Semester-Long Internship Report

On

OpenFOAM GUI Development using Python on Blender

Submitted by

Kartik Kumar Thakur

Kalinga Institute of Industrial Technology,
Bhubaneswar.

Under the guidance of

Prof. Janani Muralidharan

Department of Mechanical Engineering
IIT Bombay

And

Prof. Prabhu Ramachandran

Department of Aerospace Engineering
IIT Bombay

Acknowledgment

The Semester Long Internship, at FOSSEE, IIT Bombay selects interns from all over India based on performance in solving problems announced on their website. This ensures having the best possible team and I am glad that Co-Intern **Mr. Rajdeep Adak** (B. Tech in Electronics and Telecommunication Engineering.K J Somaiya College of Engineering, University of Mumbai) left no stone unturned in delivering the same. It gives me immense pleasure to express my sincere gratitude to **Prof. Janani Muralidharan** (Mechanical Engineering Dept. at IITB) for her valuable guidance and perceptive insight into the CFD simulation process. I would also like to express my indebtedness to **Prof. Prabhu Ramachandran** (Aerospace Engineering Dept. at IITB) whose critique formulated the very structure of our project. Regular interaction with professors ensured clarification of every concept required to strengthen our foundations. I highly value the iterative process of development as it became instrumental to devise efficient solutions and enabled Venturial to be a singular software for meshing and solving fluid flow problems. During the development process, we were presented with a plethora of information from FOSSEE CFD team members (**Ashley Melvin** and **Divyesh Variya**). They have been a source of direct feedback to evaluate the capabilities of Venturial. Most visualization features of Venturial are a result of suggestions from CFD team members. I would also like to extend my gratitude to **Mr. Ankit R. Javalkar** (Software Engineer, IITB) who helped assimilate fragments of important information from the preceding **Reynolds** software (by **Mr. Deepak Surti**). I thank **Mrs. Swetha Sridhar** (Project Manager of FOSSEE at IITB) who ensured timely completion of tasks with adequate adjustment of review meetings to assist interns in remaining academically active. Throughout the project, our ideas were welcomed with alacrity and commendations on achieving milestones were followed by encouragement to forge ahead.

I have developed an ardent admiration for FOSSEE's initiative to promote learning with the use of **FLOSS** (Free/Libre and Open Source Software) tools. I thank **Prof. Kannan Moudgalya** (Chemical Engineering Dept. at IITB) and all associated members for bestowing their knowledge and efforts in making such a venture possible.

Lastly, I hope our work has eased the process of CFD simulations with OpenFOAM. I am grateful for the holistic internship experience at FOSSEE, IITB which remained highly productive in WFH mode despite the appalling COVID pandemic. Skills gained during this internship have been a boon to career development. I thereby offer prolonged support and cooperation to FOSSEE and its future interns.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Background

OpenFOAM is a C++ based toolbox prominently used to solve fluid dynamics and heat transfer problems using customizable solvers and pre-processing utilities (such as blockmesh and snappyhexmesh). Post-Processing utilities allow solutions to be viewed graphically for performance evaluation under various parameters. However, pre-processing and geometry modelling via current methods requires the usage of multiple softwares, thorough knowledge of meshing techniques, solving techniques and the OpenFOAM case-building procedure. Reynolds, developed in 2016 at FOSSEE, IIT Bombay is a GUI add-on to Blender (a 3D open-source computer graphics software) that has blockmesh and snappyhexmesh utilities for solving. However, it was built for an older version of OpenFOAM and lacks the features required to solve multi-block geometry problems. This serves as a baseline for our work.

1.2 Related Work

SwiftBlocks: SwiftBlocks is a Blender GUI add-on for the OpenFOAM® blockmesh utility, which creates hexahedral block structured volume meshes for OpenFOAM simulations.

Reynolds-Blender: Reynolds-Blender is a reference implementation to demonstrate the integration of pre-processing components from Reynolds to build a GUI for the pre-processing steps of OpenFOAM.

1.3 Venturial

Venturial is a pair of add-ons to Blender designed by FOSSEE, IIT Bombay based on the blockmesh utility of OpenFOAM to perform pre-processing, meshing, and solving multi-block geometry fluid flow problems using a GUI. The 3D graphics development capabilities of Blender are leveraged to generate blocks (fundamental units) that can be manipulated to build a geometry. The blockmesh add-on has various panels that simplify meshing by a semi-automated process to fetch information such as vertices, blocks, edges, and faces dynamically from the geometry. Additional Geometry interaction features assist the user in mesh visualization. A separate solver add-on has in-built and editable solvers inside a dedicated solver directory. Opening a solver folder from this directory displays various sub-folders containing parameter property files in a drop-down menu. Parameters of the selected file can be edited using various editing options. Users can describe their own solvers by creating parameters from the assigned parameter feature.

1.4 Objective

Venturial caters to the needs of new learners as well as experienced individuals. Hence building algorithms to automate the process of meshing and solving to the farthest possible extent has been given paramount importance. We intend for new learners to grasp various aspects of solving fluid flow problems on OpenFOAM quickly and experienced individuals to freely experiment with custom-built solvers. New learners can use automation features of Venturial to quickly create blockmesh dictionary and parameter files without detailed knowledge of meshing and solving techniques. These automation features have been built in accordance with the meshing and solving rules of OpenFOAM. Additionally, visualization features have been developed with minimal compute usage. Venturial is supported for OpenFOAMv8 and Blender v2.8+.

1.5 Approach

Venturial ensures the accurate and error-free generation of a solver case directory since OpenFOAM has a stringent set of validity constraints for meshing. The meshing approach followed in Venturial is building a geometry using blocks only available in the GUI. Blender as a python module has been used to develop Venturial as it bundles programmable 3D graphical blocks with a GUI. Hence, it reduces the effort of calculating geometry information and removes the dependency on separate physical modeling software. Post-processing can be done on Paraview once the solver case directory has been generated accurately.

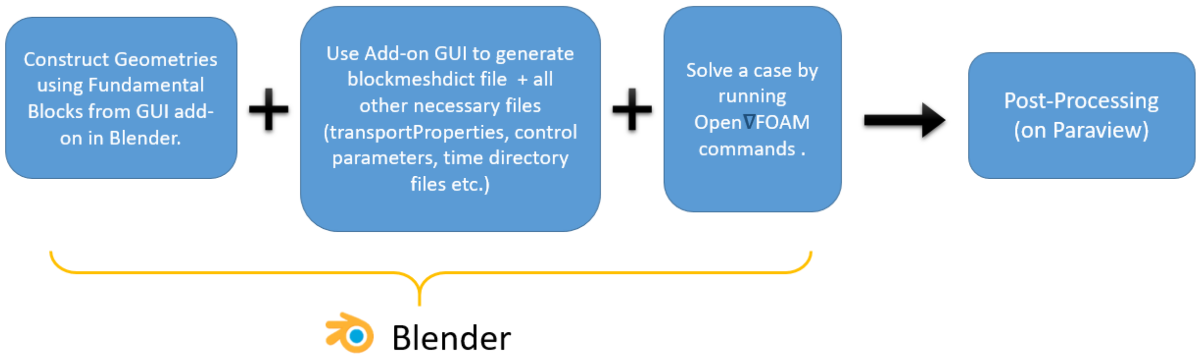

Construct Geometries using Fundamental Blocks from GUI add-on in Blender.
+
Use Add-on GUI to generate blockmeshdict file + all other necessary files (transportProperties, control parameters, time directory files etc.)
+
Solve a case by running OpenFOAM commands .
Post-Processing (on Paraview)
Blender

Figure 1: Approach of blockmesh add-on

1.6 Scope of the Project

The blockmesh add-on in its current state can produce a blockmesh dictionary for multi-block geometries via a combination of user-based input and automatic features. The solver add-on is, however, limited to files with only single-line parameters. Visualization features use custom blender handlers to display important information about the geometry to the user.

Both add-ons have a help section that documents the purpose of each feature. The default UI contains features necessary for file generation. Every panel has an advanced section that displays the visualization features when enabled.

1.7 System Requirements

- Blender 2.8+: For using the GUI add-on.
- PyFoam 0.6.8.1: ParsedParameterFile helps in parsing.
- OpenFOAMv8: Required to run simulations.
- User device that meets hardware requirements for Blender and OpenFOAM.

1.8 Report Organization

Chapter 1 introduces the idea of performing OpenFOAM CFD simulations using a GUI and previous work done for the same. 2nd Chapter mentions all parts of the Frontend aspects of the GUI pair. Architecture, Automation feature logic and version history milestones are detailed in the 3rd Chapter. For users intending to quickly begin using the add-on, Chapter 4 provides a step-by-step guide to solve a CFD problem (pipe flow simulation). Post-processing results on Paraview are also presented. All results are produced solely via the add-on and no text editing has been performed throughout the process.

Chapter 2 Frontend UI

2.1 Solver File System

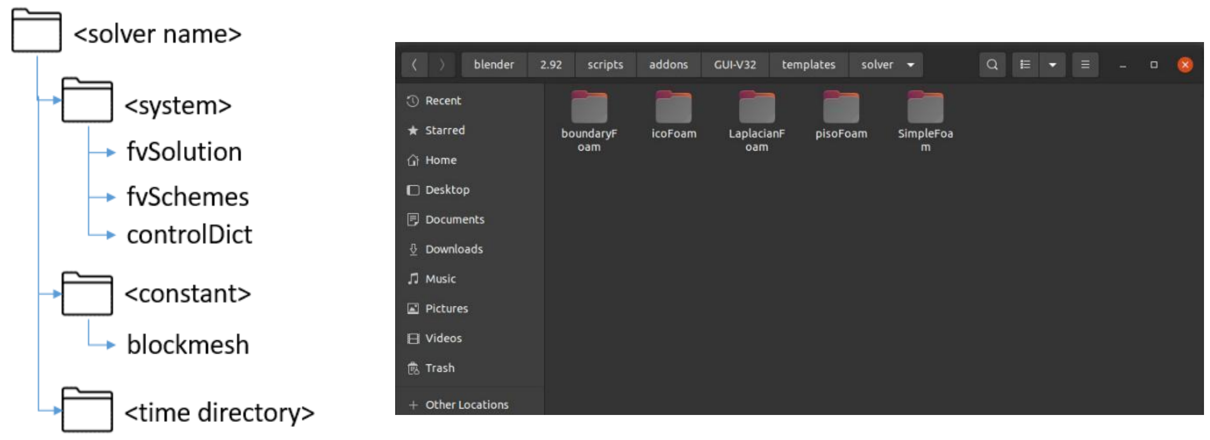


Figure 2: Solver Directory hierarchy

Venturial is intended to be developed with a library of pre-built solvers developed by the CFD Team at FOSSEE, IIT Bombay. Each solver will have a file structure mentioned in figure 2. The library of solvers will be a Solver Directory (as shown in figure 2) with a folder dedicated to each solver. The solvers within this library are readable and editable by the solver add-on. Any user can open the solver add-on and select their desired solver.

2.2 Meshing Criteria and Solving Criteria

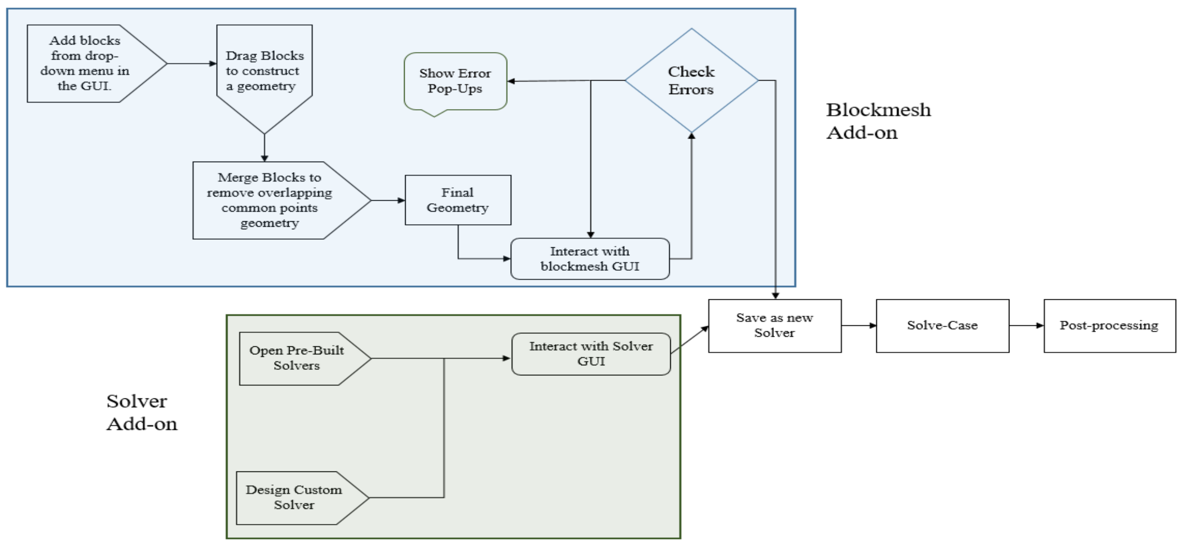


Figure 3: Meshing Flowchart

Currently, Venturial consists of blockmesh add-on and Solver Add-on separately (see above diagram). Users can use the same blender instance to run both the add-ons side-by-side. A separate add-on for blockmesh was built to segregate geometry design steps from Solver steps. However, the case directory files are generated in the same solver directory. The diagram above is an overview of the case directory formation procedure. Blockmesh add-on has error handling to display pop-ups when a mistake is identified by the software. Solver add-on restricts the user to only use parameters of pre-defined data types (Integer, Float, Boolean, and List). This eliminates the need for a text-based schema file. Users are provided with several editing options required to make changes to the parameter file and save/update them at any time.

2.3 Addressable Block Shapes

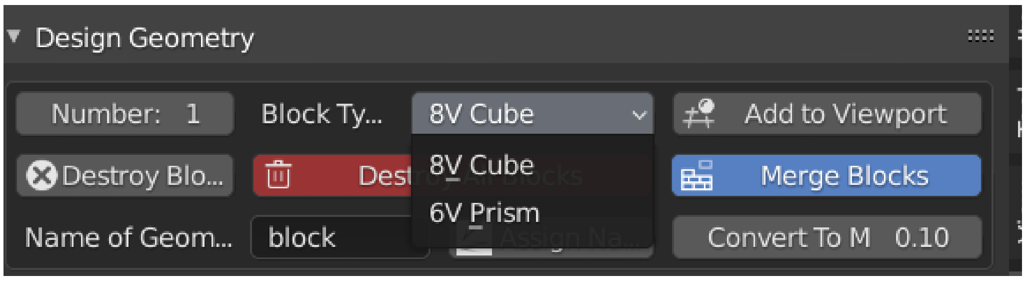


Figure 4: Design Geometry panel

Within the Design Geometry Panel a user can select among two shapes of blocks.

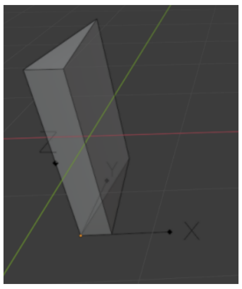


Figure 5: 6V Prism

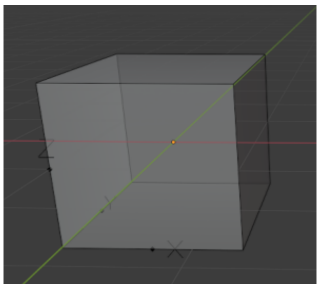


Figure 6: 8V hexahedron

8V Cube: A 3D shape consisting of 6 square planes (faces) each side of which is 2 meters in length, 12 edges and 8 vertices.

6V Prism: A 3D shape with a prism angle less than 5 degrees, 6 vertices, 9 edges and 5 faces.

These blocks can be edited in shape, size and orientation to create a desired shape within the following constraints:

a) The resultant block after editing in either case must retain the same number of edges, vertices and faces.

b) No face should be non-planar. Blender doesn't identify if a face has become non-planar during editing. Such a face will have at least one extra edge. Blockmesh utility simply calculates a face by the position of its vertices and the new edge isn't taken into account.

Currently, Venturial doesn't have a feature to identify if a face has become non-planar during editing of blocks.

2.4 Blockmesh UI

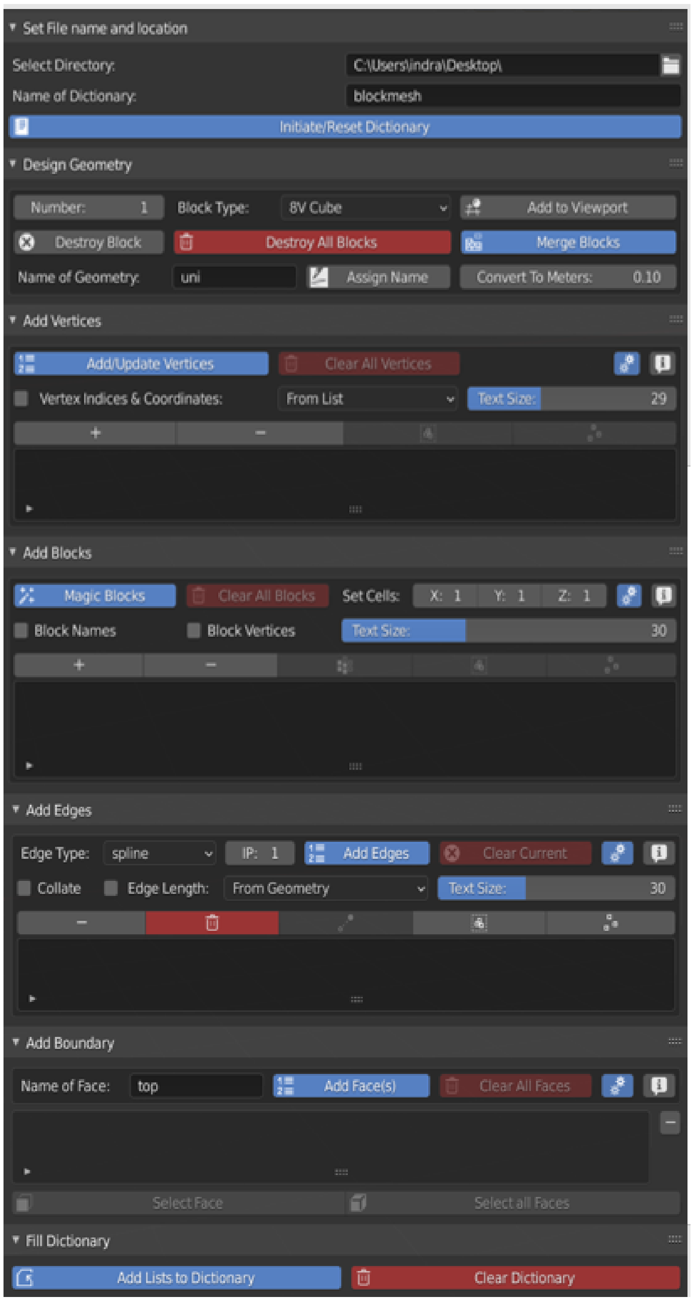

Select Directory: C:\Users\indra\Desktop\
Name of Dictionary: blockmesh
Number: 1
Block Type: 8V Cube
Name of Geometry: uni
Convert To Meters: 0.10
Vertex Indices & Coordinates: From List
Text Size: 29
Set Cells: X: 1 Y: 1 Z: 1
Text Size: 30
Edge Type: spline
IP: 1
Edge Length: From Geometry
Text Size: 30
Name of Face: top

Figure 7: Blockmesh Add-on UI

Blockmesh UI consists of 7 floating and hideable panels. By default, they appear in the standard order. Initiating a dictionary creates an empty blockmesh dictionary which acts as a workspace editable upon interaction with the UI.

The Design geometry section has features which summon eligible blocks into the viewport. Manipulating structure and orientation of the blocks are at the user's discretion. For this, Blender provides an array of simple shortcuts available here. This section is also used to name and build the final geometry. "Building" a geometry here means to join all selected blocks sharing at least one edge and remove common vertices, edges and faces.

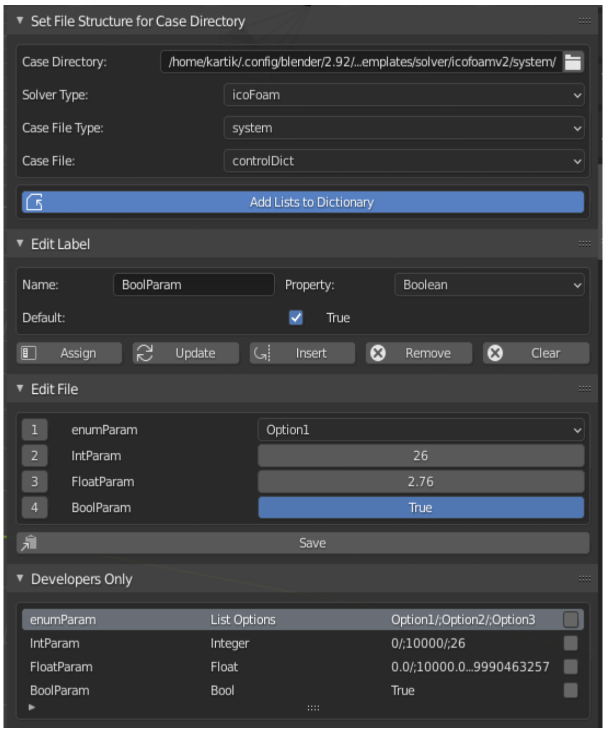
The next four panels are the sections of a blockmesh dictionary which describe the geometry in a "text" form interpretable by OpenFOAM. Users can interact with the geometry and the UI to populate the blockmesh dictionary. Each of the four panels also have visualization features.

The Fill Dictionary section is used to update the blockmesh dictionary upon clicking the Add Lists to Dictionary button.

The Blockmesh UI acts as a palette to deliver information to the blockmesh dictionary dynamically.

2.5 Solver UI

Developer Add-on:



▾ Set File Structure for Case Directory

Case Directory: /home/kartik/.config/blender/2.92/...emplates/solver/icofoamv2/system/

Solver Type: icoFoam

Case File Type: system

Case File: controlDict

Add Lists to Dictionary

▾ Edit Label

Name: BoolParam Property: Boolean

Default: ✓ True

Assign Update Insert Remove Clear

▾ Edit File

1	enumParam	Option1
2	IntParam	26
3	FloatParam	2.76
4	BoolParam	True

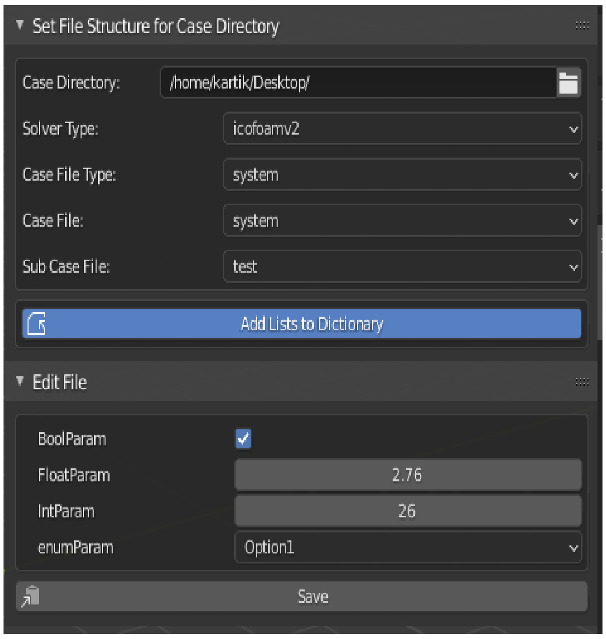
Save

▾ Developers Only

enumParam	List Options	Option1/;Option2/;Option3
IntParam	Integer	0/;10000/;26
FloatParam	Float	0.0/;10000.0...9990463257
BoolParam	Bool	True

- Set File structure for Case Directory: Contains the directory to the template folder and options to select the file that needs to be edited.
- Edit Label: Contains all the required options to add new parameters.
- Property: Type of parameter
- Assign: Adds the parameter to the list
- Remove: removes the selected parameter from the list
- Edit File: Contains the list of added parameters with their values and checkbox to select them
- Save: Creates the template in the templates folder with all parameters added to it.

Figure 8: Solver Developer Add-on UI



▾ Set File Structure for Case Directory

Case Directory: /home/kartik/Desktop/

Solver Type: icofoamv2

Case File Type: system

Case File: system

Sub Case File: test

Add Lists to Dictionary

▾ Edit File

BoolParam ✓

FloatParam 2.76

IntParam 26

enumParam Option1

Save

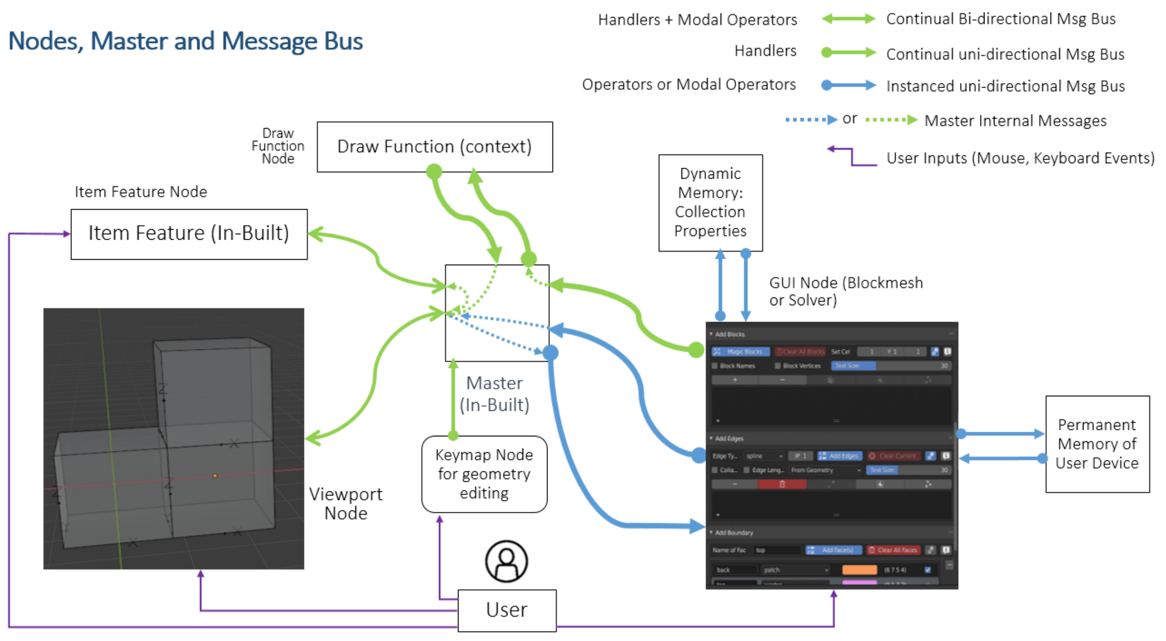
User Add-on:

- Set File structure for Case Directory: Contains the directory to the case folder and options to select the file that needs to be created.
- Edit File:Contains all the parameter files for the user to manipulate values.
- Save: Saves the file to the desired location

Figure 9: Solver User Add-on UI

Chapter 3 Backend Design

3.1 Add-on Architecture



Nodes, Master and Message Bus
Handlers + Modal Operators
Continual Bi-directional Msg Bus
Handlers
Continual uni-directional Msg Bus
Operators or Modal Operators
Instanced uni-directional Msg Bus
or
Master Internal Messages
User Inputs (Mouse, Keyboard Events)
Draw Function Node
Draw Function (context)
Item Feature Node
Item Feature (In-Built)
Dynamic Memory: Collection Properties
GUI Node (Blockmesh or Solver)
Master (In-Built)
Keymap Node for geometry editing
Viewport Node
Permanent Memory of User Device
User

Figure 10: Blockmesh add-on architecture

The above diagram depicts the interaction capabilities of various nodes with the User.
A Viewport Node is an in-built feature of blender which allows manipulation of structure, position, and orientation of 3D objects (blocks) by simple mouse clicks and drag.
An item feature node is a set of all in-built Blender options for any modifications necessary.
The GUI node consists of two add-ons which receive commands independently from the user. Based on the commands the GUI node can invoke storage privileges from Dynamic Memory and Permanent Memory of the device.
A Keymap Node is used to track a special identifiable Keyboard event for performing a geometry edge length edit function. It expects a Keyboard event(shift+F) to open up a dialog box for user input. The required change in edge length is then communicated over a unidirectional message bus. Changes executed will be reflected on the Viewport.
Draw Function Node is a set of visualization handler features which ships with the add-on. They can be enabled/disabled from the add-on interface. Inter-communication of various nodes occurs on the basis of the direction type of message bus. For example: to enable a Draw Function Node for observing data of geometry graphically on the Viewport, a command must be given to the GUI Node via a mouse event. The GUI Node then sends a response to the Master Node to enable the appropriate Visualization Handler. The Visualization Handler will produce required data over the Viewport UI by sending the data over a unidirectional message bus.
Venturial comprises the GUI Node, Item Feature Node, Keymap Node and Draw Function Node. Remaining Nodes are already present in Blender.

3.2 Version History

The blockmesh add-on and solver add-on of Venturial have been developed separately through continuous iterations. Critique and demands of CFD FOSSEE Team have been addressed based on priority of each feature present in the add-ons. Over a span of 5 months blockmesh add-on has been revised 32 times and solver add-on has been revised 7 times. During each of these revisions, a new feature has been added. As OpenFOAM requires a text based directory system, Venturial follows a method to dynamically edit text files based on user interactions with geometry. Each version has been built with the intention of minimizing the need to directly edit a text file. The following tables mentions the major milestones achieved by consequent revisions of blockmesh add-on and solver add-on versions.

3.2.1 Blockmesh Add-on Version History

Blockmesh Add-on	
Version	Milestone
Version 1	Add vertices section manually
Version 2	Add vertices section manually and automatically
Version 4	Add Blocks section manually
Version 5	Add Edges Section
Version 8	Add Faces section
Version 9	Separate section for geometry design
Version 12	Multiple features added for geometry section
Version 13	Added vertex visualization handler
Version 14	Added Block visualization handler, Save dictionary button
Version 16	Built Magicblocks feature and integrated to blockmesh add-on.
Version 19	Built advanced section for experienced users
Version 21	Added reference axes, Error handling and pop-ups
Version 24	Added Keymap feature for editing edge
Version 25	Added set cells feature
Version 28	New Look for GUI
Version 30	Advanced section modified to icons only buttons
Version 31	Help section added

Table 1: Blockmesh add-on version history

3.2.2 Solver Add-on Version History

Solver Add-on	
Version	Milestone
Version 1	Adding variables to a scene using a button
Version 3	Created method to select file
Version 4	Created Developer add-on
Version 5	Multiple editing options built
Version 6	User Add-on

Table 2: Solver add in version history

3.3 Automation Features

The automation features of Venturial are primarily meant for new learners of CFD but the original purpose is to reduce time and effort required to write a blockmesh dictionary. All automation features can also be used by advanced users. As OpenFOAM follows unique vertex indexing criteria for a geometry, algorithms were designed to automate writing various sections of blockmesh. Vertex indexing for 3D objects in Blender do not follow the same criteria in OpenFOAM. Hence, the following features were built to seamlessly generate the required case directory file system with minimum effort from the user side.

3.3.1 Blockmesh Add-on

Blockmesh Add-on	Method
Add/Update vertices	One click to add/update all vertices
Add to Viewport	One-click to add all required blocks to Viewport
Merge Blocks	One-click to remove all common faces, edges and vertices
Magic Blocks	One-click to add all blocks to a geometry
Keymap	Shift+F

Table 3: Components of Blockmesh Add-on

3.3.2 Solver Add-on

Solver Add-on	Method
Fill Blockmesh Dictionary	One click to add or update a blockmesh
Assign	One click to add a single parameter to file
Update	Update selected parameter
Insert	Insert a new parameter at desired location
Remove	Remove a parameter
Clear	Clear all parameters

Table 4: Components of Solver Add-on

3.4 Edge Resize Tool

To resize the edges of a geometry a utility tool has been added.On clicking SHIFT+F a pop up menu will appear with the help of which, an edge can be resized.

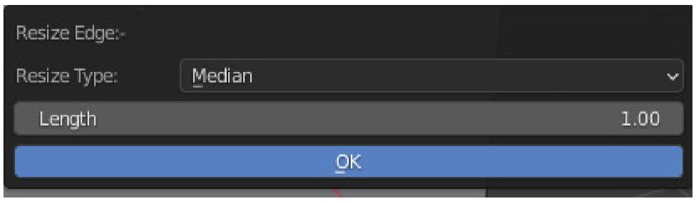

Resize Edge:-
Resize Type: Median
Length 1.00

Figure 11: Edge Resize tool

There are 2 options for resizing:

- Resize from center: It scales the vertex keeping the central point fixed
- Resize from Vertex: It scales the edge keeping one vertex of the edge fixed. The vertex to be fixed can be chosen.

3.5 Visualization Handlers

A Handler in Blender executes a function to update scene data on the basis of a predefined condition. Conditions can be predefined or user-defined. Venturial uses 2 types of handlers namely Application Handler and Draw Handlers.

3.5.1 Application Handlers

An Application Handler is used to display the reference axis of each block before and after a geometry is built. Upon adding a new block into viewport reference axes are also drawn to indicate the orientation of the block. The reference axes are parented to the 0th vertex of the block. But parenting to a vertex doesn't allow the same change in orientation. In order to keep the orientation same as well an application handler is used which constantly equalizes the orientation of the reference axes and the blocks. To make sure this occurs in the most efficient way, a dependency graph update application handler is used.

Application Handler usage:

```
def app_handler_function(scene):
        # Do something

register():
        bpy.app.handlers.depsgraph_update_post.append
        (app_handler_function)

unregister():
        bpy.app.handlers.depsgraph_update_post.remove
        (app_handler_function)
```

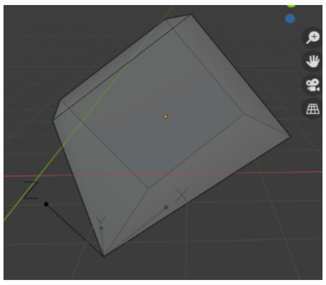


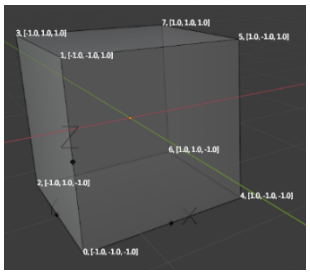
Figure 12: Display origin using handlers

The figure above shows an 8V cube where its shape has been edited within constraints and orientation change of block and reference axes remain the same.

3.5.2 Draw Handlers

The purpose of Draw Handlers is to help users visualize properties of a geometry in the Viewport. Draw Handler and the font drawing module are used in conjunction to display information in font style specified within the draw function. The draw function is passed as an argument to the draw handler. The draw handler calls the function whenever a change is observed in the scene. This causes a rise in CPU usage but only momentarily during scene change. Once scene data is drawn CPU usage returns to normal.

Draw Handler Usage:



```
def draw_handler_function():

        #Do something

bpy.types.SpaceView3D.draw_handler_add
(disp_vco_list, (bpy.context, None), 'WINDOW', 'POST_PIXEL')
```

Figure 13: Display Vertex number using handlers

The figure above displays vertex coordinates and indices drawn on the viewport using a draw handler.

3.6 Dictionary Parsers

To convert the data to a OpenFoam readable format we need to parse the data to a dictionary file.For that 2 types of parser has been used.

3.6.1 Blockmesh parser

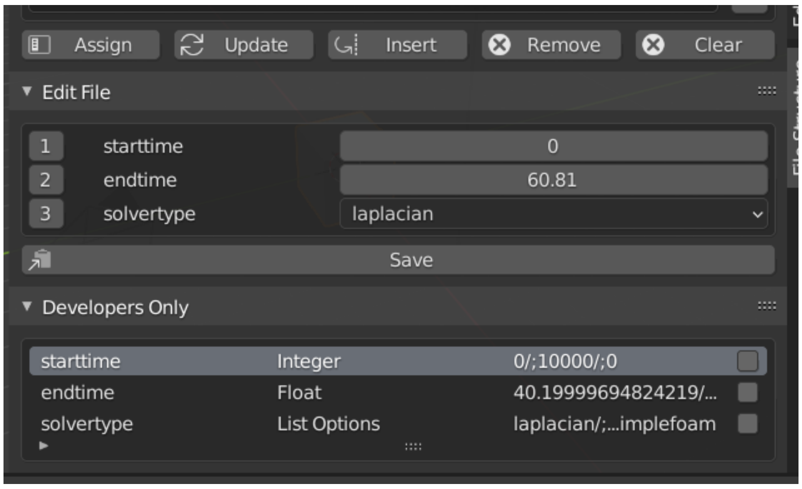
This is used in the Blockmesh add-on, when clicked on Add lists to dictionary, all the data present in the add-on is first saved into a JSON file which is stored as a data block inside blender's memory and from there is it parsed into the blockmesh dictionary in a OpenFOAM readable format.
JSON data block is used for the persistent storage of geometric data inside blenders memory as the geometric data could be required in some case file

3.6.2 Pyfoam ParsedParameterFile

This is used in the solver add in which utilizes Pyfoam's ParsedParameterFile to read and parse data into OpenFOAM readable dictionaries. It is also used to generate templates for case files that is used by the User Solver Add-on.

3.7 Solver Parameter Schema

To remove the need for drafting a schema file for every solver, the solver add-on uses dynamic memory to generate a schema within a UI List Table. As UI Lists are easily editable, they are used to preserve and edit schema information during an active instance of blender. The developers only panel in the solver add-on is a way for developers to view the schema in which file parameter information is stored. The solver add-on however doesn't allow users to view this panel. At a time only one solver can be viewed/edited. As Venturial uses a dynamic memory, schema information will be lost if not saved. If a saved solver is viewed, Venturial will read schema information to display solver parameters on the Solver add-on.


Assign | Update | Insert | Remove | Clear

Edit File

1	starttime	0
2	endtime	60.81
3	solvertype	laplacian

Save

Developers Only

starttime	Integer	0/;10000/;0
endtime	Float	40.19999694824219/...
solvertype	List Options	laplacian/;...implefoam

Figure 14: Parameters added in solver add-on

In figure 14, 3 parameters have been added using the solver add-on. The developers only panel displays the schema information in a UI List of 3 columns in the following format:

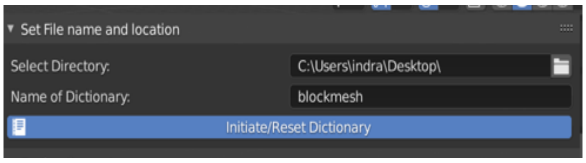
parameter name	parameter type	parameter properties

Parameter name, type and properties are added from the Edit Label Panel above the Edit File Panel.
Parameter properties are stored in the following manner:

integer	minimum/;maximum/;default
float	minimum/;maximum/;default
boolean	default
List Options	Option1/;Option2/;…… and so on

Chapter 4 Application Scenario Implementation

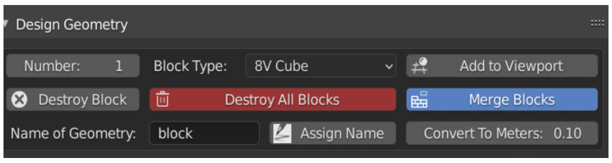
4.1 Blockmesh Add-on



Set File name and location

Select Directory:	C:\Users\indra\Desktop\
Name of Dictionary:	blockmesh

Initiate/Reset Dictionary



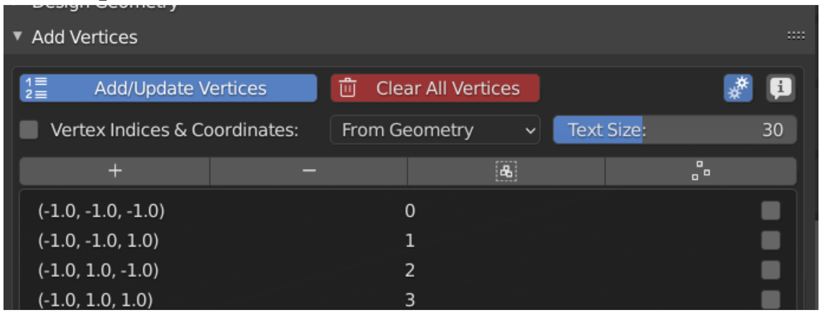
Design Geometry

Number: 1 | Block Type: 8V Cube | Add to Viewport

Destroy Block | Destroy All Blocks | Merge Blocks

Name of Geometry: block | Assign Name | Convert To Meters: 0.10

Step: 1 Mention Dictionary name and location. Step:2 Select Block type and click Add to Viewport

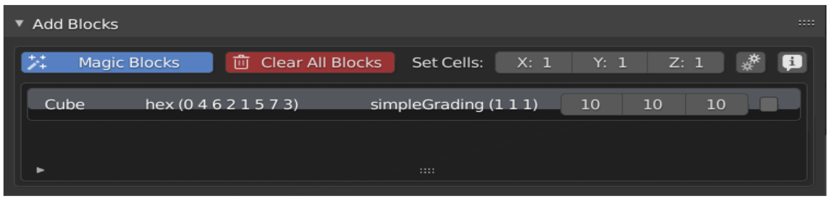


Add Vertices

Add/Update Vertices | Clear All Vertices

Vertex Indices & Coordinates: From Geometry | Text Size: 30

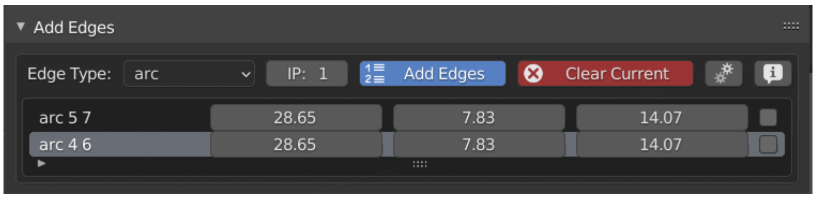
(-1.0, -1.0, -1.0)	0
(-1.0, -1.0, 1.0)	1
(-1.0, 1.0, -1.0)	2
(-1.0, 1.0, 1.0)	3



Add Blocks

Magic Blocks | Clear All Blocks | Set Cells: X: 1 Y: 1 Z: 1

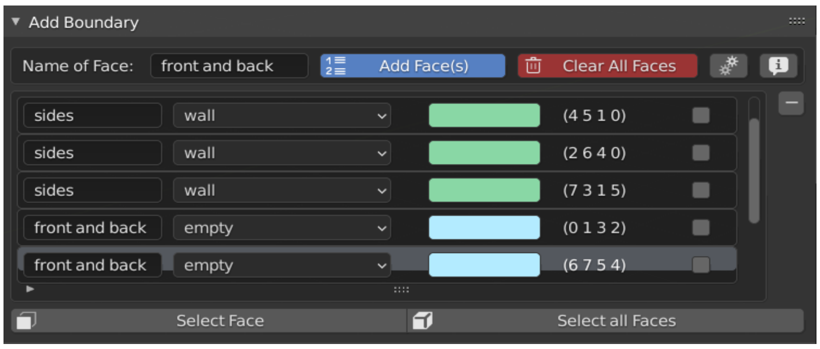
Cube	hex (0 4 6 2 1 5 7 3)	simpleGrading (1 1 1)	10	10	10



Add Edges

Edge Type: arc | IP: 1 | Add Edges | Clear Current

arc 5 7	28.65	7.83	14.07
arc 4 6	28.65	7.83	14.07

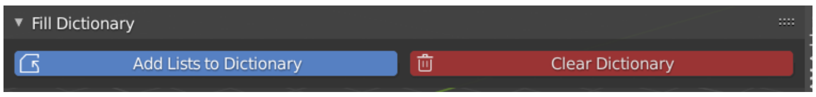


Add Boundary

Name of Face: front and back | Add Face(s) | Clear All Faces

sides	wall	(4 5 1 0)
sides	wall	(2 6 4 0)
sides	wall	(7 3 1 5)
front and back	empty	(0 1 3 2)
front and back	empty	(6 7 5 4)

Select Face | Select all Faces



Fill Dictionary

Add Lists to Dictionary | Clear Dictionary

Step 3: Name the block and click on Assign Name while the geometry is selected in the Viewport

Step 4: Click on Add/Update Vertices button to add all the vertices belonging to the block directly.

Step 5: Click on Merge Blocks while the geometry is selected. Click on Magic Blocks to view block information in Block Panel.

Step 6: Select an Edge type and number of interpolation points. Select any two vertices of an edge and click on the add edges button.

Step 7: Name faces or group of faces and click on Add Faces to append Face list.

Step 8: Select a face type from the drop down for each face or group of face.

Step 9: Once all blockmesh data is confirmed, click on Add Lists to Dictionary to create a new blockmesh or update blockmesh.

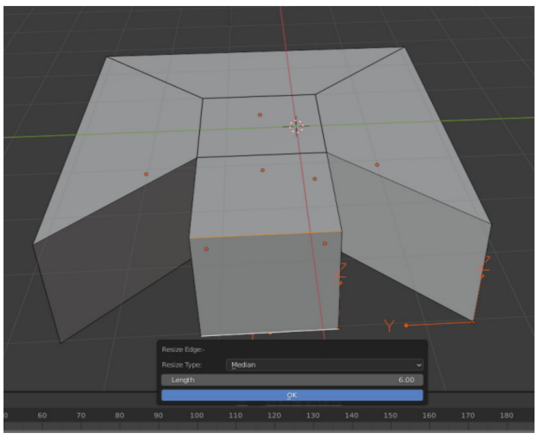
Figure 15: Blockmesh Step by Step usage

4.2 n-block Geometry (FOSSEE pipe tutorial)

Please visit this link to see a video tutorial on n-block geometry design on Blender and blockmesh file generation using blockmesh add-on.

4.2.1 n-block Geometry creation on Blender

To create a geometry with n number of blocks in blender the following steps can be used:
Step 1: With the add-on loaded and dictionary initialized, enter n in Number of blocks in the design geometry panel then click on add to viewport.
Step 2: The blocks should be visible on the viewport, now enable snapping and choose vertex mode in blender.
Step 3: Now the blocks can be arranged to block the geometry required. Once a block is brought close to another block it should snap together in place.
Step 4: To change the size of an edge press on TAB to go to the edit mode and click on the edge that needs to be resized then Press Shift +F on keyboard to bring up the edge resize menu. Now the edge length can be changed as required.
Step 5:Once the required geometry is made switch back to object mode and click on Merge blocks to merge all the blocks as one object.



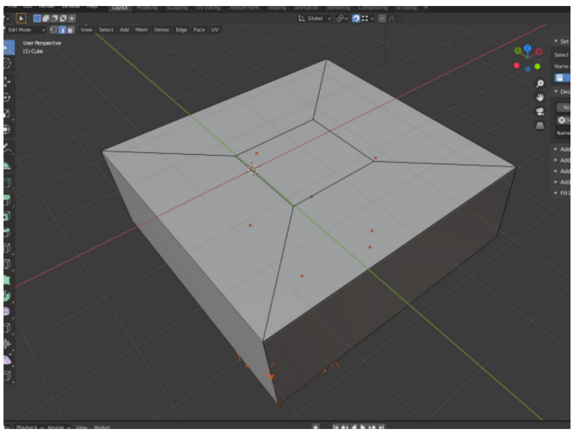


Figure 16: N-blocks geometry creation

To add vertices to blocks manually, enable advanced options in the add blocks section and then select the vertices in the order given in Figure 16 and then click on add block.

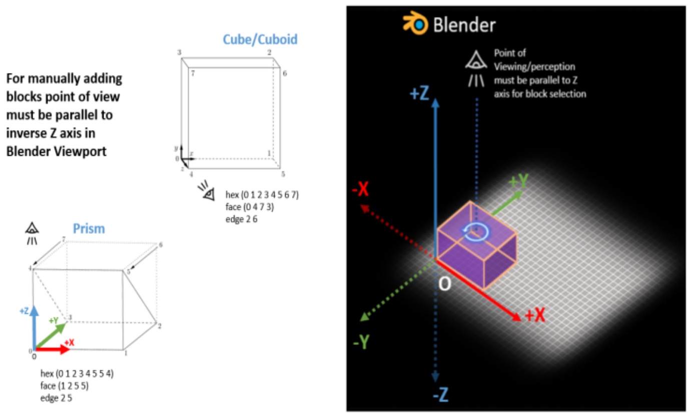
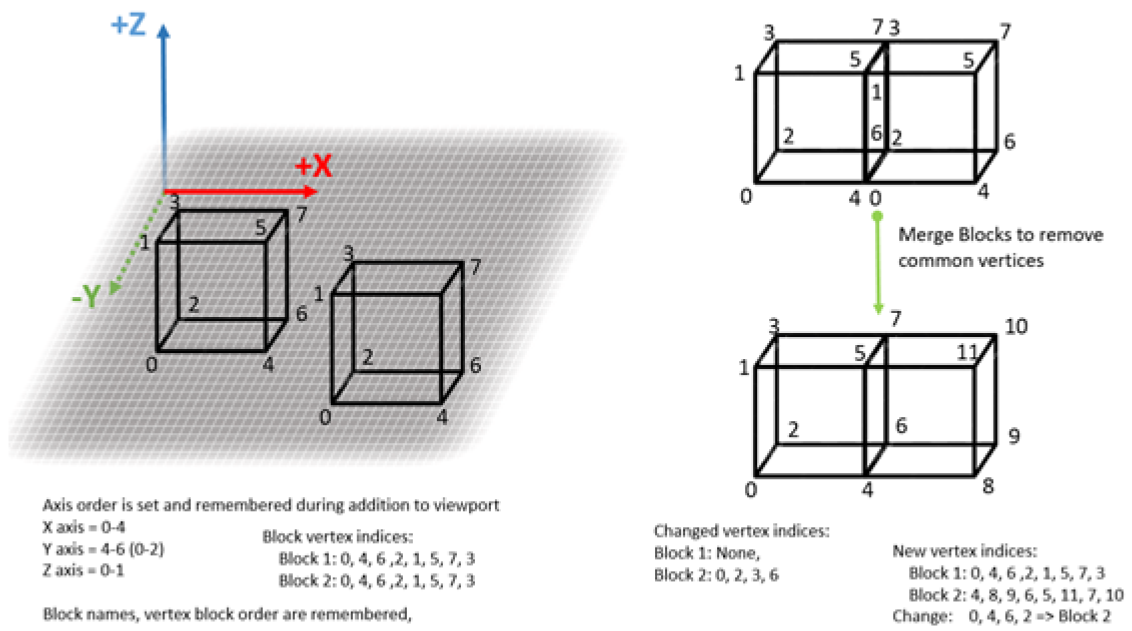

For manually adding blocks point of view must be parallel to inverse Z axis in Blender Viewport
Cube/Cuboid
hex (0 1 2 3 4 5 6 7)
face (0 4 7 3)
edge 2 6
Prism
hex (0 1 2 3 4 5 5 4)
face (1 2 5 5)
edge 2 5
Blender
Point of Viewing/perception must be parallel to Z axis for block selection
+Z
-Z
+X
-X
+Y
-Y
O

Figure 17 (a)



Axis order is set and remembered during addition to viewport
X axis = 0-4
Y axis = 4-6 (0-2)
Z axis = 0-1

Block vertex indices:
Block 1: 0, 4, 6 ,2, 1, 5, 7, 3
Block 2: 0, 4, 6 ,2, 1, 5, 7, 3

Block names, vertex block order are remembered,

Changed vertex indices:
Block 1: None,
Block 2: 0, 2, 3, 6

New vertex indices:
Block 1: 0, 4, 6 ,2, 1, 5, 7, 3
Block 2: 4, 8, 9, 6, 5, 11, 7, 10
Change: 0, 4, 6, 2 => Block 2

Figure 17a and 17b: Rules for numbering vertices in OpenFOAM

4.2.2 Dictionary Generation Steps

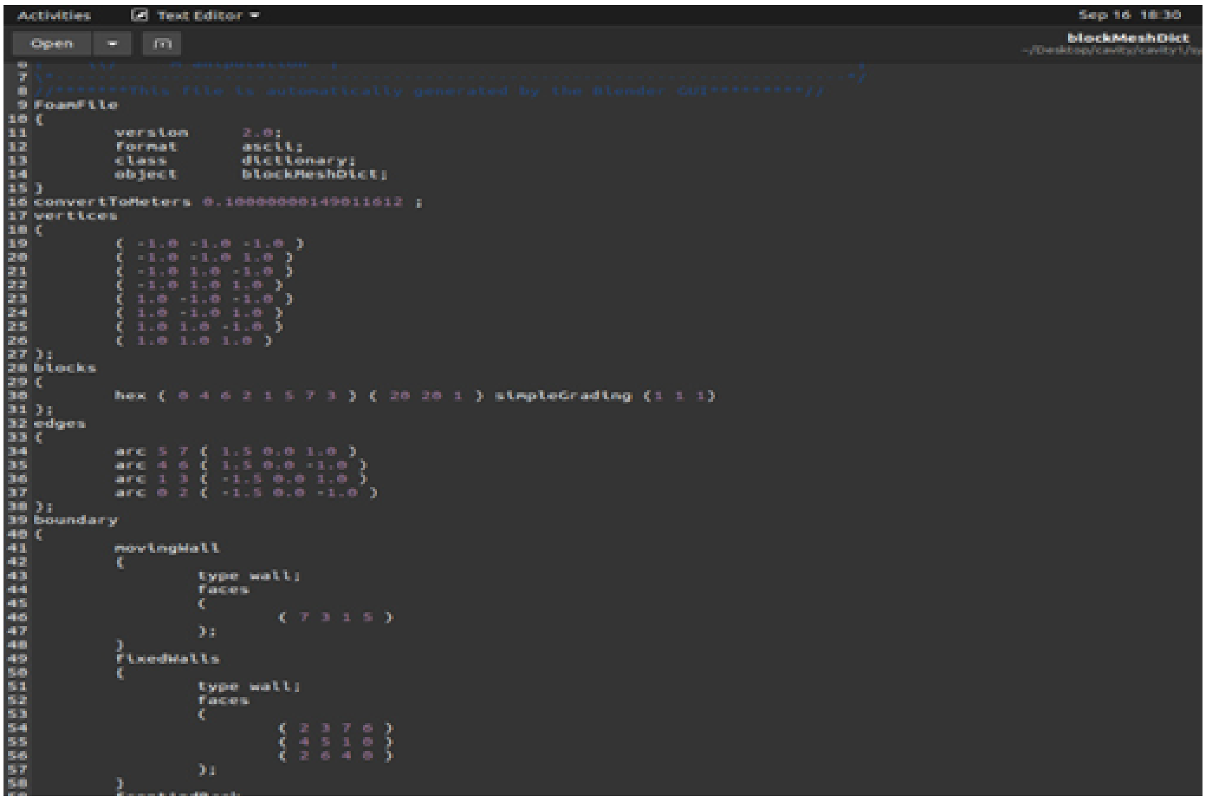
To create a new parameter file in developer Add-on (Developer Add-on):

- A basic template of a new parameter with a header is created and placed in the templates folder of the add-on.
- The templates folder of the User Add-on is chosen for the Case Directory.
- The template file which needs to be edited is selected.
- On clicking add Lists to Dictionary button, the dictionary is loaded in memory
- In the edit label section new parameters can be added.
- In the property type, the type of parameter needs to be selected and then the name of the parameter including some metadata is to be defined.
- Click on assign and it should be visible on the Edit File panel.
- New parameters can be added or old ones removed from this panel.
- Once all the parameters have been added to the Edit File section, Click on Save. This will add the template to the user add-on directory with all the required metadata.

To use the user add-on for changing values of parameters(User Add-on)-

- With the add-on installed, navigate to the add-on panel
- Select the directory where the simulation files are.
- Select the solver type and the dictionary which needs to be created.
- Click on Add Lists to Dictionary to load the parameters.

- The parameters will be visible on the edit file section. Change the values as required.
- Click on save to save the final dictionary to the case Directory.



```
Activities    Text Editor ▾                                              Sep 16  18:30
Open  ▾                                                                  blockMeshDict

\*---------------------------------------------------------------------------*/
//*******This file is automatically generated by the Blender GUI**********//
FoamFile
{
          version       2.0;
          format        ascii;
          class         dictionary;
          object        blockMeshDict;
}
convertToMeters 0.1000000000149011612 ;
vertices
(
          ( -1.0 -1.0 -1.0 )
          ( -1.0 -1.0 1.0 )
          ( -1.0 1.0 -1.0 )
          ( -1.0 1.0 1.0 )
          ( 1.0 -1.0 -1.0 )
          ( 1.0 -1.0 1.0 )
          ( 1.0 1.0 -1.0 )
          ( 1.0 1.0 1.0 )
);
blocks
(
          hex ( 0 4 6 2 1 5 7 3 ) ( 20 20 1 ) simpleGrading (1 1 1)
);
edges
(
          arc 5 7 ( 1.5 0.0 1.0 )
          arc 4 6 ( 1.5 0.0 -1.0 )
          arc 1 3 ( -1.5 0.0 1.0 )
          arc 0 2 ( -1.5 0.0 -1.0 )
);
boundary
(
          movingWall
          {
                    type wall;
                    faces
                    (
                              ( 7 3 1 5 )
                    );
          }
          fixedWalls
          {
                    type wall;
                    faces
                    (
                              ( 2 3 7 6 )
                              ( 4 5 1 0 )
                              ( 2 6 4 0 )
                    );
          }
```

Figure 18 Blockmesh Dictionary file

4.3 Results on Paraview

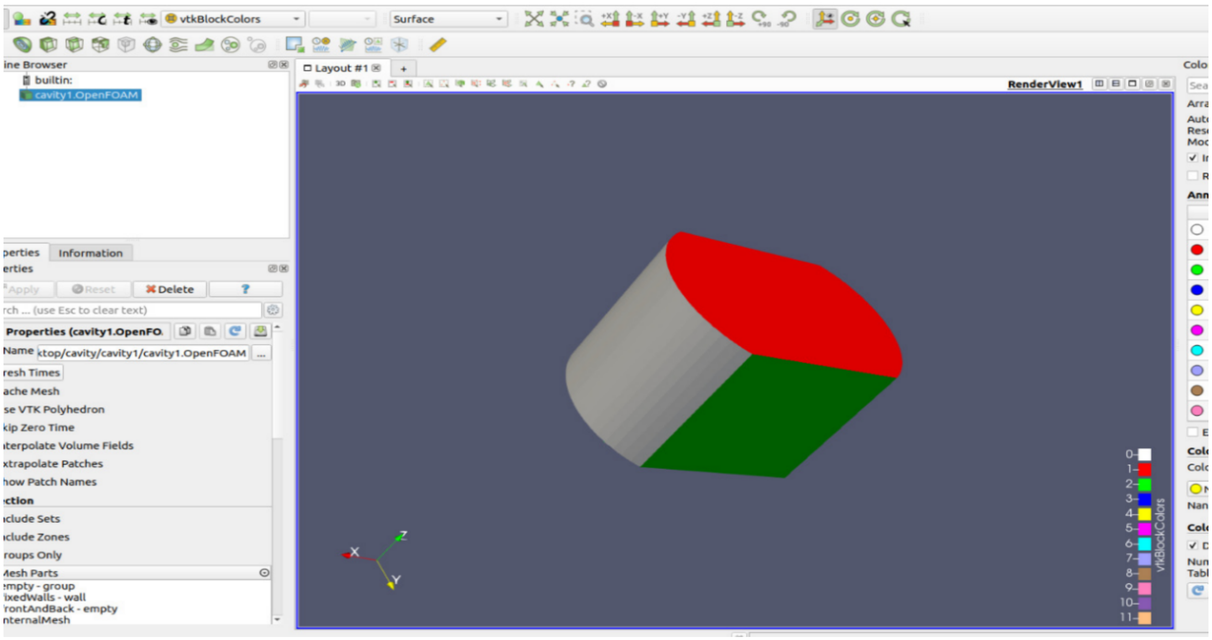


Figure 19: Geometry on Paraview

Chapter 5 Conclusion

5.1 Conclusion

Geometry design capabilities of Blender are used in conjunction with a GUI to integrate the process of geometry modelling and text file generation all within the same software with graphical visualization of geometry and a GUI to edit parameters removing the need to manually edit OpenFOAM files.

5.2 In Development

Both the add-ons are fully functional to serve their purpose but more visualization capabilities, hotkeys for geometry modelling and automation features are under development to aid faster file generation without manually editing any text.

a) Edge Length Viewing Handler: Select an edge to view its length on Viewport by enabling a checkbox in Add Edge Panel.
b) Edge Length Text Size: Enlarge or reduce size of Edge length text.
c) Face Name Viewing Handler: View name of selected Face from Add Face panel on the Viewport.
d) Face Name Text Size: Enlarge or reduce size of Face name text.
e) Face Color Checkbox: Enable this checkbox to view geometry with colored faces. By default, colors are randomly assigned to the face/group of faces. A color template for each face/group of faces in the face list is already assigned. However, a checkbox to enable viewing colored faces on viewport is under development.
f) Collate Option for Edges: Collates all edge panels together as blockmesh mentions all types of edges within the geometry under the same section.
g) Help Box: A dialog box appears when the Help button is clicked beside each Panel. Information within this dialog box provides documentation on usage of Panel features.
h) Editing of Geometry after Build: The Merge Block feature builds geometry by removing common vertices, edges and faces without remembering them. Hence blocks must be merged only when geometry is finalized. Amending this stipulation is currently under development and a revision of Merge Blocks will enable editing of geometry even after blocks are merged.
i) Free motion of geometry: Rotation and Translation of Geometry is disabled and viewing is done by changing user view perspective.
j) Hotkeys for Geometry Editing: Currently blockmesh add-on consists of only one hotkey built with a Keymap for editing the length of an edge. More hotkeys are under development to aid editing of geometry.

k) Vertex Snapping: Currently, users can snap blocks to nearby vertices by enabling snap by vertex feature which is in-built in Blender. This requires the user to remain informed about the snapping feature without which joining blocks isn't convenient. A one-click snapping feature within the add-on can help users identify this feature as it can be documented as well.

l) Mergepatchpairs: blockmesh add-on in its current state doesn't support flow problems that require mergepatchpairs.

m) Singular Add-on Structure: The blockmesh add-on and solver add-on are intended to be merged in later versions and are currently separate for ease of development.

n) Advanced Set Cells: Within the meshing rules two blocks sharing a common edge must have the same number of cells along the common edge. Venturial doesn't identify if two blocks share one or more than one edge. Hence, the user has to be aware of OpenFOAM set cells rules. Identification of blocks sharing at least one edge is currently being developed. Number of cells along a particular axis for two blocks sharing at least one common edge will automatically be equalized.

o) Assigning Nested Parameters: Some OpenFOAM files have one or more parameters nested within a single parameter. Solver add-on has an Assign button which can assign only single-line parameters upon button click. Thus Venturial can generate files like control parameter files easily. Addressing Nested Parameters is integral to the generation of discretisation schemes file, equation solvers file etc. and is currently our top priority in the development process.

p) Time Directory Files (Velocity Field, Pressure Field etc.): Some time directory files acquire information from blockmesh and data can either be initial values or boundary conditions which a user has to specify. A panel for these files is also a prioritized task under development.

q) Error Checklist for Solver Add-on will be assigned based on inputs from the CFD team to prevent errors during the solving process.

5.3 Future Scope

The Reynolds Add-on has a User interface for both blockmesh and snappyhexmesh utilities but provides support to solve a single fluid flow problem using blockmesh. Venturial doesn't support snappyhexmesh utility but envisions to provide users with full support to blockmesh for all levels of skill. As Venturial is yet to be released as a stable version, major ensuing tasks involve repetitive testing and revision based on direct feedback from CFD Team. Further developmental tasks are to present blockmesh or snappyhexmesh as an option for utility choice. As Venturial is a project under FOSSEE, it is built with the intention to help promote usage in academia and research. For this purpose, detailed standard operating procedure, tutorial videos, version control and documentation will be made available for the users.

5.4 Developer’s Note

The development team of FOSSEE Blender and Python Interns are always available for support. As Venturial is open-source community-centric software we welcome feedback from our users and work backwards to cater to their needs. This process of development has been followed since the very beginning of the project. We will also be delighted to know if Venturial has aided the learning process of individuals novice to CFD.
Lastly, we ardently hope that Venturial can become a reliable alternative to aid in performing OpenFOAM CFD simulations.

References

CFD FOSSEE	https://cfd.fossee.in/home
Blender	https://www.blender.org/
OpenFoam	https://openfoam.org/
Reynolds-Blender	https://github.com/dmsurti/reynolds-blender
SwiftBlocks	https://swiftblock.readthedocs.io/en/latest/swift.html
Vertex Ordering Blockmesh	https://www.openfoam.com/documentation/user-guide/4-mesh-generation-and-conversion/4.3-mesh-generation-with-the-blockmesh-utility
PyFoam	https://pypi.org/project/PyFoam/
N-Block geometry using Venturial	https://drive.google.com/file/d/1D_hBFSV0kfUIK4hkwhZNI3ppKOatZY0_/view?usp=sharing