|  |  |  |
| --- | --- | --- |
| CSC 436  Due 10/31/17 | **Status Report 2** |  |

OpenBurn

* *Team #12: Team Rocket*
* Daniel Tranfaglia, Abhishek Rane, Isaac Plunkett, Vicente Figueroa, Andrew Tarr

1. Introduction

* First Iteration: Command-line interface that prompted user for data regarding the propellant density, change in time, grains, and nozzle. Ran simulation calculations to produce results such as thrust vs. time and pressure vs. time, which was written to a CSV file.
* Overview of Changes: Changed the graph library originally planned (used JavaFX ScatterChart instead of JFreeChart), some team members worked on the GUI, which was different than what they were originally supposed to work on. Some roles were changed a little to compensate for time, and more work was added on to some members than originally planned.

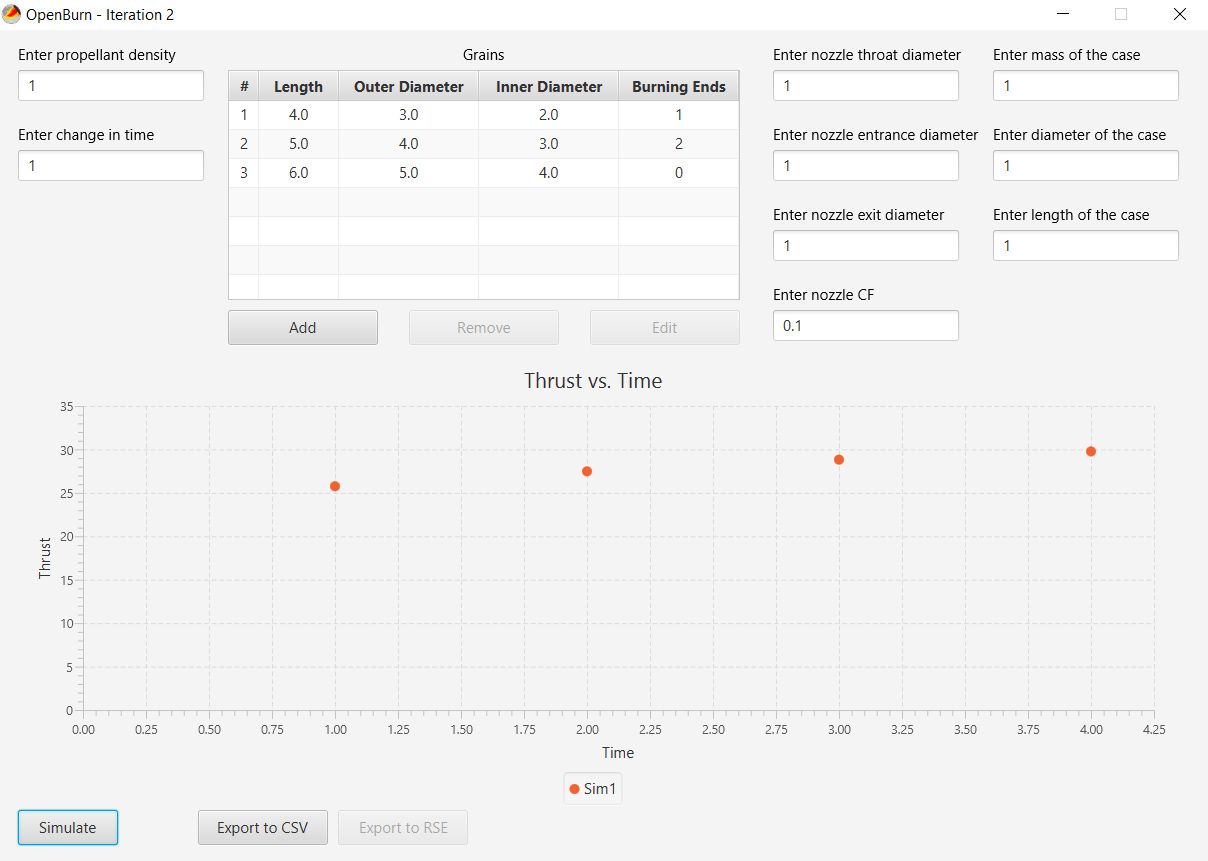
1. Customer Value

* Problem Definition: Multi-platform, open-source software, introduces multiple unit availability, accounts for erosive burning, improved Graphical User Interface.
* Customer Value: University of Arizona IREC Team, rocket enthusiasts.
* Changes from Proposal
  + 10/17/17 - No longer need to account for erosive burning.
  + The customer said we no longer need to worry about this, since they are changing the way they are building the rocket.

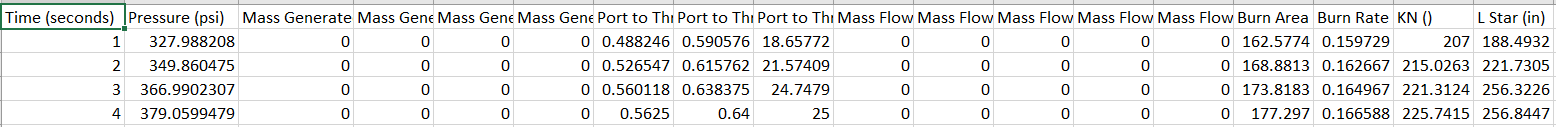
1. Technology

System at the end of this iteration

* Goals for this iteration: This iteration, our goals were to make sure that we had a working GUI that showcased the proper functionality of our App. This consisted of making most of the GUI and adding our back-end implementation to the GUI, while keeping the CSV output feature.
* What works: We got the main functionality of the GUI working, by linking the back end to the front end. This allows our app to be more user friendly and puts us one step closer to the final product. CSV output is an optional feature that can be run by clicking a button after a simulation has run.



CSV file from the above simulation, named “Sim1.csv”

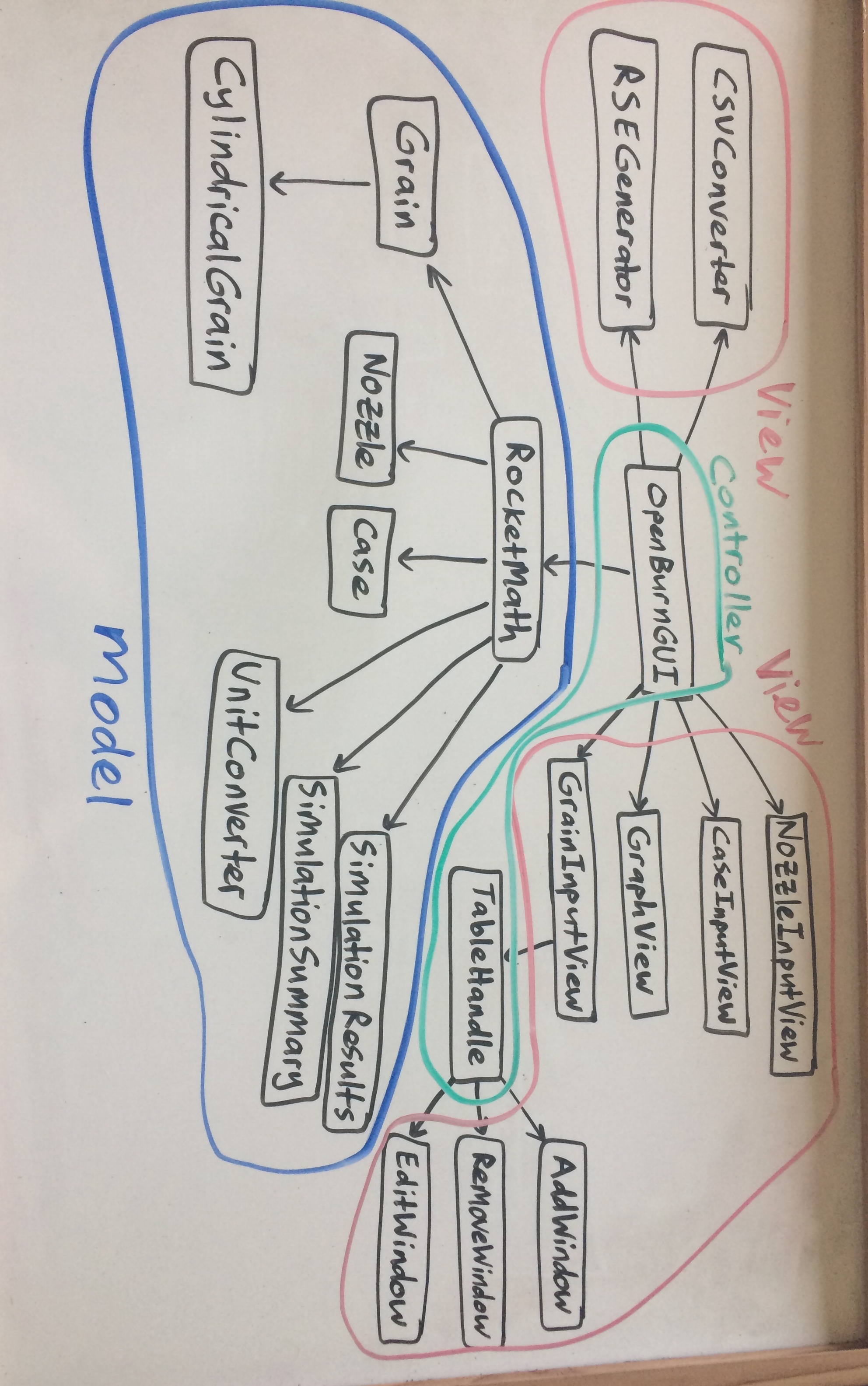


* Tests run: We have tested the output to our graph of the results that would were produced on Google, and turns out that our app is more accurate in terms of decimals. We also tested the GUI and rocket math to make sure all the calculations are correct.

Module Hierarchy and Module Guide

Follow the Module Hierarchy and Module Guide Example for the Floating Weather Station in the Architecture slides.

Provide as many levels in the Hierarchy as are needed. For example, if your project has Client and a Server, provide additional levels of details for the Client and Server.



Architecture of the system

Description:

Model - As in the previous iteration, our model still has a Grain Hierarchy to potentially allow for multiple types of grains, and a nozzle class, as well as the RocketMath class to run simulation calculations and produce a set of SimulationResults. We now have a Case class to represent the case of a motor, which goes into calculations. We also have a UnitConverter class and SimulationSummary to eventually provide details of a simulation on the GUI.

View - With the implementation of a GUI, multiple view classes were needed, including a GrainInputView which handles the grain table with the help of AddWindow, RemoveWindow, and EditWindow. We also have input views for Nozzle and Case, as well as GraphView, which is a wrapper class for the JavaFX ScatterChart. CSVConverter is still implemented as an optional feature, and RSEGenerator will be used to produce RSE files for simulation data in future iterations.

Controller - The command-line interface has been replaced with a GUI, which serves as the main controller for interaction between the user and the back-end. TableHandle was created as a means of communicating and transferring data between

1. Project Management

* Plan for the next iteration and the rest of the semester: For the next iteration, we plan to clean up the GUI, and have more features by adding ways to export and import data. We also have a logo that we plan to add for better presentation, for now we are using a first draft of the logo. We are also planning on implementing more types of grains, and allowing the user to switch unit systems.
* Track changes to design:
  + 10/23/17 - We added sub-packages and moved modules around to help organize our system, and added more files to break up the work. More of the changes to our code was refactoring and commenting.

1. Team

* Abhishek (Product Manager):
* Isaac (Front-End): Implemented Unit Converter. Wrote tests to verify to loss of precision when converting metric to imperial and back to metric. Added functionality to the csv file button on the GUI.
* Daniel (QA Engineer): Implemented graph feature, as well as the GUI mapping of where components should be placed. Produced testing for RocketMath, which completes testing of individual back-end modules. Monitored GitHub repository.
* Andrew (Back-End): Developed the components on the GUI to create grains, and make GUI easier to use.
* Vicente (Team Coordinator): Ran the Trello boards, kept up with tasks, and set up weekly meeting for the group. I also transposed the RSEGenerator file, and tested the main functionality for the GUI.

1. Reflection

* What went well: We once again had very good communication among our teammates, and never had an instance where people had to do unnecessary work or two people worked on the same thing.
* What didn’t go well: We had to be a bit rushed in our 2nd iteration because we weren’t given much time from the first iteration. But we were able to stay on track by redistributing work as needed.
* For the features that were not implemented, what were the issues: We got rid or erosive burning due to the functionality no longer being needed. All other 2nd iteration goals were met.
* How we plan to overcome the issues we encountered: We planned to shift the focus of certain tasks to the more important/difficult ones, allowing for more collaboration on certain issues.
* If you are redefining the problem, what is the change: We are not redefining the problem other than removing the erosive burning feature because the customer no longer requires this feature since they are designing their rockets differently.
* What we will do differently for the next iteration: Next iteration we plan to stay more on track when it comes to the due dates, and not make the crunch time at the end.