CS Capstone Requirements Document

March 2, 2018

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 we review the goals of this project and describe the requirements we will follow in order to achieve those goals.

Contents

1 Introduction

1.1 Purpose

The main purpose of our Requirements Document is to describe what our software will do and how it should perform by listing the requirements to be completed. This document is intended for the members of the Phoenix Solar Racing Team, and anyone who is interested in the outcome of the project.

1.2 Scope

For this project we will develop a Solar Car Simulation as a minimal piece of 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, they have created this project. There are some benefits from creating a new software including: designing a specific GUI that only includes the features they need, includes variables specific to the solar car, and can be modified by their team.

1.3 Definitions

RAM corresponds to Random Access Memory, is a form of computer data storage which stores data and machine code currently being used.

Memory refers to data as it takes up RAM.

An **operating system** simply refers to the piece of software we'll be running our program in. Think Windows, Mac, Linux. Even mobile phones have their own operating systems. An operating system will make our lives easier.

C++11 simply refers to the most stable version of C++. As it was created in 2011, there are still a wide variety of compilers which don't support it, but that shouldn't be an issue here. This isn't going to be a low level program by any means.

1.4 Overview

In this requirements document we discuss the interactions our software will have with the system it runs on and the required software functions. We briefly cover who will be using the software and what requirements come out of that. In the Specific Requirements Section (3) we address the variables we will account for in the simulation, the user stories that will cover our requirements, the requirements that we will impose to ensure completion, and finally the time line for this project.

2 General Description

2.1 Product Perspective

2.1.1 System Interface

Our program will be able to run on any computer with an operating system of Windows 7, or any more recent version of Windows. It won't really interact with the operating system it's on at all.

2.1.2 User Interface

The GUI will include a basic menu system including the most useful features as buttons with helpful icons and clear, understandable labels underneath each one. Data will be entered through a series of form fields to be filled out for each of the variables. Because parts of the car are likely to change over time and the team is currently building a new car it will be important that they are able to change those values. In order to allow for changes to the car but avoid re-entry of every detail whenever the program is used we will allow import of text files to save and load car details.

2.1.3 Software Interfaces

If we're able to implement and use a weather API for this program, we will most likely use Yahoo Weather's API. It seemed to have the fewest restrictions. Unless we find a better map API which will allow us to download the altitude data for the world before the program runs, we'll use Google Maps API.

2.1.4 Memory Constraints

The language in which we will be coding this is C++, which is an object oriented programming language. This means that almost every part of our program won't even be able to access every part of itself without explicit permission. Given this, our program won't be able to access memory outside of itself, ever! This is as it should be. Our program also shouldn't hog too much memory to itself. The computers this will be run on are unlikely to have more than four gigabytes of RAM. As a result, our program must never use more than four gigabytes of RAM at any time.

2.2 Product Functions

Our for main product functions are:

1) To provide a simple and easy to use interface to allow team members to use the software efficiently.
2) To make the simulation program estimate the energy usage of the vehicle over a given distance at a given speed.
3) To estimate the required speed to use a given amount of energy to cover a given distance.
4) To include factors like the electrical and mechanical efficiency of the vehicle and the prevailing weather conditions.

2.3 User Characteristics

Our users will be the members of the OSU Phoenix Solar team (PSRT). The overwhelming majority of the PSRT are mechanical and electrical engineers. With this in mind, the PSRT will have absolutely no issue using a program, but they may have difficulty changing such a program. One of our final requirements will be to make sure they are able to confidently use all the features of the software.

2.4 Constraints

We're allowed to code this in any language as long as it can be run on Windows. Because this simulation will almost certainly be processor intensive, C++11 will be the best language to use. It's lower than Java and C#, both of which have been used by one of their electrical engineers, but it still has room for plenty of libraries—even if some of those necessary libraries will vary on different operating systems.

2.5 Assumptions and Dependencies

Our main assumption is that PSR will have computers with Windows 7 or newer available to run the software. If this assumption proves incorrect we will need to make sure the program can run on the systems they bring with them to the race events.

3 Specific Requirements

3.1 Variable Priorities

In this section we are listing and discussing the priority of the variables. This will be the order in which we implement them to let us test the program for accuracy early on with the minimum number of variables.

1) Group 1: Our first set of variables. We can use these to test the program both for accuracy and for usability.
   a) Electric Motor Consumption
   b) Battery Charge
   c) Solar Panel Input
   d) Distance
2) Group 2: After testing the GUI and making sure the first group is working correctly we will add these next variables.
   a) Cloud Coverage
   b) Air Resistance (Drag)
   c) Course Altitude Change from Start to End
3) Group 3: These are variables that may not have enough of an effect on the car to justify the extra calculations. We will look into adding them after we are sure the other two groups are correctly implemented. These variables can be considered stretch goals.
   a) Tire friction
   b) Change in Sun Angle Over the Course
   c) Latitude/Longitude of the Course (to account for different solar intensity)

3.2 User Stories

These User Stories are goals we will achieve by the end of the project. They are small sections of the project that when completed will help us verify we have completed the project.

1) As a PSR team member I want an interface in one window with the input fields for our car variables, and the output graphs for battery charge and speed so that I can use the program quickly and without confusion. *This is something we will test and make changes till PSR is comfortable using the program.*
2) As a user I want to be able to save and load certain variables so that I don't have to re-enter information for every simulation. *This can be implemented with the GUI and will not take more than an hour or two.*
3) As a driver I want at least the variables from Group 1 and Group 2 so the simulation is accurate I can get an idea of where I will be during the race. *After implementing variable Group 2 we will test for completion of this story.*
4) As a user I don't want to type in more than the three variable groups so that I can avoid setting the program up by hand for every run. *This user story will be complete after we have implemented the save/load feature and the variables from at least Group 1 and Group 2.*
5) As a driver I want to give the program speed and battery charge to see how far the car can go with my given conditions so I can plan the race. *This can be added after the second group of variables. It should take a week with story number seven.*
6) As a driver I want to give the program battery charge and the car's information to see what speeds will save the most energy. *This can be added after the second group of variables. It should take us a week.*
7) As a driver I want the program to use information about the altitude of the course so it can take into account variation in uphill vs downhill.
8) As a driver I want the program to use information from the weather to calculate how much sunlight the solar panels will get.

3.3 Requirements

1) Provide a program that uses the variables from Group 1 and Group 2 described in Section 3.1 to calculate the performance of the simulated car.
2) Make sure *at least* Cailin Moore, Nolan Dahl, and Gray Johnson are comfortable using the software. We would like to have them test the GUI using acceptance testing before we finish. As long as the three of them give it a 4 out of 7 or above we will consider this complete.
3) Verify the program can be run without crashing on the system the PSR team will be using at least 95% of the time.
4) Make the program accurate to within 60% of the results of the real car based on a given 10 mile drive. Make sure our margin of error never exceeds 40%. *This requirement may be considered fulfilled after testing with the race logs from the team's last car in case they are unable to build the new car before we finish the project in Spring of 2018.*
5) Provide an API to plan a route and return elevation data for the simulation.

3.4 Time-line

3.4.1 The Remaining Fall Term

The focus of this term has been—and will continue to be—establishing *all* of the requirements for this project. The very end of this term will also be used to start basic programming on this project.

3.4.2 Winter Term

This term is the term in which the bulk of this project will be programmed. It is also hoped that this project will be finished by the end of this term, such a goal is an ambitious one, though.

3.4.3 Spring Term

The project will be finished within the first 3 weeks of the term, or before. If there's any remaining time before we need to start work on our project report, we may try to implement our stretch goal of calculating how the angle of the sun changes the efficiency of the solar car. At this point, it will probably be too late to implement that, though.

3.5 Stretch Goals

1) Record the angle of the sun and the direction the car is facing.
2) Include variables from Group 3.

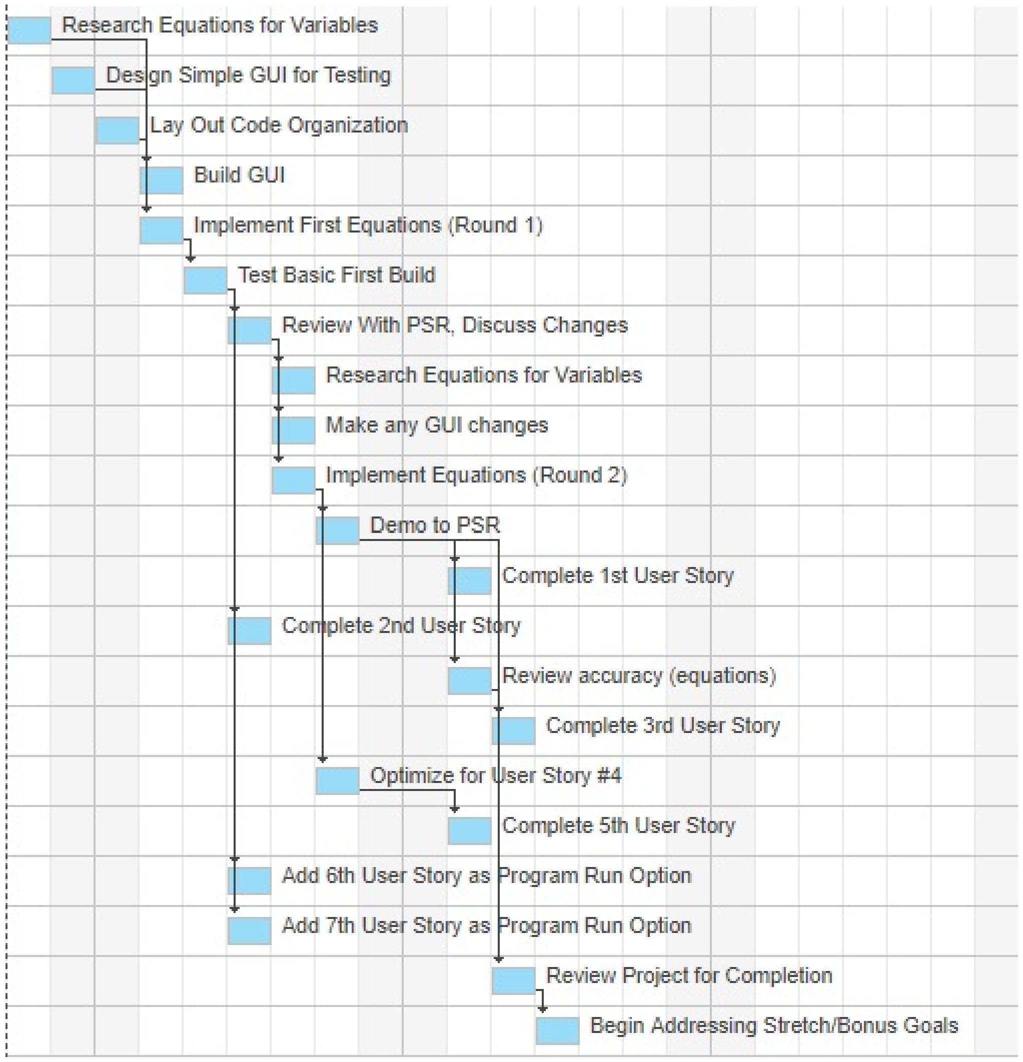

Fig. 1. Gantt Chart