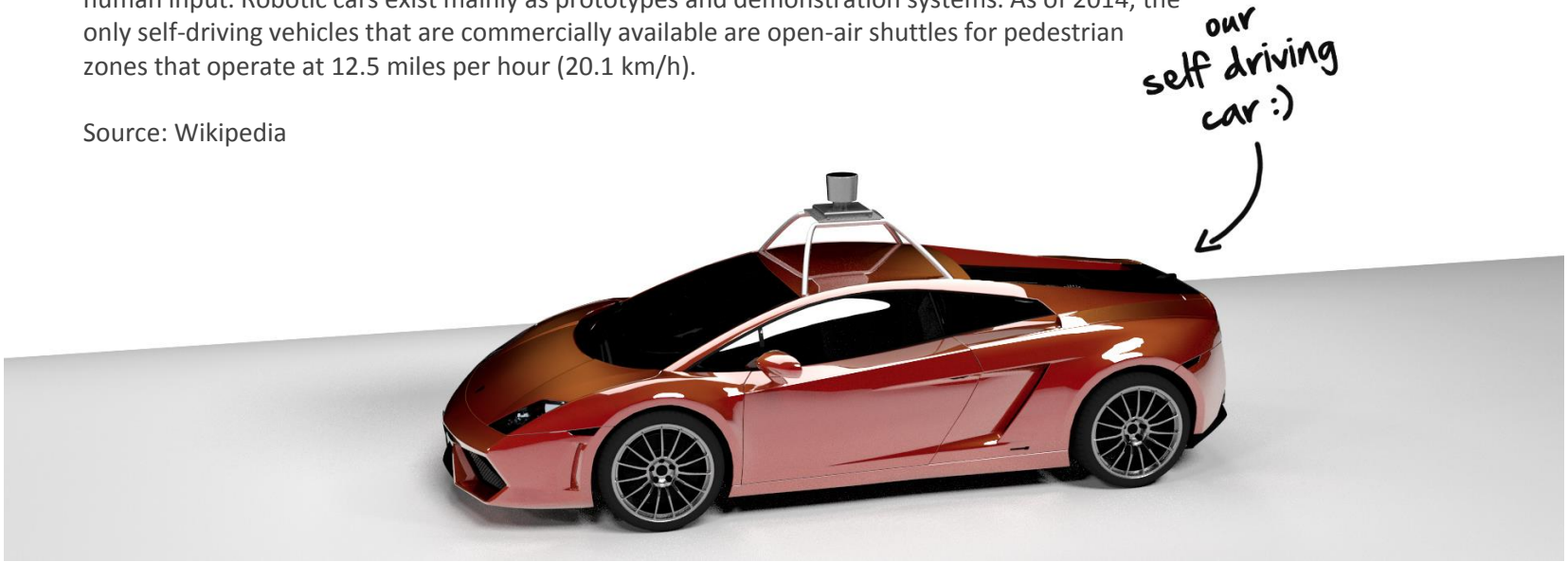
Autonomous Car

Catalysts Coding Contest
Vienna/Cluj 2014

Here we go!

An **autonomous car**, also known as a **driverless car**, **driver-free car**, **self-driving car**, or **robot car** is an autonomous vehicle capable of fulfilling the transportation capabilities of a traditional car. As an autonomous vehicle, it is capable of sensing its environment and navigating without human input. Robotic cars exist mainly as prototypes and demonstration systems. As of 2014, the only self-driving vehicles that are commercially available are open-air shuttles for pedestrian zones that operate at 12.5 miles per hour (20.1 km/h).

Source: Wikipedia



General Information

Info

In this contest you have to write an advanced cruise control system for a self driving car. The car is electrically powered and your program is in control of the throttle and brake pedal.

In every level you have to achieve goals in a simulated test drive.
You are provided with a simulator and your program has to communicate with it in real time.

Simulator

Info

The simulation runs with 60 steps per second. Every 6th step your program will receive an update of the world and is expected to send control commands for the throttle and brake pedal.

The simulator is written in Java and to run it you will need Java 7 (Update 6) or higher installed.

There are two modes of communication with the simulator. You only have to implement one method though.

Simulator - Console Mode

Info

The simulator executes your program and redirects stdin and stdout.
If you choose this method your program only has to interact with the "console". The downside of this approach is that you can't easily debug your application. Of course you can always attach your debugger after the program has started. You can also write debug output to stderr, which will be shown in a debug window of the simulator.

Example start commands:

```
java -jar simulator.jar --level=level1.in --exec=mysolution.exe
java -jar simulator.jar --level=level2.in --exec="java -cp /home/pong/classes my.company.Main"
java -jar simulator.jar --level=level3.in --exec="c:\python\python.exe solver.py"
java -jar simulator.jar --level=level7.in --exec="java -jar solver.jar "
```

Simulator - TCP Mode

Info

The simulator opens a tcp port and your program connects to it.

Example start commands:

```
java -jar simulator.jar --level=level1.in --tcp=7000
```

Simulator - Usage

java -jar simulator.jar --level=<level-description> [--exec=<solver>] [--tcp=7000] [--speed=2]

Arguments:

`--level=<level-description>`
selects the level description file (optional, default: level.in)

`--speed=<factor>`
set the speed factor (1-100) of the simulation (optional, default: 1)
(Hint: you can also set the speed of the simulation by pressing the buttons '1' until '0' on your keyboard)

`--exec=<solver>`
start in Console Mode

`--tcp=<port>`
start in TCP Mode

Communication Protocol

Info

Communication with the simulator is line based. Every message is one line of text. The simulator starts and sends multiple messages about the current state of the simulation. After that it will send "update". The client response to this message with the message “throttle” and “brake”. The simulator will wait until it has new values for “throttle” AND “brake”.

simulator messages:

`speed <real number>`	actual speed of the car in km/h (2 decimals)
`distance <real number>`	driven distance in meters (2 decimals)
`time <real number>`	elapsed time in seconds (2 decimals)
`update`	the simulator asks for new throttle and brake values

client messages:

`throttle <number>`	throttle power in percent from 0 - 100
`brake <number>`	brake power in percent from 0 - 100

Communication Example

Info

simulator messages are **red**
client messages are **green**

```
speed 0.00
distance 0.00
time 0.00
update
throttle 100
brake 0
speed 0.00
distance 0.00
time 0.10
update
throttle 100
brake 0
speed 6.23
distance 0.10
time 0.20
update
...
```

! The order of throttle and brake messages does not matter, but you have to send both. The decimal separator is “.”
The effect of the move command is not seen immediately but on the next state.

Simulator

Lvl1

Your task is to connect to the simulator with one of the two methods, drive at least 500m and then stop the car. After you have achieved all goals the simulator will write a “result.txt” file to the current working directory. This file will contain one line with all your throttle and brake commands. Upload the file to advance to the next level.

Goals Summary:

- Drive a distance of at least 500 meters
- Stop the car