マルチエージェントを用いた情報の利活用による帰宅困難者支援効果の検証の基礎的研究

計画マネジメント・皆川研究室

0718026

小泉祐亮

目次

第１章　序章

１．１　研究背景・研究目的

日本は世界でも有数の地震大国といわれており，世界の地震エネルギーの 10 分の 1 が日本周辺に集中しているとも言われている．そして地震の脅威はおさまることを知らず，今後は東海地震・東南海地震・首都直下型地震などの様々な地震の発生が予測されている．地震の発生を防ぐことは出来ないが，被害を軽減する方法であれば，様々な可能があると考えられる．本研究では震災時の歩行者の避難シミュレーションに着目する．研究は既に行われているが，現在では情報技術の進歩と共に携帯・パソコン情報端末から様々な情報を容易に手に入れることが出来る．そこで，それを考慮し被災者が効率的に避難できるための情報を被災者に与えることも可能である．そのために ICT を用いた帰宅者支援システムを踏まえ，震災時に被災者に効率的に避難できる情報を時間経過と共に与え，その動きをシミュレーションする．このように様々な情報を容易に手に入れることが出来る現在，情報利用による震災時の避難シミュレーションを行い，帰宅困難者支援のマルチエージェントによる有効性の検証することが本研究の目的である．

１．２　研究の流れ

研究を行うにあたって２つの項目に分けました．

１つ目は,「システムの内容の作成」です.今回,情報社会に着目した上での研究なので,地震発生直後に必要な情報の提供・受け取り・データ整理などを行うことのできるシステムを考える．

２つ目は,「シミュレーションを行う」です.今回の研究内容に適したシミュレーション行うために artisoc を利用して,シミュレーションを行う.必要なものとしてエージェントのルールがあり,ルールを書き込むことでシミュレーションを動かすことができる．

この２つを今回の研究の軸とし,研究を進めていき情報の利活用による帰宅困難者支援効果の検証を行う．

１．３　研究においての様々な概要

１．３．１　なぜ歩行者エージェントシミュレーションなのか

歩行者エージェントシミュレーションが現在なぜ注目を浴びているか.実用面と理論面から５つあげる．

実用面

エージェントとは,もともと計算機科学において「ひと」のアナロジーとして成立した概念である.シミュレーションとは,一組の仮定群から導き出される帰結を表現するツールであるが,人間行動を扱うエージェントシミュレーションは,これらの仮定群を従来よりも分かりやすく提示するとともに,その帰結をビジュアルに示す点に興味を持っている．

①群集事故リスクの予測と対策の事前検討

現実を模擬した人間行動モデルと空間モデルを用いてシミュレーションを行なうことにより,局所密度が上昇しやすいよう注意点を洗い出したり,群集誘導策を検討することが可能である．

②大規模施設における非常時の避難方策の検討

歩行者エージェントシミュレーションにより,商業施設・オフィスや集客施設屋内における火災等の非常時における避難方策を検討することが可能である.

③商業空間・賑わい空間における快適な歩行空間デザイン

都市デザイナーが計画する歩行空間の安全性や賑わいを事前評価し,また,賑わいイベントや広告アナウンスの効果を事前検討するツールとしての可能性が注目されている.

理論面

複雑系科学がもたらす知の地平の開拓が期待されている.

④複雑系科学による群集事故の発生メカニズムの解明

90 年代以降,複雑系科学では,層流形成やボトルネック部における対向流の振動的流動といった基礎知見のみならず,脱出パニック状態で性急な群集行動がもたらすボトルネック部での閉塞や層流秩序の崩壊といった群集事故発生メカニズムの理論モデルを提示し,注目を浴びた.

⑤自己駆動粒子のシステム特性の解明

前方に空隙があれば前進し他物が存在すれば停止するが,他物に「作用」を与えることはない～このような粒子はもはやニュートン物理学の対象となりえず,自己駆動粒子と称する概念が提唱された.自己駆動粒子は,歩行者の行動モデルに留まらず生体内のある種のたんぱく質の挙動モデル等,自然界でも用いられるシステム概念とされ,特にこれらのシステム特性の解明が期待される.

１．３．２　歩行速度

自由歩行速度には５項目の条件がある.

①肉体的条件：性別,年齢,身長,体重,健康状態

②集団的条件：グループの人数,種類,群集密度,対向者の有無,横断者の有無

③服装条件：被服,履物,携帯品の種類と重量及び形状

④心理的条件：歩行目的,感情,場所の熟知度

⑤環境条件：季節,天候,気温,風向,時刻,歩行路面の材質と形状,勾配,明るさ,視界

①肉体条件

年齢,性別などの影響：年齢別で最も歩行速度が大きいのは,青年男性と高校生であり,以降中学生・壮年男子,青年女子,小学生,壮年女子,老人の順になる.性別では男性が女性より速い.

②集団的条件

グループ歩行による影響：２人連れの場合,お互いの速度が違っている時にはお互いの速度の中間の歩行速度で歩行することが多い.３人連れの場合は,最も遅い歩行者よりやや速い速度に落ち着く.この傾向は４人以上のグループでも同様である.一般にグループの歩行速度は,単独歩行に比べて遅くなる.

③服装条件

荷物や子供などの影響：荷物の有無,子供連れなどの様々な状況における歩行速度を測定したとき,荷物を持

った人の歩行速度は荷物を持たないで単独歩行する場合とでは差が見られない.およそ 12kg までの荷物であれば歩行速度に影響がないとされている.子供連れの場合の歩行速度はベビーカーを使っている場合,抱っこやおんぶしている場合ともにグループ歩行の速度に近い.

④心理的条件

心理的条件の影響：切羽詰った状況での歩行速度は速いと考えられる.横断歩道を横断中に青信号から点滅,赤に変わる際の歩行速度は,青信号開始直後に比べて２倍の速さである.

⑤環境条件

勾配の影響：5%までの緩勾配ならば歩行速度に影響は無く,それ以上の勾配では勾配に比例して歩行速度は遅くなる.階段場合は傾斜が 5%増すごとに歩行速度が 0.1m/秒遅くなる.

時刻の影響：午前中の出勤時は歩行速度が速くなり,午後は散歩のようにのんびりと,そして夕方の帰宅時はまた歩行速度が速くなる傾向.

煙の影響：煙の種類により歩行速度に大きく影響を与える.刺激性の少ない黒煙の場合,歩行速度は煙の濃度を表す減光係数が増すにつれて直線的に減少する.刺激性の強い白煙の場合,目を長時間開けることが困難なため歩行速度は急激に低下する.

１．３．３　群集流の形状

群集流は,その形状に応じて一方向流,対交流,層流,交差流,交錯流,追越流に分類される.

①一方向流：一方向のみの歩行者の流れ.通勤・通学路で見られる.

図１－１　一方向流

②対向流：通路などにおいて,２つの方向の群集流が向き合う状態.通勤時における乗換駅の階段が典型.

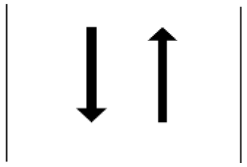


図１－２　対向流

③層流：通路などにおいて,互いに向き合う二方向・３つ以上の群集の流れが存在する状態.人通りの多い横断歩道における対向流がすれ違う際に形成することが多い.

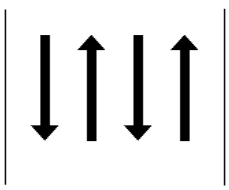


図１－３　層流

④交差流：交差点などで,二方向の群集の流れが交差する状態.地下街の交差路や十字型交差点におけるスクランブル式の横断歩道で見られる.

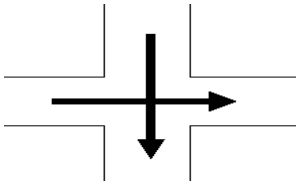


図１－４　交差流

⑤交錯流：多方向の群集の流れが交錯している状況.買い物客の流れなどで見られる.

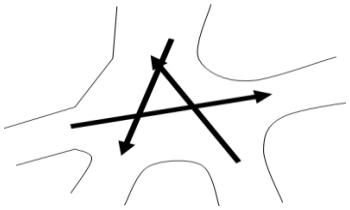


図１－５　交錯流

⑥追越流：歩行速度の遅い群集に対して,歩行速度の速い群集が追い越す状態.

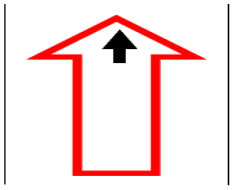


図１－６　追越流

１．３．４　地震発生後の人々の帰宅行動

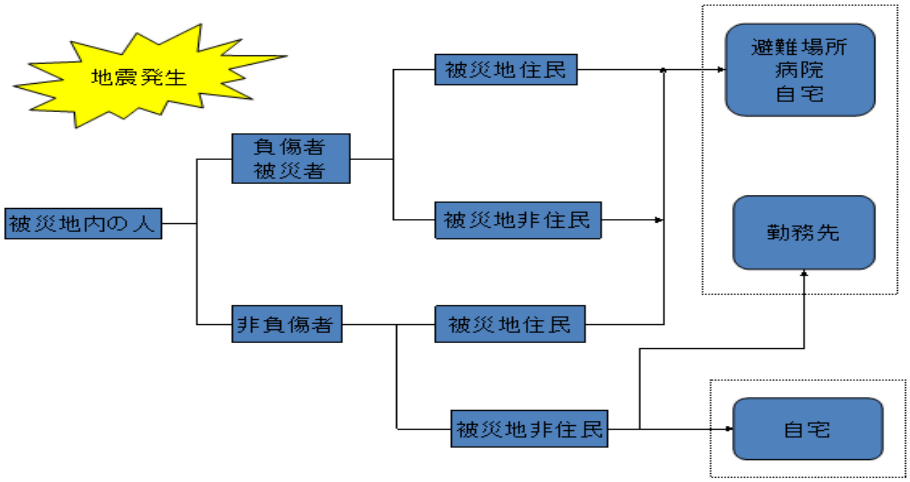


図１－７　帰宅行動

１．３．４．１　帰宅困難者

大規模災害により交通機関が運行を停止し,自宅が遠隔なため帰宅をあきらめる人々や,一旦徒歩で帰宅を開始したものの途中で帰宅が困難となり,保護が必要になる人々のこと.つまり,大地震等で交通機関がマヒしてしまった場合に,職場,学校,買い物先等から自宅へ帰れなくなってしまう人々のこと.

１．３．５　想定すべき首都直下地震

東京湾北部地震

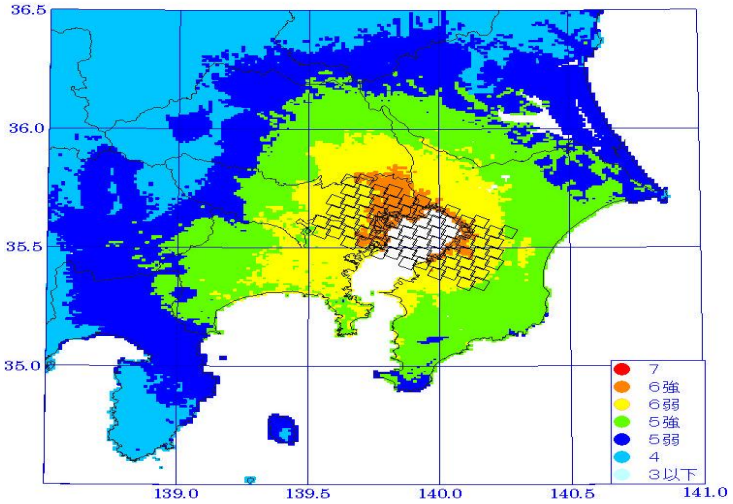
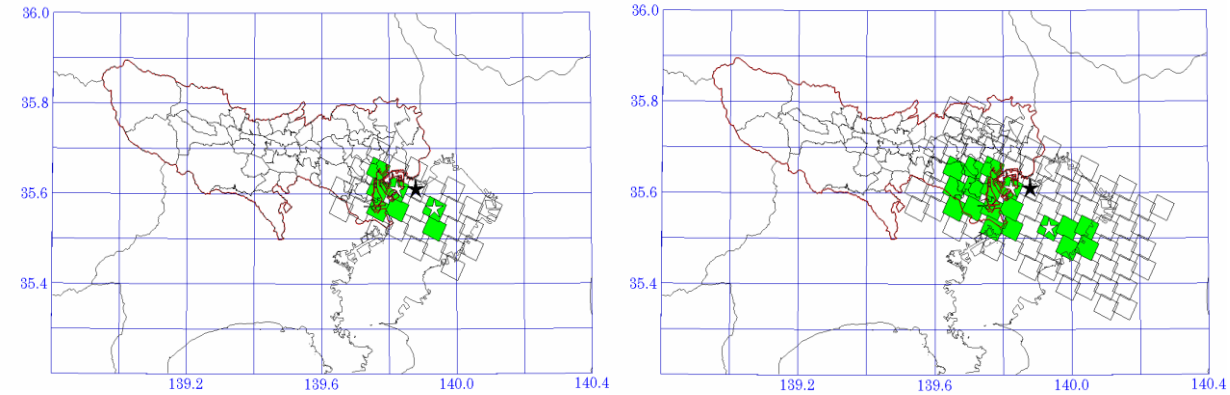


図１－８　東京湾北部地震



この首都直下地震を想定したシミュレーションを作ることが理想である.

１．３．５．１　交通

（１）細街路の閉塞

【幅員 3.5m 未満の道路】

道路閉塞率（%）＝0.9009×建物被災率＋19.845

【幅員 3.5m 以上 5.5m 未満の道路】

道路閉塞率（%）＝0.3514×建物被災率＋13.189

【幅員 5.5m 以上 13m 未満の道路】

道路閉塞率（%）＝0.2229×建物被災率－1.5026

（２）道路橋梁・橋脚の被害

道路橋梁・橋脚被害率

旧基準に準拠（耐震性低）：被害大（機能支障あり）＝8.2%

被害中小（機能支障なし）＝33.9%

新基準に準拠（耐震性高）：被害大（機能支障あり）＝0.0%

被害中小（機能支障なし）＝16.3%

第2章　帰宅困難者支援システム

２．１システムの概要

帰宅困難者支援システムとは,地震発生時に時間帯や自宅の距離など検証し,歩行で自宅に帰ることのできない帰宅困難者を支援するためのシステムである[3)4)].このシステムは主に,帰宅困難者への情報提供,利用者から情報を取得することで機能する.

２．２ システムの内容

２．２．１　システムが扱う情報

被害写真・コメント・地図情報・避難場所・地震情報・避難ルート情報・GPS 情報等

２．２．２　システムの役割

①歩行者等からの情報取得

主に歩行者等から 2.2.1)で示した情報を取得する.取得方法としては,カテゴリー（被害状況,火災,渋滞,死傷者）別による情報取得・写真,コメントによる情報取得を考えている.

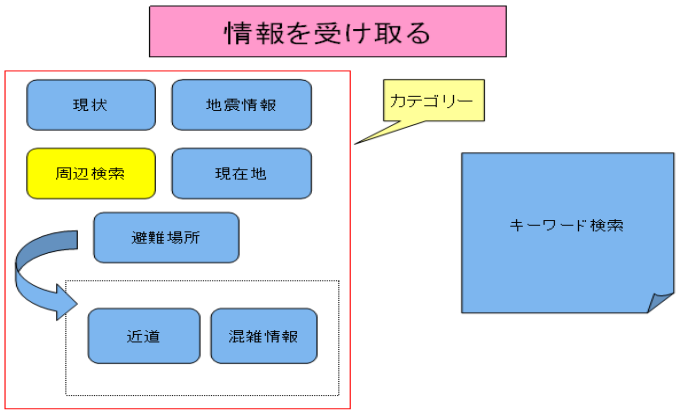


図２－１　情報の取得

②歩行者等への情報提供

このシステムは誰でも利用可能であることを前提に主に歩行者等へ情報を提供する.提供方法としては,カテゴリー（地震情報,現在地,避難場所,周辺）検索・キーワード検索・周辺検索などによる情報提供を考えている.

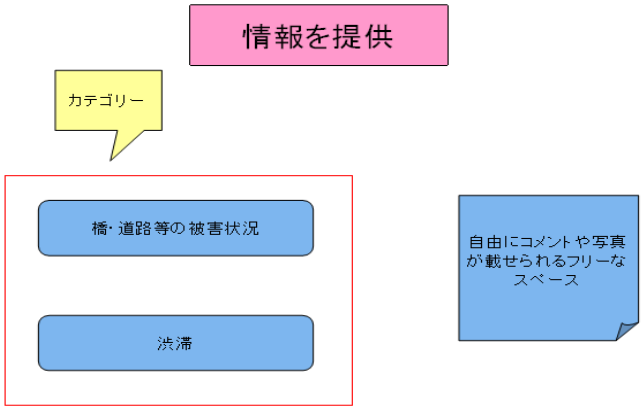


図２－２　情報の提供

２．２．３　システムの位置づけ

このように帰宅困難者が簡単に利用できそこから自治体や公共施設と連携できる位置

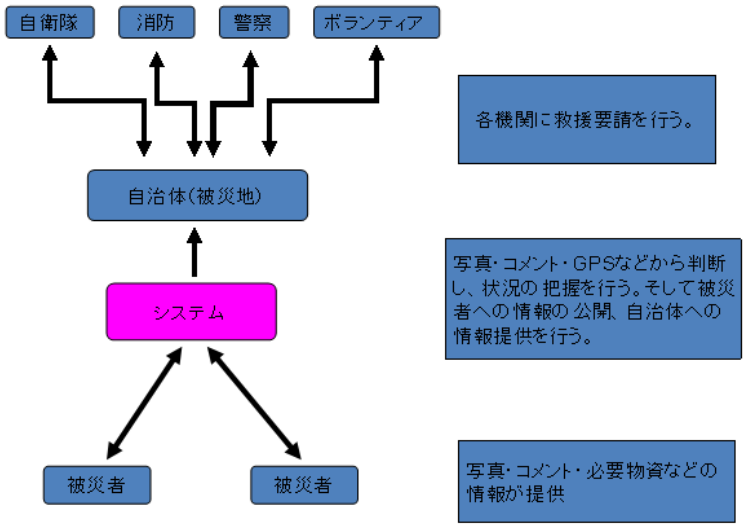


図２－３　位置づけ

２．２．４　システムの利用率

情報の利用率の変化により避難シュミレーションにどのような変化が生じるかを見るために，このシステムの利用率を設定し，0～100%の間で数値を自由に変えられるようにする．この際に，歩行者エージェントからの情報の取得と歩行者エージェントへの情報提供をシステムの利用率として一緒に考える.例えばシステムに情報を提供しかつシステムから情報を取得することと，システムから情報を取得することを，同じく情報利用とする.

２．２．５　システムの利用する機器

今研究ではシステムを利用する機器として「携帯電話」「ノートパソコン」としている.理由としては,普及率がとても高く日本は 79.6%（2006 年国連貿易開発会議）である.また,軽量であるため持ち運んで使うことが多く,身に着けていることが多い.そして,震災時に起こる機器等の被害で転倒や落下,移動が大半を占めている中,被害率がホストコンピュータが 26%であったのに対し,ノートパソコンはわずか 1%と,小型軽量になるほど被害は軽微になることが分かっているからである.

第3章　シミュレーション

３．１ マルチエージェントシステム

マルチエージェントシステムとは，まず多数の自律的に行動するエージェントから構成されるシステムであり，それぞれのエージェントは自分の目標を達成するように動き，システム全体の振る舞いはエージェント同士が相互に作用することによって決定される．つまり，複数の人々の動きをシミュレーションするには最適なシステムである．

３．１．１　シミュレーターの選択

マルチエージェントシステムを応用したソフトが様々ある上で，本研究では artisoc を用いてマルチエージェントシステムの技術を学び，自分の研究に活用していく．

３．１．２　artisoc とは

artisoc とは,KK-MAS の使いやすさや汎用性はそのままに,新機能を追加しマルチプラットフォーム（Windows や Mac など JAVA2 が稼働する環境）に対応した新世代の KK－MAS です.

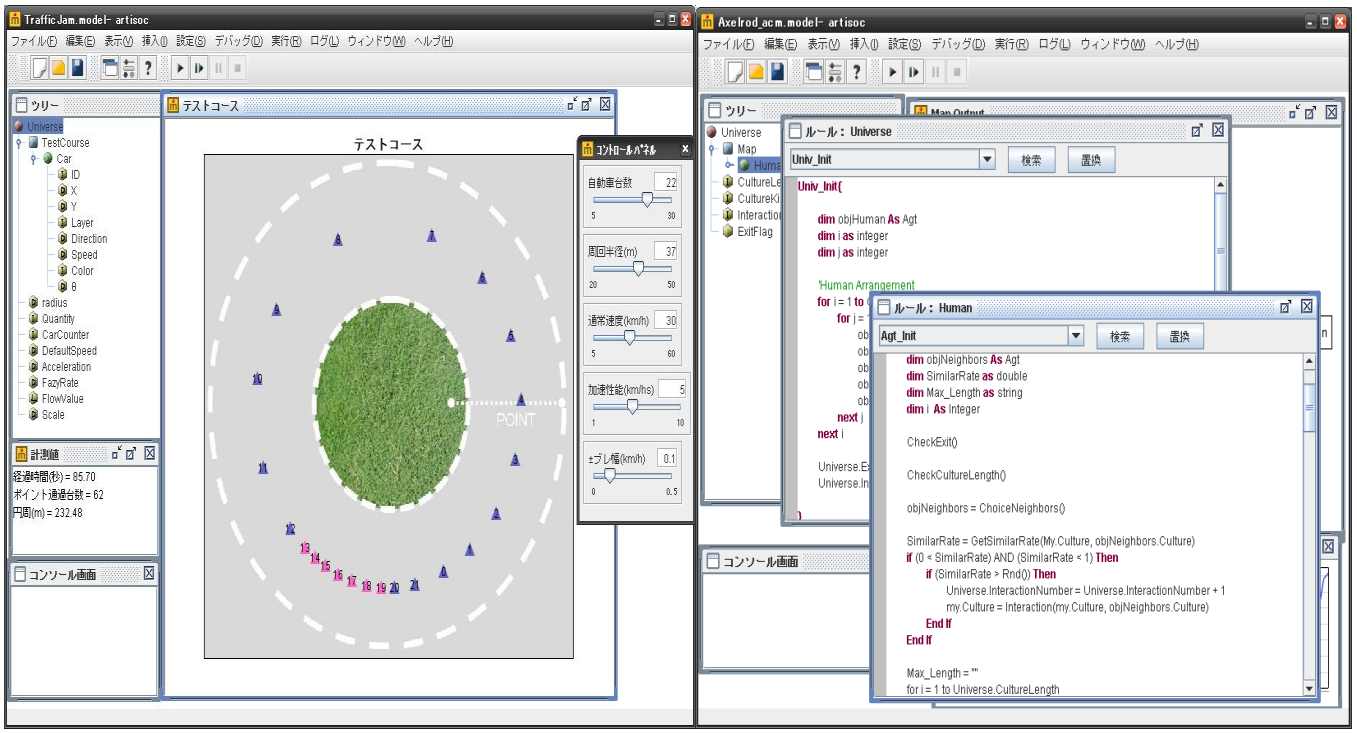


図３－１　artisoc

・KK-MAS とは

マルチエージェントシミュレーション（mas）は,社会科学における研究技法として非常に高い可能性を持っています.しかし,mas を用いた社会科学の研究は,今日まで十分に広がっているとは言えません.その大きな原因のひとつは,コンピュータ言語を用いたプログラミングが必要とされるという,一般の社会科学者が直面する

シミュレーションの敷居の高さにあります.この敷居を大幅に下げるべく開発されたのが,㈱構造計画研究所制作のマルチエージェント・シミュレータ　KK-MAS です.KK-MAS は,コンピュータ言語によるプログラミングを必要としません.一定の文法に従って,エージェントにルールを与えてやることで,マルチエージェントシミュレーションによるモデルを作成することが可能です.また,日本語環境が整っていることは,日本語ユーザーにとっては,大きなメリットとなります.
一方で,ルールで可能な表現範囲は非常に広く,汎用性も欠けていません.多くのサンプルモデルや研究用に我々が作成しているモデルを見て頂ければ,その汎用性の広さは理解していただけると思います.

３．２　マルチエージェント・シミュレーション

３．２．１　マルチエージェント・シミュレーションの誕生（分居モデル）

「分居モデル」は,トマス・シェリングが 1969 年に発表した論文で提唱したものである.彼が注目したのは,地域社会が人種毎に分居する傾向にある現実が住民の排他意識とどのような関係にあるのかという点である.しかし,シェリングが用いた方法は住民の意識ではなく,サイコロを振りながらの机上の実験でした.シェリングの発想は,ミクロレベル（人々の個人的な好みや行動様式）から推測できることが,必ずしもマクロレベル（社会全体の有り様）に反映するとは限らないことを定式化したもので,マルチエージェント・シミュレーションの発想そのものである.

３．２．２　３要素

シミュレーションを実際にやってみるには,少なくとも３つのプロセスが必要である.

①シミュレーションをするためのモデルを作る
②モデルを実際に動かす
③実行の過程・結果を見る

そしてこの３つの要素に対応して次の準備が必要である.

①モデルの枠組みの設定
②シミュレーションを実行させる環境の設定
③実行過程を見るための出力表示の設定

３．２．３　artisoc の基本

人工社会のモデルは次の５つを基本的な要素にしている.

「エージェント」と呼ばれる行動主体
個々のエージェントの属性（性質や役割）を表す「変数」
エージェントが行動する（他のエージェントと関係する）「ルール」
エージェントが行動（相互作用）する「空間」「場所」
シミュレーションやモデル全体に関わる「変数」「ルール」

第4章　シミュレーションの基礎的研究

４．１　シミュレーションの概要

今回行ったシミュレーションは,マルチエージェントを用いた情報の利活用による帰宅困難者支援効果の検証を行うために必要な歩行者エージェントの基礎的シミュレーションである.今回は歩行者エージェントがどのような判断で目的地に達するかを行う.

４．２　シミュレーション

４．２．１　シミュレーションを動かす前

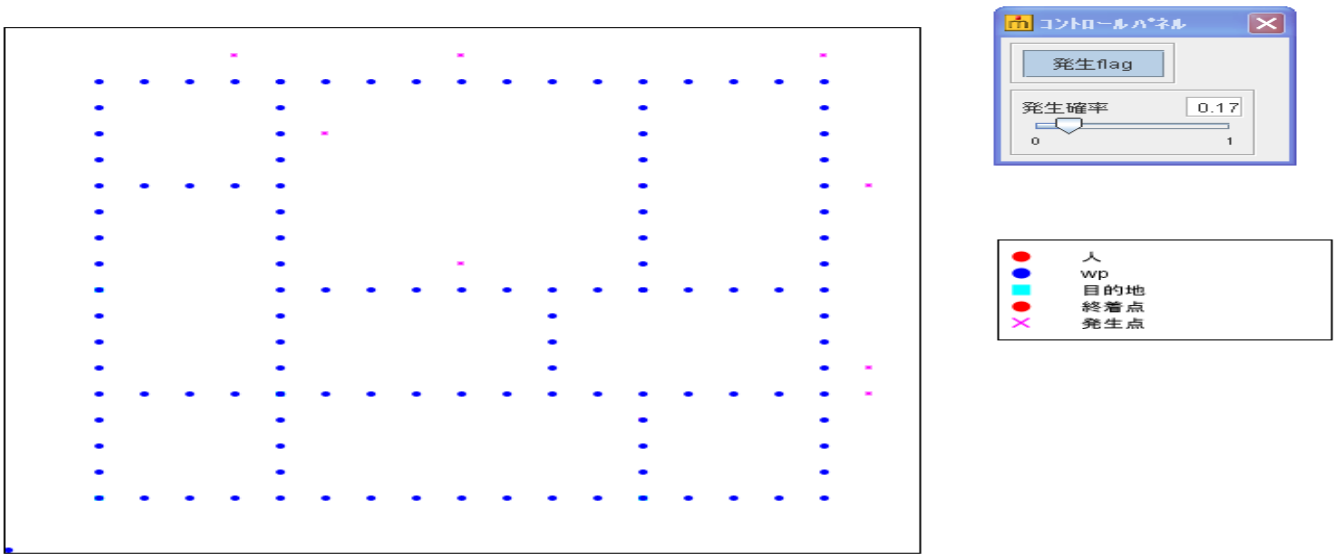


図４－１　シミュレーション起動前

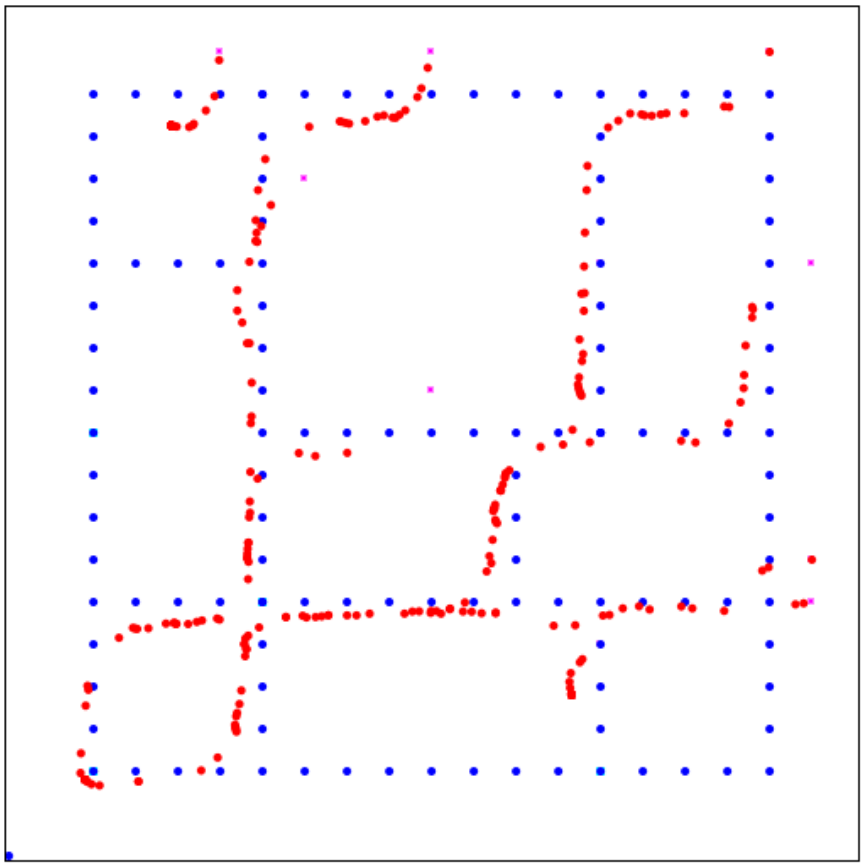
４．２．２　歩行者エージェントのルール

```
Agt_Init{

}

Agt_Step{
Dim distance As Double
Dim min As Double
Dim one As Agt
Dim threat As Agt
Dim set    As Agtset
Dim  目標 X As Double
Dim  目標 Y As integer

MakeOneAgtSetAroundOwn(set    ,5,    Universe.map.wp
False)
//半径○セルの WP を視界(set)に収める

//＝＝＝＝＝基本行動ルールテキスト＝＝＝＝＝
my.番号  = 0
my.目標到達  = 0
目標 X = 0
目標 Y = 0
my.目標  X  =  Universe.map.目的地.X(Universe.目的地
設定(0,0))
my.目標  Y  =  Universe.map.目的地.Y(Universe.目的地
設定(0,0))

    if my.番号  == 1 then
        my. 目 標  X  =  Universe.map. 目 的
地.X(Universe.目的地設定(1,0))
```

```
            my. 目 標  Y  =  Universe.map. 目 的
地.Y(Universe.目的地設定(1,0))

             elseif my.番号== 2 then
            my. 目 標  X  =  Universe.map. 目 的
地.X(Universe.目的地設定(2,0))
            my. 目 標  Y  =  Universe.map. 目 的
地.Y(Universe.目的地設定(2,0))
        End if
If CountAgtset(set) == 0 Then
            my.Direction  =  GetDirection(my.X,  my.Y,  目
標 X,  目標 Y, Universe.map)
            Forward(0.5)
Else
            min = 30
            For each one In set
                        distance  =  MeasureDistance(My.X,
My.Y, one.x, one.y, Universe.map)
                        If distance < min Then
                                    threat = one
                                    min = distance
                                    my.Direction
=
GetDirection(my.X, my.Y,    one.x, one.y, Universe.map)
                                    Forward(0.5)
                        End if
                        if distance < 0.1 Then
                                    one = threat
                                    min = 30
                                    my.Direction
=
GetDirection(my.X,  my.Y,  目  標  X,  目  標  Y,
Universe.map)
                                    Forward(0.5)
                         End if
                        if my.Direction > 30 Then
                                    one = threat
                                    min = 30
                                    my.Direction
=
GetDirection(my.X,  my.Y,  目  標  X,  目  標  Y,
Universe.map)
                                    Forward(0.5)
                        End if
                        if my.Direction < 30 Then
                                    one = threat
                                    min = 30
                                    my.Direction
=
GetDirection(my.X,  my.Y,  目  標  X,  目  標  Y,
Universe.map)
                                    Forward(0.5)
                        End if

            Next one

end if
//ここまで人の基本動作,以下は目標更新機能

}
Function  目標方向() As Double{

            Dim rtn As Double


       if my.X >=my.目標 X- 2 and my.X <= my.目標 X +
2
            and my.Y >= my.目標 Y - 2 and my.Y<= my.目
標 Y +2 then
            my.目標到達  = my.目標到達  + 1
      end if

            my.目標 X  =  Universe.map.目的地.X(目標設定
())
            my.目標 Y  =  Universe.map.目的地.Y(目標設定
())

            rtn = GetDirection(my.X, my.Y, my.目標 X, my.目
標 Y, Universe.map)


            Return(rtn )
```

```
                                        i = 1
}                                     elseif my.番号  == 2 then
                                        i = 2
Function  目標設定() as integer {     end if

  Dim i as integer                    j = my.目標到達
  Dim j as integer                    WP = Universe.目的地設定(i,j)
  Dim WP as integer

                                    return (WP)
      if my.番号  == 0 then
          i = 0                     }
       elseif my.番号  == 1 then
```

４．２．３　シミュレーション



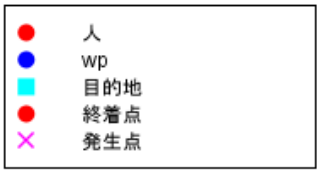


図４－２　シミュレーション起動後

このシミュレーションは 7 つの歩行者の行動ルールを書き込んだ.

①半径 5 セル以内の WP を視野に入れる

②半径 5 セル以内の視野の中に WP が 0 の場合歩行者エージェントは目的地の方向に進む

③視野の範囲が 30 度の範囲である.そして視野に入った WP と自らの距離を測る.

④もし視野 30 度の範囲に WP があった場合,その WP の方向に進む.

⑤もし WP と自らの距離が 0.1 未満だった場合,目的地の方向へ進む.

⑥もし視野 30 度以上の範囲に WP がある場合,目的地方向に進む.
⑦もし視野 30 度以内の範囲に WP がある場合,目的地方向に向かう.

このルールをまとめると,ルールから歩行者エージェントは発生点で生まれ生まれたところから視野 5 のセルの範囲内・視野 30 度以内を基準にし,視野に WP が入った場合そのポイントまでの距離を測り移動する.視野に複数の WP が入った場合は自分に一番近い目的地方向の WP に進む.この場合,近づきすぎた WP は目標で無くなり,新たに視野に入った WP に向かって移動を繰り返す.視野に何もない場合は,目的地方向に進む.

第5章　まとめ

５．１　まとめ

今回行った基礎研究により最終目的であるマルチエージェントを用いた情報の利活用による帰宅困難者支援効果の検証に必要な歩行ルールを少しだが理解することが出来た.しかし,最終目的に到達するにはまだまだシミュレーションの理解を深めていかなければならないと考える.

第６章　シミュレーション練習

６．１　条件

1)場所

「東京 23 区～横浜市・川崎市」

横浜市・川崎市は東京に隣接する大都市であり,東京 23 区に移動している人も多く地震などの災害が発生した際に多くの帰宅困難者が出ることが予測される.

2)人口

・横浜市～東京 23 区に移動する人数＝446497

・川崎市～東京 23 区に移動する人数＝256407

（東京都ではなく東京 23 区で 4 シミュレーションを行う理由として,東京 23 区以外の人口の移動は横浜市＝7%,川崎市＝11%と少ないため今回は含まないものとする）

3)幹線道路

この図のオレンジ色とグリーン色を帰宅困難者が通るルートとし,幹線道路とする.

図６－１　シミュレーションマップ

4)人口の分布

	人数(人)		人数(人)
千代田区	61,569	渋谷区	23,848
中央区	41,995	中野区	1,458
港区	82,240	杉並区	1,798
新宿区	23,783	豊島区	4,715
文京区	6,849	北区	2,337
台東区	7,276	荒川区	955
墨田区	4,066	板橋区	921
江東区	10,921	練馬区	405
品川区	38,659	足立区	699
目黒区	10,052	葛飾区	224
大田区	32,767	江戸川区	1,112
世田谷区	9,020	東京23区計	367,669

表６－１　横浜市民分布

	人数(人)		人数(人)
千代田区	26,751	渋谷区	17,556
中央区	15,788	中野区	1,620
港区	39,956	杉並区	1,504
新宿区	18,084	豊島区	4,004
文京区	3,820	北区	673
台東区	2,960	荒川区	614
墨田区	1,424	板橋区	894
江東区	5,570	練馬区	245
品川区	16,644	足立区	1,618
目黒区	8,044	葛飾区	231
大田区	11,290	江戸川区	252
世田谷区	10,892	東京23区計	190,434

表６－２　川崎市民分布

この 2 つの市をの合計の数値を各区に分布させシミュレーションを行う.

5)地震レベル

今回シミュレーションを行うにあたって,地震レベルを 3 段階にに設定する.地震レベルと言ってもシミュレーションの際に歩行者エージェントが通るルートの数を調整するものである.エージェントが利用する幹線道路を地震レベル１～地震レベル３の間をコントロールパネルによって簡単に設定できるようにする.

6)橋

①東名高速道路　　⑥ガス橋
②厚木街道　　⑦第二京浜
③玉川通り　　⑧第一京浜
④第三京浜道路　　⑨産業道路
⑤中原街道　　⑩首都高速神奈川 1 号横羽線
⑪首都高速湾岸線

この橋に続く道路をエージェントが通る主な幹線道路とし,地震レベルにより橋を壊したり主な幹線道路を通れなくする.

図６－３　23 区と神奈川をつなぐ橋

7)歩行者エージェントの移動（ASPF）

歩行者の移動パターンをエージェントに反映するにあたって ASPF の利用を考えている.

ASPF とは

ASPF(Agent Simulation of Pedestrian Flow)プロジェクトと称して,名古屋工業大学兼田研究室の卒業研究や修士研究のメンバーが取り組んできた歩行者モデル開発と一連のバージョンアップの成果である [6].図－2 は ASPF の行動パターンの一例である.

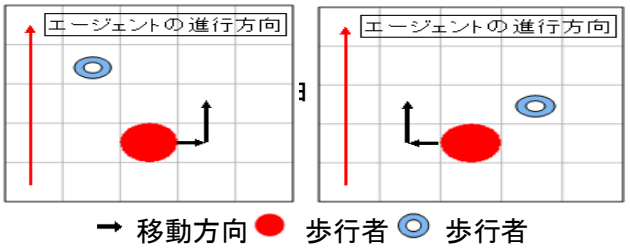
2)ASPF の歩行行動ルール

①基本行動ルール：低密度歩行時における主にエージェントの直進行動を規定する.

②対他減速ルール:低密度歩行時におもに前後のエージェントと間隔を保つため,他のエージェントに近づくにつれ減速する.

③対他回避ルール：低密度歩行時におもに左右のエージェントと間隔を保つため,他者を回避する.

④高密度ルール：高密度歩行時に人々は左右の間隔を減らすよりも前後の間隔を減らす動作をする.



図―２ASPF 歩行行動の例

2,3=×の場合

1.2.4.6=×情報ありの場合

このシミュレーションは「L 字通路」を参考にして作ったシミュレーションであり,WP を歩行者エージェントがたどっていきゴール地点で消える（避難終了）というルールで行った.理想としては歩行者エージェントに判断力を持たせ,各エージェントの意志で避難行動が出来れば人間らしい行動が出来たと考える.

第７章　シミュレーション勉強

７．１　シミュレーション例

Agt_Init・・・括弧の間の部分には,最初だけに実行するルールを書き込む

Agt_Step・・・括弧の間の部分には,毎ステップ実行させるルールを書き込む

７．１．１　Ｌ字通路

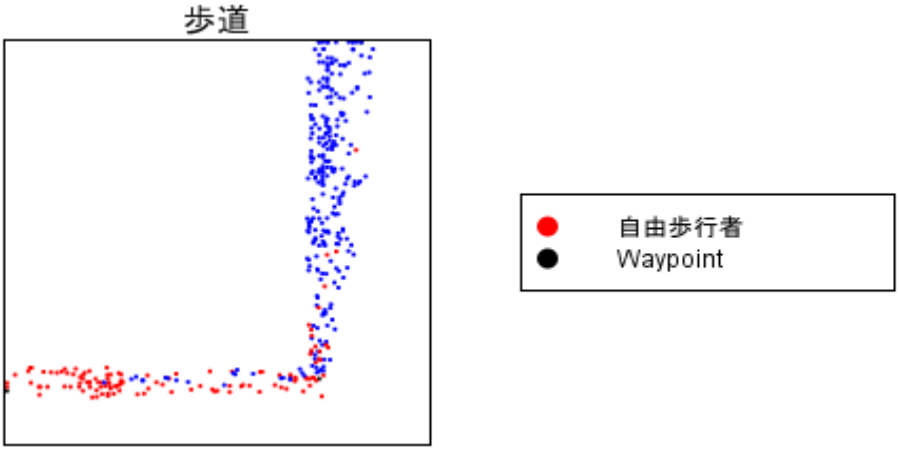


図７－１　Ｌ字通路シミュレーション

歩行者エージェントのルール

Agt_Init{

my.固有巡航速度 = NormInv(Rnd(),1.675,0.325)

my.WP 到達 = 0

my.ステップ数 = 0

if my.色 == color_blue then

my.WPX = Universe.歩道.Waypoint.X(Universe.WP 設定(0,0))

my.WPY = Universe.歩道.Waypoint.Y(Universe.WP 設定(0,0))

elseif my.色 == color_red then

my.WPX = Universe.歩道.Waypoint.X(Universe.WP 設定(1,0))

my.WPY = Universe.歩道.Waypoint.Y(Universe.WP 設定(1,0))

end if

my.Direction = GetDirection(my.X, my.Y, my.WPX, my.WPY, Universe.歩道)

}

Agt_Step {

Dim 歩行者視野 A as AgtSet

Dim 歩行者視野 B as AgtSet

Dim 歩行者視野 C as AgtSet

Dim 歩行者視野 D as AgtSet

Dim 自分 as AgtSet

Dim 歩行者視野 X as AgtSet

Dim 誰か as object

Dim 総歩行者 as AgtSet

Dim 対象歩行者 as AgtSet

Dim 壁 as AgtSet

Dim ストッパー as AgtSet

Dim 領域１ as AgtSet

Dim 領域２ as AgtSet

```
Dim 領域３　as AgtSet

Dim 領域４　as AgtSet
Dim 領域５　as AgtSet
Dim 領域６　as AgtSet

Dim deg as integer
Dim　相対 X 差 as double
Dim　相対 Y 差 as double
Dim　絶対直線距離 as double
Dim　アークタンジェント as double
Dim　RADmydirection as double
Dim　相対シータ as double
Dim　シータ相対 X 差 as integer
Dim　シータ相対 Y 差 as integer

Dim シータ絶対 X 差　as integer
Dim シータ絶対 Y 差 as integer
Dim 目標距離 as double
Dim　歩行者密度 as integer
Dim　歩行者密度 1 as integer

Dim　対向合計 as integer
Dim　確率　as double
確率 = Rnd()

Dim　対歩行者（m１．０）　as boolean
Dim　対歩行者（m１．１）　as boolean
Dim　対歩行者（m１．２）　as boolean
Dim　対歩行者（０．m１）　as boolean
Dim　対歩行者（０．１）　as boolean
Dim　対歩行者（０．２）　as boolean
Dim　対歩行者（０．３）　as boolean
Dim　対歩行者（０．４）　as boolean
Dim　対歩行者（１．m１）　as boolean
Dim　対歩行者（１．０）　as boolean
Dim　対歩行者（１．１）　as boolean
Dim　対歩行者（１．２）　as boolean
Dim　対歩行者（１．３）　as boolean
Dim　対歩行者（１．４）　as boolean
Dim　対歩行者（２．０）　as boolean
Dim　対歩行者（２．１）　as boolean
Dim　対歩行者（２．２）　as boolean
Dim　対歩行者（２．３）　as boolean
Dim　対歩行者（３．０）　as boolean
Dim　対歩行者（３．１）　as boolean
Dim　対歩行者（３．２）　as boolean

Dim　壁（m１．０）　as boolean
Dim　壁（０．m１）　as boolean
Dim　壁（０．１）　as boolean
Dim　壁（１．０）　as boolean

Dim 絶対対歩行者（m１．０）　as boolean
Dim 絶対対歩行者（１．０）　as boolean
Dim 絶対対歩行者（０．１）　as boolean
Dim 絶対対歩行者（０．m１）　as boolean

絶対対歩行者（m１．０）= false
絶対対歩行者（１．０）　= false
絶対対歩行者（０．１）　= false
絶対対歩行者（０．m１）　= false

Dim　対向（０．３）　as integer
Dim　対向（０．４）　as integer
Dim　対向（１．m１）　as integer
Dim　対向（１．０）　as integer
Dim　対向（１．１）　as integer
Dim　対向（１．２）　as integer
Dim　対向（１．３）　as integer
Dim　対向（１．４）　as integer
Dim　対向（２．０）　as integer
Dim　対向（２．１）　as integer
Dim　対向（２．２）　as integer
Dim　対向（２．３）　as integer
Dim　対向（３．０）　as integer
```

```
Dim 対向（３．１） as integer
Dim 対向（３．２） as integer

対歩行者（m１．０） = false
対歩行者（m１．１） = false
対歩行者（m１．２） = false
対歩行者（０．m１） = false
対歩行者（０．１） = false
対歩行者（０．２） = false
対歩行者（０．３） = false
対歩行者（０．４） = false
対歩行者（１．m１） = false
対歩行者（１．０） = false
対歩行者（１．１） = false
対歩行者（１．２） = false
対歩行者（１．３） = false
対歩行者（１．４） = false
対歩行者（２．０） = false
対歩行者（２．１） = false
対歩行者（２．２） = false
対歩行者（２．３） = false
対歩行者（３．０） = false
対歩行者（３．１） = false
対歩行者（３．２） = false


壁（m１．０） = false
壁（０．m１） = false
壁（０．１） = false
壁（１．０） = false


対向（０．３） = 0
対向（０．４） = 0
対向（１．m１） = 0
対向（１．０） = 0
対向（１．１） = 0
対向（１．２） = 0
対向（１．３） = 0
対向（１．４） = 0
対向（２．０） = 0
対向（２．１） = 0
対向（２．２） = 0
対向（２．３） = 0
対向（３．０） = 0
対向（３．１） = 0
対向（３．２） = 0

'*****************************************
*****************************************
**********

    my.ステップ数 = my.ステップ数 + 1
     if  ( my.ステップ数 mod 10) == 0 Then
          my.Direction = 目標方向()
    end if

'*****************************************
*****************************************
**********

  目標距離 = MeasureDistance(my.X, my.Y, my.WPX,
my.WPY, Universe.歩道)

  if 目標距離 < 5 then
      if  (GetCountStep() mod 1) == 0 Then
          my.Direction = 目標方向()
      end if
  end if

'*****************************************
*****************************************
**********

MakeOneAgtSetAroundPositionCell( 歩 行 者 視 野 A,
GetRideSpace(SpecifyAgtType(My)),my.X, my.Y, 0,
4,Universe.歩道.壁)
MakeOneAgtSetAroundPositionCell( 歩 行 者 視 野 B,
```

```
GetRideSpace(SpecifyAgtType(My)),my.X,  my.Y,  0,
4,Universe.歩道.自由歩行者)
MakeOneAgtSetAroundPositionCell( 歩 行 者 視 野  C,
GetRideSpace(SpecifyAgtType(My)),my.X,  my.Y,  0,
4,Universe.歩道.自由歩行者)

'****************************** 絶対座標の定義
**********************************************
**********

 for each 誰か in 歩行者視野 A

        シータ絶対 X 差 = 誰か.X - my.X
        シータ絶対 Y 差 = 誰か.Y - my.Y

    if シータ絶対 X 差 == -1 and シータ絶対 Y
差 == 0 then
            壁（m１．０） = true

    elseif シータ絶対 X 差 == 0 and シータ絶対
Y 差 == -1 then
            壁（０．m１） = true

    elseif シータ絶対 X 差 == 0 and シータ絶対
Y 差 == 1 then
            壁（０．１） = true

    elseif シータ絶対 X 差 == 1 and シータ絶対
Y 差 == 0 then
            壁（１．０） = true

    end if

next 誰か

for each 誰か in 歩行者視野 B

    シータ絶対 X 差 = 誰か.X - my.X
    シータ絶対 Y 差 = 誰か.Y - my.Y

    if シータ絶対 X 差 == -1 and シータ絶対 Y
差 == 0 then
            絶対対歩行者（m１．０） = true
      elseif シータ絶対 X 差 == 1 and シータ絶
対 Y 差 == 0 then
            絶対対歩行者（１．０） = true
      elseif シータ絶対 X 差 == 0 and シータ絶
対 Y 差 == -1 then
            絶対対歩行者（０．m１） = true
      elseif シータ絶対 X 差 == 0 and シータ絶
対 Y 差 == 1 then
            絶対対歩行者（０．１） = true
    end if
next 誰か

'****************************** θ 相対座標の定義
**********************************************
**********

RADmydirection = FlatRad(my.Direction)

for each 誰か in 歩行者視野 C

    絶 対 直 線 距 離 = MeasureDistance(my.X,
my.Y, 誰か.X, 誰か.Y, Universe.歩道)

    相対 X 差 = 誰か.X - my.X
    相対 Y 差 = 誰か.Y - my.Y
    if 相対 X 差 == 0 then
        相 対 X 差 =
0.00000000000000000000000000000000000000001
    end if

    アークタンジェント = Atn(相対 Y 差 / 相
対 X 差)
```

```
if my.direction >= 0 and my.direction < 180 then

        if my.X >= 誰か.X then
            相対シータ = PI() - RADmydirection - アークタンジェント

        elseif my.X < 誰か.X then
            相対シータ = -RADmydirection - アークタンジェント

        end if

elseif my.direction >= 180 and my.direction < 360 then

        if my.X >= 誰か.X then
            相対シータ = PI() - RADmydirection - アークタンジェント

        elseif my.X < 誰か.X then
            相対シータ = (2 * PI()) - RADmydirection - アークタンジェント

        end if
end if
'-----------------------------------------------------
シータ相対 X 差 = 絶対直線距離 * cos(相対シータ)
シータ相対 Y 差 = 絶対直線距離 * sin(相対シータ)

if シータ相対 X 差 == -1 and シータ相対 Y 差 == 0 then
        対歩行者（m１．０） = true

elseif シータ相対 X 差 == -1 and シータ相対 Y 差 == 1 then
        対歩行者（m１．１） = true

elseif シータ相対 X 差 == -1 and シータ相対 Y 差 == 2 then
        対歩行者（m１．２） = true

elseif シータ相対 X 差 == 0 and シータ相対 Y 差 == -1 then
        対歩行者（０．m１） = true

elseif シータ相対 X 差 == 0 and シータ相対 Y 差 == 1 then
        対歩行者（０．１） = true

elseif シータ相対 X 差 == 0 and シータ相対 Y 差 == 2 then
        対歩行者（０．２） = true

elseif シータ相対 X 差 == 0 and シータ相対 Y 差 == 3 then
        対歩行者（０．３） = true

        if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then
            対向（０．３）= 1
        elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then
            対向（０．３）= -1
        elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then
            対向（０．３）= -1
        end if

elseif シータ相対 X 差 == 0 and シータ相対 Y 差 == 4 then
        対歩行者（０．４） = true
```

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then

対向（０．４）= 1

elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then

対向（０．４）= -1

elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then

対向（０．４）= -1

end if

elseif シータ相対X差 == 1 and シータ相対Y差 == -1 then

対歩行者（１．m１） = true

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then

対向（１．m１）= 1

elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then

対向（１．m１）= -1

elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then

対向（１．m１）= -1

end if

elseif シータ相対X差 == 1 and シータ相対Y差 == 0 then

対歩行者（１．０） = true

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then

対向（１．０）= 1

elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then

対向（１．０）= -1

elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then

対向（１．０）= -1

end if

elseif シータ相対X差 == 1 and シータ相対Y差 == 1 then

対歩行者（１．１） = true

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then

対向（１．１）= 1

elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then

対向（１．１）= -1

elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then

対向（１．１）= -1

end if

elseif シータ相対X差 == 1 and シータ相対Y差 == 2 then

対歩行者（１．２） = true

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then

対向（１．２）= 1

elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰

か.DIRECTION > (my.DIRECTION + 170) then
対向（１．２）= -1
elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then
対向（１．２）= -1
end if

elseif シータ相対 X 差 == 1 and シータ相対 Y 差 == 3 then
対歩行者（１．３） = true

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then
対向（１．３）= 1
elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then
対向（１．３）= -1
elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then
対向（１．３）= -1
end if

elseif シータ相対 X 差 == 1 and シータ相対 Y 差 == 4 then
対歩行者（１．４） = true

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then
対向（１．４）= 1
elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then
対向（１．４）= -1
elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then
対向（１．４）= -1
end if

elseif シータ相対 X 差 == 2 and シータ相対 Y 差 == 0 then
対歩行者（２．０） = true

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then
対向（２．０）= 1
elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then
対向（２．０）= -1
elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then
対向（２．０）= -1
end if

elseif シータ相対 X 差 == 2 and シータ相対 Y 差 == 1 then
対歩行者（２．１） = true

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then
対向（２．１）= 1
elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then
対向（２．１）= -1
elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then
対向（２．１）= -1

end if

elseif シータ相対X差 == 2 and シータ相対Y差 == 2 then

対歩行者（２．２） = true

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then

対向（２．２）= 1

elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then

対向（２．２）= -1

elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then

対向（２．２）= -1

end if

elseif シータ相対X差 == 2 and シータ相対Y差 == 3 then

対歩行者（２．３） = true

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then

対向（２．３）= 1

elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then

対向（２．３）= -1

elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then

対向（２．３）= -1

end if

elseif シータ相対X差 == 3 and シータ相対Y差 == 0 then

対歩行者（３．０） = true

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then

対向（３．０）= 1

elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then

対向（３．０）= -1

elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then

対向（３．０）= -1

end if

elseif シータ相対X差 == 3 and シータ相対Y差 == 1 then

対歩行者（３．１） = true

if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then

対向（３．１）= 1

elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then

対向（３．１）= -1

elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then

対向（３．１）= -1

end if

elseif シータ相対X差 == 3 and シータ相対Y差 == 2 then

対歩行者（３．２） = true

```
                if 誰か.DIRECTION < (my.DIRECTION + 10) and 誰か.DIRECTION > (my.DIRECTION - 10) then
                        対向（３．２）= 1
                elseif my.DIRECTION < 180 and 誰か.DIRECTION < (my.DIRECTION + 190) and 誰か.DIRECTION > (my.DIRECTION + 170) then
                        対向（３．２）= -1
                elseif my.DIRECTION > 180 and 誰か.DIRECTION < (my.DIRECTION - 170) and 誰か.DIRECTION > (my.DIRECTION - 190) then
                        対向（３．２）= -1
                end if
        end if

next 誰か

'***************************** θ 相対座標の定義終わり ***********************************************

MakeOneAgtSetAroundPositionCell( 対象歩行者, GetRideSpace(SpecifyAgtType(My)),
my.X,my.Y,0,3,Universe.歩道.自由歩行者)
歩行者密度 = CountAgtSet(対象歩行者)



対向合計 = 対向（０．３）+対向（０．４）+対向（１．m１）+対向（１．０）+対向（１．１）+対向（１．２）+対向（１．３）+対向（１．４）+対向（２．０）+対向（２．１）+対向（２．２）+対向（２．３）+対向（３．０）+対向（３．１）+対向（３．２）

'*************************************************************************************

'***************************** 自由歩行者行動ルール群 ***********************************************

'*************************************************************************************

if 壁（m１．０） == true or 壁（０．m１） == true or 壁（０．１） == true or 壁（１．０） == true then
    goto kabe
end if

'*************************************************************************************

if 歩行者密度 > 16 then
        goto koumitu
end if

    '*************** ルール１基本行動ルール ***********************************************

'#################rule1#################################################

        if 対歩行者（０．１）== true and 対歩行者（１．０）== true and 対歩行者（m１．０）== true then
                Smove(0,0)
                goto owari
        end if

'#################rule2#################################################
```

```
        if 対歩行者（０．１） == true and 対歩行者（１．０）== false and 対歩行者（m１．０）== true then
                Smove(1,0)
                goto owari
        end if

'##################rule3#########################################################
##

        if 対歩行者（０．１） == true and 対歩行者（１．０）== true and 対歩行者（m１．０）== false then
                Smove(-1,0)
                goto owari
        end if

'##################rule4#########################################################
##

        if 対歩行者（０．１） == true and 対歩行者（１．０）== false and 対歩行者（m１．０）== false then
                if my.Direction >= 目標方向() then
                        Smove(1,0)
                        goto owari
                else
                        Smove(-1,0)
                        goto owari
                end if
        end if

    '##################rule5
#################################################################
#########################

        if 対歩行者（０．１）== false and 対歩行者（０．２）== false and
                対歩行者（０．３）== false and 対歩行者（０．４）== false then
                Smove(0,1)
                goto owari
        end if

'##################rule6#########################################################
##

        if 対歩行者（０．１）== false and 対歩行者（０．２）== false and
                対歩行者（０．３）== false and 対歩行者（０．４）== true then

                if 対歩行者（１．０）== false and 対歩行者（２．０）== false and
                        対歩行者（１．１）== false and 対歩行者（２．１）== false and
                        対歩行者（１．２）== false and 対歩行者（２．２）== false then

                        if my.Direction > 目標方向()  then
                                Smove(1,1)
                                goto owari
                        elseif my.Direction <= 目標方向() then
                                Smove(-1,1)
                                goto owari
                        else
                                Smove(0,1)
                                goto owari
                        end if
                else
                        Smove(0,1)
                        goto owari
                end if
        end if
```

```
    '*************** ルール 2　他個人減速ルール
*****************************************************
*******
'##################rule7#########################
#################################################
##


          if 対歩行者（０．２）== true and 対歩行者
（０．１）== false then
                    if 対歩行者(１．０)== false and 対
歩行者（m１．０）== false then
                              if my.Direction <= 目標方
向() then
                                        Smove(-1,0)
                                   if 対歩行者（m１．
１）== false then
                              Smove(0,1)
                                        goto owari
                               else
                                        Smove(0,0)
                                        goto owari
                               end if
                              else
                                        Smove(1,0)
                                   if 対歩行者（１．１）
== false then
                                        Smove(0,1)
                                        goto owari
                               else
                                        Smove(0,0)
                                        goto owari
                               end if
                              end if
                    end if

'##################rule8#########################
#################################################
##
                    if 対歩行者(１．０)== false and 対
歩行者（m１．０）== true then
                              Smove(1,0)
                         if 対歩行者（１．１）== false
then
                              Smove(0,1)
                              goto owari
                    else
                              Smove(0,0)
                              goto owari
                    end if
               end if

'##################rule9#########################
#################################################
##
                    if 対歩行者(１．０)== true and 対
歩行者（m１．０）== false then
                              Smove(-1,0)
                         if　対歩行者(m１．１)== false
then
                              Smove(0,1)
                              goto owari
                    else
                              Smove(0,0)
                              goto owari
                    end if
               end if

'##################rule10########################
#################################################
##
                    if 対歩行者(１．０)== true and 対
歩行者（m１．０）== true then
                              Smove(0,0.5)
```

```
                goto owari
            end if
        end if

'#################rule11#########################################
##

        if 対歩行者（m１．２）== true then
            if 対歩行者（１．０）== false then
                Smove(1,0)
                if 対歩行者（１．１）== false
then
                    Smove(0,1)
                    goto owari
                else
                    Smove(0,0)
                    goto owari
                end if
            end if

'#################rule12#########################################
##

            if 対歩行者（１．０）== true then
                Smove(0,0.5)
                goto owari
            end if
        end if


'#################rule13#########################################
##

        if 対歩行者（１．２）== true then
            if 対歩行者（m１．０）== false then
                Smove(-1,0)
                if 対歩行者（m１．１）== false
then
                    Smove(0,1)
                    goto owari
                else
                    Smove(0,0)
                    goto owari
                end if
            end if
'#################rule14#########################################
##

            if 対歩行者（m１．０）== true then
                Smove(0,0.5)
                goto owari
            end if
        end if

    '*************** ルール 3　他個人回避ルール
*********************************************
*******

'#################rule15#########################################
##

        if 対歩行者（１．１）== true and 対歩行者
（m１．０）== false then
            Smove(-1,0)
            goto owari
        end if

'#################rule16#########################################
##

        if 対歩行者（m１．１）==true and 対歩行者
（１．０）== false then
```

```
                Smove(1,0)
                goto owari
        end if

'#################rule17#########################################
##

        if 対歩行者（１．０）== true and 対歩行者（m１．０）== false then
                Smove(-1,0)
            if 対歩行者（m１．１）== false then
                    Smove(0,1)
                    goto owari
            else
                    Smove(0,0)
                    goto owari
            end if
        end if

'#################rule18#########################################
##

        if 対歩行者（m１．０）== true and 対歩行者（１．０）== false then
                Smove(1,0)
            if 対歩行者（１．１）== false then
                    Smove(0,1)
                    goto owari
            else
                    Smove(0,0)
                    goto owari
            end if
        end if

   '*************** ルール 4　高密度歩行の場合のルール(左右の間隔よりも前後の間隔を詰める) *********

koumitu:

Dim 高密度対象歩行者　as Agtset
Dim 高密度歩行者密度 as integer

MakeOneAgtSetAroundPositionCell(高密度対象歩行者, GetRideSpace(SpecifyAgtType(My)), round(my.X-sin(RADmydirection)),round(my.Y-cos(RADmydirection)),0,0,Universe.歩道.自由歩行者)
高密度歩行者密度 = CountAgtSet(高密度対象歩行者)

'#################rule19#########################################
##

        if 高密度歩行者密度 <=1　then

                Smove(0,0.5)
                goto owari
        else
                MoveToSpaceAgtSetCell(Universe.歩道, my.X, my.Y, 0, 1, 高密度対象歩行者)
                goto owari
        end if

'#################rule20#########################################
##

        if 対歩行者（０．２）== true and 対歩行者（０．１）== false then
                Smove(0,1)
                goto owari
        end if

'#################rule21#########################################
##
```

```
        if 対歩行者（０．３）== true and 対歩行者（０．２）== false and 対歩行者（０．１）== false then
                Smove(0,1)
                goto owari
        end if

   '*************** ルール 5 流れ読みルール「対向　　流　　」
*********************************************
*

'#################rule22#########################################
##

        if  対歩行者（０．m１）== false  and 対向合計 > 3 then
                if 確率 < 0.8    then
                        Smove(1,0)
                        goto owari
                end if
        elseif 対歩行者（０．１）== false  and 対向合計 < -3 then
                if 確率 < 0.9    then
                        Smove(-1,0)
                        goto owari
                end if
        end if

kabe:
   '*************** ルール 6　壁回避ルール
*********************************************
*

'#################rule23
#################################################
#########################

if 壁（０．１）== true and 壁（１．０）== false and 壁（０．m１）== false and 壁（m１．０）== false then my.Direction = 目標方向()
        if my.Direction >= 0 and my.Direction <= 90 and 絶対対歩行者（１．０）== false then
                ForwardX(my.固有巡航速度)
                goto owari
        elseif my.Direction> 90 and my.Direction <= 180 and 絶対対歩行者（m１．０）== false then
                ForwardX(-my.固有巡航速度)
                goto owari
         elseif 対歩行者（０．１）== false then
                Smove(0,1)
                goto owari
                else
                ForwardY(-1)
                goto owari
    end if
end if

'#################rule24#########################################
#############################################
###

if 壁（０．１）== false and 壁（１．０）== true and 壁（０．m１）== false and 壁（m１．０）== false then my.Direction = 目標方向()
    if my.Direction >= 0 and my.Direction <= 90 and 絶対対歩行者（１．０）== false then
                ForwardY(my.固有巡航速度)
                goto owari
          elseif my.Direction>=270 and my.Direction < 360 and 絶対対歩行者（０．m１） == false then
                        ForwardY(-my.固有巡航速度)
                        goto owari
                elseif 対歩行者（０．１）== false then
                        Smove(0,1)
```

```
                goto owari
    end if
end if

'##################rule25#########################################################
###

if 壁（０．１）== false and 壁（１．０）== false and 壁（０．m１）== true and 壁（m１．０）== false then
my.Direction = 目標方向()
        if my.Direction >= 180 and my.Direction <= 270 and 絶対対歩行者（m１．０）== false then
                ForwardX(-my.固有巡航速度)
                goto owari
            elseif my.Direction>270 and my.Direction <= 360 and 絶対対歩行者（１．０）== false then
                ForwardX(my.固有巡航速度)
                goto owari
            elseif 対歩行者（０．１）== false then
                Smove(0,1)
                goto owari
    end if
end if

'##################rule26#########################################################
###

if 壁（０．１）== false and 壁（１．０）== false and 壁（０．m１）== false and 壁（m１．０）== true then
my.Direction = 目標方向()
            if my.Direction >= 90 and my.Direction < 180 and 絶対対歩行者（０．１）== false then
                ForwardY(my.固有巡航速度)
                goto owari
            elseif my.Direction>=180 and my.Direction <= 270 and 絶対対歩行者（１．０）== false then
                ForwardY(-my.固有巡航速度)
                goto owari
            elseif 対歩行者（０．１）== false then
                Smove(0,1)
                goto owari
    end if
end if

'##################rule27#########################################################
###

if 壁（０．１）== true and 壁（１．０）== true and 壁（０．m１）== false and 壁（m１．０）== false then
my.Direction = 目標方向()
  if my.Direction > 45 and my.Direction <= 180 and 絶対対歩行者（m１．０）== false and 絶対対歩行者（m１．０）== false  then
    ForwardX(-1)
    goto owari
   elseif my.Direction> -90 and my.Direction <= 45 and 絶対対歩行者（０．m１）== false  and 絶対対歩行者（０．m１）== false then
   ForwardY(-1)
   goto owari
   elseif 対歩行者（０．１）== false then
   Smove(0,1)
   goto owari
   end if
end if

'##################rule28#########################################################
###

if 壁（０．１）== true and 壁（１．０）== false and 壁（０．m１）== true and 壁（m１．０）== false then
```

```
my.Direction = 目標方向()
  if my.Direction > -45 and my.Direction <= 90 and 絶対対歩行者（１．０）== false and 絶対対歩行者（０．１）== false  then
     ForwardY(1)
     goto owari
   elseif my.Direction>-45 and my.Direction <= 180 and 絶対対歩行者（m１．０）== false and 絶対対歩行者（m１．０）== false then
   ForwardX(-1)
   goto owari
   elseif 対歩行者（０．１）== false then
   Smove(0,1)
   goto owari
   end if
end if

'#################rule29#########################################################
###

if 壁（０．１）== true and 壁（１．０）== false and 壁（０．m１）== false and 壁（m１．０）== true then
my.Direction = 目標方向()
  if my.Direction > 0 and my.Direction <= 135 and 絶対対歩行者（１．０）== false then
     ForwardX(1)
     goto owari
   elseif my.Direction>135 and my.Direction <= 270 and 絶対対歩行者（０．m１）== false then
   ForwardY(-1)
   goto owari
   elseif 対歩行者（０．１）== false then
   Smove(0,1)
   goto owari
   end if
end if

'#################rule30#########################################################
###

if 壁（０．１）== false and 壁（１．０）== true and 壁（０．m１）== true and 壁（m１．０）== false then
my.Direction = 目標方向()
  if my.Direction > 180 and my.Direction <= 315and 絶対対歩行者（m１．０）== false  then
     ForwardX(-1)
     goto owari
   elseif my.Direction>-45 and my.Direction <= 90 and 絶対対歩行者（０．１）== false then
   ForwardY(1)
   goto owari
   elseif 対歩行者（０．１）== false then
   Smove(0,1)
   goto owari
   end if
end if

'#################rule31#########################################################
###

if 壁（０．１）== false and 壁（１．０）== true and 壁（０．m１）== false and 壁（m１．０）== true then
my.Direction = 目標方向()
  if my.Direction > 0 and my.Direction <= 180 and 絶対対歩行者（０．１）== false then
     ForwardY(1)
     goto owari
   elseif my.Direction>180 and my.Direction <= 360 and 絶対対歩行者（０．m１）== false then
   ForwardY(-1)
   goto owari
   elseif 対歩行者（０．１）== false then
   Smove(0,1)
   goto owari
   end if
```

```
end if

'#################rule32#########################################################
###

if 壁（０．１）== false and 壁（１．０）== false and 壁（０．ｍ１）== true and 壁（ｍ１．０）== true then
my.Direction = 目標方向()
  if my.Direction > 90 and my.Direction <= 225 and 絶対対歩行者（０．１）== false then
     ForwardY(1)
     goto owari
   elseif my.Direction>-135 and my.Direction <= 0 and 絶対対歩行者（１．０）== false then
   ForwardX(1)
   goto owari
   elseif 対歩行者（０．１）== false then
   Smove(0,1)
   goto owari
   end if
end if

'#################rule33#########################################################
###

if 壁（０．１）== true and 壁（１．０）== true and 壁（０．ｍ１）== true and 壁（ｍ１．０）== false then
   my.Direction = 目標方向()
      if 絶対対歩行者（ｍ１．０）== false then
         ForwardX(-1)
      goto owari
     end if
end if

'#################rule34#########################################################
###

if 壁（０．１）== true and 壁（１．０）== false and 壁（０．ｍ１）== true and 壁（ｍ１．０）== true then
   my.Direction = 目標方向()
   if 絶対対歩行者（１．０）== false then
      ForwardX(1)
      goto owari
    end if
end if

'#################rule35#########################################################
###

if 壁（０．１）== false and 壁（１．０）== true and 壁（０．ｍ１）== true and 壁（ｍ１．０）== true then
my.Direction = 目標方向()
     if 絶対対歩行者（０．１）== false then
        ForwardY(1)
        goto owari
     end if
end if

'#################rule36#########################################################
###

if 壁（０．１）== true and 壁（１．０）== true and 壁（０．ｍ１）== false and 壁（ｍ１．０）== true then
my.Direction = 目標方向()
       if 絶対対歩行者（０．ｍ１）== false then
          ForwardY(-1)
          goto owari
       end if
end if

'**********　この場所に固定**********
```

```
owari:

'**********************************************
*****************************************

'**********************************領域密度計
測
**********************************************
*******

MakeAgtSet(総歩行者, Universe.歩道.自由歩行者)
MakeAgtSet(ストッパー, Universe.歩道.ストッパー)

MakeAgtSetAroundPositionCell(領域１, Universe.歩道,
75, 16, 0, 4, 総歩行者 )
MakeAgtSetAroundPositionCell(領域２, Universe.歩道,
75, 16, 0, 4, ストッパー )

MakeAgtSetAroundPositionCell(領域３, Universe.歩道,
74, 25, 0, 4, 総歩行者 )
MakeAgtSetAroundPositionCell(領域４, Universe.歩道,
74, 25, 0, 4,ストッパー )

MakeAgtSetAroundPositionCell(領域５, Universe.歩道,
67, 16, 0, 4, 総歩行者 )
MakeAgtSetAroundPositionCell(領域６, Universe.歩道,
67, 16, 0, 4, ストッパー )

Universe.領域１ = CountAgtSet(領域１) +
CountAgtSet(領域２)
Universe.領域２ = CountAgtSet(領域３) +
CountAgtSet(領域４)
Universe.領域３ = CountAgtSet(領域５) +
CountAgtSet(領域６)

'**********************************************
*****************************************
}

Sub Smove(a as double, b as double){

if a >= 0 then
        Turn(-90)
        Forward(a * 1)
        Turn(90)
elseif a < 0 then
        Turn(90)
        Forward(-a * 1)
        Turn(-90)
end if

 Forward(b * my.固有巡航速度)

}

Function 目標方向() As Double{

    Dim rtn As Double


  if my.X >=my.WPX- 2 and my.X <= my.WPX + 2
      and my.Y >= my.WPY - 2 and my.Y<=
my.WPY +2 then
      my.WP 到達 = my.WP 到達 + 1
  end if

    my.WPX = Universe.歩道.Waypoint.X(WP 設定
())
    my.WPY = Universe.歩道.Waypoint.Y(WP 設定
())

    rtn = GetDirection(my.X, my.Y, my.WPX,
my.WPY, Universe.歩道)

    Return(rtn )
```

```
}

Function WP 設定() as integer {

    Dim i as integer
    Dim j as integer
    Dim WP as integer

        if my.色  == color_blue then
            i = 0
        elseif my.色  == color_red then
            i = 1

        end if

        j = my.WP 到達
        WP = Universe.WP 設定(i,j)

    return (WP)
}
```

このシミュレーションは WP を左・中心・上に一つずつ配置し,左から青の歩行者エージェントが発生し中心を通り上に向かい,上から赤い歩行者エージェントが発生し中心を通り左に向かうというルールになっている.歩行者がすれ違う際,ASPF ルールに基づき回避行動をとる.

７．１．２　ASPF 行動

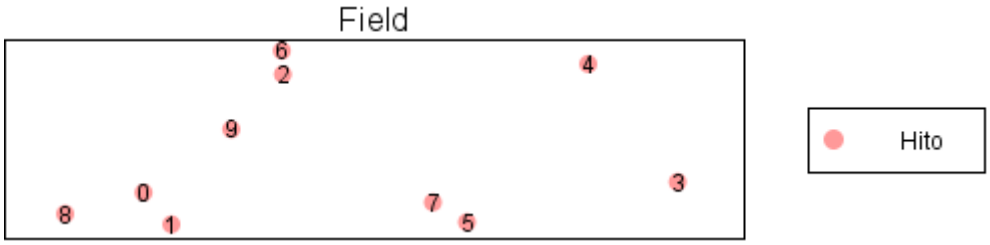


図７－２　ASPF 行動シミュレーション

歩行者エージェントのルール

```
Agt_Init{
Dim  自由歩行速度平均  As Double
Dim  自由歩行速度 SD As Double

        自由歩行速度平均  = 1.2
        自由歩行速度 SD = 0.3

        My.固有速度  =  NormInv(Rnd(),  自由歩行速度平均/1.2,  自由歩行速度 SD/1.2)

        If Universe.移動方向  == "自由方向" Then
                My.Direction=Rnd()*360//自由方向
        Else
                My.Direction=int(Rnd()*2)*180//2 方向
        End if
}

Agt_Step{
Dim  周囲の歩行者  As AgtSet
Dim  周囲の歩行者数  As Integer
Dim Ncell(12,12) As Boolean//  相対視野セル

Dim i As Integer
Dim j As Integer
Dim   X 軸上の差  As Double
Dim   Y 軸上の差  As Double
Dim   誰かとの距離  As Double
Dim   誰かの角度  As Double
Dim   誰かとの方角  As Double
```

```
Dim　相対 X 差 As Double
Dim　相対 Y 差 As Double

//＝＝＝＝＝ 相対視野セルの初期化の始まり＝＝＝＝＝
        For i = 0 to 10
                For j = 0 to 10
                        Ncell(i, j) = False
                Next j
        Next i
//＝＝＝＝＝相対視野セルの初期化の終わり＝＝＝＝＝


        MakeOneAgtSetAroundOwn(周囲の歩行者, 4, Universe.Field.Hito, False)
        //半径４セルの他者を周囲の歩行者内に収める


//＝＝＝＝＝相対座標への変換の始まり（プログラム Tips No.11-2 で解説）＝＝＝＝＝
        For Each 誰か in 周囲の歩行者
                誰かとの距離 = 0
                誰かとの方角 = 0

                誰かとの距離 = MeasureDistance(My.X, My.Y, 誰か.X, 誰か.Y, Universe.Field)

                X 軸上の差 = 誰か.X - My.X
                Y 軸上の差 = 誰か.Y - My.Y

                If X 軸上の差 == 0 then
                        If Y 軸上の差 >= 0 then
                                誰かの角度 = 90
                        Else
                                誰かの角度 = 270
                        End if
                Else
                        誰かの角度 = RadToDegree(Atn(Y 軸上の差 / X 軸上の差))
                End if

                If Y 軸上の差 >= 0　then
                        If X 軸上の差 >= 0 then//第一象限
                                誰かとの方角 = 誰かの角度-My.Direction
                        Elseif X 軸上の差 < 0 then//第二象限
                                誰かとの方角 =(誰かの角度-180)-My.Direction
                        End if
                Elseif Y 軸上の差 < 0　then
                        If　X 軸上の差 <0 then//第三象限
                                誰かとの方角 =(誰かの角度+180)-My.Direction
                        Elseif X 軸上の差 >=0 then//第四象限
                                誰かとの方角 = 誰かの角度-My.Direction
                        End if
                End if

                相対 X 差 = Round(誰かとの距離 * Cos(DegreeToRad(誰かとの方角)))
                相対 Y 差 = Round(誰かとの距離 * Sin(DegreeToRad(誰かとの方角)))

                Ncell(相対 X 差 + 4, 相対 Y 差 + 4) = True

        Next 誰か
//＝＝＝＝＝相対座標への変換の終わり＝＝＝＝＝
```

```
// ＝＝＝＝＝APPF の歩行行動ルールの始まり（プログラム Tips No.11-1 で解説）＝＝＝＝＝
        周囲の歩行者数 = CountAgtSet(周囲の歩行者)

        If 周囲の歩行者数 <16 then
        //以下は基本行動ルール：(4,4)を自己位置,(4+1,4)が一歩前
                If (Ncell(5,4) == True) And (Ncell(4,3) == True) And (Ncell(4,5) == True) Then
                        //do nothing
                        //ルール１
                Elseif (Ncell(5,4) == True) and (Ncell(4,3) == True) Then
                        Turn(-90)
                        Forward(My.固有速度)
                        Turn(90)
                        // ルール２
                Elseif (Ncell(5,4) == True) and (Ncell(4,5) == True) Then
                        Turn(90)
                        Forward(My.固有速度)
                        Turn(-90)
                        //ルール３
                Elseif Ncell(5,4) == True Then
                        If Rnd()<0.5 then
                                Turn(-90)
                                Forward(My. 固有速度)
                                Turn(90)
                        Else
                                Turn(90)
                                Forward(My. 固有速度)
                                Turn(-90)
                                // ルール４
                        End if
                Elseif (Ncell(5,4) == False) And (Ncell(6,4) == False) And (Ncell(7,4) == False) And (Ncell(8,4) == False) then
                        Forward(My.固有速度 * 3)
                        //ルール５
                Elseif (Ncell(5,4) == False) And (Ncell(6,4) == False) And (Ncell(7,4) == False) And (Ncell(8,4) == True) then
                        Forward(My.固有速度 * 2)
                        //ルール６
                Elseif (Ncell(5,4)==False) And (Ncell(6,4)==False) And (Ncell(7,4)==True) then
                        Forward(My.固有速度)
                        //ルール７
        //以下は対他減速ルール
                Elseif (Ncell(5,3)==False) And (Ncell(5,4)==False) And (Ncell(5,5)==False) And (Ncell(4,3)==False) And (Ncell(4,5)==False) And (Ncell(6,4)==True) then
                        If Rnd() <0.5 then
                                Turn(-90)
                                Forward(My. 固有速度)
                                Turn(90)
                                Forward(My. 固有速度)
                        Else
                                Turn(90)
                                Forward(My. 固有速度)
                                Turn(-90)
                                Forward(My. 固有速度)
                                //ルール８
                        End if
                Elseif (Ncell(5,4)==False) And (Ncell(5,5)==False) And (Ncell(4,3)==True) And (Ncell(4,5)==False) And (Ncell(6,4)==True) then
```

```
                Turn(-90)
                Forward(My.固有速度)
                Turn(90)
                Forward(My.固有速度)
                //ルール 9
            Elseif (Ncell(5,3)==False) And
(Ncell(5,4)==False) And (Ncell(4,3)==False) And
(Ncell(4,5)==True) And (Ncell(6,4)==True) then
                Turn(90)
                Forward(My.固有速度)
                Turn(-90)
                Forward(My.固有速度)
                //ルール 10
            Elseif (Ncell(5,4)==False) And
(Ncell(5,5)==False) And (Ncell(6,3)==True) then
                Turn(-90)
                Forward(My.固有速度)
                Turn(90)
                Forward(My.固有速度)
                //ルール 11
            Elseif (Ncell(5,3)==False) And
(Ncell(4,3)==False) And (Ncell(6,5)==True) then
                Turn(90)
                Forward(My.固有速度)
                Turn(-90)
                Forward(My.固有速度)
                //ルール 12
            Elseif (Ncell(4,4)==True) And
(Ncell(4,3)==True) And (Ncell(4,5)==True) then
                //do nothing
                //ルール 13
            Elseif (Ncell(4,3)==True) And
(Ncell(6,5)==True) then
                //do nothing
                //ルール 14
            Elseif (Ncell(6,3)==True) And
(Ncell(4,5)==True) then
                //do nothing
                //ルール 15
        //以下は対他回避ルール＝＝＝＝＝
            Elseif Ncell(5,5) == True then
                Turn(90)
                Forward(My.固有速度)
                Turn(-90)
                //ルール 16
            Elseif Ncell(5,3) == True then
                Turn(-90)
                Forward(My.固有速度)
                Turn(90)
                //ルール 17
            Elseif (Ncell(5,3)==False) And
(Ncell(4,3)==False) And (Ncell(4,5)==True) then
                Turn(90)
                Forward(My.固有速度)
                Turn(-90)
                Forward(My.固有速度)
                //ルール 18
            Elseif (Ncell(4,5)==False) And
(Ncell(5,5)==False) And (Ncell(4,3)==True) then
                Turn(-90)
                Forward(My.固有速度)
                Turn(90)
                Forward(My.固有速度)
                //ルール 19
        //以下は高密歩行ルール
            Else
                If Ncell(5,4) == True Then
                    //do nothing
                    //ルール 20
                Elseif (Ncell(5,4)==False)
And (Ncell(6,4)==True) Then
                    Forward(My. 固
有速度)
                    //ルール 21
                End if
            End if
        End if
// ＝＝＝＝＝APPF の歩行行動ルールの終わり＝＝
```

＝＝＝ }

この ASPF シミュレーションは人間らしい歩行ルールであり,左から来る歩行者エージェントと右から来る歩行者エージェントがすれ違う際に生じる避ける行動を細かくまとめたシミュレーションである.

７．１．３　障害物回避

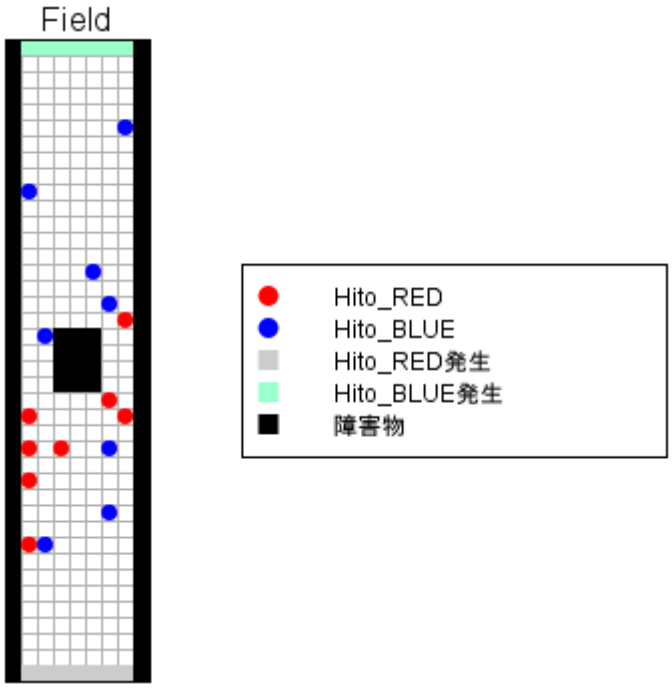


図７－３　**障害物回避シミュレーション**

歩行者エージェントのルール

```
Agt_Init{

}

Agt_Step{
Dim 前方 As Boolean
Dim 左側 As Boolean
Dim 右側 As Boolean
Dim 群集 As AgtSet
Dim 人数 As Integer

    前方 = False
    左側 = False
    右側 = False

    MakeAllAgtSetAroundOwnCell( 群 集 , 1, False)//半径 1 セル以内の他者（障害物）を探す
    人数 = CountAgtSet(群集)
    If 人数 > 0 Then//視野内に誰か（障害物）いるのなら
        For each 誰か in 群集
            If 誰か.X == My.X and 誰か.Y == My.Y + 1 Then//前方に他者（障害物）がいる
                前方 = True
            Elseif 誰か.X == My.X - 1 and 誰か.Y == My.Y Then//左側に他者（障害物）がいる
                左側 = True
            Elseif 誰か.X == My.X + 1 and 誰か.Y == My.Y Then//右側に他者（障害物）がいる
                右側 = True
            End if
        Next 群集
    End if
```

```
If 前方 == True Then//前方に他者（障害物）がいる場合
    If 右側 == False And 左側 == False Then//両側が空いているなら
        If Rnd() > 0.5 Then

ForwardXCell(1)//左側に移動
        Else

ForwardXCell(-1)//右側に移動
        End if
    Elseif 右側 == False then//右側が空いているなら
        ForwardXCell(1)
    Elseif 左側 == False Then//右側に他者（障害物）がいるが左側が空いているのなら
        ForwardXCell(-1)
    End if
Else//前方に他者（障害物）がいない場合
    ForwardYCell(1)//前方に移動
End if

If My.Y == 38 Then//上から 2 段目のセルに来たら
    KillAgt(My)
End if
}
```

このシミュレーションは歩行者エージェントが障害物エージェントを避けて通るシミュレーションである. 歩行者エージェントのルールとして, 障害物エージェントが視野に入ったらどちらかに避けるとなっている.

参考文献

http://www.bousai.metro.tokyo.jp/japanese/knowledge/material_h.html 東京都防災ホームページ

山影進：人工社会構築指南，書籍工房早川，2007 年 1 月 30 日.

渡邊裕介:震災時の避難行動に及ぼす情報活用の効果~避難行動のシミュレーション~, 平成 21 年度卒業論文.

地震調査研究推進本部：http://www.jishin.go.jp/main/index.html，2010 年 7 月 23 日閲覧.

瀧川翔：震災時の避難行動に及ぼす情報活用の効果～避難行動のシナリオ～，平成 21 年度卒業論文.

工藤知徳：震災時の早期の電子情報収集と活用に関する一提案，平成 22 年度卒業論文.

MAS コミュニティ：

http://mas.kke.co.jp/index.php，2010 年 8 月 1 日閲覧.

謝辞

本研究を進めるにあたり, ご多忙な中, 皆川勝教授, 佐藤安雄技師にはご指導, ご助言を頂きました. とても多くの迷惑を掛けてしまいましたがここまで成長させていただき, 誠ににありがとうございました.

付録

第一回中間発表概要
第二回中間発表概要
卒業論文発表会概要

マルチエージェントを用いた情報の利活用による帰宅困難者支援効果の検証

学生氏名　小泉祐亮
指導教員　皆川　勝

１．はじめに

日本は世界でも有数の地震大国といわれており，世界の地震エネルギーの 10 分の 1 が日本周辺に集中しているとも言われている．阪神・淡路大震災や新潟中越地震は記憶に新しい．そして地震の脅威はおさまることを知らず，今後は東海地震・東南海地震・首都直下型地震などの様々な地震の発生が予測されている．地震の発生を防ぐことは出来ないが，被害を軽くする方法であれば，様々な可能があると考えられる．本研究では震災時の避難のシミュレーションに着目する．研究は既に行われているが，現在では情報技術の進歩と共に携帯・パソコン情報端末から様々な情報を容易に手に入れることが出来る．そこで，それを考慮し被災者が効率的に避難できるための情報を被災者に与えることも可能である．そのために IT を用いた帰宅者支援システムを踏まえ，震災時に被災者に効率的に避難できる情報を時間経過と共に与え，その動きをシミュレーションする．

このように様々な情報を容易に手に入れることが出来る現在，情報利用による震災時の避難シミュレーションを行い，帰宅困難者支援のマルチエージェントによる有効性の検証することが本研究の目的である．

２．マルチエージェントによるシミュレーション

研究背景，研究目的を達成するに当たってマルチエージェントシステムがとても重要なのは，参考文献からも明らかになっている[5)]．

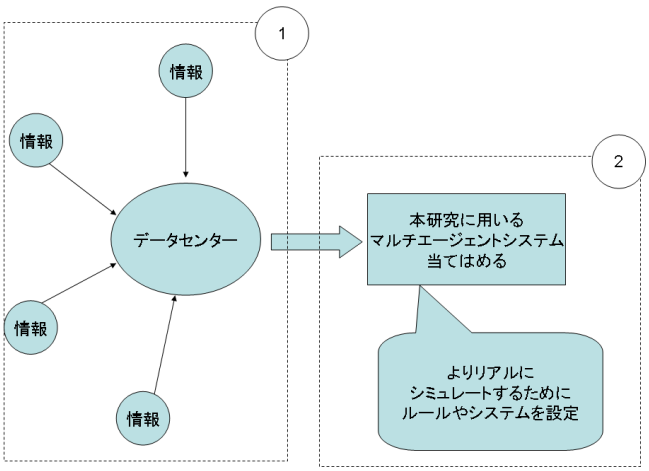
マルチエージェントシステムとは，まず多数の自律的に行動するエージェントから構成されるシステムであり，それぞれのエージェントは自分の目標を達成するように動き，システム全体の振る舞いはエージェント同士が相互に作用することによって決定される．つまり，複数の人々の動きをシミュレーションするには最適なシステムである．よって本研究ではマルチエージェントシステムを軸にシミュレーションを行うことにした．

マルチエージェントシステムを応用したソフトが様々ある上で，本研究では，artisoc を用いて，マルチエージェントシステムの技術を学び自分の研究に活用していく．

artisoc とは （アーティソック：artificial societies）は，人間同士の相互作用をコンピュータ上で誰もが簡単に再現することができ，ダイナミックに変化する社会現象を生きたまま分析できるマルチエージェント・シミュレータである．

図－１　シミュレーション例

３．帰宅困難者の行動のルール化



図－２　研究の流れ

図－2 は，被災者が携帯・パソコンなどを用いデータセンターに情報が送られるシステムである.情報の内容として，被害状況の画像・交通情報・避難所が提供される.このシステムは，参考文献から参照とする[5]. そして今回の研究では提案された情報をルール化し，シミュレーションとして動かし有効性検証していく.（提案がどれも利用可能であることを前提とし，この情報を元にシミュレーションを行う.）

エージェントのルール

１）情報の提供

混雑情報を時間ごとに提供する.

混雑度を理解した上で別のルートを提供する.

（ルートの判断は基準は，調査中である）

２）特性

年齢：年齢を分けることで歩行速度に違いが出る.

性別：男性，女性と分けることで歩行速度，歩ける距離に違いが出る.

位置：自宅と勤務地を知ることで自宅真での勤距離を算出することが出来る．算出することで帰宅困難者を判断する.

３）災害時刻

昼・夜の 2 つの時間帯でシミュレーションを行う.時間は昼が 10:00 夜が 17:00 と考えている.

理由としては，どちらも人々が街に留まっている状態であることと，夜は昼と違いは，暗いということで帰宅困難者の人数に違いが出るからである.

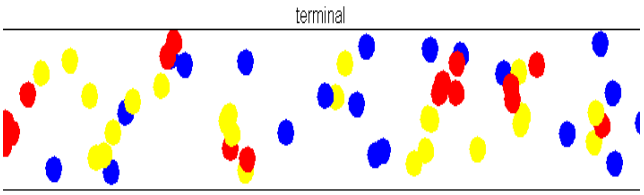
４）歩行速度

「歩行速度を混雑具合で区別する」

表－１ 混雑度ランクと歩行速度

混雑度ランク	混雑状況 [（ ）内は混雑度（人/㎡）]	混雑度（人/㎡）	歩行速度（km/h）
A	群集なだれが引き起こされる（7.2）*1	6～	～0.4
	ラッシュアワーの満員電車の状態（6.0-6.5）*3		
	ラッシュアワーの駅の改札口付近（6.0-6.5）*3		
B	ラッシュアワーの駅の階段周辺（5.5-6.0）*3	5.25～6	0.4～1
	危険性を伴う群集の圧力と心理的ストレスが大きくなり始める（5.4）*2		
C	駅の連絡路のラッシュ時で極めて混雑した状態（4.5-5.0）*3	4～5.25	1～2
	エレベータ内の満員状態（4.0-4.5）*3		
D	劇場での満員状態（3.5-4.0）*3	2.75～4	2～3
	ラッシュ時のオフィス街路（2.5-3.0）*3		
E	街路等で普通の歩行ができる（1.5-2.0）*3	1.5～2.75	3～4
F	街路で前の人を追い越せる状態（1.0-1.5）*3	～1.5	4
	街路で普通に混まずに歩ける（0.5-1.0）*3		

表－１は，エージェントの視野 1 にどれだけの人数のエージェントがいるか（混雑度）で，歩行速度を変化させるものである.



図－３　エージェントの歩行速度による色分け

青は，周りに 1 以下のエージェントがいる場合.黄は，周りに 2 以上のエージェントがいる場合．赤は，周りに 4 以上のエージェントがいる場合である.

４．おわりに

今後は，提供する情報が集まったので，まずは情報をルール化し，シミュレーションを行う．そして，出た結果から提案した情報が有効であるかを検証する．（有効であるか有効でないかを判断する基準は，資料を見ながら検討して行きたい.）そして，有効であることが検証できた場合，「特性」「災害時刻」「歩行速度」のルールを加えていき，シミュレーションの精度をあげていく.

参考文献

1)山影進：人工社会構築指南，書籍工房早川，2007 年 1 月 30 日.

2)渡邊裕介：震災時の避難行動に及ぼす情報活用の効果~避難行動のシミュレーション~, 平成 21 年度卒業論文.

3) 地震調査研究推進本部：http://www.jishin.go.jp/main/index.html，2010 年 7 月 23 日閲覧.

4)瀧川翔：震災時の避難行動に及ぼす情報活用の効果～避難行動のシナリオ～, 平成 21 年度卒業論文.

5)工藤知徳：震災時の早期の電子情報収集と活用に関する一提案，平成 22 年度卒業論文.

6)MAS コミュニティ：http://mas.kke.co.jp/index.php, 2010 年 8 月 1 日閲覧.

マルチエージェントを用いた情報の利活用による帰宅困難者支援効果の検証

計画マネジメント・皆川研究室　小泉祐亮
指導教員　皆川　勝

１．はじめに

日本は世界でも有数の地震大国といわれており，世界の地震エネルギーの 10 分の 1 が日本周辺に集中していると言われている．今後は東海地震・東南海地震・首都直下型地震などの様々な地震の発生が予測されている [1)]．地震の発生を防ぐことは出来ないが，被害を軽減する方法であれば，様々な可能性があると考えられる. その中で本研究では震災時に ICT と帰宅困難者支援システムを用いた避難シミュレーションに着目する．様々な情報を容易に手に入れることが出来る現在，情報利用による震災時の避難シミュレーションを行い，帰宅困難者支援のマルチエージェントによる有効性の検証することが本研究の目的である.

２．研究の流れ

研究を行うにあたって 2 つの項目に分けた.

1) 帰宅困難者支援システムの内容の作成

今回は情報利用に着目した上での研究であるので, 地震発生直後に必要な情報の提供・受け取り・データ整理などを行うことのできるシステムを考える.

この 2 つを今回の研究の軸とし, 研究を進めていき情報の利活用による帰宅困難者支援効果の検証を行う.

2) シミュレーション作成

今回の研究内容に適したシミュレーション行うために artisoc[2)]を利用した. その際, 歩行者の行動パターンをルール化し, それをシミュレーションに反映することで歩行者エージェントを動かす.

３．帰宅困難者支援システム

1) システムの概要

帰宅困難者支援システムとは, 地震発生時に時間帯や自宅の距離など検証し, 歩行で自宅に帰ることのできない帰宅困難者を支援するためのシステムである [3)4)]. このシステムは主に, 帰宅困難者への情報提供, 利用者から情報を取得することで機能する.

2) システムが扱う情報

被害写真・コメント・地図情報・避難場所・地震情報・避難ルート情報・GPS 情報等

3) システムの役割

①歩行者等からの情報取得

主に歩行者等から 1) で示した情報を取得する. 取得方法としては, カテゴリー（被害状況, 火災, 渋滞, 死傷者）別による情報取得・写真, コメントによる情報取得を考えている.

②歩行者等への情報提供

このシステムは誰でも利用可能であることを前提に主に歩行者等へ情報を提供する. 提供方法としては, カテゴリー（地震情報, 現在地, 避難場所, 周辺）検索・キーワード検索・周辺検索などによる情報提供を考えている.

４．マルチエージェントによるシミュレーション

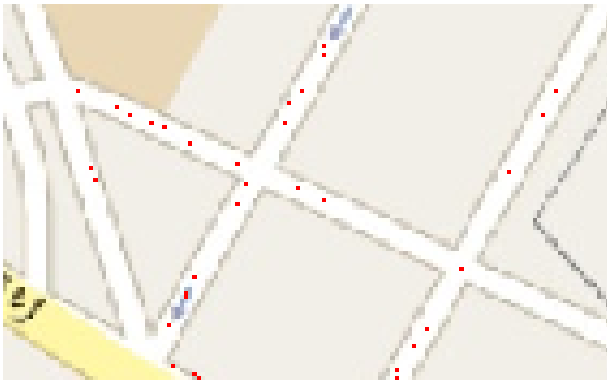
日本では, ICT が普及するようになってからは大地震はなく, 災害時のデータが少ない. そのため ICT を用いての避難行動の利用例が無い. そこでマルチエージェントシステムを使うことで地震が起きたことを想定したシミュレーションを行う.

マルチエージェントシステムとは，まず多数の自律的に行動するエージェントから構成されるシステムであり，それぞれのエージェントは自分の目標を達成するように動き，システム全体の振る舞いはエージェント同士が相互に作用することによって決定される．つまり，複数の人々の動きをシミュレーションするには最適なシステムである．よって本研究ではマルチエージェントシステムを軸にシミュレーションを行うことにした.

マルチエージェントシステムを応用したソフトが様々ある上で，本研究では artisoc を用いてマルチエ

キーワード　帰宅困難者，マルチエージェント，避難

ージェントシステムの技術を学び，自分の研究に活用していく[5]. 図－1 は, マルチエージェントを用いたシミュレーション例である.



図－１　シミュレーション例

５．歩行者エージェントの移動（ASPF）

歩行者の移動パターンをエージェントに反映するにあたって ASPF の利用を考えている.

1) ASPF とは

ASPF(Agent Simulation of Pedestrian Flow)プロジェクトと称して, 名古屋工業大学兼田研究室の卒業研究や修士研究のメンバーが取り組んできた歩行者モデル開発と一連のバージョンアップの成果である[6]. 図－2 は ASPF の行動パターンの一例である.

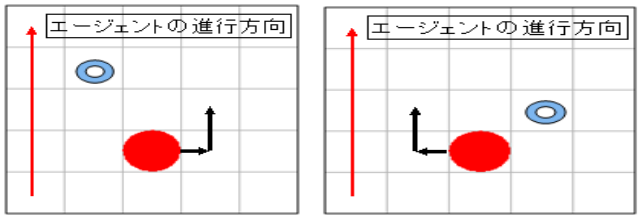
2) ASPF の歩行行動ルール

①基本行動ルール：低密度歩行時における主にエージェントの直進行動を規定する.

②対他減速ルール：低密度歩行時におもに前後のエージェントと間隔を保つため, 他のエージェントに近づくにつれ減速する.

③対他回避ルール：低密度歩行時におもに左右のエージェントと間隔を保つため, 他者を回避する.

④高密度ルール：高密度歩行時に人々は左右の間隔を減らすよりも前後の間隔を減らす動作をする.



➡移動方向　●歩行者　◎歩行者

図―２ASPF 歩行行動の例

3) システムの利用率

情報の利用率の変化により避難シュミレーションにどのような変化が生じるかを見るために，このシステムの利用率を設定し，0～100%の間で数値を自由に変えられるようにする．この際に，歩行者エージェントからの情報の取得と歩行者エージェントへの情報提供をシステムの利用率として一緒に考える. 例えばシステムに情報を提供しかつシステムから情報を取得することと，システムから情報を取得することを，同じく情報利用とする.

６．まとめ

今回，シミュレーション作成では歩行者エージェントの移動パターン(ASPF)をルール化し, それをシミュレーションに反映させることができた． システムの内容の作成では, 被災地で情報を利用する構想の大枠を考案した.

７．今後の展望

ASPF のルールを元に，具体的な場所や帰宅困難者になるであろう人数を検討・設定し，避難シュミュレーションを形にしていく. そして帰宅困難者支援システムの利用率のパラメータを変化させることで情報がシミュレーションに与える効果を検証していく. システムの内容については，情報の利用法や運用の仕方についてより明確にしていきたい.

参考文献

1)地震調査研究推進本部：
http://www.jishin.go.jp/main/index.html, 2010 年 7 月 23 日閲覧.

2)MAS コミュニティ：
http://mas.kke.co.jp/index.php，2010 年 8 月 1 日閲覧.

3)渡邊裕介：震災時の避難行動に及ぼす情報活用の効果，平成 21 年度卒業論文.

4)工藤知徳：震災時の早期の電子情報収集と活用に関する一提案，平成 22 年度卒業論文.

5)山影進：人工社会構築指南，書籍工房早川，2007 年 1 月 30 日.

6)兼田敏之 artisoc で始める歩行者エージェントシミュレーション，構造計画研究所，2010

マルチエージェントを用いた情報の利活用による帰宅困難者支援効果の検証の基礎的研究

計画マネジメント・皆川研究室　小泉祐亮
指導教員　皆川　勝

１．はじめに

日本は世界でも有数の地震大国といわれており，世界の地震エネルギーの10分の1が日本周辺に集中していると言われている．今後は東海地震・東南海地震・首都直下型地震などの様々な地震の発生が予測されている[1)]．地震の発生を防ぐことは出来ないが，被害を軽減する方法であれば，様々な可能性があると考えられる.その中で本研究では震災時にICTと帰宅困難者支援システムを用いた避難シミュレーションに着目する．様々な情報を容易に手に入れることが出来る現在，情報利用による震災時の避難シミュレーションを行い，帰宅困難者支援のマルチエージェントによる有効性の検証することが本研究の目的である.

２．研究の流れ

研究を行うにあたって2つの項目に分けた.

1)帰宅困難者支援システムの内容の検討

今回は情報利用に着目した上での研究であるので,地震発生直後に必要な情報の提供・受け取り・データ整理などを行うことのできるシステムを考える.

2)シミュレーション作成

今回の研究内容に適したシミュレーション行うためにartisoc[1)]を利用した.その際,歩行者の行動パターンをルール化し,それをシミュレーションに反映することで歩行者エージェントを動かす.

この2つを今回の研究の軸とし,研究を進めていき情報の利活用による帰宅困難者支援効果の検証を行う.

３．帰宅困難者支援システム

1)システムの概要

帰宅困難者支援システムとは,地震発生時に時間帯や自宅の距離など検証し,歩行で自宅に帰ることのできない帰宅困難者を支援するためのシステムである.このシステムは主に,帰宅困難者への情報提供,利用者から情報を取得することで機能する.

2)システムが扱う情報

被害写真・コメント・地図情報・避難場所・地震情報・避難ルート情報・GPS情報等

3)情報をどのように支援するか

今回帰宅困難者に情報をどのように支援するかの方法として,携帯電話・ノートパソコンによる情報の提供,取得を考えている.

4)システムを利用する機器

今研究ではシステムを利用する機器として「携帯電話」「ノートパソコン」としている.理由としては,普及率がとても高く日本は79.6%（2006年国連貿易開発会議）である.また,軽量であるため持ち運んで使うことが多く,身に着けていることが多い.そして,震災時に起こる機器等の被害で転倒や落下,移動が大半を占めている中,被害率がホストコンピュータが 26%であったのに対し,ノートパソコンはわずか1%と,小型軽量になるほど被害は軽微になることが分かっているからである.

４．シミュレーション

1)マルチエージェントによるシミュレーション

日本では,ICTが普及するようになってからは大地震はなく,災害時のデータが少ない.そのためICTを用いての避難行動の利用例が無い.そこでマルチエージェントシステムを使うことで地震が起きたことを想定したシミュレーションを行う.

マルチエージェントシステムとは，まず多数の自律的に行動するエージェントから構成されるシステムであり，それぞれのエージェントは自分の目標を達成するように動き，システム全体の振る舞いはエージェント同士が相互に作用することによって決定される．つまり，複数の人々の動きをシミュレーションするには最適なシステムである．よって本研究ではマルチエージェントシステムを軸にシミュレーションを行うことにした.

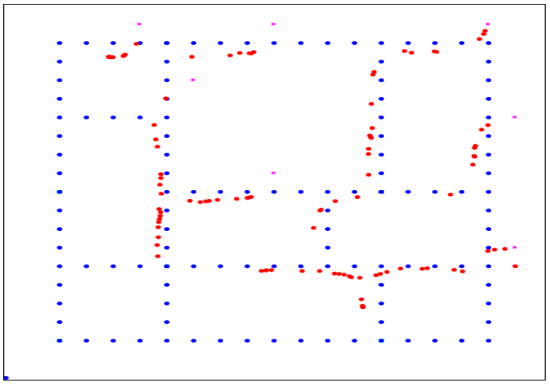
マルチエージェントシステムを応用したソフトが様々ある上で，本研究ではartisocを用いてマルチエ

ージェントシステムの技術を学び，自分の研究に活用していく[2)]．

2) シミュレーションの概要

今回行ったシミュレーションは,マルチエージェントを用いた情報の利活用による帰宅困難者支援効果の検証を行うために必要な歩行者エージェントの基礎的シミュレーションである.今回は歩行者エージェントがどのような判断で目的地に達するかを行う.

3) シミュレーション



図－1　シミュレーション

図－2

この図は,artisoc でシミュレーションを行う実際の画面である.

①人：歩行者エージェントであり,シミュレーションの中で人の役割を果たす.

②WP : Waypoint の略語であり,シミュレーションの中で道・道路を表す WP の配列は座標により設定した.

③目的地：歩行者エージェントが目指す行き先.

④発生点：歩行者エージェントが生まれる場所.

4) 歩行者エージェントの行動ルール

①半径 5 セル以内の WP を視野に入れる

②半径 5 セル以内の視野の中に WP が 0 の場合歩行者エージェントは目的地の方向に進む

③視野の範囲が 30 度の範囲である.そして視野に入った WP と自らの距離を測る.

④もし視野 30 度の範囲に WP があった場合,その WP の方向に進む.

⑤もし WP と自らの距離が 0.1 未満だった場合,目的地の方向へ進む.

⑥もし視野 30 度以上の範囲に WP がある場合,目的地方向に進む.

⑦もし視野 30 度以内の範囲に WP がある場合,目的地方向に向かう.

このルールをまとめると,歩行者エージェントは発生点で生まれ生まれたところから視野 5 のセルの範囲内･視野 30 度以内を基準にし,視野に WP が入った場合そのポイントまでの距離を測り移動する.視野に複数の WP が入った場合は自分に一番近い目的地方向の WP に進む.この場合,近づきすぎた WP は目標で無くなり,新たに視野に入った WP に向かって移動を繰り返す.視野に何もない場合は,目的地方向に進む.また,シミュレーションの際歩行者エージェントが WP の上を通らない理由として,目的地に向かうという行動を取るために目的地方向に引っ張られる形になってしまうからであると考えられる.

５．まとめ

今回は基礎研究として,マルチエージェントシステムを用いた歩行者の行動をシミュレーションすることが出来た.今後は東京 23 区で発生する帰宅困難者を支援するシミュレーションを行うために,歩行者エージェントルールの改善や情報を持たせることを行い,最終的には 23 区から神奈川に数日間をかけて帰宅すといった大掛かりなシミュレーションにたどり着ければと考える.

参考文献

1)MAS コミュニティ：
http://mas.kke.co.jp/index.php，2010 年 8 月 1 日閲覧.
2)山影進：人工社会構築指南，書籍工房早川，
2007 年 1 月 30 日.
3)兼田敏之 artisoc で始める歩行者エージェントシミュレーション，構造計画研究所，2010