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８回　複雑な属性を持つエージェント（12 月 2 日）

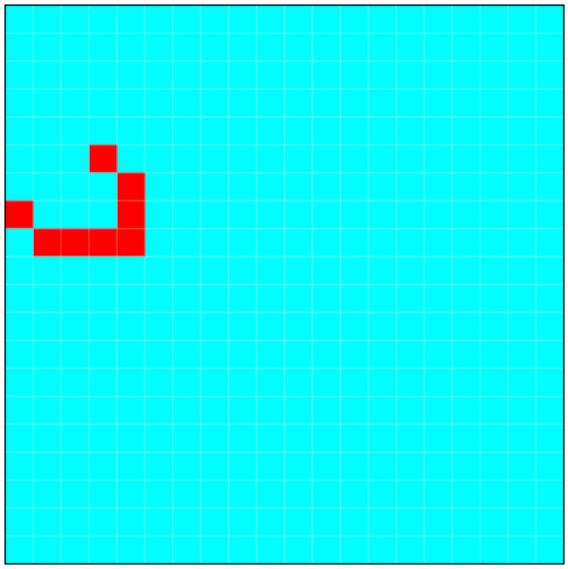
●概略

○ 前回の宿題の解答例と解説
○ 文字列による複雑な属性の表現
○ 文字列操作の基本：&による結合や Mid 関数など
○ ランダムなエージェントの取得：GetAgt 関数
○ 結果分析はじめの一歩：連続実行とファイル出力（時間があれば）
○ 宿題

●前回の宿題の解答例と解説

[４] 世界にはライフゲームを愛するマニアが多数います。インターネットで彼らのサイトを閲覧し、各自興味を持った初期配置（「グライダー」以外にも「へび」「宇宙船」「R ペントミノ」などいっぱいあります）でモデルを実行してみましょう（ただし無限増殖型の振る舞いなど、空間的な制約からダイナミクスを再現できない初期配置もあります）。コントロールパネルで初期配置を切り替えるようにできればなおよいでしょう。

＞たとえば次ページ図のような「宇宙船」の初期配置で始める場合、Univ_Init の初期状態決定部分のルールを条件文の地道な使用によって下記のように書き換えます。コントロールパネルで初期配置を切り替える場合は、If 文を使い、Universe の変数の値によって、実行される配置のルールが切り替わるように変更を加えます。

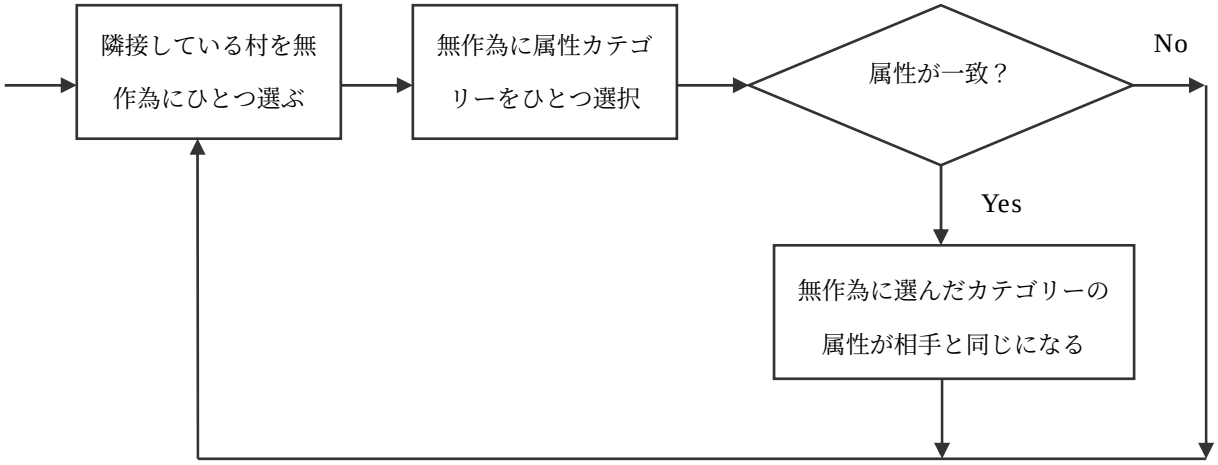


```
// 宇宙船型の配置
If cell.X == 0 Then
        If cell.Y == GetHeightSpace( Universe.Field ) - 8 Then
                cell.Color = Color_Red
        End If
ElseIf cell.X == 1 OR cell.X == 2 Then
        If cell.Y == GetHeightSpace( Universe.Field ) - 9 Then
                cell.Color = Color_Red
        End If
ElseIf cell.X == 3 Then
        If cell.Y == GetHeightSpace( Universe.Field ) - 6 OR cell.Y == GetHeightSpace( Universe.Field ) - 9 Then
                cell.Color = Color_Red
        End If
ElseIf cell.X == 4 Then
        If cell.Y <= GetHeightSpace( Universe.Field ) - 7 AND cell.Y >= GetHeightSpace( Universe.Field ) - 9 Then
                cell.Color = Color_Red
        End If
End If
```

●本日のモデルとそのポイント

今回は、政治学者ロバート・アクセルロッドが考案した「文化変容モデル」（本人は social influence model と言っています。詳しくは、著書 *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*, Princeton University Press, 1997, pp.145-177 を参照ください）を若干単純化したモデルを作ってみます。多様な文化的属性を持つエージェントが周囲との接触と相互作用を通じて、自らの属性をダイナミックに変化させていくモデルです。後述する文字列による文化の表現方法など、社会現象をモデル化する上で役に立つ様々な技法を学ぶことができます。

文化変容モデルのエージェントは、二次元の格子空間を敷き詰める仮想的な「村々」です。各村は、たとえば「言語はアラビア語、宗教はイスラーム、国籍はモロッコ…」など、複数の文化的・社会的カテゴリー（アクセルロッドは features と言っています）に関して、特定の属性（traits）を持っています。モデルでは、このような多元的な属性を持つ村々の間で下図のフローチャートに示す相互作用が展開されることになります。



シミュレーションは、各村の属性がランダムに決められた、極めて多様性の高い状態から始まります。以上のような単純な相互作用を展開させることで、この多様性にどのような変化が生じるのでしょうか。この問題を検討するのが文化変容モデルの主眼になります。

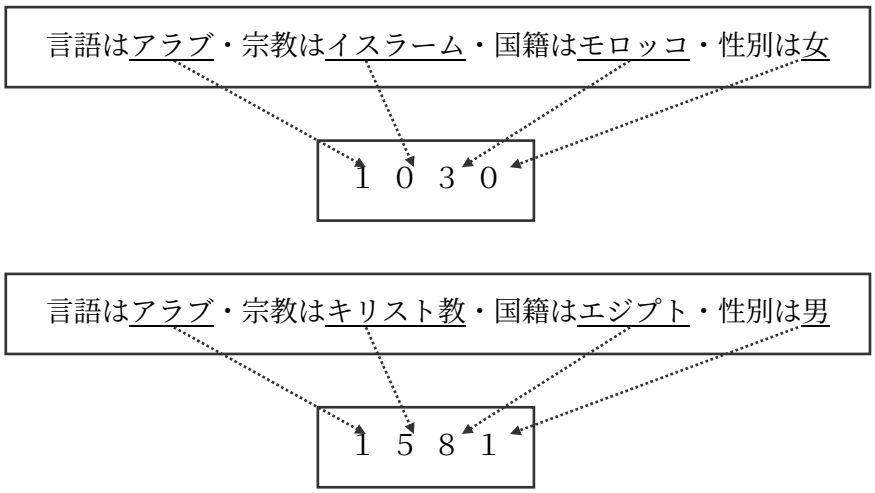
今回は、文化変容モデルを作りながら、以下のような文法や技法を学ぶことになります。

①文字列による属性表現とその操作

前々回の風邪引きモデルも、前回のライフゲームも、「風邪引き」「健康」、あるいは「生」「死」など、ある一種類の状態や属性に関する二値的な変化を表現するモデルになっていました。これに対して、文化変容モデルのエージェントの属性は、「言語に関しては□□、

宗教に関しては○○、…」など、多元的ではるかに複雑です。このような複雑な属性を扱いやすい形で表現する技法として、今回は、**文字列型変数による属性表現**の方法を学びましょう。

文字列型変数は、その名の通りテキスト（長さは任意）を代入するための変数です。この変数を用いると、エージェントの属性は、たとえば下図のような形で形式化することができます（「アラブ」「イスラーム」等はあくまで解釈例であり、数字は属性の差異以上のものを表してはいません）。アナロジーとして、遺伝子配列（ＡＧＧＣＴＴＡＡＡ・・・等）をイメージするとよいでしょう。



つまり、文字列（この場合は数字の列）の各桁が、言語、宗教など属性のカテゴリーを、「１」「８」といった文字が各カテゴリーに関してエージェントが持つ具体的な属性を表していると考えるわけです。文字列型変数なのでもちろん”Arab/Christian/Egypt/Male”などとしてもいいのですが、エージェントのルールの中で操作していく上で、数字の並びにしておく方が何かと便利です。

文化変容モデルを作るためには、エージェントの属性をランダムに決めたり、ある属性カテゴリーに関してエージェント間の属性値を比較したりするといった処理をKK-MAS上でルール化していかなければなりません。属性は文字列で表現されるわけですから、文字列型変数の様々な操作方法をあわせて学ぶ必要があります。具体的には、下記のような事項を、モデルを作りながら学んでいくことになります。

- **文字列操作の基本：””を用いた値の代入、＆による文字の結合**
- **変数型の変換：CStr関数（文字列型への変換）、CInt変数（整数型への変換）**
- **文字列操作のための組み込み関数： Mid関数（文字列中の文字の取得）、InStr関数（文字列中の文字列の検索）など**

②ランダムなエージェントの取得：GetAgt 関数

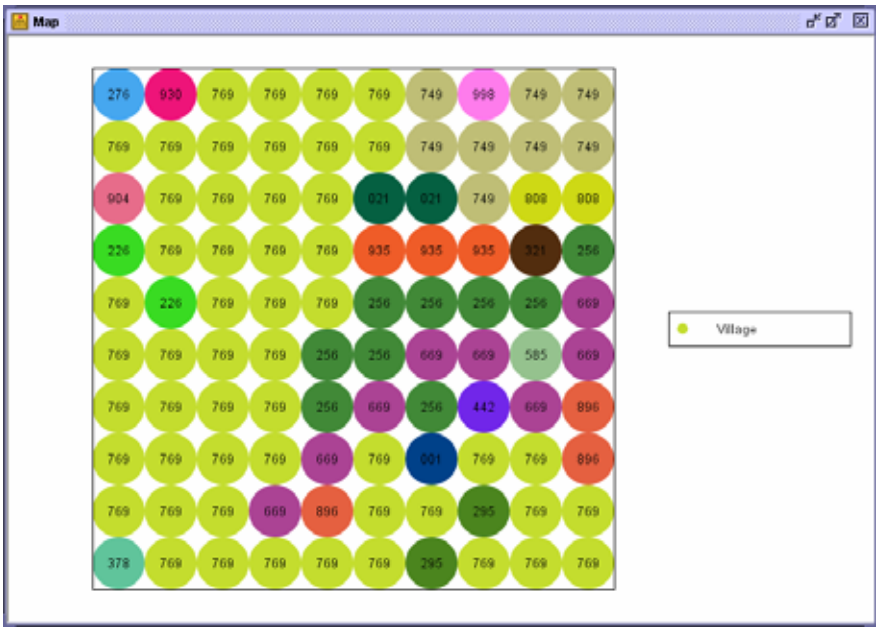
風邪引きモデルやライフゲームのエージェントは、MakeOneAgtSetAroundOwn関数等を用いて集めた近傍のエージェント全てに対して、For Each文を使って順次各々の情報を参照するという処理を行っていました。これに対して、上図のフローチャートが示すように、文化変容モデルのエージェントの相互作用は一対一の相対（あいたい）的なものです。つまり、毎ステップ、各村は、周囲の村々の中から無作為に選び出したひとつの村とのみ文化的な接触を行っています。このように一群のエージェントの中からランダムに一つを取り出すという処理は、マルチエージェント・シミュレーションのモデルでは頻繁に出てきます。KK-MASでは、このような処理を行うために、**GetAgt関数**が用意されています。今回は、この関数とその用い方についても修得しましょう。

③結果分析はじめの一歩：連続実行とファイル出力

上図のフローチャートでは「無作為」という文字が目につきます。つまり、文化変容モデルでは、一様乱数（Rnd 関数）を用いたランダムな要素が随所に組み込まれていることになります。このように、偶然性の入った非決定論的なモデルでは、あるシミュレーション試行で出てきた振る舞いが別の試行では再現されないということが往々にして起こります。たとえば、今回のモデルでは、同一の文化属性（同じ文字列）を共有する「文化圏」（cultural regions、次ページ図の同一色の部分）の数がどのように変化するのかを検証するのが大きな目的になりますが、上述のような偶然性の効いたルールのもとでは、試行ごとに出てくる文化圏の数のダイナミクスに大きなばらつきが出てくる可能性があります。

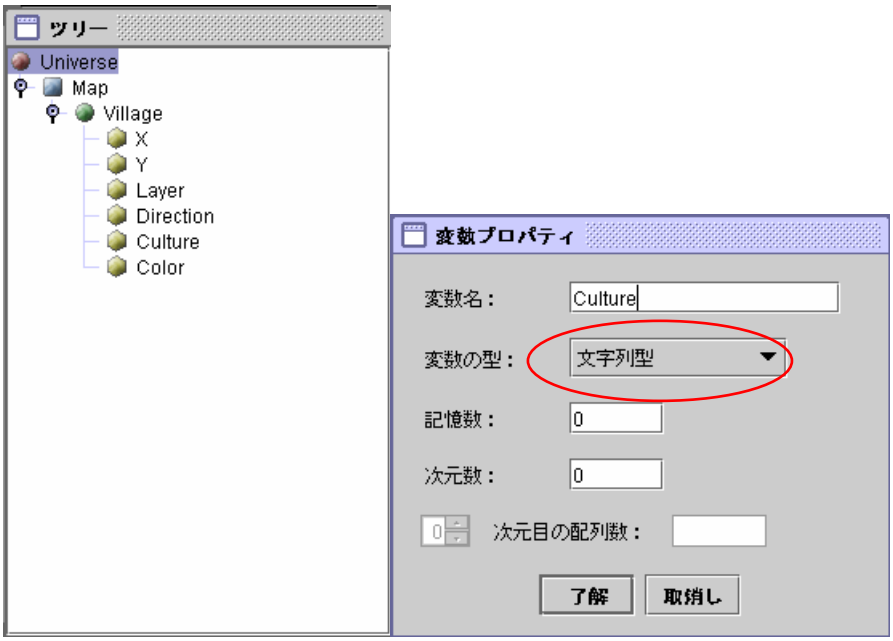
このようなモデルでは、シミュレーションを繰り返し実行し、結果の分布を統計的に分析する必要があります。つまり、偶然性を内包するエージェントの相互作用がもたらす「歴史」を繰り返し再生しデータを取るという作業が必要になるわけです。多くのシミュレーション・モデルは乱数を使った非決定論的な要素を含んでいるので、このような作業はシミュレーションを用いて本格的な研究を行っていく上で必須と言っていいものです。

今回は、こうした作業のうち基本中の基本となる**連続実行**と**ファイル出力**について学びます。前者は、シミュレーションを所定の回数繰り返し実行するものであり、KK-MASでは実行環境設定ダイアログを用いて簡単に設定することができます。後者は、連続実行の結果出てきたデータ（今回は文化圏の数）を外部ファイル（テキスト形式ないしCSV形式）に書き込む機能であり、データはExcel等の表計算ソフトを用いて分析することが可能です。手順はやはり簡単で、時系列グラフ出力の際とほとんど変わりません。出力するための変数をUniverseに作り、ルールの中で文化圏の数を数えて代入し、出力設定ダイアログでその変数がファイル出力されるよう設定するだけです。

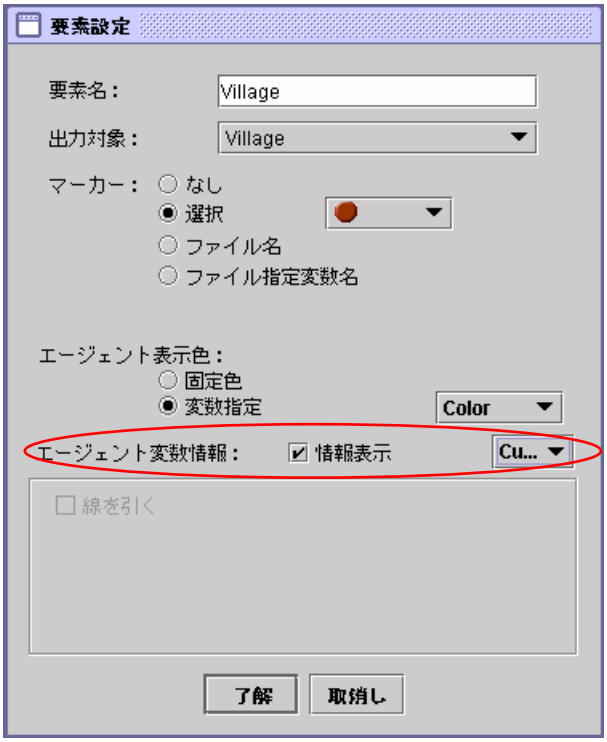


●モデル作り１：エージェントの生成と配置

　では、文化変容モデルの作成に取りかかりましょう。まず、10×10 の「ループなし」の空間（以下では Map）とエージェント（Village）を追加し、下図左のようなツリー構造を作ります。エージェントに追加する変数は、文化を表す変数（Culture）と同変数に連動して色を出力するための整数型変数（Color）の二つです。前者の変数型は下図右のように文字列型に設定します。



次にマップ出力の設定です。要素設定ダイアログで「エージェント表示色」を「変数指定」にして Color を選択します。文字列情報をマップ上に出力させるため、下図のように「エージェント変数情報」のチェックボックスをオンにし、プルダウンから変数 Culture を選択しておきましょう。



ここまでできたら、エージェントの生成と配置のためのルールを書き込んでいきます。

[1] 10×10 の二次元空間を Village エージェントで埋め尽くします。Village の文化は３つのカテゴリー（つまり３桁の文字列）を持ち、各カテゴリーについて 0～9 までの属性値を取りうるものとします。初期の属性値は各桁ランダムに与えます。表示色は文化ごとに違う色を割り振ります。

＞空間をエージェントで敷き詰めるので、前回学んだ For 文のネストによる配置方法を使ってもいいのですが、RandomPutAgtSet 関数を使って以下のように書く方がより簡単です。まず Univ_Init のルールを示しておきます。空間の大きさを変えてもルールを書き換える必要がないように、GetWidthSpace 関数等を使ってエージェント数を可変化している点に注意してください。それ以外は特に解説の必要はないでしょう。

```
Univ_Init{

        Dim i As Integer
        Dim Num As Integer
        Dim Set As AgtSet

        // 空間を敷き詰めるだけの village を生成
        Num = GetWidthSpace( Universe.Map ) * GetHeightSpace( Universe.Map )
        For i = 1 To Num
                CreateAgt( Universe.Map.Village )
        Next i

        // エージェントを集めてばらまく
        MakeAgtSet( Set, Universe.Map.Village )
        RandomPutAgtSetCell( Set, False )

}
```

空間の大きさに依存して生成されるエージェント数が変化

＞一方、Agt_Init の方は若干説明を要します。最初にルールを示しておきます。

```
Agt_Init{

        Dim i As Integer

        // ランダムに各 village の文化を生成
        My.Culture = ""
        For i = 1 To 3
                My.Culture = My.Culture & CStr( Int( 10 * Rnd( ) ) )
        Next i

        // 文化に応じて配色
        My.Color = ( Color_White / 1000 ) * CInt( My.Culture )

}
```

空文字列を代入して初期化

0～9 までの整数値を乱数生成した上で文字列型に変換し結合

000～999 までの値を取りうる文字列を整数化して色を配分

短いルールの中に文字列操作のための様々な手法が凝縮されて盛り込まれています。

まず冒頭の「My.Culture = ""」です。一般に文字列型変数に特定のテキストを代入する場合、「My.Nationality = “USA”」という具合に、代入するテキストをダブルクォーテーションで囲みます。「My.Culture = ""」ではダブルクォーテーションの間に何も書かれていませんが、これは要するに空の文字列を代入することで、Culture 変数を初期化することを意味しています。ちょうど整数型の変数で初期化のために 0 を代入するのと同じ操作を行っていることになります。

続く For 文で囲まれた部分は、初期化した空の文字列に、ランダムに生成した数字（0～9）を 3 回分付け加えることで、三桁の数字の並びをランダムに作り出しています。「My.Culture & CStr(Int(10 * Rnd( )))」の中にある「&」は数値演算の「＋」に相当する文字列の結合のための演算子です。たとえば「”A” & “B”」とすると、文字列”AB”が作られることになります。一方、同じ文中にある関数 CStr は、

CStr (変数名（整数型や実数型）)

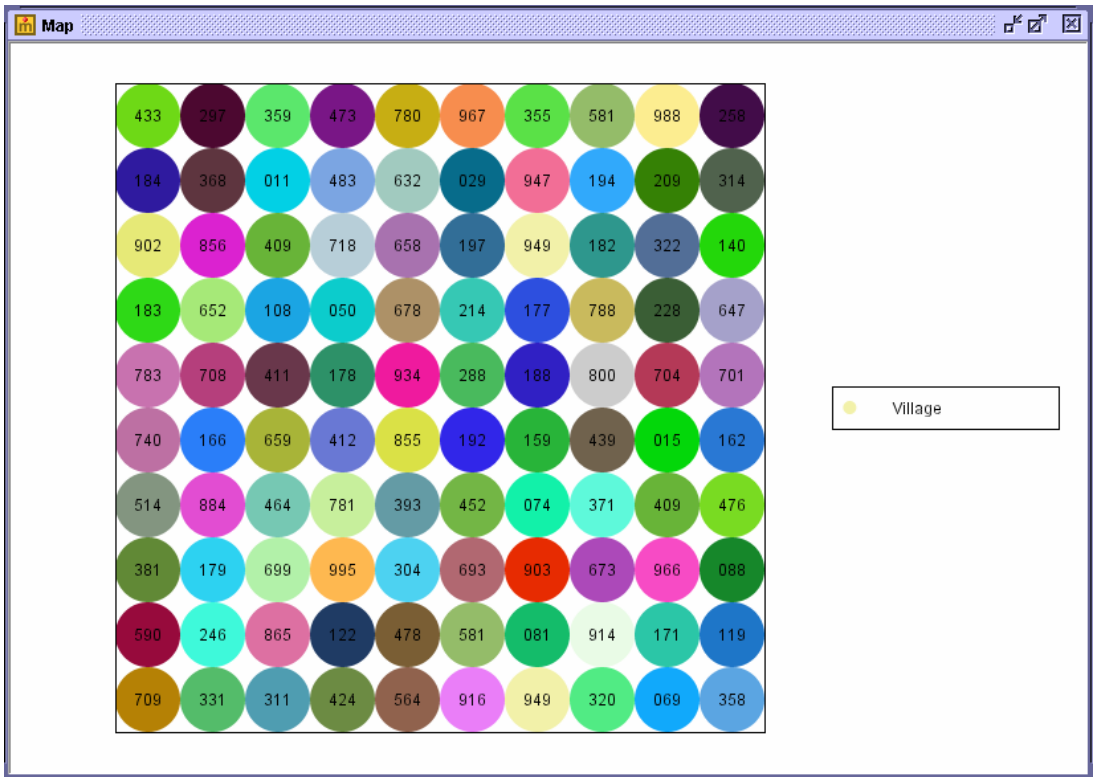
という書式で、引数の変数を文字列型に変換した値を返します。「Int(10 * Rnd( ))」で 0～9 の間の整数型の値が生成されますが、この値を文字列の中に並べるには、たとえば 3 を”3”という具合に、変数の型を文字列型に変換する必要があります。CStr 関数はこのような変換を行うための関数です。

最後に行っているのは、配色のための処理です。前々回各色には対応する整数値が割り当てられていると説明しましたが、このような数値のうち最大の値を取るのは、白色に対応する数値（16777215）になっています。最小値は黒色に対応する数値（0）です。上記のルールでは、白の値を予約語「**Color_White**」で取得して、Culture 変数が取りうる”000”から”999”までの 1000 個の文字列それぞれに、固有の色を表す整数値を黒から白までまんべんなく割り当てていることになります。このように、文字列を整数に対応づける過程で必要になるのが CInt 関数です。書式は、

CInt (変数名（文字列型や実数型）)

で、引数の変数を整数型に変換した値を返します。たとえば文字列”000”は、整数 0 に変換されます。CStr 関数とちょうど逆の型変換を行うことになります。

盛りだくさんな内容でしたが、ここまで書き上げて実行ボタンを押すと、次ページ図のような画面が出てくるはずです。



●モデル作り２：エージェントのルールの記述

次にステップごとの Village エージェントの行動ルールを、冒頭のフローチャートに従って、Agt_Step に書き込んでいきます。

[２] 各ステップ、Village は、視野１の範囲内にいる Village エージェントから無作為に一つ Village を選びます。もし、無作為に選んだ文化カテゴリーについて属性が一致していれば交流し、やはり無作為に選んだカテゴリーの属性が相手の属性と同じになります。それ以外の場合、接触は起きず、何も変化しません。

＞ルールは下記の通りです。

```
Agt_Step{

        Dim i As Integer
        Dim Num As Integer
        Dim Category1 As Integer
        Dim Category2 As Integer
        Dim CulStr As String
```

```
        Dim Neighbor As Agt
        Dim Neighborhood As AgtSet

        // 周囲の village からランダムにひとつピックアップ
        MakeOneAgtSetAroundOwnCell( Neighborhood, 1, Universe.Map.Village, False )
        Num = CountAgtSet( Neighborhood )
        Neighbor = GetAgt( Neighborhood, Int( Num * Rnd( ) ) )

        // ランダムに選び出した文化カテゴリーの属性が一致しているなら接触
        Category1 = 1 + Int( 3 * Rnd( ) )
        If Mid( My.Culture, Category1, 1 ) = = Mid( Neighbor.Culture, Category1, 1 ) Then

                // ↓接触する場合、ランダムに選び出した文化カテゴリーについて相手の属性と同じになる
                Category2 = 1 + Int( 3 * Rnd( ) )

                // 文化を表す文字列の再構成
                CulStr = ""
                For i = 1 To 3

                        // 変更するカテゴリーは相手の属性を引き継ぐ
                        If i = = Category2 Then
                                CulStr = CulStr & Mid( Neighbor.Culture, i, 1 )

                        // それ以外は以前のまま
                        Else
                                CulStr = CulStr & Mid( My.Culture, i, 1 )

                        End If

                Next i
```

```
                // 文化と配色の更新
                My.Culture = CulStr
                My.Color = ( Color_White / 1000 ) * CInt( My.Culture )

        End If

}
```

かなり長いルールになりましたが、フローチャートと見比べれば、各部分でどのようなことを行っているかおおよその見当はつくでしょう。文法的に新しい事項は二点だけです。

まず、MakeOneAgtSetAroundOwnCell 関数で集めたエージェント集合から一エージェントを取り出す際に、ポイント②で出てきた GetAgt 関数が登場します。前々回の資料の「おまけ」の部分（16 ページ）でも出てきましたが、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。「エージェント番号」は KK-MAS が内部的に割り振る番号であり、0 から Num-1（Num はエージェント集合に含まれるエージェント数）までの値を取ります。GetAgt 関数の第二引数の部分を Int(Num * Rnd( )) とすることで、エージェント集合の中からランダムに選ばれた一エージェントが返ってくることになります。なお、前々回指摘したように、エージェント集合が空集合の場合、GetAgt 関数はエラーを起こすので、If 文を使って集合の要素の数が 1 以上かチェックする必要がありますが、エージェントが空間を敷き詰める文化変容モデルでは、周囲のエージェントの集合が空集合になることはないので、このようなチェックは不要になっています。

次に、自分の属性と相手の属性とを比べる際に文字列操作関数 Mid 関数が使われています。この関数は、対象となる文字列の指定された位置から指定された文字数の部分文字列を取り出すための関数で、書式は以下の通りです。

```
Mid ( 対象となる文字列型変数 , 位置番号（整数型）, 文字数（整数型） )
```

位置番号は左端が 1、右端が n（n は対象文字列全体の文字数）となっています。上記ルール中に出てくる「Category1」「Category2」は、ランダムに生成された 1～3 の桁番号なの

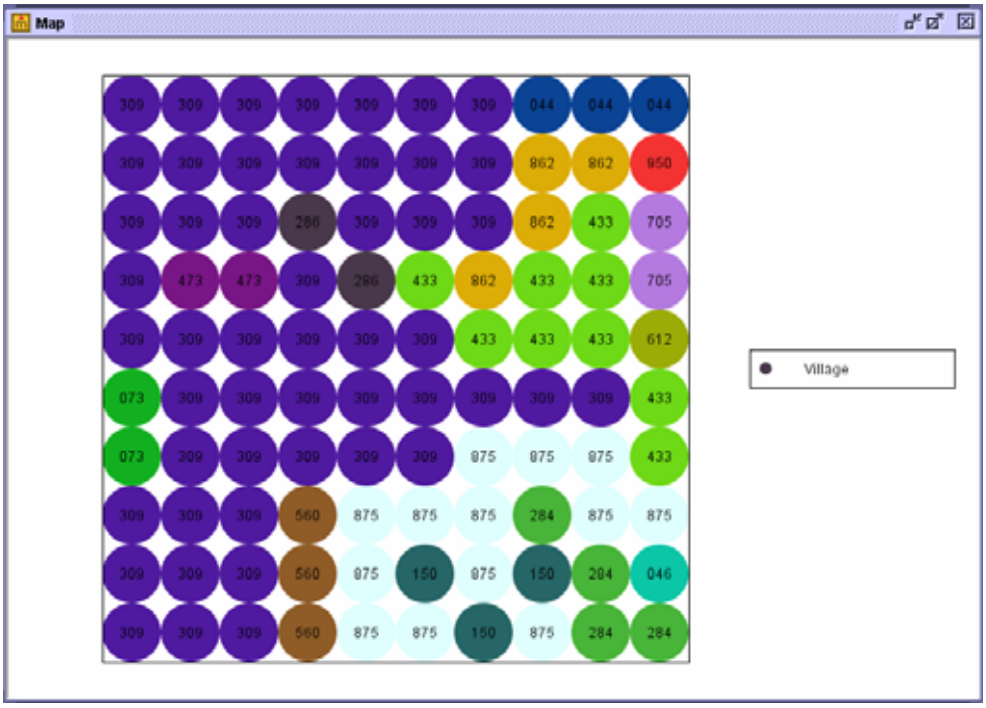
で、「Mid(My.Culture, Category1, 1)」などとすることで、文化属性を示す文字列の中からランダムに一文字が選ばれ返ってくることになります。これを使って、ランダムに選ばれた属性カテゴリーについて、自分の属性と相手の属性とが一致しているかどうかを判断することができます。なお、Mid 関数の変種として、下記のような関数が存在します。対象文字列の左端あるいは右端から指定された文字数だけ文字列を取り出す関数です。あわせて覚えておくといいでしょう。

Left (対象となる文字列型変数 ,文字数（整数型）)

Right (対象となる文字列型変数 ,文字数（整数型）)

文化的な接触に伴うエージェントの文化属性の更新の部分は、For 文と Mid 関数を組み合わせたやや技巧的なルールになっています。まず、変更する属性カテゴリー（Category2）をランダムに決めた上で、その部分だけ相手の属性値を引き継ぎ、それ以外の二桁は自分の属性値を引き継いだ３桁の文字列 CulStr を、For 文を使って新たに作り、最後に自分の Culture 変数に代入しています。For 文の参照変数 i を、「Mid(My.Culture, i, 1)」といった具合に、Mid 関数の引数として用いることで、文字列の左端から順次一文字ずつ取り出している処理がポイントになっています。

以上で、文化変容モデルの骨格は完成です。文化の斉一化はどこまで進むでしょうか。



●モデル作り３：「文化圏」の数のカウントと時系列グラフ出力

最後に出力機能を充実させましょう。冒頭でも述べたように、文化変容モデルの主眼は、初期の文化的にバラバラな状態が、Village エージェント間のローカルな相互作用を通じて、どのように変容していくのかを検証することです。そこで、同一の文化属性を共有する Village が成す「文化圏」の数をカウントし、それを時系列グラフに出力してみます。

時系列グラフへの出力方法については前々回学びました。まず、Universe 直下に文化圏の数を納める整数型の出力変数（Num_Cul）を追加し、次に出力設定ダイアログで時系列グラフを新規に追加して、この変数の値が出力されるよう要素設定します。その上で以下のような課題に取り組みます。

[３]各ステップ、Village エージェントが持つ文化属性の類型の総数を数えて変数 Num_Cul に代入します。

＞この処理を行うためのルールを書くには、若干の工夫を要します。このような課題では、様々な構文や関数を組み合わせて、特定の問題を解くための計算手続き（これを「アルゴリズム」と言います）を考えることがまず必要になります。

文化圏のように、ダイナミックに変化する属性の類型の数を数えるアルゴリズムは幾つか存在しますが、以下で紹介するルールは、そのうち最も単純で分かりやすいアルゴリズムを表現したルールになっています。基本的な発想は、”802/019/271/…”というふうに、各ステップにおいて存在する文化属性のリストを作りながら、その数を順次カウントしていくというものです。各エージェントの文化属性の文字列をリストと照合し、当該文字列がリストに存在しないならば、その文字列をリストに加えるという手続きを繰り返していくことになります。文字列とリストを照合する際に文字列を検索するための関数 InStr 関数を用います。

InStr (検索開始位置（整数型） , 検索対象文字列, 検索文字列)

返り値は、対象文字列中に検索文字列が存在するならば、その位置番号、存在しないならば 0 が返ってきます。この関数を利用して、Univ_Step_End に下記のようなルールを書き込んでいきます。

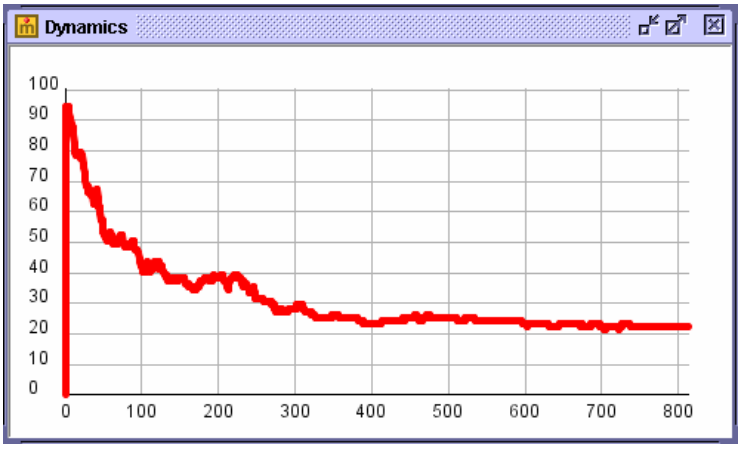
```
Univ_Step_End{

        Dim List As String
        Dim One As Agt
        Dim Set As AgtSet

        // ステップごとの文化数をカウントするので変数初期化を忘れずに
        Universe.Num_Cul = 0
        List = ""

        // 各 village の文化とリストの照合
        MakeAgtSet( Set, Universe.Map.Village )
        For Each One In Set

                // リストの中に village の文化がなければリストに加えカウント
                If InStr( 1, List, One.Culture ) == 0 Then
                        List = List & One.Culture & "/"
                        Universe.Num_Cul = Universe.Num_Cul + 1
                End If

        Next One

}
```

各ステップに存在する文化の文字列をつなげるための文字列型変数を宣言

リストの中に文字列が存在するかチェック

リストに新たな文字列を加えるたびにカウント

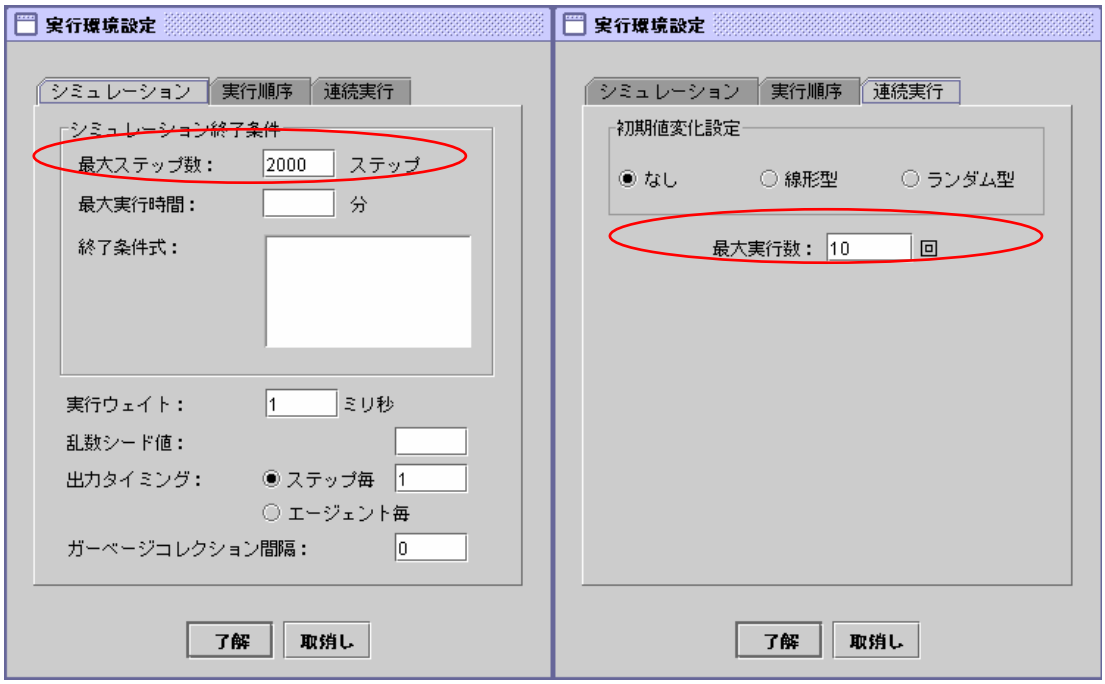
さて、実行ボタンを押してみましょう。下図のようなグラフが出力されたでしょうか。



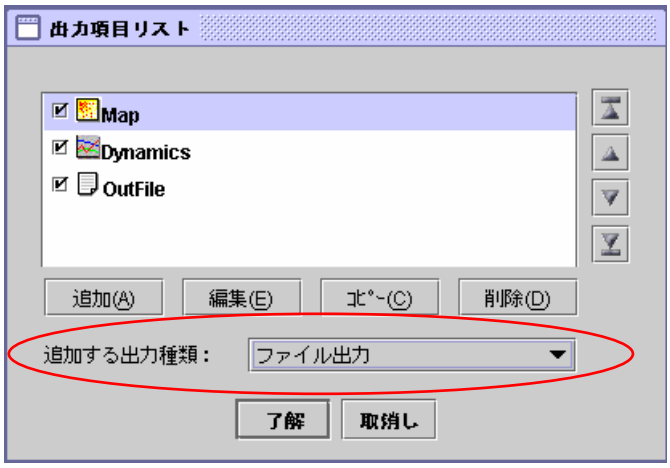
●モデル作り４：連続実行とファイル出力の設定（時間があれば）

ポイント③で議論したように、文化変容モデルのダイナミックスや長期的な傾向を知るには、シミュレーションを一回行うだけでは不十分です。そこで、まず、KK-MAS がシミュレーションを所定回数自動的に実行してくれるように、「実行環境設定」ダイアログで試行条件を設定することにします。2000 ステップの試行を 10 回繰り返すことにしましょう。

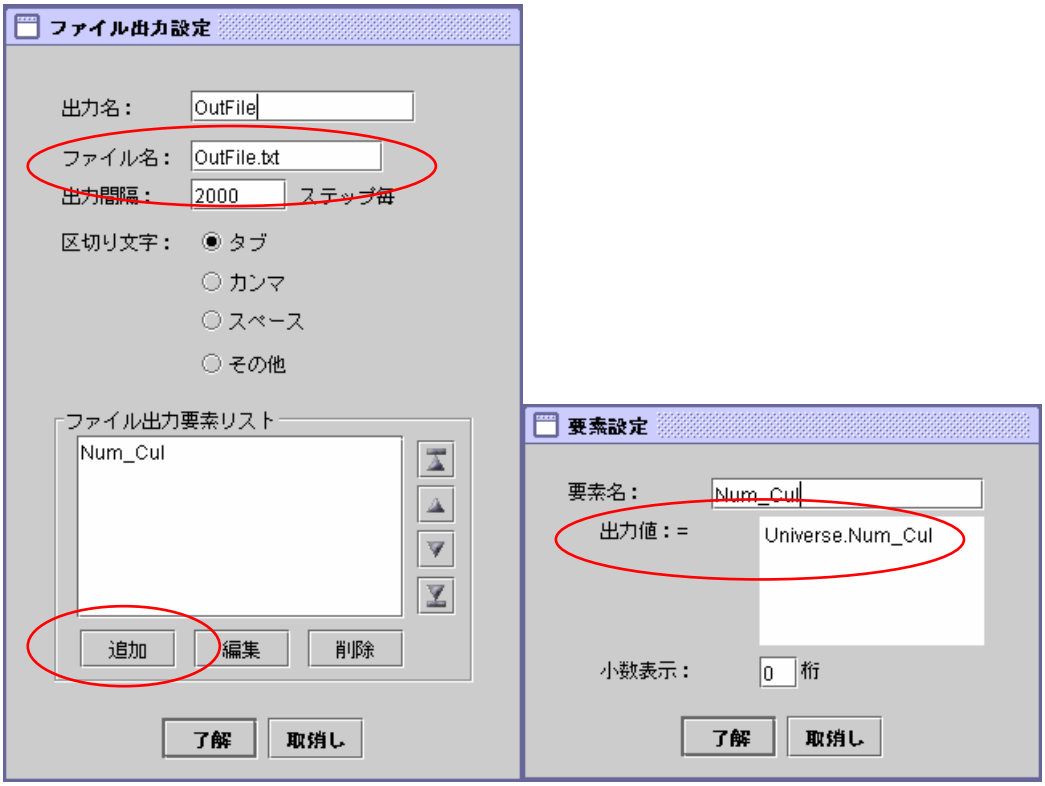
「実行環境設定」ダイアログは、メニューバーの「設定」から「実行環境設定」を選ぶことで開くことができます。最初に出てきた「シミュレーション」タグで、下図左のように「最大ステップ数」を「2000」に設定します。これにより各試行で 2000 ステップの間シミュレーションが実行されることになります。続いて「連続実行」タグを選び、ダイアログで、下図右のように「最大実行数」を「10」に設定すれば、一度実行ボタンを押すだけでシミュレーションが 10 回繰り返して実行されるようになります。



シミュレーションを繰り返し実行できるようになっても、各試行のデータが記録として残されないならば、分析の行いようがありません。最後に、各試行の終了時点（2000 ステップ）での文化圏の数を外部のファイルに出力する手順を学んでおきましょう。基本的な流れは時系列グラフ出力の場合と変わりません。まず、「出力設定」ダイアログを開き、「追加する出力種類」プルダウンから「ファイル出力」を選んで「追加」をクリックします。



続いて「ファイル出力設定」ダイアログで下図左のように設定します。「ファイル名」は任意ですが、拡張子に「.txt」を加えるようにしてください。これにより外部ファイルをテキストエディタや Excel などの表計算ソフトで開くことが可能になります。「出力間隔」は「1」ステップにしても構わないのですが、データが膨大になるので、「2000」ステップに変えて、試行終了時点でのデータのみが出力されるようにします。最後に「ファイル出力要素リスト」に、文化圏の数 Num_Cul を追加すれば作業終了です。設定は下図右の通りです。



実行ボタンを押してみるとどのような結果が出てくるでしょうか。たとえば全てのエージェントが一色に染まる完全な同化状態はどれだけの頻度で起こるのでしょうか。こうした点の検証を今回の宿題にしましょう。

●宿題

[4] 文化変容モデルのシミュレーションを 2000 ステップ 10 試行実行し、試行ごとに文化圏の数がどのような値に収束するか、データを外部ファイルに出力します。表計算ソフト等を用いて 10 試行の文化圏の数の分布をヒストグラムで表したり、平均値を求めたりしてみましょう。のべ 20000 ステップの試行はかなりの時間を消費するので、シミュレーション実行中は別の作業に取り組むことをおすすめします。

[5] 文化属性のカテゴリーの数をパラメータ化しコントロールパネルで操作できるようにした上で、カテゴリー数を 3 から 5 に増やして、[4]と同じような繰り返し試行を行ってみてください。ルールに加える変更は、カテゴリー数の数値を Universe に加えた変数で置き換えることと、配色の際の文字列から整数への変換式を書き換えることです。後者は若干の工夫を要します。

以下は余力のある人向けのかなり上級な課題です。

[6] 文化変容モデルには、これ以上文化属性の分布が変化しなくなる定常状態が存在します。つまり、全ての Village エージェントについて、周囲の全てのエージェントの属性が全く同質か、または逆に全く異質な状態が成立しているならば、文化の変容は事実上起きなくなります。今回作ったモデルではシミュレーションの終了ステップを実行環境設定で外生的に決めていましたが、この論理をもとに、システムが定常状態に達したらシミュレーションが勝手に終了するようにモデルに変更を加えてみましょう。ルールの中でシミュレーションを終了させる関数として ExitSimulation( )（引数なし）という組み込み関数が存在するので、これを利用します。