

CS 543 Course Project: Realtime High Dynamic Range Rendering

B. Burak Arslan

February 25, 2007

Abstract

The purpose of High Dynamic Range Imaging is to accurately represent the wide range of intensity levels found in real scenes ranging from direct sunlight to the deepest shadows. Analogously, High Dynamic Range Rendering (HDRR) is about using unclamped floating-point numbers for each color value of a pixel of the render target, thus representing correctly the high dynamic range in artificial scenes.

1 Motivation

High Dynamic Range(HDR) Images are usual bitmaps, with the sole difference being the structure and significance of pixel values. In HDR Images, each color value in a pixel is a floating-point number, whereas in conventional bitmaps, it is an 8-bit integer.

This creates the possibility of representing the light intensity in a scene correctly which gives way to various applications in many domains.

In computer games and most of the other real-time rendering applications, HDR rendering is used as a way to implement the automatic exposure adjustment effect.

2 Technical Background and Previous Work

High Dynamic Range Imaging is an often misunderstood topic that is made of more or less independent problems.

Roughly, it is about using HDR images as the only source of light in an artificial scene. To achieve that goal, the problems that must be solved are the following:

1. Obtaining HDR images: HDR images¹ contain radiance measures instead of "color values" as pixel values. HDR images are obtained by applying the techniques described in [1] to a series of photographs of the same scene taken with different exposure values.

Quoting from [2]:

Accurately recording light in a scene is difficult because of the high dynamic range that scenes typically exhibit; this wide range of brightness is the result of light sources being relatively concentrated, which causes the intensity of a source to be often two to six orders of magnitude larger than the intensity of the non-emissive parts of an environment. However, it is necessary to accurately record both the large areas of indirect light from the environment and the concentrated areas of direct light from the sources since both are significant parts of the illumination solution.

The HDR image that could be used to illuminate an artificial scene is obtained from a "Light Probe", which is an HDR photograph of a purely reflective sphere placed in the local scene. The image on the sphere is then mapped to a 3d model of the environment. Generally, approximating the environment to a cube gives acceptable results. Figure 7 of [2] is a good summary of the method.

The light probe transformed to a cube map can finally be used to light the artificial scene.

2. Using Image Based Lighting: This is about using a 2d image as the only light source used to solve the illumination of an artificial scene. According to Debevec [3], it's a derivative of the reflection-mapping technique. Technically,

¹called *radiance maps* by the original author

LDR images could as well be used as a light source, but it would not result in an accurate solution of illumination of a scene, as LDR images can't represent correctly the contrast of light intensity in a real-world scene.

3. As displaying HDR images on consumer hardware without further processing is not possible, the new HDR image is obtained as the result of the above HDR Rendering process must be refined further. HDR screens that are capable of displaying raw HDR images exist, but they are not accessible to the end-user because of the price, the low-availability of the product, and also because of the fact that the technology is still not yet mature enough to be sold to the mass-market.

Thus, the HDR image should be rendered to a LDR image in order to be displayed on consumer hardware. This process is called *tone mapping*. The tone mapping step is the last step in a HDR pipeline, and the only difference (besides the floating-point off-screen framebuffer) from a conventional LDR rendering pipeline.

Various linear and non-linear tone mapping operators exist, for various purposes. A list is available [here](#)².

The second paper from Debevec published the following year[2] is about seamlessly integrating artificial and real objects. Using HDR images for illuminating artificial objects is not even a subset of that paper, as that work is done by Radiance[4], which is a physically correct lighting simulation and rendering system, which means, to my understanding, it is a complex raytracing package.

A project that explores the techniques used to obtain HDR images and renderings can be found [here](#)³. Also another one that implements basic HDR Rendering with puffers can be found [here](#)⁴.

2.1 Representing a HDR image

In system memory, HDR images are represented as arbitrary precision floating-point RGB values, whereas different file formats exist for storing HDR images on non-volatile media. One of the popular file formats is the RGBE format. Instead of storing three floating point values, RGBE format stores four 8-bit integers which are the Red, Green and Blue values of that pixel, and a shared exponent (hence the E). The precision of RGBE format is not adequate for representing full high dynamic range, but it is a compact and easy-to-integrate format (with free source code available).

3 Implementation Details

HDR Rendering is a relatively new area of application for OpenGL. I sought to use the newest technologies and methods of this API.

²www.mpii.mpg.de/resources/tmo

³homepage.mac.com/eric.lee/cs348b

⁴www.ultimategameprogramming.com/zips/G1_HDR.ZIP



Figure 1: A screenshot with high brightness thus a low exposure value

The demonstration code requires an OpenGL 2.0 compliant NVIDIA graphics card. The program is tested on a GeForce 7800GT and a Quadro FX 3400.

HDR Rendering requires the render target to be a floating-point buffer. As conventional OpenGL screen buffers use integers to represent pixel values, an off-screen render target must be used. I used the floating point framebuffer object(FBO) as render target. Most of the implementations of HDR available use pbuffers, which became deprecated with the introduction of FBOs. FBOs are faster than pbuffers because they don't require context switching.

After rendering, the contents of the floating-point texture are retrieved to the computer's memory in order to calculate the average brightness of the scene, and from that value the target exposure. For each pixel i , the brightness is calculated according to the following formula:

$$\frac{1}{n} \sum_i 0.2125R_i + 0.7154G_i + 0.0721B_i^5$$

where n is the number of pixels. The formula that ties the actual exposure value with the target exposure is:

$$e_{actual} = e_{actual} + \frac{e_{target} - e_{actual}}{8}$$

Once the target exposure value is at hand, the floating point texture is mapped to a full-screen quad and rendered. The tone mapping is done in the GPU. It is a very simple operation where the pixel value is multiplied by the exposure and γ -corrected.

⁵The coefficients are Rec. 709 standard luma coefficients.

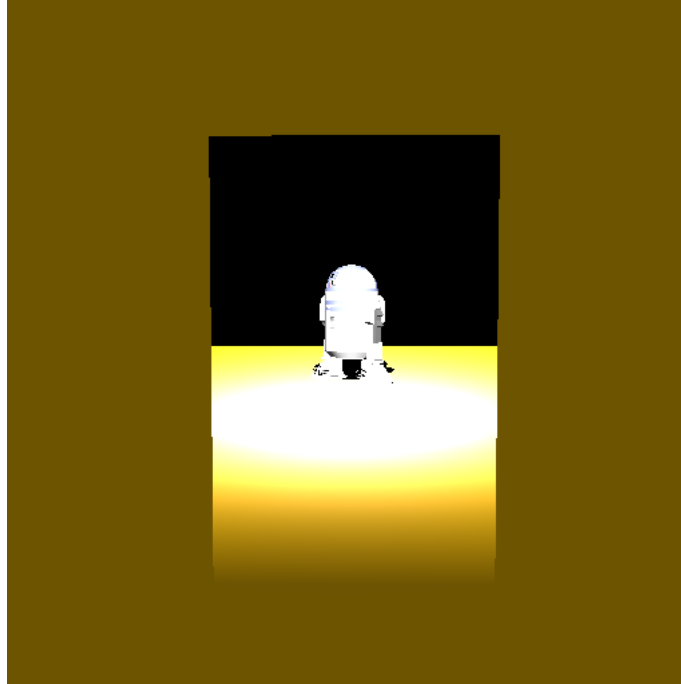


Figure 2: A screenshot with low brightness thus a high exposure value

4 Application

The above process is mostly used in computer games, to simulate the adaptability of the eye to the brightness of the current scene. But, all of the commercial applications that I've found use it in a Direct3d environment. This project is one of the first implementations of HDR in OpenGL and GLSL.

References

- [1] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pp. 369–378, Aug. 1997.
- [2] P. Debevec, "Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography," in *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pp. 189–198, July 1998.
- [3] P. Debevec, "Image-based lighting," *IEEE Computer Graphics & Applications*, vol. 22, pp. 26–34, Mar./Apr. 2002.
- [4] G. J. Ward, "The radiance lighting simulation and rendering system," in *Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, pp. 459–472, July 1994.