国際政治と情報（2005 年後期）

担当　田中明彦 tanaka@ioc.u-tokyo.ac.jp
TA　　阪本拓人 takutos@nifty.com
保城広至 hoshiro@ioc.u-tokyo.ac.jp

第 4 回　エージェント周りを見る！（10 月 28 日）

●概略

○前回の宿題の解説
○変数の型（エージェント型、エージェント集合型）
○エージェント集合型関数
○セル型の空間
○MoveToSpace による移動ルール
○宿題

●前回の宿題

[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）。

回答例

```
Agt_Init{
movetocenter()
my.Direction = rnd() * 360
}
Agt_Step{
 my.stepcount = my.stepcount + 1

If my.stepcount = =50   then
        my.Direction = my.Direction -180

elseif my.stepcount = = 100 then
        my.stepcount = 0
else
        forward(1)
end if
}
```

50 ステップで方向を反対に！

繰り返すためにステップカウントを初期化する

[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。

回答例

```
Agt_Init{
My.X = 1
My.Y = 1
My.Direction = 0
My.Speed = rnd()
}
Agt_Step{
Forward(My.Speed)
my.stepcount = my.stepcount + 1

If my.stepcount * my.Speed >= 98 then
        Turn(90)
        my.stepcount = 0
end if
}
```

歩数×スピードが 99 になったら左に 90 度曲がり、カウントを 0 に戻す（0 ステップから始めるので 99 だが、スピードが実数でぴたりと 99 にならない可能性があるので 98 とする）。

[6] 進むべき方角がそれぞれに決まっていて（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。

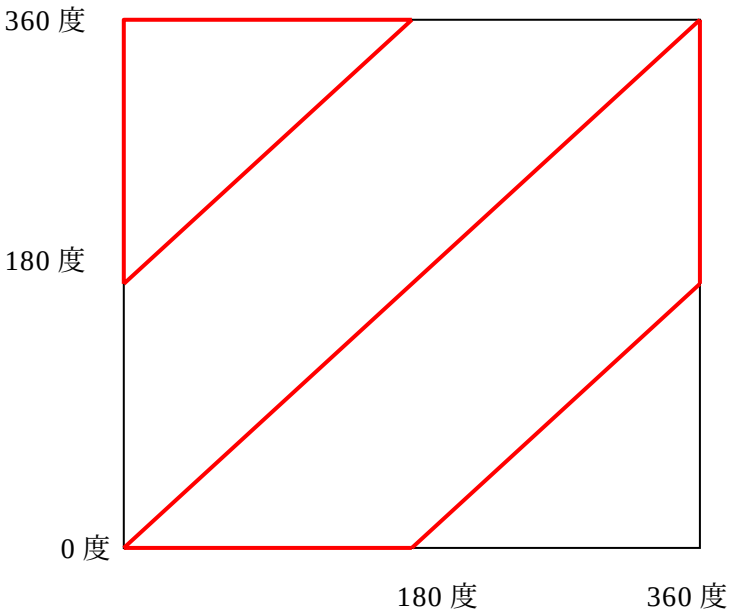
回答例

```
Agt_Init{
movetocenter()
My.Direction = rnd()*360
My.Speed = rnd()*5
My.Destination = rnd()*360
}
Agt_Step{
Forward(My.Speed)
If My.Destination> My.Direction and My.Destination< My.Direction+180   then
        Turn(10)
elseif  My.Destination< My.Direction-180    then
        Turn(10)
else
        Turn(-10)
end if
}
```

目的地変数をつくっておき、ランダムに決定する

ここのルールは、下図を参考にしてください。

進むべき方角



自分の進んでいる方角

☆赤で囲んだ箇所は、右へ曲がった方が早い（か同じ）座標位置です。この不等式を解けば上記のルールになります。

●前回まで作成したエージェントは、自分で判断していましたが、その判断は一人よがりなもので、周りの世界と相互作用はしていませんでした。今回は、幾つか新しい文法事項を学んで、周りの世界を認識する方法を学びます。

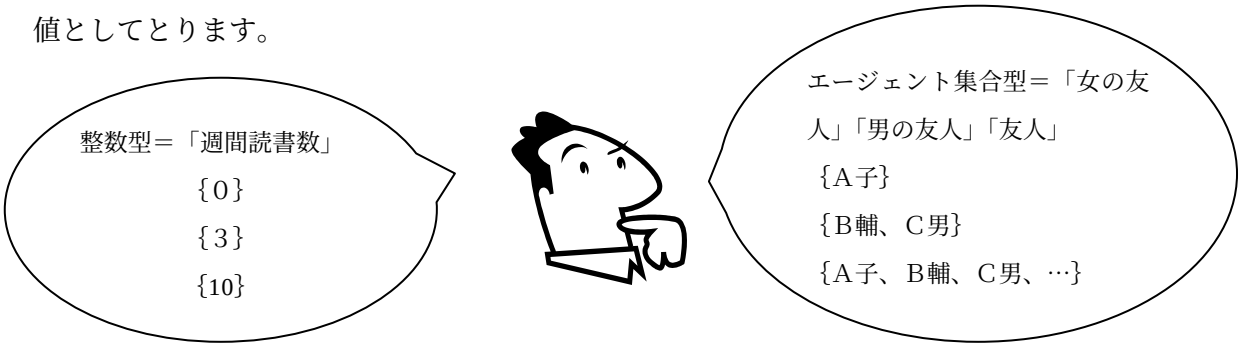
●エージェント集合型

前回、変数の型について学びました。数字を格納できる整数型と実数型、文字列を格納できる文字列型などがありました。今回はエージェント集合（セット）型の変数を学びます。

☆ エージェント集合（セット）型変数

エージェントの集合を格納できる変数です。少しわかりにくいですが、整数型の変数が、１や２といった値をとるのに対し、エージェント集合型の変数は、エージェントの集合を

値としてとります。



＞ツリーで定義するときは、「エージェント集合型」を選択してください。
（＞ルール内で定義するときは、Dim ＜変数名＞ As AgtSet としてください）

変数のいろいろ（前回と同じ表です）

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	二次元空間, 空間 X
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
エージェント集合型	AgtSet	エージェントの集合	{赤亀 01、赤亀 02}、{赤亀 00、青亀 01}

●エージェント集合型関数

KK-MAS には、エージェント集合型変数をあつかうためのルール（関数）が多数、用意されています。（ヘルプ参照）それぞれ便利なものですが、今回は、その中から２つを学びましょう。

☆MakeOneAgtSetAroundOwn　自分の周りを認識する

自分自身の座標の周辺にいる（視野範囲内の）エージェントのリストをエージェント集合型変数の中に格納する関数です。以下のように書いて用います。４つの引数を設定してやります。

MakeOneAgtSetAroundOwn(エージェント集合型変数, 範囲, エージェント種別, 自分自身を含むか含まないか)

Ex. MakeOneAgtSetAroundOwn(My.Neighbors, 1, Universe.二次元空間.Walker, False)

☆**CountAgtSet**　数える

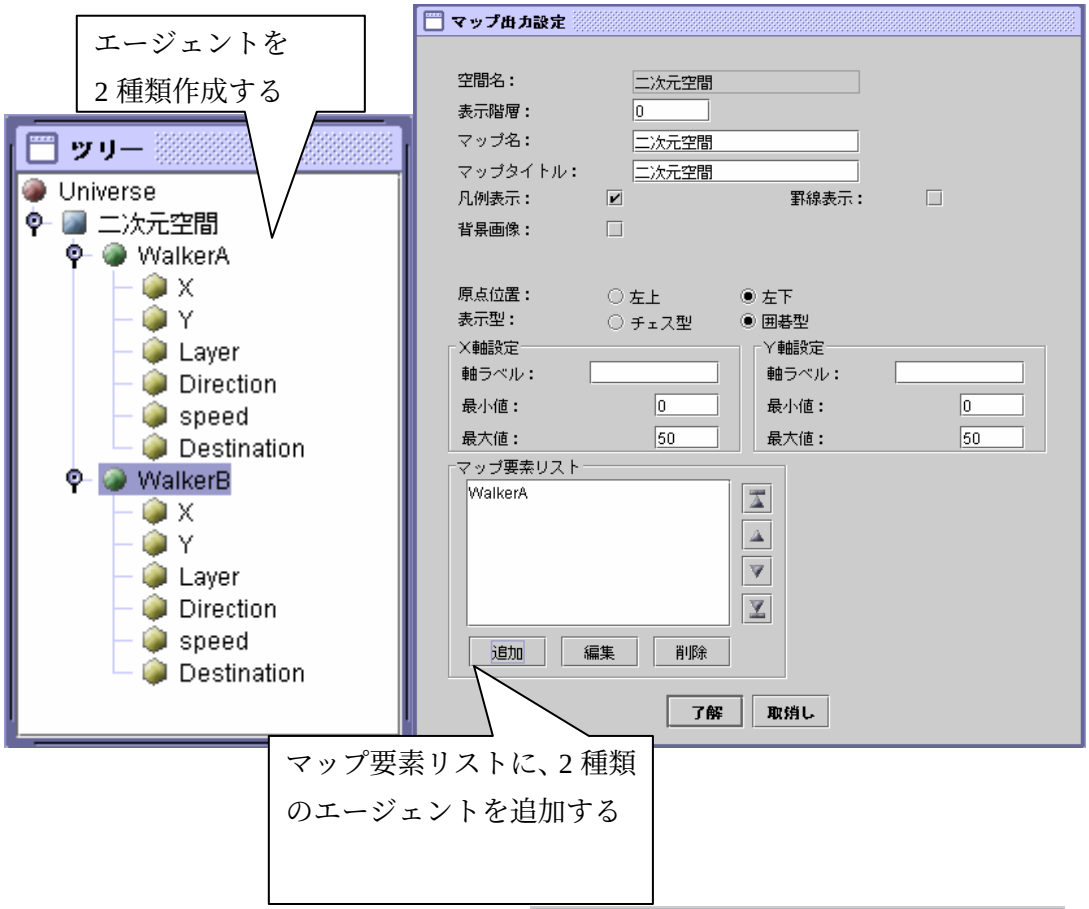
エージェント集合型変数の中に格納されているエージェントの数を数える関数です。以下のように書いて用います。１つの条件を設定してやります。

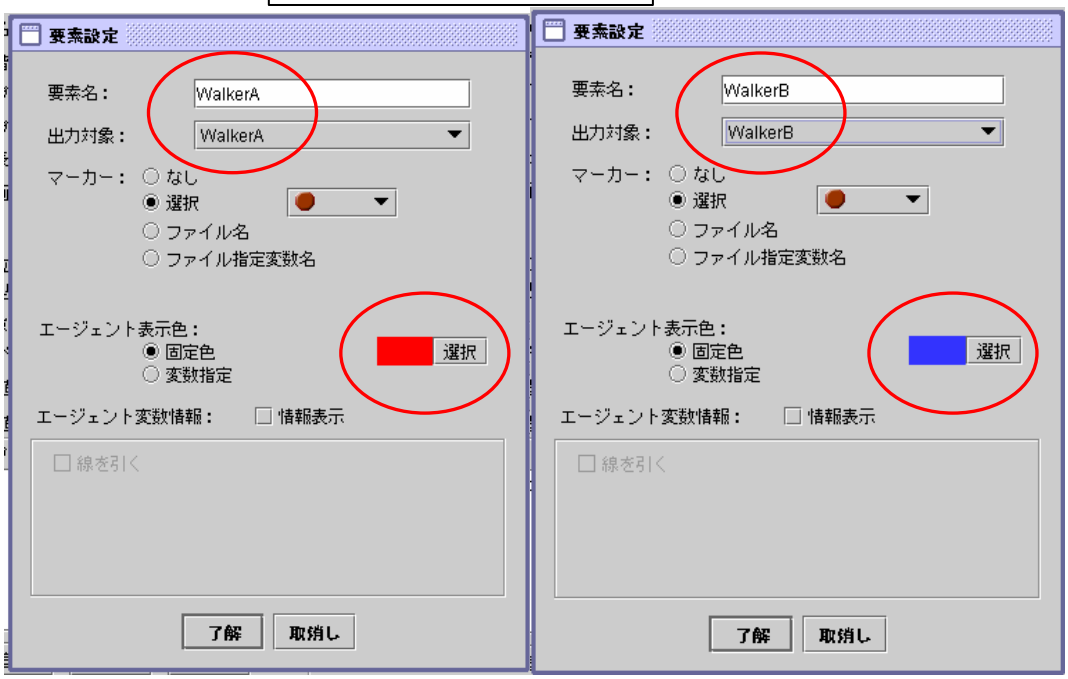
CountAgtSet(エージェント集合型変数)

Ex. My.Num_Neighbors = CountAgtSet(My.Neighbors)

それでは例題です。

[１] 50×50 の空間上に、家路にいそぐ歩行者エージェントを 2 種類用意してください。それぞれ数は 50 とします。歩行者 A は東の方角（Destination 0）に家がある人々です。歩行者 B は西の方角（Destination 180）に家がある人々です。それぞれの歩行者は最初はランダムに歩き始めますが、家の方角へと（3 度ずつ）方向を変えていきます。





要素名、出力対象をそれぞれのエージェントで設定します（色も選んでください）。

歩行者 A のエージェントルール

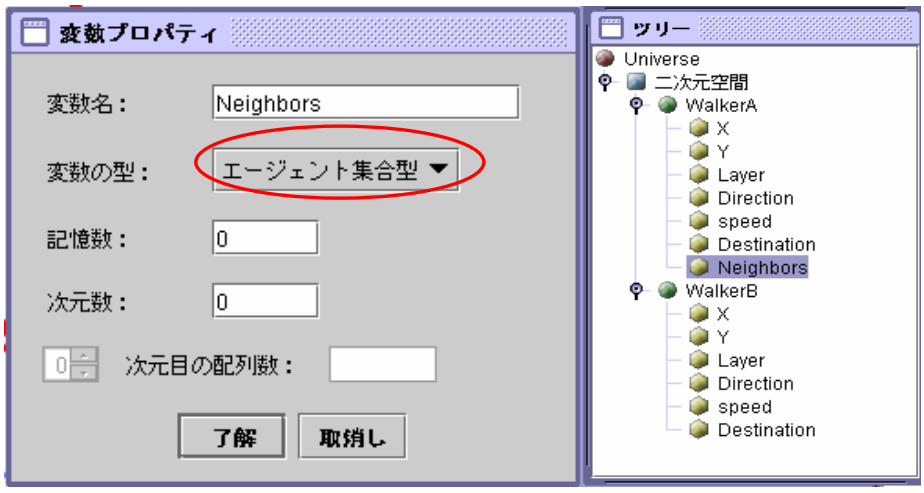
```
Agt_Init{

// 初期値を設定します
My.X = 0
My.Y = rnd()*50
My.Direction = rnd()*360
My.Speed = rnd()*0.5 + 0.5
My.Destination = 0
}
Agt_Step{
// 前に進みます
Forward(My.Speed)

If my.Direction > 0 and    My.Direction < 180    then
        Turn(-3)
else
        Turn(3)
end if}
```

歩行者 B のエージェントルール

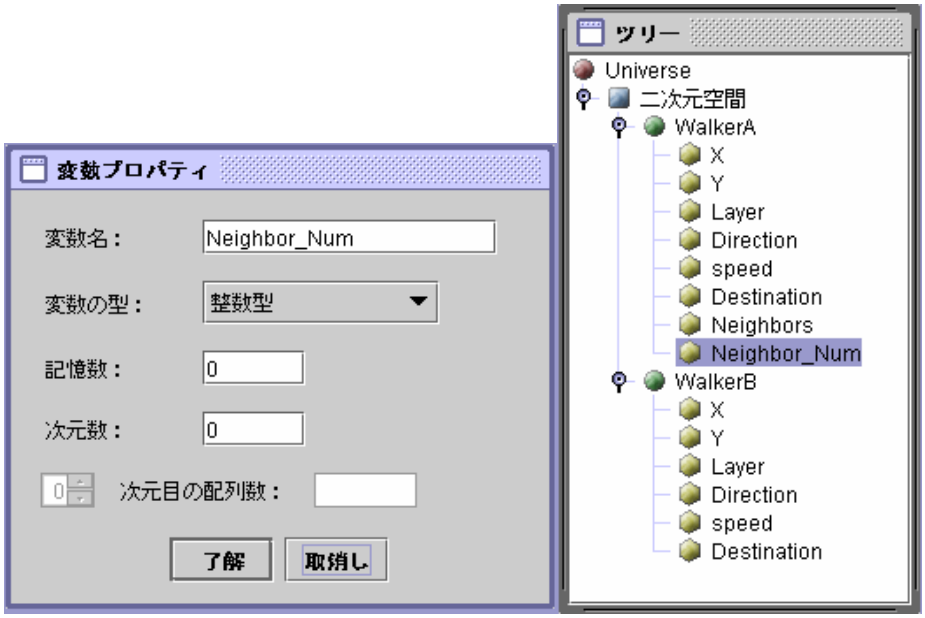
```
Agt_Init{

// 初期値を設定します
My.X = 49
My.Y = rnd()*50
My.Direction = rnd()*360
My.Speed = rnd()*0.5 + 0.5
My.Destination = 180
}
```

以下、Turn の引数内の符号だけ変更すれば同じルール

★ではエージェント集合関数を使用してみましょう。まず、歩行者 A に「近接者（Neighbors）」という名称のエージェント集合型の変数を設定してあげてください。



★次に、周りの数を格納する変数を設定します。



★ではルールエディタに戻ります。設定したエージェント集合型変数（近接者（Neighbors））に、向かってくる歩行者（A にとってはB）のうち、距離１以内に近づいたものを格納するルールをエージェントステップの最後に書きます。

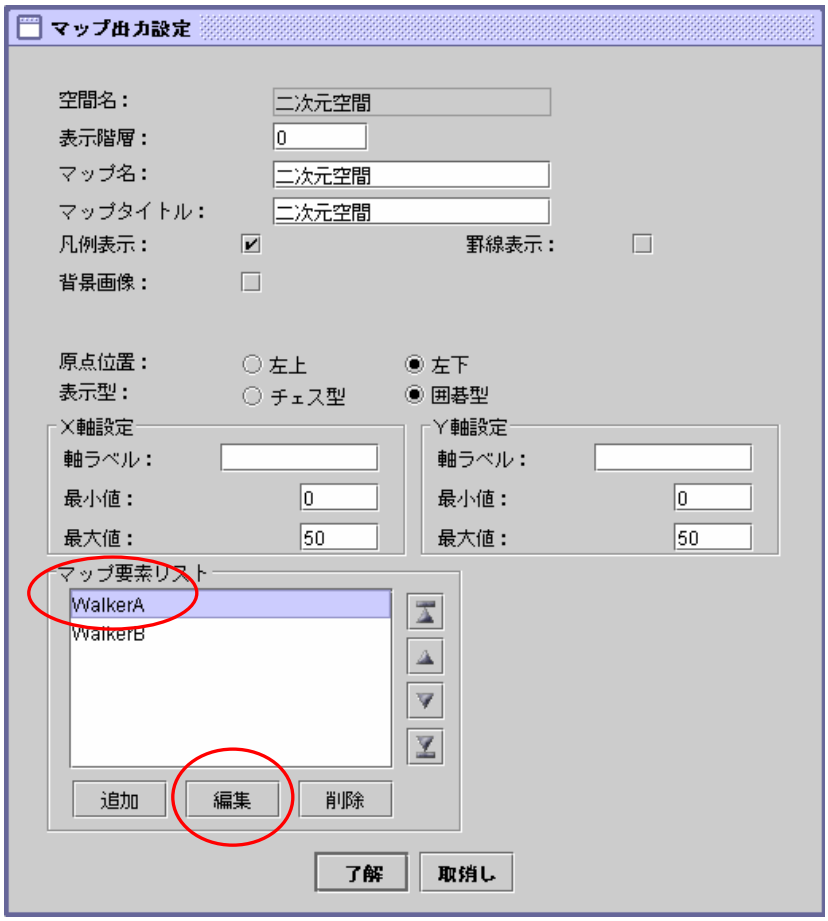
```
// 周囲を認識します
MakeOneAgtSetAroundOwn( my.Neighbors, 1, Universe.二次元空間.WalkerB, False)
```

★その下に、歩行者たちが自分の近接者の数を数えるルールを書いてみましょう。

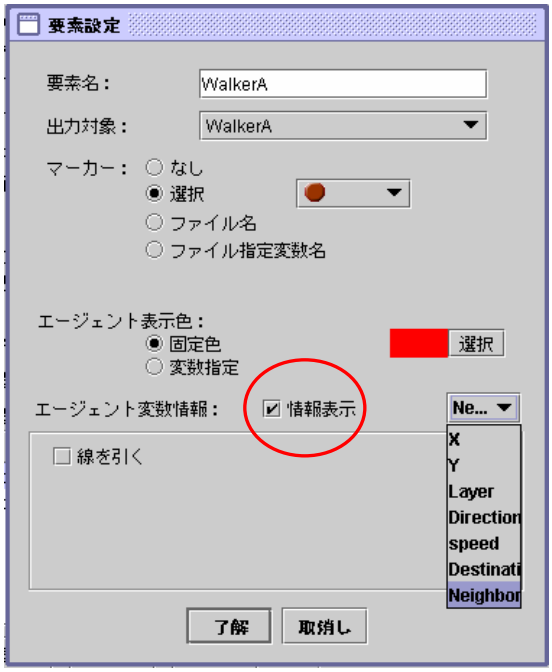
```
// 近接者を数えます
my.Neighbor_Num = CountAgtSet(my.Neighbors)
```

★情報出力で確認してみましょう。
＜設定＞－＜出力設定＞を選択して、出力項目リストを表示させます。

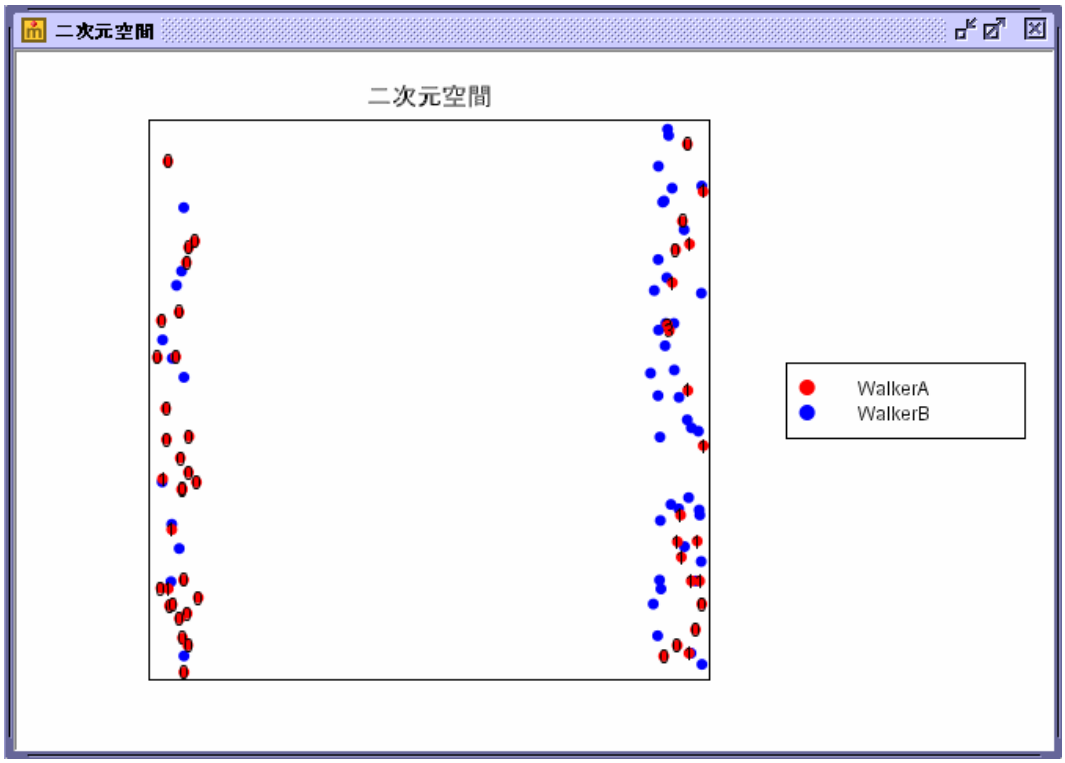
二次元空間が選択されていることを確認してから、編集をクリックします。



歩行者 A を選択して、編集をクリックします。



エージェント変数情報の情報表示にチェックを入れて、先ほど追加した周りの数を格納する変数を指定します。これで二次元マップ上に、変数が表示される準備ができました。
再生してみましょう。



いかがでしょうか？ちゃんと歩行者 A に、歩行者 B の数が認識されて表示されていますか？
歩行者 B の数を増やすとわかり易いでしょう。

［2］［1］のモデルに、歩行者 B にも同じ振る舞いをさせてあげてください。またその際、
距離 1 以内ではなく距離 2 以内に設定してください。

＞下記のように記述します。

歩行者 B のエージェントルール

```
Agt_Init{

// 初期値を設定します
My.X = 49
My.Y = rnd()*50
My.Direction = rnd()*360
My.Speed = rnd()*0.5 + 0.5
My.Destination = 180
}
Agt_Step{
// 前に進みます
Forward(My.Speed)

// 方向を修正します
If My.Destination> My.Direction and My.Destination< My.Direction+180   then
        Turn(3)
elseif  My.Destination< My.Direction-180    then
        Turn(3)
else
        Turn(-3)
end if

// 周囲を認識します
MakeOneAgtSetAroundOwn( my.Neighbors, 2, Universe.二次元空間.WalkerA, False)

// 近接者を数えます
my.Neighbor_Num = CountAgtSet(my.Neighbors)
}
```

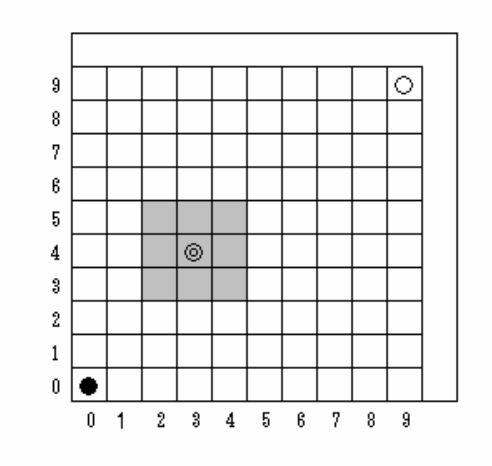
確認のため、変数情報も表示させましょう。

●セル型のモデル

これまでは、空間の中をエージェントが自由に動き回るモデルを扱いましたが、今回からは、空間にマス状の格子があってそのマスに入る形でエージェントが存在する（移動する）モデルについて学びます。将棋やオセロを思い浮かべてください。こういうタイプの空間表現をセル型と呼びます。

（１）座標は整数のみ。

（２）さらに、座標は０から数える。



☆前回までの Forward（自分の Direction の方向に進む）に代わり、セル型空間の中では、ForwardXCell と ForwardYCell、ForwarDirectionCell というルールを用います。

ForwardXCell(整数) = X 軸方向に○マス進む

ForwardYCell(整数) = Y 軸方向に○マス進む

ForwarDirectionCell(整数 1,整数 2)=整数１方向へ整数２マス進む

方向は、０：右、１：右上、２：上、３：左上、４：左、５：左下、６：下、７：右下となります。

☆**MoveToSpaceOwnCell**

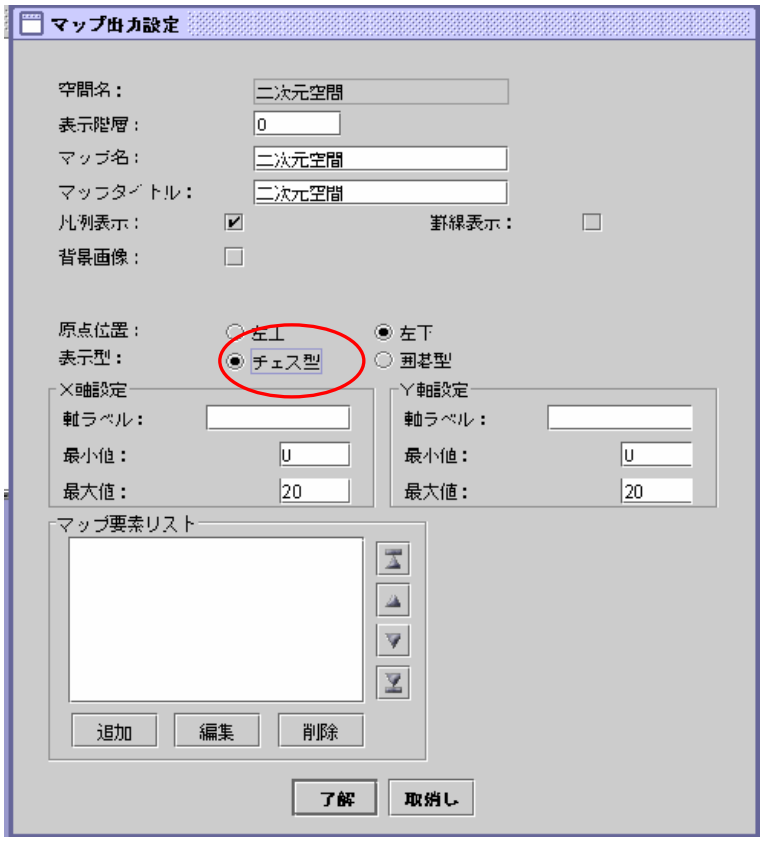
また、エージェントを移動させるのに、Turn, Forward という関数を用いないでも、セル型では、もう少しいい加減な（高度な？）移動方法もあります。「周囲に空いている空間があれば、そこに移動する」というルールです。

MoveToSpaceOwnCell(視野)

それでは簡単な例題で確かめてみましょう。

［３］10×10 のセル型の空間上に、歩行者エージェントを 2 種類ランダムに配置してくだ

さい。それぞれ数は 10 とします。歩行者 A はぶつからないように動き、歩行者 B はランダムに方向を決定して動きます。



マップ出力の際、表示型をチェス型にします。

＞ルールは以下のように記述します。

歩行者 A のエージェントルール

```
Agt_Init{
my.X = int(rnd()*10)
my.Y = int(rnd()*10)
}
Agt_Step{
MovetoSpaceowncell(1)
}
```

歩行者 B のエージェントルール

```
Agt_Init{
my.X = int(rnd()*10)
my.Y = int(rnd()*10)
}
Agt_Step{
dim dir as integer
dir = int(rnd() * 8)
Forwarddirectioncell(dir, 1)
}
```

ちなみに、歩行者 B のエージェントルールで出てきた int()は、**int(　　)**
　　の値を切り捨てします。　　の値が例えば、**3.1415** のときは、**3** となります。

●それでは宿題です。

［4］［2］で作ったモデルを使用します。歩行者 A は歩行者 B の近接者が一人でもいれば右に 20 度曲がるように、歩行者 B は逆に歩行者 A の近接者が二人以上いれば左に 10 度、曲がるようにルールを書いてください。

［5］５0×５0 のセル型空間上に、2 種類のエージェントをランダムに配置します（人数は任意）。歩行者 A は歩行者 B の近接者が一人でもいれば止まり、それ以外は空きスペースに一歩ずつ移動します。歩行者 B は逆に歩行者 A の近接者が二人以上いれば止まり、いなければ同様に空きスペースに移動するルールを書いてください。

＜ヒント＞止まるということは、今自分が位置している場所に進む、ということですね。

［6］１0×１0 のセル型空間上に、2 種類のエージェントをランダムに配置します（人数は任意）。どちらの種類も、自分と同じタイプのエージェントの近接者が二人以上いれば空きスペースに移動し、いなければ止まるルールを書いてください。分居モデルらしくなりましたか？

本日はかなり多くの事項を学習したので、しっかりと復習をしておいてください。

次回は分居モデルに挑戦です！！