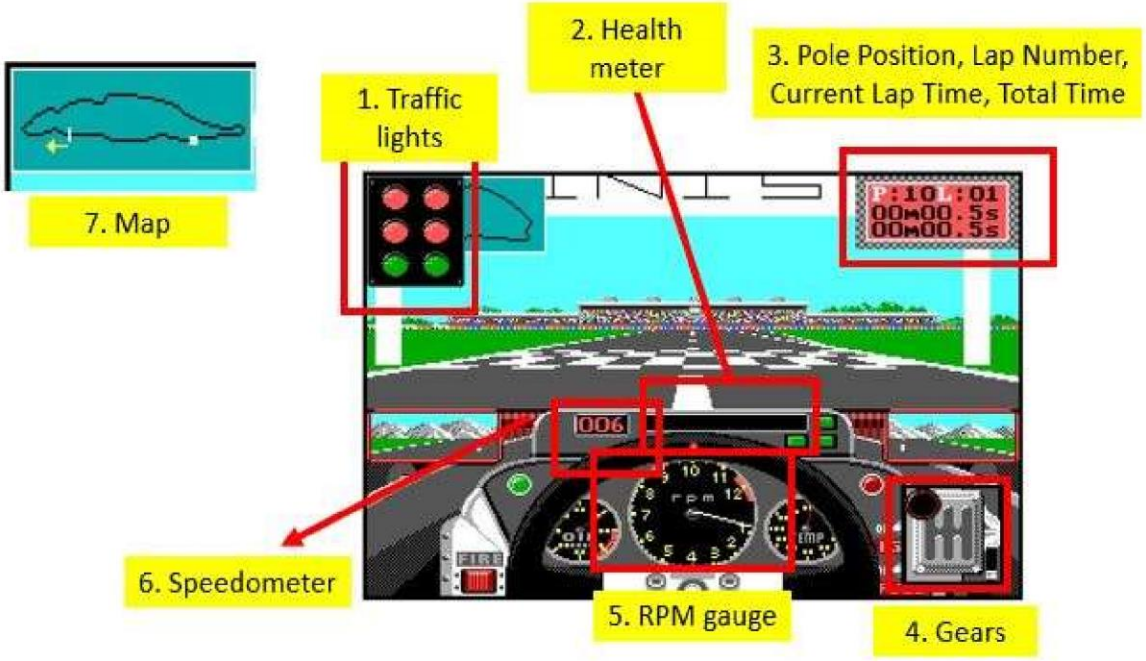
RG STEM CLUB: SELF DRIVING CAR – GRAND PRIX

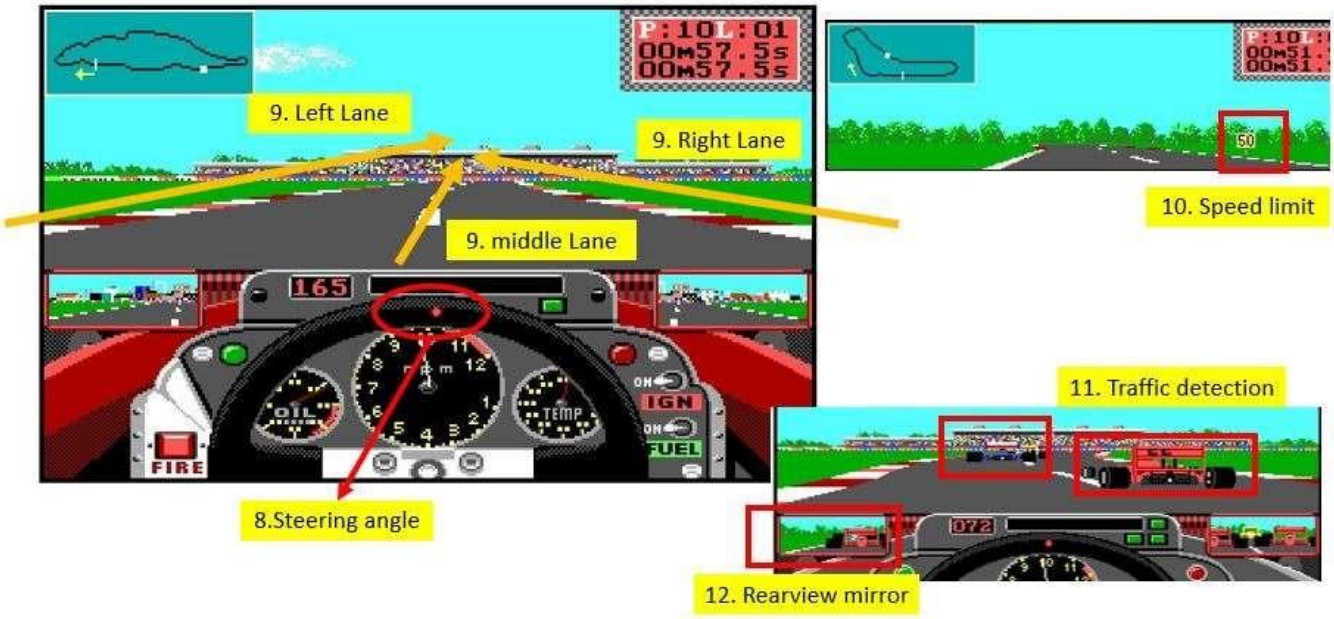
NAME: **DATE:**

Lab 1: Capturing Screen

Project Objective

In Part 1 the objective is to grab the inputs from the screen as images. The input images will be then converted to data that will be stored in a csv file. Theses inputs will be used to train neural network.





Once the image is captured, we need a strategy to convert the image into useful input. The table below shows potential strategy to make these conversions. As we see the impact to the processing speed (FPS), we may change these strategies. Also different members of the project can work on optimizing each function.

	Inputs	Challenge	Strategy
1	Traffic Lights	Detect presence of traffic light (look at last set of lights turn green)	OpenCV, hone the area of lights and look for green
2	Health Meter	In more difficult levels, when driver makes mistakes, the health meter fills up, once full, game is over	OpenCV, measure % damaged – identify in green, yellow or red zone – give feedback accordingly
3	Pole Position, Lap Number, Current Lap Time, Total Time	In the top right corner of the screen, the following information is available pole position, lap number, current lap time, total time incurred	OpenCV to grab image, use an OCR library to read the inputs and output the values
4	Gears	Under Automatic mode, gears are shifted up and down without driver intervening. In manual mode, driver decides when to shift	OpenCV identify where the gear head is located and determine shift gear
5	RPM Guage	Detect in the center of cockpit, acceleration of car is different at different gears. When running automatic it will be important to know when to shift	OpenCV – Hough lines to measure the slope of the needle within the circle
6	Speedometer	Detect the number displayed in the speedometer	OpenCV and convert with OCR
7	Map	Detect the map – will be handy for reinforcement learning and traffic management	TBD
8	Steering Angle	Calculate the steering angle from the position of the red dot in the steering wheel	OpenCV Hough Circle
9	Lanes	Lane detection will be restricted to ROI (region of interest) driving panel	OpenCV Hough lines
10	Speed limit signs	Traffic sign detection restricted to ROI	TBD, OpenCV haarcascade?
11	Traffic detection	Restricted from ROI, detection of cars – distance from other cars	TBD, OpenCV haarcascade??
12	Rearview information	Both left and right rearview mirrors – detect traffic when making turns	TBD

Acknowledgment:

The grabscreen function was developed by Sentdex, please refer to the github repository for more details:

https://github.com/Sentdex/pygta5 (https://github.com/Sentdex/pygta5)

Robo-Geek is always grateful to work with the open-source community. This project will be posted in our repository as well.

Standard OpenCV libraries

In [ ]:
```
import numpy as np
import cv2
import time
from grabscreen import grab_screen
```

With a bit of math, we can use the following grid to calculate the position of all the desired controls. Each function was created for a specific ROI (region of interest).

In [ ]:
```
def capture_full_screen():

    captured_screen = grab_screen(region=(0,27,640,425))
    rgb_full_screen = cv2.cvtColor(captured_screen, cv2.COLOR_BGR2RGB)
    return rgb_full_screen

#drivingwindow
def capture_drivingwindow():

    captured_screen = grab_screen(region=(0,140,640,245))
    rgb_drivingwindow = cv2.cvtColor(captured_screen, cv2.COLOR_BGR2RGB)
    return rgb_drivingwindow


#race map
def capture_map():

    captured_screen = grab_screen(region=(0,27,170,100))
    rgb_map = cv2.cvtColor(captured_screen, cv2.COLOR_BGR2RGB)
    return rgb_map

#scoreboard
def capture_scoreboard():

    captured_screen = grab_screen(region=(490,31,630,88))
```

```
    rgb_scoreboard = cv2.cvtColor(captured_screen, cv2.COLOR_BGR2RGB)
    return rgb_scoreboard

#speedometer
def capture_speedometer():

    captured_screen = grab_screen(region=(205,250,255,275))
    rgb_speedometer = cv2.cvtColor(captured_screen, cv2.COLOR_BGR2RGB)
    return rgb_speedometer

#health meter
def capture_healthmeter():

    captured_screen = grab_screen(region=(268,250,435,270))
    rgb_healthmeter = cv2.cvtColor(captured_screen, cv2.COLOR_BGR2RGB)
    return rgb_healthmeter

#RPM gauge
def capture_RPMgauge():

    captured_screen = grab_screen(region=(255,295,385,395))
    rgb_RPMgauge = cv2.cvtColor(captured_screen, cv2.COLOR_BGR2RGB)
    return rgb_RPMgauge

#Steering wheel
def capture_steeringwheel():

    captured_screen = grab_screen(region=(180,270,460,420))
    rgb_steeringwheel = cv2.cvtColor(captured_screen, cv2.COLOR_BGR2RGB)
    return rgb_steeringwheel

#Gear box to shift gears
def capture_gears():

    captured_screen = grab_screen(region=(525,325,610,410))
    rgb_gears = cv2.cvtColor(captured_screen, cv2.COLOR_BGR2RGB)
    return rgb_gears

#Left rearview mirror
def capture_leftRvm():

    captured_screen = grab_screen(region=(0,250,130,300))
    rgb_leftRvm = cv2.cvtColor(captured_screen, cv2.COLOR_BGR2RGB)
    return rgb_leftRvm


#Right rearview mirror
def capture_rightRvm():

    captured_screen = grab_screen(region=(510,250,(510+130),300))
    rgb_rightRvm = cv2.cvtColor(captured_screen, cv2.COLOR_BGR2RGB)
    return rgb_rightRvm
```

Finally a simple continuous loop to read the images and show them to verify the ROI are captured properly. It's important to note that this programs reads the input from the top left corner of the screen assuming the DOS simulator running GP game is on.

More information about running the simulator can be found in the introduction of the project. Simply uncomment the function that needs to be tested.

In [ ]:
```
def test_capture_functions():

    while True:

        #Capture game window and convert from BGR to RGB
        full_screen = capture_full_screen()
        drivingwindow = capture_drivingwindow()
        map_image = capture_map()
        scoreboard = capture_scoreboard()
        speedometer = capture_speedometer()
        healthmeter = capture_healthmeter()
        RPMgauge = capture_RPMgauge()
        steeringwheel = capture_steeringwheel()
        gears = capture_gears()
        leftRvm = capture_leftRvm()
        rightRvm = capture_rightRvm()


        #cv2.imshow('full screen',full_screen)
        cv2.imshow('driving window',drivingwindow)
        #cv2.imshow('map',map_image)
        #cv2.imshow('scoreboard',scoreboard)
        #cv2.imshow('speedometer',speedometer)
        #cv2.imshow('healthmeter',healthmeter)
        #cv2.imshow('RPMgauge',RPMgauge)
        cv2.imshow('steering wheel',steeringwheel)
        #cv2.imshow('gears',gears)
        #cv2.imshow('leftRvm',leftRvm)
        #cv2.imshow('rightRvm',rightRvm)


        if cv2.waitKey(25) & 0xFF == ord('q'):
            cv2.destroyAllWindows()
            break


test_capture_functions()
```