CS CAPSTONE DESIGN DOCUMENT

DECEMBER 1, 2017

SOLAR CAR SIMULATION

PREPARED FOR

PHOENIX SOLAR RACING

CAILIN MOORE

PREPARED BY

GROUP 43

INFERNO

DENNIE DEVITO

LOGAN KLING

DAKOTA ZAENGLE

Abstract

This project will simulate the performance of the Phoenix Solar Racing solar car to improve their race strategy. In this document will discuss the technology we will use to create this software and how we intend to move forward with the project.

Contents

1 INTRODUCTION

1.1 Scope

For this project we will develop a Solar Car Simulation as a minimal software that takes variables from the user and uses them to calculate the solar car's performance. The software will be used by the Solar Racing Team to estimate their cars performance. They have not used software for this purpose in previous races and because any software to estimate the efficiency of vehicles is based on non-electric engines or does not take solar power into account so they have created this project. The scope of this project will be focused entirely on the car that Phoenix Solar Racing is building and the systems they will be running this software on.

1.2 Purpose

The main purpose of this Design Document is to describe how we will be approaching this problem and creating software for Phoenix Solar Racing. This document is intended for the members of the Phoenix Solar Racing Team, ourselves, and anyone who is interested in the outcome of the project.

1.3 Definitions

1.3.0.1 **GUI**: graphical user interface, the system the end user will use to interact with the program.
1.3.0.2 **API**: Application Programming Interface.
1.3.0.3 **wxWidgets**: the library we will use to create our GUI and display output.

2 TECHNOLOGY OVERVIEW

2.1 Math Library

ArrayFire is a high performance software library for parallel computing with an easy-to-use API. Its array based function set makes parallel programming more accessible. It is also very optimized for GPU (Graphical Processing Unit or Graphics Card) computations (as a GPU often has dozens of processing units with dozens of "'threads"' each. This will be very helpful for splitting up a 2000 mile drive into a hundred 20 mile drives. Doing so would speed up the simulation by two orders of magnitude, and we can always give the Phoenix Solar Racing team the option to run the simulation in a much slower single thread to improve accuracy. Despite this library being developed by a private corporation, it is open source. [1]

2.2 Physics Framework

Newton Game Dynamics is a cross-platform physics engine written in C++ for realistically simulating rigid bodies in games and other real-time applications. Its solver is deterministic and not based on traditional Linear Complementarity Problem (LCP) or iterative methods. Its ability to be split up into multiple processes (with some work from the programmer) and permissive Zlib license are appealing. The developers of the Newtonian Game Dynamics Engine are even working on an engine that can use multiple cores. Given this, it's clear that the Newtonian Game Dynamics Engine was designed with speed and optimization in mind. [2]

2.3 Aerodynamics and CFD Packages

If the aerodynamic constants from Solid Works provided by the Phoenix Solar Racing Team are too inaccurate, we will be using SU2. The Stanford University Unstructured (SU2) suite is a cross-platform and open-source collection of software tools written in C++ for performing Partial Differential Equation (PDE) analysis and solving PDE-constrained optimization problems. The toolset is designed with computational fluid dynamics and aerodynamic shape optimization in mind. It is designed to be extremely flexible. As such it has also been used for several multi-physics simulations, including but not limited to: combustion modeling, two-phase flow simulations, magnetohydrodynamics simulation, and multi-species thermochemical non-equilibrium flow analysis. [3]

2.4 GUI and Visualization Frameworks

For our GUI framework we will be using wxWidgets. It uses the API of the native operating system. That allows us to focus more on the functionality of the interface and let it handle the look on its own. wxWidgets comes with libraries for data display so we would not need to locate our own visualization framework and hope it works easily with our GUI framework. The framework is open source and free but we will need to read and obey their license terms in order to use this library. [4]

2.5 Software Testing

The forms of software testing we will make use of are: Acceptance Testing, GUI Testing, and Compatibility Testing. [5]

2.6 Weather API

Open Weather Map will be our choice for the weather API especially because of its accuracy and consistency. It provides free weather data and forecast API that can be suitable for any cartographic services for any programming language, smartphones applications, and even web applications. It also provides a wide range of weather data such as map with current weather, week forecast, precipitation, clouds, and data from weather stations.

Acceptance testing will allow us to verify that the members of Phoenix Solar Racing are able to use the program and interact with the interface. More constraints for this testing are included in the Project Requirements Document as it is one of the criteria we use to test for completeness of this project. Acceptance testing will take place only after the software is functionally complete.

GUI testing will be used to ensure the user interface functions properly. For this test we (the programmers) will do the testing as it does not focus on the usability of the GUI but rather the functionality. These tests can be automated to some extent and will be performed regularly while we make the user interface to ensure it still functions with the rest of the program.

Compatibility testing will ensure the program can run on our systems and the systems used by Phoenix Solar Racing. These tests will take place naturally whenever we run the software on our own systems and additionally when we have a functional program to test on the PSR systems.

3 Context View

3.1 Program Design

On this section, we will talk about the black box view for our simulation program. This section covers overall design description for our simulation program including the design concern and the design element.

3.1.1 Design Concern

The primary goal of this project will be to use our program to simulate the energy usage of a solar powered vehicle. This software will calculate the optimum race speed given variable parameters that affect the car's energy usage, similar to the software used by Tesla to estimate range based on charge. This software will estimate the energy usage of the vehicle over a given time period, estimate the required speed to use a given amount of energy to cover a given distance in a specified amount of time, and the mechanical efficiency of the vehicle to run in varying weather conditions. The secondary goal will be to provide a simple, easy-to-use interface to allow any member of the Solar Racing Team to use the software efficiently. The outcome of this project is to create a car simulation software which will help the Phoenix Solar Racing team to improve their racing strategy in preparation for the ASC (American Solar Challenge).

3.1.2 Design Elements

3.1.2.1 Racers: The main user for this software will be the member of Phoenix Solar Racing team. PSR is a solar car racing team from OSU that contains of mainly mechanical engineer, electrical engineer, and computer science people. This program will be used by any of the team member to modify the car itself in order to win the race.

3.1.2.2 Design relationship: Users of this software will be able to interact with the program via car simulation and the real car. The user will be able get data from the simulation software as accurate as data they will get from driving a real car.

3.1.2.3 Design constraints: This project is limited to its software. This means the program will not be directly connected to the car. Some other higher level libraries and software that will be used in this software are:

- C++: is the main programming language that we will use for the development of this software.
- Newton Game Dynamics: is our choice for the physics framework.
- Open Weather Map: is a weather API that help the simulator to get information about weather. condition.
- WxWidgets: is a C++ embedded widget toolkit and tools library for creating graphical user interfaces (GUIs)

4 INTERFACE VIEW

4.1 Design Concerns

The design concerns from the interface viewpoint we are focused on are mainly related to the users ability to interact with this software. This is because the software will have little to no interaction with other software and will only interface with the user. The end users will not be the ones developing the software and as a result the interface will need to be friendly and straightforward. We as the team creating the software will need to perform acceptance testing in order to verify the software is appropriately usable.

4.2 Interface Attribute

The software will not be required to interact with the system other than to collect user input and perform its calculations. Due to this we will be focusing on the user interface in this section. If we are able to reach our bonus goals of including map interaction we will expand this section to reference that interface as well.

The user interface will be as minimal as possible. It will not be necessary to provide anything beyond what will be used regularly by Phoenix Solar Racing as they are the end user and likely the only users. The software will save the data related to the car, to be input in a simple text form, so it will not need to be entered every time as it is unlikely the cars attributes will change greatly between runs of the software. These variables are listed and discussed in more detail in the Project Requirements Document.

In the form of output this program will display relevant data to the user in the form of text and simple graphs. The text output will include the information entered originally as well as the output of calculations performed on that information. The results the Racing Team is looking for are discussed in detail in the Project Requirements Document. With the libraries we will be using we are able to display data any form of graph we can come up with.

5 Conclusion

The Design Document has reviewed the technologies we will be using for the project and defined the Context View and Interface View for this project. This document has restated the scope of the project and explained the terms we will be using. Moving forward this document will be expanded to explain the processes we use to create the software in the future.

References

[1] Pavan Yalamanchili, Umar Arshad, Brian Kloppenborg, Miguel Lloreda, Ghislain Antony Vaillant, Cedric Nugteren, Filipe Maia, Gatan Lehmann, Nathan Jackson, Richard Klemm, Andreas Schuh, and Vardan Akopian, “Arrayfire.” [Online]. Available: https://arrayfire.com/

[2] Julio Jerez and Alain Suero, “Newton game dynamics.” [Online]. Available: http://newtondynamics.com/forum/newton.php

[3] Heather Kline, Tim Albring, Brendan Tracey, Ruben Sanchez, Salvatore Vitale, David Thomas, Francisco D. Palacios, Antonio Rubino, and Samet Cakmakcioglu, “Su2.” [Online]. Available: https://su2code.github.io/

[4] “wxwidgets.” [Online]. Available: www.wxWidgets.org

[5] “Software testing dictionary.” [Online]. Available: https://www.tutorialspoint.com