

ゲーム A I 連続セミナー「ゲーム A I を読み解く」

第 2 回

F.E.A.R におけるゴール指向アクションプランニング

資料

Based on

Jeff Orkin, “Three States and a Plan: The A.I. of F.E.A.R”

http://www.jorkin.com/gdc2006_orkin_jeff_fear.doc

フロム・ソフトウェア 技術部

三宅 陽一郎

y_miyake@fromsoftware.co.jp

更新履歴

Ver1.0 2007 年 2 月 8 日 第 2 回セミナー前日に事前資料として配布。

Ver2.0 2007 年 4 月 2 0 日 全面改訂、資料配布。

(注) このテキストを読むのに前提となる知識は何一つありません。

目次

第 1 章	ゴール指向プランニング	9
第 1 節	ゴール指向型 A I とは？	9
第 2 節	プランニングとは？	10
第 3 節	連鎖によるプランニング	13
第 4 節	連鎖によるプランニング (2) 複数の可能なプラン	14
第 5 節	人手によるプランニング	18
第 6 節	ゴール指向型プランニングとは？	20
第 7 節	まとめ 21	
第 2 章	F.E.A.R. におけるゴール指向アクションプランニング	22
第 1 節	F.E.A.R. の AI が目指したもの ~ Shogo、NOLF2、そして、F.E.A.R. ~	22
第 1 項	ゴール指向型エージェント	24
第 2 項	知識表現、世界表現	26
第 3 項	行動の決定	28
第 2 節	F.E.A.R. におけるゴール指向プランニング	31
第 1 項	ゴール指向型エージェント	31
第 2 項	知識表現、世界表現	36
第 3 項	行動の決定	42
第 3 節	より詳しいプランニングの解説	45
第 1 項	複数のプランから一つを選ぶ	45
第 2 項	キャラクターの違いによるプランニング	47
第 4 節	リプランニング	49
第 5 節	まとめ 50	
第 3 章	クロムハウズにおける階層型ゴール指向プランニング	53
第 1 節	クロムハウズ	53
第 2 節	階層型ゴール指向プランニング	54
第 3 節	クロムハウズにおける階層型プランニング	59
第 4 節	クロムハウズにおけるリプランニング	59
第 5 節	ゴールの作り方	62
第 6 節	ゴールの実装方法	64
第 7 節	意思決定 67	
第 8 節	キャラクターごとの個性付け	72
第 9 節	オフラインミッションへの応用	72
第 10 節	チーム AI へ	73
第 11 節	まとめ 73	
第 4 章	自分の開発するゲームのために	74
第 1 節	依存グラフとプランニング	76

第2節	R P Gとプランニング	79
第3節	ストラテジーゲームとプランニング	80
第4節	集団の制御とゴール指向プランニング	81
第5節	多様な行動.....	83
第6節	A Iに時間を与える	84
第7節	まとめ	86
第8節	展望	87
Appendix	Mat Buckland によるゴール指向プランニングのコーディング例.....	88
参考文献	92

ゲーム AI 連続セミナーの目指す方向（２）

コンシューマーゲーム機、パーソナル・コンピュータにおけるハードウェアの発展は、ゲーム開発において、ゲームが扱うべきデータを増大させ、また、ゲーム内のフィールドを拡大にし、レベルデザインをより複雑にさせつつあります。この発展によって、ゲームにおける AI（以下、ゲーム AI）が処理すべき情報もまた複雑で高度に抽象的ものになりつつあります。と同時に、デジタルゲームの歴史が日々進んで行く中で、ユーザーのゲームに対する要求も日々成長し、ゲームにおける AI は、こういったユーザーの予想を超える賢明さを持つことなくしては、もはや驚かすことは出来ません。

実際のゲーム AI 開発において開発者が直面する問題は「AI が処理すべきゲーム情報（状態）の複雑さと多さ」です。例えば、COM に 1 階から 2 階へ上がって、レーザー砲で階下の敵を迎撃して欲しいとします。そのためには、まず 2 階までのパスを検索し、敵の射線をよけながら進む必要があります。しかし、移動の途中で敵の手榴弾が進路を阻んでしまうかもしれません。また、移動している最中に味方が既に全滅しているか、または、敵が全滅しているかもしれない。そういうこともなく、運良くレーザー砲に辿りついたとしても、敵プレイヤーが既にそこに先にたどりついているかもしれません。そこで交戦が始まった場合、COM は一体どうするべきなのでしょう？ こういったシークエンスで起こる全ての事象について、開発者が AI に答えを用意することが必要なのでしょうか？ それはまた可能なことなのでしょう？

簡単な行動一つにも AI には予期できる限りの全ての状況に対応する必要があります。特に、プレイヤーという不確定要素は、任意のタイミングで NPC の行動を妨害することが可能であり、AI が対応すべき状態を時間に沿って増大させて行きます。ゲームとゲームプレイヤーの発展によって、ゲーム AI は単純反射的な仕組みから、複雑な情報と膨大な情報を処理する情報収集能力と抽象度の高い思考能力が求められつつあるのです。

一方で、アカデミズムにおける人工知能の研究は、ある方向から見れば、この 50 年間、「複雑な状況に、如何に効率的に処理するか」という目的に沿って発展して来た学問と言えます。アカデミックな AI の研究においても、日常的な簡単な問題を AI に解かせようとするだけで、たいへんな複雑な世界を相手にしなければなりません（こういった問題を人工知能ではフレーム問題（[1]）と言います）。例えばロボットに「隣の部屋のコップを持って来い」と命令します。ロボットには、まず「隣の部屋」の意味がわかりません。「隣」とは何か。「部屋」とは、何か。壁で仕切れていれば部屋なのか。だったら、日本の和室はどうなのか？ よしんば、隣の部屋がわかったとします。パスを検索します。もちろん、あらかじめ地図のデータは与えておきます。そのパスに従って動きます。ドアは半開きになっているとします。「それを明ける必要があるだろうか？ そういえば、自分の体の大きさはどれぐらいなのか？」なんとかドアを開けます。隣の部屋に着きます。人がいて、邪魔をしてコップが見えない。「どう行動するべきか？」やっとのことで、コップをみつけます。どうやってつかめばよいか。カメラをよせます。距離を計算します。腕を伸ばします。つかみます。「つかんだ！」もとの部屋に戻ろうとします。もと来たドアは閉まっている…。人工知能は、こういった複雑な状況を処理するべく、人間の持つ知識をルールで与えたり、直接知識を準備できないなら学習させたり、生物のモデルを真似て反射を多層に重ねたり、様々なアプローチ試し、知識や技術を築いて来ました。

そして、全く同様に、数年において、次世代機や PC が飛躍的に複雑化させたゲーム空間において、NPC たちはゲーム世界の複雑さ、混沌さに直面しています。そして、多くのゲーム開発者が、自分たちのゲームのNPCにこの混沌をエレガントに泳ぎ渡って行くための技術を人工知能に求めており、今こそは、蓄積された人工知能技術をゲームへ応用し始める時期であると考えられます。

この連続セミナーは、そういった要求に対して、主に欧米のゲーム AI における質の高い仕事を、講義、グループワーク、パネディスカッションという形式によって全 6 回に渡って提示することにより、日本のゲーム開発者にゲーム AI が拓く次世代のゲームの可能性を示し、ゲーム開発をより広い視野のもとで展開して行くことを目標としています。本セミナーは IGDA 日本のゲーム AI 運営メンバーによって、アイデアを交わし改善を重ねつつ運営されています。

日本の作るゲームには欧米にない独創性があります。新しいアイデアのゲーム、或いは、新しいゲームのジャンルさえ作り出してしまう土壌があります。コンテンツに依存する度合い高いゲーム AI において、これの事実は非常に大きな意味を持っています。日本はゲーム AI において、欧米とは違った独創的な発展を持つ可能性を秘めています。

第 1 回では Killzone、クロムハウন্ズ、Halo2 などを題材として、AI が空間やフィールドにあるオブジェクトをうまく（賢明に）使うための技術として「知識表現」「世界表現」について解説しました（[2]）。

第 2 回では、AI が時間をうまく利用する技術として「プランニング」を紹介します。「プランニング」とは、「設定した目標を達成するために必要な一連の行動を作り出す技術」です。これによって AI は始めて自分自身の思考によって連続した時間の中で行動を展開する能力を獲得します。

この両者の技術を組み合わせることで、AI に時間と空間を自由に渡って行く力を与えることが出来ます。その能力は、ゲームにおいては「自律型エージェント」というフレームの中で最高の力を発揮します。そして、その基礎の上には、自然と「集団としての知能」の可能性が拓けて来ます。これは第 3 回のセミナーで取り上げます。

多くの卓越した技術や知識を、人工知能の研究、文献、書籍が提供してくれます。そして、アカデミックな研究の中にもデジタルゲームにおける問題に真剣に取り組んで行こうという流れがあります。しかし、それをを用いて実際の開発タイトルにおいてデジタルゲームの人工知能の可能性を引き出すことはゲーム開発者自身にしか出来ません。そして、その可能性は、ゲーム開発に真摯に向かいあう人間が開拓して行くべきものです。

このテキストがその踏み石の一つとなることを願います。

F.E.A.R におけるゴール指向アクションプランニング

このテキストでは、A I に時間に沿って高度で知性的な行動を取らせるための方法を解説します。第 1 回のセミナーではマップ全体を賢く動き回る「世界表現」について説明しましたが、今回は、連続した時間に渡って賢い行動を創造する「プランニング」の方法について説明します。この二つを合わせることで、ゲームのキャラクターに「広大な空間の中でオブジェクトを巧みに使用しながら、長い時間に渡って自分の行動を決定できる」能力を与えることが出来ます。

「単純反射」(この条件を満たせば、こう行動する) による制御に対して、「未来に達成すべきゴールに向かって、それを実現するための一連の行動を計画し実行する」技術を「プランニング」と言います。「プランニング」は、工場などにおける工程の管理から、複雑な飛行機の組み立て、ハッブル宇宙望遠鏡の観測の自動プランニングに至るまで広く世の中で「プランニング・アプリケーション」として応用され成功して来た技術です ([1] 12 章)。近年は、ゲーム A I に対して本格的な導入が始められ、F.E.A.R ([3])、Age of Empire、クロムハウنزなど、様々なジャンルのゲーム A I に応用されています。これからゲーム A I の中心となる技術の一つとして有望視されています。

しかし、工程管理や観測計画と言った状況が既に定式化されたアプリケーションレベルのプランニングに対し、ゲーム A I のプランニングには

- ◆ ゲームフィールドのような複雑な舞台装置や地形を扱わねばならない。
- ◆ リアルタイムに動作させなければならない。
- ◆ プレイヤーやゲーム状況の変化に即座に対応する必要がある。

という特徴があり、上記の例のような定式化できる問題のプランニングとは違い、導入のためには、

- ◆ プランニングの問題としてゲーム状況を定式化する
- ◆ ゲーム状況を適切に取得する
- ◆ 状況が変化すれば、適時プランニングをし直す

必要があります。こういった問題は、現実の工場などと比べてそれが含む情報量としては決して多くはないものの、「変化する状況の中で計画を建て続ける」という技術的な点では鋭く高度な問題に属します。ゲームへ応用するためには、単なる技術のスピノフではなく、それぞれのゲームにおいて応用の方法を工夫する必要があります。このテキストは、第 1 章では、プランニング技術とはどういうものであるかを概観し、第 2 章では実例として F.E.A.R、第 3 章ではクロムハウنز、そして第 4 章では様々な応用の方向を説明します。いろいろな応用の方向を俯瞰することで、実際の開発へのヒントが得て頂ければと思います。必ずしも隅々まで理解する必要はなく、あれこれと自分の開発するゲームを念頭に置きながら、時には読むのを忘れて新しいアイデアのメモを取りながら読んで頂ければと思います。

また、ゲーム A I におけるプランニング技術は、この数年で様々なゲームへの応用例が多く報告されています。Game Programming Gems や AI Game Programming Wisdom ([4]) といった書籍、インターネット上 ([5,6,7]) に多くの論文や記事があります (巻末に参考文献のリストがあります)。

「プランニング」の技術の基本はとてもシンプルです。このようなシンプルな技術がゲーム A I において強力な方法たり得ることは驚くべきことですが、同時に実装して初めて、「自分の行動を未来に向かって創造する能力」の新しい力を実感することができます。また、その能力は実装された NPC をみるととても自然に感じられます。人間にとってはあたりまえの能力を NPC に与えるのがプランニング技術であると言ってよいでしょう。

ただ「プランニング」の技術は 30 年以上の長い歴史を持っており ([1] 11 ~ 13 章)、調べようとする、たくさんの小道に迷い込んでしまう可能性があり、入り方によっては複雑に見えてしまいます。このテキストでは、ゲーム開発における応用を軸としてプランニング技術を紹介することで、実際のゲーム開発に繋がるプランニングの知識をまとめることを方針とします。

今回のテキストは情報量が多くなりましたが、シンプルなプランニング技術の仕組みに何度も立ち返りながら、最後まで読んで頂ければと思います。

[用語についての説明]

このテキストにおいては、COM、NPC、AI について以下のように言葉を使い分けます。

- ◆ COM、NPC ... ゲーム内の CPU が動かすキャラクター。この二つには区別はありません。クロムハウズでは COM という言葉で統一します。
- ◆ AI ... 人工知能技術。一般には、COM、NPC という意味で使われることもありますが、その使い方によってゲーム業界では「人工知能 = キャラクター」という固定観念が強くなる傾向があるので、敢えて区別することにします。人工知能は人工知能であって、たしかにその場合が多いとはいうものの、特にキャラクターの思考に限定されるべきものではありません。

また、単純反射型 AI と比較する場合には「ゴール指向型 AI」という言葉を使いますが、ゴール指向型のプランニングは「ゴール指向プランニング」という用語に統一しておきます。また、英語で検索される場合は、goal-oriented planning、goal-based planning、goal-driven planning と複数の表記があります。

- ◆ 階層型プランニング ... hierarchical planning。階層的プランニング、階層化プランニングと呼ばれることもあります。
- ◆ 人手によるプランニング ... 一般的な用語ではありません。

第1章 ゴール指向プランニング

この章では、第1節で「ゴール指向型」と第2節で「プランニング」という概念について説明し、第3節で「ゴール指向プランニング」という技術を順番に説明します。

第1節 ゴール指向型AIとは？

「知性が環境に投げ出された場合、どのようにして意志決定を行って行動するか」という最も単純な問から始めたいと思います。

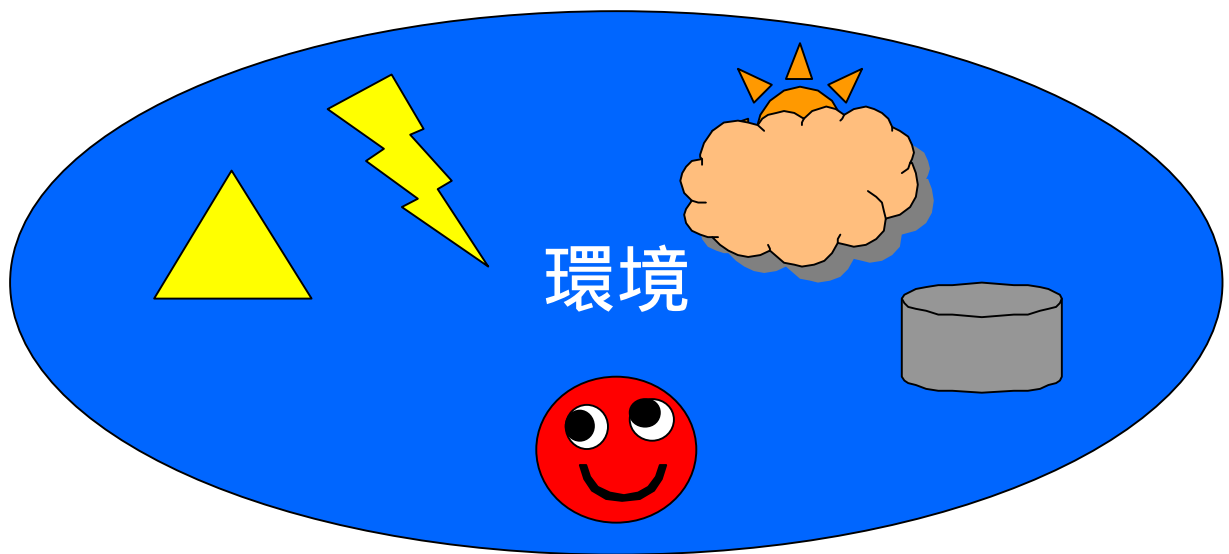


図 1-1 人工知能とは何か？ 人工知能を考える基本は、まずその知性がどういった世界（環境）に置かれているかである。知性と環境を切り離して考えることはできない。知性とは生物が環境に適応するために持った機関であって、環境との関係において初めて語ることができるものだからである。

一つの答えは、「周囲の状況を把握して、条件に応じて行動する」という方法です。「こういう場合ならこういう行動」というように条件に対して行動を対応させて意志決定を行う方法です。この形のAIを「単純反射型AI」([8])と言います。さて、もう一つの答えが、「周囲の状況から自分の達成すべき目標を決めて、それに向かって行動する」という方法です。この形のAIを「ゴール指向型AI」と言います。

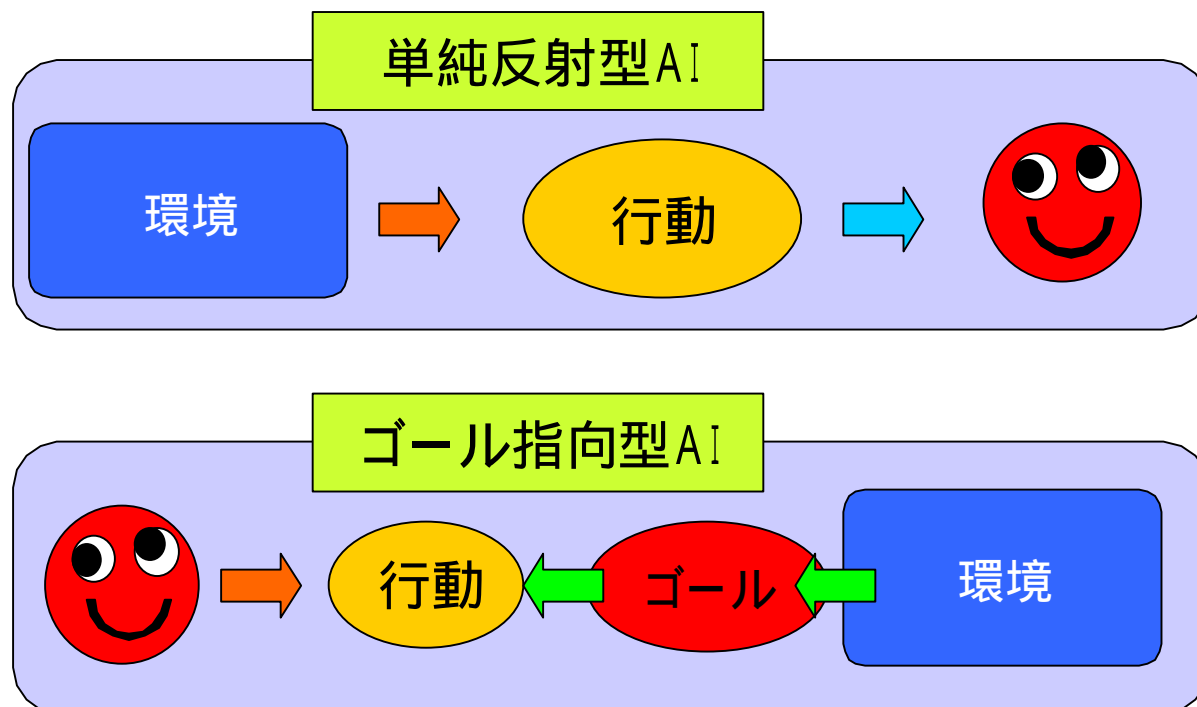


図 1-2 「単純反射型AI」と「ゴール指向型AI」の比較。行動を産み出す順序が全く違っていることに着目しよう。「単純反射型AI」では、決められた条件を満たしたときに、AI が決められた動作を行う。「ゴール指向型AI」では、まずゴールを決定し、そこから行動を決める。

人工知能では「ルール型AI」(rule-based AI)「知識型AI」(knowledge-based AI)「効用型AI」(efficient-based AI)など、そのAIが基本とする情報の形式、或いは意志決定の基準となる情報を「型」という言葉で表現します。「ルール型AI」とは「準備された、或いは学習したルールを基礎に行動を決定するAI」であり、「効用型AI」であれば「最も効用が高くなる行動を選択するAI」のことを言います。つまり、「ゴール指向型AI」とは「行動をするために、まずゴールを決定するAI」という意味になります。

第2節 プランニングとは？

この節では、プランニングについて説明します。「プランニング」とは、現在(初期状態)から達成目標(ゴール)に対して、必要な一連の行動を計画する技術です。

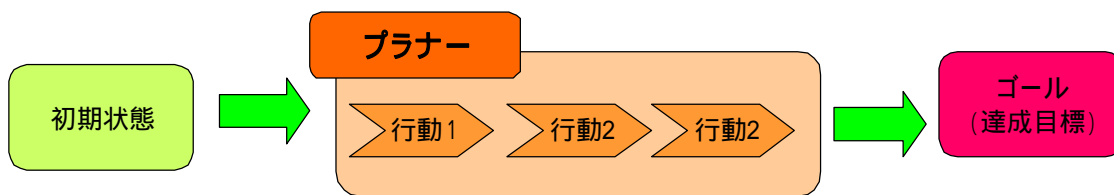


図 1-3 プランニングの基本概念を示した図。プランニング AI では、まずゴールが設定され（人によりにせよ、AI 自身の選択にせよ）、それに向かって現在の状態からの行動のステップを繋げて行く(chaining)ことで、行動のシーケンスを決める。この思考部分はプランを作るので「プランナー」と呼ばれる（[1,9]）。

プランニングには 3 つの基本概念

- | | | | |
|-------|-------|-----------------|-----------|
| (1) | ゴール | (goal state) | 達成目標 |
| (2) | 初期状態 | (initial state) | 現在の状態 |
| (3) | プランナー | (planner) | 行動を計画する思考 |

があります。この概念について簡単な例を通して説明します。

例えばロボットに積み木を並べ変えさせる、という問題を考えてみます。このロボットは、積み木を「降ろす」「載せる」という行動だけが実行可能です。(C,B,A) 並んだ積み木（初期状態）を、(A,C,B) のように並び替える（ゴール）問題を AI に与えることを考えます。

プランナーは、

1. A を降ろす。
2. B を降ろす。
3. C を A の上に載せる。
4. B を A の上に載せる。

というプランを考え（どうやって考えるかはひとまず置いて）、AI はそのプランに沿って行動することで目的を達成します。

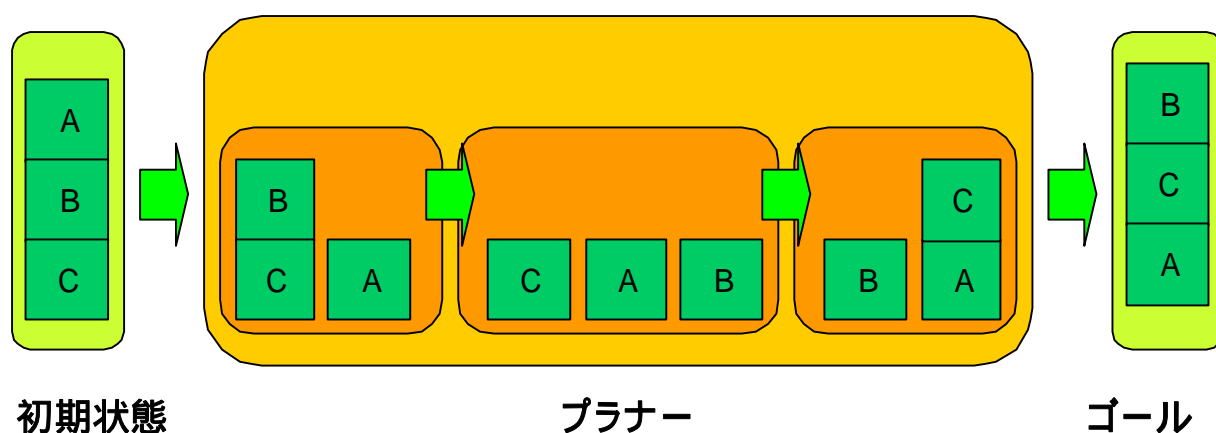


図 1-4 積み木を並べ替えるプランニング。ゴール (A,C,B) 初期状態(C,B,A)。プラナーはその間の行動のプランを探索する。ここでは、「検索ツリー」という方法によって、この行動の順序が決定される。「検索ツリー」は初期状態から、行える全ての行動を適用して行って、ゴールにたどりつくことができる手順を探す方法である。初期状態から可能なプランを探索して行く方法を「前向きプランニング」という ([1])

このように「プランニング」は、「初期状態」から「ゴール」へ向かって、「プラナー」がAIの実行すべき手順を作成し行動させる技術です。プラナーが最終的に構築するプランは、COM が直接実行できる行動から組み立てる必要があります。例えば、上記の例であれば、「ある積み木を移動せよ」というレベルの命令でなければなりません (こういった、プランの構成要素となる行為のことを「原始オペレーター (*primitive operator*)」と言います)。つまり、プランニングを実装する手順の概略は以下のようになります。

- 1) COM の行動の単位を決める (積み木を降ろしたり、載せたりする行動をロボットにできるようにする)
- 2) 指定された初期状態とゴールの間を、行動の単位の組み合わせとして解答できるプラナーを作る (積み木の例では、全ての組み合わせを検索することによって見つけることができます)
- 3) プラナーによるプランを順番に実行する (ロボットにプランに従って順番に行動させる)

第2章で F.E.A.R, 第3章でクロムハウズにおける実装を見ますが、基本はこの通りです。

しかし、プラナーをどのように実現する方法として

- 「連鎖」による方法 ([1] 1 1 章)
- 「人手」による方法 ([1] 1 3 章)

の2種類があり、前者は F.E.A.R で、後者は クロムハウズ で取られた方法です。以下に、この2つのプランニングの方法について、それぞれ説明します。

第3節 連鎖によるプランニング

ここでは、連鎖によるプランニングを説明します。積み木の例はとてもシンプルな例でした。説明のために、今度はもう少しゲーム的な状況を考えてみます。ゲームフィールドで「移動する」「飲む」という行動(*action*)を取ることができるCOMを考えます。こういった行動を単位とするプランニングを「アクションプランニング」と言います。

連鎖によるプランニングでは、プランニングのための行動を

(I)	前提条件	(<i>precondition</i>)	その行動が行えるための必要条件
(I I)	効果	(<i>effect</i>)	その行動の効果
(I I I)	振舞い	(<i>behavior</i>)	実際の行動

の3つによって記述します。

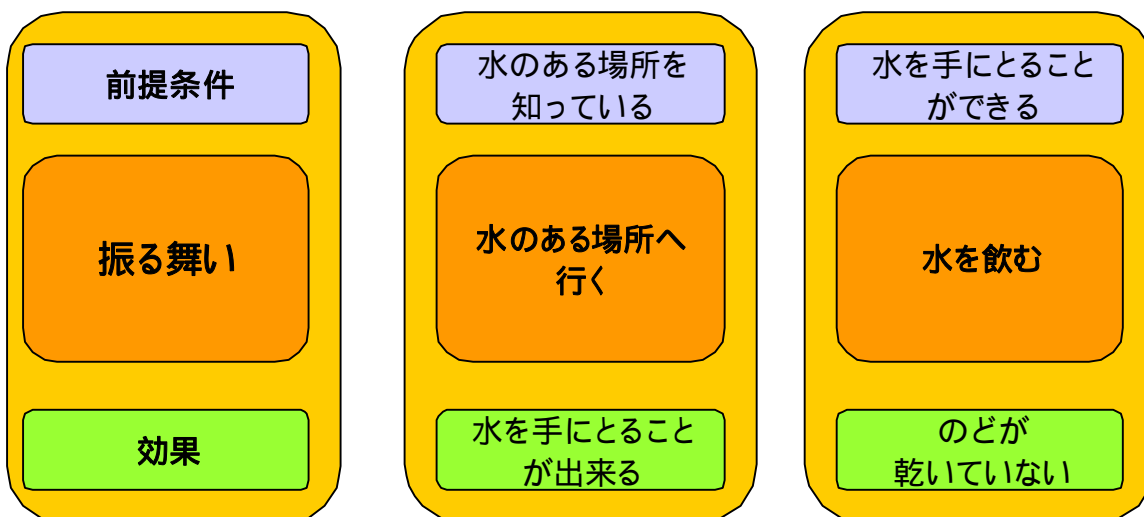


図 1-5 アクションプランニングのための行動の記述の仕方。アクションは3つの部分からなり、その行動がおこせるための「前提条件」(*precondition*)、そのアクションの「振る舞い」(*behavior*)、そして、それを行うことによる「効果」(*effect*)である。このため、アクションプランニングは時に、「振る舞いを基本とするプランニング」(*behavior-based planning*)と呼ばれることもある。

「連鎖によるプランニング」とは、この前提条件と効果が同じものを探して繋げて行くことで、行動を生成するプランニングのことを言います。プラナーが、効果と前提条件を使って行動をつなぐ過程を「連鎖」(*chaining*)と呼びます。

では、実際に上記で準備した行動によって、初期状態「水のある場所を知っていて、喉が渴いている」からゴール「水のある場所を知っているけど、喉が渴いていない」へ連鎖によるプランニングを行ってみたい。

まず、ゴールは「のどが渴いていない」ですから、プラナーはまず効果として「のどが渴いていない」を持つ行動を検索します。ここでは「水を飲む」という選択肢しかありませんので、この行動を選びます。次に、この行動の前提条件である「水を手にとることができる」を効果として持つ行動をプラナーは検索します。ここでは「水のある場所へ行く」という行動しかありませんので、これを選択します。この行動には「水のある場所を知っている」という前提条件がありますが、先に述べたとおり、ここでは、COM はあらかじめその情報を知っていますので、ここでプランニングは終了となります。実際は、一つの前提条件に対応して、効果を持つ行動が複数ありますから、一つの行動に対して導かれるプランは複数あります。しかし、ここでは説明のために簡単なケースを取り上げました。

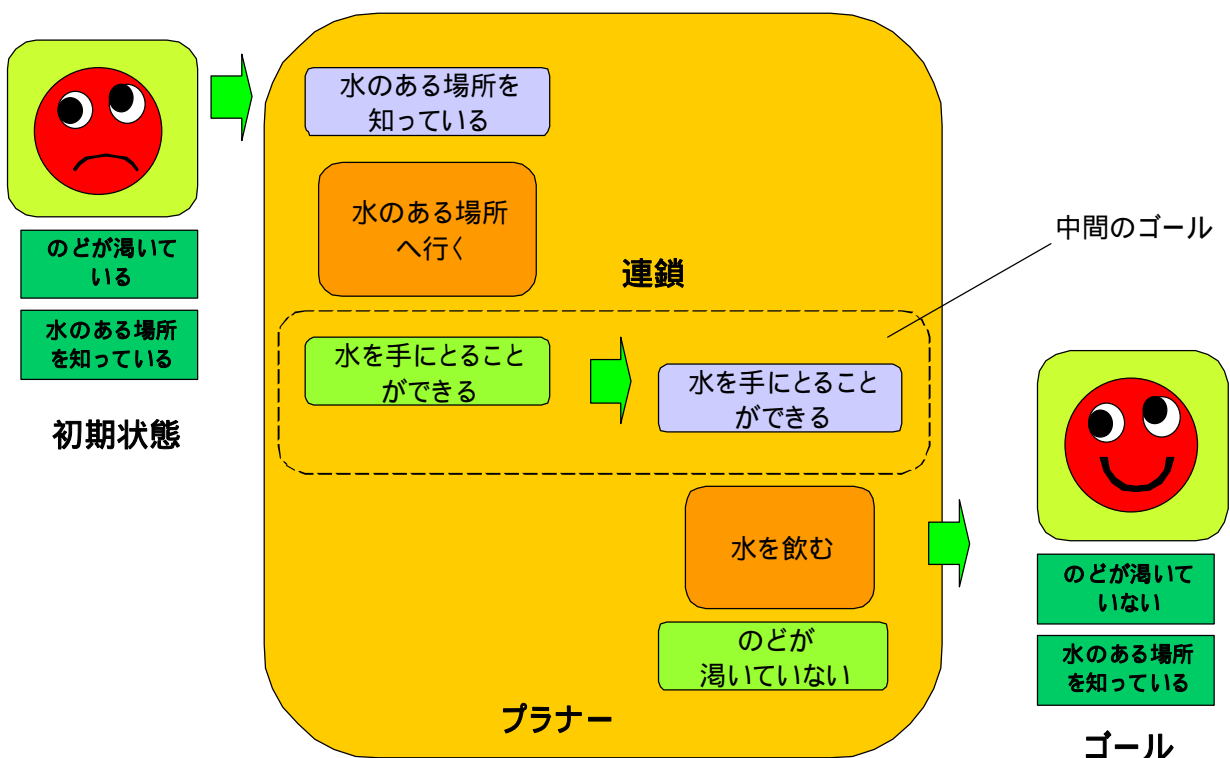


図 1-6 前提条件と効果による行動の連鎖のさせ方。これが「連鎖によるプランニング」の基本である。F.E.A.R では、この方法が、より多くの行動を連鎖する形で行われる。特に、ゴールから初期状態へたどって行くプランニングを「後ろ向きプランニング」(regression planning)と言う。第2章の F.E.A.R において、シンボルという形式で、このプランニングが機能するところを解説する。多くの場合、前向きプランニングも後ろ向きプランニングも可能なプランを全て検索する(全幅検索)の必要があるが、検索を効率的にするためには余計な場合をあらかじめ考えに入れしない(検索領域を刈る)工夫が必要となる。

第4節 連鎖によるプランニング(2) 複数の可能なプラン

前節では初期状態「水のある場所を知っていて、喉が渴いている」から、ゴール「水のある場所は知っているけど、喉が渴いていない」へ「連鎖」によって直線的に行動を組み立てることが出来ました。それは、

連鎖を行う場合に、プラナーが一つの行動しか見出すことが出来なかったからです。この節では、同じ効果を持つ複数の行動を用意することで、より複雑な形のプランニングを生成する過程を見てみることにします。

まず、先の単位行動のセットに「電話をかけて水を持って来てもらう」という行動を加えます(「前提条件」は「電話番号を知っている」、「効果」は「水が手にとれる」)。すると、以下の図のようにプラナーは二つのプランを組むことができます。ここで、どちらのプランを選ぶかはゲームに依存します。例えば、「コストがからない方を選ぶ」というプラナーなら(もし水を持って来てもらうのが有料なら)「水のある場所まで行く」の方を選択することになります。

さて簡単な例ですが、この例から

- (1) 一つのゴールに対するプランは複数生成される。
- (2) ゲーム状況に応じて可能なプランのうち一つを選ぶ必要がある。

ということが理解できます。さらに、プランニングには、

- (3) 初期状態が違えば、ゴールに対して、多様な行動のプランが選択肢として用意される。

という特徴があります。これを見るために、水のある場所を知らない場合には「水のある場所を人に聞くことが出来る」という行動ができることにして、上記と同じ条件で連鎖によるプランニングを実行してみます。すると図 8 のように、初期状態と選択する行動によって複数のプランが生成されることがわかります。

このように「連鎖によるプランニング」は単一のゴールに対して状況に応じた複数のプランを生成し、COMの行動を一連の時間に展開すると共に、複数のプランを生成することにより行動に本質的な多様性を与えます。この利点が、ゲームにおいてプランニングが有望視される理由の大きな理由の一つです。第 2 章で、F.E.A.R を題材に、ゲームにおける実際の実装を説明します。

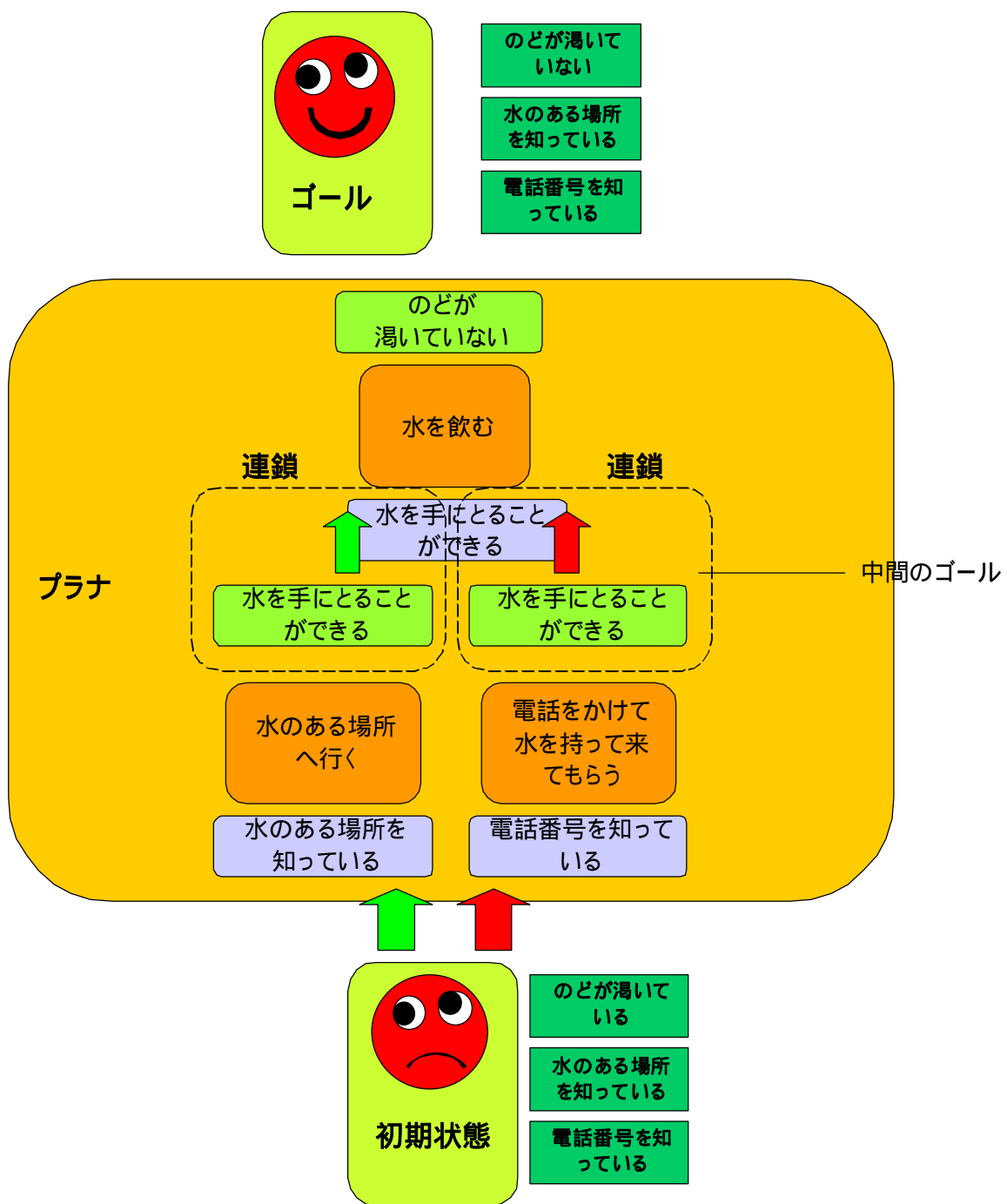


図 1-7 複数のプランが可能な場合がある。状況や条件によって自由に任意のプランを取ればよい。実は、ここにはプランニングの重要な長所が現れている。同一のゴールに対して複数のプランが可能であることは、COMに行動の多様性を与える、という点である。単純反射型の AI では「ある条件が満たされれば、ある行動を取る」というパターンが決定される。「半径 10 m に敵が入れば、敵を攻撃する」などである。一方、プランニングでは「敵を攻撃する」をゴールとして複数の攻撃プランが生成され選択することが可能になる。

図 1-8 様々な初期状態から一つのゴールへ向かって複数のプランが組み立てられる。プランの選択は、ゲームに依存する。例えば、NPC が怪我をしているなら「歩く」より「電話する」方がよいし、費用を抑えるなら、電話をかけて持って来てもらうより「歩く」方を選択する、などである、キャラクターの状態によって行動が変更されるためパターンに陥ることもない。

第5節 人手によるプランニング

「連鎖によるプランニング」は前提条件と効果をプラナーが連鎖することで、自動的に行動を生成してくれました。しかし、必ずしも現実で直面する複雑な問題に対し、前提条件と効果を簡単な形で記述できるわけではありません。そこで、実的な応用として「行動のつなぎ方をあらかじめ人が全て定義しておく」というアプローチが存在します。これを「人手によるプランニング」と呼びます[1]。

前節と同じ設定のもとで「人手によるプランニング」を説明します。「水を飲む」という行動の定義に、

- 「水のある場所を知っていれば（条件） 水のある場所へ行く（行動呼び出し）」
- 「水のある場所を知らなければ（条件） 電話をかけて水を持って来てもらう（行動呼び出し）」

と記述しておきます。すると「水を飲む」というゴールを選択した場合には、現在の条件に応じて、何れかの行動が呼び出されることになります。さらに、呼び出された行動にも同様に、条件に応じて呼び出す行動を定義しておくと、帰納的に行動が生成されて行きます。

特に、この方法は、ゴールを階層に分けて上から下へ向けて帰納的に呼び出す「階層型プランニング」（hierarchical planning）と関係します（[10] 第9章、[1] 第12章）。詳しくはクロムハウズの実装を通して第3章で説明します。

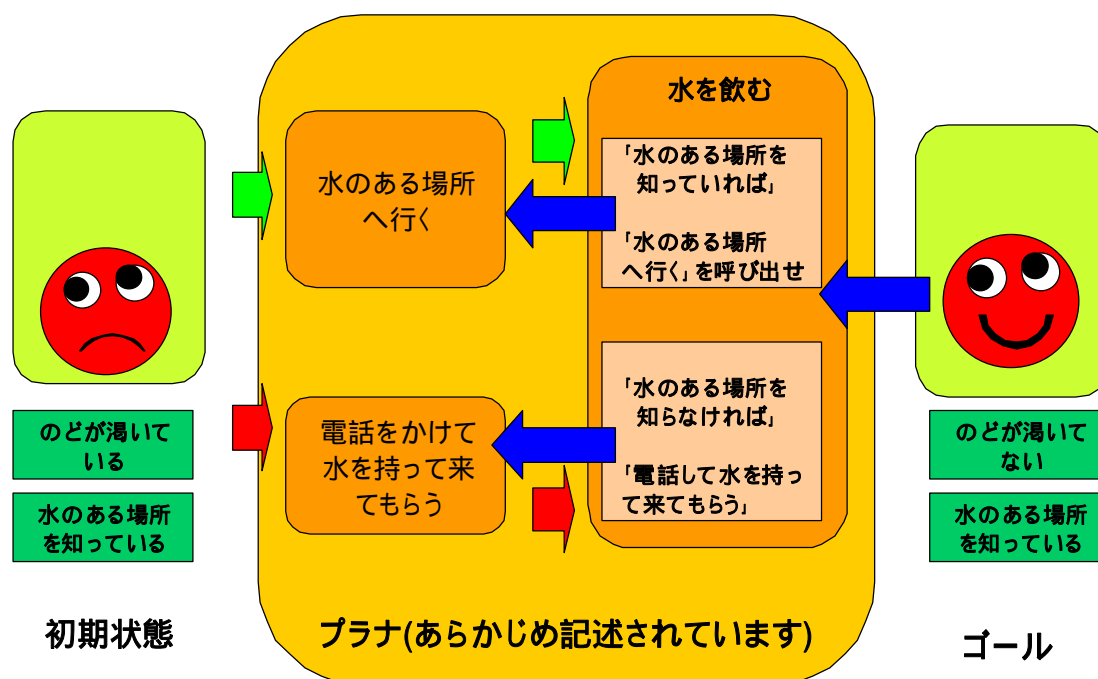


図 1-9 人手によるプランニング。用意される行動それぞれに、どの行動を呼び出すことができるか、という情報が含まれている。例えば、上の例では「水を飲む」という行動は「水がてもとにない」場合には「水のある場所へ行く」というルーチンが設定されている。そうやってあらかじめ決められたルールで行動が連鎖して行く。クロムハウズはこの方法を取っている。

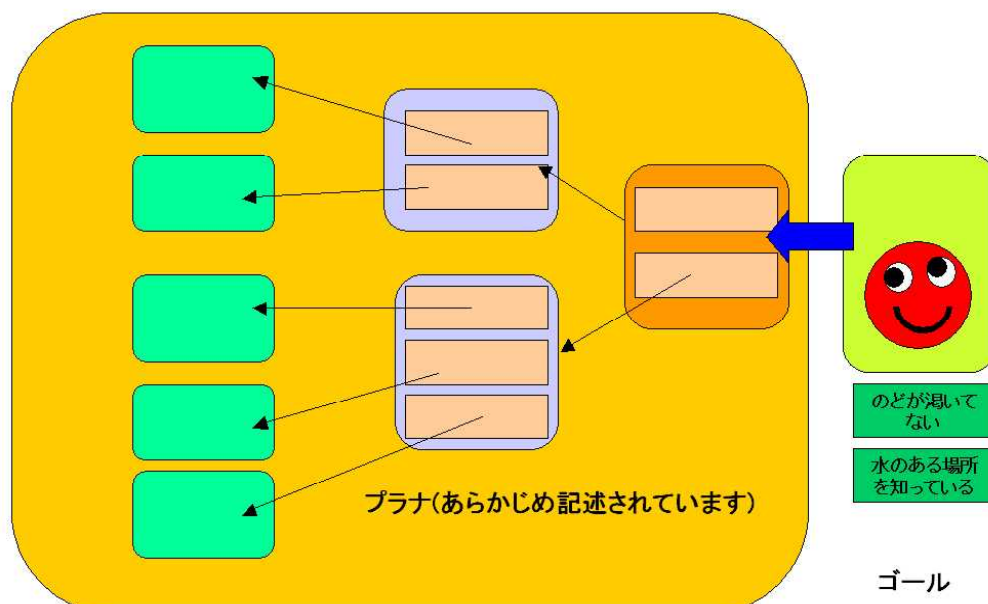


図 1-10 人手によるプランニングは、再帰的に行動を呼び出して行くので、条件によって一連の行動は決定されるが、条件分岐の数だけ行動の多様性がある。

第6節 ゴール指向型プランニングとは？

第3, 4節で「連鎖によるプランニング」、第5節で「人手によるプランニング」を簡単に説明して来ました。ここで、「ゴール指向型A I」のフレームの中でプランニングを動作させる「ゴール指向型プランニング」について説明します。

「ゴール指向型プランニング」を思考として持つA Iは複数のゴールの中からゴールを決定し、そのゴールに対してプランニングを行います。ゴールの選択は、

プレイヤーが指定して命令を効かせる。

自律型A I（自分で判断するA I）においては自ら選択する。

などのパターンが考えられます。ゴール指向型A Iにおいては「ゴールを選択する」過程が意思決定過程となります。そして、そのゴールを達成する行動を知るためにプランニングを用います。

このシステムにはまず「複数のゴール」という自由度があり、そして、その一つ一つのゴールに対して複数の行動の計画があり、これによって、C O Mが実行できる行動の幅が多様で大きな広がりを持つことになります。この可能性こそが、これからのゲームで利用できるA Iの可能性の広がりを示しています。

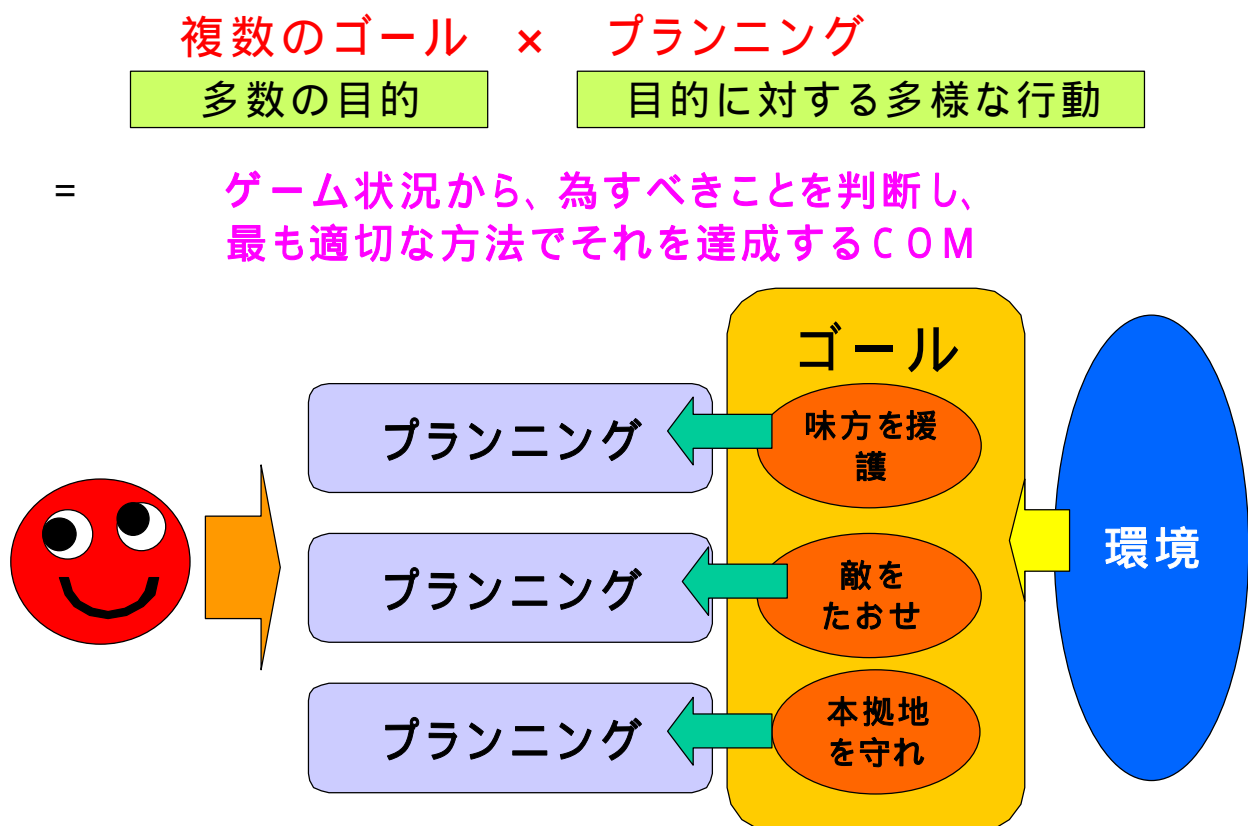


図 1-11 「ゴール指向型プランニング」を表現した図。候補となるゴールからC O Mは自ら一つのゴール選択し、そのゴールに対してプランニングをして実行する。

第7節 まとめ

この章では、本テキストの主題である「ゴール指向」という概念と、「プランニング」という技術について簡単に説明しました。

「ゴール指向型 AI」とは「自分の属する世界の中で、まずゴールを設定することから行動を始める AI」のことです。「ゴール指向」は一見、自明な概念ですが、単純反射型の AI と並んで重要なフレームです。こういった足元の部分を疎かにすると、いつの間にか狭い考えに捉われてしまいますので、簡単でも「単純反射型」と「ゴール指向型」があるんだと覚えておくと、より自由に人工知能の発想を得ることができます。

「プランニング」は選択したゴールへ向けて初期状態から必要な行動を積み上げて行く技術です。簡単な例で紹介しましたが、大切なことは、そういったプランの創作が、AI の思考によって自動的に行われているという点です。この点が、AI が自ら時間を渡って行く能力を与える強力な方法となります。また、第1回のセミナーのテーマである「世界表現」を用いたフィールドの自由な移動と合わせると、時間と空間を知的に渡って行く能力を身につけさせることが出来、自律型の AI へ向けて重要な基礎となります。

「ゴール指向プランニング」は「ゴール指向型 AI」にプランニングを応用した技術で、AI が選択したゴールに向かってプランニングを行います。つまり、ゴール指向によって「何をするか」が決まり、プランニングによって「如何に」が決まります。

ゴールの選択は、特にゲームにおいては、人間が行う場合と、自律型 AI として AI が自ら選択する場合が考えられます。前者は、例えば、プレイヤーが味方の AI に指令（ゴール）を与えて、後は AI のプランニング能力に任せる、という場合に应用できます。これは、実際「クロムハウন্ズ」のオフラインモードのコマンダーミッション（司令官のミッション）に应用されています。それ以外にも多くの応用が見つかるでしょう。後者は、AI が自らゴールを決定します。自分で選択したゴールに対して、自分で行動のプランを創造し、自分の足と手で実行するわけですから、AI はゲーム世界で意志を持つかのように行動させることが出来ます。

基礎的な事項を説明し終えたので、第2、3章では、実際のゲームにおける実装を説明します。

説明は情報量が多くやや煩雑になりますが、基本システムは、ここで説明したものが全てですので、ややこしくなったら、いつでも、この章に立ち戻ってください。

[第1章おわり]

第2章 F.E.A.R.におけるゴール指向アクションプランニング

この章では、第1章で説明した「連鎖のよるプランニング」を使ったゴール指向AIの例として、F.E.A.R.における「ゴール指向プランニング」の応用について解説します。この章で引用する全ての文献はJeff Orkinのサイト([11])にあり、解説内容はそこにある数編の論文、記事を総合したものになっています。

F.E.A.R.では、実際のキャラクターのアクションを基本としてプランを組みます。特にこれを「ゴール指向アクションプランニング」(*Goal-oriented action planning, GOAP*)と言います([12,13,14])。ここでF.E.A.R.のAIを解説するにあたって、開発プロダクションであるMonolith Productionが、F.E.A.R.に至るまでにどのようにAIのシステムを発展させて来たかを、過去のタイトルと比較しつつ解説します。そうすることで、一つのプロダクションが人工知能技術をどのように発展して行ったかを理解することが出来ます。それは、これからゲームAIを発展させて行く上でたいへん参考になると思われます。後半では、F.E.A.R.のAIのシステム全体を説明します。

第1節 F.E.A.R.のAIが目指したもの ~Shogo、NOLF2、そして、F.E.A.R.~

F.E.A.R.はFPS(First Person Shooter)のゲームです。一人称視点で、銃を撃って敵を倒して行くゲームで、FPSは日本のRPGのように欧米で最も人気の高いジャンルであり、多くの熟練したユーザーを獲得し育てて来ました。それゆえに、目の肥えたこだわりの高いユーザーが多く、その期待と要求に応えるため。他のタイトルと差別化したAIを開発をするために、それぞれのProductionが力を入れています。Monolith Productionはこのような状況の米で一貫してFPSを作りながら、一作一作そのゲームAIを発展させて来ました。

表 2-1 Monolith Productionのゲーム開発とAIの歴史。常に前作の問題点を見抜き、それに対して適切な人工知能技術をもって克服して来た。

			
目指すAI	プレイヤーに気付いて攻撃	目的(ゴール)を持って行動	目的(ゴール)と達成の仕方(プラン)を考えて行動
人工知能タイプ	単純反射型AI	ゴール指向型FSM エージェント	ゴール指向型プランニング エージェント
出版年	1998	2002	2004

1998年に発表した Shogo は、単純反射型の AI [8])で、プレイヤーの行動に反応します。2002年の NOLF2 ([15,16]) は、周囲のオブジェクトを使って行動すること、自分のゴールを決定して行動することが NPC の特徴となっています。これは「何をするか」を自分で決定する「ゴール指向型 AI」です。さて、NOLF2 はおそらく、よく出来た AI なのですが、ここからもう一段階ステップアップして、2004年に F.E.A.R を発表しました。これまでに培って来た「知識表現」「ゴール指向型 AI」の技術の上に、洗練された形で「プランニング」技術を導入することに成功し、自分の行動について自分自身で「何をどのように」を決定する「ゴール指向型プランニング」AI を作り上げました。一作ごとに反省し問題を克服して来た努力の甲斐あって、この AI は、ユーザー、開発者ともに高い評価を受けました。開発者にとってたいへんやりがいのある仕事であったに違いありません。

さて、もう少し詳しく述べると、特に F.E.A.R においては、

- 周囲の環境の情報を自ら感知し、そこから自分の思考で判断して行動する。
- レベルデザインにおけるオブジェクトを自発的に使用して行動する。
- 何をすべきか、を判断するだけでなく、如何にすべきか、も思考する。

という点に特徴がありました。このような機能は、

- ◆ エージェント技術
- ◆ 知識表現
- ◆ 行動の決定

という3つの技術的な柱に支えられて初めて可能になります。ここからは、この3点に着目して、NOLF2 と F.E.A.R の AI を明確に比較しながら、F.E.A.R が達成した「ゴール指向型アクションプランニング (GOAP)」について解説します。

表 2-2 NOLF2 と F.E.A.R の AI の比較

	NOLF2		F.E.A.R
エージェント構造	ゴール指向型	共通	ゴール指向型
知識表現	人間的な概念	改善	統一知識形式 シンボル
行動決定	有限状態マシン	改善	プランニング

第1項 ゴール指向型エージェント

まず出発点として「自律型エージェント」から説明します。「感覚」 意思決定 行動の3つの要素を持つAIを「自律型エージェント」と言います。それぞれ、センサー、意志決定、エフェクターと呼ばれます。

- | | | |
|-------|---------------------------------|---------------------------------|
| (I) | センサー (<i>sensor</i>) | 周囲の環境から情報を集める感覚を持つこと |
| (II) | 意思決定、(<i>decision-making</i>) | 集めた情報(と記憶)を基に、次の行動を決定すること |
| (III) | エフェクター (<i>effector</i>) | その世界で環境に影響を及ぼすことができる体を持ち行動できること |

生き物の知能モデルの、或いは、エージェントという名前の通り「人の代理を出来る能力を持つ」AI と言えはわかりやすいかと思います。

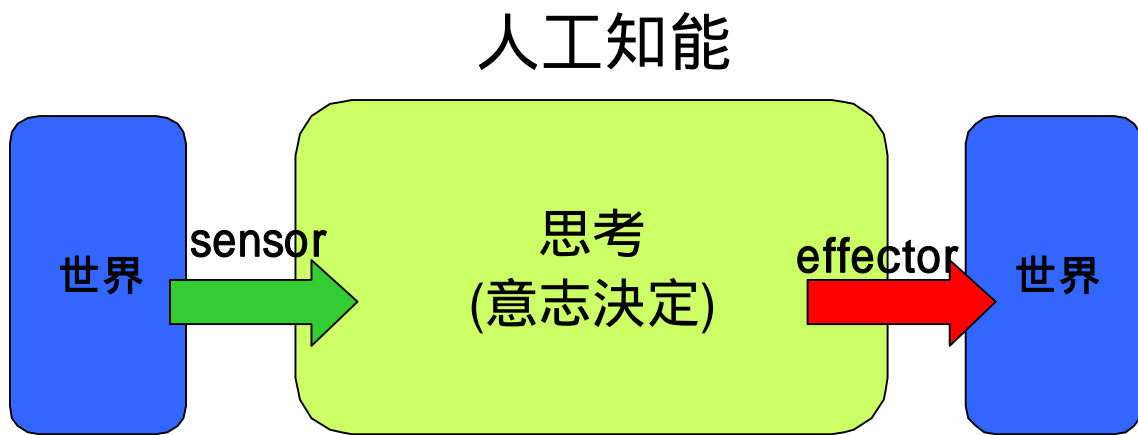


図 2-1 エージェントのフレーム([17])。エージェントとは、自分の属する世界に対して、「 感覚(sensor)を持って情報を集める その情報をもとに、自らの行動を決定する思考を持つ その決定をもとに、世界に対して行動をし影響を及ぼす」という3条件を満たすAIである。FPSにおいて「キャラクターをエージェントとして実装する」という方向が主流になりつつある。FPSにおけるNPCはまさにプレイヤーの代わりであるから、その方法が適しているのだろう。エージェントは一つの大きなフレームであり、上記3点をどう工夫して行くかで知性の決定される。ゲームにおけるエージェントには、それこそ膨大な量の文献があり、そういった文献を参考に自分の開発するゲームに合ったエージェントを構築することが出来る。

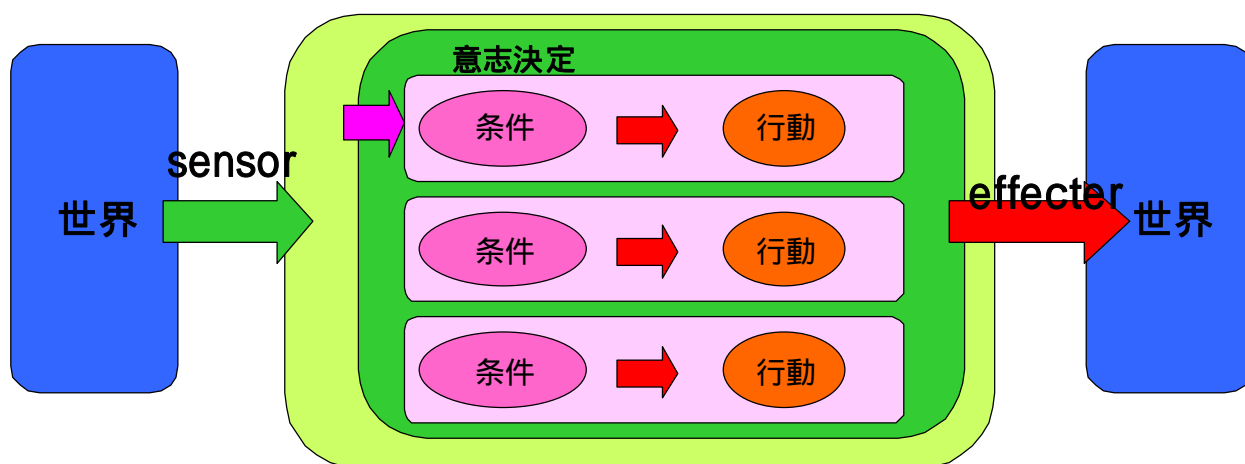
エージェントは、AIを構築するときの一つの大きなフレームです。感覚、意思決定、行動という流れを構築することで知性を構築します。それぞれの部分をどのように工夫して行くかでそのエージェントの知能の形が決定されます。そのためには、自分で考えてみることも、様々なエージェントがどのような構造になっているかを研究してみることが得策です。NOLF2、F.E.A.R、クロムハウズをエージェントという視点から順番に説明して行きますので、その比較から美点や欠点を見抜いて、自分の開発するAIの設計図のスケッチを作る参考にして頂ければと思います。

Shogo は「単純反射型 A I」であり、NOLF2 と F.E.A.R は「ゴール指向型エージェント」です。単純反射型 A I は現在の世界の情報から条件に適合する行動を選びますが、ゴール指向型エージェントでは現在の

状態から、まず何をすべきかを判断して、そこから行動を決定しています。

「単純反射型 A I」はゲームでは準備されたルールに従って行動します。このようなエージェントを「ルールを基本とするエージェント」(Rule-based AI)と言います。ルールを基本とするエージェントでは、どのように優れたルールの集合を用意できるかに、A I の出来が左右されます。一方、ゴール指向型 A I は、例えばゴールに点数をつける方法(評価値による方法) 或いは、推論によって用意された複数のゴールから一つのゴールを決定します。そして、そのゴールを達成するために、現在の状況から行動を構築して行きます。ゴール指向型エージェントでは、適切なゴールを選択できる能力と、行動のデザインの能力が出来を左右します。ゲーム AI を開発する前に「このゲームに必要な A I は単純反射型か、ゴール指向型か」を考えることは、ゲームが要求する A I の形を捉える出発点となります。

単純反射型エージェント構造図



ゴール指向型エージェント構造図

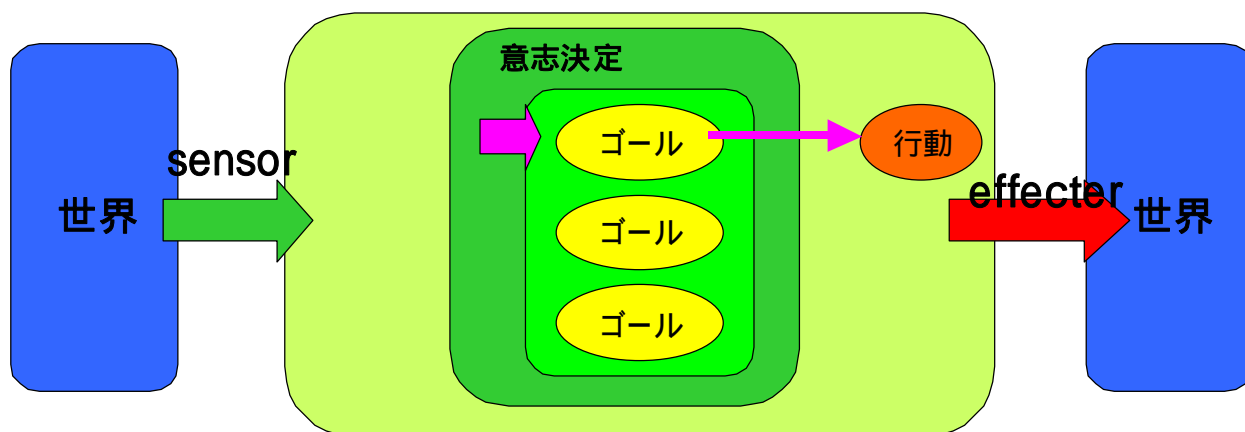


図 2-2 単純反射型エージェントとゴール指向エージェントの比較。前者は、たくさんのルール(条件、行動)の集合からなる。行動は、現在のゲーム状態に対応する条件によって決定される。後者では、まず、ゲーム状態からゴールが選択される。選択されたゴールとゲーム状態から、最適な行動が決定される。この「ゴ

ール 行動」という部分は、NOLF2 では、有限状態機械(FSM)、F.E.A.R では、プランニングとなる。

第2項 知識表現、世界表現

知識表現、世界表現は第1回のセミナーのテーマでもあったのですが、ここでは復習をかねて説明します。

AI の優秀さの一つの指標として、「周囲の環境をどれだけの確に捉え、それに対応した行動を取るか」というポイントがあります。この点は特にプレイヤーの相手をするゲームAIにとっては最も重要なものです。しかし、そもそも「環境」と一言と言ってもゲームフィールドにはたくさんのオブジェクトがあり、プレイヤーがあり、それが時間的に変化します。また、たいていのゲームでは、多数のフィールドが用意されているので、COM が処理するべき「環境」の種類は膨大なものになります。また、目に見えているもの意外にも「関係」「所有」と言った目に見えない抽象的な情報があり、高度なAIになればなるほど、このような情報を処理する必要に迫られます。そういったAIが属する世界の知識をデータとして表現することを、「知識表現」と言います。これがAIの基礎になります。「知識表現」の上に、高度な思考やアルゴリズムが環境に対応した意味を持ち、AIが「住む世界」に沿った形の知能を製作することが可能になるのです。特に「世界表現」は「知識表現」の一つであり、フィールド全体に対する情報のことです。パスデータや、ウェイポイント毎ばデータなど、地形をうまく使いながら移動するために必要な表現です。NOLF2 と F.E.A.R においても、知識表現、世界表現は大きな役割を果たします。ゴール指向型の思考もこの表現の上に展開されます。

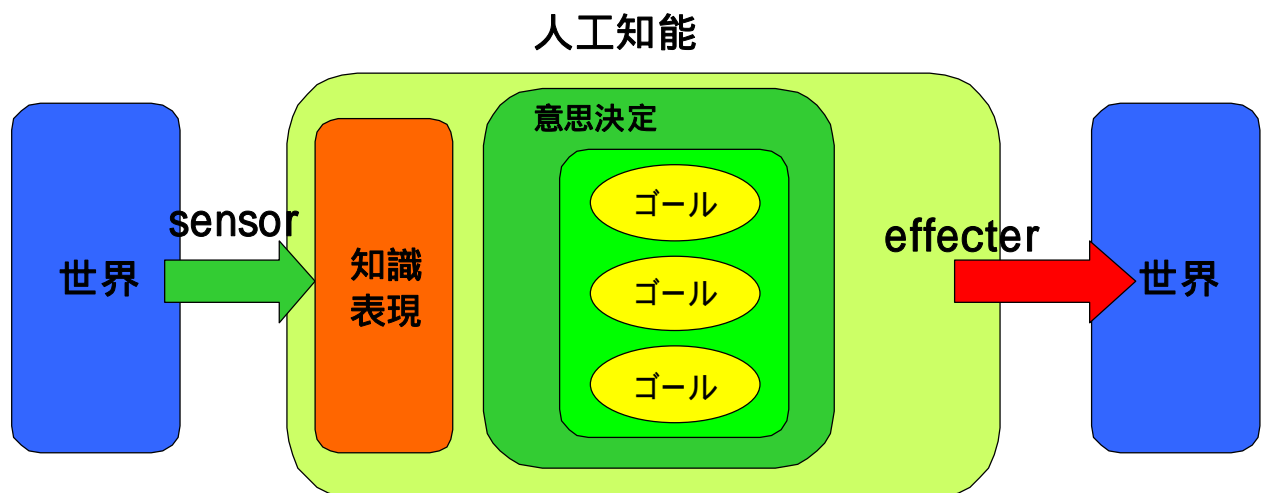


図 2-3 ゲーム世界の知識表現。「人工知能が自ら世界を解釈する能力を持つ」という機能はたいへん難しい技術で、アカデミックの分野でも簡単な場合にしか実現していない。その代わり、人工知能には「知識表現」という方法がある。知識表現とは、人工知能が世界を認識するために、人間が、解釈するべき形式を指定したり、或いは、解釈するべきようにデータを作ってしまうことである。例えば、COM に広いフィールドを自由に歩かせたい。そのためにウェイポイントを用意します。さらに、ウェイポイントに様々な情報を埋めることで、COM はその情報を使って地形を使ってだんだんと賢い行動を取ることが取らせることができる。第1回のセミナーはこの技術を解説しているので参照されたい〔2〕。

NOLF2 は、特に知識表現に力を入れたタイトルであり、ゲームフィールドに存在する各オブジェクトについて「7つのカテゴリー」からなる知識表現を行っています ([15,18])。あるNPCが机に座ると、その机はNPCが「所有」しており、他のNPCが使用することは出来なくなります。また、トイレと洗面台は、その使用順序「依存」が決められているので、洗面台へ行ってからトイレへ行くという行動もしません。また「護衛所」は兵士が「責任」を持つ場所で、他の種類のNPCが近付くことはできないようになっています。このように、各オブジェクトに情報を仕込んでおくことで、NOLF2 のNPCたちが、ゲーム世界で「それぞれのオブジェクトが何であるかを認識して振舞っている」ように見せかけることが出来ます。これがゲームにおける知識表現の効用です。

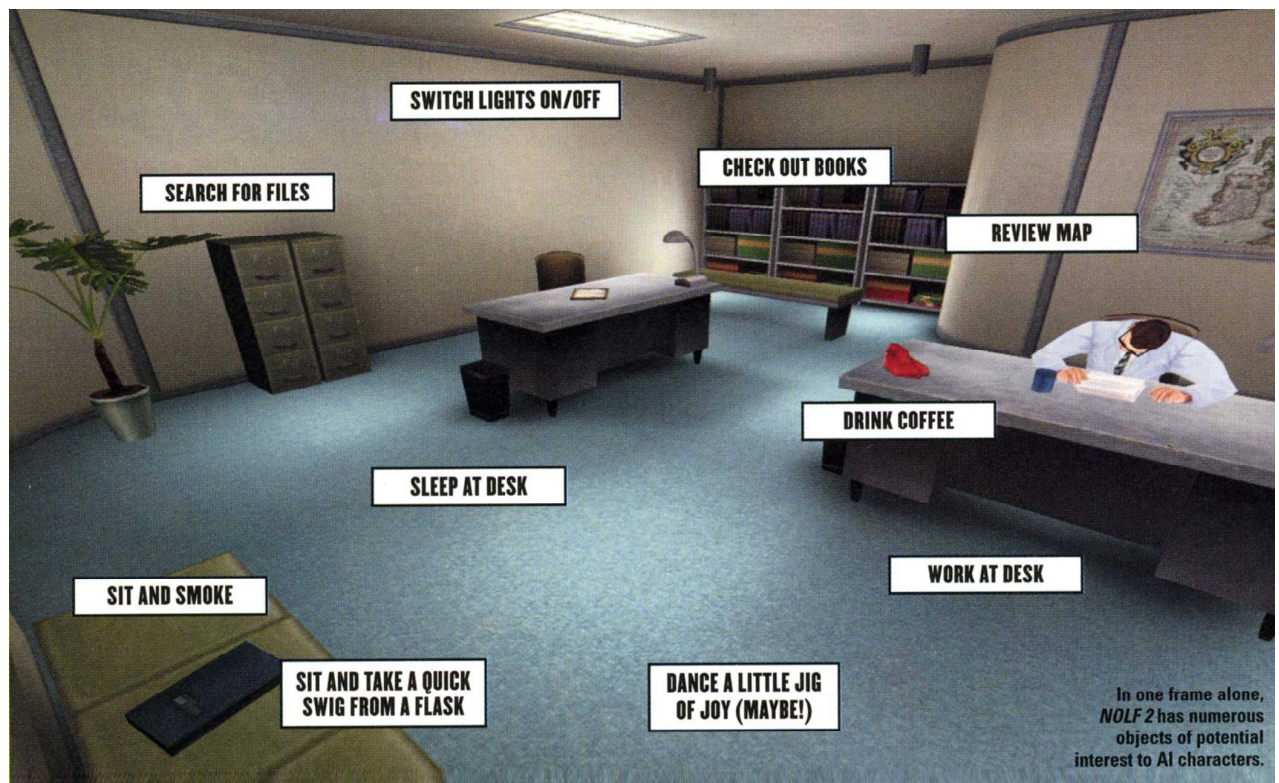


図 2-4 各オブジェクトに知識表現が用意されている NOLF2 のゲーム画面。

(Figure from No One Lives Forever 2, Monolith Production, <http://www.lith.com> Sierra <http://nolf2.sierra.com/>)

表 2-3 NOLF2 における知識表現。特に「依存」はプランニングシステムを持たないNOLFについて行動の順序を規定する役割を果たすことに注目したい。この考えは、第4章で解説する「依存グラフ」と同様のアイデアである。

項目	説明	例
所有権 (Ownership)	オブジェクトが今、誰に使用されているか	机に誰が座っているか、という情報を持つことで、同じ机に他のキャラクターが座ろうとしない。
依存 (Dependency)	オブジェクトの使用の順序を規定	洗面台はトイレの後に使う。
責任 (Responsibility)	オブジェクトをどの種類のCOMが使用する	観測機器は科学者、護衛所の周囲は兵士が担当。

	かを規定	
優先度(Priority)	ゴールの優先順位	部屋の中でデスクワークをする、部屋の中の兵士の死体を片付ける、は後者が先でなければならない。
認識状態(State of Consciousness)	COM が何に気付いているか	敵兵が死んでいるのに気付く。
予測状態 (Expected State)	状況予測	その場所に敵兵がいるはずである。
関連性(Relevance)	サウンドや効果など他のデータとの関連性	ストーリーの中でのキャラクターの役割

- 「所有権」は、例えば机に NPC (A) が座ると、机は A の所有となります。そこで、他の NPC がそこに座ることを避けます。
- 「依存」は、例えば「トイレへ行って」「手を洗う」という動作が逆にならないように、オブジェクトに使用する順序をつけておきます。
- 「責任」は、そのキャラクターが使用できるオブジェクトや場所を限定します。兵士が顕微鏡を使用したり、科学者が、駐在所の周りを歩かせないようにします。
- 「優先度」は、ゴールの優先度を決めることで、例えば、机に座っているときに敵が攻めて来た場合、「コーヒーを飲む」「机の下に隠れて攻撃」が選択可能ですが、優先度をつけて、敵の前でコーヒーを悠長に飲まないようにします。
- 「認識状態」は、COM が感覚で得た事象から、記憶として蓄積する情報の形で、この二つが COM の記憶として設定されます。つまり、現在の状況と、例えば、敵がこのドアの向こうにいるであろうという予測から、自分の行動を決定（ゴールを決定）します。

このように、NOLF 2 では人間らしい AI を目指して、人間が日常で用いる概念を知識表現として用意しています。F.E.A.R. における知識表現では、このアプローチをより精密化させた「シンボルによる表現」を構築しています。

第3項 行動の決定

ゴール指向型エージェントは「ゴールから行動を決める」と説明しました。NOLF2 において、この行動の決定は有限状態機械（以下 F S M ）によって行われます。F S M とは、おそらく現在のシステムで最もよく使われている A I のシステムであり、単位となる行動を遷移条件によってネットワーク状にしたものです。NOLF2 では、「Work」「Death」「Stunned」「AttackFromCover」などのゴールが用意され、各ゴールの下に F S M のシステムがぶらさがり形で用意されています。つまり、NOLF 2 の A I は、

- (1) ゴールを決定する。
- (2) そのゴールの F S M の遷移に従って行動する。

(F S M に不慣れな方は、一つの駒が行動の書かれたマスの上を、条件に従ってすごろくの駒のようにマスを移動する、止まったらマスの行動を A I が実行する、とイメージしてください)

という順番で、行動します。この方法の利点と欠点を考えてみましょう。

F S Mのよい点は、行動と遷移条件を指定することで、完全にA Iの行動を人の手で決定できることです。行動を増やしたければ、F S Mの中の適切な位置に行動を置いて、他の行動と遷移条件で結びます。すると、A Iは必ずこの行動ネットワークに従いますから、A Iを作り込みたければ、行動を増やしこのネットワークを精緻化すればよいのです。特に、N O L F 2ではゴール毎にF S Mが作れますから、ゴールの数だけのF S Mを作りこんで行くことが出来ます。実際、この作りこみによって、N O L F 2のN P Cたちは、どんどんと複雑な行動を取ることが出来るようになりました。

しかし、このシステムには、一方で以下のような3つの欠点があります。

ゴールを増やすたびに、F S Mを新しく作らねばならない。

新しい行動を作ったら、その行動を各ゴールのF S Mの適切な位置に置く必要がある。

後になればなるほど複雑になる。

以上が利点と欠点です。それほど、複雑でないゲームなら、このシステムが十分に適切なゲームが見つかるでしょう。しかし、N O L F 2は、このシステムを適用するには、少々複雑過ぎるゲームであったようです。まず、職業によって行動パターンを変えたい場合、F S Mは一から作り直さなければなりません。また、開発の後半で、開発チームは「暗い部屋に入ったらスイッチを押して明るくする」という行動を取らせる必要に迫られました。そして、全てのF S Mについて、その行動を適切な位置に組み込んで行きました。製作と調整がよほど難しかったと見えて、N O L F 2の製作が終わったときには次の新しいシステムへ向けての模索が始まりました。そこで採用されたのがプランニング技術です。

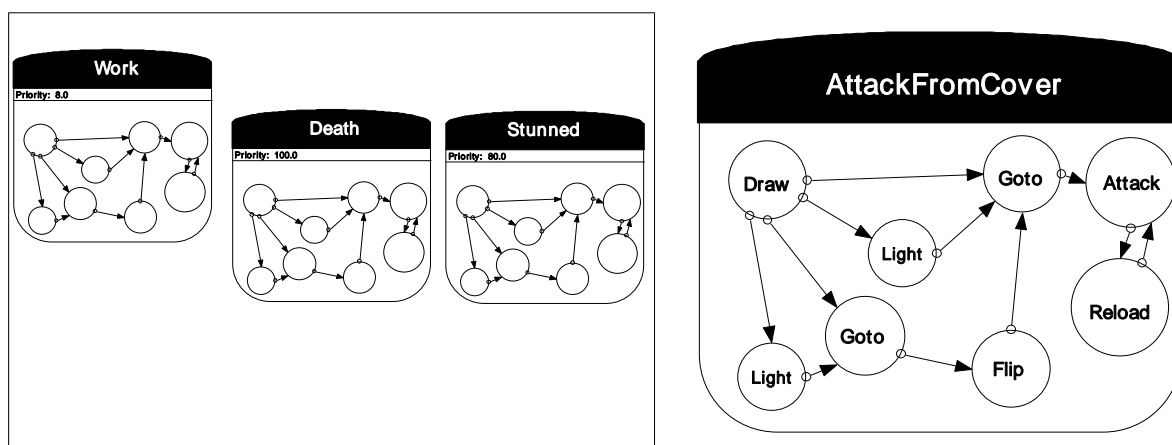


図 2 -5 NOLF2 の「各ゴールの下に用意された F S M 」概念図。「Work」「Death」「Stunned」「AttackFromCover」というゴールの下に、それぞれ FSM が設計されている。ここでは、参考に3つのゴールだけを抜き出して図示されている ([12])

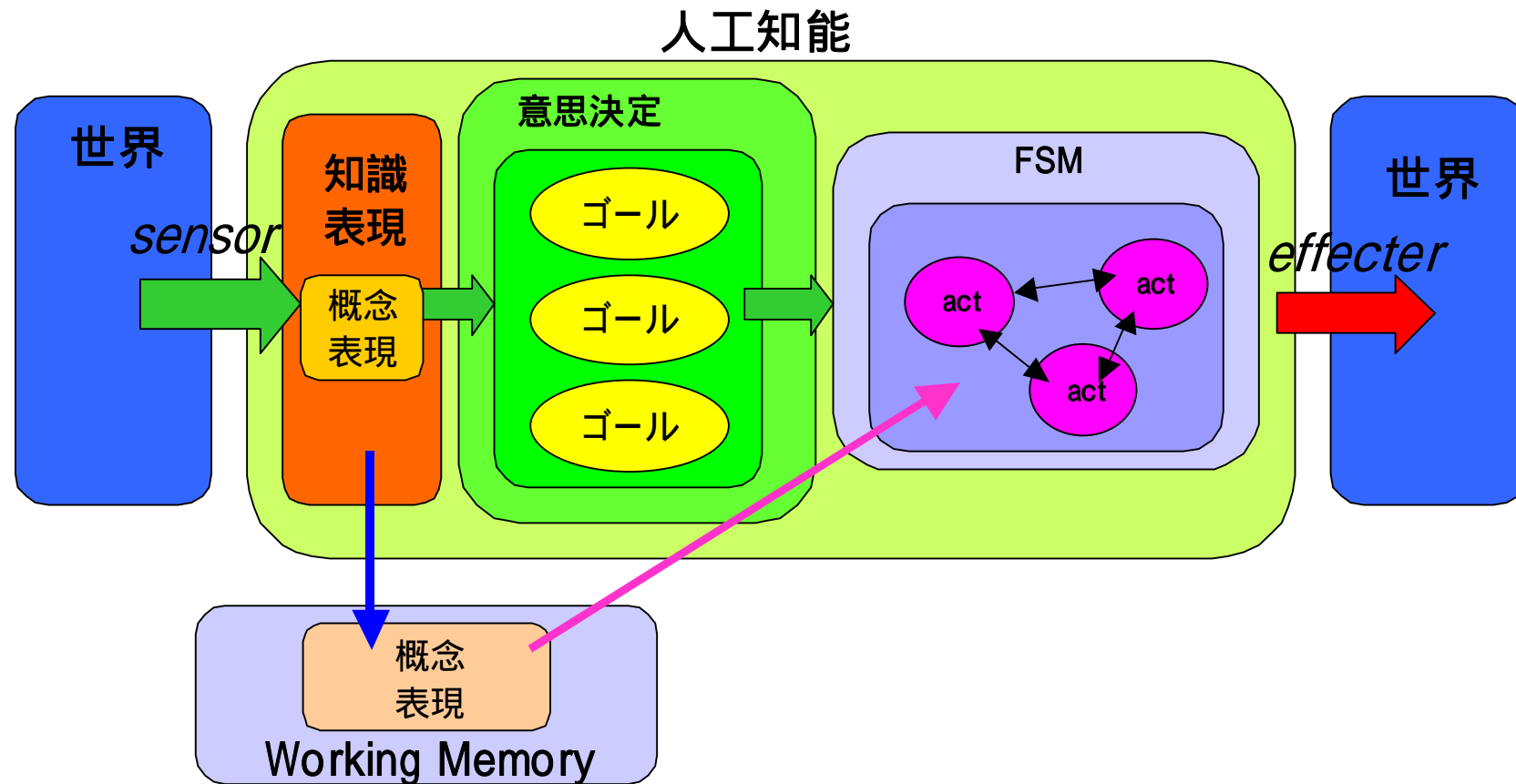


図 2-6 NOLF2 のエージェント構造 ([16])

第2節 F.E.A.R.におけるゴール指向プランニング

前節では NOLF2 の COM 全体の基本構成を説明しました。精緻な知識表現の作り込みの上に、ゴール指向と F S M を組み合わせたシステムであり、実現した機能としては期待通りであったと思われますが、工程が非常に込み入ったものとなり、見通しが悪くなってしまいました。ここで「プランニング」が F S M に代わるシステムとして現れます ([19])。プランニングによって、開発者が F S M で作り込んでいた「行動の順番のシステム」をプラナーに一任できるようになりました。それに伴い、新しい行動の追加が容易になり、また、キャラクターによる個性も、プラナーの行動に組み方にある傾向を持たせることで可能になりました。ここからは前節と同じ項目について、F.E.A.R. でどのような変更が行われたかを見ていきましょう。

第1項 ゴール指向型エージェント

F.E.A.R. においても、ゴール指向型エージェントという枠組みは変わりません。しかし、その枠組みの中に幾つかの要素が追加され、またもとなった幾つかの要素は新しく改善されました。NOLF2 と F.E.A.R. のエージェント構造図を比べると、そこには明確な違いが幾つかあります。ここでは、センサーから行動までのデータの流れを説明することで、F.E.A.R. のエージェントがどのように駆動していかを理解してみましょう。A I システムという水車が、時、或いは情報という水を受けて、くるくると駆動している感覚をつかんで頂ければと思います。

ゲーム世界から、**感覚(sensor)**によってから情報を集める。

この情報は、**統一知識表現**として**記憶領域(Working Memory)**に保存されます。

同時に**シンボル**と呼ばれる抽象的な情報が抽出され、**記憶領域**に格納されます。

記憶から**ゴール**を決定します。

ゴールに対して**プラナー**が**行動プラン**を連鎖します。

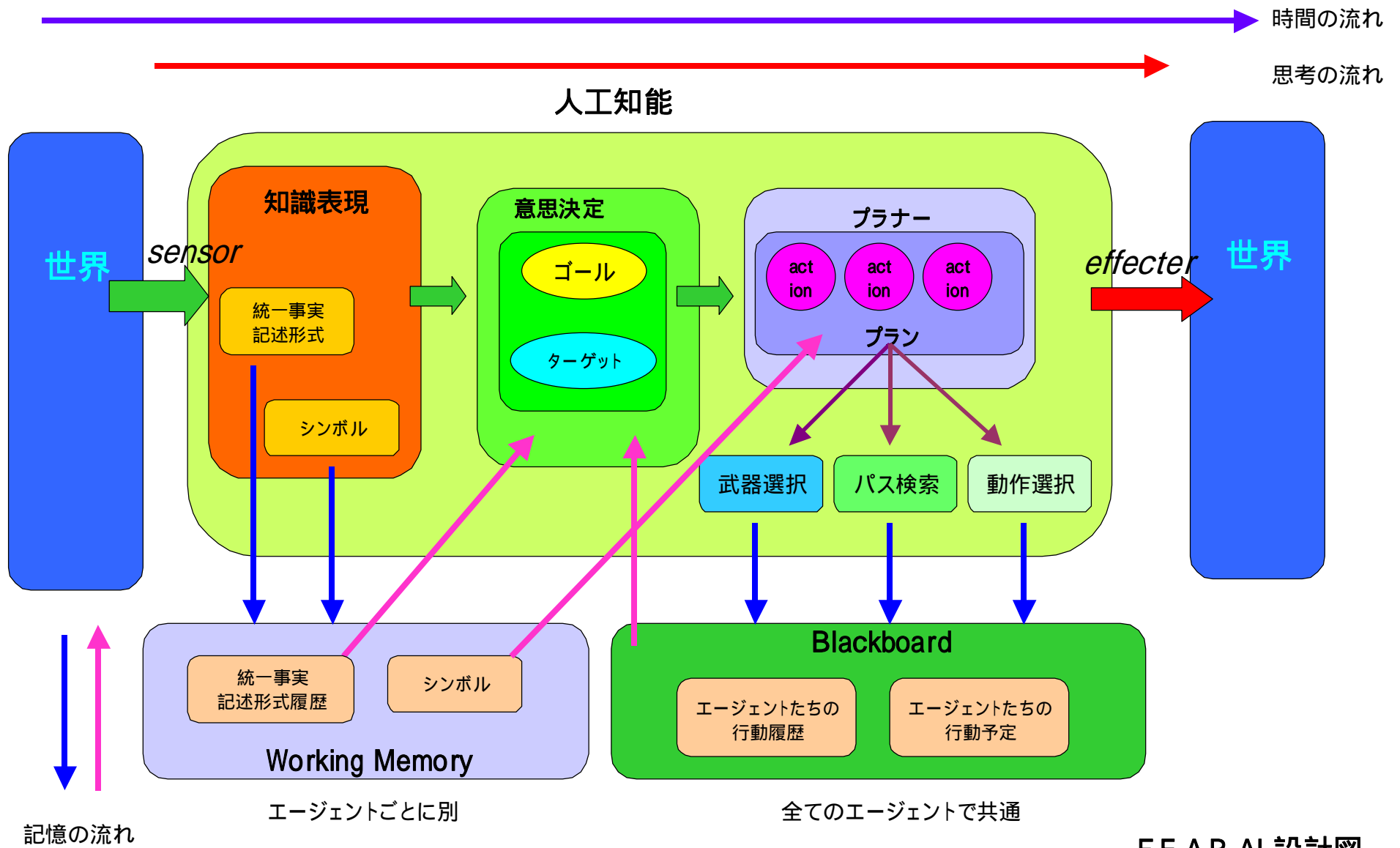
行動プランの実行に必要な**武器選択**、**パス検索**を行います。

各行動に対して必要な動作**アニメーション**を決定します。

行動プラン、選択した武器、経路は「**黒板**」(**Blackboard**)と呼ばれる共有メモリに書き込まれます。

黒板は、他のエージェントからも読むことが可能であり、黒板を通して、各エージェントは他のエージェントが**協調**します。

「世界から得た情報と記憶から、思考によって行動を作り、世界に影響を及ぼす」というサイクルを理解して頂けたでしょうか？ ただ、水車は水の流れによって回りますが、知能は、世界から情報を自分の中に通し、記憶を形成して行きます。そして、この記憶が現在の知覚と合わさって判断の材料となります。つまり、記憶は過去の情報を一つの形式で保ち、A I が過去から未来へ行動を生成する役目を持っています。A I には、記憶なしの A I と、記憶のある A I があります。後者は前者よりずっと複雑で、記憶は複雑な知能構造を可能にします。記憶にしる、プランニングにしる、知能というものが、時間とうまく付き合っていることがわかんと思います。そして、前者は過去を、後者は未来と深く関係します。記憶とプランニングは、人工知能が一瞬一瞬の現実ではなく、連続した時間の上に根を張ることを可能にします。



F.E.A.R AI 設計図

図 2-7 F.E.A.R におけるエージェント構造図 ([14])。Working Memory は、エージェントが感覚によって得た情報を蓄積する場所です。人間で言う記憶領野です。また、エージェントが行った行動は、履歴として「Blackboard」に書き込まれます。「Blackboard」とは、各エージェントが共通に書き込んで行く黒板のことです。ここに書き込んだ情報は、全てのエージェントが参照することが出来ます。イメージとしては「たくさんの COM が、自分のする行動をこの黒板に書いて行く」イメージです。そして、その情報から「あいつはこうする予定か、ならば、僕は違ったことをしようか」とか「協調しようか」などと思考します。プログラミングから見ると共有メモリーと殆ど同じですが、黒板モデルでは、そこに書き込んだ情報から不完全な情報が補完される機能がつくことがあります。たとえば、A、B、C の 3 つの部屋がある。エージェント 1 は「プレイヤーが A にいない」と書き込む。エージェント 2 は「プレイヤーは C にいない」と書き込む。すると黒板を管理するプログラムは、「プレイヤーは B にいる」という情報をアップデートする、と言ったようになります ([16,20])。F.E.A.R がそうであるかはわかりません。

さて、NOLF2 と F.E.A.R の A I の最も大きな違いは、NOLF2 が「何をすべきかを判断できるエージェント」であるのに対して、F.E.A.R は「何をすべきかを判断し、どのように達成するべきかを設計できるエージェントで」であるという点です。これは、技術的には F S M とプランニングの違いです。NOLF2 では、ゴールが一度決まると、その後は用意された FSM の遷移の中で制御されていましたが、F.E.A.R では「ゴールに対して、現在の状態（キャラクターの状態、ゲーム状況、プレイヤーの状況）から、一連の行動をデザイン」します。

表 2-4 FSM とプランニングの比較。

		FSM	プランニング
1	行動とゴールの分離	行動とゴールが分離されない。	行動とゴールが分離されている。
2	行動決定	あらかじめ、FSM によって決まれている。	プランナーがゴールとゲーム状態に応じて、つないで行ってくれる。
3	手間	FSM をゴールごとに設計。	行動を「前提条件、振る舞い、効果」という形式で書いて蓄積。
4	キャラクターカスタイズ	F S M の中でキャラクターごとに条件分岐。	キャラクターごとに新しい行動を用意。
5	アクシデントに対する適応	あらかじめ、その状態を FSM に組み込んでおく必要がある。	再プランニング (re-planning) によって、その場で新しくプランニングを行う。
6	長所	行動の遷移を事前に完全に規定できる。	準備された行動の要素から、プランナーが行動を状況に応じて自動生成。
7	短所	行動を結び合わせることで、固定された複雑なシステムになる。	プランナーがどのような行動を作成するため、完全に行動の作成を追いきれない。

NOLF 2 の AI の解説で、ゴール指向型 FSM は複雑な状況进行处理するために拡張を重ねて行くと、だんだんが見通しが悪くなる、メンテナンスがしにくくなる、という欠点挙げました。この経験を踏まえて、Jeff Orkin はその中心的問題が「行動とゴールの分離」と「行動の結合」にあると述べています。

NOLF2 では、ゴールと行動が FSM を通して完全に関係を持っています。また行動は互いに FSM の遷移条件によって結合されています。そこでは、開発者が、どんな条件からもゴールにたどりつけるように、巧みに FSM を作っておく必要がありました。

一方、F.E.A.R のプランニングでは、ゴールと行動は全く別のものとして扱われます。あらかじめ設定された行動とゴールの関係もなく、また行動同士にもまた関係はありません。行動は、第 1 章で述べたような「前提条件、振る舞い、効果」という形式で記述され、プラナーがこれを自動的に結合して一連の行動を生成するのです。

両者において、新しい行動を一つ加えること考えてみます。FSM では、どこにその行動を置いたらよいか、それぞれの FSM で検討する必要があります。一方、プランニングでは、それを一定に形式で用意しておくだけです。つまり、連鎖によるプランニングの長所は、要素となる行動の追加のしやすさにあります。

逆に欠点は、「全て行動の生成を追ってデバッグができない」という点と、「カスタマイズがしにくくなる」という点です。FSM では行動の設計図が目に見えてメンテナンスできますが、連鎖によるプランニングの場合には、プラナーを介して行動が組み上げられるため、直接カスタマイズすることは出来ません。

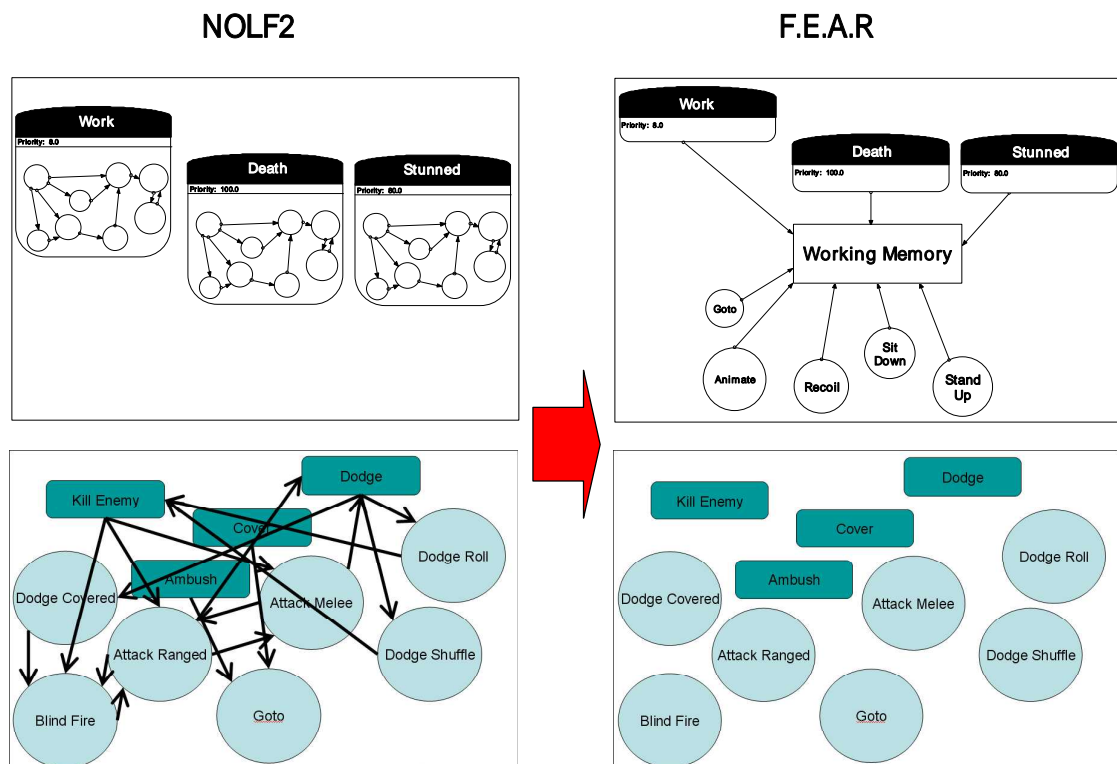


図 2-8 左が NOLF2 における COM の設計図 ([12])。それぞれのゴールの下に FSM が設定されています。右が、F.E.A.R における COM の設計図。ゴールに対して、取り得る行動(action)がプランニング(planning)されます。ここで、行動は、第 1 章で説明したように、「前提条件、振る舞い、効果」という形

式で定義されている。F.E.A.R の開発者は、この設計を、Goal-Oriented Action Planning (ゴール指向プランニング、GOAP)と命名した。この名称は、決して一般的なものではありません。ただ、おそらく、F.E.A.R の開発者は、広大なアカデミックな研究領域であるプランニングの領域の中で、自分たちの実装モデルを区別して名前を付けておきたかったのだろう。この名称は、IGDA AIISC (IGDA 人工知能インターフェース標準化委員会)においても、一つの分野の名称として採用されている ([21])。

第2項 知識表現、世界表現

エージェントは「感覚（センサー）」から得た情報から知識表現による記憶を蓄積します。さて、まず「感覚（センサー）」について説明し、そこで取得した情報をどのような形で蓄積するかを次に説明します。NOLF2では、やや力を入れすぎた感のある知識表現ですが、F.E.A.R では、非常にエレガントなシステムになっています。私の知っている中でも、これほど洗練された知識表現を構築したゲームは他にありません。

■ F.E.A.R における感覚（センサー）

生物であれば、その世界で、自然と五官を使って情報を得ます。ゲームの世界では、そういった五官もまた、計算によって代行しなければなりません。ここでは、エージェントが環境から情報を取得するために、具体的にどのような計算が為されているかを説明します（[14]）。センサーでは、

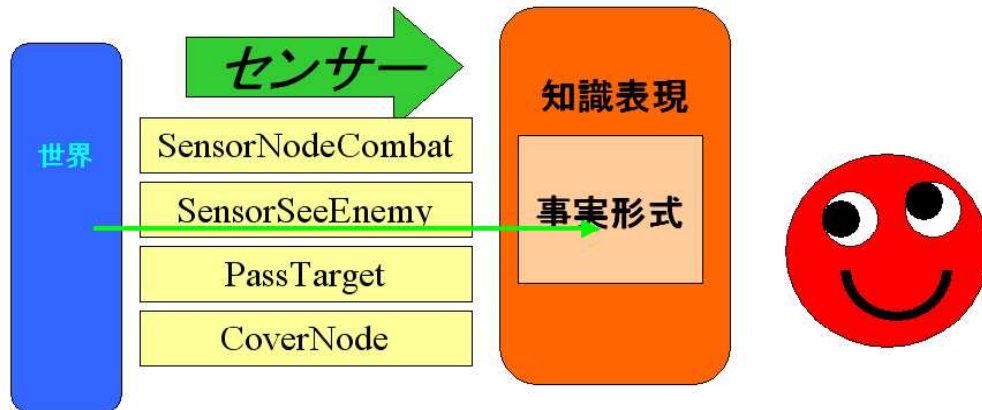
- 敵が見えているか (SensorSeeEnemy)
- 戦術ポイントまでのパスを見つけ、かつそのパスが敵から安全であるかをチェック (PassTarget)
- そのノードは隠れることができるか (CoverNode)
- 隠れるか、或いは、攻撃できるノードを探す (SensorNodeCombat)

などの情報を、光線計算（レイキャスト計算）やパス検索を使って取得します。例えば「プレイヤーを目撃した」を NPC とプレイヤーとの間の視線チェック（視線上に障害物があるかどうか）によって求めます。また、隠れる場所があるかどうかは、パス検索を使って判定します。このように、ゲームにおけるエージェントは、パス検索や光線計算といった負荷の高い計算によって視覚、聴覚など五官の代わりをさせます。

また、この情報取得の頻度が高ければ周囲の状況に敏感な NPC となりますが、そうすることで計算の負荷が高くなってしまいますので、取得すべき情報の種類ごとに、アップデートの頻度が、最大毎フレームから1秒に数回程度まで決められています。例えば、CoverNode は、一秒に3回しかアップデートされません。

少し脇道にそれますが、実際のロボット開発では、こういった情報はカメラや温度センサーなどといった得た情報を解析しますので、ゲーム AI と実際のロボットではセンシングという部分において大きな違いがあります。一旦、メモリに解析された情報が載ってしまえば、電子頭脳の中で、類似点のある AI が走ることとなりますが、ロボットの研究においては、この情報取得、さらに解析という部分に多くの研究課題を抱えています。ゲームにおいて如何に負荷の高い部分とは言っても、仮想空間と現実空間では大きな隔たりがあります。逆に、それゆえに、ゲーム AI は、その思考部分を十分に研究できる場として、大きな価値を持っています。

センサー = 五感からの情報、及び、五感から得られるはずの情報を模擬する機能



レイキャスト(視線チェック)、パス検索など重たい処理からなる関数

SensorNodeCombat 隠れる、或いは、隠れながら攻撃できる場所を探す。
SensorSeeEnemy 敵が見えるかチェック
PassTarget 戦術ポイントまでのパスを見つけ、かつそのパスが敵から安全であることをチェックする。
CoverNode 隠れることができるノード

図 2-9 F.E.A.R. における感覚(センサー)から知識表現までの流れ。ゲーム世界から、幾つかのセンサーによって情報を得、記憶として蓄積する。

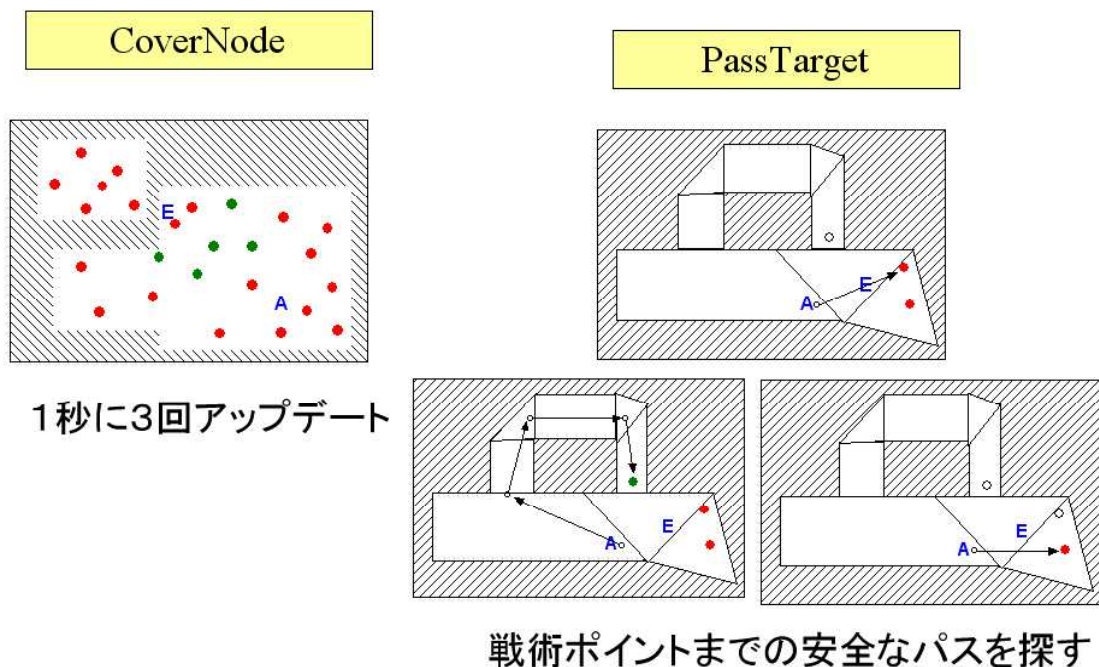


図 2-10 F.E.A.R. のCOMのセンサー ([14])。「CoverNode」では、プレイヤーから隠れることができる

ノードをアップデートするセンサー。「PassTarget」は、プレイヤーからは隠れて移動できる戦術ポイントまでのパスを発見するセンサー。ゲーム AI では、人間なら視覚で一瞬で処理するこのようなことに、最も大きな負荷を取られてしまう。将来は、各ゲーム開発会社が、このセンサー部分においてどれほど優れたアルゴリズムを組むかという競争になるだろう。既に第 1 回のセミナーで述べたような Killzone のセンサーの仕組みや、AI Game Programming Wisdom , Game Programming gens などでも多くの記事が用意されている。ミドルウェア会社も、この部分でどれほど高速で優れたセンサーを実現できる土台を提供できるかで、その出来を判定されることになるだろうと予想される。

■ F.E.A.R. における記憶

人工知能を規定するものは、一つは思考、そして、もう一つは記憶の形、知識表現です。感覚によって得た情報は、ゲームに適した形で記憶される必要があります。F.E.A.R. では 2 種類の記憶の形式

- (1) 統一事実記述形式 ゲーム世界で起こった事実についての記憶
- (2) シンボル ([22]) ゲーム世界の状況を 20 個に集約した情報

が準備されており、この二つが記憶領域(Working Memory)の中に格納されます。(1) は、ゴール、ターゲット、パスなどを決定する場合に用いられ、(2) はプランニングの前提条件、効果のために用いられます。この二つの詳細について説明します。

(1) 統一事実記述形式

統一事実記述形式とは、

- ◆ キャラクター
- ◆ オブジェクト
- ◆ ノード (ウエイポイント、ナビゲーションメッシュのノード)
- ◆ 妨害
- ◆ タスク
- ◆ パスに関する情報
- ◆ 欲求
- ... (全部で 10 個)

など、ゲームフィールドに存在するオブジェクト、ターゲット、そして、概念についての記憶です。F.E.A.R. では、これらを全て同じ形式「場所」「方向」「感覚のレベル」「オブジェクト」「情報取得時刻」など、16 個に及ぶ項目によって記述します。ある場合には、使わない項目もあるでしょうが、同じデータ構造を持つと処理のしやすさというメリットがあります。

例えば「プレイヤーがはしごを登っている」をエージェントが認識したとします。すると、それは、プ

レイヤー に関する事実として、場所(x20, y30, z 0)において、方向(0,0,1)に、オブジェクト(はしご)を、時刻(1分38秒)に登っていたのを感じ(視覚)によって見た、という形式にされ記憶されます。

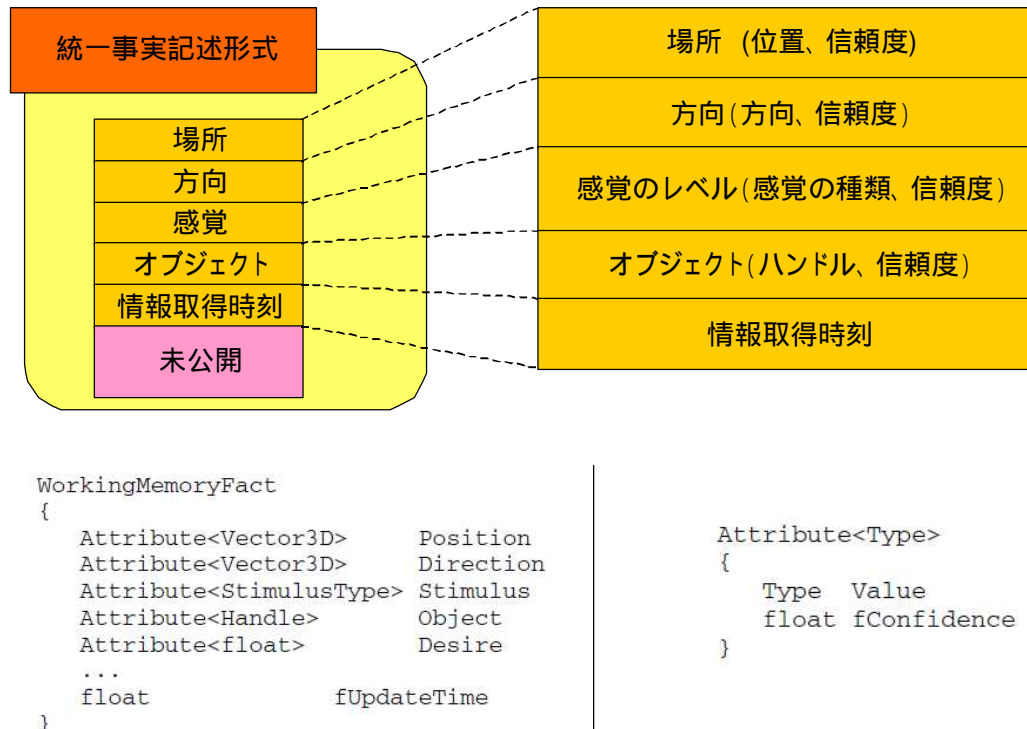


図 2-11 統一事実記述方式 ([23])。本当は16個あるが公開されているのは上記のものだけだ。

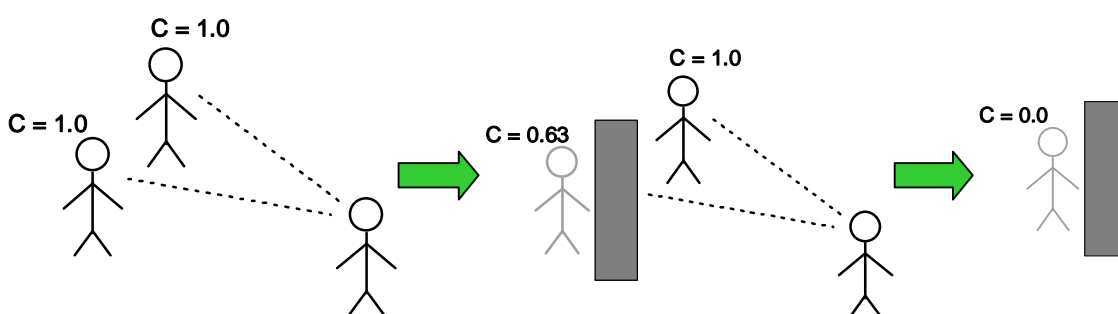


図 2-12 位置(position)に対する信頼度(confidence) ([14])。COM が視覚でキャラクターを認識しているなら、その位置は1.0(100%)の信頼度を持つが、ある時刻に隠れたキャラクターに対しては、時間と共に、信頼度が下がり、ある時点からは、0になる。

各情報に付随する信頼度 (confidence) が設定されており、たいていの場合は、その情報の確実性を現しますが、情報の種類によって異なる意味を持ちます。都合のよい強度パラメーターと言ったところでしょ

うか？例えば、キャラクターの事実 (Fact) における刺激 (stimulus) の信頼度とは、どれくらい、そのキャラクターを認識できているかの度合いを表します。位置 (position) における信頼度とは、行き先としてそれがどれだけふさわしいかの評価値です。また、欲求 (desire) における信頼度とは、そのキャラクターがどれだけ強く望んでいるかを表します。こういった知識の不正確性をパラメーターとして持つことで、AIにおける思考は、不確実性を汲んだ思考ができるようになります。たとえば、「敵がいる」という情報が複数あったときにも、それがどれだけの信頼度を持っているかで、その中からの確かな情報を選択することが出来ます。

統一記憶形式による記憶は共有メモリ領域 (Working Memory) に種類ごとにスタック (積み上げ) されます。全てが同じデータ構造なので、プログラマーから見た場合管理しやすいことは言うまでもありません。

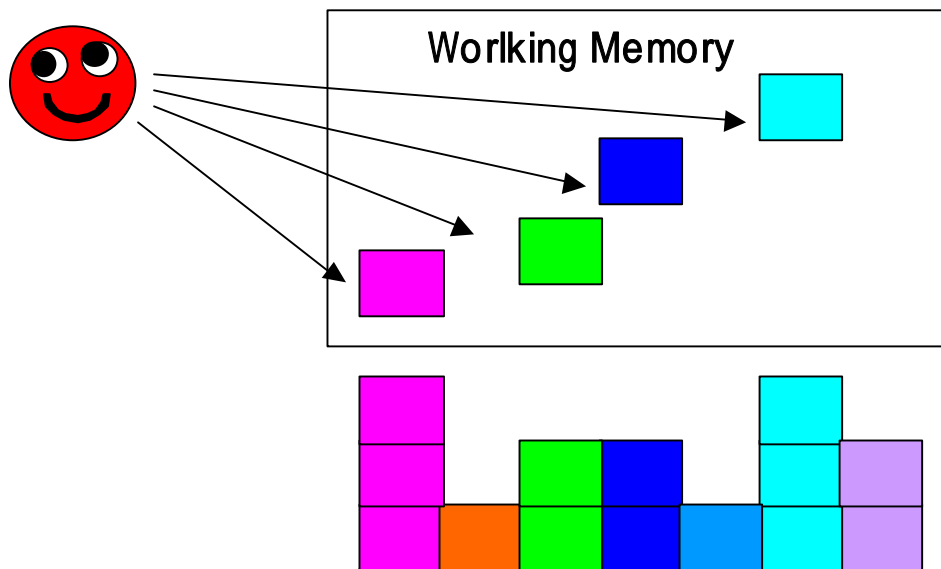


図 2-13 エージェントは共有メモリ領域にゲーム世界から得た事実情報を統一事実記述方式で蓄積して行く ([14])。情報は各エージェントを中心にした情報である。「自分からの距離」「自分が認識しているか」など。こういった知識の記述の仕方をエージェント中心 (agent-centric) という ([23])。

(注) AIに限らず、知性にとって物事をどのような形で世界を記憶しておくかということは、たいへん重要なことです。いろいろな知性の特性は、その記憶の形式で規定される、と言ってもよいぐらいです。同じ人間同士でも、記憶の仕方には個性があれ、その記憶の形の違いが、知性の個性となって現れることがあります。知識のひとまとまりのことをチャンクと言います。チャンクの形は人により様々です。

(2) シンボル

統一事実記述方式とは別に、プランニングで用いるための「シンボル」という情報があります ([13,22])。これは、複雑なゲーム世界を20個ほどの項目に集約した情報です。これは、プランニングの前提条件と効果を表現するために用いられます。連鎖によるプランニングでは、効果と同じ前提条件を検索したり、その逆の検索の処理が必要ですので、検索する領域は小さいほど高速になります。そこで、世界をある程度単純にモデル化して情報を簡略化します。シンボルは (プログラマ的に言うところの共用体であり) ある場合には数値

だったり、別のシンボルへの参照だったり、具体的なオブジェクトやウェイポイントのノードを指すものであったりします。

表 2-5 F.E.A.R で用いられるシンボル情報 ([12,13,14,22]から表を構成)。これらのシンボルは、プランニングの前提条件と効果として使用されます。シンボルのデータ型は共用体となっていて、それぞれのシンボルに応じたデータの型を取ることが出来ます。公開している情報からはここまでしか埋められません。

シンボル名	Type	説明
kSymbol_Animplayed		そのターゲットの動作状況
kSymbol_AtNode	Handle or variable*	現在のノード
kSymbol_AtNodeType		現在のノードタイプ
kSymbol_AtTargetPos		ターゲット位置
kSymbol_DisturbanceExists		
kSymbol_Idling		
kSymbol_PositionIsValid		
kSymbol_RidingVehicle		乗っている乗り物
kSymbol_ReactedToWorldStateEvent		
kSymbol_TargetIsAimingAtMe		自分を狙っているか
kSymbol_TargetIsDead	bool	ターゲットが生存しているかどうか
kSymbol_TargetIsFlushedOut		ターゲットが見えているかどうか
kSymbol_TargetIsSuppressed		ターゲットが威嚇射撃されているかどうか
kSymbol_TraversedLink		
kSymbol_UsingObject	handle	使えるオブジェクト
kSymbol_WeaponArmed	bool	武装しているか
kSymbol_WeaponLoaded	bool	武器は装填済みか

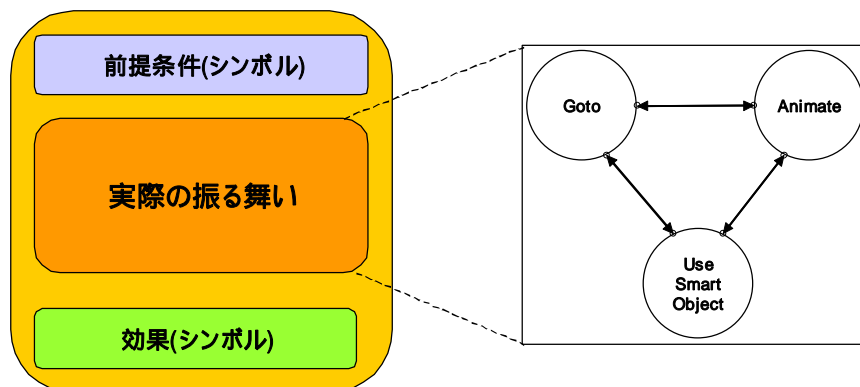
第3項 行動の決定

この節では、F.E.A.Rにおけるプランニングの実装を説明します。

第1章で解説したように「連鎖によるプランニング」における行動は、前提条件、効果、行動から記述されます。F.E.A.Rにおいては、この前提条件と効果を前節で説明したシンボルによって記述し、プラナーはその条件をたどってプランを組みます。

F.E.A.Rでは、この「振る舞い」の部分は、「動作を演じる」(Animate)「何処かへ行く」(Goto)「物を使う」(UseSmartObject)の3つの何れかの情報です。「物を使う」というのは、例えば武器を拾ったり、装填したりする動作のことです。連鎖によるプランニングは一連の行動を生み出しますが、それは、「この3つの状態がある順序で組みあわさったもの」になっています。例えば「ある場所へ行って」「武器を拾って」「銃を撃つアクションをして」「あるポイントへ移動する(隠れる)」などです。

別の見方をすれば、最終的にCOMの制御部分に渡されるのは、この3種類の状態の何れかです。COMの体の制御部分とAI(思考)部分を、独立に考えてみます。思考部分は、その行動を指定するコマンドの列を作り出し、制御部分は、思考部分からは、順次、行動を指定するコマンドを受け取ります。AIと制御部分を独立して考えると、プランニングの構造がよく見えて来ます。これは次章のクロムハウন্ズのCOM制御でより明確に理解できます。クロムハウন্ズでは、最終的にCOMの制御部分が受け取るコマンドは3種類しかありません。



```
class Action
{
    // Symbolic preconditions and effects,
    // represented as arrays of variables.
    WORLD_STATE m_Preconditions;
    WORLD_STATE m_Effects;

    // Procedural preconditions and effects.
    bool CheckProceduralPreconditions();
    void ActivateAction();
};
```

図 2-14 F.E.A.Rにおける行動の記述形式 ([12,23])。前提条件と効果はシンボルとして記述され、振る舞いは、右図のように、「ある場所へ移動」(goto)、「アニメーション再生」(Animate)、「オブジェクトを利用」の3つから構成されます。例えば、「物陰に隠れる」(cover)は、「あるノードへ移動」「かべにもたれる」

アニメーションを再生からなります ([12])

これまでの準備の上に具体的な例を使って、F.E.A.Rにおけるプラナーの動作を順を追って説明しましょう。第1章の「のどの渇きをいやす」プランニングと全く同じです。まず、前提として、F.E.A.Rのプラナーはゴールから出発して、連鎖を組んで、現在の状態まで動作をリンクして行きます。ゴールから逆に初期条件へ向かってプランニングする方法を後向き(regression)プランニングと言います。

今、ゴールが、シンボル「`kTargetIsDead==true`」(敵が死んでいるか、が `yes`)、自分の状態「`kWeaponArmed == none`」(武器を持っているか、`no`) であるとします。「`kTargetIsDead = true`」から出発して、プラナーがどういふことをするかを、逐次的に追ってみましょう ([13])。意味的に言うと「ターゲットを倒す」というゴールのためのプランニングです。

- (1) ゴールを選択します。ここでは「`kTargetIsDead==true`」とする。
- (2) ゴールは「`kTargetIsDead = true`」であるが、現在の値は、まだ実現していないので、*false*。
- (3) 「`kTargetIsDead = true`」を効果として持つ行動を探す。「`Attack`」が見つかる。前提条件は、「`kWeaponLoaded = true`」。しかし、現在の値は *false*。
- (4) 「`kWeaponLoaded = true`」を効果として持つ行動を探す。「`LoadWeapon`」が見つかる。前提条件は、「`kWeaponArmed = true`」。しかし、現在の値は *false*。
- (5) 「`kWeaponArmed = true`」を効果として持つ行動を探す。「`DrawWeapon`」が見つかる。前提条件は、「`kWeaponArmed = none`」。つまり、特になし。つまり、現在の自分の状態までたどって来ることが出来たので、プランニング終了である。

これを逆から読むと、「`DrawWeapon`」「`LoadWeapon`」「`Attack`」(武器を取って、装填して、攻撃) となります。これで、このNPCが取るプランが作成されました。

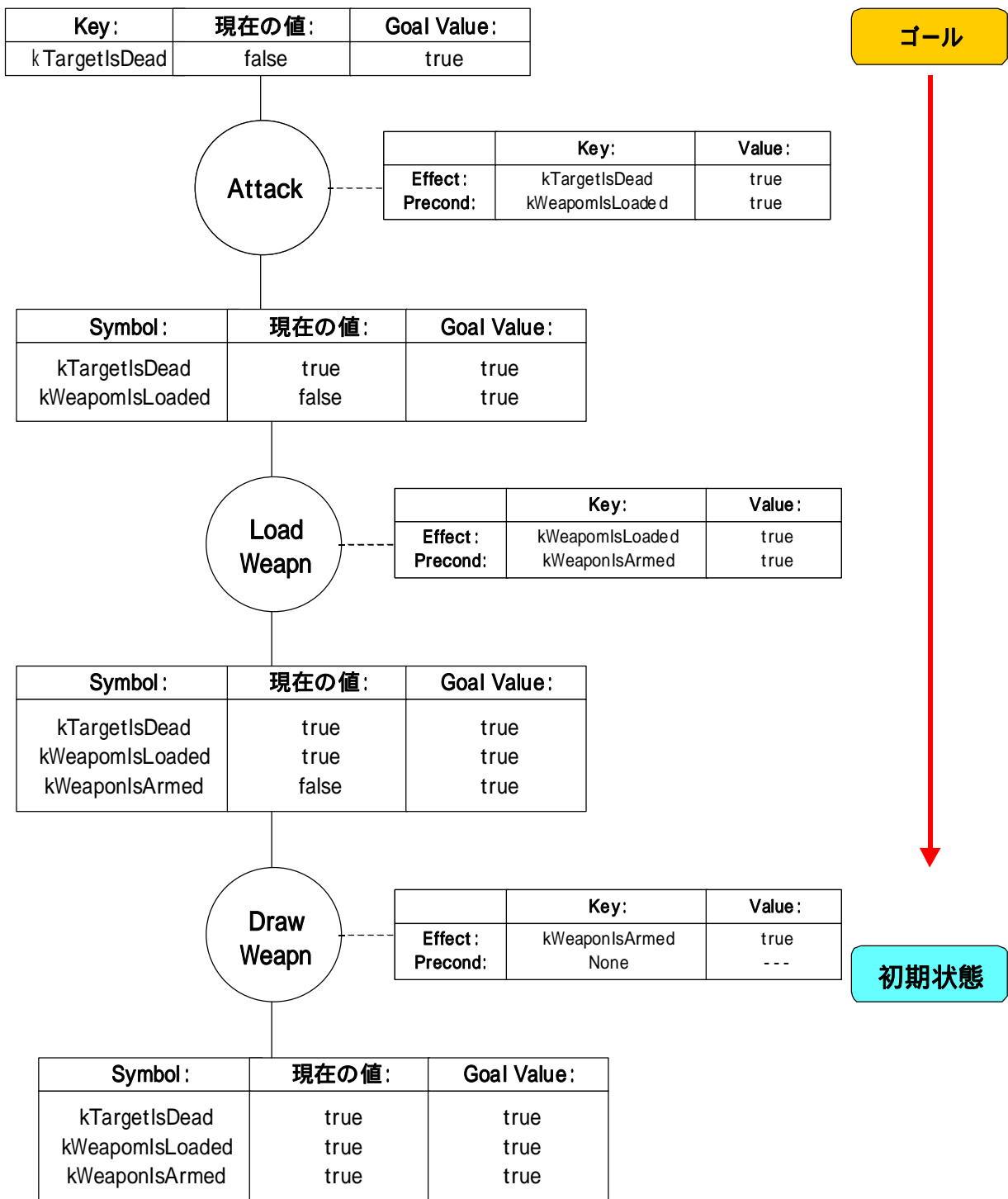


図 2-15 プラナーによる行動プラン作成過程 ([13])。これが、F.E.A.R のプランニングの基本。

第3節 より詳しいプランニングの解説

第2節を通じて、F.E.A.R _AI 全体の中の連鎖によるプランニングを説明しました。プランニングのアルゴリズム自体はシンプルですが、そのために様々な準備が必要であることを見ました。ここからは、F.E.A.R のプランニング自体のより詳細な構造を説明します。

第1項 複数のプランから一つを選ぶ

第2節では直線的にプランを構築しましたが、実際は、同じ効果を持つ行動が複数ありますので複数のプランが候補として作成されます。例えば、「kTargetIsDead = true」を満たす行動は「撃つ」「殴る」など複数あり、また「武器を取る」という行動に対しても複数の武器が選択することが可能です。つまり、通常プランニングは連鎖部分において複数に分岐し、結果として複数のプランを生成します。このプランの分岐こそが、第1章で説明しように AI の行動の多様性を生み出す仕組みです。AI は生成された可能な行動のうちから、一つ、取るべき行動を選択すればよいわけです。選択の仕方には様々な方法があります。ランダムに選んでもいいし、また、ゲーム状況に適するものを選ぶロジックを組んでもよい。この選択方法はゲーム毎に工夫するポイントです。

F.E.A.R では、この部分を「コスト」の考え方によって決定します。この方法では、各行動にどれくらい、手間がかかるか、という点数をつけます。F.E.A.R では開発者が適当に決めて行きます。そして、プランニングによって生成した複数のプランのうち、最も、総合コストの低い行動を選択するように設計されています。これは技術的には「A*によるグラフ検索のアルゴリズム」によって可能です。プランニングは行動を要素とするグラフ構造となりますが、最小コストのプランを自動的に発見する方法によって、自動的に最小コストのプランが複数のプランの中から探し出されます。A*はもともと、パス検索で多用されるアルゴリズムであり、こういった応用方法はたいへんユニークです。

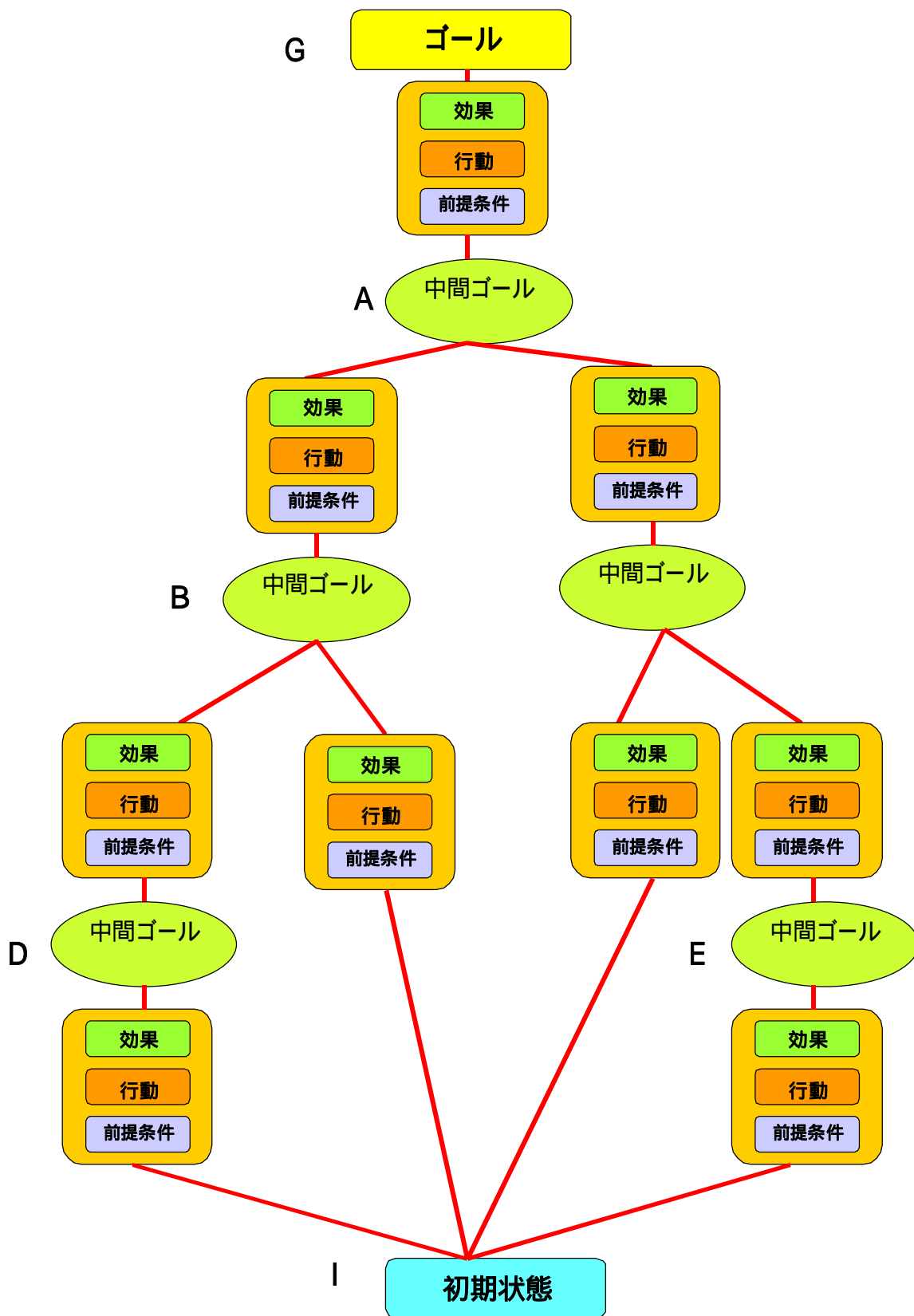


図 2-16 プランの分岐。複数のプランが作られる。

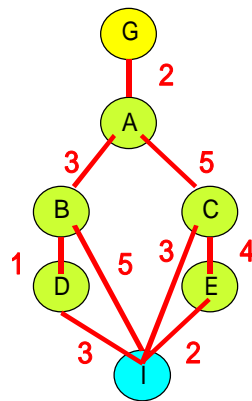


図 2-17 上の分岐頭をグラフ構造に書き直したもの。G はゴール状態。I は初期状態。緑色のノードは、中間状態を表す。赤いラインの横の数字は、その間にある行動のコストを表す。コストは、F.E.A.R では、開発者が各行動について、あらかじめ決定している。一般に、こういったネットワークグラフにおいて、ノード間の最小コストパスを検索するアルゴリズムは、ダイクストラのアルゴリズム、或いは、その発展形である A* アルゴリズムがあり、F.E.A.R では後者が採用されている。上のグラフでは (G,A,B,D,I) がコスト 9 で最小コストのパスとなるので、このパスに沿ったプランが採用される。技術的な詳細になるが、A* が用いる heuristic なコストは「条件の個数の差」を用いる ([12])。

第 2 項 キャラクターの違いによるプランニング

NOLF 2 では、AI に個性を出すためには、FSM をカスタマイズする必要がありました。プランニングでは、キャラクターごとに要素となる行動を用意しておくだけで個性化を実現できます。各プラナーに、各キャラクター固有の行動要素からなるプランを生成させるからです。同じゴールに対しても、キャラクターの特性ごとに行動が組み立てられます。

例えば、「敵を攻撃する」に対して、兵士なら「撃つ」、暗殺者なら「刺す」、ねずみなら何もできません。「相手に近付く」という効果を得るために、兵士なら「前進する」、暗殺者なら「天井をばう」、ねずみなら「床を歩く」という行動を選択します。プラナーから見れば、前提条件と効果をつなげて行くだけなので、その行動が実際どのようなものであれ関係なく、そのキャラクターが取ることが出来る行動の要素を選んで連鎖して行ってくれます。また導入した行動には前項で述べたようにコストをつけますが、ここで、このコスト付けによって、さらに AI を特徴付けることをします。例えば、F.E.A.R ではわざわざ「単に撃つ」より「隠れて撃つ」という行動のコストを小さくしています。もちろん「隠れて撃つ」方が面倒な行動なので手間がかかるはずですが、開発者は NPC が隠れてプレイヤーを攻撃するという行動をプレイヤーに見せたいので、敢えて取らせたい行動のこのコストを十分に小さくしておく、ということを行います。結果として、F.E.A.R をプレイした方はすぐに気がつくと思われますが、NPC は、物の陰から攻撃するという行動を頻繁に選択します。

ただ、FSM に比べると、プラナーというシステムを挟んだ個性づけなので、直接的でない感覚があるかと思います。人工知能技術を使うというのは、多くの場合、自分が直接手を下していたものを自律的なシステムに任せて「手を放す」ことでもあり、それが時に不安を呼び、導入への障壁になる場合があります。しかし、

「手を放し」た後にこそ、そのシステムをうまく動作させるための多くの作業が残っています。連鎖によるプランニングでは適切な行動を用意すること、適切な連鎖条件を設計すること、プラナーの挙動を確認することなどです。

Soldier	Assassin	Rat
<div> <div>Action</div> <div> <div>1 AI/Actions/Attack</div> <div>2 AI/Actions/AttackCrouch</div> <div>3 AI/Actions/SuppressionFire</div> <div>4 AI/Actions/SuppressionFireFromCover</div> <div>5 AI/Actions/FlushOutWithGrenade</div> <div>6 AI/Actions/AttackFromCover</div> <div>7 AI/Actions/BlindFireFromCover</div> <div>8 AI/Actions/AttackGrenadeFromCover</div> <div>9 AI/Actions/AttackFromView</div> <div>10 AI/Actions/DrawWeapon</div> <div>11 AI/Actions/HolsterWeapon</div> <div>12 AI/Actions/ReloadCrouch</div> <div>13 AI/Actions/ReloadCovered</div> <div>14 AI/Actions/InspectDisturbance</div> <div>15 AI/Actions/LookAtDisturbance</div> <div>16 AI/Actions/SurveyArea</div> <div>17 AI/Actions/DodgeRoll</div> <div>18 AI/Actions/DodgeShuffle</div> <div>19 AI/Actions/DodgeCovered</div> <div>20 AI/Actions/Uncover</div> <div>21 AI/Actions/AttackMelee</div> </div> </div>	<div> <div>Action</div> <div> <div>1 AI/Actions/Attack</div> <div>2 AI/Actions/InspectDisturbance</div> <div>3 AI/Actions/LookAtDisturbance</div> <div>4 AI/Actions/SurveyArea</div> <div>5 AI/Actions/AttackMeleeUncloaked</div> <div>6 AI/Actions/TraverseBlockedDoor</div> <div>7 AI/Actions/UseSmartObjectNodeMounted</div> <div>8 AI/Actions/MountNodeUncloaked</div> <div>9 AI/Actions/DismountNodeUncloaked</div> <div>10 AI/Actions/TraverseLinkUncloaked</div> <div>11 AI/Actions/AttackFromAmbush</div> <div>12 AI/Actions/DodgeRollParanoid</div> <div>13 AI/Actions/AttackLungeUncloaked</div> <div>14 AI/Actions/LopeToTargetUncloaked</div> </div> </div>	<div> <div>Action</div> <div> <div>1 AI/Actions/Animate</div> <div>2 AI/Actions/Idle</div> <div>3 AI/Actions/GotoNode</div> <div>4 AI/Actions/UseSmartObjectNode</div> </div> </div>

図 2-18 キャラクターごとに用意された行動 ([24]PPT)

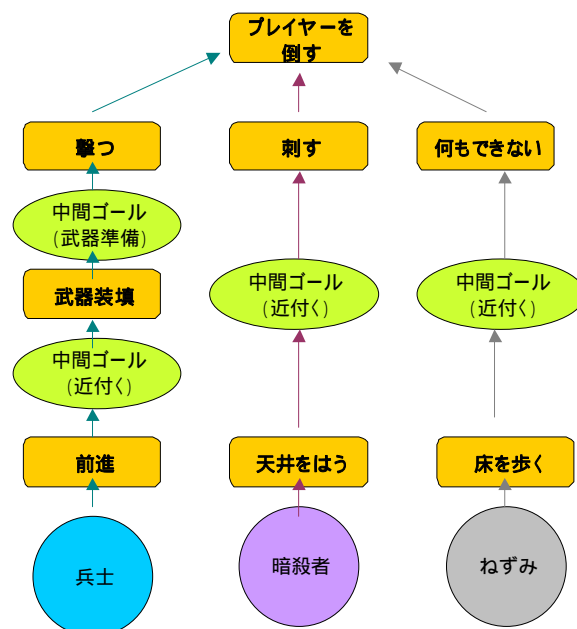


図 2-19 キャラクターの違いによって、一のゴールに向かって異なるプランが生成される様子。

第4節 リプランニング

ここまで、プランニングの長所を見て来ましたが、全ての方法がそうであるように、プランニングにも欠点があります。最大の長所は、最大の欠点であります。プランニングの欠点は、

AIは、自分が立てた行動に従わねばならない

という言葉で表現されます。プランニングはある時点から未来へ向かってプランを立てます。しかし世界の状態は、時々刻々と変わりますので。プランを実行している間に状況が変化したり、突発的な事象が起こってプランを生成した前提条件が崩れてしまうことがあります。そういった場合には、最初立てたプランを実行し続けることは、賢明でないどころか愚かしい行動とります。橋が落ちたのに橋を渡ろうとするようなものです。

例えば、レーザー砲台のある戦場で、あるCOMが戦局が不利になって来たので、「レーザー砲台を撃つ」というゴールを達成しようとしします。レーザー砲台の電源スイッチは、レーザー砲台から離れたA地点にあるので「A地点へ行く」「電源を入れる」「レーザー砲へ行く」「撃つ」というプランニングをします。さて「レーザー砲台へ行く」までは計画通り実行できていましたが、「レーザー砲」は到着する寸前に敵が占拠していて、こちらを攻撃して来ました。計画通りに実行すると、レーザー砲の台座まで進むことになりませんが、これは知的な行動とは言えません。

そこで、プランニングにおいては、プランの実行中でも、あるゲーム状態の変化を認識した場合には実行中のプランを中止しプランニングをやり直す「再プランニング」(re-planning)という方法があります。要するに、現在の条件でプランニングをし直すわけですが、ポイントはいつ、それをするべきか、ということです。つまり、AIは「プランの破れ」に敏感でなければなりません。クロムハウন্ズでは、この部分を、ゴールに寿命を設けることで対応しています。つまり、指定した時間がたっても、そのゴールが達成できないなら、それは失敗したと見直して、リプランニングに入ります。また、この方法は、パスプランニングにおいても利用できます。想定時間以内に、目的地にたどりつかないなら、どこかで行き止まっているとか、崖から落ちたかもしれませんので、時間が過ぎたところで失敗として、パス検索をやり直します。F.A.R.におけるリプランニングの詳細は説明されていませんが、一つの例が提示されているので、それを説明します。

今、AIは開いたドア越しにプレイヤーを目撃し襲いかかろうとします。プレイヤーはドアを締めることで、これに対抗します。すると、

1. COM(A)は、プレイヤー(P)が見えているので直進して攻撃しようとする。
2. しかし、Pへ向かう途中でPがドアを締めてしまう。
3. この時点で再プランニングによって、横の窓ガラスを割って侵入するというプランを立てる。

というリプランニングが実行されます。つまり、一つ目のゴール「近づく」が失敗した時点でリプランニングが開始されています。これについては、GDCで発表されたデモのムービーがありますので、それを見て頂ければ、「こちら(プレイヤー)の行動によってNPCが計画を変える」AIが、どれほど人間らしく見えるかを実感して頂ければと思います。

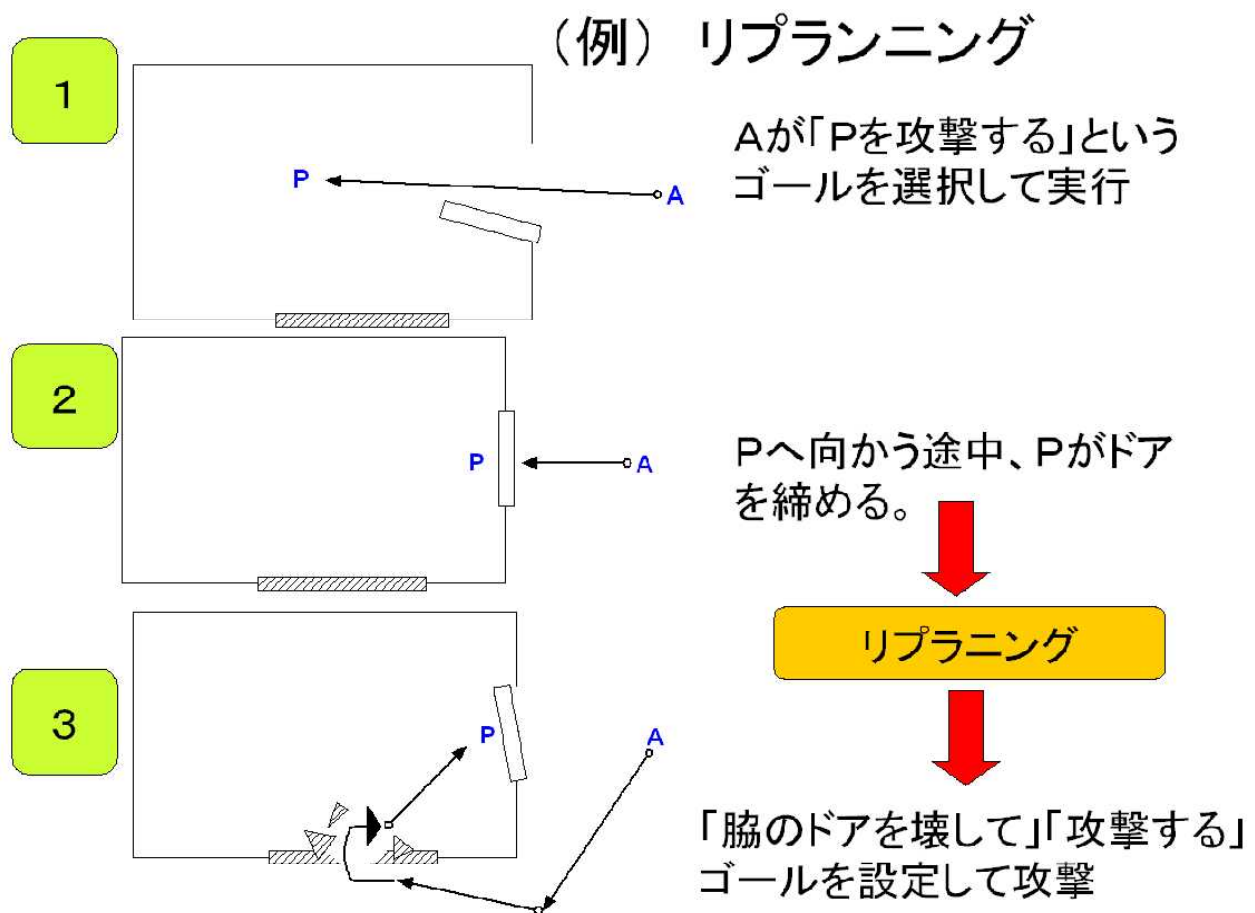


図 2-20 F.E.A.R. におけるリプランニング ([14] [12])

毎瞬間の状況に対する対応は、FSM や単純反射の得意とするところでしたが、プランニングはリプランニングによって状況の変化に対する脆弱性」という欠点を補っています。

第5節 まとめ

この章では、F.E.A.R. におけるゴール指向プランニングを説明しました。F.E.A.R. のA Iは、エージェント構造を持つA Iであり、その行動を決定する機能としてゴール指向プランニングが使われています。プランニングは連鎖による方法で、その前提条件と効果は、シンボルと呼ばれる20個のパラメーターになっていました。また、変化する状況に対応するため、プランニングは状況の変化に従って、更新されることを見ました。また、各行動にはコストが設定され、複数のプランから最小総合コストのプランを選択するシステムが組み込まれていることを解説しました。さらに、キャラクターの種類ごとの個性づけには、各種類ごとの行動を用意して、選択的にその行動からプランを組み合わせることで、個性付けされたプランの創造を可能に

しました。

以上は F.E.A.R、NOLF2 についての Jeff Orkin の論文、記事、或いは、タイトル名は書いていないが F.E.A.R 発表前に出版された記事などを総合して、書かれていないが予想できる部分は予想しながら全体を再構成しました。ただ、やはり幾つかの部分は、完全に解説が抜けているために埋めることが出来ないままになっています。例えば、意思決定の部分で、具体的にゴールをどのように選択しているのか、或いは、演出上、開発者が AI にゴールを固定して渡している場合があるのかなど、の点です。しかし、目的は F.E.A.R の全体像を全て理解することではなく、自身のゲームタイトルに役立つ要素を見つけて応用することですから、説明のない部分を自分自身で推測したり組み立ててみたりすることは有意義なことかと思います。

こうして解説してみると、応用的な部分が多く複雑に見えますが、第 1 章で見たようにプランニングの基本の原理はシンプルです。F.E.A.R の連鎖によるプランニングは、STRIPS ([1]) というプランニングのフォーマットを基本にしていますが、シンボル形式のデータ構造などはオリジナルのアイデアもあります。そういった部分的にオリジナルな技術を開発しながら人工知能の知識を応用する姿勢は非常に参考になるかと思います。

ゲームシステムは開発者から見ると、天上から俯瞰するように、単純なものに見えますが、実際のフィールドに立つプレイヤーや NPC には、混沌とした<状況>に直面させることになります。人工知能技術は、そういった混沌を、エレガントなシステムで効率よく処理する技術であり、特にプランニングはそのような状況から未来へ向かって知的な行動を創造する技術です。

NPC の立場に立って、NPC がどのような状況に直面して、どのような情報を処理しなければならないかを想像すると NPC に必要なことが見えて来ます。そして、その必要なことは、人工知能研究の歴史の中で開発されて来た某かの技術に対応するはずで、F.E.A.R の開発者たちは、その論文や記事から見る限り、そういった要求を敏感に捉える能力に長けていました。そして、必要な技術を自分たちのゲームに合わせて応用する姿勢にも富んでいました。なおいいことに、彼らはそのノウハウをこのように他の開発者のために公開してくれました。

また、プログラマーの立場から見ると、ゲーム AI の技術は、上記で説明した「NPC が直面する混沌を解消する人工知能の問題」を「通常のプログラミングの問題」まで還元する技術として捉えることが出来ます。例えば、プランニングでは、行動の決定の問題を検索の問題に還元しました。

一方、企画から見た場合、連鎖によるプランニングの技術は、FSM やルールベースなどの全てを決定できるシステムと違って、プランナーの外側から行動の要素を用意することがカスタマイズの方法となります。そこで、これまで違った意味での「作り込み」が要求されます。どのような行動を用意すべきか、前提条件と効果はどのようにすべきか、デバッグはどうやってすべきか、など、人工知能の技術の上に立った、新しい製作工程が要求されます。

幸運なことに、或いは、残念なことなのか、人工知能は必ずしも完全なる魔法の杖ではなく、ある場所で大きく作業を削減し、これまで以上の機能を実現するかわりに、それをうまく動作させるための作業と調整（デバッグ）を必要とします。そこには、人工知能技術の上に立った新しい製作工程のスタイルがあります。

[第2章おわり]

第3章 クロムハウズにおける階層型ゴール指向プランニング

この章では、クロムハウズ AI ([25]、[26]) のプランニングについて説明します。F.E.A.R. のゴール指向プランニングは「キャラクターの動きを周囲の状況に合わせて構築する」というエレガントな問題に帰着させることが出来ました。しかし、複雑な状況で前提条件や効果が明確にすることが出来ない場合には「各ゴール間の呼び出し関係を事前に決定する」という方法を採用します。また、何よりこの方法は、ゴールを「呼び出す」という縦のゴールの繋がりによってゴールの階層化を可能にし、クロムハウズのように戦略から行動までを多層なレベルのゴールをプランニングする場合に適しています。技術的な解説をする前に、クロムハウズについて簡単に紹介し、そこにおいてどのような問題を解決するためにゴール指向プランニングが必要であったのかを説明します。

第1節 クロムハウズ

「クロムハウズ」([27,28]) はプレイヤーがハウズと呼ばれるロボット(20m)を構築し搭乗し、最大6人のチームを組んで、4km以上の四方と複雑な地形、構造物を持つ広大なマップにおいて戦う3Dアクションからなるオンラインゲームです。80を超える多様なマップ(街、山岳、平原など)が用意され、機体の特性によって、幾つかの役割(偵察、防衛、重攻撃、機動力を持つ攻撃、狙撃、司令)を果たすことが要求され、最大15分に渡るチーム間の連携と個人の技量が勝敗を決定します。戦略、戦術性と共に、接近戦における機敏さも必要とされます。



図 3-1 クロムハウズのゲーム画面。プレイヤーによって様々に組み立てられたロボットと、雪山や街など多様なマップが見られる。また、通信塔(左上図、右端)や敵基地(右下図、遠景)がある。

このゲームにおけるA Iの役割は「プレイヤーチームが対戦チームを見つけられない場合に、プレイヤーチームと対戦する」ことです。つまり「AI がチームとしてプレイヤーチームと同等な条件で戦う」ということが要求されます。これは「フィールド内の自由な移動」「長時間の戦略、戦術的な行動を決定する思考」が必要であることを意味します。

以上のゲーム設定とAIへの要求を、幾つかの技術を積み上げることで実現します。

- (1) **自律型エージェント** 自分で情報を収集し、判断し、行動させる技術
- (2) **人手によるゴール指向型プランニング** 目的（ゴール）に対して計画して行動させる技術
- (3) **ナビゲーションメッシュ上の A*アルゴリズムによるパス検索** フィールド上の任意の 2 点間を移動させる技術
- (4) **チーム AI と自律的 AI の対立による自由度の高い人工知能** チームとしての知能と各 AI の判断の双方を両立させる技術

さて、ここでは「**人手によるゴール指向型プランニング**」を集中して解説したいと思います。ナビゲーションメッシュについては第 1 回セミナーを、チーム A I については第 3 回セミナーで解説します。

表 3-1 F.E.A.R とクロムハウন্ズの比較

	F.E.A.R	Chrome Hounds
プランニングの単位	アクション	戦略、戦術、行動
プラナー	前提条件と効果による連鎖	スクリプトによってゴールの呼び出しを規定
ジャンル	FPS	ミリタリーアクションゲーム
要求される A I	建築内におけるオブジェクトを利用した戦闘	フィールド全域における長期的な戦略、戦術行動
知識表現	事実表現、シンボル	NavMesh など。
ジャンル	FPS	ミリタリーアクションゲーム
出版年	2 0 0 4 年	2 0 0 6 年

第 2 節 階層型ゴール指向プランニング

F.E.A.R におけるプランニングと比較してクロムハウন্ズのプランニングには明確な違いが二つあります。それは、

- (1) 階層型プランニングであること
- (2) プラナーが全てスクリプトによってあらかじめ記述されていること

という点です。F.E.A.R のプランニングはアクションという一つの次元で、ゴールをいわば横につなげて行きました。クロムハウنزの場合にも、最終的には、行動に還元されたプランが生成されますが、戦略ゴールから戦術ゴール、振る舞いゴール、行動と、ゴールが指定された形式に従っていわば縦に分解されて行きます。こういった階層構造を持つプランニングを「階層型プランニング」と呼びます。

階層的ゴール指向プランニングでは、大きなゴールをより小さなゴールへ分解して行きます。そして、そのゴールをまた、より小さなゴールへ分解し、その作業を最終的に最小の行動単位になるまで繰り返します。

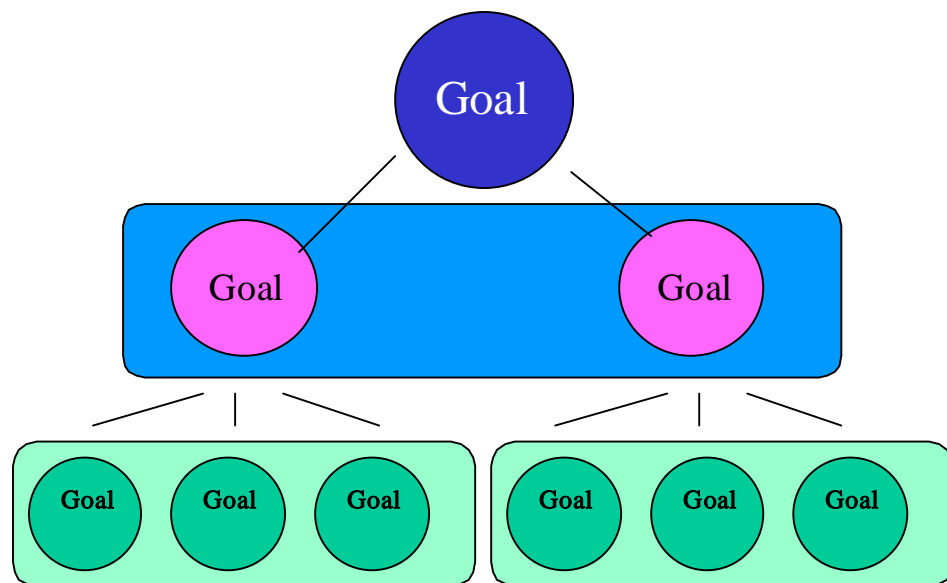


図 3-2 ゴールの階層図。最初のゴールが小さなゴールに分解される。さらに、そのゴールが、より小さなゴールへ分解されて行く。各ゴールの分解過程がプラナーである。そして、各ゴールに対して、分解され方を開発者が直接指定する。ただし、分解の仕方は完全に決定されているわけなく、場合によって別の種類のゴールへ分解されることで、多用な分解のされ方が行動の多様性を産む。例えば、「新宿へ映画を見に行く」では、雨が降っている場合とそうでない場合について、別のゴールが呼び出されている。この図では、左から右にゴールを実行するべきであるとする、行動まで分解されたゴールは、6つの行動を左から順に実行して行くことで、全体としてのゴールを達成する。

身近な例を上げて説明してみます。階層型プランニングは人間がよく使っている思考の方法でもあります。「新宿で映画を見る」というゴールがあります。これは「映画館に行く」「映画を見る」という二つの小ゴールに分解できます。さらに、「映画館に行く」というゴールは、(晴れているなら)「新宿まで行く」と「映画館まで歩く」に分解できます。「新宿まで行く」はさらに...、というように、分解できます。分解して行ったゴールを、順番に従って実行すると、新宿で映画を見ることが出来ます。

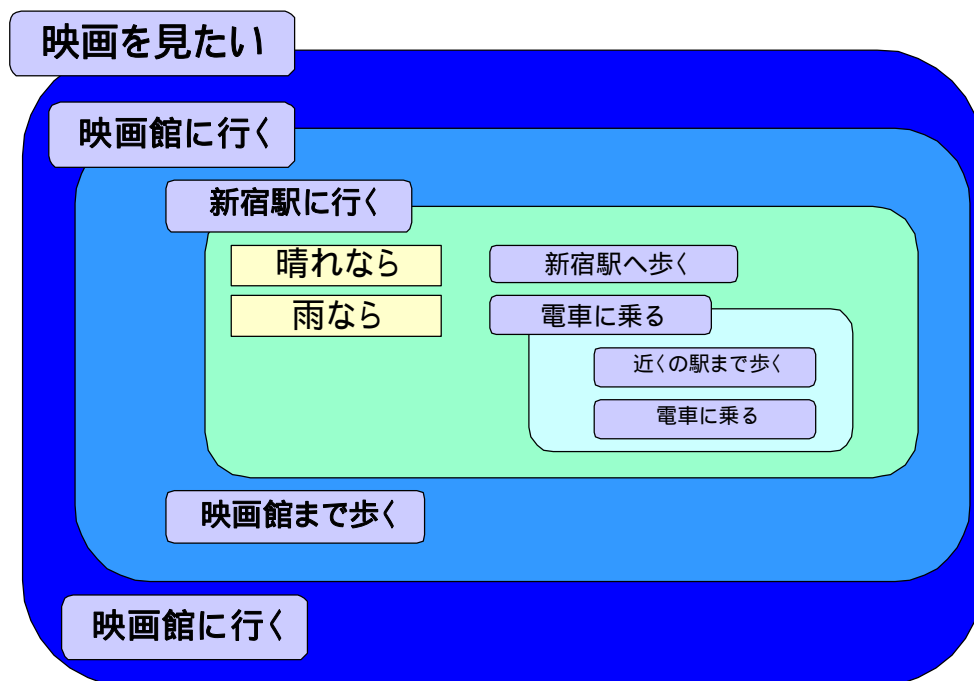
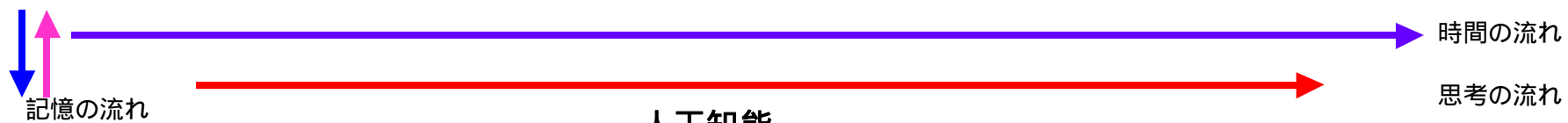


図 3-3 「新宿で映画を見る」というゴールがより小さいゴールへ分解されて行く過程。各ゴールは、「晴れなら」「雨なら」など場合わけを含んだ形でより小さなゴールへの分解のされ方が指定される。

「連鎖によるプランニング」ではプランナーが自動的に行動を組み合わせてくれますが、階層型プランニングでは各ゴールのそれ以下のゴールへの分解の仕方があらかじめ決めておきます。ただし、状況によって分解の仕方を変えておく（「雨なら」電車、「晴れなら」歩く）ことで、ゲーム状況を反映した形でプランが形成されます。

他の例を考えてみます。「工場で車を作る」というゴールを考えています。車は、フレーム、エンジン、タイヤなどの部品を組み立てる次元のプランがまずあります。その次に、各部分を、部品から組み立てる、というプランがあります。例えば、エンジンであれば、たくさんの金属パイプを組み合わせる必要があります。その下には、各部分のナットを締めたり、接合したりする作業のプランがあります。このように、大きなプランをより小さプランへ分解して行きます（[29]）。車を組み立てるというゴールを、ナットを締めたり、接合したりという細かな作業の幾万というアクションの連続によってプランニングすることは不可能です。宇宙ロケットを作ったり、大規模な建築を築いたり、という大きく複雑で多数の工程が絡み合うゴールに対しては、階層的なプランニングが有用です。

クロムハウズもゴール指向エージェントとして階層型プランニングを行動決定機能として持っています。エージェント構造の説明は、次図にまとめるとして、次節からクロムハウズにおける階層型プランニングを解説します。



人工知能

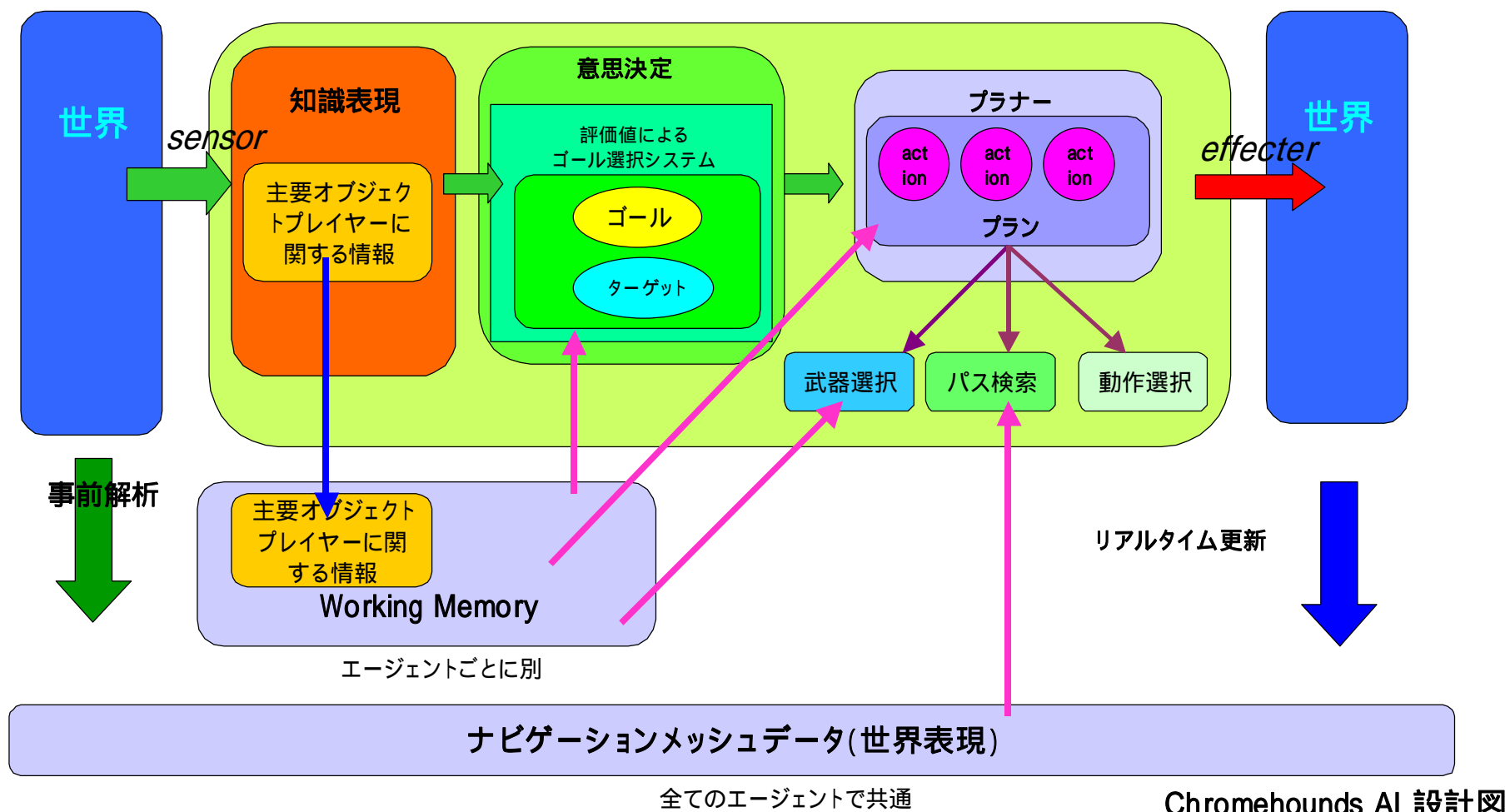


図 3-4 クロムハウنز A I 設計図。ここでクロムハウنزの全体像を説明する。F.E.A.R の全体図とよく比較されたい。

時間に沿って情報の流れを説明する。

まず、ゲーム世界から視覚を通して情報を集める。これは、敵や未発見の基地や通信塔に対して視線が通るかどうかのチェックからなる。

この情報は各エージェントごとに記憶される。

評価値から意思決定を行い標的（敵、敵基地）ゴールを選択する。

選択されたゴールに対してプランナーがゴールを分解してプランを立てる。

武器は敵との距離から武器評価値によって決定する。

標的までのパス検索を行う。

全てが決定したらプランに従って行動を行う。

エージェントの構造を F.E.A.R と比べてみると、知識表現においては、F.E.A.R はずっと発展したシステムを持っており、また、ブラックボードアーキテクチャーという協調のための共有メモリを持っている点が異なる。F.E.A.R のように連鎖を使う場合には、前提条件と効果が知識表現によって厳密に定式されなければならないため知識表現を十分に検討する必要があるが、クロムハウنزでは呼び出すゴールを指定する分岐条件を記述してしまうので、F.E.A.R ほどの知識表現を持っていない。クロムハウنزでは以下のような情報のみを常に更新する。

そのエージェントから見た情報

- ◆ 敵 認識時刻、位置、状態（生、死）
- ◆ 敵基地 本物が偽者か（数個のうち一つだけ本物であるが、確認できるまでわからない）

一般情報（全エージェント共通）

- ◆ 通信塔 どちらのチームが占拠しているか、或いはしていないか。

古い情報は常に新しい情報に更新される。

第3節 クロムハウズにおける階層型プランニング

クロムハウズを例として、階層型プランニングの実装例を説明します。

今、「通信塔を占拠する」というゴールを選択したとします。ここで「通信塔はその周囲に10秒間留まると占領したことになる」という設定になっています。「通信を取る」には「通信塔へ行く」「通信塔を占拠」という二つのゴールが必要ですので、スクリプトでこの二つが呼びされるようにしておきます。さらに、「通信塔へ行く」には「通信塔を見つける」「そこまでのパスを検索する」「パスに沿って移動する」というゴールを順番に達成することが必要ですので、そのように記述しておきます。さらに、この小ゴールは「撃つ」「歩く」「止まる」という行動のプランまで分解されます。例えば、「パスの沿って移動」は「歩く」「止まる」、「通信塔の周囲に10秒いる」は「止まる」コマンドを10秒間呼び出すことからなります。このように、どんな複雑なゴールも階層を経て、たった3つの行動の組み合わせへ分解されます。つまり、機体制御部分から見れば、時間に沿って3つの行動の何れかが指令として降って来るだけです。これが「階層型プランニング」の方法です。このような分解が条件分岐を含めてゲーム中で指定した通りに実行されます。

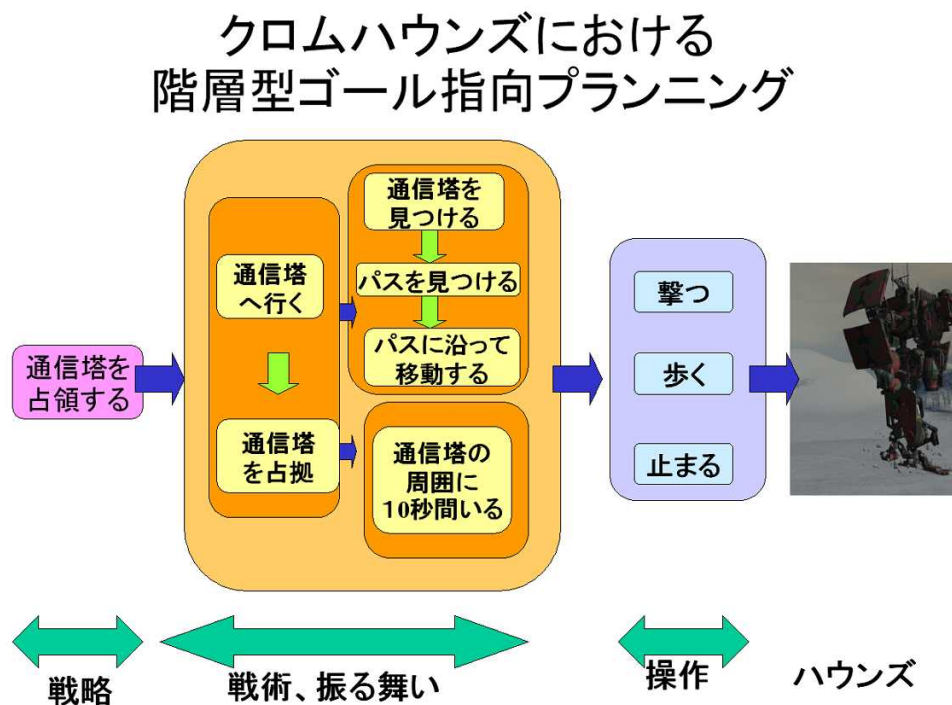


図 3-5 クロムハウズにおける階層型ゴール指向プランニング。大きな一つの戦略ゴールがより小さな戦術ゴールへ分解され、さらに小さな「振る舞い」に分解され、最終的には、タイミングに指定された「撃つ」「歩く」「止まる」の行動の指令となってCOMに渡される。開発の順番としては、まず「撃つ」「歩く」「止まる」というコマンドを実行するシステムを作り、そこからボトムアップにゴールを積み重ねて行く。

第4節 クロムハウズにおけるリプランニング

F.E.A.R.においてリプランニングということを学びました。それは、プランを立てた後、状況が変わったときにプランニングをやり直すことでした。クロムハウズの場合は、状況の変化に対して全体のプランを立て直して対応するという事はせず、中間ゴールを動的（ダイナミック）に作ることで対応します。

例えば、上記のプランで急に敵が現れて攻撃して来たとします。それをCOMが認識すると、二つ目の層

に「敵を攻撃する」というゴールが一番上にスタック（載せられ）されます。このゴールは、自身がクリアされるまでCOMを制御し続けます。そして、クリアされると、もとのゴールを再び実行し直して「通信塔を占拠」のゴールを継続します。

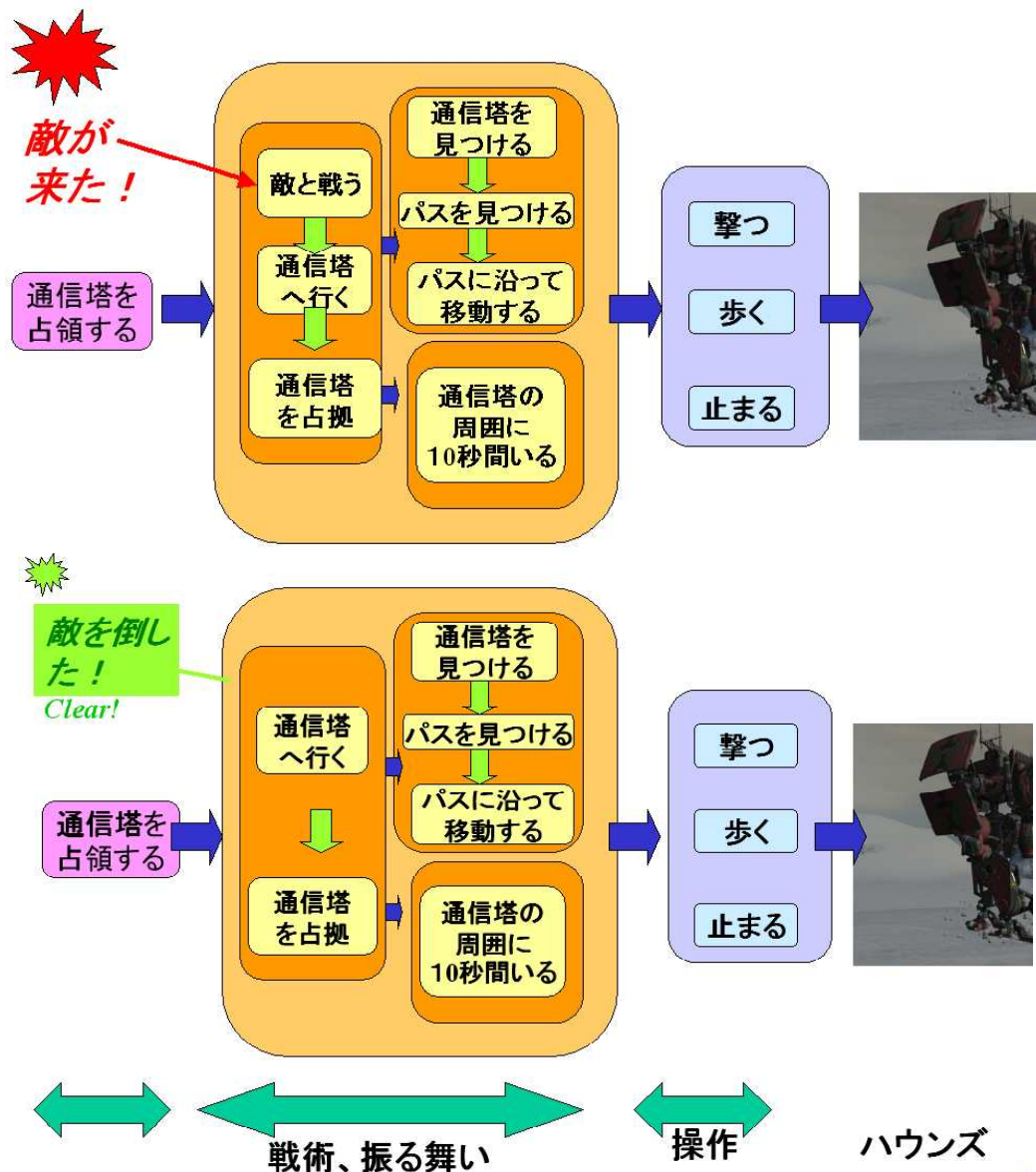


図 3-6 クロムハウズにおけるリプランニング。 クロムハウズでは、状況が変化しても、プランを最初から立て直すことはせず中間の層にゴールを挿入することで対処する。各階層のゴールはスタック状に積まれており、上から順番に実行して行くが、状況に対処する必要がある場合は、現在のゴールをサスペンドして、例えば、戦術層のゴールスタックの上に、新しいゴールを載せてそれを実行する。挿入したゴールが達成されれば、もと実行していたゴールの実行へ戻る。(注：スタックとは「積み上げたもの」という意味です)

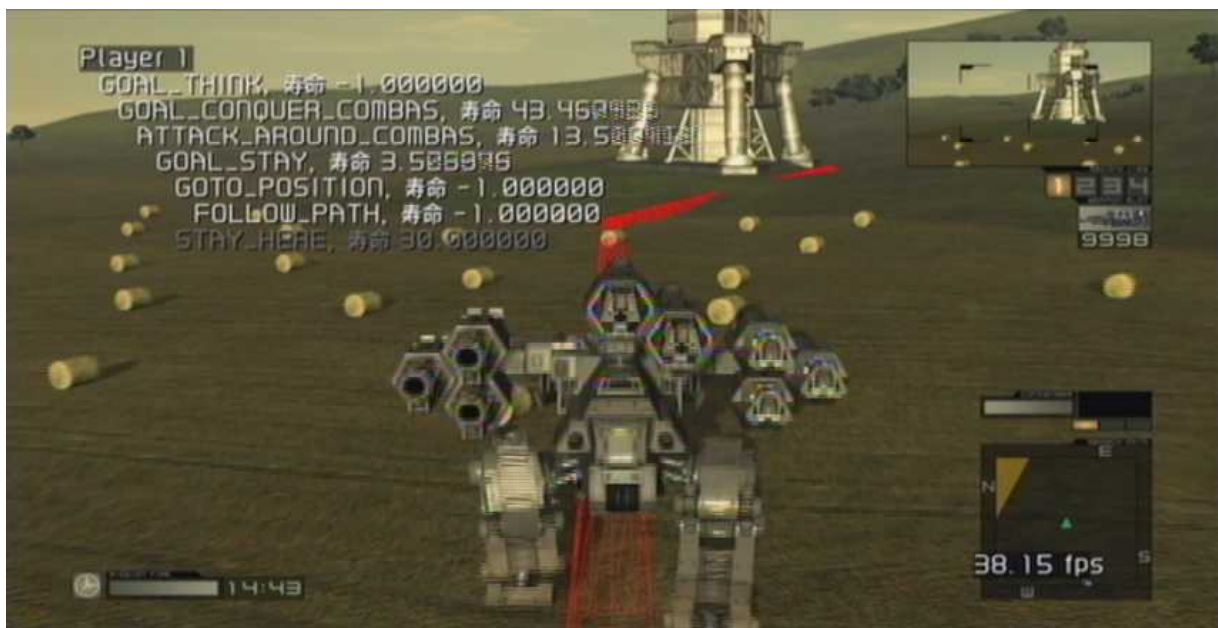


図 3-7 ハウンズがプランに従ってゴールを実行して行く様子。左上にデバッグ画面が表示されている。上から下へ大きなゴールが小さなゴールへ分解されている。数字はゴールに指定した寿命を設定している。設定した時間内に達成しなければ、そのゴールは失敗したと見なされる。ゴールが達成不可能になったときに、そのゴールに捉われたままであることを防ぐためである。赤いラインは、パス検索によって導かれた通信塔までのパスである。



図 3-8 基地攻略戦略「Conquer_Base」を実行中のハウンズ。敵影もなく、「Walk_Around」(敵基地進行)である。右上のゴール表示の一番下の敵基地攻撃は「Attack_around_base」は、「Walk_Around」(敵基地進行)が達成(クリア条件=敵基地に近くまで来て視線を通すことができる)された後、Activate(活性化されて)されオンになる。今はオフで予定に組み込まれているだけだ。



図 3-9 途中で偶然、敵と遭遇してしまったハウズは「Attack_target」(敵攻撃)を戦術ゴールに急速、挿入して対処する。「Attack_target」のクリア条件「敵をやっつけるか、追い払う」まで、このゴールが実行（プロセス）され、クリアされるもとの状態に戻る。左上のゴール表示に「Attack_target」が挿入され、もとあった「Conquer_base」ゴールが InActive（不活性化）されていることがわかる。

第5節 ゴールの作り方

第3、4節でゴールの分解のされ方とプラン生成について理解して頂けたと思います。これを基本システムとして、今度は、ゲームに適應するために内容を作って行かねばなりません。ゴール指向プランニングでは、これはゴールを増産して行く過程となります。システムを作るのは技術者の仕事ですが、A Iの形を決めるのはゲームの設計者で企画の方の仕事であり、クロムハウズでは基本システムが出来た段階で、幾つかの簡単なゴールを見てもらった後、ゲームに必要な全ゴールの仕様書を作成してもらいました。

仕様書が上がると、今度は「どのゴールをどの階層に入れてどう呼び出すか」という呼び出し方の仕様書を作ります。そこからは、一つ一つのゴールの仕様書をプログラマーにスクリプトに起こしてもらいます。そして、一つのゴールを実装するたびに実際に動作させてテストを行い、そのゴールが適切であり、設計通りに実装されているかを全員で確認します。また、逆に設計における不備が指摘されたりします。だいたい毎日一つずつゴールを実装して行きます。そして、きちんと実装されていることが確認できれば、次にどのゴールを入れるかを検討します。そうやって、漸近的にゴールを増やして、だんだんとA Iシステムを発展させて行きます。

ゴールは結果として階層に分けられ（開発中は必ずしもそうではない）特に上位の階層へ行くほど自由に変更することが出来ます。下の層は上から呼び出される側であるので、その変更は、それを呼び出す全てのゴールへ影響します。よって慎重に変更する必要があります。しかし、そのゲームにおけるA Iの行動を規定しているの最上位の層を自由に変更することで、A Iの戦略を直接変えて行くことが出来ます。この部分が、常に自由に変更できることは、A I開発に非常に大きな自由度を与えることになります。

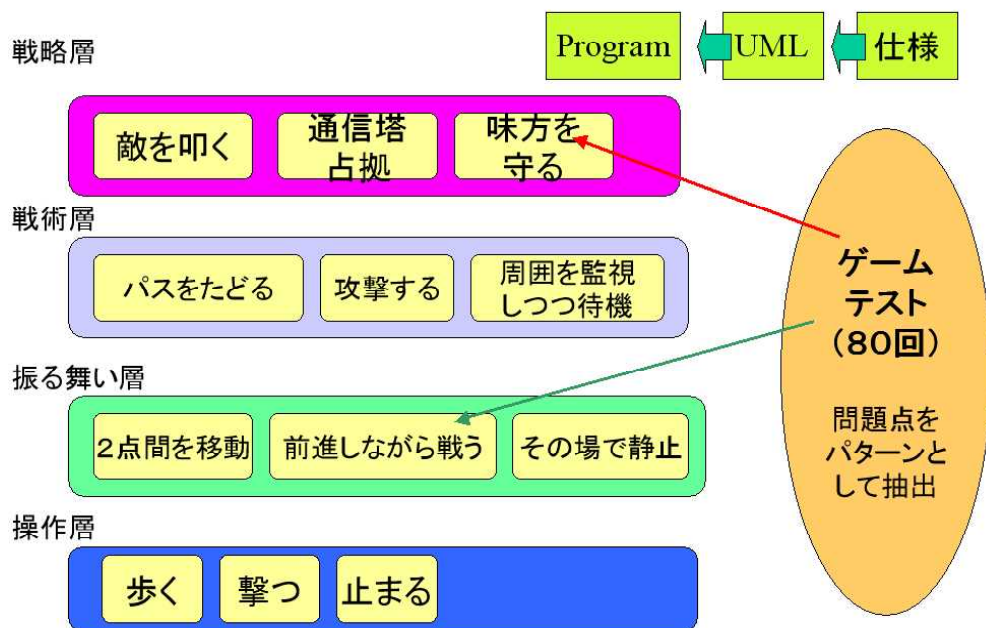


図 3-10 最初の段階では、操作層の3つの行動だけが実装される。これは実際にコーディングされる。そして、スクリプトによって簡単な振る舞い層を定義して、さらに戦術層、戦略層を作っていく。



図 3-11 クロムハウズにおける最終的なゴール総合図。各ゴールは、自分の属する層が同じか、それ以下の層の小ゴールへ分解され、最終的には、操作層の3つの「動作」へ分解される。「操作層」の3つの行動は、COM の実際の動作制御へ渡される。つまり、操作層の下には、機体制御部分にあたる。機体制御部分から見ると、3つの命令がいろいろな順番で渡されて来るようにしか見えない。このように、AI を構築するときに、多層化を行うと、複雑なシステムを見通しよく作ることが出来る。このような AI を、多層化 AI (multi-tiered AI) と呼ばれる([30,31])。ちょうど人間の腕の関節が、その運動の自由度を高めるように、知能も多層化をすることで、各階層の独立な発展が可能になり、知性としての自由度が高めることができる。また、ゴールの再利用することも可能である。それは、開発を効率化する。また、最上位の階層を変えるだけで、新しいMOD を作ることが出来る。

クロムハウズは結果として4つの階層を持つゴールシステムとなりました。各層のゴールは、同じ層か、それ以下の層のゴールから組み立てられます。例えば、「味方を守る」というゴールは、「合流する」「攻撃する」というゴールからなります。また「攻撃する」は、「静止する」「後退する」「前進する」「撃つ」の組み合わせから実現されます。その分解の仕方は技術的な領域になりますので、技術側が行うべきであると考えて技術者が行いました。最終的に、全てのゴールは、操作層の3つの単純な動作「歩く」「止まる」「撃つ」まで分解されています。

第6節 ゴールの実装方法

さて、この節はプログラマーを対象に解説したいと思います。プログラム構造の話ですので、プログラマーでない方にも、眺めて頂けるだけで構造がわかるように説明します。

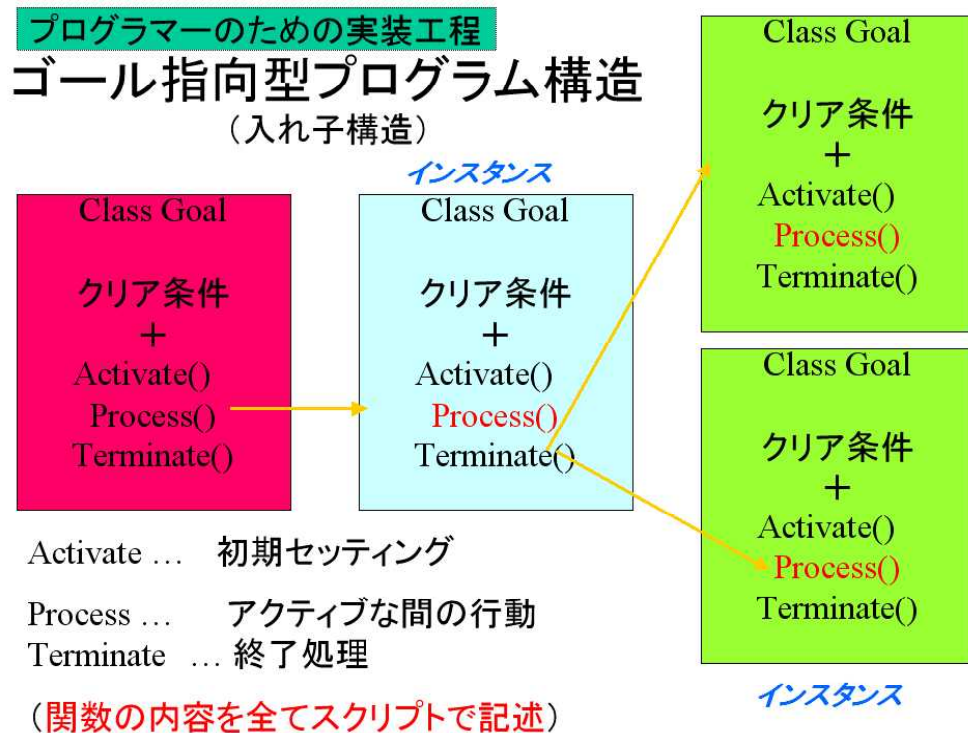
ゴール指向プランニングについては、Mat Buckland, “Programming Game AI by Example” ([10]) の第9章、及び、そのデモ、ソースコード ([32]、ダウンロード可能) から多くのことを学ぶことができます。ゴール、複合ゴール(Composite Goal)、ゴールをスタックする方法など、多くの本質的な技術が解説されています。Mat Buckland([33])は人工知能技術をゲームに導入するという情熱を長い間持ち続けている方で、この教科書からは、彼のそういった情熱を感じ取ることが出来ます。この節では、彼が作ったゴール指向のためのプログラミングのフレームに従ってゴール指向のプログラミングを説明します。

まず、各ゴールは(C++の)クラスとして実装します。各ゴールには、3つのメンバー関数を用意して、それぞれ、次のような役割を持ちます。

- **Activate** ... ゴールが呼びだれたときに実行される関数。
- **Process** ... そのゴールが呼び出されている限り、くり返し実行される関数。
- **Terminate** ... そのゴールが消されるときに呼びされる関数。
- クリア条件 そのゴールクリア条件。この条件が満たされれば、Process 終了。

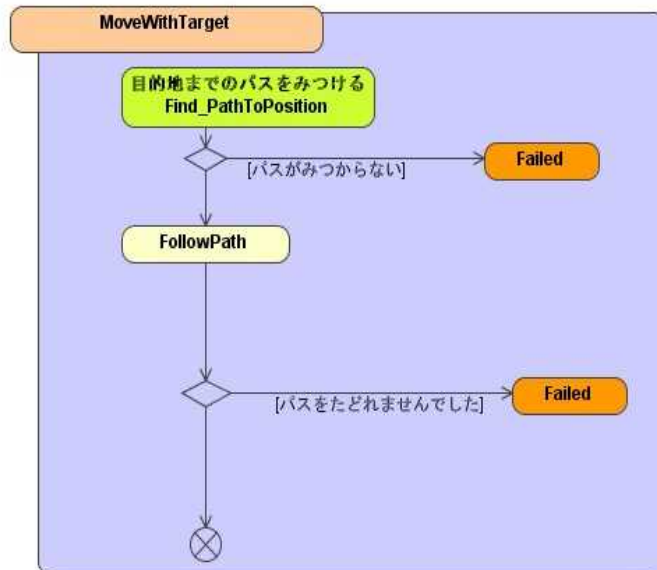
まず、呼びされたゴールは、Activate を実行し、クリア条件が満たされるか、ゴールが失敗して Terminate されるまで、Process 関数を実行します。Activate 関数には、呼び出すゴールをスタックします。つまり、ここには、このゴールが為すべきことを記述します。Process 関数は、主に、クリア条件が満たされていないかをチェックしながら、スタックしたゴールを順番に実行します。逆に、最下層の行動は、コマンド・キ

ユー方式で、指定された順番に実行します（FIFO）。



各ゴールは、それに続く小さなゴールを呼び出し、全体として構造はクラスの入れ子構造になります。デザインパターンでは、これをコンポジット・パターン（[34]）と呼びます。クロムハウন্ズでは、この関数部分をスクリプトでカスタマイズできるようにして開発のスピードを高めるとともに変更しやすいようにしています。実際のコーディングの方法を巻末の Appendix で説明します。

Move_WithTarget



Level:

Compoiste Goal

Goal: 指定したポイントに
ターゲットとともにたどりつく

変数: range: ターゲットと自分の束縛
距離

min ... 最小許容距離

max... 最大許容距離

limit_time: 作戦実行時間

クリア条件: ポイントにターゲットが
たどりつく

Note: 敵をひきつける罠行動のとき、
或いは、
友ハウズについて
行動するときを使う。

Desirability:

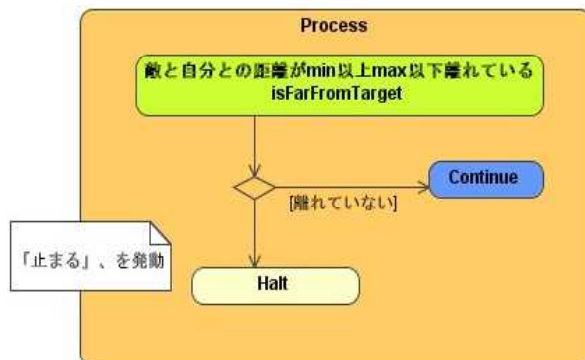
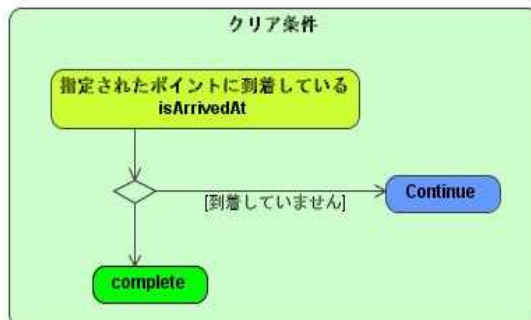


図 3 -12 クロムハウズにおけるゴール仕様書。上図は結局、実装されなかったゴール「Move_WithTarget」の仕様書であるが、どのような処理をして、他のどのゴールをどの場合に呼び出すかが指定されている。各ゴールに対して、このような仕様書が作成され、最終的にはスクリプトに起こされて実装される。Failed はゴール失敗、Continue は、プロセス関数を実行し続けることを意味する。

第7節 意思決定

さて、ゴールを全て実装すると、次に COM が「状況に合わせて最適なゴールを選ぶ」ための「意志決定機構」を作る必要があります。つまり、最上位の階層、戦略ゴールの何れを選択すべきか、という問題です。それは、どのように決定すればいいのでしょうか？ 目標はゲーム状況に即したゴールを選択する意志決定機構を作ることです。

まずゴール同士を比較できる手段が必要です。しかし、この世界で比較ができるものは、数以外ありません。そこで、ゴールに点数をつけることで、ゴールの優劣を比較することを考えます。これを評価関数の方法と呼びます ([10] P.398-402)。

では、よいゴールとは何でしょうか？ 日常でも、人はどんなゴールがもっともよいと考えるのでしょうか？ クロムハウズでは、「リスクと報酬の考え方」に基づいて「ゴールを達成する危険度が少なく、勝利のために見返りが多くあるゴール」を実行に値するゴールであるという考え、ゴールの危険度を R 、重要度を S とし、評価値を $S \times (1 - R)$ という式で定義します。

そして、各ゴールに対して、 S と R の関数を決定して行きます。ただし、この S と R は、ゲーム状況と COM の状態や性質を反映する必要がありますので、見かけ上は非常に複雑な式になります。これを、ある通信塔を占拠するというゴールに対する評価値を例として解説しましょう。

まず、通信塔の重要度を決めます。重要度のファクターを、開発者で相談します。すると、結論として、3つのファクターがあることがわかりました。

- 味方司令部に近い通信塔は重要（司令部の通信範囲を広げるから）
- 敵司令部に近い通信塔へ重要（敵に対して重要なので占拠する価値あり）
- 隣が占拠してある通信塔なら重要（通信をつなげられるので）

この3つに対して、評価式を作って、ウェイトをかけて足し合わせます。ウェイトの調整は、最終的に企画が行います。

危険度も同じように、ファクターを上げて、ウェイトを調整して足し合わせます。そして、評価の式 $S \times (1 - R)$ に入れて点数をつけます。これは、全てのゴールのターゲットに対して行います。例えば、通信塔を占拠ならば、マップ上にある全ての通信塔に対して、「通信塔 A を占拠する」が何点、「通信塔 B を占拠する」が何点、「敵を攻撃する」なら「敵 A を攻撃する」が何点、「敵 B を攻撃する」が何点、というように計算します。

結果として、戦略ゴールを各ターゲットと組み合わせて40近くのゴールを評価することになります。それでも、意志決定のタイミングは、数秒から2, 3分に一度ですから、大きな負荷にはなりません。

エージェントが意思決定をする仕組み

その戦略を達成することで得られる **見返り(S; 重要度)** と、それを達成するための **リスク(R; 危険度)** の兼ね合い

$$\text{実行評価値(E)} = S * (1 - R)$$



図 3-13 意思決定の概念図。見返り (S) に比例して、危険度 (R) に反比例するように評価値を組み立てる。

通信塔の「危険度 R」



3つのファクターによる。

- (1) 敵ハウズが通信塔からどれぐらいの距離にいるか。
- (2) ザコ敵がどれぐらいの距離にいるか。
- (3) 味方ハウズが通信塔からどれぐらいの距離にいる

$$R = W_1 * \text{敵ハウズの通信塔との距離による関数} \\ + W_2 * \text{ザコ敵と通信塔の距離による関数} \\ + W_3 * \text{味方ハウズ通信塔との距離の関数}$$

W ... 重み

パラメーターと関数の形を調整する

意志決定の形やハウズの個性が決定

テストをくり返しなが

計 100 近くのパラメーターを調整

図 3-14 危険度の計算例。ここでは「通信塔占拠」のゴールを評価している。

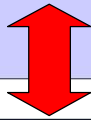
通信塔の「重要度 S」

3つのファクターから決まる。

- (1) 味方司令部との関係（通信塔- 敵司令部）
- (2) 敵司令部との関係（通信塔- 味方司令部）
- (3) 通信塔同士の関係（通信塔- Combus）

$$S = W_1 * \text{味方司令部との距離による関数} + \\ W_2 * \text{敵司令部との距離による関数} + \\ W_3 * \text{隣の通信塔の占拠状態からなる関数}$$

W ... 重み



$$S = W_EBase * (\sum F(L_EBase, L_MapScale) * Est_Base_NonConstFactor \\ + W_SBase * F(L_SBase, L_MapScale) \\ + W_InComNet * Est_InComNet)$$

$$W_EBase + W_SBase + W_InComNet = 1$$

$$Est_InComNet = W_static * Est_Static_Combus + W_dynamic * Est_Dynamic_Combus$$

$$Est_Static_Combus = (Connectable_Number - Connected_Number) / Max_Connectable_Number$$

$$Est_Dynamic_Combus = Connected_Number / Max_Connectable_Number$$

$$W_static + W_dynamic = 1$$

...

図 3-15 重要度の計算例と実際の計算式。「通信塔を占拠する」という単純なゴールでも、敵の位置、味方の位置、それぞれの基地の位置からその重要性を計算する。

以下は評価関数の調整についての詳細を解説します。クロムハウズにとっては、中心的問題なのですが、あまりに詳細でありますので、次の節へ飛ばして読んで頂いても問題ありません。

一つの評価式の中には、幾つかの固定パラメーターがあり、これによって評価関数の形を決定します。例えば、上記の例なら、自分の基地に近いほど重要度が高いわけですから、距離に反比例して重要度を上げる関数をSは含んでいます。その反比例の関数の形、特にクロムハウズは、全て線形関数によって評価関数を作っていますので、その反比例のグラフの傾きを決定するパラメーターを調整する必要があります。全体では、S,Rを定義する評価式を調整するパラメーターは100近くあり、これを相対的に調整して行きます。調整の指針として「全てのゴールが互いに競合する状態を実現すること」を目標にします。

競合とは、一方的な戦いにならず、常に抜いたり抜かれたりする程度に、よい戦いをすることを意味します。これは、人間の心理のモデルを参考にしています。人間の心も、幾つかの欲求を心の中で競合させておくことで、それぞれの欲求が対応することを可能にしています（[35]）。例えば、スポーツをするときのこと

を考えてください。テニスにおいては、サーブを受けるときに、「右へ走るか」「左へ走るか」という行動への欲求を競合させながら持っているからこそ、両方の場合にすぐに適応することが出来ます。或いは野球においても、内角を打つか、外角を打つか、欲求を競合させながらバッターボックスに立ちます。同じように、クロムハウنزでも、普段はそれぞれの戦略ゴールが競合しながら存在し、行動が迫られる状況においては、その状況に合ったゴールが大きな評価値を持ってイニシアチブを取るように調整して行きます。

もちろん、この調整は、クロムハウنزA Iの全システムの中で最も難しい問題です。調整のためのテストは、実際にプレイヤー6体、A I6体を15分間対戦させて行います。二つの点で、評価システムをチェックします。

通常の状態、ゴール評価値が競合しているか？（極端に低い値、高い値はないか？）
戦闘時など、状況が限定された場合、適切なゴールの評価値が高くなっているか？

対戦の結果をミーティングで検討することで「あの場面でA Iはこういうふうに行動した」「こういうふうに行動しなかった」という欠点を見出してパラメーターを変更して行きます。A Iの中核たるこの部分は、A Iの出発点であり、エンジン部分にあたります（私はこの部分を関数エンジンと呼んでいました。それは、数十の入力パラメーター、100近くの調整パラメーター、ゴールごとの関数、そして、たった一つのゴールの決定。A Iの頭の中は、関数が作るカオス系になっているからです）。つまり、これ以上は還元できないものです。そこには、正解というものがありませんので、調整者の個性がある程度反映される部分でもあります。



図 3-16 ゲーム中で競合する戦略ゴール評価値たち。現在は、敵基地占領ゴール「Conquer_base」が最大の評価値を持ち、この画面の少し前のタイミングで意思決定を行ったときに選択された。「Player1」という文字の下が実行中のゴール階層図で、160秒の寿命で「Conquer_base」が実行中である。リストの下4つ

のゴールはゲーム開始時だけに選択できる初期ゴールで、かなり長いプランが実装されており、序盤のゲームメイキングに大きな役割を果たす。例えば「Conquer_base_with_combas」は通信塔を占拠しながら敵基地へ進め、というゴールである。もちろん、敵基地はオレンジ色の1～3でAIにとっても行って確認するまではどれが本物かわからない。この点は、ユーザーと平等な条件になるように非常に気を使っている。また、自分の基地を守る「Defend_my_base」、基地偵察「Scout_base」の評価値が低いのは、このAIがアタッカー型（攻撃タイプ）のAIであるからで、攻撃的な意思決定をするように、破壊型のゴールの評価値が相対的に高くなるように調整されている。

また、ひょっとするヒントになるかもしれないので調整の感覚的なことを書いておきますと、例えば、通信塔を占拠するというゴールならば、近くのゴールほど強い引力（重要度が高い）を持つようにイメージします。つまり、調整パラメーターは評価関数というポテンシャルの形を決定しているようなイメージです。また、敵がいれば危険だから近付かないというのは斥力のポテンシャルのイメージです。そして、ウエイトを掛けて足し合わせるというのは、ポテンシャルの重ね合わせに対応します。

（注）ポテンシャルというのは物理学の概念で、それぞれの場所の引力の強さを関数の傾きとして現します。

遺伝的アルゴリズム、ニューラルネットワークと言った最適化の手法で、大規模なパラメーター調整を自動化できたら楽なのですが、100を超えるパラメーターをゲームという複雑な対象に対して自動調整することは不可能です。まず、開発者自身でさえ、これが正解だ、という行動の全てを挙げることが出来ません。それぞれの開発者ごとの正解があり、最終的には、その人の判断に委ねられます。そして、それをコンピューターに伝える方法も今のところありません。開発者はこのテストを通じて、このゲームが真に求めるAIの姿を見出そうとしました。どうすれば、このゲームに合うAIを作れるのか、テストを通じて我々はいわばゲームと対話を重ねていたのです。いや、それはもうAIの開発の最初から始まっていたことかもしれません。

パラメーター調整の部分を自動化することが可能なのか、果たしてそれがよいことなのかさえ、判断することは難しいことです。私は、企画の個性を十分にAIの投影できるシステムを作ることまでが技術者の仕事だと考えています。システムに不備があった場合、十分にその意図を反映させることはできません。しかし、十分に企画の意図や個性を反映できる懐のあるAIシステムを作れば、後は企画の方にとことんまで作り込んで頂ければと思います。

さて、オンラインゲームに限らず、基本としてゲームのAIには必勝法があってはいけませんが、このシステムでは、ユーザーからパターンを読まれることはありません。数十の情報を認識して、数十のゴールの中を、100近いパラメーターで評価式が計算されながら、一つのゴールが決定され、状況に応じたプランニングがされるAIにおいては、毎回行動が創造され、同じ行動が偶然作られる確率さえとても小さいものです。このような複雑系にパターンがあるとすれば、逆に、このシステムにまずいところがあるということです。例えば、あるゴールの評価値が、常に高い値を取るような実装のミスがあれば、それはパターンとなって現れます。

第8節 キャラクターごとの個性付け

さて、上記の意志決定機構の上に、キャラクターごとの個性付けを行います。

キャラクターごとの個性付けは、

- (i) 各機体特性を反映するもの
- (ii) ロールタイプを反映するもの

の二つがあります。

まず(i)から説明します。クロムハウズは自分自身でパーツから機体を組むことが出来ます。機体を組むとその重量から足の速度、攻撃力、守備力などが決まります。戦略ゴールの評価式の中には、そのパラメータを使った項が存在します。例えば、友達(味方)を助ける「Resque_friend」というゴールの評価式の中には、「味方までの距離 / 自分の足の速度」を変数とする項があります。つまり、足が遅い機体ほど助けに行くのに時間がかかるので、評価値が下がるようにしています。逆に足が速い機体は評価値が高くなります。また味方を護衛「Protect_target」というゴールは、十分に攻撃力があって機動力のある機体が行って初めて意味があるので「スカウター」などではこの評価値はとても低い値になっています。つまり、自然に機体から意志決定において個性づけが為されるような仕組みになっています。開発者としても、自分が組んだ機体がどんな個性を見せてくれる、わくわくしながらテストを重ねました。

次に(ii)を説明します。クロムハウズには6つのロールタイプ、「アタッカー」(攻撃)「ディフェンダー」(防御)「スカウター」(偵察)「スナイパー」(狙い撃ち)「ヘビーガンナー」(重火器攻撃)「コマンダー」(司令)があります。ロールタイプは最初から決まっているものではなく、機体構成の結果として定義される特性です。例えば、重火器を積んでいると重量から足が遅くなるので「ヘビーガンナー」タイプ、逆に足が速ければ「スカウター」タイプ、というようにです。チームとして行動する場合、そういったロールタイプによって役割分担をして、初めて勝機が見えるというのが、クロムハウズというゲームの基本コンセプトです。そしてAIにおいても、「ロールタイプの特徴を出す」ということが最上位の課題の一つでした。

ロールタイプの個性を出すために、戦略ゴールの評価値を計算した後に、各ロールタイプ毎のウェイトをかけました。例えばアタッカーなら攻撃的なゴール「Attack」「Conquer_base」などを相対的に高くし、ディフェンダーなら防衛的な「Defend_my_base」などのゴールを相対的に高くしています。スカウターには「Scout」というゴールがあり、これは他のロールタイプには、ウェイト0にして選択できないようにしています。F.E.A.R では、各キャラクターしか取れない行動の単位を用意しました。クロムハウズでは、各ロールタイプでしか取れないゴールを用意します。また、F.E.A.R では、各行動にコストを設定してプランを調整しました。クロムハウズでは評価値にウェイトをかけて意思決定をコントロールします。

第9節 オフラインミッションへの応用

前節では、AIが自分でゴールを選択する仕組みを説明しました。クロムハウズにはオフラインミッションとして、各ロールタイプの練習を行うモードがあります。その中のコマンダーミッション(司令官となってCOMを操る)においては、COMに命令を下すことで、ミッションを遂行します。ユーザーは用意されたインターフェースを通じて「敵基地Aを攻撃しろ」などCOMに命令を下します。実は、そこでは、命

令は戦略ゴールに対応しており、ユーザーが命令することで指定された戦略ゴールを COM が実行する、という仕組みになっています。

第10節 チーム AI へ

さて、「個としてエージェント」を解説して来ました。ゴール指向エージェントとして、ゴールを用意し、プランニングシステムを構築し、意思決定機構を作り、移動に対してはナビゲーションメッシュを使って任意の点の間の移動を可能にしました ([25])。そして総合して、クロムハウنزのフィールドで自律型エージェントとして活動する能力を得ました。しかし、ハウنزにはまだ「チームとしてプレイヤーの相手をする」という使命があります。

エージェントたちをチームとして行動させる技術として

- ◆ チーム AI
- ◆ マルチエージェント
- ◆ 群知能

という3つの技術があります。これについては、第3回のセミナーで解説します。特に「マルチエージェント」は個としての自律型エージェントの実現という礎の上に立って始めて実現できる技術です。

第11節 まとめ

この章では、「人手によるプランニング」による「階層型プランニング」を、クロムハウنزを例に説明しました。「連鎖によるプランニング」との違いは、ゴールが次にどのゴールを呼び出すかを、開発者が指定している、という点です。また、ゴールに階層を設けることで、抽象的なゴールを定義することが可能になりました。クロムハウنزでは、戦略、戦術、振る舞い、という層を形成しました。

また、ゴールを選択する評価値による意思決定機構を作りました。この機構によって、AI は自分自身で行動を決定する、即ちここではゴールを選択する、という能力を持ちます。これを自律型 AI と言います。さて、もう一步踏み込んで考えてみますと、このゴールを決定する部分を、他の AI から指定させると「連携」の機能をもたらすことになり、また、司令官のような AI が全員のゴールを決定するなら、チームとしての統制の取れた集団を構成することが出来ます。つまり、個としての AI の上に、集団としての AI の可能性が自然と見えて来ます。

また、企画とプログラマーの開発工程における仕事について解説しました。企画は、全体のシステムを理解した上で、必要なゴールのリストを作成します、そして、それが実装された後は、テストをくり返しよく観察することで、欠点や不備を見抜ながら調整をくり返します。クロムハウنز AI では、その中心の意思決定機構において、企画の個性が反映できるシステムを目指しました。結果として、それは100を超えるパラメーターの調整となりました。また、プログラマーには簡単な実装構造を紹介しました。巻末の付録には、デモの紹介とそのコーディング方法が解説してあります。

開発期間中、このシステムは70回以上に渡って人間のチームと対戦を重ね、調整されて来ました。そこで、感じられることは人間の知性のしたたかさです。この AI を作る前、何度となくプレイヤーの後ろに立って、人間が何を考えながらプレイをしているかを観察しました。なるべく人間に近づけるべくシステムを

組み上げて来ましたが、人間の知能はとても複雑です。まず、人間は必ずしも単一の思考をしません。一つのプランだけでなく、同時に複数のプランを立てて行動します（[35]）。例えば、目の前の敵と交戦していても、さりげなく基地の方へ近付きつつ、タイミングを見計らって敵基地へ前進したり、通信塔を占拠しても、それは、あるプランでは通信を確保するためであり、あるプランでは占拠した通信塔の数を増やすためであり、ある意味では敵への牽制であったり、単一の行動の下に並行して複数のプランを重ねて思考しているのです（第1章の「新宿へ映画を観に行く」の例で言えば、「映画見るか、本屋行くかわからないけど、とりあえず新宿へ行っとく」という感じです）。そのために、あるプランが駄目になっても、瞬間的に並行していたもう一つのプランに乗り換えることが出来ます。AIのプランニングと比較すると、そういった知性の柔軟性を思い知らされます。ただ、そういった人間の賢明さも、AIを作って対戦してみて初めて実感できることでもあります。

人工知能の探求には二つの目的があります（[1] 26, 27章）。一つは、人間の知性を真似た知性を作ること、もう一つは、人工知能を作ることによって人間の知性をより深く理解することです。人工知能はある面では工学であり、ある面では理学であり、その点が他の技術的分野とは異なります。

ゲーム開発であるからと言って、その点が異なるわけではなく、むしろ、プレイヤーという知性と真剣に向かい合える場として、ゲームは人工知能から見て非常に大きな可能性を持った魅力的な分野です。そして、その可能性を引き出すために、多くのAI技術があります。プランニング技術も、F.E.A.R と クロムハウنزで異なる可能性を引き出しました。とすれば、ゲームが異なれば異なるだけ、そこには異なる可能性が潜んでいて、たくさんのゲームにおいて、プランニングが応用される機会を待っているかもしれません。その可能性を見抜く目を持つことが、今後、企画にとっても、技術者にとっても、創造的な仕事をするために必要なことであると思われます。

[第3章 おわり]

第4章 自分の開発するゲームのために

第1章でプランニングの基本について理解し、二つの方法「連鎖によるプランニング」と「人手によるプランニング」を解説しました。前者は2章でF.E.A.R を具体例として説明し、後者は3章でクロムハウنزを例として説明しました。開発者にとって理解することだけでなく、自分の開発タイトルに応用することが目標です。この章では、実際に自分の開発するタイトルへつなげて行くために、ゴール指向とプランニングそれぞれについて、さまざま応用の方向を考えてみましょう。

ゲームAIにとってプランニングの本質は「行動を創造する」ということにあると考えられます。完全に準備した行動ではなく、用意された行動の単位をAI自身が組み合わせて新しい行動を作って行くこと、そこにプランニングの真髄があります。ゲームデザイナーによっては、どうしても作品の完成度を上げるために、一から十までNPCの行動を規定したいと思うものです。実際そうやってゲームの完成度を上げることで優れた作品を世に送り出して来ましたが、プランニングは、その「COMの行動の作成」領域をある程度自動化し、また、時に開発者の予想できない多様な行動を産み出します。それはCOMに高度な能力を与えると同時に、予測できない行動を生み出すという危険をも生み出します。デバックではこの獲得した自由度をどの

ようにチェックするかが課題となります。第1回の世界表現でも述べたように、新しい技術には新しいデバッグ方法が必要です。

いざプランニングからゲームを考えるとと言われても、アイデアがすぐに思い浮かぶとは限りません。逆に、慣れてしまうと、プランニングという考えの方が自然に思えて来ます。少しゲームから視野を広げて、生物界における知性を考えてみます。昆虫の行動は、殆ど環境に対する反射からなると言われています（[36]）。それが、彼らに敏捷性を与えています。逆に、哺乳類など動物に近付くと、反射的な知性を内包する形で（Brooks のサブサンプション構造[37]）、未来のことを考えて行動するようになります。冬に備えて餌たくわえたり、一日の予定を立てて行動したり、10年後のことを考えて行動することもあります。我々人間は、そういった計画を立てて行動する生き物であり、ゲームの中の敵が自分たちと違う知性を持っていると感じる一つの理由は、彼らがたとえ人間の姿をしていても、刹那的で反射的な行動を基本としているからかもしれません。このようにプランニングはとても我々に身近な方法です。

ここまでの例で、そういったプランニングが簡単な行動の組み合わせからなることを見ました。一つ一つの行動は単純で意味を持たなくても、幾つかの行動を組み合わせると、不意に一つの抽象的な意味を持ち始めます。F.E.A.R. では、「武器を拾い」「装填し」「隠れる位置に行く」「攻撃する」という行動を組み合わせることで、あたかも「自分の不利な状態を認識してプレイヤーに対して攻撃態勢を取る」という意味を獲得しました。一方、クロムハウズでは、「通信塔に近付く」「静止する」で「通信塔を取る」というアクション、さらに「通信塔を取り続ける」という、あたかも「通信塔を取ることでゲームで勝利しようとする」という意図を獲得しました。そういった創発をくり返して、極めて抽象的な戦略までを実装して来ました。「人間的な知性」を実現する方法のひとつは、それを直接的に目指すよりは、このように「一つ一つは自明なことだが、それを組み合わせると突然新しい意味を持ち始める」という方法によるかもしれません。これを、学術的には「創発」という言葉で表現します。それは単純なシステムを積み重ねているうちに、不意に自分たちの予想を超えた行動をシステムが産み出し始めるという経験の中から生み出された言葉です。

さて、この創発という考えを頭の隅においておいて、ゲーム開発に立ち返ってみましょう。「単純な行動」を組み合わせで「新しい行動」を生み出すことが出来ないかを考えてみます。それらをつつと作って行きます。そして、それが出来たら、今度は、作った行動から、さらに高度な行動を定義できないか考えてみます。何か出来そうな気がして来たら、そこにプランニングを導入する可能性があります。

クロムハウズの COM には最初、「撃つ」「歩く」「止まる」という3つの行動しかありませんでした。そこから、一つ、一つと高度な行動を定義して行きました。第3章で述べた通り、クロムハウズは、人手による方法でしたので、一つ一つその組み合わせをスクリプトで定義して行きました。それが、1層、2層と積み重なり、4層の構造を持つ AI に発展しました。

プランニングは、高度な思考を実現するために、単純な足場から出発します。ここから、これまで説明して来た以外の例で、そういった単純な層から、高度な階層を実現した例を見て行きましょう。多くの例を知っておけば、きっと自分のゲーム開発の糧になるに違いありません。それは本当に実現したいことを捉えたときに閃きとなって現れるはずで。また、自分自身でもオリジナルなプランニングを使ったゲームの応用

例を考えてみましょう。最初に作った例が、どうかと思うという出来であっても、一度獲得した思考回路は、一つの財産となってゲームを発想するときの自由度の一つとして力となるはずです。

第1節 依存グラフとプランニング

プランニングは第2, 3章で見たようにそれ自身効果的な方法ですが、他の知識と組み合わせて、より効果的なAIを築くことを可能にします。ここでは「依存グラフ」(dependency graph, [38])とプランニングの組み合わせの応用例として説明します。

まず「依存グラフ」というのは、物事の依存関係を表現するグラフ構造です。例えば、風車を作るには、斧と木がいるから、風車は、斧と木に「依存している」と捉えます。また、斧を作るには、鉄が必要であり、斧は鉄に依存している。このような依存関係は、突き詰めて行くと大きなグラフになり、このグラフ全体のことを「依存グラフ」と呼びます。

今、シミュレーションゲームの依存グラフとして、以下のような図を考えてみます。矢印は、その起点となっているものが、矢印の終点になっているものを作るのに必要であることを示します。例えば、弓兵を増産するのに、まず弓錬場が必要であり、弓錬場を作るには、島民と文明が中世まで進んでいることが必須である、と読みます。特に、このようなテクノロジーの依存グラフを技術ツリー (tech tree[39] P.104) と呼ぶこともあります。

さて、この依存グラフをプランニングから捉えると、「弓兵を作る」という行動の前提条件は「弓錬場」があることであり、効果は「弓兵が生成する」ことです。つまり、このグラフによって、AIは、行動を順序立て行うことが出来ます。例えば、最初の状態から弓兵を作るためには、このグラフ構造を追うだけで「まず島民を増やす」「時代を中世まで進める」「弓錬場を作る」「弓兵を作る」というプランを組むことが出来ます。なんだかとてもあたりまえのことですが、第1回のセミナーで説明したように、こういった動作もゲーム世界の知識表現をきっちりと作っておくことで始めて可能になることです。NOLF2 でゴールを実行する順序を決めておく、という方法もこの方法の簡単な応用です。

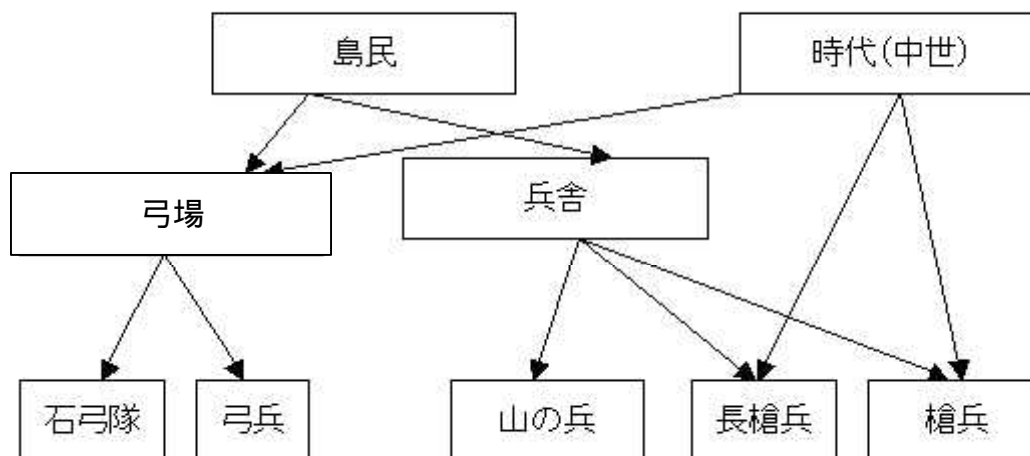


図 4-1 依存グラフの例 ([38])。矢印は、その起点がその指す先の要素を生成するのに必要であることを示す。例えば、弓錬場がなければ弓兵は作れない。島民がいなければ兵舎は作れない。

パターン 1

プランニングが他の技術と組み合わせること効果的な方法になることが多いが、特に依存グラフとの組み合わせは、シミュレーションゲームに有効である。

さて、この依存グラフですが、他に、相手の技術を推測することにも役に立ちます。例えば、弓兵が敵から攻めて来ているとすれば、敵には弓場が既に構築されているということであり、当然、石弓隊が既につくられている可能性があり、また、文明レベルは中世まで進んでいる、と推論することが出来ます。ここから敵のプランを AI が読むことも可能になるわけです。

また「依存グラフ」と「ゴール指向プランニング」という仕組みだけを考えて、他のゲームを考えてみましょう。「深い森の中で、妖精たちが、いろいろな花を掛け合わせて魔法の実のなる花を作る」というゲームを考えてみます。ここで、依存グラフを応用してみます。植物の掛け合わせで何が出来かを予め依存グラフで設定しておきます。今、妖精さん (NPC) たち「夜の真珠」という実のなる花を作るようにゴールを与えます。すると、「夜の真珠」を頂点とする依存グラフから、妖精さんは必要な植物の系譜をたどることが出来ます。この中の幾つかは既に妖精の村に既にあるかもしれません。その植物と、その植物を作るための植物は集める必要がありません。それらの植物以外は、探さなければなりませんので、この依存グラフに従って分担して植物を集めさせることが出来ます。

ここで大切なのは、常にゴールへ向かって自律的に妖精たちを自律的に動かし続けることが出来るということです。初期条件がどのようなものであれ、例えば、ある植物は既にあるとか、途中で枯らしてしまったとしても、その変化を認識して、新しいプランを作ることで対応できます。ゴール指向と依存グラフは、とても相性のよい技術なのです。

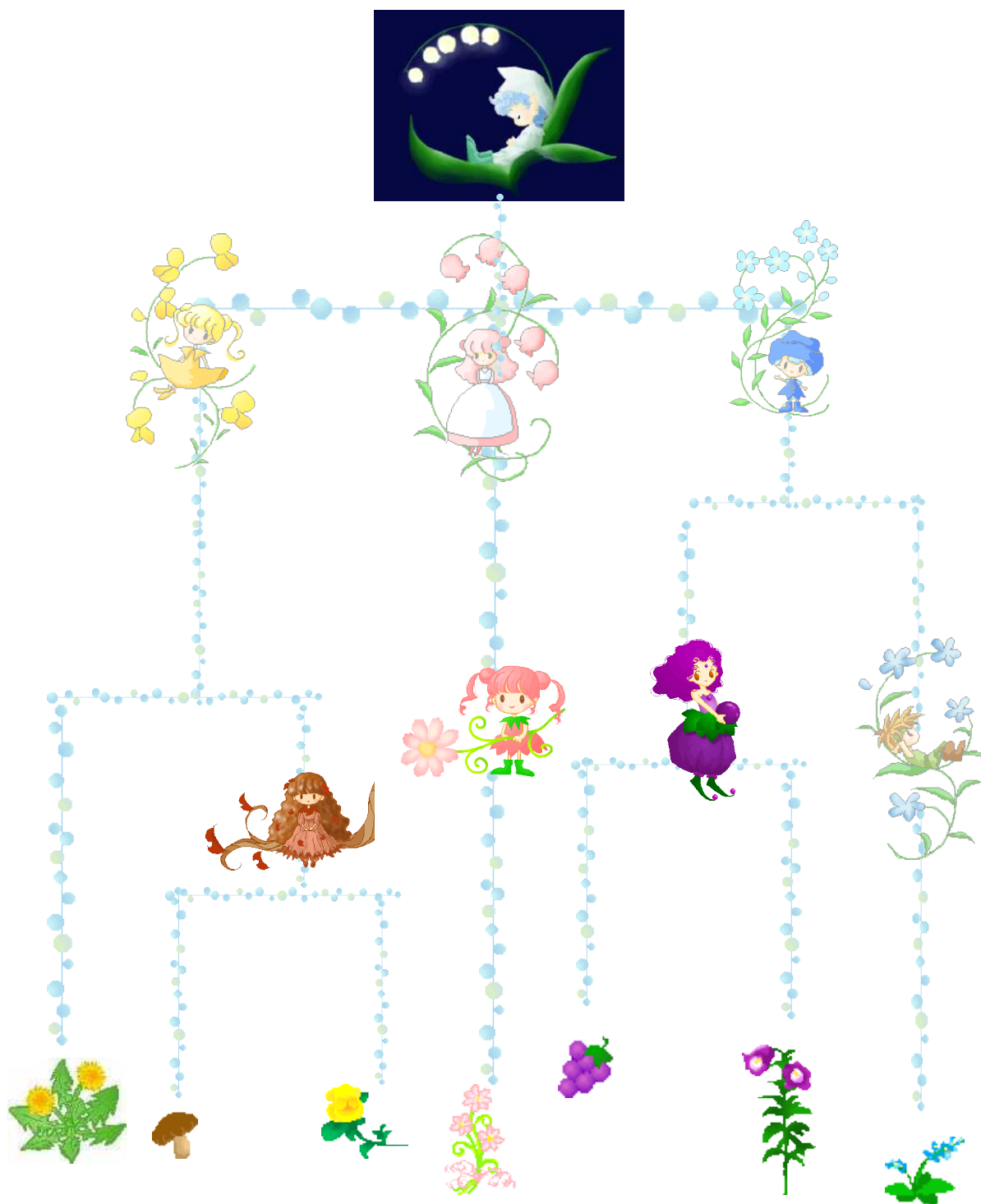


イラスト: アンの小箱 <http://www.anne-box.com/>

図 4-2 依存グラフの応用。これは、ゴール「夜の真珠」を作るために、植物のかけあわせる依存グラフ。妖精たちは、このグラフに従って「夜の真珠」を作るための植物を集める。さらに、プランニングを応用する仕掛けとして「掛け合わせた植物は3日すると枯れる」「植物は根を外されると1日で枯れる」などを入れると、妖精にスケジュールを立てて行動させることが出来る。(イラストは「アンの小箱」のフリー素材を使用させて頂いた。とても柔らかくて暖かいイラストです)

このように、依存グラフという簡単なデータ形式を用意するだけで、NPCを自律的に制御することが可能になります。また、依存グラフは抽象的なデータ構造表現の、ストラテジーゲームからアドベンチャー、RPGなど応用の幅はたいへん広いものがあります。また、敢えて「依存グラフ」と呼ばなくても、プログラマーが自然と使って来た技術であり、そこにゴール指向プランニングの思考を載せることで、さらに応用の可能性を拡げることが出来ます。

第2節 RPGとプランニング

この節では、プランニングとRPGについて考えてみます。この問題については、特に文献があるわけではないですから、私自身のアイデアとして受け止めてください。

連鎖によるプランニングからRPGの戦闘を組み立てることを考えます。それぞれのコマンドに前提条件と効果を記述しておく、プラナーは、例えば「敵の防御系の魔法をまず取り払う」「魔法効果を2倍にする」「魔法を唱える」などと、複数のターンに渡ってプランを組んでくれますので、そのプランに従って戦闘を展開させることが出来ます。プレイヤーがこちらの意図を読んでプランが壊れた場合は、リプランニングを行います。

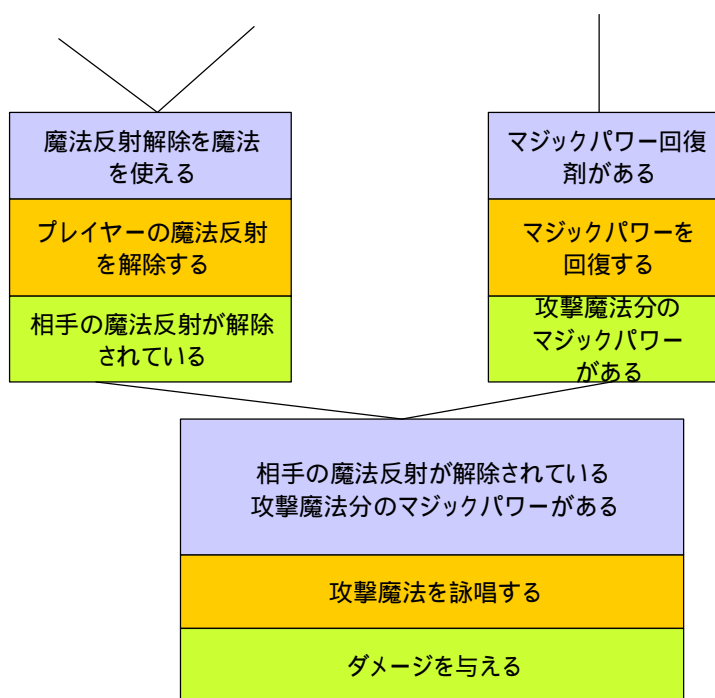


図 4-3 RPGにおける連鎖プランニングの例。階層型プランニングも応用できるはずだ。

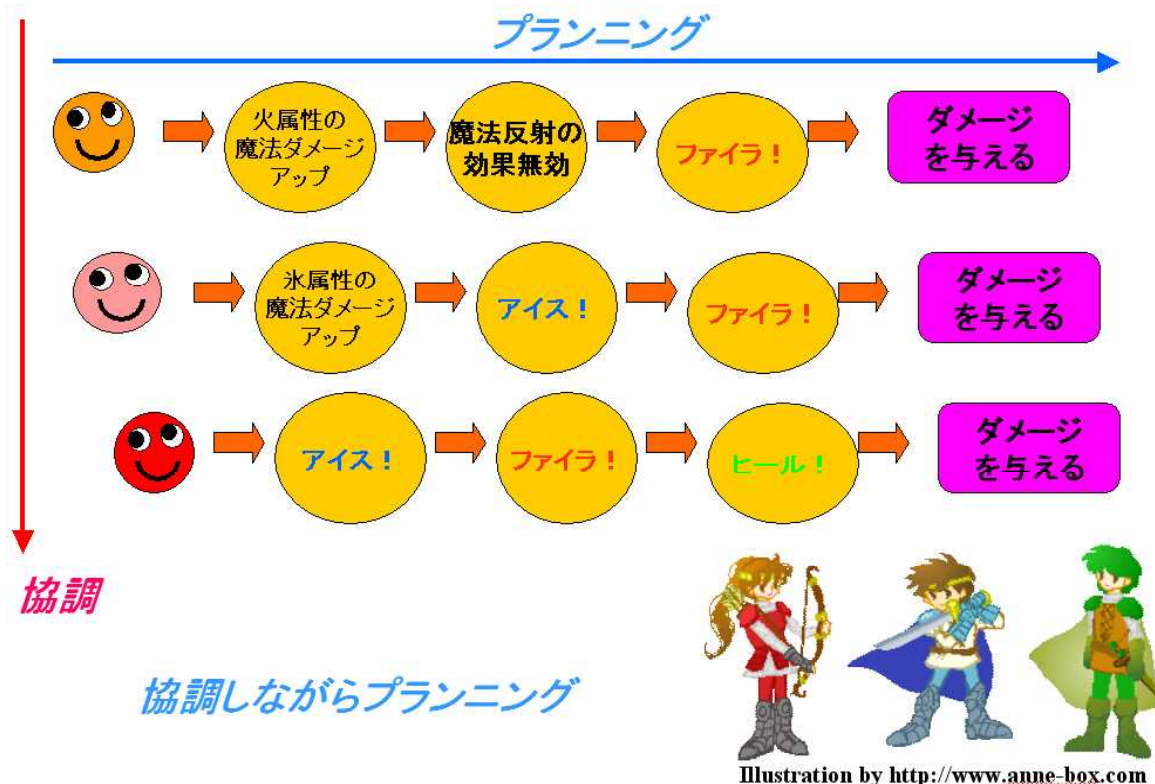


図 4-4 R P Gにおけるプランニングの応用のアイデアの例。プランニングと協調を合わせれば、A Iの敵パーティー、或いは、味方のA Iとしても面白みのあるA Iが作れるはずだ。

魔法に仕掛けや戦闘の仕組みが単純な場合には、プランニングの力を発揮する機会が少ないですが、常に2手、3手先を読んで戦う必要のあるR P Gを作る場合には、A Iにプランニングを応用することで、大きな効果が得られるはずです。また、パーティーを組んで戦う場合には、連携を組んだプランニングを行うことでさらに高度なA Iを築くことが出来る可能性があります。

また連鎖によるプランニングは、行動を必要に応じて追加して行くことが出来るので、プレイヤーのレベルに応じて行動を加えて行くことも可能です。また、行動を加えて行くことでA Iを成長させながら開発することができます。

パターン2

R P Gにおけるプランニングの応用は未開拓であるが、応用が可能であり、また、奥深いシステムを作れば作るほど、その機能が十分に発揮できる。

第3節 ストラテジーゲームとプランニング

戦略ゲームにとって、ゴール指向プランニングはたいへん相性のよい技術です ([39] P.104)。ストラテジーゲームは大きなゴールから小さなゴールまで用意され、小さなミッションをこなして大きなミッションをクリアして行くゲームシステムが準備されている場合が多く、また、プレイヤーの思考もそれをどのようなプランで行って行くか、という問題に集中する比重が高いからです。実際にストラテジーゲームにはゴール

指向プランニングが応用されています。第1節で紹介した依存グラフもその一つの例です。

ここでは、4つの国が覇権を争っているような状況を考えてみます。NPCがプレイヤーの国を征服したいとします。ここでは、クロムハウズのように、人の手によってゴールが呼び出す方式を考えてみます。「相手の国を征服する」には、「相手の兵力を自分の8割以下に削減させる」「相手国の周囲の2国を少なくとも自分の連盟につける」「自分の周囲の国を少なくとも1国味方しておく」という3つのゴールが必要とします。また、「相手国を自分と同盟を組む」ためには、「政略結婚」「贈り物をする」「威嚇して交渉」の3つがあるとします。また、「相手の兵力を削減する」には「相手国の交易を奪う」「傭兵を相手国から買う」というゴールが使用可能であるとします。すると、COMは、このゴール図に従って行動することでチュウ支障的な戦略を達成することが出来ます。

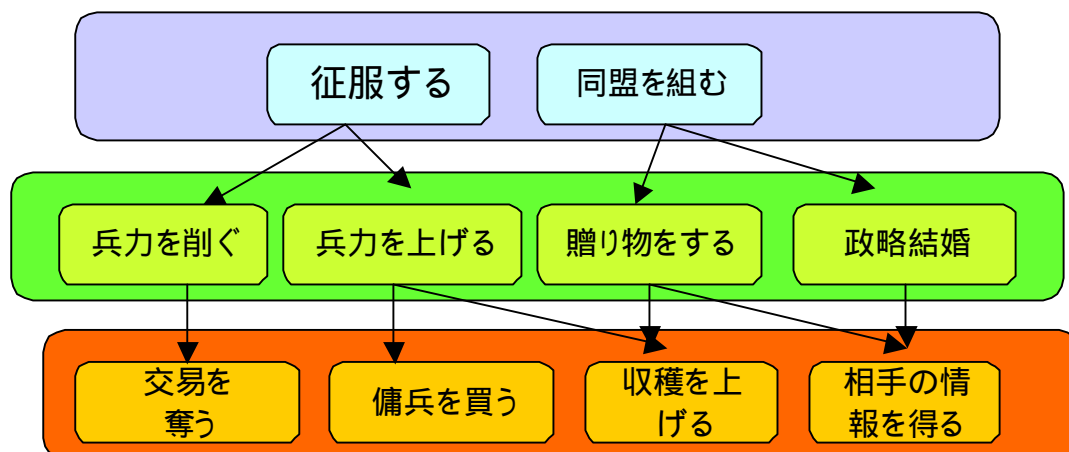


図 4-5 戦略ゲームにおける階層型プランニングの例

或いは、サッカーゲームのような戦略性のあるスポーツゲームにおいても、戦略からアクションを導くシステムを作ることによって応用できる可能性があります。

パターン3

ストラテジーゲームとゴール指向プランニングは非常に相性がよい。

第4節 集団の制御とゴール指向プランニング

チームAIとは、個々のNPC毎にある知能ではなく、全体を制御するための思考のことを言います。チームAIにおけるプランニングとは即ち、計画的に集団を制御する知能のことを言います。

さて、チームAIとプランニングを考えるために、ある集団は、まず司令官がいて、その下に大隊があり、大隊は小隊からなり、中隊は兵士からなる階層構造を準備します。このような階層構造を集団に持ち込んだ場合、集団を制御するために幾つかのメリットがあります。

まず一つ目は、各階層ごとに独立に思考を作ることができる、ということです。階層の間には、命令と報告しかありません。例えば、司令官は大隊に命令を出し、報告を受け取るので、それ以下の層とは関係を持ちません。大隊は、小隊に命令し、報告を受け、その報告をまとめて司令官に報告します。これもまた、それ以下の層とは関係しません。つまり、インプットとアウトプットだけを合わせておけば、後は独立にカスタマイズが可能です。大隊の思考が気に入らなければ、そこだけ独立に改善することができ、また、ゲームの設定が少し変わっても、司令官の思考を変えれば、全体の動きを変えることが出来ます。また、MODを作るのにも適しています。或いは、オンライン上で、司令官、大隊、小隊ごとに思考スクリプトを公開して、ユーザーがそれらをダウンロードしあってだんだんとAIを発展させて行く、という仕掛けを作ること出来ます。場合によってはコミカルな軍隊も構成されるでしょう。ユーザーの作った軍隊をオンライン上で対戦させるという仕掛けが出来れば、ゲームとしての深みをさらに持たせることが出来るでしょう。

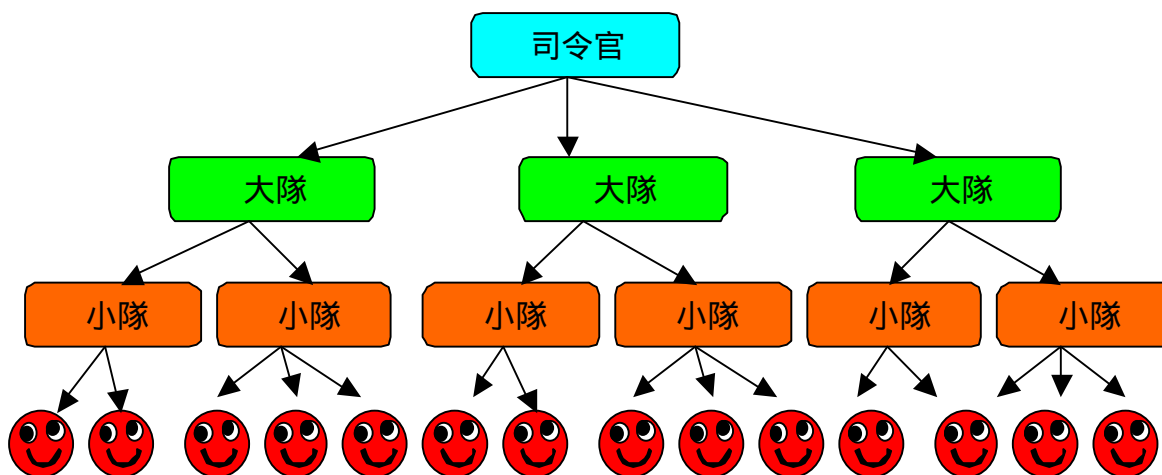


図 4-6 チームAIにおける階層型プランニングの応用例

二つ目のメリットは横の関係もあり気にしないでもいいということです。例えば、小隊同士の関係は、その上の大隊が把握しているので、小隊同士で連絡しあう必要はありません。例えば、小隊のNPCがプレイヤーに倒されて、人数が少なくなった場合には、同じ大隊に属する小隊からメンバーを融通してもらうべきですが、これも大隊がコントロールします。また大隊同士の関係は司令官がコントロールすればよいです ([40])。

さて、このような構成において、司令官はゴール指向を大隊を動かすために用います。ある目的、ここでは「敵基地を叩く」ために「ある大隊は敵陣へ直進して攻撃」「ある大隊は遠くの高台から援護射撃」「ある大隊は、その中間地点で待機」というより小さなゴールへ分解します。この3つのゴールは、それぞれの大隊へ役割分隊されます。「ある大隊は敵陣へ直進して攻撃」というゴールを受け取った大隊は、そのゴールを達成するために、さらに小さいゴールへ分けて、各小隊へゴールを割り振ります。そして、最後には、末端の兵士へ命令がくだって、全体としての行動が決定されます。

ここまでは「ゴール指向」の考えだけで十分でしたが、例えば、プランニングの要素を入れて、「大隊Aが敵陣を攻撃」「敵陣から煙が見えたら大隊Bも基地の背面から攻撃」「5分たって敵基地に白旗がたたなかつたら、大隊Cは高台から遠距離射撃」というプランを立てることが出来ます。そして、時間軸にそって、3つの分隊が役割をこなします。

このように、集団の制御においても、プランニングは大きな力を発揮します。複数のエージェントをプランニングによって行動させる技術を「マルチエージェント・プランニング」と言います。その場合には、各エージェントの行動の同期（タイミング）が問題となりますが、この問題を「マルチエージェント・プランニングにおける同期の問題」と言います。この問題については、第3回のセミナーで取り上げます。

パターン4

複数のエージェントを制御する場合に、階層構造とプランニングを組み合わせると全体として統制された行動を取らせることができる。

第5節 多様な行動

これまでは専ら争いにおける知能を説明して来ましたが、プランニングはより多様なNPCの行動を生み出すことを可能にします。例えば、以下のようなオンラインゲームを考えてみましょう。

まず、4人のプレイヤーが共有できる大きな街があります。そこでプレイヤーたちは探偵となって犯人を追います。犯人はプランニング思考を持たせたCOMであり、毎回、ゴールとして「要人の家にある機密書類」や「博物館にある宝石を盗む」が設定され、プランニングによって自分の行動を決定します。例えば、「博物館にある宝石を盗む」ためには、「隣の屋敷から屋根伝いに博物館の屋根へ飛び移る」必要があり、「その屋敷へ出入りするためには、門の合鍵を作る必要があり」「門の合鍵を作るためには、一度、屋敷へ修理工のふりをして忍び込む必要があり」「そのためには…」と、プランニングをする能力を与えておきます。

一方、プレイヤーは探偵として、犯人のゴール（目的）をそれまでに行われた事象の中から推測して、街の大勢のNPCから一人の犯人を見つけなければなりません。探偵側はいくつかの妨害ができますが、これに対してCOMにはリプランニングの能力を持たせることで、毎回、プレイヤーの行動に応じて計画の道筋を変えることが出来ます。このようにゴール指向プランニングを使って、デジタル空間の中で、人間とAIの頭脳戦をくり広げることが可能です。

上記の例のように、戦闘型のAIだけでなく、街や経済市場、株などシミュレーション型のゲームにおいてもプランニングを応用して多様なNPCを生み出すことが出来ます。そして、NPCの可能性が伸びた分、新しいゲームデザインも自然と広がっているはずであり、そこに新しいゲームを見出す企画力があれば、新しいゲームの可能性を見出すことができると考えます。

ゲームをデザインする場合に、プランニングの技術を知っておくと、発想できる幅が広がります。ゲーム製作において自分の経験が作り出した「閉じたループ」から抜け出すためにも必要なことであると思います。

パターン5

プランニングは、戦闘にのみならず、多様なAIの行動を構築できる可能性を持ち、それは新しいゲームデザインにつながっている。

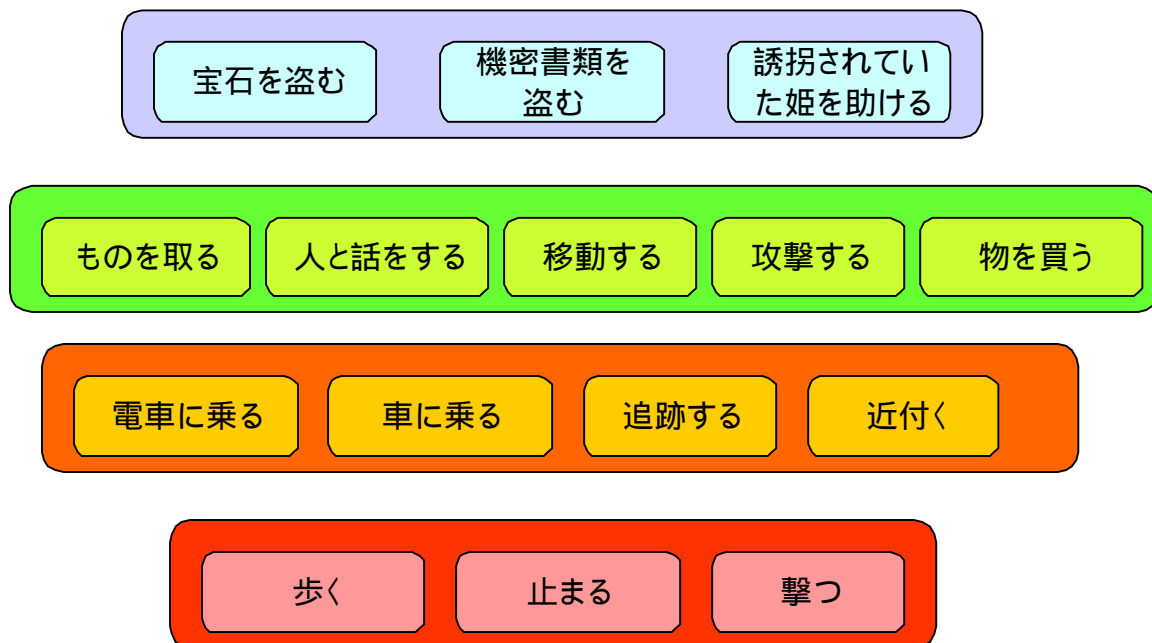


図 4-7 プランニングを用いたユニークなAIの例

第6節 AIに時間を与える

実際の解説としてはこの節が最後になりますので、冒頭で述べた「AIに時間を与える」がどういうことだったかを述べつつ、このテキスト全体を反省したいと思います。技術的解説ではなく抽象的な内容なので、飛ばして読まれても問題ありません。読んで頂ける方は、少しだけ、自分がAIになったつもりで読んでみてください。

まず、現在という時間に自分の持つ感覚によって結びついています。瞬時瞬時の条件から行動が決定されるなら、AIに存在するのは浮かんでは消えて行く「現在」という各瞬間であって、未来や過去といった連続的な時間のつながりが大切になるわけではありません。時間という観念すら持つことが出来ないかもしれません。

では、「過去」や「未来」を持つということはどういうことかという、過去とはまず「記憶」を持つということです。自分や世界がどう変化して、何を知って来たかを蓄積するのが記憶です。生き物にとって過去は、実体のあるものではありませんが、折り畳まれた記憶となって存在しています。逆に、未来という時間は、自分の行動を展開できる時間がこの先に広がっている、という感覚のもとに現れる概念です。自分の行動を未来の時間にシミュレートできる能力があって初めて獲得することが出来ます。この「未来をシミュレ

ートする」中でも最も単純な知性の能力がプランニング技術と言えます。また、未来と過去は可分ではなく、過去は記憶を通して行動のプランに影響を及ぼします。F.E.A.R やクロムハウنزのエージェント構造図において情報の流れを見て頂ければ、知能が巧みに過去を記憶にかえ、その記憶と「現在」の感覚から、未来の行動をデザインしている過程を理解して頂けると思います。

さて、今回紹介した内容の中で唯一触れていない内容があります。それは「予測」([17])です。第1回のテキストで述べた原理、

第1回 パターン3

C O MのA Iシステムを、プレイヤーに適用することでプレイヤーの動きを予測する。

を思い出してみます。Killzone では、A Iの思考ルーチンをプレイヤーに適用することで、行動の予測を行いました。プランニングでもそれが出来る可能性はあります。しかし、A Iを動かすためのプランニング以上に、人間的なプランニングまでが行える精度を持つシステムを作らなければ、予測はかえって逆効果になり兼ねません。この方向については、まだ応用の例はありません。今のところ、クロムハウنزのプランニング・システムでは、プレイヤーを予測するのに十分ではありません。

高度な生き物になればなるほど、自分だけでなく、相手の行動も予測してシミュレートします。或いは、相手と自分を合わせてシミュレーションして行動を決定します。この感覚はクロムハウنزのように、プレイヤーを相手にA Iを戦わせると如実に感じる事が出来ます。ハウنزハウズで、自分にとって最も賢明な行動を選択します。クロムハウズでは、それはゴールでした。そして、そのゴールに沿って行動をデザインします。人間は、A Iと自分の両者の動きを完全に把握した上で、A Iの動きの予想を踏まえて自分の行動を決定します。つまり、人間は自分の行動だけでなく、自分の視野に入っているもの全ての変化をシミュレートして、自分をその世界の一部としてプランニングするのです。これによって、大きな状況の変化の中でも、それに対応した行動を取ることが出来ます。

A Iが人間の知性に追いつくのは遠い未来になることでしょう。プランニングは、人間の行動決定の一端を、シンプルな形で模したものに過ぎませんが、それでも、たどたどしく、人間が持つ知性の領野に半歩でも足を踏み入れつつあります。人間は自分の知性の領域に入ってくるものに対しては非常に敏感です。200年前にオートメーションという名のもとに機械が仕事場に入ってきたとき人間が何を行ったか？ そしてコンピューターが現れたとき、そして、今現れつつある、人の生活に入り込もうとするロボット、意識的にしろ、無意識にしろ、人間は、その存在に常に好機と警戒のアンテナを張り巡らしています。逆に言えば、ゲームA Iにおいて、人間の知性の一端に触れるほどのA Iを作ることが出来れば、ユーザーに新しい感覚を喚起させることが可能です。ゲーム空間の中にユーザーが自分と似た知性を発見したときの驚きは、どのようなものでしょうか？ それは、C Gにおける現実と見間違ふほどのリアルなグラフィックを始めて見たときの衝撃に比肩し得るものでしょう。そして、その驚きこそが、ゲームA Iを発展させる原動力であると私は疑いません。

第7節 まとめ

この章では、ゴール指向、及びプランニングそれぞれの応用の幾つかの方向を紹介しました。プランニング技術は応用の幅の広い技術であり、どのような準備の上に使用するかによって様々な応用の可能性が開けています。

第1節では、Age of EmpireのようなRTS（リアルタイムストラテジー）において、依存グラフを組み合わせた応用の仕方を紹介しました。依存グラフそのものが、AIに行動の順序を知らせてくれるものとして機能することを一つのゲームアイデアを用いて説明しました。

第2節では、日本で大きな発展を遂げたRPGにおける応用のアイデアを提示しました。RPGにおけるプランニングの応用は、私の知る限り未開拓の分野であり、この分野においては日本の開発者がフロンティアを拓いて行けると期待しています。驚くようなRPGの敵キャラに出会えることを楽しみにしています。

第3節では、ストラテジーゲームにおいては、ゴール指向プランニングは非常に相性のよい技術であると指摘しました。実際、意識的にしろ、無意識的にしろ、多くのストラテジーゲームのAIは、ゴール指向的な仕組みを持っています。それを「ゴール指向プランニング」と捉えることで、様々な応用技術を取り入れる契機にして頂けたらと思います。

第4節では、集団の制御にゴール指向を使う例を紹介しました。この方法はよく使われる方法です。クロムハウズは6体のAIなので、2層ですが、この形式のチーム制御が取られています。また、ゴールという名前だけでなく、タスクという名前で階層型タスクネットワーク(Hierarchical Task-Network、HTN)が解説されていることがあります〔41〕。集団の制御については第3回のセミナーで説明します。

第5節では、日常生活やアドベンチャーゲームにおける応用の例を説明したかったので、簡単なゲームアイデアを例に説明しました。戦闘という目的がなくても、我々が日常的にプランニングを行って生活するように、AIにゴール指向プランニングを使って日常的な行動を取らせることが出来ます。逆に、ゴール指向プランニングから新しいゲームのスタイルが産まれることを期待しています。

この章は「ゲームを発想する」ということに重点を置いて説明して来ましたが、さまざまな応用の方向があることを提示するために、一節一節方向を変えて例を出しました。企画にとっても技術者にとっても、技術を身につけるということは自由を獲得することと同義です。これまで実現できなかったことが出来るようになり、また、それゆえに新しい落とし穴に陥ることもあります。この章で説明した例は、ほんの一例に過ぎません。ゴール指向プランニングからどんなゲームの応用が拓けるかは、開発者自身の肩にかかっています。そこでは企画の自由な発想と技術者の堅実な技術的基礎が必要とされます。

ゲームは技術だけから作られるものではありませんので、ここで解説した技術は、次に開発するゲームで応用できるかもしれませんし、また、ずっと先に開発するゲームで応用できるかもしれません。そのチャンスが来た時につかむことが出来るために、「とりあえずゲームAIをゴール指向プランニングで考えてみる」という習慣を身につけて頂ければと思います。また、過去に開発したゲーム、過去に遊んだゲームでこの技術を応用すれば面白いことが出来たはずだという例、今回のように実際に応用された実例を集めておかれると、よりよい開発の土壌になるのではないかと思います。

第8節 展望

クロムハウズでゴール指向プランニングを応用できることに気付いて、開発チームと一緒にA Iのシステムを作って行きました。ゴールを追加するたびに新しいことが出来るようになる、そんな生き物を育てるような感覚が楽しくて、社内サイトに開発日誌（成長日誌？）を毎日A Iの画像入りで公開したところ、案外これが受けがよく、C E D E Cで発表することになりました。いつか、ゴール指向プランニングの開発の面白さを他の開発者に伝えることが出来れば、と思っていたところ、このテキストを書ける機会も頂いて筆を運んでいます。ゴール指向プランニング以外にも、ゲームにとって人工知能には魅力的な分野が数多くあり、セミナーを通じて、その面白さの一端でもなんとか伝えることができればと思っています。

A Iは世話のかかる存在です。まずゲーム世界が何たるかを教えてやり（知識表現）そして、行動の仕方を用意してやったりしなければなりません。ただ、A Iを使わない制御とA Iを使う制御の違いは、前者が、A Iの具体的な行動を全て指定して行くのに対して、後者は、A Iを動かす装置自体を作ってチューニングして行くということです。その装置は、時間と情報という流れが注ぎ込んで動作します。逆に言えば、その流れをうまく（知的）運動に繋げるべく設計する必要があります。そして、その装置が、A Iの行動を創造してくれます。その仕組みがプランニングです。そして、多くの人工知能技術は、この装置のような性質を持っています。時々人工知能を作っていると「操り人形」を操る「操り人形」を作っている気になります。

この数年は、開発会社、メディア共に、特にコンピューター・グラフィックスにユーザーの目をひきつけることによって業界を牽引して来た感があります。人工知能はコンピューター・グラフィックスに劣らず魅力的な分野であり、今後ユーザーに人工知能がゲームにもたらす面白さを伝えて行くことが、ゲームが未来につながる一つの軸であると考えます。特に日本にはゲーム会社とメディア、ユーザーが一体となった「ゲーム文化」があり、ゲームにおける人工知能をそこでみんなで育てて行くことができれば、こんな楽しいことはないと思います。開発者自身にとっても、これまで歩んで来た以上に、広く深い荒野が目前に広がっていることは好機であると思います。ただ、人工知能はC Gのように、一見してその成果が見えるものでもないため、これから如何にしてユーザーに各社のA Iの面白さを見せて行くか、どう伝えて行くか、ということが課題として待っています。また、開発者にとっても、人工知能は決して見通しのよいとは言えませんので、ゲームA I分野の情報の整理と発信、研究の発展が望まれています。このセミナーは、その出発点の一つとなることを目標にしています。

2007年4月

yoichi-m@pk9.so-net.ne.jp

Appendix Mat Buckland によるゴール指向プランニングのコーディング例

```
//----- Initialize -----  
//-----  
void Goal_HuntTarget::Activate()  
{  
    m_iStatus = active;  
  
    //if this goal is reactivated then there may be some existing subgoals that  
    //must be removed  
    RemoveAllSubgoals();  
  
    //it is possible for the target to die whilst this goal is active so we  
    //must test to make sure the bot always has an active target  
    if (m_pOwner->GetTargetSys()->isTargetPresent())  
    {  
        //grab a local copy of the last recorded position (LRP) of the target  
        const Vector2D lrp = m_pOwner->GetTargetSys()->GetLastRecordedPosition();  
  
        //if the bot has reached the LRP and it still hasn't found the target  
        //it starts to search by using the explore goal to move to random  
        //map locations  
        if (lrp.isZero() || m_pOwner->isAtPosition(lrp))  
        {  
            AddSubgoal(new Goal_Explore(m_pOwner));  
        }  
  
        //else move to the LRP  
        else  
        {  
            AddSubgoal(new Goal_MoveToPosition(m_pOwner, lrp));  
        }  
    }  
  
    //if there is no active target then this goal can be removed from the queue  
    else  
    {  
        m_iStatus = completed;  
    }  
}  
  
//----- Process -----  
//-----  
int Goal_HuntTarget::Process()  
{  
    //if status is inactive, call Activate()  
    ActivateIfInactive();  
  
    m_iStatus = ProcessSubgoals();  
  
    //if target is in view this goal is satisfied  
    if (m_pOwner->GetTargetSys()->isTargetWithinFOV())  
    {  
        m_iStatus = completed;  
    }  
  
    return m_iStatus;  
}
```

図 0-1 Mat Buckland ,2005,Programming Game AI By Example、WORDWARE Publishing より、ゴールクラスの実装例（テキスト[10]、全体のコード [32]）。ここでは、敵を追う「HuntTarget」の Activate 関数と Process 関数の一部を取り出した。

ここでは、第3章で紹介した階層型プランニングのコーディング例を

Mat Buckland, 2005, "Programming Game AI By Example", WORDWARE Publishing

の第9章「Goal-Driven Agent Behavior」を基に解説します。特に、そのデモのソースコード ([32]よりダウンロード可能) における「Hunt Target」の *Activate* 関数と *Process* 関数の中で、どのようにゴールが登録され、実行されるかを説明します。

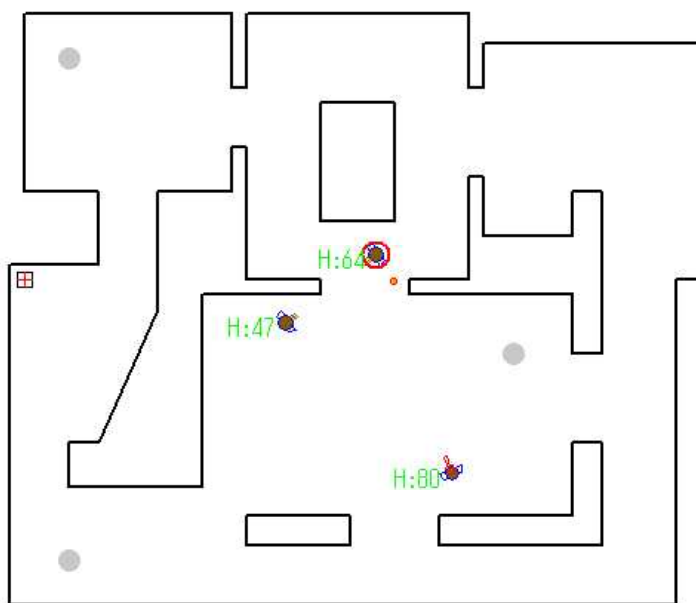


図 0-2 Mat Buckland, chapter 7-9, "Programming Game AI By Example", WORDWARE Publishing ([10]) における「Raven」のデモ ([32])。A I がゴール指向プランニングによって思考して行動を決定し、パス検索を行って移動する。説明する「HuntTarget」ゴールは、このA I の複合ゴールの一つである。このデモはゴール指向を解説するときに重宝する。コンパイルすれば実行可能である。また、ソースコードから多くのことを学ぶことができます。

まず、ゴールのクラスにはC O Mの制御の仕方が記述されています。下位のゴールクラスのインスタンスは、「クラス = (単純な) 命令」と捉えることができますが、ここで紹介する、敵を追う「HuntTarget」はそれ自身が、下位のゴールを呼び出す「複合ゴール」(Composite Goal)です。そこで、このクラスが、どのようにクラスを制御しているか見てみましょう。

クリア条件

このゴールは「敵を追いかける」というゴールです。クリア条件は「敵を視野(FOV、Field of View)の中に入れる」ことです。敵を見失ったとき、或いはターゲットを決めて攻撃を開始する場合に呼び出しておくと、ターゲットを視野に捉えてくれる便利なゴールです。Process 関数の最後の部分はクリア条件を判定しています。ゴールの作り方にもいろいろなコツがあって、うまいゴールを作ると、シンプルな構造でたくさ

んのが出来るようになります。このデモから学んで頂ければと思います。

■ *Activate*

「*HuntTarget*」ゴール自身も、さらに上位のクラスから呼び出されています。最終的には思考クラスが全ての階層ゴールの始点になります。「*HuntTarget*」ゴールが呼び出されると、まず、その *Activate* 関数が実行されます。

ゴールクラスには4つの状態が規定されていて、この関数に入ると *activate* であることが宣言されます。

m_iStatus

✧	<i>inactive</i>	待機してます
✧	<i>active</i>	実行している
✧	<i>completed</i>	完遂した
✧	<i>failed</i>	失敗した

次に注目するのが、*AddSubgoal* です。これは、サブゴールのインスタンスをスタックして行く関数です。まず、このクラスのゴールスタックを初期化 (*RemoveAllSubgoals*) した後、条件式のところは大雑把に書くと、

標的となる敵がまだ生きているなら

- ✓ 敵が見つかった場合、敵のいる位置へ移動する「*MovePosition*」ゴールをスタックに積む。
- ✓ 敵が見つからなかった場合、探索する「*Explore*」ゴールをスタックに積む。

標的となる敵は既にやられていた

- ✓ このゴールは完遂しました

という処理になっています。ここではゴールが一つしかスタックされませんでしたが、ゴールによっては、複数のゴールが同様に条件分岐のもとで、スタックされて行きます。

Process

この関数はAIのアップデートのたびに実行される関数です。*Activate* で作成したゴールスタックのゴールを順番に実行します。その関数が、*ProcessSubgoals* です。その下には、このゴールのクリア条件の判定式が記述されています。このゴールのクリア条件は、「敵を決められた範囲の視野に入れるまで近付く」ことでしたので、

- ✓ クリア条件が満たされなければ、完遂。
- ✓ クリア条件が満たされなければ、*active* のままでアップデートのタイミングでプロセスがもう一度実行される

となります。

このように、一つのクラスのインスタンスは、別のゴールクラスのインスタンスを呼び出し、また、そのインスタンスは別のゴールクラスのインスタンスを呼び出す入れ子構造(composite design pattern)になっています。これが、階層型プランニングを支える仕組みになります。

[Appendix おわり]

参考文献

URLをクリックすると、参照先が呼び出されます。WEBにおける文書は、リンク切れになる場合が多いので、必要なものはすぐに Download しておくことをお勧めします。また、リンク切れになった場合にも、著者名やタイトルで検索すれば見つかる場合が多いです。

[1] スチュワート ラッセル ,ピーター ノーヴィグ, エージェントアプローチ 人工知能, 共立出版, (1995)

【概要】人工知能研究について知りたければ、全ての分野を網羅している入門書。大学の学部の教科書として使われることが多いが、人工知能に関わる全ての人に重宝されている。ただ、英語版は第2版だが、これは初版の翻訳である。

[2] 三宅 陽一郎, IGDA日本 : ゲームAI連続セミナー「ゲームAIを読み解く」第1回「KillzoneにおけるNPCの動的な制御方法」資料, <http://www.igda.jp/modules/news/article.php?storyid=1050>

【概要】「知識表現」「世界表現」をゲームの視点から見た内容になっている。基礎的な内容だが単純でも簡単というわけでもない、今後のセミナーの基本になるので、読んでおいてもらいたい。

[3] SIERRA, F.E.A.R, <http://whatisfear.com>, (2004)

[4] AIWisdom, <http://www.aiwisdom.com/>

[5] Craig Reynolds, Game Research and Technology, <http://www.red3d.com/cwr/games/>

【概要】Craig Reynoldsのサイト内でゲーム関連の情報を集めたページ。テクノロジー毎に分類されており、開発者必見のページと言えるだろう。

[6] CGF - AIのリンクサイト, <http://www.cgf-ai.com/links.html>

【概要】ゲームAIのリンク集のページ。優れたページを集めている。リンク切れが多いが、タイトルか著者名で検索すれば見つかる場合が多い、

[7] Amit Patel, Amit's Game Programming Information,
<http://www-cs-students.stanford.edu/~amitp/gameprog.html>

【概要】ゲーム開発技術全般についてのリンク集で、ゲームテクノロジー、ゲームジャンル、ゲームデザインによって分類されている。Programmer 向けのサイトである。

[8] 徳田英幸, 村井純, 自律分散協調論 - 夢から現実への挑戦,
<http://www.soi.wide.ad.jp/class/99002/index.cgi>

【概要】大学の講義録で、解説に用いられた全12回のPPTが公開されている。エージェント、自律分散協調、そしてインターネットまで基本から応用までを概観することが出来る。

[9] コロナ社, (1996)

[10] Mat Buckland, Programming Game AI By Example, WORDWARE Publishing, (2005)

【概要】「スクリプトによる制御」「FSM」「振る舞い」「ゴール指向」など「ゲームAI」の基本事項をわかりやすく説明した書籍。各章にはUMLによるプログラムのデザインと、ソース、デモが用意されており、「ゲームAI」を解説した書籍の中では群を抜く最高の書籍である。

[11] Jeff Orkin, Jeff Orkin のサイト(文献多数), <http://web.media.mit.edu/~jorkin>

【概要】F.E.A.Rの論文の殆どは全てここから引用することが出来る。

[12] Jeff Orkin, Three States and a Plan: The A.I. of F.E.A.R, http://www.jorkin.com/gdc2006_orkin_jeff_fear.doc

【概要】 F.E.A.R のプランニングのフレームを説明した文書。開発者の視点で書かれており、技術的内容は、ピザの宅配や、ナッツや、ケーキなど、いろいろな比喻を使って理解しやすいように説明している、第2回のセミナーの解説は、この文書に従っているが、詳細は他の論文を調べる必要がある。

[13] Jeff Orkin, Applying Goal-Oriented Action Planning to Games, AI Game Programming Wisdom 2, Charles River Media., 217-228, (2003)

【概要】F.E.A.Rがリリースされる前の文献なので、F.E.A.R というタイトルは書かれていないが「連鎖によるプランニング」の中心的アルゴリズムを扱ったもの。手早く「連鎖によるプランニング」を理解したい人はこの記事を読むとよい。

[14] Jeff Orkin, Agent Architecture Considerations for Real-Time Planning in Games (PPT), http://www.jorkin.com/AIIDE05_Orkin_Planning.ppt

[15] Jeff Orkin, Constraining Autonomous Character Behavior with Human Concepts (PPT), <http://www.cse.lehigh.edu/~munoz/CSE497/classes/JonMartinpresentation3.1.ppt>

[16] Jeff Orkin, Applying Blackboard Systems to FPS, <http://www.jorkin.com/utgameAI03-Orkin.ppt>

【概要】NOLF2を題材に「黒板モデル」の導入を解説している。「黒板モデル」が如何にAIのアーキテクチャーをエレガントにしてくれるかを解説したPPT。

[17] D. Isla, R. Burke, M. Downie, B. Blumberg, A Layered Brain Architecture for Synthetic Creatures, <http://characters.media.mit.edu/Papers/tomlinsonAAAI2001FS.pdf>

【概要】 F.E.A.R のフレームは、この論文を参考に行っている。他にもゲームAI開発者にとっては必要な多くの情報があり、必読するべき論文の一つだろう。

[18] Jeff Orkin, Constraining Autonomous Character Behavior with Human Concepts, AI Game Programming Wisdom 2,3.1

【概要】NOLF2を題材に「人間的な概念」をゲームの知識表現に導入しようとする意欲的な記事だ。

[19] Jeff Orkin, Toward Real-Time Planning in Games, http://www.jorkin.com/AAAI04_Orkin_Planning.ppt

【概要】F.E.A.R. のプランニングの部分に特化してわかりやすく説明している。

[20] 石田亨、片桐恭弘、桑原和宏、分散人工知能、コロナ社

【概要】「マルチエージェント・プランニング」「黑板モデル」「市場モデル」などエージェントの協調、競争を扱っており、説明も充実している名著。是非、枕元に置いておきたい一冊だ。

[21] IGDA AIISC, <http://www.igda.org/ai/>

【概要】「The 2005 AIISC Report」の日本語訳がIGDA日本のサイトにあります。

[22] Jeff Orkin, Symbolic Representation of Game World State: Toward Real-Time Planning in Games,,
<http://www.jorkin.com/WS404OrkinJ.pdf>

【概要】Jeff Orkinの「ゲームにおけるシンボル形式によるゴール指向プランニング」という基本コンセプトが説明された論文。

[23] Jeff Orkin, Agent Architecture Considerations for Real-Time Planning in Games,AIIDE Proceedings,
<http://www.jorkin.com/aiide05OrkinJ.pdf>

【概要】とても学術的な論文だが、F.E.A.R. AIの実装の詳細についてデータ構造まで含めて記述されている。

[24] Jeff Orkin, Three States and a Plan: The A.I. Of F.E.A.R. (GDC2006) (Document,PPT,Movie),
http://www.jorkin.com/gdc2006_orkin_jeff_fear.zip

[25] 三宅 陽一郎, クロムハウズにおける人工知能開発から見るゲームAIの展望(CEDEC2006),
<http://www.igda.jp/modules/eguide/event.php?eid=41> [補足資料]からリンク

【概要】第3回のセミナー「Chome Hounds におけるチームAI」の予告ページ。「補足資料」からリンクがある。また、ここには「集団としての知能」の入門的な情報をまとめている。

[26] 三宅 陽一郎, 人工知能が拓くオンラインゲームの可能性 (AOGC2007),
<http://www.bba.or.jp/AOGC2007/2007/03/download.html>

[27] クロムハウズ公式サイト, <http://www.chromehounds.com/>

[28] クロムハウズ Xbox サイト, <http://www.xbox.com/ja-JP/games/c/chromehounds/>

[29] Neil Wallace, Hierarchical Planning in Dynamic Worlds, AI Game Programming Wisdom 2, 3.5, (2003)

[30] Tom Kent, Multi-tiered AI Layers and Terrain Analysis for RTS games, AI Game Programming Wisdom 2, 7.8, (2003)

[31] J.M.P. van Waveren, The Quake III Arena Bot,

http://www.kbs.twi.tudelft.nl/docs/MSc/2001/Waveren_Jean-Paul_van/thesis.pdf

【概要】Quake III Arena の Bot を扱った100ページを超える、オランダの「Delft University of Technology」の「Knowledge Based Systems Group」のJ.M.P. van Waveren さんの修士論文(2001)。ゲームAIの様々な技術をQuake III Arena を題材に概要を説明している。

[32] Mat Buckland, 「Programming Game AI by Example」サンプルコード, <http://www.wordware.com/files/ai/>

【概要】「Programming Game AI by Example」のサンプルコードをDownloadできる。特に第7～9章のサンプルコードは、AIの戦車がゴール指向プランニングによって対戦するデモとなっている。

[33] Mat Buckland, Mat Bucklandが主催するAIのサイト「ai-junkie」, <http://www.ai-junkie.com/ai-junkie.html>

【概要】時々、AIのコンテストとかも開催されています。人工知能技術の解説も充実。面白いので遊びに行ってみてください。

[34] エリック ガンマ、ラルフ ジョンソン、リチャード ヘルム、ジョン プリシディース, オブジェクト指向における再利用のためのデザインパターン, ソフトバンクパブリッシング, (1999)

[35] Marvin Minsky, The Society of Mind (「心の社会」という邦訳あります), Touchstone Books, (1985)

【概要】「我々の心の中もたくさんのエージェントが争いあっている社会なのだよ。だから、心はいろいろな機能を持てるのだよ」という考えに基づいた、人工知能の研究をリードして来たミンスキーの著作。一般向け。「心の社会」(産業図書)という邦訳もあります。

[36] ユクスキュル、クリサート, 生物から見た世界, 岩波文庫,

【概要】生物の知能は、感覚から得た情報から、再構築することで、知的活動の土壌となる空間を得ている。当然、それは生物の種類によって違うはずだ。この考えを環世界という。環世界という考えの発端を作った著者による、生物が再構築している世界の説明。世界的名著。

[37] Brooks, R. A Robust Layered Control System for a Mobile Robot,

<http://people.csail.mit.edu/brooks/papers/AIM-864.pdf>, (1985)

[38] Paul Tozour, Introduction to Bayesian Networks and Reasoning Under Uncertainty, AI Game Programming Wisdom, 7.2

【概要】「依存グラフ」について貴重な文献。これ以外にはまとまった情報は少ない。この記事もベイジアンネットワークの解説の中で派生的な情報として説明されている。Age of Empire の例だと思われるが、タイトル名

は何処にも銘記されていない。tech tree という呼び方もしている。tech tree については、Brian Schwab、”AI Game Engine Programming” などに解説がある。「ベイズ理論」はForza Motorsport 2 に応用された技術でもある。

[39] Brian Schwab, AI Game Engine Programming, Charles River Media

[40] Jeremy Baxter, Richard Hepplewhite, A Hierarchical Distributed Planning Framework for Simulated Battlefield Entities, <http://mcs.open.ac.uk/plansig2000/Papers/P7.pdf>

[41] HAI Hoang, Stephen Lee-Urban, Hector Munoz-Avila,, Hierarchical Plan Representations for Encoding Strategic Game AI, <http://www.cse.lehigh.edu/~munoz/Publications/AIIDE05.pdf>

[参考文献おわり]

[全文書おわり]