Gaze Directed Hybrid Rendering using Photon Mapping

Group 1023, 2008-2009
Jeppe Jensen and Simon J.K. Pedersen

M.Sc. Thesis
Vision, Graphics, and Interactive Systems

AALBORG UNIVERSITY

Aalborg University
Faculties of Engineering, Science, and Medicine
Niels Jernes Vej 10
DK-9220 Aalborg Øst
Phone +45 99 40 99 40
http://www.ins.aau.dk/

Title
Gaze Directed Hybrid Rendering using Photon Mapping

Topic
M.Sc. Thesis

Project Period
September 2nd 2008 to June 3rd 2009

Semester
3rd and 4th of master program

Group Number
1023

Group Members
Jeppe Jensen
Simon J.K. Pedersen

Supervisor
Claus B. Madsen

Circulation: 4
Report: 152 pages
Appendix: 37 pages
Total: 189 pages
Finalized: June 3rd 2009

Abstract

The initial hypothesis of the project was: *The human visual system is not capable of perceiving everything that happens on a screen. Thus the quality of a graphics rendering can be improved by using an eye tracker to track the gaze point, and concentrate the use of computational resources to render the area around the gaze point and use less resources in the peripheral of the vision, without it being perceptible to the eye.*

Preliminary tests proved that users were unable to perceive the lower quality rendering when a gaze area, of 20° visual angle, was rendered in high quality at the gaze point tracked by an eye tracker. The gaze area was rendered with global illumination using photon mapping while the outer area was rendered with local illumination using OpenGL. To fully benefit from gaze directed hybrid rendering, a gaze directed photon mapping algorithm was developed. The algorithm guides photons toward the gaze region to achieve a higher photon density in this region.

A test of the real-time visual quality revealed that 77 % of the test subjects preferred gaze directed photon mapping instead of traditional photon mapping, and a quantitative analysis of the number of photons in the gaze region showed an increase of up to 85 %. The frame rate of hybrid rendering with gaze directed photon mapping was up to five times faster than rendering the entire screen with global illumination using photon mapping.

The conclusion is that gaze directed hybrid rendering with gaze directed photon mapping can be used to achieve computer graphics of better quality at lower computation costs, without loss of visual quality.

PREFACE

This project entitled *Gaze Directed Hybrid Rendering using Photon Mapping* was completed during the 3rd and 4th semester of the master's program on the Vision, Graphics, and Interactive Systems specialization. The Vision, Graphics, and Interactive Systems specialization is a join effort between the Department of Electronic Systems and the Department of Media Technology, under the Faculties of Engineering, Science, and Medicine at Aalborg University.

Typographies

The following typography is used when a referring to a piece of code which is not syntactic complete, or a class name, `glRender = new glRender()`. While `GazePhoton* parent = null;` is used when referring to a piece of code which is syntactic complete.

Citations

Citations in the report use the following system. A citation like [Jensen, 2001] is placed at the end of a paragraph if the paragraph is based on information from the source. If background information for a section is obtained from multiple sources they are listed in the beginning of the section as e.g. [Funkhouser and Sequin, 1993, Murphy and Duchowski, 2007]. A single statement can also be supported by a citation, in that case the citation is placed at the end of the sentence.

Glossary

A short glossary of some of the new terms used in the report can be found on a foldout page, on page 183, in the back of the report.

Acknowledgment

We would like to thank Henrik Wann Jensen for the help of answering questions about photon mapping.

Attached DVD

A DVD is attached in the back of the report with the following content:

- The report in PDF format (`GDHRUPM_09gr1023.pdf`)
- A poster of the project, presented at Graphical Visionday 2009, DTU (`poster/`)
- The source code and scene files + applied libraries (`source_code/`)
- The tex-files and images of the report (`report_source/`)
- The test images/videos and test files (`test_files/`)

Authors

Jeppe Jensen

Simon J. K. Pedersen

CONTENTS

CHAPTER 1 INTRODUCTION

The properties and limitations of the Human Visual System (HVS) can be beneficial in various contexts. For instance flashing stimuli are easily perceived and are therefore often used to capture the attention, e.g. in emergencies [Helen Sharp, 2007]. Another example of a property of the HVS is that the human color perception is most sensitive to green light. This is utilized in digital cameras where a Bayer filter is 50 % green, and 25 % red, and 25 % blue [Brown, 2004].

Visual perception can also be utilized in the context of computer graphics. If rendering is based on properties of the HVS it is referred to as perceptually adaptive rendering. A subtopic of perceptually adaptive rendering is gaze directed rendering where gaze information acquired by an eye tracker is used.

When perceptually adaptive rendering, or specifically gaze directed rendering, is applied the typical purpose is to improve the quality of the rendering. This can either be by improved visual quality, increased frame rate, or a combination. But these improvements come at a cost, usually the aim is to design a system where the costs result in either visually imperceptible degradation or functional imperceptible degradation [Murphy and Duchowski, 2007]. Visually imperceptible degradation is defined as an image quality degradation that is undetectable by the HVS, while functional imperceptible degradation is a degradation that does not influence the time it takes to complete a particular task with the system.

One common approach to perceptually adaptive rendering is not to render graphics that can not be perceived, e.g. avoid rendering objects with more polygons than perceivable. With the knowledge of the gaze point this approach can be improved further, to render better graphics around the gaze point and lower quality graphics at the peripheral of the vision without it being visually perceptible. Another approach, that requires the gaze point of the user, is to simulate visual effects that are dependent of where the user is looking, e.g. glare and bleaching effects.

The initial hypothesis of this project is based on two assumption: 1) It is not possible to render photorealistic graphics in real-time. 2) The human visual system is unable to perceive everything on a screen. These assumptions are the motive for the following initial hypothesis:

> *The human visual system is not capable of perceiving everything that happens on a screen. Thus the quality of a graphics rendering can be improved by using an eye tracker to track the gaze point, and concentrate the use of computational resources to render the area around the gaze point and use less resources in the peripheral of the vision, without it being perceptible to the eye.*

In the next chapter a pre-analysis of perceptually adaptive rendering is presented to outline ideas on how eye-tracking can be used in combination with perceptually adaptive rendering.

CHAPTER 2

PERCEPTUALLY ADAPTIVE RENDERING

This pre-analysis chapter will review perceptually adaptive rendering. First the human visual system is described in terms of how it perceives visual information and what limitations there are. Secondly the concept of eye tracking as well as the used eye tracker will be presented.

As the outline of the project is to investigate the possibilities of perceptual adaptive rendering a brief survey of different computer graphics rendering methods and visual realism effects suitable for this purpose is conducted. A perceptually adaptive rendering method or visual realism effect is from this survey selected as the method of interest for the rest of the report.

2.1 Human Visual System

[Duchowski, 2007, Hornberg, 2006, Reddy, 1997, Ware, 2000]

This section contains an introduction to some of the most fundamental concepts and elements of the Human Visual System (HVS) to give an understanding of how the human perceives visual information. Understanding the HVS is crucial when working with perceptually adaptive rendering, because perceptually adaptive rendering is ultimately about deceiving the HVS.

2.1.1 Visual Angle

Visual angle is a concept that can be used to describe a certain amount of the field of vision. The concept is of particular importance when dealing with perceptual rendering because the resolution of the visual field is related to the visual angle. The visual field spans up to 180 degrees along the horizontal axis and 130 degrees along the vertical axis. But already at a visual angle of five degrees is the acuity reduced to

50 %, thus the useful field of vision is typically said to span only 30 degrees along the horizontal axis.

The magnitude of the angles used in relation to the HVS are often of such small magnitude that representing them in degrees make little sense, therefore minutes and seconds are often used instead, see Table 2.1.

Table 2.1: Representations of angles.

Unit	Value	Symbol
degree/angle	$\frac{1}{360}$ circle	$^\circ$
minute	$\frac{1}{60}$ degree	$'$
second	$\frac{1}{60}$ minute	$''$

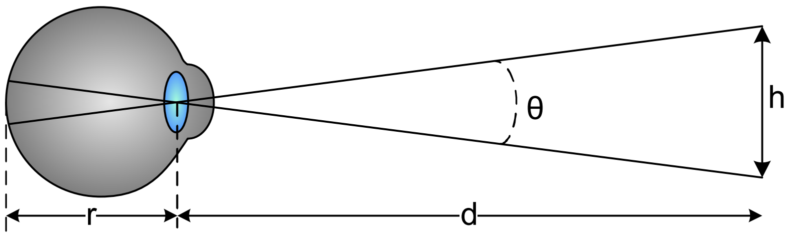


Figure 2.1: The distance, d, to an object and the height, h, of the object can be used to calculate the visual angle, θ, [Duchowski, 2007].

The visual angle, θ, needed to observe an object of size h at a distance of d, see Figure 2.1, is calculated as

$$\begin{aligned} \tan\left(\frac{\theta}{2}\right) &= \frac{h}{2}\cdot\frac{1}{d} \\ \theta &= 2\arctan\left(\frac{h}{2d}\right) \end{aligned} \tag{2.1}$$

As an example, consider a 22” computer monitor with a resolution of 1680×1050 pixels. The 22” monitor have a screen diagonal, d_{mm}, of 558.8 mm and the number of pixels per mm, $ppmm$, is calculated by

$$\begin{aligned} d_{px} &= \sqrt{w_{px}^2 + h_{px}^2} \\ ppmm &= \frac{d_{px}}{d_{mm}} \end{aligned} \tag{2.2}$$

This result in 3.55 pixels per mm or approximately 0.28 mm per pixel. If the monitor is located 650 mm from the eyes each pixel corresponds to 1.5 minutes.

The 2 degrees field of vision corresponds to 22.7 mm when the distance is 650 mm, while the entire useful field of vision of 30 degrees span 348 mm from the same distance.

2.1.2 The Eye

The eye serves as a visual sensor for humans capturing the surroundings, and is therefore by many compared to a camera. This comparison is good for pedagogical reasons since a camera and an eye have some key-concepts in common, but in details they are quite different. They share the concepts of a lens, an aperture, and a film, for the eye the later two are known as the pupil and retina,. In a camera the film contains an image of the surroundings, the same is not valid for the retina. Much of the processing is still left for other parts of the visual pathway and the visual cortex, before a human gets a final perception of the surroundings. The details of the visual cortex and the parts of the visual pathway is left out of this description of the HVS and the focus is on knowledge useful to perceptually adaptive graphics.

When working with eye tracking definitions of the different kinds of eye movement is useful. The most rapid eye movement is called saccades and is made when the point of regard is changed. Saccades have a duration of between 10 and 100 ms, with saccades of 10 ms the eye tracker and rendering system should have a frame rate of 200 frames per second (fps) for a saccade to go undetected, at least in theory. Another often used eye movement paradigm is smooth pursuits where the eyes smoothly follows an object in motion. The last and most common eye movement pattern is fixation, it is used about 90 % of the viewing time. It is used when looking at a static object too large for the visual angle of the fovea to contain the entire object. Fixations are rather slow typically with a duration between 150 and 600 ms.

Retina

The retina is located in the back of the eye and consists of approximately hundred million photoreceptor cells which are excited by the incoming light. The excitation of the cells triggers nerve impulses which are processed by the visual pathway and the visual cortex and leaves us with a perceptual view of the world.

The retina consists of two types of cells which each have different properties: rods, which constitute to about 95 % of the cells and cones. The rods are very sensitive to low light levels, this is a result of the long integration time (200 ms) over which the rods are excited. The integration time of cones is a lot shorter (20 ms) thus the cones are the most used cells during daylight. Even though the cones are a minority they are the most important for the human vision. This is illustrated by most of the cones being densely packed in a small center of the retina called the fovea. The fovea area is where the vision is best. Typically the fovea is said to cover a visual angle of two degrees, but the highest visual resolution is obtained only within a 0.5 degrees center of the fovea, the foveola.

An illustration of the distribution of the rods and cones as a function of the visual field is seen in Figure 2.2. The distribution of cones explains why the HVS is particular good only within two degrees of the fovea and why the useful field of vision only extends to 30 degrees.

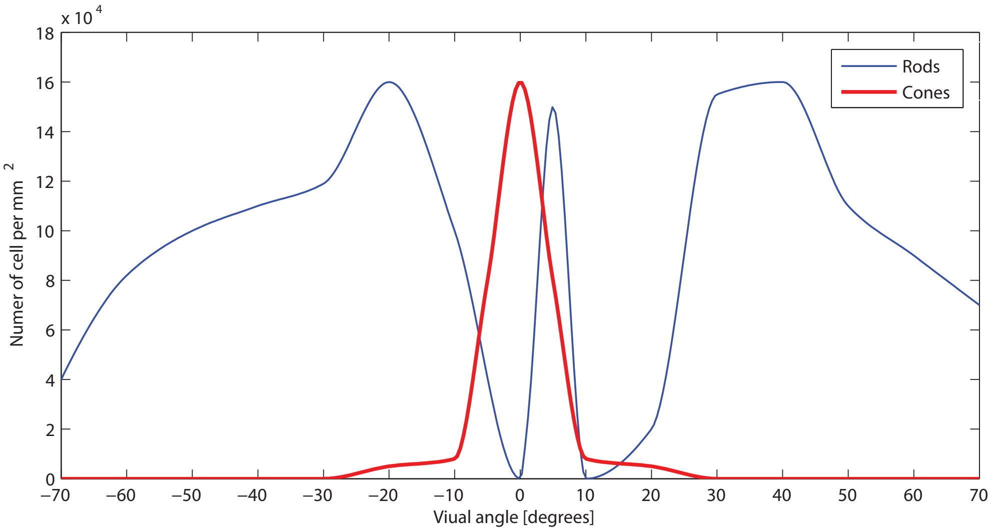


Figure 2.2: The density of the rod and cone cells as a function of the visual angle. Between 10 and 20 degrees visual angle is the so called blind spot, where the nerves leave the eye. Approximated from [Duchowski, 2007].

2.1.3 Acuity

The visual acuity is a measure of our ability to see details, loosely it can be described as the sampling frequency of the eye. The most well known acuity test is the Snellen test performed at the doctor when different sized letters/symbols have to be determined from a distance. The acuity is closely related to the spacing of the cells in the retina, such that we are only able to perceive objects which have a size twice the spacing the Nyquist limit. In visual angles this corresponds to approximately 1 minute.

But the acuity can be empirically determined to be even better than suggested by the Nyquist limit, this phenomenon is known as superacuity and is caused by the retina not acting as a simple CCD array. Each cell in the retina is interconnected with a number of neighbors that signals the unified result of the cell neighborhood to the subsequent parts of the visual pathway, resulting in better than single cell acuity. One particular test that can be used to demonstrate this is the Vernier acuity test, where two horizontal lines can be determined to be unaligned with as little spacing as 10 seconds.

2.1.4 Contrast Sensitivity

Although it might often be acuity that is associated with measurement of the HVS, another concept known as contrast sensitivity is also important. The contrast sensitivity function can be used to establish a relationship between the smallest contrast visible as a function of spatial frequency. The contrast is defined as

$$C = \frac{L_{max} - L_{min}}{L_{max} + L_{min}} \tag{2.3}$$

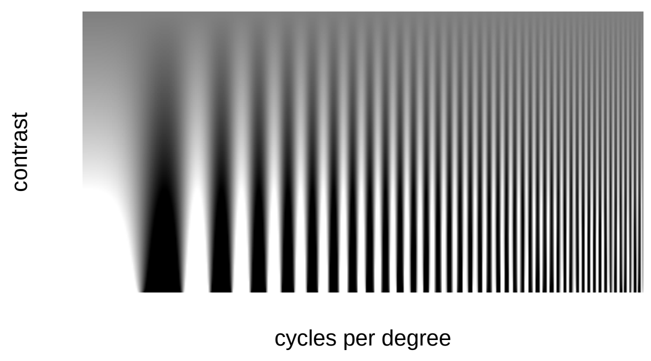


Figure 2.3: A grating pattern used to measure the contrast sensitivity of the HVS.

where:

L_{max} and L_{min} is the maximum and minimum illumination.

From empirical tests of the contrast sensitivity of the HVS, e.g. see Figure 2.3, it has been determined that it is most sensitive to gratings with a high contrast with a frequency of 2 to 3 cycles per degree visual angle. Gratings with lower frequency are less noticeable and gratings with a frequency of 60 cycles per degree (for young people, less for older people) are not perceivable by humans.

2.1.5 Vision Properties

This section groups together different properties of the human vision. These are temporal vision, color vision, depth of field, adaptation, glare, and bleaching.

Temporal

The temporal vision aspect of the HVS is used when perceiving moving objects.

As mentioned the cells in the retina have different integration times, these times determine the properties of our temporal vision. For instance a flashing stimulus will only appear flashing if the frequency is below 50 Hz. This knowledge is exploited in for instance cathode ray tube televisions where the actual frame rate only is 25 fps, but due to interlacing (odd and even scan lines are updated asynchronously) the temporal vision is tricked into believing that the actual frame rate is 50 Hz, i.e. no flicker is visible.

As it was the case with acuity the temporal vision is not constant over the 180 degrees field of vision. The best motion detection capabilities are found in the outer regions of the field of vision.

Color

Although colors might seem like an important aspect of the HVS, it is actually quite insignificant. Colors are neither used to determine motion, object shape, nor distance, their sole purpose are to spice up our perception of the world and to help distinguish otherwise identical objects.

Color perception is determined by three different types of cone cells, each is particular sensitive to a certain wavelength, namely S cones at 440 nm, M cones at 535 nm, and L cones at 565 nm, see Figure 2.4. The three cell structure of the eye is also the reason why many color models use a 3-dimensional model, like RGB, CMYK. and HSV.

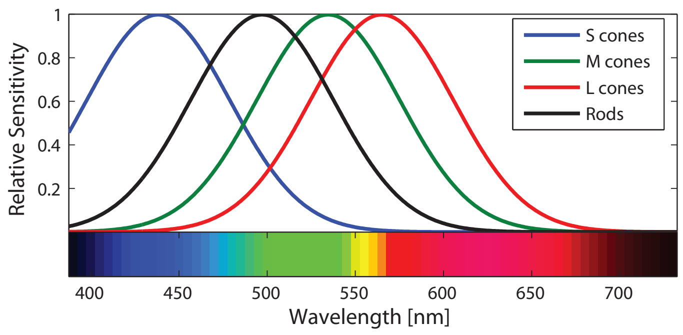


Figure 2.4: The visible spectrum and the different types of cones. (Note the cones are not normal distributed as depicted, this is a simplification). [Ware, 2000].

Since the cones are the primary color receptors, the ability to perceive colors is best inside the fovea and decrease towards the edge of the vision field.

Depth of Field

The human eye as well a camera lens can not attend the focus on an entire scene at once. The world appears sharp in the gaze region around the point of regard. The objects located in the front of and behind the point of regard appear blurred, this effect is called Depth of Field (DoF).

Adaptation

The HVS is capable of coping with a wide luminance range from bright sunlight to darkness, see Figure 2.5. The process the HVS uses to adapt to different luminance levels is called adaptation, and the adaptation to new light conditions happens at a finite rate. The effect of a finite adaptation rate can be observed when entering a dark room, as our ability to see in darkness improves over time. This effect is known as dark adaptation and the time course from not adapted to darkness to fully adapted takes up to 35 minutes. The reverse effect when moving from a dark environment to a bright environment is known as light adaptation. The light adaptation is significantly faster, for the rods 80 % of the sensitivity

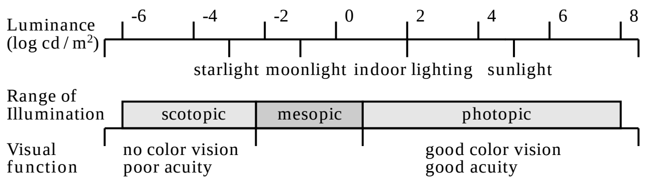
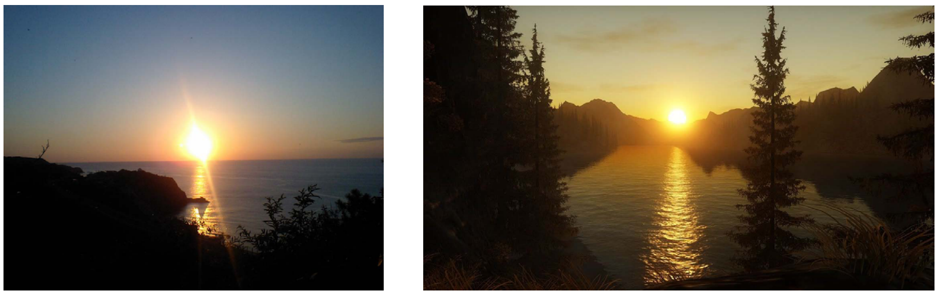


Figure 2.5: The range of luminance the HVS is capable of working under. [Ferwerda et al., 1996]



(a) Real sunset over Japan. (b) Computer generate sunset in Remedy's Alan Wake game.

Figure 2.6: Two sunset examples with visible glare effect. [Yoshida et al., 2008]

is regained within the first 200 ms, while the cones adapt in a magnitude of minutes. [Ferwerda et al., 1996]

Glare

Glare is visible as the pronounced light effect that surrounds very bright objects, see Figure 2.6. It is caused by scattering and diffraction in the eye and in particular the lens [Spencer et al., 1995]. The glare effect of the HVS is also known for cameras as lens flare.

Glare is composed of two components: bloom and flare lines. Bloom, often referred to as veiling luminance, creates the glow of light surrounding the object. Bloom results in reduced contrast around very bright objects.

The flare lines are responsible for the bright light rays and the rainbow colored concentric circles that encircle the object. The colored circles are caused by the crystalline structure of the lens, and the effect is known as lenticular halo, see Figure 2.7 and Figure 2.8.

Bleaching

[Gutierrez et al., 2005]

Eye bleaching is a phenomenon that is far less distinct than the other vision properties described here. It happens when exposed to high levels of luminance, and it results in a ”washed out” color vision,

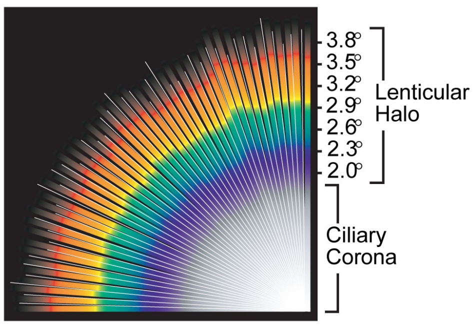


Figure 2.7: Bright rays emanate from the ciliary corona while the lenticular halo produces the colored circles. [Spencer et al., 1995]

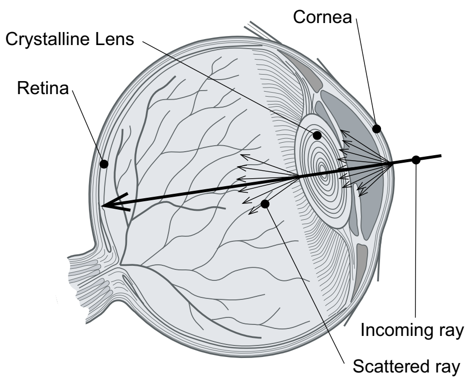


Figure 2.8: The elements of the eye that constitute to the creating of glare. [Spencer et al., 1995]

where perceived colors are distorted. The cause of the effect is a density change of the photoreceptor cells of the eye, in particular the cones. This changes the spectral sensitivity of the cones such that they are excited by a smaller range of visible light, see Figure 2.4 on page 18, where the variance of the distributions would decrease.

This concludes the review of aspects of the HVS important to perceptually adaptive rendering.

2.2 Eye Tracking

This section briefly introduces eye tracking concepts and technology, and finally presents the particular eye tracker used.

Eye tracking or Gaze-Contingent Display (GCD) systems can broadly be categorized into interactive and diagnostic systems [Nikolov et al., 2004]. In interactive systems the gaze of a user is used as input to control the system. While in diagnostic systems the gaze is used to get information about where the attention of the user is directed.

2.2.1 Eye Tacking Technology

Generally eye tracking technology can be categorized into invasive and non-invasive systems. The most common invasive system is head mounted. Head mounted systems can either offer full eye tracking or the gaze direction can be estimated from the position of the head. Non-invasive systems do typically rely on computer vision to determine the gaze direction from images of the head. These images can either be traditional images, infrared images or a combination. The huge advantage of non-invasive systems are that they are not uncomfortable to the user, but unfortunately most non-invasive systems in turn require the head to be in a relative fixed position. In some systems this is solved by using a chin rest to stabilize the head.

2.2.2 Tobii X120 Eye Tracker

The eye tracker provided for this project is a Tobii X120 eye tracker. This eye tracker is a stand alone eye tracker that can be connected to a computer. It is a non-invasive eye tracker that requires no head stabilization, see Figure 2.9. The Tobii X120 delivers an updated gaze position with a 120 Hz data rate, and it has an accuracy of 0.5 degrees [Tobii, 2007]. To interface with the eye tracker Tobii provides a stand-alone software solution and a Software Development Kit (SDK).

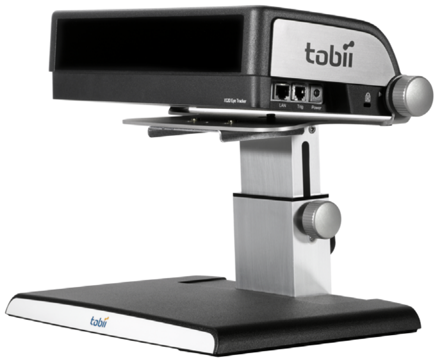


Figure 2.9: The Tobii X120 Eye Tracker [Tobii, 2007].

The eye tracker uses near infrared diodes to generate reflections on the corneas of the eyes of the user. Image data of these reflections and of the person is used to identify relevant features, like the reflections in the eyes and the eyes itself. The positions of the eyes in world space are calculated and projected as a gaze point onto the screen.

With this brief introduction to eye tracking and presentation of the used eye tracking hardware, it is time to investigate related research.

2.3 Related Research

In this section previous research within the field of perceptually adaptive rendering will be examined. The purpose of this is to provide a review of the perceptually adaptive rendering topics that is found interesting for this project, thus the examination is not meant to be exhaustive in any respect.

The first topic reviewed is perceptually adaptive global illumination techniques. Secondly research on visual realism is reviewed, before finally reviewing level of detail based on perception.

2.3.1 Global Illumination

In this section perceptually adaptive research within global illumination will be presented. Five of the most common global illumination algorithms that also may be suitable to use with an eye tracker will be presented. The algorithms are: radiosity, ray-tracing, path tracing, final gathering, and photon mapping.

A brief introduction to global illumination and the five algorithms can be found in Appendix A on page 153.

Radiosity

Several research articles have been published on the topic of perceptually adaptive radiosity. Purgathofer presents an overview of the articles published in the nineties where most of the research in the radiosity field was carried out. Purgathofer made a clear distinction between importance driven radiosity and perceptually driven algorithms. (Importance driven algorithms are algorithms where the optimizations are carried out from an object importance perspective, which does not necessary map well to perception.) In the following only perceptually driven algorithms are considered, and these are divided into object-space and image-space based methods. [Purgathofer, 1999]

Two object-space based algorithms exists [Gibson and Hubbold, 1997, Hedley et al., 1997], which both uses hierarchical radiosity. Before the radiosity is calculated a temporary perceptual transformation from calculated luminance to display intensity is established, this is done in order to optimize calculations of radiosity in areas where it cannot be perceived.

The image based algorithms by Myszkowski and Martin et al. used an approach similar to the object-space based methods. But instead of pre-estimating, a coarse radiosity simulation is conducted, which is subsequently used to refine the hierarchy of patches and then redo the simulation until some sort of convergence is obtained. [Martin et al., 1997, Myszkowski, 1998]

It has not been possible to find publications where radiosity is combined with the gaze position acquired by an eye tracker.

Ray-Tracing

Of the global illumination techniques mentioned here then ray-tracing is the one that has received most attention in combination with eye tracking.

The first attempt to combine eye tracking technology and ray-tracing was by Levoy and Whitaker. In their system the gaze direction is used to vary the amount of rays and samples per ray used to trace the scene, based on local retina acuity. This is done such that an area around the gaze direction is rendered in a higher quality than the rest of the scene. The specific rendering technology that is being optimized by using the gaze direction in their paper is volume rendering, and the gaze direction is tracked by a Head Mounted Device (HMD). [Levoy and Whitaker, 1990]

A more recent paper on gaze directed rendering, which also embody ray-tracing is ”Hybrid image-/model-based gaze-contingent rendering” by Murphy and Duchowski. In their work artificial face images with different facial expressions are rendered with a hybrid render for a user driven search task. The hybrid rendering uses both rasterization and ray-tracing. Ray-tracing is used in an area within a proximity of the gaze point. Five different visual angles are used during testing to determine the size of the ray-traced area, namely 2°, 5°, 10°, 15°, and 20°. Within the ray-traced area the number of rays to trace is found by an empirically determined contrast sensitivity function. The function basically reduces the amount of rays needed as the distance to the gaze point increases. The tests are focused on functionally imperceptible degradation of applying gaze directed rendering to the search task. The results showed that the time to complete the search task increased with smaller visual angels, and especially the 2° angle gave a significant increase in search time. [Murphy and Duchowski, 2007]

Path Tracing

Path tracing with a perceptual aspect is not a very wide-spread research topic. A path tracing algorithm that partly uses perception was developed by Tamstorf and Jensen. Their approach is to use perception to determine the number of samples per image pixel. The way they determine when to terminate sampling of a pixel is by the tolerance of the display device, such no additional paths are traced for a given pixel, if the improved precision cannot be displayed, and thus perceived. [Tamstorf and Jensen, 1997]

The article by Lai et al. also uses perception in their Population Monte Carlo path tracing algorithm. That is done by determine the sampling rate per image pixel by the perceptual variance, such that pixels that are a part of a perceptual varying region are sampled by more paths than pixels of regions with less variance. [Lai et al., 2007]

Both of the above mentioned approaches could be combined with the gaze position from an eye tracker. It has not been possible to find any research publications where the gaze position is used to control the sampling rate.

Final Gathering

The research focusing on final gathering in a perceptual context is very limited. Often final gathering is used with photon mapping to give a higher quality rendering, therefore it may be used in coherence with the photon mapping research, see next section. E.g. Suykens and Willems mentioned that their approach could be improved with final gathering. [Suykens and Willems, 2000]

Photon Mapping

It has not been possible to find any research that directly uses photon mapping together with eye tracking. But research has been made where visually important regions are populated by more photons than less important regions.

Some of the first to make use of importance driven photon mapping were Peter and Pietrek. They constructed importance information in form of an importance map containing the amount of visual importance in scene, where the visual importance is a measure of how much light that leaves a point in the direction of the receiver/camera. By combining the importance information with the direction of the photons, it is possible to direct more photons to the most visually important regions. Furthermore the importance map is used for the construction of the Probability Density Function (PDF) used when photons are reflected from the surface. Their research showed that the use of importance driven construction of photon maps leads to better results. [Peter and Pietrek, 1998]

Later research by Suykens and Willems introduced the concept of density control for the photon map. Density control limits the density of photons in very bright regions of the scene, because these regions may not be visually important, and thereby more photons can be used in darker regions that are visually more important. If the density limit is attained the photons are distributed among existing nearby photons in the photon map. The use of density control gave images of the same quality as the standard photon mapping method but with the number of photons reduced by a factor 2-4. [Suykens and Willems, 2000]

The research of Keller and Wald introduced an importance driven photon mapping algorithm similar to that of Peter and Pietrek. But instead of controlling the emission and scattering directions from the importance as Peter and Pietrek did, they control the deposition of the photons. By doing that they were able to obtain images of the same or better quality, but with reduced memory usage. [Keller and Wald, 2000]

This finalize the review of the global illumination algorithms, the following sections will review research of some rendering effects that can be used to improve frame rate or visual quality of perceptually adaptive renderings.

2.3.2 Visual Realism

The global illumination rendering algorithms described so far are based on the physics of light, for more details consult Appendix A on page 153. Although the laws of physics are obeyed, it is often fairly easy to distinguish a rendered version of a scene with the real scene. This is because the human visual system does not act as a perfect render, it introduces some artifacts that a traditional render does not introduce. Another concern is the limited dynamic range of the display device that might not be able to reproduce the large variety of luminance of the real scene. These two problems must be considered in order to produce vision realistic images.

The limited dynamic range of the display devices can be solved in two ways. One method is to get display hardware with a better dynamic range, so called High Dynamic Range (HDR) displays, this is currently an area of development [Seetzen et al., 2004]. Another approach is contrast compensation, also known as tone mapping, which is the process of reducing the dynamic range from typically 32 bit per channel in the rendered image to the 8 bit per channel of the display device as visually convincing as possible.

This section will describe computer graphics research publications related to some of the vision properties outlined in Section 2.1.5 on page 17. The purpose is to be able to render visual realistic images by using computer graphics to simulate HVS effects, or artifacts if seen from the perspective of a perfect render. It is a hypothesis that the gaze position provided by the eye tracker can be used to render these artifacts with improved realism.

Depth of Field

This sections revise research publication where eye tracking to locate the point of regard is used to simulate the Depth of Field (DoF) in computer graphics.

Mulder and Liere have made a fast perception based DoF rendering. To make the DoF rendering fast two different techniques were used – one with high accuracy around the point of regard, and one with less accuracy but high speed at the outer area. Their work is a theoretical description of the two techniques and how they can be used, but is not used together with an eye tracker. [Mulder and Liere, 2000]

Depth of field is already used in the computer game industry, for instance the Unreal game engine (from version 2) provides DoF effects [Hillaire et al., 2007]. However, this does not use the real gaze point of the user but is limited, for instance, to use the aim of the crosshair as gaze point.

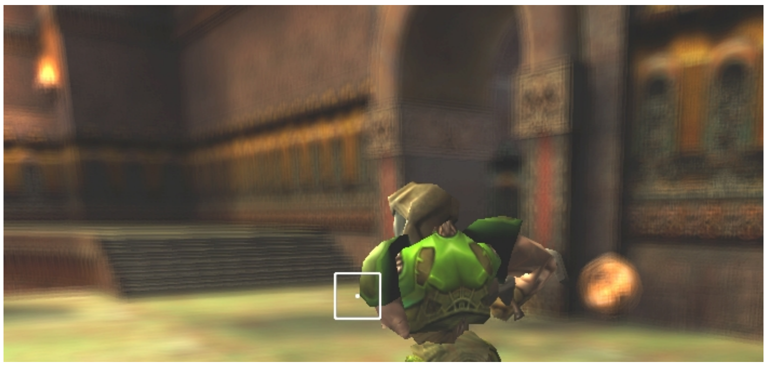


Figure 2.10: The Quake 3 computer game with depth of field effect implemented. The white square is the gaze point measured by an eye tracker. [Hillaire et al., 2008]

Hillaire et al. have implemented a depth of field blur effect, in IdSoftware's computer game, Quake 3 Arena, see Figure 2.10. In their first work eye tracking was not used. Instead the center of the screen was used as gaze point, because in a first person shooter game most of the attention is focused on the center of the screen [Hillaire et al., 2007]. A test was made to decide whether DoF was considered a positive or negative effect by the users. The results showed that DoF had no significant effect on the performance of the users, and nearly half of the participants preferred the DoF blurring effect rather than no effect. The main reason why the other half disliked it, was that the gaze point not always was on the center of the screen, which made it harder to find and identify the enemies. In the latest research by Hillaire et al. an eye tracker was used to locate the point of regard of the user. To detect the gaze direction and locate the point of regard a head-mounted eye tracker, of the model ASL eye-trac 6000, was used. Some tests were performed where eight participants should rate their experience playing the game with and without DoF blurring. Ratings (from 0 to 7) were given according to four different conditions: rendering realism, ”fun”, depth perception, and immersion in the virtual world. The results showed that, the participants gave all four condition a significant higher rate with DoF blurring than without. [Hillaire et al., 2008]

Adaptation

The adaptation of the HVS can be simulated in computer graphics, e.g. racing games typically apply some sort of adaptation when driving into a tunnel with reduced illumination. The adaptation used in this type of application is typically not a simulation of the behavior of the HVS due to the slow adaptation rates. One assumption that greatly simplifies applying adaptation is if the entire view can be assumed to be under the same illumination conditions. Removing this assumption requires a method to get the gaze position, something an eye tracker can provide.

The research of Ledda et al. outlines a method for applying adaptation to HDR images such that local contrast regions are preserved without being over or under exposed. Their method can also be applied to an image sequences in order to produce the effects of dark and light adaptation, see Figure 2.11. [Ledda et al., 2004]

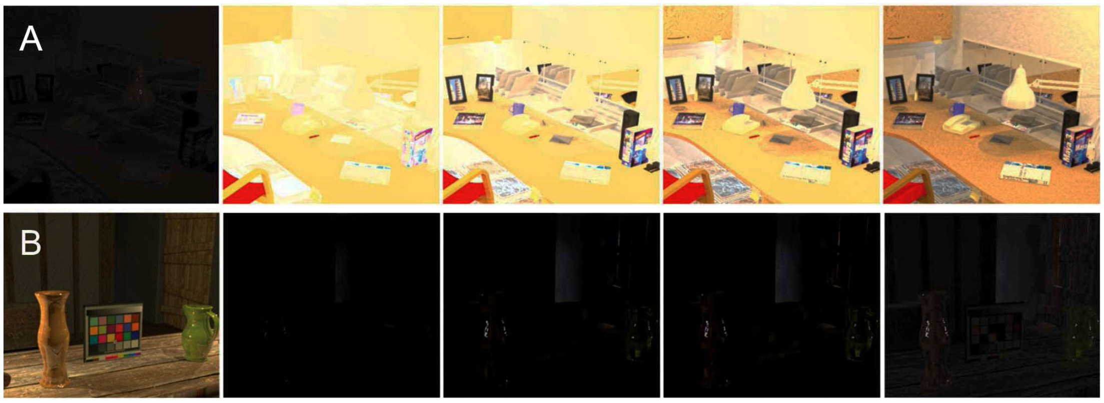


Figure 2.11: A) Light adaptation. B) Dark adaptation. Brightness adjusted for better visibility. [Ferwerda et al., 1996]

Glare

The main reason for adding glare to HDR renderings is to enhance the perceptually realism by creating an illusion of higher dynamic range than supported by the display device [Ritschel et al., 2009, Yoshida et al., 2008].

Glare algorithms can be categorized into physiology based and perceptually based glare algorithms. Algorithms of the physiology category are based on the physiologically properties of the HVS, which often involves a computational heavy point-spread function. Spencer et al. proposed to use this point-spread function to generate a glare billboard which cheaply during runtime could be superimposed onto bight light sources to give the desired glare effect [Spencer et al., 1995]. More recently Ritschel et al. proposed another physiology based glare algorithm that is capable of creating real-time dynamic glare simulations on Graphics Processing Unit (GPU) hardware. The temporal aspect of their glare algorithm is based on the dynamic behavior of the extrinsic variables: pupil size and the position of the eye-lashes which are simulated in real-time. Instead of superimposing the glare map onto the scene image a convolution is used which further improves the perceptual glare illusion, see Figure 2.12.

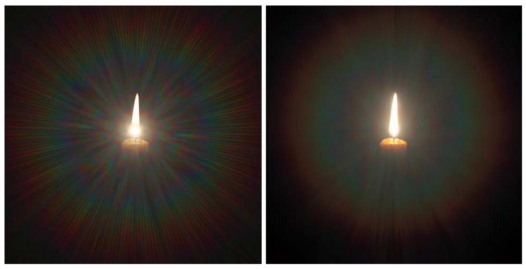


Figure 2.12: The left image shows the glare effect using a point-spread function as developed by Spencer et al.. The right image shows the glare as proposed by Ritschel et al..

Applications with real-time glare effects have typically used perceptually based algorithms. In these algorithms the point-spread function is substituted with a Gaussian function to obtain better frame rates. Perceptually this has negligible implications for small light sources, but for larger sources the Gaussian filter introduces exaggerated luminance veiling.

It is a hypothesis that by coupling glare algorithms with a gaze position acquired by an eye tracker, it is possible to improve the render quality of glares in close proximity of the gaze position. Another possible improvement is to use the eye tracker to estimate the pupil size something which is important in the physiology based algorithms.

Bleaching

Eye bleaching is the last visual realism effect that will be considered here.

The amount of computer graphics related research on this topic is very limited. Gutierrez et al. proposed in 2004 an ad-hoc model to add bleaching, while they later in 2005 developed an algorithm based on physiology for the same purpose. The second model established a relation between the size of the pupil and the amount of bleaching. In order to estimate the size of the pupil the amount of luminance at the gaze point had to be known, in their work they predefined the gaze point, but with the use of an eye tracker this can be calculated dynamically. [Gutierrez et al., 2005, 2004]

2.3.3 Level of Detail

To reduce the computational load in computer graphics 3D objects can be rendered with different Level of Detail (LoD) without it being perceptible to the user. For instance objects far away from the viewer or objects that are moving fast can be rendered with a lower level of detail. Most often level of detail is applied on the geometry, by reducing the number of polygons used to construct an object – 3D objects with a low polygon number have a low level of details.

Back in 1976, Clarke, J.H. was the first to introduce rendering of a scene with lower polygon number for small or distant objects. His approach, which is still a common approach, was to preprocess the objects with different polygon count, and in run-time decide which one to use, based on for instance the distance from the objects and to the viewer. [Clarke, J.H., 1976]

Some of the first to work with gaze-directed LoD using the approach described above was Funkhouser and Sequin, they aimed to find which combination of LoD and rendering algorithm that gave the perceptual best image within a target frame time. To do that they presented two heuristics. A cost heuristic which estimated the amount of time required when using a certain LoD and rendering algorithm, and a benefit heuristic which estimated the perceptual contribution, primary based on object sizes but also the gaze point of the user. They did not track the users actual gaze position but made an assumption that the user was looking at the center of the screen. Their tests showed that the algorithm generated a more uniform frame rate than previously described algorithms, with only a little noticeable difference in image quality of complex objects. [Funkhouser and Sequin, 1993]

Later Ohshima et al. made use of an eye tracker to find the gaze direction of the user. They used the gaze direction to determine LoD, with a high LoD at the gaze area and lower at the outer area. This quality reduction at the peripheral of the field of vision was only slightly detectable, but the frame rate was improved by a factor of five. [Ohshima et al., 1996]

The drawback of the LoD approach where the objects are preprocessed with different polygon count, is that one object can not have variable LoD, i.e. that a big object can not have a high LoD at one point an low LoD in another point. If the user is looking at an object the entire object must be rendered with the same LoD. Later another and more adaptive approach for LoD was introduced, where objects can span multiple LoD at the same time. Some of the first to introduce view dependent adaptive LoD was Xia and Varshney, Hoppe, and Luebke and Erikson, where objects or parts of objects that are invisible for the viewer (for instance out of the view frustum, or back facing regions) were rendered with a low LoD. All the propounded algorithms used a hierarchy of vertices merge operations that can be applied or reversed at run-time. [Hoppe, 1997, Luebke and Erikson, 1997, Xia and Varshney, 1996]

More recent research by Luebke et al. have used an eye tracker with a chin rest to ensure accuracy, to locate the gaze direction of the user. The objects were rendered with a variable polygon count as a function of the distance to the gaze point, see Figure 2.13. Their test showed that they were able to achieve imperceptible simplification of objects in the outer area. [Luebke et al., 2000]

The first to use a head mounted eye tracker, instead of a chin rested eye tracker, for adaptive LoD

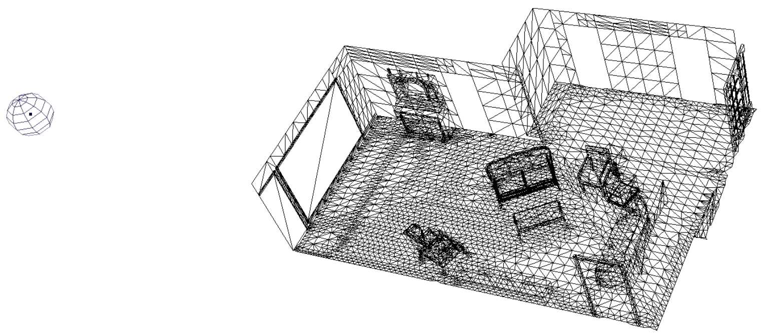


Figure 2.13: A scene of a living room with a higher LoD near the gaze point. The sphere to the left is the gaze point. [Luebke et al., 2000]

was Hunter Murphy and Andrew T. Duchowski. A test was performed to find the highest detection threshold where the degradation of LoD was not perceptible. By using this threshold an improvement in the average frame rate was achieved with a factor of four, compared to the non-gaze directed LoD. [Hunter Murphy and Andrew T. Duchowski, 2001]

Parkhurst and Niebur performed a test on how gaze directed LoD rendering affects the detection and localization time of objects in a virtual environment. Where detection time refer to the time it takes a user to find a target object, and localization time is the time for panning the center of the screen to be align with the detected object. Their results showed that the detection time was increased as the LoD decline per degree was increased, one of the reasons for this was that the eye-tracker used for the experiment was not fast enough, i.e. the user could move their gaze point faster than the eye tracker could detect it. On the other hand the localization time was decreased as the LoD decline per degree was increased, because the frame rate was increased in correlation to the LoD level. The total search time (detection plus localization time) was decreased as the LoD decline per degree was increased. [Parkhurst and Niebur, 2004]

2.4 Summary

Understanding the human visual system is important when developing a perceptually adaptive graphics rendering system. From the theory outlined it was found the human visual system can be expected to be most accurate within a region as little as five degrees visual angle, and that the useful field of view spans 30 degrees. Eye movement concepts like saccades and fixations were also explained, together with some properties of the human visual system.

The chapter also contained an introduction of the eye tracker used in this project, the Tobii X120. In a later test of the eye tracker accuracy, the HVS and eye movement concepts will be investigated further.

The section on related research reviewed five global illumination algorithms with relation to perceptually adaptive rendering. For most of the algorithms only little research had been carried out with respect to the elements of human perception. E.g. the radiosity, path tracing, final gathering, and photon mapping algorithms have never to our knowledge been coupled with the gaze direction acquired by an eye tracker.

In other areas of computer graphics eye tracking and perception are more wide-spread, this was case for both level of details and depth of field. This trend however was not present for the research on the vision properties: adaptation, glare, and bleaching. Here most research turned out to be limited to the perceptually aspects without trying to incorporate eye tracking.

CHAPTER 3 PROBLEM DESCRIPTION

Based on the analysis of perceptually adaptive rendering in the preceding chapter, this section will outline the choice of research direction together with a formal problem statement.

3.1 Choice of Research Direction

Based on the investigation of related research, and especially lack of research in some areas, a topic has been selected for additional analysis. The topic that will be focus of attention in the remaining parts of the report is perceptually adaptive photon mapping based on the gaze position acquired by the eye tracker.

Although the lack of research within the field perceptually adaptive photon mapping made it an interesting research topic, other motivations existed as well. Firstly, photon mapping was found particular interesting as a learning topic. Secondly, it is also one of the global illumination techniques that is capable of creating the best visual quality. And this superior quality does not come with a huge degradation in performance, as e.g. path tracing. Thus a lot of factors were in favor of working with perceptually adaptive photon mapping.

3.2 Problem Statement

The hypothesis of this project is that it is possible to improve computer graphics by combining traditional computer graphics algorithms with the gaze position acquired by an eye tracker.

This improvement can be in form of better frame rates or better perceived quality of the renderings without sacrificing frame rate. The later can be obtained by using high quality rendering methods such as global illumination, in particular photon mapping, at the gaze area where they have most influence on perception and neglect quality at the peripheral of the vision where it has little to no influence on perception.

To examine the hypothesis a test scenario is investigated in detail. This test scenario is based on the following problem statement.

- *How can photon mapping be combined with the gaze position acquired by an eye tracker, in order to improve the rendering performance without perceivable degradation of quality?*

This problem statement can be split into three general areas of research that different sections of the remaining chapters of the report will deal with. These are

- Acquisition of gaze position with the Tobii X120 eye tracker.
- Combination of gaze position and photon mapping into perceptually adaptive photon mapping.
- Performance improvements without perceptually degradation of quality.

The acquisition of the gaze position and the accuracy and precision hereof is dealt with in the Preliminary Tests, see Chapter 4 on the next page. Different approaches to perceptually adaptive photon mapping and their theoretical performance improvement is investigated form a design perspective in Design, see Chapter 7 on page 73. Before the final System Test, see Chapter 9 on page 127, demonstrates whether the designed and implemented algorithm for perceptually adaptive photon mapping, solves the problem statement.

CHAPTER 4 PRELIMINARY TESTS

The first preliminary test that is going to be conducted is a test of the eye tracker accuracy in order to determine the error margins for subsequent tests.

After the first accuracy test a calibration algorithm for improving the accuracy is devised and tested.

The final and most important purpose of the preliminary test is to investigate to what extend the properties of the HVS, Section 2.1 on page 13, can be accepted. The property of particular interest is to find the size, measure in visual angle, of the gaze area in which changes to the surroundings are detectable.

4.1 Eye Tracker Accuracy

The purpose of this test is to verify the accuracy of the utilized eye tracker, model: Tobii X120. The product description [Tobii, 2007] provides only a little information about the accuracy of the eye tracker. The accuracy is a measure of how correct the eye tracker detects the gaze point. In addition to accuracy, the precision will be tested, where the precision refers to how constant the error is at different positions on the screen, measured in standard deviation of some sample points. E.g. the precision is high if the standard deviation of the samples is small independent of position of the gaze point.

The initial hypothesis for the test is that he eye tracker conforms to the product specifications, with an accuracy of 0.5 degrees visual angle.

4.1.1 Test Setup

A test, very similar to the accuracy test performed by the eye tracker manufacturer [Tobii, 2007], is carried out. A test subject is placed 650 mm from a 22 inches widescreen monitor with the head aligned with the middle of the monitor. For each test subject, the system is calibrated using Tobii's calibration software, after which the test subject must, one point at the time, focusing at 64 evenly distributed points on the screen while the eye tracker is detecting the gaze point, see Figure 4.1. Each point remains on the screen for 2.2 seconds before it vanishes and the next point appears. For each point 10 samples of the gaze point of the person is captured from the eye tracker. From all the sample points the accuracy of the eye tracker is computed. The test is performed on eight different persons.

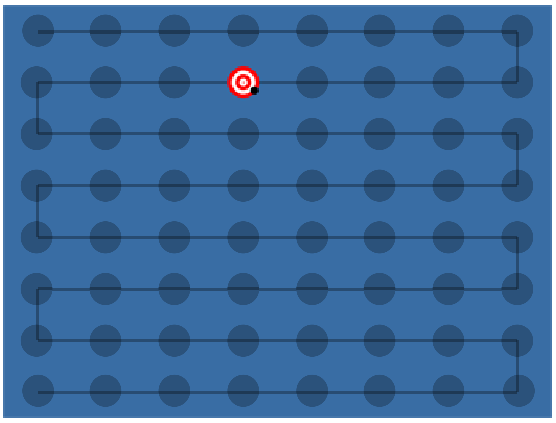


Figure 4.1: The red and white dot is where the person must focus, and the small black dot is where the eye tracker detects the eye gaze. The faded pattern is where the point appears one position at the time.

4.1.2 Results

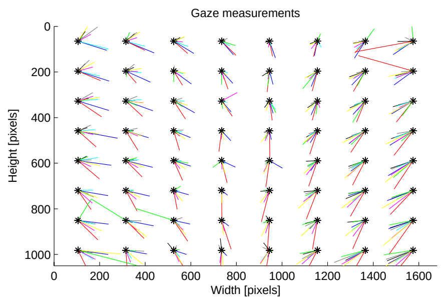
The difference between the captured sample points and the actual position of the drawn points can be calculated and evaluated to find the accuracy of the eye tracker. An absolute standard deviation (accuracy) and a relative standard deviation (precision) are computed. The absolute standard deviation, is the standard deviation for the sample points with a mean equal to the drawn point on the screen. The relative standard deviation is where the mean value is the mean of the samples. The unit for these measures is computed both in millimeters and in degrees of the visual angle.

Figure 4.2 shows where the eye tracker measures the gaze at each point for each test subject. From each of the 64 point a line is drawn to the mean point of the 10 measured sample points.

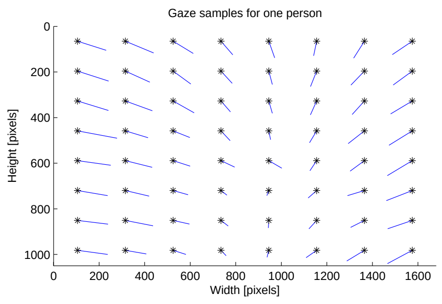
From Figure 4.2 it can be seen that the eye tracker is most accurate at center of the screen. At the left and right border of the screen the eye tracker tends to measure the gaze position towards the center of the screen, and in general the eye tracker measures the gaze at a too low position.

The amount of the error at the different points can be seen in Figure 4.3.

In the Figure 4.3 it can be seen that the plot has eight equally distributed peaks. A peak occurs every eighth point, with the first peak at point 1 and the second at point 9 etc. All these points correspond to the border of the screen and all the bases are at the middle of the screen, which confirms that the eye



(a) The mean sample at each point for all test subjects.



(b) All the samples from one test subject.

Figure 4.2: The measured samples for each test point on the screen.

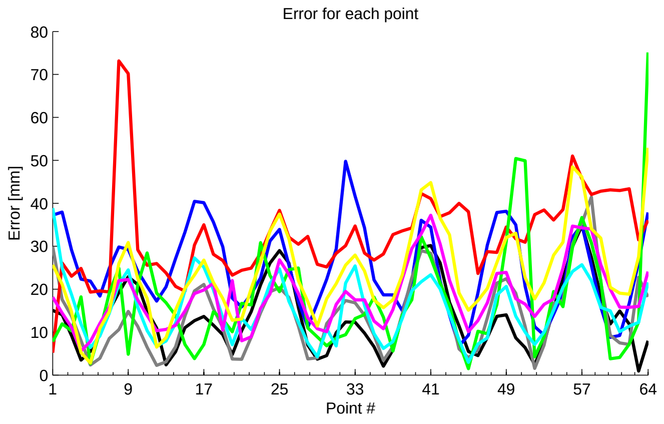


Figure 4.3: The error of the mean sample at each test point for all eight test subjects.

tracker is most accurate at center of the screen and gets worse as the distance to the center increases.

The product description of the eye tracker states that the eye tracker is very accurate, but the only numerical value stated is an accuracy of 0.5 degrees. For a better comparison of the accuracy, the accuracy in degrees is computed for the test results, see Table 4.1. The degrees are found as a conversion of the error in millimeters.

Table 4.1: The eye tracker accuracy in degrees and millimeters.

	Mean	STD	Min	Max
Degrees	1.7576°	0.5824°	1.1944°	2.8687°
Millimeters	19.95 mm	6.61 mm	13.55 mm	32.57 mm

The measured worst case accuracy is 3.5 times worse than the stated accuracy of 0.5 degrees, even the most accurate result is more than twice as bad as the stated accuracy.

Figure 4.4 shows the relative standard deviation, as the mean of the relative standard deviations for each test subject. This is also a measure of the precision of the eye tracker.

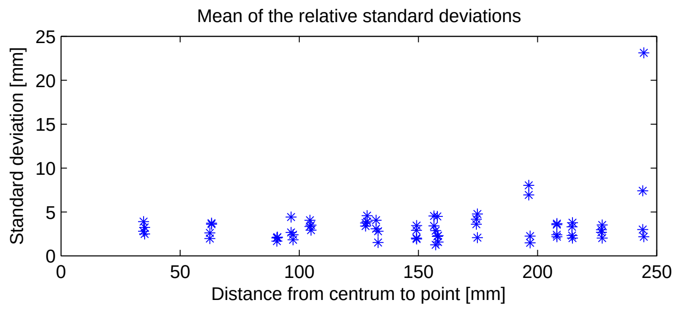


Figure 4.4: The mean of the relative standard deviation of all the test subjects arranged after distance from the point and to center of the screen.

From the figure the standard deviation seems to be independent of the distance to the center, except of some outliers at 200 and 250. Which means the precision of the eye tracker is the same across the screen, which is equal to a high precision.

4.1.3 Discussion

The initial hypothesis that the accuracy of the eye tracker was within 0.5 degrees was not demonstrable. Instead the accuracy turned out to be rather poor, 3.5 times worse than the accuracy claimed by the manufacture. The tests also showed that the level of accuracy was dependent on the distance to the center of the screen, where the best accuracy was measured.

The precision on the other hand was high as shown by the standard deviation of the sample distance for samples belonging to the same point was small. The standard deviation of the distance for samples belonging to points located in the same distance from the center was close to constant.

The lack of accuracy might pose a problem for the future use of the eye tracker, thus the following test will propose a method for improving the accuracy.

4.2 Eye Tracker Calibration

The purpose of this section is to devise an algorithm that can be used to calibrate the eye tracker and thus improve its accuracy. The desired calibration algorithm should be capable of generating a general calibration map for a particular eye tracker setup, such that each individual user can skip an exhaustive calibration step.

The test carried out aims to determine whether the proposed algorithm is capable of generalizing, and determines the amount of accuracy improvement.

4.2.1 Calibration Algorithm

The calibration algorithm used to map each pixel position to a calibrated position is similar to the bilinear interpolation. Instead of interpolating pixel values calibration vectors are interpolated. For each of the 64 evenly distributed points (the points used for the accuracy test) a calibration vector $\mathbf{v}_{ij}$ is computed from the test data. Any given point/pixel $p = (x_p, y_p)$ on the screen is mapped to a new position that is defined by the calibration vector $\mathbf{v}_p$, which is computed as a weighting of the calibration vectors $\mathbf{v}_{ij}$ of the four nearest known points. The four neighbor points are denoted, $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$, and $Q_{22} = (x_2, y_2)$, see Figure 4.5.

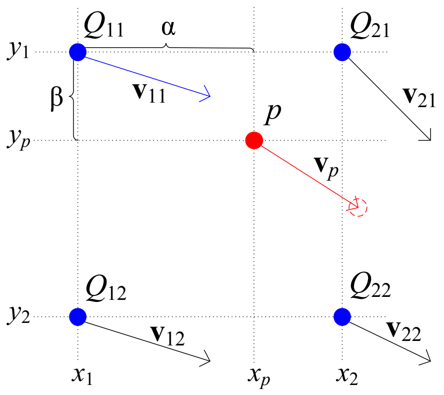


Figure 4.5: The four blue dots are the known points with their corresponding calibration vectors $\mathbf{v}_{ij}$, and the red dot is the point that must be calibrated with the vector $\mathbf{v}_p$.

The weightings of the four vectors are found as the normalized distance α and β (in the x- and y-direction) from p and to each of the Q points, where points nearest to p is weighted highest.

$$\alpha = \frac{x_p - x_1}{x_2 - x_1}, \qquad \beta = \frac{y_p - y_1}{y_2 - y_1} \tag{4.1}$$

The weights are multiplied with the four calibration vectors $\mathbf{v}_{ij}$, to find the calibration vector of p, $\mathbf{v}_p$

$$\mathbf{v}_p = (1-\alpha)\cdot(\mathbf{v}_{11}\cdot(1-\beta) + \mathbf{v}_{12}\cdot\beta) + \alpha\cdot(\mathbf{v}_{21}\cdot(1-\beta) + \mathbf{v}_{22}\cdot\beta) \tag{4.2}$$

Vector $\mathbf{v}_p$ is added to the position of the point p, which gives a new position that is the position that the point must be mapped to. This mapping is calculated for all points/pixels on the screen (1680×1050).

The pixels in the corner regions will only have one adjacent Q-point and the pixels at the border of the screen will only have two adjacent Q-points. For the corner pixels is $\mathbf{v}_p$ equal to the calibration vector of the single adjacent point. For the pixels at the top and bottom only the α weight is used to calculate $\mathbf{v}_p$, and for the pixels at the sides only the β weight is used.

4.2.2 Test Setup

A calibration map created from the results of the first accuracy test is sought used to improve the accuracy of the eye tracker. Each test subject in the new test completes one test with the calibration map enabled and one without. The purpose of this test is to determine whether a general calibration map is a suitable method for improving the accuracy of the eye tracker.

The outcome of the test is two data sets, a calibrated set and an original set without calibration. Each set contains ten samples for the each of the 64 test points for two different persons.

For each test point $p_i, i = 1, \ldots, 64$ a mean error vector $\mathbf{e}_i$ is found as the vector between the ground truth position of the point $p_i = (x_i, y_i)$ and the mean position of the samples $\bar{s_i}$

$$\bar{s_i} = \frac{1}{NP} \sum_{j=1}^{NP} s_{ij} \tag{4.3}$$

where:

$s_{ij} = (x_{ij}, y_{ij})$.
N is the number of test samples.
P is the number of test subjects.

$$\mathbf{e}_i = \bar{s_i} - p_i \tag{4.4}$$

The hypothesis is that the length of the error vector of the calibrated data set, $|\mathbf{e}_{i,calib}|$, is smaller than the length of the error vector of the original data set, $|\mathbf{e}_{i,org}|$. Rigorously speaking, the accuracy is improved by applying the calibration method.

$$\begin{aligned} \text{H0} &: |\mathbf{e}_{i,calib}| - |\mathbf{e}_{i,org}| \geq 0 \\ \text{H1} &: |\mathbf{e}_{i,calib}| - |\mathbf{e}_{i,org}| < 0 \end{aligned} \tag{4.5}$$

The selected null hypothesis is the contrary of the claim that the applied calibration method improves the accuracy. This way a strong evidence for the claim can be ensured by rejecting the null hypothesis at a chosen significant level. The hypothesis will be accepted or rejected based on a paired t-test.

4.2.3 Results

The precondition for paired t-test is first and foremost that the data must be normally distributed. To test for this a normplot of the difference, $|\mathbf{e}_{i,calib}| - |\mathbf{e}_{i,org}|$ is shown in Figure 4.6. In addition to the normplot a Lilliefors goodness-of-fit test is computed [Abdi1 and Molin, 2007], in order to provide stronger evidence for the data being normally distributed.

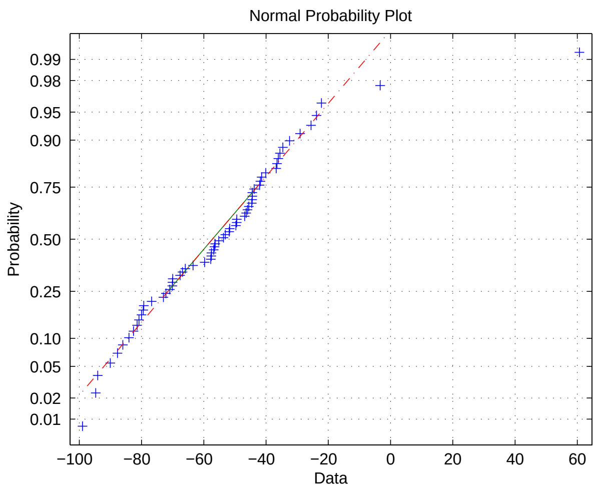


Figure 4.6: Normplot of the difference between the length of the error vector before and after calibration. The normplot indicates that the differences can be assumed to be normally distributed as desired for the paired t-test.

The Lilliefors test accepts the hypothesis that the data is normally distributed with a p-value of 0.15, thus for any significant level less than 0.15 the hypothesis would be accepted.

With the preconditions fulfilled, the paired t-test can be computed for the hypothesis shown in Equation (4.5). The null-hypothesis is rejected for any reasonable significant level, since the p-value is calculated to $2.3 \cdot 10^{-26}$, i.e. the calibration does improve the accuracy.

A more illustrative view of the results are presented in Figure 4.7.

4.2.4 Discussion

Significant accuracy improvements are obtainable by applying the proposed calibration algorithm. The amount of improvement is presented in Table 4.2. The accuracy improvement gained by applying the calibration compared to using the original eye tracker data is 60 %. This improvement is obtained by using the general calibration map computed from just eight test subjects, thus it can be concluded that the proposed algorithm can provide generalized calibration data.

Table 4.2: The eye tracker accuracy in degrees and millimeters for the calibration test data.

	Mean Calibrated	STD Calibrated	Mean Original	STD Original
Degrees	0.9230°	0.64°	2.2929°	0.66°
Millimeters	10.47 mm	7.32 mm	26.03 mm	7.50 mm

The runtime performance degradation incurred by applying the calibration is close to non-existing. This

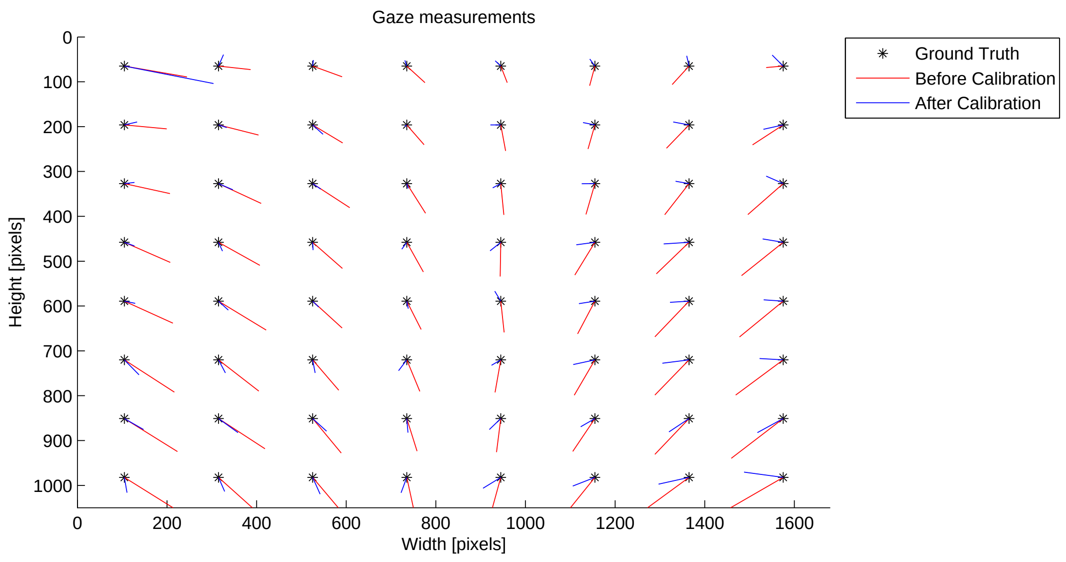


Figure 4.7: The mean error vectors plotted before and after applying the calibration.

is achieved by precomputing all the calibration data, effectively reducing the runtime cost to a memory lookup and a vector addition.

4.3 Visual Perception

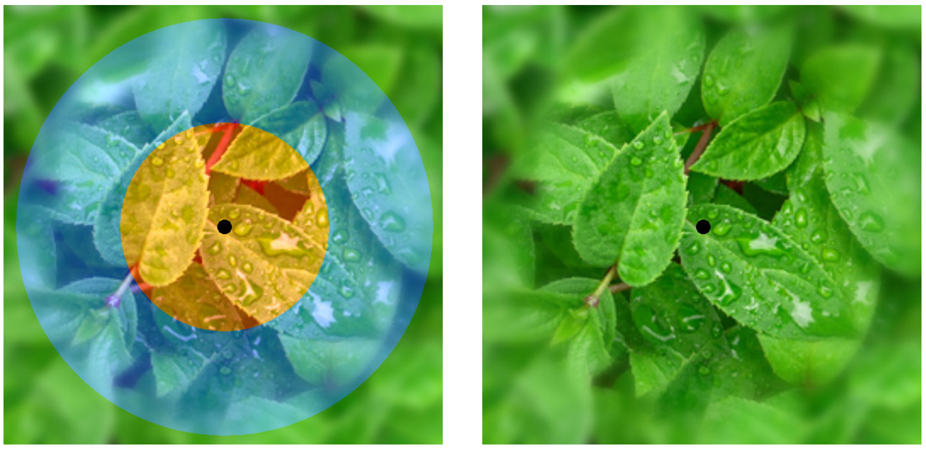
This purpose of this test is to determine to what extent inaccuracies in computer graphics is imperceptible to the HVS. This knowledge can be exploited to render computational complex graphics to only a part of the screen, when used together with an eye tracker, without the user noticing.

The effect that is used to simulate inaccuracies is Gaussian blur. The results achieved with this effect are not guaranteed to be generalized, thus alternative effects might give different results. An alternative effect could be color to gray scale, where only a small region of the screen is rendered in color with a smooth transition to gray scale at the remaining parts of the screen. Another approach is to have high quality rendering within the small region, e.g. by using ray-tracing, and render the remaining in a lower quality.

It is expected that the test subjects will only be able to perceive around 2 degrees visual angle when forced to focus on a particular point. This initial hypothesis is based upon the knowledge about the HVS presented in Section 2.1 on page 13.

4.3.1 Test Setup

The test subject is asked to focus on a small dot moving across a picture on the screen and keep the gaze point on this particular dot. The movement is carried out along a predefined path, which goes in a horizontally zig-zag pattern from top to bottom, similar to that of Figure 4.1 on page 34. While the dot moves a blur window contracts toward the dot, when the user detects the blur he or she should abort the test and the dot will stop moving. From the size of the inner region without blur the visual angle can be obtained. This visual angle determines the size of a circular area where inaccuracies in graphical rendering would be perceived.


(a) No blur is applied to the inner region (yellowish shaded area). The blue shaded region contains a gradient from no blur to full blur.

(b) An example of what the user sees during the test.

Figure 4.8: Visual perception test images.

The radius of the non-blurred area, see Figure 4.8, is the measurement of interest in this test. In order to make the transition between the non-blurred region and the outer blurred background image as undetectable to the HVS as possible a smooth gradient across the outer ring is created. The width of the outer ring is equal to the radius of the non-blurred area. The background image is created as a seamless texture of an image of small leaves which creates an image with many edges that makes the effect of smoothing more prominent, see Figure 4.10.

The initial size of the non-blurred area is selected to match the useful field of vision of 30 degrees, which at a distance to the screen of 650 mm corresponds to a radius of 174 mm. The velocity of the dot is chosen fast enough to keep the HVS occupied such that eye saccades are avoided and only smooth pursuit is used to follow the dot. The used dot movement velocity is 90 mm/s, while the contraction velocity of the non-blurred region is 6 mm/s. In order to detect whether saccades have occurred, the actual gaze point will be tracked with the eye tracker during the test.

The test is carried out on a 22 inch monitor with a resolution of 1680$\times$1050 and the parameters in Table 4.3 assume that this resolution is used.

Table 4.3: Parameters used for the visual perception test.

Parameter	Value
Head to screen distance	650 mm
Blur amount	Gaussian 9×9/15×15 kernel
Initial non-blur radius (inner radius)	$30^\circ \approx 174$ mm
Contract velocity of blur window	6 mm/s
Velocity of dot	90 mm/s

4.3.2 Results

The results are obtained from 8 different test subjects. Each positioned with the head at a distance of 650 mm from the center of the screen, and the eye-tracker individually calibrated. The region with the non-blurred area moves along the predefined pattern and the users are told to focus on the dot in the center of the non-blurred area. When the users observe a change in the surroundings of the dot they are told to terminate the test by pressing the space bar. The results of the tests are shown in Figure 4.9.

Test Observations

The first two users were tested with a 9×9 Gaussian blur kernel, see Figure 4.10(b). This amount of blur was not enough for any of the two users to detect the change in the surroundings of the dot during the initial run. Before the second run the two users were instructed to look for blur around the dot, and then both users were able to see the change. Consequently the amount of blur was increased to a 15×15 kernel, see Figure 4.10(c).

The remaining test subjects were now able to detect the change in the surroundings on the initial run. Although, as the results show, the visual angle at which the detection occurs is different from person to person. Also the amount of focus on the dot varies, as seen by the gaze to dot distance, see Figure 4.9.

4.3.3 Discussion

It is not possible from the limited number of test subjects to determine the exact visual angle wherein changes to the surroundings are visible. The results indicated that a change to the surroundings at a visual angle greater than 20 degrees would be imperceptible to most people, since most of the test subjects first detected the blur when it entered between 10 and 20 degrees visual angle. A visual angle of 20 degrees corresponds to a radius of 123 mm, which on the used monitor would cover as much as 33 % of the screen area.

Another conclusion that can be made from the results is that even though the gaze point of some of the test subjects fluctuate outside the non-blurred region it is not certain that they will detect the blur, see

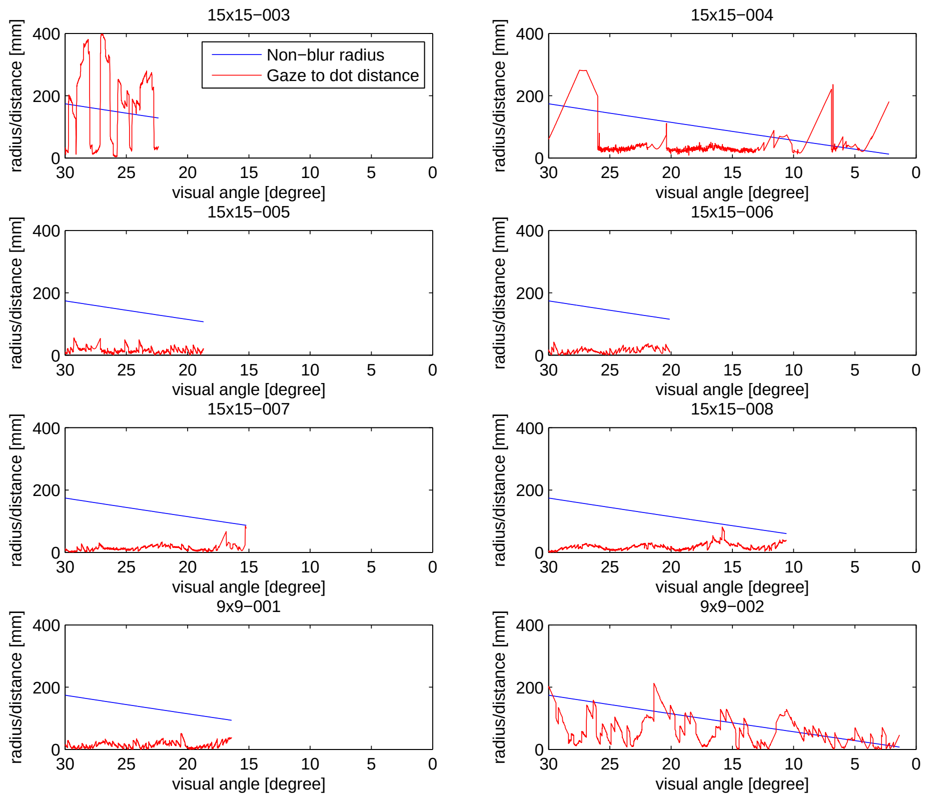
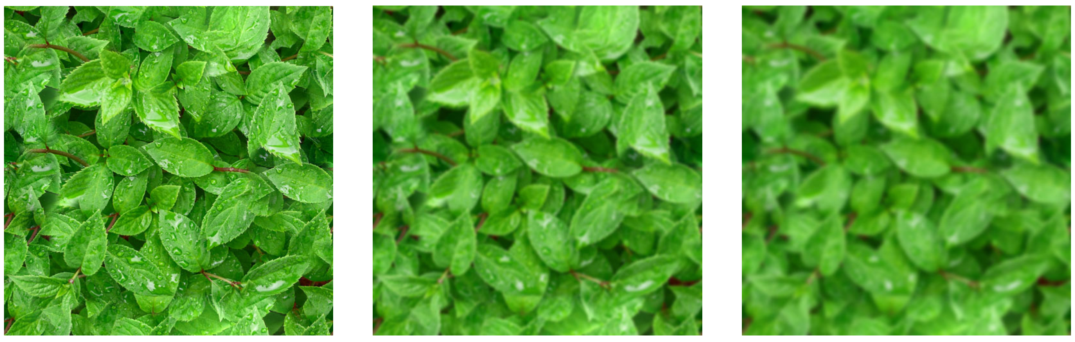


Figure 4.9: The results of the visual perception test. The red curve is the distance from the tracked gaze position to the center of the non-blurred circle. The blue curve represents the radius of the non-blurred area. When the user detected the blur they terminated the test, this is why the graphs end at different visual angles.



(a) No blur. (b) Kernel size 9×9. (c) Kernel size 15×15.

Figure 4.10: The effect of changing the kernel size on the used texture.

e.g. plot -003 or -004 in Figure 4.9.

It is also possible to reject the initial hypothesis, since the results show that the HVS, in this test, is able to perceive inaccuracies with more than two degrees visual angle.

4.4 Summary

From the preliminary test it was found by the initial accuracy test of the eye tracker that the worst case accuracy was around 3.5 times worse than the stated 0.5 degrees visual angle. The poor accuracy seemed to be a result of systematical errors in the gaze tracking, this was observable as the accuracy decreased as the distance to the center of the screen increased. Overall the precision of the measurements were close to constant, thus only the accuracy was sought to be improved by the devised calibration algorithm.

A calibration algorithm based on bilinear interpolation, was used to create a general calibration map that for a particular eye tracker setup could be used to improve the accuracy. Tests showed that from accuracy data acquired from eight test subjects, it was possible to create a calibration map that would improve the accuracy by 60 %. Thus in subsequent uses of the eye tracker this calibration map will be applied.

The final test of the visual perception, turned out to be different from the initial hypothesis. The hypothesis was that changes to the surroundings would only be detectable within a region of two degrees visual angle from the gaze point. The test results did not backup this claim. For most of the test subjects changes to the surroundings of the gaze point would be detectable at between 20 and 10 degrees. With the limited number of test subjects it was not possible to find an exact size of the region, but it was clear that a region much larger than two degrees have to be considered if changes are to be imperceivable to the HVS. This finding is most unfortunate since it limits the performance gain that can be expected from perceptually adaptive graphics.

CHAPTER 5

ILLUMINATION THEORY

The purpose of this chapter is to introduce some common and needed terms used in computer graphics, especially when building a global illumination render. First the basis of light measurement is introduced in the radiometry section. After this introduction light surface interaction models are described before concluding the chapter with an example of how radiance values are calculated in the developed system.

5.1 Radiometry

[Dutre et al., 2002, Jensen, 2001, Pharr and Humphreys, 2004]

Radiometry is the term used to describe the task of measure physical light in a scene, where physical light is described by different terms. The most important term in the context of computer graphics is radiance, because it is the quantity used to determine the color of a pixel. Radiometry involves the entire electromagnetic spectrum, but in computer graphics only the spectrum of visible light is interesting, which are in the frequency range of 380 nm to 780 nm, see Figure 2.4 on page 18.

There are some radiometric terms that are fundamental to rendering. These are radiant energy, radiant flux, irradiance, radiosity, and radiance. They will be described in the following.

5.1.1 Radiant Energy

Radiant energy Q is the basic unit of light energy. A light source emits a number of photons, where each photon has a radiant energy e_λ If a light source emits n_λ photons with wavelength λ, the spectral radiant energy Q_λ for that wavelength can be found as

$$e_\lambda = \frac{h \cdot c}{\lambda} \quad [\mathrm{J}] \tag{5.1}$$

where:
$h \approx 6.63 \cdot 10^{-34}$ J $\cdot$ s is Planck's constant.
$c = 299,792,458$ m/s is the speed of light in vacuum.
λ is the wavelength of the photon.

$$Q_\lambda = n_\lambda \cdot e_\lambda = n_\lambda \cdot \frac{h \cdot c}{\lambda} \tag{5.2}$$

The total radiant energy Q for a light source can now be found as the sum of Q_λ for all possible wavelengths λ. This is computed by integration as

$$Q = \int_0^\infty Q_\lambda d\lambda \tag{5.3}$$

All the equation and calculations in the rest in the radiometry section is also wavelength dependent, but this is not explicitly stated.

5.1.2 Radiant Flux

Radiant flux or radiant power Φ is the total amount of radiant energy passing through space or a surface per unit time

$$\Phi = \frac{dQ}{dt} \quad [\mathrm{W}] \tag{5.4}$$

Radiant flux density u is a measure of the amount of radiant flux arriving or leaving per unit area on a surface, where the surface can be real or imaginary, i.e. it can be measured anywhere in the space

$$u(x) = \frac{d\Phi}{dA} \quad \left[\frac{\mathrm{W}}{\mathrm{m}^2}\right] \tag{5.5}$$

where:
Φ is the radiant flux arriving/leaving the point x.
dA is the differential area surrounding the point.

The radiant flux that are arriving from any direction within a hemisphere to a point is also called irradiance E. The radiant flux that is leaving in any direction from a point is called radiant exitance M, also known as radiosity B.

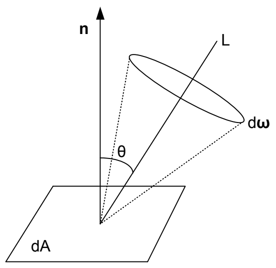


Figure 5.1: Calculation of the radiance whether it is incident or excident [Jensen, 2001].

Radiant intensity I is the radiant density per unit solid angle $\boldsymbol{\omega}$

$$I(\boldsymbol{\omega}) = \frac{d\Phi}{d\boldsymbol{\omega}} \quad \left[\frac{\mathrm{W}}{\mathrm{sr}}\right] \tag{5.6}$$

5.1.3 Radiance

The radiance L is defined as the radiant flux per unit projected area per unit solid angle. Radiance can be described as the amount of radiant flux contained in a ray of light that arrives or leaves a point on a surface in a given direction, see Figure 5.1.

The radiance of a ray intersecting a surface at the point x with the direction $\boldsymbol{\omega}$ is given by

$$L(x,\boldsymbol{\omega}) = \frac{d^2\Phi}{\cos\theta dA d\boldsymbol{\omega}} \quad \left[\frac{\mathrm{W}}{\mathrm{m}^2\mathrm{sr}}\right] \tag{5.7}$$

where:

θ is the angle between the ray and the surface normal, see Figure 5.1.

The radiance can be calculated for both incident and outgoing rays, and is constant along a line of sight (in vacuum), i.e. it does not attenuate with the distance. If the radiance is given all of the other terms can be calculated by integrating the radiance.

E.g the radiosity as

$$B(x) = \int_{\Omega_{\text{surface}}} L_o(x,\boldsymbol{\omega}_o)\cos\theta d\boldsymbol{\omega} \tag{5.8}$$

where:

Ω_{surface} is a surface encasing the point, typically a hemisphere.

$L_o(x,\boldsymbol{\omega}_o)$ is the outgoing radiance from point x.

$\boldsymbol{\omega}_o$ is the output direction.

The irradiance can be calculated as

$$E(x) = \int_{\Omega_{\text{surface}}} L_i(x,\boldsymbol{\omega}_i)\cos\theta d\boldsymbol{\omega} \tag{5.9}$$

where:

$L_i(x, \boldsymbol{\omega}_i)$ is the incident radiance at point x.

$\boldsymbol{\omega}_i$ is the incident direction.

5.1.4 Light Emission

If the power Φ_s of a point light source encased in a sphere of radius r is known, and it emits uniformly in all directions, then the irradiance E_{source} at the inside of the sphere is calculated as

$$E_{\text{source}}(x) = \frac{\Phi_s}{4\pi r^2} \cos\theta \qquad \left[\frac{\text{W}}{\text{m}^2}\right] \tag{5.10}$$

where:

r is the radius of the sphere encasing the light source.

Since the irradiance is calculated inside a sphere from the centrum $\cos\theta$ is equal to 1 and can thereby be neglected.

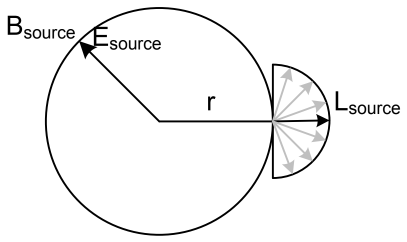


Figure 5.2: Sphere light encased in a perfect diffuser.

The radiosity is equal to the irradiance $B_{\text{source}} = E_{\text{source}}$, when the encasing sphere is a perfect diffuser without absorption, see Figure 5.2. By solving L_o in Equation (5.8) the outgoing radiance, $L_o = L_{\text{source}}$ for an arbitrary point on the surface is calculated as

$$L_{\text{source}}(x) = \frac{E_{\text{source}}(x)}{\pi} = \frac{\frac{\Phi_s}{4\pi r^2}}{\pi} = \frac{\Phi_s}{4\pi^2 r^2} \qquad \left[\frac{\text{W}}{\text{m}^2\text{sr}}\right] \tag{5.11}$$

By the definition of radiance it is known that radiance is constant along a line of sight in vacuum. Thus the incoming radiance L_i, at a distant point x in line of sight of the light source is equal to L_{source}.

5.2 Light Surface Interaction

[Dutre et al., 2002, Jensen, 2001]

The interaction between light and a surface is responsible for the distinct appearance of different materials. E.g. consider the different appearance of a wooden table and a brick. Different models have been developed to accurately model material appearances in computer graphics. Three basic types of interactions are typically considered: refraction, reflection, and absorption. These types can coexist in the same material, thus it is a difficult task to design a single model that can cope with all aspects of material appearances.

The reflectance ρ, also known as albedo, expresses how much of the incoming radiant power that is reflected

$$\rho(x) = \frac{d\Phi_r(x)}{d\Phi_i(x)} \quad [-] \tag{5.12}$$

The albedo must be between zero and one for physically based materials, and for values less than one the remaining must be either refracted or absorbed.

One of the most popular functions used for surface reflections in computer graphics is the Bidirectional Reflectance Distribution Function, BRDF.

5.2.1 Bidirectional Reflectance Distribution Function

The BRDF is a simplification of the Bidirectional Surface Scattering Reflectance Distribution Function, BSSRDF. The BSSRDF is capable of explaining the interaction with surfaces where incoming light hits the surface at one point but exits it at another, while the BRDF is limited to surfaces where the incoming and outgoing point are identical, see Figure 5.3.

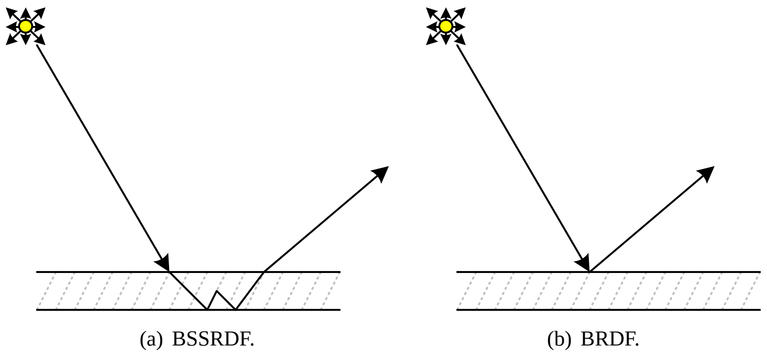

(a) BSSRDF. (b) BRDF.

Figure 5.3: The difference between BSSRDF and BRDF.

The BRDF f_r, is formally defined as the relationship between differential reflected radiance dL_r, and differential irradiance dE, at a surface point x, for an incoming $\boldsymbol{\omega}_i$, and outgoing $\boldsymbol{\omega}_o$ direction

$$f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o) = \frac{dL_r(x,\boldsymbol{\omega}_o)}{dE(x,\boldsymbol{\omega}_i)} \quad \left[\mathrm{sr}^{-1}\right] \tag{5.13}$$

With the definition of BRDF it is possible to determine the reflected radiance $L_r(x,\boldsymbol{\omega}_o)$ over the entire unit hemisphere of a surface point from the incident radiance from all directions within Ω.

$$L_r(x,\boldsymbol{\omega}_o) = \int_\Omega f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)dE(x,\boldsymbol{\omega}_i) = \int_\Omega f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)L_i(x,\boldsymbol{\omega}_i)\cos\theta d\boldsymbol{\omega} \tag{5.14}$$

If the material itself does not emit any light, this equation is also called the rendering equation. Does the material have emission properties then the rendering equation for the total outgoing radiance L_o at a point on a surface becomes

$$L_o(x,\boldsymbol{\omega}_o) = L_e(x,\boldsymbol{\omega}_o) + L_r(x,\boldsymbol{\omega}_o) \tag{5.15}$$

where:

$L_e(x,\boldsymbol{\omega}_o)$ is the self emitted radiance.

The BRDF has some important properties namely the property of reciprocity and energy conservation.

The law of reciprocity states that the BRDF is unchanged by interchanging the incident and outgoing light direction (This is basically what makes ray-tracing possible)

$$f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o) = f_r(x,\boldsymbol{\omega}_o,\boldsymbol{\omega}_i) \tag{5.16}$$

While the law of energy conservation states that the energy leaving a point must be equal to or less than the energy arriving

$$\int_\Omega f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)\cos\theta d\boldsymbol{\omega} \leq 1 \quad ,\forall\boldsymbol{\omega} \tag{5.17}$$

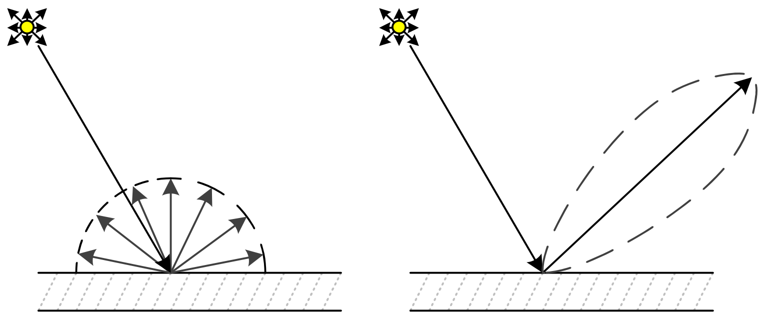

(a) Lambertian surface (b) Specular surface

Figure 5.4: Surface properties.

The BRDF depends on the surface properties, see Figure 5.4. For a Lambertian (perfect diffuse) surface, the BRDF is given as

$$f_{r,d} = k_d = \frac{\rho_d(x)}{\pi} \quad \left[\mathrm{sr}^{-1}\right] \tag{5.18}$$

where:

ρ_d is the diffuse reflectance.

The BRDF for physical materials are seldom perfectly diffuse, and as indicated by the definition not trivial to measure or implement, thus approximative shading models are typically used in computer graphics.

Phong

[Dutre et al., 2002, Phong, 1975]

The Phong shading model is the one of the most widely used shading models. The BRDF of the Phong model uses the direction to the light source **l**, the direction to the viewer **v**, the normal **n** at the surface point and the reflection vector **r**, to determine the shading at point *x*, see Figure 5.5.

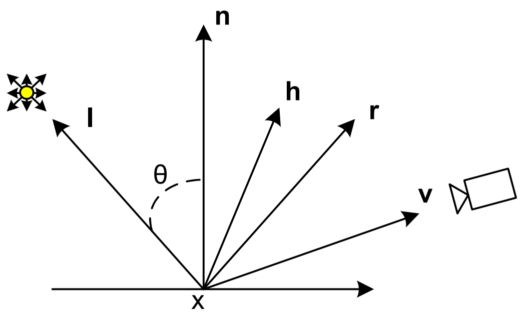


Figure 5.5: The vectors used by the Phong and modified Blinn-Phong shading model.

The reflection vector used in the Phong model is calculated as

$$\boldsymbol{r} = 2\mathbf{n}(\mathbf{n} \cdot \mathbf{l}) - \mathbf{l} \tag{5.19}$$

The BRDF of the Phong model is

$$f_r(x, \mathbf{l}, \mathbf{v}) = k_s \frac{(\mathbf{r} \cdot \mathbf{v})^n}{\mathbf{n} \cdot \mathbf{l}} + k_d \tag{5.20}$$

where:

k_s is the specular reflection coefficient.
n is the shininess coefficient.
k_d is the diffuse reflection coefficient.

The Phong model is not energy conserving and not physical correct for most real materials, a better solution is the modified Blinn-Phong model.

Modified Blinn-Phong

The modified Blinn-Phong shading model is faster to compute, and have by empirical studies been shown to provide a closer resemblance with the physical BRDF properties than Phong shading, [Addy Ngan and Matusik, 2004]. It uses the half-way vector **h** instead of the reflection vector **r** to obtain both the speed increase and higher correctness

$$\mathbf{h} = \frac{\mathbf{l}+\mathbf{v}}{|\mathbf{l}+\mathbf{v}|} \tag{5.21}$$

The half-way vector is only faster to compute for points at an infinite distance from the light source or directional lights. Because in both cases the light vector **l** is considered constant for all vertices, and the eye point in e.g. OpenGL eye-space is always located in (0,0,0), thus the half-way vector is identical for all vertices and only has to be computed once.

The BRDF for the modified Blinn-Phong model is

$$f_r(x,\mathbf{l},\mathbf{v}) = k_s\,(\mathbf{n}\cdot\mathbf{h})^n + k_d \tag{5.22}$$

Since the modified Blinn-Phong model as well as the Phong model are phenomenological models neither of them are energy conserving. [Angel, 2009, Dutre et al., 2002]

5.3 Lighting Calculations

This section introduces how lightning calculations can be made in a ray-tracer and by custom OpenGL shaders. The purpose of describing this is that it both will be used later in the developed system, and it serves as a summary on how to use the concepts and terms described in the preceding sections.

Light Source

The review of lighting calculations is based on a sphere light source, because this is a more physical tangible light source than a point light, its properties are listed in Table 5.1. A point light is in essence a sphere light with an infinitesimal radius.

Table 5.1: Properties for a sphere light.

Property	Symbol	Description
Power	Φ_s	The power of the light source. [W]
Radius	r	The radius of the diffuser. [m]

In the theory the power is wavelength dependent, but it is considered as one constant power with three intensities for the red, green, and blue color.

Material

The properties of the material are listed in Table 5.2.

Table 5.2: Properties for a material.

Property	Symbol	Description
Shininess coefficient	n	Shininess coefficient used by the shading model.
Specular reflection coefficient	k_s	Specular coefficient used by the shading model.
Diffuse reflection coefficient	k_d	Diffuse coefficient used by the shading model.
Refraction coefficient	k_r	Refraction coefficient used to determine transparency of the material.

Similar to the light source the coefficients are implemented as constants for all wavelength and three intensities for each coefficient.

Calculations

The calculation of the outgoing radiance L_o from a point x in direction $\boldsymbol{\omega}_o$ is calculated by Equation (5.14), where the modified Blinn-Phong shading model is used as BRDF function. If the scene contains more than one light source then L_o is given by

$$L_o(x,\boldsymbol{\omega}_o) = \sum_{j=1}^{N} L_{o,j}(x,\boldsymbol{\omega}_o) \tag{5.23}$$

where:

N is the number of lights in the scene.

$L_{o,j}$ is the outgoing radiance from light source j.

The irradiance, from a single light source, at a distance d from point x, is given by the radiance emitted from the light source $L_{\text{source}}(x)$ integrated over a solid angle corresponding to the area of the light source projected onto the unit hemisphere, see Figure 5.6. The angle β subtended by the projected area A, is calculated as

$$\beta = \arctan\left(\frac{r}{d}\right) \tag{5.24}$$

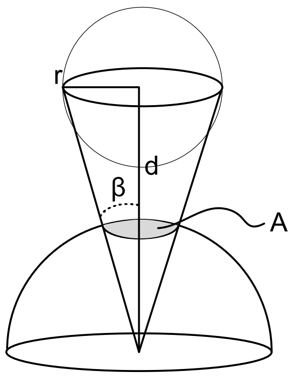


Figure 5.6: Projection onto the unit hemisphere of the area of a spherical light source as seen from a distant point *x*. From any point *x* the spherical light source appears as a circular disk.

With this knowledge the reflected radiance from *x* and a light source *j* is calculated as

$$
\begin{aligned}
L_{o,j}(x,\boldsymbol{\omega}_o) &= \int_{\Omega_{\text{source}}} f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o) L_{\text{source},j}(x,\boldsymbol{\omega}_i)\cos\theta d\boldsymbol{\omega} && (5.25)\\
&\approx L_{\text{source},j}(x,\boldsymbol{\omega}_i)\cos\theta f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)\int_{\Omega_{\text{source}}} d\boldsymbol{\omega} && (5.26)\\
&\approx L_{\text{source},j}(x,\boldsymbol{\omega}_i)\cos\theta f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)\int_0^{2\pi}\int_0^{\beta}\sin\theta d\theta d\varphi && (5.27)\\
&\approx L_{\text{source},j}(x,\boldsymbol{\omega}_i)\cos\theta f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)\cdot 2\pi(1-\cos(\beta)) && (5.28)
\end{aligned}
$$

When the expression for the outgoing radiance from the light source, see Equation (5.10)-(5.11), and the BRDF is inserted the final expression for the outgoing radiance at point *x* when split into a diffuse and specular term becomes

$$
\begin{aligned}
L_{o,j,\text{diffuse}}(x,\boldsymbol{\omega}_o) &= \frac{\Phi_s}{4\pi^2 r^2}\cdot 2\pi(1-\cos(\beta))\cos\theta k_d && (5.29)\\
&= \frac{\Phi_s}{2\pi r^2}\cdot(1-\cos(\beta))\cos\theta k_d && (5.30)
\end{aligned}
$$

$$
L_{o,j,\text{specular}}(x,\boldsymbol{\omega}_o) = \frac{\Phi_s}{2\pi r^2}\cdot(1-\cos(\beta))\cos\theta\cdot(\mathbf{n}\cdot\mathbf{h})^n k_s \qquad (5.31)
$$

Equation (5.25) can, when it is assumed that the radius of the sphere light source is much smaller than the distance to the shading point, be simplified to

$$
L_o(x,\boldsymbol{\omega}_o) \approx f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)\cdot E_{\text{surface}} \quad | \quad r \ll d \qquad (5.32)
$$

E_{surface} is calculated as

$$E_{\text{surface}} = \frac{\Phi_s}{4\pi d^2}\cos\theta \tag{5.33}$$

where:
 d is the distance from x to the light source.

The simplified version of the radiance calculation is in the implementation the preferred method for calculating the lighting because it minimizes the number of shading computations. The sphere lights are thus approximated by point lights.

The $\cos\theta$ term can be calculated by the normalized normal vector and the vector to the light source, see Figure 5.5, as

$$\cos\theta = \mathbf{n}\cdot\mathbf{l} \tag{5.34}$$

Example

The following is a short made up example of how the radiance for an arbitrary point in a scene could be calculated.

Table 5.3: Values used in the example.

Variable	Value
$\mathbf{n}$	$[0.0, 0.0, 1.0]$
$lightPos$	$[1.0, 5.0, 0.0]$
x	$[-4.0, 0.0, -13.0]$
d	$\lvert lightPos - x\rvert = 14.8$
$\mathbf{l}$	$\lVert lightPos - x\rVert = [0.34, 0.34, 0.89]$
Φ_s	500 W
ρ_d	1

First the $E_{\text{surface}}(x)$ is found at the intersection point x as

$$E_{\text{surface}}(x) = \frac{\Phi_s}{4\pi d^2}\mathbf{n}\cdot\mathbf{l} \tag{5.35}$$

$$E_{\text{surface}}(x) = \frac{500\,\text{W}}{4\pi 14.8^2\,\text{m}^2}\cdot 0.88 = 0.16\ \frac{\text{W}}{\text{m}^2} \tag{5.36}$$

Then the diffuse contribution at point x is calculated as

$$\begin{aligned} L_{o,\text{diffuse}}(x) &= f_{r,\text{diffuse}} \cdot E_{\text{surface}}(x) && (5.37)\\ &= \frac{\rho_d}{\pi} \cdot E_{\text{surface}}(x) && (5.38)\\ &= \frac{1}{\pi}\,\text{sr}^{-1} \cdot 0.16\ \frac{\text{W}}{\text{m}^2} && (5.39)\\ L_{o,\text{diffuse}}(x) &= 0.05\ \frac{\text{W}}{\text{m}^2\text{sr}} && (5.40) \end{aligned}$$

Next the specular contribution, with a shininess constant $n = 20$ and a $k_s = 0$, is calculated. That requires the half-way vector which is found to be $\mathbf{h} = [0.32, 0.18, 0.93]$.

The specular contributes, at the perfectly diffuse surface, becomes

$$L_{o,\text{specular}}(x) = k_s \cdot (\mathbf{n} \cdot \mathbf{h})^n \cdot E_{\text{surface}}(x) = 0\ \frac{\text{W}}{\text{m}^2\text{sr}} \tag{5.41}$$

The total radiance at point x is finally found as the sum of the diffuse and specular contribution $L_o(x) = L_{o,\text{diffuse}}(x) + L_{o,\text{specular}}(x) = 0.05\ \frac{\text{W}}{\text{m}^2\text{sr}}$.

5.4 Summary

This chapter introduced a set of useful definitions and methods used to do illumination calculations. Of particular importance is the definitions of radiance, irradiance and the light interaction models. The light interaction model described was the BRDF, which was combined with both Phong shading, and modified Blinn-Phong shading. The later is the shading model of choice for this project.

The chapter was concluded with an example of how to calculate the radiance at a point in space, this is essentially what a ray-tracer is doing.

The illumination theory described and exemplified in this chapter was mainly described in a local illumination framework. The next chapter, about photon mapping, will reuse a lot of this in a global illumination setting.

CHAPTER 6 PHOTON MAPPING

The problem statement stated that photon mapping is to be used together with the eye-tracker. This chapter introduces the details of photon mapping, because it is important foundation for developing a perceptually adaptive photon mapping algorithm.

6.1 Introduction

Photon mapping is a computer graphics technique to make global illumination. It was invented by the Dane Henrik Wass Jensen, and was first published in 1995 in the paper, ”Photon Maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects” [Jensen and Christensen, 1995]. Photon mapping can simulate different global illumination effects, and the advantages of photon mapping are that it is very versatile and fast [Jensen, 2001]. One global illumination effect is caustics, which is indirect light patterns that are refracted or reflected, e.g. the refraction of light through a bottle, or light patterns reflected onto walls near water surfaces. Another effect is color bleeding, which is indirect illumination between two surfaces where light from one surface reflects on the other surface. A third effect that can be simulated using photon mapping is subsurface scattering, which occurs when light penetrates translucent surfaces and exits the surface at different points.

Photon mapping is a multi-pass rendering method. The first pass is where the photons are traced through the scene to construct the photon map, and the second pass is the actual rendering.

In the first pass photons (packets of light energy) are emitted from the light source in different directions. When the photon hits a diffuse surface it is stored in a photon map. To make the photon mapping more efficient two different photon maps can be used a caustics-, and a global photon map. The photons stored in the caustics photon map are the photons that have been reflected or transmitted via a specular surface before hitting a diffuse surface. When the photon hits a diffuse surface it is stored and terminated, since diffuse surface cannot create caustics. To built the caustics photon map quickly the emission of photons must be focused toward the specular surfaces. For that a projection map can be used. Because caustics are a distinctive effect that is easily caught by the eye, the caustics photon map must be of high quality (many photons) in these areas. The global photon map represents the direct illumination, indirect illumination, and also caustics. The global photon map is build by tracing the photons in the scene and storing them when they hit a diffuse surface. Unlike the caustics photon map the diffuse surface may also reflect the photons.

In the rendering pass the scene is rendered using traditional ray-tracing together with the constructed photon maps. The rendering equation used, can be split into four parts, one for direct illumination, indirect illumination, specular reflection, and caustics. Normally the direct illumination and specular reflection are calculated from ray-tracing, whereas the caustics and indirect illumination is calculated from the photon maps.

In the rest of the chapter a more thorough description of the different elements within photon mapping will be given. The chapter is mainly based on Henrik Wann Jensen's book ”Realistic Image Synthesis Using Photon Mapping” [Jensen, 2001].

6.2 First Pass: Photon Tracing

The purpose of the first pass of photon mapping is to create a photon map by tracing photons through the scene. Henrik Wann Jensen describes it as:

> Building the photon map structure by tracing photons from the lights through the model.

6.2.1 Photon Emission

The first step in creating a photon map is emitting photons from the light source(s) into the scene. Emission of photons can be complex depending on the type of light source which is to be modeled. If the attention is limited to non-physical diffuse point lights then the emission of photons becomes fairly straightforward. The power of the diffuse point light source Φ_{photon}, has to be divided between all emitted photons. These photons are emitted uniformly over the entire unit sphere encasing the point light.

$$\Phi_{photon} = \frac{\Phi_{light}}{n} \tag{6.1}$$

where:

Φ_{photon} is the power of the photon.
Φ_{light} is the power of the light source.
n is the number of photons emitted.

With the power of the photons determined, the directions of the photons have to be determined too, this can be done by Algorithm 6.1.

Algorithm 6.1 Algorithm for uniformly emitting photons from a diffuse point light. [Jensen, 2001]

```
INPUT:
    Scene to be photon traced
OUTPUT:
    Photon map

void diffuse_point_light_emitter()

1. $n \leftarrow 0$                                    //number of emitted photons
2. while ($n <$ maximum number of photons)
    • do
        ○ random sample $x, y, z$
        ○ $x, y, z \in [-1, 1]$
    • while ($x^2 + y^2 + z^2 > 1$)
    • $d \leftarrow \langle x, y, z \rangle$
    • $p \leftarrow$ light source position
    • $dir \leftarrow d - p$
    • trace_photon($dir$)
    • $n \leftarrow n + 1$
   end while
3. scale emitted photons with $\frac{1}{n}$
```

In case of more than one light source, the number of photons per light source has to be proportional to power of the light source, such that bright light sources emit more photons. It is in general not necessary to increase the total number of photons.

On problem with this simple emission model is that many photons might not hit objects of interest, or objects at all, to prevent this two tools are useful, namely projection maps and visual importance maps.

Projection Map

A projection map is used to control the emission of photons in directions of objects, this has several benefits. Firstly it can reduce the computational complexity because in general less photons have to be emitted to get enough photons that hit objects. Secondly it can improve the quality of the photon map since all photons are usable.

The projection map is typically an image of the scene as seen from the light source. For a point light, it is a spherical projection of the scene, where regions with objects can be distinguished from regions without objects. Often the projection map is a conservative estimate of where objects are located, created from object bounding boxes for efficiency reasons.

When emitting photons the projection map can be used to emit photons that will only hit objects. This can be done by iterating over all regions with objects and emit photons in various directions inside each region. An alternative is to use a completely random approach as Algorithm 6.1, where an additional condition is added such that *d* has to hit an region on the projection map with objects. The first method works well if it is desirable that all objects are hit by an equal amount of photons, but at the cost that the total number of photons is dependent on the number of objects. The later method on the contrarily ensures a consistent total number of photons at the expense of ensuring that all objects are hit by an equal amount of photons.

No matter which of the methods that is used the power of the photons needs to be rescaled compared to Equation (6.1), this is done by

$$P_{photon} = \frac{P_{light}}{n} \cdot \frac{\text{regions with objects}}{\text{total number of regions}} \tag{6.2}$$

If information about the surface reflection is captured by the projection map this information can be used to emit more photons in direction of specular surfaces, i.e. surfaces that generate caustics.

Visual Importance

A visual importance map can be used to determine regions where photons would constitute more to the visual appearance of the scene, and thus be more important. Using visual importance maps make most sense if the scene is large with only a small portion of it being within the field of view of the viewer. In such case it is waste of computational power to emit photons into regions that are outside the field of view.

A three-pass visual importance based photon mapping algorithm is as followed.

1. Photons, so-called importons, are emitted from the viewpoint, and traced into the scene at a maximum depth of one diffuse reflection, before they are stored in the importance map.

2. A photon map is created similar to the case when no visual importance map is present, but whenever a photon is to be stored, it has to survive a visual importance check. This check consist of sampling the nearby region in the importance map to determine if a sufficient number of importons exists before the photon can be deemed visual important.

3. The image can now be rendered by e.g. ray-tracing from the photon map which embodies the visual importance.

6.2.2 Photon Scattering

The emitted photons are traced through the scene, similar to ray-tracing, with the difference that photon tracing propagate flux whereas ray-tracing gather radiance. When a photon hits a surface it can either be reflected, transmitted, or absorbed, which is decided probabilistic based on the material of the surface. The method to decide the interaction is called Russian roulette.

Russian Roulette

Russian roulette is a Monte Carlo technique introduced to speed up computations in particle systems, and later used in context to computer graphics. It is a stochastic technique used to reduce the number of photon calculations.

When a photon hits a surface it interacts with the material based on the properties of that material. The three pure types of interactions are

1. Reflection (the energy of the photon is reflected in one or multiple directions based on the specular and diffuse properties of the material)
2. Transmission (the energy of the photon is transmitted through the material, based on Snell's Law of refraction)
3. Absorption (the entire energy of the photon is absorbed in the material, nothing is reflected or refracted)

None of the above interaction types exist in their pure form in typical materials, since most materials at least reflect and absorb some energy. This creates two problems in photon mapping, which can be illustrated by tracing a single photon. When the photon hits a material, for the sake of argument a completely diffuse material without absorption, it will be reflected in several different directions thus one photon is split into several new photons with less energy. From a computational stand point this is impractical because the number of photons increases with every surface interaction, resulting in more intersections which need to be calculated. Secondly it is desirable to have a photon map with photons of equal energy when estimating the radiance in order to achieve the best estimate [Jensen and Christensen, 2007].

Russian roulette is a stochastic optimal solution to the problem outlined above. For any photon surface interaction a uniformly distributed random variable $w \in [0,1]$ is sampled. Depending on the value of w one of the following actions can be taken

$$\begin{aligned} w &\in [0,r] \rightarrow \text{reflection} \\ w &\in]r, r+t] \rightarrow \text{transmission} \\ w &\in]r+t, 1] \rightarrow \text{absorption} \end{aligned} \tag{6.3}$$

where:
r is the reflection coefficient of the material.
t is the transmission coefficient of the material.

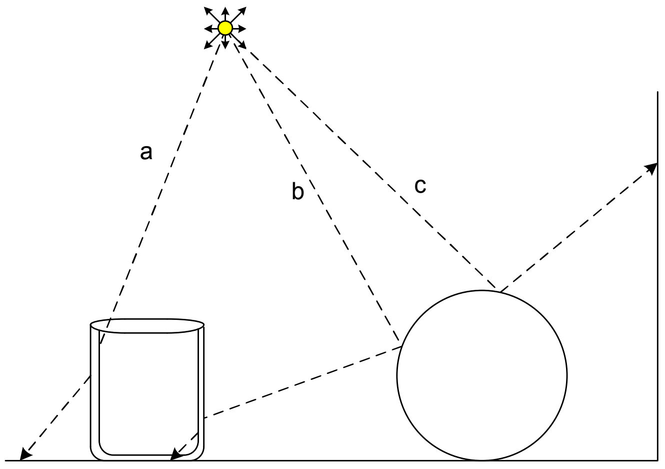


Figure 6.1: A scene, with a diffuse point light source, a transparent glass to the left and a diffuse sphere to the right. a) A photon is emitted and hits a transparent surface thus it is specular refracted before it hits the wall which absorbs it. b) The photon first hits the diffuse sphere, from which it is reflected and hits the glass resulting in a specular refraction. c) Diffuse reflection on the sphere.

For the above mentioned to make sense $r+t \leq 1$ must hold, if the sum of the reflection and transmission coefficients is less than one the remaining is accounted as absorption. The reflection coefficient r is split into both a specular and diffuse reflection coefficient based on the material properties.

When the outcome of Russian roulette is diffuse reflection, the photon is stored in the photon map and diffusely reflected. If the outcome is specular reflection the photon is not stored but reflected, next time the photon hits a diffuse surface it is in addition to being stored in the general photon map also stored in the caustics map. In the event of absorption, the photon is stored and the life of the photon is terminated. See Figure 6.1 for an illustration of the different interactions.

6.2.3 Photon Storing

A photon is stored only when it hits diffuse surfaces, because specular surfaces do not give any useful information – for that ray-tracing is used. All photons that interact with diffuse surfaces in the scene are stored in a photon map data structure. The photons in the photon map do not associate the photons with the geometry in the scene, since the photon map and the geometry in the scene are decoupled from each other.

Each photon is stored in a structure like this

```
struct photon{
    float x,y,z;          //position
    float p[3];           //photon power
    float indir[3];       //incident direction
    short plane;              //splitting plane for kd-tree
}
```

The structure/photon contains the position in space where the photon hit the surface, and the power of the photons for each RGB-component, and the incident direction of the photons. The plane defines the splitting plane used in the kd-tree, see next section.

The photon map is used for the calculation of the illumination, primary the indirect illumination and caustics, in the scene. For the calculation of the illumination at a given point the *n* nearest photons around this point are used. Therefore it is practical to use a data structure that makes it easy and fast to locate the nearest neighbors of a three dimensional point space. Furthermore the data structure should be compact because often many millions photons must be stored. The best solution is a kd-tree.

Balanced Kd-Tree

The photons are organized into a balanced kd-tree. A kd-tree is a k-dimensional binary tree and is used for organizing points in a k-dimensional space. Each point has a splitting hyperplane through one dimension that divides the space into two sub spaces along this dimension. In each of the two sub spaces another point in each space has a new splitting hyperplane through one dimension, that again divides the sub space into two new sub spaces. This is repeated until all points have a splitting hyperplane. The very first point is the root node. A simple example with two dimensional points can be seen in Figure 6.2.

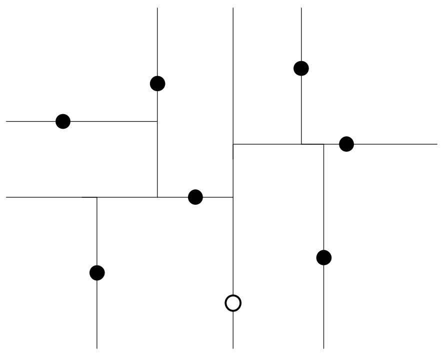


Figure 6.2: Example of a 2d-tree with eight points. The empty point is the root node.

The most efficient kd-tree to use for photon storing where many lookups are required is a balanced kd-tree, in which the child nodes have about the same distance to the parent node. To make a balanced kd-tree, the point that is chosen for the splitting through a certain dimension is the point with the median value along this dimension.

The recursive algorithm for constructing a balanced kd-tree is listed in Algorithm 6.2.

The complexity of balancing a kd-tree is $O(n_p \cdot \log n_p)$, where n_p is the number of photons in the photon map. Even for several millions photon the balancing takes less than one second. And the location of

Algorithm 6.2 Algorithm for constructing a balanced kd-tree.

```
INPUT:
    A list of points $P$
    A dimension $dim$
OUTPUT:
    The root node of the kd-tree storing $P$

kdtree balanced_kdtree( $P$, $dim$ )

1. if ($P$ is empty) return 0
2. else
    • Find median $m$ from $P$ in dimension $dim$
    • $s1 \leftarrow$ all points from $P$ below $m$ in dimension $dim$
    • $s2 \leftarrow$ all points from $P$ above $m$ in dimension $dim$
    • kdtree node $\leftarrow m$
    • $node.leftChild \leftarrow$ balanced_kdtree( $s1$, next dimension )
    • $node.rightChild \leftarrow$ balanced_kdtree( $s2$, next dimension )
    • return node
   end if
```

one photon in a balanced kd-tree is found in $O(\log n_p)$. Getting the irradiance estimate from the photon map, is done in $O(nNN \cdot \log n_p)$ time, where *nNN* is the number of neighboring photons to include in the estimate [Jensen, 1996a].

Locating the Nearest Photons

During the rendering pass the *nNN* photons nearest to the rendering point x are needed. An efficient lookup of these neighbors in the photon map is important to obtain a good performance. The algorithm used is the nearest neighbor algorithm.

The nearest neighbor search algorithm begins at the root node, and then moves down the tree recursively. It goes right or left in the tree depending on whether the point x is greater or less than the current node in its splitting dimension. When it reaches a leaf node it becomes the current nearest neighbor. Then the algorithm returns to the previous nodes which will be added to the list of nearest neighbors. If the side of the sub tree furthest from the point x is within a defined max search distance, this side is also investigated, otherwise it is skipped which reduces the search space. When *nNN* neighbors are found the distance from the furthest neighbor to the point x is used as a new max distance. Then the rest of the nodes that are investigated will only be added to one of the n nearest neighbors if its distance to x is less than the max distance.

The distance that is used is the squared distance to avoid the square root operation that is needed to calculate the Euclidean distance. The algorithm for locating the *nNN* nearest neighbors is listed in Algorithm 6.3.

Algorithm 6.3 Algorithm for locating the *nNN* nearest neighbors in a balanced kd-tree.

```
GIVEN:
  A point $x$ from where the $nNN$ nearest photons are located
  A squared maximum search distance $d_{max}$
INPUT:
  A photon $p$ (node in the kd-tree)
OUTPUT:
  A list $L$ with the $nNN$ nearest photons

list locate_photons( p )

1. if the node $p$ has children in the kd-tree
    • $\delta \leftarrow$ signed distance from $x$ to $p$ in the splitting dimension of $p$
    • if ($\delta < 0$)
        ○ locate_photons(left child of $p$)
        ○ if ($\delta^2 < d_{max}$) locate_photons(right child of $p$)
    • else
        ○ locate_photons(right child of $p$)
        ○ if ($\delta^2 < d_{max}$) locate_photons(left child of $p$)
    • end if
2. $dist \leftarrow$ squared distance from $p$ to $x$
3. if ($dist < d_{max}$)
    • Insert $p$ into the list $L$
    • if ($n$ photons are found) $d_{max}$ = squared distance from $x$ to the furthest photon in $L$
  end if
```

6.3 Second Pass: Rendering

The second pass deals with the rendering of the scene based on the information provided by the photon map created during the first pass. The rendering pass is similar to a traditional ray-tracer but differs because it uses the photon map for caustics and indirect illumination. A description of traditional ray-tracing can be found in Appendix B on page 157. Henrik Wann Jensen says the following about the second pass:

> Rendering the model using the information in the photon map to make the rendering more efficient.

The rendering pass is about solving the rendering equation. It is divided into parts where ray-tracing is used and parts where the photon map is used. The rendering equation and other fundamental global illumination theory (radiance, BRDF, etc.) is described in Chapter 5 on page 45. Only rendering theory that is specific for photon mapping will be described in this chapter, first by specifing how the rendering equation is split.

6.3.1 Splitting of the Rendering Equation

The outgoing radiance L_o, at a surface point x, is calculated by the rendering equation

$$L_o(x,\boldsymbol{\omega}_o) = L_e(x,\boldsymbol{\omega}_o) + L_r(x,\boldsymbol{\omega}_o) \tag{6.4}$$

where:

L_e is the emitted radiance.

L_e is the reflected radiance.

For the purpose of photon mapping the rendering equation is not tangible. The preferred method is to separate it into terms which can be calculated from the photon map and terms that must be calculated by traditional ray-tracing.

First the BRDF is considered. Typically the BRDF is composed of two components, namely a diffuse and a specular BRDF

$$f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o) = f_{r,D}(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o) + f_{r,S}(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o) \tag{6.5}$$

The incident radiance can also be split into several terms

$$L_i(x,\boldsymbol{\omega}_i) = L_{i,l}(x,\boldsymbol{\omega}_i) + L_{i,c}(x,\boldsymbol{\omega}_i) + L_{i,d}(x,\boldsymbol{\omega}_i) \tag{6.6}$$

where:

$L_{i,l}(x,\boldsymbol{\omega}_i)$ is the radiance directly from the light source(s).

$L_{i,c}(x,\boldsymbol{\omega}_i)$ is the caustics radiance arriving at the point after at least one specular reflection or transmission.

$L_{i,d}(x,\boldsymbol{\omega}_i)$ is the indirect radiance arriving at the point after at least one diffuse reflection.

Using these separated expressions for the BRDF and radiance makes it possible to expand the reflected radiance into four integrals each representing a particular part of the rendering.

$$
\begin{aligned}
L_r(x,\boldsymbol{\omega}_o) = &\int_\Omega f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)L_{i,l}(x,\boldsymbol{\omega}_i)\cos\theta d\boldsymbol{\omega}_i + &\text{(6.7a)}\\
&\int_\Omega f_{r,S}(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)\left(L_{i,c}(x,\boldsymbol{\omega}_i)+L_{i,d}(x,\boldsymbol{\omega}_i)\right)\cos\theta d\boldsymbol{\omega}_i + &\text{(6.7b)}\\
&\int_\Omega f_{r,D}(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)L_{i,c}(x,\boldsymbol{\omega}_i)\cos\theta d\boldsymbol{\omega}_i + &\text{(6.7c)}\\
&\int_\Omega f_{r,D}(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)L_{i,d}(x,\boldsymbol{\omega}_i)\cos\theta d\boldsymbol{\omega}_i &\text{(6.7d)}
\end{aligned}
$$

The four parts are described in more detail after an estimation of the reflected radiance using the photon map has been introduced.

6.3.2 Radiance Estimate

From the preceding introduction of the reflected radiance equation it is not easy to identify how to implement it and how to use it in relation with photon maps. It will therefore now be shown how to estimate the reflected radiance at a surface point using the photon map.

The photon map as described earlier contains a huge number of photons, and each photon explains the amount of incident flux at a diffuse surface point. To compute the reflected radiance from this knowledge the general equation for reflected radiance has to be used

$$L_r(x,\boldsymbol{\omega}_o) = \int_\Omega f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)dE(x,\boldsymbol{\omega}_i) = \int_\Omega f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)L_i(x,\boldsymbol{\omega}_i)\cos\theta d\boldsymbol{\omega} \tag{6.8}$$

The BRDF f_r is assumed to be known for the material thus the only unknown is the incident radiance L_i, which can be estimated from the photon map.

Estimation of the incident radiance from the photon map requires that the flux quantities of the photon map are converted to radiance. For this purpose the definition of radiance is used, see Equation (5.7) on page 47, which for the less general case with the incident radiance and incident flux becomes

$$L_i(x,\boldsymbol{\omega}_i) = \frac{d^2\Phi_i(x,\boldsymbol{\omega}_i)}{\cos\theta_i d\boldsymbol{\omega}_i dA_i} \tag{6.9}$$

Equation (6.9) is then inserted into the expression for reflected radiance, Equation (6.8), as

$$
\begin{aligned}
L_r(x,\boldsymbol{\omega}_o) &= \int_\Omega f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)\frac{d^2\Phi_i(x,\boldsymbol{\omega}_i)}{\cos\theta_i d\boldsymbol{\omega}_i dA_i}(\boldsymbol{\omega}_i\cdot\mathbf{n})d\boldsymbol{\omega}_i\\
&= \int_\Omega f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)\frac{d^2\Phi_i(x,\boldsymbol{\omega}_i)}{dA_i}
\end{aligned}
\tag{6.10}
$$

Now it is possible to estimate the incoming flux Φ_i from the photon map. This is achieved by taking

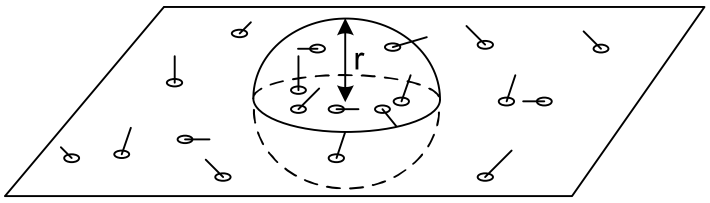


Figure 6.3: The sphere of radius r from which the nNN nearest photons are selected from.

the nNN nearest photons and assume that they all intersect at location x. If each of the photons p have a power of $\Delta\Phi_p(\boldsymbol{\omega}_p)$ then the integral of Equation (6.10) can be regarded as a summation

$$\tilde{L}_r(x,\boldsymbol{\omega}_o) = \sum_{p=1}^{n} f_r(x,\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)\frac{\Delta\Phi_p(x,\boldsymbol{\omega}_p)}{\Delta A} \tag{6.11}$$

where:

$\tilde{L}_r(x,\boldsymbol{\omega}_o)$ is the estimated reflected radiance.

$\boldsymbol{\omega}_p$ is the incident direction of the photon.

The area ΔA has to represent the metric used for finding the nNN nearest photons. Thus in the case where the nNN nearest photons within a sphere of radius r is found, the area should resemble this sphere projected onto the surface around the intersection point x. For computational efficiency it is assumed that this surface is flat, such that the projection is merely the area of a circle with radius r, $\Delta A = \pi r^2$. See Figure 6.3.

The final expression for the estimated reflected radiance then becomes

$$\tilde{L}_r(x,\boldsymbol{\omega}_o) = \frac{1}{\pi r^2}\sum_{p=1}^{n} f_r(x,\boldsymbol{\omega}_p,\boldsymbol{\omega}_o)\Delta\Phi_p(x,\boldsymbol{\omega}_p) \tag{6.12}$$

6.3.3 Direct Illumination

Direct illumination is the most important part of the rendering in typical scenes. It does in its basic form not rely on the photon map, but is solely calculated by the ray-tracer. At the point of intersection between the ray shot from the eye and the closest object shadows rays are cast towards each light source. All light sources which are directly reachable with the shadow rays contribute to the incoming direct radiance $L_{i,l}$ at the intersection point. The direct illumination at the point can now be calculated by the first term (6.7a) of Equation (6.7), where the BRDF of the material is known.

One thing to note is that casting the shadow rays is only trivial for point light sources. If e.g. area light sources are present, then these, at least in theory, have to be sampled by infinitely many shadow rays which is computationally costly. Luckily photon mapping can be used to speed-up this process by

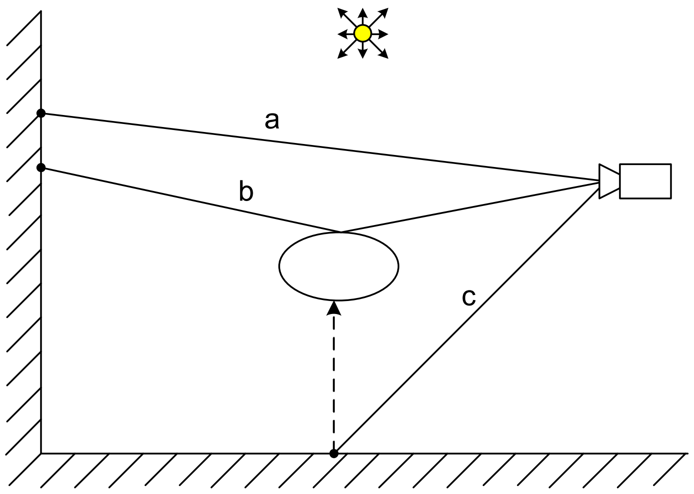


Figure 6.4: a) Direct illumination, the point of interest is directly in line of sight of the light source. b) Specular reflection, the sphere in the middle of the scene is specular so the ray is bouncing of before it hits the diffuse wall. c) Indirect illumination, the point is not lit by the light source but only by rays bouncing of the walls, calculated from the photon map.

so-called shadow photons. Shadow photons are photons with negative energy, that are traced from light sources through objects and only stored at diffuse surfaces. Combining the shadow photons with the normal photons around a point it is possible to estimate whether a region is in shadow or not.

6.3.4 Specular Reflection

Specular reflection is the distinct effect seen when light hits specular or glossy materials. It is calculated with ray-tracing by evaluating the second term (6.7b) of Equation (6.7), and does not use the photon map. The specular reflection is prominent in close proximity of the mirror direction of the incoming ray, and thus if photon mapping had to be useful it would require a densely packed photon map. Something which is infeasible both from a computational and a memory requirement standpoint.

6.3.5 Caustics

Rendering of caustics is one thing photon mapping was designed for, so the photon map should obviously be used for this purpose. Caustics can be rendered in two ways: accurately or as an estimate. If accurately caustics are desired, it is necessary to create a caustics map during the photon tracing stage. If an estimate of the caustics effect is adequate, then only the global photon map is required.

In both cases the caustics are calculated by the third term (6.7c) of Equation (6.7), using the radiance estimate from the appropriate photon map.

6.3.6 Indirect Illumination

The last term (6.7d) of the expanded rendering equation, Equation (6.7), constitute the indirect illumination. Indirect illumination includes effects like, color bleeding, and soft-shadows and is produced by light that have been diffusely reflect at least once.

It is also possible to create an accurate or an estimate of the indirect illumination. Accurately indirect illumination rendering is achievable by ray-tracing but is computational heavy since it requires tracing a large number of rays. The estimate is significantly faster because it can be calculated with a radiance estimate made from the photon map.

6.4 Summary

This summary recap the necessary knowledge needed in order to implement photon mapping in a traditional ray-tracer.

First and foremost, traditional photon mapping is a two pass procedure involving first photon tracing and afterward rendering. For static scenes the photon tracing only has to be carried out once before it can be used to render the scene from multiple views.

First Pass: Photon Tracing

The first task of the photon tracing pass is to find the light sources of the scene and their properties. For each light source a number of photons depending of the power of the light source must be emitted. The emission directions of the photons are determined by the type of light source.

With a direction of a photon and a starting point (the light source) a photon ray can be traced and at the surface intersection closest to the light source Russian roulette will be executed.

Russian roulette will in accordance with the material properties decide what to do with the photon. Thus a photon can be absorbed, transmitted or reflected. When the photon is absorbed it is stored in the photon map if the material is diffuse. When the photon is transmitted, it is not stored in the photon map right away, but it is marked for later storage in the caustics map next time it hits a diffuse surface. When the photon is reflected it can either be specular reflected or diffuse reflected depending on the material. The photon is stored in the photon map when the material is diffuse, and diffusely reflected. For specular materials the photon is reflected in the mirror direction, and marked for later storage in the caustics map.

It is possible that a photon can be traced for infinity, thus it is advisable to introduce a maximum trace depth to be tolerable before terminating the photon, and advancing to the next photon. The first pass is complete when all photons for all light sources have been emitted and traced or the photon map is full (the size of the photon map is determined on creation, and is ultimately limited by available memory).

Before advancing to the rendering pass, the data structure used for the photon map should be optimized for faster photon look up. When a kd-tree is used as photon map this optimization involves a balancing of the kd-tree.

Second Pass: Rendering

With the photon map created and stored in memory the actual rendering of the scene can be made.

As with traditional ray-tracing, rays are traced from the view point through the view plane and into the scene and at the intersection between a ray and the closest scene object the radiance is computed. It is this computation that benefits from the photon map.

Intersections with specular objects do not use the photon map, here the ray is bounced in the mirror direction and traced further until it hits a diffuse object. When a ray hits a diffuse object a contribution from the photon map is added to the radiance. This contribution is calculated by finding the radiance estimate from the nNN nearest photons within a distance of r from the point of intersection.

Thus in essence everything needed to add good looking global illumination effects to a ray-tracer is a photon map of the scene, which is consulted for a radiance estimate every time the radiance of a diffuse object is to be calculated.

CHAPTER 7 DESIGN

This chapter outlines four different design ideas within the field of perceptually adaptive computer graphics. The design ideas use the gaze position acquired by an eye tracker to improve the perceived visual quality in different ways.

Based on complexity, suitability, and a feasibility study of the four algorithms a design solution is chosen, which is later used in the implementation.

7.1 Introduction

This introduction serve as an insight into the overall design decisions made prior to the selection of the design ideas that are described in details in this chapter.

Figure 7.1 shows an overview of the possibilities with perceptually adaptive rendering. The four dif-

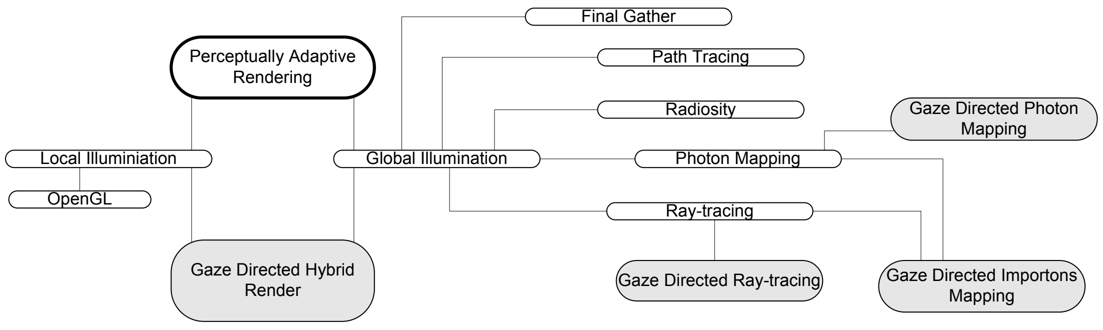

Perceptually Adaptive Rendering
Final Gather
Path Tracing
Radiosity
Gaze Directed Photon Mapping
Local Illuminiation
Global Illumination
Photon Mapping
OpenGL
Ray-tracing
Gaze Directed Hybrid Render
Gaze Directed Ray-tracing
Gaze Directed Importons Mapping

Figure 7.1: Overview of the design process.

ferent design approaches are shown in the dark boxes. The first decision was to concentrate on global illumination methods, in contrast to local illumination, since the possibilities for making global illumination visually realistic is better than local illumination.

This decision generated an interesting idea, of combining a local illumination rendering with global illumination around the gaze point, in what would be a so-called hybrid render.

The global illumination ideas for the hybrid rendering algorithm involve ray-tracing, photon mapping, and photon mapping with importon mapping.

In the following the four design ideas will be described in details.

7.2 Gaze Directed Importon Mapping

This idea extends the traditional two-pass photon mapping with an additional pass. The extra pass is carried out in the beginning of a frame, where so-called importons are emitted from the eye point through the view plane and into the scene geometry, Listing 7.1 shows the structure of an importon. As it is the case for normal photons these importons are also stored once they hit diffuse surfaces, but instead of saving them in a photon map they are stored in an importance map. As normal photons have a power, so do importons, but instead of representing the power of the light source, their power represents the importance of the importon, such that an importon close to the gaze point have a higher importance than those far away.

Listing 7.1: Structure of an importon.

```
struct importon{
    float x,y,z;            //position
    float imp;              //importance
    float indir[3]          //incident direction
    short plane;            //splitting plane for kd-tree
}
```

The basis of the idea is similar to photon mapping with visual importance maps as explained in Section 6.2.1 on page 60. The unique contribution of this idea is that by using an eye tracker it becomes possible to direct the importons to not only the field of view, but to the region in close proximity of the point of regard, the gaze region.

In the second pass photons are emitted as normal, but they are only stored in the photon map in areas where a sufficiently high amount of importons are available. Thus more photons are stored in areas of high visual importance, and less in others.

The third pass is the traditional rendering pass, using ray-tracing.

The advantage of this method is that it is independent of the previous frame. The drawback is that it requires an extra pass which might impose a frame rate penalty.

In summary the idea is that by introducing an additional pass, a photon map with fewer photons is sufficient in order to obtain the same visual quality in the gaze area as if photons were stored evenly in the entire scene. This will be elaborated further in the following algorithmic description, together with some potential pitfalls.

7.2.1 Algorithmic Description

This algorithmic description will focus on the importon emission step and photon emission step, see Figure 7.2, since the ray-tracing step is identical to that of traditional photon mapping.

Importon Emission

Emission of importons is a straightforward process with a few things to consider. When the position of the camera, or viewer of the scene, and the view plane is known, then it is possible to construct the direction vectors used to determine the emission direction of the importons. Additionally it is desired to sample the scene with as few importons as necessary, since both the complexity of lookup and balancing of the importon map is dependent on the number of importons.

The method used to implement gaze directed importon mapping is to adjust the importance of the importons in accordance with their distance, to the gaze point. Alternatively the sample rate of the view plane could be dynamic to allow for more importons close to the gaze point.

Unlike traditional photon emission, there is no need for importons to bounce past their first surface interaction. Thus if an importon does not hit a diffuse material it is simply not stored. Due to this it is possible to implement the importon emission step with either ray-tracing or rasterization. The information needed is the incident direction and the intersection with the geometry, something both techniques can provide.

Photon Emission

The photon emission step is in fact similar to the traditional photon mapping. But in order to fully describe the potential and pitfalls of gaze directed importon mapping more details will follow.

The emission technique used by gaze directed importon mapping is the naive emission model where photons are emitted randomly onto a unit sphere. This may cause a lot of photons to be emitted in directions where no importons are stored, and thus it is possible that they are traced until the maximum trace depth without ever being stored. This is one major drawback of the algorithm, and something that will be the focus of attention in one of the other algorithms, Gaze Directed Photon Mapping, described in Section 7.3 on page 78.

The emission of photons must continue until a pre-defined number of photons have been stored, and not just emitted. Depending on the size of the region where the importons is stored the emission process

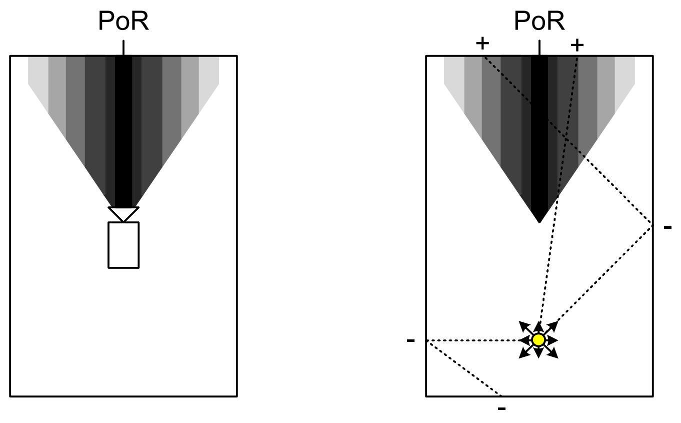


Figure 7.2: Left illustration of the importon emission step. Importons are emitted from the camera into the scene around the Point of Regard (PoR). The shading represents the importance, thus black is very important while white is not important. Right illustration of the photon emission step. Only a few of the photons are shown, by the dotted lines. At intersection points with a ”−” no photon is stored while at intersections with ”+” the photon is stored.

can take arbitrarily long, when using the naive emission model.

If the region with importons is not too small compared to the entire scene space, then the algorithm might prove useful. That can be clarified with an example. Consider a scene where importons occupy 25 % of the scene space, thus with the naive emission model 75 % of the photons will on average not be stored, and can be considered wasted. But if no importons map was used all the photons would have to be stored, in order to provide the same visual quality within the 25 % of the scene space that was the important region. This would result in a photon map with 4 times more photons, which would affect the storage requirements, the time complexity of balancing the photon map, and lookup in the photon map negatively.

The time required to balance and look up in the photon map is dependent on the number of stored photons. Therefore is the main selling point of the algorithm also its ability to reduce the total number of photons, and the algorithm can be improved by applying a more sophisticated emission model. A more thorough review of the complexity of the individual steps is given next.

7.2.2 Complexity Evaluation

The complexity will be evaluated for the following three steps:

1. Emission of importons from the view point into the view plane. Importons close to the gaze point have a higher importance.

2. Traditional photon tracing from the light source, but only store photons where significant importance is indicated by the importance map.

3. Ray-trace the scene to create the final rendering.

The complexity of importance mapping is divide into three functions, one for each step O_i where i indicate the step. The worst case scenario is used as baseline when deriving the complexity, thus balancing of the photon and importon map is carried out for every frame.

First a function describing the importance step is outlined

$$\begin{aligned} O_1(n_i, t_1, o) &= n_i \cdot t_1 \cdot o + n_i \log(n_i) \\ &= n_i \left(t_1 \cdot o + \log(n_i) \right) \end{aligned} \tag{7.1}$$

where:

n_i the number of importons in the image.
t_1 is the trace depth of the importance step $t_1 = 1$.
o is the number of objects in the scene.
$n_i \log(n_i)$ is the complexity of balancing the importance map.

Typically n_i would be determined from a number of pixels in the gaze area and a sample rate.

The second step of the importon mapping algorithm is similar to traditional photon mapping, with the addition of an importon look-up

$$\begin{aligned} O_2(n_p, t_2, o, n_i) &= n_p \cdot t_2 \cdot o \cdot \log(n_i) + n_p \log(n_p) \\ &= n_p \left(t_2 \cdot o \cdot \log(n_i) + \log(n_p) \right) \end{aligned} \tag{7.2}$$

where:

n_p is the number of photons $n_p \in [10^3; 10^7]$.
t_2 is the trace depth of the photon mapping step $t_2 \in [1; 3]$.
$\log(n_i)$ is the complexity of look-up in the importance map.
$n_p \log(n_p)$ is the complexity of balancing the photon map.

Finally the complexity of the ray-tracing step can be found from the following function

$$O_3(n, t_3, n_p) = n \cdot t_3 \cdot o \cdot nNN \log(n_p) \tag{7.3}$$

where:

n is the number of rays.
$nNN \log(n_p)$ is the complexity of looking up the irradiance estimate from the nNN nearest photons in the photon map.
t_3 is the trace depth of the ray-tracing step $t_3 \in [1; 5]$.

The number of rays n is when no supersampling is applied, given by the number of pixels of the screen.

To find a big-O expression, see Appendix F on page 179 for an introduction of the complexity, the functions O_1, O_2, O_3 have to be combined, and the terms not contributing significantly to the complexity can be excluded

$$\begin{aligned}
&O(O_1(n_i, t_1, o) + O_2(n_p, t_2, o, n_i) + O_3(n, t_3, n_p)) \\
&O(n_i\,(t_1 \cdot o + \log(n_i)) + n_p\,(t_2 \cdot o \cdot \log(n_i) + \log(n_p)) + n \cdot t_3 \cdot o \cdot nNN \log(n_p)) \\
&O\left(n_i \log(n_i) + n_p \log(n_i \cdot n_p)\right)
\end{aligned} \tag{7.4}$$

This expression for the complexity of gaze directed importon mapping will not be reduced further for now, since it later will be used for a comparison with traditional photon mapping.

7.3 Gaze Directed Photon Mapping

In this second algorithm only two passes are used, as in traditional photon mapping. The first pass encompasses construction of a gaze-contingent photon map, while the second pass render the scene with ray-tracing and information provided by the photon map.

To construct a gaze-contingent photon map efficiently, the concept of a photon emission map is introduced.

Photon Emission Map

The photon emission map is used to describe how to emit photons from a single light source. The most naive approach is to uniformly emit photons over the entire unit sphere, but when the photon map is combined with gaze directed rendering, this becomes inconvenient. The objective is to create a photon emission map that can be used to direct the photons in the direction of the point of regard.

The idea of gaze directed photon mapping is that it is possible to back-track the photons inside the gaze region, and then construct an emission map like seen in Figure 7.3. The photon emission map can be stored as e.g. a latitude longitude map which can be mapped onto the unit sphere surrounding the light source. Different colors/intensities can be used to indicate allowed emission directions. Consult Appendix C on page 165 for more information about latitude longitude photon emission maps.

Initially, when no emission map is available, the photon map is constructed with traditional photon mapping. During the first ray-tracing pass the gaze direction is used to mark photons in close proximity to the point of regard. Information stored in the photon map about how these photons ended up in the gaze region is used to construct a photon emission map for each light source. In order to enable the

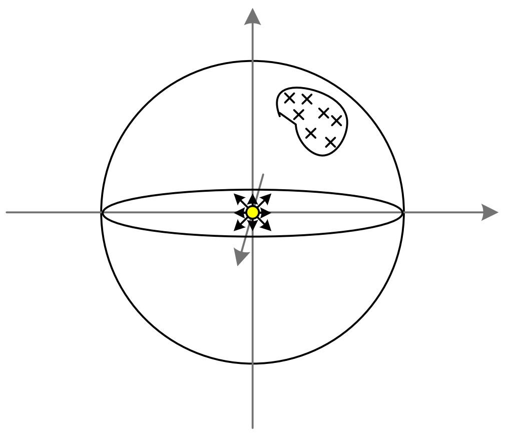


Figure 7.3: A small emission map projected onto the unit sphere encasing the light source.

emitted photons to indirectly reach the gaze region, the previous photon map is stored together with the photon emission map. In this way it is possible to consult the old photon map whenever a photon has to be reflected to obtain a direction that previously ensured that the photon ended up in the gaze region. Consequently, the gaze directed photon map is lagging one frame behind.

An advantage of the algorithm is that only two passes are needed thus it is expected to be faster than gaze directed importon mapping. A drawback is that if the gaze point between two consecutive frames is susceptible to large variations then the photon density will be high at the wrong place and low in the actual gaze region. An alternative to the requirement of small variations to the gaze point between consecutive frames is a high frame rate. Another drawback of the method compared to traditional photon mapping is a larger storage requirement. Not only is it required to store two photon maps, but the photon structure also has to be larger to store the additional information needed to build the photon emission map.

More details about the algorithm with illustrations are given next.

7.3.1 Algorithmic Description

As mentioned the photon structure had to be expanded in order be useful for creating a photon emission map efficiently, see Listing 7.2.

Listing 7.2: Structure of a photon for gaze directed photon mapping

```
struct photon{
    float x,y,z;            //position
    float p[3];             //photon power
    float indir[3];         //incident direction
    float outdir[3];        //outgoing direction used while back tracking
    char lightsource;       //id of light source
    photon* parent;         //pointer to parent photon
    short plane;            //splitting plane for kd-tree
}
```

The outdir defines the photon reflection vector if available, this is used to guide indirect illumination photons to the gaze region. The lightsource is used to identify the light source which emitted the photon, such that the correct photon emission map is updated. Finally, the parent pointer points to the last stored photon of the current photon trace, this information is also used when constructing the photon emission map.

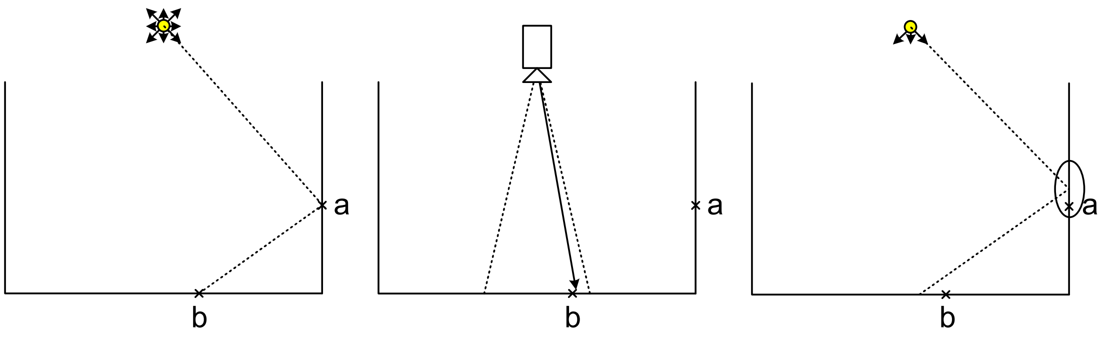


Figure 7.4: Illustration of the workings of gaze directed photon mapping.

The leftmost drawing of Figure 7.4 illustrates traditional photon mapping as used initially. Photons are stored at the crosses, first intersection is at `a` and the second is at `b`. The ray-tracing step is shown by the drawing the middle. Here a ray, indicated by the arrow emanating from the camera, uses photon `b`. This initiate an update of the photon emission map. Photon `b` locates its parent photon `a` by using the parent pointer. Then the indir of `a` is added to the photon emission map, since this is the first photon of the photon trace. The algorithm used to update of the photon emission map is shown in Algorithm 7.1.

The rightmost drawing of Figure 7.4 illustrates a photon mapping step after a photon emission map has been built. The photon emission map is indicated by the light source now emitting photons in specific directions, see Algorithm 7.2. A photon with an emission direction selected from the emission map (and some added noise) is traced from the light source, indicated by the dotted line, and intersects close to `a` of the old photon map. A look-up in the old photon map is now performed to locate nearby photons, as indicated by the ellipse. The reflection direction of the new photon is determined from the outdir of the nearby old photons, strategies for doing this are described in Section 8.4.7 on page 121 and Appendix D on page 171. Finally, this entails that the new photon is stored slightly left of the old `b` photon but still inside the gaze region of the previous frame, which was indicated by the two dotted lines emanating from the camera.

Algorithm 7.1 Recursive algorithm used to construct a photon emission map for the next photon emission step.

```
INPUT:
    A photon $photon$
OUTPUT:
    Updated photon emission map

$void$ update_photon_emission_map( $photon$ )

1. if ($photon.parent$ = null )
    • add $photon.indir$ to emission map
    • return 0
2. else
    • update_photon_emission_map( $photon.parent$ )
   end if
```

Algorithm 7.2 Algorithm for emitting photons in accordance with the emission map.

```
INPUT:
    Photon emission map $emission\_map$
OUTPUT:
    Photon map with stored photons

$void$ photon_emitter( $emission\_map$ )

1. do
    • generate random photon $dir$
2. while ($dir$ not conform to $emission\_map$)
    • tracePhoton(Ray($lightPos$, $dir + noise$), null, 0), see Algorithm 7.3
```

7.3.2 Complexity Evaluation

Gaze directed photon mapping is conceptually similar to traditional photon mapping, with the exception that the photon map of the previous frame is used to add gaze directed photons to the current frame. The algorithm consists of two passes, where the initial photon emission pass, Item 1a., is a special case and is not described here.

1. Photon Emission:

Algorithm 7.3 Algorithm for tracing photons.

```
INPUT:
    Ray ray
    Photon node
    Old photon map $pm\_old$

void tracePhoton(ray, node, level)
1. if ($level \geq trace\_depth$) return 0
2. else
    • x = getIntersectionPoint(ray, normal)
    • $plist$ = $pm\_old$.locate_nearby_photons(x, normal)          //array of nearby photons
    • create reflection dir, $rdir$, from $plist.outdir$
    • $p.pos \leftarrow x$, $p.indir \leftarrow ray$, $p.outdir \leftarrow rdir$, $p.parent \leftarrow$ addressOf(node) //create new photon, $p$
    • tracePhoton( Ray($p.pos$, $rdir$), $p$, $level + 1$)
    end if
```

a Emit photons as in traditional photon mapping.

b Emit photons in accordance with a photon emission map, constructed from the last ray-tracing step, see Algorithm 7.2.

2. Ray-trace the scene and for all photons that hit inside gaze region do a special back-track in order to construct a photon emission map for the next emission step, see Algorithm 7.1.

First a function describing the complexity of the photon mapping step is outlined

$$\begin{aligned} O_1(n_p, t_{\text{photon}}, o) &= n_p \cdot t_{\text{photon}} \cdot o \cdot \log(n_p) + n_p \log(n_p) \\ &= n_p \left(t_{\text{photon}} \log(n_p) \cdot o + \log(n_p)\right) \end{aligned} \tag{7.5}$$

where:

t_{photon} is the trace depth of the photon mapping step.
o is the number of objects in the scene.
$\log(n_p)$ is the complexity of the nearest neighbor photon search.
$n_p \log(n_p)$ is the complexity of balancing the photon map.

Secondly a function describing the complexity of the ray-tracing with back-tracking step is derived

$$O_2(n, t_{\text{ray}}, t_{\text{photon}}, o) = n \cdot o \cdot t_{\text{ray}} \cdot nNN \log(n_p) \cdot t_{\text{photon}} \tag{7.6}$$

where:

n is the number of rays to trace.

t_{ray} is the trace depth of the rays.

$nNN \log(n_p)$ is the complexity of looking up the irradiance estimate from the nNN nearest photons in the photon map.

The term t_{photon} contributes with the extra computations needed to call the recursive function used to build the photon emission map.

By combining $O_1(n_p, t_{\text{ray}}, o)$ and $O_2(n, t_{\text{ray}}, t_{\text{photon}}, o)$ and removing insignificant terms in terms of complexity it is possible to arrive at

$$\begin{aligned}
&O\left(O_1(n_p, t_{\text{ray}}, o) + O_2(n, t_{\text{ray}}, t_{\text{photon}}, o)\right)\\
&O\left(n_p\left(t_{\text{photon}} \log(n_p) \cdot o + \log(n_p)\right) + n \cdot o \cdot t_{\text{ray}} \cdot nNN \log(n_p) \cdot t_{\text{photon}}\right)\\
&O\left(n_p \log(n_p^2) + \log(n_p)\right)\\
&O\left(n_p \log(n_p^2)\right)
\end{aligned} \tag{7.7}$$

This is the complexity of the gaze directed photon mapping algorithm, later it will be compared with traditional photon mapping.

7.4 Gaze Directed Ray-tracing

The third algorithm is actually not a gaze directed photon mapping approach but merely a gaze directed ray-tracing algorithm with photon mapping.

The idea is to generate a relative high density general photon map during the photon mapping stage, but then save the extra time used to create this map in the ray-tracing pass. This is done by minimizing the amount of intersection tests by only tracing secondary rays within the gaze area. In the rest of the image only primary rays are used, the global illumination effects are then created solely by the irradiance estimate from the photon map.

7.4.1 Algorithmic Description

The algorithmic description of gaze directed ray-tracing is similar to normal ray-tracing, with the exception that the trace depth is reduced if the rays are outside the gaze area. This process is described in Algorithm 7.4.

The rest is just typical ray-tracing, see Appendix B on page 157.

Algorithm 7.4 Algorithm for varying the trace depth, for gaze directed photon mapping.

```
INPUT:
  Gaze point $gp$

void gaze_directed_ray-tracing()
1. for ($y \leftarrow 0$ to image.height)
  • for ($x \leftarrow 0$ to image.width)
    ◦ if ($\left((x-gp.x)^2+(y-gp.y)^2 > Threshold\right)$) $trace\_depth \leftarrow 1$
    ◦ else $trace\_depth \leftarrow 3$
    ◦ normal ray-tracing procedure
  • end for
  end for
```

7.4.2 Complexity Evaluation

Gaze directed ray-tracing is the simplest algorithm, and it relies on using as few rays for ray-tracing as possible, thus using secondary rays only inside the gaze area.

It is still a two pass method, where the first pass consists of photon mapping, but the photon mapping pass is not affected by the gaze direction.

1. Trace photons to create a photon map
2. Gaze directed ray-tracing, such that more rays are used in the important gaze area around the gaze point.

The function for determine complexity of the photon mapping stage of the gaze directed ray-tracing algorithm is not any different from traditional photon mapping

$$\begin{aligned} O_1(n_p, t_{\text{photon}}, o) &= n_p \cdot t_{\text{photon}} \cdot o + n_p \log(n_p) \\ &= n_p \left(t_{\text{photon}} \cdot o + \log(n_p)\right) \end{aligned} \tag{7.8}$$

In the ray-tracing stage the complexity is reduced corresponding to the size of the area that does not have to be rendered in high quality. Here 33 % of the scene is rendered with maximum quality, a number that was found from the Visual Perception test, see Section 4.3 on page 40.

$$O_2(n, t_{\text{ray}}, o, n_p) = n(0.33 t_{\text{ray}} + 0.67) \cdot o \cdot nNN \log(n_p) \tag{7.9}$$

When combining $O_1(n_p, t_{\text{photon}}, o)$ and $O_2(n, t_{\text{ray}}, o, n_p)$ to find the big-O expression for the complexity, the constants 0.33 and 0.66 are omitted thus the complexity becomes identical to traditional photon mapping

$$\begin{aligned}
&O\left(O_1(n_p, t_{\text{photon}}, o) + O_2(n, t_{\text{ray}}, o, n_p)\right)\\
&O\left(n_p\left(t_{\text{photon}} \cdot o + \log(n_p)\right) + n(0.33t_{\text{ray}} + 0.67) \cdot o \cdot nNN\log(n_p)\right)\\
&O\left(n_p \log(n_p)\right)
\end{aligned} \tag{7.10}$$

In the final comparison with traditional photon mapping the constants will return in order to show that gaze directed ray-tracing with photon mapping has a raison d'être.

7.5 Gaze Directed Hybrid Render

As an alternative to the gaze directed algorithms with full global illumination this idea tries to combine the strengths of both local and global illumination.

The strength of global illumination techniques are their ability to render visually realistic images, but this ability usually comes at a cost of render times that are unsuitable for real-time graphics.

On the other hand local illumination techniques such as those used by rasterization, are by hardware implementations very fast at rendering hundred of millions of polygons, but when it comes to visual realism they still have some shortcomings.

The gaze directed hybrid rendering algorithm is about combining the strengths and remove the shortcomings, by using eye tracking.

The idea is simple. A small region of the screen around the gaze point, the gaze area, is rendered with global illumination while the outer area is rendered with local illumination by hardware accelerated graphics. It is, with this approach, hopefully possible to achieve interactive/real-time frame rates, in addition to improved perceptual visual realism for the end user.

This alogrithm is quite similar to gaze directed ray-tracing, except that the outer area where only primary rays and the photon-map was used, is rendered with rasterization. This means that the photon map in the outer area has no importance. Therefore it may be possible to make use of the two first algorithms in the creation of the global illumination rendering in the gaze area to improve the frame rate further, by reducing the number of photons needed in the photon map.

7.5.1 Algorithmic Description

The principle of the algorithm is similar to the visual perception pre-test, see Section 4.3 on page 40, but with global illumination instead of the non-blurred image, and local illumination instead of the blurred image.

The scene is rendered with global illumination inside the gaze area, located at the gaze point acquired from the eye tracker. The ray-tracing part is carried out using the gaze point as the point where the camera is looking at, and in the gaze area within a defined radius the ray-tracing is performed. The size of the radius depends on the visual perception, in particular the visual angle. The result of the visual perception test can be seen in Section 4.3 on page 40. Photon mapping is added to capture all the global illumination effects of the scene.

The scene is also rendered with local illumination by the GPU using OpenGL. It requires that the scene can be created on the CPU using ray-tracing as well on the GPU with OpenGL. It is important that the radiance of the local illumination rendering is identical to the global illumination rendering such that they appear as one coherent rendering, more about this can be found in Chapter 5 on page 45.

When the two textures are acquired they are combined to a final rendering which is shown on the screen. The global illumination used at the gaze point, is gradually alpha blended into the local illumination background to get a smooth transition.

7.5.2 Complexity Evaluation

The complexity evaluation of the gaze directed hybrid rendering algorithm is using traditional photon mapping as the global illumination algorithm. It is also assumed that the local illumination rendering is free in terms of computation time.

The complexity of the photon mapping stage is identical to traditional photon mapping

$$\begin{aligned} O_1(n_p, t_{\text{photon}}, o) &= n_p \cdot t_{\text{photon}} \cdot o + n_p \log(n_p) \\ &= n_p \left(t_{\text{photon}} \cdot o + \log(n_p)\right) \end{aligned} \tag{7.11}$$

The second and final stage consists of the global illumination rendering, since it was assumed that the local illumination rendering and the combination of the two renderings can be made without using additional computation time. The global illumination rendering is similar to the ray-tracing stage of the gaze directed ray-tracing algorithm

$$O_2(n, t_{\text{ray}}, o, n_p) = \tfrac{1}{3} n \cdot t_{\text{ray}} \cdot o \cdot nNN \log(n_p) \tag{7.12}$$

As it can be seen only one-third of the screen is rendered with global illumination, which was found in the visual perception test, see Section 4.3 on page 40.

The total complexity of the two stages becomes

$$\begin{aligned} &O\left(O_1(n_p, t_{\text{photon}}, o) + O_2(n, t_{\text{ray}}, o, n_p)\right) \\ &O\left(n_p \left(t_{\text{photon}} \cdot o + \log(n_p)\right) + \tfrac{1}{3} n \cdot t_{\text{ray}} \cdot o \cdot nNN \log(n_p)\right) \\ &O\left(n_p \log(n_p)\right) \end{aligned} \tag{7.13}$$

7.6 Complexity Evaluation

In this section a complexity evaluation of the outlined algorithms will be presented. The evaluation is based on a comparison of the complexity of the algorithms with the complexity of traditional photon mapping. In this way each algorithm is compared to a common baseline and a fair evaluation can be made.

Since the complexity of photon mapping was not devoted much attention in the introduction to photon mapping, see Chapter 6 on page 57, a review will be given here.

7.6.1 Photon Mapping Complexity

The complexity of traditional photon mapping is based on a worst case scenario where the photon map has to be created and balanced for every frame rendered.

The complexity of the photon mapping step can be derived from

$$\begin{aligned} O_1(t_{\text{photon}}, o, n_p) &= t_{\text{photon}} \cdot o \cdot n_p + n_p \log(n_p) \\ &= n_p \left(t_{\text{photon}} \cdot o + \log(n_p) \right) \end{aligned} \tag{7.14}$$

where:

t_{photon} is the trace depth of the photons $t_{\text{photon}} \in [1;3]$.

o is the number of objects that needs to be tested for intersection $o \in [1;10^7]$.

n_p is the number of photons emitted $n_p \in [10^3;10^7]$.

$n_p \log(n_p)$ is the complexity of balancing the photon map.

The complexity of the ray-tracing step is

$$O_2(n, t_{\text{ray}}, o, n_p) = n \cdot t_{\text{ray}} \cdot o \cdot nNN \log(n_p)) \tag{7.15}$$

where:

n is the number of rays.

$nNN \log(n_p)$ is the complexity of a irradiance estimate from the photon map, from the nNN nearest photons.

t_{ray} is the trace depth in the ray-tracing step, typically $t_{ray} \in [1;5]$.

Combining the two steps and removing terms that have no significant influence on the execution time gives (it is assumed that $n_p \geq o$ to increase the probability that each object is hit by at least one photon)

$$\begin{aligned}
&O(O_1(t_{\text{photon}}, o, n_p) + O_2(t_{n,\text{ray}}, o, n_p)) \\
&O(n_p\left(t_{\text{photon}} \cdot o + \log(n_p)\right) + n \cdot t_{ray} \cdot o \cdot nNN \log(n_p)) \\
&O\left(n_p \log(n_p)\right)
\end{aligned} \tag{7.16}$$

The expression of the complexity will not be reduced further.

7.6.2 Growth Rate Evaluation

The O-notation expresses the growth rate of an algorithm and the first evaluation will compare the growth rates of the individual algorithms. Table 7.1 presents a summary of the complexity expressions found for each of the individual algorithms.

Table 7.1: Complexity differences between the different design ideas for gaze directed photon mapping.

Method	Complexity
Photon Mapping	$O\left(n_p \log(n_p)\right)$
Gaze Directed Importon Mapping	$O\left(n_i \log(n_i) + n_p \log(n_i \cdot n_p)\right)$
Gaze Directed Photon Mapping	$O\left(n_p \log(n_p^2)\right)$
Gaze Directed Ray-tracing	$O\left(n_p \log(n_p)\right)$
Gaze Directed Hybrid Render	$O\left(n_p \log(n_p)\right)$

The growth rates for increasing number of photons, from $10^3 - 10^6$, are plotted in Figure 7.5.

It can be seen that the growth rates for, traditional photon mapping, gaze directed ray-tracing and gaze directed hybrid render are better than for the two other algorithms. If for instance $7.8 \cdot 10^5$ photons are used in traditional photon mapping only $4.7 \cdot 10^5$ and $4 \cdot 10^5$ photons can be used for gaze directed importon mapping and gaze directed photon mapping respectively. When 10^4 importons are used in the importon map.

7.6.3 Complexity Comparison with Photon Mapping

The four proposed algorithms will in the following be compared with traditional photon mapping in terms of complexity.

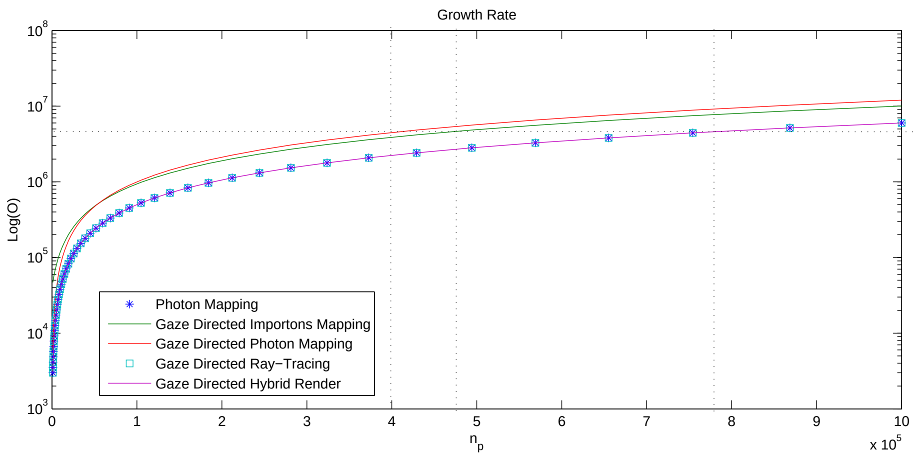


Figure 7.5: Growth rate for the four algorithms.

Gaze Directed Importon Mapping

To compare the gaze directed importon mapping approach with traditional photon mapping a ratio between the two is expressed

$$\frac{O_{\text{importance}}}{O_{\text{normal}}} = \frac{O\left(n_i \log(n_i) + n_{p1} \log(n_i \cdot n_{p1})\right)}{O\left(n_{p2} \log(n_{p2})\right)} \tag{7.17}$$

where:

n_{p1} is the number of photons used in the importance mapping approach.

n_{p2} is the number of photons used in traditional photon mapping.

The number of photons in importon mapping is denoted n_{p1} while the number of photons of traditional photon mapping is denoted n_{p2} this distinction is necessary because the two methods are supposed to yield comparable results within the gaze region at different photon levels.

The ratio between the two algorithms is set equal to one, such that they can be compared at equal complexity levels

$$\frac{O_{\text{importance}}}{O_{\text{normal}}} = \frac{n_i \log(n_i) + n_{p1} \log(n_i \cdot n_{p1})}{n_{p2} \log(n_{p2})} = 1 \tag{7.18}$$

The equation is analytically solved with MATLAB's Symbol Toolbox for n_{p1} for varying n_i, the results are plotted in Figure 7.6.

When the number of importons, n_i, are 10^4 then the number of photons, in the gaze directed importon mapping method has to be approximately $\frac{3}{5}$th of the photons in traditional photon mapping, for the two methods to achieve the same complexity. Thus for gaze directed importon mapping to achieve a lower

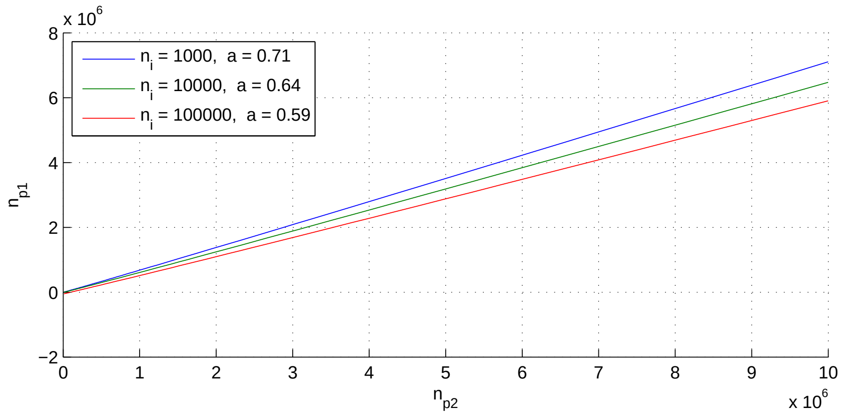


Figure 7.6: The plot shows how many photons that should be utilized in gaze directed importon mapping in order to achieve a complexity identical to traditional photon mapping, for between $10^3 - 10^5$ importons.

complexity the number of photons used in the photon mapping step must comply with

$$n_{p1} < 0.6 n_{p2} \quad | \quad n_i = 10^4 \tag{7.19}$$

Gaze Directed Photon Mapping

It is clear that the complexity of gaze directed photon mapping when compared to traditional photon mapping is worse.

Figure 7.7 shows the number of photons to use in gaze directed photon mapping to achieve the same complexity as traditional photon mapping.

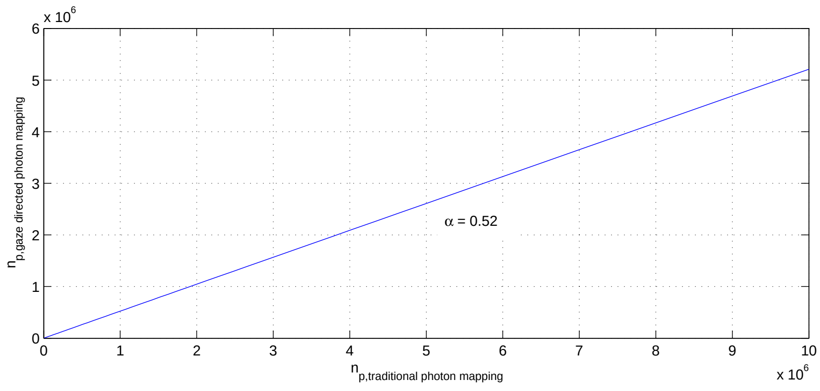


Figure 7.7: The plot shows how many photons that should be utilized in gaze directed photon mapping in order to achieve a complexity identical to traditional photon mapping.

The number of photons to use in gaze directed photon mapping in order to obtain similar or less complexity to traditional photon mapping is approximately given by

$$n_{\text{p,gaze directed photon mapping}} \leq 0.52 \cdot n_{\text{p,traditional photon mapping}} \tag{7.20}$$

This shows that gaze directed photon mapping is only slightly worse in terms of complexity than gaze directed importon mapping. This is likely to be negated by the fact that gaze directed photon mapping guide photons to the gaze region while gaze directed importon mapping might emit additional photons to ensure the desired amount is stored in the gaze region.

Gaze Directed Ray-tracing

The complexity of gaze directed ray-tracing and photon mapping is when evaluated with the big-O notation identical, see Table 7.1. But this is not a fair comparison because gaze directed ray-tracing is faster than traditional photon mapping in the ray-tracing step, which is not indicated by the big-O notation.

The ratio between the two ray-tracing steps are, when the gaze area with full global illumination spans 33 % of the screen

$$\frac{O_{2,\text{traditional photon mapping}}}{O_{2,\text{gaze directed ray-tracing}}} = \frac{n \cdot t_{\text{ray}} \cdot o \cdot nNN \log(n_p)}{n(0.33 t_{\text{ray}} + 0.67) \cdot o \cdot nNN \log(n_p)} = \frac{t_{\text{ray}}}{0.33 t_{\text{ray}} + 0.67} \tag{7.21}$$

Thus if the ray-tracing depth t_{ray} is e.g. three, which is quite common, then the ray-tracing step of the gaze directed ray-tracing algorithm will be approximately 1.8 times faster than traditional ray-tracing as used in photon mapping, when all other parameters are identical.

Gaze Directed Hybrid Render

A real complexity analysis of the gaze directed hybrid algorithm is similar to that of gaze directed ray-tracing only better, since the local illumination calculations of the outer area are considered to free in terms of complexity, since they are handled by the graphics hardware. The complexity ratio between the two ray-tracing steps are

$$\frac{O_{2,\text{traditional photon mapping}}}{O_{2,\text{gaze directed hybrid render}}} = \frac{n \cdot t_{\text{ray}} \cdot o \cdot nNN \log(n_p)}{\frac{1}{3} n \cdot t_{\text{ray}} \cdot o \cdot nNN \log(n_p)} \tag{7.22}$$

From this it can be seen that gaze directed hybrid rendering is expected to be three times faster than traditional photon mapping, just in the ray-tracing step.

7.6.4 Summary

The result of the complexity evaluation is that the proposed algorithms can be categorized in terms of their complexity, which translates to performance. The algorithms are categorized as either faster or comparable to traditional photon mapping, see Table 7.2.

Table 7.2: Algorithmic performance compared to traditional photon mapping.

Faster	Comparable
Gaze Directed Ray-tracing	Gaze Directed Importon mapping
Gaze Directed Hybrid Render	Gaze Directed Photon Mapping

Two algorithms were shown to have a complexity smaller than traditional photon mapping, and they are therefore place in the faster category. The reasoning for placing gaze directed importon mapping and gaze directed photon mapping in the category with comparable performance to traditional photon mapping, is that even though both methods have a slightly worse complexity, they are both designed to work with far less photons. This should compensate for the higher complexity. Something that will be investigate by the system test in Chapter 9 on page 127.

7.7 Gaze Directed Photon Power Scaling

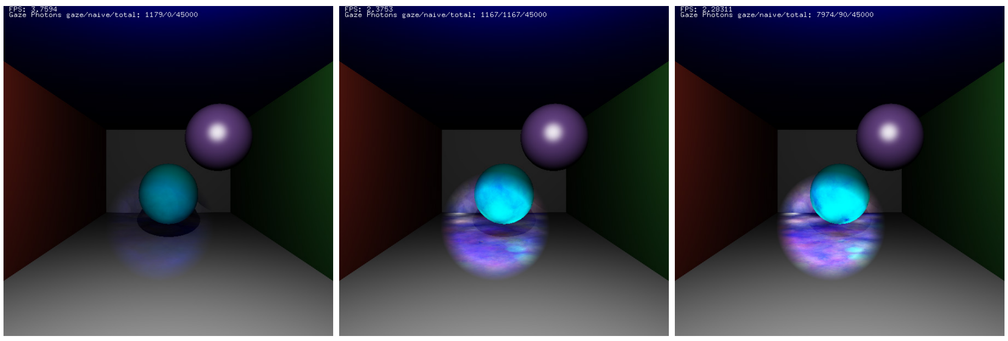
This section does not describe yet another perceptually adaptive graphics algorithm, but it describes a problem that must be dealt with.

A problem with both gaze directed importon mapping and gaze directed photon mapping is to retain the correct amount of photon power within the gaze region. When traditional photon mapping is used all photons are distributed uniformly in all directions from the light sources, which gives a uniform distribution of the power in the scene, since the power of the light source is divided equally between all emitted photons.

With both gaze directed importon mapping and gaze directed photon mapping a majority of the photons end up in the gaze region which gives a high concentration of photons in that region, and a low concentration elsewhere. It can be compared to exchanging the point light source with a spotlight which makes the gaze area very bright something which is not desirable. An example of the problem from an early prototype of the gaze directed photon mapping system is shown in Figure 7.8.

Therefore a method must be found to downscale the photon power correctly in the gaze directed photon map.

$$\Phi_{photon} = \frac{\Phi_{light}}{n} s \tag{7.23}$$



```
FPS: 3.7594
Gaze Photons gaze/naive/total: 1179/0/45000
```

```
FPS: 2.3753
Gaze Photons gaze/naive/total: 1167/1167/45000
```

```
FPS: 2.28311
Gaze Photons gaze/naive/total: 7974/90/45000
```

(a) Reference, rendered with traditional photon mapping.

(b) First iteration of gaze directed photon mapping.

(c) Second iteration of gaze directed photon mapping at the same location.

Figure 7.8: Example of gaze directed photon mapping in use without photon power scaling.

where:

Φ_{photon} is the power of the photon.

Φ_{light} is the power of the light source.

n is the number of photons emitted.

s is the scale factor $\in]0-1]$.

In the following some different solutions to find the value of the scale factor s are presented.

7.7.1 Photon Map Comparison

A straightforward solution is to always have two photon maps, one created with gaze directed photon/importon mapping and another created with traditional photon mapping, and then compare how many photons that are found in the gaze region of each map.

As an example both maps are populated with 10,000 photons. In the traditional photon map a total of 100 photons are located in the gaze region while it is 1000 for the gaze directed photon map. In other words for the traditional photon map to contain 1000 photons in the gaze region, it would have to be populated with 100,000 photons, therefore it can be concluded that the scale factor of the gaze directed photon map has to be $s = \frac{1}{10}$. The general formula for calculating the scale factor is derived in the following

$$n_{t,g} \cdot \frac{\Phi_{light}}{n_t} = n_{g,g} \cdot \frac{\Phi_{light}}{n_g} s \tag{7.24}$$

$$s = \frac{n_{t,g} \cdot n_g}{n_t \cdot n_{g,g}} \tag{7.25}$$

where:

n_t is the number of photons in the traditional photon map.

n_g is the number of photons in the gaze directed photon map.
$n_{t,g}$ is the number of photons located in the gaze region in the traditional photon map.
$n_{g,g}$ is the number of photons located in the gaze region in the gaze directed photon map.

Advantages	Disadvantages
• Ensures correct scaling of photon power independently of gaze position and scene geometry. • Conceptually straightforward.	• Computational expensive, because both a traditional and a gaze directed photon map has to be kept up-to-date each frame. • The number of traditionally emitted photons must be large enough to ensure that photons can be located in every possible gaze region such that a scale factor can be calculated.

In the following two methods for locating the gaze photons are presented, a correct but slow method and an approximative but faster method.

Nearest Neighbor Scaling

The approximative methods locates the n gaze photons inside a sphere with center at the point of regard, in both maps. This is done without considering whether the photons are located on the front facing side of the objects in the gaze region and without removing photons directly emitted to the gaze region (primary photons). These approximations are introduced to benefit the performance. This method is named Nearest Neighbor Scaling.

The radius r of the sphere is found by projecting the circular gaze area (located in the view plane) onto the scene geometry to determine the radius of the projected gaze area r_{gaze}. This radius is added to the search radius *max_dist* used when locating photons for shading, see Figure 7.9 for clarification.

The radius of the sphere r is calculated as

$$r = d \cdot \tan\left(\frac{fov}{2}\right) + max\_dist \qquad (7.26)$$

where:
d is the distance to the view plane.
fov is the field of view of the camera which captures the gaze area.

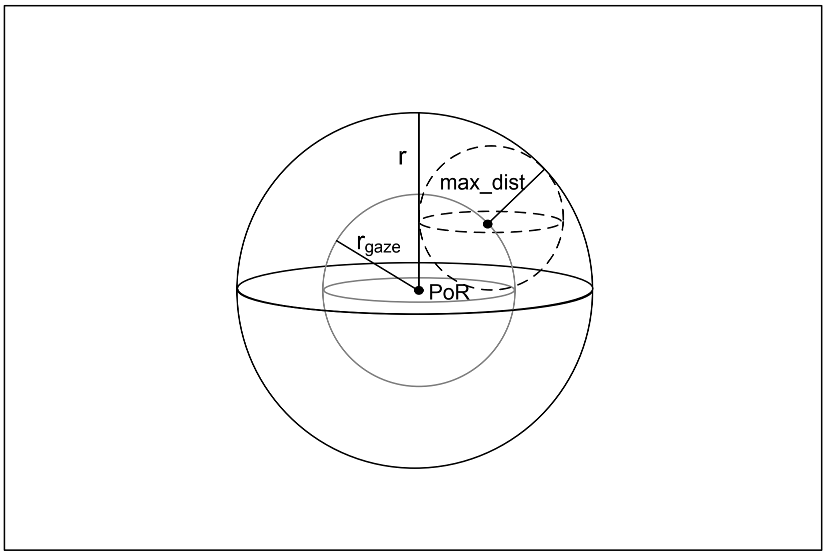


Figure 7.9: The radii involved in calculating the radius of the sphere encircling the Point of Regard (PoR).

Gaze Photon Scaling

The correct but more computational expensive solution to locate the *n* gaze photons is named Gaze Photon Scaling. This name is chosen because the methods only count the photons used to shad the gaze area, whether these photons are located in the gaze directed photon map or a traditional photon map. Thus only photons which are located on the front facing side of objects in the gaze region and secondary photons count to the total number of photons in the gaze region.

The preliminary work required to count of the photons are done during the shading of the pixels inside the gaze area. Here each photon that is deemed usable for shading, i.e. must have been indirectly traced to the gaze region and must be located on a front facing surface, is labeled as a gaze photon. When the shading is complete, the number of gaze photons can be found by counting the number of labeled photons. Obviously this is an expensive way of computing the scale factor because the photon map radiance look up has to be made in two different maps in order to mark the photons.

The scale factor can with gaze directed photon mapping be found without using a traditionally built photon map, details on how to do so are presented in Chapter 8 on page 101.

7.7.2 Solid Angles

The final solution is based on an idea proposed by Henrik Wann Jensen [Jensen, 2001]. His idea is used in combination with projection maps where the emitted photons are non-uniformly distributed and thus must be scale. The idea is to find the solid angle in which photons are emitted. For instance if the photons only are emitted within a hemisphere they must be scaled by $s = \frac{2\pi sr}{4\pi sr} = 0.5$.

The gaze directed importon mapping algorithm allows photons to be emitted in all directions, thus the solid angle of the emitted photons cannot be used. Instead the solid angle in which photons are emitted can be estimated from the gaze region where the majority of the photons will be stored. This is done by projecting the gaze region onto a unit sphere encasing the light source.

Advantages	Disadvantages
• Computational inexpensive. • Does not require photon map lookup.	• Assumes that the gaze region is a plan surface. • Does not take objects partly occluding the gaze area into consideration. • Objects occluding the light source are omitted.

7.8 Feasibility Study

It is common for all the algorithms outlined, that the gaze area is rendered with global illumination. This feasibility study will examine how the outer area should be rendered. The optimal solution includes fast rendering and is perceptually undetectable to the user. The outer area can either be rendered with local illumination, e.g. simple ray-tracing or OpenGL rasterization, or it can be rendered with global illumination in low quality, as proposed by some of the algorithms.

7.8.1 Test Setup

This test uses the same framework as the visual perception test, see Section 4.3 on page 40. Instead of using a blurred and a non-blurred image, a high quality rendering and a low quality rendering are used, and the visual angle of gaze area is kept constant at 20 degrees. The position of the high quality area is determined by the gaze position instead of a pre-defined pattern.

For the test five images of the same scene but rendered with different rendering methods are used. The resolution of the images is 1680$\times$1050. All the images are created and rendered in Autodesk 3ds Max. The most important settings for the different renderings can be seen in Table 7.3.

The GI1 rendering is the highest quality rendering, it is always is shown in the gaze area, and the four other images are used in the outer area. The local illumination images are rendered with ambient lighting in an attempt to match the illumination levels better with the global illumination images. The glass sphere in the LI2 is rendered with ray-tracing to get the reflections in the sphere, otherwise it will just be transparent and thereby look very different from the other images. Some of the renderings of the scene can be seen in Figure 7.10.

For the test, ten test subjects are used. The test subjects are ordinary computer users without any rendering or illumination expert knowledge. First the eye tracker is calibrated to the test subject to get the best eye tracking. The test subject is then asked to look at the image, and notice what they observe. The GI1 rendering is shown within the gaze area and is by alpha blending, gradually blended with one

Table 7.3: The rendered images and the most important settings.

Rendering	Ray-tracing	Amb. light	Max. samples pr. pixel	Rays pr. FG[1] point	No. photons	Rend. time
GI1	+	-	4	30	20,000	14:36
GI2	+	-	4	15	10,000	11:33
GI3	+	-	1	-	10,000	8:35
LI1	+	+	1	-	-	7:49
LI2	-	+	1	-	-	2:47

[1] Final Gather

of the other lower quality renderings. Afterwards the person is asked to answer a questionnaire, the questionnaire can be seen in Appendix E on page 175. This test procedure is repeated for the three remaining lower quality renderings. The order of the renderings is randomly chosen, to avoid biasing the answers of the questionnaires.

7.8.2 Results

The answers to the questionnaire can be seen in Appendix E on page 175. The two most important questions in the questionnaire were, whether the rendering appeared visual realistic and whether the rendering seemed like one coherent image. Eight out of ten agreed that the rendering appeared visually realistic with the lowest quality image (LI2), while seven persons agreed or strongly agreed that it seemed like one coherent image. The persons were only a little more positive about the other test images. In other words, the lowest quality image (LI2) used in the outer area and the good quality at the gaze area is sufficient for 80 % of the users.

So from the results, a low quality rendering (rasterization) in the outer area would be the optimal solution, since it is computationally simpler, and according to the test, it is perceived nearly as visually realistic as the best quality rendering.

Lastly it should be noticed that the design of the test is presumably a best case scenario in terms of detecting the differences between the two renderings. Since the test users were aware of the purpose of the test and since the test images were static they had all the time they wanted to try to discover any differences. If the general idea were implemented in e.g. a computer game, where the users would be busy with the task of the game, then differences and e.g. flicking is believed to be less of a problem.

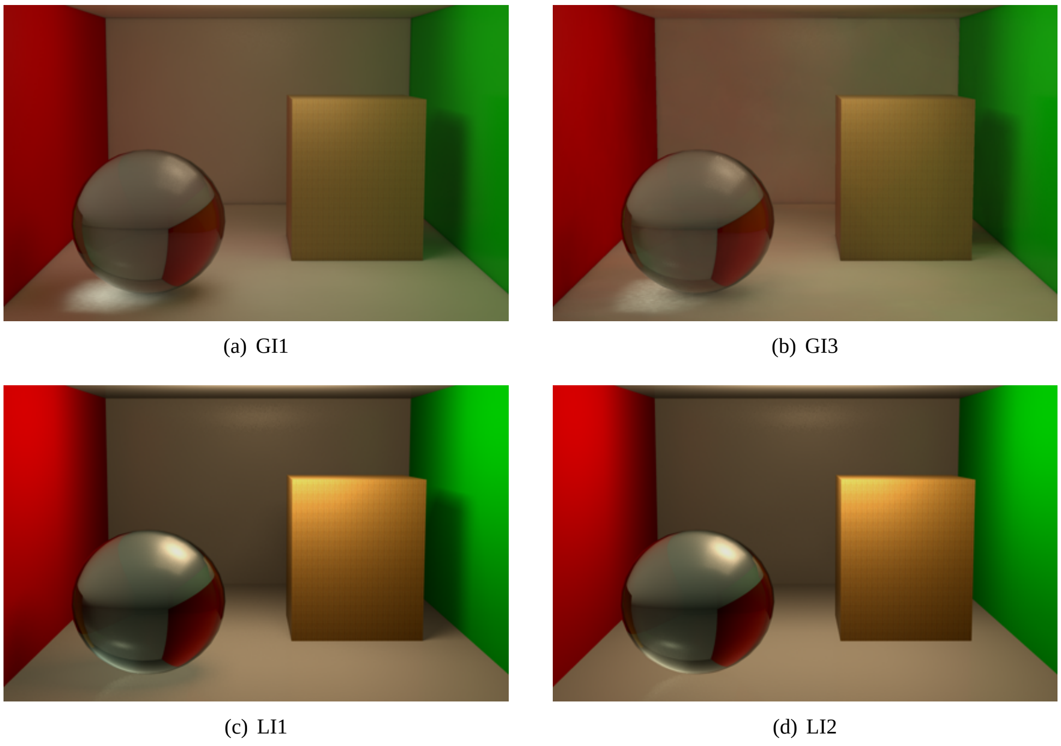


Figure 7.10: Four of the five images used. The GI2 image is omitted because it is almost impossible to see any difference from GI1. All the images are rendered with one area light above the two objects.

7.9 Algorithm Selection

This section will contain the final selection of which combination of algorithms to implement and test. The selection is based on the properties outlined in the preceding sections of the chapter.

The four algorithms to choose from are:

1. Gaze Directed Importon Mapping
2. Gaze Directed Photon Mapping
3. Gaze Directed Ray-tracing
4. Gaze Directed Hybrid Render

The basis for the selection of algorithm is the problem statement, see Section 3.2 on page 31. The problem statement emphasizes that photon mapping in combination with eye tracking should be used to improve rendering performance without perceptual degradation. Based on this it would be natural to select the algorithm with the lowest complexity, since all of the algorithms are designed to at least provide the same perceived visual quality. But since perception is also a strong part of the problem statement then it would be unwise to ignore the findings of the feasibility study and base the selection of algorithm on complexity alone.

When the two sources of information are combined one particular algorithm stands out as being the optimal solution to solve the problem, namely the gaze directed hybrid render. It is both fast and the feasibility study showed it is perceptually as good as the other algorithms.

The decisive factor in favor of the gaze directed hybrid render was the significant finding of the feasibility study that most users were unable to perceive that the outer area was of lower quality than the high quality version of the gaze area.

With this knowledge it is difficult to argue in favor algorithms based solely on global illumination and photon mapping, because no matter how they are sought optimized by eye tracking they will with current hardware not be as fast as a local illumination technique. And by combining the two techniques it is possible to use global illumination in the gaze area and simply let the outer area be rendered without global illumination. The benefits of this hybrid approach are the possibility to provide higher global illumination quality or focus on real-time performance while maintaining visual quality superior to full screen local illumination.

Inside the gaze area, it is still possible to apply some of the other proposed algorithms, namely gaze directed importon mapping or gaze directed photon mapping. The most useful algorithm for this purpose is gaze directed photon mapping, since the amount of photons in the photon map by this approach can be reduced drastically, proving a potential performance gain. The photon power scaling problem described will be alleviated by use of the gaze photon scaling approach, see Section 7.7.1 on page 93, because it favors correctness and with some possible implementation optimizations, it is believed to have neglectable impact on performance.

CHAPTER 8 IMPLEMENTATION

The implementation of gaze directed hybrid rendering and gaze directed photon mapping will be described in this chapter. The description is focused on the newsworthy aspects of the two algorithms, but will also provide a brief overview of the entire system.

8.1 Introduction

In the design section it was decided to use gaze directed hybrid rendering and gaze directed photon mapping as the global illumination algorithm in the hybrid render. Since the implementation reflects this choice, it has been divided into two main parts: one dealing with the gaze directed hybrid rendering and another dealing with gaze directed photon mapping. The complete entity of this implementation is denoted ”the system”.

The system consists of more than hybrid rendering and gaze directed photon mapping but this chapter is focused on the contributions within these two fields. The remaining parts of the system are only briefly dealt with in the system overview.

The purpose of the implementation is to provide evidence for the viability of the proposed hybrid rendering and gaze directed photon mapping algorithm. But also to provide a framework to test gaze directed photon mapping versus traditional photon mapping.

8.1.1 Guidelines

The implementation has been guided by a set of guidelines or ”good intentions” which are listed below.

- Readability of the code must not be sacrificed for computation speed.
- The system is designed with computation speed as main priority before memory consumption.
- The rendering system is designed to support High Dynamic Range (HDR) renderings.
- The dependency on external libraries is sought minimized.

Readability

It has been chosen to favor readable code on behalf of computation speed because this project is a learning and research project. And if other people are to use the results it is an advantage if the code appear as concise and readable as possible, without any dirty hacks just to save an instruction or two.

Computation Speed

When the goal of writing readable code has been accomplished, then computation speed is still essential. Therefore computation speed is a priority over memory consumption, because memory is no longer a scarce resource in modern computers.

HDR Support

When ray-tracing is calculated on the CPU then the savings of calculating colors in 8 bit compared to 32 bit floats is relative small because modern CPUs are highly optimized for floating point operations. Because of that it would not make sense to sacrifice this accuracy in the remaining parts of the rendering pipeline, which have much less work to do than the CPU, and thus easily can handle the extra work in calculating textures etc. in HDR.

External Libraries

The motivation behind this guideline is the old saying: ”learning by doing”. Since it is believed that a higher level of insight is obtained by developing a majority of the code ourselves without relying on external libraries and Application Programming Interfaces (APIs). The used libraries and APIs are listed in Table 8.1.

8.1.2 Limitations

The system has the following limitations.

- No advanced scene data structure is used.

Table 8.1: External libraries and APIs.

Name	Purpose
OpenGL	Used for rendering and display of ray-tracing [Khronos Group, 2008].
OpenCV	Used for image loading and saving, and for a output from the ray-tracer [Intel, 2008].
Tobii API	Provides access to the Tobii eye tracking hardware, as well as some calibration tools [Tobii, 2001].
glut	Used to handle OpenGL window management, keyboard and mouse input [Khronos Group, 2009].
glutMaster	C++ wrapper for glut [George Stetten and Korin Crawford , 1997].
Photonmap	The photon map code is based on the example code provide by Henrik Wann Jensen in [Jensen, 2001].

- The ray-tracer does not support texturing.
- The system supports a maximum of four light sources.
- Resolution of ray-traced image is static at run time. Changing the resolution of the window, will only stretch the ray-traced image.
- The field of view of the OpenGL rendered scene is non-changeable at run time.
- Does not support caustics created by a separate caustics map.

The main motive for these limitations is to concentrate the effort on gaze directed hybrid rendering and gaze directed photon mapping, by saving time on some subject that otherwise could constitute to a project on their own.

8.2 System Overview

A class diagram of the complete system with the most significant classes, is illustrated in Figure 8.1. The class diagram is simplified to only show the classes and their connections, i.e. all attributes and operations are omitted. The system is as illustrated divided into four main packages, these packages will in the subsequent sections be denoted subsystems.

In the following the subsystems will be described briefly to give a complete overview of the system. The `Photon Mapping` and `Display` subsystems are the most important subsystems of this project, and will therefore be described in more detail later, see Sections 8.3 and 8.4. The attached source code can be consulted for more details about the other subsystems.

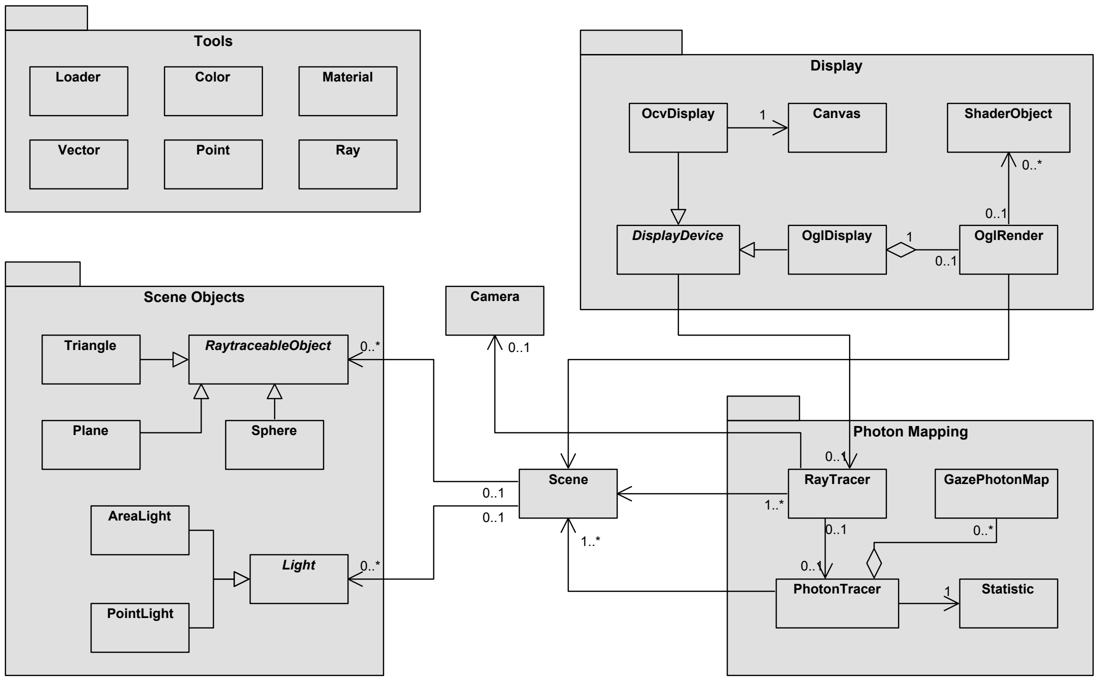


Figure 8.1: Simplified class diagram of the entire system.

Tools

The `Tools` subsystem contains a set of useful classes that is used elsewhere in the system.

The `Loader` class loads a representation of the scene written in a text file, that has to be rendered. From the information in the text file the scene is constructed with geometry, lighting, and camera settings.

The math operations required for ray-tracing and photon mapping is contained in the classes `Vector`, `Point` and `Ray`. The `Material` class is used to define the material of the objects, and the `Color` class is used for the materials and for the renderings.

Scene Objects

The `Scene Objects` subsystem contains the objects that can be used in the scene.

Three different geometry classes are implemented, `Plane`, `Sphere`, and `Triangle`. Each geometry class stores the information necessary to define the geometry, and functions needed to determine whether a ray intersects with the geometry or not.

The `Light` class is used to define the lighting in the scene. It is possible to choose either `PointLight` or `AreaLight`, whereof only the point light is fully implemented.

Camera and Scene

The `Camera` class is used to define the position and the field of view of the viewer in 3D space. It also defines the size of the view plane of the ray-tracer, which is used to create the rays that are shot into the scene. The position and the look at direction are used by both the ray-tracer and the OpenGL render.

The `Scene` class stores objects and light sources of the scene that has to be rendered. The `Scene` class is used by the photon tracer and ray-tracer to access properties of the objects and light sources of the scene, e.g. position and energy of the light sources. Since the `OglRender` class is responsible for rendering the scene in OpenGL, it also has access to `Scene`.

Display

The `Display` subsystem manages the local illumination rendering of the scene, and merges it with the global illumination rendering produced by the ray-tracer and photon mapping.

The `Display` subsystem also supports two types of output devices, namely an OpenCV or OpenGL display. The `OcvDisplay` is limited to only show the ray-traced scene, whereas the `OglDisplay` is implemented to display both the ray-traced and OpenGL rendered version of the scene.

The `OglRender` class is used to render the local illuminated scene in OpenGL and to control the shaders used in the system. It is also used to merge the local illumination rendering and global illumination rendering of the scene.

Photon Mapping

The `Photon Mapping` subsystem contains the classes necessary to construct and use both traditional and gaze directed photon mapping. The `PhotonTracer` class is responsible for emission and tracing of photons in the scene. To carry out this work it uses the `Statistic` class which provides different functions for generating random numbers.

The storage of and access to photons is handled by the `GazePhotonMap` class.

Finally, the global illumination rendering is handled by the `RayTracer` class. The ray-tracer can be self-contained, but is here using the `PhotonTracer` in order to implement photon mapping. The aspects of the ray-tracer that neither deals with hybrid rendering or gaze directed photon mapping is not devoted further attention in the report, but Appendix B on page 157 can be consulted for a review about ray-tracing.

8.3 Display Subsystem

The `Display` subsystem will in this section be described more detailed than in the previous section, since it contains several functionalities that are crucial to the system.

First an overview of the class diagram of the `Display` subsystem is given before the various important subjects are described in detail.

8.3.1 Overview

A detailed class diagram of the `Display` subsystem can be seen in Figure 8.2. The primary task of the `Display` subsystem is to render the scene with local illumination in OpenGL and merge it with the global illumination rendering, where the global illumination is shown in the gaze area. Furthermore it handles all keyboard inputs and window displaying.

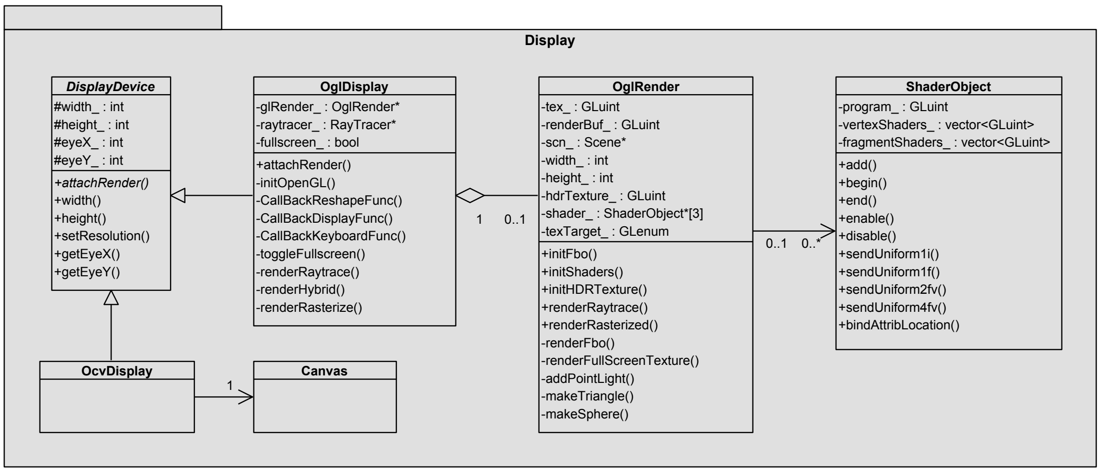


Figure 8.2: Detailed class diagram of the display subsystem, with its most essential attributes and operations.

The two most important classes are `OglDisplay` and `OglRender`, some of their functions will be explained in the following.

When the system is launched it instantiates a `DisplayDevice` object of the type `OglDisplay`. The `OglDisplay` constructor takes two arguments which are the width and height of the OpenGL window. The OpenGL window is not created at this point but first when a reference to a `RayTracer` object is bound to the `OglDisplay` by using `attachRender`.

Before this can happen a `RayTracer` object must be instantiated, which takes a `Scene` object and a `Camera` object as parameters.

When `attachRender` is called with a pointer to the `RayTracer` object, then the remaining OpenGL settings and keyboard/mouse input settings will be initialized and a window, with a local illumination rendering of the scene produced with OpenGL, will be shown. This initialization is shown in Algorithm 8.1.

Algorithm 8.1 Initialization of the display subsystem.

```
INPUT:
    Scene to render "myScene.scene"
OUTPUT:
    OglDisplay ready to use

void System_Startup()

1. Instantiate Cam cam
2. Instantiate Scene scene
3. Instantiate OglDisplay display($width, height$)
4. Instantiate Loader loader(scn, cam, display, "myScene.scene")
5. Instantiate RayTracer raytracer(scn, cam)
6. display.attachRender(raytracer)
```

Creating a `OcvDisplay` device is done in a similar manner but it has fewer options, so this section will concentrate on the `OglDisplay` device.

The `OglDisplay` object contains two important member pointers, these are `raytracer_` and `glRender_` which is used to invoke rendering with either ray-tracing or OpenGL, more on this in next section.

8.3.2 Gaze Directed Hybrid Rendering

This section describes how gaze directed hybrid rendering is implemented, by describing the different rendering types and how the scene in OpenGL is rendered to a texture and merged with the global illumination rendering to the final displayed texture.

Rendering Types

The `Display` subsystem is capable of producing three different types of renderings.

- A local illumination rendering, created with OpenGL, using the `RenderRasterized()` call.
- A global illumination rendering, created with the ray-tracer and photon-mapping if enabled, using the `RenderRaytrace()` call.
- A hybrid rendering which is a combination of the two renderings above, using the `RenderHybrid()` call.

All three renderings are created and updated at run time, and can be enabled/disabled by the user. The display loop function controlled by glut is `CallbackDisplayFunc`. Pseudo code for this function is listed in Algorithm 8.2

Algorithm 8.2 Pseudo code for the main display loop. The function calls differs from those listed in the previous item list, because this algorithm outlines what these functions actually do.

```
INPUT:
    A screen and a configured display device
OUTPUT:
    Display of selected rendering

void Display_Loop()

1. while(exit $\neq$ true)
    • switch(render type)
        ○ case local illumination
            · glRender->renderRasterized(..., true)
            · break
        ○ case global illumination
            · Set camera point of regard to $EyeX, EyeY$
            · hdrImage = raytracer_->renderScene()
            · glRender->renderRayTrace(hdrImage, ...)
            · break
        ○ case hybrid
            · glRender->renderRasterized(..., false)
            · Set camera point of regard to $EyeX, EyeY$
            · hdrImage = raytracer_->renderScene()
            · glRender->renderRayTrace(hdrImage, ...)
            · break
    • end switch
    end while
```

The display loop indicates that the hybrid rendering is just a local illumination rendering followed by a global illumination rendering. This is in fact exactly how the hybrid rendering is produced. The only difference between the individual renderings and the hybrid rendering is the second parameter in the `renderRasterized` call, which tells the `glRender` object whether to show the rendering now or save it for later. Later is in this case when the global illumination rendering is produced and about to be displayed by `renderRayTrace`.

Render to Texture

The `Display` subsystem uses the concept of render to texture. This means that everything is rendered to or converted to an OpenGL texture, such that there is a texture with the global and one with the local illumination rendering. The concept of render to texture is used because the hybrid rendering then can be created in 2D image space, as a texture operation.

Traditionally, OpenGL renderings had to be rendered directly to the frame buffer, from where they could be read back into a texture. Read backs of this type were considered performance bottlenecks and thus avoided. With the introduction of the FrameBuffer and RenderBuffer extension to OpenGL 2.0 it has been made possible to setup off-screen render targets which behave like the frame buffer but where the rendering is stored in a texture. This concept of Frame Buffer Objects (FBO) is used to render the local illumination version of the scene directly to a texture.

The textures used in this project are 32 bit textures, in order to comply with the HDR guideline, and to allow tone mapping of both the local and global illumination rendering.

A pseudo code of how the FBO is used is shown in Algorithm 8.3, this pseudo code resembles the functions `initFbo` and `renderFbo`. Since the local illumination rendering does not require stencil or accumulation buffers, the only off-screen buffers the FBO has to manage is a color buffer and a depth buffer.

Algorithm 8.3 Pseudo code for rendering to texture by using frame buffer objects.

```
INPUT:
    Camera
    Scene
OUTPUT:
    Local illumination texture

void initFbo()
1. Create FBO with size of DisplayDevice
2. Bind FBO to texture

void renderFbo()
1. Use FBO
2. Setup the modeling matrix from the camera viewing transforms
3. Render the scene
4. Use frame buffer again
```

The rendering of the scene to the FBO can use shaders and other functionalities as when rendering to a real frame buffer.

With the local illumination texture created, only the global one is missing before they can be displayed either separately or as a hybrid rendering.

To ease the creating of the global illumination texture, it was decided to let the ray-tracer store its radiance values in an array of the type `Color`. That makes it easy to copy the texture to the memory of the graphics card, see Figure 8.3.

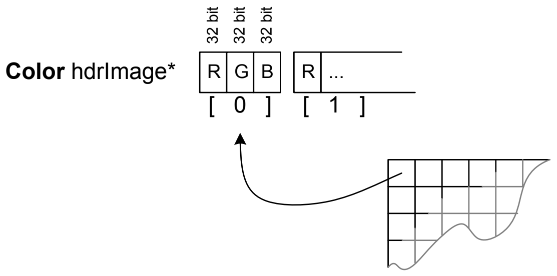


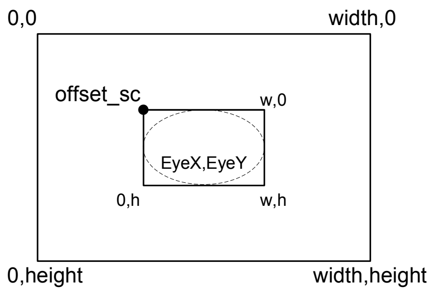
Figure 8.3: Array used to hold the pixels of the image. Each pixel is represented by a Color, which consist of three 32 bit floats. This structure allows fast copying of the rendering to the texture memory of the graphics card.

Display Textures

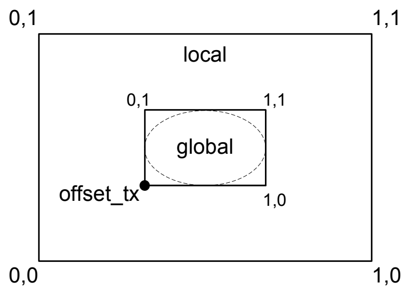
The local or global illumination rendering is displayed by rendering a full screen quad to the frame buffer with the respective texture applied. This is done with an exponential tone mapping shader enabled, to map the 32 bit texture values to 8 bit values as supported by the frame buffer and most monitors, see Section 8.3.3.

The hybrid rendering uses the same method with a full screen quad, but uses a special `hybrid_render` fragment shader that decides when to do texel lookup in the local or the global illumination texture. For the fragment shader to be able to make this decision correctly it needs information, about the ratio of the size of the two textures, and the offset (the lower left corner) of the global illumination texture in relation to the local illumination texture. Figure 8.4(a) shows how the textures are to be placed on the screen, and Figure 8.4(b) shows which information OpenGL has access to, in order to place the textures. Each texture has their own texture unit, so texels from both textures are accessed by a value from 0.0 to 1.0.

The values needed by the `hybrid_render` vertex and fragment shaders to place the textures correctly ,as shown in Figure 8.4(a), are calculated as



(a) View of the placement of the local and global illumination renderings from screen space.



(b) View of the local and global illumination textures, in texture space.

Figure 8.4: Merge of the local and global illumination texture into one coherent hybrid rendering.

$$ratio_x = w/width \tag{8.1}$$

$$ratio_y = h/height \tag{8.2}$$

$$offset\_sc_x = EyeX - 0.5w \tag{8.3}$$

$$offset\_sc_y = EyeY - 0.5h \tag{8.4}$$

$$offset\_tx_x = \frac{offset\_sc_x}{width} \tag{8.5}$$

$$offset\_tx_y = \frac{height - (offset\_sc_y + h)}{height} \tag{8.6}$$

where:

h and w is the height and width of the global illumination area/gaze area.

$height$ and $width$ is the size of the OpenGL rendering.

$EyeX$ and $EyeY$ is the coordinate of the gaze point.

In this way the $EyeX, EyeY$ coordinate set given by the eye tracker controls where the local and the global illumination texture is used. The $EyeX, EyeY$ coordinate is updated asynchronously by the eye tracker API as soon a new updated eye coordinate becomes available.

8.3.3 Shaders

This section will introduce the OpenGL shaders used for gaze directed hybrid rendering.

Hybrid Render

The vertex program of the hybrid render shader, is responsible for rendering the global illumination texture at the screen position $offset\_sc_x, offset\_sc_y$. This is done by modifying the transformed vertex position, `gl_Position`, by adding $offset\_sc_x, offset\_sc_y$ to the position of `gl_Vertex` before multiplication with the model view and projection matrix.

The task of the fragment shader is to blend the two textures by a gradient starting from fovea, which is set to 0.7 times the radius of the gaze area. It is also responsible for replacing the corner texels of the global illumination texture with texels from the local illumination texture such that the global texture looks circular, see Figure 8.5.

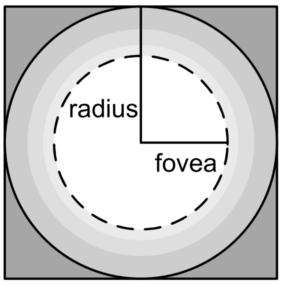


Figure 8.5: The global illumination texture is by a gradient blended into the local illumination texture. The gradient starts at the periphery of fovea and ends at the periphery of the gaze area.

Pseudo code for this is presented in Algorithm 8.4.

Tone Mapping

The tone mapping shader is a simple exponential tone mapping operator, which has the formula

$$c_{\text{out}} = 1.0 - e^{-exposure \cdot c_{\text{in}}} \tag{8.7}$$

where:

c_{out} is the color writen to the frame buffer.
$exposure$ is a user controlled exposure.
c_{in} is the color of the current fragment in HDR.

Modified Blinn-Phong

The global illumination rendering is created using modified Blinn-Phong shading, and if the hybrid rendering is to look as one coherent rendering then the local illumination rendering must use the same shading technique. The fixed function pipeline in OpenGL implements Gouraud shading, therefore must

Algorithm 8.4 Hybrid render fragment shader code.

```
INPUT:
    sampler2D $global\_illumination$, $local\_illumination$          //textures
    vec2 $GI\_tex$ and $LI\_tex$                                     //texture coordinates
    vec2 $offset = (offset\_tx_x, offset\_tx_y)$ and vec2 $ratio = (ratio_x, ratio_y)$
OUTPUT:
    Hybrid rendering

void hybrid_render_fragment_shader()

1. $radius \leftarrow 0.5$, $fovea \leftarrow 0.7 \cdot radius$          //in texture coordinates
2. $color \leftarrow$ texture2D($global\_illumination$, $GI\_tex$)
3. $color \leftarrow$ tone_mapping($color$)
4. $d \leftarrow$ distance($GI\_tex$, vec2$(0.5, 0.5)$)
5. $LI\_tex \leftarrow LI\_tex \cdot ratio$
6. $LI\_tex \leftarrow LI\_tex + offset$
7. if( $d \geq 0.5$ )                              //outside circle, use local illumination
    • $color \leftarrow$ texture2D($local\_illumination$, $LI\_tex$)
8. else                                          //inside circle, use global illumination
    • if ($d \geq fovea$)                         //outside fovea, blend with local illumination
        ○ $\alpha \leftarrow (d - fovea)/(radius - fovea)$
        ○ $color \leftarrow \alpha \cdot$ texture2D($local\_illumination$, $LI\_tex$) $+ (1.0 - \alpha) \cdot color$
    • end if
    end if
```

modified Blinn-Phong shading be implemented as a custom shader. The modified Blinn-Phong shader is implemented according to Section 5.2.1 on page 52 and it consists of both a vertex and a fragment program. For more details about the implementation look at the files `modified_blinn_phong_perfragment.vertex` and `modified_blinn_phong_perfragment.frag` on the DVD.

8.4 Photon Mapping Subsystem

This section describes the implementation details of the developed gaze directed photon mapping algorithm.

First a review of the most important functions of the photon mapping subsystem is given, after that the specific details on gaze directed photon mapping are presented.

8.4.1 Overview

Besides the developed gaze directed photon mapping the photon mapping subsystem also contains a ray-tracer. The description of the ray-tracer is focused on changes required for gaze directed photon mapping to work. For a more general ray-tracer description consult Appendix B on page 157.

A detailed class diagram of the photon mapping subsystem can be seen in Figure 8.6.

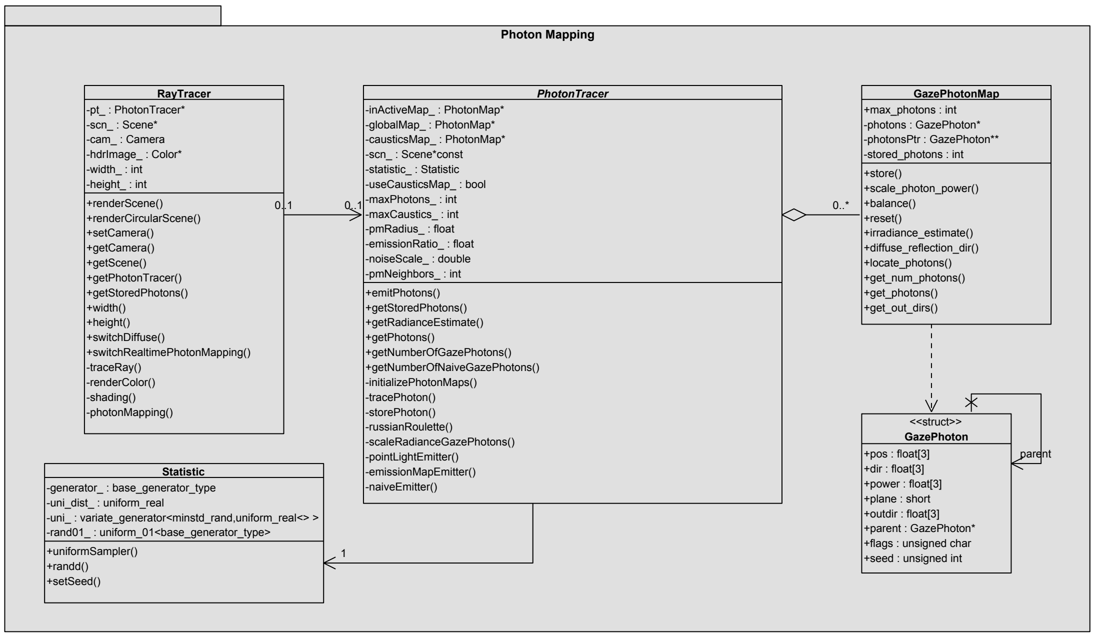


Figure 8.6: Detailed class diagram of the photon mapping subsystem, with its most essential attributes and operations.

The photon mapping subsystem is launched via `renderCircularScene` or `renderScene` in the `RayTracer`. The `renderCircularScene` function is used if the rendering is for a hybrid rendering where a circular gaze area is necessary, while the `renderScene` function is used if the gaze area is rectangular.

First a photon map is built by calling `emitPhotons`. When the photon map(s) (traditional or gaze directed) are generated then a rendering of the scene is made by shooting rays through the pixels within the gaze area. This is done with traditional ray-tracing. The `renderColor` function is used to find the radiance of the pixel the ray is shot through. First it calls `traceRay` to locate the intersection point with the closest object. The `renderColor` function is recursive, and calls itself for instance if the object is reflective. For each call it calculates the radiance contribution for both specular, direct illumination, and indirect illumination obtained from the photon map.

As stated above the ray-tracer initiates the photon mapping by calling `emitPhotons` which for each point light source calls `pointLightEmitter`. The `pointLightEmitter` calculates how many photons it should emit, and if gaze directed photon mapping is used, then `emissionMapEmitter` is called, otherwise `naiveEmitter`, that emits photons in traditional way, is called.

The review of the important functions of the photon mapping subsystem revealed that it is possible to divide the rendering with gaze directed photon mapping into three steps.

1. Photon emission
2. Ray-tracing (append gaze information to the photon map)
3. Render and display scene

The photon emission step described here mainly deals with the aspects of gaze directed photon emission. The traditional photon emission case is described in Section 6.2.1 on page 58. The rendering and display step is described in Section 8.3 on page 106.

First the photon data representation is presented as a basis. Secondly the gaze directed ray-tracing step is introduced, this order is chosen to aid the understanding of the gaze directed aspects of the photon emission which are dealt with last. A flow diagram of the gaze directed photon mapping subsystem is presented in Figure 8.7.

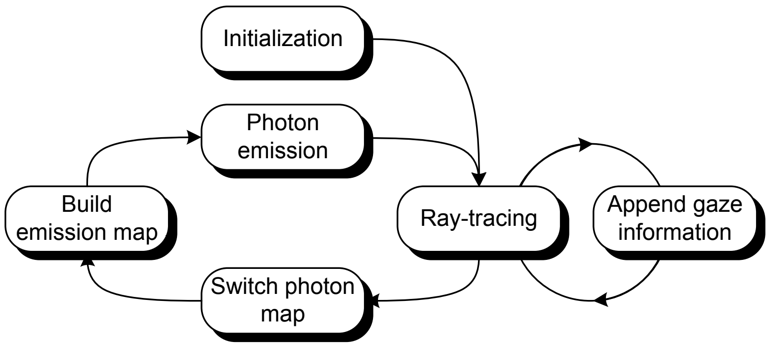


Figure 8.7: The flow of gaze directed photon mapping. The description given in the following sections is based on the flow of one cycle starting with the initialization.

The gaze directed photon mapping system contains a number of parameters that can be controlled, these parameters will in the following description be referred to by symbols, highlighted in *italic*. A complete list of the default values for these symbols will be presented at the end of this chapter, on page 123.

8.4.2 Data Representation

The photons are represented by the data structure shown in Figure 8.8, to discern this photon structure from that of traditional photon mapping, it is referred to as gaze photon.

The collection of booleans and a single 2 bit number are stored in the flag variable, as shown in Table 8.2.

```
typedef struct Photon {
  float pos[3];
  float indir[3];
  float power[3];
  short plane;
} Photon;
```

```
typedef struct GazePhoton {
    float pos[3];
    float indir[3];
    float power[3];
    short plane;
    float outdir[3];
    GazePhoton* parent;
    unsigned char flag;
    unsigned int seed;
} GazePhoton;
```

Figure 8.8: The traditional photon structure shown to the left and the gaze directed photon structure shown to the right. The structures are generally not sought optimized in terms of size, except for flag which is a collection of, among other things, booleans. The size of the Photon structure is on a x86-32bit architecture 38 bytes, while the GazePhoton takes up 59 bytes.

Table 8.2: Flags used to make photon mapping gaze directed.

Bit:	7	6	5	4	3	2	1	0
Value:	-	-	Light source id		Reserved for photon power scaling	Child photon is in gaze region	Part of path to gaze region	Located in gaze region

The first flag bit is set during ray-tracing, the second and third during construction of the photon emission map. The fourth and fifth flag bit is set when a photon is stored together with the two bit id number representing the light source.

When gaze photons are added to the photon map they are stored in an array, called photons. The GazePhoton `parent` pointer points to other photons of this array, see Figure 8.9. Because of that the kd-tree balancing can not move the actual gaze photon data because it would require all `parent` pointers to be updated. To avoid that, the kd-tree is built as a tree of pointers to the actual photon data, these pointers can then be balanced to ensure fast lookup in the photon map. A problem with this lookup scheme is the risk of additional cache misses due to scattered access to the memory, photons locate close to each other is not necessary located close in memory, because they are emitted at random.

8.4.3 Photon Map Initialization

The implementation uses two photon maps, an active map where photons in the current frame are being stored and an inactive photon map from the rendering of the previous frame. The inactive photon map is used to acquire information about how to trace photons in the photon map that is currently being built, as well as other statistics.

The precondition for initializing the photon maps is that the maximum number of photons, *maxPhotons*,

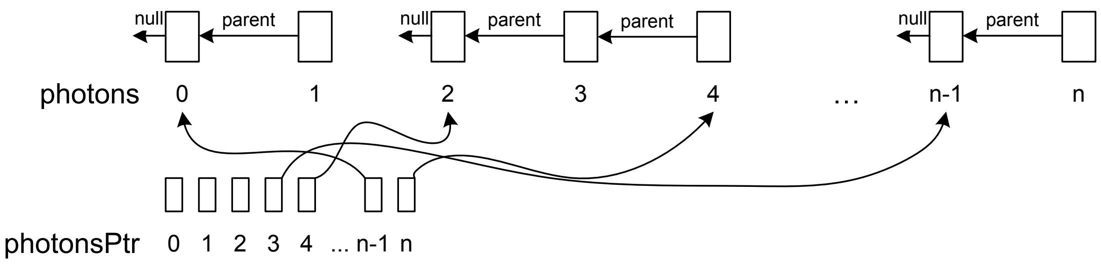


Figure 8.9: The GazePhotons are stored in the photons array, in consecutive order with the oldest GazePhoton stored first. The parent pointers which point to other photons of this array are illustrated by arrows. The array of pointers, photonsPtr, points after balancing to the GazePhotons of photons in what might seem like a random ordering. The symbol *n* corresponds to *maxPhotons*.

is known. Both maps are allocated memory to store *maxPhotons* photons.

The active photon map is at the initialization populated with photons from a traditional photon emitter. For a description of a traditional photon emission, visit Section 6.2.1 on page 58.

It is only possible to increase or reduce the maximum number of photons at run time by deallocating the memory assigned for the two current maps, and create the photon maps anew.

8.4.4 Ray-tracing

Although ray-tracing is the second step, it will be described first, because it is responsible for appending gaze information to the photon map. I.e. initially it adds gaze information to the photon map created with the traditional photon emitter.

Ray-tracing is the process of deciding the shading of every pixel by the summing the radiances from the direct, specular, caustics, and indirect terms of the rendering equation. The direct and specular contributions are calculated by the ray-tracer, and the caustics and indirect contributions are calculated by the photon map. This is done by finding the *nNN* (number of Nearest Neightbor) nearest photons located within a radius of a maximum *NNRadius* meter of the point of ray intersection. The total power of these photons is computed and divided by the size of the area from which they were located to get the irradiance, which multiplied by the BRDF of the material gives the radiance. When pixels inside the gaze area are shaded this photon map lookup will label each photon used to find the total irradiance, such that it later becomes possible to find the photons that contributed to the shading of the gaze area.

Photons are labeled by setting the least significant bit in the flag byte, see Table 8.2.

Switch Photon Map

At every new frame the following will happen.

The active photon map is saved in the inactive map (by a pointer switch). The inactive map becomes active, and all internal variables are reset, before photons are emitted and stored in the map. The memory allocated with the photon maps is not deallocated at frame switches in order to avoid allocation/deallocation at run time.

When the active map has been prepared photons are emitted.

8.4.5 Construction of Photon Emission Maps

Before photons can be emitted a so called emission map for each light source are constructed from the inactive photon map of the previous frame. The only valid emission directions which can be added to the map are those that in the previous frame ensured photons were stored in the gaze region.

The procedure for finding these directions and building the emission map are as follows.

All photons that were used in the shading of the gaze area in the previous frame had their first `flag` bit set during the ray-tracing step. Since it is unknown whether these photons found their way to the gaze region via scattering from other surfaces or directly from the light source an additional test is performed.

This test checks the `parent` field of the photon structure. If this field is equal to `NULL` then the photon has just been emitted from the light source and has not interacted with any surfaces. If the `parent` field is different from `NULL`, then the parent photon that it points to is investigated, this continues until a photon where the `parent` field is `NULL` is found. All photons visited during this process have their second `flag` bit set, to indicate that these photons constitute valid paths to the gaze region. The first parent visited in this process also has its third `flag` bit set, to indicate that its child is located in the gaze region. The `outdir` fields are also filled with `indir` of the child nodes during the process.

It is the `indir` of all photons found in this way with a `parent` field equal to `NULL` that constitutes the emission directions stored in the photon emission map. The test also ensures that the `indir` from a photon with `parent` field `NULL` is not added to the emission map twice, something which could happen if the path of the photon involves two or more surface interactions within the gaze region. This is explained in detail by Algorithm 8.5.

Note that Algorithm 8.5 fulfills the same task as Algorithm 7.1 on page 81 but in a different way. Instead of building the emission map during the ray-tracing by a recursive algorithm, it is built off-line after the ray-tracing is done. The benefit of an off-line algorithm instead of the recursive algorithm is that each photon only has to be processed once, which is faster.

The emission map can either be stored as a latitude longitude map or as an array of vectors representing the emission directions. The later approach is used here in order to avoid conversions from vectors to spherical coordinates.

8.4.6 Photon Emission

Generally the system supports two types of emissions.

Algorithm 8.5 Algorithm used to construct a photon emission map for next photon emission step.

```
INPUT:
    GazePhotonMap
OUTPUT:
    Vector of OutDirs

void get_out_dirs()

1. $j \leftarrow 0$

2. for $i \leftarrow stored\_photons$ to $i = 1$        //reverse iteration, because newest photons are stored last

    • $t \leftarrow 0$
    • GazePhoton $p \leftarrow photons[i]$
    • if ($p.flag$ is in gaze region AND $p.flag = lightId$)
      ◦ while ($p.parent \neq$ null)
          $p.parent.outdir \leftarrow p.indir$
          $p \leftarrow p.parent$
          $p.flag$ sets to part of path to gaze region
          $t \leftarrow t + 1$
      ◦ end while
      ◦ $i \leftarrow i - t$        //skip the other photons of this trace path
      ◦ if $p.flag$ is part of path to gaze region
          · $Outdirs[j].emissionDir = p.indir$
          · $Outdirs[j].seed \leftarrow p.seed$
          · $j \leftarrow j + 1$
      ◦ end if
    • end if

    end for
```

- Traditional photon emission using a naive emitter, see Algorithm 6.1 on page 59.
- Gaze directed photon emission using a photon emission map.

The final systems utilize both emission types to achieve the maximum visual quality of the gaze area. The necessity of the traditional photon emitter is two-fold. Firstly, it ensures that there will be some evenly distributed photons in the scene, such that the gaze region still contains photons after a fast repositioning. This is important because without photons in a region it is impossible to establish a photon emission map. Secondly having traditionally emitted photons in the photon maps, enables the system to correctly scale the power of photons inside the gaze region, see Section 8.4.8 on page 123 for more information.

Different strategies based on the two types of photon emission have been tested and tried, see Appendix D on page 171 for a review. The strategy that was found to be most promising is explained in details here.

The strategy is paired with the principle of frame coherent random numbers for photon tracing [Christensen et al., 2001].

Frame Coherent Random Numbers

Frame coherent random numbers, is a principle where each photon is assigned a unique seed number, such that all random numbers generated during the tracing of that photon is based on that seed. This ensures that the same photon in consecutive frames will follow the same trace path, unless something in the scene changes along its path. The benefit is that the amount of variation over consecutive frames is reduced. A high variation would in real-time photon mapping be visual as fluctuation of the photons because they are stored in different positions for each new frame, unless enough photons are used. But this would also contradict with the purpose of the algorithm which was to emit less photons but then guide them to the gaze region.

In order to implement frame coherent random numbers, the system distinguishes between three types of photons, these are: old gaze directed photons, new gaze directed photons, and traditionally emitted photons. Old gaze directed photons are photons which in the previous frame reached the gaze region, and were emitted as old gaze directed photons or new gaze directed photons. New gaze directed photons are photons which in the previous frame reached the gaze region, but were emitted as traditionally emitted photons. Traditionally emitted photons are as the name indicates simply photons emitted with the naive emitter.

Photons denoted old gaze directed photons have a seed number, which all random numbers during the tracing are generated from. This seed number guaranteed that the photon in the previous frame reached the gaze region. The seed number is stored in the photon emission map together with the emission direction.

The new gaze directed photons, do also have seed numbers, that in the previously frame ensured that they reached the gaze region, the reason why they need to be distinguished from the old gaze directed photons are for the photon power scaling procedure, which is described later.

Traditionally emitted photons get a new seed number every frame.

Emission of photons happens from three possible bins, see Figure 8.10. First old and new gaze directed

Old and new gaze directed photons	Fill	Traditionally emitted photons
$emissionRatio$		$1 - emissionRatio$

Figure 8.10: Distribution of photon emission types, the left side represents emission types obtained from the photon emission map. The size of the old and new gaze photons bin together with the fill bin is defined by the *emissionRatio* variable.

photons are emitted according to the emission direction/seed pairs given by the photon emission map. If the emission map contains more emission direction/seed pairs than needed then oldest will be left out. In case the emission map contains too few emission direction/seed pairs, then the remaining emissions until the *emissionRatio* is obtain will be filled with photons from the Fill bin. These photon are assigned an emissions direction randomly selected from the emission map, added with noise and given a new seed number. They are still emitted as old gaze directed photons although it cannot be assured that they will end in the gaze region. Traditionally emitted photons are emitted in a randomly chosen direction with a random seed number.

8.4.7 Russian Roulette

The implementation of Russian roulette used for gaze directed photon mapping is similar to the one used in traditional photon mapping, see Section 6.2.2 on page 61. The only modification made is that the implementation of frame coherent random numbers also applies to the random numbers that decide the outcome of the Russian roulette.

Photon Scattering

The following applies only to new and old gaze directed photons unless noted otherwise. Whether the described photon scattering principle is applied to traditionally emitted photons is enabled/disabled by the boolean *gazeDirectedReflections*.

In order to ensure that scattered photons end up in the gaze region, these are also guided by information acquired from the inactive photon map. This does not apply to specular reflected photons, because their reflection direction does not involve any randomness hence they can be calculated again.

Whenever a diffuse surface scattering happens, the inactive photon map is consulted to find the scattering direction. This is done by finding the nearest photon within a radius of $NNReflectRadius$ which in the previously frame was part of a path to the gaze region, indicated by having the second `flag` bit set.

This search has three possible outcomes:

1. A photon that fulfills the requirements is found.
2. A photon that fulfills the requirements is found and additionally it also has the third `flag` set, see Table 8.2 on page 116.
3. No photon is found within the search radius.

Case 1: If a photon that fulfills the requirements can be found its `outdir` is used as reflection direction and the photon is traced in that direction from the intersection point.

Case 2: When a photon that also has the third `flag` bit set is found, then the photon is traced in the `outdir` direction, but with noise added such that when the photon intersects with the gaze region it is distributed differently than the previous photons from the same trace path.

Case 3: If no photon can be located, the scattering is using traditional photon scattering, where the reflection direction is found randomly from a cosine weighted distribution around a normal vector. For traditionally emitted photons this is the default behavior unless *gazeDirectedReflections* is enabled.

In all three cases the tracing of photons continue until a predefined max trace depth, $pmTraceDepth$, is met or when the photon intersects with the gaze region.

Photon Absorption

When the outcome of the Russian roulette is absorption of a photon the process will be identical to traditional photon mapping.

Addition of Noise

The introduction of noise to the `outdir` direction is necessary to get photons distributed randomly in the gaze region. The amount of the noise is an important parameter, because there is a risk that photons will be located outside the gaze region if the noise is too dominant and too little noise will not give the desired effect of randomly distributed photons.

The noise is introduced by adding a noise vector to the normalized `outdir`. The noise vector **n** is given as

$$\mathbf{n} = (\varepsilon_x\ \varepsilon_y\ \varepsilon_z) \cdot noiseScale \tag{8.8}$$

where ε is randomly picked from the interval $\varepsilon \in [-1, 1]$. The factor $noiseScale$ is used to control the amount of noise, with e.g. $noiseScale = 0.1$ the noise will at maximum correspond to $5.7°$. The final reflection vector $\mathbf{r}_{new}$ is calculated as

$$\mathbf{r}_{new} = ||\mathbf{r}_{old} + \mathbf{n}|| \tag{8.9}$$

8.4.8 Photon Power Scaling

The implementation used for photon power scaling is identical to the Photon Map Comparison method outlined in Section 7.7 on page 92, but without using two photon maps. The method is based on calculating a ratio between old and new gaze directed photons. From this ratio the difference between the photon power of the gaze region in traditional photon mapping and gaze directed photon mapping can be found.

The number of new gaze directed photons, *nNGDP*, is counted during the construction of the emission map. This can be done because all traditionally emitted photons have their fourth `flag` bit set at emission, see Table 8.2. Likewise is the total number of gaze photons *nGP* counted, with these numbers and the *emissionRatio* it is possible to find the photon power scaling factor by Equation (7.25) on page 93 as

$$s = \frac{nNGDP \cdot (1 - emissionRatio)}{(nGP - nNGDP) \cdot emissionRatio} \tag{8.10}$$

For *emissionRatio* values close to one this method will fail, because there will be zero new gaze photons, and thus the scaling factor will be zero.

8.4.9 Default Parameters

This section summarizes the parameters of the gaze directed photon mapping subsystems and lists their default values in Table 8.3.

Table 8.3: Default system parameters.

Parameter	Default Value	Description
nNN	240	The number of nearest neighbor photons explain how many photons that at maximum should be used to compute the indirect shading.
NNRadius	0.3	The nearest neighbor radius defines the radius of the sphere in which to locate *nNN* photons for indirect shading.
emissionRatio	0.5	The emission ratio defines the ratio between photons emitted according to the emission map, and photons emitted traditionally. An *emissionRatio* of 0.75 means that 75% of the photons are emitted according to the emission map.
gazeDirectedReflections	Yes	Determines whether photons reflected diffusely should be reflected towards the gaze region or just randomly.
NNReflectRadius	0.01	Size of nearest neighborhood search radius used to find the reflection direction for gaze directed diffuse reflections.
noiseScale	0.1	Scales the amount of noise to add, see Section 8.4.7.
maxPhotons	40,000	The maximum number of photons to store in the photon map.
pmTraceDepth	3	The maximum number of intersections to trace photons for.

8.5 Summary

The implementation sections describe how the chosen algorithm designs, which were gaze directed hybrid rendering and gaze directed photon mapping, have been combined and implemented into one coherent system.

First a set of guidelines obeyed during the implementations and some limitations were outlined. Afterwards an overview was given of the entire system from the class diagrams. This overview briefly described how the system is divided into four subsystems, `Tools`, `Scene Objects`, `Display`, and `Photon Mapping` and two separate non-subsystem classes `Camera` and `Scene`. The two subsystems that contribute most to the research field of perceptually adaptive graphics, and in particular perceptually adaptive photon mapping, are the `Display` and `Photon Mapping` subsystem.

These two subsystems are in two separate sections described in such details that a replicate implementation is possible.

The description of the `Display` subsystem is focused on the implementation of gaze directed hybrid rendering, which is capable of combining a global and a local illumination rendering, created with respectively gaze directed photon mapping and OpenGL. The global illumination rendering is via a custom shader merged into the local illumination rendering at the gaze point provided by the eye tracker.

The other important subsystem, `Photon Mapping`, deals with the implementation details of making traditional photon mapping gaze directed. This involves defining a new photon structure and outlining a new photon emission strategy, which relies on the concept of photon emission maps. Since gaze directed photon mapping, have a multitude of controllable parameters, an overview and description of these is presented.

CHAPTER 9

SYSTEM TEST

In this chapter the following system tests will be presented, in order to provide results used to evaluate the system.

- **Qualitative Visual Tests**
  The qualitative tests are conducted to investigate whether gaze directed hybrid rendering with gaze directed photon mapping, is improving the perceived visual quality. Two tests are conducted one with a static image, and another with a video.
  - Static Visual Quality
  - Real-Time Visual Quality
- **Quantitative Visual Test**
  The quantitative test is designed to give a more technical insight into whether gaze directed photon mapping is an improvement to traditional photon mapping or not. This insight is given by comparing the density of photons in the gaze region for the two algorithms
- **System Performance Test**
  The system performance test is a set of measurements of the frame rate improvement of the gaze directed photon mapping system, compared to traditional photon mapping.

The test scenes used throughout this chapter is described next.

Test Scene

The test scenes used for the tests are simple. Simplicity is chosen to provide good frame rates and enable the test subjects to easily distinguish differences between two renderings, and then judge which they find most visually appealing.

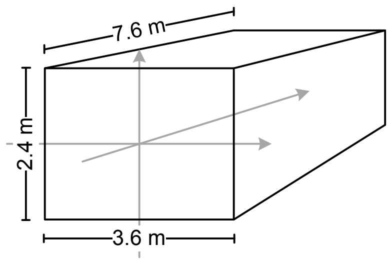


Figure 9.1: The room used for the tests, the dimension of the virtual room is based on the size of a real room.

All scenes uses a room constructed by 12 triangles with the dimensions shown in Figure 9.1. The only other object in the scene used in the static visual quality, quantitative, and performance tests is a sphere, see e.g. Figure 9.2. The objects used in the real-time visual quality test scene, are a high polygon count torso consisting of 1396 polygons [Chuang, 2003], and two rectangles representing a yellow box, see Figure 9.4.

The sphere has a diffuse reflection coefficient of 1.0 and an albedo of 1.0. The surfaces of the torso are all Lambertian with albedos of 0.8. The walls are also Lambertian surfaces with albedos of 0.5.

The camera is statically located in the middle of the scene looking at the objects. The light source is a 200 W point light placed 1.7 m above the floor. In the real-time visual quality test the light source is encircling above the objects.

9.1 Static Visual Quality

The first test is a qualitative test where the test subjects were asked to determine which of two images, rendered with gaze directed photon mapping and without, with settings that yield identical render times, they find to have the best visual quality. The purpose of this is to evaluate whether the visual quality have improved with gaze directed photon mapping or not.

Obviously such a test is unable to test the interactive and gaze directed aspect of the rendering, but this is intended because the test subjects then have fewer distractions to influence their decision.

9.1.1 Setup

The two images should be rendered with the gaze area positioned identically. The gaze area should, as earlier determined, be spanned by a visual angle of 20 degrees and both images should be rendered with a set of parameters that yields a similar frame rate. The parameter that is adjusted to achieve this is the number of photons, thus traditional photon mapping is rendered with 60,000 photons while gaze directed photons mapping is rendered with 40,000. See Table 9.2 for a complete list of parameters, based on the default parameters from Chapter 8.

The images presented to the test subjects are cropped versions of the hybrid renderings, where only the gaze area is visible.

The final verdict is a ranking of the two images based on the visual quality.

The specific test settings are listed in Table 9.1.

Table 9.1: The settings used for the test of static visual quality.

Parameter	Value
Resolution	1650×1080 pixels
Gaze area size	20° with a head distance of 600 mm = a radius of 386 pixels
Gaze point	825, 540
Scene file	SceneTestStatic.scene, see attached DVD.

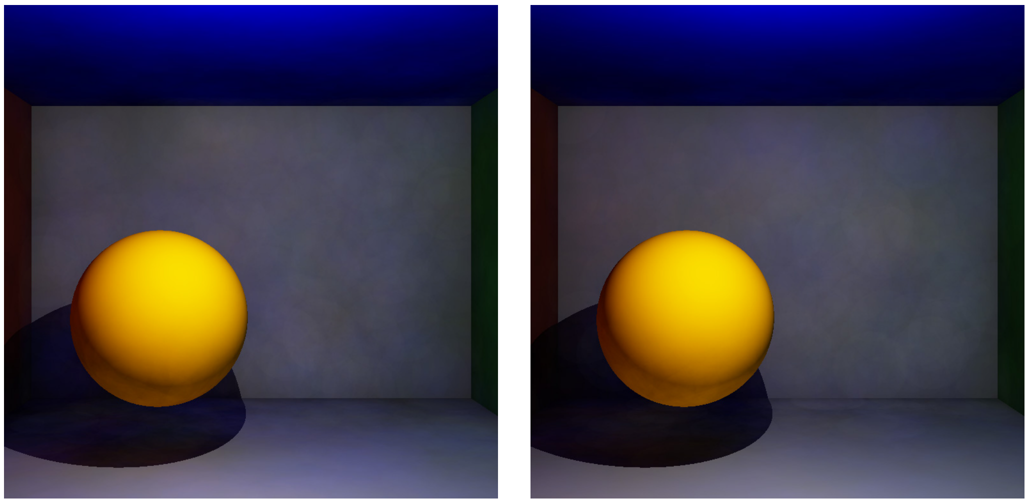
Table 9.2: Photon mapping parameters.

	nNN	$NNRadius$	$pmTraceDepth$	$maxPhotons$	$emissionRatio$	$gazeDirectedReflections$	$NNReflectionRadius$	$noiseScale$
Traditional Photon Mapping	240	0.3	3	60,000	NA	NA	NA	NA
Gaze Directed Photon Mapping	240	0.3	3	40,000	0.5	Yes	0.01	0.0

The test is based upon the images in Figure 9.2. The curious reader can for comparison see a rendering without photon mapping and a gaze directed photon mapping rendering with 60,000 photons in Figure 9.3.

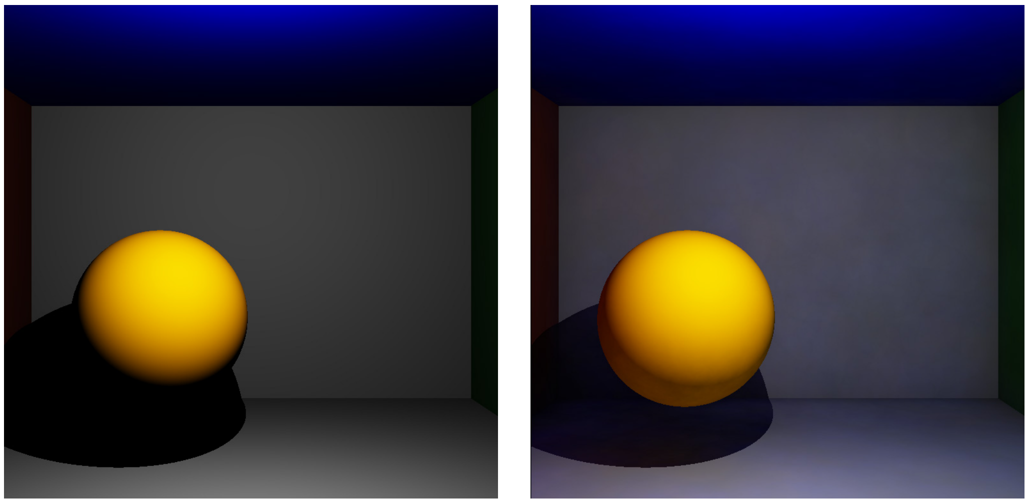
9.1.2 Results

The results of the test subjects ranking of the two images are presented in Table 9.3. A total of 35 test subjects participated in the test. The answers for each individual test subject can be found in Appendix E on page 175.



(a) Traditional photon mapping with 60,000 photons. (b) Gaze directed photon mapping with 40,000 photons.

Figure 9.2: Images used to test the perceived static quality of traditional photon mapping versus gaze directed photon mapping. The time used to render the images is approximately identical.



(a) Reference image rendered without photon mapping. (b) Reference image rendered with gaze directed photon mapping and 60,000 photons.

Figure 9.3: Reference images.

The test subjects were also asked to rank their level of computer graphics knowledge from one to three, with three being expert. Table 9.4 presents their selection grouped by knowledge.

Table 9.3: Results for the static visual quality test. ”Same” represents the percentage of test subjects that found the two images presented in Figure 9.2 to be identical, while ”traditional” and ”gaze directed” respectively represents the percentage of test subjects that preferred the image rendered with traditional photon mapping or gaze directed photon mapping.

	Same	Traditional	Gaze Directed
Percent [%]	17	60	23

Table 9.4: Static visual quality ranking in terms of computer graphics knowledge. Two test subjects had a computer graphics knowledge of level one, 18 had a knowledge level of two, while 17 had a knowledge of level three.

	Knowledge 1	Knowledge 2	Knowledge 3
Same	50 %	16.67 %	13.33 %
Traditional	0 %	66.67 %	60 %
Gaze Directed	50 %	16.67 %	26.67 %

9.1.3 Discussion

The answers show a clear tendency, 60 % of the test subjects find the image rendered with the traditional photon mapping to have the highest visual quality. Thus it can be concluded that the gaze directed photon mapping cannot pass a perceptual test when the criteria is to look identical or better than traditional photon mapping with the conditions used here. But worth remarking is that the algorithm was not designed with generation of static images in mind. Other interesting conditions could be using more or less photons. Less photons would make the photons more visible, while using additional photons would diminish the visual difference between the two images.

Several test subjects stated their reason for selecting the image generated with traditional photon mapping to be due to a more lit shadow. This difference was afterwards investigated, and found to be due to a quirk in the calculation of the scaling factor in the implementation. This would result in a slightly overestimated scaling factor, which in turn would make the gaze directed rendering appear darker than the traditional rendering.

9.2 Real-Time Visual Quality

The purpose of the real-time visual quality test is similar to the static visual quality test. The test subjects were asked to qualitatively rate two videos such that it can be determined whether gaze directed photon mapping improve the perceived quality compared to a video rendered with traditional photon mapping.

9.2.1 Setup

The current implementation is unable to rendered at real-time or interactive frame rates, for gaze areas of the desired size and with slightly complex scenes. To avoid this limitation, a sequence of images for both gaze directed and traditional photon mapping have been rendered and stored. These image sequences are merged into two videos that can be played at 25 fps. This approach removes the possibility to take advantage of real-time gaze information. The solution to this, is to insert a dot at the center of the gaze area, and ask the test subject to focus on it, like it was done in the Visual Perception Test, see Section 4.3 on page 40. The gaze area centered at the dot is moving in a predefined circle, to add real-time dynamics to the video.

The test settings for the real-time visual quality test are listed in Table 9.5

Table 9.5: The settings used for the real-time visual quality test.

Parameter	Value
Resolution	1280×720 pixels
Gaze area size	20° with a head distance of 600 mm = gaze radius of 257 pixels
Gaze point	Rotating in a circle with a radius of 150 pixels around point 640, 360. Completing three revolutions in 30 seconds.
Scene file	SceneTestRealTime.scene, see attached DVD.
Video length	30 sec
Video frame rate	25 fps

The photon mapping settings for the two videos are identical to those used for the static visual quality test, see Table 9.2. An image from the gaze directed photon mapping video can be seen in Figure 9.4, the videos are also located on the accompanying DVD.

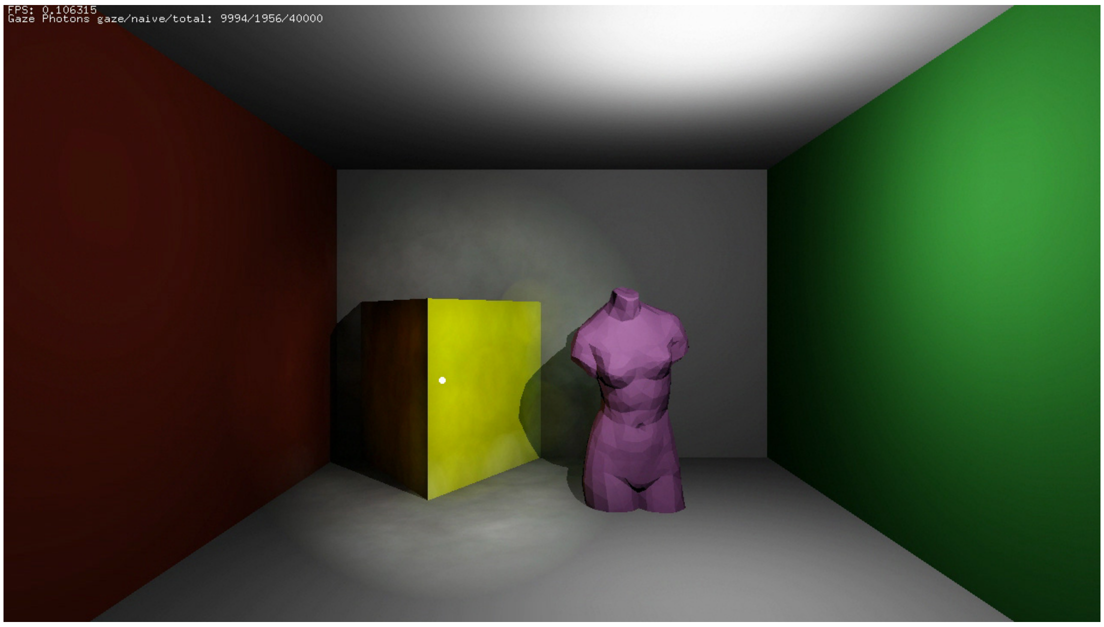
9.2.2 Results

The votes for the videos are shown in Table 9.6.

Table 9.6: Results for the real-time visual quality test. ”Same” represents users which found the two videos to have identical visual quality, while ”traditional” and ”gaze directed” respectively represents users that preferred the video rendered with traditional photon mapping or gaze directed photon mapping.

	Same	Traditional	Gaze Directed
Percent [%]	11.43	11.43	77.14

The votes in terms of computer graphics knowledge are listed in Table 9.7.



```
FPS: 0.106315
Gaze Photons gaze/naive/total: 9994/1956/40000
```

Figure 9.4: Screen dump taken from the gaze directed real-time visual quality video.

Table 9.7: Video rankings in terms of computer graphics knowledge.

	Knowledge 1	Knowledge 2	Knowledge 3
Traditional	0 %	11.11 %	13.33 %
Same	0 %	16.67 %	6.67 %
Gaze Directed	100.00 %	72.22 %	80.00 %

9.2.3 Discussion

Contrary to the static visual quality test, the real-time test shows a clear tendency that the test subjects prefer the gaze directed rendering over traditional photon mapping, even with 50 % fewer photons. A total of 77 % of the test subjects agreed that the video rendered with gaze directed photon mapping had the best visual quality. From the video ranking in terms of computer graphics knowledge, it is also apparent that all types of computer users agreed that the gaze directed photon mapping video is superior to traditional photon mapping.

When the reason for this clear picture is investigated, it seems that the reason for selecting the video rendered with gaze directed photon mapping is the implementation of frame coherent random numbers. The frame coherent random numbers, makes the gaze directed photon mapping video stand out as having much less fluctuation in the gaze area, which better reveals the details of global illumination. More information about frame coherent random numbers can be found in Section 8.4.6 on page 118. A question that can be asked is, why not use traditional photon mapping with frame coherent number and get the same or better results? The disadvantage of this approach is that if few photons reach the gaze area and consequently gives a low quality global illumination, this poor quality will be kept until the scene or gaze point changes. A region that could cause this kind of trouble could be a place illuminated only by indirect illumination. Whereas gaze directed photon mapping will direct many photons towards this region resulting in a higher density of photons, which results in photon mapping of better quality. The next test will clarify how many more photons that will be stored in the gaze area, in different types of scenes, by using gaze directed photon mapping instead of traditional photon mapping.

9.3 Quantitative

The qualitative visual tests proved that the test subjects preferred traditional photon mapping for static images and gaze directed photon mapping for real-time renderings. The purpose of the quantitative test is to investigate whether there is a technical explanation behind this behavior or it is purely based on user perception. This is done by measuring how many photons that actually hit the gaze region, in both traditional and gaze directed photon mapping, with different scenes and at various visual angles.

In this way it is possible to give a more quantitative picture of where gaze directed photon mapping excels, and where it has shortcomings.

9.3.1 Setup

The test settings are listed in Table 9.8.

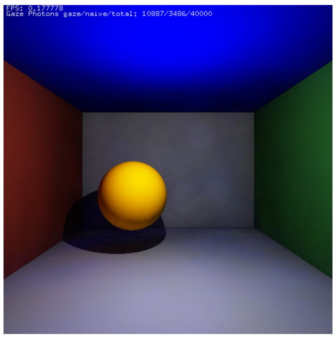
Table 9.8: Test settings used for the quantitative test.

Parameter	Value
Resolution	600×600 pixels
Gaze area size (radius)	5° (53 pixels), 10° (106 pixels), and 20° (214 pixels)
Gaze point	300, 300
Scene files:	Diffuse (QuantitativeTest.scene)
	Specular (QuantitativeTest2.scene)
	Indirect (QuantitativeTest3.scene)

The qualitative test uses the same parameters as the visual quality test, see Table 9.2, with the exception that both traditional and gaze directed photon mapping uses 40,000 photons.

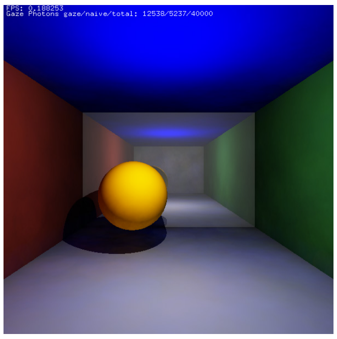
The test is based on measuring the number of gaze photons in both traditional and gaze directed photon mapping over a period of ten frames with a static gaze position. From these measurements the increase in number of gaze photons between gaze directed and traditional photon mapping is calculated.

The scenes used are chosen to excite different aspects of the system. Tests will be executed using a scene with specular objects, a scene dominated by diffuse objects, and a scene dominated by indirect illumination. The three scenes are show in Figure 9.5.



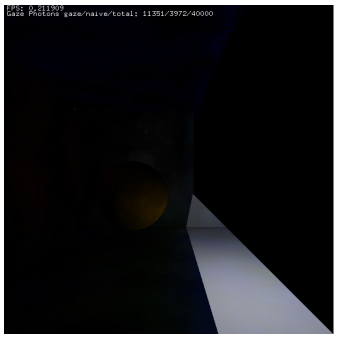
```
FPS: 0.177778
Gaze Photons gaze/naive/total: 10887/3486/40000
```

(a) Diffuse scene.



```
FPS: 0.188253
Gaze Photons gaze/naive/total: 12538/5237/40000
```

(b) Specular scene. The rear wall has a specular reflectance coefficient of 0.8.



```
FPS: 0.211909
Gaze Photons gaze/naive/total: 11351/3972/40000
```

(c) Indirectly lit scene. An extra wall is dividing the room into two sections, with the light positioned in the section to the right, while the camera is in the section to the left.

Figure 9.5: Renderings created with full screen gaze directed photon mapping of the three test scenes.

9.3.2 Results

The measured number of gaze photons for traditional and gaze directed photon mapping for different visual angels and test scenes are listed in Table 9.9.

Table 9.9: Accumulated number of gaze photons over 10 frames, at varying visual angle for both traditional and gaze directed photon mapping.

(a) A diffuse scene.

Visual Angle	Traditional	Gaze Directed
5°	5,322	24,073
10°	11,884	45,501
20°	49,878	70,707

(b) A specular scene.

Visual Angle	Traditional	Gaze Directed
5°	16,056	49,442
10°	98,374	90,008
20°	127,863	96,921

(c) A indirectly lit scene.

Visual Angle	Traditional	Gaze Directed
5°	3,331	22,824
10°	15,893	55,958
20°	38,978	72,056

The measurements shown in Table 9.9 are presented as a percent-wise increase in number of gaze photons by using gaze directed photon mapping compared to traditional photon mapping in Table 9.10.

Table 9.10: The percentage-wise increase in number of gaze photons by using gaze directed photon mapping compared to traditional photon mapping.

	Scene		
Visual Angle	Diffuse	Specular	Indirect
5°	352 %	208 %	585 %
10°	283 %	-9 %	252 %
20°	42 %	-24 %	85 %

Figure 9.6 shows how the number of gaze photons increase as the number of total photons increase.

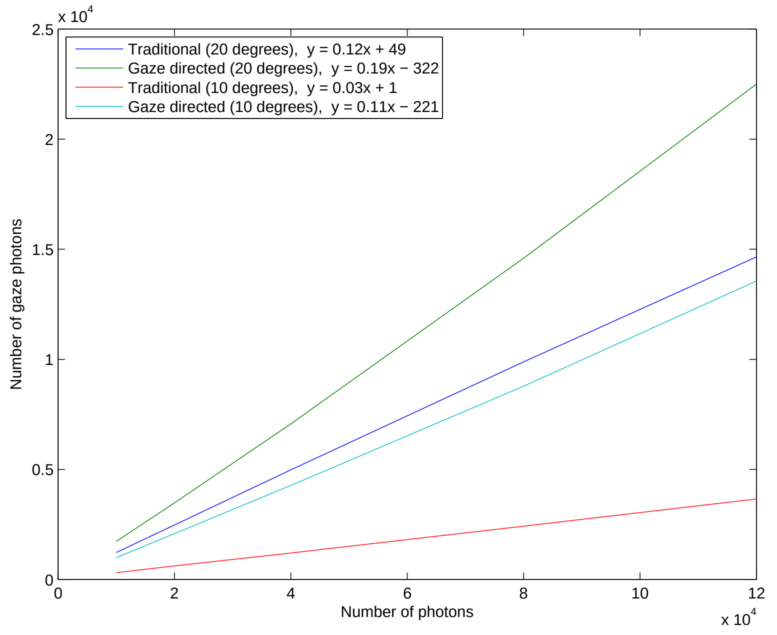


Figure 9.6: Average number of gaze photons as a function of the total number of photons emitted. Both graphs are produced with the same parameters, the diffuse scene, and a photon trace depth of three.

9.3.3 Discussion

From the values in Table 9.10 it is evident that gaze directed photon mapping in general has a higher photon density in the gaze region. The numbers also reveal best and worst case scenarios for gaze directed photon mapping.

Gaze directed photon mapping is from these numbers best suited to work with diffuse scenes, and scenes dominated by indirect illumination. At the proposed working visual angle of 20° the gaze region will contain 42 % more photons for diffuse scenes, and 85 % more photons for indirectly lit scenes when gaze directed photon mapping is used instead of traditional photon mapping.

It is also evident that gaze directed photon mapping has problems coping with highly specular surfaces, in fact 24 % less photons were stored in the gaze region in the test scene with specular surfaces with gaze directed photon mapping compared to traditional photon mapping. The reason why gaze directed photon mapping have no advantage in this scene, is that the specular surface effectively makes the entire scene a part of the gaze region. And with this particular gaze position, more photons are being after one or more iterations stored with traditional photon mapping than with gaze directed photon mapping which gives a negative increase in number of gaze photons. This worst case number is most prominent at a visual angle of 20°. When the size of the gaze area shrinks the problems with specular surfaces diminishes and gaze directed photon mapping is at 5° storing 208 % more photons in the gaze region than traditional photon mapping.

This tendency is valid for all three test scenes. When the size of the gaze area is reduced, then the photon density of the gaze region increases with gaze directed photon mapping compared to traditional photon mapping. From the numbers presented in Table 9.10, it would be advantageous if the visual angle could be reduced to 10° or even less, because then the potential of gaze directed photon mapping would be even higher.

Another factor that varies the density of photons in the gaze region is the emission ratio. The tests are performed with a ratio of 50 %. A higher ratio will guide even more photons to hit the gaze region with the gaze directed photon mapping.

Finally a graph presenting the number of gaze photons as a function of the total number of photons is shown in Figure 9.6. It shows that a higher percentage of the total number of photons will hit the gaze region for gaze directed photon mapping as the total number of photons increase. It also shows that the plots are linear, but with a larger slope coefficient for the gaze directed plot, as it would be expected.

9.4 Performance

The test of the system performance serve two purposes. First and foremost it serves the purpose of validating whether the developed system is a suitable solution to the problem statement in terms of frame rate performance. Secondly it contains information that can be valuable to determine where improvements can be obtained, either by alternative algorithms or different implementation strategies. Thirdly it is used to validate the complexity estimations of the design chapter.

9.4.1 Setup

The performance test is to a measurement of the execution times required to compute each of the many stages of the complete system. The execution time require for each stage will be measured for 100 frames to find an average. The measurements are made with the developed implementation for both gaze directed photon mapping and traditional photon mapping, such that a comparison of the two algorithms in terms of time consumption can be made. The measurements for the traditional photon map will be made with two different numbers of photons. Initially with the same number of photons as used in the gaze directed photon mapping. Then with the number of photons needed to obtain approximately the same number of photons within the gaze region as with gaze directed photon mapping, these test results are denoted ”equivalent photon mapping”.

The measurements are performed on four main stages, ray-tracing, photon mapping, OpenGL, and other. The three first stages are split into sub-stages which are measured individually.

- **Ray-tracing**
  - **Intersections** - calculation of ray-objects intersection, see Appendix B on page 157.
  - **Shadow ray** - shadow ray intersections, see Appendix B on page 157.
  - **Phong** - modified Blinn-Phong shading, see Chapter 5 on page 45.
  - **PM shading** - irradiance calculations from the photon map, see Section 6.3 on page 65.
  - **Other** - e.g. calculations of reflection and refraction directions, see Appendix B on page 157.
- **Photon Mapping**
  - **Initialization**
  - **Photon emission** - traditional or gaze directed, see Section 8.4.6 on page 118.
    - **Outdirs** - construction of emission map for gaze directed photon mapping, see Section 8.4.5 on page 118.
  - **Russian roulette**, see Section 8.4.7 on page 121.
  - **Photon storage**, see Section 8.4.2 on page 115.
  - **Balance** - kd-tree balancing, see Section 6.2.3 on page 62.
- **OpenGL**
  - **Scene generation** - render the scene with local illumination using OpenGL, see Section 8.3.2 on page 107.
  - **Merge** - ray-tracing and OpenGL renderings merged to one texture, see Section 8.3.3 on page 111.
- **Other** - initialization.

The most time demanding stages will be analyzed thoroughly to locate the most time consuming parts, with the purpose of pin pointing where improvements can be made to obtain a faster system.

A test will also be made to determine the performance improvement obtained by using a gaze area of 20 degrees visual angle instead of full screen photon mapping, which would be necessary if no eye tracker was available.

Since some of the code is parallelized, for instance the ray-tracer, a test will be performed to clarify how multiple CPUs affect the speed of the system. That test will be carried out with one, two, and four CPUs.

Specifications

The specifications of the computer used for the tests are listed in Table 9.11. All the tests are, unless noted otherwise, executed with the two cores of the test computer.

Table 9.11: Computer specifications

Operation System	Windows XP SP3
Processor	Intel Core2 6600 @ 3.0 GHz
GPU	NVIDIA GeForce 7900GT
Memory	2048 MB

The tests will be performed on the same scene (scene file SceneTestStatic.scene), but with three different resolutions. A image of the scene can be seen in Figure 9.2 on page 130. The different sizes of the scenes can be seen in Table 9.12

Table 9.12: Resolution and gaze area radius used for the performance tests.

Parameter	Size 1	Size 2	Size 3
Resolution	320×240	640×480	1024×768
Gaze area radius	86 pixels	171 pixels	274 pixels

The parameters used for the photon mapping are identical to the parameters used for the gaze directed photon mapping in the visual quality tests, see Table 9.2 on page 129. The number of photons used in the equivalent photon mapping is 95,000.

9.4.2 Results

Time Consumption

The time consumption of the different stages for the three different resolutions can be seen in the Tables 9.13, 9.14, 9.15.

Figure 9.7 shows the time consumption used at different number of photons, for both traditional and gaze directed photon mapping, with a scene resolution of 640×480.

Full Screen Photon Mapping

Table 9.16 states the times needed to render the entire screen with full screen photon mapping, and how much slower it is compared to gaze directed photon mapping.

Table 9.13: The execution times for a resolution of 320×240 pixels and a gaze radius of 86 pixels. T = traditional photon mapping (40,000 photons), G = gaze directed photon mapping (40,000 photons), E = equivalent photon mapping (95,000 photons).

Stage	Avg. per Frame [ms]			Percentage [%]		
	T	G	E	T	G	E
Ray-tracing	166	264	290	69.49	70.41	60.48
- Intersection	12	13	12	4.82	3.43	2.43
- Shadow ray	12	13	14	4.81	3.44	2.98
- Phong	4	4	2	1.83	1.08	0.51
- PM shading	140	237	270	58.49	63.16	56.41
- Other	-1	-3	-9	-0.46	-0.70	-1.84
Photon Mapping	63	102	181	26.20	27.18	37.88
- Initialization	0	0	0	0.06	0.00	0.00
- Photon emission	8	9	15	3.21	2.43	3.06
- Outdirs	0	4	0	0.00	1.05	0.00
- Russian roulette	32	69	84	13.57	18.40	17.45
- Photon storage	11	11	20	4.64	2.86	4.12
- Balance	11	13	63	4.71	3.49	13.25
OpenGL	8	7	7	3.14	1.99	1.47
- Scene generation	2	2	7	0.98	0.66	1.37
- Merge	5	5	0	2.15	1.33	0.10
Other	3	2	1	1.17	0.41	0.16
Total	239	375	479	63.73	100.00	127.73

Multiple CPUs

The time gain using multiple CPUs is listen in Table 9.17. The computer used for this test has a Intel Core 2 Quad Q9550 @ 2.83 GHz CPU and the GPU is a GeForce 8800GT. Note that the scene used for this test did also use a specular sphere, which is why the ray-tracing stage consumes a little more time.

Table 9.14: The execution times for a resolution of 640×480 pixels and a gaze radius of 171 pixels. T = traditional photon mapping (40,000 photons), G = gaze directed photon mapping (40,000 photons), E = equivalent photon mapping (95,000 photons).

Stage	Avg. per Frame [ms]			Percentage [%]		
	T	G	E	T	G	E
Ray-tracing	610	977	1144	89.40	89.90	85.90
- Intersection	44	45	49	6.49	4.18	3.70
- Shadow ray	38	38	37	5.52	3.50	2.77
- Phong	15	14	16	2.24	1.31	1.18
- PM shading	520	892	1059	76.19	82.07	79.54
- Other	-7	-13	-17	-1.04	-1.16	-1.30
Photon Mapping	62	101	179	9.14	9.28	13.48
- Initialization	0	0	0	0.00	0.00	0.00
- Photon emission	7	10	16	1.08	0.88	1.17
- Outdirs	0	4	0	0.00	0.39	0.00
- Russian roulette	36	69	86	5.24	6.32	6.46
- Photon storage	8	10	19	1.21	0.91	1.46
- Balance	11	13	58	1.60	1.17	4.39
OpenGL	8	7	7	1.12	0.66	0.50
- Scene generation	2	2	2	0.30	0.20	0.18
- Merge	6	5	4	0.83	0.46	0.33
Other	2	2	2	0.34	0.16	0.12
Total	682	1087	1332	62.74	100.00	122.54

9.4.3 Discussion

The results showed that the higher resolution the more execution time is needed to execute one frame, independently on what type of photon mapping that is used. But because the OpenGL stage only uses very little execution time (<4%), it can be concluded that it is not the resolution that introduces the primary increase in execution time, but instead the size of the gaze area that inreases with the resolution,

Table 9.15: The execution times for a resolution of 1024×768 pixels and a gaze radius of 274 pixels. T = traditional photon mapping (40,000 photons), G = gaze directed photon mapping (40,000 photons), E = equivalent photon mapping (95,000 photons).

Stage	Avg. per Frame [ms]			Percentage [%]		
	T	G	E	T	G	E
Ray-tracing	1546	2543	2830	95.45	95.58	93.43
- Intersection	119	117	114	7.34	4.40	3.78
- Shadow ray	99	109	95	6.11	4.08	3.13
- Phong	35	44	43	2.18	1.66	1.42
- PM shading	1321	2318	2619	81.54	87.13	86.47
- Other	-28	-45	-42	-1.71	-1.69	-1.37
Photon Mapping	61	101	185	3.79	3.81	6.10
- Initialization	0	0	0	0.00	0.00	0.00
- Photon emission	6	10	15	0.40	0.37	0.48
- Outdirs	0	3	0	0.00	0.10	0.00
- Russian roulette	31	70	84	1.93	2.62	2.78
- Photon storage	12	10	22	0.74	0.37	0.74
- Balance	12	12	64	0.72	0.45	2.11
OpenGL	10	11	12	0.64	0.43	0.40
- Scene generation	3	2	2	0.15	0.08	0.08
- Merge	8	9	10	0.49	0.35	0.32
Other	2	5	2	0.12	0.18	0.07
Total	1620	2661	3029	60.88	100.00	113.83

since it is inside this area ray-tracing is performed.

As it can be seen in the tables ray-tracing is using more than 95 % of the total time for the largest gaze area. The execution times for the ray-tracing sub-stages are not correct since they are multi-threaded, which gives the negative times in other. But the proportions between them, for the different photon mapping algorithms, are correct. Almost all of the execution time, up to 87 % is used in the sub-stage called PM shading, which is where the irradiance contribution from the photon map is calculated.

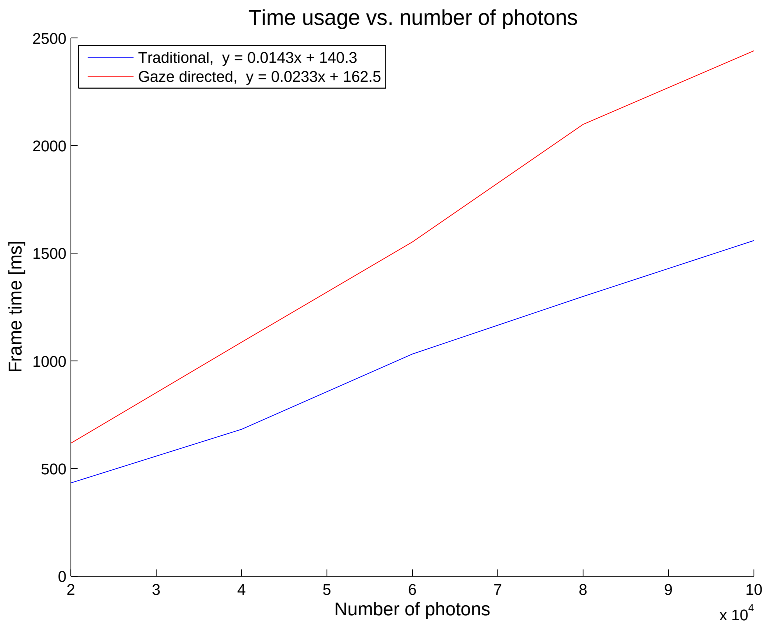


Figure 9.7: The graph shows how the time increases with the number of photons for traditional and gaze directed photon mapping, respectively. The graph is for the scene with the resolution 640×480 and with a gaze radius of 171 pixels.

Table 9.16: The times required to render photon mapping on the entire screen with a resolution of 680×480. The gain is the increase in time needed, in percent, to render the full screen compared to rendering only 20° with gaze directed photon mapping.

Type	Time [ms]	Gain [%]
20 degrees gaze directed	1087	0
Full traditional	2803	158
Full gaze directed	3504	222
Full equivalent	5326	390

More specific, it is the location of the nNN nearest photons that is very time demanding. So it is here optimizations should be made to obtain a faster system.

The execution time used for the photon mapping stage is constant for all three resolutions, which also was expected since resolution and gaze area size should not effect the performance of the photon mapping.

Compared to gaze directed photon mapping the traditional photon mapping with equal number of photons uses approximately 62 % of the time that gaze directed photon mapping uses. A little of the extra time, in the gaze directed photon mapping, is used in the Russian roulette to guide the photons in the direction towards the gaze region. But most of the extra times are used for photon mapping shading.

Table 9.17: The results of multiple CPUs, with gaze directed photon mapping on a scene with a resolution of 1024×768 pixels and a gaze radius of 274 pixels. The test is performed using 1, 2, and 4 CPU's.

Stage	Avg. per Frame [ms]			Percentage [%]		
	1	2	4	1	2	4
Ray-tracing	4219	2736	2141	97.57	96.43	95.38
- Intersection	234	221	151	5.40	7.79	6.75
- Shadow ray	183	171	130	4.24	6.02	5.79
- Phong	57	58	44	1.33	2.05	1.97
- PM shading	3621	2391	1914	83.74	84.28	85.26
- Other	124	-105	-98	2.86	-3.71	-4.38
Photon Mapping	97	95	98	2.25	3.35	4.35
- Initialization	0	0	0	0.00	0.00	0.00
- Photon emission	11	9	12	0.25	0.30	0.53
- Outdirs	4	3	3	0.10	0.10	0.12
- Russian roulette	65	67	65	1.51	2.36	2.90
- Photon storage	8	7	9	0.19	0.25	0.40
- Balance	13	12	12	0.29	0.44	0.52
OpenGL	4	3	5	0.08	0.12	0.21
- Scene generation	0	0	0	0.00	0.01	0.00
- Merge	4	3	5	0.08	0.11	0.21
Other	4	3	1	0.10	0.10	0.06
Total	4324	2838	2245	100.00	65.63	51.92

The reason is that, in the gaze directed photon mapping (many) more photons are located in the region where the nearest photons are searched, and thereby it takes longer to locate the *nNN* nearest photons when using the kd-tree structure.

For the equivalent photon mapping 95,000 photons were used, which gave approximately the same number of photons in the gaze region as the gaze directed photon mapping using 40,000 photons. This increase in number of photons makes the equivalent photon mapping 14 to 28 % slower than gaze directed photon mapping. The stages that consumes the extra time are photon mapping and PM shading, simply because there are more photons to process.

The time consumption as the number of photons increases can be observed to be linear. By regressing the functions it can be found that the time usage increases by 0.0143 ms for each additional photon used in the traditional photon mapping, and 0.0233 ms for the gaze directed. It is reasonable that the functions are linear because the number of nearest neighbors nNN used in the PM shading stage is increased linearly with the maximum number of photons. Then the PM shading stage, that uses almost all the execution time, gets a linear complexity, $nNN \cdot log(maxPhotons)$.

The advantage of using hybrid rendering with gaze directed photon mapping is that a smaller screen area has to be rendered with photon mapping. Without eye-tracking the entire screen must be rendered with photon mapping, if the goal is to achieve the same perceived quality as hybrid rendering with gaze directed photon mapping. The test showed that it would take 2.58 times longer time to render the entire scene with traditional photon mapping and 4.9 times longer with equivalent photon mapping.

The use of multiple CPUs gives a significant improvement in speed, especially from one to two CPUs. The benefit from using multiple CPUs is gained in the ray-tracing stage, which also was the only code that was implemented to use multiple CPUs. By optimizing the system additionally for multi-threading an even higher benefit from multiple CPUs can be obtained.

9.5 Summary

The purpose of the static visual quality test, was to determine whether gaze directed photon mapping gives an image of better quality than a render timewise equivalent traditional photon mapping image. The result was that 60 % of the test subjects preferred the image created by traditional photon mapping.

But the system is not developed for static images, therefore a similar test was made, but with videos of a scene with dynamic lighting. The test subjects were asked to rate two videos, one with gaze directed photon mapping and one with traditional photon mapping. The result was that 77 % preferred the video with the gaze directed photons.

The third test was a quantitative test, with the purpose to clarify how many more photons that are stored in the gaze area when using gaze directed photon mapping compared to traditional photon mapping. The test showed that gaze directed photon mapping is best suited for regions with diffuse material and/or indirect lighting. Up to 85 % more photons are located in the gaze area when a visual angle of 20 degrees is used and even more for smaller visual angles.

The last test was a performance test of the system, where execution times for the different stages were measured for both traditional, gaze directed, and equivalent photon mapping at different resolutions. The execution times showed, that traditional photon mapping only used about 62 % of the time used by gaze directed photon mapping, since fewer photons was located in the gaze region. But gaze directed photon mapping was about 20 % faster than equivalent photon mapping. The function that used most of the time, up to 87 % percent, was the irradiance estimate from the photon map, i.e. it is this function where improvement should be implemented to get a faster system. The time consumption versus number of photons was shown to be linear for both traditional and gaze directed photon mapping. Gaze directed photon mapping reduces the area that is necessary to be rendered with photon mapping, i.e. a lot of time is saved compared to rendering the entire screen with traditional photon mapping. If the entire screen was rendered with photon mapping it would take 2.6 to 4.9 times longer.

CHAPTER 10 DISCUSSION

In the following a conclusion of the project will be presented. It summarizes each individual part of the report and outlines the important findings, before giving a final conclusion.

After the conclusion a presentation of future work within the scope of the project is given. This includes implementation improvements, and areas where more theoretical work could be conducted.

10.1 Conclusion

The hypothesis for the project was that it is possible to improve computer graphics by combining traditional computer graphics algorithms with the gaze position acquired by an eye tracker. This hypothesis was transformed into a problem statement.

The problem statement that formulated the premise for this project was:

How can photon mapping be combined with the gaze position acquired by an eye tracker, in order to improve the rendering performance without perceivable degradation of quality?

10.1.1 Acquisition of Gaze Position

First the eye tracking aspect of the problem statement was covered by testing the provided Tobii X120 eye tracker. This investigation revealed that tracking the gaze position accurately, even with expensive commercial hardware, is not a simple task. The tests of the eye tracker showed that it had fairly good precision, but were lacking some accuracy, something that could be a potential problem when used for perceptually adaptive graphics, because the users might be able to observe the inaccuracies. Accuracy is the ability to track gaze position correctly, while precision is the ability to keep the position constant if it is kept constant by the user.

A solution to the suboptimal accuracy was developed. Tests showed that the accuracy error was more or less consistent for different test subjects, so by combining their calibration results, a general calibration map was built. This map was stored and used when testing new subjects and provided up to 60 % better accuracy compared to not applying the calibration map.

A user test was conducted to determine how much of the screen area, the gaze area, a typical user would be able to attend to while using the computer. It was found that a gaze area with a visual angle of 20 degrees where the maximum area that a user could attend to.

After the eye tracking aspect of the problem statement was solved, the photon mapping aspects were investigated, and some algorithm designs for perceptually adaptive rendering systems with photon mapping were outlined.

Design

Two different perceptually adaptive photon mapping algorithms together with a perceptually adaptive ray-tracing algorithm were developed and investigated on a theoretical level. The two photon mapping algorithms were named gaze directed importon mapping and gaze directed photon mapping. The first algorithm uses the concept of importons, i.e. special photons that marks important areas in the scene. The important area would, in this case, be the gaze region. The second algorithm, does not rely on an extra stage to emit importons, instead, it uses the stored photon map of the previous frame and the appended information from that map about which photons that contributes to the shading of the gaze area. This information is used when building the current photon map, to guide photons to the gaze region.

Both of the proposed algorithms have a complexity worse than traditional photon mapping if used with the same number of photons. But that is negated by the fact that both methods are developed to distribute the photons to the gaze region, such that far less photons in total have to be emitted to achieve the same photon density in the gaze region as with traditional photon mapping. In order to test the claim, gaze directed photon mapping was selected for implementation, because its photon emission strategy was more promising than gaze directed importon mapping.

The use of perceptually adaptive photon mapping, in the form of gaze directed photon mapping was not the only design choice. A feasibility study was conducted to determine whether the test subjects were able to perceive that the outer area was rendered with local illumination while the gaze area was rendered with global illumination. Most test subjects did not notice that the graphics outside a 20 degrees visual angle were without global illumination. As a result it was chosen to use a hybrid rendering approach in the system. This gaze directed hybrid rendering approach with gaze directed photon mapping would render a gaze area of 20 degrees with full global illumination. The outer area on the contrary would only be rendered with local illumination using OpenGL.

A visual angle of 20 degrees constitute to 33 % of the screen area. If the local illumination rendering is considered to be computationally free to render, then using gaze directed hybrid rendering alone would expectedly be three times faster than rendering full screen global illumination. This improvement is according to the feasibility study possible without influencing the perceived quality. Alternatively the saved computation time could be used to further improve the quality of the gaze region to achieve an even better perceived quality.

Implementation

It was chosen to create an implementation mostly from scratch, thus only the most important features of the different subsystems have been implemented. Its two main subsystems are a display subsystem and a photon mapping subsystem, responsible for the hybrid rendering and the gaze directed photon mapping, respectively.

It was decided to create the hybrid rendering in image space, by combining textures of the local and global illumination into one texture that could be shown on the screen. This was done using custom shaders, which among other things used the gaze point from the eye tracker as input to position the global illumination rendering centered at the gaze point.

The implementation of gaze directed photon mapping was developed as additions to traditional photon mapping. I.e. the photon structure was extended, handling of an additional photon map, and construction of an emission map was added. During the development it became evident that in order for the approach to look visually appealing in real-time, fluctuations of the photons in subsequent frames had to be minimized. This was done by implementing frame coherent random numbers.

The system works by first randomly emitting photons into the scene. Ray-tracing is then used to create a global illumination rendering of the gaze area. During this process all photons used by ray-tracing are marked as gaze photons. Before the photon emission stage of the next frame the current photon map is stored, and a photon emission map is constructed. The photon emission map is a map of emission directions that were used by the photons that eventually ended up as being marked as gaze photons. The next time photon emission is initiated, the emission directions can be selected from the photon emission map, and at every photon intersection a lookup is made in the saved photon map to decide the reflection direction, such that the new photons reuse the trace paths from the previous frame. The implementation of frame coherent random numbers ensures that all random numbers used to trace these photons are identical between two frames. Since the photon map would be totally static if all photons were emitted with a direction from the emission map, a percentage of the photons are still emitted traditionally.

The final implementation is a fully functional gaze directed hybrid rendering system, with gaze directed photon mapping.

System Test

Three types of system tests were conducted:

1. Tests to cover the perceived visual quality.
2. Qualitative test of the number of gaze photons.
3. Test on the computation performance.

From the test of the perceived visual quality it was concluded that gaze directed photon mapping for static images did not provide better quality than traditional photon mapping, when the two images had comparable render times. On the other hand when comparing videos with the same settings users clearly preferred the rendering produced by the gaze directed photon mapping. This is primarily attributed to the implementation of frame coherent random numbers.

The quantitative tests concluded that for most scenes the developed system will result in more photons in the gaze area. But it was also evident that for scenes with many specular objects there was no gain in using gaze directed photon mapping, because effectively the specular objects would force photons to be stored in the entire scene.

The conclusion of the performance test is that the developed system could be further optimized. The performance gain of using hybrid rendering with gaze directed photon mapping compared to gaze directed traditional photon in terms of computation time was small. For gaze regions with the same number of photons, the developed gaze directed photon mapping approach was between 14 % and 28 % faster, depending on the resolution. But when the developed system was compared to rendering the entire screen with global illumination, and enforcing that the gaze region should contain the same number of photons for both methods, then the developed gaze directed hybrid rendering with gaze directed photon mapping is approximately five times faster than standard full screen photon mapping.

Final Conclusion

The final conclusion of this project is that the proposed system design with gaze directed hybrid rendering and gaze directed photon mapping is a viable solution to the problem statement. The real-time video test showed that test subjects did prefer the developed algorithm in terms of perceived quality. The analysis of the number of gaze photons, showed most promising results for diffuse and indirectly lit scenes, with between 42 % and 85 % more photons ending up in the gaze region for the developed system at a 20 degrees visual angle, and even more for smaller visual angles.

It can be concluded that the ideas outlined in this project, can be used to achieve higher quality graphics, whether it is via gaze directed photon mapping or another approach at lower computation costs. This is mainly because of gaze directed hybrid rendering, and less because of gaze directed photon mapping. Results showed that the performance improved by up to a factor five by using gaze directed hybrid rendering with gaze directed photon mapping, compared to rendering the scene will full screen photon mapping, with an identical number of photons in the gaze area.

The two main contributions of this project are:

A) The general gaze directed hybrid rendering algorithm.
B) Gaze directed photon mapping as a specialization of the general algorithm.

The gaze directed hybrid rendering algorithm is a contribution that can be used in other contexts, for instance in computer games. Even though the graphics in computer games have become better and better, they are still not using photorealistic global illumination, since it is too computationally heavy.

The gaze area is not forced to be rendered with gaze directed photon mapping. It could be other types of global illumination (e.g. path tracing), or it could be simulations of other visual effects. The visual effects that could be used to get a more visual realistic experience could be glare and bleaching effects or depth of field, as described in the ”Perceptually Adaptive Rendering” chapter.

By using the contributions presented in this project, a smaller area of the screen has to be rendered with global illumination if the gaze point is tracked with an eye tracker. This will allow less powerful computers to make use of real-time global illumination.

10.2 Future Work

The developed system is fully functional with the fundamental functionalities, but improvements can still be made to get a better and faster system.

The performance test showed that the majority of the execution time was spent doing lookups in the photon map to locate the nearest photons to estimate the irradiance. I.e. it is here the first improvements should be made to get a faster system. Some improvements that may be considered are to perform the irradiance calculations on the GPU, which could reduce the computation time greatly. Other improvements could be to improve the data structure and memory usage, or pre-compute the irradiance value which would require the scene to be static.

Something that does not work properly is the rendering of caustics. This would be an important improvement to get global illumination rendering with better photorealism.

The ray-tracing subsystem can also be improved, both regarding the quality and speed. To reduce the execution time of ray-tracing it can be implemented to run on the GPU, something which has been done before in several real-time ray-tracing applications [Günther et al., 2007, Zhou et al., 2008]. Another way to make it faster is to reduce the number of ray-polygon intersections, since it is a time demanding process. This can be done by using scene graphs with an acceleration structure e.g a kd-tree, or by using the concept of ray-bundles where several rays next to each other are traced at the same time to make several intersection calculations at the same time [Wald et al., 2001].

To improve the ray-tracing quality supersampling, where several primary rays are send through each pixel, can be implemented. This will reduce the aliasing effects, but also increase the amount of the computations. Another possible way to improve the quality is by using Monte Carlo ray-tracing together with photon mapping, where a number of secondary rays are emitted from the intersection point to collect irradiance contribution from several directions.

APPENDIX A GLOBAL ILLUMINATION

Global illumination is a broad category of algorithms for making realistic lighting in computer graphics. Global illumination algorithms do not only take direct illumination from the light sources into account but also indirect illumination which is light reflected from other surfaces. Several algorithms exist to make global illumination. The two most common algorithms are ray-tracing that primary calculates the direct illumination contributions, and radiosity that is used for the indirect illumination. All other global illumination algorithms can roughly be described by these two algorithms or a combination of both which often is denoted as a multipass method [Dutre et al., 2002]. The other global illumination algorithms that will be described are path tracing, final gathering, and photon mapping.

Most of these algorithms are very computational heavy and can not run in real-time when rendering the entire screen/image. By only rendering a smaller area of the screen with global illumination, real-time may be feasible. If this smaller area is located around the gaze point of the user, it is a hypothesis that the missing/reduced global illumination in the rest of the scene will not be perceived by the user. A short description of the most common global illumination algorithms that can be suitable to use with an eye tracker will be presented next.

A.1 Radiosity

Radiosity is an algorithm to render indirecte illumination reflected from Lambertian (perfectly diffuse) surfaces. The scene that is to be rendered by radiosity is divided into smaller surfaces, also called patches, where the radiosity is calculated for each patch. The radiosity is a measure for the total rate of energy leaving a surface. Form factors are calculated for all the patches, which are coefficients that describe how much the patches can see each other. Patches with a high coefficient will receive more indirect illumination [Birn, 2000]. These form factors are used in a rendering equation which calculates the radiosity of the patches. Figure A.1 shows an image rendered with only direct illumination and the same image with radiosity added (indirect illumination).

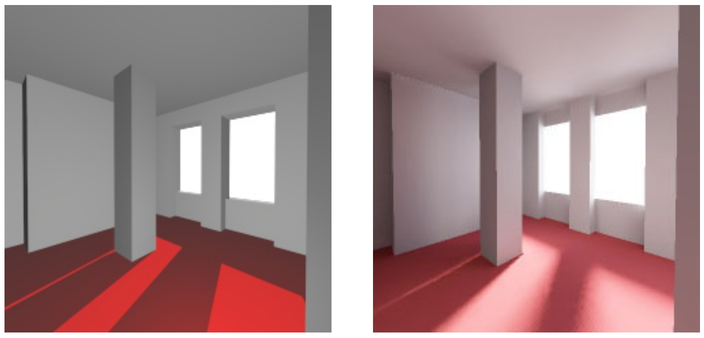


Figure A.1: **a)** A scene rendered with only direct illumination. **b)** The same scene rendered with radiosity added [Tobler, 1997].

Radiosity is a scene-based rendering algorithm, i.e. that the scene can be pre-rendered independently of the view point and view direction unlike image-based algorithms that must be re-rendered when the view point changes [Dutre et al., 2002].

A.2 Ray-tracing

In ray-tracing rays are casted from the viewer/camera and through each pixel in the image plan. If a ray intersects with an object, new rays are casted from the intersection point towards the light sources, called shadow rays. If the shadow ray hits another object before the light source, the intersected point will be shadowed from the light source. If it hits the light source the properties of the object and light source is used to calculate the radiance of the pixel. Additionally a reflection ray is casted from the intersection point. If it intersects with another object, light reflection from that object contributes to the radiance of the pixel, that the ray is sent through [Owen, 1999]. Figure A.2(a) on page 156 shows an image rendered with ray-tracing.

This type of ray-tracing where the rays are traced from the camera and to the light source is called backwards ray-tracing, whereas forward ray-tracing traces rays from the light and to the camera. By using backwards ray-tracing only the visible rays are calculated, while for forward ray-tracing infinity many rays can be calculated but only few would reach the camera [Dutre et al., 2002]. For a more detailed description of ray-tracing see Appendix B on page 157.

A.3 Path Tracing

Path tracing is an extended version of ray-tracing that often generates more realistic images, but is also much more computation heavy, because a very high number of rays is needed to prevent noise in the image. In path tracing each ray is recursively traced along a path until a light source is reached or a maximum number of bounces is completed [Tuttle, 2003].

When a ray hits a surface the ray is, from the material properties of the surface and a random factor, either reflected or refracted. If it is reflected the ray is bounced randomly in a new direction according to a Probability Density Function (PDF). This algorithm is also called Monte Carlo [Hanrahan, 2001]. When a light source is reached the light contribution along the path is calculated for the pixel that the first ray was shot through. Figure A.2(c) shows a path traced rendered image.

A.4 Final Gathering

Final gathering is used for indirect illumination between two surfaces where light from one surface reflects on the other surface. From the intersection points several rays are emitted in different directions to find other nearby surfaces, where the radiance may have an influence on the illumination at the intersection point.

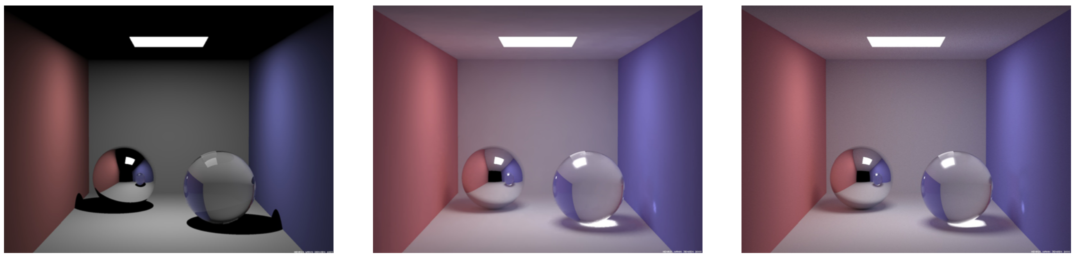
Final gathering is often used in conjunction with photon mapping, to smooth the result of photon mapping [Birn, 2000].

A.5 Photon Mapping

Photon mapping is a multipass method for global illumination, that is developed by the Dane, Henrik Wann Jensen [Waters, 2007]. Photon mapping can be divided into two passes. The first pass calculates the effects of indirect illumination and caustics (light patterns created by specular reflection or refraction) by creating photon maps, and the second pass is the actual rendering of the image [Waters, 2007].

In the first pass photons (packets of light energy) are emitted into the scene from the light sources. The photons are traced through the scene similar to path tracing. When a photon hits a diffuse surface it is stored in the photon map, and based on the material properties of the surface and a technique called Russian Roulette it is decided whether the photon is absorbed, refracted, or reflected. If it is reflected a new direction for the photon is calculated. When a photon hits the surface it is stored in a map. The information stored is the position in the space of the surface hit, energy of the photon, and the incident direction of the photon. Additionally a second photon map is used which stores the caustics information.

The second pass is the rendering where the radiance of every pixel is calculated by using the photon maps and ray-tracing. The rendering equation can be split into four parts, one for direct illumination, indirect illumination, specular reflection, and caustics. The direct illumination can be calculated from the photon map or by using ray-tracing, where ray-tracing is more accurate. Indirect illumination is calculated from the photon map where the nNN nearest photons to the intersection point is used for the calculation. As well the caustics is calculated from caustics photon map, and the specular reflection is calculated by using ray-tracing [Jensen, 1996b]. Figure A.2(b) shows an image rendered using photon mapping.



(a) Ray-tracing (b) Photon mapping (c) Path tracing

Figure A.2: The figure shows the same scene rendered with three different algorithms. **a)** Ray-tracing gives a darker image with hard shadows and without caustics. **b)** Photon mapping gives a rendered image including all the global illumination effects. **c)** Patch tracing gives an image very similar to photon mapping, but is more noisy and it is slower to render [Jensen, 2005].

APPENDIX B RAY-TRACING

Ray-tracing is a computer graphic technique to render realistic images, by tracing rays of light through the scene. It is simpler to obtain photorealistic renderings with ray-tracing than with traditional rasterization renderings, but is also more computational heavy to compute [Luebke and Parker, 2008]. Therefore ray-tracing is best suited for applications that are not time dependent, but with the extra computer power nowadays real-time ray-tracing becomes more and more common.

The idea behind ray-tracing is to simulate the physical world where rays of light are emitted from the light source and bounces around in the scene until they reach the eye/camera. Tracing the rays from the light source to the eye would require an enormous number of calculations, because the rays bounce in the entire scene and the chance of a ray to hit the eye/camera is very small. Only rays hitting the eye/camera are visible, this is what traditional ray-tracing benefit from by tracing the rays from the camera to light source, also called backwards ray-tracing.

B.1 Ray-casting

Ray-casting is the first step in ray-tracing where only primary rays are shot/casted. In ray-casting primary rays are shot from the camera through the 2D image plan into the scene, see Figure B.1. If the image plane has a resolution of 640$\times$480 pixels, then 307,200 primary rays are shot through the image plane - one for every pixel. For each ray its intersection with the geometry in the scene is calculated. If no optimizations are applied the intersection between the ray and all of the objects are calculated. The object that is intersected first, i.e. has the smallest distance from camera to intersection point, is the object that is seen in the pixel that the ray is shot through. Intersection calculation can be a very computational heavy task, especially in a scene with many polygons, therefore different methods exist, to reduce the number of intersection calculations and thereby the rendering time, e.g. scene graphs. If only ray-casting is used the value for the pixel is found from the material properties of the intersected object.

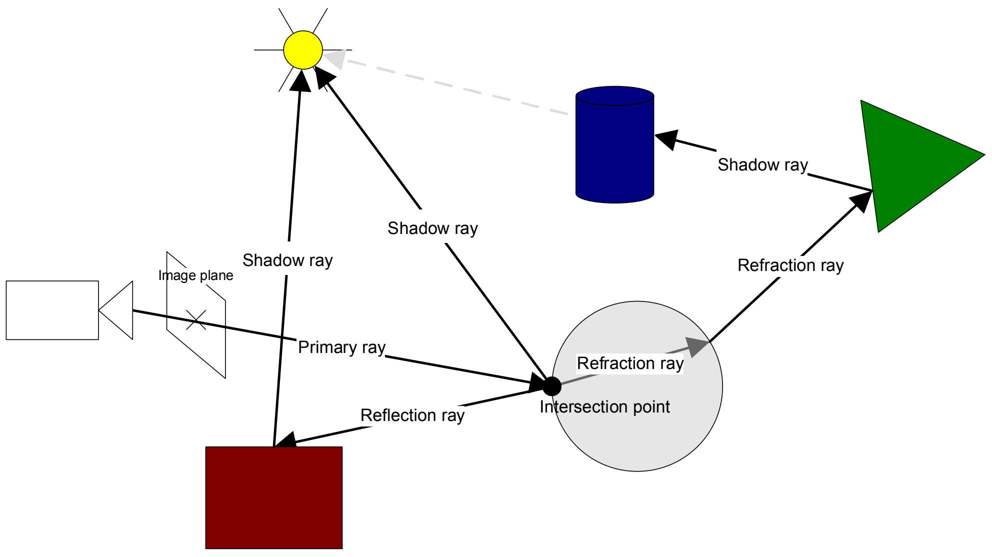


Figure B.1: Shows the different types of rays used in a ray-tracer.

B.2 Intersection Calculations

A ray is defined as a point P_0 in the space and direction $\mathbf{d}$ that is defined as a unit vector. Any point P along the ray can then be defined parametric as

$$P = P_0 + t\mathbf{d} \quad ,t > 0 \tag{B.1}$$

where:

t is the distance from P_0 to P.

The task is now to find t for the intersection of each object in the scene. The object intersected with the smallest t greater than zero is the object that is intersected first (closest to P_0).

The calculation of the ray and object intersection depends on the geometry. Two different geometries are used in this project, spheres and triangles.

B.2.1 Sphere Intersection

In vector notation the equation for a sphere is [Akenine-Möller and Haines, 2002]

$$\|P - C\| = r \tag{B.2}$$

where:

C is the center of the sphere.
r is the radius of the sphere.

A point P is located on the sphere if the distance to the center of the sphere is equal to the radius of the sphere.

If Equation (B.1) is inserted instead of P in Equation (B.2), expanded, and rearranged it becomes

$$\mathbf{d}^2 t^2 + 2\left(\mathbf{d} \cdot (P_0 - C)\right) t + (P_0 - C)^2 - r^2 = 0 \tag{B.3}$$

By using the quadratic formula the unknown quantity t can now be solved. The equation to solve the ray-sphere intersection is now found as

$$t = \frac{-\mathbf{d} \cdot (P_0 - C) \pm \sqrt{(\mathbf{d} \cdot (P_0 - C))^2 - \mathbf{d}^2 \left((P_0 - C)^2 - r^2\right)}}{\mathbf{d}^2} \tag{B.4}$$

Since the direction vector $\mathbf{d}$ is a unit vector and thus $\mathbf{d}^2 = 1$, the equation can be simplified further [Akenine-Möller and Haines, 2002]

$$t = -\mathbf{d} \cdot (P_0 - C) \pm \sqrt{(\mathbf{d} \cdot (P_0 - C))^2 - (P_0 - C)^2 - r^2} \tag{B.5}$$

A sphere can have zero, one, or two intersections with a ray. By only solving the term inside the square root it is possible to determine how many intersections exist. If the term is less than zero no intersection exists, if it is equal to zero the number of intersections is one, otherwise it is two.

B.2.2 Triangle Intersection

The idea behind ray-triangle intersection is first to find the intersection with the ray and the infinity plane defined by the triangle, and following determine if this intersection point is inside the triangle [Marschner, 2003]. To define an infinite plane a point Q in the plane and the normal vector $\mathbf{n}$ to the plane is needed. Any point P in the plane satisfy the equation

$$(P - Q) \cdot \mathbf{n} = 0 \tag{B.6}$$

By inserting Equation (B.1) and solve t, the following equation can be used to solve the ray-plane intersection:

$$t = \frac{\mathbf{n} \cdot (Q - P_0)}{\mathbf{n} \cdot \mathbf{d}} \tag{B.7}$$

If t is greater than zero the ray intersects with the plane and the next step is to determine whether the intersection point P is inside or outside the triangle.

The triangle consists of three lines - one between each vertex of the triangle. If all the lines are considered to be going counterclockwise between the vertices in the triangle the intersection point P will be on the left side of each of the three lines if it is inside the triangle, see Figure B.2. Since the triangle and intersection point is in a 3D space the easiest way to check whether the point is placed on the correct side is using the cross product. The cross product of two vectors gives the normal of the plane spanned by the two vectors. The direction of the normal depends on whether the second vector is placed clockwise or counterclockwise to the first vector, see Figure B.2.

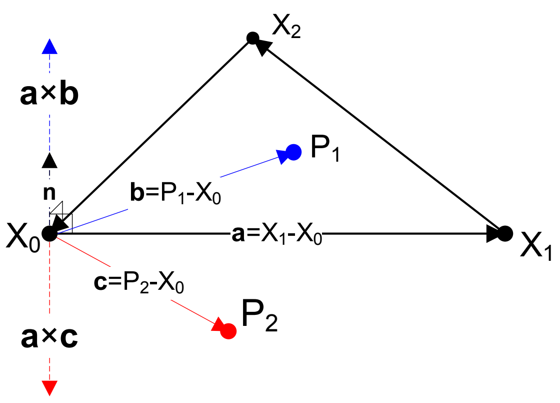


Figure B.2: The principle behind triangle intersection test. The cross product of **a** and **b** gives a normal pointing upwards, since $P1$ is placed to the left of the line from X_1 to X_2, whereas the cross product of **a** and **c** gives a normal pointing downwards.

If the line from vertex X_0 to X_1, in the triangle, is defined as a vector then the cross product between this vector and the vector from X_0 to the intersection point P gives a normal that is parallel to the normal **n** of the triangle. If the two normals are pointing in the same direction the intersection point P is placed to the left of the line from X_0 to X_1. The normals are pointing in the same direction if their dot product is greater than zero. This is done for all the three sides in the triangle and if all the normals are pointing in the same direction as **n**, then the intersection point is inside the triangle. If the vertices in the triangle are named X_0, X_1, and X_2, and are numbered counterclockwise, the following three equations are fulfilled if the intersection point P is inside the triangle [Marschner, 2003]

$$(X_1 - X_0) \times (P - X_0) \bullet \mathbf{n} > 0 \tag{B.8}$$

$$(X_2 - X_1) \times (P - X_1) \bullet \mathbf{n} > 0 \tag{B.9}$$

$$(X_0 - X_2) \times (P - X_2) \bullet \mathbf{n} > 0 \tag{B.10}$$

B.3 Secondary Rays

To get a better and more realistic rendering the primary ray is scattered to one or more secondary rays. By using secondary rays it is possible to render reflections, refractions, and shadows, see Figure B.1. Based on the properties of the material of the object a number of secondary rays are shot.

B.3.1 Reflection Ray

If the material is reflective a reflection ray is shot in the mirror reflection direction, see Figure B.3. The mirror reflection vector **r** is calculated as [Jensen, 2001]

$$\mathbf{r} = 2\mathbf{n}(\mathbf{n} \cdot \mathbf{v}) - \mathbf{v} \tag{B.11}$$

where:

v is the viewing vector from the intersection point to the camera.
n is the surface normal at the intersection point.

The reflection ray is traced further and treated as a primary ray, i.e. the intersection of the nearest object are found and secondary rays are shot from that point. All these rays contribute to the final value of the pixel that the primary rays was sent through. Usually a maximum trace depth is defined to prevent rays to be traced forever, for instance between two mirrors.

B.3.2 Refraction Ray

If the object is transparent a ray will be sent through the object, a refraction ray, see Figure B.3. Depending on the material properties (index of refraction) the ray will be refracted in a direction **t** which can be calculated by [Jensen, 2001]

$$\mathbf{t} = -\frac{\eta_1}{\eta_2}(\mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n}) - \left(\sqrt{1 - \left(\frac{\eta_1}{\eta_2}\right)^2 \left(1 - (\mathbf{v} \cdot \mathbf{n})^2\right)}\right)\mathbf{n} \tag{B.12}$$

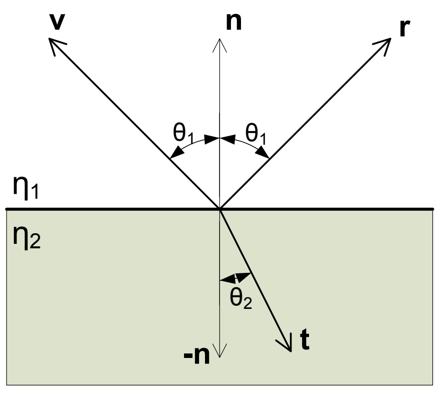


Figure B.3: The reflection and refraction vector used in ray-tracing.

where:

η_1 is the index of refraction of the medium the ray is leaving.

η_2 is the index of refraction of the medium the ray is entering.

If η_1 is greater than η_2 and the angle between the incident ray and the surface normal is sufficiently high the term inside the square root may be negative. In that case an internal reflection occurs, which is a mirror reflection inside the object.

B.3.3 Shadow Ray

Besides the material properties, the color of an object depends on the light in the scene, therefore rays are sent toward the light source(s), called shadow rays. For all shadow rays intersection calculations are made with the objects in the scene. If the ray intersects with an object (that is not transparent) placed between the light source and the point from where the shadow ray is sent, then the original intersection point will be in shadow from by the intersected object, and will therefore not receive any direct illumination from the light source.

B.4 Supersampling

A common problem in computer graphics is aliasing, which appears as jagged and pixelated edges. One method to avoid aliasing in ray-tracing is supersampling. In supersampling several rays/samples are shot through a pixel instead of only one, and an average of the samples is taken to find the radiance of the pixel, i.e. it is rendered in a higher resolution and downsampled to the displayed resolution.

If only one ray is used, the ray is usually shot through the center of the pixel. Multiple rays can be distributed within a pixel in several ways, where the simplest method is to equally align the samples in a grid structure. Supersampling can be used on all the pixels or only on the pixels at the edges to reduce the amount of computations.

B.5 Shading

At each intersection point during the tracing of a ray a radiance value is calculated, except if the material is hundred percent reflective. All these radiance values contribute to the final color of the pixel that the primary ray was shot through. How much each individual ray contributes are defined by the properties of the materials, e.g. if a material is only a little reflective the reflection ray has a corresponding small contribution to the final pixel color. How the radiance is calculated at each intersection point depends on the shading model, where some models give a more realistic image than other. In this project Modified Blinn-Phong shading is used, see Chapter 5 on page 45.

APPENDIX C

PHOTON EMISSION MAP

This appendix introduces different mappings that could be used for representing a photon emission map. First the concept of spherical coordinates is introduced, since it can play an important role in photon emission maps.

Spherical Coordinates and Latitude/Longitude Map

Figure C.1 illustrates spherical mapping, and the involved terms. The angle with the y-axis, the latitude, is denoted $u \in [0, \pi]$, while the longitude, $v \in [0, 2\pi]$, is the rotation around the y-axis. The sphere is for simplicity a unit sphere, such that all direction vectors automatically will be normalized. The Cartesian coordinate system is right-handed as the one used in OpenGL, and in the implemented ray-tracer. A latitude of 0° is found at $(0, 0, 1)$ such that the forward of the latitude/longitude map is shown in direction of the negative z-axis and left and right is shown in the direction of the negative and positive x-axis respectively.

The latitude longitude texture which is mapped on to the sphere is shown in Figure C.2.

Given a set of spherical angles *u*,*v* the location in the *x*,*y* latitude longitude texture is found by

$$\begin{aligned} x &= \left\lceil \frac{v}{\text{texture\_width}} \cdot 2\pi \right\rfloor \\ y &= \left\lceil \frac{v}{\text{texture\_height}} \cdot \pi \right\rfloor \end{aligned} \tag{C.1}$$

This sampling technique is fine for the purpose of sampling the texture for the pixel intensities. If the texture was to be mapped onto an object a sampling technique with filtering would be more appropriate, such that several texel values are averaged instead of rounding to the nearest texel.

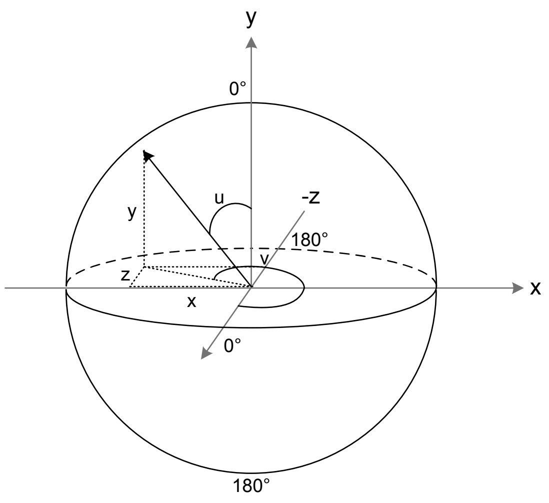


Figure C.1: Spherical coordinates

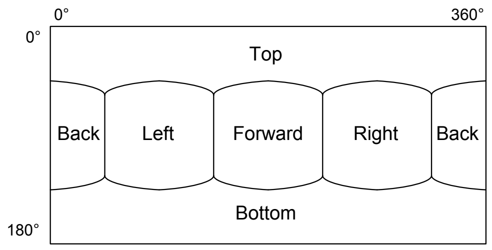


Figure C.2: Latitude/longitude map

The conversion from *u*,*v* angles to unit vectors is done by

$$\begin{aligned} x &= -\sin(v)\sin(u) \\ y &= \cos(u) \\ z &= \cos(v)\sin(u) \end{aligned} \tag{C.2}$$

The inverse from unit vector to *u*,*v* angles is calculated with

$$u = \arccos(y) \tag{C.3}$$

$$v = \arctan(z/x) \tag{C.4}$$

Where *v* has to be mapped from an interval of $[-\pi/2, \pi/2]$ to $[0, 2\pi]$. This is done by

$$v = 0.5\pi - v, \quad for, x \geq 0 \tag{C.5}$$

$$v = 1.5\pi - v, \quad for, x < 0 \tag{C.6}$$

Construction of Latitude/Longitude Map

The latitude/longitude map is just one of several method that can be used to map an environment onto a sphere. A latitude/longitude map can be constructed from, e.g. a mirror ball image or a cube map. A mirror ball image can e.g. be captured with a camera by combining two or more images captured with a 180° lens. The cube map is more suitable for computer graphics where six images in the primary directions of the coordinate system is rendered with a field of view of 90° and combined. Figure C.3 shows one format used for storing cube maps.

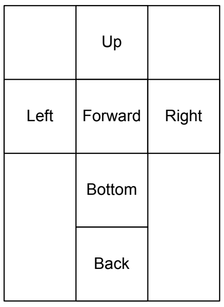


Figure C.3: Vertical cross cube map with annotations of the directions. The forward square is an image of the scene when the camera is directed along the negative z-direction, etc.

The latitude/longitude map for gaze directed photon mapping is constructed directly from the photon map vectors storing the out directions of the emitted photons, by applying Equation (C.3)-(C.6).

For traditional photon mapping a projection map, see Section 6.2.1 on page 59, can quickly be produced by rendering a cube map from the position of the light, where each object in the scene is rendered with a color/intensity according to its reflection properties. When converted to a latitude/longitude map this map can be used to guide the emission of photons in e.g. direction of specular or diffuse objects, for faster computation of the caustic map and global photon map respectively. A cube map for this purpose can be rendered with OpenGL, since global illumination knowledge is unimportant, it is only important to capture the material properties of the objects.

Photon Emission using Photon Emission Map

When a emission map, in form of a latitude/longitude texture, is available it can be used to guide emission of photons. Two ways of using the emission map is discussed here, namely naive emission map emitter and selective emission map emitter.

Naive Emission Map Emitter

This method is conceptually similar to traditional photon emitter explained in Algorithm 6.1 on page 59, with an extra test condition. When an emission direction has randomly been generated a photon is traditionally emitted in that direction. With this method the direction is checked against the photon emission map, too see whether it is an allowed emission direction. In the event of an disallowed emission direction, a new emission direction is randomly generated and checked.

A problem with this method is that, if the number of the allowed emission directions is few then it will be difficult to randomly generate a direction that is allowed by the emission map.

Selective Emission Map Emitter

The problem with the naive emission map emitter is removed by this method, but at a cost of higher memory consumption and a slower processing for emission maps with many allowed emission directions.

The basis of the selective emission map emitter, is the contents of the emission map texture. Instead of naively generating a random emission direction, and check whether it is allowed, an allowed emission direction is selected at random from the set of all allowed emission directions. The set of all allowed emission directions is generated by Algorithm C.1.

Algorithm C.1 Algorithm used to generate set of allowed emission directions.

```
INPUT:
    emissionMap
OUTPUT:
    Set of emissionDirs

void Build_Emission_Dirs(emissionMap)

1. for every (texel ∈ emissionMap)
    • if (texel = allowed direction)
        ○ find u,v, see Eq. (C.1)
        ○ calculate emissionDir, see Eq. (C.2)
        ○ add emissionDir to set of emissionDirs
    • end if
   end for
```

With a set of all allowed emission directions the final photon emission direction can be found with Algorithm C.2.

A random integer between zero and number of allowed emission directions is generated and the corresponding direction is selected. To avoid the emission directions to be uniform and dependent on the

Algorithm C.2 Algorithm used to find a unique emission direction.

INPUT:
    Set of *emissionDirs*
OUTPUT:
    Vector emissionDir

```
void find_emission_dir(emissionDirs)

1. $r \leftarrow \textbf{random}(0, \textbf{size}(emissionDirs))$
2. $emissionDir \leftarrow emissionDirs[r]$
3. $emissionDir \leftarrow emissionDir + noise$
```

allowed emission directions, a little noise can be added to each emission direction to generate truly unique emission directions.

APPENDIX D PHOTON EMISSION AND TRACING STRATEGIES

This section presents strategies for both photon emission and photon tracing that have been investigated to find the best solution to use in gaze directed photon mapping. The choice of the best solution is based on an evaluation of pros and cons of renderings generated with the outlined strategies.

First some formal definitions. The strategies described here group photons into primary photons and secondary photons. Primary photons are photons emitted from a light source before interacting with any surface. Secondary photons are photons after one or several interactions with surfaces, whether the interactions are reflection (diffuse/specular) or refraction.

D.1 Photon Emission

The investigated strategies for photon emission of primary photons are found by combining a numbered item with an alphabetic item below.

1. The emission direction of primary photons is randomly picked from the set of allowable emission directions from the previous frame, see Section 8.4.5 on page 118.

2. The emission direction is randomly chosen by a naive photon emitter.

a. Noise is added to the emission direction.

b. No noise is added.

D.2 Photon Tracing

Similar to the choice of emission strategies are the photon tracing strategies found by combining a numbered item with an alphabetic item. These strategies are only applied to photons that are to be diffusely reflected.

1. At intersections with diffuse surfaces the reflection direction is found from the nearest photon, that was a part of the path to the gaze region, in the photon map of the previous frame.

2. At intersections with diffuse surfaces the reflection direction is found as the average of the $nNNReflect$ nearest photons, that was a part of the path to the gaze region, in the photon map of the previous frame.

a. Noise is added to the reflection direction, if the reflection is the last before the photon ends in the gaze region.

b. Noise is not added to the reflection direction.

The combinations of photon emission and photon tracing strategies that have been evaluated are listed in Table D.1. The naming of the strategies is an attempt to give each strategy a more descriptive name. The name of the first strategy, is based on the initial idea, as outlined in the design chapter, see Chapter 7 on page 73, hence the name original.

Table D.1: Combination of photon emission and photon tracing strategies.

Strategy	Photon Emission	Photon Tracing
1 (Original)	1a	1b
2 (Controlled Emission /w Noise)	1b	1a
3 (Controlled Emission /w Average Noise)	1b	2a ($nNNReflect = 5$)
4 (Controlled Emission /w Average)	1b	2b ($nNNReflect = 5$)
5 (Naive Emission /w Noise)	2b	1a

Original

The original idea is very prone to fluctuation of the photons, which is a problem in real-time photon tracing with few photons because it is very visible. One good property about the strategy is that it is fast to adapt to gaze changes.

Controlled Emission with Noise

Less prone to fluctuation of photons than the original strategy. But the photons in the gaze region have a tendency to be grouped into clusters of similar trace path and thus color. In general the number of clusters is not high enough to give a randomly distribution of the photons.

Controlled Emission with Average Noise

Also less prone to fluctuation compared to the original strategy. The photons of the gaze region are not well distributed and the averaging of five nearest neighbor photons, does not provide any help in this matter. In fact the averaging tends to cluster the photons, although to a less extend than the case for controlled emission with noise strategy.

Controlled Emission with Average

The fourth strategy is similar to controlled emission with average noise but without the noise, because that is sought introduced by the averaging alone. This strategy is susceptible to the same problems that the one with noise, but with less fluctuation of the photons.

Naive Emission with Noise

The last of the tried emission combinations, is based on observations made while testing the other approaches. One common problem with the approaches relying on random selection of an emission direction was that if the gaze point was static for some time, then the random sampling would filter the emission directions such that only the directions that were well represented in the emission map were kept after subsequent frames. This is problematic because finer indirected radiance effects are removed, since they only contribute with few directions to the allowed emission directions, and with random selection they will be eliminated over a few frames.

The solution tested here was simply to emit photons as in traditionally photon mapping, and then only let the diffuse reflections be gaze directed with noise added to the last emission direction before intersection with the gaze region. The results were promising, the number of photons in the gaze region was high, and their distribution were good. But the approach did still cause photons to be fluctuating between frames.

D.3 Summary

After investigating these methods just to realize that everyone introduces photon fluctuation, it was decided to use frame coherent random numbers in the final implementation. Another idea for the final implementation, found from this review, was that it would be helpful to combine both emission techniques of primary photons to achieve good visual quality. The last conclusion made from this review was to dismiss the idea of finding the reflection direction as the average of the output direction from a number of photons, instead of just one photon. This idea was omitted because it did not provide any improvement in visual quality, and it was more computational heavy. More about the final solution can be found in Section 8.4.6 on page 118.

APPENDIX E RESULTS

This appendix lists all the results for the user tests made in the feasibility study and in the qualitative system test.

E.1 Feasibility Study

Table E.1: The results with the GI2 rendered image used in the outer area.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1: The rendering appeared visually realistic?			2	8	
2: The rendering seemed like one coherent image?	1		2	5	2

3: Rate the following aspects influence on the viewing experience (-2 = strong negative influence, 0 = no influence, +2 = strong positive influence):	-2	-1	0	+1	+2
a) Eye-tracking accuracy:		1	1	6	2
b) Transition around point of regard:		1	2	6	1
c) Image flickering:	1	4	3	7	
d) Scene geometry:			3	6	1
e) Light/shadows:	1	1	1	5	1

Table E.2: The results with the GI3 rendered image used in the outer area.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1: The rendering appeared visually realistic?			2	6	2
2: The rendering seemed like one coherent image?		1	4	5	1

3: Rate the following aspects influence on the viewing experience (-2 = strong negative influence, 0 = no influence, +2 = strong positive influence):	-2	-1	0	+1	+2
a) Eye-tracking accuracy:		2	3	3	2
b) Transition around point of regard:		2	4	4	
c) Image flickering:	1	2	5	1	1
d) Scene geometry:			5	5	
e) Light/shadows:	1			5	2

Table E.3: The results with the LI1 rendered image used in the outer area.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1: The rendering appeared visually realistic?		1	2	6	1
2: The rendering seemed like one coherent image?	1		1	7	2

3: Rate the following aspects influence on the viewing experience (-2 = strong negative influence, 0 = no influence, +2 = strong positive influence):	-2	-1	0	+1	+2
a) Eye-tracking accuracy:		3	3	3	1
b) Transition around point of regard:		1	4	2	3
c) Image flickering:	1	2	7		
d) Scene geometry:			3	7	
e) Light/shadows:	1		1	7	1

Table E.4: The results with the LI2 rendered image used in the outer area.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1: The rendering appeared visually realistic?			2	6	2
2: The rendering seemed like one coherent image?		1		4	5

3: Rate the following aspects influence on the viewing experience (-2 = strong negative influence, 0 = no influence, +2 = strong positive influence):	-2	-1	0	+1	+2
a) Eye-tracking accuracy:	1	1	6	1	1
b) Transition around point of regard:		2	5		3
c) Image flickering:			7	2	1
d) Scene geometry:			4	6	
e) Light/shadows:		1	1	6	2

E.2 Qualitative System Test

The votes from each user in the qualitative system test are listed in Table E.5. One row is the votes from one test subject. The first column (Image) lists the votes for the best quality image of static virtual quality test. The second column (Video) lists the votes for the best quality video of the real-time visual quality test. Image1 and video1 is rendered with gaze directed photon mapping, while image2 and video2 is rendered with traditional photon mapping. The third column (Experience) list the computer graphics experience of the user, where cgexp1 is low experience and cgexp3 is high experience.

Table E.5: The votes for the qualitative system test of each test subject.

Image	**Video**	**Experience**
image2	video1	cgexp3
image1	video1	cgexp2
image2	video1	cgexp2
image1	video1	cgexp3
image2	video2	cgexp3
same	video1	cgexp2
image2	video1	cgexp3
image2	video1	cgexp2
image1	video1	cgexp2
image1	video2	cgexp3
image2	video1	cgexp3
image2	video1	cgexp2
image2	same	cgexp2
same	same	cgexp3
image2	video1	cgexp2
image2	video1	cgexp2
image2	video1	cgexp2
same	video1	cgexp2
image2	video1	cgexp2
same	video1	cgexp1
image2	video1	cgexp3
image2	same	cgexp2
image2	video1	cgexp3
image2	video2	cgexp2
image2	video1	cgexp3
image2	video1	cgexp3
same	video1	cgexp3
image1	video1	cgexp3
image1	video1	cgexp3
same	video1	cgexp2
image1	video1	cgexp1
image1	same	cgexp2
image2	video2	cgexp2
image2	video2	cgexp2
image2	video2	cgexp2

APPENDIX F
ALGORITHMIC ANALYSIS

[Loudon, 1999]

Analysis of algorithms is a common task in computer science. There are several reason why it is interesting to analyze and compare algorithms, but often the objective is to find the best performing algorithm. Algorithmic performance is an ambiguous quantity as it can either be related to the time requirement of the algorithm or the storage requirement. The time requirement, sometimes referred to as time complexity of an algorithm, describes the number of steps, cycles or any other useful time measure that it takes an algorithm to process an input of fixed length. The storage requirement, or space complexity, describes the amount of memory used by an algorithm to perform a certain task.

Due to the large amount of memory available in modern computers, the attention of algorithmic analysis is in this report on the time complexity. One commonly used notation for expressing the time complexity is the big-O notation.

F.1 O-notation

The big-O notation describes the upper time complexity bound of an algorithm. It does so by expressing the growth rate of an algorithm $f(n)$ as the length of the input n increases. Obviously algorithms with slower growth rate are deemed to perform better than algorithms with faster growth rate.

The big-O notation is a theoretical analysis tool, and can in general not tell how fast a certain algorithm will run on a particular system. This is impossible since no mapping of the growth rate to a time measure exist that can cope with things like implementation details, optimizations and system properties. But the notation can give an idea about which of two algorithms that are the fastest. Table F.1 lists some common growth rates.

Table F.1: Common algorithmic growth rates in O-notation.

Complexity	Example
$O(c)$	Constant time, looking up the first element of an array.
$O(\log(n))$	Logarithmic time, binary search.
$O(n)$	Linear time, traversing a data set.
$O(n\log(n))$	Log-linear time, splitting a data set in half, and repeatably traversing each half.
$O(n^2)$	Quadratic time, traversing an image of dimension n×n.
$O(2^n)$	Exponential time, brute force comparison of two logical statements.

Using the big-O notation is straightforward, as it only relies on a few simple rules. These rules will be illustrated in the following by some examples.

Constant

$$f(n) = 4 \cdot c + 25 \tag{F.1}$$

Functions of constants or terms unrelated to the input length of the data are said to run in constant time, thus the complexity is $O(1)$.

Addition

$$f(n) = n + c \tag{F.2}$$

The complexity of this function is described by $O(n)$ since the constant term c is neglectable for most n.

$$f(n) = n + n^3 \tag{F.3}$$

The above function is reduced to $f(n) = n^3$ since the single n term is insignificant compared to n^3, and thus the complexity becomes $O(n^3)$.

Multiplication

$$f(n) = c \cdot n \tag{F.4}$$

The complexity of the above function is described by $O(n)$ since multiplicative constants are omitted as they tell little about the growth rate.

$$f(n) = 2(n+2) \cdot n \tag{F.5}$$

Expressions like this are typically rewritten into compact form such that the term with the biggest exponent is easily distinguishable, $f(n) = 2n^2 + 4n$ which according to the addition rules gives a complexity of $O(n^2)$.

GLOSSARY

The purpose of this glossary is to introduce and define the most frequently used terms in this report. Figure F.1 tries to illustrate the meaning of the terms.

Gaze point	The focus point of the user in screen space. The value provided by the eye tracker.
Gaze area	Area in image space immediately encircling the gaze point.
Point of regard (PoR)	Projection of the gaze point into the 3D space of the scene.
Gaze region	Area in scene space encircling the point of regard.
Outer area	The exterior to the gaze area, in screen space.
Visual Angle	The angle that spans the gaze area.
Local illumination rendering	The rasterized rendering made with OpenGL.
Global illumination rendering	The rendering made with ray-tracing and photon mapping.
Hybrid rendering	Term used to describe a rendering composed of two or more rendering techniques, in this case local and global illumination.

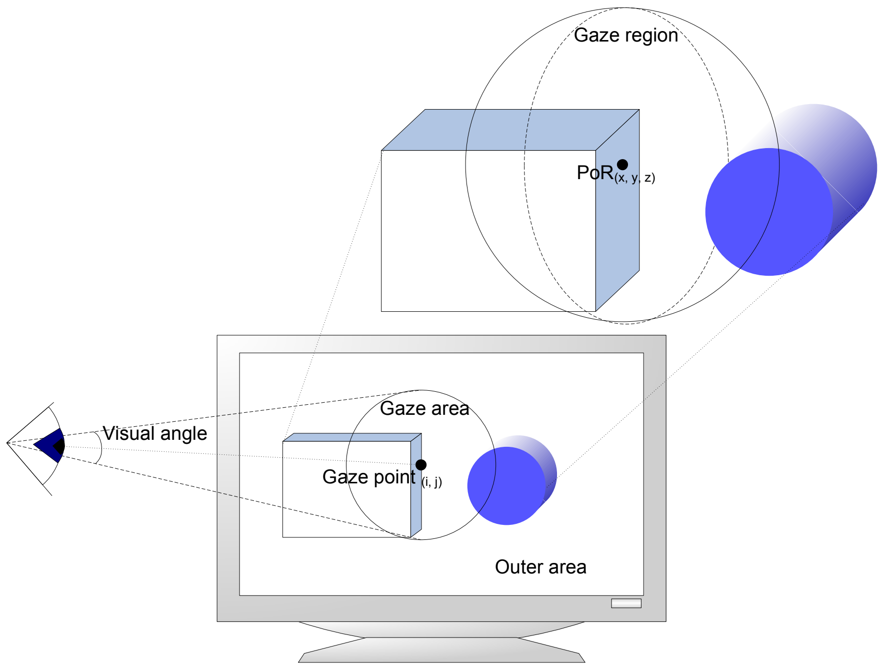


Figure F.1: The figure illustrates some of the terms used in this report regarding gaze directed rendering.

BIBLIOGRAPHY

Abdi1, H. and Molin, P. (2007). Lilliefors/Van Soest's Test of Normality.

Addy Ngan, F. D. and Matusik, W. (2004). Experimental Validation of Analytical BRDF Models. Technical Sketch Siggraph 2004. `http://people.csail.mit.edu/wojciech/BRDFValidation/index.html`.

Akenine-Möller, T. and Haines, E. (2002). *Real-Time Rendering 2nd Edition*. A. K. Peters, Ltd., Natick, MA, USA.

Angel, E. (2009). *Interactive Computer Graphics - A Top Down Approach using OpenGL*. Pearson Education, Inc., 5th edition. ISBN-10: 0-321-54943-0.

Birn, J. (2000). *Digital Lighting and Rendering*. New Riders Publishing.

Brown, M. (2004). *Advanced Digital Photography*. David O'Sullivan.

Christensen, P. H., Jensen, H. W., and Suykens, F. (2001). A Practical Guide to Global Illumination using Photon Mapping. In *SIGGRAPH '01: ACM SIGGRAPH 2001 course 38*, page 1.

Chuang, J.-H. (2003). Ray Tracing – Assignment 3. Webpage. `http://cggmwww.csie.nctu.edu.tw/courses/cg/2003/prog3/`.

Clarke, J.H. (1976). Hierarchical Geometric Models for Visible Surface Algorithms. *CACM*, 19(10):547–554.

Duchowski, A. T. (2007). *Eye Tracking Methodology*. Springer, 2nd edition.

Dutre, P., Bala, K., and Bekaert, P. (2002). *Advanced Global Illumination*. A. K. Peters, Ltd.

Ferwerda, J. A., Pattanaik, S. N., Shirley, P., and Greenberg, D. P. (1996). A Model of Visual Adaptation for Realistic Image Synthesis. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 249–258, New York, NY, USA. ACM.

Funkhouser, T. A. and Sequin, C. H. (1993). Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. pages 247–254.

George Stetten and Korin Crawford (1997). GlutMaster – version 0.3 . Webpage. `http://www.stetten.com/george/glutmaster/glutmaster.html`.

Gibson, S. and Hubbold, R. (1997). Perceptually-Driven Radiosity. In *Computer Graphics Forum 16(2)*, pages 129–140.

Günther, J., Popov, S., Seidel, H.-P., and Slusallek, P. (2007). Realtime Ray Tracing on GPU with BVH-based Packet Traversal. In *Proceedings of the IEEE/Eurographics Symposium on Interactive Ray Tracing 2007*, pages 113–118.

Gutierrez, D., Anson, O., Munoz, A., and Seron, F. J. (2005). Perception-Based Rendering: EyesWide Bleached. In *EuroGraphics 2005*.

Gutierrez, D., Seron, F. J., Anson, O., and A. Mu n. (2004). Chasing the Green Flash: A Global Illumination Solution for Inhomogeneous Media. In *SCCG '04: Proceedings of the 20th spring conference on Computer graphics*, pages 97–105, New York, NY, USA. ACM.

Hanrahan, P. (2001). Monte Carlo Path Tracing. SIGGRAPH 2001.

Hedley, D., Worrall, A., Paddon, D., and D (1997). Selective culling of discontinuity lines. In *in Rendering Techniques '97 (Proceedings of the Eigtht Eurographics Workshop on Rendering)*, pages 69–80. Springer-Verlag.

Helen Sharp, Yvonne Rogers, J. P. (2007). *Interaction Design: Beyond Human-computer Interaction*. John Wiley and Sons.

Hillaire, S., Lécuyer, A., Cozot, R., and Casiez, G. (2007). Depth-of-Field Blur Effects for First-Person Navigation in Virtual Environments.

Hillaire, S., Lécuyer, A., Cozot, R., and Casiez, G. (2008). Using an Eye-Tracking System to Improve Depth-of-Field Blur Effects and Camera Motions in Virtual Environments.

Hoppe, H. (1997). View-dependent refinement of progressive meshes. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 189–198, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Hornberg, A. (2006). *Handbook of Machine Vision*. Wiley VCH.

Hunter Murphy and Andrew T. Duchowski (2001). Gaze-Contingent Level Of Detail Rendering.

Intel (2008). Open Source Computer Vision Library. Webpage. `http://sourceforge.net/projects/opencvlibrary/`.

Jensen, H. W. (1996a). Global Illumination Using Photon Maps. Extended version of the paper in Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering).

Jensen, H. W. (1996b). Global Illumination using Photon Maps. pages 21–30. Springer-Verlag.

Jensen, H. W. (2001). *Realistic Image Synthesis using Photon Mapping*. A K Peters Natick, Massachusetts.

Jensen, H. W. (2005). Cornell Box Images. Webpage. `http://graphics.ucsd.edu/~henrik/images/cbox.html`.

Jensen, H. W. and Christensen, N. J. (1995). Photon Maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects. In *Computers & Graphics vol. 19 (2)*, pages 215–224.

Jensen, H. W. and Christensen, P. (2007). High Quality Rendering using Ray-tracing and Photon Mapping. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, page 1, New York, NY, USA. ACM.

Keller, A. and Wald, I. (2000). Efficient Importance Sampling Techniques for the Photon Map.

Khronos Group (2008). OpenGL.org. Webpage. `http://www.opengl.org`.

Khronos Group (2009). GLUT - The OpenGL Utility Toolkit. Webpage. `http://www.opengl.org/resources/libraries/glut/`.

Lai, Y.-C., Fan, S., Chenney, S., and Dyer, C. (2007). Population Monte Carlo Path Tracing.

Ledda, P., Santos, L. P., and Chalmers, A. (2004). A Local Model of Eye Adaptation for High Dynamic Range Images. In *AFRIGRAPH '04: Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 151–160, New York, NY, USA. ACM.

Levoy, M. and Whitaker, R. (1990). Gaze-directed Volume Rendering. *SIGGRAPH Comput. Graph.*, 24(2):217–223.

Loudon, K. (1999). *Mastering Algorithms with C*. O'Reilly Media Inc.

Luebke, D. and Erikson, C. (1997). View-Dependent Simplification of Arbitrary Polygonal Environments. Technical report, Chapel Hill, NC, USA.

Luebke, D., Hallen, B., Newfield, D., and Watson, B. (2000). Perceptually driven simplification using gaze-directed rendering. Technical report, Rendering Techniques 2001, Springer-Verlag (Proc. Eurographics Workshop on Rendering.

Luebke, D. and Parker, S. (2008). Interactive Ray Tracing with CUDA. `http://developer.nvidia.com/object/nvision08-IRT.html`.

Marschner, S. (2003). Simple Ray-Triangle Intersection.

Martin, I., Pueyo, X., Aplicada, D. D. I. M., and Tost, D. (1997). An Image-space Refinement Criterion for Linear Hierarchical Radiosity. In *in Graphics Interface '97*, pages 26–36. Morgan Kaufmann.

Mulder, J. D. and Liere, R. V. (2000). Fast Perception-based Depth of Field Rendering. In *In VRST 2000*, pages 129–133. ACM Press.

Murphy, H. and Duchowski, A. T. (2007). Hybrid image-/model-based gaze-contingent rendering. In *APGV '07: Proceedings of the 4th symposium on Applied perception in graphics and visualization*, pages 107–114, New York, NY, USA. ACM.

Myszkowski, K. (1998). The Visible Differences Predictor: Applications to Global Illumination Problems. In *Rendering Techniques '98 (Proceedings of the Ninth Eurographics Workshop on Rendering)*, pages 223–236.

Nikolov, S. G., Newman, T. D., Bull, D. R., Canagarajah, N. C., Jones, M. G., and Gilchrist, I. D. (2004). Gaze-contingent Display using Texture Mapping and OpenGL: System and Applications. In *ETRA '04: Proceedings of the 2004 symposium on Eye tracking research & applications*, pages 11–18, New York, NY, USA. ACM.

Ohshima, T., Yamamoto, H., and Tamura, H. (1996). Gaze-directed Adaptive Rendering for Interacting with Virtual Space. In *VRAIS '96: Proceedings of the 1996 Virtual Reality Annual International Symposium (VRAIS 96)*, page 103, Washington, DC, USA. IEEE Computer Society.

Owen, G. S. (1999). Ray Tracing. Webpage. `http://www.siggraph.org/education/materials/HyperGraph/raytrace/rtrace0.htm`.

Parkhurst, D. and Niebur, E. (2004). A Feasibility Test for Perceptually Adaptive Level of Detail Rendering on Desktop Systems. In *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, pages 49–56, New York, NY, USA. ACM.

Peter, I. and Pietrek, G. (1998). Importance Driven Construction of Photon Maps. In *In Rendering Techniques 98 (Proceedings of the 9th Eurographics Workshop on Rendering)*, pages 269–280. Springer-Verlag.

Pharr, M. and Humphreys, G. (2004). *Physically Based Rendering - From Theory to Implementation*. Elsevier.

Phong, B. T. (1975). Illumination for Computer Generated Pictures. *Commun. ACM*, 18(6):311–317.

Purgathofer, W. (1999). Overview of Perceptually-Driven Radiosity Methods. Technical report.

Reddy, M. (1997). *Perceptually Modulated Level of Detail for Virtual Environments*. PhD thesis, University of Edinburgh.

Ritschel, T., Ihrke, M., Frisvad, J. R., Coppens, J., Myszkowski, K., and Seidel, H.-P. (2009). Temporal Glare: Real-Time Dynamic Simulation of the Scattering in the Human Eye. In *EuroGraphics 2009*.

Seetzen, H., Ward, G., Whitehead, L., and Heidrich, W. (2004). High Dynamic Range Display System. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Emerging technologies*, page 8, New York, NY, USA. ACM.

Spencer, G., Shirley, P., Zimmerman, K., and Greenberg, D. P. (1995). Physically-Based Glare Effects for Digital Images. volume 29, pages 325–334.

Suykens, F. and Willems, Y. D. (2000). Density Control for Photon Maps. In *In Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pages 23–34. Springer Wien.

Tamstorf, R. and Jensen, H. W. (1997). Adaptive Sampling and Bias Estimation in Path Tracing. In *In Eurographics Rendering Workshop*, pages 285–295.

Tobii (2001). Tobii Technology AB.

Tobii (2007). Product Description: Tobii T/X Series Eye Trackers.

Tobler, R. F. (1997). Radiosity – Ray Tracing. Webpage. `http://www.cg.tuwien.ac.at/research/rendering/rays-radio/`.

Tuttle, C. (2003). An Analysis of Path Tracing and Photon Mapping in an Attempt to Simulate Full Global Illumination. University of California.

Wald, I., Slusallek, P., Benthin, C., and Wagner, M. (2001). Interactive Rendering with Coherent Ray Tracing. In *Computer Graphics Forum*, pages 153–164.

Ware, C. (2000). *Information Visualization*. Morgan Kaufmann Publisher, 1st edition.

Waters, Z. (2007). Photon Mapping. Webpage. `http://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/photon_mapping/PhotonMapping.html`.

Xia, J. C. and Varshney, A. (1996). Dynamic view-dependent simplification for polygonal models. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 327–ff., Los Alamitos, CA, USA. IEEE Computer Society Press.

Yoshida, A., Ihrke, M., Mantiuk, R., and Seidel, H.-P. (2008). Brightness of the Glare Illusion. In *APGV '08: Proceedings of the 5th symposium on Applied perception in graphics and visualization*, pages 83–90, New York, NY, USA. ACM.

Zhou, K., Hou, Q., Wang, R., and Guo, B. (2008). Real-time KD-tree Construction on Graphics Hardware. *ACM Trans. Graph.*, 27(5):1–11.