1. Vytvoreni projektu ve vyvojovem prostredi s knihovnami “tactical village package”. Adresar resources je potreba umistit do “rootu” projektu.

2. Vytvorit kod pro ovladani auta: **MujImplementovanyAgent** extends **WaypontCarAgent** (!POZOR na preklep v prvnim release knihoven!)

   Rozsireni si vynuti implementaci metod **waypointReached** pro callback pri dosazeni waypointu a **ready** volane po inicializci prostredi pri stardu simulace

```
package pah;

import tacv.universe.entity.car.WaypontCarAgent;
import tacv.universe.environment.AgentScoutEnvironment.AgentScoutEnvironmentHandler;
import tacv.util.Point;

public class MujImplementovanyAgent extends WaypontCarAgent{
    public MujImplementovanyAgent(String name, AgentScoutEnvironmentHandler handler) {
        super(name, handler);
    }

    @Override
    protected void waypointReached(Point point, Point point1) {
    }

    @Override
    protected void ready() {
    }
}
```

3. Vytvorit creator **MujNovyCreator** extends **AbstractPahCarCreator**:

```
package pah;

import tacv.AbstractPahCarCreator;
import tacv.universe.entity.car.AbstractCarAgent;

public class MujNovyCreator extends AbstractPahCarCreator {

    @Override
    public AbstractCarAgent getCarAgent() {
        return new ClickMovingCarAgent("car", universe.getEnvironment().handler());
    }
}
```

4. Spustit (pozor na “class name clash” s pripadnym creatorem a implementovanym agentem v knihovnach):

```
public static void main(String[] args) {

    CreatorFactory.createCreator(new String[]{MujNovyCreator.class.getName()}).create();
}
```

V tuto chvili vam prostredi bezi s inicializovanym jednim vozidlem, ktere najdete ve spodni casti mapy. Zobrazi se dve okna – jedno 2D a jedno s 3D pohledem. V obou se lze pohybovat pomoci mysi a zoomovat pomoci kolecka na mysi. Ve 3d lze pro rychlejsi navigaci pouzit klavesy WASD. Ve 2D okne lze zobrazit help pomoci klavesy F1.

5. Vytvorit interakci “go to klick” v **MujImplementovanyAgent** pomoci layeru ve 2D vizualizaci (umistime do konstruktoru agenta):

```
    super(name, handler);
    VisManager.registerLayer(VisualInteractionLayer.create(
            new VisualInteractionLayer.VisualInteractionListener() {

        @Override
        public void interact(double x, double y) {
            goToPoint((float) x, (float) y);
        }
    }));
}
```

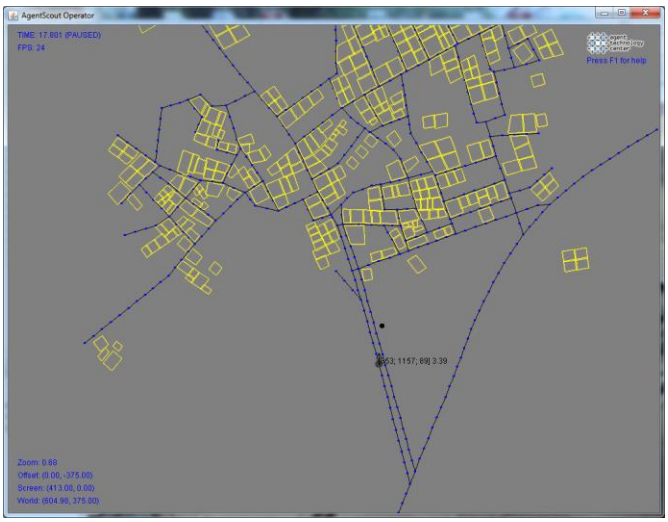
Musime take implementovat metodu goToPoint:

```
private void goToPoint(float x, float y) {
    float altitude = handler.getAltitude(x, y);
    Point point = new Point(x, y, altitude + 2);
    actGoToWaypoint(point);
}
```

Metoda **actGoToWaypoint(point)** je metoda z **WaypontCarAgent** zadavajici cilovy bod pro regulator auta. Po dosazeni ciloveho bodu (za predpokladu, ze se auto nezasekne po ceste) je volan callback **waypointReached**.

6. Pro zastaveni automobilu po dosazeni ciloveho bodu je treba jeste auto “zabrzdit”:

```
protected void waypointReached(Point wayPoint, Point position) {
    System.out.println("waypointReached: " + wayPoint + position);
    actHalt();
}
```



7. **Ukol** – rozsirte funkcionalitu o moznost naklikat seznam cilu. Aoto pak tyto cile projede postupne.

8. **Ukol** – modifikujte navigaci na cil pomoci hledani v StreetGraphu

9. **Uzitecne pomucky:**

Grafu ulic je k dispozici pomoci

```
StreetGraph streetGraph = handler.getMap().getStreetGraph();
```

Pristup k uzlum a hranam grafu pomoci (pozor, graf je neorientovany, tudiz sourceNode a targetNode nemaji semantiku zdroje a cile, ale jen oznacuji dva konce jedne hrany – je nutno tedy “rucne” zjistit jaky je druhy konec hrany k dotazovanemu nodu)

```
Set<DefaultWeightedEdge> edgesOf = streetGraph.edgesOf(MapPosition position);
MapPosition edgeSource = streetGraph.getEdgeSource(DefaultWeightedEdge edge);
MapPosition edgeTarget = streetGraph.getEdgeTarget(DefaultWeightedEdge edge);
```

Aktualni pozici a nejblizsi node grafu k aktualni pozici

```
Point senseMyPosition = carSensor.senseMyPosition();
MapPosition nearestNode = streetGraph.getNearestNode(Point point);
```

Plan auta lze zobrazit napr. pomoci PointLayeru nebo LineLayeru (`cz.agents.alite.vis.layer.terminal`), napr.

```
VisManager.registerLayer(PointLayer.create(new PointElements() {
    @Override
    public Iterable<? extends cz.agents.alite.vis.element.Point> getPoints() {
        // Insert your set of points here
        throw new UnsupportedOperationException("Not supported yet.");
    }
    @Override
    public Color getColor() {
        // Insert your point color here
        return Color.BLUE;
    }
    @Override
    public int getStrokeWidth() {
        // Insert your stroke width here
        return 2;
    }
}));
```

```
VisManager.registerLayer(LineLayer.create(new LineElements() {
    @Override
    public Iterable<? extends Line> getLines() {
        // Insert your set of lines here
        throw new UnsupportedOperationException("Not supported yet.");
    }
    @Override
    public Color getColor() {
        // Insert your line color here
        return Color.ORANGE;
    }
    @Override
    public int getStrokeWidth() {
        // Insert your stroke width here
        return 1;
    }
}));
```