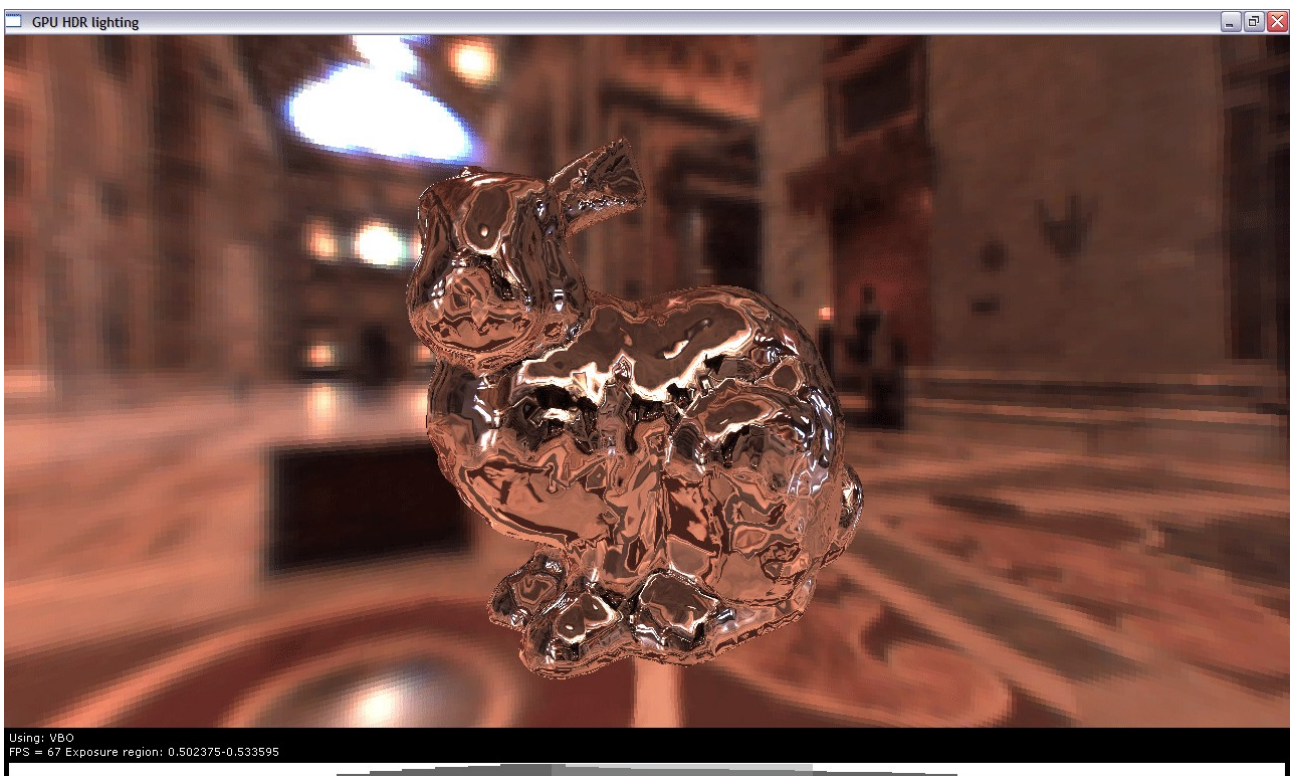


GPU HDR Lighting

Image Based Rendering on the graphics hardware



A project in the course Image Based Rendering.
Linköping University, Sweden.
March 2006.

Authors Hannes Larsson, hanla213@student.liu.se
Ebbe Strandell, ebbst916@student.liu.se
Mikael Uddholm, mikud163@student.liu.se

Advisor Mark Ollila, marol@itn.liu.se

1 Abstract

In this project a High Dynamic Range image system is implemented for real-time rendering using OpenGL Shading Language. The HDR image is used as environment map and the user is allowed to seamlessly change the exposure level of the environment. To further explore the usability of HDR images we add a synthetic mesh in our world and use the the high dynamic range image in the reflectance model that is applied to the object.

This paper deals with implementation problems of using spherical environment maps in combination with reflectance models and our learnings from the implementation of the HDR system in GLSL. Finally we propose a model for HDR screen representation where the user is able to change exposure level and range and change the contrast of the scene.

At last we propose interesting topics for future research, including multi-pass rendering and the Bidirectional Reflectance Distribution Function.

2 Introduction

This chapter aims to explain the background of the project, what we aim to achieve, limitations in the project and expected results.

2.1 Problem description

This report describes a project in the course Image Based Rendering held at Linköping University campus Norrköping during the spring 2006. The project is aimed to to be carried out in 80 working hours per group member, being three persons in the group our goal was to complete everything within 240 working hours.

The assignment was to create media within the paradigm of Image based rendering.

2.2 Project goals

The idea behind the project was to elaborate with HDR images that due to their High Dynamic Range can be used in many areas of computer graphics. To visualize the advantages of HDR images a real-time rendering approach was chosen with an implementation that use the graphics hardware to do much of the lengthy calculations needed to render HDR images. The HDR image is mapped onto a large spherical environment map and through interaction with the exposure component the user is allowed to explore the dynamic range.

Furthermore a reflectance model is implemented and used on objects placed in the environment. The reflectance model mimics glossy materials such as newly polished metal but adds the perfection that can only be found within the virtual world. The purpose of the reflectance model is to evaluate any advantages of using HDR images as environment maps in scenes making where reflectance is a major concern.

3 Approach

This chapter serves to describe the analytic work behind the project as well as how the project was carried out. Because of the small scale of the project most of the thoughts implemented are based on work from others and this chapter aims to explain those ideas, our own assumptions and the final implementation.

3.1 Analysis

This chapter explains the main components of our system with focus on previous work done by others and tools we use to implement our project.

3.1.1 High dynamic range images

The most fundamental component of the project is the High Dynamic Range image system used to create the environment and reflection maps. HDR images use floating values to represent the intensities for each color channel. Float values use a 24-bit representation in comparison to integers used in normal images that are represented using only 8 bit. This higher representation allows a much higher dynamic range, thus making it possible to store intensity levels for extremely bright and dark regions at the same time. This is a well-known problem for the 8-bit representation because 256 values are simply not enough to describe the entire range of intensities.

The development of HDR images has been much dependent of Paul Debevec and this project is much based on his techniques and the HDR images which he has created.

The HDR image is a combination of many photographs taken of the same scene at different exposure times. The variation of the exposure time will ensure that information of regions with different brightness is captured. A highly overexposed image will capture details in regions with very low brightness as well as underexposed images will capture details in very bright regions. The goal of HDR is to combine details in many photos into a single image.

After the creation of the HDR image file a screen representation of the image has to be chosen. This is because monitors today are not designed to represent more than 8 bits or 256 colors. In our approach we wanted to let the user to change the intensity exponent in order to change the brightness of the scene. Thus, the user would be able to smoothly change the corresponding exposure of the camera.

Another approach would have been to compress the entire dynamic range into a representation of 256 values, making details from all camera exposure visible at the same time.

For more information of HDR images please visit: <http://www.debevec.org/Probes/>

3.1.2 OpenGL Shading Language

OpenGL shading language, or GLSL, is used to write *shaders* used to create surfaces of objects. The advantage of using GLSL is that the shaders programs are compiled and uploaded to the graphics hardware. The Graphical Processing Unit, GPU, is designed to handle multiple processes in a parallel paradigm and the hardware is making it extremely efficient on making non-trivial and highly arithmetic calculations.

In our approach GLSL is used to calculate the intensity of every coordinate of the HDR texture for any given exposure. The texture itself contains four 8-bit RGBA-channels and is uploaded at load time. The intensity of each color channel is used to in both the environment map and the reflectance model.

Our implementation of the lighting model involves one rendering pass only. That is, for every texture coordinate the final color depends only on information of that pixel alone, without concerning any surrounding pixel. This approach is simple to implement because it is less hardware dependent and no extra modules is needed. However a multi-pass implementation would enable us to create various desired effects, for example gaussian blur that goes very nicely hand in hand with HDR rendering.

3.2 Implementation

This chapter further discusses the tools we used in this project and how we use them.

3.2.1 Framework

In order to focus on the main task which is real time HDR image rendering we have used open source software and other available classes when building the framework.

- C++
All the main program code is written in C or C++
- OpenGL
The graphics is built using OpenGL and GLSL shading language
- GLEW
The [OpenGL Extension Wrangler](#) library is used to handle extension loading.
- GLFW
The window handling is done by OpenGL FrameWork for portability.
- Other external source used is
bmpIO, used for screenshots
glFont, used for writing text to the window
aGLSL, shader management
Rply, framework for loading ply files

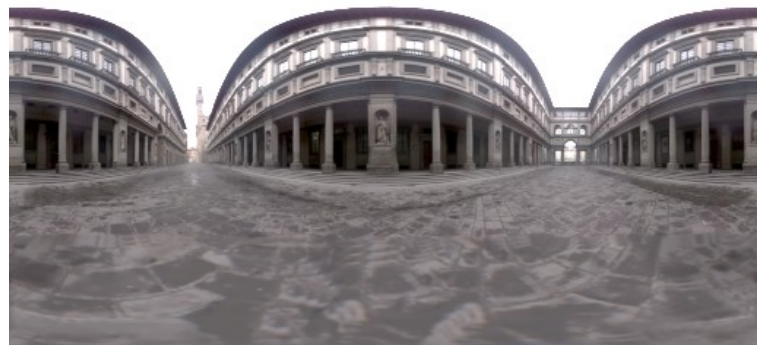
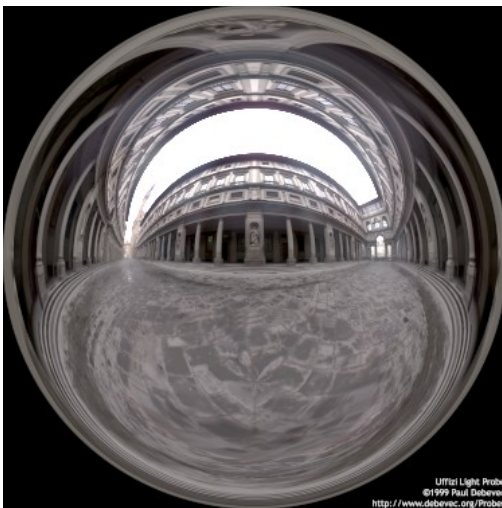
3.2.2 Mesh reader

To import an polygon object of type .ply an pre-created ply-reader was used created by Diego Nehab, Princeton University July 16 2006, called RPLY. It is a toolkit to read, manipulate and write PLY files. IT is an well documented, open-source written toolkit in ANSI C standard. It can be downloaded from <http://www.cs.princeton.edu/~diego/professional/rply>. The PLY-format was created at Stanford University for use with their 3D scanning projects. The RPLY version used in

this project is 1.01. By setting callback functions and procedures the read command is then fired. During the reading of the PLY file, RPLY calls the callback functions to read the different sorts of elements existing in the PLY file.

3.2.3 HDR image preparations

The process of creating and preparing HDR images is much based on previous work and especially Paul Debevec has made considerable work in the area. Our implementation is based on a spherical representation of the image that consists of two HDR compositions with photographs taken of a mirrored ball with 90 degrees disparity. Thus it covers entire 360 degrees in the horizontal plane. Before this HDR image is used in the scene it is converted into a rectangular latitude-longitude map, in this representation Paul Debevec is the source of both the image files used and the software HDRShop that we use to convert the spherical images into latitude-longitude maps.



Above: Latitude-longitude map

Left: Spherical map

The latitude-longitude HDR map consists of 24-bit floating values. Because our graphics card merely can handle 8-bit floating values we convert the 24-bit RGB image into a 8-bit integer RGBE representation where the E-channel contains the exposure component in which every value-step corresponds to the double intensity using the formula $I = \mathbf{RGB} * 2^E$, where I is the intensity of the pixel, RGB corresponds to the color channels and the exponent E is the exposure component.

Despite this is possible to accomplish this conversion directly in HDRShop our implementation involves a small set of MATLAB scripts, this provided us with the exact algorithm to create the RGBE image as well as tools to calculate minimum, maximum and mean values which was a very helpful during implementation. The MATLAB scripts involved steps as

- Reading the HDR image representation into memory. At this stage the color intensities are represented in 24-bit floating values.
- The 24-bit RGB intensity values is translated into 8-bit RGBE using
 $E = \text{ceil}(\log_2(\max(\mathbf{RGB})))$

$$\text{RGB} = (\text{RGB}/2^E) * 256$$

- For each $\text{RGB} > 255$

$$\text{RGB} = \text{RGB} / 2$$

$$E = E + 1$$

- For all RGB

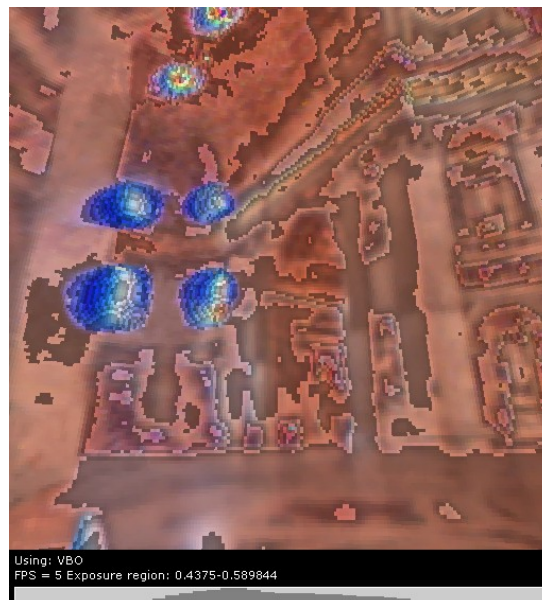
$$E = E + 128$$

Following these steps the result is four 8-bit RGBE integers that represent the entire 24-bit range with a very small distortion.

3.2.4 HDR image rendering system

The HDR image rendering system is based on the inverse of the above system. The color intensity is easily reversed using the simple formula $I = \text{RGB} * 2^E$ but due to the limitation of the monitor we can not render this entire range at the same time. Surprisingly, designing a system to represent this in a descent way ended up being one of the hardest tasks to solve in the project.

The image below shows the HDR image when its rendered using the entire range without any modifications using response curves. The psychedelic effect arise because we never let a pixel “burn out” only because of any less than all three color components. In reality one color component alone, for example red, could cause a burned out effect if its intensity goes extreme. In contrast a pixel in this representation would be mapped to red even though the intensity value by far exceeds the 256 levels the monitor can handle. This calls for a method to limit the range that we are rendering.



Color bleeding effects due to a dynamic intensity range much higher than the monitor can represent

To deal with the problem above we introduce a way for the user to select a range of exposure values that will be mapped to minimum and maximum exposure values. A logarithmic histogram of the

exposure component is created to let the user know in what regions the most exponent components are. On top of the histogram a rectangular is drawn to indicate minimum and maximum values as well as the range of the rangebar.

The final intensity for each pixel is based on the following criteria

- We *do not* want to completely erase information from pixels that fall below the minimum value of the selected range. However, we subtract the minimum value from the exposure component of each pixel. Thus, values that are below minimum will be rendered black or very close to black.
- By varying the maximum value (or the length) of the rangebar we want to adjust the intensity calculation so that exposure values above the the maximum value will be burnt out. This approach results in that short ranges give high contrast, medium ranges results in a “good-looking” representations and a big ranges results in color bleeding defects similar to above.

These two criteria were implemented as follows

exponent = (exposure – minE) * 255.0 * ((8.0/255.0)/(maxE-minE))²;

RGB = RGB * 2^{exponent}

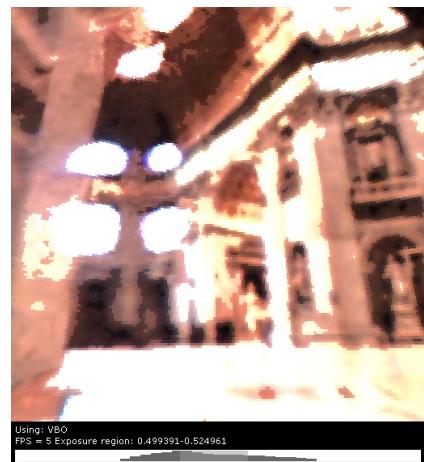
Notice that our implementation is on the form $I = RGB * 2^E$ but we introduce modifiers to let the user seamlessly interact with the scene, to change the range of exposures.



Rendering using a too low exposure level.



Rendering using higher exposure level which gives a much nicer result.



A too short range results in images with very high contrast.

3.2.5 Reflectance model

The reflection is calculated in graphics hardware using the technique presented by Blinn and Newell. The reflectance vector R is derived through the expression $R = 2(N \cdot V)N - V$, where N is the normal and V the vector from the eye to the fragment. The reflectance vector is then mapped to latitude and longitude texture coordinates using the spherical mapping formula.

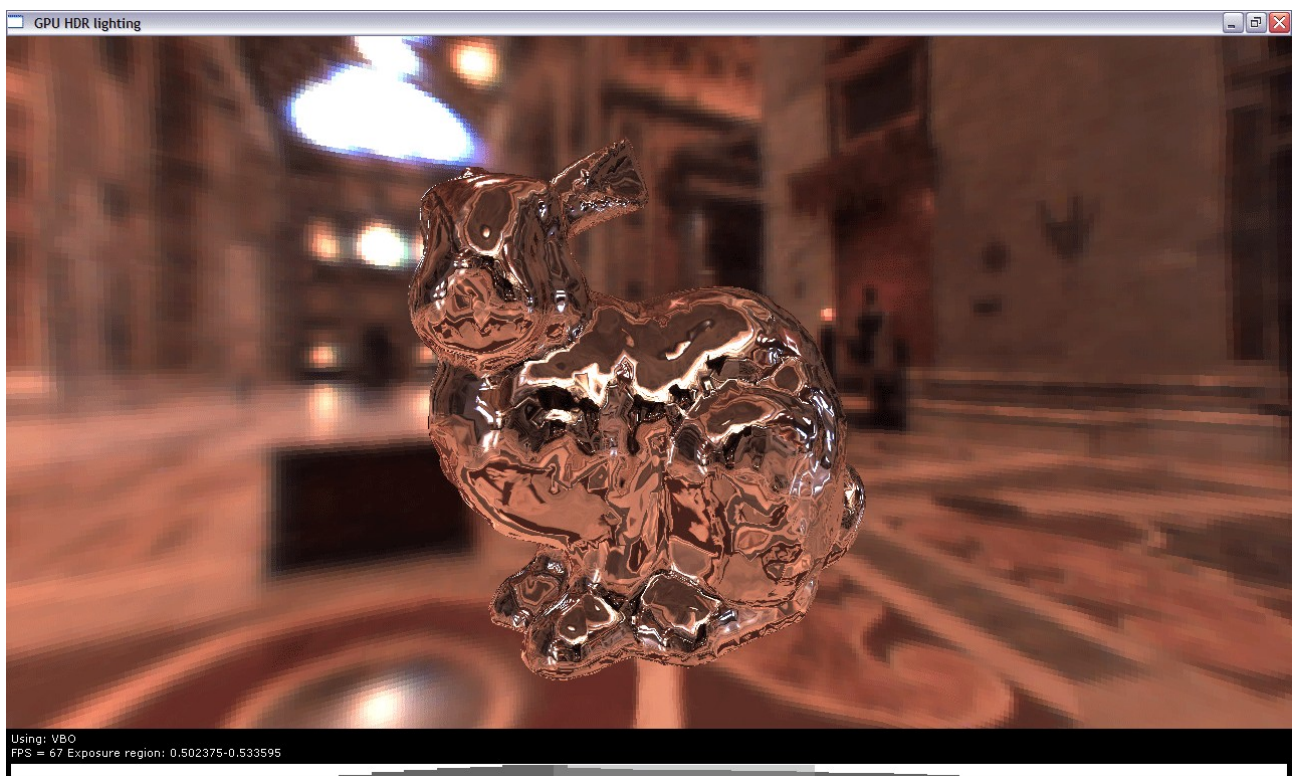
$$u = \frac{1}{2} \times \left(1 + \frac{1}{PI} \times \arctan\left(\frac{y}{x}\right)\right)$$

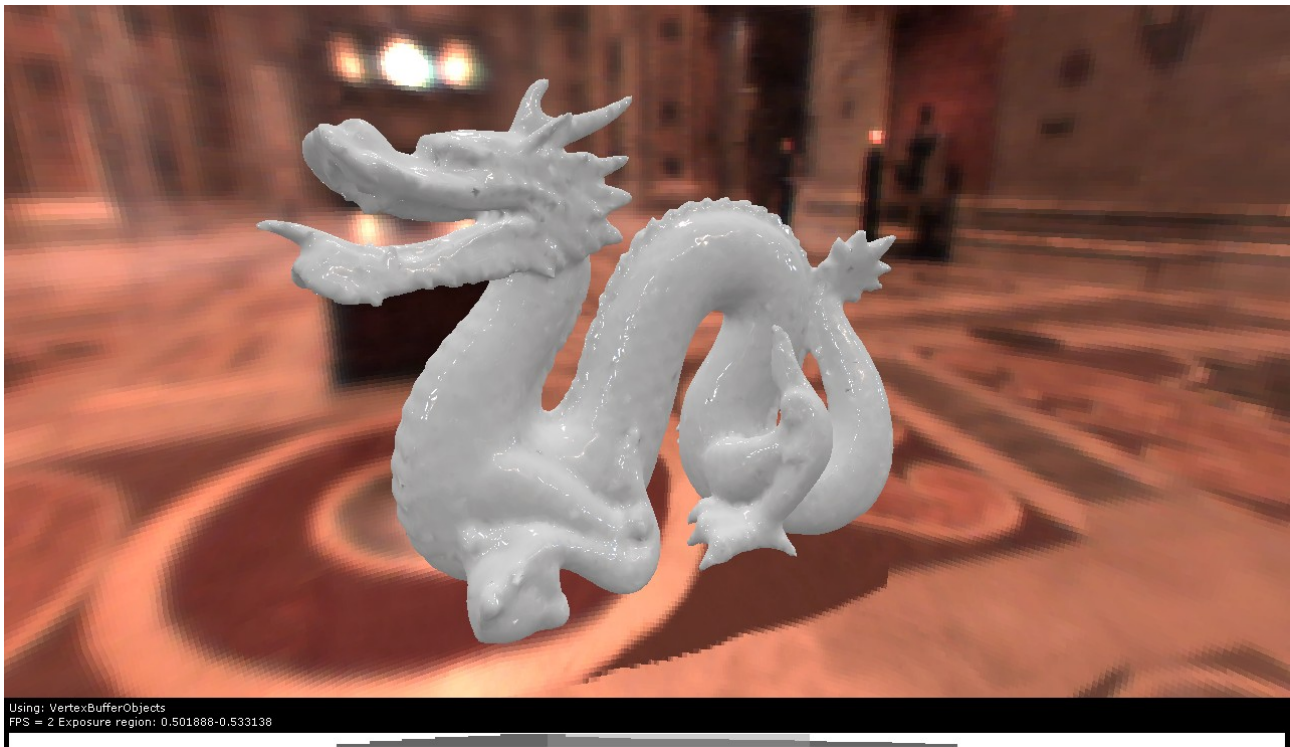
$$v = \frac{(z+2)}{2}$$

4 Results

We are very satisfied with the result of this project as we think that we have shown that HDR images are wanted as in environment maps as well as in reflectance models like the one we have implemented. However, most of the advantages of HDR would require more in-depth work, for example glooming effects and motion blur would do well in an application like this.

We think that these two images well represent our work but we advise the reader to try out our application. It requires a fairly new graphics card and 3D drivers.





4.1.1 Instructions to the program

To move around the mesh object use arrow keys left and right.

To adjust the full exposure range use keys E and R. This will move the exposure region over the histogram.

To adjust the exposure region use keys D and F. This will set the upper limit of the region.

To zoom the mesh use PageUp, zoom in, and PageDown, zoom out. This only effects the loaded mesh in center.

To change shader on mesh use the key S. This will switch between the different loaded shaders that are currently available on the mesh object.

To change the mesh object use the key O. This will show different meshes loaded in to the application.

5 Conclusions and future work

The limitation of the graphics hardware ability to calculate high range numbers makes this task both interesting and challenging. However while exploring the possibilities of performing advanced light calculations in real time one can easily

- BRDF

Implementation of BRDF for better lighting calculation would greatly improve possibility to

realistically include other surface materials. The BRDF function could be implemented in hardware as a function or a lookup table (texture).

- Diffuse lighting

The HDR texture could be used to generate a virtual light stage for diffuse lighting. By rendering the object from different points on the light stage sphere a texture containing the light to triangle visibility could be generated for realistic shading.

- Multi-pass rendering

Multi pass rendering could be used to create blooming effects where the light is intense and softening of the sharp edges.

6 References

6.1.1 Books

Alan Watt, "3D Computer Graphics, Third Edition", pp.247 (2000)

Jan Skansholm, "C++ direkt, Andra Upplagan" (2000)

Gunnar Sparr, "Linjär Algebra" (1994)

Robert A. Adams, "Calculus - A Complete Course, Fourth Edition" (1999)

6.1.2 Reference Manuals

Jhon Kessenich, Dave Baldwin, Randi Rost, "The OpenGL Shading Language" (2004)

Jonas Unger, "High Dynamic Range image format .PFM taking into account the number of channels and byte order" this is the ReadPFM, a MatLab script (2006)

Lighthouse3D tutorials for GLSL found at <http://www.lighthouse3d.com/opengl/glsl>

6.1.3 Papers

Paul E. Debevec, Jitendra Malik, "Recovering High Dynamic Range Radiance Maps from Photographs" University of California at Berkeley, SIGGRAPH '97