User's Manual

SFML Install steps Windows

(Note: Don't download SFML 3.0.0, you need the older version 2.6.2 to run this project)

This link is also a helpful reference when downloading
https://helpza.zenva.com/portal/en/kb/articles/how-to-install-sfml-for-windows#Installing_SFML

1. Navigate to the official download site and find the version that matches with your computer's specs.

    https://www.sfml-dev.org/download/sfml/2.6.2/

    For me this was GCC 13.1.0 MinGW (SEH) - 64-bit
    Run "g++ --version" in command prompt to find your compiler version

2. Right-click the ZIP and Extract All. (I put this folder onto my desktop)

3. To test this I made a folder called SFMLApp with a subfolder source that has a file main.cpp inside

    SFMLApp/
    ├── source/
    │ └── main.cpp

    Paste the code from the bottom of the official website into main for
    https://www.sfml-dev.org/tutorials/2.6/start-vc.php

4. In the command prompt, run this command:

    "g++ source/main.cpp -IC:/SFML-2.6.2/include -LC:/SFML-2.6.2/lib
    -lsfml-graphics -lsfml-window -lsfml-system -o app.exe"

    You should now have an app.exe file.

SFMLApp/
├── source/
│ └── main.cpp
├── app.exe

5. Go into where you extracted the initial zip file, click on the bin folder, and then copy the files:

sfml-graphics-2.dll
sfml-window-2.dll
sfml-system-2.dll
Into your SFMLApp folder

SFMLApp/
├── source/
│ └── main.cpp
├── app.exe
├── sfml-graphics-2.dll
├── sfml-window-2.dll
├── sfml-system-2.dll

6. In your file explorer double click on app.exe to run and you should see a green circle appear

To disable squiggles under #include <SFML/Graphics.hpp>:

1. (Make sure you are in your SFMLApp project folder)

Press Ctrl+Shift+P and navigate to C/C++: Edit Configurations (UI)

2. Click on .vscode/c_cpp_properties.json

3. Under "includePath":

Add a comma and your file path. For example,

```
"includePath": [
            "${workspaceFolder}/**"
        ],

//Turns into
"includePath": [
            "${workspaceFolder}/**",
            "C:/SFML-2.6.2/include"
        ],
```

4. Save the file (Ctrl+S)
5. Reload the window Ctrl+Shift+P ->Reload Window

And the error under #include <SFML/Graphics.hpp> will be gone.

SFML steps Mac:

1. Install SFML 2.6.1 on homebrew https://formulae.brew.sh/formula/sfml
https://www.youtube.com/watch?v=zjv4aGzFous&t=16s

2. To compile program:

```
g++ -std=c++17 source/main.cpp source/components.cpp
source/driver_input.cpp source/vehicle.cpp -o app \
   -I/opt/homebrew/opt/sfml@2/include -Iheaders \
   -L/opt/homebrew/opt/sfml@2/lib \
   -lsfml-graphics -lsfml-window -lsfml-system
```

3. To run:

```
./app
```

HOW TO RUN THE PROGRAM

Step 1

Ctrl + Shift + P
Or
CMD + Shift + P

C/C++ Edit configurations (UI)

Select c_cpp_properties.json

Paste:

```
{
    "configurations": [
        {
            "name": "macos-clang-arm64",
            "includePath": [
                "${workspaceFolder}/**",
                "${workspaceFolder}/headers",
                "/opt/homebrew/include",
                "/opt/homebrew/include/SFML"
            ],
            "defines": [
                "_DEBUG"
            ],
            "compilerPath": "/usr/bin/clang",
            "cStandard": "c17",
            "cppStandard": "c++17",
            "intelliSenseMode": "macos-clang-arm64"
        },
        {
            "name": "Win32",
            "includePath": [
                "${workspaceFolder}/**",
                "C:/SFML-2.6.2/include"
            ],
            "defines": [
                "_DEBUG",
                "UNICODE",
                "_UNICODE"
            ],
```

```
            "compilerPath": "C:\\mingw-w64\\mingw64\\bin\\gcc.exe",
            "cStandard": "c17",
            "cppStandard": "gnu++17",
            "intelliSenseMode": "windows-gcc-x64"
        }
    ],
    "version": 4
}
```

*Make sure that the compiler paths match your compiler and that the file path ("C:/SFML-2.6.2/include") matches where you installed SFML.

Step 2

Go to .vscode\tasks.json and paste the following

```
{
    "comment1": "// See https://go.microsoft.com/fwlink/?LinkId=733558",
    "comment2": "// for the documentation about the tasks.json format",
    "version": "2.0.0",
    "tasks": [
        {
            "label": "build project",
            "type": "shell",
            "command": "g++",
            "args": [
                "-o",
                "${workspaceFolder}/bin/main",
                "-I",
                "${workspaceFolder}/headers",
                "-ggdb",
                "${workspaceFolder}/source/*.cpp"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "problemMatcher": []
        },
        {
            "label": "build-windows-sfml",
```

```
    "type": "shell",
    "command": "g++",
    "args": [
        "source/main.cpp",
        "source/driver_input.cpp",
        "source/vehicle.cpp",
        "source/components.cpp",
        "-IC:/SFML-2.6.2/include",
        "-LC:/SFML-2.6.2/lib",
        "-lsfml-graphics",
        "-lsfml-window",
        "-lsfml-system",
        "-o",
        "bin/main.exe"
    ],
    "group": {
        "kind": "build",
        "isDefault": false
    },
    "problemMatcher": []
},
{
    "label": "build-mac-sfml",
    "type": "shell",
    "command": "g++",
    "args": [
        "source/main.cpp",
        "source/driver_input.cpp",
        "source/vehicle.cpp",
        "source/components.cpp",
        "-I/opt/homebrew/include",
        "-L/opt/homebrew/lib",
        "-lsfml-graphics",
        "-lsfml-window",
        "-lsfml-system",
        "-o",
        "bin/main"
    ],
    "group": {
        "kind": "build",
        "isDefault": false
    },
    "problemMatcher": []
},
```

```
        {
            "label": "run",
            "type": "shell",
            "command": "./bin/main",
            "problemMatcher": []
        },
        {
            "label": "run tests",
            "type": "shell",
            "command": "./bin/testmain",
            "problemMatcher": []
        },
        {
            "label": "run on windows cmd shell",
            "type": "shell",
            "command": "${workspaceFolder}/bin/main.exe",
            "options": {
                "cwd": "bin"
            },
            "problemMatcher": []
        },
        {
            "label": "run tests on windows cmd shell",
            "type": "shell",
            "command": "${workspaceFolder}/bin/testmain.exe",
            "options": {
                "cwd": "bin"
            },
            "problemMatcher": []
        }
    ]
}
```

*Make sure that the .cpp files have the correct relative path and are the same as in your project file structure

This will give you a build-windows and build-mac task
(View terminal -> Run Task -> build-windows).
(View terminal -> Run Task -> build-mac).

Use these to compile the project and then use

(View terminal -> Run Task -> run).

To run the project.

Step 3

After running the project you will have an output.csv file.
If you go to the graph.py file and hit run python file in the top right corner, you will get a graph of the speed and state of charge over time
*Make sure to have pip installed the pandas and matplotlib libraries.

UML DIAGRAMS

***<u>UPDATED IMAGE AT BOTTOM OF PAGE</u>**

Class Design

The project follows an object-oriented design pattern:

EV Class: Central class representing the electric vehicle. It manages system-level behaviors like power state, physical properties like vehicle mass, wheel radius, and drag coefficient, and ties together the motor and battery components. It is responsible for calling update methods across the battery and motor and interacting with both classes. Currently, we only use the EV class to get the wheelRadius is EV status (on/off), but all of its properties will be useful in the future for further improvement of the project.

Battery Class: Encapsulates battery properties such as SOC, voltage, SOH, and temperature. The battery supports discharging over speed changes and regenerative charging during braking.

Motor Class: Converts throttle input into angular velocity and torque output. It also calculates the motor's speed and manages regenerative braking based on driver input.

DriverInput Class: Stores throttle and brake input levels, and interacts with the GUI and keyboard input.

Charger Class: Simulates charging rates and power delivery limits. Tracks battery charging state with bool isCharging.

Relationships Between Classes

EV has a Battery and a Motor.
Motor interacts with both DriverInput and Battery.
Charger interacts with Battery class (using pointer).
DriverInput is a standalone class that influences motor behavior.

Classes Methods and attributes

Relationships Between Classes

EV has a Battery and a Motor.
Motor interacts with both DriverInput and Battery.
Charger interacts with Battery class (using pointer).
DriverInput is a standalone class that influences motor behavior.

Classes Methods and attributes
1. DriverInput (Handles the driver's throttle and brake inputs)

Attributes:
throttlePosition (float): How much the throttle is pressed (0 to 1)

brakePosition (float): How much the brake is pressed (0 to 1)

Methods:
DriverInput(): Constructor

get_throttle(): Returns the throttle position

get_brake(): Returns the brake position

set_throttle(float intensity): Sets the throttle position

set_brake(float intensity): Sets the brake position

2. Battery
Models the behavior of a lithium-ion battery

Attributes:

Q_max (float): Max capacity of charge in Ampere-hours

Q_now (double): Current charge level

V_max (double): Maximum voltage

R_internal (double): Internal resistance

voltage (double): Current voltage output

stateOfHealth (double): Battery health status

temperature (double): Current temperature

heatCapacity (float): Thermal mass of battery in J/°C

heatTransferCoeff (float): To environment W/°C

current (double): Current output

totalTimeSeconds (float)Total session time in seconds

totalDistanceKm (float) Total distance traveled in kilometers

Methods:
Battery(): Constructor

set_Q_max(float Q), set_Q_current(float Q), set_V_max(float V), set_R_internal(float R), set_SOH(float SOH), set_temp(float T): Setters for various battery parameters

get_SOC(): Returns State of Charge

get_Q_max(), get_Q_current(), get_V_max(), get_R_internal(), get_SOH(), get_temp(): Getters for battery state

discharge(float speed, float delta_t): Updates charge based on vehicle speed over time

charge(float V_applied, float time): Simulates charging based on voltage applied and time

3. Motor
Controls speed and torque output of the motor, based on demand

Attributes:
currentDemand (float): How much current is being drawn

speed (float): Current motor/vehicle speed (km/h)

powerRating (float): Motor power in kilowatts

maxCurrent (float): Maximum current allowed

efficiency (float): Efficiency ratio (0 to 1)

maxSpeed (float): Motor’s top speed in RPM

maxTorque (const float = 200.0): Max torque in forward drive

maxBrakeTorque (const float = 300.0): Max torque for braking

inertia (const float = 10.0): Inertial resistance of the motor

wheelRadius (const float = 0.3): Radius of the vehicle wheels

Methods:
Motor(): Constructor

updateSpeed(DriverInput&, Battery&, float deltaTime): Updates speed using throttle and battery power

encounterObstacle(): Placeholder for changing input due to sudden obstacles

4. Charger
Handles the charging process for the battery

Attributes:
isCharging (bool): Indicates whether charging is in progress

maxPowerOutput (float): Maximum power deliverable by charger

efficiency (float): How efficiently the charger works

Methods:

startCharging(Battery* battery): Begins charging the battery

stopCharging(Battery* battery): Ends the charging process

6. EV
The main vehicle class. Holds Battery and Motor class.

Attributes:
on (bool): Whether the vehicle is powered on

mass (float): Total vehicle mass

wheelRadius (float): Radius of vehicle’s wheels

dragCoefficient (float): Affects air resistance

frontalArea (float): Affects drag force

battery (Battery*): Pointer to the vehicle’s battery

motor (Motor*): Pointer to the vehicle’s motor

obstacle (bool): Whether there is an obstacle in the vehicle's path

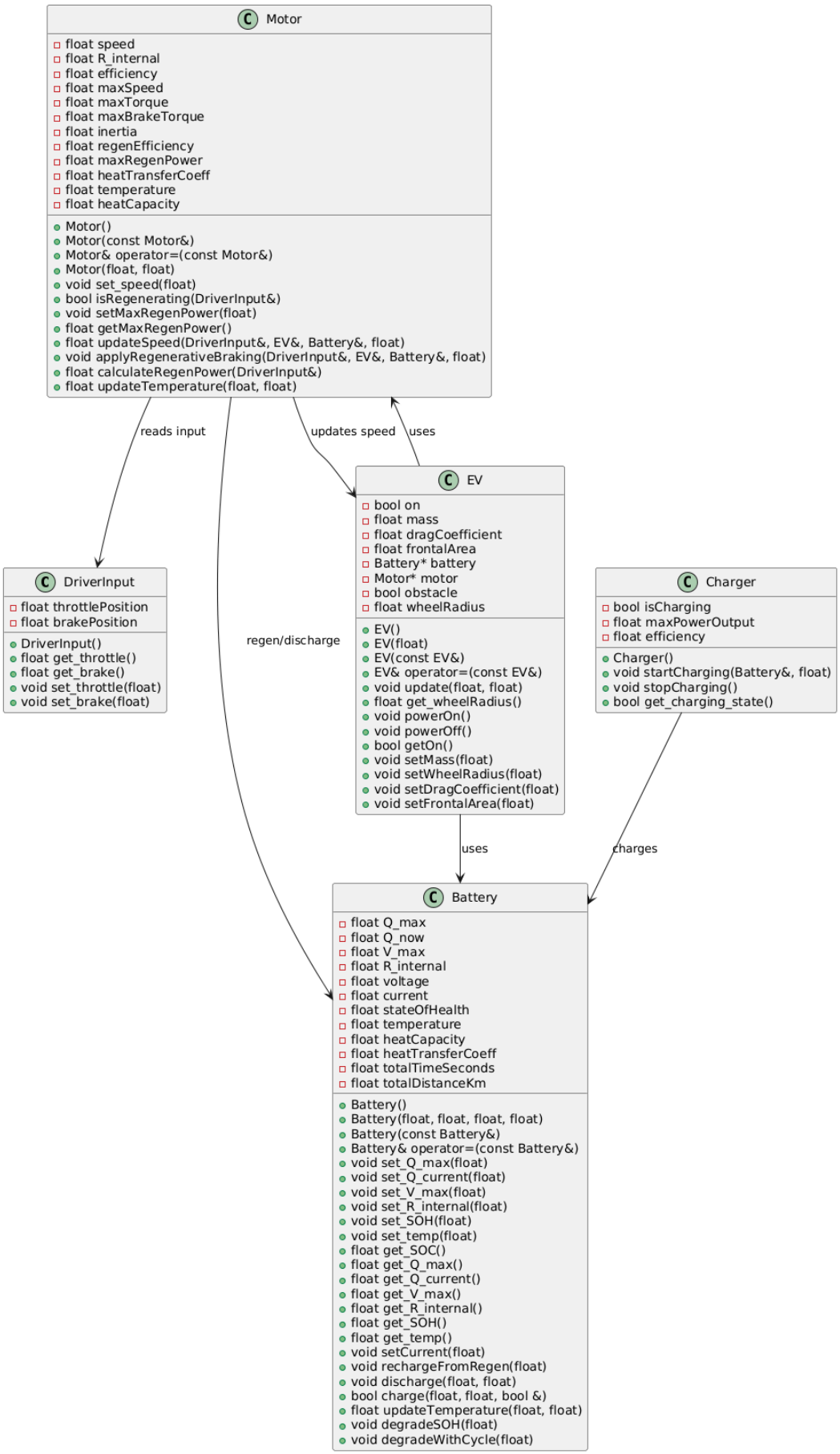
Methods:
EV(): Constructor

powerOn(): Turns on the vehicle

powerOff(): Turns off the vehicle

getOn(): Checks if vehicle is on

Setters for vehicle properties
setMass(float m)
setWheelRadius(float r)
setDragCoefficient(float c)
setFrontalArea(float a)


Motor
- float speed
- float R_internal
- float efficiency
- float maxSpeed
- float maxTorque
- float maxBrakeTorque
- float inertia
- float regenEfficiency
- float maxRegenPower
- float heatTransferCoeff
- float temperature
- float heatCapacity
+ Motor()
+ Motor(const Motor&)
+ Motor& operator=(const Motor&)
+ Motor(float, float)
+ void set_speed(float)
+ bool isRegenerating(DriverInput&)
+ void setMaxRegenPower(float)
+ float getMaxRegenPower()
+ float updateSpeed(DriverInput&, EV&, Battery&, float)
+ void applyRegenerativeBraking(DriverInput&, EV&, Battery&, float)
+ float calculateRegenPower(DriverInput&)
+ float updateTemperature(float, float)
reads input
updates speed
uses
regen/discharge
DriverInput
- float throttlePosition
- float brakePosition
+ DriverInput()
+ float get_throttle()
+ float get_brake()
+ void set_throttle(float)
+ void set_brake(float)
EV
- bool on
- float mass
- float dragCoefficient
- float frontalArea
- Battery* battery
- Motor* motor
- bool obstacle
- float wheelRadius
+ EV()
+ EV(float)
+ EV(const EV&)
+ EV& operator=(const EV&)
+ void update(float, float)
+ float get_wheelRadius()
+ void powerOn()
+ void powerOff()
+ bool getOn()
+ void setMass(float)
+ void setWheelRadius(float)
+ void setDragCoefficient(float)
+ void setFrontalArea(float)
Charger
- bool isCharging
- float maxPowerOutput
- float efficiency
+ Charger()
+ void startCharging(Battery&, float)
+ void stopCharging()
+ bool get_charging_state()
uses
charges
Battery
- float Q_max
- float Q_now
- float V_max
- float R_internal
- float voltage
- float current
- float stateOfHealth
- float temperature
- float heatCapacity
- float heatTransferCoeff
- float totalTimeSeconds
- float totalDistanceKm
+ Battery()
+ Battery(float, float, float, float)
+ Battery(const Battery&)
+ Battery& operator=(const Battery&)
+ void set_Q_max(float)
+ void set_Q_current(float)
+ void set_V_max(float)
+ void set_R_internal(float)
+ void set_SOH(float)
+ void set_temp(float)
+ float get_SOC()
+ float get_Q_max()
+ float get_Q_current()
+ float get_V_max()
+ float get_R_internal()
+ float get_SOH()
+ float get_temp()
+ void setCurrent(float)
+ void rechargeFromRegen(float)
+ void discharge(float, float)
+ bool charge(float, float, bool &)
+ float updateTemperature(float, float)
+ void degradeSOH(float)
+ void degradeWithCycle(float)