**PAC-MAN**

**Objectives and development’s principles**



**I – Original objectives (Master Branch, 1.0.20.X Version)**

Our (Aleam and Atero) first and main is rebuild from *ex nihilo* the 1980 Pac-Man game. But not necessarily with the same development nor game environment.

The game will in first place be executable on Windows machines.

For the first version (let’s call her alpha for instance), we want to have these features developed and entirely functionals:

* Loading of a few prepared mazes (map, walls, collectables, original position of Pac-man) ;
* 4 ghosts’ presence, with uncertain movement (Pokey / Clide -like)
* Pac-Man’s movements management
* Score
* Pac-Man’s Lives
* Consumption of collectables (normal points, fruits, special points)
* Ghosts’ regeneration
* Single and Simple Mode : While Pac-Man is living, he can explores the maze, finish it and try it again.

**II – Development’s Method**

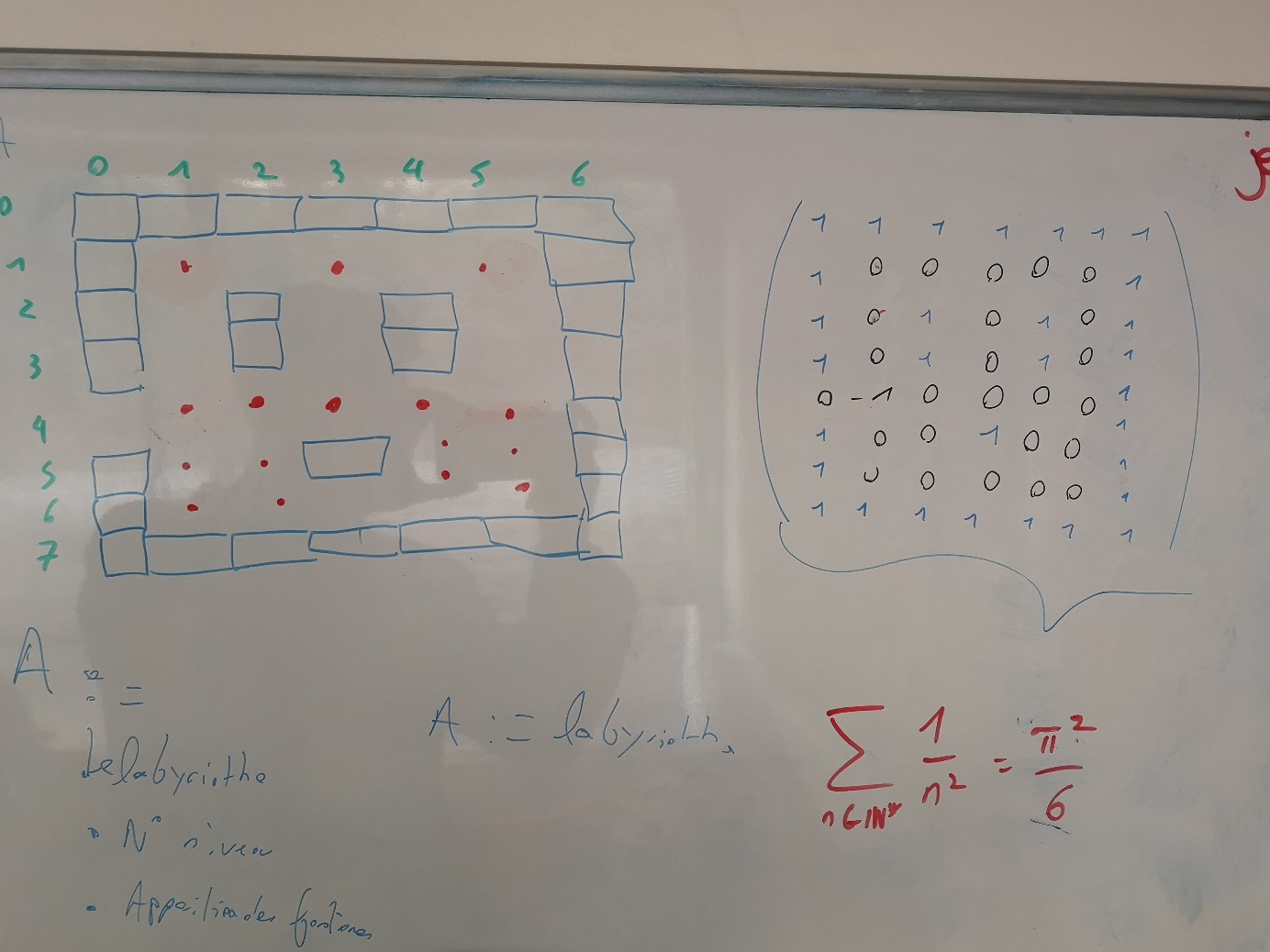
The firsts versions will be playable on a Windows machine. We decided to program this version with Java, with this environment: Eclipse IDE, Atom, Github.

II.1 – Model

Maze : Each maze will be loaded from a specific file. A maze will be represented by a Matrix filled with integers, indicating what kind of box we will have at the (i,j) position. Our coordinate system will be (O, ,  ) with O on the upper-left, x up-down oriented and y left-right oriented.

With this Matrix, the program will load all the collectables in an array. Il will also transform this matrix (conserved) into a non-oriented weighted graph, used to consider the absolute positions of the movable objects (Pac-man, ghosts…) and know if they are on an intersection:

* For Pac-Man: Know if he can change of direction
* For ghosts: Know when the program must randomize their movement. In the future : Decide of the shortest way to join Pac-man



II.2 – View

JavaFX

The walls will be 3 times smallers than the cases.

II.3 – Controller