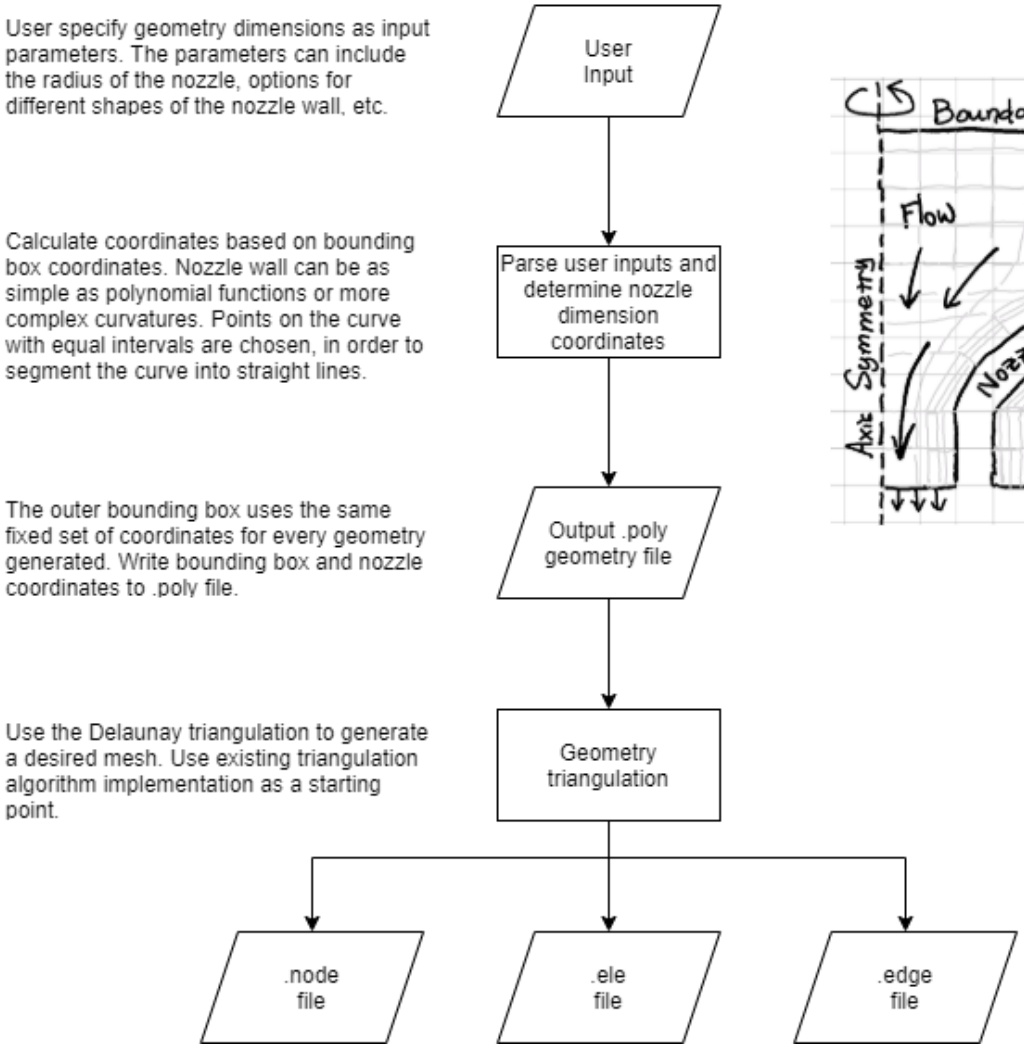
Component Design

Team: nullPtr
Team member: Allen Jiang, Bryant Porras, Kuanren Qian, Shu You, Vimalesh Vasu

Component 1: Auto Geometry/Mesh Generation (Shu You)

Given that the geometry is symmetrical, we will use 2D mesh with a specified axis of symmetry.



Reference Delaunay triangulation implementation:

[Triangle: A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator (cmu.edu)]

.poly file (raw geometry) format specification: [.poly]

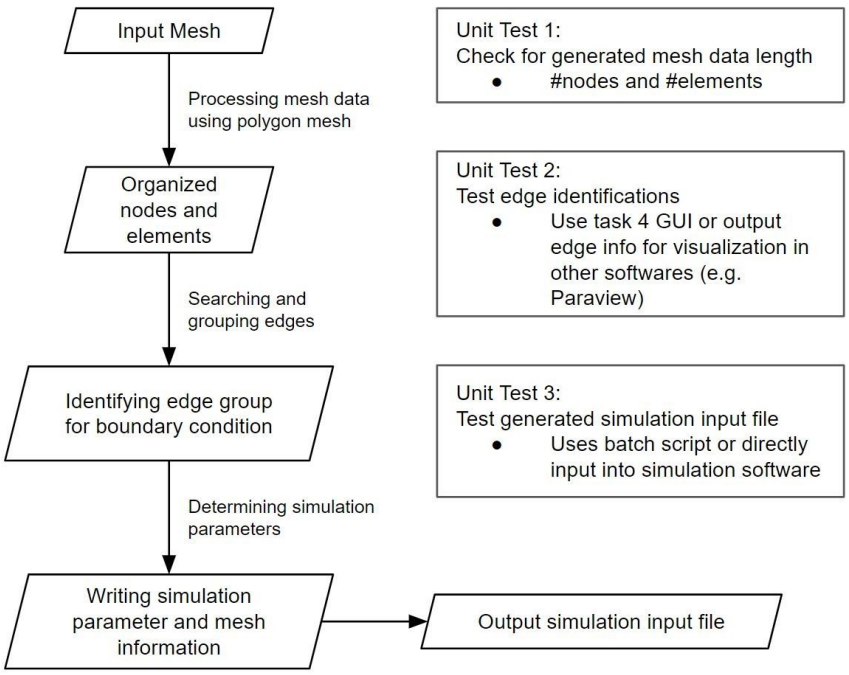
Mesh file format specification: .node .ele .edge

Unit Test

Visualize generated mesh file for manual inspection.

Component 2. Generating Simulation Input Files (Bryant Porras, Kuanren Qian)

Taking our mesh-generated file from component 1, our program will: (1) read the mesh into a correct data format (such as polygon mesh class) for faster processing of nodes and element connectivity information (i.e. format like <shape #> <node>…); (2) once mesh information is properly imported, based on the specific simulation conditions, our program will search and identify edge groups to assign boundary conditions; (3) the program will determine different sets of simulation parameters and pair with corresponding mesh data; (4) once mesh and simulation parameter are generated, the program will write and store simulation input files for each simulation case in the correct format for simulation software to import.



We plan to test this by making a few custom mesh-generated files of the file format similar to the example shown above, implement our algorithm to create the simulation input files, and to have this component be tested and relied solely on itself and not other components, we can test our simulation input files one by one by feeding them into our simulation software (most likely, ANSYS Fluent). Our individual unit tests come from how we will create each simulation input file, and we will run our simulation software on at least 3 simulation input files of different parameter combinations to ensure our component performs as expected.

Unit Tests: We will also develop 3 unit tests to ensure the data integrity of the mesh and test the simulation input file format.

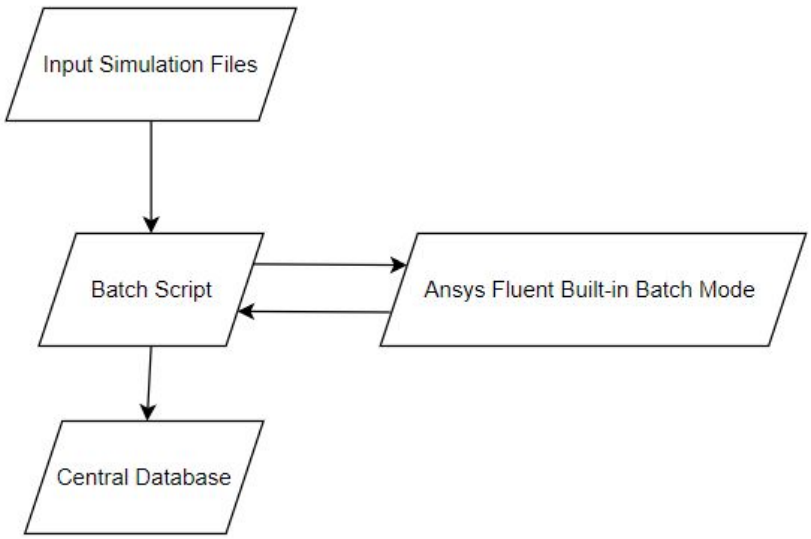
1. The first unit test will check the processed mesh nodes and elements length, and compare them with input mesh to verify the correct processing of mesh information.
2. The second unit test will output identified edge information for visualization in software such as Paraview for verification.
3. The third unit test will use commercial software to run a generated simulation file with simple mesh and simulation parameters to verify the correct simulation input file format.

Component 3: Batch Simulations (Vimalesh Vasu)

Takes in a set of simulation input files (ideally stored in one folder), and runs a batch script via the simulation software (ANSYS Fluent) to obtain all the results per simulation input file.

ANSYS Fluent (and most other simulation software packages) has a built-in batch mode function so the challenge lies in communicating with this software via C++ and generating the results as we would like to display them. For example, for the vacuum nozzle problem, we may want to specifically target the velocity profile of the nozzle after the simulation, and highlight these results as our output.

Our unit tests will have us take in at least 3 different sets of simulation input files (potentially each set created from a different mesh) and obtain our results from running our batch simulation and store them in a folder.



Component 4: GUI (Allen Jiang)

The GUI determines how the program will look like, how the user will input and process the data, and how the final results are being presented. The GUI should help the user carry out each component task easily.

For example, in Task 1, the GUI can include text fields that instruct users to fill in geometry dimensions / input parameters at the upper left, a ‘Run’ button, showing info such as ‘Calculate Coordinates’, at the left that starts the coordinate calculation if pushed, and the coordinates can be displayed at the bottom left. At corners, there can be information such as ‘Coordinates being calculated’, or ‘.poly files being generated’. The generated mesh file, if valid, can be visualized at the right side of the GUI, with a ‘Proceed to the next task’ button, so a new GUI for the next task -- very likely with a similar design -- can be displayed. The final output results can be highlighted at the center of the screen. Some drafts for the GUI and the flowchart are presented below.