WP 6.2 Project Deliverable

VIRTUALFIRES results report

Project Number	IST-2000-29266
Project Title	Virtual Real Time Fire Emergency Simulator
Deliverable Type	Evaluation report
Deliverable Class	External

Deliverable Number	D 6.2
Title of Deliverable	Virtualfires results report
Nature of the Deliverable	Report
Contributing WPs	WP 2, 4, 5, 7
Contractual Date of Delivery	01 March 04
Actual Date of Delivery	09 Sep. 04
URL	www.virtualfires.org
Authors	Javier Urruzola, Gunther Lenz, Christian Redl, Thorsten Mai, Gert Svensson
Contact Details	European Virtual Engineering / EUVE Dr. Javier Urruzola Avda. de los Huetos, 79 01010 Vitoria / Spain Tel.: +34 945 21 46 46 Fax: +34 945 21 46 47 Email: jurruzola@euve.org

Abstract	Evaluation report of Virtual Fires System
Keywords	Validation, Evaluation, Software, Simulation

1 Introduction

Workpackage 6, Evaluation and Validation, consists of two different tasks:

Task 6.1. End-user evaluation of the Virtual Fires System in the CAVE and PC versions.

Methodology: End-users take part in test cases to evaluate the quality in use of the system. This is complemented with the focus group technique to assess user needs and feelings.

Task 6.2. Functionality validation (validation against specifications).

Methodology: Developers test that code sticks to specifications and that functionality is appropriate.

Both taks have been be carried out in accordance with ISO. This deliverable contains the results of task 6.2..

2 Functionality evaluation procedure description

2.1 Introductory remarks

The information collected in this report is about the information that will be need to do the general validation of the system developed in this project.

The framework for evaluation which will be applied in the project is an ISO-based framework, and it will be an evaluation about functionality of the system. This kind of evaluation determines the extent to which a software system/component provides functions which meet stated and implied needs under specified conditions.

ISO/IEC 9126 defines the “functionality” in terms of sub-characteristic, being each sub-characteristic measurable through the measure of more sub-items.

2.2 Scope

2.2.1 Characteristics

Functionality metrics indicate if the software component satisfies the defined requirements.

Metrics for functionality include indicators for 5 sub-characteristics:

- Suitability
- Accuracy
- Interoperability
- Compliance
- Security

Suitability metrics measure the ratio of the satisfied functions during testing and user operations to the required functions; in other words, functions performing tasks which do not conform to documented requirements.

Accuracy metrics measure the precision of the system:

- The range of erroneous of calculation
- The difference between actual and expected results of tasks performed
- Inconsistency between actual operation procedure and documented procedure (for example, in manuals)

Interoperability metrics measure the communicability level of the software component with other systems, other software products other equipment:

- Data transfer capability
- Command exchange capability

Compliance metrics measure the standardisation level of the software component against regulation or rules of the environment.

Security metrics measure the defence performance level against illegal access and/or illegal operations.

2.2.2 Level of evaluation

Considering safety, economy, security, and environmental aspects, the level of evaluation of this project according to ISO/IEC 14598-6:2001(E) will be level *D*, because in this project there is no damage to property and no risk to people, the possible economic losses are negligible, there is not specific risk identified in relation to security aspects, since it is not a database containing sensitive information, and finally there is not environmental risk.

2.3 Technique

According to ISO/IEC 14598-6:2001, the appropriate techniques for the level of evaluation *D* are based on functional testing. This is a process of attempting to find discrepancies between the software product and its external specification, where it is important to well define the test cases and apply specific techniques and methods.

2.3.1 Applicability

This type of evaluation is applicable when two kinds of conditions are present:

- Condition concerning the requirement for the evaluation process:

  Applicable when the specific requirements for the evaluation process for the characteristic under evaluation are satisfied (software requirements for functionality).

- Condition concerning the documentation input to the evaluation:

  Applicable to the availability of the documentation input to the evaluation process.

2.3.2 Requirements for the Evaluation Process

Software requirements for functionality.

2.3.2.1 SUITABILITY

- All functional requirements shall be documented (This information is available in deliverables 2.4 and 2.6)
- Hardware architecture of the product shall be described in the documentation.(deliverable 2.3)
- Software architecture of the product shall be described in the documentation. (deliverables of WP 3,4 and 5)

2.3.2.2 ACCURACY

- All functions mentioned in the software documentation shall be completely and correctly executable.

2.3.2.3 SECURITY

- Not relevant.

2.3.3 Documentation INPUT to the Evaluation

Applicable to the availability of the documentation input to the evaluation process.

The software components and documentation required, according with the level of evaluation *D*, are the following:

- Executable product (this means either running the product in the evaluation environment or having the access to the target environment with the possibility of running the product there).
- Product description (this is a collection of information about what the user expects from product. It may have the form of “product coversheet”, or “user requirements”, or other information.
- User manual: not available
- Test cases (these include test data and test results. The evaluation process will utilise such information, but is not limited to it.)

2.3.3.1 Applicability

In the software life cycle, this evaluation module may be used before the delivery of the product.

2.4 Inputs and metrics

2.4.1 Input for the evaluation

The minimal input documentation for the level of evaluation of this project, according to ISO/IEC 14598-6:2001, is the following: executable product, product description, and test cases.

2.4.2 Data elements

The information to be extracted from the input documents is two types:

- Information to the “evaluation process”, for example, requirements list
- Information useful to “understand the system”

The first type of information are data to the evaluation process while the second one is not subject to evaluation but it consists of various informal documents furnished by the developer to the evaluator.

2.4.3 Metrics and measures

The following is the decomposition of the functionality in sub-characteristics and for each sub-characteristic the metric to be used for the evaluation is indicated.

2.4.3.1 Metrics for COMPLIANCE

Not applicable

2.4.3.2 Metrics for SUITABILITY

2.4.3.2.1 Functions available ratio

The ratio of functions effectively at disposal to the user to total number of specified function.

2.4.3.2.2 Project documentation ratio

The ratio of project documents available with the product to total number of project documents required.

2.4.3.2.3 Product documentation ratio

The ratio of product documents available with the product to total number of product documents required.

2.4.3.3 Metrics for ACCURACY

2.4.3.3.1 Significant digits ratio

The ratio of the implemented significant digits to the required significant digits for the data items that require specific accuracy.

2.4.3.4 Metrics for INTEROPERABILITY

Not applicable.

2.4.3.5 Metrics for SECURITY

Not applicable.

3 Functionality evaluation results

In accordance with the procedure specified above, the evaluation of functionality has been performed by checking the fulfillment of project specifications collected in several documents:

D2.4 Functionality specification

D7.1 Functionality for each testcase

D*.* Other specifications.

These deliverables together with the project plan clearly specify the partners responsible for each part of the code. In the following paragraphs, each partner contribution to the functionality evaluation is stated.

The following summarized results supported by extended reports from each partner are given:

3.1 Metrics for SUITABILITY

3.1.1 Functions available ratio

"The ratio of functions effectively at disposal to the user to total number of specified function."

100% of functions are available with respect to final functionality specification.

About 80% of functions are available with respect to the ideal list of specifications in D2.4. This type of metrics is however tricky because its value depends on the way we classiby functions. The assessment of the functions available ratio is excellent taking into account the high degree of innovation and ambition of the project: putting together a CAVE and ral time interactive fire simulation is a remarkable achievement.

3.1.2 Project documentation ratio

The ratio of project documents available with the product to total number of project documents required. 100% of project deliverables available. The assessment is escellent again.

3.1.3 Product documentation ratio

The ratio of product documents available with the product to total number of product documents required. User manual is missing. 100% of project deliverables related to product documentation available. It is clear that the project goals have been achieved, but the lack of a user manual is a limiting factor for the commercialization of the product.

3.2 Metrics for ACCURACY

3.2.1 Significant digits ratio

The ratio of the implemented significant digits to the required significant digits for the data items that require specific accuracy. According to the tests carried out by CD (see below) accuracy is satisfactory. However, taking into account the results of the focus group sessions, it can be clearly seen that in order to compete with established software, end users demand a large collection of validation examples like the Ercoftac list that would require an effort that on the other hand is not possible in the frame of this project.

3.3 IGD functionality evaluation

3.3.1 The task

For the VIRTUALFIRES project, IGD had the task to develop and implement a parallel visualization tool for the rendering of numerical simulation data in to a CAVE or a HMD. According to the different end users of the VIRTUALFIRES system, there where a number of different demands, which the rendering has to be able to do.

- Tunnel operators want the system to be capable of displaying the distribution of the temperature and the toxic gases. Furthermore they rely on a realistic rendering of the fire and the smoke in order to determine the visibility conditions in case of tunnel fire and to assess the validity of different evacuation plans.

- Fire-fighting personnel will use the system in a training scenario. Also here, a realistic rendering of fire and smoke is needed, because this is the only way to “embed” the training personnel into the scene. The also applies to a demonstration, where a scenario should be presented as impressive as possible.

- To be as flexible as possible a number of different scientific visualization methods must be developed which can be chosen according to the type of data actually displayed. Any visualization currently active can be assigned to any dataset read from the simulation output can be processed for rendering. All visualization parameters available for a given method must be changed interactively.

- Finally the rendering system must be scalable, i.e. it must adapt to a wide number of machines with a performance ranging from a laptop to a workstation driving the CAVE.

In the prototype the visualization system is embedded into a scenegraph system (“COVER”), which contains all geometrical and texture information necessary for the rendering. It natively provides different interfaces for rendering to the desktop or CAVE and HMD devices and supports a communication protocol for exchanging information between the different parts of the VIRTUALFIRES system.

3.3.2 The work done

The skeleton of the rendering engine developed by IGD is a parallel processing kernel, which assigns the available computation time to the different working procedures and a data source manager.

The data source manager is responsible for receiving the different datasets (containing temperature fields, velocity fields etc…) from the simulation part of the system. It transfers the data into a format suitable for efficient rendering and stores it as long as they are needed or – in case of a concurrent simulation – another dataset is available.

As the datasets are coming in different packages, it also is responsible for synchronizing these packages and releasing them for further use.

The actual visualization is done using different visualization methods which can be added one by one to the given system, without any interference to the others. In fact any visualization method developed “logs in” to the system, which collects all methods in a visualization method pool, which are now available for selection via the given interface. Aside from a number of parameters commonly shared by all visualization methods, the methods can have their own parameter sets, thus maximizing the flexibility while developing and adding new methods to the pool.

The probe is another main concept within the system. It encapsulates a field of the dataset – the data source, a visualization method and an area of interest. It can be seen as a bounding box, which can be moved freely throughout the data volume, rendering the field of the dataset given that lies inside the box with the visualization method selected. The probe concept enables us to the integrate different visualizations (even of different data) into a single image, thus allowing to analyze the correlations between them.

Interactivity is a crucial demand to the VIRTUALFIRES system as a whole. Correspondingly this has some implications to the design of the rendering engine. Certainly the most important of all is the use of efficient and parallelized rendering algorithms which are able to produce the geometry information, which is passed to the graphics hardware as fast as possible. Furthermore the assignment of computation resources done by the kernel is a key feature to the system. If looked upon closely, actually any input from the user necessitates a new computation. Those input include :

- The moving of a probe

- The changing of the dataset (f.e. if a new simulation step has been computed and new data is available)

- The changing of the internal parameters of a visualization

- The movement of the user

Theoretically any new input can trigger an new computation, which could result in collapse of the performance if this is done too often, hence a scheduler collects all parameters belonging to a given probe and triggers a single computation after a appropriate time interval, if resources are free. Furthermore, the user has full control over the complexity of a visualization (usually measured by the number of geometry elements in the scene). While being able to trade off the complexity/accuracy of a scene the visualization can be adapted to virtually any systems performance. Additionally if a given hardware is capable of using multiple processors (via multi-threading), the computation of the visualization methods can be parallelized, without any overhead due to preprocessing or postprocessing work.

All these features enable us to control all aspects of the scientific or realistic visualization (short of the actual rendering, which depends on the power of the underlying graphics hardware) in a responding time acceptable by the user. Including the change of time as a (non–user) input, the user is able to see and analyze the development of the simulation, i.e. the development and spreading of the fire and smoke in the tunnel, the motion of (toxic) gases and visibility conditions in real time (i.e. as fast as the simulator runs).

For the VIRTUALFIRES prototype system, four different visualization methods have been developed and incorporated. Two of them are able to display vector field data (velocity fields in our case), the other two are able to display scalar field data (temperature, smoke/toxic gases density or combinations of both). For increased flexibility, only vector fields and scalar fields are accounted as different kinds of data – any visualization method can be assigned to any source field of the given type.

The actual visualization methods include

- Streamlines visualization

- Line integral convolution

- Isosurface extraction

- Realistic fire & smoke rendering

Streamlines and Line integral convolution both are scientific visualization methods for vector fields. They are suitable to display the flow of gases or particles through the simulation domain. The former uses geometry (linestrips), the latter uses noise, which is distorted by the flow field depicted on a intersection plane. Using appropriate coloring schemes, additional information can be carried through this visualizations: A selection of “color-maps” is available which map the sampled temperature or density onto the lines or the plane.

The extraction of an isosurface is a scientific visualization method which is restricted to the depiction of scalar fields. The algorithm creates a surface of points which have the same temperature or smoke density value. Hence the hazard regions (due to high temperature or toxicity) inside the tunnel can be easily identified. As the simulation propagates, the isosurfaces move on through the tunnel, affected by the air flow.

One common parameter in all four visualization methods is the sampling rate: In general, with the benefit of increased accuracy the sampling rate can be increased up to a maximum amount. Usually this creates

lines and surfaces with “smoother” features, at the cost of more geometry information to be passed to the graphics hardware and a decreased rendering performance.

The last and most important visualization method is the realistic depiction of smoke and fire inside the tunnel – using so-called “volume-rendering”. Since it also is the most complex method the approach and performance differs depending on the type of graphics hardware available. Basically it is an algorithm to render (semi-)transparent gases or fluids in a three-dimensional domain using 3D-textures. A 3D texture simply is a box type grid with a given resolution, where all scalar field data is mapped onto. An array of semitransparent planes, perpendicular to the viewing direction, is laid through the volume. Where a plane intersects the volume the texture is sampled onto the plane (i.e. the plane will be “(3d-)textured”), using an appropriate coloring scheme (using “color-maps”, as mentioned above). As the planes are basically near-transparent, the viewer can look “behind” them. In fact, this algorithm integrates all sampling values along a ray shot from the eye-point of the user: If the volume contains regions with high smoke density, the resulting image is likely to be opaque when looking at these region, thus blocking the sight to anything behind. When looking at regions with clean air, the image simply will show the background of the scene (i.e. the tunnel walls).

The big problem with this approach is, that the resampling must be done even only if the spectator moves. In all other visualization methods the geometry does not depend on the viewing-position or –direction – one just moves around a rather “fixed” geometry. The consequence is that the geometry must be set up and rendered multiple times in a second. Depending on the capabilities of the graphics hardware, two options arise:

- The hardware is capable of 3D-texturing. In this case the texture and the planes is passed to the hardware (a “cheap” operation, since the texture is fixed anyway and planes can be defined with a few points) and the sampling and the integration is done in hardware, producing the final pixel color. This usually is fast enough to outweigh the problem, that the volume must be rendered multiple times.

- The hardware is *not* capable of 3D-texturing. In this case the sampling must be done in software, which may be quite expensive. More importantly, this operation may create large amounts of geometry as the planes must be tessellated using the sampling points. Unfortunately hardware not capable of 3D-texturing is unlikely to be able to set up and render large amounts of geometry (due to the extended setup time).

Both cases have been encountered, and both algorithms have been implemented and tested: The software solution currently runs on the SGI Onyx II, which is used to drive the CAVE engine and on the Linux based machines. Using a CAVE there is still another aspect to be considered: The algorithm as described above only displays planes which lie in front of the spectator. This is sufficient for desktop or HMD solutions. However, in the CAVE, the display covers up to six (rather than one) sides around the user. Consequently not only one array of planes should be processed but six ones, thus creating an array of nested semitransparent cubes, and increasing the workload sixfold.

For performance reasons, the cubes version has been tested but is not incorporated into the demo-version. As even the CAVE the user mostly focuses on the front side of the visualization, this is less of a problem as originally imagined.

In a still evolving prototype, we also tested the possibilities of a realistic depiction of fire and smoke, with enhancing features such as flames or smoke plumes. In a prototype using the hardware shader extensions of the latest graphics boards, we checked out the limits of 3D-texturing (currently working on windows based machine).

Shaders are programs which are executed in dedicated graphics hardware which define how the sampling coordinates and sampling results of one or more textures are combined to produce the final pixel color.

The features depicted with the volume-rendering algorithm can not be more detailed than the underlying computational grid, providing the data source for all visualizations. That the “fire” does not move between to time-steps of the simulation is even more undesirable. On the other hand, the grid resolution can not be increased arbitrarily. To model the appearance and the dynamics of the flames additional data has to be included. In our case, a noisy fractal distortion field is applied to the sampling grid prior to sampling. Noise usually generates a number of virtually unpredictable features, and the fractal guarantees that these features appear in different magnitudes of size, which gives an realistic impression as the flames are subject to turbulent flow. To avoid falsification, which makes the original data unusable, the distortion must be controlled by a factor which guarantees, that the distance between the original and the distorted sampling stays within the given bounds.

The visualizations include all features, which are demanded by the end-users. The development of tunnel fires can be analyzed and assessed and hazardous regions can be easily identified. The visualization of fire and smoke helps assessing the visibility conditions for both the fire-fighting personnel and the people enclosed in the tunnel and thus helps designing and validating the evacuation plans.

3.4 CD functionality evaluation

There were multiple objectives and specifications defined for the VIRTUAL FIRES project that are covered in the corresponding deliverables:

- Development and implementation of a flow solver based on the Lattice Boltzmann Method capable of simulating turbulent buoyant fluid flow problems

  D 5.4 – Software User Guide V1.0

- Integration of concurrent user interaction on the flow solver (change ventilation, ...)

  D 5.5 – Software User Guide V2.0

- Implementation and testing of combustion models suitable to simulate tunnel fires

  Demonstrated in a latter section of this report and in the final report

- Parallelisation of the solver in order to achieve real time simulation capabilities

  D 5.3 – Parallelisation efficiency guide

Compliance with deliverable *D 2.4 – Specification of planned system capabilities*

D 2.2 – 2.1.2 User expectations

CeTu expects the VIRTUAL FIRES simulator to be capable of two computational levels:

- Static simulation where all boundary conditons are predefined

  This requirement can be considered as a trivial minimum requirement to every simulation software and was already realised in milestone M 5.2 – Software release V1.0

- Dynamic simulaton where the boundary conditions can be changed interactively during the calculations

  In version 2.0 of the Virtual Fires simulation software it is possible to change the state of the boundary conditions (ventilations system, ...) interactively during the calculation. This was already

demonstrated at the review meeting at KTH, Stockholm on March, 31st and April, 1st 2004. The possible user interactions are described in D 5.5 – Software User Guide V2.0.

The requirements of Euve were the following:

- Simulation of the temperature field and the concentration of toxic gases

  The Virtual Fires simulation software delivers air velocity, pressure and temperature as well as a concentration of smoke or toxic gas.

- Moving objects (fire fighters, cars, ...) should be considered in the calculations

  This requirement was rated lowest in the priority list and is therefore not implemented.

FDDo stated that the following points are obligatory to define realistic scenarios:

- Possibility to define variable fire loads

  Different fire load curves may be described to used for the Virtual Fires simulation software. Details are given in D 5.5 – Software User Guide V2.0.

- Option to define different origins of the fire

  The origin of the fire can be placed arbitrarily within the computational domain. Furthermore multiple "potential" fire regions, which are activated if a certain ignition temperature is exceeded, may be defined.

- Simulation of the impacts of different types of fire fighting equipment

  In the latest version of the Virtual Fires simulator the impact of the fixed ventilation system, fans and water nozzles on the spread of fire and toxic gases can be studied. Currently the location of these fire fighting devices must be defined at the start of the simulation. It is not possible to move them during a running calculation.

Alpetunnel expressed their interest to simulate long tunnels (in the order of several kilometers) .

A first step to handle the real time simulation of domains of large extend is the introduction of a 1D/3D coupling procedure. Only the in the vicinity of the fire are the full 3D solution is obtained, whereas in the other parts of the computational domain one-dimensional transport equations are solved. Currently it is possible to extend and reduce the extension of the 3D area automatically during a running calculation.

3.4.1 Evaluation of numerical technique

Tunnel Fire

Experiment

A model tunnel fire experiment in a small scale tunnel was simulated by Xue et al. [1]. Fig. 1 shows a sketch of the experimental set up. The main test section is 6 [m] long with a rectangular cross section of 0.3 [m] height and 0.9 [m] width. Unfortunately the material of which the walls are made of is not exactly known, but a major section of the tunnel consists of perspex, whereas in the vicinity of the fire source silica glass and aluminium are used. The fire source is located 1.5 [m] from the inlet of the test section at the center of the tunnel floor. The burner area is 0.18 [m] x 0.15 [m]. An adjustable fan is used to induce a longitudinal velocity. The heat release rate was 3.15 [kW] and a longitudinal flow velocity of 0.13 [m/s] is

present. Temperatures are reported for 3 cross sections, which are located 0.9 [m] upstream, 3.3 [m] and 5.1 [m] downstream, respectively.

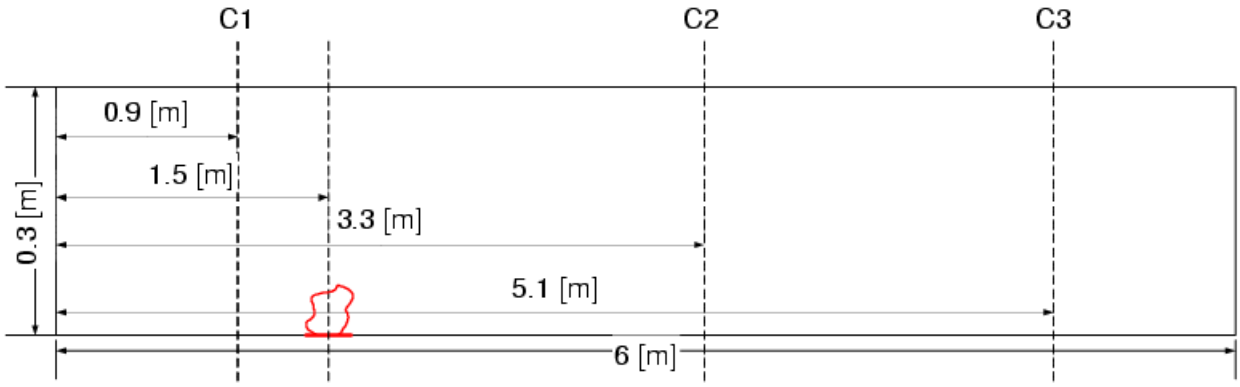

Fig. 1. Computational Model

The computational grid used in the simulation consists of 300 x 45 x 15 cells. A Smagorinsky constant of 0.14 is used and the turbulent Prandtl number is 0.2. The longitudinal ventilation is accomplished by applying a velocity boundary condition at the left boundary of the computational domain, whereas a constant pressure boundary condition is used on the other side. All walls are assumed to be adiabatic.

Results

At station C1 (fig. 2), which is the one nearest to the fire source, all models including the VHS model used by Xue et al. [1] perform very poor. In the experiment the temperature is continuosly increasing, whereas all simulations predict a nearly constant temperature above a height of 0.15 [m]. The temperature obtained by the LBE combustion models is above the reference solution. This can be explained by the assumption of adiabatic walls in the LBE calculations, whereas Xue et al. [1] applied conducting wall boundary conditions.

The experimental trend is even not very well captured at station C2 (fig. 3). Again the LBE combustion models overpredict the temperature compared to the reference solution, but in most regions the difference is small. The temperature just below the ceiling is somewhat higher in the experiment than it is obtained by the LBE mixture fraction model.

At measuring station C3 (fig. 4) furthermost from the fire the LBE VHS model corresponds quite well to the experimental data up to a height of 0.2 [m]. Also the LBE mixture fraction model agrees with the temperature from the experiment up to 0.1 [m] height and overpredicts the temperature beyond. In the experiment the temperature below the ceiling is decreasing. This is not reproduced in the LBE simulations due to the adiabatic walls. In general both LBE combustion model perform quite well some distance away from the fire. This is not surprising as the error induced by using the Boussinesq approximation is largest in near fire regions where temperature differences are considerable.

It is interesting to note that the simulations done by Xue et al. [1] which do not use the Boussinesq approximation do not yield much better results compared to the LBE calculations at the stations near the fire source. This indicates that the complex interaction between air flow and fire is not modelled adequately.

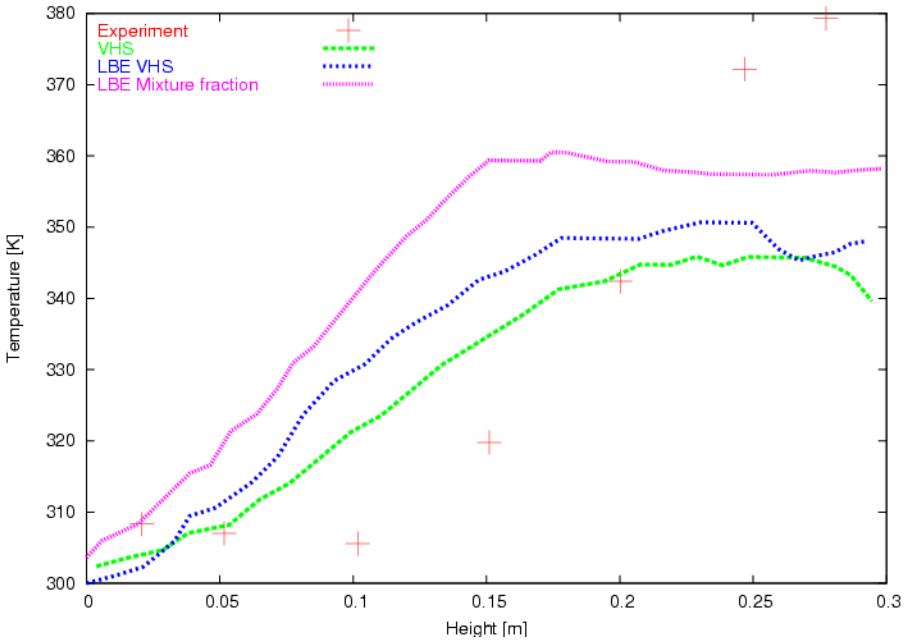

Fig. 2. Tunnel fire: Temperature distribution at station C1 (upstream)

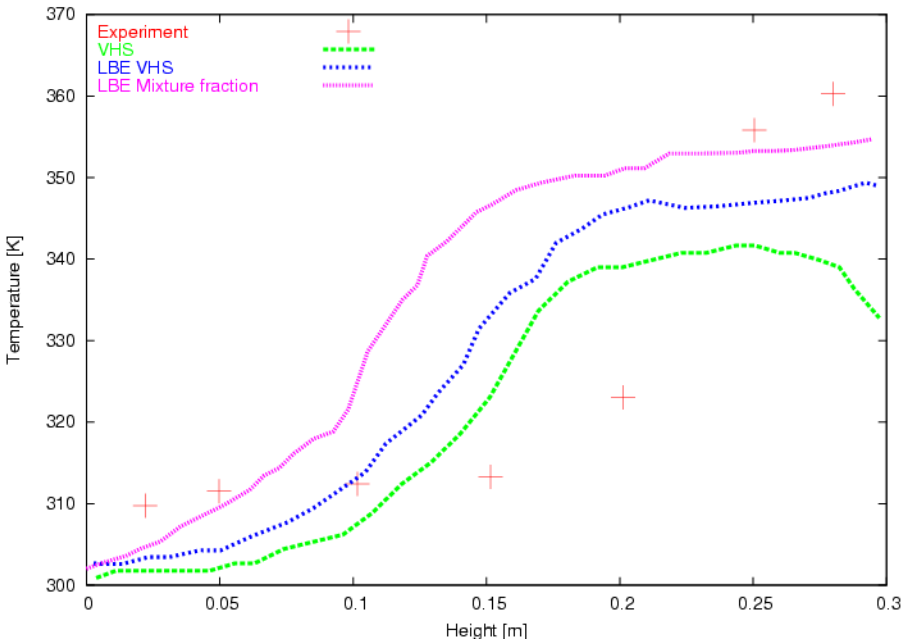

Fig. 3. Tunnel fire: Temperature distribution at station C2 (downstream)

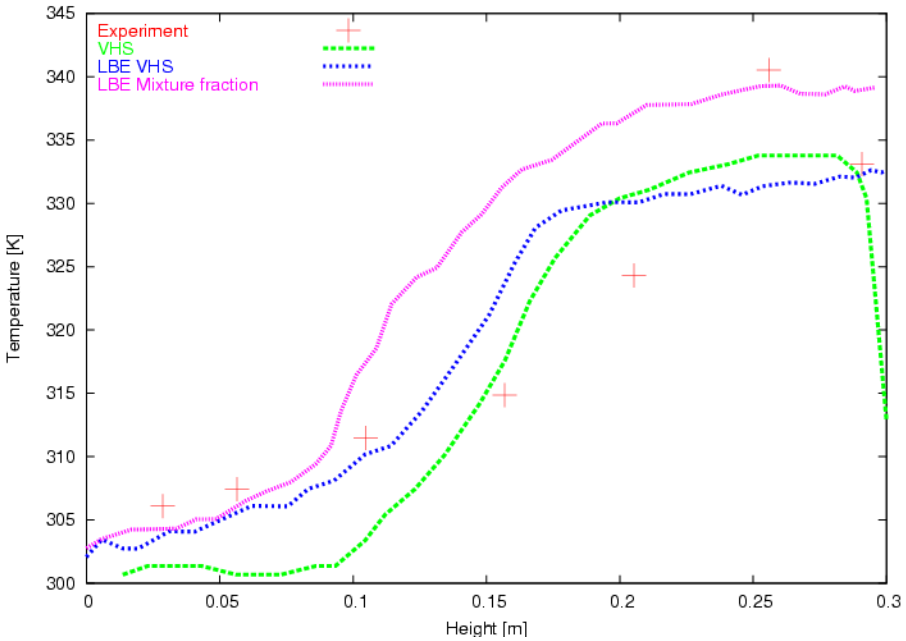

Fig. 4. Tunnel fire: Temperature distribution at station C3 (downstream)

Discussion

A few concluding comments on the performance of the tested LBE combustion models have to be made:

The use of the Boussinesq approximation for modelling buoyancy effects is doubtful especally in high temperature regions and for large temperature gradients.

Outside the near fire region both the VHS as well as the mixture fraction model give resonable results compared to reference simulations and experimental data [1].

As adiabatic walls are assumed especially the LBE mixture fraction model tends to overpredict temperatures. Nevertheless adiabatic boundaries are used as conducting walls increase the computational effort and therefore contradict the requirements of real time simulation.

Differences in the results for both LBE combustion models are a consequence of the different representations of the fire.

The use of a constant value for the Smagorinsky coefficient may be considered as a weak point. Nevertheless if the Smagorinsky coefficient is in a suitable range (C_s around 0.13) resonable results are obtained.

The test case highlighted some weak points of the implemented models. If one compares the performance of the LBE combustion models to the solutions obtained by a commercial code [1] the results are satisfactory. In view of the many additional unknown factor in real fire hazards the limitations of the LBE combustion models can be accepted. Nevertheless there is much space for future improvements.

3.4.2 Evaluation of the parallel implementation

In order to show the ability of the developed software to simulate combustion problems in real time the performance figures obtained on the Linux cluster Lucidor at PDC/KTH, Stockholm [2], are presented in the following section.

Parallel Performance of the Linux cluster Lucidor

The following performance data (Tab. 1 & 2, fig. 5) were obtained by measuring the calculation time of the parallel LBE code in a *"normal use mode"*, i. e. the standard I/O, which is one entire data set per second real time and some monitoring, is included. The hydrodynamic variables (i.e. pressure and velocity) and two scalar quantities (which can be for example mixture fraction and temperature or mixture fraction and some marker for an extinguishing agent) were calculated. The initial set up and the domain decomposition is not considered.

The computational domain consists of a simple straight channel with square cross section. The simulations were carried out for two typical domain sizes used for the Virtual Fires simulator. The smaller one consists of 75,000 cells (Testcase A), which is the target domain size for the real time compuations, the other one consisting of 240,000 cells (Testcase B). Additional one test case made up of 2,000,000 cells is considered (Testcase C).

For all configurations five simulations calculating 5000 time steps, which corresponds to 100 [s] real time, were carried out and the necessary wall clock time were averaged afterwards. Below the computation time needed for 1 [s] real time is given.

MPICH [3] is a freely available implementation of the MPI standard and was used as communication library on the HP Itanium cluster. The optimisation level used for the 64 bit Intel Fortran compiler `efc` was `-O2`.

	Number of processors				
Testcase	1	2	4	6	8
A	21.11[s]	10.05[s]	4.91[s]	3.25[s]	2.27[s]
B	80.69[s]	40.34[s]	17.93[s]	12.04[s]	9.07[s]
C	683.47[s]	310.67[s]	151.88[s]	103.56[s]	80.41[s]

	Number of processors				
Testcase	12	16	24	32	64
A	1.49[s]	1.28[s]	1.05[s]	1.07[s]	1.13[s]
B	5.93[s]	4.98[s]	3.43[s]	2.68[s]	2.06[s]
C	52.57[s]	41.68[s]	28.72[s]	21.56[s]	11.93[s]

Tab. 1. Lucidor - Computation time

	Number of processors									
Test case	1	2	4	6	8	12	16	24	32	64
A	1.0	2.1	4.3	6.5	9.3	14.2	16.5	20.1	19.8	18.7
B	1.0	2.0	4.5	6.7	8.9	13.6	16.2	23.5	30.1	39.2
C	1.0	2.2	4.5	6.6	8.5	13.0	16.4	23.8	31.7	57.3

Tab. 2. Lucidor - Speed Up

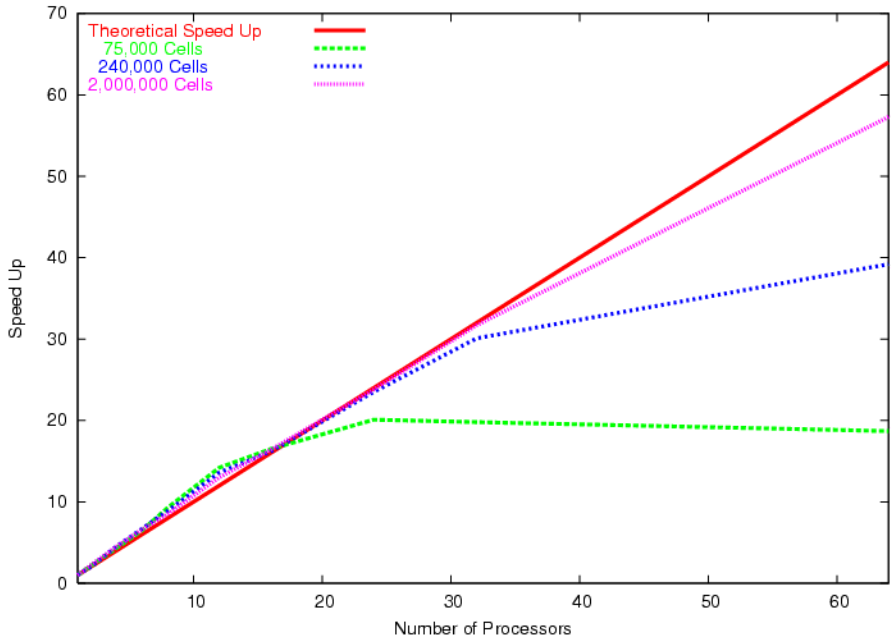

Fig. 5. Lucidor: Parallel Speed Up

The parallel speed up for Testcase A is better than the theoretical value for up to 16 processors, but breaks down dramatically for higher numbers of processors. This is not astonishing if one takes into account that the number of cells for each processor is only about 1200 if 64 processors are used. The one way data rate between the processors is about 1.1 [GBit/s], which is about 55 [%] of the theoretical value of the Myrinet connection [4]. Additionally the network latency sums up to 2 [ms], not to forget one collective data movement per second where all processors send their data to the root processor for I/O.

For higher numbers of processors the performance becomes better if larger domains are used. For Testcase C consisting of 2,000,000 cells the speed up is only about 10 [%] below the theoretical value for 64 processors.

The performance figures for Testcase A consisting of 75,000 cells show that a real time simulation is possible using about 20 processors. For this configuration one second real time event takes roughly one second CPU time. Judging this figures one must keep in mind that the I/O is included in the measured CPU time and two scalar species are considered.

Even for Testcase B consisting of 240,000 cells a real time simulation seems to be feasible in the near future as for 64 processors only about 2 seconds CPU time are required for one second real time.

Discussion

To judge the above results in view of real time simulation a few things have to be considered:

- Real time simulation in connection with real time visualisation only make sense if a suitable number of data sets can be displayed in a very short time. Ideally this would be about 25 data sets per second. In reality one data set which can be transfered to a visualisation system and displayed is assumed to be sufficient. In practice it turned out that at the moment one data set per second is the maximum which can be transfered from the computing machine to the visualisation device via a Gigabit connection.

- From physics and numerics one can conclude that the maximum timestep for stable simulations is in the range of 1/50 [s]. Therefore 50 timesteps per second real time must be calculated as a minimum.

- The number of computational cells which can be calculated in real time is rather small at present time (about 60,000 to 75,000 cells), but in view of the obtained performance data real time simulations consisting of 250,000 cells are feasible in the very next future.

- One can summarize that it is possible to successfully perform real time simulations using MPICE.

Bibliography

[1] Xue, H., Ho, J. C., Cheng, Y. M., Comparison of Different Combustion Models in Enclosure Fire Simulation, Fire Safety Journal 36, p 37 - 54 (2001).

[2] http://www.pdc.kth.se

[3] http://www-unix.mcs.anl.gov/mpi/mpich/

[4] http://www.myri.com

3.5 KTH functionality evaluation

3.5.1 Use scenarios functionality

Scenario 1

Target group: planners, can be expected to be familiar with visualisation software. Most likely do not have access to VR equipment.

-Import of geometry dataset

Datasets can be imported automatically in the visualization system and manually in the CFD solver.

-Choice of camera position

The viewer's position is defined by the normal navigation methods in the VR environment and by mouse interaction in the PC version.

-Turning on and off display of smoke (Sign visibility should be implicit through the smoke display.), gas, gas velocity, temperature, ventilation settings.

All of this functionality is supported in the User Interface.

-Adjusting iso-surface levels

All of this functionality is supported in the User Interface.

-Stepping back and forth through time

All of this functionality is supported in the User Interface.

-Placement of fire; placement of cars, lorries, trains in tunnel; setting properties of vehicles (loads of lorries, etc)

All of the above is done manually in the initialization of the CFD simulation.

-Adjusting the ventilation settings during the course of a simulation; setting weather conditions at tunnel openings (including ventilation shafts!); stopping the simulation, changing parameters, restarting.

All of the above is done with the User Interface

Scenario 2

Target group: fire fighters, presumed familiar with visualisation equipment.

Sprinklers and nozzles at fixed positions are supported.

Movable fans are not supported in the current version. It is however possible to start and stop fans placed in advance.

Scenario 3

Target group: fire fighters, need not be familiar with computer equipment. In addition to fire fighters, tunnel operating staff may also be involved to the extent that they can control fixed equipment, such as ventilation, doors, etc.

Functionality:

-With the HMD solution it would in principle be possible to wear a protective suite etc.

-Import a geometry dataset

Supported

-Record a training session and replay it from different viewpoints

All data is recorded in the database and can be freely replayed.

-Allow a trainer to place vehicles and other obstacles, and place fires.

This can be done manually before the session starts.

-Fires can allowed to develop from stereotyped models, or from actual simulations, if these can be run at sufficient speed.

Actual simulations are used.

-Break a session, change parameters and restart.

Supported by the current user interface.

-Detect if firefighters are subject to dangerous temperatures, smoke or gases.

The temperature can be viewed for example by iso surfaces.

3.5.2 Data input format and interfaces

Geometric data to the visualization can be any format Performer supports. For more formats external conversion programs exists like PolyTrans (*http://www.okino.com/conv/conv.htm*).

The system uses the most advanced interfaces like joystick and tracked glasses and a tracked PDA.

Suggested interface

Case 1: The trainees sit by ordinary workstations. A maximally simple interface should be used where the trainees use a point-and-click interface to navigate their avatars, choose tools and apply the tools to the environment (e g, indicate where to direct a hose).

Case 2:

This case has been considered beyond the scope of the current project. For multiple persons training, a CAVE environment is non-optimal, as it would be unlikely to afford sufficient physical separation of trainees. An HMD interface can be extended to an arbitrary number of participants, given constraints of inter process communication delays and physical space for training. On the other hand an HMD may conflict with the wearing of smoke-helmets - a market study to determine whether an HMD with suitable profile exists should be undertaken.

User interface

While the software is intended to be used on multiple display systems, CAVEs, HMDs and standard monitors, it seems unlikely that the identical interface can be used in all instances. While several of the relevant visualisation packages allow similar types of button-and-menu interfaces to be used both on the desktop and in an immersive environment, the latter alternatives are often sub-optimal, especially in the case of a HMD-based solution.

For the desktop version, standard graphical user interfaces will work well, but would be enhanced by support for stereo goggles and/or SpaceMouse or similar device, as these will simplify the interpretation and manipulation of data.

For an immersive interface the ideal would be a speech- and gesture-based interface, which allows generating visualisations by pointing at positions and speaking operations, rather than requiring floating menues and buttons for operation. A project currently underway at KTH aims to make it possible to use a PDA for interaction in a CAVE environment, which would then make it possible to have most of the benefits of a 2D interface (though limited to a much smaller screen) whilst still being able to work in an immersive space. In particular this will make it easy to set iso-surface levels, fire load settings and other such items which require the input of numerical values.

This was the implemented solution.

The models of the tunnel contain moving parts (doors, vents, etc) that affect the progress of a simulation and they must therefore be movable by the user. The data format must somehow indicate if objects are

movable, as well as the (at least the geometric) constraints for this motion, and the user interface must be able to handle this information. At the least these objects can be defined at their extreme positions (fully open, fully closed), and the interface be able to select between these alternatives. Which position is selected must then be reported back to the simulation, possibly along with parameters indicating what the effect of the one or other setting is.

Moving of objects is possible from the user interface perspective. It was however not possible in his project to let moving objects affect the CFD simulation.

Likewise, it must be possible to set weather parameters for all openings of a tunnel -for multiple vent openings it must be possible to just apply a standard set. Conceivably the metaphor should be a fill-in form with all relevant parameters of which copies can be dropped by the openings.

Implemented in the user interface.

Placement of vehicles and fire sources in the tunnel must be interactive. One can equip the users with a palette of vehicles that can be inserted into the environment. The contents of this palette should preferably be stored in the database described before, so that they are properly integrated with the computational modules and can be accessed by any interface developed. The description of the objects should then also contain iconic representations of the objects, as well as 3D models that can be inserted into the environment. It would be useful to be able to create new items to be stored in the database, perhaps based on already existing objects, but this can be left for later. The actual placement should use collision detection so that vehicles do not penetrate each other or the walls of the tunnel. In a desktop interface this would also help in placing objects in 3D space, as they could be slid along the floor or walls of the tunnel. However, the extremely high polygon count of the tunnel models will then require a highly optimised collision detection method which can quickly select a likely subset of polygons to test for collision. In an immersive interface on the other hand, collision detection need not necessarily operate in real time, but can instead be invoked by an explicit command or run in a lower-priority thread.

See above.

Data have to be recorded over time so that the progress of a given session can be replayed at a later time without having to redo the computation (particularly relevant for sites that do not have supercomputer resources, but still need to see the results of simulations). There should be a convenient way to indicate and annotate breakpoints at any point in a time flow and allow the user to explore alternatives, e g insert a fan at a given time, follow the development from that time on, go back and run the simulation from the same point in time but without the fan and then be able to compare the results of the two cases. It seems reasonable to use a tree-branching display for the timeline. On a desktop display this is trivial to render and interact with. In an immersive display there must be some way of separating the time display from the rest of the environment; possibly one could adjust the hidden-surface removal algorithm in such a way that certain objects, such as interface widgets, will always be rendered on top and not be obscured by the data displays. (How this will affect depth perception remains to be seen, but empirically it tends to be quite robust to similar anomalies in depth, such as those caused by a user's body parts obscuring virtual objects that actually are visually closer.) A timeline can profitably be rendered in three dimensions rather than as a flat object, allowing rotation of branches into the third dimension - which *may* help alleviate clutter. "Gravity"-based interaction, where active regions of space will attract the interaction point, will hopefully simplify for users to access the intended parts of the timeline. The aforementioned PDA-based interface will allow the writing of annotations, but an HMD user is in practice required to have a non-immersed assistant who can type for her, unless we use audio annotations, but these are difficult to overview at a glance (of course, in an HMD text cannot in general be read anyway). In addition to written and/or spoken annotations, small thumbnails of the visualisation at the given points of time may help searching for a specific setting of parameters.

Simple branching is implemented in the user interface.

User interfaces

Several user interfaces will be needed in the final product. The real-time simulation and visualization needs a GUI which can be displayed in the CAVE and which has direct influence on the underlying simulation. Furthermore an interface for the simulation itself is needed (at least for the prototypes) to put in the values, scenes and template objects as specified in 2.3.2. For example the users should be able to put out fans, modify smoke distribution rates, velocities, temperatures, smoke concentrations etc. Furthermore he should be enabled to change ventilation and adjust the parameters of the fixed fire fighting measurements. In the end of the project it would be nice to have the user being able to modify / place mobile fire fighting measurements in the virtual scene and modify them as well.

It is possible to adjust the parameters of fixed equipment or equipment placed in the start of the simulation.

Additionally a user interface for the visualization itself is needed (turn on/off several visualization methods and adjust their parameters) – see 2.3.6.

Individual visualization methods can be turned on or off and all of the parameters can be changed with the user interface.

3.6 SiTu functionality evaluation

3.6.1 Functional evaluation against the D2.2+D2.4 [1]

Quote from [1] 2.1.1 p7:

The more interesting applications of the simulator could be:

- the evaluation of the designed cross passages and other galleries in the underground station,
- the evaluation of the visibility and efficiency of emergency signs,
- the evaluation of the designed sprinkler system,
- the evaluation of the different systems for smoke extraction : one extraction point with 200m3/s, five extraction points with 40m3/s.

Result:

- The evaluation of cross passages and galleries is possible according to the parameters of smoke and temperature distribution and air velocities.
- The evaluation of the visibility and efficiency of emergency signs is limited due to the fact that there is no physical correct lightning model in the simulation. But an estimation of the degradation of visibility due to the smoke density is possible.
- The evaluation of sprinkler systems is possible.
- The evaluation of smoke extraction systems is possible.

Quote from [1] 2.1.1 p7:

Data input :

The geometrical data will be imported from AutoCAD files. The other data will be : fire load, rate and speed of air flow in the railway tunnels.

Result:

- The import of AutoCAD-files directly into the simulator is not possible because it is not directly supported by the visualization system. This data has to be converted to the OBJ-format in before.
- Definition of the fire load, rate and speed of the airflow is possible.

Quote from [1] 2.1.2 p11:

1. static simulation where situation = f(t) for a given geometry

2. dynamic simulation where situation = f(t, men action, material behaviour)

2.1: scenario to be previously defined

2.2: acting any time on boundary conditions

Result

- Static simulation is possible.
- Dynamic simulation is possible according to the specifications 2.1 and 2.2

Quote from [1] 2.1.3 p12:

Data input: the definition of the problem to be solved should be the most straight forward from the existing electronic representation. In the case of EUVE, geometrical data could be imported from our mainly used CAD program, CATIA. Perhaps, if data from civil engineering companies in Spain are extended to other countries, AutoCAD could be another possibility. For system use, most advanced interfaces will be used, and for instance a joystick should be used for movements and a tracking system in the HMD or in the glasses to know head position.

Result:

- All CAD data has to be converted to the OBJ-format at first. This is done by software outside the VFS.
- For navigation in 3D space a Spacemouse can be used additionally to the standard PC mouse in the PC environment and the wand is used in the CAVE environment.
- 3DOF tracking of the users movement of the HMD is possible.

Quote from [1] 2.1.4.1 p13:

Therefore it is required to define realistic scenarios (fig. 1), including

- realistic tunnel geometry incl. installations (geometry and textures)
- variable fire loads incl. hazardous goods (which are normally linked to vehicles in the tunnel system)

- the option to define different origins of fire
- and to simulate the impacts of different types of fire fighting resources

Result

- Realistic models of the tunnel have to be generated outside the simulator and are finally imported into to the database
- It is possible to define different fire loads and origins of fire
- It is possible to simulate the impact of different resources by changing their parameters and rerunning the cfd-calculation

Quote from [1] 2.1.4.1 p13:

A typical process of the simulation has the following steps:

For an existing tunnel with fixed geometry the fire loads and the origin of fire are defined. Based on this the simulation shows the spreading of smoke, fire, gas and heat distribution. The user then can decide to activate fixed fire fighting resources or can define missions for the fire brigade. The impact and effects of these measures modify the further simulation.

Result

- All of this is possible within the Gui in the simulator.

Quote from [1] 2.1.4.2 p13:

Structure of simulation

The object of simulation (e. g. a street tunnel or subway-tunnel) is described by the building itself, installations e.g. traffic signs, built in equipment, and the vehicles. These objects are normally fixed during the simulation, but affected by the fire, and influence the fire simulation by their fire load. So the Fire Area can be defined as a dynamic subset of the buildings, installations and vehicles.

Result

- This is implemented by the definition of the scenarios in the simulator.

Quote from [1] 2.1.4.1 p14:

Concerning the fire fighting resources there are two different types:

- The fixed fire fighting resources can be activated by the user and have an impact on the fire, they may as well be affected by the fire. Usually they have no or little delay time. Examples are sprinkler-systems, artificial ventilation, robot systems.
- Normally the mobile fire fighting equipment is operated by fire fighters. To activate mobile fire fighting resources the user has to define the mission and the attack-point, where the tasks shall be carried out. From this, a delay time may result. This delay time can be predefined and it depends on the set-up time for the fire fighters´ personal protection equipment and the size and movability of the equipment. An example is given later on. Normally mobile resources are not affected by fire.

Result

- All fixed fire fighting resources like nozzles, sprinklers and ventilation systems can be simulated.
- Delay times in activating equipments are not directly supported, they must be considered by adding the delay time to the activation time.

Quote from [1] 2.1.4.2 p14:

These simulations should include the environmental conditions outside the tunnel such as

- temperature,
- barometric pressure and
- wind direction.

These conditions may change during daytime and by this may influence the area included into the simulation.

Result

- Simulation with this boundary conditions is possible.
- Change of these BC over time is possible.

Quote from [1] 2.1.4.3 p16:

different types of stations

- single level stations with only one platform and two connected tubes and stairs and elevators
- three-level crossing stations with up to three platforms on one level, multiple connections and installations

different types of tubes

- circular, oval or rectangular cross-sections
- additional sidings, where trains can be parked
- additional emergency exits connected to the tubes

Result

- The simulator can handle all the given different types of stations and tubes.
- Complex and large domains only lead to an increased amount of required computation time.

Quote from [1] 2.1.4.4 p18:

As mentioned before mobile fire fighting resources usually are operated by fire fighters. So the user has to define so called missions, which describe

- the point of attack,
- the measures,
- the personal protection and
- additional equipment.

Result

- Mission based simulation is possible.
- Missions are defined by the timely sequence of parameter changes of the assigned equipment.

Quote from [1] 2.1.4.4 p18:

From these data the impacts and duration of a mission can be calculated during a simulation, provided that the toughness of the personnel and the fire fighting equipment are described by adequate parameters

Result

- Only the physical parameters of the firefighting equipment are taken into account.
- Influence of the environmental conditions on the firefighters is not considered directly, but can be estimated by evaluating the temperature distribution.

Quote from [1] 2.1.4.5 p20

- With a look at the time consuming operations and processes a fully real-time simulation of a tunnel fire is questionable. As seen in actual accidents the fire fighting can last for several hours or even days.

Result

- Simulation of such long events is possible but requires adequate storage space.

Quote from [1] 2.1.4.5 p21:

From this the implementation of fast motion, break points and slow motion functionalities are recommended. By this the user could define scenarios and missions to find out critical situations via fast motion. These situations and places can be inspected in detail with the help of slow motion. Additionally the simulation can be rewound to breakpoints, e.g. starting points of missions, to redefine a mission. From this it is necessary to include multiple mission into the simulation.

Result

- All these features are implemented in the user interface.

Quote from [1] 2.1.4.5 p22:

For the detailed evaluation the user will define different scenarios, equipment and missions to assess the risks of specific objects. This will be done by one person or small teams. The focus is on the scientific visualisation (colour mapping,...) of heat, smoke and gas distribution and a symbolic visualisation of

missions. Additionally the already mentioned highly interactive functionality for setting breakpoints, slow motion etc. is required and the system should produce reports of the measure respectively show the state of missions when they are identified interactively. This system for the all-day use should be high-end PC-based with adequate graphical output devices (HMD, Shutter glasses, small projection system). For specific situations (very large objects, high risks, approval processes) CAVE-implementations are useful.

Result

- The system is available in both versions: PC-based and CAVE-based
- All versions support the same functionality, they differ only in their performance.

Quote from [1] 2.1.4.6 p23:

Test Cases

As pointed out before, the simulation of missions and the impacts of fixed fire fighting resources are of great importance for the fire brigades. The Lyon meeting clarified, that simulations like this have not been done until now. From this it is recommended, that at first test cases should cover this problem, and that we turn to the simulation of complex structures later on, when the simulation of the test cases have been successful.

From this first a large tunnel with a simple structure should be the basis. Within this tunnel different fire loads can be placed and missions can be defined.

To evaluate the missions a closer look to an area of about 20m to 40 m should be possible. in the first draft the missions can be defined by approximation of geometric features and physical impacts of fire fighting resources and equipment. The effects of multiple missions could be simulated by scaling and multiple applications of the defined base-missions and resources.

Result

The following testcases were selected and simulated in the simulator:

- A part of the Mt. Blanc tunnel
- A part of the Gleinalm tunnel
- A subwaystation in Dortmund

Quote from [1] 2.2.2.1 p26:

General Overview

From the FIGD point of view, the general layout of the whole System should be designed like the structure in fig. 1.

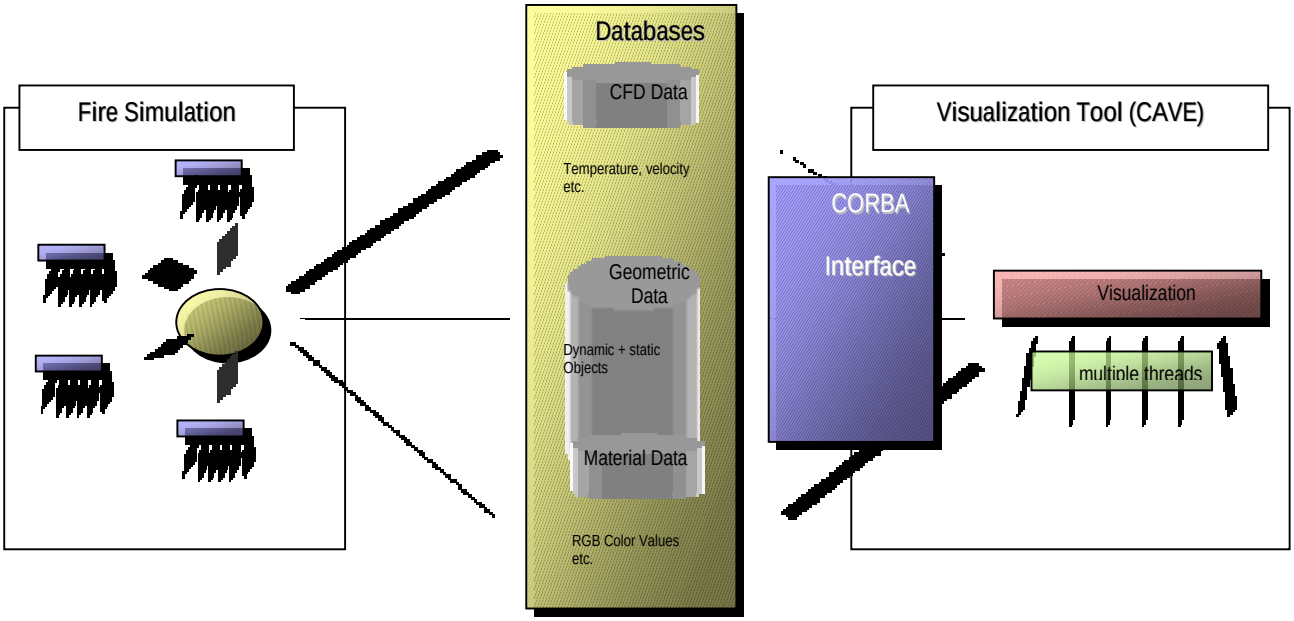

Figure 1: general system layout

Result:

- This architecture had to be changed during the development to the layout given in fig 2. The CORBA interfaces were dropped and the visualization modules were realised as plug-ins to the renderer of the VR-system. This was due to performance reasons.

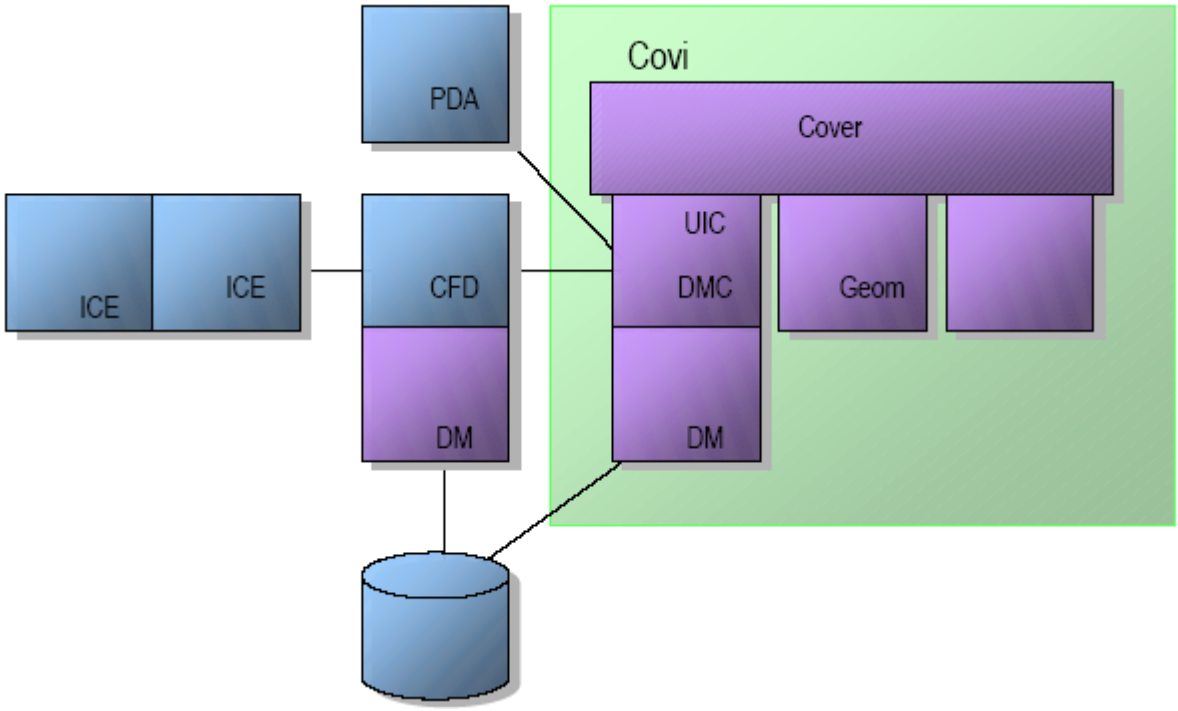

Figure 2: final system layout

Quote from [1] 2.2.2.2 p26

Simulation

The simulation is docked to a database server which holds the necessary information like the geometry (tunnel, cars, etc.) and the material data. The results of the simulation ("CFD data") like temperature or gas and smoke density are stored in another database.

Result:

- This is implemented. Furthermore the architecture of the database allows for transparent extension of the CFD result parameters because of the use of the CGNS format.

Quote from [1] 2.2.2.3 p26:

Data-server + Interface

The data server(s) hold the necessary information for the simulation and the visualization and will be reachable through an abstract interface (implemented in Corba as specified in WP 2.3). Together will modern caching techniques it will reach the required performance and abstraction levels. The abstract (Corba) interface will guarantee the independency of the visualization from the simulation and the data storage (server).

Result:

- Implemented differently. As indicated above, the change in the architecture required a drop of the CORBA interfaces for the different modules.

Quote from [1] 2.2.3.1 p28:

General Overview

SiTu's main part of the development will be the design and implementation of the database for the simulation environment. It will perform the following tasks:

Provide services for storage and retrieval of the required data for the CFD calculation

Compress and filter the resulting CFD data to preserve storage volume and transmission bandwidth

Provide the services for retrieval of the required scene data for the VR system.

Result:

- All services for storage and retrieval of the data required for the CFD calculation have been implemented.

- Compression of the CFD data has been dropped due to its uneconomic computing performance burden compared to the gained bandwidth increase.

- All services for retrieval of the data for the VR environment have been implemented.

Quote from [1] 2.2.3.2 p28:

Scenarios

Real-time simulation

Within this scenario a user of the system cannot only watch the spreading of the fire from its starting point, but is also able to set measures against it and watch their reactions immediately. Due to the large amount of data that is produced during an interactive run of the simulation and its demands for low latency a direct coupling of the CFD calculation and the VR system is preferred. In an active simulation the boundary conditions for the CFD-calculation are changed interactively and are forwarded directly to the CFD-solver. The results of the CFD calculation are transmitted back to the VR System.

A-priori simulation

Here the total timeline of actions is predetermined in advance and stored into the database. In this scenario there is a timely decoupling of the CFD calculation and its visualization in the VR system. The CFD solver performs its calculation based on predefined data. The resulting dataset for all time steps is stored into the database. The VR system fetches the model of the tunnel for an initial display and a part of the CFD-data surrounding the users position and renders it according to the selected visualization mode.

Result:

- Both application scenarios, a realtime simulation and playback of a-priori calculated simulations, have been implemented.

Quote from [1] 2.2.3.3 p29:

Data Flow

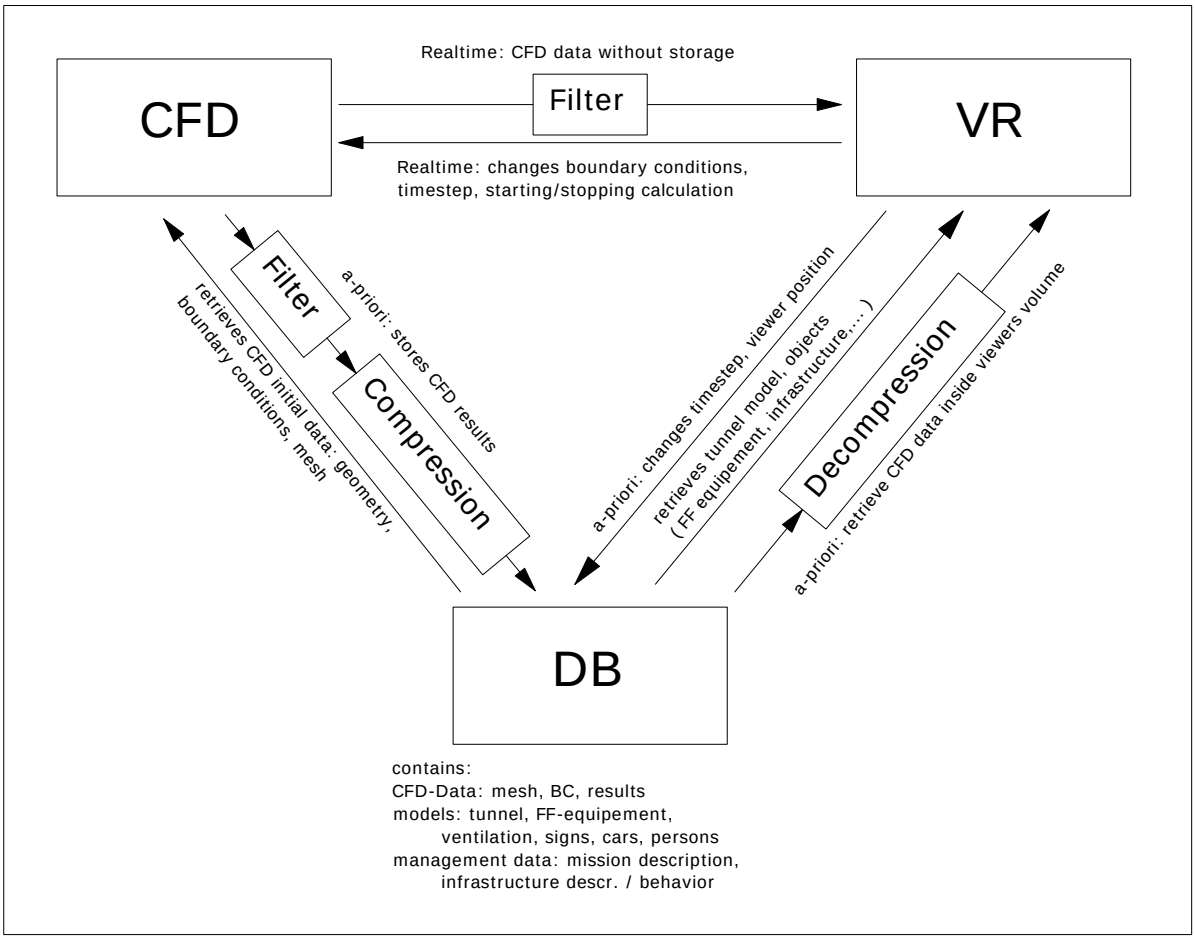

Figure 1.2.3.3: interaction between CFD, VR and DB

Result:

The depicted data flow has been implemented with the exception of the compression of the CFD data.

Quote from [1] 2.2.3.3.2 p29:

Real-time simulation

Initialization

The VR System loads the model from the database and establishes a direct connection to the CFD-calculation. The CFD solver retrieves its initial dataset (computational domain and boundary conditions) from the database. The RT solver will construct its mesh at the beginning of the calculation.

Running the simulation

While in the interactive simulation mode the boundary conditions for the CFD-calculation will be changed by user interaction like activation of ventilation systems or fire fighting equipment. These changes are forwarded directly to the CFD-solver and contribute to the results for the next time step. To reduce the amount of data to be transferred the results of the CFD calculation are filtered before being transmitted back to the VR System. Due to the huge amount of data being generated by the CFD it seems not useful to store these data into the database for later usage.

Result:

- The listed functionality has been implemented with the exception that the solver does not generate its domain at the beginning – this is done beforehand and already available at this point in time.

Quote from [1] 2.2.3.3.3 p30:

A-priori simulation

Precalculation

The boundary conditions for the whole simulation period are predefined and stored into the database. The CFD solver performs its calculation based on this predefined data. The resulting dataset is being filtered, compressed and finally stored into the database.

Inspecting the results in the VR system

The VR system fetches the model of the tunnel for an initial display. A part of the CFD-data belonging to a volume around the current viewing position of the user from the simulation is fetched from the database and rendered.

Result:

- This has been implemented with the exception of storing compressed CFD data.

Quote from [1] 2.2.3.4 p30:

Content

The content of the database will be divided into the following main parts.

CFD-Data

A geometrical and topological description is required for the CFD calculation. From this data the preprocessor of the CFD solver if needed will generate a suitable mesh. The initial boundary conditions, a description of the changes of the BC over time and the results of a-priori calculations are stored within this part of the database.

Models

Data required by the VR system. It incorporates the photo realistic models of the tunnel and its infrastructure, the firefighting equipment, cars and persons.

Management data

Data that is needed for configuring the simulator. This contains the description of fire fighting missions, of the behavior of infrastructure and its physical parameters for the simulation.

Result:

The structure of the database contains all these datasets. For a more detailed description see [3].

Quote from [1] 2.2.3.5 p30:

Interfaces

As decided in WP 2.3 all parts of communication with the database will be provided as Corba interfaces

Result:

- This has been changed to the Cover PlugIn-API as the interface between the visualisation module and the UIC module.
- The communication interfaces between the CFD and the datamanagement are based on the ACE framework.

Quote from [1] 2.2.5.1 p37:

Several prototypes

Several prototypes should be realized to approach the final product in three main steps.

The first prototype should be capable to simulate and display a tunnel with fixed geometry and a fire load in it. The second one should be able to include fixed fire fighting measurements. In the third one the fire fighting measurements should be moveable.

Result

All prototypes have been implemented. As moveable firefighting measures only nozlles are possible, because they do not influence the CFD domain geometry.

Quote from [1] *2.2.5.2* p37:

Description of tunnel & geometry

The tunnel geometry & texture should be stored in a standard 3D format (like *.obj for example). The meta information will be saved in an extra file, which is written in XML language. It describes the “special” objects, which can be placed in the tunnel and have a direct or indirect influence on the fire (fire fighting measurements, burnable objects etc.). The different geometry and texture states (“fresh”, “burned”, etc.) of these “special” objects will be stored, according to the tunnel information, in (several) 3D files (*.obj) for each state. The whole scene information (which tunnel geometry, which meta objects at which position) will be stored in another XML file (“scene” file).

Result:

- All visualisation models are stored in the OBJ-format inside the database.
- Statehandling for the simulation objects is implemented.
- Additional required meta-information is stored as an xml-string attribute in the database tables.

Quote from [1] *2.2.5.5* p38:

Data storage & communication

For a performant visualization a resampling of the simulation output data is needed. At the present point of the development it seems clear that the visualization will make use of regular grids (equidistant or octree for example). Therefore the software needs to decouple the grids of the simulation and of the visualization. The best place for this computation will be the database (~interface). By doing so the simulation will still be able to safe its whole simulation output into the database and on the other hand the visualization will be able to request just the information which is really needed to display the scene as exact as needed.

This way bandwidth and computing power can be saved on the visualization side to guarantee a performant rendering of the result. For this reason the (space) interpolation which is needed for resampling will be done in the database.

Result:

- CFD-simulation and visualisation are completely decoupled, i.e. the visualisation system decides on its required framerate independently from the timely resolution of the CFD datasets.
- Resampling is done on the visualisation side and not in the database.

Literature/Links

[1] WP 2.2+2.4 Project Deliverable: Selection of developer tools, Specification of planned system capabilities (software)

[2] WP 3 Project Deliverable D3.1: Specification of Geometrical Database

3.7 Additional remarks on functionality evaluation

3.7.1 Functional evaluation against the D2.2+D2.4 [1]

Quotation from FDDo

The main applications of the VirtualFires-simulation will be:

- the evaluation and validation of fire fighting measures and tactics inside existing tunnel and subway-systems,
- the risk analysis of planned tunnel and subway-systems,
- the evaluation of fixed fire fighting resources,
- the demonstration of tunnel fires and hazards for training and teaching purposes.

As pointed out in the Graz meeting in November three types of applications can be seen:

- the detailed evaluation of buildings, fixed fire fighting resources, risks and fire
- fighting tactics,
- the demonstration of tunnel fires and the related measures for preparatory training of fire fighters,

- a general information for laymen, e. g. drivers.

Quotation from KTH

Three different use scenarios can be identified for the VIRTUALFIRES system:

1. Planning authorities need to assess the likely outcome of a fire in a planned or existing tunnel. They may need to demonstrate the results of this assessment to laymen.

2. Fire departments need to simulate the effects of taking given actions in a fire situation.

3. Fire fighters and tunnel operators need to practice their behaviour in the case of a fire.

We now attempt to discern what requirements the different scenarios place on the software.

Result

- In general lines, all of the above applications can be covered by the Virtual Fires system. However, end-users demand improvements in the range of equipment to be simulated or in processing results to facilitate tasks such as risk analysis.