

ゲームメカニクスのデザイン

読書記録 — 第1章～第4章

Chapter 1 Designing Game Mechanics

第1章

ゲームメカニクスの デザイン

ゲームメカニクスは、ゲームの心臓部におけるルールであり、プロセスであり、データである。これらは、どのように遊びが進行し、いつ何が起きて、勝利条件と敗北条件は何なのかといったことを定義している。この章では、5種類のゲームメカニクスを紹介し、より一般的なビデオゲームジャンルにおいて、どのように使われているかを紹介する。また、ゲームデザインのプロセスの間で、どのタイミングで、メカニクスをデザインしプロトタイプすべきかを説明し、3種類のプロトタイピング技法とその長所と短所を紹介する。本章を読み終えるころには、ゲームメカニクスが何のためにあり、そのデザインを考えるにはどうすべきかを、明確に理解していることだろう。

1.1 ## ルールはゲームを定義する

ゲームの定義はさまざまだが、そのほとんどに共通しているのは、「ゲームの本質的なフィーチャー（Feature；特徴、機能）はルールである」ということだ。たとえば、アーネスト・アダマス著 *Fundamentals of Game Design* では、次のようにゲームを定義している。

ゲームとは、架空のリアリティーという文脈の中で、最低1つの任意の大きな目標を達成するために、参加者が**ルールに従って行動すること**で繰り広げられるプレイという行為の一種である。

『ルールズ・オブ・プレイ（上）ゲームデザインの基礎』（*Rules of Play*、山本貴光訳、ソフトバンク クリエイティブ刊）では、著者のケイティ・サレンとエリック・ジーマーマンは次のように述べている。

ゲームとは、プレイヤーが**ルールで決められた人工的な対立**に参加するシステムであり、定量化できる結果が生じる。

第1章 ゲームメカニクスのデザイン

Half-Real, Video Games between Real Rules and Fictional Worlds では、著者のジェスパー・ジュール（Jesper Juul）が次のように述べている。

ゲームとは、可变的かつ定量化できる結果を内包する**ルールベースのシステム**である。結果には対応する評価が存在し、プレイヤーは結果に影響を与えようと努力し、結果に対して感情的なつながりを持つ。また、ゲームの終了結果が別のもの^{†1}に変換されることもある。

（引用中の強調は筆者による。）私たちは、これらの3つの異なる定義を比較したり、どれが一番だと主張するつもりもない。ポイントは、いずれの定義もルールについて言及していることだ。ゲームにおいてルールは、プレイヤーに何ができて、ゲームはどのように反応するかを決めている。

ステートマシンのとしてのゲーム

多くのゲームやゲームコンポーネントは、ステートマシン（state machine；状態機械）であると解釈できる（例：Järvinen 2003；Grünvogel 2005；Björk & Holopainen 2005）。ステートマシンとは仮想的な機械であり、さまざまな状態（state；ステート）のうちの1つをとる。それぞれの状態は、ある状態から別の状態への遷移を制御するルールを持っている。DVD プレイヤーを例に考えてみよう。DVD が再生中のとき、この機械は「再生中」の状態である。一時停止ボタンを押せば、状態は「一時停止中」状態へ変わるだろう。停止ボタンを押せば、DVD プレイヤーは DVD メニューへ戻る——これもまた異なる状態だ。DVD プレイヤーが「再生中」の状態のうちは、「再生」ボタンを押しても何も起こらない。

ゲームは初期状態で開始し、終了状態に達するまでの間、プレイヤーの行動が（そして多くの場合メカニクスも）新しい状態を引き起こす。1人用のビデオゲームの場合は、プレイヤーは勝つか負けるか、あるいはゲームを終了する。ゲームの状態は一般に、プレイヤーの位置、他のプレイヤーの位置、同盟、敵、そして重要なゲームリソースの現在の分布を反映したものだ。ゲームの研究分野では、ゲームをステートマシンに見立てることで、ある状態から別の状態に遷移するのを引き金にしているルールを分析している。一部の有効性が認められた手法を用いれば、コンピュータサイエンティストは限られた数の（有限）状態を持つステートマシンを設計、モデル化、実装できる。しかし、DVD プレイヤーとは対照的に、ゲームは、ドキュメント化するには多すぎるほどの、膨大な数の状態を持つことがあり得る。

有限ステートマシン（finite state machine；有限状態機械）は、シンプルな人工知能を持つノンプレイヤーキャラクターの振る舞いを定義するために実際に使われることがある。戦争ゲームのユニットは、攻撃、防衛、パトロールといった状態を持つことが多い。しかしながら、本書は人工知能に関する本ではないため、ここではこれ以上そのような技法については触れないことにする。ステートマシン理論は本書で扱っているような複雑なメカニクスの類を学ぶには有用ではない。

ゲームは予測できない

ゲームの結果は、スタートの時点で明らかになっているべきではない。ゲームはある程度**予測不可能**であるべきだ。予測可能なゲームは、一般的に言って、あまり面白くない。予測不可能な結果を作り出す簡単な方法は、ダイス（サイコロ）を振ったり、ボードゲームのスピナーを振っ

†1 訳注：お金など

1.1 ルールはゲームを定義する

たりするなどといった、偶然の要素を取り入れることだ。ブラックジャックやブロンダイク（トランプで遊ぶソリティアの一種）のような短いゲームは偶然の要素にほとんど完全に依存している。しかし、長めのゲームにおいては、プレイヤーはそのスキル（技術）と戦略的意思決定が異なる結果を生み出すことを望んでいる。プレイヤーは、自分の決断やゲームプレイのスキルが重要であり、スキルなしでは、すぐにフラストレーションを感じ始める。完全に運任せのゲームは、カジノのためのものだ。それ以外の多くのゲームでは、プレイヤーのスキルがゲームの勝敗に影響する。またそういったゲームでは、プレイ時間が長くなるほどスキルの与える影響が大きくなる。

ゲームやシミュレーションでは、偶然の要素を含むプロセス（たとえば、ダイスを振って出た目の数だけ進めるようなもの）は、**確率性プロセス**と呼ばれている。初期状態から結果を決定することができるプロセスは、**決定性プロセス**と呼ばれる。

メモ

ゲームを予測不可能なものにする方法には、偶然のほかに、もっと洗練されたものが2つある。プレイヤーによる選択と、ゲームのルールによって生み出される複雑なゲームプレイである。

ジャンケンのような単純なゲームでは、プレイヤーの決断はいつでその結果が来るかをわかるため、それを予測することはできない。ルールは特定の選択の味方をしないし、特別な戦略も提案しない。対戦相手の選択を予想したり、逆手に取ろうとしても、他のプレイヤーの選択を制限することなどはほとんどできるものではない。同じようなメカニクスが、古典的なボードゲームであるディプロマシーでプレイヤーがコントロールできるのは、ごく少数の軍隊や艦隊だけだ。戦いでは、単純に最も多くの数のユニットを送り込んだほうが勝利する。しかし、すべてのプレイヤーが自軍の次の動きを他のプレイヤーに内緒にならないよう、互いに元を書いていき、選択を全員と同時に公開して、実行するというユーモアな方式をする。そのため、プレイヤーはソーシャルスキルを駆使して、対戦相手の攻撃先を見つけ出し、同盟国に自軍の攻撃や防御を支援するよう説得しなければならない。

複雑なルールはゲームを（少なくとも6人間に）予測不可能なものにする。複雑なシステムは、通常、多くの相互作用の部分を持っている。個々の部分は単純で、その挙動は理解しやすく、ルールもシンプルかもしれない。しかし、すべてのコンポーネントを一緒にすると、ときに非常に驚くべきことが起きる。チェスの対局は、この効果の古典的な例だ。16個の駒のそれぞれの動きのルールは単純なものだが、この単純なルールが深い複雑性を生む。チェスの戦略を書いた書物の数は膨大である。熟練したプレイヤーは、複数ターンにわたっても多くの駒を移動させ、対戦相手をズナに引き落とそうとする。しかし、すべてのプレイヤーが自軍の次の動きを他のプレイヤーに内緒になるよう元を書いていきますがこの種の事情は、ゲームの現在の状態を読み取り、その戦略的な複雑性を理解する能力が、最も重要なゲームプレイングスキルとなる。

ほとんどのゲームは予測不可能性の3種類の発生源をミックスしている。その3種とは、偶然

第1章 ゲームメカニクスのデザイン

の要素、プレイヤーの選択、複雑なルールだ。プレイヤーによって、これらのテクニックの組み合わせの好みも異なっている。多くの偶然の要素が絡むゲームを好むプレイヤーもいれば、複雑と戦略が鍵となるゲームを好むプレイヤーもいる。3つの選択肢のうち、偶然は実装するのが最も簡単だが、予測不可能性の発生源としては必ずしも最適ではない。一方で、プレイヤーに選択と自由を提供する仕組みは、うまくデザインすることが難しい。本書は、その仕事の助けになることだろう。とりわけ、プレイヤーのために面白い選択肢を作るルールシステムを設計することに、ゲームプレイを担当する人に向けている。第6章「一般的なメカニズム」において、私たちは乱数ジェネレータ（ダイスのソフトウェア版に相当）について扱い、同様に他のいくつかの点でそれを議論するが、私たちはそのような偶然の要素は、メカニクスデザインの中心というよりは、メカニクスデザインのサポート役を提供するものだと感じている。

ルールからメカニクスへ

ビデオゲームのゲームデザインコミュニティでは、一般に、**ゲームルール**より**ゲームメカニクス**という言葉が好まれる。なぜならば、ゲームルールとはプレイヤーが認知しているインストラクション（説明）だと考えられている一方で、ビデオゲームのメカニクスはプレイヤーから隠されており、ソフトウェア上の実装でもプレイヤーに直接的なユーザーインターフェイスを与えていないからだ。ビデオゲームのプレイヤーは、ゲームを開始するとき、ゲームのルールがどのようなものであるかを知っている必要はない。ボードゲームやカードゲームと異なり、ビデオゲームでは、ゲームプレイを通じてルールをプレイヤーに伝えているのである。ルールとメカニクスは、互いに関連する概念ではあるが、メカニクスのほうがより詳細かつ具体的である。たとえば、モノポリーのメカニクスについて説明しようとすれば、モノポリーのメカニクスには全不動産の価格、チャンスカードと共同基金カードのテキストすべてが含まれている。ゲームの進行に関係するすべてが含まれる、と言ってもいいだろう。ゲームプログラマがメカニクスを隠蔽なコードに変えていくことができるように、メカニクスは十分詳細化されている必要がある。つまり、メカニクスは必要な詳細すべてを明確に記述するものである。

最も影響力があり、ゲームの多くの側面に影響を与える、単一の一番重要のみを制御する程度の重要度の低いメカニクスを相互作用するメカニクスを**コアメカニクス**と呼ぶ。たとえば、プラットフォームゲームで重力を実現するメカニクスはコアメカニクスである。このメカニクスは、ゲーム中のほとんどすべての移動体オブジェクトに影響を与え、ジャンプもしくは落下時のダメージを制御する他のメカニクスに作用する。一方、プレイヤーがインベントリからアイテムを移すだけのメカニクスは、コアメカニクスとは言えない。自律的なノンプレイヤーキャラクターの振る舞いを制御する人工知能ルーチンもコアメカニクスとは見なされない。

ビデオゲームのメカニクスの大半は隠されているが、プレイヤーはゲームをプレイしながらプレイを通じてそれを徐々に理解していく。熟練したプレイヤーは、コアメカニクスがどのようなものであるか、何度もゲームの動作を観察することで推測する。彼らは優位に立つために、ゲーム

1.1 ルールはゲームを定義する

のコアメカニクスをどのように利用すべきか学習するのである。コアメカニクスとそうでないものの区別は明確ではない。同じゲームについてさえ、何がコアで何がコアでないのかの解釈は、ゲームデザイナーによって違ったり、ゲーム中のコンテキストによっても異なっている。

メカニクスとメカニズム

ゲームデザイナーはよく、ゲームメカニクスを単数形 (game mechanic) で使う。ゲームメカニクスと言ってもゲームエンジンの修理をする人のことではない^{†2}！彼らがゲームメカニクスを単数形 (game mechanic) で使う場合には、あるゲーム要素を制御する1つのゲームメカニクスを示す。そこで本書では、ゲーム要素またはインタラクションに関係するルールセットに対しては、以後一貫して**メカニズム**という言葉を使用する。メカニズムには複数のルールを含められる。たとえば、横スクロールのプラットフォームゲームにおける移動リフトのメカニズムは、移動リフトの移動速度、敵キャラクターがその上に立てること、移動リフト上の敵キャラクターと一緒に移動すること、移動リフトが他のゲーム要素にぶつかるか、一定距離を移動した後に進行方向が反転すること、などといった決め事を持つ。

メカニクスは媒体に依存しない

ゲームメカニクスは、さまざまな媒体で実現することができる。ボードゲームの場合、メカニクスはゲームの道具一式で実現されている。ゲーム盤、トークン (カウンター)、駒、スピナーなどだ。同じゲームを、ビデオゲームとして発表することもできる。その場合、異なる媒体であるソフトウェアで、同じメカニクスが実現されることになる。

メカニクスは媒体に依存しないことから、ゲームの研究者の多くが、ビデオゲーム、ボードゲーム、さらにフィジカルゲーム (physical game；実際に身体を動かすゲーム) でさえ区別をしていない。ゲーム内のさまざまな要素間の関係は、ゲーム盤上に実現して手で駒を動かす場合も、コンピュータの画面上に実現してソフトウェアが画像を動かす場合も、ほとんど同じである。さまざまな媒体で同じゲームをプレイすることができるだけでなく、ときには1つのゲームが複数の媒体を使用することもある。今日では、ハイブリッドなゲームが次々と登場してきている。簡単なコンピュータを同梱したボードゲーム、遠隔のコンピュータとつながったスマートデバイスを用いて遊ぶフィジカルゲームなどがそうだ。

また、ゲームメカニクスが媒体から独立していることで、ゲームデザイナーが1つのゲームのメカニクスを作った後、複数の異なる媒体でそのゲームを実現することが可能になっている。デザイン作業は一度行われるだけだから、これは開発時間を削減していることになる。

^{†2} 訳注：英語の「mechanic」には修理工とメカニズムの両義があり、一般には自動車などの修理工の意味で使われる。

ハイブリッドなゲームの例

Copenhagen Game Collective が開発したゲーム Johan Sebastian Joust は、ハイブリッドなゲームデザインの優れた例だ。ゲームはスピーカーのみを使用し、画面を使用しない。プレイヤーはオープンエリアに集まり、PlayStation Move コントローラを手にとって、ゲームを行う（図 1.1）。このゲームでは、コントローラを一定以上の速度で動かしたプレイヤーが失格となる。そこで、プレイヤーは他のプレイヤーのコントローラを押しつけて、相手を失格にさせようとする一方で、自分のコントローラを守るために慎重に、そしてゆっくりと行動する。BGM のテンポは、プレイヤーが安全に動くことができる速度を示しており、ときおりスピードアップする。Johan Sebastian Joust は、満足のいくプレイヤー体験を生み出すために、シンプルなコンピュータ実装のメカニクスと肉体的なパフォーマンスを融合したハイブリッドのマルチプレイヤーゲームである^{†3}。

図 1.1 盛り上がる Johan Sebastian Joust のゲームプレイ



図 1 図 1.1 盛り上がる Johan Sebastian Joust のゲームプレイ

写真はヨハン・ビフェル・リンデゴアによりクリエイティブコモンズ（CC BY 3.0）ライセンスの元で無料提供

異なる媒体を使用することは、プロトタイプを作成する際の助けになる。通常、ソフトウェアのプログラミングは、メカニクスをボードゲームのルールとして書きとめることに比べて、はるかに多くの作業を必要とする。コンピュータ上でルールやメカニクスを実装する手間とコストに向かい合う前に、もし同じゲームをボードゲームかフィジカルゲームの形式でプレイできるのであれば、その形式で試してみることをお勧めする。次頁で説明するように、効率的なプロトタイピング技法は、ゲームデザイナーのツールボックスの中でもかなり重要なツールである。

†3 訳注：Johan Sebastian Joust のゲームプレイ動画は公式サイト <http://gutefabrik.com/joust.html> のほか、YouTube などの動画サイトで見ることができる。

1.1 ルールはゲームを定義する

5 種類のメカニクス

メカニクスという用語はゲームの要素間のさまざまな基本的な関係を指してきた。一般にゲームで使用されている 5 種類のメカニクスを紹介する。

- **物理 (physics)**：ゲームメカニクスは、ゲーム世界における物理（運動と力の科学）を定義することがある（それは現実世界の物理学と異なっているかもしれない）。ゲーム中、一般的にキャラクターは移動、ジャンプ、飛び降り、ビークル（vehicle；乗り物）の運転をこなす。ゲーム要素の座標、移動方向、要素と要素の交差判定と衝突判定の演算は、多くのゲームにおいて演算の大部分を占めている。物理は、極めてリアルなファーストパーソンシューターから、Angry Birds のような人気の物理ベースパズルに至るまで、多くの現代ゲームで大きな役割を果たしている。だが、実装が厳格に行われることは稀であり、いわゆる **カートゥーン・フィジックス**（cartoon physics；マンガ物理学）を用いるゲームはニュートン力学を改変しており、キャラクターが空中で向きを変えると、非ニュートン的な挙動が可能になっている（私たちは、タイミングやリズムのチャレンジもゲーム物理の一部であると考えている）。
- **内部経済 (internal economy)**：ゲーム要素の収集、消費、交換といった取引のメカニクスは、ゲームの内部に経済活動を構築する。ゲームの内部経済は、典型的には、リソースであると容易に認識されるアイテム（お金、エネルギー、弾薬など）にかかわるものだ。しかし、ゲームの経済は、具体的なアイテムだけでなく、ヘルス、評判、魔力といった抽象的なものも扱うことができる。ゼルダの伝説シリーズにおける、リンクのハート（彼の生命エネルギーを可視化したもの）も内部経済の一部である。多くのロールプレイングゲームで、スキルポイントと他の定量化されたアビリティ（ability；能力）も該当する。こういったゲームは非常に複雑な内部経済を備えている。
- **進行型のメカニズム (progression mechanism)**：多くのゲームでは、レベルデザインによって、プレイヤーがゲーム世界をどう進んでいくかが決められている。ゲームにおける通例として、プレイヤーの分身となるキャラクターは、誰かを救ったり、悪者の親玉を倒してレベル^{†4} をクリアしたりするために、特定の場所へたどり着く必要がある。この種のゲームでは、プレイヤーの進行状況は、特定のエリアへの立ち入りをブロックしたりアンロック（unlock；解除）したりする数多くのメカニクスによって厳重にコントロールされている。レバー、スイッチ、特定のドアを破壊できる魔法の剣などが、このような進行型のメカニズムの典型的な例である。

† 4 訳注：「レベル」とは、ゲーム中のステージ、マップ、ゾーン、ワールドなどを指す用語。「レベルデザイン」は空間／背景レイアウトだけでなく、その空間で起こる遊びを設計することまでを含む。広範な要素を指すため、本書では「ステージ」「ステージデザイン」ではなく、「レベル」「レベルデザイン」として訳出している。

- **戦術的な操作** (tactical maneuvering)：ゲームは、攻撃や守備の優位性のために、マップ上のゲームユニットに配置を行うメカニクスを持つことができる。戦術的な操作は、多くのストラテジーゲームで非常に重要なものだが、一部のロールプレイングゲームやシミュレーションゲームにおいても重要である。戦術的な操作を制御するメカニクスは、場所や地形が各ユニットタイプに与える戦略的優位性を決定する。多くのゲームでは、チェスに代表される古典的なボードゲームのように、タイル(マス目)でユニットの座標を制限している。コンピュータで遊べる最新のストラテジーゲームでさえ、精密なビジュアルレイヤーの背後にうまく隠しているものの、タイルを実装しているものは多い。戦術的な操作は、チェスや碁といった多くのボードゲームだけでなく、StarCraft やコマンド&コンカー Red Alert のようなコンピュータストラテジーゲームなどでも見ることができる。
- **社会的インタラクション** (social interaction)：ビデオゲームは最近まで、プレイヤー間の社会的インタラクションにほとんど関与してこず、せいぜい共謀を禁止したり、特定の情報を秘密にしておくよう求める程度だった。しかし最近のオンラインゲームでは、友人にプレゼントを贈ったり、ゲームへ招待したり、その他の方法で交流を持ったりすることに対してリワード(reward)†☒を提供するメカニクスを備えたものが多い。また、ロールプレイングゲームではキャラクターの演技を制御するルールを、ストラテジーゲームではプレイヤー間の同盟締結／破棄を制御するルールを持っている場合もある。ボードゲームや、子どもたちが遊んできた昔ながらのゲームは、プレイヤー同士の相互作用を促すゲームメカニズムに関して非常に長い歴史を持っている。

メカニクスとゲームのジャンル

ゲーム産業は、ゲームが提供するゲームプレイの種類に基づいて、ゲームをジャンル分けしている。主に経済からゲームプレイを導き出しているゲームもあれば、物理、レベル進行、戦術的な操作、ソーシャルダイナミクス (social dynamics) から導き出しているゲームもある。ゲームプレイはメカニクスによって生み出されるため、ゲームジャンルはゲームが実装するルールの種類に大きな影響を持つことになる。表 1.1 は、典型的なゲームの分類体系と、これらのジャンルとそのゲームプレイがどのように各種のメカニクスと関係しているかを示している。表内のゲームのジャンルは、*Fundamentals of Game Design* の第二版による分類であり、本書で紹介した5種類のゲームルールや構造との関係を記載した。枠線の太さはルールの重要度を示しており、そのジャンルのゲームで一般的に重要であるルールほど線が太くなっている。

† 5 訳注：報酬や褒美のこと。ゲーム内のアイテムやゴールドはもちろんリワードだが、強敵が出没する険しい山道を登り切ると美しい景色が見られる、といったものもリワードの範疇に入る。本書ではそのまま「リワード」と訳出している。

表 1.1 メカニクスとゲームのジャンル

1.1 ルールはゲームを定義する

表 1.1 メカニクスとゲームのジャンル

	物理 (フィジックス)	内部経済	進行	戦術的な操作	社会的 インタラクション
アクション	移動、射撃、ジャンプ等の精密な物理	パワーアップ、アイテム、ポイント、ライフ	難易度が徐々に上昇するよう事前設計されたレベル。プレイヤーにゴールを設定するストーリーライン		
ストラテジー	移動や戦闘用の単純な物理	ユニット構築、リソース収集、ユニットのアップグレード、戦闘におけるユニット喪失のリスク	新しいチャレンジの一式を仕掛けるシナリオ	攻撃用・防衛用のユニット配備	プレイヤー間の協調アクション、同盟、競争
ロールプレイング	移動や衝突を解決する比較的単純な物理。ターン制が多い	キャラクターまたは部隊をカスタマイズする装備や経験値	プレイヤーに目的とゴールを与えるストーリーライン	パーティ戦術	プレイアクティング(役を演じる)
スポーツ	精密なシミュレーション	チームマネジメント	シーズン、大会、トーナメント	チーム戦術	
ビークル(カー)シミュレーション	精密なシミュレーション	ミッションとミッションの間で車両をチューニング	ミッション、レース、競技、大会、トーナメント		
マネジメントシミュレーション		リソース管理。経済の構築	新しいチャレンジの一式を仕掛けるシナリオ	リソース管理。経済の構築	プレイヤー間の協調アクション、同盟、競争
アドベンチャー		プレイヤーのインベントリを管理	ゲームを主導するストーリー。プレイヤーの進行を制御するロック&キー		
パズル	単純な物理(非現実的で離散的な場合が多い)によってチャレンジ(挑戦)を生成		もつと難しいチャレンジを徐々に提供する短いレベル		
ソーシャルゲーム		リソース収集とユニット構築。リソースを払ってコンテンツをパーソナライズする	プレイヤーに目的とゴールを与えるクエストとチャレンジ		プレイヤー間のゲーム内リソース交換。プレイヤーの協力/対立を奨励するメカニクス

9

図 2 表 1.1 メカニクスとゲームジャンルの関係

1.2 ## 離散的メカニクス vs 継続的メカニクス

私たちは5種類のメカニクスをリストアップしたが、これに加えてもうひとつ重要な区分がある。メカニクスには、**離散的メカニクス**と**継続的メカニクス**がある。現代のゲームは、プレイのスムーズかつ継続的なフローを作り出す緻密なメカニクスを用いて、物理（タイミングとリズムを含む）をシミュレートする傾向がある。ゲームオブジェクトを0.5ピクセル左か右に寄せることができ、それがジャンプの結果に大きな影響を持つ場合もある。最大の精度を得るために、物理的な挙動は、高精度の小数値を用いて計算される必要がある。それがここで言う継続的メカニクスである。対照的に、内部経済のルールは離散的なものにある。内部経済では、ゲーム要素とアクションは、段階的な遷移が許されない有限集合に属していることが多い。つまり、ゲーム中に、パワーアップを半分だけ拾うことは、普通はできない。これが離散的メカニクスである。ゲーム物理とゲーム経済の間のこの違いは、ゲームの媒体に依存したゲームレベル、プレイヤーのインタラクションの性質、さらにはゲームデザイナーのイノベーションへの機会にさえ影響を与えている。

物理のメカニクスを理解する

正確な物理計算は、特にリアルタイムにおいて、大量の高速な演算処理を必要とする。このことは、物理ベースのゲームはコンピュータ上で実現しなければならないことを意味する向きがある。スーパーマリオブラザーズのゲームプレイは、移動や足場へのジャンプを必要とするため、このボードゲーム版を作るのは難しい。プラットフォームゲームでは、現実の世界でサッカーをするときのような身体的な器用さが重要になるが、これらのスキルは、ボードゲームでは失われる。スーパーマリオブラザーズは、プレイヤーが実際に走ったり跳ねたりする能力を試すコースとして作り込まれていると考えることもできるだろう。ポイントとなるのは、特定のアイテムを取るとジャンプ力が2倍の状態になるルールは異なる媒体へ移植することができるが、現実にはそんなジャンプは実現できないということだ。ゲームの継続的で物理的なメカニクスは、ゲーム内経済を制御する離散的なルールよりも、多くの演算パワーを必要とする。

興味深いことに、プラットフォームゲームの初期の歴史や、その他のアーケードゲームの初期を振り返ってみると、物理演算は今日のものよりも離散的であった。ドンキーコングにおける動きは、スーパーマリオブラザーズで見られたものより、連続性ではかなり劣っていた。バルダーダッシュの重力シミュレーションは、岩がフレームごとに1タイルずつ一定速度で下に移動するというものであった。ゆっくり遊ぶことになるだろうが、バルダーダッシュのボードゲーム版を作成することは不可能ではない。当時、ゲームの物理メカニクスを生み出していたルールは、他の種類のゲームルールと大きな違いを持っていなかった。初期のゲームのコンピュータは、浮動小数点演算命令をまったく持っていなかったため、ゲーム中の物理は単純でなければなら

1.2 離散的メカニクス vs 継続的メカニクス

なかった。しかし、時代は変わった。今日のプラットフォームゲームにおける物理は、ボードゲームで再現することが不可能か、少なくとも面倒なくらいに正確かつ精密なものに成長した。

物理メカニクスと戦略的なゲームプレイをミックスする

離散的なルールでは、先を見越して動きを計画し、複雑な戦略を立てて実行することが可能である。これは必ずしも簡単ではないが、不可能ではない。実際に、多くのプレイヤーがこれを楽しんでいる。離散的メカニクスとプレイヤー間のインタラクションは、精神的／戦略的なものとなる。プレイヤーがゲーム物理を把握すると、直感的に動きと結果を予測できるようになるが、確信を持つほどではない。そのため、プレイヤーがインタラクションを行う上でスキルと器用さが重要になるのである。結果予測性が違うということはゲームプレイに極めて重要な影響を与える。これについては以下で、物理メカニクスと戦略的ゲームプレイをミックスした2つのゲーム Angry Birds と World of Goo（グーの惑星）を比較しながら見ていく。

Angry Birds では、プレイヤーはカタパルト（投石機）から、豚を守っている防御構造物へ向けて鳥を発射する（図 1.2）。カタパルトはタッチデバイスで操作する。非常に正確な物理シミュレーションが使われているため、発射速度や角度のわずかな違いによって、プレイヤーが引き起こす構造的ダメージがまったく違う結果をもたらすこともある。鳥をカタパルトから発射することは、主に物理的なスキルの事柄である。Angry Birds というゲームの戦略には、このように離散的なルール制御のゲームを表すさまざまな側面が見られる。プレイヤーは、豚の防御を破るために、そのレベルで利用可能な鳥の数と種類を最も効果的に利用する攻撃計画を立てなければならない。これには弱点を特定して攻撃計画を練る必要がある。だが、実行自体はプレイヤーの目と手の協調関係しだいであり、その効果を詳細に予見することは不可能である。

図 1.2 Angry Birds

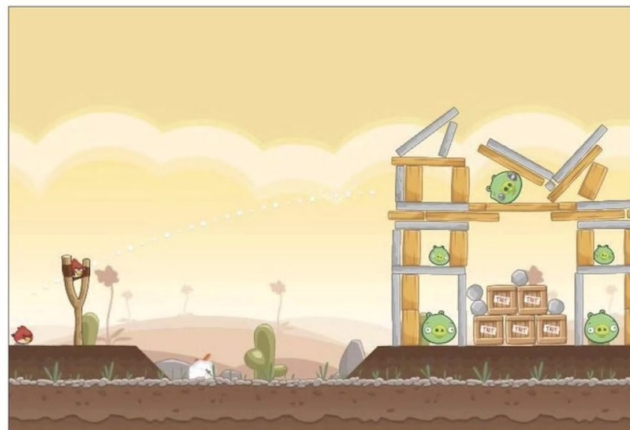


図1.2 Angry Birds

図 3 図 1.2 Angry Birds

1.3 メカニクスとゲームデザインのプロセス

ていの場合、ファーストパーソンシューターの物理を完全に変更する意味はほとんどない。実際、これらのメカニクスを扱うために物理エンジンのミドルウェアを使用するゲームはどんどん増えており、この領域でイノベーションを起こす余地は少なくなっている。他方、ゲームデザイナーは誰しも、他とは違うコンテンツを提供したいと思っている。そこで、多くのファーストパーソンシューターは競合製品とは異なるゲームプレイを生み出すために、独自のパワーアップシステムや、アイテムの収集／消費を管理する独自アイテム経済を備えている。こうした経済を制御するメカニクスには、ゲーム物理と比べて、多くの創造性とイノベーションの余地がある。本書では離散的メカニクスに重点を置いている。

コンピュータゲームの歴史 40 年を振り返ると、他の種類のメカニクスと比べて、ゲーム物理ははるかに速く進化してきたことは明らかだ。ニュートンの法則の明快さと、それをシミュレートする演算パワーの向上があって、物理をデザインすることは比較的簡単になった。しかし経済法則をゲームに組み込むのは、はるかに複雑で骨が折れる。本書の目的は、物理以外の離散的メカニクスをより簡単にゲームに組み入れられるために使える、強固な理論的フレームワークを提供することである。

本書ではゲームプレイのメカニズムについて限定的な側面のみを扱っており、ゲームが内包するさまざまな要素のなかでもとりわけゲームメカニクスに注目している。一部の読者にとっては、ゲームおよびゲームプレイのメカニクスの観点のみを扱っている、と言ったほうが分かりやすいかもしれない。ただこれは、ゲームメカニクスという観点が唯一最上のものであると主張するものではないこともご理解いただきたい。多くのゲームにおいて、アート、ストーリー、サウンド、音楽、その他のフィーチャーがゲームプレイと同じくらい、ときにはそれ以上に、プレイヤーのゲーム体験を高めている。しかし、私たちはゲームメカニクスとゲームの関係を探求するためにこの本を書いたのであり、これに焦点を当てて取り組むつもりだ。

メモ

1.3

メカニクスとゲームデザインのプロセス

ゲームをデザインする方法はゲーム会社の数だけあると言っても過言ではない。*Fundamentals of Game Design* の中で、著者アーネスト・アダムスは、**プレイヤーセントリック・ゲームデザイン**と呼ばれる、プレイヤーが経験することになる役割と経験に集中するデザインアプローチを提唱している。アダムスは**ゲームプレイ**を、ゲームによってプレイヤーに課されたチャレンジ（挑戦）とプレイヤーに許可されたアクションの実行の 2 つで構成されるもの、と定義づけた。メカニクスはゲームプレイを生み出す。マリオが谷をジャンプで越えるとき、レベルデザインはその谷の形状を定義するかもしれない。だが、マリオがとれくらい遠くまでジャンプし、重力がどのように振る舞い、マリオが谷越えに成功するか失敗するかを決定するのは、ゲーム内の物理法則（ゲームの物理メカニクス）である。

第1章 ゲームメカニクスのデザイン

メカニクスがゲームプレイを生み出すわけだから、提供したいゲームプレイの中身が分かっているなら、できるだけ早く、メカニクスのデザインを始めることをお勧めする。本節で概説する開発プロセスは、プレイヤーセントリック・ゲームデザインであり、これは複雑だがバランスのとれたゲームメカニクスを作り出すことに重きを置いている。

ゲームデザインプロセスの概要

大ざっぱに言えば、ゲームの設計プロセスには次の3つの段階がある。コンセプトステージ（concept stage；コンセプト段階）、推敲ステージ（elaboration stage）、チューニングステージ（tuning stage；調整段階）である。これらの段階について以降で説明するが、*Fundamentals of Game Design* でより詳しく解説されている。

コンセプトステージ

コンセプトステージでは、デザインチームは、ゲームの基本的なアイデア、ターゲットとなるユーザー、プレイヤーの役割を決める。この段階の結果は、ビジョンドキュメント（vision document）† 6 やゲームトリートメント（game treatment）† 7 に記載される。これらの重要な決定は、その後のデザインプロセスで変更するべきではない。

コンセプトステージ中、作りたいゲームの種類が確定していなければ、ゲームの基本的なメカニクスが楽しいゲームプレイを生み出せるかどうかを検証する、基本メカニクスの実験バージョンをごく短期間で作成することができる。こういった概念実証プロトタイプは、他のチームメンバーや資金提供者にデザインビジョンを売り込んだり、鍵となる前提をプレイテストしたりする助けになる。しかし、次の推敲ステージでは、このプロトタイプを捨てて、再びゼロからやり直すことも想定に入れておく必要がある。これにより、コンセプトステージにおいて、バグのあるものを作っているかどうかを気にすることなく、迅速な作業を行うことが可能になる。計画が変更され、努力が水の泡になる可能性があるため、このステージが完了するまでは、本番用のデザインを始めるべきではない。

推敲ステージ

推敲ステージ（通常、プロジェクトが資金提供を受けた時点で開始）に入ると、ゲームの開発は本格化する。このステージの間に、ゲームメカニクスとレベルの作成、ストーリーの起草、アートアセットの作成などを行うことになる。このステージでは、開発チームは短い、イテレーティブな（iterative；反復的）サイクルで働くことが不可欠になる。サイクルごとに、一部のプレイが可能な製品なりプロトタイプなりを制作し、デザインを先へ進める前に必ずこれを検証・評価する。最初から何もかもうまくできることを期待しないようにしよう。このステージ中、多くのフィーチャーのデザインをやり直す必要が出てくるだろう。また、このステージ中に、ユーザーの

† 6 訳注：プロジェクト初期段階においてプロジェクトの意義や目標を関係者間で共有するための文書
† 7 訳注：主要場面の構成などの概略をまとめたもの

1.3 メカニクスとゲームデザインのプロセス

代表となるプレイヤーをチームの外から呼んで、ゲームの一部をプレイテストしてもらうのも良いアイデアだ。開発チームメンバーだけでプロトタイプの実験を行っていると、本当のプレイヤーが最終的にどのようにゲームをプレイし、どのようにゲームにアプローチするのか、正しくとらえることができない。開発チームメンバーは、作っているゲームのターゲット層から外れることがあるし、そのゲームのことを知りすぎていて、良い被験者にはならないものだ。

チューニングステージ

チューニングステージは仕様の凍結（feature freeze；フィーチャーフリーズ）時点から始まる。この時点では、ゲームのフィーチャーセットに満足しており、これ以上フィーチャーを追加するつもりがないことをチームとして決定しているはずだ。代わりに、ここまで作り上げたものをポリッシュ（polish；磨き上げ、作り込み）することに焦点を当てる。仕様の凍結を実行することは困難を伴うこともある。まだゲームに取り組んでいるうちは、最初のうちに考えつかなかった新しい巧妙なアイデアが常に浮かんでくるものだ。しかし、この開発後期では、小さな変更でさえゲーム上で目に見えない壊滅的な効果をもたらす可能性があり、デバッグやチューニングのプロセスを大幅に増加させることになる。つまり、変更はやってはいけないのだ！ どちらかと言えば、チューニングステージはサブトラクティブ法（subtractive process；不要な部分を取り除いて完成させるやり方）である。うまく機能していなかったり、ゲームに価値をほとんどたらさないものを取り除き、ゲームを輝かせる仕事をしてくれるものにデザインの焦点を当てるべきである。なお、ゲームプロジェクトを計画する際、チューニングの実際の作業量は過小評価されがちだ。私たちの経験上、ポリッシュやチューニングは、開発期間全体の3分の1から半分を占めるものである。

デザインの文書化

ゲームデザインドキュメントは、ゲームが構築されていくとともに、そのデザインを記録する。こういったドキュメントは、ゲーム会社ごとに独自の標準形式を持っており、使い方も異なっている。典型的に言えば、ゲームデザインドキュメントは、ゲームのコンセプト、ターゲットユーザー、コアメカニクス、意図するアートスタイルなどの簡単な説明から始まる。多くの企業が、デザインドキュメントを最新の状態に保っている。メカニズムが追加されること、またレベルがデザインされるごとに、このドキュメントに追加していく。このような理由から、デザインドキュメントは**ライブドキュメント**（生きるドキュメント）と呼ばれることがある。ゲームとともにドキュメントも成長していくのだ。

デザインプロセスを文書化することは重要であり、その理由は多くある。ゴールやビジョンを書きとめておけば、開発の後半になっても方向性を見失わずに済む。開発中にデザイン上の決定を書きとめておくことで、過去の決定についてあれこれ考え直すこともなくなる。最後に、チームで作業するときは、チームとして約束したゴールを1つのドキュメントに明記しておくことが、非常に役に立つ。これによって、チームの努力が発散したり、結局採用されないフィーチャーに開発者が大量のエネルギーを注いだりすることがなくなる。

さしあたり、読者の皆さんには、うまく機能する方法であれば何でも使って、デザインを文書化する習慣を身につけることを提案したい。*Fundamentals of Game Design* では、デザインドキュメントについての詳しい解説と、便利なテンプレートをいくつか紹介している。

早い段階でのメカニクスデザイン

ゲームメカニクスの作成は容易ではない。そこで、推敲ステージの早い段階でゲームメカニクスに取り組み始めることをお勧めする。理由は2つある。

- ゲームプレイはゲームメカニクスから発現する。ルールを見ただけで、そのゲームプレイが面白いかどうかを判断するのは、不可能ではないにしても困難である。そのメカニクスが機能するかどうかを調べる唯一の方法は、それをプレイするか、もっと良いのは、誰か他の人にプレイしてもらうことだ。これを可能にするために、複数のプロトタイプを作る必要があるだろう。これについての詳細は、本章の後半で触れる。
- 本書で焦点を当てるゲームメカニクスは複雑なシステムであり、ゲームプレイはそのシステム上のデリケートなバランスに依存している。機能するメカニクスを持つことができたとしても、開発プロセスの後半に新しいフィーチャーを追加したり、既存のメカニクスを変更したりすれば、そのバランスはいとも簡単に破壊されてしまう。

コアメカニクスが機能するようになり、それがバランスがとれていて面白いと確信した時点で、そのメカニクスと関連したレベルやアートアセットに取り組めるようになる。

先におもちゃを作ろう

ゲームデザイナーのカイル・ゲイブラーは、2009年の第1回グローバルゲームジャムでビデオ基調講演を行った。その講演の中で、彼は短期間でゲームを開発するときに役立つ7つのヒントをくれた。これはかなり有益なヒントであり、開発期間の長短にかかわらず、あらゆるゲーム開発プロジェクトに適用してもらいたいと私たちは考えている。

7つのうちの1つのヒントは、ここでの議論にかなり関連性がある。「先におもちゃを作ろう」(Make the toy first)というものだ。ゲイブラーはアセットやコンテンツの作成に時間を費やすより、メカニクスを確実に機能させるほうが先だと提案している。これは、プロトタイプの構築か、そのメカニクスのためのプルーフ・オブ・コンセプト (proof of concept ; 概念実証) から始めるべきだ、ということを意味している。たとえ、すばらしいアート、明確なゴール、巧みなレベルデザインを伴わなくても、そのメカニクスは遊んで面白いものであるべきだ。言いかえれば、触ったりいじったりするだけで面白い、楽しい、というおもちゃをデザインし、そこからゲームを構築する必要があるということだ。言うまでもなく、私たちもゲイブラーの意見に賛成だ。読者の方にも、彼のアドバイスに耳を傾けることをお勧めしたい。

オンラインで、ゲイブラーの軽妙な基調講演を見ることができる — www.youtube.com/watch?v=aW6vgW8wc6c。

失敗しないために

前述したように、ゲームメカニクスを明確にするためには、実際に構築する必要がある。本書で言及する方法論と理論はメカニクスがどのように働くかを理解する助けになるはずだ。初期のプロトタイプを作成するための、新しい、効率の良いツールも紹介するが、これらは本物の代わりにはなりえない。バランスのとれた斬新なメカニクスを持つゲームを作り出すために、プロ

1.4 プロトタイピング技法

トタイプを構築し、できるだけ多くイテレーションを回す必要がある。

1.4

プロトタイピング技法

プロトタイプとは、本物を作り上げる前にそのユーザビリティ（有用性）を検証するために作成する、製品またはプロセスのひな形（モデル）のことであり、通常は不完全なものである。プロトタイプは最終製品の領域までポリッシュする必要がないため、（通常は）制作や変更を製品より素早くそして低コストで行うことができる。ゲームデザイナーはメカニクスとゲームプレイの検証のために、ゲームのプロトタイプを作成する。ゲームデザイナーが使用する一般的なプロトタイプ技法としては、ソフトウェアプロトタイピング（software prototyping）、ペーパープロトタイピング（paper prototyping）、フィジカルプロトタイピング（physical prototyping）がある。

用語について

長年にわたり、ソフトウェアの開発者はプロトタイプの種類を表現するために多くの用語を考案した。**高再現性プロトタイプ**（high-fidelity prototype）は、いろいろな意味で目的とする製品に非常に近いものだ。高再現性プロトタイプは改良され続けてそのまま最終製品になることもある。相対的に言って、このプロトタイプは構築に時間がかかる。

これとは対照的に、**低再現性プロトタイプ**（low-fidelity prototype）は、より短期間で作り上げるものであり、最終製品に緻密に似ている必要はない。低再現性プロトタイプには、通常、最終製品で使用されているものとは異なる技術が使用される。たとえば、3D のコンソールゲームをプロトタイプするために 2D の Flash を使ってもよいし、ゲームのインタラクティブなストーリーボードを作るために PowerPoint を使うこともできる。開発者は、アイデアを素早く検証するために低再現性プロトタイプを作り上げる。こういったプロトタイプは、ゲーム中の要素 1 つに注力して作られる傾向がある。

また、開発者はプロトタイプを用いて、目的の製品の**パーティカルスライス**を作成することもできる。パーティカルスライスとは、図 1.4 に示されているように、ソフトウェアプロジェクトを視覚的に表現した用語だ。パーティカルスライスは、ゲームのフィーチャーの 1 つもしくは少数を実装するために必要な全要素（コード、アート、オーディオ他）を備えたプロトタイプのことを指す。ゲームが未完成の状態でも、各ゲームプレイをテストし、さらにゲームの雰囲気把握してもらうことができるため、極めて有用である。**ホリゾンタルスライス**は、（マルチプレイヤーオプションの特定のモードのみ、のような）ゲームの一部分だけで構成されるプロトタイプで、その他の要素は含まない。たとえば、あるホリゾンタルスライスは、完全なユーザーインターフェイスを備えていても、動作するメカニクスは一切備えていないかもしれない。

第1章 ゲームメカニクスデザイン

図 1.4 ゲームプロジェクトのパーティカルスライスとホリゾンタルスライス

第1章 ゲームメカニクスデザイン

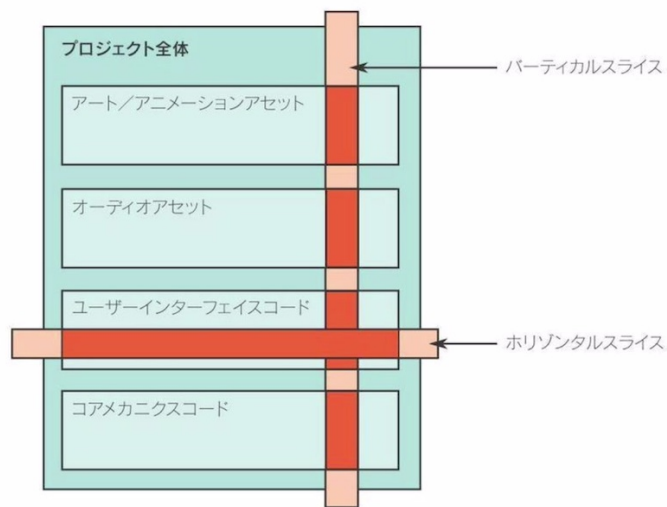


図 1.4 ゲームプロジェクトのパーティカルスライスとホリゾンタルスライス

図 4 図 1.4 パーティカルスライスとホリゾンタルスライス

(図の内容) - プロジェクト全体- アート／アニメーションアセット- オーディオアセット- ユーザーインターフェイスコード- コアメカニクスコード- パーティカルスライス（縦方向の矢印）- ホリゾンタルスライス（横方向の矢印）

ソフトウェアプロトタイピング

制作中のビデオゲームをプレイヤーがどう受け止めるかの感触を得たいのであれば、最善の方法は、できるだけ早く、そのデザインの近似となるソフトウェアのプロトタイプを作成することだ。プロトタイピングのプロセスを高速化するために、ターゲットプラットフォームとはまったく異なるものだとしても、GameMaker や Unity といったオープンソースゲームエンジンやゲーム開発環境を用いるほうが良い場合がある。

ソフトウェアプロトタイプを使うアドバンテージは、たとえアートが一時的なものだけで、フィーチャーも未完成ないしはバグだらけだったとしても、ゲームプレイに関する良い指摘を得られる点だ。しかし、他の種類のプロトタイプに比べて、作成に時間がかかることが欠点である。開発に関する制限事項や開発チームのスキルによっては、実際のゲームを作成する場合と同程度の時間がかかってしまう場合もある。しかし、たとえ最終的にプロトタイプ用アートとコードをすべて捨てることになっても、ソフトウェアプロトタイプを作ることは意義がある。初期のソフトウェアプロトタイプを持つことで、プロジェクトの方向性を見失わなくて済む。プログラマは必要なゲーム要素の種類を把握し、レベルデザイナーはデザインの方向性に関する着想を持つことができ、ゲームデザイナーはアイデアを実際に遊んで検証する環境を手にすることができる。ソフトウェアプロトタイプは、ほとんどデザインドキュメントのように機能する。開発チームは本物を作る際に、プロトタイプを参照することが可能だ。また、インタラクティブなフィーチャーのようなゲームの側面は、言葉による記述よりも、プロトタイプのほうがうまく説明することができる。

1.4 プロトタイピング技法

ソフトウェアプロトタイプを成功させる重要なポイントの1つは、プロトタイプ内で簡単なゲームのカスタマイズができることだ。ゲームプレイにとってゲーム内の重力が不可欠な3Dプラットフォームゲームの場合、ベストな感触をつかむために、必ずプレイ中にゲームデザイナーが設定を簡単に変更できるようにすること。また、リアルタイムストラテジーゲームで工場にリソースを生産させる場合は、適切なバランスを素早く見つけられるように、必ず生産速度を簡単に変更できるようにすることだ。このカスタマイズ機能のために手の込んだインターフェイスを作って時間を無駄にしてはいけない。重要な初期値をテキストファイルに書いておいて、プログラムの起動時に読みこむようにするだけでかまわない。この方法では、デザイナーはファイルを編集してプログラムを再実行するだけで、変更した値でゲームをプレイすることができる。もっとよい方法は、簡単なゲーム内コンソール（off-the-shelf console）を作って、ゲーム中に変更を加えられるようにすることだ。これで、開発とテストのサイクルはさらにスピードアップする。

Spore の多くのプロトタイプが www.spore.com/comm/prototypes で公開されている。いくつかダウンロードして遊んでみることをお勧めする。公開されているプロトタイプを見ていくと、AAA タイトルを担当する商業ゲームスタジオの開発プロセスについて、他では得られない見識が得られるだろう。

ヒント

ペーパープロトタイピング

ソフトウェアプロトタイプは、比較的作成が遅く、コストも高くつくため、多くのゲームスタジオがペーパープロトタイピング技法を活用している。ペーパープロトタイプは、コンピュータによる演算を用いない、目的のゲームに似せた卓上ゲームである。ゲームメカニクスの中には、媒体に依存しないものもある。正確なタイミング、物理、その他コンピュータ演算を必要とするメカニクスにあまり依存しないビデオゲームであれば、そのコンセプトからボードゲームを作成できるはずだ。ゲームがコンピュータの演算を必要とするメカニクスに大きく依存している場合でも、演算を必要としないゲームの側面をペーパープロトタイプに落とし込む努力に時間を割く価値はあるものだ。一般に、プロトタイプとはゲームの特定の側面に注目したものであることを思い出してほしい。たとえば、ゲームプレイの大半が傑出した物理シミュレーションから得られるゲームであっても、その内部経済のみにフォーカスしてプロトタイプを作ることは可能だ。ペーパープロトタイプをデザインする場合は、はじめに掘り下げるべきポイントを見極めておくことが重要である。

ペーパープロトタイピングをあなどってはいけない。良いボードゲームをデザインすることは、それ自体が芸術であり、少なくとも良いビデオゲームをデザインすることと同じくらい難しい。普段からさまざまなボードゲームに慣れ親しんでいれば、その知識も役に立つ。ボードゲームには、「ダイスを振って出た目を進む」だけではない、もっと多くのゲームメカニクスが存在しているのだ。

お勧めのペーパープロトタイピングキット

プロのゲームデザイナーであるコルプス・エルロッド (Corvus Elrod) は、プロトタイピングキットとして次の道具を手元に置いておくことを勧めている。

- 背面の色が異なる、ほぼ同じカードデッキ（トランプ）を2セット
- 小さなノート（大きすぎると邪魔になる）。もちろん、書きやすい鉛筆かペンも。
- ポーカーチップや、基石といった類のトークン。
- いくつかのダイス。何面体かは重要ではなく、大きな数字も必要ではない。確率を用いるメカニクスをデザインしているなら、10面ダイスが2つあれば1～100の乱数を生成するのに便利だ (Elrod 2011)。

さらに、私たちからも推薦したい道具がある。

- 付箋紙
- 3×5 インチ (75×125mm) のカード (白紙)

カードスリーブも何枚か用意しておきたい。カードスリーブとは、マジック：ザ・ギャザリングのようなトレーディングカードゲーム用のカードを保護するためにプレイヤーが使用するプラスチック製のスリーブだ。これはゲーム専門ショップで購入することができる。スリーブの中にカードを並べておけば、プレイ時に素早く見つけられるし、シャッフルも取り回しも簡単になる。おまけに、古くなったカードの上に新バージョンのカードを入れておける。こうしてカードのデザイン履歴が保持できるのだ。これらの道具をそろえれば、乱数を生成する手段はあるし、（ポーカーゲームにおいてポーカーチップがお金になるように）数値を表すトークンもあるし、ゲーム盤を含む何かを書き込めるブランク素材もあり、浮かんだアイデアを書きとめるノートもある。ペーパープロトタイピングを始めるために本当に必要となるものばかりだ。

ペーパープロトタイピングには重要なメリットが2つある。素早く作れることと、ペーパープロトタイピングの性質上、カスタマイズ可能である点だ。プログラミングも不要なので、素早く作成することができる。ペーパープロトタイピングを作るときは、カードやゲーム盤の見た目をよくするのに時間を浪費すべきではない。それより、ゲームルールを起草して、それをテストすることに時間を使うべきである。若干の技術と経験があれば、どんなゲームでもほんの数時間で、まともなペーパープロトタイピングを作成できる。これができれば、メカニクスのプレイテストやバランス調整に十分な時間が残されるはずだ。

ペーパープロトタイプでは、ルールを変更することが簡単だ。その場で変更を行うことさえできる。ゲーム中に意図したとおりに働いていない部分に気づいたら、即座に変更しよう。こうすることで、遊びながらゲームの大部分を作っていくことができる。これより短いイテレーションサイクルはないだろう。

ペーパープロトタイピングには欠点も2つある。それは、テストプレイヤーを参加させることが困難である点と、すべてのメカニクスがうまくボードゲームで再現できるわけではない点だ。新しいテストプレイヤーにペーパープロトタイプをテストさせる場合、プレイヤーにルールを口頭で説明する必要がある。いつルールを変更するか分からない以上、ルールを書きとめておくのは時間の無駄だからだ。さらに、テストプレイヤーにプレイテスト経験やボードゲームのプレイ経験がほとんどない場合、ペーパープロトタイプとビデオゲームとの関連性があまり分か

1.4 プロトタイピング技法

ってもらえない可能性がある。

どちらかと言うと、すべてのメカニクスが容易にペーパープロトタイプで再現されるわけではない、という点がより根深い問題だ。前述の通り、ゲーム内物理と絡むメカニクスを再現することは難しい。コンピュータの演算を伴う継続的メカニクスは、実際にコンピュータ上で実装することが必要になる。覚えておくべきは、ペーパープロトタイプは離散的メカニクスの検証に最適であるということだ。ゲーム内経済やゲームの進行を制御するメカニクスをデザインする場合に用いるほうが良いだろう。

フィジカルプロトタイピング

プロトタイピングはソフトウェアや紙のゲームに限定されるものではない。ルールを起草して、リアルにゲームを演じることで同様な効果が得られるものだ。継続的・物理的メカニクスの多いゲームでは特にそうだ。オフィスビルでレーザータグ銃^{†8}を装備して遊んでみれば、ファーストパーソンシューターが提供する気持ちをかなり具体的に体験できるだろう。またこの手法は、ペーパープロトタイプよりも準備の手間が少ないことが多い。ペーパープロトタイピングと同様にフィジカルプロトタイピングは柔軟性が高い。大きな効果を得るために、フィジカルとペーパーのプロトタイピング技法をミックスするゲームデザイナーもいる。しかし、欠点もペーパープロトタイピングと同様で、フィジカルプロトタイピングには困難な側面もある。それは、正しい結果を得るためには、ゲームデザイナーとプレイヤーにある種の技術と専門知識を要求する点である。

フィジカルプロトタイピングの価値を理解するには、ライブアクションロールプレイ（LARP：Live Action Role Play）セッションに参加するか、見学するのがよい。ラーパー（LARPer）と呼ばれる参加者たちが、肉体を使って実戦を交えるためにさまざまなテクニックを自ら編み出し、魔呪など、現実の世界では物理的に存在しないものを扱う手法も考案している。LARP は特別な場所で行われるので、最寄りの LARP コミュニティーを探すとよい。ウェブサイト <http://larp.meetup.com> で一覧を見ることができる^{†9}。

ヒント

プロトタイプのフォーカス

適切なプロトタイプの媒体を選択する以外にも、効果的なプロトタイピングのために重要なポイントがもう1つある。それは、どこにフォーカスを当てるかを見極めることだ。プロトタイプの構築を始める前に、このプロトタイプから何を学びたいのか、自問するとよいだろう。もし経済のバランスに関することを見いだそうとしているのであれば、新しいユーザーインターフ

^{†8} 訳注：安全にサバイバルゲームができる玩具^{†9} 訳注：非常に残念なことに、日本では LARP はほとんど見られない。もし仮に日本で体験する機会があれば、逃さず体験したい。

第1章 ゲームメカニクスデザイン

エイスを検証する意図で作ったプロトタイプとは、異なるプロトタイプが必要になる。Spore (www.spore.com/comm/prototypes) のプロトタイプを見てみると、それぞれのプロトタイプは特定の理由のために作られていることが分かる。

フォーカスを1つに絞込むことで、プロトタイプを高速に作りやすくなる。その一点に集中して作業すれば、ゲーム全体をプロトタイプする必要もない。フォーカスを絞れば、テストプレイヤーから正しいフィードバックも得やすくなる。調査したい課題とは無関係のフィーチャー（もしくはバグ）にテストプレイヤーが気を取られることも少なくなるだろう。

プロトタイプでフォーカスする内容によって、選択するプロトタイプ技術も変わってくる。物理を利用したプラットフォームゲームにおいて、パワーアップに関するバランスのとれた経済をデザインしたい場合は、たとえ物理のところをボードゲームとして再現することが難しいとしても、ペーパープロトタイプがうまく機能するだろう。その一方で、新しい入力デバイスを使った操作体系を吟味したい場合は、実際のゲームに近い高再現性のソフトウェアプロトタイプが必要になる。

以下は、よくフォーカスの対象となるゲームデザインの側面をプロトタイプ初期から後期まで、およその順で示したものだ。

● **技術デモ** (tech demo ; テックデモ) : あなたもプログラマのチームも、関係する技術に実際に対処できるかどうかを確認しておくことは得策である。技術デモでは、ゲーム技術の中でも最も難しく最も斬新な側面に取り組んで、このゲームを作り上げる力があることを自分自身に証明し、理想を言えばゲームのパブリッシャーにも立証することを目指すべきだ。開発が後半に差し掛かってから予期せぬ問題に直面しないためにも、技術デモは、早い段階で作っておく必要がある。技術デモを構築している間、興味深いゲームプレイのチャンスに目を光らせておこう。特に、新技術に取り組んでいるときは、シンプルなるものを素早く構築することで、のちにより深い洞察が得られることもある。

● **ゲーム内経済** : ゲームの経済は多くの重要なリソースを中心に回っている。ゲーム内経済は、ペーパープロトタイピング技法のような低再現性のプロトタイプで再現可能である。時期としては、デザインプロセスの早い段階で作成すべきだ。プレイテストでの一般的な質問を以下に示す——「ゲームのバランスはとれているか?」「常勝できるような絶対優位の戦略 (dominant strategy) がないか?」「プレイヤーが楽しめる選択肢が用意されているか?」「プレイヤーは選択の結果を十分に予測できるか?」。ゲーム内経済のプレイテストに適切なプレイヤーを獲得することが重要だ。あなたもチームメンバーも立派な被験者である。もちろん、あなたはゲームがどのようにプレイされることを意図されているのかを知っているため、ある程度のハンデを課す必要はある。一般的に、この手のプロトタイプのための理想的なテストプレイヤーは、素早くメカニクスを把握して、抜け道を探し出して活用することができる経験豊富なパワーゲーマーである。被験者には、できる限りゲームを破

1.4 プロトタイピング技法

綻させられるポイントがないか探してもらうように依頼しよう。もしゲームが破綻しうるものなのであれば、それを常に把握しておく必要がある。

● **インターフェイスと操作体系**：プレイヤーがゲームをコントロールできるかどうかを調べるには、ソフトウェアプロトタイプが必要だ。このプロトタイプには多くのコンテンツを用意したり、レベルをきっちりと完成させたりする必要はない。むしろ、プレイヤーがゲーム要素やインタラクションをひとつお試すことができる遊び場であればよい。通常、この種のプレイテストで確認すべき事項は次のとおりだ——「プレイヤーは提供されたアクションを正確に実行できるか?」、「プレイヤーが希望または必要としているアクションが他にないか?」、「プレイヤーが正しい決断を下すために必要な情報が与えられているか?」、「操作システムは直感的か?」、「プレイヤーに、プレイする上で必要な情報は提供されているか?」、「ゲーム内でダメージを受けたことや、重要な状態変化が生じたときにプレイヤーが気づくか?」。

● **チュートリアル**：ゲーム開発の後半に入らなければ、よいチュートリアルを作ることはできない。なんと言っても、まだ変更されるかもしれないゲームメカニクスのチュートリアルを作るために、時間とリソースを浪費してはいけない。チュートリアルのテストは、ゲームをまだ見たことのないテストプレイヤーにやってもらうことが重要だ。ゲーム開発は、それ自体が長期間にわたる詳細なチュートリアルのようなものだ。開発者はメカニクスの調整に多くの時間を費やし、その間にゲームをたくさん遊ぶことになる。開発者のあなたは、自分が開発中のゲームの腕を上げていることを簡単に忘れるものだ。それゆえ、あなたはゲームの難易度と学習曲線については、自分の判断を信用してはいけない。そのためには新しいプレイヤーが必要であり、彼らがプレイしているときに彼らの学習プロセスに干渉してはならない。チュートリアルプロトタイプのための最も重要な質問はこれだ——「プレイヤーはゲームを理解しているか、また、どのようにゲームをプレイするべきかを理解しているか?」。

参考用ゲーム：自由に使えるプロトタイプ

既存のゲームから自分のプロジェクトに合うものをひな形として活用することが、ゲームのプロトタイプを最も効率的に作成する方法となることがある。こうすれば、他の人がすでに行った多くの作業を活用できる。この方法は、ユーザーインターフェイスデザイン、コントロール、基本的な物理など、異なるゲームであってもプレイヤーが一貫性を望むのではとりわけ有用だ。イノベーションをしたがために、PCの一人称視点ゲームで伝統的に用いられている WASD キーの操作体系を、ESDF に変更しても意味がない。

言うまでもなく、他人のデザインを盗むべきではない。しかし、他者の作品から学習したり、他人が犯した過ちを回避したりすることに問題はないはずだ。自分のプロジェクトのために参考用ゲームを選ぶときは、プロジェクトの規模に注意を払おう。開発期間が2月しかないのに、大規模熟練チームが数年かけて作ったゲームを参考にしてはいけない。ゲームのインターフェイスやメカニクスを部分的に参照して研究する場合を除けば、計画中のゲームと同程度の規模と品質を持ったゲームを参考用を選ぶ。

1.5 # まとめ

ゲームメカニクスとは、ゲームの心臓部分の要素やプロセスだけでなく、そのプロセスを実行するために必要とされるデータまでを含んだ、明確に指定されたゲームのルールである。メカニクスは継続的メカニクスと離散的メカニクスとに分類することができる。継続的メカニクスは、通常、毎秒大量の浮動小数点計算を伴うリアルタイム処理で実装され、ゲーム内物理処理を実装するため用いられる。離散的メカニクスはリアルタイムに動作することもあれば、そうでない場合もある。また、ゲームの内部経済を実装するために整数値を使用する。ゲームメカニクスは早い時期でデザインを開始することが不可欠である。メカニクスがなければ、プロトタイプの作成やプレイテストができないからだ。

メカニクスの中には、創発型ゲームプレイに大きく貢献する特定の構造を持つメカニクスがある。次の第2章と第3章では、ゲームメカニクスにおけるこの構造的観点をさらに詳細に掘り下げていく。そして、こうした観点からゲームメカニクスのデザインを支援する実用的な手法とデザインツールを編み出していく。

1.6 # 演習

1. プロトタイプの技術を磨こう。既存のビデオゲームをペーパープロトタイプに置き換えてみよう。
2. あなたが作り上げたいゲームに近い参考用ゲームを見つけよう。あなたが心の中で描いたゲームを実証するために、その参考用ゲームのどの側面が使えると思ったのかを説明しよう。
3. 現在市場に出回っているゲームのうち、本章で解説した5種類のゲームメカニクス（物理、内部経済、進行、戦術的な操作、社会的インタラクション）をそれぞれを持つ離散的メカニクスと継続的メカニクスの例を見つけよう。ただし、この章で実際に挙げられた例は使用しないこと。

第 2 章

創発型と進行型

前章では、物理、内部経済、進行、戦術的な操作、そして社会的インタラクションという、5 種類のゲームメカニクスを紹介した。ゲーム研究分野で**進行型ゲーム**と呼ばれるゲームプレイを生み出すメカニクスは、この 5 種類のうちの進行メカニクスである。他の 4 種類のメカニクスは、もう 1 つのカテゴリーである**創発型ゲーム**に大きくかかわるメカニクスだ。本章では、この関係を分かりやすくするために、この 4 種類のメカニクスをまとめて**創発メカニクス**と呼ぶ。創発型ゲームと進行型ゲームという 2 つのカテゴリーは、ゲームプレイを形成するための重要な選択肢であると考えられている。本章では、この 2 つの重要な違いをより詳細に解説しながら、それぞれのカテゴリーの例を示していく。また、創発と進行を生み出すメカニクスの構造上の違いと、デザイナーが 1 つのゲームに創発と進行を統合しようとするときにそれらのメカニズムが引き起こす課題と機会について探ることにする。

2.1 ## 創発型と進行型の歴史

創発型と進行型のカテゴリーが最初に紹介されたのは、ゲーム学者ジェスパー・ジュールによる論文 “The Open and the Closed: Games of Emergence and Games of Progression (オープンとクローズド：創発型ゲームと進行型ゲーム)” (Juul 2002) によってである。簡単に言うと、創発型ゲームとは、比較的簡単なルールであるが、多くのバリエーションを持つゲームである。「創発」という言葉が使われるのは、ゲームのチャレンジとイベントの流れが事前に計画されているのではなく、プレイする間に現れ出てくるからである。創発とは、多くのルールの潜在的な組み合わせによって生成されるが、これはボードゲーム、カードゲーム、ストラテジーゲームだけでなく、一部のアクションゲームでも見られる。ジュールによると、「創発とは原始的なゲームの構造である」(Juul 2002, P.324)、つまり、最も早くから作られたゲームは創発型ゲームであったし、新しいゲームを作成する際には、多くの人は創発のデザインから始める。

第2章 創発型と進行型

創発型ゲームは、プレイ中に実にさまざまな構成や状態に変化する。可能性のあるチェスの駒の運びによってそれぞれ異なるゲームの状態が作られるのは、ポーン（歩）の動きが1マス進っても決定的な違いが生まれるからだ。チェス盤上の駒の組み合わせ可能な数は膨大であるにもかかわらず、ルールは易々と1ページに収まってしまう。似たようなことは、シミュレーションゲーム「シムシティ」の住宅地区や、ストラテジーゲーム「StarCraft」のユニットの配置にも言える。

ビデオゲーム以外の創発と進行

ジュールによる分類では、すべてのボードゲームは創発型ゲームとなる。ランダムな要素でもって開始されるゲーム、たとえばトランプやドミノといった類は、創発型ゲームに当てはまる。そのようなゲームは、駒や手札の数が少なく、ルールは事前にデザインされたデータか、あるいはゲームカードだ。モノポリーのチャンスカードと共同基金カード上のテキストは事前にデザインされたデータの例であるが、容量は1KBに満たない。

進行型ゲームは、デザイナーが事前にデザインしたデータが大量に必要とされ、それによってプレイヤーは任意の点にアクセス（**ランダムアクセス**と呼ばれる）することが可能になる。これは、ボードゲームではとても無理だが、今では数GB単位のデータを保存することもできるビデオゲームにとっては容易なことである。進行型ゲームは比較的新しい構造で、1970年代のテキストアドベンチャーゲームが嚆矢である。しかし、進行型ゲームはコンピュータ上で実行されているゲームに限られているわけではない。紙とペンを使ったテーブルトークRPGの「ダンジョンズ&ドラゴンズ」ではシナリオ本が発行されたが、こうしたシナリオ本は、子ども向けのゲーム本であるChoose Your Own Adventureシリーズのように、それ自体が進行型ゲームになっているのだ。書籍も、大量のデータの取り扱い、簡単なランダムアクセスの提供が可能な媒体である。

これとは対照的に、進行型ゲームは多くの事前設計されたチャレンジを提供するゲームだ。こうしたチャレンジはデザイナーが順番に並べたもので、一般に洗練されたレベルデザインを通じて提供される。進行は厳密に統御されたイベントのシーケンス（sequence）に依存している。ゲームデザイナーは、プレイヤーが一定の順番でしかイベントに遭遇できないようにレベルをデザインすることで、プレイヤーが遭遇するチャレンジを決定している。ジュールによれば、ひと通り順番に進展する「ウォークスルー（walkthrough）」を持つゲームはすべて進行型ゲームである。最も極端なものは、プレイヤーを「レールに乗せて運ぶ（railroaded）」もので、プレイヤーは、次々にチャレンジをこなすか、失敗になるかしかない。進行型のゲームでは、ゲームの状態の数は比較的少なく、ゲームデザイナーはゲームに仕込まれたものを完全にコントロールできる。このことは、進行型のゲームはストーリー式のゲームによく適合することを示している。

進行型ゲームという用語を、レベルアップや学習曲線、スキルツリーなどのゲームの進行にかかわる用語と混同しないように。本書ではジュールの定義を使う。つまり、進行型ゲームとは事前にデザインされたチャレンジが与えられるゲームであり、解決策はチャレンジごとに1つだけ決まっていて、そのシーケンスも固定である（あるいはわずかなバリエーションしかない）場合が多い。

ヒント

2.2 ## 創発型と進行型の比較

ジュールは最初の著作で創発型のゲームのほうが優れていると書いている。「理論的に言って、創発型のほうがより興味深い構造をしている」(Juul 2002, p.328)。彼によれば、創発型ゲームとは、プレイヤーの自由意思とデザイナーの制御がバランスをとるかたちで、デザイナーがゲームを作ることができる手法である。創発型ゲームでは、決められたルールによってあるイベントが起こりやすくなることはあるが、ゲームを公開する前にデザイナーが各イベントを詳細に指定することはない。しかし実際のところ、創発型の構造を持つゲームでさえ、かなり規則的なパターンに従うことは多い。ジュールは、Counter Strike でほとんど必ず勃発する銃撃戦について論じている (Juul 2002, p.327)。別の例が Risk で見つけられる。プレイヤーの領地は初期段階ではマップ上に点在しているが、ゲームプレイの過程で領地の所有者は変わっていき、プレイヤーたちは通常、隣接したいいくつかの領土を支配しようと手が伸びるようになる。

データ集約型とプロセス集約型

ゲームデザイナーのクリス・クロフォード (Chris Crawford) が提案した**プロセス集約型**と**データ集約型**という考え方は、ゲームにおける創発型と進行型にも通用できる。コンピュータがその他のゲーム媒体と違う点は、数値処理に優れたところだ。さらに、膨大なデータベース内のランダムな場所に高速にアクセスできる。これは進行型ゲームでうまく活用できる能力である。しかし、コンピュータが真価を発揮するのは、その場で新しいコンテンツを生み出し複雑なシミュレーションを処理する能力にこそある。コンピュータは、その登場以前の媒体が持ち得なかった、巧妙なシミュレーションや創発型のゲームプレイで、プレイヤーやデザイナーを驚かせるだけの能力を備えている。クロフォードは、ゲームはコンピュータのこの能力を生かすべきであると考えている。ゲームはデータ依存型ではなく、プロセス依存型であるべきだということだ。彼は、ビデオゲームは進行型ゲームというよりもむしろ創発型ゲームであるべきだと述べている。

ジュールの後の著作 *Half-Real, Video Games between Real Rules and Fictional Worlds* では、創発型ゲームと進行型ゲームについてもっと微妙な表現をしている (Juul 2005)。現代の大半のビデオゲームはハイブリッド型であり、創発型と進行型の両方のフィーチャーがある程度含まれている。グランド・セフト・オート・サンアンドレアスは、広大なオープンワールドを提供するだけでなく、新たな要素を少しずつ導入し、少しずつこのゲーム世界をアンロックしていくミッション構造も有している。ストーリー主導型のファーストパーソンシューター「Deus Ex」では、ストーリーラインによって次に行くべき場所は決まっているが、その途上に立ちはだかるチャレンジにはさまざまな戦略と戦術を使って対処できる。Deus Ex のウォークスルーを書くことは可能だ。ジュールの分類に従えば進行型のゲームということになるが、このゲームでは実にさまざまなウォークスルーが可能だ。少なくとも理論的には、シムシティで特定のマップのウォークスルーを作ることができるのと同じである。たとえば、プレイヤーに対して、効果的な街作りをするために、ある時期にある特定の地区やインフラを構築するように指示

第2章 創発型と進行型

することはできる。このようなワークスルーに従ってプレイすることは難しいだろうが、とにかくワークスルーの作成は可能である。

創発型が進行型よりも優れているわけではない。両者が異なっているだけのことだ。純粋な創発型ゲームと純粋な進行型ゲームは両極端に位置している。たとえばビジュエルド（Bejeweled）のようなカジュアルゲームの多くが純粋な創発型ゲームである。純粋な進行型ゲームはかなり稀である。最も典型的な例は The Longest Journey などのアドベンチャーゲームだが、アドベンチャーゲームはすでにかつてのような支配的なジャンルではない。両方の要素を持つゲームは、レベルの中に創発型の振る舞いが現れることが多いものの、プレイヤーが脱線できないようにレベルの順番は厳密に決まっている（進行型の振る舞い）ことが多い。今日では、ハーフライフやゼルダの伝説シリーズのようなアクションアドベンチャーゲームは、伝統的なアドベンチャーゲームよりもはるかに一般的である。アクションアドベンチャーゲームにはゲームプレイの一部に何らかの形式の創発型アクションが含まれている。大規模なゲームにおいては、ハイブリッド型は、最もポピュラーな形態である。

2.3 ## 創発型ゲーム

ジュールが分類する以前、ゲームにおいて「創発型」という言葉が使われたのは複雑性理論に由来している。複雑性理論とは、構成するパーツから（直接的に）派生し得ないシステムの振る舞いを指した言葉である。それと同時にジュールは、創発型の振る舞いと、デザイナーが単に予測しなかった振る舞いが起こるゲームとを混同しないよう注意を促している（Juul 2002）。ゲームでは、あらゆる複雑系がそうであるように、全体は部分の総和以上のものとなる。碁とチェスは、比較的シンプルな要素とルールでありながら、ずば抜けた奥深いゲームをもたらすことで有名である。似たようなことは、テトリス、バルダーダッシュ、World of Goo のような、比較的シンプルなコンピュータゲームでも同様に言うことができる。これらのゲームは、比較的シンプルなパーツで構成されているが、戦略やアプローチの数は膨大である。何度やっても同じプレイスルーにはならないだろう。創発型という性質を持つゲームプレイは、個々のパーツの複雑さから来るのではなく、パーツ間の多くの相互作用の結果として生み出される複雑さから来るのである。

複雑なシステムのシンプルなパーツ

複雑系科学は、実生活の中にある、あらゆる種類の複雑系を研究している。複雑系で活動している実体や要素は、それ自体が極めて複雑であるが、シミュレーションは一般にシンプルなモデルを使って行われる。たとえば、さまざまな環境における歩行者の流れを研究する上で大きな成果を上げたシミュレーションは、わずか数点の行動ルールと目標で構成されていた（Ball 2004, pp.131-147）。本書でも、これと同じようなアプローチをゲームに適用する。複雑な要素

2.3 創発型ゲーム

をいくつか使って創発型ゲームを作ることとは可能であるが、私たちはシンプルなパーツで機能しながらも創発型のゲームプレイを作り出すゲームシステムのメカニクスにより強い関心を持っている。私たちのアプローチの利点は、当初は理解するのが難しいものの、最終的に効率的よく構築できることである。

確率空間

第1章では、ゲームは、現在の状態やプレイヤーによってもたらされた入力に基づいて、ある状態から別の状態へと進行させる仮想のステートマシンと見なされることが多いと説明した。ゲームでは、状態の数は急速に増えていくことがあるが、すべての状態が可能になるわけではない。たとえば、チェス盤にランダムに配置された駒の状態すべてが、実際のゲームプレイを通して到達しうるゲーム状態を表しているわけではない。実際では自分の色のポーンを一番手前の列に置くことはできないし、自分のビショップを両方とも同じ色のマスの上に配置することはできない。ゲームの研究では、取りうる状態の数が非常に大きいとき、これらを集合的に**確率空間** (probability space) と呼んでいる。確率空間は、現在の状態から到達可能なすべての状態を表しており、広さあるいは深さを伴ったものと見なすことができる。確率空間の形状が広い場合、現在の状態から到達可能な多くの異なる状態がある。これは通常、プレイヤーが多くの選択肢を持っていることを意味する。確率空間の形状が深い場合、これから多くの選択をした後で、多くのさまざまな状態に達することができる。

C・E・シャノン (C. E. Shannon) は、初期の論文 “Programming a Computer for Playing Chess” において、チェスや碁のようなゲームに存在する潜在的なゲーム状態の数は、地球上に存在する原子の数よりも多いと見積もった (Shannon 1950)。潜在的なゲーム状態の数を決定するのはゲームのルールだが、ルール数が増えれば、ゲームの状態数が増えるわけではない。また、ゲームが多くのルールを用いることなく数多くの潜在的な状態を作れるとき、そのゲームはプレイヤーにより理解されやすいものになる。

ゲームプレイとゲームの状態

プレイヤーが可能なゲーム状態（その確率空間）を通り抜けるとき、プレイヤーが通るこの道筋は**軌跡** (trajectory) と呼ばれる。あるゲームにおいて起こり得るゲーム状態とプレイの軌跡は、そのゲームルール体系の創発的な特性である。多くの異なる興味深い軌跡を許容するゲームは、ほぼ間違いなく、ほとんど軌跡を作らなかつたり、あまり関心を寄せられることのない軌跡を作るゲームに比べて、多くのゲームプレイを提供する。しかし、ルールを見ただけで、そのゲームプレイの種類と品質を決定づけることは、不可能ではないにせよ、困難である。三目並べ (tic-tac-toe) と Connect Four (四目並べの一種) のルールを比較すると、この難しさが分かるはずだ。三目並べのルールは次の通りである。

1. 3×3 の格子上でゲームを遊ぶ。
2. プレイヤーは交互にマス目を埋める。
3. 各マス目は1回しか埋められない。
4. 先に（縦・横・斜めいずれかの形で）3つのマス目を並べたプレイヤーが勝ちとなる。

第2章 創発型と進行型

Connect Four のルールは次の通りである（三目並べとの違いを太字で示した）。

1. **7×6** の格子上でゲームを遊ぶ。
2. プレイヤーは交互にマス目を埋める。
3. 各マス目は1回しか埋められない。
4. **各列の、空いている一番下のマスから積み重ねることしかできない。**
5. 先に（縦・横・斜めいずれかの形で）**4**つのマス目を並べたプレイヤーが勝ちとなる。

この2つのゲームの間には、ルールの違いはほんのわずかしかない。だが、ゲームのルールを理解するのに求められるメンタルな努力の量に比べると、ゲームプレイでの違いは莫大なものとなる。市販の Connect Four では、最も複雑なルール（4番目の項目）は重力によって遵守させられる。つまり、プレイヤーのトークンを上方のプレイエリアに置くと、自動的に最下層の配置可能なスペースまで落下するのだ（図2.1を参照）。この自動化により、プレイヤーはいちいちこのルールの遵守を確認する煩わしさから解放され、ルールの効果に集中できる。ルールの複雑度ではほんのわずかしか違いがないのに、三目並べは小さな子どものみに適していて、一方で Connect Four は大人にも楽しまれている。Connect Four では多くのさまざまな戦略を用いることが可能であり、それによりゲームをマスターするにもずっと時間がかかる。熟練プレイヤーが対戦すると、三目並べでは確実に引き分けとなってしまっだろうが、Connect Four では白熱した対戦となるだろう。このゲームプレイの違いは、ルールの違いを見ただけでは説明が難しいものだ。

! [図2.1 Connect Four（重力付き四目並べ）では、重力によって垂直方向には一番下にしか玉を置けない仕組みになっている。]

図2.1 Connect Four（重力付き四目並べ）では、重力によって垂直方向には一番下にしか玉を置けない仕組みになっている。写真はウィキメディアコモンズの著権者 Poppericopp の許可によりクリエイティブコモンズ（3.0）ライセンスの元で無料提供

2.3 創発型ゲーム

シヴィライゼーションの例

シド・マイヤーのシヴィライゼーションは創発型ゲームの好例である。シヴィライゼーションでは、プレイヤーは約 6000 年間にわたり、文明の発展を導いていく。ゲーム中、プレイヤーは都市、道路、農地、鉱山、軍事ユニットを構築していく。築き上げた都市は、寺院、兵舎、裁判所、証券取引所などを建てることで、アップグレードする必要がある。都市はお金を生み出し、プレイヤーはそのお金を新技術の研究に投資したり、贅沢品と交換して住人を幸福にしたり、ユニットの生産やアップグレードのスピードアップに使ったりすることができる。シヴィライゼーションはタイルベースのマップを用いたターン制のゲームであり、各ターンはプレイヤーの文明の歴史における年数を表現している。プレイヤーのとり選択が、文明の成長する早さや、テクノロジーの洗練具合や、軍隊の強さ、といった要素を決定する。コンピュータが操作を担当するいくつかの文明とは、次に行くべき場所は決まっているが、その途上に立ちはだかるチャレンジにはさまざまな戦略と戦術を使って対処できる。Deus Ex のウォークスルーを書くことは可能だ。ジュールの分類に従えば進行型のゲームということになるが、このゲームでは実にさまざまなウォークスルーが可能だ。少なくとも理論的には、シムシティで特定のマップのウォークスルーを作ることができるのと同じである。たとえば、プレイヤーに対して、効果的な街作りをするために、ある時期にある特定の地区やインフラを構築するように指示

シヴィライゼーションはさまざまなゲーム要素を持った大規模ゲームである。ただし、個々の要素は驚くほど単純なものだ。都市のアップグレードに関するメカニクスは、単純なルール数点で表現することができる。たとえば、寺院は 1 ターンにつき 1 ゴールドの維持費がかかるが、都市の不幸な人の数を 2 人減らす。ユニットはシンプルな整数値をいくつか持っていて、その数値はユニットが占領できるタイルの数)、攻撃力、防御力を表している。ユニットによっては、特殊能力を持っているものもある。たとえば、入植者は新しい都市を建設することができ、大砲は離れた距離から敵ユニットを砲撃することが可能だ。地形はユニットの性能に影響を与える。山を越えるには移動力を余計に消費するが、ユニットの防御力を 2 倍にもしてくれる。山越えの余分な移動コストは、道路を敷設することで打ち消すことが可能だ。

シヴィライゼーションのメカニクスは離散的である

シヴィライゼーションを調べていくと、そのメカニクスのほとんどが離散的であることが分かる。このゲームはターン制であり、ユニットや都市の位置はタイル上に制限され、攻撃力と防御力は整数で表されている。これらのメカニクスは離散的であるので、個々のメカニクスは簡単に理解することができる。基本的に、プレイヤーはこれらのメカニクスがもたらす効果を暗算することができる。それでも、シヴィライゼーションの確率空間は巨大である。シヴィライゼーションは、比較的シンプルな離散メカニクスを用いて巨大な多様性を生み出し、戦略的なレベルでプレイヤーがゲームとインタラクトできる、優れた例である。

シヴィライゼーションの全メカニクスを徹底的に説明していくと、軽く一冊の本になってしまうだろう。すべてのユニットの種類と都市のアップグレードの詳細を列挙した場合は特にそうだ。このゲームは、こういった詳細すべてにアクセスできる独自の辞典を備えている。とはいえ、これらすべての要素を理解することは難しくはない。もっと重要なことは、これらの要素間に多くの関係性があることだ。ユニットは都市で生産され、都市は他のことにも使うことができる重要なリソースを消費する。一度ユニットが生産されると、プレイヤーは毎ターンその維持費としてゴールドを支払う必要があることが多い。道路の敷設にも時間とリソースの投資が必

第2章 創発型と進行型

要になるが、それによりプレイヤーは手持ちの軍隊をより効率的に展開することができるため、維持すべき軍隊の規模を削減できる。また、競争相手のユニットよりも自軍のユニットを強力にするために、新しいテクノロジーの研究に投資をすることができる。要するに、シヴィライゼーションのあらゆるものは、ほとんど、他のすべてのものとつながっている。これは、プレイヤーがとる選択は、多くの効果をもたらし、ときに予想外の効果を引き起こすことを意味する。早い段階で強力な軍隊を構築することで、プレイヤーはマップの大部分を占拠することができるが、他の分野の開発が大きな負担になり、長期的には後退する可能性がある。プレイヤーを取り巻く近隣の文明が行った選択が、プレイヤーの戦略の有効性に影響を及ぼすため、さらに複雑度が上がるのである。

シヴィライゼーションのゲームプレイには数多くの異なる戦略があり、プレイヤーはゲームの進行にあわせて戦略を切り替えなければならないことが多い。初期の段階では、プレイヤーの文明をすばやく拡大できるよう、領地を占領していくことが重要になる。また、領地の拡大によって、序盤のうちに重要なリソースを確認し収集することができるので、迅速にテクノロジーを開発する助けにもなる。他の文明と遭遇したら、攻撃をしかけたり、同盟を組むことができる。ゲームが初期段階のうちは、他の文明を完全に征服するほうが簡単だが、ゲーム後半になると、これは非常に困難なものになっていき、他の戦略のほうがうまく働くようになる。プレイヤーの文明は裕福であるが、隣接する文明はそうでない場合、プレイヤーは近隣都市を説得して自分の国土に参加させるような文化的な攻勢を開始することができる。このゲームは一般に特定のフェーズ——初期の領土拡張、経済発展の投資、暴力的衝突、最終的な宇宙への技術や生産の競争——を経て進行する。シヴィライゼーションの設定では、こうした戦略やゲームフェーズのすべてが、そのメカニクスから極めて自然に創発するのである。

シヴィライゼーションのゲームプレイフェーズ vs 時代と黄金期

シヴィライゼーションでは、プレイヤーの文明はゲーム内のさまざまな時代を経ることで進化していく。それは石器時代から始まり、最終的に中世、ルネサンス時代、そして現代へと発展していく。このゲームはこれらの時代の概念を使って、プレイヤーの文明のグラフィックスや表現を、その進行に調和させている。プレイヤーを新しい時代へ移動させるトリガーとなるものはかなり恣意的なものである。このトリガーは、探索、開発、紛争といったさまざまな戦略フェーズのゲームメカニクスから創発するものではない。歴史時代は視覚に訴える色彩を提供する、ほとんど表面的な追加物である。黄金時代と呼ばれる。プレイヤーの生産量を20ターンにわたって増加させることができるメカニズムは、ゲームプレイフェーズと歴史時代の間に位置する。黄金時代のトリガーとなるイベントは、時代の遷移のトリガーとほとんど同じくらい恣意的なものである。しかし、プレイヤーはこれらのイベントをコントロールして、目的に応じて黄金時代へのトリガーを狙って引くことができる。黄金時代はゲームプレイから創発するものではないが、ゲームプレイフェーズに影響を及ぼしている。

あなたがシヴィライゼーションのようなゲームメカニクスのデザインを依頼されているところを想像してみてほしい。あなたならどうやってその仕事にアプローチするだろうか？ おそら

2.4 進行型ゲーム

く多くのさまざまなイテレーションとプロトタイプを繰り返して、そのメカニクスのデザインと調整を行う必要があるだろう。賢い人なら、すべての要素を可能な限りシンプルに保って、その一方で、要素間に複数の関係性を持たせるだろう。このやり方で、ゲームが複雑になるのは確実だ。しかし、面白いゲームプレイになるかどうかの保証はほとんどない。失敗しないためにも、これらのメカニクスの構造を意識していく必要がある。構造によって、創発型の振る舞いを引き起こすものとそうでないものがある。ゲームメカニクスの中でもフィードバックループのような構造は、創発型の振る舞いを生み出す良い方法である。このフィードバックがさまざまなスケール（規模）とさまざまな速度で機能している場合は特にそうだ。今の時点では、これはおそらく漠然とした話にしか聞こえないだろう。本章と後半の章で、私たちはこれらの構造とフィードバックについてより詳細に掘り下げていく。

![メモアイコン]

ゲームデザイナーが、あるゲームメカニクスが他のゲームメカニクスに影響を与える、もしくは制御するように設定するさまざまな方法がある。私たちは「構造 (structure)」という言葉と、そのさまざまな方法を指す言葉として使っている。フィードバックループは構造であり、特定の条件が満たされたときにあるイベントを引き起こすトリガーもそうである。

2.4 進行型ゲーム

ゲームにおける創発は重要ではあるが、プロのゲームデザイナーなら進行のメカニズムを無視することはできない。ゲームプレイをかき立てるためのストーリーを持っているゲームは多く、さまざまなレベルを通じて物語が展開するのが一般的だ。個々のレベルには、通常、明確に定義されたミッションがあり、それがプレイヤーのゴールとそのレベルをクリアするために達成しなければならないタスクを構造化している。デザイナーは、ゲームがプレイヤーに一貫した体験を作り出せるように、ゲームとそのレベルを計画すべきである。多くの場合、これが意味するところは、プレイヤーがゲーム内を移動する方法を、ゲームデザイナーがさまざまなメカニクスで制御することを意味する。本書では、このようなメカニクスを**進行のメカニクス**と呼ぶ。進行のメカニクスを理解することは、興味深くインタラクティブなストーリーを伴ったすばらしいレベルとゲームをデザインする鍵となる。

進行のメカニクスは、プレイヤーが最初に出会うゲーム要素、ゲーム開始時のリソース、進行するのに必要なタスクなどをゲームデザイナーが決定する上で鍵を握るメカニクスである。ゲームデザイナーは、プレイヤーが持つ能力（アビリティ）を決定し、ゲーム内のプレイヤーの進行を制御するため、ロック、キー、重要なパワーアップなどを巧妙に配置するレベルのレイアウトを行う。これによって、プレイヤーはすんなりとゲームに入り込めるようになる。プレイヤーはゲーム空間を探索し、能力を獲得して腕前（スキル）を磨いていく。そして最終的には、レベ

第2章 創発型と進行型

ルに仕組まれたイベント、ゲームを通して発見される手掛かり、特定の場所で引き起こされるカットシーンなどで構成されたストーリーのような体験を得るのである。

学術論争（アカデミックな戦い）

ストーリーとゲームのトピックは、ゲーム研究の分野では2つの陣営に分かれた激しい議論の対象となっている。一方の物語学の陣営 (narratologists) は、ゲームを他の物語媒体の系統に加え、ゲームのストーリーテリングの側面に焦点を当てている。もう一方の、遊びやゲームを意味する Ludus を研究するルドロジー陣営 (ludologists) は、ゲームを理解するために、何よりもまずゲームメカニクスとゲームプレイに着目することから始めるべきだと主張している。ルドロジー陣営にとって、ゲームのストーリーはゲームに不可欠な要素ではない。Angry Birds はその Y 良い例である。Angry Birds にはストーリーがあるが、そのストーリーはレベルの幕間に語られるのみで、レベルの中の出来事はストーリーとは関係がない。ストーリーとゲームプレイは、互いに何の影響も与えない。Angry Birds の場合は、ルドロジー陣営は正しいということになるが、努めてストーリーとゲームを統合しているゲームもある。ロールプレイングゲームやアドベンチャーゲームなどのジャンルは特にそうだ。本書で、ゲームにおけるストーリーテリングについて語るとき、ゲームプレイのための単なる表面的な文脈以上のものを提供する、統合されたストーリーのことを意味している。

チュートリアル

ゲームデザイナーは、チュートリアルやレベルデザインの作成に進行のメカニクスを適用して、プレイヤーにゲームをクリアするために必要なスキルをトレーニングさせる。最近の市販ビデオゲームは、ルールやインターフェイスの要素、ゲームプレイの選択肢の数がたいていいや多く、大半のプレイヤーが一度にすべてを把握することはできない。インターネット上で見かける比較的規模の小さなゲームでさえ、その多くが、プレイヤーに多くのルールを学び、多様なオブジェクトを認識し、さまざまな戦略を試してみようことを要求している。こうした要素を同時にプレイヤーの眼前に晒せば、プレイヤーは圧倒されてしまい、他のゲームへ流れていくだろう。これらの問題に対処する最善の方法は、ルールを扱いやすいまとまりに分けた上で、それらを順を追ってプレイヤーに教えていくよう、レベルをデザインすることだ。初期のチュートリアルレベルでは、プレイヤーは、ほとんどエラーが引き起こされることのない安全で制御された環境下で、ゲームプレイオプションをいろいろと試すことができるようにする。

ナラティブアーキテクチャ

チュートリアルとレベルデザインを用いてプレイヤーのトレーニングができることは、ビデオゲームの強みの1つである。チュートリアルとレベルデザインは、ゲームのシミュレートされた物理空間を使って、ゲーム体験を構造化しているのだ。時系列に出来事を連ねていくのによく通じている文学や映画とは異なり、ゲームは出来事（イベント）を空間に配置するのがふさわしい。ヘンリー・ジェンキンス (Henry Jenkins) は、彼の論文 “Game Design as Narrative Architecture” において、この種の空間的なストーリーテリング技術をナラティブアーキテクチャ (narrative architecture) と呼び、ゲーム空間的物語の系統に位置づけ、伝統的な神話や英雄伝だけでなく、近現代の J・R・R・トールキンの作品にも並ぶものとした (Jenkins 2004)。端的に言えば、ゲーム空間を旅していくことによってストーリーが語られるのだ。

2.4 進行型ゲーム

ゲームにおけるストーリーテリング

ストーリーテリングを活かして、大きな効果を上げているゲームは多い。中でもハーフライフシリーズは傑出した例である。このファーストパーソンシューターのアクションゲームシリーズでは、プレイヤーは広大に見えて、実は狭い経路に絞り込まれた仮想世界を冒険する。ハーフライフのストーリーはすべてゲーム中に語られる。プレイヤーをゲームから引き離すようなカットシーンは一切なく、すべての対話はゲーム内のキャラクターによって行われており、耳を傾げるか無視するかはプレイヤーの選択に委ねられている。ハーフライフはプレイヤーのためによく構造化された体験を作り出し、ゲームを通じてプレイヤーを導く手法を美学の域まで完成させた。このプラクティスはしばしば**レールローディング** (railroading；列車旅行) と呼ばれる。ハーフライフとハーフライフ2はプレイヤーが列車で到着するシーンから始まるが、この観点から言えば、それは偶然ではないのかもしれない(図2.2 参照)。レールローディングの欠点は、プレイヤーの自由度の大部分が幻影にすぎないことだ。ゲームの意図していない方向へ進むと、この幻想はいとも簡単に打ち砕かれる。レールローディング型のゲームでは、プレイヤーがあさっての方向を探索しないように目に見えない境界を張りめぐらせ、さらにその存在をプレイヤーに悟らせないというデザイン技術が多く用いられている。

[図2.2の画像：列車内の通路、赤いシートが並ぶ車両の中を二人の人物が歩いている]

図2.2 ハーフライフ2ではプレイヤーは列車に乗ってゲームに到着した後、別の意味でそのレールから脱線することはない。



図2.2 ハーフライフ2ではプレイヤーは列車に乗ってゲームに到着した後、別の意味でそのレールから脱線することはない。

図5 図2.2 ハーフライフ2の列車シーン

第2章 創発型と進行型

ゲーム用のインタラクティブなストーリーを作成することは容易ではない。ストーリーを枝分かれさせるなどの伝統的なテクニックは、非効率であることが分かっている。プレイヤーが1回のプレイスルーでは体験しきれない、大量のコンテンツを作成しなければならない。The Elder Scrolls シリーズの多くがそうであるように、プレイヤーが冒険できる広大なオープンワールドを作れば、高い自由度をプレイヤーに提供できるが、プレイヤーがメインのストーリーラインを見失ってしまうことも多い。一貫したストーリーライクのゲームを作るには、プレイヤーに提供する自由度と、レベルデザインによるその自由度の制限という、2つの間の微妙なバランスが必要となる。

例：ゼルダの伝説

ゼルダの伝説シリーズにおけるほぼすべてのゲームとレベルが、進行型ゲームの良い例となっている。ゲームにおいて進行のメカニズムがどのように機能するのかを詳しく例示するために、ここでゼルダの伝説トワイライトプリンセスの「森の神殿」レベルを検証してみよう。このレベルでは、プレイヤーはゲームの主人公リンクをコントロールし、森の神殿にはびこる魔物にとらわれた8匹のサルを救出に向かう。このミッションは、8匹のサルを解放、中ボス（影の力に操られたサルの親玉ワーク）の打倒、「疾風のブーメラン」の発見とその習得、ボス（覚醒寄生種バラント）の討伐で、構成されている。図2.3は、森の神殿レベルのマップを示している。図2.4は、プレイヤーのタスクとその相互関係をチャートにまとめたものである。レベルのゴールに到達するためには、リンクは最後の戦闘でボスと対決しなければならない。この戦いにたどり着くには、リンクは鍵を見つけ出し、4匹のサルを救出しなければならない、そのためには疾風のブーメランが必要であり、ブーメランを手に入れるためには中ボスを倒さなければならない、と続くのである。一部のタスクは好きな順序で実行することができ、リンクがサルを解放する順番は特に問題とされない。他のタスクの実行は任意だが、行えば良いリワードが得られる。

ゼルダは純粋な進行型ゲームではない

ゼルダの伝説シリーズのゲームはすべて、創発型ゲームプレイと進行のメカニズムを組み合わせている。たとえば、戦闘は多くの創発メカニクスを特徴としており、プレイヤーはさまざまな戦闘テクニックを学んでマスターし、どの戦略がどの敵に対して一番効果があるのかを自分で見極める必要がある。すでに述べたように、純粋な進行型ゲームは今日においては非常に稀な存在である。しかし、ゼルダの伝説シリーズは、強力な進行のメカニクスを有している。つまり、レベルと長いストーリーラインを実に巧みに設計してゲームプレイ体験を構造化しているのだ。そのため、進行型ゲームの最高の例となっている。

森の神殿レベルのミッション構造には、いくつか際立ったフィーチャーがある。ひとつは、中ボスとの戦闘とブーメラン探しミッション中盤のボトルネックとなっていて、そのボトルネックの前後として2つのボトルネックの前後として2つの選択のセクションが配置されていることである。ゲーム空間は、ハブ&スポーク型レイアウトを特徴とし（コラム「ゼルダシリーズにおけるハブ&スポーク

2.4 進行型ゲーム

型レイアウト」を参照)、これによってミッション構造の並行タスクをサポートしている。中央のホール（大蜘蛛と戦う場所）から、プレイヤーは3つの方向に進むことができる。右の通路を行くと、すぐにまた通路が3つに分岐し、捕らえられたサルがいる場所に通じている。正面の通路は中ボスの居場所につながっているが、リンクが最初の4匹のサルを解放して初めて通れるようになる。プレイヤーが疾風のブーメランを取得すると、最初のハブ&スポーク構造の追加空間と、新しいハブに到達できるようになる。

[図 2.3 の画像：森の神殿のマップ（カラフルなゲームマップ図）]

凡例（マップ右側）：- 大蜘蛛- サル- 中ボス- 小さな鍵- ボス- ボスへの鍵- 爆弾虫- ブーメラン- ブーメラン+爆弾虫- ルビー- 扉- ハート- 鍵のかかった扉- 入り口

図 2.3 森の神殿のマップ

と、新しいハブに到達できるようになる。

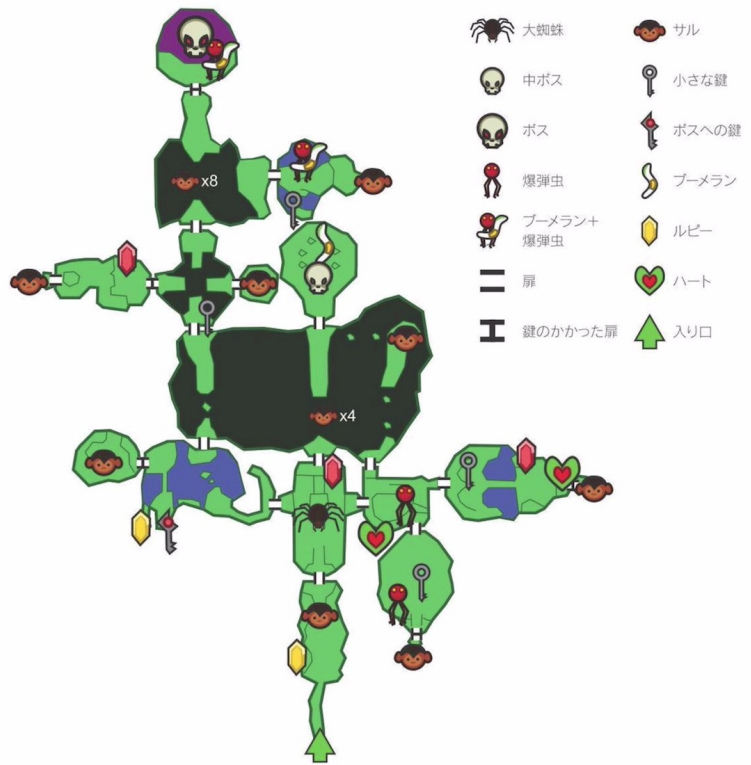


図2.3 森の神殿のマップ

図 6 図 2.3 森の神殿のマップ

第2章 創発型と進行型

[図 2.4 の画像：森の神殿のミッションのチャート（フローチャート）]

チャートの内容（上から下へ）：- サルを解放する- クモを探す-（左分岐）小さな鍵を探す → サルを解放する-（右分岐）爆弾虫を使う → 爆弾虫を使う / 小さな鍵を探す → 小さな鍵を探す / サルを解放する → サルを解放する- 谷間を渡る- ボスサルと戦う- ブーメランを寄還する-（左分岐）ブーメランを使う → ボスへの鍵を探す / 小さな鍵を探す → サルを解放する-（中央）ブーメランを使う → 爆弾虫とブーメランを使う → サルを解放する-（右分岐）ブーメランを使う → サルを解放する- 谷間を渡る- ボスと戦う

図 2.4 森の神殿のミッションのチャート

第2章 創発型と進行型

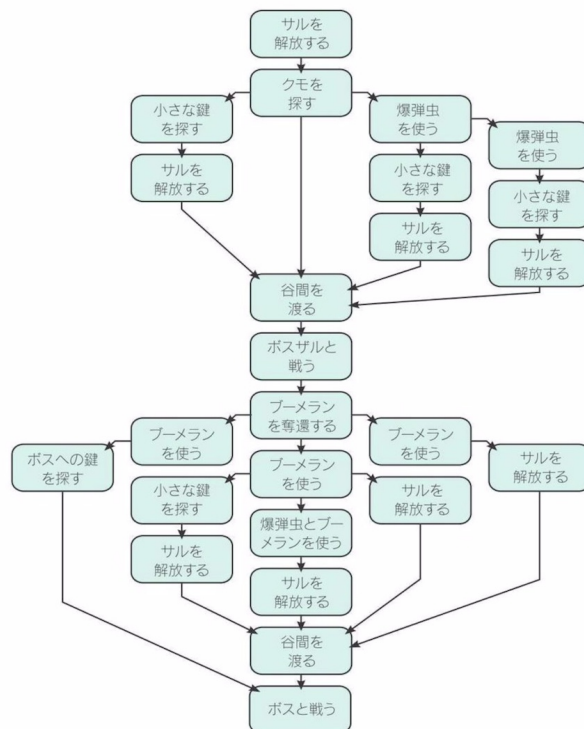


図2.4 森の神殿のミッションのチャート

図 7 図 2.4 森の神殿のミッションチャート

ゼルダシリーズにおけるハブ&スポーク型レイアウト

ゼルダシリーズのダンジョンはハブ&スポーク型でレイアウトされていることが多い。ダンジョン中央の部屋はハブとして機能している。ここからプレイヤーはダンジョンのさまざまな部分「スポーク」に進むことができる。スポークでタスクを達成した後、プレイヤーはたいていハブに戻る。ハブ&スポーク型レイアウトの利点は、プレイヤーが最初に達成するタスクを自分で決められる（そのタスクが難しければ別のタスクを選べる）ことと、ハブがセーブポイントやダンジョンの入口として適した場所であることである。ハブ&スポーク型レイアウトを使用すると、プレイヤーがすでに見つかった場所を後戻りしてしまうことを最小限に抑えられる。ハブ&スポーク型レイアウトの技法に関する詳細は、*Fundamentals of Game Design* の第 12 章を参照してほしい。

2.4 進行型ゲーム

疾風のブーメランはそれ自体が、ゼルダシリーズの代表的なロック&キーメカニズムの良い例である。アシュモア (Ashmore) とニーチェ (Nietzsche) が、彼らの論文 “The Quest in a Generated World” で述べたように、このメカニズムは多くのアクションアドベンチャーゲームで使用されている (Ashmore and Nietzsche 2007)。ロック&キーメカニズムは、ミッションの必須条件を、タスク間の関係を強制する空間構成に変換する手法の 1 つだ。このブーメランは、武器であると同時にさまざまな方法で使用可能なキー (鍵) でもある。ブーメランには、風を起こしてスイッチを入れる機能がある。リンクは、新しいエリアへ行けるように、いくつかの橋を回転させるスイッチを操作する必要がある。レベルのボスのいる最終部屋へ通じる扉のロックを解錠するマスターキーを手に入れるためには、リンクはブーメランを使って、正しい順序で 4 つのスイッチを入れなければならない。ブーメランは遠くにあるものを集めるために使用でき (小さなアイテムや生き物を拾い上げる力がある)、また同時に武器としても使用することができる。これによりデザイナーは、ミッション前半で使用された空間と同じ空間に、(中ボスを倒した後で) ミッション後半の要素を配置することができる。これは、プレイヤーが最初は克服できない障害にぶつかり、正しいキー「ブーメラン」を見つけるまでその状況が続くことを意味している。

ゼルダシリーズにおける離散的メカニクス

ゼルダシリーズは、離散的メカニクスに、物理の継続的メカニクスをミックスしている。ゼルダのゲーム空間は、大半が物理的なチャレンジであるという意味で、継続的である。しかし、ゼルダには非常に多くの離散的メカニクスが存在している。ハートパー上のハート、およびリンクと敵によるダメージは離散的である。特定の敵がリンクに与えるダメージは常に同じ量であり、リンクが敵を倒すために剣で叩く回数も決まっている。同様に、進行を制御するメカニズムも、すべて離散的である。扉のロックを解錠するには小さな鍵が必要になり、谷間を越えるにはそれを助けてくれる一定数のサルが必要になる、などだ。

ゼルダの伝説トワイライトプリンセスの森の神殿のレベルは、この特有のレイアウトとロック&キー要素を使って、英雄伝を自ら体験するようなプレイの軌跡が生み出されるように構造化されている。リンクが神殿に入ると、解放すべき 8 匹のサルのうち最初のサルを見つけるかたちで、冒険へのチャレンジが始まる。この後すぐに、リンクはこのレベルの第 1 ハブを守っている大蜘蛛に遭遇する。この蜘蛛を倒すと、レベル前半のさまざまな場所へアクセスできるようになる。数多くの試練と障害を持ち受ける中で、英雄リンクは新たな友達や敵と対面する。レベル中盤で、リンクはボスザルと対決することになるが、ここで物語は意外な展開を見せる。ボスザルは本当に倒すべき敵ではなかったことが発覚するのだ。リンクは魔法のアイテムである疾風のブーメランとともに脱出し、このブーメランでダンジョンの後半部をアンロックし、最後のクライマックスで本当の敵を打ち倒す武器としても使うのだ。この「英雄の旅路」という構造は、おとぎ話でもアドベンチャー映画でも、いつまでも陳腐化しないのと同じく、リンクの大半の冒険や他の多くのゲームで登場するのである。

第2章 創発型と進行型

敵もトリガーもロックも、ゲームストーリーの次のパートへのアクセスを制御するシンプルなメカニズムとして機能している。進行型ゲームをデザインするには、入念な計画が必要だ。レベルの物理的なレイアウトとその内部のキーアイテムの配置が、プレイヤーが進行方法を制御する最重要ツールであることを忘れてはならない。プレイヤーに向けて、スムーズで一貫した体験を作り出すために、これらの要素を使用すべきである。同時に、プレイヤーがレベルをクリアするために必要なスキルを学び、練習する機会を確実に与えることも必要である。とはいえ、一番大切なことは、攻略できなかった障害を克服することで、プレイヤー自身が自分の進歩を確実に楽しめることである。

![メモアイコン]

主人公がたどる旅路のストーリーパターンを詳細に解説するには紙面が足りないが、もし詳細を学ぶことに興味があるなら、多くのリソースが案内してくれるだろう。なかでも評判の高い書籍はクリストファー・ボグラー (Christopher Vogler) 著、*The Writer's Journey: Mythic Structure For Writers* である (Vogler 1998)。

2.5 構造的な違い

進行型と創発型の違いをより良く理解するために、この2つの異なる種類のゲームプレイを作り出すメカニズムの構造に着目しよう。創発型のゲームの特徴は、わずかなルールしかないことだ。創発型ゲームでは、複雑さは大量のルールによってではなく、このわずかなルール同士の多くのつながりや相互作用によって作られる。この種のゲームで興味深いのは、ゲームルールの複雑度が一定のポイントまで到達すると、飛躍的にゲームプレイの複雑度が上昇することである。私たちはすでに、三目並べと Connect Four の議論において、同様のゲームプレイの複雑度の飛躍を見ている。図 2.5 は、私たちが**複雑度のバリア**と呼んでいる、ターニングポイントを示している。一定のポイントを超えると、ゲームルール同士の相互作用は、**確率空間の爆発**とも呼ばれる効果を生み出す。一般的には、ゲーム内の創発に寄与するメカニズムは、多くの異なる潜在的な状態をゲームに追加する。そして、膨大な確率空間によってゲームは何度もプレイできるようになる。つまり、プレイヤーは何度遊んでも同じプレイにならないと確信できる。何度遊んでも初めてプレイしたときと同様に結果が予測不能であることで、ゲームの魅力が増すのである。

2.5 構造的な違い

複雑度のバリア

[図 2.5 のグラフ：縦軸「ゲームの複雑度」、横軸「ルールの複雑度」。赤い曲線が急激に上昇し、その後なだらかになっている。グラフ右側に「Connect Four（四目並べ）」、低い位置に「三目並べ」と記されている。]

図 2.5 複雑度のバリアは、2 本の点線の間の領域である。

2.5 構造的

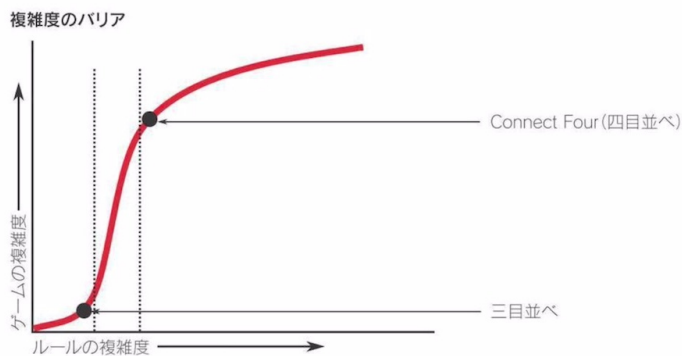


図 8 図 2.5 複雑度のバリア

一般的に、進行型ゲームには多くのルールがあるが、ルール間の相互作用はずっと少ない。レベル内のプレイヤーの進行を制御するメカニクスは、ゲーム中の同様のメカニクスと相互作用することはめったにない。こういったメカニズムの大半が、ただ 1 つの目的を持っている。プレイヤーが、あるタスクを先に達成するまで、特定の場所に到達しないようにすることだ。実際、これらのメカニクスは、2 つの単純な状態のうちのいずれかにしかならない。すなわち、ドアが開いているか閉じているか、キーが見つまっているか否かの、いずれかである。進行に貢献するメカニズムが、ゲームに多くの異なる状態を追加することはめったにないが、ゲームデザイナーからは容易に制御が可能である。進行型の利点は、ゲームデザイナーがプレイヤーの直面するチャレンジやスキル習得の順番を決定し、何よりも重要なストーリーラインに沿って常に増大するチャレンジを統合できることだ。全体的なチャレンジの体験は、創発型でデザインするよりも、進行型でデザインするほうがはるかに簡単である。

典型的な創発型メカニクスによって生成された確率空間と、進行型メカニクスによるそれとは、形状は大きく異なる。創発型ゲームは大きくて広い確率空間を持っている。なぜなら創発型ゲームは数多くの選択肢をプレイヤーに提示し、ゲームの方向性が、しばしばプレイヤーが直接コントロールできない要因（たとえばダイスを振るなど）の影響を受けるからである。これとは対照的に、進行型ゲームの確率空間は小さいが深い傾向にある。ゲームデザイナーにとっては、多くのゲームプレイ上の選択肢をシーケンシャル（順番）に作る —— ただし、各決定ポイントで選択肢はより少なくなる —— のは簡単で、今なお良いアイデアであり、可能性のある結果について十分想定でき、かつ制御できる。これが一般に、進行型ゲームのほうが創発型ゲームよりも長く、かつ一貫したストーリーを届けることができる理由だ。創発型ゲームは、ボードゲームのチェッカーなどのように、プレイ時間が短い傾向がある。プレイ時間の長い創発型ゲームでは、プレイヤーのゲーム序盤の小さなミス 1 つで、数時間後に勝ち目がなくなるようなデザ

第2章 創発型と進行型

イン上の欠陥を持ったゲームを作ってしまうリスクを冒すことになるだろう。X - COM: UFO Defense は多くの点で優れたゲームではあるが、この傾向が現れている。

創発型メカニクスは、大きな確率空間を作る点において効率的である。進行を制御するメカニクスは、その反対に、任意の時点でプレイヤーの選択肢を制限することにより、確率空間を狭めるのである。デザイナーは、進行メカニクスによってプレイヤーの体験を入念に構造化でき、上手に作られたストーリーを届けることができるのである。また、進行型メカニクスは、プレイヤーがまだ準備できていないチャレンジに直面してしまわないよう、ゲームの難易度を調整することも可能だ。表 2.1 に、両者の相違点をまとめた。

表 2.1 創発型メカニクスと進行型メカニクスの構造上の違い

構造	創発型メカニクス	進行型メカニクス
ルール数	低	高
ゲーム要素数	高	低～高
要素間の相互作用	高	低
確率空間	大、幅広	小、深い
リプレイバリュー	高	低
ゲームシーケンスへのデザイナーの制御	低	高
ゲームの長さ	比較的短い（シヴィライゼーションは稀な例外）	比較的長い
学習曲線	比較的けわしい	比較的ゆるやか

2.6 創発型と進行型の統合

創発型と進行型は、ゲームにおいてチャレンジを作る 2 つの異なる方法と考えられているが、多くのゲームは両方の要素を兼ね備えている。創発型と進行型を統合することで、デザイナーは、創発型の自由でオープンなプレイと、進行型の構造化された良いストーリーライクな体験を組み合わせようとしているのだ。進行型は通常、ストーリーテリングに使用されるが、創発型ゲームのようにプレイヤーの行動が自由度を持っている場合、一貫したプロットを作り上げるのは難しい。実際には、この 2 つが交番に使われることが多い。つまり、創発型のレベルやミッションによって、レベル間の小さなストーリー進行への扉が開かれ、その後に別の創発型レベルが続く、といった具合だ。グランド・セフト・オートシリーズがその良い例である。これらのシリーズタイトルでは、プレイヤーは実にさまざまな手段でミッションをクリアすることが可能だが、プレイヤーが選んだゲームプレイの選択肢は実際にはストーリーに影響を与えることはなく、ストーリーはミッションとミッションの間でのみ展開する。今までのところ、プレイヤーがこの 2 つの異なる構造を同時に体験できるようなかたちで統合に成功したゲームは、それほど多くない。これには多くの理由がある。

2.6 創発型と進行型の統合

- ビデオゲームは、まだ比較的新しいメディアである。こうした問題すべてが解決できることを、まだ誰も期待していない。
- ノア・ワードリップ・フルイン（Noah Wardrip-Fruin）が主張するように、創発型と進行型それぞれのメカニクスの洗練度合いには差がある（コラム「ゲームとストーリーのメカニクスにおけるミスマッチ」を参照）。創発のメカニクスは進行のメカニクスと比較して近年より高度かつ急速に進化を遂げてきた。
- これまで、ゲームメカニクスとは何であるか、またどのように構造化すべきかについての確固たる正式な理論がなかったため、このような問題に取り組むことが難しかった。本書の目標の1つは、ゲームメカニクスをデザインする方法論的アプローチを提示して、その手法を用いてこの種の問題に対処することである。

なお、ビデオゲームのまだ短い歴史の中で、この2つの構造を組み合わせる方法を考えついた興味深いゲームがいくつかある。比較的最近の例のひとつを見てみよう。

ゲームとストーリーのメカニクスにおけるミスマッチ

ノア・ワードリップ・フルインは、著書 *Expressive Processing* において、ゲームのインタラクティブなストーリーを支配するメカニクスは、運動や戦闘、その他ゲームの（物理）シミュレーションのメカニクスほどには進化を遂げていないと述べている（Wardrip-Fruin 2009）。現在、シミュレーションのメカニクスは大幅に進化し、細部にまで及んでいるが、ストーリーを通じたプレイヤーの進行は、マイルストーンの役割を果たすボトルネックやゲートをいくつか設定して、追いかけてさせているだけにすぎない。プレイヤーがマイルストーンに関連付けられているタスクを満たすごとに、ストーリーは一步前進する。彼が指摘したように、こうしたストーリー進行メカニクスの基礎形状は、他のメカニクスのそれに比べると、あまり興味を引くものではない。

実例：StarCraft から StarCraft 2 へ

StarCraft の第1作は、創発型ゲームの優れた例である。StarCraft は、リアルタイムストラテジーというジャンルを決定づけたゲームだ。シヴィライゼーションと同様、個々のゲーム要素はかなりシンプルなものであるが、要素間に数多くの相関関係があって、数多くの興味深い創発型の特性を有するゲームメカニクス体系を築き上げている。シングルプレイヤーキャンペーンには30のミッションがあり、基地の構築、リソース管理、攻撃部隊の構築とアップグレードを行って、敵を全滅させる。シングルプレイにおける各レベルの進行はどれもほぼ同じだ。これが予測可能であるために、各レベルのストーリー性が失われている。

StarCraft はレベルの前後でもストーリーが語られている。いろいろな意味で、StarCraft はその当時のほとんどのゲームよりもドラマチックな物語（ナラティブ）を持った、ゲームにおけるストーリーテリングの好例である。実際には、ストーリーラインは古典的な悲劇に似た構造に倣っているが、ゲームにおいては明らかに珍しい作りである。とはいえ、ストーリーはコアなゲ

第2章 創発型と進行型

ームプレイを包むフレーミングデバイス^{†1}にすぎない。ミッションを完了しない限りストーリーを進められないという仕様以外、プレイヤーの行動や選択肢は、そのプロットに何の影響も及ぼさない。StarCraft のストーリーはゲームのコンテキストやモチベーションを提供するが、ゲームプレイの一部として統合されていたものではない。

その後10年以上を経て StarCraft 2 が登場したとき、ストーリーとそのゲームへの統合は、最大の変更点となっていた。StarCraft 2 は、第1作のコアメカニクスについてはほとんど変更をしていない。ここでも基地の構築、リソース管理、攻撃部隊の構築とアップグレードが基本だ。しかしシングルプレイヤーキャンペーンのミッションは、第1作よりもはるかにバリエーションが豊かになっている。たとえば、「The Devil's Playground（悪魔の遊び場）」レベルでは、プレイフィールドの下層エリアは定期的に溶岩に浸食されて、捕らえられたものはすべて破壊される（図2.6参照）。このミッションの目的は、敵の基地を叩くことではなく、この過酷な環境下でサバイバルし、その間に多くのリソースを収集することである。これは、第1作の StarCraft における典型的なミッションのものとは異なるリズムや進行を作り出している。もうひとつの良い例は、「The Evacuation（撤退）」と呼ばれる初期ミッションである。このミッションでは、エイリアンに侵略された惑星から脱出しようとしている数多くの民間入植者たちを保護することが、プレイヤーの目的である。目的を達成するために、4つの民間キャラバン隊を護衛し、最寄りの宇宙港の安全地帯まで突破しなければならない。プレイヤーは基地と攻撃部隊を構築していくことになるが、ここではルートの確保と民間人の保護が目的だ。繰り返しになるが、これは典型的な StarCraft のミッションとは異なるプレイ体験を作り出している。StarCraft 2 のシングルプレイヤーキャンペーンでは、第1作のミッションの典型的な展開を踏襲して進行するミッションが見られることは稀である。基地を建設し、慎重にマップを探索し、敵の基地を1つずつ攻撃していくだけでは、もはやダメなのである。StarCraft 2 では、事前にデザインされたイベントやシナリオ（典型的な進行メカニクス）によっていつのまにかかき立てられている自分をふと発見するのだ。結果的に、StarCraft 2 のミッションは、プレイヤーに彼らの戦略とゲームプレイのパターンを始終新しいプレイ環境に適応させながら、はるかに変化に富んだ魅力的なものになっている。ミッションは通り一遍のものではないため、プレイヤーはよりストーリー性を感じることができる。

^{†1} 訳注：framing device。ここでは、各レベルのゲームプレイを正当化する枠組みを意味する。

2.7 まとめ

[図 2.6 StarCraft 2 の「The Devil's Playground (悪魔の遊び場)」]

StarCraft 2 では、プレイヤーがゲームの大まかなストーリーラインを制御できる。ある程度ミッションの順番を選べるし、ときには 2 つの選択肢から 1 つを選ぶこともできる。これにより第 1 作よりも全体的なストーリーラインをゲームにうまく統合しているが、個々のミッションレベルの進行と創発の統合はオリジナル版のほうが高度であった。

2.7

まとめ

本章では、創発型ゲームと進行型ゲームのカテゴリーを掘り下げた。この 2 つのカテゴリーは、ゲームプレイとチャレンジを生み出すための 2 つの選択肢として、ゲーム研究の分野でも頻繁に使用されている。ゲーム研究者やある種のゲームデザイナーは、進行型ゲームよりも創発型ゲームを好む傾向があるように思われる。この傾向は、ゲームに創発するより興味深い構造や、創発メカニズムによって作られる確率空間の大きさが要因である可能性がある。

創発型ゲームは、比較的少数のルール、ゲーム要素間の多数の相互作用、および大規模で広範な確率空間が特徴である。一方、進行型ゲームは、ルールが比較的多く、ゲーム要素間の相互関係は少なめで、通常狭くて深い、小さめの確率空間が特徴である。

第2章 創発型と進行型

現在のビデオゲームには、創発型ゲームと進行型ゲームの両方の要素が含まれている。しかし、プレイヤーが創発と進行の両方を同時に体験できるような統合は、容易ではない。これには、両方のメカニクスを作り出している構造について深く理解することが不可欠だ。本書では、メカニクスを考察するための、より構造化された手法を提示する。後の章では、創発型と進行型の統合について再び取り上げ、これらの手法を用いて統合時の問題に光を当てることにする。

2.8

演習

チェスのゲームには、一般的に、序盤、中盤、そして終盤の3つのフェーズがあると見なされているが、それでも、ルールはゲームを通して変化しない。この現象はチェスのルールそのものによる創発の特性であり、進行のメカニズムによって人為的に構造が作られたわけではない。

1. チェスのように異なるゲームプレイの段階を経て進行するゲームを挙げてみよう（調べられる範囲で、ビデオゲームと同様にテーブルゲームも考慮に入れるべきである）。2. この現象は何によって引き起こされているか、その理由を答えてみよう。3. そのゲームプレイの異なるフェーズは、本当に創発型だろうか？あるいはそれらはあらかじめデザインされたシナリオや任意のトリガーの結果だろうか？

第 3 章

複雑系と創発型の構造

第 1 章「ゲームメカニクスのデザイン」では、ゲームメカニクスによってどのようにゲームプレイが発現するかについて解説した。第 2 章「創発型と進行型」では、創発型ゲームのメカニクスには特定の構造があり、その構造によって比較的シンプルなルールから多くの異なるゲームプレイ状況が生まれていることを示した。これは一般に、創発型ゲームはリプレイバリュー（再プレイする価値）が高いことを意味する。本章では、創発性とゲームメカニクスの構造、そしてゲームプレイの関係をさらに掘り下げて解説する。ゲームプレイが発現するためには、そのメカニクスが秩序と無秩序（カオス）とのバランスをとる必要があることを検証していく。秩序と無秩序のバランスは容易に崩れるため、このバランス調整はデザイナーにとっての課題となる。実際、創発型をデザインするというタスクはそれ自体が矛盾するパラドックスのようなものだ。なにしろ、創発型の決定的な振る舞いとは、システムが動き出すと自然発生することを特徴としているからだ。

創発型はゲームの領域に限ったものではない。さまざまな複雑系が創発型の振る舞いを見せているが、こうした複雑系はその多くがすでに研究の対象となっている。複雑系の科学は一般に**カオス理論**として知られており、ゲームとは別の創発型システムを研究する分野である。本章では、複雑系科学における先進的な研究成果を取り上げて、創発型の振る舞いに貢献している複雑系の構造について学んでいくことにする。だがその前に、創発型とゲームプレイの関係を深く掘り下げてみたい。

3.1

ゲームの創発型特性としてのゲームプレイ

ゲームプレイとは、ゲームがプレイヤーに提示するチャレンジと、ゲーム中にプレイヤーが行うアクションであると定義した。大半のアクションはプレイヤーがチャレンジを切り抜けるためのものだが、中にはチャレンジとは関係のないアクションもある（レーシングカーの色を変えたりチャットをしたりするアクション）。チャレンジに関連するアクションはゲームメカニクスによって制御される。たとえば、アバターがジャンプできるのはジャンプメカニクスが実装されているときのみだ。

チャレンジとその解決方法については、プログラム上、各チャレンジに固有の解決方法を用意することも可能である。第2章で解説したとおり、テキストアドベンチャーゲームなどの古典的な進行型ゲームがそうだ。つまり、1つのチャレンジがそれ自体ユニークなパズルであり、パズルごとにそれを解決するユニークなアクションを持たせている。とはいえ、大半のゲームでは少なくとも一部のアクションやチャレンジが異なるように作られている、とも解説した。テトリスで、可能性のあるすべての落下ピースの組み合わせをプログラムする人はいない。このゲームではランダムにピースを落としていくだけである。テトリスのチャレンジは、ピースのランダムな落下順序と、それに備えたプレイヤーによる事前の対処アクションとの組み合わせによって生まれている。この組み合わせは毎回異なるものであり、プレイヤーは対面するチャレンジをある程度コントロールできる。このゲームでは、無限に組み合わせられるチャレンジを処理するためにごくわずかなアクションが要求されるのだ。ソリティア（トランプの1人遊び）も同じである。

プレイヤーに予想もしない行動をとらせるメカニズムを実装するゲームもある。ゲームデザイナーのハーヴェイ・スミス（Harvey Smith）は、2001年の記事“The Future of Game Design”で、表現力豊かなさまざまな方法でプレイヤーが行動できるようにゲームシステムを設定する必要性を説いている（Smith 2001）。これを可能にするには、ゲームデザイナーは、個別にチャレンジを事前設計してそれに解法を当てはめるといった考え方を捨て、たとえ奇妙な結果をもたらしても、面白いかたちで組み合わせることのできるシンプルで一貫したゲームメカニクスの方向を目指す必要がある。ロケットジャンプがこの例だ。ファーストパーソンシューターでロケットを爆発させると周辺のオブジェクトに力の作用を及ぼすことが多いが、あるタイトルにおいては、熟達したプレイヤーがこの力を利用してより高く遠くへジャンプする方法を見つけ出したのである。この創発型のプレイヤー行動は、問題ではなくチャンスであるとスミスは言うのだ。彼は、多くのゲームが、表現型システムが許容する自由や創造性を中心にデザインされるべきだと主張している。

リアリズムより一貫性

ロケットジャンプは、意図せずに生まれた風変わりなゲームプレイの例だが、非現実的でありながら実に楽しいプレイだ。これはスティーブン・プール（Steven Poole）が彼の著書 *Trigger Happy* において、ゲームでは現実的であるよりも一貫性を持つことのほうが重要である、と述べたことを実証している（Poole 2000）。ゲームを遊ぶとはゲームメカニクスによって作られた人工の世界に熱中することだ、とプールは述べている。プレイヤーは完全に現実的なメカニクスを望んでいるわけではない。現実的な F1 のレーシングゲームを例にとってみよう。レースに出場するには何年もの練習が必要だが、それは大半のプレイヤーにとって面白いものではない。プレイヤーは、スペースシューターに「スターウォーズ」のブラスターのような武器を期待しているのであって、光のスピードで飛び、撃ち抜かれるまで目に見えない本物のレーザーを期待しているわけではない。プレイヤーは現実の世界では不可能で危険なことをするためにゲームで遊ぶ。ロケットジャンプのようなおかしな効果もその楽しみの一環である。とはいえ、プレイヤーはゲームメカニクスに一貫性を求めている。強力な敵を殺した現実的なロケットが薄い木製の扉を破壊できないなど、メカニクスに一貫性がないと、プレイヤーはフラストレーションを感じるものだ。

創発型ゲームで何度もプレイを楽しめるのは、チャレンジや、おそらくアクションもプレイをするごとに違うからだ。どのセッションもゲームとそのプレイヤーとの協力の結果がユニークである。だが、ルールを見ただけではそのゲームから面白いプレイが発現するかどうかを判断するのは非常に難しい。第2章で解説した三目並べと Connect Four（四目並べ）の違いから明らかになったように、ルールをたくさん増やすだけでは創発型ゲームプレイを生み出すことはできない。ルールの複雑度とゲームの振る舞いの複雑度との関係は単純な比例関係にはない。ルールを多くするほどゲームが面白くなるわけではないのだ。むしろ、本当に面白い創発型ゲームプレイを生み出すためには、ルールの数を減らしたほうが効果的な場合がある。

秩序と混沌のはざま

複雑系（コラム「複雑系とは何か」を参照）の振る舞いは、秩序と混沌のはざまの秩序だった度合いで分類できる。秩序系は単純に予測できるが、複雑系は、構成するパーツの機能を完全に理解していても予測することは不可能である。創発とは、秩序と混沌のはざまのどこかで力強く生まれてくるものだ。

複雑系とは何か

本書で扱う複雑系とは、理解するのが難しいシステムという意味ではない。ここでは「複雑」という言葉は、多くのパーツで構成されたシステムを意味するために使われている。複雑系の科学が研究対象としているシステムに多く見られるように、これらのパーツは理解するのも個別にモデル化するのも極めてシンプルである。こうしたパーツが寄せ集められると、大半の複雑系は、個々のパーツを見ただけでは説明の難しい、予測できない驚くべき振る舞いを示す。複雑系の科学の文献ではゲームがその典型として挙げられている。このようなゲームでは個々のルールはシンプルで理解しやすい傾向にあるが、ゲームの結果は予測できない。本書では、ゲームの個別のパーツとゲーム全体の振る舞いとの関係を掘り下げていく。

第 3 章 複雑系と創発型の構造

秩序と混沌の両極端の間には周期系と創発系の 2 つの段階がある（図 3.1）。周期系は連続的かつ予測が容易な順番で明確に決められた数の段階を経て進展する。大規模なものでは気象系や季節の循環の振る舞いがこれに相当する。地球上のどこにいるかによって、1 年の季節の数が決まる。位置によっては季節サイクルが厳密であり、季節の開始日が毎年決まっている場合がある。季節による温度変化が多少あったり、季節の開始日が異なったりしても、気象系はたいいていバランスがとれており、同じサイクルを通じて何度も繰り返して進展する（現在、地球温暖化が気象系を変えているように見えるが、永久的な変化であるのか、非常に長いサイクルの一部であるのかについては大きな議論が起こっている）。

秩序	周期系	創発系の振る舞い	混沌
機械時計	季節のサイクル	日々の天候	乱気流
進行型ゲーム	MMORPG のティック	シヴィライゼーションのゲームプレイフェーズ	ダイスの目
スクリプトによる レベルデザイン	モノポリーのラウンド		

→ 複雑度

図 3.1 複雑系の振る舞いの 4 つのカテゴリー

創発系は周期系より秩序だっておらず、混沌としている。創発系は安定したパターンを持った振る舞いを示すが、あるパターンから別のパターンに突然切り替わることがある。気象系がこの好例だ。ある地域に限って言えば季節のサイクルは全体として基準があるものの、特定の年の天候を予測するのは難しい。大気圧、海水温度、気温による複雑な相互作用があるため、次にひどい霜が降りる日にちを言い当てたり、冬の積雪量を正確に予測したりするのはほぼ不可能だ。それでも統計的な基準値に基づいて推測を立て「聖金曜日に豆を植える」といった経験則を作ることはできる。だが、これが毎年正確であるとは限らない。

シンプルな実験で創発系を体験しよう

シンプルな実験によって複雑な振る舞いの 4 つのカテゴリーを体験できる。水道の蛇口が 1 つあればよい（もっともこの実験では蛇口によって結果にばらつきがある）。そっと優しく蛇口をひねると、ある時点で水はゆっくりと一定のスピードでしただり落ち始める。この状態は、蛇口を大きくひねってからゆっくり閉じると簡単に作り出すことができる。開じられた蛇口は、簡単に予測できる周期的な振る舞いを表す。つまり水はまったく流れない。水がしたたっている状態は周期的な状態だ。さて、ここでゆっくりと、もう少し蛇口を開くと、したたる速度がどんどん速くなる。ところが、ある時点に達すると、完全に開いた状態になる直前に水の流れのパターンは不規則になる。この蛇口は、あなたが複雑系を混沌状態に向けて素早く遷移させているのだ。混沌状態と周期的なしただりとの間のある時点で、もっと複雑な周期的パターンを作り出すこともできる。たとえば、水滴を 2 つ落としてからしばらく止めることだ。ときには水滴が落ちるときとゆっくりと不定期に落ちるときとが交互に現れることもある。そして、さらに蛇口を開くと、安定した秩序だった水流状態に素早く戻るのである。

3.2 複雑系の構造的な特徴

ゲームでは振る舞いのパターンを見分けることができるし、同時に複数のパターンが存在するゲームのほうが多い。進行型ゲームは、チャレンジやアクションの潜在的なシーケンスがすべて予測できるため、秩序系である。プレイヤーは次に何が起こるか分らないが、デザイナーならメカニクスを見れば確実に予測できる。ボードゲームでは順番にターンを持たせることで、かなり巧妙な周期系を構築している。多人数参加型オンラインロールプレイングゲーム（MMORPG）でよく使われる離散的な時間単位「ティック」は、プレイヤーの戦略に影響を与える。シド・マイヤーのシヴィライゼーションにおける明確な進行フェーズは、ゲームにおける創発系の振る舞いを明確に示す良例と言えるだろう。それから、ダイスや乱数生成、さらに他のプレイヤーも混沌を生み出す要素となる。創発型ゲームをデザインするには、ゲーム全体の振る舞いが創発型カテゴリーのどこかに該当するように、これらの要素すべてのバランスをとる必要がある。

創発性をデザインすることは可能か

創発性とは、複雑系においてそのシステムが動き出してから生ずるものだ。これが、ゲームデザインがプロトタイプやテストに大きく依存する理由である。ゲームは複雑系であり、ゲームプレイが面白く楽しくバランスがとれているかどうかを判断するには、何らかのかたちでそのゲームを人に遊んでもらうしか方法はない。

通常、デザインとは、作り上げる対象を思い描いてから、それを作り出すためのプロセスであると考えられている。創発系をデザインするのは矛盾した行為だ。なぜならそのシステムが生み出す最終的な状態について明確に把握していないのに、そこにたどり着く体験をデザインすることになるからだ。とはいえ、第1章「ゲームメカニクスのデザイン」で解説したように、ゲームメカニクスの特定の構造によって特定の種類の結果が生み出されている。こうした構造を理解すれば、大量のテストが必要であることに変わりはないものの、デザイナーが自分の求める効果を作り出すことは可能だ。本書では、この構造を明確にしていくことで、読者自身のゲームや他のゲームでこの構造を特定し、求めるゲームプレイを生み出せるようにする。

ゲームにフォーカスを戻す前に、複雑系の科学における典型的な例を見ていこう。

3.2

複雑系の構造的な特徴

複雑系の科学では、通常、膨大で複雑なシステムを取り扱う。気象系が典型的な例だ。こうしたシステムでは小さな変化が時間の経過とともに大きな影響をもたらすことがある。これは**バタフライ効果**として知られている。仮説上では、1匹の蝶が地球上のある地点で羽をはばたくと、これが大気の動きのきっかけとなり、積もり積もって地球の反対側にハリケーンをもたらす可能性があるというものだ。複雑系の科学が研究するシステムにはこの他に、株式市場、道路交通、

第3章 複雑系と創発型の構造

歩行者の流れ、鳥の群れ、天体運動などがある。こうしたシステムはゲームに見られるシステムに比べてはるかに複雑だ。幸いなことに、創発型の振る舞いを生み出すシステムにはシンプルなものも多い。こうしたシステムを研究して、そこから創発型の振る舞いに貢献する重要な構造的特質を導き出すほうが簡単である。

アクティブで相互に連結したパーツ

数学とコンピュータ科学、ゲームとの境界には、**セルオートマトン** (cellular automaton) (単数形では**セルオートマタ** (cellular automata)) と呼ばれる独特の研究分野がある。セルオートマトンとは、直線やグリッド (格子) における空間またはマス目 (セル) の形状を制御するシンプルなルールセットのことだ。各マス目は白か黒かのいずれかである。ルールによってあるマス目が白になるか黒になるか (またはその逆) が決まり、マス目の色が周りのマス目に影響を与える方法が決定する。通常、マス目の色を変更するルールは、そのマス目の現在の色とそれを囲む8つのマス目 (2次元グリッドの場合)、または2つのマス目 (直線の場合) の色に基づいて決められる。

数学者はこのようなルールの集合を、人間が介することなくそれ自体で作動する仮想的なマシンであると考えている。それが、自動装置を意味する**オートマタ**という呼び名が付いた理由だ。

セルオートマトンは与えられた構成のマス目 (あるマス目は白、あるマス目は黒) で開始し、各マス目に対し色を変化させるルールを適用する。ただちにマス目の色が変化するわけではない。まずグリッド内の各マス目をチェックし、変更すべきマス目をマークし、その後すべてのマス目を変化させてから次のイテレーションに進む。こうしてこのプロセスを繰り返すのだ。各イテレーションは**ジェネレーション** (世代) と呼ばれる。

英国の科学者、スティーブン・ウルフラム (Stephen Wolfram) は、創発型の振る舞いを見せるシンプルなセルオートマトンを作り出した。これは複数のマス目を一列に並べたもので、各マス目のグリッド内の状態 (色) は、そのマス目の前の状態と近傍の2つのマス目の状態によって決まる。マス目には2つの状態 (白または黒) しかないため、これによって8つの組み合わせが生じる。図3.2は、可能性のある1つのルールの集合 (一番下) と、その結果となる驚くほど複雑なパターンを示している。これは、前の状態の下にシステムの各新ジェネレーションを出力して作成されている。開始時点では黒のマス目が1つで、他はすべて白である。

図の一番下のイメージは、マス目の色を変更するルールである。一番左のルールは「黒いマス目の両わきが黒の場合、新しいジェネレーションは白になる」。4番目のルールは「白いマス目の左のみが黒の場合、新しいジェネレーションは黒になる」である。

ここにはランダムなルールが1つもないのに、このセルオートマトンがランダムに**見える**独特なパターンを生み出していることに注目してほしい。

3.2 複雑系の構造的な特徴

[図 3.2 スティーブン・ウルフラムのセルオートマトン]

3.2 複雑系

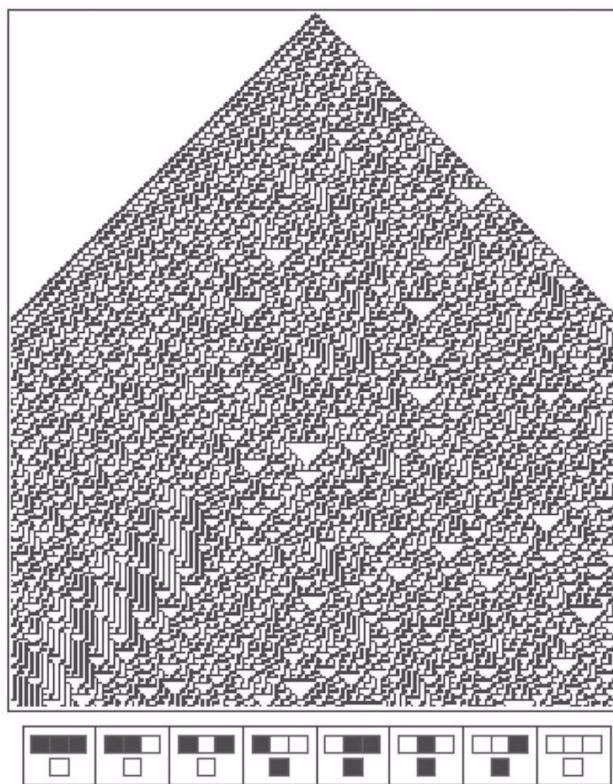


図3.2 スティーブン・ウルフラムのセルオートマトン

図9 図 3.2 スティーブン・ウルフラムのセルオートマトン

スティーブン・ウルフラムは著書 *A New Kind of Science* で自身の研究を詳細に解説している (Wolfram 2002)。彼の膨大な研究によって、動的な振る舞いを示すシステムには、不可欠な特徴が3つあることが分かった。

- ローカル定義のルールを持つシンプルなマス目で構成されること。これは、システムの構成パーツ自体は比較的簡単に記述できることを意味する。ウルフラムのセルオートマトンの例では、8つのシンプルなルールによって個別のマス目の振る舞いが決められている。
- 長期的なコミュニケーションが許可されるシステムであること。複雑系のあるパーツの状態における変化は、空間的にも時間的にも離れたパーツの状態に変化を起こすことができる必要がある。長期的なコミュニケーションこそがバタフライ効果を可能にしているのだ。ウルフラムのセルオートマトンでは、各マス目が近傍のマス目に直接影響しているため、パーツ間のコミュニケーションが行われている。こうした近傍のマス目にはそれ自体にも近傍のマス目が存在するため、システム内の各マス目は互いに間接的につながっている。

第3章 複雑系と創発型の構造

● マス目の活動レベルがそのシステムの振る舞いの複雑度を示す指標になること。活動的なマス目のごくわずかしかなシステムでは、興味深く複雑な振る舞いが創発する可能性は低い。ウルフラムのセルオートマトンでは、活動はマス目の状態への変化として理解できる。つまり、マス目は黒から白または白から黒へ変化するとき活動的である。

興味深いことに、こうした特徴の中には各マス目の振る舞いを制御するルールから「読み取る」ことのできるものがある。各マス目がそれ自身と2つの近傍のマス目から入力を受け取ること自体が長期的なコミュニケーションの可能性を十分に作り出している。つまり、すべてのマス目がつながっているのだ。さらに、図3.2の8つのルールのうち4つがマス目の色を変えるものであるため、このシステムでは活動量が膨大になることを示している。セルオートマトンは、複雑度へのしきい値が意外に低いことを示している。比較的シンプルなルールによって、十分なパーツや活動、コネクションがある限り、複雑な振る舞いを生じさせることができるのだ。多くのゲームがこれに似た手法で構築されている。比較的シンプルなメカニクスによって制御されるさまざまな要素でゲームを構成しているのだ。通常、ゲーム要素間には多くの潜在的なインタラクションが存在する。当然ながら、プレイヤーはシステム内の重要な活動のソースであるが、セルオートマトンが示したとおり、創発性は人間の入力なしに生じる。この特徴を見事に実証するゲームにタワーディフェンスがある(図3.3)。タワーディフェンスは比較的シンプルなパーツの数々で構成されている。敵は事前に定められたルートでプレイヤーの本拠地に向かって進行する。敵には決められたスピードやヒットポイント、そして、敵を特徴付けるいくつかの属性をおそらく持っている。プレイヤーは自分の陣地にタワーを設置して防衛する。タワーは一定の範囲内に入った敵に向かって、一定の間隔で砲撃する。ダメージを与えるタワーもあれば、敵の進撃をスローダウンさせるなど、特別な効果を引き起こすタワーもある。付近のタワーの性能を上げるタワーを備えたゲームもある。タワーディフェンスでは、ローカルメカニズム(敵とタワー)によって定義される多くの要素がある。セルオートマトンのように、これらの要素は活動的で(敵が動き、タワーが敵に反応する)、相互に接続されている(タワーが敵に向かって砲撃し、タワー同士が互いの性能を上げられる)。活動の量と要素間の接続の数は、進行型ゲームから創発型ゲームを見分ける指標となる。一般的な進行型ゲームでは、すべての要素(パズルやキャラクターなど)がプレイヤーアバターとしかインタラクションを持たず、要素間のインタラクションは生じない。しかも、スクリーンに登場しない限り活動的にはならないのだ。進行型ゲームでは、現在スクリーンに表示されていない要素は通常、活動していない。同様に、要素間の接続数も少ない。ゲーム要素は事前設計した限られた手法でしかインタラクトできないのである。当然ながら、これによってデザイナーは進行型ゲームの要素を十分に制御できる。だが、第2章で解説したように予測可能なゲームにもなり、事前設計された選択肢を遊び尽くしてしまうと、楽しみも終わってしまう。

3.2 複雑系の構造的な特徴

図 3.3 タワーディフェンス：Lost Earth HD

フィードバックループでシステムの安定化／不安定化の双方が可能に

複雑系の典型例として生態系がある。生態系は非常にバランスがとれているように見える。つまり、ある生態系内部ではさまざまな動物の個体数は時間を経ても変化がない。さらに、自然はあらゆる種類のメカニズムを使ってこのバランスを保持しているように見える。これは捕食動物と被捕食動物の個体数を見るとよく分かる。被捕食動物が多く生存する場所では捕食動物は簡単に食べ物を見つけることができる。これによって捕食動物の個体数は増える。しかし、捕食動物の個体数がどんどん増えると、被捕食動物の個体数が減る。そして、捕食動物が増えすぎると状況が反転するポイントが存在する。つまり、捕食動物が十分な食料を得られず、その個体数が減り始めるポイントだ。捕食動物の数が減ると、生き残る被捕食動物が増えて子どもを産み、その個体数が再び増加するのである。生態系における捕食動物と被捕食動物の個体数のバランスはまさにフィードバックループに起因している。フィードバックループは、システム内の 1 パーツ（たとえば捕食動物の個体数）の変化の影響が、後になって元のパーツに跳ね返ってできあがるものだ。この場合、捕食動物の個体数の増加によって被捕食動物の個体数が減り、被捕食動物の増加によって捕食動物が再び

第3章 複雑系と創発型の構造

減少する。捕食動物の個体数変化がもたらす効果は、文字通り捕食動物の個体数に「フィードバック」されたのである。あるシステム内でバランスを保持するように機能するフィードバックループは**ネガティブフィードバックループ**と呼ばれる。一般にネガティブフィードバックループは、電子機器の設計に使われる。たとえばサーモスタットだ。サーモスタットは気温を検知し、一定温度より低くなるとヒーターを作動させる。作動したヒーターによって気温は上昇し、それによってサーモスタットはヒーターをオフに戻す。機械の調速機もこの例である。機械の速度が（負荷が下がるなどして）上がると、調速機は機械の出力を落として速度を下げる。機械の速度が落ちると出力を上げて速度を上げるのだ。これによって機械の速度は一定に保たれる。調速機が使われるのは、負荷が急になくとも危険な速度にならないように確実に機械を最適なスピードに保つためだ。ネガティブフィードバックはゲームでも活用されている。たとえば、シヴィライゼーション（図3.4）で、都市の人口は、捕食動物／被捕食動物の例とは異なるネガティブフィードバックによって影響を受ける。都市の発展にともない増加する人口は、食料需要も増加させる。この食料需要の増加効果により、都市はその時点での技術レベルと地理的特徴に合致した大きさまで発展するのだ。

図3.4 シヴィライゼーション V の古代ギリシャの都、テーベの人口と食料供給

3.2 複雑系の構造的な特徴

ネガティブフィードバックループの反対は、当然、**ポジティブフィードバックループ**である。フィードバックループによって生じた変化に対し、反対の現象を引き起こしてバランスをとるネガティブ版とは逆に、ポジティブフィードバックループは、それを引き起こした効果を強化するのである。音響フィードバックが好例だ。マイクが音を拾い上げ、アンプがそれを増幅し、スピーカーが大きな音で再生する。するとマイクはスピーカーからこの新たな音を拾い上げ、それが再び増幅され、これが繰り返されるのである。この結果、高いピッチの金切り声のハウリングが発生し、これを止めるにはマイクをスピーカーから離すしかない。ポジティブフィードバックもゲームで活用されている。たとえば、チェスで相手のピースをとれば、相手より持ち駒が多くなるため、他のピースもとやすくなる。こうして、ポジティブフィードバックによって素早く変化する揮発系（volatile system）を構築できる。本書では全般にわたってフィードバックを解説していく。大半の創発型ゲームでは、さまざまなフィードバックループが同時に働いている。今の時点では、複雑系においてフィードバックループが存在することを覚えておくことが重要である。ネガティブフィードバックループはシステムのバランスをとるように機能するが、ポジティブフィードバックループはシステムを不安定にするのだ。

スケールの違いによってさまざまな振る舞いのパターンが創発する

スティーブン・ウルフラムだけがセルオートマトンを研究した数学者ではない。おそらく最も有名なセルオートマトンはジョン・コンウェイ（John Conway）が発明した「ライフゲーム」だろう。コンウェイのセルオートマトンは2次元グリッド上に並べられたマス目で構成されている。理論的には、このグリッドはすべての方向に無限に広がる。グリッド上の各マス目には8つのマス目が隣接している。つまり、マス目は直交上にも対角線上にも囲まれている。各マス目には2つの異なる状態、すなわち「生か死か」しかない。大半の例では、死んだマス目が白、生きたマス目が黒に塗られている。各イテレーションで各マス目に次のルールが適用される。

- 生きているマス目に隣接した生きたマス目が、1つ以下しかない場合、そのマス目は寂しさで死ぬ。
- 生きているマス目に隣接した生きたマス目が、4つ以上ある場合、そのマス目は過密すぎて死ぬ。
- 生きているマス目に隣接した生きたマス目が、2〜3つの場合、そのマス目は生き残る。
- 死んでいるマス目に隣接した生きたマス目が、3つの場合、そのマス目は生き返る。

ライフゲームを始めるには、グリッドを用意し、初期状態で生きているマス目数を決める。これらのルールを適用して発現した効果の例を図3.5に示す。とはいえ、ライフゲームの創発型の振る舞いの真価を味わうには、オンライン上で入手できるインタラクティブ版を自分で試すことを推奨する。

第3章 複雑系と創発型の構造

[ヒント] オープンソースでクロスプラットフォーム版のライフゲームを <http://golly.sourceforge.net> からダウンロードできる。英語版 Wikipedia の「Conway's Game of Life」には、オンラインで入手できるバージョンのリンクがいくつもある。

図 3.5 ライフゲームのイテレーションの例

ライフゲームが動き出すと、通常、元の生きたマス目の爆発的な活動によって極めてカオス的な結果になる。よく観察されるケースとしては、イテレーションを何度も繰り返したあとで、数個からなるマス目のグループが2種類のパターンを交互に描き続ける安定状態に入ることもある。ライフゲームについての初期の研究で問題にされたのは「生きたマス目が無限に増え続けるような初期パターンは存在するか」だった。その後すぐに、意外な振る舞いを見せるパターンが見つかった。こうした振る舞いの1つが**グライダー**である。グライダーは、4 イテレーションで1つずつ隣のタイルに自らを複製する5つの生きたマス目のグループだ。グライダーの効果は、グリッド上を横断する小さな生き物のようだ。さらに、同じ場所にとどまるが、30 イテレーションごとに発射する**グライダー銃** (glider gun) などのもっと面白いパターンも見つかった。

図 3.6 ライフゲームのグライダー

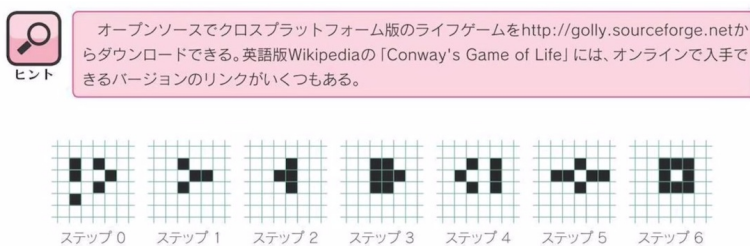


図3.5 ライフゲームのイテレーションの例

ライフゲームが動き出すと、通常、元の生きたマス目の爆発的な活動によって極めてカオス的な結果になる。よく観察されるケースとしては、イテレーションを何度も繰り返したあとで、数個からなるマス目のグループが2種類のパターンを交互に描き続ける安定状態に入ることもある。

ライフゲームについての初期の研究で問題にされたのは「生きたマス目が無限に増え続けるような初期パターンは存在するか」だった。その後すぐに、意外な振る舞いを見せるパターンが見つかった。こうした振る舞いの1つが**グライダー**である。グライダーは、4 イテレーションで1つずつ隣のタイルに自らを複製する5つの生きたマス目のグループだ。グライダーの効果は、グリッド上を横断する小さな生き物のようだ。さらに、同じ場所にとどまるが、30 イテレーションごとに発射する**グライダー銃** (glider gun) などのもっと面白いパターンも見つかった。

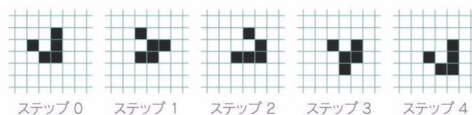


図3.6 ライフゲームのグライダー

図 10 図 3.5/3.6 ライフゲーム

グライダーとグライダー銃で分かるように、複雑系における一番面白い振る舞いは個々のパーツという個別の規模ではなく、グループ全体の規模に起こる。これは他の多くの複雑系にも見られる現象だ。鳥の群れがこの好例である。鳥の群れは1つになって移動する。つまり、このグループは全体として1つの特有の形状(シェイプ)、方向、目的を持っているように見えるのだ(図 3.7)。この場合、鳥の群れを操縦する「ルール」は双方の規模に影響を及ぼしている。また、群れの鳥それぞれに、グループの中心に向かう動作、近くの鳥の速度と方向を合わせる動作、他の鳥に近づきすぎない回避の動作の3つをバランスよく行わせれば、群れのシミュレーションもできる。

3.2 複雑系の構造的な特徴

図 3.7 鳥の群れ

ゲームでも似たような現象が見られる。ここ何年もの間、パックマンのモンスターはプレイヤーの邪魔をするためにわざと徒党を組んでプレイヤーをワナに陥れようとしているのではないかと、との憶測が流れていた。実のところモンスターは協力しているのではなく、彼ら全体としての振る舞いが実際よりもずっと賢く見えるだけだ。パックマンのモンスターは、シンプルなルールに従うシンプルなマシンである。このゲームでは2つの状態が交互に繰り返されている。「分散」状態ではプレイヤーを追いかけることはなく、各モンスターがそれぞれ画面端の角に向かって移動する。だが、ゲームの大半の時間は、プレイヤーを追いかける「追跡」状態になっている。モンスターたちはプレイヤーをとらえるために、迷路の交差点で決断を下さなくてはならない。このアルゴリズムは、モンスターをプレイヤーに接近させる方向を選択するものだ。モンスターとプレイヤーの間に壁があっても、これは無視され、アルゴリズムの挙動に変化はない。モンスターの振る舞いは個別に若干の違いがある。赤いモンスター「プリンキー」はプレイヤーの現在地点に行こうとし、ピンクのモンスター「ピンキー」はプレイヤーのいる地点より4マス先を行こうとし、水色モンスター「インキー」はプレイヤーの現在地点と「プリンキー」の現在地点を組み合わせる行き先を決定し、オレンジのモンスター「クライド」はプレイヤーが遠くにいる場合は追いかけ、近くにくと迷路の左下の隅へ行こうとする。これらを合わせた動きの効果は驚くほど賢く見える。つまり、プリンキーはプレイヤーを追いかけて、ピンキーとインキーはプレイヤーの先を行き、クライドがノイズを加えるのだ。モンスターたちは互いの位置を感知

第3章 複雑系と創発型の構造

していないにもかかわらず、総体として見れば極めて効率的なハンター集団として行動しているのである。このゲームでは、モンスターが協力して追い込んでくるような印象を生み出しているが、実際にはシンプルな振る舞いを組み合わせただけであり、モンスターはお互いを補完し合う戦略を持っているにすぎない。

[ヒント] パックマンのモンスターの振る舞いに関する考察の詳細は <http://gameinternals.com/post/2072558330/understanding-pacmanghost-behavior> を参照してほしい。

創発性の分類

複雑系にはさまざまなレベルの創発性があることが研究で判明している。創発性の度合いが多い効果もあれば少ない効果もある。ある複雑系に存在するフィードバックループとさまざまなスケールを一緒にすると、さまざまなレベルの創発性を説明できる。科学者のヨヘン・フロム（Jochen Fromm）は論文“Types and Forms of Emergence”でフィードバックとスケールを使って創発性を以下のように分類している（Fromm 2005）。最もシンプルな形式では**名目上の創発性**（nominal emergence）ないし**意図的な創発性**（intentional emergence）では、フィードバックはないか、あるいは同階層のエージェント間のみのフィードバックが存在する。たとえば、世の中にある大半の機械装置がこの分類に当てはまる。これらの機械装置では、装置を構成するコンポーネント間で製作者の意図する（設計通りの）創発型の特性を発揮することが期待されている。意図的な創発特性を示す機械の振る舞いは、決定論的であり予測可能だが、柔軟性と適応性に欠ける。調速機やサーモスタットがこの種の予測可能なフィードバックの例である。フロムが第2に分類している**弱い創発性**（weak emergence）では、システム内の異なるレベル間のトップダウン方式のフィードバックがある。この種の振る舞いの例として、フロムは鳥の群れを挙げている。群れの中の鳥は、他の鳥と近さに反応すると同時に（エージェント＝エージェント間フィードバック）、群れをグループとしてとらえている（グループ＝エージェント間フィードバック）。鳥の群れ全体は、個々の鳥とは異なる規模の性質である。鳥はこの双方を知覚し反応しているのだ。この振る舞いは鳥に限ったことではない。魚の群れも同様に振る舞う。群れという行動は、身の回りの環境と全体としてのグループ状態とを認識可能な任意のユニットとして一般化できる。弱い創発性から複雑性を一歩進めると、**複合的創発性**（multiple emergence）を示すシステムになる。このシステムでは、複合的なフィードバックがシステムのさまざまなレベルを行きかっている。フロムによれば、短期ポジティブフィードバックと長期ネガティブフィードバックを持つシステムには面白い創発性が見られるという。このような振る舞いを見せる例に株式市場がある。株価が上がると、人々はそれに気づいて買うようになり、株価はどんどん上がる（短

3.3 ゲームで創発性を活用する

期ポジティブフィードバック)。だが、経験から株価が最終的にはピークに達することも知っているため、ピークに達したと確信した時点でその株を売ろうと計画を立てる。こうして価格は下がっていく（長期ネガティブフィードバック）。この現象は逆方向にも働く。株価が下がりだすとその株式を売り払うが、底値に達したと思った時点で買い戻すのである。ジョン・コンウェイのライフゲームでもこの種の創発性が見られる。ライフゲームにはポジティブフィードバック（マス目の誕生を決めるルール）とネガティブフィードバック（マス目の死を決めるルール）の双方があるのだ。ライフゲームには組織のさまざまなスケールも見られる。つまり、最低レベルには個々のマス目のスケールがあり、組織の高いレベルではグライダーやグライダー銃などの持続的なパターンや振る舞いが見られる。フロムの分類の最後は**強い創発性**（strong emergence）である。主な例として、遺伝系の創発特性としての「生命」と、言語や著述の創発特性としての「文化」の2つを挙げている。強い創発性は、創発性が影響を及ぼすスケール間の違いの大きさと、システム内の中間スケールの存在に起因している。強い創発性とは、最高レベルにおける創発型振る舞いの結果が、システムの最低レベルのエージェントから分離することができるマルチレベルの創発性である。たとえば、ライフゲームでは、高レベルではシンプルな計算を行うコンピュータとして機能させ、そこから新しい複雑系（ゲームなど）を構築できるようにグリッド上にマス目を設定することは可能だ。この場合、コンピュータとライフゲームそのものが示す振る舞い間の因果関係を示す依存性はごくわずかである。こうした分類で分かることは、ゲームにおいてもさまざまなレベルの創発型の振る舞いが、しかも同時に存在することが多いということだ。もっと重要なことは、ゲームメカニクスの構造的特徴（たとえばフィードバックループやさまざまなスケールが存在すること）が複雑で面白い振る舞いの創発において重要な役割を果たしているという事実である。

3.3 ゲームで創発性を活用する

ゲームは予測不可能な結果を生成できる複雑系であるが、その一方で、自然なユーザー体験をもたらすように、設計を十分に練る必要がある。これを達成するために、ゲームデザイナーは一般的な創発型の振る舞いの性質を理解し、さらにゲームにおけるこの性質を特に頭に入れておく必要がある。私たちは、システムとしてのゲームの「構造的な特徴」とは、多くの活動的で相互接続したパーツ、フィードバックループ、さまざまなスケールだと考えている。ゲームでは、このような構造的な特徴が創発型ゲームプレイを生み出す上で重要な役割を果たしている。ゲームメカニクスを研究すれば、こうした構造（他の構造も）を詳細に深く理解することが可能だ。本書の残りの部分ではこの研究に専念していく。

第3章 複雑系と創発型の構造

本章のテーマである3つの構造的特徴は、ゲームにおける創発性そのものを扱う**マキネーション**と呼ばれる応用理論フレームワークを構築する第一の足掛かりでもある。マキネーションフレームワークを使うと、創発型の振る舞いを示す質の高いゲームを構築するという、とらえどころのないプロセスをしっかりとつかむことができる。次の章では内部経済のゲームメカニクスに焦点を当てる。マキネーションフレームワークでゲームメカニクスを視覚化する方法を説明し、そこからメカニクスの構造的特徴を読み取る手法を解説していく。第10章「レベルデザインとメカニクスの統合」と第11章「進行型のメカニズム」では、焦点を変えて大きなスケールでメカニクスをグループ化する方法を示し、それを使って進行型と創発型の双方を用いた興味深いレベルデザインの手法を解説する。

3.4 まとめ

本章では、「複雑系」の定義を示し、そこからどのようにゲームプレイが創発するのかについて解説した。厳密に秩序だったシステムから完全に混沌としたシステムまでの連続体を説明し、創発性がこの両者の間のどこかで起こることを示した。創発性をもたらす複雑系の3つの構造的特徴は、活動的で相互接続したパーツ、フィードバックループ、異なるスケール間のインタラクションである。例としてセルオートマトンを使って、シンプルなシステムが創発性を生み出せることを示し、タワーディフェンスがセルオートマトンのように動くゲームであることを解説した。最後に、フロムによる創発性の分類を紹介し、創発性は、異なるスケールでシステム内のパーツ間のフィードバックループやインタラクションのさまざまな組み合わせによって作り出されていることを示した。

3.5 演習

1. 図3.2（53ページ）に示したウルフラムのルールをいくつか書き直して、8つの可能性のある組み合わせのうちのいくつかを、ウルフラムのオリジナルとは異なる結果が出るようにしよう。方眼紙と鉛筆を使って、1つのマス目から始めてページの下まで新しく作ったルールを繰り返し適用しよう。その結果は図3.2とどのように違うか？
2. コンウェイのライフゲームは格子状のグリッド上で、隣接する8つのマス目の状態に基づいて、マス目の状態を変えるルールを適用している。六角格子では隣接するマス目が8つから6つになり、三角格子では3つになる。ライフゲームのルールを変更して六角格子や三角格子でプレイするとどんな結果が出るか、試してみよう。

第4章 内部経済

第1章では、物理、内部経済、進行、戦略的な操作、社会的インタラクションの5種類のメカニクスを紹介した。本章では、内部経済を取り上げる。現実の世界では、**経済**は1つのシステムとなっており、その中でリソースが生産されたり消費されたりして、定量的な数値で交換される。多くのゲームも経済を持っており、ゲームが操作する対象のリソースと、それを生産および消費する方法を規定するルールとで構成されている。ところがゲームでは、現実世界には存在しない種類のリソースでも内部経済に含めることが可能だ。ゲームではヘルス（体力値）や経験値、スキルなどがお金や品物、サービスなどと同様に経済の一部として扱える。Doomではお金は持てないが、武器、弾丸、ヘルス、アーマーポイントを持つことができる。ボードゲームのRiskでは、軍隊が重要なリソースだ。序盤では、失うリスクをかけても先手を打って軍隊を派遣して領土を占領する。スーパーマリオギャラクシーでは、星（スター）とパワーアップアイテムを集め、ライフ（命）を獲得してゲームを進める。ゲームでは、現実世界の経済とは似ても似つかないものであっても、ほぼすべてのジャンルで内部経済が存在するのである（詳しい例については9ページの表1.1を参照）。

ゲーム内経済を理解しなければ、そのゲームのゲームプレイを理解することはできない。内部経済がシンプルで小さいゲームもあるが、経済の大小にかかわらず、内部経済を構築することはデザインの中でも重要なタスクである。しかも、ゲームデザイナーにしかできない、他人に任せられないタスクの1つだ。物理を正しく機能させるにはプログラマと緊密に協力する必要があり、レベルをしっかりとデザインするにはストーリーライターやレベルデザイナーと協力しなくてはならない。だが、内部経済はゲームデザイナー自身が行う仕事だ。つまり、インタラクトするのが楽しく、かつチャレンジのあるゲームシステムを生み出すメカニクスを作り上げることはゲームデザイナーという職業のコアである。アーネスト・アダムスは *Fundamentals of Game Design* でゲームの内部経済について解説している。本書ではアダムスが提起した要点を紹介しながら、内部経済という概念そのものを拡大していく。

第4章 内部経済

本書では「経済」という用語を広い意味で使っている。金銭にかかわるものだけではないのだ。情報経済においては、データを生産する者、データを処理する者、データを消費する者が存在する。政治経済学は政治によって政府の政策がどのように影響を受けるかを研究する。お金についての経済学は市場経済学と呼ばれる。だが本書では、経済という言葉をもっと抽象的に使って、あらゆる種類のリソースを生産、交換、消費できる、あらゆる種類のシステムを指している。

メモ

4.1 ## 内部経済の諸要素

本節では、ゲーム経済の基本的な要素、すなわちリソース、エンティティ、およびリソースの生産・交換・消費を可能にする4つのメカニクスを簡単に紹介する。これは概要レベルのものだ。もっと深い理解をしたい場合は *Fundamentals of Game Design* の第10章 “Core Mechanics” を参照してほしい。

リソース

すべての経済はリソースの流れを中心に回転している。リソースとは数値で計測できるすべての概念を指す。ゲームにおいて、ほぼすべてがリソースとして機能する。つまり、お金、エネルギー、時間、プレイヤーが制御するユニットなどすべてがリソースの例だし、アイテム、パワーアップ、プレイヤーに敵対する敵もそうだ。プレイヤーが生産、収集、破壊するものも何らかのリソースであるが、すべてのリソースがプレイヤーの制御下にあるわけではない。時間は通常、自然に消えてしまうリソースであり、プレイヤーがそれを変えることはできない。スピードもリソースだが、普通は物理エンジンの一部として使われるのであって、内部経済の一部ではない。ただし、ゲーム内にはリソースでないものも当然存在する。足場（プラットフォーム）、壁、非アクティブな物体や固定レベルのような地形に類するものはリソースではない。リソースには有形のものと無形のものがある。**有形リソース**はゲームの世界で物理的な特性を持っている。ある場所に存在するものであり、どこか別のところに運ばなくてはいけないことが多い。たとえば、ウォークラフトではアバターがインベントリや木を持ち歩くが、こうしたアイテムはゲーム内で採集できるものだ。ストラテジーゲームでは、プレイヤーのユニットも有形リソースであり、ゲーム世界内で移動させる必要がある。

無形リソースはゲームの世界において物理的な特性を持たない。つまり、空間を占有したり、ある場所に存在したりすることはない。たとえば、ウォークラフトで木が採集されると、木材という無形のリソースに変わる。木材は数値にすぎず、ある場所に存在するものではないのだ。プレイヤーは、建物を建てるために木材を物理的にそこまで運ぶ必要はない。たとえその木材が採集された場所よりはるかに離れた土地で建設に使われるにしても、必要な数の木材を集めるだけで建設が始まるのだ。ウォークラフトの木と木材は、ゲームにおいて有形リソースと無形リ

4.1 内部経済の諸要素

ソースを交換する処理方法の優れた例だ。シューターゲームにおける救急キット（有形）とヘルスポイント（無形）もこの例である。

リソースが**具体的**か**抽象的**かの区別も役に立つ。抽象的リソースは、ゲームに本当に存在するわけではないが、ゲームの現在の状態によって計算することが可能である。たとえばチェスでは、対戦相手より戦略的に優位に立つために、ある駒を犠牲にすることがある。この場合、「戦略的に優位であること」は抽象的リソースと考えることができる（抽象的リソースは無形リソースである。当然ながら、「戦略的に優位であること」はある場所に保存できる物ではない）。同様に、アバターまたはユニットの高度、すなわち高所にいることはプラットフォームゲームやストラテジーゲームで優位に働く。この場合、ある位置を確保することが戦略的な価値として計算できるリソースのみ、高度をリソースとして処理することに意味がある。通常、ゲームではプレイヤーに抽象的リソースを明示的に伝えることはない。内部の計算用に使われるだけである。

ビデオゲームでは、抽象的に見えるリソースが実際には具体的なリソースの場合もあることに注意してほしい。たとえば、経験値はロールプレイングゲームでは抽象リソースではない。無形ではあるが、お金と同様に働いて得たり、何かのために使ったりする実際の必需品（商品）である。幸福度と名声も多くのゲームで使われるリソースだが、両方とも無形であるにもかかわらず、ゲーム内では具体的なものだ。

既存ゲームの内部経済を研究する場合も同じだが、ゲームの内部経済をデザインするには、まず、主なリソースを見極めることから始めよう。これを見極めてから、主なリソース間の関係を支配するメカニズムの記述と、主なリソースがどのように生産され、消費されるかの記述を行おう。

エンティティ

あるリソースの固有の量は**エンティティ**（entity）に保存される（プログラマにとってエンティティ（実体）とは事実上、変数である）。リソースは一般的な概念だが、エンティティは、あるリソースの具体的な量を保存するものだ。たとえば、「タイマー」という名前のエンティティは「時間」のリソースだ。おそらくそれは、ゲーム終了までの残りの秒数などを保存する。モノポリーでは各プレイヤーが、手持ちの現金リソースを保存するエンティティを持っている。プレイヤーが売買したり、賃貸料や罰金を支払ったりすれば、エンティティ内の現金の量が変化する。プレイヤー A がプレイヤー B に賃貸料を支払うと、A のエンティティから B のエンティティに現金が移動する。

1 つの値を保存するエンティティは**シンプルエンティティ**（simple entity）と呼ばれる。**複合エンティティ**（compound entity）とは、関連したシンプルエンティティの集合であり、ある複合エンティティには 2 つ以上の値を含めることができる。たとえば、ストラテジーゲームのユニットには、ヘルス、ダメージ能力、最大スピードなど多くのシンプルエンティティが含まれている。これらを集めると複合エンティティとなり、各構成要素は**属性**として扱われる。よって、ユニッ

第4章 内部経済

トのヘルスはそのユニットの属性の1つになるのである。

4つの経済メカニクス

通常、経済には、リソースに影響を与え、リソースを移動させる4つの機能がある。この4つの機能とは、**ソース**、**ドレイン**、**コンバータ**、**トレーダー**と呼ばれるメカニクスである。この4つをここで紹介するが、これは要約である。詳細については *Fundamentals of Game Design* の第10章 “Core Mechanics” を参照してほしい。

- **ソース (source)** とは、新たなリソースをゼロから生み出すメカニクスである。決められた時間、あるいは決められた条件が満たされると、ソースは新たなリソースを生産し、どこかのエンティティに保存する。ゲーム内のイベントによってソースがトリガーされる場合もあるし、継続的に一定の速度でリソースを生産する場合もある。また、ソースをオンにしたりオフにしたりすることも可能だ。シミュレーションゲームでは、一般に、ソースによって人口に比例した額のお金が周期的に生産される。また、戦闘を伴うゲームで、一定時間が経過したときに自動的にヘルスが再生するのも、この例である。
- **ドレイン (drain)** とはソースの反対のメカニクスである。エンティティに保存されているリソース量を減らして、完全に消去することで、ゲームからリソースを取り除く。人々に食料を与えなければならないシミュレーションゲームでは、食料が人口に比例した量で定期的に消去されていく。なくなった食料はどこかに移動したり何かに変わったりするわけではなく、消えてしまうのである。シューターゲームでは、銃で撃つことで弾丸が消去される。
- **コンバータ (converter)** とは、ある種類のリソースを別の種類のリソースに変換するメカニクスだ。先に述べたように、ウォークラフトでは、木（有形リソース）は、採集されると木材（無形リソース）に変わる。採集という行為は、一定の比率で木を木材に変換する「コンバータ」メカニクスである。つまり、木の本数によって変換される木材の量が決まる。多くのシミュレーションゲームでは、テクノロジーのアップグレードを行うことでコンバータメカニクスの効率を高められる。これを行うと、アップグレード後は従来よりも大量のリソースを生産できるようになる。
- **トレーダー (trader)** とは、交換ルールに従って、あるエンティティから別のエンティティにリソースを移動したり、逆方向に戻したりするメカニクスだ。プレイヤーがゴールド3枚で鍛冶屋（ブラックスミス）から盾（シールド）を買うと、「トレーダー」メカニクスはプレイヤーの現金エンティティから鍛冶屋の現金エンティティにゴールド3枚を移動し、鍛冶屋のインベントリからプレイヤーのインベントリに盾を移動する。トレーダーはコンバータとは異なり、何も生産されず、何も破壊されない。物が交換されるだけだ。

4.2 内部経済の構造

4.2 ## 内部経済の構造

経済を構成するエンティティとリソースを特定することは特に難しいことではないが、経済システム全体を把握する広い視点を獲得するのは難しい。経済の構成要素でグラフを作るとしたら、どのような形になるだろうか。あるリソースの量は時間とともに増えるだろうか。リソースの配分はどのように変わっていくだろうか。リソースは特定のプレイヤーの手に蓄積される傾向があるか、それとも、システムによって広く配分される傾向にあるだろうか。これらの答えは内部経済の構造を理解することで得られる。

経済の形

現実の世界では、経済の特徴を表すのにチャートや数字を使う（図 4.1）。こうしたグラフには興味深い特徴がいくつかある。小さなスケールで見ると、線グラフはカオス的な動きを見せるが、大きなスケールではしだいにパターンが見えてくるものだ。長期的に見て線グラフが上昇するか下降するかは簡単に分かるし、好調な時期と不調な時期を見極めるのも簡単である。言い換えると、この種のチャートから特徴ある形やパターンを読み取ることができるのである。

ウォール街大暴落 ダウ工業株平均グラフ（1929 年）

（グラフ：縦軸 0～400、横軸 10 月～10 月（1929 年～1930 年）、ピーク付近に赤いハイライト）

図 4.1 大恐慌を引き起こした株式市場の暴落のグラフ。大半の動きはカオス的だが、暴落ははっきりと見て取れる。

が、大きなスケールではしだいにパターンが見えてくるものだ。長期的に見て線グラフが上昇するか下降するかは簡単に分かるし、好調な時期と不調な時期を見極めるのも簡単である。言い換えると、この種のチャートから特徴のある形やパターンを読み取ることができるのである。

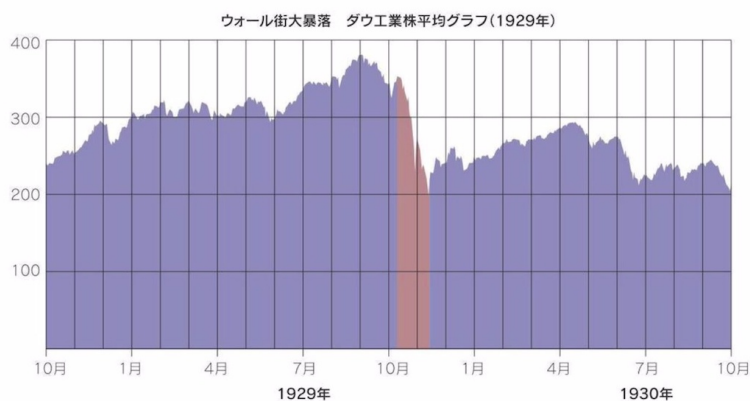


図4.1 大恐慌を引き起こした株式市場の暴落のグラフ。
大半の動きはカオス的だが、暴落ははっきりと見て取れる。

図 11 図 4.1 株式市場の暴落グラフ

では、同様に、あるゲームにおけるプレイヤーの資産（リソースの豊富さ）を表わすグラフを作ってみよう。ご覧のとおり、ゲームの内部経済から独特の形とパターンが出現している。とはいえ、ゲームプレイの質を見極める形というものが存在するわけではない。何が優れたゲームプレイをもたらすのかは、あなた自身が自分のゲームに設定した目標と、それをとり巻く文脈に大

第4章 内部経済

きく依存するのである。たとえば、プレイヤーが長い間苦勞してからやっと頂点に立てるようにゲームを設定することもできる（図4.2）。あるいは、すぐに運命のどんでん返しを狙える、ずっと短いプレイスルーにすることも可能だ（図4.3）。

（図4.2：縦軸「運」、横軸「時間」。青線（敵）が上から始まり下降、赤線（プレイヤー）が下から上昇し、最終的にプレイヤーが逆転する折れ線グラフ）

図4.2 長大なゲーム。プレイヤーは長時間にわたって強力な敵と格闘し、ついに勝利を得た。

（図4.3：縦軸「運」、横軸「時間」。赤線（プレイヤー1）と青線（プレイヤー2）が細かく上下に交差し、最終的にプレイヤー1が上回る折れ線グラフ）

図4.3 運命のどんでん返しがある短いゲーム

きく依存するのである。たとえば、プレイヤーが長い間苦勞してからやっと頂点に立てるようにゲームを設定することもできる（図4.2）。あるいは、すぐに運命のどんでん返しを狙える、ずっと短いプレイスルーにすることも可能だ（図4.3）。

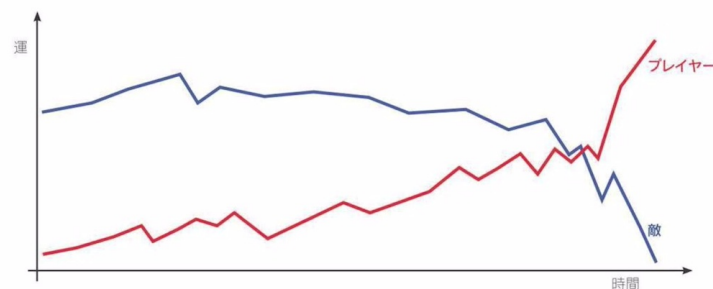


図4.2 長大なゲーム。プレイヤーは長時間にわたって強力な敵と格闘し、ついに勝利を得た。

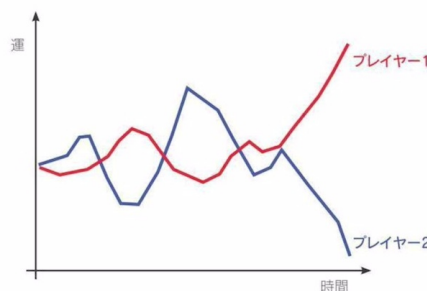


図12 図4.2/4.3 ゲームの形

チェスにおけるゲームの形

ゲーム内経済の形を研究するベースとして、チェスにおけるプレイヤーの持つ財産の推移を見ていこう。チェスにおいて重要なリソースはプレイヤーの駒である。プレイヤー（そしてコンピュータのチェスプログラム）は、各駒に対し、その種類によって点数を割り当てる。たとえば、あるシステムでは、ポーン=1点、ルーク=5点、クイーン=9点だ。全ピースの合計値は、そのプレイヤーの盤上の**マテリアル**（material）と呼ばれる点数になる。プレイヤーは、自分の駒を使って盤上で戦略的に優位に立とうとする。**戦略的アドバンテージ**（strategic advantage）は、ゲームの抽象的リソースとして計測できる。図4.4は、あるチェス戦における2人のプレイヤーのプレイの流れをグラフ化したものだ。

4.2 内部経済の構造

(図 4.4：縦軸「運」、横軸「ターン数」。グラフ全体が緑系の背景色で覆われ「序盤」「中盤」「終盤」の 3 ゾーンに区分される。白と黒の 2 本の階段状折れ線グラフが描かれ、それぞれ「マテリアル」と「優位性 (アドバンテージ)」の 2 層を示す)

図 4.4 あるチェス戦の流れ。グラフの色 (白と黒) はプレイヤーの色に対応する。

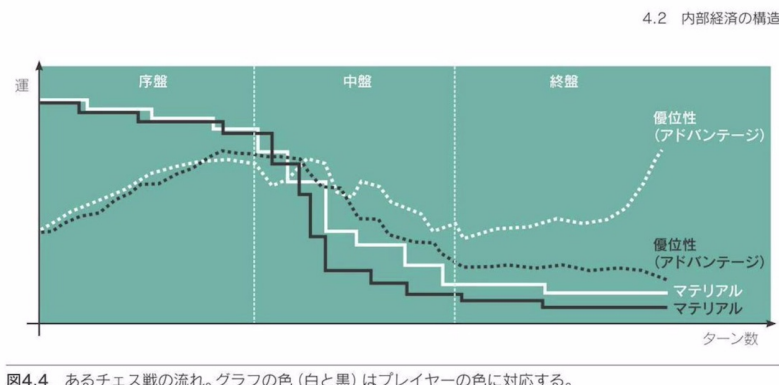


図4.4 あるチェス戦の流れ。グラフの色 (白と黒) はプレイヤーの色に対応する。

図 13 図 4.4 チェス戦の流れ

このグラフを見ると、重要なパターンがいくつかあることが分かる。まず、両プレイヤーの主なリソース (マテリアル) は長期的には下降する傾向がある。プレイが進むにつれ、両プレイヤーはピースを失ったり犠牲にしたりするのだ。マテリアルの獲得は非常に難しい。チェスでは、ピース獲得の唯一の方法はポーンを一番奥まで進めて強い駒に昇格させてマテリアルを増やすことだ。実際に昇格できることは稀で、昇格できた場合にはプレイヤーの資産バランスが劇的に変わることになる。マテリアルだけを考慮すると、チェスは消耗戦のようなものだ。マテリアルを長持ちさせたプレイヤーのほうが、勝算は高くなる。

戦略的アドバンテージはゲームにおいて、より動的なものになる。つまり、プレイの流れの中で獲得したり失ったりするものになるのだ。プレイヤーは、自分のマテリアルを使って戦略的アドバンテージを得たり、相手の戦略的アドバンテージを減らしたりする。マテリアル量の差と、戦略的アドバンテージを獲得する能力の間には、間接的な関係がある。マテリアルを多く持っているほうが戦略的アドバンテージを獲得しやすくなるのだ。そして、その戦略的アドバンテージを活用すれば、相手の駒をさらに取って、相手のマテリアルをもっと減らすことができるだろう。ときには、自分のピースを犠牲にして戦略的アドバンテージを得ることもできるし、相手が戦略的な差を失うように誘い出すことも可能だ。

一般に、チェスは、**序盤**、**中盤**、**終盤**の3つの段階を経て進行する。各段階はゲームの中で固有の役割を持っており、分析方法も異なる。序盤は通常、事前準備が完了した、研究の終わっている定石に沿って進行する場合が多い。序盤では、プレイヤーは自駒の布陣を有利に展開しようとする。終盤は駒数が少なくなってから始まり、キングを参戦させることが安全につながるようになる。中盤はこの序盤と終盤の間のどこかになるが、その境界線は明確ではない。この3つの段階を経済分析によって明らかにしたのが図 4.4 だ。序盤では、駒の数の減り方はゆっくりで、両プレイヤーが布陣を有利に展開しようとする。中盤は、プレイヤーが自分の戦略的アドバンテージを使って対戦相手の駒をとろうとしたときに始まる。これはマテリアルの急激な落ち込み

第4章 内部経済

によって特徴づけられている。終盤では、両プレイヤーが戦略的アドバンテージを勝利への最終チャンスに集中させようとするため、マテリアルは再び安定する。

このチェスの分析は、なじみのあるゲームを使って経済原則を説明するためのハイレベルの抽象概念である。チェス理論の古典で経済的な用語が使われないのは、チェスとはキングをチェックメイトするゲームであって、より多くの駒を取るゲームではないからだ。だが、経済のゲームでなくても、ゲームプレイとゲーム進行は経済用語によって説明できることを示すためにこのように解説した。

メモ

メカニクスから形へ

ある特定の経済の形を作るには、その形を作るメカニクスの構造の種類が分かっているなければならない。幸い、ゲームの経済の形と、そのメカニクスの構造との間には直接的な関係がある。この後の数項にわたって、経済の形において最も重要な構成部品について解説する。

ネガティブフィードバックによる平衡点の生成

ネガティブフィードバック（第3章「複雑系と創発型の構造」を参照）は、動的システムにおいて安定性を生み出すために使われる。ネガティブフィードバックによってシステムは変更に対して抵抗を示すようになる。たとえば冷蔵庫の温度は、冷蔵庫の周りの温度が変わっても一定に保たれる。システムが安定する地点は**平衡点**（equilibrium）と呼ばれる。図4.5はネガティブフィードバックの影響を図に表したものだ。

（図4.5：縦軸「運」、横軸「時間」。赤い折れ線「プレイヤーの運」が最初に山なりを描いた後、点線「平衡点」に収束していく）

図4.5 ネガティブフィードバックの影響

保たれる。システムが安定する地点は**平衡点**（equilibrium）と呼ばれる。図4.5はネガティブフィードバックの影響を図に表したものだ。

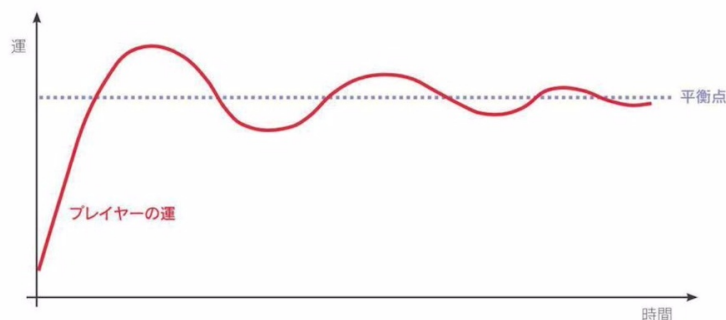


図4.5 ネガティブフィードバックの影響

図14 図4.5 ネガティブフィードバックの影響

最もシンプルな平衡では水平線の形をとるが、平衡点が異なるシステムもある。平衡点は時間が経つにつれ、あるいは周期的に、変化する可能性がある（図4.6）。平衡点を変えるには、ネガティブフィードバックメカニズムをある程度独立して変更する動的要因†1が必要だ。1年を通じた外部温度は、日照と照射量の変化に伴う周期的な平衡点を描く例である。

†1 訳注：経済用語では拡散指数、DI

4.2 内部経済の構造

(図 4.6 左：縦軸「運」、横軸「時間」。赤い折れ線が全体的に右上がりに上昇しながら点線（平衡点）に沿って揺れる。左図キャプション：平衡点が上昇し続ける）

(図 4.6 右：縦軸「運」、横軸「時間」。赤い折れ線が急な上下を繰り返し、点線（平衡点）が周期的に高低する。右図キャプション：周期的に平衡点に変化している）

図 4.6 変わりゆく平衡点のネガティブフィードバック。左は平衡点が上昇し続け、右は周期的に平衡点に変化している。

ポジティブフィードバックによる軍拡競争の生成

ポジティブフィードバックは指数曲線を描く（図 4.7）。預金口座に利子がたまっていくのがこの曲線の典型的な例だ。預金口座に利子が入っていく一方だと、時間が経つにつれ蓄積された額にさらに利子がつくため、総額は急上昇する。この種のポジティブフィードバックを使ってマルチプレイヤーの軍拡競争（アームズレース）を作り出すゲームは多い。例として、StarCraft（あるいは似た構成を持つリアルタイムストラテジーゲーム）における原材料の採集が挙げられる。StarCraft では、鉱物を 50 使って採掘ユニット —— SCV（Space Construction Vehicle）と呼ばれる宇宙建設車両 —— を構築すると、新しい鉱物を採集できる。StarCraft のプレイヤーが SCV を構築するために自分の鉱物収入の一部をとっておけば、この鉱物は預金口座に預けたお金と同じ曲線を描くのである。

(図 4.7：縦軸「運」、横軸「時間」。赤線「プレイヤー 1」と青線「プレイヤー 2」が両方とも指数的に上昇しており、プレイヤー 1 の方が急な上昇曲線を描く）

図 4.7 ポジティブフィードバックは指数曲線を描く。

4.2 内部経済の構造

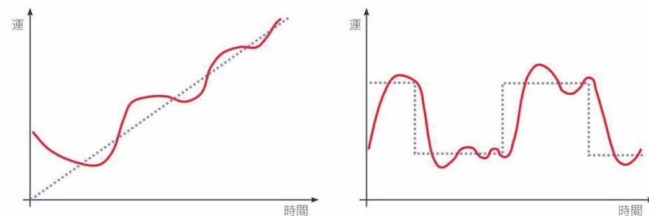


図4.6 変わりゆく平衡点のネガティブフィードバック。
左は平衡点が上昇を続け、右は周期的に平衡点に変化している。

ポジティブフィードバックによる軍拡競争の生成

ポジティブフィードバックは指数曲線を描く（図4.7）。預金口座に利子がたまっていくのがこの曲線の典型的な例だ。預金口座に利子が入っていく一方だと、時間が経つにつれ蓄積された額にさらに利子がつくため、総額は急上昇する。この種のポジティブフィードバックを使ってマルチプレイヤーの軍拡競争（アームズレース）を作り出すゲームは多い。例として、StarCraft（あるいは似た構成を持つリアルタイムストラテジーゲーム）における原材料の採集が挙げられる。StarCraftでは、鉱物を50使って採掘ユニット —— SCV (Space Construction Vehicle) と呼ばれる宇宙建設車両 —— を構築すると、新しい鉱物を採集できる。StarCraftのプレイヤーがSCVを構築するために自分の鉱物収入の一部をとっておけば、この鉱物は預金口座に預けたお金と同じ曲線を描くのである。

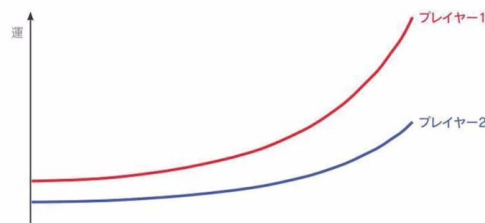


図 15 図 4.6/4.7 フィードバックと指数曲線

当然ながら、StarCraft のプレイヤーは SCV ユニットのみにリソースを費やすわけではない。軍事ユニットを構築したり、基地を拡張したり、新しいテクノロジーを開発したりする必要がある。だが、長期的に見ると、基地の経済成長の潜在能力が重要になってくる。大半のプレイヤーは、守りを固めて多くのリソースを採集し、軍事ユニットを構築する能力を優勢にしてから敵の殲滅にとりかかるのである。

第4章 内部経済

デッドロックと相互依存性

ポジティブフィードバックメカニズムによってデッドロックや相互依存が生じることがある。StarCraft では、鉱物を採集するには SCV ユニットが必要であり、SCV ユニットを獲得するには鉱物が必要だ。この2つのリソースには互いに依存関係があり、この依存性によってデッドロック（deadlock；行き詰まり）状況に陥る可能性がある。鉱物も SCV ユニットも持たないプレイヤーは、まったく生産を始められないのだ。実際、ある程度の鉱物と最低1台の SCV ユニットがなければ、このフィードバックループを可能にする3番目のリソースである司令本部を構築できない。このデッドロック状態は潜在的な脅威である。敵のプレイヤーが SCV ユニットをすべて破壊する可能性があるのだ。鉱物をすべて軍事ユニットにつぎ込んだときにこれをやられると、大変なことになる。これはレベルデザインのベースとしても使えるものだ。たとえば、軍事ユニットといくらかの鉱物でミッションを開始するが、SCV ユニットや司令本部はないとする。この場合、SCV ユニットを見つけて救出しなくてはいけない。デッドロックと相互依存はメカニクスのある特定の構造が持つ特徴である。

ポジティブフィードバックをゲームで有効に活用するベストな方法は、決定的な差が生じた時点でプレイヤーがすぐに勝利を収められるようにすることだ。図4.7で明らかになったように、ポジティブフィードバックは小さな違いを増幅する。つまり、利子が同じ場合、2つの銀行口座の残高の違いは時間とともにもっと大きくなる。ポジティブフィードバックのこの効果は、決定的な差が生じた時点でゲームを終了に向かわせる能力に使える。誰が勝つのか明らかになってからもだらだらとゲームを続けたい人は、どのみちいない。

破壊メカニズムへのポジティブフィードバック

ポジティブフィードバックは必ずしもプレイヤーを勝たせるわけではない。プレイヤーを負かすこともできる。たとえば、チェスで駒を失うと、状況が悪くなり、もっと駒を失う危険性も高まるが、これはポジティブフィードバックループの結果だ。ポジティブフィードバックは、チェスでマテリアルを失うケースで見えてきたように、破壊メカニズムに応用することもできる。この場合、**下方スパイラル**と呼ばれることがある。破壊メカニズムへのポジティブフィードバックは、ネガティブフィードバックと同じではないことに注意してほしい。ネガティブフィードバックは効果を減らして平衡をもたらすものだ。ネガティブフィードバックを破壊メカニズムに結び付けることもできる。この場合、シューターゲームのハーフライフである。プレイヤーのヒットポイントが低くなると、回復アイテムであるヘルスパックを放出するようになっている。

長期投資と短期ゲイン

StarCraft が鉱物をできるだけ多く収集するゲームで、他のことは何も考慮しなくてよいとしたら、鉱物を収集するたびに新しい SCV ユニットを構築するのがベストの戦略だろうか。いや、必ずしもそうとは限らない。新しい SCV ユニットに収入をつぎ込み続けると、鉱物をためておくことはできず、ゲームで勝つことができなくなる。鉱物を収集するには、ある時点で SCV ユニットの構築をやめて鉱物をため始める必要がある。それをいつ行うのがベストなのかは、ゲームの目標と制約、さらに他のプレイヤーの動向によって決まる。限られた時間内に最大数の鉱物を収集すること、または、できるだけ早く一定数の鉱物を収集することを目標とするなら、それによって構築すべき SCV ユニットの理想的な数値が決まるはずだ。

4.2 内部経済の構造

この効果を理解するには図 4.8 を見てほしい。このグラフのとおり、新しい SCV ユニットに投資を続けている限り、鉱物が蓄積されることはない。しかし、投資をやめるとすぐに鉱物は安定した率で増加する。この増加率は保有する SCV ユニット数によって決まる。ユニット数が多ければ、鉱物の増加率が高くなるのだ。投資を続ける期間が長ければ長いほど、鉱物の蓄積を開始するのが遅くなるが、最終的には追いつき、自分より早く蓄積を開始したプレイヤーを追い越すことができる。目標値によってこれらのうちで最も効果のあるものが決まるのだ。

(図 4.8：縦軸「鉱物」、横軸「時間」。赤線「6 SCV ユニット」と青線「4 SCV ユニット」の 2 本のグラフが描かれる。どちらも投資期間中は階段状の鋸歯波を描いており、点線「50 の鉱物」で両者が投資をやめる時点が示され、その後それぞれが安定した傾きで上昇する。6 SCV ユニットの方が急勾配)

図 4.8 蓄積競争



図 16 図 4.8 蓄積競争

StarCraft が単に鉱物を収集するゲームでなかったことは幸いだ。SCV ユニットにすべての鉱物をつぎ込むという戦略はまずい。なぜなら、やがて攻撃されてしまうからだ。長期的な目標と、基地の守りを固めるなどの短期的要件とのバランスをとらなくてはならない。さらに、序盤で攻撃戦力を素早く構築して、まだ防御を固めていない敵を圧倒する戦略を好むプレイヤーもいる。これは「タンクラッシュ (tank rush)」と呼ばれ、コマンド&コンカー Red Alert で初めて使われて有名になった戦略だ。マップによっては、リソースへの初期アクセスが限られているところがあり、マップ上を素早く動き回って将来のリソースへのアクセスを確固たるものにする必要もある。SCV ユニットへの投資は長期的に見ると良い戦略だが、序盤でリスクをとることになる。タンクラッシュで迅速な勝利を得ることはあきらめることになるだろう。

プレイヤーパフォーマンスとリソース配分によるバリエーション

StarCraft で、鉱物の収集率を決めるのは SCV ユニットの数だけではない。鉱物は、マップ上の特定の場所にあるクリスタル鉱床から採集するからだ。基地の場所として最善の地を決め、SCV ユニットがクリスタル鉱床から鉱物を効率的に採集するように管理するにはスキルが必要となる。これは、プレイヤースキルとゲーム世界の地形によって、ゲームの経済の振る舞いに影響を与える入力バリエーションがいかに生成されるかを示す好例である。プレイヤー入力はもちろん経済に影響を及ぼすわけだが、プレイヤー入力を頻繁に発生させながらも、1つの入力が大きなエフェクトにならないようにするのがベストだ。

相対的スコアに基づいたフィードバック

マーク・ルブラン (Marc LeBlanc) が 1999 年のゲームデベロッパーズカンファレンスで「ゲームにおけるフィードバックメカニズム」について講演を行ったとき、バスケットボールの 2 つのバージョンを解説した。「ネガティブフィードバックバスケットボール」では、5 点差がつくごと

第4章 内部経済

とに、負けている側のチームがプレイヤーを1人追加することができる。「ポジティブフィードバックバスケットボール」では効果を逆にする。つまり、5点差がつくごとに、勝っている側のチームがプレイヤーを1人追加できる。両プレイヤーの差を使って構築されるフィードバックメカニズムは、絶対値を受け取って動作するメカニズムとは若干異なる影響を生み出す。フィードバックメカニズムが作用するのは、プレイヤーの絶対的なリソース量に対してではなく、プレイヤー間の差に対してであるからだ。これによって直感に反した効果が生まれる。たとえば、ネガティブフィードバックバスケットボールの経済グラフでは、勝ちチームのリード点数は負けチームの追加プレイヤーで相殺され、一定の距離を安定して保っている（図4.9）。

（グラフ）

図4.9 ネガティブフィードバックバスケットボールのスコアグラフ

ネガティブフィードバックバスケットボールの経済グラフでは、勝ちチームのリード点数は負けチームの追加プレイヤーで相殺され、一定の距離を安定して保っている（図4.9）。

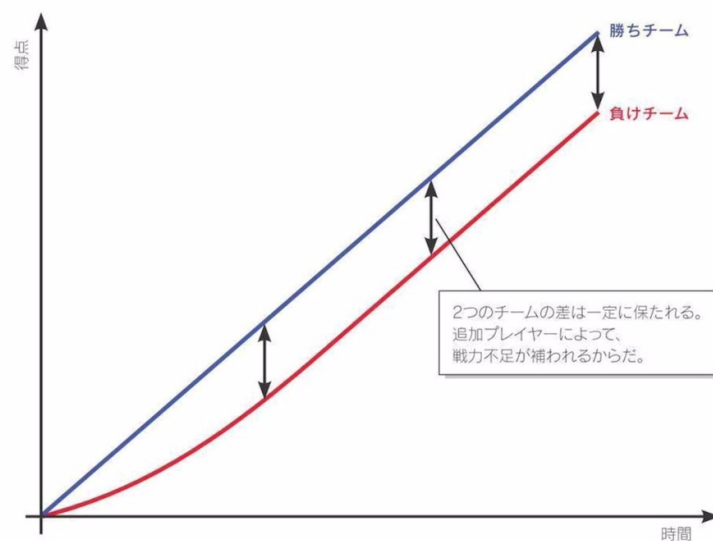


図4.9 ネガティブフィードバックバスケットボールのスコアグラフ

図17 図4.9 ネガティブフィードバックバスケットボール

（グラフの説明）縦軸：点差横軸：時間青い線：勝ちチーム赤い線：負けチーム吹き出し：「2つのチームの差は一定に保たれる。追加プレイヤーによって、戦力不足が補われるからだ。」

動的平衡

2人のプレイヤーのリソース差を与えたネガティブフィードバックメカニズムで作られる平衡は動的平衡である。つまり、ある固定値に向かうのではなく、ゲームの変更要因である相手に相互依存するようになるのだ。ゲームでネガティブフィードバックを使用して面白くなるのは、このように動的な形であることが分かるだろう。複数のプレイヤーの資産の相対的なバランス、あるいはゲーム内の他の要因に基づいて、ネガティブフィードバックループの平衡点を動的にすると、非動的な平衡で見られる予測可能なバランスとは違う優れた手法になる。経験や知識、スキルを積みめば、要因をいくつか組み合わせて、周期的、進行的、あるいは別の望ましい形をとる動的平衡を作ることができるはずだ。

2つのチームがポジティブフィードバックバスケットボールをプレイすると、スキルの差はあっというまに大きくなる。チームの実力が異なると、かなり一方的な試合になるのだ。だが、両チー

4.3 ゲームにおける内部経済の活用法

ムの実力が拮抗していると、別のパターンが出現する。それは、試合はかなりの接戦になるが、どちらかが決定的なリードを奪うと、やはり一方的な試合になるというものだ。後者の場合、わずかなスキル差がいつも以上のがんばり、あるいは純粋な運が決定的要因になる。

バスケットボールにおけるポジティブ／ネガティブフィードバックのゲームプレイ効果については第6章で詳細に解説する。

ラバーバンド効果は相対的な位置へのネガティブフィードバック

レーシングゲームでは、興奮を呼ぶ張りつめたレースにするためにプレイヤーの位置に基づいたネガティブフィードバックを使う。このメカニズムはラバーバンド効果と呼ばれる。これは、プレイヤーから見ると、他のレースカーが輪ゴムで引っ張られているように見え、前に出すぎることなければ、遅れすぎることもないのだ。先頭のレースカーの速度を落とし、後方のレースカーの速度を上げる。ラバーバンドを実装しているゲームもある。もっと巧みなネガティブフィードバックメカニズムを使って、同様の効果を達成しているゲームもある。マリオカートでは、パワーアップアイテムを拾うとランダムで特別なパワーがもらえる。ところが、遅いプレイヤーには、遅いプレイヤーよりも強力なパワーアップアイテムを拾うチャンスが大きくなっているのだ。さらに、マリオカートの攻撃アイテムはプレイヤーの前方の敵に使うため、遅いプレイヤーよりリースの先頭のプレイヤーのほうがされやすい。このメカニズムによってトップが頻繁に入れ替わり、ゲームの興奮を高め、土壇場でトップをぶっちぎるチャンスを引き出している。

4.3 ゲームにおける内部経済の活用法

前節ではゲームの内部経済の要素と共通構造について解説した。本節では、さまざまなジャンルのゲームで一般にゲーム経済がどのように使用されているかについて解説する。内部経済の一部になっているメカニクスの概要は9ページの表1.1に示してある。では、どのゲームジャンルにも使われている典型的な経済の構造について詳細を説明しよう。

内部経済を使って物理を補完する

当然ながら、アクションゲームのコアメカニクスの大半は物理だ。物理を使ってプレイヤーの器用さ、タイミング、正確性をテストする。それでも、大半のアクションゲームが内部経済を追加して重要なリワードシステムを構築したり、リソースを必要とするパワーアップシステムを確立したりしている。ある意味では、得点システムを使うだけで、アクションゲームでは経済メカニクスを追加することになる。敵を倒せばポイントがもらえる場合、敵を倒すための投資量を考慮する必要がある。リスクにさらすべきか、弾丸を浪費すべきか、簡単には回復できない類のエネルギーを消費すべきか？

スーパーマリオブラザーズをはじめとした多くのプラットフォームゲームでは、リワードシステムを構築している。スーパーマリオブラザーズでは、コインを集めるとライフが増える。たくさんのコインを集める必要があるため、デザイナーはレベル全体にたくさんのコインを配置

第4章 内部経済

して、経済に大きな影響を与えることなく、プレイテスト中に追加したり削除したりできる。こうすることで、コインはプレイヤーにとってレベルの案内役になる（プレイヤーをガイドする収集可能なオブジェクトは**パンくず**（breadcrumb）と呼ばれる）。通常、すべてのコインは入手可能と想定されるため、コインを配置するなら、手に入れる方法も存在しなくてははいけない。これは、難しい場所に到達するスキルのあるプレイヤーに見返りを与えるチャンスだ。この方法を使うと、ゲームの内部経済は極めてシンプルにできる。だが、このようなシンプルな経済でさえフィードバックループが絡んでくるのだ。プレイヤーが多くのコインを集めるために努力すると、多くのライフを獲得することができ、その結果、もっとリスクをとってもっと多くのコインを集めようとする。

このようなシステムを設定するときには、リスクとリワードのバランスに注意する必要がある。生死を分けるようなワナの奥にコインを1枚だけ配置したりすると、プレイヤーはコイン1枚のためにライフを賭けることになってしまう。これでは公平とは言えないし、プレイヤーはだまされた気分になるはずだ。デザイナーとして、あなたにはリスクとリワードをマッチさせる責任がある。初心者のプレイヤーが通りそうな経路の近くに配置するときは、特に気をつけよう（プレイヤーに見える決して到達できないリワードを作るのはもっとも罪深い。決して得られないリワードのためにプレイヤーにリスクを取らせることになるからだ）。

ファーストパーソンシューターにおける武器や弾丸を含むパワーアップも同様な経済を構築している。パワーアップと弾丸はそれ自体を、プレイヤーがレベル内のすべて敵を倒すというチャレンジに対するリワードにすることができる。ゲームデザイナーとして、このバランスを正しくとらなければならない。ゲームによっては、敵を倒していくときの弾丸の消費量が、敵の死体から回収した弾薬の量を平均で上回っても、まったく問題ないものもある。とはいえ、これによってボスキャラとの最終戦で適切な弾丸が足りなくなるようでは、努力してきたプレイヤーを罰する危険がある。サバイバル志向のファーストパーソンシューターでは、武器と弾丸には乏しい経済を構築するのが一般に良い効果をもたらす。これによって緊張感とドラマが生まれるからだ。しかしこれは、作るのが難しいバランスだ。アクション志向のシューターなら、プレイヤーに多くの弾丸を確保するのがベストだろう。必要以上に敵を排除したら適切なリワードを与えることだ。

内部経済による進行操作

ゲームの内部経済は、移動を行うゲームで進行に影響を及ぼすことにも活用できる。たとえば、パワーアップやユニークな武器はアクションゲームの経済において特別な役割を果たしている。これらは新しい場所へのアクセスをもたらすのに使用できるのだ。プラットフォームゲームにおけるダブルジャンプ能力は、初期には行けなかった高い足場にプレイヤーを到達させることができる。こうした能力は、経済の観点では、抽象的なリソースの**アクセス**を生成する新しいリソースと言える。

4.3 ゲームにおける内部経済の活用法

アクセスとは、さらなるリワードの獲得に使用されたり、ゲームの進行に必要になったりするものだ。

どちらの場合でも、デザイナーとして、あなたはデッドロック状態には細心の注意を払わなくてはいけない。たとえば、レベルを終了する出口に監視の敵キャラを配置したとする。この敵キャラを1発でやつつけられる専用の武器をそのレベルのどこかに配置する。この武器はこのレベル内でいつでも使用できる。プレイヤーがこの武器を見つけると、10発の弾丸が入っており、次のレベルに行くまで補充されない（だが、初回のプレイではプレイヤーにこのことは分からない）。さて、初めて遊ぶプレイヤーがこの武器を発見し、2〜3回試し撃ちをして、数体の敵にお見舞いし、レベルの出口に着くころには1発しか残っていないことに気づいた。そして、プレイヤーはこの敵を撃つが、外してしまう。あなたはデッドロック状態を作ったのである。プレイヤーは弾丸を補充するために次のレベルにアクセスする必要があるが、そのアクセスを獲得するには弾丸が必要なのである。

ゼルダの伝説におけるデッドロック状態の解決策

ゼルダの伝説シリーズでは、弓矢、爆弾などの消耗アイテムを使って新しいエリアにアクセスをする。プレイヤーに必要なアイテムがなくなると、デッドロック状態が起こる危険性がある。ゼルダの伝説シリーズのデザイナーたちは、この勝ち目のない状況を起こさないために、必要なリソースには再生可能なソースを豊富に確保している。地下室には役に立つ壺がばらまかれており、プレイヤーが壺を破壊するとこれらのリソースが出てくる仕掛けになっている（図4.10）。破壊された壺は、プレイヤーが部屋から部屋へと移動するうちにどういふわけか復活するため、時間で補充されないソースになっている。壺には何でも入れられるため、このようなメカニズムを使うと、デザイナーは必要なあらゆるリソースをプレイヤーに提供できる。また、ゲームプレイのヒントにも使用できる。プレイヤーが矢をたくさん見つけると、もうすぐ弓が必要になると推測できる。

（ゲーム画面画像）

図 4.10 ゼルダの伝説における壺は役立つソースだ。



図4.10 ゼルダの伝説における壺は役立つソースだ。

図 18 図 4.10 ゼルダの伝説の壺

第4章 内部経済

内部経済による戦略的なゲームプレイ

リアルタイムストラテジーゲームの戦略的なチャレンジの多くが、経済の特徴を持っていることは驚くばかりだ。StarCraft では、戦場で戦うよりも経済を管理するほうにずっと多くの時間を費やすものだ。物理アクションや戦略アクションよりもずっと長期的な影響を及ぼす戦略的な次元を導入するのに、内部経済を組み込むのは優れた手法だ。

リアルタイムの大半に手の込んだ内部経済が使われている理由は、こうした経済によって、計画や長期的投資にリワードを与えられるからだ。計画も立てず長期的な投資もしない軍事ゲームは、おそらく戦場でユニットを操作することが中心になるため、ストラテジー（戦略）ゲームというよりタクティクス（戦術）ゲームだ。戦略的なインタラクションを一定のレベルに保つには、アクションゲームで物理を補完する以上のもっと複雑な内部経済が必要になる。ストラテジーゲームの経済には複数のリソースがかかわり、多くのフィードバックループと相互関係がある。この種の経済を初めて設定するのは難しく、適切なバランスを見つけることはもっと大変だ。デザイナーとしては、経済の要素を理解し、その動的な効果を判断する鋭い感覚を磨く必要がある。どれほど豊かな経験があってもミスは起こるものだ。StarCraft のようなゲームでは、リリース後、プレイヤーに新しい戦略を編み出され続けるが、繰り返し内部経済に調整を施すことで、リリースから長い時間が経過しても適切なバランスを保持しているのである。

生産の経済にフォーカスしない場合でも（例：StarCraft の鉱物や SCV ユニット）、ほぼすべてのゲームにおいて内部経済は戦略的な深みを加えることが可能だ。大半のケースで、これには利用可能なリソースを賢く使う計画が必要となる。すでに解説したとおり、チェスの経済はマテリアル（持ち駒）および戦略的アドバンテージという視点を用いて説明することができる。チェスは生産のゲームではなく、通常、駒を獲得することはできない。むしろ、自分のマテリアルを使って、ときには犠牲にして、できる限りの戦略的アドバンテージを生むことが中心だ。言い換えると、チェスは持ち駒を最大限に活用するゲームである。

これと似たようなことは「プリンス・オブ・ペルシャ 時間の砂」にも見られる。このアクションアドベンチャーゲームでは、プレイヤーには器用さが要求され、戦闘のチャレンジに満ちあふれた多くのレベルをたどっていく。ゲームの初期段階で、魔法の短剣を獲得すると、時間を制御できるようになる。何か不都合が起きれば、プレイヤーは短剣の砂を使って時間を巻き戻し、再挑戦できるのだ。このパワーは、大ダメージをくらった直後など、戦闘中に使うこともできる。さらに、砂の魔法の力を使って時間を停止させることも可能だ。これは複数の敵との対戦に役立つ。だが、砂は無限ではない。プレイヤーが巻き戻せる時間には限りがあるが、幸いなことに、敵を倒すと新しい砂が手に入る。これは、通常のアクション中心のゲームプレイに加えて、プレイヤーが重要なリソースを管理できることを意味する。プレイヤーは砂を使うという投資の最適な場面を判断する必要がある。砂を使う場所はプレイヤーによって異なってくる。戦闘で多く

4.3 ゲームにおける内部経済の活用法

使うプレイヤーもいれば、難しいジャンプパズルにとっておくプレイヤーもいる。このように、砂は多目的なリソースである。プレイヤーは一番必要な場面で能力を増幅するために砂を使えるのだ。

内部経済によって確率空間を大きく

内部経済が複雑になるにつれ、ゲームの確率空間も急速に拡大する。一般に、確率空間の大きいゲームはリプレイバリューが高い。これは、一度のプレイスルーで達成するよりずっと多くの選択肢がプレイヤーに与えられるからだ。また、個人的な体験が得られることもメリットだ。これは、プレイヤーが自分の腕前と選択によって、確率空間のどの部分を開いて探検するかを直接選ぶことができるからだ。

内部経済を使ってキャラクターの育成や、テクノロジー、成長、車両アップグレードなどを管理するゲームでは、内部通貨を使ってプレイヤーに選択肢を与えることが多い。これはロールプレイングゲームで使われる典型的なゲームプレイフィーチャーであり、プレイヤーはゲーム内通貨を使ってキャラクターの装備を整えたり、経験値を使ってスキルや能力を身につけたりできる。内部経済は、レース前後に（ときにはレース中でも）、プレイヤーが車両を調整したりアップグレードしたりできるレーシングゲームでも使われている。十分な選択肢を用意し、選択肢によって立ちはだかる問題に対してまったく異なる解決策を提示したり、他のかたちでプレイヤーにとって重要な選択肢になったりすれば、優れた戦略だと言える。

内部経済を使ってゲームプレイをカスタマイズする際に注意すべきポイントが3つある。第1は、オンラインロールプレイングゲームでは、特定のアイテムないしスキルの組み合わせが他と比べて効率が良い場合、プレイヤーはすぐにそれを発見して情報共有するため、経済のバランスが崩れることだ。どのプレイヤーもこの選択肢を選ぶようになるため、確率空間を実質的に狭めてしまい、ゲーム体験が単調になったり、さもなくば、情報を持っていないプレイヤーが先行するプレイヤーに追いつけないことに不満を持つだろう。このようなゲームでは、カスタマイズ機能は何らかのネガティブフィードバックによってバランスをとることが最適である。

この理由から、ロールプレイングゲームには多数のネガティブフィードバックメカニズムが実装される傾向が強い。たとえば、キャラクターがレベルアップしてスキルを強化すると、次のレベルアップまでにはさらに多くの経験値が必要になる。これによって、事実上レベルや能力の違いが少なくなり、プレイヤーはレベル獲得のために投資を増やす必要性にせまられるのだ。

第2は、確率空間の大きさを十分確保して、プレイヤーが1セッションですべてを探検できないようにすることだ。たとえば、強度、器用さ、賢さという属性をそれぞれ1~5でランク付けし、ゲームを進めながらどの属性のランクを上げるかをプレイヤーが選択できるロールプレイングゲームがあったとする。この場合、この全属性を最大値までアップグレードしなくてはゲームをクリアできないようにすることは、一般にまずいデザインだ。同様に、属性をアップグレードする順番に限られた選択肢しかない、あまり意味のない選択になってしまう。

第4章 内部経済

現実的な影響を及ぼす選択肢を作るには、互いに排他的にすることだ。たとえば、ロールプレイングゲームでは、一般的に、プレイヤーは自分のキャラクターに1つしかクラス（役職、階級）を選べない。各クラスにはユニークなスキルや能力があるべきだ。Deus Exでは、ゲームプレイに大きく影響するサイボーグキャラクターを改良する選択肢もプレイヤーに与えられる。キャラクターを一時的に透明にする光学迷彩か、ダメージへの抵抗力を増す特殊な皮下アーマーか、いずれかのモジュールをインストールするように促されるのだ。

第3に、プレイヤーが異なる戦略を使ってもクリアできるようにレベルをデザインすることが理想である。たとえば Deus Ex では、キャラクターをさまざまな方向性へ成長させることができる。プレイヤーは戦闘もしくは諜報活動にフォーカスしたり、ハッキングにフォーカスしてゲーム中の多くのチャレンジを代替手段で解決できる。これは、ほぼすべてのレベルに複数の解決策が存在することを意味する。このバランスをとるのは容易なことではない。プレイヤーがあるレベルに行く前に3つの選択肢のアップグレードができると想定した場合、戦闘能力を3回アップグレードするケース、諜報活動能力を3回アップグレードするケース、ハッキング能力を3回アップグレードするケース、あるいはこの3つすべてを1回ずつアップグレードするケースを考慮する必要がある。Deus Ex では、アップグレードに必要な経験値の獲得源は一度取得すると消えてしまうため、問題はさらに顕著になる。プレイヤーは、特定のサイドクエストを進めることでそれを獲得することになる。前にいた場所へ戻って、さらに経験値を獲得する選択肢はないのだ。

この例は私たちに、カスタマイズ性が高いゲームではプレイヤーのアバターがどんな能力を持つことになるか分からないため、一般的なアクションゲームよりもレベルに高い柔軟性と多様な解決策が求められることを教えてくれる。シリーズ最新作の Deus Ex Human Revolution（邦題は Deus Ex）には問題がある。プレイヤーはさまざまな手段でゲームをプレイできるのに、ボスキャラを倒す手段は1つしかないのだ。これではアバターをプレイヤーにカスタマイズさせる意味がない。

経済構築ゲームのヒント

建設や管理のシミュレーションによってプレイヤーが経済を構築するゲームでは、巨大で複雑な内部経済が形成される傾向がある。シムシティがその好例だ。プレイヤーはエリアに区画を割り当て、インフラを構築し、こうした構成部品を使って経済構造を構築し、そこからさらなる発展に必要なリソースを生産させる。このようなゲームを作るには、ゲームデザイナーが多数のゲームメカニクスを組み合わせ、ユーザーが面白い遊び方を編み出せるような仕組みを設計しなければならない。これは、完全に機能的なバランスのとれた経済そのものをデザインするよりもさらに大変だ。経済の構成部品の組み合わせ方法をすべて把握しておく必要があるからだ。うまくやれば、こうしたゲームで遊ぶことから大きな満足感を得ることができる。プレイを通じて構築した経済は、プレイヤーの選択と戦略を直接反映させているからだ。これがシムシティで同じ都市が存在しない理由である。

4.3 ゲームにおける内部経済の活用法

経済構築ゲームをデザインする場合、この複雑な作業をコントロールする戦略が3つある。

- **プレイヤーの構成部品すべてを一度に導入しないこと。**建設や管理のシミュレーションゲームでは、通常、経済上の役割を果たす基本ユニットや構成部品で農場や工場、都市などを建設する（シムシティでは区画された土地や専用の建物がこれに相当する）。プレイヤーにはゲーム内のさまざまな要素を少しずつ親切に導入するほうがよい。こうすれば、少なくとも初期は確率空間を制御しやすくなる。導入する構成部品をある程度絞り込めば、シナリオを構成したり、特定のチャレンジを作ったりできる。レベルやシナリオが特でないゲームの場合でも、スタート時点ですべての選択肢を使えるようにはしないことだ。プレイヤーにはリソースを蓄積してから、高度な構成部品を使えるようにして、その後、新しい選択肢のロックを解除すればよい。経済構築ゲームとして優れた例となるシヴィライゼーションでは、ゲームのスタート時には大半の構成部品にロックがかけており、1つ1つロックを解除しないとプレイヤーは使えないようになっている。
- **メタ経済構造に注意すること。**理想的な経済構築ゲームでは、経済の構成部品を組み合わせる方法が無限である。とはいえ、こうしたゲームの大半で、有効な方法とそうでない方法とがあるものだ（勝敗のあるゲームの場合、勝てない方法も存在する）。デザイナーとして、**メタ経済構造**とも呼べる典型的な構成があることを頭に入れておくべきだ。たとえばシムシティでは、工業区画、住宅区画、商業区画の特定の組み合わせが非常に効率が良いことがしだいに判明する。プレイヤーはこの構造を素早く発見して、厳密に従おうとする。こうした主流になりすぎるパターンへの対処法として、難しいが効果のある方法がある。それは、ゲーム初期に効率的であったパターンの効果を、後半では必ずなくすようにすることだ。たとえば、初期には人口を増やす効果のあった区画の組み合わせでも、長い間使うと、多くの公害を引き起こすようにする。時間をかけてゆっくりと効果を発揮するポジティブフィードバックがこの手の効果を作るには有効なメカニズムである。
- **マップを使ってバラエティーを作り、確率空間を制約すること。**シムシティやシヴィライゼーションで都市や帝国を理想的な一等地に作ってもあまり楽しくない。こうしたゲームのチャレンジは、バーチャル環境の初期状態の限界に挑戦することだ。デザイナーとしては、プレイヤーを制約したり、チャンスを与えたりできるようにマップをデザインすること。そうすると、完全に機能的なバランスのとれた経済そのものをデザインするよりもさらに大変だ。経済の最適な方法（絶対優位のメタ経済構造とも呼べるもの）があったとしても、特定の地形ではまったく使いものにならない。これによってプレイヤーは工夫せざるを得なくなり、柔軟性があって融通の利くプレイヤーにリワードを与えられるのだ。シムシティの場合は、プレイヤーが自分の都市で自然災害を起こすことができる。これはプレイヤーが自らの即応力と柔軟性を試す災害シナリオだが、一方で普通にプレイしていても、システムはランダムに災害を引き起こしてプレイヤーの進行を妨害してくる。

第4章 内部経済

4.4 まとめ

本章では、内部経済の不可欠な要素、すなわちリソース、エンティティ、およびこれらを実操作するメカニクス（ソース、ドレイン、コンバータ、トレーダー）を紹介した。グラフに見られる「経済の形」というコンセプトを解説し、メカニクスの構造によって異なる形が生成されることを示した。ネガティブフィードバックは平衡を作り、ポジティブフィードバックは対戦者同士の軍拡競争を作る。ポジティブフィードバックのもう1つの実装方法に下方スパイラルの生成がある。プレイヤーにとって経済を成長させるのがどんどん難しくなるのだ。2人のプレイヤーの関係に基づいたフィードバックシステムは、ゲームを接戦にする効果をもたらしたり、リードしているプレイヤーを勝ちっ放しにしたりする傾向がある。

ゲームデザイナーは内部経済をさまざまな方法で使ってゲームを面白くし、ゲームの進行もプレイヤーの戦略的な選択も豊かにできる。内部経済は、マルチプレイヤーゲームにおいてレベルの異なるプレイヤーや力が拮抗したプレイヤーの競争にも影響を与えられる。本章の最後では、シムシティのような経済構築ゲームの構築方法について提案を行った。

4.5 演習

1. 公開されているゲームを使って、その中のリソースと経済機能を特定しよう（インストラクターが研究対象のゲームを指定する場合もある）。
2. 周期的平衡のあるネガティブフィードバック、下方スパイラル、短期投資と長期投資のトレードオフ、プレイヤーの相対的スコアに基づいたフィードバック、ラバーバンド効果などの特徴を示すゲームを挙げよう（本章で取り上げたゲーム以外）。どのようなリソースが使われているかを説明し、発見した効果や特徴がどのようにゲームメカニクスによって生成されているかを明確にしよう。
3. デッドロックが起こる可能性のあるゲームの例を見つけよう（ゼルダの伝説以外）。そのゲームにデッドロック状態を覆す手段はあるだろうか。説明してほしい。

第 5 章 マキネーション

第 4 章では、ゲームの内部経済がいかにゲームメカニクスの重要な側面の 1 つであるかを明らかにした。また、図を使って経済の構造とその効果を可視化した。本章では、ゲームのメカニクスについてのこの視点を形式化する手法として、マキネーションという視覚言語フレームワークを紹介する。マキネーションは、ゲームの内部経済の作成、ドキュメント化、シミュレーション、テストを行うゲームデザイナーやゲームデザインを学ぶ学生を支援するためにヨリス・ドーマンズが考案したものだ。このフレームワークの中心は、ゲームの内部経済をビジュアルで表現したマキネーションダイアグラムである。マキネーションダイアグラムのアドバンテージは、明確に定義された構文を持っていることだ。マキネーションダイアグラムを使うと、明確かつ一貫した方法でデザインを記録し、伝えることが可能になる。

本書では全体にわたってマキネーションダイアグラムを使うため、読者の方がこのダイアグラムの読み方を身につけることは重要だ。本章では、マキネーションダイアグラムをマークアップする大半の要素を体験してもらおう。だが、注意点が 1 つある。マキネーションフレームワークを一気に学ぶのは大変である。このフレームワークには、あわせて理解すべき相互に関連した概念が多く存在する。したがって、これらの概念を説明するにあたり、理解しやすくなる「正しい」順番というものは存在しない。論理的な順番でマキネーションダイアグラムの要素を説明するように心がけたが、ときには、前の概念に戻ったように思うこともあるだろう。

マキネーションは、ダイアグラムを作成する視覚言語以上のものである。ドーマンズは、マキネーションダイアグラムを描いてリアルタイムでシミュレーションできるオンラインツールを構築した。これを使うと、マキネーションダイアグラムを簡単に構築・保存したり、ゲームの内部経済の振る舞いを研究したりできる。このツールは、www.jorisdormans.nl/machinations で入手できる。

マキネーションツールのチュートリアルはオンラインで入手できる（詳細については付録 C を参照）。マキネーションダイアグラムで最重要になる要素についてのクイックリファレンスは付録 A に記載した。