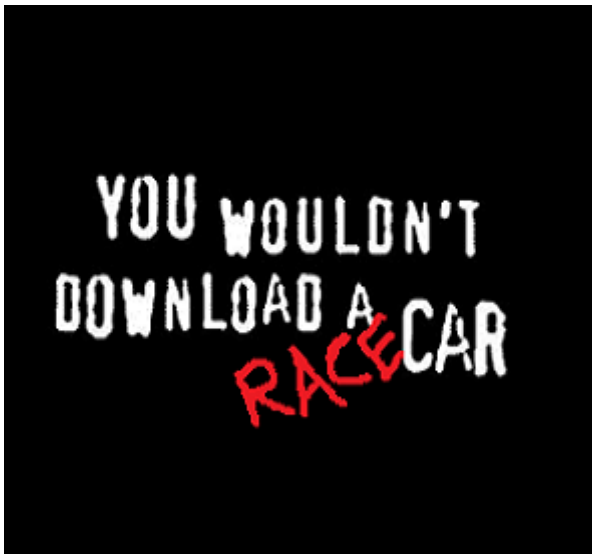
COLORADO SCHOOL OF MINES
EARTH • ENERGY • ENVIRONMENT

CSCI 370 Final Report

You Wouldn't Download A Race Car

Nicholas Estoll
Caleb Edwards
Brianna Dalton
Jasmine McCrary

Revised June 15, 2022



CSCI 370 Summer 2022

Prof. Kathleen Kelly

Table 1: Revision History

Revision	Date	Comments
New	May 17, 2022	Completed Sections: I. Introduction II. Functional Requirements III. Non-functional Requirements IV. Risks V. Definition of Done Appendix A – Key Terms
Rev – 2	May 23, 2022	Started Sections: VI. System Architecture VII. Software Tests and Quality
Rev – 3	June 10, 2022	Completed Sections: VIII. Project Ethical Consideration Started Sections: IX. Project Completion Status X. Future work XI. Lesson learned XII. Team Profile References
Rev - 4	June 13, 2022	Completed Sections: IX. Project Completion Status X. Future work XI. Lesson learned XII. Team Profile References

Table of Contents

I. Introduction

The Mines Formula SAE team is a student organization at Colorado School of Mines with members from all majors and all with different educational experiences ranging from first-year undergraduate students to students pursuing a graduate degree. The team is responsible for designing, building, testing, and driving a car each year in the Formula SAE competition. Through this process, the team has collected copious amounts of data and would like to use that data to create a digital model of the vehicle to test changes to the vehicle and provide practice to the driver. The ability to import specific data and have a virtual car built using it would allow the team to simulate changes before they consume resources to modify the vehicle, resources to reserve a time at the track, and time transporting the vehicle, driving it, and then evaluating what happened. Thus, for small changes such as the final drive ratio or a small change in suspension, insightful data can be gained via simulation as opposed to a full-fledged test.

Our job, as given by the client, was to use the data retrieved from testing the car and create a user interface (UI) that will take the data and import it as a digital model of the car in the driving game Assetto Corsa. We built upon a 3-D model of the formula car that was created previously by a team member so that we did not have to focus on modeling the car and instead could focus on the software that would best fit our client's needs. In the initial state, the software was based on an older model of the vehicle and did not capture enough real-world characteristics for the team to be able to use it to test changes. The model does not allow for sudden changes to the parameters such as suspension, powertrain, and aero, therefore making it unable to meet the needs of our client in its current state.

II. Functional Requirements

After meeting with the client, the functional requirements of the software were made clear. These requirements are critical in meeting our client's needs and ensuring that the software created is what they desire. The software created must be able to produce a GUI-based tool that will have the user specify all necessary input files. Upon submitting the files, a model of the car is generated that can be drivable in Assetto Corsa. The input files are .csv files that contain data for the powertrain and the suspension parameters. The software must be able to read the files and output the appropriate .ini files used in the Assetto Corsa configuration. A configuration file for the program should be loaded automatically to specify options outside of the scope of uploaded .csv files and should be easily modifiable by the Formula Team members to change as needed. The solution needs to be an executable .jar file, runnable on a popular system, assuming the appropriate version of Java is installed.

- Read in .csv input files for powertrain, aerodynamics, and suspension parameters and output appropriate Assetto Corsa configuration .ini files.
- A GUI-based tool has the user specify all necessary input files, and upon submission, a car model will be generated and should be drivable in-game.
- A configuration file for the program should be loaded automatically to specify options outside the scope of uploaded .csv files. It should be easily modifiable by the Formula Team members (with comments included in the configuration file specifying each parameter means).
- The solution should be compilable as an executable .jar file, runnable on popular systems assuming the appropriate version of Java is installed.

III. Non-Functional Requirements

The non-functional requirements were given by the client at the initial meeting. These requirements were implemented in the software to improve the quality and give our client the best possible outcome when combined with functional requirements. With the functional requirement of a GUI that will collect the files, it was important for the GUI to be easy to use. This was established as a non-functional requirement by the client, who wanted the interface to be simple for all members of the Formula SAE team to use without having prior computer science knowledge.

The non-functional requirement for this project is to specify real-world data into the outputted .ini files, which will correlate with the functional requirement of reading the powertrain data and suspension from .csv files. The two requirements work together to help produce the most realistic simulation in Assetto Corsa. With a realistic simulation produced by inputting the data, the client can view the impact of changes as close as possible to what happens in reality without driving the physical vehicle.

IV. Risks

Risks in the project are limited. Data corruption is an unlikely risk in the project. Game corruption is also an unlikely risk. For either one to occur, something would have to go wrong. If data or game corruption occurs, it will have a significant impact. The impact would extend the project completion timeline. Inaccurate design data could lead to corruption in-game files, leading to an unusable car in-game or a broken game installation. Modeling a car based on defects and representing that model as a real-life vehicle could lead to real-life conditions such as a crash as the driver would not have an accurate basis of reality when training in the simulation. The defects in suspension data can lead the car to tilt or pull to one side unexpectedly while the car is in turning. If the powertrain data were inaccurate or defective, the driver training would have an inaccurate understanding of the correspondence between gears and speed and may shift into a gear that's not ideal for achieving maximum acceleration. Defects on the game or the data could lead to grave consequences. Corruption of game files would require a reinstallation of the game. Corruption of development files could be a hindrance, so we have chosen to use a version control system to mitigate this risk. This allows us to save versions of code working when they work before adding more code. In the case of corruption, we can go back to previous versions without starting all over at the beginning.

V. Definition of Done

- List of minimal useful feature sets:
    a. Powertrain and Suspension data need to be uploaded and represented in in-game files
- Tests the client will run before accepting the finished software:
    a. Be able to open files (.ini) and hand check compared to design data (forces, CAD, etc.)
    b. The software creates a drivable car in Assetto Corsa
- How and when the software will be delivered:
    a. Take the entire project folder with all the code, zip it, and put it on the Formula server
    b. Anything needed to run the program will be installed on the computer running the driving sim
    c. Transfer GitHub repository to Mines Formula GitHub

VI. System Architecture

The software produces a GUI that takes in desired .csv files containing data. The data is converted to a .ini file outputted by the software and is taken in by the game Assetto Corsa. There are separate classes for UI, Data management, and File interface to perform this task. These classes work together to produce the required functionality. The GUI created is simple and user-friendly. This was a requirement given by the client. Classes utilized for the GUI are the Data Interface class, GUI class, File import, and Tab Menu class. Below are four images that demonstrate the architecture of our project.

Figures 1.1 and 1.2 show the GUI design we made for the project. It will have different tabs that will represent the design data. The first tab, called Basic, will have the vehicle's general information and specify the configuration file and show whether Assetto Corsa is present in the user's directory. In Figure 1.2, the GUI set-up shows the specific requirements for the Powertrain tab, including Gearing and Torque Curve data. The suspension tab will also have requirements, such as front, rear, and suspension vehicle data.

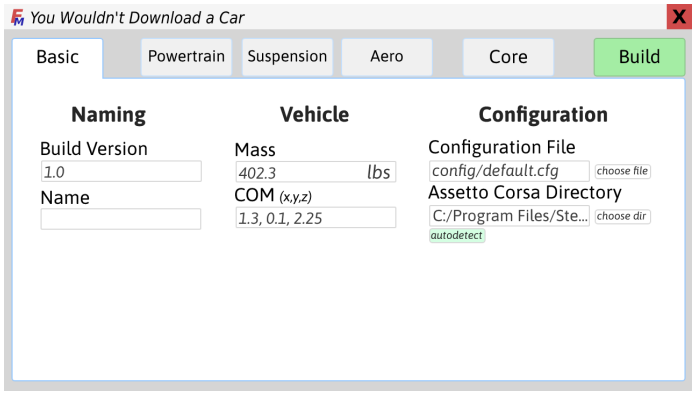


Fig 1.1 - GUI Mock-up

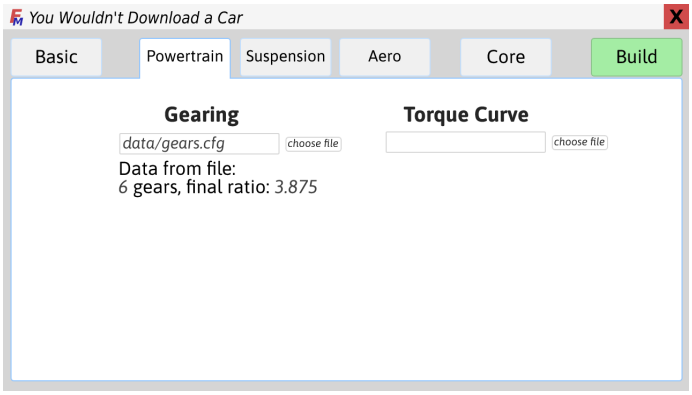


Fig 1.2 - GUI Mock-up for Powertrain tab

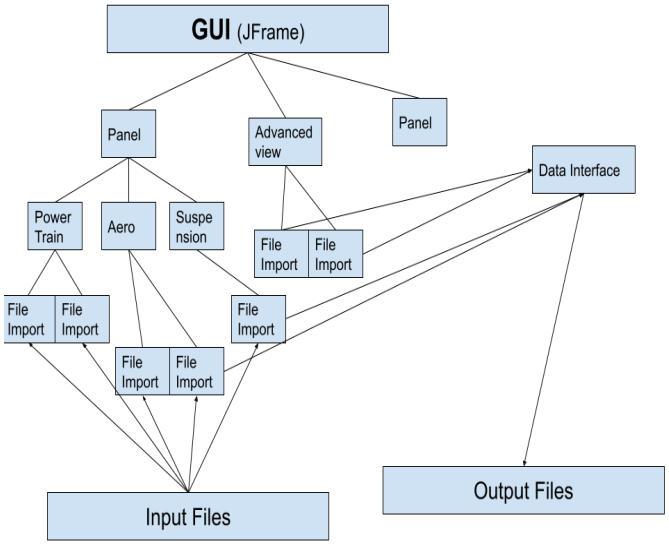


Fig 1.3 - Diagram demonstrating the connections

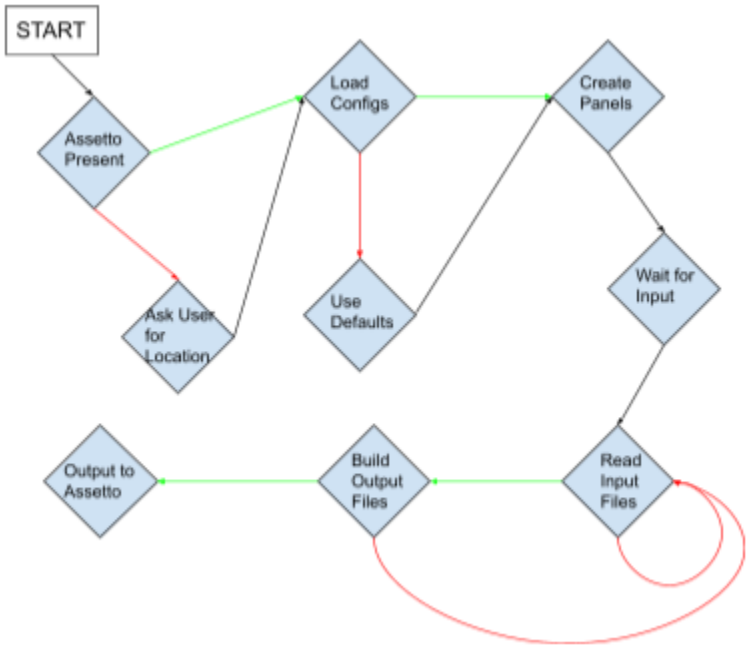


Fig 1.4 - Program Flow

Figure 1.3 shows the system architecture of our product is based on the GUI JFrame, which consists of different panels that focus on input .csv files and output .ini files. The three main design data, Powertrain, Suspension, and Aero, are divided into separate panels that will take input files. After the input files are taken, it is converted into .ini files and outputted to the game Assetto Corsa.

Figure 1.4 shows the program flow by checking whether Assetto Corsa is available and where it is located. It will load the configurations, and the default configurations will be used if the user doesn't make any changes. Each panel has various aspects of the design data that ask for user input. Once they are uploaded, the input files are read as .csv files. This creates the .ini files, which are outputted to the game. The green arrows seen in Figure 1.4 indicate the program's flow if the user has Assetto Corsa in their directory and goes straight to loading configurations and so forth. It shows a direct flow if all the input and output files are correct. The red lines indicate the flow depending on the user's response to the program. It depends on whether they need to re-upload different files to have different outcomes for the output files. It also checks whether the user has Assetto Corsa in the first place. The black arrows seen in Figure 1.4 indicates the program's flow after getting a user response.

VII. Software Test and Quality

Unit testing was integrated with our software through TDD, and as such, a framework that will verify many of the functional requirements already exists. We utilized three distinct categories of tests: data tests, GUI tests, and general tests, which test both the frontend and backend of the system, as well as their integration, as well as miscellaneous tests (like launching Assetto Corsa or using Content Manager to verify file structure). As this project has many requirements related to inputting and outputting files, we made classes abstract that away from the program's core logic. We also created unit tests for these classes to ensure that they worked as intended, even if things changed. Another core requirement was to ensure that the data was correctly parsed and modified to fit the format that Assetto Corsa needs to render the car in the game. We tested this in two different ways, firstly by ensuring the dummy data files used for unit testing were parsed correctly, and secondly by manually opening Assetto Corsa periodically and ensuring the car loads correctly and is drivable. The final requirement that the final deliverable should be an executable .jar file was tested manually by building a jar and running it in a "production" environment.

VIII. Project Ethical Considerations

The applicable ACM principles to our product are Principles 2.05, 3.08, and 6.08 [1]. Principle 2.05 states that any confidential information gained during professional work will stay confidential and such confidentiality will be consistent with the public interest and the law. Principle 3.08 states that there must be documentation of the specifications for the software and match with the users' requirements. Principle 6.08 states that the team should take responsibility for detecting, correcting, and reporting errors in the software and have appropriate documents related to it.

There are two ACM principles that we think are in danger of being violated: Principles 2.01 and 8.04. Principle 2.01 focuses on being honest and forthright about any limitations of their experience and education and providing service in areas of competence. Principle 8.04 focuses on improving how the team understands the software, the environment in which they will be used, and the related documents on which they work. We as a team had to learn about various aspects of the car and what kind of data is taken into consideration while building our product. Our team had little knowledge of the driving simulator and how it came into play with the future product.

We applied two tests articulated by Michael Davis to our product: the Common Practice test and Publicity test. We used the common practice test in a situation where the team members didn't respect one another and behaved in a discriminatory manner throughout the project. This would lead to the failure of completion of the product, and the team wouldn't be successful. The ACM principles violated in this scenario are Principles 1.3 and 1.4. Everyone in the team should be trustworthy and honest regarding any limitations that can arise while making the product. We used the publicity test in a situation where our product causes a crash or an injury that could potentially lead to legal issues and make the team seem as irresponsible. The ACM principle that is violated in this case is Principle 2.9. If the product is not finished and isn't designed to the required level, then it cannot be secure. The public wouldn't trust the team and lose credibility for their actions.

Our completed project will have a quality assurance plan to protect our stakeholders and ourselves. If the plan does not succeed, the output will not be of the highest quality. This makes it so that the completed work might not match the client's needs and can cause injury as our software is being used to test a car. It can violate many ethical principles if someone gets hurt while testing the car.

IX. Results

Summary of project completion:

- Features implemented and summary of feature performance
    a. Uploading powertrain data for torque and gearing
    b. Uploading suspension data for the overall vehicle and front/rear geometry
    c. Intuitive user Interface as an intermediary between design data and Assetto Corsa
- Features not implemented
    a. Uploading aero data/utilizing extended physics features for aero maps

Our resulting program is a user interface with four tabs the user can switch between to change pages in the UI: Configuration, Power Train, Suspension, and Aerodynamics. The first three of these four are functional and allow the user to specify the following for each page. Additionally,

our program automatically locates the folder containing the Assetto Corsa game and prompts the user to specify this location if unsuccessful. This is necessary to output files in the appropriate location within the game's directories.

The Configuration page, pictured in figure 2.1, consists of the vehicle's name and version. The user can use either Load a previously created vehicle from the game or Build, which creates the vehicle in the game. Loading a vehicle also loads in the configuration options used to create that vehicle, consisting of the name, version number, and paths to configuration files for powertrain and suspension to allow for faster iteration–the user won't have to specify unchanged configuration files.

The Power Train page, pictured in figure 2.2, allows the user to specify two files: powertrain data (A torque curve .csv which compares engine RPM to Newton Meters of torque at the wheels) and gearbox data, specifying a dynamic number of gears and the corresponding ratio. Each field has a button next to it that provides information about the uploaded file.

The Suspension page allows three file uploads to specify the overall vehicle setup, front suspension data, and rear suspension data. These files come from software already used by the Mines Formula team and are input in .csv format. There are info buttons for each field.

The Aerodynamics page allows for manipulation of a file called aero.ini and other corresponding files, which the game uses to model aerodynamics. Unfortunately, we couldn't get this page's implementation to the scale we had hoped for, but it was a stretch goal and not included in our definition of done. The functionality that is present will still be useful.

At the bare minimum, the program user can specify a name for the vehicle and click build. Once complete, a vehicle with the name specified will be drivable in Assetto Corsa, matching a default configuration. More advanced configurations are possible using the additional pages in our program UI. An image of a drivable configuration generated by our program can be seen in figure 2.3.

In conclusion, we have achieved "doneness" regarding functional requirements as specified by our Definition of Done. We have yet to deliver our software: The Mines Formula team is currently competing this week and racing their Mines Formula 9 vehicle, and the team is remarkably busy. At the minimum, we will upload our work to their google drive and provide documentation for its usage.

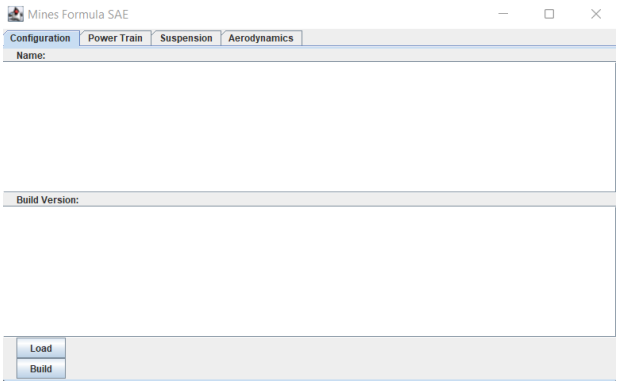


Fig 2.1 - Final Result Configuration Page

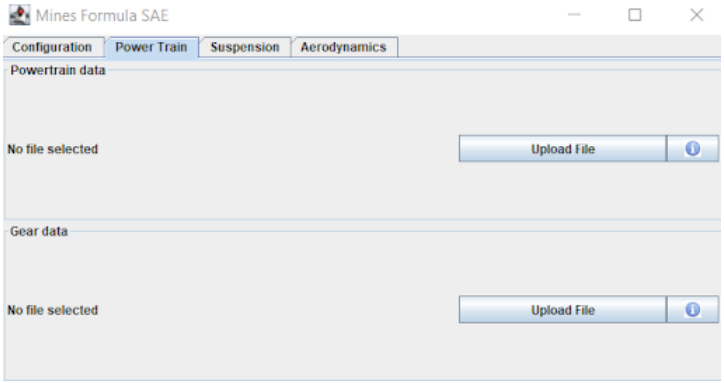


Fig 2.2 - Final Result Powertrain Page

Fig 2.3 - Final Result Driving in Assetto Corsa

X. Future Work

In the future, work can still be done to improve the software's current functionality. To improve the functionality, future software developers will need Java and access to GitHub. This will allow future developers to view the current code to make changes and improvements. Assetto Corsa will be needed, not only to perform tests but as it is the primary software used by the client. Knowledge of Java programming and cars would be beneficial. Understanding the basics of how cars work would decrease the amount of time needed to learn about cars. The decreased time would allow for more time to be spent programming.

Other than programming, future software developers should also know how aerodynamic maps work and have a general idea about cars to continue expanding this project. This is because aerodynamic aspects that were given as a stretch goal by the client are a high priority. Completing the aerodynamic aspects will be doable in five weeks. The timespan comes from needing to understand aero maps, vehicle dynamics, and advancing the physics used in the game currently. Gaining the understanding to complete the work successfully will require time.

A GitHub repository and documentation can be accessed to aid in future work.. GitHub includes all the code written for the project. The code has comments that explain the code itself. The documentation explains in more detail what was accomplished in the five-week session. Included documentation that is present in addition to the documentation that is present in the software

XI. Lessons Learned

This project has made us learn many valuable lessons. We learned different skills in communication, working as a team, technicality, and programming. Since we had little to no background on cars or the driving simulator, Assetto Corsa, this project helped us understand how they worked.

This project has served as an introduction to vehicle dynamics since it pertained to high-efficiency racing. In this regard, the project taught us that when working in the industry, it will often not be enough to rely on our knowledge as computer scientists, but we will have to use those skills in conjunction with new skills learned on the job.

The utilization of Scrum methodology was also informative, and we have learned the basics of how the Scrum Agile framework functions and witnessed some benefits that it can bring to a team setting.

XII. Team Profile

Nicholas Estoll

Junior
Computer Science major
Hometown: Lafayette, Colorado
Work Experience: Human-Centered Robotics Research Assistant, Play the Whistle Referee
Clubs: Cybersecurity
Hobbies: Game Design, Reading, Soccer

Caleb Edwards

Senior
Computer Science + Data science
Hometown: Manitou Springs, Colorado
Work Experience: Graphics Design, Summer Camp Counselor
Clubs: Club Penguin circa 2010
Hobbies: Cooking, Weightlifting, Hiking

Brianna Dalton

Senior
Computer Science / Teaching minor
Hometown: Thornton, Colorado
Work Experience: Colorado School of Mines Computer Science Front Desk, Colorado School of Mines Multicultural Engineering Program, Walmart
Clubs: Society of Women Engineers, Kappa Alpha Theta
Hobbies: Reading, Crocheting

Jasmine McCrary

Senior
Computer Science: Computer Engineering Track
Hometown: Kathmandu, Nepal
Work Experience: Marketing Intern, Psychology Department General Assistant
Club: Society of Women Engineers
Hobbies: Traveling, drawing, reading

References

Links to documentation:
Assetto Corsa physics pipeline
https://docs.google.com/document/d/1YSHpJfAcz66rzcUZl83No0hpQw3fXgPByf-PdnlBsK4/
Assetto Corsa CSV to INI Mapping (Part of our development)
https://docs.google.com/spreadsheets/d/1t4Ge5Fu7gY8NKwT0g_dCfjwRerowOhe7NPJK-uSM-tY/edit?usp=sharing

[1]"The code affirms an obligation of computing professionals to use their skills for the benefit of society.," Code of Ethics, 2018. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: 07-Jun-2022].

Appendix A – Key Terms

Term	Definition
UI	*User Interface*
Assetto Corsa	*A hyper-realistic racing simulator game*
TDD	*Test-driven development*
ACM	*Association for Computing Machinery*