**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**Viện công nghệ thông tin và truyền thông**



**BÀI TẬP LỚN MÔN HỆ THỐNG TÁC TỬ**

***Đề tài :* Tìm hiểu về NetLogo và mô phỏng đàn cá bằng NetLogo**

**Hà Nội, 11 – 2012**

Mục lục

[I – Tìm hiểu về NetLogo: 3](#_Toc340437020)

[1. NetLogo là gì? 3](#_Toc340437021)

[2. Đặc điểm của NetLogo: 3](#_Toc340437022)

[II - Sử dụng NetLogo: 4](#_Toc340437023)

[1. Các mô hình : 4](#_Toc340437024)

[1.1. Điều khiển mô hình với các nút (buttons): 5](#_Toc340437025)

[1.2. Điều khiển tốc độ mô hình với Speed Slider: 5](#_Toc340437026)

[1.3. Điều chỉnh các cài đặt bằng Sliders và Switches : 6](#_Toc340437027)

[1.4. Thu thập thông tin bằng Plots và Monitors: 7](#_Toc340437028)

[1.5. Điều khiển phần View: 8](#_Toc340437029)

[2. Câu lệnh : 9](#_Toc340437030)

[2.1. Trung tâm câu lệnh : 10](#_Toc340437031)

[2.2. Điều khiển màu sắc : 10](#_Toc340437032)

[3. Hàm trong NetLogo: 12](#_Toc340437033)

[3.1. Tác tử và hàm,thủ tục: 12](#_Toc340437034)

[3.2. Thử nghiệm các câu lệnh: 17](#_Toc340437035)

[3.3. Patches và biến : 18](#_Toc340437036)

[3.4. Monitors : 21](#_Toc340437037)

[III – Mô hình mô phỏng đàn cá : 29](#_Toc340437038)

[1. Ý tưởng : 29](#_Toc340437039)

[2. Thiết kế: 29](#_Toc340437040)

[3. Lập trình: 30](#_Toc340437041)

# I – Tìm hiểu về NetLogo:

## NetLogo là gì?

NetLogo là một môi trường có thể lập trình được để mô phỏng tự nhiên và các hiện tượng xã hội. NetLogo được Uri Wilensky bắt đầu vào năm 1999 và tiếp tục phát triển tại Center for Connected Learning and Computer-Based Modeling.

NetLogo thích hợp để mô phỏng các hệ thống có sự phát triển phức tạp. Các modeler có thể mô hình cho hàng nghìn các tác tử hoạt động độc lập. Điều này làm cho nó có khả năng tìm hiểu được các liên kết giữa các phần tử dựa theo sự tương tác giữa chúng.

NetLogo cho phép các sinh viên có thể mô phỏng hay chơi với chúng, tìm hiểu các hành vi của chúng dưới nhiều điều kiện môi trường khác nhau. Nó cũng là môi trường để các sinh viên, giảng viên có thể tự tạo cho họ các mô hình riêng. NetLogo đủ đơn giản để các sinh viên hay giáo viên sử dụng, và cũng đủ nâng cao để các server có thể thực hiện các nghiên cứu với hàng trăm các trường khác nhau.

NetLogo là thế hệ tiếp theo của ngôn ngữ mô hình đa tác tử gồm StarLogo và StarLogoT. NetLog chạy trên máy ảo Java, vì thế nó có thể làm việc được trên nhiều nền tảng ( Window, Mac, Linux, …). Nó chạy như một ứng dụng độc lập. Các mô hình và HubNet có thể chạy như một Java applets trên trình duyệt web. Các câu lệnh cũng được hỗ trợ.

## Đặc điểm của NetLogo:

* Về hệ thống:
* Là một phần mềm miễn phí,mã nguồn mở.
* Độc lập nền tảng : chạy trên Mac,Window, Linux,…
* Hỗ trợ các các đặc điểm quốc tế.
* Khả năng lập trình:
* Có thể lập trình hoàn toàn.
* Ngữ pháp dễ hiểu
* Ngôn ngữ là ngôn ngữ mở rộng của Logo để hỗ trợ các tác tử.
* Các tác tử di động ( turtles ) di chuyển qua các ô của các tác tử cố định ( patches).
* Các tác tử link sẽ kết nối các turtles để tạo nên mạng lưới, sơ đồ và tập hợp
* Nhiều từ vựng được tích hợp trong ngôn ngữ.
* Độ chính xác cao với các số dấu phẩy động
* Chạy độc lập với các nền tảng.
* Môi trường:
* Trung tâm lệnh cho các tương tác trực tiếp
* Giao diện dựng sẵn với các nút, các thanh trượt , công tắc , choosers, monitors, text boxes, output area.
* Các tab thông tin để chú thích cho các model của bạn với các text và ảnh.
* HubNet: Mô phỏng nhiều người tham gia sử dụng mạng kết nối.
* Monitors agent để xem xét và kiểm soát các tác tử.
* Các chức năng import và export ( export data, lưu trữ các trạng thái của model, có thể tạo một bộ phim).
* BehaviorSpace là một công cụ mã nguồn sử dụng để thu thập dữ liệu từ nhiều model chạy song song.
* System Dynamics Modeler.
* NetLogo 3D để mô hình thế giới 3D.
* Khả năng hiển thị và trực quan hóa:
* Sử dụng các đường, thanh bar và các sơ đồ rời rạc.
* Thanh trượt tốc độ cho phép bạn đẩy nhanh tốc độ mô hình hoặc thấy nó trong trạng thái chậm.
* Xem các mô hình của bạn ở dạng 2D hoặc 3D.
* Các vector có thể xoay và thu phóng được.
* Có các nhãn cho turtle và patch.
* Web :
* Các mô hình và các client HubNet có thể lưu lại như là một applets để có thể nhúng vào trang web .
* APIs:
* Các APIs điều khiển cho phép nhúng NetLogo vào các script hay các ứng dụng.
* Các APIs mở rộng cho phép thêm các dòng lệnh và các báo cáo tới ngôn ngữ NetLogo.

# II - Sử dụng NetLogo:

## Các mô hình :

VD : Mô hình Sói và cừu :

Mở mô hình sói và cừu, giao diện sẽ hiện ra với nhiều thành phần như là các buttons, switches, sliders và monitors. Các thành phần giao diện đó cho phép bạn có thể tương tác với mô hình. Các nút có màu xanh da trời, dùng để cài đặt, chạy và dừng mô hình. Các sliders và switches có màu xanh lá cây, là các cài đặt của mô hình. Các monitors và plots có màu vải len, dùng để hiển thị dữ liệu.

Nếu bạn muốn tạo ra một cửa sổ to hơn để nhìn rõ mọi thứ hơn, hãy dùng công cụ Zoom.

Khi bạn mở mô hình lần đầu tiên, bạn sẽ thấy phần view bị trống. Để bắt đầu chạy một mô hình, bạn cần phải cài đặt nó:

* Ấn nút “setup”.
* Ấn nút “go” để bắt đầu chạy mô hình mô phỏng.
* Ấn nút “go” lần nữa để dừng mô hình.

### Điều khiển mô hình với các nút (buttons):

Khi các nút được ấn, mô hình sẽ phản hồi với các hành động tương ứng. Một nút có thể là nút tác dụng “một lần”, hoặc nút có tác dụng “mãi mãi”. Bạn có thể thấy sự khác biệt giữa các loại nút bằng cách nhìn vào bề mặt của các nút. Nút có tác dụng mãi mãi sẽ có 2 mũi tên ở góc dưới bên phải của nút. Giống như thế này:

C:\Users\Duy Minh\Desktop\button.PNG

Nút có tác dụng một lần sẽ không có các mũi tên, giống như thế này :

C:\Users\Duy Minh\Desktop\setup.PNG

Nút có tác dụng một lần sẽ tạo hành động và sau đó dừng lại. Khi một hành động được hoàn thành, nút sẽ nổi lên trở lại.

Nút có tác dụng mãi mãi sẽ thực hiện một hành động lặp lại nhiều lần. Khi bạn muốn hành động dừng lại, ấn nút đó lại lần nữa. Nếu nó hoàn thành hành động hiện tại, nó sẽ bật trở lại.

Hầu hết các mô hình, bao gồm cả mô hình sói và cừu này, sẽ có một nút gọi là nút “setup” và một nút mãi mãi là “go”. Nhiều mô hình cũng có một nút một lần là “go once” hay “step once” giống như là nút “go” trừ khi mô hình chỉ chạy từng bước một thay vì lặp lại nhiều lần. Sử nút một lần giống như việc bạn nhìn các proccess của mô hình một cách chi tiết.

Dừng một nút mãi mãi là cách thông thường để tạm dừng hay dừng hẳn một mô hình. Sau khi tạm dừng bạn có thể vào lại bằng cách ấn nút đó lần nữa. ( Bạn cũng có thể dừng mô hình lại với item “Halt” ở trên menu Tools, nhưng mà bạn chỉ nên làm điều này nếu mô hình bị treo vì lí do gì đó. Sử dụng “Halt” có thể gây ra việc đổ vỡ mô hình tại thời điểm giữa sự kiện và gây ra kết quả không tốt).

### Điều khiển tốc độ mô hình với Speed Slider:

Speed Slider cho phép bạn có thể điều khiển tốc độ của mô hình như là tốc độ di chuyển của các turtles, sự thay đổi màu sắc của patches và tương tự thế.

C:\Users\Duy Minh\Desktop\speedSlider.PNG

Khi bạn chuyển slider sang bên trái, mô hình sẽ chậm lại và có nhiều sự dừng lại trong mỗi stick ( một bước chạy của thời gian). Điều đó làm cho nó dễ dàng để thấy được những sự việc diễn ra. Bạn thậm chí có thể làm mô hình chậm lại để thấy được chính xác từng phần tử hoạt động.

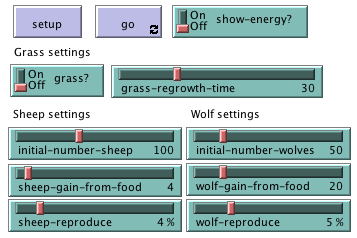
Khi bạn chuyển slider sang bên phải, tốc độ của mô hình sẽ tăng lên. NetLogo sẽ bắt đầu bỏ qua những frame, vì thế nó sẽ không cập nhật các khung cảnh ở mỗi lần thời gian chạy, mà chỉ dừng ở một số frame nào đó. Điều này làm cho ta thấy diễn biến mô hình sẽ xảy ra nhanh hơn.

Chú ý rằng khi bạn ấn vào speed slider sang bên phải, khung hình có thể cập nhật ít hơn vì thế các hình ảnh trên mô hình sẽ xuất hiện ít hơn và chậm đi.

### Điều chỉnh các cài đặt bằng Sliders và Switches :

Một cài đặt của mô hình giúp bạn có thể tìm hiểu các điều kiện, kịch bản và giả thiết khác nhau. Thay đổi các cài đặt và sau đó chạy các mô hình để thấy sự tương tác của nó giúp bạn có thể hiểu sâu hơn hiện tượng được mô hình hóa.

Các switches và sliders đưa cho bạn các thông tin và điều khiển cài đặt cho mô hình. Chúng ta có thể thấy được các sliders và switches trong mô hình cừu và sói như sau:



Bên cạnh switches, một mô hình có thể có các sliders. Trong khi một switch chỉ có hai giá trị là on hay off thì slider có một dải giá trị số. Ví dụ, slider “initial-nmber-sheep” có giá trị nhỏ nhất là 0 và lớn nhất là 250 con cừu. Mô hình có thể chạy với 0 con cừu hay có thể chạy với 250 con cừu, và bất cứ giá trị nào ở giữa hai số đó. Khi bạn chuyển giá trị từ nhỏ nhất đến lớn nhất, số tương ứng ở bên phải thanh slider sẽ thay đổi.

Tab info để đưa ra các hướng dẫn và chỉ rõ bên trong mô hình. Trong tab này bạn sẽ tìm được phần giải thích cho mô hình, gợi ý ra các thứ để thử, và các thông tin khác. Bạn có thể cần phải đọc các thông tin ở Tab info trước khi chạy mô hình, hoặc bạn có thể muốn bắt đầu mô hình trước sau đó xem xét ở tab info sau.

Khi bạn mở một mô hình lên, tất cả các sliders và switches đều mặc định là bật. Nếu bạn mở một mô hình mới lên hoặc một chương trình đã tồn tại, các thay đổi cài đặt của bạn sẽ không được lâu trừ khi bạn chọn để lưu chúng lại.

Chú ý, ngoài sliders và switches , một số mô hình có choosers và input boxes.

### Thu thập thông tin bằng Plots và Monitors:

Phần khung cảnh giúp bạn thấy được điều gì đang xảy ra ở trong mô hình.

NetLogo cũng cung cấp một số cách để bạn có thể thu thập thông tin về mô hình đang chạy, như là plots và monitors.

Plots ( Đồ thị ) :

Đồ thị ở trong mô hình Sói và cừu chứa 3 đường: số cừu, số sói và (số cỏ)/4. Số cỏ được chia cho 4 giúp cho plots không quá cao. Các đường này chỉ ra điều gì đang diễn ra trong mô hình trong quá trình thời gian diễn ra. Ghi chú của đồ thị chỉ ra thông tin của mỗi đường. Trong trường hợp này, là các thông tin về số đếm.

Khi một đồ thị đã vẽ hoàn thành, đường tạo độ chiều ngang sẽ được nén lại tất cả các dữ liệu vào thành một phần nhỏ.

Nếu bạn muốn lưu các dữ liệu từ một đồ thị để xem và phân tích nó trong ứng dụng khác, sử dụng item “Export Plot” trong menu “File”. Nó lưu dữ liệu data trong một định dạng có thể đọc được bằng bảng tính và các chương trình dữ liệu như Excel. Bạn cũng có thể xuất một đồ thị ra bằng cách ấn chuột phải và chọn “Export…” từ popup menu.

Monitors:

Monitors là một công cụ khác dùng để hiển thị thông tin từ các mô hình. Trong mô hình sói và cừu, các monitors là :

C:\Users\Duy Minh\Desktop\monitors.PNG

Các monitors sẽ hiển thị cho chúng ta thấy số cừu và số sói, và lượng cỏ. ( Nên nhớ rằng, lượng cỏ được chia cho 4 để giữ cho đồ thị không quá cao).

Số lượng được hiển thị trong monitors sẽ thay đổi khi mô hình chạy, trong khi đồ thị sẽ hiển thị tất cả dữ liệu từ toàn bộ tiến trình của mô hình chạy.

### Điều khiển phần View:

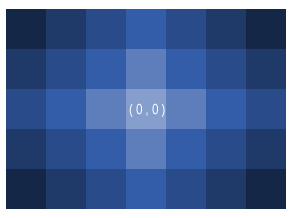
Ở tab Interface, bạn sẽ thấy thanh toolbar để điều khiển. Tăng tốc mô hình và tắt đi phần update của phần view rất có ích nếu bạn muốn mô hình chạy nhanh hơn. Tăng tốc độ ( bằng cách chuyển thanh slider sang bên phải ) sẽ bỏ phần update của view và mô hình sẽ chạy nhanh, vì quá trình cập nhật phần view tốn thời gian để chạy mô hình.

Khi việc cập nhật phần view bị tắt, mô hình sẽ tiếp tục chạy nền, và đồ thị và monitors vẫn tiếp tục cập nhật. Nhưng nếu bạn muốn thấy những sự việc đang diễn ra, bạn cần phải bật phần cập nhật view lên bằng cách đánh dấu trở lại và box. Nhiều mô hình chạy nhanh hơn nhiều khi phần cập nhật view bị tắt.

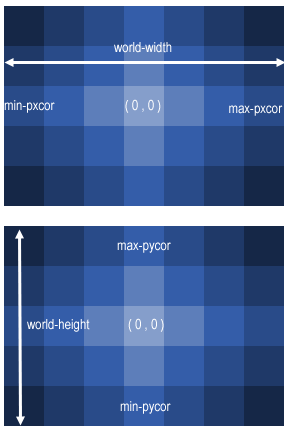
Độ lớn của phần view được quyết định bởi năm phần cài đặt: min-pxcor, max-pxcor, min-pycor, max-pycor và độ lớn của patch. Chúng ta hãy nhìn việc gì xảy ra khi chúng ta thay đổi kích thước của phần view.

Thế giới NetLogo là một khung 2 chiều gồm nhiều mẩu nhỏ. Mẩu (“Patches”) là các hình vuông nhỏ riêng biệt trên một mảng lưới. Trong mô hình Sói-Cừu, khi số lượng cỏ được bật lên thì các phần nhỏ của các patches sẽ dễ thấy hơn, bởi vì một số có màu xanh lá cây, một số thì có màu nâu.

Hãy xem các patches giống như các phần hình vuông trong một phòng với một cái nền nhiều gạch. Mặc định, phần gạch ở trung tâm phòng có giá trị là (0,0) , điều đó có nghĩa là nếu phòng được chia thành một nửa này và nửa kia, có hai đường sẽ giao nhau qua viên gạch trung tâm này. Chúng ta có một hệ thống tỷ lệ giúp chúng ta định vị được phần tử đang ở đâu trong phòng.



Trong NetLogo, số lượng viên gạch từ trái sang phải được gọi là độ rộng của thế giới. Và số lượng mẩu gạch từ trên xuống dưới được gọi là chiều cao của thế giới. Các số đó được định nghĩa bởi biên top, bottom, left và right.



Như ở trên, max-pxcor = 3, min-pxcor = -3, max-pycor = 2, min-pycor = -2.

Khi bạn thay đổi kích thước của patch, số lượng của patches không thay đổi, patches chỉ to ra hay nhỏ hơn thôi.

Nhìn vào hiệu quả của việc thay đổi từ nhỏ nhất đến lớn nhất trong thế giới. Việc edit phần view cũng giúp bạn thay đổi cài đặt khác.

Một khi bạn khám phá mô hình Sói-Cừu, bạn có thể muốn có thời gian để kiểm tra các mô hình khác ở trong thư viện các mô hình.

Thư viện mô hình:

Thư viện chứa 4 phần : Các mô hình mẫu, mô hình giảng dạy, mã nguồn ví dụ, và các activies HubNet.

## Câu lệnh :

Trong phần này, chúng ta sẽ tập trung vào việc để điều khiển các mô hình. Bạn sẽ thấy sự hoạt động bên trong của các mô hình và có thể để thay đổi phần view của chúng.

Mô hình mẫu: Giao thông cơ bản:

Trong mô hình này, bạn chú ý đến một chiếc xe màu đỏ trong một dải nhiều xe màu xanh. Dòng xe này sẽ di chuyển cùng hướng. Vì có nhiều xe nên chúng sẽ phải dừng lại và chờ đợi. Mô hình này mô tả lại một trạng thái giao thông khi không có điều bất ngờ xảy ra như là tai nạn giao thông.

Bạn có thể thay đổi các cài đặt và chạy thử chương trình một vài lần để có thể hiểu rõ về mô hình hơn.

Nhìn vào mô hình Giao thông, bạn có thể để ý là môi trường khá đơn giản, một nền màu đen với một đường phố màu trắng và một số xe xanh và một chiếc xe đỏ. Các thay đổi có thể thiết lập bao gồm : thay đổi màu sắc và kích thước của những chiếc xe, thêm vào các ngôi nhà và đèn đường, tạo ra một đèn dừng hay thậm chí là tạo ra một làn đường khác. Một số gợi ý cho các thay đổi có thể thấy rất rõ ràng, nâng cao khả năng nhận diện của mô hình. Chúng ta sẽ tập trung hơn vào các thay đổi đơn giản và làm đẹp.

### Trung tâm câu lệnh :

Trung tâm lệnh được tìm thấy ở trên tab Interface. Nó cho phép bạn có thể thêm các câu lệnh hay điều hướng cho một mô hình. Các câu lệnh được cấu trúc để bạn có thể tác động đến các tác tử của NetLogo như là :turtles, patches, links và observer.

Thế giới NetLogo là thế giới 2 chiều được tạo bởi turtles, patches, links và observer. Tác tử patches là phần nền cho các tác tử turtles di chuyển. Links là phần kết nối giữa các turtles. Tác tử observer giám sát thế giới.

Trong trung tâm lệnh bạn có thể đưa ra một câu lệnh cho bất kì tác tử nào. Bạn chọn loại tác tử nào bằng cách sử dụng popup menu nằm ở góc dưới bên trái. Bạn cũng có thể sử dụng phím tab để xoay vòng qua các loại khác nhau.

Tác tử observer giám sát thế giới và vì thế chúng có thể đưa ra một câu lệnh để các tác động đến turtles và patches bằng cách sử dụng câu lệnh “ask”. Ví dụ như observer có thể ask các patches để set màu sắc của các tác này thành màu vàng. Nhưng khi một câu lệnh được đưa ra trực tiếp đến một nhóm các tác thì bạn chỉ cần đưa ra câu lệnh là được.

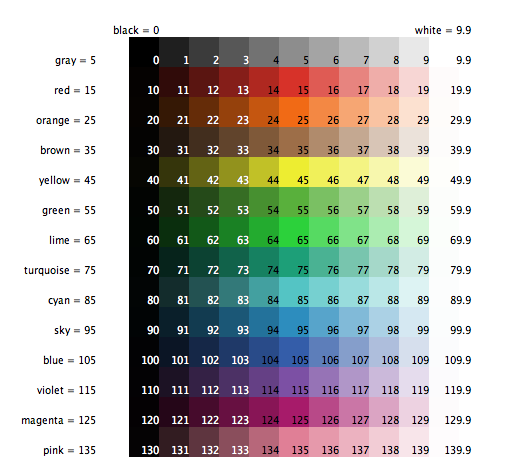
Trung tâm lệnh không thay đổi vĩnh viễn mô hình. Nó cho phép bạn điều khiển thế giới NetLogo trực tiếp ngay trên mô hình.

### Điều khiển màu sắc :

Trong ví dụ,chúng ta có 2 cách để thay đổi màu sắc : color và pcolor. Chúng ta gọi là biến color và pcolor . Một số command và biến đặc biệt với các turtles và một số patches. Ví dụ, biến color là biến của turtles, trong khi đó biến pcolor là biến của patch.

Để có thể thay đổi màu sắc của turtles và patches, hay như chúng ta nói là thay đổi màu sắc của xe oto và màu nền, chúng ta cần tìm hiểu cách mà NetLogo tương tác với màu sắc.

Trong NetLogo, màu sắc có là số có giá trị. Trong tất cả bài học, chúng ta đã sử dụng tên màu sác. Điều này là do NetLogo tổ chức 16 tên màu sắc khác nhau. Điều này không có nghĩa là NetLogo chỉ có 16 màu. Có nhiều màu giữa các màu đó cũng được sử dụng. Chúng ta cùng xem xét không gian màu của NetLogo:



Để lấy trực tiếp một màu, bạn sẽ cần lấy nó thay vì là các số của nó, hay là bằng cách thêm hay trừ đi một số từ tên của màu. Ví dụ, khi bạn set color red, điều này cũng tương tự như khi bạn viết là set color 15. Và bạn có thể lấy các màu tối hay sáng hơn của cùng một màu bằng cách sử dụng một số nhỏ hơn hay lớn hơn.

Định hướng và điều khiển các tác tử:

Ở các hoạt động trước, chúng ta sử dụng câu lệnh set để thay đổi màu sắc của các xe oto. Nhưng khi bạn gọi lại, mô hình ban đầu chứa 1 xe đỏ giữa nhiều xe xanh.

## Hàm trong NetLogo:

### Tác tử và hàm,thủ tục:

Trong phần trước, chúng ta đã học cách để sử dụng trung tâm lệnh và quan sát các tác tử để xem xét và thay đổi tác tử và điều khiển chúng thực hiện hành vi. Bây giờ, bạn sẽ học phần cốt lõi của NetLogo: là phần code.

Bạn đã thấy các tác tử trong NetLogo được chia thành patches, turtles, links và observer. Patches là phần đứng yên và tạo ra mạng lưới thế giới. Turtles di chuyển giữa các patches. Links là phần kết nối giữa hai turtles. Còn observer giám sát mọi thứ như là turtles, patches và links.

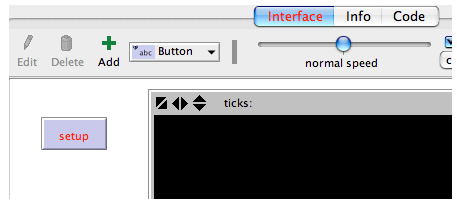
Bạn sẽ học để viết các thủ tục ( hàm ) điều khiển turtles di chuyển, ăn uống sinh ra và chết đi. Bạn cũng sẽ học làm thế nào để tạo ra monitors , sliders và plots. Mô hình chúng ta sẽ xây dựng là một hệ thống sinh thái không giống như mô hình Sói – cừu ở phần 1.

Tạo ra nút Setup :

Để bắt đầu mô hình mới, chọn “New” từ menu File. Sau đó tạo ra nút Setup bằng cách:

* Ấn icon “Add” trên toolbar ở phần trên của tab Interface.
* Trên menu tiếp để thêm ấn chọn Button nếu nó chưa được chọn.
* Ấn vào bất kì nơi nào mà bạn muốn thêm nút trong vùng trắng trống của tab Interface.
* Một dialog hiện ra để thay đổi cài đặt của nút hiện ra. Viết setup vào ô “Command”.
* Ấn OK khi bạn hoàn thành, dialog box sẽ tắt.

Bây giờ bạn đã có nút setup. Ấn vào nút chạy một thủ tục có tên là “setup”. Một thủ tục là một dãy các lệnh của NetLogo mà chúng ta khai báo tên. Chúng ta sẽ định nghĩa hàm này sớm, nhưng giờ thì chưa. Nút mà chưa được định nghĩa hàm thì có màu đỏ :



Nếu bạn muốn thấy thông báo lỗi, bạn ấn nút sẽ có.

Bây giờ chúng ra sẽ tạo ra hàm “setup”, và thông báo lỗi sẽ không còn nữa:

Chuyển sang tab Code. Viết vào như sau:

to setup

clear-all

create-turtles 100 [ setxy random-xcor random-ycor ]

reset-ticks

end

Khi bạn hoàn thành, tab Code sẽ như sau :



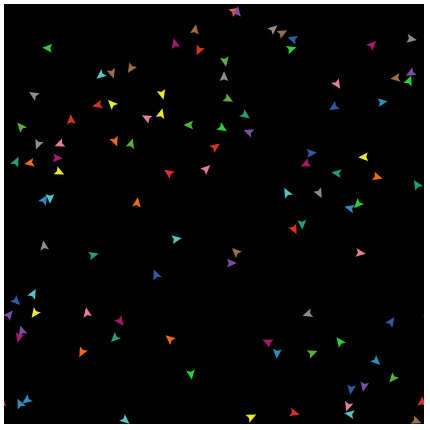
Chú ý rằng một số dòng không đều. Hầu hết mọi người đều thấy nó có ích để xác định code của họ. Nó không bắt buộc, nhưng nó khiến cho code dễ hiểu và dễ thay đổi hơn.

Hàm của bạn bắt đầu với to và kết thúc với end. Mọi hàm đều bắt đầu và kết thúc bằng các từ đó.

Bây giờ hãy nhìn vào code mà bạn đã viết vào và xem xét mỗi dòng code thực hiện:

* To setup bắt đầu định nghĩa một hàm gọi là “setup”.
* clear-all khởi tạo thế giới với một trạng thái trống. Tất cả các patches đều màu đen và bất kì turtles nào mà bạn tạo ra đều không xuất hiện.
* create-turtles 100 tạo ra 100 turtles. Chúng bắt đầu đứng im ở vị trí trung tâm là 0,0.
* Sau khi create-turtles chúng ta có thể thêm lệnh để các turtles mới hoạt động trong dấu ngoặc vuông.
* setxy random-xcor random-yconr là một lệnh sử dụng “reporters”. Một reporters, trái ngược với một lệnh, để báo lại một kết quả. Mỗi turtles chạy reporter random-xcor để báo cáo một số trong dải cho phép của hệ tọa độ turtles dài là X. Sau đó mỗi turtles chạy reporter random-ycor đối với tọa độ Y. Cuối cùng mỗi turtle chạy lệnh setxy với 2 số đó như là đầu vào. Điều đó làm cho turtles di chuyển giữa các điểm trên hệ tọa độ đó.
* Reset-ticks bắt đầu đếm các tích tắc, bây giờ thì cài đặt đã xong.
* end sẽ hoàn thành hàm “setup”.

Khi bạn viết xong hàm, chuyển sang tab Interface và ấn nút setup bạn đã tạo ra trước đó. Bạn sẽ thấy các turtles chạy tán loạn trên màn hình.



Ấn nút setup thêm nhiều lần nữa, và quan sát cách mà các tác tử turtles thay đổi sau mỗi lần. Chú ý rằng một số turtles có thể ở phía trên bên phải những tác tử khác.

Suy nghĩ một chút về điều mà bạn cần làm để điều khiển các tác tử hoạt động. Bạn cần phải làm các nút ở trên giao diện và tạo hàm tương ứng với nút sử dụng. Nút chỉ hoạt động được một khi bạn hoàn thành cả hai bước trên. Như đã nói ở trên, bạn sẽ thường xuyên hoàn thành 2 bước trên hoặc các bước tương tự khác để thêm chức năng của mô hình. Nếu một số thứ không xuất hiện để hoạt động sau khi bạn đã hoàn thành bước mà bạn nghĩ là bước cuối cùng thì bạn phải tiếp tục đọc và xem xét thêm những việc cần phải làm. Sau khi đọc tiếp thì bạn nên quay trở lại để xem xét có bước nào bị bỏ qua hay không?

Chuyển sang phần tick-based để xem update:

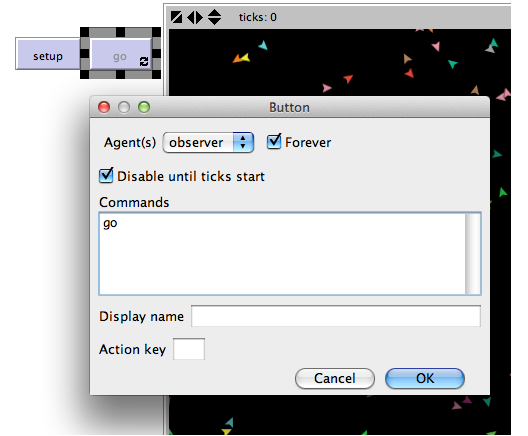
Bây giờ chúng ta sử dụng cách đếm tick ( với reset-ticks) chúng ta nên nói với NetLogo chỉ cần cập nhật phần view sau mỗi tick, thay vì tiếp tục update nó.

* Tìm menu cập nhật phần view. Nó ở phía trên phần view và mặc định là “continuous”.
* Chọn “on ticks”.

Điều này khiến cho mô hình của bạn chạy nhanh hơn và đảm bảo hình ảnh mượt mà hơn. Xem phần hướng dẫn lập trình để hiểu rõ về phần cập nhật view.

Tạo nút go:

Bây giờ có thể tạo nút “go”. Làm theo các bước như đã làm để tạo nút “setup”, trừ một số chi tiết sau:

* Trong phần commands gõ “go” thay vì “setup”.
* Đánh dấu vào checkbox “Forever” trong dialog edit.
* Đánh dấu vào checkbox “Disable util ticks start”. 

Checkbox “Forever” khiến cho nút tiếp tục giữ với chỉ một lần ấn, vì thế lệnh của nó có thể chạy lặp lại, không chỉ với một lần.

Checkbox “Disable util ticks start” tránh việc bạn ấn nút go trước nút setup.

Sau đó thêm hàm go vào phần Code:

to go

move-turtles

tick

end

tick là một từ nguyên thủy với các biến tick được đếm bởi 1 lần thời gian.

Nhưng với từ move-turtles thì sao? Nó có phải cũng là một từ trong ngôn ngữ NetLogo. Không, nó là một hàm khác mà bạn sẽ phải thêm vào. Như vậy, bạn đã được giới thiệu hai hàm là setup và go.

Thêm hàm move-turtles sau hàm go như sau :

to go

move-turtles

tick

end

to move-turtles

ask turtles [

right random 360

forward 1

]

end

Chú ý rằng không có khoảng trắng trong từ move-turtles. Trong phần 2 chúng ta sử dụng red – 2 , có khoảng trắng, để trừ hai số, nhưng ở đây chúng ta muốn move-turtles mà không có khoảng trắng. Dấu “-”giữa “move” và “turtles” đã kết hợp 2 từ thành một tên.

Đây là hành động mà hàm move-turtles thực hiện:

* Ask turtles […] ra lệnh cho mỗi turtles chạy câu lệnh ở trong dấu ngoặc vuông.
* Right random 360 là một lệnh khác sử dụng một reporter. Trước hết, mỗi turtles chọn một số ngẫu nhiên giữa 0 và 359. Sau đó turtles sẽ xoay sang phải số độ .
* Forward 1 ra lệnh cho turtles di chuyển lên một bước.

Tại sao chúng ta không viết tất cả vào trong một hàm là go mà lại chia ra các hàm? Chúng ta có thể làm thế,nhưng trong suốt quá trình học để tạo ra project của bạn, nó thuận tiện cho bạn sẽ thêm nhiều phần khác. Chúng ra muốn hàm go càng đơn giản càng tốt, để dễ hiểu hơn. Thậm chí, nó có thể gồm nhiều phần khác diễn ra khi mô hình chạy, như là việc tính toán một số thứ hay là vẽ đồ thị một kết quả. Mỗi thứ đó khi thực hiện sẽ có hàm riêng của nó và mỗi hàm sẽ có tên riêng.

Nút ‘go’ bạn đã tạo ra ở phần Interface là nút forever, điều đó có nghĩa là nó sẽ tiếp tục chạy các câu lệnh của nó cho tới khi bạn ấn nó. Sau khi bạn ấn nút setup một lần, để tạo ra các turtles, ấn nút ‘go’. Hãy quan sát những gì diễn ra. Tắt nó đi, bạn sẽ thấy tất cả các turtles đều dừng lại.

Chú ý rằng các turtles khi di chuyển ra vùng biên thì nó sẽ biến sang phía bên kia.

### Thử nghiệm các câu lệnh:

Chúng tôi đã gợi ý cho bạn thử nghiệm với các câu lệnh khác của turtles. Viết câu lệnh vào Command Center ( như là turtles> set color red) , hay thêm câu lệnh vào hàm setup, go và move-turtles.

Chú ý rằng khi bạn thêm lệnh vào Command Center, bạn cần phải chọn turtles>, patches>, links>, hay observer> trong menu popup ở bên trái, phụ thuộc vào tác tử nào sẽ thực hiện lệnh đó. Nó giống như sử dụng lệnh ask turtles hay ask patches nhưng đã được lưu lại. Bạn có thể sử dụng phím tab để thay đổi các loại tác tử.

Bạn có thể thử ấn turtles> pen-down trong Command Center và sau đó ấn nút go.

Trong hàm move-turtles bạn cũng thử thay đổi right random 360 thành right random 45.

Chơi lại vài lần. Nó rất dễ dàng và kết quả sẽ hiện ra trực tiếp.

Khi bạn cảm thấy bạn đủ kinh nghiệm để thử nghiệm, bạn sẽ tiếp tục phát triển thêm mô hình mà bạn đã xây dựng.

### Patches và biến :

Bây giờ chúng ta có 100 turtles di chuyển lung tung, không có thông tin gì về chúng cả. Chúng ta sẽ làm cho chúng hấp dẫn hơn bằng cách làm cho nền của chúng đẹp hơn.

Trở lại hàm setup, chúng ta viết lại hàm như sau :

to setup

clear-all

setup-patches

setup-turtles

reset-ticks

end

Hàm setup mới có 2 hàm. Để định nghĩa hàm setup-patches, thêm vào như sau :

to setup-patches

ask patches [ set pcolor green ]

end

Hàm setup-patches set màu cho các patches là màu xanh . Biến màu cho turtles là color còn biến màu cho patches là pcolor.

Phần còn lại của hàm setup và chưa được định nghĩa là setup-turtles :

Thêm vào hàm sau :

to setup-turtles

create-turtles 100

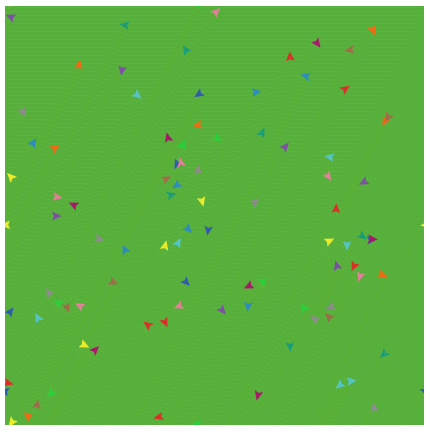
ask turtles [ setxy random-xcor random-ycor ]

end

Bạn có để ý rằng hàm setup-turtles mới có câu lệnh giống với hàm setup cũ?

* Trở lại phần Interface.
* Ấn nút setup.

Một thế giới của NetLogo hiện ra với các turtles và các patches màu xanh :



Sau khi thấy hàm setup mới thực hiện một vài lần, bạn có thể tìm ra cách định nghĩa hàm sao cho hợp lý.

Biến của turtles :

Chúng ta có một số turtles hoạt động giữa thế giới, nhưng mà chúng không tác động gì đến thế giới cả. Chúng ta thêm một số tương tác giữa turtles và patches.

Chúng ta sẽ làm cho các turtles ăn cỏ ( chính là các tác tử patches màu xanh ) , sinh ra và chết. Cỏ sẽ dần dần phát triển trở lại sau khi bị ăn.

Chúng ta cần một cách để điều khiển khi một turtles sinh ra và chết đi. Chúng ta sẽ quyết định như thế nào dựa vào năng lượng của mỗi turtles có. Để làm điều đó, chúng ta cần thêm biến cho turtles .

Bạn đã thấy biến có sẵn của turtles như là color. Để tạo một biến mới của turtles, chúng ta khai báo thêm turtles-own ở đầu phần Code, trước tất cả các hàm khác. Gọi nó là năng lượng:

turtles-own [energy]

to go

move-turtles

eat-grass

tick

end

Sử dụng biến mới khai báo năng lượng ( energy) cho phép các turtles để ăn.

* Trở lại Code tab
* Viết lại hàm go như sau :

to go

move-turtles

eat-grass

tick

end

* Thêm hàm eat-grass:

to eat-grass

ask turtles [

if pcolor = green [

set pcolor black

set energy energy + 10

]

]

End

Chúng ta sử dụng câu lệnh if cho lần đầu. Nhìn vào code một cách cẩn thận, Mỗi turtles khi chạy các câu lệnh của nó,hãy so sánh với giá trị của màu patch với giá trị của màu green. Nếu màu của patch là màu xanh, giá trị so sánh được báo là true, và chỉ sau đó các turtles chạy các câu lệnh bên trong ngoặc vuông. Câu lệnh ra lệnh cho các turtles thay đổi màu sắc của patch thành màu đen và tăng năng lượng của nó thêm 10. Patch khi trở thành màu đen báo hiệu rằng cỏ đã bị ăn. Và turtles có thêm năng lượng khi ăn cỏ.

Tiếp theo, ta sẽ làm cho việc di chuyển của các turtles tiêu tốn năng lượng:

Viết lại hàm move-turtles như sau :

to move-turtles

ask turtles [

right random 360

forward 1

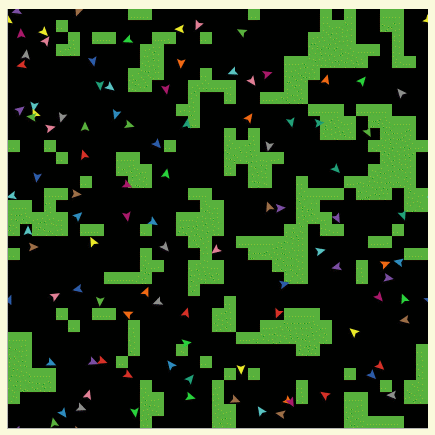
set energy energy - 1

]

end

Khi mỗi turtles di chuyển một bước, nó sẽ mất đi một đơn vị năng lượng.

* Chuyển sang phần Interface và ấn nút setup sau đó ấn nút go. Bạn sẽ thấy patches sẽ chuyển sang màu đen mỗi khi các turtles di chuyển giữa chúng.

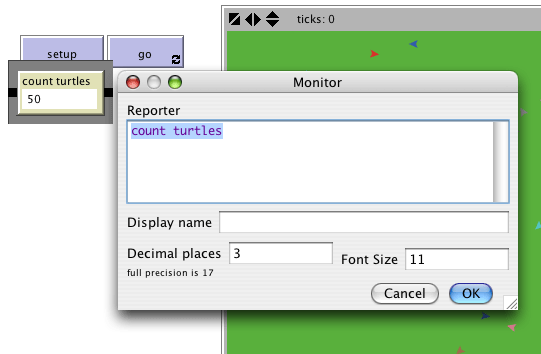


### Monitors :

Tiếp theo, bạn sẽ tạo ra hai monoitors trên phần giao diện với thanh công cụ. ( Bạn tạo ra chúng giống như tạo ra nút và sliders , bằng cách sử dụng icon Add trên thanh công cụ ) .

Chúng ta thử tạo một monitors .

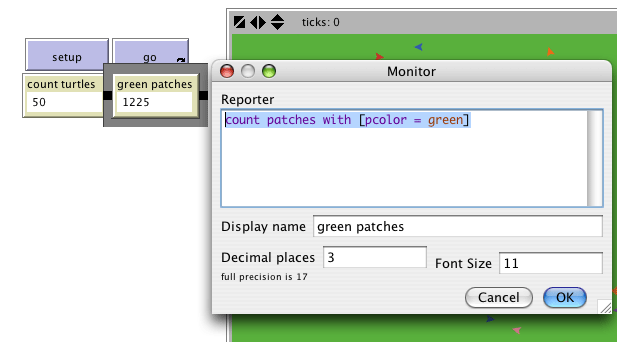
* Tạo ra một monitors bằng cách ấn icon Add trên thanh công cụ, chọn Monitor, ấn vào và một đồ thị xuất hiện ở trên giao diện.
* Một dialog sẽ hiện ra.
* Trong phần dialog viết ra count turtles .
* Ấn nút OK để tắt dialog.



Turtles là một chuỗi các tác tử. Hàm count nói cho chúng ta biết có bao nhiêu tác tử có trong chuỗi các tác tử.

Chúng ta hãy tạo ra monitors thứ 2:

* Tạo ra monitors bằng cách ấn vào icon Add trên thanh công cụ, chọn Monitors và ấn vào một đồ thị xuất hiện.
* Một dialog hiện ra
* Trong phần reporter hiện ra viết vào : count patches with [pcolor = green]
* Ở phần Display name của dialog box viết vào green patches.
* Ấn nút Ok để tắt dialog.



Ở đây, chúng ta sử dụng lại hàm count để thấy được có bao nhiêu tác tử có trong chuỗi các tác tử. Patches là chuỗi các tác tử patches, nhưng chúng ta muốn biết có bao nhiêu tất cả , có bao nhiêu tác tử màu xanh. Với một câu lệnh with, nó sẽ nhóm chuỗi tác tử thỏa mãn điều kiện có trong ngoặc vuông. Điều kiện pcolor = green, vì vậy nó sẽ đưa ra thôn tin các tác tử patches có màu xanh.

Bây giờ chúng ta có 2 monitors để báo cáo lại có bao nhiêu turtles và có bao nhiêu patches có màu xanh, để giúp chúng ta tìm ra được chuyện gì đang diễn ra trên mô hình. Khi mô hình chạy, số lượng ở trong monitors sẽ tự động thay đổi.

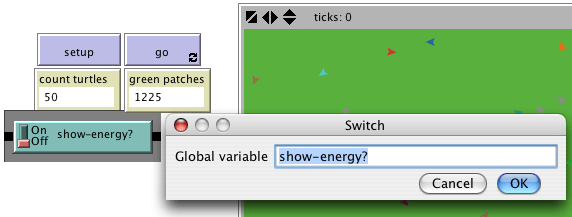
Sử dụng nút setup và nút go và xem số ở trong monitors thay đổi.

Switches và labels:

Các turtles chỉ làm cho các patches thành màu đen. Chúng cũng giành được hoặc mất đi năng lượng. Khi mô hình chạy, thử sử dụng monitor turtles để xem được năng lượng của turtles lên hay xuống .

Sẽ thật tốt nếu chúng ta có thể thấy được năng lượng của tác tử turtles trong suốt thời gian. Chúng ta sẽ làm như vậy và thêm vào một switch ( công tắc ) vì thế chúng ta có thể thêm thông tin và tắt chúng đi.

* Ấn vào icon Add trên toolbar
* Chọn switch từ mnenu và Add
* Ấn vào và một thứ hiện ra trên giao diện.
* Một dialog hiện ra.
* Trong phần biến Global , nhập vào show-energy? Đừng quên có dấu hỏi .



* Bây giờ trở lại hàm ‘go’ ở trong phần Code với thanh công cụ.
* Viết lại hàm eat-grass như sau :

to eat-grass

ask turtles [

if pcolor = green [

set pcolor black

set energy energy + 10

]

ifelse show-energy?

[ set label energy ]

[ set label "" ]

]

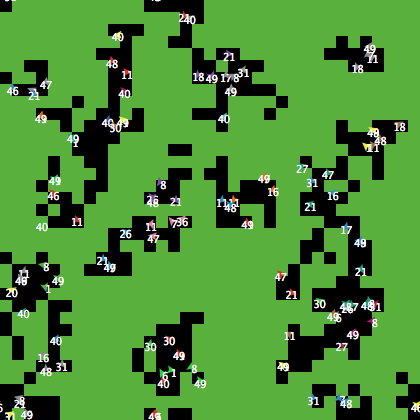
end

Hàm eat-grass có một hàm mới là ifelse. Nhìn vào phần code cẩn thận. Mỗi turtles khi chạy các câu lệnh đó đều kiểm tra giá trị của show-energy? . Nếu công tắc được mở , sự so sánh là true và turtles sẽ chạy câu lệnh bên trong ngoặc vuông trước. Trong trường hợp này , nó sẽ hiển thị giá trị của năng lượng trên label của turtles. Nếu sự so sánh là false ( công tắc tắt) thì turtles chạy câu lệnh trong ngoặc vuông thứ 2. Trong trường hợp này, nó sẽ xóa phần text có trong label.

Trong NetLogo, một phần text được gọi là string, phần nhỏ của string gọi là ký tự. Một string chứa nhiều ký tự , được viết giữa dấu ngoặc nháy kép. Ở đây dấu kép nháy ở bên phải các tác tử, và không có ký tự gì cả. Đó là string rỗng. Nếu label của các turtles là string rỗng thì nó không được hiển thị lên.

Khi công tắc được mở, bạn sẽ thấy năng lượng của mỗi turtles tăng lên nếu nó ăn grass.

Bạn cũng có thể thấy năng lượng giảm xuống khi mà nó di chuyển.



Thêm các hàm khác :

Bây giờ các turtles đang ăn. Chúng ta hãy làm thêm hàm để chúng sinh sôi và chết đi. Và chúng ta cũng làm cho grass mọc trở lại. Chúng ta thêm tất cả các hành vị đó bằng cách tạo ra 3 hàm riêng biệt.

* Tới phần code.
* Viết lại hàm go như sau :

to go

move-turtles

eat-grass

reproduce

check-death

regrow-grass

tick

end

* Thêm hàm reproduce, check-death và regrow-grass như sau:

to reproduce

ask turtles [

if energy > 50 [

set energy energy - 50

hatch 1 [ set energy 50 ]

]

]

end

to check-death

ask turtles [

if energy <= 0 [ die ]

]

end

to regrow-grass

ask patches [

if random 100 < 3 [ set pcolor green ]

]

end

Mỗi hàm đó đều sử dụng lệnh if. Mỗi turtles khi chạy đều kiểm tra check-death.

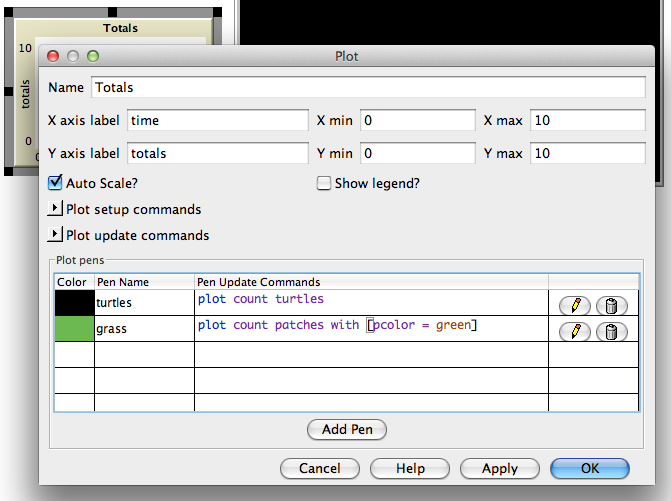
Đồ thị :

Để làm cho đồ thị hoạt động, chúng ta cần phải tạo ra một đồ thị ở trên giao diện và nhập vào đó một số câu lệnh.

Các câu lệnh chúng ta nhập vào trong đồ thị sẽ chạy tự động khi hàm setup của chúng ta gọi lệnh reset-ticks và khi hàm go gọi lệnh stick.

* Tạo ra đồ thị bằng cách ấn icon Add trên thanh công cụ, chọn Plot và ấn vao để tạo Plot.
* Đặt tên cho nó là “Totals”
* Đặt tọa độ X thành “time”
* Đặt tọa độ Y thành “totals”
* Thay đổi tên “default ” thành “turtles”
* Nhập vào plot count turtles dưới phần cập nhật Pen
* Ấn vào nút “Add Pen”
* Thay đổi tên thành “grass”
* Ấn nút plot count patches with [pcolor = green]

Khi bạn hoàn thành, dialog sẽ giống như thế này:

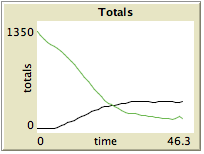


Ấn nút OK ở trong Plot để hoàn thành việc cài đặt.

Chú ý rằng khi bạn tạo ra đồ thị bạn cũng có thể đặt giá trị nhỏ nhất và lớn nhất của X và Y. Bạn sẽ muốn chọn chế độ “Auto Scale”, vì khi đó khi đồ thị hoạt động thì nó sẽ tự động tăng giảm theo thời gian.

Chú ý rằng chúng ta sử dụng lệnh plot để thêm điểm mới vào một đồ thị.

Bạn có thể quan sát các điểm được vẽ bởi đồ thị khi mô hình chạy. Đồ thị của bạn có thể sinh ra một hình giống như sau.



Nhớ rằng chúng ta đã chọn chế độ “Auto Scale”. Điều này cho phép đồ thị sẽ thay đổi kích thước khi mà mô hình chạy vượt quá giá trị.

Bộ đếm Tick:

So sánh sự khác nhau giữa các đồ thị khi mô hình chạy , người ta có thể so sánh tính chất mô hình với cùng một khoảng thời gian. Học cách dừng hay chạy một sự kiện tại một thời điểm đặc biệt mỗi khi mô hình chạy. Giữ lại vết của mô hình khi hàm go chạy có thể giúp ta nhận ra nhiều điều. Đó là việc mà bộ đếm làm.

Bạn đã sẵn sàng sử dụng bộ đếm tick trong mô hình của bạn, với lệnh reset-ticks và tick, nó sẽ chia đồ thị ra.

Bạn cũng có thể sử dụng bộ đếm tick cho việc khác, như là set giới hạn cho mô hình chạy.

* Thay đổi hàm go như sau :

to go

if ticks >= 500 [ stop ]

move-turtles

eat-grass

check-death

reproduce

regrow-grass

tick

end

# III – Mô hình mô phỏng đàn cá :

## Ý tưởng :

Mô phỏng một đàn cá đang bơi. Cá là sinh vật có tính bầy đàn cao, khi bơi chúng thường bơi theo một đàn gồm nhiều cá thể. Và khi cá bơi chúng thường bơi theo dòng tức là các con các sau sẽ bơi theo con cá đầu. Khi có việc xảy ra ( ví dụ như có hòn đá ném xuống, bị cá khác tấn công), chúng sẽ tản ra và sau đó chúng sẽ bơi lại gần nhau.

## Thiết kế:

Sử dụng NetLogo để mô phỏng mô hình này. Các tính chất của mô hình này :

* Cá bơi giữa đại dương.
* Cá có thể di chuyển, đại dương thường màu xanh
* Có thể nghĩ đến việc cá ăn tảo,…
* Cá di chuyển theo đàn.

Từ các tính chất trên, ta sẽ thiết kế cá là các tác tử turtles vì tác tử turtles có khả năng di chuyển thôi. Đại dương ( màu nền hoặc có tảo ) là các patches vì các patches là nền để các turtles di chuyển.

Sau khi chọn được các tác tử phù hợp với các thiết kế của mình. Ta tiến hành phân tích các hàm cần phải viết.

Trước hết, ta cần có hai hàm chính là hàm setup và hàm go. Hàm setup là hàm bắt đầu để cài đặt tất cả những giá trị đầu tiên cho hoạt động của các con cá. Hàm go là hàm sẽ mô tả hoạt động chính của những con cá.

Để mô phỏng đàn cá chúng ta cần chỉ ra số con cá mà chúng ta sẽ tạo ra là bao nhiêu. Như chúng ta đã phân tích, các con cá sẽ được mô phỏng bằng các tác tử turtles. Trong NetLogo, mặc định khi tạo ra một tác tử turtles thì nó có hình tam giác, chúng ta cần phải tạo cho chúng hình con cá bằng cách import từ trong mô hình. Trong mô hình NetLogo đã có sẵn hình của một số tác tử turtles, với cá nói riêng, ta có hình cá bình thường và hình cá mập. Chúng ta sẽ chọn lựa hình con cá bình thường vì ở đây ta đang mô phỏng đàn cá có xu hướng đi với nhau. Còn cá mập thường ít cá thể và ít khi đi theo đàn.

## Lập trình:

Ta có hàm setup cho đàn cá như sau :

to setup

clear-all

crt fishnumber

[ set color yellow - 2 + random 7

set size 1.0

set shape "fish"

setxy random-xcor random-ycor ]

setup-patches

reset-ticks

end

Ở đây, ta đã cài đặt cho đàn cá như sau :

Tạo ra một số cá thể turtles bằng với số “fishnumber”. “fishnumber” ở đây là một thanh slider để chúng ta có thể thay đổi khi cần. Màu sắc của các con cá là màu random từ màu vàng gốc ban đầu cộng trừ thêm một số random. Và hình dáng (“shape”) của các tác tử ở đây là “fish”. Lưu ý rằng, “fish” nằm trong thư viện của NetLogo nên chỉ cần viết như vậy. Trong hàm setup còn có một hàm nữa, đó là hàm setup-patches. Như chúng ta đã phân tích, đàn cá sẽ bơi lội giữa thế giới là các patches. Các patches này cần phải cài đặt. Hàm setup-patches như sau :

to setup-patches

ask patches [set pcolor green]

end

Hàm trên sẽ cài đặt các patches và set màu xanh cho nó. Như ta đã phân tích đàn cá sẽ bơi và ăn các tảo biển nên ở đây, mặc định ban đầu ta sẽ cho tảo biển ở khắp nơi.

Hàm quan trọng nhất của mô hình này chính là hàm go. Ta cần phân tích để đưa ra hàm go hợp lý. Chúng ta mô phỏng đàn cá nên việc quan trọng đầu tiên là phải mô phỏng quá trình bơi theo đàn của đàn cá. Và đàn cá thì sẽ ăn tảo biển. Tảo biển bị ăn sẽ mất đi, như vậy ta cần phải tạo ra một hàm để tảo biển có khả năng phát triển trở lại. Ngoài ra, ta còn có thể đặt màu nền cho đàn cá giống với đàn cá trong bể cá. Cụ thể hàm go như sau :

to go

ask turtles [

move-fish

eat-seaweed

]

regrow-seaweed

ifelse turn-on-background?

[change-environment]

[clear-drawing]

repeat 5 [ ask turtles [ fd 0.2 ] display ]

tick

end

Trong hàm go này ta chú ý các hàm quan trọng là hàm move-fish,eat-seaweed, regrow-seaweed và hàm thay đổi môi trường change-enviroment.

Trong hàm go trên chúng ta sẽ mô phỏng lại đàn cá đang di chuyển và ăn tảo biển. Vì tảo biển bị ăn nên nó cần có hàm để phát triển trở lại cho cá ăn. Và một hàm nữa là hàm thay đổi môi trường cho đàn cá. Ta sẽ đi phân tích kỹ từng hàm.

Hàm move-fish:

to move-fish

find-fishfriend

if any? fishfriend

[ find-nearest-neighbor

ifelse distance nearest-neighbor < minimum-separation

[ separate ]

[ align

cohere ] ]

end

Hàm move-fish sẽ có các hàm con như sau :

* + find-fishfriend
  + find-nearest-neighbor
  + separate
  + align
  + cohere

Cụ thể, hàm di chuyển của các con cá được mô tả như sau :

Các con cá sẽ tìm những con cá bạn xung quanh mình. Trong các con cá bạn, nó sẽ tìm con cá gần với nó nhất. Nếu con cá gần với nó nhất mà nhỏ hơn khoảng cách quy định giữa các con cá thì con cá này sẽ tách ra để tránh việc các con cá đụng vào nhau hoặc lấn vào đường đi của nhau. Nếu khoảng cách giữa các con cá không nhỏ hơn khoảng cách quy định thì cá sẽ thực thi hai hàm là align và cohere. Nghĩa là các con cá sẽ có xu hướng bơi theo hướng của các con cá khác, đồng thời nó cũng sẽ bơi gần với đàn cá của mình nhất. Như vậy, hàm move-fish sẽ mô phỏng sự di chuyển của đàn cá như sau :

Các con cá sẽ tìm những con cá bạn trong vòng bao nhiêu patches. Trong những con cá bạn đó, nó sẽ tìm con cá gần nhất. Nếu 2 con cá quá gần nhau, nhỏ hơn với quy định thì nó sẽ tách ra, tránh việc đụng độ vào nhau, cản trở đường đi của nhau. Nếu khoảng cách giữa các con cá không quá nhỏ, nó sẽ có xu hướng đi gần nhau và đi theo hướng của cả đàn.

Hàm find-fishfriend:

to find-fishfriend

set fishfriend other turtles in-radius friend-distance

end

Hàm này sẽ tìm những con cá nào cách nó hiện tại một khoảng cách là friend-distance để coi đó là bạn của nó.

Hàm find-nearest-neighbor như sau :

to find-nearest-neighbor

set nearest-neighbor min-one-of fishfriend [distance myself]

end

Hàm này sẽ tìm trong những con cá bạn của nó một con cá có khoảng cách đến nó là nhỏ nhất.

Hàm separate như sau :

to separate

turn-away ([heading] of nearest-neighbor) 1.5

end

Hàm này sẽ mô tả việc một con cá sẽ tránh ra con cá khác một góc bao nhiêu độ ( ở đây là 1.5 ). Như vậy là nếu quá gần nhau thì 2 con cá sẽ tránh ra nhau, quay đi một góc khác để đi.

Khi mà nó đi ra quá khoảng cách nhỏ nhất, nó sẽ thực hiện hàm align và cohere:

Hàm align:

to align

turn-towards average-fishmate-heading 5

end

với turn-towards :

to turn-towards [new-heading max-turn]

turn-at-most (subtract-headings new-heading heading) max-turn

end

và turn-at-most :

to turn-at-most [turn max-turn]

ifelse abs turn > max-turn

[ ifelse turn > 0

[ rt max-turn ]

[ lt max-turn]

]

[ rt turn ]

end

Chi tiết hàm align như sau :

Hàm align sẽ so sánh hiệu giá trị độ trung bình của tất cả các cá bạn tới nó với 5. Nếu giá trị tuyệt đối hiệu đó lớn hơn 5 và hiệu đó lớn hơn 0 thì nó sẽ quay sang phải một góc 5 độ, còn nếu hiệu đó nhỏ hơn 0 thì nó sẽ quay sang trái một góc 5 độ. Còn nếu giá trị tuyệt đối nhỏ hơn 0 thì nó sẽ quay sang phải một góc bằng giá trị hiệu này. Nói đơn giản là hàm này sẽ di chuyển theo giá trị trung bình của độ tất cả các tác tử. Nếu mà giá trị độ trung bình của tất cả tác tử lớn hơn 5 thì thì nó sẽ quay theo 5 độ, còn nếu không nó sẽ quay theo độ của nó. Nói cách khác là nó chỉ có thể quay hướng tối đa là 5 độ. Điều này giúp các tác tử không quay hướng quá nhiều. Nó giúp các tác tử duy trì được hướng chung của đàn.

Hàm cohere:

to cohere

turn-towards average-heading-towards-fishfriend 3

end

với turn-towards :

to turn-towards [new-heading max-turn]

turn-at-most (subtract-headings new-heading heading) max-turn

end

và turn-at-most :

to turn-at-most [turn max-turn]

ifelse abs turn > max-turn

[ ifelse turn > 0

[ rt max-turn ]

[ lt max-turn]

]

[ rt turn ]

end

Hàm cohere cũng tương tự như hàm align trên nhưng ở đây giá trị cần so sánh là giá trị độ trung bình tiến lên của các tác tử. Nghĩa là bây giờ, tác tử cần xác định các tác tử khác sẽ di chuyển như thế nào để nó cũng sẽ di chuyển theo như vậy. Nghĩa là tác tử sẽ dự đoán trước hướng di chuyển của đàn và đi theo. Với việc đi dự đoán trước hướng di chuyển của đàn để đi theo giúp tác tử có thể bám theo đàn với một khoảng cách cho phép để không bị bật ra khỏi đàn.

Hàm eat-seaweed :

to eat-seaweed;;an tao bien

if pcolor = green[

set pcolor blue

]

end

Hàm eat-seaweed là của tác tử turtles. Hàm này sẽ thực hiện như sau, khi đi đến chỗ nào có tảo biển nó sẽ ăn tảo biển ( chính là patches ) và màu tảo biển sẽ biến mất và chỉ còn màu của nước biển. Hay nói theo ngôn ngữ NetLogo, khi một tác tử turtles di chuyển qua một tác tử patches có màu xanh rong biển, nó sẽ set màu lại thành màu nước biển.

Vì tảo biển bị ăn sẽ biến mất nên ta cần có hàm để thực hiện việc các tảo biển sẽ sinh sôi trở lại để cung cấp cho cá ăn.

Hàm regrow-seaweed:

to regrow-seaweed

ask patches [

if random 100 < 3 [

set pcolor green

]

]

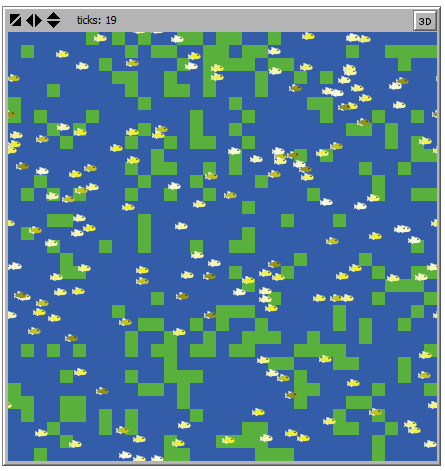
End

Hàm regrow-seaweed thực hiện như sau :

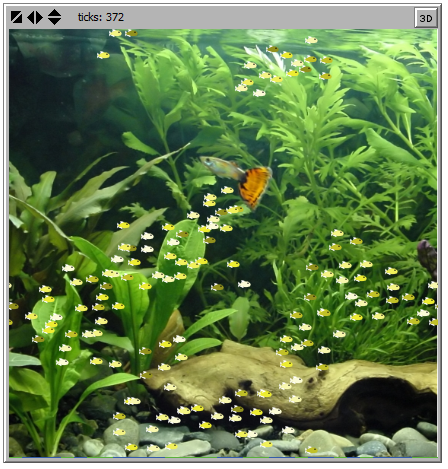
Hàm này có tác động đến các patches, sẽ random một giá trị nào đó, nếu nhỏ hơn 3, nó sẽ mọc lại tảo.

Hàm thay đổi môi trường, hàm này sẽ biến môi trường thành một bể cá với nhiều tác tử. Việc này đơn giản là chúng ta sẽ cập nhật một màu nền cho mô hình. Ví dụ như sau :

Mô hình chạy bình thường :



Còn bể cá khi có hình nền:



Tài liệu tham khảo :

Trang web : <http://ccl.northwestern.edu/netlogo/docs/>

Slide bài giảng môn hệ thống tác tử của TS. Cao Tuấn Dũng.