Programmation Orienté Agent : NetLogo

Master 2 MISC
2019 / 2020

L'auteur de ces exercices est Philippe Collard, Université Nice. Un grand merci à lui !

Exercice 1 : agent tortue

Questions :

a - Créer une particule (turtle) positionnée (xcor ycor) au centre de son environnement (0,0), dirigée (heading) au nord-est, de couleur (color) rouge et de forme (shape) "arrow".

```
to setup
  clear-all
  create-turtles 1 ; créer une seule tortue (elle porte le numéro 0)
  ask turtle 0 [
     set xcor 0
     set ycor 0
     set heading 45
     set color red
     set shape "arrow"
   ]
end
```

Pour exécuter ce code, vous devez créer un bouton setup dans l'interface NetLogo.

b - Définir dans une procédure `go` le comportement individuel de cette particule de façon à ce qu'elle parcoure un carré puis revienne à son état initial.

- Une particule peut avancer selon sa direction (heading) d'une certaine longueur :
  `forward number`
- Une particule peut tourner d'un certain angle et ainsi modifier son heading :
  `right number`
- Une particule peut laisser une trace en se déplaçant :
  `pen-down`
  `pen-erase`
  `pen-up`

```
to go
   ask turtle 0 [
      pen-down
      forward ...
```

```
      right ...
   ]
end
```

c - Pour éviter d'exécuter quatre fois la procédure `go`, utiliser une structure de contrôle repeat :

```
to go
   ask turtle 0 [
      pen-down
      repeat 4 [
           forward ...
           right ...
      ]
   ]
end
```

d - Introduire une variable locale par : `let nombreDeCotes 4`
dans la procédure go et modifier le code en conséquence.

e - Remplacer la variable locale par une variable globale :

```
; à placer au début du code
globals [ nombreDeCotes ]
```

et initialiser cette variable dans la procédure setup par :

```
set nombreDeCotes 4
```

Notez bien la différence pour une affectation entre let et set.

f - Ajouter un second Slider afin de contrôler la longueur du polygone régulier tracé par la particule.

g - Définir un Slider dans l'interface pour remplacer la déclaration et l'initialisation de la variable globale nombreDeCotes.

h - Editer le bouton go pour "boucler" forever sur l'exécution de la procédure. Vous pouvez régler la vitesse d'exécution de la simulation en "jouant" sur le curseur qui se trouve en haut et au milieu de l'interface !

Exercice 2 :

Questions :

a - Créer une particule qui se déplace en ligne droite en suivant uniquement les directions diagonales et qui rebondit comme une boule de billard lorsqu'elle touche un des quatre bords. On pourra initialiser son heading à 45° et modifier la forme de l'espace pour obtenir une "table de billard rectangulaire" (bouton setting de l'interface).

```
to setup
  clear-all
  create-turtles 1
  ask turtle 0 [
     setxy 0 0
     set heading 45
     set color red
     set shape "arrow"
     pen-down
   ]
end
```

```
to go
   ask turtles [
        ifelse     (ycor <= min-pycor + 2 and heading = 135) [set heading  ... ]
           [ifelse (ycor <= min-pycor + 2 and heading = 225) [set heading ... ]
           [ifelse (ycor >= max-pycor - 2 and heading =  45) [set heading ... ]
           [ifelse (ycor >= max-pycor - 2 and heading = 315) [set heading ... ]
           [ifelse (xcor <= min-pxcor + 2 and heading = 315) [set heading  ... ]
           [ifelse (xcor <= min-pxcor + 2 and heading = 225) [set heading ... ]
           [ifelse (xcor >= max-pxcor - 2 and heading = 135) [set heading ... ]
           [if     (xcor >= max-pxcor - 2 and heading =  45) [set heading ... ]]]]]]]]
        forward 1
   ]
end
```

Créer trois boutons `go`, `goo` et `gooo` dans l'interface pour lancer l'exécution de la simulation sur un pas de temps, dix pas de temps ou forever. Pour le bouton `goo` vous pouvez définir une nouvelle procédure :

```
to goo
   repeat 10 [
     go
   ]
end
```

b - Créer simultanément deux particules sur le modèle de la boule de billard. On pourra les différencier par leur position initiale et par leur couleur. Utiliser `ask turtles ...` pour spécifier un comportement commun à toutes les particules.

c - Modifier le programme précédent pour que pour la première particule devienne peureuse : elle devra rebrousser chemin dès qu'elle se trouve à proximité de la seconde. On émettra un signal sonore (beep) lors de chaque rencontre. On pourra ajouter un Slider dans l'interface afin de pouvoir contrôler dynamiquement le rayon de proximité.

d - Modifier le programme précédent afin de disposer de nombreuses peureuses ; chacune devra rebrousser chemin dès qu'elle se trouve à proximité de la seule particule "normale". On émettra un signal sonore (beep) lors de chaque rencontre. On pourra ajouter un Slider dans l'interface afin de pouvoir contrôler dynamiquement le `nombreDePeureuses`.

Exercice 3 :

Questions :

a - Créer une particule qui se déplace au hasard dans son espace et qui rebrousse chemin dès qu'elle touche un des quatre bords de son espace. On pourra utiliser les reporter suivants pour savoir si, oui ou non, la particule a atteint un des quatre bords (un reporter est une procédure qui retourne un résultat).

```
to-report toucherBord? report (bordD? or bordG? or bordH? or bordB?) end

to-report bordD? report (max-pxcor - 2 <= xcor) end

to-report bordG? report (xcor <= min-pxcor + 2) end
```

```
to-report bordH? report (max-pycor - 2 <= ycor) end

to-report bordB? report (ycor <= min-pycor + 2) end
```

b - Créer une particule leader qui possède un angle de vision ; un leader avance dans une direction qui dévie aléatoirement par rapport à son heading sans toutefois aller au delà de la zone couverte par son angle de vision. Si un leader touche un des bords de son domaine alors il rebrousse chemin. On pourra ajouter un Slider dans l'interface afin de pouvoir contrôler dynamiquement l'angle de vision.

- Créer trois espèces (breed) de particules : les browniens, les leaders, et les bouleBillards.

```
breed [browniens brownien]
breed [leaders leader]
breed [bouleBillards bouleBillard]
puis créer et initialiser deux individus de chaque espèce :
  create-browniens 2
  create-leaders 2
  create-bouleBillards 2
  ask browniens [
     setxy 0 0
     set heading 0
     set color red
     set shape "arrow"
     pen-down
   ]
   ask leaders [
     setxy 0 0
     set heading 0
     set color blue
     set shape "triangle"
     pen-down
   ]
   ask bouleBillards [
     setxy random-pxcor random-pycor
     set heading 45
     set color green
     set shape "circle"
     pen-down
   ]
```

puis définir les trois reporters, newCapBrownien, newCapLeader, newCapBillard, utilisés dans la procédure go ; chacun, selon l'espèce considérée, retourne le nouveau cap (heading) en fonction du cap précédent passé en argument.

```
to go
   ask browniens [
        set heading newCapBrownien(heading)
        forward 1
   ]
```

```
   ask leaders [
        set heading newCapLeader(heading)
        forward 1
   ]
   ask bouleBillards [
        set heading newCapBillard(heading)
        forward 1
   ]
end

to-report newCapBrownien [cap]
   ifelse (toucherBord?) [report cap + 180 ] [report cap + ???]
end

to-report newCapLeader [cap]
   ifelse (toucherBord?) [report cap + 180] [report cap + ???]
end

to-report newCapBillard [cap]
    ifelse     (bordB? and heading = 135) [report  45 ]
       [ifelse (bordB? and heading = 225) [report 315 ]
       [ifelse (bordH? and heading =  45) [report 135 ]
       [ifelse (bordH? and heading = 315) [report 225 ]
       [ifelse (bordG? and heading = 315) [report  45 ]
       [ifelse (bordG? and heading = 225) [report 135 ]
       [ifelse (bordD? and heading = 135) [report 225 ]
       [ifelse (bordD? and heading =  45) [report 315 ]
                 [report cap]]]]]]]]
end
```

Exercice 4 :

Questions :

a - Créer une particule leader qui se déplace en suivant la souris à partir d'un point de départ fixé jusqu'à un point d'arrivée fixé. Avec un setting de l'interface de max-pxcor = 150, max-pycor = 150 et patch size = 2, on pourra avoir l'initialisation suivante :

```
breed [ leaders leader ]

globals [
  depart-x      depart-y        ; patch de depart
  destination-x destination-y   ; patch d'arrivee
]

to setup
  ca ; clear-all
  set depart-x 20 + min-pxcor
```

```
  set depart-y -20
  set destination-x max-pxcor - 20
  set destination-y -20

  ask patch destination-x destination-y [set pcolor violet]

  create-leaders (1) [
      setxy depart-x depart-y
      set size 2
      set color blue
      set pen-size 2 pd
  ]
end
```

Analyser et exécuter le code de la procédure go :

```
to go
   if tousArrive? [stop]
   ask leaders [
      if (not arrive?) [ while [not mouse-down?] []]
      ifelse (xcor > (destination-x - 5 ))
         [ setxy destination-x destination-y]
         [ setxy mouse-xcor mouse-ycor ]
   ]
  tick
end

to-report arrive?
    report (distancexy destination-x destination-y <= 1 )
end

to-report tousArrive? ; tous les particules sont arrivées à destination
    report (all? turtles [arrive?])
end
```

b - Créer une nouvelle espèce de particules suiveuses. La première d'entre elles suivra le leader, la deuxième suivra la première, etc.

— Une suiveuse aura donc comme caractéristique propre la particule qui se trouve devant elle : `suiveuses-own [devant]`.

— Lors d'une création, un numéro (accessible via la variable who) est associé à chaque turtle ; la première turtle créée aura 0 comme who, la seconde 1, etc.

Ajouter le code suivant dans la procédure setup après l'instruction create-leaders

```
create-suiveuses (nombreSuiveuses) [
      setxy depart-x depart-y
      set devant (turtle (who - 1))
      set color red
      set size 2
      set heading random 180
  ]
```

Pour une suiveuse, suivre une particule consiste simplement à regarder dans sa direction (face ...) et à avancer tout droit d'un pas. Coder dans la procédure go le comportement individuel d'une suiveuse :

```
ask suiveuses [ if parti? and (not arrive?) [...] ]
```

On utilisera le reporter suivant :

```
ask suiveuses [ if parti? and (not arrive?) [...] ]
to-report parti? report (([xcor] of devant) > (depart-x + tempsAttente)) end
```

Expérimenter le système complexe leader+suiveuses et caractériser le comportement global émergent.

c - Ajouter le code suivant dans la procédure setup afin que la dernière suiveuse matérialise sa trajectoire.

```
ask turtle (...) [ ; la dernière particule
     set color green  set pen-size 2 pen-down
 ]
```

d - On suppose maintenant que pour mettre à jour sa direction une suiveuse réalise un compromis entre sa direction courant (heading) et la direction dans laquelle elle "voit" la particule qui la précède. Ce compromis est fonction de son inertie, c'est à dire du poids relatif de sa direction courante. Modifier en conséquence le code de la procédure go et ajouter un Slider dans l'interface avec des valeurs de l'inertie comprises entre 0 et 1.

```
ask suiveuses [
       let oldHeading heading
       if parti? and (not arrive?)
            [ face devant ;
              set heading  inertie * oldHeading +  (1 - inertie) * heading
              fd 1
            ]
   ]
```

Expérimenter ce nouveau comportement individuel selon les valeurs du coefficient d'inertie. Vous déterminerez s'il existe pour ce paramètre une valeur critique qui correspond à un changement de dynamique à même de modifier la nature du phénomène émergent.