Introduction à NetLogo

Valerian.Guivarch@irit.fr

Présentation de NetLogo

- NetLogo est un environnement de programmation pour la modélisation/ simulation de phénomènes collectifs naturels

- Bien adapté à la modélisation de systèmes complexes composés de centaines, de milliers d'agents agissant en parallèle

- Possibilité de réaliser de nombreuses simulations couvrant de nombreux domaines d'applications

- Possibilité de créer ses propres modèles

Exemple : simulation Proie - Prédateur

- Dans une prairie se trouvent des moutons et des loups :
  - Les moutons et les loups se déplacent
    - L'énergie des agents est décrémentée
  - Si un mouton rencontre de l'herbe, il la mange
    - L'énergie du mouton est augmentée
  - Si un loup rencontre un mouton, il le mange
    - L'énergie du loup est augmentée
  - De temps en temps, un animal se reproduit
    - L'énergie de l'animal est divisée par deux entre lui et son clone

- Stabilisation des populations?

Exemple : simulation Proie - Prédateur

- Variables paramétrables
  - L'herbe
    - Consommable?
    - Durée
  - Nombre initial de moutons et de loups
  - Gain d'énergie
    - Energie gagnée par le mouton en mangeant de l'herbe
    - Energie gagnée par le loup en mangeant un mouton
  - Probabilité de se reproduire pour les moutons et les loups

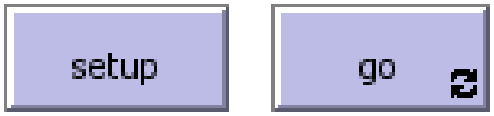
- Un total de 8 paramètres pour cette simulation

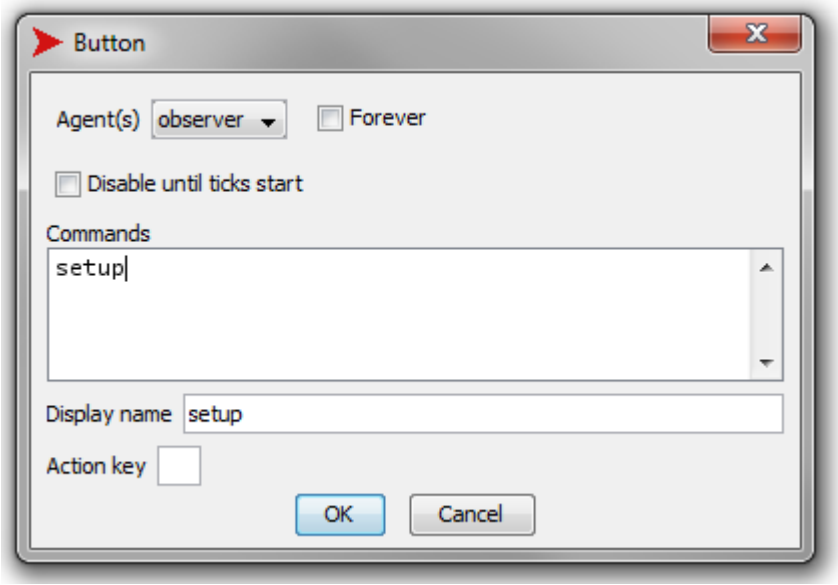
Interface


Interface
Info
Code
Edit
Delete
Add
Button
Button
Slider
Switch
Chooser
Input
Monitor
Plot
Output
Note
normal speed
view updates
continuous
Settings...
ticks:
3D

Interface : contrôle

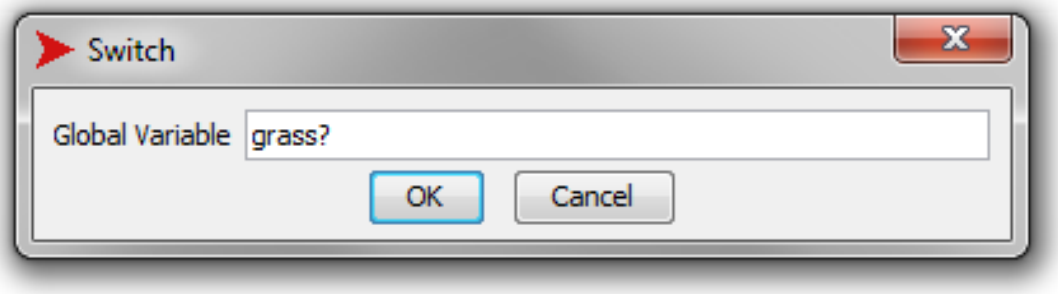
- Bouton
  - Associé à une commande
  - Appel
    - Unique (once)
    - Répété (forever)





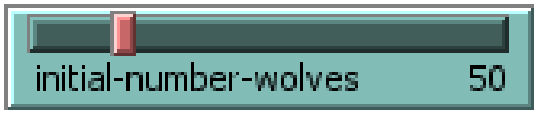
Interface : paramétrage

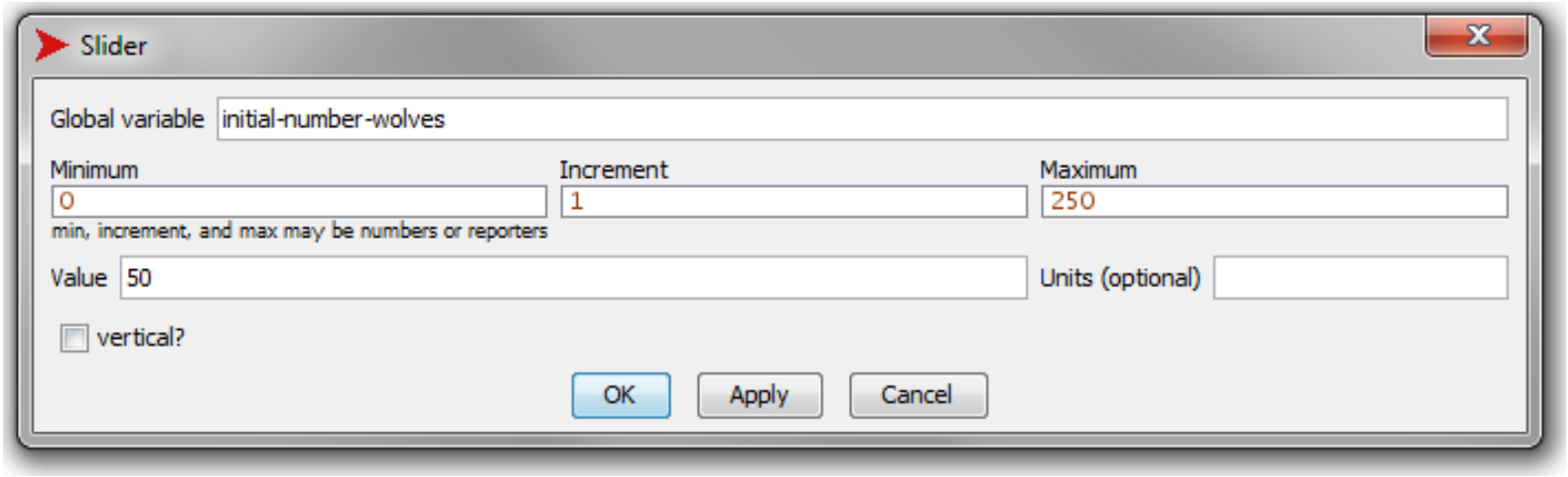
- Switch
  - Défini une valeur booléenne



Interface : paramétrage

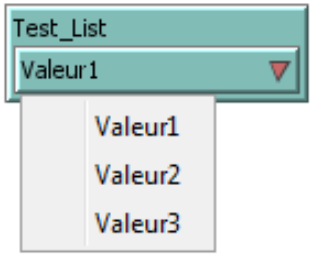
- Slider
  - Ajuster une valeur entre *min* et *max* au travers d'un *increment*

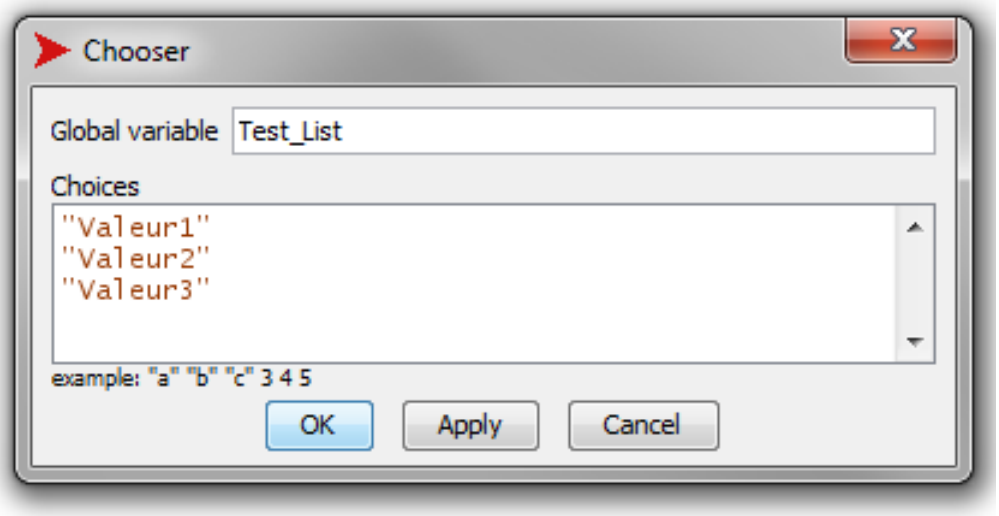




Interface : paramétrage

- Chooser
  - Sélectionner une valeur dans une liste





Agents

- **Tortues** : agents qui se déplacent dans le monde virtuel <=> *Agents*

- **Patch** : une portion de sol sur laquelle les tortues peuvent se situer et se déplacer. L'ensemble des patchs forme le monde, divisé selon une grille circulaire de patchs <=> *Environnement*

- **Observateur** : regarde de l'extérieur le monde des tortues et des patchs (n'est pas situé dans le monde)

Initialisation du monde (moutons)

- Définition d'un type d'agents

  breed [*nom_liste_agents nom_espece*]

  Exemples

  breed [sheep a-sheep]
  breed [wolves wolf]

- Créations d'agents

  create-*nom_liste_agents nb_agents*
  [
  ;;instructions exécutées à la création des agents
  ]

Initialisation du monde (moutons)

- Ajout attribut à un agent

  nom_agent-**own** [nom_attribut]

  - sheep-own [energy] ; ajout de l'attribut energy aux moutons
  - turtles-own [energy] ; ajout de l'attribut energy à tous les agents

- Déclaration d'une procédure

  to *nom_procedure*

  ;;*instructions de la procédure*

  end

- Modification de l'état d'une variable

  set *nom_var nouvelle_valeur*

Initialisation du monde (moutons)

```
breed [sheep a-sheep]
sheep-own[energy]

to setup
;;initialisation du monde
clear-all
set-default-shape sheep "sheep"
create-sheep initial-number-sheep
[
    ;;initialisation des moutons
    set color white
    set size 1.5
    set energy random (2 * sheep-gain-from-food)
    setxy random-xcor random-ycor
]
end
```

Initialisation du monde (loups et moutons)

```
breed [sheep a-sheep]
breed [wolf a-wolf]
turtles-own[energy]

to setup
;;initialisation du monde
clear-all
set-default-shape sheep "sheep"
create-sheep  initial-number-sheep
[
    ;;initialisation des moutons
    set color white
    set size 1.5
    set energy random (2 * sheep-gain-
    from-food)
    setxy random-xcor random-ycor
]
set-default-shape wolves "wolf"
 create-wolves initial-number-wolves
 [
    ;;initialisation des moutons
    set color black
    set size 2
    set energy random (2 * wolf-gain-
    from-food)
    setxy random-xcor random-ycor
 ]
end
```

Initialisation du monde (patches)

- Parcours d'un ensemble d'agents

```
ask nom_liste_agents
[
  ;;traitements sur les agents de nom_liste_agents
]
```

- Condition

```
if condition
  [ ;;instructions exécutées si condition respectée]

if-else condition
  [ ;;instructions exécutées si condition respectée]
  [ ;;instructions exécutées si condition non respectée]
```

Exercice

- Ajouter l'initialisation des patches, sachant que
  - Les patches sont représentés par l'ensemble *patches*
  - Caractéristiques de l'herbe
    - Soit vert (mangeable), soit marron (non mangeable)
    - Peut être « consommable »
      - Passe de mangeable à non mangeable quand mangé par un mouton
      - Repasse à mangeable après un certain nombre de cycles
  - Variables de l'interface
    - La variable *grass?* Détermine si l'herbe est ``consommable''
      - Si non, toute l'herbe reste toujours mangeable
      - Si oui, seule la moitié de l'herbe est mangeable au début
        - (une valeur au hasard dans un ensemble : **one-of** [*valeur1 valeur2 valeur3*])
    - De l'herbe consommable est soit verte (mangeable par un mouton), soit marron (non-mangeable)
      - (la couleur d'un patch est représenté par l'attribut *pcolor*)

Initialisation du monde (patches)

```
patches-own [countdown]

to setup
ask patches [ set pcolor green ]
if grass? [
    ask patches [
        set pcolor one-of [green brown]
        if-else pcolor = green
                [ set countdown grass-regrowth-time ]
                [ set countdown random grass-regrowth-time ]
    ]
]
end
```

Évolution du monde (procédure go)

```
to go
if not any? sheep
    [stop]
ask sheep [
    move
    death
    reproduce-sheep
    ]
tick
end

to move
rt random 50
lt random 50
fd 1
end
```

```
to death
if energy < 0
    [ die ]
end

to reproduce-sheep
if random-float 100 < sheep-reproduce [
    set energy (energy / 2)
    hatch 1 [
        rt random-float 360
        fd 1
        ]
]
end
```

Évolution du monde (procédure go)

```
to go
if not any? Turtles
    [stop]
ask sheep [
    if grass? [
    set energy energy -1
    eat-grass
    ]
    move
    death
    reproduce-sheep
    ]
;;traitement des loups
if grass? [ ask patches [ grow-grass ] ]
 tick
end
```

```
to eat-grass
if pcolor = green [
    set pcolor brown
    set energy energy + sheep-gain-
    from-food
]
end

to grow-grass
 if pcolor = brown [
   ifelse countdown <= 0
    [ set pcolor green
      set countdown grass-regrowth-
    time ]
    [ set countdown countdown - 1 ]
 ]
end
```

Exercice

- Ajouter le traitement pour un cycle des loups
  - Traitement similaire aux moutons, sauf que
    - La nécessité de manger n'est pas optionnelle
    - Ils se nourrissent de moutons situés au même endroit qu'eux
    - L'énergie gagné en mangeant un mouton est un paramètre fixé par l'utilisateur depuis l'interface

- Commandes requises

  let var_locale valeur ;; déclaration variable locale

  agents-here ;; ensemble des agents sur le même patch

  one-of retourne *nobody* si appliqué sur un ensemble vide

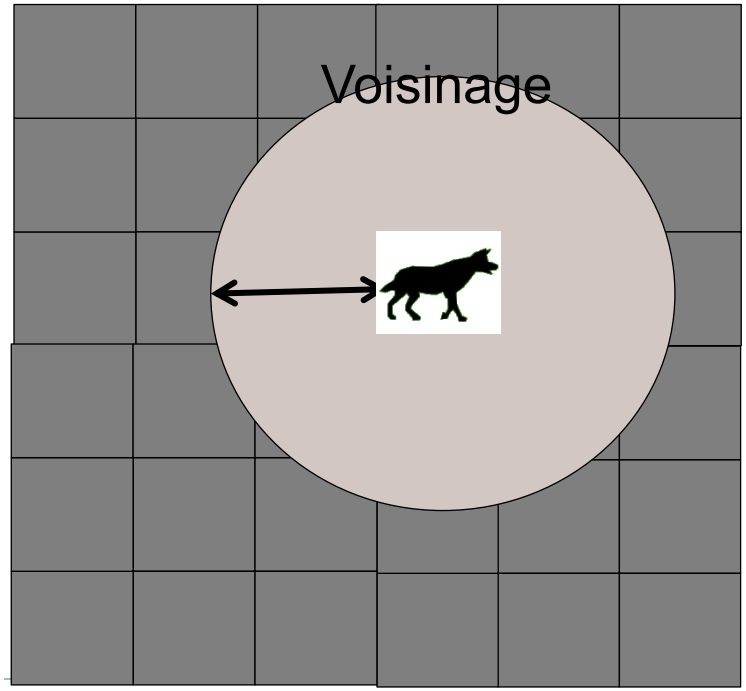
Évolution du monde (procédure go)

```
to go
if not any? Turtles
    [stop]
;;traitement des moutons
ask wolves [
    move
    set energy energy - 1
    catch-sheep
    death
    reproduce-wolves
    ]
if grass? [ ask patches [ grow-grass ] ]
 tick
end
```

```
to catch-sheep
let prey one-of sheep-here
if prey != nobody [
    ask prey [ die ]
    set energy energy + wolf-gain-from-food
    ]
end
to reproduce-wolves
if random-float 100 < wolves-reproduce [
    set energy (energy / 2)
    hatch 1
        [rt random-float 360
        fd 1]
    ]
end
```

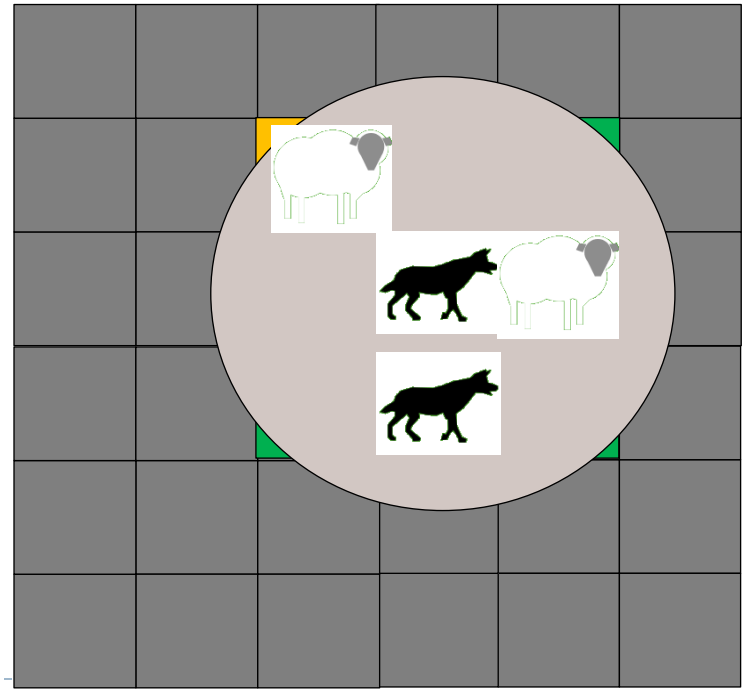
Voisinage

- Exemples d'application de la notion de voisinage
  - Ne permettre à un agent (turtle) d'aller à une position (patch) que si celle-ci n'est pas déjà prise
  - Observer l'état des agents (turtle ou patch) à proximité



Voisinage

- Exemple d'application de la notion de voisinage
  - Ne permettre à un agent (turtle) d'aller à une position (patch) que si celle-ci n'est pas déjà prise
  - Observer l'état des agents (turtle ou patch) à proximité



Tester accessibilité d'une position

- Agents à proximité

  agents-here ;; ensemble des agents sur le même patch

  agents ***in-radius*** *distance*

  Exemple : *patches in-radius 1*

- Sous-ensemble avec condition

  ensemble ***with*** *condition*

  Exemple :

  ```
  to move
  let patches-possibles (patches in-radius 5)
  if (any? patches-possibles) [
    move-to one-of patches-possibles
  ]
  end
  ```

- Patches voisins : *neighbors ;; les 8 patches entourant un agent*

Tester accessibilité d'une position

- Agents à proximité

  agents-here **;;** ensemble des agents sur le même patch

  agents ***in-radius*** *distance*

  Exemple : *patches in-radius 1*

- Sous-ensemble avec condition

  ensemble ***with*** *condition*

  Exemple :

```
to move
let patches-possibles (patches in-radius 5) with [not any? turtles-here]
if (any? patches-possibles) [
  move-to one-of patches-possibles
]
end
```

Exercice

- Un loup peut manger un mouton à côté de lui

```
to catch-sheep
let prey one-of sheep-here
if prey != nobody [
    ask prey [ die ]
    set energy energy + wolf-gain-from-food
    ]
end
```

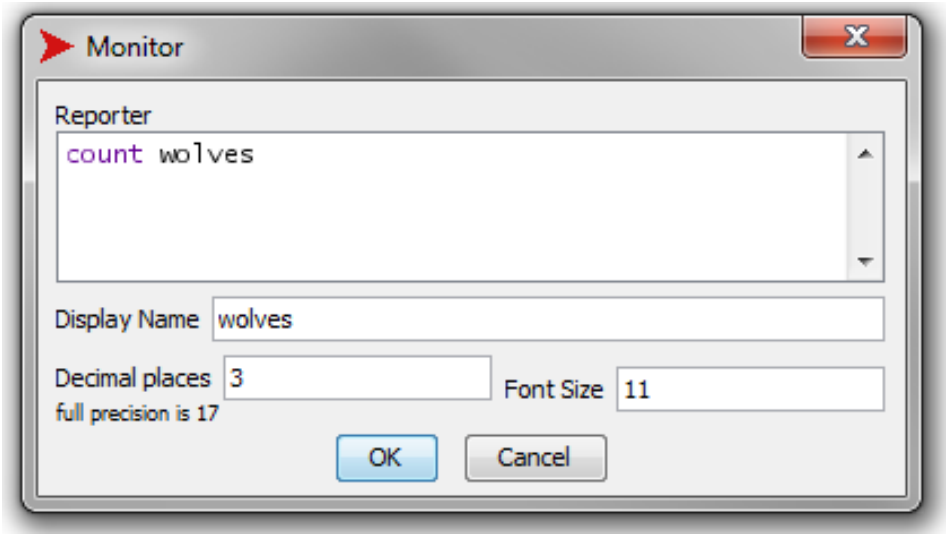
Exercice

- Un loup peut manger un mouton à côté de lui

```
to catch-sheep
let prey one-of sheepes in-radius 1
if prey != nobody [
    ask prey [ die ]
    set energy energy + wolf-gain-from-food
    ]
end
```

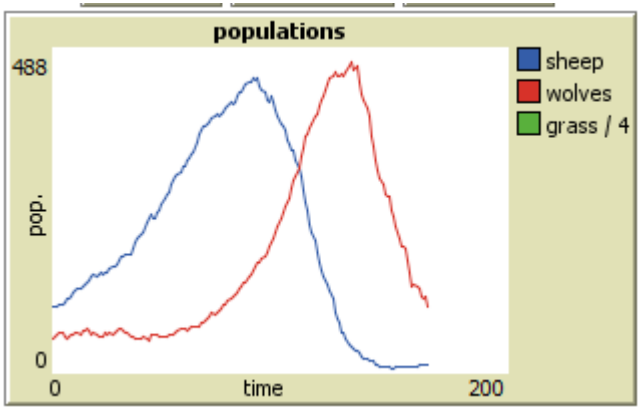
Interface : vue

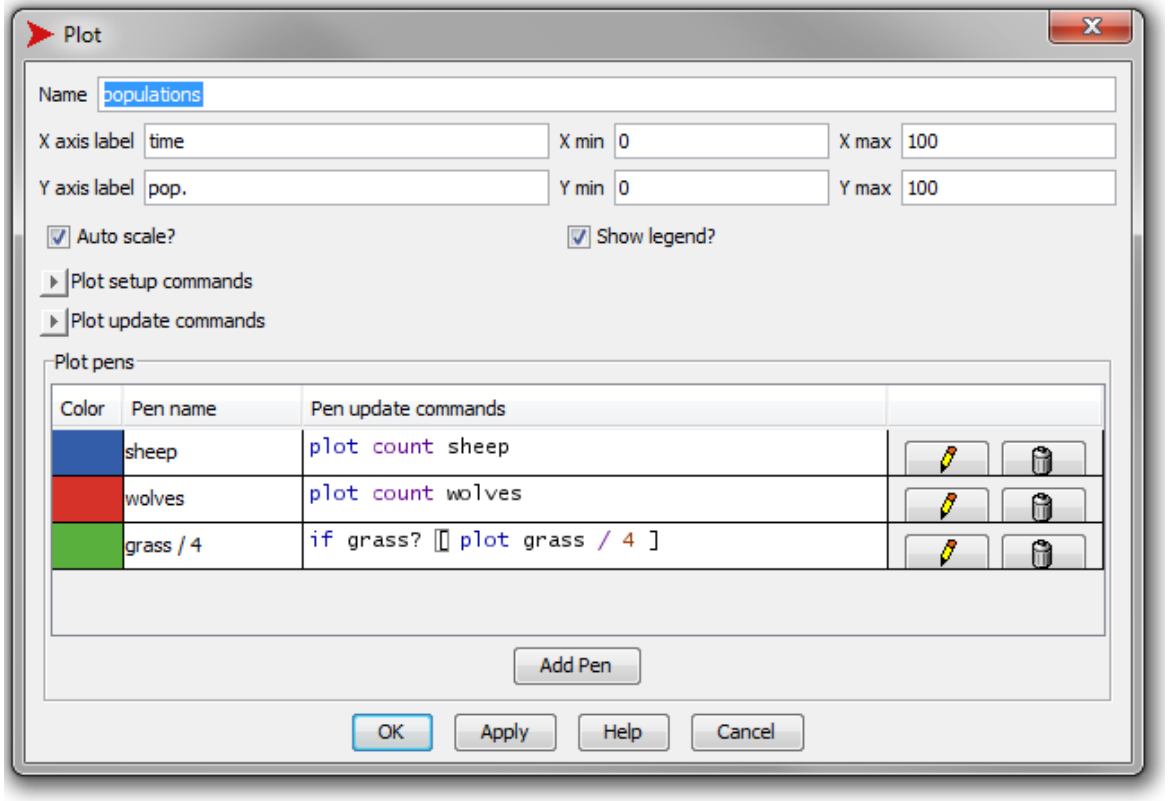
- Monitor
  - Affichage d'une valeur
    - Etat d'une variable
    - Valeur calculée



Interface : vue

- Plot
  - Affichage de valeurs au cours de la simulation





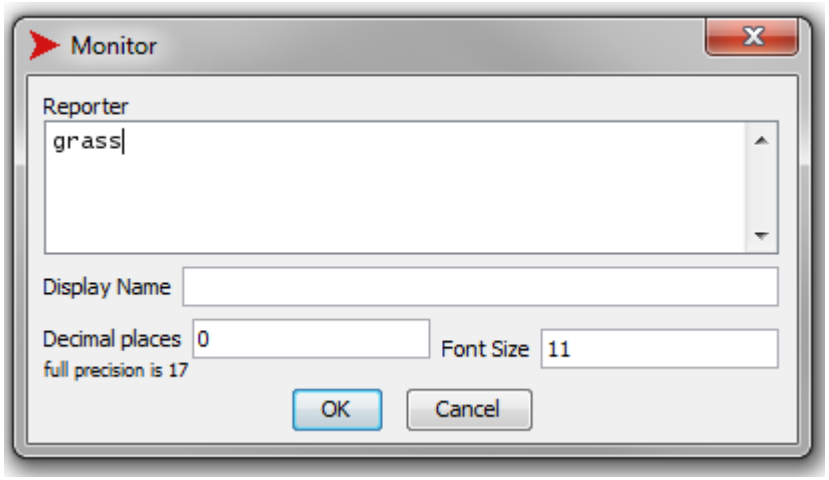
Variable globale

- Déclaration d'une variable globale

  globals [nom_variable]

- Exemple

```
globals [grass ]
to go
 set grass count patches with [pcolor = green]
tick
end
```



Commandes supplémentaires

- Récupérer plusieurs agents d'un ensemble

  n-of *nb_agents liste_agents*

- Taille d'un ensemble d'agents

  count *liste_agents*

- Tester une condition sur toute une liste d'agents :

  all? *liste_agents* **[***condition***]**

- Récupération de l'agent minimisant (maximisant) une certaine expressions dans une liste d'agents :

  min-one-of *liste-agents* **[***expression***]**

  max-one-of *liste-agents* **[***expression***]**

Exemple : let mouton-proche min-one-of (sheep) [distance myself]

Commandes supplémentaires

- Se déplacer d'un certain nombre de pas (un à la fois)
  - **forward** *nb_pas*
  - (équivalent : **fd** *nb_pas)*
- Se déplacer d'un coup d'un certain nombre de pas
  - **jump** *nb_pas*
- Se tourner vers un agent/une position
  - **facexy** *agent* **facexy** *pos_x pos_y*
  - **towards** *agent* **towards** *pos_x pos_y*
- Autres commandes
  ccl.northwestern.edu/netlogo/docs/dictionary.html

Exercices

- La survie des espèces (considérer une vue locale)
  - **Loups**
    - **V1 : Se dirige vers le mouton le plus proche**
    - **V2 : Se dirige vers le mouton le plus proche/le plus faible**
    - **V3 : Se dirige vers le mouton le plus proche/le plus faible et qui soit loin des autres loups**
  - **Moutons**
    - **Fuir les loups**
    - **S'arrêter pour manger quand c'est nécessaire**
      - **Plus le loup est loin**
      - **Plus l'énergie est faible**