世界モデル実習（SR）・人間の安全保障実験実習Ⅲ（GR）　　2006 年度夏学期

第十回 複雑なルールをシンプルにかく

●前回の復習と今回の趣旨

○ 前回は、空間に変数を設定して、「場」が影響力をもつ世界を再現してもらいました。課題では、消防士も登場しました。作成してみたでしょうか。

○ いよいよ artisoc の勉強も佳境です。今回は、複雑になっていくモデルを（複雑ではない我々の頭脳で）より自由にシンプルに書くための技法を勉強します。

○ 些末なようですが、mas という手法ではちょっとの違い（間違い？）が大きく結果に影響を与えます。ミスを無くすために、ルールをシンプルに書くということは、実はとても重要です。

○ 追加で、複数種類のエージェントが登場したときの実行順序の管理についても勉強しておきましょう。

☆ユーザー定義関数をつくる
☆実行順序を管理する

○今回は、「鳥の群のモデル」を作成します。Craig Reynolds が発表している Boids Model（1986 年）が有名です。ワールドロップ『複雑系』（新潮文庫）でも取り上げられているので知っている方も多いと思います。こんなことも artisoc を使って簡単に試せます。（彼が書いているより、もっとシンプルなルールで群が形成されることが分かります）

●［第一工程］モデルの準備をする

［01］ 空間や鳥エージェントをとりあえず用意しましょう。空間は名称 Sky、設定はデフォルトで構いません。鳥エージェントを空間のなかに作ります。エージェント名は Bird、数はとりあえず、10 羽作りましょう。あと、鳥エージェントに速さをあらわす変数（名称 Speed、実数型）を追加してください。

［02］ まず、最初にランダムに座標を与え、ランダムな方向に、ランダムな速さで飛ぶ鳥を作っておきましょう。慣れてきていると思いますが、

```
Agt_Init{
My.X = Rnd()*50
My.Y = Rnd()*50
My.Direction = Rnd()*360
My.Speed = Rnd()*0.3+0.2
```

```
}
Agt_Step{
Forward(My.Speed)
}
```

［03］ 忘れずに、出力設定（マップ出力）をしておきましょう。鳥のマーカーは△にしておくと、それっぽいです。（保存してから実行してみてください。）速すぎて描画や我々の動態視力が追いつかないときは、実行環境でウェイトをかけてましょう。

● ［第二工程］鳥に行動ルールを与える（１）

［04］ 今のところ、鳥は勝手気ままに自分の速度（速さと向き）を決め、前進しています。鳥が前進する前に少し、自分の速度を調節するルールを書いてみましょう。鳥の思考は、大きく二つの部分から構成されます。（A）周りの仲間の鳥に速度（速さと向き）を合わせる［同調］。（B）障害物（鳥を含む）があったら避ける［回避］。

［05］ まずは、［A. 同調］のルールから書いてみましょう。いろいろな書き方があると思いますが、ここでは、「周囲を（視野 4）で見回して、誰かいたら、そのなかから一羽に目をつけて、その鳥に速度（速さと向き）を合わせる」と書いていましょう。自分でモデルを作る日も近いので、できるだけ自力で書いてみてください。

＜ヒント＞

```
MakeOneAgtsetAroundOwn(Neighbors, 4, Universe.Sky.Bird, False)
CountAgtset(Neighbors)
Neighbor = GetAgt(Neighbors, Int(Rnd()*N))
```

などなど

［06］ では、試しに飛ばしてみましょう。鳥っぽいとは言えませんが、とりあえず群れは作ります。

● ［第三工程］鳥に行動ルールを与える（２）

［07］ 次に［B. 回避］のルールを書きましょう。今回の文法事項、ユーザー定義関数というものを学びます。これまで、Forward や MakeAgtset などなど、さまざまな関数を勉強してきました。artisoc ではこういった関数を自分で作ることができます。こういった自分で作った関数のことを「ユーザー定義関数」といいます。

［08］ 例えば、「ゴール前に味方がいれば、彼（彼女）をターゲットに高いボールを入れる」といった一連の行動を「ほうりこむ」という一つの関数にして定義しておけば、ルールのいろいろなところで「ほうりこむ」という関数をつかうことができます。（W 杯仕様の説明です。いまいちですね）

［09］ 今回は、「自分から角度 a 距離 b のあたりにいる障害物の数を数える」という関数を作ってみましょう。関数には（多くの場合）、入力値（引数）と出力値（返り値）があります。今回は、角度 a、距離 b が入力値、障害物の数が出力値ということになります。

［10］ 関数の定義は、ルールの{}の下に書いておきます。以下のような様式で書くことになっています。

Function 関数名(引数名 as 変数型, 引数名 as 変数型) As 返り値の型{

ルール内で使用する変数の宣言

関数の処理

Return(返り値)
}

［11］ 具体的には、以下のような書き方になります。これで、CountAgtAround(45, 10) とすれば、右 45 度距離 10 のあたり（視野は 1 とします）にある障害物の数を数えてくれます。

```
Function CountAgtAround(Angle as Double, Distance as Double) As Integer{
                                    //関数の名称は、CountAgtAround にします
                                    //角度（Angle）、距離（Distance）を入力値とします
                                    //出力値＝障害物の数は、整数型で返します
Dim Obstacles As Agtset
Dim Number As Integer
                                    //変数を宣言しておきます
Turn(Angle)                         //まず、与えられた角度だけ方向転換して
Forward(Distance)                   //与えられた距離だけ前に進んで
MakeAllAgtsetAroundOwn(Obstacles, 1, False)
                                    //そのへんにある全てのものを障害物として認識して
Number = CountAgtset(Obstacles)
                                    //その数を数えておく
Forward(-Distance)                  //与えられた距離だけ戻って
Turn(-Angle)                        //与えられた角度だけ方向を戻す
```

```
Return(Number)              //障害物の数を戻す（出力する）
}
```

● **［第四工程］鳥に行動ルールを与える（３）**

［12］ 今度こそ、鳥に回避のルールを書きましょう。ルールの内容は、「**自分の正面、距離１に障害物があれば**（CountAgtAround)、右前方（右 45 度、距離 1 の周辺）と左前方（左 45 度、距離 1 の周辺）の障害物の数を確認する（CountAgtAround)。そして、右前方の障害物数のほうが多ければ左へ、左前方の障害物数のほうが多ければ右へ方向転換する(45 度)。たまたま左右の障害物数が同じなら 30%減速する。**そうでなければ（=自分の正面、距離１に障害物がなければ)、**10%加速する。ただし、最大でも速さは 1 とする」としましょう。

［13］ 障害物がないと、盛り上がらないので、森を作っておきましょう。木エージェントを追加します（名称は Tree、本数は 50)。初期ルールに以下のように書いてみましょう。出力設定も忘れずにしておいてください。

```
MoveToCenter()
My.Direction = rnd()*360
Forward(10)
```

［14］ 鳥の飛び方に少しブレを作ってみても面白いです。(不可欠ではありません)

```
My.Speed = My.Speed+rnd()*0.1-0.05
If My.Speed > 1   then
        My.Speed = 1
End if
My.Direction = My.Direction + rnd()*10-5
```

● **［第五工程］森にも行動してもらう**

［15］森が何もしないのも色気がないので、鳥が近くに来たら、(さわさわとゆれて）色を変えるというルールを書いてみましょう。普段の色は My.Color = RGB(0, 150, 0)（濃緑)、さわさわゆれるときは、My.Color = RGB(200, 200, 240)（薄い青）などがお薦めかな。

```
Dim Birds as Agtset
MakeOneAgtsetAroundOwn(Birds, 1, Universe.Sky.Bird, False)
If CountAgtset(Birds) > 0   then
```

```
            My.Color = RGB(200, 200, 240)
    Else
            My.Color = RGB(0, 150, 0)
    End if
```

［16］これでモデルは、ほぼ完成です。非常に分かりにくいですが、ちょっとした欠陥があります。鳥が近くを通っても、木が色を変えないことがあるのです。これはルールの処理順序（実行順序）の問題です。

［17］気にならないことも多いですが、きちんとモデルを作るときには、注意しておきたい問題です。artisoc では、何も再設定しなければ、全ての種類の全エージェント（つまり、全鳥エージェントと全木エージェント）が毎回シャッフルされて、無作為な順番でルールを処理されていくことになります。（多くの場合、これで問題ありませんが、こうなっていることはよく覚えておいてください）

［18］今回の場合、ある木の処理が終わってから鳥が近くに飛んでくるとその木は色を変えません。次のステップで鳥が先に処理されて飛んでいくと、近くを鳥が通ったのに、木がそれに気づかない（ので、色を変えないという）ことも起こっているのです。このことを回避するために、鳥のルールを処理してから、木のルールを処理するというモデルに変更することにします。

［19］設定＞実行設定＞実行順序を選んで下さい。実行順序をランダムからエージェント種別設定を選択してください。ここで、エージェント種ごとに数字がふられているのが分かると思います。この数字が若い順にルールは処理されます。そして、数字が同じもの（と各同一エージェント種内）同士は、ランダムに処理順が選択されることになります。

［20］デフォルトでは鳥も木も「1」なので、全部の鳥と全部の木のなかから、無作為に一人ずつ選ばれてルールが処理されています。鳥のルールを先に実行し、木をあとで実行するには、木エージェントを選択して、下ボタンを押して下さい。木エージェントの数字が「2」になったと思います。これで。。

●まとめ

Reynolds は、（１）他の物体との距離を（最低限？）ある最小値に保とうとする［＝回避］（２）近隣のボイドと速度を合わせようとする［＝同調］（３）近隣のボイドたちの質量中心を知覚し、その中心に向かって移動しようとする［＝集中？］という３つの簡単な規則で、群が再現できるとしている。そして、どのルールも鳥に「群をつくれ」とは言っていないのに、群ができるのが驚きだと。（３）のルールは無くても群ができるのが驚き？

●課題

課題を三種類用意しました。課題 C は必須とします。来週提出してください。課題 B はなかなか歯ごたえがあると思います。トライしてみてください。

○課題 A　＜頭の体操＞いろいろな森を作って鳥をとばそう！！
（１）二重円のかたちに木が並んではえている森
（２）V の字に木が並んではえている森
（３）８の字に木が並んではえている森

○課題 B　＜もっと賢く！難問です＞森の狐の登場
群をなして飛んでいる鳥をねらう狐が登場しました。狐は基本的には止まっていて、群から孤立している鳥がいると、それを狙って（ターゲットとして）追跡します。ただし足は遅く、速さは 0.3 です。ターゲットが群のなかにもどってしまうと、追跡を止めてしまいます。距離が 1.5 以内になると、ターゲットの鳥を食べて、新たなターゲットを探します。

☆今まで習ったルールでほとんど書けます（半分ウソ）。
☆GetDirection(A の X 座標, A の Y 座標, B の X 座標, B の Y 座標, 空間名)で、AB 間の方角を計算させることができます。ループしている空間でも最短距離を結ぶ直線の方角を計算してくれます。
☆MeasureDistance(A の X 座標, A の Y 座標, B の X 座標, B の Y 座標, 空間名)で、AB 間の距離を計算させることができます。ループしている空間でも最短距離を測ってくれます。
☆ターゲットを追跡するためにはターゲットを記憶しておかなければなりません。ターゲットを記憶しておくための変数（エージェント集合型が便利）が必要です。
☆ターゲットを持っているときと、ターゲットがいないときで行動は全く異なります。
☆ターゲットの近くに来たら、KillAgt(エージェント)で、殺す（食べる）ことが出来ます。
☆（仕様の問題で）ターゲットを食べたら、記憶をクリアして（ClearAgtset(エージェント集合)）おいてください。

○課題 C　オリジナルモデルの作成準備
再来週から、皆さんに自作のモデルを作成してもらいます。来週までに、A4１枚以内（書式自由）で、どんなモデルを作りたいか、「モデルの構想」を書いて、次回の講義で提出してください。記名を忘れないでください。

A　同調

```
Dim Neighbors as Agtset
Dim Neighbor as Agt
Dim N as Integer

MakeOneAgtsetAroundOwn(Neighbors, 4, Universe.Sky.Bird, False)
N = CountAgtset(Neighbors)
If N > 0          then                          >>この部分に注意しましょう。
        Neighbor = GetAgt(Neighbors, Int(Rnd()*N))
        My.Speed = Neighbor.Speed
        My.Direction = Neighbor.Direction
End if
```

B　回避

```
Dim right as integer
Dim left as integer

If CountAgtAround(0, 1) > 0 then
        right = CountAgtAround(-45, 1)
        left = CountAgtAround(45, 1)
        If      right > left            then
                Turn(45)
        Elseif  right < left            then
                Turn(-45)
        Elseif  right == left     then
                My.Speed = My.Speed*0.7
        End if
Else
        My.Speed = My.Speed*1.1
        If My.Speed > 1   then
                My.Speed = 1
        End if
End if
```