Solar Car Performance Modeling Application

Maintenance Document

Contents

[Introduction 3](#_Toc480740075)

[System Overview 3](#_Toc480740076)

[Application Maintenance 3](#_Toc480740077)

[APIs 3](#_Toc480740078)

[Google Maps Elevation API 3](#_Toc480740079)

[OpenWeatherMap 3](#_Toc480740080)

[Models 4](#_Toc480740081)

[Aerodynamic 4](#_Toc480740082)

[Gravitational 4](#_Toc480740083)

[Motor 5](#_Toc480740084)

[Parasitic 5](#_Toc480740085)

[Rolling 5](#_Toc480740086)

[Solar 5](#_Toc480740087)

[User Interface 6](#_Toc480740088)

[JavaFX 6](#_Toc480740089)

[Scenebuilder 6](#_Toc480740090)

[Speed Limit Tool 6](#_Toc480740091)

[References 7](#_Toc480740092)

# Introduction

This document aims to describe the Solar Car Modeling Application in sufficient detail so that a programmer familiar with the Java language, but not initially familiar with this software, can maintain it properly.

# System Overview

The Solar Car Modeling Application is designed to assist with energy management during the American Solar Challenge. The application takes the Google Maps route file provided by the ASC and simulates solar car performance over legs of the race, incorporating location data along with weather and elevation along the route to estimate energy usage. The application runs in the lead car during the race and assists with race strategy that can be communicated to the solar car driver.

# Application Maintenance

## APIs

### Google Maps Elevation API

The Google Maps Elevation API is used to retrieve elevation data for points along the route. When a leg of the race is selected at application launch, the corresponding leg .csv file is loaded, and elevation for each point is retrieved. After the elevations have been retrieved, the road angle between each pair of points is calculated and stored in the list of positions.

The API key for Google Maps Elevation is specified in the code, at the time the data request is made.

### OpenWeatherMap

OpenWeatherMap is used to retrieve cloud cover, wind speed, and wind direction along the route. Current and forecasted weather data is cached in files for later retrieval. For each position on the route, the current weather data and forecasted weather data is pulled from the API. At application start time, the current weather data and forecasted data for each position are averaged together, which serves as the final value used in the model calculations.

The weather module contains a class “ApiKey”, where the OpenWeatherMap API key is stored.

## Models

Modeling is used to measure solar car performance under various conditions. These models are meant to assist with both design and race strategy. The models used in the application come from *The Winning Solar Car* by Douglas. R. Carroll, Chapter 2. All of the equations used in modeling are given in terms of wattage gained or lost by the various forces being modeled. The application calculates total watt-hours lost or gained at each point, and subtracts the result from a running total of remaining energy capacity.

### Aerodynamic

Aerodynamic drag represents a large portion of the total resistive force when the car is traveling faster than 40 km/h (25 mph). The Aerodynamic model uses the drag force multiplied by the velocity of the car to calculate power consumed by aerodynamics.

Where is the total power consumed by aerodynamics (in watts),

*p* is the density of air,

*V* is velocity,

is the drag area coefficient of the car.

The default drag area coefficient for the application is 0.12, as specified in *The Winning Solar Car*. Coast-down tests and/or wind tunnel tests could be used to obtain a drag area coefficient for the Sunseeker team’s specific car.

### Gravitational

When going uphill, gravitational resistance is increased; when going downhill, the resistance is decreased. Gravitational resistance is calculated by

Where is the resistive force on the car

W is the weight of the car

is the angle of the hill.

### Motor

Motor Efficiency is a constant specified in the Car Configuration file. This value will likely be found on the data sheet for the motor. The default value used by the application is 0.94. The average motor efficiency is in the 90-95% range, depending on how good the motor is and how well it is matched to the car’s power requirements [1].

### Parasitic

Parasitic losses refer to electrical losses that occur from devices such as fans and power converters, which are on all the time. In addition, connectors, wires, brake lights, etc. all absorb some amount of power. The values for charging and driving parasitic losses are separated; the default values are 10W power loss when charging and 30W when driving.

### Rolling

Rolling resistance represents most of the resistive drag force on the car at low speeds, and is a significant factor at high speeds. The rolling resistance energy used per kilometer traveled is given by

Where is the watt-hours of energy used per kilometer of travel,

is the rolling resistance coefficient specified in the Configuration File,

W is the weight of the car in Newtons.

The default value for the rolling coefficient is 0.0055. Carroll states that the rolling coefficient is approximately 0.005 on smooth roads, and 0.006 on rough roads, so the default value of 0.0055 has worked in modeling solar cars. Empirical tests may be able to get a more accurate value for the rolling coefficient.

### Solar

Time of day and latitude effect the amount of power available from the sun. For each position, the power the solar array receives from the sun is calculated by

Where is the total wattage produced by the solar array,

is the wattage the solar array would produce at high noon,

is the current sun angle based on time of day and latitude (radians),

is the noon angle for the current location (radians).

## User Interface

### JavaFX

JavaFX is a library used to build Graphic User Interface applications with Java, and is the basis of the application’s user interface. The layouts for user interface components are specified in the FXML scripting language, and connected to the application logic in Java code.

### Scenebuilder

Scenebuilder is the tool that was used to create the vast majority of the user interface. It is a GUI application that allows developers to design a user interface while writing minimal amounts of code. Scenebuilder automatically generates FXML code, as well as skeleton Java declarations and functions to begin implementing application logic for user interface elements. The main user interface layout is specified in ui/mainPage.fxml. Scenebuilder comes highly recommended for working on the user interface. More information on Scenebuilder can be found here:

http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html

## Speed Limit Tool

The speed limit tool was created to solve the issue of keeping speed limit data for the entire race. Google Maps has an option for speed limit data, but only in the Enterprise edition which costs $10,000 per year. The turn-by-turn PDF directions supplied by the American Solar Challenge include speed limits, but parsing them correctly from a PDF would be a considerable challenge.

The speed limit tool saves speed limits to the corresponding data points in the positions file. The user simply right-clicks two points to choose a segment of the route, and enters the speed limit.

# References

|  |  |
| --- | --- |
| [1] | D. R. Carroll, The Winning Solar Car- A Design Guide for Solar Race Car Team, Warrendale, Pennsylvania: SAE International, 2003. |