第6回講義：ゲームの自動生成

第6回：ゲームの自動生成

概要

1. モデル駆動工学のゲームデザインへの応用

   モデル駆動工学（Model Driven Engineering: MDE）

2. 形式文法（Formal Grammars）

   基本要素

   チョムスキー階層

   ミッション記述のための文法

3. 書き換えシステム（Rewrite Systems）

   正規形（Normal Form）

   合流性（Confluence）

   停止性（Termination）

4. グラフ文法（Graph Grammars）

   基本概念

   ミッションからスペースへの変換

5. シェイプ文法（Shape Grammars）

   プリミティブ

   空間レイアウトの生成

6. ロックと鍵の変換例

   6つの書き換え規則

   変換プロセス

7. スペースの生成

8. メカニクスの生成

   Machinationsダイアグラムの書き換え規則

9. プロシージャルコンテンツ生成

   代表的な事例

   アクションアドベンチャーへの応用

10. 適応型ゲーム

Infinite Mario Bros

Infinite Boulder Dash

11. 自動化デザインツール

混合主導型アプローチ（Mixed-Initiative）

Ludoscope ツール

まとめ

第6回：ゲームの自動生成

概要

ゲームデザインの各側面（ミッション、スペース、メカニクス）を**形式的なモデル**として扱い、モデル変換やプロシージャル生成によってゲームコンテンツを自動生成する手法を学ぶ。形式文法、書き換えシステム、グラフ文法などの理論的基盤から、実際のプロシージャルコンテンツ生成や適応型ゲームへの応用までを扱う。

---

1. モデル駆動工学のゲームデザインへの応用

モデル駆動工学（Model Driven Engineering: MDE）

ソフトウェア工学の手法で、**抽象モデルからコードやシステムを自動生成**するアプローチ。

モデル駆動工学の概念図

モデル駆動工学の概念図

- **モデル変換**: あるモデル（例: ミッション構造）から別のモデル（例: 空間レイアウト）を生成する
- ゲームデザインの3つの側面をモデルとして扱う:
  - **ミッションモデル**: プレイヤーが達成すべきタスクの構造
  - **スペースモデル**: ゲーム世界の空間的構成
  - **メカニクスモデル**: ゲームの内部経済とルール（Machinationsダイアグラム）
- これらのモデル間の**変換規則**を定義することで、一貫性のあるゲームコンテンツを自動生成できる

> **重要ポイント:** モデル駆動工学をゲームデザインに適用することで、ミッション→スペース→メカニクスの変換を形式的・自動的に行える。各モデルは独立して設計・検証可能である。

---

2. 形式文法（Formal Grammars）

ゲーム構造の生成に用いる数学的基盤。

基本要素

要素	説明	例
終端記号	それ以上書き換えられない記号	`fight`, `key`, `lock`
非終端記号	書き換え可能な記号	`Mission`, `Challenge`
開始記号	生成の出発点となる非終端記号	`Start`
書き換え規則	非終端記号を別の記号列に変換する規則	`Mission → Challenge lock Mission`

チョムスキー階層

形式文法の表現力による分類体系。

チョムスキー階層

チョムスキー階層

タイプ	名称	特徴	用途例
タイプ3	正規文法	最も制限的。右辺は終端記号＋非終端記号1つ	正規表現、単純なパターン
タイプ2	文脈自由文法	左辺は非終端記号1つ	プログラミング言語の構文解析

タイプ	名称	特徴	用途例
タイプ1	文脈依存文法	周囲の文脈に応じた書き換え	自然言語の一部
タイプ0	無制限文法	制約なし。最も表現力が高い	チューリングマシンと等価

ミッション記述のための文法

```
Start     → Mission
Mission   → fight Boss
Mission   → fight Mission
Mission   → key lock Mission
Boss      → miniboss | finalboss
```

この文法から「fight → key → lock → fight → fight → finalboss」のようなミッション列を生成できる。

> **重要ポイント:** 形式文法を用いることで、ミッション構造の生成ルールを厳密に定義できる。チョムスキー階層はゲーム構造の複雑さに応じた文法の選択指針となる。タイプ2（文脈自由文法）がゲーム生成で最もよく使われる。

---

3. 書き換えシステム（Rewrite Systems）

形式文法を一般化した変換システム。

正規形（Normal Form）

- 書き換えシステムの標準的な表現形式
- 規則の適用方法を統一し、分析を容易にする

合流性（Confluence）

合流性の概念図

合流性の概念図

- **定義:** 異なる順序で規則を適用しても、最終的に**同じ結果に到達**する性質
- 合流性がある書き換えシステムでは、規則の適用順序に依存せず一意な結果が得られる
- ゲーム生成において、合流性は**結果の予測可能性**を保証する

停止性（Termination）

- **定義:** 書き換えが**有限ステップ**で終了する性質
- 停止しないシステムは無限ループに陥る
- ゲーム生成では停止性がなければ生成プロセスが完了しない

> **重要ポイント:** 合流性と停止性は書き換えシステムの健全性を保証する重要な性質である。ゲーム生成に用いる文法は、これらの性質を満たすよう設計すべきである。

---

4. グラフ文法（Graph Grammars）

文字列ではなく**グラフ構造**を書き換える文法。

グラフ文法の書き換え規則例

グラフ文法の書き換え規則例

基本概念

- 書き換え規則の左辺と右辺がともに**グラフ**（ノードとエッジの集合）
- 左辺のパターンにマッチする部分グラフを、右辺のグラフで置換する
- ゲームのミッショングラフやスペースグラフの生成に適用

ミッションからスペースへの変換

1. ミッション構造をグラフとして表現
2. グラフ文法の規則を適用し、各ミッション要素を空間的な構造に変換
3. 結果として、ミッションに対応したゲーム空間のレイアウトが生成される

> **重要ポイント:** グラフ文法は、ゲームのミッション構造やスペース構造のように、線形ではない複雑な関係性を持つ構造の生成と変換に適している。

---

5. シェイプ文法（Shape Grammars）

幾何学的な形状を対象とする文法。

> シェイプ文法による空間生成

シェイプ文法による空間生成

プリミティブ

- **点:** 位置の指定
- **線分:** 壁、通路、境界
- **四角形:** 部屋、エリア
- その他の幾何学的図形

空間レイアウトの生成

- 書き換え規則が幾何学的変換（移動、回転、拡大縮小）を含む
- 建築設計やゲームのレベル生成に応用
- 例: 「四角形 → 四角形を4分割」「線分 → 線分＋ドア」

> **重要ポイント:** シェイプ文法は、グラフ文法がトポロジー（接続関係）を扱うのに対し、幾何学（実際の形状と配置）を扱う。ゲームレベルの物理的なレイアウト生成に有効である。

---

6. ロックと鍵の変換例

ミッション構造からロック＆キーを含む空間構造を自動生成する具体例。

6つの書き換え規則

ロック＆キー変換規則

ロック＆キー変換規則

規則	入力	出力	説明
規則1	直線ミッション	部屋の連鎖	基本的な部屋の連結
規則2	鍵の取得タスク	分岐路＋鍵部屋	鍵を取得するための脇道を生成
規則3	ロックされた進行	ロック付きドア	鍵が必要な扉を配置
規則4	分岐ミッション	分岐する通路	複数経路の空間構造
規則5	ボス戦	大部屋＋入口制限	ボス部屋の生成
規則6	オプション要素	隠し通路	任意探索の空間構造

変換プロセス

1. ミッション文法でミッション構造を生成
2. グラフ文法でミッション→空間の変換規則を適用
3. シェイプ文法で空間→幾何学的レイアウトに変換
4. 最終的な物理的レベルが生成される

> **重要ポイント:** ミッション→スペースの変換は、複数段階の文法変換として形式化できる。6つの基本規則の組み合わせでも、多様なレベル構造を生成可能である。

---

7. スペースの生成

ミッション構造から空間的構成を自動生成する手法。

- ミッショングラフの各ノードを**部屋や区画**に対応付ける
- エッジを**通路やドア**に変換する

- 制約条件（部屋サイズ、通路の長さ、鍵とロックの整合性）を満たすよう配置を最適化
- トポロジー（接続関係）から幾何学（物理配置）への変換が必要

> **重要ポイント:** スペース生成はミッション構造との一貫性を保ちながら、物理的に実現可能なレイアウトを生成する必要がある。トポロジーと幾何学の両方を考慮しなければならない。

---

8. メカニクスの生成

Machinationsダイアグラムの書き換え規則

メカニクス生成の書き換え規則

メカニクス生成の書き換え規則

- Machinationsの基本パターン（エンジン、ドレイン、コンバータなど）を書き換え規則として定義
- ゲームのジャンルや目的に応じた**テンプレート規則**を用意
- 規則の適用により、バランスの取れた内部経済を自動生成
- 生成後にシミュレーションで**バランスを検証**できる

> **重要ポイント:** メカニクスの生成は、Machinationsのパターンライブラリと書き換え規則を組み合わせることで実現できる。生成と検証をサイクルで回すことが実用上重要である。

---

9. プロシージャルコンテンツ生成

アルゴリズムによるゲームコンテンツの自動生成。

代表的な事例

ゲーム	生成対象	手法
Rogue (1980)	ダンジョンレイアウト	ランダム生成の先駆け
Diablo	ダンジョン、アイテム	ランダム＋テンプレート
Torchlight	ダンジョン階層	タイルベースの接続
Minecraft	地形全体	パーリンノイズ＋バイオーム規則

アクションアドベンチャーへの応用

- ロック＆キー構造を含むダンジョンの自動生成
- ミッション文法 → グラフ文法 → シェイプ文法のパイプライン
- ゼルダ風のダンジョンをプロシージャルに生成する研究

> **重要ポイント:** プロシージャルコンテンツ生成は、リプレイ性の向上とコンテンツ制作コストの削減を両立する。ただし、品質制御が課題であり、生成物の検証メカニズムが不可欠である。

---

10. 適応型ゲーム

プレイヤーの行動に基づいてコンテンツを動的に生成・調整するゲーム。

Infinite Mario Bros

Infinite Mario Brosの適応的レベル生成

Infinite Mario Brosの適応的レベル生成

- **プレイヤーモデル**に基づくレベル生成:
  - プレイヤーの行動パターン（ジャンプ頻度、敵との戦闘傾向など）を分析
  - プレイヤーの好みに合ったレベルを動的に生成
- 探索好きなプレイヤーには広い空間を、アクション好きには敵の多いレベルを提供

- 機械学習によるプレイヤーモデリング

Infinite Boulder Dash

- Boulder Dashのルールに基づく無限レベル生成
- パズル要素の難易度をプレイヤーのスキルに適応
- 文法ベースの生成と評価関数の組み合わせ

> **重要ポイント:** 適応型ゲームは、プロシージャル生成にプレイヤーモデルを組み合わせることで、個々のプレイヤーに最適化された体験を提供する。生成アルゴリズムとプレイヤー分析の統合が鍵となる。

---

11. 自動化デザインツール

混合主導型アプローチ（Mixed-Initiative）

- **人間のデザイナー**と**アルゴリズム**が協働するアプローチ
- 完全自動生成ではなく、デザイナーの意図を尊重しつつ自動化で支援
- デザイナーが方向性を決め、アルゴリズムが詳細を埋める

Ludoscope ツール

Ludoscopeの概念図

Ludoscopeの概念図

- ゲームレベルの**混合主導型**生成ツール
- **レシピ（Recipe）**：文法規則の適用順序を定義した自動生成プラン
- **マニュアル編集:** デザイナーが生成結果を手動で調整
- 生成パイプライン:
    1. ミッション文法でミッション構造を生成
    2. グラフ変換でスペース構造に変換
    3. シェイプ文法で幾何学的レイアウトに変換
    4. 各段階でデザイナーが介入・修正可能

・レシピとマニュアル編集を自由に切り替えながらレベルを制作できる

> **重要ポイント:** 混合主導型アプローチは、完全自動生成の品質問題と完全手動設計のコスト問題の両方を解決する。Ludoscopeは、本書で学んだミッション・スペース・メカニクスのモデルを統合的に扱うツールの実例である。

まとめ

手法	入力	出力	特徴
形式文法	書き換え規則	ミッション列	理論的基盤が明確
グラフ文法	グラフ書き換え規則	ミッション/スペースグラフ	非線形構造に対応
シェイプ文法	幾何学的規則	空間レイアウト	物理的配置を生成
プロシージャル生成	アルゴリズム＋パラメータ	ゲームコンテンツ全般	リプレイ性向上
適応型生成	プレイヤーモデル＋生成規則	個別最適化コンテンツ	パーソナライズ
混合主導型	デザイナー入力＋生成規則	高品質レベル	効率と品質の両立

ゲーム生成の各手法は、ゲームデザインの形式的モデル（ミッション、スペース、メカニクス）を基盤としている。モデルを厳密に定義することで、自動生成・変換・検証が可能になる。