# Zadání

## Část 1: pasivní vizualizace (až 10 bodů)

Základní funkční požadavky (5 bodů)​ : Program bude možné spustit z příkazové řádky (dávkou Run.cmd) s volitelným celočíselným parametrem určujícím, který ze scénářů dopravní simulace (poskytnutých instancí třídy ​ Simulator metodou ​ getScenarios​ ) má být spuštěn. Nebude-li parametr zadán, spustí se scénář 0. Program zobrazí všechny klíčové elementy dopravní sítě (tj. křižovatky, silnice a vozidla na silnicích) spuštěného scénáře způsobem, který umožňuje posoudit jejich aktuální stav. Vizualizace se bude pravidelně aktualizovat.

Další požadavky: ​ Elementy budou rozmístěny v okně tak, aby se maximálně využil jeho prostor, ale zároveň nedošlo ke zkreslení souřadnic v nějakém směru (tj. Zobrazení zachovává poměr vzdáleností a úhly). Velikost okna musí být možné měnit a vizualizace se musí nové velikosti přizpůsobit. Součástí odevzdání bude dokumentace v předepsaném formátu.

**Část 2: interaktivní vizualizace**

Základní funkční požadavky (7 bodů)​ : Doplňte do vizualizace následující funkcionalitu:

1. V okně s vizualizací budou ovládací prvky umožňující změnu rychlosti běhu simulace, tj. umožňující zpomalovat nebo zrychlovat simulaci oproti reálnému času (viz hodnotu parametru předávaného metodě ​ nextStep​ třídy ​ Simulator​ ).

2. Po kliknutí na jízdní pruh silnice se otevře okno s vizualizací aktuálního a celkového počtu vozidel, která projela jízdním pruhem (viz metody ​ getNumberOfCarsCurrent a ​ getNumberOfCarsTotal ​ třídy ​ Lane​ ), v závislosti na čase od okamžiku spuštění simulace. Vizualizace musí plně využívat velikost okna, musí obsahovat popsané osy a musí být především správně. Velikost okna s grafem (resp. grafy) musí být možné měnit a graf (resp. grafy) se musí velikosti okna správně přizpůsobit.

3. Jednotlivé jízdní pruhy budou vhodně obarveny podle aktuální průměrné rychlosti vozidel (viz metodu ​ getSpeedAverage ​ třídy ​ Lane​ ) nebo aktuální obsazenosti (viz metodu ​ getNumberOfCarsCurrent ​ třídy ​ Lane​ ), přičemž mezi těmito způsoby lze přepínat přímo v programu.

## Řešení

Prvním krokem při tvorbě aplikace bylo rozdělit aplikaci na třídy pro lepší a jednodušší programovaní. Ve třídě View jsem vytvořila jednoduché GUI s dvěma tlačítky pro ovládaní Timeru a JComboBoxu pro nastavení scenáře přímo z aplikace. Ovladání tlačítek zajišťuje třída Controller, která volá příslušné metody na reakci od uživatele.

Dalším nejdůležitějším krokem v tvorbě samotné simulace dopravy bylo přepočítat všechny souřadnice z třídy Simulator, které byly zadány v metrech na souřadnice okna (pixely). V metodě computeModel2WindowTransformation(int, int) jsem zjistila max a min body okna, které se obnovují vždycky podle změny velikosti okna. Při každé změně scenáře Simulatoru, volá se metoda computeModelDimensions(), která se nachazí v třídě DrawPanel.java. Metoda prochazí přes pole silnic (sim.getRoadSegments()) a připravuje pro kreslení:

* Pole (List<MyLane> laneList) všech jizdních pruhů.
* Pole (List<Road> connectionList) pruhů, které spojují silnici v křižovatce.
* Pole (List<Point2D>points\_array) budů modelu, pro zjištění min a max bodů.

Poznamka: ty pole se inicializují po každé změně scenáře.

Taky bylo nutné určit vhodnou změnu měřítka pro posun ve směru X a Y. To je naimplementováno v hlavní vykreslovací třídě DrawPanel.java v metodě model2window(Point2D). Také se v této metodě určuje scale, který je dalé použivan k vykraslení silnic.

**private** Point2D model2window(Point2D p) {

**double** x = 0;

**double** y = 0;

**double** sx = (max\_X\_px - min\_X\_px) / (Xmax\_X\_in\_m - Xmin\_X\_in\_m);

**double** sy = (max\_Y\_px - min\_Y\_px) / (Xmax\_Y\_in\_m - Xmin\_Y\_in\_m);

scale = Math.*min*(sx, sy);

x = (p.getX() - Xmin\_X\_in\_m) \* scale;

y = (max\_Y\_px - min\_Y\_px) - (p.getY() - Xmin\_Y\_in\_m) \* scale;

**return** **new** Point2D.Double(x, y);

}

Po každém vykreslování okna se volá drawTrafficState(), která vykresluje křižovatky a auta. Křižovatky se vykreslují v metodě drawCrossRoad().

**private** **void** drawCrossRoad(Graphics2D g2d) {

drawRoadSegment(g2d);

CrossRoad[] cross = sim.getCrossroads();

**for** (CrossRoad crossRoad : cross) {

Lane[] lanes = crossRoad.getLanes();

**for** (Lane lane : lanes) {

connectLanes(lane, g2d);

}

}

}

Nejprvé metoda vykreslí všechné již připravené silnice tvořicí křižovatku z pole laneList a pak projde přes všechné křižovatky a propojí všechné silnici v ní.

Na konci se vykreslují auta, ve tvaru Line2D.

## Třidy

Celá aplikace je rozdělena na několik tříd, které se rozmístěny v balíku applikation. Struktura kódu připomíná softwarový vzor MVC. Applikace obsahuje kontroller, kde se vyvolavají Listennery pro grafické objekty. Ve třídě View jsou naimplementovany prvky GUI a jejich gettery a settery. Hlavní třida Main jen spojuje Controller a View a spouští se applikace.

* Main.java

Je hlavní třída projektu, ve která vytváří instance Controller a View. Také se tam zvaledují vstupní argumenty a naplní Jcombobox scénaři. Obsahuje metodu main.

* CrossRoadController.java

Controller kontroluje akci po stísknuti tlačítek a obsahuje timer Simulatoru. ​Timer zajišťuje pravidelné překreslování okna v intervalu 100 ms. Při změně scenaria v JcomboBoxu se vytvoří nová instance simulatoru a naimlementují se nové max, min body scenaria a nově se vyplní pole silnic (laneList, connectionList).

* View.java

Třída která odpovída za zobrazení prvků na okně. Obsahuje gettery a settery každého grafického objektu.

* DrawPanel.java

Je hlavní třída semestrální práce, ktera vykresluje dopravní simulace. Dědí od JPanel z Java Swing.

* + drawTrafficState(Simulator, Graphics2D)
  + computeModelDimensions()
  + computeModel2WindowTransformation(int, int)
  + model2window(Point2D)
  + drawCrossRoad(Graphics2D)
  + drawRoadSegment(RoadSegment, Graphics2D)
  + drawBackwardLanes(double, double, double, double, RoadSegment, Graphics2D, int)
    - Najde nové souřadnice pro zpětné pruhy a přída je do List<Road> a List<MyLine>.
  + drawForwardLanes(double, double, double, double, RoadSegment, Graphics2D, int)
    - Najde nové souřadnice pro pruhy jdoucí druhým směrem a přída je do List<Road> a List<MyLine>.
  + connectLanes(Lane, Graphics2D)
    - Vykreslí všechny silnice z List<Road> a propojí křižovatku. Pro hledání silnic v poli byl použit stream vyhledávaní v poli.
  + drawLane(Point2D, Point2D, int, Graphics2D)
  + drawCar(Point2D, double, int, int, Graphics2D)
  + drawComponent(Graphics2D)
    - Vykreslí celý scenař.
  + print(Graphics, PageFormat, int)
    - Metoda od interface Printable, která připaví a vytiskne křižovatku.
  + updateDataSet(double)
    - Každy krok časovače do HashMap<Lane, List<DataSet>> se přidá nový zaznam dat pro vytvoření grafu.
* Road.java

Pomocná třída pro ukladání souřadnic pro spojení křižovatky.

* MyLine.java

Pomocná třída pro ukladání souřadníc všech silnic křižovatky.

* GraphView.java

Třída, kteŕa se stará o připravení a zobrazení grafu v okně.

* + makeLineChart(XYSeries)
    - Vytvoří spojnicový graf k jedné silnici.
  + setData(Lane, HashMap<Lane, List<DataSet>>)
    - Nastaví data pro jednu vybranou silnicí.
  + setAllData(HashMap<Lane, List<DataSet>>)
    - Nastaví data pro všechné silnici v křižovatce.
  + makeAllLineChart(List<XYSeries>)
    - Vytvoří spojnicový pro všechné silnici v křižovatce.
* GraphController.java

Třída obsahující vytvoření nového okna s grafem a inicializace GUI prvků.

* DataSet.java

Pomocná třída pro uložení dat pro zobrazení v grafu.

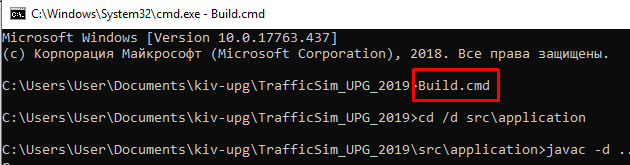
## Knihovny

1. jcommon-1.0.23.jar
2. jfreechart-1.0.19.jar
3. jfreesvg-3.3.jar
4. TrafficSim.jar

## Uživatelská dokumentace

## Instalace

1. Překlad programu pomoci příkazu Build.cmd v příkazpové řadce.



1. Vytvoření spustiteného souboru run.jar



## Popis GUI

## 

1. **Menu**
   1. File/Print map – Vytiskné scenař
   2. File/Save to bitmap – Uloží scenař do obrazku.
   3. File/Save to SVG – Uloží scenař do SVG obrazku.
2. **Top panel**
   1. Combobox - výběr scenaria.
3. **Left panel**
   1. „Speed Average“ – obarvení silnic podle jejích střední rychlostí.
   2. Number of cars – obarvení silnic podle actualního počtu vozidel.
   3. „Select road“ – Kliknutí na silnici pro zobrazení grafu vybrané silnici.
   4. „Show cars speed“ – Zobrazení popisek s rychlostí vedle aut.
   5. „Show graph“ – Zobrazení grafu všech silníc.
   6. Allow zoom and pan – Povolí posouvat křižovatku a zoom.
   7. Zoom in
   8. Zoom out

## Závěr

Zadaní práce bylo splněno. Při práce jsem se naučila používat knihovnu Java Swing. V příští etapě by bylo vhodné opravi výběr silnice po kliknutí myší. Aktuálně je funkce aktivní, ale pro výběr silnice je možné kliknout pouze na její spodní okraj.