Engineering Emergence: ゲームデザインのための応用理論

Joris Dormans 著

アムステルダム大学 博士論文（2012年）

Joris Dormans, 2012. ISBN: 978-94-6190-752-3

審査委員会

- 主任指導教員（Promotor）：prof. dr. ir. R.J.H. Scha
- 副指導教員（Co-promotor）：dr. ir. J.J. Brunekreef
- その他の委員（Overige leden）：
    - dr. ir. A.R. Bidarra
    - prof. dr. P. van Emde Boas
    - prof. dr. P. Klint
    - prof. dr. ir. B.J.A. Kröse
    - dr. M.J. Mateas
    - prof. dr. E.S.H. Tan

人文学部（Faculteit der Geesteswetenschappen）

序文

ゲームの国の中心に位置するPhD山の頂上への私の道のりは、長く素晴らしい旅であった。その多くの回り道の途中で、私は多くの偉大で刺激的な人々に出会った。短い間だけ出会った人もいれば、しばらく一緒に旅をしてくれた人もいたが、誰もが貴重な助言と揺るぎない支援で私の旅を明るく照らしてくれた。この曲がりくねった道を振り返ると、すべての紆余曲折が今日の私をこの場所に導くために必要であったことが今ではわかる。

私はゲームの国の原住民であると言えるかもしれない。子ども時代、私はあらゆる種類のゲームで膨大な時間を費やした：ボードゲーム、ペンと紙のロールプレイングゲーム、そしてビデオゲーム。そして私たちは最初から自分たち自身のゲームを発明していた。野心的すぎるゲームのためのボードをデザインしたり、終わりなきロールプレイングセッションのためのダンジョンを作ったり、Commodore　64で結局完成しなかったゲームの計画を立てたりした紙は数え切れない。

大人になり学生になると、私はゲームの国を離れ、それが最も長い回り道であることが判明するものへと向かった。何年もの間、私はアルファ大洋の向こう側にある立派な教育の殿堂で過ごした。この期間に、最終的にここへの旅に導く信頼できるツールとなる知識とスキルを獲得した。

この海岸に戻ってくることは喜びであった。私はより賢くなり、私の視野は研ぎ澄まされていた。いくつかのことは変わっていたが、多くの他のことは同じままであった。すぐに私はどこに行きたいかを知った。正しい道を見つけることは必ずしも容易ではなかったが、今では成功したと言える。この旅に沿って支援、助言、そして仲間を提供してくれた多くの人々に感謝しなければならない。

私の母Claartjeは、意志の強さと固い決意を通じて、自分の運命を形作り、道に現れるどんな山でも登ることができることを幼い頃に教えてくれた。お母さん、何が起きても、あなたの内なる強さは永遠に私の導きの光であり続けるでしょう。

Marijeは私がこの道に出発するずっと前から私の愛するパートナーであり、ずっと一緒にいてくれた。今、あなたはこの旅の結論において私のパラニンフとして隣に立っている。あなたの精神的な支えとテキストに関する助言がなければ、私はこの高みに到達することはなかっただろう。

私の指導教員Remkoは、メンターの役割を見事に果たしてくれた。ゲームの国に対する相対的な部外者であったにもかかわらず、私が軌道修正する必要があるときに、すべての正しい質問をすることを怠らなかった。

私の副指導教員Jacobは、エンジニアリングの海岸をうまく航海するための正しい地図を提供してくれた。その地図がなければ、私はこれほど遠くまで旅することはできなかっただろう。

委員会の他のメンバー、Ben、Ed、Michael、Paul、Peter、Rafaelにも感謝する。あなた方の中には遠い地で出会った人もいれば、道中で出会った人もいる。最後の登頂の際に同行する時間を取ってくれたことに非常に感謝している。

アムステルダム応用科学大学の同僚たちほど陽気な旅の仲間を望むことはできなかっただろう。私たちはこの山に導く曲がりくねった道で多くの冒険的な日々を共有した。私が設定した道筋への、そして他の人々が容易に後を辿れるように道を舗装することへのあなた方の信頼と支援は、触れないわけにはいかない。

私の学生たちは、特に私の旅の一部を再現しているときに、私を驚かせることを怠らなかった。彼らは常に自分自身のルートを選び、それは私にとってインスピレーションの源であった。

私の進歩に対する関心が誠実で支持的であった家族や友人にも感謝しなければならない。かつて多くのよくプレイされたゲームで対戦相手であったあなた方すべてに深くお辞儀をする。

この旅に沿ったすべての人々の中から、もう二人に感謝する。まず、しばらく海で迷った後にこの海岸に上陸することを最終的に可能にしてくれたすべての正しい人々に私を紹介してくれたCarla。そして最後に、しかし決して軽視するのではなく、子ども時代にまで遡る友情を持つ私のもう一人のパラニンフJasper。私たちが楽しんだ多くのゲームは、これらの土地の記憶を生き続けさせ、私の帰還への道を切り開いた。

第1章 序論

> ゲームを魅力的にするものを説明する単一の文は存在しない。
>
> — Jesper Juul (2003)

ゲームをデザインすることは難しい。ゲームは非常に長い間存在してきたが、過去数十年にわたるコンピュータゲーム産業の台頭が、この問題を顕在化させた。その短い歴史の中で、コンピュータゲーム産業は個人の開発者や小さなチームから、数百人の従業員を伴う数百万ドル規模のプロジェクトへと成長した。現代のゲーム産業には、ミスのための余地がほとんどない：財政的な賭け金があまりにも高くなっている。今日ほど、そのようなミスを防ぐために成功するゲームをデザインするプロセスをよりよく理解する必要性が高まったことはない。ゲームデザイナーをその仕事において支援するためのより優れた応用理論と知的ツールが必要である。

同時に、これまで以上に多くの人々がビデオゲームをプレイしている。より広い観客は、質の高いゲームプレイを持つゲームに対する需要が増え続けていることを意味する。ゲームプレイヤーがより経験豊富になるにつれて、より洗練されたゲームへの食欲が増す。他の芸術やメディアの形態と比較して、コンピュータゲームはかなり最近の発明である。開発とイノベーションの余地はまだ十分にある。

この論文の一般的な前提は、ゲームのデザインにおける困難は、一方では多くの創発的特性を示す複雑なルールベースシステムとしてのゲームの性質の中にあり、他方ではよく設計された自然な流れのユーザー体験を提供しなければならないということにある。これらの困難に直面して、ゲームデザイナーのツールボックスはかなり空っぽである。ゲームの性質と創発的振る舞いは十分に理解されていない。レベルデザインは、ゲームの創発的振る舞いを制御するための一つの方法であり、ゲームプレイをストーリーラインによって緩くつなぎ合わされた一連の比較的単純なタスクに制限する。しかし、質の高いコンテンツは制作にコストがかかる。多くの手作りのレベルを持つゲームは制作にコストがかかり、ルールシステムから生まれるオープンなゲームワールドの真の表現力を活用できていない。

この論文は、ゲームにおける創発の性質を検討し、ゲームにおける創発に正面から対処するための応用理論を構築する。この理論は、デザイナーが創発的振る舞いを示す質の高いゲームを構築するという捉えどころのないプロセスに対してより多くのグリップを得るこ

とを可能にするだろう。この論文で開発された理論は、ゲームメカニクスとレベルに適用される。しかし、多くの学者やデザイナーがレベルとメカニクスをゲームデザインの二つの大きく異なる要素として扱うのに対し、この論文はその二つを統合しようと試みる：ルールとレベルの両方について、この論文はゲームのこれらの側面をデザインするプロセスが高められ統一されうるような形式的で抽象的な表現を見つけようとする。これらの表現を通じて、ゲームデザイナーが扱う素材はより具体的なものとなるはずである。これが、この論文の中心的な研究課題につながる：ゲームルールとゲームレベルのどのような構造的特質が、創発的ゲームプレイのデザインを支援するための応用理論とゲームデザインツールの作成に使用できるか？

ゲームデザイナーをその仕事で支援するためのソフトウェアツールの開発は、この論文の重要な側面である。あまりにも多くの場合、デザイン理論はゲームデザインの実践から孤立して生み出されてきた。ここで提示された理論を実装するソフトウェアツールの作成は、これらの理論が非常に具体的で適用可能であることを意味する。さらに、それはデザインプロセスの特定の部分の自動化を可能にする。これらの部分を自動化することにより、デザイナーはその仕事を支援され、人間の創造性と独創性を最も必要とするデザインの側面に集中することができる。

この章では、この論文の中心的な概念を紹介する：ゲーム、ゲームプレイ、メカニクス、レベル、創発、そしてプログレッション。また、一般的なアプローチと後続の章の内容も概説する。

1.1 ゲーム

ゲームとは何か？ 多くの人がゲームをプレイするが、その性質について立ち止まって考える人はほとんどいない。ゲームの研究、特にその現在の形でのゲームの研究は非常に若い。Espen Aarsethがその年をゲーム研究の「元年」と宣言したのは2001年のことであった（Aarseth, 2001）。その年は、最初の査読付きオンラインジャーナルの立ち上げと、ゲームに特化した最初の国際学術会議が行われた。ゲームは以前にも研究されていたが、ゲーム研究が独立した学術分野として認識されるのに十分な勢いを得たのは2001年まではなかった。

ゲームの研究は最初からマルチディシプリナリーな取り組みであり、異なる分野の研究者が異なる視点からゲームを研究してきた。ゲーム研究の最初の数年間は、ナラトロジストとルドロジストの間の激しい論争によって特徴づけられた。前者のグループは、しばらくの間その視点からゲームを研究してきた、文学のバックグラウンドを持つことが多い学者

で構成されていた。彼らはゲームをストーリーテリングのための新しいメディアとみなし、文学やメディア研究の文脈にゲームを位置づけた（Laurel, 1986; Murray, 1997; Ryan, 2001）。ルドロジストはこの立場に反対した。彼らにとって、ゲームは第一にルールに基づくプレイ体験である。ストーリーとビジュアルはルールに対して二次的であり、ルールこそがゲームの質にとって最も重要な要因である。彼らの主張は、良いルールとより低い品質のビジュアルとストーリーを持つゲームでも良いゲームになるが、その逆は成り立たないというものであった（Eskelinen, 2001; Juul, 2005）。

今日では、両方の立場はかなり極端であると考えられている。文学や映画で物語を研究するために使用されるパラダイムがゲームに直接適用できると主張する人を見つけることは困難である。いかなるゲームの研究においても、ルールとゲームプレイを無視することはできない。一方で、テトリスをRaph Kosterの仮想的な「Mass Murderer Game」（図1.1参照）にリスキニングすること——プレイヤーが不器用な形の死体で穴を埋めようとする——は、ストーリーとビジュアルがプレイの体験に確かに影響を与えることを明確に示している（Koster, 2005b, 166-169）。September 12のようなゲームの辛辣な皮肉——プレイヤーがアラブの都市でテロリストにミサイルを撃つよう招かれ、その行動の結果を探求する——は、単純で典型的なゲーム的なシューティングアクションを支えるルールと、ゲームの外の非常に現実的な状況へのゲームの意味のある参照との間の鋭い対比によってのみ可能となっている（図1.2参照）。ゲームは孤立して存在するのではなく、物語が意味を引き出す異質なメディアランドスケープと社会構造の一部である。この場合、September 12の開発者であるGonzalo Frascaが、著名なルドロジストでもあり、事実上ルドロジーという用語を生み出した最初のゲーム研究者であったことは注目に値する（Frasca, 1999）。

> **図1.1：** テトリスに基づくRaph Kosterの仮想的なMass Murderers Game。
>
> **図1.2：** 風刺的なSeptember 12でテロリストにミサイルを撃つ。

Mass Murderer GameとSeptember 12の例は、ゲームにおいて最も重要なのはプレイヤーが何を「する」かであることをうまく示している。これらのアクションは一方ではルールによって決定され、他方ではゲームのアートとストーリーがこれらのアクションを枠づけし、意味を与える。ゲームにおいて、ルールは可能なインタラクションを設定するが、巧みなレベルのデザインを通じて開発者はプレイヤーがゲーム要素に遭遇する順序や、それらが突きつけるチャレンジに対してある程度のコントロールを持つ。開発者がアクションの順序を主にコントロールするのは、レベルを通じてである。

ゲーム研究における一般的なコンセンサスは、ゲームとはルールベースのアーティファクトであり、1人以上のプレイヤーが体験するように設計されており、何らかの種類のゴールを達成しようとするものである、ということである。ルールがなければゲームは存在しないが、プレイヤーの構造化された体験は他のメディアで遭遇する構造や体験と似ていないわけではない。

歴史家のJohan Huizingaからゲームデザイナーのgreg Costikyan、Katie SalenとEric Zimmermanに至るまでの8つの異なるゲームの定義を綿密に検討した後、SalenとZimmermanはゲームを以下のように定義している：

> 「ゲームとは、プレイヤーが人工的な葛藤に従事するシステムであり、ルールによって定義され、定量化可能な結果をもたらす。」（2004, 80）

彼らの定義において、システム、プレイヤー、人工性、葛藤、ルール、そして定量化可能な結果が鍵となる概念である。すべてのゲームは、複雑な全体を形成する多くの部分から成るシステムである（Salen & Zimmerman, 2004, 55）。システムは、プレイヤーが何ができ何ができないかを決定するルールによって定義される。それらのルールに従って、プレイヤーは互いに対して、またはゲームシステムに対して葛藤に従事する。葛藤はゲームが現実の生活とは別に設定されているという意味で人工的であり、時間と空間の両方において、プレイヤーがゲームのルールに従う空間である。この意味で、ゲームはJohan Huizinga（1997）の仕事に倣って、「魔法の輪」の中で行われるとよく言われる。最後に、ゲームには定量化可能な結果がある：プレイヤーは勝つか負けるか、あるいは何らかのスコアで自分のパフォーマンスを測定することができる。

SalenとZimmermanの定義は、彼ら自身が調査していないものも含め、多くの他の定義と類似している。Mark J. P. Wolfは葛藤、ルール、プレイヤーの能力、価値づけられた結果をゲームを定義するために使用している（2001，14）。Alexander Gallowayは「ゲームとは、プレイヤーが何らかの種類のゴールに到達しようとするルールによって定義される活動である」と述べている（2006, 1）。Ernest AdamsとAndrew Rollingsはルール、プレイ、ゴール、ふり（pretending）をゲームの主要な要素として特定している。後者の要素は魔法の輪に結び付けられ、延いてはSalenとZimmermanの人工性の概念に結び付けられる（Adams & Rollings, 2007, 5-11）。Tracy Fullertonにとってゲームとは「プレイヤーを構造化された葛藤に従事させ、不均等な結果においてその不確実性を解決する、閉じた形式的なシステム」である（2008, 43）。

Jesper　Juulは、SalenとZimmermanの定義を含む多くの同じゲームの定義を検討している。彼は、以下の6つの特徴がゲームを定義すると結論づけている（Juul, 2005, 36）：

1. ゲームはルールに基づいている、
2. そして可変的で定量化可能な結果を持つ、
3. それはプレイヤーの努力によって影響される、
4. そしてそれに異なる価値が割り当てられる、
5. そしてプレイヤーは感情的に愛着を持つ、
6. そして結果は交渉可能である。

これら6つの特徴のうち、最初の3つだけが形式的なシステムとしてのゲームの特性である。残りの3つは、ゲームとプレイヤーの間の関係、またはゲームと外の世界との間の関係のいずれかの特性である（Juul, 2005, 37）。

SalenとZimmermanの定義と比較して、Juulの定義はいくつかの追加的要素を組み込んでいる。まず、Juulの定義ではゲームの結果は定量化可能であるだけでなく、可変的でもある。ゲームはゲームとして機能するためには異なる結果を持たなければならない。ゲームが常に同じ結果になる場合、プレイする意味がほとんどない。これは、大きく異なるレベルの二人のプレイヤーが競っている場合に起こりうる。一方が必ず勝つ場合、開始前に結果がわかっている場合、ゲームはゲームとして機能しなくなる。次に、Juulにとって、プレイヤーは努力を投入することによってゲームに影響を与えることができなければならない。努力なしには、プレイヤーのアクションは無意味であり、プレイヤーはゲームの結果に感情的に愛着を持つことはない。Juulにとって、これはプレイヤーが結果にいかなる方法でも影響を与えることができない純粋な偶然のすべてのゲームをボーダーラインケースにする（Juul，2005，44）。第三に、Juulはゲームとプレイヤーの間の関係、そしてゲームと世界の間の関係により多くの注意を払っており、それは彼の最後の4つのポイントに示されている。

同時に、JuulはSalenとZimmermanの定義から人工性を省いている。SalenとZimmermanの定義において、人工性は「魔法の輪」の概念に類似した役割を果たしている——ゲームが現実の生活の外側に現実を作り出すJohan　Huizingaの概念である（Huizinga，1997）。魔法の輪は間違いなく多孔質であるが（Copier，2007）、ゲームの人工的な性質は疑いのないものである。ゲームルールはデザイナーによって作られ、プレイヤーによって支持されて体験を作り出す。プレイヤーはゲームを体験するためにこれらのルールに従う。ゲームはまた、ゲームに先立って存在するルール——ほぼすべてのスポーツを制約する重力の法則など——によっても制約されるが、すべてのゲームは人工的

なゴール、葛藤、チャレンジを設定するためにルールを追加する。ゲームは、その空間が明確な境界を持つか否かにかかわらず、ゲームがプレイされる空間を作り出す。

この論文では、私はSalenとZimmermanの定義の上に構築することを選択する。ただし、ゲームの可変的な結果に影響を与えるJuulのプレイヤーの努力と能力を追加する。Juulの他の追加は、この論文が形式的なシステムとしてのゲームに焦点を当てており、ゲームとプレイヤーの間の関係には焦点を当てていないため、除外することを選択する。したがって、この論文では、ゲームは以下のように定義される：

> ゲームとは、プレイヤーが人工的な葛藤に従事するシステムであり、ルールによって定義され、プレイヤーの努力と能力によって影響される可変的で定量化可能な結果をもたらす。

ゲームプレイは、プレイヤーのアクションとプレイの体験に関連する重要な概念であり、定義するのがより困難である。ゲームプレイはなんとなくプレイヤーがすることから成る。同時に、この用語はゲームそのものが持つ質を記述するためにも使用される。ゲームのレビュアーはこの意味でゲームプレイについてよく語る。このように使用されると、「ゲームプレイは良いゲームプレイの同義語になった」とNiels 't Hooftがかつて述べたように。^1　つまり、ゲームがゲームプレイを持つかどうかが、その質の評価になっているのである：良いゲームにはゲームプレイがある。

> ^1 Niels 't HooftはBasher.nlとオランダの新聞NRC Nextで活動するフリーランスのゲームジャーナリストである。彼は2011年2月2日にアムステルダムのPakhuis de Zwijgerで開催されたGameLabのゲームプレイに関する会合でこの発言をした。

この論文では、私はこの意味でゲームプレイを使用する。デザイナーがゲームプレイを作ることに取り組んでいるとき、彼らは常に魅力的なゲーム体験を作ることに取り組んでいる。製品としてのゲームは、この体験の主要な源である。ここから導かれるのは、ゲームプレイはゲームが構築されるやり方からなんとなく生まれるということである。ルールベースシステムとしてのゲームのこれらの構造的特質こそが、この論文の焦点である。

1.2 メカニクス

ゲームデザインコミュニティがゲームシステムについて語るとき、彼らは「ゲームルール」よりも「ゲームメカニクス」という用語を好む。「ゲームメカニクス」はしばしばルールの同義語として使用されるが、この用語はより正確さを含意し、通常は実装により

近い。ここでの実装はまだいかなるプラットフォームやメディアからも比較的独立しているものの。ゲームデザイナーのErnest AdamsとAndrew Rollingsは、以下の例で両者の違いを説明している：ゲームのルールは、毛虫はカタツムリより速く動くと規定するかもしれないが、メカニクスはその違いを明確にする。メカニクスは毛虫がどのくらい速く動き、カタツムリがどのくらい速く動くかを指定する。メカニクスは、ゲームプログラマーが混乱なくコードに変換できるほど、あるいはボードゲームのプレイヤーが失敗なく実行できるほど正確である必要がある。メカニクスは必要な詳細をすべて指定する（Adams & Rollings, 2007, 43）。

同様に、Morgan McGuireとOdest Chadwicke Jenkinsは「メカニクスはゲームプレイを生み出す数学的機械である。それらは抽象的なゲームを作り出す」と述べている（2009, 19）。これによって彼らは、メカニクスはメディア非依存であることを指摘している：メカニクスは画像や音声から分離可能なゲームの部分の中にあり、実際にあるメディアから別のメディアへと移植される可能性がある。ボードゲームが、異なるアートと異なるテーマを持つコンピュータゲームとして、メカニクスを変更することなく再現されるかもしれない。

ゲームデザイナーは、単数形で「ゲームメカニック」について語ることに完全に慣れている（McGuire & Jenkins, 2009; Brathwaite, 2010）。これは、ゲームメカニクスの扱いに熟練した人物を指しているのではない——単数形の「mechanic（メカニック）」の一般的な使用法が暗示するように。^2　むしろ、彼らは特定のゲーム要素を支配する単一のゲームメカニズムを指している。そのようなメカニズムの一つは、いくつかのルールを含むかもしれない。例えば、横スクロールプラットフォームゲームにおける動くプラットフォームの「メカニック」は、プラットフォームの移動速度、クリーチャーがその上に乗れること、乗っているときに一緒に移動すること、だけでなく、他のゲーム要素にぶつかるとプラットフォームの速度が反転すること、あるいは特定の距離を移動した後に反転すること、も含むかもしれない。この論文では、私は単一のゲーム要素やインタラクションに関連する単一のゲームルールのセットを示す単数形として「メカニズム」を使用することを好む。

> ^2 Oxford Advanced Learner's Dictionaryは「機械の操作や修理に熟練した労働者」をmechanicという語の唯一の意味として記載している。

いくつかのメカニクスは他のものよりゲームにとってより中心的であるかもしれない。「コアメカニクス」という用語は、プレイヤーが最も頻繁にインタラクトし、ゲームプレイに最大の影響を与えるメカニクスを示すためにしばしば使用される（Adams & Rollings, 2007; McGuire & Jenkins, 2009）。移動とジャンプは、例えば、ほとんどの

プラットフォームゲームのコアメカニクスである。対照的に、プレイヤーが100個の星を集めるごとに1つのエクストラライフを獲得するというメカニクスは、ゲームのコアであると見なされるかもしれないし、そうでないかもしれない。エクストラライフが単なるちょっとしたボーナスであるゲームでは、それはおそらくコアメカニズムではないが、星が豊富でプレイヤーが簡単にライフを失うゲームではおそらくコアメカニズムである。コアメカニクスと非コアメカニクスの間の区別は明確ではない。同じゲームであっても、何がコアで何がそうでないかの解釈は、デザイナー間で、あるいはゲーム内の異なる瞬間の間で変わりうる。

メカニクスは、ゲームにおいて多くの異なるタイプのルールを示すようになってきた。この用語は時として、物理学の意味でのメカニクスを指す：運動と力の科学。ゲームにおいて、キャラクターは一般的に移動し、ジャンプし、乗り物を運転する。ゲーム要素がどこにあるか、どの方向に動いているか、他の要素と交差しているか衝突しているかを知ることは、多くのゲームにおけるすべての計算の大部分を占める。ここでのメカニクスは、ゲーム内の運動と力を支配する物理法則の実装として文字通り解釈されるかもしれない。同時に、ゲームには物理学とは何の関係もないメカニクスも含まれる：例えば、エクストラライフを獲得するためにいくつのコインを集める必要があるかを指定するメカニクス。パワーアップ、コレクティブル、その他のタイプのゲームリソースを扱うメカニクスは、内部経済と呼ばれうるものを構成する（Adams & Rollings, 2007, 331-340）。経済メカニズムとゲーム物理学の性質は、いくつかの重要な点で異なる。両方にメカニクスという用語を使用することの一つの問題は、これらの重要な違いを曖昧にすることである。

現代のゲームにおける物理学は、ほぼ連続的なゲームシミュレーションを生み出す正確なメカニクスでシミュレートされる傾向がある。ゲームオブジェクトが半ピクセル左右に位置するかもしれず、これがジャンプの結果に大きな影響を与えるかもしれない。対照的に、内部経済のルールは離散的である傾向がある。ゲーム要素とアクションは有限のセットであり、緩やかな遷移を許さない：ゲームにおいてパワーアップの半分を拾うことは通常できない。ゲーム物理学のこの連続的な性質対ゲーム経済の離散的な性質は、メディアの（非）依存性、プレイヤーインタラクションの性質、そしてデザインとイノベーションの機会にさえ影響を及ぼす。これらの効果は以下で論じる。

その連続的な性質のため、物理学の実装はゲーム経済よりもメディアやプラットフォームにはるかに密接に結びついている傾向がある。経済メカニクスはゲームのメディアから確かに分離可能であるが、物理学はそれほどではない。例えば、物理学に大きく依存するゲームはボードゲームとして容易にメディエートすることはできない。スーパーマリオブラザーズのボードゲームを作ること（図1.3参照）——ゲームプレイがプラットフォームか

らプラットフォームへの移動とジャンプから生まれる——は非常に難しい。プラットフォームゲームの連続的な物理学はボードゲームの離散的な性質にうまく変換されない。サイコロには限られた面しかなく、ゲームをアクセスしやすく保つためには過度に複雑な計算は避けるのが最善である。プラットフォームゲームでは身体的な器用さが重要であり、ちょうど全体的な身体的スキルの無数が誰かが実際のサッカーをうまくプレイするかどうかを決定するように、それらのスキルはボードゲームでは失われるだろう。スーパーマリオブラザーズはおそらく、プレイヤーの実際の走りとジャンプの能力をテストする身体的なコースとしてメディエートされるほうがよい。要点は、特定のアイテムを拾った後に二倍の高さジャンプできるというルールは異なるメディア間で容易に翻訳できるが、ジャンプの物理学を実装するルールはそうはいかないということである。ゲームの物理的メカニクスは、ゲームの経済を支配する離散的なルールよりもメディアにより密接に結びついているように思われる。

> **図1.3：** スーパーマリオブラザーズにおけるゲームプレイは、走りとジャンプの連続的な物理学から大部分生まれている。
>
> **図1.4：** ボルダーダッシュにおける物理学は離散的なシステムを通じて実装されている。ダイヤモンドや岩のようなオブジェクトは常にゲームのタイルグリッドに整列している。

興味深いことに、プラットフォームゲームや他の初期のアーケードゲームの初期の歴史を振り返ると、物理学はかなり異なって、はるかに離散的に扱われていたことがある。ドンキーコングの動きはスーパーマリオブラザーズよりもはるかに連続性が低かった。ボルダーダッシュ（図1.4参照）では、重力は1フレームごとに1タイルの一定速度で岩を落下させることによってシミュレートされている。ゆっくりプレイすることになるかもしれないが、ボルダーダッシュのボードゲームを作ることは可能である。その当時、ゲームのメカニクス（物理学的な意味での）を生み出すルールは、他のタイプのゲームルールとそれほど異なってはいなかった。しかし時代は変わった。今日、プラットフォームゲームの物理学は非常に正確で詳細になり、ボードゲームで表現することは不可能になった、あるいは少なくとも非常に不便になった。

離散的なルールでは、先を見通し、計画を立て、複雑な戦略を作成し実行することが可能である。これは容易である必要はないが、可能であり、プレイヤーはそうすることを奨励される。このタイプのルールとのプレイヤーインタラクションはより戦略的なレベルで行われる。一方で、プレイヤーがゲームの物理学を（シミュレートされたものであれそうでなければ）把握すると、直感的に動きと結果を予測することができるが、確実性は低い。スキルと器用さがインタラクションのより重要な側面になる。この違いは、ゲームを教育

に使用する場合に極めて重要である。Angry Birds（図1.5参照）は物理学について楽しく学ぶことでシリアスゲーム賞を受賞した。Angry Birdsが楽しく物理学を含んでいることに疑いはないが、プレイヤーが本当に力、重力、運動量について科学教育に適用可能な方法で学ぶかどうかは疑わしい。Angry Birdsのプレイヤーはそれらの側面に戦略のレベルよりもスキルのレベルで主に関わっている。彼らは力、重力、運動量の効果に対する直感的な感覚を発達させるかもしれないが、それは真にそれらを理解することとは全く同じではない。Angry Birdsにおける戦略は、離散的なルールによって支配されるゲームの側面を含む。

> **図1.5：** Angry Birdsではプレイヤーは鳥を発射して石、木、ガラスの構造物で守られた豚を破壊する。
>
> **図1.6：** World of Gooではプレイヤーは限られた供給の「グーボール」からタワー、橋、その他の構造物を建設する。

プレイヤーは、利用可能な鳥の数とタイプをどのように使用して豚の構造物を最も効果的に攻撃するかを計画しなければならないだろう。これは弱点を特定し攻撃の計画を立てることを要するが、実行自体はスキルに基づいており、効果は決して完全に予見できない。World Goo（図1.6参照）と比較してみよう。そこではプレイヤーは限られた供給のグーボールから構造物を建てる必要がある。重力、運動量、重心のような物理的概念はこのゲームのメカニクスにおいて重要な役割を果たしている。確かに、プレイヤーはWorld of Gooをプレイすることでこれらの概念に対する直感的な理解を形成するかもしれない。しかし、もっと重要なのは、プレイヤーは最も重要な（そして離散的な）リソース——グーボール——をどのように管理し、それらを使って効果的な構造物を建てるかを学ぶことである。Angry BirdsとWorld of Gooの違いは、連続的な物理学の効果を考慮するとき非常に明確になる。Angry Birdsでは1ピクセルの違いがクリティカルヒットか完全なミスかに変わりうるのに対し、World Gooではその効果はより小さい。後者のゲームでは、配置はピクセル精度ではない：グーボールを少し左右に放しても、結果の構造物は通常同じであり、バネがボールを同じ場所に押し込む。ゲームはプレイヤーがボールを放す前に、どの接続が形成されるかさえ視覚化する（図1.6で見られるように）。どちらのゲームがより楽しいかを論じるつもりはないが、プレイヤーはAngry Birdsで物理学について学ぶよりも、World of Gooで構造について多くを学ぶと言えるだろう。

物理学と経済は、ゲームにおいてデザインとイノベーションにも異なる影響を与える。ゲームとジャンルが進化するにつれて、物理的メカニクスはすべてゲームジャンルと密接に対応するいくつかの方向に進化していると言えるかもしれない：ほとんどの場合、ファーストパーソンシューターの物理学を完全に変更する意味はほとんどない。^3 実

際、ゲームが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ますます使用するようになるにつれて、その部門でイノベーションの余地は少なくなっている。一方で、すべてのゲームはユニークなコンテンツを作ろうとしており、多くのファーストパーソンシューターは競合他社とゲームプレイを差別化するためにパワーアップやアイテムのユニークなシステム、あるいはアイテムの経済を確かに作り出している。創造性とイノベーションの余地があるのは、これらの経済を支配するメカニクスにあるのであって、ゲームの物理学にあるのではない。

> ^3 ただし、Portalのような特定のゲームは、確立されたジャンルに革新的な物理システムをうまく導入している。

それでも、コンピュータゲームの歴史を40年振り返ると、物理学がゲームにおける他のどのタイプのメカニクスよりもはるかに速く進化してきたことを認めなければならない。物理学は比較的進化しやすい。なぜなら、ニュートン力学とますます増大する計算能力にアクセスできるからである。同じ解決策は他のタイプのメカニクスには適用されない。すべてのゲームルールを「メカニクス」と呼ぶことは、すべてのタイプのルールが同じように理解できるという事実から開発者の注意をそらすかもしれない。さらに悪いことに、開発者はより詳細なルールとより多くの処理能力を投入し続ける限り、それらの他のタイプのメカニクスもうまくいくと誤って仮定するかもしれない。メカニクスという用語は不幸な誤称である。まさにそれが、異なるタイプのゲームルールの適切な理解の発展を妨げているかもしれないからである。物理的メカニクスの一部ではないルールの人工的で離散的な性質に目をつぶらせるかもしれないが、それらはゲームを真に賢く、ユニークにするものの同様に重要な側面である。非物理的で離散的なメカニクスのための堅固な理論的フレームワークなしには、あるポイントを超えてそのタイプのメカニクスを進化させることは困難である。

私はこの論文を通じて、ゲーム産業内で慣習的であるように、メカニクスという用語を使い続ける。しかし、この用語を使用する際、私は連続的な物理的運動のメカニクスに言及するよりも、ゲーム経済を生成する離散的なメカニズムに言及することの方が多い。適切な場合には、メカニクスがすべてのタイプのゲームに等しく影響するわけではないため、これらと他のタイプのルールを区別する。

1.3 ゲーム分類

どのタイプのルールが特定のゲームのゲームプレイを駆動するかは、ゲームとジャンルの間で大きく異なる。一部のゲームは主に経済からゲームプレイを引き出し、他のゲームは

物理学、レベルプログレッション、戦術的機動、あるいは社会的ダイナミクスからゲームプレイを引き出す。ゲーム産業やゲームジャーナリストによる異なるジャンルでのゲームの分類は、通常ゲームプレイのタイプに基づいており（Veugen, 2011, 42）、したがって延いてはこれらのジャンルで多かれ少なかれ顕著に特徴づけられる異なるタイプのルールに基づいている。図1.7は、典型的なゲーム分類スキームと、これらのジャンルおよびそれに関連するゲームプレイが異なるタイプのルールシステムにどのように関連するかの概要を提供する。ただし、この分類は多くのうちの一つであることに注意されたい。いくつかの分類スキームの間には、深刻なコンセンサスの欠如がある。ここでのポイントは、決定的なジャンル分類を提示することではない。むしろ、異なるタイプのルールが異なるタイプのゲームプレイにどのように相関するかを示すことである。この基本的な分類から派生できる、はるかに多くのジャンルとサブジャンルがある。例えば、ファーストパーソンシューターはアクションゲームの特定のサブジャンルであり、アクションアドベンチャーゲームはアクションとアドベンチャーのゲームジャンルの一般的なハイブリッドである。^4

> ^4　実際、アクションアドベンチャーは非常に一般的であるため、ほとんどの他のジャンル分類において独立したジャンルを構成している。

> **図1.7：** Adams & Rollings (2007)から取られたゲームジャンルを5つの異なるタイプのゲームルールまたは構造と相関させたもの。輪郭の太さと暗さは、そのジャンルのほとんどのゲームにとってのそれらのタイプのルールの相対的な重要性を示している。

図1.7において、私は5つの異なるタイプのメカニクスを区別している。これらのタイプのメカニクスの境界はそれほど厳密ではなく、一つのゲームが複数のタイプのメカニクスを持つことができる。この図は、ゲームジャンルを越えて頻繁に見られるこれらのタイプのメカニクスの典型的な構成を示しているが、各個別のゲームが独自のユニークなゲームメカニクスの構成を持ちうることは明らかであるべきである。物理学と経済のメカニクスはすでに前のセクションで詳細に論じた。プログレッション、戦術的機動、社会的インタラクションは新しいものであり、以下で論じる。

プログレッションは、質の高いレベルデザインとこれらのレベルを通じたプレイヤーの進行を制御するメカニクスから生まれるゲームプレイの側面を扱う。これらのゲームでは、デザイナーはプレイヤーが事前に定義されたチャレンジのセットを克服しなければならないレベルを作成している。特定のチャレンジを完了することは、しばしば他のチャレンジのロックを解除し、こうしてプレイヤーは特定のゴールに向かって進行する。ほとんどのこれらのゲームでは、ゴールは特定の場所に到達することである（通常、最後のチャレン

ジが「ボス戦」の形で待ち構えている）。このタイプのゲームでは、レベルの慎重なレイアウトがスムーズな体験を生み出す。それらはレベルプログレッションに依存しないゲームよりも完了するのに長い時間がかかる傾向があるが、一度完了すると、リプレイバリューをほとんど提供しない：多くのプレイヤーはこのタイプのゲームを一度しかプレイしない。プレイ体験とプログレッション駆動型ゲームを通じての進行はデザイナーによって緊密にコントロールできるため、このタイプのゲームはストーリーも伝えるゲームに特に適している。レベル駆動型ゲームの典型的な例には、ゼルダの伝説やアサシンクリードなどのアクションアドベンチャーゲーム、ハーフライフやHaloなどのファーストパーソンシューターゲーム、バルダーズ・ゲートやThe Elder Scrolls IV: Oblivionなどのロールプレイングゲームが含まれる。

戦術的機動は、攻撃的または防御的な優位性のためにマップ上にゲームユニットを配置することを扱うメカニクスを含む。戦術的機動はほとんどのストラテジーゲームにおいて重要であるが、特定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も特徴として見られる。戦術的機動を支配するメカニクスは、特定の場所にいることでユニットが得る戦略的優位性を典型的に指定する。これらのメカニクスは連続的か離散的かもしれないが、離散的でタイルベースのメカニクスがまだ一般的であるように思われる。戦術的機動はチェスや囲碁のような多くのボードゲームで重要であるが、スタークラフトやCommand & Conquer: Red Alertなどのコンピュータストラテジーゲームでも重要である。

ゲームをプレイすることから生まれる多くの社会的インタラクションは、メカニクスでは捉えられない。マルチプレイヤーゲームが直接の、ゲーム内のインタラクションを許すとすぐに、ルールの外側の社会的インタラクションが生まれる。一部のゲームはそのようなインタラクションをより明示的に扱うメカニクスを含んでいる。例えば、ロールプレイングゲームにはキャラクターの演技をガイドするルールがあるかもしれないし、ストラテジーゲームにはプレイヤー間の同盟の形成と解消を支配するルールが含まれるかもしれない。

この論文は主に、経済とプログレッションの離散的なメカニクスにズームインする。そのための3つの理由がある：

1. 図1.7から明らかになるように、これらのタイプのメカニクスはほとんどのゲームジャンルにおいて役割を果たしている。それらは戦術的機動や社会的インタラクションよりも一般的である。

2. 上で論じたように、ある程度離散的なメカニクスにはデザインの自由度が通常より大きい。連続的な物理学は一般に現実のまたは想像上の設定を正確にシミュレート

することを目指しており、必要な知識は実際の物理学から直接取ることができる。離散的なメカニクスとゲーム経済については、既製の解決策ははるかに少ない。この論文はこれを改善するための応用理論の発展に貢献することを目指している。

3. この論文の範囲を制御するために、すべてのタイプのメカニクスを等しく詳細に論じることはできない。ゲームにおける戦術的機動と社会的インタラクションは、両方とも独立した詳細な研究を正当化する非常に大きなトピックである。

1.4 創発とプログレッション

プログレッションのメカニクスは、Jesper　Juulがゲームにおける「創発の構造」から分離する「プログレッションの構造」と呼ぶものに対応する（Juul,　2002）。彼の分類はゲーム研究の中で非常に影響力があり、この論文におけるメカニクスの研究のための関連するフレームワークを提供する。簡単に言えば、創発は比較的単純なルールが多くのバリエーションをもたらすことを示し、プログレッションは多くの事前に設計されたチャレンジが順序付けられていることを示す。Juulによれば、「創発はゲームの原初的な構造である」（Juul, 2002, 324）。これはボードゲーム、カードゲーム、ストラテジーゲーム、そしてほとんどのアクションゲームにおけるルールの多くの可能な組み合わせによって引き起こされる。このタイプのゲームは多くの異なる構成や状態をとりうる：チェスにおける駒のすべての可能な配置は異なるゲーム状態を構成し、単一のポーンの1マスの移動でさえ重要な違いとなる。チェス盤上の駒の可能な組み合わせの数は膨大であるが、ルールは1ページに容易に収まる。シミュレーションゲームSimCityにおける住宅ゾーンの配置や、ストラテジーゲームStarCraftにおけるユニットの配置についても同様のことが言える。

プログレッションは一方で、厳密にコントロールされたイベントの順序に依存する。基本的に、ゲームデザイナーはプレイヤーが遭遇するチャレンジを、プレイヤーが特定の順序でこれらのイベントに遭遇しなければならないようにレベルをデザインすることによって決定する。この形式のゲームをメディエートすることを可能にしたのはコンピュータの使用である。プログレッションは、ゲームが事前に準備された多くのコンテンツとともに公開されることを要求する。ボードゲームにとってこれは不便である。^5　そのため、プログレッションは1970年代のテキストアドベンチャーゲームに始まるより新しい構造である。最も極端な形では、プレイヤーはゲームを通して「レールに乗せられ」、一つのチャレンジから次のチャレンジへと進むか、試みに失敗する。プログレッションでは状態の数は比較的少なく、デザイナーはゲームに何を入れるかについて完全なコントロールを持つ。これはプログレッションのゲームをストーリーを語るゲームに適したものにする。

> ^5 ペンと紙のロールプレイングゲームの出版されたシナリオは、非デジタルのプログレッションゲームの例である。それらは、設定、キャラクター、可能なストーリーラインを指定する本の形式をとる。しかし、それらはコンピュータゲームよりも古い形のゲームにおけるプログレッションであるとは考えられない。なぜなら、ペンと紙のロールプレイングはコンピュータのプログレッションゲームと同じ時期に起源を持つからである。

もとの論文で、Jesper Juulは創発を含むゲームへの選好を表明している：「理論的なレベルでは、創発はより興味深い構造である」（Juul，2002，328）。彼は創発を、プレイヤーの自由とデザイナーのコントロールがバランスされる構造として捉えている：創発のゲームでは、デザイナーはゲームが公開される前にすべてのイベントを詳細に指定しないが、ルールは特定のイベントを非常にありそうにするかもしれない。実際、創発的な構造を持つゲームはしばしばかなり規則的なパターンに従う。Juulは、Counter-Strikeのゲームでほぼ常に勃発する銃撃戦を論じている（Juul，2002，327）。別の例はRiskに見られる。そこではプレイヤーの領土は最初はマップ全体に散らばっているが、プレイの経過とともにその所有権は変化し、プレイヤーは一般的に隣接する領域の一つまたはいくつかの地域をコントロールするようになる。これらの出現するパターンにもかかわらず、Juulはほとんどのゲームが創発とプログレッションを組み合わせていることを認めている。Juulの論文における主な例であるEverQuestは「創発のゲームであり、プログレッション構造が埋め込まれている」（Juul, 2002, 327）。

著書Half-Realにおいて、Juulは創発とプログレッションの議論においてよりニュアンスを持っている（Juul，2005）。ほとんどの現代のゲームは創発のゲームとプログレッションのゲームの間のどこかに位置する。Grand Theft Auto: San Andreasは広大なオープンワールドを持つが、同時に新しい要素を導入しこのワールドを部分ごとにアンロックするミッション構造も持つ。ストーリー駆動型ファーストパーソンシューターゲームDeus Exでは、ストーリーラインがプレイヤーが次にどこに行くべきかを指定するが、プレイヤーは途中で遭遇する問題に対処するために多くの異なる戦略と戦術を持っている。Juulの分類によればDeus Exをプログレッションのゲームと定義して「ウォークスルー」を書くことは可能であるが、Deus Exには多くの可能なウォークスルーがある。SimCityの特定のマップについて、効果的な都市を建設するために特定の時間に特定のゾーンやインフラを建設するよう指示するウォークスルーを作成することも、少なくとも理論的には可能である。そのようなウォークスルーに従うことは難しいだろうが、作成することは可能である。純粋な創発のゲームと純粋なプログレッションのゲームは双極的なスケールの二つの極端を表すが、ほとんどのゲームは両方の要素を持っている。しかし同時に、創発とプログレッションは、ゲームにおいてチャレンジを作成する二つの代替的なモードとして提示

されており、ゲーム内で共存するかもしれないが、統合するのは困難である。この論文はこの視点に疑問を呈し、構造化されたレベルデザインと創発的でルールベースのプレイをより効果的に統合する戦略を探求する。

その答えに向けた一つの軌跡は、創発的振る舞いはカオスと秩序の境界のどこかで繁栄するというものである（cf. Salen & Zimmerman, 2004, 155）。真に混沌としたシステムはほとんどの観察者にとってランダムで無意味に見えるが、ゲームにおいては、プレイヤーが何が起こっているかを理解できるなら助けになる。ルールが動的な振る舞いを導入することによってゲームをカオスの方向に押しやる一方で、レベルは構造を課すことによってゲームを秩序の方向に引き戻す。ゲームがあまりにも引き戻されると、空間的構造がルールを支配しわずかな動的なプレイしか残らないプログレッションのゲームになる。

この論文は、ほとんどのゲームが複雑で創発的な振る舞いを示すことを認めている。多くのゲームはレベルデザインを通じてこの振る舞いを構造化するが、一部のゲームはそれなしでもやっていける。Bejeweledのような多くのカジュアルゲームは純粋に創発のゲームである。Angry Birdsのような他の多くのカジュアルゲームには、プレイヤーに新しいチャレンジを突きつける事前にデザインされたレベルが多くあるが、それらは構造化されたストーリーのようなプレイ体験というよりはパズルである。これらのゲームでは、メカニクスが整えば、多くの新しいパズルとレベルを無限に生成できる。Tangramのゲームと似ている。

一方で、純粋なプログレッションのゲームはかなりまれである。最も典型的なこれらのゲームの例は、Colossal Cave AdventureやZorkのようなテキストアドベンチャーである。しかしそのゲームジャンルは20年以上前にほぼ絶滅した。今日のアドベンチャーゲームはほぼ常にアクションアドベンチャーゲームであり、ほぼ常にゲームプレイの一部としてメカニクス駆動の創発的アクションの何らかの形を含んでいる。つまり、ゲームが創発とプログレッションの両方を持つことができるとしても、現代のゲームは前者なしにはやっていけないが、後者なしにはやっていけるように思われる。

いわゆる「サンドボックスゲーム」は、プレイヤーを特定のゴールに向けて導くようには設計されていない、オープンな仮想世界を作り出す。サンドボックスゲームは、図1.7のマネジメントシミュレーションジャンルとおおよそ対応する。このタイプのゲームでは、Grand Theft Auto IIIのようなファーストパーソンの視点からであれ、SimCityやCivilization（図1.8参照）のようなゲームにおける神のような視点からであれ、プレイヤーは自分の好きなように探索することができる。典型的なサンドボックスゲームでは制限は少なく、プレイヤーが追求するための多くのオプションのゴールがあり、これらのゴールの一部はゲームではなくプレイヤーによって設定される。SimCityのデザイナーで

あるWill　Wrightは、彼のゲームはいかなるゴールも指定しないためおもちゃにより近いと述べたと引用されている（Costikyan，1994）。これらのゲームは可変的で定量化可能な結果を定義しない。代わりに、プレイヤーは自分自身のゴールを設定し、価値を見出す。

> **図1.8：**　Civilization　IIIはプレイヤーが探索し、征服し、形作るための広大なオープンワールドを持つゲームの好例である。

ストーリーを語るためのメディアとして、あるいは簡潔なプレイ体験を提供するために、広大なオープンワールドは必ずしも最良の選択肢ではない。世界は非常に大きくなり、プレイヤーはメインストーリーラインを簡単に見失うことがある。ストーリーを追いかけたり、また別のダンジョンを進むことはかなり退屈に感じられうる。これはThe Elder Scrolls IV: OblivionやFallout 3のような大規模ゲームが悩まされる欠点である。それはまた、Chris Crawfordがインタラクティブストーリーテリングのためのこの構造にあまり信頼を置かない理由でもある（2003b，261-262）。しかし、ゲームに見られる人工的な世界は新しいリリースのたびに大きく詳細になっているようであり、これはプログレッションがゲームデザインの関連する側面であり続けていることを示している。

1.5 創発

ゲームにおける創発という用語の使用は、Juulのカテゴリーに先行するものであり、複雑性の科学における用語の使用に関連してよく使われる（例えばSmith，2001を参照）。そこでは、システムの振る舞いのうち、その構成部分から（直接的に）導き出すことができないものを指す。同時にJuulは、創発的振る舞いとデザイナーが単に予見しなかった振る舞いを示すゲームを混同しないよう注意を促している（Juul，2002）。ゲームにおいて、いかなる複雑なシステムにおいてもそうであるように、全体は部分の総和以上のものである。複雑なシステムにおけるアクティブなエージェントやアクティブな要素はそれ自体はかなり洗練されたものでありうるが、通常はかなり単純なモデルとしてシミュレートすることができる。研究が異なる環境における歩行者の流れについてのものであっても、わずかな行動ルールとゴールだけでシミュレートすることで優れた結果が達成されてきた（Ball，2004，131-147）。同様に、ゲームを構成する要素は複雑性の科学によって研究される典型的なシステムの要素よりもはるかに複雑でありうるが、少なくとも囲碁やチェスなどのいくつかのゲームは、比較的単純な要素とルールで膨大な深さのプレイを生み出すことで有名である。これらのゲームのアクティブな実体は、個々の部分の複雑さではなく、部分間の多くのインタラクションの結果である複雑さである。

この論文の主要な仮定は、他のシステムにおいて創発に寄与する要素の複雑なシステムへの特定の構成が、興味深いゲームプレイをも引き起こすということである。言い換えれば：ゲームプレイはそのルールによって定義されるゲームシステムの創発的特性である。ゲームデザイナーにとって、これは一般的な創発システムの構造的特性を理解すること、そして特に自分のゲームにおけるそれらを理解することが、不可欠な知識であることを意味する。

創発的振る舞いを示す最も単純なシステムの一つは、Stephen　Wolfram（2002）によって研究されたセルオートマトンの特定のクラスである。セルオートマトンのセルは、ローカルルールにのみ従う比較的単純な機械である。その振る舞いを定義するアルゴリズムは、直近の周囲からのみ入力を取る。セルオートマトンのこの特定のクラスでは、セルはライン上に配置されており、各セルの状態または色は、そのセルとその二つの直近の隣接セルの以前の状態によって決定される。2つの可能な色のみで、これは8つの可能な組み合わせを作り出す。図1.9は、可能なルールのセットの一つ（下部）と、結果として生じる驚くほど複雑なパターン（上部）を表示している。このパターンは、システムの各イテレーションが前のイテレーションの下に新しい水平線として表示されるために作り出される。Wolframの広範な研究は、動的振る舞いを示すシステムの3つの重要な特質を明らかにした：1）それらは、ルールがローカルに定義された単純なセルで構成されていなければならない、2）システムは長距離通信を可能にしなければならない、そして3）セルの活動レベルは、システムの振る舞いの複雑さの良い指標である。これらの特質は以下で論じる。

> **図1.9：**　Stephen　Wolframの「ルール30オートマトン」、http://www.stephenwolfram.com より

コンピュータ上でセルオートマトンを実装する最も簡単な方法は、各イテレーション後の新しい状態を決定する関数の入力として、自身の状態と直近の隣接セルの状態を取る単純な状態機械をプログラムすることである。各セルの状態のこの通信は、出現する振る舞いにおいて重要な役割を果たしている。そのような入力がなければ、すべてのセルは個別に振る舞い、システム全体の振る舞いはまったく不可能であろう。より動的な振る舞いを得るためには、隣接するセル間の通信が長距離通信につながらなければならない。このタイプの長距離通信は間接的であり、システムを通じて広がるのに時間がかかる。通信のポケット間にほとんどまたはまったく通信がないシステムは、そのようなポケットが発生しないか、より頻繁に発生しないシステムよりも複雑さの低い振る舞いを示す（Wolfram, 2002，252）。接続性はシステムにおける長距離通信の良い指標である。長距離通信の特殊なケースはフィードバックである：セルまたはセルのグループがシグナルを生成し、そ

れが最終的に将来のどこかで自身の状態にフィードバックされる。長距離通信はシステムを通じて長い距離を移動するか、あるいは時間を通じて遅延効果を生み出す。この論文を通じて見るように、この種のフィードバックはゲームにとって非常に重要である。

アクティブなセル（その状態を変化させるセル）の数は、システム全体の振る舞いにとって重要である。予測しにくいが、何らかの種類のロジックや隠れたパターンに従っているように見える複雑な振る舞いは、多くのアクティブなセルを持つシステムにおいて主に発生する（Wolfram, 2002, 76）。

セルオートマトンは、複雑さのための閾値は驚くほど低いことを教えてくれる。比較的単純なルールが複雑な振る舞いを生み出すことができる。この閾値が超えられると、追加のルールを導入しても振る舞いの複雑さにはそれほど影響しない（Wolfram, 2002, 106）。

別の創発の研究において、Jochen Frommは4つのタイプの創発（タイプI、II、IIIおよびIV）からなる創発の分類法を構築している。これらのタイプは、システム内のフィードバックすなわち通信の性質によって区別できる（Fromm, 2005）。フィードバックは、システム内にコミュニケーションの閉回路が存在するときに作り出される。事実上、特定の要素の状態変化が直接的または間接的に同じ要素の状態に後で影響を与えるとき。フィードバックはそれらの効果が自らを強化するときに正と呼ばれる。ギターのフィードバックがそのケースであり、弦が振動して音を出し、音の増幅が弦をさらに振動させる。フィードバックはその効果が自らを減衰させるとき負と呼ばれる。サーモスタットは典型的な例である。温度計が空気の温度を検出し、低すぎるとヒーターを作動させ、ヒーターは温度を上昇させ、それがやがてサーモスタットにヒーターを切るようにさせる。負のフィードバックは、このようにシステムのバランスを維持するためにしばしば使用される。

最も単純な形の創発、名目的または意図的な創発（タイプI）では、組織の同じレベルにおけるエージェント間にフィードバックがないか、フィードバックのみがある。そのようなシステムの例には、ほとんどの人工機械が含まれ、機械の機能はそのコンポーネントの意図的な（そして設計された）創発的特性である。意図的な創発を示す機械の振る舞いは決定論的で予測可能であるが、柔軟性や適応性に欠ける。ギターのフィードバックとサーモスタットの両方がこのタイプの予測可能なフィードバックの例である。

Frommの二番目のタイプの創発、弱い創発（タイプII）は、システム内の異なるレベル間のトップダウンフィードバックを導入する。フロッキングは、彼がこのタイプの振る舞いを説明するために使用する例である。群れのメンバーは周囲の群れのメンバーに反応し（エージェント対エージェントフィードバック）、同時に群れを全体としてのグループと

して認識する（グループ対エージェントフィードバック）。群れ全体は個々の群れメンバーとは異なるスケールを構成する。群れのメンバーは両方を認識し反応する。

弱い創発システムから複雑さの梯子を一段上がると、多重創発を示すシステム（タイプIII）が見つかる。これらのシステムでは、複数のフィードバックが組織の異なるレベルを横断する。Frommは、興味深い創発が短距離の正のフィードバックと長距離の負のフィードバックを持つシステムに見られることを説明することによって、このカテゴリーを例証している。それは動物のコートの縞模様やスポット、そして株式市場の変動を推進する。John ConwayのGame of Lifeもまた、このタイプの創発の一例である（Gardner, 1970）。^6 Game of Lifeは正のフィードバック（セルの誕生を支配するルール）と負のフィードバック（セルの死を支配するルール）の両方を含むことが容易に示せる。Game of Lifeはまた、異なるスケールの組織を示す：最低端にはセルのスケールがあり、より高いレベルの組織ではグライダーやグライダー銃のような持続的なパターンや振る舞いを認識できる。

> ^6 Game of Lifeと呼ばれるが、Conwayのセルオートマトンはこの章の前半で定義されたゲームのカテゴリーには入らない。定量化可能なゴールを持たず、プレイヤーのいかなる努力も必要としない。しかし、おもちゃの場合と同様に、Game of Lifeのプレイヤーは自ら目標を設定することができる——非常に長く生き延びる構成を見つけたり、安定したシステムに成長したりなど。これらのゴールは定量化可能であり、達成するための努力を必要とする。
>
> ^7　両方のケースで一方が他方から生じるのか、あるいは一斉に進化したのかは疑問の余地がある。創発はそれらの謎を説明する最良の方法ではないかもしれない。実際、一部の研究者は強い創発が存在しうるかどうかについて深刻な疑念を表明している（Chalmers, 2006）。

Frommの最後のカテゴリーは強い創発（タイプIV）である。彼の二つの主な例は、遺伝システムの創発的特性としての生命と、言語と文字の創発的特性としての文化である。強い創発は、創発が作用するスケールと、システム内の中間スケールの存在との間の大きな違いに帰せられる。強い創発は、最高レベルの創発的振る舞いの結果を最低レベルのエージェントから分離できるマルチレベルの創発である。例えば、Game of Lifeに使用されるセルオートマトンのグリッドを、より高いレベルではそれ自体が創発的振る舞いを示すチューリングマシンとして機能するように設定することが可能である。この場合、チューリングマシンが示す振る舞いとGame of Life自体との間の因果関係の依存性は最小限である。^8

> ^8 これは、Game of Lifeの特定の、一見ランダムな開始条件からチューリングマシンが出現しうると主張することとは同じではないが。

この創発の性質に関するこの簡単な議論から、いくつかの重要な観察が浮かび上がる。この論文の中で、創発的振る舞いはシステムにおけるフィードバックループ、そして好ましくは複数のフィードバックループに帰せられる。一つだけでは名目的（タイプI）フィードバックにしかならない。したがって、創発的システムは多かれ少なかれ独立して行動する複数の要素で構成されていなければならない。十分なレベルの活動が必要である。わずかなアクティブ要素しかないシステムは、あまりにも安定的で予測可能であり、興味深いゲームを作る傾向がある。これらの要素間にはローカルスケールで通信（またはインタラクション）が存在しなければならず、このローカルな通信が間接的に長距離通信を可能にしなければならない。情報やアクションが源にフィードバックされる通信の形式であるフィードバックは、特に複数のフィードバックループがシステムに影響を与える場合に、しばしば創発的振る舞いを引き起こす。最後に、創発的システムは異なるスケールの組織をしばしば示し、通信とフィードバックがこれらのスケールを横断する。

1.6 プログレッション

ゲームにおける創発の重要性にもかかわらず、プロのゲームデザイナーはレベルデザインとプログレッションのメカニクスに目を向けないわけにはいかない。ゲームルールに自らを委ねることは、魔法の輪の境界を越え、ゲームのフィクション空間に没入することである。その空間の中でプレイヤーはゲームとその可能な状態を探索し始める。現代のリテールビデオゲームのルール、インターフェース要素、ゲームプレイオプションの数は、通常ほとんどのプレイヤーが一度に把握するには大きすぎる。インターネットで見つかるより小さなゲームでさえ、多くのルールを学び、多くの異なるオブジェクトを認識し、異なる戦略を試すことをプレイヤーに頻繁に要求する。プレイヤーをこれらすべてに同時にさらすことは、圧倒的な体験をもたらす可能性があり、プレイヤーはすぐに他のゲームを選んで去っていくだろう。これらの問題に対処する最良の方法は、巧みなレベルデザインでゲーム体験を構造化し、把握しやすいチャンクでプレイヤーにルールを教えることである。多くの場合、ゲームにはプレイヤーにコアコンセプトを紹介する特別なチュートリアルレベルが含まれ、それでも新しいコンセプトは極めて慎重に導入される。

チュートリアルとレベルデザインを使用してプレイヤーを訓練することは、ゲームというメディアの強みの一つを示している：プレイヤー体験を構造化するためのゲーム空間の使用。時間における出来事を描写するのに適した文学や映画とは異なり、ゲームは空間を描

写するのに適している。Henry　Jenkinsはゲームを空間的な物語の伝統に位置づけ、伝統的な神話やヒーローの探求、そしてJ.R.R.　Tolkienの現代作品と共通する栄誉を与えている（Jenkins, 2004）。単にゲーム空間を旅することで、物語が語られる。Ted FriedmanのCivilizationに関するエッセイにも同様の感情が見られ（1999）、そこではそのゲームのドラマはプレイヤーの旅と仮想世界の征服から直接生まれている。

多くのゲームがこの能力を大きな効果で活用している。Half-Lifeシリーズは特に良い例として際立っている。このシリーズのゲームは、プレイヤーが広大に見えるが実際にはかなり狭い道に限定されている仮想世界を移動するファーストパーソンシューターアクションゲームである。Half-Lifeの全ストーリーはゲーム内で語られ、プレイヤーをゲームから引き出すカットシーンはなく、すべてのダイアログはゲーム内のキャラクターによって行われ、プレイヤーは聞くか完全に無視するかを選べる。Half-Lifeはプレイヤーをゲームを通じてガイドし、プレイヤーのためによく構造化された体験を作り出す技術を完成させた。この実践はしばしば「レールに乗せる」と呼ばれる。この観点からすれば、Half-LifeとHalf-Life　2でプレイヤーが列車の中に到着することはおそらく偶然ではない（図1.10参照）。

> **図1.10：**　Half-Life　2ではプレイヤーは列車でゲームに到着するが、決してレールを離れない。

1.7 アプローチと論文の概要

離散的なメカニクスを用いて創発的なゲームプレイと一貫したプログレッションをデザインすることは、デザイナーに多くの問題を突きつける。おそらくゲーム産業にとって最大のハンディキャップは、創発的なゲームプレイの複雑さに対処し、ゲームのデザインを支援するための形式的で理論的なツールの欠如である。数人の著名なデザイナーと学者がDoug　　Churchの「形式的な抽象デザインツール」を開発する呼びかけに応えてきた（Church,　1999）。この論文もまたその呼びかけに応える。離散的なゲームメカニクスとレベルプログレッションのための適用可能なデザイン理論を開発することにより、私はデザイナーがゲームシステムの複雑な性質とゲームプレイの捉えどころのない概念についてより良い理解を得、プログレッションを効率的に作成する手助けをしたいと考えている。この点において、この論文はボードゲームとコンピュータゲームを区別しない。両方とも本質的にルールベースのアーティファクトであり、創発的なゲームプレイを持つことができる。この論文で議論されるゲームは、両方のカテゴリーからほぼ同等に来ている。

デザインプロセスを支援または自動化するためのプロトタイプソフトウェアツールの開発は、この論文の重要な側面である。多くの点で、これらのプロトタイプの開発は理論の適用可能性の検証における重要なステップであった。同時に、実装は理論のさらなる改善に必ず導き、非常に反復的なプロセスであることが判明した。

次の章では、ルールベースシステムとしてのゲームをより深く探求する。ゲームはシミュレーションといくつかの関係を共有し、それらもルールベースシステムである。しかし、シミュレーションがソースシステムを正確にモデル化することを目指すのに対し、ゲームは異なるゴールを持つ：興味深い体験を作り出すことを目指す。これは、ゲームがルールを異なる方法で扱うことができることを意味する。

第3章では、ゲーム産業と学術界からのゲームデザイン理論を論じる。これらの理論はすべてそれぞれのメリットを持つが、産業または学術の標準として出現した理論はない。実際、ゲームのデザインに対していかなる理論的またはメソドロジカルなアプローチも機能しうるかを疑う人もいる。なぜなら、ゲームデザインに含まれる創造性にいかなるアプローチも正義を果たせないからである。第3章はまた、この意見に賛同する人々が提出する議論にも対処する。

第4章では、内部経済と創発的なゲームプレイに焦点を当てた代替的なゲームデザイン理論として、私のMachinationsフレームワークを提示する。このフレームワークの設計は、第3章で議論された懸念を考慮に入れている。離散的なゲームメカニクスを表現するために抽象的で視覚的な表記法を利用している。これらのMachinationsダイアグラムのデジタル版は、ゲームをシミュレートするために実行することができる。Machinationsフレームワークは、（良い）ゲームプレイに寄与するゲームメカニクスの構造的特質を前景化することを目指している。

Machinationsフレームワークはゲーム経済に焦点を当て、レベルデザインを無視している。第5章では、ゲームレベルとプログレッションのメカニクスが中心となる。この章では、ゲームデザインのこれらの要素に対する構造的な視点を提供するMission/Spaceフレームワークを開発する。MachinationsフレームワークとMission/Spaceフレームワークがどのように理論的なレベルでゲームデザインに情報を与えるために使用できるかの例証として、第6章では両方のフレームワークを活用して、創発とプログレッションがどのように統合されうるか、そして時に統合されてきたかを探求する。

第7章では、第4章と第5章で提示されたゲームに関する形式的な視点を、モデル変換の一連の概念の下にゲームデザインの概念として統合する。ソフトウェアエンジニアリングから取られた概念であるモデル変換は、主題に対するそれぞれ異なる視点を表すモデルが、

形式文法と書き換えシステムの使用を通じて他のモデルに変換されうる方法を記述する。ゲームデザインのためのモデル変換は、デザインプロセスの特定の側面を自動化するための理論的なてこを提供し、ゲームコンテンツを生成するいくつかの実験的なソフトウェアプロトタイプの議論によって例証される。

第8章では、この研究の間に開発された応用理論とツールを評価し、主要な研究質問に答える：創発的なゲームプレイのデザインを支援する応用理論とゲームデザインツールの作成に、ゲームルールとゲームレベルのどのような構造的特質が使用できるか？ アムステルダム応用科学大学で教えられたゲーム開発コースにそれらを統合した結果と、いくつかの産業的および学術的な会議で私が主催したワークショップについて論じる。この論文で提示されたツールと理論の学生、産業のベテラン、学術的な同僚による受容は、その検証の重要な側面である。

1.8 用語

本論文を通じて以下の用語が使用される：

- ゲームとは、プレイヤーがルールによって定義された人工的な葛藤に従事するシステムであり、プレイヤーの努力と能力によって影響される可変的で定量化可能な結果をもたらす。
- ゲームプレイとは、そのルールによって定義されるゲームの創発的特性である。ゲームプレイはプレイヤーのアクションと体験の定性的な尺度である。良いゲームはゲームプレイを持つと言われる。
- ゲームのメカニクスは、単一のゲーム要素の振る舞いを支配するルールのセットである。これらのルールは具体的である：例えば、メカニズムはキャラクターがどのくらい速く動くか、キャラクターがどのくらい高くジャンプするか、これにどれだけのエネルギーを消費するかを正確に指定する。この論文では、やや不自然ではあるが一般的に使用される「ゲームメカニック」の単数形ではなく、「ゲームメカニズム」を使用することを好む。この論文は連続的な物理学ではなく、離散的なゲームメカニクスに焦点を当てている。なぜなら、これらのメカニクスは一般により多くのデザインの自由を提供し、離散的なメカニクスをモデル化するために利用可能な参照システムがあまり確立されていないため、あまりよく理解されていないからである。

- ゲームのコアメカニクスは、プレイヤーが最も頻繁にインタラクトし、ゲームプレイに最も強く影響するメカニクスである。このセットの境界はあいまいである。

- ゲームの内部経済は、ゲームリソースの生産、流通、消費によって構成される。これらのリソースには、パワーアップ、コレクティブル、ポイント、アイテム、弾薬、通貨、体力、プレイヤーのライフが含まれるが、これらに限定されない。これらのリソースは有形または抽象的でありうる。生産、流通、消費を決定するメカニクスの構造は、ゲームの創発的なゲームプレイにおいて重要な役割を果たす。

- レベルは、プレイヤーが遭遇するチャレンジを規定する、ゲームにおける特定の空間的および/またはロジスティック的な構造である。典型的には、レベルは配置されたゲーム要素のセットと、特別なイベントやゲームを通じたプレイヤーの進行を制御するスクリプトを含む。

- ゲームにおける創発は、特定のゲームの振る舞いが複雑で動的なルールのシステムの結果であるという事実を指す。これは、これらのゲームでは可能な状態の数が膨大であることを意味する：比較的少なく、しばしば離散的なメカニクスが大きな数、時には無限の可能な状態を作り出すことができる。創発はゲームプレイとリプレイバリューの重要な源であるが、予測、デザイン、コントロールも非常に困難である。

- ゲームにおけるプログレッションは、デザイナーが可能なゲーム状態を事前に概説した、通常はレベルデザインまたは何らかの形のゲームスクリプティングを通じてのゲームの構造を指す。プログレッションはプレイ体験の多くのコントロールを提供するが、比較的低いリプレイバリューを生成するという欠点がある。

- フィードバックは、システム内の特定の要素の現在の状態の変化が、後の時点で同じ要素の状態に影響を与えるときに発生する。フィードバックは、因果関係の閉回路が状態変化の効果を新しい状態にフィードバックさせることを要求する。フィードバックは、創発と創発的なゲームプレイに寄与するゲームメカニクスの構造的特質において重要な役割を果たす。

第2章 ルール、表現、リアリズム

> 完璧に到達したと思われるのは、付け加えるものが何もなくなったときではなく、取り去るものが何もなくなったときである。
>
> — Antoine de Saint-Exupéry (1939)

^1 この章は、学術誌 Simulation & Gaming のジャーナル論文としてわずかに改変された形でも掲載された（Dormans, 2011a）。

エンターテインメントゲーム産業の中では、ゲームをよりリアルにすることに多大な労力が費やされている。グラフィックス、物理モデリング、人工知能がかつてないほどリアルなシミュレーションに向けて大きな飛躍を遂げるにつれ、ゲーム制作は年々大きくなっている。これらの発展はゲームというメディアに対する我々の理解を前進させるが、代替的なアプローチに積極的な関心を示す開発者は少ない。シリアスゲームのような特定のタイプのゲームはこの傾向に悩まされている。他の教育メディアと比較して、シリアスゲームはすでにかなり高価に制作されている。これらのゲームのプレイヤーや制作者がトリプルAタイトルに見られるレベルに近いリアルな洗練度を期待するなら、これらのゲームは追いつくことに失敗するだろう。

この章は、リアリズムから離れ、ゲームを抽象的で非リアルなルールベースの表現の形式として探究する代替的な視点を提示する。この視点からすれば、ゲームの強みはファンタジーワールドの正確なモデリングにあるのではなく、比較的単純なルールで複雑なシステムを捉えることにあり、なおかつゲームがモデル化しようとしている元のシステムの全体的な動的振る舞いを保持する。この章では、ルールベースシステムとしてのゲームは、それらのゲームが楽しみのために作られたか教育のために作られたかにかかわらず、知識、学習、エンターテインメントのための優れた手段であると論じる。フォトリアリスティックなアセット、詳細な物理シミュレーション、数百万ドルの予算で作られていなくてもである。

ゲームは、非インタラクティブなテキスト、画像、音声による静的な表現とは根本的に異なる新しい形のルールベース表現を構成する。Rune Klevjerによれば、シミュレーションはプロシージャルな表現の形であり、シミュレーションはイベントの代わりにルールを表現する（2002）。Gonzalo Frascaはシミュレーションをナラティブまたは表現に対する代替として分類している（2003, 223）。Ian Bogostは、Frascaの仕事を引き継いでシ

ミュレーションを以下のように定義している：「シミュレーションとは、より複雑でないシステムを介したソースシステムの表現であり、ユーザーのソースシステムに対する理解に情報を与える」（2006, 98）。

ゲーム、ルール、シミュレーションのこのリンクは、多くの現代のゲームに見られるリアリズムへの現代的な焦点の文脈において特に明確である。年月を経て、ゲームはますますリアルになってきた。現代のコンピュータのパワーは、ほぼフォトリアリスティックな画像をリアルタイムでレンダリングすることを可能にしている。ゲームの視覚的・聴覚的品質は映画に見られる品質に急速に近づいている。ゲームはしばしば実生活から認識できる要素を参照する：本物の車、本物の環境、本物の武器。リアルに見えリアルに感じられるゲームはよく売れる。場合によっては、ゲームが参照するリアリティは純粋にフィクションである。スター・ウォーズのユニバースを舞台にしたゲームは現実ではない多くのことを描写するが、それでもプレイヤーはスター・ウォーズのゲームがどのように見え、音がし、感じるべきかについて明確なアイデアを持っている。業界研究の多くはゲームをよりリアルにすることに向けられている。リアリズムは、Popular　Science誌のウェブサイトに掲載された「今日のゲームデザイナーが直面するトップ10のハードル」に顕著に取り上げられている。すべてが実生活の現象の正確でリアルなシミュレーションに関わっている。水と炎のエフェクトを正しく再現すること、リアルな動き、人間の顔のレンダリング、リアルな振る舞いを捉えるようにデザインされた人工知能（Ward et al., 2007）。

一方で、ゲーム批評家や学者の特定のサークルでは、リアリズムはゲームの本質ではないと指摘することが流行している。Steven PooleのTrigger Happyは、ゲームの想定されるリアリズムを脱構築している。彼は、ほとんどのプレイヤーがゲームをプレイするのは、現実にはできないことをゲームが可能にするからだと論じている。例えば、完全にリアルなレースゲームは、レーシングサーキットの1周を試みる前にプレイヤーに徹底的なトレーニングを受けることを要求するだろう。完全にリアルなゲームはゲームであることをやめる（Poole，2000，77）。彼は結論づける：「ビデオゲームは、『現実の』世界に似た何かを再現する思想を放棄し、代わりに驚くべきだが一貫した方法で機能するまったく新しい新しいものを発明しようとすれば、芸術的にもっと面白くなるだろう」（Poole, 2000, 240）。

ゲームがリアルで正確なシミュレーションとは異なるという感情は、すでにChris Crawfordの初期の仕事に見出すことができる。彼はこう述べている：

> 「正確さはシミュレーションの必要条件であり、明瞭さはゲームの必要条件である。シミュレーションはゲームに対して、技術的なドローイングが絵画に対するのと同じ関係にある。ゲームは単にシミュレーションが持つ詳細さの程度が不足

> しているだけの小さなシミュレーションではない。ゲームは意図的に詳細を抑制し、デザイナーが提示したいと思うより広いメッセージを際立たせる。シミュレーションが詳細であるのに対して、ゲームは様式化されている。」（1984, 9）^2

> ^2　最も様式化された絵画は、非リアルな手段を通じて描写するものの本質を捉えようとする現代絵画である。Crawfordはこのタイプの絵画を指していると思われる。

Jesper Juulもまた指摘している：「ゲームはしばしば様式化されたシミュレーションである。それらのソースドメインへの忠実さのためだけでなく、美的目的のために開発されている。これらは現実世界の要素の適応である。シミュレーションはサッカー、テニス、あるいは現代の都市での犯罪者であることの知覚された興味深い側面に向けて指向している」（2005, 172）。ゲームは私たちに現実生活では利用できないことをさせてくれるものであり、プレイヤーがルールに従う限り、私たちにこのパワーを与えるのはルールである。しかし、ルールは制限とアフォーダンスの両方を生み出す。ルールがなければ、ゲームにはほとんど構造がなく、アクションにはほとんど意味がないだろう（Juul, 2005, 58）。同様の理由で、私はペンと紙のロールプレイングゲームに関する研究でルールをエージェンシーに結びつけた。たとえ多くの熱心なロールプレイヤーがインタラクティブな演劇を支持してルールの重要性を軽視する傾向があっても。ペンと紙のロールプレイングゲームでは、ルールはフィクションの世界とのインターフェースを形成し、プレイヤーがその世界に影響を与えることができるのはルールを通じてである。ゲームルールはエージェンシーを生み出す（Dormans, 2006b）。ゲームを見る際には、ルールがプレイヤーをどのように制限するかではなく、ルールが何を許すかに注目する方がより重要である（cf. Wardrip-Fruin et al., 2009）。スーパーマリオブラザーズのゲームでジャンプボタンを押すことは、画面上のアバターを人間の能力をはるかに超えてジャンプさせる満足感のある効果を持つ。ゲームルールは私たちの能力を増幅し、現実生活では危険、破壊的、非実用的、または不可能な人工的な葛藤における戦略や戦術を探求することを可能にする。

2.1 イコニックの誤謬

Ian Bogostのシミュレーションの定義――「より複雑でないシステムを介したソースシステムの表現であり、ユーザーのソースシステムに対する理解に情報を与える」（2006, 98）――は、Charles S. Peirceによって起草された記号の記号論的三部モデルに密接に類

似している。この類似は、異なる記号論的な視点からゲームにおけるリアリズムを調査する機会を提供する。この視点は、リアリズムへの現在の傾向がゲームにおいて可能な形式のごく一部――イコニックな形式――にのみ没頭していることを明らかにする。同時に、Peirceの記号論的理論はまた、リアリズムを超えてゲームとルールを探求するためにどこを見るべきかも示唆している。

Peirceの三部モデルでは、記号はオブジェクト――記号が表すもの――と解釈項――記号が呼び起こすメンタルコンセプト――に接続されている。解釈項は「解釈者でも記号のユーザーでもない」（Kim，1996，12）。この記号モデルは、記号のアイコン、インデックス、シンボルへの分類で最もよく知られている。この分類は記号とそのオブジェクトとの間の関係の性質に基づいている：記号がそのオブジェクトに類似するとき、それはアイコンであり、記号がそのオブジェクトに対して存在的な接続を持つとき、それはインデックスであり、接続が恣意的であるとき、それはシンボルである（Kim，1996，19-21）。図2.1はBogostのシミュレーションの定義とPeirceの記号モデルを組み合わせている。

> **図2.1：** 記号とシミュレーションの三部モデル。

ゲームとシミュレーションが表現の形式であるならば、その形式（シミュレーション）とそれが表すもの（ソースシステム）との間の関係の同じカテゴリーがゲームに適用される。この視点は、いかなる形の表現と同様にゲームも常にゲームの外側の何かを意味することを規定する。これはエンターテインメントの目的のために作られたゲームに対してさえ当てはまる。エンターテインメントゲームの「詩的機能」（Jakobson，1960）がその表象的形式に注意を向けるとしても、それはゲームの外側の多くの意味ある認識可能な要素を参照するコミュニケーションの形であることに変わりはない。パックマン（Poole，2000，178-183）やテトリス（Murray，1997，143-144）のような比較的シンプルで抽象的なエンターテインメントゲームの文化的解釈がなされてきたが、これらの解釈はときにかなりこじつけであっても、要点は、いかなる芸術形式と同様に、いかなるゲームも社会的・文化的な真空の中には存在しないということである。David Myersが指摘するように：「人間のプレイは認知的かつ象徴的な行為であり、人間の表象過程にとって根本的である」（1999a，486）。

このゲームとシミュレーションの記号論的モデルにおいて、リアリズムとイコニシティは結びついている。シミュレーション（システムとして）がソースシステムに密接に類似するとき、我々はシミュレーションをリアルと呼ぶ。シミュレーションがイコニックであるとき、リアルと呼ぶ。この類推から、シミュレーションの他の二つの形式が自ずと示唆される：インデクシカルなシミュレーションとシンボリックなシミュレーション。ゲームが究極的にリアルでないのであれば、インデクシカルでシンボリックなシミュレーションは

ゲームをより良く理解するのに役立つ興味深い概念かもしれない。次の二つのセクションで見るように、インデクシカルまたはシンボリックと呼べる構造はゲームにおいて大きな効果で使用されてきた。

インデクシカルおよびシンボリックなシミュレーションを探求する前に、言語記号とシミュレーションの類推をもう一歩押し進めて、イコニックなゲームへの現在の焦点と言語の高度にシンボリックな性質との間の興味深い不一致を明らかにしたい。自然言語はその性質上、非常に抽象的であり、リアルではない。ほとんどの単語はそれが指すものに類似していない。そして、言語の偉大な表現力に寄与するのは、言語の抽象的な性質である。この概念は非常に遠くまで遡ることができる。それは17世紀の哲学者John Lockeの著作にすでに明らかであった。彼は観察している：

> 「人間は抽象的な観念を作り、名前をそれらに付けて心に定着させることにより、それによって事物を考え論じることを可能にする。いわば束にして、知識のより容易で迅速な改善とコミュニケーションのためにである。もし言葉と思考が個別的なものにのみ限定されていたならば、知識はゆっくりとしか進まなかっただろう。」（Locke, 1975, 420）

ほぼ一世紀後に、哲学者Edmund Burkeが、彼の時代のリアルな絵画よりも詩に大きな美的力を認めたのも同様の根拠に基づいている。詩人は言葉を使って、伝えようとするイメージを「曖昧にする」。逆説的に、これは同じものの完全で詳細な絵を描くよりも、より生き生きとして喚起的なメンタルイメージをもたらす（Burke, 1990, 55）。近年、抽象芸術の発展はこれをすべて変え、イメージの表現力を劇的に増大させた。これは、美術史が特定のジャンルを特定するために使用する名称——印象派、表現主義、抽象表現主義など——によって例証される。

Ferdinand de Saussureは言語記号の恣意的な性格をその主要な特徴として特定している。非恣意的な記号の可能性を排除してはいないが、人間の言語ではほとんどの記号がその意味に恣意的に結びついていると論じている。我々が使用する言葉に接続された、我々が参照しているものの特性は通常存在しない（de Saussure, 1983, 67-69）。言い換えれば、言語は主にシンボルで構成されている。言語的なアイコンとインデックスはわずかしかない。Saussureにとっても、「異なる観念に対応する異なる記号のシステム」を構築するのは人間の能力であり、それが言語を可能にしている（de Saussure, 1983, 10）。抽象的な意味を取り束にして扱う人間の能力を通じて、人間の表現と理解は特定の事物のレベルを超えて一般的知識の領域に持ち上げられる。言い換えれば、抽象的で非イコニックなプレゼンテーションは、リアルまたはイコニックな表現よりも多くの表現力と表象力を含んでいる。

上で引用されたIan Bogostのシミュレーションの定義は不完全である。Bogostは、主観性がシミュレーションに内在することを強調している：「シミュレーションとは、より複雑でないシステムを介したソースシステムの表現であり、主観的な方法でユーザーのソースシステムに対する理解に情報を与える」（Bogost, 2006, 98）。シミュレーションにおいてシステムは別のシステムを通じて表現され、第二のシステムの構築においてなされた選択はその作成者の価値を反映する：「いかなるシミュレーションもイデオロギー的文脈から逃れることはできない」（Bogost, 2006, 99）。Bogostが主張するように、この主観性は、シミュレーションにおいてシミュレートするシステムがそのソースシステムよりも必然的に複雑でないという事実に部分的に帰せられうる。シミュレートするシステムは常にそのソースシステムから逸脱し、その逸脱においてなされた選択はシミュレーションを作成した人物またはグループの理解および/またはイデオロギーを反映する。Bogostが「より複雑でない」で正確に何を意味するかは明示されていない。ここでは、「より複雑でない」を「より少ない部分で構成されている」と解釈する。シミュレーション内の部分の数は通常、ソースシステム内の対応するものの数よりも低い。これはまた、ほとんどの場合、それらの部分がソースシステム内のより複雑なサブシステムの抽象化であることを意味する。例えば、シミュレートされた気象システムを構成する部分は、実際の気象を構成する多くの実際の空気分子を束にしている。これはシミュレーションをより扱いやすくする。あるいはLockeを言い換えるなら、シミュレートされたシステムの部分の多数をより容易な理解のための束として考え、より容易で迅速なコミュニケーションとその理解の改善のために考えることを可能にする。

したがって、シミュレートされたシステムとそのシミュレーションの間には常にギャップが存在し、そのギャップは常にシミュレーションを多かれ少なかれ主観的にする。しかし、この主観性は、主観的なシミュレーションが許す便利さと強化された理解のために我々が支払う代価である。ほとんどの場合、表現力の獲得は特定のインスタンスにおける類似性の喪失を上回る。

シミュレーションを本質的に主観的なものとして考えるとき、リアリズムへのいかなる主張もそれ自体がイデオロギー的な策略になることに注目する価値がある。例えば、America's Armyにおける高いレベルのリアリティは、視覚的表現における見かけのリアリズムと正確さがイデオロギー的領域に拡張できるというレトリック的な主張として読むことができる：「ゲームが物理学を正しく再現したなら、その倫理的主張もリアルでなければならない」（Bogost, 2007, 78）。一方で、商業的エンターテインメントゲームにおいてリアリズムはしばしば特殊効果としてレンダリングされる。これらのゲームにおいてリアリズムと本物らしさは、観客を感動させ鑑賞されるようにデザインされたスペクタクルとなる。リアリズムはその高いポリゴン数、プラズマ効果、パーティクルエンジンで前

景化され、ハイパーリアルとなる。あるいはGeoff Kingがブロックバスター映画における非常に類似した現象を記述した言葉を使えば、それは「本物らしさのハイパーリアリスティックなスペクタクルであり、本物らしさそのものではない」（King, 2000, 136）。

> **図2.2：** Diabloのインベントリ画面の概略スケッチ。

2.2 インデクシカルなシミュレーション

イコニックなシミュレーションを超えて見始めるにあたり、インデクシカルおよびシンボリックなシミュレーションの概念は明白な出発点である。このセクションでは最初の概念を論じ、次のセクションでは二番目の概念を論じる。Diabloで最初に登場し、以来多くの他のゲームに取り入れられてきた「インベントリシステム」は、最初の概念の例として見ることができる。それはDeus Exの開発者であるWarren Spectorに「Diabloはインベントリを正しく実装した。うまくいっているものをいじる必要はない…」と言わしめた。^3 かなり以前から、多くのコンピュータゲームが「インベントリ」を含んできた：ゲームは主人公がオブジェクトを拾って持ち歩くことを許す。プレイヤーはゲームのインベントリ画面でこれらのオブジェクトを管理できる。ほとんどのゲームはキャラクターが運べるオブジェクトの数を何らかの方法で制限している。キャラクターが拾えるオブジェクトの固定数があるかもしれないし、すべてのゲームオブジェクトに重量値が付いていてキャラクターが特定の荷重まで しかオブジェクトを運べないかもしれない。

> ^3 S. T. LavavejのDeus Exウェブページに引用。URL: http://nuwen.net/dx.html（最終アクセス2011年6月23日）。ただし、このシステムに対する意見は分かれていると言わなければならない。Warren Spectorほど熱狂的でない人もいる（例えばAdams & Rollings, 2007, 516-518を参照）。インベントリの管理でプレイヤーに負担をかけることが良いデザインの判断かどうかは、ゲームの種類と意図するゲームプレイによる。

Diabloのインベントリシステムはオブジェクトのサイズを主な制限要因としている（図2.2参照）。各アイテムはいくつかのインベントリ「スロット」を占め、利用可能なスロットは限られてグリッドに整理されている。アイテムは例えば1x1、2x2、1x4スロットを占めるかもしれない。インベントリの空き状況に応じてオブジェクトを拾えるか拾えないかが決まる。画面の上半分はメインキャラクターが現在装備しているオブジェクトに充てられている。

私は、これがゲームにおけるインデクシカルな表現の一例であると論じる。現実生活で誰かがオブジェクトを運ぶための主な制限要因（形状、サイズ、重量）は、容易に理解できる二次元の形状で表現されている。これらの形状とその相対的なサイズは、シミュレートされた対応物のサイズと重量に存在的に接続されていると言える。したがって、このシミュレーションは、記号とそのオブジェクトとの間の関係が（類似や恣意的な慣習ではなく）存在的な接続に基づいているインデクシカルな記号に平行しているため、インデクシカルな構造として適格である。

この構造を何らかの形でコピーしたゲームの数は、この構造の質の高さの証である。内部のルールと制約はすぐに明らかである（画面上の視覚的表現に合わせて調整されているからこそ）。このシステムが生み出す管理上の問題は、現実生活のそれらの問題と非常によく似ている。このシステムはプレイヤーがインベントリを非効率的に散らかすことさえ許し、整理の必要性について何かを教えてくれる。

Diabloのインベントリシステムが非常に効果的に行っていることは、多くの関連する同様に機能するゲームルールを取り、それらすべてをビデオゲームのメディアに適した単一のメカニズムで置き換えることである。明らかに、シミュレーションの正確さの一部は失われている（アイテムは同時に大きくて軽いことはできない）が、全体的な振る舞いは保持される（プレイヤーが運べるものは制限されている）。Diabloのインベントリの巧みさは、インベントリの管理のすべてのニュアンスをサイズの問題に折りたたんだことにある。それはコンピュータ画面で容易に表現でき、以前はより一般的な選択肢であった重量の代わりに、コンピュータの視覚的メディアにはあまりうまく変換されなかったものを使用している。

インデクシカルなシミュレーションのもう一つの例は、ほとんどのゲームが「体力」を扱う方法である。キャラクターやユニットの体力は、パーセンテージや「ヒットポイント」の数といった単純な指標で表現されることが多い。明らかに、現実生活において人の身体的健康や車両の構造的状態は、多くの異なる側面が寄与する複雑な問題である。単一のキャラクターに汎用的な体力を使用することにより、ゲームはこれらのすべての側面を一つの便利なメカニズムに束ねている。プレイヤーもコンピュータも、束を表現するための数値指標を容易に扱い理解することができる。

2.3 シンボリックなシミュレーション

シンボリックなシミュレーションは、ソースシステムのメカニズムに密接に類似するルールを持つシステムのモデリングからさらに一歩離れる。多くのボードゲームにおけるサイ

コロの使用はシンボリックである傾向がある。例えば、Riskのゲームでは、わずかなサイコロの一振りが完全な戦闘を代表することができる。この場合、サイコロを振ることと戦闘の関係は恣意的であり、他のゲームからよく知られた一つのシンプルなアクションが、ほとんどのプレイヤーが専門知識を欠くであろう多数のアクションをシミュレートするために使用されている。サイコロがこれらの戦闘を置き換えることができるのは、ゲームの目的上、プレイヤーが個々の戦闘の結果にほとんど影響力を持つべきではないからである。Riskはグローバルな戦略についてのゲームであり、戦場における戦術的な機動についてではない。プレイヤーは（不正をしない限り）サイコロの結果をコントロールできないが、それはちょうど最高の軍司令官が個人的にすべての戦闘に勝つことができないのと同じである。しかし、プレイヤーはある種の影響力を必要としており、ルールはサイコロの振りと連動している：戦闘により多くの軍隊を投入することで、プレイヤーはより多くのサイコロを振ることができ、成功の確率を改善できる。同様のことがKriegsspiel（図2.3）とその後の多くの戦争ゲームにも当てはまる。Riskとは対照的に、これらのゲームは戦場での戦術的機動がすべてである。そのため、これらの機動のルールはかなり精巧である。しかし、実際の戦闘をカバーするルールはサイコロと消耗表に委ねられている。繰り返しになるが、これらのゲームは戦術スキルを訓練するためのものであり、銃の使い方のためではない。

> **図2.3：** Kriegsspielは実際の地図の周りで機動することによってプレイされ、戦闘はサイコロを使って解決される。
>
> **図2.4：** スーパーマリオブラザーズで敵を避けるか倒すためにジャンプする。

サイコロは、詳細なルールを必要とせずに非決定論的な効果を生み出すための素晴らしいデバイスである。適切に高いレベルの抽象化から、戦闘のような複雑で非決定論的なシステムは、わずかなサイコロの振りと同様の効果を持つ。特にプレイヤーがこのシステムに対してそれほど影響力を持つべきでない場合、サイコロのメカニクスはより複雑なシステムを置き換えるために使用できる。異なるサイコロメカニクスの特徴的なランダム性は、より複雑なシステムが生み出す多くの表面的で非決定論的なパターンに合致させるために使用できる。ペンと紙のロールプレイングゲームは、プレイヤーにより多くまたはより少ない影響力を許す、多くの巧みで興味深いサイコロの使い方を考案してきた。実際、スキルと属性を表すキャラクターの特性セットに関連するサイコロメカニクスは、ほとんどのペンと紙のロールプレイングシステムのコアを形成する。しばしば、同じメカニズムが広範なアクションを表現するために使用される。

その他の例、例えばクラシックなビデオゲーム、スーパーマリオブラザーズにおいて敵を倒すためにその上にジャンプすることは、シンボリックとインデクシカルなシミュレー

ション形式の中間に位置する（図2.4参照）。正確な実装は敵ごとに異なり、確かにすべての敵に対して機能するわけではないが、ゲーム全体とそれが属するシリーズを通じて頻繁な特徴である。この方法が少しおかしいということを指摘した最初の人物が私であるとは思えない。しかし、それはプラットフォームゲーム内での慣習となっており、ゲーマーにすぐに認識でき、そのジャンルの定義的なアクションであるプラットフォームからプラットフォームへのジャンプと結びついている。

何かの上にジャンプすることとそれを倒すことの間の接続は完全に恣意的ではないが、プラットフォームゲームでの使用は非常に慣習的になり、言語におけるシンボリックな記号の定義に平行している。現実世界には、上にジャンプすることで踏み潰せる生き物は存在する。しかし、ぶつかったときは致命的だが踏みつけられたときはそうでない生き物は私が知る限り存在しない。これはこの正確なメカニズムをいくぶん恣意的にする。^4　さらに重要なのは、スーパーマリオブラザーズにおけるこの戦闘方法は、リアリズムへのいかなる主張よりも、そのジャンルの最も顕著なアクションであるジャンプの使用によってより動機づけられている。シミュレーションとシミュレートされるものとの間のリンクは恣意的かつ慣習的である。特にスーパーマリオブラザーズが設定した例に倣った多数のプラットフォームゲームにおいて。

> ^4 そのような生き物が存在するとしても、確かにそれは亀ではない。

しかし、スーパーマリオブラザーズで敵を倒すために必要なスキルと現実生活との間には親和性がある。ゲームではタイミングと正確さが必要であり、それは実際の格闘に関わるスキルの中にあるものである。要点は、ゲーム内の単純な表現は、それらのスキルを磨き訓練すること以上のことを私たちにさせてくれるということである。敵の上にジャンプするというシンプルなメタファーはプレイヤーにとって把握しやすいが、ゲームはそこからプレイヤーを実験へと誘い、戦略を発展させることへと続いていく。ほとんどのプラットフォームゲームでは、各レベルの最後に「ボス」敵が待ち構えており、プレイヤーの戦略の効果をテストするために典型的にデザインされている。それはプレイヤーがシミュレーションを理解し習得し、異なる動きを組み合わせることができることを示す究極のテストである。^5

> ^5　これらの教訓はゲームを超えた状況にまで及ぶ。これらの教訓を学んだプレイヤーのメンタリティは、John BeckとMitchell Wadeによって見事に記述されている：彼らは解決策がいずれ現れることを知っており、人生の多くの問題に対する試行錯誤のアプローチを習得している（2004, 11-14）。

敵の上にジャンプするメカニズムが達成していることは、ジャンプゲームに戦闘ルールを追加する非常に賢い方法であるということである。プレイヤーのための新しいアクションを導入しない。シミュレートしようとするアクションを、ゲームに既に実装されている他の恣意的なルールで置き換えることによってそれを管理している。これはプレイヤーが学ぶ必要のあるアクションの数を減らし、インターフェースをいじり回す代わりに、より深い、より戦術的または戦略的なゲームとのインタラクションに素早く移行することを可能にする。以下で論じるように、シンボリックなシミュレーションはシステムを多かれ少なかれ同等の動的振る舞いを持つよりシンプルな構造に効果的に縮小する。

2.4 少ないほど多い

インデクシカルおよびシンボリックなシミュレーションは、イコニックなシミュレーションよりもシンプルなゲームシステムを作り出す傾向がある。これらのシミュレーション形式がルールに対して許す縮小は、一般に有益である。シンプルなゲームは学びやすいが、それでもマスターするのはかなり難しいことがある。ゲームは「少ないほど多い」という表現が当てはまる唯一のメディアではない。ほとんどすべての形式の表現芸術において、より少ない手段でより多くを語ることは、特に批評家や鑑定家に評価される。Christopher　Alexanderは、建築とデザインのためのパターン言語に詩からインスピレーションを得て、こう述べている：

> 「この言語は、英語と同様に、散文のためのメディアにも、詩のためのメディアにもなりうる。散文と詩の違いは、異なる言語が使われることではなく、同じ言語が異なって使われることにある。普通の英語の文では、各単語は一つの意味を持ち、文もまた一つのシンプルな意味を持つ。詩では、意味ははるかに密度が高い。各単語は複数の意味を持ち、文全体はインターロックする意味の巨大な密度を持ち、それらが全体を照らし出す。」（Alexander et al., 1977, xli）

そして：

> 「この言語を使うことを学んだならば、最小の空間に、組み合わせた多くのパターンを圧縮する可能性に注意を払うことが不可欠である。この圧縮のプロセスを、必要なパターンを備えた最も安価な建物を作る方法として考えるかもしれない。パターン言語を使って詩的な建物を作る唯一の方法もまた、そうすることである。」（Alexander et al., 1977, xliv）

詩的な言語、あるいはあらゆる形式の表現芸術にとって、この質は非常に重要であり、抽象的な記号の使用だけから生まれるのではない。これらの記号の組み合わせと構造、あるいは言語学用語を使えばこれらの記号間のシンタクティカルな関係もまた重要な役割を果たす。この観点から、Noam　Chomskyは言語が話者に有限の手段を無限に使用させることを可能にすると観察した：我々が持つ単語の数は限られているかもしれないが（そして現実における個別のものによってはるかに数が上回られるが）、それらで作ることができる組み合わせの数は無限である（Chomsky, 1972, 17）。言語のこの特性はしばしば離散的無限性と呼ばれる。

ゲームがいくつのルールを持つべきかを正確に量化することは不可能であり、少ないほど多いがどの程度の「多い」かを量化することも不可能である。各個別のデザインにはそれ自身のバランスがある。特定の数のルールが一つのゲームには少なすぎ、別のゲームには多すぎることもありうる。ゲームが追求すべきバランスは、ルールが一方で生み出すゲームプレイの選択肢の数と、他方でそれらのルールを理解または操作するために必要な認知的負担との間にある。Antoine de Saint-Exupéryの有名な引用「完璧に到達したと思われるのは、付け加えるものが何もなくなったときではなく、取り去るものが何もなくなったときである」（1939）はゲームに極めてよく当てはまる。

一般に、ゲームはわずかなルールのみで無限の可能性を生み出すことに非常に長けている。チェスや囲碁のようなゲームには地球上の原子の数よりも多くの可能なゲーム状態があると推定されている（Shannon, 1950参照）。可能な状態の数を決定するのはゲームのルールであるが、より多くのルールがより多くの可能な状態につながるとは必ずしも限らない。さらに、ゲームが多くのルールを使用せずに大量の可能な状態を作り出すことができるとき、ゲームはよりアクセスしやすくなる。

可能なゲーム状態とゲーム状態空間を通じた軌跡は、ゲームルールシステムの創発的特性である。捉えどころのないゲームプレイの概念はこれらの特性に関連している。多くの興味深い軌跡を許すゲームは、より少ないまたはより興味のない軌跡を生成するゲームよりも間違いなくより多くのゲームプレイを持つ。しかし、ゲームプレイのタイプと質を、単にルールを見るだけで決定することは困難であり、不可能でないにしても。三目並べとConnect　Fourのルールの比較は、これらの困難の良い例証となる。三目並べのルールは：

1. ゲームは3×3のグリッドでプレイされる。
2. プレイヤーは交互にマスを占有する。
3. マスは一度しか占有できない。

4. 一列に3つのマスを（直交的にまたは対角的に）最初に占有したプレイヤーが勝つ。

Connect Fourのルールは（違いを強調して）：

1. ゲームは7×6のグリッドでプレイされる。
2. プレイヤーは交互にマスを占有する。
3. マスは一度しか占有できない。
4. 与えられた列の最も下の未占有のマスのみが占有できる。
5. 一列に4つのマスを（直交的にまたは対角的に）最初に占有したプレイヤーが勝つ。

> **図2.5：** Connect Fourでは重力がプレイヤーに各列の最も下の未占有のマスしか占有できないようにする。

これら二つのゲームのルールの違いはわずかであるが、ゲームプレイの違いは巨大である。ルールを理解するのに必要な認知的努力の違いよりもはるかに大きい。市販版のConnect Fourでは、最も複雑なルール（ルール4）は重力によって強制される：直立したプレイ領域の最も低い利用可能なスペースにプレイヤーのトークンが自動的に落ちる（図2.5参照）。これはプレイヤーが手動でこのルールを強制する必要をなくし、代わりにルールの効果に集中することを可能にする。ルールの複雑さの小さな違いにもかかわらず、三目並べは小さな子供にしか適さないが、Connect Fourは大人も楽しめる。後者のゲームは多くの異なる戦略を許し、マスターするのにかなり長い時間がかかる。二人の経験豊富なプレイヤーがゲームをプレイすると、三目並べの場合のような引き分けではなく、エキサイティングなマッチになるだろう。これらの違いをルールの違いだけを見て説明するのは困難である。

最近、比較的シンプルな要素からの複雑な振る舞いの創発は、数学、物理学、社会科学の多くの分野の研究の重要な側面である。ゲームの研究においても、創発はますます重要な概念になっている。計算的側面から、創発は人工知能の開発から水と炎のリアルなレンダリングに至るまで、あらゆるものに使用される重要な技術である。Penelope Sweetserにとって、創発のために設定されたシステムにおける創造的コントロールの不利な喪失は、そのようなシステムが許すより一貫した直感的なプレイヤーインタラクションによって相殺される（Sweetser, 2006, 14）。同様に、ゲームデザイナーのHarvey Smithは、リッチなインタラクションを許す完全にコントロールされたゲーム環境をデザインしようとすることはもはや経済的に実行可能ではないと主張している。純粋な詳細量を手動で効率的に生産することはできないからである（Smith, 2001）。

創発的なゲームプレイを特徴とするゲームの一つの大きな利点は、ルールシステムがしばしばプレイヤーを招いて、ゲームデザイナーが意図した動きを単に繰り返すのではなく、ゲームで実験することを可能にすることである。究極的には、創発的なゲームはゲームルール自体の変容を許す（Myers, 1999b）。これは教育的なゲームを構築する場合に深刻な結果をもたらすが、ゲームデザイナーが特定のストーリーやメッセージを念頭に置いている場合にも同様である。Jan Klabbersにとって、ゲームシステム全体を、デザイン仕様に適合する振る舞いがそのコンポーネントから創発するように形作ることは、ゲームデザイナーの責任である。同時に、システムはプレイヤーが自らの戦略、ゴール、インセンティブに従って行動するための十分な自由を残すべきであり、プレイヤーの立場を反省的なアクターのそれへと高めるべきである。これは「デザインにおける主要なボトルネックの一つである」（Klabbers, 2006, 102）。

しかし、いくつかの点で、コンピュータゲームは創発のトレンドに逆行しているように見える。Jesper Juulは「プログレッションのゲーム」を「創発のゲーム」からコンピュータゲームに関連する歴史的により新しいカテゴリーとして区別している。コンピュータゲーム、特にアドベンチャーゲームの台頭は、プログレッションのゲームを可能にした。コンピュータなしではデータ量と、多数のゲーム空間を通じてプログレッションを促進する特殊ケースのルールの数は扱いにくくなっていただろう（Juul, 2005, 5）。

Chris Crawfordのデータ集約性とプロセス集約性の概念（Crawford, 2003a, 89-92）は、プログレッションのゲームはより新しい形であり、プログレッションがメディアの自然な進化の結果であるという含意に対するJuulの観察と対置できる。Crawfordは、コンピュータは大量のデータを扱うことと膨大な量の数をクランチすることの両方に適しているが、コンピュータをほとんどの他のメディアから際立たせるのは後者の能力であると主張している。データの取り扱いはすべてのメディアが得意とすることである。コンピュータはしばしばデータ内のリモートな場所への高速アクセスを可能にし、それはハイパーテキスト内でうまく活用される能力である（Lister et al., 2003, 23-30）。しかし、コンピュータが真に輝くのは、その場で新しいコンテンツを作成する能力である。他のいかなるメディアも以前には持っていなかった、コンピュータにはプレイヤーとデザイナーの両方を驚かせる能力がある（Smith, 2001も参照）。Chris Crawfordにとって、ゲームはコンピュータのこの能力を活用すべきであり、ゲームはデータ集約的ではなくプロセス集約的であるべきである。言い換えれば、ゲームはプログレッションのゲームではなく創発のゲームであるべきである。

2.5 創発のデザイン

創発のデザインは、悪名高く困難で、やや逆説的なタスクである。システムの創発的特性はシステムが動き始めたときにのみ表面化する。システムがすべてのテスト中にある方法で振る舞ったとしても、常にそうするという保証はない。このことから、リアリスティックの誤謬は、真に創発的なゲームをデザインする困難さを回避するためのかなり保守的な戦略であるように思われる：比較的シンプルなシステムを持ち、興味深い複雑な振る舞いを示すゲームではなく、より多くのより詳細なルールを追加し続けることは、リーンでエレガントなルールシステムを通じて複雑なゲームプレイを創り出すための貧弱な代用品にすぎない。

創発は比較的シンプルなルールの結果でありうるため、ゲームは興味深いゲームプレイを作り出すために複雑なルールシステムに依存する必要はない。それどころか、シンプルな手段を使って複雑なゲームプレイを生成することには多くの利点がある。デザインはデザイナーにとって管理しやすくなり、ゲームはプレイヤーにとって学びやすくなる。上記の非イコニックなシミュレーションの例（Diabloのインベントリ、ヒットポイントの使用、Kriegsspielのサイコロ、スーパーマリオブラザーズのジャンプ）において、インデクシカルおよびシンボリックなシミュレーションの使用は、イコニックなシミュレーションが持つであろうものよりもシンプルなルールシステムをもたらした。これは上で論じた例の特性ではなく、ゲームにおいて非イコニックなルールを使用する利点である。正確な詳細を通じてシミュレートしようとする完全に詳細でリアルなシステムと比較して、インデクシカルおよびシンボリックなシミュレーションはより少ない手段でソースシステムの本質を捉えることを目指している。正しく行われれば、結果は部分を最小化し表現力を最大化するよりリーンでエレガントなシステムとなる。

本質的に、インデクシカルなシミュレーションは多くの関連するおよびおおよそ同型のルールを一つのゲームメカニズムに束ねる。シンボリックなシミュレーションはさらに一歩進み、ソースシステムでは接続されていないであろう箇所でゲーム内のルールを接続する。言語におけるシンボルの使用と同様に、他のものよりうまく機能するシンボルがある。最もうまく機能するシンボルは、それでもなお何らかの親和性を持つ二つの無関係なルールを接続するように見える。スーパーマリオブラザーズの場合、ジャンプと格闘の両方に関わる身体的スキルとタイミングの間には自然な親和性がある（Lakoff, 1987, 448も参照）。

私が参加したシリアスボードゲームGet H2Oの開発は、非イコニック的縮小の適用の非常に良い例である。このゲームは東アフリカの青少年向け教育プログラムの一部として制作

されたもので、プレイヤーはアフリカの大都市の貧しい住宅地域で生き残るために奮闘する（図2.6参照）。不可欠なリソースは希少であり、プレイヤーは個人的な利益とコミュニティの取り組みの間で慎重にバランスを取る必要がある。プレイヤーは起こりうる悪い出来事に対して間接的な影響力しか持たないが、これらの出来事から利益を得ることもあり、紛争の種を蒔く。ゲームはアフリカの大都市での生活をシミュレートしており、プレイヤーに自分の生活のトップダウンの視点を与えるようにデザインされている。それは探索、議論、振り返りのための手段として機能するようにデザインされている。

> **図2.6：** ナイロビでプレイされているGet H2Oのプロトタイプ（写真：Butterfly Works）。

東アフリカの都市生活を詳細にシミュレートしようとする代わりに、ゲームはリソースとルールの数を比較的シンプルなセットに縮小している。ゲームは3つのメインリソースを使用する：お金、家、きれいな水。後者の二つは各ターンにプレイヤーが取れるアクションの数を決定し、お金はより多くの家を建てるために使用される。これらのリソースは常に脅威にさらされている。お金と水は盗まれるかもしれないし、家は焼かれるかもしれない。現実には、アフリカの家族はもっと多くのニーズを持っているが、これら3つのリソースは、都市アフリカ地域の現実の経済と同様に振る舞う希少の経済をシミュレートするのに十分である。このシミュレーションのインデクシカルな性質により、その地域で育った人々にとってまだ認識可能な比較的シンプルなシステムを作ることが可能になった。実際、ゲームは詳細を欠いたことで認識しやすくなった：より少ない詳細は個人的な解釈のためのより多くの余地を作り、ゲームが個人の体験と一致しない可能性を減らす。さらに重要なのは、この経済が短期的な個人的利益と長期的なコミュニティの利益の間の特定のバランスを作り出し、社会的不安定を引き起こすことである。この不安定性を作り出すことがゲームの主要なデザインゴールであった。ゲームはそもそもそのような状況に対処することについて人々を訓練するためのものであった。簡単に言えば、東アフリカの都市スプロールの不安定な社会システムを複製するために、より多くのリソースとより複雑な経済は必要なかった。

ゲームはシンボリックなシミュレーションも使用している。すべてのプレイヤーがターンを取った後、すべてのプレイヤーはそれを見せることなく1枚のトランプを捨てる。これらのカードは通常、プレイヤーのアクションに使用される。捨てられたカードはシャッフルされて公開される。すべてのカードには悪いことが起こる可能性を表すシンボルが付いている——汚職や公害から放火や干ばつまで。同じシンボルが二回プレイされると効果は悪化する：干ばつのシンボル1つは何も起こらないが、干ばつのシンボル2つは壊滅的な効果をともなう干ばつが起きることを示す。明らかに、トランプは現実の干ばつの発生とは

何の関係もない。カードはアフリカの大都市に住む人々のコントロールをほとんど超えた悪い出来事をシミュレートする方法である。それはまた便利にゲームの他のメカニクスと結びついている：すべてのプレイヤーは汚職、スケープゴーティング、放火のような悪い出来事から利益を得ることができる秘密の役割も得る。

正しく使用されれば、インデクシカルおよびシンボリックな縮小は、その構造的で創発的な特性にあまり影響を与えることなく、システム内の要素の数を減らす。Get H2Oの例では、日常的に必要な多くの同様のリソースは一つだけで置き換えられている：水。これらのリソースへのアクセスを持つことを伴うフィードバック構造は、ほぼ変わらない。例えば、フィードバックループの数は影響を受けない。実際、ゲームはこれらの構造的特徴を、不要な詳細を取り除くことによって強調する。ゲームシステム内の要素の数を減らすことにより、すべてを追跡するプレイヤーの認知的負担も減り、プレイヤーはこれらの特徴とそれらが許す戦略的インタラクションに、あるいはGet H2Oの場合はそれらが持つ社会的含意に、より集中できるようになる。

さらに多くの利点がある。インデクシカルおよびシンボリックなシミュレーションを使用するシステムは体験を凝縮でき、プレイの完全なセッションを、プレイがリアルタイムで表すものよりもはるかに速く実行することを可能にする。プレイヤーは自分のアクションの結果に素早く効率的に直面する。プレイヤーがルールを束にして「彼らのシステム理解のより容易で迅速な改善のために」扱うことを可能にする。一方ではこれによりプレイヤーはプロセスをより頻繁に経験でき、他方ではこれは多くの商業的エンターテインメントゲームを駆動するエージェンシーとパワーの楽しい体験に貢献する。Get H2Oゲームでは、これは確かにデザインゴールの一つであった。ゲームはおよそ45分でプレイでき、プレイヤーは異なる戦略を効率的に実験しながら失敗のコストを軽減できる。

ゲームデザイナーにとっても利点がある。その本質に縮小されたゲームシステムは、よりよく管理でき、バランスを取りやすくなる。多くの部分がなくても、デザイナーはゲームの創発的振る舞いに直接寄与する要素と構造に集中でき、その振る舞いをより容易に望ましい形に微調整できる。ゲームはリアリズムではなく非イコニック的、離散的無限性を目指すのがよい。これは経済的により実行可能であるだけでなく、芸術的にもより興味深く、より効果的なコミュニケーションを可能にする。

ゼルダの伝説シリーズは、わずかなゲームオブジェクトと関連するルールだけが多くの興味深いチャレンジに組み合わされるゲームプレイデザインの素晴らしい例である。これらのオブジェクトとそのルールの各々の価値は、ダンジョンを冒険することの何らかのリアルな側面を表現するパワーから来るのではなく、他のオブジェクトとルールとの潜在的な組み合わせを形成するところから来る。このシリーズが有名な探索チャレンジは、ほとん

ど常にシンプルで再利用可能なゲームプレイメカニクスの組み合わせの結果であり、しばしばかなりインデクシカルまたはシンボリックである。例えば、ゼルダの伝説：トワイライトプリンセスでは、プレイヤーは「Forest　Temple」レベルで「ゲイルブーメラン」を見つけることができ、それは風力で作動するスイッチを起動し、小さなアイテムをプレイヤーのところに運ぶのに十分な突風を発生させる。このブーメランは「ボムリング」——プレイヤーがつかんだ後数秒で爆発する小さなクリーチャー——を運ぶために使用できる。この組み合わせの効果的な使用は、そのレベルの最終ボスを倒すために必要である。同じブーメランはニンテンドーDS版のゼルダの伝説ゲームでも使用されているが、この場合プレイヤーはスタイラスを使ってブーメランの軌道を自由に描き、障害物の周りに非現実的に誘導することができる。プレイヤーはこのような構造を、本質的に一貫性があるため評価する。そして以前にも指摘されたように、一貫性はゲームプレイへの強い貢献要因である（Poole, 2000, 64-66）。そのような構造の鑑賞は本質的に美的鑑賞であるとさえ論じることができる（Huizinga, 1997, 25）。それは、興味深い構造的特質から興味深い振る舞いが生まれるシステムを構築するゲームデザイナーの技術の鑑賞である。それはゲームがどのように構築されたか、そしてそれがどのように構造化されているかに注意を払うことを強いる（cf. Ryan, 2001, 176）。

これらのゲームから創発する意味は、詳細でリアルなシミュレーションを目指すゲームよりも必ずしも詳細でなかったり価値が低かったりするわけではない。それどころか、ゼルダの伝説における探索のチャレンジがよりabstract（抽象的）であるように、ゲームが扱うスキルと知識はより汎用的であり、イコニックでないゲームのメッセージは、ゲームの特定の設定の外側にはるかにうまく適用可能である。これは特に、ゲームとそのすぐのプレミス（前提）を超えた価値を持つ何かを、ゲームを通じて表現したいときに有用である。

2.6 結論

ゲームとシミュレーションは表現的な形式を共有している：ルールのシステムを通じたソースシステムの表現。純粋な娯楽としてプレイされるゲームでさえ何かをシミュレートしている。ルールベースの表現は多くの異なる形をとりうる。伝統的なシミュレーションは正確なモデリングを主要なゴールとしているのに対し、ほとんどのゲームは楽しませるためにデザインされている。教育を目的とするゲームはその中間に位置すると言える：ある程度の正確さを追求するが、一般的にシミュレーションよりも多くのデザインの自由を享受する。しかし、多くのゲームはまだシミュレーションの規範に従っている。つまり、ソースをできるだけ近く正確に類似するルールを作成することによってソースシステムを

表現しようとする。このタイプのルールベースの表現は、一般的な記号論との類推によって、イコニックなシミュレーションと呼ばれる。

記号論との類推において、インデクシカルおよびシンボリックなシミュレーションの概念も、ゲームをシミュレーションから差別化する可能な道筋として探求された。上で指摘したように、ゲームのゴールはシミュレーションのゴールと同じではない。インデクシカルなシミュレーション——ゲームのルールがソースのルールと何らかの因果関係を持つ——とシンボリックなシミュレーション——ゲームのルールが慣習によってソースにリンクされている——は、はるかにシンプルなゲームシステムを可能にする。これらのシステムはより少ないルールと部分で構成されているが、その振る舞いや意味はそれほど複雑でなくてよい。ゲームのような非イコニックなシミュレーションのパワーは、ソースシステムを正確にモデル化する能力や、広大でリアルなゲームワールドの創造にではなく、表現力豊かなゲームメカニクスの効率的な使用にある。イコニックな表現の可能性が完全に探求されたと主張するつもりはない。しかし、私にとっては、シミュレーションのためのインデクシカルおよびシンボリックな構成要素を開発し、さらに重要なことに、そのような構成要素の特定の構成の効果を調査することによって、はるかに多くの進歩が可能であることは明らかである。創発的振る舞いは部分そのものよりも、ゲーム部分の相互関係から生まれる可能性がより高い。適切でシンプルな要素から複雑なシステムを作り出すのは、ゲームデザイナーの技術である。同等に複雑な（またはさらに複雑な）手段で複雑なシミュレーションを作り出すことにはほとんど技術がない。しかし、それは多くの開発者が目指しているところのようである。

この章で論じられたインデクシカルおよびシンボリックなシミュレーションは、イコニックなシミュレーションを超えることへの提案である。それらは、ゲームプレイが生まれる構造に影響を与えることなく、ゲームシステムをその最小限にまで縮小するのに役立つ理論的概念である。これにより、デザイナーは創発的振る舞いのバランスに集中でき、プレイヤーにはゲームによって体系化された遊びの意義、つまりジェネリックな知識を探求するためのより良い機会を提供する。

しかし、創発的なゲームプレイを持つゲームをデザインすることは、ルールがシンプルに保たれている場合でさえも、決して容易なタスクではない。創発的振る舞いは定義上予測不可能である。ゲームデザイナーの最善の策は、多くのプロトタイプを作成し、テストを続けることである。しかし頻繁なテストをもってしても、ゲームデザイナーは自分の経験と直感に頼ってゲームを作らなければならない。この実践は、より良いツールを作ることによって改善できる。特に、ゲームの創発的側面を認識し、それを駆動する構造的特質に焦点を当てたデザインツールである。このことから、第3章では以前の取り組みと既存の

デザインツールを論じ、第4章ではゲームメカニクスとゲームにおける創発の構造に対処する新しい代替的なスキームとしてMachinationsフレームワークを提示する。

第3章 ゲームデザイン理論

> その仕事を完璧にしようとする者は、まずその道具を研がなければならない。
> — 孔子

前の章ではゲームの性質に焦点を当てた。本章では、開発者がゲームをデザインする際に支援するための利用可能なツールと理論を論じる。多くのデザインガイドライン、方法、理論、ツールが過去数年にわたって開発されてきた。これらの中にはデザインプロセスを支援するために特に開発されたものもあれば、分析ツール、作業方法、または文書化技法として開発されたものもある。これまでに開発された、ゲームデザイナーを支援する主なアプローチと試みで、本章で論じるものは：デザインドキュメント、MDAフレームワーク、プレイ中心のデザイン方法論、ゲームボキャブラリー、デザインパターン、有限状態機械ダイアグラム、ペトリネット、そして最後に異なるタイプのゲームダイアグラムである。これらの方法のほとんどはデザインツールとして開発されたものではないが、それでもすべてがデザインツールとして使用できるか、新しいデザイン方法やツールの開発に影響を与える可能性がある。本章はデザイン方法とツールを開発する目的で、これらすべての方法の利点を論じる。

読者は、これらの方法に関する一般的な言説がかなり散漫であることに注意すべきである。ゲーム業界内では、そしてゲーム研究においても程度は低いが、固定されたボキャブラリーはない。多くの概念がかなり非公式に使用されており、用語はしばしば重複したり矛盾したりする。例えば、「ゲームデザインドキュメント」という用語は、さまざまな異なる理由のために作成されるさまざまな異なるドキュメントの幅広い多様性を包含する。さらに、分析方法とデザイン方法の間にはほとんど区別がないようで、二つの用語は時に互換的に使用される。私はできる限り原典の用語を使用するようにした。混乱を生じさせていないことを願う。

加えて、ゲーム業界のすべての人が、本章および以下の章で論じ提示されるようなゲームデザインへの方法論的アプローチの利点を認めているわけではない。デザイン方法論に対する最も一般的な二つの反論は、それらがゲームデザインにとってほとんど実用的価値がないこと、そしてゲームをデザインするという創造的プロセスを置き換えることはできないということである。結論を導く前に、最後のセクションでこれらの反論を取り上げる。

3.1 デザインドキュメント

ほぼすべてのゲーム会社がデザインドキュメントを作成する。インターネット上にはさまざまな企業が使用する多くの異なるテンプレートが見つかり、ゲームデザインを論じる事実上すべての本が独自のテンプレートを持っている。デザインドキュメントの概念と実践は、多様であるのと同様に散漫である。これらのドキュメントを書く多くの異なる理由があり、企業がそうするデザインプロセスにおける多くの異なる場面がある。ゲームデザインドキュメントは、構築される前のデザインを記録するために使用されることもあれば、ゲームが構築された後のデザインを記録するために使用されることもある。それらは通常、ゲームのコアメカニクスの説明、レベルデザイン、アートディレクションに関する注記、キャラクターとその背景などを含む。すべての詳細をカバーする詳細な長い記述を提唱する者もいれば、デザインのターゲットとデザイン哲学を捉える簡潔なドキュメントを好む者もいる。

長年にわたり、ゲームデザインドキュメントを書くことは一般的な業界慣行となったが、これらのドキュメントがどのように、いつ、またはどのような目的で書かれるべきかについての標準は生まれなかった。デザインの異なる部分に焦点を当てたり、デザインプロセスの異なる段階を促進したりするドキュメントの全セットを作成することは珍しくない（例えばAdams & Rollings, 2007; Rogers, 2010を参照）。デザインドキュメントに広く受け入れられたテンプレートがなければ、会社から会社へ、または大学からプロのキャリアへ持ち越すことができない。広く受け入れられたテンプレートがなければ、デザインドキュメントは標準的な方法論に成長することができない。ほとんどのデザインドキュメントが独自のスタイルを持ち、ゲームを記述するために独自のユニークな概念を使用しているという事実は、各個別のプロジェクトや会社の範囲を超えた汎用的な知識体系の作成を助けない。業界全体の標準が見えない現状では、デザインドキュメントが近い将来に有効になる可能性は低い。

さらに、デザインドキュメントは、ゲームの動的で創発的な振る舞いを扱うための適切なツールではないかもしれない。プロトタイプが作られる前に書かれるゲームデザインドキュメントは、ソフトウェア工学における要件ドキュメントに相当する。要件ドキュメントは、新しいカスタムビルトのソフトウェアアプリケーションの要件と機能を一覧にする。その作成は、ソフトウェア開発の「ウォーターフォールメソッド」における最初のステップの一つであり、各ステップが次に進む前に完了する。このドキュメントは通常ソフトウェアが構築される前に書かれ、しばしば請負業者とクライアントの間の合意の一部である。ウォーターフォールメソッドは、すべての要件が既知であり、ソフトウェアが構築される前に記録できることを前提としている。ソフトウェア工学において、機能設計を作

成し文書化することは長い実績のある慣行であるが、最近のアジャイル開発手法の人気の高まりにより、新しいソフトウェアアプリケーションの完全な機能設計を青写真として書くという慣行はその魅力を失いつつある。

ゲームのデザインとウォーターフォールメソッドを使用したビジネスアプリケーションのデザインとの間には、この慣行を一般的なカスタムソフトウェア開発からゲーム開発へ移転することを困難または非効率にする三つの重要な違いがある：

1. ゲームデザインは非常に反復的なプロセスである。デザイナーがどれほど経験豊富であっても、ゲームの構築中にゲームのデザインが変わる可能性が高い。ゲームの創発的な性質のため（第1章参照）、ゲームの振る舞いが実装される前にそれを正確に予測することはしばしば不可能である。ゲームにおいては、実装に対するデザインの変更は予想されるべきものである。[1]

2. すべてのゲームが請負業者とクライアントの文脈で作成されるわけではない。これは特にエンターテインメントゲームに当てはまり、その開発は商用のオフ・ザ・シェルフのソフトウェア開発とより多くの共通点を持つ。この文脈がなければ、ゲームが構築される前にデザインを文書化する必要性は低くなる。エンターテインメントゲームのパブリッシャーやファンダーはゴールとマイルストーンを設定するが、同じようには機能しない。

3. ゲームはメンテナンスや将来の開発を念頭に置いて構築されることはめったになく、将来の開発者を助ける文書を作成する必要性を減らしている。ゲーム業界で多くの続編が制作されているという事実にもかかわらず、多くの続編はゼロから構築され、驚くほど少ないコードが再利用される。ソフトウェア工学の観点からはこれは悪い慣行である。しかし、開発技術がまだ急速に進化しており、ハードウェアの新世代がおよそ5〜6年ごとに利用可能になるため、この慣行はゲーム業界の観点からはより合理的である。[2]

多くのデザイナーはゲームデザインドキュメントを必要悪と見なしており、その慣行を完全に否定した者もいる（Kreimeier, 2003）。デザインは文書化され、チームに伝達される必要があることには誰もが同意するが、実際にはデザインドキュメントをほとんど見ない（Keith, 2010, 85-87）。Electronic Artsのクリエイティブディレクター、Stone Librandeは、彼がワンページデザインと呼ぶ技法を使ってこれらの問題のいくつかを回避する実験を行った（2010）。彼のアプローチは、複数ページの書かれたテキストの代わりに、データの可視化に近いデザインドキュメントを作成することである。ワンページデ

ザインドキュメントは、ゲームの本質を視覚的に捉えるポスターである（図3.1参照）。これらのドキュメントは、従来のデザインドキュメントに対して四つの利点を持つ：

1. ほとんどのデザイナーはそれらを作ることをより面白いと感じ、デザインドキュメントを作成する作業をより退屈でなくする。

2. 空間的な制約があるため（ページのサイズはデザイナーの気まぐれに委ねられるが）、デザイナーはゲームの本質に焦点を当てることを強いられる。これはドキュメントをゲームにしばしば見られるアジャイル開発プロセスにより適したものにする。

3. 人々はこれらのドキュメントの見た目を気に入る傾向があるため、壁に貼る傾向があり、その露出とインパクトを増加させる。

4. Stone Librandeは、チームメンバーがメモを書き込むことを招くために、ドキュメントに十分な余白を残すことを提案している。これによりドキュメントは最新に保たれる。

> **図3.1：** Librande（2010）によるワンページデザインドキュメントの例。

ワンページデザインドキュメントは、デザインドキュメントに関連する問題のいくつかを解決するが、すべてではない。ゲームプレイ、メカニクス、ルールを文書化するための標準はまだない。各ワンページデザインドキュメントは特定のゲームのために作成され、その製品は理解可能でコミュニケーション力があるべきではあるが、標準を設定することはできない。加えて、ディテールの欠如は——それがワンページデザインドキュメントをより柔軟にし、したがってその強みの一つであるが——デザインの記録にはそれほど適さなくする。ワンページデザインドキュメントはデザインのビジョン、ゴール、方向を捉えることには非常に優れているが、同時に技術的な青写真として機能することはできない。

3.2 MDAフレームワーク

MDAフレームワーク——MDAはメカニクス（mechanics）、ダイナミクス（dynamics）、エステティクス（aesthetics）の略——は、ゲームデベロッパーズカンファレンス（GDC）のゲームデザインワークショップを構造化するために、少なくとも11年連続で使用されてきた（2001年から2011年）。[3]デザインドキュメントの慣行とは対照的に、MDAフレームワークはゲームデザインに関わる困難に対するジェネリックなアプローチを非常に意識的に提示しようとしている。それは非常に影響力があり、世界中の

大学のゲームデザインプログラムで最も頻繁に見られるフレームワークの一つであるように思われる。おそらくそれは、業界が標準化されたゲームデザイン手法に最も近いものである。

> **図3.2：** MDAフレームワーク（Hunicke et al., 2004に基づく）。

MDAフレームワークはゲームを三つのコンポーネントに分解する：メカニクス、ダイナミクス、エステティクス——これらはゲームのルール、そのシステム、そしてそれがもたらす楽しさに対応する（Hunicke et al., 2004）。MDAフレームワークは、ゲームのデザイナーと消費者がゲームに対して異なる視点を持っていることを教えている。消費者はまずエステティクスに気づき、その後ダイナミクスとメカニクスに気づく。デザイナーはその逆に動く。デザイナーはまずメカニクスを作り、その上にダイナミクスとエステティクスを構築する（図3.2参照）。

MDAフレームワークは反復的なデザインプロセスをサポートし、各レイヤーにおける変更がゲーム全体にどのように影響するかをデザイナーが評価するのを助けるために設計されている。各レイヤーにはそれ自身のデザインゴールとゲームへの効果がある。メカニクスはゲームが許すアクションを決定し、ゲームの動的な振る舞いに影響を与える。ダイナミクスレイヤーはランダム性や複雑さのような概念を取り扱い、ゲームの振る舞いを説明する。最後に、エステティクスレイヤーはゲームの感情的なターゲット——プレイヤーへの効果——に関わる。MDAフレームワークは主要なエステティックターゲットとして8種類の楽しみを記述している。8種類の楽しみとは：感覚（sensation）、空想（fantasy）、ナラティブ（narrative）、チャレンジ（challenge）、仲間意識（fellowship）、発見（discovery）、表現（expression）、そして服従（submission）である（Hunicke et al., 2004; LeBlanc, 2004）。

MDAフレームワークの影響力と、長きにわたるGDCゲームデザインワークショップにもかかわらず、概念的フレームワークとしてのMDAはその予備的段階を超えたことがないように思われる。メカニクスとダイナミクスのレイヤーの区別は、元の著者たちにとっても必ずしも明確ではない（LeBlanc， 2004参照）。メカニクスは明らかにゲームルールである。しかしダイナミクスは同じルールから創発する。しかしながら、元のMDA論文はサイコロやその他の乱数生成器のようなゲームデバイスをダイナミクスのレイヤーに配置している。私にとっては、それらのデバイスはメカニクスのレイヤーにある方がよりふさわしく思える。同様に、エステティクスレイヤーはプレイヤーの感情的反応のみを含むように見える。これらの反応のきっかけとなるビジュアルやストーリーは——一般的にエステティクスの一部として理解されるであろう——フレームワークからは欠けているように思われる。8種類の楽しみは、ほとんど深く探求されていない、かなり恣意的な感情的ター

ゲットのリストで構成されている。短い一文の説明を除けば、Hunicke et al.は楽しみの種類が何を伴うかについて正確な記述を提供していない。彼らはリストが完全ではないと述べているが、なぜこれら8つを記述するのか正当化しておらず、あといくつの楽しみの種類が見つかると予想しているのかのヒントさえ与えていない。さらに、ゲームの主なターゲット感情としての「楽しさ」という概念全体が、Ernest AdamsとAndrew Rollings（2007，119）やSteven Johnson（2005，25-26）らによって批判されている。ゲームはより広い多様性の感情的反応をターゲットにすることを目指すことができる。追加のMDA論文がいくつか年を追って発表されたが（LeBlanc，2006など）、これらの懸念を払拭するものではなかった。

3.3 プレイ中心デザイン

反復的ゲームデザインのより徹底した方法は、Tracy Fullerton et al.（2006）によって記述されている。自らの方法を記述するために「プレイ中心デザイン」という用語を造語し、Fullerton et al.はプレイヤーを短いデザインサイクルの中心に置くことを提唱している。ゲームプロトタイプを素早くビルドし、頻繁にテストすることを推奨する。短いデザインサイクルのため、より革新的なオプションを、労力と時間のリスクを低減しながら探求できる。

プレイ中心デザインはゲーム内の二つのレベルを区別する：フォーマルなコアとそれを取り囲むドラマティックなシェル。ゲームのフォーマルなコアはルール、目的、手続きで構成され、一方ドラマティックなシェルは前提、キャラクター、ストーリーで構成される。組み合わさることで、この二つのレイヤーはプレイをサポートする動的で創発的な振る舞いに寄与する。この振る舞いを特定のターゲット体験にチューニングするのが、プレイ中心デザインの目的である。この文脈で、Fullerton et al.はKatie SalenとEric Zimmermanのゲームの記述を「二次デザイン問題」（2004， 168）として再述している。デザイナーはゲームをデザインするが、ゲームが体験を提供する。デザイナーは体験を直接作り出すのではない。

このトレンド——プレイヤーをデザインプロセスに巻き込むこと——は、学界と開発スタジオの両方で勢いを増している。ソフトウェア工学に起源を持つヒューマン・センタード・デザインまたはユーザー・センタード・デザインは、このトレンドに大きな影響を与えてきた。この点に関してMicrosoftのゲームスタジオが先駆者であることは驚くべきことではない。Microsoftは通常のソフトウェア開発で同様の方法に関して多くの経験を持っているからである。Pagulayan et. al.（2003）は、Microsoft内でいくつかのゲーム

を開発するために使用されたヒューリスティクスと構造化されたユーザーテストを記述している。じわじわとではあるが確実に、これらの方法はゲームデザインの不可欠な部分となり、ヒューリスティック評価（Sweetser & Wyeth, 2005; Schaffer, 2008）のような定性的方法と、プレイヤーの死亡位置をレベルマップ上にプロットするような定量的メトリクス（Swain, 2008）の組み合わせにますます依拠するようになっている。

プレイ中心デザインは、ゲームをデザインするプロセスに焦点を当てている。デザインプロセスを構造化し、プレイヤーを巻き込み、プロトタイプからデータを収集し、何度も何度も反復することにより、最終製品、完成したゲームは、デザインチームが作り得る限り良いものになることを確実にするためにすべてが行われる。プロのゲームデザイナーにとって、これらの方法は（少なくともそうあるべき）定番のツールである。それらはゲームデザインのプロセスをより楽にするものではないが、デザイナーがトラックに留まるのを助け、タスクをより小さなサブタスクの連続に分解し、高品質の最終製品に向けて着実に前進させる。

適切な方法とツールがあれば、このプロセスを洗練させることができる。プレイ中心アプローチは、各反復を加速したり、各反復で行われる改善を増加させたりする方法から恩恵を受けるだろう。適用可能と思われるいくつかのタイプの方法がある。確かに、ゲームデザインの正式なモデルは広く受け入れられているものの中にあるが、プレイテスト中に収集したデータを集め処理する技法もまた同様である。

3.4 ゲームボキャブラリー

ゲームをデザインすることだけでなく、それについて話すことも既に困難である：その内部の仕組みを記述するための共通の言語が存在しない。ゲームをルールベースのシステムとして論じる本や記事は数多くあるが、それらのほぼすべてが自分自身の言葉を選択している。多くの場合、これらのボキャブラリーは特定のゲームを記述することには非常に長けているが、より汎用的なボキャブラリーに昇華されることはめったにない。

1999年のGamasutraの記事で、ゲームデザイナーのDoug　Churchはゲームデザインのための共通のボキャブラリーのフレームワークを作ることに着手した（Church,　1999）。このフレームワークによれば、ゲームデザインボキャブラリーは「フォーマルな抽象的デザインツール」で構成されるべきであり、「フォーマル」はボキャブラリーが正確である必要があることを示し、「抽象的」はボキャブラリーが単一のゲームの特殊性を超越しなければならないことを示す。Churchにとって、ボキャブラリーはツールのセットとして機

能するべきであり、異なるツールは異なるタスクに適し、すべてのツールがすべての特定のゲームに適用可能というわけではない。

Doug Churchは自身の記事で三つのフォーマルな抽象的デザインツールを記述している：

1. **意図（Intention）：** プレイヤーは、ゲームワールドにおける現在の状況とゲームプレイオプションの理解に応じて、自らの創造による実行可能な計画を作ることができるべきである。

2. **知覚可能な結果（Perceivable Consequence）：** ゲームワールドはプレイヤーのアクションに明確に反応する必要がある。プレイヤーのアクションの結果は明確であるべきである。

3. **ストーリー（Story）：** ゲームにはナラティブの糸があるかもしれない——デザイナー主導かプレイヤー主導かを問わず——出来事を結びつけ、プレイヤーをゲーム完了に向かって駆り立てる。

これら三つのツールは決して完全でも網羅的でもないリストを形成しており、Doug Churchもそう意図していなかった。1999年から2002年の間、Gamasutraのウェブサイトはこのフレームワークについて人々が議論し拡張できるフォーラムをホストしていた。「デザインツール」という用語は「デザインレキシコン」という用語にすぐに置き換えられ、フォーマルな抽象的デザインツールがデザインツールよりも分析ツールとしてより成功していることを示していた。Bernd Kreimeierは「少なくとも25の用語がほぼ同数の寄稿者によって提出された」と報告している（Kreimeier, 2003）。プロジェクトとしてのフォーマルな抽象的デザインツールは放棄された。しかしながら、Doug Churchの記事は、彼のフレームワークが定着することはなかったとはいえ、ゲームデザインのためのボキャブラリーの欠如に対処する最も初期の試みの一つとしてしばしば認められている。

Doug Churchが灯した松明を引き継ぐ研究者は数人いる。Hal BarwoodとNoah Falsteinが始めた「400プロジェクト」はその一例である（Falstein, 2002）。BarwoodとFalsteinは、より良いゲームにつながるべきゲームデザインの400のルールを見つけ記述するという目標を設定した。プロジェクトのウェブサイトは112のルールを掲載している。[4]しかしながら、プロジェクトは放棄されたようでもある。ウェブサイトの最終更新は2006年であった。

Craig Lindley（2003）は直交するタクソノミーを使用して、ゲームを、ナラトロジー（ストーリーに焦点）、ルドロジー（ゲームプレイに焦点）、シミュレーション（リアリズムに焦点）のような支配的なゲーム形式によって定義される仮想的空間にマッピングす

る。個々のゲームは、これらの極端のそれぞれへの相対的近さに応じて空間にマッピングできる（図3.3参照）。Lindleyはいくつかの可能な、相補的なタクソノミーを、一次元、二次元、三次元の空間を使って記述している。彼はこれらのタクソノミーを、特定のゲームプロジェクトの意図されるデザインターゲットをデザインチームに知らせるためのハイレベルなロードマップとしてデザインした。タクソノミーはまた、他の分野から借用した適切なツールと技法も提案する。ナラティブ側に偏るゲームは、主にシミュレーションであるゲームよりも伝統的なストーリーテリング技法からより多くの恩恵を受ける。Lindleyのゲームタクソノミーは、多くのフォーマルな抽象的デザインツールが埋め込まれうる体系的なフレームワークを提供し、そうでなければルーズなラベルの集合体のままであったものに構造を与えている。

> **図3.3：** Craig Lindleyのタクソノミーの一つ（Lindley, 2003に基づく）。

ゲームオントロジープロジェクトは、ゲームの共通ボキャブラリーの概念をさらに別の方向に発展させている。このプロジェクトはゲームデザインの知恵の断片を一つの大きなオントロジーにまとめようとしている。オントロジーとは階層的な組織を持つ大規模な分類スキームである。オントロジーの各エントリーはゲームに見られる共通構造を記述する。それは強い例と弱い例をリストし、親と子のカテゴリーをリストする。例えば、オントロジーのエントリー「所有する」は、ゲームエンティティが他のゲームエンティティを所有できるゲーム構造を記述するために使用される。例としては、「キノコ」や「スター」などを集めるゲームエンティティ「マリオ」が挙げられる。「所有する」は「エンティティ操作」エントリーの子であり、そしてそれは三つの子を持つ：「獲得する」、「所持する」、「交換する」（Zagal et al., 2005）。[5]

ゲームオントロジープロジェクトは、良いゲームの作り方を処方することなく、ゲームのデザインスペースを探求することを目指している。Doug Churchのフォーマルな抽象的デザインツールよりも、それは主に分析ツールであり、ゲームを構築するよりも理解することを目指している。これはこのゲームボキャブラリーおよび他のゲームボキャブラリーの一般的な特性である。分析ツールとしての成功は、デザインツールとしての成功に容易に変換されない。明らかに、共通のゲーム構造を記述する高レベルで一貫した言語の開発は長期的にはデザイナーを助けるだろうし、すべてのボキャブラリー構築者が指摘するように、ゲームデザインの比較的未開拓な領域をマッピングするのに大きな助けとなりうる。実際、すべての著者はゲームの共通の側面を記述する概念のランダムな組み合わせを選択するだけで、自分たちのボキャブラリーがブレインストーミングツールとしてどのように使用できるかを記述している。しかし、どれほどこの実践が有用であっても、それは通常アイデアの生成を助けるだけである。これはゲームを構築する全プロセスのほんの一部に

すぎず、ボキャブラリーを学ぶために多くの新しい概念に精通しなければならないデザイナーの側にはかなりの投資が必要である。例えばゲームオントロジープロジェクトは、オントロジー内の他の複数のエントリーとそれぞれが関連する百以上の個別のエントリーで構成されている。ゲーム開発者にとって、そのサイズのボキャブラリーを学ぶことへの投資に対する実際のリターンが何であるかを見ることは難しいかもしれない。

ゲームの共通ボキャブラリーに向けた多くの異なるアプローチはこの問題を悪化させる。[6]すべてのボキャブラリーは独自のユニークなアプローチと用語を持っている。これらすべてのアプローチがどこで重複し衝突するかを単純に決定するだけで、大規模な学術研究プロジェクトとなる。たとえゲームデザイナーがこれらのボキャブラリーの一つを学ぶための時間と労力を投資したとしても、異なるボキャブラリーを学んだ誰かと効果的に協力したり知識を共有したりすることは依然として問題となるだろう。これらすべてのボキャブラリーが共有しているように思われる唯一のことは、ゲームデザイナーに拒否されていることである。Daniel　Cookの言葉を借りれば：「ゲームデザインの学術的定義は言葉が多すぎ、提案される用語を人々が実際に使用できる十分に明白な実用的応用がない」（2006）。

3.5 デザインパターン

Staffan BjörkとJussi Holopainenのゲームデザインパターンに関する研究もまた、ゲームデザインのためのボキャブラリーの欠如に対処しようとしている（2005, 4）。しかし、彼らのアプローチは、Christopher　Alexanderの著作に探求された建築とアーバンデザインに見られるデザインパターンからインスピレーションを得ている点でやや異なる。Alexanderによれば：「ある中心的な質があり、それは人、街、建物、あるいは荒野における生命と精神の根本的な基準である。この質は客観的で正確であるが、名前を付けることはできない」（Alexander et al., 1979, ix）。Alexanderはこの質を記述する方法を模索し、それは彼がパターンと呼ぶもの――特定の問題に対する繰り返し見られる解決策――に見出される。各パターンは、関連する問題と文脈とともに記述される。相互に関連するパターンのネットワークがAlexanderのパターン言語を構成する。パターンのシステムとしてはパターン言語が重要であるが、パターン言語を作る試みの多くは言語としてのシステム的側面を無視し、代わりに個々のパターンに焦点を当てている。

BjörkとHolopainenの研究は、200を超えるパターンを記述する大きなコレクションをもたらした。[7]各パターンはテンプレートに従って記述される：名前、コアの定義、簡単な説明、パターンの使用法、その結果、そしてBjörkとHolopainenのコレクション内の他の

パターンとの関係。パターンは互いの文脈において定義されるため、Alexanderのパターン言語の特性の一部を保持している。パターン間の関係はパターンのシステムを形成する。パターンのコレクションが大きくなるにつれて、そのネットワークと関係はより良い方法でゲームのデザインスペースを記述するようになる。個々のパターンはかなり漠然と定義されている場合があるが、その関係のネットワークは潜在的にはるかに強力であり、一般的なボキャブラリーの欠如に対処するためにより良い位置にある。

しかし、ゲームデザインパターンにはいくつかの欠点がある。まず、パターンのコレクションはかなり大きく、依然として増え続けている。ゲームデザインのためのボキャブラリーとしての使用を検討する前に、投資すべき学習の量が多い。次に、パターンはゲームプレイの体験のレベルで記述される傾向がある。パターンとゲームのルールの間にはかなりのギャップがある。BjörkとHolopainenの本では、ゲームの構造コンポーネントの説明がパターンの説明に先行しているが、構造コンポーネントとパターンの間の正式な関係は確立されていない。一つのパターンはさまざまな方法でゲームに実装でき、コンポーネントの一つの構成はさまざまなパターンを引き起こすかもしれない。ゲームのルールとパターンの間のこのギャップは、パターンコレクションをデザインツールとして使用する有効性を制限する。デザイナーがパターンの組み合わせに到達したとしても、その組み合わせを実際のゲームルールの構成に変換することは、その時点ではまだかなり未指定のままである。第三に、ゲームデザインパターンは、Alexanderのパターン言語とは重要な点で異なっている。Alexanderのパターン言語では、個々のパターンは「良い」デザインのインスタンスである。しかし、BjörkとHolopainenのゲームデザインパターンでは、パターンはより記述的な役割を果たしている。パターンが存在するかしないかは良くも悪くもない。実際、BjörkとHolopainenのコレクションには明らかにネガティブなパターンが見られる。例えば、Analysis　Paralysis（分析麻痺）パターンはプレイヤーが選択肢に圧倒される問題を記述しており、Kingmakerパターンは弱いプレイヤーがゲームの勝者を決定できるデザインの問題を記述している。

ゲームデザインパターンはソフトウェア開発のデザインパターンとより共通性があるかもしれない。ソフトウェアデザインパターンの概念はGang　of　Four（Gamma　et　al., 1994）によって広められ、最終的にはAlexanderのパターン言語にもインスパイアされている。ソフトウェアデザインパターンはオブジェクト指向プログラミングにおける繰り返し見られるデザインの問題に対する認知された解決策を記述しており、ソフトウェアエンジニア間のコミュニケーションツールとして成功を収めている。BjörkとHolopainenの研究における重要な違いは、ソフトウェアのデザインパターンはすべてUMLの共通言語を使用して記述されるということである。UMLはソフトウェアエンジニアリングのビジュアル言語であり、パターンはUMLダイアグラムとして視覚化される。UMLは、ソフトウェアデ

ザインパターンに対してソフトウェアの仕組みの的確で検証されたダイアグラムを提供することにおいて最高に重要であり、同じことがゲームデザインパターンには欠けている。BjörkとHolopainenのコレクションには、ゲームメカニクスの的確で検証されたダイアグラムが不足しているのである。

3.6 有限状態機械

有限状態機械はソフトウェアエンジニアリングとコンピュータサイエンスにおいて一般的なダイアグラムの一種である。それらは有限数の状態を持つプロセスを視覚化し分析するために使用できる。図3.4に示すダイアグラムは三つの状態と、状態間の遷移として視覚化される六つのイベントを持つ簡単な例を示している。有限状態機械にはさまざまな形式がある。ヒエラルキカルな有限状態機械は状態を入れ子にすることを許し、ステートチャートは遷移にガードとアクションを追加し、並列有限状態機械は同時状態を許す。

> **図3.4：** 三つの状態と六つの遷移を持つ有限状態機械。

有限状態機械はゲームAI（人工知能）の駆動メカニズムの背後で一般的に使用されている。例えばFu and Houlette（2004）やBuckland（2005）に記述されている。ゲームAIの文脈では、有限状態機械のダイアグラムはAIが制御する個々のゲームエンティティの振る舞いを記述するために使用される。通常、AIの多くの異なるインスタンスが同時にゲーム内で動作している。これにより、有限状態機械が記述する振る舞いから複雑なシステム振る舞いが創発することが可能になる。しかし、有限状態機械のダイアグラム自体はゲーム全体よりも個々のコンポーネントに焦点を当てている。Brent Fox（2006）は有限状態機械をゲームデザイン全般のツールとして使用している。しかし、彼は主にゲームの状態（メニュー画面対プレイ画面対インベントリ画面、など）をモデル化するためにそれらを使用しており、必ずしもゲームメカニクスをモデル化するためではない。

有限状態機械に関連する問題は、その爆発する状態問題である。これは遷移が存在する状態の数よりもはるかに速く増加するという事実に関連する。三つの状態は最大で六つの遷移を持つことができるが、四つの状態には既に最大十二の遷移があり得る。状態の数が多いシステムでは、ダイアグラムはすぐに判読不能なスパゲッティ状になる。ヒエラルキカルな有限状態機械はこの問題に対するエレガントな解決策を提供するが、一般的に遷移は有限状態機械のダイアグラム内の最も重要なゲームイベントをキャプチャし、多くの状態を持つ有限状態機械はデザインツールとして実用的でなくなる。ゲーム全体を記述するために必要とされるほとんどの有限状態機械はあまりに多くの状態を持ち、読みやすいダイアグラムには収まりきらない。

有限状態機械は確かにゲームの形式的な記述を構成する。しかし、有限状態機械はゲームの状態空間を直接モデル化する：それらはゲームのルールではなく振る舞いを記述する。ゲームの状態空間をモデル化することは、状態空間を直接分析できるという利点を持つ。一方、状態空間は一般的にルールよりもはるかに大きいため、創発的な振る舞いを持つ興味深いゲームをモデル化するには多くの状態が必要である。実際、有限状態機械は状態空間が爆発する傾向のあるゲームにとって好ましい形式的な記述ではない。

3.7 ペトリネット

ペトリネットは並行するプロセスを記述しモデル化するために使用される図式的技法の一種である。それらはCarl Adam Petriによって発明され、1962年に彼の論文（Petri, 1962）で紹介されて以来、かなりの注目を集めてきた。ペトリネットはスーパーバイザリーコントロール理論で使用される一般的なツールであり、つまり制御された並行プロセスの振る舞いをモデル化するために使用される。ペトリネットについてはPeterson（1981）やMurata（1989）による調査論文など、多くの資料が書かれている。

ペトリネットはプレースとトランジションからなる二部有向グラフとして表現される。プレースはアークによってトランジションに接続され、トランジションはアークによってプレースに接続される。プレースは任意の数のトークンを保持することができる。ペトリネット内のすべてのプレースにわたるトークンの分布は、ネットのマーキングと呼ばれ、モデル化されているシステムの状態を表現する。トランジションは発火（fire）してトークンを移動させることでシステムの状態を変化させることができる。トランジションが発火すると、その入力アークによって接続されたプレースからトークンを消費し、その出力アークによって接続されたプレースにトークンを生成する（図3.5参照）。トランジションは入力プレースのすべてが少なくとも一つのトークンを含む場合にのみ発火できる。

> **図3.5：** 二つのプレースと一つのトランジションからなるペトリネット。トランジションは発火でき、上のプレースから一つのトークンを消費し、下のプレースに一つのトークンを生成する。

ペトリネットは強力な数学的基盤を持ち、多くの分析技法が存在する。例えば、ペトリネットの可達性グラフとカバラビリティグラフ（これらは有限状態機械として描かれる）の助けを借りて、ネットを介してトークンの特定の分布に到達できるかどうかを推論することが可能である。カバラビリティツリーはネットが無限であるか（境界なしで増加するトークン数を持つ）否かを決定でき、他の分析方法はデッドロック（いかなるトランジションも発火できないネット状態）の可能性を推論できる。

ペトリネットは、ゲームのAI（Brom and Ondráček, 2009参照）やゲームのレベルデザイン（Araújo and Roque, 2009参照）のモデル化に使用されてきた。Brom and Ondráčekにとって、ペトリネットの利点は、ペトリネットが視覚的であること、並行プロセスに適していること、そしてゲームコンテンツの作成者がそれらの使用方法を学ぶことが比較的容易であることである。Araújo and Roqueはペトリネットを使用してゲームのレベル構造をモデル化し、ペトリネットの分析はレベルの完了可能性を検証するのに役立てることができると論じている。

これらの利点にもかかわらず、ペトリネットにはゲームのモデル化ツールとしてのいくつかの欠点がある。まず、ペトリネットはプレースとトランジション、そしてそれらの間の接続という限られたボキャブラリーに限定される。これは数学的分析に有利であるが、ペトリネットのダイアグラムを読みにくくし、したがってゲームデザイナー間のコミュニケーションツールとしての使用を制限する。次に、ペトリネットの分析方法のほとんどは非常に大きなネットに対して非実用的であり、最も興味深いゲームはかなり大きなネットを必要とする。さらに、純粋なペトリネットのトークンは区別できない——「色なし」である——ため、異なるタイプのリソースが使用されるゲームの記述が複雑になる。カラードペトリネットの分析技法はまだ未開拓の面が大きい。

3.8 ゲームダイアグラム

ゲームの研究者、デザイナー、そして愛好家は、ゲームを記述するために多くの異なるダイアグラムの形式を使用してきた。これらのダイアグラムのいくつかは、ゲーム内の特定の構造、例えばゲーム空間のレイアウト、スキルツリー、あるいは人工知能の振る舞いなどに焦点を当てている。これらのダイアグラムは、それぞれの特定の目的には非常に有用であるが、ゲーム全体を記述しデザインするためのツールとしては限定的な使用にとどまる。

一般的に使用されるダイアグラムの形式の一つは、フローチャートを使用してゲームプレイの進行をマッピングするものである。図3.6に示されるフローチャートは典型的な戦略ゲームのプレイのフローを記述しており、プレイヤーが毎ターン決定を下さなければならないことを示している。同様のダイアグラムはKatie SalenとEric Zimmerman（2004, 332、Hallfordの初出）によっても提示されている。

> **図3.6：** ゲーム内の典型的な戦略ゲームのフローのフローチャート。

これらのダイアグラムはゲーム内のアクションのフローを追跡するのには役立つが、ルールの効果を記述するのには適していない。良いゲームはルール間の多くのインタラクションを特徴とし、そのような、しばしば循環的な、インタラクションはフローチャートではうまく記述できない。さらに、フローチャートの直線的な性質は、非線形的で多くの可能な道筋を特徴とするゲームプレイの構造に合っていない。

Raph　Kosterは、ゲームが報酬構造としてどのように機能するかを図式化する独自のダイアグラムを使用している（Koster, 2005, 86-91）。彼のダイアグラムは一般的なフローチャートよりも少し精巧であるが、それでも動的なゲームプレイのすべてのニュアンスを表現するには限界がある。

より興味深いアプローチはDan Cookのスキルアトムである（Cook, 2007）。これはゲームプレイのプリミティブ――スキルを磨く最小単位――をモデル化しようとするダイアグラム技法である。各スキルアトムはアクション、シミュレーション、フィードバック、そしてプレイヤーのモデルの更新で構成される。スキルアトムはそのようなフィードバックループを記述するためにうまく機能するが、まだメカニクス間のインタラクションを記述する方法としては限定的である。

Katie SalenとEric Zimmermanは彼らのゲームデザインの教科書Rules of Play（2004, 152）において、インタラクティブなシステムの構成的な記述と結合させたダイアグラムを提案している。彼らのダイアグラムは、特にゲームの経済――リソースがどのようにフローし、交換され、変換されるか――を視覚化するのに優れている。

Ernest AdamsとJoris Dormans（2012）は、ゲームにおける「内部経済」の概念と、それをダイアグラム化する標準的な方法を開発した。ゲームの内部経済は、ゲーム内のリソースのフローと交換を記述する。これは、ゲームのマクロレベルの力学――プレイヤーがリソースをどのように獲得し、使い、蓄積するか――を理解するための有用なツールとなる。このダイアグラム形式は本論文の第4章以降で詳しく論じるMachinationsフレームワークの基礎の一つとなっている。

ゲーム経済は、ソース、シンク、コンバーター、トレーダーの四つの基本要素で構成される。ソースはリソースを生成し、シンクはそれらを消費する。コンバーターはあるタイプのリソースを別のタイプに変換し、トレーダーはプレイヤーがリソースを交換できるようにする。これらの基本要素を使用して、多くのゲームの経済構造を効果的に記述し可視化することができる。

ゲームの内部経済をダイアグラムとして図式化することは、有限状態機械やペトリネットよりもゲームのダイナミクスをゲームデザイナーにとって直感的に理解しやすい方法で表

現する。これらのダイアグラムはゲーム内のリソースのフロー、フィードバックループ、ボトルネック、そして他の構造的特徴を明確にし、デザインの問題や不均衡を特定するのに役立つ。

3.9 業界の懐疑論

ゲーム業界の多くの人々は、ゲームデザインへの方法論的アプローチに懐疑的である。形式的な方法やモデルに対する最も一般的な二つの反論は次の通りである：1）形式的なモデルにはゲームデザインにとって実用的価値がほとんどない、2）形式的なモデルはゲームデザインの創造的プロセスを置き換えることはできない。

最初の反論に対しては、本章で論じたツールの多くが実際にゲームデザインの実践に直接的な実用的応用を持っていないことを認めなければならない。ゲームボキャブラリーはゲームについて考え議論する方法を提供するが、デザインの問題を解決する直接的な手段を提供するものではない。同様に、MDAフレームワークのような概念的フレームワークは視点を提供するが、実際に何かを生み出すツールではない。しかし、形式的なアプローチがゲームデザインに対して何の実用的価値も持たないという議論は、あまりにも急ぎすぎた結論であると思われる。これらのアプローチの多くはまだ初期段階にあり、その潜在的な価値はまだ十分に探求されていない。デザインパターン、有限状態機械、ペトリネット、そしてゲーム経済のダイアグラムは、すべてゲームデザインへの直接的な応用の可能性を示している。

二番目の反論——形式的なモデルはゲームの創造的プロセスを置き換えることはできない——に対しては、この反論が正しいことを直ちに認めなければならない。形式的なモデルは確かにゲームをデザインする際の創造性の必要性を置き換えることはできない。しかし、これは形式的なモデルが不要であることを意味するものではない。ゲームデザインは確かに創造的なプロセスであるが、他の創造的な分野——例えば建築、映画制作、ソフトウェアエンジニアリング——はすべて、創造的プロセスを阻害することなく、それをサポートし強化するために、形式的なツールと方法を活用している。建築においてブループリントと構造計算は、建物のデザインをより創造的にするのではなく、デザイナーが自分のアイデアを実現可能な方法で具体化するのを助ける。同様に、ゲームデザインの形式的ツールは創造性を置き換えるためではなく、創造的なアイデアがより効果的に実現されるのを支援するために意図されている。

ゲームデザインツールの目的は、デザインプロセスを自動化したり、創造性を不要にしたりすることではない。そうではなく、それらはデザイナーがより情報に基づいた決定を下

せるようにすること、デザインのコミュニケーションを改善すること、そしてプロトタイピングと反復のプロセスを加速することを目指している。良いツールはデザイナーの能力を拡張し、そうでなければ見えにくかったゲームシステムの特性を可視化する。

3.10 結論

本章ではゲームデザインに利用可能な既存のツールと方法をレビューした。デザインドキュメントは業界標準であるが、標準化の欠如と動的なゲームプレイの記述の難しさによって限界がある。MDAフレームワークはゲームをメカニクス、ダイナミクス、エステティクスの三つのレイヤーに分解する有用な概念的フレームワークを提供するが、予備的段階を超えていない。プレイ中心デザインはプレイヤーを巻き込む反復的なデザインプロセスに焦点を当てており、プロセスとしての健全さはあるが、各反復を改善するためのより良いツールが必要である。

ゲームボキャブラリーとデザインパターンは、ゲームの共通構造を記述するためのさまざまな試みを代表している。これらのアプローチはすべて、共通の課題に直面している：学習への投資が大きい割に実用的なリターンが限られること、そしてゲームデザイナーに広く受け入れられていないことである。

有限状態機械とペトリネットのような形式的な方法は、ゲームの特定の側面をモデル化するためには有用であるが、それぞれ固有の限界を持つ。有限状態機械は状態の爆発的増大に悩まされ、ペトリネットは限られたボキャブラリーによる可読性の問題がある。ゲームダイアグラムはゲームの構造を視覚化するためのさまざまなアプローチを提供しているが、ゲームメカニクスの的確で検証されたダイアグラムに共通する標準はまだ存在しない。

本章でレビューしたすべてのアプローチに共通する観察は、ゲームメカニクスとそこから生まれるダイナミクスを的確かつ包括的に表現する標準化されたダイアグラムツールが欠けているということである。ソフトウェアエンジニアリングにUMLがあるように、ゲームデザインにも、ゲームメカニクスを記述し、その動的な振る舞いを予測し分析するための共通のビジュアル言語が必要である。次の第4章では、この欠落に対処するために開発されたMachinationsフレームワークを紹介する。Machinationsはゲームの内部経済をモデル化するためのビジュアル言語であり、ゲームメカニクスの図式化、シミュレーション、そして分析を可能にするツールである。

1. ただし、これはビジネスソフトウェアにもますます当てはまりつつあることを指摘しておかなければならない。[?]

2. ゲームスタジオに販売される再利用可能なゲームコンポーネント（「ミドルウェア」）の開発者はおそらくこのルールの例外である。彼らはコードを維持し再利用しているが、彼らの中核事業はゲーム開発ではないため、この慣行はゲームデザインに持ち越されない。[?]

3. 残念ながら、ソフトウェア工学では同じ頭字語がモデル駆動アーキテクチャを示すために広く使われており、混乱を招く可能性がある。本論文ではMDAは常にメカニクス、ダイナミクス、エステティクスのフレームワークを指す。[?]

4. http://www.theinspiracy.com/400_project.htm（2011年6月23日最終閲覧）。[?]

5. プロジェクトにはすべてのエントリーが見つかるアクティブなウェブページがある：http://www.gameontology.com（2011年7月8日最終閲覧）。[?]

6. ここでの議論はすべてのボキャブラリーを取り上げることはできない。言及する価値のあるアプローチの一つは、ゲームプレイのイノベーションのトラッキングに焦点を当てるGame Innovation Databaseで、http://www.gameinnovation.org/（2010年10月22日最終閲覧）に見つかる。[?]

7. パターンはhttp://www.gameplaydesignpatterns.org（2011年7月8日最終閲覧）でオンラインで見つけることもできる。[?]