PROJECT 4 DESIGN DOCUMENT

**Introduction**

This project develops a simulation model for the spread of a virus within a limited area and with a predefined number of people. The area is divided into regions and people move from one region to another in random directions.

Useful variables to better understand the implementation:

* N: total number of individuals.
* P: number of processes.
* R: number of regions:
* Ax: length of the Area.
* Ay: width of the Area.
* Rx: length of each region.
* Ry: width of each region.
* t: time after which the position of individuals is recomputed.
* d: spreading distance.
* v: moving speed of each individual.

Some of the individuals are already infected and can infect others. If a susceptible individual stays close for more than 10 minutes to an infected individual it becomes infected. After 10 days of infection an individual becomes immune. After 3 months of immunity an individual returns susceptible.

At the end of each simulated day the number of individuals for each region is shown and the health state of those individuals.

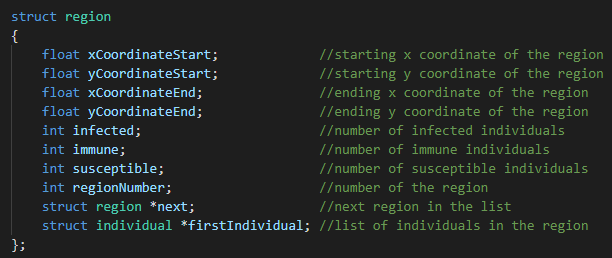
**Implementation**

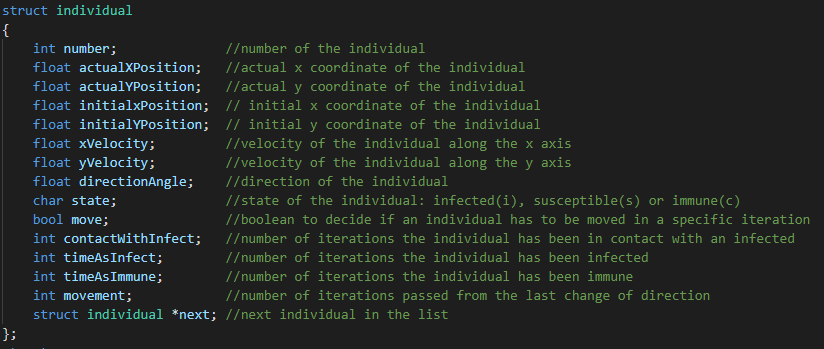
Given the fact that the computation is performed by P different processes, the first useful computation is to divide the individuals and regions among the different processes. Each process is responsible for a certain number of regions, and the individual are initially equally divided in each region. So in the beginning each process will have:

number of individuals

number of regions

The eventual rest of the division is given to the last process. Each process will create a linked list containing each region it deals with. Inside each region there is another linked list containing the individuals useful for that region. This includes not only the individuals directly inside the region, but also those that are near the border.



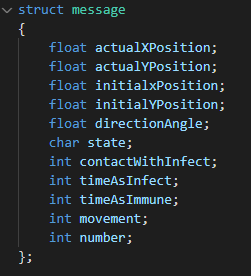


After this initial computation, every process creates **two arrays**. One is used to save all the individuals that has to be sent to another process, and the second for saving the individuals received from another process.

Once this operation is completed, the main cycle that marks the operation of each process begins. Since the time t represents the time that passes in the simulation after which the position of the individuals must be updated, a single cycle represents that time t of simulation.

The operations that each process does in a cycle are:

* Checking if a day has passed. Since one cycle is t seconds, a day is a number of cycles equal to . If a day has passed each process prints the state of each individual in each region.
* For each individual in each region:
  + Check if there’s a change in the condition of the individual. This can happen if one of the conditions depicted in the introduction is met.
  + Compute the updated position of the individual. The movement is a simple uniform motion in one random direction. So at each time the new position is. . Where vx and vy are simply the velocity of the individual in the corresponding direction. If, in the computation, an individual is found outside the area, his position is recomputed with a new direction in the opposite part of the area.
  + Check if the individual is outside the area of influence of the region; if that’s the case remove the individual from the list and start considering the next in the list.
  + Check if the individual is near the border of the region it is contained. In this circumstance the computations follow by understanding the number of regions the individual is close to (excluding the region the individual is currently in this number cannot be greater than 3). Then the objective is to find those regions. At first the process considers the regions in its list; if the regions are found the individual is added to their list (or updated if it was already in the list). If any of those regions is not in the list then another process needs to be informed, in this case a **message** is created and added to the corresponding array. The message contains all the useful information about the individual.



* Once updated all the individuals, the process checks the array of messages.
  + If there is at least one message in the array, the process sends the array to all the other processes using an MPI\_Isend.
  + Resets the array.
* The process then checks if there are messages coming from the other processes using an MPI\_Iprobe.
  + If there is a message receives it using MPI\_Recv and checks for each of the individuals contained if they are in the area of influence of any of its regions. Then adds them to the list or updates their information.
* In this situation each region of each process contains in its list only the individuals that are useful for their computation, which include:
  + Checking if the susceptible individuals are near an infected one.
  + Updating the number of susceptible, infected, and immune individuals contained in each region.
* Before starting the next cycle there is an MPI Barrier, to make sure that each process starts each cycle in the same moment.