特　集「ゲーム産業における人工知能」

汎用ゲームAIエンジン構築の試みとゲームタイトルでの事例

General-purpose Game AI Engine: Theory and Practice

長谷　洋平　株式会社バンダイナムコスタジオ
Yohei Hase　BANDAI NAMCO Studios Inc.
y2-hase@bns-g.com, https://www.bandainamcostudios.com

Keywords: behavior tree, HTN planning, BDI architecture, AI director, emotion.

1. はじめに

ビデオゲーム（以下，ゲーム）はグラフィック，物理シミュレーション，ネットワークなどの高度な技術の上に成り立つエンタテイメントコンテンツである．人工知能もゲームを構成する技術の一つであり，代表的な使用方法はプレーヤとインタラクトするキャラクタの制御である．ゲームにおける人工知能技術は，業務システムなどの一般的に目にする人工知能技術とは少し趣が異なることからゲームAIと呼ばれることが多い．

ゲームで使用される各種技術は日々高度化しており，技術の共有化が重要な課題となっている．共有化を目指したものとしては，Unreal Engine 4[*1]やUnity[*2]に代表されるようなゲームエンジンがその代表格であり，特定の技術に絞ったものはグラフィックエンジンや物理エンジンなどと呼ばれる．ゲームAIエンジンもその一つで，ゲーム内で使われる人工知能技術について処理を共通化し，効率的に開発できるようにすることを目的としたプログラムである．

ゲームAIはプレーヤとのインタラクションを受けもつという特性上，どのようなゲームかによって求められるAIは変わってくる．そのため，ゲームにおけるAIの開発は依然としてゲームタイトルごとに行うことが多く，技術の共有化は遅れている．これは商用化されているゲーム向けのAIミドルウェアが経路探索などの一部の技術しかサポートしないことからもわかるとおりである．

当社では，このような現状を打開すべく，どのようなゲームジャンルであっても使用できることを目指した汎用的なゲームAIエンジンの開発を行った．本稿では，エージェントAIなどのゲームAIでの各課題において，それを解決するためのゲームAIエンジンの設計と実際のゲームタイトルでの事例を紹介する．

2. エージェントアーキテクチャ

ゲーム内のキャラクタを制御するAIをつくる場合，周りから収集した情報を使って意思決定をし，実際に行動に移すまでの処理の流れを設計する必要がある．そうしたキャラクタ内部の処理の流れに関する設計図をエージェントアーキテクチャと呼ぶ．ゲームにおけるエージェントアーキテクチャの内部は大きく分けて三つの項目から構成される（図1）．

「Perception」はエージェント周囲の環境の認識を行う．環境とは，通路などの移動可能な範囲や敵キャラクタ，アイテムなどゲーム世界全般である．「Brain」は周囲から集めた情報と自分の記憶にある情報などから，次にどのような行動をすべきかを決定する．「Action」は意思決定の結果を実際に実行する項目である．アニメーションの再生という形で体を動かしたり，セリフを発したり，武器のトリガーを引いたりする．

実際のエージェントは多数のモジュールから構成されることになるが，各モジュールはこの三つのどこかに属

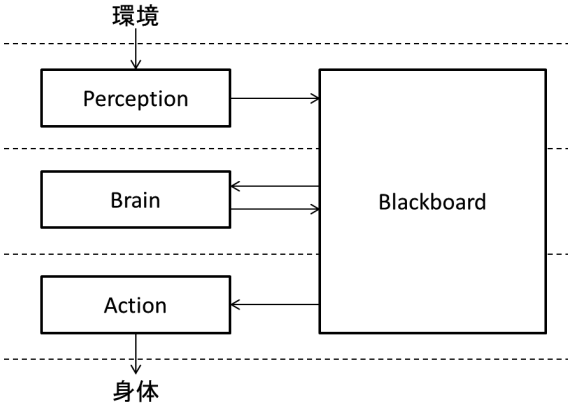

図1　エージェントアーキテクチャの概観

*1 https://www.unrealengine.com/ja/unreal-engine-4
*2 http://japan.unity3d.com/

している．また，「Blackboard」という記憶を扱う特別なモジュールが存在し，各モジュールはBlackboardを通して情報を共有する．モジュールはメッセージングの機能も有している．メッセージとは，メッセージの種類とそのパラメータをセットにしたデータである．このメッセージをモジュール同士がやり取りして処理の依頼や完了の通知を行う．Blackboardとメッセージングによりモジュール間は疎結合になり，モジュールの再利用性を高めている．

エージェントの作成では，どのようなAIをつくりたいかによってどのモジュールをどのように組み合わせるかを考えることになる．モジュールの組合せを変えることでエージェントのタイプは変わってくる．例えば，歩行を扱うモジュールを飛行を扱うモジュールに切り替えるだけで空を飛ぶキャラクタを作成することも可能である．

実際にこのゲームAIエンジンが使われたタイムクライシス5[*3]とLOST REAVERS[*4]では，エージェントアーキテクチャを構成するモジュールとしてLayer-based Perception System（以下，LPS），Behavior Tree，HTNプランニングの三つの技術がコアになっている．ここではまずこれらの技術について解説し，次にそれらを組み合わせたアーキテクチャについて説明する．

2·1 Layer-based Perception System

ゲーム内のエージェントには目や耳といった仮想的な感覚器官を設定しており，人間を含む自然界の生き物同様，環境から得た情報をもとに行動を決定する．

エージェントが外部の環境を知覚する方法には2種類ある．音波や痛みといった外部からの刺激を受けてその発生源を知覚する方法と，隠れられる場所を探すといったような自ら能動的に環境を調べて知覚する方法である．前者はプログラムではイベントとして通知され，それをトリガーとしてLPSが起動する．後者は意思決定を行う層であるBrainのモジュールから必要に応じて起動される．

また，エージェントの種類によって備えている感覚器官が違うのはもちろん，同じ情報を知覚したとしてもそれを同様に認識するとは限らない．生物学では環世界[Uexküll 34]と呼ばれる概念があり，生物はそれぞれがその種特有の知覚世界をもっており，その世界の中で主体的に行動していると考えられている．ゲーム内のエージェントも同様であり，茂みを隠れ場所だと認識するエージェントもいれば，餌だと認識するエージェントもいる．

このように，どのような感覚器官からどのような情報を知覚して，それをどのように認識するかの過程を表したものがLPSであり，その集合としてそのエージェントの環世界を定義する．

LPSの処理は以下の三つの工程から成る．

（1）受動的，または能動的に環境の情報を取得する
（2）情報のフィルタリングや加工を行う
（3）情報の評価を行いスコアを付ける

これらの処理は単機能のレイヤとして実装されており，一つのLPSは多くのレイヤを組み合わせて構成する．（1）では，使用する感覚器官と取得する情報の種類の設定を行っている．複数の感覚器官からの情報を統合することも可能である．（2）では，不必要な情報を削除したり，情報に加工を行ったりする．推論のような既存の情報から新たな情報を生成することも含まれる．（3）では，Brainで情報を使う際に参考にするスコアを付ける．敵対しているキャラクタの場合は危険度であったり，隠れる場所を探す場合はどこが好ましいかなど情報によってさまざまな尺度で評価を行う．

図2は射撃によって攻撃を行うターゲットを選択する例である．周囲にいる敵対しているキャラクタを収集するレイヤ，間に障害物がないかをチェックするレイヤ，対象との距離をもとにスコアリングするレイヤを組み合わせて，銃による攻撃が可能な対象の中で最も適切なものを選んでいる．

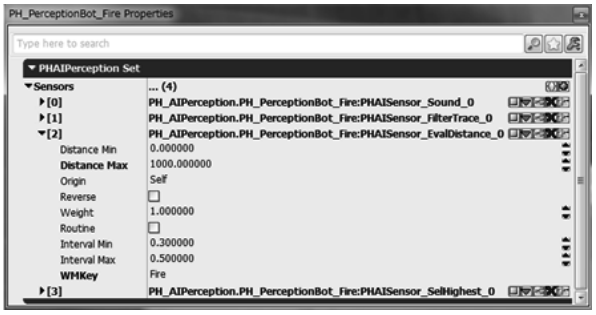

図2 Layer-based Perception Systemによる環境認識の定義例

2·2 Behavior Tree

Behavior TreeはAIの振舞いをツリー状に表現する手法である．AIの思考をグラフィカルに構築することができ，AIに関する知識の有無によらず構築が容易なことからゲーム産業では広く使われている．

ツリーを構成するノードはTask，Composite，Decoratorの3種類に分けられる．

Taskはツリーのリーフにあたるノードである．Taskは主に条件判定やアクションの実行を担当し，結果として成功か失敗を返す．

Compositeは子ノードの実行を制御するノードである．基本となるCompositeには，SequenceとPriority Selectorの二つがある．Sequenceは子ノードを順番に実行することを目的としたノードで，子ノードの実行が成功すれば次の子ノードに処理を移し，すべての子ノードの実行が成功すればSequenceも成功を返す．子ノー

*3 http://bandainamcoent.co.jp/am/vg/timecrisis5/
*4 http://pj-treasure.bn-ent.net/

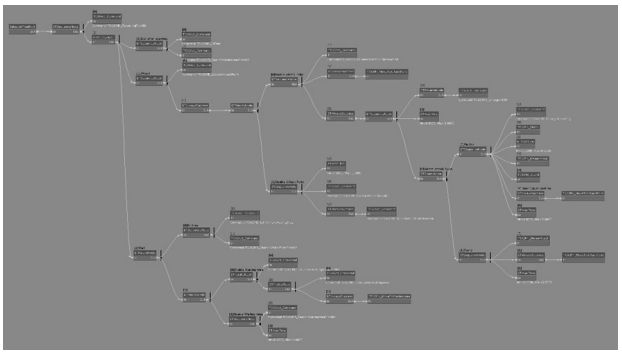

図 3　Behavior Tree エディタ

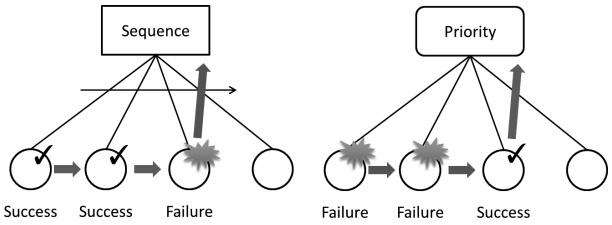

図 4　基本的な Composite ノード

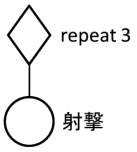

図 5　Decorator ノードの例

ドの実行が失敗すればそこで処理を中断し Sequence も失敗を返す．Priority Selector は子ノードの中から一つを選択することを目的としたノードで，子ノードの実行が失敗すれば次の子ノードに処理を移し，子ノードの実行が成功した段階で Priority Selector も成功を返す．すべての子ノードが失敗すれば Priority Selector も失敗を返す．ほかにも，すべての子ノードを並列に実行する Parallel や，指定した確率分布に従って子ノードを選択する Probability Selector などが存在する[*5]．

Decorator は子ノードの機能を拡張するノードで，子ノードは一つしかとらない．例えば，射撃を 1 回行うノードに 3 回繰り返す Decorator を付けることで，3 回連続で射撃を行わせることができる．ほかにも，指定された秒数経過すると強制的に処理を中断するノードなどがある．

図 6 は周りに敵がいない場合は周囲を巡回し，敵を見つけると攻撃するというシンプルな行動を Behavior Tree で表した例である．ルートの Priority Selector はまず巡回の条件である周りに敵がいないかをチェックする．周りに敵がいない場合はそのまま巡回のアクションを実行していき，敵がいる場合は攻撃のサブツリーに移る．攻撃のツリーのほうでも，弾をもっているかどうかで遠距離から攻撃するか近づいて攻撃するかを判断している．このように単純な条件判定やアクションを木構造で組み合わせていくだけで複雑な振舞いも構築できるので非常に再利用性が高くなる．

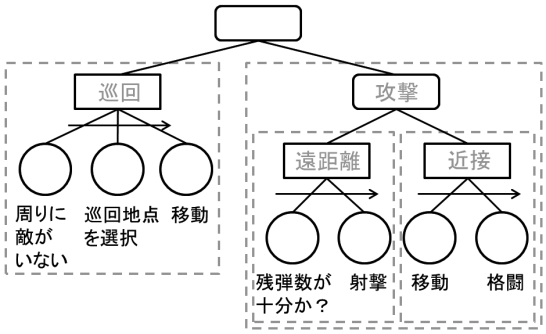

図 6　シンプルな Behavior Tree の例

2·3　HTN プランニング

HTN プランニングとは，現在の状態から目標を達成するための行動を自動で計画するプログラムの一種である．HTN プランニングにおいて，問題解決を行う計画器をプランナと呼び，プランナに渡す問題定義をドメインと呼ぶ．プランナには，初期状態と達成すべきタスクが与えられ，タスクをより小さなサブタスクに分解していくことで具体的なタスクの列であるプランを生成する．

ドメインにはプリミティブタスクと複合タスクという 2 種類のタスクが含まれている．プリミティブタスクとは，それ以上分解しようのない基本的なアクションであり，そのアクションを行うための条件となる状態とアクションを行った後の状態の変化で定義されている．複合タスクとは，タスクをどのように分解するかを定義した抽象的なタスクであり，分解の条件となる状態と分解した結果となるサブタスクの列で定義されている．

2·4　プランとしての Behavior Tree

HTN プランニングにおいて，プランナが生成するプランは基本的に実行すべきタスクの列である．単純に目標を達成することだけを目指すのであれば，実直にタスクを一つずつ実行していけばよい．ただし，ゲームのキャラクタの場合は，そのキャラクタがまるで生きているかのように自然に振る舞う必要がある．

例として，現代戦を想定したゲームにおける機銃をもった兵士キャラクタを考える．敵に対して攻撃を行う場合，機銃で射撃を行うことになるので対象から距離をとって戦うことになる．また，機銃に弾が装填されていない場合はリロードを行う必要もある．単に敵を攻撃するという目標を達成するだけであれば，移動のタスクを実行してからリロードのタスクを実行して敵を攻撃しても達成できる．ただし，戦場というやるかやられるかの場面においては移動のタスクとリロードのタスクを同時に実行したほうがより自然である．さらに，移動とリロー

*5 ノードの具体的な名称は実装により異なる

ドが全く同時に開始されると機械的に見えるのでリロードの実行を少し遅らせたいという要求も出てくる．

このように，ゲームにおいては複数のタスクを並列に実行したり状況の変化に応じて柔軟に行動を選択できたりしなければならず，より柔軟なプランの表現方法が必要になってくる．我々はプランとしてBehavior Treeを生成することでこの問題を解決した．

Behavior TreeにおけるSequenceは子ノードを順番に実行するためのノードである．プランに含まれるプリミティブタスクをBehavior TreeにおけるTaskノードとし，すべてのタスクをSequenceの子ノードとして追加すれば，従来のプランを単純なBehavior Treeで表現することができる．状況が変化しタスクを実行できなくなれば， タスクが失敗を返すことでプランが無効になったことを通知することもできる．

タスクの分解方法は複合タスクに定義されているが，本手法では分解の種類も同時に記述することにする．分解の種類とはそのサブタスクを順番に実行するのか，並列に実行するのかなどを表すタグである．実際の分解の際には，Behavior TreeにおけるCompositeを設定されたタグに応じて選択することで，単純なシーケンスはもちろん，並列実行や確率分布に基づいた実行など多様に展開することができる．Parallelを用いることで，複数

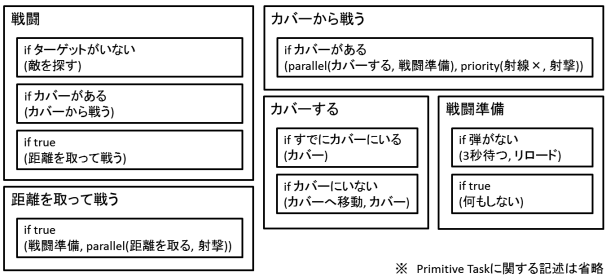

図7 タグ付けをしたドメインの例

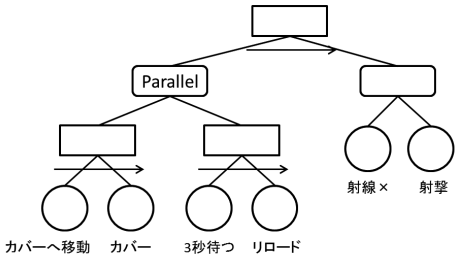

図8 カバーがあり弾が装填されていないときのプラン

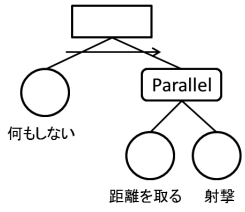

図9 カバーがなく弾が装填されているときのプラン

の行動を同時に行わせることができ，Priority Selectorを用いれば，プレーヤの行動などプランニング時には考慮できない評価項目をプランの実行時にもっていくこともできる．プランの中にリプランニングを要求するTaskを追加することも可能である．

図7～図9は，例としてあげた兵士キャラクタの簡易なAIの本手法での実装例である．状態に応じて適切なプランが生成されるのはもちろん，単なるタスクのシーケンスだけでない柔軟なプランになっている．このように，Behavior Treeのわかりやすさや再利用性の高さなどの利点を生かしつつ，非常に表現力の高いプランを生成することができる．

2·5 BDIアーキテクチャ

自然界の生き物同様の合理的な判断をするキャラクタをつくるには，自然界の生き物の行動決定過程を知る必要がある．哲学者であるBratmanが提唱した「意図の理論」[Bratman 87]によれば，人間は複数の目的をもって生きている．現在の状態から達成すべき目標が決定され，その目標を達成するための計画を立案し，意図を形成する．意図は心的状態であり，人間は意図に沿って行動している．この「意図の理論」に基づいたエージェントモデルをBDIモデルと呼び，BDIモデルに基づいたエージェントアーキテクチャをBDIアーキテクチャと呼ぶ．

BDIアーキテクチャでは，エージェント内部の心的状態である信念，願望，意図をもとに行動を決定することになる．信念とは，エージェント自身を含む環境の事柄についてエージェントが信じている内容であり，主観的なものである．願望とは，エージェントが達成したいと思っている目標であり，行動の動機付けになるものである．意図とは，エージェントがそうしようと考えていることであり，願望を達成するための手段を意味する．

BDIアーキテクチャの動作手順は以下のようになる．

(1) 信念から達成すべき目標（願望）を決め，達成手段の候補を求める
(2) 達成手段の候補から実際に行う手段を熟考して決定する
(3) 決定した手段を意図として実行する
(4) 外部からの知覚をもとに信念を更新する

エージェント内部に心的状態として保持された信念をもとに願望が決められ，それを達成するための手順である意図が生成される．信念自身はエージェント外部からの知覚をもとに更新されるため，時間経過により変化することになる．意図が心的状態として保持されることにより行動に一貫性が生まれ，信念の変化によりその場その場で最適な行動を選べるので，エージェントの振舞いに合理性が増すのである．

BDIアーキテクチャはすでにロボットの分野などで多く使われているが，既存の実装では編集環境や計算資源

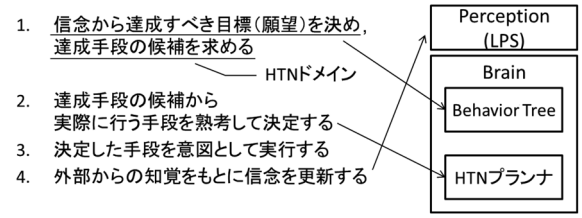

図 10　BDI の動作手順とアーキテクチャの対応

などの問題からゲームでの使用には向いていない．そこで我々は，前述した LPS，Behavior Tree，HTN プランニングを組み合わせて BDI アーキテクチャをゲーム向けに再構築した．BDI アーキテクチャの動作手順と我々のアーキテクチャの対応は図 10 のようになる．

2·1 節で説明したように，LPS はエージェントの環世界を定義している．環世界とはエージェントから見た世界であるから，LPS を通して環境を知覚するということは，環境の情報をそのエージェントの信念空間へ投影していることを意味する．信念は Blackboard に格納されており，LPS によって得た信念も Blackboard に格納されることになる．Behavior Tree には願望の選択ロジックが記述されており，Blackboard に格納された信念を参照してどの願望を達成すべきかを決定する．Parallel で複数の願望が選ばれることもあれば，Priority Selector で優先度に従って願望が評価されることもある．HTN プランナもプランとして Behavior Tree を生成することから，この Behavior Tree を「願望の Behavior Tree」と呼ぶことにする．願望の Behavior Tree で決定された願望は HTN プランナに渡される．ここでいう願望とは具体的には HTN プランニングにおける達成すべきタスクのことであり，HTN プランナは渡されたタスクと初期状態からプランとなる Behavior Tree を生成する．つまり，プランとして生成された Behavior Tree が本手法における意図ということになる．この Behavior Tree を「意図の Behavior Tree」と呼ぶことにする．

意図の Behavior Tree では，手順となる行動の実行とともに意図が有効かどうかのチェックも行う．行動の前提条件が崩れるなどし，意図が無効になると意図の Behavior Tree は失敗を返す．この結果は願望の Behavior Tree にも伝わり，通常の Behavior Tree の動作規則に沿って次の願望が選ばれる．同じ願望を達成するための別の方法を模索するためにリプランニングする，今の願望はあきらめて違う願望の達成を目指すなど，Behavior Tree の組み方でいかようにもできる．同様に，より優先度の高い願望の発生の検知も願望の Behavior Tree で行い，適切に意図の切替えが行われる．

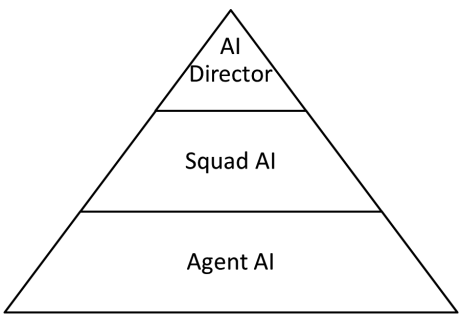

図 11　ゲーム AI のヒエラルキー

3. ゲーム AI のヒエラルキー

アクションゲームなど多くのゲームには AI で制御されたエージェントが複数おり，それぞれが協調して動いている．例えば，同じ敵を囲うような位置取りをしたり，傷ついた仲間を助けに行ったりする．そのような協調動作を行えるようにするために，我々のシステムでは AI を階層的に管理できる仕組みを用意している．

AI はその役割に応じて 3 種類に分けられる．Agent AI はその名のとおりエージェントを制御する AI であり，階層の一番下に位置する．Squad AI は AI のグループを管理する AI であり，仲間の状況を教えたり指示を出したりする．階層の一番トップに位置するのが AI Director である．AI Director はメタ AI とも呼ばれ [三宅 15]，ほかの AI を含めたゲーム全体を大局的に制御する AI である．

この AI の層の数はゲームによって変わってくる．格闘ゲームのようなせいぜい 1 ～ 2 体のエージェントしか制御しないゲームの場合は Agent AI の 1 層のみにもなり得るし，戦争を模した戦略ゲームのように多数のエージェントを扱う場合は Squad AI を複数の層に分けることもある．

AI Director と Squad AI は 3D モデルという目に見える身体はもっていないが，神のような概念的なエージェントだと捉えれば Agent AI のアーキテクチャを流用しつつ作成することができる．

異なる層の間での通信にはエージェントアーキテクチャ内のモジュール間通信と同様，Blackboard とメッセージングを用いる．Blackboard では，情報の共有を行う．例えば，ターゲットの共有やセマフォのような資源管理などである．Blackboard はほかの Blackboard と親子付けできる構造になっており，階層上で自分の親にあたる AI の Blackboard を自分の Blackboard の親に指定する．情報にアクセスする際，自分の Blackboard に情報があればそれを使い，なければ親の Blackboard から情報を取得しようと試みる．こうすることで，親の情報も自分の情報と同じように透過的にアクセスできるので上の層を意識しなくてよくなる．メッセージングでは，AI 間のコミュニケーションを実現する．Squad AI などの上位の AI はネットワークにおけるルータのようにメッセージを配下全員にブロードキャストしたり適切な AI に転送したりする機能をもつ．この機能を使うことで，敵を見つけたことを仲間に知らせたり，助けを呼んだときに適切なエージェントをアサインしたりできる．

親にメッセージを投げれば親が適切に処理してくれるのでエージェント自身は個々の仲間を意識する必要はなくなる．

前述したように層の数はゲームによって変わってくる．これはゲームの開発中に仕様の変更が発生する可能性があることも意味する．それだけでなく，ゲーム中に階層の構成が動的に変わることもあり得る．このようなことから，ゲームにおいてはほかのAIをあまり意識せずとも協調動作ができる仕組みを用意しておくことは重要になる．

4. AI Director

ゲームにおいて，キャラクタが動けるエリア，アイテムや敵の配置，イベントなどのゲームをプレイするためのステージ構成全体をレベルと呼ぶ．AI Directorはプレーヤがよりゲームを楽しめるように，レベルを動的に制御するAIである．Agent AIやSquad AIはゲーム中に複数存在するが，AI Directorは階層のトップに位置するAIのため，ゲーム中には一つしか存在しない．ここではLOST REAVERSで実装したAI Directorについて紹介する．

4·1 ペーシング

ゲームには緩急の波を付けると面白くなることが一般的に知られている[稲葉16, 大野16]．ここでの緩急の波とは，「ハラハラする場面」と「落ち着ける場面」が適度に繰り返される状態のことである．緩急の波を付けるとよいのはゲームに限ったことではない．よく知られるストーリー構成法にThree Act Structure（三幕構成）[Field 79]がある（図12）．Three Act Structureはストーリー全体を三幕に分け，どこにどういった展開をもってくればよいのかを示したものである．小説や映画はおおむねこの構成に従っており，観客をひきつける緩急の波がある程度パターン化されていることを意味する．ゲームでもこの構成法は使われるが，インタラクティブなメディアであるゲームでは，あらかじめ決まった波を提供するだけでは十分でないことが多い．プレーヤはゲームの中で自由に行動し，ゲームの物語を主観的に体験することになるため，決まった波ではプレーヤの感情とかい離することがあるのである．そこで我々はプレーヤの行動履歴をもとに波を動的に演出するAIを作成した．このAIは主に以下の三つの要素から構成される．

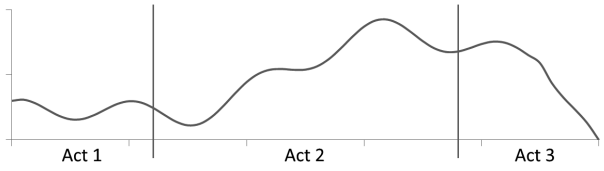

図12 Three Act Structure

●感情の推測
●波の形成
●クラスタリング

4·2 感情の推測

プレーヤに合わせて感情を揺さぶるには，まずは感情を推測できる必要がある．ここでの感情とはどれだけハラハラしているかであり，言うなれば緊張度である．通常，ゲーム機には脈拍，脳波などの生体情報を計測できるセンサは付いておらず，唯一の入力であるコントローラのボタン入力から推測を行わなくてはならない．そこで，どのようなデータを用いればよいかを調べるために，AI Directorを搭載せずに制作していたレベルをプレイし，プレイログの収集を開始した．プレイログにはボタン入力はもちろん，入力の結果行ったキャラクタのアクションや体力などのプレーヤキャラクタの状態も含めている．次にプレイログ中のそれぞれの場面について，どの程度ハラハラしていたかを数段階でつけた．このハラハラ度合いをもとにプレイログの比較を行い，感情の推測に有力そうな説明変数を抽出し，この説明変数を使用して線形回帰による分析を行い，モデルを構築した．

このモデルにより推測した感情の度合いをゲーム中に表示した例が図13である．分析に使用した感情度は主観的で段階も少ないことから，このままでは推測の精度は著しく悪い．そのため，感情度を表示した状態のゲームを何度も遊び，感情度の推測値と実際の感情が近づくように係数の調整を行った．

今回は開発スケジュールの都合とそこまでの高精度は必要でなかったことから説明変数の選択などで手作業に頼った部分が多かったが，モデル構築の自動化が今後の課題である．

4·3 波の形成

感情の推測ができれば，次はその感情を揺さぶることで波をつくる．具体的には，感情度が低い場合はよりハラハラするように，感情度が高い場合は落ち着けるようにレベルを制御する．

LOST REAVERSで採用した主たるレベルの制御方法は2種類ある．一つは敵の数のコントロールで，もう一つは敵の行動のコントロールである．

敵の数のコントロールでは，ハラハラさせたいときに

図13 推測した感情度のゲーム中での表示

敵を生成し対処が困難な状況をつくる．落ち着かせたいときは敵の生成を止めればいつかはプレーヤが敵を倒し切って落ち着ける状況になる．一口に敵の生成といってもプレーヤの目の前にぱっと現れても不自然で，プレーヤに到達できないところに生成しても意味がない．部屋の奥など見えないところで生成された敵がプレーヤのところに向かっていって，気付けば囲まれているような状況になるのが好ましい．これを行うためにはゲームのステージデータの分析が必要になる．LOST REAVERSでは，ゲーム中のキャラクタが歩ける場所にAIの経路探索などで用いるナビゲーションメッシュと呼ばれるデータが設定されており（図14），このナビゲーションメッシュを分析して壁などでプレーヤからは見えない場所でプレーヤの所まで歩いて行ける場所を検索して生成場所を決定している．

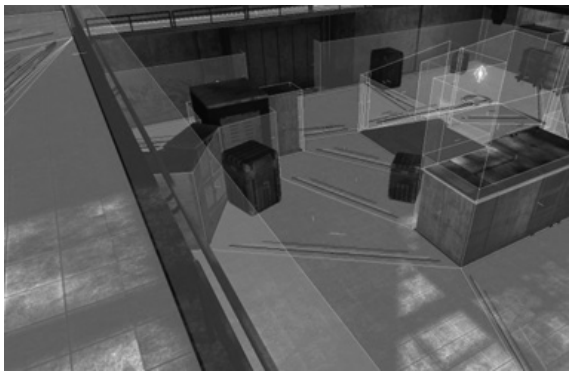

図14　ナビゲーションメッシュ

敵の数を制御してその状況をつくるには，生成した敵がプレーヤのもとにたどり着くまでの時間やプレーヤが敵を倒し切るまでの時間が必要で感情をすぐさま揺さぶれなかったり，処理負荷などによる数の上限もあるため，敵の行動自体も制御して状況をつくっている．アプローチとしては次のようになる．まず，感情度の推測と同様の方法でプレーヤのゲームスキルを推測する．キャラクタには装備している武器などによる強さの概念があるためその値を使うこともできるが，わざわざ行動履歴から推測しているのはプレーヤのゲームの腕前も加味するためである．次に，プレーヤをハラハラさせたい場面では，敵が下手なプレーヤを狙うことでうまく対処できず苦戦する状況をつくり出す．落ち着かせたい場面では，うまいプレーヤを狙うことで効率良く対処しやすいようにしてサクサク進める状況をつくり出している．実際には，ターゲットの選択だけでなくステージのどこが安全でどこが危険かも分析して，やられやすい，もしくはやられにくい行動をとっている．分析には2·1節で説明したLPSを使用している．LPSではプレーヤの情報を含めた危険な地点と味方がいる場所などの安全な地点の情報を収集し，実際にどれほど危険か（安全か）を評価している．このLPSの結果を周囲に伝搬させてステージの各地点の危険度を表したInfluence Mapを作成する（図15）．Influence Mapを参照することであらゆる地点の危険度がわかるので，敵がわざと危険な地点に陣取って倒されやすくしたりできる．

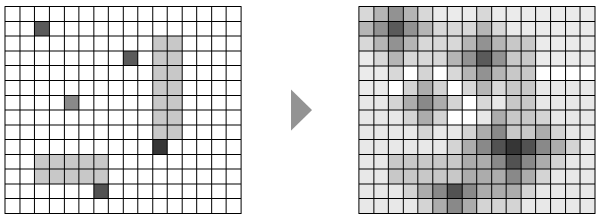

図15　Influence Mapの作成手順

波の周波数や大きさに関わるパラメータはレベルデザイナによって設定されている．AI Directorはレベルデザイナが設定した理想とする波に極力沿うように敵の生成や行動をコントロールする．このパラメータはステージの進行度に応じて変化させることができるので，Three Act Structureのように序盤は落ち着いた状況をつくっておき，クライマックスに近づくにつれて激しくするようなことも行っている．

4·4　クラスタリング

LOST REAVERSは複数人が一緒に遊ぶオンラインゲームである．また，一緒に遊んでいる人同士が必ずしも一緒に行動しないといけないわけではなく，バラバラに行動することも許容したゲームデザインになっている．一緒に行動している人同士ではそれぞれの人が同じ状況を体験しているため感情の共有も可能だが，バラバラに行動している人の間では現在体験している出来事が違うため感情の共有はできない．そのため，一緒に行動しているグループ単位で波の形成をする必要がある．

このグループの判別は動的に行う必要がある．一緒に行動していた人が別行動をとったり，後から合流したりすることも想定しないといけないためである．そのため，グループの判別にはウォード法による階層的クラスタリングを用いた．階層的クラスタリングでは結果として図16のような樹形図が得られる．この樹形図をもとに後からクラスタへの分割を行えるので，あらかじめクラスタ数を決めておく必要がない．プレーヤキャラクタ間の距離を基準としてクラスタリングを行い，良い感じのまとまりだと認められる値をしきい値としてクラスタに分けることで，一緒に行動しているグループを動的に把握

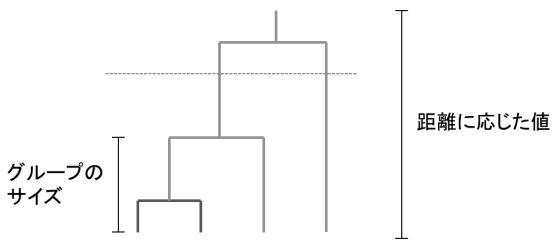

図16　階層的クラスタリング

することができている．

5. ま と め

本稿では，AIを構成するエージェントアーキテクチャとマルチエージェントを管理する仕組みについて，コアとなる理論から実際の事例までを紹介した．汎用的なゲームAIエンジンとしてこれらをサポートすることで，ゲームタイトルごとに閉じていたAI技術をオープンにし，高度な技術を使いやすくすることができた．

ここで紹介した課題はごく一部であり，ほかにもさまざまな課題が存在する．例えば，VR技術の発達により，自然言語などのより自然なインタフェースでのキャラクタとのインタラクションの需要が増えていたり[長谷16]，フリーミアムの台頭によって長く遊ばれるゲームが必要になり，機械学習などを使った開発工程の自動化による運営コストの軽減が求められていたりする[友部16]．このようなさまざまな課題にも対応すべく，現在も新たな技術の開発を進めている．

◇ 参 考 文 献 ◇

[Bratman 87] Bratman, M. E.: *Intention, Plans, and Practical Reason,* Harvard University Press（1987）（門脇俊介，高橋久一郎 訳：意図と行為―合理性，計画，実践的推論―，産業図書（1994））

[Field 79] Field, S.: *Screenplay: The Foundations of Screenwriting,* Dell Publishing Company（1979）（安藤紘平，加藤正人 訳：映画を書くためにあなたがしなくてはならないこと シド・フィールドの脚本術，フィルムアート社（2009））

[長谷 16] 長谷洋平：ゲーム産業における人工知能技術（人工知能の未来へ―全脳アーキテクチャ，ディープラーニングを用いた人工生命，ゲームAI―の一部），*CEDEC*（2016）

[稲葉 16] 稲葉敦志：Action games without borders: Making platinum-quality games for the World, *GDC*（2016）

[三宅 15] 三宅陽一郎：ディジタルゲームにおける人工知能技術の応用の現在，人工知能，Vol. 30, No. 1, pp. 45-64（2015）

[大野 16] 大野功二：「コントラスト」で考えるゲームデザイン・レベルデザイン，*CEDEC*（2016）

[友部 16] 友部博教，半田豊和：AIによるゲームアプリ運用の課題解決へのアプローチ，*CEDEC*（2016）

[Uexküll 34] Uexküll, von J. J. B. and Kriszat, G.: *Streifzüge durch die Umwelten von Tieren und Menschen Ein Bilderbuch unsichtbarer Welten,* Springer Berlin Heidelberg（1934）（日高敏隆，羽田節子 訳：生物から見た世界，岩波書店（2005））

2017年1月20日 受理

著 者 紹 介

長谷 洋平（正会員）

2009年株式会社バンダイナムコゲームス（当時）入社．フライトシューティングゲームのエースコンバットアサルト・ホライゾンにてオンライン対戦用のAIなどを担当．以来，多くのゲームタイトルにてゲームAIの設計，実装を担当．現在はそれに加えて，最新の人工知能技術のリサーチや社内教育にも取り組む．ゲーム開発者向けカンファレンスであるCEDECにて多数講演．