NTNU
Norwegian University of
Science and Technology

Framework for real-time forest fire animation

Simulating fire spread using the GPU

Øystein Kjærnet

Master of Science in Computer Science

Submission date:	June 2010
Supervisor:	Torbjørn Hallgren, IDI
Co-supervisor:	Jo Skjermo, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

A framework for simulating and animating forest fire in real-time applications is to be developed and implemented. The framework is a two-part system, consisting of a large-scale simulator, describing the spread of fire between trees in a forest, and a small-scale simulator for fire spread on a single tree. Only the small-scale part is to be developed in this project. The framework should support external factors affecting the spread rate, a simple flame representation and should use a quasi-physically based spread simulation model.

Assignment given: 18. January 2010
Supervisor: Torbjørn Hallgren, IDI

Abstract

In 2009 Odd Erik Gundersen and Jo Skjermo described a conceptual framework for animating physically based forest fires. This project expands on their ideas with a focus on how modern graphics hardware can be utilized to achieve real-time performance. A prototype demonstrating some of the concepts suggested for the framework have been implemented and tested, successfully achieving real-time frame rates on a simple animation of a burning tree.

Contents

1 Introduction

In nature forest fire is a most devastating phenomenon - an estimated cost of 2.5 billion USD and 22 deaths was reported for a single wildfire in Southern California in October – November 2003 [Lott and Ross, 2006]. Still, fire can also be a dramatic and beautiful effect in visual entertainment, and is used extensively in the movie industry and to a lesser degree in interactive entertainment like video games. The problem with using computer animated fire for video games and similar is it's complex and highly dynamic nature. Much work has gone into making the flames of computer animated fires look realistic, but the spread of fire through fuel has received less attention from the gaming society. Just as the appearance of the flames, this is a complex phenomenon that can be computation intensive to simulate, and thus a challenge to implement in real-time computer applications. Fire fighters have used computer simulation as a tool in fire prediction for some time, but these simulations are slow and must be run off-line. This project aims to develop a framework for supporting realistic fire spread in real-time applications. It's goals are simplicity in use and implementation and low computational overhead, while maintaining a spread pattern that is believable both from a close-up perspective of a few trees and over whole forests.

1.1 Goals

The main goal of this project is to describe a general method for adding support for realistic fire spread to real-time, interactive 3D forest scenes. The approach chosen was to use the framework described in [Gundersen and Skjermo, 2009] as a basis to expand on and concretize. The framework should be described in sufficient detail to be a useful basis for a future implementation. This means potential problems should be thoroughly investigated and possible solutions should be suggested. The framework should be general enough to fit the architecture of already existing real-time applications, but it's scope is fire in forests and as such it does not need to support fire spread over arbitrary 3D-objects. It should be easy to understand and implement. The spread of the fire should seem plausible from the user's viewpoint, both at a large scale when considering the whole forest, and at a detailed level when considering the spread over a single tree. It is, however, aimed at entertainment applications and need not provide a degree of accuracy suitable for use in forest-fire prediction. As it is a framework for use in real-time applications it should not impose a large computational overhead. To achieve this as much as possible of the computational intensive parts of the simulation should be implemented on graphics hardware.

2 Background Theory

To fully understand fire one must touch on several fields of science, including physics and chemistry. However, the main concerns of this project are the mechanisms involved in spreading fire across a fuel bed, and it turns out that for this specific aspect of fire physical factors may be more important than the chemical nature of fire [Drysdale, 1998, p. 16].

This chapter first introduces some basic topics related to fire in general to provide some foundation for the following section, which focus on fire spread. Furthermore, as this project aims only to achieve believable animations of spreading fire rather than predicting the behaviour of a real fire for purposes such firefighting, only the most general cases are considered.

2.1 Fundamentals of Fire

Fire is a complex process and involves a number of chemical reactions. Central is *combustion*, a process involving rapid reaction between an oxidizer, and some combustible species [Anonymous, 2010], accompanied by the release of energy, primarily in the form of heat [Drysdale, 1998, p. 17]. Reactions that release energy is called *exothermic*, while those that absorb energy are *endothermic* [Griffiths and Barnard, 1995; Strahle, 1993, p. 12; p. 15]. Many chemical compounds are combustible, and the term “fuel” is used throughout this text to mean whatever compounds are reacting with the oxidizer in the combustion process, though the case of carbon based fuels from wood reacting with the oxygen in air will be the one most relevant.

The rate of the chemical reaction, and thereby the rate at which heat is released, depends on the energies of the fuel and the oxidizer [Griffiths and Barnard, 1995, p. 3, 145]. The hotter the reactants, the quicker heat is released in the combustion. If the rate at which heat is generated equals the rate at which heat is lost to the surroundings, a self sustained chain reaction can be achieved, as the heat generated in on cycle of combustion heats up unburnt fuel maintaining the heat generation rate. Furthermore, if the rate of heat generation *exceeds* the rate of heat loss, heat can build up, continually accelerating the heat release and causing a thermal runaway [Strahle, 1993; Quintiere, 2006, p. 92; p. 82]. This point is called “ignition”, and the temperature at which it occurs is called the “ignition temperature”[Griffiths and Barnard, 1995, p. 3].

When a volume of fuel and air mixture at a temperature less than the ignition temperature is subjected to a local source of energy, such as a spark or a matchstick, that raises the temperature locally to a point at which some of the mixture reaches ignition, the ignition can propagate through the fuel. This is called “piloted ignition”, and the speed at which the ignition propagates is called the “flame speed” [Quintiere, 2006;

Griffiths and Barnard, 1995, p. 85-88; p. 3].

In naturally occurring fire the limiting factor of the ignition propagation is usually the availability of air and fuel mix; the fuel is originally separate from the air, and they must first mix by diffusion. In larger fires, such as bonfires or forest fires, this process is helped by the turbulence caused by the effect of the heat on the surrounding air, and the flames that results are characterized as "turbulent diffusion flames" [Beever, 2008, p. 1].

Often when a complex compound burns it goes through another chemical reaction before it takes part in the combustion: Triggered by heat the compound decomposes by a process known as *pyrolysis*, leaving smaller molecules that may become fuel for the combustion or may be left untouched [Beever, 2008, p. 1]. This is an endothermic reaction, meaning heat is consumed in the process. When the fire originates from wood, which consists mainly of cellulose, hemicellulose and ligning [Pettersen, 1984], pyrolysis may release volatile species leading to flaming combustion [Pyne et al., 1996, p 15]. In a forest fire, it is normally these volatiles that ignite first, thus driving the propagation [Pyne et al., 1996; Quintiere, 2006, p. 6;p. 159].

Due to several different phenomena volatile species from the pyrolysis frequently escapes from the combustion zone without oxidizing. Heated by combustion these species can form soot particles that rise with the buoyant air flow, and if hot enough can emit light due to a phenomenon called *incandescence*. The characteristic yellow light from most flames come mainly from such incandescent soot, while soot that has cooled down and is no longer incandescent contributes to the smoke rising from the tip of the flame [Drysdale, 1998; Beever, 2008; Griffiths and Barnard, 1995, p. 25; p. 1; p116].

2.2 Fire Spread

A fire spreading in wood can be seen as a series of alternately endothermic and exothermic processes, with the following playing a major role [Pyne et al., 1996; Rothermel, 1972, p. 6; p. 8]:

Dehydration : One of the first effects of the heat transfered from a nearby combustion is the endothermic dehydration of the wood. Water and other liquids with relatively low boiling points evaporates.

Pyrolysis : As the temperature increases, pyrolysis becomes more dominant, breaking down the wood into tar, char and several other components, including volatile species that detaches from the surface into the air. Pyrolysis is also an endothermic process.

Combustion : Some of the products from the pyrolysis reaches ignition temperature, and a self-sustained exothermic combustion is initiated.

Some of the heat from the combustion then reaches nearby, unburnt fuel, and starts a new cycle. The combustion continues as long as there is enough heat to sustain the process, either as flaming combustion if volatile fuel is available, or as a glowing combustion of solid carbon based fuel. In this way an ever increasing area of the fuel is, or has been, involved in combustion and we say that the fire spreads over the fuel. It can be useful to define the boundary between the areas of the fuel that has not been reached by the combustion and the burning areas. This is often called "the surface of fire inception" or "inception boundary" and the speed with which this moves in the direction of it's normal is called the "spread rate" [Williams, 1977].

2.2.1 Heat transfer

If the energy released in a combustion process is to have any affect on nearby fuel, clearly a way to transfer energy between a source and a recipient is needed. Heat is defined in physics as energy transferred between systems as a result of their temperature difference [Reif and Reif, 1965; Pyne et al., 1996, p. 67; p. 12]. A useful measure for describing the heat transfer is "heat flux", which is defined as flow of energy per unit of area per unit of time. It is a vectorial quantity and it's magnitude is often measured in watt per square meter (W/m^2) [Strahle, 1993, p. 84]. The spread rate, can be related to the magnitude of the heat flux through the surface of fire inception, using equation 1 known as "the fundamental equation of fire spread":

$$\rho V \Delta h = q \tag{1}$$

derived from applying conservation of energy across the boundary [Perry and Picard, 1994]. In addition to the spread rate, V, and the heat flux, q, it is dependent on the fuel density, ρ and the difference in "thermal enthalpy" per unit mass, Δh, between the fuel at it's ignition temperature and it's original temperature. Thermal enthalpy is a measure of a chemical species' energy, involving it's internal energy, pressure and volume [Griffiths and Barnard, 1995, p. 11].

The heat can be transfered in three by three different mechanisms, often referred to as the *mechanisms of heat transfer* [Drysdale, 1998, p. 31]: Conduction, Convection and Radiation.

Conduction is transfer of heat between substances or parts of a substance in direct contact, when substantial movement is not involved [Pyne et al., 1996, p. 12].

The heat flux for conductive heat transfer in the x direction is given by:

$$q = -k\frac{\Delta T}{\Delta x} \tag{2}$$

where ΔT is the temperature difference over the distance Δx and k is the thermal conductivity - a material constant [Drysdale, 1998, eq. 2.1. p. 32].

Convection transfers heat between a solid and a surrounding, moving fluid [Drysdale, 1998, p. 32]. For convective heat transfer an empirical relationship for the heat flux is

$$q = h\Delta T \tag{3}$$

where h depends on both the system, ΔT and other factors and is known as the convective heat transfer coefficient.

Radiation is the transfer of heat by electromagnetic waves [Pyne et al., 1996, p. 13] and does not require direct contact between the source and the receiver [Drysdale, 1998, p. 31]. The relatively large soot particles (often 10-100 nm in diameter) present in the yellow luminous flames in naturally occurring fire are important sources of radiative heat transfer [Drysdale, 1998, p. 69]. The heat flux for this type of heat transfer is given by:

$$q = \phi\varepsilon\sigma T^4 \tag{4}$$

. The parameters are: ϕ - a factor accounting for the geometrical relationship between the emitter and the receiver, ε - the emissivity of the radiator and σ - the Stefan-Boltzmann constant. The temperature, T, is in Kelvin in this equation.

2.2.2 Dominant Heat Transfer Mechanism

In reality all three mechanisms probably contribute to the spread of any given fire, but it is found that a "dominant heat transfer mechanism" can be identified for some usual situations [Pyne et al., 1996; Drysdale, 1998; Fernandez-Pello and Hirano, 1982; Williams, 1977, p. 12; p. 31]. An important difference is how the direction of spread relates to the direction of the surrounding gas flow - classified into opposed flow spread and concurrent spread. The gas flow can be determined by external wind or it can arise from the buoyancy effect caused by the rising hot gases from the fire. The slope of the fuel surface plays an important role in the effect of buoyancy induced flow. As illustrated in figure 1, the buoyancy from the fire on a flat surface, will create a draft resulting in opposed spread, while on a slope the hot gases will create a upslope flow. Thus a fire spreading up the slope will be concurrent and spreading down the slope it will oppose the flow [Drysdale, 1998]. It is also usual to differentiate between spread in thermally thick and thin solids [Fernandez-Pello and Hirano, 1982]. When an object is so thin that the internal spatial temperature gradient can be ignored, it is called thermally thin [Quintiere, 2006, p. 171]. In thermally thick solids the thickness has an impact on the rate of spread [Williams, 1977, p. 1285], but the variation with thickness decreases drastically for solids thicker than 1.5 mm and is virtually zero if thicker than 5 mm [Drysdale, 1998, p. 244]. The heat transfer mechanism that dominates the the spread rate for concurrent spread is radiation from the flame that leans over the unburnt fuel. For opposed spread the dominating mechanism is slightly different for thermally thin and thick fuel. While opposed spread through thermally thick fuels is dominated by heat conduction through the solid, the spread through thermally thin solids is dominated by heat conduction through the gas as the path for the transfer of heat through the solid is very limited [Fernandez-Pello and Hirano, 1982; Williams, 1977].

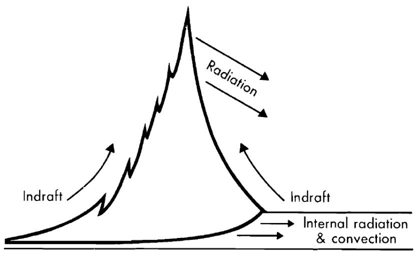
2.2.3 Factors affecting fire propagation

There are a number of physical factors affecting the speed of the propagation, including:

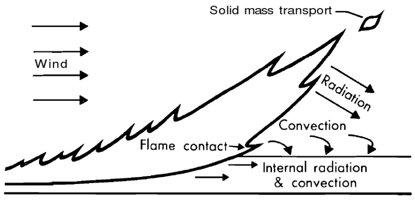
Wind is considered the most important factor of forest fire propagation by a large margin, being able to accelerate the propagation by a factor of 100 compared to windless fire [Viegas, 1998, p. 6, 17]. Among other things, this acceleration is caused by the gas flow driving the flame ahead of the pyrolysis front, enhancing the transfer of heat from the flame to the unburnt fuel [Fernandez-Pello and Hirano, 1982, p. 19], especially radiative heat transfer as discussed in 2.2.2. Larger particles, sparks and similar can also be blown by the wind to ignite virgin fuel. Wind opposing the spread direction can for small air velocities increase the spread rate as it promotes air and fuel mixing and increases combustion. For higher velocities it can cool the unburnt fuel decreasing the spread rate.

Slope has similar effect on the spread rate to the effect of wind because of the buoyancy-induced gas flow as explained in section 2.2.2.

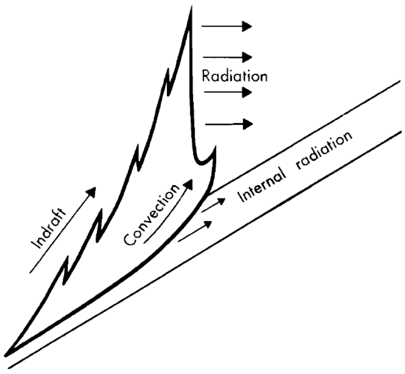
Fuel density is another major factor on the spread rate, as low density means only a small mass of material at the surface needs to be heated for the flame to spread [Drysdale, 1998, p. 245]. However, when the fuel becomes discontinuous the increased distance between fuel elements means heat has to be transfered a longer distance.



(a) Horizontal fire spread without wind



(b) Horizontal fire spread in wind



(c) Fire on a slope

Figure 1: The effect of wind and slope on the fire. Without wind (1a) the draft causes opposed spread in any direction. With wind (1b) or slope (1c) the spread is concurrent if spreading from left to right on the figure and opposed if spreading from right to left. Illustrations from [Rothermel, 1972].

Fuel geometry , in particular the width of a fuel bed can be an important factor: the size of the flames on a bed of substantial width means radiative heat transfer can become the dominating heat transfer mechanism [Drysdale, 1998, p. 246].

Fuel moisture can be a heat sink or a chemical inhibitor in the pyrolysis process [Viegas, 1998, p. 6].

Fuel temperature influences the amount of heat required to pyrolyze and ignite unburnt fuel. The fuel temperature for a forest fire depends on environmental factors such as the ambient temperature and wind conditions.

2.2.4 Regimes of propagation

Forest fires spread mainly through three different layers of the forest: beneath the surface, on the surface and through the tree crowns. The fire behaves quite differently in the different layers, and it is normal to classify fire spread into three “regimes” according to which of the layers that mainly fuel the fire [Viegas, 1998, p. 4–5]:

Ground fire Pyrolysis and combustion can propagate in the organic layer of the ground. This does usually not involve flames but can evolve into surface fire.

Surface fire The forest surface can contain a lot of combustible fuel such as dead vegetation litter and living plants. This is the most common regime for forest fire propagation and also the best studied.

Crown fire Foliage can be highly combustible if sufficiently dry, and if the crowns of the trees form a relatively continuous canopy fire can spread very fast (over 11 km/h [Pyne et al., 1996, p. 69]) in this layer of the forest.

3 Related Work

There has been done some research in the computer graphics society on the topic of realistic animation of fire. However, most of it has focused on the appearance of the flames, rather than how the fire spreads over an object or an area with continuous or discrete fuel, which has been more or less reserved for research aimed at predicting fire for fire safety or -fighting. As the main focus of this framework is on the spread of a forest fire work related to fire spread is treated especially in section 3.2, while work focusing on the appearance of flames or smoke is mentioned somewhat more superficial in section 3.1.

3.1 Smoke and flame rendering

There is especially one class of fire modeling methods that has received much attention by computer graphics researchers in recent years: the ones that models the motion of gases by descretizing it and solving sets of physically derived equations. They often use techniques from the field of Computational fluid dynamics (CFD), and a few of them are summarized in section 3.1.1. Other techniques are reviewed in section 3.1.2. The animation of smoke is very similar to that of clouds, and the two are not distinguished between in the following sections.

3.1.1 CFD Based Methods

Computational fluid dynamics (CFD) is a field dedicated to solving problems involving fluid flows by the use of numerical methods. The equations used to describe the motion of the fluids are often the Navier-Stokes equations (NSEs) or some derivation of these. CFD techniques can achieve very accurate results and is used to evaluate new aircraft designs among other things[Taft, 2000]. Most of these methods require solving a large set of equations [Norton and Sun, 2006] and have traditionally been considered too computationally intensive for real-time 3D animation. However, fairly recent research in the computer graphics society, as well as the advances in computers' processing power, make CFD based techniques more suitable for such applications.

Simulating fluid motion numerically requires a discrete representation of the problem. There are two major ways to represent fluid flows, often referred in CFD as the Eulerian representation and the Lagrangian representation [Youquan et al., 2005; Price, 2006]. The Lagrangian representation defines a flow by tracking parcels of a specific volume of the fluid through the flow over time. A natural way to discretize a problem represented this way is by using particle systems, as introduced in [Reeves, 1983]. The Eulerian representation describe fluid velocity at fixed points in space. The discrete version of this representation is normally defining a finite number of such points by visualizing a grid over the simulation space.

One of the earliest applications of CFD techniques in the computer graphics society was [Yaeger et al., 1986], which describes how an Eulerian, two dimensional fluid simulation was used for the animation of the surface of Jupiter for the movie "2010". This simulation was run off line on a supercomputer at the time. Similar techniques were consequently used on several occasions for depicting fluid and gaseous phenomena such as water and smoke [Stam and Fiume, 1993] and for visualizing flames in [Stam and Fiume, 1995]. Real-time performance was achieved in [Kass and Miller, 1990], in simulating the hight field of a water surface with waves in shallow water, but for complex, turbulent structures such as fire CFD techniques were long considered too slow for interactive simulation. One of the problems with traditional fluid solvers is that they put a strict constraint on the size of the time step to guarantee convergence. [Foster and Metaxas, 1997] demonstrated that a visually realistic animations of turbulent gases can be achieved with Eulerian CFD techniques even at very coarse grids, thus yielding reasonable simulation performance. In [Stam, 1999] Stam showed how to overcome the problem of small time steps using a Semi-Lagrangian advection scheme and an implicit solver. He achieved unconditionally stable fluid simulations, allowing for arbitrarily large time steps. This fueled new interest in CFD based simulation in the graphics society, and several researchers have achieved highly realistic flame animations at interactive or near interactive frame rates using Stam's solver: [Melek and Keyser, 2002] uses a Stable Fluids-based method to simulate the development of fuel and exhaust gases, air and heat and supports burning of solids at a reported 20 frames per second (FPS) while [Nguyen et al., 2002] produced close to photo realistic flames using 5 minutes per frame (0.0033 FPS) with the addition of a technique called "vorticity confinement", invented by [Steinhoff and Underhill, 1994] and first used with the stable fluids method in [Fedkiw et al., 2001]. [Harris et al., 2003] uses the stable fluids method with vorticity confinement running all simulation on graphics hardware. By distributing the simulation over several rendered frames they successfully implemented the cloud simulation in an interactive flight application. This approach exploits the relatively slow changes in a cloud's appearance and may not be as suited for simulating flames. [Rødal et al., 2006] simulates 2D fire using a model similar to that in [Stam, 2000] and maps the result to a billboard to produce realistic small fires, such as torches and bonfires for real-time 3D animations. [Keenan Crane and Tariq, 2007] describes some techniques used to produce realistic fluid simulation in the game "Hellgate: London". Using the MacCormack advection scheme from [Selle et al., 2008] they reduced the numerical dissipation that the Stable Fluids approach struggles with. The methods were all implemented on the GPU to achieve real-time performance. [Melek and Keyser, 2003] expands on the authors' work in [Melek and Keyser, 2002] by adding deformation of the burning solid. They accomplish this by storing the burning object as a distance field, moving the implicit surface as it burns and converting it to a polygonal representation before rendering.

Another method based on CFD techniques is the Lattice Boltzmann Method (LBM), which instead of focusing on macroscopic qualities of fluids, like velocity, density and pressure, as the Navier-Stokes equations do, tries to simulate the behavior of microscopic particles [Wagner, 2008; Chen and Doolen, 1998]. It uses a technique similar to "cellular automata" with simple local rules for each cell in a regular grid. The technique was introduced to the computer graphics society in [Wei et al., 2002], who realized it was very well suited for being implemented as a parallel algorithm to be run on the GPU. They render the fire using a volume rendering method known as "texture splats", introduced in [Crawfis and Max, 1993]. This is an extension to "splatting" volume rendering that adds detail using small textures for each voxel in the rendering volume.

3.1.2 Other Methods

Most of the CFD-based techniques are computationally costly and were for the most considered impractical until recent years. Even now most real-time applications can not afford the overhead and use simpler methods. An early procedurally generated flame is described in [Perlin, 1985], where a static rendering of a turbulent 2D fire is created using a noise function. The method is extended to 3D in [Perlin and Hoffert, 1989]. A very popular technique in games is particle systems. These were introduced in [Reeves, 1983] where such a method was used to visualize an expanding ring of fire in the movie "Star Trek II: The Wrath of Khan". Real-time applications often use textured or billboard particles to reduce the number of particles needed for detailed animation [Van der Burg, 2000]. The flames in Intel's real-time demo application "Smoke demo"[Smith and Freeman, 2008] are produced by a particle system optimized for parallel execution on multi-core CPUs. [Kipfer et al., 2004] and [Latta, 2005] describe particle systems running entirely on the GPU, taking advantage of it's massive parallel computation capability and avoiding any slow transfers of particle data to GPU memory at run-time.

3.2 Fire Spread

Again, the fire spread methods are divided into those using CFD-techniques, and other methods including those based on models found in fire safety literature or simply the researcher's intuition.

3.2.1 CFD Based Methods

[Chiba et al., 1994] describes an early method for simulating fire spread in two dimensions by simulating particles flowing through a vortex field. [Stam and Fiume, 1995] simulates flames using CFD-techniques, and spreads the fire over solid objects by making them the source of fuel. [Ishikawa et al., 2005] simulates fire spread over solid fuels by enclosing the solid object in a voxel grid and calculating the evolution of a velocity field using Stam's Stable Fluids-approach. The ideas of simulating flames using a method based on the Lattice Boltzmann Method presented in [Wei et al., 2002] was later expanded by some of the same authors to simulate fire propagation on a volumetric object in [Zhao et al., 2003]. By running the simulation on graphics hardware they reportedly achieved a simulation rate of over 14 FPS on commodity hardware[1].

3.2.2 Other Methods

[Perry and Picard, 1994] takes another approach to simulating fire spread. Instead of calculating gas motion in a volume, the motion of the fire front is tracked directly as it expands over the object's surface. It is represented by a connected set of points, and the velocity perpendicular to the fire front, or *flame speed* is calculated by local rules depending on wind, slope and curvature. The flame speed calculation is based on [Fernandez-Pello and Hirano, 1982] and [Williams, 1977], that identified a "dominant spread mechanism" in different situations. For example will the contribution from radiation dominate the transfer of heat when the fire burns upslope, while downslope conductive heat transfer will contribute the most.

Other fire spread animations use similar techniques but with the flame speed calculated from intuition derived equations. [Beaudoin et al., 2001] and later [Lee et al., 2001] expand on [Perry and Picard, 1994]'s fire spread method with means to ensure correct expansion of the front without leaving the object's surface and without the front crossing itself. [Lee et al., 2001] also augments the fire front spreading with a particle system (the same as the one used to visualize the flames): "flame particles" emitted by the burning regions fly through the air and ignite any unburnt fuel they collide with. The collision check is accelerated by using a pre-computed distance field, storing the distance to the nearest surface point for each point of a coarse grid enclosing ignitable objects. A similar particle system is used in [Smith and Freeman, 2008] which is a real-time application made for demonstration purposes, in which fire is spread by emitting "heat particles" from burning objects. Another real-world example of fire spread in

[1] A PC having a 2.53 GHz P4 CPU with 1 GB of memory and an Nvidia GeForce4 Ti 4600 graphics card with 128 MB of memory

real-time applications is the computer game “Far Cry 2” which has procedural spread of fire through trees, grass and other flammable objects[Remo, 2008].

Simulators developed to predict fire spread for fire safety purposes often use several different models either choosing the most fitting for the specific case or combining them to account for different aspects of fire spread. CFAST[Peacock et al., 2008] is a model used to predict compartment fires. It is a “zone model”, meaning it partitions the space to be modeled into zones, like upper and lower gas layers, and uses physically based equations to calculate the development of the collection of zones. BEHAVE[Andrews and Bevins, 1999] models forest fire based on the semi-empirical mathematical model described by [Rothermel, 1972]. FARSITE[Finney and Andrews, 1999; Finney, 1998] uses, among other models, BEHAVE to produce maps of the fire growth that can be viewed in geographic information systems (GISs).

4 The Framework

The framework presented here is modeled after the one described in [Gundersen and Skjermo, 2009]. The additions to the work of Gundersen and Skjermo is mainly in exploring alternative simulation and visualization techniques and trying to elaborate on some specific practical issues. Both are still mainly at the conceptual level, but for this project there has been developed a prototype that implements a small part of the framework. The prototype is described in chapter 5, while this chapter details the framework as a whole. The main focus is on the spread of the fire but possible visual representations of the flames, smoke and trees are also discussed.

4.1 Framework Architecture

An important feature of Gundersen and Skjermo's framework is the use of two different levels of simulation for the fire spread: a coarse simulation governs the spread of fire through the whole forest, while close-up scenes of a small collection of trees use a separate, more detailed simulation. This is analogous to using "level of detail (LOD)" schemes when rendering complex 3D scenes: a popular class of techniques dating back to [Clark, 1976] that exploit the fact that objects that are distant in the scene are rendered as smaller because of perspective effects. Distant object can therefore be represented by simpler models to reduce computation cost. Although most often associated with rendering 3D scenes LOD schemes has also been suggested for simulations previously (e.g. [Carlson and Hodgins, 1997]).

The basic idea is that, just as for rendering 3D scenes, it is not necessary to simulate the fire spread over trees in the distance with the same precision as for trees close to the viewer. While a close-up view of a burning tree should provide a convincing animation of the fire spreading through the branches and leaves these details are not visible from a long range. As mentioned Gundersen and Skjermo suggest using two separate simulations each having their own grid, and using different spread models. The simulation responsible for the overall spread through the forest, as seen from a distance are by Gundersen and Skjermo called "large scale" simulation, and will here be called "forest-scale" simulation and is described in section 4.3. The detailed simulation of spread over a small collection of trees when observed at close range is handled by the "tree-scale" simulation, called "small-scale" by Gundersen and Skjermo. This is detailed in section 4.5. Both simulations can be accelerated by utilizing the highly parallel computing platform that modern programmable graphics cards provide, as demonstrated in the prototype described in section 5.

4.2 Simulation Synchronization

When animating a forest fire in a real-time interactive application in which the user can freely change perspective at any time it is important that the framework maintains continuity in the overall development of the forest fire. A few situations need special attention:

1. When a user changes from a point of view where a part of the forest is far away to a close-up view of that part of the forest, he will expect the general state of the trees in that section to be the same as when viewed from afar.

2. The reverse is also true: The trees seen in a close-up view need to be in the same state if moving the camera to a distant view.

3. Finally, the user will expect the fire to evolve also when not viewed.

To achieve this, Gundersen and Skjermo proposes to let the forest-scale simulation run continuously, even when not directly seen by the user. A tree-scale simulation is started only when there are a small number of trees in the user's immediate view and only when these trees are close enough that a detailed simulation is necessary. As soon as this group of trees exits the user's field of view the tree-scale simulation is discarded and it's state is forgotten. However, to account for the situations mentioned, it is important that the tree- and forest-scale simulations are kept in synchronization at all times. Gundersen and Skjermo suggests the following course of actions:

1. The user enters a close-up perspective of a group of trees. Tree-scale simulation is prepared for the nearest trees, but before the simulation starts, it queries the forest-scale simulation for the initial state of the trees.

2. The tree-scale and forest-scale simulations run in parallel. Gundersen and Skjermo do not mention if the two simulations are to be synchronized during this step but not doing so can result in apparent faults in the overall forest-spread. The synchronization can be done by making the forest-scale simulation periodically update it's data with data from the running tree-scale simulation.

3. When the user changes perspective away from the group of trees, the forest-scale simulation updates it's data with the last state of the tree-scale simulation.

As the forest-scale simulation does not hold any information about which parts of a tree is burned there is a problem of figuring out the exact state of the tree when starting a tree-scale simulation. One possible way to deal with this is to run a sped-up simulation before the close-up scene is rendered. This requires the forest-scale simulation to keep some information of the propagation history, such as from which direction each tree received most heat when it was ignited. It may very well be sufficient to assume the tree ignited at a random or predefined point.

4.3 Forest-scale Simulation

The forest-scale simulation keeps track of the overall progress of the fire. Gundersen and Skjermo suggest a CFD-based approach partitioning the forest into a discrete grid and solving the Navier-Stokes equations as described in section 3.2.1. This is however a relatively complex and costly model and several of the other fire spread models mentioned in section 3.2.2 should be just as suitable for this framework. For example should a method based on moving a fire front, such as [Beaudoin et al., 2001], be able to give a visually interesting and believable fire behavior if taking into account slope, wind and other factors. CFD-based approaches has previously been implemented on graphics hardware, some examples of which are mentioned in section 3. An example of a GPU-accelerated two-dimensional method can be found in the prototype implemented for this project, which is described in section 5. Although only the tree-scale simulation is implemented in the prototype the principle of executing the simulation on the GPU should be very similar for the forest-scale simulation.

As the forest-scale simulation must run continuously during the whole lifetime of the fire it should not induce a large computation overhead on the real-time application. A number of observations are done by Gundersen and Skjermo that validates the use of a fairly small simulation domain.

First, the speed of the propagation is found to be quite small compared to the size of the forest. The simulation can therefore use a large timestep and still keep the propagation moving steadily and smoothly as long as the model can handle it without becoming unstable.

Furthermore, for a forest fire scene at this scale the spread along the surface is the most interesting. Even if the surface is not perfectly planar, as long as the forest-floor does not self-overlap the forest-scale simulation can use a two-dimensional grid. Hight differences, which is an important factor affecting the spread due to buoyancy (c.f. section 2.2.3) can be accounted for using a hight map as suggested by Gundersen and Skjermo.

Forest scenes can contain thousands of trees so the resolution of the top-level simulation needs to be quite coarse. Figure 3a shows a forest fire seen from a distance. From this distance one can imagine it would be very difficult to make out the details of how the fire propagate in each tree, as the density of the trees makes it difficult even to distinguish between different trees. One can probably see the development from a barely burning tree to a fully developed fire to a burned down stump, but probably not pinpoint where the fire starts or whether it burns from left to right or from right to left. Therefore it should suffice to simulate the top-level spread on at most a per-tree basis.

Taking these factors into consideration Gundersen and Skjermo suggests using a regular 2D grid covering the area of the forest with each cell holding data such as fuel density, height, temperature and so on. This data can be stored in a set of textures with pixels' color values representing the data. Fuel- and height maps can easily be constructed from the landscape model. A suitable resolution for the grid will balance the realism of the spread versus the computational cost of the simulation. This is a representation very well suited for CFD-based simulation. If, however, other simulation methods are used different representations might be just as suitable. For example the prototype implemented for this project (see chapter 5) use an irregular, triangular mesh as it's simulation domain. In a similar manner, a mesh could be constructed for the forest-scale simulation in which each node represents a tree and the edges store distances between neighboring trees. A method similar to one of those used for the prototype could then be used to model the spread over this mesh. Other possible models such as the mentioned [Beaudoin et al., 2001] might not need a simulation grid at all, simply storing the data needed for each tree (or group of trees) in an indexed list.

4.4 Forest-scale Visualization

As a general method for supporting fire spread in 3D forest scenes, this framework should not put too hard restrictions on how the scene is rendered, neither for the trees or the fire at any scale. Still, the visual model must be compatible with the simulation model and in the forest-scale be able to draw a large amount of trees without dropping in framerate. In addition, it should be able to give a convincing visual representation reflecting the state of the tree according to the simulation. This means a dynamic representation is needed that changes appearance as the tree ignites, the fire spreads and finally extinguishes and leaves charred remains. Gundersen and Skjermo distinguishes between three states that a tree can be in: non-burning, burning and burned-out. Both the non-burning and burned-out state are static as far the fire simulation is concerned. The burning state should be a smooth transition between these two states. In addition, flames and smoke must of course be rendered.

4.4.1 Tree Rendering

Forest scenes with a large amount of trees typically use billboard representations of the trees, like in [Fuhrmann et al., 2005] and [Guerrero, 2006]. In the non-burning state the trees are of course rendered as usual. For the burned-out state, the visual representation can just be replaced by a charred version of the same tree, assuming the result of the burning is given in advance. In general the result of the burning is *not* given in advance, as the fire could extinguish at any point leaving the tree at any state between almost untouched by the fire to completely burned-down. Taking this into account the appearance of the burned-out state should if possible be dynamically chosen as the last appearance of the burning state. The visual representation in the burning state should change gradually as the fire devours the tree. It could be some kind of animation changing for example as a function of how much fuel is left in the corresponding simulation cell.

4.4.2 Flames and Smoke Rendering

There are many different methods for visualizing fire and smoke in 3D animations. The method chosen for fire should be able to represent different degrees of burning intensity as a slow barely burning fire, which may occur when spreading against the wind or downhill, can look significantly different from a violent, full-blown wildfire. Possible methods suggested by Gundersen and Skjermo are particle systems or physically based, two dimensional fire textures drawn on billboards like in [Gundersen et al., 2006].

Smoke can be simulated in a 3D grid using CFD-based techniques and be rendered using volume rendering techniques, as described in [Fedkiw et al., 2001]. This would require a simulation domain that can hold the whole smoke cloud, which can grow quite large for a forest fire, and thus be a computationally costly simulation to run. A much used technique for animating smoke in computer games is using a textured particle system. This can produce quite realistic smoke using relatively few, simple particles making it a computationally cheap choice.

4.5 Tree-scale Simulation

As for the forest-scale simulation there are a number methods to choose from for modeling in detail the fire spreading through a small collection of trees. The prototype implements a simple simulator for the spread through a single tree. A description and assessment of this is given in chapter 5. Gundersen and Skjermo suggest using a CFD-based model as in the forest-scale simulation, but using a finer and three dimensional simulation grid. There are, however, a number of factors complicating this simulation compared to at the forest-scale:

1. A tree is a much more complex structure than a landscape, wrapping around and overlapping itself making two-dimensional simulation meshes hard to fit to the geometry. The prototype creates a two-dimensional mesh over the surface of the tree, but the problem with this approach is that fire does not only spread along

the surface, but can jump directly between to spots that are far apart if measuring along the the surface but close in three dimensional space, like points at two different branches directly above each other.

2. Compared to viewing the forest fire at forest-scale a close-up animation of a few trees reveals a lot more details about the fire propagation. Factors such as what part of the tree that first ignites, which direction it spreads the fastest and how fast it spreads, burns down and extinguishes are now apparent to the user and needs to appear plausible.

3. In contrast to the spread velocity at forest-scale the velocity relative to the size of the simulation domain at this scale is much higher and a large time step might produce jerky progression of the fire front.

These factors suggest that a 3D grid in relatively high resolution would be a fitting simulation domain for the tree-scale simulation. Gundersen and Skjermo suggest enclosing the "main actor" tree, the one tree in the user's main focus, in a 3D simulation volume and simulating the fire spread over this using CFD-techniques. Again, this is a costly technique and alternatives can be found. As mentioned in point 1 in the list of factors above, the prototype uses a simple, two-dimensional grid that follows the tree's surface, with the drawback that fire can not take "shortcuts" through the air. Another possibility is to spread the fire using a particle system. Examples are found in [Smith and Freeman, 2008] and [Lee et al., 2001] where particles are shot from burning regions and start new fires if they collide with unburnt fuel. This requires collision checks between particles and fuel objects, which can amount to quite many and costly operations.

In their paper, Gundersen and Skjermo do not mention how the fire should spread between the trees in the user's view, which may be an important factor. Imagine, for example, watching two neighboring trees, one burning and about to ignite the other tree. You would probably expect the non-burning tree to catch on fire at a point near the burning tree first, not a random spot that could be further away from the heat. This means that the tree-scale simulation needs to know where the heat comes from and which point to ignite first. Some communication between neighboring trees is needed.

One possible solution of the problem of which part of the tree ignites first is as follows: Each tree has a list of it's closest neighbors (the ones it is most likely to catch fire from), the direction to each of these neighbors, the distance to it and the point of the tree that is closest to the neighbor. When a non-burning tree viewed in a close-up scene is about to ignite (the ignition time can be decided by the forest-scale simulation) it investigates it's closest neighbors to see which is giving off the most heat and ignites the point closest to that neighbor. To be as realistic as possible factors such as distance, wind and height-difference should be taken into account when finding the igniting neighbor. On the other hand would it probably be sufficient to approximate the temperature distribution of each of the neighboring trees with a single temperature for each tree located at it's base, geometric center or center of mass.

Another, more accurate solution could be to expand the simulation volume to include the neighbors. This can make the number of cells very high if the resolution is to be kept the same.

A third solution is again using a particle system. If particles are used for spreading the fire within a tree, it would only be natural to extend the scope of these to also ignite nearby trees. However, such methods could also be used in combination with other single-tree spread models, as done in [Lee et al., 2001] where the particles used in the particle based flame rendering doubles as ignition-particles that augment the surface spread method.

Fire spread through the foliage may need special treatment as it is often represented by billboards or billboard clouds. Few billboards with very simple geometry can represent a large amount of foliage and stretch over a significant part of the tree. Leaves are also very fine fuel and often burn more violently than wood, making the fire behave very differently here, and stressing the importance of finding a good spread simulation for the foliage. It may even be necessary to use a separate, third "foliage-scale" simulation domain, with close communication between it and the tree-scale simulation. A practical solution to this problem should be a priority for future work.

As already mentioned, there exists several examples of successful implementations of CFD-based simulations on the GPU, of which some examples are summarized in section 3.

4.6 Tree-scale Visualization

Also at tree-scale should the framework allow for different visual representations of trees and fire, but perhaps to a higher degree here than on the forest-scale visualization are there techniques that go better with some simulation models than other.

4.6.1 Tree Rendering

For rendering solids, like a tree, two possible representations that often use different rendering methods are volumetric representations and boundary representations. Volumetric representations, like voxel data sets fit very well with simulation methods using regular 3D simulation volumes as demonstrated in [Melek and Keyser, 2005]. Furthermore, changing the shape of a volumetric representation is often a relatively straight-forward operation, as shown in [Melek and Keyser, 2003] and [Riensche and

Lewis, 2009]. This is a great advantage if parts of the tree is to crumble or burn up. Boundary representations like the commonly used polygon meshes, on the other hand, might intuitively seem better suited for a 2D surface simulation grid, like the one used in the prototype. Still, they have also been successfully paired with 3D grids, for example by [Ishikawa et al., 2005]. It is, however, found necessary in several situations to use voxel-representations of objects inside a fluid simulation domain for the simulation, even if the rendering is done using polygon meshes. This requires at least two representations of the same object, as demonstrated in [Melek and Keyser, 2005] and [Keenan Crane and Tariq, 2007]. [Melek and Keyser, 2005] use this set of multiple representations among other things to support decomposition of the burning object. The prototype tries to accomplish a similar effect on a simple polygon mesh using a vertex displacement technique. This approach has some weaknesses however, as discussed in 5. The most commonly used high-detail representation in real-time 3D animated applications is using polygon meshes for the stems and largest branches of the trees, and rendering the foliage using billboard techniques.

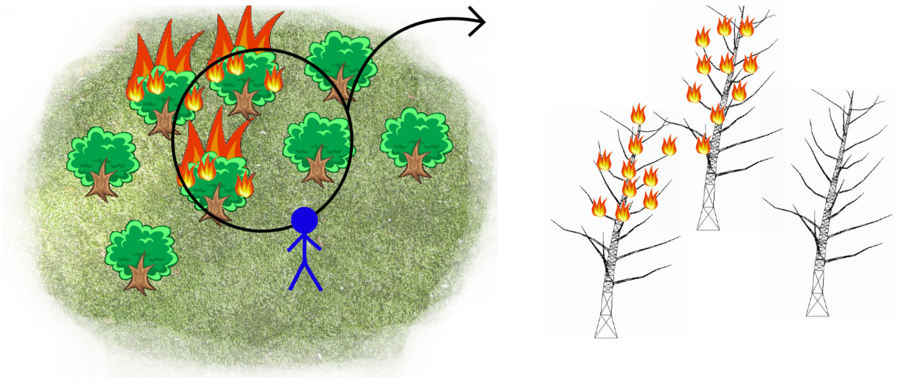
It is usual for real-time applications to use more than two detail levels when drawing trees. A popular method used in games is to create progressively simpler versions of each detailed model, as in [Chen et al., 2006]. Which model is used depends on how fare away the object is from the camera. This may present a problem for the fire spread simulation as the simulation must be able to continue the simulation between model switches, which can require costly conversion steps and data transfers between main- and GPU-memory. Another method, more commonly encountered in off-line rendered animations than in real-time applications is subdivision surfaces which achieves detailed models by applying a displacement map to a low resolution polygon models. A recent addition to graphics hardware introduces programmable hardware tessellation which can make this approach viable also for real-time applications [Loop et al., 2009; Boubekeur and Schlick, 2007]. As this is done entirely on the GPU it may be better suited for GPU accelerated fire spread simulation.

4.6.2 Flames and Smoke Rendering

On the rendering of flames and smoke many of the points mentioned in the forest-scale section also applies here. Flames and smoke can be animated using for example CFD-based techniques or particle systems.

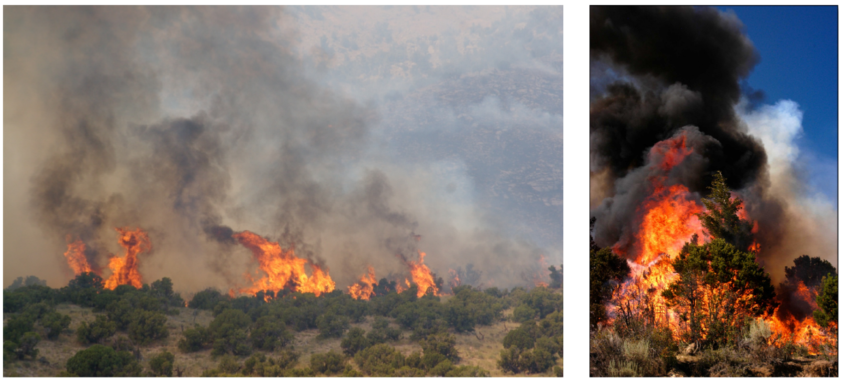
If using a CFD-based simulation on a 3D grid for animating the fire spread it might seem intuitive to simply render the flames directly using the data from this simulation. However, achieving a satisfactory visualization of the large, turbulent flames typically seen in forest fires demands quite high grid resolution, and other representations may be better suited for real-time applications. Gundersen and Skjermo suggests using procedurally created billboard flames, like those suggested for the forest-scale flame rendering. These can replace, or be combined with, the billboards used for the foliage if this technique is used. Fire spread through the foliage is, as mentioned, a feature that need some more consideration. Depending on the number and size of the foliage billboards a correct fire spread may need to be procedurally drawn directly on the billboard texture. If a large number of small billboards are used however, flames could be drawn on top of them while they fade away and vanish as they are devoured by the fire. Again, a practical solution needs to be researched.

Smoke which is a more stable and simpler phenomenon can do with a coarser grid than fire, but on the other hand may need a very large simulation domain, as mentioned in the forest-scale section. A particle system is again a viable choice.



(a) Forest-scale (b) Tree-scale

Figure 2: When the user changes perspective from an overview (2a) to a close-up (2b) a tree-scale simulation is started for the trees closest to the user (figure from [Kjærnet, 2009]).



(a) A distant forest fire (b) A close forest fire

Figure 3: Forest fires seen from a distant perspective and a close-up perspective. Photos by Jon Marshall with permission.

5 The Implemented Prototype

A prototype application implementing part of the framework has been developed to demonstrate and test some of the ideas presented in this report. The prototype implements a subset of the tree-scale simulation described in section 4.5 and the visualization techniques mentioned in section 4.6. The prototype simulates fire spreading over the surface of a single tree without foliage. The visualization of the fire is done by making the surface glow - no flames or smoke is rendered. All visualization and simulation is executed entirely on the GPU.

The development of the prototype was started as part of [Kjærnet, 2009] and has received several improvements during this project including a better spread model affected by wind, support for textured tree models and vertex displacement to simulate the fire devouring the tree. Two supporting applications have also been added that ease the tasks of debugging the simulation code and preparing the input 3D model data to make it compatible with the framework. Missing features that should be implemented in the future are discussed in section 7.4.2.

5.1 Prototype Architecture

The prototype consists of a vertex and a fragment shader written in OpenGL's shading language "GLSL", a simulator written in OpenCL, a framework and programming language designed to utilize parallel compute devices such as the GPU, and a base component in C++ to tie it all together and draw a simple tree animation on the screen. Three main tasks can be identified for the prototype, and for each task, a part of the application that has the main responsibility in solving it:

Initializing and controlling the application This is the task of the base component. This largest subtask here is to read the 3D model data from file and convert it to a format that is usable both for the spread simulation and the visualization. Other subtasks include setting up the environment for the OpenCL and OpenGL frameworks and running the main loop that executes the simulation and visualization steps.

Simulating the fire spread Calculations has to be done on the simulation domain to figure out how the fire spreads. This is done by the OpenCL kernels.

Rendering the frame The animation frame has to be rendered to screen. This includes the tree, the visual products of the fire such as flames and smoke and any surrounding objects in the scene. This is performed by the OpenGL shaders.

The program- and data-flow is illustrated in figure 4. The first that happens when running the prototype is, as mentioned, that the environments for the OpenCL and OpenGL frameworks are set up. Standard initialization steps are taken to locate the correct computational device for the OpenCL kernels to use, create a window for OpenGL to draw on and read the source code for the OpenCL kernels and the OpenGL shaders from file and compile them. Next, the data for the tree-model is read from file and converted to a collection of data to be used by the drawing routines and a collection of data used by the simulation. These are loaded into the appropriate OpenGL and OpenCL buffers once, after which no further slow data transfers between CPU- and GPU-memory are needed, with the exception of updating simulation and rendering parameters. At this point the base component enters a loop calling the simulation kernels and the shader program sending updated parameters such as the size of the current time-step or the wind direction for the simulation, or model- or view matrices when modifying the model or changing the viewer's perspective. The simulation kernels communicate directly with the shaders by writing to shared buffers, updating the values for vertex offsets and glow colors.

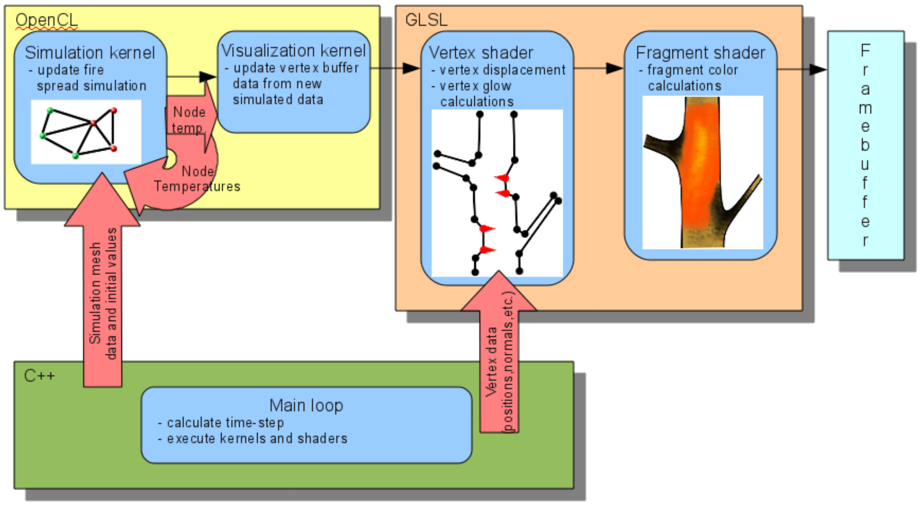

OpenCL
Simulation kernel
- update fire spread simulation
Visualization kernel
- update vertex buffer data from new simulated data
Node temp
Node Temperatures
GLSL
Vertex shader
- vertex displacement
- vertex glow calculations
Fragment shader
- fragment color calculations
Framebuffer
Simulation mesh data and initial values
C++
Main loop
- calculate time-step
- execute kernels and shaders
Vertex data (positions,normals,etc.)

Figure 4: Figure illustrating the program and data flow.

5.2 Prototype Spread Model

The prototype does not use the CFD-based suggested by [Gundersen and Skjermo, 2009] that encloses the tree model in a 3D voxel-grid simulation domain. Instead it constructs a 2D triangular mesh corresponding to the surface of the tree model. The model's vertex positions are used as nodes and the edges of the model's triangle surfaces as links connecting neighboring nodes in the mesh. This mesh is used as simulation domain for the fire spread, with nodes storing data such as temperature and remaining fuel and the links storing distances and directions between neighbors. An important aspect of the simulation mesh is that there can be only one node at each position, even if multiple vertices are occupying that position. This happens whenever vertices of neighboring faces differ in any other property than the position, usually in seams of geometry (as illustrated in figure 5) or of texture. The prototype relies on the 3D model to supply the radius of the tree at each vertex. This is used to decide how much fuel each nodes should start with, and how far a vertex can be displaced along the normal for the decomposition effect (c.f. section 5.3). Two models with this data included was provided by fellow student Jørgen Nystad who works on a parallel project developing an application for producing 3D tree models.

The simulation model used in the prototype in [Kjærnet, 2009] to describe the spread of the fire was a very simple intuitively derived model inspired by one-dimensional heat conduction. Temperature distribution over the simulation mesh was calculated by each node updating it's temperature based on the temperatures of it's neighbors weighted by their distance and a surrounding temperature. When a node reached a predetermined ignition temperature it would produce heat in an amount which was also predetermined. Fuel was stored for each node, initiated to the radius of the tree under that node and being consumed at a constant rate as long as the node was above the ignition temperature. The node stopped burning when the fuel was depleted. This model had several problems, one of the most serious being the representation of the tree as a connected set of discrete fuel sources. In a tree model with unevenly spaced nodes,

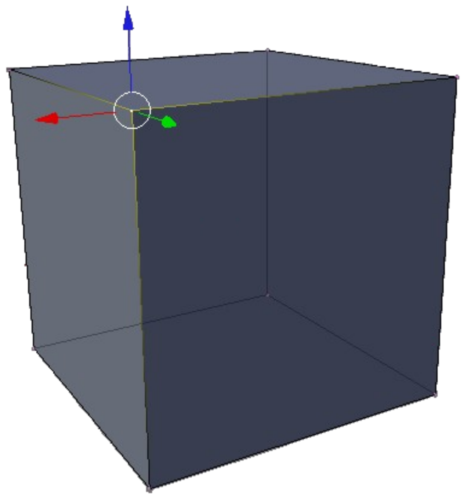


Figure 5: The encircled corner of the cube holds one node but three vertices with three different normals drawn as red blue and green arrows.

which is usual for any tree model having a
thick stem and thin branches, nodes that were relatively far away from it's neighbor nodes ended up not receiving enough heat to reach ignition temperature. This is unlikely behavior for a continuous fuel such as the tree the model tries to approximate. The only ways found to remedy this without changing the vertex data of the 3D-model was to increase the heat production of burning nodes or decrease the effect of the distance on the heat transfer. This gave the unwanted side effects of higher spread rates for respectively the whole tree or the parts with few nodes.

The new simulation model developed for this project is based on the one used in [Perry and Picard, 1994] which models the motion of the inception boundary (c.f. section 2.2) over a polyhedron. Instead of representing the inception boundary directly using a connected set of "spread control points" moving over the surface, as done in [Perry and Picard, 1994], the prototype tracks the movement of the boundary along the links of the simulation mesh. Each link can be seen as a fuse with it's length initialized to the length between it's endpoint nodes. In each time-step each node in the mesh not yet ignited checks it's neighbors to see if any of them have ignited, ignoring whether they are currently burning or have burned out. For each of it's ignited neighbors it decreases the length of the corresponding fuse according to the spread rate. When any of the fuses reach zero, the node ignites itself. The heat produced by burning node increases polynomially with the node's temperature as discussed in section 2.1. It is determined in the prototype by equation 5:

$$T_{prod,t} = c_{prod} max(T_{t-1}, T_{min}) * \Delta t \tag{5}$$

To give an easily controllable and predictable temperature behaviour the heat lost to the surroundings also increases polynomially, but with the difference between the node's temperature and the surrounding temperature according to equation 6:

$$T_{lost,t} = c_{lost}(T_{t-1} - T_{\infty})^2 * \Delta t \tag{6}$$

Finally, the temperature of the node is updated by the difference between the two

$$T_t = T_{prod,t} - T_{lost,t} \tag{7}$$

and the fuel is decreased by a constant factor

$$F_t = F_{t-1} - c_{fuel} * \Delta t \tag{8}$$

In equations 5–8 $T_{prod,t}$ denotes the heat produced in time-step t, $T_{lost,t}$ is the heat lost to the surroundings in, T_{t-1} is the temperature of the node in the previous time-step,

$T_{min} > 0$ is the minimum threshold for the node's temperature, T_∞ is the temperature of the surroundings, T_t is the temperature of the node after time-step t, F_t is the node's fuel after the current time-step, F_{t-1} is the node's fuel in the previous time-step, Δt is the length of the current time-step and c_{prod}, c_{lost} and c_{fuel} are predetermined coefficients for the heat production, heat loss and fuel loss.

The calculation of the spread rate is quite similar to that found in [Perry and Picard, 1994]. It is based on solving the fundamental equation of fire spread (equation 1) by approximating the heat flux using the notion of "dominant heat transfer mechanism" (see section 2.2.2). For opposed spread conductive heat transfer, for which the heat flux is given by equation 2, is approximated using:

$$q_o = \lambda(T_f - T_i)/L \tag{9}$$

where q_o is the heat flux for opposed spread, T_f is the flame temperature, T_i the temperature of the unburnt fuel, L is the fuel's thickness and λ is the thermal conductivity - for the gas if the fuel is thermally thin or for the fuel material if it is thermally thick. As explained in section 2.2.2 the thickness of a material is only considered important for thick solids *under* 1.5 mm in radius. In practice this means that for the purpose of modeling the spread over a tree, as seen in the relatively large perspective that this framework is intended for, the thickness L and the conductivity λ can be combined into a single material-dependent tuning parameter. In fact, Perry and Picard approximates the whole opposed spread heat flux with a constant, and that simplification is also adopted for this project's prototype. The dominant heat mechanism for concurrent spread is radiation, and the heat flux is then given by equation 4 and approximated in [Perry and Picard, 1994] with the equation

$$q_c = \epsilon_f \sigma_b T_f^4 h_f sin\Theta_f / L \tag{10}$$

with ϵ_f and σ_b being the emissivity of the flame and Boltzmann's constant, h_f is the height of the flame and Θ_f it's inclination angle relative to the surface. As the prototype do not model a flame the height and inclination of the flame is approximated using the wind vector's magnitude and orientation angle relative to the surface. This factor is found by calculating the dot product between the wind vector and the normalized direction vector for each link. It is updated each time the wind is changed and stored for use by the spread algorithm. Buoyancy is accounted for in the prototype by adding a vertical wind to the external wind vector.

There are done a couple of simplifications in the prototype compared to the model used in [Perry and Picard, 1994]. Firstly Perry and Picard models spread control points that move according to the spread rate perpendicular to the inception boundary. As the prototype does not model the boundary directly it's orientation is difficult to find and the spread rate is used without modification as the burning rate of the "fuses" in the simulation mesh. This gives an error in the fire's spread rate, but given enough links in the triangular simulation mesh it should give a reasonable approximation. The second simplification done is that the effect of curvature is ignored in the prototype. Intuitively the impact the curvature has on the spread over a tree is limited due to the following observations:

1. for the stem the curvature is often nearly constant for horizontal spread (around the tree) and close to zero for vertical spread (lengthwise). It seems probable that the effect of curvature on the spread over the stem only counteract the effect of buoyancy, and therefore can be accounted for by simply reducing the effect of buoyancy.

2. For the branches there is significant curvature around the branch, but in any case the circumference of the branch is usually so short that the fire spreads all the way around it virtually instantaneous Along the branch, the curvature is again almost zero. Branching-points has more interesting curvature features but only constitutes a small part of the tree model and problems with visualization in these areas (c.f. section 5.3) probably masks any inaccuracies here.

The implemented model can be called quasi-physically-based, as it is has it's foundation in the physically derived fundamental equation of fire spread, but ignores several important factors such as curvature and approximates several variables by constants among others flame geometry, and temperature. There are, however, given grounds for most of the simplifications made and some factors considered especially important, such as wind and slope, are accounted for. Most likely the single most serious drawback of the model is it's two-dimensional representation of a three-dimensional problem. This becomes most apparent when the tip of a burning branch is close to an unburnt branch. In reality the heat from the burning branch would likely have ignited the other branch, but the model can only spread the fire between nodes that are direct neighbors so the other branch will not be ignited before the spread front has reached it by moving over the surface.

The mesh is stored in memory as a set of node tables and a set of neighbor tables. The node tables are one-dimensional arrays each storing a property of the nodes such as temperature, remaining fuel and number of neighbors. The neighbor tables represent the links of the mesh and are two-dimensional arrays that contains a row for each node, in which properties of each of the node's links is stored. The rows of one of the neighbor tables store the other endpoint of each link, i.e. the node's neighbors, while other tables store the length and direction of these links. An illustrating example of the mesh's tables is shown in figure 6.

5.3 Prototype Fire Visualization

As mentioned, the main focus of this project is on the spread of the fire and the animation of flames or smoke has not been prioritized. The prototype therefore only use a simple temperature-dependent glow to visualize the spread of the fire. Possible methods for flame and smoke animation are mentioned in section 4.6.2. The glow color and intensity are functions of the node's temperature and are computed by an OpenCL kernel tasked with managing the visual representations. The fragment shader blends the color fetched from the texture with the glow color according to equation 11:

$$c = g * i + t * (1 - i) \tag{11}$$

where c is the output color, g is the glow color, t is the texture color and i is the glow intensity. The vertex shader creates a simple glowing effect by choosing the maximum value of the external light intensity and the glow intensity for any given vertex. A very similar method is used to visualize the charring of burned parts of the tree.

To visualize the decomposition of the tree as it burns a very basic vertex displacement technique is deployed. Along with calculating the glow color the visualization kernel calculates a displacement amount for each node based on how much of it's fuel has been spent. This is used by the vertex shader to displace the vertex a negative distance along it's normal. The effect of this can be seen in figure 10. One can also see some visual bugs that result from the displacement method not handling seams in the 3D model correctly. Wherever there are multiple vertices with different normals sharing a position in space, such as at the branching points and the branch tips, geometrical distortions can arise when displacing them along their normals. This is apparent on the screenshots as thick, black branching points. An improvement to the displacement method of the prototype should identify seams in advance and treat these with special care. Simply not displacing vertices that are part of a seam or for each group of vertices sharing a position randomly choose one of the vertices' normals, or a linear interpolation of all, for the displacement could provide better visual results, but this is a problem that needs more research.

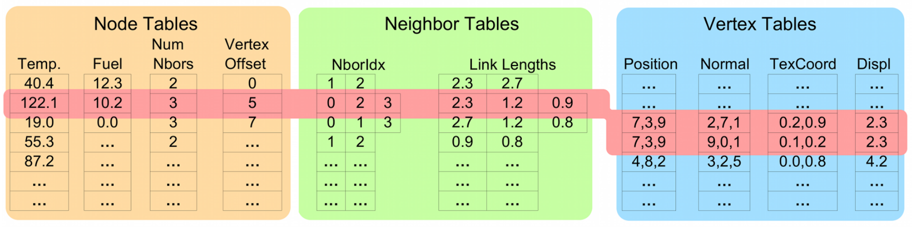
When an OpenCL kernel needs to communicate a change in a vertex' data, such as it's glow color or displacement to an OpenGL shader it uses a section of the GPU memory declared to be accessible for both OpenCL and OpenGL called a "shared buffer". The OpenCL visualization kernel writes glow color and displacement data to the shared buffer and the vertex shader use this data in it's color and displacement calculations. Because each node in the simulation mesh can correspond to more than one vertex some method is needed to match nodes to vertices to be able to alter vertex data based on the results from the simulation. As the prototype sorts the vertex data based on which node it corresponds to only the offsets to the first vertex of each such group is needed to identify which vertices are to be modified based on which node.

5.4 Supporting Applications

During the development the prototype a need arose for functionality not well suited for implementation in the prototype application. Two applications were made to provide supporting functionality for the development of the prototype.

Firstly, development of good spread algorithms was made difficult by the fact that they are executed on the GPU using OpenCL. Debugging facilities for OpenCL are still limited and there are few possibilities for generating feedback for programs running entirely on the GPU. Small errors in the program often ended up freezing the whole computer without generating any error message. For these reasons an application was developed to provide a platform for rapid prototyping and testing of spread algorithms. The application has a graphical user interface that lets the user change simulation parameters and initial values and run a step by step or continuous execution with immediate feedback of the simulation mesh's numeric values. Simulation algorithms can easily be changed or replaced and are run for each node in a way very similar to the OpenCL implementation only with each node being calculated sequentially rather than in parallel. This means it is not suited for debugging problems related to concurrent execution but it proved to be very helpful for several other problems.

The second supporting application is a script that processes the 3D model data to comply with the format needed by the prototype. This involves removing unused vertex positions and faces with no surface area. No visual changes are done to the model. Some processing is also done by the prototype itself, but to keep it as simple as possible it does certain assumptions about the 3D model.



Node Tables

Temp.	Fuel	Num Nbors	Vertex Offset
40.4	12.3	2	0
122.1	10.2	3	5
19.0	0.0	3	7
55.3	...	2	...
87.2	...	...	...
...	...	...	...
...	...	...	...

Neighbor Tables

NborIdx			Link Lengths		
1	2		2.3	2.7	
0	2	3	2.3	1.2	0.9
0	1	3	2.7	1.2	0.8
1	2		0.9	0.8	
...	...		...	...	
...	...		...	...	
...	...		...	...	

Vertex Tables

Position	Normal	TexCoord	Displ
...	...	...	...
...	...	...	...
7,3,9	2,7,1	0.2,0.9	2.3
7,3,9	9,0,1	0.1,0.2	2.3
4,8,2	3,2,5	0.0,0.8	4.2
...	...	...	...
...	...	...	...

Figure 6: Some of the tables representing the simulation mesh. From the node tables the example shows temperature, remaining fuel, number of neighbors in the mesh and the node's offset into the vertex table. The neighbor tables shown are the tables holding the indices of the node's neighbors and the lengths of the links to those. Vertex tables are exemplified by position-, normal- and texture coordinates and the vertex displacement distance. The lines marked in red are the data belonging to the node with index 1. The vertex data corresponding to the node is found by using the vertex offset table.

6 Results

To analyze the visual realism achieved in the prototype and the computational overhead it might impose on a real-time application, this chapter presents some tabulated performance results and a series of screenshots from running the application.

6.1 Performance Results

The performance results are from a very simple animation of one tree slowly spinning around it's vertical axis. The simulation is updated before each frame of the simulation, and the simulation update frequency and the draw frequency is therefore the same.

The application was tested on the following hardware:

Processor: Intel Core 2 Duo 3.0 GHz with 4 GB DDR RAM main memory.
Graphics card: AMD Radeon HD 4850 with 800 stream processing units and 512 MB RAM.

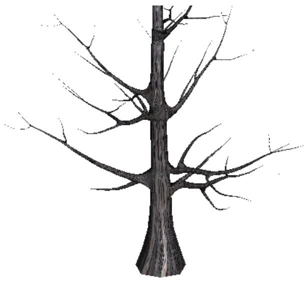
The simulation was tested on two tree models: A high-resolution model with 22,369 nodes, 23,031 vertices and 44,728 faces and a low-resolution model with 541 nodes, 604 vertices and 1,076 faces. Frame rates of 60 and 829 FPS was achieved when using the high-resolution and low-resolution model respectively in the animation. The results are summarized in table 1.

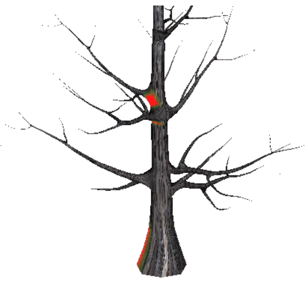
Table 1: Tabulated performance results for the two tree models.

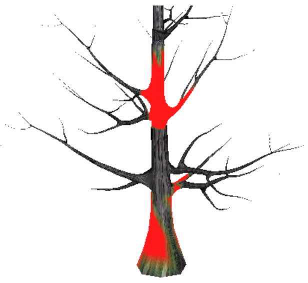
Model	Vertices	Faces	Nodes	FPS min - max	FPS avg.
Low-resolution	604	1,076	541	522 - 883	829
High-resolution	23,031	44,728	22,369	50 - 72	60

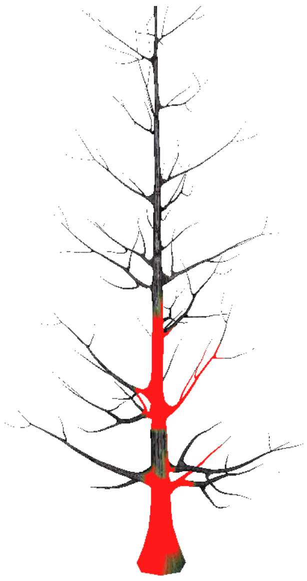
6.2 Visual Results

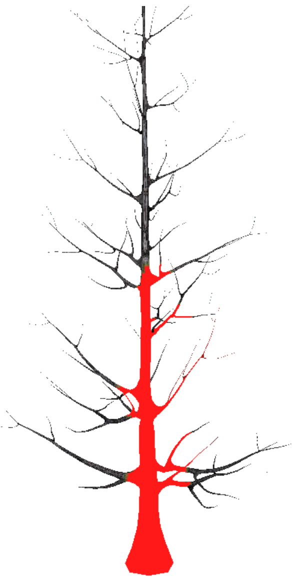
Figures 7 and 8 shows two sequences of screenshots taken from the animation of the high-resolution tree model. In this animation the tree does not spin to better show the progress of the fire spread. Figure 9 shows a sequence from using the low-resolution model in the animation, and figure 10 shows a detailed view of a branching point after it has burned out.


(a) t=0


(b) t=2


(c) t=4


(d) t=6


(e) t=8

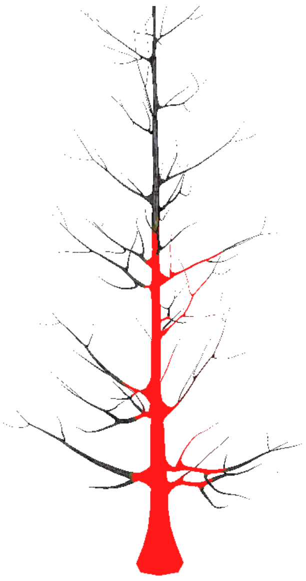
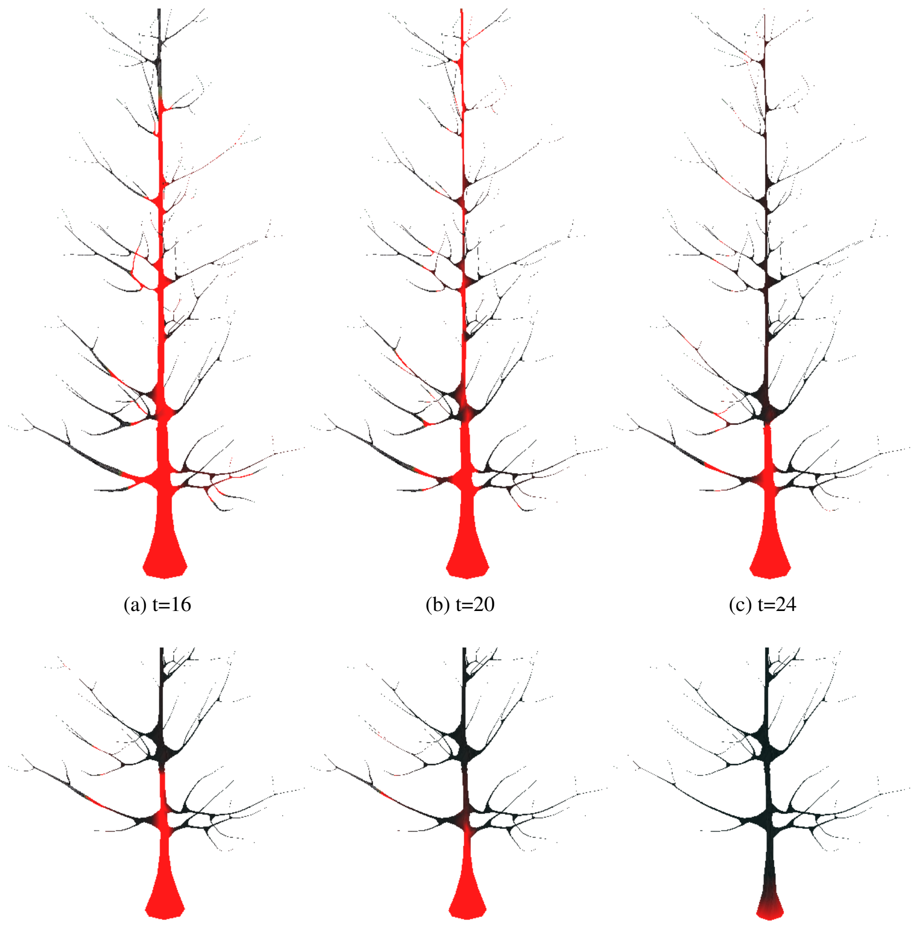

(f) t=10

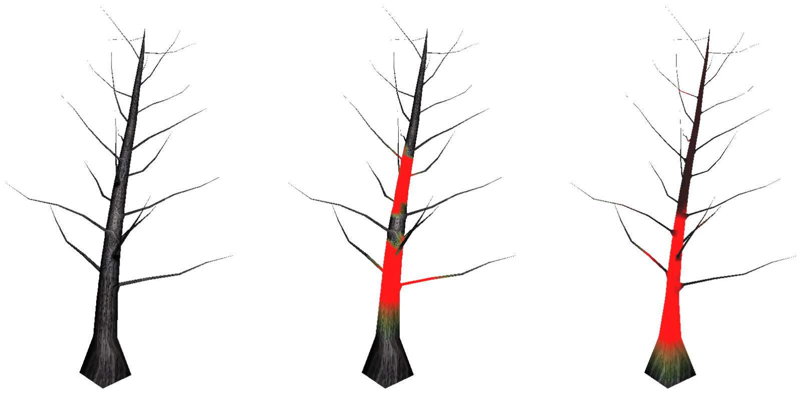
Figure 7: Screenshots of the tree animation using the detailed tree model. The shots are taken two seconds apart starting at t = 0 seconds. The sequence is continued in figure 8



(a) t=16 (b) t=20 (c) t=24

(d) t=28 (e) t=32 (f) t=56

Figure 8: Screenshots of the tree animation using the detailed tree model. The shots are taken four seconds apart starting at t = 16 seconds, with the exception of the last which is taken at t = 56.



(a) t=0 (b) t=6 (c) t=20

Figure 9: Screenshots of the tree animation using the simple tree model. The shots are not equally distributed over time.

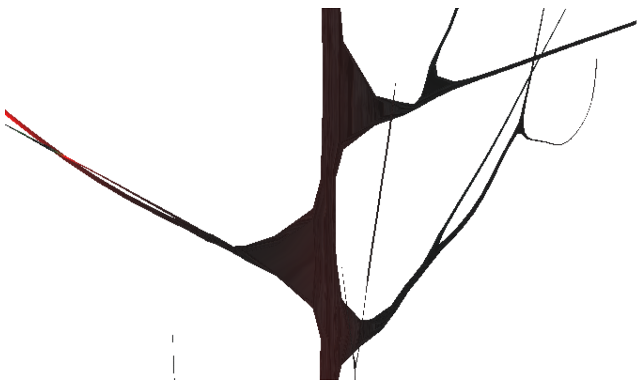


Figure 10: Detailed view of a burned branching.

7 Conclusion

A framework for animating forest fires in real-time has been described and several alternative implementation details have been discussed. Some potential difficulties have been identified, and solutions have been suggested for some of them. Most of the framework is still on a conceptual level, but a small part has been implemented and tested in a prototype application. The success of the project in achieving it's goals can be measured along three axis according to the goals set up in section 1.1: the computational overhead the framework imposes on the application, or it's *performance*, the perceived degree of *realism* of the fire spread and the *suitability* of the framework for integration in a real-time application.

7.1 Performance

The total computational overhead imposed by the framework is difficult to analyze without a working implementation. Several different simulation and visualization methods are discussed and only a small subset have been implemented and tested in the prototype. However, this subset should give a rough pointer of the kind of performance to be expected from the framework.

The methods chosen to be implemented in the prototype was a simple 2D spread simulation model and a few basic visualization techniques such as a primitive vertex displacement and programmed texture and lightning.

The frame rate achieved on the animation was very high for the low-resolution tree model (829 FPS) but dropped significantly for the high-resolution model (60 FPS). It is still very difficult to perceive any lag at these frame rates, but keeping in mind that these results were achieved on a very simple animation a somewhat higher frame rate may be desired for the high-resolution model if it were to be used in a more complicated scene. A couple of changes to the simulation can, be suggested that may contribute to improve the performance.

Firstly, to thoroughly test the overhead of the simulation on the application it was executed before every frame of the animation. In practice it would probably suffice to update the spread simulation far more infrequent than this. If run on a screen with a refresh rate of 200 Hz, indeed a very high rate in today's standard, changes in the animation occurring less than 5 milliseconds apart would not even be drawn on the screen. Given the maximum spread speed achieved in the simulation and the scale of the animation one could probably find a minimum simulation frequency to give a smooth fire progression. However, the simulation method might impose stricter constraints on the size of the time-step. For example, the method used in the prototype can not advance the spread front more than one link at a time. Time-steps that results in fire propagation distances longer than the shortest of the links should be avoided.

Secondly, the optimization of the simulation algorithm for efficiency and memory usage has not received much focus in this project. Memory alignment and efficiency improvements on the calculations can have a big impact on performance on parallel executions, especially on the GPU with it's massively parallel architecture and limited memory space.

7.2 Realism

Again, the evaluation of the realism of the framework will have to rely on the impression given by the small part implemented in the prototype. Furthermore, perceived realism is difficult to measure objectively. Still, some measure can be given by considering the number of important affecting factors taken into account versus those ignored by the simulation.

There is undoubtedly one major shortcoming in the simulation model regarding the realism of the fire's behavior: the use of a two-dimensional simulation domain to model a three-dimensional phenomenon. Because the heat transfer through air can be substantial a model using only the two dimensional object-surface as simulation domain will never behave very realistically. To capture this effect the simulation must be done on a three dimensional grid enclosing the burning object, or other spread simulation methods must be used, like Intel's procedural fire spread with "heat particles" emitted from hot places and traveling through space. However, both the use of a full 3D simulation domain and the high number of collision checks needed for use of heat particles can require too high computational costs to be suited for real-time applications. [Lee et al., 2001], which uses a spread model very similar to the one implemented in the prototype, solves this by creating a distance field surrounding the burning object in advance and using this to check for collisions between fire particles and unburnt fuel. As the implemented model uses the object's vertices as the simulation mesh, it might suffice to store the distances to some nearby vertices in effect creating a mesh with links not only following the object surface but also directly through the air. It might be necessary to treat the air-links different from the surface-links, as the surface-links represent continuous fuel and should burn steadily without being extinguished while the air-links more closely should mimic the heat transfer between discrete fuel.

On the other hand, as the model does account for the effect by wind and gravity on the fire spread the movement of the spread front is not uninteresting, and might very well be perceived as plausible by a viewer with limited experience with actual forest fires. With a fast spreading fire with violent flames and large amounts of smoke the inaccuracies of the fire propagation would likely be masked by all the other elements drawing focus.

The visualization of the fire is another field where the prototype obviously lack in realism. The complete lack of flames and smoke is unacceptable for any fire animation. This is, however, phenomena that are well known in the computer graphics society and there exists a multitude of potential methods that can provide this for the framework, and some of these have been discussed in section 5.3. Nevertheless, some interesting visual features have been implemented in the prototype, including fuel decomposition, glow and charring.

7.3 Suitability

The third group of goals were related to the suitability of the described techniques for integration into existing real-time applications. This is difficult to say much about without detailed knowledge of the design of any existing real-time applications. There are open source applications available that can provide suitable test platforms for integration with the framework. This is further discussed in section 7.4.

An effort has been made to rely only on commonly used techniques for visualizing the forest and the fire. However, some phenomena, such as the decomposition of a burning tree, is not immediately compatible with normal techniques. The prototype does use a very common object representation, and simulates decomposition with a certain degree of success, although several features are missing. There are, for example, not implemented a method for displaying foliage.

7.4 Future Work

The final verdict must be that although the described framework may very well be a practical solution to provide spreading forest fire to real-time applications and some of the features implemented in the prototype seem promising, with so much of the framework still to be implemented and tested there are still many unknown factors. Several aspects need more research and much work remains to provide a fully working fire spread framework.

7.4.1 Research

Some research is still to be desired in some of the problem areas.

The synchronization between the forest-scale and the tree-scale is a difficult aspect. A method to initialize the tree-scale simulation to an in-progress fire from the data available in the forest-scale simulation is yet to be found.

Level-of-detail schemes present another difficulty when simulating fire spread. Either should a way be found to make the suggested simulation methods compatible with existing LOD schemes, or new good LOD schemes that work well with the simulation should be investigated. A possible candidate is the mentioned subdivision surfaces approach.

The burning of foliage is one of the areas that have been somewhat neglected in this project. Both methods for simulating realistic fire propagation and for visualizing the fire are needed. This is also a very visible feature of forest fire, and finding solutions to these problems should be of high priority in further work on this framework.

7.4.2 Implementation

To be complete the framework should also be implemented in it's entirety and integrated into a real-time application for testing and demonstration. Clearly a lot of work remains on this prototype implementation.

The implementation of the forest-scale simulation and visualization is not even started, and the implemented prototype only scratches the surface of the tree-scale part. Spread between neighboring trees in a tree-scale scene, foliage fire spread and spread through the air are just some of the features missing from the tree-scale simulation. The visualization lacks any flame, smoke or foliage representation and the glow and tree decomposition are still unfinished.

A full-scale forest fire framework need all these parts in place and a method to synchronize the two scales. If it is to be made use of it should also demonstrate how to integrate into an existing real-time application. A candidate real-time application to test integration against is the "Object-oriented Graphics Rendering Engine", or OGRE, used among other things to demonstrate an LOD-scheme for forest rendering in [Guerrero, 2006].

References

Andrews, P. and Bevins, C. (1999). BEHAVE Fire Modeling System: Redesign and Expansion. *Fire Management Notes*, 59(2):16–19.

Anonymous (2010). combustion. http://www.britannica.com/EBchecked/topic/127367/combustion (visited May 11th 2010).

Beaudoin, P., Paquet, S., Poulin, P., and Graphics, M. (2001). Realistic and Controllable fire Simulation. *Proceedings: Graphics Interface 2001: Ottawa, Ontario, Canada, 7-9 June 2001*, page 159.

Beever, P. (2008). fire dynamics. http://www.britannica.com/bps/additionalcontent/18/30035543/fire-dynamics (visited May 11th 2010).

Boubekeur, T. and Schlick, C. (2007). A Flexible Kernel for Adaptive Mesh Refinement on GPU. In *Computer Graphics Forum*, volume 27, pages 102–113. John Wiley & Sons.

Carlson, D. A. and Hodgins, J. K. (1997). Simulation levels of detail for real-time animation. In *Proceedings of the conference on Graphics interface '97*, pages 1–8, Toronto, Ont., Canada, Canada. Canadian Information Processing Society.

Chen, H., Fahn, C., Tsai, J., Chen, R., and Lin, M. (2006). Generating high-quality discrete LOD meshes for 3D computer games in linear time. *Multimedia Systems*, 11(5):480–494.

Chen, S. and Doolen, G. (1998). Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30(1):329–364.

Chiba, N., Muraoka, K., Takahashi, H., and Miura, M. (1994). Two-dimensional visual simulation of flames, smoke and the spread of fire. *The Journal of Visualization and Computer Animation*, 5(1):37–53.

Clark, J. H. (1976). Hierarchical geometric models for visible surface algorithms. *Commun. ACM*, 19(10):547–554.

Crawfis, R. A. and Max, N. (1993). Texture splats for 3d scalar and vector field visualization. In *VIS '93: Proceedings of the 4th conference on Visualization '93*, pages 261–266, Washington, DC, USA. IEEE Computer Society.

Drysdale, D. (1998). *An introduction to fire dynamics*. John Wiley & Sons Inc.

Fedkiw, R., Stam, J., and Jensen, H. (2001). Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM New York, NY, USA.

Fernandez-Pello, A. and Hirano, T. (1982). Controlling Mechanisms of Flame Spread. *Fire Science and Technology*, 2(1):17–54.

Finney, M. (1998). FARSITE: Fire Area Simulator—model development and evaluation. *Evaluation*.

Finney, M. and Andrews, P. (1999). FARSITE—A Program for Fire Growth Simulation. *Fire Management Notes*, 59(2):13–15.

Foster, N. and Metaxas, D. (1997). Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 181–188, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Fuhrmann, A., Umlauf, E., and Mantler, S. (2005). Extreme model simplification for forest rendering. *Natural Phenomena*, pages 57–66.

Griffiths, J. and Barnard, J. (1995). *Flame and combustion*. CRC.

Guerrero, P. (2006). Rendering of Forest Scenes. *Technische Universität Wien, Institut für Computergraphik und Algorithmen*.

Gundersen, O., Rødal, S., and Storli, G. (2006). Realistic 2D Fire in Real-Time.

Gundersen, O. E. and Skjermo, J. (2009). A framework for physically based forest fire animation. In *Theory and Practice of Computer Graphics 2009*, Eurographics UK Chapter Proccedings. Cardiff University, UK.

Harris, M. J., Baxter, W. V., Scheuermann, T., and Lastra, A. (2003). Simulation of cloud dynamics on graphics hardware. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 92–101, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Ishikawa, T., Miyazaki, R., Dobashi, Y., and Nishita, T. (2005). Visual Simulation of Spreading Fire. *NICOGRAPH Internation*, 5:43–48.

Kass, M. and Miller, G. (1990). Rapid, stable fluid dynamics for computer graphics. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 49–57, New York, NY, USA. ACM.

Keenan Crane, I. L. and Tariq, S. (2007). *Real-Time Simulation and Rendering of 3D Fluids*, pages 633–675. Addison-Wesley Professional.

Kipfer, P., Segal, M., and Westermann, R. (2004). Uberflow: a gpu-based particle engine. In *HWWS '04: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 115–122, New York, NY, USA. ACM.

Kjærnet, Ø. (2009). Real-time forest fire on the gpu.

Latta, L. (2005). Massively parallel particle systems on the gpu. *ShaderX3: advanced rendering with DirectX and OpenGL*, page 119.

Lee, H., Kim, L., Meyer, M., and Desbrun, M. (2001). Meshes on fire. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, page 84. Springer-Verlag New York, Inc.

Loop, C., Schaefer, S., Ni, T., and Casta no, I. (2009). Approximating subdivision surfaces with gregory patches for hardware tessellation. *ACM Trans. Graph.*, 28(5):1–9.

Lott, N. and Ross, T. (2006). Tracking and evaluating US billion dollar weather disasters, 1980-2005. *NOAA's National Climactic Data Center, Asheville, North Carolina*.

Melek, Z. and Keyser, J. (2002). Interactive Simulation of Fire. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*. IEEE Computer Society Washington, DC, USA.

Melek, Z. and Keyser, J. (2003). Interactive simulation of burning objects. pages 462 – 466.

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*, page 196. ACM.

Nguyen, D., Fedkiw, R., and Jensen, H. (2002). Physically based modeling and animation of fire. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 721–728. ACM New York, NY, USA.

Norton, T. and Sun, D.-W. (2006). Computational fluid dynamics (cfd) - an effective and efficient design and analysis tool for the food industry: A review. *Trends in Food Science & Technology*, 17(11):600 – 620.

Peacock, R., Jones, W., Reneke, P., and Forney, G. (2008). CFAST–Consolidated Model of Fire Growth and Smoke Transport (Version 6). *NIST Special Publication*, 1041:103.

Perlin, K. (1985). An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296.

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

Perry, C. and Picard, R. (1994). Synthesizing flames and their spread. *SIGGRAPH 94 Technical Sketches Notes*.

Pettersen, R. (1984). The chemical composition of wood. *Advances in chemistry series*, (207):57–126.

Price, J. (2006). Lagrangian and eulerian representations of fluid flow: Kinematics and the equations of motion.

Pyne, S., Andrews, P., and Laven, R. (1996). *Introduction to wildland fire*. John Wiley & Sons Inc.

Quintiere, J. (2006). *Fundamentals of fire phenomena*. John Wiley & Sons Inc.

Reeves, W. (1983). Particle systems—a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics (TOG)*, 2(2):108.

Reif, F. and Reif, F. (1965). *Fundamentals of statistical and thermal physics*. McGraw-Hill New York.

Remo, C. (2008). Interview: How far cry 2's fire fuels, spreads. http://www.gamasutra.com/php-bin/news_index.php?story=20901 (visited December 15th 2009).

Riensche, R. and Lewis, R. (2009). Modeling and Rendering Physically-Based Wood Combustion. In *Proceedings of the 5th International Symposium on Advances in Visual Computing: Part I*, pages 896–905. Springer-Verlag.

Rothermel, R. (1972). A mathematical model for predicting fire spread in wildland fuels. *USDA Forest Service Research Paper INT (USA)*.

Rødal, S., Storli, G., and Gundersen, O. (2006). Physically based simulation and visualization of fire in real-time using the GPU. *Theory and practice of computer graphics 2006: Eurographics UK Chapter proceedings*, pages 13–20.

Selle, A., Fedkiw, R., Kim, B., Liu, Y., and Rossignac, J. (2008). An unconditionally stable maccormack method. *J. Sci. Comput.*, 35(2-3):350–371.

Smith, H. and Freeman, J. (2008). An Overview of Procedural fire. `http://software.intel.com/en-us/articles/an-overview-of-procedural-fire/` (visited December 3rd 2009).

Stam, J. (1999). Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.

Stam, J. (2000). Interacting with smoke and fire in real time. *Communications of the ACM*, 43(7):76–83.

Stam, J. and Fiume, E. (1993). Turbulent wind fields for gaseous phenomena. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 369–376, New York, NY, USA. ACM.

Stam, J. and Fiume, E. (1995). Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 129–136, New York, NY, USA. ACM.

Steinhoff, J. and Underhill, D. (1994). Modification of the Euler equations for "vorticity confinement": Application to the computation of interacting vortex rings. *Physics of Fluids*, 6(8):2738–2744.

Strahle, W. (1993). *An introduction to combustion*. CRC.

Taft, J. (2000). Performance of the OVERFLOWMLP CFD Code on the NASA/Ames 512-CPU Origin System.

Van der Burg, J. (2000). Building an advanced particle system. *Game Developer Magazine*, 3.

Viegas, D. (1998). Forest fire propagation. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, pages 2907–2928.

Wagner, A. J. (2008). A Practical Introduction to the Lattice Boltzmann Method. *Adt. notes for Statistical Mechanics*, 463:663.

Wei, X., Li, W., Mueller, K., and Kaufman, A. (2002). Simulating fire with texture splats. *IEEE Visualization, 2002. VIS 2002*, pages 227–234.

Williams, F. (1977). Mechanisms of fire spread. In *Symposium (International) on Combustion*, volume 16, pages 1281–1294. Elsevier.

Yaeger, L., Upson, C., and Myers, R. (1986). Combining physical and visual simulation—creation of the planet jupiter for the film "2010". *SIGGRAPH Comput. Graph.*, 20(4):85–93.

Youquan, L., Xue-hui, L., Hong-bin, Z., and Wu, E. (2005). Physically Based Fluid Simulation in Computer Animation. *JOURNAL OF COMPUTER AIDED DESIGN AND COMPUTER GRAPHICS*, 17(12):2581.

Zhao, Y., Wei, X., Fan, Z., Kaufman, A., and Qin, H. (2003). Voxels on fire. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 36. IEEE Computer Society.