名古屋文理大学紀要　第4号（2004）

オンラインゲームのための戦略立案支援ツールの開発

A Decision Support Software to Play Online Game "Kachohugetsu Tactics"

田近一郎，　柴田啓介*
Ichiro TAJIKA,　Keisuke SHIBATA

インターネット上での多人数参加型ゲームの一つに花鳥風月タクティクスという陣地取りゲームがある．各プレイヤーはモンスター隊を使って戦闘を繰り返し勝利を目指す．このゲームでは戦闘に際し確実に勝利できるモンスター隊を編成することがゲーム勝利の鍵となる．そこで，本研究ではモンスター隊の編成を入力すると戦闘過程をシミュレートし戦闘結果を予測する補助ツールを開発した．これによりゲームプレイヤーは実際の戦闘の前にこのツールを用いて戦闘に勝利できるモンスター隊を容易に編成することが可能となる．

キーワード：多人数参加型ゲーム，モデル化，シミュレーション，予測，アルゴリズム
Online Multiplayer Game, Modeling, Simulation, Prediction, Algorithm

1．はじめに

インターネット回線を使った多人数参加型のオンラインゲームの一つに花鳥風月タクティクスという陣地取りゲームがある．このゲームでは8人のプレイヤーが互いに隣国と領土をめぐって戦闘を繰り返し，最終的に4つの特別な領土を獲得したプレイヤーがゲームの勝者となる．戦闘では一つの領土をめぐり進攻側，守備側それぞれのプレイヤーがモンスター隊を編成して戦わせる．敵モンスター隊を壊滅させるか，領土を守りきったプレイヤーがその戦闘の勝者となる．このゲームのプレイの過程は次の通りである．

ゲームの開始時，8人のプレイヤーはそれぞれただ一つの領土と少数のモンスターを所有する．ゲームの初期には少ない手持ちモンスターを駆使して領土周辺で戦闘を繰り返し，領土拡張とモンスターの増強をはかる．この時点では自国の領土が他のプレイヤーの領土と接することはまれであり，ゲームサーバがシミュレートする弱小国との戦闘が主である．しかし，後述するゲームの中期～後期において様々な攻撃／守備の状況に対応できるよう，できるだけ多くのモンスターを獲得しておくことが重要である．戦闘もそれを主眼に進められる．

一方，ゲームの中期になると各プレイヤーの領土は広く，その国境線は長くなる．そのため複数のプレイヤーと領土を互いに接することが多くプレイヤー同士でおこなわれる戦闘が戦闘の大半を占める．この時点ではゲームの初期のように戦闘をとりあえずこなしてゆくだけでは生き残りは困難である．なぜなら，単に大量のモンスター隊をある地域への進攻にまわせば，長い国境線の守備が手薄になり隣国から容易に進攻されるし，逆に，ある地域への進攻のためのモンスター隊を出し惜しみすれば，敵国との戦闘でモンスター隊をすべて失いいたずらに貴重なモンスターを消耗してしまうことになるからである．従って，進攻に際して

* 平成15年度名古屋文理大学情報文化学部情報文化学科卒業生

は，モンスターの消耗を減らすべく，敵国の守備モンスター隊の質・量を見極め，必要最小限のモンスター隊を編成する必要がある．また同時に，領土が広いためモンスター隊の移動時間も考慮した上で，周辺国から進攻されないようにモンスター隊を領土内の要所の城に適切に配置するというモンスター隊の戦略的な配置計画を立てなければならない．

ゲームの後期から終盤にかけては生き残った少数のプレイヤーが演じる大国の間で大規模な戦闘が繰り返される．質・量ともに強力なモンスターを備え，守備の堅い城を多数領土内に持つプレイヤーがゲームを優位にすすめていける．

以上のように進攻と守備の絶妙なバランスを保ちつつ領土を拡大してゆくところがこのゲームの難しく同時に面白い部分である．プレイヤーがそのバランスを保ちつつゲームを思い通りに進めてゆくには繰り返される戦闘で確実に勝利してゆく必要がある．しかし，このゲームはオンラインゲームであるため，一度戦闘を開始すると後戻りができない．つまり，ある戦闘で誤った手を選ぶとモンスターを失うだけでなく，その後の戦況が悪化しゲームに負ける事態も生じかねない．また，ゲームの初期・中期・後期の進展状況により戦闘の様相はまったく異なる．このように様々な状況下での戦闘に際し，敵を攻撃／迎撃するための効果的なモンスター隊の編成をプレイヤーが自力でおこなうのはプレイヤーのゲームに関する経験が深まったとしてもかなり困難である．そこで，戦闘に際し編成したモンスター隊が敵に有効な打撃を与えるか否かを判断するための補助ツールがあればその戦闘での勝利，ひいてはゲーム全体の勝利に大きく貢献するであろう．言い換えると，実際の戦闘の前にそのツールを用いて戦闘をシミュレートできれば，ゲームを優位に進めることが可能となるであろう．

以上の理由から，本研究では，この目的を達成するツール「戦闘シミュレータ」プログラムを提案する(図1)．このプログラムは戦闘に参加する敵モンスター隊の編成と自軍モンスター隊の編成を入力すると，戦闘過程をシミュレートし，戦闘の勝敗を出力する．ユーザは，様々な編成の自軍モスター隊で戦闘を繰り返しシミュレートし，戦闘に勝てる必要最小限のモン

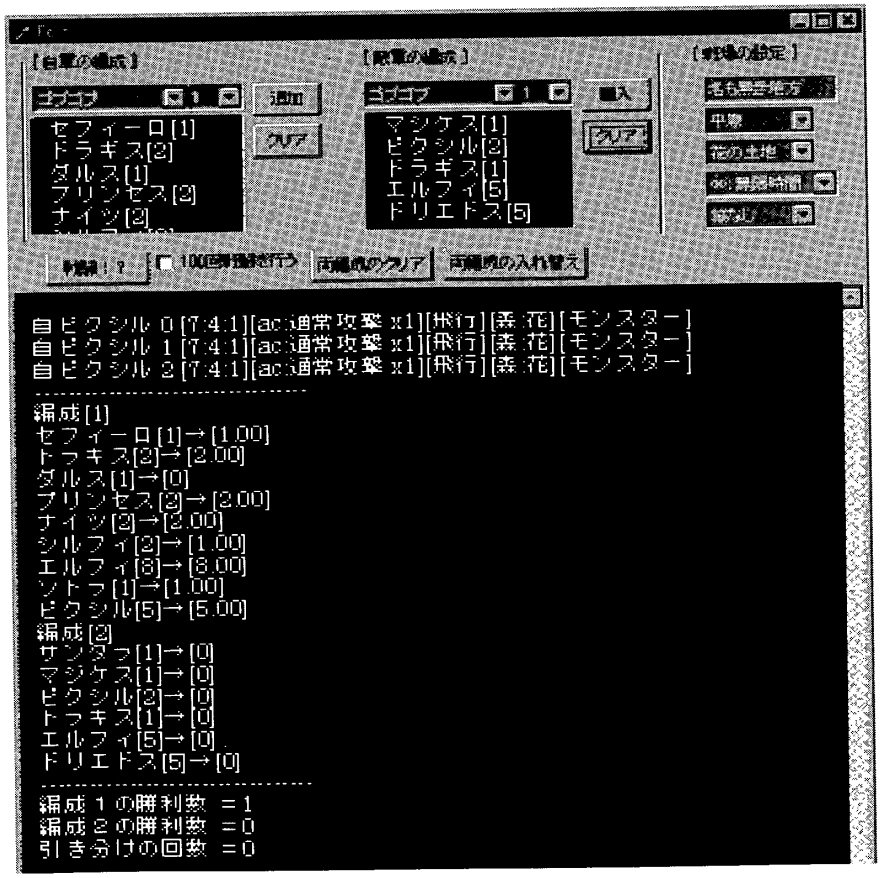

図1　戦闘シミュレータ

スター隊をすみやかに見つけることが可能となる．なお，このプログラムはボーランド社のフリーのソフトウェアパッケージである Delphi6 pesonal edition を用いて開発した．

本稿の構成は以下の通りである．2．では実際のゲームの戦闘の過程を説明し，シミュレーションの対象を明確にする．3．では戦闘シミュレータのGUI，データ構造およびアルゴリズムについて説明する．4．では戦闘シミュレータを用いた予備的な評価実験について述べる．5．ではまとめと今後の課題について述べる．

2．シミュレーションの対象としての戦闘

このゲームの戦闘は次のような過程で進められる．

(1) 進攻側プレイヤーが戦場となる領土を選択．領土は守備側プレイヤーが編成したモンスター隊で守備している．

(2) 進攻側プレイヤーは守備側モンスター隊の編成と領土の属性に関する情報をもとに進攻のためのモンスター隊を編成しゲームサーバに送信する．

(3) ゲームサーバは両モンスター隊の編成データをインターネットを介して受信すると領土の属性に応じて戦闘に参加する全モンスターの耐久力，攻撃力，防御力等のパラメータを微調整し，戦闘を開始する．

(4) ゲームサーバが戦闘をおこなう．その間プレイヤーは待機する．戦闘に参加する全モンスターがそれぞれ攻撃対象を選び攻撃することをターンと呼ぶ．ゲームサーバは戦闘中このターンを何度か繰り返す．

(5) ゲームサーバが戦闘の勝敗を決める．勝敗は次の3つの場合に分類される．

- 進攻側が守備側を壊滅させる場合，進攻側の勝利
- 守備側が進攻側を壊滅させる場合，守備側の勝利
- ターンを10回繰り返した時点で守備側が進攻側の攻撃に耐え抜いた場合，守備側の勝利

戦闘の過程のうちプレイヤーが手を下せるのは(1)(2)のみであり，残りの(3)(4)(5)はゲームサーバ側で処理をする．つまり，戦闘シミュレータの目標は(3)(4)(5)を適切な方法でシミュレートし，実際にゲームサーバで(3)(4)(5)を処理するときと同じ戦闘結果を出力することである．この目標を達成するため，ゲームサーバの動作をできる限りまねることを戦闘シミュレータ開発の方針とする．もし，戦闘シミュレータが，(3)(4)(5)を処理するためにゲームサーバが用いている方法を忠実に再現できているならば，ほとんどの場合でゲームサーバと同じ戦闘結果を出力し，予測精度を非常に高くできるだろう．(ゲームサーバが乱数を用いるアルゴリズムの場合，シミュレータも乱数を用いる適当なアルゴリズムとして構成すれば，あるデータで繰り返しシミュレータを動作させたときの戦闘結果の傾向は，同じデータで繰り返しゲームサーバを動作させたときの戦闘結果の傾向と一致するであろう．)

しかし，実際にはゲームサーバが(3)(4)(5)を処理するのに用いている具体的なアルゴリズムは未知であるので，ゲームサーバの動作から用いられているアルゴリズムを推測し，ゲームサーバの動作を再現（シミュレート）せざるを得ない．特に，シミュレーションの主な対象は戦闘の実質である(4)のターンであるから，ゲームサーバが用いているターン処理アルゴリズムの動作に似たふるまいをするターン処理アルゴリズムを構成することが予測精度向上の鍵となる．

実際のゲームで(4)の過程を観察した結果，ターンにおけるゲームサーバの処理内容は次の(a)(b)(c)(d)にまとめられる．

(a) 戦闘に参加するすべてのモンスターから順次，攻撃をおこなうモンスターをランダムに重複がないように選択すること，

(b) その際，攻撃対象となるモンスターは敵モンスター隊の生き残りからランダムに一体選択すること，

(c) 勝負はそれぞれのモンスターの耐久力，攻撃力，防御力等のパラメータで決まること，

(d) どちらかのモンスター隊が全滅する場合を除き，すべてのモンスターが攻撃を終えてはじめてそのターンは終了すること．

上記のうち(c)は各モンスターのパラメータ値，戦場の属性等のデータが与えられれば，比較的容易に計算できる．従って，(a)，(b)，(d)を満たすようなターン処理アルゴリズムを適当に構成すればよい．

3．戦闘シミュレータ

3．1．戦闘シミュレータのGUI

戦闘シミュレータは，進攻側モンスター隊・守備側モンスター隊の編成，戦場の属性等を入力データとして受け取ると，ターンを逐一シミュレートし最終的な戦闘結果を出力する．したがってそのGUIレイアウトは，

(a) モンスターの編成データ等，煩雑なデータ入力に煩わされない入力方法と入力データのわかりやすい

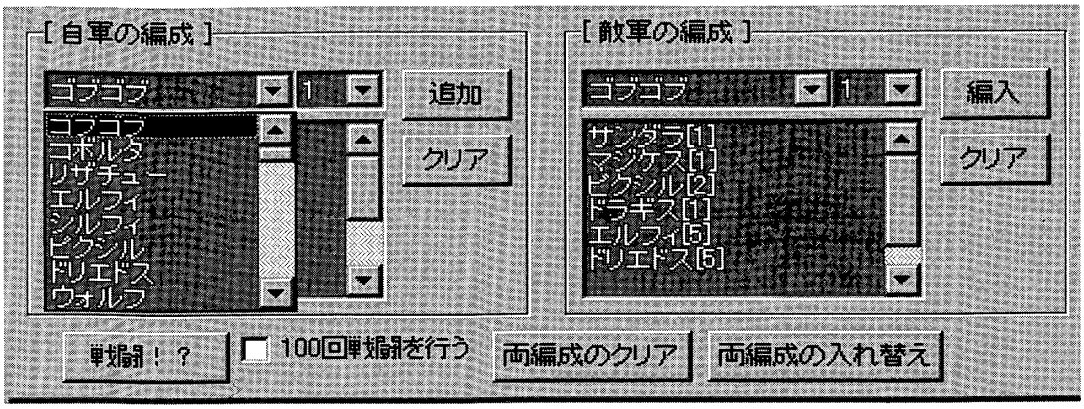

図2　モンスター隊編成用 GUI

表示，

(b)　戦闘開始に関するいくつかのオプション，

(c)　戦闘の途中経過の逐次表示，

(d)　最終的な戦闘結果の表示，

の4つを考慮して設計した．

(a)に関しては，進攻側モンスター隊・守備側モンスター隊の編成をすばやくできるように，プルダウンメニュー方式でモンスター一覧表から戦闘に参加するモンスターの種類と数を選ぶしくみとした(図2)．また，戦場の種類・属性など，戦闘結果の予測に必要な他のデータについてもプルダウンメニュー方式を用いた(図3)．これらのデータ入力インタフェースはウィンドウ上部に並べて配置した．その他，同じモンスター隊の編成でもそれが進攻側であるか守備側であるかにより戦闘開始時のパラメータ調整（2．の戦闘の過程(3)を参照）に違いが生じるので進攻側と守備側でモンスター隊の編成をそっくり入れ換えるためのボタンもつけた．

(b)に関しては，ウィンドウ中央に戦闘のシミュレーションを開始するためのボタンを配置した．また，自軍モンスター隊と敵モンスター隊の実力がほぼ同レベルの場合，同じモンスター隊の編成であってもシミュレーションで用いる乱数に依存して毎回の戦闘結果は変化しやすい．このように予測の信頼性を高くできない場合に備え，「100回戦闘を行う」チェックボックスを配置した．このチェックボックスを“ON”にすると同じモンスター編成で100回戦闘をシミュレートする．

(c)，(d)に関しては，ウィンドウ下部に戦闘の途中経過（モンスターの消耗状況）と最終的な戦闘結果を表示するためのテキストボックスを配置した．戦闘結果の表示と共に戦闘終了時点でのモンスターの消耗状況を一目で把握できるように戦闘開始時および終了時の各モンスターの数を並べて表示させた(図4)．なお，100回戦闘をシミュレートした場合には各シミュレー

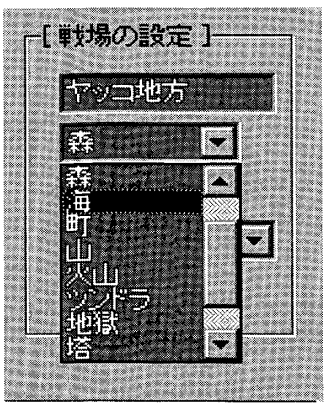

図3　戦場の属性指定用 GUI

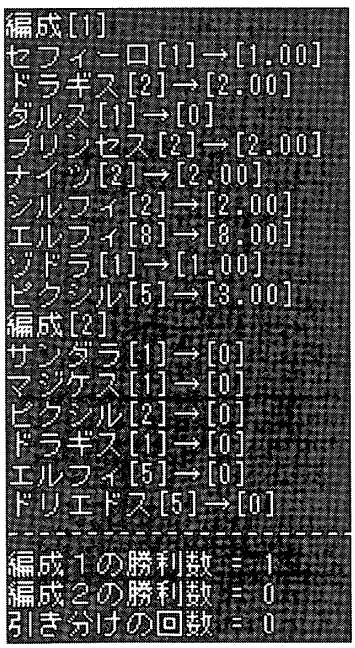

図4　戦闘終了時のモンスター残存数と戦闘結果

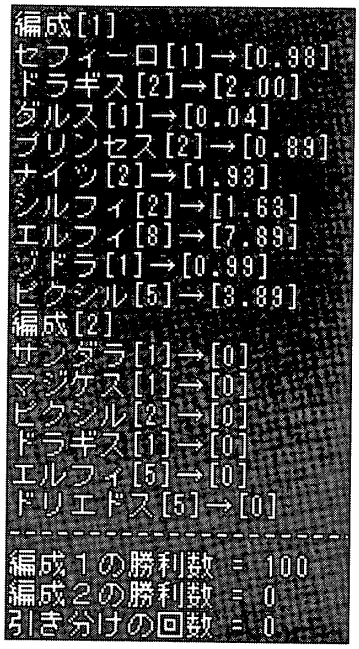

図５　100回の戦闘に関するモンスターの残存数の平均値

トの結果をすべて表示する代わりに勝利の回数，敗北の回数，モンスターの残存数を100回の戦闘に関して平均した値を表示し，モンスター編成に手を加えるときの参考となるようにした(図５)．

３．２．戦闘シミュレータのデータ構造

データを効率良く処理するプログラムを設計するにはそのデータが持つ性質に合わせて適当にデータ構造を決める必要がある．戦闘シミュレータの場合，扱わなければならないデータは主に戦闘に参加するモンスター一体一体のデータ，それも戦闘が進むにつれて時々刻々変化するデータである．また，モンスターの集団に対しておこなわなければならない操作は２．で議論したように主に次の２つである．

(a)　戦闘に参加するすべてのモンスターから順次，攻撃をおこなうモンスターをランダムに重複がないように選択すること，

(b)　その際，攻撃対象となるモンスターは敵モンスター隊の生き残りからランダムに一体選択すること．

これらの要求を満たすデータ構造として戦闘シミュレータではリストを採用した．ここで，リストとはデータを入れる箱（以下，リスト要素と呼ぶ）が複数個連結されたデータ構造のことである．リストを構成する各リスト要素が，それぞれ次に参照すべきリスト要素の場所（ポインタと呼ぶ）を保持することでリストの先頭から次々とリスト要素を参照することができる．リストの特長として

・新しいデータの挿入や不要なデータの削除が頻繁におこなわれ，データの個数の上限が決まらない場合でもデータ構造保持のための手間が少ない，

・複数のリストを連結して１つの長いリストを作ることができる．つまり，容易にデータのマージが可能，

が挙げられる．

戦闘に参加するモンスターデータはリストを用いて次のように対応づけられる．また，戦闘中のモンスターのさまざまなふるまいはリストに対する次のような操作で実現される．

＜モンスターデータとリストの対応＞

●モンスター一体の属性値，行動履歴

⇔リスト要素１つに格納されたデータ

●守備側モンスター隊

⇔守備側モンスター隊を表すリスト：LIST_defence

●進攻側モンスター隊

⇔進攻側モンスター隊を表すリスト：LIST_attack

＜戦闘中のモンスターのふるまいとそれに対応するリスト操作＞

●攻撃されたモンスターの属性値の変化

⇔対応するリスト要素のデータ更新

●戦闘不能のモンスターの戦場離脱

⇔対応するリスト要素を所属するモンスター隊を表すリストから削除

●各ターン時に戦闘に参加するすべてのモンスターから順次，攻撃をおこなわせるモンスターをランダムに重複なく選択

⇔各ターンの開始時に守備側リスト：LIST_defence，攻撃側リスト：LIST_attack の２つのリストを結合してシャッフルし，攻撃順序を表すリスト LIST_all を生成．攻撃するモンスターを一体一体選択する代わりにリスト LIST_all を先頭からたどる

ここで，どのターンでも生き残りモンスターだけが攻撃をおこなうことができ，戦闘不能モンスターは攻撃をおこなうことはできない．これを手間をかけずに実現するためモンスターが戦闘不能になった時点でそのモンスターに対応するリスト要素をただちに所属するモンスター隊（守備側か進攻側）のリストから削除し，以降のターンで攻撃順リストを守備側・攻撃側リストから生成するとき，そのモンスターが誤って攻撃順リストに加えられることを避けている．また同時に，所属するモンスター隊から戦闘不能モンスターを削除

することでそのモンスターを無意味に攻撃対象とすることも避けている．

3．3．戦闘シミュレータのアルゴリズム

戦闘シミュレータ(図6)のアルゴリズムは大きく分けて次の3つのブロックから成る．各ブロックの内容はそれぞれ次の通りである．

(I) 戦闘シミュレータは起動時に外部にあるモンスターデータベースファイルから各種モンスターの属性に関するデータを読み込み配列に格納する．配列要素一つが一種類のモンスターに関する全属性データを保持する．したがって，新種のモンスターを戦闘で使うにはあらかじめそのモンスターのデータをデータベースに追加登録しておく必要がある．なお，生成された配列は(II)で守備側・進攻側の両モンスター隊を表すリスト：LIST_defence，LIST_attackを作る際の基本データとして使用される．

(II) プレイヤーが入力した守備側・進攻側の両モンスター隊の編成をもとに守備側と進攻側のモンスター隊の編成を表すリスト LIST_defence，LIST_attack を生成する．リスト中の各リスト要素が戦闘に参加する一体のモンスターの属性値，行動履歴を保持する．これらのリストは戦闘中モンスターの属性値が変化するたびに繰り返し更新される．

(III) 各ターンの開始時にそのターンに参加するすべてのモンスターの攻撃順序を次のようにして決定する．まず，守備側・攻撃側モンスター隊のリストを結合し，次に出来たリストをシャッフルして新たなリスト LIST_all を作成する．従って，アルゴリズムはこのリストを先頭からたどると自動的に戦闘に参加するすべてのモンスターからランダムな順序で攻撃者を選択することになる．攻撃モンスターがきまれば，攻撃対象となるモンスターを敵側モンスター隊の中からランダムに一体選び攻撃する．攻撃を受けたモンスターの耐久力 HP から受けたダメージ（＝攻撃モンスターの攻撃力－攻撃されたモンスターの防御力）分を減らす．耐久力が0になるとそのモンスターは戦闘不能となる．一度戦闘不能になれば所属するモンスターリストから削除される．

以上のアルゴリズムの擬似プログラムを図7に示す．

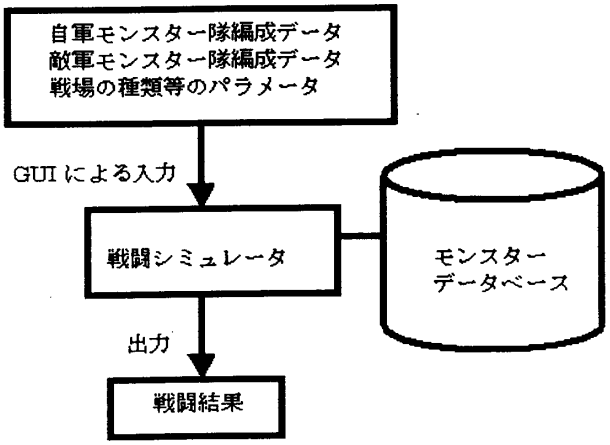

図6　戦闘シミュレータのシステム構成

4．予測性能についての予備的な評価実験

現時点では戦闘シミュレータの予測性能についての詳細な評価はおこなっていない．ここでは，複数の戦闘事例からわかったシミュレータの予測の傾向を列挙する．

- 両軍モンスター隊の編成から容易に戦闘結果の予測がつく場合，戦闘シミュレータも高い確率でゲームサーバと同じ戦闘結果を予測する．このような事例については戦闘シミュレータを使う必要はない．
- しかし，編成を見る限り勝てそうだが実際には負けたという戦闘事例があり，その事例では100回のシミュレーション中60回のシミュレーションで勝利した．つまり，シミュレータは見た目ほどには勝てる戦闘ではないという情報をユーザに与えることができている．
- 両軍モンスター隊を構成するモンスターの数が多い場合，戦闘結果の予測が難しい．このような事例について戦闘シミュレータはその威力を発揮する．
- 両軍の実力がほとんど互角でまったく戦闘結果の予測がつかない場合，戦闘シミュレータは100回のシミュレーション中50回前後のシミュレーションで勝利する．このような事例についてはゲームサーバも勝利か敗北をそれぞれほぼ2分の1の確率で出力すると思われ，予測は原理的に不可能である．

5．まとめと今後の課題

本稿では戦闘シミュレータを提案した．戦闘シミュレータの課題として以下の4つが挙げられる．

- シミュレータは「100回戦闘を行う」チェックボックスを“ON”にすると，同じモンスター編成で100回戦闘をシミュレートし，戦闘結果の一つとして両軍で生き残ったモンスター数の平均値を出力する．し

```
入力データ：進攻側モンスター隊の編成,
            守備側モンスター隊の編成,
            戦場となる領土の属性
出力        ：戦闘結果

(1)  全モンスター情報の配列を生成；
(2)  進攻側モンスター隊の連結リスト LIST_attack を生成；
(3)  守備側モンスター隊の連結リスト LIST_defence を生成；
(4)  領土の属性値から両軍モンスターの耐久力，攻撃力，防御力を微調整；
(5)  for TURN=1 to 10 do
       begin
(6)      LIST_attack，LIST_defence を結合し，できたリストをシャッフルし，
         攻撃順序を表す連結リスト LIST_all を生成；
(7)      for LIST_all の各モンスター MNST do
           begin
(8)          if 攻撃するモンスター MNST が LIST_attack に属する then
               begin
(9)              攻撃対象を LIST_defence からランダムに選択；
(10)             攻撃されたモンスターの耐久力を更新；
(11)             戦闘不能になればそのモンスターを LIST_defence から削除し，
                 同時に LIST_all のそのモンスターにスキップフラグを立てる；
               end
(12)         else // 攻撃するモンスター MNST が LIST_defence に属する //
               begin
(13)             攻撃対象を LIST_attack からランダムに選択；
(14)             攻撃されたモンスターの耐久力を更新；
(15)             戦闘不能になればそのモンスターを LIST_attack から削除し，
                 同時に LIST_all のそのモンスターにスキップフラグを立てる；
               end
           end;
(16)     if LIST_attack が空 then “守備側プレイヤーの勝利” を表示
         elseif LIST_defence が空 then “進攻側プレイヤーの勝利” を表示；
       end;
(17) “守備側プレイヤーの勝利” を表示；
```

図7　戦闘シミュレータのアルゴリズム

かし，同じ編成であるにもかかわらず戦闘によって生き残りモンスターの数が大きくゆらぐ場合があり，その場合残存数の平均値は意味をなさない．そこで，残存数の平均値だけでなくその標準偏差も同時に表示すれば，戦闘ごとのばらつきが明らかになりモンスター編成のための信頼できる情報として役立てることができるであろう．

・本プログラムではモンスター隊編成のためのデータ入力はプルダウンメニュー方式でおこなう．これにより入力の手間はかなり減るが，それでも戦闘に参加させるモンスターの種類が多ければかなり煩雑である．そこで例えば，常套手段的に組み合わせて使うことの多いモンスターの組み合わせなどは，あらかじめテンプレートとして用意しておくことが考えられる．

・本プログラムでは，ユーザは戦闘のシミュレーションに必要な全データを入力する必要がある．もし敵モンスター隊の編成，領土の属性等のデータを入力するだけで最適な味方モンスター隊の編成を自動生成するAI機能が実現できれば，ゲームの勝利に大きく貢献する．このような機能は，しらみつぶし的方法に基づくアルゴリズムにより原理的には実現可能である．しかし，この方法では結果を出力するのに要する時間がかかりすぎ，とうてい実用的なものとはならないと思われる．また，アルゴリズムを工夫するという観点からもこの方法は魅力がない．これ以外の方法を用いる効率の良いアルゴリズムの開発は今後挑戦すべき大きな問題である．

・戦闘シミュレータで用意できるモンスターの種類は外部のモンスターデータベースに依存する．新種のモンスターで戦闘をおこなうにはまずデータベース(DAT ファイル) を更新し，その後，あらためて戦闘シミュレータを起動する必要がある．戦闘シミュレータが内部にモンスターデータベースを (DAT ファイルの形で) 保持でき，モンスターデータの更新を戦闘シミュレータが用意した GUI を使ってできるならそのような手間は不要となるだろう．

参考文献

・森川幸人，マッチ箱の脳(AI)使える人工知能のお話，新紀元社，2000.
・オンラインゲーム「花鳥風月タクティクス」のサイト：http://nyan.algolab.co.jp/~tinyan/kacho/kacho.htm