Photorealistic rendering of scenes with physically-based sky light

Andreas Agvard
Lund University

Fredrik Lanker
Lund University

Supervisor: Tomas Akenine-Möller
Department of Computer Science
Lund University

2005

Abstract

This master thesis presents an application that outputs images of skies that can be used to produce photorealistic renderings of outdoor scenes. These images are called light probes. The application is divided into two parts, a real time part and a part where an actual rendering of a light probe is done. In the real time part the user chooses a time and position on Earth whereafter an approximation of the sky with these parameters is visualized. These parameters can then be transferred into the second part where a high quality light probe of the sky is created. In the latter part, our approach is to make a physically correct rendering of the sky where both single and multiple scattering of the light due to particles in the atmosphere are taken into account. The scattering is computed by using well-established theories developed by Rayleigh and Mie. The image of the sky is output as a high dynamic range light probe in the RGBE format created by Ward. This light probe can then be used as a light source when rendering three dimensional scenes in any renderer supporting light probes and high dynamic range images.

This thesis also showes examples of two postprocessors that can be applied to the rendered scene, a glare effect and tone mapping. The glare effect is used to simulate the phenomenon that can be seen as a hazy glow in the area closest to a bright light source. Tone mapping is used to present high dynamic range images on a monitor with a limited dynamic gamut.

Sammanfattning

Detta examensarbete presenterar en applikation som beräknar bilder på himlar som kan användas för att rendera fotorealistiska utomhusscener. Dessa bilder kallas “light probes”. Applikationen är uppdelad i två delar, en realtidsdel och en del där den slutgiltiga bilden beräknas. I realtidsdelen väljer användaren tid och plats på Jorden varefter en approximation av himlen med de valda parametrarna visualiseras i realtid. Parametrarna kan därefter föras över till applikationens andra del där en högkvalitativ bild av himlen skapas. I denna andra del försöker vi skapa en fysikaliskt korrekt himmel där vi tar hänsyn till ljus som bryts både en och flera gånger på grund av partiklar i atmosfären. Spridningen på ljuset beräknas med hjälp av de väletablerade teorierna utformade av Rayleigh och Mie. Bilden på himlen presenteras som en “light probe” med ett stort dynamiskt omfång som fås genom att använda RGBE-formatet som är utvecklat av Ward. Denna “light probe” kan sedan användas som en ljuskälla för rendering av tredimensionella scener i något renderarprogram som stödjer “light probes” i RGBE-format.

Detta arbete visar också två exempel på hur de färdiga bilderna kan efterbearbetas. Det ena exemplet är en effekt som simulerar skenet som kan ses i området närmst en stark ljuskälla. Det andra exemplet är ett exempel på hur man kan presentera bilder med stor dynamisk spännvidd på en monitor där denna spännvidd är begränsad.

Contents

1 Introduction **3**
- 1.1 Background . . . 3
- 1.2 Objective . . . 3
- 1.3 Overview . . . 3

2 HDR images and light probes **3**
- 2.1 Light probes . . . 4

3 Our model **4**

4 Common calculations **5**
- 4.1 Positioning sky elements . . . 5
  - 4.1.1 Coordinate conversion . . . 5
  - 4.1.2 Star positions . . . 5
  - 4.1.3 Sun position . . . 5
  - 4.1.4 Moon position . . . 5

5 Real time version **6**
- 5.1 Computing sky light . . . 6
- 5.2 The Sun . . . 6
- 5.3 The zodiacal light and gegenschein . . . 6

6 Light probe rendering **6**
- 6.1 Sunlight . . . 6
- 6.2 Light scattering . . . 7
  - 6.2.1 Rayleigh scattering . . . 7
  - 6.2.2 Mie scattering . . . 8
- 6.3 Ray marching . . . 9
- 6.4 Light probe creation . . . 11

7 Postprocessing **12**
- 7.1 Glare . . . 12
- 7.2 Tone mapping . . . 13

8 Discussion **14**
- 8.1 Encountered problems and solutions to them . . . 14
- 8.2 Possible improvements . . . 15
- 8.3 Conclusions . . . 16

A Coordinate conversion **20**
- A.1 Julian date . . . 20
- A.2 Local mean sidereal time . . . 20
- A.3 Equatorial to horizontal and vice versa . . . 20
- A.4 Equatorial to ecliptic and vice versa . . . 21

B Position computations **21**
- B.1 Sun . . . 21
- B.2 Moon . . . 22

C Sky light formulas **22**

D Color spaces **23**

1 Introduction

1.1 Background

Since the birth of computer graphics, artists and researchers have tried to create photorealistic images. Such images have then been used in, for example, movies to add things that could not have been done in real life, games for making them more realistic, or just for the fun of it.

One way to achieve photorealism is by trying to follow the laws of physics in the process of creating the imagery. This involves, among other things, making physical models of how the light interacts with different materials and with the particles in the air. To be able to do these models there is a demand for a way to simulate the source of the light. The aim for this thesis is to create images of realistic skies that can be used as light sources when rendering three-dimensional scenes. By doing this, photorealistic images of outdoor scenes can be obtained.

1.2 Objective

Our goal is to fulfill the following objectives:

1. Be able to calculate the correct position of the Sun, the Moon and the stars in the sky.
2. Be able to compute and, in real time, visualize an approximative sky given any time and any position on Earth.
3. Be able to render a physically correct sky with both single and multiple scattering and to present this as a light probe that can be used to simulate a sky in a 3d scene.
4. Be able to render a simple scene using our light probe.
5. Be able to postprocess the rendered scene to simulate the human vision, e.g. applying tone mapping and glare effects.
6. If time allows, be able to simulate clouds and use them in our system.

1.3 Overview

This thesis begins with a brief explanation of high dynamic range images (section 2) and light probes (section 2.1). It then goes on by describing our model (section 3) and why we chose to do a stand alone application.

Next, section 4 deals with the common calculations that is done in the various parts of the application. The real time version (section 5) is explained next and after that the light probe rendering (section 6) and all the different parts of it. In section 7 the two used postprocessors are explained. The thesis is concluded with a discussion (section 8) where encountered problems, possible improvements and some final thoughts are presented.

2 HDR images and light probes

In the digital world, colors are usually represented in a 24-bit RGB space, with 8-bits assigned to each of the three primaries. Consequently, most available monitors can

only handle this representation, in the form of sRGB. This means that colors outside the sRGB gamut cannot be displayed, especially values that are either too dark or too bright, since the dynamic range is less than 2 orders of magnitudes, compared to human observers who can distinguish details in a scene spanning over 9 orders of magnitudes. The sRGB gamut is also lacking a large region of the perceivable colors, most notable in the blue-green and the violet regions. Hence, the RGB space is not very good at representing what human observers can see. This is why we need to store our images in a high dynamic range, or HDR, format.

The most simple way to obtain a HDR format is to use a floating-point color space. The disadvantage of using floating-point is that it takes up too much space, therefore many other formats have been invented. One of these formats is the RGBE (Red-Green-Blue-Exponent) format that was created by Ward for his physically-based renderer, Radiance [15]. The RGBE format uses 32-bit per pixel, 8-bits for each of the primaries and another 8-bits for an exponent, which makes it cover about 76 orders of magnitudes. This thesis utilize the RGBE format.

2.1 Light probes

A light probe is a high dynamic range image that contains the incident illumination from a full 360 x 180 degrees, or 4π steradians, to a specific point in space. This image can then be used in image-based lightning techniques to achieve photorealistic renderings. One way to create light probes is to acquire a high dynamic range image by taking a series of images with different exposure settings, of a highly reflective metal ball. Another way is by doing as we do in this report, physically-based rendering. The light probes we create are in longitude/latitude format where the longitude is on the x-axis and is in the range of 0 to 360 degrees. The y-axis represents the latitude and is in the range of -90 to 90 degrees, where 0 degrees is the horizon. In the early stages of our application we utilized another common light probe format called the fisheye lens format. In a fish eye lens image coordinates work like on the unit circle with the longitude as the angle and the latitude as the distance from the center.

3 Our model

Our system is divided into two parts. We have one part which shows the sky in real time using OpenGL (for more information on OpenGL, see www.opengl.org). In this part you can also choose your position, date and time of day. The other part of our system is where we render a high quality light probe of the sky with the parameters chosen in the real time part. We have chosen to do a stand alone application that outputs a light probe of the sky instead of doing a module to be used with a renderer. This has some disadvantages, for example adding effects that has to do with the position of the Sun and atmosphere, like aerial perspective, cannot easily be done. The approach of doing it as a stand alone application that outputs standard light probe has, however, the advantage that it can be used with every renderer that supports light probes and high dynamic range images. Since our goal with this master thesis were not to implement a high-end renderer we chose the latter approach.

4 Common calculations

This section describes the computations that is required in both the real time part and the part that renders light probes.

4.1 Positioning sky elements

An important task for rendering realistic skies is to accuratly position the different sky objects. Since the Earth and other objects are constantly moving, this has to be computed for every frame.

4.1.1 Coordinate conversion

To calculate positions we need to be able to convert between different coordinate systems. In our model we use three systems, the horizontal, the equatorial and the ecliptic system [8]. The horizontal system has the horizontal plane as its base, this makes it time and position dependent and thus it cannot be used in for example a star catalog. We solely use the horizontal system to determine the position that the stars and other objects should have in our sky. In a star catalog you can use the equatorial coordinate system which have the equatorial plane as its reference plane. This coordinate system is independent of both time and the position of the observer.

For the computations we use the eclipitic coordinate system which have the orbital plane of the Earth, the ecliptic, as its reference plane. This coordinate system is also time and position independent and can thus be used for position computations.

Formulas for converting between the coordinate systems can be found in appendix A.

4.1.2 Star positions

We get the position of the stars from The Bright Star Catalogue [5] which contains 9110 entries of stars with a magnitude brighter than 6.5, which corresponds to stars visible to the naked eye. The position is given in equatorial coordinates which we then convert to horizontal coordinates and use these to compute the direction of the stars. We do not take the distance to the stars into account since we do not need it for our calculations. For example the brightness of a star is calculated from the apparent magnitude of the star where the distance is not relevant.

4.1.3 Sun position

The position of the Sun is computed in ecliptic coordinates by the use of formulas in appendix B.1. These formulas do not compensate for the small fluctuation in the Sun's position. Just like for the star calculations we transform the ecliptic coordinates into horizontal coordinates and use these to find the direction of the Sun.

4.1.4 Moon position

We compute the Moon position in a similar way as we do for the Sun. The formulas can be found in appendix B.2. A major difference to the Sun formulas is that the Moon needs more corrective terms to compensate for the perturbation caused by the Sun.

5 Real time version

The real time version is implemented using a hemisphere, a sky dome, where the color of the vertices are set to our computed sky color. The stars are blended with the sky dome and the brightness of a star is set according to its visual magnitude, which we get from [5]. The Moon is modeled as a sphere with a texture created from a photograph of the real Moon. The Moon is illuminated by a light positioned at the center of the Sun, creating the different phases.

An example of the approximation in real time version can be seen in figure 16, where an approximation of the sky at longitude 2 degrees east and latitude 38 degrees north at 1 pm on January 1, 2005 is shown.

5.1 Computing sky light

The sky light consist of many components, most notable is, of course, the Sun. We use Preetham et al.'s [10] formulas for computing the sky light. The sky color in their model depends on two variables, turbidity and the angle between the Sun and zenith. Turbidity is the ratio of the vertical optical thickness of the haze particles and molecules to the vertical optical thickness of the atmosphere with only molecules. Common values for turbidity are in the range 1 to 64, where 1 is pure air and 64 light fog.

By computing the sky spectral radiance for a number of different Sun positions and turbidities and then fit these to a parametric function Preetham et al. get formulas for the spectral radiance. These formulas for computing sky light can be found in appendix C.

5.2 The Sun

The Sun in the real time version is created by multiplying an exponential function, f_M, that tries to mimic the brightness of the Sun, to the sky light previously calculated. The following function is used:

$$f_M = 1 + e^{c(\cos\gamma - 1)} \tag{1}$$

where c is a constant specifying the size of the Sun, higher values gives a smaller Sun. γ is the angle to the Sun.

5.3 The zodiacal light and gegenschein

The zodiacal light and gegenschein are two faint light phenomena caused by interplanetary dust near the plane of the ecliptic reflecting the sunlight. This phenomena can be seen above the rising or setting Sun (zodiacal light) or opposite the Sun (gegenschein) [8]. To simulate zodiacal light and gegenschein in our model we use a table of measured values [12]. For each vertex in the sky dome we convert its coordinate from horizon to ecliptic coordinates and use these to make a bilinear look up in the table.

6 Light probe rendering

6.1 Sunlight

The primary source for light in a sky is the Sun, therefore the sunlight is of great importance when rendering a sky. The sunlight consist of many different wavelengths,

all with different intensity. Since we aim to produce color pictures we must somehow estimate the intensity for each of the wavelength we use. One way to approximate the intensity curve from the Sun is by considering it as a black body radiator and use Planck's Radiation Formula for calculating the intensities. This gives a fairly accurate approximation and can easily be computed. However, an even faster and more precise way to compute the intensities is to use a look up table of measured values. We use a table where the intensity is measured for every nanometer from 300 to 830 nm [17].

A comparison of the two different approaches can be seen in figure 1.

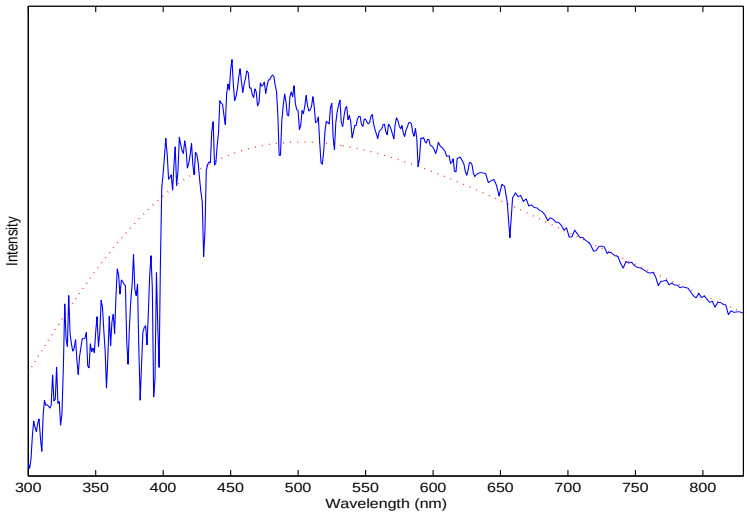

Figure 1: Radiation from a black body (dotted curve) and measured values from the Sun.

6.2 Light scattering

Although it seems like the air is totally empty, it is crowded with small, small particles causing the light to scatter. These particles makes up the atmosphere and without them the sky would be pitch black even in the middle of the day and the Sun would not appear larger than the Moon. There are two common models for describing this scattering, one for large particles, Mie scattering, and another for small particles, Rayleigh scattering. The scattering is described with phase functions which are probability functions that determines the probability for a ray to scatter in a certain direction. Both types of scattering have also a scattering coefficient, σ_r for Rayleigh and σ_m for Mie, which specifies the probability for the light to scatter, and an albedo, Λ which specifies the probability for the light to be absorbed.

6.2.1 Rayleigh scattering

Lord Rayleigh III gave in 1871, as part of his scattering theory, the first correct explanation of why the sky is blue. This is due to the scattering of light in particles smaller than about one tenth of the wavelength of the light. Rayleigh discovered that the wavelengths in the blue end of the visible spectrum, short wavelengths, are scattered and absorbed much more than the ones in the red end, long wavelengths. This explains

why the day sky is blue while sunsets have more of a yellow to red color. During the day short wavelengths scatter more which gives the sky it's blue color, but in the morning and the evening light has to pass through a much thicker layer of atmosphere which absorbs much of the blue wavelengths.

The Rayleigh scattering is described by the phase function in equation (2). See figure 2 for a plot of the function. Rayleigh's phase function assumes that the air molecules are spherical objects, which they are not. In equation (3) the molecular anisotropy has been taken into account [6].

$$\beta(\alpha) = \frac{3}{4}(1 + \cos^2\alpha) \tag{2}$$

$$\beta'(\alpha) = 0.7629(1 + 0.9324\cos^2\alpha) \tag{3}$$

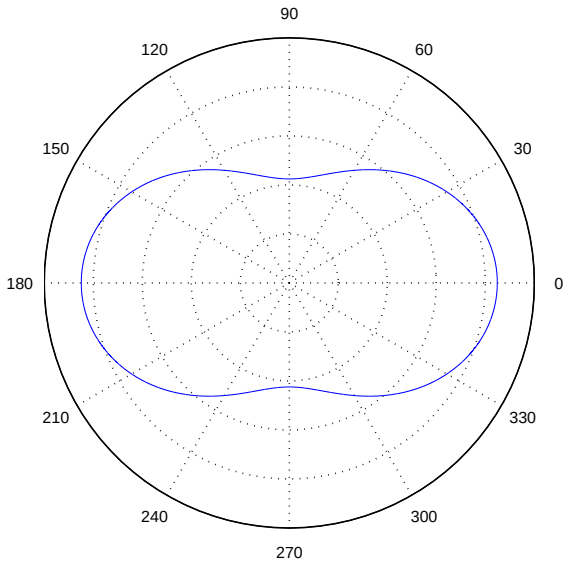

Figure 2: The phase function for Rayleigh scattering.

The scattering coefficient for Rayleigh is:

$$\sigma_r = \frac{8\pi^3(n^2 - 1)^2}{3N\lambda^4} \tag{4}$$

where N is the molecular number density of air, n its refractive index and λ the wavelength of the light. As can be seen, σ_r is proportional to $1/\lambda^4$, this explains why blue light is scattered more than red.

6.2.2 Mie scattering

When light scatters in particles with a size larger than a wavelength, Mie scattering predominate. This sort of scattering is much less wavelength dependant than Rayleigh

scattering, making the medium appear white, which is evident in for example clouds. Mie scattering also affects the appearance of the Sun. While in reality the size of the Sun when watched from the Earth is almost the same as the size of the Moon, Mie scattering scatters the sunlight spreading its rays and making it appear larger. As can be seen in the phase function in figure 3, the forward scattering dominate. This is intensified when the particles becomes larger.

The phase function for Mie scattering is often approximated by combining Henyey-Greenstein functions with different variables. The Henyey- Greenstein function is:

$$P_{HG} = \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)^{3/2}} \tag{5}$$

where g is the asymmetry factor and θ the scattering angle. The asymmetry factor determines the amount of back or forward scattering, depending on the sign. We use a g that is very close to one, which means that forward scattering predominates.

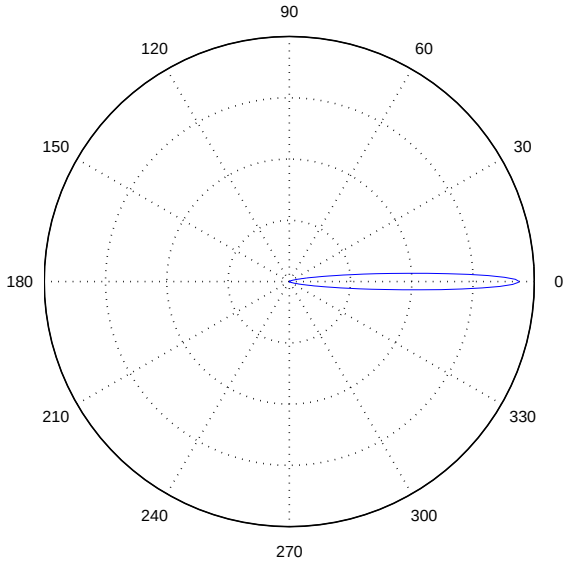

Figure 3: The phase function for Mie scattering.

The scattering coefficient for Mie scattering is much harder to calculate than for Rayleigh, therefore we estimate it with a function proportional to $\lambda^{-0.84}$.

6.3 Ray marching

The actual rendering of the light probes in done with a technique known as ray tracing, or more specific, ray marching. In ray tracing, a ray is emitted from the viewpoint towards a point on the focal plane, which corresponds to a pixel on the image plane. The ray is then traced into the scene, interacting with the objects in the scene and its materials until it hits a light source or the background. The contributed color from that particular ray is then calculated and the pixel on the image plane is set accordingly.

In order to render scenes containing participating media, e.g. fog or as in this case the atmosphere, the volume rendering equation (equation (6)) must be solved:

$$
\begin{aligned}
L(x,\vec{w}) = \quad & \int_0^s e^{-\tau(x,x')}\sigma_a(x')L_e(x')\ dx' + \\
& \int_0^s e^{-\tau(x,x')}\sigma_s(x') \int_{\Omega 4\pi} p(x',\vec{w}',\vec{w})L_i(x',\vec{w}')\ d\vec{w}'dx' + \\
& e^{-\tau(x,x+s\vec{w})}L(x+s\vec{w},\vec{w})
\end{aligned}
\tag{6}
$$

where $\tau(x,x')$ is the optical depth and is given by:

$$
\tau(x,x') = \int_x^{x'} \sigma_t(t)\ dt. \tag{7}
$$

For an explanation of the other symbols used in equation (6), see table 1.

Equation (6) can normally only be solved for a scene by using numerical integration. Ray marching is a method for doing numerical integration by taking small steps through the medium and making simplifying assumptions within the segment considered. Each of these segments is assumed to be homogeneous and the incoming light is assumed to be constant. In each of the segments the direct illumination (from the Sun) is computed and attenuated according to the distance the light have traveled through the medium. This is determined by the extinction coefficient, σ_t. The extinction coefficient at height h is:

$$
\sigma_t = \sigma_t^0 e^{-\frac{h}{T}} \tag{8}
$$

where σ_t^0 is the value of the coefficient at the Earth's surface and T a scale height which is 8.4 km. Instead of using only the height of the sample point in the calculation of the attenuation of the extinction coefficient, Simpson's rule is used to find an average attenuation for the whole segment.

In our model we use the following numerical approximation of equation (6):

$$
\begin{aligned}
L_{n+1}(x,\vec{w}) = \quad & L_{sun}(x,\vec{w_{sun}}')p(x,\vec{w_{sun}}',\vec{w})\sigma_s(x)\Delta x + \\
& \sum_l^N L_l(x,\vec{w_l}')p(x,\vec{w_l}',\vec{w})\sigma_s(x)\Delta x + \\
& e^{-\sigma_t\Delta x}L_n(x+\vec{w}\Delta x,\vec{w})
\end{aligned}
\tag{9}
$$

where the symbols are as explained in table 1.

The center of each segment is our sample point. We find this point by adding a small random length, Δx, to our last sample point, x, in the direction of the ray. Δx is computed as:

$$
\Delta x = -\frac{\log \xi}{\sigma_t(x)}, \tag{10}
$$

where $\xi \in]0:1[$ is a uniformly distributed random number.

To compute the contribution of light scattered more than once, multiple scattering, one has to consider light reaching the segment from all directions, not only from the

Sun. This is made numerically by randomizing a number of rays from the segment using Rayleigh distribution. Multiple Mie scattering is not considered since it makes a small contribution and introduces a vast amount of errors. Since the Rayleigh phase function cannot be inverted, a lookup table has been constructed to aid in the random sampling of scattering rays. For each of these sampling rays the procedure has to be repeated which is why multiple scattering is so time consuming.

The difference between images rendered using only single scattering and images using multiple scattering can be seen in figures 14 and 15. The single scattering images lacks some of the lighter blue color that can be seen in the multiple scattering images. The multiple scattering images are obviously more physically correct but they also are much more time consuming to render.

An irradiance caching system has also been implemented inspired by, but not very similay to, Ward's irradiance cache [16]. This cache is used to selectively choose which pixels to render and which to interpolate from previous cached values. More samples are taken in areas where radiance changes rapidly. To determine if a pixel should be rendered, an approximative error between close pixels is computed using the real time rendered image. If this error exceeds a user specified error tolerance the pixel is rendered otherwise it is interpolated. Figures 11, 12 and 13 shows the result of using different amounts of error tolerance.

For more on ray marching, see [7].

x	position
x'	position of incoming light
$\vec{w}$	direction
$\vec{w}'$	direction of incoming radiance
$d\vec{w}$	differential solid angle
L	radiance
$L(x, \vec{w})$	radiance at x in direction $\vec{w}$
$L(x, \vec{w}')$	incident radiance at x from direction $\vec{w}'$
$L_l(x, \vec{w}')$	incident radiance from light source l at x from direction $\vec{w}'$
$L_{sun}(x, \vec{w}')$	incident radiance from the sun at x from direction $\vec{w}'$
L_n	incident radiance from recursive ray marching step
L_e	emitted radiance
L_r	reflected radiance
L_i	incident radiance
σ_a	total absorption coefficient
σ_s	total scattering coefficient
σ_t	total extinction coefficient
$p(x', \vec{w}', \vec{w})$	probability of radiance entering point x' from direction $\vec{w}'$ leaving in direction $\vec{w}$

Table 1: Explanation of used symbols (from [7]).

6.4 Light probe creation

As mentioned before, we will create light probes of the sky in the longitude/latitude format which have longitude on the x-axis and latitude on the y-axis. The longitude will be, as expected, in the range of 0 to 360 degrees. As a person standing on the

ground only sees 90 degrees vertically, assuming that the ground is an infinite plane, we only render 90 degrees instead of 180. This does normally not cause any problems since the vast majority of the scenes that will be using the light probe will only use the upper 90 degrees — you rarely see the sky below the horizon.

For every pixel in the light probe image, we compute the direction of incoming light for the longitude and the latitude of the pixel. Using the above method, the incoming radiance, i.e. the color of the pixel, can be calculated.

An example of a light probe created by our application can be seen in figure 17.

7 Postprocessing

7.1 Glare

Glare is a phenomenon that can be seen around bright light sources, most noticeable in dark surroundings. It manifests itself by the hazy glow in the area closest to the light, the streaks that seems to emanate from the center of the light, and the series of concentric colored rings around the source, also known as the lenticular halo. These effects gives an impression of greater brightness and are all primarily caused by the interaction of light rays with the physiology of the human eye. The glare effect is implemented according to Spencer et al.'s physically-based glare effect [13].

Three different point spread functions (PSF) are used based on the state of the viewer. These states are photopic (day vision), mesopic (mixed night and day vision) and scotopic (night vision). The PSF for the eye is made from three functions, $f_0(\theta)$, $f_1(\theta)$ and $f_2(\theta)$, where θ is the angle in degrees between a point in the filter and the glare source. $f_0(\theta)$ represents the unscattered component, $f_1(\theta)$ dominates for non-zero θ less than one degrees and $f_2(\theta)$ dominates for angles above one degree. The functions are:

$$f_0(\theta) = \quad 2.61 \times 10^6 e^{-(\frac{\theta}{0.02})^2} \tag{11}$$

$$f_1(\theta) = \quad \frac{20.91}{(\theta + 0.02)^3} \tag{12}$$

$$f_2(\theta) = \quad \frac{72.37}{(\theta + 0.02)^2}. \tag{13}$$

The normalized PSF for the photopic state is then computed as:

$$\begin{aligned} P_p(\theta) = \quad & 0.384 f_0(\theta) + \\ & 0.478 f_1(\theta) + \\ & 0.138 f_2(\theta) \end{aligned} \tag{14}$$

where the subscript p is for photopic.

The PSF for mesopic and scotopic is somewhat more complex to calculate, a lenticular halo has to be added. For this, another PSF is constructed:

$$f_3(\theta, \lambda) = 436.9 \frac{568}{\lambda} e^{-(\theta - 3\frac{\lambda}{568})^2} \tag{15}$$

where λ is the wavelength. The normalized PSF for a observer whose pupil is large but

whose cones are still active, a mesopic observer, is:

$$\begin{aligned} P_m(\theta,\lambda) = \quad & 0.368 f_0(\theta) + \\ & 0.478 f_1(\theta) + \\ & 0.138 f_2(\theta) + \\ & 0.016 f_3(\theta). \end{aligned} \tag{16}$$

Finally, the PSF for a scotopic observer:

$$\begin{aligned} P_s(\theta,\lambda) = \quad & 0.282 f_0(\theta) + \\ & 0.478 f_1(\theta) + \\ & 0.207 f_2(\theta) + \\ & 0.033 f_3(\theta). \end{aligned} \tag{17}$$

The filters still lacking the streaks that seems to emanate from the center of the glare source. To add these, two images are drawn with random lines of random intensity in the range $[0, 1]$. The resulting images are then multiplied by the original point spread functions, one of the images with $f_1(\theta)$ and $f_2(\theta)$ and the other with $f_3(\theta)$, resulting in a complete filter. An example of a filter can be seen in figure 4, showing the filter for the scotopic state.

Figure 4: The glare PSF for scotopic.

The filter is only applied to pixels above a certain threshold set by the user. The filter is centered over each of the pixels above the threshold and then applied to them and the surrounding pixels. The width and height of the filter should ideally be twice the width and height of the input image, but since much time can be saved by making it smaller, the user can decide its size.

7.2 Tone mapping

In order to present high dynamic range images on a monitor with a limited dynamic gamut, colors in the images have to be mapped to colors that the monitor can handle. This is called tone mapping [14]. Tone mappers are normally constructed in such a way that the output is as close as possible to what the human vision would produce. In our images we have dynamic ranges of up to $1 : 1000000$ and very intense brightness produced by the Sun. Therefore we need a tone mapper powerful enough to map

brightness to monitor values without loosing color or contrast. The research on tone mappers have rendered a large amount of different algorithms. Three of them were chosen to be implemented:

Linear tone mapping is the simplest tone mapper. Colors over a specific value in the input image are clamped and then linearly rescaled to fit the color range of the monitor.

Logarithmic tone mapping is a somewhat better method than linear mapping. Colors are still clamped, but logarithmically rescaled. This produces a brighter image, which resembles more to the human visual system.

Exposure tone mapping [4] is the tone mapper most used in our system. It uses the formula $1 - e^{-color \times exposure}$, where $exposure$ is a value that can be changed by the user. The exposure tone mapper is constructed to resemble the process involved in photographing, where the amount of chemicals on the film decrease exponentially when its exposed to light. This makes the brightness inverse exponentially increasing.

We have tested two other more complex tone mappers, specifically Reinhard's [11] and Ashikhmin's [1], but came to the conclusion that exposure tone mapping renders the best results for our needs. Examples of the different tone mappers can be seen in figures 5, 6, 7 and 8.

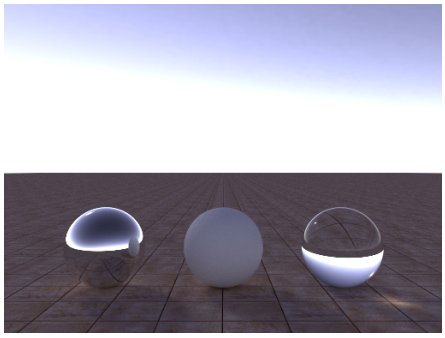

Figure 5: Image tone mapped with a simply logarithmic tone mapper. This produces a quite good result, but bright pixels are clamped leaving large areas white.

8 Discussion

8.1 Encountered problems and solutions to them

We first tried to render the sky images by using photon tracing to simulate multiple scattering. However, this approach was not successful as can be seen in figure 9. Therefore, this idea was abandoned and instead a path marching method were chosen for the rendering of the sky images.

We had some problems on how to make the Sun look somewhat realistic in the real time version. By using only the formulas for the sky light listed in appendix C the Sun were not distinct enough. This problem was solved by finding a function (see equation (1)) that was multiplied with the calculated sky light.

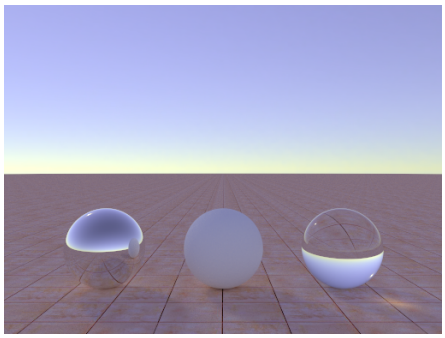

Figure 6: Image tone mapped with Ashikhmin's tonemapper. Ashikhmin's tonemapper can saturate colors as can be seen in this image, it also seems to introduce a gray tone to the image making colors look faded.

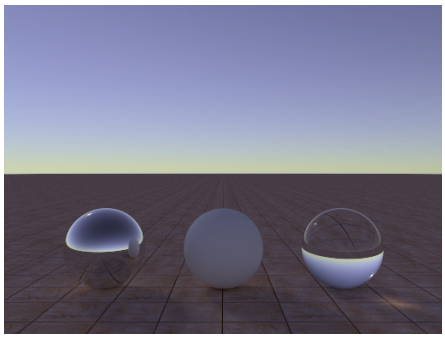

Figure 7: Image tone mapped with Reinhard's tone mapper. Reinhard's tone mapper also introduces some distortion in the colors but at least it does not introduce as much gray tone as Ashikhmin's tone mapper.

8.2 Possible improvements

A way to make the sky look more realistic is to add clouds. This is however a rather difficult task. Some attempts were made to implement clouds (see figure 10). This was done by implementing a non-real time version of "A Simple, Efficient Method for Realistic Animation of Clouds" by Dobashi et al. [3], but due to lack of time we chose not to look into it further.

The night sky is another area that could be improved further. The Moon is currently not rotated correctly. This has been neglected because the Moon is represented by such a small number of pixels making it very hard to distinguish details in the texture of the Moon. To make the night sky detailed to a greater extent, more planets could be added. However, since they would be represented by an even smaller amount of pixels, it would be hard to differentiate them from the stars.

Another possible enhancement would be to add a better tone mapper. By making a tone mapper that was specifically designed to be used with sky images, a better result would probably be possible. However, designing such a tone mapper would be beyond the scope of this thesis.

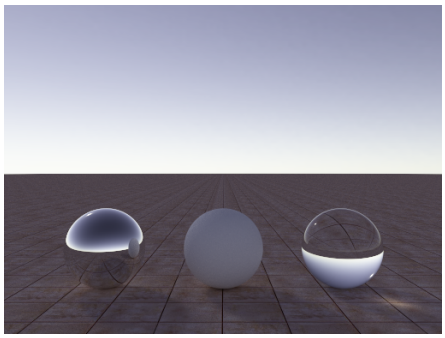

Figure 8: Image tone mapped with exposure tone mapper. Produces good, non-saturated images with much color preserved and still no visible clamped white areas.

The light probes produced do not completely follow the standard of light probes; the y-axis only represent 90 instead of 180 degrees. Since you never see below the horizon this should not be a great drawback. However, it could cause some problems with rendering programs that only supports light probes where the y-axis represents 180 degrees. The problem could be solved by making a copy of a light probe, flip it horizontally and stitch it together with the original.

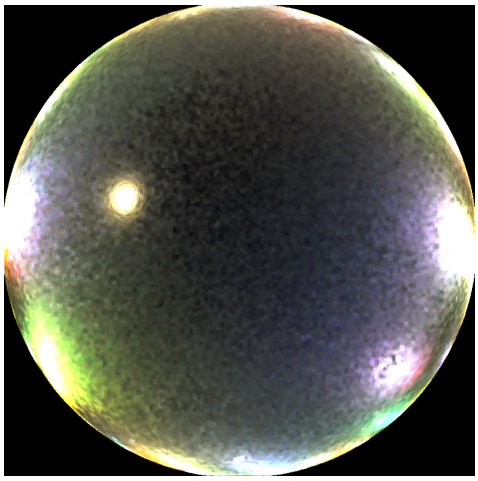

Figure 9: The photon tracing approach, image is in fisheye lens format.

8.3 Conclusions

The objectives of this thesis were:

1. Be able to calculate the correct position of the Sun, the Moon and the stars in the sky.
2. Be able to compute and, in real time, visualize an approximative sky given any time and any position on Earth.

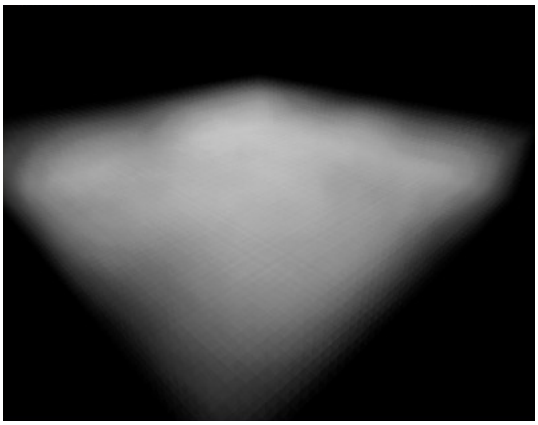

Figure 10: Our attempt with clouds.

3. Be able to render a physically correct sky with both single and multiple scattering and to present this as a light probe that can be used to simulate a sky in a 3d scene.
4. Be able to render a simple scene using our light probe.
5. Be able to postprocess the rendered scene to simulate the human vision, e.g. applying tone mapping and glare effects.
6. If time allows, be able to simulate clouds and use them in our system.

Most of these goals have been fulfilled. The position of the objects in the sky; the Sun, the Moon and the stars, are accurate although the rotation of the Moon is left out. A real time version is implemented where an approximative sky is visualized at any given time and position. Physically correct light probe images of the sky can be computed in both a single and a multiple scattering mode and these images can subsequently be used as light sources in 3d scenes to give an accurate and physically correct outdoor lighting. Both tone mapping and glare effects has been implemented. Finally, attempts to implement clouds has been made although they have not been incorporated into the final application.

A complete scene rendered with one of our light probes can be seen in figure 18.

By tweaking parameters some not so realistic images, that could be useful for non-photorealistic renderings, can be achieved. Two examples of this is in figure 19 and 20 where the first image is a somewhat more “dramatic” sunset and the latter one a totally unrealistic sky.

References

[1] ASHIKHMIN, M. A tone mapping algorithm for high contrast images. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering* (2002), Eurographics Association, pp. 145–156.

[2] brucelindbloom.com.
http://www.brucelindbloom.com accessed January 16, 2005.

[3] DOBASHI, Y., KANEDA, K., YAMASHITA, H., OKITA, T., AND NISHITA, T. A simple, efficient method for realistic animation of clouds. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 19–28.

[4] Exposure.
http://freespace.virgin.net/hugo.elias/graphics/x_posure.htm accessed January 16, 2005.

[5] HOFFLEIT, D., AND WARREN, W. *The Bright Star Catalogue, 5th ed.* Yale University Observatory, 1991.

[6] IRWIN, J. Full-Spectral Rendering of the Earth's Atmosphere using a Physical Model of Rayleigh Scattering. In *Proceedings of the 14th Annual Eurographics UK Conference* (1996).

[7] JENSEN, H. W. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., 2001.

[8] KARTTUNEN, H., KRÖGER, P., OJA, H., POUTANEN, M., AND DONNER, K. J. *Fundamental Astronomy, fourth edition*. Springer-Verlag, 2003.

[9] PEREZ, R., SEALS, R., AND MICHALSKY, J. All-weather model for sky luminance distribution-preliminary configuration and validation. *Solar Energy 50*, 3 (1993), 235–245.

[10] PREETHAM, A. J., SHIRLEY, P., AND SMITS, B. A practical analytic model for daylight. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 91–100.

[11] REINHARD, E., STARK, M., SHIRLEY, P., AND FERWERDA, J. Photographic tone reproduction for digital images. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 267–276.

[12] ROACH, F., AND GORDON, J. *The Light of the Night Sky*. D. Reidel Publ. Co., 1973.

[13] SPENCER, G., SHIRLEY, P., ZIMMERMAN, K., AND GREENBERG, D. P. Physically-based glare effects for digital images. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM Press, pp. 325–334.

[14] TUMBLIN, J., AND RUSHMEIER, H. Tone reproduction for realistic images. *IEEE Comput. Graph. Appl. 13*, 6 (1993), 42–48.

[15] WALTER, B. RGBE File Format. http://www.graphics.cornell.edu/ bjw/rgbe.html accessed May 12, 2005.

[16] WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM Press, pp. 85–92.

[17] Wehrli 1985 AM0 Spectrum. http://rredc.nrel.gov/solar/spectra/am0/wehrli1985.new.html accessed January 16, 2005.

A Coordinate conversion

A.1 Julian date

The Julian date is the time in decimal days since noon on January 1, -4712 (i.e. January 1, 4713 BC). In this appendix we use a variable j_d which is the number of decimal days since noon on January 1, 2000. We compute this by calculate the current Julian date and then subtract the number of days between year -4712 and year 2000. For $Y > 1582$ this is given by:

$$\begin{aligned} j_d = \quad & 1720996.5 - \lfloor Y'/100 \rfloor + \lfloor Y'/400 \rfloor + \lfloor 365.25Y' \rfloor + \lfloor 30.6001(M'+1) \rfloor \\ & + D + (H + (M + S/60)/60)/24 - 2451545 \end{aligned} \tag{18}$$

$$Y' = \begin{cases} Y - 1, & M = 1 \; or \; M = 2 \\ Y, & otherwise \end{cases} \tag{19}$$

$$M' = \begin{cases} M + 12, & M = 1 \; or \; M = 2 \\ M, & otherwise\,. \end{cases} \tag{20}$$

A.2 Local mean sidereal time

To calculate local mean sidereal time we first calculate Greenwich mean sidereal time and add the longitude of the observer. The result in hours is as follows:

$$GMST = 18.69737375 + 24.06570982 j_d \tag{21}$$

$$LMST = GMST + \lambda_h \tag{22}$$

where λ_h is compensation for the time zone.

A.3 Equatorial to horizontal and vice versa

To convert between the equatorial right ascension (α) and declination (δ) and the horizontal altitude (a) and azimuth (A) we also have to compute the hour angle (h) by using the local mean sidereal time ($LMST$, see A.2).

$$h = LMST - \alpha. \tag{23}$$

The horizontal coordinates are then computed by using the following equations:

$$\sin A \cos a = \sin h \cos \delta \tag{24}$$

$$\cos A \cos a = \cos h \cos \delta \sin \phi - \sin \delta \cos \phi \tag{25}$$

$$\sin a = \cos h \cos \delta \cos \phi + \sin \delta \sin \phi, \tag{26}$$

where ϕ is the altitude of the celestial pole, or the latitude of the observer.

To convert from the horizontal system to the equatorial system these equations should be satisfied:

$$\sin h \cos \delta = \sin A \cos a \tag{27}$$

$$\cos h \cos \delta = \cos A \cos a \sin \phi + \sin a \cos \phi \tag{28}$$

$$\sin \delta = -\cos A \cos a \cos \phi + \sin a \sin \phi\,. \tag{29}$$

A.4 Equatorial to ecliptic and vice versa

The conversion between the equatorial right ascension (α) and declination (δ) and the ecliptic longitude (λ) and latitude (β) is achieved by satisfying the following equations:

$$\sin\lambda\cos\beta = \sin\delta\sin\epsilon + \cos\delta\cos\epsilon\sin\alpha \tag{30}$$
$$\cos\lambda\cos\beta = \cos\delta\cos\alpha \tag{31}$$
$$\sin\beta = \sin\delta\cos\epsilon - \cos\delta\sin\epsilon\sin\alpha \tag{32}$$

$$\sin\alpha\cos\delta = -\sin\beta\sin\epsilon + \cos\beta\cos\epsilon\sin\lambda \tag{33}$$
$$\cos\alpha\cos\delta = \cos\delta\cos\beta \tag{34}$$
$$\sin\delta = \sin\beta\cos\epsilon + \cos\beta\sin\epsilon\sin\lambda\,. \tag{35}$$

The angle ϵ in these equations is the obliquity of the ecliptic, or in other words, the angle between the equatorial and ecliptic planes.

B Position computations

B.1 Sun

The position of the Sun is calculated in ecliptic coordinates (λ, β). To know where to put the Sun in our model, we also compute the distance in kilometers to the Sun, r.

$$\begin{aligned}\lambda = \;& 4.895048 + 0.0172027913 j_d + (0.033417 - 2.299794661\cdot 10^{-9} j_d)\sin(M) \\ & + 0.000351\sin(2M) \\ r = \;& 149600000(1.000140 - (0.016708 - 1.149897331\cdot 10^{-9} j_d)\cos(M) \\ & - 0.000141\cos(2M))\end{aligned} \tag{36}$$

where $M = 6.24 + 0.0172019713 j_d$. Since the Sun is in the ecliptic plane we get $\beta = 0$.

B.2 Moon

To compute the ecliptic coordinates (λ, β) of the Moon and the distance Moon to Earth we use the following formulas:

$$\begin{aligned}
l' = \;& 3.8104 + 0.2299715017 j_d \\
m = \;& 6.2300 + 0.0172019685 j_d \\
f = \;& 1.6280 + 0.2308957235 j_d \\
m' = \;& 2.3554 + 0.2280271348 j_d \\
d = \;& 5.1985 + 0.2127687118 j_d \\
\lambda = \;& l' + 0.1098 \sin(m') + 0.0222 \sin(2d - m') + 0.0115 \sin(2d) \\
& + 0.0037 \sin(2m') - 0.0032 \sin(m) - 0.0020 \sin(2f) \\
& + 0.0010 \sin(2d - 2m') + 0.0010 \sin(2d - m - m') \\
& + 0.0009 \sin(2d + m') + 0.0008 \sin(2d - m) + 0.0007 \sin(m' - m) \\
& - 0.0006 \sin(d) - 0.0005 \sin(m + m') \\
\beta = \;& 0.0895 \sin(f) + 0.0049 \sin(m' + f) + 0.0048 \sin(m' - f) \\
& + 0.0030 \sin(2d - f) + 0.0010 \sin(2d + f - m') \\
& + 0.0008 \sin(2d - f - m') + 0.0006 \sin(2d + f) \\
\pi' = \;& 0.016593 + 0.000904 \cos(m') + 0.000166 \cos(2d - m') \\
& + 0.000137 \cos(2d) + 0.000049 \cos(2m') + 0.000015 \cos(2d + m') \\
& + 0.000009 \cos(2d - m)
\end{aligned} \tag{37}$$

The distance is $1/\pi'$ and the unit is Earth radii. By computing the angle between the Moon-Earth vector and the Moon-Sun vector, the current phase can be determined.

C Sky light formulas

In Perez et al.'s [9] model, the luminance distribution is given by:

$$\mathcal{F}(\theta, \gamma) = (1 + Ae^{B/\cos\theta})(1 + Ce^{D\gamma} + E\cos^2\gamma) \tag{38}$$

where A, B, C, D and E are parameters that each has a specific physical effect. γ is the angle between a point on the sky dome and the Sun and θ is the angle between zenith and a point on the sky dome.

The luminance, Y and the chromaticity, x and y, for an arbitrary point on the sky dome is computed as:

$$Y = Y_z \frac{\mathcal{F}(\theta, \gamma)}{\mathcal{F}(0, \theta_s)} \tag{39}$$

$$x = x_z \frac{\mathcal{F}(\theta, \gamma)}{\mathcal{F}(0, \theta_s)} \tag{40}$$

$$y = y_z \frac{\mathcal{F}(\theta, \gamma)}{\mathcal{F}(0, \theta_s)} \tag{41}$$

where θ_s is the angle between the Sun and zenith. Y_z, x_z and y_z is the zenith values

given by:

$$Y_z = (4.0453T - 4.9710)\tan\chi - 0.2155T + 2.4192 \tag{42}$$

$$\chi = (\frac{4}{9} - \frac{T}{120})(\pi - 2\theta_s) \tag{43}$$

$$x_z = \begin{bmatrix} T^2 & T & 1 \end{bmatrix} \begin{bmatrix} 0.0017 & -0.0037 & 0.0021 & 0.000 \\ -0.0290 & 0.0638 & -0.0320 & 0.0039 \\ 0.1169 & -0.2120 & 0.0605 & 0.2589 \end{bmatrix} \begin{bmatrix} \theta_s^3 \\ \theta_s^2 \\ \theta_s \\ 1 \end{bmatrix} \tag{44}$$

$$y_z = \begin{bmatrix} T^2 & T & 1 \end{bmatrix} \begin{bmatrix} 0.0028 & -0.0061 & 0.0032 & 0.000 \\ -0.0421 & 0.0897 & -0.0415 & 0.0052 \\ 0.1535 & -0.2676 & 0.0667 & 0.2669 \end{bmatrix} \begin{bmatrix} \theta_s^3 \\ \theta_s^2 \\ \theta_s \\ 1 \end{bmatrix} \tag{45}$$

where T is the turbidity.

D Color spaces

In our system we use different color spaces for different tasks. The spaces we use is RGB, xyY and CIEXYZ. Since most computer monitors uses RGB (or more specific sRGB) we need to be able to convert between these color spaces. The following formulas are used for conversion [2]:

sRGB → XYZ

$$\begin{bmatrix} X & Y & Z \end{bmatrix} = \begin{bmatrix} r & g & b \end{bmatrix} \begin{bmatrix} M \end{bmatrix} \tag{46}$$

$$r = \begin{cases} R/12.92, & R \leq 0.04045 \\ ((R + 0.055)/1.055)^{2.4}, & R > 0.04045 \end{cases} \tag{47}$$

$$g = \begin{cases} G/12.92, & G \leq 0.04045 \\ ((G + 0.055)/1.055)^{2.4}, & G > 0.04045 \end{cases} \tag{48}$$

$$b = \begin{cases} B/12.92, & B \leq 0.04045 \\ ((B + 0.055)/1.055)^{2.4}, & B > 0.04045 \end{cases} \tag{49}$$

$$M = \begin{bmatrix} 0.412424 & 0.212656 & 0.0193324 \\ 0.357579 & 0.715158 & 0.119193 \\ 0.180464 & 0.0721856 & 0.950444 \end{bmatrix} \tag{50}$$

where X, Y, Z, R, G and B are in the range $[0, 1]$.

XYZ → sRGB

$$\begin{bmatrix} r & g & b \end{bmatrix} = \begin{bmatrix} X & Y & Z \end{bmatrix} \begin{bmatrix} M \end{bmatrix} \tag{51}$$

$$M = \begin{bmatrix} 3.24071 & -0.969258 & 0.0556352 \\ -1.53726 & 1.87599 & -0.203996 \\ -0.498571 & 0.0415557 & 1.05707 \end{bmatrix} \tag{52}$$

$$R = \begin{cases} 12.92\,r, & r \leq 0.0031308 \\ 1.055\,r^{1/2.4} - 0.055, & r > 0.0031308 \end{cases} \tag{53}$$

$$G = \begin{cases} 12.92\,g, & g \leq 0.0031308 \\ 1.055\,g^{1/2.4} - 0.055, & g > 0.0031308 \end{cases} \tag{54}$$

$$B = \begin{cases} 12.92\,b, & b \leq 0.0031308 \\ 1.055\,b^{1/2.4} - 0.055, & b > 0.0031308 \end{cases} \tag{55}$$

where X, Y, Z, R, G and B are in the range $[0, 1]$.

XYZ → xyY

$$W = X + Y + Z \tag{56}$$
$$x_{xyY} = X/W \tag{57}$$
$$y_{xyY} = Y/W \tag{58}$$
$$Y_{xyY} = Y \tag{59}$$

xyY → XYZ

$$X_{XYZ} = x\frac{Y}{y} \tag{60}$$
$$Y_{XYZ} = Y \tag{61}$$
$$Z_{XYZ} = (1 - x - y)\frac{Y}{y} \tag{62}$$

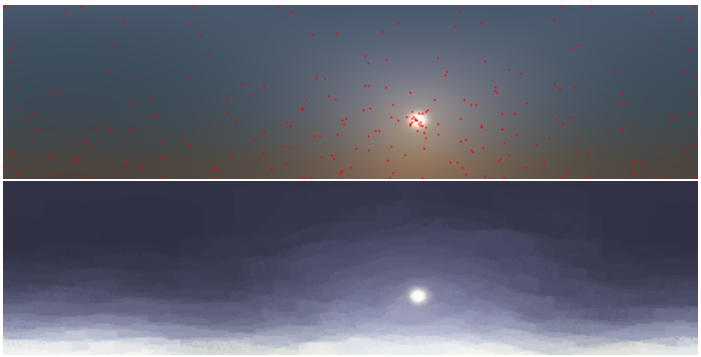

Figure 11: Sample points and rendered image using a 20 % cache error tolerance.

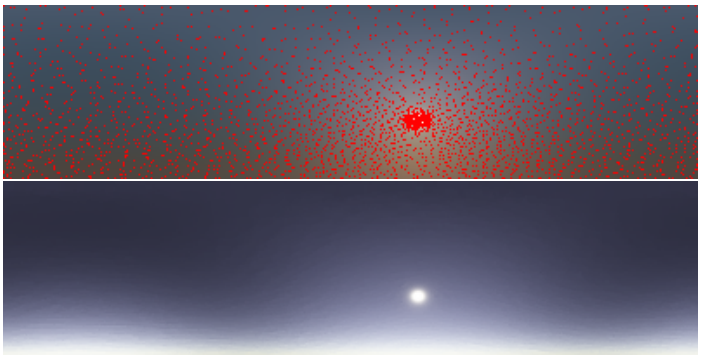

Figure 12: Sample points and rendered image using a 5 % cache error tolerance.

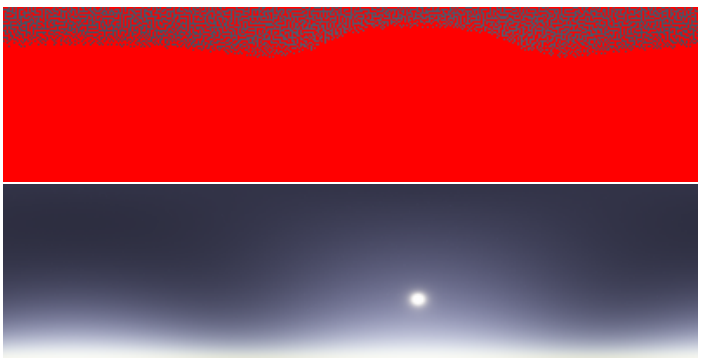

Figure 13: Sample points and rendered image using a 1 % cache error tolerance.

Figure 14: A rendered version with only single scattering.

Figure 15: A rendered version with both single and multiple scattering. As can be seen, the multiple scattering makes the sky more blue.

Figure 16: Approximation from real time version.

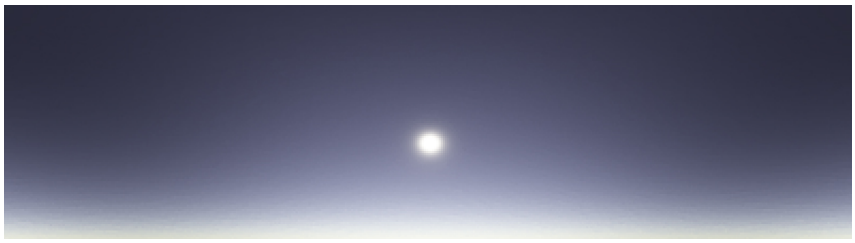

Figure 17: The rendered version of figure 16.

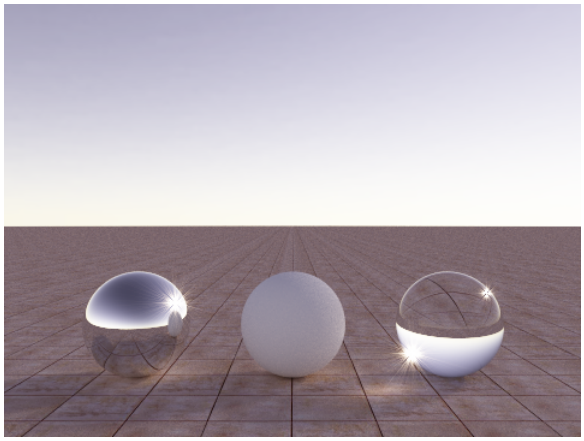

Figure 18: A scene rendered with one of our light probes. Glare effect is also added.

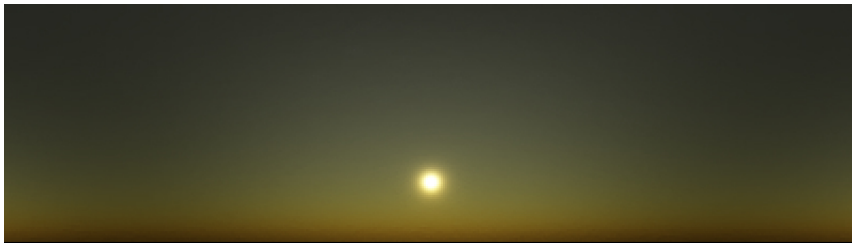

Figure 19: A somewhat more dramatic sunset that is the result of changing some parameters.

Figure 20: Example of a totally unrealistic sky that can be achieved by tweaking some parameters. A glare effect is also added.