Rendering of Navigation Lights

Martin Skytte Kristensen

Kongens Lyngby 2012
IMM-MSc-2012-125

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk IMM-MSc-2012-125

Summary

The goal of the thesis is to improve the rendering of Aids to Navigation (ATON) in the ship simulator developed by FORCE Technology using a simplified model for light diffraction in the human eye. The rendering is based on High Dynamic Range (HDR) intensities specified in candelas instead of empirical RGB values relative to display intensity. The light sources are modeled as angularly masked isotropic light sources.

The thesis explains the background and related works on how to display a HDR image on a display with limited dynamic range.

The thesis presents a real-time method for rendering the glare of ATON lights using billboards in a consistent way for sub pixel and supra pixel sizes. The method can generate glare based on spectral rendering for actual light spectra.

Preface

This thesis was prepared at the department of Informatics and Mathematical Modelling at the Technical University of Denmark in fulfilment of the requirements for acquiring an M.Sc. in Digital Media Engineering.

The thesis deals with rendering Aids to Navigation lights in a ship simulator and consists of seven chapters and an appendix section.

Lyngby, 01-October-2012

Martin Skytte Kristensen

Acknowledgements

I would like to thank Jørgen Royal Petersen at the Danish Maritime Authority for lending me three buoy lanterns to study the glare phenomenon for LED lights.

For feedback on the report, I thank my supervisor at DTU, Jeppe Revall Frisvad, my supervisor from FORCE Technology, Peter Jensen Schjeldahl and my friends Sune Keller, Jacob Kjær and Meletis Stathis.

Contents

Glossary

ATON Aids to Navigation. 1–3, 6, 11, 15, 33, 34, 37, 39, 41, 45, 46, 50, 51, 59, 107

CIE Commission internationale de l'éclairage. 11

FFT Fast Fourier Transform. 53, 67

GPGPU General Purpose GPU. 52, 53

HDR High Dynamic Range. 2, 4–8, 37, 40, 43, 44, 52, 59, 107–109

HVS Human Visual system. 2, 4, 5, 19, 23, 25, 31–34, 37, 39, 60

IALA International Association of Marine Aids to Navigation and Lighthouse Authorities. 1, 11, 12, 17, 29, 46, 111, 112

JND Just Notable Difference. 33

LDR Low Dynamic Range. 2, 4, 5, 40, 44, 45, 108

LED light-emitting diode. 13, 14, 19, 21, 79, 91, 107

NDC normalized device coordinates. 72, 75

PSF Point-Spread Function. 36, 44, 52, 54–57, 67, 74, 78, 108

SPD Spectral Power Distribution. 29, 46, 52, 55, 79, 107

TMO tone map operator. 39, 40, 45, 59, 61, 88, 92, 96, 99

TVI Threshold-versus-Intensity. 33, 38, 39, 59, 61, 107

Chapter 1

Introduction

The purpose of this thesis is to investigate how well we can simulate the appearance of Aids to Navigation (ATON) lights as perceived by ship navigators. This is important in ship simulators developed for the training of navigators. To be useful in a training simulator, the method we develop must be suitable for implementation in a real-time rendering system that renders to several projectors or screens.

Aids to Navigation In weather conditions with low visibility (e.g. low light or dense fog), a ship navigator can use standardized signal lights as navigational aids (called ATON lights). The visual characteristics of a signal light allows the navigator to identify the signal source as for instance a light house that has color information to guide ships safely through passages, a buoy that signifies a recent, unmapped ship wreck or a nearby ship on collision course.

The appearance and behavior of ATON are standardized in the form of International Association of Marine Aids to Navigation and Lighthouse Authorities (IALA) recommendations to ensure consistency and safety for international sea travel.

Light Source Perception When the human navigator perceives a navigation light it is sensed using the Human Visual system (HVS), which has a major and individual influence on how the light is perceived. This is clearly demonstrated by the fact that some people are partially or completely color blind.

A less known effect is glare or veiling luminance, which is caused by the scattering and diffraction of light as it passes through the eye and is sensed by the photo-receptors on the retina. The glare causes the light to "bleed" to the surroundings causing both decreased contrast and increased brightness. It can appear as a faint glow around the light source or as fine radial needles depending on the angular size of the light source in the visual field (see figure 1.1) so looking at a distant light source with high contrast to background intensity, the light appears larger than the actual physical object.

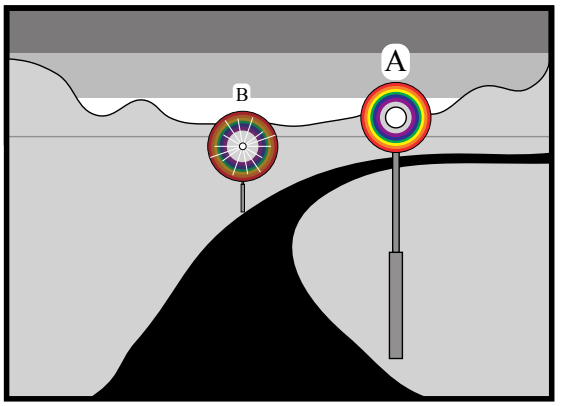


Figure 1.1: Glare from two light sources **A** and **B**. The distant light - which covers a smaller angle of the visual field - shows the fine needle pattern. From [SSZG95]

The lower ambient illumination at night increases the contrast to the light sources so the glare from ATON lights is more strongly perceived.

Ordinary projectors and monitors cannot display light intensely enough to produce glare as real light sources would because the range of displayable intensities are much lower (Low Dynamic Range (LDR)) than the intensities perceived by the HVS (High Dynamic Range (HDR)). The highest intensity a display can produce is called the *white level* and lowest intensity is called the *black level*. The static contrast of a display is then the ratio of the white level to black level[1]. A

[1]Some displays analyze the image and dynamically alters the intensity - which influences black level - to produce a larger contrast ratio, but this may lead to inconsistent behavior.

high black level (such as from a projector where light is reflected from a screen) cannot display the low ambient illumination of night scenes, which means the contrast to the ATON lights would be further reduced. Ambient light in the observer room also contributes to the effective black level of a display.

If we do not simulate glare, light sources will appear dull, not as bright and not as big (or not at all if the light source becomes smaller than a pixel) as we would perceive them in real life. Thus the glare phenomenon - and how to display it on available devices - is important in a ship simulator, especially in night simulations.

Atmospheric phenomena When light travels through a participating medium such as fog and mist from rain, the photons are scattered in different directions. The surrounding particles will be lit causing a glow around the light source and giving indirect illumination to nearby objects. In addition the scattering will change the specular reflection of materials.

Ignoring the indirect lighting from the scattering will cause the scene to appear duller than expected.

FORCE Technology This project is being done in cooperation with FORCE Technology, who has developed a ship simulator, SimFlex, and they are interested in researching how a physically based model of navigation lights can increase the realism and confidence in the simulation.

(a) Outside the setup **(b)** Inside the setup

Figure 1.2: A 360° training simulator at FORCE based on projector displays. Courtesy Force

FORCE has built replica of ship bridges to enhance the realism of the simulators. Some of the simulators have 360 degrees view, built from multiple tiled monitors (shown in figure 1.3) or overlapping projectors (shown in figure 1.2). Currently

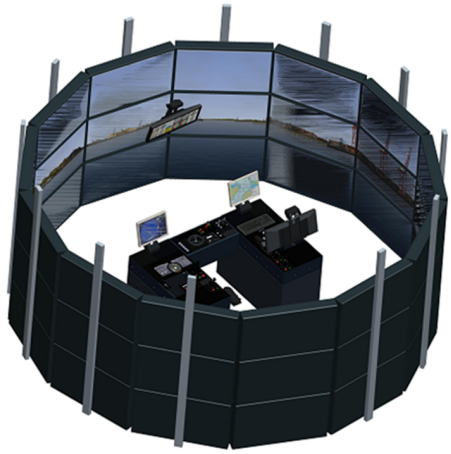


Figure 1.3: 360° LCD training simulator at FORCE

the rendered output is sent directly to the display (LCD monitors or projectors) and as such the lighting computations must directly compensate for the LDR nature of the displays.

The simulation may be observed by multiple people at the same time (e.g. teaching scenarios) which makes some of the HVS impossible or impractical to simulate.

Rendering Challenges When the aim is to render realistic lights based on actual physical lights, the first step would reasonably be to render the lights at their luminous intensity. The intensities would cause the glare effect automatically.

Here we encounter the problem of displaying the results on a device that cannot reproduce the rendered intensities (figure 1.4), and the produced intensities and contrast are not high enough to cause glare; it has to be added manually. If rendered intensity is either lower than the black level, or brighter than the white level, then details are lost. HDR displays exist, but they are expensive, not used by FORCE and not (yet) suited for 360° projection so we will focus on LDR displays.

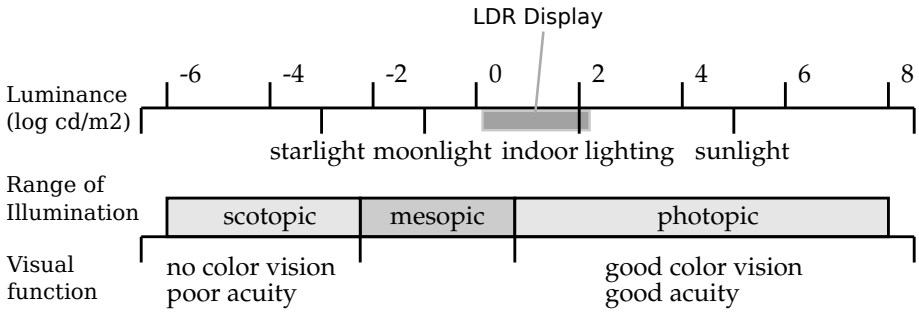


Figure 1.4: The dynamic range of the HVS. After [FPSG96]

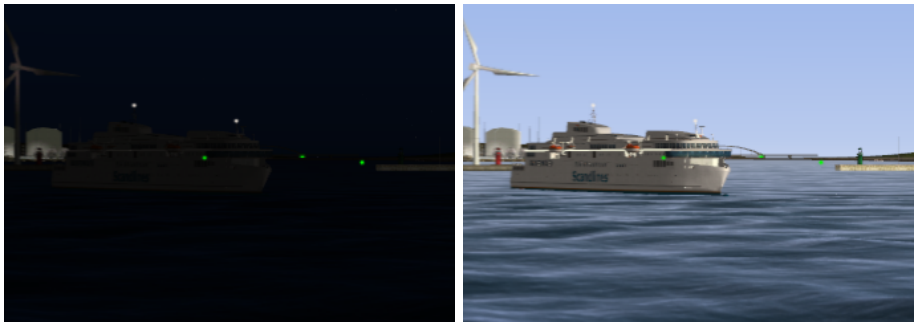
We need a way to map the absolute scene luminance intensities to intensities that can be showed reasonable faithfully on a LDR display. This process is called Tone Mapping and is itself a large and active research area, though much of the literature is concerned with static images.

One challenge for this thesis is to find a method that has a low computational footprint, takes into account the HVS behavior allowing adaptation to night and day illuminance levels and produces convincing results. The subjective experience of the HVS, which changes with age, makes it difficult attain convincing results for all observers. A user study that takes the actual observer environment (such as ambient lighting and field of view) into account will be needed to tune the method, but that is outside the scope of this project.

Another challenge is screen-resolution. A monitor has less resolution than the retina and at a certain distance, navigation lights become sub-pixel sized, but still perceptually visible through the glare phenomenon. Rendering polygons with sub-pixel sizes causes flicker as they are rendered in some frames and not in others, which is not acceptable in an accurate simulation.

Light rendering in SimFlex To give an impression of the level of realism in the current SimFlex ship simulator, figure 1.5 illustrates how navigation lights are visualized an early February morning.

The light sources are rendered as billboards with an additional glare billboard that changes size and color depending on distance and visibility. For shading of objects, SimFlex computes a list of lights that contributes to the shading of a material using forward rendering. The list is thresholded as in the worst case more than 1000 lights can be active. As the engine does not support HDR, the light intensities are specified relative to the display in range $[0, 1]$.



(a) Feb. 6. 6.30 AM **(b)** Feb. 6. 8.00 AM

Figure 1.5: Screen dump from the SimFlex simulator

The amount of specular water reflection is controlled by wind speed, the higher the wind speed, the fainter the reflections.

In this thesis, we strive to improve the modeling and direct appearance of ATON lights using physically correct HDR light intensity values based on actual ATON light specifications and glare based on a simplified model of the human eye.

1.1 Project scope

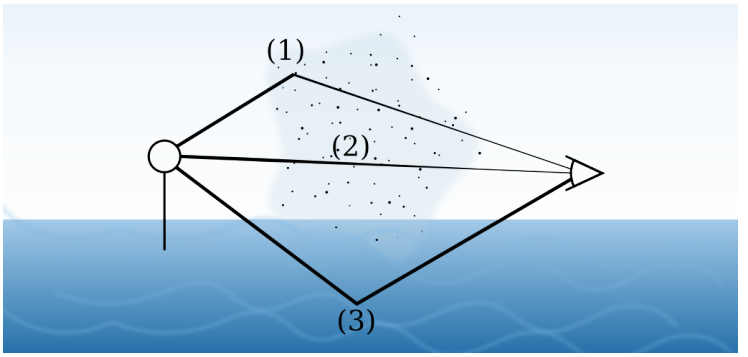


Figure 1.6: A loose overview of some of the external components in rendering Aids to Navigation lights.

For open projects such as this, limiting the scope is critically important and it must be recognized that only a subset of the problem can be solved in the time-scope of a Master's thesis project.

Modeling of the environment As SimFlex does not support HDR, and for maximal flexibility, this project is implemented as a prototype outside the SimFlex code base. As such I need an environment to display the light sources in. Modeling the atmospheric weather conditions such as sky-color and clouds are beyond the scope of this thesis. The implementation will build upon the SilverLining evaluation SDK from Sundog Software which will render clouds sky and provide HDR values for direct sunlight and ambient light.

Focus on direct appearance of point light sources How the lights illuminate surfaces will not be part of this project (ray (3) in figure 1.6). Likewise shadows are also out of scope. These are important features, especially concerning lighting effects and appearance in participating media such as dense fog (such as ray (1) in figure 1.6), but time constraints will not allow it in this project. Instead we will focus on (2) in figure 1.6.

Scalable Real-time performance As the simulator is interactive, the method should have real-time performance and scale to hundreds of light sources.

Single-channel rendering Rendering 360° horizontal field of view is a computationally expensive task. At FORCE it is done using multiple networked workstations. This introduces latency and architectural challenges that are beyond the scope of this project. I will, however, make notes about possible issues concerning such a setup and, if possible, give some directions on how to work around them. I will strive to make the core parts of the method compatible with multichannel rendering.

Convolution constraints To prevent discontinuities in multichannel setups, the image-planes of neighboring channels have to be increased with the radius of the filters. In practice, this is not an issue for small filter widths (such as 5 pixels). However, the performance hit of increasing the viewport with the radius of filters used for physically based glare rendering probably quickly becomes prohibitive.

No Peripheral Effects As peripheral effects are not possible because there can be multiple viewers in the simulators at FORCE. Even for one viewer, the center of the screen is not a good approximation for observer focus because the camera is tied to the orientation of the ship. For the single observer scenario

eye tracking might be useful for further investigation with regards to peripheral effects.

Tone mapping constraints The HDR nature of this project opens up the general tone mapping problem. Severe assumptions have to be made to allow the project to finish and keep focus on lights.

1.2 Related Works

For this project I need solutions to the tone-mapping problem for real-time HDR rendering over time, glare appearance when looking at light sources, both very distant and close.

Light appearance for driving simulators was investigated by Nakamae et al. as part of their work on work on appearance of road surfaces [NKON90]. Their model was based on pre-computing an analytical approximation of diffraction through the pupil and eyelashes and convolving the image.

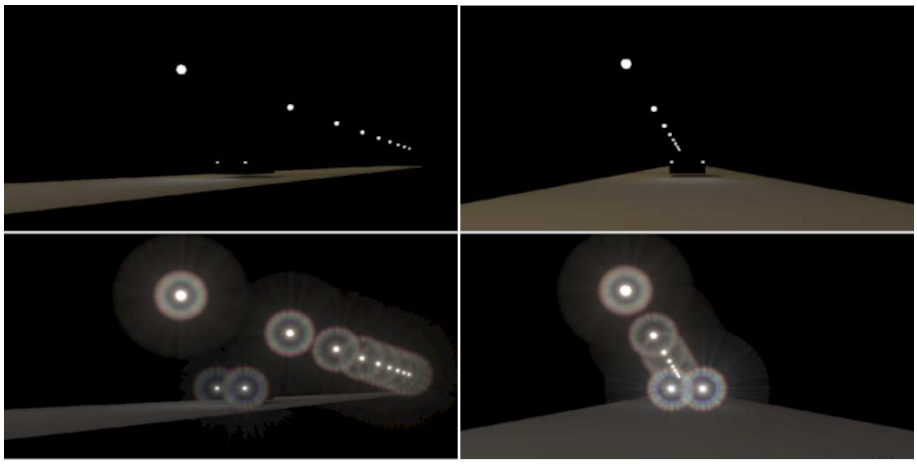


Figure 1.7: Applied glare pattern from Spencer et al. [SSZG95]

The glare phenomenon has been discussed and investigated in the literature. Simpson et al. described the characteristics and appearance of glare in [Sim53]. He described experiments for studying the phenomenon and gave the radius of the *lenticular halo*. Their work formed the empirical basis for Spencer et al. [SSZG95] who generated a 2D filter based on the observations of Simpson.

The model has been used in later interactive works ([DD00], using hardware with dedicated convolution support) and was shown to increase the perceived brightness in the study performed by Yoshida et al. [YIMS08]. Different filter kernels was proposed for day vision, night and low-light vision. For this project the proposed filter kernel is too large for interactive use and their results for synthetic scenes are not impressive (see figure 1.7).

Kakimoto et al. used wave-optics theory to compute the diffraction of eye lashes and the pupil for car headlights [KMN+05b] (see figure 1.8).

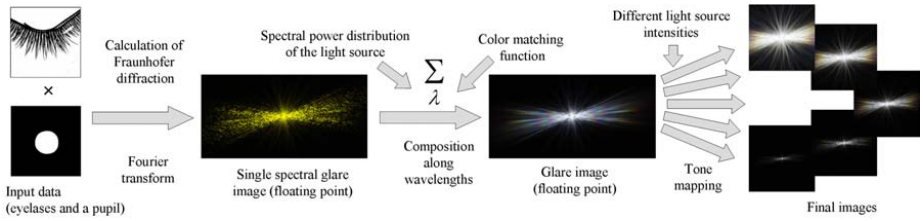


Figure 1.8: The glare pipeline from Kakimoto et al. [KMN+05b]

Ritschel et al. [RIF+09] focused on the temporal dynamics of the particles in the eye. Their proposed model was based on diffraction, where multiple parts of the eye's internal structure were part of the model (lens fibers, impurities in the eye fluid and pupil contractions based on luminance level). Like [SSZG95], they computed the glare pattern as a 2D filter kernel which was used to spread the intensity of a pixel to the surrounding pixels in a process called *convolution*. The effect of convolving the brightest pixels with the glare filter kernel compared to placing a single billboard with the kernel kernel is shown in figure 1.9 They performed a study showing the brightness enhancement effects of the temporal aspect. Their work forms the basis of the perceptual glare part of this project.

For distant lights where the surface geometry is smaller than a pixel, the closest work is the phone wire anti-aliasing method by Persson [Per12] where the phone wire forced to a minimum screen pixel size and then the intensity is attenuated according to distance.

For the tone-mapping problem, a vast number of methods have been proposed. Variations of the global operator from Reinhard et al. [RSSF02] have been widely used for real-time rendering [Luk06, AMHH08, EHK+07] using temporal light and dark adaptation from Pattanaik et al. [PTYG00]. Perceptual effects (glare, loss of color and detail under low light) was added by Krawczyk et al. [KMS05], through their model of glare is a post-process mono-chromatic Gaussian blur and too simplified for this project.

Figure 1.9: Glare applied with convolution versus billboarding. From [RIF+09]

An analytical model for isotropic point lights with single scattering is described in [SRNN05] that models the glow, indirect illumination (described as airlight) and change in specular reflectance. Shader source code and lookup table data for parts of the analytical equation are provided from their homepage as well. To be applicable in a ship simulator, the method will need careful optimizations to scale to hundreds of lights without visual artifacts, as performance scales linearly with the number of lights. As a result, this thesis will not explore atmospheric single scattering.

Chapter 2

Appearance of Aids to Navigation

ATON have been standardized by IALA as recommendations. Relevant for this project is the recommendation for color [IAL08a] and luminous range [IAL08b]. Further national regulations [Sø07] describe requirements to navigation aids on ships regarding how and where they emit light.

2.1 Properties

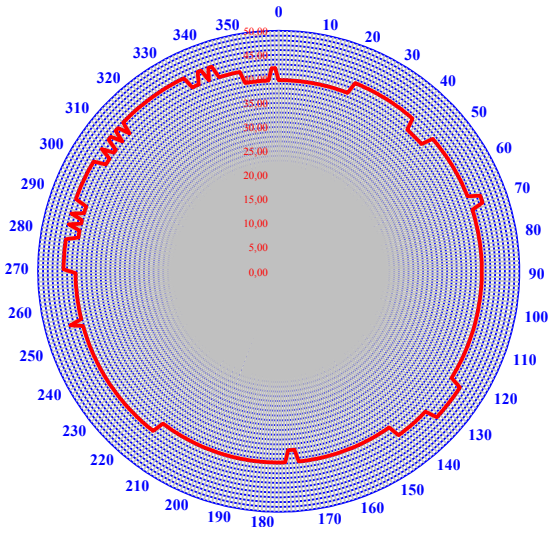
The following properties are relevant to the modeling of ATON lights:

Color The navigation aids can be blue, green, red, white and yellow, depending on use. The IALA recommendations specify ranges of color variations for each color in Commission internationale de l'éclairage (CIE) 1931 xy chromacity coordinates (which will be explained in section 3.3).

Sectoring The light emission can be horizontally split into sectors defined as an arc where light is emitted. Intensity might fall off at the edges of the sector

and might overlap neighbor sectors (the maximum of overlap is regulated and depends on where the light source is used).

Additionally, the horizontal emission may be masked by internal components in the light. A measured horizontal profile is shown in figure 2.1 for a Sabik LED-155 (though this profile does not show significant masking, light house lanterns do [Pet12]).



RESULTS

Io* -	value: (cd)	40
Mean -	value: (cd)	42
Max. -	value: (cd)	45
Min. -	value: (cd)	37

* 10th Percentile Intensity

Figure 2.1: Horizontal emission profile of a Sabik LED155 white. Courtesy [Pet12]

Vertical emission profile To increase horizontal intensity, lanterns usually utilize a Fresnel lens to focus light horizontally at the expense of vertical intensity. Figure 2.3 shows how a Fresnel profile lens and mirrors can focus the light for a light house lantern. A measured vertical profile for a Sabik LED-155 lantern is shown in figure 2.2.

Nominal range The minimum distance, measured in nautical miles (1 nautical mile = 1.852 km), under nominal atmospheric conditions, at which the light is visible on top of the background illuminance, is called *nominal range*.

The luminous intensity of a light source can be computed using Allard's Law (see IALA recommendation E200-2, [IAL08b]) from the nominal range d, the

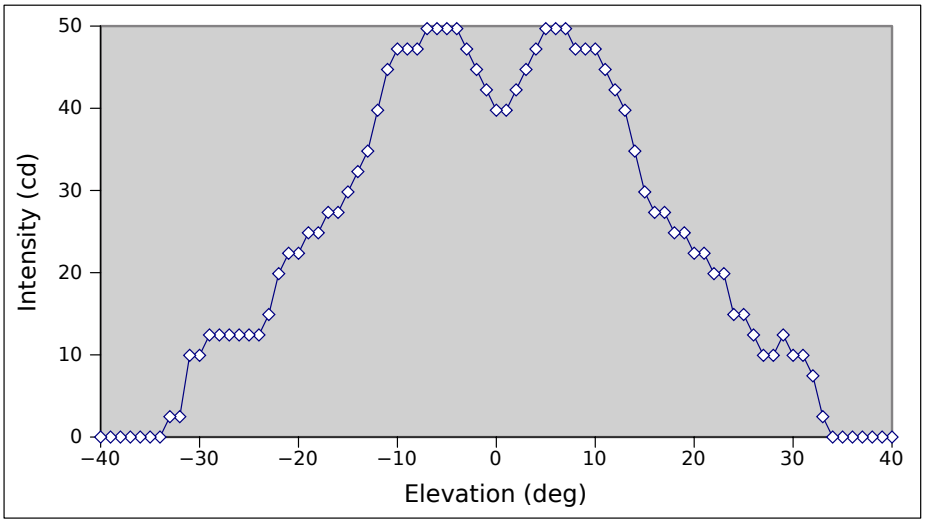


Figure 2.2: Vertical emission profile of a Sabik LED155 white. Data courtesy [Pet12]

illuminance E_t and atmospheric visibility V:

$$I(d) = 3.43 \cdot 10^6 E_t d^2 0.05^{-\frac{d}{V}} \tag{2.1}$$

The atmospheric visibility is assumed 10 nautical miles and standard required illuminance E for day time is $1 \cdot 10^{-3}$ lux and for night time is $2 \cdot 10^{-7}$ lux, but the recommendations also specify that the background illuminance has to be factored in, which may increase the required illuminance with a factor 100 under "substantial background lighting".

In the real world, light intensity is given in candelas, the photometric unit for luminous intensity. At daylight levels the E200-2 notes that to have a nominal range of one nautical mile or more, kilocandela intensities are required.

Blinking Blinking (or flashing) allows the light to communicate more than just the color can and it increases the perceived luminance. Different patterns are shown in figure 2.4.

Light source types Light towers, beacons and buoys currently use light-emitting diode (LED) and tungsten sources. The LED sources are designed to

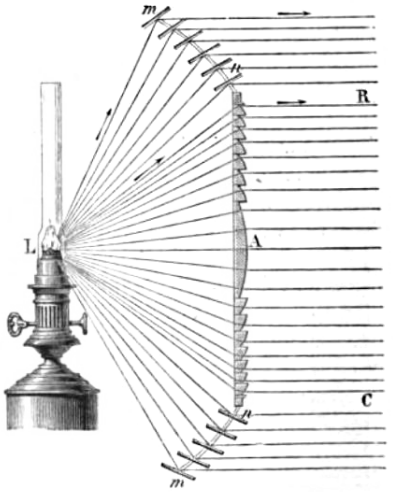


Figure 2.3: How a Fresnel lens focus light. From [Wik12c]

Description	Characteristic	Chart Abbreviation
Alternating		Alt. R.W.G.
Fixed		F.
Flashing		Fl.
Group flashing		Gp Fl.(2)
Occulting		Occ.
Group occulting		Gp Occ(3)
Quick flashing		Qk.Fl.
Very quick flashing		V.Qk.Fl.
Isophase		Iso.
Morse		Mo.(letter)

Figure 2.4: A list of flashing patterns. From [Wik12f]

emit a specific color whereas tungsten sources emit "white" light and use colored filters to get the desired appearance. The tungsten sources are in the process of being replaced with the more power efficient LED sources [Pet12].

2.2 ATON types

Here the ATON light types are explained.

Buoys and beacons These are single sectored omni-directional light sources with a vertical profile that focuses the light horizontally (figure 2.2 and 2.1). Beacons are stationary light sources, usually placed on the coast line and buoys are floating, anchored with a concrete block. Wind and water waves combined with the vertical profile will cause buoys to have a varying intensity when the observer position is fixed.

According to the Danish Maritime Authority [Pet12], the lights are controlled by a photometer that turns the light off in daylight to conserve energy, making daylight appearance less important for this project.

Light Houses Some light houses, such as PEL light houses, have sharp sectors, but usually intensity falls off over a few degrees and neighbor sectors overlap. This gradual change between sectors is used by navigators to control the ship course by interpreting the color variance [Pet12].

A light houses usually has at least three sectors: Green, white and red. Navigators should set a course where only the white light is seen. If the green or red sector is visible then the course should be adjusted starboard or port.

Nautical charts show the position, sectors and nominal range for charted light houses (see figure 2.5).

Signal Lights on Ships There are many rules for light setups on ships for different ship classes and situations [Sø07]. A standard navigation setup under cruise for ships longer than 20m, is shown in figure 2.6. For shorter ships, the side and rear lights may be combined to a single light with three sectors.

In addition to the standard setup, the larger ships have a “Christmas tree” of signal lights on the top of the mast that can communicate different operational states.

In general, the signal lights can be sectored with either 112.5°, 225°, 135° or 360° horizontal angles. Sharply sectoring the light emission (and keeping an uniform intensity over the whole sector) is not practical so the intensity at the

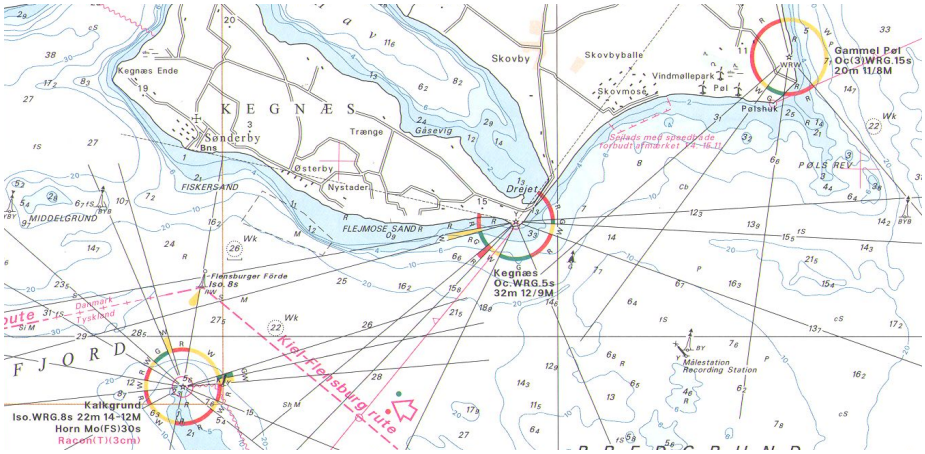


Figure 2.5: Scanned cutout from a nautical chart near Sønderborg, Denmark. Shows light house sector angles and colors. White sectors are shown as yellow.

sector boundaries are allowed to fall off over a few degrees. For the red and green light in figure 2.6, the overlap is regulated such that the intensity is "practically zero 1° to 3° outside the sectors." [Sø07].

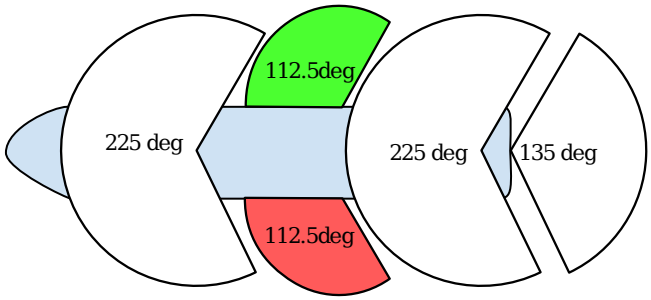


Figure 2.6: A setup of sectored lights on a ship longer than 20m. From [Sø07]

2.3 Real-life Examples

In Denmark Sabik lanterns are almost exclusively used for buoys and beacons [Pet12]. For this project I have used the Sabik VP LED as reference (shown in figure 2.7).

The data sheet reports the following luminous intensities: Red at 120 cd, green at 180 cd, white at 250 cd and yellow at 100 cd[1]. It has a narrow horizontal angular emission profile with 50% peak intensity at 10° and 10% peak intensity at 20°.

Figure 2.7: Sabik VP LED marine light

According to IALA E200-2 the Sabik VP will at night have a nominal range of 6 (108cd to 203cd) nautical miles for red, green and yellow and 7 (204cd to 364cd) nautical miles for white. This is consistent with the ranges (2-6 nautical miles) given by Sabik.

Lights should then be visible 11-13 km away (11.000-13.000 units in simulation) at night.

[1] http://sabik.com/images/pdf/marinelanterns_vpled.pdf

Chapter 3

Background

To solve the problem at hand, some background knowledge is needed. This chapter covers the background for rendering the glare patterns and the color of navigation aids.

A good textbook such as [AMHH08] gives a more detailed description of the theory and simplifications behind real-time rendering.

3.1 Radiometry

Light sources used as navigation aids radiate energy and efficient sources radiate most of their energy in the part of the electromagnetic spectrum that the HVS can perceive, roughly from 380nm to 780nm, called the visible spectrum, see figure 3.1. Examples of spectra for light sources based on tungsten filament and LED are shown in figure 3.2.

Radiometry is the science of the measurement of electromagnetic radiation. The quantities and their units are shown in table 3.1.

Figure 3.3 visualizes radiant flux, radiant intensity and irradiance from a single point source.

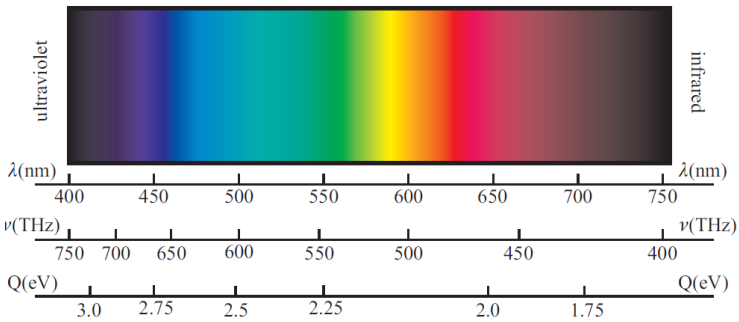
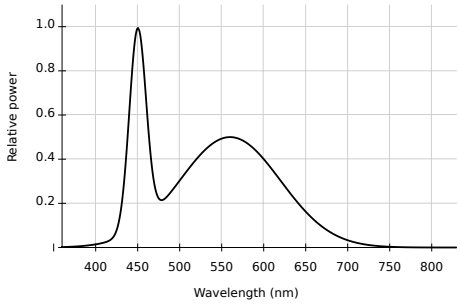


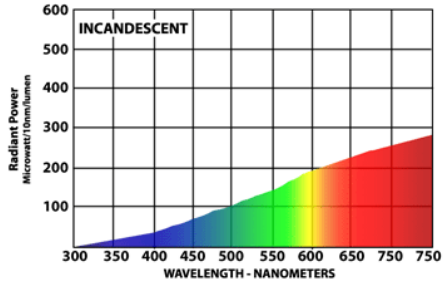
Figure 3.1: Colors of the visible spectrum. From [AMHH08]

Quantiy	Unit	Symbol
Radiant energy	joule (J)	Q
Radiant flux	watt (W)	Φ
Irradiance	W/m^2	E
Radiant exitance	W/m^2	M
Radiant intensity	W/sr	I
Radiance	W/m^2/sr	L

Table 3.1: Radiometric SI Units



(a) Gaussian approximated White LED. After [Wik12d]



(b) Incandescent

Figure 3.2: Spectra for two light sources. (a) shows the narrow peaked spectrum for a LED and (b) shows the broad spectrum for a white incandescent source.

The *radiant flux* of a light bulb is its power consumption multiplied by its efficiency.

For describing point light sources, the *radiant intensity* measures the radiant flux per solid angle. For isotropic point sources (sources that radiate equally in all directions), the radiant intensity is

$$I = \frac{\Phi}{4\pi} \tag{3.1}$$

When computing the radiant flux incident on a differential surface, the quantity is called *irradiance* and the radiant flux exiting a surface is called *radiant exitance* (or *radiosity*). This quantity is relevant for computing the color of the light source surface. The irradiance perpendicular to the light direction at distance d decreases according to Kepler's inverse-square law of radiation (see figure 3.4):

$$E_L \propto \frac{1}{d^2}$$

In computer graphics, a very useful abstraction is transporting radiant flux along infinitely thin rays. Such a ray covers an infinitely small area and an infinitely small solid angle and the quantity is called *radiance*. Radiance is constant along the ray (assuming vacuum) from point to point. In rendering (without multisampling), each pixel in the image plane contains radiance sampled with one ray from the camera position through the pixel and the sample is assumed representative of the whole pixel.

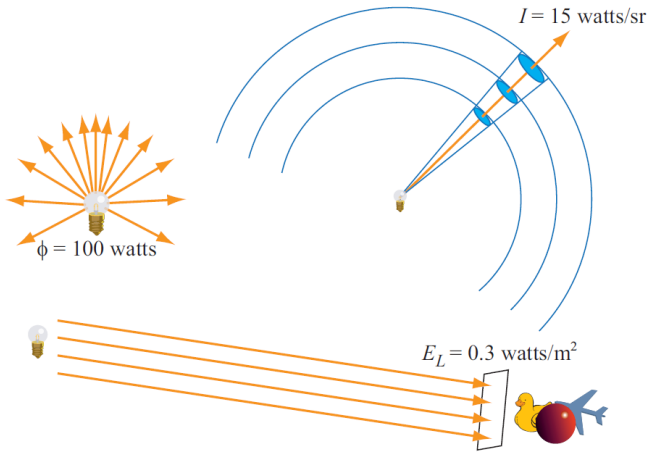


Figure 3.3: Measures of a light bulb in different units. From [AMHH08]

Isotropic light source The radiant exitance M through the surface of an isotropic light source with radius r is

$$M = \frac{d\Phi}{dA} = \frac{4\pi I}{4\pi r^2} \tag{3.2}$$

The for a point on the source, the radiant exitance can also be computed by integrating the isotropically emitted radiance over the hemisphere:

$$M = \int_{\Omega} L_e \cos\theta d\omega = \pi L_e \tag{3.3}$$

By combining and rearranging equations 3.2 and 3.3, the emitted radiance L_e on the surface of an isotropic light source with radiant intensity I is then

$$\begin{aligned} \frac{4\pi I}{4\pi r^2} &= \pi L_e \\ L_e &= \frac{I}{\pi r^2} \end{aligned} \tag{3.4}$$

When the light source covers a fraction of a pixel, the assumption that the radiance from the light source surface can be sampled as a full pixel breaks. In this case, the incident radiance follows the Inverse-square law (figure 3.4) and is given by

$$L_i = \frac{I}{d^2} \tag{3.5}$$

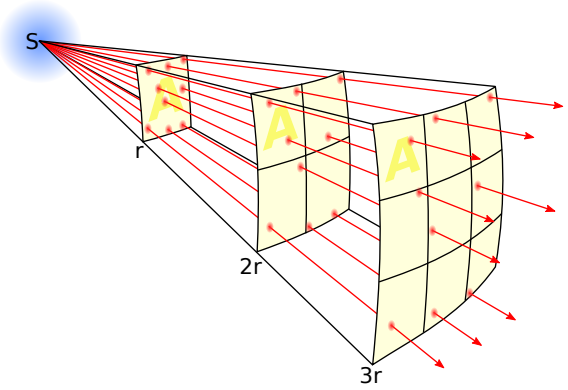


Figure 3.4: Kepler's inverse-square law. The density of the of flux lines decreases with the inverse-square law so that at distance $3r$ the density is 9 times smaller than at distance r. From [Wik12e]

3.2 Photometry

Recall the Sabik VP LED from section 2.3 where it's brightness was stated in the SI unit candela (cd). This is a photometric quantity called *Luminous Intensity* which corresponds to radiant intensity.

Photometry is the science of measuring brightness perceived by the HVS. Each of the radiometric quantities has a corresponding photometric quantity (listed in figure 3.2) that is weighted against the luminosity function. Figure 3.5 shows two luminosity functions: the photopic $V(\lambda)$ for daylight adaptation and scotopic $V'(\lambda)$ for dark vision. The $V(\lambda)$ is most sensitive to green light peaking at

Quantity	**Unit**	**Symbol**	**Radiometric**
Luminous energy	lumen-second ($\mathrm{lm} \cdot \mathrm{s}$)	Q_v	Radiant energy
Luminous flux	lumen (lm)	Φ_v	Radiant flux
Illuminance	lux ($\mathrm{lm/m^2}$)	E_v	Irradiance
Luminous emittance	lux ($\mathrm{lm/m^2}$)	M_v	Radiant exitance
Luminous intensity	candela ($\mathrm{cd} = \mathrm{lm/sr}$)	I_v	Radiant intensity
Luminance	nits ($\mathrm{cd/m^2}$)	L_v	Radiance

Table 3.2: Photometric SI Units

560nm, which is why green lights need less power than red or blue lights for the same perceived brightness.

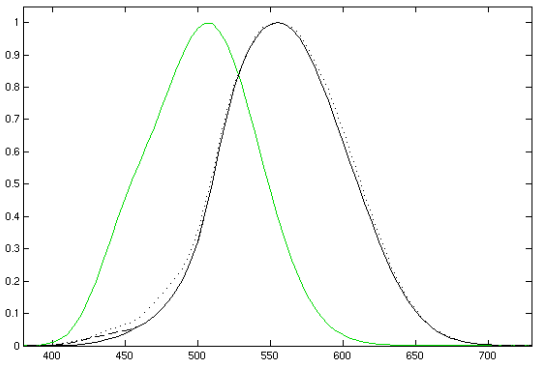


Figure 3.5: Photopic (black curve) $V(\lambda)$ and scotopic $V'(\lambda)$ (green curve) luminosity functions. From [Wik12g]

Converting a radiometric to photometric quantity will "flatten" the spectrum into one brightness value by

$$L_v = \frac{1}{683} \int_{380}^{830} L_\lambda V(\lambda) d\lambda \tag{3.6}$$

Representative luminance values are shown in table 3.3 to give an overview of the dynamic range an outdoor simulation should be able to handle.

Condition	**Luminance** (cd/m^2)
Sun at horizon	600 000
60-watt light bulb	120 000
Clear sky	8 000
Typical office	100 - 1 000
Typical computer display	1-100
Street lighting	1-10
Cloudy moonlight	0.25

Table 3.3: Representative luminance values. From [PH04]

3.3 Colorimetry

The recommended colors of navigation aids are defined in colorimetric terms.

The following description of colorimetry is based on [RWP+10].

Experiments show that almost all perceivable colors can be generated using a combination of three suitable pure primary colors. The science of quantifying and specifying the HVS perception of color is called *colorimetry*. Using colorimetry, we can assign a tristimulus color to the spectral power distribution emitted by navigation aids (examples shown in figure 3.2). Allowing the color of the full electromagnetic spectrum to be represented as three scalars also makes rendering much more efficient.

3.3.1 Color matching functions

From experiments using three primaries (red, green and blue), three curves were defined $\bar{r}$, $\bar{g}$ and $\bar{b}$ for a "standard observer" called the *CIE 1931 RGB color matching functions* (shown in figure 3.6).

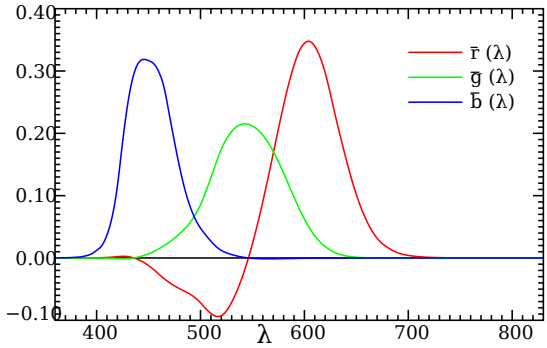


Figure 3.6: The CIE 1931 RGB color matching functions. The curves show the power of the three primaries that will generate the color hue at a given wavelength. From [Wik12b]

The curves peak at $(\lambda_R, \lambda_G, \lambda_B) = (645.2\text{nm}, 525.3\text{nm}, 444.4\text{nm})$ and some combinations require negative amount of power (thus not realizable, light cannot be subtracted).

A linear combination of three scalars R, G, B and the color matching functions

define the spectral color stimulus $C(\lambda)$.

$$C(\lambda) = \bar{r}(\lambda)R + \bar{g}(\lambda)G + \bar{b}(\lambda)B$$

The (R, G, B) triplet will then be the tristimulus values of $C(\lambda)$.

Three idealized primaries (X, Y, Z) whose color matching functions $\bar{x}$, $\bar{y}$ and $\bar{z}$ are all positive can be used as a neutral basis.

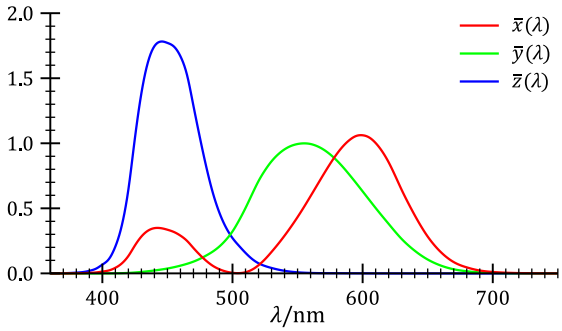


Figure 3.7: CIE Color matching functions. From [Wik12b]

$$C(\lambda) = \bar{x}(\lambda)X + \bar{y}(\lambda)Y + \bar{z}(\lambda)Z \tag{3.7}$$

To convert the spectral stimulus $C(\lambda)$ to tristimulus values (X, Y, Z):

$$\begin{aligned} X &= \int_{380}^{830} C(\lambda)\bar{x}(\lambda)d\lambda \\ Y &= \int_{380}^{830} C(\lambda)\bar{y}(\lambda)d\lambda \\ Z &= \int_{380}^{830} C(\lambda)\bar{z}(\lambda)d\lambda \end{aligned} \tag{3.8}$$

The CIE XYZ color matching functions are defined such that a theoretical equal energy source with radiant power of one for all wavelengths maps to tristimulus value $(1, 1, 1)$. The $\bar{y}$ function corresponds to the photopic luminosity function $V(\lambda)$ and so the Y stimulus is the photometric response. Very different spectra can resolve to the same tristimulus triplet (i.e. the same color). This is called *metamers*.

From the tristimulus value (X, Y, Z), the *chromaticity* coordinates can be derived.

$$\begin{aligned} x &= \frac{X}{X+Y+Z} \\ y &= \frac{Y}{X+Y+Z} \\ z &= \frac{Z}{X+Y+Z} = 1-x-y \end{aligned} \tag{3.9}$$

As z can be derived from x and y usually only x and y are stored.

To be able to restore the original XYZ value from xy-chromaticity, the luminance Y has to be stored as well (xyY). Converting xyY back to XYZ is done using the following transform:

$$\begin{aligned} X &= \frac{Y}{y}x \\ Y &= Y \\ Z &= \frac{Y}{y}(1-x-y) \end{aligned} \tag{3.10}$$

3.3.2 Color spaces

An XYZ triplet is a *device independent* color specification where colors can be specified independent of the actual primaries of a given output device. By not taking the actual primaries into account, the colors cannot be directly displayed. Using knowledge about the primaries and white point of the particular output device, a 3×3 matrix can be constructed that converts the XYZ triplet into *device dependent* RGB color (inverting the matrix will give the inverse mapping).

Historically, displays (e.g. CRT displays) had a non-linear relationship between input voltage and output luminance that can be approximated by a power law ($L \propto V_{in}^{\gamma}$), which means that if the RGB input in $[0, 1]$ is linear, then it needs to be pre-corrected with the inverse gamma $(1/\gamma)$ value of the display. This process is called *gamma correction*. LCD displays do not have this non-linear property, but the power law is usually applied for sake of compatibility [Ngu07, chapter 24].

As a consequence, linear RGB values had to be pre-adjusted with the inverse power law curve to compensate.

$$(R', G', B') = (R, G, B)^{1/\gamma}$$

As the gamma correction is done to counter the power law of the display device, it should be the very last step before displaying the rendered image.

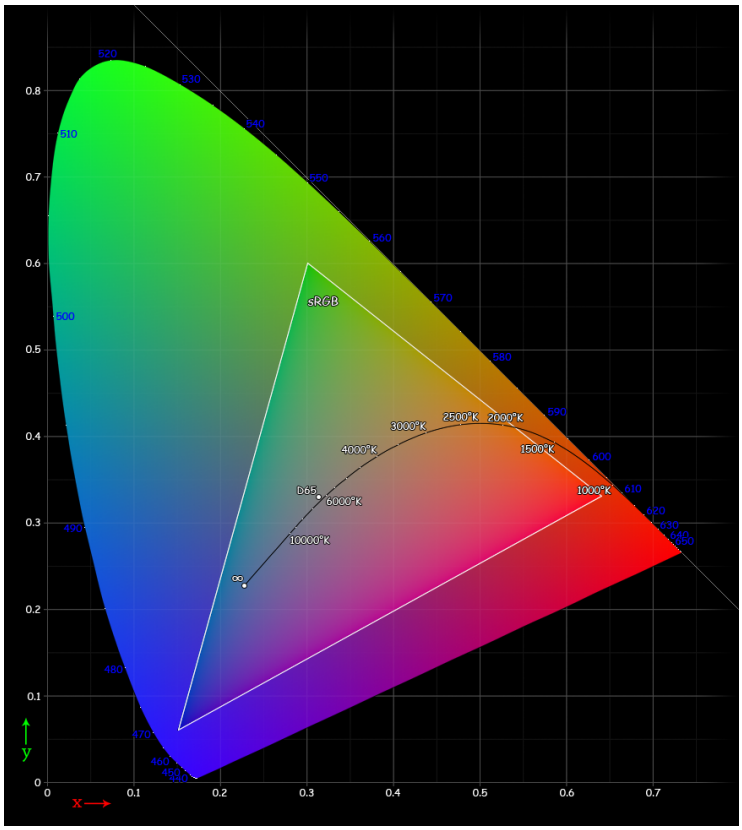


Figure 3.8: CIE chromaticity plot of sRGB primaries. From [Wik12h]

Figure 3.8 shows a CIE chromaticity plot of x and y and three primaries of the sRGB (also called Rec. 701 [Rec90]) color space. The triangle that contains the hues realizable by the primaries is called the *gamut* and colors outside are called out-of-gamut colors which cannot be correctly represented by the color space. The bounding curve is the hues of pure monochromatic colors at a given wavelengths, the "spectral locus".

The sRGB color space is designed to fit most displays (though most consumer displays are not calibrated) to ensure a certain subset of colors (i.e. the colors inside the triangle) are the same on different displays.

FORCE uses sRGB projectors so the final render output should be in sRGB

color space.

The white point of a color space, which defines the chromaticity of white, is defined by the Spectral Power Distribution (SPD) of an *illuminant*. For sRGB, the illuminant is D65 (shown in figure 3.9) which corresponds roughly to the SPD of the midday sun in western / northern Europe. Integrating the SPD using equation 3.8 yields the white point chromacities of table 3.4. Normally only the white point chromaticities - and not the full spectrum - are used, but the scattering of light in the eye is wavelength dependent.

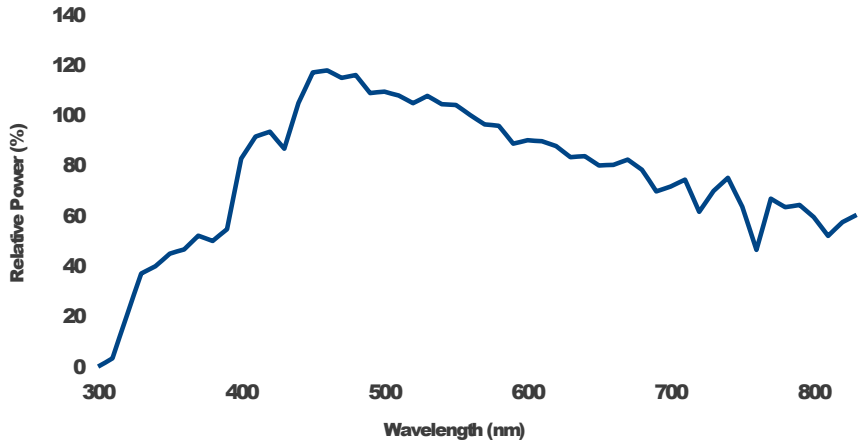


Figure 3.9: The Relative Power Spectrum of Illuminant D65. The spectrum is normalized to 100 at 560nm at the peak of the photopic luminosity function.

When discussing colors, the chromaticity plot allows comparing the realizable colors of different media (e.g. paper, LCD displays and projector displays).

The IALA recommendations specify that red, green, blue, yellow and white colors can be used for navigations aids and specifies regions on a CIE xy-chromaticity chart (figure 3.10). Unfortunately red and yellow lies outside the common sRGB color space.

A tristimulus color space can be converted to another tristimulus color space using a 3×3 matrix and back using the inverse matrix.

For the color space defined by International Telecommunication Union as ITU-R

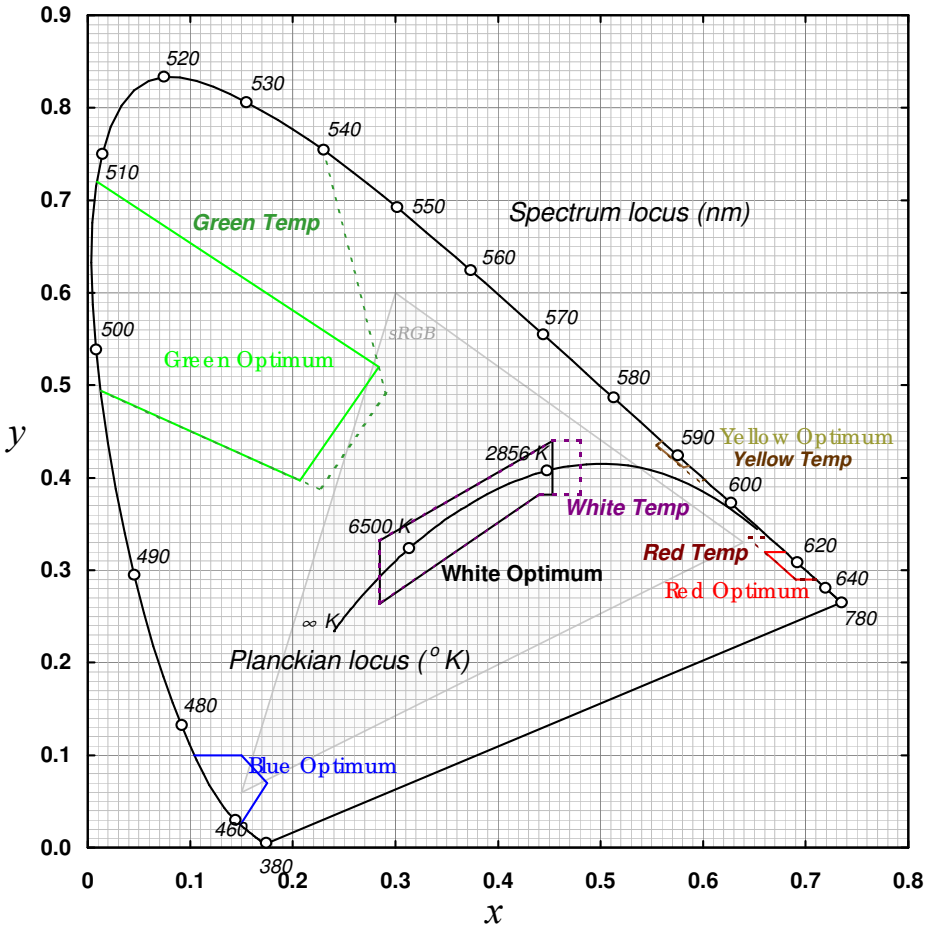


Figure 3.10: IALA Recommended Color xy-chromaticity Color Regions for Marine Lights. Note that the red and yellow regions are outside the sRGB gamut. From [IAL08a]

	Red	Green	Blue	White
x	0.6400	0.3000	0.015	0.3127
y	0.3300	0.6000	0.060	0.3290

Table 3.4: CIE xy-chromacities for sRGB primaries and white point

Recommendation BT.709 - also called sRGB[Rec90] - the conversion matrices are:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.11}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.2405 & -1.5371 & -0.4985 \\ -0.9693 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0572 \end{bmatrix} \begin{bmatrix} Z \\ Y \\ Z \end{bmatrix} \tag{3.12}$$

The non-linear transformations approximates the gamma 2.2 curve with

$$f(x) = \begin{cases} 1.055x^{1/2.4} - 0.055 & \text{for } x > 0.0031308 \\ 12.92x & \text{otherwise} \end{cases}$$

$$\begin{bmatrix} R_{sRGB} \\ G_{rRGB} \\ B_{sRGB} \end{bmatrix} = \begin{bmatrix} f(R_{linear}) \\ f(G_{linear}) \\ f(B_{linear}) \end{bmatrix} \tag{3.13}$$

sRGB hardware support Graphics hardware has direct support for sRGB encoded textures and have automatic sRGB output encoding, which means that the rendering output can be linear RGB and then the hardware will apply the non-linear transformation. This means that the hardware will decode non-linear sRGB textures and framebuffers when executing blending operations and texture lookups, allowing the shaders to work with linear values.

3.4 Human Visual System

The HVS is a complex system with many components that each influence our visual perception.

When light enters through the pupil and hits the retina at the back of the eye, the photo-receptors receive input and they send it to the brain through the optic nerve. There are two kinds of photoreceptors: *rods* and *cones* [RWP+10]. At low illumination levels the rods are responsible for our monochromatic night vision (the illumination received by the rods is perceived as bluish [JDD+01] and the effect is also known as *blueshift*). In higher illumination, the three cone types (long, medium and short wavelength) give us trichomatic color vision.

The neural photoreceptor response R to stimulus I can be modeled by the "Naka-Rushton" equation

$$\frac{R}{R_{max}} = \frac{I^n}{I^n + \sigma^n} \tag{3.14}$$

where σ is called the semi-saturation constant and n is a sensitivity control constant between 0.7 and 1 [RWP+10]. The equation forms an S-curve on a log-linear plot and appears "repeatedly in psychophysical experiments and widely diverse, direct neural experiments"[RWP+10].

A plot of the relative rod and cone response is shown in figure 3.11. Note that for a few orders of magnitude, the response (perceived brightness) is roughly logarithmic (the straight line part on the log-linear plot) and the sigmoid curve also explains a maximum simultaneous range of around five orders of magnitude.

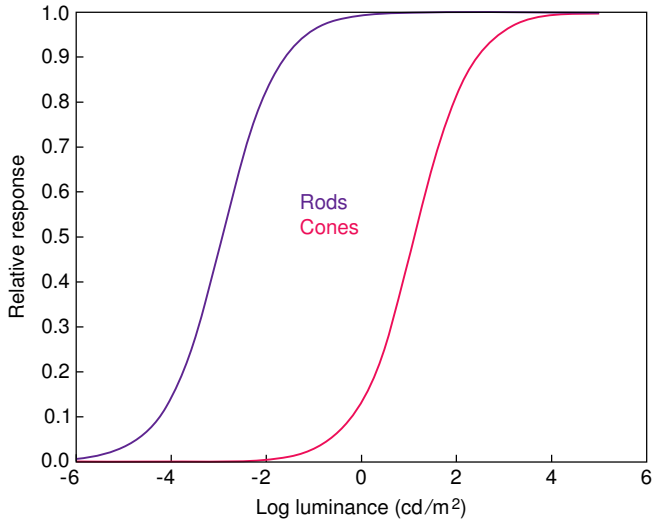


Figure 3.11: The relative response for rods and cones modeled by Equation 3.14 by shifting the semi-saturations constant σ. From [RWP+10]

Temporal adaptation Over time, the HVS adapts to changing background intensities. Adapting to brighter background intensties (light adaptation) is a relatively fast process whereas dark adaptation is much slower. Pattanaik et al. [PTYG00] gives a model for both dark and light temporal adaptation for rods and cones. The HVS is able to adapt locally to regions of different background intensities which enhances local contrast perception. The adaptation can be seen as modifying the semi-saturation σ, moving the response curve towards the background intensity.

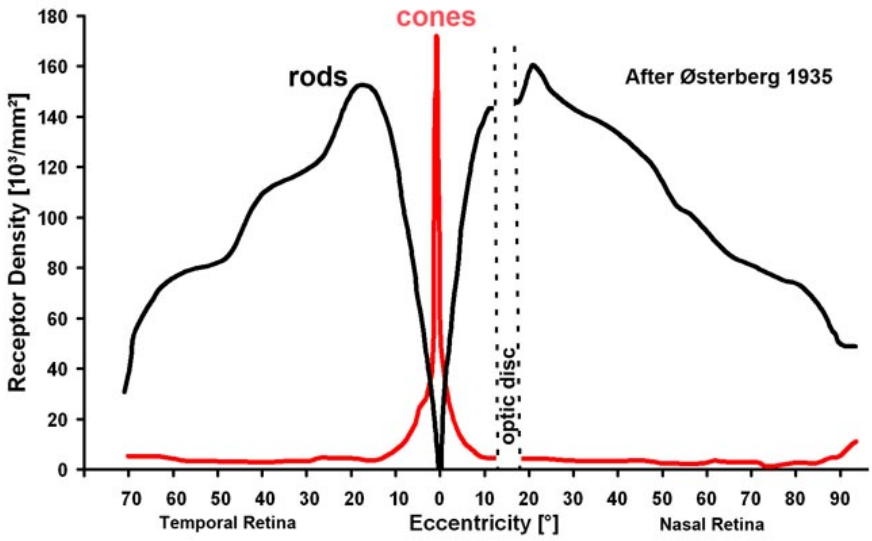


Figure 3.12: The distribution of rods and cones. Webvision, [KFNJ04]

Distinguishing details Due to the distribution of rods and cones (figure 3.12, daylight adaptation happens primarily in the central 2° of the visual field (and *visual acuity* - the ability to separate details - is highest there) [KFNJ04]. At night when the illumination is lower than the cones' sensitivity, the situation actually changes and the fovea is practically blind. Another interesting fact is that at night, the sensitivity of the rods (see the scoptopic luminosity curve in figure 3.5) is not sensitive to the high red wavelengths so a dark adapted human can use red lighting without loosing the scotopic dark-vision of the rods.

Contrast sensitivity Another way to model perception is through Just Notable Difference (JND) which is the intensity difference ΔI to background intensity I_b needed by the HVS to perceptually notice the change.

To perceive an ATON light as different from the background, the HVS contrast of the light intensity to background intensity needs to be above the JND.

By measuring the ΔI for a wide range of intensities, we can estimate the Threshold-versus-Intensity (TVI) (measured data is shown in figure 3.13).

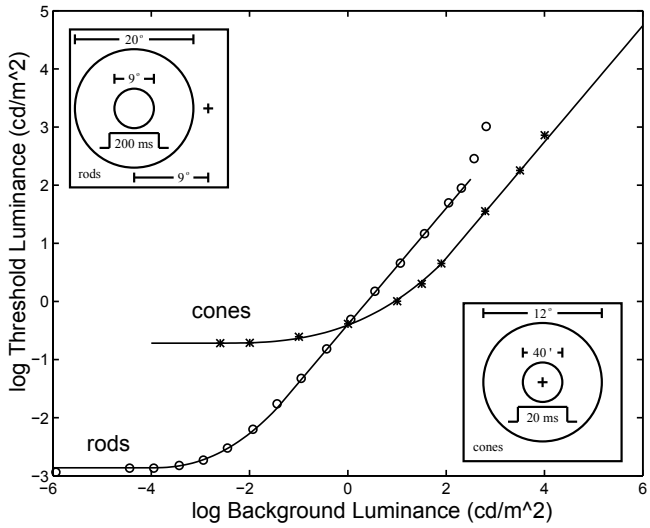


Figure 3.13: Threshold-versus-intensity curves for rods and cones. From [FPSG96]

Chromatic adaption The HVS will adapt perceived colors to the dominant light source. Gradually changing the color of a light source (e.g. with change in light source temperature), will not change the perceived color chromaticity. This is relevant for the lighting in the room of the display.

3.4.1 Light Diffraction and Scattering in the eye

The glare phenomenon mentioned in the introduction is caused by internal scattering in the eye. For ATON lights, the glare allow us to perceive the color and position of a light when the actual emissive part of the light is too small (i.e. when the light is far away from the observer) to distinguish from the background.

In his studies of halos, Simpson found that the glare appears as thin, radial needle-like lines, the *ciliary corona*, which appear when when the source of the light source covers less than 20 minutes of arc (20/60°) [Sim53] of the visual field. A quadrant of the glare from a white source found by Simpson is shown in figure 3.14. He found the upper radius of the *lenticular halo* to be around 4°.

A schematic over scatterers in the eye is in figure 3.15. According to [RIF+09] and [Sim53], the majority of the scattering happens in the lens and cornea.

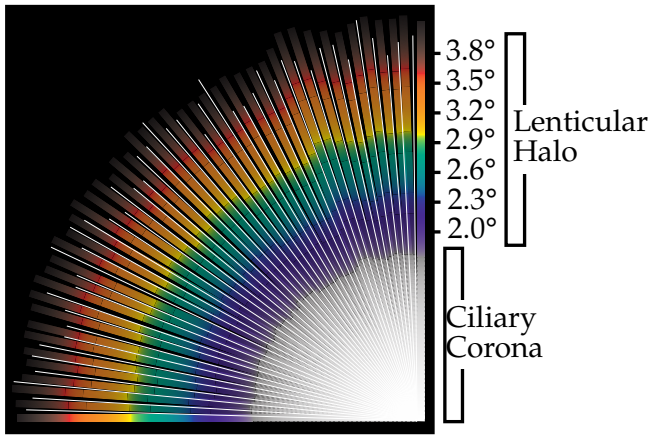


Figure 3.14: The lenticular halo and ciliary corona from a small white point source, with the radius in visual angles. From [SSZG95]

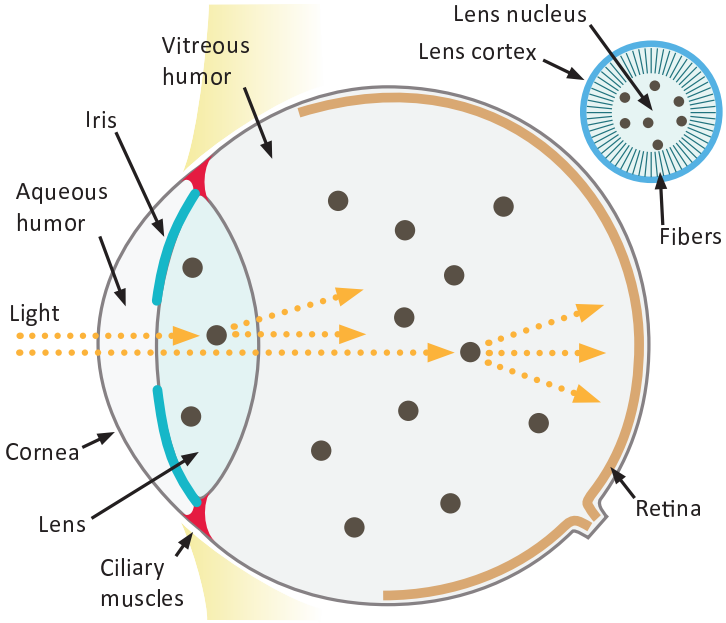
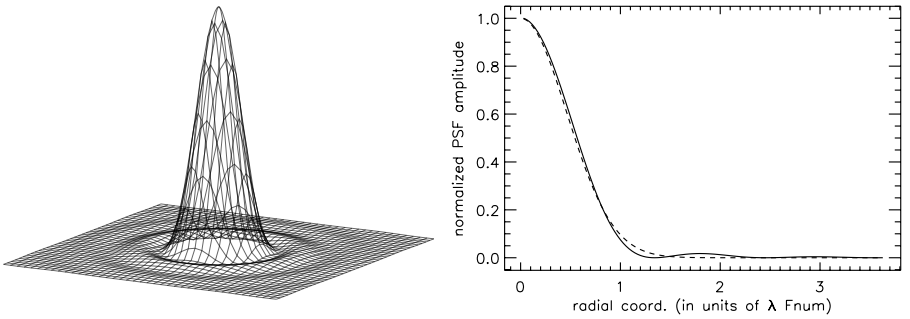


Figure 3.15: Anatomy of a human eye. The inset in the upper right corner shows the structure of the lens. From [RIF+09]

The pupil, lens and vitreous particles along with the eye lashes are dynamic scatterers that change the glare pattern over time, causing a fluid-like, pulsating look [RIF+09].



(a) Surface plot **(b)** Gaussian approximation

Figure 3.16: Airy diffraction pattern from a circular aperture. From [Wik12a]

Every aperture diffracts incoming light waves, causing the light at one point to be distributed to nearby points, this is called a Point-Spread Function (PSF). The PSF actually distributes light to all other points, but it falls rapidly off with distance. Perfect circular apertures distribute light in Airy patterns (figure 3.16). The general shape of the Airy pattern can be approximated with a Gaussian curve which is separable and efficient on graphics hardware. This simple approach was used by [Kaw05] and [KMS05] among others to model glare.

The amount of light entering the eye is determined by the size of the pupil (and of course the eye lids). At low illumination levels the diameter increases to gather more light and vice versa at higher levels. The diameter ranges from 3mm to 9mm and can be described as a function of average scene luminance [WS67]:

$$p = 4.9 - 3\tanh(0.4(\log L_a + 1)) \tag{3.15}$$

where L_a is average adaptation luminance (mentioned as field luminance in [RIF+09]). I assume for simplicity that the light adaptation luminance for the pupil size and the retinal light adaptation luminance (for both rods and cones) are equal.

Simple Glare Test I discovered a simple method to test if the glow around a light source is caused by the eye: Look at the light source with one eye (and

see the glow), then eclipse the light source with an object smaller than the glow (e.g. a finger). If the glow disappears, then it was caused by scattering in the eye, otherwise the glow phenomena is caused by scattering in the atmosphere (e.g. scattered by particles in dense fog or moonlight scattered by clouds).

3.5 Tone mapping

As mentioned in the introduction, perceptual tone mapping is an active research area where most of the recent publications employ sophisticated methods to enhance details. A comprehensive, comparative study of the different methods applied for real-time simulations is not possible given the time constraints of this project.

Natural night scenes do not usually have high contrast [JDD+01]. Adding ATON will usually not increase background contrast because of the vertical emission profiles (and not at all when disregarding light interaction between the ATON and the environment as mentioned in section 1.1). Contrast between background intensity and foreground intensity (e.g. the lights) can be high /(several orders of magnitude) and the tone mapper should be able to handle that in a consistent way.

Ideally, the simulation would output and display the simulated intensities directly and let the HVS adapt. This would give a perceptually real experience to multiple simultaneous viewers. Unfortunately this is not yet practical so the simulated dynamic range has to be reduced and several aspects of the HVS have to be approximated.

Figure 3.17 shows the high level tone mapping process for captured HDR data. For this project, the HDR data is synthetic.

Classification There are many approaches to tone mapping so to ease discussion and comparison a simple classification inspired by [IFM05]:

Global The same tone map function (which maps world intensities to display intensities, for instance a variation of equation 3.14 visualized figure 3.11) is used for all pixels relying on image global parameters. Usually relies on a global average background intensity. The global nature might cause a loss of details for high contrast scenes.

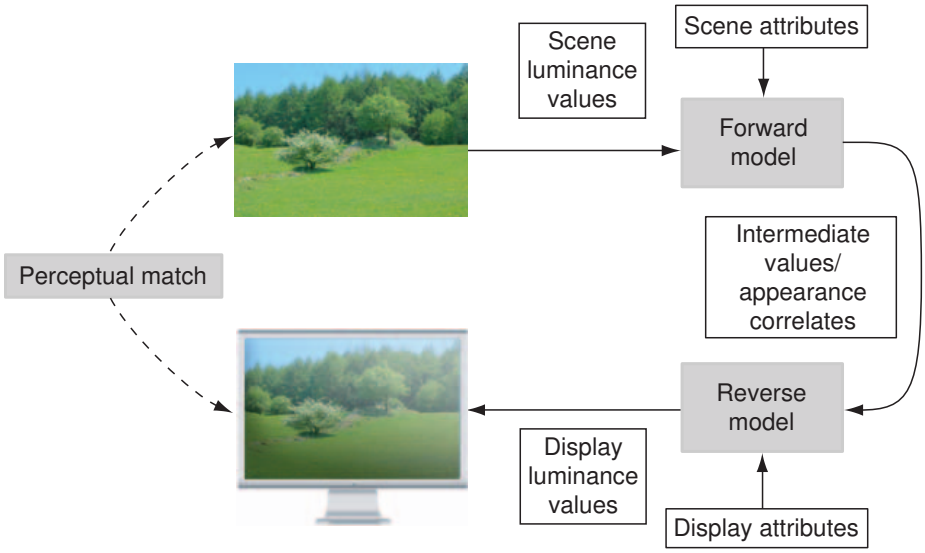


Figure 3.17: High level overview of the tone mapping problem for photographs. From [RWP+10]

Local spatially varying tone map function that uses local informations such as luminance of neighboring pixels, usually to give a better estimate of background intensity.

Static Temporal adaptation to background intensity is not taken into account. Primarily for photographs and usually have user-parameters that needs to be adjusted per image.

Dynamic Opposite to static. Usually the whole tone map process is fully automatic without essential user-parameters.

Perceptual Operator based on psychophysical models for perception such as TVI or equation 3.14.

Empirical Methods not directly based on psychophysical models and more concerned with dynamic range compression, detail enhancements, a desired artistic expression. Tone mapping in games are usually fully empiric.

For a ship simulator, all potential operators will have to be *dynamic* to cope with day-night cycle and wide range of intensities (e.g. sudden cloud cover, etc.). High quality *local* operators generally carry a heavy performance hit whereas *global* operators map well to GPU hardware.

A *perceptual* operator is desirable when rendering ATON lights in a ship simulator to ensure that visibility is preserved on the output medium, but *empirical* can be simple and can work well in practice.

Approaches to tone mapping To properly preserve visibility, the environment, display and adaptation of the observer has to be considered (see Display Adaptive Tone Mapping by Mantiuk et al. [MDK08]). Night illumination may also be less than the black level of the display, thus presenting a scotopic scene to a photopically adapted viewer.

The projector setup at FORCE, shown in figure 1.2, has the edges of neighboring projectors overlap. This doubles the black level in the overlapping region and to ensure visual consistency, the black level across the whole screen is artificially increased to match. This effectively cuts the contrast ratio in half.

As the HVS can adapt to different background illumination levels, a useful operator will have to determine an adaptation value. It can be global or local.

A simple tone map operator (TMO) inspired by the photoreceptors is given by [RD05]. It builds upon equation 3.14 for dynamic range reduction and estimates the semi-saturation constant as a function of average background intensity and has two user parameters for brightness and contrast, f and m. The method is as presented *static* and *perceptual*. The authors propose computing m from the log-average, minimum and maximum luminances, but still f is a free user parameter which makes the method incomplete as is for use in this project so it does not fully qualify as *dynamic*.

When the contrast is not too high, a single global scaling factor based on the TVI functions can be used to map scene luminances to display luminances [War94, FPSG96, DD00]. Such linear scaling factors (though empirical instead of the TVI curves, analogous to setting exposure in cameras) are also used in computer games (Source engine [MMG06, Vla08], Black and white 2 [Car06]) and using a sigmoid S-curve inspired by the film industry to enhance colors (Uncharted 2 [Hab10]). In games, the dynamic range is controlled by artists and can be kept in a reasonable range where global operators perform well.

Slightly different approaches to scotopic blueshift are covered in [JDD+01, KMS05, DD00].

A cumulative histogram can be used as a tone mapping curve to assign display levels to scene intensities. This is called *histogram equalization*. Constraining the histogram to the contrast sensitivity of the HVS was done in [LRP97].

In games, the tone-mapping is directed by artists and the dynamic range can be controlled. For instance the Source engine from Valve uses a linear mapping with a single global scalar driven by a 16 bin histogram [Vla08]. Recently games have begun to use the same curves as used in film productions [Hab10] to get more saturation for darker colors.

The photographic TMO by Reinhard et al. [RSSF02] is an empirical method inspired by photography. The method has a global and a local step. The first is a linear mapping where the average luminance is mapped to "middle grey" dependent on the "key" of the scene. Bright scenes are "high key" and dark scenes are "low key". The second step is locally enhancing the contrast based on local averages. For real-time rendering, the second step is usually omitted for performance reasons [AMHH08, Luk06, EHK+07].

Krawszyk et al. [KMS05] used the non-perceptual photographic zone-system based operator [RSSF02] as basis for adding real-time perceptual effects to HDR movies. Their process is fully automatic and though they sacrifice performance for a local contrast enhancement their approach seems viable for real-time use. The local parts and their GPU implementations have shown to be very expensive. They are interactive, but not real-time for high resolutions [KMS05, RAM+07, GWWH05].

For high contrast scenes, global operators can cause a loss of detail in regions with low variations. Bilateral filtering can be used to separate the image into a HDR base layer and a LDR detail layer [DD02]. The base layer can then be tone mapped using a global operator and details can be restored by adding back the detail layer. The filtering is complex and naive implementations would carry a heavy performance hit.

The iCAM06 TMO [KJF07] uses the bilateral filter and models perceptual effects such as loss of contrast and color at scotopic illumination levels using advanced color appearance models. It has good results, but the method is too complex with many constants, making implementation too risky for this project.

This project builds primarily upon [DD00] for a perceptual and [RSSF02] for an empirical tone mapping approach. The method is described in section 4.

3.6 Building blocks

3.6.1 ATON Environment

The simplest scene that gives visual context to the rendering of ATON lights has water and sky. This project uses SilverLining from SunDog software to render the atmosphere and clouds.

Simple water can be shaded using the Blinn-Phong shading equation from [AMHH08] based on the half-vector between light direction and view direction.

$$f(\vec{l},\vec{v}) = \frac{\vec{c}_{diff}}{\pi} + \frac{m+8}{8\pi} R_F(\alpha_h)\overline{\cos}^m\theta_h \tag{3.16}$$

where α_h is the angle between the light direction $\vec{l}$ and half-vector $\vec{h}$, θ_h is the angle between the surface normal $\vec{n}$ and the half-vector $\vec{h}$, m is the specularity control parameter and R_F is the Fresnel factor. The $\overline{\cos}$ notation means the value is clamped (e.g. using $\max(0, \cos x)$).

I use Schlick's Fresnel approximation given by

$$R_F(\alpha_h) = R_F(0°) + (1 - R_F(0°))(1 - \cos\alpha_h)^5 \tag{3.17}$$

For the water I use $R_F(0°) = (0.02, 0.02, 0.02)$.

3.6.2 GPU pipeline

Rendering high quality real-time, interactive 3D simulations will require dedicated hardware. A block diagram of an OpenGL 3.2 / DirectX 10 class GPU is shown in figure 3.18. All four programmable stages; vertex, geometry, pixel and compute will be used in the implementation. The compute stage is separate from the others and is always standalone. Vertex, geometry and pixel stages together form a pipeline.

Vertex shaders acts on vertex primitive data (such as points, lines and triangles) submitted by the cpu.

If a geometry shader stage is present, it acts on the primitive stream from the vertex shader and is allowed to manipulate the number of primitives emitted for rasterization. This property allows for creating two triangles (a quad) from a point primitive stream, which is used by the method to show the glare effect.

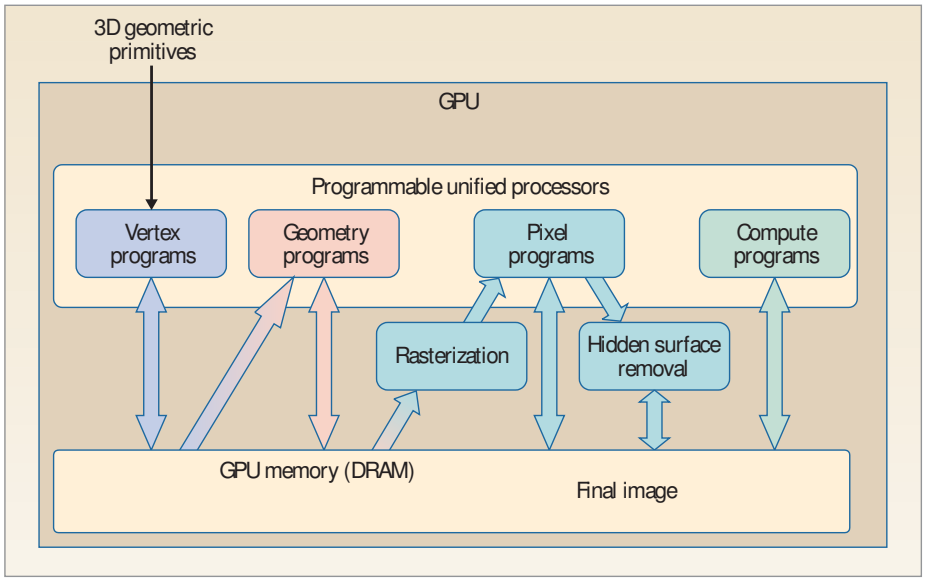


Figure 3.18: A block diagram for OpenGL 3.3/DirectX 10 class hardware. From [LH07].

As the geometry shader is also allowed *not* to emit anything, occluded lights can be culled without CPU intervention.

After the vertex shader or optional geometry shader, the primitives are clipped to the viewport and rasterized and the pixel shader is run for each rasterized fragment.

After pixel shadeing and hidden surface removal, the fragments are blended or copied to the target framebuffer.

Chapter 4

Method

The method as described in this chapter has these components:

- Modeling of navigation aids, explained in section 4.1.
- Rendering the scene and environments.
- Glare pattern generation and rendering, explained in sections 4.2 and 4.3.
- Tone mapping the HDR results, explained in section 4.4

The rendering of the scene and environments are outside the scope of this project, but integration issues are not. Rendering the scene and skybox happens before tone mapping and glare from light sources. Water rendering was briefly explained in section 3.6.1.

Every frame, the following high level steps are executed:

- Run general simulation and update light transforms
- Render scene
- Compute average scene luminance/incident scene luminance

- Compute glare PSF according to adaptation luminance level.
- Draw the fixed-size, screen-aligned glare billboards for visible light sources to the offscreen HDR buffer.
- Compute the log-average luminance for global tonemap operator.
- Tonemap to LDR and output to framebuffer

Figure 4.1 shows a high level overview of the method.

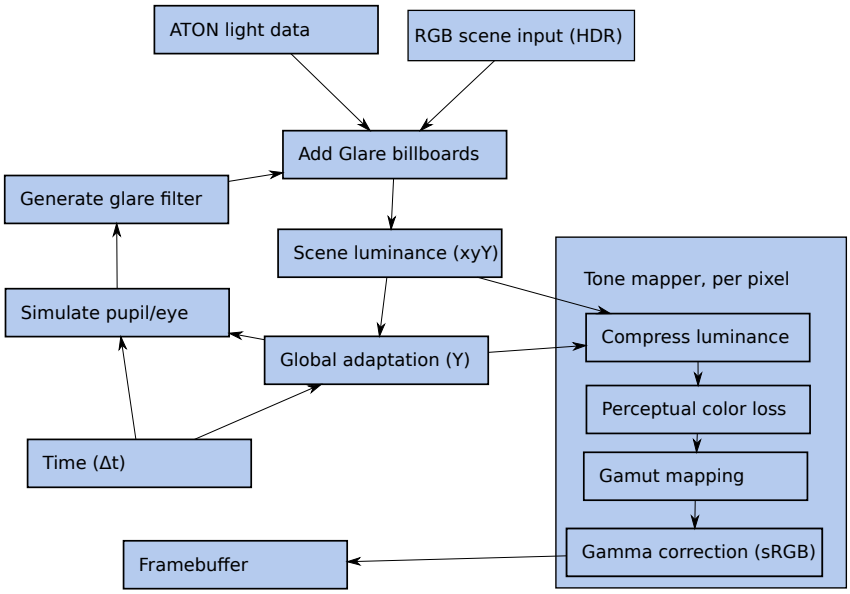


Figure 4.1: A high level overview of the method

Adding glare to a LDR renderer In an LDR renderer such as SimFlex, the scene is implicitly tone mapped and the contrast reduced. Adding glare is then done after general tone mapping and blended additively with the framebuffer, which is not strictly physically correct. The glare billboards are still HDR so they still need to be tone mapped with empirical adaptation luminance for night and day scenes.

Tone mapping the glare separately was done by Kakimoto et al. [KMN+05a]. The glare billboards can be tone mapped and additively blended in the same pass, but rendering all glare billboards to a separate buffer (that is tone mapped and then additively blended with the LDR scene) will allow proper linear blending of the glare billboards if a non-linear TMO is used.

4.1 Modeling of ATON sources

In section 2, the properties of ATON lights were discussed. I make the simplifying assumption that they can be modeled as isotropic spherical light sources where some angles are shielded (Figure 4.2).

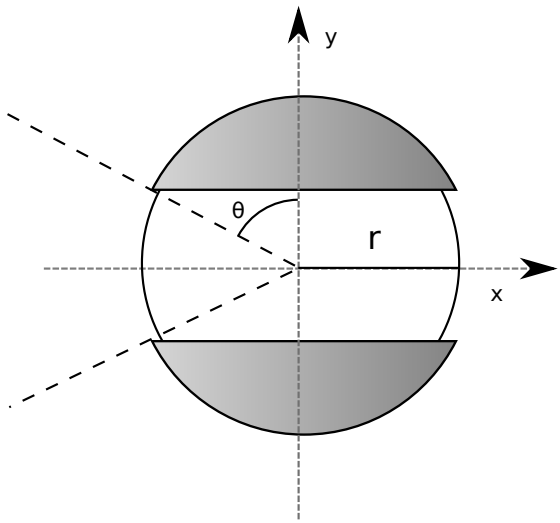


Figure 4.2: The simplified light source model. The masking implicit and modeled by a vertical profile.

The model of the navigation aids consists of the following properties:

Position and orientation Spatial position $\mathbf{x}_l$ and orientation $\mathbf{R}_l$ that can be updated during the simulation for floating buoys on the water waves and the ATON lights on moving ships.

Radius The lights are assumed spherical for computation purposes and this is the radius of that sphere.

Intensity and emission spectrum For performance reasons, the emission spectra for all ATON light sources are assumed to follow the SPD of Illuminant D65 and the color hue defined by the IALA color regions (from figure 3.10) in CIE xy chromaticity coordinates.

The radiant RGB intensity is then derived from the xy chromaticity and a specified luminous intensity by converting the xyY to XYZ using eqn. 3.10 and then to RGB using equation 3.12.

Figure 4.3 shows the xy positions of the sector colors and the approximate linear RGB color values are shown in table 4.1. For blue, white and green colors, IALA chromaticities within the sRGB spectrum are used. Yellow coordinates are chosen on the sRGB triangle boundary on a line straight from middle of the regions towards the D65 white point. Red coordinates are chosen from the Temp regions which can be mapped linearly to the sRGB red primary. If the center point in the optimum red region is chosen, the hue is slightly magenta (which is shown in the results).

Blink profile The blink profile pattern that models the flashing patterns shown in figure 2.4. The profile is tabulated function of time relative to light activation time and models arbitrary turn on and off ramps to approximate the behavior. Blink modulation of intensity is assumed uniform over the light[1].

Vertical and horizontal profile Controls the emission sectoring of a light such as shown in figure 2.2 and 2.1. The vertical profile is a function of the vertical angle to observer (polar angle θ in spherical coordinates) relative to orientation $\mathbf{R}_l$ and models shielding and the Fresnel lens profile. The horizontal profile is a function of horizontal angle to observer (azimuthal angle φ in spherical coordinates) relative to orientation $\mathbf{R}_l$ and models sectoring. Figure 4.4 shows φ and θ. To make the functions independent of the position and orientation of the lights, they are parameterized in the local frame of reference (i.e. light-space).

As the blink profile these are also tabulated.

Lights with Multiple Color Sectors This model does not natively support multiple colors (e.g. the green, white, red sectors of light houses), but such setups can be emulated by having multiple lights at the same position and orientation, but with different colors. They can share horizontal profiles because because the tabulated format supports multiple sectors.

[1] This is slightly inaccurate for light houses with rotating blinds

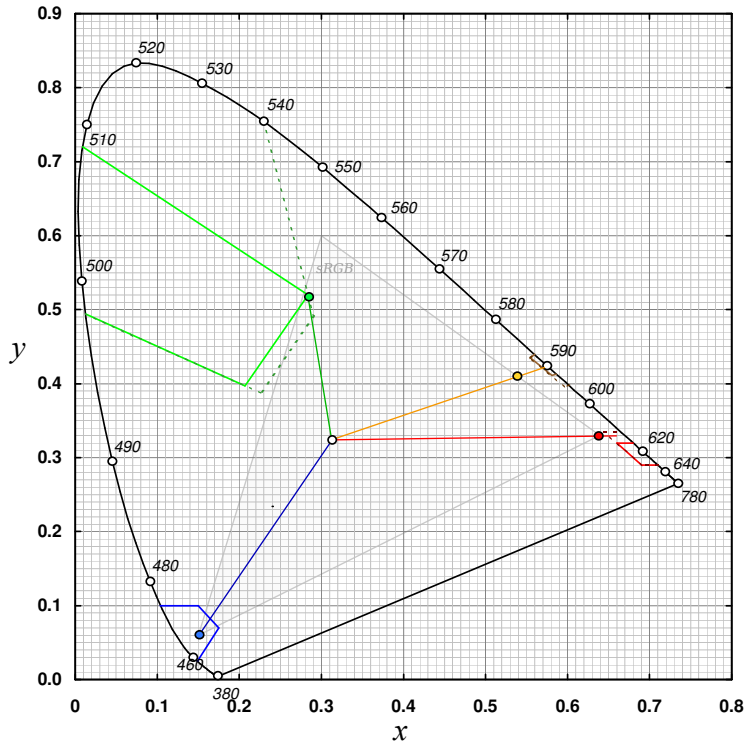


Figure 4.3: Gamut mapping the recommended IALA colors to sRGB color space.

Since I am using billboards to display the glare for sub-pixel lights and there might be some overlap between the sectors I might need to display two billboards for a given (θ, ϕ) to observer.

4.1.1 Vertical profile parameterization

The vertical profile lookup texture can be indexed by angle θ in radians, or more computation efficient, $\cos\theta$. A side effect of using $\cos\theta$ is that this parameterization is not uniform and actually has more precision at close to 90° where the change in vertical profile is highest.

Given the angle from zenith to observer θ, the parameterization of the vertical

Color	Closest sRGB xy	Normalized linear RGB
White	(0.31, 0.33)	(1.0000, 1.0000,1.0000)
Red	(0.64, 0.33)	(4.7021, 0.0000, 0.0000)
Red (optimum)	(0.62, 0.32)	(4.6478, 0.0059, 0.1020)
Green	(0.28, 0.52)	(0.0160, 1.3701, 0.2326)
Blue	(0.15, 0.06)	(0.0000, 0.0000, 13.8552)
Yellow	(0.54, 0.41)	(2.6700, 0.6045, 0.0000)

Table 4.1: The normalized linear RGB colors of the sRGB gamut mapping from figure 4.3. RGB colors normalized to luminance $Y = 1$ in xyY color space.

profile is:

$$p(\theta) = \frac{\cos(\frac{\pi}{180^\circ}\theta) + 1}{2} \qquad \theta \in [0, 180^\circ] \tag{4.1}$$

$$p^{-1}(x) = \frac{180^\circ}{\pi} \arccos(2x - 1) \qquad x \in [0, 1] \tag{4.2}$$

where p is the transformation from angle to normalized lookup index and p^{-1} is the table parameterization angle given a normalized lookup index.

4.1.2 Horizontal profile parameterization

In contrast, the horizontal profiles vary a lot more so a uniform sampling parameterization is more appropriate.

For omni-directional incandescent light bulbs, measurements of the horizontal profiles show pronounced internal shadowing. For LED lights, I assume this shadowing to be irrelevant because a LED marine light uses many small LED emitters uniformly spaced and as such they should not suffer the same shadowing artifacts. Using tabulated data also allows one source to have multiple segments of the same color (e.g. light towers may have several sectors with the same color).

The horizontal profile is specified from 0° to 360° with uniform parameterization based on the azimuthal angle φ

$$p(\varphi) = \frac{\varphi}{360^\circ} \qquad \varphi \in [0, 360^\circ] \tag{4.3}$$

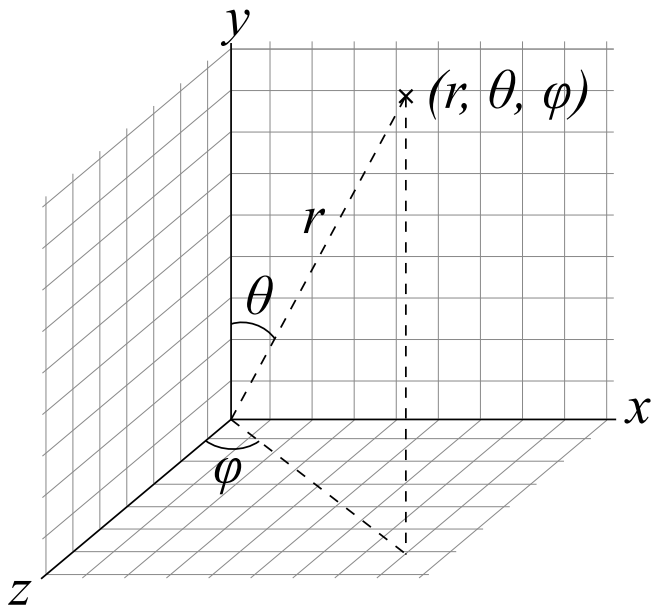


Figure 4.4: Spherical coordinates with polar angle θ and azimuthal angle φ.

Given the direction towards the eye $\vec{e}_L$ in the local light space defined by $\mathbf{R}_l$, the azimuthal angle φ is:

$$\varphi_{rad} = \frac{\pi + \text{atan2}(\vec{e}_L.x, \vec{e}_L.z)}{2\pi} \tag{4.4}$$

where atan2 follows this definition:

$$\text{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$

and returns values in range $[-\pi, \pi]$.

4.1.3 Intensity and radiance value of emissive pixels

Approximating the ATON lights as isotropic point light sources with radius r, the radiance emitted on the surface of the light source is given by equation 3.4, restated here for reference:

$$L_e = \frac{I}{\pi r^2}$$

Attenuation for sub pixel sources Rasterizing the geometry of 10cm ATON lights with view distances of tens of kilometers reaches the practical resolution constraints: At these distances, the screen-projected surface geometry is smaller than the display resolution. In this case, the assumption that each sample covers the whole pixel breaks which is further exemplified as some frames might show the rasterized fragment and others might not when the viewport changes.

This is an issue for thin geometry in general and a solution for telephone wires given by Persson [Per12] is to enforce a minimum pixel size and decrease the radiance with the ratio of a pixel and the screen-projected width of the wire which is inversely proportional to the squared distance.

If every pixel is assumed to cover the same solid angle, then the projected area on the image plane is also the same for every pixel, so when the area is less than one pixel, the outgoing radiance decreases with

$$\frac{A_{\text{projected light source}}}{A_{\text{pixel}}}$$

To compute the pixel area of the light source on the image plane for a spherical light source with radius r and distance to viewer d, the camera focal distance in pixels for field of view θ and image plane pixel height h is

$$f = \frac{\frac{h}{2}}{\tan\frac{\theta}{2}}$$

The 2D projection of a spherical light source is approximated as a disc with area πr_{proj}^2 where

$$r_{proj} = f\frac{r}{d}$$

In principle, the 2D perspective projection of a sphere is an ellipse because d is different for every point on the sphere, but for simplicity the approximation is used.

In wide field of view simulations at FORCE (such as shown in figure 1.3), each display column has it's own viewport to minimize perspective distortions. This would also reduce the error of the disc approximation.

A screen pixel has area 1px^2 and the attenuation is only valid for sub pixel sizes, the final attenuation for all source sizes is:

$$\min\left(1, \frac{A_{\text{projected light source}}}{A_{\text{pixel}}}\right) = \min\left(1, \pi(f\frac{r}{d})^2\right)$$

so the emitted radiance for sub-pixel ATON lights is

$$L_{e,subpixel}(d) = L_e \min\left(1, \pi(f\frac{r}{d})^2\right) \tag{4.5}$$

This is consistent with the Inverse-square law and equation 3.5. Together with equation 3.4, the π and r^2 factors cancel out and the focal distance f provides a pixel area scaling factor.

4.2 Glare pattern generation

The glare pattern generation method is based on Ritschel et al. [RIF+09]. The method has a user parameter that adjusts the size of the glare so it can match the 4° radius found by [Sim53].

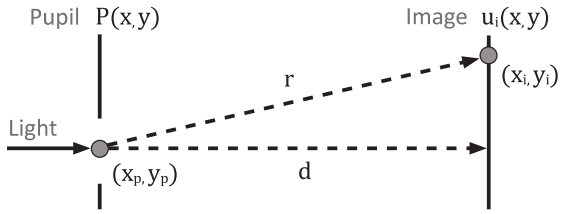


Figure 4.5: Simplified optical system. From [RIF+09]

Given a simple optical system with parallel pupil and image/retina planes (figure 4.5), the diffracted radiance at (x_i, y_i) on the image plane is approximated using Fresnel diffraction:

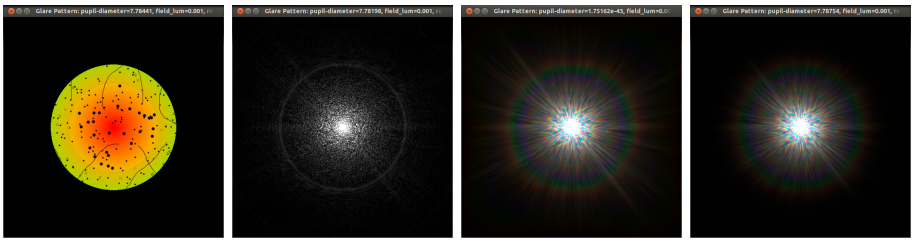
$$\begin{aligned} L_i(x_i, y_i) &= K|\mathcal{F}\{P(x_p, y_p)E(x_p, y_p)\}_{p=\frac{x_i}{\lambda d}, q=\frac{y_i}{\lambda d}}|^2 \\ K &= 1/(\lambda d)^2 \\ E(x_p, y_p) &= e^{i\frac{\pi}{\lambda d}(x_p^2+y_p^2)} \end{aligned} \tag{4.6}$$

The function $P(x, y)$ is the pupil function that returns 1 inside the pupil and 0 outside, λ is the wavelength of the light, d is the pupil-retina distance and $\mathcal{F}$ is the Fourier transform evaluated at coordinates (p, q). I refer to [RIF+09] for the derivation of Fresnel diffraction.

The following steps generate the glare pattern:

1. Render pupil image multiplied by the complex exponential E to a two-channel texture (shown in figure 4.6a. The real part is drawn to the red color channel and the imaginary part is drawn to the green channel).
2. Compute FFT using General Purpose GPU (GPGPU).
3. Compute mono-chromatic radiance PSFfrom the electromagnetic field computed by the FFT, store in a one-channel texture (shown in figure 4.6b).
4. Compute PSF normalization factor using mipmapping
5. Compute chromatic blur using the XYZ color matching functions.
6. Convert to HDR sRGB space and normalize with $\int_{360nm}^{830nm} V(\lambda)d\lambda$ and the computed PSF factor (shown in figure 4.6c).
7. Precompute billboards for larger visual angles (estimated in pixels) and apply a radial falloff kernel (falloff kernel applied to the base billboard is shown in figure 4.6d).

Steps 5, 6 and 7 depend on the actual SPD. If performance allows, glare PSFs for multiple SPDs could be computed.



(a) Pupil model **(b)** Monochromatic diffraction **(c)** Spectral blur **(d)** Final with falloff using eqn. 4.12

Figure 4.6: Intermediate states in the glare generation process.

4.2.1 Pupil image construction

To generate the glare diffraction pattern, the pupil aperture, lens particles and gratings and particles in the vitreous humor are projected orthogonally onto a plane.

Looking at a glare source, the pupil is subject to periodic fluctuations, the *pupilary hippus*, presumed to be caused by the iris trying the adjust to the huge dynamic range of intensities. Based on observed data, Ritschel et al. [RIF+09] proposed modeling the phenomenon by

$$h(p,t) = p + \text{noise}\left(\frac{t}{p}\right)\frac{p_{max}}{p}\sqrt{1-\frac{p}{p_{max}}} \tag{4.7}$$

where p is the mean pupil diameter given by equation 3.15 and noise is a smooth noise function with three octaves (for a comprehensive description on noise, see [EMP+03])

The pupil, lens and cornea are parallel projected onto the image plane / retina. The vitreous humor particles are drawn as large opaque point primitives with a uniform random distribution. The lens gratings are drawn as radial line primitives starting at some distance from the center of the pupil to the end of the pupil plane. The lens particles are drawn as many small opaque point primitives. All components are drawn to an offscreen buffer.

4.2.2 PSF generation using FFT

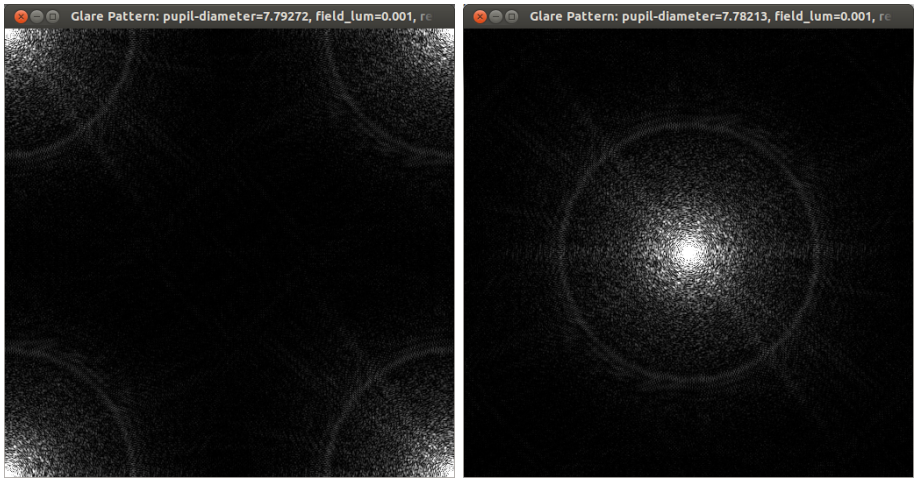
Using GPGPU the Fast Fourier Transform (FFT) of the pupil image can be computed efficiently on a GPU with greater performance compared to a CPU implementation. Additionally, computing the FFT on the GPU avoids transferring data from the GPU to the CPU and back again after computation.

For use as a billboard the image needs a cyclic shift to rearrange the quadrants of the texture (see figure 4.7).

If the input coordinates are given in $[0,1] \times [0,1]$, then the output coordinates for the cyclic shift is:

$$p(\vec{x}) = \left(x + \frac{1}{2}\right) \mod 1 \tag{4.8}$$

The Fourier transform is the approximation of the electromagnetic field at the retina (see equation 4.6), so to get the radiance I take the square magnitude



(a) Without **(b)** With

Figure 4.7: Effect of cyclic shifting the Fourier Transformation

of the complex FT. This is done as a single pass that takes a two component texture and outputs to a single component texture.

4.2.3 Monochromatic PSF Normalization

Spencer et al. [SSZG95] normalized their empirical glare PSF over the incoming hemisphere. For simplicity I normalize the monochromatic PSF such that it sums to 1, which is fast and simple to do on a GPU.

Parallelizing the summing of all pixels can be done using the Summed Area Table (SAT) algorithm [Ngu07, Ch. 8] where the sum can be directly read from the far corner texel.

Using mip mapping, the average can be computed in $\log_2 n$ passes, where n is the maximum of texture width and height. The sum is then the product of the average and the number of texels.

Memory wise, mipmapping is more efficient when only the sum is needed: SAT requires 2x memory (a separate buffer where the SAT is computed), mip mapping only requires $\frac{4}{3}$x. Additionally, mipmapping the PSF texture prevents artifacts when computing the chromatic blur.

4.2.4 Chromatic blur

In [vdBHC05] van den Berg et al. showed that the PSF for wavelength λ, $F_\lambda(\mathbf{x})$ can be reasonably approximated with the PSF for wavelength λ_0, $F_{\lambda_0}(\mathbf{x})$, scaled by $\frac{\lambda}{\lambda_0}$ (see equation 4.9). This is not strictly true for the Fresnel approximation because E also depends on wavelength, but it works well in practice and allows computing only once the Fourier Transform of the pupil image, which is very important from a performance point of view.

$$F_\lambda(\mathbf{x}) = F_{\lambda_0}\left(\frac{\lambda}{\lambda_0}\mathbf{x}\right) \tag{4.9}$$

As the PSF is discretized to a texture, this equation transfers to scaling PSF texture lookup coordinates based on wavelength ratio.

The glare pattern is output to an sRGB calibrated display so when using CIE xy coordinates to specify hue, I assume the SPD follows Illuminant D65 so the spectral glare stimulus $C(\lambda)$ under D65 white point is given by

$$C(\lambda) = F_\lambda(\mathbf{x}) I_{D65}(\lambda) \tag{4.10}$$

where I_{D65} is the normalized spectral power distribution of Illuminant D65. Using the D65 SPD, a constant F integrates to white on an sRGB display.

If the relative SPD of the actual light source is available, then that could be used instead. In this case, the tristimulus emissive color (used for surface reflection) should also be computed from the spectrum using the equation 3.8 and to linear RGB color space using equation 3.12.

The XYZ tristimulus values are computed by solving equation 3.8 for the spectral glare stimulus $C(\lambda)$ with $\lambda_0 = 575\text{nm}$.

The final linear RGB color is then transformed from CIE XYZ to linear sRGB using equation 3.12.

4.2.5 Radial falloff for billboards

The PSF is infinite, so size limits have to be chosen to conserve bandwidth and limit discontinuities.

Spencer et al. [SSZG95] used a PSF with the four times the resolution of the image to allow the bright pixels to spread light to every other pixel. This is

clearly infeasible for real-time use for HD (1280x720) or Full HD (1920x1080) resolutions, but a 512 by 512 texture should be sufficient to show the central parts (i.e. the ciliary corona and lenticular halo).

In many cases, the whole dynamic range of the glare pattern will not be visible against the background and so using a discrete window of the PSF is fine, but for very bright lights discontinuities will appear. To fix this issue I use an Epanechnikov kernel [Sil86] for a radial falloff so that the glare billboards will have a smooth transition given by:

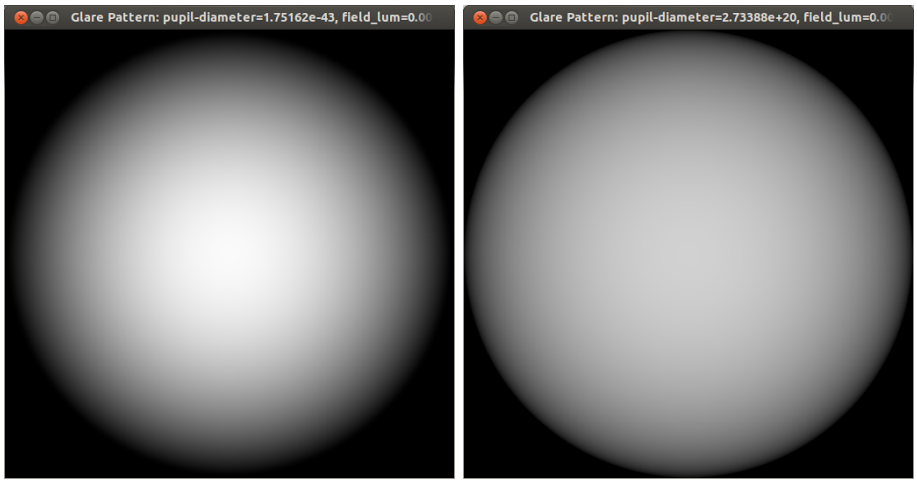
$$K_2(\vec{x}) = \begin{cases} \frac{2}{\pi}(1 - \vec{x} \cdot \vec{x}) & \text{if } \vec{x} \cdot \vec{x} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

where $\vec{x}$ is the normalized position from the center of the billboard.

Alternatively, Silverman's second order kernel [Sil86] can be used if the contrast in light intensity to background intensity is very high:

$$K_2^2(\vec{x}) = \begin{cases} \frac{3}{\pi}(1 - \vec{x} \cdot \vec{x})^2 & \text{if } \vec{x} \cdot \vec{x} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

A visualization of both kernels is shown in figure 4.8.



(a) Silverman's second order kernel (eqn. 4.12)

(b) Epanechnikov kernel (eqn. 4.11)

Figure 4.8: Falloff kernel weigths

4.2.6 Area light sources

When a light source covers multiple pixels, the glare PSF should be applied to all visible pixels as shown in figure 1.9. Ritschel et al. [RIF+09] use convolution to apply the PSF which per definition applies the PSF to visible pixels (and all other pixels, so glare from high intensity specular reflections is free).

I approximate this effect using precomputed (whenever the glare pattern changes) billboards where the convolution of the PSF with N disks (i.e. the approximation of N screen-projected spheres with increasing radii). Figure 4.9 shows the precomputed billboards for $N = 4$. The convolution is applied by looping over

(a) Radius 2 **(b)** Radius 5 **(c)** Radius 10 **(d)** Radius 20

Figure 4.9: Precomputed convolution of screen-projected light sources. The radiance of the convoluted pixels are the same in all images. Note how the Epanechnikov filter from equation 4.11 creates a sharper and sharper boundary.

all visible pixels in the disk placed at the center and then additively blending the billboard. The area of disc i is approximately $A = \pi r_i^2$ because the disk is discretized on a grid with simple thresholding.

I use a linear parameterization on radius for N billboards (e.g. $r_i = i, i \in 1, 2, \cdots, N$) for simplicity and it works well in practice.

As the projected area is proportional to the inverse squared distance and the area the layers cover increases with radius squared, I can linearly interpolate between the area billboards to get a smooth transition when moving closer or away from the light source.

Given the approximated projected radius r_{proj} and the radius of the maximum area layer r_{max}, then the normalized lookup coordinate in the $[0, 1]$ range is

$$i = \frac{r_{proj}}{r_{max}} \tag{4.13}$$

4.3 Glare pattern application

To render the glare I use viewport oriented *billboards* [AMHH08, section 10.6]. Normally, billboarding just orients a quad primitive in worldspace to face a certain direction (which could be towards the camera, but aligned to world y-axis, etc). In this case the mapping between the screen pixels and the glare texture texels should be constant (and as close to 1:1 for best visual results). This property allows the billboards to shrink and show a subset of the glare pattern for performance.

For correct area source transitions, the pixel mapping must also match so the resolution of the generated glare image should follow the framebuffer resolution.

The glare texture is modified by the radiant exitance of the light source which is computed from the blinking, horizontal and vertical profiles evaluated according to the direction to the eye.

The glare is applied as the last step before tone mapping as the glare is diffracted and scattered light received by the photoreceptors.

Convolution As mentioned in the Introduction, wide filter convolution is problematic regarding sub pixel lights, but the billboarding can be used alongside general glare approximation based on Gaussian convolution. In this case, the general Gaussian convolution should happen before the lights are rendered to prevent light glare from contributing twice. By combining the two approaches, intense specular reflections can potentially also generate glare.

4.3.1 Fog extinction

Fog extinction is applied to the outgoing radiance using Bouguer's Law [Per48]

$$T = e^{-\rho d} \tag{4.14}$$

where ρ is the extinction coefficient and d is the distance traveled.

4.3.2 Glare occlusion

As glare is a phenomenon that happens inside the eye, occluded lights must not render billboards. Rendering the billboards after normal geometry allows

manual depth testing of the centers lights and depth occluded lights can be discarded for billboard rendering.

Multi-channel setups Unfortunately this does not work for multichannel setup as one channel does not have access to the depth buffers of other channels and thus suffers from the same fundamental issue as a post process convolution. The one solution is the same: Increase the viewport of each channel with the radius of the glare filter kernel. Another is to do the visibility checks on the CPU using ray casting and then only submit the visible sources for rendering.

4.4 Tone mapping

As mentioned in the background, there are many approaches to tone mapping and since tone mapping is a tool for rendering ATON lights I have chosen to implement two simple *global* and *dynamic* operators: one *perceptual* global linear scale factor based on TVI from Durand et al. [DD00] and one *empirical* global non-linear TMO based on photographic dynamic range reduction Reinhard et al. and Krawscyk et al. [RSSF02]. Both employ the perceptual "blueshift" for rod-dominant illumination from Krawszyk et al. [KMS05].

The tone map process is:

- Render scene to RGB HDR buffer.
- Compute luminance for each pixel in XYZ color space.
- Compute global adaptation target based on log-average scene luminance using equation 4.15.
- Compute current temporal light/dark adaptation luminance using exponential decay using equation 4.16.
- Compute the global scene "key" or scaling factors based on current adaptation.
- For each pixel:
  - Apply dynamic range compression
  - Compute rod saturation factor (in $[0, 1]$) based on adaptation luminance.

– Tonemap the input RGB to display luminance and linearly blend between monochromatic and color vision based on rod saturation.
– Output tonemapped value.

Global target adaptation luminance Average luminance based on geometric mean.

$$\bar{L}_w = \exp\left(\frac{1}{N}\sum^{N}\log(L_w + \epsilon)\right) \tag{4.15}$$

where ϵ is a small positive number that prevents undefined behavior if the luminance is 0. Some implementations make the average value sticky to prevent global illumination changes when the light suddenly changes. This has not been a problem for this implementation with blinking lights, so it has not been considered. Durand et al. [DD00] and Tumblin et al. [THG99] further restricts the adaptation luminance to the fovea area, but I use the whole area because focus is not given.

Temporal adaptation As the HVS adapts to change in background intensity, the temporal adaptation for rods and cones are simulated using exponential decay

$$\bar{L}_{wa}^{new} = \bar{L}_{wa} + (\bar{L}_w - \bar{L}_{wa})(1 - \exp\left(-\frac{\Delta t}{\tau(\bar{L}_w)}\right)) \tag{4.16}$$

where Δt is the discrete time-step in seconds between two frames and

$$\tau(L_w) = \sigma(L_w)\tau_{rod} + (1 - \sigma(L_w))\tau_{cone} \tag{4.17}$$

with $\tau_{rod} = 0.4s$, $\tau_{cone} = 0.1s$ for light adaptation and rod sensitivity σ defined by equation 4.27. Dark adaptation much slower than light adaptation and is not modeled here.

Empirical dynamic range compression The mapping of the average luminance to "middle gray", which gives the relative luminance L_r is done using linear mapping:

$$L_r = \alpha(\bar{L}_{wa})\frac{L_w}{\bar{L}_{wa}} \tag{4.18}$$

The key α of the scene is estimated using a modified version of the equation 11 in [KMS05]:

$$\alpha(\bar{L}_{wa}) = 1.002 - \frac{2}{2 + \log_{10}\bar{L}_{wa}} \tag{4.19}$$

The key value in the paper was estimated empirically and I have modified it because the original would cause too bright night scenes. Luksch [Luk06] also modified the automatic key estimation in his implementation, but whereas I only modified the first constant, he made major changes.

Afterwards, a non-linear transform is applied to the relative luminance based on Maximum-to-white from [RSSF02] to prevent excessive desaturation. White value is relative to L_r. I use $L_{white} = 2$ for good results.

$$L_d = \frac{L_r \cdot \left(1 + \frac{L_r}{L_{white}^2}\right)}{L_r + 1} \tag{4.20}$$

Perceptual dynamic range compression The perceptual TMO is a global, linear scale factor that maps world luminances to display luminances.

$$L_d = \frac{m\, L_w}{L_{dmax}} \tag{4.21}$$

where $m = \frac{\epsilon(L_{da})}{\epsilon(L_{wa})}$. Here $\epsilon(L)$ is the TVI threshold function for luminance L. This maps the contrast threshold of the scene adaptation $\epsilon(L_{wa})$ onto the contrast threshold of the display $\epsilon(L_{da})$ given viewer adaptation L_{da}. Here the viewer is assumed to be adapted to half the display's maximum luminance $L_{dmax}/2$ which can be measured using a photometer.

The scaling factor m is based on TVI, which is different for rods and cones, so two scaling factors are used:

$$m_C = \frac{\epsilon_C(L_{daC})}{\epsilon_C(L_{waC})}, m_R = \frac{\epsilon_C(L_{daC})}{\epsilon_R(L_{waR})} \tag{4.22}$$

The contrast threshold for the display for cones are used to compute the scaling factor for rods because the viewer is assumed to be photopically adapted.

The contrast thresholds for rods and cones based on TVI given by the piecewise functions

$$\log \epsilon_R(L_{aR}) = \begin{cases} -2.86 & \text{if } \log L_{aR} \leq -3.94 \\ \log L_{aR} - 0.395 & \text{if } \log L_{aR} \geq -1.44 \\ (0.405 \log L_{aR} + 1.6)^{2.18} - 2.86 & \text{otherwise} \end{cases} \tag{4.23}$$

$$\log \epsilon_C(L_{aC}) = \begin{cases} -0.72 & \text{if } \log L_{aC} \leq -2.6 \\ \log L_{aC} - 1.255 & \text{if } \log L_{aC} \geq 1.9 \\ (0.249 \log L_{aC} + 0.65)^{2.7} - 0.72 & \text{otherwise} \end{cases} \tag{4.24}$$

The scotopic luminance V is approximated from XYZ_w using the following transform [LRP97]:

$$V_w = Y_w \left(1.33 \left(1 + \frac{X_w + Y_w}{Z_w}\right) - 1.68\right) \tag{4.25}$$

As the rods and cones are two separate systems that both contribute to vision,

$$L_d = \frac{m_C L_w + \sigma(L_w) m_R V_w}{L_{dmax}} \tag{4.26}$$

Blueshift The mono-chromatic vision at illumination levels where only rods are active (explained in section 3.4) are modeled by the rods sensitivity, approximated [KMS05] by

$$\sigma(L_w) = \frac{0.04}{0.04 + L_w} \tag{4.27}$$

The hue (x_b, y_b) of the "blueshift" is a somewhat empirical subject. Jensen et al. [JDD+01] looked towards paintings for inspirations and found that an average blue hue of $(0.28, 0.29)$, but chose $(0.25, 0.25)$ in their paper. The hue used by Durand et al. [DD00] was $(0.3, 0.3)$ based on psychophysical data from Hunt [Hun52].

Integration In xyY color space, the tone mapped color accounting for scotopic blueshift is:

$$\begin{bmatrix} x_d \\ y_d \\ Y_d \end{bmatrix} = \begin{bmatrix} x_w \cdot (1 - \sigma(L_w)) + x_b \cdot \sigma(L_w) \\ y_w \cdot (1 - \sigma(L_w)) + y_b \cdot \sigma(L_w) \\ \max(1, L_d) \end{bmatrix} \tag{4.28}$$

where $\max(1, L_d)$ performs rudimentary hue preservation, which can be disabled by simply using L_d instead. The final linear RGB can then be computed using equations 3.10 and 3.12.

Alternatively in RGB color space, the final tone mapped color using the linear sRGB input value $[R_w, G_w, B_w]^T$

$$\begin{bmatrix} R_d \\ G_d \\ B_d \end{bmatrix} = \begin{bmatrix} R_w \\ G_w \\ B_w \end{bmatrix} \cdot \frac{L_d}{L_w} \cdot (1 - \sigma(L_w)) + \begin{bmatrix} 1.016784 \\ 0.983176 \\ 1.107126 \end{bmatrix} \cdot L_d \cdot \sigma(L_w) \tag{4.29}$$

Optionally hue preservation (preventing mixed rgb triplets from over-saturating and distorting colors) can be applied by scaling the final RGB triplet with the largest component, preserving the ratio between components:

$$\begin{bmatrix} R'_d \\ G'_d \\ B'_d \end{bmatrix} = \begin{bmatrix} R_d \\ G_d \\ B_d \end{bmatrix} \cdot \frac{1}{\max(1, \max(R_d, \max(G_d, B_d)))} \tag{4.30}$$

After tone-mapping, the inverse gamma is applied to the linear RGB values using hardware sRGB support. Otherwise, equation 3.13 is used to apply gamma corrected, non-linear sRGB.

Chapter 5

Implementation

In this chapter I discuss the implementation details of the method. My implementation uses modern OpenGL 3.3 with a few extensions that backports OpenGL 4 features to OpenGL 3.3 compatible hardware, but the concepts should be transferable to a Direct3D 10/11 implementation. This chapter assumes a faily deep knowledge of OpenGL and shaders.

5.1 Light model data specification

Orientation Several methods can be used for orientation. A full 3×3 rotation matrix, Euler angles with (yaw, pitch, roll) or a quaternion. The rotation matrix requires 9 floats, but can be directly used for up, right and forward basis vectors. Euler angles require 3 floats, but need to be converted to a rotation matrix. Quaternions require 4 floats and transforming a point or vector requires less ALU instructions than a full rotation matrix, but to get the basis vectors up, right, forward, three such transformations have to be done.

I use quaternions for point light orientations to minimize the bandwidth requirements. A good description of quaternions is given by [AMHH08].

Blinking Blink patterns are stored using 1D texture arrays. All patterns have the same length in time and the same number of samples.

For in-between sampling, I use linear filtering and repeat wrapping.

Vertical and horizontal profiles As binding textures is a relatively expensive operation, I use 1D Texture Arrays to store the discretized profiles and just store the lookup id in the shader attribute. As a consequence, all profiles of the same type need to have the same resolution.

I use linear filtering when sampling the profile textures. Unfortunately very sharp profiles then require larger profile resolution so the resolution must be tweaked to find the right balance between memory requirements and visual accuracy. Though on modern GPU with lots of memory this is most likely not an issue.

As the horizontal and vertical profiles in theory are specific to a particular light source, I can use a single ID property to look up both profiles. But for convenience I use separate IDs so different types can share profiles.

I assume the view direction constant for the whole light so I only need to look up the profile texture once. This is a reasonable approximation because the angular diameter of the point sources is small. The code used to compute the lookup parameterization is shown in listing 5.1.

When sampling the texture I use Clamp To Edge wrapping mode.

```
// compute world space light up and to eye vectors.
vec3 light_up_ws = qrot(q_orientation, vec3(0, 1, 0));
vec3 to_eye_ws = mat3(_inverse_mv) * normalize(-eye_center.xyz);

// compute vertical profile parameterization
float cos_theta = dot(to_eye_ws, light_up_ws);
float xi_ver = 0.5 * cos_theta + 0.5;

// compute horizontal profile parameterization
vec3 to_eye_ls = qrot(qinv(q_orientation), to_eye_ws);
float phi = atan(to_eye_ls.x, to_eye_ls.z);
float xi_hor = (phi + pi) / (2*pi);
```

Listing 5.1: Horizontal and vertical parameterization GLSL code

5.2 Glare pattern generation

The Fresnel diffraction (equation 4.6) evaluates the Fourier Transform at $(\frac{x_i}{\lambda d}, \frac{y_i}{\lambda d})$. In the computer simulated Fresnel diffraction method by Trester [Tre99], they use $\lambda d = N$, where N is the side length of the pupil-plane and image-plane. This approximation cancels out the intrinsic $\sqrt{N}M$ scaling factor of the forward FFT.

In this method, the resolution of the glare image is decoupled from the resolution of the screen and the final glare patterns are drawn unscaled to match the screen-projected area. To be able to adjust the size of the glare, I use λd as a scaling factor which I set empirically. As a consequence, I have to scale the intensities in the PSF with $1/N$ (though when using the simple normalization, linear scaling does not matter).

I use the OpenCL FFT implementation from Apple [App12]. It works on buffers and not textures so I use pixel buffer objects (PIXEL PACK) to copy the texture data to an OpenCL buffer. This happens on the device so this is a quite fast solution, but this copy step could be eliminated with a custom OpenCL FFT that works on the actual OpenGL texture using OpenGL/OpenCL interoperability extension. Also, the current NVIDIA drivers do not support the sync object extension that allows fine grained synchronization between OpenGL and OpenCL, thus requireing a complete pipeline synchronization with `glFinish`.

Because the pupil texture is transferred via PBO, I do the FFT in-place and then copy the result to the texture using a PIXEL UNPACK buffer.

For my implementation with OpenCL FFT I need five renderable Textures:

Pupil, 2chan 32bit where the pupil, lens gratings, lens particles and cornea particles are parallel projected on to, multiplied by the complex exponential E from equation 4.6.

Pupil FFT, 2chan 32bit is the forward Fourier Transform of the complex pupil image in the frequency domain. The values are scaled with $\sqrt{N}$ where N is the number of texels as a result of the FFT.

Pupil PSF, 1chan 16bit, mipmapped where the Pupil FFT image is cyclic shifted, scaled with $1/\sqrt{N}$ and converted from complex electromagnetic field to real radiance. The lowest mipmap level at

$$\lfloor \log_2 \max(\text{width}, \text{height}) \rfloor$$

multipled by N is the energy conserving normalization factor.

Pupil RGB, 4chan 16bit where the Pupil PSF is integrated over the visible spectrum (using the XYZ color matching function, XYZ CMF and the sRGB illuminant D65) and the final RGB glare pattern converted from Pupil XYZ using XYZ to sRGB matrix and normalized using the Pupil PSF normalization factor. The fourth channel is unused, but added for 4 byte alignment.

Pupil RGB, 4chan 16bit, 3D texture is the precomputed convolution of light sources covering multiple pixels. Silverman's second order kernel (equation 4.12) is multiplied to the result and linear texture filtering allows smooth interpolation between layers. The fourth channel is left unused for 4 byte alignment.

Because only one of the textures is bound for rendering at the time, I use one Framebuffer Object and just change attachment. This is faster than changing Framebuffer Object binding.

The Pupil and Pupil FFT images are 2 channel because they are complex numbers.

As line smoothing the lens gratings are only supported with alpha blending and the pupil image is two channel complex number without alpha, I use multisampling to prevent aliasing artifacts. Using multisampling means that the samples have to be resolved (by blitting the multisampled Framebuffer Object to a non-multisampled Framebuffer Object) before running the FFT.

The modulo operator in the FT cyclic shift (equation 4.8) is implemented using REPEAT texture sampling.

5.2.1 Chromatic Blur

The chromatic blur can be efficiently drawn in one pass using multiple texture lookups. Compared to a multi-pass method with one pass per sample additively blended, the single pass method is 10 times faster ($\tilde{1}$ms vs $\tilde{1}$1 ms on a mobile GeForce 650M GPU).

The shaders for the chromatic blur are shown in listings 5.2 and 5.3.

The `_Color` array is the $\bar{x}, \bar{y}, \bar{z}$ weighted samples of the light spectrum (equation 4.10) and `_Scale` array is the wavelength dependent scaling computed using equation 4.9.

```
layout(location = 0) in vec4 in_vertex;
out vec2 texcoord;

void main() {
  texcoord = in_vertex.xy;
  gl_Position = in_vertex;
}
```

Listing 5.2: Single-pass chromatic blur vertex shader

```
#define SAMPLES 30
uniform sampler2D _MainTex;
uniform vec4 _Color[SAMPLES];
uniform float _Scale[SAMPLES];

in vec2 texcoord;
out vec3 frag_color;

void main() {
  frag_color = vec3(0);
  for (int i = 0; i < SAMPLES; ++i) {
    vec2 t_lambda = texcoord * _Scale[i] * 0.5 + 0.5;
    float c = texture(_MainTex, t_lambda).r;
    frag_color += c * _Color[i].rgb;
  }
}
```

Listing 5.3: Single-pass chromatic bluring fragment shader

When scaling the PSF, mipmap based (trilinear) minification filtering (with `GL_LINEAR_MIPMAP_LINEAR` is required to prevent Moirè pattern artifacts (see figure 5.1).

5.2.2 Area source glare

The area glare billboards are stored in a 3D texture with linear filtering. As the falloff kernel forces the glare pattern to be zero at the edge, I use Clamp To Border with black border color for the xy texture coordinates. For z, I use Clamp To Edge to fall back on using the largest precomputed area source when the actual projected area exceeds the area of the precomputations.

Like the chromatic blur going from rendering many quads to doing all the lookups in the shader the computation time went from 40ms to 4ms (again the 10x increase).

The shader code for generating the layers are shown in listings 5.4 and 5.5.

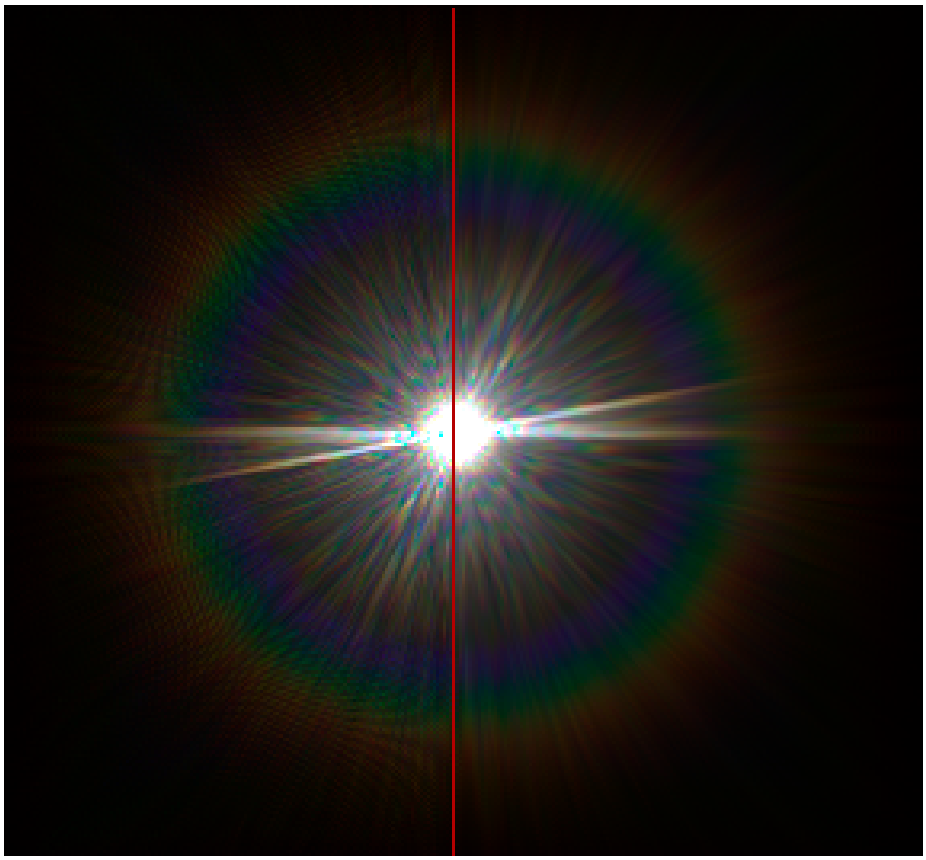


Figure 5.1: Comparison of bilinear and trilinear filtering

```
void main() {
uniform float _GlareScale = 1.6;

// input is a fullscreen quad in NDC space
layout(location = 0) in vec3 in_vertex;
out vec2 texcoord;
out vec2 position;
void main() {
  texcoord = in_vertex.xy * 0.5 + 0.5;
  position = in_vertex.xy * _GlareScale;
  gl_Position = vec4(in_vertex, 1);
}
```

Listing 5.4: Area layer program: Vertex shader

```
uniform sampler2D _InputTex;
uniform int _Radius = 1;
in  vec2 texcoord;
in  vec2 position;
out vec4 frag_color;

void main() {
  frag_color = vec4(0);
  for (int i = -_Radius; i <= _Radius; ++i) {
    for (int j = -_Radius; j <= _Radius; ++j) {
      vec2 t = texcoord + vec2(i,j)/textureSize(_InputTex, 0).xy
      vec3 c = texture(_InputTex, t).rgb;
      // avoid dependent texture read
      if (length(vec2(i,j)) <= _Radius)
      frag_color.rgb += c;
    }
  }
}
```

Listing 5.5: Area layer program: Fragment shader

5.3 Glare pattern application using Geometry Shader Billboards

For efficiency I render the light source geometry as general geometry and then afterwards I render all billboards as points that are expanded using the Geometry Shader.

Alternatively Point Sprites could have been used as the problem maps quite well to those for a single framebuffer output as the geometry shader stage could then be omitted. Occluded lights could still be culled by manually setting the depth

in the vertex shader outside the normalized device coordinates (NDC) frustum and relying on hardware clipping. The drawbacks are that the point sprites are culled by the center of the sprites (which is problematic with multiple adjacent channels) and the maximum point sprite size is implementation defined.

Since the actual data for the billboards are limited I just render all of them in one go and let the geometry shader do visibility testing.

The information needed to draw the billboards will be supplied as attribute data to the shader program.

Since some of the attributes are static and some are dynamic, it makes sense to store them in different buffer objects so they can be easily updated independently, but in my implementation I simply brute force submit light data every frame as I could not measure any performance benefit for the relatively small data

The profiles are supplied as texture arrays and the profile attribute is the index. Storing the profiles as textures allows for easy extension and customization.

The shader pseudo code for the geometry shader billboarding is shown in listings 5.6 and 5.7. The vertex shader is trivial and simply forwards the vertex attribute data to the geometry shader.

```
// uniforms
float _ScreenWidth, _ScreenHeight;
float _CameraFocalDistance;
vec2  _BillboardExtends; // the billboard extends (half-size) in
    NDC
float _GlareScale;
float _MaximumProjDiscRadius;
float _FogDensity;

in VertexShaderOutput {
  float radius; // radius of the spherical light source
  float radiant_intensity;
} IN; // the geometry shader takes point primitives as input

// outputs
out struct FragmentShaderInput {
  vec3 Le; // outgoing radiance
  vec3 texcoord;
} OUT;

void emit(vec4 ss_center, vec3 offset, vec3 texcoord) {
  OUT.texcoord.xy = 0.5 + 0.5 * (2.0 * texcoord.xy - 1.0) /
      _GlareScale;
  OUT.texcoord.z = texcoord.z;
```

```
  gl_Position.xy = ss_center.xy + offset.xy * _BillboardExtends;
  gl_Position.z = ss_center.z;
  gl_Position.w = 1;

  EmitVertex();
}
void main() {
  vec2 eye_center = // compute light position in eye space
  vec2 ss_pos = // compute screen space position
  scene_depth = // lookup linear depth from depthbuffer
  if (scene_depth < world_pos.z)
    return; // don't emit billboard vertices

  profile = // compute horizontal and vertical profile falloffs();

  float r_proj = -_CameraFocalDistance * IN.radius / eye_center.z;
  float pixel_area = 1;
  float source_area = pi * r_proj * r_proj;

  vec3 Le = IN.radiant_intensity / (IN.radius * IN.radius) / pi;
  float fog = exp(-length(eye_center.xyz) * _FogDensity);
  OUT.Le = Le
         * fog
         * min(1, source_area / pixel_area)
         * profile;

  float area_layer = r_proj / _MaximumProjDiscRadius;

  // The billboard is aligned to camera (in screen space), not
      spatial orientation.
  vec3 right = vec3(1,0,0);
  vec3 up = vec3(0,1,0);

  emit(ss_center, -up - right, vec3(0,0, area_layer)); // bottom
      left
  emit(ss_center, -up + right, vec3(1,0, area_layer)); // bottom
      right
  emit(ss_center,  up - right, vec3(0,1, area_layer)); // top left
  emit(ss_center,  up + right, vec3(1,1, area_layer)); // top right

}
```

Listing 5.6: Billboard program: Geometry shader

```
in struct FragmentShaderInput {
  vec3 Le; // outgoing radiance
  vec3 texcoord;
} OUT;

out vec4 frag_color;
void main() {
  vec4 glare = texture(_GlareTex, IN.texcoord);
  vec3 emission = IN.Le * glare.rgb;
  vec3 epsilon = vec3(0.0000001)

```

```
  if (all(lessThan(emission, epsilon))) discard;
13
  frag_color = vec4(emission, 1);
15 }
```

Listing 5.7: Billboard program: Fragment shader

Future work: If multiple glare PSFs are precomputed, then the use of 3D texture for the area layers becomes a minor issue. There are no native GPU support for 3D texture arrays so glare sources should either be batched by type (and then bind the target area layer texture before each batch) or - if there are enough available texture units - all glare texture can be bound and then use branching in the billboarding shader to sample the correct one based on attribute input.

Future work: Coarse visibility testing can be done on the CPU using for instance a grid based acceleration structure. Clustering many similar sources together (such as all sources on a ship or all sources in a port).

Future work: Disable per-frame glare simulation when the closest light source is far enough away that no change is visible.

5.3.1 Falloff kernels

I have found that the Epanechnikov kernel (equation 4.11) requires 32bit precision in the glare texture and the Silverman's second order kernel (equation 4.12) can use 16 bit without floating point precision artifacts when the light intensity and background contrast is high.

5.3.2 Depth buffer lookup for occlusion tests

The light source needs to be tested for occlusion before applying the billboard, not the billboard itself. If the source is visible, then the billboard is fully visible as well. If the source is partially occluded then only the intensity of the glare is affected, not the spatial extend.

This can be done by a manual depth test (or multiple if performance allows and the source covers multiple pixels) using the current depth buffer.

To look up a depth value for comparison I compute the screen projected position of the light source using the camera view and projection matrices and do the $1/w$ perspective divide.

I linearly scale the screen position xy from NDC $[-1, 1]$ to texture space $[0, 1]$ to get the depth texture lookup coordinates.

The native depth buffer stores the fragment depth in NDC $[-1, 1]$ space, which for perspective projection is non-linear in depth.

If the billboard light position needs to be offset compared to the light surface geometry, then it makes sense to do the depth comparison in linear eye space instead of non-linear NDC otherwise the comparison could just compare NDC depth.

Assuming OpenGL default depth range $[-1, 1]$ (Direct3D uses $[0, 1]$), the conversion from non-linear depth z_b in $[0, 1]$ to linear eye space z_e in $[0, z_{far}]$ is

$$z_n = 2 * z_b - 1 \tag{5.1}$$

$$z_e = \frac{2 * z_{near} * z_{far}}{z_{far} + z_{near} - z_n * (z_{far} - z_{near})} \tag{5.2}$$

If the light is rendered as geometry, then the occlusion test must take care that the glare billboard is not occluded by it's own emissive surface (e.g. for practical purposes the position of a point light source might be in the center of the geometry and thus rendering the light source geometry first will then fill the depth buffer with values less than the depth of the source).

Depending on scaling, a simple depth offset can be used in the comparison. I model the light sources as spheres with radius r, so I use offset $r + \epsilon$.

If the depth value is read from the Depth Attachment of the currently bound framebuffer, even though depth testing and depth writes are disabled, then behavior is undefined without explicit synchronization. On my nVidia test hardware, the read back worked in fragment shaders, but not in geometry shader where it instead got wrong values.

Calling `glFinish` before rendering the billboards will force synchronization. OpenGL 4 supports more finegrained synchronization with `glMemoryBarrier`.

Alternatively, doing a depth pre-pass will fully circumvent this issue.

5.3.3 Optimizing fill rate

The billboards are drawn at constant screen-space size independent of distance and intensity. This causes a lot of unnecessary overdraw for distant lights where

only the central part of the glare pattern is visible. The billboard shader is fairly simple, so discarding fragments might not help much with regards to fragment shader execution time, but theoretically, it may save time at the blend stage.

Ideally, with closer integration with the tone mapper, the geometry shader can dynamically crop the billboard based on local background contrast, but that is left for future work.

Chapter 6

Results

This chapter shows and discusses the results of this project. Many of the following images are from night simulations and might not be properly displayed in the print version of this report. If so, I refer to the digital version (where further details might also be visible by zooming).

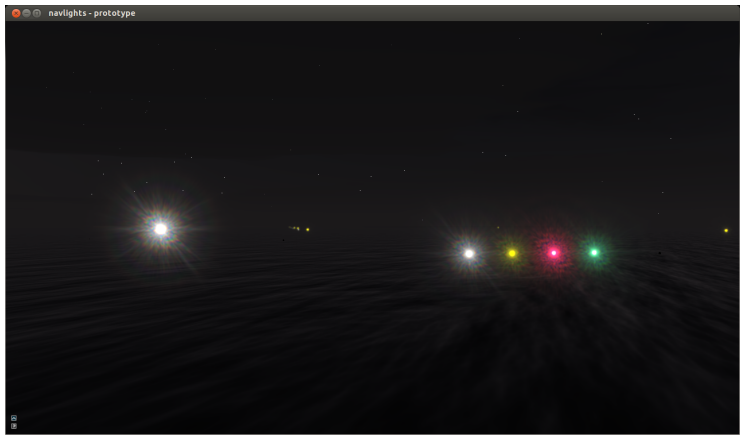


Figure 6.1: A mesopic scene with many point lights

A screenshot of the prototype - using SilverLining atmosphere and simple planar

water - is shown in figure 6.1. It shows the Sabik VP LED (in white (250 cd), yellow (100cd), green (180cd) and red (120cd)) and a bright white 10.000cd light. The lenticular halo is faintly visible around brightest light.

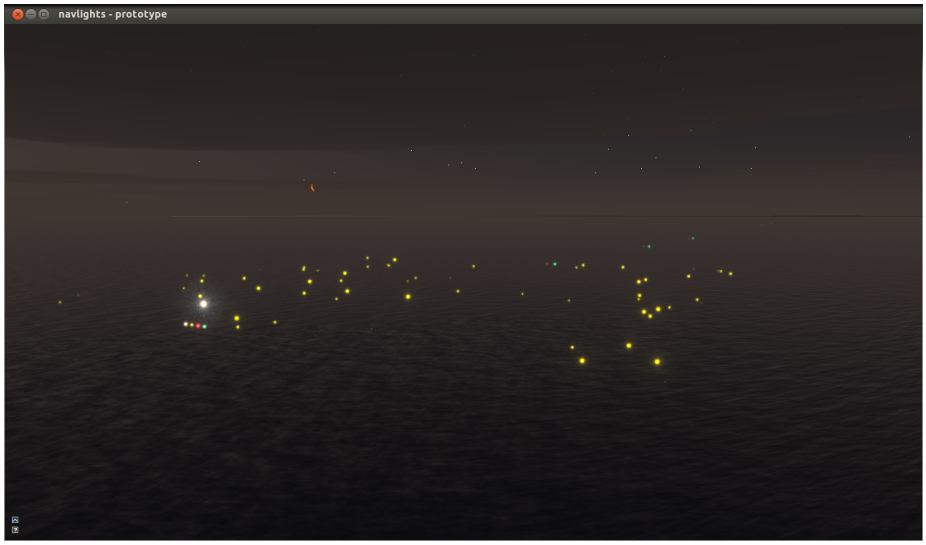


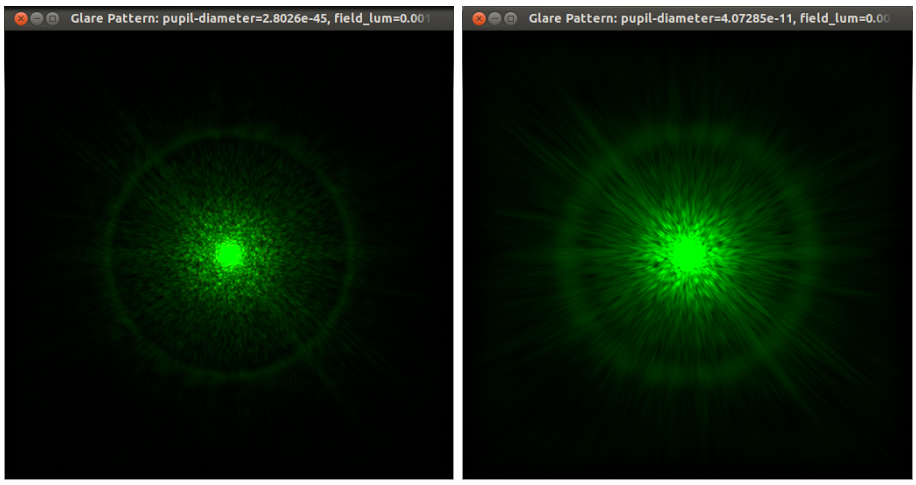
Figure 6.2: A mesopic scene with 100+ visible lights at 28 FPS on a NVIDIA Quadro 600

6.1 Glare pattern images

All glare images are shown with linear tone mapping (converted to sRGB by OpenGL/hardware) using 40000 as scale factor. The resolution is 512 by 512 pixels and no scaling is applied to the chromatic blur to better show the result.

The eye model that generates the glare pattern has many parameters. This section explores the visual effect of varying the parameters.

Simulated LED spectra The quality of the RGB approximation to mono-chromatic glare is shown in figure 6.3. The RGB approximation appears much larger than the LED approximation and in the LED approximation, the flare lines of the ciliary corona are mostly absent and instead it has the grainy look of the mono-chromatic PSF diffraction pattern of figure 4.6b.



(a) LED approximated by Gaussian at 550nm

(b) White Glare pattern filtered with RGB $(0, 1, 0)$.

Figure 6.3: Comparison of monochromatic glare pattern approximations

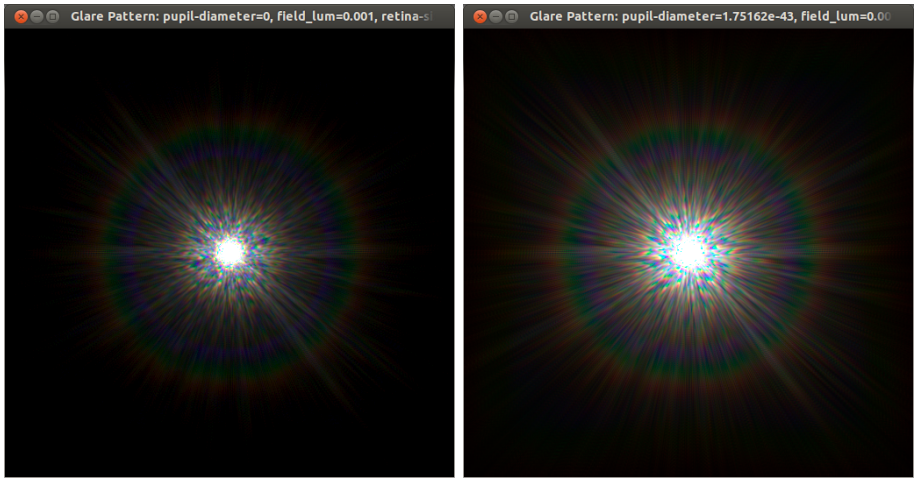
The glare from a modern white LED (spectrum shown in figure 3.2a) is shown in figure 6.4a. The relative SPD of the modern LED is empirically approximated by two Gaussian functions; one tall, narrow peaked Gaussian at 450nm and a smaller, but wider Gaussian at 560nm:

$$I(\lambda) = 0.5 \exp\left(-\frac{(\lambda - 560)^2}{2 \cdot 60^2}\right) + 0.9 \exp\left(-\frac{(\lambda - 450)^2}{2 \cdot 10^2}\right)$$

Compared to the glare from broad spectrum of Illuminant D65 in figure 6.4b, the LED glare appears colder, less dense and less bright. The lenticular halo appears to have a slight gap between the blue and green bands. This is expected because a broader spectrum will contribute more in the chromatic blur.

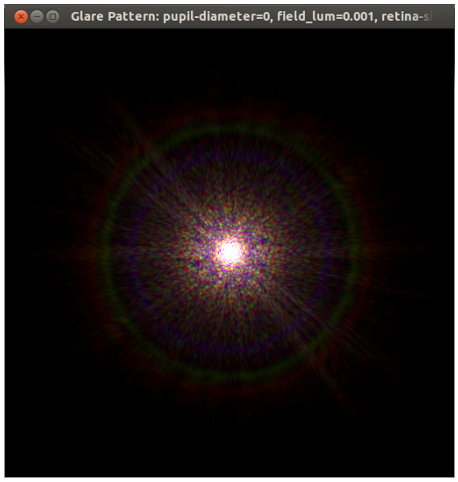
For reference, glare generated from a simulated low quality LED source using three narrow peaked Gaussian lobes is shown in figure 6.5. Instead of the smooth transition between the rings in the lenticular halo, the white LED has three distinct rings. In addition, the ciliary corona is more grainy than the previous broad spectrum whites.

White point illuminant The Illuminant E has a reddish cast compared to Illuminant D65 for spectral integration (shown in figure 6.6).

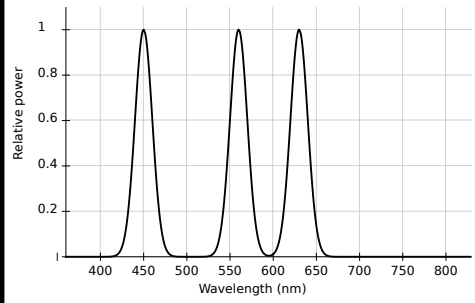


(a) White LED, using spectrum from figure 3.2a **(b)** D65, using spectrum from figure 3.9

Figure 6.4: Comparison based on White LED and D65 spectra



(a) "White" LED



(b) Poor quality "white" LED spectrum.

Figure 6.5: Glare pattern from simulated narrow peaked white LED of poor quality based on three Gaussian lobes centered around the peak HVS sensitivity.

Pupil diameter based on background intensity Variations in background intensity that guides the pupil diameter (figure 6.7) shows that the halo is not

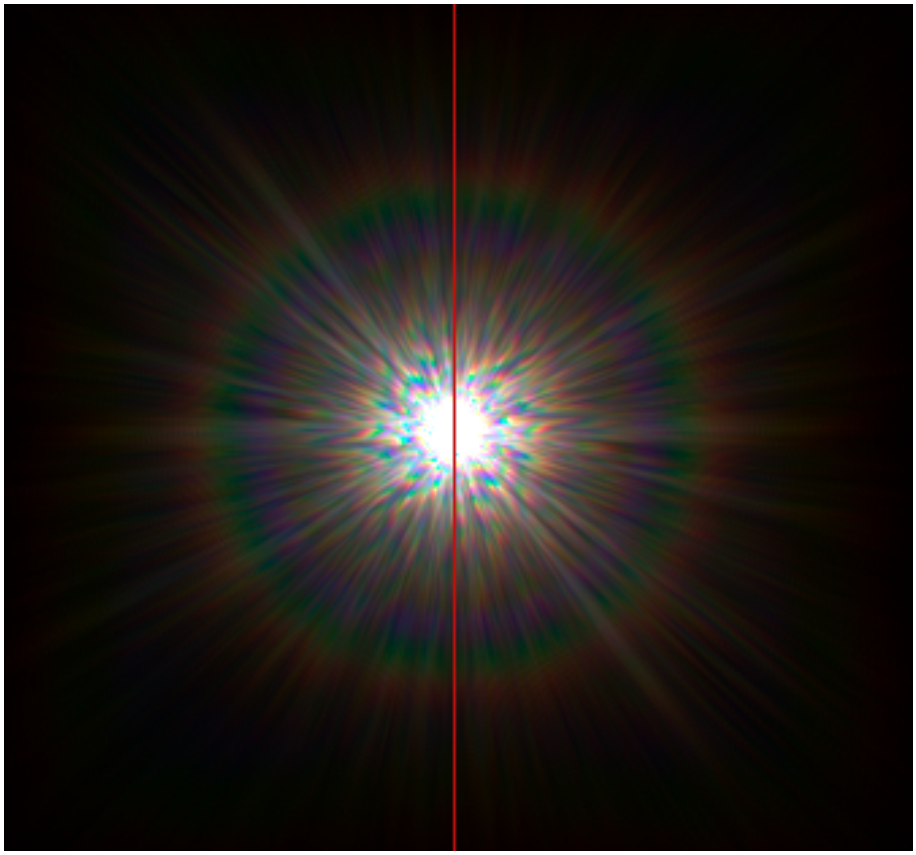


Figure 6.6: Comparison of illuminants for chromatic blur. Left half is D65 and right half is E

visible for photopic illumination levels.

Chromatic blur Varying the number of spectral samples (figure 6.8) shows that 60 samples is almost completely smooth and 15 samples exhibits stepping. 30 samples seems a good compromise.

Particle size and numbers The influence of of the particle sizes is shown in figure 6.9. Large particles generate a halo like effect (enhanced by more particles, figure 6.10) where small particles generate finer needles, the ciliary

corona (figure 6.9).

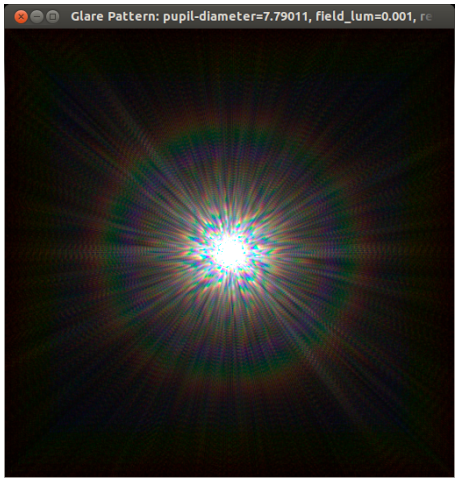
Eye lashes Eye lashes (figure 6.11) cause more extreme streaks (an effect seen more at night where the pupil is larger and more of the lashes might be visible)

Lens gratings The number of the lens gratings greatly influence the appearance as can be seen in figure 6.12. The lens gratings produce the lenticular halo (though if the number is too low (figure 6.12a), the "halo" is almost unrecognizable). As the implementation use line segments, the number, the width and the spacing of gratings are dependent on image resolution so tuning for lesser resolution might require the three parameters to be tweaked.

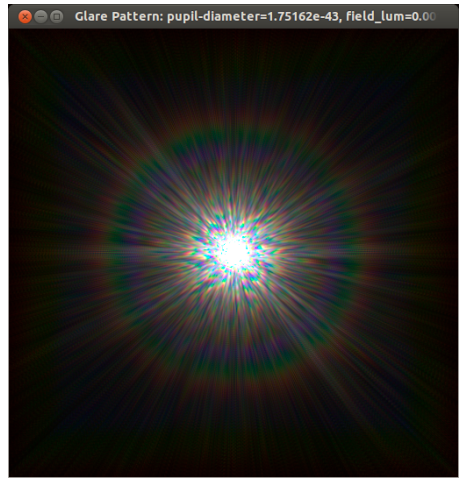
(a) 0.001cd/m^2 **(b)** 0.01cd/m^2

(c) 0.1cd/m^2 **(d)** 1cd/m^2

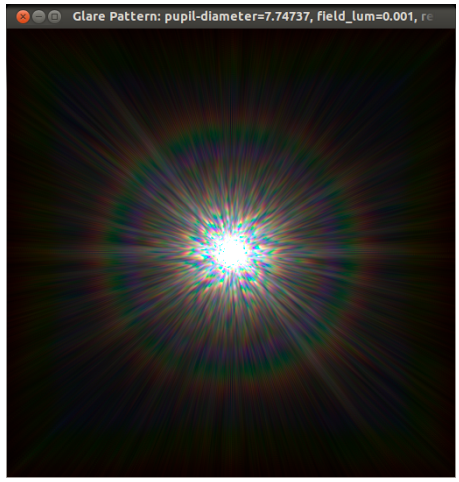
Figure 6.7: Comparison of background intensity.



(a) 15 samples

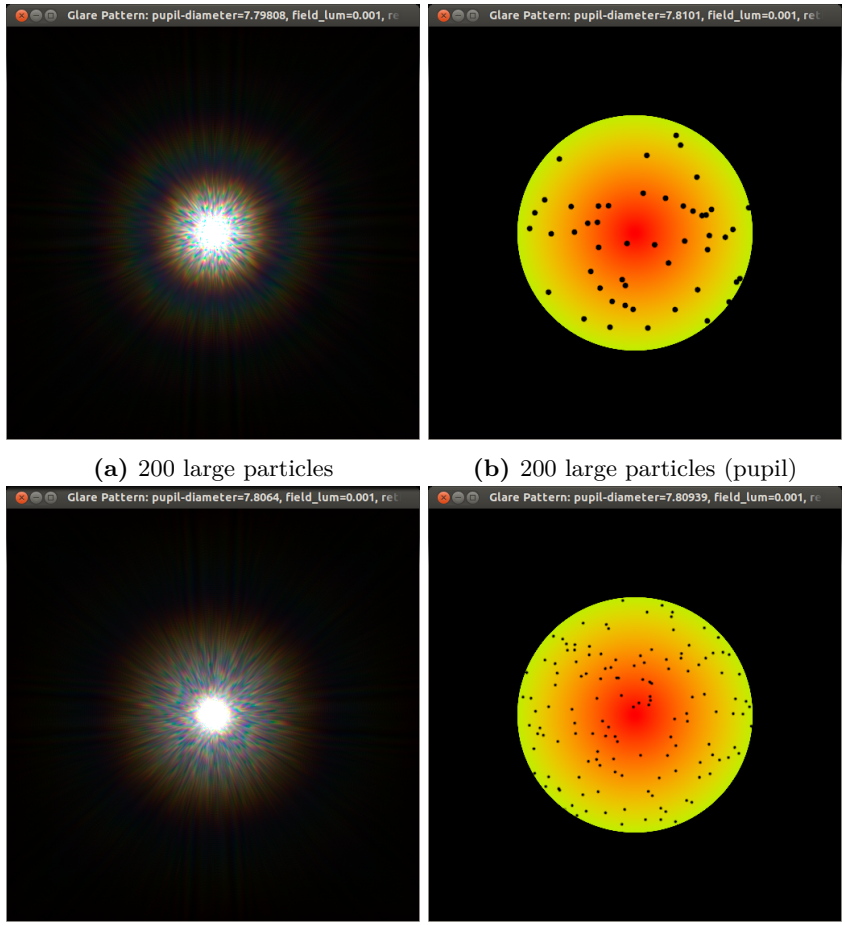


(b) 30 samples



(c) 60 samples

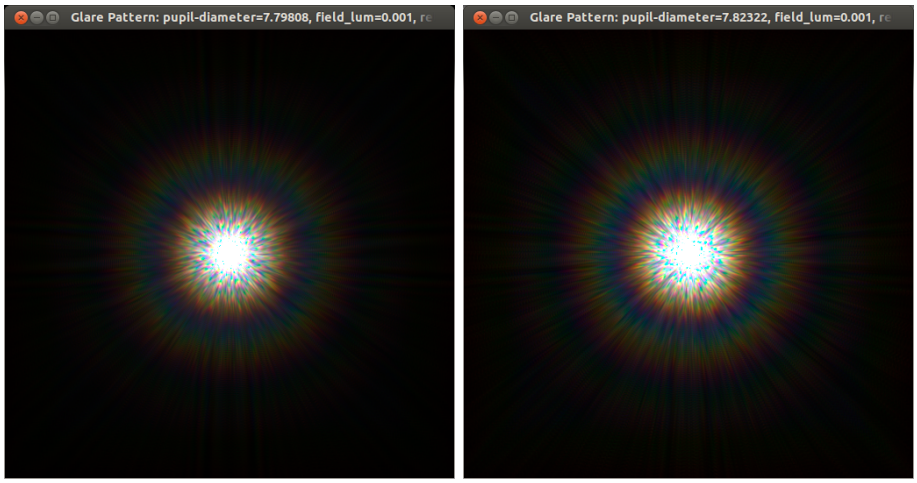
Figure 6.8: Comparison of varying spectral samples.



(a) 200 large particles **(b)** 200 large particles (pupil)

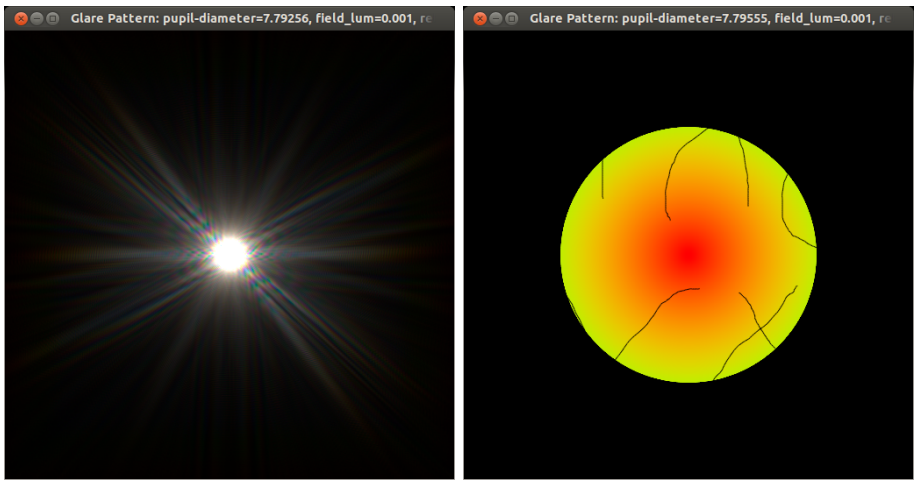
(c) 500 small particles **(d)** 500 small particles (pupil))

Figure 6.9: Comparing particle size and numbers



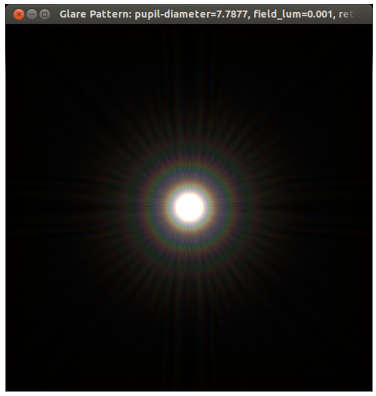
(a) 200 large particles **(b)** 400 large particles

Figure 6.10: Comparing numbers of large particles

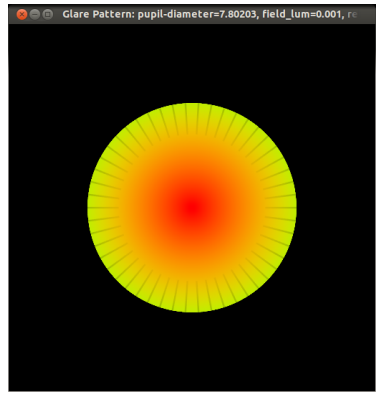


(a) Eye lashes **(b)** Eye lashes (pupil)

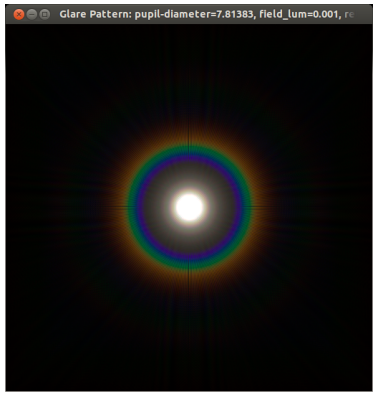
Figure 6.11: Eye lashes



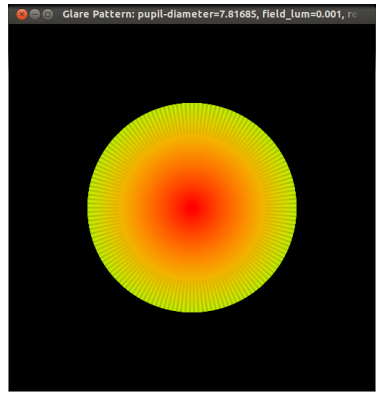
(a) 50 gratings



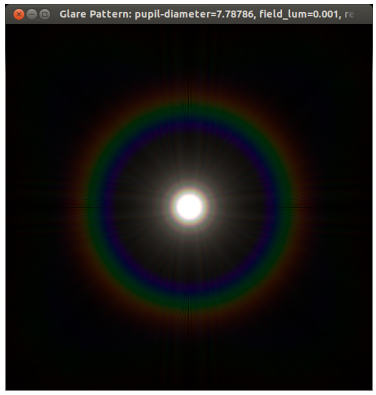
(b) 50 gratings (pupil)



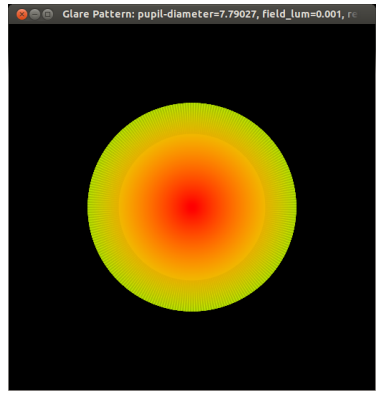
(c) 150 gratings



(d) 150 gratings (pupil)



(e) 200 gratings



(f) 200 gratings (pupil)

Figure 6.12: Comparison of varying number of lens gratings

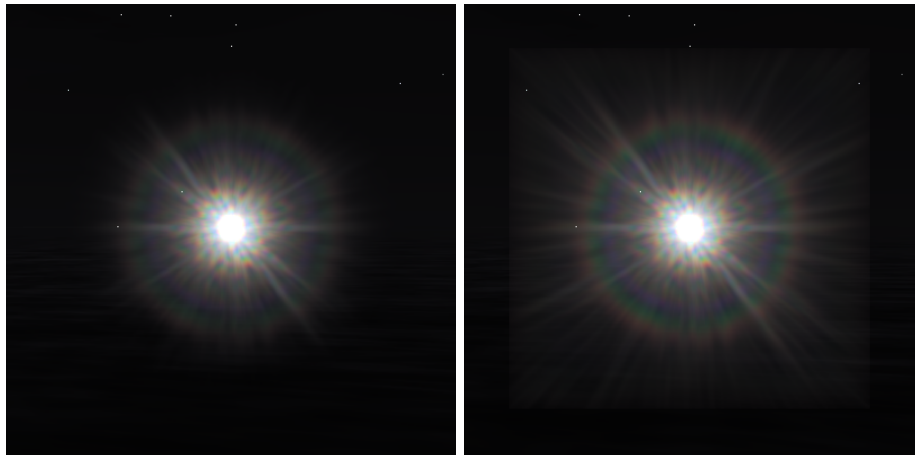
6.2 Glare pattern applied for virtual light sources

Unless otherwise stated, scenes in natural environment use the empirical Reinhard derived TMO and is rendered with 45° vertical field of view in $1280x720$ pixels (some images are cropped to better present for the result).

Sub-pixel distance attenuation Figure 6.13 demonstrates the distance attenuation. The scene consists of 10 lights spaced horizontally by 5° and with linear increasing distances from 1km to 19km (1, 3, 5, ..., 17, 19). With luminous intensity of 100cd, the ciliary corona is visible for the first sources at 1km only. Compared to empirical studies, the method does not display sharp flares in the distance.

Figure 6.13: Ten light sources with increasing distances left to right from 1km to 19km. Adaptation luminance of $1.94 \cdot 10^{-3} \mathrm{cd/m^2}$. The cutouts are enlarged 9 times with no filtering

Radial falloff The effect of using a falloff kernel for the glare billboard in a night scene is shown in figure 6.14. The second image (figure 6.14b) shows the discontinuity that using a glare billboard with limited resolution (e.g lower than the framebuffer) when contrast in light intensity and background intensity is high. The intensity of the lenticular halo is also diminished by the falloff kernel (equation 4.12).



(a) With falloff using eqn. 4.12 **(b)** Without falloff

Figure 6.14: Comparing effect of glare falloff applied to a scene, tone mapped with the non-linear Reinhard photographic operator.

Horizontal sectors A gradual transition between white and green sectors is shown in figure 6.15. Here four identical, two sectored lights are placed with an angular spacing of 15° and rotated towards the observer, showing the transition over 4 degrees.

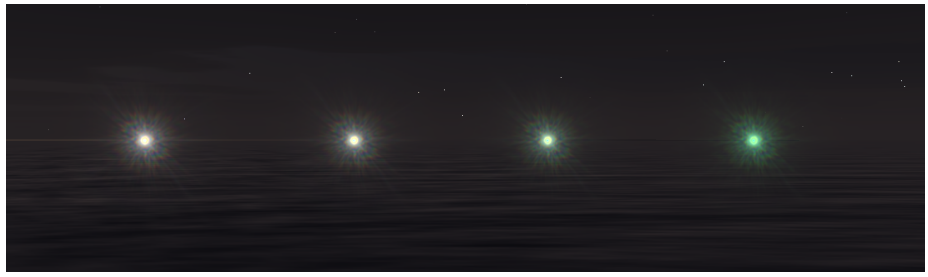


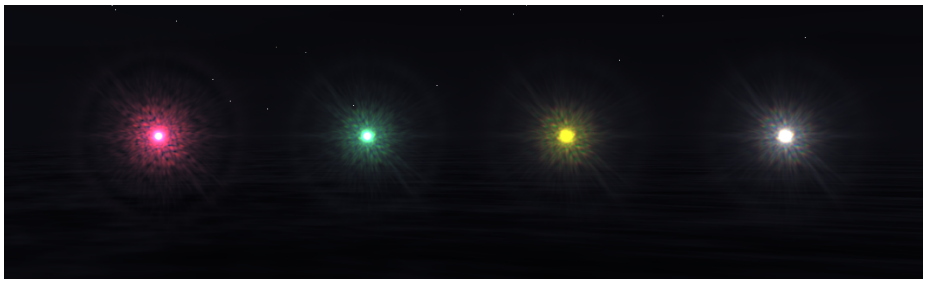
Figure 6.15: The horizontal sector overlap over 4 degrees

Chromaticity preservation In figure 6.16 the effect of hue preservation in tone mapping is shown. The red and green lights oversaturate to white in figure 6.16a and the hue preservation in xyY space and RGB space both (figure 6.16b and 6.16c) preserve the original hue.

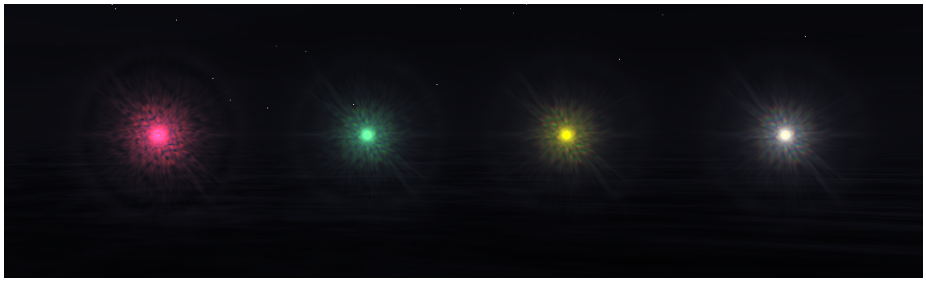
Hue preservation seems to work best in xyY space (figure 6.16b) compared to

RGB (figure 6.16c). In xyY space, the center of the glare pattern seems less over-saturated overall.

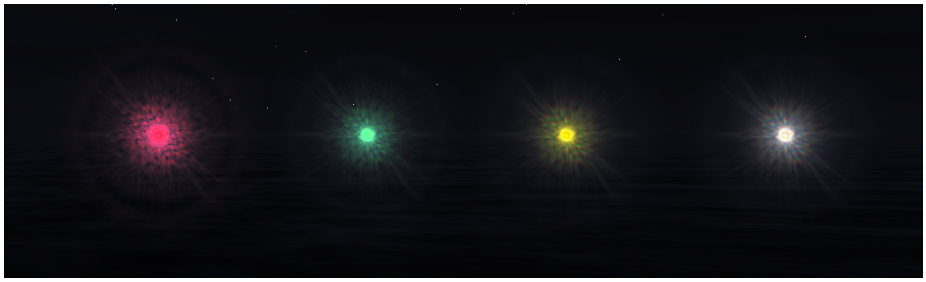
Perceptually, the white oversaturated glare pattern appears brighter than with either method of hue preservation applied.



(a) No hue preservation



(b) Hue preservation applied in xyY space



(c) Hue preservation applied in RGB

Figure 6.16: Tone over-saturation and solutions

Area light sources In figure 6.17 the effect of the precomputed area convolution is shown and compared against not using glare at all. The result clearly shows a brightness enhancing effect and that the glare is clearly larger than the actual source where at 100m the source only covers a few pixels.

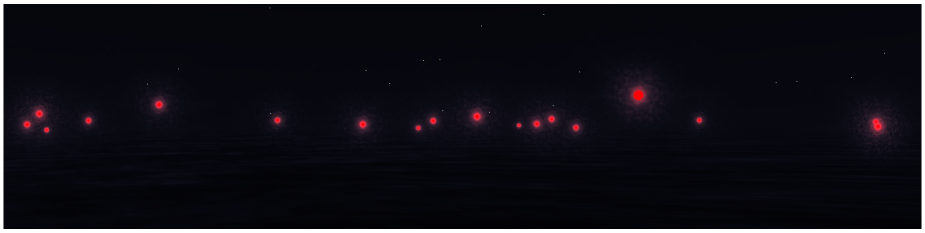
(a) No glare

(b) Glare with falloff using eqn. 4.11

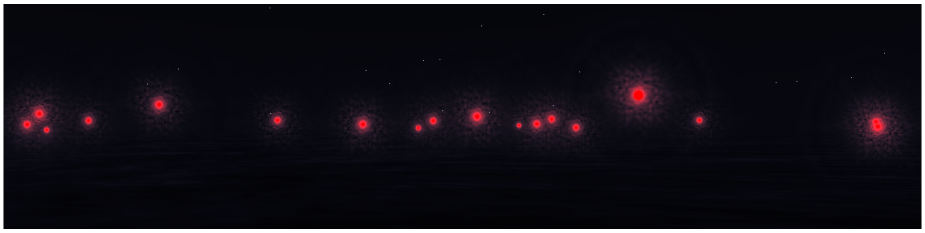
(c) Glare with falloff using eqn. 4.12

Figure 6.17: Effect precomputed area convolutions for light sources with different projected area. All four light sources are 100 cd white, placed at 10m, 20m, 50m and 100m from the observer.

Spectrum comparison in environment Figure 6.18 compares glare from red, Gaussian approximated LED spectrum with D65 filtered using RGB $(1, 0, 0)$ in a scene. As expected from figure 6.3, the RGB approximation in figure 6.18b overestimates the intensity of the glare compared to the LED approximation in 6.18a. The difference in structure, which was very visible in figure 6.3, are much harder to notice when applied in a virtual scene.



(a) Gaussian approximated LED spectrum



(b) Red RGB filtered D65 spectrum

Figure 6.18: Comparing spectrum approximations in virtual scene

Many distant lights Figure 6.19 shows four scenes with many lights. The accumulated glare of the top image clearly shows a glow around the lights. The figure shows a setup that will give worst case performance (performance is discussed in section 6.3) because the glare billboards overlap.

Time of day variations Figure 6.20 shows a comparison between moon light and star light with no cloud cover. Here, the glare is much more pronounced at star light.

Figure 6.21 shows the effect of "blueshifting" scotopic luminance levels for four 100cd lights at a distance of 100m. In general, the blueshift causes a heavy desaturation of glare, most pronounced for the red light source and the colorful appearance of the lenticular halo is lost in this setup.

Glare from 100cd lights at five different time of day for 100cd lights is shown in figure 6.22. The adaptation spans over five orders of magnitude and it is handled by the same empirical TMO. The figure demonstrates that high contrast in light intensity to background is needed for the glare to be visible. At daylight, the contrast is too low for a 100cd light to be visible against the horizon sky.

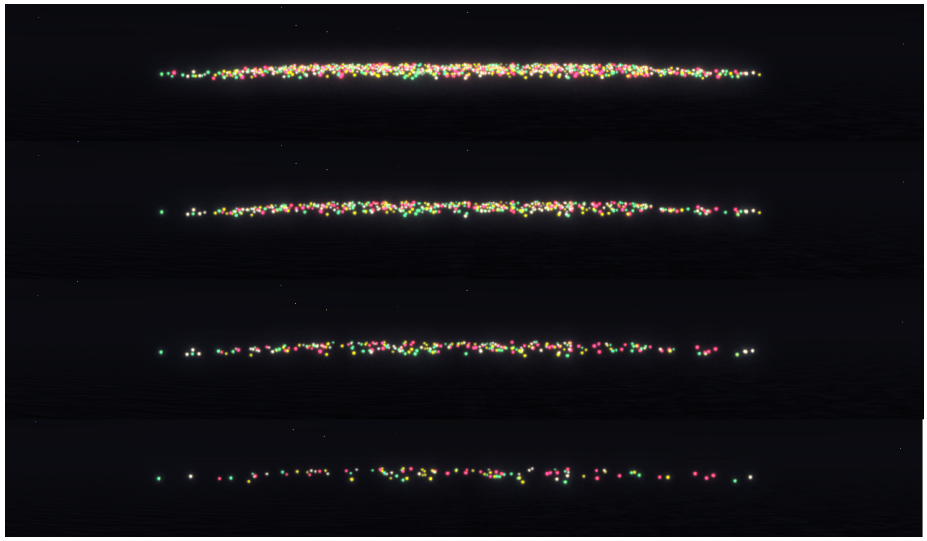
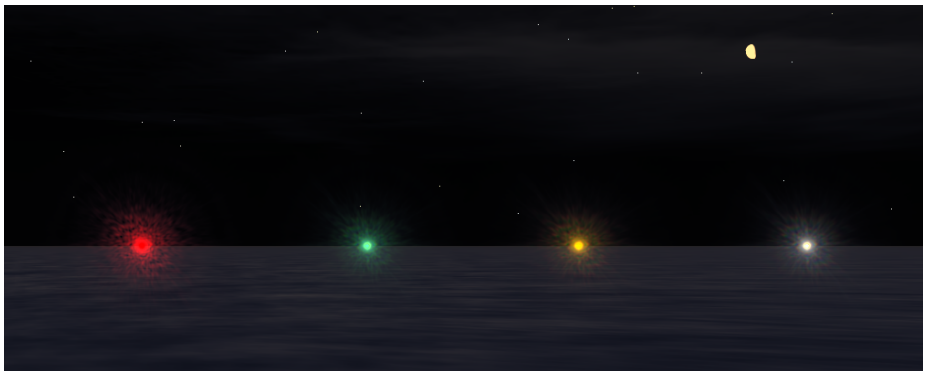
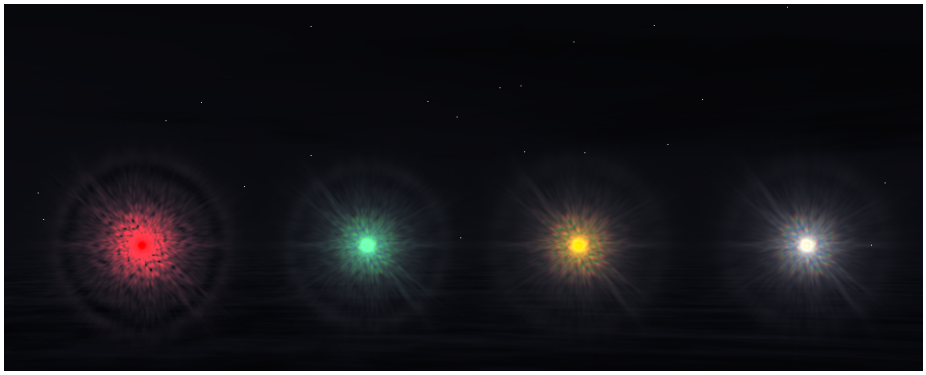


Figure 6.19: Four scenes with 100, 250, 500 and 1000 glare overlapping lights

6.2.1 Tone mapping comparison



(a) Moon light illumation. Light adaptation = 0.030cd/m^2



(b) Star light illumation. Light adaptation = 0.0010cd/m^2

Figure 6.20: Comparing glare from four 100cd lights at half moon and star light background illumination levels.

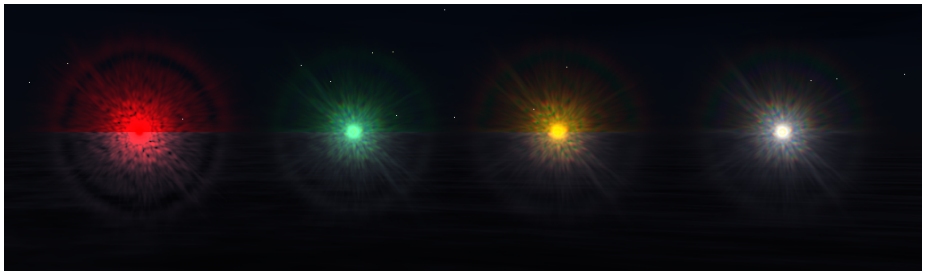
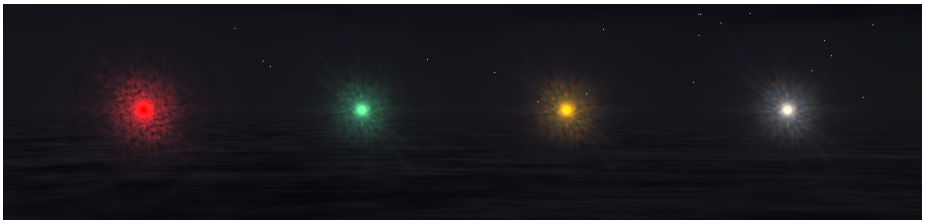
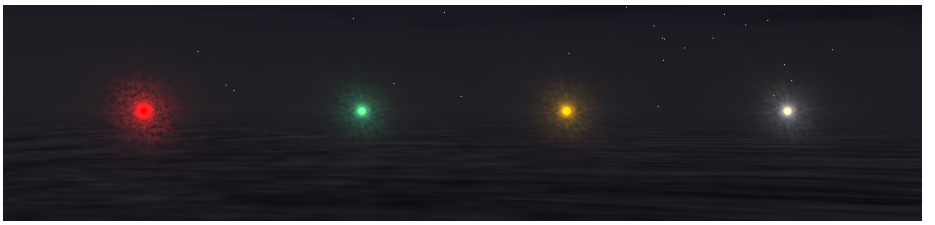


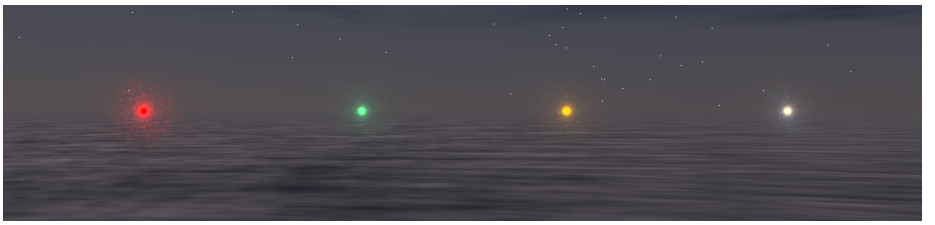
Figure 6.21: The effect of the "blueshift" for scotopic illumination levels using the empirical TMO. The upper part shows full-color and the lower part shows the blue shift of scotopic illuminance. The four 100cd light sources are placed at 100m.



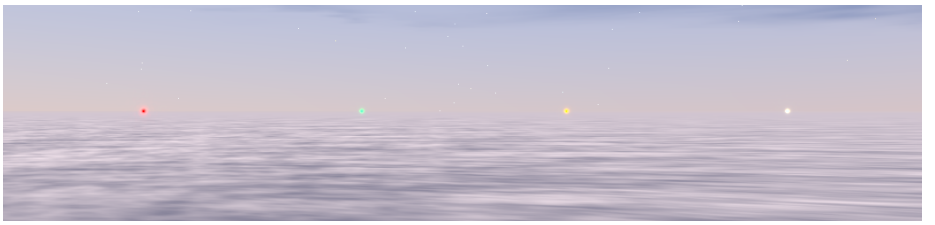
(a) Light adaptation = 0.0051cd/m^2



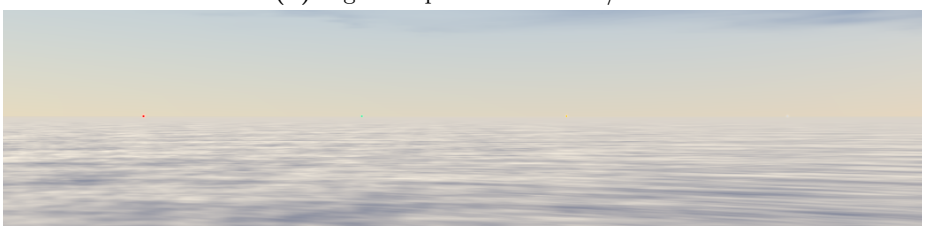
(b) Light adaptation = 0.016cd/m^2



(c) Light adaptation = 0.15cd/m^2



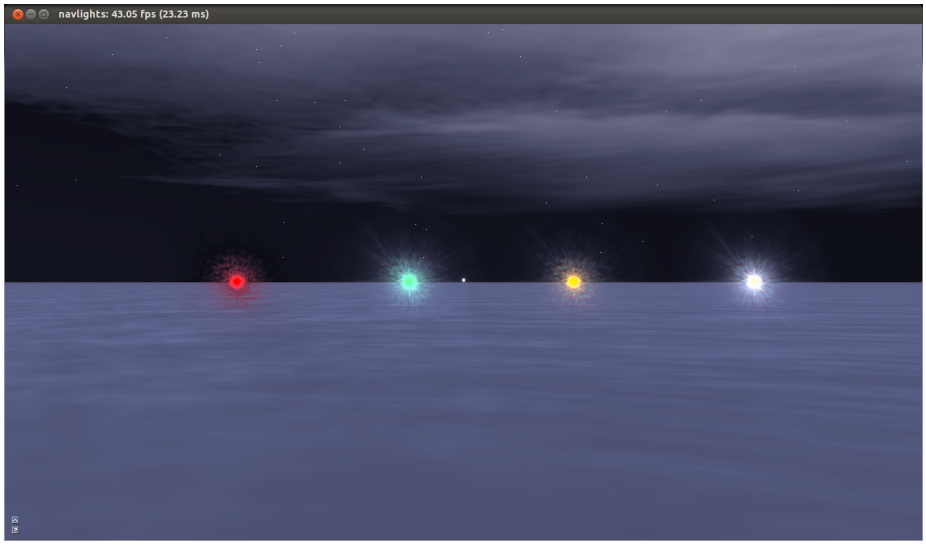
(d) Light adaptation = 24.0cd/m^2



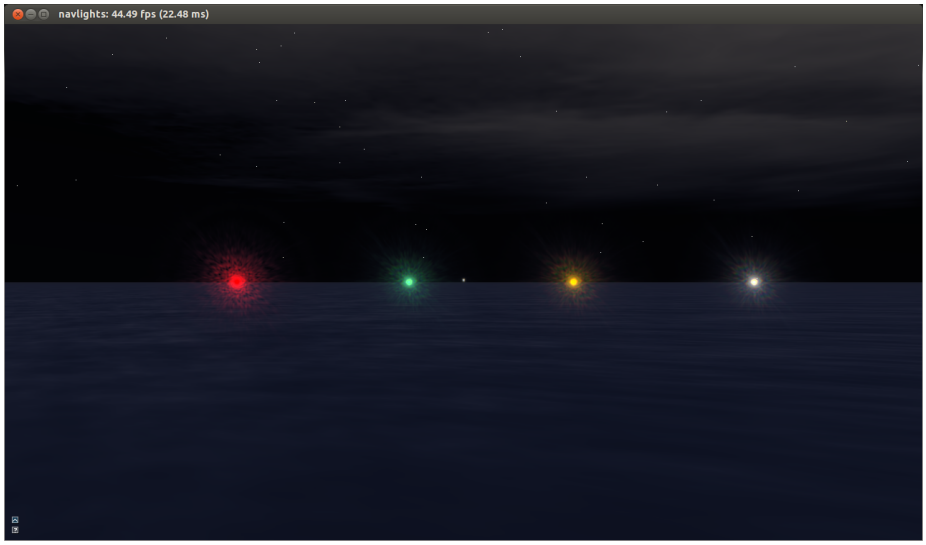
(e) Light adaptation = 1570cd/m^2

Figure 6.22: Comparing glare at different times of the day.

Figures 6.23, 6.24 and 6.25 shows comparisons between the empirical and perceptual TMO. The perceptual TMO is generally brighter and has a tendency to oversaturate.

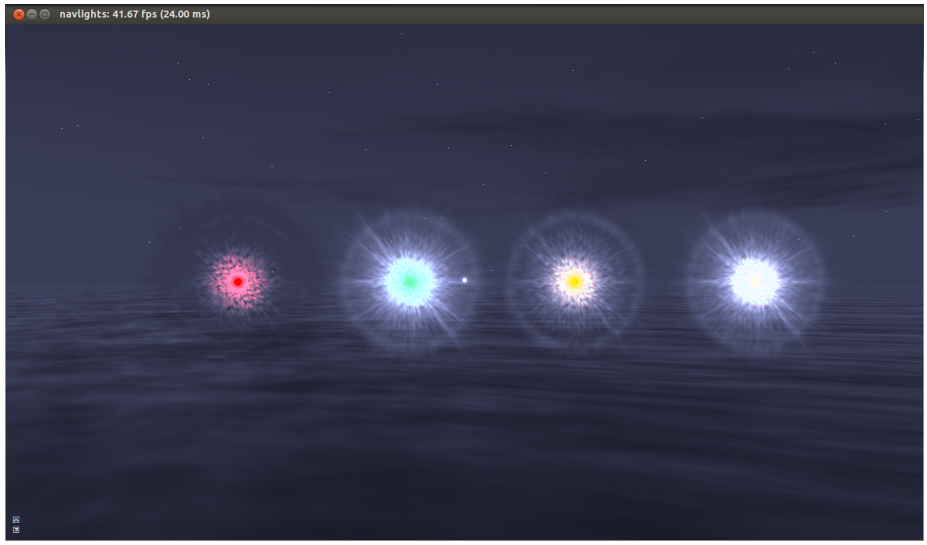


(a) Perceptual TMO

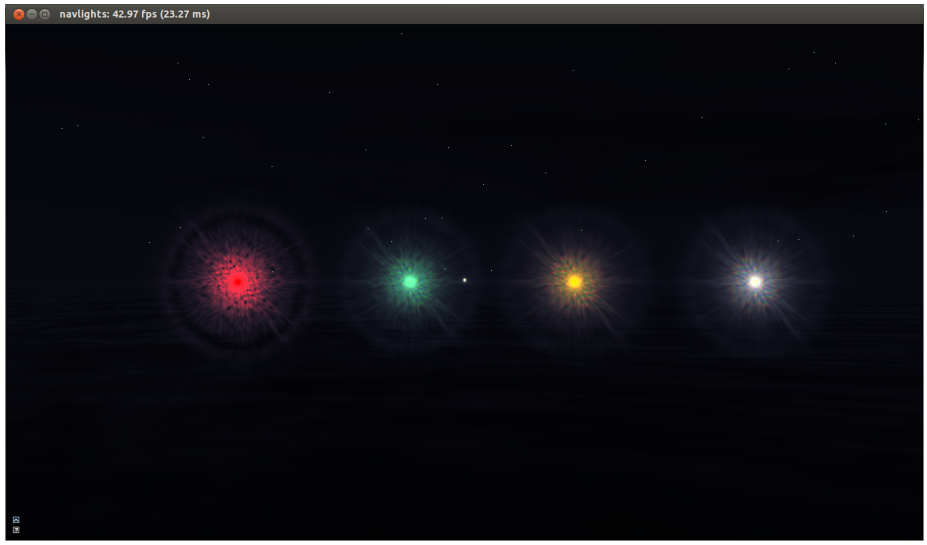


(b) Empirical TMO

Figure 6.23: Comparing tone map operators for moon light scene.



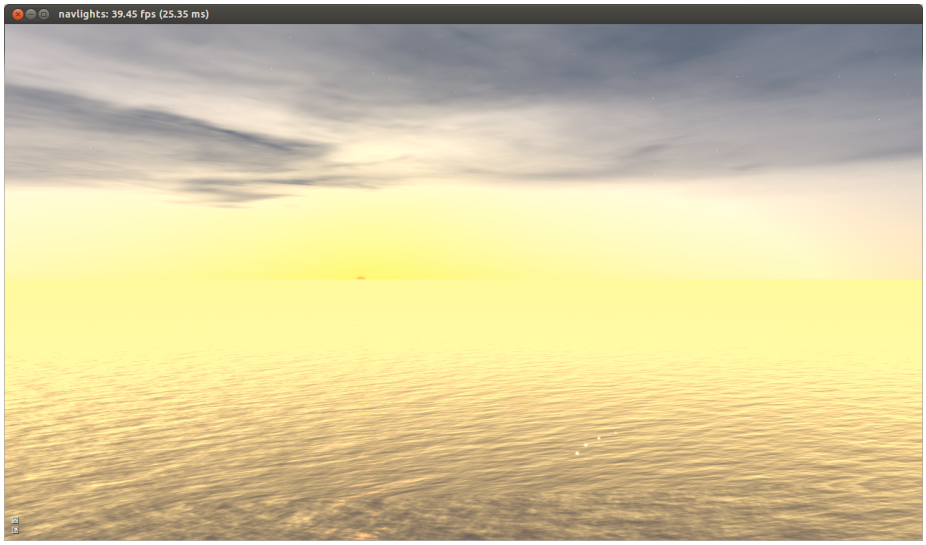
(a) Perceptual TMO



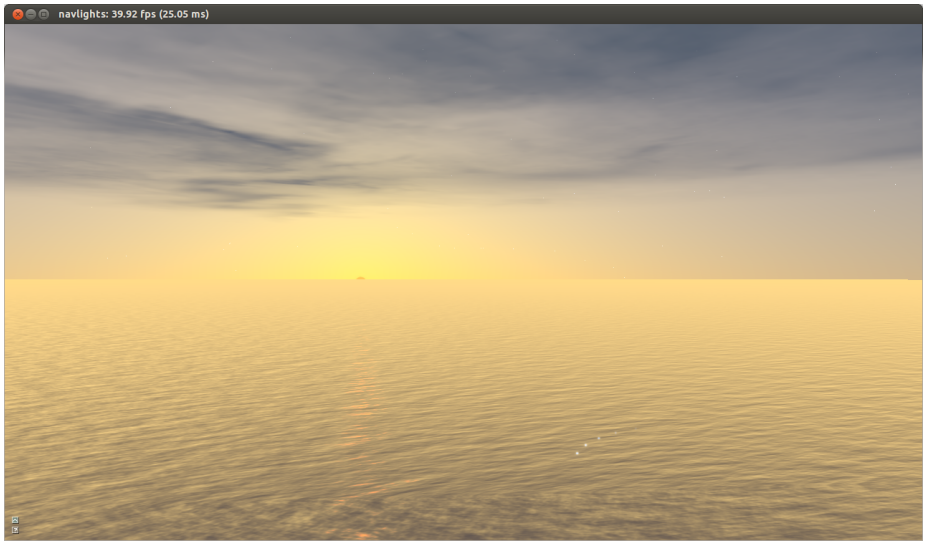
(b) Empirical TMO

Figure 6.24: Comparing tone map operators for starlight scene.

6.2.2 Glare billboards with LDR scene



(a) Perceptual TMO



(b) Empirical TMO

Figure 6.25: Comparing tone map operators for day light scene.

In this section, the results show glare patterns added after scene tone mapping using the built-in SilverLining tone mapper. The image pairs are not directly comparable in terms of scene tone mapping as using the built-in SilverLining

TMO changes code path for sun intensity and ambient light.

Figure 6.26 shows an LDR night scene enhanced with glare.

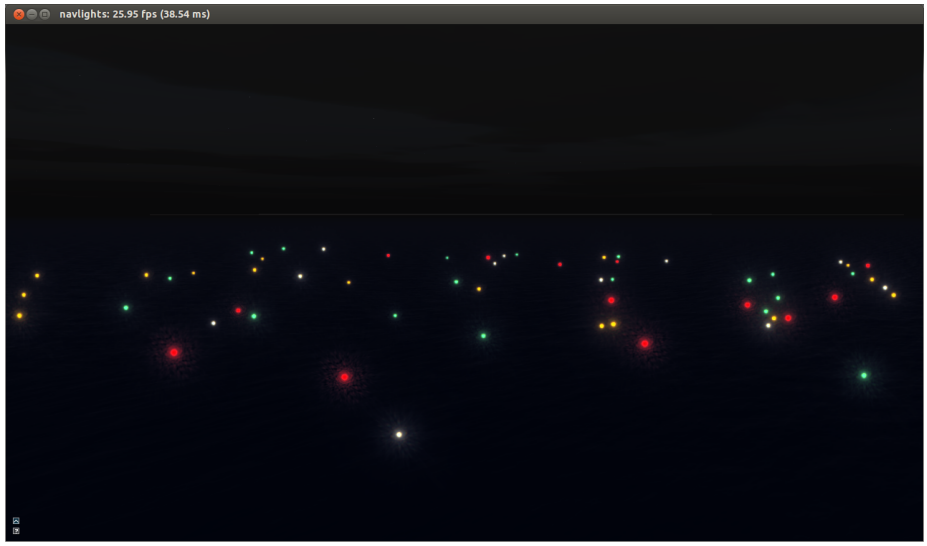
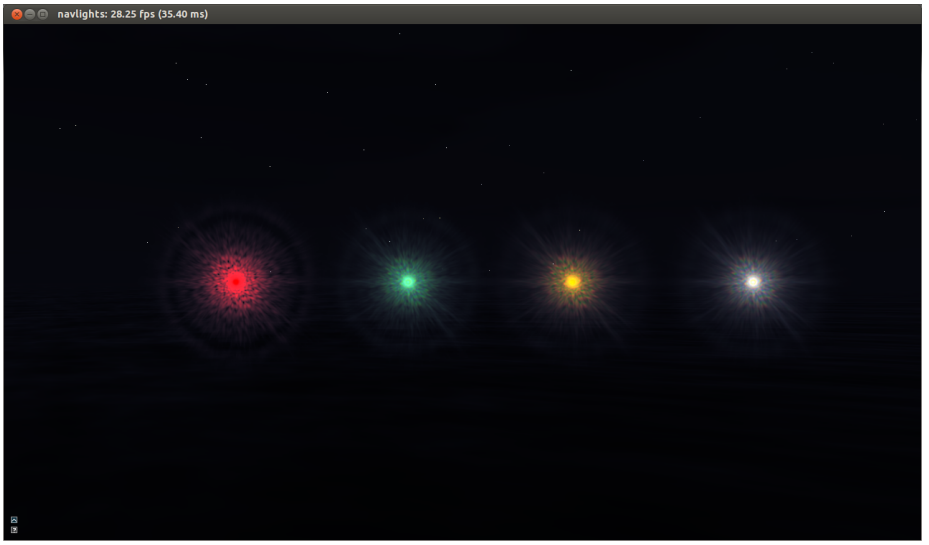
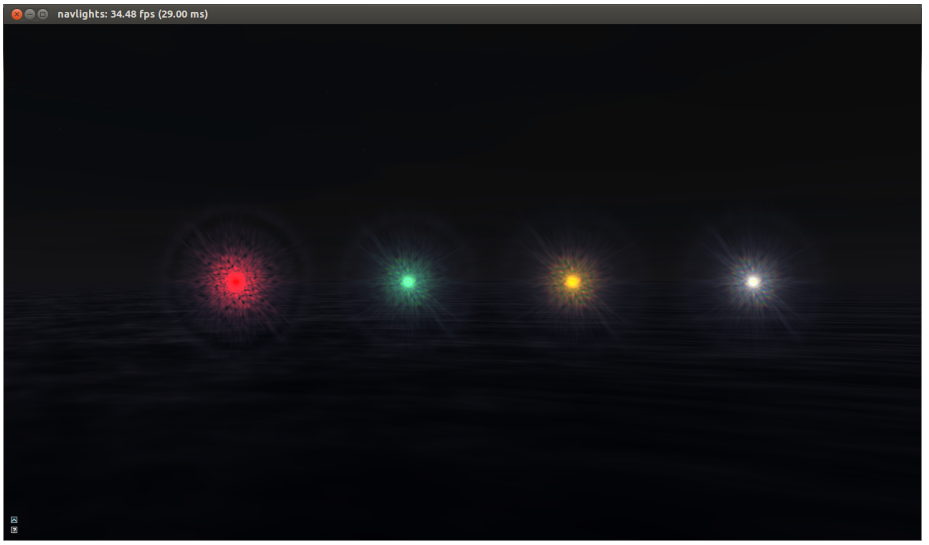


Figure 6.26: Glare added to an LDR scene using the empiric operator adapted to $L_a = 0.01\text{cd}/\text{m}^2$

For night scenes (figures 6.27 and 6.27), adding glare to an LDR scene using the empirical TMO yields comparable glare appearance. For early day scenes (figure 6.29), the appearance differ and the glare seems more washed out.

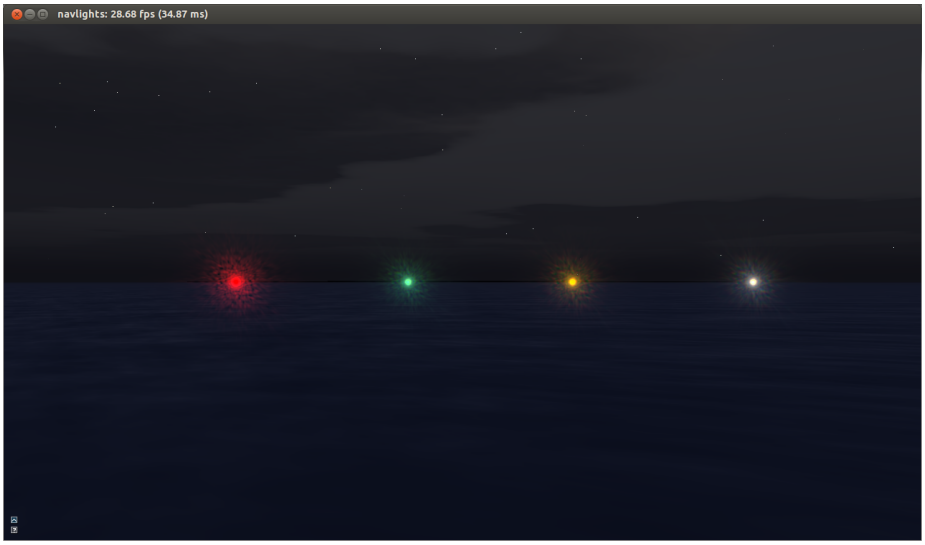


(a) HDR

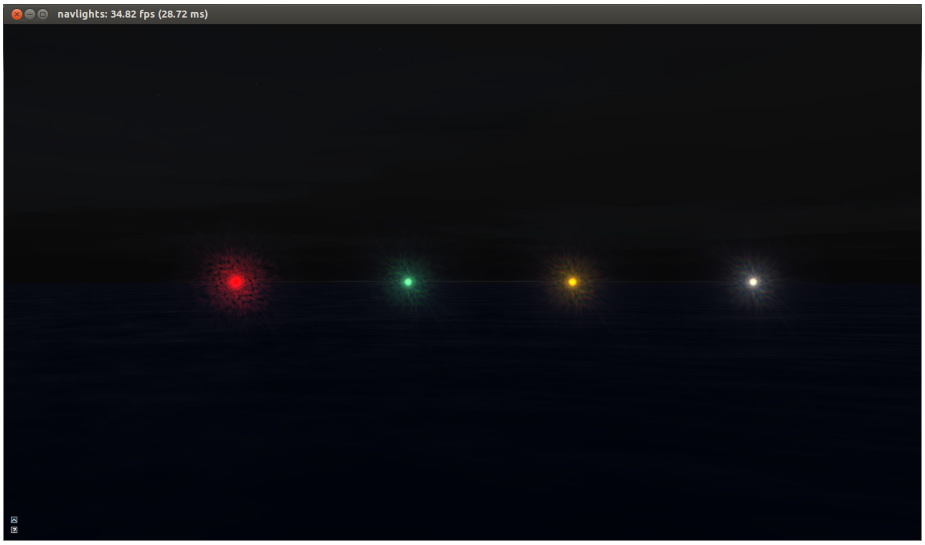


(b) LDR

Figure 6.27: Comparing tone map operators for day light scene. Light adaptation: $L_a = 0.00196\text{cd/m}^2$

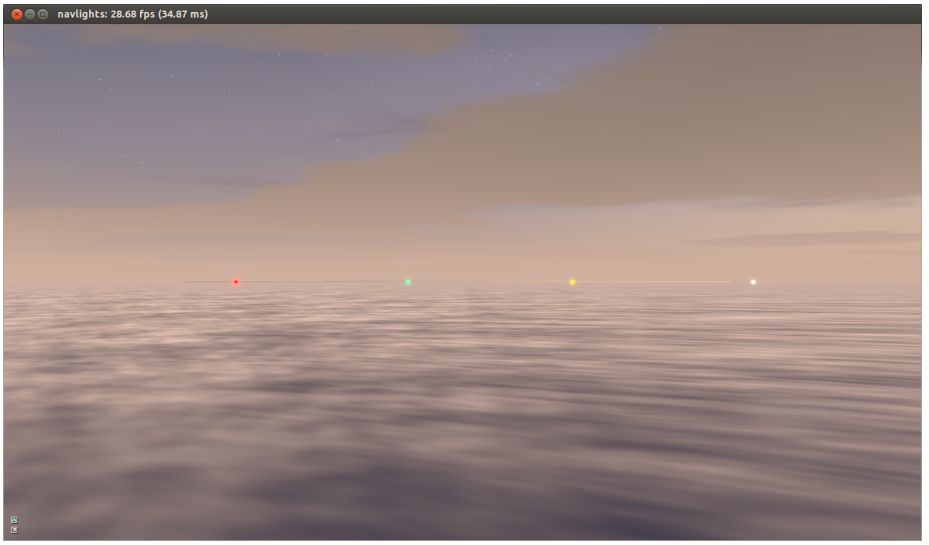


(a) HDR

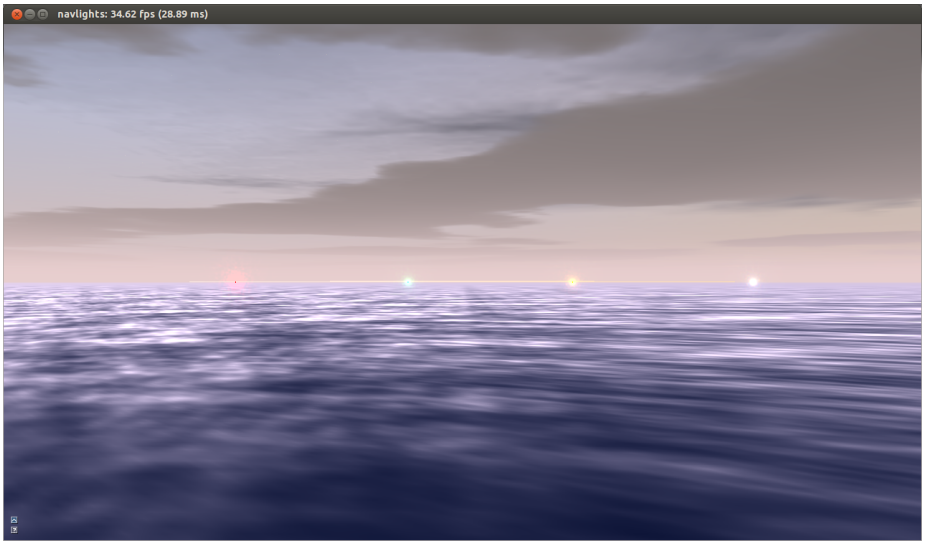


(b) LDR

Figure 6.28: Comparing tone map operators for day light scene. Light adaptation: $L_a = 0.0735\text{cd}/\text{m}^2$



(a) HDR



(b) LDR

Figure 6.29: Comparing tone map operators for day light scene. Light adaptation: $L_a = 2.05\text{cd/m}^2$

6.3 Performance Evaluation

Here I measure execution time of computing the glare pattern and applying the glare pattern in a scene with atmospheric effects from SilverLining.

Execution time is measured in milliseconds for both CPU and GPU using the third party library *VSProfileLib*[1]. It uses asynchronous OpenGL timer queries for GPU timing and high resolution CPU timer for CPU timing. The OpenGL timer queries are asynchronous to prevent pipeline stalls - which can drastically decrease performance - but as a consequence, the GPU timings are delayed one frame whereas the CPU timings are immediate and only iterated simulations have GPU timings.

The GPU timings are most interesting because the method is primarily implemented on the GPU.

For performance numbers measures in Frames Per Second (FPS), the numbers are averaged over time.

All performance measurements are done with Intel i5 2500 quad core CPU and NVIDIA Quadro 600 GPU and 8 GB ram.

6.3.1 Glare generation

The performance of the glare generation is shown in table 6.1. Looking at the GPU time, the lowend Quadro 600 can compute the glare pattern at 512x512 with three area layers (with 1, 9 and 25 pixels) in about 10ms (±2ms). The high CPU time for the FFT is a costly synchronization point between OpenGL and OpenCL.

There exists an OpenCL extension and an OpenGL extensions which would allow more finegrained synchronization, but at the moment the NVIDIA driver does not support them.

The findings show that its the chromatic blur and precomputed area convolution steps that takes roughly half the glare computation time.

[1] http://www.lighthouse3d.com/very-simple-libs/vspl/

Step	CPU (ms)	GPU (ms)
Generate Glare	13.94	10.05
Render eye model	0.27	0.61
FFT	13.21	1.57
Compute PSF	0.06	0.29
Compute RGB PSF	0.02	2.52
Compute 3 area layers	0.06	3.26

Table 6.1: Break down of major components of the glare generation. Total render time: CPU 13.94 ms/ GPU 10.05ms

Precomputed convolution performance scaling The performance scaling of precomputing convolution is shown in figure 6.30. Performance quickly becomes prohibitive on the Quadro 600 where only three or four area layers can be computed in real-time. For the "needle" pattern to disappear, at least a radius of 10 pixels is needed (figure 4.9) so if the close up glare is to be simulated, dynamic glare simulation has to be either disabled, amortized over many frames or simply precompute at load-time the layers that are too expensive to update real-time.

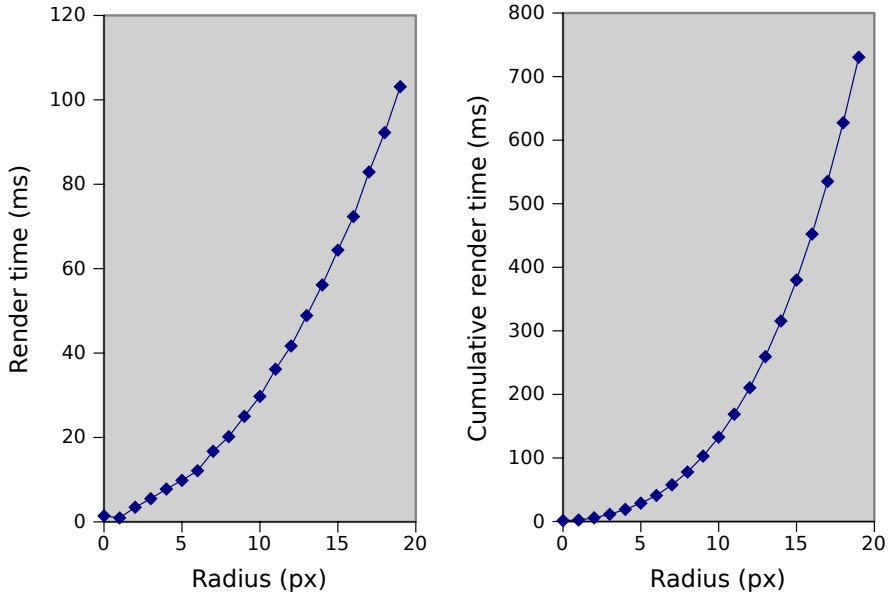
6.3.2 Glare pattern application

The performance of scaling the number of lights from 100 to 1000 is shown in table 6.2. Here the light sources are placed closely together so that the glare billboards overlap (shown in figure 6.19).

Number of lights	Average FPS	Average frame time
0	48.80 fps	20.49 ms
100	29.70 fps	33.67 ms
250	21.03 fps	47.55 ms
500	14.49 fps	69.01 ms
1000	8.87 fps	112.74 ms

Table 6.2: Worst case performance for increasing number of lights, including glare generation and environment

For 100 visible lights, the prototype implementation renders at $\sim$ 30 frames



(a) Area layer render time scaling **(b)** Cumulative render time scaling

Figure 6.30: Performance scaling of precomputing convolution of light sources.

per second, but for thousand visible lights, the frame rate drops to 8.87 FPS. The performance scales linearly with the number of lights as shown in figure 6.31. Rendering a scene with 1000 non-visible lights has virtually the same performance as not rendering the lights at all, so brute force culling lights in the Geometry Shader is efficient and the vertex data submission time is insignificant.

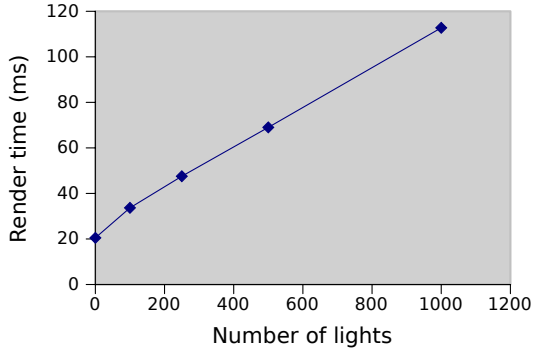


Figure 6.31: Plot of performance measurements from table 6.2

Chapter 7

Conclusion

The work of this report has showed how the appearance of ATON lights can be modeled using physical intensities. The results show consistent appearance for supra pixel sized lights, sub pixels lights and the transition between them.

The method was implemented as a prototype separate from the SimFlex simulator to allow more freedom with regards to HDR rendering. To provide an outdoor environment for the lights, SilverLining by SunDog Software was used and a planar surface for water.

Two different tone mapping and contrast reduction techniques were used to show the HDR simulation. The empirical approach was superior to the perceptual. The TVI curves also seems unsuited for a single global scale factor.

The method allows for modeling the light source emission using the relative SPD, though for performance reasons all lights currently share the same spectrum and color is modeled by CIE 1931 xy chromaticity as computing the glare for both tungsten filament and LED would take 50ms (five sector colors times two light types times 10ms to compute) per frame to compute. But the results show that using a single white spectrum and then use RGB color filtering is a good approximation in practice.

In absence of a psychophysical study, I subjectively judge the glare applied by

method to increase the brightness of the light sources.

I find the visual quality of the method highest in twilight scenes where the contrast between the background and lights is not too high (compared to night scenes). When the contrast is too high, the limited resolution and the glare intensity falloff looks a bit too unrealistic.

The method runs almost exclusively on the GPU. The performance of the implementation allows the glare pattern to be simulated, and applied to 10-100 directly visible lights in real-time (30 Hz) on a low-end NVIDIA Quadro 600 graphics card. By tuning parameters such as the number of precomputed area layers, dynamic glare generation and number of light spectra simulated, performance and quality can be adjusted target hardware.

In my implementation, the lenticular halo is only clearly visible for light sources covering many pixels (i.e. relatively close to the observer) or light sources having a very high intensity.

My method can show a physically based approximation of glare for sub pixel lights which pure convolution cannot do unless resolution is vastly increased.

Due to my subjective experience of not seeing the lenticular halo, I find using the filter kernel on a smaller window of the PSF and decreasing the intensity a good trade off.

The linear perceptual operator is less stable compared to the non-linear empiric operator.

The method for rendering glare is independent of the actual glare pattern and tone mapping.

Integration with SimFlex My prototype implementation is separate from SimFlex, which allowed me maximal flexibility. Adding proper HDR to SimFlex might be a large engineering and quality assurance task, with modifying surface shaders to use physically based BRDF's and ensure a fully gamma correct assets pipeline.

The results show that glare can be added to an LDR renderer by rendering tone mapped glare images additively onto the LDR framebuffer. The adaptation for the glare images would be empirically chosen for best visual results given time of day. Short term, this would probably be the best approach for integration with SimFlex.

The billboard based glare rendering is easily adapted to multichannel rendering because view frustum culling for lights visible on other channels can be disabled.

7.1 Future work

Better integration with participating media so in-scattered light is taken into account instead of only extinction (out-scatted and absorbed light).

Integrate a general glare post-process effect for non-light sources (such as specular reflections) would enhance general image quality, though good results rely on HDR scene intensities.

The performance of the precomputed convolution layers might benefit for an improved parameterization that decreases the number of layers without sacrificing image quality.

Tone mapping on a 360 degree simulation needs further study on how to choose adaptation luminance in a consistent way so that, for instance, the sun does not cause overexposure. Network synchronization is also a minor issue, but as the eye adapts over time, introducing latency might not be an issue. The adaptation luminance choice might also be improved by a more sophisticated algorithm such that the outliers luminances are ignored.

The two tone map operators presented here are simple and do not properly take into account the high black levels of projectors. Further investigation of the tone map operators presented in the background chapter could allow the simulation to better predict the visibility of the light sources when taking the display device and observer into account.

Further work could adapt tone mapping to the actual color gamut of the displays if they support larger gamuts than sRGB. In general, color perception and adaptation was largely ignored in this project and could be improved.

For finding the optimal glare appearance (for instance whether to display the lenticular halo at all) a psychophysical study could be conducted.

In particular I would like to investigate how the method can produce more convincing results for distant lights, where in real-life I have observed that bright distant lights produce small sharp flares.

Appendix A

Notes from a meeting with the Danish Maritime Authority

I visited Jørgen Royal Petersen at DMA (Danish Maritime Authority), the department of the danish government responsible for maintenance of aids to navigation in Denmark to get the practical point of view in addition to the IALA recommendations.

Meeting at Søfartsstyrelsen in Korsør about light houses and beacons

- Only a few selected light houses are constantly lit, most are turned off at day
- Beacons and buoys have a photometer that turns off the light at day to conserve energy.
- Beacons and buoys are painted with fluorescent paint that matches the lit color.
- For colored beacons, monochromatic LED lights are used.

- Light houses with angular sectors uses color filters, which means the white light is brighter
- As the light source inside a light house has a cylindrical form, there will be a gradual shift from sector to sector, which, by spec, is at most 0.1[1] degrees.
- PEL light houses do not have this gradual shift.
- The light is focused using a lens with a Fresnel profile
- The lights have both a horizontal and vertical profile.
- For conventional lights (such as halogen) the 360 deg horizontal profile is irregular because of shadowing in the light filament. Furthermore, if the light has reserve bulbs, they may cause shadowing as well.
- The buoys are tied to a concrete block with a chain so the water current may tilt the beacon. This creates a need for a vertical light profile that ensures the beacon can be seen from low or high angles. In Denmark this is a particular issue.
- Navigation charts note the nominal range of lights under specific atmospheric conditions. Allard's Law can be used to find the luminous intensity I of the light source. $E = \frac{I}{d^2}T^d$ where E is the incident illuminance , d is the distance and T is the atmospheric transmissitivity. For night time lights, IALA sets the standard required illuminance E at the nominal range to be $2 \cdot 10^{-}7lux$, but note the intensity should also compensate for background illuminance (e.g. "substantial" background lighting should increase the required illuminance by factor 100)

[1] Or 10 deg

Bibliography

[AMHH08] Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. *Real-Time Rendering*. A K Peters, Ltd, third edition, 2008.

[App12] Apple. OpenCL FFT, 2012. [Online; accessed 20-September-2012]. URL: http://developer.apple.com/library/mac/#samplecode/OpenCL_FFT/Introduction/Intro.html.

[Car06] F. Carucci. Hdr meets black & white 2, 2006. Presented at Game Developers Conference. URL: http://www.slideshare.net/fcarucci/HDR-Meets-Black-And-White-2-2006.

[DD00] F. Durand and J. Dorsey. Interactive tone mapping. In *Eurographics Workshop on Rendering*, pages 219–230, 2000.

[DD02] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 257–266. ACM, 2002.

[EHK+07] W. Engel, J. Hoxley, R. Kornmann, N. Suni, and J. Zink. *Programming Vertex, Geometry and Pixel Shaders*. Charles River Media, 2007. URL: http://content.gpwiki.org/index.php/D3DBook:Book_Cover.

[EMP+03] D.S. Ebert, F.K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and modeling: a procedural approach*. Morgan Kaufmann, third edition, 2003.

[FPSG96] J.A. Ferwerda, S.N. Pattanaik, P. Shirley, and D.P. Greenberg. A model of visual adaptation for realistic image synthesis. In *Pro-*

ceedings of the 23rd annual conference on Computer graphics and interactive techniques, pages 249–258. ACM, 1996.

[GWWH05] N. Goodnight, R. Wang, C. Woolley, and G. Humphreys. Interactive time-dependent tone mapping using programmable graphics hardware. In *ACM SIGGRAPH 2005 Courses*, page 180. ACM, 2005.

[Hab10] J. Hable. Uncharted 2: Hdr lighting, 2010. Presented at Game Developers Conference. URL: http://www.slideshare.net/ozlael/hable-john-uncharted2-hdr-lighting.

[Hun52] R.W.G. Hunt. Light and dark adaptation and the perception of color. *JOSA*, 42(3):190–199, 1952.

[IAL08a] IALA. E-200-1: Colors. IALA Recommendation on Marine Signal Lights, E-200 1, IALA, December 2008. URL: http://www.iala-aism.org/iala/publications/documentspdf/doc_227_eng.pdf.

[IAL08b] IALA. E-200-2: Calculation, definition and notation of luminous range. IALA Recommendation on Marine Signal Lights, E-200 2, IALA, December 2008. URL: http://www.iala-aism.org/iala/publications/documentspdf/doc_228_eng.pdf.

[IFM05] P. Irawan, J.A. Ferwerda, and S.R. Marschner. Perceptually based tone mapping of high dynamic range image streams. In *Proceedings of the Eurographics Symposium on Rendering*, pages 231–242, 2005.

[JDD+01] H.W. Jensen, F. Durand, J. Dorsey, M.M. Stark, P. Shirley, and S. Premože. A physically-based night sky model. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 399–408. ACM, 2001.

[Kaw05] M. Kawase. Practical implementation of high dynamic range rendering, 2005. Presented at Game Developers Conference.

[KFNJ04] H. Kolb, E. Fernandez, R. Nelson, and B. Jones. Webvision: The organization of the retina and visual system–john moran eye center. *University of Utah*, 2004. URL: http://webvision.med.utah.edu/.

[KJF07] J. Kuang, G.M. Johnson, and M.D. Fairchild. icam06: A refined image appearance model for hdr image rendering. *Journal of Visual Communication and Image Representation*, 18(5):406–414, 2007.

[KMN+05a] M. Kakimoto, K. Matsuoka, T. Nishita, T. Naemura, and H. Harashima. Glare generation based on wave optics. In *Computer Graphics Forum*, volume 24, pages 185–193. Wiley Online Library, 2005.

[KMN+05b] M. Kakimoto, K. Matsuoka, T. Nishita, T. Naemura, and H. Harashima. Glare simulation and its application to evaluation of bright lights with spectral power distribution. In *ACM SIGGRAPH 2005 Posters*, page 42. ACM, 2005.

[KMS05] G. Krawczyk, K. Myszkowski, and H.P. Seidel. Perceptual effects in real-time tone mapping. In *Proceedings of the 21st spring conference on Computer graphics*, pages 195–202. ACM, 2005.

[LH07] D. Luebke and G. Humphreys. How gpus work. *Computer*, 40(2):96–100, 2007.

[LRP97] Gregory Ward Larson, Holly Rushmeier, and Christine Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4):291–306, October 1997.

[Luk06] C. Luksch. Realtime hdr rendering. *Projektpraktikum mit Bakkalaureatsarbeit (2006/07)*, 2006.

[MDK08] R. Mantiuk, S. Daly, and L. Kerofsky. Display adaptive tone mapping. In *ACM Transactions on Graphics (TOG)*, volume 27, page 68. ACM, 2008.

[MMG06] J. Mitchell, G. McTaggart, and C. Green. Shading in valve's source engine. In *ACM SIGGRAPH 2006 Courses*, volume 6, pages 129–142, 2006.

[Ngu07] H. Nguyen. *Gpu Gems 3*. Addison-Wesley Professional, 2007. URL: http://http.developer.nvidia.com/GPUGems3/gpugems3_pref01.html.

[NKON90] E. Nakamae, K. Kaneda, T. Okamoto, and T. Nishita. A lighting model aiming at drive simulators. *ACM SIGGRAPH Computer Graphics*, 24(4):395–404, 1990.

[Per48] F.H. Perrin. Whose absorption law? *JOSA*, 38(1):72–74, 1948.

[Per12] Emil Persson. Graphics gems for games - findings from avalanche studios. In *ACM SIGGRAPH 2012 Courses*, 2012.

[Pet12] Jørgen Steen Royal Petersen. Personal communication, 2012. Danish Maritime Authority.

[PH04] Matt Phar and Greg Humphreys. *Physically Based Rendering.* Morgan Kaufmann Publishers, first edition, 2004.

[PTYG00] S.N. Pattanaik, J. Tumblin, H. Yee, and D.P. Greenberg. Time-dependent visual adaptation for fast realistic image display. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 47–54. ACM Press/Addison-Wesley Publishing Co., 2000.

[RAM+07] B. Roch, A. Artusi, D. Michael, Y. Crisanthou, and A. Chalmers. Interactive local tone mapping operator with the support of graphics hardware. In *ACM Proceedings, SCCG Conference*, 2007.

[RD05] E. Reinhard and K. Devlin. Dynamic range reduction inspired by photoreceptor physiology. *Visualization and Computer Graphics, IEEE Transactions on*, 11(1):13–24, 2005.

[Rec90] I. Recommendation. 709, basic parameter values for the hdtv standard for the studio and for international programme exchange (1990)[formerly ccir rec. 709]. *Geneva: ITU*, 1990.

[RIF+09] T. Ritschel, M. Ihrke, J.R. Frisvad, J. Coppens, K. Myszkowski, and H.P. Seidel. Temporal glare: Real-time dynamic simulation of the scattering in the human eye. In *Computer Graphics Forum*, volume 28, pages 183–192. Wiley Online Library, 2009.

[RSSF02] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. *ACM Transactions on Graphics*, 21(3):267–276, 2002.

[RWP+10] Erik Reinhard, Greg Ward, Sumanta Pattanaik, Paul Devebec, Wolfgang Heidrich, and Karol Myszkowski. *High Dynamic Range Imaging - Aquisition, Display and Image-Based Lighting.* Morgan Kaufmann, second edition, 2010.

[Sil86] B.W. Silverman. *Density estimation for statistics and data analysis*, volume 26. Chapman & Hall/CRC, 1986.

[Sim53] GC Simpson. Ocular haloes and coronas. *British Journal of Ophthalmology*, 37(8):450–486, 1953.

[SRNN05] Bo Sun, Ravi Ramamoorthi, Srinivasa G. Narasimhan, and Shree K. Nayar. A practical analytic single scattering model for real time rendering. pages 1040–1049, 2005.

[SSZG95] G. Spencer, P. Shirley, K. Zimmerman, and D.P. Greenberg. Physically-based glare effects for digital images. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 325–334. ACM, 1995.

[Sø07] Søfartsstyrelsen, editor. *Søvejsregler*. Ivar C. Weilbach & Co A/S, ninth edition, 2007.

[THG99] Jack Tumblin, Jessica K. Hodgins, and Brian K. Guenter. Two methods for display of high contrast images. *ACM Trans. Graph.*, 18(1):56–94, January 1999.

[Tre99] S. Trester. Computer-simulated fresnel diffraction using the fourier transform. *Computing in science & engineering*, 1(5):77–83, 1999.

[vdBHC05] T.J.T.P. van den Berg, M.P.J. Hagenouw, and J.E. Coppens. The ciliary corona: physical model and simulation of the fine needles radiating from point light sources. *Investigative ophthalmology & visual science*, 46(7):2627–2632, 2005.

[Vla08] A. Vlachos. Post processing in the orange box, 2008. Presented at Game Developers Conference. URL: http://www.valvesoftware.com/publications/2008/GDC2008_PostProcessingInTheOrangeBox.pdf.

[War94] G. Ward. A contrast-based scalefactor for luminance display. *Graphics gems IV*, pages 415–421, 1994.

[Wik12a] Wikipedia. Airy disk — Wikipedia, the free encyclopedia, 2012. [Online; accessed 20-September-2012]. URL: http://en.wikipedia.org/w/index.php?title=Airy_disk&oldid=514005683.

[Wik12b] Wikipedia. Cie 1931 color space— Wikipedia, the free encyclopedia, 2012. [Online; accessed 20-September-2012]. URL: http://en.wikipedia.org/w/index.php?title=CIE_1931_color_space&oldid=509071050.

[Wik12c] Wikipedia. Fresnel lens — Wikipedia, the free encyclopedia, 2012. [Online; accessed 29-September-2012]. URL: http://en.wikipedia.org/w/index.php?title=Fresnel_lens&oldid=514135728.

[Wik12d] Wikipedia. Indium gallium nitride — Wikipedia, the free encyclopedia, 2012. [Online; accessed 25-September-2012]. URL: http://en.wikipedia.org/w/index.php?title=Indium_gallium_nitride&oldid=501008647.

[Wik12e] Wikipedia. Inverse-square law — Wikipedia, the free encyclopedia, 2012. [Online; accessed 20-September-2012]. URL: http://en.wikipedia.org/w/index.php?title=Inverse-square_law&oldid=509175131.

[Wik12f] Wikipedia. Light characteristic — Wikipedia, the free encyclopedia, 2012. [Online; accessed 20-September-2012]. URL: http://en.wikipedia.org/w/index.php?title=Light_characteristic&oldid=51006980.

[Wik12g] Wikipedia. Luminosity function — Wikipedia, the free encyclopedia, 2012. [Online; accessed 20-September-2012]. URL: http://en.wikipedia.org/w/index.php?title=Luminosity_function&oldid=499237770.

[Wik12h] Wikipedia. srgb — Wikipedia, the free encyclopedia, 2012. [Online; accessed 20-September-2012]. URL: http://en.wikipedia.org/w/index.php?title=SRGB&oldid=510738475.

[WS67] G. Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley New York, second edition, 1967.

[YIMS08] A. Yoshida, M. Ihrke, R. Mantiuk, and H.P. Seidel. Brightness of the glare illusion. In *Proceedings of the 5th symposium on Applied perception in graphics and visualization*, pages 83–90. ACM, 2008.