マルチエージェントシステムを用いた歩きスマホ混在の交差点シミュレーション解析

The analysis of Simulation which is crossroads included both normal worker and the worker with looking at smartphone by Multi-Agent system

所属: 法政大学 情報科学部 コンピュータ科学科

学籍番号: 11k0105

氏名: 池田歩生

Name: Ayumu Ikeda
E-mail: Ayumu.Ikeda.3e@stu.hosei.ac.jp

指導教官: 小池誠彦 教授

目次

Abstract

This paper, using a multi-agent system, I make the simulation which simulate the scramble intersection mixed the general worker and walking with using smartphone, and analyze the results. Why I want to do so is the experiments of a simulation of the only the walker with smartphone. I used the law of Helbing representing the behavior at the time of panic in one expression, and I want to simulate general pedestrian was also included in the scramble crossing in a state in which close to reality as possible. There are some conditions that were implemented for. One thing is if the traffic light is blinking, people were in hurry, and one is to add various kinds of people, including general pedestrians. I implement this kind of conditions, while changing the ratio of the worker with smartphone and general worker, in total 500, and I simulate this. And as a result, I evaluated the agent by the ratio of the collision number to the number of all agents and ratio of what all agents arrive at the destination which agent set by itself. As a result of evaluation, normal workers almost arrive at the destination, but persons who have a smartphone during walking can't arrive about 5%. In addition, the probability of collision was increased each time the ratio of number of people increases. Just as the collision rate either be only on one side of the agent was down from the peak.

1. まえがき

2014 年 3 月 29 日，NTT ドコモからある動画が発表された．[1] それは「もし渋谷のスクランブル交差点で全員歩きスマホをした場合，どのような結果が出るのか」というシミュレーションを録画した動画であった．歩きスマホというのは歩きながらスマートフォンを操作する行為である．この行為は前を見ることが少ないので，危険を予測することが難しく，推奨されない行為である．しかし，私はこれを「あまりにも非現実的ではないか」と考え，これを現実に則したものにしたらどうなるのだろうかという疑問を持った．

また，類似した研究として，1 つ目に「知的マルチエージェント交通流シミュレータ MATES の開発」[2]という論文がある．この研究は車の交通状況をマルチエージェントシステムで再現し，その後仮想で追加の道を作った時に，元の交通状況と比べてどのような影響が生まれるかということをシミュレーションで行い，予測しようと試みたものである．この論文に用いられたマルチエージェントシステムを見ると，前述した歩きスマホのシミュレーションのシステムに用いられたのではないかと考えられる．

そしてもう 1 つは，「Simulating dynamical features of escape panic」[3]という論文である．これは大災害などが起こったと仮定し，パニック状態に陥った歩行者のシミュレーションができないかと試みた研究であり，結果，ある 1 つの法則を導入するだけでパニック時の人の動きをシミュレートできるという結論を出した研究である．

よって，本研究の目的は，主に歩きスマホをする歩行者と前を見て普通に歩く歩行者の存在する比率を変えシミュレーションし，その結果を比較し評価する．

また，先ほどの，スマホ歩きをする歩行者のみでシミュレーションを行ったシステムは，前述した交通流の研究をした論文も鑑みると，マルチエージェントシステムを用いたのではないかと考えられる．よって，本研究でもマルチエージェントシステムを用いシミュレーションを行う．マルチエージェントシステムは，それぞれのエージェントに対し異なった判定アルゴリズムを設定できるという特徴があり，それによりそれぞれの歩行者を個別に特徴づけることが容易であるため，その結果，比較的簡単に擬似的な社会を構成することが可能なのである．

また，シミュレーションする際の行動規則として，こちらも先程述べたパニック時の行動を研究した際の法則を用いる．これはただ 1 つの法則のみを使用しシミュレートしているため，私でも容易に実装できるのではないかと考えたためである．

これらを利用し，一般の歩行者と歩きスマホをしている者が混在した交差点でのシミュレーションを行う．そして、それぞれの生成されたエージェント数に対するそれぞれの衝突率や時間内に目的地に到達した人数の割合，到達率を主に評価基準とした．実行条件は両方の総エージェント数を 500 に固定して，その比率を 100 人ずつ変え 45 秒シミュレーションする．そのシミュレーションを 10 回繰り返し結果の考察を行った．

まずシミュレーションを実行した結果の到達率に関して．これは歩きスマホをしている

ものの方が 5%ほど低い結果になった．そして衝突率については，歩きスマホと一般歩行者の差が激しい比率の時ほど値が大きくなるが，0 になると衝突率が下がるという結果が得られたので報告する．

2. 既存技術

本研究で扱うものの詳しい説明をする.

2.1. 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，それぞれの要素がどのような行動をするか理解しシステムに組み込めても，それを集めた集合全体の挙動をシミュレートできない時に主に使う．全体の挙動が個々の要素に分けられない時のことを複雑系と呼ぶが，その複雑系というのは，構成している要素の相互作用が比較的簡単なものであっても，全体としてみると非常に複雑な現象が生じることがある．これを創発と言う．その創発されたシステムをシミュレートするときや，そうでないただ単なる複雑系をシミュレートするときにマルチエージェントシステムは有用に働くのである.

このマルチエージェントシステムを実装したソフトとしてartisoc[4]というソフトを用いる。このソフトは他のソフトと比べて主に違うことは，座標があり，それを用いることが可能なためである．また，学生であれば使用が無料という部分も採用した重要な要素である.

2.2. ヘルビングの法則

この法則は大災害時等のパニックに陥った時の人の動きをモデル化し予測できないかという考えのもと考えだされた法則である．統計学的な観点から，もし人がパニックに陥った時は次のような特徴にまとめることが出来る，とこの論文は言っている.

(1) パニック時の人は通常時より早く移動しようとする.
(2) 群衆の中の個人同士は自然に相互作用が物理的なものになる.
(3) 出入口等の通路が一時的に狭くなる場所，つまりボトルネックの通過が協調的でなくなる.
(4) そうして，アーチ型の「詰まり」が生じる.
(5) その「詰まり」が時間に比例して肥大化する.
(6) その「詰まり」が最高潮になると，危険な圧力が発生する．具体的にどのような力かというと，約 4,450Nm-1 という力になる．これがどの様な力かというと，煉瓦の壁を壊せるほどまでになる.
(7) また、倒れた人やけが人が更に「障害物」となって局所的なボトルネックが生じ，さらに「詰まり」が肥大化する.
(8) また，人は自分以外の大衆の行動と同じ行動をしようとする.
(9) そのために，人が集中している出口に更に人が集中し，代わりの出口は使用されないことがしばしばある.

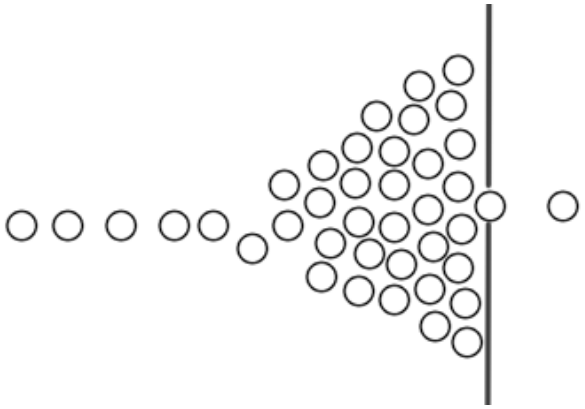


図 1　ヘルビングの法則のイメージ図

このような特徴があると伝えている．

そして，この力を示す数式は，ひとつの加速式で表示される．

$$m_i \frac{dv_i}{dt} = m_i \frac{v_i{}^0(t)e_i{}^0(t) - v_i(t)}{dt} + \sum_{j(\neq i)} f_{ij} + \sum_{W} f_{iw}$$

i はある個人に付けられた数字とする．mi は歩行者の質量であり，一定の方向 e0i の方向に速度 v0i で移動する．vi は他人の存在で変更した後の、つまり現在の速度になる．そしてΣfij は他人から受ける力の合計，Σfiw は壁から受ける力の合計を表している．

また，fij は次のように示される．j は i とは異なる人を指す．

$$\boldsymbol{f}_{ij} = \left\{ A_i \exp\left[\frac{(r_{ij} - d_{ij})}{B_i}\right] + \mathrm{k}_1 g(r_{ij} - d_{ij}) \right\} \boldsymbol{n}_{ij} + k_2 g(r_{ij} - d_{ij}) \Delta \mathrm{v}_{ij}^t \boldsymbol{t}_{ij}$$

Ai・Bi は定数，rij は二者の位置の変化の和，dij は両者の位置の差の絶対値を示す．また, もし rij – dij の値が負の場合であるとき, 両者はぶつかっていることになる. また, k1(rij - dij) * nij は i と j の間に働く心理的斥力であり，k2(rij - dij) Δvtji * tij は i と j の相対的な接線方向の運動を妨げる力である．tij は接線方向を意味し，Δvtij は i・j 両者の接線の速度の差を示している．k1，k2 は大きな定数を示す.

そして，fiw は次のように示される．

$$\boldsymbol{f}_{\mathrm{iw}} = \left\{ A_i \exp\left[\frac{(r_{ij} - d_{ij})}{B_i}\right] + \mathrm{k}_1 g(r_i - d_{iw}) \right\} \boldsymbol{n}_{iw} + k_2 g(r_i - d_{iw})(v_i \cdot \boldsymbol{t}_{iw}) \boldsymbol{t}_{iw}$$

これは先程の式 fij について，対象を壁に変えたのみである．niw は壁に垂直な方向への力を示し，tiw は接線方向への力を示す.

つまり，この式は人の速度を加速度で管理し，その加速度の変化は今までの速度から換算される加速度に加え他の人や壁からも加速度への力を受ける事を示す.

3. 実験システムの構築

現実に則したものとして実装したい事柄としては，次の通り．

① 出発地と目的地を設定し，向かう．また，目的地と出発地を違う場所にする．

② 視界内に他のエージェントが居る場合，回避する．

③ 目的地の周辺に到着した場合，消える．周辺にした理由としては，交差点には歩道と道路があり，目的地の歩道についた時点で歩道に沿って歩行者が行きたい方向へ向かうため，本研究の目的である交差点内のシミュレーションに入らないと考えたためである．

④ 衝突した場合，カウントし，回避行動を取る．

⑤ また，渋谷スクランブル交差点でのシミュレーションではエージェントの再発生がないようだったため，信号が青の時間内であれば新たに移動速度など諸々の設定をし直し再発生させる．

そして，歩きスマホをしている歩行者と一般の歩行者の違いとして，

Ⅰ 視界の広さに違いをもたせる．

Ⅱ 進行速度に違いをもたせる．

おおまかにはこのようなものだが，②，④に関しては更に追加の条件を加えなくてはならないと考えている．

②については，ヘルビングの法則の通り，相手からのベクトルを受けて進行方向を変え，それにより回避行動を行っているように見せる．さらに，今回の場合はパニック時の行動シミュレーションではないので，人は単調な動きをしない．そのため，条件を追加しなくてはならない．そのため，考えだした条件は，相手の進行方向とは逆の方向にベクトルを強く受けるということである．普段の歩行を自分で考察してみたところ，人や車の進行方向に方向を定めないようなルートを導き出しているように感じた．特に，相手が自分より速い速度である場合や，相手が車など自分より強い物の場合は相手の前に出ないようなルートを選択しやすい．

また，④について．衝突の判定自体は容易に可能である．だが，衝突した後の行動が難しく，衝突し、回避した先でまた衝突する可能性があることや，衝突した後，回避行動を取る方向の設定をどう解決するかが問題である．まず，回避中の衝突に関しては，① 衝突した場合の行動をするかどうか，② 今衝突中であるかどうか，③ 目的地についたかどうか，④ ①～③のどれにも当てはまらない時通常の行動を行う，の順で判定をすることによって問題の解決とする．そして，回避行動を取る方向については，進行方向を軸として左右の視界内の人数の比率を計算し，より少ない方向へ向かうようにすれば良い．

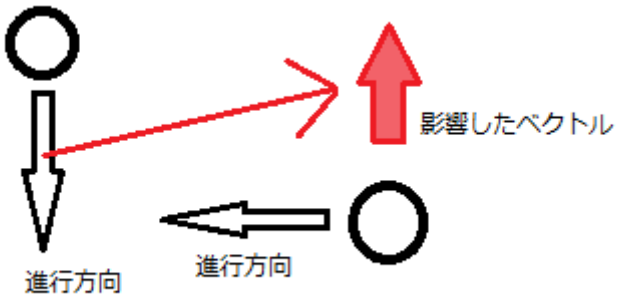


図 2　影響したイメージ図

現実に則したものにするのならば，他にも多くの条件を加えなければならない．例えば，現実には歩きスマホと一般の歩行者しかいないわけではなく，自転車を運転するものであったり，老人であったり，二人組やそれ以上の人数で固まって動いていたりである．もちろんそういった条件がなくとも人それぞれ歩行速度が異なる．他にも，途中で歩きスマホをやめたり，逆にし始めたりする人もいるだろう．老人や自転車を表現する方法としては，移動速度をエージェントそれぞれで変化させることで可能である．ただ，ペアもしくはそれ以上の人数で固まっての行動の場合，行動の条件が複雑多岐に渡る．まず相方を設定し，相方と離れず行動する．もし前に人がいる場合別れて避けるか同じ方向に避けるかを決め避ける．もし衝突してしまった場合，衝突した方としていない方でまた行動が異なる．つまり，人数が多くなった時の行動パターンは 1 人で動く場合と異なってくるのである．

また，人は信号が変わりそうになると急ぐ．現実に則したものならばこの条件も反映させなければいけない．これは開始時間を記憶し，ここから一定時間経った時に速度の上昇をかけることで実現可能である．

4. 実装・実行

4.1. 実装条件

3 で色々と提案を行ったが，この中で実現させたのは以下のとおり．まずは両者に共通した条件について．

① 出発地と目的地を設定し，向かう．また，目的地と出発地を違う場所にする．

② 視界内に他のエージェントが居る場合，ヘルビングの法則を用い，回避する．

③ 目的地の周辺に到着した場合，消える．

④ 衝突した場合，カウントし，回避行動を取る．回避行動の方向としては，進行方向を軸として左右のエージェントの比率を割り出し，より数が少ない方向へ 45 度傾ける．また，衝突中であっても再度衝突した場合，また衝突したと判定するようになっている．

⑤ スマホ歩きをする者と一般の歩行者の中でも移動速度に変化をつける．

⑥ ある一定時間が経った時，つまり信号が変わりそうになった時に全体の進行速度を上昇させる．

両者の違いとして実装したのは，

(1) スマホ歩きと一般の歩行者の違いとして，視界の広さに違いを生じさせ，また歩行速度にも違いをもたせる．

のみである．

逆に実装しなかった主な条件は，

Ⅰ エージェントのペアを実装する．

Ⅱ 時々歩きスマホをする歩行者の実装．

Ⅲ ヘルビングの法則に加え，他のエージェントの進行方向から影響を受け進行方向や進行速度に反映させる．

となっている．

実行した画面は以下のとおり．

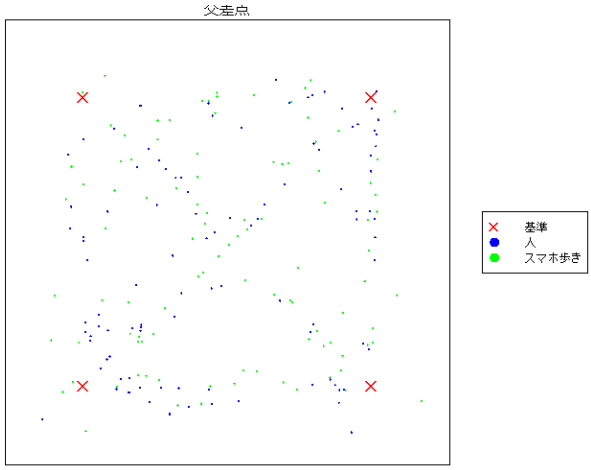


図 3　実行画面

実行する条件としては,

① 縦と横の長さは 200.

② エージェントの移動速度は一般人が 1 ミリ秒につき 0.4，0.5，0.6 のうちからランダムで決められ，歩きスマホは 0.1，0.2，0.3 の中からランダムで決められる.

③ 交差点の基準となる位置は(35,35) (165,35) (35,165) (165,165).

④ 反発する際の基準となる係数はそれぞれのスピードとなる.

⑤ 歩きスマホをする者の視界は 0.15，一般の歩行者の視界は 3.

⑥ ゴールしたら即座に新規エージェントを再生成する.

⑦ 信号が青である時間は 45 秒，青の点滅状態と判定するのは 35 秒，また 40 秒でエージェントの再生成を停止する．これは青の点滅状態でも入る人はいるが，それでも渡れないと判断した時間からは入らないことからこういう条件にした.

⑧ 歩きスマホをする者と一般の歩行者の合計を 500 人とし, シミュレーションとしてそれぞれの比率を 100 人ずつ変えて行う.

⑨ 上記すべての条件を内包したシミュレーションを 1 回とし, 比率 100 人毎に同じ条件で 10 回行う.

⑩ 10 回行った結果，それぞれのエージェントが目的地に到達した比率，それぞれのエージェントが生成された数に対する衝突の比率を主に評価の基準とする.

となっている．それぞれの歩行者の歩行スピードの平均の数字や実際のスクランブル交差点を基準の位置に合わせる等，出来る限り現実に則したものにしたかったが，背景画像が原因不明の使用不可等の理由により諦めた.

ただ、先の渋谷のスクランブル交差点での歩きスマホのみのシミュレーションによると，歩きスマホをしている者の視界は一般の歩行者の 20 分の 1 であることが判明しているために，歩きスマホをしている者の視界は 0.15 で一般の歩行者の視界は 3 という数字にしている．開始時の人数は機械のスペックを考慮し，安定に動きかつ計算しやすい 500 人と設定した．衝突の判定はぶつかられた方ではなくぶつかりに行った方の件数を増加させている.

4.2. 実装コード

すべてのコードの流れについて説明する.

まず，このソフトの実行するプログラムコードは 5 段階に分かれていて，

① 最初に 1 度行うコード

② エージェントの行動を設定したコードを実行する前に行うコード

③ エージェントの行動を設定したコード

④ エージェントの行動を設定したコードを実行した後に行うコード

⑤ 最後に 1 回だけ行うコード

になっている．そして，②～④のプログラムコードが繰り返し行われシミュレーションを実行している．これを②～④の流れをステップと呼ぶ．よって，このソフトは①→②→③→④→②→③→④→・・・→③→④→⑤という流れで実行される．また，シミュレーション中のエージェントのコードが実行される順番はランダムになっており，どちらか一方のみの種類のエージェントだけが先に動くということはない．また，これ以降交差点の基準となるエージェントを基準エージェント，スマホ歩きをしているエージェントをスマホ歩きエージェント，一般の歩行者のエージェントを一般エージェントと呼ぶことにする.

最初に①番に当たるコードが行う事柄は 4 つある．まず，各々の数字の初期化を行う．また，ここで現在の時刻を記憶し，実行する時間の設定やエージェントの移動時間が早くなる時間の設定も行う．次に，あらかじめ設定した基準のエージェント数を元に交差点の基準を設定する．その次にこちらも事前に決めていたエージェント数を元に一般エージェントと歩きスマホエージェントを作り出す．そして最後に，個々のエージェントのゴールの基準となるエージェントの設定や，自分のスタートの位置を決める.

②番に当たるプログラムコードが行うことは 1 つ．ランダムにスタート位置を決めているため，どうしても初期で何人か衝突判定が出てしまう．そのため，ある変数が 0 の時衝突判定が出た数字を自身に記憶するという事を行っている．ただ，今回のシミュレーションに関しては②番に当たるコードと④番に当たるコードにあまり違いがないため，これは④番に設定しても問題はない.

③番に行うプログラムコードはエージェントの行動規則を設定するものなので一番長い.また、ほぼ一般の歩行者とスマホ歩きをしている者の行動が 4.1 で述べた条件により同じである．異なるのはここで設定する「視界」と「移動速度」のみである．まず，ある一定時間以降は移動速度が上昇するのでその条件に合うかを問い，上昇させるか決定する．その後，もし画面内であれば移動する．次に周囲のエージェントを探し出し，基準エージェントとそれ以外のエージェントで分ける．もし探しだした基準エージェントがゴールだった場合，ゴールの判定に飛ぶように設定されループを抜ける．それ以外は処理を続ける．また、この時基準エージェントは独自の視界をもっている様に設定し，エージェントが持っている視界とは異なって判定している．そして次に衝突判定を行い，もし衝突していると判定されれば衝突の処理へ飛ぶ．また，この時同時に周囲にエージェントが何人居るかど

うかを判定し，後のヘルビングの法則適用時の行動の時に反映させるようにしている．そしてここで，実際にそれぞれ判定された行動をするメソッドへ飛ぶ．もし衝突判定が ON の場合，衝突のメソッドへ飛ぶ．次にもし衝突中という場合，衝突中の行動をするメソッドへ飛ぶ．その次，もし目的地へ到達したという判定の時，つまり基準エージェントの視界に触れたと判定された時は画面外へ飛ぶ．また，ここで衝突件数が追加される．最後にヘルビングの法則を用いた移動をするメソッドへ飛ぶが，視界内に誰もいない時は法則を通して無駄な計算をさせても機械の処理が遅くなるのみなので，その時は目的地へ移動するという式を使用している．

衝突のメソッドは自身が 1 歩下がり，更に自身は衝突中であるというフラグを立てる．

衝突中であるというメソッドは，最初のみ自身の進行方向を 45 度傾ける行動をする．傾ける方向は視界内のエージェントが少ない方向へ傾けている．そしてその傾けた方向に進む．またこの行為を 4 回行うと通常の行動へ戻るようにもなっている．

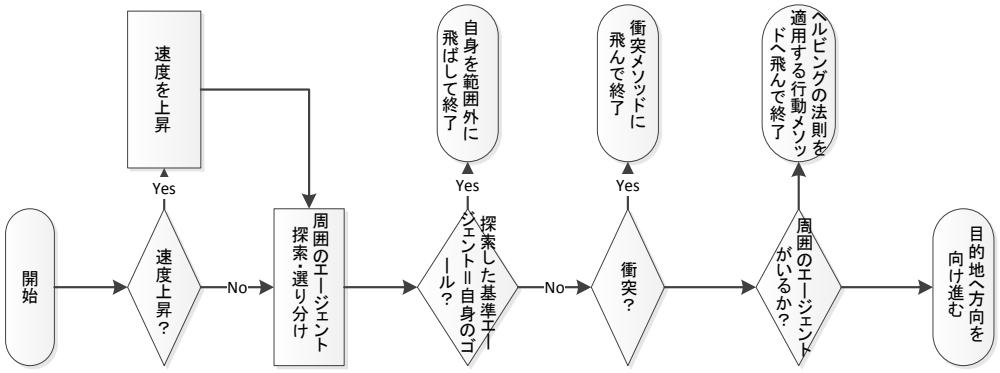


図 4　行動分岐図

次に④番のプログラムコードの紹介を行う．まず現在時刻を取得し，始めた時間からどのくらい経ったか計算し，まず終了時間だったら終了の判定をする．次に青が点滅している時間かつエージェントが再発生しない時間の時の判定をし，その後青が点滅している時間でかつエージェントが再発生する時間の時の判定をする．それ以外の時は普通に再発生させる．再発生の判定は画面の表示範囲外にエージェントが居るかどうか探し，もしいた時そのエージェントを再発生させるというようにしている．

最後に⑤番について．ここでは各変数に記憶された結果をコンソール画面に表示する．

また，ユーザーが初期に設定するものとしてまず

4.3.　各変数

ここでは各プログラムの変数について説明する．

4.3.1. Universe 内の変数

Universe はすべてのエージェントに対して使う変数やプログラムコードを記憶しておく場所である．そこに配置した変数について説明する．

・StandardValue，NormalValue，SmartValue

この変数自体はなにも記憶していないが，コントロールパネルでこの値を操作できるようになっている．そして，それぞれのエージェントの数はこの値を参照していくつ作成するかを決めている．

・NormalAgtVal，SmartAgtVal

再発生したそれぞれのエージェント数を記憶する．これと NormalValue 等と合計することで発生したエージェントの総数を出すことが出来る．

・NormalFinishVal，SmartFinishVal

ゴールに到達したそれぞれのエージェント数を記憶する．これと発生したエージェントの総数を使用することで未到達のエージェントを導き出すことが出来る．

・CountNomalCol，CountSmartCol

それぞれのエージェントが衝突した回数を記憶する．

・crashValueOfStartNo，crashValueOfStartSm

それぞれのエージェントがシミュレーションの開始時に衝突した回数を記憶する．この変数と CountNormalCol と CountSmartCol を使用して移動時の衝突回数を導き出す．

・startTime，nowTime，quickTime，endTime，Quick

それぞれ開始時間，今の時間，エージェントの移動が上昇する時間，終わった時間を記憶する．Quick は速度を上昇させるか否かを記憶する．

・result

結果を文字列で記憶する．

4.3.2. それぞれのエージェント内に設定した変数

一般エージェントと歩きスマホエージェント内に設定した変数は共通なので．両方同時に説明する．

・ID，X，Y，Layer，Direction

エージェントを設定した時に自動で生成される変数．この中で使っている変数は X，Y，の座標と方向を記憶する Direction のみである．

・gA，gAX，gAY

ゴールの座標エージェントを記憶する．

・Speed，DefaultSp

速度を記憶する．Speed は現在のスピードを記憶し，初期のスピードを DefaultSp に記憶する．

・ViewArea

個々の視界の範囲を記憶する．

・crashing

衝突中かどうか，また Int 型にして衝突が何段階目かを記憶する．

5. 実行環境・結果

実行環境は

OS：Windows 7 Enterprise

CPU：Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz 2.50Ghz

メモリ ：6.00GB

OS：64Bit 版

ソフトは aritsoc academic 3.0(x64)を使用する.

シミュレーションを 10 回行い，その総数を数字に使用する．実行結果は以下のとおり.

表の「生成総数」は 10 回シミュレーションを行った結果、それぞれのエージェントが生成された総合の数,「到達率」はそれぞれのエージェントの総数、つまり生成総数を母数として到達したエージェントの割合,「衝突率」は生成総数を母数として衝突した件数となっている.

表 1 実行結果

一般とスマホの割合 (一般：スマホ，人)	生成総数 (一般・スマホ，人)	到達率 (一般・スマホ，%)	生成数に対する衝突率 (一般・スマホ，%)
0：500	0・16760	0・95.2	0・2.98
100：400	5782・　12459	99.9・94.9	2.61・3.03
200：300	11291・9187	99.8・95.8	2.78・2.85
300：200	15003・5588	99.7・94.4	2.97・2.68
400：100	19029・2682	99.8・94.0	3.30・2.23
500：0	22015・0	99.6・0	2.98・0

到達率，衝突率に関しては表示圏外の数字については切り捨ての処理を行っている.

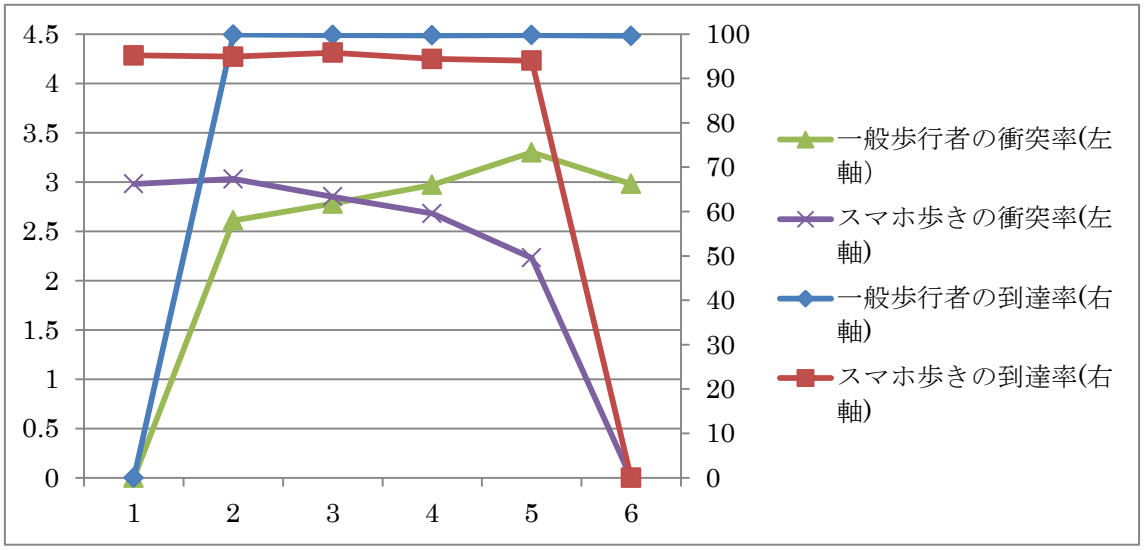

4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
一般歩行者の衝突率(左軸)
スマホ歩きの衝突率(左軸)
一般歩行者の到達率(右軸)
スマホ歩きの到達率(右軸)

図 5　結果のグラフ

6. 考察

まず明確に異なる部分として，時間内での到達率である．一般の歩行者はほぼ全員が時間内に到達しているが，スマホ歩きをしている者に関しては小さいが無視できない程度の数字の人数が到達できていない．この様子を見る限り，一般歩行者のものに関してはゴールの交差点近くにいるだろうことも容易に推測出来る．プログラムの稚拙さが読んだ誤差か，また現実でも厳密に見ると着いていないと判定できる人も居る事(後 1 歩で目的地の歩道に着く)を考えると，ほぼ現実に則したものとなっていると考えてもいいかもしれない．スマホ歩きをしている者の到達率については，現実ではスマホ歩きをしている者も一般歩行者と同じようにほぼ全員目的地に着いていると考えると，多少の誤差はある．ただ、信号が変わりそうになっていても終始スマホ歩きをしている，自分の移動速度の事を考えずにギリギリで交差点に進入している，等の事を考えると，もし終始歩きスマホをし，あまり周囲を見ていない者の場合，こういう結果になってもおかしくはないとも考えられる．両者の比率の変化による影響は，比率が変化してもそこまで大きな差が生まれると言うわけではないが，徐々にスマホの到達率が下がっているようにも見える．誤差の範囲内とも読み取れるが，確実に影響がないともいえない．

次に衝突率は，両者とも比率の変化で決して小さくない差が生まれている．母数が多いのも関係しているのではないかと考えられるが，人数が増える度に件数も増えていることも考えると，人数の増加以上のペースで件数が増加していることが考えられる．また，一般の歩行者の衝突率がスマホ歩きの歩行者に比べ高い理由は，一般の歩行者の移動速度が，スマホ歩きをしている歩行者より速いためではないかと考えられる．また，スマホ歩きをしている者でも一般の歩行者であっても共通して言えることは，「自身のエージェントの種類の数が相手のエージェントの種類の数より大きい場合，その差が大きければ大きいほど自身の衝突率が上昇する」、「そして相手のエージェントの数が 0 になると衝突率はまた下がる」ということである．この事から考えると，「エージェントの種類が混じってしまう時は，下手に数人交じるよりスマホ歩きの人数と一般の歩行者の人数の差が少なくなるように大人数混じったほうが衝突率は減少する」という条件が考えられる．

7. まとめ

本研究は，現実に近づけた状態でのスマホ歩きのシミュレーションを目指したが，いくつか実装したい条件を達成できずに終わった．ただ，シミュレーションの結果を見ると評価としては自分では満足の行くものが出来上がったと感じる．実装できなかった条件の影響で細かい部分まで再現出来なかったのが悔やまれる部分である．ペアの実装は当初から難しいだろうと予想していたので実装出来なかったことについてまだ後悔等はないが，スマホ歩きと一般の歩行者の状態を時間でランダムに遷移するエージェントの種類が作れなかったことが一番の心残りである．だが，そういったことを鑑みても元のシミュレーション実験と比較すると現実に則したものになり，より正確なシミュレーションになっていると感じる．

実行結果については，概ねこちらの思った通りの物となった．ただ予想外だったのが，衝突率に関係した事である．衝突率について，エージェントの数の差が大きくなるにつれて，数が大きい方のエージェントの衝突率が，他の比率から予想できる数字より大きく上昇するという結果が得られた．この結果から考えると，スマホ歩きがもし混じってしまうとしたら下手に少ない人数が交じることより一般の歩行者と同じくらいの人数が交じる方がいいということになる．ただ，今回の研究の比率では設定した条件が荒かったため，この条件をより現実に則したものにして，シミュレーションを正確にし，数をより詳しく検証していきたい．

8. 謝辞

本研究を進めるにあたり、ご指導を頂いた小池誠彦教授に感謝致します。また、日常の議論を通じて多くの知識や示唆を頂いた研究室の皆様に感謝します。

文　献

[1] 携帯電話のマナー／お知らせ，https://www.nttdocomo.co.jp/info/manner/

[2] 吉村 忍, 西川 紘史, 守安 智,”知的マルチエージェント交通流シミュレータ MATES の開発”，日本シミュレーション学会，23(3), 228-237, 2004　9 月

[3] Dirk Helbing，Illés Farkas，and Tamás Vicsek ，“Simulating dynamical features of escape panic” Nature，407，pp.487－490，September, 2000.

[4] MAS コミュニティ，http://mas.kke.co.jp/.

付録
実装したコードを載せる.
Universe

```
Univ_Init{
valueInit()
setStandard()
setAgent()
setStartAndGoal()
}

sub valueInit(){
	Universe.CountNormalCol = 0
	Universe.CountSmartCol = 0
	Universe.startTime = GetRealTime()
	Universe.quickTime = Universe.StartTime + 35000
	Universe.endTime = Universe.StartTime + 45000
	Universe.NormalAgtVal = 0
	Universe.SmartAgtVal = 0
	Universe.NormalFinishVal = 0
	Universe.SmartFinishVal = 0
	Universe.crashValueOfStartNo = 0
	Universe.crashValueOfStartSm = 0
}

sub setStandard(){
	Dim i As Integer
	if Universe.StandardValue != 0 Then
		CreateAgtMulti(Universe.Crossing.Standard, Universe.StandardValue)
	End if

	For i = 0 to Universe.StandardValue -1
		Universe.Crossing.Standard(i).X = Universe.Crossing.defaultstan(i).X
		Universe.Crossing.Standard(i).Y = Universe.Crossing.defaultstan(i).Y
	Next i
}

sub setAgent(){
	Dim i As Integer
	if Universe.NormalValue != 0 Then
		CreateAgtMulti(Universe.Crossing.Normal, Universe.NormalValue)
	End if
	if Universe.SmartValue != 0 Then
		CreateAgtMulti(Universe.Crossing.SmartPhone, Universe.SmartValue)
	End if
}

sub setStartAndGoal (){//ゴール地点の設定
```

```
	Dim i As Integer
	Dim iA As Integer
	Dim iAng As Double//角度
	Dim iRad As Integer//半径

	For i = 0 to CountAgt(Universe.Crossing.Normal) -1
		iA = Int(Rnd() * Universe.StandardValue)//エージェント人に対して
		Universe.Crossing.Normal(i).gA = iA//ゴール地点のエージェント
		Universe.Crossing.Normal(i).gAX = Universe.Crossing.Standard(iA).X//ゴール地点のエージェントの X 軸
		Universe.Crossing.Normal(i).gAY = Universe.Crossing.Standard(iA).Y//ゴール地点のエージェントの Y 軸
		Do While iA == Universe.Crossing.Normal(i).gA//同じエージェントにならないように
			iA = Int(Rnd() * CountAgt(Universe.Crossing.Standard))
		Loop
		iAng = DegreeToRad(Int(Rnd() * 360))//中心となるエージェントに対しての角度
		iRad = Int(Rnd() * 30)//どれだけ離れているか
		Universe.Crossing.Normal(i).X = Universe.Crossing.Standard(iA).X + (iRad * Cos (iAng))
		Universe.Crossing.Normal(i).Y = Universe.Crossing.Standard(iA).Y + (iRad * Sin(iAng))
	Next i

	For i = 0 to CountAgt(Universe.Crossing.SmartPhone) -1
		iA = Int(Rnd() * CountAgt(Universe.Crossing.Standard))//エージェントスマホに対して
		Universe.Crossing.SmartPhone(i).gA = iA
		Universe.Crossing.SmartPhone(i).gAX = Universe.Crossing.Standard(iA).X
		Universe.Crossing.SmartPhone(i).gAY = Universe.Crossing.Standard(iA).Y
		Do While iA == Universe.Crossing.SmartPhone(i).gA
			iA = Int(Rnd() * CountAgt(Universe.Crossing.Standard))
		Loop
		iAng = DegreeToRad(Int(Rnd() * 360))
		iRad = Int(Rnd() * 30)
		Universe.Crossing.SmartPhone(i).X = Universe.Crossing.Standard(iA).X + (iRad * Cos(iAng))
		Universe.Crossing.SmartPhone(i).Y = Universe.Crossing.Standard(iA).Y + (iRad * Sin(iAng))
	Next i
}

Univ_Step_Begin{
	if Universe.crashValueOfStartNo == 0 And Universe.crashValueOfStartSm == 0 Then
		Universe.crashValueOfStartNo = Universe.CountNormalCol
		Universe.crashValueOfStartSm = Universe.CountSmartCol
	End if
}
```

```
Univ_Step_End{
	Universe.nowTime = GetRealTime()
	If Universe.nowTime >= Universe.endTime Then
		ExitSimulation()
	ElseIf Universe.nowTime >= Universe.endTime - 5000 Then
		Universe.Quick = true
	ElseIf Universe.nowTime >= Universe.quickTime Then
		Universe.Quick = true
		resetMain()
	Else
		resetMain()
	End If
}

sub resetMain(){
	Dim i As Integer
	Dim newA As agt
	For i = 0 To CountAgt(Universe.Crossing.Normal) -1
		if Universe.Crossing.Normal(i).X > 200    Then
			reset(Universe.Crossing.Normal(i))
			Universe.Crossing.Normal(i).Speed =( Int(Rnd() * 3)    + 1 ) * 0.025
			Universe.NormalAgtVal = Universe.NormalAgtVal + 1
		End If
	Next i
	For i = 0 To CountAgt(Universe.Crossing.SmartPhone) -1
		if Universe.Crossing.SmartPhone(i).X > 200 Then
			reset(Universe.Crossing.SmartPhone(i))
			Universe.Crossing.SmartPhone(i).Speed = (Int(Rnd() * 3) + 1) * 0.01
			Universe.SmartAgtVal = Universe.SmartAgtVal + 1
		End If
	Next i

}

sub reset(iAgt As Agt){
	iAgt.gA = Int(Rnd() * Universe.StandardValue)
	iAgt.gAX = Universe.Crossing.Standard(iAgt.gA).X
	iAgt.gAY = Universe.Crossing.Standard(iAgt.gA).Y

	Dim sA As Integer
	sA = Int(Rnd() * Universe.StandardValue)
	Do While sA == iAgt.gA
		sA = Int(Rnd() * Universe.StandardValue)
	Loop

	Dim iAng As Double
	Dim iRad As Integer
```

```
        iAng = DegreeToRad(Int(Rnd() * 360))//中心となるエージェントに対しての角度
        iRad = Int(Rnd() * 30)//どれだけ離れているか

        iAgt.X = Universe.Crossing.Standard(sA).X + (iRad * Cos(iAng))
        iAgt.Y = Universe.Crossing.Standard(sA).Y + (iRad * Sin(iAng))
        iAgt.Direction = GetDirection(iAgt.X, iAgt.Y, iAgt.gAX, iAgt.gAY, Universe.Crossing)

}



Univ_Finish{
        Dim AllValue As Integer
        Dim startAll As Integer
        Dim result As String
        AllValue = Universe.NormalValue + Universe.SmartValue
        startAll = AllValue
        Print("初期エージェント数　Normal：" & Universe.NormalValue　)
        Print("｜Smart：" & Universe.SmartValue　)
        Println("｜初期エージェント総数："& startAll )
        result = Universe.NormalValue &","& Universe.SmartValue　&","& startAll　&","

        AllValue = Universe.CountNormalCol + Universe.CountSmartCol
        Print("衝突回数：" & AllValue )
        Print("｜初期衝突回数　Normal：" & Universe.crashValueOfStartNo )
        Println("｜Smart：" & Universe.crashValueOfStartSm )
        Print("移動時の衝突回数　Normal：" & Universe.CountNormalCol - Universe.crashValueOfStartNo )
        Print("｜Smart：" & Universe.CountSmartCol - Universe.crashValueOfStartSm )
        Println("｜移動時の総衝突回数：" & AllValue - Universe.crashValueOfStartNo - Universe.crashValueOfStartSm)
        result = result & AllValue &","& Universe.crashValueOfStartNo &","& Universe.crashValueOfStartSm &","&Universe.CountNormalCol - Universe.crashValueOfStartNo &"," & Universe.CountSmartCol - Universe.crashValueOfStartSm &"," &AllValue - Universe.crashValueOfStartNo - Universe.crashValueOfStartSm &","
        Println("")

        AllValue = Universe.NormalAgtVal + Universe.SmartAgtVal
        Print("再発生エージェント数　Normal：" & Universe.NormalAgtVal)
        Println("｜Smart：" & Universe.SmartAgtVal)
        Print("再発生エージェント総数：" & AllValue)
        Println("｜総発生エージェント数："& startAll + AllValue)
        result = result & Universe.NormalAgtVal &","&Universe.SmartAgtVal &","&AllValue &","&　startAll + AllValue &","
        Println("")

        AllValue = Universe.NormalFinishVal + Universe.SmartFinishVal
        Print("到達エージェント数　Normal：" & Universe.NormalFinishVal　)
        Print("｜Smart：" & Universe.SmartFinishVal )
```

```
	Println("｜到達エージェント総数：" & AllValue )
	result = result & Universe.NormalFinishVal &","& Universe.SmartFinishVal &","& AllValue &","
	Println("")

	Print("未到達エージェント数　Normal：" & Universe.NormalValue + Universe.NormalAgtVal - Universe.NormalFinishVal )
	Print("｜Smart：" & Universe.SmartValue + Universe.SmartAgtVal - Universe.SmartFinishVal)
	Println("｜未到達エージェント総数："& Universe.NormalValue + Universe.NormalAgtVal - Universe.NormalFinishVal + Universe.SmartValue + Universe.SmartAgtVal - Universe.SmartFinishVal)
	Println("ステップ数："&GetCountStep())
	result = result & Universe.NormalValue + Universe.NormalAgtVal - Universe.NormalFinishVal &","& Universe.SmartValue + Universe.SmartAgtVal - Universe.SmartFinishVal &","& Universe.NormalValue + Universe.NormalAgtVal - Universe.NormalFinishVal + Universe.SmartValue + Universe.SmartAgtVal - Universe.SmartFinishVal

//	Println(result)


	}
```

Normal エージェント

```
Agt_Init{
My.Speed =( Int(Rnd() * 3)  + 1 ) * 0.1 + 0.3
My.DefaultSp = My.Speed
My.ViewArea = 3
My.ValueSp = 0

}

Agt_Step{
	My.Direction = GetDirection(My.X, My.Y, My.gAX, My.gAY, Universe.Crossing)
	My.Speed = My.DefaultSp

	if Universe.quick == true Then
		My.ValueSp = 0.3
	Else
		My.ValueSp = 0
	End if
		If My.X >=0 And My.X <= GetWidthSpaceOwn() And My.X >=0 And My.X <= GetHeightSpaceOwn() Then
		//エリア範囲内なら行動する
		moveMain()
	End If
}

sub moveMain(){
```

```
Dim memoryStandard As AgtSet
Dim ViewAgents As AgtSet
Dim obj As Agt
Dim obj2 As Agt

Dim thr As Integer
thr = 0//デフォルト

Dim countSur As Integer
countSur = 0

MakeAllAgtSetAroundOwn(ViewAgents, My.ViewArea, False)
//周囲のエージェント取り出し
MakeOneAgtSetAroundOwn(memoryStandard, 15, Universe.Crossing.defaultstan, False)
//周囲の基準エージェント取り出し
DelAgtSet(ViewAgents, memoryStandard)
//基準エージェント取り除き

For Each obj In memoryStandard
        if obj.X == My.gAX And obj.Y == My.gAY Then
                thr = 1
                break
        End If
Next obj

if thr != 1 Then
        For Each obj In ViewAgents
                if Abs(MeasureDistance(obj.X, obj.Y, My.X, My.Y,
Universe.Crossing)) < 0.01 Then
                        thr = 2
                        break
                Else
                        countSur = countSur + 1
                End If
        Next obj
End if

ClearAgtSet(ViewAgents)

If thr == 2 Then//衝突時
        crash()
Elseif My.crashing != 0  Then
        nowcrashing()//衝突中かどうか
Elseif thr == 1 Then//普通に動く
        My.X = GetWidthSpaceOwn() + GetWidthSpaceOwn()
        My.Y = GetHeightSpaceOwn() + GetHeightSpaceOwn()
        Universe.NormalFinishVal = Universe.NormalFinishVal + 1
ElseIf thr == 0 Then//目標についた時表示圏外に飛んで消える
```

```
		if countSur > 0 Then//もし視界に誰かいる時
			moving(CountSur)
		Else//視界にいない時
			Pursue(Universe.Crossing.Standard(My.gA), My.Speed + My.ValueSp)
		End If
	End if
}


}

sub moving(AgtVal As Integer){
	Dim exDir As Double
	Dim memoX As Double
	Dim memoY As Double
	Dim memoryDis As Double
	Dim memoDir As Double

	Dim SurroundingAgents As AgtSet
	SurroundingAgents = removeSta()

	memoX = Cos(My.Direction) * (My.Speed + My.ValueSp)
	memoY = Sin(My.Direction) * (My.Speed + My.ValueSp)


	For Each obj In SurroundingAgents
		If obj <> My Then
			memoryDis = MeasureDistance(obj.X, obj.Y, My.X, My.Y, Universe.Crossing)
			exDir = GetDirection(obj.X, obj.Y, My.X, My.Y, Universe.Crossing)
			memoX = memoX + (My.Speed + My.ValueSp) * Cos(exDir) / memoryDis
			memoY = memoY + (My.Speed + My.ValueSp) * Sin(exDir) / memoryDis
		End If
	Next obj

	memoDir = RadToDegree(Atn(memoY / memoX))
	if memoX != Abs(memoX) Then
		memoDir =  memoDir + 180
	End If
	if AgtVal == 1 Then
		if Abs(memoDir-My.Direction) < 10 Or Abs(memoDir-My.Direction - 180) < 10 Then
			My.Direction = memoDir + 20
		End if
	End if
	My.Direction = memoDir
```

```
	My.Speed = Sqr(memoX * memoX + memoY * memoY)

	Forward(My.Speed + My.ValueSp)
}

sub crash(){
	Universe.CountNormalCol = Universe.CountNormalCol + 1
	My.Direction = My.Direction + 180
	Forward(My.Speed + My.ValueSp)
	My.crashing = 1
}

sub nowCrashing(){
		Dim SurroundingAgents As AgtSet
		Dim memoryStandard As AgtSet
		Dim obj As Agt

		Dim    memoryDis As Double
		Dim    exdir As Integer
		Dim memoX As Integer

		memoX = 0

	if My.crashing == 1 Then
		SurroundingAgents = removeSta()

		For Each obj In SurroundingAgents
			If obj <> My Then
				memoryDis = MeasureDistance(obj.X, obj.Y, My.X, My.Y,
Universe.Crossing)
				exDir = GetDirection(obj.X, obj.Y, My.X, My.Y,
Universe.Crossing)
				memoX = memoX + memoryDis *
Cos(DegreeToRad(My.Direction + exDir))
			End If
		Next obj

		If memoX >= 0 Then
			My.Direction = My.Direction + 45
		Else
			My.Direction = My.Direction - 45
		End if
	End if
	My.crashing = My.crashing + 1
	if My.crashing >= 4 Then
		My.crashing = 0
	end if
	Forward(My.Speed + My.ValueSp)
}
```

```
Function removeSta() As Agtset{//基準を取り除く
        Dim ViewAgents As AgtSet
        Dim memoryStandard As AgtSet
        Dim obj As Agt

        MakeAllAgtSetAroundOwn(ViewAgents, My.ViewArea, False)//周囲の視界内エージェント取り出し
        MakeAgtSet(memoryStandard, Universe.Crossing.defaultstan)
        //自分の周囲にある基準エージェント取り出し
        DelAgtSet(ViewAgents, memoryStandard)//基準エージェント取り除き

        return    ViewAgents
}
```

SmartPhone エージェント

```
Agt_Init{
My.Speed = (Int(Rnd() * 3) + 1) * 0.1
My.DefaultSp = My.Speed
My.ViewArea = 0.15
My.ValueSp = 0

}

Agt_Step{
        My.Direction = GetDirection(My.X, My.Y, My.gAX, My.gAY, Universe.Crossing)
        My.Speed = My.DefaultSp

        if Universe.quick == true Then
                My.ValueSp = 0.3
        Else
                My.ValueSp = 0
        End if
        If My.X >=0 And My.X <= GetWidthSpaceOwn() And My.X >=0 And My.X <= GetHeightSpaceOwn() Then
                //エリア範囲内なら行動する
                moveMain()
        End If
}

sub moveMain(){
        Dim memoryStandard As AgtSet
        Dim ViewAgents As AgtSet
        Dim obj As Agt
        Dim obj2 As Agt

        Dim thr As Integer
        thr = 0//デフォルト
```

```
Dim countSur As Integer
countSur = 0

MakeAllAgtSetAroundOwn(ViewAgents, My.ViewArea, False)
//周囲のエージェント取り出し
MakeOneAgtSetAroundOwn(memoryStandard, 15, Universe.Crossing.defaultstan, False)
//周囲の基準エージェント取り出し
DelAgtSet(ViewAgents, memoryStandard)
//基準エージェント取り除き

//もし範囲内の相手が自分の目標としている地点の時消えるフラグ立て
For Each obj In memoryStandard
	if obj.X == My.gAX And obj.Y == My.gAY Then
		thr = 1
		break
	End If
Next obj

if thr != 1 Then
	For Each obj In ViewAgents
		if Abs(MeasureDistance(obj.X, obj.Y, My.X, My.Y, Universe.Crossing)) < 0.01 Then
			thr = 2
			break
		Else
			countSur = countSur + 1
		End If
	Next obj
End if

ClearAgtSet(ViewAgents)

If thr == 2 Then//衝突時
	crash()
Elseif My.crashing != 0  Then
	nowcrashing()//衝突中かどうか
Elseif thr == 1 Then//普通に動く
	My.X = GetWidthSpaceOwn() + GetWidthSpaceOwn()
	My.Y = GetHeightSpaceOwn() + GetHeightSpaceOwn()
	Universe.SmartFinishVal = Universe.SmartFinishVal + 1
ElseIf thr == 0 Then//目標についた時表示圏外に飛んで消える
	if countSur > 0 Then//もし視界に誰かいる時
		moving(CountSur)
	Else//視界にいない時
		Pursue(Universe.Crossing.Standard(My.gA), My.Speed + My.ValueSp)
	End If
End if
```

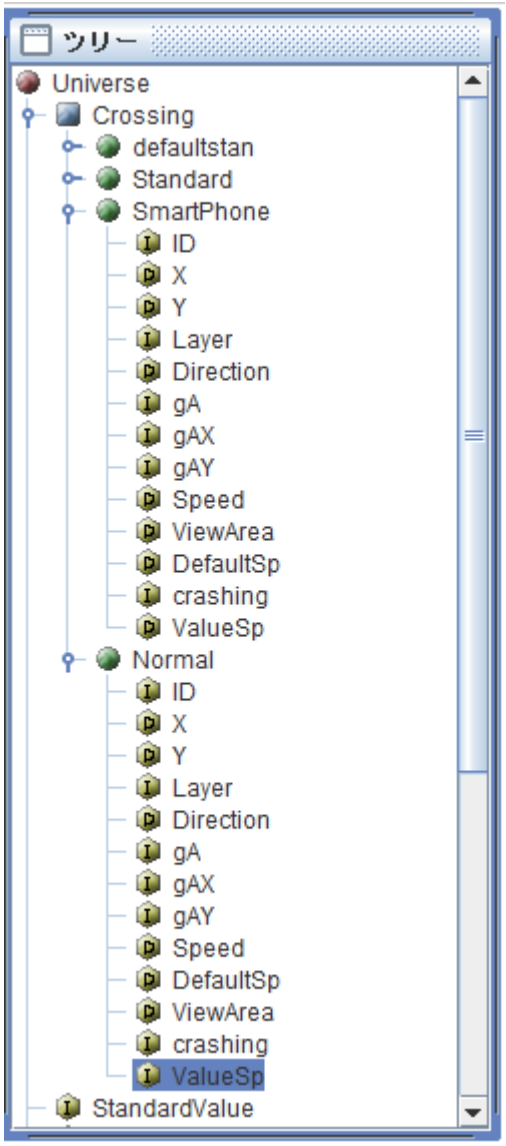
```
}

sub moving(AgtVal As Integer){
	Dim exDir As Double
	Dim memoX As Double
	Dim memoY As Double
	Dim memoryDis As Double
	Dim memoDir As Double

	Dim SurroundingAgents As AgtSet
	SurroundingAgents = removeSta()

	memoX = Cos(My.Direction) * (My.Speed + My.ValueSp)
	memoY = Sin(My.Direction) * (My.Speed + My.ValueSp)

	For Each obj In SurroundingAgents
		If obj <> My Then
			memoryDis = MeasureDistance(obj.X, obj.Y, My.X, My.Y,
Universe.Crossing)
			exDir = GetDirection(obj.X, obj.Y, My.X, My.Y, Universe.Crossing)
			memoX = memoX + (My.Speed + My.ValueSp) * Cos(exDir) /
memoryDis
			memoY = memoY + (My.Speed + My.ValueSp) * Sin(exDir) /
memoryDis
		End If
	Next obj

	memoDir = RadToDegree(Atn(memoY / memoX))

		if memoX != Abs(memoX) Then
		memoDir =    memoDir + 180
	End If
	if AgtVal == 1 Then
		if Abs(memoDir-My.Direction) < 5 Or Abs(memoDir-My.Direction - 180) < 5
Then
			My.Direction = memoDir + 20
		End if
	End if
	My.Direction = memoDir
	My.Speed = Sqr(memoX * memoX + memoY * memoY)

	Forward(My.Speed + My.ValueSp )
}

sub crash(){
	Universe.CountSmartCol = Universe.CountSmartCol + 1
	My.Direction = My.Direction + 180
	Forward(My.Speed + My.ValueSp)
	My.crashing = 1
```

```
}

sub nowCrashing(){
		Dim SurroundingAgents As AgtSet
		Dim memoryStandard As AgtSet
		Dim obj As Agt

		Dim	memoryDis As Double
		Dim	exdir As Integer
		Dim memoX As Integer

		memoX = 0

	if My.crashing == 1 Then
		SurroundingAgents = removeSta()

		For Each obj In SurroundingAgents
			If obj <> My Then
				memoryDis = MeasureDistance(obj.X, obj.Y, My.X, My.Y,
Universe.Crossing)
				exDir = GetDirection(obj.X, obj.Y, My.X, My.Y,
Universe.Crossing)
				memoX = memoX + memoryDis *
Cos(DegreeToRad(My.Direction + exDir))
			End If
		Next obj

		If memoX >= 0 Then
			My.Direction = My.Direction + 45

		Else
			My.Direction = My.Direction - 45
		End if
	End if
	My.crashing = My.crashing + 1
	if My.crashing >= 4 Then
		My.crashing = 0
	end if
	Forward(My.Speed + My.ValueSp)
}

Function removeSta() As Agtset{//基準を取り除く
	Dim ViewAgents As AgtSet
	Dim memoryStandard As AgtSet
	Dim obj As Agt

	MakeAllAgtSetAroundOwn(ViewAgents, My.ViewArea, False)//周囲の視界内エージェ
ント取り出し
	MakeAgtSet(memoryStandard, Universe.Crossing.defaultstan)
```

```
        //自分の周囲にある基準エージェント取り出し
        DelAgtSet(ViewAgents, memoryStandard)//基準エージェント取り除き

        return    ViewAgents
}
```

階層(Universe)

階層(エージェント)



また，defaultstan エージェントに交差点の基準の座標を覚えさせている．