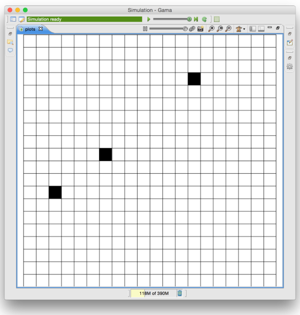
Shortest Path on Grid by diffusion

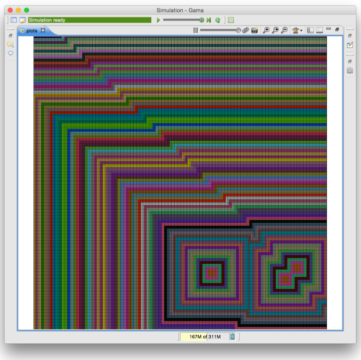
We consider a grid of plots on which will move people agents. Some of the plots will be possible targets of people and some will be obstacles (plots on which people cannot move). The aim of the exercise is to compute on each plot its distance to targets in order to help people to move to their target with the shortest path.

Questions

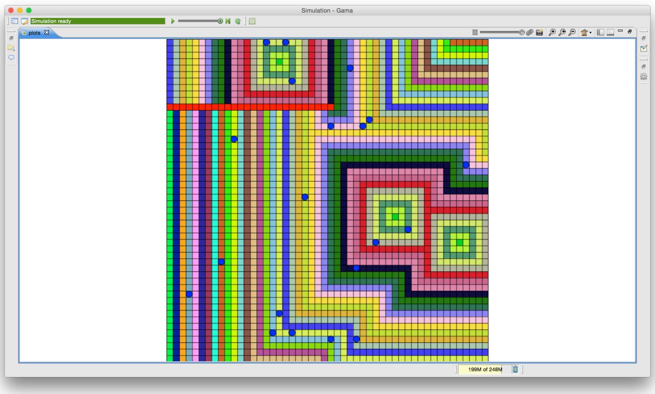
1. (Question 1) Implement a first model containing only the grid of plots with 2 possible covers (target or free). Display this model, target plot will be black, free plot will be white. The target plots are chosen randomly among all the plots. The number of targets will be a parameter of the simulation. The size of the grid is 20x20.



2. (Question 2) For each plot, compute and store its distance to the closest target plot, using a dedicated GAML operator. The plots with the same distance to any target should be displayed with the same color.
3. (Question 3) Increase the number of grid cells to 100, and observe how slow is the previous computation.
   We will define a faster algorithm by using diffusion of the distance over the plots from the targets to the furthest ones.
   The idea is:
   (i) to compute first the distance of the set of targets to the closest target (themselves), which is thus 0,
   (ii) then their neighbours (with a distance not yet defined) will be at a distance 1,
   (iii) their neighbours at 2 …
   Implement this algorithm (hits in Note 1).

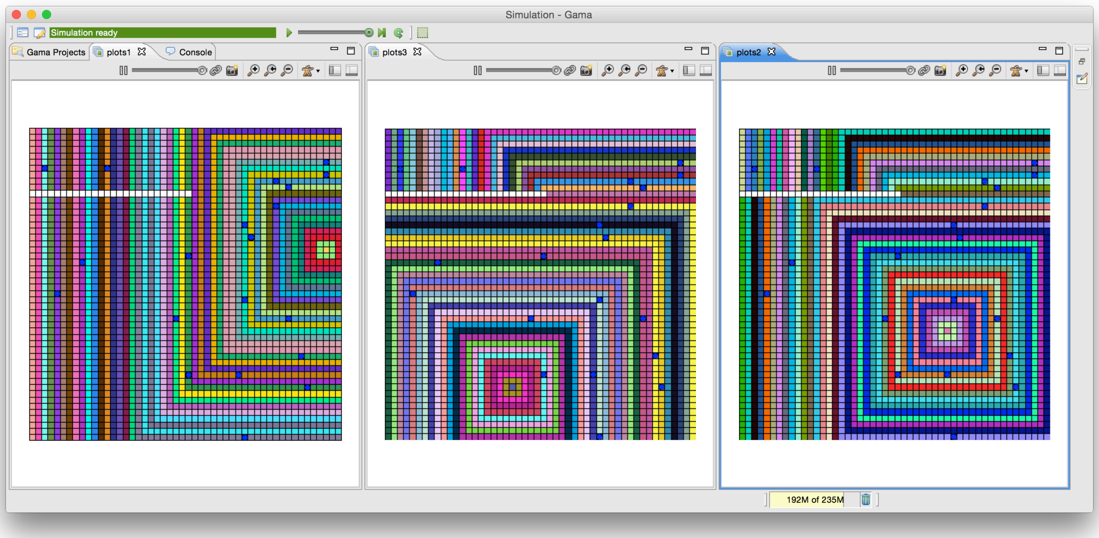


4. (Question 4) Create 20 people agents, locate them on plots and give them the behavior to move to the closest target (using the distance variable in plots) and to stop them when they reached a plot with a distance of 0;
5. (Question 5) Add a new type of covers for plots: they can also be obstacle. At initialization, initialize the cover plots with indices from {0,10} to {grid_size/2,10} with the value "obstacle" and a red color.
   Modify the algorithm of Q3 to have a shortest path to the closest target taken into account obstacles.



Additional question

6. (Question 6) Each plot now stores its distance to every target plot (with an associated color). People agents have now a random target among these target plots and will move to it. Use several displays to show the distance gradient to each target plot.



Notes

1. (Question 3) To compute the closest distance to a target, we follow the following algorithm (from the question 1 model):
   a. The distance attribute for each plot is initialized to -1.
   b. In the global init, after having created the target plots:
      i. Create the list of plot nextPlots and initialize it with the target plots.
      ii. Declare and initialize (to 0) an int variable (dist) that will contain the distance to targets.
      iii. Do a loop while nextPlots is not empty
         1. Initialize a random color
         2. Ask to each plot of nextPlots to set their color to this random color and their distance to dist.
      iv. Update nextPlots : nextPlots gets the set of plots that are neighbors to plots of nextPlots and have a distance attribute value to -1.
      Be careful: 2 plots can have common neighbors (which should be removed to the computation).
      v. Increment by 1 the dist variable
2. (Question 4) To avoid computing several times the neighborhood of the cells, add an attribute to each plot that contains the list of neighbor cells (and computed at the initialization of the simulation).
3. (Question 5) Be careful at the order of initialization in the plots: obstacles should be created first and then the targets on plots that are not obstacles.
   A set of neighbors is computed for each plot and is used for the computation of the distance and the people moves. It would be efficient to remove the obstacle plots from this neighbors set.
4. (Question 6) A map can be used as data structure to store the distance to each target and a color associated to each target.