

High Dynamic Range Rendering

Implementation notes, part 2

Kennet Johansson

2008 - 2009

Introduction

This document serves as a continuation to the implementation notes accompanying the demo produced for the second milestone of my HDR Rendering project. The previous files can be found at <http://gscept.com/projects/08/ip16/delivery/MS2/>. Since that point I have continued to refine and add to the HDR rendering algorithm as well as implemented loading of float textures (although admittedly most of this is rightly attributed to the excellent tools provided by Industrial Lights and Magic for usage with their OpenEXR[1] format, which is what I've used).

Float textures

The implementation of float textures was made simple through the usage of ILMs tools for loading and storing textures of their open .exr format. Actually compiling the library under my environment was a task in itself which taught me a fair bit about how these things work (if there is one thing about programming that courses generally lack, it must be low level tasks like compiling and linking), but I shall not go deeper into that.

The data loaded by the library was then used by sending it into a texture object. This texture was then used as normal. However, as mentioned in the previous implementation document I have had to add existing modifications as well as myself modifying the Irrlicht engine in order to allow the usage of 64-bit floating point textures (rgba16, a 16-bit float is known as a half). The general additions to the engine were described in the previous document, the compatibility modifications done by me were mostly limited to adding cases for usage of the added texture types. I should also mention that I only worked with OpenGL and as such the DirectX part of the engine remain unchanged and as such would not work.

Further work on HDR algorithm

Generally the further developments of my implementation of the HDR rendering algorithm have been focused on generalizing the part by freeing the parameters to be set at runtime instead of within the files. This has enabled me to experiment on the best ways of using each parameter individually and separately.

Refinement in luminance calculation

Since the last delivery, aside from generalizing the final post processor, I have finished the luminance calculation and solved some problems in it. These problems were largely related to divisions by zero and similar which at times caused black screens or even worse images that were slightly darker than intended. The reason why this is worse than blatantly erroneous images is because they are much less obvious. There wasn't really much to be done in the algorithm itself though experiments with the parameters have shown the proper usage of the technique.

Refinement in effects

Bloom has remained an area where a great deal of time has been spent. The reason for this is because of how hard it is to properly use and how many ways there are to use it. Even now I'm not certain that I'm satisfied with it. When calculating the bright pixels there is incentive to tone map them using the final calculated luminance in order to find the bright areas for a specific general luminance; and indeed, I do use a linear mapping when checking which pixels are bright but at this time the original pixels are the ones that are saved. It is hard to find what works best, but from my experiments with the current implementation I've found some ways of effectively using this effect. No other effects have been implemented, there was no time.

Luminance adaptation

The implementation of luminance adaptation is made up of a simple algorithm taking the luminance of the scene, the previous adapted luminance and the time since the last frame for inputs. In a system set to a constant framerate one could use a constant for this, and this would indeed be simpler. For I had some trouble getting small and even enough timesteps to make the transition smooth. I solved this by calculating an average time between frames using a frame count and the total time of the simulation. There might be a better way of doing this (by calculating the average of the most recent frames for example) but I didn't place much focus on this aspect.

Experiments

As I got the scene set up for the demo I still had to make a good configuration for the shaders, thus I set about testing the various parameters. Ultimately based on my findings I created the three presets seen in the demo. With the presets I've attempted to create interesting and hopefully good looking lighting for the scene. Because the lighting shader is so badly implemented I'm afraid the overall presentation suffers a bit. Also, the bloom really could be better.

Conclusion

Over these last few months I've implemented HDR rendering. I feel the result could have been better in areas such as glares and similar effects and also in computation efficiency which I never had serious problems with but which I suspect could become troublesome if there were other computation heavy tasks to do as well (like physics or AI). I have no plans to continue working on this project but if I did then those would be my most important focuses.

References

[1] <http://www.openexr.com/>