FRANCOIS Rémy

M2 IVI
SCI – TP3

BILLES

Compilation : javac *.java

Exécution : java –jar Billes.jar

Les billes se déplacent dans une direction, en fonction de si elles rencontrent une bille ou un obstacle, sa direction sera modifiée. La génération est aléatoire de manière à ce que 30% de l'espace soit utilisé.

WATOR

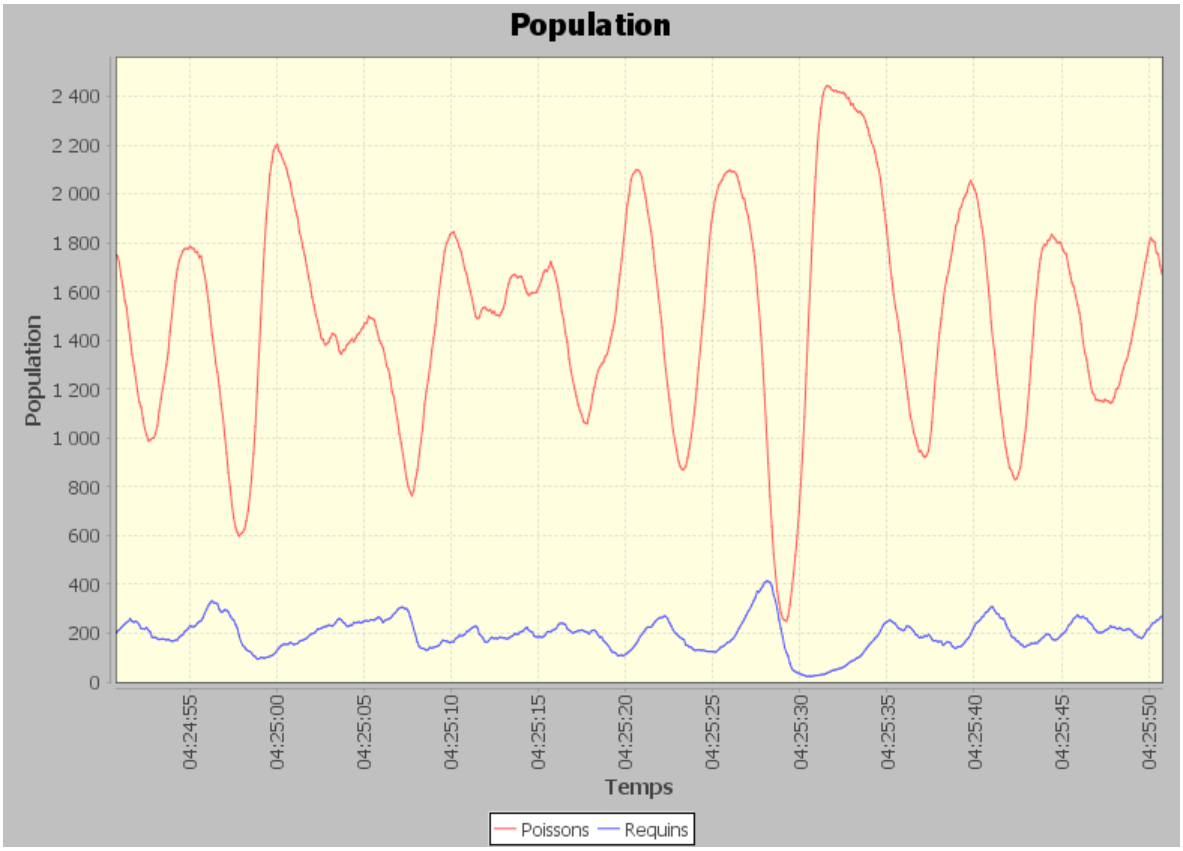
Compilation :

Exécution :

La grille par défaut est de 50*50 mais celle-ci est modifiable via un menu. Si l'on met un environnement suffisamment petit (exemple 20*20), il arrive que l'une ou l'autre des populations disparaisse.

Le contenu de la grille est généré aléatoirement de manière à ce qu'environ 70% des cases soient utilisées et qu'il n'y ait que 5% de requin dans la population totale.

Par contre, il n'arrive jamais que la population de requins ne dépasse celle de poissons, même si nous avons des fluctuations dans les 2 qui sont liées. Plus il y a de requin, moins il y a de poissons.

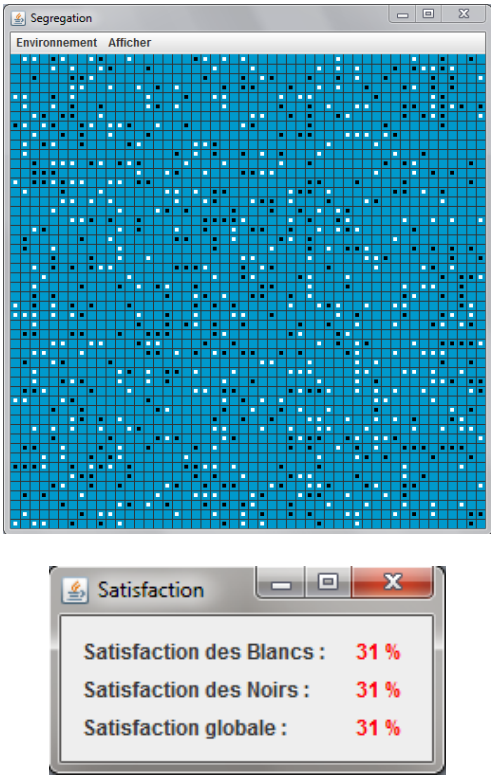


SEGREGATION

Compilation : javac *.java

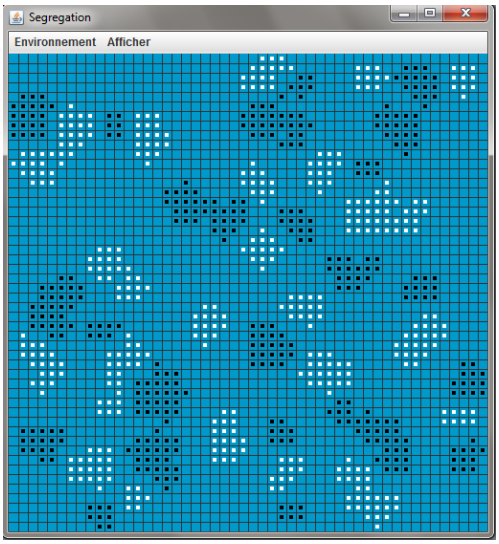
Exécution : java –jar Segregation.jar

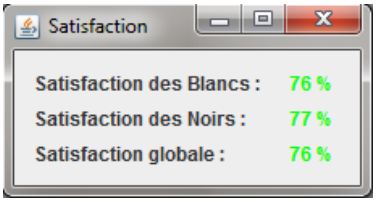
Les populations sont à peu près égales et la génération est telle qu'elle occupe 30% de l'espace. Les agents se déplacent uniquement sur les cases voisines si celles-ci sont atteignables. Ce qui entraine une difficulté à trouver une situation stable. En effet, avec un seuil de satisfaction minimal de 70%, nous arrivons à la situation suivante :



Taux de satisfaction des différentes populations

Par contre, dès que ce seuil passe à 50%, nous arrivons par arriver à une situation stable avec de petites communautés et un taux de satisfaction plus fort.





PACMAN

Compilation : javac *.java

Exécution : java –jar PacMan.jar

Les agents se déplacent en croix, les monstres (en rouge) poursuivent le PacMan (en jaune) qui se déplacent aléatoirement. Il existe cependant un problème, si un monstre est à 1 case du PacMan, alors il aura un petit problème pour déterminer la case suivante. La valeur minimum étant 0 et correspondant au PacMan, le monstre ne peut s'y rendre. Et comme il se déplace en croix, il ne peut pas se rendre sur une case ayant la valeur 1 (les cases diagonales autour de PacMan sont considérés comme étant à 2 de distance, puisqu'il faudrait 2 mouvements pour atteindre la case 0 avec un déplacement en croix), du coup, la valeur minimum accessible et 2, et il se peut alors que le monstre s'éloigne du PacMan. Et comme les déplacements sont effectués en même temps, il arrive qu'une distance se crée entre le PacMan et un monstre.