

プランナーのための AI システム考察

－ 「知識ベース」の活用 －

2014 年 3 月 13 日 初稿

板垣 衛

■ 改訂履歴

稿	改訂日	改訂者	改訂内容
初稿	2014 年 3 月 13 日	板垣 衛	(初稿)

■ 目次

■ 概略	1
■ 目的	1
■ 方法論	1
▼ 「知識ベース」の AI	1
● 「Prolog」とは	1
● 「知識ベース」の活用	3
● 学習型 AI : 「フラグ」よりも「知識ベース」	4
▼ 「範囲ベース」の NPC 配置とモンタージュ	4
▼ 有限オートマトン	5
● 「有限オートマトン」とは	5
● 「有限オートマトン」と「知識ベース」	6
▼ データベースの活用	7

■ 概略

RPG の村人のような NPC に対して、効果的で生産性の高い AI の管理方法について考察する。

プログラマーに限らず、プランナーにも向けた内容である。

■ 目的

本書は、ゲーム機のスペック向上に合わせ、雑踏シーンなどの雰囲気を作り盛立てるための大量 NPC 配置を実現するための方法を考察し、生産性が高く効果的な実装方法を実践するための手がかりとすることを目的とする。

群衆の中であって、外見に見合った個性的な行動を取らせるための設定を、プランナーが意図したとおりに行えるようにすることを目指す。かつ、その生産性も現実的なものとする。

複雑な AI の処理を設計することではなく、「大量のキャラに対する AI の設定を、プランナーが分かり易く簡単に行えるようにすること」（そのような設計の土台を考案すること）の実現を目指す。

■ 方法論

▼ 「知識ベース」の AI

プログラミング言語「Prolog」の「知識ベース」の手法を模倣し、シンプルな設定で柔軟性と拡張性を持った AI を構築する。

● 「Prolog」とは

まずは「Prolog」について説明する。

「Prolog」とは、「知識ベース」として「事実」と「ルール」を定義しておく、「質問」に回答してくれる、プログラミング言語の一種（とされているもの）である。

以下、Prolog のサンプルを示す。細かい表記の説明等は割愛し、分かり易いように日本語で示す。（本来は小文字ならアトム＝固定値、大文字なら変数といった表記上の

ルールがあるので、Prolog のサンプルとしては若干逸脱している)

【サンプル①】

事実の定義：

```
人間(ソクラテス).           //事実:「ソクラテス」は「人間」
人間(プラトン).             //事実:「プラトン」は「人間」
人間(アリストテレス).       //事実:「アリストテレス」は「人間」
悪魔(メフィストフェレス).   //事実:「メフィストフェレス」は「悪魔」
```

ルールの定義：

```
死ぬ(X) :- 人間(X).          % ルール:「X」が「人間」なら、「X」は「死ぬ」
```

質問：

```
?- 死ぬ(ソクラテス).         % 質問:「ソクラテス」は「死ぬ」?
yes
?- 死ぬ(メフィストフェレス). % 質問:「メフィストフェレス」は「死ぬ」?
no
?- 死ぬ(Who).                 % 質問:「死ぬ」のは「Who」?
Who = ソクラテス ;
Who = プラトン ;
Who = アリストテレス .
```

【サンプル②】

事実の定義：

```
好き(太郎, 寿司).           % 事実:「太郎」は「寿司」が「好き」
好き(次郎, ケーキ).          % 事実:「次郎」は「ケーキ」が「好き」
好き(三郎, 寿司).            % 事実:「三郎」は「寿司」が「好き」
好き(三郎, すき焼き).         % 事実:「三郎」は「すき焼き」が「好き」
好き(四郎, すき焼き).         % 事実:「四郎」は「すき焼き」が「好き」
```

ルールの定義：

```
友達(X, Y) :- \+(X = Y), 好き(X, Z), 好き(Y, Z). % ルール:「X」と「Y」が同じではなく、
                                                    % 「X」が「Z」を「好き」で、
                                                    % 「Y」が「Z」を「好き」なら、
                                                    % 「X」と「Y」は「友達」
```

質問：

```
?- 好き(太郎, 寿司).         % 質問:「太郎」は「寿司」が「好き」?
yes
?- 好き(太郎, ケーキ).        % 質問:「太郎」は「ケーキ」が「好き」?
no
?- 好き(三郎, What).          % 質問:「三郎」が「好き」なのは「What」?
What = 寿司 ;
What = すき焼き .
?- 好き(Who, 寿司).           % 質問:「寿司」が「好き」なのは「Who」?
Who = 太郎 ;
Who = 三郎 .
?- 友達(太郎, 次郎).          % 質問:「太郎」と「次郎」は「友達」?
no
?- 友達(太郎, 三郎).          % 質問:「太郎」と「三郎」は「友達」?
Yes
?- 友達(三郎, 太郎).          % 質問:「三郎」と「太郎」は「友達」?
Yes
?- 友達(太郎, 太郎).          % 質問:「太郎」と「太郎」は「友達」?
no
?- 友達(次郎, Who).           % 質問:「次郎」と「友達」なのは「Who」?
no
?- 友達(三郎, Who).           % 質問:「三郎」と「友達」なのは「Who」?
```

Who = 太郎 ;
Who = 四朗 .

【サンプル③】

事実の定義：

父親(家康, 秀忠).	% 事実:「家康」は「秀忠」の「父親」
父親(秀忠, 家光).	% 事実:「秀忠」は「家光」の「父親」
父親(家光, 家綱).	% 事実:「家光」は「家綱」の「父親」

ルールの定義：

祖先(X, Y) :- 父親(X, Y).	% ルール: 「X」が「Y」の「父親」なら、「X」は「Y」の「祖先」
祖先(X, Y) :- 父親(X, Z), 祖先(Z, Y).	% ルール: 「Z」が「Y」の「祖先」で、「X」が「Z」の「父親」なら、 % 「X」は「Y」の「祖先」

質問：

```
?- 父親(家康, 秀忠).      % 質問:「家康」は「秀忠」の「父親」？
yes
?- 父親(家康, 家光).      % 質問:「家康」は「家光」の「父親」？
no
?- 祖先(家康, 秀忠).      % 質問:「家康」は「秀忠」の「祖先」？
yes
?- 祖先(家康, 家光).      % 質問:「家康」は「家光」の「祖先」？
yes
?- 祖先(家康, Who).      % 質問:「家康」が「祖先」なのは「Who」？
Who = 秀忠 ;
Who = 家光 ;
Who = 家綱 .
```

この他、Prolog には大きなデータを処理するための「リスト」(配列)もサポートしているが、そこまでの紹介は省略。

Prolog は、事実とルールに基づいて回答を得たい時に用いられる。交代勤務のシフト決めやパズルを解くといったことができる。Excel の「ゴールシーク」や「ソルバー」といった機能に近い。

- 「知識ベース」の活用

「知識ベース」は直感的に理解し易い。この性質を利用し、例えば、以下のような NPC の行動設定を設けると、分かり易く量産ができそうである。

サンプル：NPC 設定

- ・「ジェームズ」の設定：性別（男）、世代（子ども）、母親（メアリー）
- ・「ジョン」の設定：性別（男）、世代（青年）、所属（グループA）
- ・「ロバート」の設定：性別（男）、世代（青年）、所属（グループB）
- ・「メアリー」の設定：性別（女）、世代（おばさん）、好き（食べ物、にんじん）

サンプル：NPC 行動設定

- ・「性別（男）＋世代（子ども）」の行動設定：「母親（Who）」に「ついて行く」
- ・「所属（グループA）」の行動設定：「（近くの）グループA（Who）」に「挨拶」
- ・「所属（グループA）」の行動設定：「（近くの）グループB（Who）」を「避ける」
- ・「世代（おばさん）」の行動設定：「好き（食べ物、What）」を「売っている店に行く」

「行動」の処理は一つ一つプログラマーに依頼することになるが、「その行動をど

のような条件に適用するか」という設定は、プランナーでも分かり易く行うことができる。キャラクター一体一体の行動を細かに設定していなくても、キャラ設定と行動パターンさえ用意すれば、だいたいそれっぽく動いてくれるので、大量配置が難しくない。

なお、「移動」の行動については、基本的に経路パスを用いて、行先までのルートを決めることを想定。「ルート探索」については本書の範囲外とする。

この知識ベースの AI 設定と、後述の「有限オートマトン」を組み合わせると、更に生き生きとした行動を与えることが可能と考える。

● 学習型 AI : 「フラグ」よりも「知識ベース」

「知識ベース」はセーブデータにも活用し、ゲームプレイに応じた NPC の行動変化にも利用する。

例えば、ある NPC にお使いを頼まれるようなことがあるとして、依頼を達成できたら「好き(主人公)」という知識ベースが追加され、達成できなければ「嫌い(主人公)」という知識ベースが追加される。この NPC の普段の行動設定に「好き(Who)に挨拶」「嫌い(Who)に背を向ける」といったものを用意すると、ゲームプレイに応じた行動をとるようになり、面白味が増す。

このような行動の処理は、通例ではフラグ管理とスクリプトで行うものである。しかし、以上のような「知識ベース」に基づく処理（およびワークフロー）を確立すれば、直感的で一貫した手法で、このような凝った行動をとらせることができる。

▼ 「範囲ベース」の NPC 配置とモンタージュ

「知識ベース」を活用して量産の手間を軽減するとしても、大量の NPC を一体一体配置するのはやはり手間がかかりすぎる。

そこで、「範囲」に対して NPC を配置するような対応を行う。

【範囲設定】

まず、あらかじめ NPC の初期位置となる範囲（例えば「市場の通り一帯」など）を囲むボックスを用意する。その範囲内で、実際に出現可能な位置には「コリジョン属性」や数点～数十点の「ポイント」を設定しておく。

【論理配置設定】

この範囲に対して「民族 A の若い男性 80%+民族 B の老人女性 10%+民族 C の母子 10%」のようなおおまかな配置を設定する。

【実配置設定】

この範囲に対して「1～3章は20人」「4章は10人」といった実際の配置を別途与える。（「4章」では事件が起こって外出する人が減っていることを表現。更に、「5章では部族Dが大量に移民してきたので論理配置も変更」といった表現にも対応。）

【モンタージュ設定】

「アサシンクリード」シリーズのようなNPCのモンタージュ設定も併用する。

例えば、「部族Aの若い男性」に対して「服」3パターン、「ズボンと靴」3パターン、「髪」3パターン、「顔」3パターン、「体系」3パターンを用意し、さらに、「太った体系の比重が80%」といった設定を行う。これにより、十分に外見のばらついたキャラが配置される。

【配置設定に対するAI設定】

範囲に対して、「知識ベース」より優先的に適用されるAIも設定可能にする。例えば、お立ち台の上に「演説キャラ」を固定配置し、その周囲に範囲設定でNPCを配置する。この範囲に対しては、「演説キャラに目を向ける」という行動AIを設定することに、「場面」にふさわしい行動を、極力手間をかけずに与えることができる。

▼ 有限オートマトン

「知識ベース」だけに頼ると、行動パターンが単調になり過ぎてしまう上、頻繁な知識ベースの照合により、AI処理が重くなってしまう。

そこで、「知識ベース」で行動を決定したあとしばらくの行動パターンを「有限オートマトン」で定義する手法を取る。

● 「有限オートマトン」とは

まずは「有限オートマトン」について説明する。

「有限オートマトン」は、有名なところでは「正規表現」の文字列照合のアルゴリズムに用いられている。そのごく簡単なサンプルで説明する。

【サンプル①】

検索パターン（正規表現）：

`0[0-9]+ ... 「0」を頭文字に「0～9」が1個以上連続する文字列にマッチする`

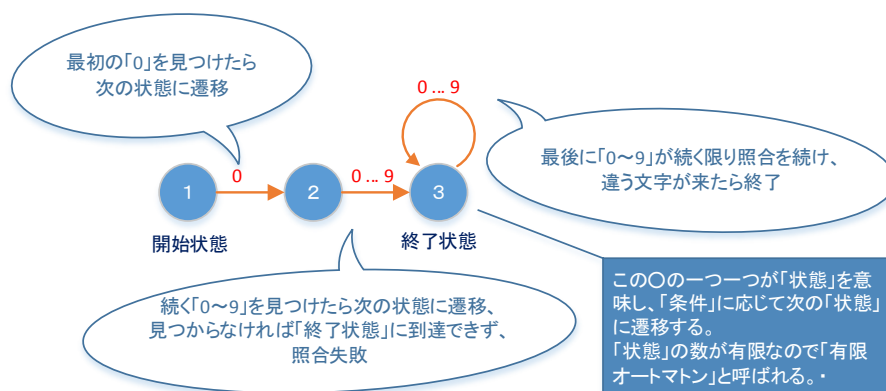
検索対象文字列：

`389 ポイントのダメージ、0 個のアイテム、No. 015 のカード`

結果：

`015`

有限オートマトン：



● 「有限オートマトン」と「知識ベース」

「有限オートマトン」と「知識ベース」を組み合わせた AI の設定を考察する。

サンプル：「世代（おばさん）」の行動設定 ※大雑把なイメージ

行動①：「料理(What)」を「テーブルに運ぶ」【開始状態】
 ⇒行動：目的地に到着
 ⇒行動：席について食べる（時間経過）
 ⇒行動：一時知識ベース「料理()」を削除
 ⇒行動：一時知識ベース「買ったもの()」を削除【終了状態】

行動②：「買ったもの(食べ物, What)」を「自分の家(Where)に持って帰る」【開始状態】
 ⇒行動：目的地に到着
 ⇒行動：台所に移動
 ⇒行動：目的地に到着
 ⇒行動：料理中（時間経過）
 ⇒行動：一時知識ベース「料理(スープ)」を記録【終了状態】

行動③：「好き(食べ物, What)」を「売っている店(Where)に行く」【開始状態】
 ⇒行動：目的地に到着
 ⇒行動：買い物（時間経過）※店主との掛け合い
 ⇒行動：一時知識ベース「買ったもの(食べ物, 好き(食べ物, What))」を記録【終了状態】

行動④：無条件【開始状態】
 ⇒行動：ぼーっとしている（時間経過）【終了状態】

【解説】

最初は「料理()」も「買ったもの()」も知識ベースにないので、「行動③」が選ばれる。ただし、知識ベースに「好き(食べ物)」が設定されていないキャラは「行動③」も選択できないので、「行動④」が選ばれる。

「行動③」が終了状態となると、また頭から行動の照合を行う。「買ったもの()」があるので、今度は「行動②」が選択される。

同様に「行動②」が済んだあとは「行動①」が選択され、その後はまた「行動③」が選択されるといった流れが繰り返される。

一つ一つの行動に実現が難しい要素があるとしても、AI のデータとしては、条件（開

始状態)と行動(状態)を列挙するだけなので、データ化がし易く、プランナーにも分かり易い。

このように、「知識ベース」と「有限オートマトン」の組み合わせは、下手にスクリプトで制御するよりも、簡潔で直感的で多彩なフローを実現できるものとする。

▼ データベースの活用

AIとは直接関係ないが、NPCの設定に関する考察として追記する。

NPCの設定をExcelベースで管理する場合、レベル別や章別などの区分を意識して、時には複数のファイルに同じデータを記述するようなことになる。

これはゲームの動作上の都合(データの読み替えの都合)によるものではあるが、データを作るプランナーの立場からしたら、面倒な気遣いをしなければならず、問題も起こし易い。

そこで、別途「ゲームデータ管理DBシステム」で提案しているように、データベースで一元管理し、実機用のデータを出力する際に、都合に合わせた切り分けて出力するようになっていれば、プランナーとしては扱い易い。

このような仕組みであれば、「ゲームの節目節目で街から街に渡り歩くNPC」のようなものもスムーズに設定できるものとする。

■■以上■■

■ 索引

索引項目が見つかりません。

プランナーのための AI システム考察

以 上