大規模デジタルゲームにおける人工知能の一般的体系と実装 —FINAL FANTASY XV の実例を基に—

Game AI General Theory and its Implementation in AAA Digital Game —A Case Study of AI System in FINAL FANTASY XV—

三宅 陽一郎 株式会社スクウェア・エニックス
Youichiro Miyake SQUARE ENIX CO., LTD.
miyakey@square-enix.com, <http://www.jp.square-enix.com/>

Keywords: game AI, behavior tree, state machine, navigation AI, meta-AI, character AI, decision-making, AI Graph

Summary

A game AI general theory has been researched and developed in game industry in the world, and a new game AI general theory of AI in digital game is supposed with three AIs. For a large scale of game, a game AI system consists of three types of AI such as meta-AI, character AI, and navigation AI. A meta-AI is to control a game dynamically from a bird-view by watching a player's behavior. A character AI is a brain of a game character such as a buddy, a monster, or a villager to make a decision in real-time. A navigation AI is to recognize an environment of a game to find a path or a best location to move dynamically. Especially, character AI is a main topic to study in game development, and it includes many fields such as multi-layered structure, character animation, agent architecture, decision-making modules, and so on. A new method of decision-making of combination of behavior trees and state machines is supposed. It is called AI Graph. The game AI general theory was applied to an AI system of an action-RPG game "FINAL FANTASY XV". The results are showed in the paper. All characters' decision-making system in FINAL FANTASY XV are based on AI Graphs. An AI Graph Editor is a tool to make an AI Graph only by using a mouse and simple text inputs. A dynamics of the new method is showed by explaining AI Graph Editor precisely.

1. 背景

本論文は大規模なデジタルゲームにおける人工知能技術の理論の提示と、その実装事例である。大規模デジタルゲームから要求される人工知能を実現する設計と具体的な手法を解説する。

本文でゲームと言えばデジタルゲームのことを意味する。大規模なデジタルゲームとは、ここでは PlayStation 4 (ソニー・インタラクティブエンタテインメント社, 2013) や Xbox One (Microsoft 社, 2013) など据え置き型の大型ゲーム機や、ハイスペックなパソコン用の大容量のゲームを指す。AAA ゲームタイトルとも呼ばれる。家庭用ゲーム専用機は「コンシューマ・ゲーム機」と呼ばれる。

大型ゲーム機の特徴はグラフィックや物理計算を始めとする強力なシミュレーション能力と大きなメモリ領域である。そこではまた人工知能にも大きな計算、メモリが割り当てられること、物理法則を含む正確な三次元空間がゲームワールドとして再現されていることから、デジタルゲームにおける人工知能技術の導入と発展がこの 15 年間継続されて来た。たとえば、それはゲーム産業における各企業の AI エンジニアがそれぞれのゲームタイトルの AI の解説を寄稿した文献の集積として見られる[三宅 17, Rabin 02, Rabin

04, Rabin 06, Rabin 08, Rabin 13, Rabin 15, Rabin 17]。

デジタルゲームに登場する人物・モンスターは総称してキャラクターと呼ばれる。このうち、プレイヤーが操作するキャラクターのことはプレイヤーキャラクター (PC), それ以外の AI が動かすキャラクターはノンプレイヤーキャラクター (NPC) と呼ばれる。

ゲームステージ全体は「環境」と呼ばれる。地形、物、空など、すべてを含む用語である。またゲーム内の岩、草、来、ドアなどのものは「オブジェクト」と呼ばれる。ゲームを構成するそれぞれのデータは「アセット」と呼ばれる。

1.1 静的システムから動的システムへ

80, 90 年代前半までのファミリーコンピュータ (任天堂社, 1983), スーパーファミコン (任天堂社, 1990) の時代のデジタルゲームにおける人工知能は、以下に挙げるように「静的なシステム」であった[三宅 08a]。

- (1) ノンプレイヤーキャラクターが移動できる領域が部屋やエリアとして限定されている。
- (2) その領域に限定された活動をするキャラクターの知能が用意されている。
- (3) ノンプレイヤーキャラクターに AI が搭載されておらず、ゲームシステムから操作される。

「ゲームシステム」はゲームの進行を司るプログラムである。ゲームステージを入れ替えたり、敵を出現させたり、メニュー画面を出し入れするなど、ゲーム全体の基本的な動作を行う。(3)は、キャラクターに自律的人工知能がなく、他のオブジェクトと同じように、ゲームシステムによって動かされることを意味している。たとえば、プレイヤーが左から来たので、右に登場させて左向きに動かす、などである。

図1 3つの人工知能の協調図

このような静的システムの特徴は、キャラクターがゲームステージの一部（ギミック）として動く、ギミックより少しインタラクティブな要素の強いオブジェクトとして決められた役割を果たすところにある。特に日本のゲームはこのようなギミックとオブジェクトを積み上げたレベルデザイン（ゲームステージを組み上げギミックとキャラクターの動作を調整すること）に長けてきた。静的なシステムから、90年代後半のゲームの3D化と大規模化を経て、ゲームの人工知能は動的な方向に進化してきた。

ここで言う「大規模化」とは、たとえば、80年代ではキャラクターの縦横のスケールを1とすると、80年代のゲームマップ全体のスケールは、数十×数十ほどであった。90年代から00年代にはそれが3Dとなり数百×数百のスケールとなった。10年代では10万×10万のスケールとなった。また、マップ上に起伏や山や湖などの地形が展開し、岩や木々のようなオブジェクトが数十万個置かれ複雑化している。このようにゲームステージは大規模化、複雑化の傾向をたどっており、その環境内でキャラクターは、環境を認識し、自らの目的を果たすのに必要な行動を形成することが求められる。この変化は以下のように表現できる。

A) ノンプレイヤーキャラクターの人工知能（キャラクターAI）が環境から独立した自律的知能となる [Isla 01, Isla 02]

B) ゲームシステムからノンプレイヤーキャラクターを操作していた人工知能は「メタ AI」となる [岩谷 05a, 岩谷 05b].

C) 複雑で大規模化した環境を認識する「ナビゲーション AI」が発展する [Snook 01].

「ナビゲーション AI」が発展する [Snook 01].

A) の「環境から独立」とは、ノンプレイヤーキャラクターが局所的な場所でのみ動くことができるのではなく、いかなる場所でも、地形や岩や池といったオブジェクトを認識し、自分自身の身体動作を組み立てることができるキャラクターとなる、ということである。ここで言う「ナビゲーション AI」は、多様な地形を統一的に解釈する人工知能である。C) は、ノンプレイヤーキャラクターが環境を認識するために行っていた、地形解析、パス検索などの機能が肥大化したために、「ナビゲーション AI」モジュールと独立した、という変化である。

80年代、90年代前半ではゲームシステム部分と AI が、グラフィクス、メニュー画面処理、スコア処理、破壊処理など、各種ゲーム機能のコード群の中で混在していた。そこから95年から2000年にかけて3つの独立した人工知能「キャラクターAI」、「ナビゲーション AI」、「メタ AI」へと独立すると同時に、互いに協調するシステムとなった（図1）。

デジタルゲームの人工知能はエンターテインメント AI であり、その主目的はゲームタイトルの狙いとするユーザー体験（ユーザー・エクスペリエンス）を作り出すことにある。そして、そのような体験を作り出すことは主にゲームデザイナーの仕事へと近年になるほど分化していった。

また2010年代には、ゲームの大型化に伴い大量のノンプレイヤーキャラクターの身体動作とその人工知能を作り出さねばならなくなった。簡易プログラミング言語であるスクリプト言語を複数のゲームデザイナーが記述することは、可視性がなく、またその書き方の自由度の高さが逆にバグ発生の温床となっていた。そこで、キャラクターAIの製作工程は、スクリプト言語を使わず、グラフ構造によって意思決定システムをグラフィカルに表現し（たとえば、4.5 AI Graph Editor など、図12～図14）、GUIベースで、ゲームデザイナーがマウスでノードを生成し、必要なパラメータを入力して意思決定システムを作成していくような操作ができるようにツール群が整備されることとなった。ツール作りもゲーム AI 開発技術の一環となった。

本文では、以上の背景の上に、現代（2010年代）のAAAゲームタイトルのゲームにおける人工知能の作り方を提示し、現代の大型ゲームが共通に直面する普遍な問題とそれを解決する技術を提示する。その実例として「FINAL FANTASY XV」（スクウェア・エニックス、2016、以下FFXV）への実装とAI専用ツールについて報告する（図2）。

図2 「FINAL FANTASY XV」のゲーム画面

2. 大型ゲーム人工知能の全体の構成

大型ゲームにおける人工知能の全体の構成は、ゲーム完成までの作業フロー（開発者全員の製作ワークフロー全体）を見越し、最小限のシステムで最大限の人工知能の効果をもたらす必要がある。最適な手法は、ゲームのスケールごとに異なる。

80年代、90年代を通じて、最もよく取られた手法はゲームの各ステージ、状況ごとにキャラクターを動かすプログラムを書くことである。そのような手法は開発者がマップのそれぞれの場所と状況でキャラクターが為すべきことを思考し、その解釈を開発会社独自のオリジナルスクリプト言語や一般に流布しているスクリプト言語を使用して記述することが多いために一般に「スクリプテッド AI」（Scripted AI）と呼ばれる。スクリプテッド AI は開発者がゲーム状態をある程度想定できる小型・中型のゲームで最も効果的である。

しかし、ゲームの複雑化・大規模化に伴いスクリプトが複雑化・肥大化すると、膨大なスクリプト量となり、可視性に乏しく、バグが出やすく、長期開発における保守性（メンテナンス）が減少する。これを解決するためにゲーム開発はちょうど 2000 年頃に「スクリプテッド AI」から「自律型 AI」へ移行してきた。大規模なゲームで最適な手法は、キャラクター自身が動的に周囲の環境を認識し、意思決定し、運動を作り出す「自律型 AI」である。ここからデジタルゲームにおける人工知能の急速な進化が始まる[Griesemer 02]。

自律型 AI への移行で、ゲーム産業は多くの自律型人工知能技術をロボティクスから輸入して発展させていくことになる。同時に、俯瞰的な視点からキャラクターを制御していた部分は「メタ AI」として独立し、地形を解釈していた部分は「ナビゲーション AI」として独立した。

この3つに分散した人工知能が互いに協調するシステムが現代のデジタルゲームの人工知能システムである[三宅 15, 三宅 16a]。以下に、FFXV におけるこの

3つの AI がどのように実装され、どのような知的機能を実現するかを解説していく。

2.1 大規模デジタルゲームの人工知能理論

大規模なデジタルゲームにおいてノンプレイヤーキャラクターに求められるのは、環境で自律的に活動する自律型エージェントとしての思考と、同時にゲームシナリオ上で与えられた役割の通りに演技をする能力である。たとえば、ノンプレイヤーキャラクターは「戦闘が始まる前に、剣を振り上げて掛け声をかける」などをしてやる気があることを見せたり、街に到着した時にこれから行うミッションをプレイヤーキャラクターの側へ来て身振りや言葉で説明したり、仲間が負傷した時に泣きながら側に駆け寄るなど、その場に合った演技をする必要がある。

そこでキャラクターAI には、自律性（autonomous）と他律性（controllable）を兼ね合わせるシステムが要求される。ここでは前者を「自律型キャラクターAI」、後者を「演技用キャラクターAI」と呼ぶ。そこで、

- (1) 自律性を持つ自律型キャラクターAI
- (2) ゲームを俯瞰的に見て文脈に沿った演技をさせる「メタ AI」
- (3) 「メタ AI」からの命令を受けて、指示通りに動く演技用キャラクターAI

が必要である（図 3）。「メタ AI」に関しては第 3 章、「キャラクターAI」に関しては第 4 章で述べる。またさまざまな特徴を持つ地形ごとにコードを書く必要がないように、地形を抽象化する「ナビゲーション AI」は第 5 章で説明する。「ナビゲーション AI」は環境を抽象化した状況情報を「キャラクターAI」、「メタ AI」に提供する。具体的には、敵へ弓を放つのに適した高台の地点を地形解析から提示する、そこに至る最も安全な経路を提供する、などである。

メタ AI からキャラクターAI に指示を与えるインターフェースが必要である。このインターフェースを通じて、メタ AI の「ここに行け」、「この地点に立て」「待機せよ」などシンプルな指示を受け取り、キャラクターAI は状況に合わせてその指示を実行する。

キャラクターAI には、ゲーム産業では一般に「多様性」、「拡張性」、「カスタマイズ性」が必要とされる。「多様性」は統一的な仕組みでさまざまな AI を作ることができる仕組みであり、「拡張性」は必要に応じて知的機能を拡張して行ける機能であり、「カスタマイズ性」はパラメータやグラフ構造をエディターやツールから調整できることである。これらを統一的に解決する仕組みとして「AI Graph」が導入される（4.4.4.5 節）。「自律型 AI」、「演技用 AI」も、後述する AI Graph を用いて構築される。

次に大規模デジタルゲームならではの問題として、大量のキャラクターAIを制作する必要がある。そのために、多種多様なキャラクターAIを効率的に実装する必要がある。これはAI Graphの機能として「アセット化・オーバーライド機能」として実現される(4・7 節)。

またAI Graphの機能として「AI モード」があり、メタAIはこのキャラクターAIのAIモードを指定することで、キャラクターにその場に適した演技をさせることができる(4・8 節)。たとえば、「プレイヤーキャラクターの側に行ってセリフを喋らせる」、「プレイヤーキャラクターの側へ行って回復する」などである。

この詳細を第3章から第5章にわたって解説する。

図3 一般的なゲームAIの構成

2.2 AI開発チーム構成

ここで大規模開発のAIに限ったチーム編成を、FFXVを例に解説する。まず技術開発のチームがあり、ここでゲームタイトル開発前から、ゲームタイトルに予想される要件を考慮しつつ汎用AIツールとランタイム（実行ソフトウェア）の設計と製作がなされる。後述するAI Graph、ナビゲーションAIは、このチームで製作された。実際のゲームタイトル開発が始まると、「モンスター」、「仲間キャラクター」、「敵兵」それぞれの担当チームが同一のシステムを使用し、そのフィードバックの個々の要求に応えながら、改善を重ねて行く。このうち「モンスター」、「仲間キャラクター」、「敵兵」チームは、それぞれのキャラクターAIを「AI Graph」によって高品質なAIとして行った。「街」チームは大量の群衆(Crowd)を作る必要があり、それはスマートロケーションという別手法が実装された[三宅17]。

3. メタAI

大型ゲームにおけるメタAIの役割は年々増加して

いる。キャラクターが直面する局所的な状況が複雑になる中で、メタAIはゲーム全体の流れを作り出す役割を持つ。

メタAIは古典的メタAIの時代(1980~90年代)と現代のメタAI(00年代)以降に分かれる。古典的メタAIは、ユーザーのゲームプレイからユーザースキル(ユーザーのゲームのうまさ)を判定し、ゲームの難易度を調整する人工知能であった[岩谷05a, 岩谷05b]。これは「セルフゲームコントロールシステム」の名前で使用されていた。現代のメタAIは、出現キャラクターの選択、キャラクターの配置位置、出現タイミング、ダンジョンの形、ストーリーなどを変化させる機能を持ち、ユーザーのスキル、特性を解析した上で、変化の仕方を決定する、より積極的にゲームへ干渉する機能を持つ[Booth09a, Booth09b]。

FFXVにおけるメタAIは、ゲームステージ上の状態を俯瞰的に観測しながら、キャラクターに指示を与える[上段16, 三宅16d]。一般にメタAIには強いメタAIと弱いメタAIがある。「強い」、「弱い」はそのメタAIのゲームへの影響力・干渉力の強弱を表現している。強いメタAIの場合、ゲームデザインとメタAIの強い関係がある。たとえば、プレイヤーキャラクターが不利になると風向きや天候を変える、プレイヤーが勝っている時は近くに岩を転がして敵キャラクターを有利にする、などである。メタAIは、頻繁に敵キャラクターやゲーム内オブジェクトに制御をかけることで、ゲーム状態を大きく変化させる。たとえば、戦闘では、キャラクターの周囲の敵・オブジェクトが一度はメタAIからの干渉を受ける、といった頻度である。

弱いメタAIとは、最小限の制御に留めることで、ピンポイントでゲームへ干渉し、ゲーム展開を良い方向へ微調整するAIである。たとえば、一戦闘シーンで一度から数度、という頻度である。FFXVの場合は弱いメタAIである(図4)。たとえば、FFXVは仲間キャラクター三人とプレイヤーキャラクターで戦闘を行うが、プレイヤーが体力を失って戦闘不能になると、それぞれの仲間キャラクターがプレイヤーキャラクターを回復しに近づいて来て、プレイヤーを回復しようとする。このシーンは仲間キャラクターがとても賢くなく見える。理想的なシーンは、一人は助けに来るが、二人はそのまま戦闘を続けることである。そこで、回復する場合は、それぞれのキャラクターがいったんメタAIに許可の申請を出す形にし、メタAIは各キャラクターの状態をチェックし、遠いキャラクター、他の行動で忙しいキャラクターを除外し、その時に最も戦闘していないキャラクターを選択して申請した行動の許可を出す。残された二人のキャラクターには、戦闘を継続しながら、助けに行く仲間キャラクターに「任せた」というセリフを言うようにメタAIから指示を出す。このように、弱いメタAIの場合は、それな

しでもゲームデザインが成立するが、ゲームデザインの偶発的な破れを補完するという役割である。特に開発後半で現れる諸問題をキャラクターAIの修正ではなく、メタAIの機能の実装という方法で解消する、という利点を持つ。

図4 メタAIからの指示でプレイヤー・キャラクターを助ける仲間キャラクター（FFXV）

具体的に、FFXVのメタAIは、

- ・適切な指示を
- ・適切な相手に
- ・適切なタイミングで命令し、
- ・全体をゲーム状況に沿った感じにコントロールする

という役割を担う。例えばレベルシステム（そのステージにおけるゲームシステムのこと）から、「仲間キャラクターの誰かにここまでプレイヤーキャラクターを先導して欲しい」という指示があった場合、各キャラクターAIが判断すると全てのキャラクターが先導しようとしてしまうので、調停が必要になる。キャラクター間の調停をする代わりに、指示はメタAIに出され、メタAIが最も適切なキャラクターを一つだけ選ぶことでこのようなことが起きることを防ぐことができる。

メタAIの機能によって、自律型AIを持つキャラクターの行動をゲームの状況・文脈に沿った行動へ適時修正することが可能となる。

4. キャラクターAI

ノンプレイヤーキャラクター全体を作成することは、複数の技術の複合である。モンスター、仲間、兵士など、それぞれのノンプレイヤーキャラクターに対してゲーム開発者による製作ワークフローがあり、特に開発作業が集中する開発プロセスである。一体のキャラクターに付き数人～数十人が携わることが多い。身体動作を決める「ボーン」（キャラクターモデルの中

心に入れる骨構造）「リグ」（首飾りや長髪などキャラクターの身体以外の骨構造）「アニメーション」、見た目を決める「コンセプトアート」、「モデリング」、「グラフィクス」、足音や装備品、環境との衝突音を決める「効果音（SE）」、知的な振る舞いを決める「意思決定」、「セリフ」など複数の工程を繰り返し巡回しながらキャラクターは完成される。

キャラクターAI全体の設計において、アニメーション・モジュール（モジュールはここではソフトウェアの部分という意味で用いる）と意思決定モジュールの接続関係をいかに成立させるか、という問題は、現在もゲーム産業で最も大きな未解決な課題である。

意思決定モジュールは行動を決定しどのようなモーションを再生するかを決定する。アニメーション・モジュールはモーションデータを持ち、モーションを再生するモジュールである。意思決定モジュールから、一つのモーション再生のメッセージがアニメーション・モジュールに伝えられ、そのモーションが再生されるのが基本的な動作である。しかし、意思決定モジュール、アニメーション・モジュール、どちらのモジュールも継続的に動作し、ノンプレイヤーキャラクターはこの2つのモジュールの動的な連携によって環境の中で目的に沿うように身体を動かさねばならない。二つの動的なシステムを環境という制約の中で動的に同期させ、意思決定モジュールの決定した行動を完遂すること、が目標である。

図5 「身体層」（ボディ・レイヤー）の導入

まず最上層にあるのが「意思決定」の層である。意思決定層の機能は、行動や方針を決定し出力することである。意思決定からモーションを一つ一つ直接指定する形は90年代後半から2000年前半まではよく取られていた形式だった。しかし意思決定が複雑になり、モーション数が増えていくと互いの構造のパッケージ性と拡張性を壊すようになった。そこで、互いの独立性を維持するために、その間に中間層を導入した設

計がなされるようになった。この中間層は時に五層、六層となる場合もある。

FFXV では、意思決定とアニメーション層の間に「身体層」(ボディ・レイヤー)を導入した。「身体層」は身体の状態を大きく定義している(図 5) [Imamura 16]。「走っている」、「ジャンプしている」、「梯子を登っている」状態(ステート)を管理する。身体層は状態ごとに行える行動を制限する。たとえば「梯子を登っている」時に「剣を振る」ことはできない、などの調整を行うことで、意思決定とアニメーションの独立性を高める。さらに反射的に行った行動(ダメージリアクションなど)による身体の状態変化を意思決定層に知らせる役割を持つ[三宅 16c]。

最下層にあるのが「アニメーション層」である。その構造はモーションデータを持つノードからなるグラフである。アニメーション層ではモーション・ノード間の遷移が滑らかに行われるように制御される。キャラクターの身体を制御するには、知能レイヤーから身体レイヤーへ「走る」、「剣を振る」、「歩く」、「座る」といった行動を指定するメッセージを出す必要がある。中間層を通して、これらの行動に対応するモーションを再生する。モーションの選択やタイミング制御を行うのが「AnimGraph」である。AnimGraph はステートマシン構造になっており、再生するモーションデータに対応するノード間を遷移する(図 6)。

図 6 FFXV の AnimGraph

意思決定は離散的なシステムであり(ニューラルネットは連続的なシステムであるが、ゲームタイトルで採用された例は今のところ少ない)、アニメーションは連続的なシステムである。意思決定モジュールはある時点で次の行動を決定するが、アニメーション・モジュールはモーションデータ(キャラクターの短い動作のモーションのこと)を持ち、アニメーターによって設定された時間幅の中でモーションを連続的に再生する。意思決定モジュールからアニメーション・モジュールへの指令は、アニメーション・モジュールが先に指定されたモーションを実行中でも発行される。その場合、現在再生中のモーションから次のモーションへ移行せねばならない。一つのモーションから次の

モーションへ移ることは「モーション遷移」と呼ばれる。モーション遷移には、あらかじめ AnimGraph で指定されたモーション間の「ブレンドの仕方」が設定されており、再生中のモーションは徐々に減衰されつつ、そのモーションが終わりきる前に、次に指定されるモーションの再生が始まり、二つのモーションが定義されたブレンド率で加算されながら再生される。この結果、モーションから滑らかに次のモーションに接続されるモーション再生となる。

基本的に、知能レイヤーは AI Graph で記述される。AI Graph のうち、最終的なアウトプットとして、アニメーション層のモーションデータを直接指定することで、そのモーションが再生される。現在再生中のモーションとの兼ね合いは、アニメーション・レイヤーの中でブレンド処理によって滑らかに接続される。しかし、この実装では、アニメーション・レイヤーのモーション ID の変更やデータの拡張に対して、そのモーションを参照したすべての AI Graph を変更する必要がある。それを避けるために、ボディ・レイヤーによってアニメーション・レイヤーを抽象化しておく AI Graph のノードからは身体レイヤーのステートを指定する。身体レイヤーのステートは対応するモーション ID を所持しており、アニメーション・レイヤーはそこから指定されたモーションを再生する。モーション ID やモーションの拡張があった場合でも、ステートのみを変更すれば良い。

このような階層の独立性とその接続は、開発チームの独立性にも対応している。アニメーション・チームはアニメーション・レイヤー、AI 開発チームは知能レイヤーを作成する。もし知能レイヤーの AI Graph を作成する時に、アニメーション・レイヤーのデータ構造を知り、AI Graph のノード群とアニメーション・レイヤーのノード群の間に直接的なリンクを張る必要があるとすると、知能レイヤーの製作者はアニメーション・レイヤーの状態と仕様を常に追いつつ作成せねばならない。またアニメーション・レイヤーのデータは当然、開発中に変更され続けるのであるから、その変更のたびに AI Graph を変更する必要に迫られる。また逆も同様であり、アニメーション・レイヤーでアニメーション・グラフのノードを結ぶときに、AI Graph のノードからいかに呼び出されるかを想定して記述するには、AI Graph の仕様を知らねばならない。そこで、解決策としては3つ考えられる。

- 知能レイヤーの担当者たちがアニメーション・レイヤーの仕様を理解しながら、AI Graph を書く。
- アニメーション・レイヤーの担当者たちが、知能レイヤーの仕様を理解し、アニメーション・グラフを書く。
- 知能レイヤーとアニメーション・レイヤーが直接

関係しないように、中間レイヤーを作る。

AI チーム、アニメーション・チームの討論の結果、中央に「ボディ・レイヤー」を敷くことで、お互いが独立に開発されつつ、中央のボディ・レイヤーによって結ばれる構造となった。お互いのデータの直接的なリンクをしないことで、お互いの仕様を理解せず、それぞれの層を形成していくことが可能となる。この構造はお互いの仕事への干渉を最小限にしつつ連携することを可能にした。

4.1 エージェント・アーキテクチャ

センサーとエフェクターを持ち知能全体を定義するアーキテクチャは一般に「エージェント・アーキテクチャ」と呼ばれる。デジタルゲームにおけるキャラクターには 2000 年以降に導入された[Isla 01, Isla 02, Orkin 05a, Orkin 05b]。知能・身体まで含めたキャラクターを作るアーキテクチャである(図 7)。まず知能と環境を明確に分離し、知能は「身体」、「記憶」モジュールを内部に持ち、「センサー」モジュール群が環境から情報を知能へ向けて取得し、認識モジュールはそこから「認識」を形成し、記憶モジュールに書き込む。そしてそこから「意思決定」モジュールが行動方針を決定し、「身体運動生成」モジュールに伝え、伝えられた「身体運動生成」モジュールが身体運動を作り出す[三宅 08b]。

思考モジュールと記憶モジュールを分離して設計するアーキテクチャを「C4 アーキテクチャ」という。2000 年に MIT の Synthetic Creatures Group[MIT 03]によって提案され、ゲーム産業で広く採用されている[Isla 01]。FFXV のキャラクターもエージェント・アーキテクチャに沿って設計された[三宅 16a]。

エンターテインメントの目的は、純粋なキャラクターのシミュレーションではなく、キャラクターの挙動を用いてユーザーに様々な体験を与えることで、楽しませることが目的である。そこで、ユーザーの挙動に対する反応性が重要になる。たとえば、敵キャラクターに慎重に近づくという体験を質高く形成するには、敵キャラクターのセンサー、思考、挙動が感度高く実装されていなければならない。ゲームが実現しようとする体験に応じて、エージェント・アーキテクチャの詳細が決定される。

4.2 センサー

キャラクターのセンサーのうち視覚、聴覚、他のキャラクターにぶつかられた衝突力検出の 3 つは、アクションゲームの基本として実装される。他の感覚についてはゲームタイトルがノンプレイヤーキャラクターに求める役割を果たすために必要な五感の性能に応じて実装される。特に重要なのは視覚である。また

ステルスゲーム(潜入ゲーム)では視覚と聴覚である。通常、視覚はそのキャラクターを中心とした扇状の視覚領域を設定し、その中に入った他のキャラクターやオブジェクトに対して、(だいたい)目の位置から、あるいは上空から見た二次元のキャラクターの中心座標からレイキャスト(光線を引くこと)することで視覚を判定する。間に障害物がなくレイが通れば「視覚が通った」と見なす[Walsh 14]。

図 7 エージェント・アーキテクチャ

ノンプレイヤーキャラクターのセンサーは、いかにプレイヤーに対して反応できるか、という点と、そのノンプレイヤーキャラクターがゲーム内の環境世界に生きているか、というリアリティを演出するために必要である。前者に対しては、「敵キャラクターがプレイヤーキャラクターに気づく」事象が最もゲームプレイ経験に結びつく事象である。この気づくまでの時間は早ければ良い、というものではなく、反応速度に個体差を出すことで、さまざまなモンスターごとに認知の多様性を出すことができる。また、後ろから近づいた場合と、前から近づいた場合とで、気づくまでの時間が異なるなど、方向の差異化を行うことで、ゲームプレイに多様性をもたらすことができる。

FFXV のモンスターでは 2 種類の扇状センサーが設定されている。一つは前方に展開した中心視野としての扇状センサー領域で、もう一つはやや後方にも広がった周辺視野としての扇状センサー領域である。それぞれの領域で取得できる情報の種類が異なり、対象発見までの時間が領域によって異なる。中心視野の方が認知するまでの時間が早い処理となっている(図 8)。後方の視野は、ぼんやりとした認知であり、明確にターゲットとなるまでに中心視野よりも時間がかかるように差異化している。

図 8 キャラクターの扇状センサー

このようなセンサーはまた認識した敵の優先度を付ける「ターゲティング・システム」と密接な関係にあり、さまざまな優先度付けの方法が準備されており、各キャラクターにその一つの方法が設定されている。

4.3 意思決定

意思決定は、人工知能の中心である。現在のゲームで最もよく使われる意思決定アルゴリズムは 8 つあり、いずれか一つを用いるか、これらを組み合わせることによって構築する[三宅 12a, 三宅 12b].

(反射型)

ルールベース (Rule-based)

ステートベース (State-based) [Gregory 09a,b]

ビヘイビアベース (Behavior-based) [Isla 05a,Isla 05b]

(非反射型)

ユーティリティベース (Utility-based) [並木 11]

ゴールベース (Goal-based) [三宅 06, 三宅 08b, Orkin 03]

タスクベース (Task-based) [Nau 03, Straatman 13]

シミュレーションベース (Simulation-based) [岡村 11a, b]

ケースベース (Case-based) [Hayles 15]

意思決定アルゴリズムにおける「～ベース」という言葉は、「～」を意思決定の構成する要素として、選択する、という意味である。

4.3.1 ステートマシンとビヘイビアツリー

ステートベースとビヘイビアベースについて解説する。ステートベースと言え、ば、「ステート」(状態)を要素として意思決定システムを組むことである。ステートにはキャラクターの取るべき行動が記述されている。ステートから次のステートへ遷移する条件を設定した、ステート間を結ぶグラフ構造は「ステートマシン」呼ばれる。以上のように、ステートマシンはゲーム状況を状態に分割し、分割したそれぞれの状態に行動を割り当てるスタイルである (図 9)。

図 9 階層型ステートマシン

ビヘイビアベースは、キャラクターの実際のビヘイビア (身体的運動) をベースとする意思決定システムである。ステートマシンがキャラクターの状態を捉えることに長けているが、ビヘイビアベースは、キャラクターの身体動作のシークエンスを実現することを指向した意思決定手法である。キャラクターの抽象的な行動から具体的な行動、そして、身体的行動へ向かってリダクションしていく手法である。たとえば、「バトルする」から「攻撃する」、「隠れる」、さらに「攻撃する」は「弓を放つ」、「攻撃魔法を打つ」、「剣を振る」というビヘイビアに分解されて行くように、常にキャラクターを主体とした行動を指向してノードが記述される。

具体的には、ビヘイビアツリーという形式が最も良く用いられる (図 10)。ビヘイビアツリーは以下に詳細を記述するが、大きく捉えると末端がキャラクターの具体的なビヘイビアを指定するノードとなっており、このノードを随時、環境の変化に合わせて指定する。以下に詳細を記述する。

ビヘイビアツリーは『Halo2』(2004, Bungie) において Damian Isla 氏によって発案された手法であり、ルートから末端のビヘイビアに向かって行動を選択していく階層型ツリーの構造である (図 9) [Isla 05a]. 末端のノードがすべてビヘイビアであり、ビヘイビアをルートから中間層のノードが指定することで、連続的にビヘイビアが実行され、キャラクターに滑らかな連続運動を実現する。

図 10 ビヘイビアツリー

ビヘイビアツリーの構造は、層状の階層構造からなる。ルートの方を上位、末端の方は下位という。上位から下位へ向かって処理の流れがある。一つの層には複数のノードが含まれており、それぞれのノードが下位の層を持つ。ただし、末端はそれ以上の層を持たない。末端のノードはビヘイビア・ノードと呼ばれ、キャラクターのビヘイビアを指定するためのノードである。通常、この末端のノードはモーションデータを指定する。

各層に一つの選択ルール (Selection Rule) が指定されている。この選択ルールに従って、層の中で実行すべき複数のノードとその順番が決定される。グラフ上で、それぞれの層の中では、優先度の高いノードから順番に並べて記述する、という決まりがある。ただ、この選択ルールが実行される前に、各ノードが現在、実行かどうかの判定が入り、実行可能でなければ候補から外れることになる。たとえば、「魔法」はそれを放つのに必要なマジックポイントがない場合は候補から外れることになる。

実行時は、ルートから末端へ向かって選択がくり返され、末端へたどり着いて初めてビヘイビアが実行される。末端で実行が終わるとルートに戻り、再び末端へ向かった選択がくり返される。各層 (ノードの集合) は子ノード同士が競合するモデルであり、現在実行可能なノードの中で「シーケンス・ルール」はあらかじめ定められた順番で実行し、「プライオリティ・ルール」はあらかじめ定められた優先度に従って現在実行可能なノードの中で最も高い優先度を持つ実行可能なノードを実行し、「ランダムルール」はランダムにノードを実行する。

例えば、図 10 のビヘイビアツリーでは、最初の層は「プライオリティ」が選択ルールであり、現在敵がいるならば「バトル」が選択される。もし敵がいなければ「バトル」、「防御」は実行不能なので「休憩」が選択される。今「バトル」が選択されたとする。次のバ

トルを持つ層が実行される。この層は「シーケンス」が選択ルールであるから、「攻撃」、「隠れる」が順番に実行される。次に「攻撃」ノードの層が選択ルール「シーケンス」に従って実行され、「弓を放つ」、「攻撃魔法」、「剣を振る」が順番に実行される。このうち「攻撃魔法」は層を持つので、「ランダム」ルールに従って使う魔法が選択される。次に「隠れる」ノードに処理が戻り、このノードが持つ層が「ランダムルール」に従って実行され、たとえば「森に潜む」ノードが実行される。そしてルートへ処理が戻る。

以上のようにビヘイビアツリーは、状況に応じてどのようにビヘイビアを繋げていく意思決定アルゴリズムである。

4.4 AI Graph

ステートマシンは、状況認識が優れており、ビヘイビアツリーは連続した行動の小プラン (一連のビヘイビア) を形成する点に長所がある。この両者の長所を取り入れたステートマシンとビヘイビアツリーの入れ子構造で組み合わせる新しい手法を開発した。

図 11 ハイブリッド型ノードフォーマット

たとえばステートマシンの 1 つのノードの中にビヘイビアツリーやステートマシンがある、そのビヘイビアツリーのノードの中に更にステートマシンがある (図 11)。この入れ子構造によってステートマシンの状態の分割による制御と、ビヘイビアツリーの滑らかな連続動作の記述、双方の長所を取り入れた意思決定表現を獲得することができる。また上層から下層へ向かって問題のスケールを小さくすることができ、階層的に課題を解決するアーキテクチャにもなっている。

この複合型のグラフを「AI Graph」と呼び、GUI (グラフィカルユーザーインターフェース) によって構築するツール「AI Graph Editor」を製作した (図 12) [白神 15]。

4.5 AI Graph の問題点

AI Graph の問題点は、表形式のデータをコンパクトに表現することが難しい点にある。ゲーム開発では、一般的にキャラクターごとの設定表がある。縦軸にモンスターの種類、横軸に各種の条件が書かれている。たとえば、

敵の出現テーブルであれば、横軸には「草地」、「砂浜」、「高山」などの項目があり、出現させる場所にチェックをする。ゲームプログラムはこのテーブルを読み込み、それぞれの場所に合ったモンスターを出現させる。知能レイヤーにおいても、キャラクターごとに AI Graph やパラメータを変化させる表が必要となる。モンスターであれば、中間レイヤーにおいてモンスターごとのパラメータ設定と AI Graph を指定する設定表、仲間キャラクターであれば、末端における状況における行動を指定する設定表が使用された。AI Graph だけでなく、この表形式のデータによって、キャラクターごとの知能のバリエーションが実現された。

4.6 AI Graph Editor

「AI Graph Editor」によってキャラクターの意思決定を作る仕事は、ゲームデザイナーの仕事として確立することが可能となった。マウス操作でグラフを製作し、キーボードから数値を入力することでノードのカスタマイズを行う。これによって開発中の仕様変更に伴う拡張に耐え、数値の調整をゲームデザイナーが独立して行えるようになり、FFXV において AI の量産が可能となった。

AI Graph Editor は階層型グラフシステムを導入し、データを適切な粒度で階層化できるようにすることで、AI データの拡張性（スケーラビリティ）を確保している。また、トレイ（各階層のグラフ）を単位として階層化することで、データを適切な粒度に分割する。

AI Graph Editor のデータはバイナリデータとしてシリアライズ（バイナリ形式によるデータパッケージング）され、ゲーム時にキャラクターの知能として実行される。AI Graph はメモリ上 20MB 程の使用量である。最上層から下位層に向かって実行するトレイが探索され、実行には下位階層がすべて終了して上位層に戻る。適切な粒度で階層化すると、不要な状態遷移の記述を減らし組み合わせ爆発を減らすことが可能となる[三宅 16b]。多い場合は数階層に及ぶグラフが作成された。

図 12 AI Graph Editor

AI の情報と、ゲームの情報を、柔軟かつスムーズに受け渡せる領域として「ブラックボード」を用意する

(図 13) [石田 96, Isla 02, Nii 86a, Nii 86b]. これはそれぞれの階層がローカルブラックボードを持ち、センサーが収集した敵への距離などの情報を、グラフ上で線を結ぶことで高い階層から低い階層へ共有することができる。グローバルブラックボードに置いた情報は、すべての階層から参照することが可能になる。この仕組みによって、それぞれのキャラクターの持つ記憶を管理し、その情報を用いて各ノードのカスタマイズを行う。

キャラクターの持つ記憶としては、ブラックボード上には身体の状態、ゲームシステム情報（HP、アビリティ、装備、ゲームの進行情報など）、仲間の情報（メタ AI からの指示や、パーティの情報）、敵の情報（ターゲットの強さ、ターゲットまでの距離や角度の情報）、ナビ情報（位置検索結果）などが管理される。また逆に、そのトレイにおける思考の結果が書き込まれ、他のエージェントと共有されることもある。

図 13 AI Graph のブラックボード

4.7 アセット化・オーバーライド

AI Graph Editor を用いてキャラクター AI の意思決定部分を記述する。しかし、さまざまなキャラクター AI を作っていく上で気づくことは、それぞれのキャラクターの AI Graph の構造がお互いによく似ているということである。いくつかの AI Graph 同士はパラメータが違うだけであり、いくつかの AI Graph 同士は一部分が違うだけである。そこで、キャラクターの行動を効率良く作成・編集するために、一度作った AI Graph をアセット化（固定した再利用可能なデータとすること）し再利用可能とし、さらにそれらを他の AI Graph から呼び出すことができると同時に、部分だけを書き換えることができるようにする「オーバーライド」機能を実装した（図 14）。たとえば、あるステートマシンをアセット化し、そのアセットを AI Graph から呼び出し、その中の一つのステートを書き換えることができる。

アセット化したトレイの部分的なオーバーライドにより、メイン処理を変える必要なく、各部分を書き換えては試すイテレーション・サイクルの環境が実現した。

また、それぞれのキャラクターの特定部分のグラフだけを作れば全体のグラフを作れることになり、大幅な工程の効率化を実現した。

図 14 AI Graph のオーバーライド

4・8 AI モード

すべてのキャラクターは、「モード」と呼ばれるステートが用意されている。「モード」はどのようなキャラクターでも持つ基本的な行動である。モードは複数あり「ウェイトモード」(待機)「Go To モード」(指定された地点へ向かう)「リード」(先導する)「フォロー」(ついて行く)などである。モード間の遷移は共通に定義されている。このようなモードは、特に外部からキャラクターにゲームシーンに従った行動を行わせるために活用される。これらは 2・1 節で述べた「演技用キャラクターAI」(図 3) に対応する。

また「ノーマルモード」としてデフォルト状態を規定する「通常稼働 AI」があり、これは「自律型キャラクターAI」(図 3) に対応する。ノンプレイヤーキャラクターは自律型エージェントとしての側面と、プレイヤーを楽しませる役者としての側面があり、この両者を AI モードによって実現している。

どのようなキャラクターも「モード」群を含めた基本となる「トレイ」が最上層にある(図 14)。基本的な実装はモードとしてされているため、どのようなキャラクターもこの層だけで基本的な動作をすることが可能となる。各モードをオーバーライドすることによって、キャラクターごとの特徴を出すことができる。この仕組みによって、モード間の遷移は共通であるが、各モード内はキャラクターごとに異なるように作ることが可能となる。

4・9 AI Graph Editor の利用実績

すべてのキャラクターは、AI モードを要素とするステートマシンのアセット化されたトレイからなる(図 14)。「ノーマルモード」にキャラクターの通常思考の AI Graph が記述される。また、それぞれのノンプレイヤーキャラクターに合わせて、各モードをオーバーライドすることによって、キャラクターの特色を効率的に出すことが可能となる。たとえば、待機しているときに「頭をかく」、「きよろきよする」など、オーバーライドによ

って、各キャラクターに行動的特徴を加えることができる。

ノーマルモードにおける「モンスター」、「仲間キャラクター」、「敵兵」において使用された AI Graph は、それぞれのキャラクター種別において、それぞれ使用傾向が異なる結果となった(図 15)。これは、それぞれのキャラクターに求められる役割と知的機能が異なるからである。以下、それぞれについて説明を行う。

キャラクター種別	モンスター	兵士	仲間
AI Graph 要素			
ステートマシン	○	○	○
ビヘイビアツリー	○	○	×
AIモード	○	○	○
階層構造	○ (3階層)	○ (3階層)	○ (15階層)
アセット化	○	○	○
(AIモード以外) オーバーライド	×	○	×
ブラックボード	○	○	○

図 15 AI Graph のキャラクター種別ごとの使用結果

4・9・1 モンスター・キャラクターの AI Graph

モンスターはおおよそ 40 種類あり、また同じ種類でも 3 段階の大きさや様々なバリエーションがある。そこで、基本的な思考は同一でも、そのバリエーションを出す仕組みとして、AI Graph のアセット化が多用された。

知能レイヤーは、全部で 3 階層になっている。第 1 層は、AI Graph 上のステートマシンである。これが全モンスター共通である。第 2 階層は AI Graph を呼び出すのではなく、表形式のデータとなっている。ここで、各モンスターに応じたパラメータと、第 3 層で呼び出す AI Graph が指定される。多数のモンスターの管理をこの表形式のデータを用いて行う開発体制である。

第 3 階層として呼び出されるビヘイビアツリーは、すべてアセット化されたビヘイビアツリーであり、第 2 階層の表からアセット化 ID によって呼び出される。

オーバーライド機能はボディ・レイヤーの AI Graph で飛行モンスターと地上モンスターの差異を出すために使われただけで、知能レイヤーの中では使われなかった。

ブラックボードは通常的にどの AI Graph でも活用された。また、モンスターが一斉にプレイヤーに飛び掛かって一方的な戦闘にならないように、ゲーム側から戦闘するモンスター以外、戦闘させないように「クールダウン時間」(戦闘禁止時間)がブラックボードを通じて設定している。この時間が 0 にならないうちは、モンスターはプレイヤーに襲い掛かることができない。

全体として、第 1,3 階層の AI Graph を汎用的に用意し、第 2 階層の表からパラメータと使用する第 3 階層の AI Graph を指定することで、できるだけ既存のアセット化

された AI Graph を再利用し、多数のモンスターの AI Graph を量産する仕組みとなっている。

4・9・2 兵士キャラクターの AI Graph

兵士を作るチームの担当は、「ニフル兵」と呼ばれる機械兵隊と、数 m の中型の搭乗型兵器「魔導アーマー」である。両者とも、知能レイヤーは 3 階層の構造になっている。

ニフル兵も魔導アーマーも、フィールドや基地などゲーム全域で行動し、また作戦イベント（ゲームシナリオ上発生するさまざまな戦闘イベント）に参加することが多い。そこで最上位層はステートマシンで、作戦や戦闘のタイプなど、対応する状況を状態として分類することで、AI Graph の肥大化を全体が発散することを防いでいる。第 2 階層は、ビヘイビアツリーで、ここで固有の行動ロジックが構成される。第 3 階層はステートマシンで、ボディ・レイヤーに対して身体動作を指定する。

また異なる作戦において、類似した行動を取ることも多く、一度記述した AI Graph を「アセット化」し再利用することで、二重に記述する手間を省いた。オーバーライドを使う機会は極めて少なかった。また、部隊として、味方同士でコミュニケーションを取る必要があり、各キャラクター AI が知能レイヤーの AI Graph において思考した結果をブラックボードに書き込み、上位のチーム AI がそれを読み取り、兵士全体の意思決定状況をリアルタイムに把握しチーム全体の管理を行っている。

4・9・3 仲間キャラクターの AI Graph

仲間キャラクターの知能レイヤーでは、ステートマシンが多用されており、ビヘイビアツリーはほとんど使用されていない。兵士やモンスターは、プレイヤーの前に数秒～数分の間、現れて倒されるか、ゲーム進行と共に画面の外へ消えてしまうのに対して、仲間キャラクターは継続的にプレイヤーの周囲で活動し続ける。そこで、ステートで継続的に状態を管理する方針のもと、ステートマシンのみで 15 階層の AI Graph が構築された。これは仲間キャラクターが対応すべき状態の多さを反映している。

仲間キャラクターは通常三人であるが、それぞれ個別の AI Graph ではなく、同一の AI Graph のアセットを使用して、ステートの分岐でキャラクターごとに個別化されている。

最上位階層では、AI モードごとに状態が管理されている。またその下の階層では、「身体動作」を管理するステートと、「顔の向きや台詞など」意識的な動作を管理するステートが並行的に動作するように作られている。身体動作をブラックボードに書き込みながら、意識的動作ステートから身体状態を参照し動作を決定する形となっている。たとえば、「走る」、「歩く」、「剣を振る」などが身体状態であり、その時の表情や台詞は意識

的動作のステートであり、身体状態を考慮した意識的動作が決定される。

末端の動作選択は表データが使用されており、各行動のトリガー条件と優先度フラグが設定されている。AI Graph の末端では、この表が呼び出され、その時の状態の条件に適合する行動を選び出す。表データが用いられたのは、キャラクターが取りうる挙動の数が十数個と多いために、AI Graph で書くには適していないからである。たとえば、「走る」にしても、いくつかのバリエーションがあり、体力や状況によってモーションを変える必要があり、その選択は表データを参照することで決定される。

4・10 AI Graph 結果と考察

AI Graph の導入によって、キャラクターの意思決定システムは AI Graph Tool 上の製作に集約され、すべてのキャラクターの意思決定システムは、統一的なフォーマットで記述されることになった。AI Graph によるキャラクター AI の製作過程において、自律型エージェントとしての思考と、ステージごとにゲームシナリオから要求されるキャラクターとしての行動を使い分ける必要が生まれ、「AI モード」が導入された。どのようなキャラクター AI を作る場合も、まず共通の「AI モード」のステートマシンのトレイを準備し、各モードをオーバーライドによってカスタマイズしていく手順が設定された。

「AI モード」をステートとするステートマシンを、すべての AI Graph の基本とすることで、AI Graph の製作者は、各モードの AI Graph の設計に集中できるようになった。また、モードごとの AI Graph が実装されることで、再利用性が高くなった。AI モードの中でも「ノーマルモード」は自律的に行動するモードであるが、キャラクターに求められる行動のタイプに応じて、AI Graph の組み方（ビヘイビアツリー、ステートマシンの組み合わせ方）や技術的要素（パス検索、表データの使用）が異なり、用意した技術の組み合わせによる効率的な AI Graph 作成を可能にしたと同時に、人工知能の様々なバリエーションの実現を可能にした。

どのキャラクターも最上位層はステートマシンが使用された。全体として、階層型ステートマシンで状態数を発散しないように抑えて、末端の層は多様な身体的行動が取れるようにビヘイビアツリー、あるいは表データが用いられた。この開発基盤の上で AI Graph に慣れていない開発者でも、迅速に AI Graph の作業に入ることが可能となった。

4・10・1 ビヘイビアツリーとステートマシンの効果

ビヘイビアツリーとステートマシンを組み合わせた AI Graph は、キャラクターの思考のデータに統一的なフォーマットを与え可読性を高めた。また、グラフ製作を AI Graph Tool 上で可能にしたことで、エンジニア以外のゲ

ームデザイナー、プランナーに AI の製作に参加する体制を可能とした。また AI Graph に起因するバグは、AI Graph Tool から生み出されたバグとして、その再発を防ぐために AI Graph Tool 上に制約が加えられる形でフィードバックされ、開発の進行と共に安定性を獲得していった。

4・10・2 表, および AI モードの効果

また末端のノードにおいて、キャラクターや場所依存の分岐処理は、表データの活用によって集中的に管理され、AI Graph の記載量は大きく削減された。

AI モードの導入は、全キャラクターの AI Graph の基本部分に統一的な形式を与え、AI Graph に一定の品質を担保すると同時に、その拡張によって思考のバリエーションを実装しやすい基礎を与えた。

これらの開発効率化効果は、AI Graph をコンパクトにすると同時に、その品質と安定性を高め、ゲーム全体の品質の向上に貢献した。

5. ナビゲーション AI

ナビゲーション AI は地形を認識する人工知能である。この人工知能は、キャラクターAI やメタ AI が地形の認識を必要とする時に呼び出されることで機能する。広大で複雑な地形の認識はメモリ負荷・処理負荷の大きな過程であり、ナビゲーション AI は巨大な人工知能になってしまうことが多い。また環境情報を客観的かつ正確に解析することで、キャラクターを環境の中で正確な振る舞わせる能力を持つ[Gravot 12a, Gravot 12b]。さらにリアルタイムの地形の変化にも動的な対応する必要がある。このような理由により、ナビゲーション AI は、自律的な人工知能として構成され、独立に動作しながらもキャラクターAI、メタ AI をサポートする役割を持つ。

レベルデザインの地形全体・空間全体に対する表現は特に世界表現 (WR: World Representation) と呼ばれる [Straatman 05]。世界表現とはマップ全体に関する知識表現 (Knowledge Representation) である [Isla 09]。

90 年代初頭までデジタルゲームの多くはナビゲーション・データを持っていなかった。キャラクターがその上を移動するポイント・データか、固定領域 (たとえば部屋の中) が指定されているだけであった。しかし、90 年代半ばの 3D ゲームの出現のタイミングから、徐々にナビゲーション・データが導入され、その上の任意の 2 点に対して A*アルゴリズムによって動的にパス検索を行う手法[Buckland 04, Snook 01]が急速に広まり、現在では標準的な方法となっている。90 年代後半にはデータの作成のしやすさからポイント群を地形に沿って分布し、ポイントの近接関係をネットワークグラフに落としたウェイポイント法が最も多く使われていた。2000 年初頭頃

からは、ナビゲーション・メッシュ法が広く使われるようになった。パス検索によって、キャラクターはレベル上の任意の二点間を自由に移動する能力を得たのである。ナビゲーション・メッシュは地形を凸多角形で覆ったデータであり、近接メッシュ同士がリンクを持つネットワーク型データである。

5・1 世界表現

ナビゲーション AI は、さらに小さい地形認識のための人工知能モジュールの集合である。地形を把握する方法、特徴抽出の方法はモジュールごとに異なる。

FFXV ではパス検索のための世界表現はナビゲーション・メッシュ (Navigation Mesh) である。ナビゲーション・メッシュはキャラクターの移動可能な領域を、互いに接続された多角形のポリゴンの集合によって表現したデータである (図 16)。このポリゴンをノードとする連結ネットワークグラフ上で出発ノードから到着ノードまで A*アルゴリズムのパス検索アルゴリズムで最小コスト検索を行う。

図 16 FFXV のナビゲーション・メッシュとパス検索

ナビゲーション・メッシュは地形とキャラクターモデルの衝突を司る「地形の衝突モデル」に沿って生成される [Gravot 13]。FFXV では、自動生成アルゴリズムが夜間に作成するように「ナイトリー・ビルド」の形でプログラムが動いており、朝方には各マップの最新のナビゲーション・メッシュが作成される [Gravot 19]。

ナビゲーション・メッシュは、ノンプレイヤーキャラクターの移動の基本となる。ところが、開発中は地形モデル、オブジェクト・モデル、キャラクターモデルなどゲームの要素がほぼ毎日追加・更新される。新しく置かれたオブジェクトにナビゲーション・メッシュが対応するためには、なるべく高い頻度のナビゲーション・メッシュの更新が必要である。開発者が更新したアセットデータは、任意のタイミングで、データサーバーにアップロードされる。それらがいったん終息するのは、開発者の退社後の夜であり、そこで夜のうちにナビゲーション・メッシュと追加・修正されたコンテンツの整合性を取る

ために、ナイトリー・ビルドが行われる。翌朝には、ナビゲーション・メッシュとゲームステージの整合性が取れるようになる。

5.2 パス検索

FFXV では、パス検索モジュールはこのナビゲーション・メッシュ上の A*パス検索アルゴリズムでコストを最小にする経路を計算する。ここで導かれる経路はジグザグであるため、実際はファンネル・アルゴリズムでスムージングを行い、パスを整形する。しかし、近距離 (100m 以内) の場合にはパス検索の前に目的地に向かって 2D レイキャスト (ナビゲーション・メッシュ上のレイキャスト) を行い、パス検索の代替として用いる。

6. まとめ

産業における大型ゲームの開発では、その長期に亘る開発の中で、研究成果を実装し、より差別化され他社から抜きん出た特徴を持たせることが求められる。ゲーム産業では大型ゲームの開発は一般に 3~7 年の期間を要する。その期間には研究を積み上げる期間と、それをゲームへ実装していく期間がある。「AI Graph」, 「メタ AI」は基礎的な研究の上にゲームタイトルに導入した技術であり、それ以外は開発を進めながら開発された (図 17)。

ビヘイビアツリーとステートマシンを融合した「AI Graph」は本開発の要となる技術であり、この技術の開発によってキャラクター AI に多様な知能の形を、拡張性を保って開発し続けることができた。またメタ AI は自律型 AI を持つキャラクターの行動をゲームの流れに沿った行動に調整する役割を果たした。ナビゲーション AI は多様な地形のバリエーションを吸収し汎用的な移動を可能にした。本論文では、この 3 つの AI の連携をモデル化し、FFXV への実装を示し、その効果を実証した。今後のゲーム AI 全体の礎となるものである。

人工知能技術	どんな問題を解決するか?
メタ AI	ゲームの流れを調整する
ナビゲーション AI	地形を認識する
ナビゲーション・メッシュ	大局的な経路を検索する
スマートウェイポイント	局所的な経路を設定する
PQS (Point Query System)	戦術的な位置をリアルタイムで発見する
キャラクター AI	キャラクターの頭脳を作る。
エージェントアーキテクチャ	キャラクターの知能全体の枠組み
意思決定 (ステートマシン)	ステートを基本とする意思決定の仕組み
意思決定 (ビヘイビアツリー)	ビヘイビアを基本とする意思決定の仕組み
意思決定 (AI Graph)	上記二つのハイブリッド型意思決定システム
アニメーション・グラフ	アニメーションの遷移を定義する
ボディ・レイヤー	身体の状態をステートで制御・管理する
モーション・シミュレーションによる学習	シミュレーションによってモーションの特徴を抽出する

図 17 FFXV の基本技術リスト

◇ 参考文献 ◇

- [Booth 09a] Booth, M. : Replayable Cooperative Game Design: Left 4 Dead, GDC (2009)
<http://www.valvesoftware.com/company/publications.html>
- [Booth 09b] Booth, M. : The AI Systems of Left 4 Dead, AIIDE (2009)
<http://www.valvesoftware.com/company/publications.html>
- [Buckland 04] Buckland, M. : Practical Path Planning, Chapter 8, Programming Game AI by Example, Wordware (2004)
- [Gregory 09a] Gregory, J. : Game Engine Architecture, A K Peters/CRC Press (2009)
- [Gregory 09b] Gregory, J. : State-Based Scripting in Uncharted 2: Among Thieves, GDC (2009) ,
<http://www.gdcvault.com/play/1730/State-Based-Scripting-in-UNCHARTED>
- [Griesemer 02] Griesemer, J. : The Illusion of Intelligence: The Integration of AI and Level Design in Halo, GDC 2002 Proceedings Archive (2002) ,
<http://halo.bungie.net/inside/publications.aspx>
- [Gravot 12a] Gravot, F., Yokoyama, T., and Miyake, Y. : パス移動 (動画) (2012) ,
https://www.youtube.com/watch?v=9_vdpndn4jI
- [Gravot 12b] Gravot, F., Yokoyama, T., and Miyake, Y. : 落下移動 (動画) (2012) ,
<https://www.youtube.com/watch?v=o3ihN2kSzMs>
- [Gravot 13] Gravot, F., Yokoyama, T., and Miyake, Y. : Precomputed Pathfinding for Large and Detailed Worlds on MMO Servers, GAME AI PRO, Chapter 20, pp.269-287 (2013)
- [Gravot 19] Gravot, F. : ナビゲーションシステム, FINAL FANTASY XV の人工知能, Chapter 3, ボーンデジタル社 (2019)
- [Hayles 15] Hayles, B. : Case-based Reasoning for Player Behavior Cloning in Killer Instinct, nucl.ai Conference 2015 (2015),
<https://archives.nucl.ai/recording/case-based-reasoning-for-player-behavior-cloning-in-killer-instinct/>
- [Imamura 16] Imamura, N. et al. Final Fantasy XV: Pulse and Traction of Characters, Proceeding siggraph '16 acm siggraph 2016 Talks Article No. 4 (2016)
- [石田 96] 石田 亨, 桑原 和宏, 片桐 恭弘 : 黑板モデル, 分散人工知能, 第 4 章, コロナ社 (1996)
- [Isla 01] Isla, D., R. Burke, M. Downie, and B. Blumberg. : A Layered Brain Architecture for Synthetic Creatures, Proceedings of IJCAI (2001)
- [Isla 02] Isla, D. and Blumberg, B. : Blackboard Architectures, AI Game Programming Wisdom, Vol.1, No.7.1, pp.333-344 (2002)
- [Isla 05a] Isla, D. : Managing Complexity in the Halo2 AI, Game Developer's Conference Proceedings (2005) ,
http://www.gamasutra.com/view/feature/130663/gdc_2005_proceeding_handling_.php
- [Isla 05b] Isla, D. : Dude, Where's my Warthog? From Pathfinding to General Spatial Competence, AIIDE (2005) .
<http://naimadgames.com/publications.html>
- [Isla 09] Isla, D. and Gorniak, P. : Beyond Behavior: An Introduction to Knowledge Representation, AI Summit, GDC (2009) ,

- <http://naimadgames.com/publications.html>
- [岩谷 05a] 岩谷徹：パックマンのゲーム学入門, pp.101, エンターブレイン (2005)
- [岩谷 05b] 岩谷徹：セルフゲームコントロールシステム, 「パックマン」岩谷氏, 「Rez」水口氏ら4人のクリエイターが語る世界のゲームデザイン論「International Game Designers Panel」(講演記事) (2005)
http://game.watch.impress.co.jp/docs/20050312/gdc_int.htm
- [上段 16] 上段 達弘, 下川 和也, 高橋 光佑, 並木 幸介：FINAL FANTASY XV におけるレベルメタ AI 制御システム, CEDEC 2016 (2016)
https://cedil.cesa.or.jp/cedil_sessions/view/1544
- [MIT 03] MIT Synthetic Character Group (2003), <http://characters.media.mit.edu/>
- [三宅 06] 三宅陽一郎：クロムハウন্ズにおける人工知能開発から見るゲーム AI の展望, CEDEC (2006)
<http://cedil.cesa.or.jp/session/detail/50>
- [三宅 08a] 三宅陽一郎：デジタルゲームにおける人工知能技術の応用, 人工知能学会誌, Vol. 23, No. 1, pp. 44-51 (2008)
- [三宅 08b] 三宅 陽一郎, 横山 貴規, 北崎 雄之：エージェント・アーキテクチャに基づくキャラクターAI の実装, 第4回デジタルコンテンツシンポジウム予稿集, 2-2 (2008)
- [三宅 12a] 三宅陽一郎：はじめてのゲーム AI ～意思を持つかのように行動するしくみ～, WEB+DB PRESS, Vol. 68, pp.87-120, 技術評論社 (2012)
- [三宅 12b] 三宅陽一郎：次世代デジタルゲームにおける人工知能の研究課題について, ゲームプログラミングワークショップ2012 論文集, Vol. 2012, No. 6, pp. 108-113 (2012)
- [三宅 15] 三宅陽一郎：デ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)
- [三宅 16a] 三宅陽一郎：AI 最前線の現場から【スクウェア・エニックス】デジタルゲームのための人工知能入門, Think IT (2016)
<https://thinkit.co.jp/article/10010>
- [三宅 16b] 三宅陽一郎 et al.：AI 最前線の現場から【スクウェア・エニックス】ゲーム・キャラクターはどのように意志決定するのか, Think IT (2016)
<https://thinkit.co.jp/article/10012>
- [三宅 16c] 三宅陽一郎 et al.：AI 最前線の現場から【スクウェア・エニックス】キャラクターの身体を作る, Think IT (2016). <https://thinkit.co.jp/article/10015>
- [三宅 16d] 三宅陽一郎 et al. AI:最前線の現場から【スクウェア・エニックス】ゲームそのものを認識するメタ AI, Think IT (2016). <https://thinkit.co.jp/article/10016>
- [三宅 17] 三宅陽一郎 et al. 大規模ゲームにおける人工知能, 人工知能学会誌, Vol.32, No.2, pp.197-213 (2017)
- [並木 11] 並木幸介：ぽかぽかアイルー村における, アフォーダンス指向の AI 事例, AI に多様な振る舞いをさせる手法, CEDEC (2011)
https://cedil.cesa.or.jp/cedil_sessions/view/697
- [Nau 03] Nau, D. et al.：SHOP2: An HTN Planning System, Journal of Artificial Intelligence Research, Vol.20, pp.379-404 (2003)
- [Nii 86a] Nii, H. P.: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures, AI Magazine, Vol.7 No.2, pp38-53 (1986), <http://www.aaai.org/ojs/index.php/aimagazine/article/view/537>
- [Nii 86b] Nii, H. P.: Blackboard Application Systems, Blackboard Systems and a Knowledge Engineering Perspective, AI Magazine, Vol.7 No.3, pp82-107 (1986)
<http://www.aaai.org/ojs/index.php/aimagazine/article/view/550>
- [岡村 11a] 岡村信幸：ARMORED CORE V の階層型ゴール指向プランニングと機体制御, CEDEC (2011)
<http://cedil.cesa.or.jp/session/detail/591>
- [岡村 11b] 岡村信幸：ARMORED CORE V のパス検索, CEDEC (2011)
<http://cedil.cesa.or.jp/session/detail/593>
- [Orkin 03] Orkin, J.: Applying Goal-Oriented planning for Games, AI Game Programming Wisdom, Vol. 2, No. 3.4, pp. 217-227 (2003)
- [Orkin 05a] Orkin, J.: 3 States & a Plan: The AI of F.E.A.R., Game Developer's Conf. Proc. (2006), <http://web.media.mit.edu/~jorkin/>
- [Orkin 05b] Orkin, J.: Agent architecture considerations for realtime planning in games, AIIDE 2005 (2005), <http://web.media.mit.edu/~jorkin/>
- [Rabin 02] Rabin, S.: AI Game Programming Wisdom, Charles River Media (2002)
- [Rabin 04] Rabin, S.: AI Game Programming Wisdom 2, Charles River Media (2003)
- [Rabin 06] Rabin, S.: AI Game Programming Wisdom 3, Charles River Media (2006)
- [Rabin 08] Rabin, S.: AI Game Programming Wisdom 4, Charles River Media (2008)
- [Rabin 13] Rabin, S.: Game AI PRO, Charles River Media (2013)
- [Rabin 15] Rabin, S.: Game AI PRO 2, Charles River Media (2015)
- [Rabin 17] Rabin, S.: Game AI PRO 3, Charles River Media (2017)
- [白神 15] 白神 陽嗣, 並木 幸介, 横山 貴規, 三宅陽一郎：FINAL FANTASY XV -EPISODE DUSCAE-におけるキャラクターAI の意思決定システム, CEDEC 2015 (2015)
https://cedil.cesa.or.jp/cedil_sessions/view/1437
- [Snook 01] Snook, G.: ナビゲーション メッシュによる 3D 移動とパス発見の単純化, Game Programming Gems, Vol.1, No.3.6, pp.279-294 (2001).
- [Straatman 05] Straatman, R., Beij, A., and Sterren, W.V.D.: Killzone's AI: Dynamic Procedural Combat Tactics" (2005), http://www.cgfaai.com/docs/straatman_remco_killzone_ai.pdf
- [Straatman 13] Straatman, R., Verweij, T., and Champandard, A., Morcus, R., Kleve, H.: Hierarchical AI for Multiplayer Bots in Killzone 3, GAME AI PRO, Chapter 29, pp.377-390 (2013).
- [Walsh 14] Walsh, M.: Modeling AI Perception and Awareness in Splinter Cell: Blacklist, GDC (2014), <http://www.gdcvault.com/play/1020195/Modeling-AI-Perception-and-Awareness>
- [商標について]
図 2, 図 4, 図 12, 図 16, : © 2016-2020 SQUARE ENIX CO., LTD. All Rights Reserved. MAIN CHARACTER DESIGN : TETSUYA NOMURA
図 1, 図 3, 図 5, 図 6, 図 7, 図 8, 図 9, 図 10, 図 11, 図 13, 図 14, 図 15, 図 17 : © 2020 SQUARE ENIX CO., LTD. All Rights Reserved.
All other trademarks are the property of their respective owners.
PlayStation 4 はソニー・インタラクティブエンタテインメント社の商標または登録商標です。Xbox One はマイクロソフト社の商標または登録商標です。ファミリーコンピュータは任天堂社の商標または登録商標です。スーパーファミコンは任天堂社の

商標または登録商標です。その他掲載されている会社名、商品名は、各社の商標または登録商標です。

[担当委員：櫻井 祐子]
2019 年 6 月 12 日 受理

著 者 紹 介

三宅 陽一郎 (正会員)

株式会社スクウェア・エニックス テクノロジー推進部リード AI リサーチャー。
2011 年、株式会社スクウェア・エニックス入社、現職。本学会編集委員、日本デジタルゲーム学会理事、芸術科学会理事、国際ゲーム開発者協会日本ゲーム AI 専門部会代表。