Modelare bazată pe agenţi în NetLogo

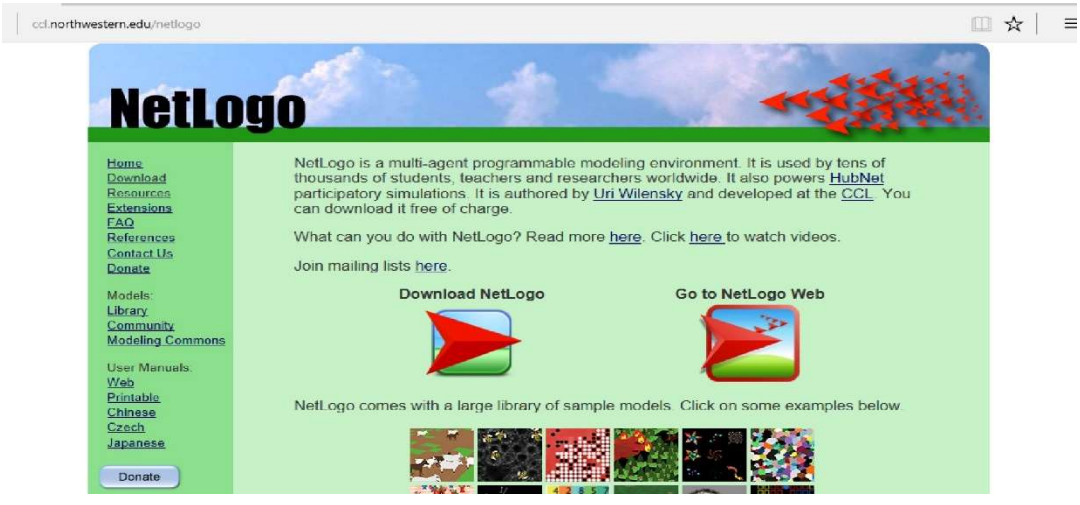
Conceptele cheie utilizate în definirea agentului

Conceptele cheie utilizate în definirea agentului sunt:

- **poziţionarea în raport cu mediul** - agentul primeşte inputuri de la mediul său şi poate executa acţiuni care schimbă mediul într-un anumit fel;
- **autonomia** - un agent funcţionează fără intervenţia directă a omului sau a altor agenţi, agentul îşi poate controla propriile acţiuni şi starea sa internă
- **raţionalitatea** - un agent este capabil să exercite un comportament orientat către un anumit scop şi să iniţieze acţiuni în scopul maximizării performanţei sale în raport cu o funcţie de evaluare, pentru a-l apropia de atingerea scopului;
- **responsivitatea** - agentul percepe mediul şi comportamentul altor agenți, răspunzând la timp la schimbările ce apar;
- **latura socială -** agentul este capabil să interacţioneze cu alţi agenţi de calcul sau umani pentru a-şi rezolva propriile probleme şi pentru a ajuta alţi agenţi în activităţile lor.

NetLogo

NetLogo este o aplicaţie dedicată modelării bazate pe agenţi. Pentru accesarea NetLogo : https://ccl.northwestern.edu/netlogo/

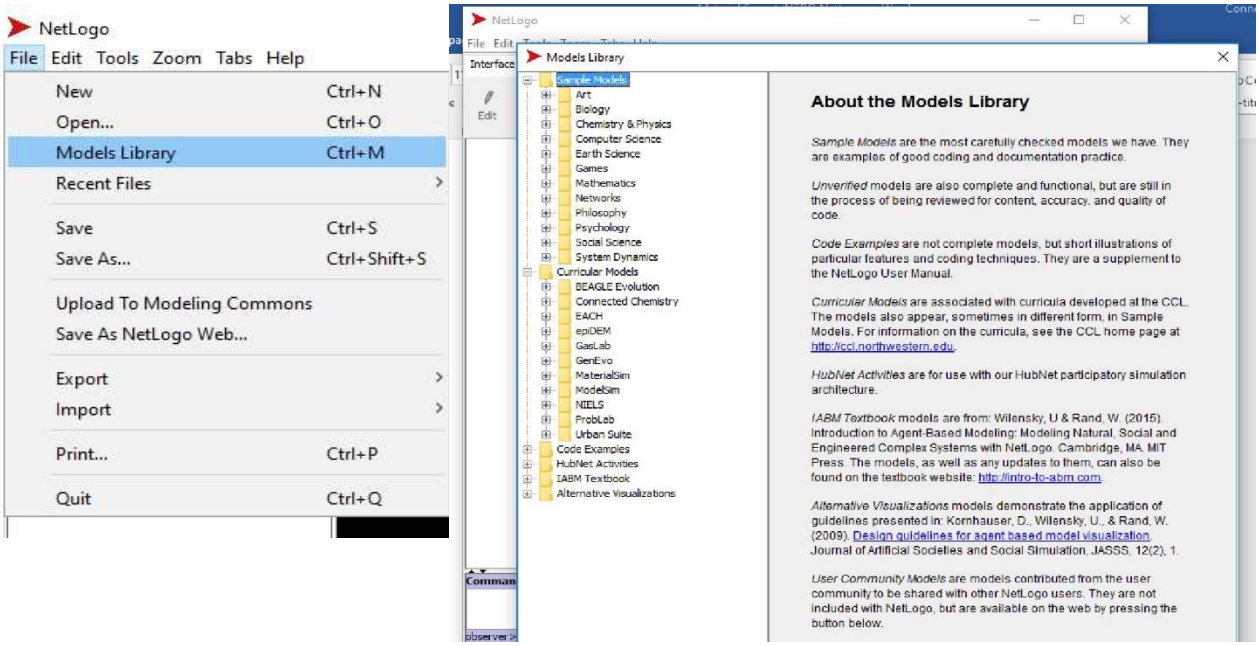


Pentru accesarea librăriei de modele NetLogo: din meniul *File*, se alege opţiunea *Models Library.*

Se accesează astfel o colecţie de peste 200 modele (*sample models*) din diferite arii de aplicabilitate: artă, biologie, chimie şi fizică, calculatoare, geografie, jocuri, matematică, reţele, filozofie, psihologie, ştiinţe sociale, dinamică de sistem. O parte dintre modelele NetLogo din librărie servesc drept suport de curs şi/sau seminar în cadrul unor discipine predate la diverse universităţi (*curricular models*). Câteva funcţionalităţi ale NetLogo sunt exemplificate prin porţiuni de cod sursă (*code examples*): pentru citirea unui fişier în NetLogo (*File Input Exampl*), pentru construirea unui graf complet în care fiecare nod este conectat cu fiecare alt nod, etc.

Versiunea Web a manualului de utilizare *NetLogo*:

http://ccl.northwestern.edu/netlogo/docs/

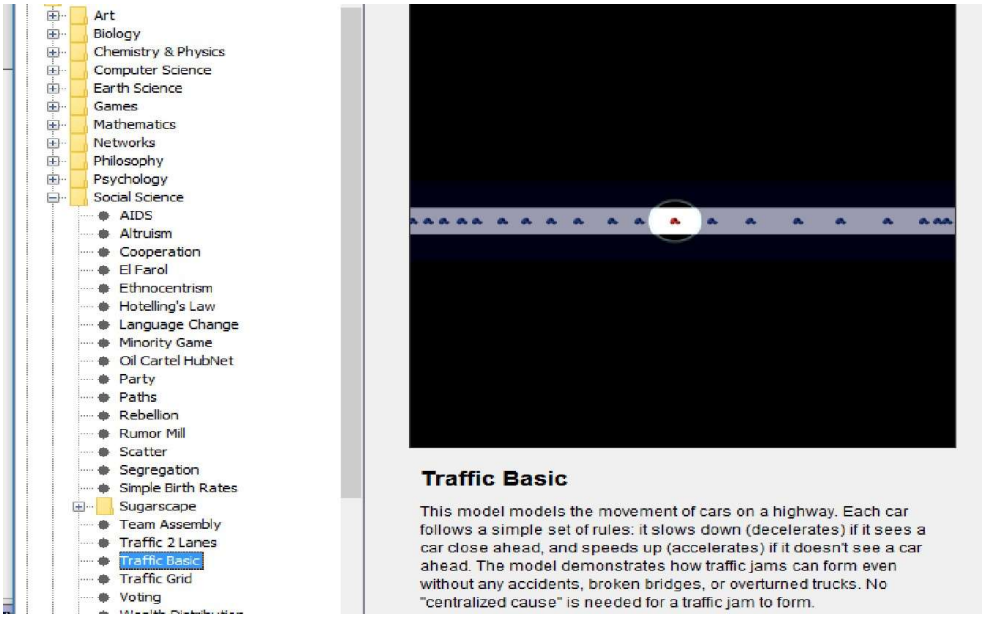


Modelul NetLogo *Traffic Basic* exemplifică agenţii şi caracteristicile lor

Se accesează modelul **«Traffic Basic »** din Models Library \ Sample Models\Social Science.

Traffic Basic modelează mişcarea maşinilor pe un drum rutier, cu scopul de a facilita studiul formării blocajelor în trafic.

Modelul arată că se pot forma blocaje în trafic fără o cauză specială, cum ar fi un accident între 2 maşini. Comportamentul unui şofer - agent de a frâna are drept consecinţe în model faptul că toţi ceilalţi agenţi se vor deplasa cu o viteză redusă pentru o perioadă de timp.



Agenţii în model sunt reprezentaţi de maşini participante la trafic.

- **poziţionarea în raport cu mediul** - agentul primeşte inputuri de la mediul său (ce alte maşini sunt în trafic, banda şi direcţia pe care se poate deplasa, etc) şi poate executa acţiuni care schimbă mediul (frânarea sau accelerarea din pare unui agent va schimba viteza traficului de maşini pentru o perioadă de timp);

- **autonomia** – o maşină – agent din model poate funcţiona fără intervenţia utilizatorului, odată modelul pornit în execuţie; prin intermediul regulilor de frânare şi de accelerare programate în model, un agent îşi poate controla acţiunile de frânare şi de accelerare în trafic;

- **raţionalitatea** -fiecare agent este capabil să exercite un comportament orientat către scopul de a evita coleziunea cu un alt agent şi cel de a accelera dacă nu are alţi agenţi în faţa sa;

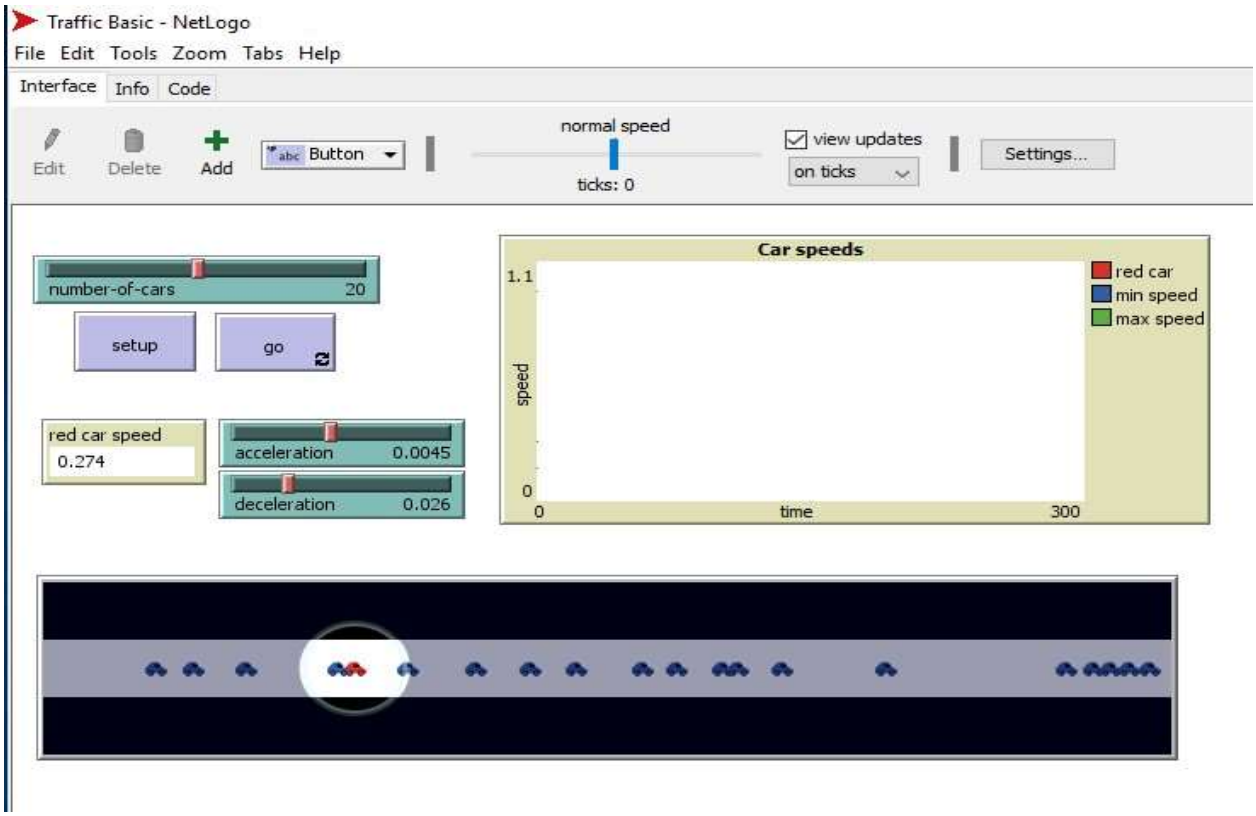
- **responsivitatea** – agenţii percep mediul şi comportamentul altor agenți, răspunzând la schimbările ce apar

    - utilizatorul poate schimba numărul de maşini de pe drumul rutier, prin intermediul slider-ului poziţionat pe interfaţă (mişcare stânga - dreapta), iar agenţii din model se vor adapta la noul număr de participanţi la trafic

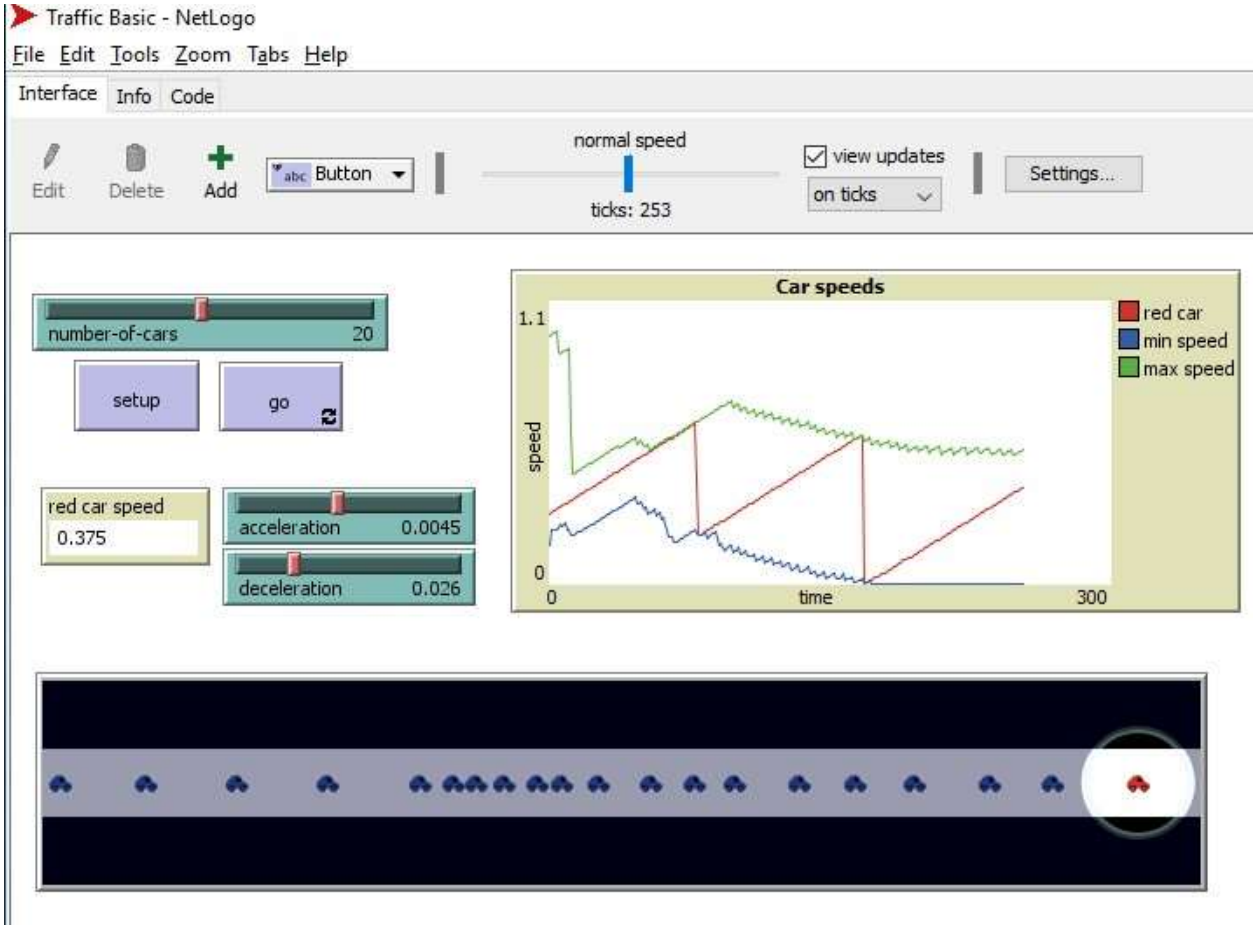
    - utilizatorul poate de asemenea schimba: cu cât va accelera o maşină care nu are alte maşini în faţă (acceleration slider) sau cu cât va merge mai încet o maşină

faţă de maşina din faţă când frânează (deceleration slider), iar agenţii din model se vor adapta la noile valori ale puterii de accelerare, respective frânare

- **latura socială –** agenţii pot interacţiona cu utilizatorul atunci când acesta lansează comenzi în fereastra *Command Center* sau direct in codul sursă al modelului; cu alţi agenţi din model, putem considera interacţiune comportamentul de frânare, care are în model rolul de a evita un contact direct între doi agenţi.



- In secţiunea de grafice privind evoluţia modelului (« Plot area ») utilizatorul poate monitoriza trei valori:
  - cea mai mare viteză din trafic (a oricărei maşini)
  - cea mai mică viteză din trafic (a oricărei maşini)
  - viteza uneia dintre maşini care de la început este colorată în model cu o culoare distinctă

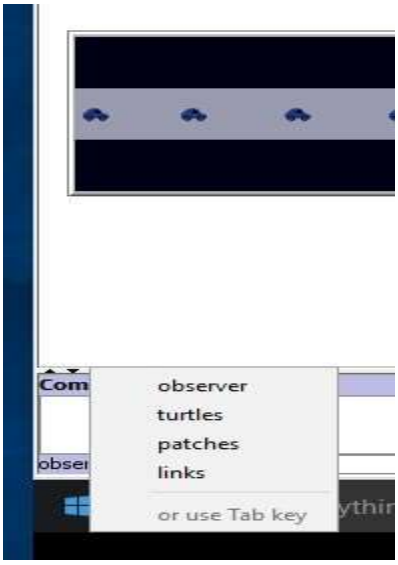


Provocare: schimbarea unor setari implicite din interfata pentru fluidizarea traficului:

- micsoram setarea implicita pentru “deceleration”;
- marim setarea implicita pentru “acceleration”;
- scadem setarea implicita pentru “number of cars”

Utilizarea unor comenzi în fereastra « Command Center »

- utilizatorul poate lansa comenzi in cadrul modelului din patru perspective:



- “turtles” – perspectiva agenţilor modelului
- “patches” – perspectiva mediului în care agenţii se deplaseaza, interactioneaza
- “links” – perspectiva conexiunilor dintre agenţi
- “observer” – perspectiva persoanei care observă setarea şi execuţia modelului

Din perspectiva observatorului, se va utiliza sintagma “ask” pentru a da comenzi, iar din perspectiva agenţilor “turtle”, nu este nevoie să se utilizeze cuvântul “ask” pentru a lansa o coandă în model.

Comenzile scrise în fereastra “Command Center” nu au efect permanent (sunt valabile doar până la o nouă apăsare a butonului «Setup »)

Setarea şi modificarea culorilor, formelor, mărimilor şi etichetelor în NetLogo

Opţiunea color – permite schimbarea culorii agenţilor.
Opţiunea pcolor – permite schimbarea culorii mediului în care activează agenţii.
Opţiunea shape – permite schimbarea formei agenţilor.
Opţiunea size - permite schimbarea marimii agenţilor
Opţiunea label – permite afisarea unui text in dreptul unui agent sau în mediu

Net Logo recunoaşte 16 nume de culori care se pot folosi în comenzi, restul culorilor se identifică numeric.

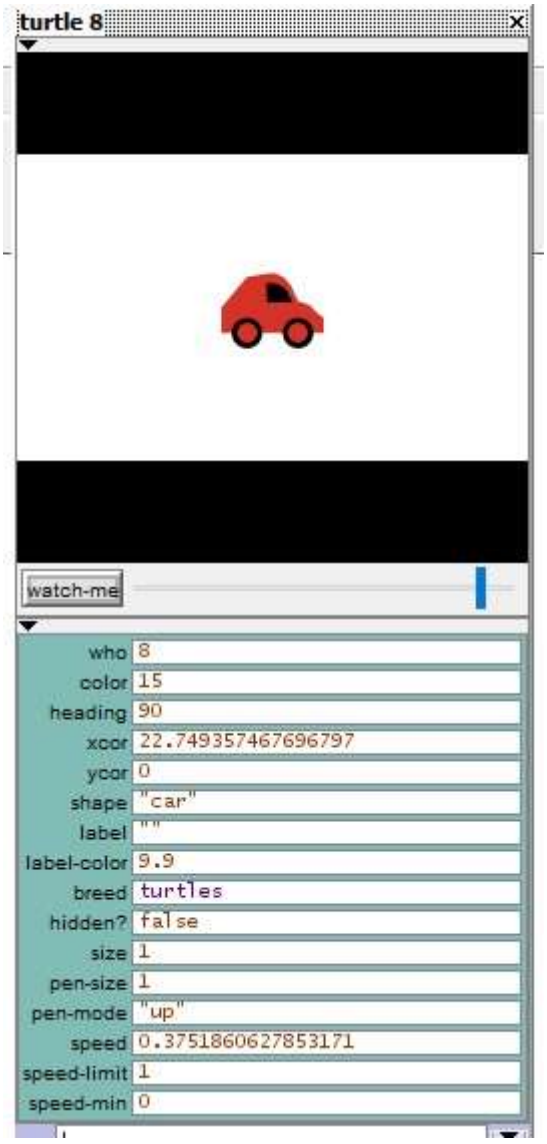
Exemple de comenzi pentru **setarea şi schimbarea culorilor, mărimilor, formelor**:

- observer> ask patches [set pcolor orange]
- observer> ask turtles [set color black]
- patches> set pcolor yellow
- turtles> set color white
- turtles> set color red – 2 (o nuantă mai inchisă de rosu)
- turtles> set color red + 2 (o nuantă mai deschisă de rosu)
- observer> ask turtles [set size 2]

- turtles> set shape “flower”
- turtles> set shape “airplane”

- Catalog culori NetLogo:
  http://ccl.northwestern.edu/netlogo/docs/programming.html#colors

- Lista de forme disponibile si posibilitatea customizarii unor forme de agenti se gaseste in meniul *Tools / Turtle Shapes Editor*



• **Provocare:**

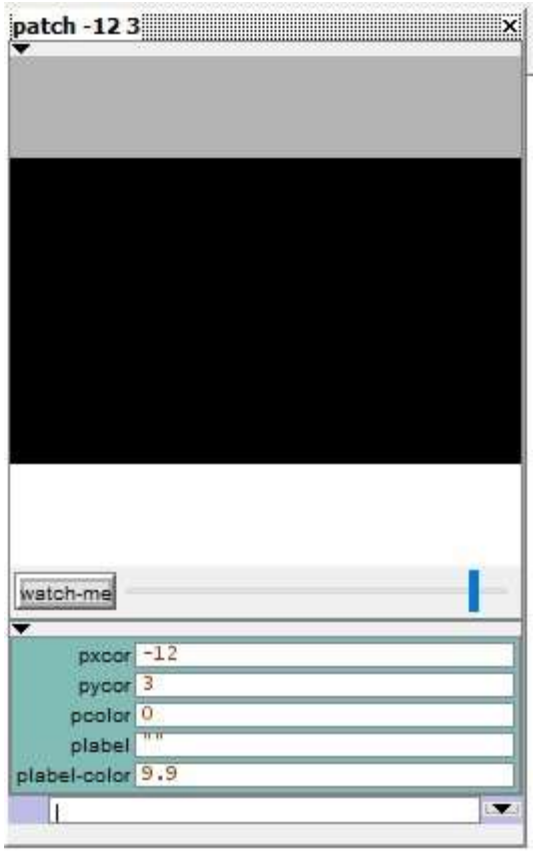
- Schimbati forma tuturor agentilor folosind Command Center
- Schimbati marimea tuturor agentilor folosind Command Center
- Descoperiti opţiunea label şi alte opţiuni

Modificarea unor setari în fereastra «Agent Monitors » şi « Agent Commanders »

- Click dreapta pe unul dintre agenţii de tip “turtle” , alege “inspect turtle”

Modificarea unor setari în fereastra «Patch Monitors » şi « Patch Commanders »

- Click dreapta pe unul dintre pătratele care formează mediul, alege de exemplu “inspect patch 12-3”



- Opţiunea *plabel* – permite afisarea unui text ▪
  Exemplu: înlocuieşte " " cu "Sunt grabit!" . - - -

Opţiunea *plabel-color* – permite colorarea textului afisat cu optiunea *plabel*

-

- **Provocare:** descoperiti celelate opţiuni

Provocare: reveniţi la fereastra "Command Center" si executaţi diverse comenzi, folosind elemente de limbaj descoperite anterior ("inspect turtle", "inspect patch")

Studierea codului sursa a modelului *Traffic Basic*

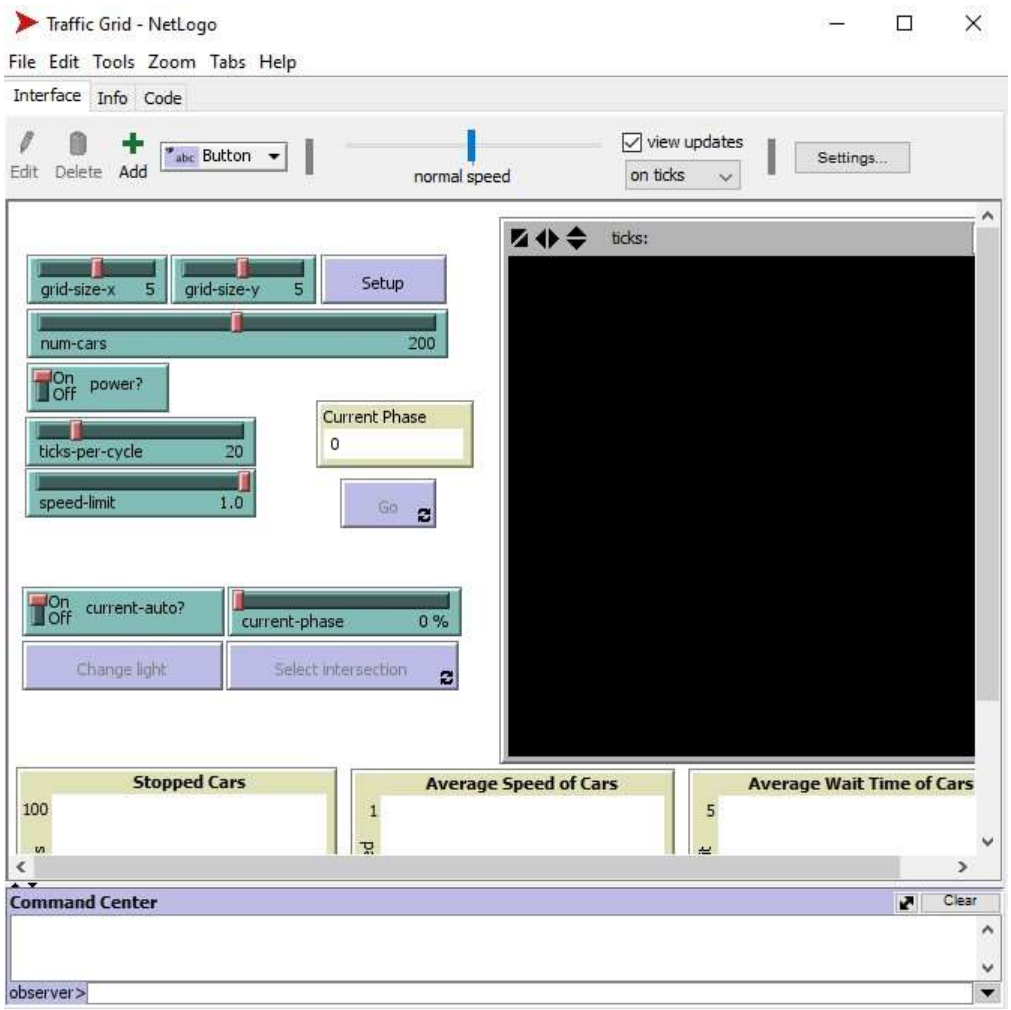
- partea de cod care surprinde regula de incetinire sau accelerare este:

```
ifelse car-ahead != nobody
 [ slow-down-car car-ahead ]
 [ speed-up-car ]
```

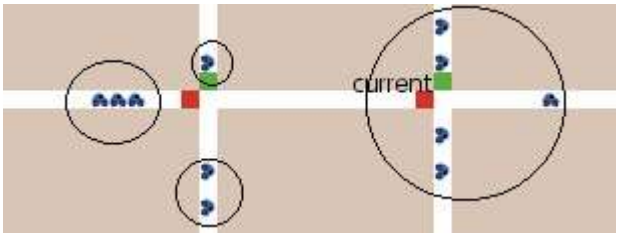
- daca exista o masina in fata, atunci se executa [slow-down-car car-ahead] , adica masina incetineste; altfel se executa [speed-up-car] , masina accelerand.

Modelul NetLogo *Traffic Grid*

Se accesează modelul **«Traffic Grid »** din Models Library \ Sample Models\Social Science. Caracteristicile agenţilor de a fi capabili să se poziţioneze în raport cu mediul, autonomi, raţionali, responsivi, sociali se pun în evidenţă cu ajutorul acestui model *NetLogo*.

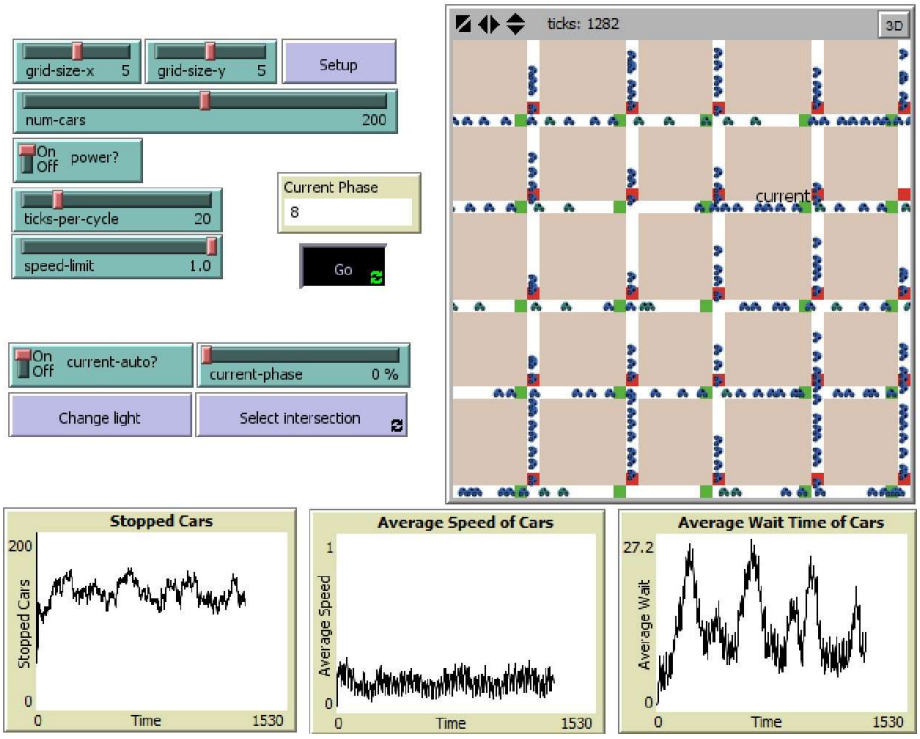


- Agenții sunt reprezentați de către automobile (mașini), similar modelului *TrafficBasic*.



- Privind comportamentul agenților, observăm că:

- dacă viteza curentă este mai mică decât limita de viteză (*speed-limit = 1.0 comform setarilor initiale*) și nu există o mașină în fața lor, masinile accelerează;
- dacă în fața lor se află o mașină mai lentă, agentii își reglează viteza la acelasi nivel cu cel al mașinii mai lente, franand;
- dacă în fața lor se află o lumină roșie (semafor)sau o mașină oprită, agentii se opresc.

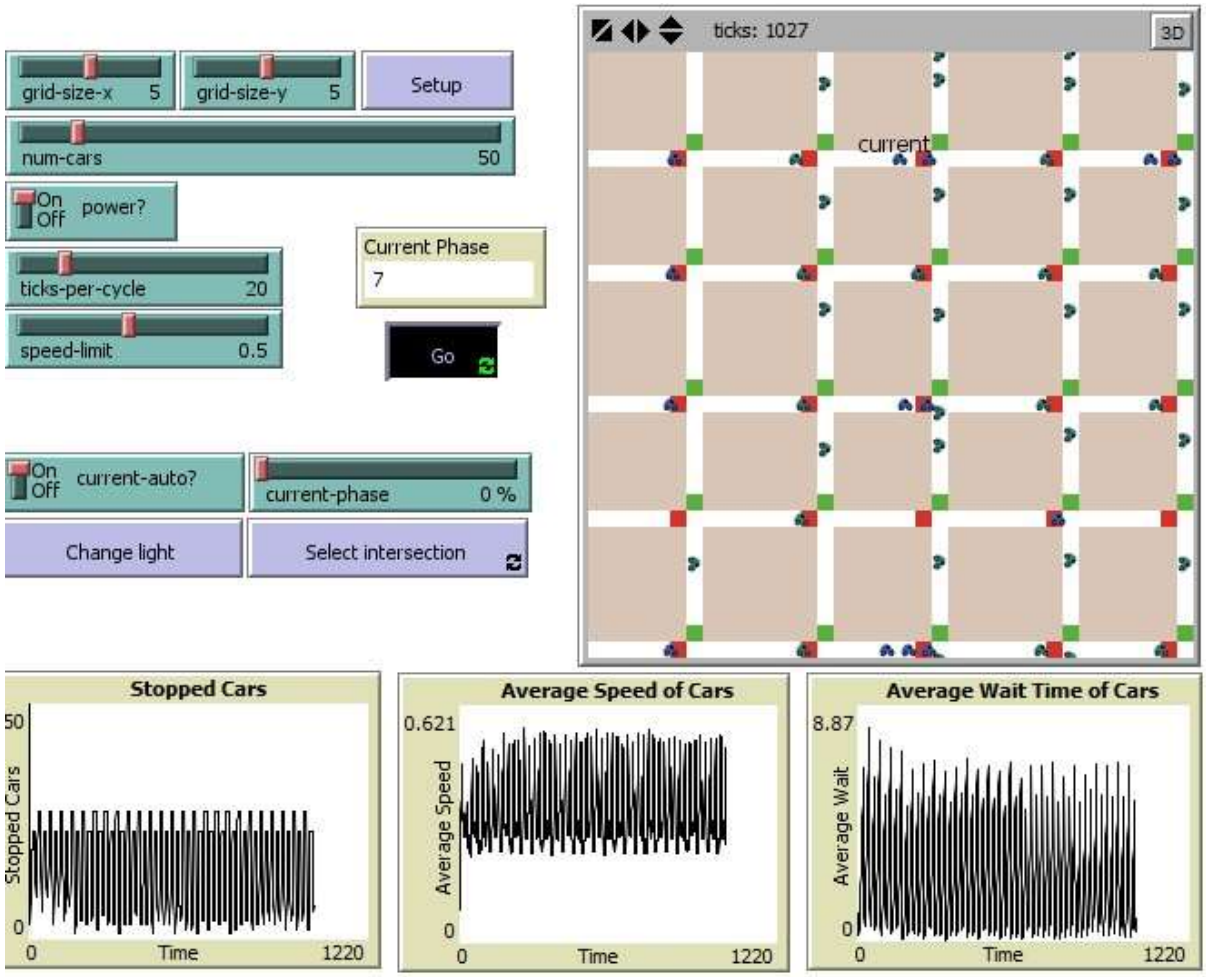


Setarile implicite ale modelului:

- Limita de viteza: *Speed-limit = 1.0*
- Numar de masini*: Num-cars = 50*
- Numar de strazi pozitionate vertical si orizontal: *Grid-size-x = Grid-size-y = 5* de fiecare tip
- Asteptare la semafor: *ticks-per-cycle: 20*

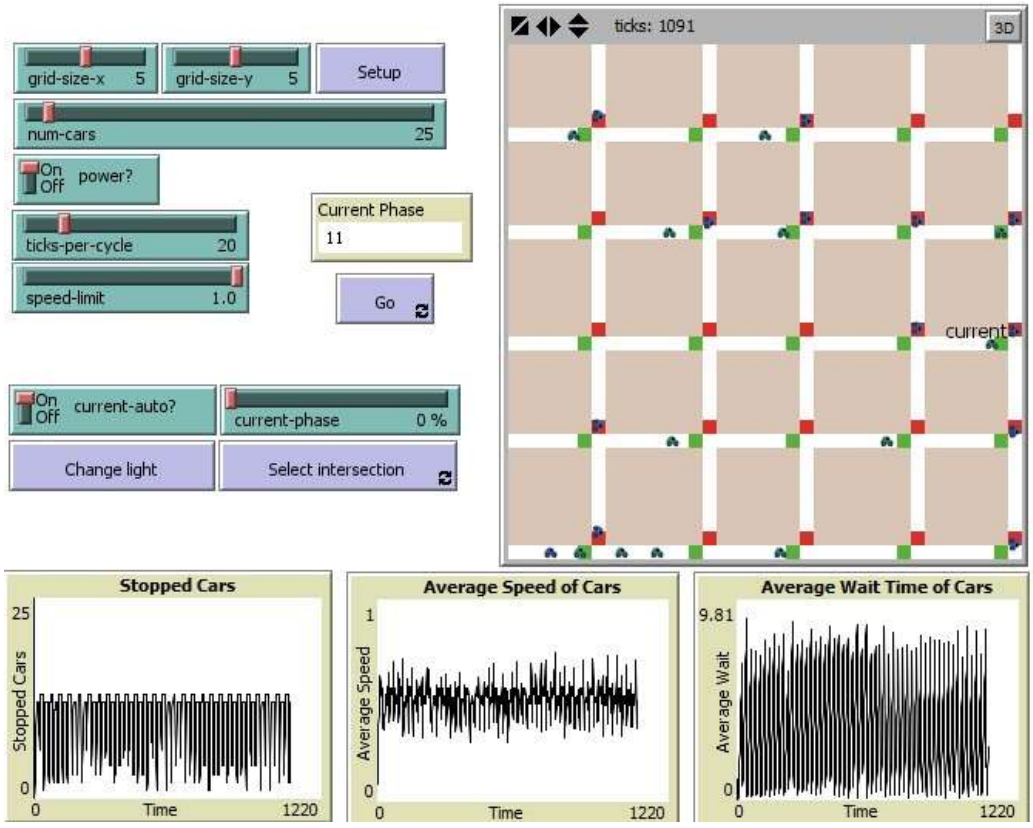
Modificarea limitei de viteză

Dacă nu este adusă nicio modificare altor setari, cu excepția limitei de viteză (speedlimit devine 0,5), **observăm că reducând limita de viteză, fluiditatea traficului creste**.



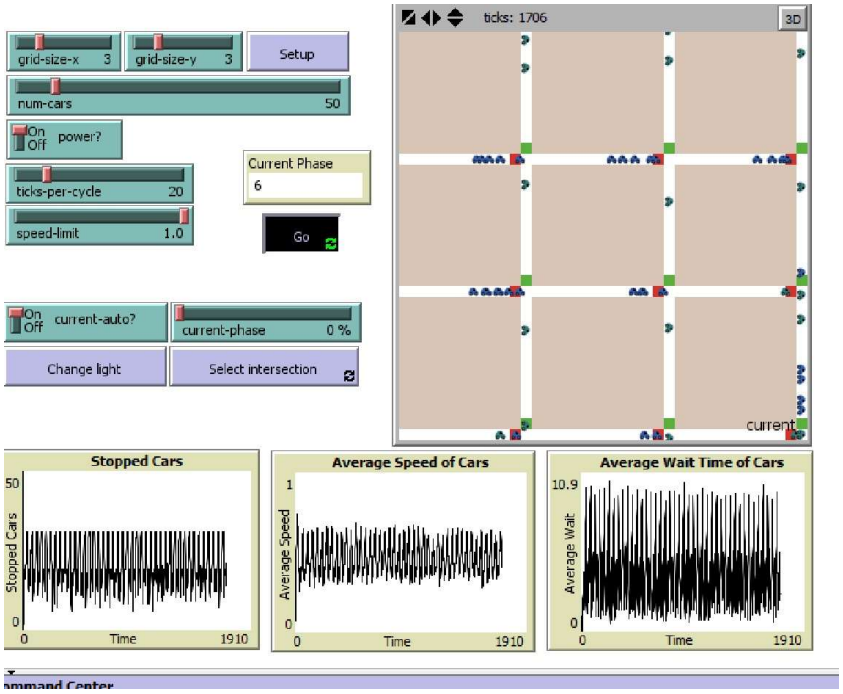
- *Modificarea numărului de agenți*

Dacă păstrăm setarile implicite ale modelului, cu excepția numărului de agenți care scade (*num-cars* mai mic), observăm că nivelul de aglomerare de pe strazi este mai redus. Iar **la o creștere a numarului de mașini, efectul este invers, de aglomerare**.



- *Modificarea numărului de străzi.*

Observăm că daca reducem numărul de străzi din cadrul modelului (grid-size), traficul se ingreuneaza, numărul mediu de maşini oprite creste, la fel timpul mediu de aşteptare. Iar **la o creştere a numarului de străzi pe care agenţii se pot deplasa, efectul este invers, de fluidizare a traficului**.



- Studierea codului sursă a modelului

Secțiunea de cod a modelului răspunzătoare pentru oprirea la semafor a mașinilor, pentru comportamentul de accelerare sau franare este următoarea:

```
to set-car-speed
ifelse pcolor = red
  [ set speed 0 ]
  [
    ifelse up-car?     [
set-speed 0 -1 ]
    [ set-speed 1 0 ]
  ]
End
```

Aşadar, dacă patch-ul cu care se intalneste maşina (agentul) are culoarea roşie (*ifelse pcolor = red*), atunci viteza automobilului va fi setată la 0 (*set speed 0*). Altfel, adică în cazul în care patch-ul imediat următor din fața mașinii nu este roșu (ramura else a structurii if), se va verifica dacă există vreo mașină în fața agentului (*ifelse up-car?*). În cazul în care există, mașina va frâna, își va ajusta viteza (*set-speed 0 -1*), altfel mașina agent va accelera, își va continua drumul [*set-speed 1 0*].

Conceptele cheie utilizate în definirea sistemelor adaptive complexe

Conceptele cheie utilizate în definirea sistemelor adaptive complexe sunt:

- ***sistemul multiagent*** - un ansamblu de agenți, interconectați între ei şi cu mediul inconjurator, pentru atingerea unui scop comun;
- ***complexitatea*** - o multitudine de agenți şi de legături între ei şi cu mediul înconjurător, care au libertatea de a acţiona în moduri care nu sunt total predictibile, în care agenții desfăşoară acţiuni interconectate, astfel încât acţiunile unui agent schimbă contextul pentru alţi agenţi;
- ***comportamentul adaptiv*** - restructurarea sistemului sub influența factorilor perturbatori, sistemul fiind capabil să îşi creeze o nouă structură, fără intervenții din mediul extern.

Modelul NetLogo *Wolf Sheep Predation*

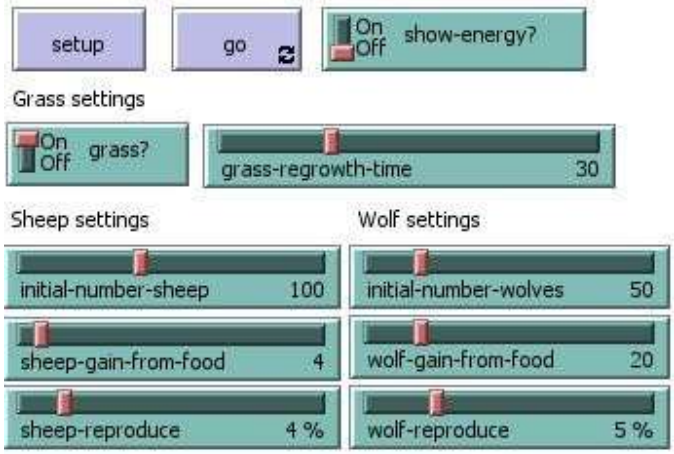
Modelul *Wolf Sheep Predation* exemplifică sistemele adaptive complexe. Modelul se accesează din librăria programului NetLogo: Models Library \ Sample Models \ Biology \ Wolf Sheep Predation.

În cadrul modelului putem observa un **sistem de tip multiagent**: un **ansamblu de agenţi** de tip lupi, conectaţi cu agenţi de tip oi, conectaţi cu utilizatorul, dar şi cu mediul înconjurător, mediu care poate oferi hrană de tip iarbă agenţilor oi. **Scopul comun** al sistemului este atingerea unei stări de stabilitate în ecosistemul pradă – prădător, care să se menţină în timp: lupii se hrănesc cu oile, cu scopul de a nu duce la extincţia acestei populaţii, oile se hrănesc cu iarbă, cu scopul de a nu consuma până la dispariţia totală a resursei din mediu. Creșterile și scăderile dimensiunilor fiecărei populații sunt **interconectate,** după cum se observă din execuţia modelului.

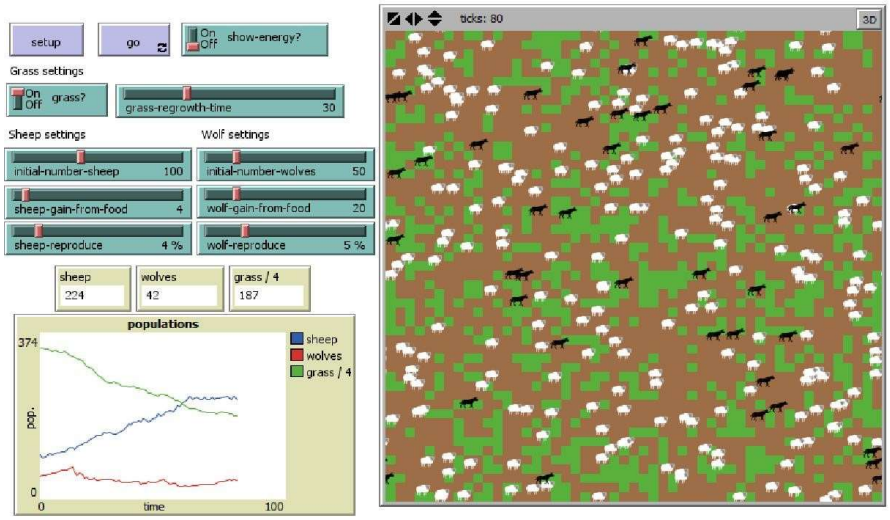
Complexitatea este pusă în evidenţă prin **numărul mare de agenţi** care interacţionează: iniţial modelul are 100 de agenţi oi şi 50 de agenţi lupi (utilizatorul poate creşte complexitatea, mărind numărul de agenţi din model). Rezultatul acţiunilor agenţilor de a se hrăni, de a se deplasa prin mediu, de a se reproduce este **impredictibil**: utilizatorul nu va şti cu exactitate dacă numărul de agenţi de un anumit tip va creşte, va scădea sau va rămâne constant, nici dacă resursa din mediu va fi suficientă sau nu. Remarcăm că **acţiunile unui agent schimbă contextul pentru alţi agenţi**: de exemplu dacă utilizatorul va creşte numărul de agenţi de un anumit tip, iar pentru celălalt tip va scădea dimensiunea sau o va păstra constantă, una dintre populaţii va dispărea, ceea ce va face întreg sistemul instabil. O schimbare de context pentru agenţi se observă în cazul modificării oricărei setări a modelului:

- INITIAL-NUMBER-SHEEP: Mărimea inițială a populației de oi.
- INITIAL-NUMBER-WOLVES: Mărimea inițială a populației de lupi.
- SHEEP-GAIN-FROM-FOOD: Energia pe care un agent oaie o primește atunci când mănâncă iarbă.
- WOLF-GAIN-FROM-FOOD: Energia pe care un agent lup o primește atunci când mănâncă o oaie.
- SHEEP-REPRODUCE: Probabilitatea ca o oaie să se reproducă.

- WOLF-REPRODUCE: Probabilitatea ca un lup să se reproducă.
- GRASS-REGROWTH-TIME: parametru pentru regenerarea ierbii.
- GRASS?: Dacă este inclusă sau nu resursa iarbă în model.
- SHOW-ENERGY?: Dacă este inclusă sau nu energia în model.



Comportamentul adaptiv este vizibil în model sub influenţa factorilor perturbatori cum ar fi scăderea probabilităţii de reproducere a agenţilor sau scăderea drastică a numărului de agenţi. Acţiunile agenţilor de a se hrăni, de a se deplasa prin mediu, sau de a se reproduce se vor desfăşura în sistemul restructurat.



Există două variante de execuţie ale acestui model:

- În prima variantă, lupii și oile se mișcă în mod aleatoriu în mediul definit, în timp ce lupii caută oile pentru a se hrăni cu acestea. Resursa iarbă nu este prezentă. Energia lupillor este consumată cu fiecare pas și ei trebuie să mănânce oile pentru a-și reîmprospăta energia. Când nu mai au energie, ei mor. Pentru a permite populației să continue, fiecare lup sau oaie are o probabilitate dată de a se

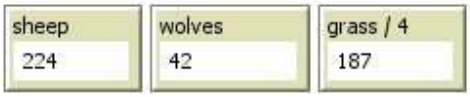
reproduce. Această variant de execuţie generează o dinamică interesantă a populației, dar este destul de instabilă.

- Cea de-a doua variantă include iarba (verde) în execuţia modelului, în plus față de agenţii lupi și agenţii oi. Comportamentul lupilor este identic cu prima variantă, dar de această dată oile trebuie să mănânce iarbă pentru a-și menține energia, iar când nu mai au energie, mor. Odată ce iarba este mâncată, aceasta se va regenera după o perioadă dată de timp. Această variantă de model este mai complexă decât prima şi duce, în general, la rezultate stabile privind dimensiunile populaţiilor şi resursei din mediu.

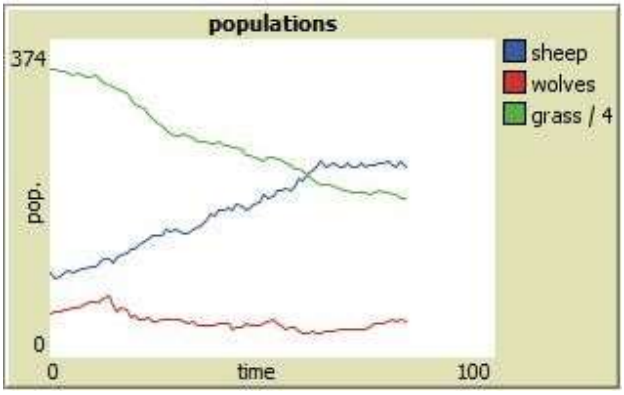
Observații:

- pentru fiecare pas pe care îl face un lup, i se deduce o unitate de energie
- când este inclusă iarba, se deduce o unitate de energie pentru fiecare pas pe care îl face o oaie

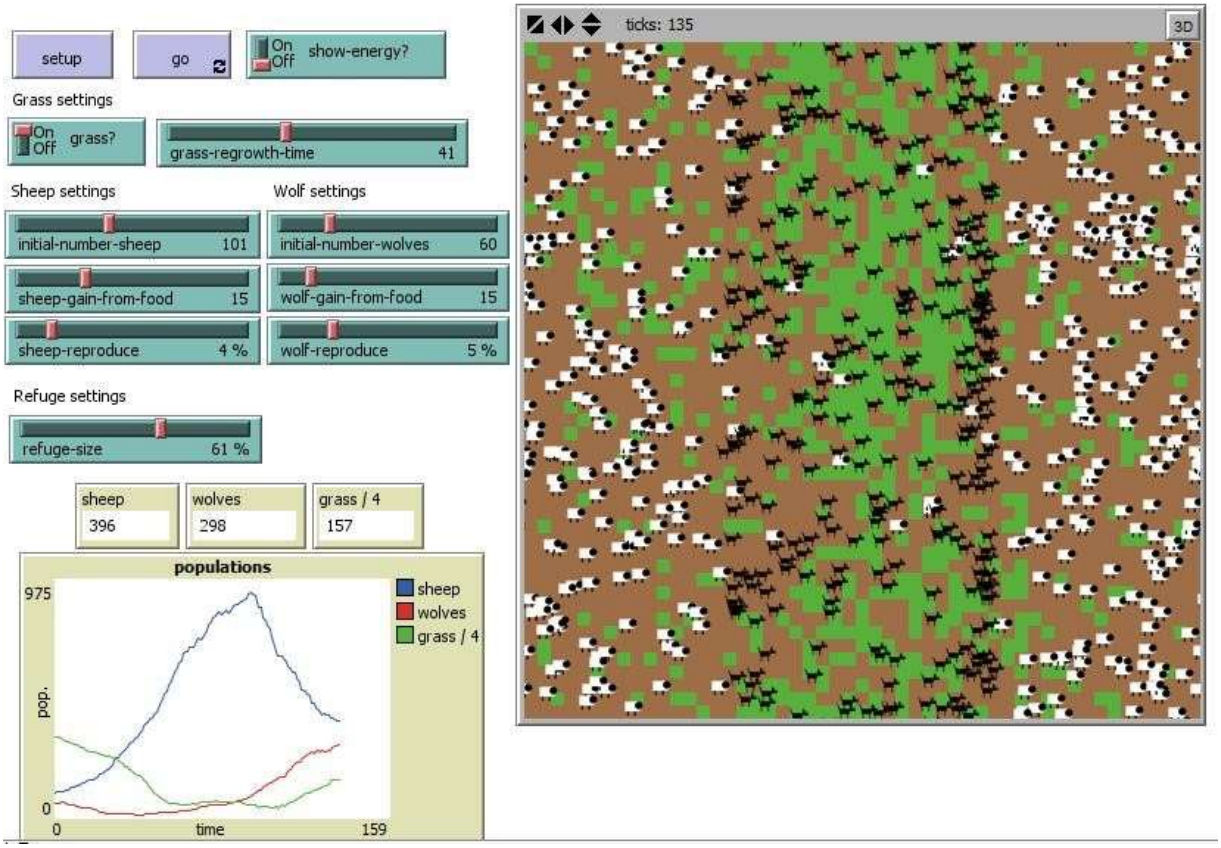
Interfaţa conţine monitoare pentru a vedea mărimile populaţiilor la un moment dat:



Graficul inclus pe interfaţa permite vizualizarea fluctuaţiilor speciilor: oi, lupi, resursă iarbă.



EXTENSIE LA MODELUL DE BAZĂ. Wolf Sheep Predation Refuge



Acest model *NetLogo* este similar cu cel anterior prezentat, iar în plus introduce o zonă de refugiu pentru pradă (în cazul nostru un refugiu pentru agenţii oi). În această zonă de refugiu nu este permis accesul prădătorilor, în cazul nostru, lupii.

Prezența unui refugiu definit de utilizator face posibilă studierea stabilității ecosistemului pradă-prădător în prezența unei zone unde nu sunt permiși prădătorii.

Oile se pot mișca aleatoriu oriunde în mediu. Lupii se mișcă, de asemenea, aleatoriu, dar nu au acces în interiorul zonei de refugiu.

Această zonă de refugiu poate fi modificată în mărime de către utilizator. Modificând dimensiunea refugiului, se vede cum afectează dinamica relațiilor dintre prădători și pradă, supraviețuirea speciilor în mediu.

Pentru implementarea extensiei de model, se adaugă pe interfaţă un slider denumit “refuge-size” şi se utilizează codul sursă de mai jos.

COD SURSĂ Wolf Sheep Predation Refuge *

** Porţiunile suplimentare de cod sunt evidenţiate cu galben.*

```
;; Sheep and wolves are both breeds of turtle.
breed [sheep a-sheep]  ;; sheep is its own plural, so we use "a-sheep" as the singular. sheep
reproduce and give a sheep breed [wolves wolf] ;; wolves reproduce and give a wolf turtles-own
[energy]       ;; both wolves and sheep have energy patches-own [countdown]


to setup
  ;; (for this model to work with NetLogo's new plotting features,
  ;; __clear-all-and-reset-ticks should be replaced with clear-all at
  ;; the beginning of your setup procedure and reset-ticks at the end
  ;; of the procedure.)   __clear-all-
and-reset-ticks   ask patches [ set
pcolor green ]   ;; check GRASS?
switch.
  ;; if it is true, then grass grows and the sheep eat it   ;; if it false, then the sheep don't need to
eat   if grass? [     ask patches [       set countdown random grass-regrowth-time ;; initialize grass
grow clocks randomly       set pcolor one-of [green brown]
    ]
  ]
  set-default-shape sheep "sheep"    create-sheep initial-number-sheep  ;; create the sheep,
then initialize their variables   [
    set color white     set size 1.5  ;; easier to see     set label-color blue - 2     set energy
random (2 * sheep-gain-from-food) ;; sheep gain energy from grass     setxy random-
xcor random-ycor
  ]
  set-default-shape wolves "wolf"    create-wolves initial-number-wolves  ;; create the wolves,
then initialize their variables
  [
    set color black     set size 1.5  ;; easier to see     set energy random (2 * wolf-gain-from-food) ;;
wolf gain energy from sheep     setxy 0 - xcor 0 - ycor ;; wolf begin in the center of the world, being
that the center is 0;0
```

```
 ]
 display-labels
update-plot end

to go  if not any? turtles [ stop ]  ask sheep [    move-sheep ;; sheep move randomly
through the entire world    if grass? [      set energy energy - 1  ;; deduct energy for
sheep only if grass? switch is on      eat-grass ;; sheep gain energy from grass
  ]
  reproduce-sheep ;; sheep reproduce when energy permits    death ;;
sheep die
 ]
 ask wolves [
  move-wolf ;; wolf movement is limited    set energy energy - 1  ;;
wolves lose energy as they move    catch-sheep ;; wolves gain
energy from sheep    reproduce-wolves ;; wolves reproduce when
energy permits    death ;; wolves die
 ]
 if grass? [ ask patches [ grow-grass ] ]  tick
update-plot  display-labels end

to move-sheep  ;; sheep move randomly through world  rt
random-float 50 - random-float 50   fd 1 end

to move-wolf  ;; wolves move randomly through world    rt
random-float 50 - random-float 50
 if abs (xcor + dx) > max-pxcor - (max-pxcor * refuge-size / 100) [move-wolf] ;; wolf rotates
randomly
 ;; if by moving forward one step wolf reaches refuge it rotates again  fd 1
end

to eat-grass  ;; sheep procedure  ;; sheep eat grass, turn the patch brown  if
pcolor = green [    set pcolor brown    set energy energy + sheep-gain-from-food  ;;
sheep gain energy by eating
```

```
 ]
end

to reproduce-sheep  ;; sheep procedure   if random-float 100 < sheep-reproduce [  ;; throw
"dice" to see if you will reproduce     set energy (energy / 2)               ;; divide energy between
parent and offspring     hatch 1 [ rt random-float 360 fd 1 ]   ;; hatch an offspring and move it
forward 1 step
 ]
end

to reproduce-wolves  ;; wolf procedure   if random-float 100 < wolf-reproduce [  ;; throw
"dice" to see if you will reproduce     set energy (energy / 2)               ;; divide energy between
parent and offspring     hatch 1 [ rt random-float 360 fd 1 ]  ;; hatch an offspring and move it
forward 1 step
 ]
end

to catch-sheep  ;; wolf procedure   let prey one-of sheep-here
;; grab a random sheep   if prey != nobody                          ;; did we
get one?  if so,
   [ ask prey [ die ]                    ;; kill it
     set energy energy + wolf-gain-from-food ] ;; get energy from eating end

to death  ;; turtle procedure   ;; when
energy dips below zero, die   if energy < 0
[ die ] end

to grow-grass  ;; patch procedure
  ;; countdown on brown patches: if reach 0, grow some grass   if
pcolor = brown [
   ifelse countdown <= 0       [ set pcolor green
set countdown grass-regrowth-time ]
```

```
    [ set countdown countdown - 1 ]
  ]
end

to update-plot  set-current-plot "populations"  set-current-plot-pen "sheep"  plot count sheep
set-current-plot-pen "wolves"  plot count wolves  if grass? [  set-current-plot-pen "grass / 4"
plot count patches with [pcolor = green] / 4  ;; divide by four to keep it within similar
                                  ;; range as wolf and sheep populations
  ]
end

to display-labels  ask turtles [ set label "" ]  if show-
energy? [  ask wolves [ set label round energy ]  if
grass? [ ask sheep [ set label round energy ] ]
  ]
end
```

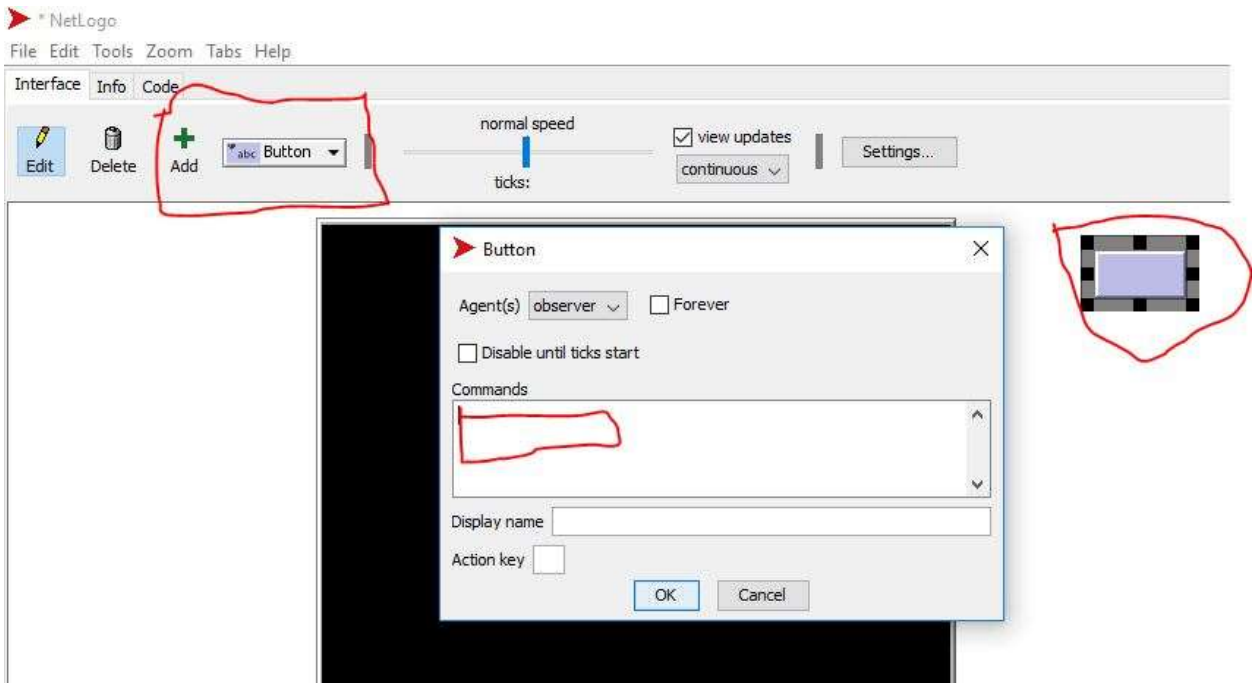
CREAREA UNUI NOU MODEL ÎN NET LOGO

Un model NetLogo este constituit dintr-o serie de proceduri detaliate în codul sursă şi o serie de butoane adăugate pe interfaţa modelului.

În *NetLogo*, o procedură combină o serie de comenzi ale limbajului *NetLogo* într-o nouă comandă gata de execuţie. De exemplu o nouă procedură poate face ca un agent de tip “turtle” să se dezvolte, să se mişte, să se reproducă, să dispară, etc

CREAREA BUTOANELOR ŞI PROCEDURILOR “SETUP”, “GO”

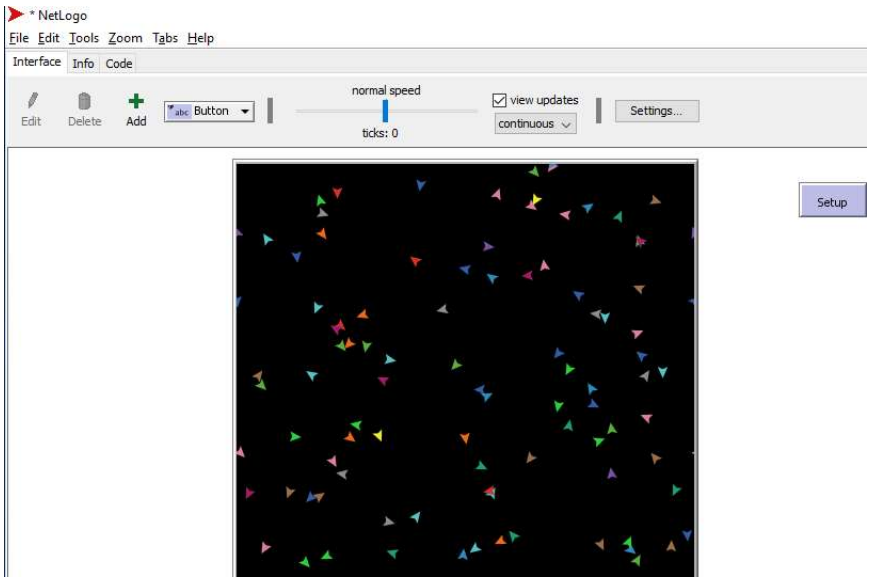
- Crearea butonului “SETUP”
  - Din interfaţa NetLogo se alege adăugarea unui nou buton
  - Cu un click în zona albă a ecranului se alege locaţia butonului
  - In «popup» se completează un minim de informaţii (numele butonului creat) şi se pot schimba alte opţiuni faţă de cele «default »
  - Butonul creat rămâne roşu până când utilizatorul va asocia o procedură respectivului buton



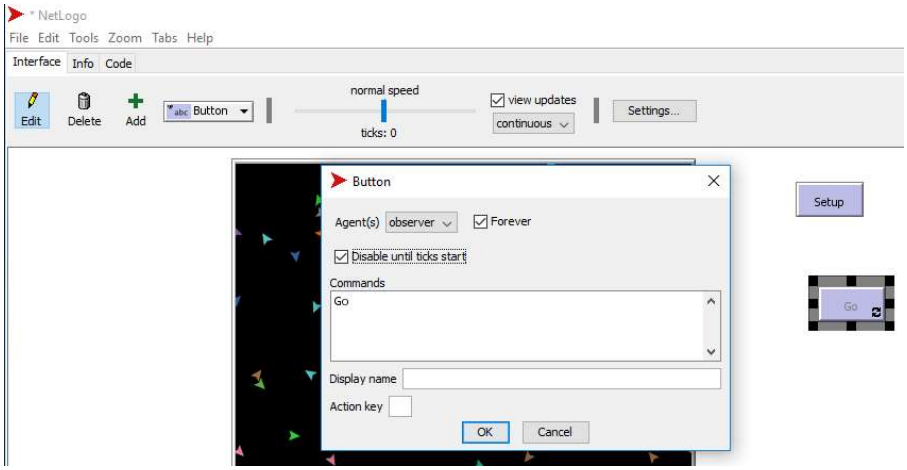
- Crearea procedurii “SETUP”
  - Se navighează în al 3-lea tab al interfeţei (tabul “Code”)
  - Codul procedurii “Setup” poate arăta astfel:

```
to setup
  clear-all
  create-turtles 100 [ setxy random-xcor random-ycor ]
  reset-ticks
end
```

  - În exemplul prezentat, se crează 100 de agenţi de tip “turtle”, având o poziţie întâmplătoare atât pe axa orizontală cât şi pe cea verticală
  - O procedură întotdeauna începe cu “to” şi se termină cu “end”
  - Pentru executarea procedurii create şi vizualizarea rezultatelor, este suficientă navigarea în fereastra “Interface” şi apăsarea butonului creat anterior (“Setup”)



- Crearea butonului “Go”
  - Butonul “Go” se crează similar cu butonul “Setup”
  - în exemplul prezentat se vor schimba unele opţiuni implicite, astfel:
    - Se va alege opţiunea “Forever”
    - Se va alege « Disable until ticks start » întrucât previne utilizarea butonului înainte de finalizarea procedurii “Setup”



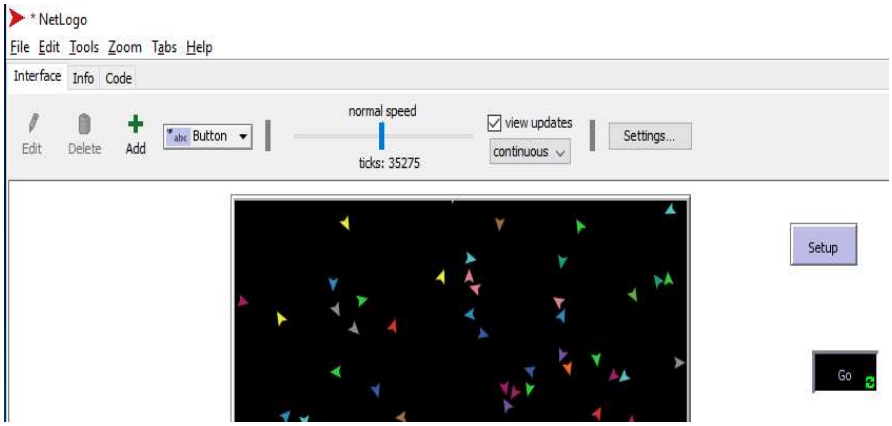
- Crearea procedurii “Go”
  - Se navighează în al 3-lea tab al interfeţei (tabul “Code”)
  - Codul procedurii “Go” poate arăta astfel:

```
to go
  move-turtles
  tick
end

to move-turtles
  ask turtles [
    right random 360
```

```
    forward 1
  ]
end
```

- În exemplul prezentat:
  - se apelează o nouă procedură, «move-turtles » în interiorul procedurii « Go »
  - cei 100 de agenţi de tip “turtle” creaţi anterior se deplasează aletoriu, după un unghi < 360^0, distanţa parcursă fiind tot timpul de un pas



Provocare 1:

Creaţi 3 agenţi care se deplasează aletoriu, după un unghi < 45^0, distanţa parcursă fiind tot timpul de 10 paşi

Provocare 2:

Schimbaţi culoarea agenţilor în alb şi modificaţi culoarea mediului în albastru, folosind instrucţiuni executate în fereastra *Command Center*

Provocare 3:

Modificaţi codul sursă astfel încât în cadrul procedurii “Setup” să se apeleze procedurile “Setup-patches” şi “Setup-turtles”, mediul să fie de culoare verde , iar agenţii să fie albi, în număr de 100, având o poziţie întâmplătoare în spaţiu

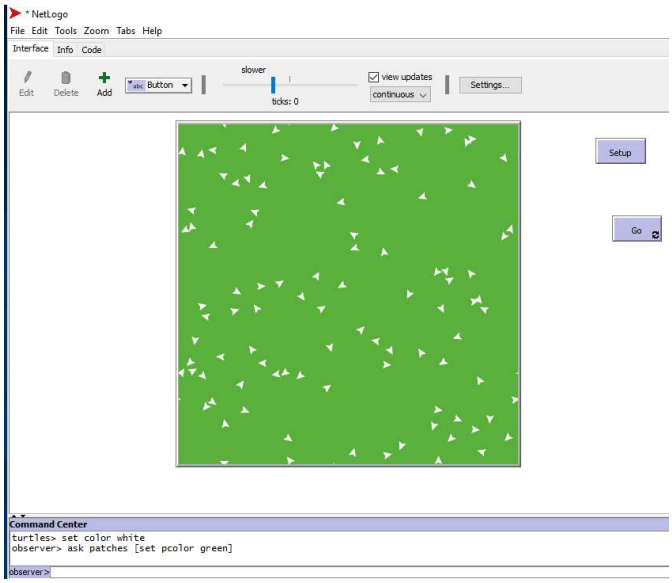
- *Răspuns provocarea 1 : se utilizează sintaxele « create-turtles 3 “, «right random 45” şi “forward 10” pentru a vizualiza schimbarea în rezultate. Este nevoie de o nouă apăsare a butonului “Setup” înainte de «Go »*
- *Răspuns provocarea 2 : din fereastra “Command Center”: turtles> set color white & observer> ask patches [set pcolor blue]*

- *Răspuns provocarea 3 :*

```
to setup
  clear-all
  setup-patches
  setup-turtles
  reset-ticks
end
```

```
to setup-patches
  ask patches [ set pcolor green ]
end

to setup-turtles
  create-turtles 100
  ask turtles [ setxy random-xcor random-ycor ]
  ask turtles [set color white]
end
```



CREAREA INTERACŢIUNILOR ÎNTRE AGENŢI. DEFINIREA UNOR COMPORTAMENTE

- Agenţii se hrănesc din mediu şi câştigă energie astfel
  - Definim la începutul codului sursă o variabilă asociată agenţilor de tip « turtle” pentru a ţine evidenţa nivelului de energie pe care fiecare agent o deţine la un moment de timp :«turtles-own [energy] »
  - Adăugăm procedura “eat-grass” în cadrul procedurii “go”
  - Agenţii de tip «turtle » se vor hrăni din mediu, câştigând astfel energie. Aceştia vor mânca agenţii de tip «patches». In exemplul prezentat câştigă 10 unităţi de energie la fiecare « patch » consumat

```
to go
```

```
  move-turtles
  eat-grass
  tick
end

to eat-grass
          ask turtles [
            if pcolor = green [
              set pcolor black
              set energy energy + 10
            ]
          ]
        end
```

Provocare 4: **Realizaţi modificările necesare pentru a vizualiza rezultatle modelului pas cu pas.**

Răspuns provocarea 4 : Interface -> Edit «Go » button -> Uncheck « Forever »

- Agenţii pierd energie la fiecare pas de mişcare efectuat (de exemplu o unitate pierdută la fiecare pas efectuat) şi dispar când nu mai au energie
    - rescrierem procedura « move-turtles” astfel:

```
to move-turtles
  ask turtles [
    right random 360
    forward 1
    set energy energy - 1
  ]
end
```

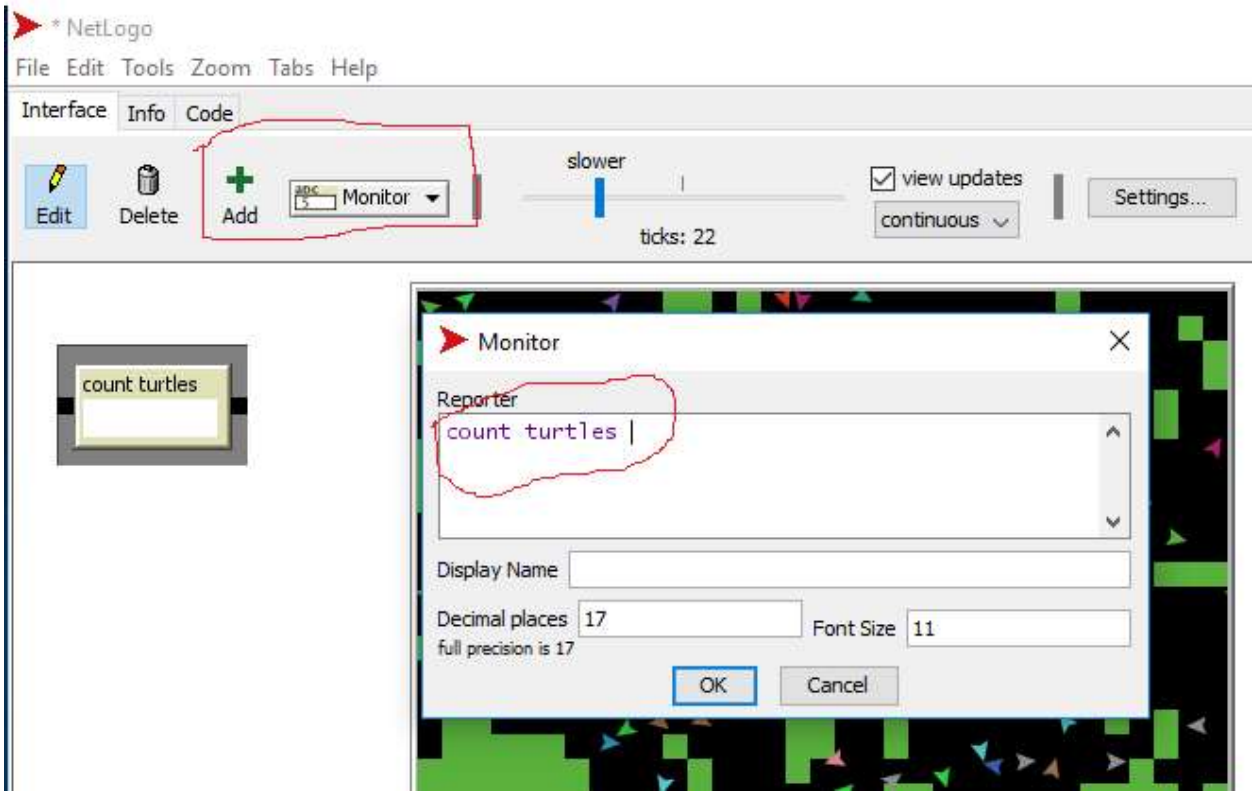
    - introducem procedura «check-death » în cadrul procedurii «Go »

```
to check-death
  ask turtles [
```

```
    if energy <= 0 [ die ]
  ]
end
```

ADĂUGAREA UNOR BUTOANE SI GRAFICE PE INTERFATA PENTRU MONITORIZAREA EVOLUŢIEI AGENŢILOR

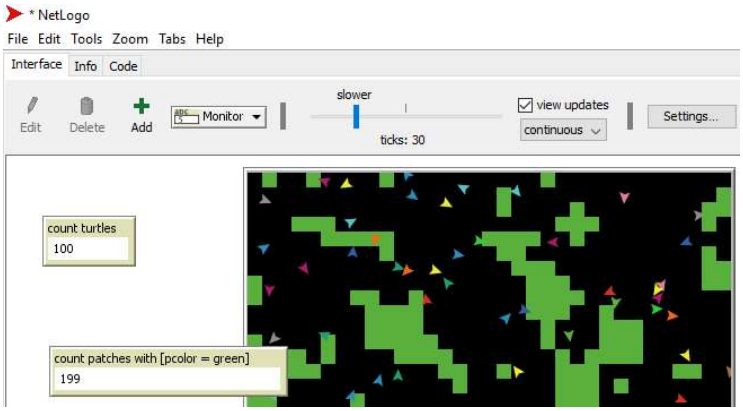
- monitorizarea numărului de agenţi de tip « turtle »
  - din interfaţă se alege “Add Monitor”, iar printr-un click se stabileşte locaţia graficului pe ecran
  - adăugăm sintaxa «`count turtles` »



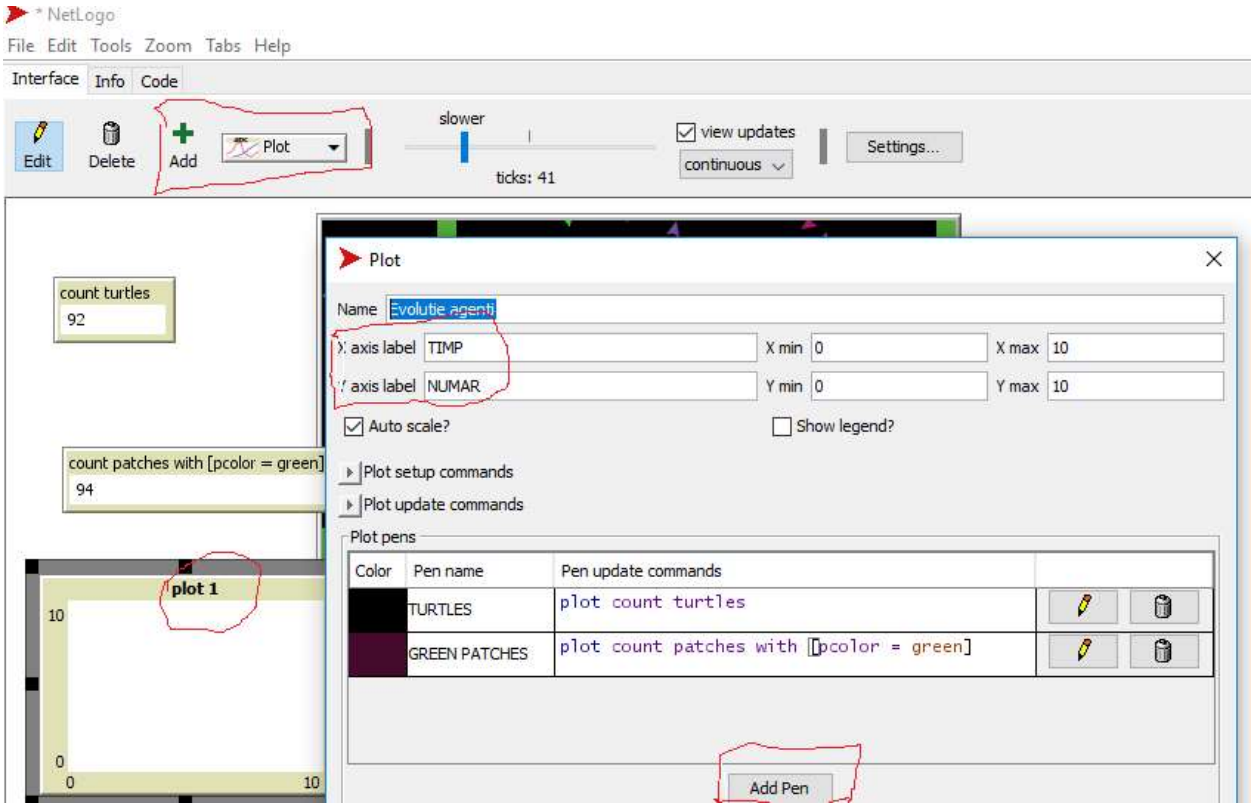
Provocare 5:

Adăugaţi un buton pentru monitorizarea numărului de agenţi de tip « patches » care au culoarea verde

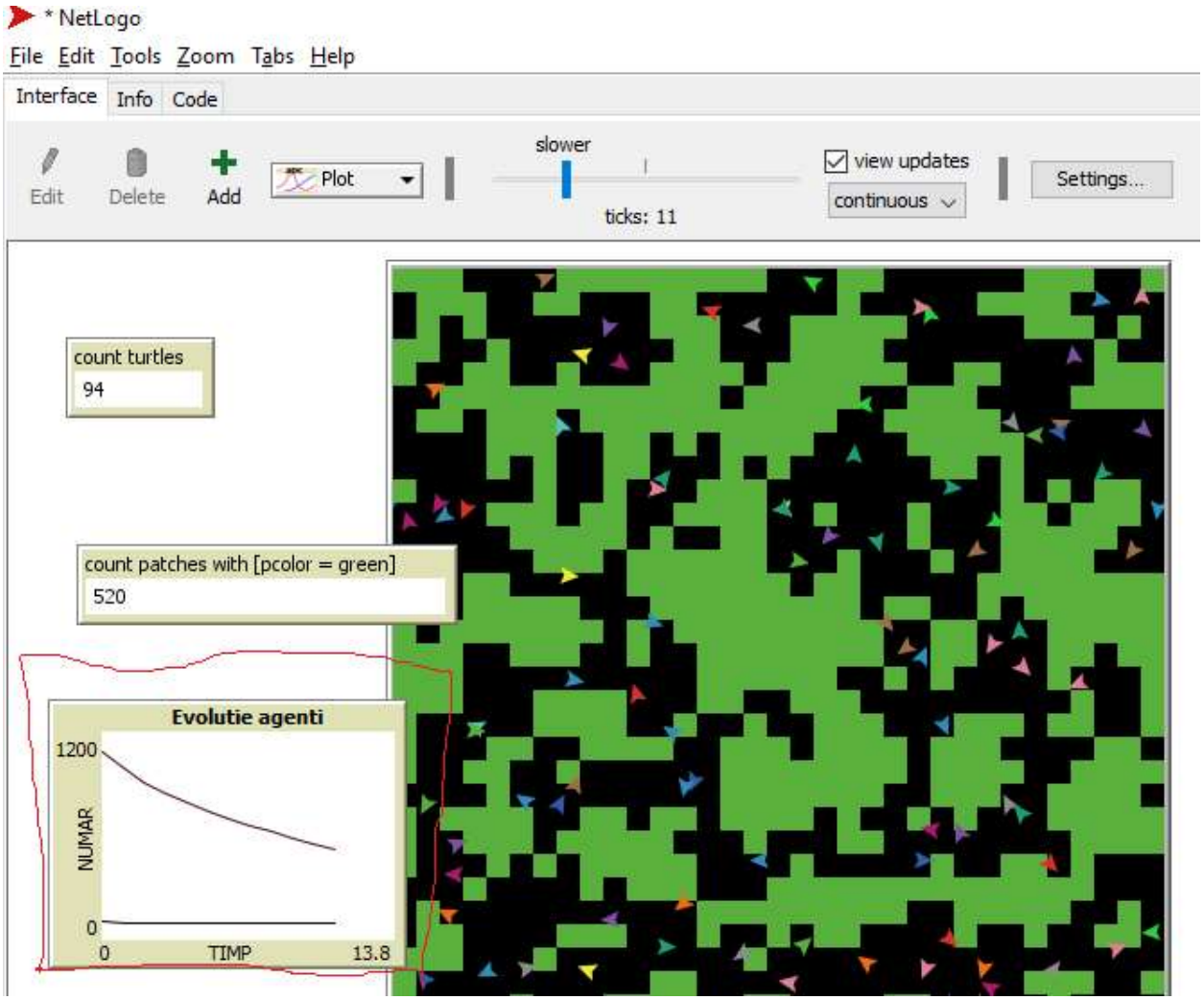
Răspuns provocarea 5: Interface -> Add Monitor -> sintaxa «`count patches with [pcolor = green]`»



- din interfaţă se alege “Add Plot”, iar printr-un click se stabileşte locaţia graficului pe ecran
- exemplu: pentru monitorizarea în acelaşi grafic a numărului de agenţi de tip « turtle » şi a numărului de agenţi de tip « patches » care au culoarea verde se utilizează sintaxele «plot `count turtles` » şi respectiv « `plot count patches with [pcolor = green]`»



- de exemplu după 13.8 secunde de execuţie a modelului, mai sunt 94 de agenţi «turtle » şi 520 de agenţi de tip « patches » care au culoarea verde



Temă seminar modelare bazată pe agenţi

Navigaţi în biblioteca NetLogo, alegeţi un model (altul decât cele parcuse la seminar) şi justificaţi elementele de sistem multiagent, complexitate şi comportement adaptiv din definiţia sistemelor adaptive complexe. Adăugaţi şi capturi de ecran relevante !

Alternativ, puteţi dezvolta modelul creat la seminar cu noi elemente de sintaxă, astfel încât noul model să aibă mai bine puse în evidenţă elementele de sistem multiagent, complexitate şi comportement adaptiv din definiţia sistemelor adaptive complexe. Adăugaţi codul sursă şi capturi de ecran relevante !

Bibliografie

- https://ccl.northwestern.edu/netlogo/models/index.cgi
- https://ccl.northwestern.edu/netlogo/docs/
- Net Logo \Help\Net Logo User Manual \Tutorial#3 Procedures