Motivaition

My goal is to create an explosion process with fire and smoke. Creating natural phenomena is interesting to me. And the fact that we are only using some physical information to render these complicated process is also intriguing.

Thus I extended PBRT to support emissive participating media, both homogeneous and heterogeneous with OpenVDB support and also implemented multiple importance sampling for emissive volumes.

Explosion

Approach

Blackbody radiation

The radiance emitted by fire/explosion can be approximated by black-body radiation. According to the property of an ideal black body in thermal equilibrium: the energy is radiated isotropically, independent of direction, we know the phase function is isotropoic.

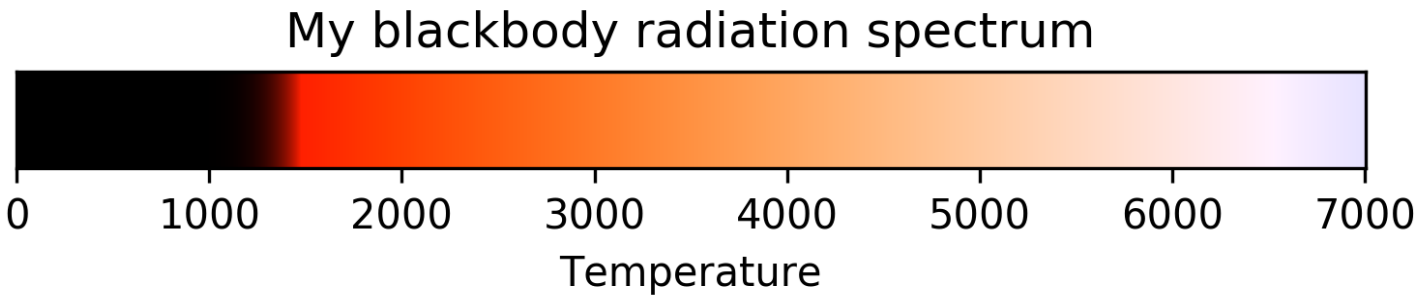
Planck's Law of blackbody radiation, a formula to determine the spectral energy density of the emission at each wavelength at a particular absolute temperature (T):

$$E_\lambda = \frac{2hc^2}{\lambda^5(e^{hc/\lambda k_b T} - 1)}$$

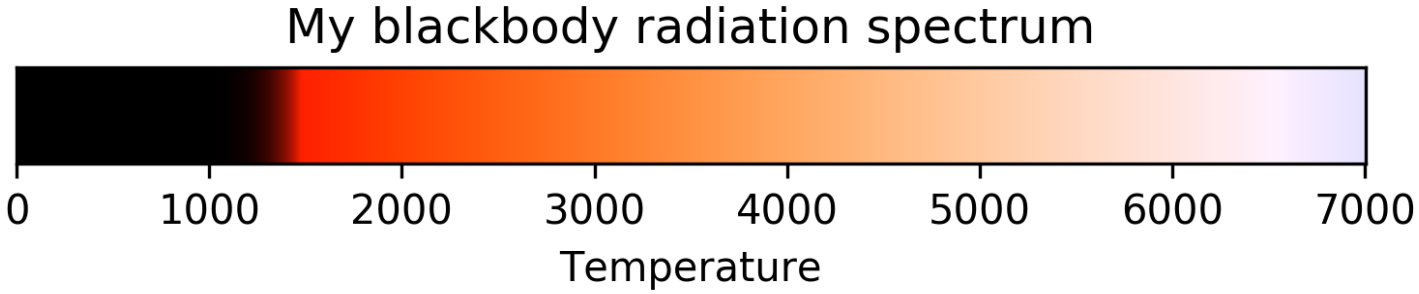
Plank's Law looks simple, but implemented it correctly in PBRT requires certain efforts since PBRT uses RGB spectrum for ray tracing, so we need to compute the RGB value from temperature. Basically, the algorithm I use is as follows:

1. Sample a bunch of wavelengths, and calculate the radiance at each wavelength by Planck's law to get an emissive spectrum power distribution (SPD). I use the wavelenghs with sampled XYZ spectral matching curve values to reduce approximation errors in step 3.
2. Normalize the radiance by the peak radiance of each wavelength by Wien's displacement law.
3. Convert the SPD to XYZ color by integration – A remarkable property of the human visual system makes it possible to represent colors for human perception with just three floating-point numbers. The tristimulus theory of color perception says that all visible SPDs can be accurately represented for human observers with three values (the XYZ). Given an emissive SPD, these values are computed by integrating its product with the spectral matching curves for 3 XYZ channels. These curves were determined by CIE after a series of experiments with human test subjects. Therefore, we perform a numeric integration to get XYZ. (I see some implementation that doesn't do step 2, but linearly normalize the XYZ values so that they sum up to 1, the generated spectrums are visually similar.)
4. Convert XYZ to RGB.
5. The above conversion is linear transformation and the obtained RGB values may not lie in [0, 1], so we need further normalizations: we first linearly normalize it so that the maximum value among the 3 channels is a value x. We finally clamp the negative RGB values to 0.

The RGB spectrum of different temperatures obtained from my algoirthm as a verification:



I also implemented the chromatic adaptation of fire in [1,2], the result is as follows. Since this spectrum is less dynamic, and I aim to create a general purpose emissive spectrum, I stick to the version without chromatic adaptation.



Closed-form tracking for emissive homogeneous medium (extending the volume path tracer)

Since the implementation for emissive heterogeneous medium can be very complicated and can be hard to debug, I started from homogeneous medium, which shares the same emissive volume path tracer with the heterogeneous medium.

At first, I tried to reimplement [1], but while ray marching once served as the go-to volumetric rendering method, tracking algorithms (inherited from nuclear and plasma physics) are now the norm [2]. Personally, I feel tracking algorithms are way better integrated with the monte carlo path tracing integrator and are really elegant and efficient with good performance.

The light transport equation for close-form tacking is:

$$L(\mathbf{x}_j, \omega_j) = \int_{t=0}^{d} p(t_j)\Big[P_a L_e(\mathbf{x}_{j+1}, \omega_j) + P_s L_s(\mathbf{x}_{j+1}, \omega_j)\Big]dt + L_d(\mathbf{x}_{d,j}, \omega_j) \int_{t=d}^{\infty} p(t_j)dt$$

,
where

$$P_a = \frac{\sigma_a}{\sigma_t}, \qquad P_s(\mathbf{x}) = \frac{\sigma_s}{\sigma_t}$$

and

$$p(t) = \sigma_t \exp(-\sigma_t t).$$

[3].

I made an emissive homogeneous medium class and created a new emissive volume path tracer. Indeed, it's very general and supports non-emissive medium as well (the same applies to my heterogeneous implementation).

I figured out how to make the volume path tracer myself. And the LTE is different from the textbook – the emission term is multiplied by the absorption parameter, this is indeed correct since I found that we need to terminate the ray tracing when the particle the ray met is emitting – which is indeed absorption, otherwise, the rendered results are extremely bright.

OpenVDB support for large volume data files

Now that we move on to the heterogeneous medium, at first I tried to use the grid medium input format in PBRT, however, as users may use large volumetric data, simply opening large text files is slow for users to modify the medium's properties. Also, lots of volumetric data are in the OpenVDB file format by DreamWorks Animation [5].

I linked PBRT with the OpenVDB library, and then made my emissive heterogeneous medium to read directly from OpenVDB. This implementation needs some work, and it turns out to be really helpful: I can now experiment with different volumes by simply changing the path to the input openvdb file.

It is also a necessary step, since later I found the openvdb files exported from Blender has different grid scales for temperature and density, incoporating libopenvdb and using its API can be very efficient to deal with this situation.

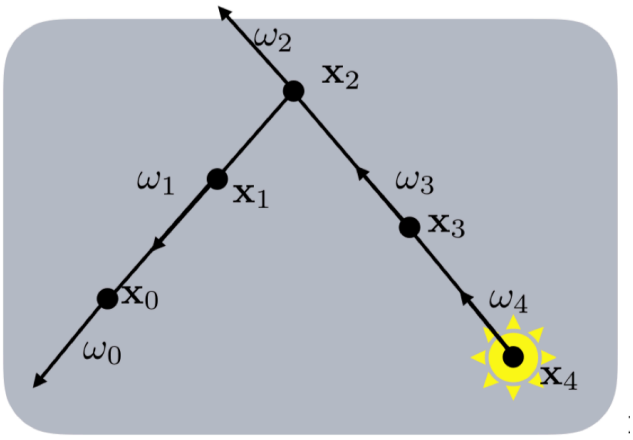
However, as I switched from grid data to openvdb library, I found that the rendering becomes slower – each time reading from the grid is now using a tree structure rather than an array. (the always-reading-from-openvdb implementation saves memory and is suitable for super large volumes though, so it has

its own advantage but is not suitable for my project).

So I came up with a new solution that combines the flexibility of reading from openvdb files directly and the efficiency of arrays – I implemented a new class that reads from openvdb and then saves the data at each density voxel to arrays (need to interpolate to get values from the temperature grid with different scales).

Delta tracking for emissive heterogeneous medium

Delta tracing is an unbiased estimation for non-uniform particapating medium. Its main idea is to populate the medium with non-collision particles so that the medium is uniform. When the tracking meets non-collision particles, it simply does nothing and goes forward, as is shown in the illustration below (x1 and x3 are non-collision particles).



The LTE is:

$$L(\mathbf{x}_j, \omega_j) = \int_{t=0}^{\infty} p_n(t_j)\Big[P_a(\mathbf{x})L_e(\mathbf{x}_{j+1}, \omega_j) + P_s(\mathbf{x})L_s(\mathbf{x}_{j+1}, \omega_j) + P_n(\mathbf{x})L(\mathbf{x}_{j+1}, \omega_j)\Big]dt.$$

, where

$$P_a(\mathbf{x}) = \frac{\sigma_a(\mathbf{x})}{\bar{\sigma}}, \qquad P_s(\mathbf{x}) = \frac{\sigma_s(\mathbf{x})}{\bar{\sigma}}, \qquad P_n(\mathbf{x}) = \frac{\sigma_n(\mathbf{x})}{\bar{\sigma}}$$

[3].

I used delta tracking with emission for my emissive heterogeneous medium. The algorithm itself is brilliant, I cannot see any adaptation needed. Finding the working algorithm itself is very difficult, since I first tried to use [1], and then [4], and the delta tracking is the third method I tried.

Multiple importance sampling for emissive medium

I found the MIS alorithm very interesting and its advantage is rigorously proved. So I implemented [4] for my emissive media, otherwise when the surface is diffuse and the emissive volume is small, sampling by BRDF may miss the emissive volume.

PBRT's MIS implementation integrates over solid angles. I chooses to integrates over volume since to compute the propability of sampling each solid angle, we still need to resort to the voxels and integration, and the integration could be slow. I simplified the equations here and there to reduce errors. And I spent some time to figure out that p(x) and p(t | omega) can be solved by ratio tracking rather than complicated integration as in the paper. I also implemented a Distribution3D class to sample from the 3D volume based on each voxel's radiance. Since we need a single value for the probability of sampling each grid and we use the 3-channel RGB to represent the radiance, I found that using the illuminance is a proper way.

Problems

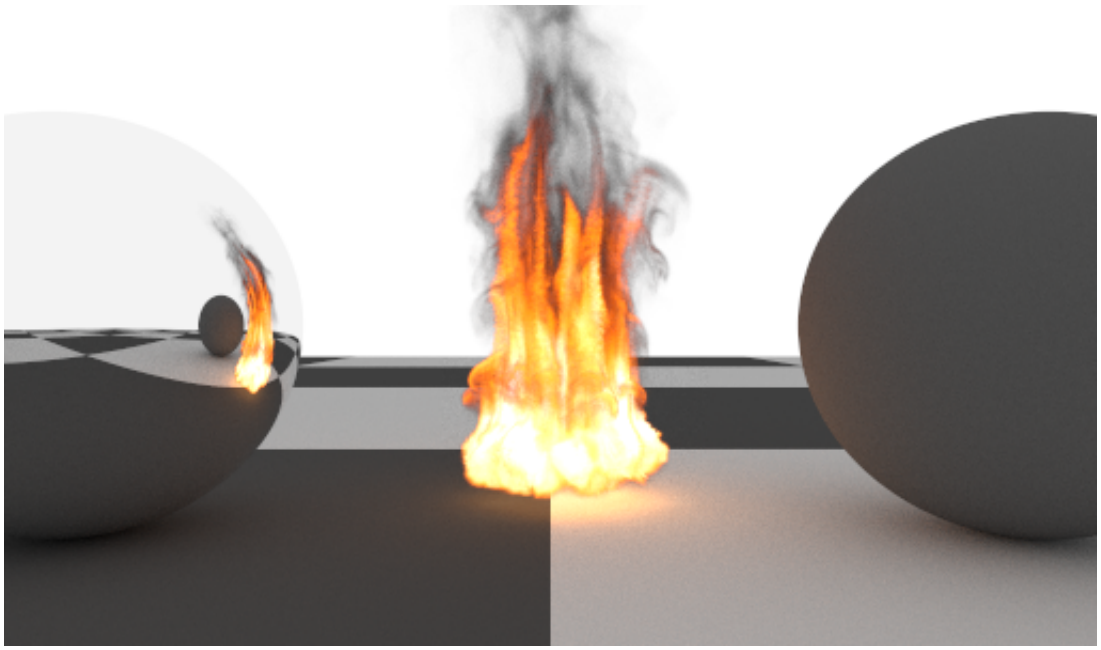
- I spent a very long time understanding participating medium, delta tracking, MIS [6], and PBRT's implementation. Since my project requires to modify PBRT here and there, I need to understand PBRT's implementation pretty well, otherwise, the debugging could be difficult. For example, I tried to use PBRT's implementation of blackbody radiation directly, but it turns out to be incorrect in my case.
- As mentioned before, finding a good algorithm is very difficult. I feel the results could be much worse and less efficient if I chose ray marching. Delta tracking is a new algorithm, the original paper seems to not deal with emission (I could be wrong here), so I feel lucky to find [3].
- I was worried about the the sigma s and sigma a functions are never clearly stated in any references, but as I correctly implemented the delta tracking, I found that whatever value works, it's just the results are different.
- OpenVDB at first is intimidating to me, as its documentation is poor. I finally probed into OpenVDB's source code to find the APIs I needed.
- I'm new to Blender, I spent a long to get familar with it. Its documentation doesn't record what each channel in the exported data of the simulation means. Its lacking documentation also makes me debug for a long time to find out that it stores density and temperature on grids of different scales; I thought it was because I interpreted the given channels incorrectly, e.g., heat is some energy rather than temperature.

- [4] assumes flame doesn’t scatter, then it is indeed a special light source. For scattering medium, it is very difficult, since when a ray meets light source, the ray tracing stops; while for scattering medium, the ray could still scatter. So we should not treat it as a light source, but reweighting the ray tracing part for scattering and the light sampling part correctly.

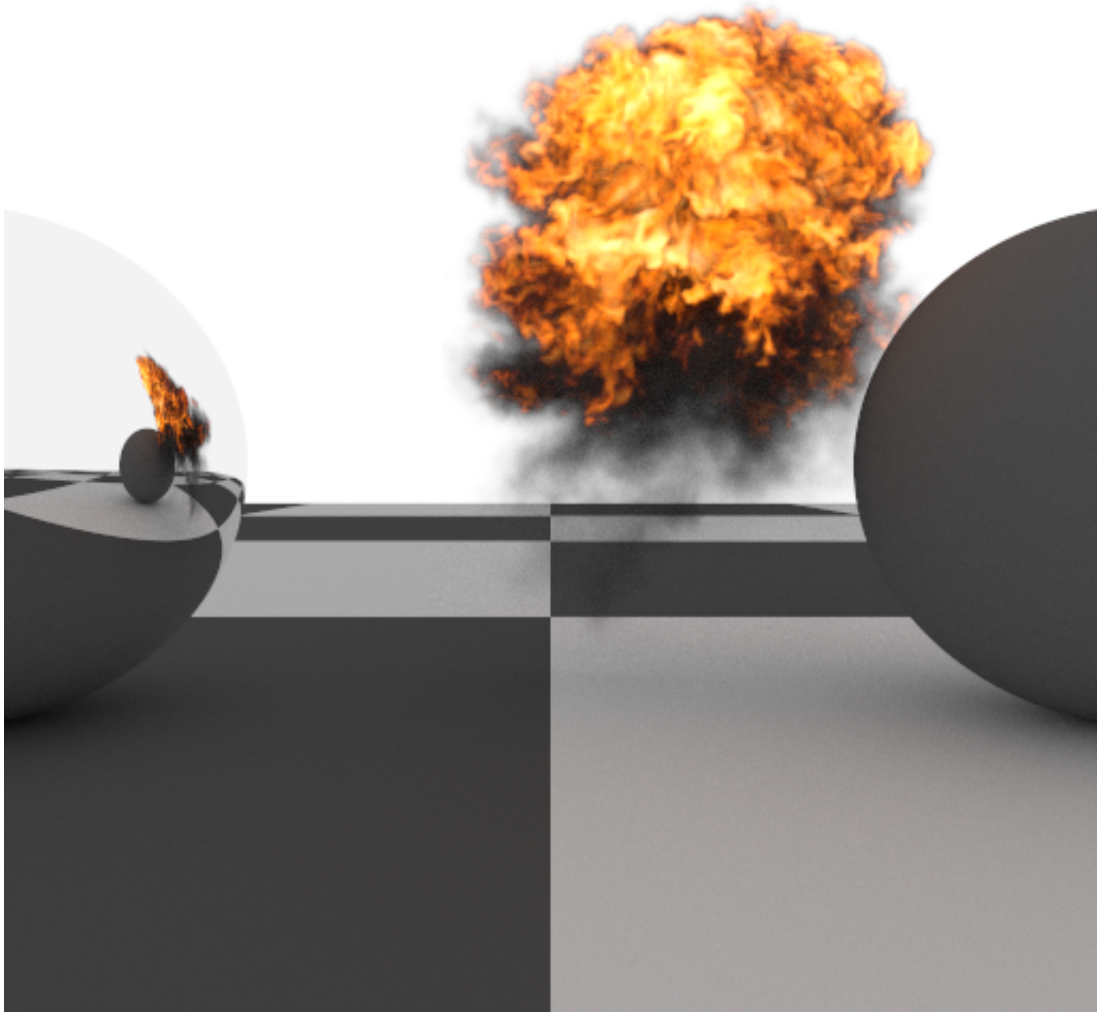
Results

Heterogeneous medium

Fire:

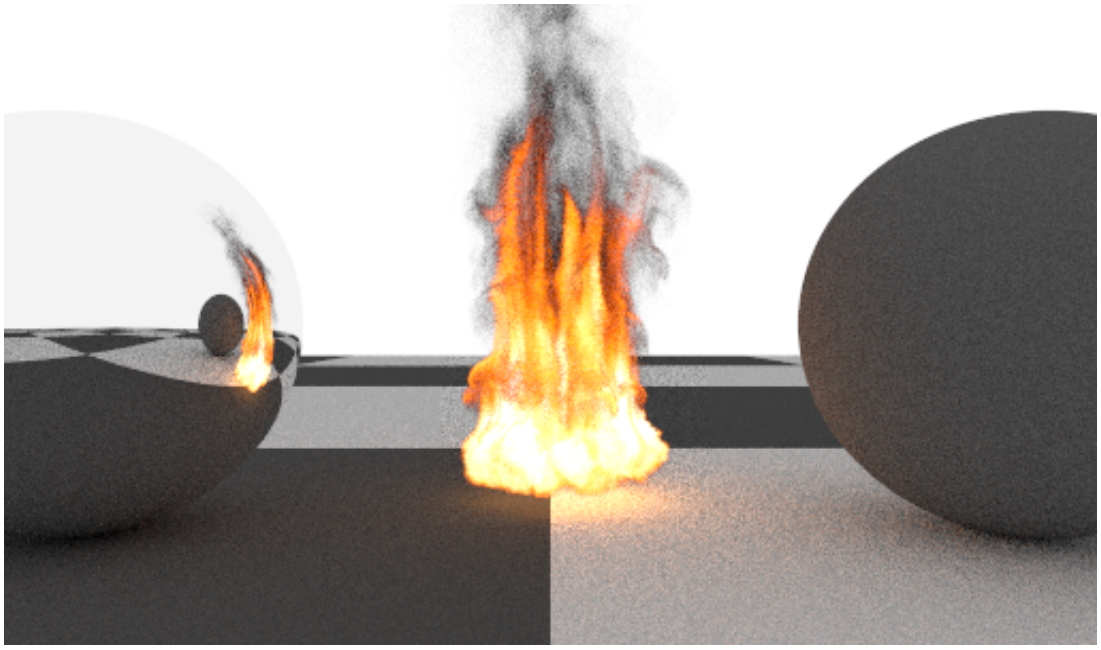


Explosion:



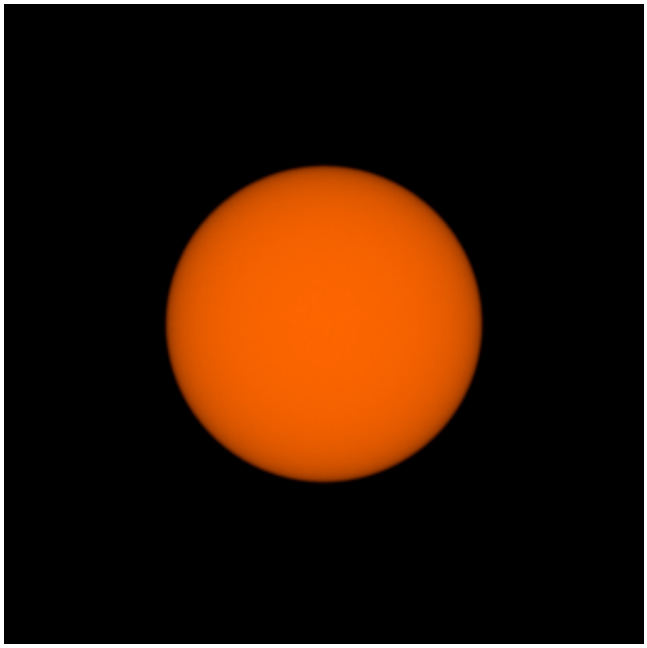
We see that both the fire and explosion are vivid, have a diversity of colors, and have no artifacts with an spp of 1024.

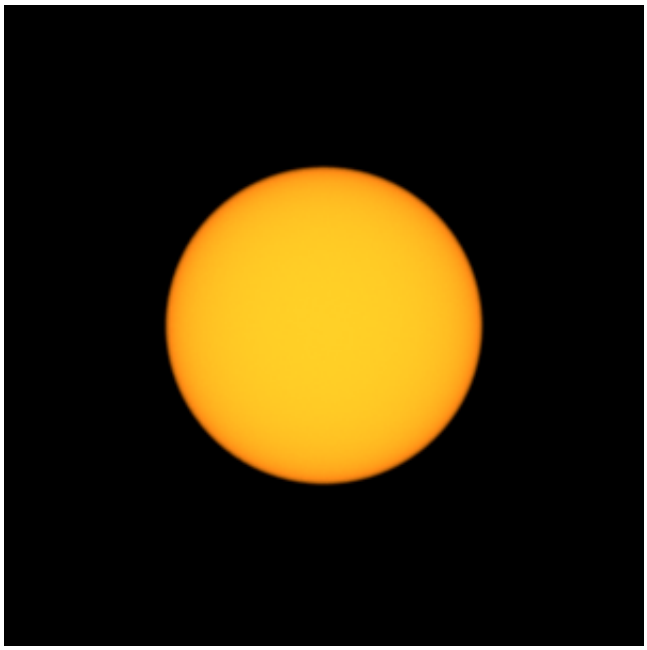
Even with an spp of 64, the variance is not very high, and the volume rendering is still good:

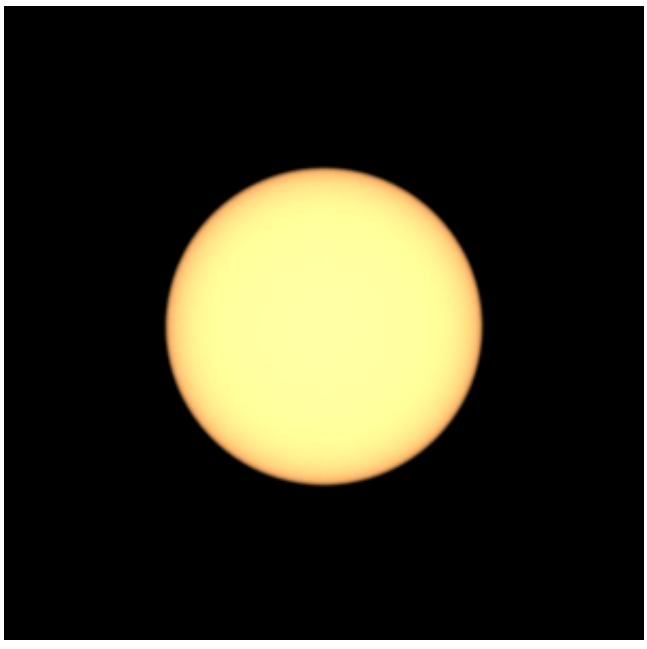


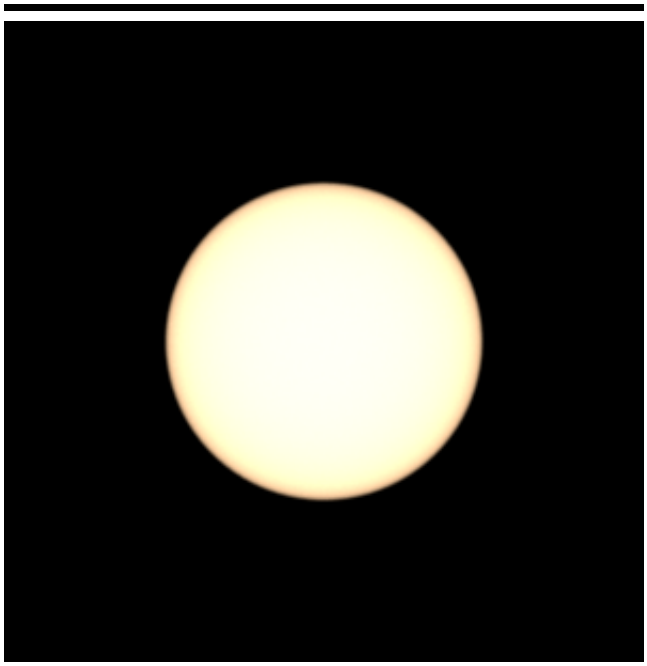
Homogeneous medium

Spheres of homogeneous emissive medium at different temperatures (with no other light source in the scene):









Multiple importance sampling

with MIS:

without MIS:

We see that the mountain is more lighted by the explosion and is thus more reddish when adopting MIS (the texture map for the mountain has no red color). Since I made the sampling from light function to sample from the explosion with some probability, the whole scene receives less light from the environment light map and is darker in general.

Explosion process

With the successful implementation of emissive volumes, I modeled and simulated an explosion process with flying rubbles in Blender. I used one particle system for the smoke and fire, one particle system for flying rubbles, and a mesh surface flow object for the mushroom cloud.

For a video with the original resolution (1980x1080), see video.

We see the single emissive volume has both fire and smoke based on the voxels' temeprature. And during the process, when the fire is raging the mountain is lighted by the explosion, and this effect gradually decreases as the fire goes out. Unfortunately, Blender is not totally physically based, so the

exported data is not very realistic.

From a different view point:

Reference

[1] Nguyen, Duc Quang, Ronald Fedkiw, and Henrik Wann Jensen. “Physically based modeling and animation of fire.” ACM Transactions on Graphics (TOG). Vol. 21. No. 3. ACM, 2002.

[2] Fairchild , M. 1998. Color Appearance Models. Addison Wesley Longman, Inc.

[3] Fong, Julian, et al. “Production volume rendering: SIGGRAPH 2017 course.” ACM SIGGRAPH 2017 Courses. ACM, 2017.

[4] Villemin, Ryusuke, Christophe Hery, and Pixar Animation Studios. “Practical illumination from flames.” Journal of Computer Graphics Techniques 2.2 (2013).

[5] Museth, Ken. “VDB: High-resolution sparse volumes with dynamic topology.” ACM Transactions on Graphics (TOG) 32.3 (2013): 27.

[6] Veach, Eric. Robust Monte Carlo methods for light transport simulation. Vol. 1610. PhD thesis: Stanford University, 1997.