Czech Technical University in Prague
Faculty of Electrical Engineering
Computer Science and Engineering Department

Real-time Visualization Techniques for Modelling of Combustion and Fluids

by

Marek Gayer

A thesis submitted to
the Faculty of Electrical Engineering, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

Doctoral study programme: Electrical Engineering and Informatics
Study Branch: Informatics and Computer Science

November 2005

Thesis supervisor:

Prof. Ing. Pavel Slavík, CSc.
Department of Computer Science and Engineering
Czech Technical University in Prague
Karlovo nám. 13
121 35 Prague 2
Czech Republic

Abstract

Nowadays, complex simulation programs are used for modelling of combustion processes in power plant boilers. They allow designers to experiment extensively with the computer models of the boilers without necessity to build their physical models. These simulation methods are widely used and several program packages (e.g. FLUENT) are available on the market. The use of these systems led to increased quality of the design and their use has been widely adopted by designers around the world. General disadvantage of these simulation programs is the complexity of simulation, which results in very time-consuming calculations. This situation does not allow quickly investigate various conditions in the boiler and the results of parameter changes during the combustion process nor study in the real-time the dynamics of the combustion process.

In the thesis, we present techniques, which together form a solution allowing real-time simulation and visualization of these processes. We propose a solution based on simplified fluid flow and combustion process computation. We use structured cell grid representation of the combustion space. We use fast fluid simulation, based on Euler equation and continuity equation. Thus, for the specified time step, we compute changes of the mass flow pressure, velocity and other characteristics inside each cell. The changes of the cell characteristic are also based on the characteristics of the nearest neighbours of the cell. The results of the flow simulation are immediately used for the other parts of the computation.

The concept of the virtual coal particles, based on the interaction of the air mass inside the cell with the coal particles, allows fast combustion simulation. It also gives possibility of easy tracking of the flow of combustibles and allows visualizing dynamics of the entire combustion process.

We also describe our concept of pre-calculated Fluid Simulator States. It is an extension for structured fluid simulators and solvers, which are used widely in computer graphics and simulation applications. It is based on storing pre-calculated Fluid Simulator States (FSS). The simulation using our extension is based on partial computation with synchronous utilization of pre-calculated Fluid Simulator States stored on disk drives. The disk space requirements are less demanding by several orders than the ones needed for saving corresponding unsteady data sets. This allows better scalability and storing and replaying results of complex tasks with large grids and/or ten thousands of particles.

We organize pre-calculated Fluid Simulator States (FSS) in hierarchical tree structures allowing incremental solving, interactive replaying and modifications of the simulated tasks. Thus, the parameters and boundary conditions of the simulation can be modified in real-time during replaying. We proposed also hierarchical tree structures with Unsteady (time-varying) Datasets, which brings interesting interactive features to applications using these datasets.

We have also designed and implemented visualization of the computed results from our fluid simulator. We use the OpenGL graphics library for fast and portable visualization of our results, including fast visualization of virtual coal particle system and high quality grid cell visualization using

interpolation using bicubic splines. For that purpose, we developed hardware-accelerated visualization using the current generation of graphics accelerators. The presented method offers real-time rendering of grid-structured data. The method uses bicubic spline interpolation and uses graphics hardware ability to map numeric values to a texture. It allows us also to render isolines of the visualized data and dynamically change the visualization parameters. We have compared the visual quality of produced images with the commonly used linear interpolation visualization methods.

The implementation of the above-described techniques has been used to develop the application *My Pulverized Coal Combustion*, which allows real-time interaction with the boiler model – e.g. allows changing up the inlet characteristic, such as mass flow of the air and coal, velocity of the flow, positions of the inlet without need to restart the simulation. A common, low-cost hardware was used to perform all computations and visualization. Overall, the solution allows real-time, immediate interaction of the user with the model during simulation and visualization – e.g. changing coal inlets, combustible properties and other input parameters during simulation. The solution allows real-time monitoring of about 50 basic cell volumes characteristics and statistics inside the boiler, and about 10 pulverized coal particle characteristics. All these features are available immediately, without need to wait hours for complex calculations to finish. Our solution is especially suitable for education purposes in power engineering.

The concept of our fluid simulator and the other techniques are implemented in 2D. They may be used in general fluids computation and modelling, thus not only for the combustion applications. The components have been designed as independent blocks, which could reusable in other projects.

Keywords: Combustion, Visualization, Particle systems, Simulation, Unsteady datasets, Hardware acceleration, Education

Acknowledgment

I would like to want thank to my supervisor Pavel Slavík. I especially value his experience, management abilities and sense of distinguish what is and what is not important during research work. He was very helpful and friendly in many situations and problems, which I have encountered during the times of my study. I am glad I could co-worked with such a person, who does so much for his students and research projects.

I would like to thank to František Hrdlička for his introduction into complex problems of combustion. His advices were very valuable during the development of the combustion model developed in the framework of this thesis.

I want to thank Petr Kadlec for implementation of visualization library VISLA, which allows visualization of some of visualization techniques described in the thesis.

I want to thank Michal Fořtík for sharing his experiences in programming techniques for combustion models with me, which was important for me namely in beginning of my work.

I want to thank Adam Sporka for language corrections in this thesis.

I want to thank my colleagues and friends from computer graphics laboratory, namely Jiří Bittner, Jiří Chludil, Martin Klíma, Michal Máša, Zdeněk Míkovec, Jaroslav Sloup, Adam Sporka, Roman Ženka, and Pavel Žikovský for either their support, knowledge or sense of humour. I want to thank to people from the Technical University of Crete, Environmental Engineering department, namely Vassilis Gekas, Kaliope Balta, Anna Patsioura, Carloss Serra, George Triantafyllou, Argiro Voutetaki, and also Dimitra Havre from International Relations Office, for being great hosts during my research visit in Chania, Greece.

I want to thank the head of our department Josef Kolář for taking care of my financial support during my studies.

Last but certainly not least, I want to thank all my friends and my family for their constant support and encouragement, and for their patience during the period of writing up the thesis.

Dedication

To all the people who truly effort in various ways to make this world a better place

Contents

The list of figures

The list of tables

1 Introduction and motivation

1.1 Scope of our work

The goal of this thesis is to find a way to present and propose some original techniques and possibilities for interactive, real-time model of combustion processes, which will allow immediate, interactive visualization. The solution should be as general as possible to allow also parts of our work to be applicable and reusable in general fluid tasks and projects. The further text summarizes this effort. In this and the next chapter, we are presenting the reader general introduction of basics of modelling of fluids and some visualization methods as well.

1.2 The economical and ecological reasons

In recent years, the world's attention has focused increasingly on our use of energy as well as technologies for energy production available, and consequences of their use. Combustion of fossil fuels causes major problems of pollution of the global ecosystem (see Fig. 1-1). The energy resources are limited and therefore we should try not to waste them in both industry and our homes.

Figure 1-1: Illustrative photos of the today's world major problem: pollution

Today's coal power plants also cause pollution problems. The heart part inside the coal power plant is combusting boiler. A common goal is to improve design of boilers to reduce pollution, find optimal ways of preparing fuel, determine coal particle sizes and quantity, speed of combustion etc. Not only the ecological reasons, but also economical reasons (effective combustion) are important for boiler designers.

In engineering practice, it is very difficult to investigate the combustion processes of various kinds of combustibles directly in the boiler. Modelling of fluid based physical processes such as coal combustion processes is a very time consuming activity accompanied by immense financial costs. The traditional approach is based on theoretical boiler design and verification of the design on the real model of the boiler designed. The tests performed on the boiler are used for modification of the design. This cycle (design-verification-modification) is repeated until the quality of the design reaches some satisfactory level.

In last decades, this process has been accelerated considerably by using complex simulation applications. They allow the designer to experiment extensively with the computer model of the boiler designed without necessity to build a physical model of the boiler. These applications are widely used and several program packages (e.g. FLUENT [FluentInc-WWW]) are available on the market. The use of these systems led to increased quality of the design and their use has been widely adopted by designers around the World. Efficient boiler design is based on simulation accompanied by optimal visualization models.

1.3 Visualization and simulation of the problem

Firstly, simulation allows appropriate computation of most characteristics, that are the most important for designers (such as global and local temperature, pressures, velocities of combustibles and air, NOX concentrations [Magel95], problems with dry bottom ash [Carrea00], tendency to corrosion, slagging, and fouling [Klasen00] etc. Some of them can be verified by special methods such as Laser Doppler Velocimetry [Most00] and measurements in scale models [Sapede01], [Sayre99]. For that purpose, various modelling technology is being used [Eaton99]. It includes computation of the flow air and coal, combustion processes and heat transfers.

Secondly, visualization can give synoptical and optimal view and presentation readable even by non-experts (see Fig. 1-2). The visualization is probably the only general means of making the dynamic processes inside the boilers understandable to a human. In an ideal case, the visualization of these processes should be physically precise, interactive and real-time.

1.4 Interactive visualization and usability in education

Education is a very important application field on which the advantages of visualization may be employed. Combustion and other dynamic processes contain parameters and features that are sometimes not very effective to be teaching using traditional approaches. One of the crucial problems is the lack of practical experience, which students should gain during their study. Practical experience can help the students deeply understand the nature of physical processes that exists in a power plant. The traditional way of education is based mostly on mastering the theory of these processes, which

creates a good base for their later deep understanding in practice. In some cases, real models of some parts of power plants are used, on which the students can perform experiments. This approach has several disadvantages - these models are rather costly and parameter settings during experiments are usually accompanied by various problems.

The visualization of power engineering processes gives the students an opportunity to understand better the nature of these processes. This approach solves one of the main problems in power engineering education: the students usually have no chance to see various types of power plants in practice. The existence of models and their corresponding visualization allows students to perform experiments with various devices and in such a way acquire much deeper knowledge about the subjects taught [Slavik99].

Thus, education on systems allowing the real-time visualization can make without doubt education process more effective and interesting. We address these and similar issues in Chapter 10. In next part of the thesis, we will focus on the current simulation and visualization in this scientific area.

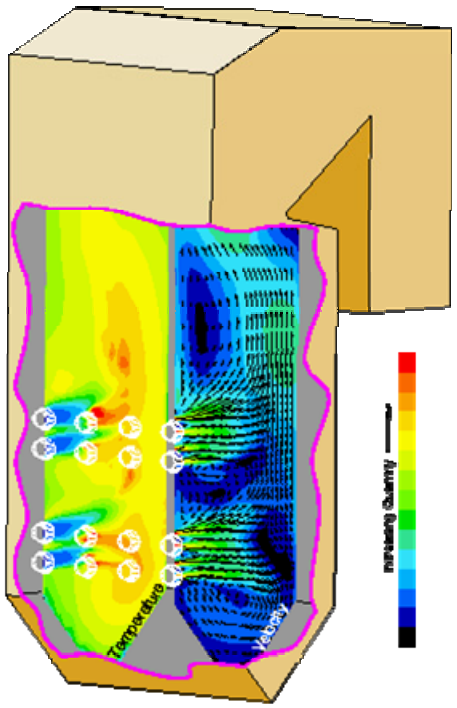


Figure 1-2: Scheme profile of the 3D boiler with the two of the most frequently monitored characteristics: the total temperature and the mass velocity

1.5 Goal of the thesis

The goal of the thesis is to design and implement techniques that together could be used to form a solution allowing interactively visualize pulverized coal combustion processes in real time. Up to now, we do not know about any projects (with exception of some early projects and attempts made by undergraduate students at CTU for their master theses – their work is briefly discussed later in this

document) that would have the same goal. These techniques should be general enough to be suitable for reusing in similar applications regarding real-time simulation and visualization of fluids. In addition, these techniques and proposals should be independent on each other. Such generality would eventually allow others to use our proposals in existing and new projects involved in simulation and modelling of fluids. If one or more of our proposals (components) would be not properly applicable - either by performance, desired precision, overall concept design or similar drawback to fulfil concrete task, it should be relatively easy replaced by a new one with more desirable results.

On the other hand, the goal of the thesis, namely due to requirement of interactive, real-time performance, is not to propose solutions, that would get closer to robustness, richness and precision offered by professional CFD packages and numerical methods. Such demanding goal would be nearly impossible to finish due to following reasons: 1) the fact, that current computers are not powerful enough to run such solution in real-time and 2) computer science, visualization and software design and engineering education specialization of the author. It would also limit the scale of the dissertation thesis (due to the effort directed to investigation of these methods), which is to offer the whole collection of techniques forming a complete solution, not only some very fine-tuned simulation method. The goal of the thesis was not to implement a perfect, production quality system. Instead of that, from the beginning, the intended purpose of the work was to create a solution, which could be used for the purposes of education at the Department of Mechanical Engineering CTU in Prague. These are not very demanding, especially considering the precision of physical model. There did not exist any team of software developers for implementation of the system, with an exception of implementation and testing of hardware, bicubic spline interpolation and IBFV visualization technique made by Petr Kadlec. No other persons, except author were involved in implementing and testing all techniques proposed in the thesis. The described goal of generality allowing to further enhance or redesign the proposed techniques, forming a complete solution, was intended to allow eventual future pioneers of interactive, real-time visualization of combustion processes, to carry on from the stage, which this thesis reached.

1.6 Our effort

Traditional methods of visualization and simulation of combustion and general fluids, which will be described in detail in Chapters 2 and 3, yield high-quality results. However, they usually have major drawback – today's common computer equipment does not offer computational power sufficient for running solutions based on these methods in real-time performance. Therefore, we have decided to propose some new solutions to attempt address this drawback.

In the field of combustion, namely combustion of pulverized coal, we are focused on creating interactive model and visualization methods. We are also focused on finding new methods and concepts, which can be easily used and implemented. Our effort has resulted in finding various

techniques, covering multiple directions leading to an interactive model and real-time of combustion processes.

Except of the implementation of high quality spline visualisation described in Chapter 9 and in paper [Kadlec04] that was done by Petr Kadlec, and development of suitable combustion equations used in combustion models suggested by Frantisek Hrdlicka, all the presented results were developed and implemented by the author of this thesis (Marek Gayer). This also applies to all the papers where Marek Gayer was a co-author, with an exception of a paper [Slavik03]. This paper consists of parts that were developed and implemented by Marek Gayer (covers approximately the second half of the paper) and parts invented by Pavel Slavik, Frantisek Hrdlicka and Ondrej Kubelka (covers approximately the first half of the paper, that deals with an issue different from the issue of the thesis submitted).

1.7 The structure of the thesis

The structure of the thesis can be summarized as follows:

1. **To find a new way for fast generation of combustion data, which can be visualized in real-time:**
   - Isotherm free stream combined with air and coal particle system (Chapter 4, published in paper [Gayer02a])
   - Fast fluid simulation combined with virtual coal particle system (Chapter 5, [Gayer02b])

2. **Special visualization data generators allowing real-time interpretation based on high capacity disk drives**
   - Fast data generator using Fluid Simulator States (Chapter 6, [Gayer03a])
   - Hierarchical data generators based on Fluid Simulator States (Chapter 7, [Gayer03b])
   - Hierarchical data generators based on unsteady data sets (Chapter 8, [Gayer04a])

3. **Interactive visualization methods and concepts**
   - Fast, high quality visualization method using programmable shaders of new graphics accelerators (Chapter 9, [Kadlec04])
   - Interactive model for education, allowing real-time visualization of characteristics and statistics (Chapter 10, [Gayer03c])

The results of our effort are more described in the next chapters of the thesis.

2 Introduction to simulation of fluids and combustion processes

2.1 The modelling of flow using CFD

Before visualization of physical processes, we must find a way for obtaining the numerical data, which represent simulated situation in combustion boiler. For that purpose, we are seeking for a mathematical model, which in some way approaches physical reality. By implementing and running such a model on computers, we simulate these processes. An important and complex task for simulation and visualization of the combustion processes is without doubt the modelling of flow (usually the flows of the air and of the coal during the combustion process). This area is subject to intensive research. The motion of fluids (gases, liquids) has been a topic of study for hundreds of years. Nowadays, Computational Fluid Dynamics (CFD) is used in almost all fields of fluid dynamics. CFD commonly represents broad family of numerical solutions and computational methods, which solve governing equations describing fluid flow. CFD is thoroughly studied since ’60s and is covered by many papers, handbooks and practical applications. For a brief introduction, overview and future of the CFD the reader could consult the paper [Gadelhak98]. For engineering practice, it is a good idea to start with books on Computational Fluid Dynamics - The basics with the applications [Anderson95] or Computational Fluid Dynamics: An Introduction for Engineers [Abbot89]. There are many others, which can be used as a replacement of these mentioned.[1].

All forms of CFD are based on the fundamental governing equations of fluid dynamics, the continuity, momentum and energy equations. These equations describe physical aspects of the problem. They mathematically express the three fundamental physical principles upon all of fluid dynamics are based:

- Conservation of mass
- Newton’s second law
- Conservation of energy

These physical principles are applied to models of flow in a 2D or 3D space that are based on:

- Fixed finite control volume

[1] The fundamentals of the CFD in the Czech language can be found in the Czech Technical University textbooks as well. One could consult [Hemzal01], [Dvorak96], [Kozel00] and [Kozel01]

- Moving finite control volume
- Fixed infinitesimally small volume
- Moving infinitesimally small volume

According to these models of flow, we apply following governing equations of fluid flow:

- Continuity equation
- Momentum equation
- Energy equation

Those equations can be summarized in various mathematical forms and together represent the Navier-Stokes equations [Anderson95]. These may be simplified into Euler equations provided we assume the inviscous flow. We define also physical boundary conditions, which represent for example walls of inlets and outlets (explicitly set pressure and velocity of air or coal). The equations are usually supplemented by other equations such as the thermal equation of perfect gas state and caloric equation state (for the heat transfer). Using numerical methods, these equations are then solved in steps of selected length resolution. For some specific cases, these equations can be further reduced to forms needed for special cases, for example incompressible or compressible flows.

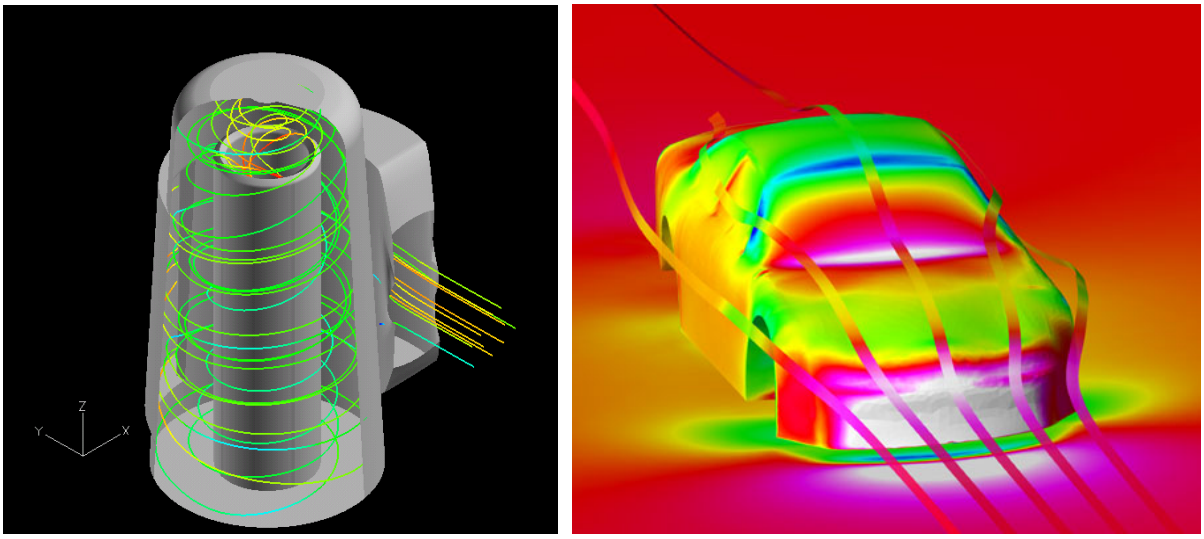


Figure 2-1: Left: CFD Finland Oy's NOVA CFD solver - Airflow inside a cyclone separator. Right: CFD data sets visualized using Amtec Plot visualization software [Amtec-WWW].

2.2 CFD software

Many academic institution and software companies worldwide offer various CFD packages, and solutions [CFD-Online-WWW]. In the world, there are nowadays spread hundreds of academic institutions and companies offering various CFD packages and solutions and various institutions

[CFD-Online-WWW].

2.2.1 Visualization in CFD software

Many CFD packages include visualization software, and the amount of options keep increasing [Waterman00]. For example, Algor [Algor-WWW] users can perform dynamic rotation, export to a VRML file, and replay analyses with different time steps. Adaptive-Researches [CFD2000-WWW] incorporate Stormview, a visualization package with tools such as isosurfaces, particle tracking, and key frame animation. FLUENT allows quantitative analysis, visualization, and flow animation any time during the analysis [FLUENTInc-WWW].

Other solvers make use of various third-party products for the presentation of their results. Amtec Engineering, maker of Tecplot data visualization software [Tecplot-WWW], has developed an add-on toolkit called CFD Analyzer [CFDAnalyzer-WWW] that helps determine the quality of the computational grids and locate problematic areas and other post processing capabilities. The software is able to visualize the pressure coefficients or vorticity magnitudes in a CFD solution, along with velocity, pressure and temperature. It performs error analysis, particularly useful for non-expert CFD users. It also features an integration feature for calculating forces, movements, flow rates, etc.

Intelligent Light's FieldView visualization software offers features such as surface contour generation, scalar and vector quantity creation, multiple streamline representations, imported particle paths, 2D plots, transient data support, and animation [iLight-WWW].

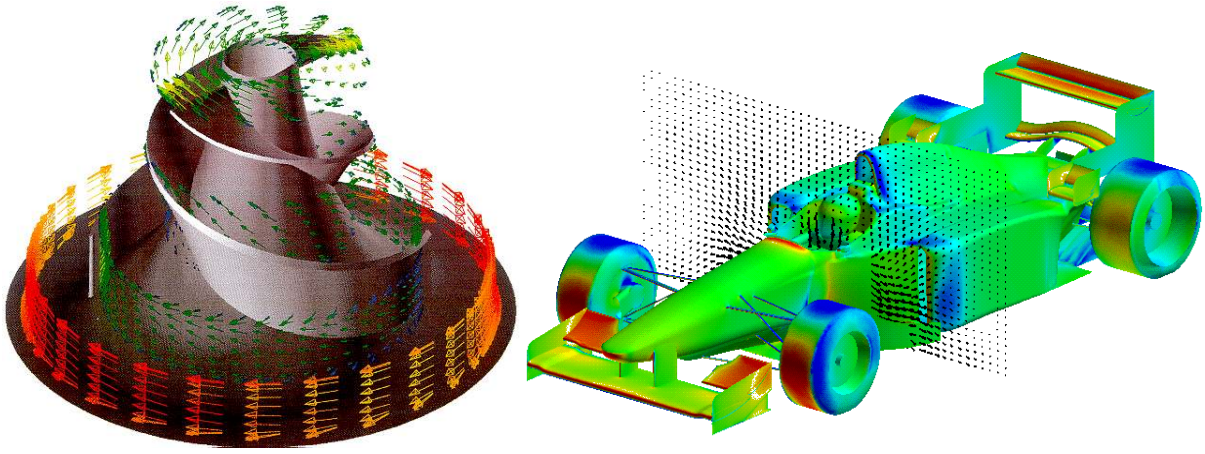


Figure 2-2: Sample of iLight graphics output [iLight-WWW]

2.2.2 FLUENT

FLUENT Inc. is the largest general-purpose CFD software code vendor. They develop and sell a wide range of CFD packages, such as FLUENT (a general-purpose finite volume package), Fidap (a FEM code developed by FDI, later acquired by FLUENT), Polyflow (polymer processing), Nekton (a

spectral element code for thin-film simulations etc.), IcePak (electronics cooling) and MixSim (mixing vessels). They also sell their own mesh generation package named Gambit. Their web site [FLUENTInc-WWW] gives further information about products, typical applications and more. FLUENT is the most used professional universal CFD application for modelling fluid flow and heat transfer in complex geometries.

Modelling of geometry in FLUENT is based on constructing a mesh, which allows proper computation of the CFD code for object (e.g. boiler that is being modelled). Supported are 2D and 3D meshes, see Fig. 2-3. After creating of the mesh, we define boundary conditions – walls, inlets, outlets, and physical properties and models of used materials and environments. The FLUENT package offers also excellent ways of visualization of computed results. Various conditions such as temperature arrays, mass tracks and heat flux may be displayed. By using a special extension called PREPDF pre-processor, the system can be applied for solving coal combustion computations.

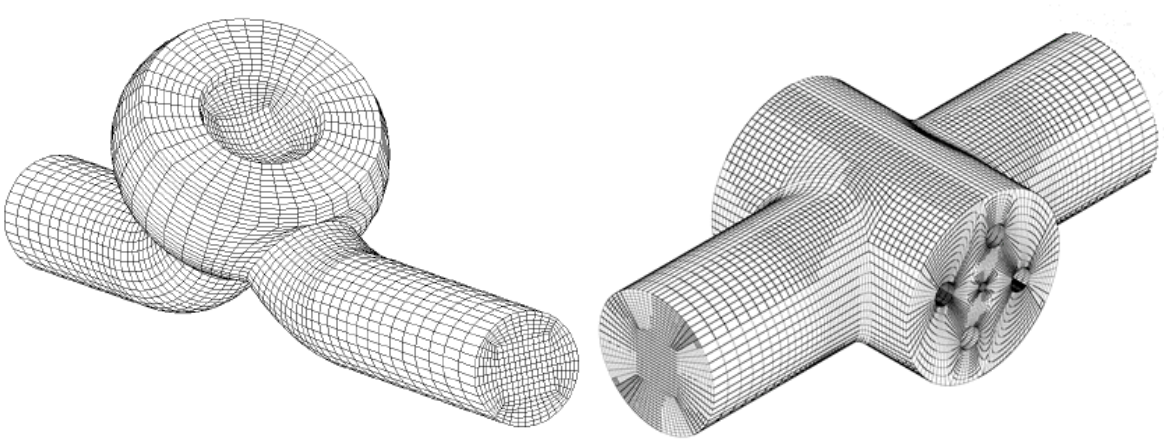


Figure 2-3: Mesh examples

In short, FLUENT is suited for incompressible and compressible fluid flow simulations in complex geometries. Additional information may be found at [FLUENTInc-WWW] and [FLUENTMan-WWW].

2.3 Fluid Simulator and Solvers

The fluid simulators and solvers based on the Navier-Stokes equations are used for the various practical computer graphics applications such as the animation of water and other liquids [Foster00], [Foster01], gases [Ihm04], [Foster97] and smoke phenomena's modelling [Rasmussen03], [Fedkiw01], [Treuille03] (including smoke flow with obstacles – [Yoshida00]). Further, aerodynamics simulation [Wejchert91], animation of the water surface [Chen97], [O'Brien95] and waves [Enright02], fluid flows on smooth surfaces [Stam03a], and ocean [Liu03]. Many others are being

investigated. Currently, utilizing results as the data for visualization, the CFD methodology by using these equations allows generation of very attractive pictures such are pictures of clouds [Stam99] - see Fig. 2-4. Other visualization results of some of the described projects are shown in Fig. 2-5.

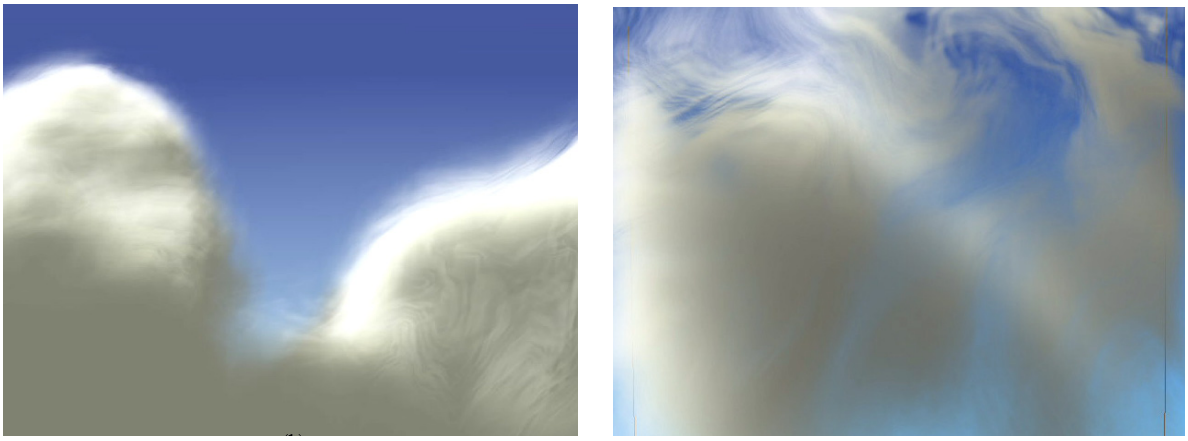


Figure 2-4: Pictures of clouds created by the interactive Fluid Simulator [Stam99].

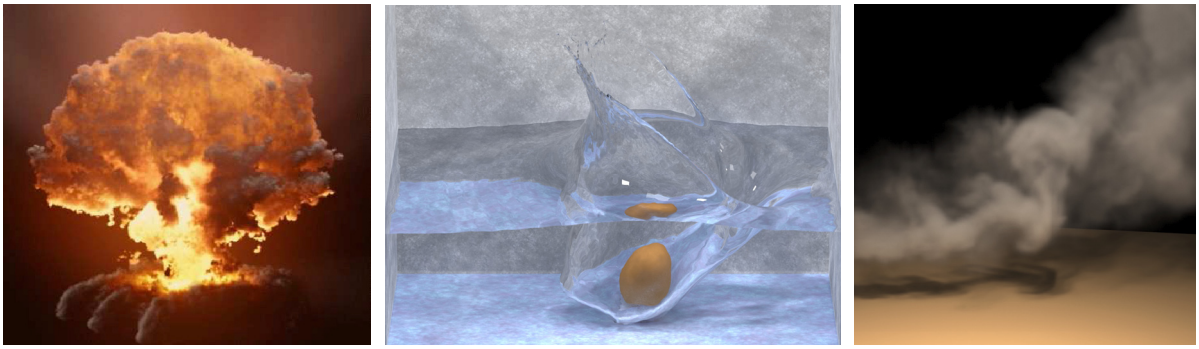


Figure 2-5: Left: Nuclear Explosion by [Rasmussen04] Middle: Rendering of water surface by [Enright02] Right: Rising smoke by [Fedkiw01].

Some of them are even used for animations and special effects where the resulting pictures play decisive role for the applications such as movies [Witting99], [Rasmussen04] and games are [Stam03b]. Some of them are used for special effects such as melting [Carlson02], [Wei03] and viscoelastic fluids [Goktekin04]. They are also used in special modelling software packages such as Maya (see Fig. 2-6).

Figure 2-6: Visualization of clouds phenomena in Maya Fluid Effects™ [Stam03b].

We can speak about two types of the Navier-Stokes based fluid simulators. Those such as [Foster96] use unstable, time step dependent solutions of the Navier-Stokes equations. Others – such as [Kass90] or [Stam99] – use stable fluid models that are able to determine the progress of flow independently on length of the time steps, but at cost of increased computational speed of single frames. In most cases, the solvers are built to meet requirements of particular applications, however usually they may be modified and reused in either general or similar applications. Fluid simulator proposed by [Stam99] can be implemented in just a few lines of code in the C language when using Fast Fourier Transformation [Stam01].

2.4 Coal combustion computation and simulation basics

Coal combustion is quite a complex process and even now not all physical aspects are well understood. In this thesis, it is not possible to describe all its theoretical and practical aspects. Thus, we are focusing on approaches and methods suitable for our work. We recommend the reader to consult books such as [Warnatz95] and [Teyssler88] for more information.

2.4.1 Coal properties

The coal particles consist of combustible part H, ash matter A and water W, see 2.1. The combustible part consists of carbon, hydrogen, oxygen, sulphur and nitrogen.

$$H + A + W = 1 \qquad 2.1$$

For basic study of coal properties and its combustion, we can use this approximation of composition together with fuel value (the amount of the heat, which is released by perfect burning of 1 kg of the combustible and its subsequent cooling to the initial temperature) and granulation of coal particles. When requesting more detailed overview, component based analysis of combustibles and other values (such as combustion heat, fuel value of the combustible part, rheological properties of ash, grinding properties, caking properties, disposition for self ignition on a dumping ground and stack, and combustible homogeneity) are being taken into consideration.

It is also important to determine the ratio of the volatile part V and involatile part I (fixed carbon) of the combustible, see Eq. 2.2. The volatile combustible is the amount of the gaseous matter, released from combustible while warming up without presence of air to the temperature of about 900^{o} C. It has major importance for the ignition process.

$$H = V + I \qquad 2.2$$

Other important characteristic is the amount of combustion air and combustion gas, which occurred during the burning of the coal, both with and without excess of the air (*imperfect and perfect*

combustion). The requested amounts of the air and combustion gas are determined from the stechiometric coefficients of the combustion equations or approximately from the empiric dependences for the known fuel value.

2.4.2 Coal combustion dynamics

Burning in the industrial combustion boilers is an organized and controlled combustion of the combustibles, which consists of many physical and chemical processes. If the total time of burning needed for passing all the chemical reaction is by orders longer than the time needed for finishing the physical processes (supplying the coal and air into the boiler, floating inside the boiler and leaving the combustion chamber through the outlet) we speak about the *kinetic combustion*. Otherwise, we speak about the *diffuse combustion*. The diffusion combustion is taking place within the most of the industrial boilers. The speed of the combustion is determined also by the physical factors. Not only the major importance have the properties of combustibles and air, but also the velocities, turbulences, concentrations of combustibles and air, the shape and dimensions of the combustion chamber, the location and direction of the fuel inlets and heat transfer of the combustion gas and boiler walls are important too.

2.4.3 The combustion speed

The combustion speed, see Eq. 2.3 (e.g. decrease of its mass for the time step) is proportional to:

- The oxygen concentration in the surrounding environment (p_{02})
- The surface, on which the combustion reaction is being passed (surface reaction between the solid and gas phase coal particles versus oxygen – a)
- The kinetic constant K_s, which represents the reaction velocity; this can be determined experimentally by measuring the decrease of the mass in relation to the particle surface. This constant is proportional to the activation energy E.

$$\frac{dm}{dt} = p_{0_2} a K_s \quad [kgs^{-1}] \tag{2.3}$$

The rate coefficient K_s can be determined by the Arrhenius equation, see Eq. 2.4, k_s is a constant, E is the activation energy, R is the universal gas constant, and T is the temperature (in Kelvins) and has kinetic component K and diffuse component $D\sigma$, see Eq. 2.5

$$K_s = k_s \exp\left(\frac{-E}{RT}\right) \quad [kgm^{-2}s^{-1}] \tag{2.4}$$

$$K_s = \frac{1}{\frac{1}{K} + \frac{r}{D\sigma}} \quad [kgm^{-2}s^{-1}] \tag{2.5}$$

The burning process proceeds in the following three steps:

- warming up the particle and releasing the volatile combustible
- burning up the volatile combustible
- burning up the involatile rest

The combustion speed is being determined for every of this phase, so that the $K_{1,2,3}$, $E_{1,2,3}$ are determined. From this basic equation can be further derived other more complicated and more accurate equations, with coefficients often depending on the experimental works for various types of combustibles.

2.4.4 The heat transfer computation

As well as for the coal combustion, there already are exact methods available that use 1D, 2D - and 3D models of the boiler that use finite elements method as the basic principle of the computation model. In each space element, the equations for balance of heat transfer flow and equation of heat transfer sharing are computed. The mathematical formulation tends to be complicated [Langtangen01], and therefore several simplifications are being used, e.g. in [Arques99].

The speed of the releasing of heat during the combustion depends above all on the temperature. The speed of removal of heat released during the combustion process depends on the difference of the combustion gas and temperatures of the wall, which surrounds the combustion chamber with the surface F. For lower temperatures, the dependence is linear and is computed using Eq. 2.6:

$$Q_o = F \cdot C_{1.2} \cdot (T_1 - T_2) \cdot dt \quad [J] \tag{2.6}$$

For the greater temperatures the heat distribution is more influenced by the radiation of the surface of boiler chamber, thus we use Eq. 2.7:

$$Q_o = F \cdot C_{1.2} \cdot \left\{ \left(\frac{T_1}{100} \right)^4 - \left(\frac{T_2}{100} \right)^4 \right\} \cdot dt \quad [J] \tag{2.7}$$

In these equations, Q_o is the heat released during time dt, $C_{1,2}$ is conductivity of surfaces of boiler chamber and T_1 is temperature of the boiler chamber cells and T_2 is temperature of boiler walls. More information about combustion and thermodynamics can be found e.g. in [Tomeczek95], [Warnatz95] and [Teyssler88].

2.5 Earlier attempts for simulation and visualization of combustion

As we mentioned in the chapter *Goal of the thesis*, there were already attempts to simulate and visualize combustion processes done at CTU. Rais [Rais97] proposed to simulate fluid layer in swirling vortex furnace completely by particle system, and created very simple model based on this approach, see Fig 2-7. Lately, Faltýn [Faltyn99] and Fortik [Fortik02] used Isotherm Free Stream (described in Chapter 4) to precalculate trajectories of air, coal and burned gas and dust particles flowing in combustion boiler, see Fig. 2-8 and Fig. 2-9. Due to static character of Isotherm Free Stream concept, these systems do not behave well with dynamic changes of configuration due to changed physical parameters during the combustion and fluid flow.

With increasing amount of air inlets on non-trivial and general cases, the modelling of the flow array is more and more hocus-pocus than any serious attempt to approach reality described by physical laws. In addition, these systems have serious problems with more complex tasks as the physical model of airflow is based on experimental approximation based on behaviour of air statically flowing from a loose inlet, as opposed to physical reality, or at least its rough approximation. Thus, the concept of static flow array is inappropriate for combustion simulation, because it does not properly react on dynamic changes of fluid flow inside boiler, due to mass and pressure changes caused by burning. Together with conclusion in Fortik's Master Thesis, [Fortik02] we discourage to build further systems based on flow array and isotherm free stream concept.

Moreover, the quality of visualization generated by these systems cannot be compared to quality of pictures generated by CFD programs, see figures below. The dynamics of the air fluid flows, that is also represented by particle system [Fortik02], [Gayer02a], implies to impossibility to use most of the current methods suitable for visualization of fluid flow by vector fields.

On the other hand, these systems were very fast on very wide range of computer system. They also demonstrated that usage of particle system for combustion and these combustion equations, simplified in certain extent, represent an interesting perspective to approach real-time simulation and visualization of combustion processes.

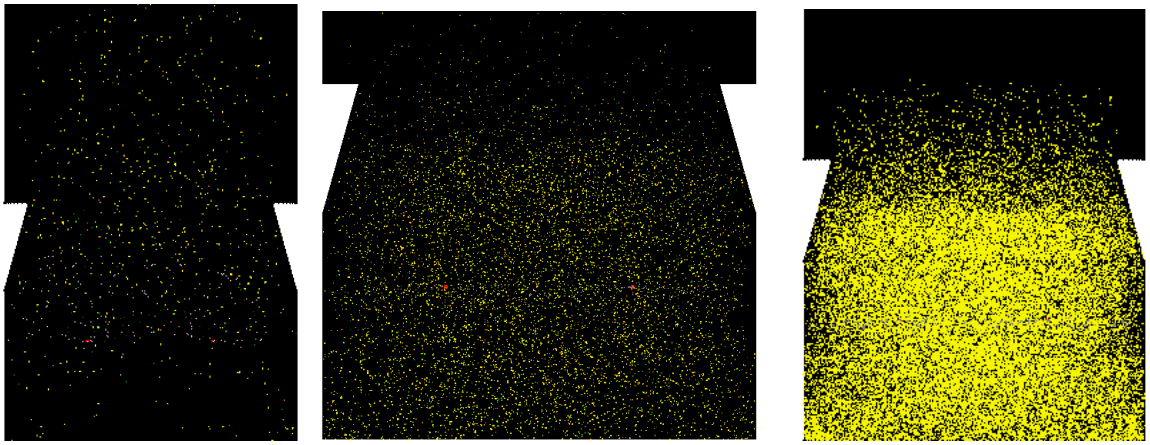


Figure 2-7: Visualization output from furnace model [Rais97]

Figure 2-8: Visualization output of combustion boiler model [Faltyn99]

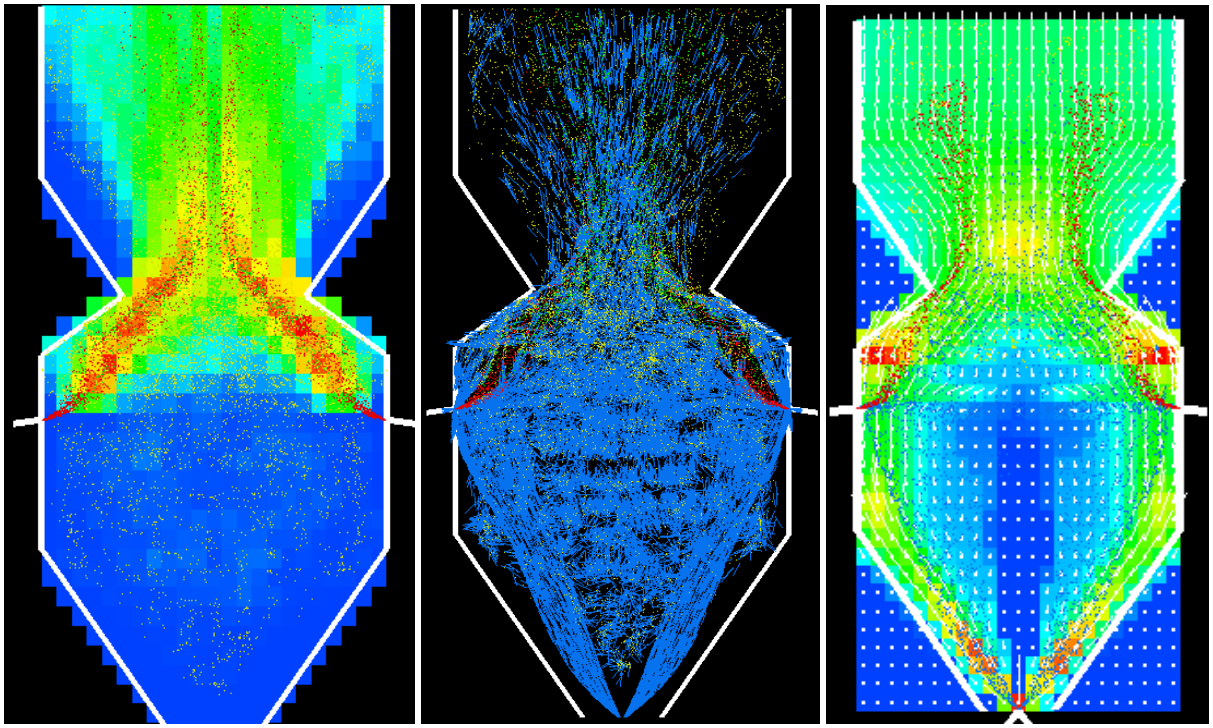


Figure 2-9: Visualization output of combustion boiler model [Fortik02].

3 Selected existing visualization methods for fluids modelling

In this chapter, we outline some of the visualization methods, which could be useful for visualization of the combustion processes. As we mentioned in *Introduction*, visualization is an important part of effort of investigation of combustion processes, by means of which we can display computed values in human readable and understandable form. It plays significant role in education also. As this thesis deals with question of creating real-time visualization of such processes, existing visualization methods are important for us and we will discuss them in this chapter.

Many scientists from different fields study and develop various methods and techniques of the fluid visualization. These techniques and methods use either immediate visualization of simulated results or data sets of pre-calculated values to visualize.

Choosing the right methods of flow visualization often depends on the compromise between the requested visualization speed and quality of the image.

3.1 Visualization of discrete values

Simulation model, which serves as a data generator for data visualization is discrete if it provides values only in selected points within the simulated volume. Some models allow to compute values in any place of the simulated volume, but even in such case, the number of points that may be computed is limited by the CPU speed and other factors. Thus, the visualization model deals with set of points and corresponding values. The typical representation is a structured grid. When visualizing such discrete data sets, we should address following tasks: interpolation and converting numeric values to visual representation. Interpolation allows us to find a value that would be corresponding in a point located between discrete points. Several various interpolation methods for that purpose already exist: linear, polynomial interpolation (such as cubical), spline interpolation, and others. Those methods vary in speed and quality. We refer reader to [Lichtenbelt98] for further details, where these methods are well described.

3.2 Visualization of scalars and contours visualization

While visualizing vector fields is particularly important in many applications, it is also important to quantify and visually characterize scalar features of the flow field such as temperature and magnitudes of vector quantities. In 2D we can use simple quads or contours, see Fig. 3-1. In 3D, we can use volume-rendering techniques to visualize scalar features in the flow field.

Contours allow visualization of areas, which have similar value, with colour maps with same

colour tone, which are easy to read and understand. Contour plots are very frequently used in presenting CFD results. They are lines connecting points of constant properties, drawn in 2D or 3D space. Regions in which contour lines are grouped together are regions of high gradients in the flow. In colour contour plots, the contour lines are replaced with a continuous changing in colour shades so that the entire flow-field picture becomes a continuous painting. Adding contours increases the amount of information that the picture may communicate. A possible way is to enhance comprehensibility of the picture by adding isolines, i.e. curves, which in the picture links places with the same value (see Fig. 3-1 right).

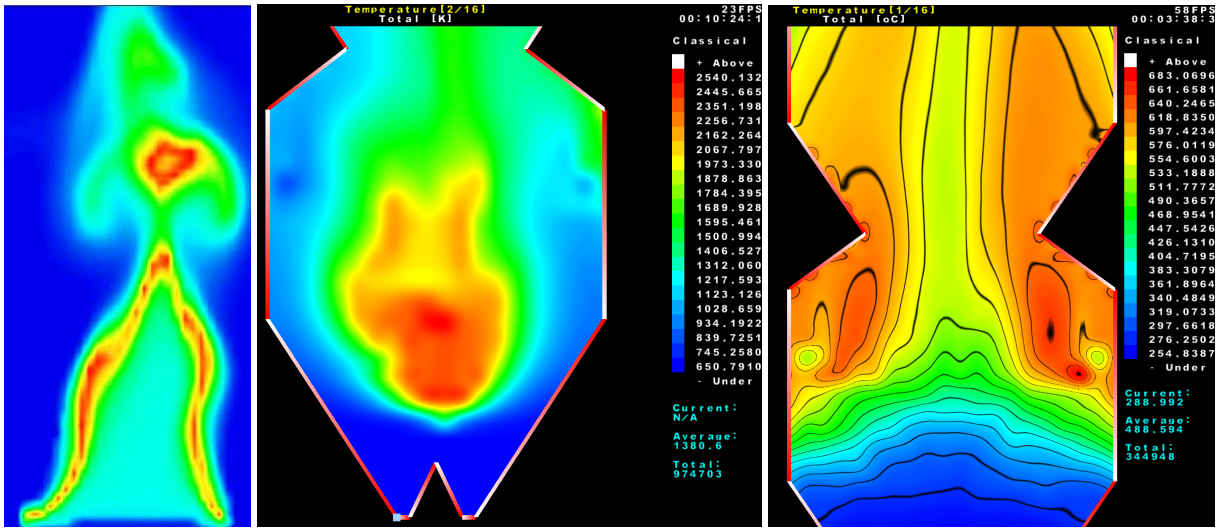


Figure 3-1: Left: Visualization using contours of temperature in a slice, one meter heptane pool fire. Combustion Group of Utah [Smith-WWW]; Middle: Visualization of temperatures inside combustion boiler using contours (generated by our system), Right: Visualization of temperatures with isolines (generated by our system)

3.3 Visualization of vector fields

The vector fields can be easily visualized in real-time [Gelder92]. The flows can be visualized by arrow plots (sometimes called *hedgehogs*) or by simple lines (with various ending) that represent the vectors. The arrows could be easily rendered using simple lines (for example by means of OpenGL). However, when visualizing many values organized in grids, arrow plots representing all the discrete values may become small, allowing only a limited amount of values to be displayed.

Unlike surface rendering methods, direct volume-rendering methods can be used to visualize 3D scalar data without intermediate geometric primitives [Swann91], [Crawfis92], [Crawfis93]. Sample output using this technique is shown on Fig. 3-2.

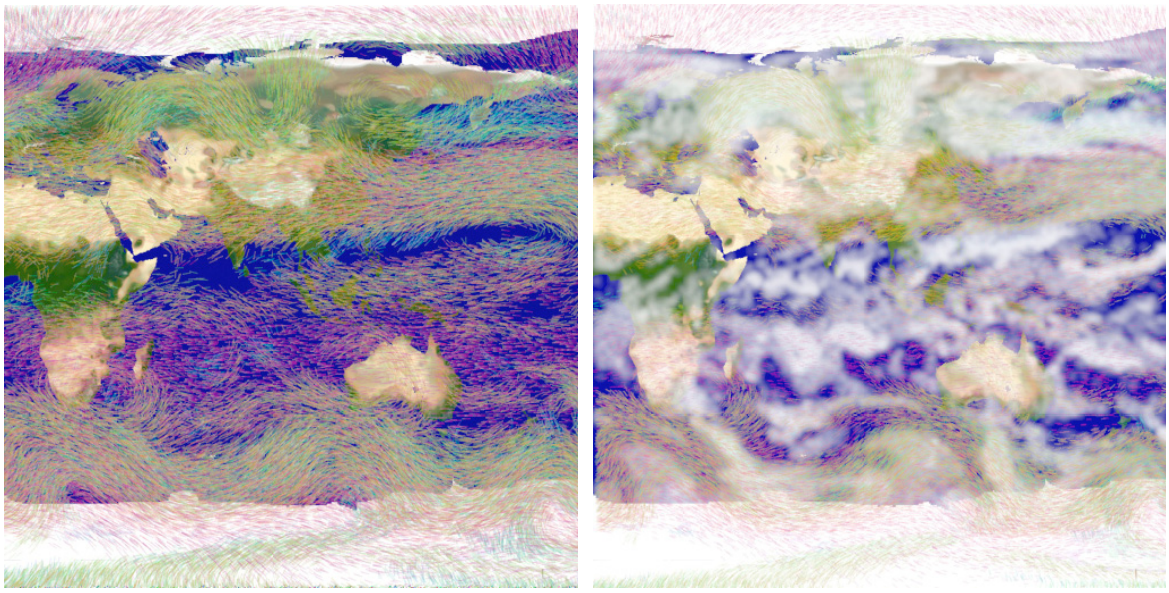


Figure 3-2: Global Climate Model winds colour coded by altitude (left) together with percent cloudiness (right) [Crawfis92].

We can use some enhancement and additions to these methods to visualize multiple data values. For example, we can use a combination of discrete and continuous visual elements arranged in multiple layers to represent visually the data. For example, [Kirby99] uses concepts of painting inspired by the brush strokes the artists apply in layers to create an oil painting. With these techniques, we are able to visualize several levels of information. For example, Fig. 3-4 (left) shows visualization of characteristics of velocity, vorticity, rate of strain, turbulent charge and turbulent current.

3.3.1 Highlighting places of interest and feature extraction techniques

In some cases, the common flow visualization methods could yield images which would be either unclear, or would not give good overview of the visualized data, or in which it would be hard to recognize a selected behaviour of the flow. We need to find ways to highlight important places in flow fields. For that purpose, we use various methods. For example, a method for producing compact vector field visualizations, as described in [Telea94] (see Fig. 3-3), uses a hierarchical simplification of the vector data. The simplification is based on the definition of the vector similarity function for visualization of different aspects of the vector field. The method can visualize vector data at different levels of detail by interactively choosing the simplification level.

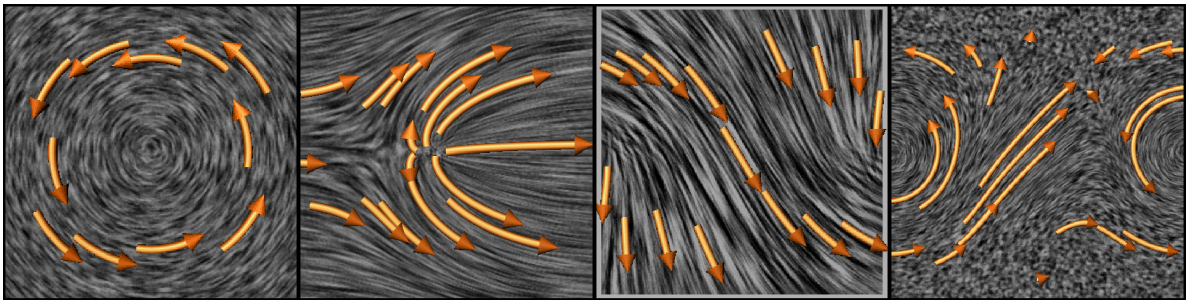


Figure 3-3: Flow visualizations using curved arrows on spot noise textured backgrounds [Telea94].

We can use various feature extraction concepts to focus visualization on interactively selected parts or simplify complex and/or dense flow fields, such as clustering methods for vector fields [Garcke00]. Others can be found in numerous papers, e.g. [Post02], [Westermann00], [Theisel03], [Polthier00], [Sohn02], [Sadarjoen99].

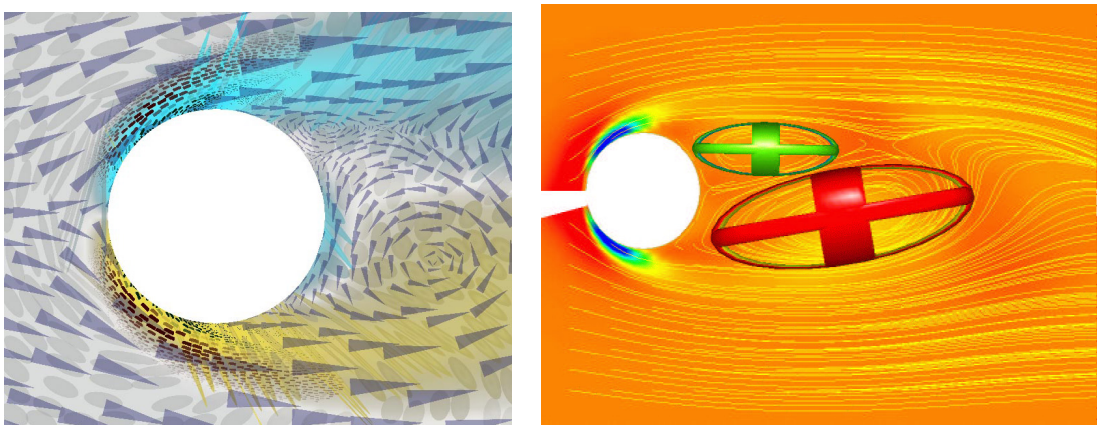


Figure 3-4: Left: Combination of velocity, vorticity, rate of strain, turbulent charge and turbulent current [Kirby99]. Right: Flow past a tapered cylinder with vortices approximated by ellipses, and streamlines released in a slice [Sadarjoen99]

3.3.2 Spot noise

Another method of vector field visualization is based on random selection of data points. Only data in this subset are visualized. By combining all of these separate visualization methods, we can obtain an image, which well describes structure of the vector field. We can then use this image as a texture to visualize the flow field directly. These texture-based techniques differ in a way they visualize the value in discrete points.

Spot noise techniques [Wijk91], [deLeeuw95] draw spots of random intensity on random places. Every spot is altered regarding the direction of value of vector field in the corresponding point. On the picture, which is synthesized by drawing large amount of such spots, the global structure of field and feature details can be found. This technique is fully suitable for visualizing flows (see Fig. 3-5).

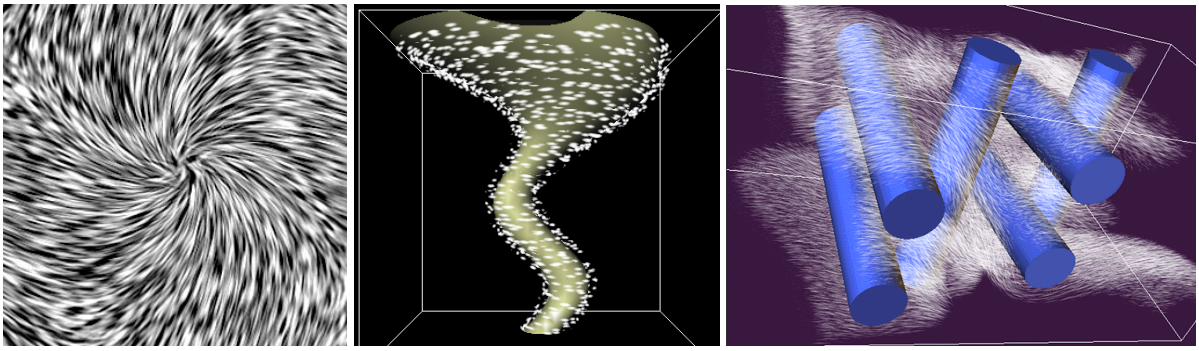


Figure 3-5: Left: spot noise used to visualize a vector field [deLeeuw95], Middle: Spots noise with contour surface [Max94], Right: Spot noise rendering of HEPA filter [Max94].

3.3.3 Line Integral Convolution (LIC)

The Line Integral Convolution technique (LIC, [Cabral93], [Forsell95]) is based on filtering the white noise texture (usually with same resolution as vector field) in corresponding places of streamlines found in visualized vector field. Sample results of this method are in Fig. 3-7. This technique is similar to spot noise [deLeeuw98].

Later, the LIC and spot noise methods have been improved and extended. OLIC [Wegenkittl97a] uses less dense texture then LIC and asymmetric filtering, which gives effect of colour drops inserted into the vector field. FROLIC does not use filtration of texture at all, which allow increase visualization speed [Wegenkittl97b]. UFLIC [Shen97], [Shen98] and AUFLIC [Liu02] (see Fig. 3-6) optimizes LIC for unsteady flows animations. Other enhancements and extensions to this method can be also found for example in [Forsell95], [Stalling95], [Shen96], [Bordoloi02a], and [Interrante97] (see Fig. 3-8).

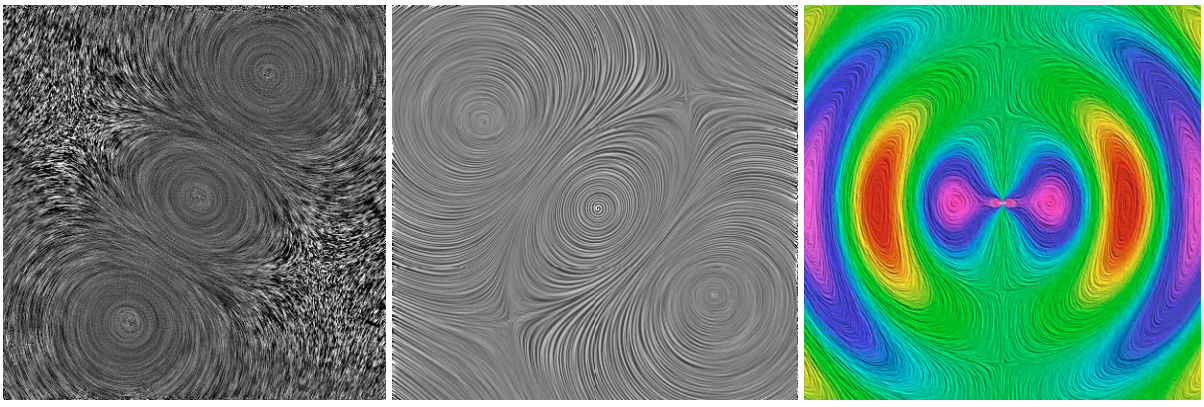


Figure 3-6: Left: LIC image (hierarchical LIC algorithm) [Bordoloi02a], Middle and right: AUFLIC image [Liu02].

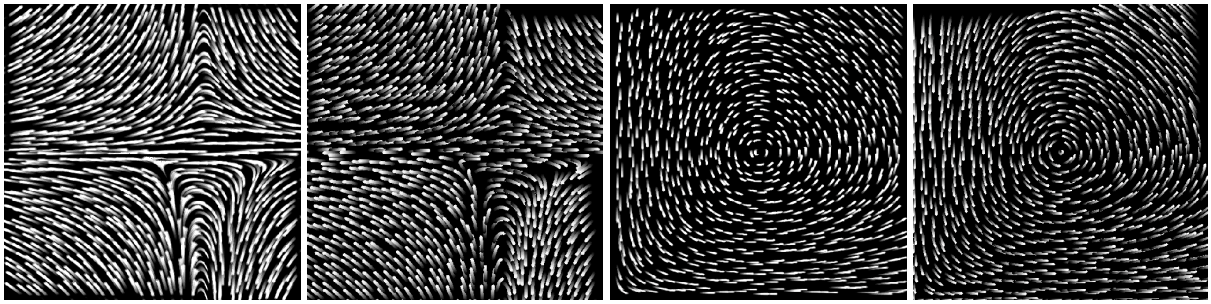


Figure 3-7: Left: Econometric data with OLIC and FROLIC, Right: Circular flow with OLIC and FROLIC [Wegenkittl97b]

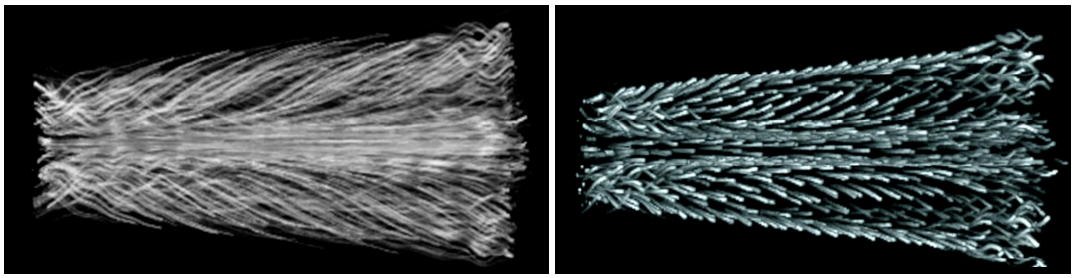


Figure 3-8: 3D Flow dataset visualized by Volume LIC [Interrante97].

Other methods of flow visualization include visualization of unsteady vector fields using Lagrangian-Eulerian Advection (LEA) scheme [Jobard02], and Advected Radial Basis Functions [Pighin04]. Further, Unsteady Flow Advection–Convolution (UFAC) [Weiskopf03], visualization of flows using streamlines [Zockler96], [Wischgoll02], [Mattausch03] and streaklines [Sanna00], Markov Random Field Texture Synthesis [Taponecco03], anisotropic nonlinear diffusion [Preusser99]. An interesting approach for rendering animated streamlines using textures is proposed in [Fuhrmann98]. An attempt to visualize pulsative flow using special particle pathlines called Particle Flurries has been described in [Sobel04]. By combining flow visualization in 3D with light models, we can obtain even better results [Stock02]. A possibility of utilizing streamarrows for streamsurfaces visualization is discussed in [Loeffelmann97]. We recommend [Crawfis00], [Post02], [Post03], [Laramee04a] and [Laramee04b] for comprehensive overview of Flow Visualization method.

3.3.4 Image Based Flow Visualization

IBFV (*Image Based Flow Visualization*) is a technique suitable for 2D flow visualization [Wijk02] that gives visual results similar to previous described methods, even that the used algorithm considerably differs. Visualization works directly with resulting bitmap, which is transformed in each step according to values of vector field and mixed with filtered noise texture. The technique can be easily implemented on graphical accelerators, because it uses their features of texture processing and alpha blending. The visualization runs in real-time and has two steps. In the first, the image generated by previous visualization step is divided by grid. Each cell of the grid is transformed and deformed by

values in its corners and modified cell is drawn over the original image.

In the second step, the image is blended with one of the prepared image of noise textures, created during initialization. The resulted image is then drawn and saved for next animation step.

In process of visualization, we can modify many parameters, which determine the visualization result, such as coefficient of transparency of noise texture, size and scale of noise texture etc.. By altering these parameters, we can get results similar to spot noise, OLIC etc. The method offers great performance on current graphical hardware. It has been also extended to 3D [Telea03], (see Fig. 3-10). Together with [Lefer04], this technique can be used for animation of steady vector fields.

We can easily modify and extend IBFV by drawing of various objects inserted into the flow field, such as traces of colours or grids. Because IBFV in each step transforms a bitmap, we can draw to this bitmap selected object and continue with animation. The object is then modified and transformed together with bitmap in fluid flow, without need of extending the algorithm in any way and without additional performance requirements.

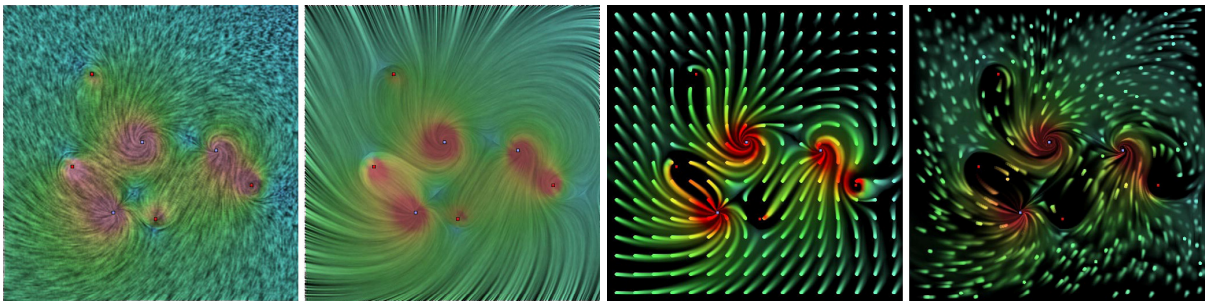


Figure 3-9: Left: IBFV images visually similar to Spot-Noise and LIC techniques, Right: IBFV images similar to OLIC and FROLIC [Wijk02].

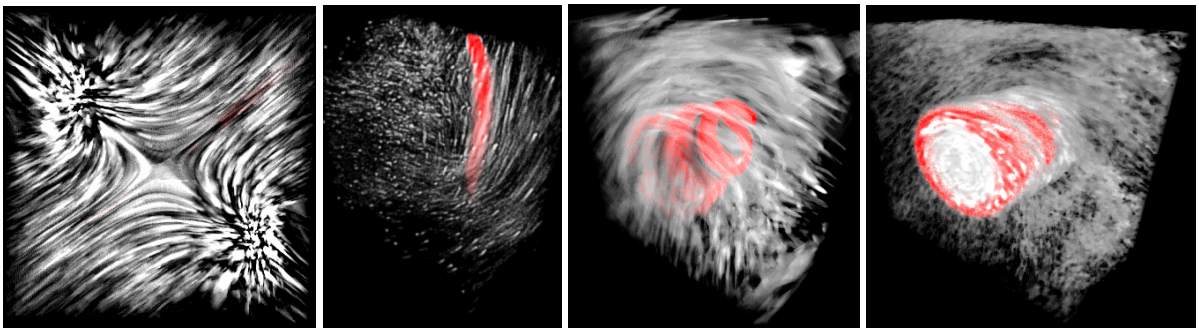


Figure 3-10: Flow Images generated using 3D IBFV [Telea03].

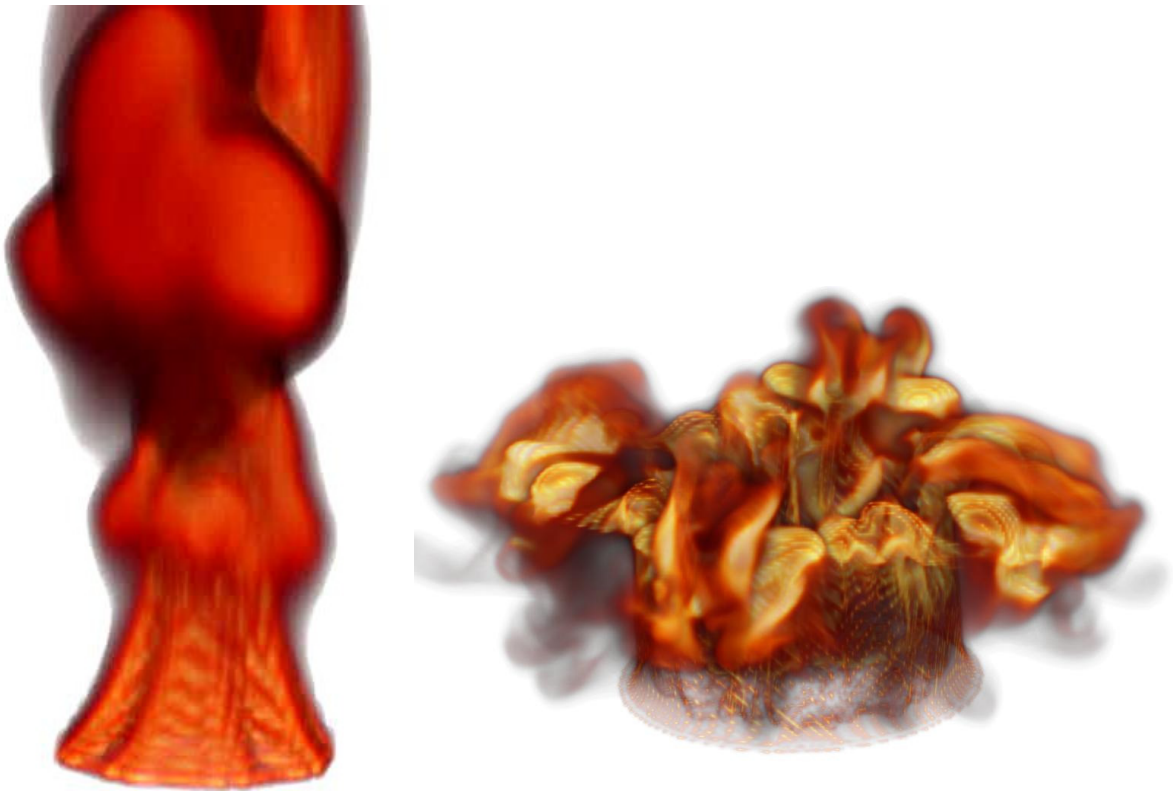
3.4 Realistic visualization of fire and flames

Current models, which deal with simulation and visualization of behaviour of flames, are more concentrated on the visualization part of the combustion process. They use several other approaches such as cellular automata [Takai95] (which is also used for other physical simulation purposes – e.g.

excellent clouds modelling in [Dobashi99], [Dobashi00]), diffusion processes [Stam95], Lattice-Boltzman model [Zhao03], [Wei02] and tomographic method for reconstructing a volumetric model from multiple images fire [Ihrke04]. [Nguyen02] presented a physically based method for modelling and animating of burning of either solid or gas fuels. This method uses the incompressible Navier-Stokes equations to model independently both vaporized fuel and hot gaseous products.

Models of this type are more concentrated on the visualization part of the combustion process (see Fig. 3-12). The resulting pictures can be used in applications where the quality of visual effect plays decisive role (e.g. movies, computer games etc.).

Remarkable research is being done by the Combustion Group of Utah [Smith-WWW]. Using special software CSAFE [C-Safe-WWW] on supercomputers equipped by hundreds of processors, they are able to study various types of fuels and bring the detailed and precise visualization and statistics of various fuels (especially jet-fuel fires). Of course, these and similar applications are very computationally expensive even on the today's specialized hardware (e.g. 30 seconds per animation frame in [Nguyen02]).



:

Figure 3-11: Left: One time step of a one meter, heptane pool fire as it puffs
Right: The temperature within a ten meter, heptane pool fire

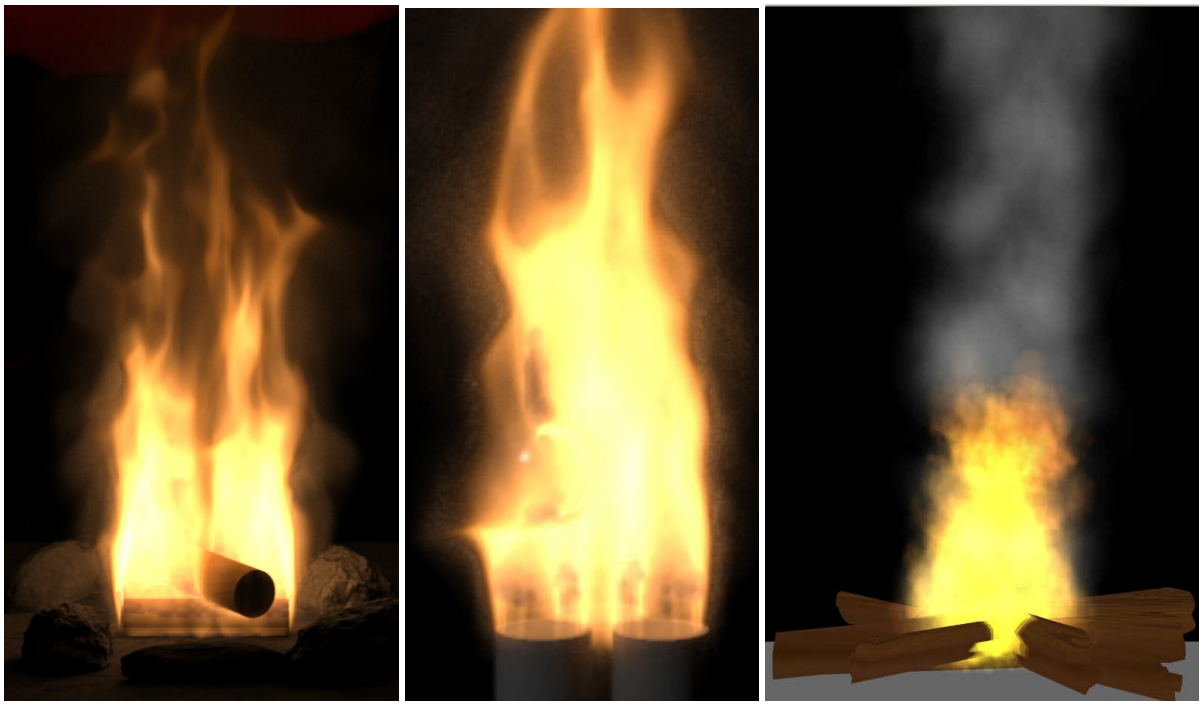


Figure 3-12: Left: Solid fuel burning [Nguyen02], Middle: Gas fuel burning [Nguyen02], Right: A camp fire with smoke [Wei02].

3.5 Particle systems

The particle systems have been introduced by Reeves [Reeves83]. He described a particle system as a collection of many small particles that together represent the visualized object(s). Over a period of time particles are inserted into a system, they move and change within the system, and are eventually removed from the system, depending on specified conditions. Each particle has assigned its individual attributes.

Figure 3-13: Visualization using particle systems [McAllister00]

Visualization of particles provides intuitive and efficient means for the exploration and analysis of complex flow fields. With many particles simulated and visualized, it is possible to gain well-arranged overview of flow.

By using large number of particles with suitable distribution we can use them for visualization of flow volumes [Guthe02], see Fig. 3-14. Particle systems can be used for modelling of turbulent flows – e.g. [Gamito95].

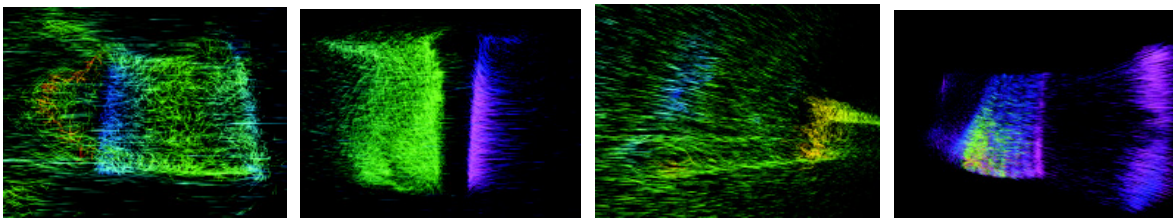


Figure 3-14: Visualization of datasets volumetric vector field datasets by texture based particles [Guthe02]

Bruckshen at al. [Bruckschen01], [Kuester01] developed a method enabling to visualize up to 60000 particles into a flow field while maintaining an interactive frame rate. Their method was based on the pre-calculated particle trajectories stored in a scheme optimized for selection of sub-grids. They used high-capacity disk system (Redundant Array of Inexpensive Disk) for this application. They used data sets were for a classical problem: simulating the flow of a fluid around a spherical object, see Fig. 3-15.

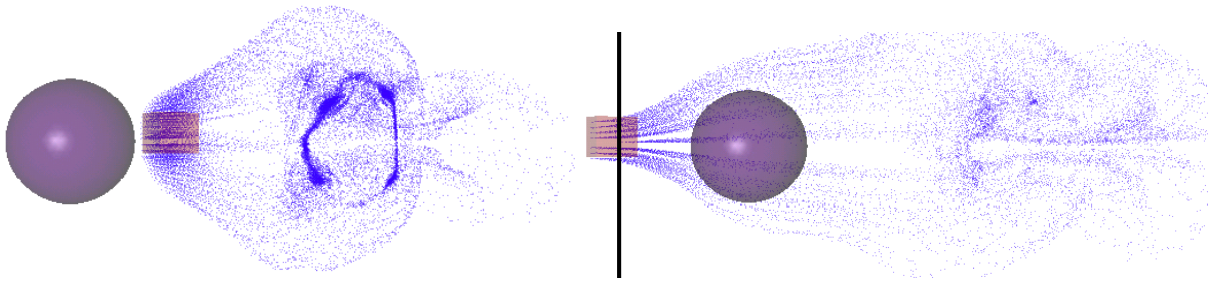


Figure 3-15: The interactive particle system based by the pre-calculated particle trajectories [Bruckschen01]

An example of successful usage of particle system can be found in [Hrdlicka96], **[**Hrdlicka04**]**. They used a particle system as a solution for modelling the situation in gas cleanup filters. High precision simulation had been gained as well as the possibility to study in details the behaviour of these filters, and the possibility to perform large number of experiments in short time, thus proposing optimal cleanup filter design.

Particle systems by their nature offer an easy way for display particle tracks and easy construct streamlines, see left and middle of Fig. 3-16. In CFD, particle tracing allows the visualization of vector fields resulting from simulation [Lane97], [Sadarjoen98]. Sample of particle tracing taking into account collisions with the car body geometry using visualization with streaklines, ribbons and glyphs [Schulz99] can be found on Fig. 3-16.

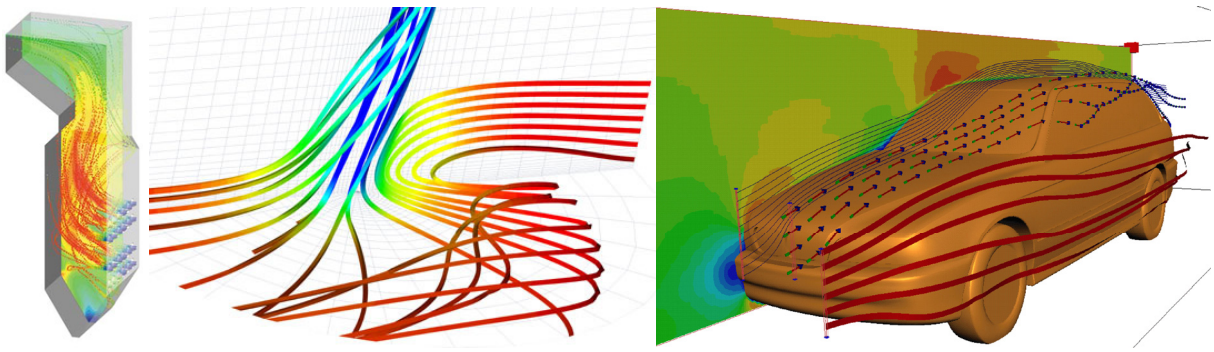


Figure 3-16: Left: Streamlines in the combustion chamber, Middle: Streamlines detail, Right: particle tracing taking into account collisions with the car body geometry using visualization with streaklines, ribbons and glyphs [Schulz99].

3.6 Hardware acceleration for scientific visualization

Nowadays, real-time visualization of various unsteady data is more often maintained by using techniques and approaches regarding to new programmable abilities of current graphical accelerators, such as described in [Heidrich99], [Weiskopf01] (see Fig. 3-18), [Telea03] and [Weiskopf04a]. Bordoloi and Shen introduced hardware accelerated visualization of dense 2D vector fields with flexible level of detail control [Bordoloi02b] (see Fig. 3-17), [Shen04]. Hjertager [Hjertager00] introduced new approach for accurate shading model of lightning models with multitexturing for volume rendering. Other projects aim toward acceleration of various phases of volume rendering process [Roettger03]. Abilities of current graphics hardware may be utilized even for raytracing [Weiskopf04b] simulation purposes, performed directly on the graphics hardware as described in [Wu04], [Harris02], including simulation of fluids using Lattice Boltzman method [Wei04], [Li03], Navier-Stokes [Liu04] and Euler equations [Harris03].

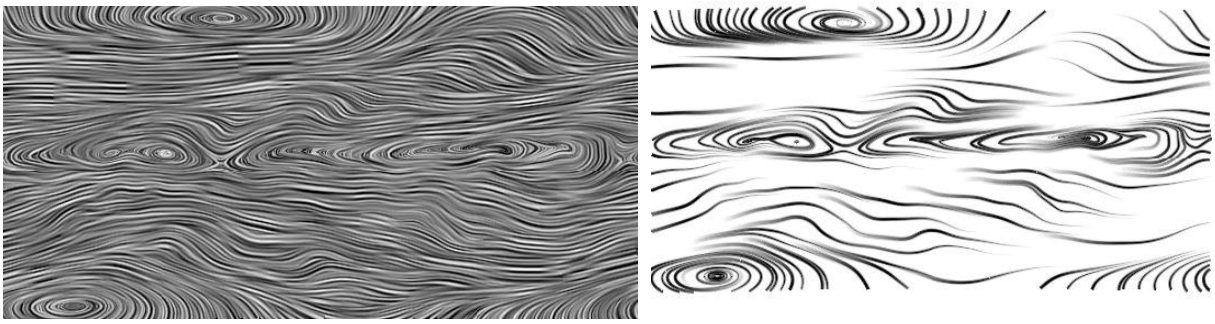


Figure 3-17: Output of level of detail visualization of ocean dataset based on hardware texture mapping [Bordoloi02b]

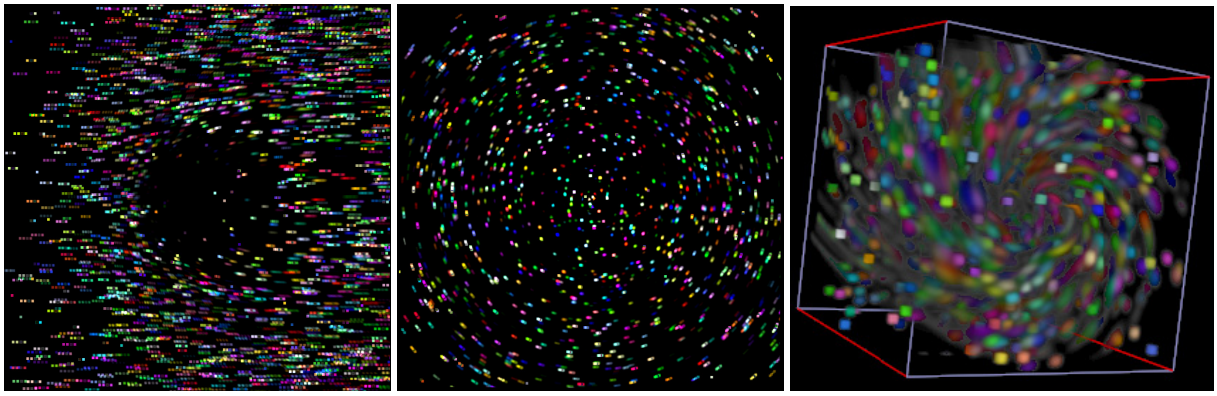
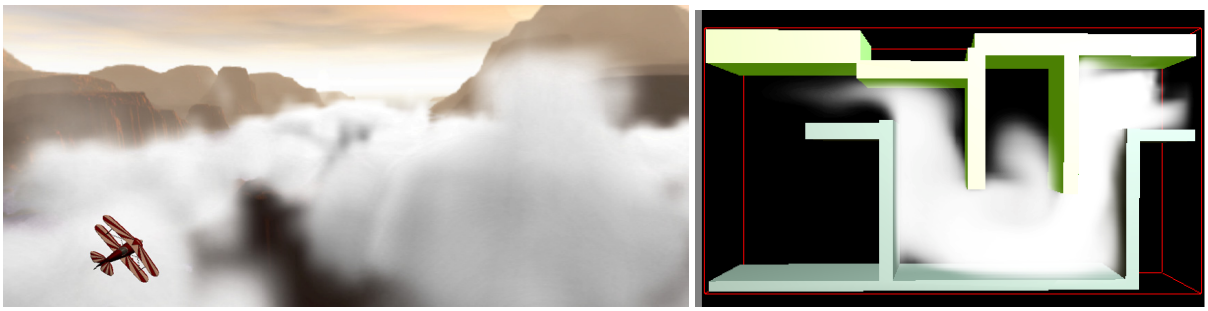


Figure 3-18: Visualization of a 2D circular flow using hardware-accelerated texture advection technique, based on randomly injected particles [Weiskopf01].



Left: Clouds simulated at GPU in interactive flight application, SkyWorks. Right: Visualization results of Fluid Simulation with complex boundary conditions on GPU [Liu04].

4 Isotherm-free stream

4.1 Fast computation of air movement

One of the methods of increasing visualization speed towards the real-time frame rates is to speed up the computation of the computation of air movement inside the boiler. For that purpose, we have investigated the pre-calculated flow arrays. Our first system that results from our investigation has been based on this technique. It uses static, non-time varying grid of pre-calculated air velocities inside the boiler.

There are various possible ways to obtain such flow grid. The classical way would be based on using classical CFD differential equations – by continuously solving these equations, until stable state is reached. However, we did not use this approach due to issues of complexity, performance and due to desire to investigate new solution, not previously used.

That was the reason, why we have selected a technique based on the idea of the isotherm, a loose flow that runs from a circle inlet. We have adopted technology based on the half empirical and analytical solutions of the air stream behaviour. The air stream flows through inlets to the boiler. The solution of the streams in a limited area in the boiler is quite complex, especially for the non-isothermal flow. That is why we calculate it as an isothermal free stream, which flows from the circle inlet. Our solution is based on the idea of G. N. Abramovič that can be found in the [Cihelka69].

The air stream forces to move surrounding air due to the influence of the turbulence. This approach allows us to speed up considerably the calculation of flow array. The stream can be considered as a cone. The top angle 2α depends on the level of the turbulence of the stream in the inlet (see Fig. 4-1).

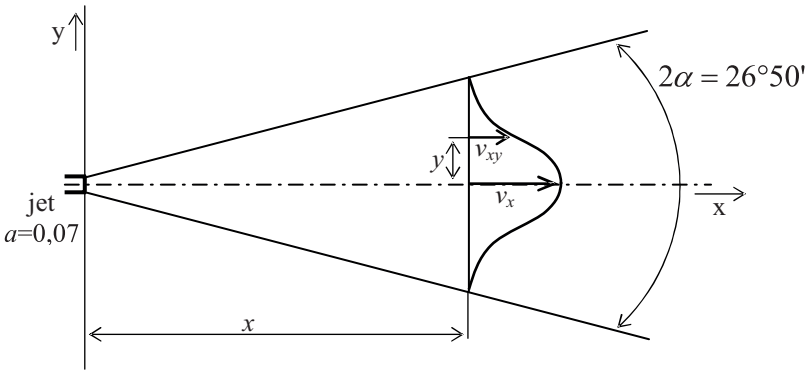


Figure 4-1: Isotherm free stream flowing from the inlet

For any distance x from the input of the stream, the maximal speed in the x-axis and y-axis is decreasing to zero with the Gauss distribution (see Fig. 4-2) described as Eq. 4.1 (where sigma corresponds to variability and mi is mean value):

$$f(y) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{\frac{-(y-\mu)^2}{2\sigma^2}} \qquad 4.1$$

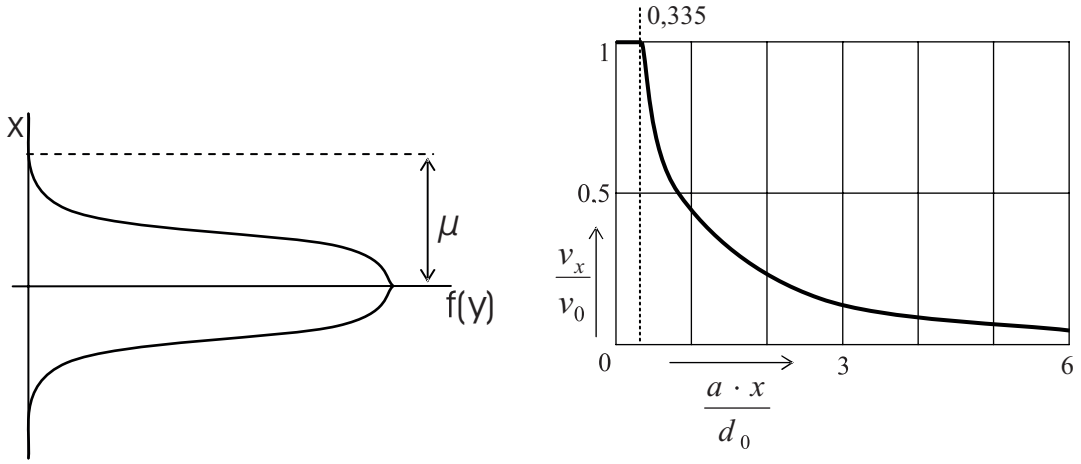


Figure 4-2: Left: Graph of the Gauss distribution, Right: Curve of the axial speed v_x

The x-axis speed v_x gradually decreases and is determinate by the Eq. 4.2, where d_0 is diameter of the inlet, a is coefficient of vorticity and v_0 is initial velocity in the inlet. See Fig. 4-2 for an illustration of all the above given approach.

$$v_x = v_0 \frac{0{,}48}{\frac{a \cdot x}{d_0} + 0{,}145} \qquad 4.2$$

However, a situation may arise that a stream collides with the wall and/or there is a collision with another air stream. In such a case, other virtual inlets are added to handle this situation and to approach the real situation. On the other hand, in complex situations this methodology is quite complex and hard to implement. That is why we later were to develop principles of direct simulation of the physical process inside the boiler, which we described in Chapter 5).

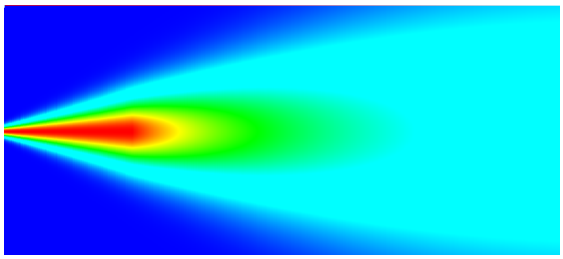


Figure 4-3: Distribution of the velocities of the air stream in the space

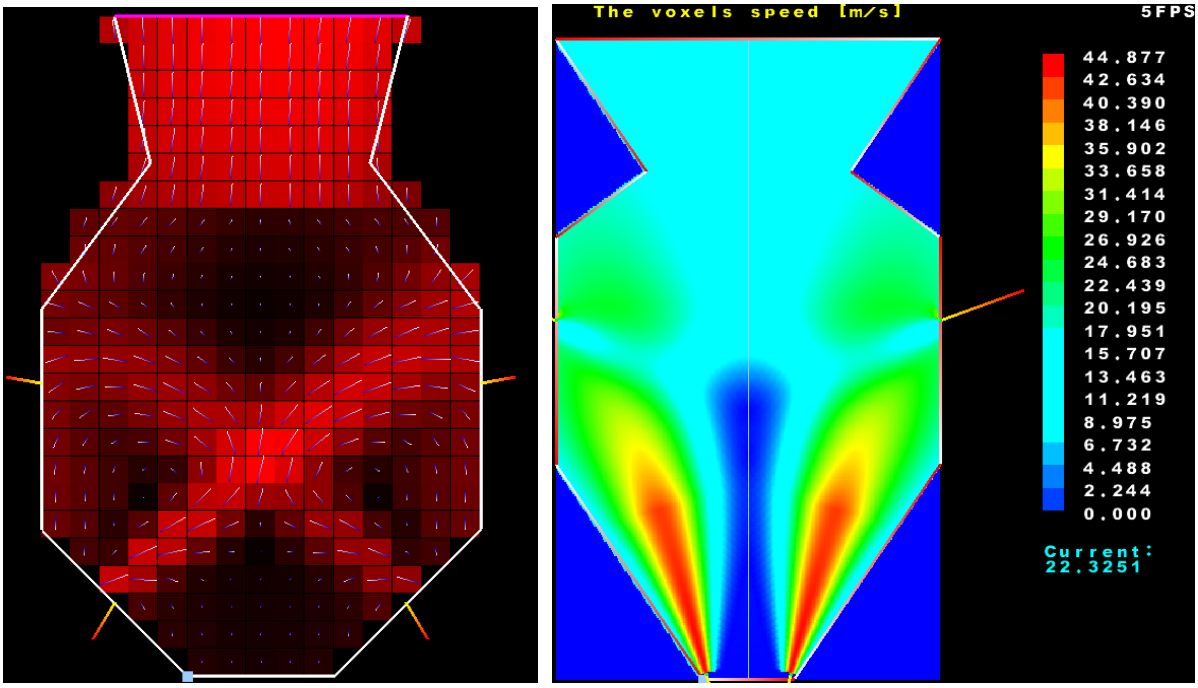


Figure 4-4: Our not very successful attempt to model combustion processes using isotherm-free stream

4.2 Air and coal particle system

We have used the combination of the computed flow and a particle system. The model makes use of both the air and coal particles to determine important characteristics such as speed of combustion and heat fluxes. They have been also directly used in visualization. We used simplified physical and thermodynamics equations for that purpose [Gayer02a].

4.3 The method applications, advantages and drawbacks

We have been using this method in older versions of our system to perform a fast determination of flow array. However, we have found this method only suitable for only very simple boiler configurations. We have faced serious problems when using complex cases including collisions of several inlets. The method does not allow changing the boiler configuration on the fly - e.g. changing properties of the air inlets. Therefore, we have further investigated fluid simulator techniques based on simplified Navier-Stokes equations instead of this simple technique. However, the above described method could be possible useful when modelling various air streams, including combustion of loose, fuel stream flowing from rockets in rendering and general computer graphics, more than our not very successful attempt to use this technique to approach physical reality, and that is the reason we decided shortly described it here also. The method is also described in our paper [Gayer02a].

5 Modelling of the physical behaviour in the boiler

5.1 Introduction and overview

Our goal is to implement interactive visualization of the behaviour inside the boiler. To fulfil this goal, we have to face a serious problem. We needed a method that would work as a data-provider and generator for further real-time visualization. The important requirement for such generator would be real-time performance and optimization towards requirements the visualization process. For such purpose, we decided to design a solution based on simplified CFD. Our effort resulted in an original methodology based on fast fluid simulation of behaviour inside the boiler combined with concept of virtual coal particle systems. These particles can be used directly and efficiently in the visualization process.

Our system is a application which makes use of grid cell division. The heart of this application consists of the fluid simulator. It is based on a simple approach, which, on the top level, consists only of application of the two laws of physics – Euler Equation and The Continuity Equation. This approach could be applied to the general computer graphics visualization and animations tasks involving the move of the air mass. The heat and temperature changes generated during the combustion processes are computed using a simplified coal pulverized combustion engine. Our fluid simulator also allows distribution of the heat by moving mass. We use a virtual coal particle system, which is used for both the simulation of interaction (and combustion) with the hot air volumes, and the visualization of the coal flow.

Our simulator, as well as others such as [Foster96], is unstable. The simulated area is divided to 2D structured grid cells. In each step, we calculate the new characteristics (e.g. velocities and masses of air) for all grid cells. All calculations are reduced on nearest neighbours of the calculated cells (see Fig. 5-1). We periodically repeat these computations in each time step of the simulation. The stability of unstable solvers depends on proper selection of dimensions of selected grid cell, and time step in regards of speed of value changes (such are velocities and mass) in cells during simulation. For details, see e.g. [Foster96].

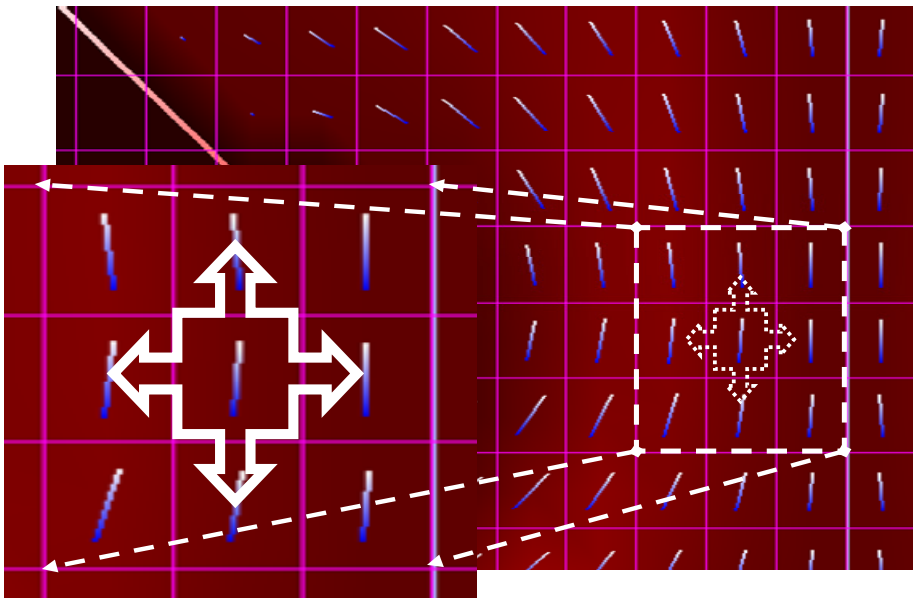


Figure 5-1: Division of the boiler area to 2D grid cells. The cell values in the next time step are computed from nearest neighbours only.

5.2 The fluid simulator physical background

Rather than using complex and stable (but computationally more expensive methods, because they must solve systems of differential equations in each time step) methods such is in [Stam99], [Stam00] or finite differences based methods such as in [Foster96], we do not perform any global computation over all cells in the boiler. Instead of that, we determine the situation only in nearest neighbouring cells. In our approach, the air in the boiler could be considered as compressible, viscous and turbulent flow. We solve fluid behaviour arising from Euler equation and The Continuity Equation (mass should be conserved). See Eq. 5.1 and 5.2.

$$\frac{\partial v}{\partial t} + (v \cdot \nabla)v = F - \frac{1}{\rho}\nabla p \qquad 5.1$$

$$\frac{\partial m}{\partial t} = \oiint v \cdot \rho \cdot dS \quad [kgs^{-1}] \qquad 5.2$$

Where F is outer interacting force (e.g. gravitation force), v is vector of velocity, m is mass, p is press and ρ is density. They form nonlinear system of partial differential equations. When solving these equations, we use Newton's Second Law (see Eq. 5.3). This law says that change of the velocity per specified time dt of an object is dependent on the forces acting on it. In our case, the force can be expressed as the differences of pressures (dp) between the neighbouring cells (Eq. 5.4).

$$\frac{dv}{dt} = \frac{F}{m} \quad [m^2 s^{-1}] \qquad 5.3$$

$$F = dp \cdot S \quad [N] \tag{5.4}$$

The S is the face surface between the neighbouring cells. This is needed for the equations of the face surface. Thus for the face surface between the neighbouring cells in the x and y direction in 2D space we get (Eq. 5.5 and 5.6):

$$S_x = h \cdot d \quad [m^2] \tag{5.5}$$

$$S_y = w \cdot d \quad [m^2] \tag{5.6}$$

The *h*, *w* and *d* are the height, width, and depth of the cell. The difference of the pressure *dp* between the two cells can be calculated from the total air mass m in the cell using the state equation of the perfect gas (Eq. 5.7 and 5.8).

$$p = k \cdot m \quad [Pa] \tag{5.7}$$

$$dp = k \cdot (m_2 - m_1) \quad [Pa] \tag{5.8}$$

The constant is *k*, which can be derived from the perfect gas equation (see Eq.).

$$k = \frac{R \cdot T}{V \cdot M} \tag{5.9}$$

Using Newton's Second Law and Eq. 5.4 for a selected cell we get the increase of the selected cell velocity *dv* per time step *dt* as Eq. 5.10:

$$dv = \frac{dp \cdot S \cdot dt}{m} \quad [m^2] \tag{5.10}$$

We calculate this equation for all the components of the current cell velocity (*dvx*, *dvy* for 2D space and *dvx*, *dvy* and *dvz* for 3D space).

Next, for the second fluid simulator step, we must calculate the change of the mass in the current cell during the time step *dt*. By using the continuity equation, we get (Eq. 5.11):

$$dm = dv \cdot \rho \cdot dS \cdot dt \quad [kg] \tag{5.11}$$

The ρ is the density of the cell and could be easily calculated from the mass in the cell and the volume of the cells. The *dm* is the mass change in the selected direction. We calculate this equation for all the directions (X,Y for the 2D case, X, Y, Z for the 3D case). We also update the temperature

inside the cell by the weighted average of the temperatures of the mass, which is present in the cell and the mass flowing from neighbour cell toward the current cell.

The viscosity is added to the solver using a simple equation. The friction tension between the next cells is determined by:

$$\tau = \mu \cdot \frac{dv}{dS} \quad [Pas] \tag{5.12}$$

The is τ friction tension (pressure), μ is dynamic viscosity, *dv* is difference of the velocities of the neighbouring cells and *dS* is the touching space face.

We set the proper boundary conditions corresponding to the surrounding environment. For cells, which neighbour with the boiler walls, we assume that no air can pass through/from it and thus the normal component of the air velocity is zero. Behind the outlet cells of the boiler we assume atmospheric pressure $p=10^5$ Pa regardless situation in the boiler.

In each step, we calculate the new values of the velocity and mass for all the cells in the boiler. We periodically repeat the computation until the required time step of the boiler simulation is reached. We have implemented this methodology into working code which simulates the movement of the air inside the streams in real-time.

The main advantage of this method is very fast convergence of the results. No calculations are performed over the global cell array. The results of this method for the specified time could be immediately used at the runtime. Our method, as well as other numeric methods, is principally independent of the boundary conditions, which can moreover be changed anytime during the simulation.

On the other hand, this method is conditionally stable, like [Foster96]. The stability depends on the size of the cell, time step and maximum velocity and mass values in the cells during solution. This is caused by only reducing all the calculations on the nearest neighbours of the calculated cells. Thus, if the velocity in the cell exceeds the maximum speed, it allows the longer distance than the length of the cell to pass as mass for the time *dt*. Other instabilities can occur in situations, when for certain cells the result of calculations is, that more air mass should flow out than the total mass in the cell. Thus, we must carefully set or compute the *dt* parameter regarding the cell size to avoid such cases.

5.3 Implementation of the fluid simulator

Our fluid simulator algorithm has been implemented in 2D cell space, with variable depth of the boiler (z-axis). However, it should be relatively easily extended to 3D space (by means of extending the proposed computational model, not by implementation demands and performance issues, which we estimate demanding).

The fluid simulator consists of the following arrays:

1. The velocity array [m/s]
2. The air mass flow array which can be easily recomputed to the array of the air flow pressures [kg]
3. The temperature array [K]
4. The O2 concentration array [values from 0.0 to 1.0]

The chamber of the boiler is divided to the cell area of the dimensions of the XRES × YRES arrays. We keep two instances in the memory of each of the arrays. These are accessed by the two pointers that define which of this array is the source (holds the state at the given time t) and destination (holds the state at the time $t + dt$). At the end of the simulation step, we swap the source and destination pointers. Regarding the equations above, we choose the following representations of the arrays:

- **The mass flow (M0, M1), temperature (T0, T1) and concentration of O_2 (C0, C1) arrays:**
  M0,T0,C0[XRES][YRES], M1,T1,C1[XRES][YRES].
  The values in these arrays represent overall characteristics of the cell and are related to the middle of the cell.
- **The velocity arrays:**
  VX0[XRES+1][YRES], VX1[XRES+1][YRES]
  VY0[XRES][YRES+1], VY1[XRES][YRES+1]. The values of the velocity arrays are related to the centre of the walls of the margins of each cell. For example, to the cell with the mass `M[0][0]` correspond the velocities `VX[0][0]`, `VX[1][0]`, `VY[0][0]` and `VY[0][1]`. This representation allows fast computation of the mass, which flows between the neighbouring cells. It is also used for example in [Foster96]. The situation is shown on the Fig. 5-4.

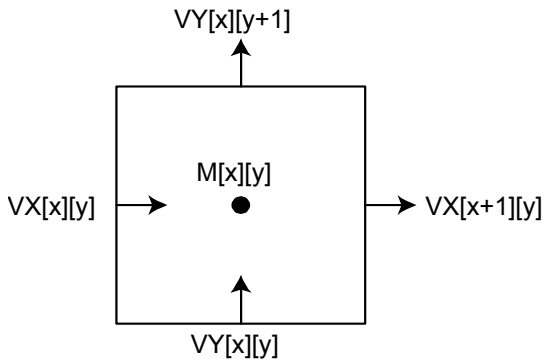


Figure 5-2: Representations of velocity and mass arrays

The air flow simulation is divided to the following steps (see Fig. 5-3):

```
Initialize;
while( simulating ) {
     VelocityStep(VX0, VY0, M0, VX1, VY1, dt);
     MassAndTemperatureStep(VX0, VY0, M0, M1, T0, T1, C0, C1, dt);
     SwapAllThePointers;
     CombustionSystemSimulatorCode(dt);
}
```

Figure 5-3: The airflow simulation steps

The `Initialize` routine sets all the initial conditions, such as the pressure, velocities and obstacles, at places where the boiler walls are detected. The `VelocityStep` routine counts new velocities for all the individual cells. Depending on these velocities, the `MassAndTemperatureStep` routine calculates all the mass changes from the nearest neighbour cells for the corresponding cells. After that, the `SwapAllPointers` routine swaps the pointers of the source and destination arrays. The program control is then passed through the `CombustionSystemSimulatorCode` routine to the system of simulator coal combustion. This part is independent of the fluid simulator code. It handles various actions, such as interacting and moving of the coal particles with the air in the cell, changing the boundary conditions accordingly to the user interaction, visualization of the results, etc. These parts are briefly described in the following text. After the code of the described single step of the simulator is executed, the fluid simulator code continues running in next steps.

5.4 Implementation notes

During implementation and testing, we discovered sensibility of our system on appropriate setting of time step and size of the computational grid. These parameters depend on maximum pressure and velocities inside the boiler. We use the following additional modifications for stability of computation.

1) We limit maximum velocity of flow field. Because we use only nearest neighbourhood in fluid our simulator, in one time step we must not allow a flow of the mass from one cell farther then to one of neighbourhood cells. To omit such behaviour, we limit maximum velocity. Maximum velocity limit is *mxvel=max(D[x], D[y])/dt*. We use the following equation, which limits low velocities only very little, while high can only asymptotically approach to maximum velocity limit.

$$v = v + dv * \left(1 - \left(\frac{v + dv}{mxvel}\right)^2\right) \qquad [Pas] \tag{5.13}$$

2) In each time step, it is necessary to propagate small part of mass to neighbourhood cells to prevent generation of abnormal waves causing unstability of solution that would be caused after several frames of simulation. This behaviour is caused by discretization and can be easily avoided by filtering computed values according to the neighbouring cells. Without the filtering, the entire mass, due to high difference of pressure between computed cells, could flow into a neighbouring cell. That way, we would receive press difference in opposite direction and in next step, we would be in the same situation as we started. Thus, we compound the resulting mass in cell from majority of its value (less than 97%) and rest from its neighbourhoods. Moreover, this behaviour does not influence precision of computation at all (even if we compound all values from neighbouring cells and none from computed cell). Propagation of values to surrounding cells express the physical behaviour of diffusion, which exists in fluids (as in our case) and balance differences of pressure.

5.5 The concept of the virtual coal particles

Particle systems in computer graphics can simulate and visualize various natural phenomena such as dust, rain, fog and other phenomena, which could involve particle object primitives. Particle object abstraction is also used for industrial technology [Rhodes98]. We have found that particle systems also could represent a suitable way for modelling the pulverized coal combustion process and we have investigated their use for this purpose.

In our system, the virtual coal particle system allows both computation and visualization of coal mass elements in the boiler. The particles displayed and calculated do not correspond to the real coal particles in the boiler. Instead, they represent the corresponding mass of coal in the cell under investigation. Therefore, we call them virtual coal particles. Thus, one virtual coal particle carries many (e.g. millions) of real coal particles. The quality and speed of both simulation and visualization could be altered by increasing or decreasing the amount of these virtual coal described particles. Currently, the amount of particles being used for simulation varies between thousands and tens of thousands.

Our solution of choosing particle system as a building block for interactive visualization and simulation of combustion processes is unique. We do not know about any other solution, based on a similar idea. Using of particle system adapted to combustion process allow great overview of dynamic of the process.

The combustion and heat transfers and fluxes are being computed separately for single grid cells and corresponding particles inside them.

The movement of the virtual coal particles through the cells is determined by the gravity force Fg and aerodynamic resistance F0. These forces are computed from the following Eq. 5.14 and 5.15.

$$\vec{F}_g = m \cdot g \qquad [N] \tag{5.14}$$

$$\vec{F}_0 = \frac{1}{2} c\, \rho\, S\, dv^2 \qquad [N] \tag{5.15}$$

The m is the mass of the real coal particle, c is the coefficient of the air resistance, ρ is the density of the air in the present cell, S is the surface of the particle cross section. The dv is the difference of the velocities of the particle and air velocity in the cell, where the particle is located (the air flow around the particle). From 5.14 and 5.15 we get for the particle acceleration (Eq. 5.16)

$$\vec{a} = \frac{c\, \rho\, S\, dv^2}{2\, m} - \vec{g} \qquad [m^2 s^{-1}] \tag{5.16}$$

The velocity of the virtual coal particle is then modified using Eq. 5.17:

$$\vec{v}_p = \vec{v}_p + \vec{a} \cdot dt \qquad [ms^{-1}] \tag{5.17}$$

Before moving a particle to the predicted destination, we must check for possible collisions with the wall. If this is the case, the particle track is mirrored and bounced off the wall. For each cell, we have list of walls the cell interferes with. We first determine, in which cell the particle is located, and according to that, we check for possible collisions. If this is the case, the particle track is mirrored and bounced from the wall (see Fig. 5-4). Particles are generated from the inlets (usually located in the walls of the boiler).

Currently, we are using a simplified model of particle system, because we ignore collisions between individual particles/ Sample visualization of our particle system can be found in Fig. 3-13.

Some factors could change their motion. For coal particles, we cannot omit the force of gravity. The coal particles are attracted to the bottom of the boiler. The acceleration is determined by the weight of the particle and the surrounding environment.

5.6 Combustion and heat transfer

The combustion process of the pulverized coal and resulting heat convection is a quite complex problem [Warnatz95], [Arques99]. Again, we use some simplifications due to the need for fast computation. First, we are using a simplified combustion process. Instead of simulation of these processes using complex differential equations, we use a statistical view of the combustion process. We consider the fuel value of the coal and molar capacities of the combustibles. The combustion is

being computed separately for each single cell.

To start combusting coal, two conditions must be satisfied: in the computed cell, there must be at least some minimal combustion temperature (e.g. 573 K), and a proper mass of coal and air that is to be burned.

Depending on the current temperature, weight and proportion of the coal, the virtual coal particles are being burned. For air mass, we just decrease the corresponding O_2 concentration. For coal particles, we decrease the amount of the combustible part of the particle.. If the mass of the combustible part reaches some minimal value, we assume that the coal particle is burnt out and we change it to the burnt gas particle. The situation is illustrated on the Fig. 5-5.

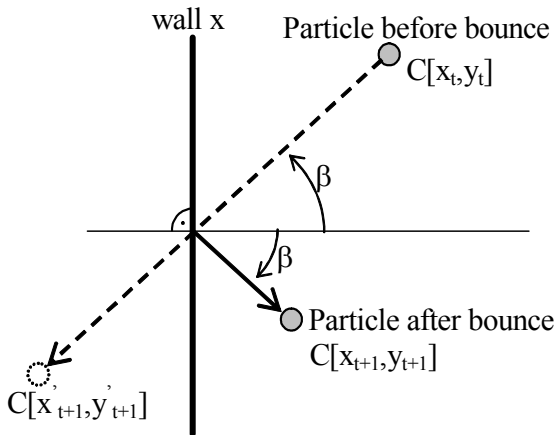


Figure 5-4: A particle bouncing off the wall

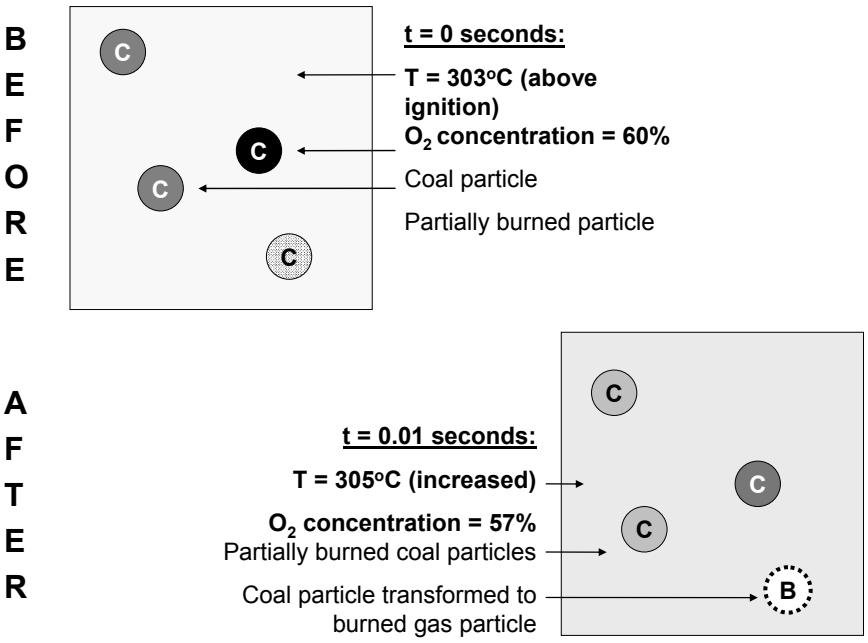


Figure 5-5: Example interaction of coal particles during the combustion process for the time dt in a selected cell

Between these processes, depending on the reaction heat transfer, the released heat is transferred to the air mass, which is present in the current cell. Therefore, the overall temperature of the cell increases.

Because of the dynamic processes in the boiler, the heat is distributed by the air mass to the other cells, thus increasing the temperature and making possible to start other combustion reactions. We also count the heat radiation between the walls and the mass in the cells. The heat transferred from the given surface of the cell F during the time *dt* is comparable to differences of the temperatures of the wall and the cell (power of four) [Warnatz95]. We also need to determine the coefficient of the radiation $C_{1,2}$ regarding the present mass in the current cell. These ideas are summarized in Eq. 5.18.

$$Q = F \cdot C_{1.2} \cdot \left\{ \left(\frac{T_1}{100} \right)^4 - \left(\frac{T_2}{100} \right)^4 \right\} \cdot dt \qquad [J] \tag{5.18}$$

We assume, for the sake of simplicity, that the temperature of the walls is constant (typically below the minimal combustion temperature).

5.7 Visualization

To maintain a reliable and fast visualization of the generated data, we use industry standard OpenGL platform. We chose OpenGL over the possibility of competitive DirectX, because it allows portability to non-windows platforms, is easier to develop in, and we have already created supplement libraries for it. In addition, students at the CTU in Prague usually have much better knowledge of this API, because courses are available on Department of Computer Science, on which OpenGL is being taught.

Thanks to this, nowadays, our system could be used on a standard, even a very cheap graphics accelerator. Furthermore, our system uses MGL graphics library on the backend to maintain easier visualizations of common OpenGL primitives [Gayer00]. This is an OpenGL based library optimized for visualization of common 2D graphics objects and graphical user interface (supports images and fonts directly). We use it to implement an easy user interface, in the common 2D coordinates. Therefore, such components are much easier to implement than in a native OpenGL.

Windowing interface is maintained by the GLUT library [Mason98]. Therefore, our system is easily and fully portable to other systems.

The boiler walls and outlets are approximated by straight lines. The particles are displayed using standard OpenGL pixels. The selected local characteristics in the cell, such as the total temperature, mass storage, the wattage, and heat flux state and/or changes can be visualized in real-time. The particle tracks can be easily determined by the fast particle system animation. Currently, the characteristics in a cell are simply visualized by the quads. Although the quality of the visualization could be improved by choosing smaller cell sizes, we implemented contours to bring smoother

graphics output. The more detailed overview of the proposed visualization methods are discussed in later chapters. For sample visualization outputs, see Fig. 5-6 and Fig. 5-7.

We described only basic visualization features of the system to illustrate and clarify the architecture of real-time visualization system based on proposed data generation. Further possibilities of visualization will be discussed later in the thesis, namely in Chapters 9 and 10.

5.8 Results

The created application is primarily intended for the education, resulting in preference of computation speed over the precision. Although our requirements for precision are not demanding due to this fact, we still wanted to know, if our computation can approach the precision of production quality of professional CFD software. On a real test boiler (dimensions 6.4 m x 13.7 m), we have simulated and visualized combustion processes, see Fig. 5-8.

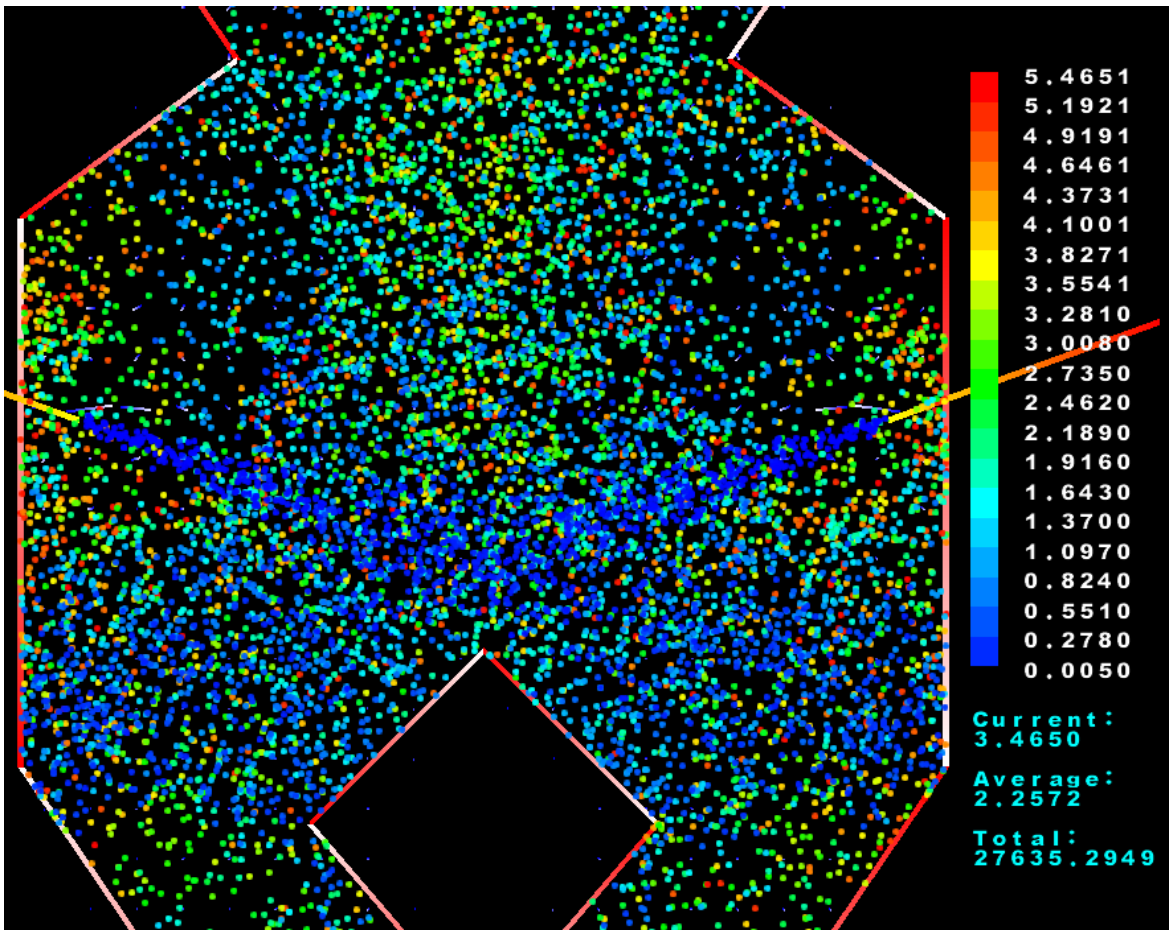


Figure 5-6: Particles flowing from the inlets

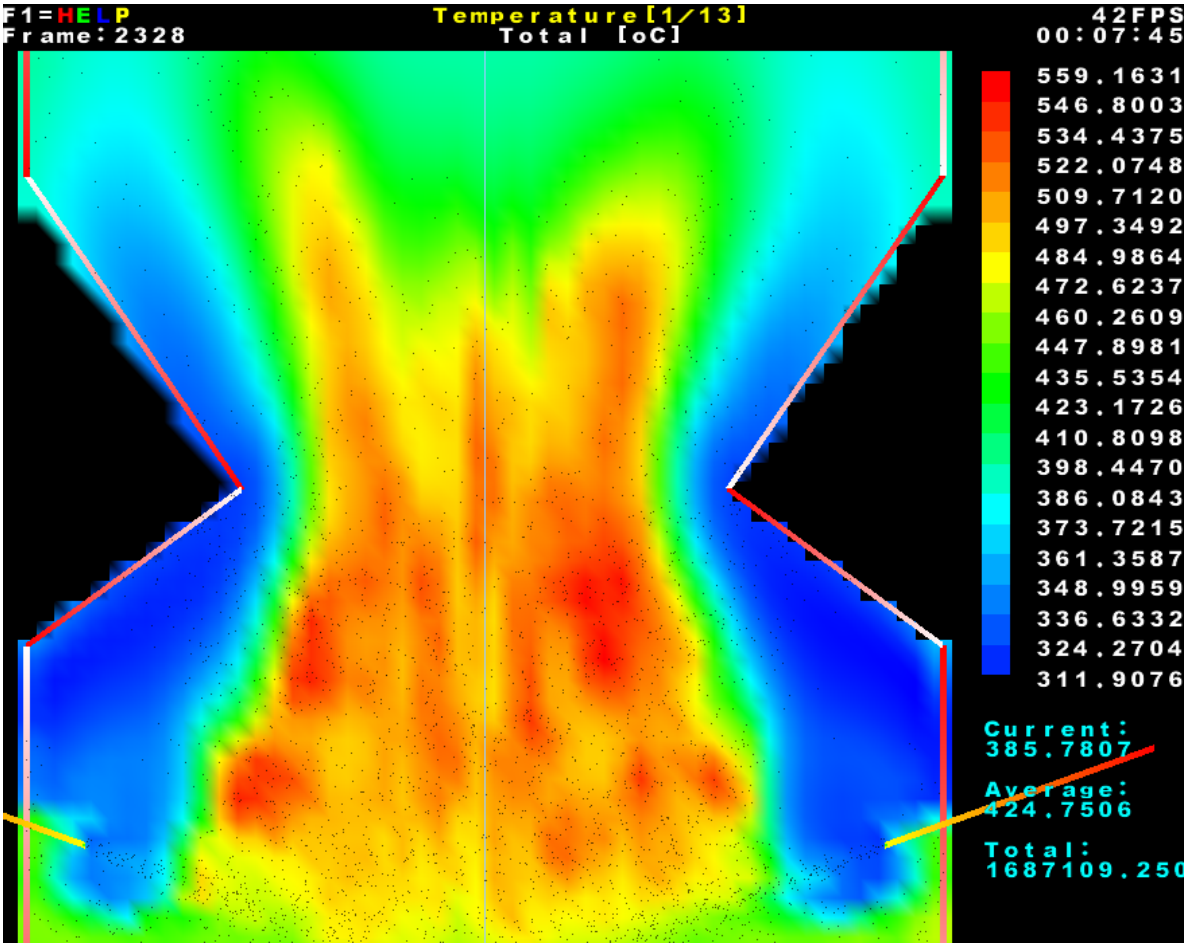


Figure 5-7: Sample visualization of temperatures inside the boiler

To compare our results with current CFD methodology, we have used the professional CFD software package FLUENT 5.5. Our system has been installed on a commodity PC with AMD Athlon 1333 processor, while FLUENT has been running on a SGI server installed on Department of Mechanics Engineering, at CTU in Prague.

In Table 5-1 we summarize the global parameter results of our current implementation in the comparison with FLUENT. Among the global parameters, we have compared the contours of the temperature and velocity maps. They are visually similar - see Fig. 5-9. Moreover, we are developing numeric comparison software that is able to statistically compare the results in each of the cells between our system and FLUENT and quantify corresponding errors and differences. Preliminary results from the active part of the boiler show that the share of cells with deflection up to 20% from values obtained from FLUENT varies between 60% and 80% for temperature, flow directions and other values. However, even these results can only be reached by setting manually the computation parameters and the multiplying constants before the computation begins, regardless the known physical condition inside the boiler. Note, that in that case we must first model this concrete solution in FLUENT solver, and then adjust those parameters manually regarding the particular task that is being solved by the model. Fortunately, this is not serious problem in education; on the other hand it

could cause problems designing boilers from scratch. The computation model would still need considerable improvements, to be used in production design.

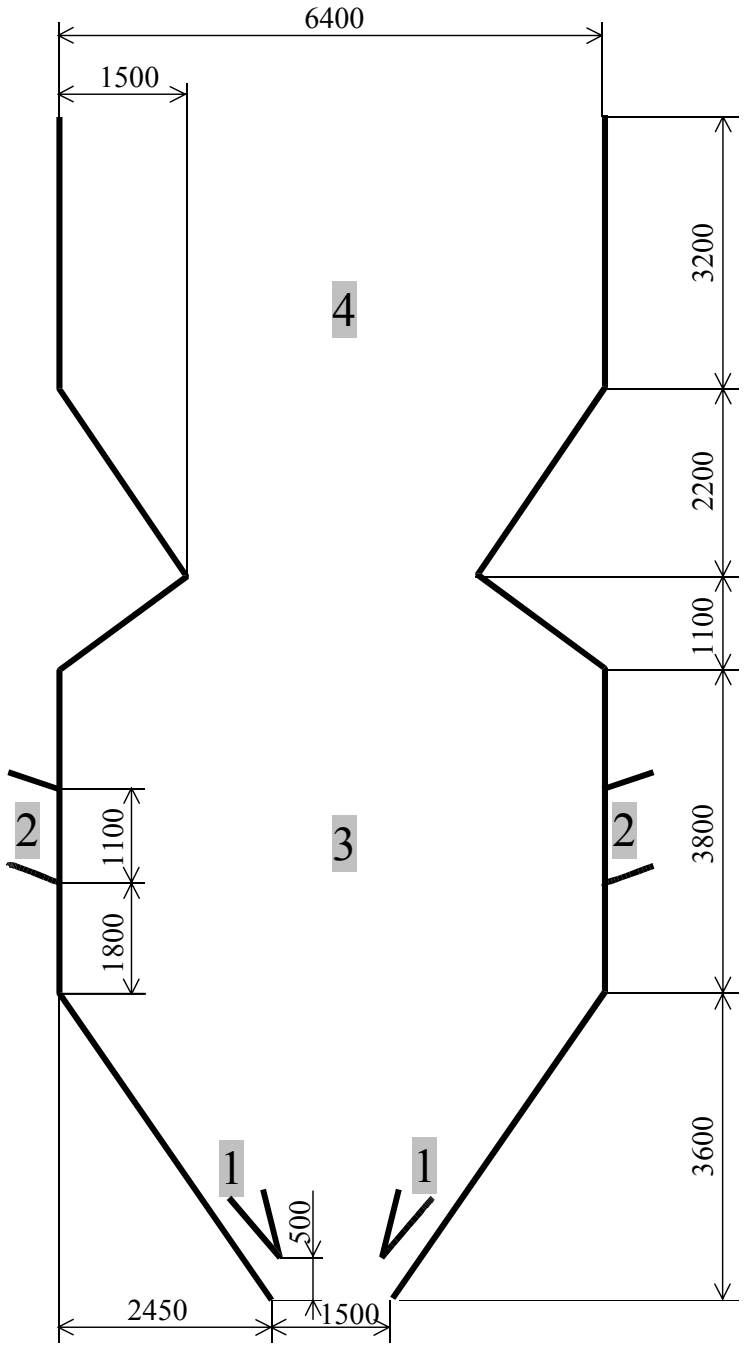


Figure 5-8: Shape of a real boiler on which the tests have been performed

5.9 Comparison of our results with the FLUENT solver

The comparison has been made to the FLUENT 5.5 solver. Both our system and FLUENT used the 2D space model. We used a grid of 1000 cells, which allows real-time performance of simulation and visualization.

Here is a summary of the basic parameters used in the FLUENT:

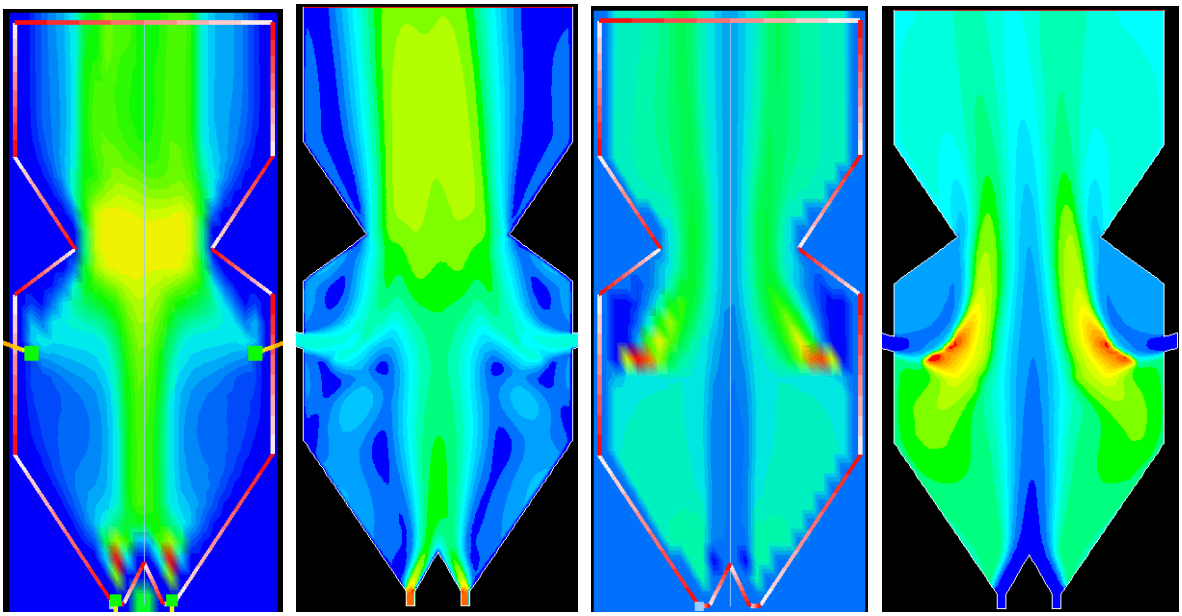
- Gambit pre-processor for the 2D grid modelling
- Solver – segregated, time steady
- Materials – PDF (probability density function) mixture created by the PrePDF pre-processor
- Viscous model – k – epsilon
- Radiation model – P1
- Segregated solution method

For the global parameters, which could be easily compared, the overall design and implementation of our ideas match well (see the Table 5-1).

Parameter	**Our system**	**FLUENT 5.5**
Average temperature	890 ^{o}C	1002 ^{o}C
Outlet temperature	814 ^{o}C	1068 ^{o}C
Maximum temperature	2546 ^{o}C	2488 ^{o}C
Average stream velocity	14 m/s	11 m/s
Average outlet velocity	25 m/s	21 m/s
Maximum velocity	56 m/s	48 m/s
Wattage	187 w/m^3	232 w/m^3
Mass total	21.1 kg	21.3 kg
Time in orders needed to converge a stable state solution on commodity hardware	**Minutes**	**Hours**
Time needed for obtaining one simulation frame	**Under 1/20s (real-time)**	**Seconds –Minutes**

Table 5-1: Global parameters results in the test boiler

The Fig. 5.9 reveal some differences compared to the CFD software packages (especially in the lower section of the boiler), compared to the real model. On the other hand, this drawback is balanced by the interactivity of the system. Many parameters could be altered during the simulation and thus fast and easy experiment with the system.



a) b) c) d)

Figure 5-9: Comparison of visualization output of our system and FLUENT: a) – velocities – our system, b) velocities – FLUENT, c) temperatures – our system, d) temperatures – FLUENT

5.10 An comparison with particle explosion modelling solved at Berkeley

After creating our pulverized coal combustion system and introducing it in [Gayer02b], we have found very interesting article describing similar effort made at University of California in Berkeley. Although that their work differs in several important aspects, we still highly recommend reader interested in using particle systems and fluids together (as we did) to read about their work in article [Feldman03], which is aimed to modelling of explosions, attempting to obtain realistic look of explosions, see Fig. 5-10.

They use both simplified fluid equations for mass and heat transfer, and velocities and press determination. They use millions of particles for representation of fuel and soot, for both simulation and visualization of explosions. For the visualization itself, unlike the traditional approach of modelling flames described in [Lamorlette02], [Nguyen02] and [Stam95], they use direct rendering of fuel and soot together with illumination model, including blackbody radiation, light in environment, shadow maps [Lokovic00] and illumination of other objects [Jensen02].

In contrary to our research (due to usage of large amount of particles 3D grid and probably choosing of MATLAB package for simulation, which cannot offer as high performance as optimized C++ code), their method is not real-time, and has goals of quality of results than the real-time performance. Simulation frame rate is in order of seconds and visualization rendering frame rate at about 1 minute per frame. However, their results are still interesting, as a very few authors refer successful projects, which use particle systems for both simulation and visualization. Their visualization output, see Fig. 5-10, well approaches reality. We can recommend also their

mathematical model as a possible alternative for consideration and analysis of building application pioneers searching for suitable implementations of fast simulation and visualization of combustion processes, or as a possible source of inspiration when trying to replace our fluid simulator and particle interaction code with more physical precise system of governing equations.

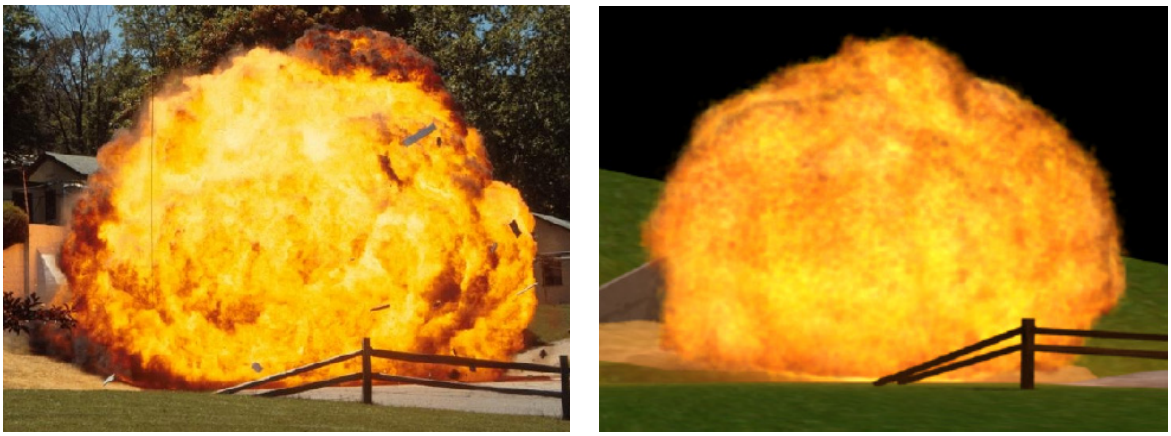


Figure 5-10: Left: Photography of real explosion. Right: Explosion rendered by [Feldman03].

6 Pre-calculated Fluid Simulator States

As described in previous chapter, we designed simple fluid simulator for fast data generation and subsequent visualization of combustion processes in pulverized coal boilers used in power engineering [Gayer02b]. We have extended the fluid simulator with virtual coal particle system. Particle systems are used for both simulation and visualization of combustion processes dynamics. In this chapter, we describe our effort towards the ability to use acceleration of interactive model of combustion due to visualization data generator based on Pre-calculated Fluid Simulator States. The acceleration is especially suitable for fluid simulators with additional data structures and governing equations – e.g. virtual coal particle system used in our case. By using more precise data-structures and equations, we face increasing demands on computational power in order to be still able to provide real-time performance.

In other words, after creation of a fast system for generation of data for real-time visualization as described above, we considered to enhance it. The goal was to find possibilities for creating visualization generator, which would be still fast enough and yet it would produce data that are more precise by either more precise computational grid or governing equations. The fluid simulator seems to be a considerable bottleneck for the real-time visualization performance. We have decided to enhance speed by developing special methods for acceleration of computation by storing acceleration data to disk device. By lowering the demands of computational part of problem, we approach to the goal of enabling real-time visualization.

6.1 Discussion of conditions of real time fluid simulator

Although many of the current fluid simulators offer real-time modelling and visualization, they are always limited by solution conditions under they achieve interactive frame rates. In most cases, it is determined by choosing low resolution or even 2D grid for computations. Interactive frame rate of the fluid simulators can depend on the type of simulated problem. For instance, required precision or expected increase of velocities and mass in the time moment, that could cause occurrence of possible instabilities of computation in the fluid simulators, must be kept in mind.

A typical example of these limitations is modelling of combustion processes, where due to sudden increase of temperature the pressure raises quickly, and thus the fluid simulator can be led to instabilities. This could lead to necessity for dynamic or permanent increasing number of time-steps in computation with the consequence of slowing down the simulation.

On one hand, drawback of slowing down the simulation due to increased time step can be in

some cases avoided by choosing stable fluid simulators, such as [Stam99], [Kass90]. These are more independent on the time-steps. On the other hand, stable fluid simulators are in general not only harder to implement but they also can have serious problems when using fixed obstacles (such as walls) in the computed scene.

For example, we have tried to adopt our system to stable fluid simulator scheme based on [Stam99]. Although this system is without doubt excellent for modelling nature phenomena such as liquids and clouds, we found it unsuitable for coal combustion processes due to problems with static obstacles.

Moreover, when using fluid simulators for real-time simulation and visualization in practice, the optimizations of the fluid simulator code to maximize the performance of the fluid simulator should be done as well.

True real-time feature of the fluid simulators is especially important in those applications, in which we need to present our results or CFD applications, e.g. in education. In such case, common known pitfalls of fluid simulators such as slow computation, disabling real-time visualization at all, *blow-ups* (serious and unrecoverable error in computation of fluid simulator), and local freezes caused by dynamic increase of simulation time-steps of fluid simulator should be avoided.

6.2 Accelerating fluid simulator performance

We have decided to design, investigate and test a new architecture of applications based on fluid simulators with partial support by pre-calculated states of fluid simulator stored on external storage. In this architecture, we benefit from today's both high capacity and performance hard drives in today's commodity PCs. This allows storing the generated data for later replaying and processing of the results. Instead of saving complete unsteady data sets (with complete core cell and particle characteristics that our system is able to visualize), we store only partial data, which results in orders of magnitude less demands on disk capacity and performance.

6.3 Fluid simulator state

Our fluid simulator state consists of the following arrays, which contains values of all grid cells:

- The velocity array [m/s]
- The mass of air array (can be easily recomputed to the array of pressures) [kg]
- The temperature array [K]
- The O_2 concentration array [%]

The values in these arrays represent overall characteristics of the cell. When computing values for the next step, the values from current state are being used. New values are computed from previous

solved state using internal fluid simulator code.

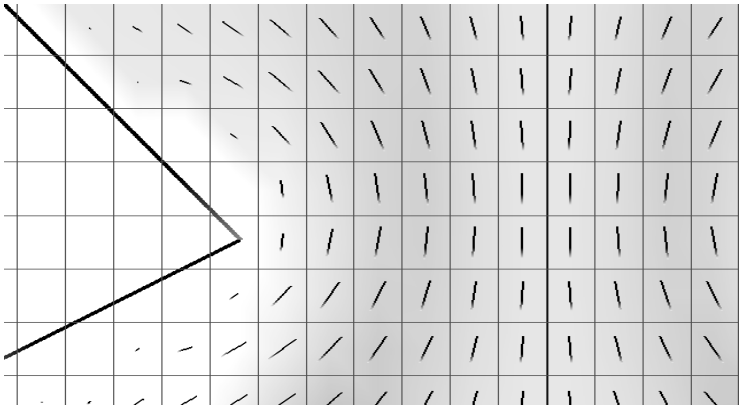


Figure 6-1: When we save velocity array, we save both of the vector components of x and y direction. When saving temperatures, we save scalar values of each cell (in figure, darker values correspond to greater values of temperature).

6.4 Fluid simulator system architecture

For schematic architecture of a possible solution of interactive simulation and visualization system for combustion processes, see Fig. 6-2. The particular parts of our system would be more described in the following text. The original approach is divided into the following steps (see Fig. 6-3).

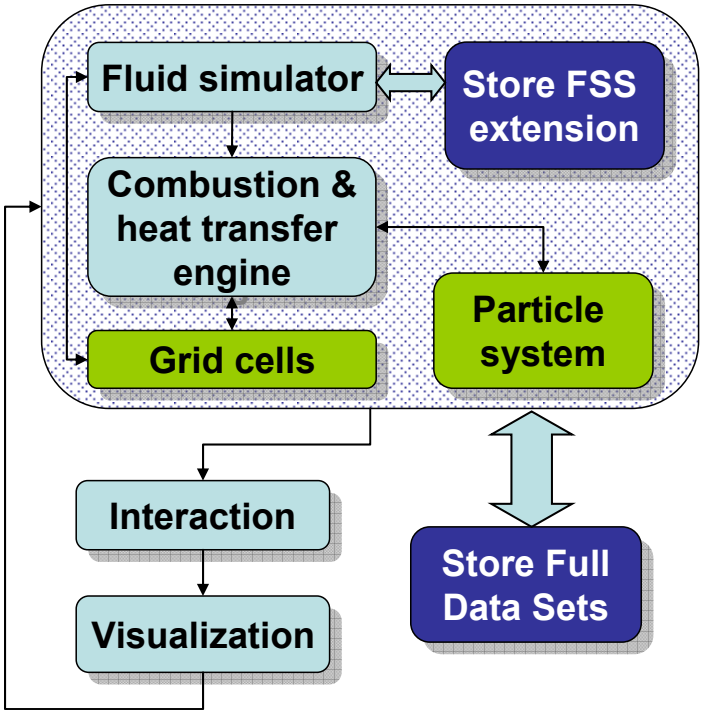


Figure 6-2: Schematic architecture of our interactive simulation and visualization system

```
While(Simulating)
     FluidSimulatorUpdate();
     CombustionEngine();
     Interaction();
     Visualization();
Endwhile
```

Figure 6-3: Original approach progress

The `FluidSimulatorUpdate` computes new velocities masses, oxygen concentrations and temperatures for the individual cells in the current time step.

The application control and computed values are then passed to the `CombustionEngine` routine to the simulator of coal combustion system. This part is independent on the fluid simulator code. It handles various actions, such as burning and moving the coal particles within the air present in the grid cell. `Interaction` routine reacts on user input, sets or modifies various solved task input parameters and boundary conditions, parameters and values. `Visualization` routine maintains visualization of computed cell values and particle characteristics. After that, time is increased by a selected time step and another iteration of the calculation is commenced.

```
While (Simulating)
     FluidSimulatorUpdate();
     CombustionEngine();
     if (Required)
          SaveFullDataSet();
          FullDataSetsSaved = true;
     else
          SaveFluidSimulatorState();
           FullDataSetsSaved = false;
     endif
     Interaction();
     Visualization();
Endwhile
```

Figure 6-4: Storing phase

6.5 Extending fluid simulator with pre-calculate states feature

Each state of the fluid simulator corresponds to values computed in selected time step. In

general, the state consists of variables (e.g. stored in arrays such described in 6.2). We extend fluid simulator by dividing the original progress from Fig. 6-3 to storing and replaying phase.

In the storing phase, we add the routine `SaveFluidSimulatorState`, which stores the computed variables and state of the fluid simulator, see Fig. 6-4. After storing the pre-calculated Fluid Simulator States, we can present results in the replay phase. For that purpose, instead of computing next fluid simulator state via `FluidSimulatorUpdate` routine, we restore the state from storage device using routine `LoadFluidSimulatorState`, see Fig. 6-5. Values from pre-calculated state are then passed to `CombustionEngine`, which maintains the rest of simulation, namely combustion and movement of coal particles. Thus, with support of pre-calculated values, only partial computations are performed. Thus, the simulation is replaced by interpretation of already saved results.

```
while(Simulating)
     if (FullDataSetsSaved)
          LoadFullDataSet();
     else
          LoadFluidSimulatorState();
     CombustionEngine();
     endif
     Interaction();
     Visualization();
Endwhile
```

Figure 6-5: Replaying phase

6.6 Storing complete unsteady datasets

Instead of storing pre-calculated Fluid Simulator States we can store all computed data to unsteady datasets (e.g. we store characteristics computed by fluid simulator such are velocities, concentrations, masses, temperatures together with characteristics computed using `CombustionEngine` - burned heat, heat transfers in cells and burnout ratio, diameters, masses etc. of particles located in cells). This is realized by calling routines `SaveFullDataSet` and `LoadFullDataSet`.

This could be useful for simulation, where there are further time-consuming computations, on the top of the fluid simulator. It is also suitable for further processing (e.g. out-of-core visualization), using either own code or an external real-time visualization system such as AVS [Upson89] or IBM Data Explorer or VTK [Schroeder96].

The selection of optimal variant depends on the type of task we solve, requirements on the storage devices, the performance of the device and the time/space requirements ratio between the fluid-simulator code and post-processing code (e.g. combustion engine). In most cases, the fluid

simulator part would be much more time consuming than the post-processing part or there would be no post-processing at all. In such cases, it would be necessary to store only pre-calculated Fluid Simulator States. However, in this case, at the first frame, the full data set must be also stored to allow restoring the complete state before replaying.

6.7 A demonstration example of our results

The following figures display example simulation results of our application based on FSS. Fig. 6-6 shows a captured frame of real-time visualization of cell temperatures together with floating virtual coal particles inside the boiler chamber area. Our application can arbitrarily select and switch among about 50 characteristics in the visualization phase. The results were simulated using our pulverized coal combustion application based on a simple fluid simulator. The same results (including switching to all other characteristics) can be gained using the pre-calculated Fluid Simulator States extension described. In such a case, the system runs up to six times faster. At the same time, it consumes only a fraction of the disk space, which would be required for storing corresponding complete unsteady data sets. In our test, it consumed only 1.6GB of disk space for Fluid Simulator States per 13 minutes animation, while for storing corresponding data sets, 19.1GB would be needed. The detailed configuration setup on which this demonstration example has been performed can be found in Table 6-1.

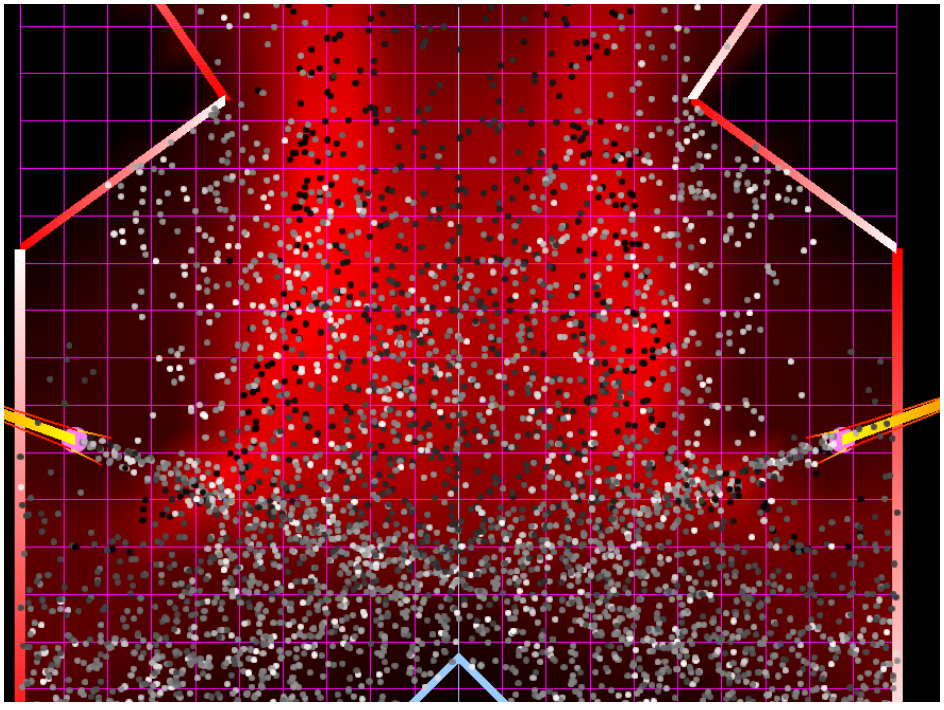


Figure 6-6: Sample visualization of selected cell characteristics together with floating virtual coal particles ran up to 6x faster with using FSS..

6.8 The results

We have successfully implemented our concept of pre-calculated Fluid Simulator States on a structured grid fluid simulator for simulation and visualization of combustion processes. We have simulated the starting of pulverized coal combustion in a 2D approximation of a power plant boiler (width 6.4 meters, height 13.9 meters). We have measured two configurations. In the first case, the grid was set to 20 × 40 cells. In the second case, the grid was set to 50 × 100 cells, and the computation code was more precise due to a decreased time step. Furthermore, we have compared the features, performance, and requirements of the following two versions: pre-computing only the Fluid-Simulator States (FSS) and storing complete unsteady data sets (UDS), containing both cells and particle characteristics.

For each cell, FSS contains four values (as described in Chapter 6.4) and UDS contains extra 10 values, namely those that are stored as result of combustion engine. We do not need to store them in FSS, because they are computed additionally due to described concept of FSS. For each particle, we store eight characteristics in our implementation.

Each of the cases contained about 12500 animation frames and average about 10000 particles per animation frame. The results are summarized in the following Table 6-1. In each of the cases, we got acceleration from 1.9 to 5.9 of the total amount of frames per second (FPS) compared to direct simulation and immediate visualization. Storing the data to disk drive consumed virtually no additional time cost compared to the version without storing the data (with fast Ultra DMA 5 data transfer mode used).

As a storage drive for pre-calculated states and complete unsteady data sets, we used 120GB Seagate Barracuda Ultra Ata V. Binary uncompressed data files were used for both storing FSS and FULL versions. All results and measurements were performed on a commodity AMD Athlon 1333Mhz system equipped with 512MB SDRAM PC 133 memory. The input parameters and results are summarized in the following Table 6-1 and Table 6-2.

6.9 The results discussion

In both cases, we have gained multiple acceleration of the fluid simulator based animation. Storing the data to disk drive consumed virtually none additional time cost. This is caused due to Ultra DMA 5 transfer mode of disk data into the PC memory with minimal assistance of system processor. Without pre-calculated states and data sets, the boiler simulation may run in full interactivity (e.g. changing of visualization parameters, moving the coal and air inlets, modification of various characteristics and simulation parameters such is the fuel value of coal).

Grid size	**20×40**	**50×100**
Interactivity	Full	Full
Seek & change simulation speed ability	No	No
Simulation time step set	0.005s	0.0009s
AVG Particles/Frame	10671	9173
Total frames	11999	13329
Simulation time	1212s	5090s
AVG frame rate [Fps]	**9.9**	**2.6**

Table 6-1: Pulverized coal combustion simulation start setup

Store method / Grid size	**FSS / 20 ×40**	**FULL/ 20 × 40**	**FSS / 50 × 100**	**FULL / 50×100**
Interactivity	Partial	Partial	Partial	Partial
Seek & change speed ability	No	**Yes**	No	**Yes**
Generation time	1214s	1230s	5128s	5133s
Write [MB/s]	0.16	8.0	0.3	3.7
Replay time	627s	603s	816s	864s
Read [MB/s]	0.31	14.6	1.9	**21.95**
AVG Fps	**19.1**	**19.9**	**16.3**	**15.4**
Disk space GB	**0.2**	**9.4**	**1.6**	**19.1**
Total Acceleration	**x 1.9**	**x 2.0**	**x 6.2**	**x 5.9**

Table 6-2: Results gained using pre-calculated fluid simulator state engine and full data set

When replaying the stored results, on one hand the interactivity is limited to visualization part only. We can change all the visualization parameters and methods, choose selected simulated area, immediate switch to any visualized characteristics of particles and cells, and more. On the other hand, in certain cases it is fully sufficient e.g. when presenting the simulation results in education.

When complete unsteady data sets are stored, it is possible to access randomly any time point within the rendered simulation. Moreover, play speed of visualization can be changed arbitrarily and

easily (by choosing skipping of selected frames) or could be even back-reversed. On the other hand, storing complete unsteady data sets is more limited to specific used storage drive.

When running demanding tasks with large cell grids, this method can suffer from both space and performance limitations of the drive. In configuration FULL (50 × 100), the disk read transfer reached 21.95MB/s. This is nearly approaching the limit of the current commodity disk drives read speed, which may be a performance bottleneck when replaying the simulation of even larger datasets. It may be avoided by storing selected frames (e.g. only each 1 from 10), to keep traversing in stored simulation fast. On the other hand, if we want to approach simulation with visualization in which one second of time of the model would correspond to one second in real time, we need to store only selected frames anyway. In addition, when saving results with large computational grid, we will still gain acceleration even after exceeding the disk speed. The reason for this is, that the computation on large grids is much more time expensive not only due need to calculate more data, but also due to need to increase time step to keep computation stable, as we had to did in case with the 50 × 100 grid.

When storing only the pre-calculated states of the fluid simulator, the disk space requirements are smaller than these needed for storing corresponding data sets with simulation results. Due to the low storage space requirements, this approach is also suitable for large, possibly even 3D grids with a particle system containing ten thousands of particles.

6.10 Comparison of FSS and unsteady data sets

Storing complete unsteady data sets is more limited by the specific storage drive used. When running demanding tasks with large cell grids, this method would suffer from both space and performance limitations of the drive. Then due to the speed limit of the drive, a performance bottleneck could appear when replaying the simulation. This could be fixed by storing only selected frames (e.g. 1 out of 10, or even 1 out of 100), but with losing the precision and resolution of the data stored. It is also not suitable to form the complete data sets for interactive, hierarchical structures, because the disk space requirements would be too demanding.

7 Pre-Calculated Fluid Simulator States Tree

We found our results described in previous chapter interesting. We have shown that the methods described may be used for various real-time visualization techniques. In case of need, very precise data (if someone would implement such precise fluid simulator or would like to accelerate already an existing one, even very slow and non-real time) of tens of characteristics can be generated in real-time and immediately used without demanding disk requirements. However, in spite of its advantages, an important requirement remains unaddressed by this solution: the need for not only the real-time performance, but also an interactive visualization that would allow investigating modifications of the modelled task. The previously proposed method offers to accelerate only linear fragments of the solved task not allowing any modification of modelled task. We address our effort towards acceleration of such tasks with possible interactive changes to solution in this chapter.

7.1 Introduction

Providing the real-time visualization of various physical phenomena and processes is a common goal of many research projects and applications. Real-time visualization offers many significant benefits to users of these applications:

- The results can be obtained for any selected time moment of simulation
- Possibility to get good a overview of the dynamics of the modelled process
- Easy manipulation and interaction with the simulated model
- User can make interactive changes to the simulated process with immediate visualization response
- Real-time visualization gives the results in readable, easily understandable and attractive form. This is very important in education and demonstration of these processes in general.

Without doubt, we could find many other benefits of the real-time visualization. Unfortunately, in many complex tasks, the full and namely precise real-time simulation, which is needed for data generation for visualization, cannot be always achieved easily and simply, even on high performance computer systems. A typical example is the modelling of various natural phenomena and manufacturing processes involving behaviour of fluids using CFD simulation techniques, e.g. for combustion modelling [Gillespie01]. In most cases, this is caused mainly due to a complex

mathematical and physical description of the process and resulting time-demanding simulation calculations. In these cases, researchers and developers of simulation applications are trying to find various concepts, algorithms and tricks to achieve all or at least some of the above described real-time visualization benefits.

7.2 Speed up and optimization of simulation

The most common method is to try to simplify the mathematical model or to choose a model, which can be effectively solved on existing computer architectures, and easily and quickly implemented. To give an example, for fluid simulations, various simplified forms of Navier-Stokes equations are being used in Fluid simulators and solvers. This topic is covered by hundreds of research papers dealing with simulation and namely computer graphics fields. The practical demand of such projects is very large, and results are very interesting for many researchers. Often, namely in commercial products, simulation code optimization (choosing proper data structures and effective and fast algorithms and other implementation issues) are very important. The performance difference between not optimized and optimized code can be measured in tens or even hundreds of percents.

Although the methods described so far can be often used with real-time performance in many applications, several possible drawbacks remain. First, simplification of the mathematical model leads more or less to reduction of accuracy of results. Often, various simplifications, such as low resolution of grid cells for calculations and a reduced number of particles are used. Some applications run in real time, but with necessity of lowering the time steps of computation of results. Other applications give visually very acceptable results, but with considerably decreased accuracy of simulation. Nevertheless, in many applications, this approach (including described simulation code optimization) is still not sufficient to allow real-time simulation at all.

7.3 Storing and replaying animations and unsteady datasets

Another common way for providing real-time presentation of results is dividing the simulation and visualization into two separate steps. In the first step, the data are calculated and stored on a storage device (e.g. hard disk). In the second step, the data are played back at interactive frame rates. One common solution to this problem that still has a real-time performance is to use a file storage.

The simulated data are either stored to the standard animation formats (such as AVI and MPEG) or to data sets. However, in this case we get several significant disadvantages. The first is total loss of interactivity. In case of animations, the interactive and other advantages of above described real-time simulation are completely lost. There is no possibility for further change in the visualization method. Only one of the selected characteristics can be visualized at a time. In addition, post-processing (e.g. presenting simulation results statistics or data-analysis in form of various graphs) is not available at

all. In addition, the quality of image is reduced (especially when a zoom into specified area in the animation file is required) and the disk space requirements are relatively large in order to keep the results of the simulation. It is virtually impossible to zoom even to some area of the simulated problem without losing the visualization quality. Many applications moreover visualize not only one parameter, but several ones, such as velocity, mass, temperature, pressure, heat fluxes, and many others and allow arbitrarily switch between them. The fluid simulator applications can also offer statistics features, which could summarize global and local parameters of the solved task (such are coal particle diameters) at selected time step upon request. Again, quick change to one of these parameters is virtually impossible in conventional image based animation formats.

The data sets allow storing of one or more computed characteristics. The interactivity is still limited; the additional changes to the already computed simulation setup and simulation configuration are of course not available. Moreover, unsteady datasets, if they contain several stored characteristics, have considerable requirements on the disk space. As well as in the previous case, the simulation is limited to the strict time portion based on the data set saved on the disk. On the other hand, datasets offer full interaction in the visualization part of the simulation (e.g. changing of visualized characteristic, zooming to any simulated area, offering various visualization methods, out-of-core visualization and meshing [DeCoro02], post-processing and others). Nowadays, high-capacity storage devices are often used to provide real-time visualization. They are used for volume rendering [Lum01], large data sets [Guthe04] and out-of-core processing [Cox97], [Bruckschen01] such as construction of streamlines and particle traces [Bryson99] and isosurface extraction [Chiang01] and for highly complex virtual environments [Klein02].

7.4 Our effort

Our research effort is primarily focused on real-time visualization of combustion processes. In our previous effort, we had developed a simple fluid simulator with combination of coal particle systems for modelling and fast visualization of combustion process dynamics in the combustion boiler chamber [Gayer02b]. This way, we were able to interactively and in real-time, simulate and visualize tens of various particle and cell characteristics, regarding the combustion process inside the boiler chamber (such as temperatures, velocities, pressures, mass fluxes, heat radiation and many others).

Currently, our application is able to run in real time at interactive frame rates, but with certain simplifications (namely setting low-resolution 2D computation grid and using only about in orders 10,000 particles). The setting of a higher resolution computation grid and computational time steps leads to a considerable slowing down of the simulation.

Therefore, in our current research, we have decided to design and implement an architecture, which would combine the benefits of interactive, real-time simulation, allowing for testing various boiler modifications, combined with utilization of speeding up the visualization using pre-calculated

results stored in data sets.

Instead of saving simulated results to data sets, we only save partial computations of the most time demanding part of the simulation. Thus, we save only the states of the fluid simulator data, and not any other results such as particle data. This results in a considerable speed-up of simulation and consequent visualization. Thus, our approach is similar to using data sets, but with drastically reduced disk space requirements. Next, namely low disk space requirements allow us to construct hierarchical structures forming those pre-calculated results. It allows for investigating various configurations and modifications of boilers with accelerated speed. Our concept can be easily reusable in other applications based on fluid simulators and solvers.

7.5 Forming FSS to tree cluster structures

The advantages, specifically low disk space requirements, allow constructing and storing Fluid Simulator States for even large simulations.

The same reasoning also allows for creation and organization of these states in hierarchical tree structures. In such trees, a node corresponds to one pre-calculated state. Each of the nodes can be extended easily and interactively by simply storing another pre-calculated FSS file. The new pre-calculated state uses as boundary conditions the results of the state from which it was derived (corresponds to the ancestor node). By choosing tree structures, we obtain many interesting features and advantages, which will be discussed later.

7.5.1 The nodes of the tree

Each node of the tree contains collection of FSS, which can be used to accelerate the generation of datasets. For example, a node can contain data for acceleration of 500 frames, which could equal of 10 seconds of simulation. Each frame also contains two complete states of the simulation stored at the beginning and end of the frame. These frames contain all volume characteristics, complete particle characteristics and global characteristics (such as heat conductivity of the walls). From these characteristics, the simulation can be completely restored and continue, either by running new simulation with accelerated FSS data (typically when the first frame is used) or by running without acceleration (typically when we reach the last frame of the node). This situation is shown on Fig. 7-1. In corresponding inert frames of computation in each the nodes, the simulation configuration such as number and position of inlets cannot be modified. We can do such changes only in the first frame of the node.

Each of the nodes has its unique ID of the tree where it belongs and unique number of nodes in the tree (e.g. node 2/1 in Fig. 7-2). This way, sequence of connected simulated task modifications are being defined in every node.

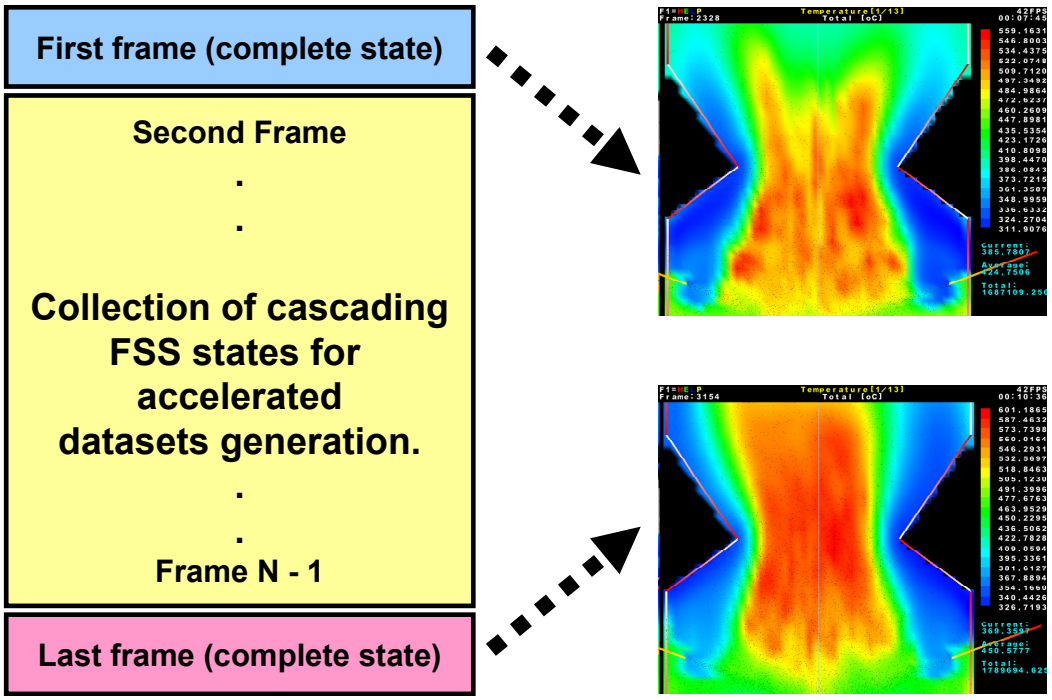


Figure 7-1: Stored frames in a node of FSS tree. First and last frames contain complete frames, while between them are frames containing only data for acceleration of datasets generation allowing to compute all simulation frames between the first and last frame

7.5.2 Traversing the tree and replaying the simulation

Replaying the path in the tree from top to bottom results in sequential replaying of connected saved states of the boiler with selected modifications of configuration in each of the nodes. After finishing running a part of the simulation, which utilizes pre-calculated FSS, the simulation can easily continue without any pre-calculated data (the last values computed using FSS are used as boundary conditions for further simulation, which runs at the original, not the accelerated speed). On Fig. 7-2, it would happen after reaching node with ID 2/15. Replaying of the tree based by interpretation of FSS states described in previous chapter can be for better understanding compared to replaying interactive DVD movie. In each of the node, we select a simulation configuration (equivalent to selection of various scenario possibilities), which will be further used. However, when replaying a node of tree consisting of FSS tree, we have possibility to choose interactively visualization parameters.

The whole tree can be visualized and implemented to accept interactive user actions and modifications – e.g. extending the tree with nodes for another part of simulation, deleting parts of the partial simulation solution etc. An example of an FSS tree is shown on Fig. 7-2 and Fig. 7-3.

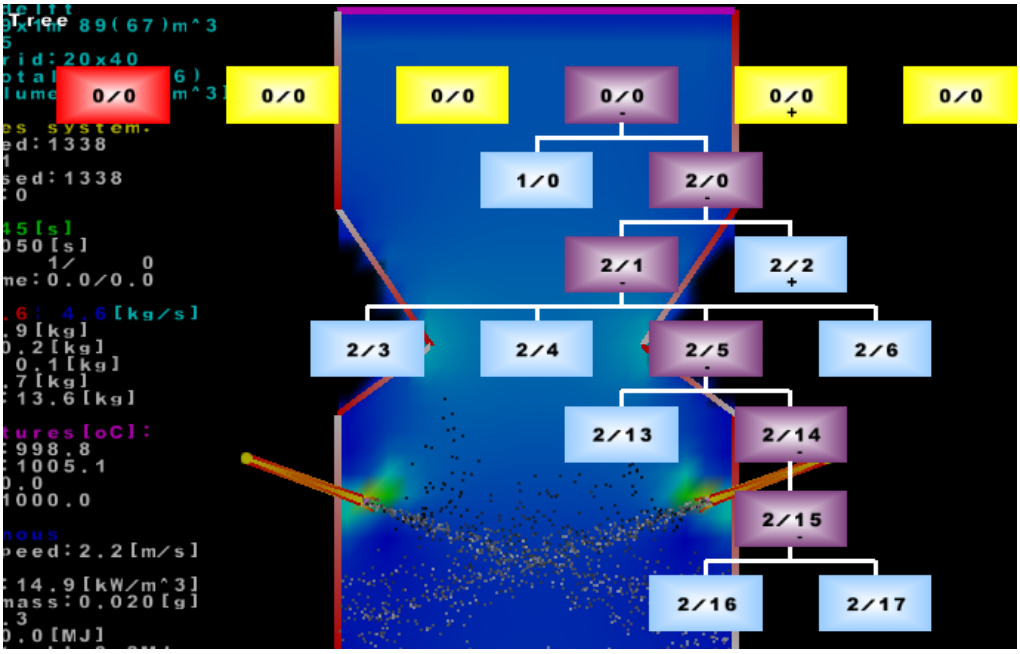


Figure 7-2: Hierarchical tree of pre-calculated Fluid Simulator States. Each node of the tree represents one file with saved FSS. The current selected path of simulation with configurations modifications, which can be replayed, is being highlighted. The nodes that contain – or + sign marks nodes that contain another subtrees.

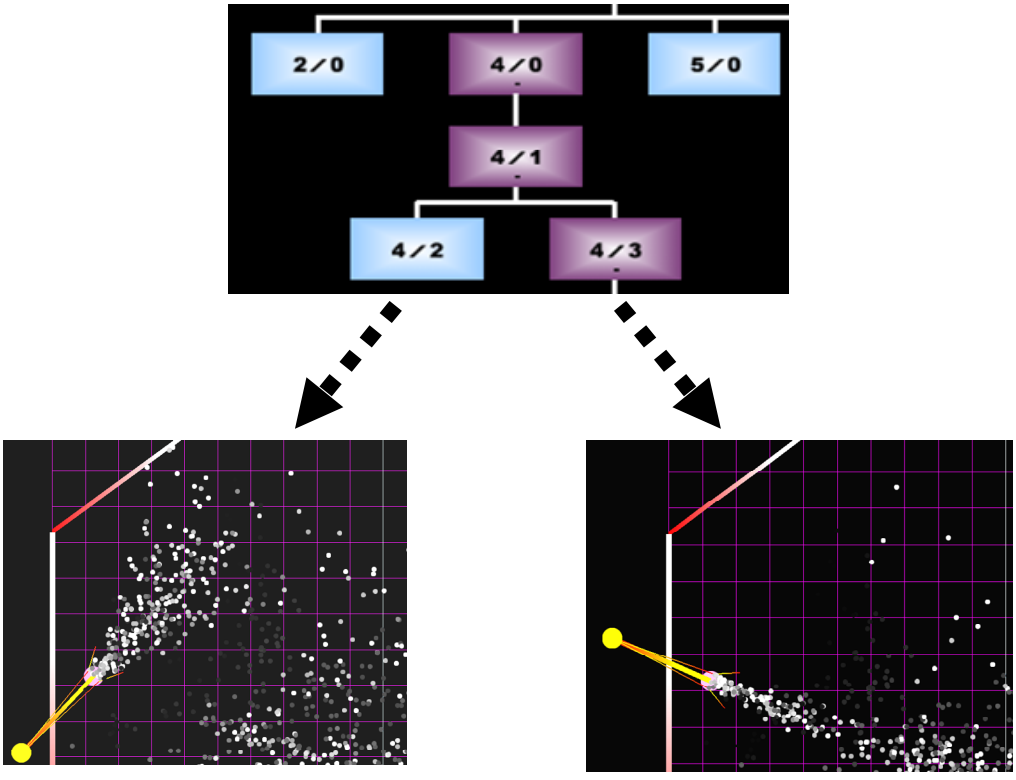


Figure 7-3: Illustration of interactive change of parameters of simulation in different FSS subtrees. In the left subtree (4/2) – the inlet is kept in its original position; in the right subtree (4/3) – the position and spread angle of the inlet has been changed. The FSS accelerated simulation continues with changed configuration.

7.5.3 Advantages of the described storage

This concept brings the following advantages over classical data sets:

- Considerable speedup of replaying results, while requesting only a fraction of the disk space requirements that would be required for the corresponding data sets.
- Incremental, step-by-step solution and obtaining the results of the simulated problem. This is advantageous for solution of the complex tasks.
- Possibilities of interactive addition, deletion of other parts of the solution, and modifications of the solved task.
- Hierarchical storage of states allows utilizing of previous values without needing to restart the simulation. Utilizing the pre-calculated states in ancestral subtrees save yet more disk space.
- Possibility to interactively change the simulation state and boundary conditions of the simulation in each of the nodes with immediate reflection (e.g. in each node we can change the coal inlet parameters, combustible properties, change the amount of oxygen and many more). This is shown on Fig. 7-3. In the first subtree of a FSS node, the simulation continues with the original inlet position. In the second subtree, we have changed the direction and spread angle of the inlet and the simulation continues with changed parameters with simulation acceleration being kept. The user decides which part of the tree (and corresponding configuration) would be replayed.
- Possibility of selecting and constructing various paths through the tree to replay interactively parts of the various simulation configurations. These paths can be separately replayed and each of them represents another interactive simulation with modifications and other actions of the simulation user reflected.
- It enables students and users of simulation applications to extend the prepared and pre-calculated simulation solutions with their own modifications while keeping a high, accelerated speed of the replaying.
- Selected nodes can be made read-only, to disable accidental modification and deletion of the base FSS tree structure, which is intended to be used by all students and other users.

7.5.4 Interaction

The entire data structure of the states tree allows implementation of interactivity with the model, – e.g. extending the tree with nodes for another part of simulation, deleting parts of the partial simulation solution etc. An example of a dataset tree is shown in Fig. 8-6. Each node of the tree represents one stored simulation dataset. Every interactively selected path in the dataset tree corresponds to one modified simulation solution. The current selected path of simulation with configurations modifications is highlighted (see Fig. 8-6). In Fig. 8-7, the ability of changing the

simulation configuration (boundary conditions) is being demonstrated. In the first sub-node, the simulation continues with the original inlet position. In the second sub-node, we have changed the direction and spread angle of the combustible injection from the inlet. Simulation continues with modified parameters with acceleration of visualization being kept regardless the interactive change of the simulation configuration in the node.

Selected nodes can be made read-only, to for example, disable accidental modification and deleting of the base tree structure, which is intended to be used by all users.

7.6 Storage implementation of FSS tree

The Fluid Simulator States for the corresponding FSS tree nodes are being stored in binary files. Every node of the FSS tree contains pre-calculated values of part simulation. Each node is also described by text files containing simulated case data and user modifications in the node, such as combustible and inlet properties. It also includes unique numbers of corresponding nodes and the node from which the node was derived from, to allow later reconstruction of the tree from the already stored files on the disk.

7.6.1 File representation and conventions

The binary dataset file consists of a header at the beginning of file, where fields such as number of total frames are stored. In addition, index of all frames at the end is present for faster seeking. After the header, the single frames are successively stored. The frames consist of arrays of characteristics of structured grid and all particles, which are needed to restore the particle system. We use 64-bit file access pointers to be able to work with files larger than 2 gigabytes.

For each node, we also keep a plain text file with basic configuration, which contains the initial simulated task boundary conditions. Applying these boundary conditions, we prepare the first frame of the simulation. Either we can continue the whole computation, or we can use the dataset to replay the stored data.

For the simulation tree nodes stored in our proposed file organization, we have specified the following file naming convection. Each file name consists of a number of the tree and number of the node. In the beginning, we restore the whole tree by reading the list of files, sorting them and then recreating the tree from the root to the bottom leaves. Extensions of the tree with new nodes are based on saving other files with appropriate names to the disk and extending the tree with new dataset. The example of a name would be test-002-007, where *test* is the name of the solved case, 2 is the ID number of the tree and 7 is ID number of the node.

7.6.2 Other implementation possibilities

The concept described above is only one possible solution, not a strict definition. It is also possible to use different solutions and data organizations, which would fulfil the needs of various ranges of solved tasks. For example, the list of the nodes, frames and the tree structure and boundary conditions and configuration could be stored in XML files, and the binary datasets of the whole tree nodes could be stored in a one file instead of many. Other storage, such as a SQL database based solution, could be used as well.

8 Hierarchical trees of unsteady simulation datasets

8.1 Introduction

As we mentioned above, the visualization of various physical and environmental phenomena and processes is a common goal of many different tasks. Real-time visualization offers many significant benefits, such as possibility to obtain quickly the results, the possibility to get a good overview of the dynamics of the simulated process and easy manipulation with the simulated model. In addition, it allows make interactive changes to the process modelled with immediate response and the visualization of the results in readable and easily understandable form. To achieve these goals, various concepts are used, which were described in the previous chapter (optimization, animation formats or our proposition of FSS Trees). In this chapter, we introduce another approach. This approach offers some of the features, that were gained using FSS Tree, such as fast generation of data for visualization and support for interactivity, but brings another – such as independence on fluid simulator, ability to skip visualization frames, arbitrarily change order of replaying saved frames. However, it differs in disk bandwidth requirements, depending of type of task we solve. It combines above proposed tree storage with conventional datasets. It can be used as a supplement or alternative for already developed FSS Tree, and will be useful in different types of the tasks solved. Upon this approach, we propose another special generator for our primary goal – real-time visualization of combustion and fluid processes.

8.2 Unsteady datasets

In many complex cases, the optimization is still not sufficient to obtain real-time visualization or it would lead to significant decrease of the computation precision. In such cases, the unsteady (time varying or time dependent) simulation results are stored in the datasets, which can be replayed and visualized in real-time, are used in various applications. Those datasets allow storage of one or more computed characteristics. The interactivity is limited; the additional changes to the already computed simulation setup and configuration are not available. In replaying part, we are limited to time portion based on the dataset stored on the disk. On the other hand, unsteady datasets offer full interaction in the visualization part of the simulation (e.g. changing of visualized characteristic, zooming to any simulated area, offering various visualization methods including out-of-core visualization, post-processing and others). The typical scientific area of this approach is the CFD – where we can easily have insight in spatial and temporal behaviours of various physical processes in any area of interest

[Bryson96], which is important for the applications such as combustion [Gillespie01]. For a detailed introduction and overview of the unsteady datasets, we refer the reader to [Cox99] and [Bryson99].

8.3 Our effort

We have proposed and investigated a new concept of storage representation of the unsteady datasets. It is based on hierarchical tree structures. The tree structures consist of complete unsteady datasets with simulated results stored (in each of the tree node). Storage of the unsteady datasets can be used in general applications. It allows to use some interesting features, such as possibility to increase replaying speed by both saving only one of several simulated frames to dataset (which can result in orders less disk space requirements) and skipping frames when replaying dataset. Instead of using the unsteady datasets, we can also use only the partial states to accelerate computation and visualization, such as pre-calculated Fluid Simulator States. Those were developed and used in our previous research [Gayer03b], but this approach is limited only to the solutions based on the fluid simulator and solvers , does not allows us to increase replaying speed, and does not allow us to save only selected frames. We had implemented the concept of unsteady datasets tree to our pulverized coal combustion application, but it can be also used in other applications.

8.3.1 The hierarchical datasets concept

The concept is similar to the already described FSS tree idea of the FSS tree. In each of the node, the user of the datasets tree can choose which of the connected datasets should be used. This way, the datasets can be replayed and created incrementally from any node in the tree. The replaying of the results is similar to replaying of interactive movies. It allows us to investigate and visualize various configurations and modifications. In each node, one of the sub-node datasets can be selected. It adds very important interactive dimension to the common datasets. We can replay the selected path in the tree from the top to bottom. It results in sequential replaying and visualization of connected stored states of the simulation with selected modifications of configuration in each of the nodes. After the playback of a dataset at a leaf node of the tree is completed, the simulation and visualization can easily continue without any dataset (the last computed values from the dataset are used as boundary conditions for further simulation) with optional availability of storing the simulated results to datasets. See Fig. 8-1 for a sample tree.

Important difference from the FSS tree is in the data content in the nodes. Instead of storing partial data for acceleration for the second and (N-1) frame in the node (see Fig. 7-1), we store the full simulation frame (same as those stored in first and last frame).

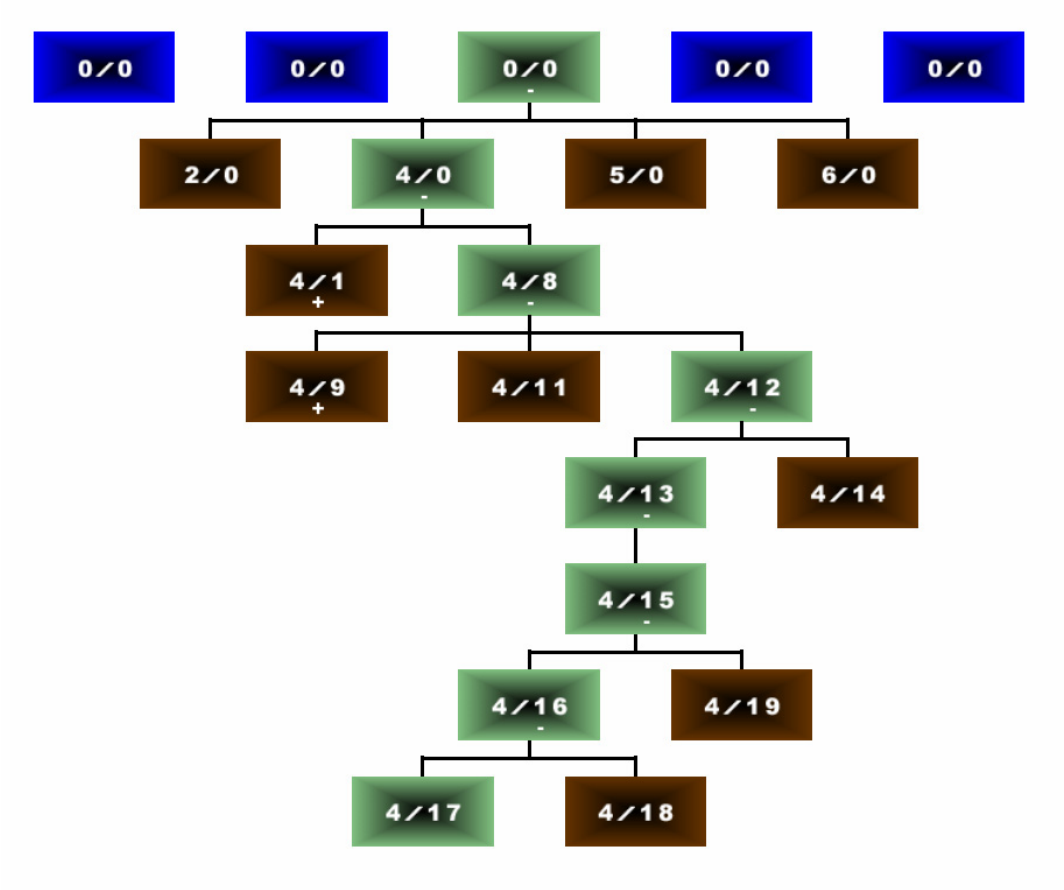


Figure 8-1: Hierarchical tree of simulation datasets – the data representation of nodes and selected paths is exactly same as in the case of FSS Tree. The selected path can be selected by activating nodes (by using left mouse click). The nodes containing "+" in bottom of the node are by activation extended and contains other parts of the tree, that are after activation selected as active part of the tree, and are marked by "-" sign at the bottom. The node without – or + sign are end nodes of the tree.

8.3.2 Incremental, reusable solution of the simulation task

This representation also features an easy modification of the simulation configuration for both the simulation solver (during solution) and simulation user (during replaying). Because each configuration uses the already computed simulation results in subtrees (and these subtrees can be used by several such configurations), it results in less disk requirements, which would be needed for all of the cases, if computed from the scratch. The structure and the content of the tree can be in any time extended and modified easily and interactively by simply storing another dataset. This allows incremental and step-by-step solution and creation of hierarchy of re-playable results of the simulated problem. This is especially advantageous for solution of complex tasks. The new dataset uses as the boundary conditions results of state, from which has been derived (corresponds to the ancestor node). Each of the nodes has its unique ID of the tree where it belongs and unique number of nodes in the tree (e.g. 4/15 in Fig. 1 and Fig. 2). This way, the sequences of connected simulated task modifications are defined in every node.

8.3.3 Visualization of the simulation data

Our concept allows the subsequent real-time visualization of the datasets. A wide range of general visualization methods and approaches suitable for unsteady datasets such as LIC, contour or tiling power spectra visualization [Robbins00], volume rendering [Crawfis00], flow volumes [Becker95] various types of streamlines [Jobard00] and other techniques [Lum01], [Bauer02] can be used. Pre-computation based visualization for other techniques, such as the motion maps [Jobard97] can be also used. In such case, the pre-calculated structures can be placed together with other data to datasets for selected nodes. This way, our solution could be possibly extended also for saving the whole particle structures for special visualization methods such as [Guthe02]. By modifying the structure and content of the nodes of the tree, possibly various encoding and data compression techniques e.g. [Deutsch96], [Guthe01], [Ibarria03] (even using its own data structures, such as special trees) such as [Machiraju98], [Ma98], [Ma00], [Shen99] and [Ellsworth00] could be used.

8.3.4 Zooming features and time navigation

We can set any zoom level and change the current view position to any area of the simulated object in our model. We can furthermore arbitrarily set the selected time position in the dataset and further replay it from that position. We can easily change the speed (slower, faster) or even back reverse the combustion process. It is important because we can easily seek to a particular part of the simulation as easy as we can set an area of concern in space. We can control the time resolution and minimum speed of replaying. For example, when creating an interactive hierarchical unsteady datasets for applications based on unstable fluid simulators [Foster96], [Gayer02b] we can increase the time step to make the computation more precise, and at the same time store only selected frames to be able to replay the simulation at the original speed without unnecessary frames. This way, the common drawback of unstable fluid simulation, which forces at the interactive level to slow computation and interaction, due to too small computation time step required, can be easily avoided.

8.4 Implementation recommendations

We had implemented our concept for the 2D datasets and only for those cases, when all of the frames can fit in main computer memory. However, our concept can also be used for large datasets, even those that would need out-of-core processing. In our case, we stored the data from the structured grid and particle system. The general implementation of the concept depends on the type of the task solved.

It is important to choose an optimal storage representation of the whole structure. There are many possible solutions. In our test implementation for combustion, we use a straightforward approach, when all the dataset for nodes are stored in the separate binary files.

8.5 Unsteady datasets tree applications

Using the above-described concept, we can implement interactive and fast visualization applications. The unsteady dataset tree has been originally implemented and tested for our fluid application for combustion processes. However, it can be reused in other simulation and visualization applications, which either use or can be extended to be using the unsteady datasets for storing the results. This could possibly include for example also the field of computer graphics, where tasks such as animation of liquids and water [Enright02] and gaseous phenomena [Stam99] are being currently solved.

8.6 Tests and measurements

We have successfully implemented our concept of hierarchical datasets to a structured grid based, unstable fluid simulator for visualization of combustion processes. This simulator has been described in [Gayer02b]. We have simulated the beginning of pulverized coal combustion in a 2D approximation of a power plant boiler. We have compared interactive features of direct simulation and visualization for the both performance and interactive features and requirements.

We have measured 2 cases. In the first case, the grid size was set to 20 x 40 cells. In the second case, the grid size was 50 x 100 cells, and the computation code was more precise due to a decreased time step.

As a storage drive for a tree of unsteady datasets we used a 120GB Seagate Barracuda Ultra Ata V. All results and measurements were performed on a commodity AMD Athlon 1333Mhz system equipped with 512MB SDRAM PC 133 memory.

The goal of all of these cases was to provide an interactive simulation reflecting changes of every 0.05 seconds in a boiler. The SIM case shows the direct simulation without of using hierarchical datasets. The UDT is the same case but with interactivity based on datasets trees. Because of using unstable type of fluid simulator, the simulation time step had to be set to 0.005s, which results in more frames being computed. Choosing less value would lead in the test case to computation instabilities and errors. The basic features of the task are summarized in the Table 8-1.

Features	SIM	UDT
Interactive change of the simulation	Full	In every tree node
Simulation time step set	0.005s	0.05s
Total played frames	10000	1000
Space shift and zoom	Yes	Yes
Time zoom features	No	Yes

Table 8-1: Setup and features comparison of direct simulation (SIM) and unsteady datasets tree (UDT)

Each of the cases contained about 10000 particles per animation frame. The grid size has been set to 20 × 40 and 50 × 100 cells. The results are summarized in the Table 8-2. In UDT cases, we received acceleration of 24 and 70 times of the total simulation replaying time compared to direct simulation and immediate visualization. Storing the data on a disk drive consumed virtually no additional time cost compared to the version without storing the data (with fast Ultra DMA 5 data transfer mode used).

Store method / Grid size	SIM / 20×40	UDT / 20×40	SIM / 50×100	UDT / 50×100
Simulation time	1089s	1103s	4617s	4665s
Write [MB/s]	N/A	0.7	N/A	0.4
Replay time	1062s	49s	4659s	68s
Read [MB/s]	N/A	15.9	N/A	26.4
Drawing speed [FPS]	**9.4**	**22.5**	**2.1**	**14.7**
Disk space [GB]	**N/A**	**0.78**	**N/A**	**1.8**
Total acceleration	**N/A**	**x 24**	**N/A**	**x 70**

Table 8-2: Results gained using direct simulation (SIM) and unsteady datasets tree (UDT)

8.7 Limitation of the datasets tree method

From the results of the measurements is obvious, that the amount of data stored in unsteady datasets is in our case has been limited by the specific storage drive used. When running demanding tasks with large cell grids, multiple characteristics and very long simulation time or frames, this

method would suffer from both capacity and performance limitations of the drive. Then, due to the speed limit of the drive, a performance bottleneck could appear when replaying and visualizing the dataset. For example, if we further request twice as data as our storage drive would be able to serve per second, the performance could drop up to half of the possible frame rate.

This could only be fixed by storing only selected frames (e.g. only 1 from 10 or even only 1 from 100 and more), but with losing the precision and resolution of the data stored. However, in many cases, such as in commonly used unsteady fluid simulators, it is not very important, because the computation step differs from the simulation step, and the saving only selected frames is often fully sufficient. In another cases, we can use other methods – either use the already proposed FSS tree method, or use direct simulation when possible.

8.8 Measurements with high number of particles

Here we present results of measurements that we performed to determine, what behaviour we could expect when using very large datasets in proposed UDS concept. Namely, we were interested by possible limitations that would be caused by performance limits of today's commodity computers and disk drives. For that purpose, we created datasets with large number of data per simulation/visualization frame - by choosing large number of particles (up to one million). We tested replay time of these datasets, with and without concurrent visualization. The grid size was set to 30 x 60 with time step 0.005s. We stored every simulation frame (we were not using frame skipping). Note that by using them, the acceleration of visualization would be correspondingly higher (e.g. up to 10 when saving only 1 of 10 frames). In addition, if we would choose large grids instead of particles, the performance gain when using datasets would be much higher, because both simulation and visualization of cells is much more time expensive than in the case of particles.

The tests were performed on 4 commodity computer systems:

- OMEGA – Pentium 4 3000 MHz, 1 GB RAM, GeForceFX 5500, Western Digital 2000 JD (Serial ATA), screen resolution 1280x1024x24bpp@60Mhz (see Fig. 8-2)
- BETA – Pentium 4 3000 MHz, 1 GB RAM, Matrox Millenium P650, Western Digital 2500jb (Ultra ATA), screen resolution 1280x1024x32bpp@60 MHz (see Fig. 8-3)
- NAUTILUS – Pentium III, 1000 MHz, 512 MB RAM, GeForceFX 5600, Western Digital 400BB (Ultra ATA), screen resolution 1152x864x24bpp@85Mhz (see Fig. 8-4)
- VISUAL2 – Athlon XP 2000+ (1733Mhz), 512 MB RAM, GeForce 4 TI 4200, Seagate Barracuda ATA IV, screen resolution 1280x1024x24bpp@60Mhz (see Fig. 8-5)

We used seven datasets, see Table 8-3. The number of particles per frame slightly varied, average number of particles is shown in the table as well. We also measured the performance in

Frames per Second, for various counts of particles, that each of the computers had to visualize (without any simulation, in paused state). For that purpose, we used OpenGL Points (stored and not stored in vertex arrays) and standard OpenGL Point Sprites using standard ARB extension, which offer better visualization results then the OpenGL points, see Fig. 10-3 (not available on BETA). This measurement would further answer question of approximate number of particles that are reasonable to run on today's hardware when desiring real-time visualization (with our system and eventually on others that would be based on our concepts).

The graphs with measurement revealed the following conclusion:

- On today's commodity hardware, it is not possible to expect real-time performance when using more then 300000 particles. On our most powerful computer OMEGA, we received 24 FPS, but it was while the calculation was paused.
- Except the NAUTILUS computer, we did find a remarkable performance bottleneck caused by the disk drive system. The current speed of disk drives, which is about 20 – 30 MB's per second, was sufficient to supply data for hundreds of thousand particles per animation frame. This was, however not the case of disk drive used in Nautilus computer. The additional measurements showed that read speed of this disk drive was only about 5BM's per second. This started to be as serious performance bottleneck, when performance with dataset was up to 3x times slower then without it.
- With increasing number of particles and amount of data to transfer, the acceleration of simulation dropped on all systems
- Optimal amount of particles with decent performance, when using unsteady dataset (with exception of already mentioned NAUTILUS, and probably similar systems with slow disk system, such as notebooks) is maximally 100000 particles.
- Due to out choice of OpenGL and GLUT, FPS in a test of particle performance measurement was never greater then the display refresh rate (85 MHz for NAUTILUS and 60 Mhz for rest). Thus, the value 85 and 60 for FPS in graphs mean "at least 85 and 60". It also shows that the framerate is fully interactive in this case.

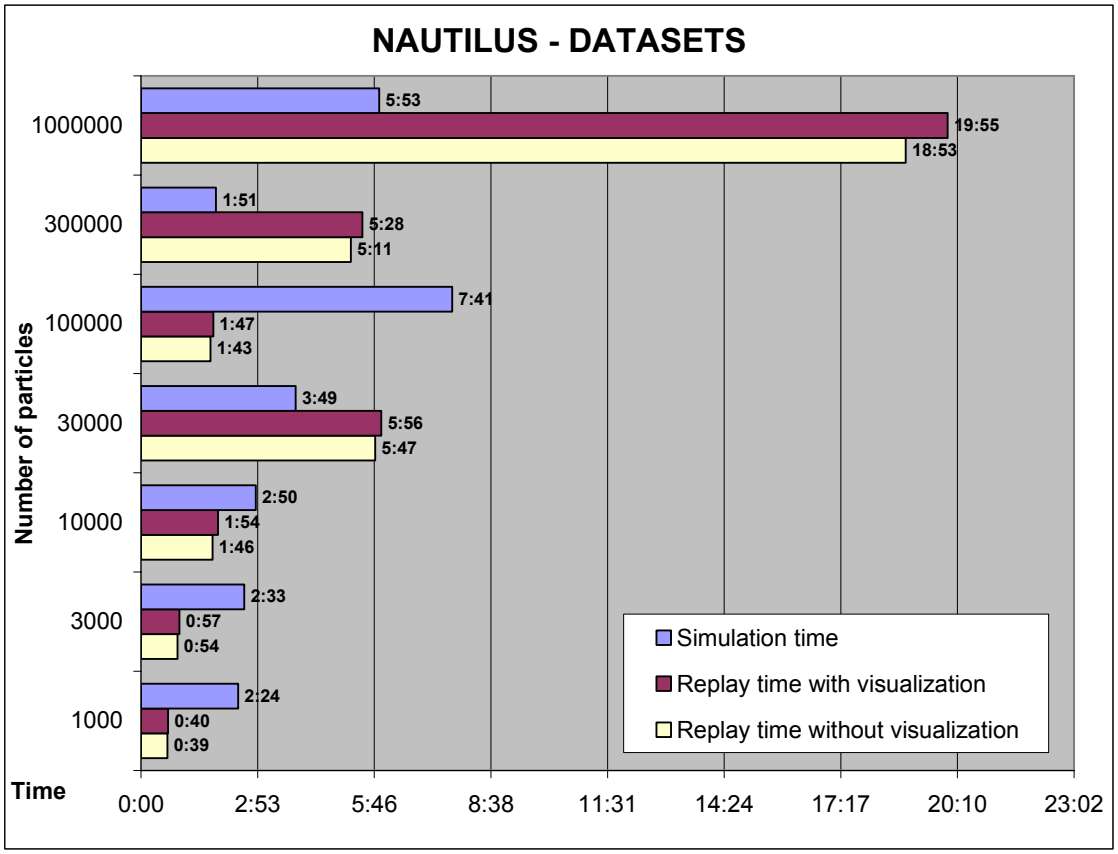
8.9 A demonstration example of our results

The following figures display examples of the results of our pulverized coal combustion simulation system based on unsteady dataset trees. Our application can arbitrarily select and switch among a total of about 50 cell and particle characteristics and statistics in the visualization phase. Fig. 8-6 shows a captured frame of real-time visualization of cell temperatures together with floating virtual coal particles inside the boiler chamber area. The results were directly obtained using our simple fluid simulator [Gayer02b]. The same results and features (including switching to all other

characteristics and statistics, see Fig. 8-7) can be gained using the unsteady datasets tree, with possibility to interactively change the simulation configuration in each of the node. In such a case, the system runs in orders faster than the original simulation. Because of the ability to save only necessary frames in fluid simulation, the disk space demands are acceptable. In our test, it consumed 1.8GB of the disk space for Fluid Simulator States for 68 seconds long animation.

Number of particles in beginning	Size of the dataset	Total frames in dataset	Average number of particle / frame
1000	131MB	1000	1001
3000	191 MB	1000	2381
10000	418 MB	1000	7657
30000	1411 MB	1000	30779
100000	422 MB	100	96136
300000	1249 MB	100	288688
1000000	4570 MB	100	1061627

Table 8-3: Results gained using direct simulation and unsteady datasets



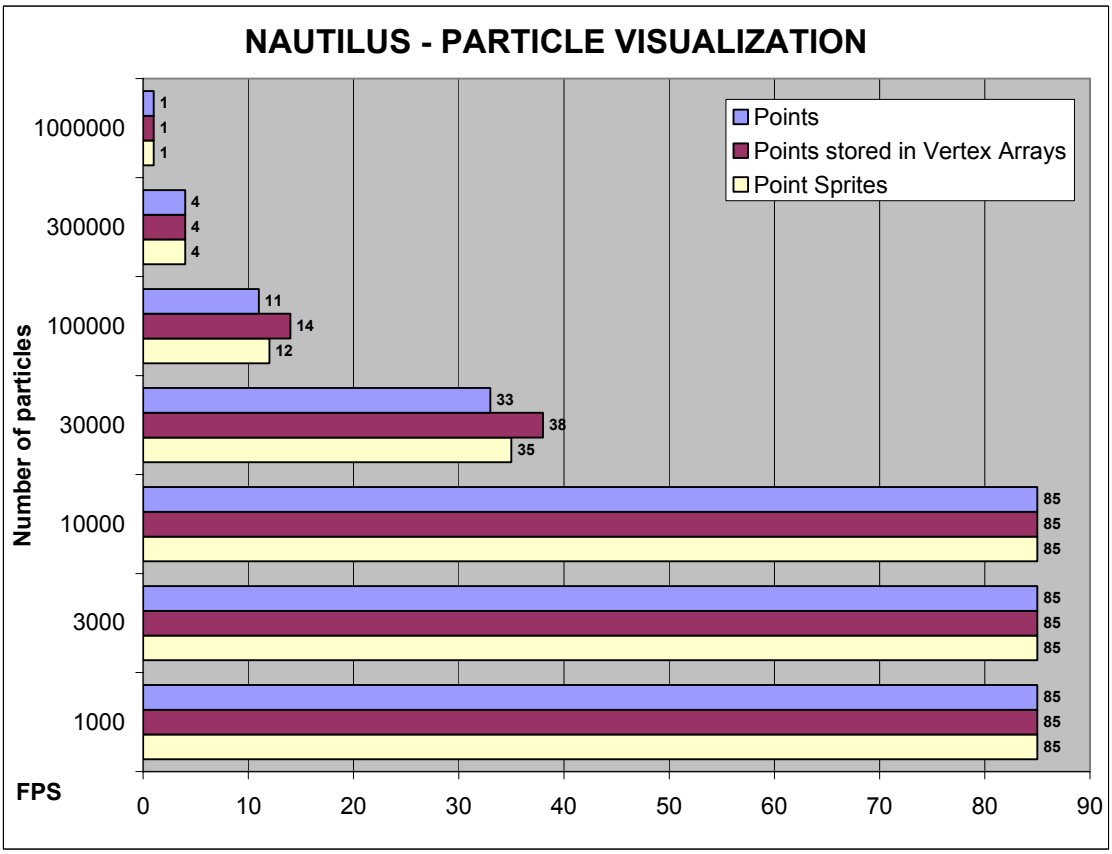
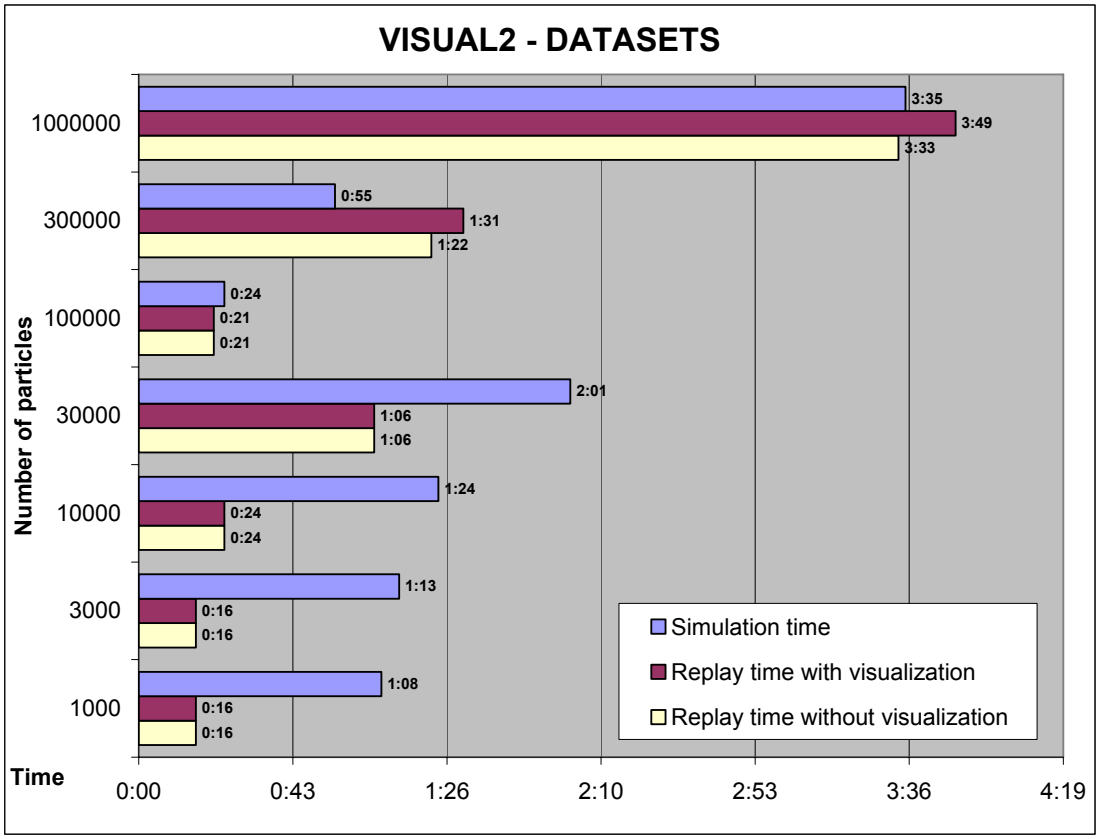


Figure 8-2: Measurements of datasets and particle visualization performance on NAUTILUS



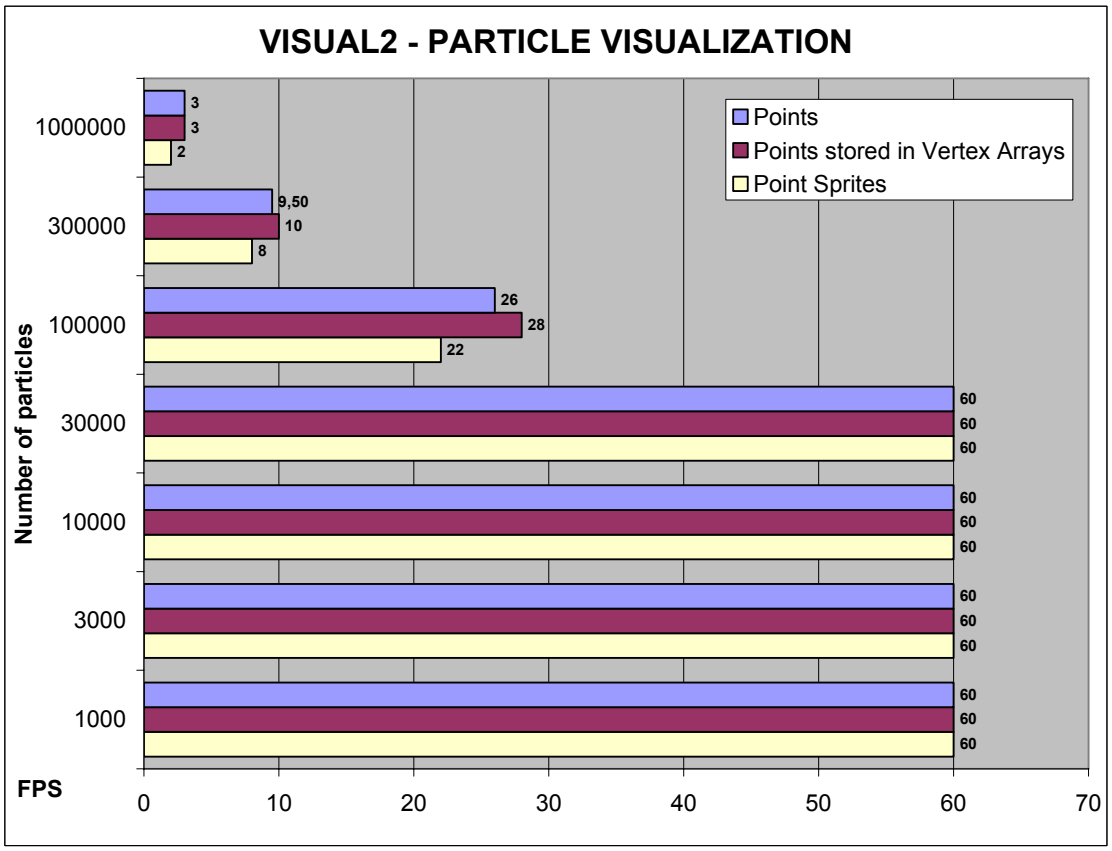
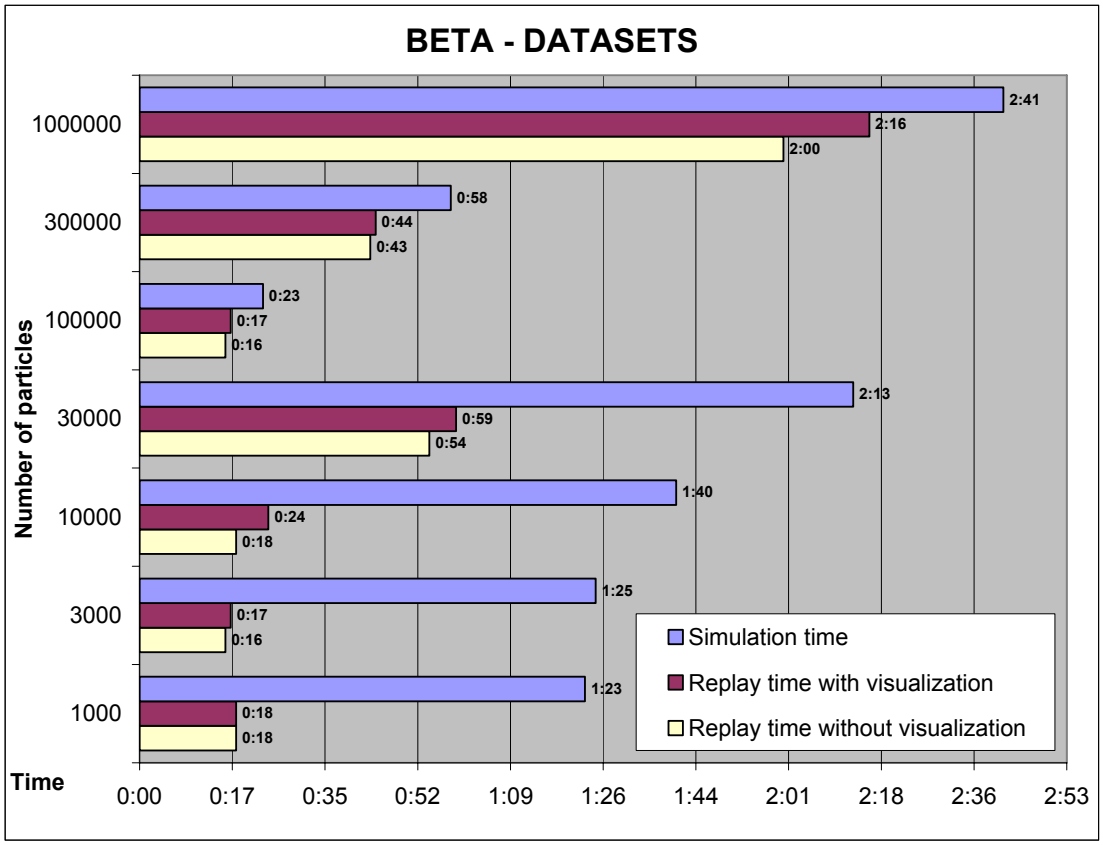


Figure 8-3: Measurements of datasets and particle visualization performance on VISUAL2



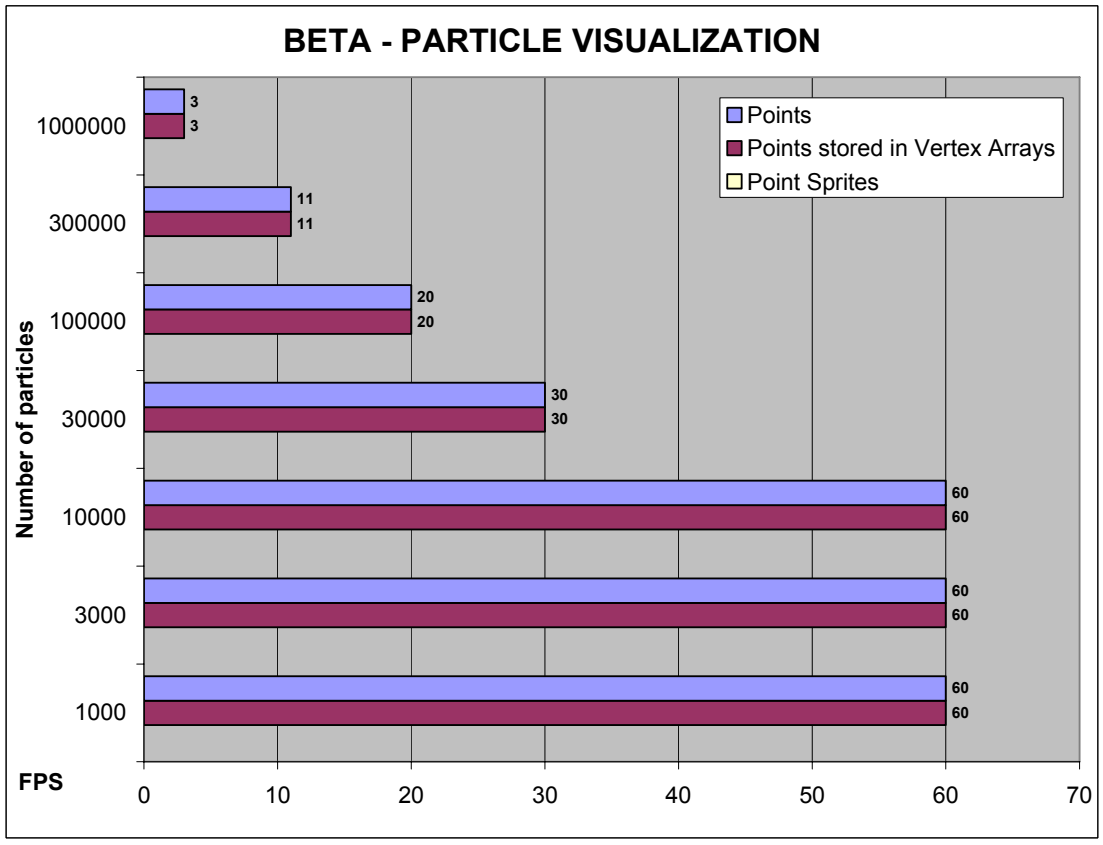
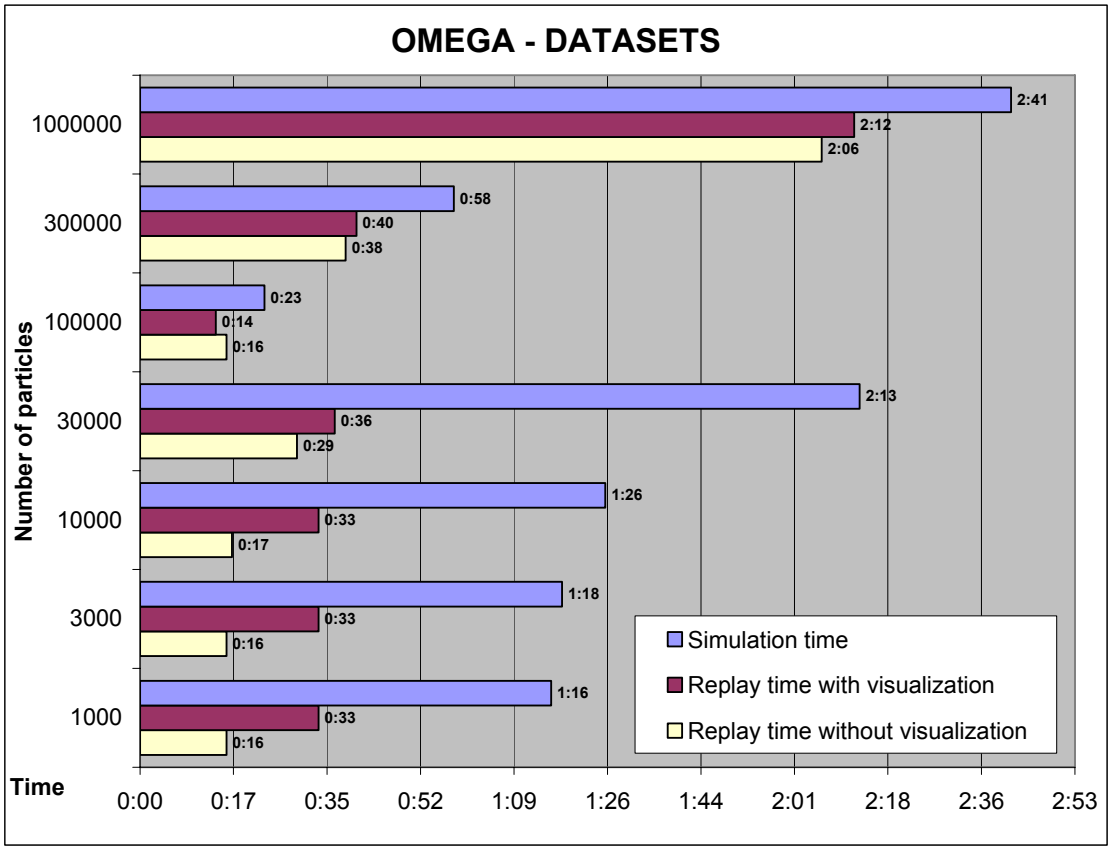


Figure 8-4: Measurements of datasets and particle visualization performance on BETA



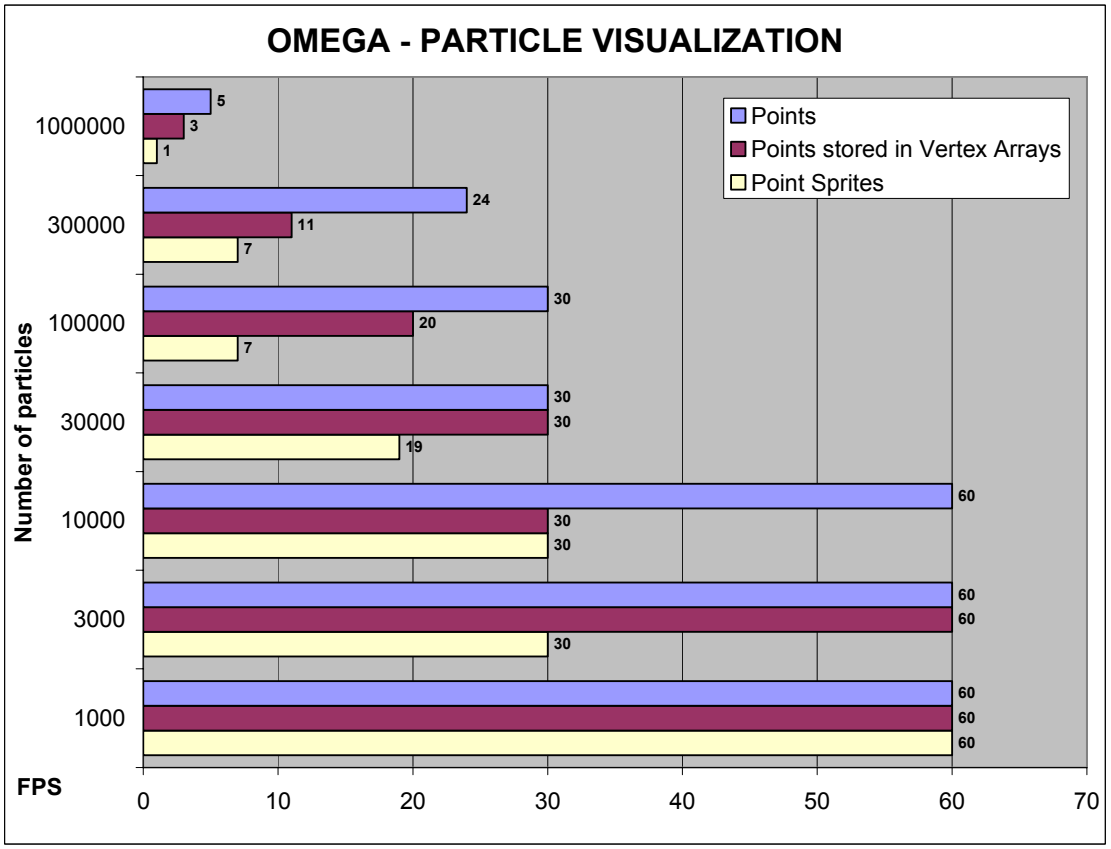


Figure 8-5: Measurements of datasets and particle visualization performance on OMEGA

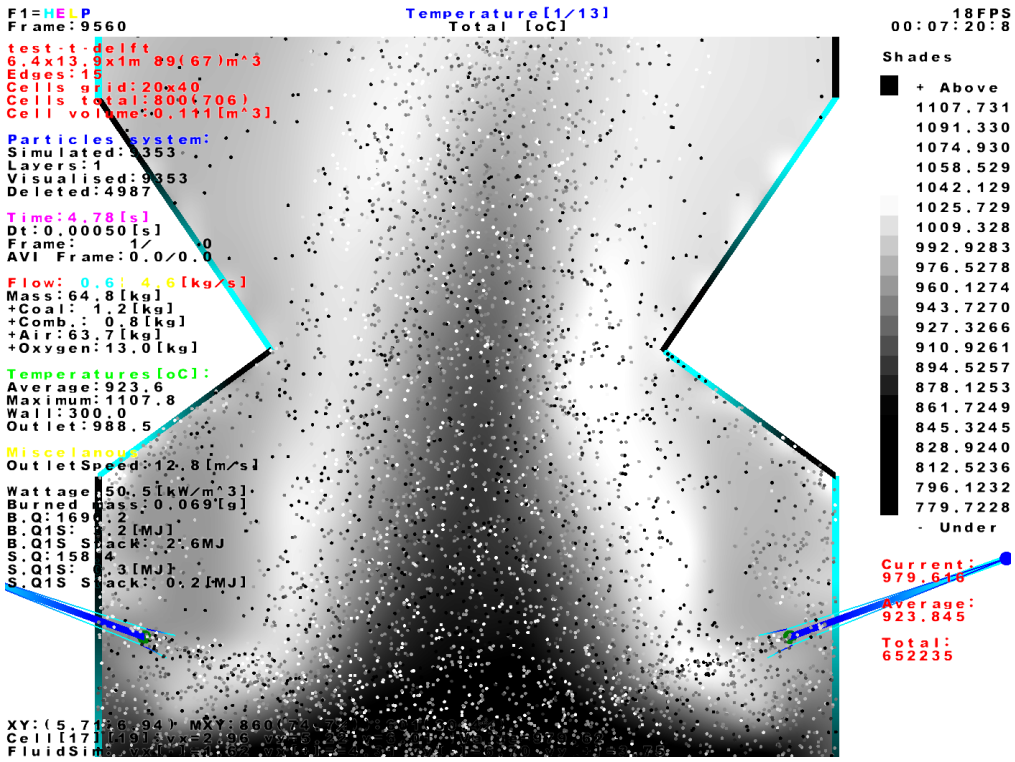


Figure 8-6: Sample visualization output of our coal combustion simulation application based on Unsteady Datasets Tree.

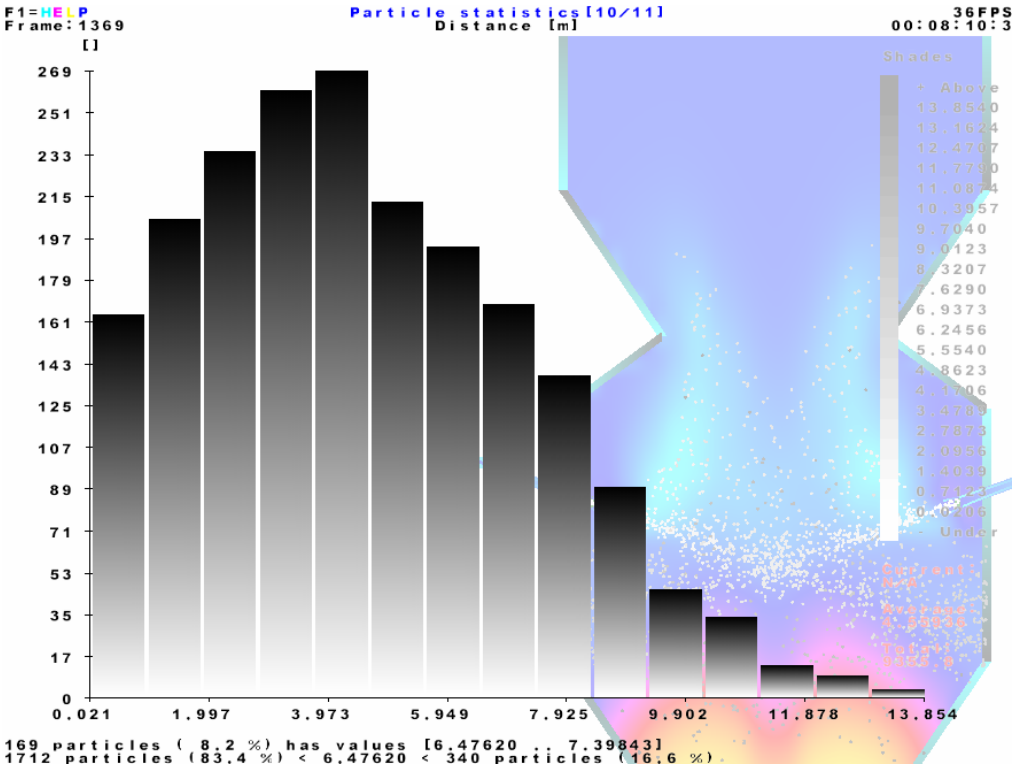


Figure 8-7: Our application runs using unsteady datasets tree enabled in orders faster than without them

9 Flow visualization using hardware accelerated spline interpolation

9.1 Introduction

In previous chapters, we proposed several ways, which allow us to design and implement fast data generators, suitable for real-time and interactive visualization of these processes. We have also proposed the use of simple grid visualization together with particle systems as a methodology giving good dynamics overview of these processes. Instead of grid visualization, one can also use the existing methods, some of which has been described in introductory chapters of the thesis. We have also decided to find an original, new solution (namely, when we speak about performance and design for current graphics hardware), which would help us to reach good visualization quality results suitable for many computed characteristics, that the above-described methods generate. In this chapter, we introduce our approach to reach high visualization quality of cell grid in real-time using hardware accelerated visualization. We notably improved speed of the spline interpolation, used in computer graphics for scalar visualization towards our needs of interactive, real-time visualization.

Many research projects and applications demand real-time simulation and visualization of various processes. Real-time visualization offers many significant advantages. Unfortunately, when using a traditional approach, the speed of the simulation is limited because of the requirements of the visualization quality. In other words, we have to choose whether the CPU will spend time calculating the simulation, or displaying it. In addition, if we want to have an outstanding visualization, the speed of the simulation must be sacrificed. Today, we have an option – if we utilize capabilities of current graphics hardware, we may achieve both visualization speed and quality.

A typical example of a visualization method is a Cartesian grid containing values computed by the simulation. We want to display the simulated values in a way, which helps us to see the most important features of the modelled situation; such are various fields in combustion boiler.

9.2 Splines, isolines

The simulation (or other source for data we have) results in ordinal, numerical data values. We discuss the case when, the data are stored in 2D or 3D computational grid. We have to convert them to some visual attributes, usually colour. This mapping can be direct, or indirect, using texture. Both methods work quite well when the grid values are almost linear. However, even if this is the case, the way the rendered values are interpolated [Heckbert91] may introduce artifacts into the resulting image.

A typical example of such artifact is a triangular pattern, which occurs at those grid cells that are not "planar", i.e. linear in both grid dimensions, such can visible on Fig. 9-6. In order to eliminate this weakness, we do not use the conventional linear interpolation; we apply spline (cubic polynomial) interpolation instead. Because we require such non-standard mapping technique, we cannot use the standard texture mapping equations that are implemented in graphics libraries and hardware. In this case, our only feasible option was to write a custom software renderer – in which case we could not utilize modern, contemporary graphics accelerators features.

If we use a texture to convert the data values to colour, we immediately have one more capability at hand – the isolines. Isoline drawing is one of the best methods to show up the features of a model in many applications is. This simple notion has a very good effect for understanding the model. (That is probably the reason why the isolines are used widely [Gilmartin81].) There are numerous algorithms to determine isolines of a value array [Kreveld94], [Laszlo92]. Their main drawback is apparent: they are not designed to work in real time. Today's hardware may be able to perform similar algorithms in real time, but at the price of the CPU computing power, dedicated solely to a single visualization feature. If we are willing to trade the accuracy of these methods for the performance, we can use simpler methods that offer only an approximation of the isolines, but with substantially smaller computation requirements.

Our method belongs to such category. We use a pre-calculated texture that is mapped to the displayed grid in such a way that the texture coordinates corresponds with the value stored at the respective points. In the resulting image, the isolines appear at those points, where the interpolated texture coordinates match the isoline value.

Again, if the used texture mapping would apply on the linear interpolation, the isolines would appear quite jagged. However, as we can use the capabilities of modern graphics accelerators, we are able to perform the interpolation using splines.

9.3 Modern graphics hardware

Modern graphics hardware offers more processing power than the graphics hardware used in the past (more triangles per second, higher frame rates, more textures, higher resolution, etc). These improvements are immediately available to any user that posses a modern graphics accelerator. The programs just run faster in the moment you replace the hardware. Although, it is of course good to have more processing power available, this is not the effect most important of the innovation. The best improvement is the programmability of graphics accelerators. An accelerator now allows us to change its mode of operation in a variety of ways. We can change the transformation and lighting calculations done by the accelerator, and we can change the texturing calculations. In addition, whatever program we pass to the accelerator, it runs directly on the graphics processor, separately and simultaneously with the conventional system CPU.

By utilizing these features, we can write a custom texture-mapping program for graphics hardware, performing spline interpolation instead of linear and send the program into the accelerator to be run on it. Using this interpolation, the isolines rendered by the aforementioned method are (although still equally fast and simple) far superior to the isolines rendered using the linear interpolation. The same is true about the overall quality of the rendered grid. All features of the image are smooth without blocky appearance caused by linear interpolation of non-linear data.

9.4 Our effort

In the process of rendering grid-structured data, we have to answer the following three questions:

- How to convert the numeric data into some form that can be represented visually
- How to interpolate the values between the grid points
- How to render the resulting data both fast and with high quality

Our method addresses all these issues: The conversion from the numeric data into visualizable parameters is done using textures. When the visualized data represents one-dimensional values (like temperature), the texture used is just one-dimensional (we could use two- or three- dimensional textures to visualize structured data). Data values in the grid are interpolated using spline (bicubic) patches. This interpolation is simple enough to be very fast, and still results in high quality images. The third topic is quite a problem for traditional methods that suffer from the contravening requirements of speed and quality.

A typical modern graphics accelerator contains a Graphics Processing Unit (GPU), which is a programmable processor capable of doing all computations required for the traditional rendering. It is worth mentioning that the GPU is a standalone hardware device that is capable of doing calculations completely independently of CPU. This results in a desirable new level of parallelism that allows us to relieve the CPU of some low-level repeated work and let it do the “hard” computations for the simulation.

The GPU is programmable in a number of ways: The two basic features are vertex programs and fragment programs (also called vertex and pixel shaders). A vertex program is a replacement for traditional graphics transformation and lighting calculations. A fragment program is a replacement for traditional texturing computations (see Fig. 9-1). We can use only vertex, or only fragment program, but we have most possibilities when we use them in cooperation.

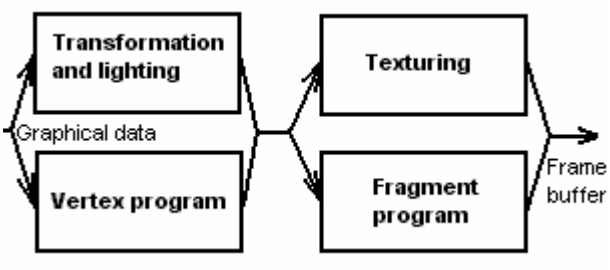


Figure 9-1: Vertex and fragment programs in the rendering pipeline

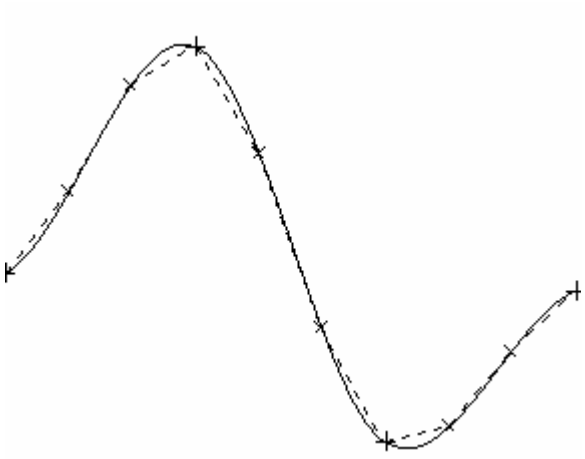


Figure 9-2: 1-D illustration of spline evaluation at discrete points

In general, the vertex and fragment programs can do any computation the GPU is capable of; but there are also some disadvantages: the GPU has limited precision (this is not an issue for our use). The computation results are retrieved with difficulties (we display the results directly, so that we do not need to retrieve them explicitly); and the vertex (fragment) program must be executed for each vertex (fragment, i.e. generally said, each pixel), so that repeated calculations should be left to the CPU. In addition, fragment and vertex programs are more complicated to implement, debug and deploy.

In our method, we use vertex and (optionally) fragment programs to do the spline interpolation and value-to-texture lookups. The basic rendering core looks like on Fig. 9-3:

```
For every grid cell do
    precompute spline coefficients
    for every pixel in cell do
        compute the spline value
        convert the value to colour
```

Figure 9-3: Scheme of basic rendering core

The important fact here is that the "for every pixel" loop executes entirely on the GPU.

We chose Catmull-Rom spline interpolation [Catmull74], because it is easily to use to do the interpolation on 2D Cartesian grid (creating a bicubic patch). The spline coefficients are computed only once for a whole grid cell, so this computation is done in the main program, on the CPU. These coefficients are sent to the vertex and/or the fragment program (as program local parameters), which maintain actual interpolation (in addition to the traditional viewing transformation computations, etc.).

The interpolation may be done with varying precision: We can evaluate the spline patch at every pixel (using the fragment program), or only at selected points (using the vertex program) and then interpolate between these points only linearly (see Fig. 9-2). This choice results in a trade-off between the visualization quality and speed. If we decide to interpolate only at selected number of points, we can also choose the number of interpolation points. As the visualization proceeds in real-time, we may let the user choose the visualization parameters at run-time and change them dynamically (when rendering a single grid, the choice is fixed, i.e. we do not use any adaptive subdivision [Weimer99] or similar techniques.) According to our experience, the per-pixel interpolation is slightly slower, but the difference in quality is almost negligible, when the number of subdivision points is high enough.

9.5 Converting to colors with possible spline interpolation

After the pixel value is determined using the spline interpolation, we convert it to colour. For that purpose, we use a pre-calculated texture containing colours corresponding to the range of data values. By properly choosing the texture, we very simply achieve rendering of isolines (see Fig. 9-8). The basic idea is to convert chosen values to the isoline colour (e.g. black) instead of the colour normally corresponding to the value. This way, the isoline appears in the picture at each point where the interpolation yields the respective value. Because of the spline interpolation, the isolines are smooth. The advantage of the use of textures is also that texture mapping is very fast on current graphics hardware. We can also dynamically change the texture and thereby choose palette, isoline equidistance and other parameters in the way that suits the visualized data at the exact moment. The drawback of the isoline drawing method is that the isolines may be blurry in situations when gradient is small, and, in extreme (the whole data grid equal to the isoline value), the whole picture could get black (as one big "isoline").

9.6 Implementation

Petr Kadlec fully implemented the presented concepts as a library in the standard ANSI C++ language, using the OpenGL API. Because of this choice, the implementation may be used on a wide range of computer systems, as the OpenGL is a broadly supported industry standard. He used GLUT as the windowing interface, because it is also supported on a majority of platforms. These choices

ensure that our implementation is easily portable to other systems.

Because the OpenGL standard has been specified some time ago, most of improvements in current graphics hardware are supported through use of the OpenGL extensions [OGL-ExReg-WWW]. In the time of implementation, OpenGL version 2.0 (which contain many of these extensions as a part of the basic API) was expected in near time horizon. Even now is currently not widely supported. Therefore, he decided to use various OpenGL extensions for interfacing the new technologies (such as vertex and fragment programs). The library uses extensions for vertex and fragment programs, vertex arrays, and point sprites. Support for the standardized ARB extensions (ARB_vertex_program, ARB_fragment_program, etc.) and for nVidia versions (NV_vertex_program, NV_fragment_program, etc.) of these extensions is available. Implementation is capable of choosing at run-time those extensions that are supported by the host system. The implementation is easily extensible to support any other extensions that have similar interface. Currently, we are also adding OpenGL 2.0 Shading Language to add possible support for future graphics adapters.

9.7 Hardware Support

The extensions mentioned are today supported on many low-cost graphics accelerators. The vertex programs are supported (in hardware) on most of the common graphics hardware sold today (e.g. even an old nVidia GeForce2MX has hardware support for vertex programs).

The hardware support for fragment program has been not so wide in the past, but recently, new models of graphics cards have begun to support the fragment shaders also, even in the current market low-end models (e.g. nVidia GeForce FX5200 and ATI Radeon 9200).

In theory, it is possible to run the program on a system without hardware support for the technologies used. Nevertheless, as the method is designed for hardware support, it would make no sense, because all advantages of the method would be gained at the cost of the emulation overhead.

9.8 Application and tests

At the Czech Technical University at Prague, we are developing an application for real-time simulation and visualization of combustion processes. The system was described in the text above. Until now, we were unable to obtain real-time, high quality visualization of about 40 various computed volume characteristics (such as temperatures, velocities and mass fluxes). For such visualization, we were using a simple approach of linear colour interpolation of quads, supported and rendered fast by current modern graphics accelerators.

By incorporating the spline-interpolation methodology to our system, we could not only prove the idea, but also to verify the above-described methodology. The method has substantially improve the quality of graphics output compared to the original linear interpolation method. We also tested the

performance of different visualization methods on various platforms with different overall and graphics performance.

9.9 Picture quality enhancements

The difference between the common linear interpolation method and the spline interpolation method is typically visible on every picture, even at the first sight (see Fig. 2 and Fig. 3). In every picture, the visualized areas come smoother, edginess of the margins between areas of passage of major value differences are avoided and the picture is more realistic and attractive to viewer.

Significant differences are being visible on places with large values differences, such as on Fig. 9-5. Probably the most significant enhancements are being visible when visualizing details of an area inside the boiler using higher zoom levels (see Fig. 9-7). In addition, a dramatic gain in visual quality is achieved in cases, where (due to demand of high simulation speed) low-resolution grids – e.g. 20 to 40 cells – are used (see Fig. 9-6). The overall visual quality enhancement in all cases is significant.

9.10 Performance discussion

The considerable advantage of our method is that it can be used on almost every today's common graphics accelerator (which supports vertex shaders). Thus, the high quality visualization output can be achieved even on the old and very cheap GeForce2 MX family of cards. On today's new graphics accelerators, with improved vertex and pixel shaders, we are able to receive real-time performance when visualizing cells grid up to 10000 cells. This is sufficient for the purpose of our 2D simulation system – we are not able to receive interactive framerates with simulation with more than 5000 grid cells

The following Fig. 9-4 summarizes the overall performance measured in drawing frames per second (FPS) when visualizing the test grid consisting of 1000, 3000 and 10000 cells using the above-described methods. All computer systems supported vertex shaders, but not all supported fragment shaders. On each system, three types of tests were performed: visualization using high-performance OpenGL linear interpolation (we have very significantly improved and reimplemented these codes over those used in [Kadlec04]), bicubic spline interpolation using vertex shaders and bicubic spline interpolation using fragment shaders. No simulation or other visualization was running during the measurements.

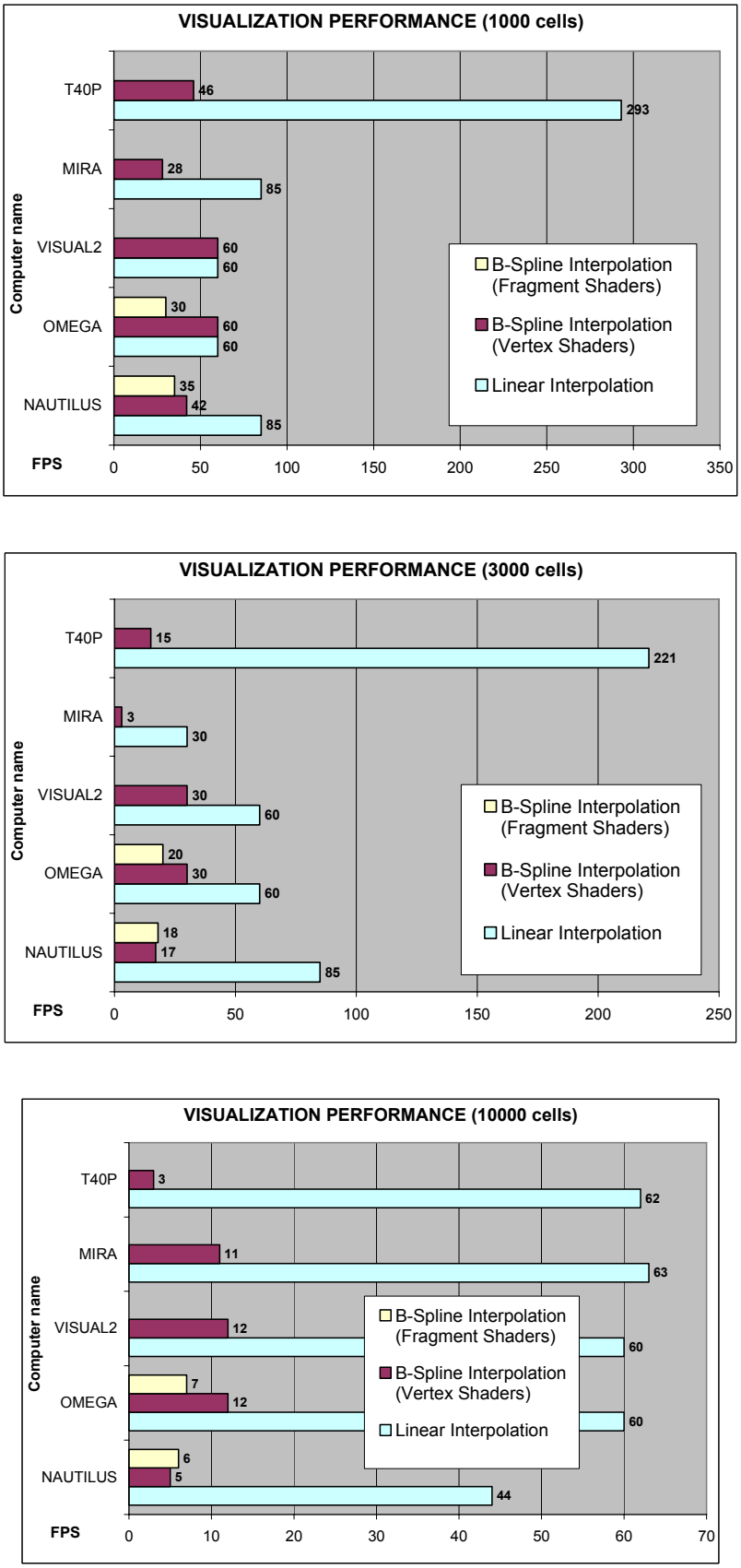


Figure 9-4: Visualization performance of interpolation methods with 1000, 3000 and 10000 cells

The test has been performed on the following systems:

- OMEGA – Pentium 4 3000 MHz, 1 GB RAM, GeForceFX 5500, Western Digital 2000 JD (Serial ATA), screen resolution 1280x1024x24bpp@60Mhz
- NAUTILUS – Pentium III, 1000 MHz, 512 MB RAM, GeForceFX 5600, Western Digital 400BB (Ultra ATA), screen resolution 1152x864x24bpp@85Mhz
- VISUAL2 – Athlon XP 2000+ (1733Mhz), 512 MB RAM, GeForce 4 TI 4200, Seagate Barracuda ATA IV, screen resolution 1280x1024x24bpp@60Mhz
- MIRA2 - AMD Athlon 900 MHz, 256 MB RAM, Geforce2 Ultra, screen resolution 1600x1200x24bpp@85Mhz
- T40P – Intel Centrino 1.6 Mhz, 512 MB RAM, ATI Mobility Fire GL, screen resolution 1400x1050@60Mhz

Except of T40P (due to ATI chipset), the tests did not measured visualization speed in frames per second higher than the display frequency used (85 and 60). In such cases, 85 and 60 means "at least 85 or 60". However, the tests revealed, that spline based methods are 6 – 20 times slower than the interpolation method. On the other hand, the tests were performed, when no simulation where running. The more demanding simulation is running, the less considerable will be this performance difference. For grids up to 10000 cells, it is still suitable for nearly real-time visualization on commodity, low priced graphics boards (better results would be certainly gained on contemporary graphics chips nVidia GeForce 6800 and ATI X800). Our spline interpolation method gives results that are comparable to high-quality visual outputs of common systems, such as the well-known CFD package FLUENT [FLUENTInc-WWW]. In contrast to this and other similarly based systems, our system is able to offer scalable performance to visualize tens of data frames per second with hardware-accelerated spline-interpolation.

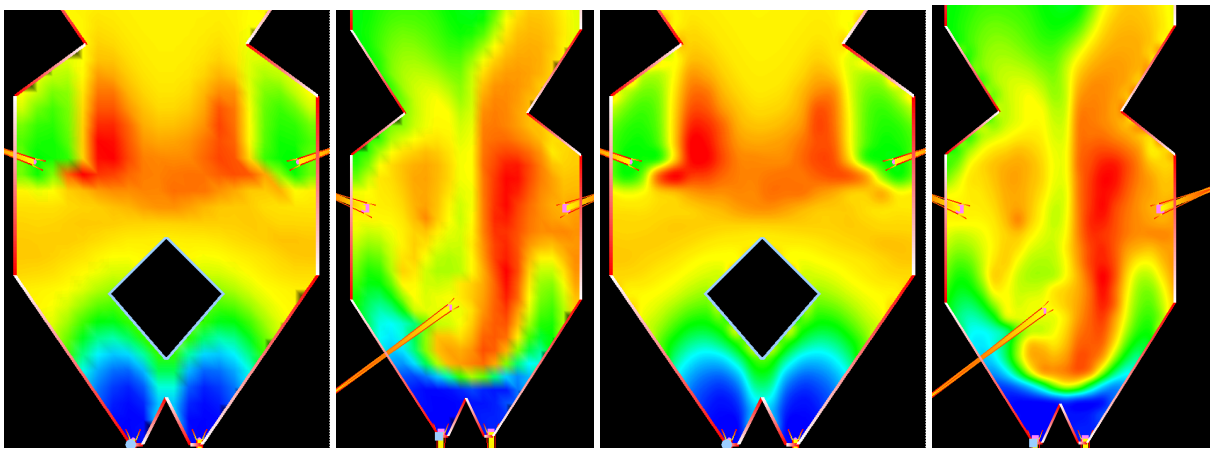


Figure 9-5: Left: The original picture of the boiler area temperatures, generated by linear interpolation of OpenGL quads. Right: The visual enhancement is typically visible on every picture of the visualization of cell characteristics with any zoom level.

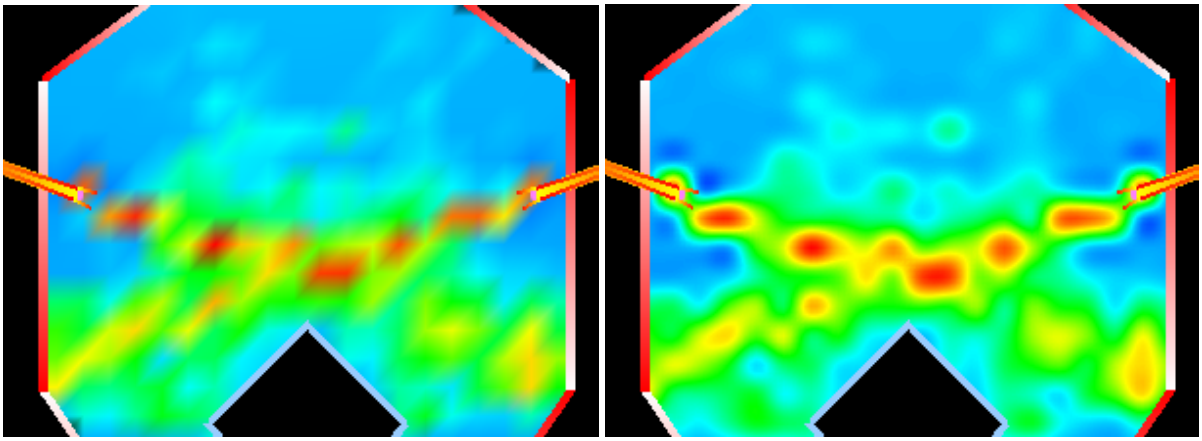


Figure 9-6: Original visualization of combustible masses inside the boiler area. Right: Visualization of the mass using the spline interpolation method

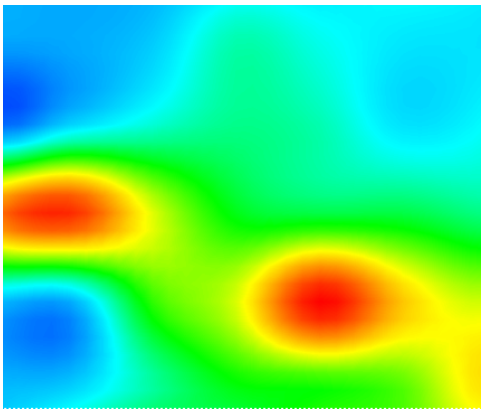


Figure 9-7: The original visualization output suffered considerably in picture quality when using high zoom levels. Even with high zoom levels, the spline interpolation method gives acceptable, realistic and attractive visual output

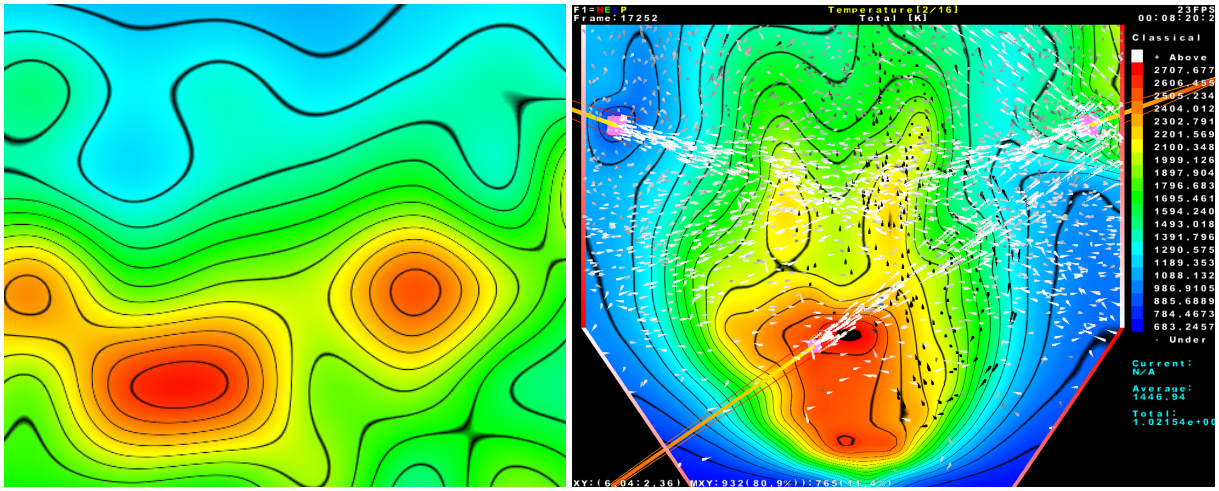


Figure 9-8: Left: Using the pre-calculated texture palette concept, we can easily visualize isolines, with no additional performance cost over the basic spline interpolation. Right: Visualization of isolines together with particle system

9.11 Vertex shaders versus pixel shaders

We have implemented the spline interpolation with both vertex shaders (computing the spline value only at discrete points) and pixel shaders (computing the spline value at every pixel). After testing of visualizations of various characteristics and areas of the boiler, we have realized that the gain of the visual quality when using per-pixel interpolation in comparison with vertex-based interpolation is very low, while requesting significant performance. Thus, we recommend using vertex shader interpolation instead of pixel shaders as the default technique for accelerated spline interpolation.

10 Interactive model of combustion for education

In the previous parts, we created various concepts leading to real-time, interactive visualization of combustion. We have them implemented and tested in an educational application for combustion processes. We describe this part of our work in this chapter.

10.1 Background and motivation

In recent years, modern high performance desktop computers and workstations have made a revolution to the power engineering industry and computer graphics area. Currently, there is a vast amount of research projects, applications and commercial products, which are able to simulate and visualize many natural production and technological processes. They allow the designer to experiment extensively with the model of the boiler designed without the necessity to build physically the boiler itself.

Although today's commodity PC's have multiplied their performance, there are still many tasks, for which the current computing power is insufficient. The simulation of various fluid flow related tasks and combustion processes is a typical example. For these tasks, we must use large simplifications in the description of corresponding physical descriptions and equations. Nevertheless, even with these simplifications, modelling and solving complex tasks such as combustion processes in today's packages and commodity systems can take hours or more. This is the price for reaching an acceptable level of precision of computation needed for professional industry design. A good example of the above is modelling of the combustion in boilers using the well-known and widely used FLUENT package [FLUENTInc-WWW]. General disadvantage of this approach, especially in education, is the complexity of simulation, which results in very time-consuming calculations. Another certain drawback of this and the similar systems is simply the fact that such systems are not easy to learn and to work with for the beginners.

We are in a different situation when we can dispense from the reliability and high precision required for industry and production applications. The reason for such compromise can be the desire for the real-time, interactive combustion simulation available on today's commodity PC's and workstations. This can be especially useful when designing tools for engineering education and training.

10.2 Interactive visualization and education

Nowadays, common availability of high performance PC's, easy to use operating systems, high quality projectors and slightly increasing general computer knowledge, leads to choosing more effective, synoptical and visual forms of education. Such forms of education can use illustrative schemes and pictures, or using common animation formats (like MPEG and AVI files).

One of the most interesting and favourite forms of education is the education with interactive features. It is especially suited for complex, practical tasks, when long, theoretical explanation would be ineffective or could even lead to confusion of students.

An interactive form of education (if used in a proper and meaningful way) offers the "doing by learning" and "what if" features. It can fulfil dynamic requirements of the teacher and students. If it is used in an individual learning and practicing part of education, it can motivate the students to use their creativity and can easily answer some of their questions. Furthermore, it can motivate them in the learning process, which now becomes more interesting and is more "fun".

Usual problems of the seminar courses and lessons efficiency are parts with individual works of each of the students. An interactive educational system for coal combustion modelling is one of the options for more efficient education. A challenging part of education in the field of combustion processes is the explanation of dynamic behaviour of burning coal particles. An important task is to introduce and describe particle traces and their changes (with respective heat release). In addition, it is important to demonstrate clearly the concrete power output (specified volume load) and its changes, heat transfers into the combustion chamber walls and temperature field into furnaces.

All these requirements can be met when using our application My Pulverized Coal Combustion, which is based on a simple Fluid Simulator and Virtual Coal Particle System (described in Chapter 5). The application will be described in the following text. The model respects a specific fuel system, its specific features and the necessity to respect inlets of fuel and combustion air. Our education system uses the industry standard OpenGL platform for reliable and fast visualization. This means that our system could be used on a standard graphics accelerator.

The values of particle and volume characteristics are visualized using colour attribute of drawn graphics primitives.

10.3 Characteristics in grid cells

The selected local characteristics in the grid cell, such as the total temperature, mass values of the combustibles and the air, local wattages, and heat fluxes, heat radiations, pressures, burned mass, released heat, oxygen concentrations and many others (total about 50) can be visualized. We use OpenGL linear interpolated quads with the support of the graphics hardware acceleration for visualization of the cell characteristics or hardware spline interpolation of data grid (see Fig. 10-2).

This concept allows the real-time visualization, which gives results similar to widely used isosurfaces technique. We can also use the flow visualization using the Image Based Flow Visualization (IBFV) (see Fig. 10-1), technique by Wijk [Wijk02].

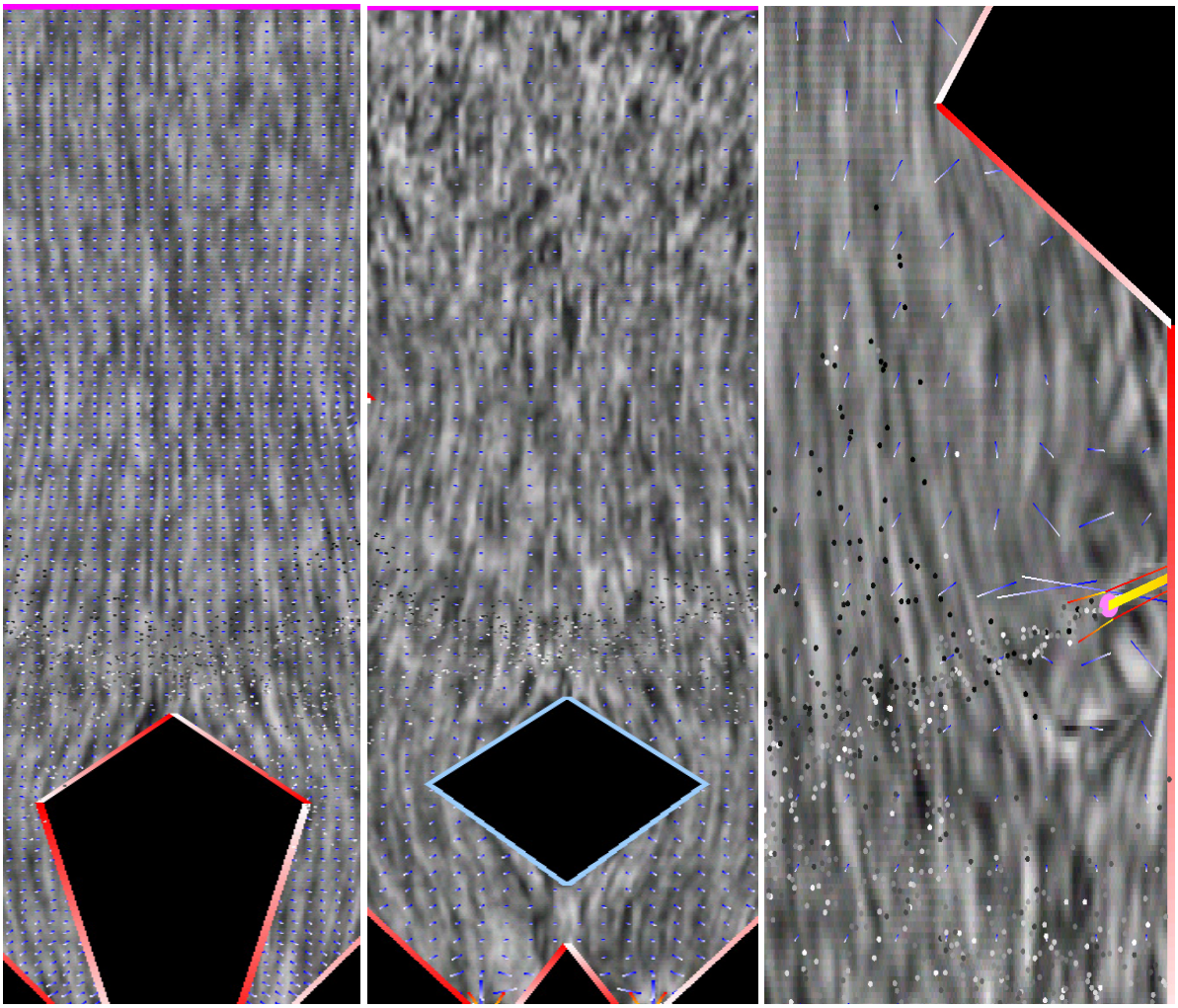


Figure 10-1: Visualization of flow in the boiler using IBFV. Left: Flow in boiler with grid 50 x 100 cells, Middle: Flow in boiler with grid 20 x 40, Right: Detail of flow with grid 20 x 40

10.4 Particle characteristics

The coal particles are visualized using standard OpenGL function of drawing pixels (see Fig. 10-2 left). We utilize built-in hardware support for visualization of smoothed pixels and alpha blending, available and supported in today's common graphics accelerators. With these settings, display drivers on ATI and nVidia display boards visualize particles as circles. Thus, even at higher zoom levels and/or used particle sizes we obtain visually acceptable representation of the particles. Another method we use is visualization with point sprites, available as OpenGL extensions of almost all new graphics accelerators. This method allows drawing custom sprite (graphics bitmap) instead of classic points (see Fig. 10-3).

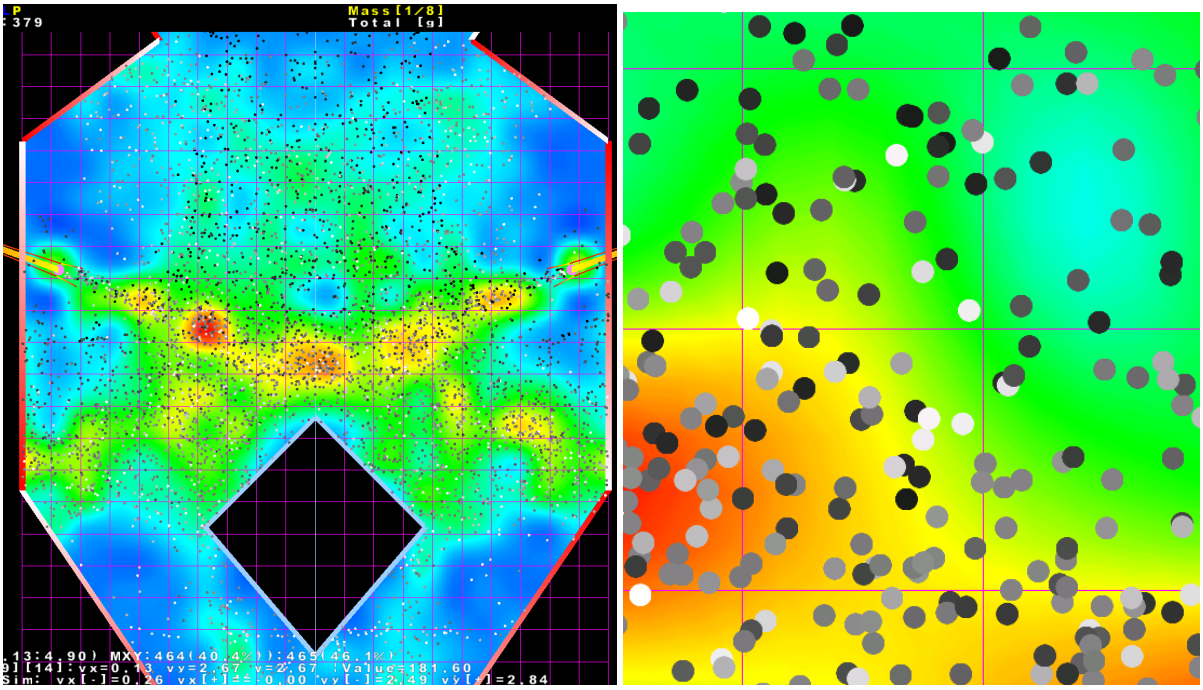


Figure 10-2: Visualization of thousands of virtual coal particles characteristics together with selected cell grid characteristic (drawn on the background). Right: Visualizing particles using full hardware accelerated, combined smoothing and blending of pixels. It gives acceptable visual quality even with a high zoom level.

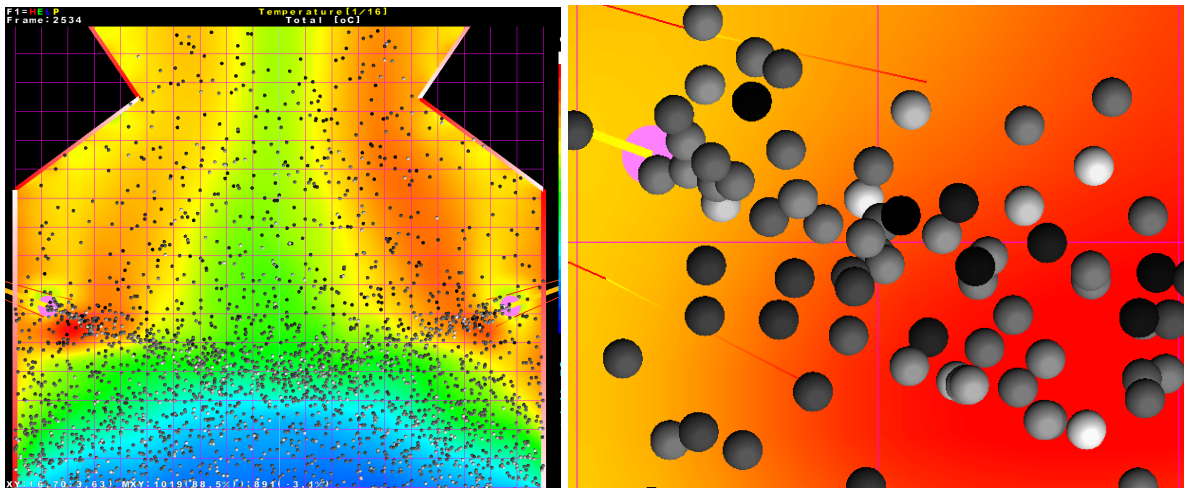


Figure 10-3: Visualization of virtual coal particles using point sprites, resulting in better picture quality

This way, we may visualize particle diameters, mass of particles, the time and distance particles spend in boiler, the distance it arrived inside the boiler chamber, combustible part of the particle and more (total about 10).

Utilizing the advantage of the particle system concept, we can easily construct the particle traces. We produce this effect by keeping the previous particle positions and characteristics in main memory. After storing these positions, we can draw the particles in current time step together with

these we have kept. The particle traces can clearly indicate the velocities and direction of motion of coal particles, which are visible even on the static state picture, see Fig. 10-4.

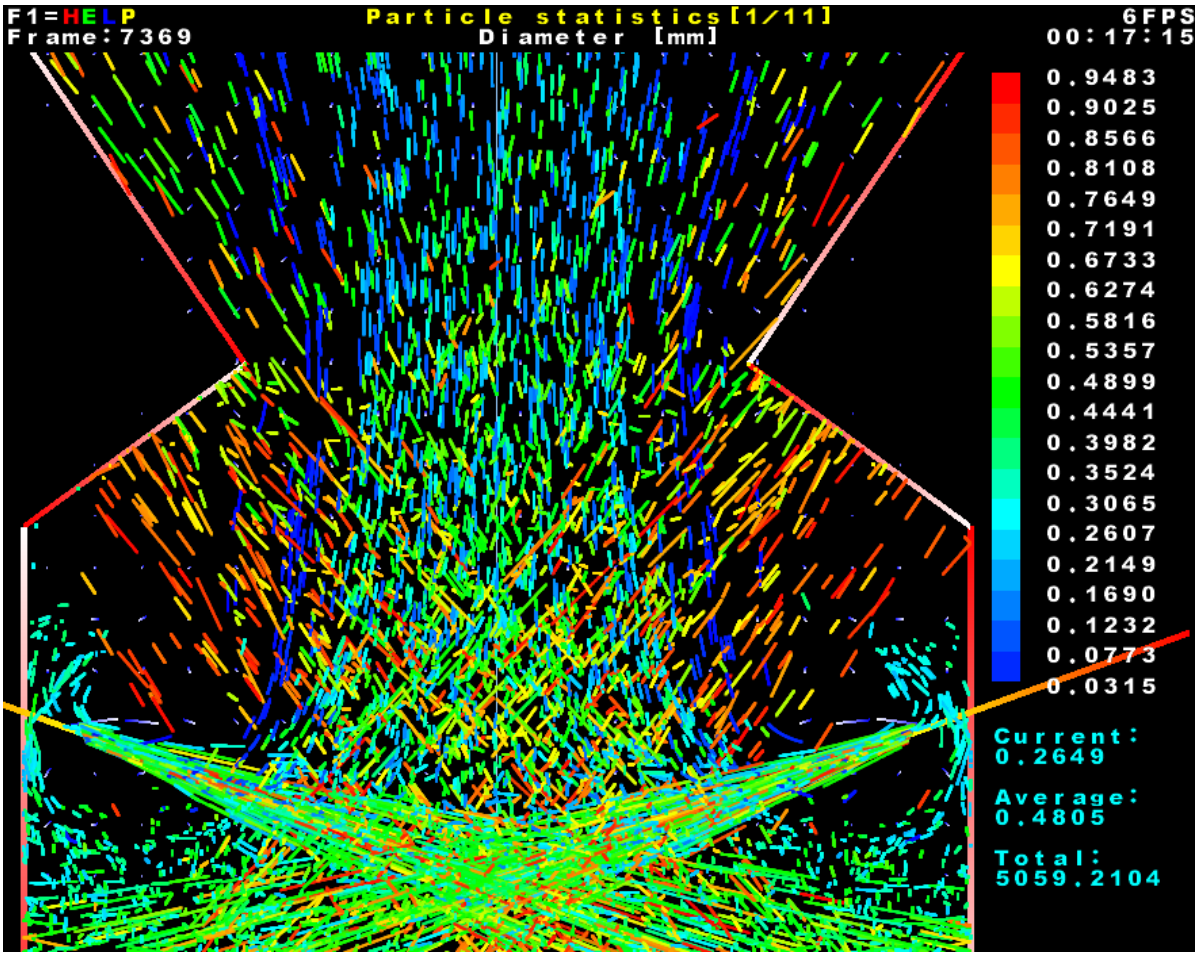


Figure 10-4: Real-time visualization of partial particle traces helps in determining particle speed, direction and dynamics even in static images. However, the visually best overview of dynamics is gained in real-time mode of our system.

10.5 Particle and volume statistics

Another way of presenting the computed values is utilizing statistics feature offered by our system. The inputs for statistics are either values of any selected cell grids characteristics or values of any described characteristics of the particles. We can measure and visualize the values distribution in the grid cells and particles for all the above-described characteristics. The sample visualization output is shown on Fig. 10-5.

10.6 Interactive simulation

During the simulation, we can interactively modify any of about 45 simulation parameters. On the fly, we can for example increase the number of generated particles (improving the simulation

precision at certain cost of visualization and simulation speed, see the Fig. 10-6) or change the mass of average coal particles flowing from the inlets of the boiler.

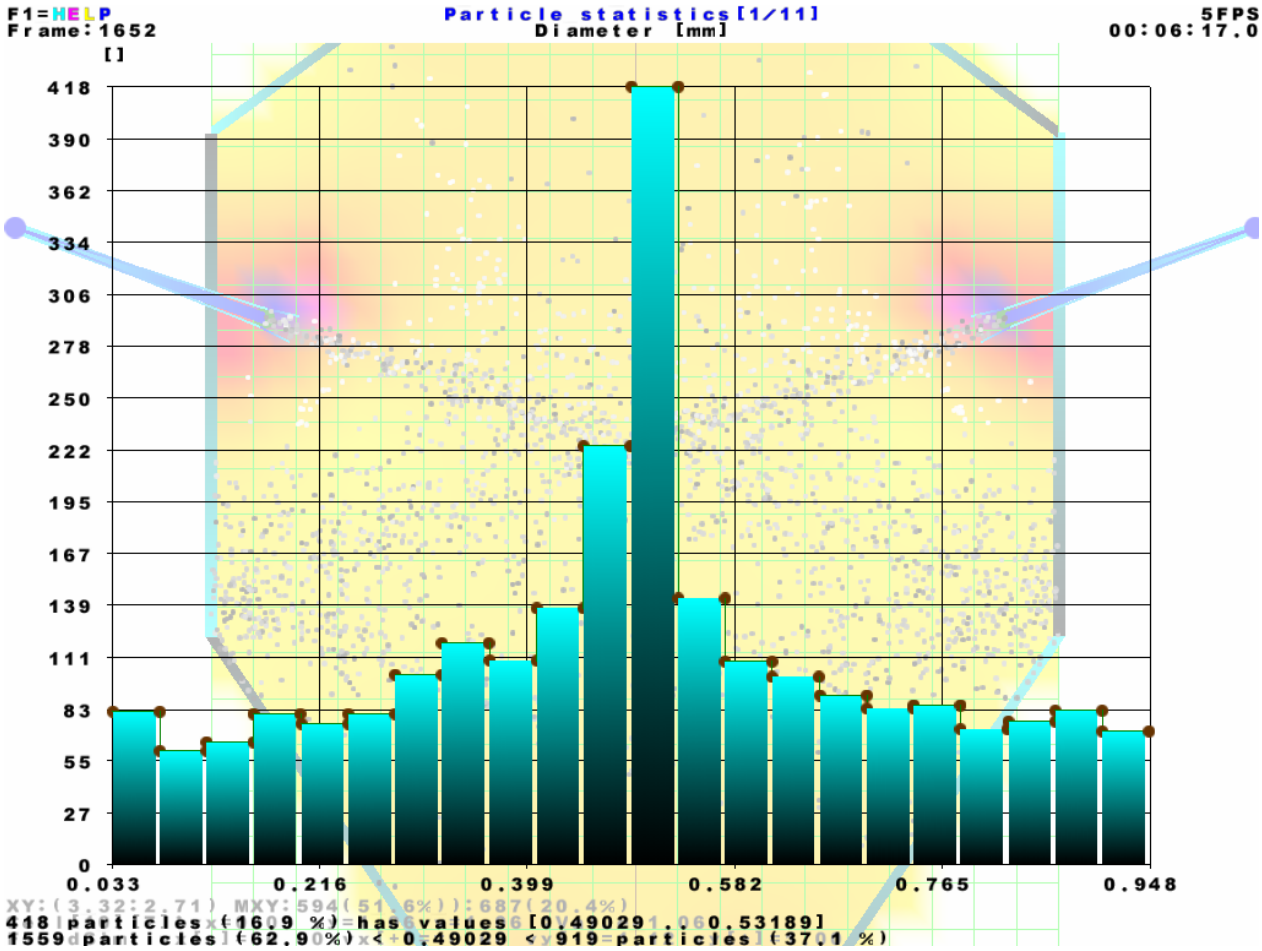


Figure 10-5: Sample statistics of coal particle diameters distribution inside the boiler chamber

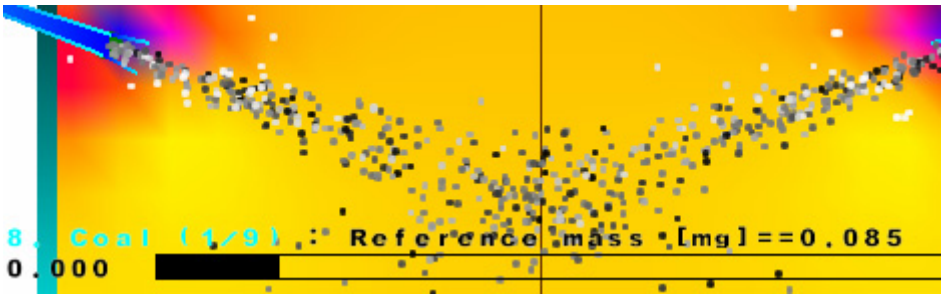


Figure 10-6: The count reference mass of virtual coal particles is being modified and immediately applied to the simulation computation on the fly

Moreover, we can interactively change the parameters of the inlets – completely change velocities, position, direction, and angle of spread, air and coal masses. Further we can change diameters and coefficients of air and coal flowing to the boiler chamber. The interactive modification of boiler inlet is shown on the Fig. 10-7.

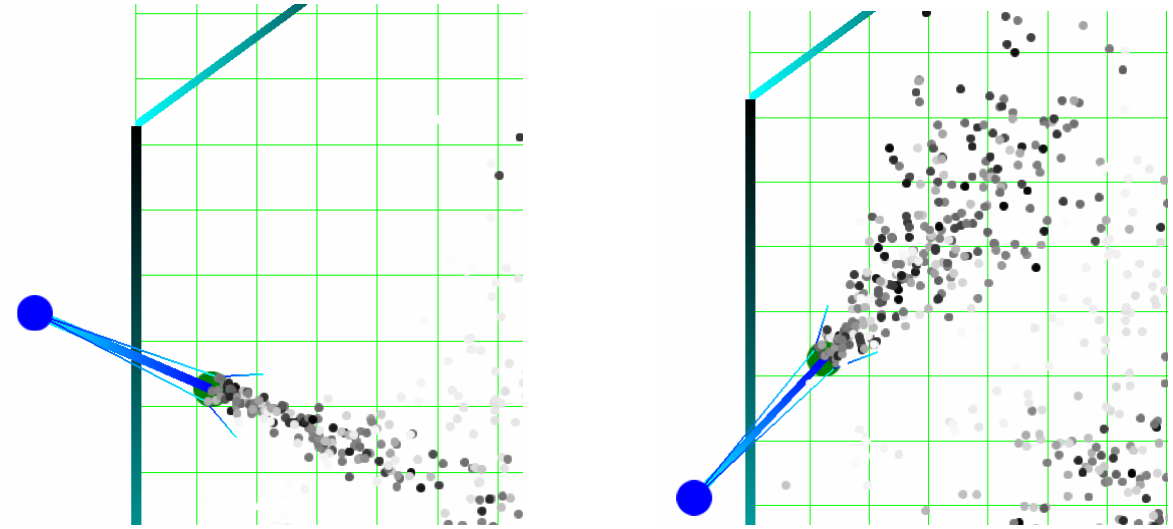


Figure 10-7: Changing interactively coal inlet parameters. The state of the inlet and particles before change is on the left side. The changed state after next 5 seconds of interactive simulation is shown on the right side.

Further, we can select any one of the particles (representing corresponding mass of the coal under investigation), change its parameters on the fly, including position and then watch how it behaves after applying such changes, see Fig. 10-8.

10.7 Interactive visualization of results

We can select any combination of the particle characteristics, grid cell characteristics, and statistics. We can change visualization parameters, e.g. size of the particles, select area in the boiler under investigation, change visualization method of drawing pixels and set the zoom level. Furthermore, we can adjust colour the palettes and the colour ranges for the visualization, setup the brightness or invert the screen colours (choosing colour scheme of light background and dark particles and grid cells) suitable for printing and several others adjustments.

10.8 Integrated support for FSS and UDS Trees

Our system supports hierarchical structures and its advantages described in previous chapters 7 and 8. For example, teachers could create prepared solution, while keeping interactive possibilities, so that students could explore and extend them, and they have possibility to run combustion simulation with more precise computation (time steps, computational grid). Even, that the fluid simulator on current hardware offers sufficient performance for real-time exploration of combustion inside pulverized coal boiler, when using low resolution computational grid (e.g. 25 x 50), this techniques would be essential, if we would require such larger computational grid or we would be to implement even more precise fluid solvers.

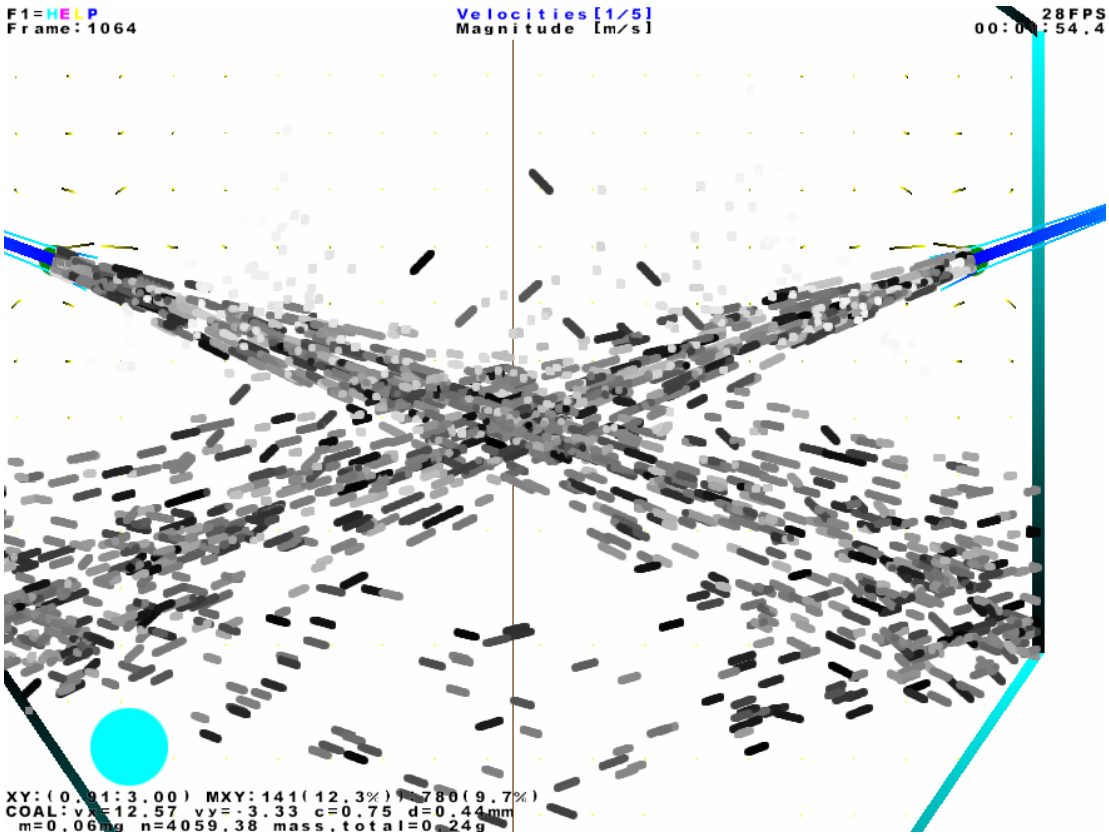


Figure 10-8: Tracking and monitoring the characteristics of the selected particle (highlighted as the disc) inside the boiler chamber. We can lock the viewing area to its position and watch the coal particle trajectories as it flows through the boiler with any zoom level.

10.9 Feedback from students

The response of students to interactive education is very positive. This kind of education allows the students to perform own experiments individually in a practical way allowing to answer questions emerging from theoretical foundations of theory of combustion. The concept of system allows an individual experience based on concrete choice of correct and wrong parameters (like size and other properties of particles, ratio between amount fuel and air, velocities of the air flow, positions and directions of inlets etc.) their influence on the combustion process and watching corresponding dynamic behaviour. This possibility to gain an individual experience based on experiments is valued very much by the CTU students. This fact is especially important, because the course for which this system has been used belongs to very basic courses in the power-engineering study track. It is a transition from the traditional way of education based on the mediation of knowledge (from professors to students) to the education based on gaining an individual experience. In such way the students can master new knowledge in a more interesting way and much more deeply than in the traditional approach.

11 Summary and contribution

11.1 Fast fluid simulation combined with virtual coal particles

In the beginning of our effort, we used a modelling based on the Isotherm-Free concept, described in Chapter 4. However, this approach had considerable limitations and was suitable only very simple boiler configurations. We had faced serious problems when using complex setup and changing the boiler configuration during the simulation.

As described in Chapter 5, our coal combustion system is currently based on a fast fluid simulator core. The fluid simulator allows real-time computation of the airflow. It utilizes Euler Equation and The Continuity Equation. The simulator is very easy to implement and thus can be reusable in various computer graphics and simulation tasks related to the airflow simulation and visualization.

The high speed of the fluid simulator and combustion system allows real-time visualization of the results (using the OpenGL graphics interface). The system has been implemented in a 2D structured cell grid (with variable depth – z-axis), and considering the methodology used, it could be relatively easily to extended to 3D grid cell, namely due to easily adaptation of computational model, however the implementation would be demanding in terms of development time and computation requirements. The unique concept of the virtual coal particles, based on the interaction of the air mass inside the cell with the coal mass, allows us to both speed up the combustion simulation and also gives us a possibility of easy tracking the flow of the combustibles. Virtual coal particles also allow visualizing dynamics of the combustion process.

We have tested our system by modelling a boiler of real dimensions, characteristics and parameters. The behaviour and quantitative features predicted by our system were comparable with those of a real boiler as well as with the results gained from the professional CFD software package FLUENT 5.5. However, we had to use manual tuning, that are dependent on concrete solved tasks, to approach to these results.

On the other hand, our fluid simulator is conditionally stable, thus it needs to setup fix time step dependent on grid size (increasing with grids that are more precise) as similar existing methods. In addition, when requesting precise results, the professional CFD packages and software, or even possibly some other fluid simulators and solvers based on more precise mathematical and physical model could give better results, but at the cost of the computation speed. Our fluid simulator is made as a component, which could be possibly replaced by such another fluid simulators or solvers.

For cases, where more precise computations are used (by e.g. increasing size of computational grid in our fluid simulator), resulting in computational demands, which would disallow real-time visualization, we proposed further described approaches based on interpretation of generated data stored on disk drives. Our fluid simulator was used as source of such data in those cases.

11.2 Fluid Simulator States (FSS)

As described in Chapter 6, the simulation with pre-calculated Fluid Simulator States extension is compound of partial computation with synchronous utilization of pre-calculated Fluid Simulator States stored on disk device. This concept can drastically improve the simulation and subsequent visualization speed of wide spectrum of computer graphics applications based on fluid simulator while keeping the precision of computation unchanged. Pre-calculating states of the fluid simulator rather than storing complete unsteady data sets of simulation results in much less disk space requirements, with virtually unchanged acceleration. In visualization part, all interactive actions and features such as changing the visualization parameters and visualization of arbitrary characteristics are kept. Even simple and not performance-optimized applications based on 2D or 3D fluid simulators (even non-real-time) can benefit from our concept. In other words, pre-calculated fluid simulators extension can help to overcome performance bottleneck of time-consuming fluid simulator codes, namely when using high-resolution grids or more precise, complex computation methods. Our concept could be utilized by either replacing our simulator in our system, with another, more precise one, or adapting and incorporating described ideas to already designed system.

We have performed tests of the architecture on our coal combustion simulation and visualization system based on fluid simulator and particle system. We have demonstrated that with this concept considerable acceleration of simulation and subsequential visualization can be gained, while requesting only small fraction of the disk space requirements. Such space would be needed for storing whole frames either as movie files (with total loss of interactivity) or complete unsteady data sets with much higher disk demands, while keeping the acceleration virtually either unchanged or better (in case of fluid simulator more demanding then the rest codes, which was the case of our application).

The limitation of this method is namely in loosing the interactive possibilities, namely changing the simulated configuration and boundary conditions during replaying the stored data and impossibility to change the playback speed with the possibility to skip selected frames. We address these issues in data generators based on tree structures. We are also somehow limited by the disk drive storage requirements, but the demands are considerably less, then it would be needed for storing corresponding Unsteady Data Sets.

11.3 Fluid Simulator States Tree

As described in Chapter 7, we have designed and implemented hierarchical tree structures built from pre-calculated Fluid Simulator States, which allow incremental, progressive and easy construction of various configurations of the boiler with high speed, interactive visualization and the playback of results. The modifications of simulation boundary conditions are available in every node of the FSS tree. Thus, every interactively selected path in the FSS tree corresponds to one modified simulation solution.

The original concept of a pre-calculated Fluid Simulator States Tree can be easily utilized in various applications based on fluid simulators and solvers as well. However, the impossibility to change the playback of interpreted data or store only selected simulation frames remains. We address these drawbacks in UDS Tree.

11.4 Unsteady Datasets Tree

As described in Chapter 8, we have proposed, implemented and tested the concept of hierarchical datasets tree structures in combustion processes simulation and visualization powered by the unstable fluid simulator. Choosing unsteady dataset helps us to avoid the common drawback of slow simulation of unstable fluid simulators, due to small timesteps.

Our concept overcomes the common drawback of the commonly used unsteady datasets, which results in loosing of interactivity on the simulation side. Proposed solution of hierarchical structures consisting of simulation results datasets over classical datasets allow incremental, progressive and easy construction and playback of various simulation configurations with interactive visualization of the results. The modifications of simulation boundary conditions are available in every node of the dataset tree. Every interactively selected path in the datasets tree corresponds to one modified simulation solution. We have proved that we can even use up to hundred thousand of particles without much worry about performance bottlenecks of the current commodity disk drives.

Our results can be easily utilized in general simulation and modelling applications, which use unsteady datasets for storage of the results. The users of simulation and visualization applications can extend the prepared and pre-calculated simulation results with their own modifications with possibility to use the results of previous solutions stored in tree while keeping a high speed of replaying, because of the utilization of unsteady datasets.

This method is limited by the speed of the disk drive, which should be enough to allow transfer large amounts of data into system memory, otherwise the acceleration could be considerably lower. However, we have demonstrated that on properly configured common Ultra ATA and Serial ATA drives, there is no problem with acceleration performance. In cases, that disk drive capacity storage is limited, we can still use the FSS and FSS Tree methods described earlier.

11.5 Hardware accelerated interpolation techniques

After having proposed methods for fast data generation based on fast simulation, and accelerated hierarchical storage, a question remains - how to visualize these data in optimal performance and quality. As we described in Chapter 9, our method achieves real-time rendering of grid-structured data using pre-calculated texture. We interpolate the data using spline interpolation that runs directly on the graphics accelerator, therefore leaving the CPU to compute the simulation in parallel. The texture used may be changed dynamically, which results in a possibility of interactive adjustments of visualization parameters. The method allows us to display isolines of the displayed data with no additional performance requirements. The isoline displaying may be enabled or disabled by simply choosing the texture.

We used implementation of this technique to improve quality of visualization in our coal combustion and visualization system. The success of the implementation proves the significant contribution of this hardware-accelerated technique for maintaining real-time, high-quality visualization of the cell characteristics. This concept can be used in similar applications regarding scientific, isocontour based visualization of computed or simulated data. Although, the new method is currently about 10 times slower than common raw OpenGL visualization using linear interpolated quads, it is still suitable for real-time visualization, with dramatic enhancement of the visual quality. In applications, which need to visualize 2D grids with up to 10.000 cells we can expect real-time performance on today's commodity graphics hardware. With more precise grids, we can still use mentioned simple OpenGL visualization.

11.6 Educational System of Pulverized Coal Combustion

As we described, in Chapter 10, our educational system *My Pulverized Coal Combustion* for pulverized coal combustion is based on a simple fluid simulator and virtual coal particle system. The fluid simulator allows real-time computation of the air flowing inside the boiler. We selected particle systems for maintaining visualization of combustion process dynamics. By their nature, they can even be utilized in the simulation and computation part.

The high speed of the fluid simulator and combustion powered by the virtual coal particle system and simplified combustion engine allows real-time visualization of the resulting characteristics and dynamics using hardware accelerated contour visualization, Image Based Flow Visualization (IBFV) and particle system (using OpenGL graphics interface). The system has been implemented in 2D grid cell space, with variable depth – z-axis. The system is suitable for educational purposes, where clarity, real-time interactivity and universality of computation if important.

The most powerful and new feature of our system is the simulation and visualization interactivity, which is available during real-time computation of the combustion process, without

needing to stop or restart the system. Tens of input parameters, including coal inlets setup can be modified on the fly.

Our results make it possible to get a good preview of the dynamics of combustion processes in a boiler. Based on this concept, the students and eventually designers of boilers, making use of the fast, preview based design and approach, could now test many configurations and modifications of the pulverized coal boilers interactively. The system by itself can in an interactive, efficient and attractive form, give an overview of how powerplant boilers work, with an overview of fundamental boiler parameters. The system optionally supports above described hierarchical tree storage and playback based on the pre-calculated Fluid Simulator States Trees and Unsteady Data Sets trees.

The interactive modelling can bring the student basic knowledge and fundamentals of constructing performance and efficient boiler solutions and more. We can recommend it as an introductory application for combustion processes overview in power plants. With this technique, we can get a good preview of the dynamics of combustion processes in a boiler. The stable version of the system, resulting from the work during PhD studies, will be used in the educational process in the Faculty of Mechanical Engineering at the CTU Prague. Quality and simulation precision is sufficient for these purposes.

Currently, our system is implemented in Microsoft Visual C++ and runs at interactive frame rates even on a commodity PC equipped with only AMD Athlon 1333 MHz and nVidia GeForce2 MX based graphics card.

12 Future work

During work on this PhD thesis, a real-time pulverized coal combustion application has been created, based on techniques presented in described in the above part of the thesis. In addition, it was suited as a framework for development and testing the methods and concepts.

It is based on a simple fluid simulator, which allows real-time simulation of air flowing inside the boiler area. The computation of combustion process is maintained using a special particle system, which is by its nature suitable for both visualisation and simulation of the combustion processes inside the boilers. The application is suited also for education purposes and its parts and concepts are described in the upper sections of the text.

Due to limited time, lack of appropriate funding and lack of software team, during the PhD studies, we were not concerned on industry precision and production quality, but on fast performance and possibility to use it effectively in education. The limited resources mentioned also caused that the system was designed and implemented in 2D only (this was also caused by situation of commodity hardware in 2000 when author started PhD studies, when common hardware would not be able to such computationally expensive calculations). Now in 2005 (and after implementation of our 2D system), there comes the right time for converting this project to 3D. Thus, we feel further possibilities for further development and enhancement of our techniques are namely increasing of the data generator (computational model) precision and conversion from 2D to 3D. While increasing the precision would only improve/replace the fluid simulator described in Chapter 5, conversion to 3D would affect nearly all techniques presented here in both the generation of data and visualization part. With many other advantages, it would allow more intuitive and more precise way of the boiler simulation and visualization. On the other hand, we feel that conversion of all techniques described here would be very demanding and would need either considerable funding or new PhD candidate(s) to be successful. All of the possible enhancements are summarized in the following list:

Major tasks

- In general, create a more precise computational model or enhance and extend current equations
- Implement more accurate heat distribution between the particles, cells and walls.
- Further enhancements to the combustion engine and fluid simulator
- Re-implement the whole project to 3D
- In new 3D implementation, create 3D FSS and UDS Trees and perform measurements for

these cases.

- Create user interface, that would be more user-friendly

Minor tasks

- Add more visualization methods for particle systems (e.g. using lines and triangles)
- Implement additional interactive features of the boiler design and visualization
- Implementation of optimal existing compression and encoding methods for unsteady datasets files to decrease both the disk space and traffic requirements for complex tasks.

We hope, that under the condition of proper funding, a project based on these proposals would create very attractive and interesting solutions, which would made great contribution to modelling and visualization of combustion, including but not limited to education.

13 Conclusion

In this work, we have described effort during the PhD study and described possibilities useful for next possible research. We have described commonly used methods usable for the simulation and visualization of the fluids and combustion processes. We have presented existing visualization features and methods usable for the visualization of the general fluids and for combustion processes.

We have designed our fluid simulator and virtual coal particle concepts for both the simulation and visualization of the combustion processes. We use these components as data generators for storage techniques and visualization methods. For the simulation and visualization of a combustion system, our current investigation brings an interesting alternative to the classical CFD applications.

Major part of our work is concerned in using the speeding up the simulation using Fluid Simulator States. We extended this concept with tree structures allowing incremental and interactive concept of solution of fluid tasks including combustion. The hierarchical concept of computed data storage can be also well used for storing general unsteady datasets.

We had also worked on visualization of combustion processes and general fluids. Using the concept of virtual coal particles, we had created an attractive visualization allowing us to investigate the dynamics of the combustion process. We had used the features of the last generation graphics cards to propose and implement real-time and high quality visualization of flow fields by using hardware accelerated bicubic spline interpolation and IBFV.

The mentioned parts has been integrated and implemented into and tested in an application My Pulverized Coal Combustion that has been developed with demands of interactive enabled education being kept in mind. The system can be used for educational purposes in order to give students idea about the behaviour of boilers under various conditions. This application allows this with many interactive features. All of the presented techniques are independent and loosely bind together and thus parts of them, including concepts and ideas could be reused in other research projects or replaced by their more advanced versions.

We hope that some of the presented results would be used in either next research projects or practical projects trying to keep the environment we are living in clean. If we really still have to burn fossil fuels, at least let us try to do it effectively and ecologically.

14 Awards

The real-time pulverized coal combustion system and its portions formed by techniques presented at previous chapters of this thesis received the following awards:

- **2003, CTU FEE at Prague, Czech Republic**

Award of Dean of the Faculty of Electrical Engineering of the Czech Technical University in Prague for work Simulation and Visualization of Combustion Powered by Fluid Simulator, presented at conference CTU Poster 2003

- **2003, Brno University of Technology, Faculty of Electrical Engineering and Communication (BUT FEEC), Czech Republic**

Award of Dean for the best work in the International Competition of student creative projects Student EEICT 2003

- **2003, CTU in Prague, Czech Republic**

Price of the rector for placing between the best projects of PhD students, that were solved within scope of CTU internal grants and that were presented on conference CTU Workshop 2003.

15 References

[Abbot89] M. B. Abbott. Computational Fluid Dynamics: An Introduction for Engineers. Wiley, New York, 1989.

[Algor-WWW] Algor, Inc. Finite element analysis software. http://www.algor.com.

[Anderson95] J. Anderson. Computational Fluid Dynamics - The basics with the applications. McGraw Hill, 1995.

[Arques99] D. Arques, E. Felgines, S. Michelin, and K. Zampieri. Thermal convection in turbulent flow, image synthesi and heat animation. In V. Skala, editor, Proceedings of WSCG 1999, volume 2, pages 337-345. University of West Bohemia Press, Czech Republic, Plzen, 1999.

[Bauer02] D. Bauer, R. Peikert, M. Sato, and M. Sick. A case study in selective visualization of unsteady 3D flow. In Proceedings of the conference on Visualization '02, Boston, Massachusetts, pages 525-528. IEEE Computer Society, 2002.

[Becker95] B. G. Becker, N. L. Max, and D. A. Lane. Unsteady flow volumes. In G. Nielson and D. Silver, editors, Proceedings of the 6th IEEE Visualization Conference, Atlanta, Georgia, US, pages 329-333. IEEE Computer Society Press, 1995.

[Bordoloi02b] U. Bordoloi and H.-W. Shen. Hardware accelerated interactive vector field visualization: A level of detail approach. Computer Graphics Forum, 21(3):605-605, 2002.

[Bordoloi02a] U. Bordoloi and H. wei Shen. Hierarchical LIC for vector field visualization, Aug. 10 2002.

[Bruckschen01] R. Bruckschen, F. Kuester, B. Hamann, and K. I. Joy. Real-time out-of-core visualization of particle traces. In Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics, San Diego, California, pages 45-50. IEEE Press, 2001.

[Bryson96] S. Bryson and S. Johan. Time management, simultaneity and time-critical computation in interactive unsteady visualization environments. In Proceedings of the 7th conference on Visualization '96, San Francisco, California, United States, pages 255-ff. IEEE Computer Society Press, 1996.

[Bryson99] S. Bryson, D. Kenwright, M. Cox, D. Ellsworth, and R. Haimes. Visually exploring giga-

byte data sets in real time. Communications of the ACM, 42(8):82-90, 1999.

[C-Safe-WWW] Combustion research group of University of Utah. C-safe - center for the simulation of accidental fires and explosions. http://www.csafe.utah.edu/.

[Cabral93] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pages 263-270, 1993.

[Carlson02] M. Carlson, P. J. Mucha, I. R. Brooks Van Horn, and G. Turk. Melting and flowing. In Proceedings of the ACM SIGGRAPH symposium on Computer animation, San Antonio, Texas, pages 167-174, 2002.

[Carrea00] E. Carrea, S. Orsino, and J. Barsin. Full-scale trials and numerical modeling of a dry bottom ash extraction system from a 330mw coal fired boiler. In Proceedings of ASME IJPGC'00 International Joint Power Generation Conference Exposition, 2000.

[Catmull74] E. Catmull and R. Rom. A Class of Local Interpolating Splines. In R. E. Barnhill and R. F. Riesenfeld, editors, Computer Aided Geometric Design, pages 317-326. Academic Press, 1974.

[CFD2000-WWW] Adaptive Research company. CFD2000 product information. http://www.adaptive-research.com/cfd2000frm.htm.

[CFDAnalyzer-WWW] Amtec company. CFD analyzer version 2.0 . http://www.amtec.com/Product_pages/cfd_analyzer.html.

[CFDOnline-WWW] CFD Online. CFD resources online an online center for computational fluid dynamics. http://www.cfd-online.com/Resources/homes.html.

[Chen97] J. X. Chen, N. da Vittoria Lobo, C. E. Hughes, and J. M. Moshell. Real-time fluid simulation in a dynamic virtual environment. IEEE Computer Graphics and Applications, 17(3):52-61, 1997.

[Cihelka69] J. Cihelka. Vytápění a větrání. SNTL, Praha, 1969.

[Cox97] M. B. Cox and D. Ellsworth. Application-controlled demand paging for Out-of-Core visualization. In R. Yagel and H. Hagen, editors, IEEE Visualization \'97. IEEE, 235-244 1997.

[Cox99] M. Cox. Large data management for interactive visualization design. In Proceedings of the SIGGRAPH '99 System Designs for Visualizing Large-Scale Scientific Data course notes, pages 5-29. ACM Press, 1999.

[Crawfis93] R. Crawfis, N. Max, B. Beccker, and B. Cabral. Volume rendering of 3D scalar and vector fields at LLNL. In IEEE, editor, Proceedings, Supercomputing '93: Portland, Oregon, November 15-19, 1993, pages 570-576, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1993. IEEE Computer Society Press.

[Crawfis92] R. Crawfis and N. Max. Direct volume visualization of three-dimensional vector fields. In VVS '92: Proceedings of the 1992 workshop on Volume visualization, Boston, Massachusetts, United States, pages 55-60. ACM Press, 1992.

[Crawfis00] R. Crawfis, H.-W. Shen, and N. Max. Flow visualization techniques for CFD using volume rendering. In Proceedings of the 9th International Symposium on Flow Visualization, Edinburg, Scotland, UK, 2000.

[DeCoro02] C. DeCoro and R. Pajarola. XFastMesh: Fast view-dependent meshing from external memory. In R. Moorhead, M. Gross, and K. I. Joy, editors, Proceedings of the 13th IEEE Visualization 2002 Conference (VIS-02), pages 363-370, Piscataway, NJ, 27- 2002. IEEE Computer Society.

[Deutsch96] L. Deutsch. ZLIB compressed data format specification version 3, 1996.

[deLeeuw95] W. C. de Leeuw and J. J. V. Wijk. Enhanced spot noise for vector field visualization. In VIS '95: Proceedings of the 6th conference on Visualization '95, page 233. IEEE Computer Society, 1995.

[deLeeuw98] W. de Leeuw and R. van Liere. Comparing lic and spot noise. In Proceedings of the conference on Visualization '98, Research Triangle Park, North Carolina, United States, pages 359-365. IEEE Computer Society Press, 1998.

[Dobashi00] Y. Dobashi, K. Kaneda, Y. Yamashita, T. Okita, and T. Nishita. A simple, efficient method for realistic animation of clouds. In K. Akeley, editor, Siggraph 2000, Computer Graphics Proceedings, Annual Conference Series, pages 19-28. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[Dobashi99] Y. Dobashi, T. Nishita, and T. Okita. Animation of clouds using cellular automaton. In Computer Graphics and Imaging '99 (CGIM'99), pages 251-256, 1999.

[Dvorak96] R. Dvořák and K. Kozel. Matematické modelování v aerodynamice. Vydavatelstí ČVUT Praha, 1996.

[Eaton99] A. Eaton1, L. Smoot, S. Hill, and C. Eatough. Components, formulations, solutions, evaluation, and application of comprehensive combustion models. Progress in Energy and Combustion Sci-

ence, 25:387-436, 1999.

[Ellsworth00] D. Ellsworth and H. wei Shen. Accelerating time-varying hardware volume rendering using TSP trees and color-based error metrics, 22 2000.

[Enright02] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques, San Antonio, Texas, pages 736-744. ACM Press, 2002.

[Faltyn99] R. Faltýn. Simulation and visualization of combustion processes in vortex furnace. Master's thesis, CTU FEE Prague, 1999.

[Fedkiw01] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 15-22. ACM Press, 2001.

[Feldman03] B. E. Feldman, J. F. O'Brien, and A. Arikan. Animating suspended particle explosions. In J. Hodgins and J. C. Hart, editors, Proceedings of ACM SIGGRAPH, volume 22(3) of ACM Transactions on Graphics, pages 708-715, 2003.

[FluentInc-WWW] Fluent Inc. Flow modelling software and services. http://www.fluent.com/.

[FluentMan-WWW] Information server FSI CTU. Complete gambit, FLUENT and ANSYS documentation at fsid. http://www.fsid.cvut.cz/man.

[Forsell95] L. K. Forssell and S. D. Cohen. Using line integral convolution for flow visualization: Curvilinear grids, variable-speed animation, and unsteady flows. IEEE Transactions on Visualization and Computer Graphics, 1(2):133-141, 1995.

[Fortik02] M. Fortik. Visualization of combustion processes. Master's thesis, CTU Prague, 2002.

[Foster01] N. Foster and R. Fedkiw. Practical animation of liquids. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 23-30, 2001.

[Foster96] N. Foster and D. Metaxas. Realistic animation of liquids. Graphical Models and Image Processing, 58(5):471-483, 1996.

[Foster97] N. Foster and D. Metaxas. Modeling the motion of a hot, turbulent gas. Computer Graphics (SIGGRAPH 97 Conference Proceedings), 31(3A):181-188, 1997.

[Foster00] N. Foster and D. Metaxas. Modeling water for computer animation. Communications of the

ACM, 43(7):60-67, 2000.

[Fuhrmann98] A. L. Fuhrmann and E. Gröller. Real-time techniques for 3D flow visualization. In D. Ebert, H. Hagen, and H. Rushmeier, editors, IEEE Visualization '98, pages 305-312. IEEE, 1998.

[Gadelhak98] M. G. el Hak. Fluid mechanics from the beginning to the third millennium. International Journal of Engineering Education, 14:177-185, 1998.

[Gamito95] M. N. Gamito, P. F. Lopes, and M. R. Gomes. Two-dimensional simulation of gaseous phenomena using vortex particles. In D. Terzopoulos and D. Thalmann, editors, Computer Animation and Simulation '95, pages 2-15. Eurographics, Springer-Verlag, ISBN 3-211-82738-2 1995.

[Garcke00] M. Garcke, T. Preußer, M. Rumpf, A. Telea, U. Weikard, and J. van Wijk. A continuous clustering method for vector fields. In Proc. of the 11th Ann. IEEE Visualization Conference (Vis) 2000, Salt Lake City, Utah, United States, pages 351-358. IEEE Computer Society Press, 2000.

[Gelder92] A. van Gelder and J. Wilhelms. Interactive visualization of flow fields. In Proceedings of the 1992 workshop on Volume visualization, Boston, Massachusetts, United States, pages 47-54, 1992.

[Gillespie01] J. Gillespie. The use of CFD in design of combustion equipment. Progress in Computational Fluid Dynamics, 1(1/2/3):80-90, 2001.

[Gilmartin81] P. P. Gilmartin. The interface of cognitive and psychophysical research in cartography. Cartographica, 18(3):9-20, 1981.

[Goktekin04] T. G. Goktekin, A. W. Bargteil, and O. O'Brien. A method for animating viscoelastic fluids. ACM Transactions on Graphics, 23(3):463-468, 2004 2004.

[Guthe02] S. Guthe, S. Gumhold, and W. Strasser. Interactive visualization of volumetric vector fields using texture based particles. In V. Skala, editor, Journal of WSCG 2002, volume 10(3), 2002.

[Guthe01] S. Guthe and W. Stra\ßser. Real-time decompression and visualization of animated volume data. In T. Ertl, K. Joy, and A. Varshney, editors, Proceedings of the Conference on Visualization 2001 (VIS-01), pages 349-356, Piscataway, NJ, 21-26 2001. IEEE Computer Society.

[Guthe04] S. Guthe and W. Strasser. Advanced techniques for high-quality multi-resolution volume rendering. Computers and Graphics, 28(1):51-58, 2004 2004.

[Harris03] M. J. Harris, W. V. Baxter, T. Scheuermann, and A. Lastra. Simulation of cloud dynamics on graphics hardware. In HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS con-

ference on Graphics hardware, San Diego, California, pages 92-101. Eurographics Association, 2003.

[Harris02] M. J. Harris, G. Coombe, T. Scheuermann, and L. Lastra. Physically-based visual simulation on graphics hardware. In SIGGRAPH/Eurographics Workshop on Graphics Hardware, Saarbrücken, Germany, pages 109-118. Eurographics Association, 2002.

[Heckbert91] P. Heckbert and H. Moreton. Interpolation for polygon texture mapping and shading. In State of the Art in Computer Graphics: Visualization and Modeling, pages 101-111. Springer-Verlag, 1991.

[Heidrich99] W. Heidrich, R. Westermann, H.-P. Seidel, and E. Ertl. Applications of pixel textures in visualization and realistic image synthesis. In ACM Symposium on Interactive 3D Graphics. ACM/Siggraph, 1999.

[Hemzal01] K. Hemzal. Aerodynamika větrání. Vydavatelstí ČVUT Praha, 2001.

[Hjertager00] B. H. Hjertager, T. Solberg, and M. Mathiesen. Predictions of gas/particle flow with an eulerian model including a realistic particle size distribution, 24 2000.

[Hrdlicka04] F. Hrdlicka, P. Slavik, O. Kubelka, and P. Hartel. New methodology for design of moving bed filters. WIT Transactions on Ecology and the Environment, 2004(78):57-65, 2004.

[Hrdlicka96] F. Hrdlicka, P. Slavik, and O. Kubelka. Simulation model of the moving granular bed gas cleanup filter. Air Pollution, 8:573-582, 2000.

[Chiang01] Yi-Jen Chiang, B. Wei, C. T. Silva, and R. Farias. A unified infrastructure for parallel out-of-core isosurface extraction and volume rendering of unstructured grids, 06 2001.

[Ibarria03] L. Ibarria, P. Lindstrom, R. Rossignac, and A. Szymczak. Out-of-core compression and decompression of large n-dimensional scalar fields. Computer Graphics Forum, 22(3):343-343, 2003.

[Ihm04] I. Ihm, B. Kang, and D. Cha. Animation of reactive gaseous fluids through chemical kinetics. In SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, Grenoble, France, pages 203-212, 2004.

[Ihrke04] I. Ihrke and M. Magnor. Image-based tomographic reconstruction of flames. In SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, Grenoble, France, pages 365-373, 2004.

[iLight-WWW] Inteligent Light company. Fieldview 8 web site. http://www.ilight.com/.

[Interrante97] V. Interrante and C. Grosch. Strategies for effectively visualizing 3D flow with volume LIC. In Proceedings of Visualization '97, pages 421-424, 1997.

[Jensen02] H. W. Jensen and J. Buhler. A rapid hierarchical rendering technique for translucent materials. In J. Hughes, editor, SIGGRAPH 2002 Conference Proceedings, Annual Conference Series, pages 576-581. ACM Press/ACM SIGGRAPH, 2002.

[Jobard02] B. Jobard, G. Erlebacher, and M. Y. Hussaini. Lagrangian-eulerian advection of noise and dye textures for unsteady flow visualization. IEEE Trans. Vis. Comput. Graph., 8(3):211-222, 2002.

[Jobard97] B. Jobard and W. Lefer. The motion map: efficient computation of steady flow animations. In Proceedings of the 8th conference on Visualization '97, Phoenix, Arizona, United States, pages 323-328. IEEE Computer Society Press, Los Alamitos, US, 1997.

[Jobard00] B. Jobard and W. Lefer. Unsteady flow visualization by animating evenly-spaced streamlines. In Computer Graphics Forum (Proceedings of Eurographics 2000), volume 19 of 3. Blackwell Publishers, 2000.

[Kadlec04] P. Kadlec, M. Gayer, and P. Slavík. Visualization using hardware accelerated spline interpolation. In V. Skala, editor, WSCG Short Communication Papers proceedings, Pilsen, Czech republic. UNION Agency - Science Press, Plzen, Czech Republic, 2004.

[Kass90] M. Kass and G. Miller. Rapid, stable fluid dynamics for computer graphics. In Proceedings of the 17th annual conference on Computer graphics and interactive techniques, pages 49-57, 1990.

[Kirby99] R. M. Kirby, H. Marmanis, and D. H. Laidlaw. Visualizing multivalued data from 2d incompressible flows using concepts from painting. In Proceedings of the conference on Visualization '99 : Celebrating ten years, San Francisco, California, United States, pages 333-340. IEEE Computer Society Press, 1999.

[Klasen00] T. Klasen and K. Gorner. The use of cfd for the prediction of problem areas inside a waste incinerator with regard to slagging, fouling and corrosion. In 5 th European Conference on Industrial Furnaces and Boilers, 2000.

[Klein02] J. Klein and F. M. A. D. Heide, K. Krokowski, M. Fischer, M. Wand, and R. Wanka. The randomized sample tree: A data structure for interactive, 2002.

[Kozel00] K. Kozel. Numerické řešení parciálních diferenciálních rovnic. Vydavatelstí ČVUT Praha, 2000.

[Kozel01] K. Kozel. Numerické metody řešení problémů proudění I. Vydavatelstí ČVUT Praha, 2001.

[Kreveld94] M. V. Kreveld. Efficient methods for isoline extraction from a digital elevation model based on triangulated irregular networks. In Sixth International Symposium on Spatial Data Handling, volume 2, pages 835-847, Edinburgh, Scotland, 1994.

[Kuester01] F. Kuester, R. Bruckschen, B. Hamann, and K. I. Joy. Visualization of particle traces in virtual environments. In Proceedings of the ACM symposium on Virtual reality software and technology, Baniff, Alberta, Canada, pages 151-157, 2001.

[Lamorlette02] A. Lamorlette and N. Foster. Structural modeling of flames for a production environment. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques, San Antonio, Texas, pages 729-735, 2002.

[Lane97] D. A. Lane. Scientific visualization of large-scale unsteady fluid flows, 23 1997.

[Langtangen01] H. P. Langtangen and O. Munthe. Solving systems of partial differential equations using object-oriented programming techniques with coupled heat and fluid flow as example. ACM Transactions on Mathematical Software, 27(1):1-26, 2001.

[Laramee04a] R. S. Laramee, H. Hauser, H. Doleisch, P. Post, B. Vrolijk, and D. Weiskopf. The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. Computer Graphics Forum, 23(2):203-221, 2004.

[Laramee04b] R. S. Laramee. Interactive 3D Flow Visualization Using Textures and Geometric Primitives. PhD thesis, Vienna University of Technology, Institute for Computer Graphics and Algorithms, Vienna, Austria, 2004.

[Laszlo92] M. J. Laszlo. Fast generation and display of iso-surface wireframes. CVGIP: Graph. Models Image Process., 54(6):473-483, 1992.

[Lefer04] W. Lefer, B. Jobard, and C. Leduc. High-quality animation of 2d steady vector fields. IEEE Trans. Vis. Comput. Graph., 10(1):2-14, 2004.

[Lichtenbelt98] B. Lichtenbelt, R. Crane, and S. Naqvi. Introduction to volume rendering. Prentice-Hall, Inc., 1998.

[Liu04] Y. Liu, X. Liu, and E. Wu. Real-time 3d fluid simulation on gpu with complex obstacles. In Pacific Conference on Computer Graphics and Applications, pages 247-256, 2004.

[Liu03] Z. Liu, R. J. Moorhead, and S. B. Ziegeler. Ocean flow visualization in virtual environment.

Technical Report MSSU-COE-ERC-03-03, Engineering Research Center, Mississippi State University, 2003.

[Liu02] Z. Liu and I. Robert James Moorhead. Auflic: an accelerated algorithm for unsteady flow line integral convolution. In VISSYM '02: Proceedings of the symposium on Data Visualisation 2002, Barcelona, Spain, pages 43-ff. Eurographics Association, 2002.

[Li03] W. Li, Z. Fan, X. Wei, and A. Kaufman. Gpu-based flow simulation with complex boundaries. Technical Report 031105, Computer Science Department, SUNY at Stony Brook, 2003.

[Loeffelmann97] H. Löffelmann, L. Mroz, G. Groeller, and W. Purgathofer. Stream arrows: enhancing the use of stream surfaces for the visualization of dynamical systems. The Visual Computer, 13(8):359-369, 1997. ISSN 0178-2789.

[Lokovic00] T. Lokovic and E. Veach. Deep shadow maps. In K. Akeley, editor, Siggraph 2000, Computer Graphics Proceedings, Annual Conference Series, pages 385-392. ACM SIGGRAPH / Addison Wesley Longman, 2000.

[Lum01] E. B. Lum, K. L. Ma, and J. Clyne. Texture hardware assisted rendering of time-varying volume data. In Proceedings of the conference on Visualization 2001, San Diego, California, pages 263-270. IEEE Computer Society Press, 2001.

[Magel95] H. C. Magel, B. Risio, K. R. G. Hein, S. Schneider, and U. Schnell. Numerical simulation of utility boilers with advanced combustion technologies, 11 1997.

[Machiraju98] R. Machiraju, Z. Zhu, B. Fry, and R. Moorhead. Structure-significant representation of structured datasets. IEEE Transactions on Visualization and Computer Graphics, 4(2):117-132, 1998.

[Mason98] W. Mason, N. Jackie, T. Davis, D. Shreiner, and OpenGL Architectural Review Board. The OpenGL Programming Guide. Addison-Wesley, 1998.

[Mattausch03] O. Mattausch, T. Theußl, H. Hauser, and E. Groller. Strategies for interactive exploration of 3d flow using evenly-spaced illuminated streamlines. In SCCG '03: Proceedings of the 19th spring conference on Computer graphics, Budmerice, Slovakia, pages 213-222, 2003.

[Max94] N. Max, R. Crawfis, and C. Grant. Visualizing 3d velocity fields near contour surfaces. In VIS '94: Proceedings of the conference on Visualization '94, Washinton, D.C., pages 248-255. IEEE Computer Society Press, 1994.

[Ma00] K.-L. Ma and D. M. Camp. High performance visualization of time-varying volume data over

a wide-area network status. In Proceedings of Supercomputing'2000 (CD-ROM), Dallas, TX, 2000 2000. IEEE and ACM SIGARCH. Univ. of California, Davis.

[Ma98] K. Ma, D. Smith, M. Shih, and H. Shen. Efficient encoding and rendering of time-varying volume data. ICASE Report No. 98-22 NASA/CR-1998-208424, Institute for Computer Applications in Science and Engineering, NASA Langsley Research Center, 1998. Presented at SIGGRAPH '99.

[McAllister00] D. McAllister. The design of an api for partic le systems, 2000.

[Most00] J.-M. Most, P. Trouillet, P. Mandin, F. Marchand, C. Le-Masson, P. Witwicki, and D. Rampelberg. Development of a ldv probe for velocity measurements in a 600 mw pulverized coal power plant. In 10th International Symposium of Applications of Laser Techniques to Fluid Mechanics, 2000.

[Nguyen02] D. Q. Nguyen, R. Fedkiw, and H. W. Jensen. Physically based modeling and animation of fire. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques, San Antonio, Texas, pages 721-728. ACM Press, 2002.

[O'Brien95] J. F. O'Brien and J. K. Hodgins. Dynamic simulation of splashing fluids. In CA '95: Proceedings of the Computer Animation, page 198. IEEE Computer Society, 1995.

[OGL-ExReg-WWW] Silicon Graphics, Inc. OpenGL® Extension Registry. http://oss.sgi.com/projects/ogl-sample/registry/.

[Pighin04] F. Pighin, J. M. Cohen, and M. Shah. Modeling and editing flows using advected radial basis functions. In SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, Grenoble, France, pages 223-232. ACM Press, 2004.

[Polthier00] K. Polthier and E. Preu. Variational approach to vector field decomposition, 29 2000.

[Post02] B. Vrolijk, F. H. Post, H. Doleisch, H. Hauser, and R. S. Laramee. EUROGRAPHICS 2002 STAR - state of the art report feature extraction and visualisation of flow fields, 08 2002.

[Post03] F. H. Post, B. Vrolijk, H. Hauser, L. Laramee, and H. Doleisch. The State of the Art in Flow Visualization: Feature Extraction and Tracking. Computer Graphics Forum, 22(4):775-792, 2003. forthcoming.

[Preusser99] T. Preußer and M. Rumpf. Anisotropic nonlinear diffusion in flow visualization. In D. Ebert, M. Gross, and B. Hamann, editors, IEEE Visualization '99, pages 325-332, San Francisco, 1999. IEEE.

[Rais97] D. Rais. Visualization of combustion processes in fluidized bed boilers. Master's thesis, CTU FEE Prague, 199.

[Rasmussen04] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. In SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, Grenoble, France, pages 193-202, 2004.

[Rasmussen03] N. Rasmussen, D. Q. Nguyen, W. Geiger, and R. Fedkiw. Smoke simulation for large scale phenomena. In J. Hodgins and J. C. Hart, editors, Proceedings of ACM SIGGRAPH 2003, volume 22(3) of ACM Transactions on Graphics, pages 708-715, 2003.

[Reeves83] W. T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. In Proceedings of SIGGRAPH '83, vol. 17, n. 3, pages 359-376. ACM Computer Graphics, 1983.

[Rhodes98] M. Rhodes. Introduction to particle technology. John Wiley Sons Ltd., 1998.

[Robbins00] K. A. Robbins and M. Gorman. Fast visualization methods for comparing dynamics: A case study in combustion. In Proceedings of the 11th IEEE Visualization 2000 Conference (VIS 2000). IEEE Computer Society, 2000.

[Roettger03] S. Roettger, S. Guthe, D. Weiskopf, T. Ertl, and W. Strasser. Smart hardware-accelerated volume rendering. In C. D. H. G.-P. Bonneau, S. Hahmann, editor, Proceedings of the symposium on Data visualisation 2003, Grenoble, France, pages 231-238. Eurographics Association, 2003.

[Sadarjoen98] I. A. Sadarjoen, A. J. de Boer, F. H. Post, and A. E. Mynett. Patricle tracing in σ-transformed grids using tetrahedral 6-decomposition. In D. Bartz, editor, Visualization in Sientific Computing '98, Eurographics, pages 71-80. Springer-Verlag Wien New York, 1998.

[Sadarjoen99] I. Sadarjoen and F. Post. Geometric Methods for Vortex Detection. pages 53-62, 1999.

[Sanna00] A. Sanna, B. Montrucchio, and R. Arina. Visualizing unsteady flows by adaptive streaklines, 22 2000.

[Sapede01] J. Sapede, J.-L. Harion, S. Caillat, and B. Baudoin.. Complex coaxial rectangular jet with density differences : Numerical simulations vs. experimental datas. In Proceedings of the 2nd International Conference on Computational Heat and Mass Transfer, 2001.

[Sayre99] A. Sayre and M. Milobowski. Validation of numerical models of flow through scr units. In EPRI-DOE-EPA Combined Utility Air Pollutant Control Symposium, 1999.

[Schroeder96] W. J. Schroeder, K. M. Martin, and W. E. Lorensen. The design and implementation of an object-oriented toolkit for 3D graphics and visualization. In R. Yagel and G. M. Nielson, editors, IEEE Visualization '96, pages 93-100, 1996.

[Schulz99] M. Schulz, F. Reck, W. Bartelheimer, and T. Ertl. Interactive visualization of fluid dynamics simulations in locally refined cartesian grids (case study). In Proceedings of the conference on Visualization '99 : Celebrating ten years, San Francisco, California, United States, pages 413-416. IEEE Computer Society Press, 1999.

[Shen99] H.-W. Shen, L.-J. Chiang, and K.-L. Ma. A fast volume rendering algorithm for time-varying fields using a time-space partitioning (TSP) tree. In D. Ebert, M. Gross, and B. Hamann, editors, IEEE Visualization '99, pages 371-378, San Francisco, 1999. IEEE.

[Shen96] H.-W. Shen, C. R. Johnson, and K.-L. Ma. Visualizing vector fields using line integral convolution and dye advection. In Proceedings of the 1996 symposium on Volume visualization, San Francisco, California, United States, pages 63-ff. IEEE Press, 1996.

[Shen97] H.-W. Shen and D. L. Kao. UFLIC: A line integral convolution algorithm for visualizing unsteady flows. In R. Yagel and H. Hagen, editors, IEEE Visualization '97, pages 317-323. IEEE, 1997.

[Shen98] H.-W. Shen and D. L. Kao. A new line integral convolution algorithm for visualizing time-varying flow fields. IEEE Transactions on Visualization and Computer Graphics, 4:98-108, 2 1998. Abstract: http://www.computer.org/tvcg/tg1998/v0098abs.htm.

[Shen04] H.-W. Shen, G.-S. Li, and U. D. Bordoloi. Interactive visualization of three-dimensional vector fields with flexible appearance control. IEEE Transactions on Visualization and Computer Graphics, 10(4):434-445, July Aug. 2004.

[Slavik99] P. Slavík. Scientific visualization as a tool for power engineering education. In Gve'99 Computer Graphics And Visualization Education '99, pages 165-170, 1999. http://www.siggraph.org/education/conferences/GVE99/papers/GVE99.P.Slavik.pdf.

[Smith-WWW] The University of Utah. Combustion research group. http://www.combustion.utah.edu.

[Sobel04] J. S. Sobel, A. S. Forsberg, L. Laidlaw, R. C. Zeleznik, D. F. Keefe, I. Pivkin, G. E. Karniadakis, R. Richardson, and S. Swartz. Particle flurries: Synoptic 3D pulsatile flow visualization. IEEE Computer Graphics and Applications, 24(2):76-85, 2004.

[Sohn02] B.-S. Sohn, C. Bajaj, and S. Siddavanahalli. Feature based volumetric video compression for

interactive playback. In S. N. Spencer, editor, Proceedings of the 2002 IEEE symposium on Volume visualization and graphics (VOLVIS-02), pages 89-96, Piscataway, NJ, 28-29 2002. IEEE.

[Stalling95] D. Stalling and H. Hege. Fast and resolution independent line integral convolution. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pages 249-256, 1995.

[Stam95] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In R. Cook, editor, SIGGRAPH 95 Conference Proceedings, Annual Conference Series, pages 129-136. ACM SIGGRAPH, Addison Wesley, held in Los Angeles, California, 06-11 August 1995 1995.

[Stam99] J. Stam. Stable fluids. In Proceedings of Computer Graphics and interactive techniques, volume 33, pages 121-128. ACM Press, 1999.

[Stam00] J. Stam. Interacting with smoke and fire in real time. Communications of the ACM, 43(7):76-83, 2000 2000.

[Stam01] J. Stam. A simple fluid solver based on the FFT. Journal of Graphics Tools: JGT, 6(2):43-52, 2001.

[Stam03a] J. Stam. Flows on surfaces of arbitrary topology. ACM Trans. Graph., 22(3):724-731, 2003.

[Stam03b] J. Stam. Real-time fluid dynamics for games, 2003.

[Stock02] M. J. Stock. Creative flow visualization using a physically accurate lighting model. In 2nd Conference on Computational Semiotics for Games and New Media, pages 77-83, 2002.

[Swann91] P. Swann and S. Semwal. Volume rendering of flow-visualization point data. In In Proceedings of IEEE Visualization '91, pages 25-32. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[Takai95] Y. Takai, K. Ecchu, and N. Takai. A cellular automaton model of particle motions and its applications. The Visual Computer, 11(5):240-252, 1995.

[Taponecco03] F. Taponecco and M. Alexa. Vector field visualization using markov random field texture synthesis. In C. D. H. G.-P. Bonneau, S. Hahmann, editor, Proceedings of the symposium on Data visualisation 2003, Grenoble, France, pages 195-202. Eurographics Association, 2003.

[Tecplot-WWW] Amtec company. Teplot 9.0 data visualisation software .

http://www.amtec.com/Product_pages/tecplot9/index.html.

[Telea94] A. Telea and J. J. van Wijk. Simplified representation of vector fields. In Proceedings of the conference on Visualization '99 : Celebrating ten years, San Francisco, California, United States, pages 35-42. IEEE Computer Society Press, 1999.

[Telea03] A. Telea and J. J. van Wijk. 3D IBFV: Hardware-accelerated 3D flow visualization. In Proceedings of the IEEE Visualization '03. ACM Press, 2003.

[Teyssler88] J. Teyssler. Spalování popelnatých hnědých uhlí. SNTL Praha, 1988.

[Theisel03] H. Theisel and H.-P. Seidel. Feature flow fields. In C. D. H. G.-P. Bonneau, S. Hahmann, editor, Proceedings of the symposium on Data visualisation 2003, Grenoble, France, pages 141-148. Eurographics Association, 2003.

[Tomeczek95] J. Tomeczek. Coal Combustion. Krieger Publishing Company, 1994.

[Treuille03] A. Treuille, A. McNamara, P. Popovic, and J. Stam. Keyframe control of smoke simulations. In J. Hodgins and J. C. Hart, editors, Proceedings of ACM SIGGRAPH, volume 22(3) of ACM Transactions on Graphics, pages 716-723, 2003.

[Upson89] C. Upson, J. T. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam. The application visualization system: A computational environment for scientific visualization. IEEE Computer Graphics and Applications, 9(4):30-42, 1989.

[Wijk91] J. J. van Wijk. Spot noise texture synthesis for data visualization. In Proceedings of the 18th annual conference on Computer graphics and interactive techniques, pages 309-318, 1991.

[Wijk02] J. J. van Wijk. Image based flow visualization. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques, San Antonio, Texas, pages 745-754, 2002.

[Warnatz95] J. Warnatz, U.Maas, and R. Dibble. Combustion Physical and Chemical Fundamentals, Modelling and Simulation, Experiments, Pollutant Formation. Springer-Verlag Berlin Heidelberg, 1995.

[Waterman00] P. J. Waterman. CFD in the trenches - desktop engineering. http://www.deskeng.com/articles/00/July/CFD/index.htm, 2000.

[Wegenkittl97a] R. Wegenkittl, E. Groller, and W. Purgathofer. Animating flow fields: Rendering of oriented line integral convolution. In CA '97: Proceedings of the Computer Animation, page 15. IEEE Computer Society, 1997.

[Wegenkittl97b] R. Wegenkittl and E. Gröller. Fast oriented line integral convolution for vector field visualization via the internet. In IEEE Visualization '97 Proceedings, pages 309-316. IEEE Computer Society, 1997 1997.

[Weimer99] H. Weimer and J. Warren. Subdivision schemes for fluid flow. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques, Malaga, Spain, pages 111-120, 1999.

[Weiskopf03] D. Weiskopf, G. Erlebacher, and T. Ertl. A Texture-Based Framework for Spacetime-Coherent Visualization of Time-Dependent Vector Fields. In Procceedings of IEEE Visualization '03, pages 107-114, 2003.

[Weiskopf04a] D. Weiskopf and T. Ertl. GPU-based 3D texture advection for the visualization of unsteady flow fields. In V. Skala, editor, WSCG Short Communication Papers proceedings. UNION Agency - Science Press, Plzen, Czech Republic, 2004.

[Weiskopf01] D. Weiskopf, M. Hopf, and T. Ertl. Hardware-accelerated visualization of time-varying 2D and 3D vector fields by texture advection via programmable per-pixel operations. In T. Ertl, B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, editors, Proceedings of the Vision, Modeling, and Visualization VMV '01 Conference, Stuttgart, Germany, pages 439-446, 2001.

[Weiskopf04b] D. Weiskopf, T. Schafhitzel, and T. Ertl. Gpu-based nonlinear ray tracing. Comput. Graph. Forum, 23(3):625-634, 2004.

[Wei03] X. Wei, W. Li, and A. Kaufman. Melting and flowing of viscous volumes. In CASA '03: Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003), page 54. IEEE Computer Society, 2003.

[Wei02] X. Wei, W. Li, K. Mueller, and K. Kaufman. Simulating fire with texture splats. In R. Moorhead, M. Gross, and K. I. Joy, editors, Proceedings of the 13th IEEE Visualization 2002 Conference (VIS-02), pages 227-234, Piscataway, NJ, 27- 2002. IEEE Computer Society.

[Wei04] X. Wei, Y. Zhao, Z. Fan, W. Li, F. Qiu, S. Yoakum-Stover, and K. Kaufman. Lattice-based flow field modeling. IEEE Transactions on Visualization and Computer Graphics, 10(6):719-729, 2004.

[Wejchert91] J. Wejchert and D. Haumann. Animation aerodynamics. In Proceedings of the 18th annual conference on Computer graphics and interactive techniques, pages 19-22, 1991.

[Westermann00] R. Westermann, C. Johnson, and T. Ertl. A level-set method for flow visualization.

In Proc. of the 11th Ann. IEEE Visualization Conference (Vis) 2000, pages 147-154, 2000.

[Wischgoll02] T. Wischgoll and G. Scheuermann. Locating closed streamlines in 3D vector fields. In Proceedings of the symposium on Data visualisation 2002, Barcelona, Spain, pages 227-232. Eurographics Association, 2002.

[Witting99] P. Witting. Computational fluid dynamics in a traditional animation environment. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pages 129-136, 1999.

[Wu04] E. Wu, Y. Liu, and X. Liu. Journal of computer animation and virtual world. Communications of the ACM, 15(3-4):139-146, 2004.

[Yoshida00] S. Yoshida and T. Nishita. Modelling of smoke flow taking obstacles into account. In 8th Pacific Conference, pages 135-144, 2000.

[Zhao03] Y. Zhao, X. Wei, Z. Fan, A. Kaufman, and H. Qin. Voxels on fire. In IEEE Visualization2003, San Diego, California, pages 271-278. Eurographics Association, 2003.

[Zockler96] M. Zöckler, D. Stalling, and H. C. Hege. Interactive visualization of 3d-vector fields using illuminated stream lines. In Proceedings of the conference on Visualization '96, San Francisco, California, United States, pages 107-113. IEEE Computer Society Press, 1996.

Publications of the author

Publications in refereed journal papers

[Gayer02f] M. Gayer, F. Hrdlička, and P. Slavík. Dynamic visualisation of the combustion processes in boilers. *Journal of WSCG*, pages 25-32, 2002.

Publications on international conferences

[Gayer02a] M. Gayer, F. Hrdlička, and P. Slavík. Dynamic visualisation of the combustion processes in boilers. In *Proceedings of The 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2002 (CDROM)*. University of West Bohemia, Pilsen, 2002.

[Gayer02b] M. Gayer, P. Slavík, and F. Hrdlička. Interactive system for pulverized coal combustion visualization with fluid simulator. In *Proceedings of the Visualization, Imaging, and Image Processing Conference*, Pilsen, Czech republic, pages 288-293. IASTED, Anaheim: Acta Press, 2002.

[Gayer03a] M. Gayer, P. Slavík, and F. Hrdlička. Real-time simulation and visualization using pre-calculated fluid simulator states. In *Proceedings of the Visualization, Imaging, and Image Processing Conference*, London, UK, pages 440-445. IEEE Computer Society Press, Los Alamitos, 2003.

[Gayer03b] M. Gayer and P. Slavík. Pre-calculated fluid simulator states tree. In *Proceedings of 12th IASTED International Conference on Applied Simulation and Modelling*, Marbella, Spain, pages 610-615. IASTED, Anaheim: Acta Press, 2003.

[Gayer03c] M. Gayer, P. Slavík, and F. Hrdlička. Interactive educational system for coal combustion modeling in power plant boilers. In *Proceedings of the EUROCON 2003 - Computer as a Tool*, Ljubljana, Slovenia, volume 2, pages 220-224. IEEE Computer Society Press, 2003.

[Gayer03f] M. Gayer. Modeling and visualization of combustion using fluid simulator and particle systems. In *Proceedings of the International Conference and Competition STUDENT EEICT 2003*, pages 306-309. VUT Brno, 2003.

[Slavik03] P. Slavík, M. Gayer, F. Hrdlička, and O. Kubelka. Problems of visualization of technological processes. In *Proceedings of the 2003 Winter Simulation Conference*, New Orleans, pages 746-754. Omnipress, 2003.

[Gayer04a] M. Gayer and P. Slavík. Hierarchical trees of unsteady simulation datasets. In *Proceedings of 13th IASTED International Conference on Applied Simulation and Modelling*, Rhodes, Greece,

pages 303-308. IASTED, Anaheim: Acta Press, 2004.

[Kadlec04] P. Kadlec, M. Gayer, and P. Slavík. Visualization using hardware accelerated spline interpolation. In V. Skala, editor, *WSCG Short Communication Papers proceedings*, Pilsen, Czech republic. UNION Agency - Science Press, Plzen, Czech Republic, 2004.

Other publications

[Gayer00] M. Gayer. MGL graphics library. In *User Interfaces Visualization, 8th Bilateral student workshop*. HTW Dresden, 2000.

[Gayer01] M. Gayer and M. Snorek. Modern graphical accelerators. *Softwarové noviny*, 12(7):72-84, 2001.

[Gayer02c] M. Gayer. Simulation and visualization of combustion processes in pulverized coal boilers. In *Proceedings of Workshop 2002*, volume B, pages 782-783. CTU in Prague, 2002.

[Gayer02d] M. Gayer. Coal combustion processes visualisation using particle system. In *Proceedings of CTU Poster 2002*. CTU in Prague, 2002.

[Gayer02e] M. Gayer and P. Slavik. Interactive simulation of pulverized coal combustion. In *User Interfaces Visualization, 10th Bilateral student workshop*. HTW Dresden, 2002.

[Gayer03d] M. Gayer. Simulation and visualization of combustion processes in pulverized coal boilers. In *Proceedings of Workshop 2003*. CTU in Prague.

[Gayer03e] M. Gayer. Simulation and visualization of combustion powered by fluid simulator. In *Proceedings of Poster 2003*, page PE10. CVUT FEE Prague, 2003.

[Gayer04b] M. Gayer. Fast simulation of fluids. In *Proceedings of Workshop 2004*. CTU in Prague, 2004.