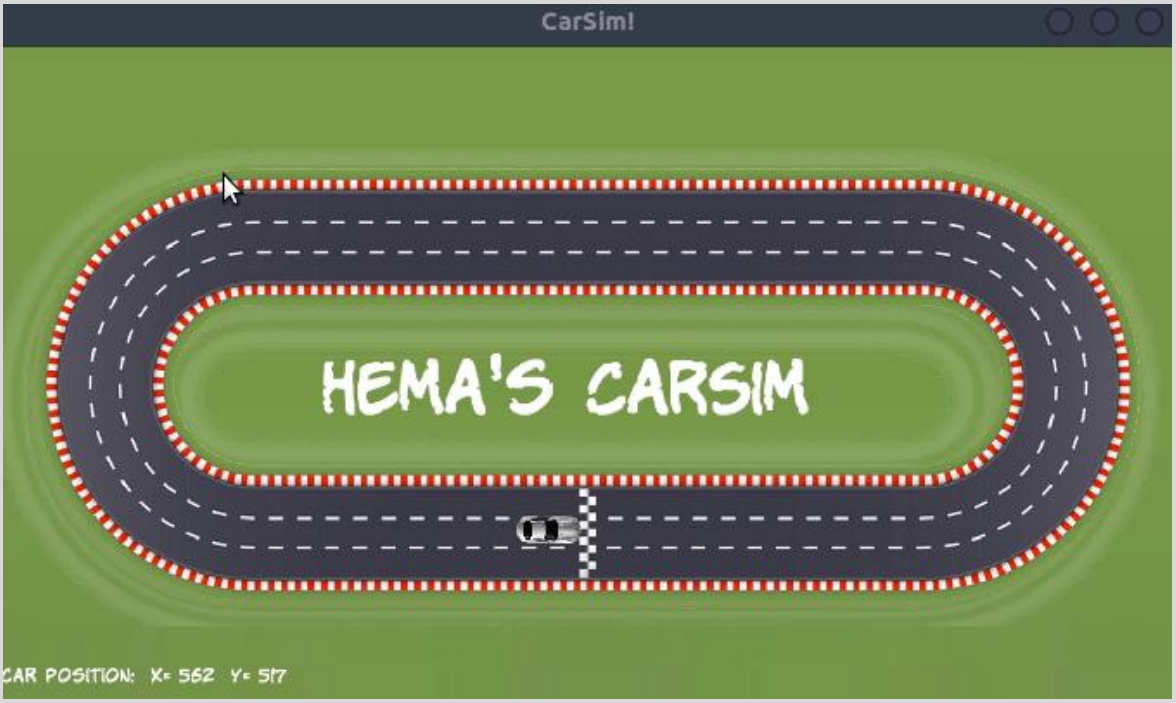
Udacity C++ Nano-Degree Capstone Project



Hema's CarSim

https://github.com/HemaZ/CarSim

Ibrahim Essam Ibrahim Abdelmonem

#	Criteria	Meets Specifications
Loops, Functions, I/O		
1	The project demonstrates an understanding of C++ functions and control structures.	* The project Code was clearly organized as classes and followed the rule of divide and conquer approach to divide each part of the project logic to functions. * For example the rendering task was handled by the member function “Render” of the “Renderer” Class. * A variety of control structures like loops and conditions were used in the project logic. For example in “Game.cpp” in “*void Game::Run()*” function we can see a while loop to make sure the simulator windows is still running and if conditions to handle the user inputs.
2	The project reads data from a file and process the data, or the program writes data to a file.	* the project read several data from the file to handle the graphic rendering of the simulator Background, Car and the desired fonts. * this can be seen in the “Renderer.cpp” in the class constructor, we use SFML “loadFromFile()” function to load the desired data.
3	The project accepts user input and processes the input.	* The project accepts two type of user inputs. The first type is through the Keyboard from the keys “Q,E,Up,Down,Backspace, ESC” to control the car steering angle, speed, state and close the simulator. * The second input type is through ROS rostopic command line interface. By publishing the steering angle to the ROS topic “CarSim/steer/”
Object Oriented Programming		
4	The project uses Object Oriented Programming techniques.	* The whole project is divided to several classes to follow the OOP concepts. * Every class has its own attributes and member function to support the class logic.
5	Classes use appropriate access specifiers for class members.	* Each class cleary define the access for every member attribute. For exmaple the “Game” class has several private attributes like “_car, _sub” which are the Simulated Car and the ROS subscriber.
6	Class constructors utilize member initialization lists.	* Members initialization list is used whenever it was suitable in different classes constructors like “Game, Renderer” constructors.
7	Classes abstract implementation details from their interfaces.	* One great use of OOP in the implementation abstraction through interfaces. For example the “Renderer” class provide “Render()” interface to handle the rendering of the simulator window without the user knowing anything about the underlying SFML library implementation.
Memory Management		
8	The project makes use of references in function declarations.	* Passing by refernce is heavily used in the classes member functions for example. The “Void Render(Car &car)” function in the Renderer class, “void Run(Renderer &rnd)” in Game class and the Game class constructor *“Game(int height, int width, ros::NodeHandle &nh)”*
9	The project uses smart pointers instead of raw pointers.	* A shared smart pointer was used to handle the simulator window in the Renderer Class *“std::shared_ptr<RenderWindow> _window”*. no Raw pointers were used in the project.