# Analysis

## Problem Identification

Currently, there is a lack of truly free CFD (Computational Fluid Dynamics) softwares. Most programs require a subscription to use, and those that advertise themselves as free provide very limited usage without paying. This is a problem for the amateur engineer – something that is becoming more and more common with the advances in 3D printing technology. A common project that many of these people undergo is the making of a remote controlled drone or airplane. These people are likely to want to be able to run CFD simulations on their parts to find out how they will perform in the real world. However, due to the lack of free solutions, this is not something they are necessarily able to do.

This problem lends itself to a computational solution as there is no realistic way to do comprehensive fluid dynamics testing in the real world. Without expending a prohibitively large amount of resources and money (i.e., hiring a wind tunnel, measurement equipment), the task is inachievable. The digital environment allows for total control of the simulation and fast iteration of scenarios. Additionally, it is not possible to do the fluid dynamics calculations by hand, as they are far too complex and would require likely hundreds of hours of calculation by hand.

## Stakeholders

The stakeholders of this program would be engineering hobbyists and companies that provide a physical product.

The individual hobbyists would likely make occasional, infrequent use of the program – to check and inform their designs for any projects where fluid dynamics are important. It is not necessarily likely that hobbyists will often use it, as not all projects will require fluid dynamics simulation. This is particularly why the program would be of interest to hobbyists. Because it is unlikely it will regularly be used, it would not be convenient to pay for a licence to use a commercial software and so would much prefer to use a free alternative – the proposed program.

Companies that provide products would be much more likely to make extensive use of the program, especially in the design and pre-production of a product. The program would inform iterative development of products, allowing for producers to design products that perform optimally without incurring any extra cost. The cost for commercial usage of CFD programs is generally large, so a free alternative that performs equal to established others would be highly preferable.

## Research

TODO: Autodesk CFD, simFlow

## Essential Features

The final program must be able to complete a fluid simulation of arbitrary length, *n* iterations. The user should not be limited to a fixed timescale for their solution, and it would be actively unhelpful to restrict the user in such a way. The user should be able to set up their own simulation using their own models and flow characteristics and run it without issue. If this were not the case then the whole simulation software would be redundant – it would be totally ineffective as an analysis tool if it cannot be used to analyse different conditions. Additionally, it should be well documented, such that starting out with the program is not needlessly confusing, making for a steep learning curve. Else, adoption for the solution would be poor and those who do choose to use it may be unable to utilize it to its best ability.

## Limitations of the Proposed Solution

The proposed solution is not suitable for large models, like in the case of climate modelling. This is because models like this are designed to run on large supercomputers, with hundreds of gigabytes of available memory. The proposed solution is a general solution to the problem of fluid dynamics simulation on a given object. The problem of climate modelling is a highly specific, highly intensive case where various different forces and local conditions vary and change during the course of the simulation. This is far better suited to a custom approach, and therefore will not be catered to in the proposed solution.

Additionally, the accuracy of the simulation is governed by the length of each timestep (how far ahead in time to skip for each iteration). Because the environment is modelled using differential equations and the equations used to model it are accurate to the physical world, the only limiting factor for the accuracy of the simulation is the length of the timestep. In a mathematical context it makes sense for time to be considered continuous – not discrete. However, this causes the problem to be incomputable, so we must discretize time. We can minimize the effect this has on the simulation by choosing as small a timestep as feasible. However, choosing such small timesteps causes the simulation time to increase. There has to be a comprimise made with respect to accuracy and time.

Finally, the time taken for a simulation to run is largely dependent on the hardware at hand. Fluid simulations are not suited to being run on a regular home desktop or laptop. Devices such as these would take far too long to compute solutions and may even get uncomfortably hot in the case of a laptop. This can only be helped as far as writing efficient code, and the onus is on the end-user to ensure their hardware is capable.

## Hardware Requirements

This solution will not require any special hardware. It would be preferable to have a high-performance GPU available on the machine as this would allow for faster computation, taking advantage of the many thousands of cores a standard GPU has. However, this would not be strictly necessary to run the simulations. Additionally, it may even be possible to write the program in such a way that it could be accessed from a command line, such that simulations can be run on a headless machine. This would be done by passing a simulation file as a command argument.

## Measureable Success Criteria

|  |  |
| --- | --- |
| Criteria | Evidence |
| Simple user interface | Screenshot of the user interface |
| Ability to save and load a simulation file | Screenshot of save/load interface |
| Easy to understand documentation | Screenshot of simple documentation |
| Comprehensive documentation | Screenshot of all topics covered by documentation |
| Ability to import objects from files | Screenshot of load object interface |
| Ability to control flow characteristics | Screenshot of flow characteristics menu |
| Capability to utilize GPU-compute instead of CPU-compute | Screenshot of settings, showing selection of GPU or CPU |
| Moveable camera in workspace | Multiple screenshots showing camera in different positions |
|  |  |
|  |  |

# Design

## Problem Decomposition

Here problem decomposition is represented by a tree diagram:

### 

The whole program can be split into two larger portions – what the user will interact with, the UI, and what the user will not have to interact with, the simulation itself.

### UI

There will be two different major user interfaces, being the menu that the user sees when they first open the program and the workspace, where the user will spend most of their time working on the simulation before running it.

#### Menu

The design I have decided on for the menu is simplistic, and can be seen below:

Simulation Example

Picture

Preferences

New

This simple design avoids confusing the user with unnecessary clutter or options. The user has the choice to either:

* Create a new simulation workspace
* Continue editing a simulation workspace
* Edit the software preferences and/or settings

This will require that I develop a graphics pipeline, which will need to be capable of rendering both 2D and 3D graphics, 3D being for the workspace itself. It will also require an input system to detect the user clicking on buttons.

#### Workspace

Figuring out exactly what the workspace should look like took a long time, and there will likely be revisions during the development process as I begin to use the interface myself, but this is the design I have currently:

Object

Properties

File

Add FlowAdd MeshRun Simulation

This design is liable to change and is likely to become somewhat more complex over time, but I have tried to keep it simple here. It offers the basic functionality that I can forsee currently. Users will want to, and will be able to:

* Save the current file, open a different file, or create a new file
* Add flow objects
* Add mesh objects (effectors)
* Change object properties after selecting them in the 3D workspace
* Run the simulation itself

This will require the 3D and 2D graphics pipeline that was developed for the Menu, as well as the input system that was developed for the Menu.

The workspace will also require a 3D camera that the user can move around using their mouse and the capability to load objects from files.

### Simulation

The simulation itself is split broadly into two parts – preprocessing of meshes and flows, and the flow calculations themselves.

#### Preprocessing