Symulacja silnika i skrzyni biegów na przykładzie wyścigu na ¼ mili

Żadne duże oprogramowanie, żaden duży projekt nie powstaje nagle z dnia na dzień, a budowany jest krok po kroczku. W naszym projekcie na samym początku samochodzik jedzie, kiedy wciśnięty jest klawisz w górę:

Symulacją by tego jeszcze nikt nie nazwał. Grą też nie.

```
if keys[pygame.K_UP]:
    throttle=1
else:
    throttle=0
```

Zmodyfikujemy to odrobinę, żeby samochód przyspieszał, kiedy wciśnięty jest gaz, a zwalniał, kiedy nie. Gra wykonuje się 60 razy na sekundę, czyli tak często odświeża się obraz i tak często wykonują się obliczenia. Na początek zrobimy tak, żeby autko zwiększało swoją prędkość o 1 za każdym razem, kiedy w momencie wykonania wykryje, że wciśnięty jest klawisz w górę, a zmniejszało, kiedy nie jest wciśnięty.

```
if keys[pygame.K_UP]:
    speed=speed+1
else:
    speed=speed-1
    if speed < 0:
        speed = 0
```

Prędkość autka to tak naprawdę prędkość przesuwania tła, to autko na ekranie stoi stale w jednej pozycji. Ze względu na to, że mamy dwa tła tworzące jako taką perspektywę (jedno porusza się szybciej, drugie wolniej), przesuwanie się tła znajduje się w zmiennej *movement*.

```
movement = METER*speed/FPS*(-1)
```

A oba tła przesuwamy odpowiednio tutaj:

```
bgrect1.move_ip(movement, 0)
bgrect2.move_ip(movement/4, 0)
```

Poprzednio przesuwało się o 5 pikseli co *tick* (obejście pętli), teraz zmienimy to tak, żeby przesuwało się o prędkość naszego autka. Prędkość wyrażamy w metrach na sekundę, czyli jeśli nasze autko jedzie $20m/s$ $(72\ km/h)$, to w ciągu sekundy powinno przesunąć się o 320 pikseli

– w naszej grze jeden metr odpowiada 16px. W związku z tym, pomiędzy aktualizacjami ekranu powinno przesunąć się o 5px. $16 \times 20/60 = 5$

Oczywiście powyższy przykład, w którym co tick zwiększamy prędkość o 1 powoduje, że nasze auto ma przyspieszenie $60\ m/s^2$ czyli po sekundzie będzie pędziło $60\ m/s$, czyli $216\ km/h$, a po dwóch $432\ km/h$. Takiego przyspieszenia nie ma ani Tesla ani Bugatti Chiron, ani nawet samochody F1, nie mówiąc już o Citroenie C4 Picasso z 2011 roku, którego tutaj symulujemy.

Urzeczywistnijmy najpierw to przyspieszenie. C4 Picasso z silnikiem 2.0 HDi osiąga $100\ km/h$ ($27{,}7\ m/s$) w $12.4s$. Przyspieszenie, czyli zmianę prędkości w czasie możemy w prosty sposób z tego policzyć:

$$a = \frac{\Delta v}{\Delta t} \rightarrow \frac{27.7m/s}{12.4s} \approx \frac{2.24m}{s^2}$$

Pamiętać jednak należy, że ta prędkość nie zwiększa się z każdym tickiem, a z każdą sekundą, więc tę wartość musimy podzielić przez liczbę klatek na sekundę.

```
if keys[pygame.K_UP]:
    speed=speed+2.24/FPS
else:
    speed=speed-2.24/FPS
    if speed < 0:
        speed = 0
```

Deklarowana przez producenta prędkość maksymalna C4 Picasso to $192\ km/h$, czyli $53{,}3\ m/s$. Ustawmy ją i zobaczmy, jak nam się będzie jechało z maksymalną prędkością. Napiszemy jeszcze hamowanie, hamulce są w samochodzie całkiem ważnym elementem. Załóżmy, że auto z prędkości $130\ km/h$ $(36\ m/s)$ zatrzymuje się w $72m$, czyli do pełnego zatrzymania potrzebuje 2 sekund. Czyli hamujemy z przyspieszeniem $-18m/s^2$

```
if keys[pygame.K_DOWN]:
    speed=speed-18/FPS
    if speed < 0:
        speed = 0
```

Już nasza gra przypomina jadące auto, a dopiero się rozkręcamy.

Wyświetlimy sobie prędkościomierz (cyfrowy, spokojnie). Aby w pygame wyświetlać tekst musimy go sobie zainicjalizować i powiedzieć, jakiej czcionki będziemy używać. Dlatego na górze, przy inicjalizacji pygame dopiszemy:

```
pygame.init()
pygame.font.init()
myfont = pygame.font.SysFont('Comic Sans MS', 30)
```

Natomiast w naszej pętli umieścimy to:

```
textsurface = myfont.render(str(speed), False, (0, 0, 0))
WIN.blit(textsurface,(0,0))
```

Tworzy to nową powierzchnię z tekstem (dlatego speed jest rzutowane do stringa), a następnie ją wyświetla. Ważna jest też kolejność, chcemy bowiem, aby tekst wyświetlił się na tłem, dlatego:

```
WIN.blit(background, bgrect)
WIN.blit(background, bgrect.move(bgrect.width, 0))
WIN.blit(vehicle,(40, HEIGHT-140))
textsurface = myfont.render(str(round(speed)), False, (0, 0, 0))
WIN.blit(textsurface,(0,0))
bgrect.move_ip(METER*speed/FPS*(-1), 0)
print (bgrect.right)
```

Prędkość wyświetla się m/s, ale dla nas czytelniejsze są jednak kilometry na godzinę, dlatego przemnóżmy sobie tę zaokrągloną prędkość przez $3{,}6$.

Teraz zmierzymy sobie przejechany dystans. To jest akurat proste, bowiem mamy prędkość i czas. Z fizyki wiemy, że

$$v = \frac{s}{t} \rightarrow s = vt$$

Stworzymy sobie zatem nową zmienną, do której co *ticka* będziemy dodawali prędkość/FPS

```
distance = distance+speed/FPS
```

Pamiętać też trzeba o utworzeniu nowej powierzchni i jej wyświetleniu

```
dist_surface = myfont.render(str(round(distance)) + " m", True, (0, 0, 0))
WIN.blit(dist_surface,(0,30))
```

WIN.blit znaczy, że w naszym oknie gry umieszczamy powierzchnię. Pierwszym argumentem jest dana powierzchnia, drugim piksel (szerokość, wysokość), w której ma znajdować się lewy górny róg powierzchni.

Na tej samej zasadzie możemy dodać timer: do zmiennej *time* dodajemy co *ticka* $1/60$. To już zadanie samodzielne.

Teraz jak mamy czas i prędkość, to możemy sprawdzić, czy matma się z grubsza zgadza. Jeśli od razu po uruchomieniu programu oprzemy się o strzałkę w górę, to powinniśmy osiągnąć prędkość 100 km/h w niecałe 13s na dystansie około 180 metrów

$$S = \frac{at^2}{2}$$

Zmodyfikujemy teraz nasz czasomierz, aby uruchamiał się dopiero, gdy samochód ruszy. W tym celu wprowadzić wystarczy zmienną typu boolean, która zmieni się na true, gdy wciśniemy gaz. Ta sama zmienna będzie nam uruchamiać timer.

```
if start == True:
    time = time+1/FPS
```

Chcielibyśmy też poznać garść statystyk - czas, w którym osiągnęliśmy $100\ km/h$ oraz prędkość, jaką osiągamy przy przekroczeniu linii ¼ mili. Zrealizujemy to dwoma ifami, które będą nam wyświetlały tekst w konsoli, następnie wypiszemy te dane na ekranie gry.

```
if round(distance) == 402:
    race_finished = True
    qtr_mile_time = time

if race_finished:
    print("1/4 mi: " + str(round(qtr_mile_time,2)) + " s")
```

W dokładnie ten sam sposób powinniśmy zrobić wyświetlanie czasu, w którym osiągnęliśmy 100 km/h. Jeżeli prędkość zaokrąglona do jednego miejsca po przecinku == $27.7\ m/s$, to powinno nam wydrukować informację na terminalu.

Teraz trochę o tym, jak działa silnik i skrzynia biegów, czyli trochę matematyki – Wchodzimy w poważniejszą symulację

Silnik samochodu spalinowego obracając się generuje określoną siłę. Ta siła, a dokładniej moment obrotowy przenoszony jest na koła, które obracając się przesuwają nasze auto do przodu. Ale pomiędzy silnikiem a kołami jest jeszcze skrzynia biegów – każdy samochód ma obecnie 5 do 6 przełożeń. Gdy samochód jest „na biegu", czyli mamy załączony bieg i sprzęgło nie jest wciśnięte to koła są bezpośrednio sprzężone z silnikiem.

Skrzynia biegów działa w dość prosty sposób – przekłada równocześnie obroty kół na obroty silnika oraz moment obrotowy silnika na koła. Dla przykładu, posłużę się pierwszym biegiem: W C4 Picasso pierwszy bieg ma *ratio* 3.538, czyli 1 obrót koła równałby się około 3.5 obrotom silnika. Ale skrzynie biegów mają jeszcze coś takiego jak *final drive*, czyli przełożenie końcowe, które odnosi się do wszystkich biegów. Dla omawianego samochodu to przełożenie jest równe 4.18, je też musimy dołożyć do naszego mnożenia, więc 1 obrót kół na pierwszym biegu równać się będzie $3.538 \times 4.18 = 14.8$ obrotów silnika.

I	3.538 (14.8)
II	1.92 (8.03)
III	1.322 (5.52)
IV	0.975 (4.08)
V	0.76 (3.18)
VI	0.645 (2.7)

Analogicznie ma się sytuacja z momentem obrotowym. Jeśli silnik produkuje $100Nm$, to na pierwszym biegu na koła przekładane będzie $1480Nm$. Moment obrotowy jest jednak miarą siły i dystansu, dlatego musimy jeszcze podzielić go przez promień koła. Obliczyć możemy go dość prosto. C4 Picassso ma opony 215/45ZR18, Oznacza to, że Opona ma 215mm szerokości, ściana boczna ma 45% szerokości, a opona wchodzi na felgi 18-calowe. Czyli promień naszego koła to $18/2 \times 2.54 \ + \ 21.5 \times 0.45$(połowa średnicy przeliczona z cali na cm plus wysokość opony). Czyli w badanym przypadku promień koła to $32.535\ cm$.

$$\frac{1480Nm}{0{,}32535m} = \ 4548{,}94\ N$$

Jest to siła, która pcha nasze auto do przodu. Od tego powinniśmy odjąć jeszcze opór powietrza, opory toczenia i osiągniemy siłę, która jest przykładana, żeby napędzić masę naszego auta. Opór powietrza stawiany przez nasz samochód jest niewielki, bo wygląda on jak jajko, jego współczynnik oporu aerodynamicznego wynosi 0.30. Jako że jest to siła działająca wprost przeciwnie do siły popychającej auto, po prostu musimy odjąć jakąś wartość oporu od siły działającej na samochód. Opór aerodynamiczny rośnie wraz z prędkością, ale przyjmiemy go na razie za stałą wartość 600N, czyli efektywnie samochód będzie popychany naprzód z siłą $4548{,}94N - 600N = \ 3948{,}94N$.

Z Fizyki wiemy, że przyspieszenie to siła/masa:

$$F = ma \Longrightarrow a = \frac{F}{m}$$

więc w naszym wypadku $3948{,}94N/1600kg \ \approx \ 2{,}46\ m/s^2$. I to tą wartość podzieloną przez FPS będziemy co obejście pętli dodawać do naszej szybkości (albo odejmować, o tym później)

To teraz implementacja:

```
 9    WHEEL_RAD = 0.32535
10    AERO_DRAG = 0.3
11    FINAL_DRIVE = 4.18
```

Deklarujemy sobie stałe zawierające promień koła, końcowe przełożenie i współczynnik oporu aerodynamicznego, następnie zadeklarujemy tablicę zawierającą przełożenia.

```
GEAR_RATIO = [0, 3.538, 1.92, 2.322, 0.975, 0.76, 0.645]
```

Tak, że zerowy element tablicy wynosi 0, czyli będzie symulował nam bieg jałowy (luz).

Bieg, na którym się aktualnie znajdujemy umieścimy w zmiennej *gear* i napiszemy implementację zmiany biegów - dopiszemy do niej warunek, który nie pozwoli wrzucić biegu wyższego, niż 6.

```
if keys[pygame.K_e] and gear < len(GEAR_RATIO):
    gear = gear+1
```

Analogicznie powinniśmy napisać redukcję, tylko ona powinna się dać wykonać, jeśli gear > 0. Dołożyć powinniśmy jeszcze jakąś informację, na którym biegu się znajdujemy:

```
gear_surface = myfont.render("bieg: " + str(gear), False, (0, 0, 0))
WIN.blit(gear_surface,(440,0))
```

Po uruchomieniu gry pojawi się jednak problem, bo wciśnięcie klawisza wykrywane jest jako przytrzymanie go przez jakąś liczbę *ticków*, a co za tym idzie, biegi zmieniają się wszystkie naraz. Dlatego dołożymy blokadę, która pozwoli zmienić nam bieg raz na pół sekundy:

```
if keys[pygame.K_e] and gear < len(GEAR_RATIO)-1 and block == False:
    block = True
    gear = gear+1

if keys[pygame.K_d] and gear > 0 and block == False:
    block = True
    gear = gear-1

if block == True and pygame.time.get_ticks()%FPS/2 == 0:
    block = False
```

Zmienną *block* należy oczywiście najpierw zainicjować przed pętlą wykonującą grę.

Teraz zadeklarujemy sobie zmienną *torque*, na razie ustawimy ją na 100, potem będziemy ją zmieniać. Jak wspominaliśmy wcześniej, siła działająca na nasze auto to:

$$(torque \times GEAR\_RATIO[gear] \times final\_drive)/r$$

a przyspieszenie to: $a = \frac{F}{m}$

1/FPS tego przyspieszenia dodajemy do prędkości co obejście pętli. Odjąć od tego trzeba opory.

```
force = torque*GEAR_RATIO[gear]*FINAL_DRIVE/WHEEL_RAD
acceletarion = (force-AERO_DRAG)/WEIGHT

if keys[pygame.K_UP]:
    start = True
    speed=speed+acceletarion/FPS
    if speed > 53.3:
        speed=53.3
else:
    speed=speed-2.24/FPS
    if speed < 0:
        speed = 0
```

Teraz musimy trochę zmodyfikować nasz program, ponieważ przyspieszenie może być ujemne. Trzeba wprowadzić do programu zmienną *throttle*, czyli gaz wartą 1, gdy wciśnięty jest klawisz w górę lub 0, gdy nie jest wciśnięty (lub coś pomiędzy, jeśli używamy kontrolera analogowego). *Throttle* mnożymy przy obliczaniu siły pchającej nasz samochód, a następnie niezależnie od wciśnięcia klawisza dodajemy przyspieszenie do prędkości. Po posprzątaniu wygląda to tak:

```
if keys[pygame.K_UP]:
    start = True
    throttle=1
else:
    throttle=0

force = throttle*torque*GEAR_RATIO[gear]*FINAL_DRIVE/WHEEL_RAD
acceletarion = (force-aero_drag)/WEIGHT
speed=speed+acceletarion/FPS

torque = TORQUE[round(eng_RPM/250)-2]
```

Poprawimy teraz opory. Na samochód działają dwa główne opory – toczenia oraz powietrza. Opór toczenia jest praktycznie stały, zwiększa się minimalnie wraz z prędkością, obliczymy go ze wzoru

$$F_{fr} = \frac{Wb}{\sqrt{r^2 + b^2}}$$

Gdzie:

- W – siła działająca na koła, czyli $W = mg$, gdzie m to masa pojazdu, g – przyspieszenie grawitacyjne, $g = 9.81 m/s^2$
- b – powierzchnia styku opony z podłożem, przyjmujemy $1cm = 0.01m$

- r – promień koła

$$\frac{1600 \times 9.81 \times 0.01}{\sqrt{0{,}105 - 0{,}0001}} \approx \frac{157}{0{,}323} \approx 484[N]$$

To jest opór toczenia. Aby utrzymać stałą prędkość musimy generować silnikiem dokładnie tyle. Dochodzi jeszcze opór powietrza – ten zależny jest od prędkości samochodu.

$$F = \frac{1}{2} pV^2 DA.$$

- p to stała gęstość powietrza,
- D to stała współczynnika oporu (dla nas 0.3),
- A to stała powierzchni napierającej, czyli powierzchnia przodu auta. Dla nas to $2{,}61m^2$
- V to obecna prędkość w m/s

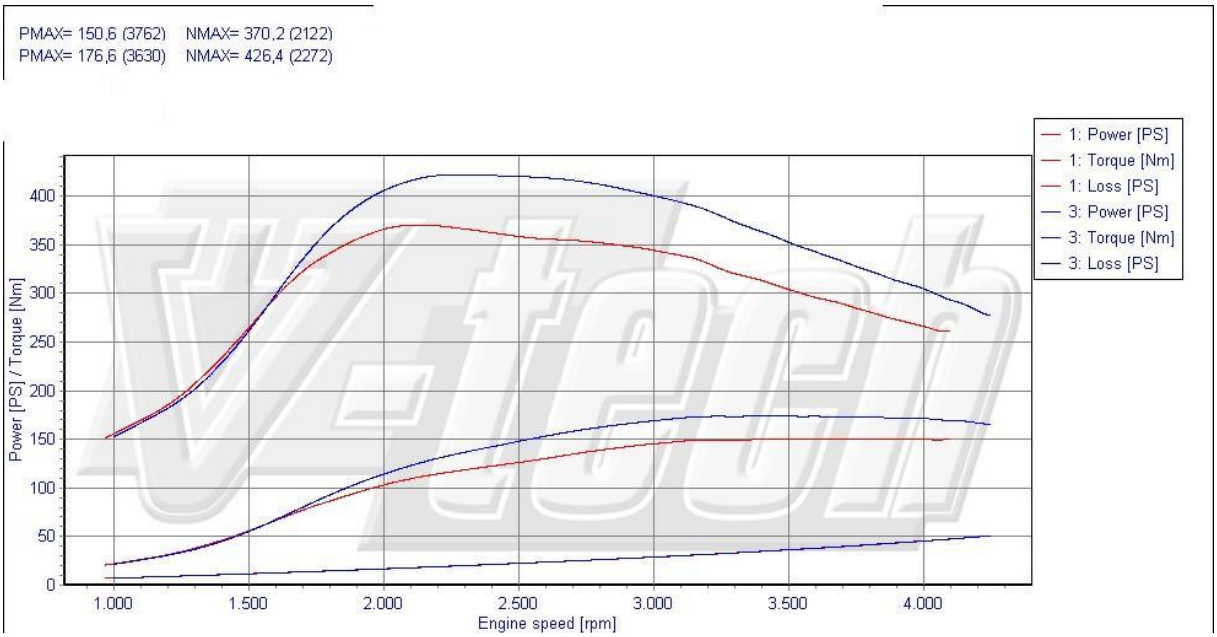
Dla samochodu jadącego $10\,m/s$, czyli $36\,km/h$ wartość oporu powietrza to $\frac{1.2 \times 10^2 \times 0.3 \times 2.61}{2} = 46N$. Przy prędkości maksymalnej naszego samochodu ten opór to już $1334N$. Dużo. Teraz to umieścimy w obliczeniach.

```
force = throttle*torque*GEAR_RATIO[gear]*FINAL_DRIVE/WHEEL_RAD

rolling_resistance = WEIGHT*9.81*0.01/math.sqrt(WHEEL_RAD*WHEEL_RAD+0.01*0.01)
aero_drag = 0.5*1.2*speed*speed*AERO_DRAG*2.61

acceletarion = (force-rolling_resistance-aero_drag)/WEIGHT
speed=speed+acceletarion/FPS
```

Teraz nasz symulator jest w niektórych momentach bardzo dokładny, bo wyliczamy opory z porządnych wzorów, ale silnik nadal produkuje stałe $100Nm$ i nie mamy zaprogramowanych obrotów. Teraz to naprawimy.

Silniki spalinowe generują moment obrotowy zależnie od obrotów. W silnikach spalinowych największy moment obrotowy dostępny jest zazwyczaj w okolicach 4000-5000 obr/min (RPM). W silnikach diesla, maksymalny moment obrotowy jest zwykle niżej, w okolicach 2000 RPM.

Poniżej znajdziemy wykres z hamowni dla C4 Picasso 2.0 HDi FAP 150 Exclusive



Z tego wykresu możemy odczytać, że największy moment obrotowy ten silnik generuje dla 2000 obrotów na minutę. – patrzymy na czerwoną linię u góry. Największy moment obrotowy przekłada się na największą siłę pchającą auto do przodu, także jeśli chcemy się jak najszybciej rozpędzać, to musimy tak zmieniać biegi, żeby silnik trzymał się w zakresie, w którym produkuje największy moment, czyli w okolicach 2-3k RPM, potem moment maleje.

Zadeklarujmy sobie zmienną *eng_RPM* z zabezpieczeniem, które nie pozwoli nam przekroczyć minimalnych (750 RPM) oraz maksymalnych (4250 RPM)

```
if eng_RPM < 750:
    print("Silnik gaśnie")
    eng_RPM = 750
if eng_RPM > 4250:
    print("Odcina!")
    eng_RPM = 4250
```

Musimy jeszcze te obroty jakoś wyliczyć. Wspominaliśmy wcześniej, że obroty kół przekładają się bezpośrednio na silnik (i odwrotnie) więc użyjemy ich, aby policzyć obecne obroty silnika.

```
wheel_RPM = speed/((WHEEL_RAD*2*math.pi)/60)
eng_RPM = wheel_RPM*GEAR_RATIO[gear]*FINAL_DRIVE
```

Proponuję to też gdzieś wyświetlić

Pozostała nam w zasadzie ostatnia rzecz, bo cały czas jedziemy na momencie obrotowym 100Nm, co powoduje, że przy 125 km/h na 5 biegu samochód traci obroty, bo opory są większe

niż siła generowana przez silnik. Proponuję stworzyć tablicę zawierającą moment co 250 RPM – do odczytania z powyższego wykresu. Na zerowym elemencie będziemy mieli moment dla 750 RPM, na pierwszym dla 1000, na drugim dla 1250 RPM itd.

```
TORQUE = [150, 160, 185, 260, 330, 370, 360, 355, 350, 340, 325, 300, 275, 260, 255, 0]
```

I teraz – moment obrotowy dla bieżących obrotów wyliczymy dzieląc i zaokrąglając bieżące obroty silnika przez 250. Czyli dla przykładu 3000RPM/250RPM = 12. Dzięki temu, że zaokrąglamy, to jeśli akurat mamy 3200 RPM, to 3200/250 = 12,8, ale zaokrąglenie spowoduje, że wybierzemy bliższy element z tablicy, czyli 13.

Teraz chcielibyśmy wyciągnąć odpowiedni element z tablicy, ale tablica zaczyna się od momentu dla 750 RPM, więc 12 element tablicy tak naprawdę odpowiada 3500 RPM. Dlatego musimy albo na początku tablicy dać dwa elementy 0 albo wyliczony indeks zmniejszyć o 2. Ja wybrałem tę drugą opcję:

```
torque = TORQUE[round(eng_RPM/250)-2]
```

Ważnym jest, aby nie umieścić obliczania momentu obrotowego pomiędzy obliczaniem obrotów silnika a ifami, które je blokują aby nie przekroczyły dopuszczalnych zakresów. To jest krótki moment, w którym obroty mogą przekroczyć dopuszczalną wartość i wtedy wykroczymy poza rozmiar tablicy.

Na tym kończymy gotowe materiały, a zagłębiać się można dalej. Zwróćcie uwagę, że zaczęliśmy od „symulatora”, w którym samochód jechał ze stałą prędkością, dodaliśmy do niego przyspieszenie wyliczone na podstawie czasu 0-100km/h. Potem dodaliśmy opory, sztywny moment obrotowy, potem obroty silnika i na ich podstawie wyliczaliśmy moment obrotowy. A gdzie jest koniec rozwoju tego oprogramowania? Trudno ustalić. Jeśli naszym założeniem pozostaje jazda w jednym kierunku, to samochód ma jeszcze wiele mechanizmów, których nie przewidzieliśmy. C4 Picasso ma napęd na przednią oś, a w momencie przyspieszania środek masy (punkt ciężkości) przesuwa się do tyłu, czyli to tylna oś jest bardziej dociskana – to zmniejsza tarcie z przednich kołach i równocześnie siłę, z którą napędzają pojazd. Idąc dalej – sprzęgło nie przenosi 100% mocy silnika na koła – część jest tracona na tarcie i oddawana w postaci ciepła. Mówiąc o sprzęgle – tego mechanizmu też nie zaimplementowaliśmy. Idąc jeszcze głębiej w silnik, który jest przecież złożonym mechanizmem. Tłoki, kompresja, cykle pracy tłoków, stopień sprężenia paliwa, nawet jakość mieszanki – to wszystko jest w jakiejś mierze opisywalne matematycznie, a co za tym idzie, możliwe do zaprogramowania.

Nie wspominając już o tym, że nasz samochód. Nie ma odwzorowanych jakichkolwiek skrętów, porusza się w jednym kierunku. Droga również może być wyboista, może się wznosić i opadać – a to bezpośrednio wpływa na opory, bo do sił, które musimy pokonać dochodzi grawitacja.

Jest bardzo wiele rzeczy, które musieliśmy bądź chcieliśmy w trakcie tego spotkania pominąć. To od założeń projektu zależy, w którym miejscu się zatrzymamy.

Miłego dzionka!