_i,\omega_r)\langle\left|P_{dtft}(\frac{2\pi u}{\lambda},\frac{2\pi v}{\lambda})\right|^2\rangle \tag{3.11}$$

Using the previous definition for the relative reflectance radiance 3.7:

$$\rho_\lambda(\omega_i, \omega_r) = \frac{L_\lambda(\omega_r)}{L_\lambda^{spec}(\omega_0)} \tag{3.12}$$

Which we can rearrange to the expression:

$$L_\lambda(\omega_r) = \rho_\lambda(\omega_i, \omega_r) L_\lambda^{spec}(\omega_0) \tag{3.13}$$

Let us choose $L_\lambda^{spec}(w_0) = S(\lambda)$ such that is has the same profile as the relative spectral power distribution of CIE Standard Illuminant $D65$ discussed in 4.4.3. Furthermore, when integrating over λ for a specular surface, we should get CIE_{XYZ} values corresponding to the white point for $D65$. The corresponding tristimulus values using CIE colormatching functions 2.4 for the CIE_{XYZ} values look like:

$$\begin{aligned} X &= \int_\lambda L_\lambda(\omega_r)\overline{x}(\lambda)d\lambda \\ Y &= \int_\lambda L_\lambda(\omega_r)\overline{y}(\lambda)d\lambda \\ Z &= \int_\lambda L_\lambda(\omega_r)\overline{z}(\lambda)d\lambda \end{aligned} \tag{3.14}$$

where $\overline{x}$, $\overline{y}$, $\overline{z}$ are the color matching functions. Using our last finding 3.13 for $L_\lambda(\omega_r)$ with the definition for the tristimulus values 3.14, we can actually derive an expression for computing the colors for our initial BRDF formula 3.1. Without any loss of generality it satisfies to derive an explicit expression for just one tristimulus term, for example X. Since The other have a similar formulation, except the we have to replace all X with Y or Z respectively. Therefore, we get:

$$\begin{aligned} X &= \int_\lambda L_\lambda(\omega_r)\overline{x}(\lambda)d\lambda \\ &= \int_\lambda \rho_\lambda(\omega_i, \omega_r) L_\lambda^{spec}(\omega_0)\overline{x}(\lambda)d\lambda \\ &= \int_\lambda \rho_\lambda(\omega_i, \omega_r) S(\lambda)\overline{x}(\lambda)d\lambda \\ &= \int_\lambda C(\omega_i, \omega_r)\langle \left| P_{dtft}(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}) \right|^2 \rangle S(\lambda)\overline{x}(\lambda)d\lambda \\ &= C(\omega_i, \omega_r) \int_\lambda \langle \left| P_{dtft}(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}) \right|^2 \rangle S(\lambda)\overline{x}(\lambda)d\lambda \\ &= C(\omega_i, \omega_r) \int_\lambda \langle \left| P_{dtft}(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}) \right|^2 \rangle S_x(\lambda)d\lambda \end{aligned} \tag{3.15}$$

Where we used the definition $S_x(\lambda)\overline{x}(\lambda)$ in the last step.

3.1.3 Taylor approximation for BRDF

In this section, we will deliver an approximation for the inverse Fourier Transformation of Stam's auxiliary function $p(x, y)$. This derivation will rely on the

definition of Taylor Series expansion $A.8$. Further, we will provide an error bound for our approximation approach for a given number of iterations. Last, we will extend our current BRDF formula by the findings derived within this section.

Given $p(x,y) = e^{ikwh(x,y)}$ form Stam's Paper 2.3 where $h(x,y)$ is a given height field. Let be y real or even complex value, and lets consider the power series for the the exponential function

$$e^t = 1 + t + \frac{t^2}{2!} + \frac{t^3}{3!} + ... = \sum_{n=0}^{\infty} \frac{t^n}{n!} \tag{3.16}$$

Let us define

$$t = t(x,y) = ikwh(x,y) \tag{3.17}$$

where i is the imaginary number. For simplification, let us denote $h(x,y)$ as h. Then it follows by our previous stated identities:

$$\begin{aligned} e^t &= 1 + (ikwh) + \frac{1}{2!}(ikwh)^2 + \frac{1}{3!}(ikwh)^3 + ... \\ &= \sum_{n=0}^{\infty} \frac{(ikwh)^n}{n!}. \end{aligned} \tag{3.18}$$

Hence it holds $p(x,y) = \sum_{n=0}^{\infty} \frac{(ikwh(x,y))^n}{n!}$. Let us now compute the Fourier Transformation of p(x,y) form above:

$$\begin{aligned} \mathcal{F}\{p\}(u,v) &= \mathcal{F}\left\{ \sum_{n=0}^{\infty} \frac{(ikwh)^n}{n!}. \right\}(u,v) \\ &=^{\mathcal{F}\, lin\, Operator} \sum_{n=0}^{\infty} \mathcal{F}\left\{ \frac{(ikwh)^n}{n!} \right\}(u,v) \\ &= \sum_{n=0}^{\infty} \frac{(ikw)^n}{n!} \mathcal{F}\{h^n\}(u,v) \end{aligned} \tag{3.19}$$

Therefore it follows: $P(\alpha,\beta) = \sum_{n=0}^{\infty} \frac{(ikw)^n}{n!} \mathcal{F}\{h^n\}(\alpha,\beta)$ for which $\mathcal{F}_{FT}\{h^n\}(u,v)$. Next we are going to look for an $N \in \mathbb{N}$ such that

$$\sum_{n=0}^{N} \frac{(ikwh)^n}{n!} \mathcal{F}\{h^n\}(\alpha,\beta) \approx P(\alpha,\beta) \tag{3.20}$$

is a good approximation. But first the following two facts have to be proven:

1. Show that there exist such an $N \in \mathbb{N}$s.t the approximation holds true.
2. Find a value for B s.t. this approximation is below a certain error bound, for example machine precision ϵ.

Proof Sketch of 1.

By the **ratio test** (see [**1**]) It is possible to show that the series $\sum_{n=0}^{N} \frac{(ikwh)^n}{n!} \mathcal{F}\{h^n\}(\alpha, \beta)$ converges absolutely:

Proof: Consider $\sum_{k=0}^{\infty} \frac{y^n}{n!}$ where $a_k = \frac{y^k}{k!}$. By applying the definition of the ratio test for this series it follows:

$$\forall y : limsup_{k\to\infty} |\frac{a_{k+1}}{a_k}| = limsup_{k\to\infty} \frac{y}{k+1} = 0 \tag{3.21}$$

Thus this series converges absolutely, no matter what value we will pick for y.

Part 2: Find such an N
Let $f(x) = e^x$. We can formulate its Taylor-Series, stated above. Let $P_n(x)$denote the n-th Taylor polynom,

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k \tag{3.22}$$

where a is our developing point (here a is equal zero).

We can define the error of the n-th Taylor polynom to be $E_n(x) = f(x) - P_n(x)$. the error of the n-th Taylor polynom is difference between the value of the function and the Taylor polynomial This directly implies $|E_n(x)| = |f(x) - P_n(x)|$. By using the Lagrangian Error Bound it follows:

$$|E_n(x)| \leq \frac{M}{(n+1)!}|x-a|^{n+1} \tag{3.23}$$

with $a = 0$, where **M** is some value satisfying $|f^{(n+1)}(x)| \leq M$ on the interval $I = [a, x]$. Since we are interested in an upper bound of the error and since **a** is known, we can reformulate the interval as $I = [0, x_{max}]$, where

$$x_{max} = \|i\| k_{max} w_{max} h_{max} \tag{3.24}$$

We are interested in computing an error bound for $e^{ikwh(x,y)}$. Assuming the following parameters and facts used within Stam's Paper:

- Height of bump: 0.15micro meters
- Width of a bump: 0.5micro meters
- Length of a bump: 1micro meters
- $k = \frac{2\pi}{\lambda}$ is the wavenumber, $\lambda \in [\lambda_{min,}\lambda_{max}]$ and thus $k_{max} = \frac{2\pi}{\lambda_{min}}$. Since $(u, v, w) = -\omega_i - \omega_r$ and both are unit direction vectors, each component can have a value in range [-2, 2].
- for simplification, assume$[\lambda_{min,}\lambda_{max}] = [400nm, 700nm]$.

We get:

$$\begin{aligned} x_{max} &= \|i\| * k_{max} * w_{max} * h_{max} \\ &= k_{max} * w_{max} * h_{max} \\ &= 2 * (\frac{2\pi}{4 * 10^{-7} m}) * 1.5 * 10^{-7} \\ &= 1.5\pi \end{aligned} \tag{3.25}$$

and it follows for our interval $I = [0, 1.5\pi]$.

Next we are going to find the value for M. Since the exponential function is monotonically growing (on the interval I) and the derivative of the **exp** function is the exponential function itself, we can find such an M:

$$\begin{aligned} M &= e^{x_{max}} \\ &= exp(1.5\pi) \end{aligned}$$

and $|f^{(n+1)}(x)| \leq M$ holds. With

$$\begin{aligned} |E_n(x_{max})| &\leq \frac{M}{(n+1)!} |x_{max} - a|^{n+1} \\ &= \frac{exp(1.5\pi) * (1.5\pi)^{n+1}}{(n+1)!} \end{aligned} \tag{3.26}$$

we now can find a value of n for a given bound, i.e. we can find an value of $N \in \mathbb{N}$ s.t. $\frac{exp(1.5\pi)*(1.5\pi)^{N+1}}{(N+1)!} \leq \epsilon$. With Octave/Matlab we can see:

- if N=20 then $\epsilon \approx 2.9950 * 10^{-4}$
- if N=25 then $\epsilon \approx 8.8150 * 10^{-8}$
- if N=30 then $\epsilon \approx 1.0050 * 10^{-11}$

With this approach we have that $\sum_{n=0}^{25} \frac{(ikwh)^n}{n!} \mathcal{F}\{h^n\}(\alpha, \beta)$ is an approximation of $P(u, v)$ with error $\epsilon \approx 8.8150 * 10^{-8}$. This means we can precompute 25 Fourier Transformations in order to approximate P(u,v) having an error $\epsilon \approx 8.8150 * 10^{-8}$.

Using now our approximation for $P_{dtft} = \mathcal{F}^{-1}\{p\}(u, v)$ for the tristimulus value X, we will get:

$$\begin{aligned} X &= C(w_i, w_r) \int_\lambda \langle \left| P_{dtft}(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}) \right|^2 \rangle S_x(\lambda) d\lambda \\ &= C(w_i, w_r) \int_\lambda \left| \sum_{n=0}^{N} \frac{(wk)^n}{n!} \mathcal{F}^{-1}\{i^n h^n\}(\frac{2\pi u}{\lambda}, \frac{2\pi v}{\lambda}) \right|^2 S_x(\lambda) d\lambda \end{aligned} \tag{3.27}$$

3.1.4 Sampling: Gaussian Window

Practically, we cannot compute the DTFT $A.3$ numerically due to finite computer arithmetic, since w is a continuous function for the DTFT. The DFT $A.4$ of a discrete height field patch is equivalent to the DTFT of an infinitely periodic function consisting of replicas of the same discrete patch. By windowing with a window function that is zero outside the central replica, the convolution of either the DFT or the DTFT of height field with the Fourier Transform of the window becomes equivalent.

Let $window_g$ denote the gaussian window with $4\sigma_s\ \mu m$ where $\sigma_f = \frac{1}{2\pi\sigma_s}$ let us further substitute $\mathbf{t}(\mathbf{x},\mathbf{y}) = i^n h(x,y)^n$

$$\mathcal{F}^{-1}_{dtft}\{\mathbf{t}\}(u,v) = \mathcal{F}^{-1}_{fft}\{\mathbf{t}\}(u,v) window_g(\sigma_f) \tag{3.28}$$

Therefore we can deduce the following expression from this:

$$\begin{aligned}
\mathcal{F}^{-1}_{dtft}\{\mathbf{t}\}(u,v) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F^{-1}_{fft}\{\mathbf{t}\}(w_u,w_v)\phi(u-w_u,v-w_v)dw_u dw_v \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \sum_i\sum_j F^{-1}_{fft}\{\mathbf{t}\}(w_u,w_v) \\
&\qquad \delta(w_u-w_i,w_v-w_j)\phi(u-w_u,v-w_v)dw_u dw_v \\
&= \sum_i\sum_j \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F^{-1}_{fft}\{\mathbf{t}\}(w_u,w_v) \\
&\qquad \delta(w_u-w_i,w_v-w_j)\phi(u-w_u,v-w_v)dw_u dw_v \\
&= \sum_i\sum_j F^{-1}_{fft}\{\mathbf{t}\}(w_u,w_v)\phi(u-w_u,v-w_v)
\end{aligned} \tag{3.29}$$

where

$$\phi(x,y) = \pi e^{-\frac{x^2+y^2}{2\sigma_f^2}} \tag{3.30}$$

3.1.5 Final Expression

As the last step of our series of derivations, we plug all our findings together to one big equation in order to compute the color for each pixel on our mesh in the CIE_{XYZ} colorspace. For any given heigh-field $h(x,y)$ representing a small patch of a nano structure of a surface and the direction vectors w_s and w_r from figure 2.12 the resulting color caused by the effect of diffraction can be computed like: Let

$$P_\lambda(u,v) = F^{-1}_{fft}\{i^n h^n\}(\frac{2\pi u}{\lambda},\frac{2\pi v}{\lambda}) \tag{3.31}$$

Then our final expression using our previous derivations will look like:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = C(\omega_i, \omega_r) \int_\lambda \sum_{n=0}^{N} \frac{(wk)^n}{n!} \sum_{(r,s)\in\mathcal{N}_1(u,v)} |P_\lambda(u - w_r, v - w_s)|^2 \\ \phi(u - w_r, v - w_s) \begin{pmatrix} S_x(\lambda) \\ S_y(\lambda) \\ S_z(\lambda) \end{pmatrix} d\lambda \tag{3.32}$$

where $\phi(x,y) = \pi e^{-\frac{x^2+y^2}{2\sigma_f^2}}$ is the Gaussian window 3.1.4.

3.2 Alternative Approach

3.2.1 PQ factors

In this section we are presenting an alternative approach to the previous Gaussian window approach 3.1.4 in order to solve the issue working with $DTFT$ instead the DFT. We assume, that a given surface S is covered by a number of replicas of a provided representative surface patch f. In a simplified, one dimensional scenario, mathematically speaking, f is assumed to be a periodic function, i.e. $\forall x \in \mathbb{R} : f(x) = f(x + nT)$, where T is its period and $n \in \mathbb{N}_0$. Thus, the surfaces can be written formally as:

$$S(x) = \sum_{n=0}^{N} f(x + nT) \tag{3.33}$$

What we are looking for is an identity for the inverse Fourier transform of our surface S, required in order to simplify the (X, Y, Z) colors from 3.27:

$$\begin{aligned} \mathcal{F}^{-1}\{S\}(w) &= \int f(x) e^{iwx} dx \\ &= \int_{-\infty}^{\infty} \sum_{n=0}^{N} f(x + nT) e^{iwx} dx \\ &= \sum_{n=0}^{N} \int_{-\infty}^{\infty} f(x + nT) e^{iwx} dx \end{aligned} \tag{3.34}$$

Next, apply the following substitution $x + nT = y$ which will lead us to:

$$\begin{aligned} x &= y - nT \\ dx &= dy \end{aligned} \tag{3.35}$$

Plugging this substitution back into equation 3.34 we will get:

$$\begin{aligned}
\mathcal{F}^{-1}\{S\}(w) &= \sum_{n=0}^{N} \int_{-\infty}^{\infty} f(x+nT)e^{iwx}dx \\
&= \sum_{n=0}^{N} \int_{-\infty}^{\infty} f(y)e^{iw(y-nT)}dy \\
&= \sum_{n=0}^{N} e^{-iwnT} \int_{-\infty}^{\infty} f(y)e^{iwy}dy \\
&= \sum_{n=0}^{N} e^{-iwnT}\mathcal{F}^{-1}\{f\}(w) \\
&= \mathcal{F}^{-1}\{f\}(w) \sum_{n=0}^{N} e^{-iwnT}
\end{aligned} \tag{3.36}$$

We used the fact that the exponential term e^{-iwnT} is a constant factor when integrating along dy and the identity for the inverse Fourier transform of the function f. Next, let us examine the series $\sum_{n=0}^{N} e^{-iwnT}$ closer:

$$\begin{aligned}
\sum_{n=0}^{N} e^{-uwnT} &= \sum_{n=0}^{N} (e^{-uwT})^n \\
&= \frac{1-e^{iwT(N+1)}}{1-e^{-iwT}}
\end{aligned} \tag{3.37}$$

We recognize the geometric series identity for the left-hand-side of equation 3.37. Since our series is bounded, we can simplify the right-hand-side of equation 3.37.

Note that e^{-ix} is a complex number. Every complex number can be written in its polar form, i.e.

$$e^{-ix} = cos(x) + isin(x) \tag{3.38}$$

Using the following trigonometric identities

$$\begin{aligned}
cos(-x) &= cos(x) \\
sin(-x) &= -sin(x)
\end{aligned} \tag{3.39}$$

combined with 3.38 we can simplify the series 3.37 even further to:

$$\frac{1-e^{iwT(N+1)}}{1-e^{-iwT}} = \frac{1-cos(wT(N+1))+isin(wT(N+1))}{1-cos(wT)+isin(wT)} \tag{3.40}$$

Equation 3.40 is still a complex number, denoted as $(p+iq)$. Generally, every complex number can be written as a fraction of two complex numbers. This implies that the complex number $(p+iq)$ can be written as $(p+iq) = \frac{(a+ib)}{(c+id)}$ for any $(a+ib), (c+id) \neq 0$. Let us use the following substitutions:

$$\begin{aligned} a &:= 1 - cos(wT(N+1)) & b &= sin(wT(N+1)) \\ c &= 1 - cos(wT) & d &= sin(wT) \end{aligned} \tag{3.41}$$

Hence, using 3.41, it follows

$$\frac{1 - e^{iwT(N+1)}}{1 - e^{-iwT}} = \frac{(a + ib)}{(c + id)} \tag{3.42}$$

By rearranging the terms, it follows $(a + ib) = (c + id)(p + iq)$ and by multiplying its right hand-side out we get the following system of equations:

$$\begin{aligned} (cp - dq) &= a \\ (dp + cq) &= b \end{aligned} \tag{3.43}$$

After multiplying the first equation of 3.43 by c and the second by d and then adding them together, we get using the law of distributivity new identities for p and q:

$$\begin{aligned} p &= \frac{(ac + bd)}{c^2 + d^2} \\ q &= \frac{(bc + ad)}{c^2 + d^2} \end{aligned} \tag{3.44}$$

Using some trigonometric identities and putting our substitution from 3.41 for a, b, c, d back into the current representation 3.44 of p and q we will get:

$$\begin{aligned} p &= \frac{1}{2} + \frac{1}{2}\left(\frac{cos(wTN) - cos(wT(N+1))}{1 - cos(wT)}\right) \\ q &= \frac{sin(wT(N+1)) - sin(wTN) - sin(wT)}{2(1 - cos(wT))} \end{aligned} \tag{3.45}$$

Since we have seen, that $\sum_{n=0}^{N} e^{-uwnT}$ is a complex number and can be written as $(p + iq)$, we now know an explicit expression for p and q. Therefore, the one dimensional inverse Fourier transform of S is equal:

$$\begin{aligned} \mathcal{F}^{-1}\{S\}(w) &= \mathcal{F}^{-1}\{f\}(w) \sum_{n=0}^{N} e^{-iwnT} \\ &= (p + iq)\mathcal{F}^{-1}\{f\}(w) \end{aligned} \tag{3.46}$$

Now lets consider our actual problem description. Given a patch of a nano-scaled sureface snake shed represented as a two dimensional heightfield $h(x, y)$. We once again assume that this provided patch is representing the whole surface S of our geometry by some number of replicas of itself. Therefore, $S(x, y) = \sum_{n=0}^{N} h(x + nT_1, y + mT_2)$, assuming the given height field has the dimensions T_1 by T_2. In order to derive an identity for the two dimensional inverse Fourier transformation of S we can similarly proceed like we did to derive equation 3.46.

$$
\begin{aligned}
\mathcal{F}^{-1}\{S\}(w_1,w_2) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\sum_{n_2=0}^{N_1}\sum_{n_2=0}^{N_2} h(x_1+n_1T_1, x_2+n_2T_2)e^{iw(x_1+x_2)}dx_1dx_2 \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\sum_{n_2=0}^{N_1}\sum_{n_2=0}^{N_2} h(y_1,y_2)e^{iw((y_1-n_1T_1)+(y_2+n_2T_2))}dx_1dx_2 \\
&= \sum_{n_2=0}^{N_1}\sum_{n_2=0}^{N_2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(y_1,y_2)e^{iw(y_1+y_2)}e^{-iw(n_1T_1+n_2T_2)}dy_1dy_2 \\
&= \sum_{n_2=0}^{N_1}\sum_{n_2=0}^{N_2} e^{-iw(n_1T_1+n_2T_2)}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Box(y_1,y_2)e^{iw(y_1+y_2)}dy_1dy_2 \\
&= \left(\sum_{n_2=0}^{N_1}\sum_{n_2=0}^{N_2} e^{-iw(n_1T_1+n_2T_2)}\right)\mathcal{F}^{-1}\{h\}(w_1,w_2) \\
&= \left(\sum_{n_2=0}^{N_1} e^{-iwn_1T_1}\right)\left(\sum_{n_2=0}^{N_2} e^{-iwn_2T_2}\right)\mathcal{F}^{-1}\{h\}(w_1,w_2) \\
&= (p_1+iq_1)(p_2+iq_2)\mathcal{F}^{-1}\{h\}(w_1,w_2) \\
&= ((p_1p_2-q_1q_2)+i(p_1p_2+q_1q_2))\mathcal{F}^{-1}\{h\}(w_1,w_2) \\
&= (p+iq)\mathcal{F}^{-1}_{DTFT}\{h\}(w_1,w_2) \qquad (3.47)
\end{aligned}
$$

Where we have defined

$$
\begin{aligned}
p &:= (p_1p_2-q_1q_2) \\
q &:= (p_1p_2+q_1q_2) \qquad (3.48)
\end{aligned}
$$

For the identity of equation 3.47 we made use of Green's integration rule which allowed us to split the double integral to the product of two single integrations. Also, we used the definition of the 2-dimensional inverse Fourier transform of the height field function. We applied a similar substitution like we did in 3.35, but this time twice, once for x_1 and once for x_2 separately. The last step in equation 3.47, substituting with p and q in equation 3.48 will be useful later in the implementation. The insight should be, that the product of two complex numbers is again a complex number. We will have to compute the absolute value of $\mathcal{F}^{-1}\{S\}(w_1,w_2)$ which will then be equal $(p^2+q^2)^{\frac{1}{2}}\left|\mathcal{F}^{-1}\{h\}(w_1,w_2)\right|$

3.2.2 Interpolation

In 3.2.1 we have derived an alternative approach when we are working with a periodic signal instead using the gaussian window approach from sec $sec : gaussianwindow$. Its main finding 3.47 that we can just integrate over one of its period instead iterating over the whole domain. Nevertheless, this main finding is using the inverse DTFT. Since we are using

We are interested in recovering an original analog signal $x(t)$ from its samples $x[t] =$

Therfore, for a given sequence of real numbers $x[n]$, representing a digital signal, its correspond continuous function is:

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] sinc\left(\frac{t - nT}{T}\right) \tag{3.49}$$

which has the Fourier transformation $X(f)$ whose non-zero values are confined to the region $|f| \leq \frac{1}{2T} = B$. When $x[n]$ represents time samples at interval T of a continuous function, then the quantity $f_s = \frac{1}{T}$ is known as its sample rate and $\frac{f_s}{2}$ denotes the Nyquist frequency. The sampling Theorem states that when a function has a Bandlimit B less than the Nyquist frequency, then $x(t)$ is a perfect reconstruction of the original function.

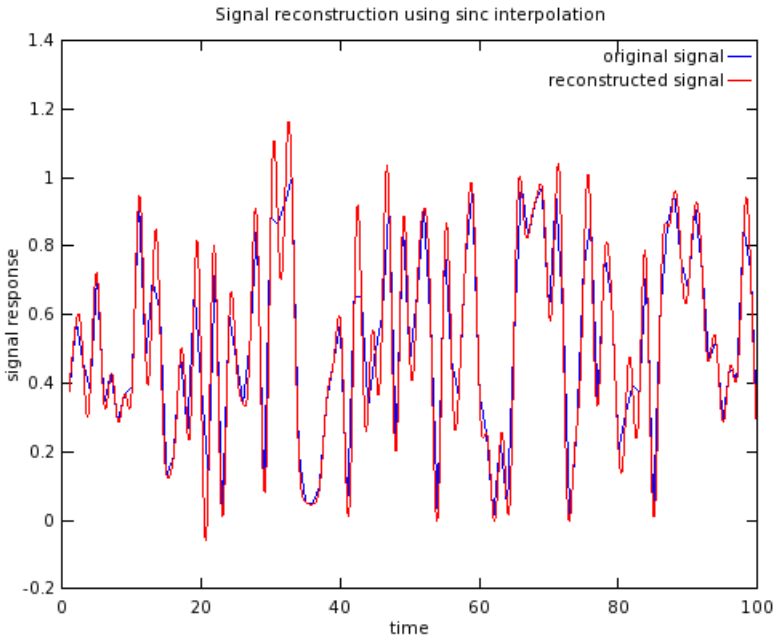


Abbildung 3.1: Comparission between a given random one dimensional input signal $s(t)$ and its sinc interpolation $\hat{s}(t)$. Notice that for the interpolation there were $N = 100$ samples from the original signal provided.

Kapitel 4

Implementation

In computergraphics, we are interested in synthesizing 2d images from a given scene containing our 3d geometries by using so called shader programs. This process is denoted as rendering. The purpose of shader programs, which are executed directly on the GPU hardware device, is to compute the colorization and illumination of the objects living in our scene. All these computations happen in several stages and depend on the provided scene-input parameters like the camera, light sources, objects material constants and the desired rendering effect one is interested in to model. The shader stages are implemented sequentially as small little programs, the so called vertex-, geometry- and fragment-shaders. Those stages are applied within the rendering pipeline sequentially.

Our shaders which we use are written in a high-level language called GLSL, the OpengGL Shading Language. The decision for using OpenGL has been made since my underlying framework, which is responsible for the precomputation of all scene date, is based on another framework, written in Java using JOGL in oder to communicate with the GPU and is also responsible to precompute all the relevant scene data. This framework, the so called jrtr framework, has been developed as an exercise during the class computer graphics held by M. Zwicker which I attended in autumn 2012. The framework itself has been used and extended during this thesis quite a lot. All necessary input data required within our java framework in order to perform the shading is precomputed in Matlab. This is basically addressing all the required precomputations for the provided heigh-fields, referring to computation of the inverse two dimensional Fourier transformations which are further explained within this chapter. The Matlab scripts themself rely on the provided snake nano-scaled sheds images, taken by AFM.

It's noteworthy that all the vertices and their associated data are processed within the vertex-shader, whereas the fragement shader's responsibility is to perform pixelwise rendering, using the input from the vertex shader. Just remember, fragements are determined by a triple of vertices. hence each pixel has assigned a trilinear interpolated value of all input parameters of its spanning vertices. Usually, all necessary transformations are applied vertex-wise, considering the vertex-shader as the precomputation stage for the later rendering within the rendering pipeline, in the fragment-shader. In the geometry shader, new vertices around a considered vertex can be created. this is useful for debugging - displaying normals graphically for example.

In this part of thesis we are going to explain how we render our BRDF formulation derived in the last section in practice. All the necessary computations in order to simulate the effect of diffraction are performed within a fragment shader. This implies that we are modeling pixelwise the effect of diffraction and hence the overall rendering quality and runtime complexity depends on rendering window's resolution.

By the end of this chapter we will have seen how our render works, what we have to precompute and how our shaders work.

4.1 Precomputations in Matlab

Our first task is to precompute the inverse two dimensional discrete Fourier transformations for a given snake shed patch of interest taken by AFM. For that purpose we have written a small Matlab script conceptialized algorithmically in 4.1. Matlab is a interpreted scripting language which offers a huge collection of mathematical and numerically fast and stable algorithms. Our Matlab script reads a given image, which is representing a nano-scaled height field, and computes its inverse two dimensional DFT by using Matlab's internal inverse fast Fourier Transformation function, denoted by $ifft2$. Note that we only require one color channel of the input image since the input image is representing an height field, encoded by just one color. Basically, we are interested in computing the $ifft2$ for different powers of the input image times the imaginary number i since our taylor series approximation 3.1.3 relies on this. Keep in mind that taking the Fourier transformation of an arbitrary function will result in a complex valued output which implies that we will get a complex value for each pixel of our input image. Therefore, for each input image we get as many output images, representing the two dimensional inverse Fourier Transformation, as the minimal amount of taylor terms required for a well-enough approximation. In order to store our output images, we have to use two color channels instead just one like it was for the given input image. Some rendered images are shown in figure 4.1. Instead storing the computed values in plain images, we store our results in binary files. This allows us to have much higher precision for the output values and also it does not waste color channels. In our script every pixel value is normalized by its corresponding Fourier image extrema values such that is it lays in the range $[0, 1]$. Therefore, we have to remember store four scaling factors for each output image as well. Those are the real and imaginary minimum and maximum values. Later, using linear interpolation within the shader, we will get back the rescaled image's original pixel values.

Algorithm 4.1 Precomputation: Fourier images

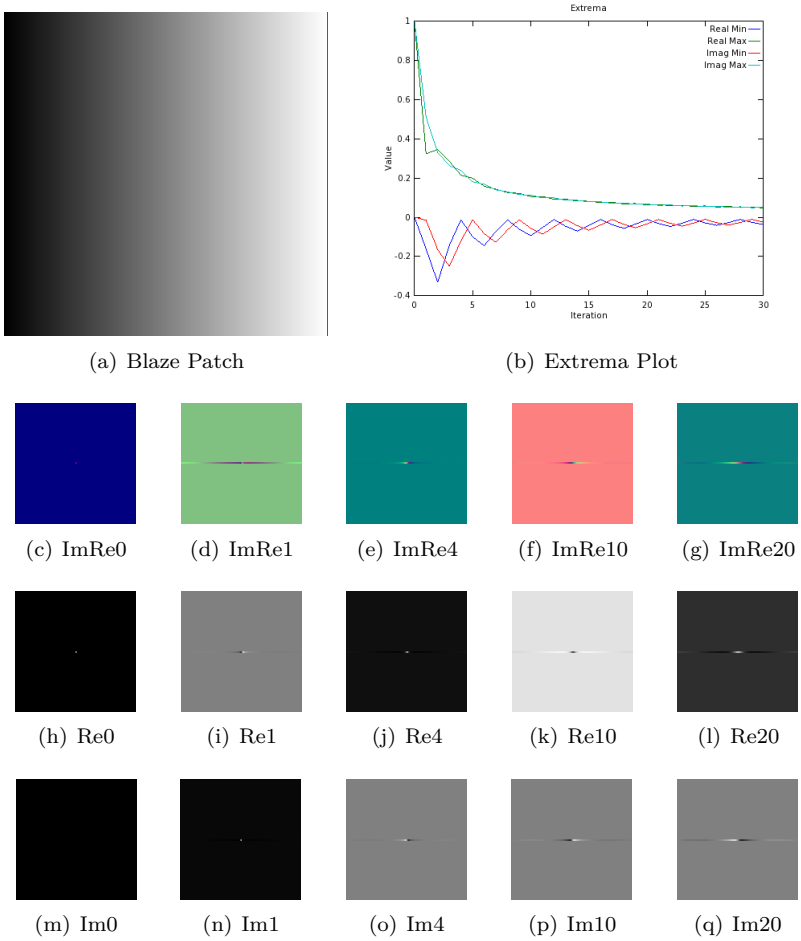
```
% maxH:     A floating-point number specifying
%           the value of maximum height of the
%           height-field in MICRONS, where the
%           minimum-height is zero.
%
% dH:       A floating-point number specifying
%           the resolution (pixel-size) of the
%           'discrete' height-field in MICRONS.
%           It must less than 0.1 MICRONS to
%           ensure proper response for
%           visible-range of light spectrum.
%
% termCnt: An integer specifying the number of
%           Taylor series terms to use.

function []= ComputeFFTImages(maxH, dh, termCnt)
dH = dh*1E-6;
% load patch into patchImg
patchImg = patchImg.*maxH;
% perform imrotate(patchImg, angle)
for t = 0 : termCnt
  patchFFT = power(1j*patchImg, t);
  fftTerm{t+1} = fftshift(ifft2(patchFFT));

  imOut(:,:,1)  = real(fftTerm{t+1});
  imOut(:,:,2)  = imag(fftTerm{t+1});
  imOut(:,:,3)  = 0.5;

  % perform imrotate(imOut, -angle)
  % find real and imaginary extrema of
  % write imOut, extrema, dH, into files.
end
```

The command fftshift rearranges the output of the ifft2 by moving the zero frequency component to the centre of the image. This is useful for visualizing a Fourier Transform with zero frequency components in the middle of the spectrum.



(a) Blaze Patch (b) Extrema Plot

(c) ImRe0 (d) ImRe1 (e) ImRe4 (f) ImRe10 (g) ImRe20

(h) Re0 (i) Re1 (j) Re4 (k) Re10 (l) Re20

(m) Im0 (n) Im1 (o) Im4 (p) Im10 (q) Im20

Abbildung 4.1: Blaze

In figure 4.1 we see some Fourier images rendered by our Matlab script for the blaze grating $4.1(a)$ used as input image. For example figure $4.1(e)$ represents the two dimensional Fourier transform of the input image to the power of four times the imgaginary number i stored in a RGB image. Note that the red color channel $4.1(j)$ contains the real- and the green color channel $4.1(j)$ the imaginary part of the Fourier transform. The plot in figure $4.1(b)$ shows the development of blaze grating's extreme values for different powers.

4.2 Java Renderer

In autumn 2012, during the semester I have attented the class computer graphics held by M. Zwicker where we have developed a real time renderer program written in java. The architecture of the program is divided into two parts: a rendering engine, the so called jrtr (java real time renderer) and an application program. Figure 4.2 outlines the architecture of our renderer.

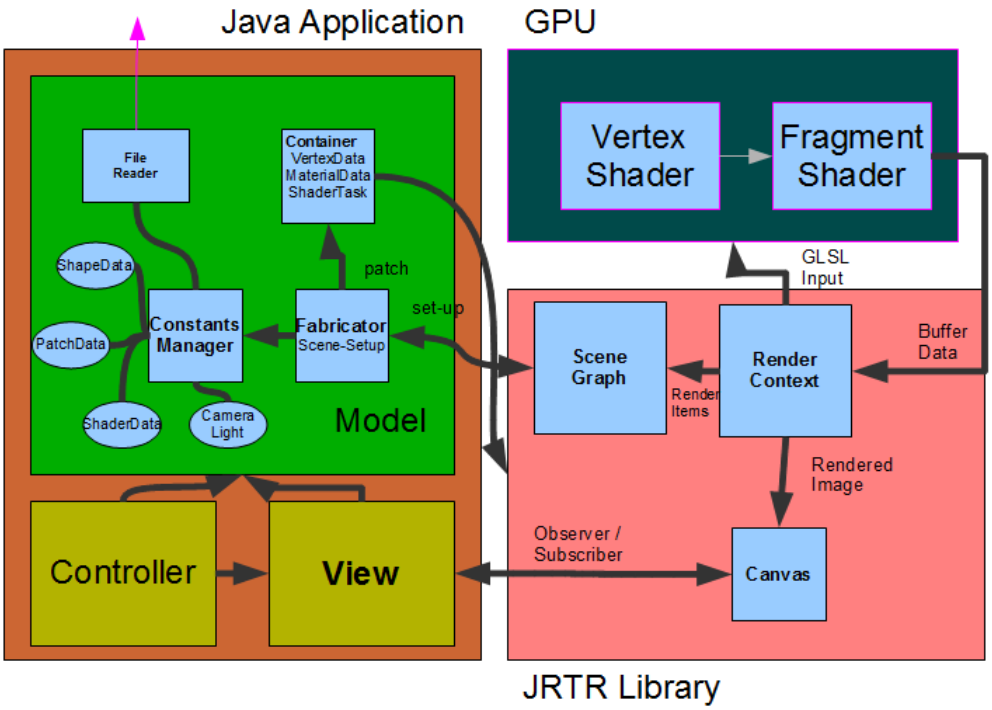


Abbildung 4.2: Renderer Architecture

The application program relies on the MVC (Model-View-Controller) architecture pattern. The View just represents a canvas in which the rendered images are shown. The Controller implements the event listener functionalities in order manipulate the rendered shape within the canvas. The Model of our application program consists of a Fabricator, a file reader and a constants manager. The main purpose of a Fabricator is to set up a rendering scene by accessing a constant manager containing many predefined scene constants. A scene consists of a camera, a light source, a frustum, shapes and their associated material constants. Such materials contain a shape's texture, associated Fourier images $reffig : matlabBlazeFourierImages$ for a given height field and other height field constants such as the maximal height of a bump. A shapes is a geometrical object defined by a wireframe mesh as shown in figure 4.3.

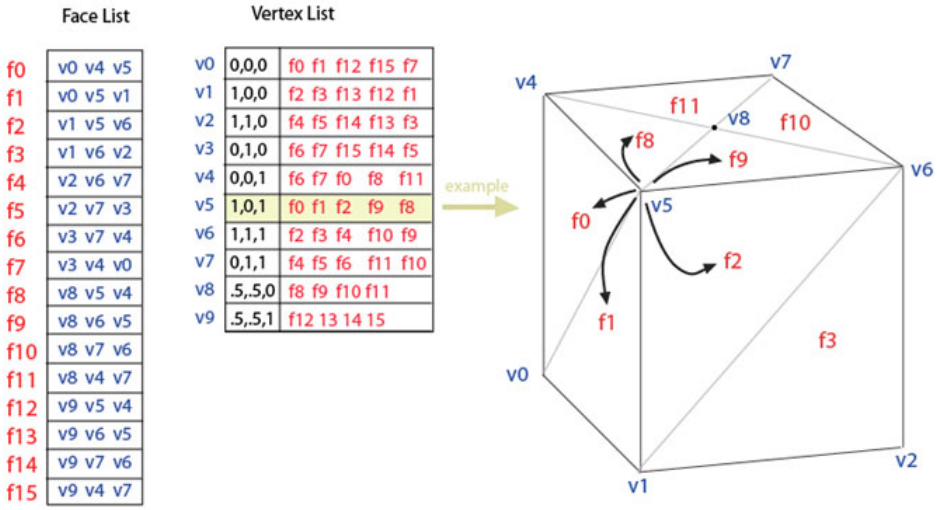


Abbildung 4.3: A wireframe mesh represents an object as a set of faces and a set of vertices.

Such a mesh is a special data structure consisting of vertices, each stored as a triple of xyz positions in an float array and triangles, each defined by a triple of vertex-indices which form a fragment each stored in an integer array. It is also possible to assign additional geometry data like a color, normals and texture coordinates associated to each vertex. The whole scene is stored within container data-structures, defined and managed within jrtr. In our case we rely on a scene graph, which contains all geometries and their transformations in a tree like structured hierarchy. The geometries are stored within an container, including all vertex attributes and the material constants. The jrtr rendering engine uses a low-level API, called OpenGL in order to communicate with the graphics processor unit (GPU) where the actual shading happens. Within jrtr's render context object, the whole resource-management for the rendering pipeline takes place. This means all required low-level buffers are allocated, flushed and assigned by the scene data attributes. The GPU's rendering pipeline will use those buffers for its shading process. Its first stage is the vertex shader 4.3.1 followed by the fragment shader 4.3.2. The jrtr framework also offers the possibility to assign arbitrary shaders.

4.3 GLSL Diffraction Shader

4.3.1 Vertex Shader

The Vertex shader is the first shading stage within our rendering pipeline and responsible for computing all necessary per vertex data. Usually, within a vertex shader each vertex position is transformed into a projective space:

$$p_{projective} = P \cdot C^{-1} \cdot M \cdot p_{obj} \tag{4.1}$$

Where M is a transformation from the local object space to the reference

coordinate system, called world space, C^{-1} camera matrix C and P the projection matrix. The camera matrix defines transformation from camera to world coordinates as shown in figure 4.4.

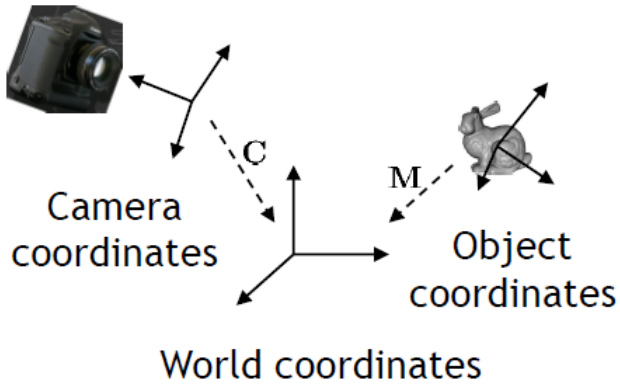


Abbildung 4.4: Camera coordinate system where its origin defines the center of projection of camera

The camera matrix is constructed from its center of projection e, the position the cameras looks at d and up vector denoted by up given in world coordinates like illustrated in figure 4.5

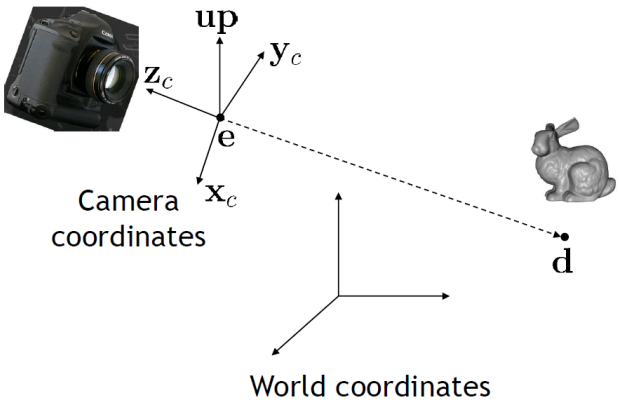


Abbildung 4.5: Illustration of involved components in order to construct the camera matrix. We introduce some helper vectors $z_c = \frac{e-d}{||e-d||}$, $x_c = \frac{up \times z_c}{||up \times z_c||}$ and $z_c \times x_c$ for the actual construction of the camera matrix

The mathematical representation of the camera matrix, using the helper vectors introduced in figure 4.5, looks like:

$$C = \begin{bmatrix} x_c & y_c & z_c & e \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

All vertex shader output will be used within the fragment shader 4.3.2. In our vertex shader we also compute for every vertex in our current geometry the

direction vectors ω_i and ω_r described like in figure 2.12. Those direction vectors are transformed onto the tangent space, a local coordinate system spanned by a vertex's normal, tangent and binormal vector. Have a look at the appendix B.3 for further information and insight about the tangent space. The algorithm 4.2 stated below shows our vertex shader.

Algorithm 4.2 Vertex diffraction shader

Foreach *Vertex* $v \in$ *Shape* **do**
$vec3N = normalize(modelM * vec4(normal, 0.0)).xyz$
$vec3T = normalize(modelM * vec4(tangent, 0.0)).xyz$
$vec3B = normalize(cross(N, T))$
$vec3Pos = ((cop_w - position)).xyz$
$vec4lightDir = (directionArray[0])$
$lightDir = normalize(lightDir)$
$l = projectVectorOnTo(lightDir, TangentSpace)$
$p = projectVectorOnTo(Pos, TangentSpace)$
$normalize(l); normalize(p)$
$p_{per} = P \cdot C^{-1} \cdot M \cdot p_{obj}$
end for

As already mentioned within our derivations 3.1.1, our light source is a directional light source 4.6.

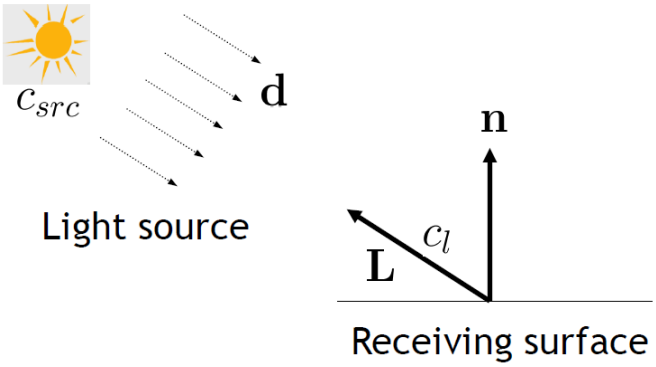


Abbildung 4.6: For a directional light source all light rays are in parallel.

4.3.2 Fragment Shader

The purpose of a fragment shader is to render per fragment. A fragment is spanned by three vertices of a given mesh. For each pixel within a fragment in the fragment shader, the output of from its spanning vertices computed in the vertex shaders 4.2 is trilinearly interpolated depending on the pixel's position within the fragment. Furthermore, there can be additional input be assigned which is not directly interpolated from the output of vertex shader programs. In our fragment shader 4.3 this will be: all the references to the image buffers, containing the Fourier images computed in Matlab 4.1, the number steps for the taylor approximation (in our shader 30), the minimal and maximal wavelength, scaling factors, a reference to a lookup table containing the CIE_{XYZ} color

weights. Basically the whole computation within our fragment shader relies on the the direction of light and the viewing direction. Our shader performs a numerical integration for our final derived expression 3.32 using the trapezoidal-rule with uniform discretization of the wavelength spectrum at $5nm$ step sizes. This implies we are compressing sampled frequencies to the region near to the origin of their frequency domain due to the fact we are dividing the (u, v) by the wavelength. The Gaussian window approach derived in 3.1.4 is performed for each discrete λ value using a window large enough to span $4\sigma_f$ in both dimensions. For computing DFT tables we generally use nanostructure height fields that span at least $65\mu m^2$ and are sampled with resolution of at least 100nm. This ensures that the spectral response encompasses al the wavelengths in the visible spectrum, i.e. from 380nm to 780nm. For natural structures in nano-scale, most of their spectral energy lies at lower spatial frequencies which maps closer to region $(u, v) = (0, 0)$ than higher frequencies. This is why we have chosen to sample (u, v) space non-linearly.

Algorithm 4.3 Fragment diffraction shader

Foreach *Pixel* $p \in$ *Fragment* **do**
  INIT $BRDF_{XYZ}, BRDF_{RGB}$ **TO** $vec4(0.0)$
  $(u, v, w) = -\omega_i - \omega_r$
  for $(\lambda = \lambda_{min};\ \lambda \leq \lambda_{max};\ \lambda = \lambda + \lambda_{step})$ **do**
    $xyzWeights = ColorWeights(\lambda)$
    $lookupCoord = lookupCoord(u, v, \lambda)$
    INIT P **TO** $vec2(0.0)$
    $k = \frac{2\pi}{\lambda}$
    for $(n = 0$ **TO** $T)$ **do**
      $taylorScaleF = \frac{(kw)^n}{n!}$
      INIT F_{fft} **TO** $vec2(0.0)$
      $anchorX = int(floor(center.x + lookupCoord.x * fftImWidth)$
      $anchorY = int(floor(center.y + lookupCoord.y * fftImHeight)$
      for $(i = (anchorX - winW)$ **TO** $(anchorX + winW + 1))$ **do**
        for $(j = (anchorY - winW)$ **TO** $(anchorY + winW + 1))$ **do**
          $dist = distVecFromOriginTo(i, j)$
          $pos = localLookUp(i, j, n)$
          $fftVal = rescaledFourierValueAt(pos)$
          $fftVal *= gaussWeightOf(dist)$
          $F_{fft} += fftVal$
        end for
      end for
      $P += taylorScaleF * F_{fft}$
    end for
    $xyzPixelColor += dot(vec3(|P|^2), xyzWeights)$
  end for
  $BRDF_{XYZ} = xyzPixelColor * C(\omega_i, \omega_r) * shadowF$
  $BRDF_{RGB}.xyz = D_{65} * M_{XYZ-RGB} * BRDF_{XYZ}.xyz$
  $BRDF_{RGB} = gammaCorrect(BRDF_{RGB})$
end for

From line 4 to 26:
Within this loop happens the uniform sampling along wavelength-space. $ColorWeights(\lambda)$ computes the color weight for the current wavelength λ by bilinear interpolation between the color weight for $\lceil\lambda\rceil$ and $\lfloor\lambda\rfloor$ which are stored in a external weights-table (assuming this table contains wavelengths in 1nm steps). At line 6: $lookupCoord(u, v, \lambda)$ the coordinates for the texture lookup are computed - See 4.5. Line 25 sums up the diffraction color contribution for the current wavelength in iteration λ.

From line 9 to 24:
Within this loop happens the Taylor series approximation until a predefined upper bound, denoted by T, has been reached. Basically, the spectral response is approximated for our current (u, v, λ). Furthermore, neighborhood boundaries for the gaussian-window sampling are computed, denoted as anchorX and anchorY.

From line 14 to 22:
In this most inner loop, the convolution of the gaussian window with the inverse FFT of the patch is pixel-wise performed. $gaussWeightOf(dist)$ computes the weights (3.30) from the distance between the current pixel's coordinates and the current neighbor's position in texture space. Local lookup coordinates for the current fourier coefficient $fftVal$ value are computed at line 17 and computed like described in 4.7. The actual texture lookup is performed at line 18 using those local coordinates. Inside $rescaledFourierValueAt$ the values $fftVal$ is rescaled by its extrema, i,e. $(fftVal * Max + Min)$ is computed, since $fftVal$ is normalized 4.1. The current $fftVal$ values in iteration is scaled by the current gaussian weight and then summed to the final neighborhood FFT contribution at line 20.

After line 26:
At line 27 the gain factor $C(\omega_i, \omega_r)$ 3.10 is multiplied by the current computed pixel color like formulated in 3.11. The gain factor contains the geometric term $refeq : geometricterm$ and the Fresnel term F. W approximate F by the Schlick approximation $B.1$, using an reactive index at 1.5 since this is close to the measured value from snake sheds. Our BRDF values are scaled by s shadowing function as described in (SEE REFERENCES - PAPER), since most of the grooves in the snake skin nano-structures would from a V-cavity along the plane for a wave front with their top-edges at almost the same height.

Last, we transform our colors from the CIE_XYZ colorspace to the CIE_RGB space using the CIE Standard Illuminant D65, followed by a gamma correction. See 4.4.3 for further insight.

4.4 Technical details

4.4.1 Texture lookup

In a GLSL shader the texture coordinates are normalized which means that the size of the texture maps to the coordinates on the range $[0, 1]$ in each dimension.

By convention the the bottom left corner of an image has the coordinates $(0,0)$, whereas the top right corner has the value $(1,1)$ assigned.

Given a nano-scaled surface patch P with a resolution A by A microns stored as an N by N pixel image I. Then one pixel in any direction corresponds to $dH = \frac{A}{N}\mu m$. In Matlab we compute a series of n output images $\{I_{out_1}, ..., I_{out_n}\}$ from I, which we will use for the lookup in our shader - See figure 4.7. For the lookup we use scaled and shifted (u,v) coordinates from 2.9.

Since the zero frequency component of output images was shifted towards the centre of each image, we have to shift u, v to the center of the current N by N pixel image by a bias b. Mathematically, the bias is a constant value is computed the following:

$$b = (N\%2 == 0) \quad ? \quad \frac{N}{2} : \frac{N-1}{2} \tag{4.3}$$

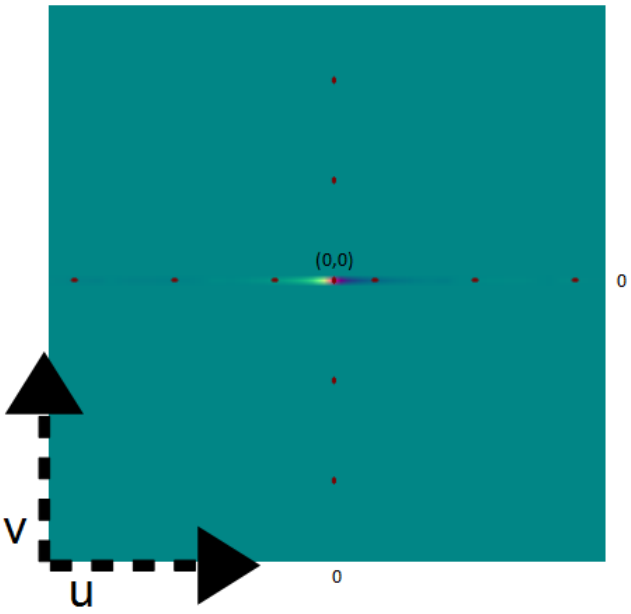


Abbildung 4.7: (u,v) lookup image

For the scaling we have to think a little further: lets consider a T periodic signal in time, i.e. $x(t) = x(t+nT)$ for any integer n. After applying the DFT, we have its discrete spectrum $X[n]$ with frequency interval $w0 = 2pi/T$ and time interval $t0$. Let $k = \frac{2\pi}{\lambda}$ denote the wavenumber for the current wavelength λ. Then the signal is both periodic with time period T and discrete with time interval t_0 then its spectrum should be both discrete with frequency interval w_0 and periodic with frequency period $\Omega = \frac{2\pi}{t_0}$. This gives us the idea how to discretize the spectrum: Let us consider our Patch P assuming it is distributed as a periodic function on our surface. Then, its frequency interval along the x direction is $w_0 = \frac{2\pi}{T} = \frac{2\pi}{N*dH}$. Thus only wave numbers that are integer multiples of w_0 after a multiplication with u must be considered, i.e. ku is integer multiple of w_0. Hence the lookup for the u-direction will look like:

$$\frac{ku}{w_0} = \frac{kuNdH}{2\pi} \tag{4.4}$$

$$= \frac{uNdH}{\lambda} \tag{4.5}$$

Using those findings 4.3, 4.5, the final (u, v) texture lookup-coorinates for the current wavelength λ in iteration, will then look like:

$$(u_{lookup}, v_{lookup}) = \left(\frac{uNdH}{\lambda} + b, \frac{vNdH}{\lambda} + b \right) \tag{4.6}$$

Note for the windowing approach we are visiting a one pixel neighborhood for each pixel p. This is like a base change with (u_{lookup}, v_{lookup}) as new coordinate system origin. The lookup coordinates for the neighbor-pixel (i, j) are:

$$(u_{lookup}, v_{lookup}) = (i, j) - (u_{lookup}, v_{lookup}) \tag{4.7}$$

4.4.2 Texture Blending

The final rendered color for each pixel is a weighted average of different color components, such as the diffraction color, the texture color and the diffuse color. In our shader the diffraction color is weighted by a constant $w_{diffuse}$. the texture color is once scales by a binary weight determined by the absolute value of the Fresnel Term F and once by $1 - w_{diffuse}$.

Algorithm 4.4 Texture Blending

$\alpha = (abs(F) > 1)?1 : 0$
$c_{out} = (1 - w_{diffuse}) * c_{diffraction} + (1 - \alpha) * c_{texture} + w_{diffuse} * c_{texture})$

4.4.3 Color Transformation

In our shader we access a table which contains precomputed CIE's color matching functions values from $\lambda_{min} = 380nm$ to $\lambda_{max} = 780nm$ in $5nm$ steps. Such a function value table can be found at downloaded at cvrl.ioo.ucl.ac.uk for example. We compute the (X, Y, Z) CIE_{XYZ} color values as described in **??**.

We can transform the color values into CIE_{RGB} by performing the following linear transformation:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = M \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.8}$$

where one possible transformation is:

$$M = \begin{bmatrix} 0.41847 & -0.15866 & -0.082835 \\ -0.091169 & 0.25243 & 0.015708 \\ 0.00092090 & -0.0025498 & 0.17860 \end{bmatrix} \tag{4.9}$$

There are some other color space transformation. The shader uses the CIE Standard Illuminant D65 which is intended to represent average daylight. Using D65 the whole colorspace transformation will look like:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = M \cdot \begin{bmatrix} X \cdot D65.x \\ Y \cdot D65.y \\ Z \cdot D65.z \end{bmatrix} \tag{4.10}$$

Last we perfrom gamma correction on each pixel's (R, G, B) value. Gamma correction is a non linear transformation which controls the overall brightness of an image.

4.5 Discussion

The fragment shader algorithm described in 4.3 performs the gaussian window approach by sampling over the whole wavelength spectrum in uniform step sizes. This algorithm is valid but also slow since we iterate for each pixel over the whole lambda spectrum. Furthermore, for any pixel, we iterate over its 1 neighborhood. Considering the loop for the taylor approximation as well, we will have a run-time complexity of $O(\#spectrtumIter \cdot \#taylorIter \cdot neighborhoodRadius^2)$. Hence, Instead sampling over the whole wavelength spectrum, we could instead integrate over just a few required lambdas which are elicited like the following: Lets consider (u, v, w) defined as 2.9. Let d be the spacing between two slits of a grating. For any $L(\lambda) \neq 0$ it follows $\lambda_n^u = \frac{du}{n}$ and $\lambda_n^v = \frac{du}{n}$. For $n = 0$ there it follows $(u, v) = (0, 0)$. If $u, v > 0$

$$N_{min}^u = \frac{du}{\lambda_{max}} \leq n_u \leq \frac{du}{\lambda_{min}} = N_{min}^u$$
$$N_{min}^v = \frac{dv}{\lambda_{max}} \leq n_v \leq \frac{dv}{\lambda_{min}} = N_{min}^v$$

If $u, v < 0$

$$N_{min}^u = \frac{du}{\lambda_{min}} \leq n_u \leq \frac{du}{\lambda_{min}} = N_{max}^u$$
$$N_{min}^v = \frac{dv}{\lambda_{min}} \leq n_v \leq \frac{dv}{\lambda_{min}} = N_{max}^v$$

By transforming those equation to $(\lambda_{min}^u, \lambda_{min}^u)$, $(\lambda_{min}^v, \lambda_{min}^v)$ respectively for any (u, v, w) for each pixel we can reduce the total number of required iterations in our shader.

Another variant is the PQ approach described in chapter 2 3.2.1. Depending on the interpolation method, there are two possible variants we can think of as described in 3.2.2. Either we try to interpolate linearly or use sinc interpolation. The first variant does not require to iterate over a pixel's neighborhood, it is also faster than the gaussian window approach. One could think of a combination of those tho optimization approaches. Keep in mind, both of these approaches are further approximation. The quality of the rendered images will suffer using those two approaches. The second variant, using the sinc function interpolation is well understood in the field of signal processing and will give us reliable

results. The drawback of this approach is that we again have to iterate over a neighborhood within the fragment shader which will slow down the whole shading. The following algorithm describes the modification of the fragment shader 4.3 in oder to use sinc interpolation for the pq approach 3.2.1.

Algorithm 4.5 Sinc interpolation for pq approach

Foreach *Pixel* $p \in$ *Image* I **do**
$w_p = \sum_{(i,j)\in\mathcal{N}_1(p)} sinc(\Delta_{p,(i,j)} \cdot \pi + \epsilon) \cdot I(i,j)$
$c_p = w_p \cdot (p^2 + q^2)^{\frac{1}{2}}$
$render(c_p)$
end for

In a fragment shader we compute for each pixel p in the current fragment its reconstructed function value $f(p)$ stores in w_p. w_p is the reconstructed signal value at $f(p)$ by the sinc function as described in 3.2.2. We calculate the distance $\Delta_{p,(i,j)}$ between the current pixel p and each of its neighbor pixels $(i,j) \in \mathcal{N}_1(p)$ in its one-neighborhood. Multiplying this distance by π gives us the an angle used for the sinc function interpilation. We add a small integer ϵ in order to avoid division by zeros side-effects.

Kapitel 5

Evaluation and data acquisition

5.1 Data Acquisition

For measurement on the true surface topography of snake sheds, samples are stuck on glass plates using double face tape, the animal was pushed up below a hollowed plate letting the skin emerging from the top of the plate using an atomic force microscope (AFM). An AFM is a microscope that uses a tiny probe mounted on a cantilever to scan the surface of an object. The probe is extremely close tobut does not touch the surface. As the probe traverses the surface, attractive and repulsive forces arising between it and the atoms on the surface induce forces on the probe that bend the cantilever. The amount of bending is measured and recorded, providing a map of the atoms on the surface. Atomic force microscopes is a very high-resolution type of scanning probe microscopy, with demonstrated resolution on the order of fractions of a nanometer, more than 1000 times better than the optical diffraction limit.

5.2 Diffraction Gratings

An idealised grating like in figure 5.2 is made of a very large number of parallel, evenly spaced slits in an opaque sheet. Typically, it would have about 10,000 slits. In order to cause diffraction, the spacing between slits must be wider than the wavelength of the incoming light beam. Each slit in the grating acts as a quasi point light source from which light propagates in all directions. Figure 5.1 illustrates this behaviour for a monochromatic light source passing through a grating and shows that the the outgoing angle will be different from the incident angle. Hence, the diffracted light **??** is composed of the sum of interfering wave components emanating from each slit in the grating.

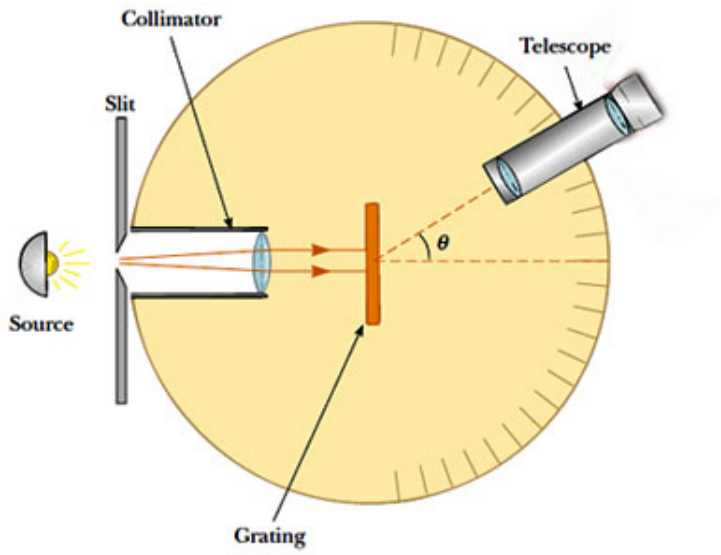


Abbildung 5.1: Spectometer: When a beam of monochromatic light passes through a grating placed in a spectrometer, images of the sources can be seen through the telescope at different angles.

Suppose monochromatic light is directed at the grating parallel to its axis as shown in figure 5.1. Let the distance between successive slits be equal d.

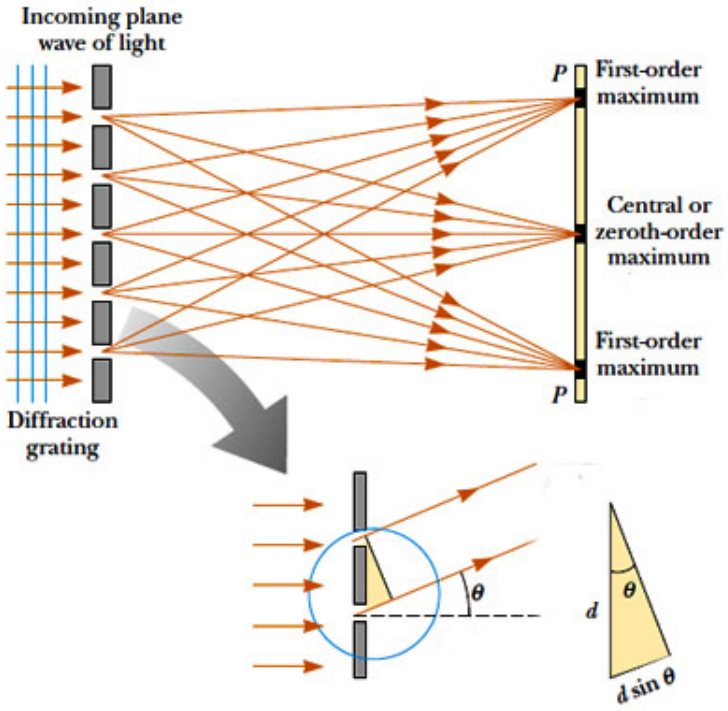


Abbildung 5.2: Light directed to parallel to grating:

The diffraction pattern on the screen is the result of the combined effects of diffraction and interference. Each slit causes diffraction, and the diffracted beams in turn interfere with one another to produce the pattern. The path difference between waves from any two adjacent slits can be found by dropping a perpendicular line between the parallel waves. By geometry, this path difference is $dsin(\theta)$. If the path difference equals one wavelength or some integral multiple of a wavelength, waves from all slits will be in phase and a bright line will be observed at that point. Therefore, the condition for maxima in the interference pattern at the angle θ is:

$$dsin(\theta) = m\lambda \tag{5.1}$$

where $m \in \mathbb{N}_0$ is the order of diffraction.

Because d is very small for a diffraction grating, a beam of monochromatic light passing through a diffraction grating is splitted into very narrow bright fringes at large angles θ.

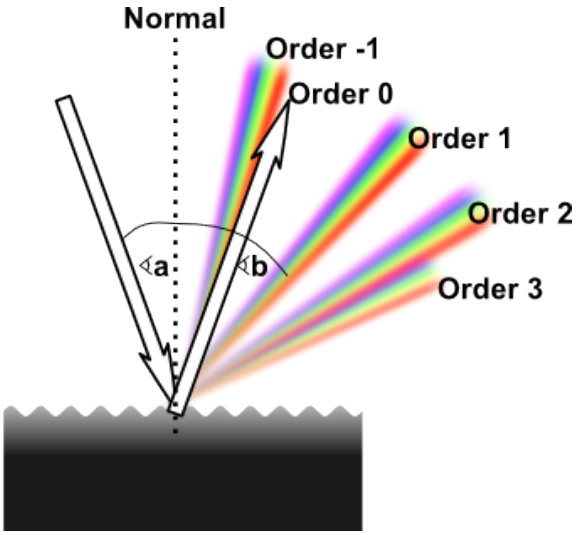


Abbildung 5.3: Different Orders of diffraction

When a narrow beam of white light is directed at a diffraction grating along its axis, instead of a monochromatic bright fringe, a set of colored spectra are observed on both sides of the central white band as shown in figure 5.3.

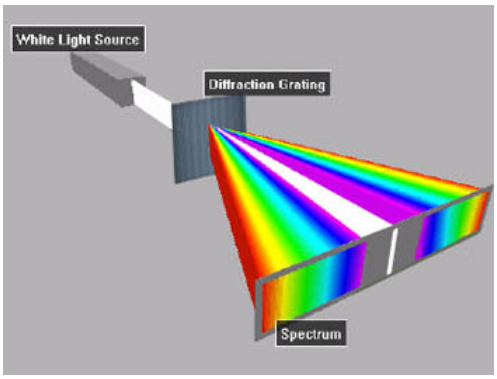


Abbildung 5.4: White Light beam causes coloured diffraction spectra

Since the angle θ increases with wavelength λ, red light, which has the longest wavelength, is diffracted through the largest angle. Similarly violet light has the shortest wavelength and is therefore diffracted the least. This relationship between angle and wavelength is illustrated in figure 5.4. Thus, white light is split into its component colors from violet to red light. The spectrum is repeated in the different orders of diffraction, emphasizing certain colors differently, depending on their order of diffraction like shown in figure 5.3. Note that only the zero order spectrum is pure white. Figure 5.5 shows the relative intensity resulting when a beam of light hits a diffraction grating for different number of periods. From the graph we recognise that the more slits a grating has, the sharper more slopes the function of intensity gets. This is similar like saying that, the more periods a grating has, the sharper the diffracted color spectrum

gets like shown in figure 5.6.

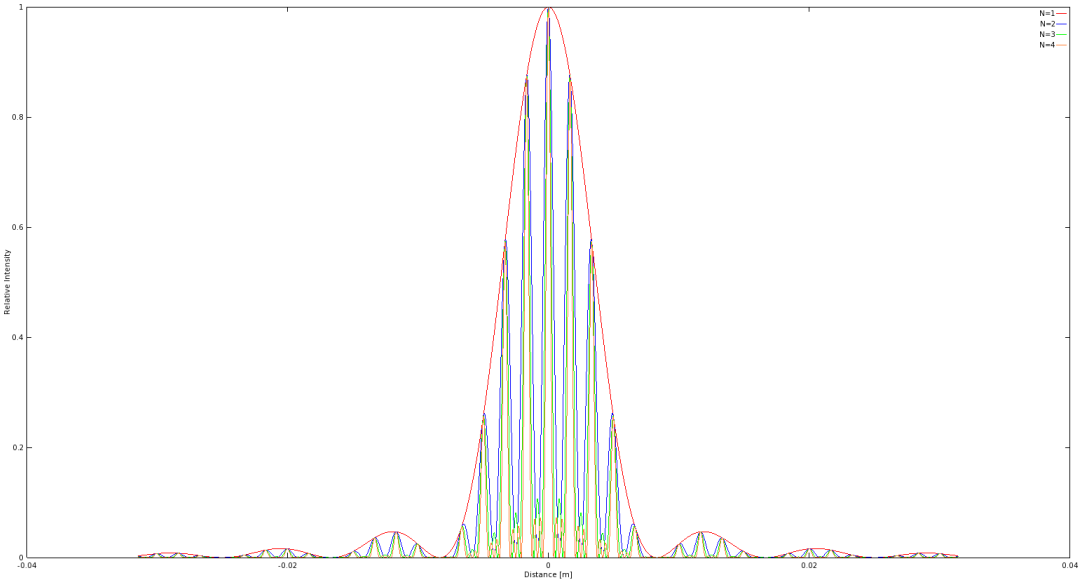
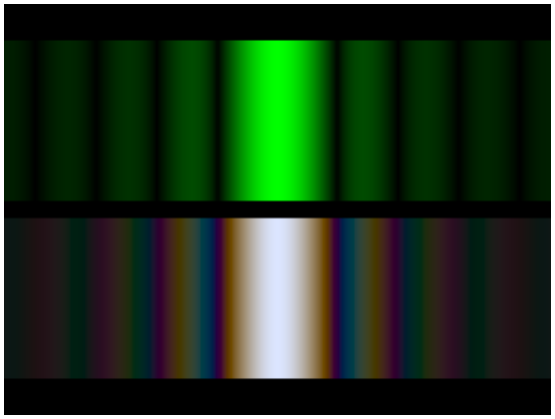
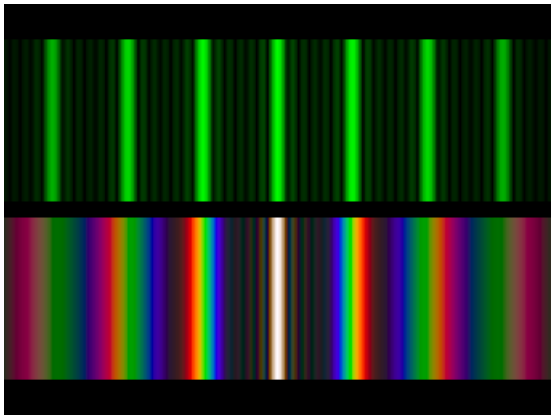


Abbildung 5.5: Relative intensitiies of a diffracted beam of light at wavelength $\lambda = 500nm$ on a grating for different number of periods N width slit width of 30 microns and slit seperation of 0.15 mm each. The viewer is 0.5m apart from the grating.



(a) one slit



(b) seven slits

Abbildung 5.6: Difference of diffraction pattern between a monochromatic (top) and a white (bottom) light spectra for different number of slits.

5.3 Evaluation

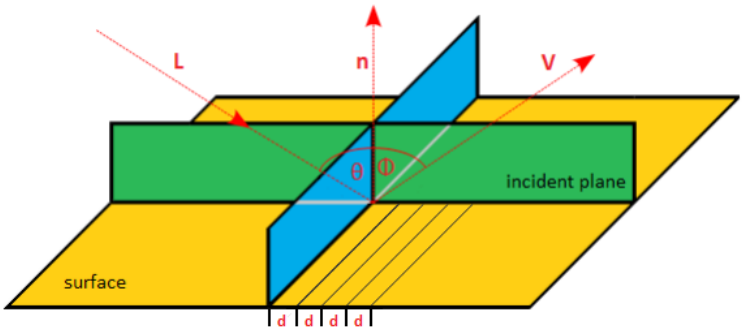


Abbildung 5.7: Experimental setup for evaluation: A light beam with direction L hits the surface, representing a grating pattern with periodicity d, at the incident plane relative to the surface normal n at angle θ and emerges at angle ϕ with direction V.

The physical reliability of our BRDF models has been verified by applying those on various patches which are a synthetic blazed grating, an Elaphe and a Xenopeltis snake shed sample patch. We compared the resulting response against the response resulting by the grating equation, which models the relationship between the grating spacing and the angles of the incident and diffracted beams of light. Figure 5.7 illustrates the geometrical setup for our evaluation approach: A monochromatic beam of light with wavelength λ hits a surface with periodicity d at an angle θ relative to the normal n along its incident plane. The beam emerges from the surface at the angle ϕ.

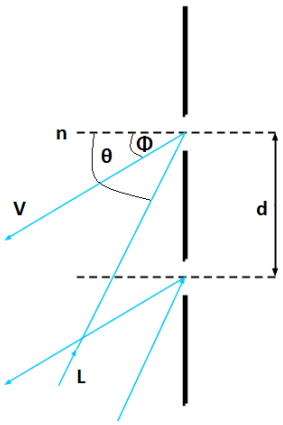


Abbildung 5.8: Reflecting grating: When the incident light direction is not parallel to its axis at the grating, there is another $sin(\phi)$ involved. See also the grating equation 5.2.

The maximum in intensity is given by the grating equation derived from the

equation 5.1 following figure 5.8:

$$sin(\theta) = sin(\phi) + \frac{m\lambda}{d} \tag{5.2}$$

In our evaluation we are interested in the first order diffraction, i.e. m equals one which. We further assume that the incident light direction ω_i is given. In contrast the direction of the reflected wave ω_r is not given. In Mathematics, a three dimensional direction vector is fully defined by two two angles, i.e. it can be represented by spherical coordinates with radius $r = 1$. By convention, we denote those two vectors by θ and ϕ like in figure 5.7. Hence, θ_i, ϕ_i and ϕ_r are given constants whereas θ_i is a free parameter for our evaluation simulation. Therefore, we are going to compare the maxima for peak viewing angles corresponding to each wavelength using data produced by our method against the maxima resulting by the grating equation 5.2.

5.3.1 Precomputation

For evaluation purposes we have implemented our brdf models in java. We once again use our geometrical setup as illustrated in figure 2.12 where θ_i, ϕ_i and ϕ_r are provided as input values and θ_i is a free parameter. Within our evaluation we have set them to $\theta_i = 75$ $\phi_i = 0$ $\phi_r = 180$ degrees. The wavelength space Λ and the range Θ of our free parameter θ_i are discretized in equidistant steps whereas their step sizes are given as input arguments for our Java program:

$$\Lambda = \{\lambda | \lambda = \lambda_{min} + k \cdot \lambda_{step}, \quad k \in \{0, .., C-1\}\} \tag{5.3}$$

where $\lambda_{step} = \frac{\lambda_{max} - \lambda_{min}}{C-1}$ and C is the discretisation level of the lambda space. We similarly discretise the angle space by predefining an minimal and maximal angle boundary and $ceil(angMax - angMin)/angInc$ is the number of angles. Our Java BRDF model implementations are applied on the grid $[\Lambda, \Theta]$ and will store their spectral response in a matrix

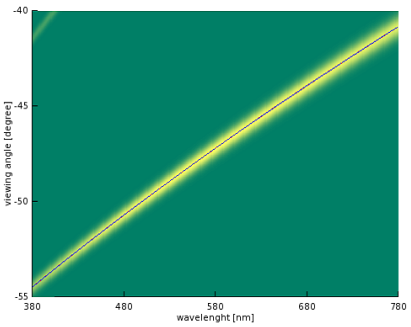
$$R = \{response(\lambda_i, \theta_r^j) | i \in Index(\Lambda), \quad j \in Index(\Theta)\} \tag{5.4}$$

We will plot this matrix and compare its graph against the grating equation for similar condition like in stated in algorithm 5.1.

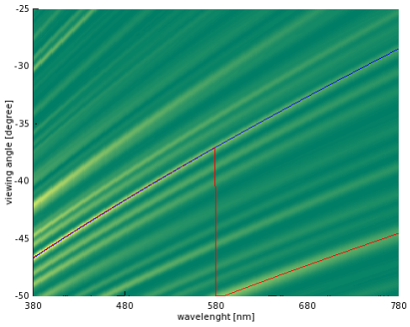
Algorithm 5.1 Vertex diffraction shader

load matrix R 5.4
$\lambda_{count} = |\Lambda|$
$\lambda_{inc} = \frac{\lambda_{max} - \lambda_{min}}{\lambda_{count}}$
$\lambda = \lambda_{min} + \lambda_{inc} \cdot (-1 + [1 : \lambda_{count}])$
$[maxCmaxI] = max(R)$
$viewAngForMax = angMin + angInc \cdot (maxI - 1)$
$thetaV = asin\left(\frac{\lambda}{d} - \sin\left(\frac{\theta_I \pi}{180}\right)\right) \cdot \frac{180}{\pi}$
$plot(\lambda, viewAngForMax)$ ▷ graph resulting by our brdf model
$plot(lambda, thetaV)$ ▷ graph resulting by grating equation

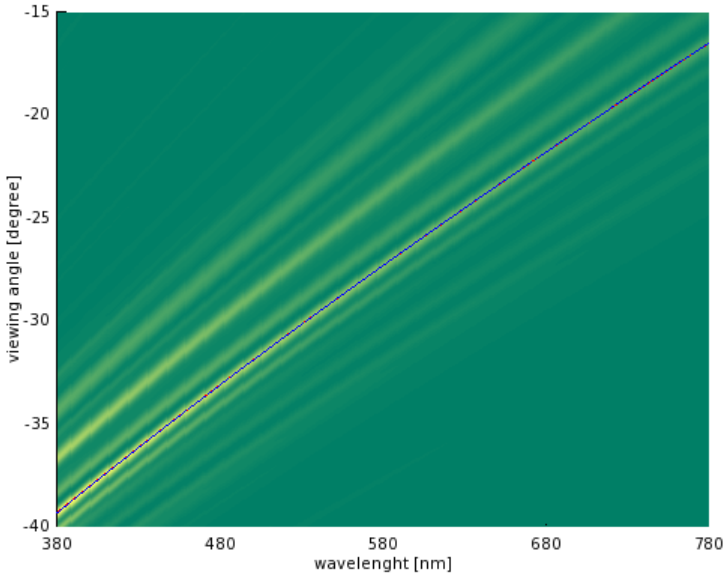
5.3.2 Evaluation graphs



(a) Blaze grating



(b) Xenopeltis grating



(c) Elaphe grating

Abbildung 5.9: Reflectance obtained by using the shading approach described in algorithm 4.3 simulating a BRDF which models the effect of diffraction at different viewing angles over the spectrum of visible light.

In this section we discuss the quality of our BRDF models applied to different surface structures. For that purpose we compare the resulting relative reflectance computed as described in section 5.3.1 for each of our BRDF models to the idealized grating equation 5.2.

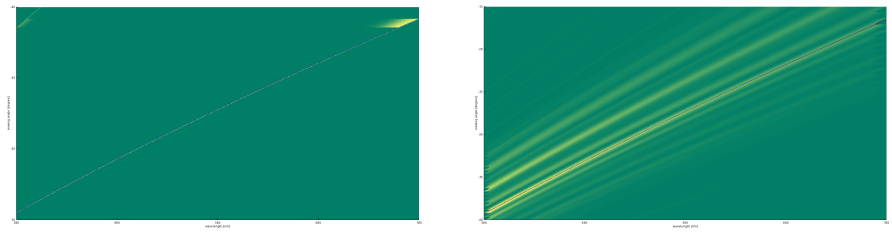
Patch	Mean[mm]	Variance[mm]
Blazed grating **??**	2500.34	0.16
Elaphe grating **??**	1144.28	0.15
Xenopeltis grating **??**	1552.27	0.45

Tabelle 5.1: Statistics of periodicity d of our used gratings **??** estimated by using the grating equation 5.2. This table was provided by Mr. D.Singh.

Figure 5.9 shows the reflectance graphs resulting by the shading approach of sampling the whole lambda space descriped in algorithm 4.3. This evaluation has been applied to different idealized periodic structures, namely to the Blaze- 5.9(a), Elaphe- 5.9(c) and Xenopeltis-grating 5.9(b), using an illumination angle $\theta_i = 75$ degrees. Note that higher response values are plotted in yellow and lower values in green. For each of the graphs we determine the viewing angles with peak reflectance for various wavelengths and then plot this peak viewing angles against their wavelength as solid red curves. The blue curve represents diffraction angles for an idealized periodic structure with a certain periodicity d according to the grating equation 5.2. The corresponding periodicity for every grating structure is estimated using the precomputed response data using again the grating equation and are tabulated in table 5.1.

The red and blue curve are closely overlapping in our figures 5.9(a) and 5.9(c). For Blaze and Elaphe there is only diffraction along only along one direction perceivable. Since the Blazed grating is synthetic we use its exact periodicity to plot the blue curve instead of estimating it. The Xenopeltis grating is evaluated just along the direction for the finger like structures. For Xenopeltis it is interesting to see that the red curve for the peak viewing angle toggles between two ridges corresponding to two different periodicities. this happens because there are multiple sub regions of the nanostructure with slightly different orientations and periodicity. Each sub region carves out a different yellowish ridge. depending on the viewing angle, reflectance due to one such subregion can be higher than from the others.

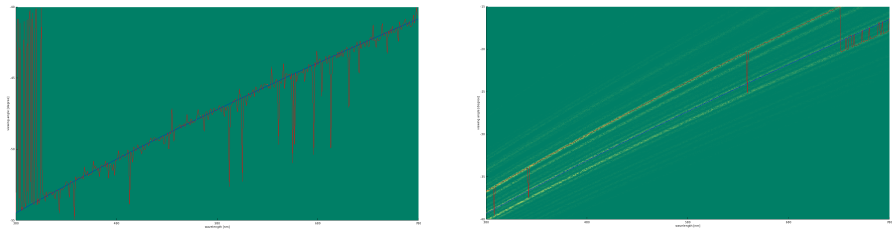
Figure 5.10 shows the evaluation plots for the (N_{min}, N_{max}) shading approach which integrates over a reduced wavelength spectrum applied to the Blaze- 5.10(b) and the Elaphe-grating 5.10(b). This optimization apporach is mentioned within the discussion section of the implementation chapter 4.5 as a run-time complexity enhancement of the whole lambda space sampling approach 5.9. The response curve again closely matches the correspinding grating equation curve for both evaluation graphs and also look similar to the corresponding evaluation plots when integrating over the whole lambda space shown previously in figure . Therefore we may assume this optimization to be valid.



(a) Blaze grating (b) Elaphe grating

Abbildung 5.10: Reflectance obtained using $N_{min}N_{max}$ optimization apporach

Last let us consider the evaluation graphs of the PQ approach 4.5 in figure 5.11. The PQ approach assumes the given grating being periodically distributed on a shape's surface. For this approach we have plotted evaluation graphs of the Blaze- 5.11(a) and Elaphe grating 5.11(b). For both graphs their response curves have some similarities but also some differences compared to their corresponding grating equation curve. We could say that the response curve of the blaze grating is weakly oscillating around the grating equation curve (blue) but basically following it even there are some outliers. The response curve of the Elpahe grating is not following its corresponding first order grating equation curve rather another response curve for the pq approach. This could be due to the assumption of the PQ approach that a given patch must be periodically distributed along the surface which is actually not that case. Nevertheless, the red curve fits one of the response curves.



(a) Blaze grating (b) Elaphe grating

Abbildung 5.11: Reflectance obtained using PQ optimization apporach

Anhang A

Appendix

A.1 Signal Processing Basics

A signal is a function that conveys information about the behavior or attributes of some phenomenon. In the physical world, any quantity exhibiting variation in time or variation in space (such as an image) is potentially a signal that might provide information on the status of a physical system, or convey a message between observers.

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image.

A.1.1 Fourier Transformation

The Fourier-Transform is a mathematical tool which allows to transform a given function or rather a given signal from defined over a time- (or spatial-) domain into its corresponding frequency-domain.

Let f an measurable function over $\mathbb{R}^n$. Then, the continuous Fourier Transformation(**FT**), denoted as $\mathcal{F}\{f\}$ of f, ignoring all constant factors in the formula, is defined as:

$$\mathcal{F}_{FT}\{f\}(w) = \int_{\mathbb{R}^n} f(x)e^{-iwt}dt \tag{A.1}$$

whereas its inverse transform is defined like the following which allows us to obtain back the original signal:

$$\mathcal{F}_{FT}^{-1}\{f\}(w) = \int_{\mathbb{R}} \mathcal{F}\{w\}e^{iwt}dt \tag{A.2}$$

Usual w is identified by the angular frequency which is equal $w = \frac{2\pi}{T} = 2\pi v_f$. In this connection, T is the period of the resulting spectrum and v_f is its corresponding frequency.

By using Fourier Analysis, which is the approach to approximate any function by sums of simpler trigonometric functions, we gain the so called Discrete

Time Fourier Transform (in short **DTFT**). The DTFT operates on a discrete function. Usually, such an input function is often created by digitally sampling a continuous function. The DTFT itself is operation on a discretized signal on a continuous, periodic frequency domain and looks like the following:

$$\mathcal{F}_{DTFT}\{f\}(w) = \sum_{-\infty}^{\infty} f(x)e^{-iwk} \tag{A.3}$$

Note that the DTFT is not practically suitable for digital signal processing since there a signal can be measured only in a finite number of points. Thus, we can further discretize the frequency domain and will get then the Discrete Fourier Transformation (in short **DFT**) of the input signal:

$$\mathcal{F}_{DFT}\{f\}(w) = \sum_{n=0}^{N-1} f(x)e^{-iw_n k} \tag{A.4}$$

Where the angular frequency w_n is defined like the following $w_n = \frac{2\pi n}{N}$ and N is the number of samples within an equidistant period sampling.

Any continuous function $f(t)$ can be expressed as a series of sines and cosines. This representation is called the Fourier Series (denoted by FS) of $f(t)$.

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n cos(nt) + \sum_{n=1}^{\infty} b_n cos(nt) \tag{A.5}$$

where

$$\begin{aligned} a_0 &= \int_{-\pi}^{\pi} f(t)dt \\ a_n &= \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)cos(nt)dt \\ b_n &= \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)sin(nt)dt \end{aligned} \tag{A.6}$$

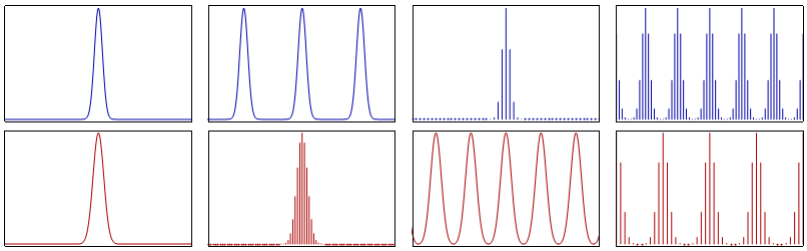


Abbildung A.1: Relationship[1] between the continuous Fourier transform and the discrete Fourier transform: Left column: A continuous function (top) and its Fourier transform *A*.1 (bottom). Center-left column: Periodic summation of the original function (top). Fourier transform (bottom) is zero except at discrete points. The inverse transform is a sum of sinusoids called Fourier series *A*.5. Center-right column: Original function is discretized (multiplied by a Dirac comb) (top). Its Fourier transform (bottom) is a periodic summation (DTFT) of the original transform. Right column: The DFT *A*.4 (bottom) computes discrete samples of the continuous DTFT *A*.3. The inverse DFT (top) is a periodic summation of the original samples.

Spetail signal $f(t)$ is	Operator	Transformed frequency signal $\hat{f}(\omega)$ is
continuous and periodic in t	FS *A*.5	only discrete in ω
only continuous in t	FT *A*.1	only continuous in ω
only discrete in t	DTFT *A*.3	continuous and periodic in ω
discrete and periodic in t	DFT *A*.4	discrete and periodic in ω

Tabelle A.1: Fourier operator to apply for a given spatial input signal and the properties of its resulting output signal in frequency space

A.1.2 Convolution

The convolution $f * g$ of two functions f, g: $\mathbb{R}^n \to \mathbb{C}$ is defined as:

$$(f * g)(t) = \int_{\mathbb{R}^n} f(t)g(t-x)dx \tag{A.7}$$

Note that the Fourier transform of the convolution of two functions is the product of their Fourier transforms. This is equivalent to the fact that Convolution in spatial domain is equivalent to multiplication in frequency domain. Therefore, the inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms. Last an illustration of the relationships between the previous presented Fourier transformations and different given input signals. First an concrete example shown in Figure *A*.1. Table *A*.1 tells what Fourier transformation operator has to be applied to which kind of input signal and what properties its resulting Fourier transform will have.

A.1.3 Taylor Series

Taylor series is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point.

[1] image of illustration has been taken from wikipedia

The Taylor series $\mathcal{T}$ of a real or complex-valued function $f(x)$ that is infinitely differentiable at a real or complex number a is the power series:

$$\mathcal{T}(f;a)(x) = \sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n \tag{A.8}$$

Anhang B

Appendix

B.1 Schlick's approximation

The Fresnel's equations describe the reflection and transmission of electromagnetic waves at an interface. That is, they give the reflection and transmission coefficients for waves parallel and perpendicular to the plane of incidence. Schlick's approximation is a formula for approximating the contribution of the Fresnel term where the specular reflection coefficient R can be approximated by:

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos\theta)^5 \tag{B.1}$$

and

$$R_0 = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2$$

where θ is the angle between the viewing direction and the half-angle direction, which is halfway between the incident light direction and the viewing direction, hence $\cos\theta = (H \cdot V)$. And n_1, n_2 are the indices of refraction of the two medias at the interface and R_0 is the reflection coefficient for light incoming parallel to the normal (i.e., the value of the Fresnel term when $\theta = 0$ or minimal reflection). In computer graphics, one of the interfaces is usually air, meaning that n_1 very well can be approximated as 1.

B.2 Spherical Coordinates

$$\forall \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3 : \exists r \in [0, \infty) \exists \phi \in [0, 2\pi] \exists \theta \in [0, \pi] \text{ s.t.}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} rsin(\theta)cos(\phi) \\ rsin(\theta)sin(\phi) \\ rcos(\theta) \end{pmatrix}$$

B.3 Tangent Space

The concept of tangentspace-transformation of tangent space is used in order to convert a point between world and tangent space. GLSL fragment shaders require normals and other vertex primitives declared at each pixel point, which mean that we have one normal vector at each texel and the normal vector axis will vary for every texel.

Think of it as a bumpy suface defined on a flat plane. If those normals were declared in the world space coordinate system, we would have to rotate these normals every time the model is rotated, even when just for a small amount. Since the lights, cameras and other objects are usually defined in world space coordinate system, and therefore, when they are involved in an calculation within the fragment shader, we would to have to rotate them as well for every pixel. This would involve almost countless many object to world matrix transformations meed to take place at the pixel level. Therefore, instead doing so, we transform all vertex primitives into tangent space within the vertex shader.

To make this point clear an example: Even we would rotate the cube in figure *B*.1, the tangent space axis will remain aligned with respect to the face. Which practically speaking, will save us from performing many space transformations applied pixel-wise within the fragment shader and instead allows us to perform us the tangenspace transformation of every involved vertex primitive in the vertex-shader.

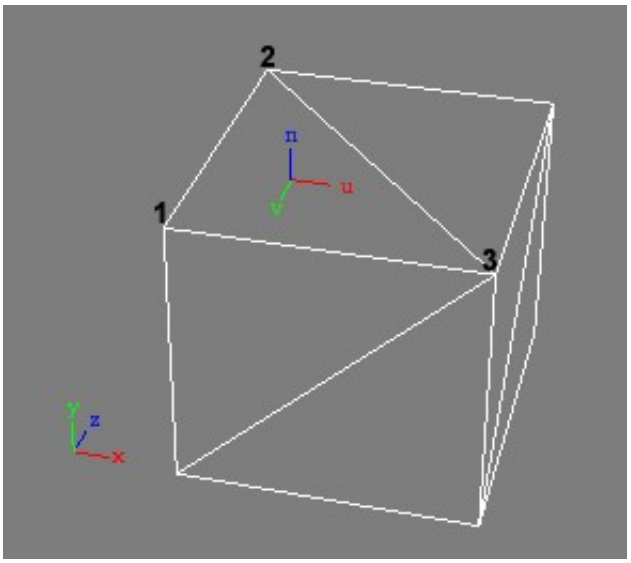


Abbildung B.1: Cube in world space (x, y, z) showing the tangen space (u, v, n) of its face $(2, 1, 3)$

Tabellenverzeichnis

Abbildungsverzeichnis

List of Algorithms

Literaturverzeichnis

[al.10] AL., Matin T.: Correlating Nanostructures with Function: Structurral Colors on the Wings of a Malaysian Bee. (2010), August

[Bar07] BARTSCH, Hans-Jochen: *Taschenbuch Mathematischer Formeln*. 21th edition. HASNER, 2007. – ISBN 978–3–8348–1232–2

[DSD14] D. S. DHILLON, et a.: Interactive Diffraction from Biological Nanostructures. In: *XYZ* (2014), January

[For11] FORSTER, Otto: *Analysis 3*. 6th edition. VIEWEG+TEUBNER, 2011. – ISBN 978–3–8348–1232–2

[JG04] JUAN GUARDADO, NVIDIA: Simulating Diffraction. In: *GPU Gems* (2004). `https://developer.nvidia.com/content/gpu-gems-chapter-8-simulating-diffraction`

[PAT09] PAUL A. TIPLER, Gene M.: *Physik für Wissenschaftler und Ingenieure*. 6th edition. Spektrum Verlag, 2009. – ISBN 978–3–8274–1945–3

[PS09] P. SHIRLEY, S. M.: *Fundamentals of Computer Graphics*. 3rd edition. A K Peters, Ltd, 2009. – ISBN 978–1–56881–469–8

[RW11] R. WRIGHT, et a.: *OpenGL SuperBible*. 5th edition. Addison-Wesley, 2011. – ISBN 978–0–32–171261–5

[Sta99] STAM, Jos: Diffraction Shaders. In: *SIGGRPAH 99 Conference Proceedings* (1999), August

Erklärung

gemäss Art. 28 Abs. 2 RSL 05

Name/Vorname: .....

Matrikelnummer: .....

Studiengang: ……

Bachelor ☐ Master ☐ Dissertation ☐

Titel der Arbeit: .....

.....

.....

LeiterIn der Arbeit: .....

.....

Ich erkläre hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe o des Gesetztes vom 5. September 1996 über die Universität zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist.

.....
Ort/Datum

.....
Unterschrift