Traffic Simulator 🚦

Specifikáció

A cél

A feladat célja egy forgalmi szimulációs program létrehozása, amely modellezi és szimulálja a forgalmi lámpák és a járművek közötti interakciót egy kereszteződésben, majd erről statisztikát generál.

A modell

A **kereszteződés** két út kereszteződéséből áll, egy **út** két sávból áll (két irány), így tehát egyszerre legfeljebb két jármű tartózkodhat a kereszteződésben.

A **forgalmi lámpák** szabad és tilos jelzést mutatnak, ha az egyik út lámpája szabad jelzést mutat, a másiknak tilos jelzést kell mutatnia, ezek felváltják egymást 10 másodpercenként. Alapértelmezetten mindegyik lámpa tilos jelzést mutat.

A **járművek** a forgalmi lámpák szabad jelzésénél előre haladnak, tilos jelzésénél megállnak. Az utakon különböző fajta járművek vannak (személyautó, busz, teherautó, stb.), különböző tulajdonságokkal (pl.: szállítható személyek száma, plusz teher szállítás, stb.).

A **szimuláció** időt a program másodpercekben méri, szimuláció végén **statisztika** jelenik meg (pl.: hány autó haladt át a kereszteződésen, azon belül milyen féle, hány kg teher haladt át, hány ember haladt át, hányszor váltottak a lámpák).

Feltételezhetjük hogy a járművek csak előre haladhatnak, azonos sebességgel mozognak, a gyorsulástól és lassítástól eltekinthetünk.

A program

A program a felhasználótól szimulációs időt (szám) kér inputként másodpercekben (ha nem kap számot, hibát jelez és újra kéri a számot), ezután elkezdi a szimulációt, ehhez hasonló formátumban:

```
Enter simulation time in seconds: 11

Time: 11s <- folyamatosan számolja az időt

BUDAFOKI UT RED
IRINYI JOZSEF UTCA GREEN

Car passed through the intersection on IRINYI JOZSEF UTCA
Car passed through the intersection on IRINYI JOZSEF UTCA
Motorcycle passed through the intersection on IRINYI JOZSEF UTCA
Truck passed through the intersection on IRINYI JOZSEF UTCA
Car passed through the intersection on IRINYI JOZSEF UTCA
Truck passed through the intersection on IRINYI JOZSEF UTCA
Car passed through the intersection on IRINYI JOZSEF UTCA
Bus passed through the intersection on IRINYI JOZSEF UTCA
Bus passed through the intersection on IRINYI JOZSEF UTCA
Car passed through the intersection on IRINYI JOZSEF UTCA
```

IRINYI JOZSEF UTCA RED
BUDAFOKI UT GREEN

Car passed through the intersection on BUDAFOKI UT

…

STATISTICS

Traffic lamp cycles = 2

Cars passed = 6
Trucks passed = 1
Busses passed = 2
Motorcycles passed = 1
————————————————
Total vehicles passed = 10

Number of people passed ≈ 133

Total gross weight passed ≈ 21380 kg

Traffic Simulator 🚦

1. A cél

A feladat célja egy forgalmi szimulációs program létrehozása, amely modellezi és szimulálja a forgalmi lámpák és a járművek közötti interakciót egy kereszteződésben, majd erről statisztikát generál.

2. A modell 2.0

A **kereszteződés** négy út kereszteződéséből, és két forgalmi lámpából áll.

A **forgalmi lámpák** szabad és tilos jelzést mutatnak, ha az egyik út lámpája szabad jelzést mutat, a másiknak tilos jelzést kell mutatnia, alapértelmezetten mindegyik lámpa tilos jelzést mutat, két féle stratégiájú jelzőlámpa van. 1. időzítő lámpa: amely figyelembe veszi az utolsó váltás időpontját, 2. okos lámpa: amely figyelembe veszi a két irányból várakozó autók számát.

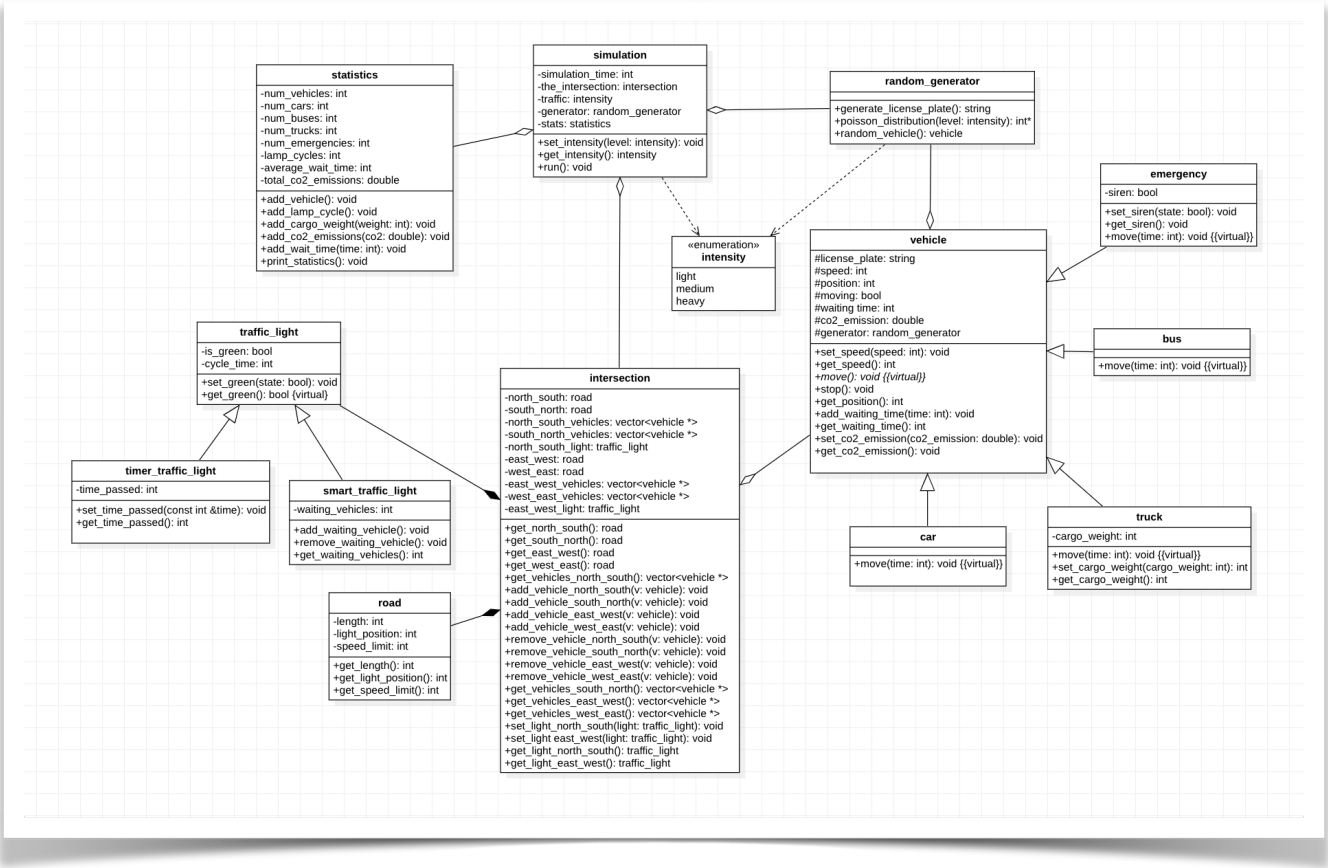
A **járművek** a forgalmi lámpák szabad jelzésénél előre haladnak, tilos jelzésénél megállnak. Az utakon különböző fajta járművek vannak (személyautó, busz, teherautó, vészhelyzeti autó) amelyeknek különböző tulajdonságai lehetnek, például különböző sebességekkel haladnak. A járművek Poisson-eloszlás szerint érkeznek az utakra, a forgalom intenzitás szintjétől függően.

A **szimuláció** időt a program másodpercekben méri, szimuláció végén **statisztika** jelenik meg (hány autó haladt át a kereszteződésen, azon belül milyen féle, hányszor váltottak a lámpák, mennyi volt a járművek átlagos várakozási ideje, mennyi a járművek széndioxid emissziója).

Feltételezhetjük hogy a járművek csak előre haladhatnak, a gyorsulástól és lassítástól eltekintünk.

3. A Terv

A szimuláció modellt a következő osztálydiagram ábrázolja



Traffic Simulator 🚦

4. Az Algoritmusok

Poisson eloszlás

A járművek Poisson-eloszlás[1] alapján generált időintervallumok alapján érkeznek az utakra, a következő Python-kód mutatja az algoritmus alkalmazását a feladatban:

```
simulation_time = 100
poisson_lambda = 10
arrival_times = []

while sum(arrival_times) <= simulation_time:
    u = random.uniform(0,1)
    x = round(-math.log(u) / poisson_lambda + 1)
    arrival_times.append(x)
```

Rendszám generálás 

Minden járműnek generálódik egy új formátumú magyar rendszám[2]:

```
if random.uniform(0,1) == 0:
    for i in range(2):
        license_plate += random.choice(vowels)
else:
    for i in range(2):
            license_plate += random.choice(consonants)

for i in range(2):
    license_plate += chr(random.randint(65, 90))


for i in range(3):
    license_plate += str(random.randint(0, 9))
```

[1] https://www.johndcook.com/blog/2010/06/14/generating-poisson-random-values/

[2] https://hu.wikipedia.org/wiki/Magyarországi_forgalmi_rendszámok#

Traffic Simulator 🚦

Dokumentáció

Felhasználói dokumentáció:

A program egy útkereszteződés forgalmát szimulálja, ahol két út kereszteződik. Minden úton két irányban lehet közlekedni előre. A forgalmat két irányjelző lámpa irányítja, amelyek 10 másodpercenként váltják egymást (zöld, piros). A járművek poisson-eloszlás szerint érkeznek az útra. A program a járművek közlekedését rendszámmal, jármű típussal, mozgás állapottal, mozgás iránnyal, út nevével és a jármű pozíciójával írja ki a kimenetre. Egy másodpercet várva soronként kiírja az adatokat. Ha a jármű zöld lámpán halad át, akkor a sort zöld színben jeleníti meg. Ha a jármű piroson áll meg, akkor piros színben jeleníti meg. Ha a jármű átment a kereszteződésen, és azon az úton a lámpa már piros, akkor sárga színben jeleníti meg a sort.
A program futtatása után (./traffic_simulator) először kérni fog egy szimuláció időt (másodperc), majd a forgalom intenzitását (1. light, 2. medium, 3. heavy).

A szimuláció végén a program statisztikát jelenít meg a szimulációról.

Fejlesztői dokumentáció:

A forgalom szimulációjához a Simulation osztály előre legenerál random számokat poisson-eloszlás szerint, mindegyik útnak az autók ebben a sűrűségben fognak megérkezni másodpercenként. Ezután poisson-eloszlás szerinti számszor generálódik egy random jármű minden úton, és hozzáadódik az adott úthoz. Közben a jármű hozzáadódik a statisztikához, beleértve az adatait is.

Ha az egyik úton piros a lámpa, azon az úton levő autóknak meg kell állniuk, kivéve, ha már elhagyták a kereszteződést. A másik úton a lámpa zöldre vált, és ott indulhatnak a járművek.
A program eközben statisztikát generál.

Ha a programnak előre megadjuk a szimuláció időt és az intenzitást, akkor tesztprogramként fog lefutni, 5 unit tesztet futtatva. Az első teszt a járművek hozzáadását teszteli a kereszteződéshez, a második a poisson-eloszlás elemeit vizsgálja, hogy ugyanannyiak-e, mint a szimuláció idő. A harmadik teszt a poisson-eloszlás helyességét vizsgálja, hogy egyenlő-e a középérték és a variancia a lambda értékével. A negyedik teszt az autók mozgatását teszteli, míg az ötödik teszt az autók megállítását vizsgálja.

A program részletes fejlesztői dokumentációja Doxygen segítségével lett generálva.

Traffic Simulator

Készítette Doxygen 1.9.3

1. Hierarchikus mutató

1.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

2. Osztálymutató

2.1. Osztálylista

Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

3. Fájlmutató

3.1. Fájllista

Az összes dokumentált fájl listája rövid leírásokkal:

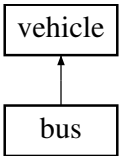
4. Osztályok dokumentációja

4.1. bus osztályreferencia

Busz típus.

```
#include <vehicle.hpp>
```

A bus osztály származási diagramja:



Publikus tagfüggvények

- **bus** ()

  konstruktor
- virtual void move (const int &time)

  Pozíció beállítása/mozgás.

További örökölt tagok

4.1.1. Részletes leírás

Busz típus.

4.1.2. Tagfüggvények dokumentációja

4.1.2.1. move() `void bus::move ( const int & time ) [virtual]`

Pozíció beállítása/mozgás.

Megvalósítja a következőket: vehicle.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

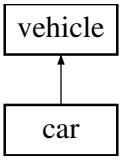
- vehicle.hpp
- vehicle.cpp

4.2. car osztályreferencia

Autó típus.

`#include <vehicle.hpp>`

A car osztály származási diagramja:



Publikus tagfüggvények

- **car** ()
  Autó konstruktor.
- virtual void move (const int &time)
  Pozíció beállítása/mozgás.

További örökölt tagok

4.2.1. Részletes leírás

Autó típus.

4.2.2. Tagfüggvények dokumentációja

4.2.2.1. move() `void car::move ( const int & time ) [virtual]`

Pozíció beállítása/mozgás.

Megvalósítja a következőket: vehicle.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

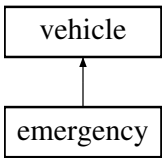
- vehicle.hpp
- vehicle.cpp

4.3. emergency osztályreferencia

Mentő típus.

`#include <vehicle.hpp>`

Az emergency osztály származási diagramja:



Publikus tagfüggvények

- **emergency** ()
  konstruktor
- bool get_siren () const
  szirena allapotanak beallitasa
- void move (const int &time)
  Pozíció beállítása/mozgás.

További örökölt tagok

4.3.1. Részletes leírás

Mentő típus.

4.3.2. Tagfüggvények dokumentációja

4.3.2.1. get_siren() `bool emergency::get_siren ( ) const`

szirena allapotanak beallitasa

Paraméterek

state	a szirena allapota

4.3.2.2. move() `void emergency::move ( const int & time ) [virtual]`

Pozíció beállítása/mozgás.

Megvalósítja a következőket: vehicle.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- vehicle.hpp
- vehicle.cpp

4.4. intersection osztályreferencia

Az utak kereszteződését reprezentáló osztály.

`#include <intersection.hpp>`

Publikus tagfüggvények

- **intersection** (road &north_south, road &south_north, traffic_light &north_south_light, road &east_west, road &west_east, traffic_light &east_west_light)

  Kereszteződés konstruktor.
- road & **get_north_south** () const

  Észak-Déli út lekérdezése.
- road & **get_south_north** () const

  Dél-Északi út lekérdezése.
- road & **get_east_west** () const

  Kelet-Nyugati út lekérdezése.
- road & **get_west_east** () const

  Nyugat-Keleti út lekérdezése.
- void add_vehicle_north_south (vehicle ∗const v)

  Autó hozzáadása Észak-Déli útra.
- void add_vehicle_south_north (vehicle ∗const v)

  Autó hozzáadása Dél-Északi útra.
- void add_vehicle_east_west (vehicle ∗const v)

  Autó hozzáadása Kelet-Nyugati útra.
- void add_vehicle_west_east (vehicle ∗const v)

  Autó hozzáadása Nyugat-Keleti útra.
- const std::vector< vehicle ∗ > & **get_vehicles_north_south** () const

  Észak-Déli úton lévő autók lekérdezése.
- const std::vector< vehicle ∗ > & **get_vehicles_south_north** () const

  Dél-Északi úton lévő autók lekérdezése.
- const std::vector< vehicle ∗ > & **get_vehicles_east_west** () const

  Kelet-Nyugati úton lévő autók lekérdezése.
- const std::vector< vehicle ∗ > & **get_vehicles_west_east** () const

  Nyugat-Keleti úton lévő autók lekérdezése.
- void **set_light_north_south** (const traffic_light &light)

  Észak-Dél lámpa beállitása.
- void **set_light_east_west** (const traffic_light &light)

  Kelet-Nyugat lámpa beállitása.
- traffic_light & **get_light_north_south** () const

  Észak-Dél lámpa lekérdezése.
- traffic_light & **get_light_east_west** () const

  Észak-Dél lámpa lekérdezése.

4.4.1. Részletes leírás

Az utak kereszteződését reprezentáló osztály.

4.4.2. Tagfüggvények dokumentációja

4.4.2.1. add_vehicle_east_west() `void intersection::add_vehicle_east_west ( vehicle *const v )`

Autó hozzáadása Kelet-Nyugati útra.

Paraméterek

v	Hozzáadandó autó

4.4.2.2. add_vehicle_north_south()

```
void intersection::add_vehicle_north_south (
        vehicle *const v )
```

Autó hozzáadása Észak-Déli útra.

Paraméterek

v	Hozzáadandó autó

4.4.2.3. add_vehicle_south_north()

```
void intersection::add_vehicle_south_north (
        vehicle *const v )
```

Autó hozzáadása Dél-Északi útra.

Paraméterek

v	Hozzáadandó autó

4.4.2.4. add_vehicle_west_east()

```
void intersection::add_vehicle_west_east (
        vehicle *const v )
```

Autó hozzáadása Nyugat-Keleti útra.

Paraméterek

v	Hozzáadandó autó

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- intersection.hpp
- intersection.cpp

4.5. random_generator osztályreferencia

Random generáló függvényeket tartalmazó osztály.

`#include <random_generator.hpp>`

Publikus tagfüggvények

- std::string **generate_license_plate** ()
  Random rendszám generáló
- std::vector< int > poisson_distribution (const intensity &level, const int &simulation_time)
  Random Poisson-eloszlás generáló
- vehicle * **random_vehicle** ()
  Random jármű generáló

4.5.1. Részletes leírás

Random generáló függvényeket tartalmazó osztály.

4.5.2. Tagfüggvények dokumentációja

4.5.2.1. poisson_distribution()

```
std::vector< int > random_generator::poisson_distribution (
        const intensity & level,
        const int & simulation_time )
```

Random Poisson-eloszlás generáló

Paraméterek

level	a forgalom intenzitása
simulation_time	a szimuláció ideje

Visszatérési érték

vektor, ami a másodpercenként generálandó autókat tartalmazza

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- random_generator.hpp
- random_generator.cpp

4.6. road osztályreferencia

Út osztály.

`#include <road.hpp>`

Publikus tagfüggvények

- **road** ()
  Út konstruktor.
- **road** (const int &length, const int &speed_limit, const int &light_position)
  Út inicializáló konstruktor.
- int **get_length** () const
  Út hossz lekerdezese.
- int **get_light_position** ()
  lámpa pozíció lekerdezese
- int **get_speed_limit** () const
  sebesség limit lekerdezese

4.6.1. Részletes leírás

Út osztály.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- road.hpp
- road.cpp

4.7. simulation osztályreferencia

Szimuláció osztály.

`#include <simulation.hpp>`

Publikus tagfüggvények

- **simulation** (const int &time, intersection &the_intersection, const intensity &level)
  Szimuláció konstruktor.
- statistics **get_stats** ()
  Statisztika lekérdezés.
- void run (bool readable)
  szimuláció futtatása
- void stop_vehicles (const std::vector< vehicle * > &vehicles, road &current_road, const std::string &road_name)
  autók megállítása
- void move_vehicles (const std::vector< vehicle * > &vehicles, road &current_road, const std::string &road_name)
  autók mozgatása
- void remove_vehicle (vehicle *current_vehicle, road &current_road)
  autók eltávolítása
- bool get_light (road &current_road)
  lámpa állapotának lekérdezése

4.7.1. Részletes leírás

Szimuláció osztály.

4.7.2. Tagfüggvények dokumentációja

4.7.2.1. get_light() `bool simulation::get_light ( road & current_road )`

lámpa állapotának lekérdezése

Paraméterek

current_road	az út, amin a lámpa van

4.7.2.2. move_vehicles() `void simulation::move_vehicles ( const std::vector< vehicle * > & vehicles, road & current_road, const std::string & road_name )`

autók mozgatása

Paraméterek

vehicles	az autók
current_road	az út, amin az autók vannak
road_name	az út neve

4.7.2.3. remove_vehicle() `void simulation::remove_vehicle ( vehicle * current_vehicle, road & current_road )`

autók eltávolítása

Paraméterek

current_road	az út, amin az autók vannak
road_name	az út neve

4.7.2.4. run()

```
void simulation::run (
bool readable = true )
```

szimuláció futtatása

Paraméterek

readable	ha igaz, akkor a szimulációt lassan írja ki a kimenetre

4.7.2.5. stop_vehicles()

```
void simulation::stop_vehicles (
const std::vector< vehicle * > & vehicles,
road & current_road,
const std::string & road_name )
```

autók megállítása

Paraméterek

vehicles	az autók
current_road	az út, amin az autók vannak
road_name	az út neve

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- simulation.hpp
- simulation.cpp

4.8. statistics osztályreferencia

Statisztika osztály.

`#include <statistics.hpp>`

Publikus tagfüggvények

- **statistics** ()
  statisztika konstruktor, beállít mindent 0-ra
- void add_vehicle (const vehicle &v)
  jármű hozzáadása
- void **add_lamp_cycle** ()
  lámpa ciklus számláló
- void add_cargo_weight (const int &weight)
  teher számoló
- void add_co2_emissions (const int &emissions)
  co2 számoló
- void add_average_speed (const int &speed)
  átlagos sebesség számoló
- int **get_average_speed** ()
  átlagos sebesség lekérdezése
- void **print_statistics** ()
  Statisztika kiírása.
- void **add_sirens** ()
  sziréna számláló

4.8.1. Részletes leírás

Statisztika osztály.

4.8.2. Tagfüggvények dokumentációja

4.8.2.1. add_average_speed() `void statistics::add_average_speed ( const int & speed )`

átlagos sebesség számoló

Paraméterek

speed	sebesség

4.8.2.2. add_cargo_weight() `void statistics::add_cargo_weight ( const int & weight )`

teher számoló

Paraméterek

weight	teher súlya

4.8.2.3. add_co2_emissions() `void statistics::add_co2_emissions ( const int & emissions )`

co2 számoló

Paraméterek

emissions	co2 emisszió

4.8.2.4. add_vehicle() `void statistics::add_vehicle ( const vehicle & v )`

jármű hozzáadása

Paraméterek

v	jármű

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- statistics.hpp
- statistics.cpp

4.9. traffic_light osztályreferencia

Lámpa osztály.

`#include <traffic_light.hpp>`

Publikus tagfüggvények

- **traffic_light** ()

  Konstruktor.
- void set_green (const bool &state)

  Lámpa állapotának beállítása.
- bool **get_green** () const

  Lámpa állapotának lekérdezése.
- void **cycle** ()

  Lámpa váltakozása.

Védett attribútumok

- bool **is_green**

  Lámpa állapota.
- int **cycle_time**

  Lámpa ciklus ideje.

4.9.1. Részletes leírás

Lámpa osztály.

4.9.2. Tagfüggvények dokumentációja

4.9.2.1. set_green() `void traffic_light::set_green ( const bool & state )`

Lámpa állapotának beállítása.

Paraméterek

state	a lámpa állapota

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

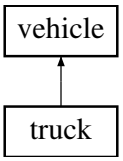
- traffic_light.hpp
- traffic_light.cpp

4.10. truck osztályreferencia

Teherautó típus.

`#include <vehicle.hpp>`

A truck osztály származási diagramja:



Publikus tagfüggvények

- **truck** ()

  Teherautó konstruktor.
- int get_cargo_weight () const

  Teher súly beallitasa.
- virtual void move (const int &time)

  Pozíció beállítása/mozgás.

További örökölt tagok

4.10.1. Részletes leírás

Teherautó típus.

4.10.2. Tagfüggvények dokumentációja

4.10.2.1. get_cargo_weight() `int truck::get_cargo_weight ( ) const`

Teher súly beallitasa.

Paraméterek

cargo_weight	a teher sulya

Teher súly lekérdezése

4.10.2.2. move() `void truck::move ( const int & time ) [virtual]`

Pozíció beállítása/mozgás.

Megvalósítja a következőket: vehicle.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

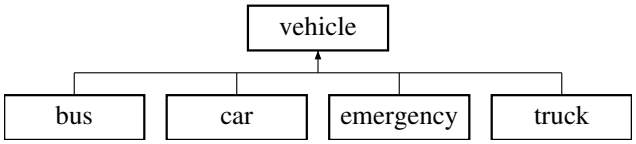
- vehicle.hpp
- vehicle.cpp

4.11. vehicle osztályreferencia

Jármű osztály.

`#include <vehicle.hpp>`

A vehicle osztály származási diagramja:



Publikus tagfüggvények

- **vehicle** (const vehicle_type &type)
  Jármű konstruktor.
- void **set_speed** (const int &speed)
  Sebesseg beállítása.
- int **get_speed** () const
  Sebesség beállítása.
- virtual void move (const int &time)=0
  Pozíció beállítása/mozgás.
- void **stop** ()
  Megállítás.
- bool **is_moving** () const
  Mozság állapota.
- int **get_position** () const
  Pozíció lekérdezése.
- void set_co2_emission (const int &co2_emission)
  CO2 kibocsátás beállítása.
- double **get_co2_emission** () const
  CO2 kibocsátás lekérdezése.
- vehicle_type **get_type** ()
  Jármű típusának lekérdezése.
- std::string **get_type_name** () const
  Jármű típus nevének lekérdezése.
- std::string generate_license_plate ()
  Rendszám generálása.
- std::string **get_license_plate** () const
  Renszám lekérdezése.
- virtual ~**vehicle** ()
  Jármű destruktor.

Védett attribútumok

- vehicle_type **type**

  Jármű típusa.
- int **speed**

  Jármű sebessége.
- double **position**

  Jármű sebessége.
- bool **moving**

  Jármű sebessége.
- std::string **license_plate**

  Jármű rendszáma.
- double **co2_emission**

  Jármű CO2 kibocsátása.

4.11.1. Részletes leírás

Jármű osztály.

4.11.2. Tagfüggvények dokumentációja

4.11.2.1. generate_license_plate() `std::string vehicle::generate_license_plate ( )`

Rendszám generálása.

Az új formátumú magyar rendszámok 2 random magánhanzó vagy mássalhangzóból, 2 random betűből, és 3 random számból állnak

4.11.2.2. move() `virtual void vehicle::move ( const int & time ) [pure virtual]`

Pozíció beállítása/mozgás.

Megvalósítják a következők: car, bus, truck és emergency.

4.11.2.3. set_co2_emission() `void vehicle::set_co2_emission ( const int & co2_emission )`

CO2 kibocsátás beállítása.

Paraméterek

co2_emission	a CO2 érték

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- vehicle.hpp
- vehicle.cpp

5. Fájlok dokumentációja

5.1. intensity.hpp

```
1 #ifndef INTENSITY_HPP
2 #define INTENSITY_HPP
3
5 enum intensity
6 {
7     light,
8     medium,
9     heavy,
10 };
11
12 #endif
```

5.2. intersection.hpp

```
1 #ifndef INTERSECTION_HPP
2 #define INTERSECTION_HPP
3
4 #include <iostream>
5 #include <vector>
6 #include "memtrace.h"
7 #include "road.hpp"
8 #include "traffic_light.hpp"
9 #include "vehicle.hpp"
10
12 class intersection
13 {
15     road &north_south;
17     road &south_north;
19     std::vector<vehicle *> north_south_vehicles;
21     std::vector<vehicle *> south_north_vehicles;
23     traffic_light &north_south_light;
24
26     road &east_west;
28     road &west_east;
30     std::vector<vehicle *> east_west_vehicles;
32     std::vector<vehicle *> west_east_vehicles;
34     traffic_light &east_west_light;
35
36 public:
38     intersection(road &north_south, road &south_north, traffic_light &north_south_light, road &east_west,
39                  road &west_east, traffic_light &east_west_light) : north_south(north_south),
       south_north(south_north),
40                                                                     north_south_light(north_south_light),
       east_west(east_west), west_east(west_east), east_west_light(east_west_light) {}
41
43     road &get_north_south() const;
45     road &get_south_north() const;
47     road &get_east_west() const;
49     road &get_west_east() const;
50
53     void add_vehicle_north_south(vehicle *const v);
56     void add_vehicle_south_north(vehicle *const v);
59     void add_vehicle_east_west(vehicle *const v);
62     void add_vehicle_west_east(vehicle *const v);
63
65     const std::vector<vehicle *> &get_vehicles_north_south() const;
67     const std::vector<vehicle *> &get_vehicles_south_north() const;
69     const std::vector<vehicle *> &get_vehicles_east_west() const;
71     const std::vector<vehicle *> &get_vehicles_west_east() const;
72
74     void set_light_north_south(const traffic_light &light);
76     void set_light_east_west(const traffic_light &light);
77
79     traffic_light &get_light_north_south() const;
81     traffic_light &get_light_east_west() const;
82
83     ~intersection();
84 };
85
86 #endif
```

5.3. random_generator.hpp

```
1 #ifndef RANDOM_GENERATOR_HPP
2 #define RANDOM_GENERATOR_HPP
3
4 #include <string>
5 #include <random>
6 #include "vehicle.hpp"
7 #include "intensity.hpp"
8 #include "memtrace.h"
9
11 class random_generator
12 {
13
14 public:
16     std::string generate_license_plate();
21     std::vector<int> poisson_distribution(const intensity &level, const int &simulation_time);
23     vehicle *random_vehicle();
24 };
25
26 #endif
```

5.4. road.hpp

```
1 #ifndef ROAD_HPP
2 #define ROAD_HPP
3
4 #include "traffic_light.hpp"
5 #include "memtrace.h"
6
8 class road
9 {
11     int length;
13     int light_position;
15     int speed_limit;
16
17 public:
19     road();
21     road(const int &length, const int &speed_limit, const int &light_position) : length(length),
       light_position(light_position), speed_limit(speed_limit) {}
23     int get_length() const;
25     int get_light_position();
27     int get_speed_limit() const;
28 };
29
30 #endif
```

5.5. simulation.hpp

```
1 #ifndef SIMULATION_HPP
2 #define SIMULATION_HPP
3
4 #include <iostream>
5 #include <iomanip>
6 #include <chrono>
7 #include <thread>
8 #include "road.hpp"
9 #include "traffic_light.hpp"
10 #include "vehicle.hpp"
11 #include "intersection.hpp"
12 #include "random_generator.hpp"
13 #include "statistics.hpp"
14 #include "memtrace.h"
15
17 class simulation
18 {
20     int simulation_time;
22     intersection the_intersection;
24     intensity traffic;
26     random_generator rand_gen;
28     statistics stats;
29
30 public:
32     simulation(const int &time, intersection &the_intersection, const intensity &level) :
       simulation_time(time), the_intersection(the_intersection), traffic(level) {}
34     statistics get_stats();
37     void run(bool readable);
42     void stop_vehicles(const std::vector<vehicle *> &vehicles, road &current_road, const std::string
       &road_name);
47     void move_vehicles(const std::vector<vehicle *> &vehicles, road &current_road, const std::string
       &road_name);
```

```
51      void remove_vehicle(vehicle *current_vehicle, road &current_road);
54      bool get_light(road &current_road);
55  };
56
57  #endif
```

5.6. statistics.hpp

```
1   #ifndef STATISTICS_HPP
2   #define STATISTICS_HPP
3
4   #include "vehicle.hpp"
5   #include "memtrace.h"
6
8   class statistics
9   {
11      int num_vehicles;
13      int num_cars;
15      int num_buses;
17      int num_trucks;
19      int num_emergencies;
21      int lamp_cycles;
23      int average_wait_time;
25      double average_speed;
27      double speeds;
29      int total_co2_emissions;
31      int total_cargo_weight;
33      int num_sirens;
34
35  public:
37      statistics()
38      {
39          num_vehicles = 0;
40          num_cars = 0;
41          num_buses = 0;
42          num_trucks = 0;
43          num_emergencies = 0;
44          lamp_cycles = 0;
45          average_wait_time = 0;
46          average_speed = 0;
47          speeds = 0;
48          total_co2_emissions = 0;
49          total_cargo_weight = 0;
50          num_sirens = 0;
51      }
54      void add_vehicle(const vehicle &v);
56      void add_lamp_cycle();
59      void add_cargo_weight(const int &weight);
62      void add_co2_emissions(const int &emissions);
65      void add_average_speed(const int &speed);
67      int get_average_speed();
69      void print_statistics();
71      void add_sirens();
72  };
73
74  #endif
```

5.7. traffic_light.hpp

```
1   #ifndef TRAFFIC_LIGHT_HPP
2   #define TRAFFIC_LIGHT_HPP
3
4   #include "memtrace.h"
5
7   class traffic_light
8   {
9   protected:
11      bool is_green;
13      int cycle_time;
14
15  public:
17      traffic_light() : is_green(false), cycle_time(10){};
20      void set_green(const bool &state);
22      bool get_green() const;
24      void cycle();
25
26  };
27
28
29  #endif
```

5.8. vehicle.hpp

```
1 #ifndef VEHICLE_HPP
2 #define VEHICLE_HPP
3
4 #include <iostream>
5
6 #include "memtrace.h"
7
9 enum vehicle_type
10 {
11     car_,
12     bus_,
13     truck_,
14     emergency_
15 };
16
18 class vehicle
19 {
20 protected:
22     vehicle_type type;
24     int speed;
26     double position;
28     bool moving;
30     std::string license_plate;
32     double co2_emission;
33
34 public:
36     vehicle(const vehicle_type &type) : type(type), speed(0), position(0), moving(false),
       license_plate(generate_license_plate()){} //)
38     void set_speed(const int &speed);
40     int get_speed() const;
42     virtual void move(const int &time) = 0;
44     void stop();
46     bool is_moving() const;
48     int get_position() const;
51     void set_co2_emission(const int &co2_emission);
53     double get_co2_emission() const;
55     vehicle_type get_type();
57     std::string get_type_name() const;
59     std::string generate_license_plate();
61     std::string get_license_plate() const;
63     virtual ~vehicle();
64 };
65
67 class car : public vehicle
68 {
69 public:
71     car() : vehicle(car_) {co2_emission = (rand() % 2 == 0) ? 120 : 0;} //lehet zero emission, elektromos
72     virtual void move(const int &time);
73 };
74
76 class bus : public vehicle
77 {
78 public:
80     bus() : vehicle(bus_) {co2_emission = 250;}
81     virtual void move(const int &time);
82 };
83
85 class truck : public vehicle
86 {
88     int cargo_weight;
89
90 public:
92     truck() : vehicle(truck_) {co2_emission = 300; cargo_weight = rand() % 3000;} // teher 0 es 3000
       kozott barmi lehet
96     int get_cargo_weight() const;
97     virtual void move(const int &time);
98 };
99
101 class emergency : public vehicle
102 {
104     int siren;
105
106 public:
108     emergency() : vehicle(emergency_) {co2_emission = 180; siren = rand() % 2 == 0;} // szirena random
       be/ki kapcsolva
111     bool get_siren() const;
112     void move(const int &time);
113
114
115 };
116
117 #endif
```