Volume rendering of Physically Based Fire

TNCG14 Advanced Computer Graphics Programming - Project report

Viktor Nilsson
Linköping University
vikni067@student.liu.se

Axel Kinner
Linköping University
axeki412@student.liu.se

September 2, 2013

Abstract

We present a method for rendering physically based fire with help of volume rendering. The simulation produces a 3D dataset of temperatures which is then rendered with help of a basic ray casting technique and combined with a Black-body radiation conversion.

1 Introduction

Volume rendering is a widely used technique in scientific visualizations and computer graphics. The most common cases are sampled datasets out of CT scannings where the rendering is focused on the scanned density.

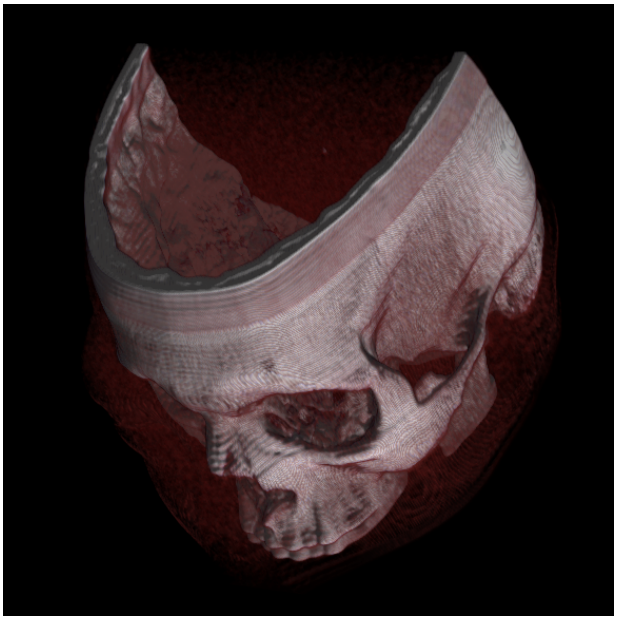

Figure 1: *Image of a cadavar head CT scan rendered using view aligned slicing with alpha blending. Lower density flesh and brain matter are assigned lower alpha values.* [3]

There are a wide range of techniques when rendering state of the art fluid simulations where the most common ones renders the actual surface of the fluid. There are also a big consideration when choosing between realtime and offline rendering.

Instead of focusing on rendering the density or the surface of a level set one could render the velocity or even the temperature. Since the human eye reacts to the light produced by the heat of a combustion i.e. flame, the temperature is the key element to the colors we see in a fire.

Our method is focusing on quality and the best looking results rather than speed and realtime effects. This resulted in that we render the simulation offline on the CPU as images, notice that we could reduce the quality and constants for the simulation and render so that we would achieve a realtime simulation and volume rendering but with very low detail level.

2 Previous work

Our project is an expansion of an already implemented fire simulation. The simulation is based on Nguyen02[7] and consists of two fluids, one for the ignited fuel and one for the surrounding fuel e.g. air. The simulation is described

by two separate Navier-Stokes equations, which can be solved numerically. The simulations are then coupled together using Ghost fluid method at the interface between the fluids. The Ghost fluid method enforces the conservation of mass and momentum when one fluid particle transforms from one fluid to another. This gives the fire the important expansion of the fluid, when the fuel is transformed into hot gas.

3 Black-body radiation

To be able to convert a temperature sample in the grid to a color, we use the black-body radiation model.

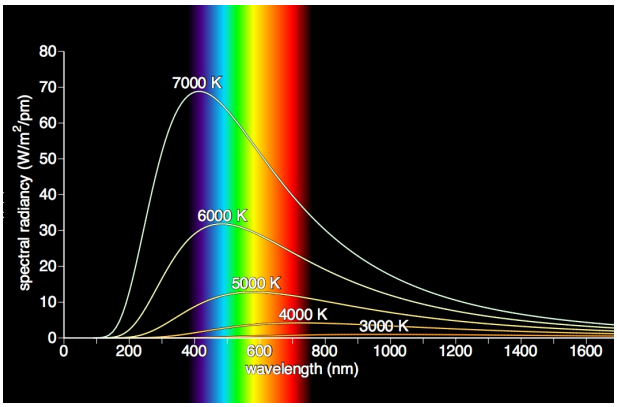

Figure 2: *Five different temperatures and their respective spectral radiancy depending on the wavelength. Notice the color spectrum that the human eye can observe, from 400 to 700 nm.* [2]

The black-body radiation model calculates an emitted radiance for a given wavelength and temperature, Planck's formula[7] Equation 1. Where $C_1 \approx 3.7418 \cdot 10^{-16} Wm^2$ and $C_2 \approx 1.4388 \cdot 10^{-2} m^\circ K$

$$L_{e,\lambda}(x) = \frac{2C_1}{\lambda^5(e^{C_2/(\lambda T)} - 1)} \tag{1}$$

This allows us to calculate the spectral radiance L for a given wavelength λ and position using the radiative transport, Equation 2, using the blackbody radiation as the emitted radiance.

$$\begin{aligned}(\vec{\omega} \cdot \nabla)L_\lambda(x,\vec{\omega}) = -\sigma_t(x)L_\lambda(x,\vec{\omega}) + \\ \sigma_s(x)\int_{4\pi} p(\vec{\omega},\vec{\omega}')L_\lambda(x,\vec{\omega}')d\vec{\omega}' + \\ \sigma_a(x)L_{e,\lambda}(x,)\end{aligned} \tag{2}$$

The scattering part, $p(\vec{\omega},\vec{\omega}')$, is neglected due to the high computation time and the fact that there are few scattering effects in flames.

The achieved radiance is then mapped onto the XYZ-color space by using the CIE standard observer 1931[1]. We then convert the XYZ colors to the LMS-color space with the CAT02 transformation method [4], Equation 3.

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.7328 & 0.4296 & -0.1624 \\ -0.7036 & 1.6975 & 0.0061 \\ 0.0030 & 0.0136 & 0.9834 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3}$$

We also add an chromatic adaption at this step to get a result as if the observers eyes and thereby vision have adapted to the intensity in the scene. Without this step a too bright flame was produced which is the case when looking at a flame with a normal to large sized pupil.

The LMS values are then converted back by inverting Equation 3. And finally the XYZ values are converted into the RGB-color space with an sRGB conversion, Equation 4[6].

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.2410 & -1.5374 & -0.4986 \\ -0.9692 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4}$$

4 Ray casting method

The radiative transport equation can be solved discreetly using ray-casting, by traversing a ray that goes from the eye and then through the volume. The temperature is then sampled several times along the ray. That gives us the discrete formulate Equation 5[7] and because we neglect the scattering effect as mentioned earlier we can also neglect $p(\vec{\omega},\vec{\omega}')$.

$$\begin{aligned}L_{n,\lambda}(x,\vec{\omega}) = e^{-\sigma_t \Delta x}L_{(n-1),\lambda}(x+\Delta x,\vec{\omega}) + \\ L_\lambda(x,\vec{\omega})p(\vec{\omega},\vec{\omega}')\sigma_s\Delta x + \\ \sigma_a L_{e,\lambda}(x)\Delta x\end{aligned} \tag{5}$$

The equation tells us that we have to traverse the ray backwards from its endpoint back to the eye, to simulate the energy loss from absorption, scattering and distance. We use a constant Δx during the whole rendering.

The ray-caster is using a perspective to give a more realistic view, which is calculated by first defining a forward, up and right vector (in unit lengths) from the eye. These vectors is then used to calculate the near plane position that belongs to a pixel (which has a u, v coordinate), Equation 6.

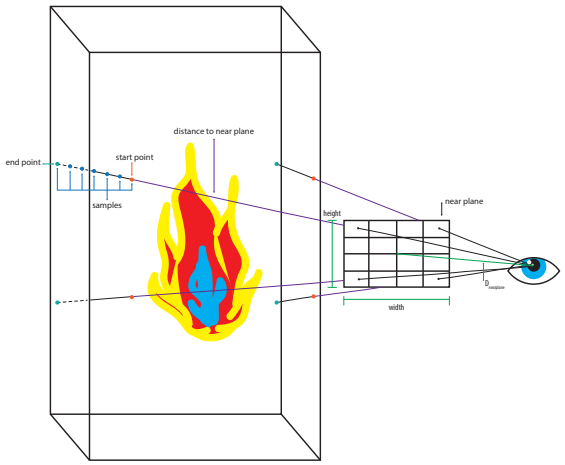

Figure 3: *Illustration of the raycasting algorithm. It illustrates how a ray starting from the viewer $\vec{x}_{eye}$, goes through the near plane $\vec{x}_{nearplane}$ and then through the first intersection point (start point) and then ends at the last intersection point (end point) of the box. The distance to near plane is the distance where the ray only lose energy. It also illustrates how the samples only exists between the start point and the end point (in this case no sample is in the fire, and would probably result in a pixel only lit by the wall). To simplify the illustraton only 4 rays are shown, in the real application one ray would go through each pixel in the nearplane, which would be 16 rays in this case.*

$$\vec{x}_{nearplane} = \vec{x}_{eye} + \vec{F} \cdot D_{nearplane} + \vec{R} \cdot u \cdot \frac{w}{2} + \vec{U} \cdot v \cdot \frac{h}{2} \quad (6)$$

Where u and v is screenspace coordinates, defined from -1 to 1 and w and h is the near plane width and height (which we calculate using a field of view calculation). This near plane position can then be used to calculate the direction of the ray.

To speed up the ray-caster we find the intersection points between the volume and the ray using a ray-box intersection algorithm. This allows us to have a start (closest intersection to the eye) and end sample point, which avoid redundant calculations outside the volume. If the eye is in the fluid, the near plane position is the start point. If the start position differs from the near plane position we have to add the energy loss between those positions.

To get a sense of depth in the rendering we render the sides of the volume as walls. We calculate the radiance that is reflected from a given point at the walls by integrate the radiance over all voxels incomming to that point, and using that value as the start value for λ in Equation 5, because the point is at the end position.

5 Discussion

We use OpenMP[5] for improving the speed when volume rendering our simulation. An average speedup on a Intel Core i7-3610QM CPU (2.30GHz, 4 cores, 8 threads) was calculated to 3.83 times faster. Notice that in some cases for specific time steps the speedup could be above 4, which is an superlinear speedup. Since the CPU has 4 cores the speedup would theoretically be impossible but due to the cache effect, where already calculated values and memories are reused; the speedup can sometimes be achieved.

The calculation of the reflected radiance from the walls that is described in x (blackbody chapter) is very naive, and therefor very slow. To improve the performance greatly we calculate the mean radiance and its mean position, for every wavelength. We then calculate the reflected radiance for each wall position from these values instead. This gives us almost the same results, but the rendering miss some small details like when a flame burst close to the roof or wall. This could also be solved more elegantly by using a Monte Carlo method.

Future work would be to add another render for the blue core. Since the blue core is due to a chemical reaction and cannot be rendered as with the flame. Also another future work is to add the very common smoke which occurs when the fire is getting a small amount of oxygen.

6 Results

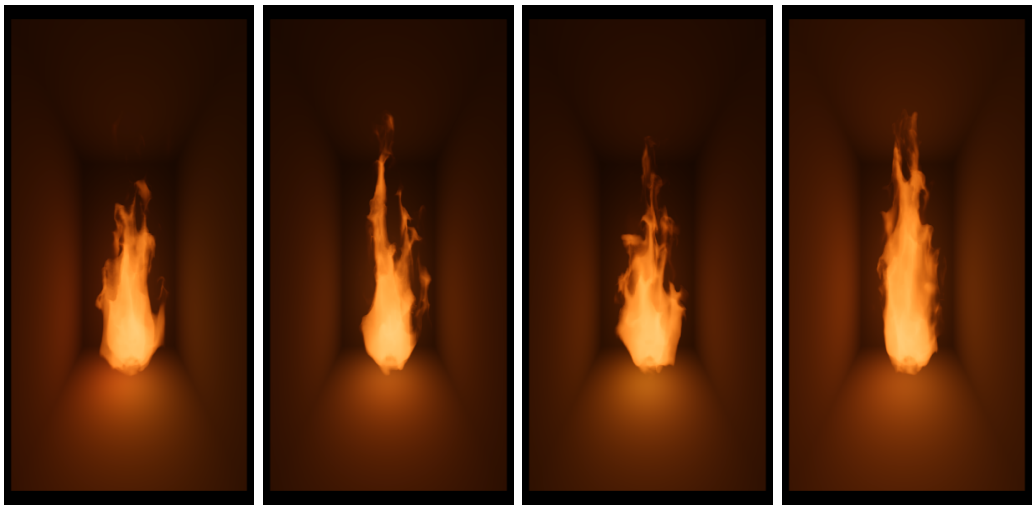

Figure 4: *Four images from our flame simulation and rendering (frame 90, 109, 224 and 454). Notice the intensity and position change of the reflected light at the walls and floor. Also notice that since we only use one average point as light source the radiance is hardly visible at the roof. Simulation grid size (30x60x30), temperature grid size (120x240x120), image size (300x600 pixels), flame speed* $S = 0.1$*, all wavelengths.*

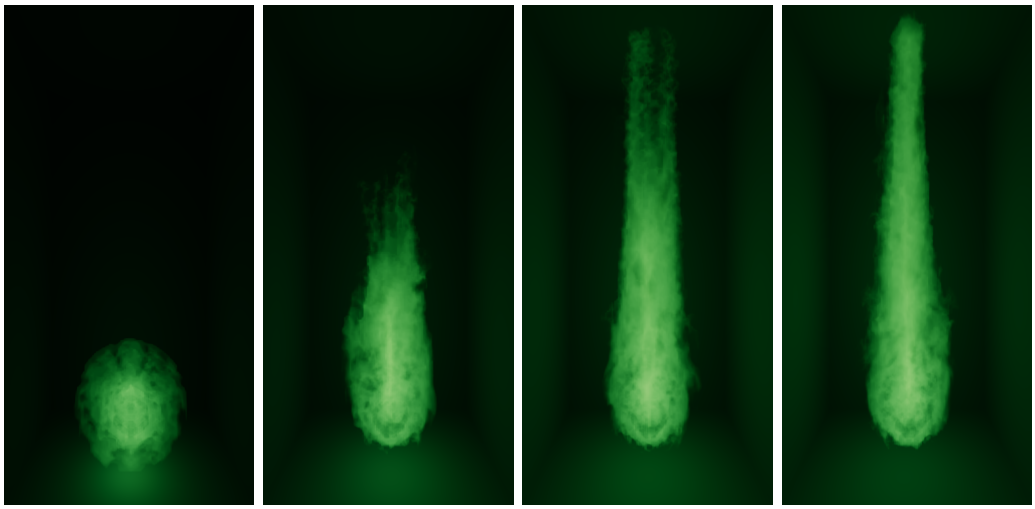

Figure 5: *Four images from our flame simulation and rendering (frame 7, 43, 89 and 214). Notice the intensity and position change of the reflected light at the walls and floor. Also notice that since we only use one average point as light source the radiance is hardly visible at the roof Simulation grid size (60x120x60), temperature grid size (120x240x120), image size (500x100 pixels), flame speed* $S = 0.2$*, only the green wavelenghts.*

References

[1] Cie starndard observer. 1931. `http://www.cis.rit.edu/mcsl/online/cie.php`.

[2] Blackbody. 2013. `http://www.exo.net/~pauld/workshops/Stars/Stars.htm`.

[3] Ct scan. 2013. `http://en.wikipedia.org/wiki/Volume_rendering`.

[4] Lms color space, cat02. 2013. `http://en.wikipedia.org/wiki/LMS_color_space`.

[5] Openmp. 2013. `http://openmp.org/wp/`.

[6] srgb. 2013. http://en.wikipedia.org/wiki/SRGB.

[7] D. Q. Nguyen, R. Fedkiw, and H. W. Jensen. Physically based modeling and animation of fire, 2002.