Development and testing of BlenderFDS, the open, community-based, user interface for NIST FDS

Gissi, E. [1]; Notaro, F. [1]; Longobardo, G. [1]; Valpreda, F [2]

[1] Corpo nazionale dei Vigili del fuoco. Ministero dell'Interno. 00100. Roma. Italy.

[2] Dipartimento di Architettura e Design. Politecnico di Torino. Castello del Valentino. 10125. Torino. Italy.

ABSTRACT

Computational Fluid Dynamics (CFD) tools have increasingly begun to play an important role in risk assessments for fire safety design. NIST Fire dynamics simulator (FDS) seems the CFD modeling tool of first choice for the world-wide community of fire engineers.

FDS is being developed as a CFD code only, so no user interface is available to the user, and the input data pre-processing phase is completely left to the user responsibility. In real world cases, especially where complex or curved geometries are to be described, the data input phase represents an important cost for the fire engineer.

Back in 2009, in the opinion of the authors, no satisfactory pre-processing tool for this purpose existed. No multi-platform, open source and freely available pre-processing tool existed at all. The lack of an open pre-processor tool motivated the development of BlenderFDS, the open, community-based, user interface for FDS.

This paper describes the design process and development choices of this new tool. Then the results of a recent, through evaluation of the tool are presented: BlenderFDS was employed for a fire safety study on *Castel Thun*, a fascinating medieval castle located at the foot of the Italian Alps.

BlenderFDS allowed for a satisfactory control over the input data, and the generated namelists groups. The graphical user interface for 3D solid modeling and intense data sharing between BlenderFDS entities prevented duplication of efforts and lowered the risk of input data errors.

This study demonstrates the value of a fully open tool-chain for CFD fire safety analysis. BlenderFDS tool is following the evolution of FDS ecosystem, both in terms of new FDS features and in terms of FDS users' community needs. Its open, bottom-up development model seems to be mostly appropriate to withstand such a challenge.

1 INTRODUCTION

Computational Fluid Dynamics (CFD) tools have increasingly begun to play an important role in risk assessments for fire safety design. While CFD-based analyses have largely been limited to research applications in the past, it is expected that these kinds of calculations will be used more and more for engineering applications in fire safety and extends beyond the fire research laboratories into the engineering, fire service and legal communities.

This is driven by the large interest in the possibility of precisely tailoring preventive and protective means to the specificity of the concerned civil and industrial activities, both for accurately check safety levels and reducing the safety investments costs, especially in the light of the current global concern about economic stability and growth.

Further, simplified tools for fire simulation in buildings are losing their appeal to the fire engineering community as they lack the ability to model the physical processes well which is a prerequisite for performing proper consequence modeling. CFD tools have the potential to model the relevant physics and predict the effects of a certain incident.

However, the CFD tools are intended for use only by those competent in the fields of fluid dynamics, thermodynamics, heat transfer, combustion, and fire science, and are intended only to supplement the informed judgment of the qualified user [1]. This peculiarity still represents an important barrier to the adoption of such powerful tools.

NIST Fire dynamics simulator (FDS) is a computational fluid dynamics model of fire-driven fluid flow, mainly developed at the National Institute of Standards and Technology (NIST), a federal agency within the Department of Commerce of the United States.

The software solves numerically a form of the Navier-Stokes equations appropriate for low-speed, thermally-driven flow, with an emphasis on smoke and heat transport from fires. FDS has been publicly released on 2000. Since its first release, continued improvements have been made to the software based largely on its user base feedback.

Currently FDS seems the CFD modeling tool of first choice for the world-wide community of fire engineers. In fact FDS is considered a state-of-the-art tool in the very specialized field of fire simulation. Another important aspect that drives that choice is the fact that FDS source code is in the public domain and is freely available, an invaluable advantage over closed source tools, as this peculiarity allows the advanced user full control over the complex CFD calculations she is performing.

As for the NIST developers explicit choice, FDS is exclusively being developed as a CFD code only, so no user interface is available to the user, except for a command line interaction mode, but limited to the mere processing control.

Thus the input data pre-processing phase is completely left to the user responsibility: all the necessary information to perform an FDS simulation has to be contained in a single text file. The input file is written using a specialized description language, following the syntax of the *namelist groups* from the *Fortran* programming language and a semantic specification peculiar to FDS.

In real world cases, especially where complex or curved geometries are to be described, the data input phase represents an important cost for the fire engineer.

Back in 2009, several third-party pre-processing tools and add-ons to FDS that simplified and streamlined the data input phase were available; in the opinion of the authors, each of them had several limitations, the perceived most notable one was the lack of flexibility.

No open source and freely available pre-processing tool existed at the time.

This is the context where the *Comando provinciale dei Vigili del fuoco di Genova*, a branch of the *Corpo nazionale dei Vigili del Fuoco, Ministero dell'Interno* (Italian national fire and rescue service) fostered the development of a new, open-source, community-based user interface for FDS, in the frame of a wider project whose goal was lowering the entry barriers for fire engineers to CFD fire simulation tools [2].

Several research institutions, as the *DIPRADI – Politecnico di Torino,* the *DII – Università del Salento*, and numerous individuals from the FDS user community contributed resources, ideas and code to what has evolved to be the *BlenderFDS* tool [3].

2 METHODS

2.1 Analysis and design

In 2009 the working group constituted by the *Comando provinciale dei Vigili del fuoco di Genova* produced a draft blueprint of the new FDS pre-processing tool. Following a bottom-up approach, the draft blueprint was informally submitted to the members of the FDS user community for review.

Following that review, an existing 3D modeling platform was selected as the base for further development: the *Blender* platform [4], an open source 3D content creation suite available for all major operating systems, easily extendable through its well supported and documented *Python* application programming interfaces (APIs).

As for good programming practice, the new tool inherited its name from its parents: Blender and FDS.

The revised blueprint was then presented at the *Blender Conference 2009* [5], and received several additional contributions by 3D modeling specialists and Python developers from the Blender community.

2.2 Development

2.2.1 BlenderFDS philosophy

Blender files have a hierarchical structure: each Blender file contains a database of geometric features. This database contains all the \`scenes\`, \`objects\`, \`meshes\`, \`materials\`, \`textures\` etc. that are described in the Blender file and visualized in Blender UI.

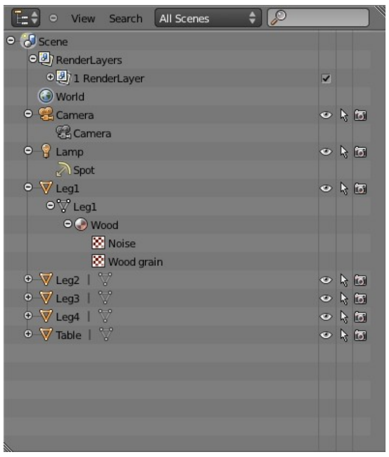


Fig. 1. An example outliner panel shows Blender data organization

A file can contain multiple `scenes`, and each `scene` can contain multiple `objects`. `Objects` are composed by `meshes` and can contain multiple `materials`, which in turn can contain many `textures`. Most datablocks can be shared among other datablocks, re-use is encouraged. For example, the same material can be shared by several objects.

A Blender `scene` is a collection of `objects`. `Object` properties are manipulated in `Blender Object mode`. Each Blender `object` groups:

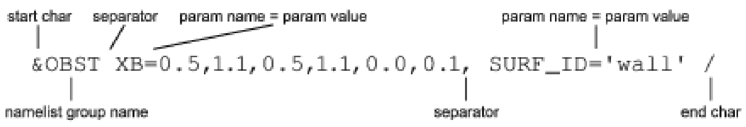
- its `mesh`, the shape of the `object`,
- its `material`, the aspect of its surface (color, transparency...),
- and the `texture` applied to its `material`.

A Blender `mesh` is a collection of faces, edges, and vertices which can be modeled and manipulated in `Blender Edit Mode`. The Blender `mesh` is not to be confused with an FDS `MESH`. In fact, a Blender `mesh` is only a volume or a surface defined by vertices, edges and faces: it is the geometric description of the parent `object`.

A `vertex` is a 3-dimensional coordinate, representing a point in space. An `edge` is a straight, wire-like line representing the boundary of 2 adjacent vertices. A `face` is a planar connection of edges representing a boundary, field, or solid surface. Faces are always flat. Edges are always straight. An edge is formed by the intersection of 2 faces. Vertices are formed by the intersection of 3 or more faces. For example, a cube has 6 faces, 12 edges and 8 vertices.

Every Blender entity can be duplicated and linked to several parents. It is possible to create links between different Blender entities, so that they share some information. For example, if you have a car `mesh`, you can use that car `mesh` for six cars in a parking lot scene. If you modify the original car `mesh`, then all the six cars of the parking lot inherit that modification.

FDS input is a single text file. Input data is specified by namelist groups.



```
&OBST XB=0.5,1.1,0.5,1.1,0.0,0.1, SURF_ID='wall' /
```

Fig. 2. An example FDS namelist group

Namelist groups are used to describe every aspect of the fire simulation: simulation name, duration, computational domain, geometry, boundary conditions, materials, output quantities, control logic...

Many namelist groups extend their action in the computational space to volumes, faces, segments, points, or planes. The simulated-ambient geometric description is entered line by line; the geometric entities are defined in each namelist group by the `XB, XYZ, PBX, PBY, PBZ` parameters.

FDS uses an absolute global reference coordinate system, that conforms to the right hand rule. By default, the z axis is considered the vertical.

An FDS *volume* is always represented by a single right parallelepiped with edges parallel to the axis. Its position and dimensions are described by the coordinates of two opposite vertices: if point A=(x_A, y_A, z_A) and point B=(x_B, y_B, z_B) are the opposite vertices, its coordinates are entered as the following vector (x_A, x_B, y_A, y_B, z_A, z_B) as, for example, in `&OBST XB=0.5,1.5,2.0,3.5,-2.0,0., SURF_ID='wall' /`. The `XB` parameter is used to define a solid obstacle that spans the volume starting at the origin (0.5, 2.0, -2.0) and extending 1 m in the positive x direction, 1.5 m in the positive y direction, and 2 m in the positive z direction.

Same or similar rules apply for FDS *faces*, *segments*, *points*, and *planes*, as clearly explained in FDS official documentation [1] and other sources [2].

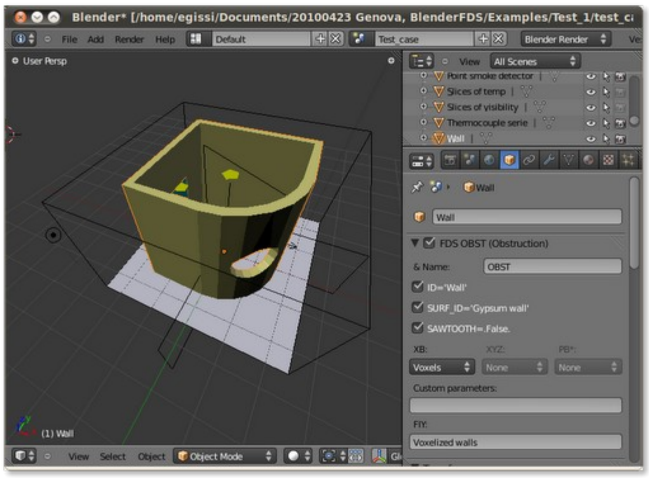
BlenderFDS links each Blender entity to a single corresponding FDS namelist group or to a series of them, as shown in Table 1.

Blender	**FDS**
Each file	One or more FDS input files
Each `scene`	An FDS case described by one FDS input file
Each `object`	One or more namelist groups (`OBST, DEVC, VENT, HOLE, SLCF`...)
Each `material`	One `SURF` namelist group

Tab. 1. Blender to FDS corresponding entities

Each Blender `scene` (as in Figure 3) is exported to an `.fds` text input file. A single Blender file can contain several `scenes`, thus several FDS cases, one for each simulated scenario.

For example, the same parking lot can be duplicated in several `scenes`. The geometrical `meshes` and the thermo-physical parameters of the cars can be shared between all the cars in all the `scenes`; the position of the cars in the single `scenes` can be varied according to the simulated scenario.



Wall

FDS OBST (Obstruction)

& Name: OBST

- [✓] ID='Wall'
- [✓] SURF_ID='Gypsum wall'
- [✓] SAWTOOTH=.False.

XB: Voxels XYZ: None PB*: None

Custom parameters:

FYI: Voxelized walls

Fig. 3. A BlenderFDS `scene` as visualized by the user interface

Each Blender `object` is linked to one or more FDS namelist groups. The exported namelist groups extend their action to the geometry covered by the Blender `object mesh`.

That yellow Blender *Wall* `object` is exported as many `OBST` namelist groups that describe the round and carved geometry of the wall, as shown in Table Errore: sorgente del riferimento non trovata and Figure 4. The Blender `material` linked to the *Wall* `object` is named *Gypsum wall* and rendered in yellow colour. The *Gypsum wall* Blender material is exported as an FDS `SURF` namelist containing all the custom parameters specified by the user.

```
&SURF ID='Gypsum wall', RGB=204,199,56, MATL_ID='Gypsum plaster', THICKNESS=0.03 /
&OBST ID='Wall', XB=-1.575,-1.375,-1.375,-0.475,-0.025,1.975, SURF_ID='Gypsum wall',
    SAWTOOTH=.FALSE., FYI='Voxelized walls' /
&OBST ID='Wall_1', XB=-1.575,-1.375,-1.375,1.375,1.975,3.025, SURF_ID='Gypsum wall',
    SAWTOOTH=.FALSE., FYI='Voxelized walls' /
&OBST ID='Wall_2', XB=-1.575,-1.375,0.475,1.375,-0.025,1.975, SURF_ID='Gypsum wall',
    SAWTOOTH=.FALSE., FYI='Voxelized walls' /
&OBST ID='Wall_3', XB=-1.575,-0.675,-1.575,-1.525,-0.025,3.025, SURF_ID='Gypsum wall',
    SAWTOOTH=.FALSE., FYI='Voxelized walls' /
&OBST ID='Wall_4', XB=-1.575,0.625,-1.525,-1.475,-0.025,3.025, SURF_ID='Gypsum wall',
    SAWTOOTH=.FALSE., FYI='Voxelized walls' /
...
```

Tab. 2. An excerpt of the generated FDS input file

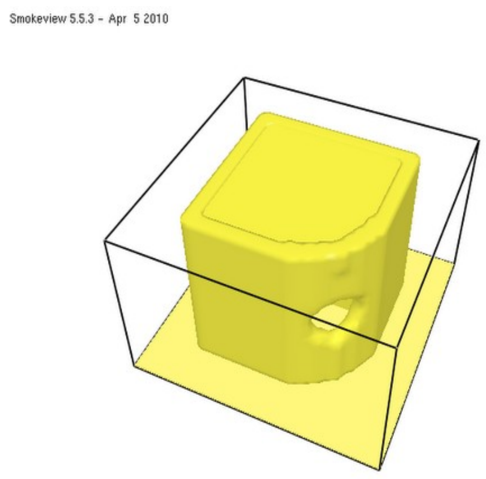


Fig. 4. The exported Blender `object` as visualized in Smokeview

2.2.2 Generating FDS solids from curved geometries

The entire geometry of the FDS model is made up entirely of rectangular solids, each one introduced on a single line in the input file. The efficiency of FDS is due to the simplicity of its numerical mesh. However, there are situations in which certain geometric features do not conform to the rectangular mesh, such as a roof or a dome [1].

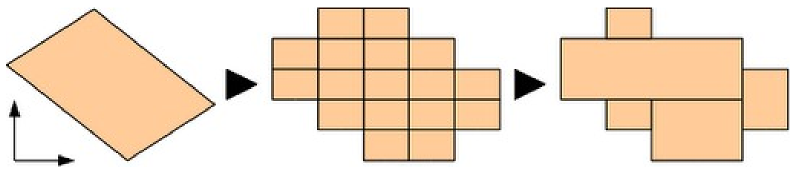


Fig. 5. Processing a geometry through voxelization and boxelization algorithms

BlenderFDS overcomes this notorious FDS limitation by processing complex geometries with a *voxelization* algorithm, as schematized in Figure 5: geometric objects are converted from their continuous geometric representation into a set of voxels that best approximates the continuous object. The voxel size is specified by the user, according to the desired accuracy (Figure 6).

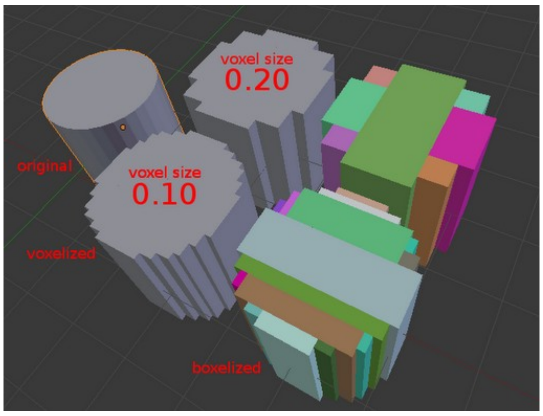


Fig. 6. The exported geometry accuracy depends on the chosen voxel size

The current stable release of BlenderFDS uses a common *marching cube algorithm* [6, 7] implemented in Python by Michael Schardt for Blender 2.48 and released under a GPL license [8]. Due to its inherent slowness, this algorithm is being replaced by a direct employ of the Blender *Remesh modifier* contributed by Nicholas Bishop to Blender 2.62. This modifier uses a new method for contouring a signed grid developed by Tao Ju [9]. The new algorithm is 6-10 times faster than the previous one.

When possible, the voxelized geometry is further treated by BlenderFDS: a *boxelization* algorithm joins the voxels together in larger *boxes* to minimize the number of generated namelist groups in the exported FDS input file. The boxelization algorithm was specifically implemented in Python for the purpose by the authors.

2.3 Implementation

BlenderFDS code was first publicly released on May, 2010, under a GPL license [8]. The code is entirely developed in the *Python* language as a Blender add-on, and it is available on Linux, MS Windows, MacOS X platforms. BlenderFDS development is public and open. User support and documentation is community-based.

BlenderFDS effort is completely independent from FDS development performed at NIST and other organizations. None of these organizations finance or support BlenderFDS.

BlenderFDS tool tries to be as flexible as possible, and is intended for users that already have a basic knowledge on how FDS works, because no data validation is automatically performed and input data correctness is the sole responsibility of the user. Geometry and thermo-physical properties are modeled in a graphical environment, pre-existing 2D and 3D data of buildings can be imported from many CAD file formats.

2.4 Evaluation

During the period of March-June 2012, BlenderFDS received intense testing by application on a complex case study.

Following the request from the Autonomous Province of Trento, the authors performed a fire safety study on *Castel Thun*, a fascinating historic castle located at the foot of the Italian Alps (Val di Non, Trento, Italy).

Built in the 13th century, the castle was the residence of the *Thun* noble family for 800 years. The vast monumental complex, with its turreted palace, well-constructed fortifications and extensive gardens, represents a unique example of a noble residence, entirely furnished and rich in valuable collections. Purchased by the Autonomous Province of Trento in 1992, Castel Thun was opened to the general public in 2010 as one of the venues of the Castello del Buonconsiglio Museum.

Fig. 7. Castel Thun overview

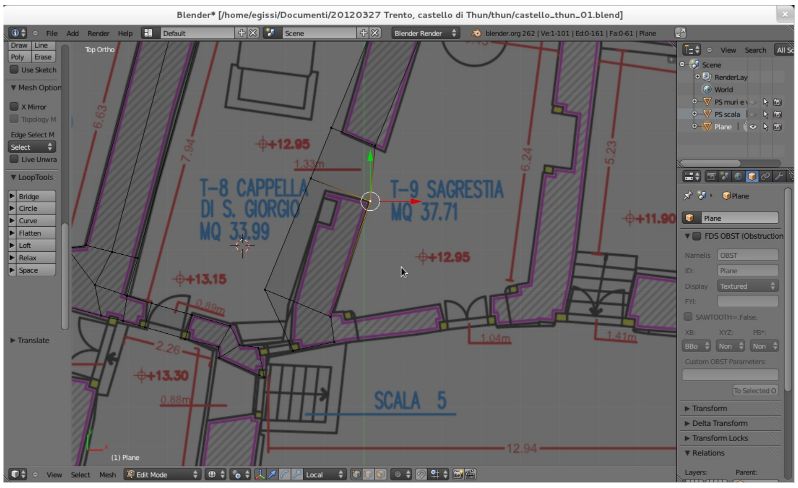
In fact, the fire safety study of Castel Thun required a CFD simulation of the diffusion of heat and combustion products from selected fire scenarios, involving the whole building. As it is the case for many medieval buildings, Castle Thun geometry is complex and irregular.

A BlenderFDS development version, installed as add-on of Blender 2.63 on a Fedora Linux 16 platform, was used for the whole pre-processing phase: solid modeling, boundary condition description, and definition of fire scenarios.

Fig. 8. Some interior details of Castel Thun

Several bugs where discovered in BlenderFDS code base and corrected.

Many FDS input files composed of more than 50 000 namelist groups were generated by the final BlenderFDS model. FDS processing has been started and is still ongoing. Preliminary results of the fire safety study were recently presented [10].


Blender* [/home/egissi/Documenti/20120327 Trento, castello di Thun/thun/castello_thun_01.blend]
T-8 CAPPELLA DI S. GIORGIO MQ 33.99
T-9 SAGRESTIA MQ 37.71
SCALA 5
FDS OBST (Obstruction)
Edit Mode
Fig. 9. BlenderFDS user interface, solid modelling phase

The next stable release of BlenderFDS package is going to include a simplified model of Castel Thun as an example and test case.

3 RESULTS

Figure Errore: sorgente del riferimento non trovata shows the final Castel Thun solid model in BlenderFDS (left) and the generated fluid dynamics calculation domain as processed by FDS and visualized by Smokeview (right).

The data input process alone required around 24 man-hours in BlenderFDS. Several iterations were performed to check the precise description of the solid and the appropriate generation of FDS solids and boundary conditions.

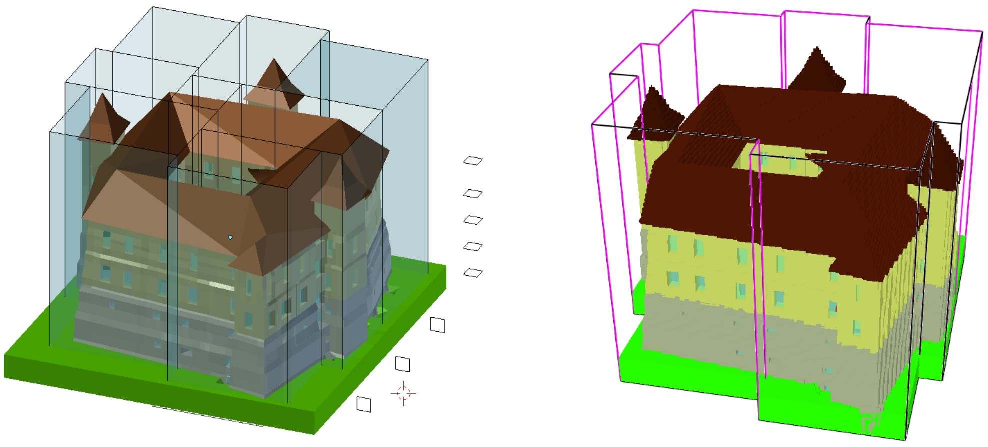


Fig. 10. Castel Thun exterior in BlenderFDS (left) and Smokeview (right)

Figure Errore: sorgente del riferimento non trovata shows an example of preliminary results of FDS calculations. The scenario sc0-1 describes the *business as usual* situation, involving no additional protection mean than existing ones, no changes from current safety management system. A fire is assumed to ignite at t = 0 in the castle basement, as a fire occurring on lower floors is supposed to pose more hazards to the occupants than a fire occurring on the upper floors. The heat release rate curve is modeled as a t-squared fast fire, capping at HRR_{max} = 2.5 MW.

The model forecasts that at t = 360.6 s from ignition the smoke has spread to the upper floors, generating untenable conditions for the occupants visiting several exhibition halls.

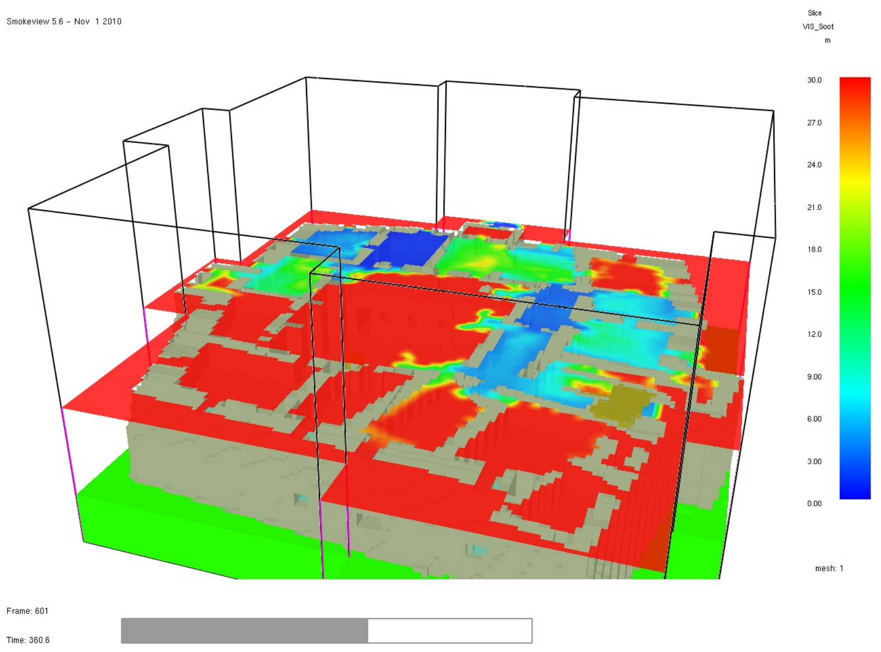


Fig. 11. Scenario sc0-1 (business as usual)
soot visibility at z = 18.97 m (1st floor) and t = 360.6 s, preliminary results.

4 DISCUSSION

The authors exclusively employed open source, freely available software tools for all the phases of a CFD simulation of smoke and heat transport from fire. Processing and post-processing, where performed respectively with FDS and Smokeview. The preliminary pre-processing phase was rapidly and precisely completed by employing BlenderFDS tool.

Except for closed source, commercial tools, available on proprietary closed platforms only, the only available alternative could have been typing the FDS text input file by hand, following the rules of FDS syntax and semantic.

This procedure would had allowed for a direct and full control over the input data. But, due to geometric complexity, the authors considered that alternative as being objectively unpractical.

BlenderFDS allowed for a satisfactory control over the input data, and the generated namelists groups:

- simulation control parameters, boundary conditions, physical and chemical properties, control logic description were entered following the rules of FDS syntax and semantic in BlenderFDS user interface. This data is not interpreted, checked, or otherwise modified by BlenderFDS: it is exported to the FDS input file as it is.
- the geometrical features of the model were described using the BlenderFDS state-of-the-art graphical user interface, and exported to FDS namelist groups using the included predictable algorithm, obtaining reproducible and readable results.

A single Blender data file contained all the required data for all the simulated scenarios regarding the Castel Thun historical building. Data sharing between scenarios and objects prevented duplication of efforts and lowered the risk of incoherent description or typos.

The full simulation tool-chain was completely under user control and allowed for easy to implement modifications to suit the users' needs.

Some limitations were recognized during the evaluation:

- Except for community-based support offered on a volunteer basis and online documentation, no support was available for BlenderFDS users during the development of the project.
- Both the inherent complexity of 3D modeling tools and the peculiar interface of BlenderFDS can involve the need of a non-negligible investment of resources for the new user to become productive.
- BlenderFDS is intended for users that already have a basic knowledge on how FDS works and can autonomously write FDS input files. BlenderFDS does not facilitate in any way the learning curve of students and professionals approaching the fire safety engineering topics.

These three limitations had little or no impact on the described evaluation, because of the deep understanding of the inner workings of the tool possessed by the authors, and their professional experience on fire safety engineering topics, but could adversely affect other users' efficiency and efficacy while following the same route.

5 CONCLUSION

This study demonstrates the value of a fully open tool-chain for CFD fire safety analysis. Knowing that CFD results can only be at best as good as the entered input data, the experienced user desires to retain full control over the entire process, and particularly on the data input phase. The set of tools evaluated in this study allows for a satisfactory user control over data input, processing, and visualization during the complex procedures of a CFD fire safety analysis.

The lack of an open pre-processor tool motivated the development of BlenderFDS, the open, community-based, user interface for FDS. Then a through evaluation was performed, applying the tool to the fire safety analysis of a complex real world test case.

Some limitations were recognized, that need proper solution: the process of lowering the entry barriers for fire engineers to CFD fire simulation in engineering applications seems still to have a long way ahead.

BlenderFDS tool shall follow future evolution of FDS ecosystem, both in terms of new FDS features and in terms of FDS users' community needs, as it is currently doing. Its open, bottom-up development model seems to be mostly appropriate to withstand such a challenge.

ACKNOWLEDGMENTS

We gratefully acknowledge the *Corpo nazionale dei Vigili del fuoco* and the *Autonomous Province of Trento* for supporting this applied research project.

REFERENCES

[1] McGrattan , K., Hostikka , S., Floyd, J., Fire Dynamics Simulator (Version 5) User's Guide, National Institute of Standards and Technology, Special Publication 1019-5

[2] Gissi, E., An Introduction to Fire Simulation with FDS and Smokeview, Corpo nazionale dei Vigili del fuoco – Ministero dell'Interno, coursebook, http://www.emanuelegissi.eu

[3] BlenderFDS project web site: http://www.blenderfds.org

[4] Blender project web site: http://www.blender.org

[5] De Santis, N., Faletti, G., Gissi, E., Valpreda, F., Blender for Fire Safety, Blender Conference, Amsterdam, 2009

[6] Lorensen, W. E., Cline, H. E., Marching Cubes: A high resolution 3D surface construction algorithm. Computer Graphics, Vol. 21, Nr. 4, July 1987 .

[7] Newman, T. S., Yi, H., A survey of the marching cubes algorithm, Computers Graphics, Volume 30, Issue 5, October 2006, Pages 854-879.

[8] FSF project web site: http://www.gnu.org/copyleft/gpl.html

[9] Ju, T., Losasso, F., Schaefer, S., Warren, J., Dual Contouring of Hermite Data, Proceedings of ACM SIGGRAPH, 21-26 July 2002.

[10] Gissi, E., Longobardo, G., Gli strumenti di modellazione applicati all'ingegneria della sicurezza antincendio; presentazione del lavoro di simulazione dell'incendio di Castel Thun, Convegno Vigili del fuoco del Distretto Mezzolombardo, Ton (TN), Italy, June 2012.