1E3-4

シナリオ記述に基づくゲーミングシミュレーションの自動生成

Automatic Gaming Simulation Generation based on Scenario Description

鈴木格*1　　†角田啓介*1　　菱山玲子*1
Itaru SUZUKI　Keisuke TSUNODA　Reiko HISHIYAMA

早稲田大学大学院創造理工学研究科経営システム工学専攻
Graduate School of Creative Science and Engineering, Waseda University

Computer-aided Gaming Simulations has attracted attention as useful tool for“ learning by doing ”style education. Among them, games which people can participate in through the Internet become more important because of nowadays diffusion of computers. But it is difficult for non-programmers to make such games, and most of people who need to make such games — teachers, scholars and so on — are non-programmer.

In this study, a new framework is proposed. This framework enables non-programmers to make such games. This framework includes 2 parts: the domain specific language “Karina” and its translator. Karina is a language used to describe a game scenario. The translator translates the scenario into a CGI program. By sending the CGI program to a server anyone with an internet connection can play the game.

Using this framework, non-programmers can make games by themselves and these can be used for education and research.

1. はじめに

ゲーミングシミュレーション (以下 GS) は，従来の教室での講義という形式にとらわれず，現実に密着した体験による学習をするためのツールとして着目されている．特に計算機シミュレーションと情報通信技術を組み合わせた形の GS は，今日のコンピュータの普及も相まって，今後重要な教育ツールとなっていくことが考えられる．しかし，汎用プログラミング言語を用いて計算機シミュレーションを利用した GS を作成することは，非プログラマには困難である．

そこで本研究では，非プログラマの GS 作成を容易化するためのフレームワークを提案する．それは GS を記述するために専用の言語を使用し，その言語による記述を Web アプリケーションに自動変換するというフレームワークである．提案言語によって GS の記述が十分に可能であり，またそれが Web アプリケーションに容易に変換可能であることを，言語とトランスレータの実装を通して示す．

2. 関連研究

2.1 ゲーミングシミュレーションと学習

新井ら [1] によれば現在，学習者と環境および学習者同士のインタラクションに基づいて学習者が自らの知識と自己認識を新たにしていくという学習モデルが注目されており，そのためのフィールドとして GS を利用することが考えられている．

Greenblat[2] はそのような学習のための GS を実際に作成するための枠組みを提案している．それによれば，デザイナが設計すべきは環境の表現に現実を忠実に再現するシミュレーションではなく，学習したいテーマを強調するために主要な要素を抜き出して作ったシミュレーションであるとされており，学習者と環境とのインタラクションについても，ゲームを進める手続きとして形式的に定めるようにしている．

連絡先: 鈴木格，早稲田大学大学院創造理工学研究科，itaru-suzuki(at)asagi.waseda.jp
† 現在は日本電信電話株式会社に勤務

2.2 シミュレーションフレームワーク

ゲーミングシミュレーション作成のために利用することもできるマルチエージェントシミュレーションのためのフレームワークとして，Tanuma, Deguchi ら [3] の提案する SOARS がある．SOARS ではエージェントおよびフィールドについてその動作を記述し，それらが並行に動作を繰り返すことでシミュレーションが進行する．意思決定を人間にさせるためのインタフェースを作成し，ゲームとして利用することも可能である．しかし SOARS によって作成されたゲームはフレームワーク内でのみ動作し，単独で動作させることや拡張を想定してはいない．

また，シミュレーション記述を容易化する手法として Ishida[4] による Q 言語がある．これはエージェントの内部構造ではなくインタラクションを記述するための言語である．しかし Q 言語は，ゲームがコンピュータの専門家を含む複数のデザイナのコラボレーションによって作成されることを前提としているため，非プログラマがそれを利用することは必ずしも容易ではない．

一方，ゲーム記述に特化した手法として，柴崎 [5] による通信サービス仕様記述言語 SAL を利用した多人数ゲームシナリオ記述がある．SAL によって記述されたシステムは，プレイヤからのメッセージを受け取り，メッセージの種類に応じたメッセージを返して，プレイヤからのさらなるメッセージの入力を促す．こちらはゲーム記述としてより直感的だが，数値シミュレーションの記述を想定してはいない．

3. ゲーミングシミュレーションの共通構造

3.1 対象とする GS

本研究で対象とする GS は，プレイヤが計算機によってシミュレートされた，テーマに沿って簡略化された環境を観測して意思決定を行い，その結果が環境に与えた影響を観測することを通じて学習することを狙いとするような GS である．

より具体的には，単純な構造体と数値を用いた小規模なシミュレーションを環境として扱い，プレイヤがルールによって定められた可能な行動のうちいずれを行うかを意思決定する

ことにより，またそれによってのみ環境が変化するような GS である．

3.2 GS 記述の概要

このような GS を記述するには 3 つの情報が必要とすると考えられる．それは環境を表現するパラメータの一覧，プレイヤがとりうる行動の一覧，プレイヤがいつ・どの行動を取れるかを制限するルールである．これらの情報をすべて記述した GS 記述を今後 “シナリオ” と呼ぶ．

本研究で提案するゲーム記述言語 “Karina” は，このシナリオを記述するための言語であり，そのソースコードはこれら 3 つの情報を記述する部分から構成され，順にデフォルト部，カード宣言部，シーン宣言部と呼ぶ．各部の詳細については，4 章にて述べる．

3.3 インタラクションの共通構造

プレイヤの行動のうち，環境に影響を与えるものはあらかじめ決定されている．それがどんな行動であるかはシミュレートしたゲームであるかによって異なるが，プレイヤが行動を宣言し，それが環境をシミュレートする計算システムに影響を与えるまでのプロセスについてはいずれのゲームも共通であると考えられる．

3.3.1 基本的行動とカード

まず，各プレイヤの行動は，いくつかの基本的な行動の系列であると考えられる．ここで言う基本的な行動とは，環境に影響を与える可能性のある，なおかつ他プレイヤが割り込めない一連の行動であるとする．そうであるならば，複数プレイヤの同時行動は，各プレイヤの基本的な行動を何らかの方法で順序づけて順に処理していくのだと考えればよい．

この基本的行動を図 1 のようなカードとして表現する．カードは名前と，シミュレートする行動の説明を含み，引数を与えることで程度や対象を変更できるものとする．

引数の入力が数値の入力と選択肢だけである理由は，引数のうち選択肢は変化させるパラメータを選択することであり，引数のうち数値は変化の程度を決めることであると考えれば，環境への影響を記述するのに十分だからである．

前述のカード宣言部では，このカードの情報について記述する．

3.3.2 行動許可とシーン

次に，プレイヤが行動を許可されるタイミングについて考える．これは，プレイヤにいつどのカードを渡すか，と言い換えることができる．

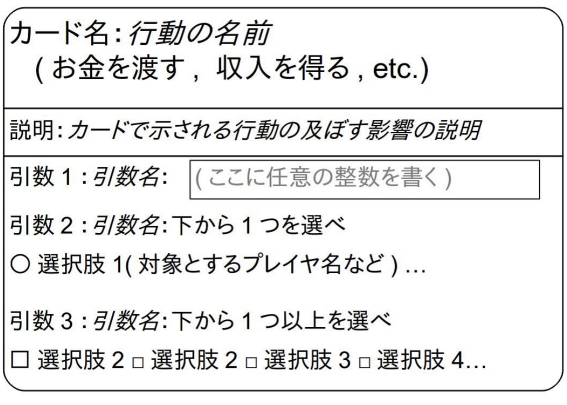

図 1: カードの例

配布条件は，プレイヤの状況あるいは環境の状態の，どちらかで決めることができるだろう．教育上は，例えばプレイヤが興味を持った段階で配布するというルールも考えうる．しかし，ここではシミュレートされた環境の中で可能であるかどうかに限定し，柔軟な対応についてはファシリテータの判断で対応するものとする．従って，環境のパラメータを監視し，それが条件を満たしたらカードを配る，という形式でルールを記述すればよい．

しかしそれは煩雑であり，プレイヤにとっても分かりづらいことが予想される．そこで，シミュレーションはいくつかの段階を追って進むものとし，各段階ごとに可能な行動を割り当てることにする．例えば経営シミュレーションなら生産計画，仕入れ，販売，決算のような段階があり，プレイヤはそれぞれの段階で異なった行動を許可されるだろう．この段階を “シーン” と呼ぶ．シーンは GS の状態のひとつ (環境の状態ではない) であると考えられる．

そして，シーンが変わる条件は，環境が決まった状態になるか，あるいはプレイヤが望む行動をすべて終えた場合であろう．プレイヤがそれ以上行動しないことの意思表明もカードで表現すれば，シーンの推移は環境のパラメータの監視と，推移条件の記述として表現できる．

前述のシーン宣言部では，このシーンを記述する．

3.4 実装の共通構造

このように記述された GS を実際にプレイできるように実装するとき，必ず含むコンポーネントが 4 つあると考えられる．それは環境の現状を記憶するデータベース，データベースを更新するプロセッサ，プレイヤに環境の状態とプレイできるカードを示すユーザインタフェース，プレイヤにカードを配り，カード提出をプロセッサ呼び出しに変える仲介役である．

これらのコンポーネントは図 2 のように関連する．矢印は情報の流れ，番号は処理の順番を示している．ゲームを自動生成する場合，このような構造のプログラムを生成することになる．

4. ゲーム記述言語の提案

上述の共通構造に基づいてゲームを記述したい．そのための言語が Karina であり，次のようなシンタックスでこれを記述する．

4.1 デフォルト部

デフォルト部は，ユーザ定義データ型を宣言するデフォルト節，パラメータと初期値を宣言する初期化節，ゲームの終了条

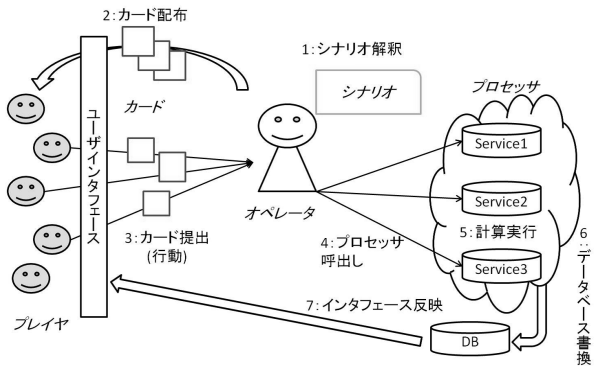

図 2: 実装の共通構造

件を記述する終了節からなり，以下のように記述する．

```
//デフォルト節
default
  //例：プレイヤ型
  Player
    //メンバ変数定義文の例：money を 10 で初期化
    money = 10;
    ...
  end
end

//初期化節
initialization
  //A というパラメータを 0，B をプレイヤ型で初期化
  A = 0;
  B = new Player;
  //メンバ変数に違う初期値を与えて初期化
  C = new Player specified
    money = 20;
  end
  ...
end

//終了節．条件を満たせばゲームは終了
gameover
  //when 文のひとつでも真になればゲーム終了
  //シーン A，あるいは B で，
  //  パラメータ A が 0 以上なら終了
  when(A > 0) in [SceneA, SceneB]
  ...
end
```

4.2 カード記述部

カード宣言部は複数のカード宣言節から成る．ひとつのカード宣言節は 1 枚のカードに対応し，以下のように記述する．

```
defcard CandName "カードの説明"
  //引数副節：arg_one に数値を入力させる
  arg_one = input;
  //check は候補から 0 個以上，radio は 1 つ選択
  arg_two = check([PlayerA, PlayerB, PlayerC]);
  ...

  //処理副節．カードに対応して行われる処理．
  //プロセッサ呼び出しによって環境を変更
  //条件分岐と繰り返しも記述できる
  P.set(Variable, new_value);
  ...
end
```

4.3 シーン記述部

シーン宣言部は複数のシーン宣言節から成る．ひとつのシーン宣言節は 1 つのシーンに対応し，以下のように記述する．

```
defscene SceneName "シーンの説明"
  //配布副節．シーンごと使えるカードを配る．
  //deal_to 文と条件分岐によって配布を記述
  //プレイヤ A と B にカード A と B を配る場合
  deal_to([PlayerA, PlayerB], [CardA, CardB])
  ...

  //推移副節．条件により別シーンへ推移．
  //ひとつの NEXT_SCENE_IF 文により条件分岐
  next_scene_if test_one then
    //シーン A へ推移
    next_scene_is(SceneA);
  else
    next_scene_is(SceneB);
  end
end
```

表 1: 概念的な構造と実装の対応

概念的な構造中の要素	要素の実装
オペレータ	CGI プログラム
プロセッサ	サブルーチンもしくは計算サービス
データベース	サーバ上データベース
ユーザインタフェース	Web ページ

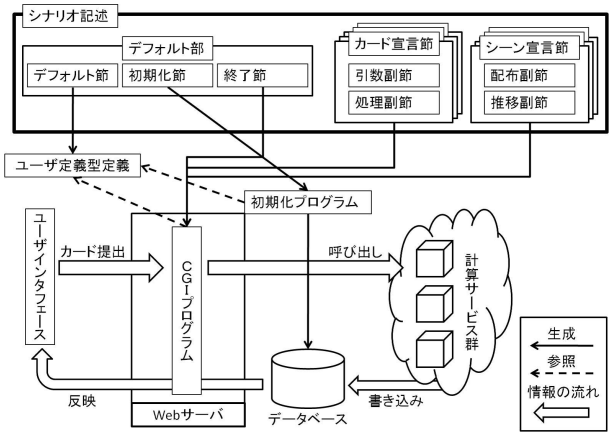

図 3: 実装への変換

5. トランスレータとゲームの実装

5.1 トランスレータ

Karina によるシナリオ記述が実装へと自動的に変換可能であることを示すため，シナリオを Web アプリケーションに変換するトランスレータを実装した．

すでに 3.4 節と図 2 において，Karina が求める実装の概念的な構造を示した．トランスレータは記述に基づいて，これに実装を与えていくものであると言うことができる．概念的な構造と実際の実装の対応は表 1 に示したとおりであり，記述を参照して実装が生成されるプロセスは，図 3 で示す通りである．

プレイヤはゲームの状況を，それを記述した Web ページを通じて知り，フォームを通じて CGI プログラムに行動を表明する．つまり Web ページに表示されているフォームがカードである．

CGI プログラムは呼び出されるとフォームに応じたサブルーチンを呼び出す．サブルーチンはデータベースを変更する機能を持つ．最後に CGI プログラムは環境の情報を記述した表と，シーンに応じたカードを Web ページに書き出し，プレイヤに新たな状況を伝える．

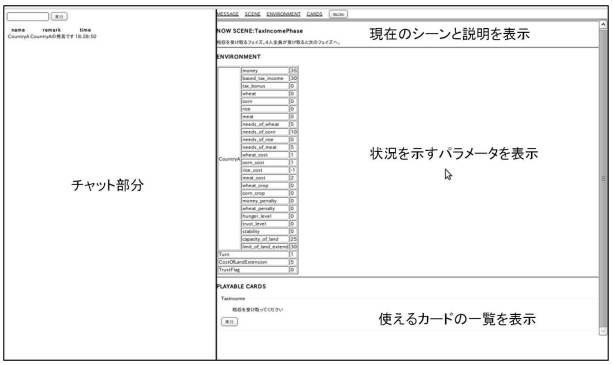

図 4: スクリーンショット

5.2 検証

このトランスレータによってシナリオから実装を生成できることを示すため，Tsunoda ら [6] による国際食料問題を扱った GS を実装した．

この GS では，プレイヤはそれぞれの国を統治する．プレイヤは自国民に必要な食料をするために，年のはじめに生産計画を立て，ランダムなイベントの後収穫が行われる．ただし，国ごとに作れない作物があったり，資金や土地が不足している場合がある．食料が不足すると治安が悪化し，ランダムなイベントによって破壊的な事態が起こる場合があるので，各プレイヤは食料や資金を融通して食糧問題に対処せねばならない．

この GS における各国の資金，土地の状態，生産計画や収穫量などは簡単な数値シミュレーションであり，プレイヤが取りうる行動はすべてルールで定められている．そして，ランダムイベントについてもそれが発生するタイミングははっきりしているので，全員が生産計画立案の終了を宣言すると同時にイベントが発生することにすれば，プレイヤの意思決定なしに環境が変化することはないといえる．以上のことから，この GS は本研究が対象とする GS であるといえる．

実装されたゲームのユーザインタフェースを図 4 に示す．画面左はチャットであり，右にはカードが表示されている．プレイヤはフォームに値を入力したり，チェックボックスをクリックすることで，行動のパラメータを設定し，実行ボタンをクリックすることでカードを提示する．

提案フレームワークを用いることで，対象とした GS のシナリオを記述することができた．このシナリオは 451 行であり，元の PHP, Javascript, MySQL による 1772 行のプログラムから大幅に少なくなっている．

またこの記述には Web アプリケーションの実装に関する事項は書かれていないが，トランスレータによって実際に動作するゲームの CGI プログラムを自動生成することができた．

このことから，GS の記述を容易にすると共に，GS の実装コストを低減することができたと考えられる．

6. 拡張：プレイヤエージェント

上述の共通構造にしたがってゲームを記述するならば，これに対応したエージェントも共通の構造に基づいて作成できるはずである．すなわち，プレイヤ同様にユーザインタフェースを通して環境を認知し，何らかの条件に従ってカードを提出するエージェントを作成できるはずである．

これを実証するために，簡単な IF-THEN ルール記述から，エージェントを生成するプログラムを作成した．

ルール記述は，次のようなシーンごとの動作ルールの集合として書ける．

```
(in SceneA
  (cond ((> ParameterA 1)
         (CardA
           (arg1 1)
           (arg2 (check (Player1 Player2)))
           (arg3 (select (Player1 Player2)))))
        ...))
```

この例では，エージェントは SceneA においては ParameterA の値を参照し，それが 1 より大きければ CardA を提出することになる．CardA の引数は，直接数値を入れるか，候補から選べばよい．

前述のとおり，ゲームのデータは表で表現され，カードの提出はフォームの送信である．そのためルールの解析器に HTML の解析モジュールと Web スクレイピングモジュールを組み合わせることで，ユーザインタフェースの HTML を解析して状況を認知し，HTML を解析して状況を認知し，ルールに基づいて意思決定し，フォームを通して情報を送信するエージェントを生成することができた．

7. おわりに

本研究では，シナリオ記述言語 Karina とそのトランスレータを利用したゲーミングシミュレーション作成フレームワークを利用することで，非プログラマにも容易にゲームを記述できるようになり，Web アプリケーションとしてのゲームを自動生成することで実装のコストを低減することができることを示した．

多様な実装に対応することと，その品質，特に今回シナリオから捨象された見た目についての制御を可能にすることが今後の課題である．

参考文献

[1] 新井潔・出口弘・兼田敏之・加藤文俊・中村美枝子:ゲーミングシミュレーション, 日科技連出版社, 1998

[2] Cathy S.Greenblat:ゲーミングシミュレーション作法, 共立出版, 1994.

[3] H. Tanuma, H. Deguchi, and T. Shimizu, SOARS: Spot Oriented Agent Role Simulator: Design and Implementation, *Post-Proceeding of the AESCS International Workshop* 2004, pp.49-56, 2004.

[4] Toru Ishida. Q: A Scenario Description Language for Interactive Agents. *IEEE Computer*, Vol.35, No. 11, pp. 42-47, 2002.

[5] 柴崎雅史: 多人数ゲームシナリオの記述と検証法, 情報処理学会研究報告 マルチメディア通信と分散処理, Vol.47, No.5, pp47-54, 1990.

[6] Keisuke Tsunoda and Reiko Hishiyama, Design of Multilingual Participatory Gaming Simulations with a Communication Support Agent, *28th ACM International Conference on Design of Communication (SIGDOC 2010)*, pp17-25, 2010.