

第 3 回 エージェントをうごかす! (10月21日)

概略

前回の課題 (千鳥足、こちゃごちゃ、forward 以外の歩くルール)

if 文のいろいろ

変数の型、種類

宿題

前回の宿題

[7] の回答例

```
Agt_Init{
  my.X = 25
  my.Y = 25
}
Agt_Step{
  my.Speed = rnd()
  my.Direction = rnd()*60
  forward(my.Speed)
}
```

Speed は
ランダムに

Direction を千鳥足っぽく 60 度の範囲で指定します。

[8] の回答例

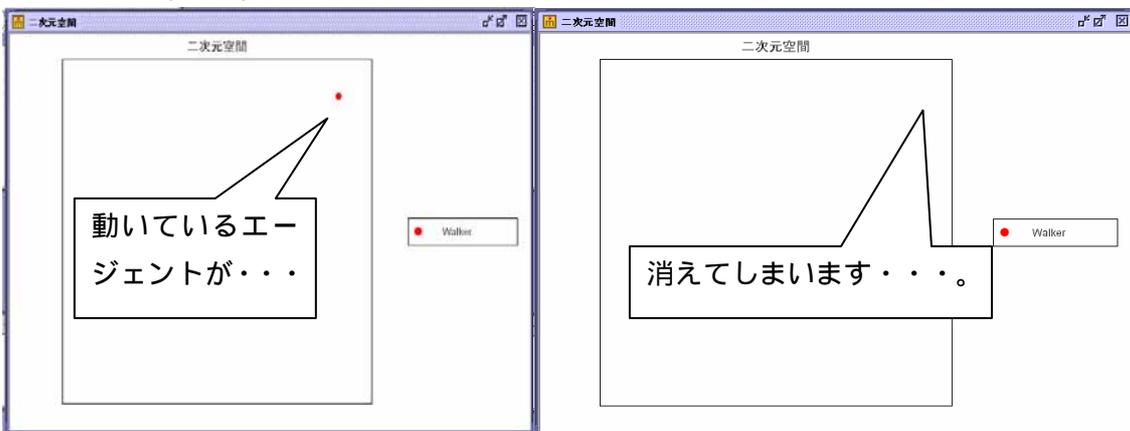
```
Agt_Init{
  my.X = rnd() * 100
  my.Y = rnd() * 100
  my.Direction = rnd() * 360
  my.Speed = rnd() * 10
}
Agt_Step{
  forward(my.Speed)
}
```

自分の位置を 0 ~ 100
の範囲で指定します。

[9]の回答例

```
Agt_Init{
}
Agt_Step{
my.X = my.X + 1
my.Y = my.Y + 1
}
```

>ただし、my.X,my.Y とともにどんどんと値が追加されるだけなので、ループはしません。



if 文のいろいろ

基本の if 文 「もし～～だったら、 下さい」

```
if [ 条件式～～ ] then
    [ 実行したいルール ]
end if
```

基本的な if 文の書き方です。エージェントにさまざまな判断をさせるときに用います。例えば、お金のない人とは友達にならないエージェントの行動は、以下のようなルールをステップ (Agt_Step) に書くことで表現できます。

```
if < 友達の所持金 == 0 > then
    < つきあわない >
end if
```

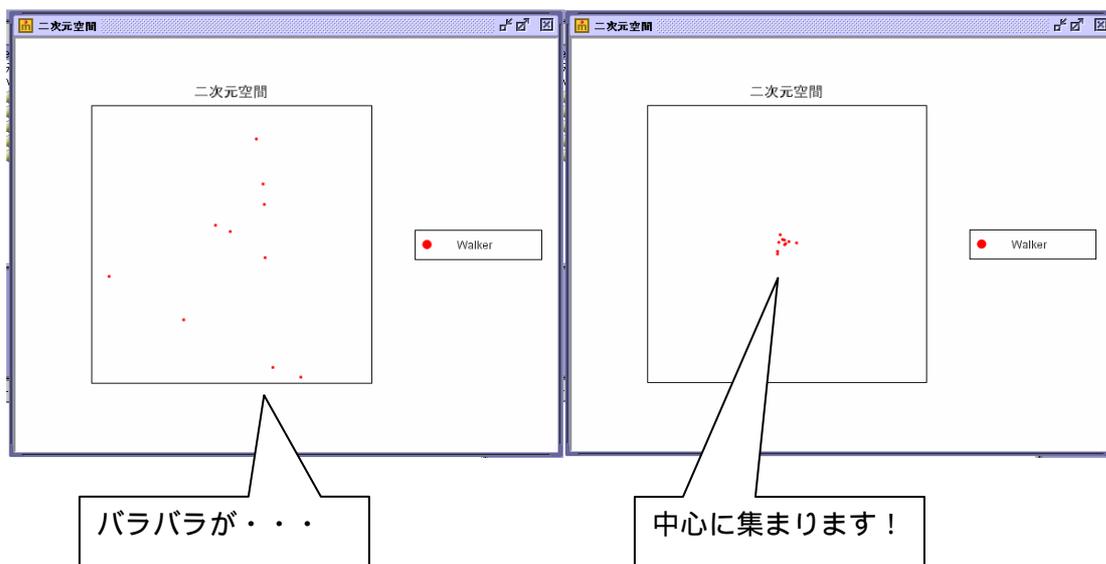

ちなみに、//に続く文を「コメント」と言い、ルールには反映されません。後にルールエディタを見て何をしているのかわかるようにコメントを書くのがプログラムのコツです。

Movetocenter()という関数は、どんな大きさの二次元空間でも、真ん中にエージェントを配置してくれる便利な関数です。覚えておきましょう。

> このモデルはなかなかおもしろい動きをします。

ある一定のステップが経過すると、エージェントたちは中心に集まってくるのです！

なぜかは各自で考えてみてください。



elseif 文

「もし～1～だったら、<1>しなさい。
そうでなくもし～2～だったら、<2>しなさい。
もしそうでなかったら、<3>しなさい。」

```
if [ 条件式～1～ ]      then
    [ 実行したいルール<1> ]
elseif [ 条件式～2～ ]  then
    [ 実行したいルール<2> ]
else
    [ 実行したいルール<3> ]
end if
```

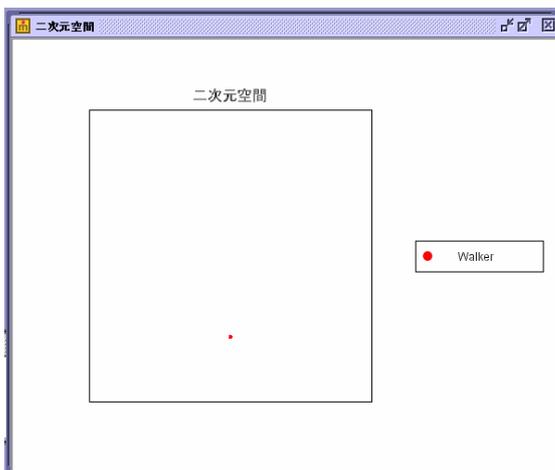
等号と不等号

if文と一緒によく使うことになる等号不等号についてまとめておきます。

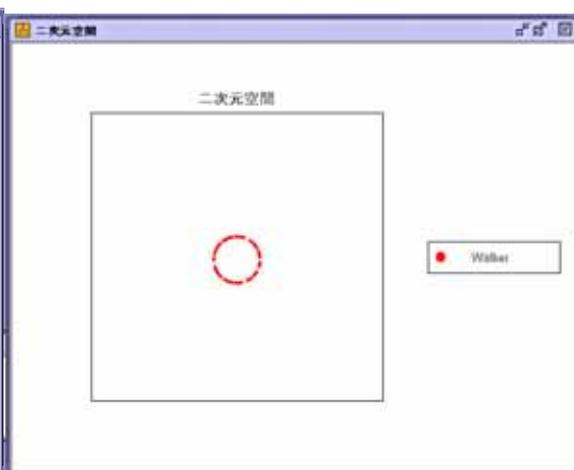
==	等しい
<>	等しくない
!=	等しくない
<	より小さい
<=	より小さいか等しい
>	より大きい
>=	より大きいか等しい

それではまたまた例題です。

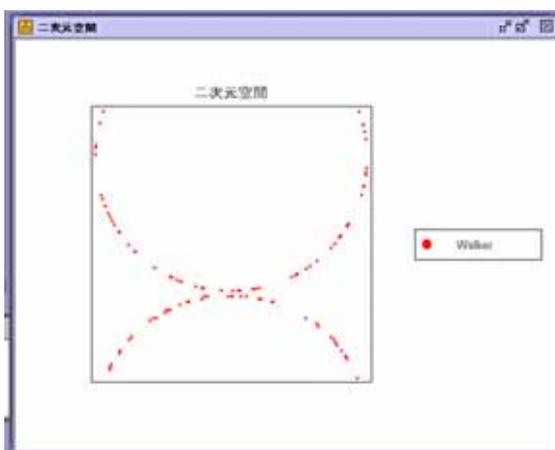
[2][1]で作った空間そのまま、エージェント数も 100 とする。X 位置が真ん中、Y 位置は一番下から真上に一步ずつ進み、50 歩進めばランダムに色々な方向へ動く。さらに 100 歩進めばまた真上に一步ずつ進む行動に戻る。



まずは一丸となって真上に進みます。



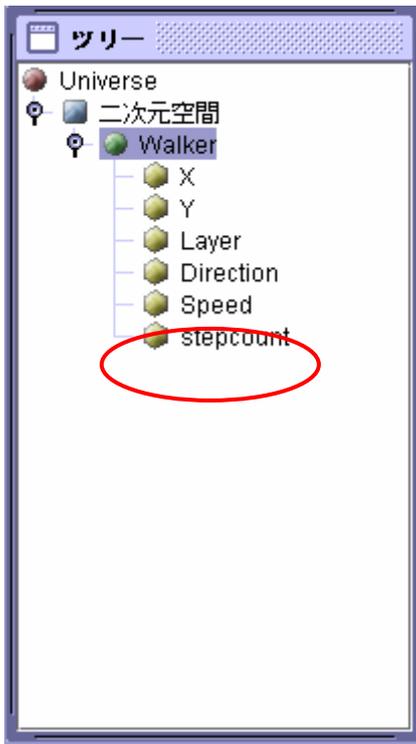
真ん中で輪になるように広がります。



全体に広がりきると、再び真上に進みます。

> 下記のように作成します。

まず、ステップ数をカウントするために”stepcount”という変数を追加します。変数の型は整数型にしておいてください。ルールは下記のように記述しますが、これが唯一の答えではありません。その他にも色々やり方がありますので、余裕のある人は考えてみてください。



```
Agt_Init{
// 初期設定をする
my.X = 50
my.Y = 0
my.Direction = 90
}
Agt_Step{
//ステップを数える
my.stepcount = my.stepcount + 1
//条件分岐
If my.stepcount == 100 then
my.Direction = 90
elseif my.stepcount == 50 then
my.Direction = rnd()*360
else
forward(1)
end if
}
```

ちなみに、ステップ数を数える便利な関数”GetCountStep”も存在します。慣れたらそちらを使うと良いでしょう。

変数の型

変数は、「型」を必ず持っていて（変数を作る際に設定させられましたね）、あらかじめどのようなタイプの値をとるのかを限定しておく必要があります。最も、イメージしやすく、一般的なものは、「整数型」や「実数型」ですが、以下のようなさまざまな型の変数を設定することができます。

型の種類	正式名称	中に入るもの	具体的な値
ブール型	Boolean	真偽	True, False
文字列型	String	文字列	Tanaka, Sakamoto
整数型	Integer	整数	1,2,3,365
長整数型	Long	大きな整数	87799990000387772...
実数型	Double	実数	3.1415..., 1.1415...
空間型	Space	空間名	SpaceK, 空間
エージェント種別型	AgtType	エージェントの種類	赤亀, 青亀
エージェント型	Agt	エージェント	赤亀 01, 赤亀 02
エージェントセット型	AgtSet	エージェントの集合	{ 赤亀 01, 赤亀 02 } { 赤亀 00, 青亀 01 }

今日は上の5つについて、把握しておいてください。

とりうる値の限界については、ヘルプを参照にしてください。



変数の種類（ツリーの中の変数、ルールの中の変数）＜重要！＞

変数には、大きく2種類のものがあります。一つは、ツリーに表示される『正式な』変数です。前回、作成した「Speed」という変数はこれに該当します。もう一つはルールの中で定義される『一時的な』変数です。これは、ルールを記述するうえで必要に応じて作成される変数で、ツリーには表示させずに用います。

正式な変数：

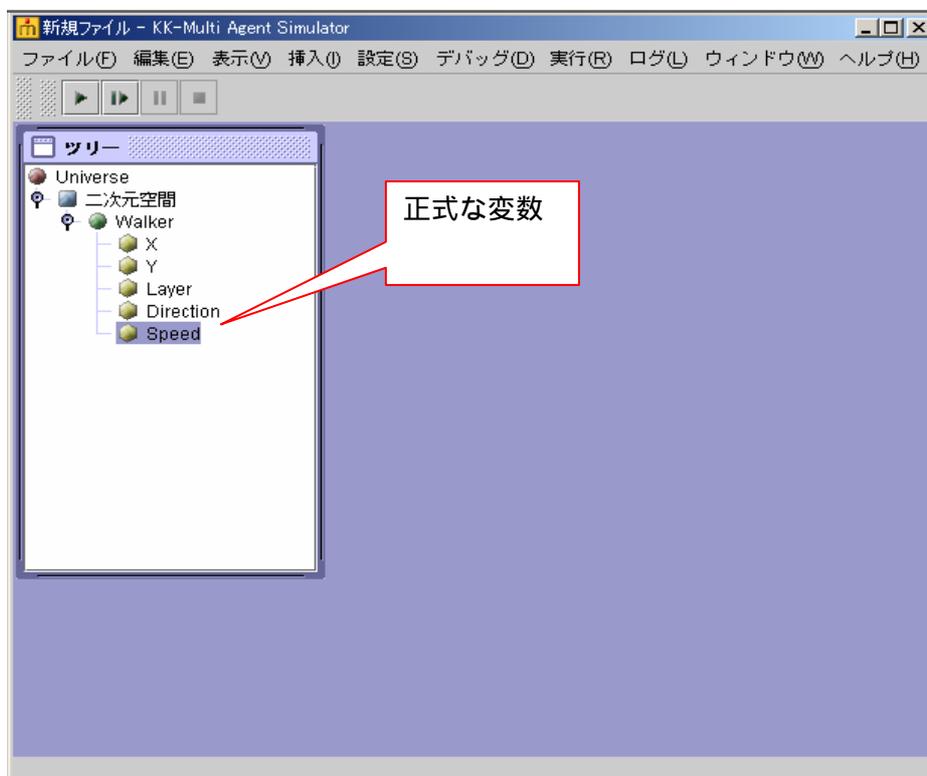
ツリーの中で定義します。それぞれのエージェントが、それぞれ異なった値を持ち歩かなければならないようなタイプの変数にはこちらの変数を定義してやります。

一時的な変数：

ルールの中で「dim (変数名) as (変数の型)」の形で定義します。変数をルールの中で使ったあと、その値を保存する必要のないようなタイプの変数にはこちらの変数を定義してやります。

両者の最大の違いは、「正式な変数」はそれぞれのエージェント（例えば赤亀）が、その変数のそれぞれの値を恒常的に備えるのに対し、「一時的な変数」はそのとき処理しているエージェントについてのみ、その変数の値が設定され、処理の順番が次のエージェントにうつった時点で、その値は初期化されてしまうという点にあります。それぞれに特徴があるので、うまく使い分ける必要があります。書いたルールは、たくさんいる全てのエージェントに適用されるものであることを頭にいれておいて、値の保存の必要があれば、ツリーの中に変数を設定し、不要であればルールの中で定義してやってください。

『正式な』変数においても、『一時的な』変数においても型は、必ず必要です。



では早速例題をやってみましょう。

[3][1]で保存したモデルを改良して、10%の確率で、自分の方向とスピードをランダムに設定し直す歩行者たち。

> 以下のように記述します。

```
Agt_Init{
// 初期設定をする
My.X = 50
My.Y = 50
My.Direction = rnd()*360
My.Speed = rnd()*5
}
Agt_Step{
dim a as double
// 前に進む
Forward(My.Speed)

// 回転方向を判断する
If my.Speed > 2.5 then
    Turn(-10)
else
    Turn(10)
end if

// 方向や速さを変更するか、どうかを判断する
a = rnd()
if a < 0.1 then
    My.Direction = rnd()*360
    My.Speed = rnd()*5
end if
}
```

最初に「一時的な」変数を定義します

定義した変数に値を代入します。

宿題

さてさて、本日も宿題を出します。以下のモデルを作成してください（空間の広さ・人数は任意）。

[4] 空間の真ん中からそれぞれバラバラの方向へ一歩ずつ動き、空間全体に広がり切ったら元の位置に戻ってくる歩行者（繰り返し）。

[5] それぞればらばらのスピードで空間の端を壁にそって回り続ける歩行者。

[6] 進むべき方角がそれぞれに決まっています（ランダムに決定する）。最初はばらばらに歩き始めるが、徐々に「自分の」進むべき方角に方向を変えていく歩行者。

<ヒント！> 「進むべき方向」の変数を新たに追加してください。