Realistic fire rendering

Garoe Dorta-Perez

CM50170 - Research project

Unit Leader: Dr Peter Hall

University of Bath

August 2015

Contents

Chapter 1

Introduction

"*A case that can be made for fire being, next to the life process, the most complex of phenomena to understand.*" - Hoyt Hottle

(a) Paper fire[1]. (b) Bone fire[2].

Figure 1-1: Examples of real fires.

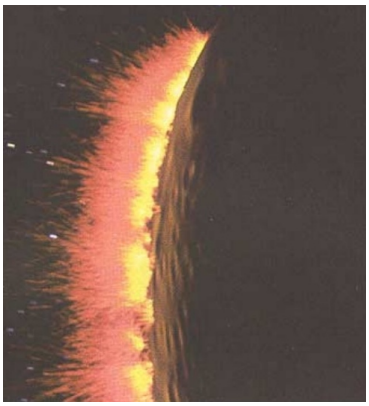
Computer simulations are widely used to model a diverse range of natural phenomena. Their popularity lies in their inexpensive nature, predictive capabilities given good mathematical models, parameters can be changed at will to produce new results, the equations governing the system can be modified to create scenarios that are not physically attainable, etc. Computer generated graphics are used to display the results of the simulations. Rendering the data on the screen involves transforming a scene into an image that can be displayed using some light transport and illumination model.

For centuries humans has been fascinated with fire due to its attractive presence and its dangerous nature,

[1] https://en.wikipedia.org/wiki/File:Fire.JPG
[2] https://en.wikipedia.org/wiki/File:Large_bonfire.jpg

real fire examples are shown in Figure 1-1. Combustion phenomena are prevalent in daily life, candles, camp fires, explosions, car engines, cooking appliances, etc. Simulating and visualizing fire related processes has many applications, for example it is widely used for visual effects in the film industry, simulated as part of the virtual environment in the computer games industry; or in the engineering community, where modelling engine combustion and fire safety evaluations are frequently demanded.

Computer generated examples in films include, a planet explosion in Star Trek II, as shown in Figure 1-2a, where a particle-based technique by [Reeves, 1983] was used; Shrek featured a dragon exhaling fire, as shown in Figure 1-2b, where parametric curves were used to drive the flames [Lamorlette and Foster, 2002]; or the more recent work of [Horvath and Geiger, 2009] based on 2D screen projections for the film Harry Potter and the Deathly Hallows, as shown in Figure1-2c. In these and in many other applications, using real flames is an expensive and hazardous endeavour.



(a) Explosion in Star Trek II [Reeves, 1983].

(b) Dragon fire in Shrek [Lamorlette and Foster, 2002].

(c) Flames used in Harry Potter and the Deathly Hallows [Horvath and Geiger, 2009].

Figure 1-2: Examples of visual effects with fire in the film industry.

The computer graphics community has intensively researched the fluid behaviour of water and smoke. Fire can also be modelled as a fluid, however due to its multiphase flow, chemical reactions and radiative

heat transport, the techniques used for water or smoke cannot be directly applied to flames. As a result of the aforementioned complexity and the interdisciplinary nature of the problem, fire simulation is still an open problem in computer graphics.

A great deal of work done in the area has sacrificed complexity for interactiveness, therefore producing simplified models which hope to deceive the observer by exploiting the chaotic behaviour present in fire motion. Nevertheless, physically-based simulations incorporate the intrinsic processes that occur in a combustion scenario in order to be able to produce realistic results. In general the following effects are taken into consideration.

Flame motion: under normal gravity conditions, convection, external wind forces, front propagation and gas expansion drive the shape and position of the flame.

Fuel erosion: when the fuel reaches a certain temperature, it is vaporized into a gaseous state, which rises under the influence of buoyancy.

Flame colour: the chemical species present in the fuel and the byproducts of the combustions emit energy in various wavelengths due to black body radiation.

In this report a short review of fire simulation and rendering methods is presented in Chapter 2. In Chapter 3, a state-of-the-art physically-based model to render fire, proposed by [Pegoraro and Parker, 2006], is discussed extensively. Implementation details of a Mental Ray® shader for the the given model are outlined in Chapter 2, and results are discussed in Chapter 5. Lastly, conclusions and future research directions are considered in Chapter 6.

Chapter 2

Previous Work

In order to display a realistic fire scene in a computer generated world, two procedures need to be performed. Firstly, the fire dynamics have to be collected, this can either be done through a data capture session or simulated using fluid-based solvers. Present techniques in the area are discussed below. Secondly, the gathered data will be visualized on the screen using some rendering method, a brief state-of-the-art review of fire rendering procedures is presented in Section 2.2. We refer the interested readers to the more detailed survey on the topic, which has been recently presented by [Huang et al., 2014].

Fire simulation has received significantly more attention from the computer graphics community than fire rendering. As the output data of the simulation methods is used in the rendering stage, an outline of the different simulation techniques is given below. Still, the main focus of this report lies in the rendering phase of the combustion simulation pipeline.

2.1 Simulation

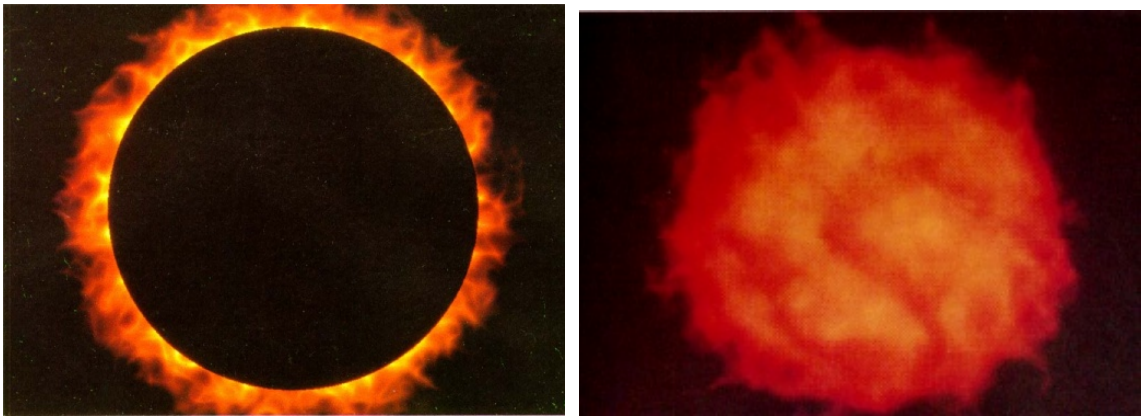
The techniques presented in this section are divided adopting the same criteria that was used by [Huang et al., 2014]. Since the categories are roughly chronological, the evolution of the field is easier to follow.

Particle-based methods were the first approach to simulate the visual animation of fire, as shown in Figures 1-2a and 2-1. A number of particles are emitted from certain locations, each particle has a set of attributes such as shape, velocity, color or lifetime. The first model with particle systems was presented by [Reeves, 1983], the particles speed and colour were perturbed with a Gaussian noise at each time step, and the colour was subject to an additional linear perturbation on its lifetime. Two particle systems were used in a hierarchy, one would control fire spread and the other a single explosion effect. An extension was proposed by [Perry and Picard, 1994]; the authors modified the particle system such that each particle shape would be defined by a series of non-overlapping coplanar triangles. The transparency of each triangle increases towards the outer vertices, thus providing an improved visual effect.

Noise-based methods focus on synthesizing the high fluctuation present in fire procedurally, as shown in Figure 2-2. The objective is to approximate the turbulence present in fire with an appropriate statistical

Figure 2-1: Particle-based fire [Perry and Picard, 1994].

model. Using a variation of Perlin noise, [Perlin, 1985] presented images of a corona of flames. The method is limited to 2D, where the color is a combination of non-linear arbitrary functions. This work was extended by [Perlin and Hoffert, 1989] to 3D, where they used volumetric rendering to achieve improved results. However, both methods have difficulties to handle interactions with other objects, and the variations in the synthesized images makes them incompatible for animation production.



(a) [Perlin, 1985]. (b) [Perlin and Hoffert, 1989].

Figure 2-2: Noise-based fire examples.

Geometry skeleton are driven by rendering primitives that represent the main features of fire, the high frequency details are built on top of the geometry structure, as shown in Figures 1-2b and 2-3. [Lee et al., 2001] presented a technique to animate fire on meshes, the flame front propagation is based on a geodesic flow field on the surface. Although they achieved the effects of merging and separation of multiple flame fronts, the animators have no control over the evolution of the fire, as the only parameters are the initial fire conditions. [Lamorlette and Foster, 2002] proposed a B-Spline curves interpolation method, details are added from a library of images which are then cast onto the profile of the flame. The authors' technique supports a range of flame behaviour, such as spreading, flickering, and flame separation and merging. In order to achieve real time simulations, [Bridault-Louchez et al., 2006] modelled the flame shape on NURBS surfaces on which a transparent 2D texture were mapped.

(a) [Lee et al., 2001]. (b) [Bridault-Louchez et al., 2006].

Figure 2-3: Geometry skeleton fire examples.

Data driven methods are used to simulate fire with data from real flames, as shown in Figure 2-4. The quality of the animation depends directly on the data, and by avoiding the simulation stage the computational overhead is reduced significantly, as solving the underlying heat transport and other chemical equations is no longer needed. However, there are severe limitations when producing new animations and interactions with other objects. [Rushmeier et al., 1995] captured a heptane pool fire with water-cooled nitrogen-purged probes. Using two images from orthographic camera views, [Hasinoff and Kutulakos, 2003] reconstructed 3D volumetric fire using tomographic techniques. [Ihrke and Magnor, 2004] extended the work by [Hasinoff and Kutulakos, 2003], the authors presented a method to reconstruct fire volumes using trilinear basis functions in scenes recorded with eight cameras. [Zhang et al., 2011] presented a method to synthesize new animation sequences based on previously simulated data. A flow graph whose nodes connect resembling simulation states is built, similarity between nodes is measured based on flow pathline distances. New frames with smooth transitions are created by traversing the graph through different paths. [Okabe et al., 2015] proposed a reconstruction technique using a set of sparse images, that can also be applied to generate novel animations. The authors approach is to define an optimization problem in terms of colour histograms and steerable coefficients, which they can use to retrieve an approximation of the appearance of the volume at new viewing angles.

Physically-based methods simulate the fire combustion processes, including flame propagation or the chemical reactions that convert fuel into gaseous products, as shown in Figures 1-2c and 2-5. Incompressible flow equations were used by [Stam and Fiume, 1995] to drive a fire simulation. Given initial fuel conditions, the fire spread is advected on a grid using an advection-diffusion type equation. Building on a semi-Lagrangian fluid solver of [Stam, 1999], a model which includes gaseous fuel and gaseous byproducts was proposed by [Nguyen et al., 2002]. In order to include the characteristics of the noise-based methods, [Hong et al., 2007] combined the previous model with a set of third-order equations from detonation shock dynamics presented by [Yao and Stewart, 1996]. As with the noise-based methods, this addition is visually attractive, yet it is not physically based. Capitalizing on the recent advances in GPUs parallel processing power, [Horvath and Geiger, 2009] proposed a fixed camera model. Particle properties are computed on a three-dimensional coarse grid, which are then projected into several view-dependant two-dimensional slices. The authors' model is based on the assumption that fine variations, which are perpendicular to the projection plane, are not individually visible and, they do not significantly affect the overall flow.

(a) [Hasinoff and Kutulakos, 2003]. (b) [Ihrke and Magnor, 2004]. (c) [Zhang et al., 2011]. (d) [Okabe et al., 2015].

Figure 2-4: Date driven fire examples.

(a) [Stam and Fiume, 1995]. (b) [Nguyen et al., 2002]. (c) [Hong et al., 2007].

Figure 2-5: Physically-based fire examples.

Other effects directly related to fire have also been explored, as shown in Figure 2-6. [Feldman et al., 2003] presented a model to simulate suspended particles during explosions. An incompressible fluid model drives the motion of air and hot gases, and the suspended particles follow their movements. [Melek and Keyser, 2005] simulated the decomposition of an object as it undergoes combustion. Heat is transferred between the different parts of the object and the air; the decomposition is modelled as a moving boundary spreading in the distance field of the object, where level set methods are used to compute the moving interfaces. Credible sound is an important factor to increase the believability of a finished fire animation. [Chadwick and James, 2011] proposed a method to automatically generate plausible noise given for a given fire simulation. Low frequency sound is estimated using a physical model whose two given inputs are the flame front and heat release. A data driven sound synthesis approach, based on the work by [Wei and Levoy, 2000], is applied to generate the high frequency content.

(a) [Feldman et al., 2003]. (b) [Melek and Keyser, 2005]. (c) [Chadwick and James, 2011].

Figure 2-6: Other effects with fire.

2.2 Rendering

Rendering fire is more challenging than rendering other types of participating media, the main reason being that fire is a light-emitting source. The luminescence radiated by a flame is generated by black body radiation due to the high temperature of the particles during the combustion process. In order to realistically render fire, light absorption and scattering in the media, including air, has to be taken into consideration. A succinct review of the current methods used by the computer graphics community for fire rendering is presented below.

2.2.1 Raster-Based Methods

Raster-based techniques sacrifice quality in the interest of interactive frame rates, renderings are shown in Figure 2-7. Generally, some form of texture mapping is applied, only the surface of the flame is considered, and the illumination of the scene is approximated using parameters which can be easily computed such as the fire height.

[Reeves, 1983] applied a linear colour assignment to each particle in their simulation based on their lifetime. [Lee et al., 2001] applied the same technique to render fires on mesh surfaces, where a particle would begin with a light yellow colour, evolve to red and finish in black at the end of their lifetime. [Lamorlette and Foster, 2002] presented a technique were a flame picture is mapped onto the two-dimensional flame profile with a base colour. For each particle an intermediate emitting value is computed, and the final colour profile is super-sampled from an approximation of the cross-sectional area of the flame, as it would appear from the camera. [Westover, 1990] introduced texture splats as a new rendering primitive, [Wei et al., 2002] rendered fire using Westover's technique. Direct illumination is approximated by rendering the fire particles separately to create a light image, which is texture projected into the surrounding objects. [Zhao et al., 2003] also proposed a method with flame particle splats, which are added with splats from other objects and sorted based on their distance to the screen plane. The splats are projected in front-to-back order and alpha-blended to produce the end result. [Ihrke and Magnor, 2004] rendered fire realistically using a technique based on tomography reconstructions of real flames. Precomputed ray-marching data is stored in a matrix, that is later used for fast reconstruction of new view points. [Bridault-Louchez et al., 2006] used

a spectrophotometer to capture photometric distributions of candles. The intensities are stored on a texture and changes in illumination over time are introduced with an attenuation factor which is proportional to the size of the flame. [Zhang et al., 2011] proposed a method were fire particles and their attributes are first projected onto a set of slicing planes, which are orthogonal to the camera direction. The planes are then blended in back-to-front order, and a one-dimensional colour texture is used as a transfer function to convert flow attributes to colours and opacities. [Jamriška et al., 2015] recently proposed a texture synthesis method to generate fire animations. The method requires a hand made motion field and an alpha mask of the desired result, both are used to generate the new sequence using data from an existing video exemplar.

(a) [Wei et al., 2002]. (b) [Zhao et al., 2003]. (c) [Jamriška et al., 2015].

Figure 2-7: Raster-based techniques.

2.2.2 Ray-Tracing-Based Methods

Volume ray-tracing techniques offer astonishing results, however the associated computational costs are considerable. Rays are shot from the view plane and evaluated at small increments; the total radiance at the origin of the ray is computed by integrating the radiance at each step. A further drawback of ray tracing techniques, in comparison to raster-based methods, is the lack of a standard ray-tracing pipeline.

[Rushmeier et al., 1995] presented a method to perform accurate ray casting on sparse measured data. The fire was modelled as a series of stacked cylindrical rings, where each ring has uniform properties. The total radiance at each point is integrated using a Monte Carlo method, summing up the measured irradiances at sample locations. [Nguyen et al., 2002] proposed a ray marching technique to solve the Radiative Transport Equation, see Chapter 3. The emitted light is computed using Planck's formula of black body radiation, scattering in the media and visual adaptation to the fire spectra are modelled. [Feldman et al., 2003] also included black body radiation in their animation of fire with suspended particles, however the mapping to RGB was manually adjusted to match the images of real explosions. Direct illumination shadows were computed using deep shadow maps [Lokovic and Veach, 2000], while scattering and illumination by other objects in the scene used the technique proposed by [Jensen and Buhler, 2002]. An extension to [Nguyen et al., 2002] was presented by [Pegoraro and Parker, 2006], the authors' model has physically-based absorption, emission and scattering properties. The spectroscopy characteristics of different fuels are achieved by modelling the transitions of electrons between different states in the molecules. Non-linear trajectories of light in the medium due to light refraction effects are included as well. In order to minimize

the effects induced by the limitations of the RGB colour space, the visual adaptation process is presented as a post-processing effect. [Horvath and Geiger, 2009] proposed a rendering method whose main objective was user-friendliness for artists. Using the fixed camera slices described in Section 2.1, the authors perform simple volume rendering to join them in a single image. Black body radiation is used for the light, the images are motion-blurred with a filter based on the velocities in the slices, and the heat distortion is added as post-processing filter defined by the user.

Chapter 3

Methodology

Given simulated or captured fire data, that could come from any of the methods discussed in Section 2.1, we want to generate a photo-realistic image from a camera view of the scene. From this point onwards we will assume that we have a volumetric data structure, which holds the relevant input data for the fire. An overview of the pipeline based on the work by [Pegoraro and Parker, 2006] is shown in Figure 3-1. The inputs are volumetric data of density and temperature, from those values a series of intermediate constants for each voxel for the current frame are computed, and with the previous data a ray marcher is used to generate the final image.

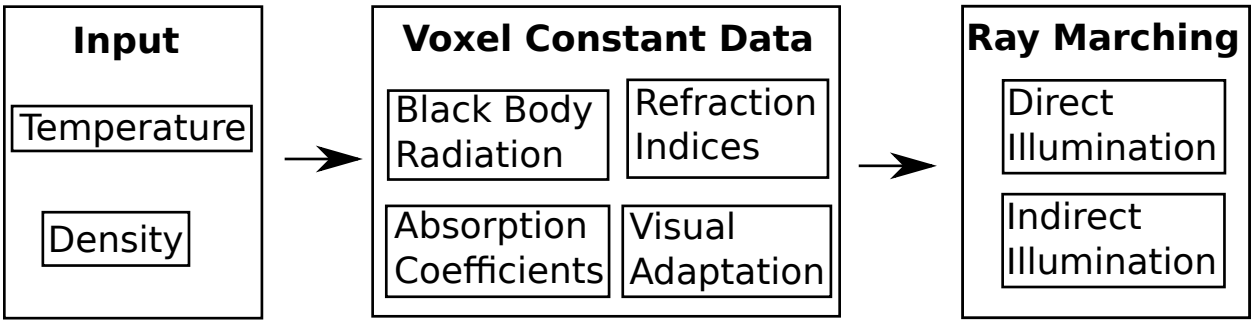


Figure 3-1: An overview of the fire rendering pipeline.

3.1 Fundamental Concepts

In this section an overview of the fundamentals concepts of volume rendering is presented, readers familiar with the area may proceed to Section 3.2. As the main objective is to give a intuitive introduction into the area, the steps and concepts presented here will not be formally derived. A simple framework to generate images when light interacts with a participating media will be described.

The world is modelled as scene were there are a number of elements, the camera, fire volumes, lights and other objects. The **camera** is defined as a point and orientation in space, with a projection matrix. A

voxel is defined as a cube in space with some data associated with it. A **voxel dataset** is a collection on voxels in space, typically a cubic section of the scene which is divided uniformly into voxels. There could also be other **lights sources** in the scene, several types are supported, point, area, directional and cone lights. **Other objects** may also be part of the scene, such as polygonal or nurbs meshes.

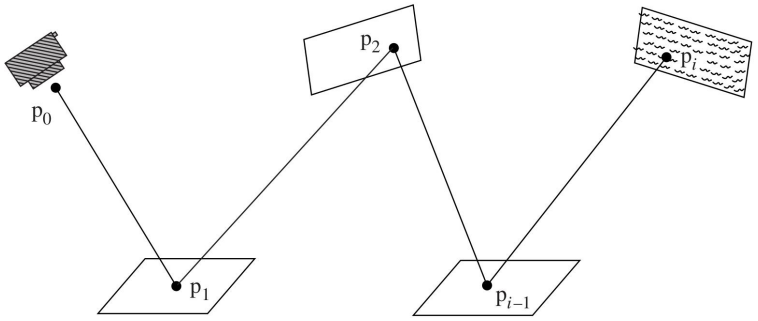


Figure 3-2: Computing the colour of p_1 using path tracing [Pharr and Humphreys, 2010].

In order to generate an image from the camera view, path tracing is used to compute the colour of each pixel, as shown in Figure 3-2. For each pixel in the camera view, a **ray** is shot into the scene; whose direction is defined by the camera origin, orientation and the pixel position. When the ray hits a primitive in the scene, it will be reflected and the colour will depend on the contribution of further intersections along the path. The final point in the path will be directly connected to a light in the scene, in Figure 3-2, p_{i-1} is the final intersection and p_i is the light position.

3.2 Radiative Transport Equation

The Radiative Transport Equation (RTE) [Howell and Siegel, 2002] is the model chosen to describe the appearance of any flame in Pegoraro and Parker's method. The RTE models the variation of spectral radiance $(\omega\nabla)L(\lambda, \mathbf{x}, \boldsymbol{\omega})$ in the medium, where λ is a given wavelength in metres, $\mathbf{x}$ is the point of interest in space, and $\boldsymbol{\omega}$ is a vector that points towards the viewing direction. Formally, the objective is to get a solution for the RTE for each visible point in the scene, practically, an approximate solution will be computed.

The integro-differential Radiative Transport Equation is defined as

$$\begin{aligned}(\omega\nabla)L(\lambda, \mathbf{x}, \boldsymbol{\omega}) = &- \sigma_a(\lambda, \mathbf{x})L(\lambda, \mathbf{x}, \boldsymbol{\omega}) + \sigma_a(\lambda, \mathbf{x})L_e(\lambda, \mathbf{x}, \boldsymbol{\omega}) \\ &- \sigma_s(\lambda, \mathbf{x})L(\lambda, \mathbf{x}, \boldsymbol{\omega}) + \sigma_s(\lambda, \mathbf{x})L_i(\lambda, \mathbf{x}, \boldsymbol{\omega}),\end{aligned} \tag{3.1}$$

where L_i is defined by

$$L_i(\lambda, \mathbf{x}, \boldsymbol{\omega}) = \int_{4\pi} L(\lambda, \mathbf{x}, \boldsymbol{\omega}_i)\Phi(\lambda, \boldsymbol{\omega}, \boldsymbol{\omega}_i)d\boldsymbol{\omega}_i, \tag{3.2}$$

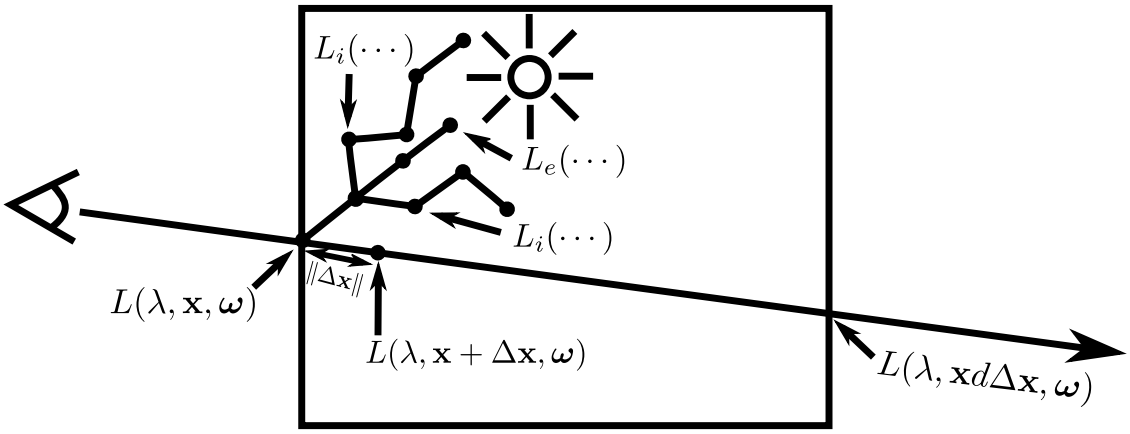


Figure 3-3: A visual representation of the RTE, where d is the number of $\Delta\mathbf{x}$ between the ray entrance point to the ray exit point.

where σ_a is an absorption coefficient, σ_s is a scattering coefficient, L_e is the emitted spectral radiance at the point, L_i is the in-scattering radiance, Φ is a scattering phase function and ω_i is a scattering sampling direction. In order to get an analytical solution to the aforementioned equation, the properties of the medium are assumed to be homogeneous over a small segment $\|\Delta\mathbf{x}\|$ in space,

$$L(\lambda, \mathbf{x} + \Delta\mathbf{x}, \boldsymbol{\omega}) = e^{-\sigma_t(\lambda,\mathbf{x})\|\Delta\mathbf{x}\|} L(\lambda, \mathbf{x}, \boldsymbol{\omega}) + \left(1 - e^{-\sigma_t(\lambda,\mathbf{x})\|\Delta\mathbf{x}\|}\right) \frac{\sigma_a(\lambda, \mathbf{x}) L_e(\lambda, \mathbf{x}, \boldsymbol{\omega}) + \sigma_s(\lambda, \mathbf{x}) L_i(\lambda, \mathbf{x}, \boldsymbol{\omega})}{\sigma_t(\lambda, \mathbf{x})}, \tag{3.3}$$

where $\sigma_t = \sigma_a + \sigma_s$ is the extinction coefficient, a visual depiction of this solution is shown in Figure 3-3. To be able to calculate the terms in Equations 3.2 and 3.3, we will explain in more detail the following factors:

- Light **scattering** in the media $L_i(\lambda, \mathbf{x}, \boldsymbol{\omega})$ with a phase function $\Phi(\lambda, \boldsymbol{\omega}, \boldsymbol{\omega}_i)$.
- Emitted light $L_e(\lambda, \mathbf{x}, \boldsymbol{\omega})$ due to **black body radiation**.
- Electromagnetic **absorption**, σ_a, of certain wavelengths.
- Due to spatially varying **refractive** indices, photons may follow non linear paths $\Delta\mathbf{x}$.
- **Visual adaptation** processes in the human eye affect the perceived final radiance $L(\lambda, \mathbf{x}, \boldsymbol{\omega})$.

3.3 Scattering

Some photons travelling along arbitrary directions $\boldsymbol{\omega}_i$, might be scattered in the direction of interest $\boldsymbol{\omega}$. A simple diagram of the process is shown in Figure 3-4, where two photons travelling in directions $\boldsymbol{\omega}_1$ and

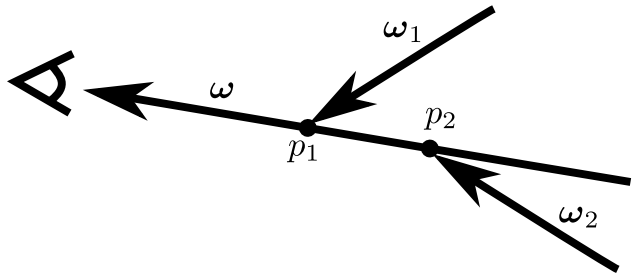


Figure 3-4: Photons scattering in the direction of interest.

ω_2, are scattered in the direction of interest ω, at the points p_1 and p_2 respectively. The number of photons which are subjected to this process is controlled by a coefficient σ_s, the radiance carried by the photons is denoted by $L_i(\lambda, \mathbf{x}, \omega)$ and follows a spherical distribution $\Phi(\lambda, \omega, \omega_i)$, which was first proposed by [Henyey and Greenstein, 1941] to model diffuse radiation in astronomy,

$$\Phi(\lambda, \omega, \omega_i) = \frac{1 - g(\lambda)^2}{4\pi \left(1 + g(\lambda)^2 - 2g(\lambda)\omega\omega_i\right)^{\frac{3}{2}}}, \tag{3.4}$$

where g is an asymmetry coefficient which can be a function of the wavelength, although in most cases it is chosen to be constant. The value of g must be in the range $(-1, 1)$, where $g < 0$ corresponds to backwards scattering, $g = 0$ to isotropic scattering, and $g > 0$ to forwards scattering. Monte Carlo techniques can be used to sample the light scattering directions ω_i.

3.4 Black Body Radiation

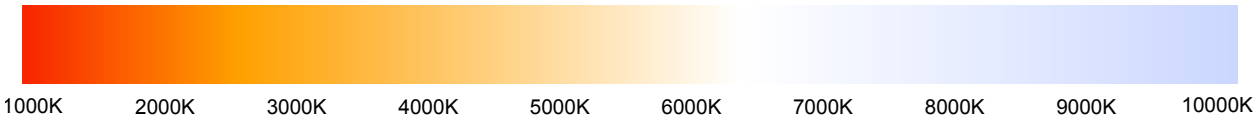


Figure 3-5: Black body emission colours normalised to ignore brightness[1].

Emitted light from fire, which will illuminate the flame itself and other objects in the scene, is denoted as $L_e(\lambda, \mathbf{x}, \omega)$. Radiation from an idealized physical body that absorbs all incident electromagnetic radiation can be modelled with Planck's equation for black body radiation, as shown in Figure 3-5. This equation characterizes the electromagnetic radiation $B(T, \lambda, \eta)$ emitted by a black body in thermal equilibrium at a given temperature T.

$$L_e(\lambda, \mathbf{x}, \omega) = B(T, \lambda, \eta) = \frac{2hc^2}{\lambda^5(e^{\frac{hc}{\lambda kT}} - 1)}, \tag{3.5}$$

[1] https://en.wikipedia.org/wiki/File:Blackbody-colours-vertical.svg

where $h = 6.62606957 \times 10^{-34}\ J/s$ is Planck's constant, $k = 1.3806488 \times 10^{-23}\ J/K$ is Boltzmann constant, η is the refraction index of the medium and $c = c_0/\eta$ is the speed of light in the current medium, where $c_0 = 299792458\ m/s$ is the speed of light in a vacuum. The refraction index η varies across the medium, the procedure to compute it will be described in Section 3.6.

3.5 Absorption coefficients

As fire is not an idealized black body radiator, certain wavelengths are not present in the emitted radiance. This behaviour can be modelled using absorption coefficients for each chemical that would mask the undesired wavelengths, such that

$$\sigma_a L_e(\lambda, \mathbf{x}, \boldsymbol{\omega}) = \sum_{i=0}^{N} \sigma_{ai} B_i(T, \lambda, \eta), \tag{3.6}$$

where N is the number of different chemical species in the mix, σ_{ai} and $B_i(T, \lambda, \eta)$ are the absorption coefficient and light emission of the i^{th} chemical. In the interest of simplicity, the absorption and radiation effects for **soot** will be treated separately from those caused by other **chemical species**.

3.5.1 Soot Absorption

Light generated by incandescent soot accounts for most of the total radiance emitted by a flame. The spectral absorption coefficient of soot is defined as

$$\sigma_a(\lambda, \mathbf{x}) = \frac{48 N(\mathbf{x}) \pi R^3 nm}{\lambda^{\alpha(\lambda)}((n^2 - m^2 + 2)^2 + 4n^2m^2)}, \tag{3.7}$$

where $N(\mathbf{x})$ is the number density, density per unit volume, R is the radius of a soot particle, n, m and $\alpha(\lambda) = 1.39$ are optical constants for different types of soot. In Table 3.1, values for the optical constants n and nm are provided for several materials, the data was obtained from [Dalzell and Sarofim, 1969]. Note that the Table provides values for nm instead of the obvious m, Dalzell and Sarofim chose to supply the data in such way in their paper. The radius of soot particles was determined in the range $R \in \{50\ \text{Å} \ldots 800\ \text{Å}\}$ by [Dalzell and Sarofim, 1969]. Since our data is defined with soot densities, we have chosen the radius to be the mean value $R = 425 \times 10^{-10}$ metres.

Table 3.1: Absorption constants for propane and acetylene, wavelengths are in nanometres.

		Wavelengths			
		435.8	450	550	650
Propane	n	1.57	1.56	1.57	1.56
	nm	0.46	0.5	0.53	0.52
Acetylene	n	1.56	1.56	1.56	1.57
	nm	0.46	0.48	0.46	0.44

3.5.2 Absorption From Other Chemical Species

Flames of more uncommon colours are generally produced when using salts of different chemicals. The spectral line emitted by an particular atom depends of the electronic transitions between different energy levels. We will only treat spontaneous emission in thermodynamic equilibrium, caused by an electron in an upper energy stage going to a lower energy state, releasing electromagnetic energy in the process. The absorption coefficients associated for a given spectral frequency and a known atomic element can be computed as

$$\sigma_a(\lambda, \mathbf{x}) = \frac{\phi(\lambda) N_2 A_{21} \lambda^4 (e^{\frac{hc}{\lambda kT}} - 1)}{8\pi c}, \tag{3.8}$$

where $\phi(\lambda)$ is the normalized spectral line, N_2 is the number density in the upper state (number of particles per unit volume), A_{21} is an Einstein coefficient measuring the transition probabilities of spontaneous emission from the upper state. The number of particles in the upper energy level N_2 is given by the Maxwell-Boltzmann distribution, which is defined as

$$N_2 = \frac{N g_2 e^{-\frac{E_2}{kT}}}{g_1 e^{-\frac{E_1}{kT}}} = \frac{g_2}{g_1} N e^{\frac{E_1 - E_2}{kT}}, \tag{3.9}$$

where N is the total number density (number of particles per unit volume), g_1 and g_2 are degeneracy numbers for the lower and upper states, E_1 and E_2 are the energy levels in nm^{-1} for the lower and upper states, k is the the Boltzmann constant and T is the temperature.

3.6 Refraction

Heat emanating from the flame is transferred to the air which surrounds it. Since the refractive index of a medium depends on its temperature, light transiting the space near the flame will follow non linear paths due to varying refractive indices. Fresnel equations describe in detail both reflection and refraction effects, as shown in Figure 3-6, when a wave moves between media with different refractive indices. Schlick equations [Schlick, 1994] are widely used by the computer graphics community to approximate the Fresnel coefficients.

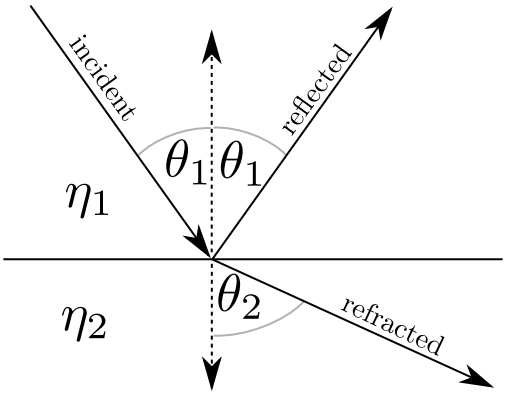


Figure 3-6: Ray being refracted and reflected in the boundary between two media.

The fraction of light which is reflected is assumed to be negligible for combustion phenomena; thus, the refraction angles for the rays can be easily computed using Snell's law

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{\eta_2}{\eta_1}, \tag{3.10}$$

where θ_1 is the incident angle, θ_2 is the refracted angle, η_1 is the index of refraction of the media which the ray is coming from and η_2 is the index of refraction of the media which the ray is going to. The values for all the constants in this section are defined in Table 3.2.

[Ciddor, 1996] proposed a method to compute the refractive indices of air

$$\eta_{air} = 1 + \frac{\rho_a \eta_{axs}}{\rho_{axs}} + \frac{\rho_w \eta_{ws}}{\rho_{ws}}, \tag{3.11}$$

where ρ_{axs} is the density of dry air, ρ_{ws} is the density of pure water vapour, ρ_a and ρ_w are the equivalent quantities for dry air, η_{axs} is the refractive index for CO_2 and η_{ws} is the refractive index for water vapour. The refractive index for CO_2 is estimated as

$$\eta_{axs} = \eta_{as}\left(1 + 0.534 \times 10^{-6}\,(x_c - 450)\right), \tag{3.12}$$

$$\eta_{as} = 10^{-8}\left(\frac{k_1}{k_0 - \lambda} + \frac{k_3}{k_2 - \lambda}\right), \tag{3.13}$$

where $x_c \in \{300 \ldots 450\}$ is the proportion in parts per million of CO_2, k_0, k_1 k_2 and k_3 are constants, and λ is the wave number (reciprocal of the vacuum wavelength). The refractive index for water vapour is computed as

$$\eta_{ws} = 1.022 \times 10^{-8}\left(w_0 + w_1\lambda + w_1\lambda^2 + w_3\lambda^3\right), \tag{3.14}$$

where w_0, w_1, w_2 and w_3 are constants.

The ρ densities are computed as follows

$$\rho = \frac{pm_a}{zrT}\left(1 - x_w\left(1 - \frac{m_w}{m_a}\right)\right), \tag{3.15}$$

where p is the pressure in pascals, r and m_w are constants,

$$m_a = 0.0289635 + 12.011 \times 10^{-8}(x_c - 400)), \tag{3.16}$$

$$z = 1 - \frac{p}{T}\left(a_0 + a_1 t + a_2 t^2 + (b_0 + b_1 t)\, x_w + (c_0 + c_1 t)\, x_w^2\right) + \left(\frac{p}{T}\right)^2\left(d + e x_w^2\right), \tag{3.17}$$

where a_0, a_1, a_2, b_0, b_1, c_0, c_1, d and e are constants.

$$T = t + 273.15, \tag{3.18}$$

$$x_w = \frac{fhv}{p}, \tag{3.19}$$

$$f = \alpha + \beta p + \gamma t^2, \tag{3.20}$$

where α, β and γ are constants, and $h \in \{0 \ldots 1\}$ is the air relative fractional humidity, 0 for dry air and given by the user for moist air.

The pressure p can also be computed using the ideal gas law

$$pV = \frac{NRT}{V} = nT, \tag{3.21}$$

where N is the number of molecules, V is the total volume of the gas, T is the temperature of the gas, n is the number density (number of particles per unit volume), $R = kN_a$, where k is the Boltzmann constant and N_a is Avogadro constant.

Depending on the situation, there are two acceptable methods to compute the vapour pressure v, both are described below.

3.6.1 Davis' Saturation Vapour Pressure

In [Ciddor, 1996] paper, the author used the method proposed by [Davis, 1992] to compute the saturation of vapour pressure

$$v = e^{AT^2+BT+C+D/T}, \tag{3.22}$$

where A, B, C and D are constants. This equation is relatively easy to compute, however incorrect results will be obtained for temperatures below 0 °C. As we are concerned with fire rendering, which entails high temperatures, this method is preferred due to its simplicity.

3.6.2 Huang's Saturation Vapour Pressure

The International Association for the Properties of Water and Steam (IAPWS) adopted an alternative technique to [Davis, 1992], which was proposed by [Huang, 1998]. The more recent method addresses the drawbacks of the previous technique, giving reasonable results for temperatures below 0 °C. Nevertheless, the generality comes with increased complexity in the equations

$$v = 10^6 \left(\frac{2C}{X}\right)^4, \tag{3.23}$$

$$X = -B + \left(B^2 - 4AC\right)^{\frac{1}{2}}, \tag{3.24}$$

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 1 & K_1 & K_2 \\ K_3 & K_4 & K_5 \\ K_6 & K_7 & K_8 \end{bmatrix} \begin{bmatrix} \Omega^2 \\ \Omega \\ 1 \end{bmatrix}, \tag{3.25}$$

$$\Omega = T + \frac{K_9}{T - K_{10}}, \tag{3.26}$$

where K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8, K_9, and K_{10} are constants.

3.6.3 Ciddor's Method Summary

The inputs of Ciddor's technique are a wavelength λ_0, a temperature t, a pressure p, a relative humidity value h and a CO_2 concentration x_c. The values for all the constants needed for the method are defined in Table 3.2. A refraction index is computed following this method for each voxel in the volumetric data structure. The steps to compute the index of refraction are listed below

1. Let $z_a = 0.9995922115$, computed using Equation 3.17 with $t = 20$ °C, $p = 101.325$ Pa and $x_w = 0$.
2. Calculate $\lambda = 1/\lambda_0^2$, and T with Equation 3.18.
3. Calculate v with the method described in Section 3.6.1 or in Section 3.6.2.
4. Calculate x_w using Equations 3.20 and 3.19.
5. Calculate η_{as} using Equation 3.13.

6. Calculate η_{ws} using Equation 3.14.
7. Calculate m_a using Equation 3.16.
8. Calculate η_{axs} using Equation 3.12.
9. Calculate z_m using Equation 3.17.
10. Calculate $\rho_{axs} = (p_0 m_a)/(z_a r T_0)$, where $p_0 = 101325$, and $T_0 = 288.15$, using Equation 3.15.
11. Calculate $\rho_w = (x_w p m_w)/(z_m r T)$ using Equation 3.15.
12. Calculate $\rho_a = ((1 - x_w) p m_a)/(z_m r T)$ using Equation 3.15.
13. Calculate the index of refraction η_{air} using Equation 3.11.

Table 3.2: Values for the constants in the equations used to compute the index of refraction.

Name	Value
ρ_{ws}	0.00985938
x_c	450 ppm
k_0	238.0185 μm^{-2}
k_1	5792105 μm^{-2}
k_2	57.362 μm^{-2}
k_3	167917 μm^{-2}
w_0	295.235 μm^{-2}
w_1	2.6422 μm^{-2}
w_2	−0.03238 μm^{-2}
w_3	0.004028 μm^{-2}
p	1 atm = 101325 Pa
r	8.31451 $Jmol^{-1}K^{-1}$
m_w	0.018015 kg/mol
N_a	$6.02214129 \times 10^{23}$ mol^{-1}
A	$1.2378847 \times 10^{-5} K^{-2}$
B	$-1.9121316 \times 10^{-2} K^{-1}$
C	33.93711047
D	$-6.3431645 \times 10^{3} K$
K_1	$1.16705214528 \times 10^{3}$
K_2	$-7.24213167032 \times 10^{5}$
K_3	-1.70738469401×10

Name	Value
a_0	$1.58123 \times 10^{-6} KPa^{-1}$
a_1	$-2.9331 \times 10^{-8} Pa^{-1}$
a_2	$1.1043 \times 10^{-10} Pa^{-1}$
b_0	$5.707 \times 10^{-6} KPa^{-1}$
b_1	$-2.051 \times 10^{-8} Pa^{-1}$
c_0	$1.9898 \times 10^{-4} KPa^{-1}$
c_1	$-2.376 \times 10^{-6} KPa^{-1}$
d	$1.83 \times 10^{-11} K^2 Pa^{-2}$
e	$-0.765 \times 10^{-8} K^2 Pa^{-2}$
α	1.00062
β	$3.14 \times 10^{-8} Pa^{-1}$
γ	5.6×10^{-7} °C^{-2}
k	$1.3806488 \times 10^{-23}$ J/K
K_4	$1.20208247025 \times 10^{4}$
K_5	$-3.23255503223 \times 10^{6}$
K_6	1.49151086135×10
K_7	$-4.82326573616 \times 10^{3}$
K_8	$4.05113405421 \times 10^{5}$
K_9	$-2.38555575678 \times 10^{-1}$
K_{10}	$6.50175348448 \times 10^{2}$

3.7 Visual Adaptation

The human eye presents a non-linear response to incident radiance L, as shown in Figure 3-7. If a flame is the dominant light source in a scene, it will appear with yellow-white colours, while the same flame in an environment with comparable or brighter lights might appear orange-yellow.

Figure 3-7: Fire variations in colour due to eye adaptation mechanisms [Pegoraro and Parker, 2006].

A simple equation which can model this phenomena was proposed by [Naka and Rushton, 1966]

$$R(L, \tau) = \frac{L}{L + \tau}, \tag{3.27}$$

where τ is a non-linear adaptation state. This state is determined by the visual system to maximize the perception of features for a given scene. Several approaches have been proposed to determine which features are used and how to compute the optimal state. We refer the readers to [Fairchild, 2005] for an in depth review of visual adaptation methods. The Von Kries model [Fairchild, 2005], the new values for the LMS cone responses for a given stimulus are simply multiplied by the inverse of the maximum response in the scene. The Retinex model [Fairchild, 2005], extends Von Kries' work by normalizing the the ratio of the stimulus with the average response in a retinex space. The retinex space is defined as a three dimensional space with the spectral responses of the cones. [Irawan et al., 2005] presented neural network-based model which includes time varying adaptations, and variations due to intrinsic observer characteristics, such as their age or visual impairment, are also considered.

Chapter 4

Implementation details

In this chapter we will explain how to use the shaders which have been implemented and the differences between the theoretical pipeline explained in Chapter 3 and the actual software implementation. An overview of the pipeline is shown in Figure 4-1, note that the modules in grey are not present in the software.

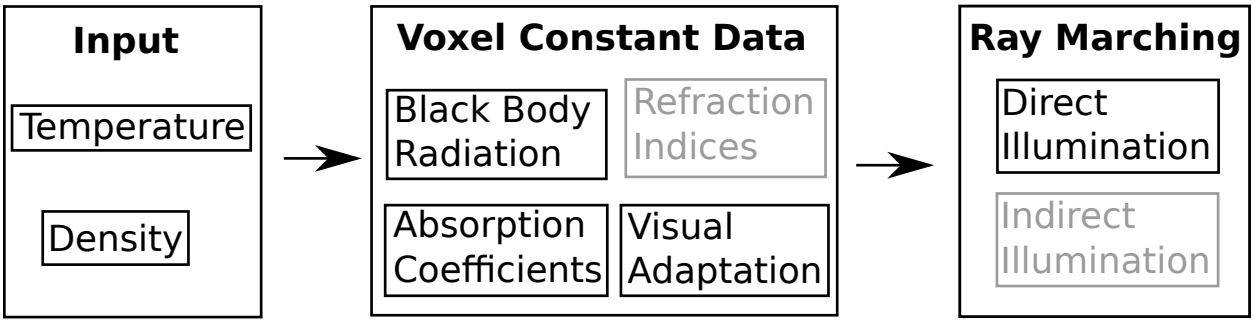
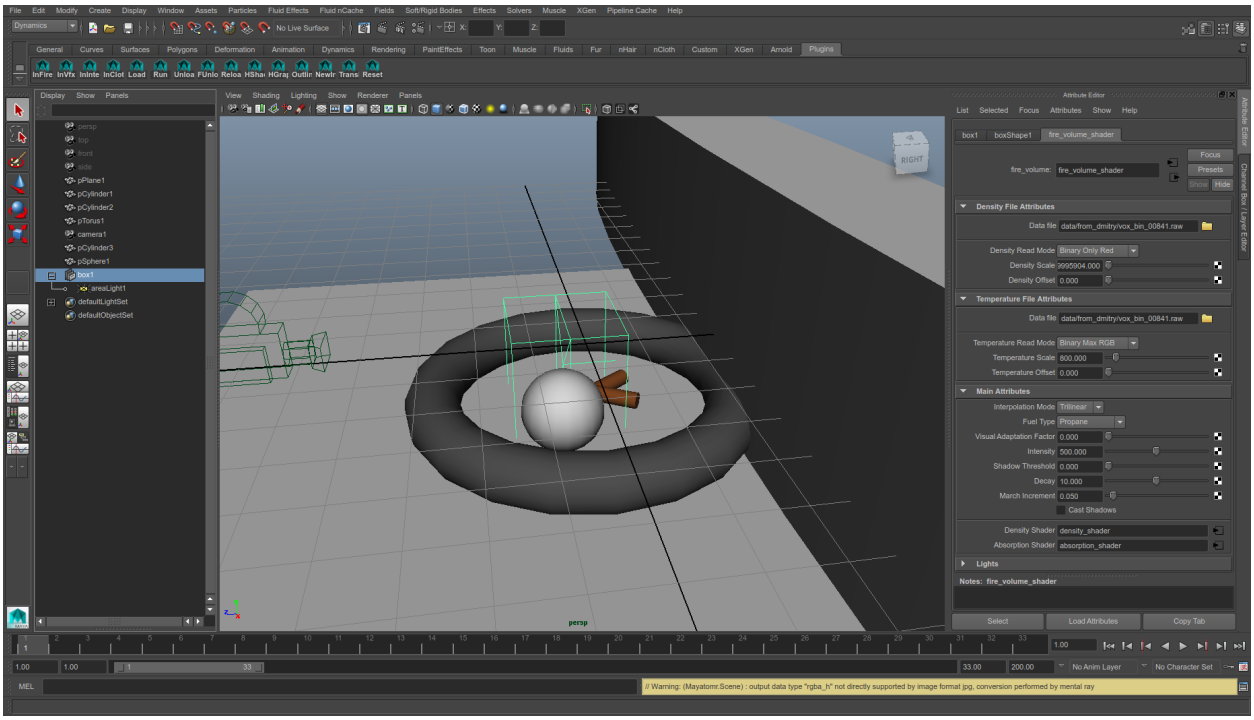


Figure 4-1: An overview of the implemented fire rendering pipeline.

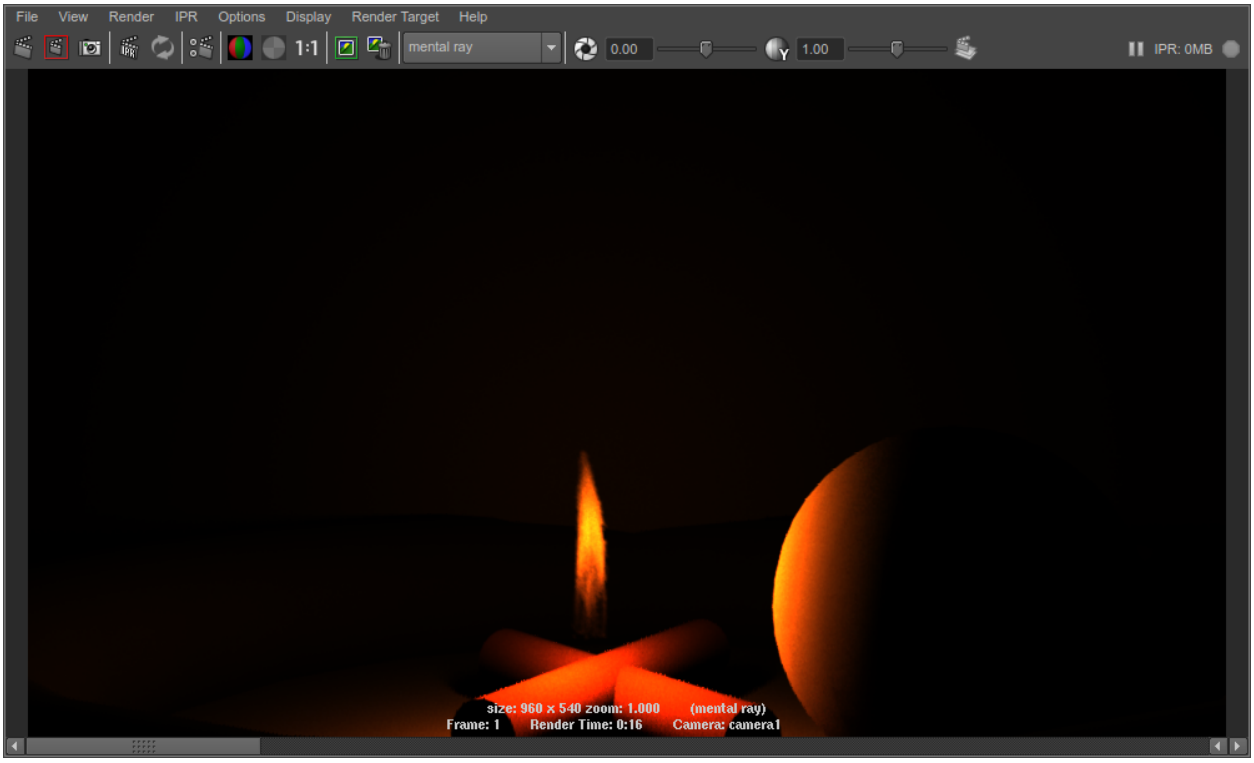
The implementation of the model was realised with several shaders for Mental Ray® to be used in Maya®. Using a rendering engine like Mental Ray®, which provides the basic functionality needed for ray tracing, means that the code is implemented using a predefined abstraction layer, thus reducing bugs in the code and increasing the programmer's productivity. The integration with Maya® is desirable, as the software gives the user an intuitive environment in which to create complex scenes, that are compatible with off-the-shelf effects like cloth or fluid simulations.

4.1 Application Overview

Figure 4-2a shows the Maya® interface for our shaders in a test scene. From the user view point the volumetric data is enclosed in a cube, that can be manipulated like any other object in the scene. The GUI for the shader includes two input files and sliders for the numeric parameters. The scene rendered from the camera viewpoint is shown in Figure 4-2b.



(a) Test scene, note on the right the shader interface.



(b) Low quality rendered image from the scene above.

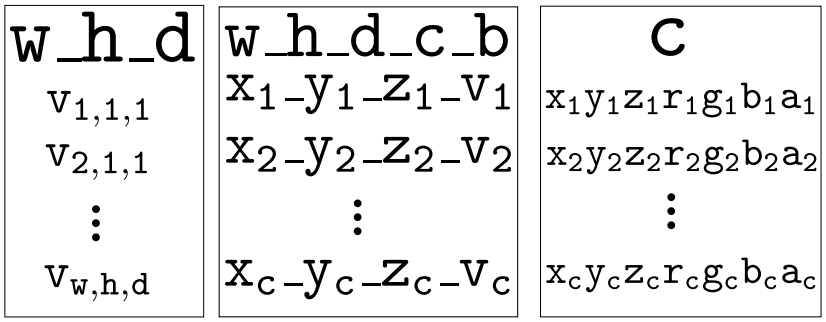
Figure 4-2: Application interface in Maya®.

Although the focus of the project is to render any input data as realistic as possible, as brief overview of the system used to record the flames gives a better idea of the full pipeline, from an initial capture session to a final render. Our input data was captured with six mvBlueCOUGAR-XD104 cameras, as shown in Figure 4-3. We experimented during several capture sessions with different illumination conditions, sensor gains, flame types, and other parameters. The captured data was reconstructed and edited using techniques that are outside of the scope of this report.

(a) Overview. (b) Close up.

Figure 4-3: Our capture system.

The model used for our input data is that of a three-dimensional voxel dataset, a cube in space is uniformly divided and a value is stored for each voxel. This data in the voxel is either a soot density estimate for the *fire volume shader* or a temperature estimate for the *fire light shader*. Three input file formats for single-precision floating-point data are supported, dense or sparse values in plain ASCII files, and sparse RGBA values in binary files, as shown in Figure 4-4. In the ASCII dense format, the first line of the file contains three integers (separated by spaces) declaring the width, height and depth of the voxel volume. This is followed by $w \times h \times d$ lines with the values for each voxel. The ASCII sparse format, has two extra parameters in the first line, c is the number of data points in the file and b is the default value for any voxel that is not specified in the file. The data values in this case are preceded by the x, y, z coordinates separated by spaces. The RGBA binary sparse format starts with a 4 bytes integer declaring how many data points are in the file, the x, y, z coordinates are each stored as 4 bytes integers, followed by an RGBA colour where each channel is encoded as double-precision floating-point value with 8 bytes. Any voxel that is not included in the file is considered to be initialized to zero.



(a) ASCII dense (*.vol). (b) ASCII sparse (*.uintah). (c) Binary sparse (*.raw).

Figure 4-4: Suported file formats, _ represents the character space.

4.2 Mental Ray® Rendering Approach

Mental Ray® approach to solving the rendering equation is based on path tracing, as shown in Figure 4-5, for each pixel in the camera view, an eye ray will be shot into the scene. On an intersection with an object in the scene, its material shader will be called, this shader will shoot a light ray for each light in the scene, which in effect calls the light shader of the given light. In order to compute the irradiance at the intersection point, the light shader will trace a shadow ray from the light to the intersection point. Eventually, the material shader will compute the final colour with the information received from the light shader.

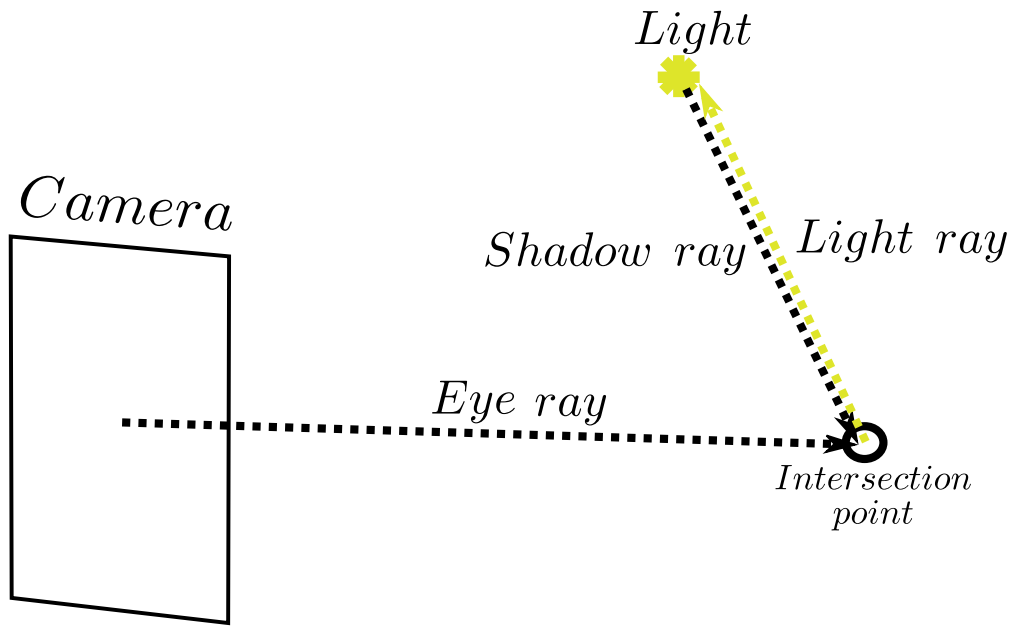


Figure 4-5: Mental Ray® simple ray casting example.

Under this assumptions, the steps required to solve the RTE, Equation 3.3, are:

1. Shoot a ray from the eye into the scene.
2. If the ray intersects the fire volume, perform ray marching in the volume.
   (a) Compute direct illumination at the current point.
      i. Sample all the lights $\rightarrow L_e$
      ii. Attenuate each with its own absorption coefficient $\rightarrow (1 - e^{-\sigma_a}) L_e$
   (b) Compute indirect illumination at the current point $\rightarrow \sigma_s L_i$
      i. Use phase function to get the scattered light distribution $\rightarrow \Phi$
   (c) Add current contribution to accumulated color $\rightarrow e^{-\sigma_t} L + (1 - e^{-\sigma_a}) L_e$
   (d) Advance to the next point in ray marching $\rightarrow \mathbf{x} = \mathbf{x} + \Delta\mathbf{x}$
      i. Change ray direction using refraction index $\rightarrow \Delta\mathbf{x} = refract(\Delta\mathbf{x})$

4.3 Simplifications

Equation 3.3 provides a radiance value $L(\lambda, \mathbf{x} + \Delta\mathbf{x}, \omega)$ for the next march increment, however it is more intuitive to think about the final radiance $L(\lambda, \mathbf{x}, \omega)$ for the first intersection point. The radiance at the interest point is given by

$$L(\lambda, \mathbf{x}, \omega) = e^{-\sigma_t(\lambda,\mathbf{x})\|\Delta\mathbf{x}\|} L(\lambda, \mathbf{x} + \Delta\mathbf{x}, \omega) + \left(1 - e^{-\sigma_t(\lambda,\mathbf{x})\|\Delta\mathbf{x}\|}\right) \frac{\sigma_a(\lambda, \mathbf{x}) L_e(\lambda, \mathbf{x}, \omega) + \sigma_s(\lambda, \mathbf{x}) L_i(\lambda, \mathbf{x}, \omega)}{\sigma_t(\lambda, \mathbf{x})}. \tag{4.1}$$

In fire phenomena the effect of scattered light in the final image is practically imperceptible [Pegoraro and Parker, 2006]. Intuitively, it means that as the medium is highly emissive and thin, most of the emitted photons have their initial paths unaltered. The scattering directions ω_i are in essence sampling directions over a sphere of scattered light L_i. In practice, the simplification will significantly decrease rendering time, as ω_i would have to be sampled by means of a computationally expensive Monte Carlo approximation. Exploiting that prior knowledge, the scattering contributions can be safely ignored by setting $\sigma_s = 0$, which simplifies the previous equation to

$$L(\lambda, \mathbf{x}, \omega) = e^{-\sigma_a(\lambda,\mathbf{x})\|\Delta\mathbf{x}\|} L(\lambda, x + \Delta\mathbf{x}, \omega) + \left(1 - e^{-\sigma_a(\lambda,\mathbf{x})\|\Delta\mathbf{x}\|}\right) L_e(\lambda, \mathbf{x}, \omega). \tag{4.2}$$

A visual representation of the implications of the aforementioned simplification is shown in Figure 4-6. Note that a single point light is used in the diagram to avoid clutter, yet in our implementation there is a point light located at the centre of each voxel whose temperature is high enough to be emissive. The non-linear trajectories of the photons, of the rays in practice, due to varying refraction indices in the media were also outside of the scope of this project. Deformations produced by the refraction phenomena can be noticeable, however there are important simplifications in the implementation if we choose to ignore them. Namely, we

can reduce the recursiveness significantly, instead of starting in the first intersection and recursively solving the equation, we can start from the end point and accumulate the result backwards with a loop, as computing the end point in this situation is trivial.

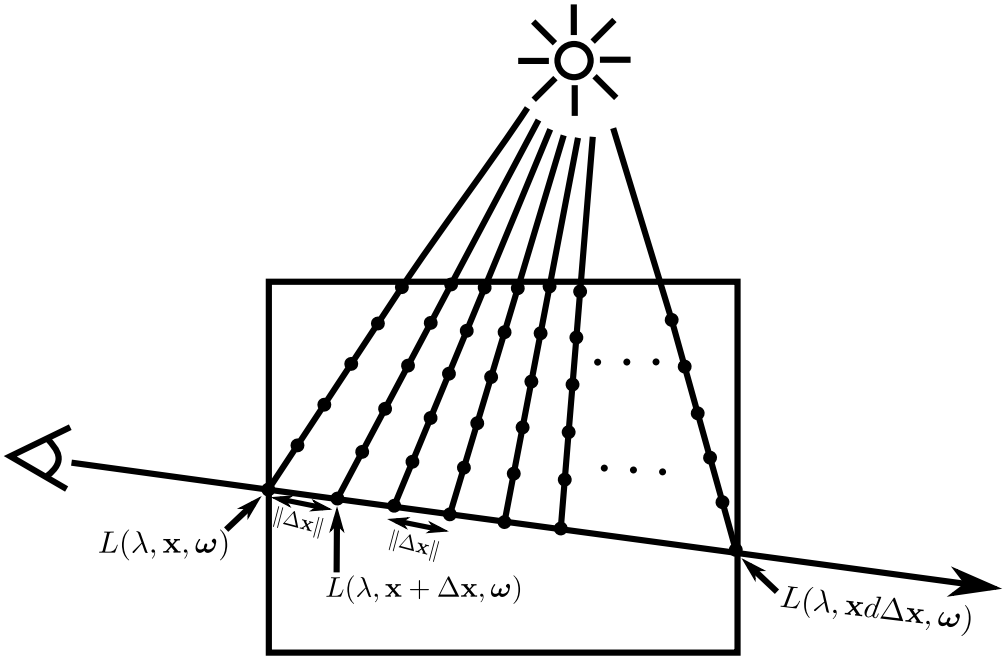


Figure 4-6: Calculating the illumination values for samples along a ray.

4.4 Shaders Overview

Given the complexity of the task at hand, a modular architecture for the system was chosen. As depicted in Figure 4-7, the main shader (Fire Volume) delegates the computation of several values to other auxiliary shaders. The temperature and density shaders read the raw input values, the emission and absorption shaders compute black body radiation and absorption coefficients, respectively. The fire light shader manages light rays by precomputing and sampling the emitted light L_e. The fire volume shader is in charge of including the absorption coefficient with shadow rays, and to perform the main ray marching computations, eye rays.

4.5 Spectrum to RGB

To speed up the render time, the precomputed spectral values are converted to RGB coefficients before ray marching. Our spectrum class has a number of samples that can be set at compile time, 30 samples were used in all of our renderings. To compute the absorption coefficients that were introduced in Section 3.5, data from Table 3.1 is used. As there is not enough data to compute the 30 spectrum coefficients, a linear interpolation is used to fill in the missing values. RGB coefficients are the standard colour representation used by computer screens, however not every colour visible by the human eye can be expressed in RGB

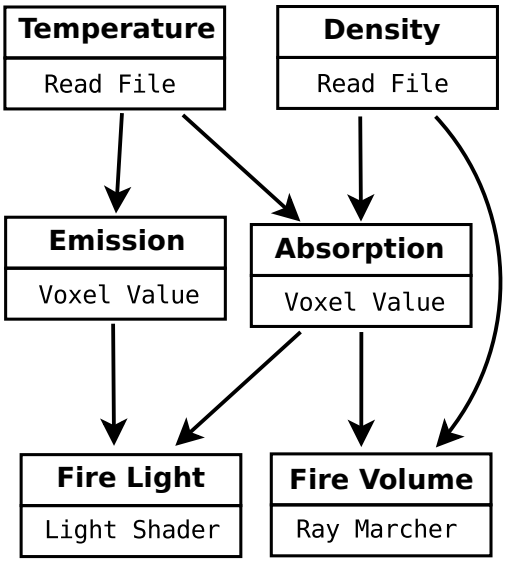


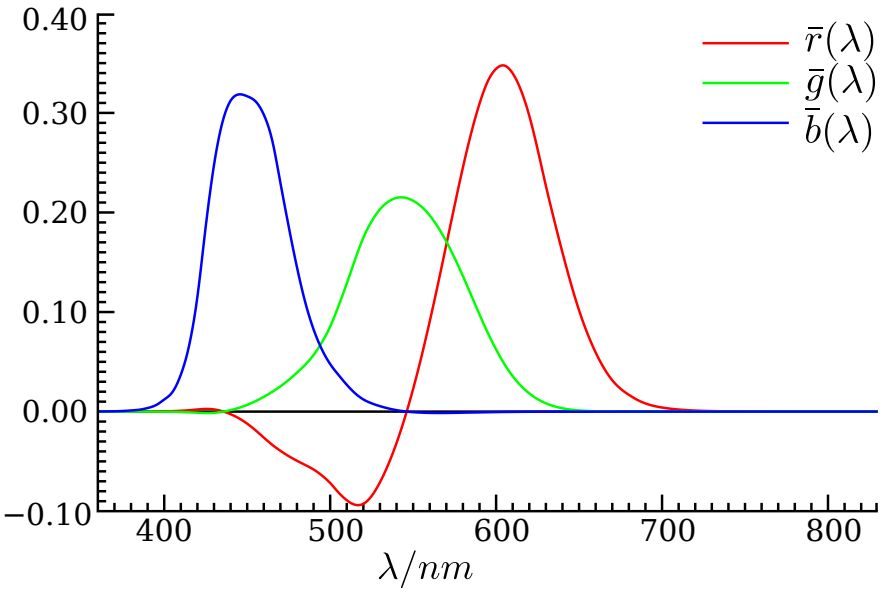
Figure 4-7: Diagram depicting the shaders and their interconnections.

space. There are further disadvantages inherent to this representation, for example certain colours visible by the human eye would require negative values for the $\bar{r}(\lambda)$ coefficient, as shown in Figure 4-8a. The XYZ colour space is built using three sets of imaginary primaries, which have a series of favourable properties. Any colour which can be seen by the human eye can be represented with some $x(\lambda)$, $y(\lambda)$, $z(\lambda)$ coefficients, the Y channel is equal to the photopic luminous efficiency function $V(\lambda)$, which models the variation of perceived brightness, and as shown in Figure 4-8b, the coefficients are always positive.

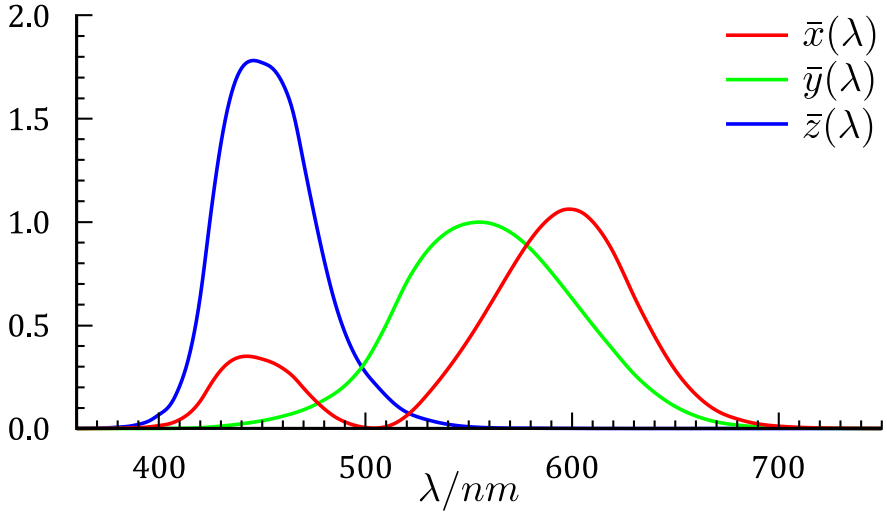
Given a Spectral Power Distribution (SPD) $S(\lambda)$, the x, y, z coefficients for $S(\lambda)$ in the XYZ space are defined with respect to spectral matching curves $X(\lambda)$, $Y(\lambda)$ and $Z(\lambda)$, as depicted in Equation 4.3.

$$\begin{aligned} x &= \frac{1}{\int Y(\lambda)d\lambda} \int_{\lambda} S(\lambda)X(\lambda)d\lambda, \\ y &= \frac{1}{\int Y(\lambda)d\lambda} \int_{\lambda} S(\lambda)Y(\lambda)d\lambda, \\ z &= \frac{1}{\int Y(\lambda)d\lambda} \int_{\lambda} S(\lambda)Z(\lambda)d\lambda. \end{aligned} \tag{4.3}$$

The term $1/\int Y(\lambda)d\lambda$ is added in each equation to act as a normalization factor for the colour brightness. As we are concerned with sampled values on a discrete domain, the integration for XYZ coefficients is approximated by a Riemann sum



(a) Spectral tristimulus values for CIE RGB[1].



(b) Spectral tristimulus values for CIE XYZ[2].

Figure 4-8: Colour matching curves for RGB and XYZ spaces.

[1] https://commons.wikimedia.org/wiki/File:CIE1931_RGBCMF.svg#/media/File:CIE1931_RGBCMF.svg

[2] https://commons.wikimedia.org/wiki/File:CIE_1931_XYZ_Color_Matching_Functions.svg#/media/File:CIE_1931_XYZ_Color_Matching_Functions.svg

$$
\begin{aligned}
x &\approx \frac{1}{\int Y(\lambda)d\lambda} \sum_i X_i c_i, \\
y &\approx \frac{1}{\int Y(\lambda)d\lambda} \sum_i Y_i c_i, \\
z &\approx \frac{1}{\int Y(\lambda)d\lambda} \sum_i Z_i c_i,
\end{aligned}
\tag{4.4}
$$

where c_i is the i^{th} coefficient in the SPD, X_i is the area for the corresponding sample range of the $X(\lambda)$ curve and equivalently for Y_i and Z_i. Note that $1/\left(\int Y(\lambda)d\lambda\right)$, X_i, Y_i and Z_i are constants and they can be precomputed for efficiency. For the Riemann sum, the area under the samples is approximated using piecewise linear interpolation. The conversion from XYZ to RGB space is performed using

$$
\begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} \int R(\lambda)X(\lambda)d\lambda & \int R(\lambda)Y(\lambda)d\lambda & \int R(\lambda)Z(\lambda)d\lambda \\ \int G(\lambda)X(\lambda)d\lambda & \int G(\lambda)Y(\lambda)d\lambda & \int G(\lambda)Z(\lambda)d\lambda \\ \int B(\lambda)X(\lambda)d\lambda & \int B(\lambda)Y(\lambda)d\lambda & \int B(\lambda)Z(\lambda)d\lambda \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix},
\tag{4.5}
$$

where $R(\lambda)$, $G(\lambda)$, $B(\lambda)$ are the spectral curves for the red, green and blue colours respectively. Note that all the factors in the transformation matrix are constants and they can be precomputed for efficiency.

4.5.1 Performance considerations

Volumetric data tends to be quite sparse, and usually we are working with volumes of $256 \times 256 \times 256$ voxels, which assuming a 32 bits floating point value per voxel in the density and temperature shader, and three floating point values for the emission and absorption shaders, accumulates respectively to 64 and 192 megabytes of data. With the shader architecture shown in Figure 4-7, there will be at least two copies of each structure per fire in the scene. Since a total of 512 megabytes per voxel dataset is unacceptable, we use an open source sparse voxel dataset library [OpenVDB, 2015]. The sparse data structure reduced the total memory consumption to approximately 20 megabytes per dataset, while increasing the total rendering time by less than 10 seconds on average.

In order to achieve smoother rendering results, a trilinear interpolation is computed in each step in the ray-marching algorithm. Computing the interpolation smoothed the shape of the flames and reduced the effects of outliers in the input data. Tricubic interpolation was also considered, however it was discarded due to the significant increase in computational overhead.

Our system follows the recommendations of the Mental Ray® API, which allows performance to scale automatically as the number of hardware processors increases. Batch rendering for animations is also supported, the shaders will automatically choose the correct input files for each frame.

4.6 Visual Adaptation

The implementation of the eye visual adaptation to the colours in fire, described in Equation 3.27 is performed as described in [Nguyen et al., 2002], which uses the Von Kries model [Fairchild, 2005]

$$\begin{aligned}
\mathbf{x}_a &= \mathbf{M}^{-1}\mathbf{T}_w\mathbf{M}\mathbf{x}_i, \\
\mathbf{l}_{max} &= \mathbf{M}\mathbf{x}_{max}, \\
\mathbf{T}_w &= \begin{bmatrix} 1/l_{max} & 0 & 0 \\ 0 & 1/m_{max} & 0 \\ 0 & 0 & 1/s_{max} \end{bmatrix}, \\
\mathbf{M} &= \begin{bmatrix} 0.4002 & 0.7076 & -0.0808 \\ -0.2263 & 1.1653 & 0.0457 \\ 0 & 0 & 0.9182 \end{bmatrix},
\end{aligned} \tag{4.6}$$

where $\mathbf{x}_a = [x_a, y_a, z_a]^T$ is a column vector with the XYZ adapted coefficients, $\mathbf{T}_w$ is the Von Kries transformation, x_{max} are the coefficients for the maximum temperature present in the fire, $\mathbf{l}_{max} = [l_{max}, m_{max}, s_{max}]^T$ are the LMS coefficients of x_{max}, $\mathbf{M}$ is a XYZ to LMS transformation matrix and, $\mathbf{x}_i = [x_i, y_i, z_i]^T$ is a column vector with the XYZ input coefficients. The $\mathbf{M}$ matrix is a Hunt-Pointer-Estevez transformation, proposed by [Hunt and Pointer, 1985], which has been normalized to the CIE Standard Illuminant D65. The coefficients for $\mathbf{M}^{-1}$ are not provided in the CIE specification. Given that $\mathbf{M}$ is a small 3×3 matrix, the coefficients of $\mathbf{M}^{-1}$ can be evaluated with any of the standard matrix inversion methods, in our case the default algorithm in Matlab® *inv()* function was used, and the output was hard-coded in the shader. The aforementioned equation is solved in the XYZ space, in order to delay clamping coefficients to RGB as long as possible.

4.7 Miscellaneous

4.7.1 Utilities

Several auxiliary scripts have been developed in order to aid and provide some automation while using the software. A bash render script "render.sh" is provided, this script batch renders a given Maya® scene and creates a video with the output images.

New fuel types can be added to extend the default selection using spectral lines from the NIST database [NIST, 2015]. A Matlab® script "save_nist_data.m" can be used to download and save new ".specline" files for new atoms. Once a new file has been added to the data folder, the files "fire_shader.mi" and "FuelTypes.h" have to be updated to include the name of the new data.

Uintah [Uintah, 2015] is the combustion simulator used by [Pegoraro and Parker, 2006] as input for their rendering method. Detailed instructions for its installation on Ubuntu, how to run examples, as well as scripts to convert from their output data format to a format compatible with the shaders is provided with the source code.

4.7.2 Known bugs and workarounds

Batch rendering through the Maya® GUI does not automatically update the input files for the shaders. A workaround is to use the command line render utility as shown below

Listing 4.1: Batch render command

```
$ Render -r mr -perframe -s <start_frame_number> -e <end_frame_number> <path_to_scene_file>
```

Saving images, either from batch render or single frame GUI render, will produce unexpected results, normally a completely white output, if the image format natively supports transparency, such as tif, tga or exr. In order to get consistent results, the scene should be surrounded by solid objects.

Any Maya® scene which uses the shaders should be saved in Maya® ASCII format, "<scene>.ma". The saved files are not platform independent, as the input file names for the shaders use absolute paths. If the scene is to be opened in a new environment, the file paths have to be modified to match the current locations.

Chapter 5

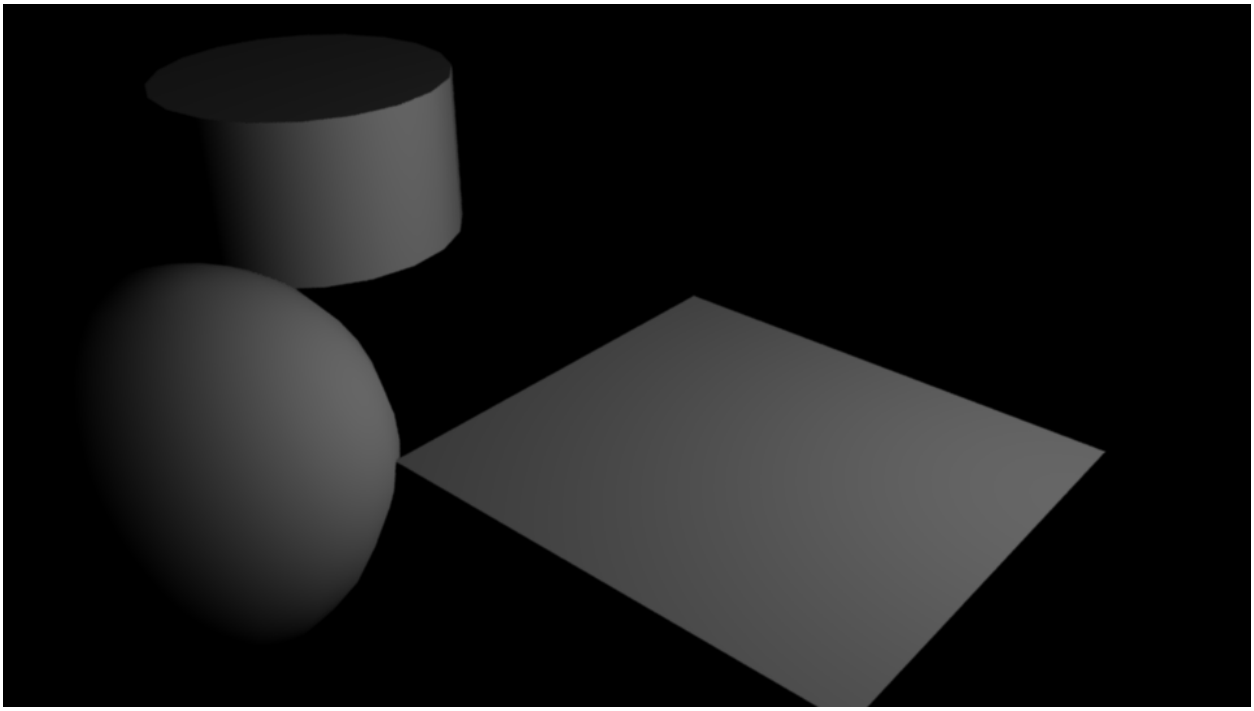
Results

In this chapter we will show several tests that were executed using our shaders under different conditions. In order to provide a wider perspective, one that is not only limited to academic results, we will also compare our results with images generated using industry software. FumeFX [FumeFX, 2015] is a popular tool for modelling volumetric effects, it has been used in several films and video games, e.g. Ghost Rider 2, Thor, Green Lantern, Warhammer Online and Tomb Raider: Underworld. A frame from a fire scene in Ghost Rider 2 is shown in Figure 5-1.

Figure 5-1: Fire created with FumeFX [FumeFX, 2015].

Since Maya® was used as the base platform for the development of our software, an interesting com-

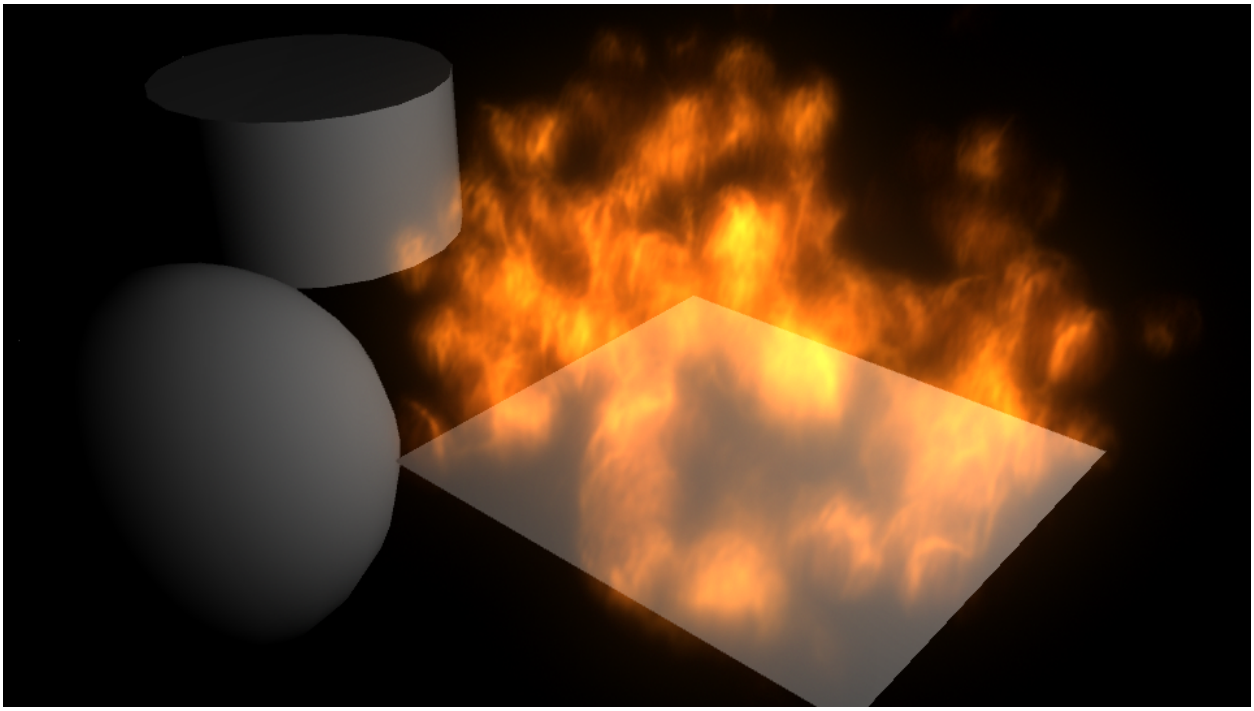
parison should include Maya®'s fire effect. A simple scene with plane, a sphere, a cylinder and a light was created, as shown in Figure 5-2a. For the majority of the results that will be presented in the rest of the chapter, the scene shown in Figure 5-3 was used. The scene was chosen because it has a few external objects for the flame to interact with, yet it is simple enough so that multiple tests can be executed within reasonable render times. The result of rendering the scene with Maya®'s software renderer is shown in Figure 5-2b, note how the sphere and cylinder do not experience any change in their illumination with respect to the scene without the fire. If Mental Ray® is used to render the same scene, the image shown in Figure 5-2c is generated. Although the fire itself looks more realistic, the only alteration in neighbouring objects is the appearance of erroneous shadows. Placing on the flame examples provided by Maya® in our test scene produced the result shown in Figure 5-4. More specifically, the 14th frame of the simulation in the file fire fluid examples "Flame.ma" was used. Note that a point light had to be placed in the centre of the flame, and the emitted colour had to be matched manually to the fire colour, to be able to produce the desired effect.



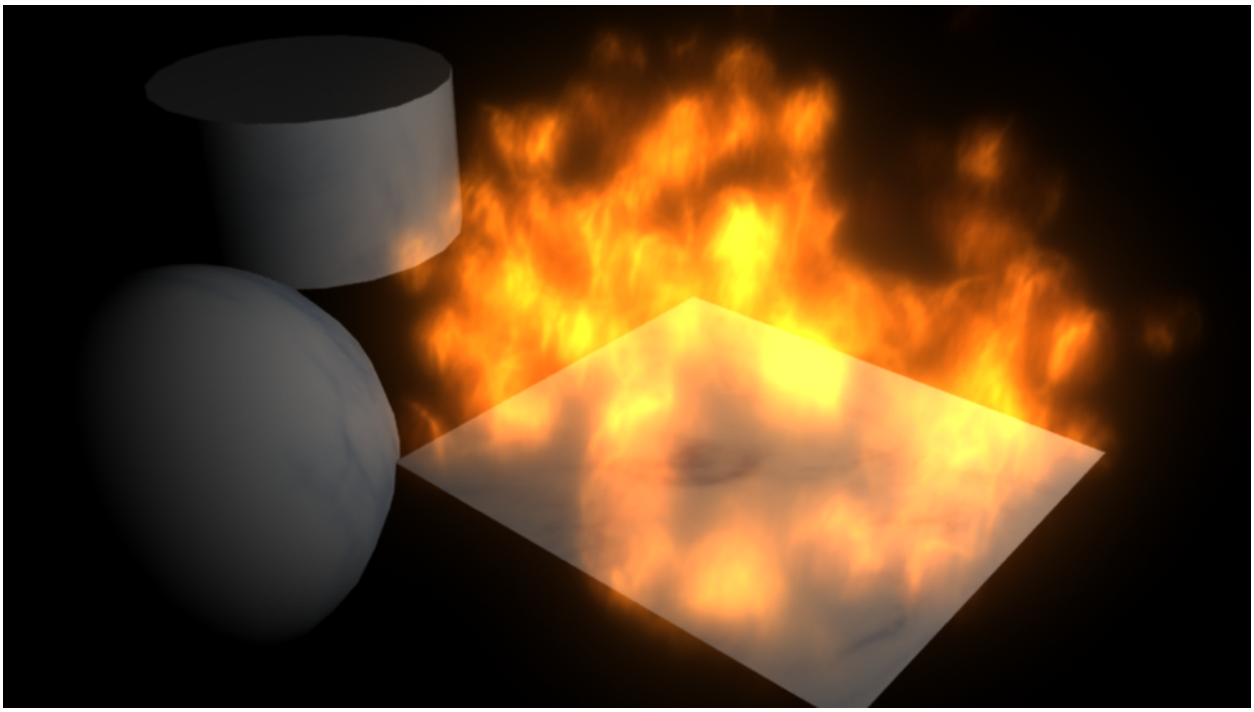
(a) Scene without fire in Maya® , rendered with Mental Ray® .

All the images generated with our shader use 256 light samples, the volumes have resolutions of $256 \times 256 \times 256$ voxels and a ray march step size of half a voxel size. To give some perspective of the evolution of the software, an image from an early stage of the application is shown in Figure 5-5. At that point standard ray marching was used, a white point light source was placed in the centre of the cube, and the colour of the volume was set manually to orange. The final colour of each pixel was computed using alpha blending with the densities along the direction of the ray. The shadows form due to an attenuation factor being computed in the shadow rays.

For the next step, we assume that our flame is a black body radiator, as shown in Figure 5-6. In each step of the ray marching, the 256 lights scattered throughout the fire are sampled to compute the colour. Since no fuel is burning, there is no absorption coefficient to be computed, so the alpha blending is still used.



(b) Maya® fire effect on a plane, rendered with Maya® software.



(c) Maya® fire effect on a plane, rendered with Mental Ray® .

Figure 5-2: Render tests for Maya® default fire effect.

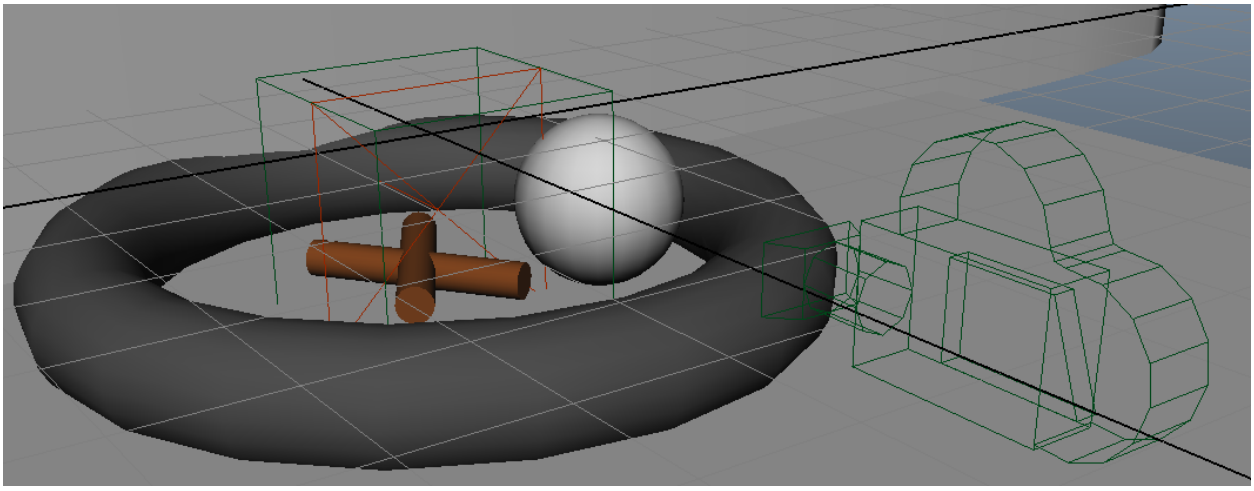


Figure 5-3: Our test scene in Maya®.

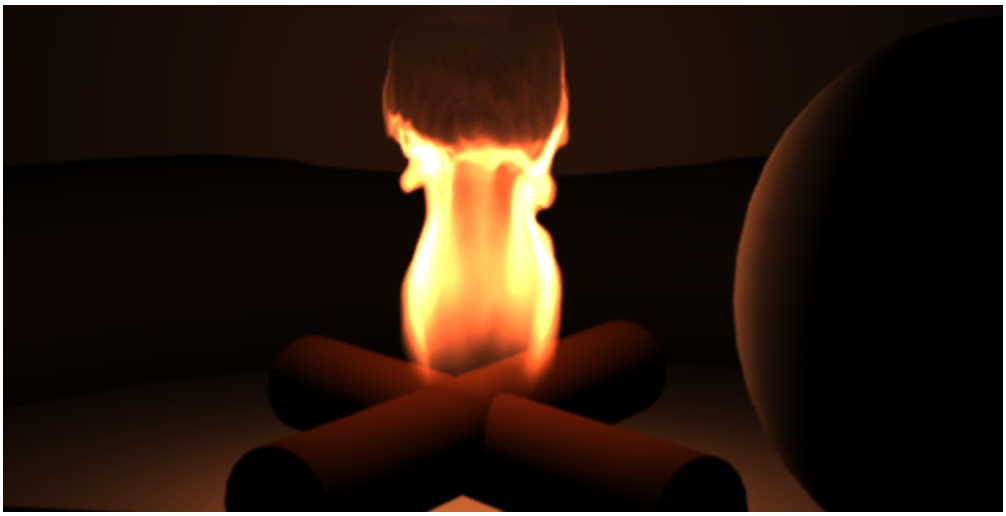


Figure 5-4: One of the Maya® fire fluid examples in our test scene.

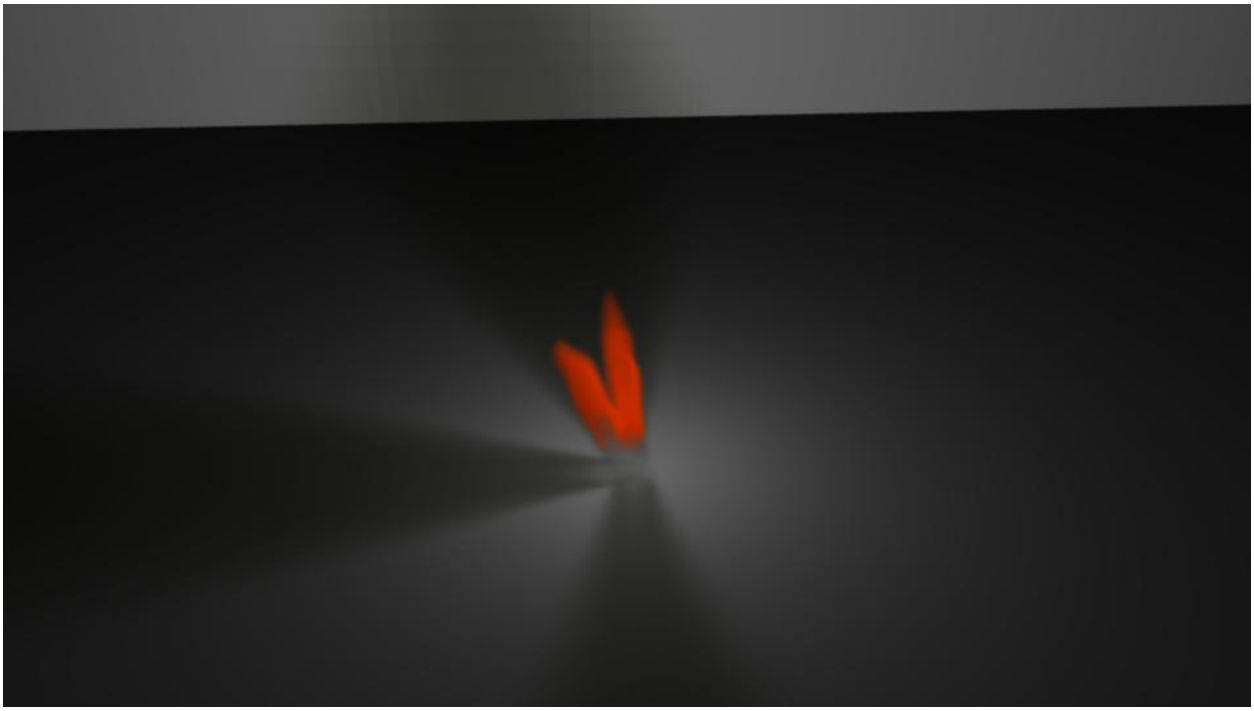


Figure 5-5: Our shader, early test, basic ray marching.

Figure 5-6: Our shader, flame as a black body radiator.

If we model the fire with a real fuel, for example propane, the image shown in Figure 5-7 is generated. At this point we are rendering with the simplified RTE Equation 4.2. Note how the appearance of the flame improves significantly with respect to the previous result. If we omit the computation of the attenuation factor for the emitted light L_e, rendering times are reduced significantly. The result is shown in Figure 5-8, the error incurred with this simplification is shown in Figures 5-9 and 5-10, computed as the absolute of the difference between both images. The bias introduced is recognizable when there is a ground truth image to compare to, yet if only one image were to be shown, it would be challenging to notice the difference.

Figure 5-7: Our shader, propane flame.

Figure 5-8: Our shader, propane without emitted light attenuation.

More exotic fuels are also supported by our system, images with copper and sulphur as the burning chemicals are shown in Figures 5-11 and 5-12. To be able to render this data, the temperature of the flame must be increased, otherwise the emitted radiation would be outside of the visible spectrum.

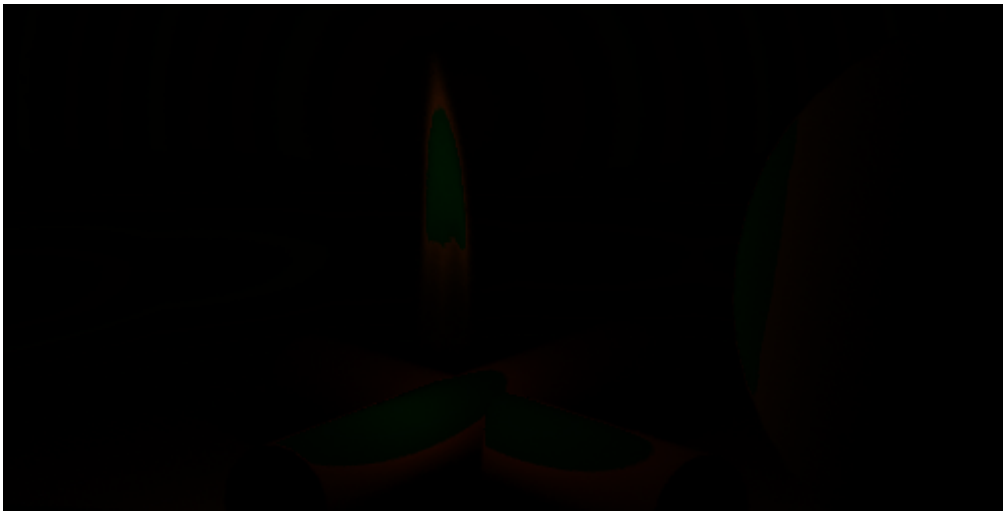


Figure 5-9: Error introduced when the attenuation is not computed.

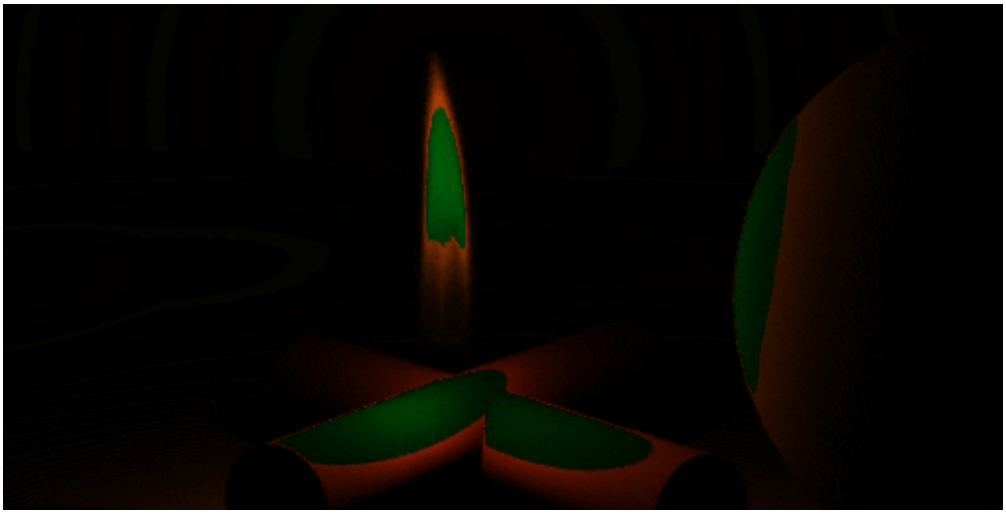


Figure 5-10: Previous error highlighted by a factor of 5.

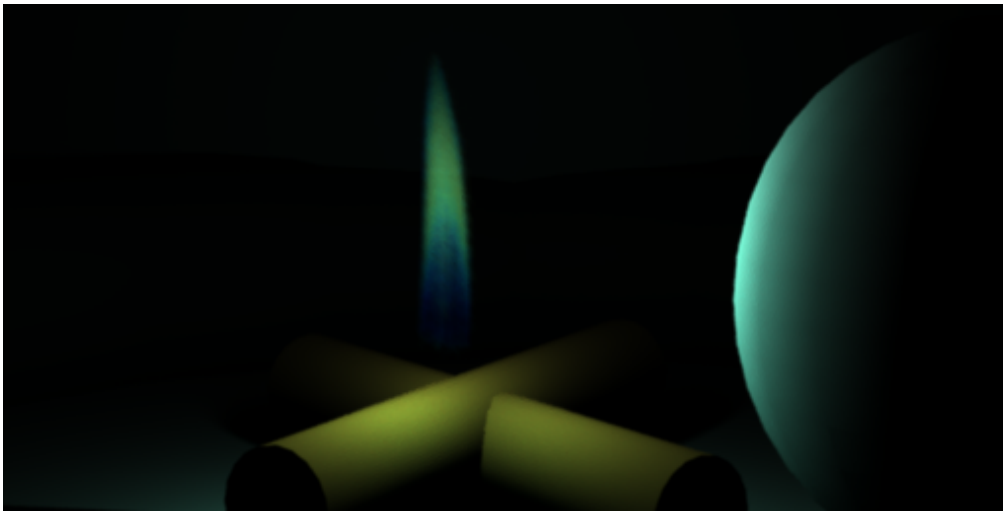


Figure 5-11: Our shader, copper flame.

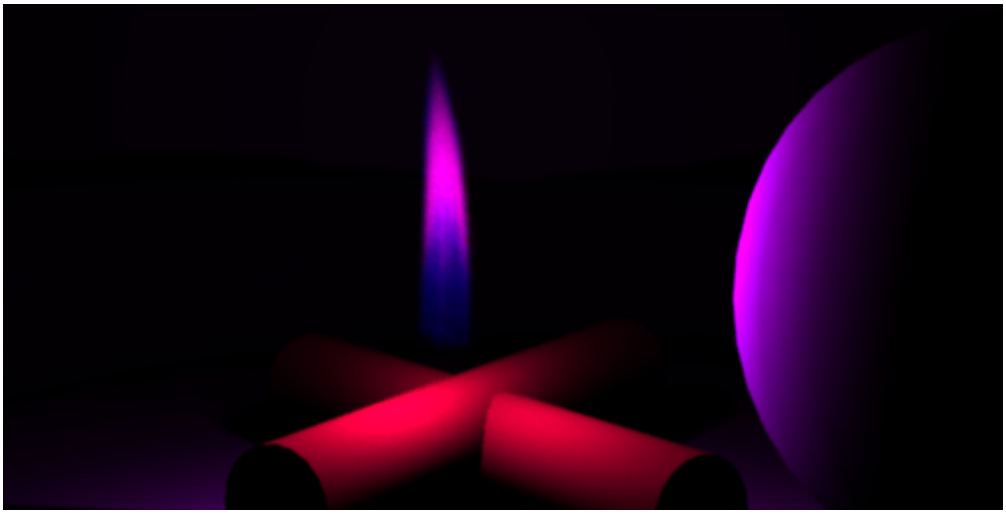
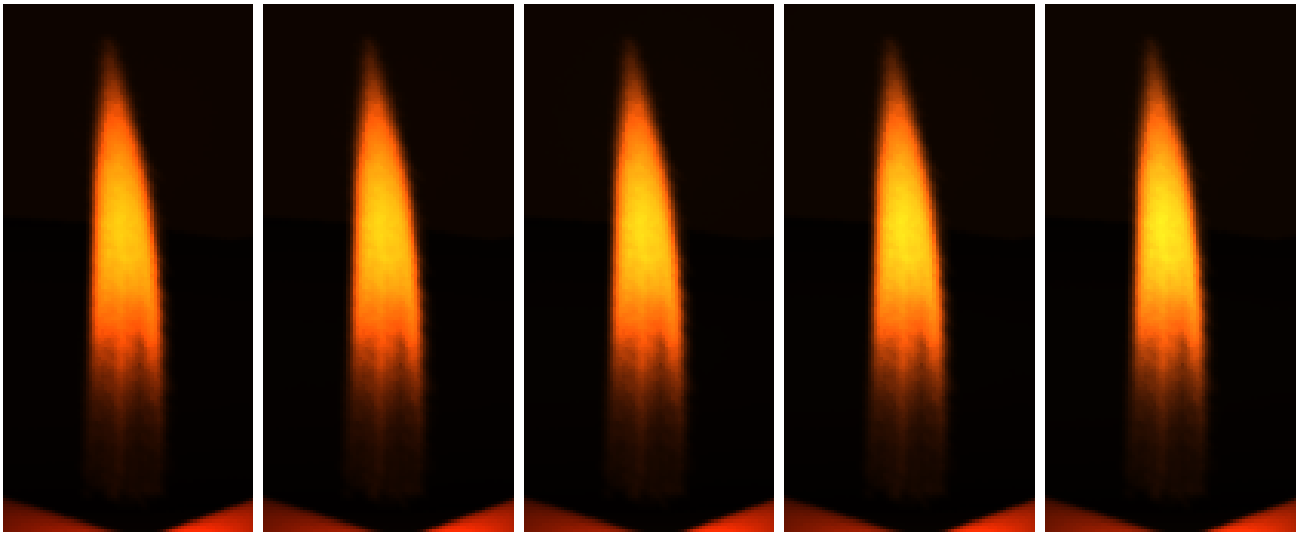


Figure 5-12: Our shader, sulfur flame.

The effects of the visual adaptation transformation are shown in Figure 5-13. As expected, when the visual adaptation factor increases, the centre of the flame shifts to a whiter colour.

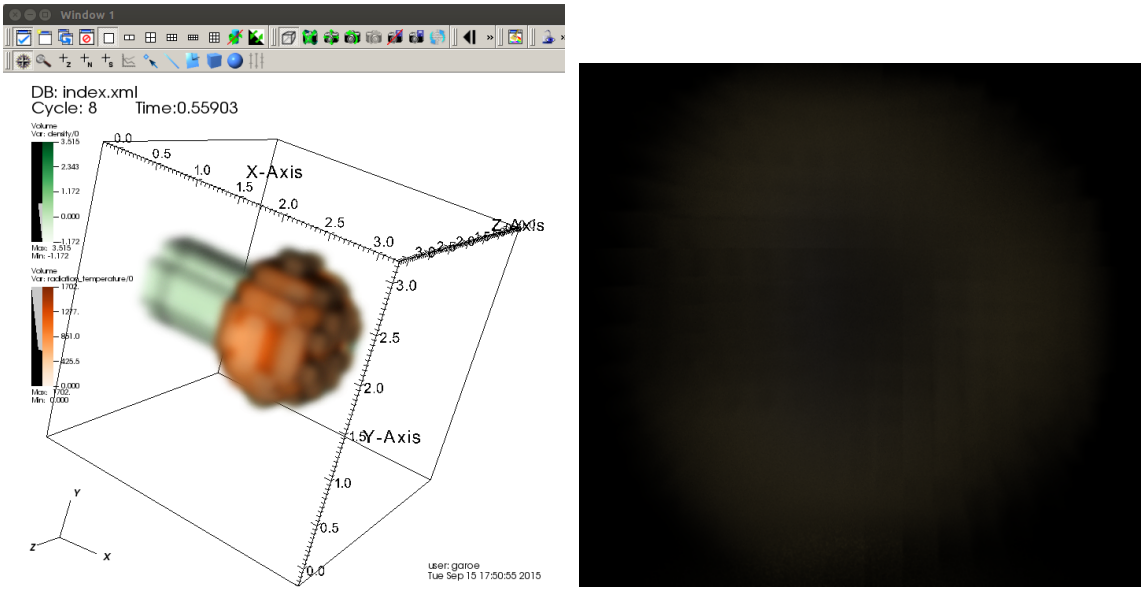


(a) 0%. (b) 25%. (c) 50%. (d) 75%. (e) 100%.

Figure 5-13: The effect of visual adaptation, left to right from no visual adaptation to maximum value.

In the work of Pegoraro and Parker, the authors use data from the Uintah library [Uintah, 2015] as the base input for their rendering results. We have ran some of the example simulations provided with their simulation tools. The data was first rendered using the VisIt volumetric visualization software, that was developed by the Lawrence Livermore National Laboratory [VisIt, 2015]. Figure 5-14a shows the result in VisIt using texture splats, densities are depicted using green and temperatures with orange, darker colours indicate higher densities/temperatures. Some ranges of data values were manually set to be fully transparent, as the whole cube is filled with voxels which occlude the view of the simulation. The data rendered with our shaders is shown in Figure 5-14. The obvious differences between both images are due to several factors, the camera position and orientation is not the same, and the parameters for our shader have been poorly chosen in this case.

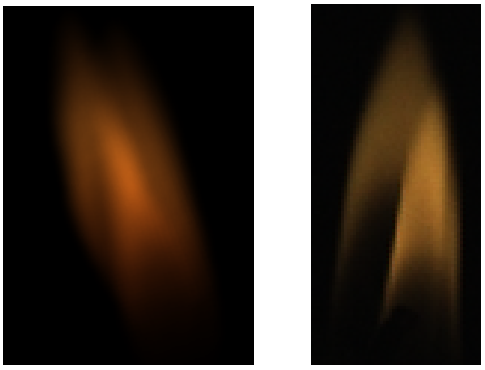
An interesting phenomena is the visual appearance of flames which are superimposed, as the addition of the colours and transparency is not trivial, as shown in Figure 5-15a. To better illustrate this effect, an image of similar situation with real flames captured with our system is shown in Figure 5-15b.



(a) Visualization using VisIt.

(b) Visualization using our shader.

Figure 5-14: Simulation data from the heptane example in the Uintah software [Uintah, 2015].



(a) Our shader, two flames superimposed.

(b) Captured image of real flames.

Figure 5-15: Superimposed fire effects.

Chapter 6

Conclusions and Future work

6.1 Conclusions

We have presented a physically-based rendering algorithm for flames, the model uses the Radiative Transport Equation, which describes heat transfer in real flames; the variations produced when using different types of fuels, and the effects of the visual adaptation processes in the human eye. The model carries all the limitations inherent to ray-tracing techniques, such as large computational costs, with render times per frame varying from a few seconds in low quality settings, to several hours for converged images.

The method assumes a spectral representation of the world, however even if we were to compute the final result using the spectral domain, a final transformation to RGB space and gamma correction is always needed in order to display the image on a monitor. The RGB conversion incurs in an unavoidable loss of information, moreover in order to achieve practical rendering times, the transformation is often computed earlier.

Several authors such as [Hong et al., 2007], [Horvath and Geiger, 2009] and [Jamriška et al., 2015] use simpler rendering techniques than the one discussed in this report [Pegoraro and Parker, 2006]. Nevertheless, the results obtained by Pegoraro and Parker appear to be surpassed by the aforementioned authors; superior input data seems to be the cause of the disparity in quality. This effect is more evident in the case of [Hong et al., 2007], where data with added "vorticity" gives the impression of a significant output improvement, even if it is not based on any physical phenomena, .

Although the method is physically based, Pegoraro and Parker did not conduct any formal analysis of the validity of their model. Such study would require the measurement of the physical properties of a real flame, such as densities, temperatures and spectral emissions; and comparing the data with values computed using the authors' technique.

6.2 Future Work

An interesting area of future work involves automatically setting the shader parameters for a given scene. There are at least two paths that be used to give a good parameter estimation if we have captured data. The methods are, gradient descent using image derivatives or reconstructing the spectrum which produced the cameras RGB responses. Both techniques will be explained in greater detail in the Sections 6.2.1 and 6.2.2.

A common line of work in the computer graphics community is the development of importance sampling for BxDF rendering models [Lawrence et al., 2004], [Ou et al., 2012] and [Wang et al., 2014]. The core idea is to sample more often the values that contribute more to the final image, by means of a biased distribution instead of an uniform sampling scheme. In our implementation the emitted radiance L_e is uniformly sampled, and in the general case the ω_i directions for in-scattering light L_i would have been naively sampled as well. In order to design a "good" biased distribution, prior knowledge of the location of the important regions is needed. For L_e and L_i we have some intuitive insight on were such regions would lie, samples whose origin is close to the interest point should have greater contributions, as the influence of further ones will be diminished by the light distance falloff effect. Another factor which must be considered is the relative intensity emitted at source by a sample, as it is expected that brighter points will have larger contributions.

6.2.1 Image differences

Given a ground truth captured image I_f, we would like to transform an image I_0, rendered with our method, so that it resembles I_f. Formally the transformation is defined as

$$I_{i+1} = t(I_i,\ d(I_i,\ I_f)), \tag{6.1}$$

where I_i is the image in the i^{th} step, d is an image difference function, and t is a function that transform I_i in the direction of the gradient given by d. There are several terms in the aforementioned equation that can be explored in more detail. A number of image difference filters have been proposed, such as the Sobel and the Scharr filters. The transformation function t must convert from image derivatives space to the physical parameter space that our shaders use, e.g temperature and density. The simplest technique to compute the next step in the parameter space would be gradient descent. As there are several parameters to explore, and their behaviour is non-linear, it is reasonable to expect that the function will have a number of local minima, thus requiring more advanced optimization methods, such as genetic programming [Dobashi et al., 2012].

6.2.2 Spectrum reconstruction

In a captured image I_f, each pixel stores the RGB values that were measured by the sensor at that position. When the camera is exposed to a spectral signal, the spectrum is collapsed to RGB space using the sensor spectral sensitivity curves for each color, the curves are shown in Figure 6-1. In continuous form, the collapsing of the signal is driven by

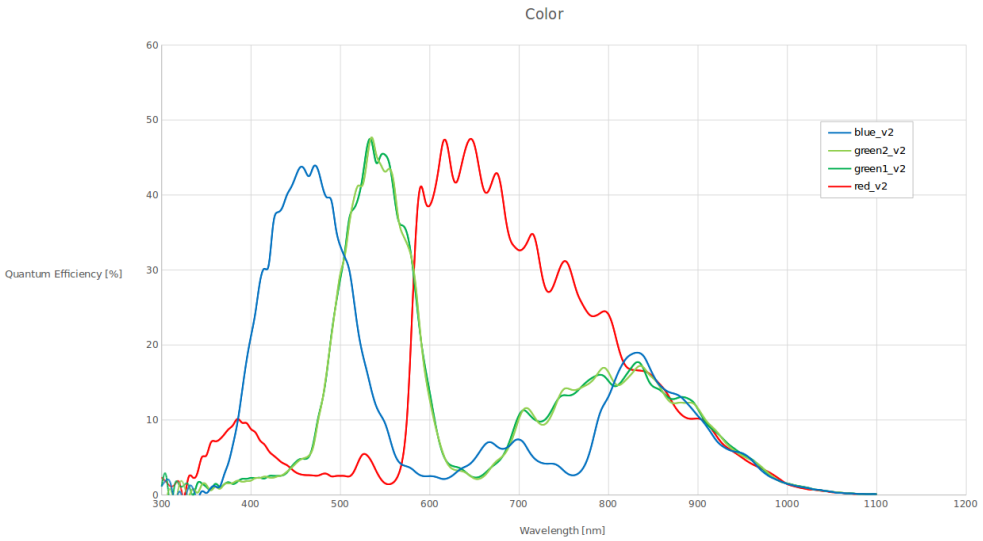


Figure 6-1: Spectral sensitivity curves for the sensors in our cameras.

$$
\begin{aligned}
r &= \int i(\lambda) s_r(\lambda) d\lambda, \\
g &= \int i(\lambda) s_g(\lambda) d\lambda, \\
b &= \int i(\lambda) s_b(\lambda) d\lambda,
\end{aligned}
\tag{6.2}
$$

where r, g and b are the output coefficients, $i(\lambda)$ is the input spectrum, $s_r(\lambda)$, $s_g(\lambda)$ and $s_b(\lambda)$ are the spectral sensitivity curves and λ is a wavelength number. Discretized for n samples the previous equation can be written as

$$
\begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} s_r(\lambda_0) & s_r(\lambda_1) & \cdots & s_r(\lambda_n) \\ s_g(\lambda_0) & s_g(\lambda_1) & \cdots & s_g(\lambda_n) \\ s_b(\lambda_0) & s_b(\lambda_1) & \cdots & s_b(\lambda_n) \end{bmatrix} \begin{bmatrix} i(\lambda_0) \\ i(\lambda_1) \\ \vdots \\ i(\lambda_n) \end{bmatrix}. \tag{6.3}
$$

If the original signal $i(\lambda)$ were to be reconstructed, the physical parameters could be computed using the equations presented in Chapter 3. An image rendered with those parameters would closely resemble the original data. The main challenge in this situation lies in the highly underconstrained nature of the problem, n unknowns in $i(\lambda)$ are to be solved with only three equations from r, g and b. Fortunately, several methods have been proposed, [Smits, 1999], [Sun et al., 2001] and [Drew and Finlayson, 2003], to compute an approximation of $i(\lambda)$ using optimization techniques with a set of prior constrains, such as smooth spectral curves and spatial coherency within the image.

Bibliography

[Bridault-Louchez et al., 2006] Bridault-Louchez, F., Leblond, M., and Rousselle, F. (2006). Enhanced illumination of reconstructed dynamic environments using a real-time flame model. In *Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, AFRIGRAPH '06, pages 31–40, New York, NY, USA. ACM.

[Chadwick and James, 2011] Chadwick, J. N. and James, D. L. (2011). Animating fire with sound. *ACM Trans. Graph.*, 30(4):84:1–84:8.

[Ciddor, 1996] Ciddor, P. E. (1996). Refractive index of air: new equations for the visible and near infrared. *Appl. Opt.*, 35(9):1566–1573.

[Dalzell and Sarofim, 1969] Dalzell, W. H. and Sarofim, A. F. (1969). Optical constants of soot and their application to heat-flux calculations. *ASME Journal of Heat Transfer*, 91(1):100–104.

[Davis, 1992] Davis, R. S. (1992). Equation for the determination of the density of moist air (1981/91). *Metrologia*, 29(1):67.

[Dobashi et al., 2012] Dobashi, Y., Iwasaki, W., Ono, A., Yamamoto, T., Yue, Y., and Nishita, T. (2012). An inverse problem approach for automatically adjusting the parameters for rendering clouds using photographs. *ACM Trans. Graph.*, 31(6):145:1–145:10.

[Drew and Finlayson, 2003] Drew, M. S. and Finlayson, G. D. (2003). Multispectral processing without spectra. *J. Opt. Soc. Am. A*, 20(7):1181–1193.

[Fairchild, 2005] Fairchild, M. D. (2005). *Color appearance models*. John Wiley & Sons.

[Feldman et al., 2003] Feldman, B. E., O'Brien, J. F., and Arikan, O. (2003). Animating suspended particle explosions. *ACM Trans. Graph.*, 22(3):708–715.

[FumeFX, 2015] FumeFX (2015). FumeFX. http://www.afterworks.com.

[Hasinoff and Kutulakos, 2003] Hasinoff, S. and Kutulakos, K. (2003). Photo-consistent 3d fire by flame-sheet decomposition. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1184–1191 vol.2.

[Henyey and Greenstein, 1941] Henyey, L. G. and Greenstein, J. L. (1941). Diffuse radiation in the Galaxy. *The Astrophysical Journal*, 93:70–83.

[Hong et al., 2007] Hong, J.-M., Shinar, T., and Fedkiw, R. (2007). Wrinkled flames and cellular patterns. *ACM Trans. Graph.*, 26(3).

[Horvath and Geiger, 2009] Horvath, C. and Geiger, W. (2009). Directable, high-resolution simulation of fire on the gpu. *ACM Trans. Graph.*, 28(3):41:1–41:8.

[Howell and Siegel, 2002] Howell, J. and Siegel, R. (2002). *Thermal radiation heat transfer*. CRC press.

[Huang, 1998] Huang, P. H. (1998). New equations for water vapor pressure in the temperature range-100 c to 100 c for use with the 1997 nist/asme steam tables. In *Proc, of the 3rd International Symposium on Humidity and Moisture (Teddington: NPL)*, volume 1, pages 69–76.

[Huang et al., 2014] Huang, Z., Gong, G., and Han, L. (2014). Physically-based modeling, simulation and rendering of fire for computer animation. *Multimedia Tools and Applications*, 71(3):1283–1309.

[Hunt and Pointer, 1985] Hunt, R. W. G. and Pointer, M. R. (1985). A colour-appearance transform for the cie 1931 standard colorimetric observer. *Color Research & Application*, 10(3):165–179.

[Ihrke and Magnor, 2004] Ihrke, I. and Magnor, M. (2004). Image-based tomographic reconstruction of flames. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, pages 365–373, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Irawan et al., 2005] Irawan, P., Ferwerda, J. A., and Marschner, S. R. (2005). Perceptually based tone mapping of high dynamic range image streams. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*, EGSR '05, pages 231–242, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Jamriška et al., 2015] Jamriška, O., Fišer, J., Asente, P., Lu, J., Shechtman, E., and Sýkora, D. (2015). Lazyfluids: Appearance transfer for fluid animations. *ACM Transactions on Graphics*, 34(4).

[Jensen and Buhler, 2002] Jensen, H. W. and Buhler, J. (2002). A rapid hierarchical rendering technique for translucent materials. *ACM Trans. Graph.*, 21(3):576–581.

[Lamorlette and Foster, 2002] Lamorlette, A. and Foster, N. (2002). Structural modeling of flames for a production environment. *ACM Trans. Graph.*, 21(3):729–735.

[Lawrence et al., 2004] Lawrence, J., Rusinkiewicz, S., and Ramamoorthi, R. (2004). Efficient brdf importance sampling using a factored representation. *ACM Trans. Graph.*, 23(3):496–505.

[Lee et al., 2001] Lee, H., Kim, L., Meyer, M., and Desbrun, M. (2001). Meshes on fire. In Magnenat-Thalmann, N. and Thalmann, D., editors, *Computer Animation and Simulation 2001*, Eurographics, pages 75–84. Springer Vienna.

[Lokovic and Veach, 2000] Lokovic, T. and Veach, E. (2000). Deep shadow maps. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 385–392, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Melek and Keyser, 2005] Melek, Z. and Keyser, J. (2005). Multi-representation interaction for physically based modeling. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling*, SPM '05, pages 187–196, New York, NY, USA. ACM.

[Naka and Rushton, 1966] Naka, K. I. and Rushton, W. A. H. (1966). S-potentials from luminosity units in the retina of fish (cyprinidae). *The Journal of Physiology*, 185(3):587–599.

[Nguyen et al., 2002] Nguyen, D. Q., Fedkiw, R., and Jensen, H. W. (2002). Physically based modeling and animation of fire. *ACM Trans. Graph.*, 21(3):721–728.

[NIST, 2015] NIST (2015). NIST Database. http://www.nist.gov/pml/data/asd.cfm.

[Okabe et al., 2015] Okabe, M., Dobashi, Y., Anjyo, K., and Onai, R. (2015). Fluid volume modeling from sparse multi-view images by appearance transfer. *ACM Transactions on Graphics (Proc. SIGGRAPH 2015)*, 34(4):93:1–93:10.

[OpenVDB, 2015] OpenVDB (2015). OpenVDB Library. http://www.openvdb.org/.

[Ou et al., 2012] Ou, J., Xie, F., Krishnamachari, P., and Pellacini, F. (2012). Ishair: Importance sampling for hair scattering. In *ACM SIGGRAPH 2012 Talks*, SIGGRAPH '12, pages 28:1–28:1, New York, NY, USA. ACM.

[Pegoraro and Parker, 2006] Pegoraro, V. and Parker, S. G. (2006). Physically-based realistic fire rendering. *Natural Phenomena*, pages 51–59.

[Perlin, 1985] Perlin, K. (1985). An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296.

[Perlin and Hoffert, 1989] Perlin, K. and Hoffert, E. M. (1989). Hypertexture. *SIGGRAPH Comput. Graph.*, 23(3):253–262.

[Perry and Picard, 1994] Perry, C. H. and Picard, R. W. (1994). Synthesizing flames and their spreading. In *Proceedings of 5th Eurographics workshop on animation and simulation*, pages 105–117.

[Pharr and Humphreys, 2010] Pharr, M. and Humphreys, G. (2010). *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition.

[Reeves, 1983] Reeves, W. T. (1983). Particle Systems - a Technique for Modeling a Class of Fuzzy Objects. *ACM Trans. Graph.*, 2(2):91–108.

[Rushmeier et al., 1995] Rushmeier, H., Hamins, A., and Choi, M. Y. (1995). Volume rendering of pool fire data. *Computer Graphics and Applications, IEEE*, 15(4):62–67.

[Schlick, 1994] Schlick, C. (1994). An Inexpensive BRDF Model for Phsyically-based Rendering. In *Computer Graphics Forum*, volume 13, pages 233–246.

[Smits, 1999] Smits, B. (1999). An rgb-to-spectrum conversion for reflectances. *Journal of Graphics Tools*, 4(4):11–22.

[Stam, 1999] Stam, J. (1999). Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 121–128, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Stam and Fiume, 1995] Stam, J. and Fiume, E. (1995). Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 129–136, New York, NY, USA. ACM.

[Sun et al., 2001] Sun, Y., Fracchia, F. D., Drew, M. S., and Calvert, T. W. (2001). A spectrally based framework for realistic image synthesis. *The Visual Computer*, 17(7):429–444.

[Uintah, 2015] Uintah (2015). Uintah Library. http://uintah.utah.edu/.

[VisIt, 2015] VisIt (2015). VisIt Toolkit. https://wci.llnl.gov/simulation/computer-codes/visit/.

[Wang et al., 2014] Wang, C., Xie, F., and Krishnamachari, P. (2014). Importance sampling for a micro-cylinder based cloth bsdf. In *ACM SIGGRAPH 2014 Talks*, SIGGRAPH '14, pages 41:1–41:1, New York, NY, USA. ACM.

[Wei and Levoy, 2000] Wei, L.-Y. and Levoy, M. (2000). Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 479–488, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Wei et al., 2002] Wei, X., Li, W., Mueller, K., and Kaufman, A. (2002). Simulating fire with texture splats. In *Proceedings of the Conference on Visualization '02*, VIS '02, pages 227–235, Washington, DC, USA. IEEE Computer Society.

[Westover, 1990] Westover, L. (1990). Footprint evaluation for volume rendering. *SIGGRAPH Comput. Graph.*, 24(4):367–376.

[Yao and Stewart, 1996] Yao, J. and Stewart, D. S. (1996). On the dynamics of multi-dimensional detonation. *Journal of Fluid Mechanics*, 309:225–275.

[Zhang et al., 2011] Zhang, Y., Correa, C. D., and Ma, K.-L. (2011). Graph-based fire synthesis. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, pages 187–194, New York, NY, USA. ACM.

[Zhao et al., 2003] Zhao, Y., Wei, X., Fan, Z., Kaufman, A., and Qin, H. (2003). Voxels on fire [computer animation]. In *Visualization, 2003. VIS 2003. IEEE*, pages 271–278.