Semestrální práce „Car simulator“

Vypracoval: Artyom Voronin

Cíle:

1. Naprogramovat simulátor auta , které bude ovladatelné pomoci vstupu z klávesnice.
2. Modelovat natáčení kol auta, pomocí Ackermannově formule.
3. Použit vhodnou vizualizace

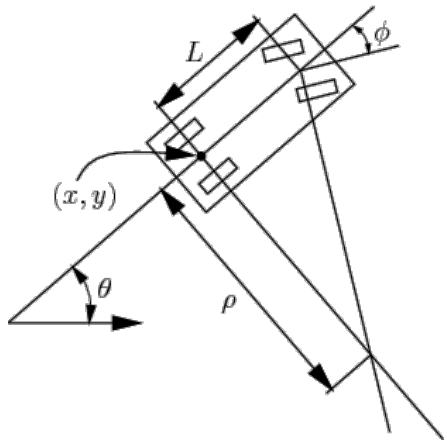
Obsah:

- Kinematika auta
- Simulace natáčení kol, pomocí Ackermannové formule
- Realizace

Kinematika auta:

Model:

Pro reprezentace auta jsem použil následující zjednodušený model auta, tzv. model kola:



obr. 1 model kola

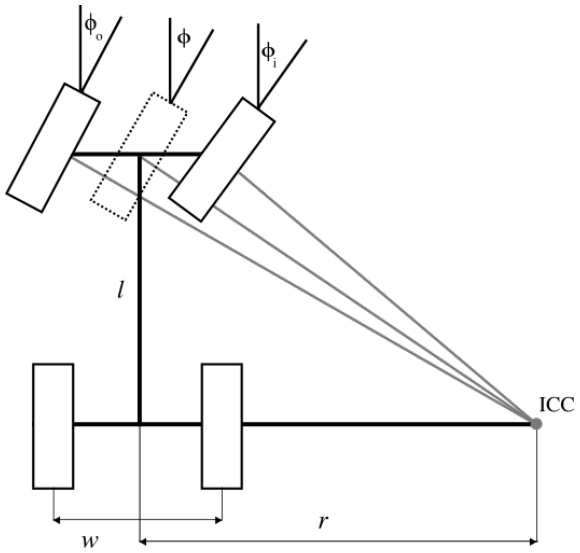
Pro simulace kinematiky byli použity následující vztahy:

$$\dot{x} = v \cdot cos(\theta)$$
$$\dot{y} = v \cdot sin(\theta)$$
$$\dot{\theta} = \frac{v}{l} tan(\phi) \approx \frac{v}{l}\phi$$

Model auta znázorněna červeným obdélníkem. Parametry auta byli empirický zvoleny i neodpovídají hodnotám reálných aut.

Simulace natočení kol, pomocí Ackermannové formule:

Ackermannová formulí řeší problematiku natočení předních kol vozidel. Hlavní idea, spočívá v tom, že každé kolo má se natáčet pod různým uhlem pří průchodů zatáčkou. Příslušný uhly lze vypočítat následujícím způsobem:



obr. 2 model pro stanovení uhlu

Pro uhly ϕ_i a ϕ_o platí

$$\phi_i = tan^{-1}(\frac{2l \cdot sin(\phi)}{2l \cdot cos(\phi) - 2w \cdot sin(\phi)})$$
$$\phi_o = tan^{-1}(\frac{2l \cdot sin(\phi)}{2l \cdot cos(\phi) + 2w \cdot sin(\phi)})$$

Realizace:

Simulator byl realizován pomocí programovacího jazyka Python.
Pro grafickou vizualizace byla zvolena knihovna Pygame, nicméně většina vypočtu tykajících se umístění a natočení objektu jako „obdélník“, „čára“ , apd. byli provedeny ručně, pomoci rotačních matic.
Byli zpracovaný zatím 2 mody:

1. Volná simulace
2. Simulace na draze

Pro instalace potřebných knihoven lze spustit soubor requirements.txt
s příkazového řádku:

$ pip install -r requirements.txt

Spustit skript simulator.py lze pomocí příkazového řádku:

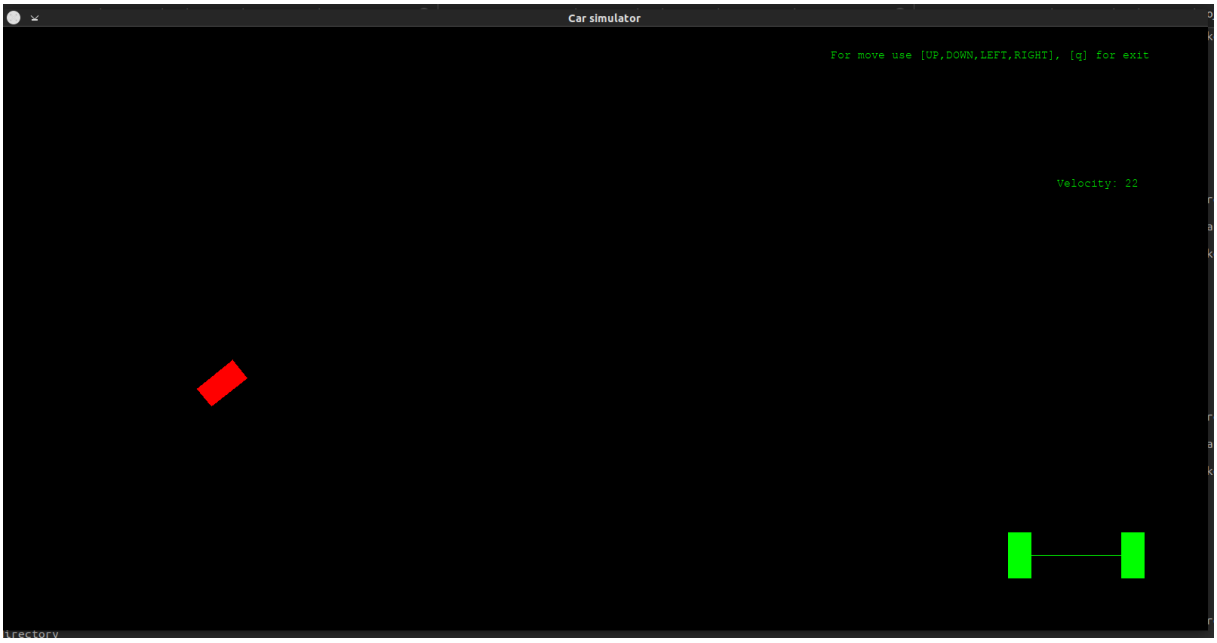
$python simulator.py

Následovně se zobrazí možností simulace v příkazovém řádku. A po vyberu modu se zobrazí okno se simulace. Ovládat lze pomocí šipek na klávesnice.

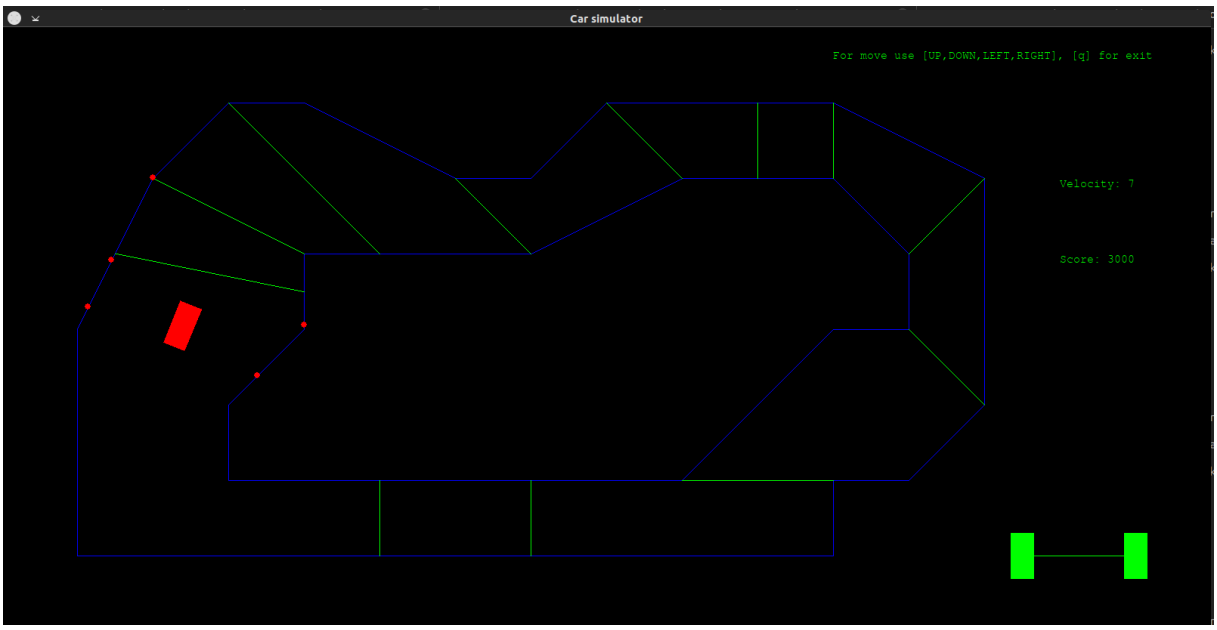
Architektura skriptu je následující:

- Import knihoven
- Globální proměny
- Globální funkce
- Class Ray, Rays – slouží k určení vzdáleností od „zdí“, potřebný pří implementace AI (Q learning, DQN)
- Class Track – vytváří dráhu
- Class Car – obsahuje v sebe kinematiku a další výpočty tykající se vozidla
- Class Game_player – provádí logiku simulace
- Class Game_start – inicializuje parametry simulace

Příklad simulace znázorněn na obrazcích 3, 4.



obr. 3 simulace volné jízdy



obr. 4 simulace jízdy po draze

Následující postup:

Vytváření AI, které bude řídit vozidlo.
Implementace závodu mezí hráčem a AI.

Zdroje:
https://medium.com/@ioarun/build-a-2d-robotic-car-96a19efae2c5
https://asawicki.info/Mirror/Car%20Physics%20for%20Games/Car%20Physics%20for%20Games.html
https://www.xarg.org/book/kinematics/ackerman-steering/
https://www.pygame.org/news
https://keras.io/
https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection