GAMA platform: exercice 2 - Road traffic

Alexis Drogoul *(a, b)*, **Benoit Gaudou** *(c)*, **Patrick Taillandier** *(d)*

(a) UMI 209 UMMISCO, IRD / UPMC

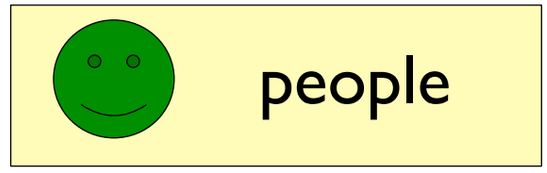
(b) JEAI DREAM, IRD / Université de Can Tho

(c) UMR 5505 IRIT, Université de Toulouse 1 / CNRS

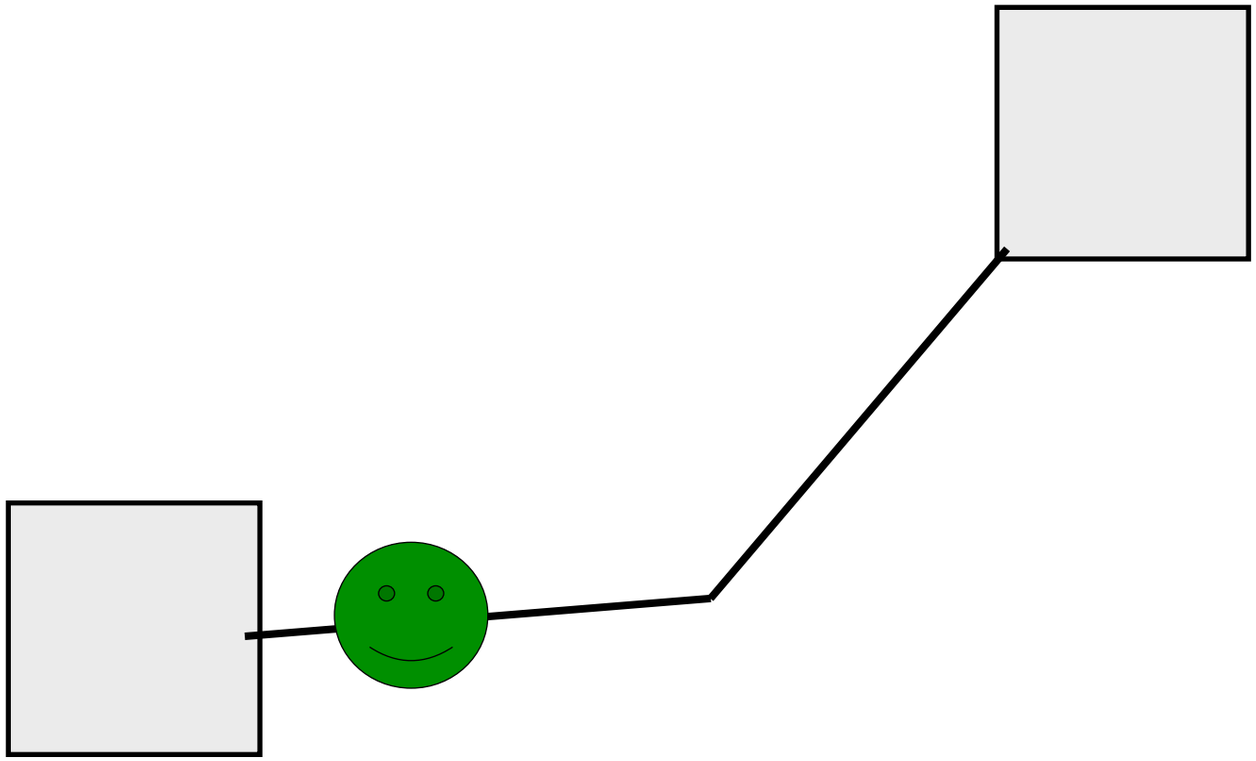
(d) UMR 6266 IDEES, Université de Rouen / CNRS

Introduction: a basic road traffic model

- People are moving from building to building

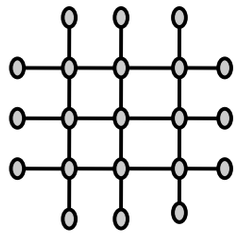


Stay for a certain time: at each simulation step, probability to leave: *proba_stay*



Step 1: global variables

- **Exercice 1**: In the world agent defines 2 new variables :
  - road_graph: type: graph
  - step: type: float, init value: 1#mn



The symbol # allows also to precise the unity of a value

step is a built-in variable of the world that precise the duration of one simulation step (by default: 1 second). It is possible to override it to modify its value

- **Exercice 2**: In the *bounds* section of the world agent:
  - choose the « file » type to define the bounds size of the world
  - As path, choose the buildings.shp shapefile that is located in the folder includes

Step 2: building species

- **Exercice 3**: define inside the world a new species called « building »

- **Exercice 4**: Add an aspect to the *building* species called « default »:
  - Add a layer called « geom » that draws the geometry of the agent with a gray color, with a depth of 30 (height of the building), a the following texture: ["../includes/roof_top.png","../includes/texture5.jpg"]

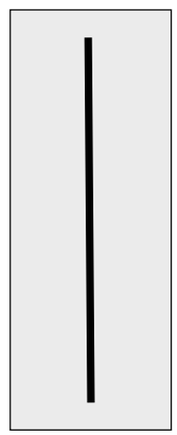
- **Exercice 5**: in the display « my_display » add two new layers:
  - A layer called « Image » where the image (type: « image ») ../includes/soil.jpg is displayed
  - A layer called « Building » in which the species « building » is displayed with aspect « default » (the one previously defined), set refresh to false.
  - set the display type to *OpenGL*

- **Exercice 6**: In the init section of the *world* agent create the building agents from the « ../includes/buildings.shp » shapefile

- **Exercice 7**: define inside the world a new species called « road »

- **Exercice 8**: Add an aspect to the *road* species called « default »:
- Add a layer called « geom » that draws the geometry of the agent plus a buffer of 5 meters with a black color:

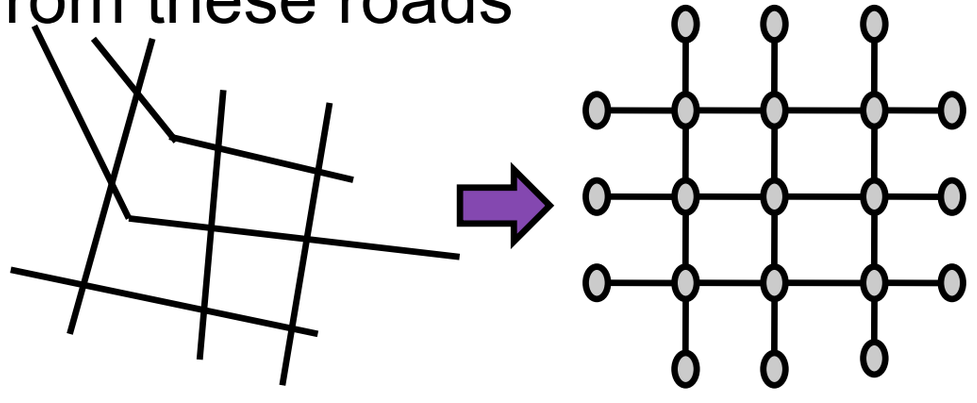
```
shape + 5
```



- **Exercice 9**: in the display « my_display » add one new layer:
- A layer called « Roads » in which the species « road » is displayed with aspect « default » (the one previously defined), set refresh to false.

- **Exercice 10**: In the init section of the *world* agent create the road agents from the « ../includes/roads.shp » shapefile and init the value of the *road_graph* variable from these roads

The **as_edge_graph(***list of polylines***)** operator allows to build a graph from a list of polylines



- **Exercice 11**: define inside the world a new species called « people » with one skill « moving » and with 3 variables:

  - *target*: type: point
  - *speed*: type: float, init value: 5 km/h
  - *proba_stay*: type: float, init value: 0.05

A *skilll* is a module integrating variables and actions coded in Java

With the *moving skill, t*he *people* agents will have some supplementary variables (*speed*, *heading*, *destination*) and actions (*follow*, *goto*, *move*, *wander*)

- **Exercice 12**: at initialization (init block) of the *people* agents, place the agent (location) to a random location inside the building

- **Exercice 13**: at initialization (init block) of the *world* agent, create 1000 *people* agents.

Step 5: people species display

- **Exercice 13**: Add an aspect to the *people* species called « Geom »:
- Add a layer called « Geom » that draws a *cube* of side size *10* of blue color.

- **Exercice 14**: in the display « my_display » add a new layer called « People », in which the species « people » is displayed with aspect « Geom »

Step 6: people choose target reflex

- **Exercice 15**: Add a reflex to the *people* species called « choose_target »:

➡ the reflex is activated when it has no target (*target* = *nil*) and with the probability *proba_stay*

➡ the reflex set the value of the *target* variable by one point randomly drawn from one of the buildings

Step 7: people move reflex

- **Exercice 16**: Add a reflex to the *people* species called « move »:

- the reflex is activated when it has a target (*target != nil*)

- First, the move, for that it applies the *goto* action with for *target:* argument, its *target* variable and for the *on:* argument the *road_graph*

- Then, if it is arrived at its target point, i.e. location = target, it sets its *target* to nil.

The ***goto*** action: the agent computes (once) the shortest path using the graph given by the *on:* facet then use this path for moving (in particular for the computation of the distance travelled)

Step 8: road use display

- **Exercice 17**: add a new variable for road agents called « nb_people » :
  - type: int
  - *update*: nb of people at a distance of 0.1 m

A variable with a *update* facet is re-computed at every steps

The *a_species* **at_distance** *distance* returns all the agents of the species *a_species* at a distance *distance* from the geometry of the agent

- **Exercice 18**: Add an aspect to the *road* species called « road_use »:
  - Add a layer called « Road use » that draws the *shape* of the road plus a buffer of size *(1 + nb_people)* with red color

- **Exercice 19**: Define a new display called « road_use » in which the species « road » is displayed with aspect « road_use »

Step 9: traffic jam management

- **Exercice 20**: add two new variables for *road* agents called
- *capacity*: type: int, init value: perimeter / 30.0; min value: 1
- *speed_coeff*: type: float, init value: 1.0, update: if nb_people = 0, then 1.0, otherwise capacity/nb_people; min value: 0.3, max value: 1.0

The min and max value of a variable allows to ensure that the variable will never be lower than the min value and higher than the max value

- **Exercice 21**: Add a global variable (world agent):
- *speed_map*: type: map, update:

```
road as_map (each::each.shape.perimeter / each.speed_coeff);
```

A map is a classic structure of data: a list of pairs (key, value): to each (unique) key is associated a value

The *list* **as_map** *expression_key:expression_value* operator allows to build a map from a list by applying expression on the elements of the list

- **Exercice 22**: Add, at the end of the init section of the world agent the initialization of the *speed_map* variable (same expression than the *update* facet)
- **Exercice 23**: modify the *move* reflex of the *people* species in order to take into account the traffic jam: add to the application of the *goto* action a new facet called **move_weights:** with for value: *speed_map*

The ***move_weights*** facet allows to differentiate the weights used for the shortest path computation and the ones used for moving

Conclusion of model: it is already finished!