Gaze Adaptive Image Viewer

Informatikpraktikum 1

Author: Markus Radits

Adviser: Matthias Bernhard

WS 2010

Technical University Vienna

Institute for Computer Graphics and Algorithms

Abstract

Within this practical work we started to built a Gaze Adaptive Image Viewer which is capable of rendering a tone mapped Image according to extracted image parameters around the point of gaze respectively the area of gaze whose size is adjustable by the user. This work was inspired by the tremendous amount of research that has been made in HDR Imaging the last couple of years and the attention that was given to model the human visual system in Tonemapping. General approach in Tonemapping is to use either local or global Operators to compress a high dynamic range to a low dynamic range image. A new approach to Tonemapping is to use the point of gaze to compute Tonemapping Operators. The area around the point of gaze can be used to compute an Operator that is applied globally to an image. That is to adjust Tonemapping when looking at different areas of an image to get a more naturalistic impression. We used a OpenGL Shader to implement real - time Tonemapping.

Inhaltsverzeichnis

1. Introduction

There exists a lot of literature explaining Tone Mapping in general and different kind of Tone Mapping Operators, so this document only comprises a short overview of what Tone Mapping is and how it is applied to HDR Images. During research within this project we found a concise but interesting paper [**1**] discussing a pretty similar system compared to what we started to build. But at the point of working on this project the appropriate paper details were still not published.

So the overall aim of this practical work is to create or to implement a Gaze Adapative Image Viewer that is capable of rendering HDR Images in real – time according to Tonemapping Operators that are computed by considering the Point of Gaze. In the beginning of this prototype application we rather used mouse movements instead of an Eyetracker to test the computation of the Adaptive Tomemapping Operator.

This document describes the overall design of the application, the approach that has been taken as well as the problems that arose.

2. HDR Imaging and Tone Mapping

An HDR Image is an image that comprises the whole natural luminance that has been captured with a camera. The main problem of HDR Imaging is that a common output device cannot depict the full dynamic range of luminance that has been captured with a camera. A single pixel component, for example RED, stored with 8 Bit can have 255:1 different contrast ratios. In real world we can have a contrast ratio up to 50.000:1. A common computer monitor can have contrast ratios between 500:1 and 1000:1 [**2**]. Although research of HDR displays has made progress we still need to convert the full dynamic image information to low dynamic information in a non – linear way, to preserve details of an image.

For solving this task we can use Tonemapping. Tonemapping in general is to provide a function that is transferring the high dynamic range image information to the dynamic range of an output device[**2**]. So the process of Tonemapping can be done in various ways and different kind of Tonemapper produce different results. A Tonemapping Operator analyzes image regions in global or local space and computes a factor that the original pixel is scaled with. Here is a list of some popular Tone Mapping Operators that are all implemented in the pfstmo library (amongst others) [**3**]:

- Reinhard 05 [**4**].
- Mantiuk 08 [**5**].
- Drago 03 [**6**].

3. Technical Documentation

This part describes the necessary tools to run and to compile the prototype system as well as it explains the overall architecture of the application. One aim within the project was to guarantee platform independence. Only open source tools were used during development.

3.1.Necessary Tools

3.1.1. Minimalist GNU for Windows

As the application was developed with C++ under a Windows 7/64 bit machine, the MinGW compiler was used. Downloads can be found on the web [**7**]. Ensure to include MinGW bin directory to the path.

3.1.2. Qt 4.7

Get the Qt Framework from Nokia Website.

3.1.3. QtCreator

In fact any C++ IDE can be used for development. I found the Qt Creator to be a good, stable and light weight environment [**8**].

3.1.4. Pfs Tools

The Pfs Tools library [**3**] is a set of functions to read, write and manipulate HDR Images. The Application loads a radiance HDR Image into a Pfs frame using the Pfs Tools library. Detailed documentation about the specifications of the Pfs format can be found here [**9**]. To get a quick overview:

The Pfs format is a byte stream including one or more frames. Each frame consists of channels for each color component. Color data are given in XYZ color space. The header of a frame holds information about channels, size and additional tag information. The binary data follows the header and has the structure of a 2d floating point array [**9**].

3.2.Application Design

Currently the application is ported into a separation of a GPU context and a CPU context. The design is shown in Figure 3.1.

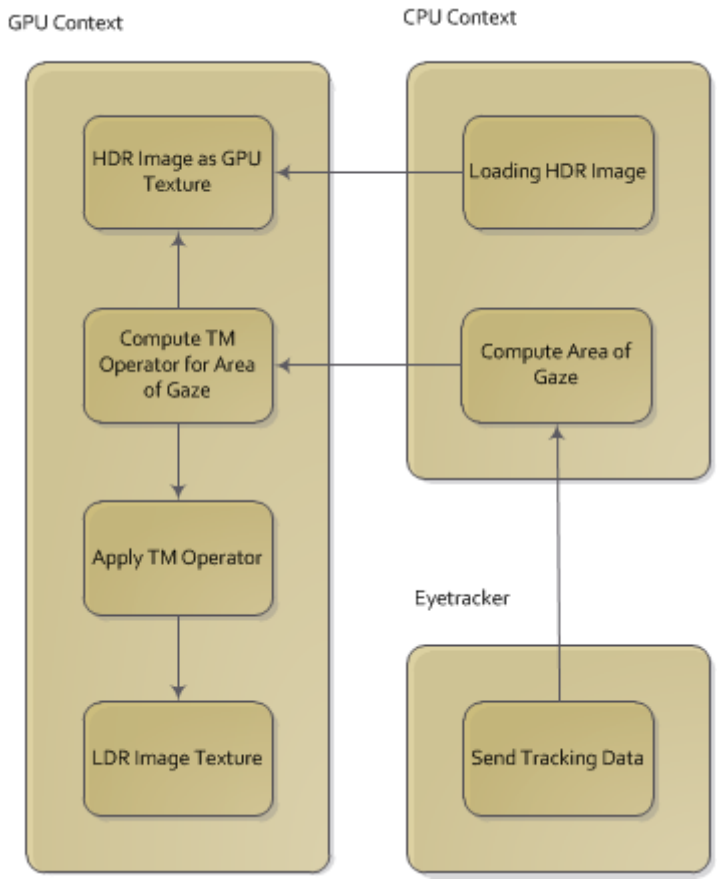

Figure 3.1 An architectural overview of the application

3.3. API Documentation

The API documentation was generated by using Doxygen and is available as HTML and Latex output. The documentation folder can be found within the QT Creator project folder.

4. Conclusion

Before starting to write the demo application it was a huge benefit to analyze how Tonemapping was achieved with the Open Source framework Luminance HDR [**10**]. The Luminance framework uses the pfstmo library to accomplish Tonemapping in offline mode – not suitable for real – time application. The biggest problem was in fact to implement fast and real – time rendering of HDR frames. A first approach in creating a prototype using QImage containers for animation ended in performance problems. Pixel iterations over QImages is very costly and should be avoided in real – time rendering. In a next step we decided to implement a pointer arithmetic to work directly on byte arrays of a Bitmap when rendering the HDR Image. This approach was a huge gain in performance. Especially when working on lower resolutions the animation ran fluently. But working on larger scales, for example on a full 24 Inch screen still some flickering was noticeable. The next step in solving this problem was to dislocate the computationally intensive process of Tone Mapping from the CPU to the GPU. As the application was developed on a machine with ATI Radeon 4500/5100 graphic hardware it was not possible to use CUDA, because this framework is not supported to work with ATI hardware. A first quick research on this topic guided me to use OpenCL [**11**] in such a use – case. After some evaluation it was decided to use OpenGL Shaders instead, because OpenCL is too general purpose for the results we wanted to accomplish. There exists some OpenGL Shader implementations for Tone Mapping but none of them were to integrate easily into our application. So the next step was to use the OpenGL Shader – Wrapper provided by Qt framework. As this wrapper is not documented at best it took a little time to see how things work. Performance problems were completely eliminated after performing Tonemapping on the graphic card by using an OpenGL Shader.

5. Future Work

The point of gaze in the current application is controlled by moving the mouse pointer. In a next step a Gaze Tracker should be connected for testing in real – world and for starting an evaluation process to get out more about potential benefits in such a use – case.

A recommended way of integrating the Eyetracker Position is to call the

EyeSettings->computeEyeZone function with the current position as a parameter.

When the eye zone has been computed the local Luminance is computed and the Shader uses the updated values to compute the Tonemapping.

Currently this process is triggered by mouse move events `GLWidget` class:

```
void GLWidget::mouseMoveEvent(QMouseEvent * e){
    eyeSettings->computeEyeZone(e->pos());
    emit computeLocalLuminance();
    if(eyeSettings->getEyeRectangleCheckbox()->isChecked()){
        emit passEyeZone(&(eyeSettings->eyeZone));
    }
}
```

6. References

1 Susanto R, Farzam F, Corey M, Huang Z, Jamie NSL, Ishtiaq RK, Ong EP, Song P. Eye HDR: gaze-adaptive system for displaying high-dynamic-range images. 2009.

2 Fabien H, Stéphane M. High Dynamic Range Rendering in OpenGL.

3 Pfs Tools. [Internet]. [cited 2011 Mar 11]. Available from: http://pfstools.sourceforge.net/pfstmo.html.

4 Erik R, Kate D. Dynamic Range Reduction inspired by Photoreceptor Physiology. 2005.

5 Rafał M, Scott D, Louis K. Display Adaptive Tone Mapping. 2008.

6 F. D, K. M, T. A, N. C. Adaptive Logarithmic Mapping For Displaying High. 2003.

7 MinGW. [Internet]. [cited 2011 Mar 11]. Available from: http://www.mingw.org/.

8 Qt Creator. [Internet]. [cited 2011 Mar 11]. Available from: http://labs.qt.nokia.com/category/qtcreator/.

9 Pfs Documenation. [Internet]. [cited 2011 Mar 11]. Available from: http://pfstools.sourceforge.net/documentation.html.

10 Luminance HDR. [Internet]. [cited 2011 Mar 11]. Available from: http://qtpfsgui.sourceforge.net/.

11 Wikipedia/OpenCL. [Internet]. [cited 2011 Mar 11]. Available from: http://de.wikipedia.org/wiki/OpenCL.

12 Laurence M. Tone Mapping for High Dynamic Range Images. 2006.

13 NVidia. [Internet]. [cited 2011 Mar 11]. Available from: http://developer.nvidia.com/object/cuda_3_2_downloads.html.

7. List of Figures

8. Appendix

8.1.File Structure of the Projects Folder

File/Folder	Description
AdaptiveGazeViewer	The QT Creator project including all sources to compile the project. API documentation.
Example Code	Some code snippets that helped me during the implementation.
Graphics	Includes the Architectural Overview as a Visio Diagram.
HDR Literatur	A collection of literature related to the field of HDR imaging.
Images	Includes HDR images for testing purpose.
opgengl	A collection of tutorials and documentation related to opengl and shaders.
Tools	A collection of tools and applications that helped me during the implementation.
Operating Manual.pdf	The Operating Manual for the Adaptive Gaze Viewer
Report_x.x.pdf	This report describing the whole project