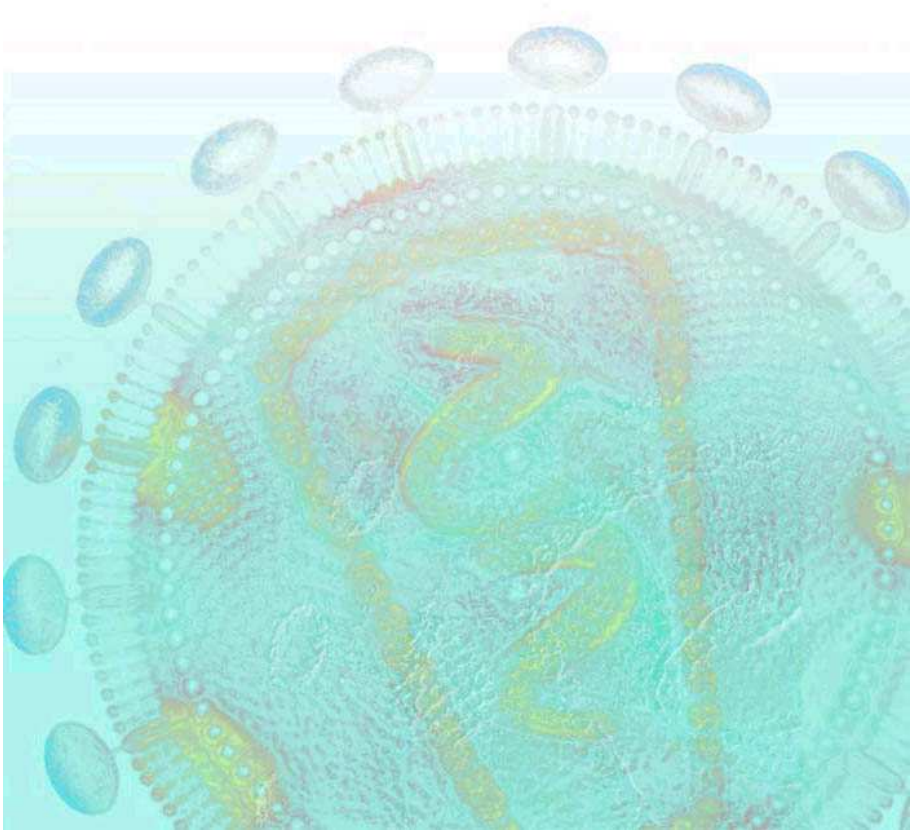
PLANQUART Stéphane
KUNZ Elodie

Licence Informatique
Groupe 1

Université du Havre
Faculté des Sciences et Techniques

Simulations individus-centrées de phénomènes d'épidémiologie avec NetLogo



Encadrants : C. Bertelle, N. Lhuissier et D. Olivier

2003/2004

Ce projet consiste à utiliser la bibliothèque logicielle « NetLogo » de l'université américaine Northwestern pour développer des simulations individus-centrées de phénomènes épidémiologiques. Les simulations sont élaborées à partir de modèles théoriques relatifs à l'épidémiologie et à la diffusion des maladies infectieuses. Que ce soit en matière de dynamique de population ou en matière de modèles de type individus-centrés, les modélisations de phénomènes épidémiologiques font aujourd'hui l'objet de nombreuses recherches mises en œuvre par des organismes tels que le CIRAD ou le CEMAGREF. Le logiciel « NetLogo », qui est un logiciel de modélisation de phénomènes naturels et sociaux, va nous permettre d'écrire une simultation orientée agent concernant l'évolution d'une population infectée par un virus.

Mots-clés : simulation, système multi-agent, épidémiologie, virus, individu-centré, NetLogo

This project consists in using the « NetLogo » software elaborated by the Northwestern university (USA) in order to develop modelling multi-agent systems concerning epidemiologic phenomena. Simulations are elaborated from theoric models of epidemiology and propagation of infectious diseases. Nowadays numerous organizations such as CIRAD or CEMAGREF make researches about epidemiologic phenomena modellings, whether it concerns dynamics of population or multi-agent systems. The « NetLogo » software, a programmable modeling environment for simulating natural and social phenomena, is used in order to study the propagation of a virus in a population composed of autonomous characters.

Keywords : simulation, modelling multi-agent system, epidemiology, virus, NetLogo

TABLE DES MATIERES

De nos jours, des informaticiens mettent leurs compétences au service de la biologie afin de modéliser des phénomènes naturels et sociaux tels que la propagation d'un feu, la prédation ou l'évolution d'une espèce animale dans un environnement donné. Une approche possible en matière de modélisation sont les simulations individus-centrées, également appelées simulations multi-agents. Ces simulations consistent en un environnement dans lequel des agents, définis par leur comportement et un certain nombre de caractéristiques, interagissent entre eux. Ainsi une même simulation peut mettre en scène plusieurs espèces différentes. Les recherches dans ce domaine sont menées par des organismes tels que le CIRAD et le CEMAGREF en France ou les universités de Northwestern et de Santa-Fe aux Etats-Unis. Dans le cadre de notre projet, nous nous sommes intéressés aux phénomènes épidémiologiques afin d'élaborer une simulation de propagation d'un virus au sein d'une population. Cette simulation s'appuie sur des modèles théoriques élaborés par Mélanie Toutain, une étudiante en maîtrise de biologie. Après avoir fait un bref état des lieux en ce qui concerne l'existant en matière de simulation s'apparentant à notre sujet, nous avons présenté la bibliothèque logicielle NetLogo dont nous nous sommes servis pour l'élaboration de notre simulation.

I- Les modèles analytiques de type infections :

L'analyse des différents types de modèles a été réalisée d'après l'ouvrage « *Essential Mathematical Biology* » de NF. Britton.

Selon la théorie, il existe deux grands types de modèles dans la modélisation de maladies infectieuses.

En effet, selon le mode de propagation, on définit une maladie comme ***épidémique*** ou ***endémique***. A savoir qu'une maladie endémique est une maladie qui persiste dans une région en se manifestant en permanence ou périodiquement et qu'une maladie épidémique apparaît subitement et se propage rapidement au sein de la population par la contagion d'un grand nombre de personnes dans une région donnée.

D’autre part, dans chacun de ces cas, on peut avoir deux types de modèles :

- **Le modèle dit « SI »**

Soit un modèle qui décrit la propagation d’une maladie entre deux populations : la population infectée et la population susceptible d’être infectée.

S ⇨ I

- **Le modèle dit « SIR »**

Soit un modèle décrivant la propagation d’une maladie entre trois populations : la population infectée, la population susceptible d’être infectée et enfin la population immunisée contre la maladie.

S ⇨ I ⇨ R

Notre but étant de modéliser un phénomène épidémiologique, on portera notre étude sur la modélisation de type SI et SIR dans le cas de maladies épidémiques, soit à infection subite et propagation par contagion.

1.1- Le modèle SI :

S ⇨ I

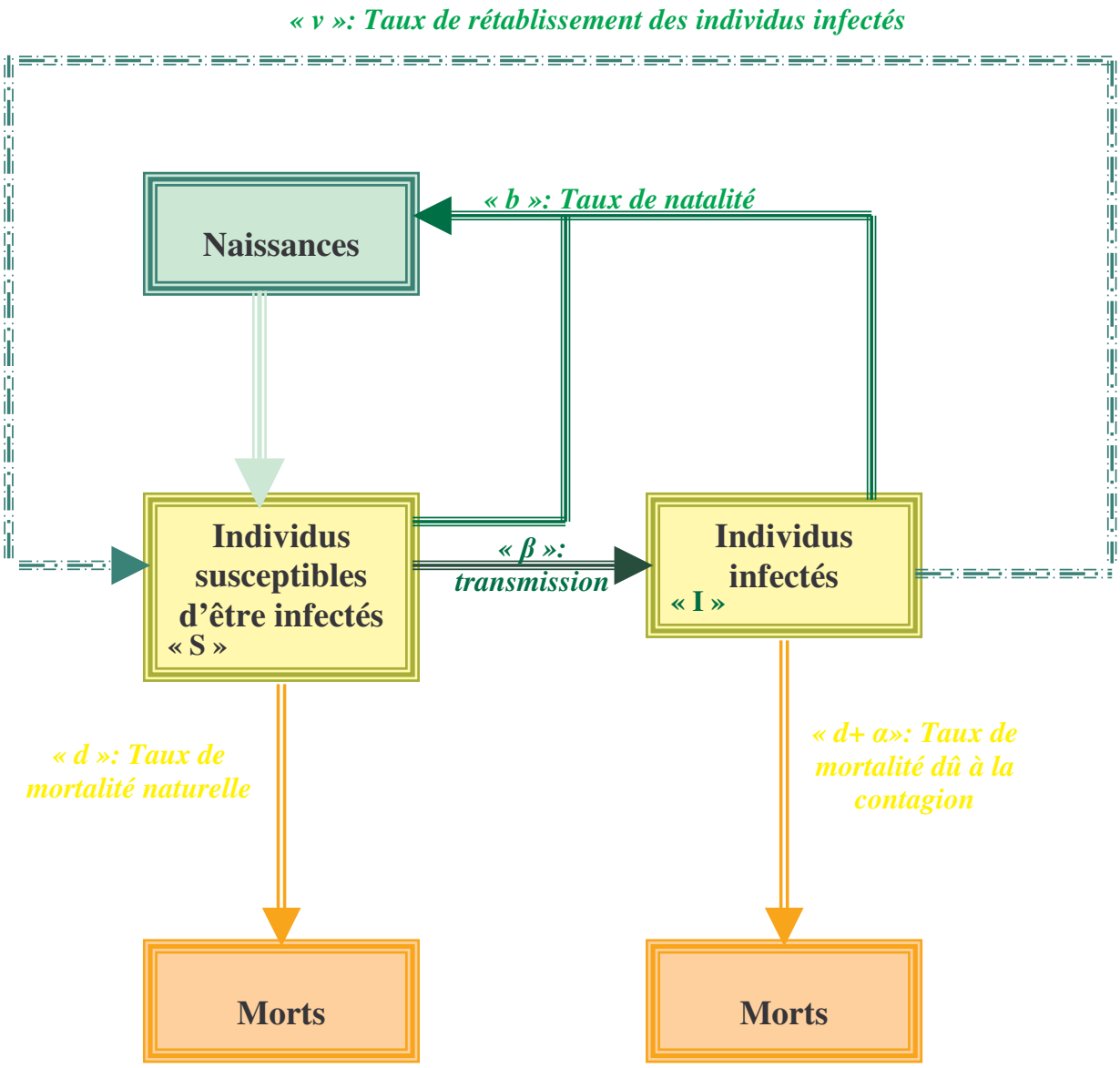
La population « **S** » regroupe les individus sains, donc susceptibles d’être contaminés, et la population « **I »** regroupe les individus infectés.

Les individus de la population S intégrent la population I lorsqu’ils sont contaminés par la maladie. Les individus de la population I réintégrent la population S lorsqu’ils se

rétablissent de la maladie. Ces derniers sont alors à nouveau susceptibles d'être contaminés par la maladie.

La population S additionnée à la population I représente la population totale N.

On peut représenter le phénomène par le schéma suivant :



Sachant l'hypothèse selon laquelle tous les nouveaux nés sont sains, on peut établir les équations suivantes concernant l'évolution des différentes populations impliquées dans ce système.

- Le taux d'accroissement de la population « S » est défini par l'équation suivante :

$$dS/dt = b\,(S+I) + vI - dS - \beta\,S\,I$$

Avec:

En entrée dans la population S :

+ b(S+I) = Natalité des deux populations puisque toutes les naissances donnent des individus sains.

+ vI = Nombre d'individus infectés se rétablissant et devenant des individus sains.

En sortie de la population S :

– dS = Nombre d'individus disparaissant de la population S par mort naturelle.

– β S I = Nombre d'individus étant contaminés par la maladie, donc devenant infectés.

- Le taux d'accroissement de la population « I » est défini par l'équation suivante :

$$dI/dt = \beta\,S\,I - (\alpha + d + v)\,I$$

Avec:

En entrée dans la population I :

+ β S I = Nombre d'individus étant contaminés par la maladie, donc devenant infectés.

En sortie de la population I :

– (α + d + v) I = Nombre d'individus disparaissant de la population I par mort naturelle, par mortalité due à l'infection ou par rétablissement (redevenant donc des individus de la population S).

1.2- Le modèle SIR :

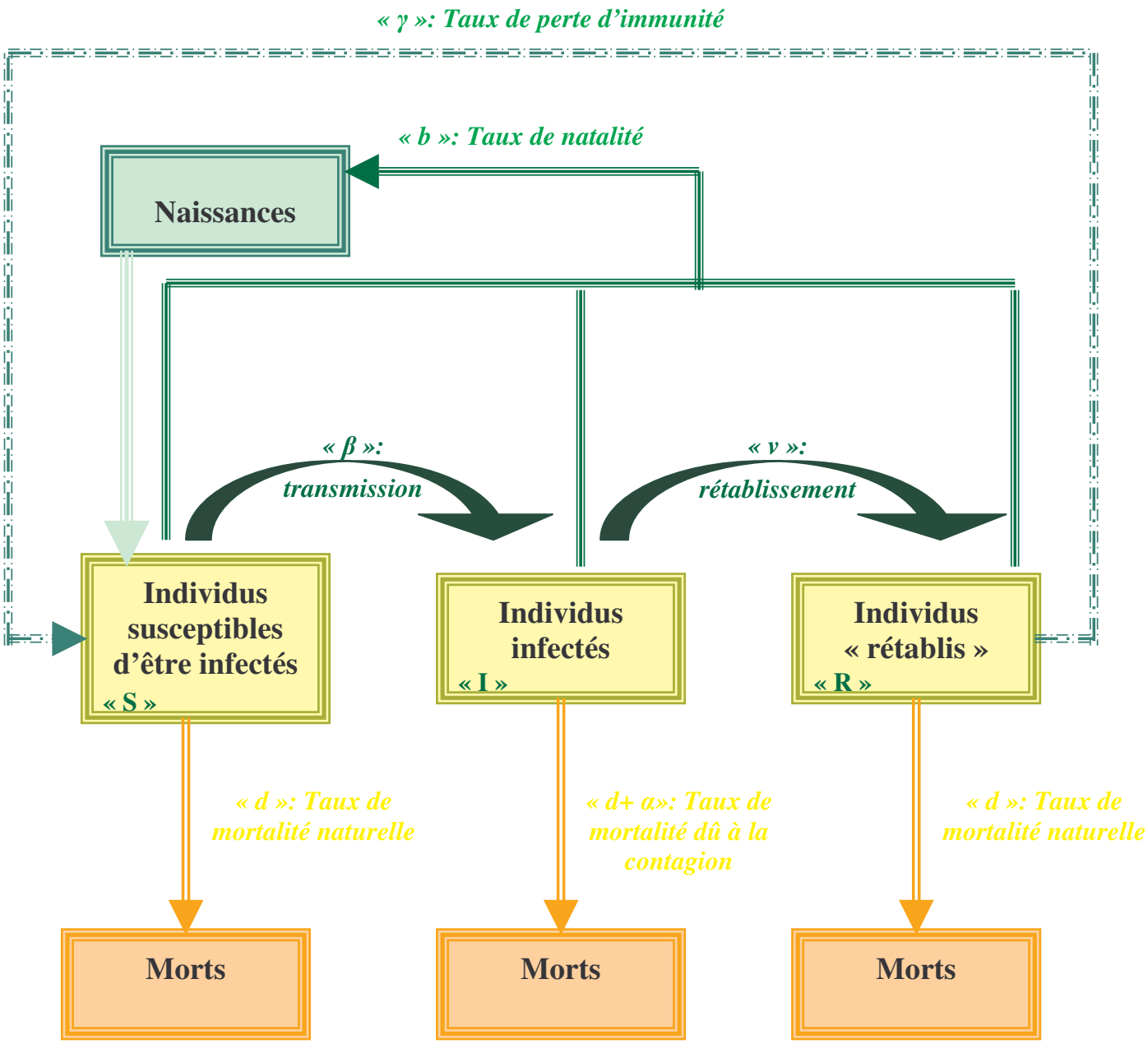
S ⇨ I ⇨ R

La population « **S** » regroupe les individus susceptibles d'être contaminés, la population « **I »** les individus infectés, et la population « **R** » les individus immunisés pour cette infection.

Les individus de la population S intégrent la population I lorsqu'ils sont contaminés par la maladie. Les individus de la population I intégrent la population R lorsqu'ils se rétablissent. Ils deviennent alors immunisés. Ces derniers peuvent toutefois réintégrer la population S en perdant leur immunité. Ils sont alors à nouveau susceptibles d'être contaminés par la maladie.

La population S additionnée à la population I et à la population R représente la population totale N.

On peut représenter le phénomène par le schéma suivant :



Sachant l'hypothèse selon laquelle tous les nouveaux nés sont sains, on peut établir les équations suivantes concernant l'évolution des différentes populations impliquées dans ce système.

- Le taux d'accroissement de la population « S » est défini par l'équation suivante :

$$dS/dt = b(S+I+R) + \gamma R - dS - \beta S I$$

Avec:

En entrée dans la population S :

+ b(S+I+R) = Natalité des trois populations puisque toutes les naissances donnent des individus sains.

+ γR = Nombre d'individus immunisés perdant cette immunité et devenant des individus sains susceptibles d'être à nouveau infectés.

En sortie de la population S :

– dS = Nombre d'individus disparaissant de la population S par mort naturelle.

– β S I = Nombre d'individus étant contaminés par la maladie, donc devenant infectés.

- Le taux d'accroissement de la population « I » est défini par l'équation suivante :

$$dI/dt = \beta S I - (\alpha + d + v) I$$

Avec:

En entrée dans la population I :

+ β S I = Nombre d'individus étant contaminés par la maladie, donc devenant infectés.

En sortie de la population I :

– (α + d + v) I = Nombre d'individus de la population I disparaissant par mort naturelle, par mortalité due à l'infection ou par rétablissement (devenant donc des individus de la population I par gain d'immunité).

• Le taux d'accroissement de la population « R » est défini par l'équation suivante :

$$dR/dt = v\,I - (d + \gamma)\,R$$

Avec :

En entrée dans la population R :

$+ v\,I$ = Nombre d'individus rétablis par le gain d'une immunité contre la maladie.

En sortie de la population R :

$-(d + \gamma)\,R$ = Nombre d'individus de la population R disparaissant par mort naturelle ou par perte de l'immunité devenant donc des individus sains susceptibles d'être réinfectés.

1.3- Mise en équations des modèles :

Afin de modéliser les systèmes présentés précedemment, il est nécessaire de les transformer en modèles discrets. En effet, certaines fonctions ne peuvent être mises sous forme « continue ». Pour remédier à cela, on utilise ce que l'on appelle des modèles discrets afin de décrire un événement à un temps déterminé encore appelé pas de temps ou intervalle de temps (Δt).

On peut donc dire que la modélisation est en quelque sorte fixée dans le temps. Ceci nous permet de définir des paramètres tels que la population à un temps donné et le pas de temps.

- Le modèle SI :

On obtient, dans le cas de ce modèle, les équations de populations suivantes au pas de temps t+1 :

La population S au temps t+1 est donnée par :

$$S_{t+1} = S_t + (b (S_t + I_t) + v I_t - d S_t - \beta S_t I_t)$$

La population I au temps t+1 est donnée par :

$$I_{t+1} = I_t + (\beta S_t I_t - (\alpha + d + v) I_t)$$

- Le modèle SIR :

On obtient, dans le cas de ce modèle, les équations de populations suivantes au pas de temps t+1 :

La population S au temps t+1 est donnée par :

$$S_{t+1} = S_t + (b (S_t + I_t + R_t) + \gamma R_t - d S_t - \beta S_t I_t)$$

La population I au temps t+1 est donnée par :

$$I_{t+1} = I_t + (\beta S_t I_t - (\alpha + d + v) I_t)$$

La population R au temps t+1 est donnée par :

$$R_{t+1} = R_t (v I_t - (d + \gamma) R_t)$$

II- Les recherches en simulations de propagation de maladies infectieuses :

De nos jours, de nombreux biologistes et de nombreux informaticiens mettent en commun leurs compétences afin d'élaborer des outils permettant de simuler les modèles de propagation de maladies infectieuses dans le but de vérifier le fondement de ces modèles. Deux approches sont fréquemment mises en œuvre. Une première approche en matière de dynamique de population est basée sur la résolution des équations du système différentiel correspondant au modèle étudié. Une seconde approche consiste en des simulations individus-centrées, à savoir l'étude des intéractions entre différents individus.

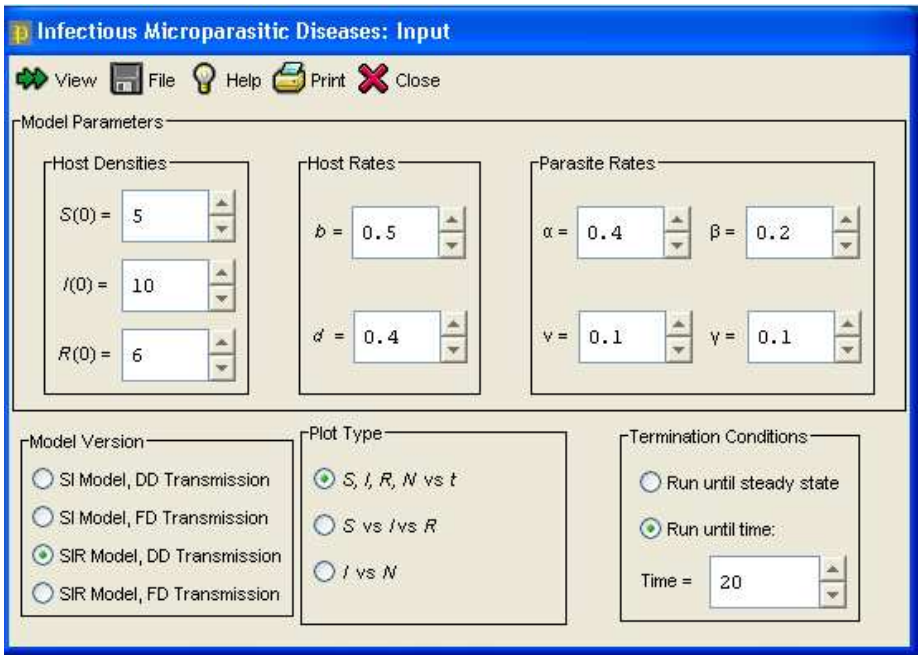
2.1- L'existant en matière de dynamique de population :

Les simulations de dynamique de population reposent uniquement sur du calcul, à savoir la résolution des systèmes différentiels des modèles vus précédemment. Ce type de simulation convient particulièrement à l'étude de l'influence d'un virus sur l'ensemble d'une population, sans tenir compte des particularités de la population telles que la sédentarisation, les habitudes alimentaires ou l'hygiène, et sans tenir compte des particularités du virus telles que le mode de propagation (par voies aériennes ou terrestres, par la nourriture ou l'eau, par le contact avec d'autres individus).

Le logiciel Populus met en œuvre ce type de simulations. Le développement de ce logiciel débuta en 1986 par quatre chercheurs de l'université du Minnesota (Etats-Unis), et fut distribué à partir de 1991. Ce logiciel était à l'origine destiné aux étudiants en biologie afin de leur épargner la résolution de systèmes d'équations et également de leur fournir une approche visuelle afin qu'ils se familiarisent avec la relation entre les paramètres des équations et l'influence que ces derniers ont sur les courbes obtenues. (ALSTAD, 2003).

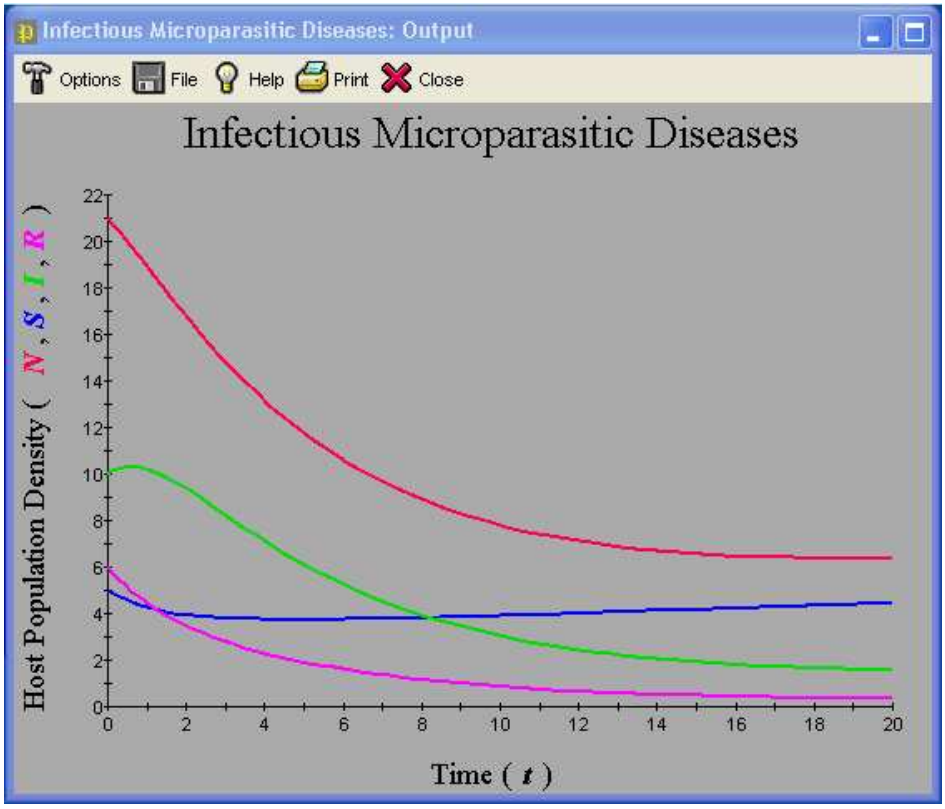
Populus, qui en est à la version 5.3, offre un large choix de modèles dont le modèle *Infectious Microparasitic Diseases* qui permet de traiter les modèles SI et SIR vus précédemment. Ce logiciel est simple d'utilisation. Nous allons voir sur un exemple son fonctionnement. On prendra ici le modèle SIR.

A la demande du modèle *Infectious Microparasitic Diseases*, la boite de dialogue suivante apparaît :

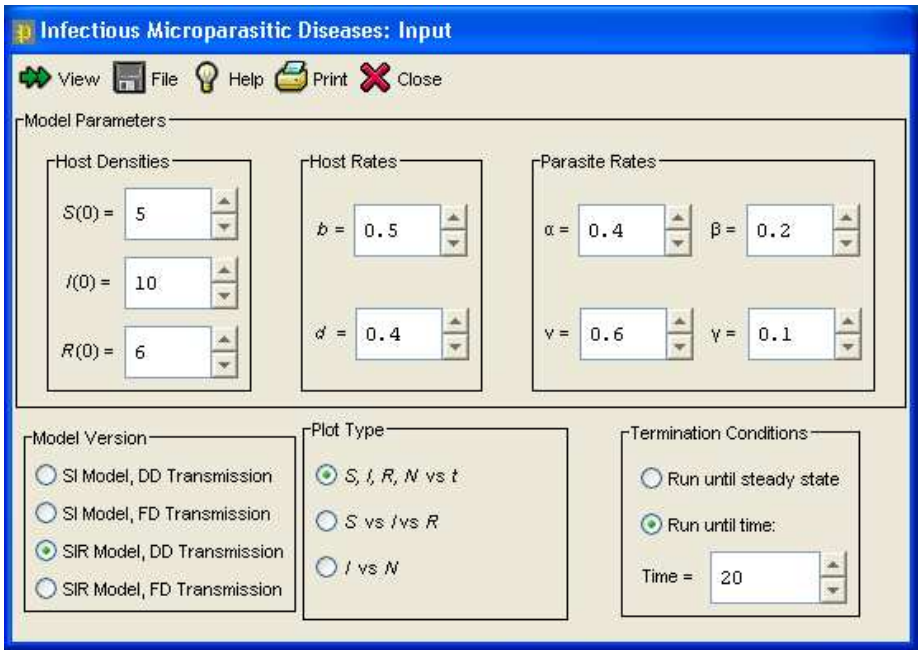


L'utilisateur choisit le modèle. Les différents paramètres qui lui sont demandés correspondent exactement aux paramètres des équations du modèle.

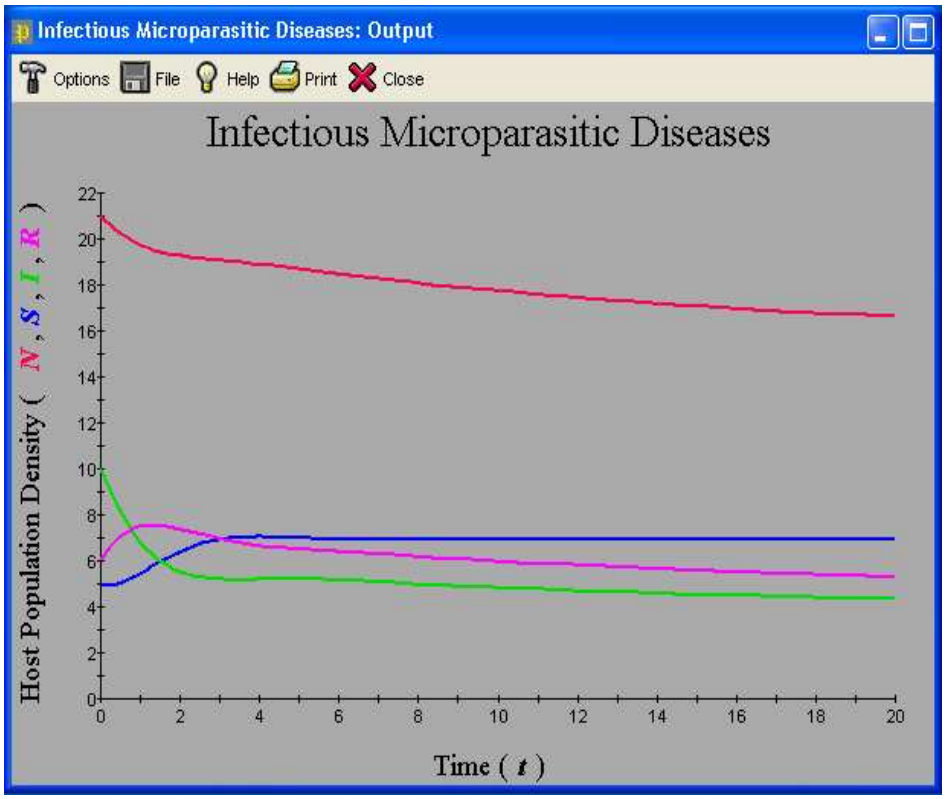
On obtient alors la représentation suivante :



On modifie alors la valeur d'un paramètre : le taux de rétablissement.



On constate qu'on obtient une représentation tout à fait différente de la précédente.



2.2- L'existant en matière de modèles de type individus-centrés :

Les modèles de type individus-centrés sont des simulations basées sur les conséquences globales d'interactions locales entre membres d'une population. Les « individus » peuvent être des personnes, des animaux, ou toute autre entité. Ces modèles consistent en un environnement dans lequel un certain nombre d'individus sont définis. Ces individus obéissent à des lois (ou règles comportementales), et ce sont les interactions entre ces individus qui sont étudiées. Avec ce type de modèle, le comportement de chaque individu peut être observé, contrairement à une modélisation en dynamique de population qui se contente de simuler des changements dans les caractéristiques moyennes d'une population globale. Grâce à ce modèle, il est également possible de modéliser le déplacement d'individus dans le cas d'individus mobiles.

Le projet MOBIDYC (MOdélisation Basée sur les Individus pour la DYnamique des Communautés) élaboré par l'INRA-Avignon (Institut National de la Recherche Agronomique) développe un environnement de création et d'utilisation de modèles de type individu-centré. (INRA, 2004). Il est accessible aux chercheurs car il ne nécessite pas de connaître un langage. Cependant des utilisateurs avertis peuvent l'utiliser en mode programmation. Ce projet s'appuie sur le langage à objet SmallTalk.

Le projet MIMOSA, en cours de réalisation, a pour but de réaliser une plate-forme internationale de simulation générique offrant aux modélisateurs la possibilité de décrire des modèles variés, selon différents points de vue et d'en effectuer la simulation. (LIL-Calais, 2004). Ce projet est sous la responsabilité de plusieurs organismes dont le LIRMM-Montpellier, le CIRAD-Montpellier, l'INRA-Avignon et le CEMAGREF-Clermont-Ferrand. La plate-forme sera validée à partir d'un certain nombre de domaines d'application tels que les migrations urbaines, les bancs de poissons, l'hydrodynamique et structure des sols ou le ruissellement et pourra très certainement être appliquée à l'étude des phénomènes épidémiologiques.

Il existe encore de nombreux logiciels de ce type tels que Cormas développé par le CIRAD qui est spécifique au domaine de la gestion des ressources renouvelables, ou Swarm développé par le Swarm Development Group à Santa Fe. (CIRAD, 2003 et SWARM DEVELOPEMENT GROUP, 2004).

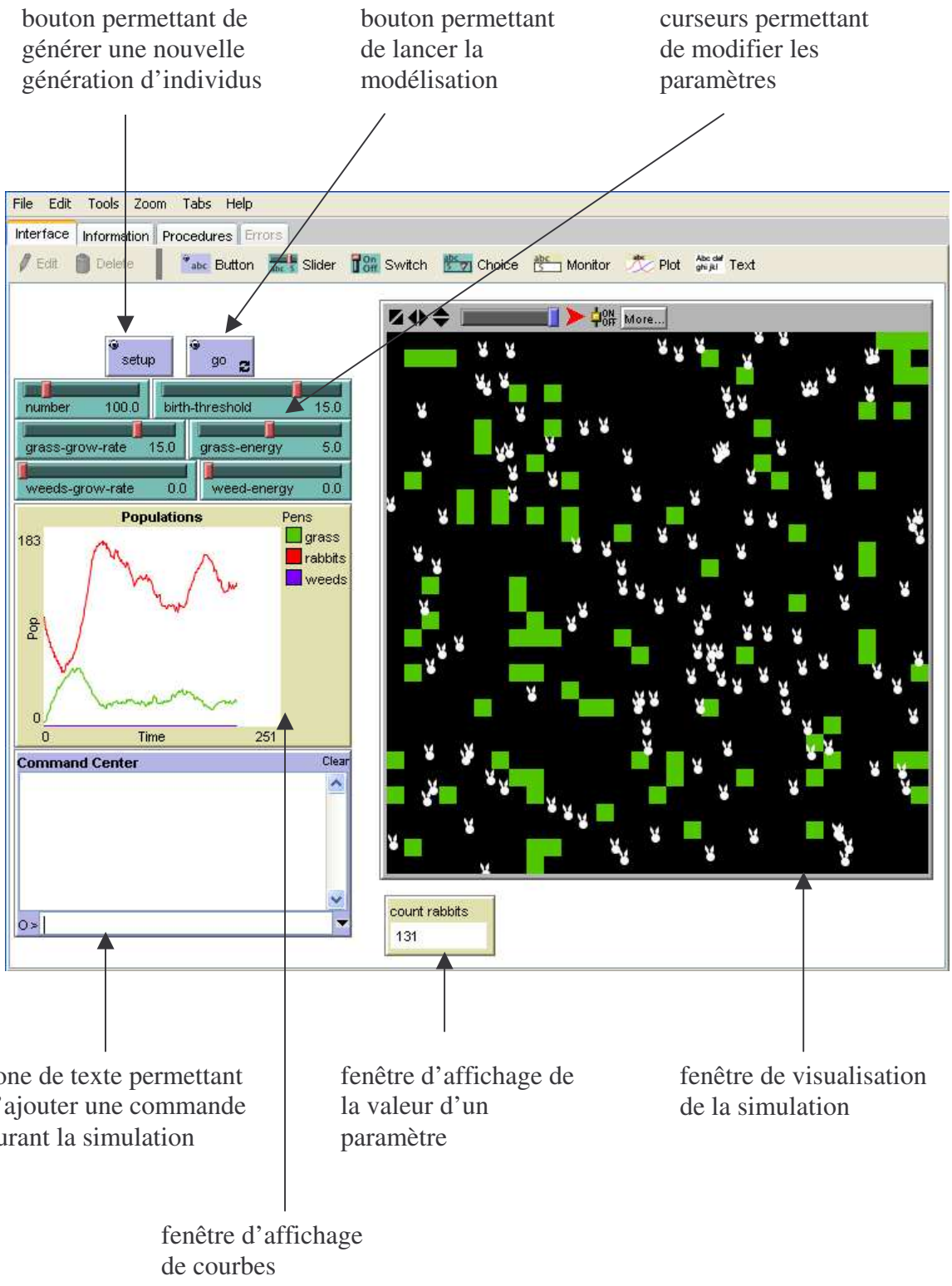
Enfin le logiciel NetLogo, réalisé à l'université américaine Northwestern, fait l'objet de la partie suivante. Il est écrit en Java et on le rencontre encore dans sa version ancienne StarLogo.

III- Le logiciel NetLogo :

NetLogo est un logiciel de modélisation d'environnement pour simuler des phénomènes naturels et sociaux. Il est particulièrement adapté pour modéliser des systèmes complexes de développement à long terme. Les utilisateurs de NetLogo peuvent appliquer des instructions à des centaines ou des milliers d'agents opérant en parallèle. Le logiciel permet ainsi d'étudier la connexion qui existe entre le comportement d'individus isolés et d'en déduire des modèles qui découlent de l'interaction entre de nombreux individus. Ces simulations s'appliquent aux sciences naturelles et sociales, à savoir la biologie et la médecine, la physique et la chimie, les mathématiques et l'informatique, et l'économie.

3.1- L'interface graphique :

Voici sous quelle forme se présente l'interface graphique du logiciel NetLogo :



Le code de la simulation est accessible dans l'onglet « procedures ». Les boutons, les curseurs, les fenêtres d'affichage de courbes et les fenêtres d'affichage de paramètres sont créés à partir de la barre d'outils de l'interface graphique.

3.2- Le langage de programmation :

Les informations suivantes sont tirées du manuel de NetLogo disponible en ligne.

3.2.1- NetLogo et les agents :

Un agent est une entité qui a la possibilité d'interagir avec son environnement et avec d'autres agents. Cette entité peut être une personne, un animal, un insecte, une cellule, un pays, etc… Les agents ont la particularité de pouvoir suivre des instructions ou des lois. En effet, le comportement de chaque agent est déterminé par un ensemble de lois, en général des lois stimulus-réponse typiques et simples.

NetLogo est un logiciel permettant de simuler les interactions entre un grand nombre d'agents, puis de voir ce qui se passe lorsque les agents sont programmés pour suivre des règles spécifiques. NetLogo permet de montrer que lorsque des agents suivent des lois simples, le résultat est parfois complexe et inattendu.

Les agents vivent sur un support 2D constitué de cellules appelées « patches ». Ces cellules ont des coordonnées.

Il y a trois types d'agents :
- les tortues (turtles)
- les patches
- l'observateur.

<u>Les tortues</u> :

Ce sont les êtres vivants. Ils sont appelés « tortues » en hommage au langage de programmation dont dérive NetLogo dans lequel le programmeur contrôle des tortues sur un

écran. Les tortues répondent à des commandes ou des fonctions. Elles peuvent être des centaines, voire des milliers, sur l'écran au même moment.

Les tortues ont pour coordonnées xcor et ycor. Ces coordonnées peuvent être des flottants.

Les patches :

Chaque patch est un carré de sol (une case) sur lequel les tortues peuvent se déplacer. Le patch central a pour coordonnées (0 ;0). Les coordonnées d'un patch sont appelées pxcor et pycor. Le nombre total de patches est déterminé par les variables screen-edge-x et screen-edge-y.

Screen-edge-x : valeur maximale en abscisses.
Screen-edge-y : valeur maximale en ordonnées.

Par défaut, ces deux variables sont à 17. Il y a donc 1225 patches au total sur l'écran. On peut changer ce nombre dans la fenêtre NetLogo's Graphics. Les coordonnées des patches sont des entiers.

Le support des patches n'est pas borné. Chaque patch a le même nombre de voisins. Ainsi si une tortue disparaît d'un côté de l'écran, elle réapparaît du côté opposé.

L'observateur :

L'observateur peut être utilisé pour attribuer des ordres spécifiques à des patches ou à des tortues. Il collecte également des données pour créer des graphiques.

3.2.2- Les différents types de variables :

Il y a trois types de variables : • variable globale
• variable tortue
• variable patch.

Une variable globale n'a qu'une seule valeur pour cette variable, et tous les agents peuvent y accéder. Chaque tortue a sa propre valeur pour toute les variables tortues et chaque patch a sa propre valeur pour toutes les variables patches.

Les variables patches commençent par un « p » pour ne pas les confondre avec les variables tortues.
Ex : Chaque tortue a une variable « color » et chaque patch a une variable « pcolor ».

Il y a des variables prédéfinies : xcor, ycor, heading, pxcor, pycor….
On peut également définir nos propres variables.

Définition des variables :

Les différentes variables se définissent de la manière suivante:

- Variable globale : globals [clock]
- Variable tortue : turtles-own [energy speed]
- Variable patch: patches-own [friction]

On utilise la commande "set" pour les initialiser. Par défaut, elles sont initialisées à 0.

Les variables globales peuvent être lues et modifiées par n'importe quel agent à tout moment. Une tortue peut lire et modifier les variables patches du patch sur lequel elle agit. Lorsque l'on veut qu'un agent lise ou modifie la variable d'un autre agent, il faut mettre –of après le nom de la variable et spécifier, après, de quel agent il s'agit.

Exemples : set color-of turtle 5 red
→ la tortue n° 5 devient rouge.

set pcolor-of patch 2 3 green
→ le patch de coordonnées (2 ;3) devient vert

3.2.3- Les variables locales :

C'est une variable exclusive à une fonction.

On la définit de la manière suivante : locals [var1 var2 …]

Ces variables ne sont utilisables qu'à l'intérieur de la fonction. Elles doivent être définies au début de la fonction, avant toute commande.

3.2.4- Les variables prédéfinies :

Il existe plusieurs variables prédéfinies relatives aux variables « tortues » :

breed : (= race)

Elle permet de sélectionner l'ensemble des tortues de la même race.

Ex : if breed = chat [show « miaou ! »]

breeds:

Ce mot clé permet de définir l'ensemble des races. Il ne peut être utilisé qu'en début de programme, avant toute définition de fonction.

Ex : breeds [souris grenouille]….

color :

Elle définit la couleur d'une tortue.

heading :

Elle indique vers quelle direction la tortue est dirigée.

heading $\in$ [0 ; 360].

0 = Nord, 90 = Est, 180 = Sud, 270 = Ouest

hidden ?

Elle contient un booléen indiquant si la tortue est visible ou non. On peut modifier cette variable pour faire apparaître ou disparaître une tortue.

label :

Elle permet d'associer une valeur à une tortue. Cette valeur peut être de n'importe quel type.

label-color :

Elle détermine la couleur du label d'une tortue (si elle en a un).
label-color ∈ [0 ; 140].

shape :

Elle contient une chaîne de caractères représentant le nom de la configuration courante de la tortue.

size :

La taille par défaut est 1.0. La tortue a la même taille qu'un patch. Toutes les tortues ont la même taille sauf si on coche la case « turtle-sizes ».

who :

Elle contient l'identificateur de la tortue (>= 0). On ne peut pas modifier cette variable.
Ex : show values-from (turtles with [color = red]) [who]
→ liste les identificateurs des tortues rouges.

xcor, ycor :

Ce sont les coordonnées de la tortue.

Il existe également plusieurs variables patches prédéfinies : pcolor, plabel, plabel-color, pxcor, pycor. Leur fonction est la même que pour les variables tortues.

3.2.5- Les fonctions :

Une fonction se définit de la manière suivante : to nom_fonction
corps_de_la_fonction end

Les fonctions prédéfinies du langage NetLogo sont répertoriées dans une page du logiciel appelée *The Primitives Dictionary*.

Il y a deux fonctions à définir obligatoirement dans un programme NetLogo : la fonction setup et la fonction go. C'est à l'utilisateur de définir ces deux fonctions. La fonction setup définit l'état initial du modèle. La fonction go démarre le processus de modélisation.

3.2.6- Les structures :

La structure if :

Syntaxe : if condition [traitement]

Exemple : ask patches [if pxcor > 0 [set pcolor blue]

La structure ifelse:

Syntaxe: ifelse condition [traitement_si_la_condition_est_vérifiée] [traitement_sinon]

Exemple : ask patches [ifelse pxcor > 0 [set pcolor blue] [set pcolor red]]

La structure ask:

Cette structure permet d'appliquer une ou plusieurs instructions à toutes les variables d'un même type. En effet, la structure suivante :

```
ask turtles
[
   …..]
```

permet d'appliquer à toutes les variables de type tortue les instructions qui se situeront dans le corps de la structure. On peut également affiner la sélection en précisant les spécificités de ces variables. En effet, la structure suivante :

```
ask turtles with [ malade? ]
[
   …..
]
```

permet d'appliquer à toutes les variables de type tortue étant malades les instructions qui se situeront dans le corps de la structure.

3.2.7- La gestion des fenêtres d'affichage :

En ce qui concerne l'affichage d'une courbe : après avoir créer une fenêtre permettant l'affichage d'une courbe dans l'interface graphique, il suffit de mettre à jour à chaque pas de temps la valeur de la courbe (ou des courbes) que l'on souhaite afficher. Considérons les lignes de code suivantes :

```
set-current-plot "population"
set-current-plot-pen "total"
plot count turtles
```

La fonction set-current-plot permet de spécifier que l'on va mettre à jour la fenêtre d'affichage appelée « population ». On spécifie alors que l'on va considérer la courbe appelée « total » de cette fenêtre à l'aide de la fonction set-current-plot-pen. Finalement on met à jour la valeur de la courbe en fournissant en paramètre de la fonction plot l'opération à effectuer. Dans notre exemple, on compte le nombre total d'individus.

En ce qui concerne l'affichage d'une fenêtre contenant un paramètre (qui est une variable globale) : après avoir créer une fenêtre permettant l'affichage d'un paramètre, il suffit de remettre à jour à chaque pas de temps la valeur de ce paramètre.

IV- La réalisation du projet :

Nous avons utilisé la bibliothèque logicielle « NetLogo » pour développer des simulations individus-centrées de phénomènes d'épidémiologie. Nous nous sommes basés sur le modèle analytique SIR de propagation de virus qui a été décrit précédemment et nous l'avons adapté à une simulation individu-centrée. Avant de s'intéresser au projet à propement parler, nous nous sommes tout d'abord familiariser avec le langage de programmation de NetLogo à l'aide des programmes disponibles dans la bibliothèque de NetLogo et en créant un programme simple. Nous avons ensuite procéder à l'élaboration du programme principal par étapes successives. Ce programme principal modélise la propagation d'un virus quelconque se transmettant par contacts entre individus. Finalement nous avons modifié le programme obtenu pour simuler la propagation d'un virus « de ville en ville ».

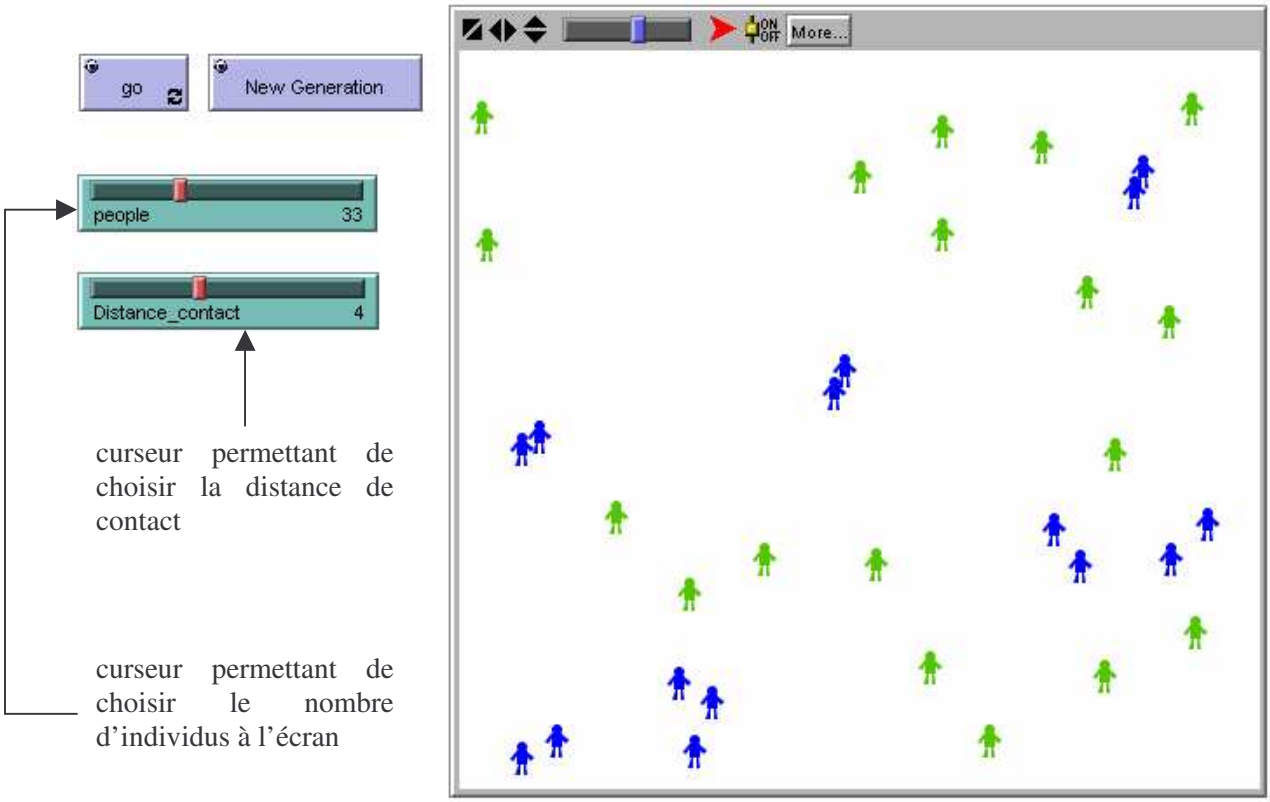
4.1- Familiarisation avec le langage :

La bibliothèque de NetLogo propose un large panel de programmes mettant en œuvre différents types de modèles comme la propagation de maladies, les relations proies-prédateurs ou le déplacement de fourmis. Nous nous sommes aidés du manuel d'utilisation de NetLogo pour comprendre ces différents programmes.

Nous nous sommes ensuite exercés à utiliser ce langage de programmation en créant un petit programme simple qui permet de mettre en mouvement un certain nombre d'individus et de les faire changer de couleur si la distance entre un individu et l'un de ses voisins est inférieure à une distance donnée.

Pour commencer, nous avons créé une fonction, appelée new_generation(), qui définit une population de départ. Cette fonction crée autant d'individus que demandé. Ces individus sont verts et disposés aléatoirement sur l'écran. Le nombre d'individus est défini par l'utilisateur à l'aide d'un curseur. On lance alors le processus de modélisation. Les individus se déplacent sur l'écran. Pour cela, nous avons défini une fonction qui, à chaque pas de temps, modifie l'orientation de l'individu et le fait avancer d'une unité. Une autre fonction permet de calculer, à chaque pas de temps également, la distance entre chaque individu et chacun de ses voisins. Si la distance entre deux individus est inférieure à une distance qui a été préalablement définie par l'utilisateur à l'aide d'un curseur, les deux individus deviennent rouges.

Voici une illustration de ce programme :



On choisit une distance de 4 unités. Les individus bleus sont ceux dont au moins une distance les séparant de l'un de leurs voisins est inférieure à 4 unités, et les individus verts sont ceux dont les distances les séparant de leurs voisins sont toutes supérieures à 4 unités.

Le code de ce programme est fourni en annexe.

4.2- Les étapes de l'élaboration du programme final :

A partir d'une population donnée, nous avons commencé par faire se reproduire les individus et les faire mourir au bout d'une durée déterminée. Nous avons par la suite introduit le virus au sein de la population, ce qui a consisté à faire mourir prématurément les individus infectés et à propager le virus par contacts entre individus. Enfin nous avons donné la possibilité aux individus de se rétablir et de devenir immunisés.

4.2.1- Mise en place de naissances et de morts d'individus :

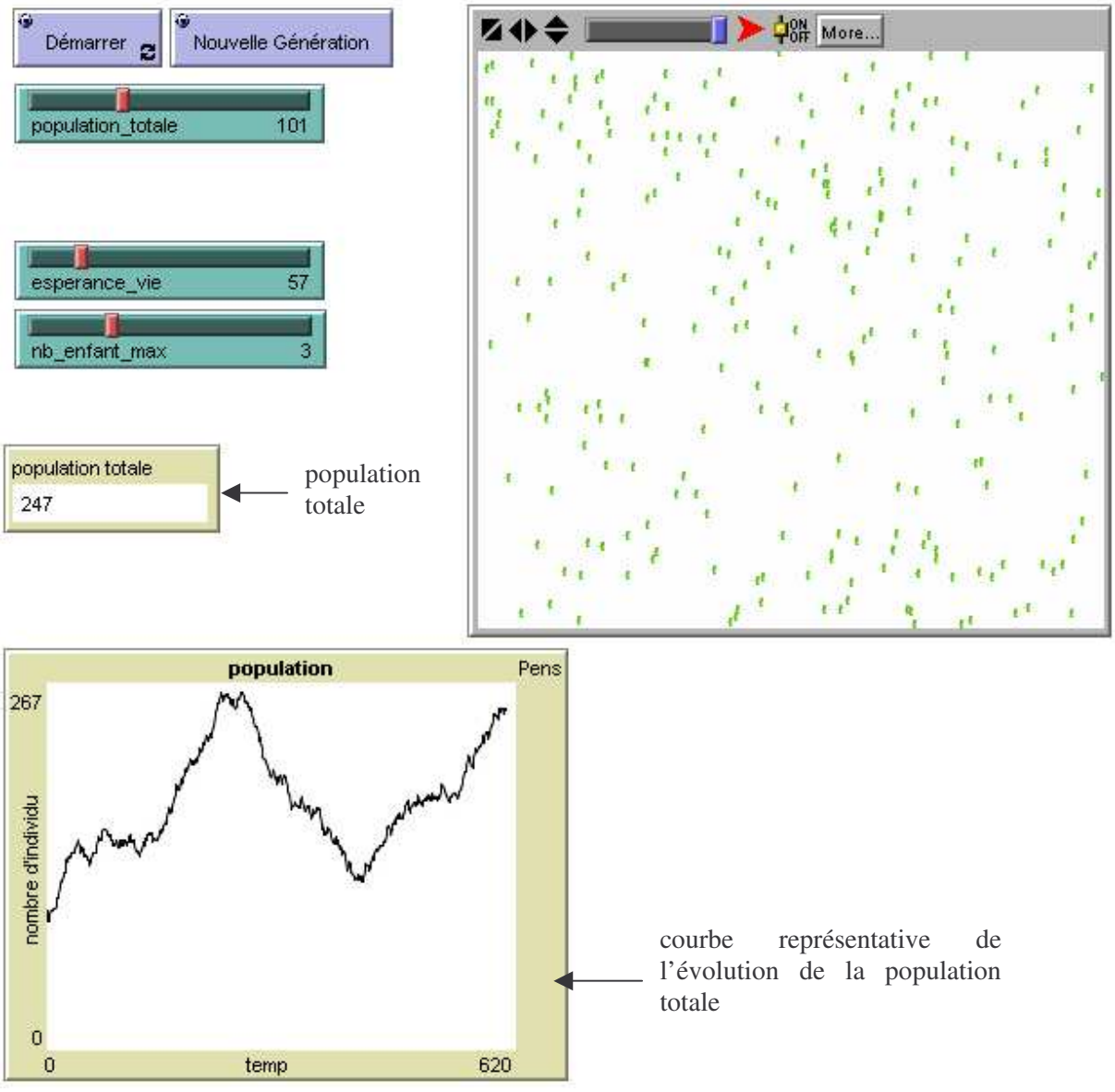
Nous avons conservé le déplacement d'individus du programme décrit dans la partie précédente. Contrairement aux modèles présentés dans la première partie, le nombre de naissances et de morts n'est pas calculé à l'aide d'un taux car dans le cas contraire, la population atteindrait systématiquement et rapidement l'équilibre.

En ce qui concerne les naissances, nous avons défini un nombre maximal de naissances par individu, dont la valeur est propre à chaque individu. En effet, l'utilisateur définit à l'aide d'un curseur le nombre maximal global d'enfants par individu. Puis, à chaque fois qu'un individu est créé, nous attribuons à cet individu un nombre maximal d'enfants aléatoire qui lui est propre et qui est inférieur au nombre défini par l'utilisateur. Pour déclencher une naissance, on considère à chaque pas de temps les individus qui n'ont pas encore atteints le nombre maximal d'enfants qui leur a été attribué. On génère alors un nombre aléatoire entre 0 et 100. Si ce nombre est inférieur au nombre d'enfants que l'individu peut avoir multiplié par 1.5, alors on crée un nouvel individu. La multiplication du nombre maximal d'enfants par individu par 1.5 s'explique par le fait que l'on considère qu'un individu a de fortes chances d'avoir eu tous ses enfants durant les trois premiers quarts de sa vie.

Le système de génération des morts est similaire à celui des naissances. Nous avons défini une espérance de vie propre à chaque individu. L'utilisateur définit à l'aide d'un

curseur une espérance de vie moyenne par individu. Puis, à chaque fois qu'un individu est créé, on lui attribut une espérance de vie propre qui est l'espérance de vie moyenne plus ou moins 20. On vérifie alors pour chaque individu et à chaque pas de temps que l'âge de l'individu ne dépasse pas son espérance de vie, sinon il est tué.

Voici une illustration de ce programme :



population totale

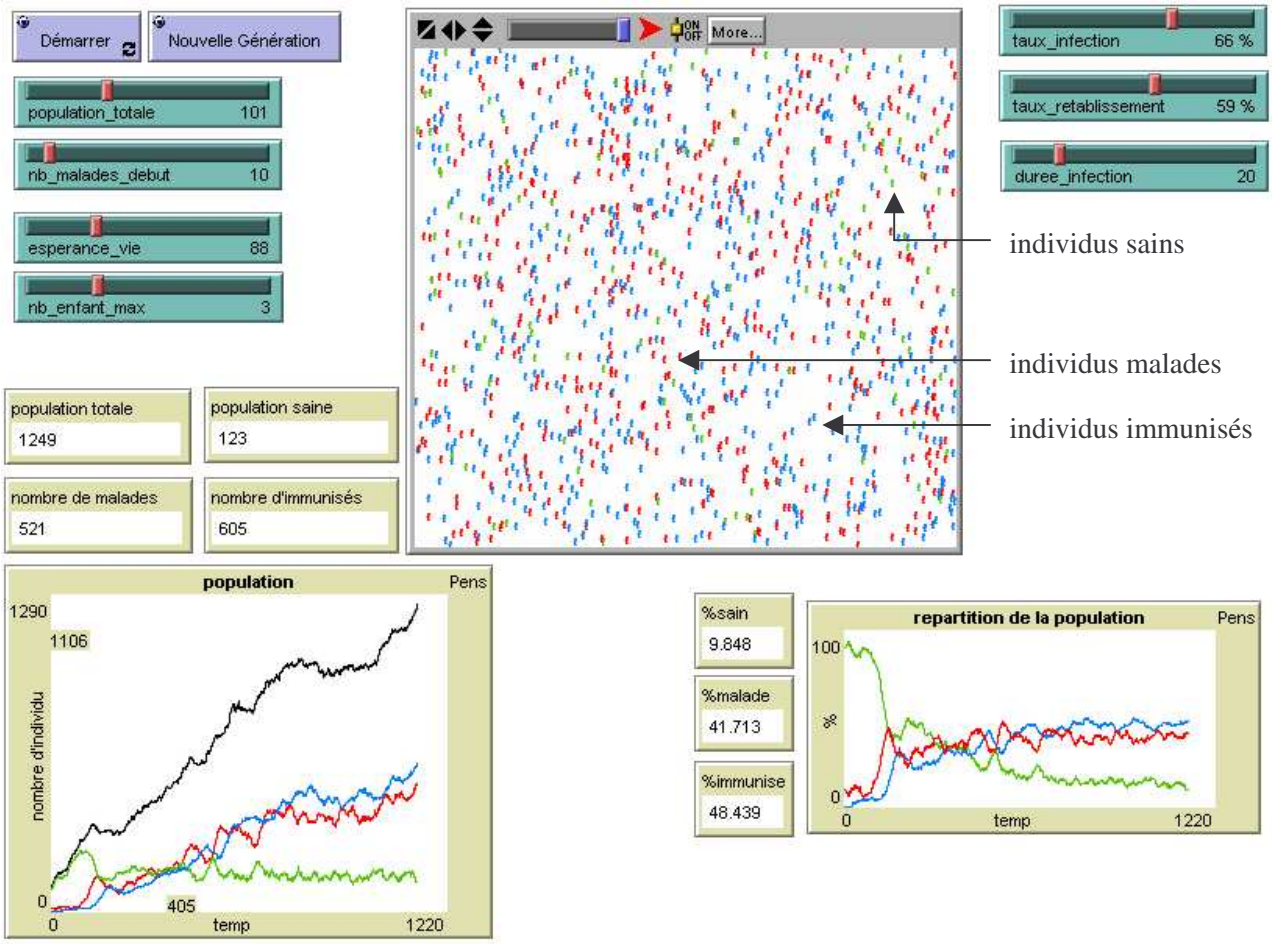
courbe représentative de l'évolution de la population totale

Tous les individus sont ici identiques. C'est la courbe représentative de l'évolution de la population qui est intéressante à observer. On voit que de manière générale la population augmente. C'est une évolution normale car les individus ne meurent que de mort naturelle. La population diminue seulement s'il y a peu de reproductions durant une certaine période.

4.2.2- Introduction du virus :

Nous allons à présent introduire un virus dans la population. On considère un virus qui se transmet par contact entre les individus. On commence par fixer le nombre d'individus malades de départ. On considère ensuite chaque individu malade. Pour chacun d'entre eux, on regarde s'il y a un individu sain à proximité. Si oui, on génère aléatoirement un nombre entre 0 et 100 et si ce nombre est inférieur au taux d'infection défini par l'utilisateur (ce qui correspond au taux « β » dans le modèle analytique SIR), l'individu sain devient malade. On considère alors une durée d'infection minimum définie par l'utilisateur. L'individu devenu malade reste infecté durant au moins cette durée d'infection minimum. Puis, étant donné que pour une modélisation individus-centrée on ne peut pas se contenter d'avoir des caractéristiques identiques pour chaque individu, on fait en sorte que la durée d'infection de chaque individu infecté soit différente. A partir du moment où l'individu a dépassé la durée d'infection minimum, plus le temps passe plus l'individu a de chances de devenir immunisé ou de risques de mourir. Le choix du rétablissement ou de la mort de l'individu se fait en fonction du taux d'immunisation (ce qui correspond au taux de rétablissement « ν » dans le modèle analytique SIR) : on génère à nouveau un nombre aléatoire entre 0 et 100, si ce nombre est inférieur au taux d'immunisation l'individu devient immunisé sinon il meurt.

Voici une illustration de ce programme :



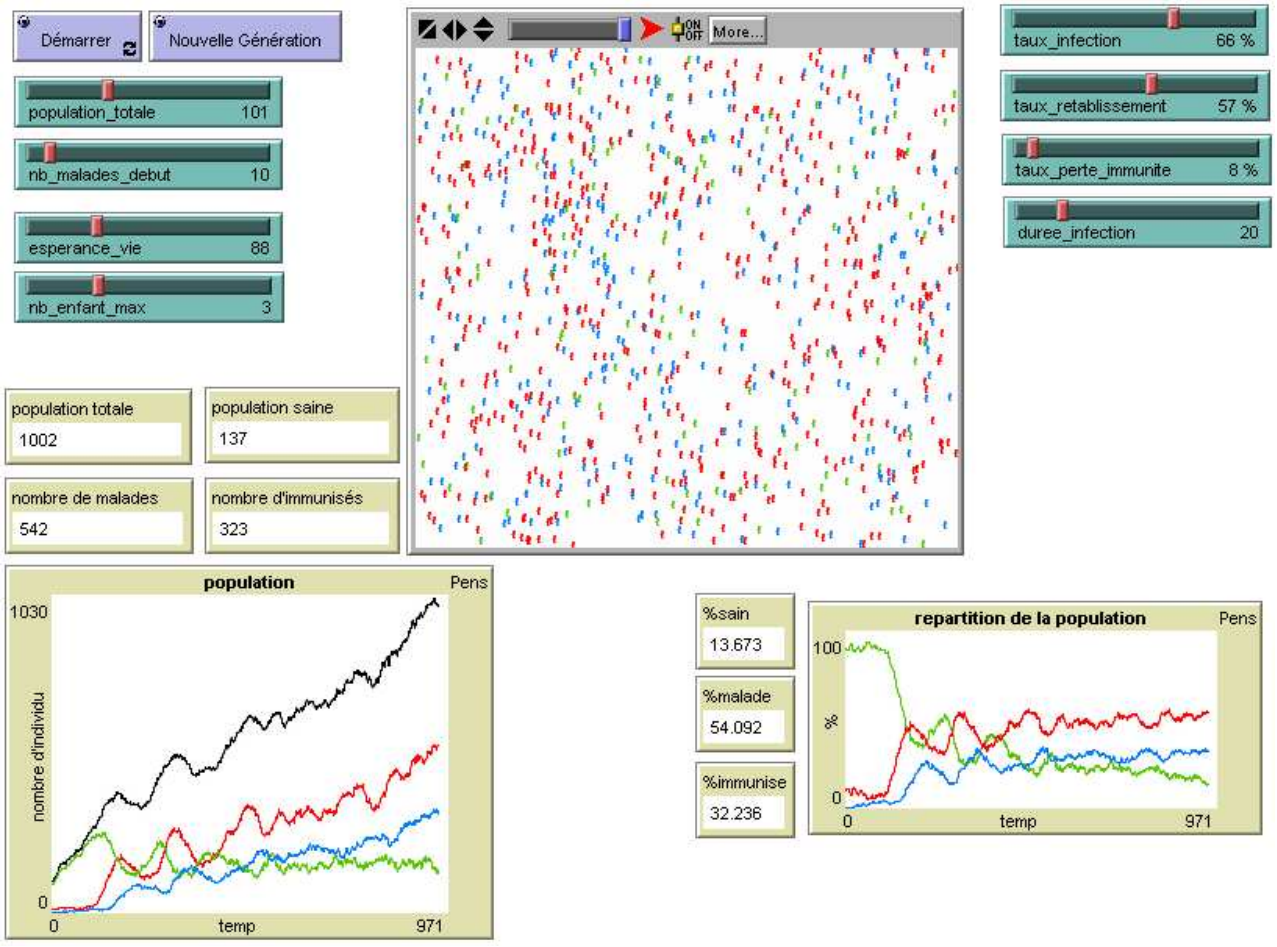
La population saine est représentée en vert, la population infectée en rouge et la populaton immunisée en bleu. On observe que, bien que la population totale ne cesse d'augmenter, la répartition de ces trois populations est relativement stable dans le temps. Le maintien d'un pourcentage moyen d'individus sains est assuré par les naissances.

4.2.3- Mise en place de la perte d'immunité :

Maintenant que le virus est introduit dans la population, nous pouvons considérer les déficiences immunitaires qui peuvent entrainer une perte d'immunité. Pour cela, on considère un taux de perte d'immunité (ce qui correspond au taux « γ » dans le modèle analytique SIR) qui est exprimé en pourcentage par tranche de 10% de l'espérance de vie. Cette unité a été choisie en prenant en compte que les rappels de vaccination sont à faire tous les 10 ans

environ. A chaque pas de temps et pour chaque individu, nous prenons un nombre aléatoire entre 0 et 100 et si ce nombre est inférieur au taux de perte d'immunité divisé par 10% de l'espérance de vie, l'individu redevient sain.

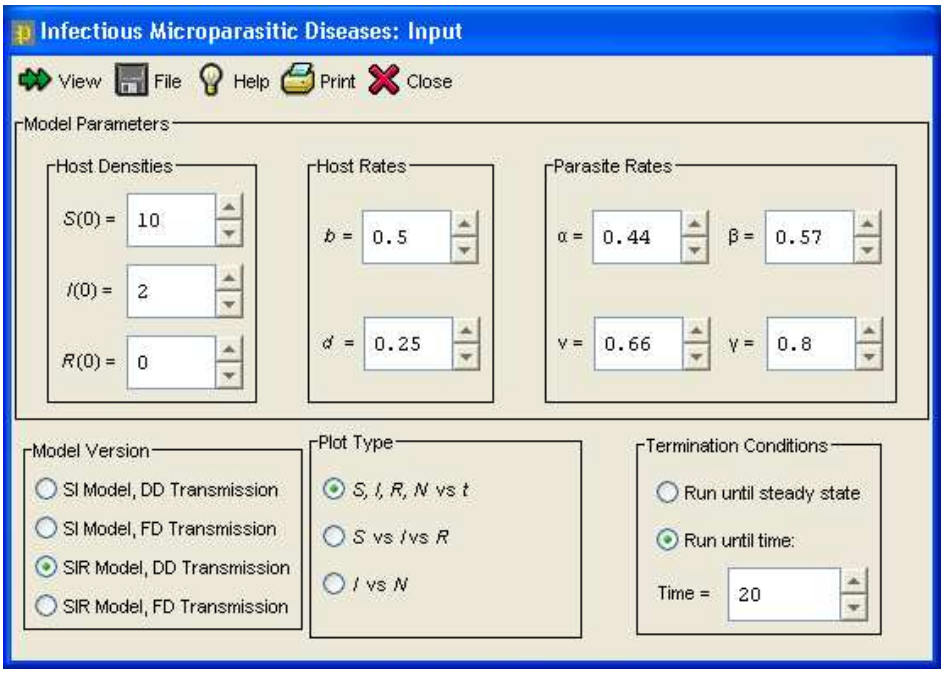
Voici une illustration de ce programme :



On constate que le pourcentage d'individus sains et le pourcentages d'individus malades dans la population totale restent quasiment les mêmes. Seul le pourcentage d'individus immunisés varie : il est cette fois-ci plus faible que le pourcentage d'individus malades dans la population. Ceci s'explique par le fait que de nombreux individus malades redeviennent à présent sains par perte de leur immunité. La répartition des trois populations est également stable dans le temps.

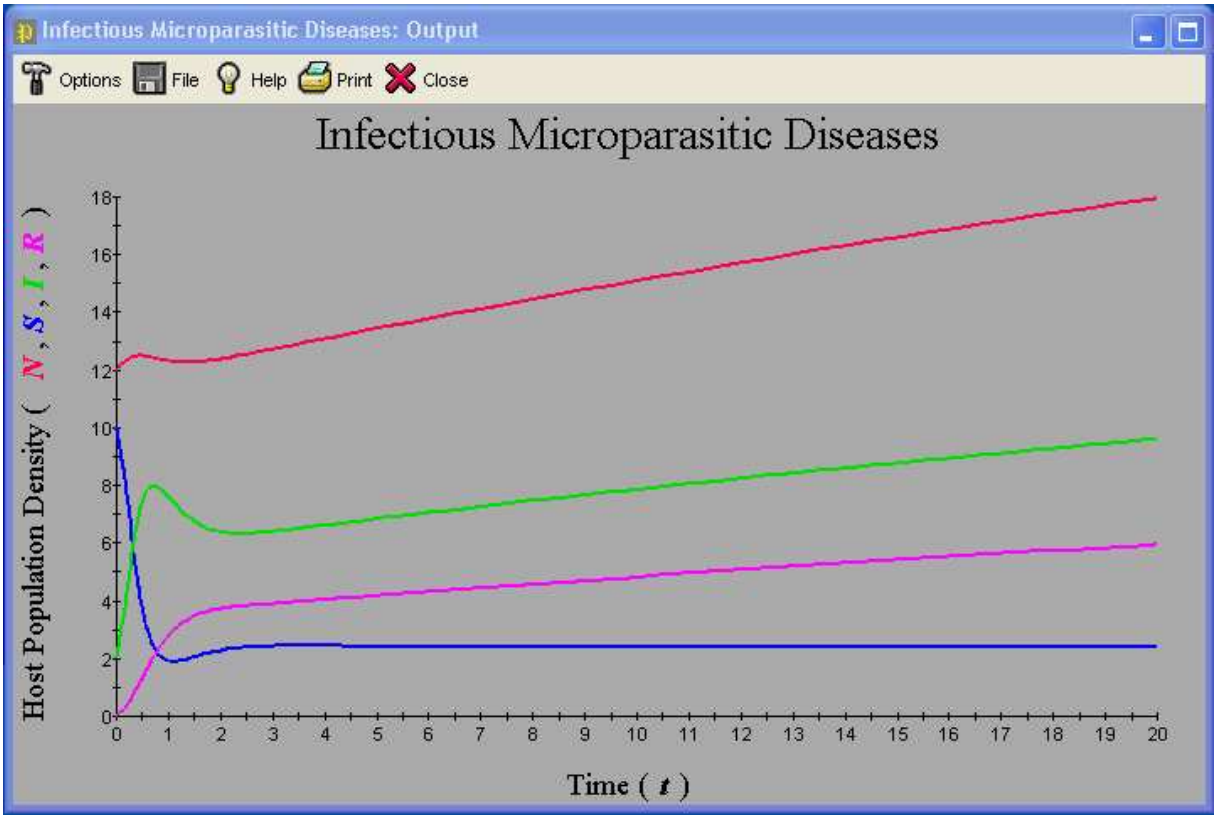
4.3- Comparaison avec un autre logiciel : Populus :

Nous avons comparé les résultats obtenus précédemment à d'autres résultats obtenus à l'aide du logiciel Populus qui permet d'étudier des dynamiques de population. Pour cela, nous avons entrer approximativement les mêmes valeurs de paramètres que celles choisies dans NetLogo.



La répartition de la population de départ (population saine et population infectée) a été faite de manière proportionnelle à celle de NetLogo. Cependant la valeur de ces paramètres a peu d'importance car on ne s'intéresse à la répartition de la population qu'une fois que celle-ci a atteint un état stable dans le temps. Intéressons nous à présent aux taux concernant la propagation du virus. Le taux de transmission et le taux de rétablissement sont conservés. Le taux α correspondant au taux de mortalité due à l'infection est fixé à $1-\nu$ car on considère que, suite à la période d'infection, un individu meurt ou se rétablit. Le taux de perte d'immunité γ est quant à lui multiplié par 10 car dans notre simulation nous considérons que le taux de perte d'immunité est un taux calculé sur 10% de l'espérance de vie. Ceci s'explique par le fait que la perte d'immunité ne peut pas se représenter de manière continue dans le temps. Nous avons donc fixer la durée de perte d'immunité à 10% de l'espérance de vie compte tenu du fait que les rappels de vaccination s'effectuent en moyenne tous les 10 ans. Enfin, étant donné que les taux de natalité et de mortalité n'existent pas dans notre simulation NetLogo puisque nous

utilisons une espérance de vie moyenne ainsi qu'un nombre d'enfants maximum par individu, ces derniers sont fixés de manière à ce que l'évolution moyenne de la population totale corresponde à celle obtenue avec la simulation NetLogo. On obtient alors les courbes suivantes :

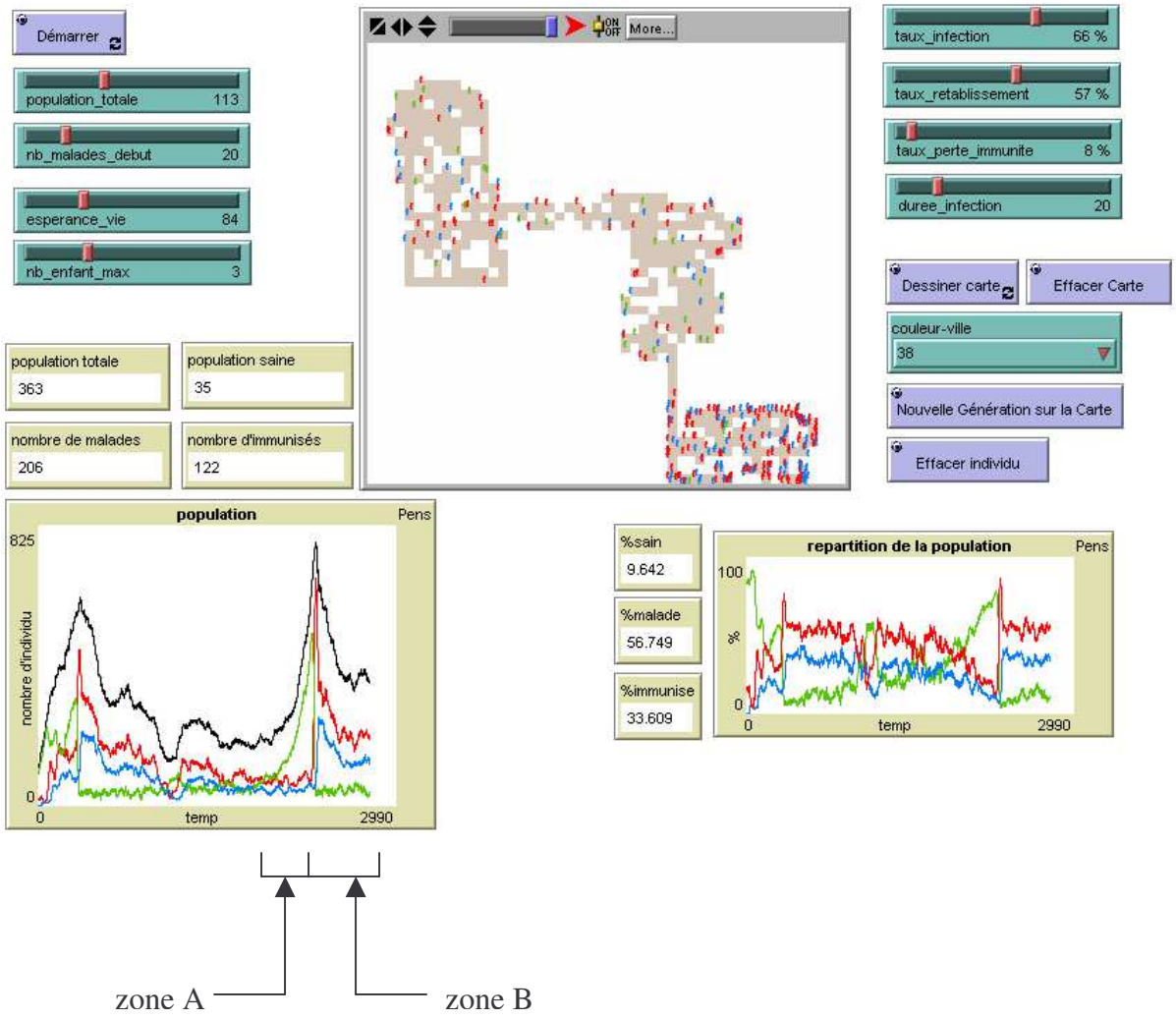


On constate que la répartition des individus sains, infectés et immunisés est la même que pour notre simulation.

4.4- Une extension du projet : propagation d'un virus « de ville en ville » :

Comme une simulation multi-agent permet de modéliser des systèmes plus complexes que les modèles basés sur des équations différentielles d'évolution de populations globales, nous avons tenté de construire une simulation un peu plus complexe. Cette simulation limite le déplacement des individus à certaines zones, ce qui permettrait par exemple de visualiser la propagation d'un virus de ville en ville.

A l'aide de patches, nous pouvons définir avec la souris des zones où les individus peuvent se déplacer et d'autres qui leurs sont interdites.



Avec cette nouvelle simulation, nous constatons que les courbes d'évolution des populations sont totalement différentes des précédentes. Ceci est dû au fait que le déplacement des individus est assez aisé dans certaines zones et plus difficile dans d'autres. Ainsi plus le déplacement dans une zone est difficile, plus la propagation du virus dans cette zone est lente. On peut alors assister à la disparition du virus dans une zone donnée à un moment donné, ce qui implique une augmentation brutale de la population saine (cf zone A). Puis lorsque le virus parvient à nouveau dans cette zone, il infecte brutalement les individus présents qui étaient jusqu'alors isolés du virus (cf zone B).

L'élaboration de notre simulation s'est appuyée sur le modèle SIR de propagation des maladies infectieuses. Des adaptations de ce modèle ont été effectuées afin de définir pour les agents un comportement individuel, notamment concernant le taux de mortalité naturelle qui a été remplacé par une espérence de vie moyenne, et le taux de natalité qui s'est transformé en un nombre maximum d 'enfants par individu. Les résultats de la simulation ont été confirmés par le logiciel Populus avec lequel nous avons obtenus des courbes semblables aux nôtres. Cependant, il serait certainement plus pertinent d'élaborer les comportements individuels des agents à partir de données concernant la propagation d'un virus d'individu à individu et non au sein d'une population globale.

Ce projet a été enrichissant dans la mesure où il nous a permis de découvrir un nouveau mode de programmation orienté agent grâce au logiciel NetLogo. Il a également été intéressant de voir ce que peut apporter l'informatique à une autre discipline, en l'occurrence la biologie.

BIBLIOGRAPHIE

Ressources papier :

N. F. BRITTON. *Essential Mathematical Biology*. Springer. Chapitre 3.

Ressources électroniques :

NORTHWESTERN UNIVERSITY. (2004). *NetLogo*. Page consultée le 12/04/04. http://ccl.northwestern.edu/netlogo/

NORTHWESTERN UNIVERSITY. (2004). *NetLogo User Manual version 2.0.1*. Page consultée le 12/04/04. http://ccl.northwestern.edu/netlogo/docs/

ALSTAD D., Université du Minnesota (Etats-Unis). (2003). *Populus, Simulations of Population Biology*. Page consultée le 29/04/04. http://www.cbs.umn.edu/populus/

INRA. (2004). *Le projet Mobidyc*. Page consultée le 12/06/04. http://www.avignon.inra.fr/internet/unites/biometrie/mobidyc_projet/version_index_html

LIL-Calais. (2004). *Projet MIMOSA*. Page consultée le 11/06/04. http://lil.univ-littoral.fr/Mimosa/

CIRAD. (2003). *Ressources naturelles et simulations multi-agents*. Page consultée le 12/06/04. http://cormas.cirad.fr/

SWARM DEVELOPMENT GROUP. (2004). *Main Page – Swarm Wiki*. Page consultée le 11/06/04. http://wiki.swarm.org/wiki/Main_Page

ANNEXES