

Blueprint for **Blender+FDS**

Emanuele Gissi¹, Roberto Bartola²,
Nicola De Santis³, Fabrizio Valpreda⁴, Gianluca Faletti⁵
Kristopher Overholt⁶, Johannes Dimyadi⁷,
Roberto Orvieto⁸

November 10, 2009

¹Comando provinciale dei Vigili del Fuoco di Genova, Italy

²Freelance 3D designer, Ancona, Italy

³DII, Università del Salento, Italy

⁴DIPRADI, Politecnico di Torino, Italy

⁵Freelance multimedia design consultant, Torino, Italy

⁶Worcester Polytechnic Institute, USA

⁷University of Auckland, New Zealand

⁸Fire protection consultant, Genova, Italy

Development

Download a revised version of this document from:

- <http://www.fdscommunity.org/>

You can participate in the development of this document. Please, contact us at:

- emanuele.gissi@gmail.com

This document was produced using LyX and OpenOffice.org on Ubuntu Linux. The \LaTeX file `blender+fds_blueprint.tex` was compiled on November 10, 2009.

Copyright notice

This work is licensed under the Creative Commons Attribution-Share Alike 3.0 License. To view a copy of this license, visit:

- <http://creativecommons.org/licenses/by-sa/3.0/>

You are free to use, share, adapt this work. Remember to cite the sources and to use the same open license for derivative work.



Contents

1	What is Blender+FDS	1
1.1	Goals	1
1.2	Development notes	2
1.3	Workflows	2
1.3.1	Current workflow	3
1.3.2	New workflow	3
2	Writing an FDS input file	5
2.1	Basic syntax	5
2.1.1	Namelist groups and comments	7
2.1.2	Parameters	7
2.2	Prescribing geometric entities	9
2.3	Prescribing colors and aspect	11
2.4	Basic logic	11
2.4.1	Computational domain	11
2.4.2	Solid objects and boundary conditions	12
2.4.3	Types of boundary conditions	15
2.4.4	Other geometric entities	16
2.4.5	Other data	16

3	Exporting geometry from Blender to FDS	17
3.1	FDS surfaces ↔ Blender materials	17
3.2	FDS geometric entities ↔ Blender objects	18
3.2.1	FDS volume entities	18
3.2.2	FDS face entities	21
3.2.3	FDS segment entities	21
3.2.4	FDS point entities	22
3.2.5	FDS plane entities	23

Chapter 1

What is Blender+FDS

1.1 Goals

Fire Dynamics Simulator (FDS) is a free open source computational fluid dynamics model of fire-driven fluid flow, available for all major operating systems. The model 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. The partial derivatives of the conservation equations of mass, momentum and energy are approximated as finite differences, and the solution is updated in time on a three-dimensional, rectilinear grid.

Blender is a free open source 3D content creation suite, available for all major operating systems.

The goal of the *Blender+FDS* project is to create a multi-platform graphical user interface for FDS based on Blender platform.

Blender+FDS simplifies the input of geometric data into FDS, while leaving the full control over the input file to the user. Blender+FDS does not mask the complexity of the CFD analysis.

All the code from this project will be released under a GPL license, and will be *open* and *free*.

In this first iteration of the project, we want Blender to manage FDS geometric entities and boundary conditions. Blender shall know nothing of the FDS meaning of these objects. No sanity check shall be performed on FDS specific parts of FDS input file; this is left to FDS processor.

If this project is successful, more developments will follow, in particular:

- Detailed management of FDS objects in Blender.
- Blender import and visualization of fire and smoke data from FDS.

1.2 Development notes

- Blender interface shall be simplified. Blender will be basically used for architectural modeling, thus unneeded tools shall be hidden. These advanced tools shall remain available to power users.
- The interface and the workflow shall be as Blender-like as possible. Blender shall remain “recognizable”.
- FDS-specific additions to Blender UI shall be kept separate from standard Blender interface elements.
- FDS specific Blender objects shall be developed by inheriting standard Blender objects.
- Automated test cases shall be included in Blender+FDS code.
- Blender target release is 2.5x, the new python API shall be used.
- FDS target release is version 5.
- All the special FDS properties shall be saved in the .blend file. The .blend file should also record:
 - the global `voxel_size` chosen by the user.
 - the FDS case working directory.
 - the path to the text header file.

1.3 Workflows

Here the current FDS-simulation workflow is compared to the new designed workflow.

1.3.1 Current workflow

All the required information to perform an FDS simulation has to be contained in a single text file. The user writes a text file containing all input data. The simulated-ambient geometric description is entered line by line:

```
&HEAD CHID='pplume5', TITLE='Plume case' /
&TIME T_END=10.0 /
&MISC SURF_DEFAULT='wall', TMPA=25. /
&REAC ID='polyurethane', SOOT_YIELD=0.10,
&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,0.0,0.8 /
&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,0.8,1.6 /
&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,1.6,2.4 /
...
```

When the text input file is ready, the user runs the simulation from the command line.

The current process is prone to errors, long, and tedious.

1.3.2 New workflow

The new proposed workflow is the following:

- Build the geometry with the user preferred CAD application (or directly with Blender).
- Import the geometry into Blender.
- Divide solid elements into different Blender objects. Each Blender object represents an FDS geometric entity and it shall have homogeneous FDS properties: same boundary conditions...
- Associate FDS types of boundary conditions to Blender materials (SURF, see Section 2.4.3 on page 15).
- Apply Blender materials to Blender objects.
- Create new geometric FDS entities, as FDS VENT, FDS MESH objects (see Section 2.4.1 on page 11) and FDS output parameters (DEVC, SLCF... see Section 2.4.4 on page 16).

- Write the FDS header text file inside Blender editor (with FDS syntax highlighting) or with the user preferred editor (see Section 2.4.5 on page 16).
- Save the .blend file, containing all the geometric information and special FDS properties.
- Export Blender objects to a text file in FDS notation.
- Join the header text file and the Blender exported geometry, thus obtaining the final FDS input file.
- Run the FDS simulation as usual, from the command line.

Finally the user obtains:

- A Blender .blend file that contains all the geometrical aspects of the simulation and specific FDS properties.
- An FDS header text file: it contains all the non geometrical parameters of the simulation.

Chapter 2

Writing an FDS input file

2.1 Basic syntax

All the necessary information to perform an FDS simulation has to be contained in a single text file. The input file is saved with a name such as `mycase.fds`. There should be no blank spaces in the job name.

The following is an example of an FDS input file:

```
### General configuration

&HEAD CHID='pplume5', TITLE='Plume case' /
      name of the case and a brief explanation
&TIME T_END=10.0 /
      the simulation will end at 10 seconds
&MISC SURF_DEFAULT='wall', TMPA=25. /
      all bounding surfaces have
      a 'wall' boundary condition
      unless otherwise specified,
      the ambient temperature is set to 25°C.
&REAC ID='polyurethane', SOOT_YIELD=0.10,
      N=1.0, C=6.3, H=7.1, O=2.1 /
      predominant fuel gas for the mixture fraction model
      of gas phase combustion

### Computational domain

&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,0.0,0.8 /
```

```

&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,0.8,1.6 /
&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,1.6,2.4 /
&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,2.4,3.2 /
    four connected calculation meshes
    and their cell numbers

### Properties

&MATL ID='gypsum_plaster', CONDUCTIVITY=0.48,
    SPECIFIC_HEAT=0.84, DENSITY=1440. /
    thermophysical properties of 'gypsum plaster' material
&PART ID='tracers', MASSLESS=.TRUE., SAMPLING_FACTOR=1 /
    a type of Lagrangian particles
&SURF ID='burner', HRRPUA=600.,
    PART_ID='tracers', COLOR='RASPBERRY' /
    a type of boundary conditions named 'burner'
&SURF ID='wall', RGB=200,200,200, MATL_ID='gypsum_plaster',
    THICKNESS=0.012 /
    a type of boundary conditions named 'wall'

### Solid geometry

&VENT XB=0.5,1.1,0.5,1.1,0.1,0.1, SURF_ID='burner' /
    the 'burner' boundary condition
    is imposed to a plane face
&OBST XB=0.5,1.1,0.5,1.1,0.0,0.1, SURF_ID='wall' /
    a solid is created, 'wall' boundary condition
    is imposed to all its faces
&VENT XB=0.0,0.0,0.0,1.6,0.0,3.2, SURF_ID='OPEN' /
&VENT XB=1.6,1.6,0.0,1.6,0.0,3.2, SURF_ID='OPEN' /
&VENT XB=0.0,1.6,0.0,0.0,0.0,3.2, SURF_ID='OPEN' /
&VENT XB=0.0,1.6,1.6,1.6,0.0,3.2, SURF_ID='OPEN' /
&VENT XB=0.0,1.6,0.0,1.6,3.2,3.2, SURF_ID='OPEN' /
    the 'OPEN' boundary condition is imposed to
    the exterior boundaries of the computational domain

### Output

&DEVC XYZ=1.2,1.2,2.9, QUANTITY='THERMOCOUPLE', ID='tc1' /
    send to output: the data collected by a thermocouple
&ISOE QUANTITY='TEMPERATURE', VALUE(1)=100.0 /

```

```

3D contours of temperature at 100°C
&SLCF PBX=0.8, QUANTITY='TEMPERATURE', VECTOR=.TRUE. /
    vector slices colored by temperature
&BNDF QUANTITY='WALL TEMPERATURE' /
    surface 'WALL_TEMPERATURE' at all solid obstructions
&TAIL / end of file

```

2.1.1 Namelist groups and comments

Data is specified within the input file by using *namelist groups*. Each namelist group record occupies a line of text and begins with the & character. The & must be the first character of the line of text and should be followed by the name of the namelist group. Then a comma-delimited list of the input parameters is inserted. Finally a forward slash / character closes the namelist group, as shown in Figure 2.1.

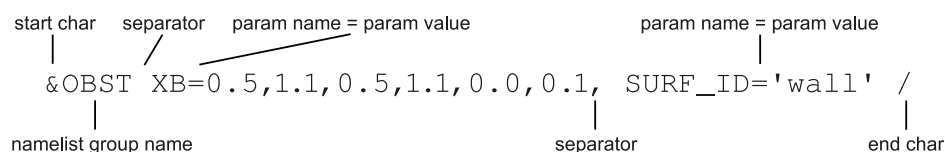


Figure 2.1: The structure of an FDS namelist group

Spaces and new lines can be freely inserted to visually format the namelist group. FDS uses the information contained between the & / delimiters, the rest is ignored. Thus, comments and notes can be written outside the & / delimiters. Comments For example:

```

&OBST XB=0.5,1.1,0.5,1.1,0.0,0.1 / A comment
Another comment

```

2.1.2 Parameters

The parameter values can be of the following types:

Parameter values

Integers, as in T_END=5400

Real numbers, as in CO_YIELD=0.008

Groups of real numbers, as in `XYZ=6.04,0.28,3.65`

Groups of integers, as in `IJK=90,36,38`

Character strings, as in `CHID='this_is_a_string'`

Groups of character strings, as in `SURF_IDS='burner','steel'`

Logical parameters, as in `POROUS_FLOOR=.FALSE.` or `POROUS_FLOOR=.TRUE.`
The periods must be included.

Parameter arrays Parameters can be entered as multidimensional arrays. For example:

```
MATL_ID(2,3)='brick'
```

indicates that the third material component of the second layer is brick.

To speed up data input, you can use this notation:

```
MATL_ID(1:3,1)='plastic','insulation','steel'
```

which means that the surface is composed by three different layers made respectively of plastic, insulation and steel. The notation `1:3` means array element 1 through 3, inclusive.

A simplified notation is accepted, too:

```
MATL_ID='plastic','steel'
```

is equivalent to:

```
MATL_ID(1:2,1)='plastic','steel'
```

These last surfaces are composed by two different layers made respectively of plastic and steel.

Case sensitivity The code is case sensitive: `my_burner` is not the same as `MY_BURNER`.

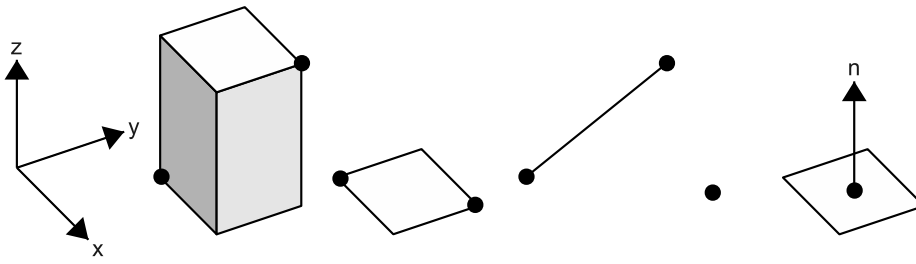


Figure 2.2: The reference system, a volume, a face, a segment, a point, and a plane

2.2 Prescribing geometric entities

Many namelist groups extend their action to volumes, faces, segments, points or planes. As shown in Figure 2.2, FDS geometrical entities are always described using some conventional rules.

Reference FDS *reference coordinate system* conforms to the right hand rule. By default, the z axis is considered the vertical.

Volumes A *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 vertexes: if point $A = (x_A, y_A, z_A)$ and point $B = (x_B, y_B, z_B)$ are the opposite vertexes, its coordinates are entered as $x_A, x_B, y_A, y_B, z_A, z_B$. For example,

```
&OBST XB=0.5,1.5,2.0,3.5,-2.0,0., SURF_ID='wall' /
```

uses the parameter XB 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.

A *face* is represented by a right plane face with edges parallel to the axis. Its position and dimensions are described by the coordinates of two opposite vertexes, that must lie on the same plane. For example:

```
&VENT XB=0.5,1.1,2.0,3.1,-2.0,-2.0, SURF_ID='fire' /
```

uses the parameter XB to define a flat face perpendicular to the z axis imposing a particular boundary condition over a solid. Two of the six coordinates are the same, denoting a flat face as opposed to a solid.

A *segment* is bounded by two end points. If point $A = (x_A, y_A, z_A)$ and point $B = (x_B, y_B, z_B)$ are the end points, its coordinates are entered following the same convention valid for volumes. For example,

```
&DEVC XB=0.5,1.5,2.0,3.5,-2.0,0., QUANTITY='PATH OBSCURATION',
      ID='beam1', SETPOINT=0.33 /
```

is a beam smoke detector between $(0.5, 2.0, -2.0)$ and $(1.5, 3.5, 0.)$ end points.

Points A *point* is simply identified by its 3 coordinates. For example, the line:

```
&DEVC XYZ=2.,3.,4., QUANTITY='THERMOCOUPLE', ID='termo1' /
```

uses the parameter XYZ to insert a thermocouple at the point of coordinates $(2., 3., 4.)$.

Planes A *plane* is represented by a right plane perpendicular to one of the reference axis. For example, these lines:

```
&SLCF PBX=0.5, QUANTITY='TEMPERATURE' /
```

is a plane perpendicular to the x axis and intersecting its point $(.5, 0., 0.)$.

```
&SLCF PBX=1.5, QUANTITY='TEMPERATURE' /
```

is a plane perpendicular to the y axis and intersecting its point $(0., 1.5, 0.)$.

```
&SLCF PBZ=-.5, QUANTITY='TEMPERATURE' /
```

is a plane perpendicular to the z axis and intersecting its point $(0., 0., -.5)$.

All use the parameters PBX, PBX, PBZ to specify the coordinate in the direction of the perpendicular axis.

Units FDS employs the *units of measurement* from the International System (SI). Lengths are expressed in m.

2.3 Prescribing colors and aspect

Colors of objects can be prescribed with two parameters: RGB and COLOR.

RGB The RGB parameter is followed by a triplet of integer numbers in the range from 0 to 255, indicating the amount of red, green and blue that make up the color.

COLOR The COLOR parameter calls the name of a predefined color that must be entered exactly as it is listed in the color table: ACQUAMARINE, BANANA, BEIGE, BLACK, BLUE, BRICK, BROWN, CADMIUM ORANGE, CARROT, COBALT, CORAL, CRIMSON, CYAN, FIREBRICK, FLESH, GOLD, GRAY, GREEN, INDIGO, MAGENTA, MAROON, MELON, MINT, NAVY, OLIVE, ORANGE, ORCHID, PINK, PURPLE, RASPBERRY, RED, SALMON, SEPIA, SIENNA, SILVER, TAN, TEAL, TOMATO, TURQUOISE, VIOLET, WHITE, YELLOW, ...

You can rapidly find the whole color table of more than 500 colors by *googling* for FDS COLOR TABLE on the Internet.

For example, both the parameter RGB=0,0,255 and the parameter COLOR='BLUE' can be used to obtain a blue object.

Objects can be made semi-transparent by assigning a TRANSPARENCY parameter. The parameter value is a real ranging from 0 to 1, with 0 being fully transparent. The parameter should always be set along with RGB or COLOR.

Using COLOR='INVISIBLE' causes the object not to be drawn in Smokeview. The parameter OUTLINE=.TRUE. causes the object to be drawn as an outline.

2.4 Basic logic

2.4.1 Computational domain

FDS fire simulation are performed inside a computational domain that is made up of rectilinear volumes called meshes. Each mesh is divided into rectangular cells, the number of which depends on the desired resolution of the flow dynamics.

The computational domain can consist of many connected mesh. Each mesh must have its MESH namelist group.

For example,

```
&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,0.0,0.8 /
&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,0.8,1.6 /
&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,1.6,2.4 /
&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,2.4,3.2 /
```

Smokeyview 5.2.2 - Jul 18 2008

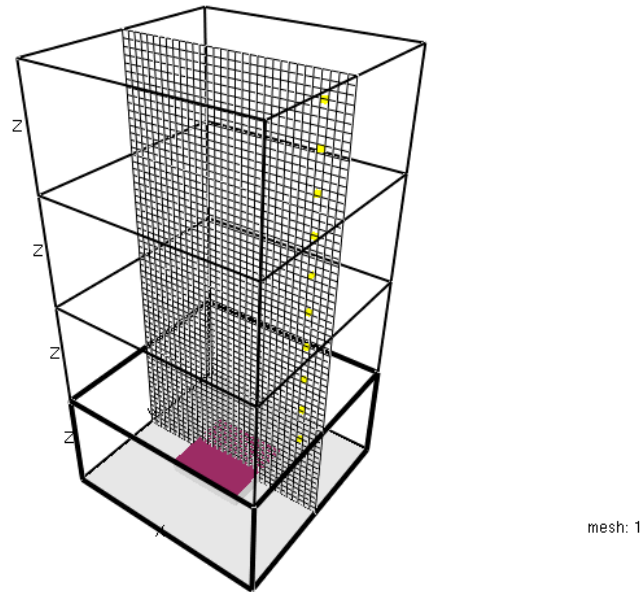


Figure 2.3: The computational domain composed by four meshes

describes a domain composed of four connected meshes, as in Figure 2.3.

All geometric objects must conform to the rectangular mesh. If you create geometrical objects that do not precisely conform to the underlying mesh, FDS shifts them to the closest mesh cell as shown in Figure 2.4.

2.4.2 Solid objects and boundary conditions

Usually the computational domain is full of solid objects that obstacle the fluid flow. Thus, the user should enter the geometric data describing solid obstacles.

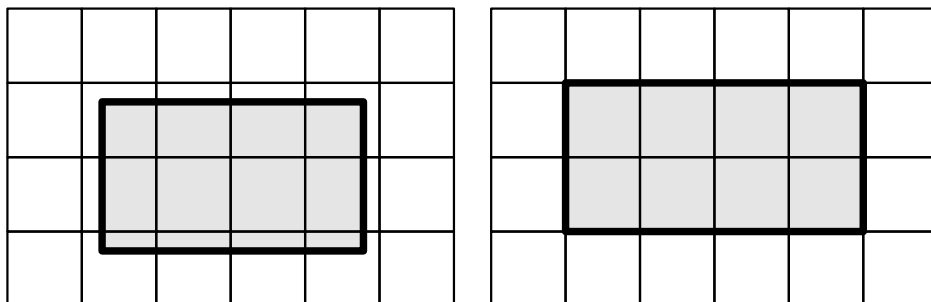


Figure 2.4: Geometric object: before and after automatic shifting

While building the solid geometry, the user have to apply boundary conditions to each of the bounding surfaces of the flow domain.

In FDS jargon the *boundary conditions* are often called *surfaces*.

Boundary conditions or surfaces are equivalent to a “paint” that the user should apply on all the bounding surfaces of the flow domain: the *faces* of the solid obstructions and the *exterior boundaries* of the computational domain.

For example, the namelist group OBST contains parameters used to define a solid obstruction:

```
&OBST XB=2.3,4.5,1.3,4.8,0.0,9.2, SURF_ID='brick wall' /
```

builds a solid obstruction within the volume $2.3 < x < 4.5$, $1.3 < y < 4.8$, $0.0 < z < 9.2$ and applies the brick wall boundary condition to all its six faces. The parameter SURF_ID calls the brick wall type of boundary condition by referring to its identifier string.

The HOLE namelist group is used to carve a hole out of an existing obstruction or set of obstructions. For example:

```
&HOLE XB=2.0,4.5,1.9,4.8,0.0,9.2 /
```

Any solid mesh cells within the volume $2.0 < x < 4.5$, $1.9 < y < 4.8$, $0.0 < z < 9.2$ are removed. Obstructions intersecting the volume are broken up into smaller blocks.

The user often needs to apply a particular boundary condition to a rectangular patch of an entire face or to the exterior boundaries of the computational domain.

The VENT namelist group is used to prescribe these particular boundary conditions:

- on *flat faces* adjacent to obstructions,
- or *exterior boundaries* of the computational domain.

For example, the lines:

```
&VENT XB=1.0,2.0,2.0,2.0,1.0,3.0, SURF_ID='burner' /
&OBST XB=0.0,5.0,2.0,3.0,0.0,4.0, SURF_ID='brick wall' /
```

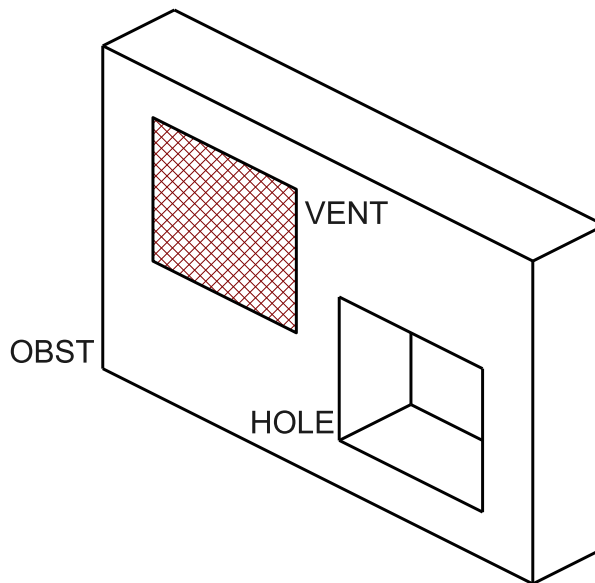


Figure 2.5: OBST, HOLE and VENT

build a solid obstacle made of brick wall and apply a burner boundary condition to a rectangular patch on the solid face in $-y$ direction.

To set boundary conditions to exterior boundaries of the computational domain the user can proceed as in the following example:

```

#### Computational domain
&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,0.0,0.8 /
&MESH IJK=32,32,16, XB=0.0,1.6,0.0,1.6,0.8,1.6 /
#### Properties
&SURF ID='brick wall', COLOR='BROWN' /
&SURF ID='floor', COLOR='SILVER' /
&SURF ID='ceiling', COLOR='SLATE GRAY' /
#### Solid geometry
&VENT XB=0.0,0.0,0.0,1.6,0.0,1.4, SURF_ID='brick wall' /
    lower part of -x exterior boundary
&VENT XB=0.0,0.0,0.0,1.6,1.4,1.6, SURF_ID='OPEN' /
    upper part of -x exterior boundary
&VENT XB=1.6,1.6,0.0,1.6,0.0,1.6, SURF_ID='OPEN' /
    +x exterior boundary
&VENT XB=0.0,1.6,0.0,0.0,0.0,1.6, SURF_ID='brick wall' /
    -y exterior boundary
&VENT XB=0.0,1.6,1.6,1.6,0.0,1.6, SURF_ID='OPEN' /

```

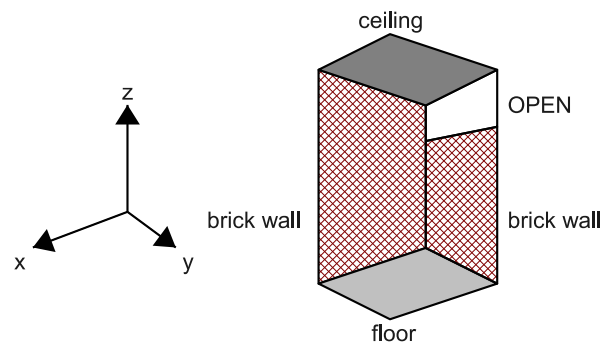


Figure 2.6: Setting boundary conditions to exterior boundaries of the computational domain

```
+y exterior boundary
&VENT XB=0.0,1.6,0.0,1.6,0.0,0.0, SURF_ID='floor' /
-z exterior boundary
&VENT XB=0.0,1.6,0.0,1.6,1.6,1.6, SURF_ID='ceiling' /
+z exterior boundary
```

The result is shown in Figure 2.6.

FDS is not a Computer Aided Design (CAD) tool, but a CFD code: not all geometrical details need to be entered in the input file. Looking at the example proposed in Figure 2.7 on the next page, chair and table frame effect on the fluid flow can be considered negligible. On the contrary, the influences to fluid flow of the separating wall, the table top and seats can become important, depending on the objective of the analysis.

2.4.3 Types of boundary conditions

The namelist group that defines the *types of boundary conditions* to be applied to solid obstructions is SURF.

For example,

```
&SURF ID='warm_surface', TMP_FRONT=25. /
```

defines a type of surface named `warm_surface`. Its temperature is fixed to 25°C. The SURF namelist provides the “paint pots” that the user will use to color all the bounding surfaces of the flow domain.

FDS contains some predefined boundary conditions that do not need to be set within a SURF namelist group: INERT, OPEN, and MIRROR.

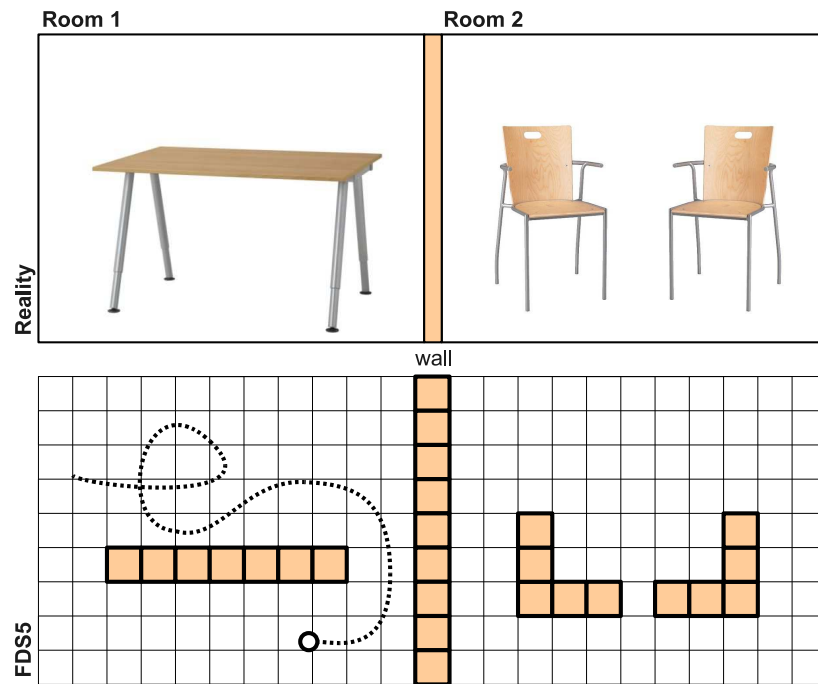


Figure 2.7: Modeling reality in FDS

2.4.4 Other geometric entities

Many other geometric entities can be inserted in the computational domain: devices to measure output quantities, initial conditions for the flow domain...

Also these geometric entities can be described as volumes, faces, segments, points or planes, as explained in Section 2.2 on page 9.

2.4.5 Other data

FDS needs much more configuration data: description of the case, time bounds of the simulation, gas phase reaction, description of involved materials...

This configuration data is not directly linked to geometric entities and is not managed by Blender.

Chapter 3

Exporting geometry from Blender to FDS

Blender shall manage:

- FDS surfaces, that correspond to Blender materials.
- FDS geometric entities, that correspond to Blender objects.

3.1 FDS surfaces ↔ Blender materials

Blender representation

Blender shall manage the list of FDS surfaces and their parameters, using the list of Blender materials. The additional properties for Blender materials shall be:

id a text string containing FDS identifier. This id is referred by other namelist groups to call the appropriate surface. This can be the Blender material name.

fds_param a multiline text string containing FDS parameters. For example:
CONVECTIVE_HEAT_FLUX=25., TMP_FRONT=150., EMISSIVITY=.9

rgb a triplet of integer numbers in the range from 0 to 255, indicating the amount of red, green and blue that make up the color. For example: 255,0,0. This can be the Blender material color.

comment a multiline text string containing user comment. For example: this is a warm surface

Exporting to FDS notation

When exported from Blender to FDS notation, this example material becomes the following FDS surface:

```
&SURF ID='warm_surface', CONVECTIVE_HEAT_FLUX=25.,
      TMP_FRONT=150., EMISSIVITY=.9, RGB=255,0,0 /
      this is a warm surface
```

In the future we could add TRANSPARENCY and OUTLINE FDS parameters.

FDS predefined boundary conditions (INERT, OPEN, and MIRROR) do not need to be set within a SURF namelist group. But they shall be available to the user and exist as predefined Blender materials.

3.2 FDS geometric entities ↔ Blender objects

Blender shall manage all FDS geometric entities. FDS geometric entities correspond to Blender objects.

FDS geometrical entities can be of the following types: volumes, faces, segments, points, planes, as explained in Section 2.2 on page 9.

3.2.1 FDS volume entities

Blender representation

An FDS volume entity is a Blender object containing a manifold mesh.

The additional properties for a volume entity shall be:

id a text string containing the FDS identifier. This can be the Blender object identifier. For example: `that wall`

namelist the namelist group name to be used when exporting¹. For example: `OBST`

fds_param a multiline text string containing FDS parameters. For example: `THICKEN=.TRUE.`

¹A pop down menu of known namelists?

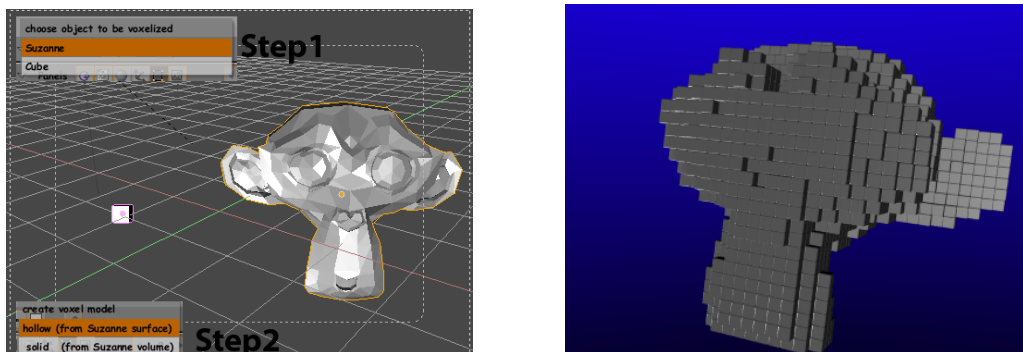


Figure 3.1: Voxelization process of a complex shape

surf_id the reference to the surface. In Blender it is the material linked to the object. For example: brick wall. This can be empty, in case of geometric entities not needing a surface as an HOLE, output quantities...

voxelize a boolean value. If true the object is voxelized, else its bounding box is used for FDS geometric coordinates. The voxel size is taken from the global variable `voxel_size`.

comment a multiline text string containing user comment. For example: this is a solid object

Exporting to FDS notation

In FDS, volumes can only be represented by single right parallelepipeds with edges parallel to the axis. So complex and round Blender shapes must be simplified to a sum of parallelepiped shapes that can be understood by FDS, as shown in Figure 3.1. This process is called voxelization².

Here is a ready to use voxelization script for Blender named Cells that somehow fits the task:

http://wiki.blender.org/index.php/Extensions:Py/Scripts/Manual/Add/Cells_v1.2

This Cells script lacks some aspects of the required transformation to FDS notation:

- The Cell script transforms the object in small cubes. The cubes should be joined in bigger parallelepipeds, whenever possible, to minimize the number of FDS lines of code.

²What are voxels? Please, have a look at this:

<http://bullandgate.blogspot.com/2009/05/mum-whats-voxel.html>

- It should conserve the object material.
- It should check if the Blender object mesh is manifold.
- It should check the triangulation of the Blender object mesh, and eventually transform it (ctrl-T in Blender edit mode).
- It should apply size and rotation to all objects (ctrl-A in Blender object mode).
- The voxels should be single right parallelepiped with edges parallel to the Blender global axis.
- Blender should ask the user for the minimum size of voxels.

When exported to an FDS input file, the example volume entity shall become:

- If voxelize parameter is false, the geometric coordinates are calculated using the object bounding box:

```
&OBST ID='that wall', XB=0.3,2.1,3.4,5.4,9.0,12.0,
      THICKEN=.TRUE.,
      SURF_ID='brick wall' /
      this is a solid object
```

- If voxelize parameter is true, the object is voxelized to represent its real shape.

```
Voxelized object: that wall
Comment: this is a solid object
Bounding box: 0.3,2.1,3.4,5.4,9.0,12.0
Voxels:
&OBST ID='that wall', XB=0.1,0.4,3.4,5.4,9.0,12.0,
      THICKEN=.TRUE.,
      SURF_ID='brick wall' /
&OBST ID='that wall', XB=0.2,2.0,3.4,5.4,9.0,12.0,
      THICKEN=.TRUE.,
      SURF_ID='brick wall' /
&OBST ID='that wall', XB=0.3,1.9,3.4,5.4,9.0,12.0,
      THICKEN=.TRUE.,
      SURF_ID='brick wall' /
...voxels...
```


3.2.2 FDS face entities

An FDS face entity is a Blender object containing a flat mesh.

Blender representation

The additional properties for a face entity shall be:

id a text string containing the FDS identifier. This can be the Blender object identifier. For example: `that burner`

namelist the namelist group name to be used when exporting. For example:
`VENT`

fds_param a multiline text string containing FDS parameters. For example:
`DEVC_ID='device'`

surf_id the reference to the surface. In Blender it is the material linked to the object. For example: `burner`. This can be empty, in case of geometric entities not needing a surface...

comment a multiline text string containing user comment. For example: `this is a burner`

Exporting to FDS notation

In FDS, faces can only be represented by a right plane face with edges parallel to the axis. So complex and round Blender faces must be simplified to a parallelepiped shape that can be understood by FDS.

When exported to an FDS input file, the geometric coordinates are calculated using the object bounding box:

```
&VENT ID='that burner', XB=0.5,1.1,0.5,1.1,0.1,0.1,  
      DEVC_ID='device',  
      SURF_ID='burner' /  
      this is a burner
```

3.2.3 FDS segment entities

An FDS segment entity is a new kind of Blender object.

Blender representation

The additional properties for a segment entity shall be:

id a text string containing the FDS identifier. This can be the Blender object identifier. For example: `that detector`

P1 the coordinates of the first extreme point. For example: `(0.5, 2.0, -2.0)`

P2 the coordinates of the second extreme point. For example: `(1.5, 3.5, 0.0)`

namelist the namelist group name to be used when exporting. For example: `DEVC`

fds_param a multiline text string containing FDS parameters. For example: `QUANTITY='PATH OBSCURATION', SETPOINT=0.33`

comment a multiline text string containing user comment. For example: `this is a beam detector`

Exporting to FDS notation

This is what shall be exported to an FDS input file:

```
&DEVC ID='that detector', XB=0.5,1.5,2.0,3.5,-2.0,0.,
      QUANTITY='PATH OBSCURATION', SETPOINT=0.33 /
      this is a beam detector
```

3.2.4 FDS point entities

An FDS point entity is a new kind of Blender object.

Blender representation

The additional properties for a point entity shall be:

id a text string containing the FDS identifier. This can be the Blender object identifier. For example: `termo1`

P1 the coordinates of the point. For example: `(2., 3., 4.)`

namelist the namelist group name to be used when exporting. For example:
 DEVC

fds_param a multiline text string containing FDS parameters. For example:
 QUANTITY='THERMOCOUPLE'

comment a multiline text string containing user comment. For example: this
 is a thermocouple

Exporting to FDS notation

This is what shall be exported to an FDS input file:

```
&DEVC ID='termo1', XYZ=2.,3.,4.,
      QUANTITY='THERMOCOUPLE', /
      this is a thermocouple
```

3.2.5 FDS plane entities

An FDS plane entity is a new kind of Blender object.

Blender representation

The additional properties for a plane entity shall be:

id a text string containing the FDS identifier. This can be the Blender object identifier. For example: that slice file

P1 the coordinates of the plane. For example: (0.5,0.,0.), this is a plane perpendicular to x axis, with $x = 0.5$

namelist the namelist group name to be used when exporting. For example:
 SLCF

fds_param a multiline text string containing FDS parameters. For example:
 QUANTITY='TEMPERATURE'

comment a multiline text string containing user comment. For example: this
 is a slice file

Exporting to FDS notation

This is what shall be exported to an FDS input file:

```
&SLCF ID='that slice file', PBX=0.5,  
      QUANTITY='TEMPERATURE' /  
      this is a slice file
```