TD Netlogo 1

Objectif

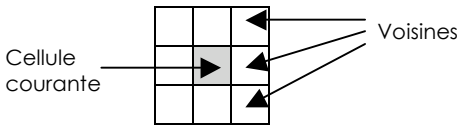
Le but de ce premier TD est de prendre en main l'environnement de la plate-forme Netlogo, et de programmer des premiers modèles simples en faisant bien la distinction entre ce qui concerne l'observer, les patchs et les tortues.

Les automates cellulaires

Les automates cellulaires constituent une classe de systèmes dans laquelle une grille de cellules évolue de cycle en cycle en fonction de règles définies au niveau de chaque cellule et spécifiant quel doit être l'état de la cellule au cycle suivant en fonction de son état courant et de l'état de ses plus proches voisines. A chaque cycle, toutes les cellules recalculent leur nouvel état puis elles changent toutes d'état simultanément, de manière synchrone.
Dans la plate-forme Netlogo, nous considérerons les patchs comme une grille de cellules dont l'état sera recalculé à chaque cycle.

Le jeu de la vie

Le jeu de la vie, proposé par J. Conway dans les années 1970 est l'exemple le plus connu et le plus étudié de ce type de systèmes. Dans cet exemple, chaque cellule ne peut se trouver à chaque instant que dans l'un des 2 états **on** (allumée) ou **off** (éteinte). Lorsqu'elle est allumée, la cellule ne restera allumée au cycle suivant que si 2 ou 3 de ses 8 voisines sont elles-mêmes allumées. Lorsqu'elle est éteinte, la cellule ne s'allume au cycle suivant que si exactement 3 de ses 8 voisines sont allumées.



Travail demandé

- **Setup**
  - Ajouter une variable pour les patchs qui indique si le patch est dans l'état on ou off
  - Ecrire la procédure `setup` qui initialise aléatoirement chaque patch dans l'état on ou off et l'affiche en blanc ou noir suivant les cas
  - Ajouter à l'interface un bouton **Setup** qui exécute la procédure `setup`
- **Go**
  - Ecrire la procédure `go` qui demande aux patchs de :
    - compter le nombre leur voisins (voir `nsum`)
    - calculer leur nouvel état
    - se réafficher en blanc ou noir suivant l'état
  - Ajouter à l'interface un bouton **Go** qui exécute la procédure `go` de manière répétitive
- **Observation**
  - Observer les différents types de comportements de groupes de cellules ;
  - Ecrire une procédure qui permette à l'utilisateur de modifier l'état des cellules grâce à la souris (voir `mouse-down?`, `mouse-xcor`, `mouse-ycor`, préfixe `-at`)
  - Ajouter un bouton pour activer ou désactiver la fonction de dessin
- **Expérimentation**
  - Ajouter des sliders pour pouvoir paramétrer les conditions de changement d'état des cellules ; tester différentes règles ;
  - En s'inspirant du modèle « Wolf Sheep Predation », ajouter un graphique pour tracer le nombre de cellules allumées à chaque cycle ;
  - Expérimenter l'outil « Behavior Space »

Application à un système de vote

Appliquer le modèle précédant à un modèle de vote : on considère que la couleur d'une cellule correspond à un choix de vote. Pour savoir comment voter, une cellule compte le nombre de ses voisines qui ont voté d'une certaine manière. Elle vote alors de la même manière que le majorité des cellules, ou elle maintien son vote précédant en cas d'égalité.