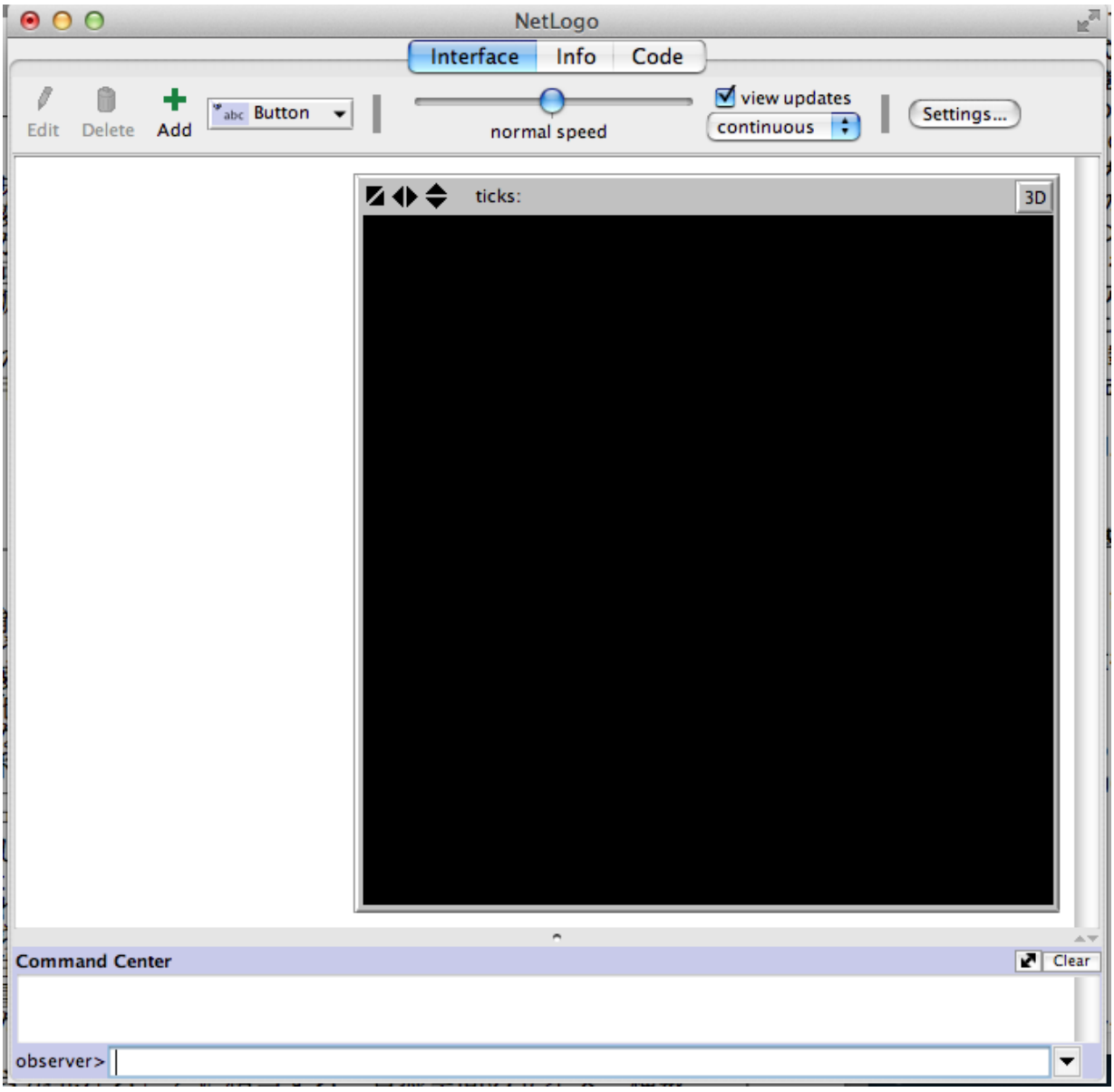
NetLogoの基礎

コマンドセンターで雰囲気をつかもう

NetLogoではいろんな命令を組み合わせて実行することで，人工世界を動かします．

最初はいくつかの命令を直接打ち込んで，どんな風に人工世界の住人をつくったり，動かしたりするか感覚をつかみます．画面下のCommand centerとかかれたところのさらに下，"observer>"とあるところの右にある入力欄に，命令を入力していきます．



１）タートルを一匹つくる

observer> create-turtles 1

真ん中に矢印の先のようなのが1つ現れます．これがタートルです．

２）タートルを動かす

observer> ask turtles [forward 1]

タートルを1歩前へ前進させます

foward 1は1歩前に動かす命令．ask turtles [...]はタートルに対して括弧のなかの命令を実行させるという意味．

３）タートルの向きを変える

observer> ask turtles [rt 30]

タートルを右に30度回転させます．左回転はlt

４）最初の状態に戻す

observer> clear-all

世界を起動したときの状態に戻します．

５）タートルにペンを持たせて書かせる

observer> ask turtles [pendown]

６）繰り返し実行する

observer> repeat 10 [ask turtles [forward 1]]

repeat 10 [...]は括弧内の命令を10回繰り返します．

Q）タートルをいくつか作成し，それぞれにペンで四角形を描かせてみましょう．他の多角形はどうでしょうか．
ヒント：括弧の中には，複数の命令を続けて並べて書くことができます．なので．．．

７）パッチ（フィールドを構成するマス）の色を変える
observer> ask patches [if (random 100 < 5) [set pcolor red]]

各パッチを確率5%で赤色にする

とか，タートルがいる状態で，

observer> ask turtles [ask patches in-radius 2 [set pcolor red]]

各タートルに自分の周りを塗りつぶさせる

８）（ペンを降ろしたタートルがいる状態で）赤いパッチに来たら30度回転，で前進

observer> repeat 100 [ask turtles [if (pcolor = red) [rt 30] forward 1]]

例題１：ラングトンのアリ

ここからは簡単なモデルを題材にして，プログラムを順番に組み立てていきたいと思います.

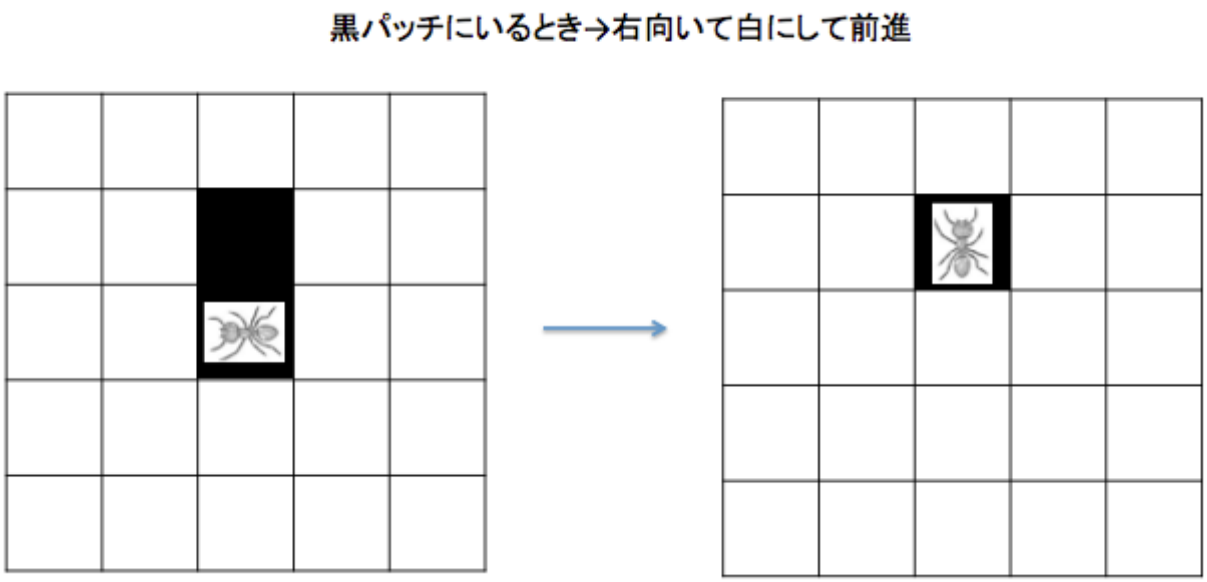
最初のお題は”ラングトンのアリ”です.

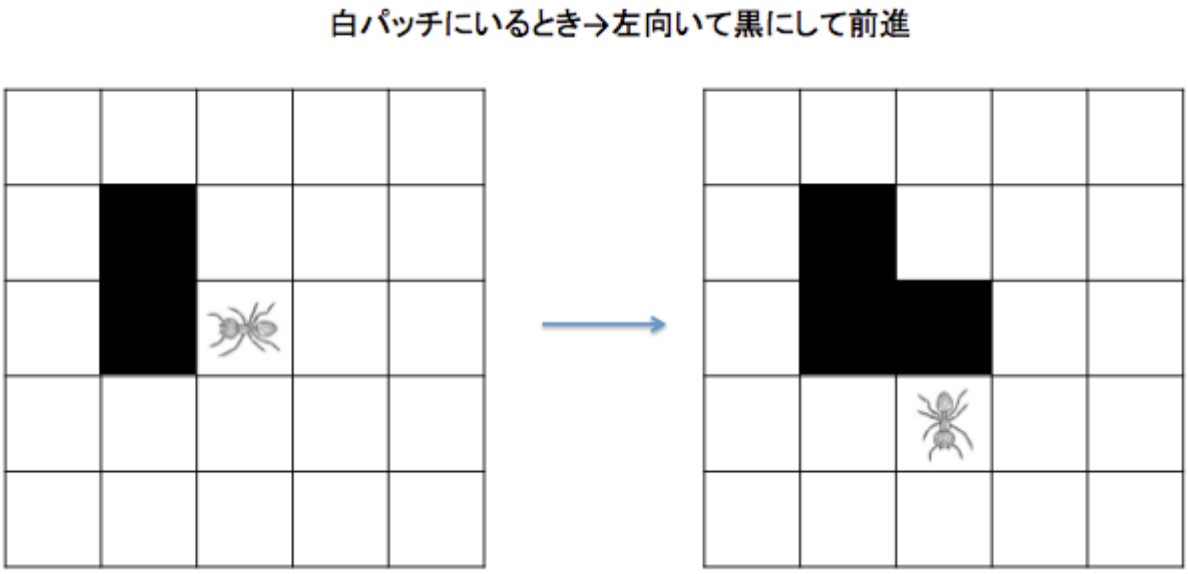
ラングトンとは，このモデルを提案した人工生命研究の創始者の名前です．一人工生命研究者として敬意を払って，また，シンプルでありながらNetLogoの 特徴を反映した良い題材として，このモデルを取り上げます.

具体的には次の様なモデルです．なお，表現はNetLogoでのものに合わせてあります.

- 沢山の四角いパッチ（マス）からできたフィールド上に一匹のアリ（タートル）を置く．最初は，東西南北（0，270，180，90度）のうちいずれか の方向を向いているものとする．各パッチは白または黒の色をとり，最初は全て黒色の状態にあるとする．各ステップでアリは次の通りに行動する.
- 黒いパッチにアリがいる場合，90度右に方向転換し，そのパッチの色を白にして，1だけ前進する.
- 白いパッチにアリがいる場合，90度左に方向転換し，そのパッチの色を黒にして，1だけ前進する.

アリの動きは，たった二行です．たったこれだけの命令を繰り返すだけで，アリはどんな振る舞いを示すか，早速NetLogoで試してみましょう.

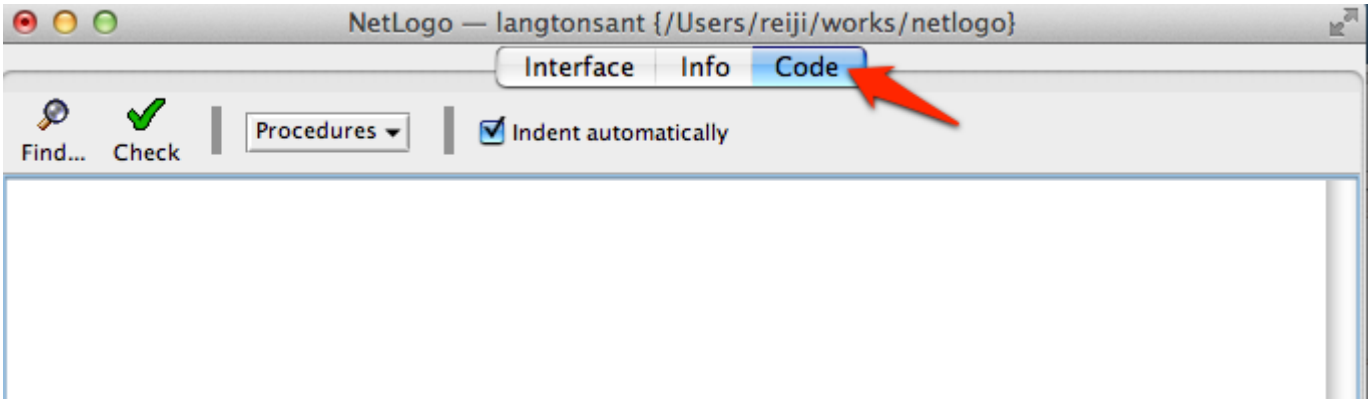




セットアップとプロシージャ

ほとんどのマルチエージェントモデルは，まずモデルの初期設定を行います．前節では命令を一行一行直接入力していましたが，今回は何度も繰り返し実行できる ように，ソースコード（とインターフェイス）として記述します．

NetLogoウィンドウ上にある，"Code"というタブをクリックすると，以下のような画面が現れます．

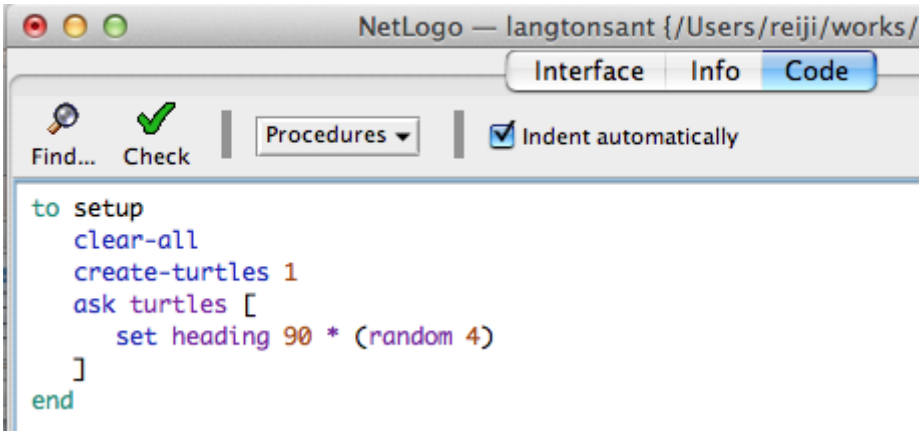


真っ白なテキスト画面です．ここに，具体的にはある機能を持った一連の命令をまとめた，プロシージャと呼ばれる関数を記述します．以後，この画面をコード画 面と呼び，元のフィールドが表示された画面をインターフェイス画面と呼ぶことにします．プロシージャは以下の形式で記述します．

```
to プロシージャ名
  命令１
  命令２
  ...
end
```

例えば，ラングトンのアリの初期設定を行う，"setup"という名前のついたプロシージャの例を示します．

```
to setup
  clear-all
  create-turtles 1
  ask turtles [
    set heading 90 * (random 4)
  ]
end
```



2行目の"clear-all"はNetLogoの状態を全て初期化する命令です．3行目の"create-turtles タートル数"は前節にも出てきたタートルを新たに作成する命令です．4-6行目では，タートル全体にカギ括弧内の命令を実行させるask turtles [...]命令を，中身がわかりやすいように改行を入れて書いたものです．その中身の5行目では，変数に値を設定するいかの形式のset命令が実行されています．

set 変数名 値

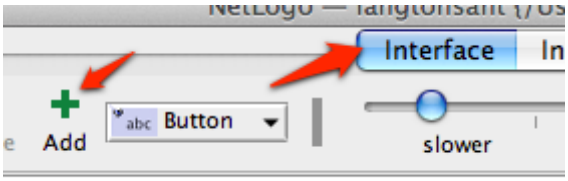
上の例では，タートルがあらかじめ持っている，自身の向いている向きを0〜360で表現したheadingという変数の値を変更しています．変更する値"90 * (random 4)"の中の"random 値"という命令は，0から値より小さい整数の中からランダムに1つの整数値を返すものです．したがって，headingの値はそれらを90倍し た，0，90，180，270のうちのいずれか1つの値が入ることになります．

このようにして定義したプロシージャは，プログラム中で従来の命令を同様に利用することができます．例えば，ウィンドウ上の"Interface"タブをク リックしてインターフェイス画面を出し，コマンドセンターにsetupと入力する度に，フィールドの真ん中に東西南北ランダムな向きを向いたタートルが現れる はずです．

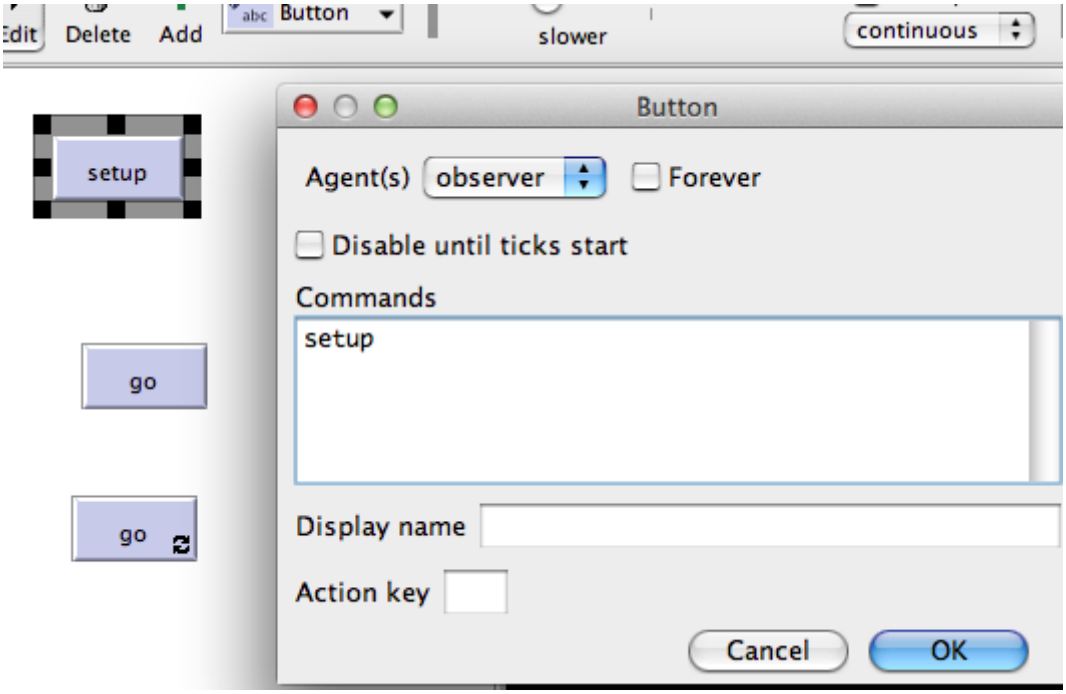
セットアップボタンの作成

初期化の作業はモデルで実験する上で何度も利用するので，ボタン1つで実行できると便利です．NetLogoには，よく使う命令を実行するボタンなどの様々 なインターフェイスを作成することができます．ここではsetupプロシージャを実行するボタンをつくってみましょう．

インターフェイス画面を出します．ウィンドウ上部に，小さく"Button"と書かれたボタンの隣にある”＋マーク下にAdd"と書かれたボタンをクリックし ます．



ボタンが押された状態になったまま，マウスカーソルをフィールドの横の空白の部分に動かすと，カーソルが十字になります．この状態でクリックすると，その場 所にボタンが作られます．すぐにボタンの設定を行うダイアログが出ます．ここで，Commandsというテキストボックスの中にボタンを押したときに実行した い命令を記述します．この場合，"setup"とだけ書いて，OKボタンを押します．



できあがったボタンにもsetupと書かれていて，ボタンを押す度に向きの異なるタートルが現れることを確認しましょう．このようなモデルの初期化を行うプロ シージャとボタンは，setupと名付けるのがNetLogoでは一般的なようです．

アリを動かす

いよいよアリを動かします．setupプロシージャの下に続けて，モデルを1ステップ分動かすためのプロシージャを名前をgoとして以下の様に定義します．

```
to go
  ask turtles [
    ifelse(pcolor = black)[
      set pcolor white
      rt 90
    ][
      set pcolor black
```

```
    lt 90
  ]
  forward 1
 ]
end
```

日本語で言い換えると．．．

```
ここからgoというプロシージャの定義はじまり
  次の事をそれぞれにタートルにお願いします [
    もし自分が乗っているパッチの色が黒だったら次の事をしてね[
      今自分が乗っているパッチの色を白にしてね
      右に90度回転してね
    ]それ以外の場合は次の事をしてね[
      今自分が乗っているパッチの色を黒にしてね
      左に90度回転してね
    ]
    一歩前に進んでね
]
goの定義おわり
```

ask turtles[...]はカギ括弧内の命令を全てのタートルに行わせる命令です（ただしこの場合は一匹のみ）．その中身は大部分がifelse文で構成されています．ifelse文は以下の書 式をとります．

```
ifelse (条件式) [
  条件式を満たした時に実行する命令
][
  条件式を満たさなかったときに実行する命令
]
```

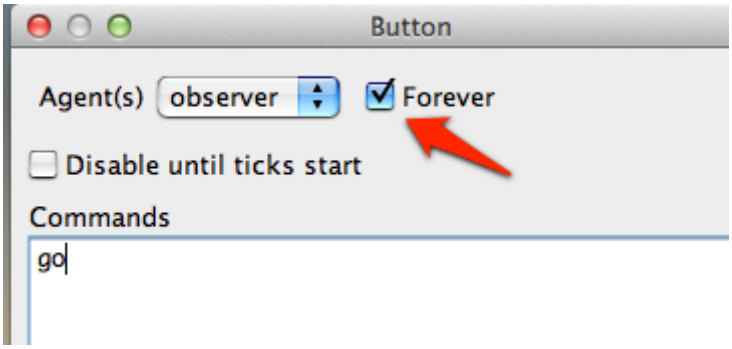
この場合，条件式には(pcolor = black)が入っています．pcolorはパッチの色を表す変数ですが，このように，タートルに命令を実行させる場合，pcolorはそのタートルがいま乗っているパッ チのpcolorを表します．つまり，"pcolor = black"とは，"そのタートルが乗っているパッチの色が黒色である"という条件を表すものです．この条件を満たした場合，今乗っているパッチの色を白色にし（set pcolor white），右に90度向きを変える（rt 90）という一連の命令を実行します．条件式を満たさない場合，つまり，パッチの色が白色の場合も同様にして処理を記述していることを確認しましょう．最後に，どちらの パッチの色にいた場合にも共通した一歩進む（forward 1）処理を行います．

モデルの実行

setupプロシージャと同様の手順で，goプロシージャを一回実行するボタンを作成し，ステップを繰り返すごとにタートルがどのように動くか様子を見てみ ましょう．アリはフィールドに模様を描きながら動き回る様子が見えるでしょう．アリは上のシンプルなルールに従うにもかかわらず，その挙動は大変複雑に見えま すね．

では，これをもっとずっと繰り返したらどうなるでしょう．毎回ボタンをポチポチするのは大変

なので，goプロシージャを繰り返し自動で実行してくれるボタンを 新たに作成します．先ほどのgoボタンと同様にして新しくボタンをもう一つ作りますが，ボタンの設定を行うダイアログの上にある"Foever"というチェッ クボックスをオンにした上で，OKボタンを押します．できあがったボタンの名前は同じくgoですが，右下にくるっと回った矢印がついているのがわかると思います．これはダイアログで指定した一連の命令，この場合はgoプロシージャを，ボタンが押された状態にある間繰り返し実行し続けることを示します．このボタンを 繰り返しボタンと呼ぶことにします．

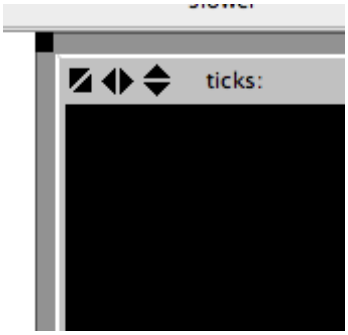


新しくつくったgo繰り返しボタンをクリックすると，ボタンが押された状態になり，アリがせっせと動き続けます．もう一度ボタンを押すと止まります．

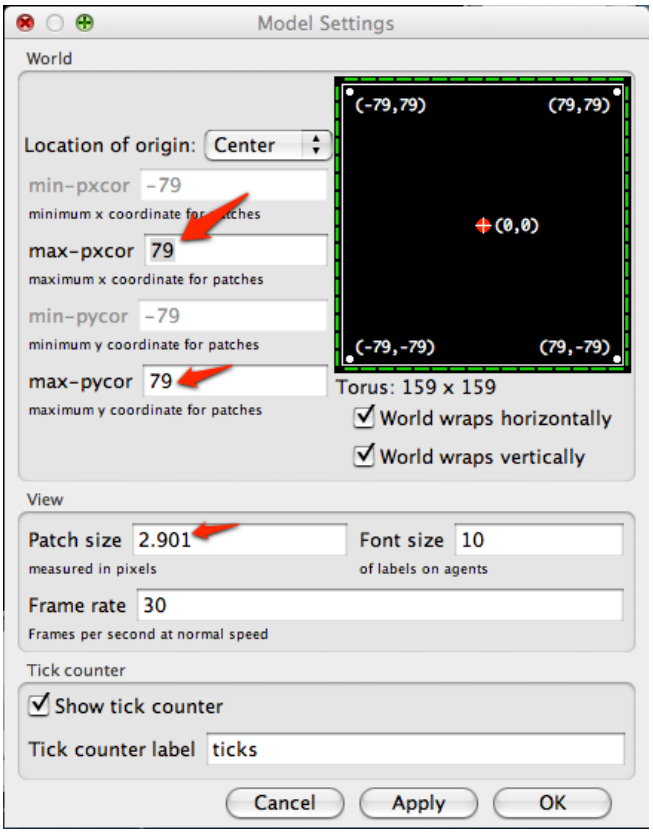
Q）さて，ずーっと動かすと・・・どうなるでしょう．

おそらく初期のフィールドの大きさだとパッチが荒いので，フィールド画面左上の斜め矢印ボタンの右下の部分を何度か押して，パッチの数をふやしてみるとよい と思います．

ただしそのまま増やすとフィールドが大きくなりすぎるので，パッチのサイズを変えて全体を縮小します．白い領域からドラッグを開始してフィールドに触れると 以下の様に選択状態になります．ここで，角の黒い四角をドラッグすると，サイズが変更できます．



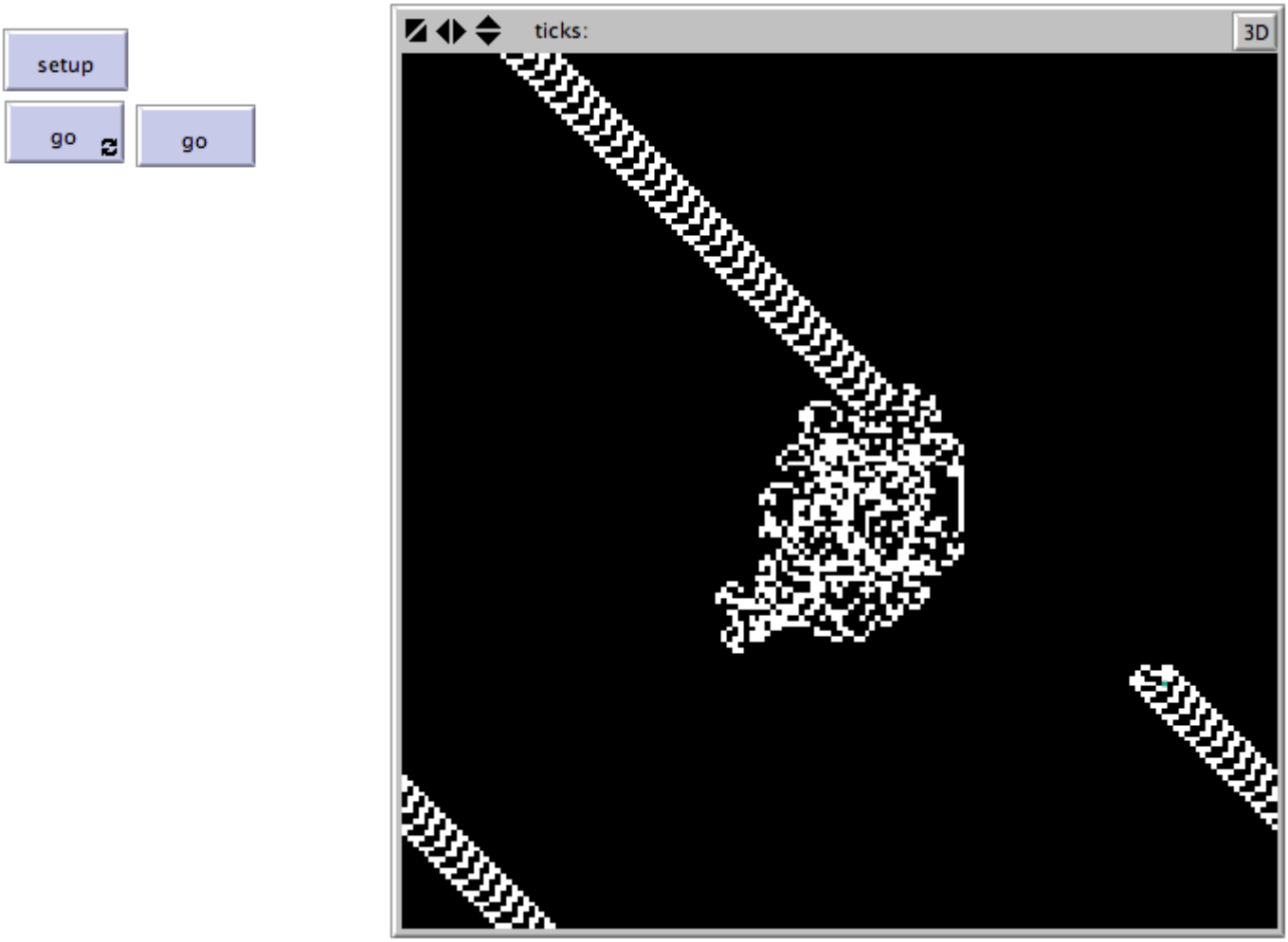
もしくは，上の状態でフィールドをダブルクリックして以下の画面をだし，max-pcorとmax-pycorを任意の縦横のパッチ数，patch sizeを任意のパッチの大きさに指定します．



Q）いろいろ変えてみて，その結果を調べましょう.

- パッチの色を変える.
- アリを複数にしてみる.
- 一歩の歩幅を変えてみる.
- 方向転換の角度を変えてみる.
- パッチの種類を増やしてみる.
    - 白，黒に加えて黄色いパッチも加える.
    - 訪れたパッチの書き換え方を"黒→白→黄→黒．．．”とする
    - 訪れたパッチごとに回転の仕方を"黒→右90，白→左90，黄→右90"とか（ifelse文を入れ子にして使うとできます）
- とか．．．

[完成したソースコード]



例題２：鬼ごっこ

次は，平和な青鬼の村に赤鬼が現れて襲う設定を考えます.

エージェントの種類の設定

青鬼と赤鬼という2種類の住人がいることを考えるので，まずそれぞれに名前をつけます.

NetLogoのウィンドウの"Code"の部分をクリックしてください．真っ白なテキストフィールドが出てきます．NetLogoではここにプログラムを 書いていきます.

以下の2行を書き加えましょう.

```
breed [aoonies aooni]
breed [akaonies akaoni]
```

これは，エージェントの種類として，青鬼aooni（複数形aoonies）と赤鬼を表すakaoni（複数形akaonies）を定義するものです.

セットアッププロシージャ

次に，例によってプログラムの初期設定をする部分をつくります．

例えば，”初期化して青鬼を100人つくってその色を青色にしランダムな場所におく”，setupという名前の関数をつくるには，次の通りにします．

```
to setup
  clear-all
  create-aoonies 100
  ask aoonies [
    set color blue
    setxy (random world-width) (random world-height)
  ]
end
```

create-aooniesはcreate-turtlesと同じで，青鬼を100人つくる命令です．
ask aoonies[...]は青鬼に対してのみ括弧内の命令を行います．
colorは各タートルの色を表す変数で，それをblueに設定しています．
setxyの一行は，青鬼をフィールド上のランダムに位置に移動させる命令です．

ラングトンのアリと同様にして，setupボタンを作ります．

押すとsetupが実行されるのを確認しましょう．

世界を動かす

では次に，この世界を動かすプロシージャをつくっていきます．再び，この世界で時間が1つ進む間に実行される命令をまとめたgoという名前のプロシージャを つ くりましょう．

最初はただ単純に，青鬼を少し進めるだけにします．以下をCodeのテキストフィールドのsetupプロシージャの下に続けて書きます．

```
to go
  ask aoonies [
    forward 0.5
  ]
end
```

つぎに，setupプロシージャと同様の手順で，goプロシージャを一回実行するボタンを作成し，ステップを繰り返すごとにタートルがどのように動くか様子 を見てみましょう．この世界は右端と左端，上端と下端がつながっていることがわかります（トーラスといいます）．

go繰り返しボタンもつくって押すと，ボタンが押された状態になり，青鬼が歩きつづけます．

これで，とりあえずの土台ができました．

赤鬼登場

ではいよいよ，赤鬼に登場してもらいましょう．

今回は，赤鬼を投入するボタンを作ってみます．

setupボタンを作ったときと同様に，まず，以下のakaoni-tounyuプロシージャをつくり，それを実行するボタンを作りましょう．

```
to akaoni-tounyu
    create-akaonies 1 [
     set color red
     setxy (random world-width) (random world-height)
   ]
end
```

例によってcreate-...を使っていますが，その後のかぎ括弧は，今つくったエージェントに対してのみ行われる命令です．ここでは，まず色を表す color変数をredにしています．そして赤鬼の出現位置をランダムにするために，setxy命令を使ってフィールド上のランダムに位置に移動させています．

次に，赤鬼のふるまいを決めましょう．赤鬼は半径3以内に青鬼が一人でもいたら，そのうち一人の方向を向き，もし，半径1以内に青鬼がいたら，そのうち一人 を食べてしまうことにします．

具体的には，goプロシージャに以下の赤色の部分を追加します．

```
to go
  ask aoonies [
    forward 0.5
  ]

 ask akaonies [
   if (any? aoonies in-radius 3)[
      face (one-of aoonies in-radius 3)
   ]
   if (any? aoonies in-radius 1)[
      ask (one-of aoonies in-radius 1) [
        die
      ]
   ]
   forward 0.5
 ]
end
```

だんだんややこしくなってきましたが，前半は最初のものと同じです．
後半の最初は，赤鬼に対して，"もし半径3以内に青鬼がいたら，その中の誰か一人の方向を向く”ということをしています．最後は，"もし半径1以内に青鬼がいた ら，その中の誰か一人を死亡させる（取り除く）"ことを表しています．

赤鬼を投入すると，次第に青鬼はどうなるでしょう．

鬼の数を調べる

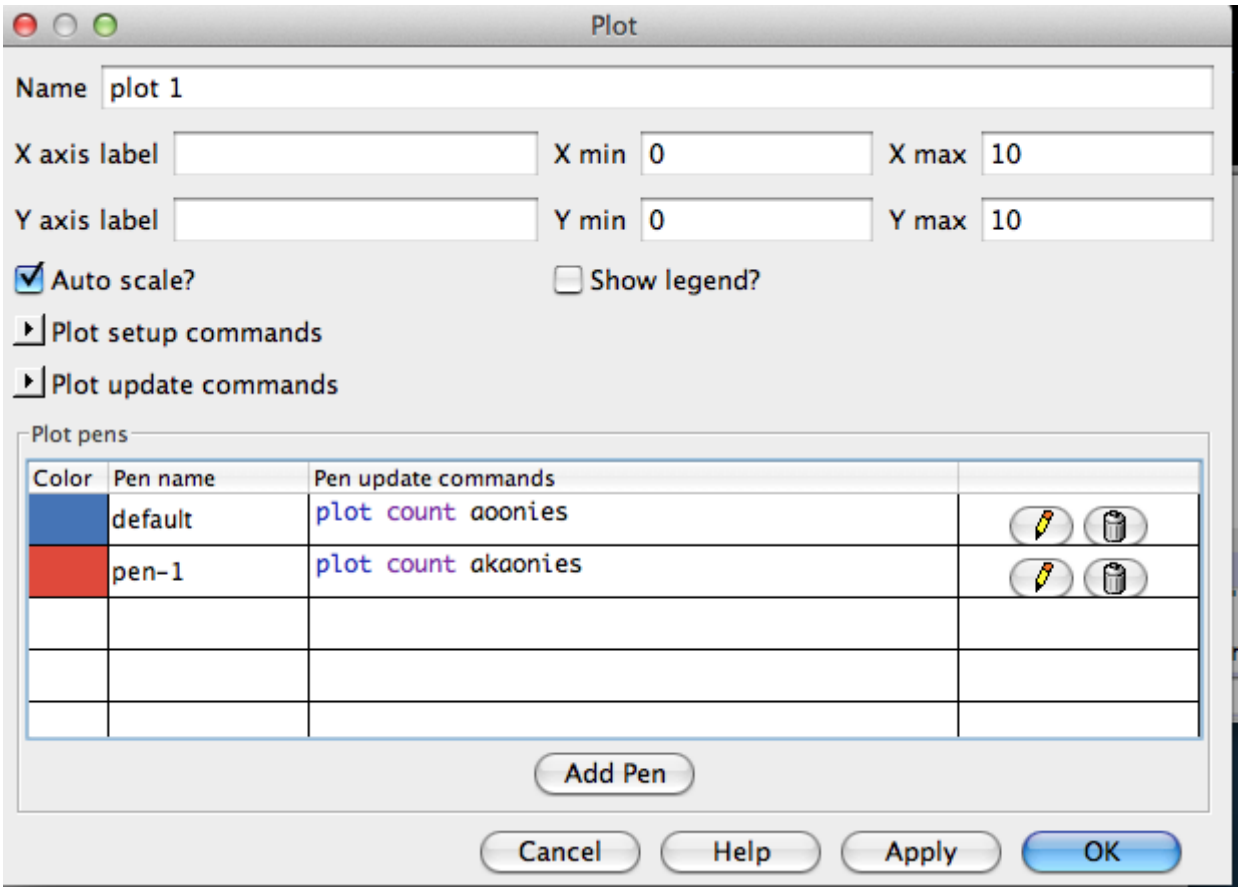
どの鬼が今何人いるか，その推移をグラフにすることができます．

インターフェイス画面を出します．ウィンドウ上部に，小さく"Button"と書かれたボタンの隣にあ

る”＋マーク下にAdd"と書かれたボタンをクリッ クします．下に向かってリストが出るので，そこから7番目のPlotを選びます．

ボタンが押された状態になったまま，マウスカーソルをフィールドの横の空白の部分に動かすと，カーソルが十字になります．この状態でクリックすると，その場所にプ ロットと呼ばれるグラフが作られます．すぐにプロットの設定を行うダイアログが出ます．

そこで，以下の様に設定します（注意：Add penを一回押してペンを2つにします．色はダブルクリックしてパレットを出し選びます）．



その後，
setupプロシージャの最後（endの1つ上）に

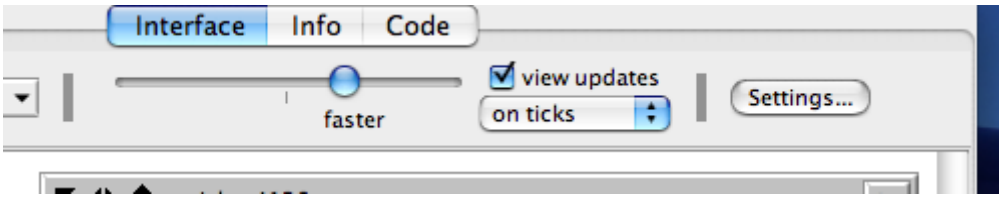
reset-ticks

を追加し，goプロシージャの最後（endの1つ上）に

tick

を追加します．

なお，上の設定後，フィールド画面を出したときに上にある"continuous"と表示されたボックスから"on ticks"を選択すると，フィールド上の エージェントの数に関係無く一定速度でgoが実行されます．



青鬼は逃げる

生存の危機が迫っているのに，青鬼はただぼーっと歩いているのは変ですね．上のgoプロシージャで赤鬼が青鬼を追いかけるのに習って，各青鬼は”もし半径3 以内に赤鬼がいたら，その反対の方向を向く"ことにしてみましょう．

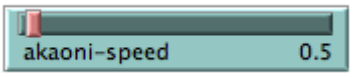
Q）青鬼は赤鬼からうまく逃げられますか？

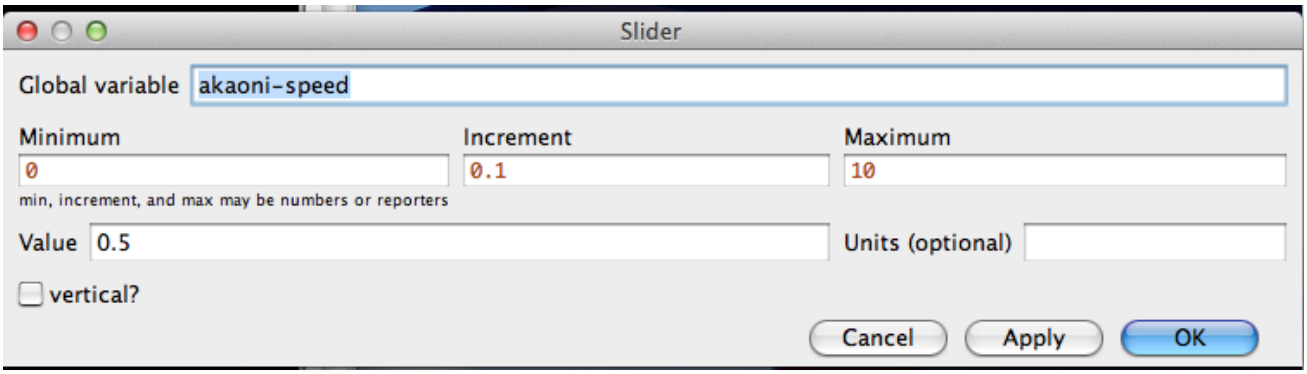
足の速さをかえてみる

赤鬼の足の速さを簡単に調整できる様にしてみましょう．

インターフェイス画面を出します．ウィンドウ上部に，小さく"Button"と書かれたボタンの隣にある”＋マーク下にAdd"と書かれたボタンをクリッ クします．下に向かってリストが出るので，そこから2番目のSliderを選びます．

ボタンが押された状態になったまま，マウスカーソルをフィールドの横の空白の部分に動かすと，カーソルが十字になります．この状態でクリックすると，その場所にス ライダが作られます．すぐにスライダの設定を行うダイアログが出ます．





ここで，以下の様に設定してOKします．

- Global variable：akaoni-speed
- Increment：0.1
- Maximum: 10

そのあと，goの中の赤鬼を一歩進ませる

forward 0.5

の部分を,

forward akaoni-speed

に書き換えます.

プログラムを実行している最中にスライダを調整すると，赤鬼のスピードがそれに応じて変化するはずです.

つまり，スライダの現在の値が常にakaoni-speedという名前の変数で表されていて，それをfowardの命令に使っているということです.

Q）赤鬼の足の速さを変えると，鬼たちの数の変化の仕方に影響はありますか？

青鬼は自然に増える

ここは青鬼の村なので，青鬼は自然に増えることを考えます．ごく簡単に，毎ステップに一人青鬼が増えることにしてみます．goの一番最後（tickの前）に以下を 記述します.

```
create-aoonies 1 [
  set color blue
  setxy (random world-width) (random world-height)
]
```

赤鬼は青鬼を食べると増え，ほおっておくと餓死する

一方，赤鬼は青鬼を食べると増えることにしましょう.

goの中の赤鬼が青鬼をつかまえたところに以下の赤字の命令を追加します.

```
if (any? aoonies in-radius 1)[
    ask (one-of aoonies in-radius 1) [
      die
    ]
    hatch 1
]
```

具体的にはhatch 1を加えただけです．これは，自分のコピーを一人その場に産むという命令です.

また，赤鬼はほおっておくと餓死することも加えましょう．例えば，毎ステップ各赤鬼が20％の確率で死亡するには，以下をgoの一番最後（tickの前）に記述し ます．100面ダイスを振って値が20以下だったら死ぬようにします.

```
ask akaonies [
  if((random 100) < 20)[
    die
  ]
]
```

Q）青鬼と赤鬼が長く共存するのはどんな設定の時か，いろいろいじって探してみましょう．また，その仕組みも考えてみましょう．（設定の値をいちいちcodeを書 き換えて変更するのは面倒だとおもいませんか？赤鬼の速度と同様にスライダを使って変えてみたい値も操作できるようにしたら便利ですね．）

できあがったソースコード