Imperial College London
Department of Computing

C417: Advanced Graphics
Coursework 1

Jean Kossaifi (jk712)
Romain Brault (rb812)
.

Imperial College
London

Module leader & Lecturer: Dr Abhijeet Ghosh.
February 18, 2013.

Contents

1 Introduction

The aim of this coursework is to implement to simple algorithm. The first one assemble an HDR image and tone map the result so it can be rendered on a screen. The second algorithm map a latitude longitude map on a sphere.

To obtain the maximum throughput we have followed the advices proposed by Intel [**?**] to represent an image. The idea is to represent an image as blocks of 8 pixelsn with continuous red green and blue pixels chanel. This representation allows the compiler to vectorise the code of some loops, allowing them do compute multiple values at the same time. Of course writing and reading images on the hard drive takes more time, however we the improvement during the HDR assembly or the sphere mapping compensate this overhead.

R_1	G_1	B_1	R_2	G_2	B_2	R_3	G_3	B_3	R_4	G_4	B_4
R_5	G_5	B_6	R_6	G_6	B_6	R_7	G_7	B_7	R_8	G_8	B_8

Table 1.1: Classic image representation.

R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	G_1	G_2	G_3	G_4
G_5	G_6	G_7	G_8	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8

Table 1.2: Vectorised image representation.

We have also slightly alter the proposed algorithm to obtain better visual results. Theses modifications are local and do not change the general flow of the algorithms. They are all explained below.

We have choosen to do this coursework in C++ since this language allows us to use some high level tools and simplify the developpement but is also a language widely use for High Performance Computing and Computer Vision. We also used the Qt libray to build a simple GUI and OPenMP to parallelise the computations. The binaries *HDRimage* and *relightning* should be compatible on every x86_64 linux machine (we tried them in the lab and on our personal computer). To use them:

- clic on *Files* then Open and select all the memorial images;
- if necessary, adjust the coefficients and clic on *Set Coefs*;
- clic on *Create HDR*;
- adjust the Stops and Gamma with the sliders;
- clic on *Files* then *Save* to save the image as a *.pfm* or *.ppm*. The *.pfm* file is saved as the raw HDR image, whereas the *.ppm* file is saved after the tone mapping and gamma correction.

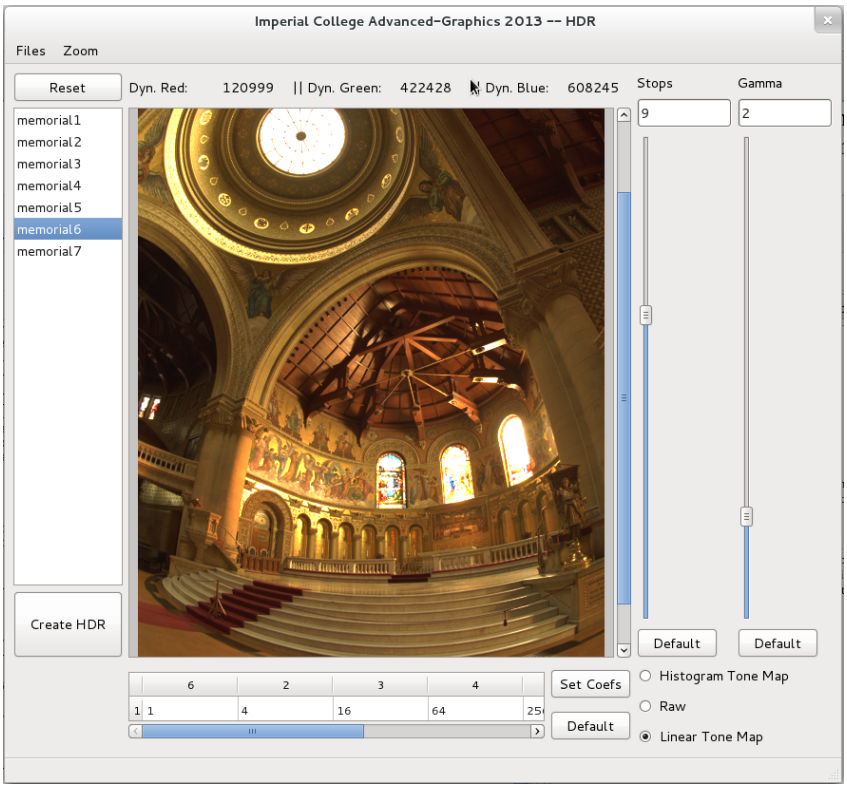

Figure 1.1: GUI for HDR assembly.

2 Assemble an HDR Image

2.1 Algorithm

We first followed the instruction proposed by the coursework and the publication of Debevec & Malik [**?**]. We used the following polynom: $w(x) = 16x^2(x-1)^2$. We have $w(0) = 0$, $w(1) = 0$, $w(0.5) = 1$ and $w'(x) = 32x(2x^2 - 3x + 1)$ thus $w'(0) = 0$ and $w'(1) = 0$. We have choosen a polynom rather than a trigonometric function because a polynom is faster to compute.

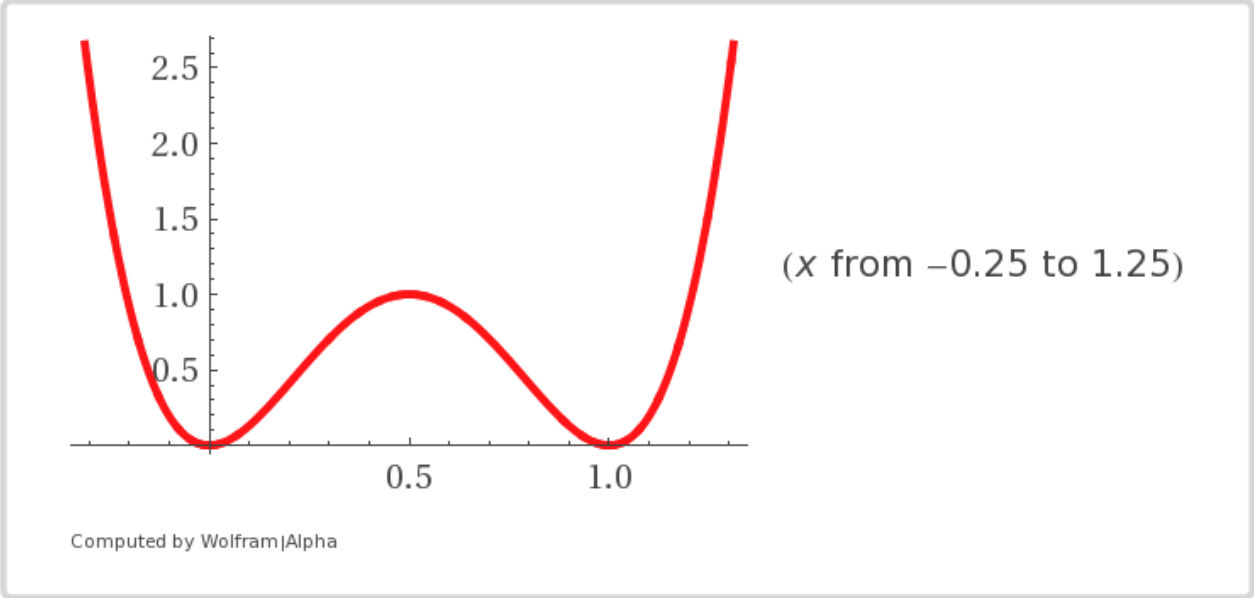

Figure 2.1: Polynomial curve used for the HDR weightening

However the result were not as good as expected. We noticed mutiple chromatic artifacts, circled in red on Figure **??**.

Figure 2.2: First result, stops = 9, gamma=2

One can notice a bunch of blue points on the window (2) which should be white, and two blue points on the wood which should be brown. The blue artifacts on the window are due to an indeterminate value of the exponential. Some pixels are so bright that they are ignored on each base picture. Thus the total sum of weight is null and the value of the fraction $\frac{\sum_i log(E_i(x,y))w(Z_i(x,y))}{\sum_i w(Z_i(x,y))}$ undeterminate. We have opted for the following solution: each time a pixel is ignored, we change the value of a *balance* variable. We increase it by one if we are over 0.92 and decrease it by one if we are under 0.005. After having scan all the images for a pixel (x,y) we replace the pixel value by 1 if the balance is positive and 0 if the balance is negative. If the balance is 0 then we do the normal computation: $F(x,y) = exp\left(\frac{\sum_i log(E_i(x,y))w(Z_i(x,y))}{\sum_i w(Z_i(x,y))}\right)$. We obtained the following result:

Figure 2.3: Second result, stops = 9, gamma=2

We can see on Figure **??** that the blue artifacts on the window have been removed. However we can still see the two blue points on the wood. We analysed the seven given pictures and noticed that this blue points were due to two white points on the memorial 2 ($\Delta_t = 2$).

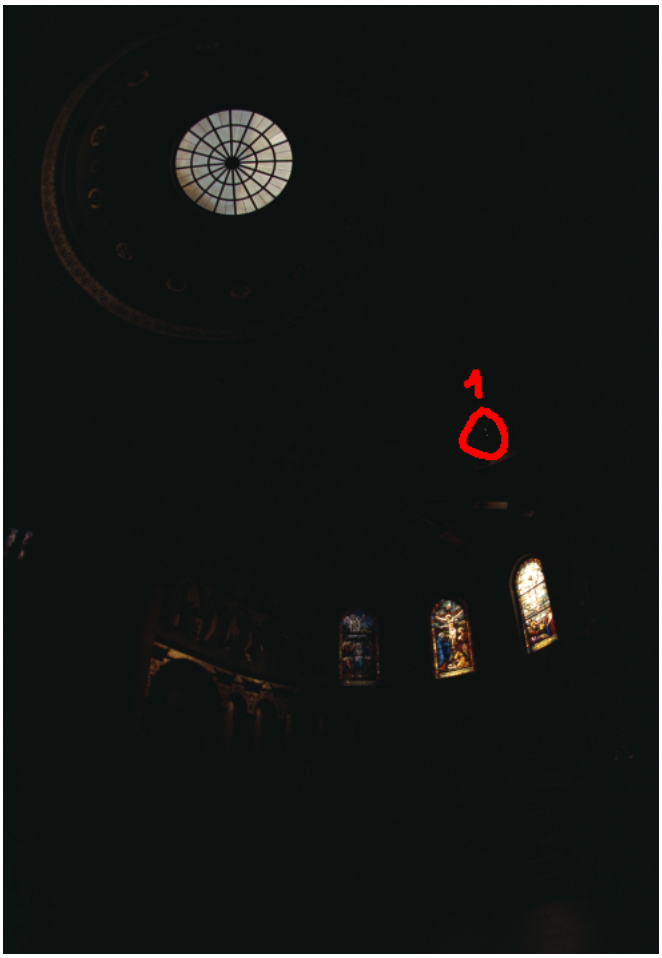

Figure 2.4: Memorial $\Delta_t = 2$

This points should not be white and are probably caused by a defect of the camera. Indeed, when the exposure time Δ_t increase, the luminosity of a pixel should only increase since we are trying to find a constant radiance value. Hence we have decided to also ignore a pixel if its luminosity on picture l, with exposure Δ_t is over its luminosity on picture $l+1$ with an exposure of $\Delta_{t+1} > \Delta_t$.

2.2 Final results

We found the following dynamic range on each color chanel:

- **red:** $1,20999 * 10^5$;
- **blue:** $4.22428 * 10^5$;
- **green:** $6.08245 * 10^5$;

Of course stops=0 means that we tone map from $[0;1] \subset \mathbb{R} \rightarrow [0;255] \subset \mathbb{N}$, thus no exposure function has been applyed. Simillary, if gamma=1, no gamma correction has been applyed. In this report only the base and the best looking pictures are presented. The others are availables in the folder "*Results/part1/ppm*".

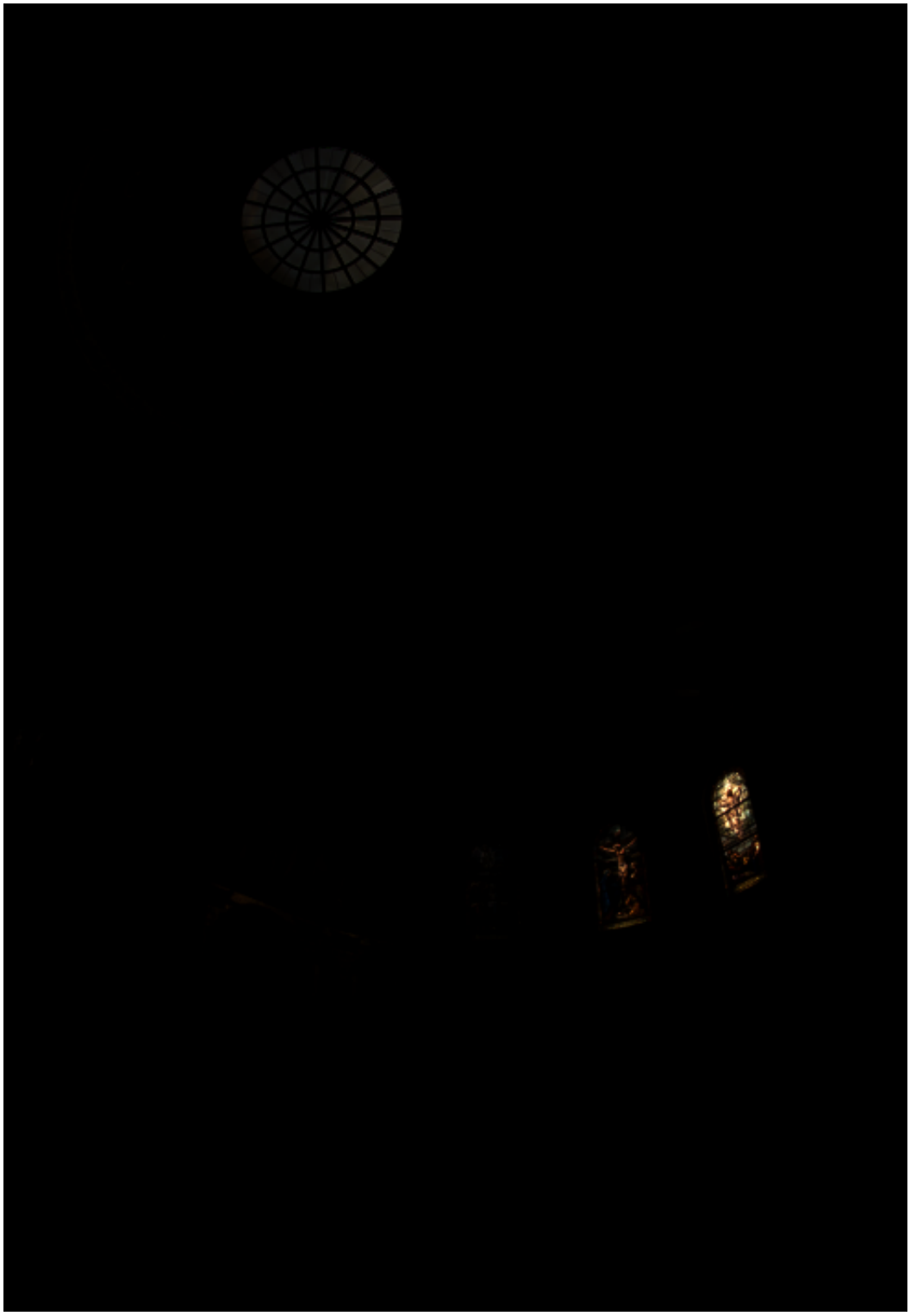

Figure 2.5: Memorial stops=0, gamma=1. This is the image with the simplest tone map function.

Figure 2.6: Memorial stops=9, gamma=1. This is the best stops parameter.

Figure 2.7: Memorial stops=9, gamma=1.5. This is the best looking picture we found.

Figure 2.8: Memorial stops=9, gamma=2. Here gamma is too large, colors are flatten.

Figure 2.9: Memorial stops=9, gamma=2. Here we tried with a new function $w(x) = sin(2\pi x - \frac{\pi}{2}) + 1$. We can see no differences with bare eyes from the picture generated with the polynom. The new dynamic range is red=$1,20931 * 10^5$, green=$6.08245 * 10^5$, blue=$4.22428 * 10^5$. With 6 digit precision, only the dynamic range of the red chanel have changed, showing that de difference between those two function is slight.

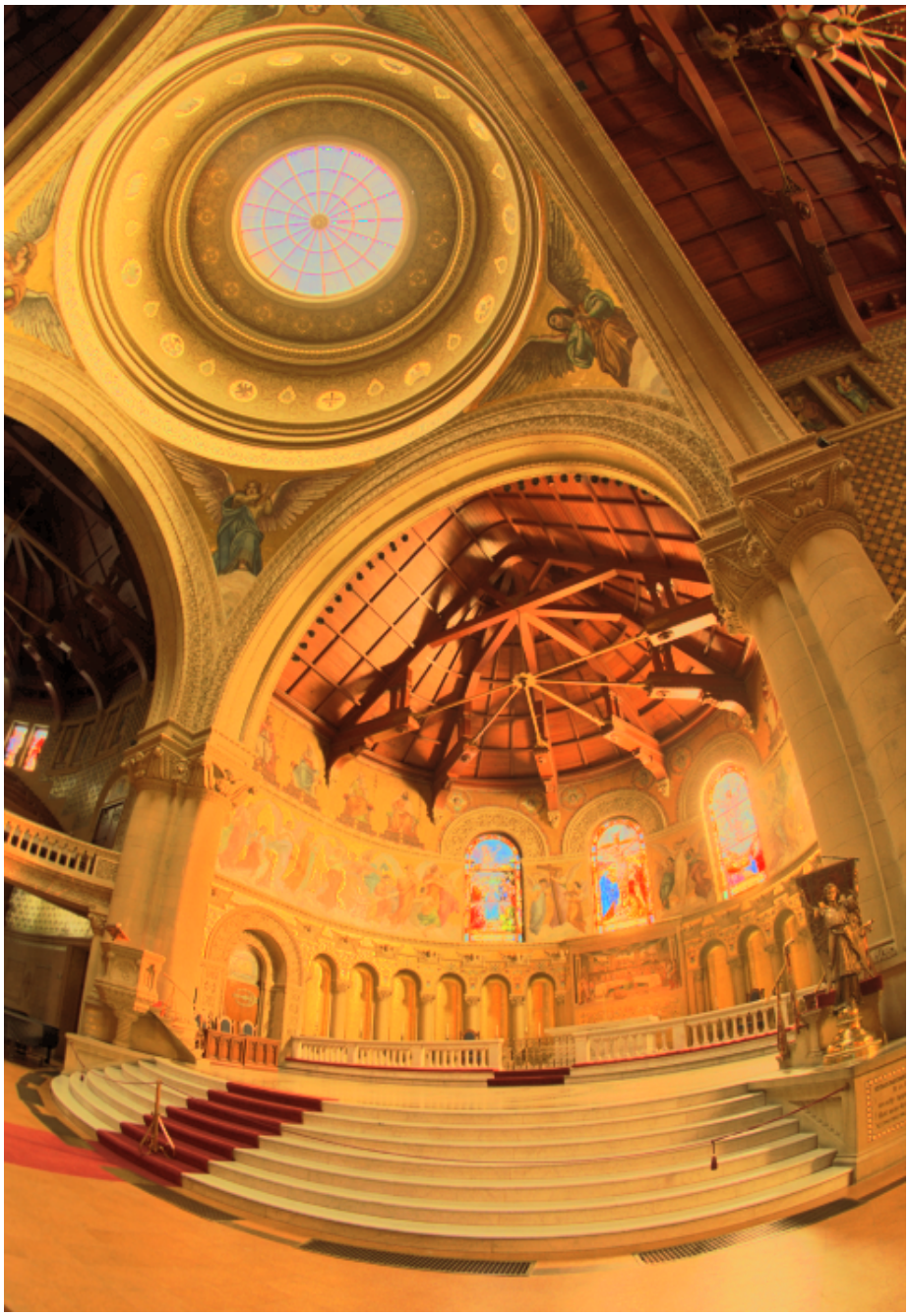

Figure 2.10: Memorial stops=−0.63, gamma=1.64. We tried another tone map based on an histogram expansion on grey value. The image looks less realistic (colors are not preserved) however we can see more details on the windows. The purple colors at the top of the image seems to be some artifacts due to the saturation of the pixel luminosity. Maybe a local histogram equalisation would have lead to better results.

Figure 2.11: Memorial stops=0, gamma=1. Eventually we tried an histogram equalisation independently on each chanel. The results seems more natural but the colors are not preserved. This feature has been removed from the program since this tone map does not preserve the colors.

3 Simple Image Based Lighting

3.1 Reflection sphere

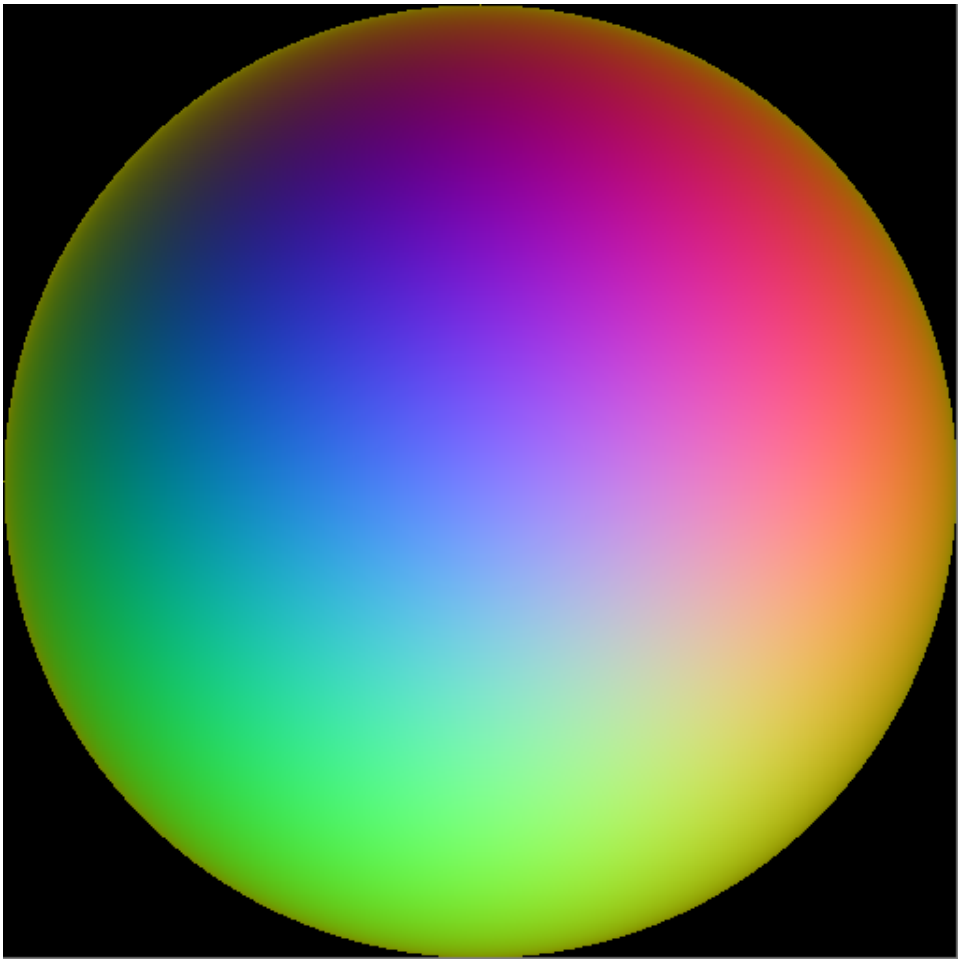

Figure 3.1: Reflection sphere for $v = (0, 0, 1)$, reflectivity $= 1$.

3.2 Final results

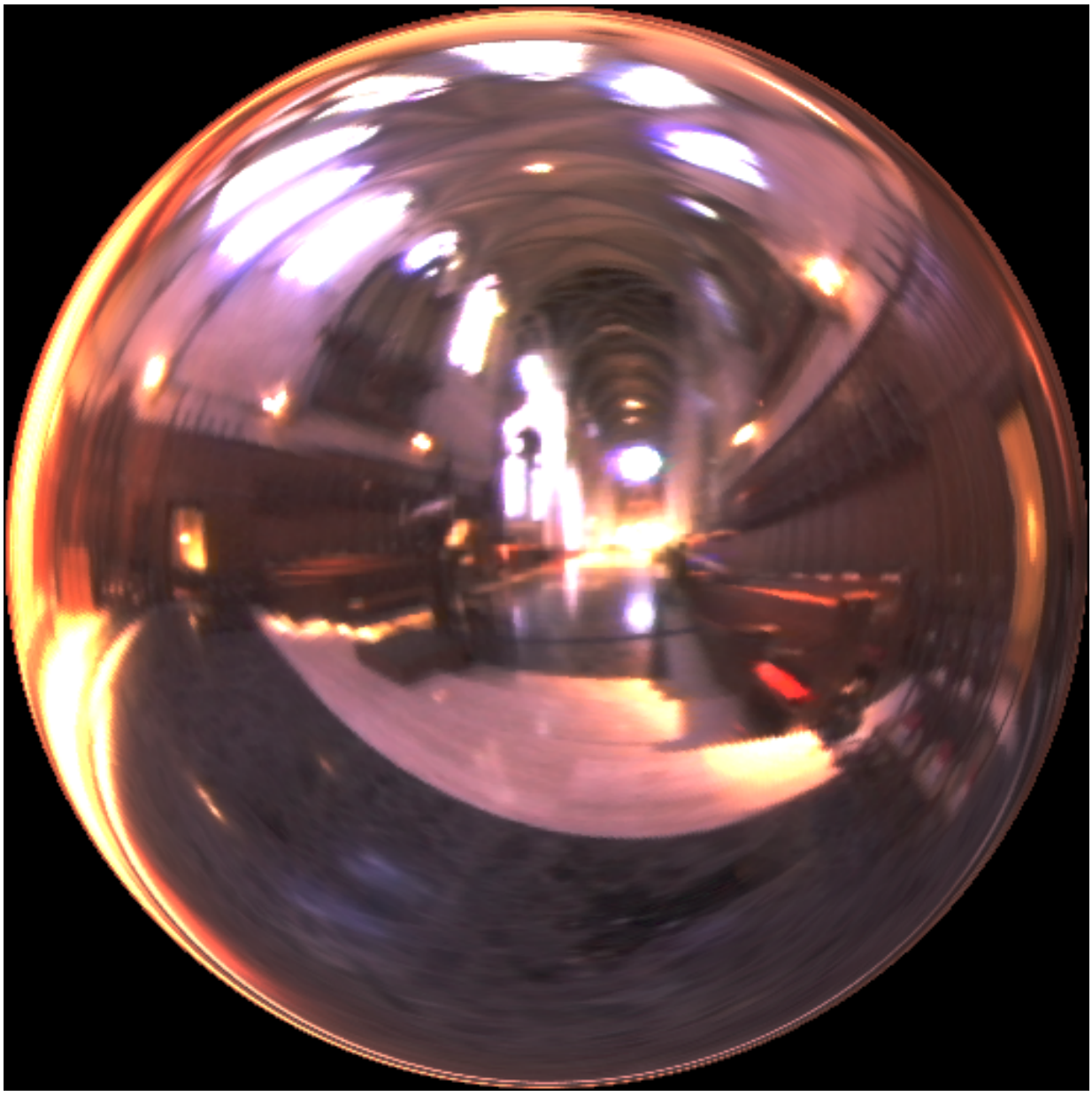

Figure 3.2: Latlong mapped on the sphere.

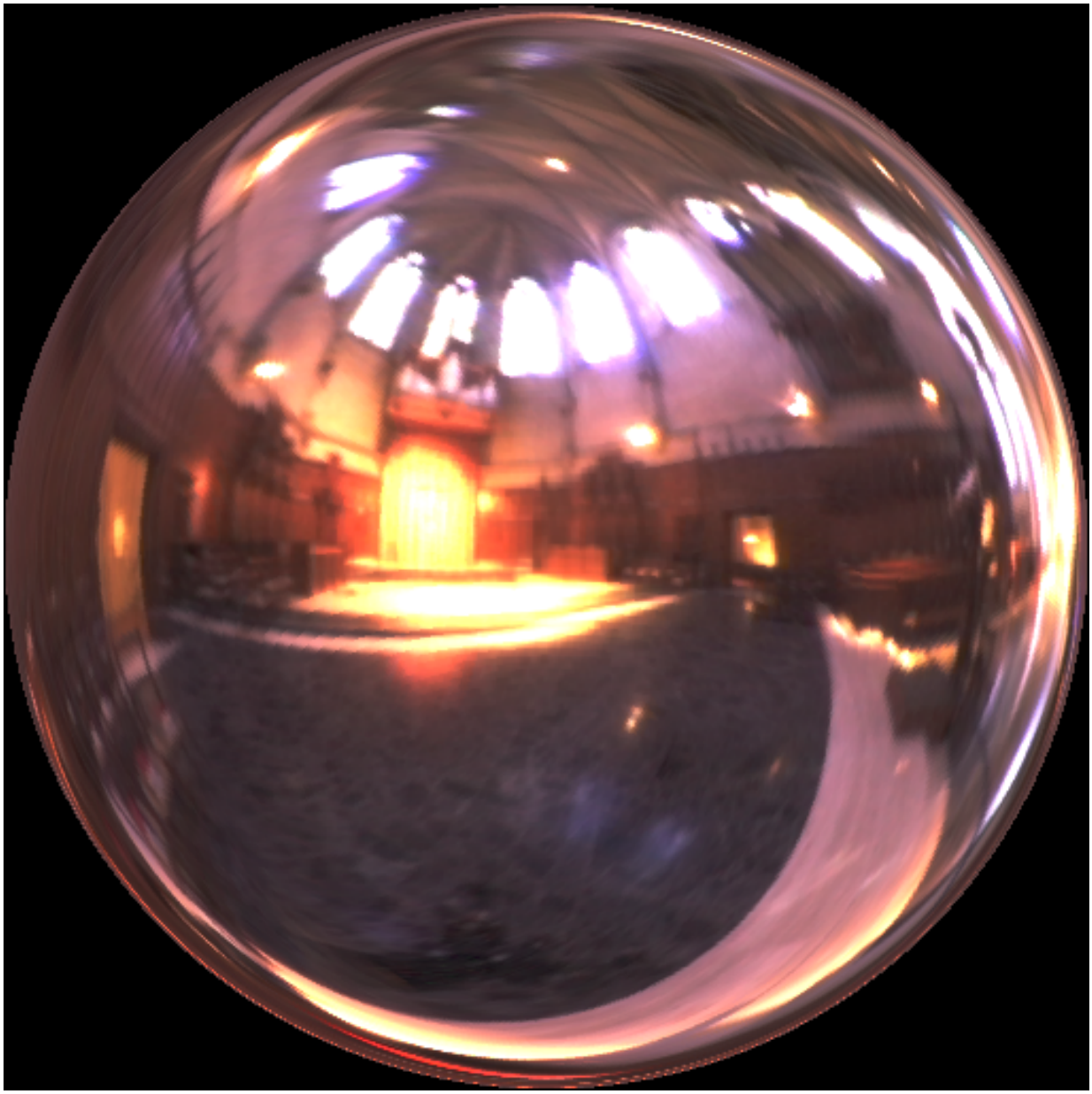

Figure 3.3: Latlong mapped on the sphere. We can add a rotation effect by applying an offset on the ϕ axis.