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 6 回　エージェント、変化する！（11 月 11 日）

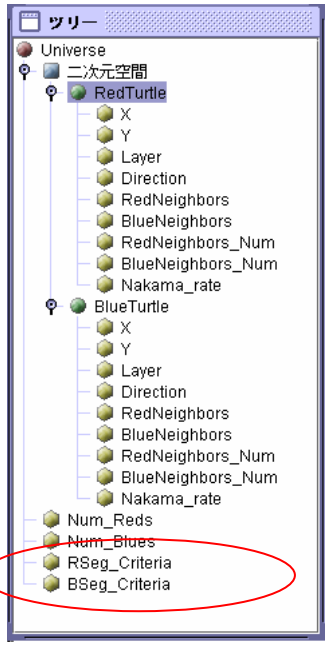
●概略

○前回の宿題の解答例と解説
○変化するエージェント：状態と表示色の変化
○周囲のエージェントの情報参照１：エージェント型変数と For Each 文
○時系列グラフによる出力方法
○周囲のエージェントの情報参照２（時間があれば）： GetAgt 関数
○宿題

●前回の宿題の解答例と解説

［4］赤亀と青亀の持つ満足水準をコントロールパネルで別々に設定できるようにしてみましょう。

＞これは難しくありませんね。前回学んだコントロールパネル設定の方法にのっとり、まず下図のように赤亀、青亀別々の閾値を設定するための変数 RSeg_Criteria、BSeg_Criteria を Universe 直下に追加します（実数型にすることを忘れずに！）。



次に各々の変数をコントロールパネル設定で操作できるようにします。
ルールの部分は、これまで一律だった Seg_Criteria を RedTurtle、BlueTurtle 別々に書き換えます。RedTurtle の場合、Agt_Step ルールの末尾を下記のように変更します。

```
//不満なら移動します
If My.Nakama_Rate < Universe.RSeg_Criteria then
        MoveToSpaceOwnCell(2)
end if
```

［5］分居モデルは、一定の比率の同種別の亀がいないと引っ越しするというルールでした。では、ある一定の異種別の亀がいると、嫌がって引っ越しするというルールを作ってみましょう。

＞このルールも、分居モデルのルールにちょっとした修正を加えることで作ることができます。分居モデルの亀の行動は、同種の仲間の比率 Nakama_rate に基づいていましたが、今度は異種の仲間の比率（これを Ishu_rate としておきましょう）に基づいて行動するので、ツリーの変数の名称を変更した上で、RedTurtle、BlueTurtle の Agt_Step ルールの後半に下記（RedTurtle の例です）のような変更を加えます。赤亀、青亀のカウント部分はそのままです。

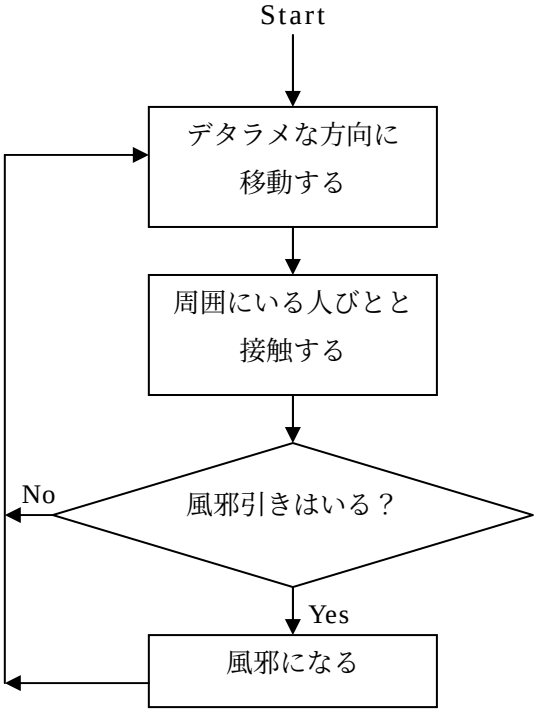
```
//異種率を計算します
If AllNum == 0 then
        My.Isyu_Rate = 0
else
        My.Isyu_Rate = My.BlueNeighbors_Num / AllNum
end if

//不満なら移動します
If My.Isyu_Rate > Universe.Seg_Criteria then
        MoveToSpaceOwnCell(2)
end if
```

●本日のモデルとそのポイント

前回までの五回の講義で、基本的な文法事項と KK-MAS を用いてマルチエージェント・シミュレーションのモデルを作るための「イロハ」を一通り学び終えました。今回の講義からは、一回に一個のペースでまとまったモデルを作りながら、マルチエージェント・シミュレーションの様々な技法を学んでいきます。気合いを入れて頑張っていきましょう。

突然ですが、急激に冷え込んできた今日この頃、周囲で風邪引きさんをちらほら見かけるようになりました。そこで、今回は、動き回るヒトの間で風邪が蔓延していく「風邪引き」モデルを作ってみましょう。風邪を引いているヒトと接触すると自分も風邪を引くというごく単純なモデルです。情報の伝播やファッションの流行など、モデルの解釈次第で他の事例にも適用できる、汎用性のあるモデルにもなっています。

具体的には、エージェントは概略下図（これをフローチャートといいます）のようなルールに従って動きます。



エージェントにこのような動きをさせるためには、新たな文法事項や技法を幾つか学ばなければなりません。今日のポイントは以下の通りです。

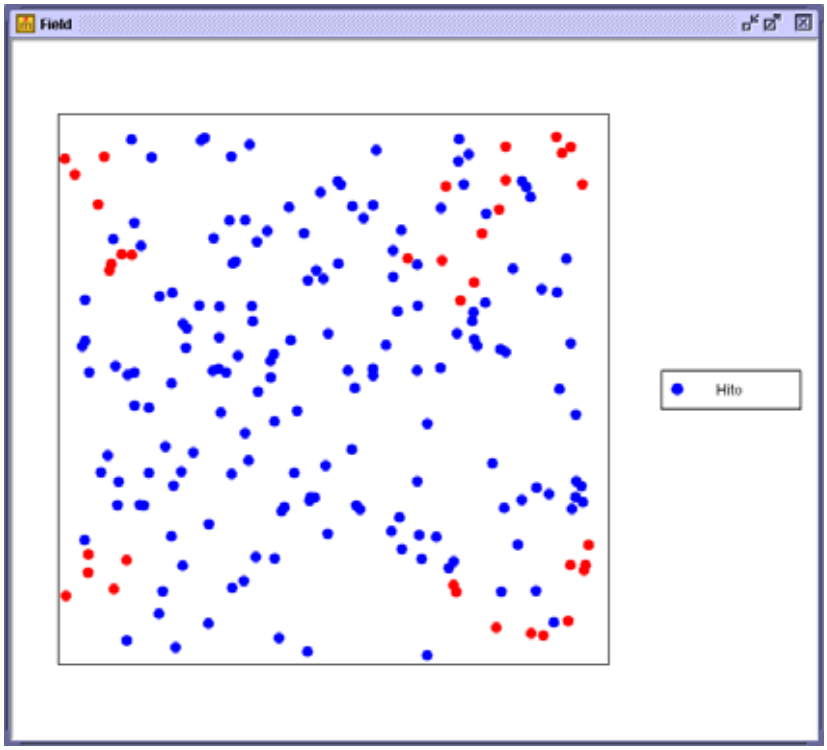
①エージェントの状態と表示色の設定

これまで作ってきたモデルで登場したエージェントは、動き回りこそしましたが、自身の属性や状態は全く変化しませんでした。「walker」はいつまでたっても同じ「walker」、「赤亀」はいつまでたっても同じ「赤亀」だったわけです。

今回登場するエージェントは、「風邪を引いている」「引いていない（つまり健康）」という二つの状態を持っています。このようにエージェントが複数の状態を持っている状況は、ある情報を「持っている」「持っていない」、「赤色の服を着る」「青色の服を着る」、パックマンが「無敵状態」かそうでないか…などなど、いくらでも考えることができます。

KK-MAS において、エージェントに複数の状態を持たせるために必要なことは、状態を表す整数型や実数型の変数（たとえば State）をツリー上でエージェントに追加することだけです。その上で、State=0 の場合に「健康」、State=1 の場合に「風邪引き」などと対応付ければよいわけです。

エージェントが自らの状態を変化させるようになると、たとえば下図のように「健康」の場合は青、「風邪引き」の場合は赤といった具合に、状態に応じた色でエージェントを出力させたい場合も出てくるはずです。そこで今回は、マップ上のエージェントの表示色をこれまでの「固定色」ではなく、**色変数を使った「変数指定」**で出力させる方法を学びましょう。色変数といっても、色を表す整数型の変数をエージェントに追加し、そこに値を代入するだけです。KK-MASにはColor_Blue、Color_Redなど特定の色を表す便利な予約語が用意されているので、これを色変数に代入して表示させることにします。



②**For Each文**（周囲のエージェントの情報の参照）

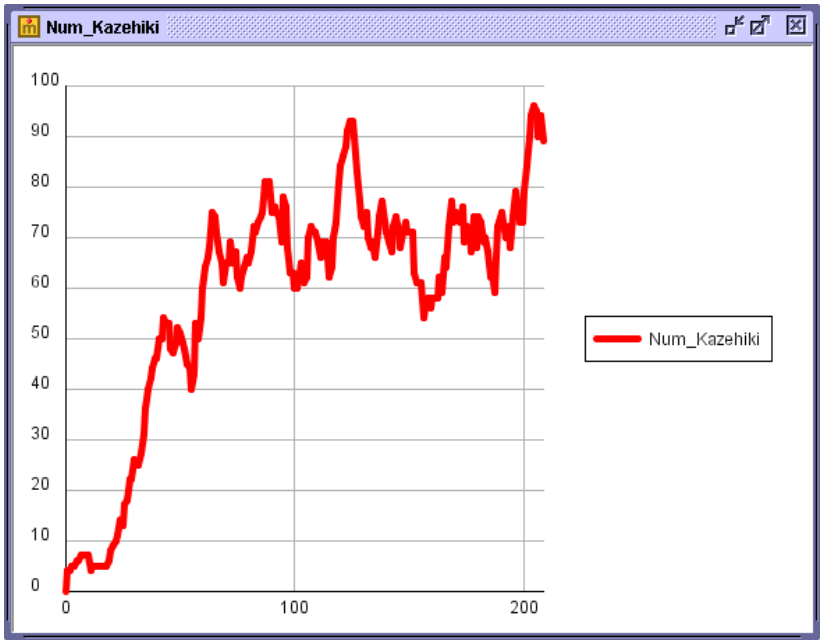
上図のフローチャートにもあるように、今回のモデルのエージェントは、単に自分の周りにいるエージェントの数をひとまとめにカウントするのではなく、周囲の個々のエージ

ェントの状態（「風邪引き」か「健康」か）を順次参照していくというさらに高度な処理を行っています。このように一群のエージェントの情報を個別に参照したり、さらにそれに基づいてエージェントを選別したりする場合に必要になるのが、**For Each文**という構文です。すでに学んだMakeOneAgtSetAroundOwn関数などと組み合わせて用いることで、あるエージェント集合（隣人の集合など）に属する個別のエージェントに関して、繰り返し特定の処理を行うことができます。

For Each文を用いて個々のエージェントを参照する際に、これまで学んだエージェント集合型変数に加え、個別のエージェントを代入するための変数**「エージェント型変数」**（前々回の資料 4 ページの表を参照）を使う必要があります。これについても学びましょう。

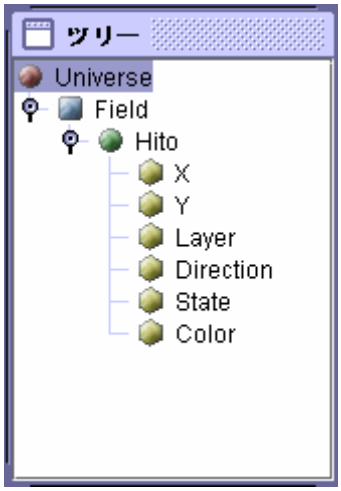
③時系列グラフ出力

マップ上をエージェントが動き回る姿を眺めるだけでは、そろそろ飽き足らなくなってきたのではないでしょうか。実際、今回のモデルでは、マップ出力だけでは、風邪の蔓延の様子を十分に捉えることができません。KK-MASには「マップ出力」以外にも、「時系列グラフ」「棒グラフ」「数値出力」などシステムの状態を出力する様々な機能が備わっています。今回は**「時系列グラフ」出力**を取り上げて、下図のように、風邪にかかっているヒトの数の変化を出力させる方法を学びましょう。具体的にはUniverse直下に出力のための変数を作り、ステップごとに風邪を引いているヒトの数をカウントして変数に代入し、出力設定でその変数値を出力させるという手順を踏むことになります。

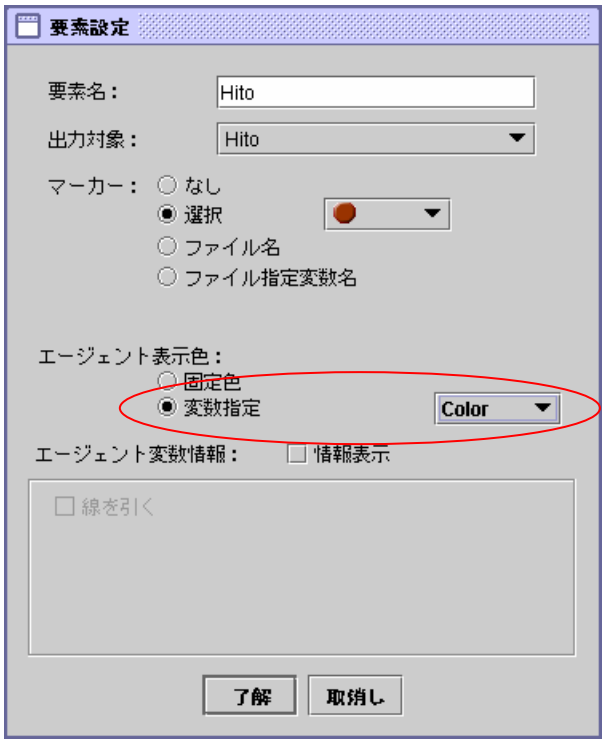


●モデル作り１：エージェントの生成と配置

まず、いつものようにツリー上に空間とエージェントを追加することから開始します。空間（以下の例では Field）はデフォルトの 50×50 の空間、エージェント（Hito）は一種類のみです。ポイント①で述べたように、今回のエージェントは「健康」「風邪引き」の状態を区別するための変数（State）とそれぞれの状態に応じた色を示すための色変数（Color）を持つことになります。それぞれ整数型のまま Hito エージェントに追加してください。



次に、やはりいつものように、出力設定でマップ出力のための設定をしますが、エージェントの表示色が色変数に連動して変化するように、要素設定で「エージェント表示色」を「変数指定」とした上で、プルダウンから色変数 Color を選びます。



その上で、Univ_Init および Agt_Init に初期状態を作るためのルールを記述します。

[1] Hito を 200 人生成し、空間上にランダムに配置します。200 の Hito のうち 3%の Hito が「風邪引き」の状態、残りが「健康」の状態になるようにランダムに各 Hito の状態を決めます。さらに、「風邪引き」の Hito は色が赤、「健康」の Hito は色が青になるようにします。

＞まず、Universe_Init には、下記のように記述します。

```
Univ_Init{

        Dim i As Integer
        Dim hitobito As AgtSet

        // For 文を使って Hito を 200 登場させる
        For i = 0 To 199
                CreateAgt( Universe.Field.Hito )
        Next i

        // Hito をエージェント集合に格納してランダムにばらまく
        MakeAgtSet( hitobito, Universe.Field.Hito )
        RandomPutAgtSet( hitobito )

}
```

先週学んだ RandomPut AgtSetCell 関数の実数空間バージョン

これは、前回学んだ分居モデルの赤亀・青亀の配置の仕方と基本的に同じです。「For 文と CreateAgt 関数を使って指定数のエージェントを生成」→「生成したエージェントをエージェント集合型変数に集める」→「集めたエージェントをランダムにばらまく」という流れです。前回と違うのは、ばらまく際に使っている関数がセル空間対応の RandomPutAgtSetCell 関数ではなく（前回資料 7 ページ参照）、実数空間対応の RandomPutAgtSet 関数である点だけです。この関数の仕様は下記の通りです。

```
RandomPutAgtSet ( エージェント集合型変数 )
```

＞一方、Agt_Init には、下記のように記述します。

```
Agt_Init{

        // 3%の確率で最初から風邪を引いているとする（状態は 1、色は赤）
        If Rnd( ) < 0.03 Then
                My.State = 1
                My.Color = Color_Red
        // それ以外の場合は健康（状態は 0、色は青）
        Else
                My.State = 0
                My.Color = Color_Blue
        End If

}
```

「赤」を表す予約語

「青」を表す予約語

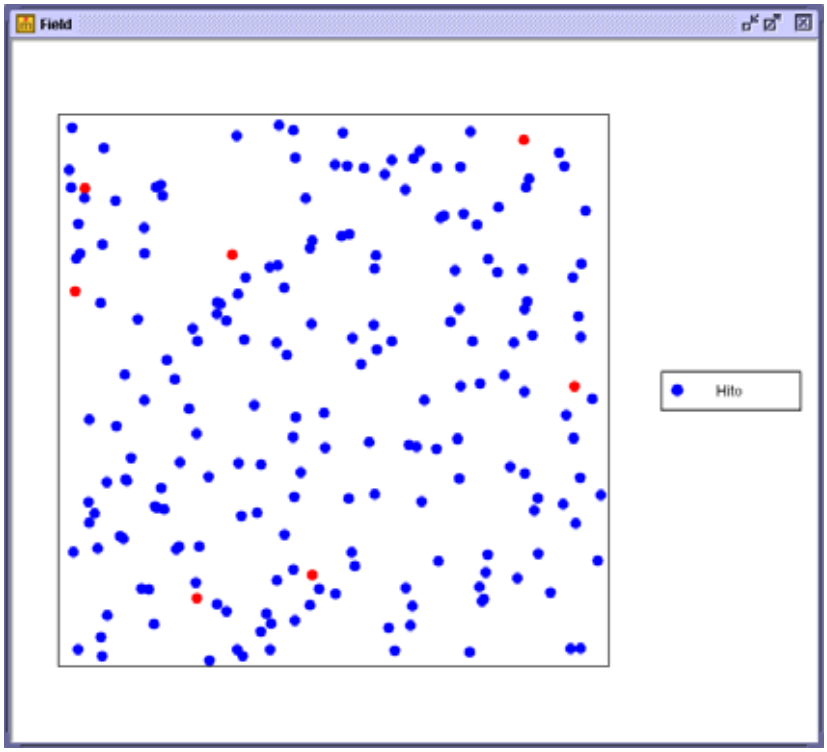
一様乱数 Rnd( )を用いて、一定確率（ここでは 0.03）である処理（ここでは「風邪引き」エージェントを生成する）をする技法については、すでに第三回の例題[3]で学んでいます。

上では、State 変数が 1 の場合は「風邪引き」を、State 変数が 0 の場合を「健康」として、各々に対応する Color 変数に赤と青を割り当てています。「Color_Red」「Color_Blue」はそれぞれ「赤」と「青」を表す「予約語」です。色は本来整数値で指定されるのですが、これらの予約語を記述することで、ルールを実行する際に KK-MAS の側で自動的に整数値に変換して出力に反映してくれます。このように予約語で指定できる色には、

Color_Red 赤
Color_Green 緑
Color_Blue 青
Color_Yellow 黄
Color_Cyan 水色
Color_Mazenta 紫
Color_Black 黒
Color_White 白

があります（もっと細かな色を指定したければ、RGB()という関数を使います。これについては次々回取り上げる予定です。興味のある方はヘルプを参照してください）。

さて、このあたりでいったんモデルのファイルを保存して実行ボタンを押してみてください。指定どおりの割合で赤と青のエージェントが表示されているでしょうか。



●モデル作り２：「風邪引き」ルールの記述

次にステップごとの Hito エージェントの行動ルールを Agt_Step に書き込んでいきましょう。冒頭のフローチャートで示したルールを KK-MAS の言葉に置き換えていくわけです。

[２] 各ステップ、Hito は視野１の範囲内にいる全ての Hito エージェント（自分は除く）と接触します。このとき、自分の状態が「健康」で、かつ相手の状態が「風邪引き」なら風邪をもらい、自分も「風邪引き」になってしまいます。

＞以下のように記述します。

```
Agt_Step{

        Dim aite As Agt
        Dim rinjin As AgtSet

        // デタラメな方向に移動する
        My.Direction = 360 * Rnd( )
        Forward( 2 )
```

個別のエージェントを代入する「エージェント型」変数の定義

```
        // 隣接している他の Hito を集める
        MakeOneAgtSetAroundOwn( rinjin, 1, Universe.Field.Hito, False )

        // 隣接している aite ひとりひとりと接触
        For Each aite In rinjin

            // もし自分が健康で aite が風邪を引いていたら風邪をもらう
            If My.State == 0 AND aite.State == 1 Then
                My.State = 1
                My.Color = Color_Red
            End If

        Next aite

}
```

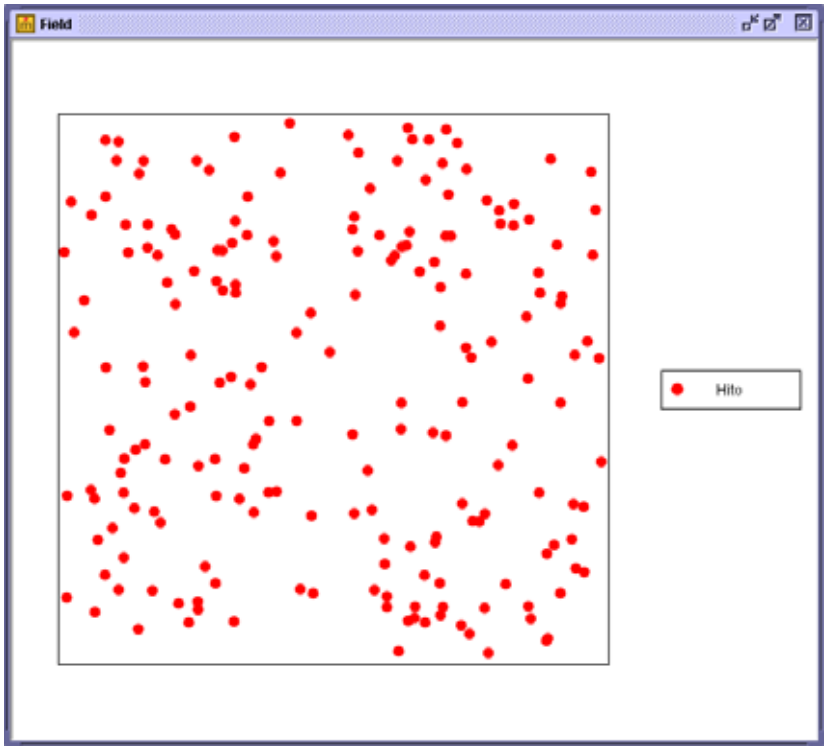
隣人が風邪を引いているかどうか一人ひとりチェック

ばらばらな方向に移動していくルールについては、皆さんにとってお手のものでしょう。MakeOneAgtSetAroundOwn 関数を使って周囲のエージェントを集める方法についても前回までに学びました。ここでの中心は、ポイント②に出てきた For Each 文です。上の例から分かるように、この構文は、

```
For Each エージェント型変数 In エージェント集合型変数
        ＜ルール＞
Next エージェント型変数
```

という書式で、エージェント集合型の変数（ここでは rinjin）に属するエージェント型の変数（ここでは aite）各々について、＜ルール＞で指定された処理を繰り返し実行するというものです。ここで新たな変数型として出てきた「エージェント型」の変数は、個別のエージェントを代入するための変数で、「My.State」等で自分の変数を参照できたように、「aite.State」とすることで当該エージェントの変数を参照することが可能です（残念ながらこの場合はピリオド入力後の入力支援機能が出てきません）。エージェント型変数は、ツリーで作る際には「変数の型」を「エージェント型」にすることで、上の例のように一時的な変数として用いる際にはルールエディタで「Dim aite As Agt」というふうに宣言することで、それぞれ定義することができます。

ここまでできたらファイルを保存して、さっそくモデルを動かしてみましょう。しばらく動かすと、図のように一面まっかっか、全員風邪引きさんになってしまいます。



●モデル作り３：「快復」ルールの導入

シミュレーションの結果が上図のようになったのは、Hito エージェントのルールが、一度風邪を引いたら二度と快復しない、つまり一生風邪を引いたままの状態でいるルールになっているからです。そこで、Agt_Step に以下のようなルールを書き加えてみましょう。

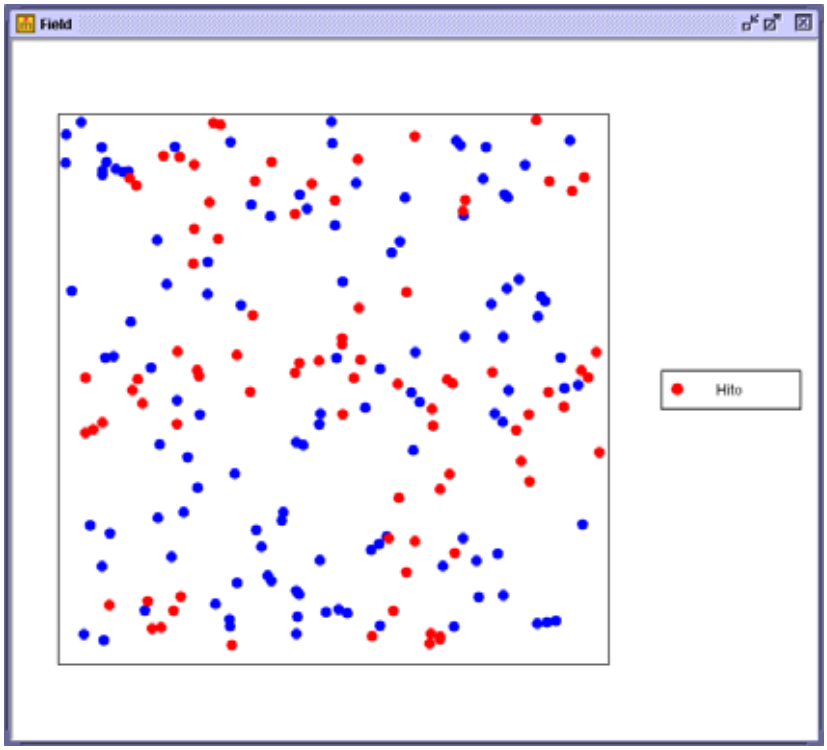
[３]「風邪引き」の状態になった Hito は、10 ステップの「感染期間」を経たのち再び「健康」の状態に戻ります。

＞文法的に新しいことはありません。Hito に自分の「感染ステップ数」を記録する変数（以下の例では Counter）を加えた上で、Agt_Step の末尾に下記のように記述します。

```
// 風邪を引いている場合は感染期間をチェック
If My.State == 1 Then

    // 10 ステップ過ぎたら快復しカウンターを初期化
    If My.Counter == 10 Then
        My.State = 0
        My.Color = Color_Blue
        My.Counter = 0
    End If
```

```
                    // 感染ステップをカウント
                    My.Counter = My.Counter + 1


            End If
```

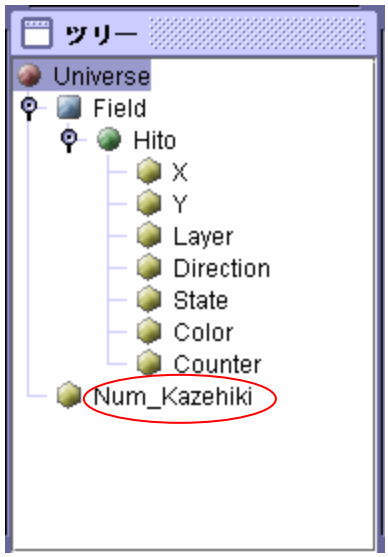
モデルを動かしてみてください。少しはそれらしい動きが出てきましたね。



●モデル作り４：時系列グラフによる出力

ここまででモデルの中身はほぼ出来上がりましたが、実際にモデルを動かしてみれば分かるように、エージェントが動き回っていることもあって、マップ出力だけでは、風邪の蔓延の仕方の動態がどうなっているのかいまいち判然としません。そこで、ポイント③で述べた手順に従って、時系列グラフを用いて、ステップごとの風邪引きさんの数を刻一刻と表示させてみましょう。

時系列グラフに限らず、ここでは扱わない「棒グラフ」や「値画面出力」など、KK-MAS の出力全般において、一般に、出力させる数値は Universe 直下の変数として定義します。そこでまず、ステップごとの風邪引きエージェントの数を納める整数型の変数（以下 Num_Kazehiki）を Universe に追加しましょう。



[４]各ステップ、「風邪引き」の状態にある Hito の数を数えて変数 Num_Kazehiki に代入します。

＞このような処理を行うには、幾つか方法があります。たとえば、各エージェントの Agt_Step において、Hito が「風邪引き」状態にあるなら Num_Kazehiki を１増やすというような処理がそのひとつです。ここでは、For Each 文の復習も兼ねて、ステップ終了時点で風邪引きエージェントの数をまとめて数えるルールを紹介します。前回学んだように（前回資料 8 ページ参照）、全エージェントが Agt_Step のルールを一通り実行し終えた後に実行されるのが、Univ_Step_End でした。ルールはここに書き込みます。

```
Univ_Step_End{

        Dim kojin As Agt
        Dim hitobito As AgtSet

        // Hito をエージェント集合に格納
        MakeAgtSet( hitobito, Universe.Field.Hito )

        // ステップごとの数を数えるのでカウンターを初期化
        Universe.Num_Kazehiki = 0
```

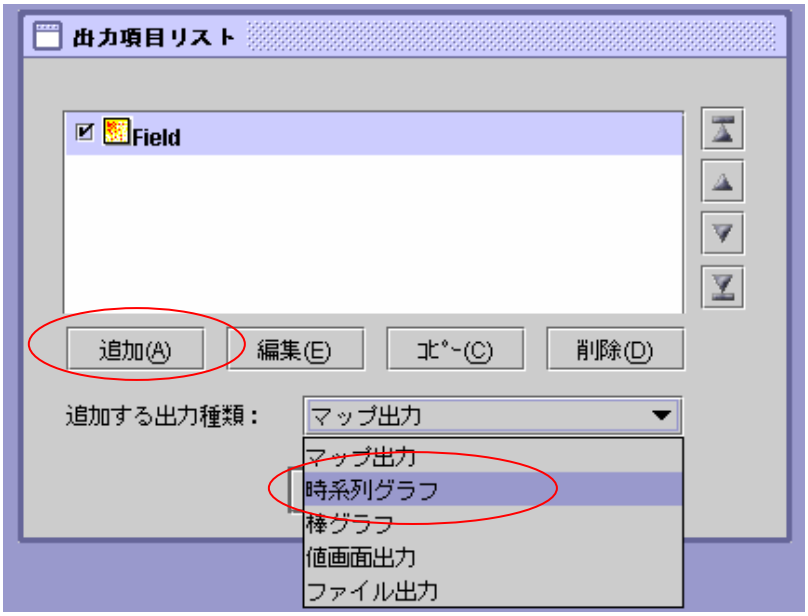
これがないと風邪引きさんの累積数が記録されてしまう

```
        // 全エージェントをチェックし風邪引きさんの数を数える
        For Each kojin In hitobito
                If kojin.State == 1 Then
                        Universe.Num_Kazehiki = Universe.Num_Kazehiki + 1
                End If
        Next kojin

}
```

今回は、MakeAgtSet 関数を用いて全ての Hito エージェントをエージェント集合型変数（hitobito）に集め、この集合に属する個別のエージェント（kojin）について、各々の状態を順次チェックし、「風邪引き」なら Num_Kazehiki が 1 ずつ加算されていきます。

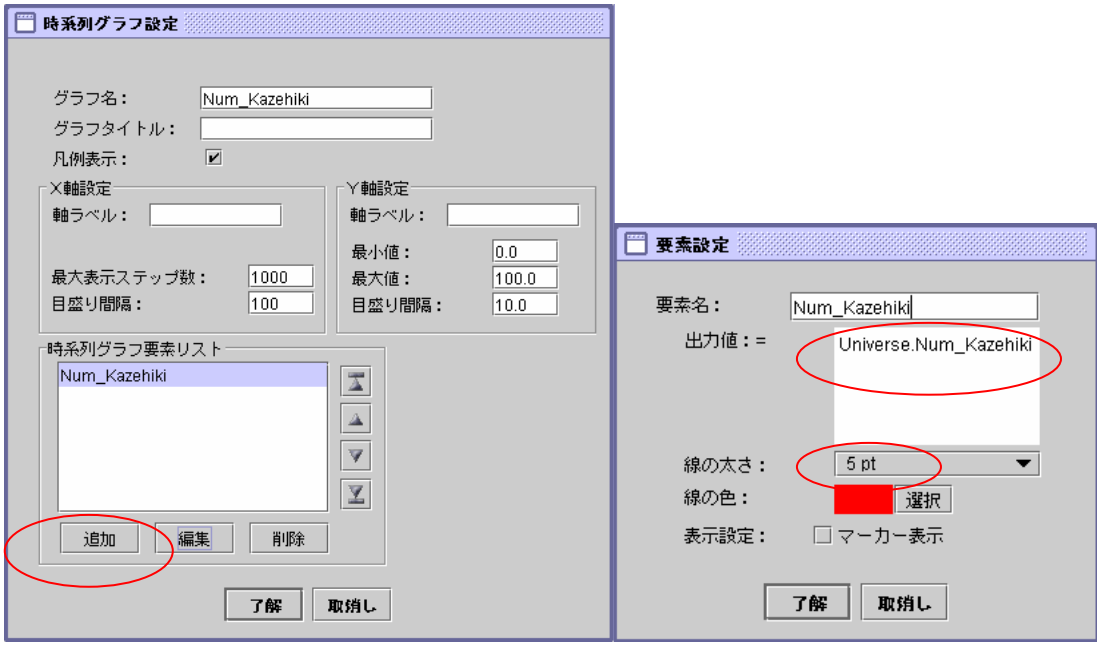
上記ルール中にあるように、ステップごとに Num_Kazehiki を 0 に戻さないと、当該ステップまでの風邪引きさんの累積数がこの変数に代入されてしまうので、注意が必要です。

ここまで準備をした上で出力設定に移りましょう。マップ出力の際と同様に「設定」メニューから「出力設定」を選んで「出力項目リスト」ダイアログを開いた上で、下図のように「追加する出力種類」プルダウンから「時系列グラフ」を選んで「追加」をクリックします。

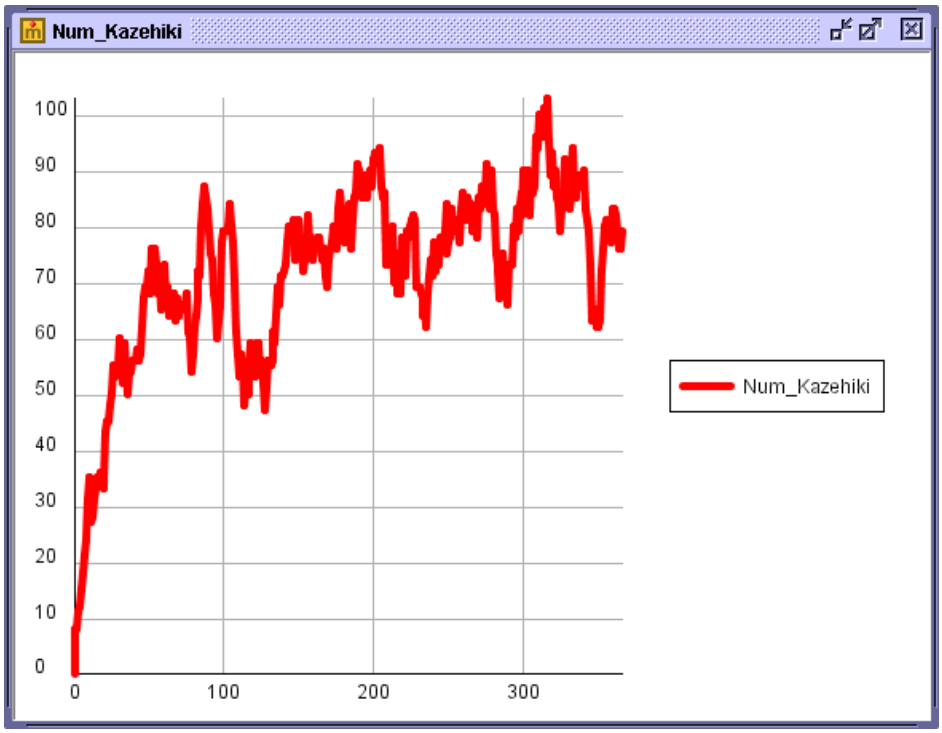


すると、「時系列グラフ設定」ダイアログが出てくるので、マップ出力の際と同じ感覚で「グラフ名」等を入力し、「時系列グラフ要素リスト」下の「追加」を選んで「要素設定」画面を出します（次ページ図）。

「出力値」には、出力値を納めた Universe の変数、つまり「Universe.Num_Kazehiki」を入力します。グラフの見栄えをよくするため、「線の太さ」を「5pt」に変えておきます。



モデルのファイルを保存して実行ボタンを押してみましょう。下図のようなリアルタイムで変化する時系列グラフが出てくれば、本日の課題は見事クリアです。



●おまけ：GetAgt 関数によるエージェントの取り出し

For Each 文に関連して、意欲旺盛な方のために、もうひとつ文法事項の話題を提供しておきます。この文法事項は次々回改めて取り上げる予定ですので、手が回らない方は読み飛ばしてもらって結構です。

For Each 文は、一群のエージェント集合に属する全ての個別エージェントについて、情報を参照したり作用したりするために用いる構文でした。それに対して、あるエージェント集合に属するエージェントのうち、ひとつだけ選択したい場合も存在するでしょう。たとえば、周囲の人の中から話し相手を選ぶ、お見合い候補の中から一人相手を選ぶといったケースです。このような処理を行いたいときに便利なのが GetAgt 関数です。書式は、

```
GetAgt ( エージェント集合型変数 , エージェント番号（整数型） )
```

で、第一引数のエージェント集合の中から、第二引数の番号によって指定された個別エージェント（エージェント型変数）が選ばれ、返ってきます。後者の「エージェント番号」は、各エージェントに割り振られた一種の ID（0～n-1、n は集合の中のエージェントの数）ですが、ここでは詳しい説明を省略します。実用上、GetAgt 関数は下記のような形で、エージェント集合の中から無作為にひとつエージェントを取り出す際に頻繁に用いられます。

```
Dim hitobito As AgtSet
Dim ninzuu As Integer
Dim aite As Agt

MakeAgtSet( hitobito, Universe.Syakai.Hito )
ninzuu = CountAgtSet( hitobito )
If ninzuu >= 1 Then
        aite = GetAgt( hitobito, Int( ninzuu*Rnd( ) ) )
End If
```

ランダムにエージェント番号を指定

GetAgt 関数を用いる際に注意しなければならないのは、空っぽのエージェント集合からエージェントを取り出そうとすると、エラーになってしまう点です。上記の例のように If 文で中身が入っているかどうか確かめてから、取り出すようにしてください。

力試しのために、この構文を利用して「風邪引き」モデルに変更を加える宿題を用意しておきました。

●宿題

[5] 前回学んだコントロールパネルを用いて、「風邪引き」モデルの Hito の数（今回作ったモデルでは 200 に固定されていました)、シミュレーション開始時の「風邪引き」エージェントの割合（同 0.03)、および風邪の回復までにかかる「感染期間」(同 10 ステップ）を操作できるようにしてみましょう。その上で、各々関心のあるパラメータを動かしてみて、風邪の感染の仕方がどのように変化するのかを確かめてみてください。

[6] 「風邪引き」モデルを修正して、Hito が一度風邪から快復すれば、風邪への耐性を身に付ける、つまり風邪に一切感染しなくなる状態を新たに追加してみてください。この「無敵」状態の表示色は黄色にしましょう。感染の仕方にどのような変化が生じるでしょうか。

以下は余力のある人向けの課題です。

[7] 「風邪引き」モデルを修正して、Hito が周囲の Hito から無作為に一人「話し相手」を選び、この相手が風邪を引いていれば、自分も風邪を引くというふうにルールを変更してみましょう。GetAgt 関数を用います。

次週は休講になります。本講義での皆さんの最終的な目標は、各々が関心を持つ社会現象をマルチエージェント・シミュレーションのモデルで表現することですので、この機会を利用して、これまで学んだことをしっかりと復習し、モデルの構想をあたためておいてください。